**Laura Grindei**

**Claudia Constantinescu**

**Marius Purcar**

# Aplicaţii C/C++/C# şi Arduino în Inginerie Electrică


**U.T. PRESS**

**Cluj-Napoca, 2020**

**ISBN 978-606-737-435-3**

Editura U.T. PRESS
Str. Observatorului nr. 34
C.P. 42, O.P. 2, 400775 Cluj-Napoca
Tel.:0264-401.999
e-mail: utpress@biblio.utcluj.ro
http://biblioteca.utcluj.ro/editura

Director: Ing. Călin D. Câmpean

Recenzia: Prof.dr.ing. Călin Munteanu
Conf.dr.ing.ec. Claudia Păcurar



**ISBN 978-606-737-435-3**

## CUPRINS

# INTRODUCERE

Cartea de faţă prezintă o serie de aplicatii din domeniul Ingineriei Electrice implementate in limbajele C, C++, C# şi Arduino , fiind destinată în principal studenţilor din anul I de la Facultatea de Inginerie Electrica, din Universitatea Tehnica, care studiaza disciplina de Programarea Calculatoarelor si Limbaje de Programare dar si studentilor din anii terminali la nivel de licenta si Master care doresc sa realizeze lucrari de licenta si dizertatie utilizand aceste limbaje de programare. De asemenea , exemplele complete prezentate in această carte vor fi utile celor care vor să realizeze aplicaţii în aceste limbaje facilitând însuşirea atât a noţiunilor de bază cât şi a celor avansate şi formarea unor aptitudini practice de abordare si implementare a problemelor matematice clasice şi a aplicaţiilor tehnice complexe din domeniul ingineriei electrice.

În primele capitole aplicaţiile prezentate sunt generale, fiind utile pentru înţelegerea noţiunilor de programare, pentru însuşirea sintaxei si a abordării limbajelor de programare C/C++, insa treptat aplicatiile propuse sunt orientate spre domeniul Ingineriei Electrice.

Aplicaţiile prezentate in primele unsprezece capitole au fost implementate în ambele variante: C si C++ , ultimele capitole constând din aplicatii implementate in limbajul C# si respectiv Arduino.

Fiecare capitol este structurat similar fiind împărţit in trei sectiuni distincte: Consideratii teoretice, Probleme rezolvate si Probleme propuse. În prima secţiune a fiecarui capitol sunt prezentate în rezumat noţiuni teoretice privind elementele de limbaj utilizate in problemele rezolvate.

Problemele rezolvate reprezintă aplicaţiile concrete în care se aplică noţiunile teoretice prezentate şi includ programele complete implementate în C ,C++, C# sau Arduino precum şi un set reprezentativ de date de intrare/ieşire. Toate problemele rezolvate includ si aplicaţii simple ce pot fi rezolvate pe baza problemelor rezolvate. Secţiunea de probleme propuse constă din aplicaţii mai complexe propuse spre rezolvare.

Aplicaţiile implementate în limbajele C/C++ incluse în secţiunea Probleme rezolvate au fost realizate în mediul de programare CodeBlocks care este disponibil gratuit pe Internet: http://www.codeblocks.org/ iar cele implementate în C#, utilizând Microsoft Visual Studio Community disponibil pentru download la adresa: http://visualstudio.microsoft.com/vs/community

Mediul de programare Codeblocks ofera un cadru gratuit, complex şi performant de editare, compilare, şi executie al programelor scrise în limbajele C/C++ şi pune la dispoziţia utilizatorilor un suport (help) consistent, un compilator performant care permite corectarea rapidă a erorilor şi o serie de alte opţiuni utile pentru implementarea practică a aplicaţiilor.

Primul capitol al cărtii prezintă noţiunile de bază privind abordarea problemelor de programare, descrierea tipurilor de date specifice limbajelor C/C++, prezentarea sintaxei diferitelor tipuri de instrucţiuni, prezentarea bibliotecilor şi funcţiilor predefinite, descrierea modului de declarare şi a principalelor operaţii specifice tablourilor.

Capitolele 2 - 5 prezintă noţiunile de bază referitoare la pointeri, operaţii specifice acestora, pointeri la tablouri , tablouri de pointeri , pointeri la funcţii, tablouri de pointeri la funcţii, etc. iar capitolul 6 tratează alocarea dinamică a memoriei in C/C utilizand funcţii specifice de alocare dinamică si respectiv in C++ utilizand operatorii new si delete.

In capitolul 8 sunt prezentate argumentele funcţiei main() şi modul de lansare a programelor din linia de comandă, in timp ce urmatoarele doua capitole prezintă modul de definire a tipului de date structurat, declararea şi sortarea tablourilor de structuri, definirea uniunilor, enumerărilor şi listelor.

Capitolul 9 include modul de definire şi utilizare al fişierelor, iar capitolul 10 ilustrează utilizarea stream-urilor cin şi cout in limbajul C++. În capitolul 11 este prezentat modul de structurare al programelor C/C++ în mai multe module (fisiere) şi utilizarea fişierelor antet. Capitolul 12 , prezintă noţiunile introductive privind programarea orientată pe obiecte,principalele diferenţe dintre limbajele C/C++.

Capitolele 13 şi 14 includ exemple de programare a aplicatiilor Windows în limbajul C# , utilizând Microsoft Visual Studio Community Edition, iar ultimul capitol prezinta câteva aplicaţii simple realizate cu plăcuţe Arduino si implementate în mediul de programare Arduino IDE.

Ţinând cont de faptul că limbajul de programare C/C++ a devenit un standard, iar noile limbaje de programare sau de scripting care se impun pe piaţa dezvoltatorilor de software au la baza sintaxa C/C++ (exemple : C#, Java, Javascript, ActionScript, PHP, Pyton, etc.) studiul acestei cărţi poate fi un punct de plecare şi pentru insuşirea altor limbaje înrudite.

Cluj-Napoca, februarie 2020,
Autorii

# Capitolul 1
# Noţiuni de bază ale limbajelor C/C++

C este un limbaj de programare standardizat, compilat, de nivel mediu. Este implementat pe marea majoritate a platformelor de calcul existente azi, şi este cel mai popular limbaj de programare pentru scrierea de software de sistem. Este apreciat pentru eficienţa codului obiect pe care îl poate genera, şi pentru portabilitatea sa. [L1]. Sintaxa limbajului C a stat la baza multor limbaje create ulterior şi foarte populare in prezent: C++, Java, JavaScript, C#, Python, etc.

Desi limbajul de programare C++ a fost iniţial derivat din C, totuşi, nu orice program scris în C este valid C++. Deoarece C şi C++ au evoluat independent, au apărut, o serie de incompatibilităţi între cele două limbaje de programare. În capitolele următoare se vor specifica diferentele de sintaxă care apar între cele două limbaje.

## 1. Structura generală unui program în C/C++

In general orice program implementat în C include următoarele secţiuni:

```
//comentariu general privind scopul programului
SECTIUNEA PREPROCESOR
#include <... >
#include <... >
…..

DIRECTIVE GLOBALE PENTRU ÎNTREGUL PROGRAM

SECTIUNEA FUNCŢIEI MAIN()
int main()
{
  Declaratii variabile şi constante locale funcţiei main()
  SECTIUNEA DE INSTRUCTIUNI A PROGRAMULUI
return 0;
}

SECTIUNEA FUNCŢIILOR UTILIZATOR
```

Secţiunile funcţiei main() si a functiilor utilizator pot fi inversate.

## 2. Tipuri de date în C/C++

În **Tabelul 1.1** sunt prezentate tipurile de bază în C/C++. Pentru fiecare tip de date este specificat cuvântul cheie ce va fi utilizat pentru declarare, domeniul de valori şi numărul de octeţi ce îi vor fi alocaţi.

**Comentariile** în C/C++ se introduc fie utilizând caracterele // fie caracterele /* */.

**Variabilele** sunt utilizate pentru memorarea unor valori. Deoarece diferite tipuri de variabile sunt folosite să memoreze diferite tipuri de date, tipul fiecărei variabile trebuie specificat în program

printr-o declaraţie. În C/C++, toate variabilele trebuie declarate înainte de a fi utilizate în expresii şi instrucţiuni.

**Tabelul 1.1**

| Tip variabilă C | Nr. octeţi (bytes) | Limita inferioară | Limită superioară | Utilizare |
|---|---|---|---|---|
| char | 1 | -128 | 127 | Caractere sau numere |
| unsigned char | 1 | 0 | 255 | Numere mici pozitive |
| short int | 2 | -32768 | +32767 | Numere întregi |
| unsigned short int | 2 | 0 | 65536 | Numere întregi pozitive |
| (long) int | 4 | -2^31 | +2^31-1 | Numere întregi foarte mari |
| float | 4 | -3.2*10^38 | 3.2*10^38 | Numere reale |
| double | 8 | -1.7*10^308 | 1.7*10^308 | Numere reale foarte mari |
| void | 0 | - | - | Nici un tip |

Un **nume de variabilă**, numit şi **identificator**, constă dintr-o secvenţă de unul sau mai multe caractere (litere, cifre , fără a începe cu cifră sau liniuţă de subliniere). Cuvintele cheie, numite şi cuvinte rezervate, nu pot fi utilizate ca nume de variabile. Exemple de cuvinte cheie: char, int, float, if,while, etc.

**Formatul de declarare al unei variabile**: **<tip de date><identificator>=<initializator optional>;**
unde:

- **tip de date** este unul din tipurile acceptate de C/C++ (standard, pointer, structurat)
- **identificator** trebuie să respecte următoarele condiţii:
  - nu poate începe cu o cifră
  - nu poate conţine spaţii, şi caractere speciale: .,;'"
  - nu pot fi identice cu cuvintele cheie C/C++ şi nici cu numele funcţiilor din bibliotecile C/C++

**Formatul de declarare al unei constante**: **const <tip de date><identificator>=<constanta>;**
sau utilizând directiva preprocesor: **#define identificator constanta**

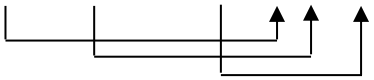
Funcţiile de I/O (intrare/ieşire) utilizate în C/C++ sunt **printf()** şi **scanf().** Prototipurile lor se gasesc în biblioteca <stdio.h>. În C++ se utilizează cin si cout care vor fi detaliate în capitolul 10.

**Formatul instrucţiunii de afişare/tipărire: printf (control,arg1,arg2,…)**
unde:

- **control**="(sir_caractere)(%)(caracter_conversie)(caracter_negrafic)"
- **arg1, arg2,...** sunt expresii sau variabile ce vor fi tipărite

**Ex.**:

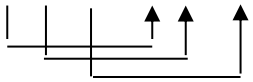
```
int a=250,b=2;float rez;rez=a/b;
printf("a=%d, b=%d, rezultat=%f", a,  b,  rez);
```



**Formatul instrucţiunii de citire**: **scanf (control,&arg1,&arg2,…)**
unde:

- **control**="(sir_caractere)(%)(caracter_conversie)(caracter_negrafic)"
- **arg1, arg2,...** sunt argumente care corespund adreselor zonelor de memorie în care se păstreaza datele citite după ce au fost convertite la formatul specificat de caracterul de control.

**Ex.**:
int a,b;float var;
scanf("%d, %d, %f",&a,&b,&var);



În **Tabelul 1.2** sunt prezentate caracterele de conversie asociate tipurilor de date de bază: caracter, şir de caractere, întreg, real, etc.

**Tabelul 1.2**

| Tip de date | Tip variabilă | Caracter de conversie utilizat în printf()/scanf() |
|---|---|---|
| caracter | char | **%c** |
| şir de caractere | char | **%s** |
| întreg | int | **%d** |
| | unsigned | **%u** |
| întreg cu semn | int | **%i** |
| întreg fara semn în baza 16 | int/unsigned | **%x** |
| întreg fara semn în baza 8 | int/unsigned | **%o** |
| întreg | long | **%ld, %li, %lu, %lx,%lo** |
| real | float | **%f** |
| | double | **%lf** |
| real în format exponenţial | float/double | **%e sau %E sau %g sau %G** |
| real | double | **%lf, %le, lg** |

**Operatorii aritmetici** sunt prezentaţi în **Tabelul 1.3**:

**Tabelul 1.3**

| Operator | Simbol |
|---|---|
| Adunare | **+** |
| Scadere | **-** |
| Inmultire | ***** |
| Impartire | **/** |
| Modulo | **%** |
| Incrementare | **++** |
| Decrementare | **--** |

**Operatorii +,-,*,/** sunt utilizaţi în C/C++ cu aceeaşi semnificaţie ca şi în celelalte limbaje de programare, reprezentând operaţiile aritmetice de bază.

**Operatorul %** este utilizat pentru a obţine restul împărţirii a două numere întregi şi nu poate fi utilizat cu numere în virgulă mobilă.

**Ex:**
dacă x=5; y=2 atunci
x/y =2 (câtul împărţirii întregi)
x%y=1 (restul împărţirii întregi)

**Operatorii de incrementare: ++ şi decrementare --** adună şi respectiv scad 1 la/din variabila operată. Astfel operaţia x++ este echivalentă cu operaţia de atribuire: x=x+1. Notaţiile x++ şi ++x sunt echivalente dacă nu sunt utilizate într-o expresie. În exemplul de mai jos se pot constata diferenţele dintre diversele moduri de utilizare ale operatorilor ++ şi – în expresii.

**Ex:**
dacă x=10 atunci după evaluarea expresiei:

- y=x++ □ y=10, x=11

y este iniţializat cu valoarea lui x înainte de incrementare, abia apoi se incrementează x .

- y=++x □ y=11, x=11

y este iniţializat cu valoarea lui x după incrementare .

- y=x-- □ y=10, x=9

y este iniţializat cu valoarea lui x înainte de decrementare .

- y=--x □ y=9, x=9

y este iniţializat cu valoarea lui x după decrementare.

**Operatorii booleeni** se clasifică în operatori: relaţionali şi logici.

**Operatorii relaţionali** sunt prezentaţi în **Tabelul 1.4**:

**Tabelul 1.4**

| Simbol matematic | Operator în C/C++ | Semnificaţie |
|---|---|---|
| = | == | egal |
| ≠ | != | diferit |
| < | < | mai mic decât |
| ≤ | <= | mai mic sau egal cu |
| > | > | mai mare decât |
| ≥ | >= | mai mare sau egal cu |

Petru testarea egalităţii dintre 2 variabile sau expresii se utilizează **operatorul ==** . Operatorul **=** este operatorul de atribuire.

**Operatorii logici** sunt prezentaţi în **Tabelul 1.5**:

**Tabelul 1.5**

| Operator | Semnificaţie |
|---|---|
| && | Si (AND) |
| \|\| | Sau (OR) |
| ! | Negat (NOT) |

**Tabelul 1.6**

| *p* | *q* | *p&&q* | *p\|\|q* | *!p* |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Tabela de adevăr a operatorilor logici prezentată în **Tabelul 1.6** utilizează valorile 0 şi 1, unde 0 corespunde valorii logice fals iar 1 corespunde valorii logice adevărat. În C/C++ orice valoare diferită de 0 corespunde valorii logice adevărat (true), iar valoarea egală cu 0 corespunde valorii logice fals (false).
În C/C++ nu există operatorul logic sau exclusiv (XOR), dar se poate creea o funcţie cu acest efect.

**Operatorii de atribuire** se găsesc în **Tabelul 1.7** şi vor fi utilizaţi în special în cadrul instrucţiunilor repetitive:

**Tabelul 1.7**

| Operator | Semnificatie |
|---|---|
| = | Atribuire simpla |
| += | Atribuire şi adunare |
| -= | Atribuire şi scadere |
| *= | Atribuire şi inmultire |
| /= | Atribuire şi impartire |
| %= | Atribuire şi modulo |

**Operatorul condiţional ?** are următorul format de utilizare: expresie1 **?** expresie2**:** expresie3**;**

Se evaluează prima dată expresie1. Dacă expresie1 $\neq 0$ (adică relaţia este adevărată sau TRUE) atunci rezultatul = expresie2, altfel rezultatul = expresie3.
**Ex:**
Se consideră x,y,z de tip int. Se cere să se specifice care este efectul evaluării expresiei conditionale: z=(x= = y ? x : 1); Se evaluează mai întâi expresia x==y. Dacă aceasta este adevărată atunci z=x altfel z=1.

**Operatorul sizeof** se utilizează cu formatul: **sizeof(**operand**)** şi returnează dimensiunea în octeţi a operandului.

**Operatorii () şi []** sunt utilizaţi pentru a mări prioritatea operaţiilor din interiorul lor. Parantezele pătrate se utilizează la tablouri.

**Operatorul virgulă,** asigură înşiruirea mai multor expresii.

**Ex:**
Se consideră x,y de tip int. Se cere să se specifice care este rezultatul evaluării expresiei: x=(y=2,y+5);

Astfel, se execută pe rând instrucţiunile de atribuire: y=2 şi x=y+5 (deci x=7).

**Operatorul punct . şi operatorul săgeată ->** sunt utilizaţi la datele de tip structuri şi uniuni.
**Operatori cu acţiune pe biţi** sunt prezentaţi în Tabel 1.8. In Tabelul 1.9 este prezentată tabela de adevăr pentru sau exclusiv (XOR):

**Tabelul 1.8**

| Operator | Semnificaţie |
|---|---|
| **&** | **si** (AND) |
| **\|** | **sau** (OR) |
| **^** | **sau exclusiv** (XOR) |
| **~** | **complement fata de 1** (NOT) |
| **>>** | **deplasare la dreapta**, Format: variabila >> nr.pozitii |
| **<<** | **deplasare la stanga**, Format: variabila << nr.pozitii |

**Tabelul 1.9**

| p | q | p^q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Precedenţa tuturor operatorilor** în C/C++ este prezentată în Fig.1.1:

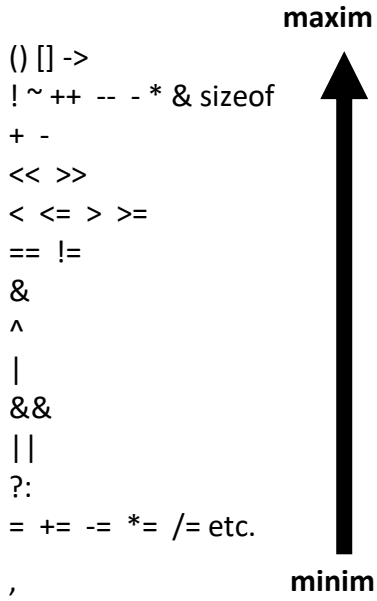



**Fig.1.1.** Precedenţa operatorilor în C/C++

### 3. Instructiuni C/C++

Clasificarea instrucţiunilor în C/C++:

- **Instrucţiune expresie**: instrucţiune de atribuire, apel de funcţie, etc.
- **Instrucţiune secvenţială**: sau instrucţiune compusă care grupează o secvenţă de instrucţiuni între 2 acolade.
- **Instrucţiuni condiţionale**: alternative (IF, IF/ELSE, IF imbricate) sau selective (SWITCH).
- **Instrucţiuni repetitive**: instrucţiuni condiţionate anterior (WHILE, FOR), instrucţiuni condiţionate posterior (DO WHILE)
- **Alte tipuri**: instrucţiune exit(), CONTINUE, BREAK, return, etc.

**Instrucţiunile condiţionale** IF, WHILE , DO WHILE sunt foarte utile în rezolvarea problemelor.

Instrucţiunea **IF** este o instrucţiune condiţională care permite efectuarea de teste asupra unor variabile şi expresii. Se utilizează atunci când este necesar să se execute o instrucţiune sau o secvenţă de instrucţiuni numai dacă o anumită condiţie este adevărată. Dacă condiţia testată este falsă nu se execută nici o instrucţiune. Schema logică din Fig.1.2, ilustrează modul de funcţionare al instrucţiuni IF.

**Formatul de utilizare al intrucţiunii IF** este:
**if**(*expresie*)
{ *instructiune*;
...}
Dacă *expresie* este adevărată se execută *instrucţiune* (care poate fi şi o secvenţă de instrucţiuni).

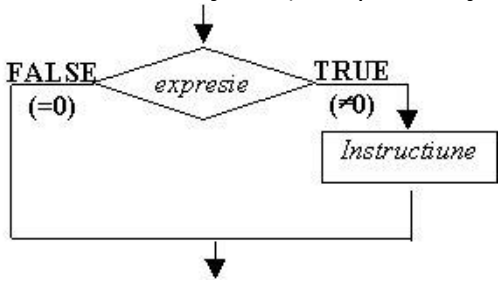



**Fig.1.2.** Schema logică asociată instrucţiunii IF

Instrucţiunea **IF/ELSE** este o generalizare a instrucţiunii condiţionale IF, care se utilizează atunci când trebuie executăte instrucţiuni în ambele situaţii, şi pentru expresie adevărată şi pentru expresie falsă. Schema logică din Fig. 1.3 ilustrează modul de funcţionare al instrucţiunii IF/ ELSE.

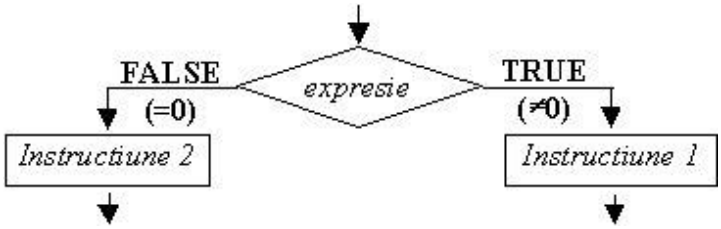



**Fig. 1.3.** Schema logică asociată instrucţiunii IF/ELSE

**Formatul de utilizare al instrucţiunii IF/ELSE** este:
**if**(*expresie*)
{ *Instructiune 1*;
…}
**else**
{ *Instructiune 2*;
…}
**Instrucţiunile IF imbricate** sunt compuse din instrucţiuni IF şi IF/ELSE , incluse una în interiorul alteia. Un astfel de exemplu este prezentat mai jos:

**Ex:**
```
if(expresie1)
{  if(expresie2 ) Instr 1;
   else Instr.2;}
else if(expresie3) Instr.3;
```

Schema logică din Fig. 1.6 ilustrează modul de funcţionare al instrucţiunilor IF imbricate conform exemplului prezentat mai sus:

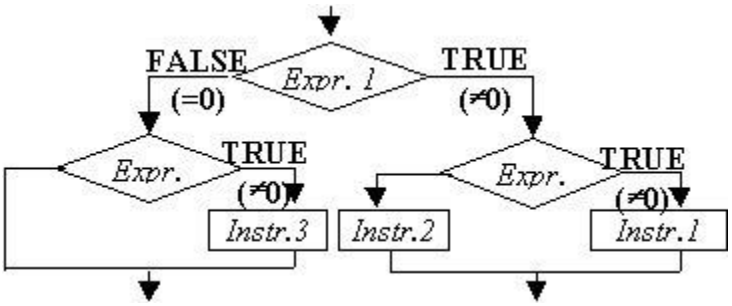



**Fig. 1.6.** Schema logică asociată exemplului cu instrucţiunilor IF imbricate

Instrucţiunea condiţională **SWITCH** este o instrucţiune condiţională selectivă şi are următorul **format**:
**switch**(expresie)
{ **case** const1: instructiune 1; break;
**case** const2: instructiune 1'; break; …
**default**: instructiune 1'';}

Instrucţiunea **FOR** este o instrucţiune repetitivă condiţionată anterior, şi în acest caz condiţia se testează înaintea efectuării instrucţiunii (sau secvenţei de instrucţiuni) din buclă.

**Instructiunile repetitive** în C, C++ sunt **WHILE, DO WHILE** si **FOR .** Instrucţiunea **WHILE** este o instrucţiune repetitivă condiţionată anterior, şi în acest caz condiţia se testează înaintea efectuării instrucţiunii sau secvenţei de instrucţiuni din buclă.

**Formatul instrucţiunii WHILE** este:
**while** (expresie)
{ instructiune 1;
… }
Schema logică asociată instrucţiunii WHILE este prezentată în Fig.1.4.

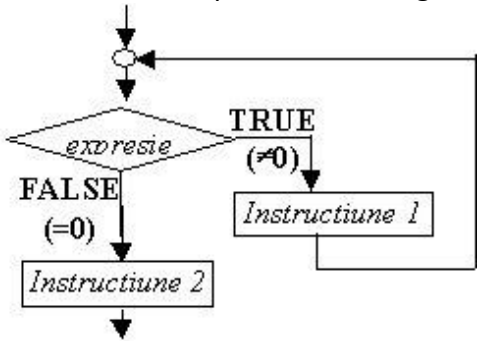



**Fig. 1.4.** Schema logică pentru instrucţiunea WHILE

Instrucţiunea **DO WHILE** este o instrucţiune repetitivă condiţionată posterior, şi în acest caz condiţia se testează după efectuarea instrucţiunii sau instrucţiunilor din buclă.

**Formatul instrucţiunii WHILE** este:
**do**
{ instructiune 1
…}
**while** (expresie)
Schema logică asociată instrucţiunii DO WHILE este prezentată în Fig.1.5.

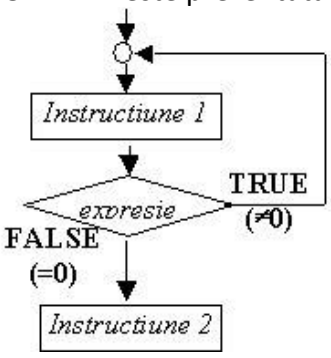



**Fig. 1.5.** Schema logică pentru instrucţiunea DO WHILE

Se execută *Instructiune 1* atâta timp cât *expresie* este adevărată, altfel se execută *Instructiune 2.* Bucla se parcurge cel puţin o dată indiferent dacă *expresie* este sau nu adevărată.Dacă *expresie* este adevărată se execută *Instrucţiune 1* iar Dacă este falsă se execută *Instrucţiune 2*.

**Formatul instrucţiunii FOR** este:
**for**(contor=val initiala; expresie test contor;++/-- contor)
{ instructiuni
… }

Instrucţiunea **for** conţine trei părţi distincte separate prin ";" astfel:

- o instrucţiune de atribuire pentru iniţializarea contorului buclei,
- o expresie relatională ce va fi testată şi care reprezintă condiţia de abandonare a buclei şi
- o instrucţiune de incrementare/decrementare care specifică modul în care se modifică variabila de control a buclei (contorul) la fiecare pas de parcurgere a buclei.

Schema logică în care este ilustrat modul de funcţionare al instrucţiunii FOR este prezentată în Fig. 1.7. Se iniţializează contorul buclei şi se evaluează expresia de test asupra contorului. Dacă această expresie este adevărată atunci se execută Instructiune 1 (sau secvenţa de instructiuni) şi incrementarea contorului, altfel se iese din buclă.

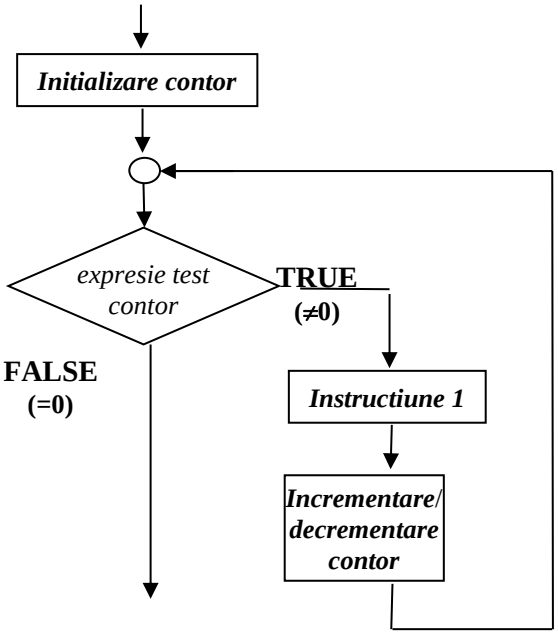



**Fig. 1.7.** Schema logică pentru instrucţiunea FOR

Instrucţiunea **BREAK** se utilizează în interiorul unei bucle şi are rolul de a termina execuţia buclei pentru o anumită condiţie. Execuţia instrucţiunilor din afara buclei respective se reia de la pasul următor.

**Formatul instrucţiunii BREAK** este**:**

```
while(expresie1)
 { Instructiune 1
   …
   if (expresie 2) break;
   Instructiune 2 }
Instructiune 3;
```

Instrucţiunea **CONTINUE** se utilizează în interiorul unei bucle, în mod analog cu instrucţiunea BREAK. Dacă bucla este de tip WHILE, DO WHILE, instrucţiunea CONTINUE are ca scop abandonarea iteraţiei curente şi evaluarea expresiei care stabileşte continuarea buclei. Dacă bucla este de tip FOR, se abandonează iteraţia curentă şi se trece la execuţia pasului de reiniţializare.

**Formatul instrucţiunii CONTINUE** este**:**

```
for(contor=val initiala; expresie test contor; ++/-- contor)
 {  Instructiune 1
```

```
    …
    if(expresie) continue;
    Intructiune 2 }
Instructiune 3;
```

Instrucţiunea **GOTO** se utilizează pentru întreruperea execu'iei secvenţiale a instrucţiunilor unui program, realizând trecerea forţată la o altă instrucţiune marcată de o etichetă.

**Formatul instrucţiunii GOTO** este:

```
…
        eticheta:
……
goto eticheta;
```

**4. Functii**

**Structura funcţiilor** în C/C++ este următoarea:

```
SECTIUNEA HEADER A FUNCŢIEI
<tip de date returnat><numele funcţiei>(<lista parametrii>)
{
SECTIUNEA DECLARATII VARIABILE şi CONSTANTE LOCALE
      <definire variabile şi constante locale>
SECTIUNEA DE INSTRUCTIUNI (CORPUL FUNCŢIEI)
       instructiune 1;
       . . .
       return(<valoare returnata>);
}
```

**Headerul funcţiei** are următorul format :

**<tip de date returnat> <numele funcţiei> (<lista parametri>)**

unde:

- **tipul de date returnat** de funcţie: este unul din tipurile de bază, spre ex. int, float, double, char, etc .
- **numele funcţiei:** un caracter sau şir de caractere respectând regulile C, astfel încât:
  - numele funcţiei nu poate fi utilizat în interiorul funcţiei,
  - identificatorul (numele) de funcţie nu poate fi folosit în expresie de atribuire.
- lista de parametri: conţine variabile ce sunt transmise de programul apelant şi vor fi evaluate de către funcţie.

**Secţiunea de declaraţii** a variabilelor şi constantelor locale urmează după definirea headerului funcţiei. Secţiunea de instrucţiuni poate cuprinde orice tip de instrucţiune şi urmează după declararea variabilelor şi constantelor locale.

**Prototipul unei funcţii** trebuie declarat la începutul programului după secţiunea preprocesor şi înainte de definirea oricărei alte funcţii şi are următorul format:

**<tip date returnat><nume funcţie>(<tip p1,tip p2,…,tip pN>)**

Dacă nu se declară explicit **tip date returnat** de funcţie, i se atribuie automat tipul int. Utilizarea parametrilor **p1, p2**, ... este opţională, totuşi se recomandă specificarea lor.

Dacă numărul de parametri ai funcţiei este variabil atunci prototipul funcţiei se defineşte după exemplul: funcţie (int a, int b, …); este incorect formatul: funcţie( …); Specificarea tipului parametrilor este impusă în C++, deşi se admite încă şi formatul clasic de definire al unei funcţii.

O funcţie poate fi apelată astfel:
- în orice punct al programului principal main(), de ex. printr-o expresie de atribuire, sau
- ca parte a unei expresii aritmetice sau relaţionale.

**Parametri formali** sunt variabilele utilizate în header-ul funcţiei. **Parametrii actuali** sunt variabilele utilizate în apelul funcţiei şi trebuie definiţi în programul apelant.

Tipul de date corespunzător parametrilor formali şi actuali trebuie să fie acelaşi, iar numărul de parametri formali trebuie să fie egal cu numărul parametrilor actuali. Corespondenţa parametrilor actuali şi formali se face pe baza poziţiei lor în listă.

**Transmiterea parametrilor** se poate realiza în două moduri:
- **prin valoare**: un parametru valoare asigură doar un singur sens de comunicare a datelor: de la programul apelant la funcţie;
- **prin referinţă**: un parametru referinţă asigură transmiterea datelor în ambele sensuri între programul apelant şi funcţie (se adaugă caracterul “&” în faţa parametrilor formali).

**Variabilele locale** sunt variabilele care sunt declarate în interiorul unei funcţii şi sunt cunoscute numai în interiorul acelei funcţii (între acolade).

**Variabile globale** sunt variabilele care sunt declarate în afara oricarei funcţii din program, sunt cunoscute în întreg programul şi pot fi utilizate în orice punct al programului.

**5. Tablouri**

Prin definiţie un tablou reprezintă o colecţie de variabile de acelaşi tip apelate cu acelaşi nume, accesul la elemente realizându-se prin indici. Un tablou unidimensional este o structură de date care este caracterizată prin următoarele componente [6],[7]:
- elemente: sunt date de acelaşi tip memorate în tablou; pot fi de tip întreg, float, caracter, enumerare sau alte tablouri de acelaşi tip.
- dimensiunea tabloului (notaţie: n) ;
- indicii: primul indice asociat este 0, ultimul este n-1

Formatul de declarare al unui tablou unidimensional este:
**<tip date elemente tablou><nume tablou>[<nr.elem.tablou>];**
unde :
- **tip date elemente tablou:** este tipul elementelor tabloului (ex. int, float, double , char, etc)
- **nume tablou** este numele variabilei tablou
- **[<nr.elem.tablou>]** este dimensiunea maximă a tabloului.

Un şir de caractere este format din unul sau mai multe caractere, din care ultimul este ‘\0’.

Principalele operaţii asociate şirurilor de caractere sunt: comparare, con-catenare, copiere. Câteva dintre cele mai importante funcţii din biblioteca <string.h> asociate şirurilor de caractere sunt prezentate în Tabelul 1.10:

**Tabelul 1.10**

| **Funcţie** | **Efect** |
|---|---|
| strcpy(s1,s2) | copiază s1 în s2 |
| strcat(s1,s2) | concatenează s2 la sfirsitul lui s1 |
| strlen(s1) | returneaza lungimea lui s1 |
| strcmp(s1,s2) | comparare şiruri; returnează 0 dacă s1 şi s2 sunt identice, <0 dacă s1<s2, >0 dacă s1>s2 |
| strchr(s1,ch) | Se caută caracterul ch în şirul s1; funcţia returnează un pointer la prima apariţie a lui ch în s1 |
| strstr(s1,s2) | Se caută un şir s2 în şirul s1; funcţia returneaza un pointer la prima apariţie a lui s2 în s1 |

În continuare sunt prezentate operaţiile de bază asupra elementelor tablourilor unidimensionale: iniţializarea, inserarea şi extragerea lor .

**Iniţializarea elementelor** prin atribuire directă are formatul:
**<tip date tablou><nume tablou>[nr.elem]=valori initiale**

Iniţializarea elementelor prin citire sau prin atribuire cu buclă for, while sau do/while se realizează în mod similar.

**Ex:** bucla cu instrucţiunea for:
for (indice=0;indice<nr.elem.tablou; indice++)
{<initializarea elem.tablou[indice] prin atribuire/citire>}

**Inserarea elementelor în tablou** prin atribuire directă are formatul:
**<nume tablou>[indice tablou]= valoare element;**

Inserarea elementelor prin citire sau prin atribuire cu buclă for, while, do/while sunt similare.

**Ex:** bucla cu instrucţiunea for :
for (indice=0;indice<nr.elem.tablou; indice++)
{<inserare elem.tablou[indice] prin atribuire sau citire>}

**Extragerea elementelor** prin atribuire directă are formatul:
**<identificator variabila>=<nume tablou>[indice tablou];**

**Sortarea tablourilor**

O altă operaţie de bază este **sortarea** elementelor tabloulurilor, şi constă în ordonarea crescătoare sau descrescătoare a elementelor utilizând diferite metode.

Există mai multe metode de sortare:

1. **Sortarea prin interschimbare** care consta în comparări şi schimbări succesive ale elementelor tabloului pâna când acestea se află în ordinea dorită. Din această categorie fac parte:
   - **Metoda Bubble Sort**: se mai numeşte “metoda bulelor" şi este o tehnică de sortare prin interschimbare. Denumirea îi provine din faptul că elementele (bule) se deplasează spre dreapta (valorile mari) sau spre stânga (valorile mici), la fel cum bulele se ridică la suprafaţa apei. Este cea mai simplă şi cea mai des utilizată metodă de sortare însă utilizează mai multă memorie şi timp pentru efectuarea sortării. In sectiunea problemelor rezolvate sunt implementate mai exemple realizate cu aceasta metodă.
   - **Metoda Quick Sort (sortare rapidă)**: este una dintre cele mai rapide metode de sortare. Se consideră tabloul ca o listă de valori. Când începe sortarea, se selectează valoarea de mijloc a listei ca separator de listă. Se împarte lista în două, una cu valorile mai mici decât separatorul de listă şi a doua cu valorile mai mari sau egale cu acesta. Sortarea se invocă apoi recursiv în ambele liste. La fiecare invocare ea împarte elementele în liste mai mici.

2. **Sortarea prin selecţie** care se bazează pe următorul algoritm: se determină din tablou elementul minim care se plasează pe prima poziţie în tabloul sortat. Apoi, se găseşte următorul minim din elementele rămase, care se depune pe a doua poziţie, etc. până când tabloul este sortat integral.

3. **Sortarea prin inserţie** care se bazează pe următorul algoritm: pornind de la al doilea element al tabloului, fiecare element găsit este inserat în subşirul din stânga indexului i, astfel ca acest subşir să fie ordonat crescător. Din această categorie face parte:
   - **Metoda Shell:** se compară elementele şirului separate printr-o distanţă specifică (gap) până când elementele din interval sunt ordonate. Se împarte intervalul în două şi se continuă procesul până când intervalul devine 1 şi nu mai apare nici o modificare la parcurgerea elementelor tabloului.

Alte tipuri de sortare sunt **sortarea prin interclasare**, **sortarea prin numărare**, etc.

**Formatul de declarare a unui tablou bidimensional** (matrice) este :
**<tip tablou><nume tablou>[<nr.max.el.lin>][<nr.max.el.col>];**
unde:
- **nr.max.el.lin** reprezintă numărul maxim de elemente pe linie
- **nr.max.el.col** reprezintă numărul maxim de elemente pe coloană

Iniţializarea elementelor unei matrici prin atribuire directă se poate realiza după următorul format:
<tip tablou><nume tablou>[nr.max.linie][nr.max.col] = valori iniţiale

Initializarea elementelor unei matrici prin citire sau atribuire cu buclă for (analog cu bucla while, do/while) se poate realiza după următorul format:

```
for (i=0;i<m; i++)
  for (j=0;j<n; j++)
    <initializarea elem.tablou[i][j] prin atribuire sau citire>
```

**Extragerea elementelor dintr-o matrice** prin atribuire directă se poate realiza după următorul format:
**<variabila>=<nume tablou>[indice linie][indice coloana]**
unde *variabila* reprezintă o variabilă declarată în program căreia i se va atribui valoarea elementului cu indicele specificat.

# Capitolul 2
# Pointeri . Operaţii cu pointeri

În acest capitol sunt prezentate consideraţii teoretice privind definirea şi utilizarea pointerilor fiind prezentate câteva probleme rezolvate cu pointeri la date de tip întreg, la tipul caracter şi la şir de caractere.

## CONSIDERAŢII TEORETICE

Variabila de tip **pointer** este o variabilă care are ca şi valoare o adresă de memorie.

Pointerii se clasifică astfel:
- **Pointer la date** = conţine adresa unei variabile sau a unei constante
- **Pointer la funcţii** = conţine adresa codului executabil al unei funcţii
- **Pointer la obiecte** = conţine adresa unui obiect în memorie = adrese de date şi funcţii

Pointerii conţin adrese de memorie şi nu valori ca alte tipuri de variabile sau constante.

Dacă la o adresă de memorie se află altă adresă acest lucru are semnificaţia de indirectare ("pointare") (o variabilă indică spre alta ).

În Fig.2.1 este prezentat un exemplu de adrese de memorie în care sunt memorate date sau alte adrese ale altor date .

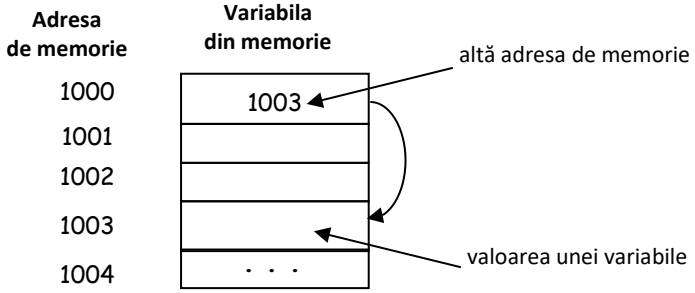



**Fig.2.1.** Reprezentarea variabilelor în memorie

**Formatul de declarare** al unei variabile de tip **pointer** este : **tip *nume ;**

unde **tip**= tipul de baza al pointerului, defineşte tipul variabilei la care indică acesta;
poate fi orice tip de variabilă;
**nume**= numele variabilei pointer

**Operatorii specifici pointerilor** sunt : **&, *.**

**Operatorul de adresare (referentiere): & ("adresa lui")** este un operator unar care asociat unei variabile sau obiect, returnează adresa de memorie a acelei variabile sau obiect.

**Operatorul de indirectare (dereferentiere):* ("de la adresa")** este un operator unar, complementar lui &, returnează valoarea înregistrata la adresa de memorie specificată.

**Observatii**:

- Operatorul * nu trebuie confundat cu operatorul aritmetic utilizat pentru înmulţire (*).
- Operatorul & nu trebuie confundat cu operatorul AND pentru biţi (&) .
- Operatorii * şi & au prioritate faţă de toţi operatorii aritmetici (cu exceptia lui –unar cu care au aceeasi precedenţă).

După declararea pointerului trebuie specificat către ce anume indică acesta (o alta variabila de tip data sau un alt pointer) deoarece implicit nu indică nimic.

**Ex.1:** Instrucţiunile din paragraful INCORECT vor genera eroare , deoarece variabila de tip pointer nu a fost iniţializată.

**INCORECT:**

```
int *ip;
*ip = 100;
```

**CORECT**:

```
int *ip,  x;
ip = &x;
*ip = 100;
```

**Ex.2:** Un alt exemplu de declarare şi iniţializare de pointer

- Prin 2 instrucţiuni distincte:

```
int a=1, *ptoa;
ptoa = &a;       //pointerului ptoa i se atribuie adresa variabilei  a
```

- Printr-o singură instrucţiune:

```
int a=1, *ptoa= &a;  //pointerului ptoa i se atribuie adresa variabilei  a
```

Pentru exemplul anterior în Fig.2.2. este reprezentat modul de definire a pointerului *ptoa într-o zonă de memorie. Adresele de memorie sunt reprezentate prin numere în baza 16 (0x00,…, 0xFF).

Pointerul se numeste *ptoa, este iniţializat cu adresa &a, iar locaţia  de memorie care este rezervată lui *ptoa va conţine numărul 1, adică valoarea variabilei a.

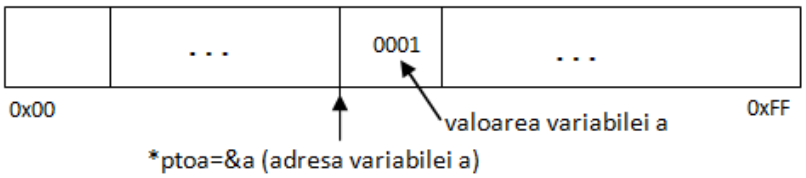



**Fig.2.2.** Reprezentarea unui pointer în memorie prin adresă şi conţinut

**Operaţiile nepermise** asupra pointerilor sunt:

- Adunarea a 2 sau mai mulţi pointeri
- Adunarea/scăderea tipului float sau double la/din pointeri
- Înmulţirea a 2 sau mai mulţi pointeri
- Împărţirea a 2 sau mai mulţi pointeri
- Aplicarea operatorilor pe bit  pointerilor

**Operaţiile permise** asupra pointerilor sunt:

- Comparare
- Iniţializare. Atribuire
- Adunarea/Scăderea  unui nr. întreg la/dintr-un pointer
- Scăderea a 2 pointeri
- Incrementare/Decrementare

**Compararea** pointerilor se poate realiza în două moduri:

**a) Compararea a doi pointeri**

Operatorii relaţionali permit compararea a 2 pointeri într-o expresie numai dacă aceştia indică spre obiecte de acelaşi tip.

**Ex.:**

```
int *p, *q, k,j; p=&k;    q=&j;
if (p<q) printf (“p indica o adresa de memorie mai mica decit q”);
printf(“p=%p q=%p\n”,p,q );
```

**b) Comparare pointeri cu NULL**

Operatorii de egalitate = = și ! = permit compararea pointerilor cu constanta NULL (definită în <stdio.h> )

**Ex.:**

```
#define NULL 0
void *p //p=pointer generic,nu este asociat nici unui tip de date
p==NULL;
p!=NULL //verifica daca lui p ii este asociata sau nu o valoare
```

În C++ se recomandă utilizarea comparației cu 0 a pointerilor nu cu NULL : **Ex.:** p= =0 si p!=0

**Iniţializarea pointerilor** se realizează utilizând următorul format: **tip *nume =const;**

unde **tip** = tipul de bază al pointerului, defineşte tipul variabilei la care indică acesta;
**nume** = numele variabilei pointer
**const** = expresie cu valoare constantă

Operatorul de atribuire ”=” permite inițializarea și atribuirea de valori unui pointer, precum în exemplul de mai jos.

**Ex.:**

```
p=NULL;          //initializare
*p=2;            //atribuire
 p=&nr; //atribuire
```

**Adunarea/scăderea unui nr. întreg la/dintr-un pointer**

Se considera un pointer *p declarat astfel: tip *p;
Adunarea/scăderea unui număr întreg la/din acest pointer , p+n și p-n reprezintă adunarea/scăderea la adresa indicată de p a valorii **n*sizeof(tip).**

**Ex.:** p=p+10; // p va indica la al 10-lea element de acelasi tip cu p

**Scăderea a doi pointeri**

Se considera doi pointeri declarați astfel: tip *p1,*p2; Scăderea acestor doi pointeri, p1-p2 este permisă numai pentru elemente de același tip, rezultatul reprezentând nr. de obiecte dintre cei doi pointeri.

**Ex.:**

```
int q; float j[3], *p1=&j[1],*p2=&j[3];
q=p2-p1; //variabilei q i se atribuie nr.de obiecte dintre cele doua adrese:2
```

**Incrementarea/decrementarea pointerilor**

Incrementarea/decrementarea nu reprezintă adunarea/ scăderea propriuzisă cu 1 (nici a adresei nici a valorii spre care indică pointerul respectiv). Incrementarea/decrementarea semnifică adunarea/ scăderea cu sizeof (tip), unde tip este tipul pointerului respectiv. După incrementarea/decrementarea unui pointer acesta va indica spre elementul următor/anterior de același tip cu tipul său de bază.

**Ex.1:** Dacă p1 are valoarea 2000, atunci p1++ are valoarea 2004 (nu 2001, pentru că int se reprezintă pe 4 octeți) și deci va indica spre următorul întreg .
int *p1; p1++;

**Ex.2 :** Daca p1 are valoarea 2000, p1-- are valoarea 1996
int *p1; p1--;

## PROBLEME REZOLVATE

***Ex.1: Programul este un exemplu de declarare a 2 pointeri la date de tip int si respectiv real (float). Programul citeste de la tastatura o valoare intreaga x si o valoare de tip float corespunzatoare lui y si apoi afiseaza adresele variabilelor x si y si valorile lor si ale patratelor lor prin pointeri.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{int x, *p1;float y,*p2;
p1=&x; p2=&y;
printf("Introduceti o valoare intreaga x=");
scanf("%d", &x);
printf("Introduceti o valoare reala y=");
scanf("%f", &y);
printf("\nAdresa lui x este:\n");
printf("p1=%p\n", p1);printf("\nValoarea lui x
este:\n");printf("x=*p=%d\n",*p1);
printf("\nValoarea lui x^2 este:\n");
printf("*p1^2=%d\n", *p1* *p1);
printf("\nAdresa lui y este:\n");
printf("p1=%p\n", p2);
printf("\nValoarea lui y este:\n");
printf("x=*p2=%.2f\n",*p2);
printf("\nValoarea lui y^2 este:\n");
printf("*p1^2=%.2f\n", *p2* *p2);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{int x, *p1;float y,*p2;
p1=&x; p2=&y;
cout<<"Introduceti o valoare intreaga x=";
cin>>x ;
cout <<"\nIntroduceti o valoare reala y=";
cin>>y;
cout<<"\nAdresa lui x este:"<<p1;
cout<<"\nValoarea lui x este:"<<*p1;
cout<<"\nValoarea lui x^2 este:"<<*p1**p1;
cout<<"\nAdresa lui y este:"<<p2;
cout<<"\nValoarea lui y este:"<<*p2;
cout<<"\nValoarea lui y^2 este:"<<*p2* *p2;
return 0;}
```

***Rezultate:***

```
Introduceti o valoare intreaga x=2
Introduceti o valoare reala y=5.5

Adresa lui x este:
p1=0060FF04
Valoarea lui x este:
x=*p=2
Valoarea lui x^2 este:
*p1^2=4
Adresa lui y este:
p1=0060FF00
Valoarea lui y este:
x=*p2=5.50
Valoarea lui y^2 este:
*p1^2=30.25
```

***Aplicatie:***

*Să se modifice programul de mai sus astfel încât, utilizând pointeri, să se afişeze rezultatul calculului expresiei: $x^2+ 3y^2$ .*

***Ex.2 Programul este un exemplu de declarare a pointerilor la date de tip caracter si initializarea si afisarea datelor prin intermediul pointerilor.***

**Varianta in C**

```
#include <stdio.h>
int main()
{char c = 'A'; char *p = &c;
printf("Initial variabila c este %c\n",c);
printf("Continutul variabilei c se poate afisa utilizand\n");
printf("a)variabila c sau b) pointerul *p=&c\n");
printf("astfel a) c=%c b)*p=%c\n", c, *p);
printf("\nSchimbarea valorii initiale se poate realiza:\n");
printf("a) prin variabila sau b)prin pointer\n");
printf("Introduceti un caracter:");  scanf("%c", p);
printf("astfel c=%c *p=%c\n", c, *p);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{char c = 'A'; char *p = &c;
cout<<"Initial variabila c este "<<c<<endl;
cout<<"Continutul variabilei c se poate afisa utilizand"
<<endl;
cout<<"a)variabila c sau b) pointerul p=&c"<<endl;
cout<<"astfel a)c="<<c<< "b)*p="<<*p<<endl;
cout<<"\nSchimbarea valorii initiale se poate
realiza:"<<endl;
cout<<"a) prin variabila sau b)prin pointer"<<endl;
cout<<"Introduceti un caracter:";cin>>*p;
cout<<"astfel c="<<c<<" *p="<<*p;return 0;}
```

***Rezultate:***

```
Initial variabila c este A
Continutul variabilei c se poate afisa utilizand
a)variabila c sau b) pointerul *p=&c
astfel a) c=A b)*p=A

Schimbarea valorii initiale se poate realiza:
a) prin variabila sau b)prin pointer
Introduceti un caracter:B
astfel c=B *p=B
```

***Aplicatie:***

*Sa se tipareasca şi adresa variabilei c utilizand pointerul *p.*

***Ex.3. Programul calculeaza si afiseaza rezultatele unor expresii cu numere intregi prin intermediul pointerilor. Initial se considera $i_1$=8 si se cere sa se afiseze $i_2=i_1/2+1$ direct si prin intermediul pointerului *p1, apoi se va vor afisa rezultatele operatiilor aritmetice $i_1+i_2$, $i_1-i_2$, $i_1*i_2$, $i_1/i_2$ prin pointeri.***

**Varianta in C**

```
#include <stdio.h>
int main()
{int i1=8, i2, *p1, *p2;
p1 = &i1; p2 = &i2;
printf("daca i1=8 si p1=&i1\n");
printf("atunci operatiile sunt echivalente:\n");
printf("i2=i1/2+1=%d <=>i2=*p1/2+1=%d\n",
i1/2+1,*p1/2+1);i2=*p1/2+1;
printf("i1=%d, i2=%d\n", *p1, *p2);
printf("operatiile aritmetice se pot realiza in 2 moduri:\n");
printf("a)i1+i2=%d,i1-i2=%d,i1*i2=%d, i1/i2=%d\n", i1+i2,i1-
i2,i1*i2,i1/i2);
printf("b)*p1+*p2=%d,*p1-*p2=%d, *p1**p2=%d,
*p1/*p2=%d\n", *p1+*p2, *p1-*p2, *p1**p2,*p1/(*p2));
return 0;
}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{int i1=8, i2, *p1, *p2;
p1 = &i1; p2 = &i2;
cout<<"daca i1=8 si p1=&i1"<<endl;
cout<<"atunci operatiile sunt echivalente:"<<endl;
cout<<"i2=i1/2+1="<<i1/2+1<<" <=>
i2=*p1/2+1="<<*p1/2+1<<endl;i2=*p1/2+1;
cout<<"i1="<<*p1<<", i2="<<*p2<<endl;
cout<<"operatiile aritmetice se pot realiza in 2
moduri:"<<endl;
cout<<"a)i1+i2="<<*p1+*p2<<", i1-i2="<<i1+i2<<",
i1*i2="<<i1*i2<<", i1/i2="<<i1/i2<<endl;
cout<<"b)*p1+*p2="<<*p1+*p2<<", *p1-*p2="<< *p1-
*p2<<", *p1**p2="<<*p1**p2<<",
*p1/*p2="<<*p1/(*p2)<<endl;
return 0;
}
```

***Rezultate:***

```
daca i1=8 si p1=&i1
atunci operatiile sunt echivalente:
i2=i1/2+1=5 <=>i2=*p1/2+1=5
i1=8, i2=5
operatiile aritmetice se pot realiza in 2 moduri:
a)i1+i2=13,i1-i2=3,i1*i2=40, i1/i2=1
b)*p1+*p2=13,*p1-*p2=3, *p1**p2=40, *p1/*p2=1
```

***Aplicatie:***

*Să se modifice programul astfel încât sa se afişeze şi adresele variabilelor i1 şi i2*

***Ex.4. In program se considera 4 variabile: de tip int, float, double si char carora le corespunde cate un pointer:*a,*c,*d, *e. Programul realizeaza diverse operatii cu pointeri.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{int i=1, *a,*a1;
float x=1.,*c,*c1;
double pi=3.14, *d,*d1;
char ch='A',*e,*e1;a=&i;
printf("i=variabila de tip int,i=%d\n",i);
printf("a=variabila de tip pointer la int,a=&i=%p\n",a);
printf("a-1=%p a+1=%p\n", a-1,a+1);
printf("sizeof(int):%d\n",sizeof(int) );
printf("comparare a si a+1: ");
if (a<(a+1)) printf("%p < %p\n",a,a+1);
else printf("%p > %p\n",a,a+1);
a1=a+3;
printf("a1=a+3=%p\n", a1);
printf("a1-a=%d\n", a1-a);
printf("-------------------------------------------------\n");
c=&x;
printf("x=variabila de float,x=%f\n",x);
printf("c=variabila de tip pointer la float,  c=&x=%p\n",c);
printf("c-1=%p c+1=%p\n", c-1,c+1);
printf("sizeof(float):%d\n",sizeof(float) );
c1=c+2;
printf("c+2=%p\n", c1);
printf("c1=c+2;c1-c=%d\n", c1-c);
printf("-------------------------------------------------\n");
d=&pi;
printf("pi=variabila de tip double,pi=%lf\n",pi);
printf("d=variabila de tip pointer la
double,d=&pi=%p\n",d);
printf("d-1=%p d+1=%p\n", d-1,d+1);
printf("sizeof(double):%d\n",sizeof(double) );
d1=d-2;
printf("d-2=%p\n", d1);
printf("d1=d-2;d1-d=%d\n", d1-d);
printf("-------------------------------------------------\n");
e=&ch;
printf("ch=variabila de tip char,ch=%c\n",ch);
printf("e=variabila de tip pointer la char,  e=&ch=%p\n",e);
printf("e-1=%p e+1=%p\n", e-1,e+1);
printf("sizeof(char):%d\n",sizeof(char) );e1=e+3;
printf("e+3=%p\n", e1);
printf("e1=e+3,e1-e=%d\n", e1-e);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{int i=1, *a,*a1;float x=1.,*c,*c1;
double pi=3.14, *d,*d1;
char ch='A',*e,*e1;a=&i;
cout<<"i=variabila de tip int,i="<<i<<endl;
cout<<"a=variabila de tip pointer la int,a=&i="<<a<<endl;
cout<<"a-1="<<a-1<<", a+1="<<a+1<<endl;
cout<<"sizeof(int):"<<sizeof(int)<<endl;
cout<<"comparare a si a+1: ";
if (a<(a+1)) cout<<a<<"<"<<a+1<<endl;
else cout<<a<<">"<<a+1<<endl;a1=a+3;
cout<<"a1=a+3="<<a1<<endl;
cout<<"a1-a="<<a1-a<<endl;
cout<<"-------------------------------------------------"<<endl;
c=&x;
cout<<"x=variabila de float,x="<<x<<endl;
cout<<"c=variabila de tip pointer la float,
c=&x="<<c<<endl;
cout<<"c-1="<<c-1<<" c+1="<<c+1<<endl;
cout<<"sizeof(float):"<<sizeof(float)<<endl;c1=c+2;
cout<<"c+2="<<c1<<endl;
cout<<"c1=c+2;c1-c="<<c1-c<<endl;
cout<<"-------------------------------------------------"<<endl;
d=&pi;
cout<<"pi=variabila de tip double,pi="<<pi<<endl;
cout<<"d=variabila de tip pointer la
double,d=&pi="<<d<<endl;
cout<<"d-1="<<d-1<<" d+1="<<d+1<<endl;
cout<<"sizeof(double):"<<sizeof(double)<<endl;d1=d-2;
cout<<"d-2="<<d1<<endl;
cout<<"d1=d-2;d1-d="<<d1-d<<endl;
cout<<"-------------------------------------------------"<<endl;
e=&ch;
cout<<"ch=variabila de tip char,ch="<<ch<<endl;
cout<<"e=variabila de tip pointer la char, e=&ch="<<
(void *)e<<endl;
cout<<"e-1="<<(void *)(e-1)<<" e+1="<<(void
*)(e+1)<<endl;
cout<<"sizeof(char):"<<sizeof(char)<<endl;
e1=e+3;
cout<<"e+3="<<(void *)e1<<endl;
cout<<"e1=e+3,e1-e="<<e1-e<<endl;
return 0;}
```

***Rezultate:***

```
a-1=0060FEE8 a+1=0060FEF0
sizeof(int):4
comparare a si a+1: 0060FEEC < 0060FEF0
a1=a+3=0060FEF8
a1-a=3
-------------------------------------------------
x=variabila de float,x=1.000000
c=variabila de tip pointer la float,  c=&x=0060FEE8
c-1=0060FEE4 c+1=0060FEEC
sizeof(float):4
c+2=0060FEF0
c1=c+2;c1-c=2
-------------------------------------------------
pi=variabila de tip double,pi=3.140000
d=variabila de tip pointer la double,d=&pi=0060FEE0
d-1=0060FED8 d+1=0060FEE8
sizeof(double):8
d-2=0060FED0
d1=d-2;d1-d=-2
-------------------------------------------------
ch=variabila de tip char,ch=A
e=variabila de tip pointer la char,  e=&ch=0060FEDF
e-1=0060FEDE e+1=0060FEE0
sizeof(char):1
e+3=0060FEE2
e1=e+3,e1-e=3
```

***Aplicație:***

*Să se modifice programul de mai sus astfel încât, utilizând pointeri, să se afişeze adresele: a+2, c1, d+2 si e+2.*

## PROBLEME PROPUSE

1. **Sa se scrie un program care afiseaza solutia ecuatiei de gradul I unde coeficientii se citesc de la tastatura utilizand pointeri si operatii prin pointeri**

2. **Sa se scrie un program care calculeaza si afiseaza valoarea functiei f(x) = $x^2$-1 utilizand pointeri si operatii prin pointeri.**

# Capitolul 3
# Pointeri la tablouri

În acest capitol sunt prezentate consideraţii teoretice privind definirea şi utilizarea pointerilor, operațiile permise și nepermise utilizând pointeri şi câteva probleme rezolvate cu pointeri la date de tip standard: întreg, caracter şi respectiv şir de caractere.

## CONSIDERAŢII TEORETICE

### a) Pointeri la tablouri unidimensionale (siruri)

Accesul la elementele tablourilor se realizează în C++ în 2 moduri:

- prin intermediul indicilor
- prin aritmetica pointerilor. Avantaj: accesul e mult mai rapid și crește viteza de execuție a programului

Numele unui tablou (șir/matrice) - scris fără indice = **pointer constant** și este de tipul elementelor tabloului și are ca și valoare adresa primului element al tabloului

Se consideră tabloul unidimensional a[i], i=0,n-1. Prin convenție: denumirea tabloului = pointerul la tablou = adresa primului element al tabloului (a[0])

**Notațiile** de mai jos sunt **echivalente**:

**a** **&a** **&a[0]**

unde a = constantă (pointer constant) și deci operațiile aritmetice cu pointeri nu se pot aplica asupra pointerilor constanți.

Din această cauză instrucțiunea: a++ este incorectă pentru că a este un pointer constant! De aceea pointerul la un tablou trebuie declarat ca și pointer variabil:

```
tip a[10], *p;
p=&a[0];  // sau p=a;  sau p=&a;
```

In acest caz *p este o variabilă pointer, iar a este pointer constant.

De asemenea dacă se consideră tabloul unidimensional a[i], i=0,n-1, a este pointerul constant la șir, iar **notațiile** de mai jos sunt **echivalente**:

termenul **a[i] :** ***(a+i)**

adresa termenului **a[i]: &a[i]** **a+i**

Dacă se declară o variabilă de tip pointer la acest șir, *p și se inițializează cu instrucțiunea p=&a[0]; sau p=a; sau p=&a, atunci **notațiile** de mai jos sunt **echivalente**:

**a[i]** ***(a+i)** ***(p+i)** **p[i]**

### b) Pointeri la tablouri bidimensionale (matrici)

Se consideră tabloul bidimensional a[i][j], i=0,m-1; j=0,n-1. Prin convenție: denumirea tabloului = pointerul la tablou = adresa primului element al tabloului (a[0][0]).

**Notațiile** de mai jos sunt **echivalente**:

**a** **&a** **&a[0][0]**

unde a = numele tabloului= constantă (pointer constant) și deci operațiile aritmetice cu pointeri nu sunt permise ,

De aceea pointerul la tablou trebuie declarat ca și pointer variabil:

**tip a[linii][coloane], *pa;**
**pa=&a[0][0];  // sau pa=a;**

In acest caz *pa este o variabila de tip pointer.

**Adresa** lui a[i][j] se poate determina utilizând una din **notațiile echivalente** de mai jos:

**&a[i][j]  a[i]+j  p+k  *(a+i)+j**

**Valoarea** lui a[i][j] se poate determina utilizând una din notațiile echivalente :

**a[i][j]  *(a[i]+j)  *(p+k)  *(*(a+i)+j)**

unde k=i*n+j .

## PROBLEME REZOLVATE

***Ex 1. Programul citeste elementele a 2 vectori  x[i] si y[i], de dimensiune egala cu n, si calculeaza si afiseaza produsul lor scalar, utilizand variabile de tip pointer.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{int i,n,x[100],y[100],s, *p,*q;
//citire dimensiune vectori
printf("Introduceti dimensiunea vectorilor x si y, n=");
scanf("%d", &n);
//citire elemente vectori x, y
p=x,q=y;
for(i=0;i<n;i++)
            { printf("x[%d]=",i); scanf("%d", p+i);
              printf("y[%d]=",i); scanf("%d", q+i);}
//calcul prod scalar vectori
s=0;
for (i=0;i<n;i++,p++,q++)  s+=*p* *q;
printf(" produsul scalar (x,y)=%d\n", s);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{int i,n,x[100],y[100],s, *p,*q;
//citire dimensiune vectori
cout<<"Introduceti dimensiunea vectorilor x si y,
n="<<endl;
cin>>n;
//citire elemente vectori x, y
p=x,q=y;
for(i=0;i<n;i++)
            { cout<<"x["<<i<<"]=";   cin>>*(p+i);
              cout<<"y["<<i<<"]=";  cin>>*(q+i);}
//calcul prod scalar vectori
s=0;
for (i=0;i<n;i++,p++,q++)  s+=*p* *q;
cout<<" produsul scalar (x,y)="<<s<<endl;
return 0;}
```

***Rezultate:***

```
Introduceti dimensiunea vectorilor x si y, n=3
x[0]=1
y[0]=2
x[1]=1
y[1]=2
x[2]=1
y[2]=2
 produsul scalar (x,y)=6
```

***Aplicație:***

*Să se modifice programul de mai sus astfel încât, vectorii să aibă elemente de tip real (double)*

***Ex.2. Programul realizează inițializarea unui șir de numere întregi cu valori cuprinse în intervalele (1,10),(11,20), … etc.,utilizând pointeri și aritmetica de pointeri.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{int i,j,a[100], *p;
printf ("\na.Initializare sir cu nr de la 1-10");
printf("\nfolosind numele sirului si un index\n");
for (i=1;i<11;i++)  {a[i]=i;  printf("%4d", a[i]);}
printf ("\n\nb.Initializare sir cu nr. de la 11-20");
printf("\nfolosind un pointer si un index\n");
for (p=a,i=11;i<21;i++) {p[i]=i; printf("%4d", p[i]);}
printf("\n\nc.Initializare sir cu nr. de la 21 la 30");
printf("\nfolosind un pointer si aritmetica pointerilor\n");
for (p=a,j=21;p<a+10;j++,p++)
  {*p=j;printf("%4d",*p);}
printf ("\n\nd.Initializare sir cu nr.de la 31 la 40");
printf("\nfolosind un pointer si aritmetica de pointeri\n");
for (p=a,j=31;p<&a[10];j++,p++)  {*p=j;printf("%4d",*p);}
printf("\n");
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{int i,j,a[100], *p;
cout<<endl<<"a.Initializare sir cu nr de la 1-10"<<endl;
cout<<"folosind numele sirului si un index"<<endl;
for (i=1;i<11;i++)    {a[i]=i; cout<<a[i]<<"  ";}
cout<<"\n\nb.Initializare sir cu nr. de la 11-20"<<endl;
cout<<"folosind un pointer si un index"<<endl;
for (p=a,i=11;i<21;i++)       {p[i]=i; cout<< p[i]<<"  ";}
cout<<"\n\nc.Initializare sir cu nr.de la 21 la 30"<<endl;
cout<<"folosind un pointer si aritmetica
pointerilor"<<endl;
for (p=a,j=21;p<a+10;j++,p++)
  {*p=j;cout<<*p<<"  ";}
cout<<"\n\nd.Initializare sir cu nr.de la 31 la 40"<<endl;
cout<<"folosind un pointer si aritmetica de
pointeri"<<endl;
for (p=a,j=31;p<&a[10];j++,p++) {*p=j;cout<<*p<<" ";}
cout<<endl;return 0;}
```

***Rezultate:***

```
a.Initializare sir cu nr de la 1-10
folosind numele sirului si un index
   1   2   3   4   5   6   7   8   9  10

b.Initializare sir cu nr. de la 11-20
folosind un pointer si un index
  11  12  13  14  15  16  17  18  19  20

c.Initializare sir cu nr. de la 21 la 30
folosind un pointer si aritmetica pointerilor
  21  22  23  24  25  26  27  28  29  30

d.Initializare sir cu nr. de la 31 la 40
folosind un pointer si aritmetica de pointeri
  31  32  33  34  35  36  37  38  39  40
```

***Aplicație:***

*Să se modifice programul astfel încât să se afișeze șirul inițializat cu valori reprezentând cubul numerelor întregi din intervalul [1,100].*

***Ex. 3. Programul calculează suma tuturor elementelor și produsul elementelor strict pozitive ale unui șir de numere întregi, utilizând pointeri .***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{int a[10]={-1,0,1,10,2,-2,0,-5,1,-4};
int *pa, s=0, p=1,i; pa=a;
printf("Tiparirea valorilor sirului\n   a[i]\t   *(a+i)\n");
for (i=0;i<10;i++)
{printf("a[%d]=%3d, *(a+%d)=%3d\n",i,a[i],i,*(a+i));
            s+=*(pa+i);       //sau s+=a[i];
            if(*(pa+i)>0) p*=*(pa+i);  //sau if (a[i]>0) p*=a[i];}
printf("Suma tuturor elementelor sirului:\n");
printf("s=s+a[i]=%d\n",s);
printf("Produsul elementelor strict pozitive:\n");
printf("p=p*a[i])=%d\n", p);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{int a[10]={-1,0,1,10,2,-2,0,-5,1,-4};
int *pa, s=0, p=1,i; pa=a;
cout<<"Tiparirea valorilor sirului\n a[i]\t *(a+i)"<<endl;
for (i=0;i<10;i++)
{cout<<"a["<<i<<"]="<<a[i]<<",*(a+"<<i<<")="<<
*(a+i) <<endl;
            s+=*(pa+i);       //sau s+=a[i];
            if(*(pa+i)>0) p*=*(pa+i);//sau if (a[i]>0) p*=a[i]; }
cout<<"Suma tuturor elementelor sirului:"<<endl;
cout<<"s=s+a[i]="<<s<<endl;
cout<<"Produsul elementelor strict pozitive:"<<endl;
cout<<"p=p*a[i]="<<p<<endl;return 0;}
```

***Rezultate:***

```
Tiparirea valorilor sirului
   a[i]    *(a+i)
a[0]= -1, *(a+0)= -1
a[1]=  0, *(a+1)=  0
a[2]=  1, *(a+2)=  1
a[3]= 10, *(a+3)= 10
a[4]=  2, *(a+4)=  2
a[5]= -2, *(a+5)= -2
a[6]=  0, *(a+6)=  0
a[7]= -5, *(a+7)= -5
a[8]=  1, *(a+8)=  1
a[9]= -4, *(a+9)= -4
Suma tuturor elementelor sirului:
s=s+a[i]=2
Produsul elementelor strict pozitive:
p=p*a[i])=20
```

***Aplicație:***

*Să se modifice programul astfel încât să se citească șirul de la tastatură (citind înainte dimensiunea sa) și să se calculeze și afișeze suma elementelor din intervalul [1,100] și produsul elementelor din intervalul [10,50].*

***Ex.4. Programul afișează un șir de caractere in 2 moduri: fără pointer și utilizând un pointer indexat.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{char sir[]="Pointeri la siruri de caractere",*p;
int i; p=sir;
printf("Sirul tiparit fara pointer:\n");
printf("%s",sir);
printf("\n\nSirul tiparit cu pointer indexat:\n");
for (i=0;p[i];i++) printf("%c", p[i]);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{char sir[]="Pointeri la siruri de caractere",*p;
int i; p=sir;
cout<<"Sirul tiparit fara pointer:"<<endl;
cout<<sir<<endl<<endl;
cout<<"Sirul tiparit cu pointer indexat:"<<endl;
for (i=0;p[i];i++) cout<<p[i];
return 0;}
```

***Rezultate:***

```
Sirul tiparit fara pointer:
Pointeri la siruri de caractere

Sirul tiparit cu pointer indexat:
Pointeri la siruri de caractere
```

***Aplicație:***

*Să se modifice programul astfel încât să se afișeze șirul inversat, prin pointeri.*

***Ex.5. Programul citeşte un şir de numere reale a[i], i=1,n, n întreg citit de la tastatură, şi creează un şir b[i] format după formula:*** $b_i = \frac{a_i}{(1 - a_i)^2}$ ***, utilizând pointeri.***

**Varianta in C**

```
#include <stdio.h>
#include<stdlib.h>
#include <math.h>
int main()
{double a[100],*p; int i, n; p=a;
printf("n=");scanf("%d", &n);
for(i=0;i<n;i++) { printf("a[%d]=",i); scanf("%lf", p+i);}
for (i=0;i<n;i++)
 {if (pow((1-*(p+i)),2)==0) {printf("numitor 0!"); exit(1); }
   else
  printf("b[%d]=%.2lf\n",i,*(p+i)/pow((1-*(p+i)),2));}
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <math.h>
#include<stdlib.h>
using namespace std;
int main()
{double a[100],*p; int i, n; p=a;
cout<<"n=";cin>>n;
for(i=0;i<n;i++) { cout<<"a["<<i<<"]="; cin>>*(p+i);}
for (i=0;i<n;i++)
 {if (pow((1-*(p+i)),2)==0) {cout<<"numitor 0!"; exit(1); }
   else  cout<<"b["<<i<<"]="<<*(p+i)/pow((1-
*(p+i)),2)<<endl;} return 0;}
```

***Rezultate:***

```
n=5
a[0]=10
a[1]=11
a[2]=12
a[3]=13
a[4]=14
b[0]=0.12
b[1]=0.11
b[2]=0.10
b[3]=0.09
b[4]=0.08
```

***Aplicație:***

*Să se modifice programul astfel încât să se afișeze șirul* $b_i = 5a_i + 7$ *prin pointeri.*

***Ex. 6. Programul calculează și afișează primii 10 de termeni ai celor 2 şiruri definite după formulele de mai jos:***

$$a_n = \frac{1}{1} - \frac{1}{2} + \frac{1}{3} - \ldots - \frac{1}{2n},\ b_n = \frac{1}{n+1} + \frac{1}{n+2} + \frac{1}{n+3} + \ldots + \frac{1}{2n}$$ ***,utilizând pointeri.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
# include<math.h>
int main()
{ float a[100],b[100],*pa,*pb; int i,n; pa=a;pb=b;
printf(" n=");
scanf("%d",&n);
*(pa+1)=1;
printf("a[1]=%lf", *(pa+1));
*(pb+1)=1./(n+1);
printf ("\tb[1]=%lf", *(pb+1));
for(i=2;i<=n;i++)
{  *(pa+i)=*(pa+i-1)+pow(-1,i+1)*(1./i);
printf("\na[%d]=%f",i,*(pa+i));
  *(pb+i)=*(pb+i-1)+1./(n+i);
printf("\tb[%d]=%f",i, *(pb+i));}
return 0;}
```

**Varianta in C++**

```
#include <iostream>
# include<math.h>
using namespace std;
int main()
{ float a[100],b[100],*pa,*pb; int i,n; pa=a;pb=b;
*(pa+1)=1;
cout<<"a[1]="<<*(pa+1);
*(pb+1)=1./(n+1);
cout<<"\tb[1]="<<*(pb+1);
for(i=2;i<=n;i++)
{  *(pa+i)=*(pa+i-1)+pow(-1,i+1)*(1./i);
cout<<"\na["<<i<<"]="<<*(pa+i);
   *(pb+i)=*(pb+i-1)+1./(n+i);
cout<<"\tb["<<i<<"]="<<*(pb+i);}
return 0;}
```

***Rezultate:***

```
 n=10
a[1]=1.000000   b[1]=0.090909
a[2]=0.500000   b[2]=0.174242
a[3]=0.833333   b[3]=0.251166
a[4]=0.583333   b[4]=0.322594
a[5]=0.783333   b[5]=0.389261
a[6]=0.616667   b[6]=0.451761
a[7]=0.759524   b[7]=0.510584
a[8]=0.634524   b[8]=0.566140
a[9]=0.745635   b[9]=0.618771
a[10]=0.645635  b[10]=0.668771
```

***Aplicație:***

*Să se modifice programul astfel încât să se afișeze șirul format din produsul elementelor a[i]*b[i] utilizând pointeri.*

***Ex.7. Programul citește elementele unei matrici de dimensiune nxm, și le afișează prin trei metode diferite utilizând pointeri.***

**Varianta in C**

```
#include <stdio.h>
int main()
{int i, j, n,m,a[100][100];
printf ("n=");scanf("%d", &n);
printf ("m=");scanf("%d", &m);
for (i=0;i<n;i++) for (j=0;j<m;j++)
{printf("  a[%d,%d]=", i,j); scanf("%d", *(a+i)+j); }
for (i=0;i<n;i++) for (j=0;j<m;j++)
{printf("\na[%d][%d]=%3d", i,j,a[i][j]);
printf("  *a[%d]+%d=%3d", i,j,*(a[i]+j));
printf("  *(*(a+%d)+%2d)=%d", i,j,*(*(a+i)+j));}
printf("\n");
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{int i, j, n,m,a[100][100];
cout<<" n=";cin>>n;
cout<<" m=";cin>>m;
for (i=0;i<n;i++) for (j=0;j<m;j++)
{cout<<"  a["<<i<<","<<j<<"]=";cin>>*(*(a+i)+j); }
for (i=0;i<n;i++) for (j=0;j<m;j++)
{cout<<"\na["<<i<<"]["<<j<<"]="<<a[i][j];
cout<<"  *a["<<i<<"]+"<<j<<"="<<*(a[i]+j);
cout<<"  *(*(a+"<<i<<")+"<<j<<")="<<*(*(a+i)+j);}
cout<<endl;return 0;}
```

***Rezultate:***

```
n=2
m=3
  a[0,0]=1
  a[0,1]=2
  a[0,2]=3
  a[1,0]=4
  a[1,1]=5
  a[1,2]=6

a[0][0]=  1  *a[0]+0=  1  *(*(a+0)+ 0)=1
a[0][1]=  2  *a[0]+1=  2  *(*(a+0)+ 1)=2
a[0][2]=  3  *a[0]+2=  3  *(*(a+0)+ 2)=3
a[1][0]=  4  *a[1]+0=  4  *(*(a+1)+ 0)=4
a[1][1]=  5  *a[1]+1=  5  *(*(a+1)+ 1)=5
a[1][2]=  6  *a[1]+2=  6  *(*(a+1)+ 2)=6
```

***Aplicație:***

*Să se modifice programul astfel încât să se calculeze și afișeze elementele matricii multiplicate cu 100.*

***Ex.8. Programul determină și afișează maximul elementelor unei matrici de nxm elemente citite de la tastatură utilizând pointeri.***

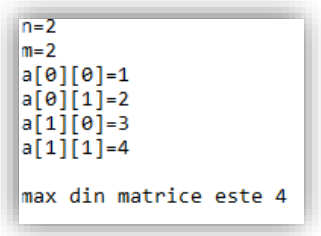
**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{int max;
int a[10][10],i,j,n,m,*p;
p=&a[0][0];
printf("n=");scanf("%d", &n);
printf("m=");scanf("%d", &m);
for (i=0;i<n;i++)
for (j=0;j<m;j++)
{ printf("a[%d][%d]=",i,j); scanf("%d", *(a+i)+j);}
max=*p;
for (i=0; i < n; i++)
  for (j=0; j < m; j++)
    if (max < *(*(a+i)+j))  max = *(*(a+i)+j);
printf("\n max din matrice este %d\n",max);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{int maxi;
int a[10][10],i,j,n,m,*p;
p=&a[0][0];
cout<<"n=";cin>>n;
cout<<"m=";cin>>m;
for (i=0;i<n;i++)
for (j=0;j<m;j++)
{ cout<<"a["<<i<<"]["<<j<<"]="; cin>>*(*(a+i)+j);}
maxi=*p;
for (i=0; i < n; i++)
  for (j=0; j < m; j++)
    if (maxi < *(*(a+i)+j))  maxi = *(*(a+i)+j);
cout<<endl<<"max din matrice este "<<maxi<<endl;
return 0;}
```

***Rezultate:***




***Aplicație:***

*Să se modifice programul astfel încât să se determine și minimul elementelor matricii și pozitia minimului și a maximului în matrice.*

***Ex. 9. Programul defineste un pointer la un sir de caractere prin intermediul caruia se va afisa sirul inversat.***

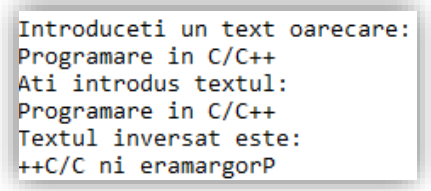
**Varianta in C**

```
#include <stdio.h>
#include <string.h>
int main()
{int i; char c[100], *p;
printf("Introduceti un text oarecare:\n");gets(c);
p=c;    //<=>p=&c, sau p=&c[0]
printf("Ati introdus textul:\n");printf(p);
printf("\nTextul inversat este:\n");
for (i=strlen(p)-1;i>=0;i--) printf("%c", p[i]);
printf("\n");
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{int i; char c[100], *p;
cout<<"Introduceti un text oarecare:"<<endl;
cin.get(c,100);
p=c;    //<=>p=&c, sau p=&c[0]
cout<<"Ati introdus textul:"<<endl<<p;
cout<<endl<<"Textul inversat este:"<<endl;
for (i=strlen(p)-1;i>=0;i--)     cout<<p[i];
cout<<endl;
return 0;}
```

***Rezultate:***




***Aplicatie:***

*Sa se modifice programul astfel încât sa se afişeze sirul inversat, utilizand pointeri si o alta metoda de tiparire*

***Ex.10. Programul afisează un sir de caractere (caracter cu caracter) introdus de la tastatura prin intermediul pointerilor si numără cate caractere are acest sir.***

**Varianta I in C**

```
#include <stdio.h>
#include <string.h>
int main()
{int i=0; char c[100], *p;
printf("Introduceti un text oarecare:\n");gets(c);
p=c; //<=>p=&c, sau p=&c[0]
printf("Ati introdus textul:\n");
while(*p) {i++;putchar(*p++);}
printf("\nNr de caractere (inclusiv spatiile) din text:%d\n", i);
return 0;
}
```

**Varianta I in C++**

```
#include <iostream>
#include <stdio.h>
#include <string.h>
using namespace std;
int main()
{int i=0; char c[100], *p;
cout<<"Introduceti un text oarecare:"<<endl;
cin.get(c,100);
p=c; //<=>p=&c, sau p=&c[0]
cout<<"Ati introdus textul:"<<endl;
while(*p) {i++;putchar(*p++);}
cout<<endl<<"Nr de caractere (inclusiv spatiile) din
text:"<< i;
return 0;}
```

**Varianta II in C**

```
#include <stdio.h>
#include <string.h>
int main()
{int i=0; char c[100], *p;
printf("Introduceti un text oarecare:\n");gets(c);
p=c; //<=>p=&c, sau p=&c[0]
printf("Ati introdus textul:\n");
for (i=0;p[i];i++) printf("%c", p[i]);
printf("\nNr de caractere (inclusiv spatiile) din text:%d\n", i);
return 0;}
```

**Varianta II in C++**

```
#include <iostream>
#include <stdio.h>
#include <string.h>
using namespace std;
int main()
{int i=0; char c[100], *p;
cout<<"Introduceti un text oarecare:"<<endl;
cin.get(c,100);
p=c; //<=>p=&c, sau p=&c[0]
cout<<"Ati introdus textul:"<<endl;
for (i=0;p[i];i++) cout<<p[i];
cout<<endl<<"Nr de caractere (inclusiv spatiile) din
text:"<< i;
return 0;}
```

***Rezultate:***

```
Introduceti un text oarecare:
Programare in C/C++
Ati introdus textul:
Programare in C/C++
Nr de caractere (inclusiv spatiile) din text:19
```

***Aplicatie:***

*Să se modifice programul astfel incat sa se utilizeze o altă modalitate de afişare a şirului.*

***Ex.12. Programul realizeaza copierea unui sir in alt sir utilizand 2 pointeri la siruri si 3 functii (functiile de citire/afisare sir si functia de copiere prin intermediul acestor pointeri).Functia de copiere este similara cu functia strcpy din biblioteca <string.h>***

**Varianta in C**

```
#include <stdio.h>
void strcp(char *s,char *t); //copiaza t in s
void scriesir(char *s);
void citiresir(char *s);
int main ()
{char sir1[100],sir2[100]; char *s,*t;  s=sir1;  t=sir2;
printf("Introduceti primul sir: ");
citiresir(s);scriesir(s);
printf("\n\nIntroduceti sirul al doilea: ");
citiresir(t);scriesir(t);
strcp(s,t); printf("\nCopiem sirul al doilea peste primul
sir\n");
printf("Dupa copiere sirurile sunt :\n");
printf("primul sir: ");scriesir(s);printf("\n");
printf("sirul al doilea: "); scriesir(t);printf("\n");
return 0;}
void citiresir(char *p)
{ gets(p);}
void scriesir(char *q)
{ while(*q) putchar(*q++);}
void strcp(char *p,char *q)
{while ((*p=*q)!='\0')
{p++; q++;}}
```

**Varianta in C++**

```
#include <iostream>
#include <stdio.h>
using namespace std;
void strcp(char *s,char *t); //copiaza t in s
void scriesir(char *s);
void citiresir(char *s);
int main ()
{char sir1[100],sir2[100]; char *s,*t;  s=sir1;  t=sir2;
cout<<"Introduceti primul sir: ";
citiresir(s);scriesir(s);
cout<<endl<<endl<<"Introduceti sirul al doilea: ";
citiresir(t);scriesir(t);
strcp(s,t); cout<<endl<<"Copiem sirul al doilea peste
primul sir"<<endl;
cout<<"Dupa copiere sirurile sunt :"<<endl;
cout<<"primul sir: ";scriesir(s);cout<<endl;
cout<<"sirul al doilea: "; scriesir(t);cout<<endl;
return 0;}
void citiresir(char *p)
{ gets(p);}
void scriesir(char *q)
{ while(*q) putchar(*q++);}
void strcp(char *p,char *q)
{while ((*p=*q)!='\0')
 {p++; q++;}}
```

***Rezultate:***

```
Introduceti primul sir: Curs Programare
Curs Programare

Introduceti sirul al doilea: C/C++
C/C++
Copiem sirul al doilea peste primul sir
Dupa copiere sirurile sunt :
primul sir: C/C++
sirul al doilea: C/C++
```

***Aplicatie:***

*In mod similar, sa se scrie programul care rezolva concatenarea şi compararea a 2 siruri utilizînd pointeri la şiruri de caractere.*

***Ex.13. Programul afiseaza 2 siruri de caractere prin intermediul pointerilor în diverse variante şi schimba sirurile între ele.***

**Varianta in C**

```
#include <stdio.h>
#include <string.h>
int main()
{char *p="Curs  de ";
char *q="Programare in C/C++!";
char *aux; int i;
printf("Afisarea sirurilor de caractere\n");
printf("prin intermediul pointerilor\n");
//varianta 1 de afisare a sirurilor
printf("\nVarianta I\n");
printf("pointerul p indica:"); printf(p); printf("\n");
printf("pointerul q indica:"); printf(q); printf("\n");
//varianta 2 de afisare a sirurilor
printf("\n\nVarianta II\n");
printf("pointerul p indica:");
for (i=0; i<strlen(p)+1;i++) printf("%c", p[i]);printf("\n");
printf("pointerul q indica:");
for (i=0; i<strlen(q)+1;i++) printf("%c", q[i]); printf("\n");
//varianta 3 de afisare a sirurilor
printf("\n\nVarianta III\n");
printf("pointerul p indica:");printf("%s\n", p);
printf("pointerul q indica:");printf("%s\n", q);
printf("pointerul p indica:");
//inversarea sirurilor
printf("\n\ninversam sirurile prin inversarea pointerilor\n");
aux=p; p=q; q=aux;
printf("pointerul p indica:"); printf(p); printf("\n");
printf("pointerul q indica:") ;printf(q); printf("\n");
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <stdio.h>
#include <string.h>
using namespace std;
int main()
{char *p="Curs  de ";
char *q="Programare in C/C++!";
char *aux; int i;
cout<<"Afisarea sirurilor de caractere"<<endl;
cout<<"prin intermediul pointerilor"<<endl;
//varianta 1 de afisare a sirurilor
cout<<"Varianta I"<<endl;
cout<<"pointerul p indica:"<<p<<endl;
cout<<"pointerul q indica:"<<q<<endl;
//varianta 2 de afisare a sirurilor
cout<<endl<<"Varianta II"<<endl;
cout<<"pointerul p indica:"<<endl;
for (i=0; i<strlen(p)+1;i++) cout<<p[i]<<endl;
cout<<"pointerul q indica:";
for (i=0; i<strlen(q)+1;i++) cout<<p[i]<<endl;
//varianta 3 de afisare a sirurilor
cout<<endl<<"Varianta III"<<endl;
cout<<"pointerul p indica:"<<p<<endl;
cout<<"pointerul q indica:"<< q<<endl;
cout<<"pointerul p indica:"<<endl;
//inversarea sirurilor
cout<<"inversam sirurile prin inversarea
pointerilor"<<endl;
aux=p; p=q; q=aux;
cout<<"pointerul p indica:"<<p<<endl;
cout<<"pointerul q indica:"<<q<<endl;
return 0;}
```

***Rezultate:***

```
Afisarea sirurilor de caractere
prin intermediul pointerilor

Varianta I
pointerul p indica:Curs  de
pointerul q indica:Programare in C/C++!


Varianta II
pointerul p indica:Curs  de
pointerul q indica:Programare in C/C++!


Varianta III
pointerul p indica:Curs  de
pointerul q indica:Programare in C/C++!
pointerul p indica:

inversam sirurile prin inversarea pointerilor
pointerul p indica:Programare in C/C++!
pointerul q indica:Curs  de
```

***Aplicatie:***

Sa se modifice programul astfel încât sa se concateneze sirurile utilizand pointeri **.**

## PROBLEME PROPUSE

1. ***Să se scrie programul care ordonează un şir de numere reale prin cel puţin două metode diferite de sortare, utilizând pointeri.***

2. ***Să se scrie un program care calculează primii 30 de termeni ai celor 2 şiruri definite după formulele de mai jos:***
   $a_n = \frac{1}{1} - \frac{1}{2} + \frac{1}{3} - \ldots - \frac{1}{2n}$ $b_n = \frac{1}{n+1} + \frac{1}{n+2} + \frac{1}{n+3} + \ldots + \frac{1}{2n}$ ***utilizând pointeri.***
3. ***Sa se scrie un program care realizează citirea şi afişarea elementelor unei matrici diagonale, a unei matrici triunghiulare superioare şi a unei matrici tridiagonale, utilizând pointeri.***

4. ***Să se scrie un program care determină dacă o matrice pătratică este pătrat magic, utilizând pointeri. Dacă se consideră că dimensiunea matricii este (nxn), n impar, şi matricea conţine toate numerele naturale de la 1 la n2 astfel încât suma elementelor de pe fiecare linie, coloana şi diagonală este aceeaşi .***
   ***Ex.***
   ***6 1 8***
   ***7 5 3***
   ***2 9 4***

5. ***Sa se scrie un program care realizeaza compararea a doua siruri de caractere prin pointeri.***

6. ***Sa se scrie un program care numara cate vocale şi cate consoane se afla intr-un sir, utilizand pointeri.***

7. ***Sa se scrie un program care citeste de la tastatura un sir de n (n intreg) numere reale şi determina care este maximul si minimul dintre sir, utilizand pointeri.***

8. ***Sa se scrie un program care returneaza lungimea unui sir de caractere utilizand pointeri.***

9. ***Sa se scrie un program care determina prima pozitie la care se gaseste un caracterul c (citit de la tastatura) intr-un sir utilizand pointeri. Daca c nu se afla in sir se returneaza un mesaj de eroare.***

# Capitolul 4
# Tablouri de pointeri.Pointeri la pointeri.
# Pointeri la funcții. Tablou de pointeri la functii

În acest capitol sunt prezentate consideraţii teoretice privind definirea şi utilizarea tablourilor de pointeri, a pointerilor la pointeri şi legătura dintre pointeri şi funcţii: pointeri ca argumente de funcții şi pointeri la funcţii. Sunt prezentate câteva probleme rezolvate cu aceste tipuri de pointeri.

**Consideraţii teoretice**

**Tablourile de pointeri** sunt tablouri care conţin ca și elemente pointeri, adică adrese de memorie.

Formatul de declarare al unui tablou de pointeri este:

- Pentru un **tablou unidimensional de pointeri:**

**tip *nume_pointer[max]**

unde max=dimesiunea maximă a șirului

- Pentru un **tablou bidimensional de pointeri:**

**tip *nume_pointer[max1][max2]**

unde max1/max2=nr maxim de elemente pe linie /coloană

Un tablou de pointeri se poate transmite unei funcții, în mod similar cu transmiterea numelui tabloului fără indici.

**Ex.:**

```
void afis_tab(int *q[])
{int i;
for (i=0;i<10;i++)
printf("%d", *q[i]);        }
// q nu este un pointer la intregi! q este pointer catre un tablou de pointeri la intregi
```

**Pointeri la pointeri**

Declararea unui pointer la pointer se realizează după următorul format:

**tip **nume_pointer;**

**Pointeri ca argumente de funcții**

În C transferul parametrilor este efectuat implicit prin valoare și reprezintă transfer sigur pentru că nu modifică parametri de apel. Dacă se dorește modificarea unei variabile parametru atunci trebuie transmisă funcției adresa variabilei, argumente= adrese, parametri formali=pointeri unde se vor copia aceste adrese .

**Pointeri la funcții**

Formatul de declarare al unui pointer la o funcție este:

**tip (*nume_pointer) (lista_param_formali);**

unde tip = tipul de bază al pointerului , poate fi void dacă nu returnează nici o valoare, sau unul din tipurile char, int, float, double.

nume = numele pointerului la funcție

Formatul de apel al funcției: **(*nume_pointer) (lista_param_efectivi);**

Orice funcție are o localizare (adresă) în memorie ce poate fi atribuită unui pointer. Adresa unei funcții se obține utilizând numele funcției fără paranteze și argumente (în mod analog cu numele tablourilor).

## PROBLEME REZOLVATE

***Ex.1. Programul definește un tablou de pointeri la caracter, care se inițializează cu zilele săptămânii și care afișează pe rând elementele tabloului și șirul de caractere corespunzând elementului al 6-lea din tabloul de pointeri.***

**Varianta in C**

```
#include <stdio.h>
int main()
{char *zile[]={"Luni","Marti","Miercuri","Joi","Vineri","Sambata","Duminica"}; int i;
for(i=0;i<7;i++) printf("zile[%d]=%-10s;zile[%d]=%p\n",i,zile[i], i,zile[i]);
for(i=0;i<7;i++)  printf("*(zile[5]++)=%c\n", *(zile[5]++));
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{const char *zile[]={"Luni","Marti","Miercuri","Joi","Vineri","Sambata","Duminica"};int i;
for(i=0;i<7;i++) cout<<"zile["<<i<<"]="<<zile[i]<<setw(20)<<"    zile["<<i<<"]="<<(void*)zile[i]<<endl;
for(i=0;i<7;i++)  cout<<"*(zile[5]++)="<<*(zile[5]++)<<endl;
return 0;}
```

***Rezultate:***

```
zile[0]=Luni        ;zile[0]=00403024
zile[1]=Marti       ;zile[1]=00403029
zile[2]=Miercuri    ;zile[2]=0040302F
zile[3]=Joi         ;zile[3]=00403038
zile[4]=Vineri      ;zile[4]=0040303C
zile[5]=Sambata     ;zile[5]=00403043
zile[6]=Duminica    ;zile[6]=0040304B
*(zile[5]++)=S
*(zile[5]++)=a
*(zile[5]++)=m
*(zile[5]++)=b
*(zile[5]++)=a
*(zile[5]++)=t
*(zile[5]++)=a
```

***Aplicație:***

Să se modifice programul astfel încât să se afișeze șirul de caractere corespunzător elementului al 3-lea și al 7-lea din tabloul de pointeri.

***Ex.2. Să se scrie un program în care se declară și inițializează un tablou de pointeri la șiruri de caractere și se afișează aceste șiruri în diverse moduri prin intermediul tabloului de pointeri.***

**Varianta in C**

```
#include <stdio.h>
int main()
{char *nume[]={"Alex","Ion","Mircea",NULL};
printf("%c\n", *nume[0]);
printf("%c\n", *(nume[0]+1));
printf("%c\n", *(nume[0]+2));
printf("%c\n", *(nume[0]+3)); printf("\n");
printf("%c\n", *nume[1]);
printf("%c\n", *(nume[1]+1));
printf("%c\n", *(nume[1]+2)); printf("\n");
printf("%c\n", *nume[2]);
printf("%c\n", *(nume[2]+1));
printf("%c\n", *(nume[2]+2));
printf("%c\n", *(nume[2]+3));
printf("%c\n", *(nume[2]+4));
printf("%c\n", *(nume[2]+5)); printf("\n");
printf("%c\n", *(nume[1]+3) );
printf("%s\n", *nume);  printf("%s\n", *nume+1);
printf("%s\n", *(nume+1));  printf("%s\n", *(nume+2));
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{const char *nume[]={"Alex","Ion","Mircea",NULL};
cout<<*nume[0]<<endl;
cout<<*(nume[0]+1)<<endl;
cout<<*(nume[0]+2)<<endl;
cout<<*(nume[0]+3)<<endl<<endl;
cout<<*nume[1]<<endl;
cout<<*(nume[1]+1)<<endl;
cout<<*(nume[1]+2)<<endl<<endl;
cout<<*nume[2]<<endl;
cout<<*(nume[2]+1)<<endl;
cout<<*(nume[2]+2)<<endl;
cout<<*(nume[2]+3)<<endl;
cout<<*(nume[2]+4)<<endl;
cout<<*(nume[2]+5)<<endl<<endl;
cout<<*(nume[1]+3)<<endl;
cout<<*nume<<endl;  cout<<*nume+1<<endl;
cout<<*(nume+1)<<endl;  cout<<*(nume+2)<<endl;
return 0;}
```

***Rezultate:***

```
A
l
e
x

I
o
n

M
i
r
c
e
a


Alex
lex
Ion
Mircea
```

***Aplicație:***

Să se modifice programul astfel încât să se afișeze *toate șirurile de caractere, caracter cu caracter printr-o buclă for, while, sau  do while .*

***Ex.3: Programul este un exemplu de declarare și afișare a unui pointer la pointer.***

**Varianta in C**

```
#include <stdio.h>
int main()
{ int x,*p,**q; x=10; p=&x; q=&p;printf("x=%d\n", x);
printf("*p este pointer la variabila x\n");
printf("\n**q este pointer la variabila pointer *p\n");
printf("\nvaloarea stocata la adresa indicata de \n");
printf("pointerul p este adresa lui x:%p\n", *p);
printf("valoarea stocata la adresa indicata de\n");
printf("pointerul q este chiar x:%d", **q);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{ int x,*p,**q; x=10; p=&x; q=&p; cout<<"x="<<x<<endl;
cout<<"*p este pointer la variabila x"<< endl;
cout<<"\n**q este pointer la variabila pointer *p"<<endl;
cout<<"\nvaloarea stocata la adresa indicata de "<<endl;
cout<<"pointerul p este adresa lui x:"<<(void*)*p<<endl;
cout<<"valoarea stocata la adresa indicata de"<<endl;
cout<<"pointerul q este chiar x:"<<**q<<endl;return 0;}
```

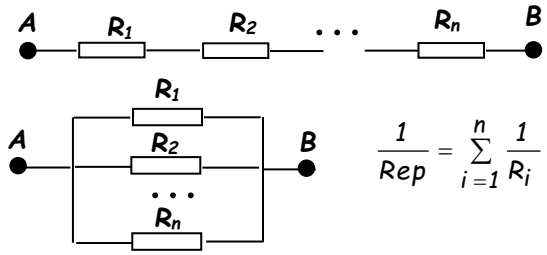
***Rezultate:***

***Aplicație:***

```
x=10
*p este pointer la variabila x

**q este pointer la variabila pointer *p

valoarea stocata la adresa indicata de
pointerul p este adresa lui x:0000000A
valoarea stocata la adresa indicata de
pointerul q este chiar x:10
```

Să se modifice programul astfel încât să se afișeze rezultatul expresiei $2x^2+1$ utilizând pointerul la pointer **q.

***Ex.4: Programul realizează calculul și afișarea valorii rezistenței echivalente pentru conectarea în serie și respectiv în paralel a rezistențelor, a căror număr și valori se citesc de la tastatură, utilizând un pointer la o funcție.***




$$Res = \sum_{i=1}^{n} R_i$$




$$\frac{1}{Rep} = \sum_{i=1}^{n} \frac{1}{R_i}$$

**Varianta in C**

```
#include <stdio.h>
#include <ctype.h>
#include <conio.h>
double calcul(double *, int, double(*)(double*,int));
double serie(double *,int); //calcul rezistenta serie
double paralel(double *,int); //calcul rezistenta paralel
int main()
{int i,n;
double rez[10],rec;
char c;
printf("cate rezistente sunt? n="); scanf("%d", &n);
printf("Valorile rezistentelor sunt:\n");
for (i=0;i<n;i++)
{printf("rez[%d]=", i); scanf("%lf",&rez[i]);  }
printf("legare in serie(s) sau paralel(p):");
scanf("\n%c", &c);
if(toupper(c)=='S') rec=calcul(rez,n,serie);
else if(toupper(c)=='P') rec=calcul(rez,n, paralel);
printf("\nRezistenta echivalenta este:%lf", rec);
return 0;}
double calcul(double *rez,int n,double (*p)(double *,int))
{return(p(rez,n));}
double serie(double *rez, int n)
{double s=0.;
while(n) s+=rez[--n];
return(s);}
double paralel(double *rez, int n)
{double s=0.;
while(n)
s+=1./rez[--n];
return(1./s);}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
double calcul(double *, int, double(*)(double*,int));
double serie(double *,int); //calcul rezistenta serie
double paralel(double *,int); //calcul rezistenta paralel
int main()
{int i,n;
double rez[10],rec;
char c;
cout<<"cate rezistente sunt? n="; cin>>n;
cout<<"Valorile rezistentelor sunt:"<<endl;
for (i=0;i<n;i++)
{cout<<"rez["<<i<<"]="; cin>>rez[i];  }
cout<<"legare in serie(s) sau paralel(p):";
cin>>c;
if(toupper(c)=='S') rec=calcul(rez,n,serie);
else if(toupper(c)=='P') rec=calcul(rez,n, paralel);
cout<<endl<<"Rezistenta echivalenta este:"<<rec;
return 0;}
double calcul(double *rez,int n,double (*p)(double *,int))
{return(p(rez,n));}
double serie(double *rez, int n)
{double s=0.;
while(n) s+=rez[--n];
return(s);}
double paralel(double *rez, int n)
{double s=0.;
while(n)
s+=1./rez[--n];
return(1./s);}
```

***Rezultate:***

```
cate rezistente sunt? n=4
Valorile rezistentelor in ohmi sunt:
rez[0]=1
rez[1]=1
rez[2]=1
rez[3]=1
legare in serie(s) sau paralel(p):s

Rezistenta echivalenta este:4.000000
```

```
cate rezistente sunt? n=4
Valorile rezistentelor in ohmi sunt:
rez[0]=1
rez[1]=1
rez[2]=1
rez[3]=1
legare in serie(s) sau paralel(p):p

Rezistenta echivalenta este:0.250000
```

***Aplicație:***

Să se modifice programul astfel încât să se calculeze și afișeze valorile capacităților unor condensatoare conectate în serie .

***Ex.5: Programul afișează valorile a 3 funcții trigonometrice: sin(π/6), cos(π/6), tg(π/4) utilizând pointeri la funcțiile sin(), cos() si tan() (din biblioteca <math.h>).***

**Varianta in C**

```
#include <stdio.h>
#include <math.h>
#define PI 3.1415926
int main()
{double (*ps)(double), (*pc)(double), (*pt)(double);
ps=sin; pc=cos; pt=tan;
printf("\n\t pointer la sin = %p", ps);
printf("\n\t pointer la cos = %p", pc);
printf("\n\t pointer la tg  = %p", pt);
printf("\n\napelul functiilor trigonometrice\nprin pointeri
la aceste functii\n");
printf("\n\t sin(pi/6) = %.2lf", (*ps)(PI/6) );
printf("\n\t cos(pi/6) = %.2lf", (*pc)(PI/6) );
printf("\n\t  tg(pi/4) = %.2lf\n", (*pt)(PI/4) );
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <math.h>
#define PI 3.1415926
using namespace std;
int main()
{double (*ps)(double), (*pc)(double), (*pt)(double);
ps=sin; pc=cos; pt=tan;
cout<<"\n\t pointer la sin = "<<(void*)ps<<endl;
cout<<"\t pointer la cos = "<<(void*)pc<<endl;
cout<<"\t pointer la tg  = "<<(void*)pt<<endl;
cout<<"\napelul functiilor trigonometrice"<<endl<<"prin
pointeri la aceste functii"<<endl;
cout<<"\t sin(pi/6) = "<<(*ps)(PI/6)<<endl;
cout<<"\t cos(pi/6) = "<<(*pc)(PI/6)<<endl;
cout<<"\t  tg(pi/4) = "<<(*pt)(PI/4)<<endl;
return 0;}
```

***Rezultate:***

```
        pointer la sin = 00401C40
        pointer la cos = 00401C48
        pointer la tg  = 00401C50

apelul functiilor trigonometrice
prin pointeri la aceste functii

        sin(pi/6) = 0.50
        cos(pi/6) = 0.87
         tg(pi/4) = 1.00
```

***Aplicație:***

*Să se modifice programul astfel încât să se* utilizeze și pointeri la alte funcții din biblioteca <math.h>

***Ex.6: Programul compară 2 șiruri de caractere utilizând un pointer la funcția strcmp din biblioteca <string.h> sau <cstring> (pentru C++).***

**Varianta in C**

```
#include <stdio.h>
#include <string.h>
void compara(char *a,char *b, int(*comp)(const char *,
const char *));
int main()
{char s1[80],s2[80];
int (*p) (const char *, const char *);p=strcmp;
printf("Introduceti sirul 1:");  gets(s1);
printf("Introduceti sirul 2:");  gets(s2);
compara(s1,s2,p); return 0;}
void compara(char *a,char *b, int(*comp)(const char *,
const char *))
{printf("Testeaza egalitatea dintre siruri:\n");
if(!(*comp)(a,b)) printf("SIRURILE SUNT EGALE!\n");
else printf("SIRURILE SUNT DIFERITE!\n"); }
```

**Varianta in C++**

```
#include <iostream>
#include<stdio.h>
#include<string.h>
using namespace std;
void compara(char *a,char *b, int(*comp)(const char *, const
char *));
int main()
{char s1[80],s2[80];
int (*p) (const char *, const char *);p=strcmp;
cout<<"Introduceti sirul 1:";  gets(s1);
cout<<"Introduceti sirul 2:";  gets(s2);
compara(s1,s2,p); return 0;}
void compara(char *a,char *b, int(*comp)(const char *, const
char *))
{cout<<"Testeaza egalitatea dintre siruri:"<<endl;
if(!(*comp)(a,b)) cout<<"SIRURILE SUNT EGALE!"<<endl;
else cout<<"SIRURILE SUNT DIFERITE!"<<endl; }
```

***Rezultate:***

```
Introduceti sirul 1:Programare C/C++
Introduceti sirul 2:Programare C
Testeaza egalitatea dintre siruri:
SIRURILE SUNT DIFERITE!
```

```
Introduceti sirul 1:Programare C/C++
Introduceti sirul 2:Programare C/C++
Testeaza egalitatea dintre siruri:
SIRURILE SUNT EGALE!
```

***Aplicație:***

*Să se modifice programul astfel încât să se utilizeze pointeri și la alte funcții din biblioteca <string.h>, de ex: la strcat(), strlen(), strchr(), strstr(), etc.*

***Ex.7. Programul realizează sortarea în ordine alfabetică a unui tablou de pointeri la șiruri de caractere. Șirurile de caractere reprezintă numele și prenumele unor persoane.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void sort(char*a[],int n);
int main() {char *sir[10]={"Popescu Alin","Morar Calin","Arcan Silviu","Zidane Zinedin"};
int i; sort(sir,4);
for (i=0;i<4;i++)  printf("%s\n",sir[i]); return 0;}
void sort(char*a[],int n)
{int i,j; char *aux;
for (i=0;i<n;i++) for (j=0;j<n;j++)   if(strcmp(a[i],a[j])<0)    {aux=a[i];    a[i]=a[j]; a[j]=aux;}}
```

**Varianta in C++**

```
#include <iostream>
#include<string.h>
using namespace std;
void sort(const char*a[],int n);
int main()
{ const char *sir[10]={"Popescu Alin","Morar Calin","Arcan Silviu","Zidane Zinedin"};
int i; sort(sir,4);
for (i=0;i<4;i++) cout<<sir[i]<<endl; return 0;}
void sort(const char*a[],int n) {int i,j; const char *aux;
for (i=0;i<n;i++) for (j=0;j<n;j++)    if(strcmp(a[i],a[j])<0)    {aux=a[i];    a[i]=a[j]; a[j]=aux;}}
```

***Rezultate:***

```
Arcan Silviu
Morar Calin
Popescu Alin
Zidane Zinedin
```

***Aplicație:***

*Să se modifice programul astfel încât inițializarea șirurilor să se realizeze prin citire de la tastatură .*

***Ex.8. Programul calculeaza si afiseaza valoarea sin(x), unde x real citit de la tastatura prin apelul functiei sin() din <math.h> si prin descompunerea in serie Taylor:***

$$sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^{2i+1}}{(2i+1)!}$$

**Varianta in C**

```
#include <stdio.h>
#include <math.h>
double fact(int j)
{double t; int k; t=1;
for (k=1;k<=j;k++)t=t*k;
return t;}
int main()
{int i,n; double x,fx;
double (*p)(double, double); p=pow;
double (*q) (int); q=fact;
printf("x=");scanf("%lf",&x);
printf("n=");scanf("%d",&n);
fx=x;
for (i=1;i<=n;i++)
  //fx+=pow(-1,i)*pow(x,(2*i+1))/fact(2*i+1);
  fx+=(*p)(-1,i)*(*p)(x,(2*i+1))/(*q)(2*i+1);
printf("Val functiei sin(x) calculata\ncu descompunerea in
serie este:\n");
printf("sin1(%lf)=%.20lf\n", x, fx);
printf("\nval functiei sin(x) calculata\ncu functia sin din
<math.h> este:\n");
printf("sin2(%lf)=%.20lf\n", x ,sin(x));
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <math.h>
using namespace std;
double fact(int j)
{double t; int k; t=1;
for (k=1;k<=j;k++)t=t*k;
return t;}
int main()
{int i,n; double x,fx;
double (*p)(double, double); p=pow;
double (*q) (int); q=fact;
cout<<"x=";cin>>x;
cout<<"n=";cin>>n;
fx=x;
for (i=1;i<=n;i++)
  //fx+=pow(-1,i)*pow(x,(2*i+1))/fact(2*i+1);
  fx+=(*p)(-1,i)*(*p)(x,(2*i+1))/(*q)(2*i+1);
cout<<"Val functiei sin(x) calculata\ncu descompunerea
in serie este:"<<endl;
cout<<"sin1("<<x<<")="<<fx<<endl;
cout<<endl<<"val functiei sin(x) calculata"<<endl<<"cu
functia sin din <math.h> este:"<<endl;
cout<<"sin2("<<x<<")="<<sin(x)<<endl;
return 0;}
```

***Rezultate:***

```
x=0.5
n=5
Val functiei sin(x) calculata
cu descompunerea in serie este:
sin1(0.500000)=0.47942553860418341000

val functiei sin(x) calculata
cu functia sin din <math.h> este:
sin2(0.500000)=0.47942553860420301000
```

***Aplicație:***

*Să se modifice programul astfel încât sa se calculeze cos(x), cu descompunerea in serie Taylor:*

$$\cos(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!}$$

***Ex.9. Programul utilizeaza un tablou de pointeri la functii predefinite din <math.h> prin care se calculeaza si afiseaza valorile: $e^3$, log(10), $\sqrt{9}$, sin(π/6), cos(π/6):***

**Varianta in C**

```
#include <stdio.h>
#include <math.h>
#define pi 3.14
int main ()
{double (*tabfun[])(double) = {sin, cos, exp, log, sqrt};
printf("exp(3)=%lf", (*tabfun[2])(3));
printf("\nlog(10)=%lf", (*tabfun[3])(3));
printf("\nsqrt(9)=%lf", (*tabfun[4])(9));
printf("\nsin((pi/6)=%lf", (*tabfun[0])(pi/6));
printf("\ncos((pi/6)=%lf", (*tabfun[1])(pi/6));
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <math.h>
#define pi 3.14
using namespace std;
int main ()
{double (*tabfun[])(double) = {sin, cos, exp, log, sqrt};
cout<<"exp(3)="<<(*tabfun[2])(3);
cout<<"\nlog(10)="<<(*tabfun[3])(3);
cout<<"\nsqrt(9)="<<(*tabfun[4])(9);
cout<<"\nsin((pi/6)="<<(*tabfun[0])(pi/6);
cout<<"\ncos((pi/6)="<<(*tabfun[1])(pi/6);
return 0;}
```

***Rezultate:***

```
exp(3)=20.085537
log(10)=1.098612
sqrt(9)=3.000000
sin((pi/6)=0.499770
cos((pi/6)=0.866158
```

***Aplicație:***

*Să se modifice programul astfel încât sa se calculeze valoarea absoluta a lui x cu functia abs(x)*

## PROBLEME PROPUSE

1. ***Să se scrie programul în care se definește un tablou de pointeri la șiruri de caractere, care se inițializează cu numele lunilor anului și se afișează pe rând elementele tabloului precum și numărul de zile corespunzătoare fiecărei luni calendaristice.***

2. ***Să se scrie un program în care se utilizează pentru calculul valorilor unor funcții matematice, pointerii la aceste funcții predefinite in <math.h> sau <cmath>: exp(), abs(), log(), pow(), sqrt(),round(), etc.***

3. ***Să se scrie un program care realizează sortarea mai multor șiruri de numere reale prin intermediul unui tablou de pointeri.***

4. ***Să se scrie un program care declara un tablou de pointeri la functii predefinite din <math.h> sau <cmath> prin care se calculeaza si afiseaza valorile unor functii pentru cateva valori constante: asin(), acos(), atan(), atan2(), sinh(), cosh(), tanh().***

5. ***Să se scrie un program care calculeaza si afiseaza valorile functiilor $e^x$,1/tan(x), log(1+x), $\sin^2(x)$, cosh(x) prin descompuneri in serii Taylor dupa formulele de mai jos si afiseaza si valorile obtinute prin apelul functiilor din <math.h> sau <cmath> :***

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \qquad \tan^{-1}(x) = \sum_{k=0}^{\infty} \frac{(-1)^k\, x^{1+2k}}{1+2k} \quad \text{for } |x| < 1 \qquad \log(1+x) = -\sum_{k=1}^{\infty} \frac{(-1)^k\, x^k}{k} \quad \text{for } |x| < 1$$

,

$$\sin^2(x) = -\sum_{k=1}^{\infty} \frac{(-1)^k\, 2^{-1+2k}\, x^{2k}}{(2k)!} \qquad \cosh(x) = \sum_{k=0}^{\infty} \frac{x^{2k}}{(2k)!}$$

# Capitolul 5
# Alocarea dinamică a memoriei

În acest capitol sunt prezentate consideraţii teoretice privind alocarea dinamică a memoriei şi utilizarea funcţiilor predefinite în C : malloc(), calloc(), realloc, free(), etc. şi a noilor funcţii new şi delete introduse în C++.

## CONSIDERAŢII TEORETICE

Cele mai importante **funcţii** pentru **alocarea dinamică a memoriei sunt :**

**a) în C** : malloc(), realloc(), calloc(), free(), etc.

**b) în C++:** toate cele din C și adițional se utilizează operatorii new şi delete.

**Funcţia malloc**() are următorul prototip:

**void *malloc(unsigned n);**

unde:

**n** = numărul de octeţi de memorie ce va fi alocat dinamic.

La apelare, funcţia malloc() returnează un pointer spre primul octet al regiunii de memorie alocate în memoria (heap) liberă. Funcţia alocă în memoria heap o zonă contiguă de n octeţi = numărul de octeţi specificat în paranteza funcţiei malloc().

Dacă nu este suficientă memorie disponibilă atunci malloc() returnează “null”.
Bibliotecile ce trebuie incluse în program pentru utilizarea acestei funcţii sunt: <stdlib.h> şi <malloc.h>

Alocarea unei zone de memorie contigue, neinițializate se realizează diferit în C sau C++ după cum este ilustrat în cele ce urmează.

**a) secvenţa în C**

```
tip *p;  p=malloc(n*sizeof(tip);
```

**b) secvenţa în C++**

```
  tip *p; p=(tip*)malloc(n*sizeof(tip);
```

În C nu este necesară specificarea tipului pentru a atribui lui p valoarea returnată de malloc(), deoarece un pointer de tip *void este automat convertit în tipul pointerului din partea stângă a atribuirii. În C++ este obligatorie specificarea explicită de tip atunci când se atribuie un pointer de tip *void altui tip de pointer. O altă observație importantă legată de utilizarea funcţiei malloc() este că zona de memorie alocată nu este inițializată.

Pentru evitarea erorilor trebuie inclus în program un test din care să rezulte dacă există memorie disponibilă sau nu (“null”). Testarea valorii returnate de funcţia malloc(), deci a existenţei unei zone de memorie libere, se poate realiza ca şi în exemplu următor:

**Ex.:** alocarea dinamica a unei zone de memorie pentru 100 de numere întregi =100*4 octeți=400 octeți. Se testează dacă există memorie liberă, iar dacă nu există se afişează mesajul de eroare „memorie insuficienta”:

```
int *ip = malloc(100 * sizeof(int));
if(ip == NULL)
   {printf("memorie insuficienta\n");
    exit(1)     }
```

O altă funcţie importantă este **funcţia free()** reprezentând funcţia opusă funcţiei malloc() care are ca efect eliberarea memoriei alocate dinamic anterior cu funcţia malloc().

**Funcţia free()** are următorul prototip: **void *free(void *p);**

unde: **p** este un pointer spre memoria alocată anterior cu funcţia malloc().

Bibliotecile ce trebuie incluse în program pentru utilizarea funcţiei free() sunt : <stdlib.h> si <malloc.h>. Un exemplu de utilizare a funcţiei free() este prezentat mai jos.

**Ex.:** Alocarea și dezalocarea unei zone de memorie pentru 50 de numere întregi:

```
int *p;
p= (int *)malloc(50*sizeof(int));
…
free(*p);
```

**Functia realloc()** are următorul prototip: **void *realloc(void *p, unsigned n);**

unde: **p** este un pointer spre memoria realocată dinamic

Funcţia realloc() realocă dinamic zona de memorie specificată prin numărul de octeți, n, spre care indică pointerul p .
Bibliotecile ce trebuie incluse în program pentru utilizarea acestei funcţii sunt: <stdlib.h> şi <malloc.h>

**Ex:** Iniţial se alocă dinamic memorie pentru 50 de numere întregi . Ulterior se realocă dinamic o zona de memorie pentru 100 de numere întregi , acelaşi pointer p conţinând adresa de început a zonei alocate dinamic:

```
int *p;
p= (int *)malloc(50*sizeof(int));
…
p= (int *)realloc(p, 100 * sizeof(int));
```

Dacă realocarea nu este posibilă din lipsa de memorie libera atunci realloc() returnează ‘NULL’, altfel funcția returnează un pointer spre zona de memorie realocată.

**Funcţia calloc()** este **funcția de alocare și inițializare cu 0 a memoriei** şi are următorul prototip:

**void *calloc(unsigned nrelem, unsigned dimelem);**

unde
**nrelem=**numărul de obiecte pentru care se rezervă memorie
**dimelem=** numărul de octeţi de memorie ce va fi alocat pentru fiecare element

Efectul apelării acestei funcţii este alocarea unei zone de memorie de dimensiune nrelem*dimelem, în memoria heap care se inițializează cu 0, funcţia returnând un pointer la zona de memorie sau 'null' dacă alocarea nu s-a putut realiza.

Bibliotecile ce trebuie incluse în program pentru utilizarea funcţiei calloc sunt: <stdlib.h> şi <malloc.h>

Un exemplu de utilizare a acestei funcţii este prezentat în continuare:

**Ex.:** alocarea unei zone de memorie pentru 100 de numere întregi iniţializate cu 0 :

```
int *p;
p = (int *) calloc(100, sizeof(int));
```

Eliberarea zonei alocate dinamic cu funcţia calloc() se poate face de asemenea utilizând funcţia free() .

Alocarea dinamică a memoriei în C++ se realizează cel mai frecvent prin intermediul operatorilor **new** şi **delete.**

Formatul de declarare al operatorului **new** este :

**new tip** sau **new (tip)** sau
**new tip (expresie)** sau **new tip[exp]**

unde: **tip** = numele unui tip predefinit sau definit de utilizator
**expresie** = expresie a cărei valoare inițializează zona de memorie alocată prin new
**exp** = expresie de tip int folosită la alocarea dinamică a tablourilor.

Operatorul **new** permite alocarea memoriei în zona heap şi are ca şi valoare adresa de început a zonei de memorie alocate sau 0 dacă alocarea eşuează . Operatorul **delete** este utilizat pentru eliberarea zonei de memorie alocată cu operatorul new. Un exemplu de utilizare al operatorilor new și delete este prezentat mai jos.

**Ex.:**

```
tip *p;
p=new tip;
…
delete p;
```

Un alt exemplu de utilizare al acestor operatori, cu alocarea/dealocarea memoriei pentru un tablou este prezentat mai jos:

**Ex.:**

```
tip *p=new tip[exp];        //alocarea unui tablou
delete [exp] p;             //eliberarea memoriei ocupate de tablou
```

## PROBLEME REZOLVATE

***Ex.1: Programul realizează alocarea dinamică a memoriei pentru un șir de n numere întregi, unde n este un număr întreg citit de la tastatură și calculează suma și produsul elementelor șirului, precum și elementul minim și maxim din sir.***

**Varianta in C**

```
#include <stdio.h>
#include <malloc.h> //pentru functia malloc()
int main()
{int *p, n,i,sum=0, prod=1, max, min;
printf("Introduceti n intreg pozitiv: ");scanf("%d", &n);
//alocarea dinamica
p=(int*)malloc(n*sizeof(int));
if(p==NULL) printf("\n eroare de alocare!");
printf("Introduceti %d nr. intregi:\n", n);
for (i=0;i<n;i++)
    {printf("nr %d:",i+1); scanf("%d", p+i); }
//afisarea sirului de nr
printf("ati introdus nr:");
max=min=*p;
for (i=0;i<n;i++)
  {printf("%d ",*(p+i));
  sum+=*(p+i);prod*=*(p+i);
  if (max<*(p+i)) max=*(p+i);
  if (min>*(p+i)) min=*(p+i);}
printf("\nSuma elementelor =%d\n", sum);
printf("Produsul elementelor=%d\n",prod);
printf("maxim=%d\n", max);
printf("minim=%d\n", min);
//eliberarea memoriei
free (p);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <malloc.h>
using namespace std;
int main()
{int *p, n,i,sum=0, prod=1, max, min;
cout<<"Introduceti n intreg pozitiv: ";cin>>n;
//alocarea dinamica
p=new int[n];
if(p==NULL) cout<<endl<<" eroare de alocare!";
cout<<"Introduceti"<<n<<"nr. intregi:"<<endl;
for (i=0;i<n;i++)
    {cout<<"nr"<<i+1<<":";
    cin>>*(p+i); }
//afisarea sirului de nr
cout<<"ati introdus nr:";
max=min=*p;
for (i=0;i<n;i++)
  {cout<<*(p+i)<<"  ";
  sum+=*(p+i);prod*=*(p+i);
  if (max<*(p+i)) max=*(p+i);
  if (min>*(p+i)) min=*(p+i);}
cout<<"\nSuma elementelor ="<<sum<<endl;
cout<<"Produsul elementelor="<<prod<<endl;
cout<<"maxim="<<max<<endl;
cout<<"minim="<<min<<endl;
//eliberarea memoriei
delete p;
return 0;}
```

***Rezultate:***

```
Introduceti n intreg pozitiv: 5
Introduceti 5 nr. intregi:
nr 1:5
nr 2:10
nr 3:9
nr 4:4
nr 5:3
ati introdus nr:5 10 9 4 3
Suma elementelor =31
Produsul elementelor=5400
maxim=10
minim=3
```

***Aplicație:***

Să se rescrie programul in C++ astfel încât să se utilizeze operatorii malloc și free pentru alocarea dinamică a memoriei și efectuarea calculelor și afișării rezultatelor.

***Ex2.: Programul realizează scăderea a două matrici pătratice de aceeași dimensiune n, unde n este un număr întreg introdus de la tastatură, utilizând alocarea dinamică a memoriei pentru cele două matrici .***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
#include<malloc.h>
int main()
{int n,i,j, *p,*q;
printf("Introduceti dimensiunea matricilor patratice, n=");
scanf("\n%d",&n);
if((p=(int*)malloc(n*n*sizeof(int)))==NULL)
{ printf("eroare alocare memorie matrice 1\n");  }
if((q=(int*)malloc(n*n*sizeof(int)))==NULL)
          {  printf("eroare alocare memorie matrice 2\n"); }
//Citirea elementelor primei matrici
printf("\nIntroduceti elementele matricii 1:\n");
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
     { printf("a[%d][%d]=",i,j); scanf("%d",p+i*n+j);  }
//tiparirea elementelor primei matrici
printf("Matrice 1:");
for(i=0;i<n;i++)    { printf("\n");
   for(j=0;j<n;j++)  printf("%4d",*(p+i*n+j));}
//citirea elementelor matricii a doua
printf("\nIntroduceti elementele matricii 2:\n");
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
          { printf("b[%d][%d]=",i,j);  scanf("%d",q+i*n+j);}
//tiparirea elementelor matricii a doua
printf("Matrice 2:");
for(i=0;i<n;i++){ printf("\n");
   for(j=0;j<n;j++)   printf("%4d ",*(q+i*n+j)); }
printf("\n");
//scaderea matricilor si tiparirea rezultatului
printf("rezultatul scaderii este:\n");
for(i=0;i<n;i++){  printf("\n");
   for(j=0;j<n;j++)  printf("%4d ",*(p+i*n+j)-*(q+i*n+j));}
free(p);free(q);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
#include<malloc.h>
using namespace std;
int main()
{int n,i,j, *p,*q;
cout<<"Introduceti dimensiunea matricilor patratice, n=";
cin>>n;
p=new int[n*n];
if(p==NULL)
 cout<<"eroare alocare memorie matrice 1"<<endl;
 q=new int[n*n];
if(q==NULL)
 cout<<"eroare alocare memorie matrice 2"<<endl;
//Citirea elementelor primei matrici
cout<<"\nIntroduceti elementele matricii 1:"<<endl;
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
     { cout<<"a["<<i<<"]["<<j<<"]=";
       cin>>*(p+i*n+j);  }
//tiparirea elementelor primei matrici
cout<<"Matrice 1:";
for(i=0;i<n;i++)    { cout<<endl;
   for(j=0;j<n;j++)  cout<<*(p+i*n+j)<<"\t";}
//citirea elementelor matricii a doua
cout<<"\nIntroduceti elementele matricii 2:"<<endl;
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
  { cout<<"b["<<i<<"]["<<j<<"]=";  cin>>*(q+i*n+j);}
//tiparirea elementelor matricii a doua
cout<<"Matrice 2:";
for(i=0;i<n;i++){ cout<<endl;
   for(j=0;j<n;j++)   cout<<*(q+i*n+j)<<"\t"; }
cout<<endl;
//scaderea matricilor si tiparirea rezultatului
cout<<"rezultatul scaderii este:"<<endl;
for(i=0;i<n;i++){  cout<<endl;
   for(j=0;j<n;j++)
    cout<<*(p+i*n+j)-*(q+i*n+j)<<"\t";}
delete p;delete q;
return 0;}
```

***Rezultate:***

```
Introduceti elementele matricii 1:
a[0][0]=10
a[0][1]=10
a[1][0]=10
a[1][1]=10
Matrice 1:
  10  10
  10  10
Introduceti elementele matricii 2:
b[0][0]=5
b[0][1]=4
b[1][0]=3
b[1][1]=2
Matrice 2:
   5    4
   3    2
rezultatul scaderii este:

   5    6
   7    8
```

***Aplicație:***

Să se modifice programul astfel încât să se calculeze și afișeze matricea rezultată din expresia:
10*matrice1-2*matrice2.

***Ex.3: Programul realizează citirea și afișarea unei matrici cu elemente întregi, de dimensiune nxn, unde n este număr întreg introduse de la tastatură, și realizează înmulțirea sa cu o constantă întreagă citită de la tastatură. După afișarea rezultatului se cere să se realoce dinamic memorie pentru un șir de n numere întregi ce vor fi citite de la tastatură și apoi afișate la puterea a 3-a.***

**Varianta in C**

```
#include <stdio.h>
#include <malloc.h>
#include <math.h>
int main()
{int n,i,j,*p,x,y;
printf("Introduceti dimensiunea matricilor patratice, n=");
scanf("\n%d",&n);
//alocarea dinamica si testare pointer
if(p=(int*)malloc(n*n*sizeof(int)))
{printf("Introduceti elementele matricii:\n");
for (i=0;i<n;i++) for (j=0;j<n;j++)
{printf("a[%d][%d]=",i,j);  scanf("%d", p+i*n+j);}
//afisarea matricii
printf("matricea este:\n");
for (i=0;i<n;i++)
{for (j=0;j<n;j++)
printf("%4d",*(p+i*n+j));  printf("\n");}
printf("introduceti un nr intreg x="); scanf("%d",&x);
printf("matricea inmultita cu %d este:\n",x);
for (i=0;i<n;i++)
{for (j=0;j<n;j++)
      printf("%4d",(*(p+i*n+j)*x)); printf("\n"); }}
else {printf("eroare de alocare memorie!");}
printf("\nIntroduceti un sir, de dimensiune n= ");
scanf("%d", &n);
p=(int*)realloc(p,n*sizeof(int));
if(p==NULL) printf("\n eroare de alocare!");
printf("Introduceti elementele sirului:\n");
for (i=0;i<n;i++) {printf("a[%d]=",i); scanf("%d", p+i);}
printf("sirul la puterea 3-a:\n");
for (i=0;i<n;i++) {y=pow(*(p+i),3);
printf("\na[%d]^3=%d",i,y);} printf("\n");
free (p); return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <malloc.h>
#include <math.h>
using namespace std;
int main()
{int n,i,j,*p,x,y;
cout<<"Introduceti dimensiunea matricilor patratice, n=";
cin>>n;
//alocarea dinamica si testare pointer
p=(int*)malloc(n*n*sizeof(int));
if(p!=NULL) {cout<<"Introduceti elementele matricii:" <<
endl;
for (i=0;i<n;i++) for (j=0;j<n;j++)
    {cout<<"a["<<i<<"]["<<j<<"]=";  cin>>*(p+i*n+j);}
//afisarea matricii
cout<<"matricea este:"<<endl;
for (i=0;i<n;i++)  {for (j=0;j<n;j++)
   cout<<*(p+i*n+j)<<"\t"; cout<<endl;}
cout<<"introduceti un nr intreg x="; cin>>x;
cout<<"matricea innultita cu "<<x<<" este:"<<endl;
for (i=0;i<n;i++) {for (j=0;j<n;j++)
    cout<<*(p+i*n+j)*x<<"\t"; cout<<endl;}}
else {cout<<"eroare de alocare memorie!";}
cout<<"\nIntroduceti un sir, de dimensiune n= "; cin>>n;
p=(int*)realloc(p,n*sizeof(int));
if(p==NULL) cout<<"\n eroare de alocare!";
cout<<"Introduceti elementele sirului:"<<endl;
for (i=0;i<n;i++) {cout<<"a["<<i<<"]="; cin>>*(p+i);}
cout<<"sirul la puterea 3-a:"<<endl;
for (i=0;i<n;i++) {y=pow(*(p+i),3);
   cout<<endl<<"a["<<i<<"]^3="<<y;}
cout<<endl;
free (p); return 0;}
```

***Rezultate:***

```
a[0][0]=1
a[0][1]=2
a[1][0]=3
a[1][1]=4
matricea este:
1       2
3       4
introduceti un nr intreg x=2
matricea innultita cu 2 este:
2       4
6       8

Introduceti un sir, de dimensiune n= 5
Introduceti elementele sirului:
a[0]=10
a[1]=10
a[2]=2
a[3]=5
a[4]=1
sirul la puterea 3-a:

a[0]^3=1000
a[1]^3=1000
a[2]^3=8
a[3]^3=124
a[4]^3=1
```

***Aplicație:***

Să se modifice programul astfel încât să se calculeze și afișeze maximul și minimul din șirul realocat dinamic.

***Ex.4. Programul realizează înmulțirea a două matrici de numere întregi, prima matrice de dimensiune mxn (m linii și n coloane) iar cea de-a doua matrice de nxl (n linii și l coloane) utilizând alocarea dinamică a memoriei.***

**Varianta in C**

```
#include <stdio.h>
#include <malloc.h>
int main()
{int n,m,i,s,j,k,l,*p,*q, a[10][10],b[10][10];
p=&a[0][0]; q=&b[0][0];
printf("nr. de linii prima matrice,m=");scanf("%d",&m);
printf("\nnr. de coloane prima matrice=\nnr. de linii a doua
matrice,n=");
scanf("%d",&n);
printf("nr de coloane la a doua matrice l=");scanf("%d",&l);
if((p=(int*)malloc(m*n*sizeof(int)))==NULL)
{printf("nu exista suficienta memorie\n");}
printf("\n Prima matrice:\n");
for (i=0;i<m;i++)
  for (j=0;j<n;j++)
  { printf("a[%d][%d]=",i+1,j+1); scanf("%d",*(a+i)+j);   }
if((q=(int*)malloc(n*l*sizeof(int)))==NULL)
{printf("nu exista suficienta memorie\n");}
printf("\n A doua matrice:\n");
for (i=0;i<n;i++)   for (j=0;j<l;j++)
  {  printf("b[%d][%d]=",i+1,j+1);scanf("%d",*(b+i)+j);  }
printf("\nMatricea produs este");
for (i=0;i<m;i++)  {  printf("\n");
     for (j=0;j<l;j++)   { s=0;
       for (k=0;k<n;k++)
       s+=*(*(a+i)+k)* (*(*(b+k)+j));  printf("%4d  ",s);}}
printf("\n");
free(p);free(q);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <malloc.h>
using namespace std;
int main()
{int n,m,i,s,j,k,l,*p,*q,a[10][10],b[10][10];
p=&a[0][0]; q=&b[0][0];
cout<<"nr. de linii prima matrice,m=";cin>>m;
cout<<"\nnr. de coloane prima matrice=\nnr. de linii a
doua matrice,n="; cin>>n;
cout<<"nr de coloane la a doua matrice l=";cin>>l;
p=new int[m*n];
if(p==NULL)
{cout<<"nu exista suficienta memorie"<<endl;}
cout<<endl<<"Prima matrice:"<<endl;
for (i=0;i<m;i++)   for (j=0;j<n;j++)
  {cout<<"a["<<i+1<<"]["<<j+1<<"]="; cin>>*(*(a+i)+j); }
q=new int[n*l];
if(q==NULL)
{cout<<"nu exista suficienta memorie"<<endl;}
cout<<endl<<" A doua matrice:"<<endl;
for (i=0;i<n;i++)   for (j=0;j<l;j++)
  {cout<<"b["<<i+1<<"]["<<j+1<<"]="; cin>>*(*(b+i)+j); }
cout<<endl<<"Matricea produs este";
for (i=0;i<m;i++)  {  cout<<endl;
     for (j=0;j<l;j++)   { s=0;
       for (k=0;k<n;k++)
       s+=*(*(a+i)+k)* (*(*(b+k)+j));  cout<<s<<"\t";}}
cout<<endl;
delete p; delete q;return 0;}
```

***Rezultate:***

```
nr. de linii prima matrice,m=2

nr. de coloane prima matrice=
nr. de linii a doua matrice,n=3
nr de coloane la a doua matrice l=2

 Prima matrice:
a[1][1]=1
a[1][2]=2
a[1][3]=3
a[2][1]=1
a[2][2]=2
a[2][3]=3

 A doua matrice:
b[1][1]=2
b[1][2]=2
b[2][1]=2
b[2][2]=10
b[3][1]=10
b[3][2]=10

Matricea produs este
  36    52
  36    52
```

***Aplicație:***

Să se modifice programul astfel încât să se calculeze și afișeze matricile la pătrat

***Ex.5. Programul (în C++) alocă memorie dinamică pentru trei șiruri de câte 256 de caractere și eliberează memoria alocată dinamic după inițializarea celor trei șiruri cu valori constante.***

**Varianta 1 in C++**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{char *sir1=new char[10];
char *sir2,*sir3;
int i;
sir2=new char[10];
for (i=0;i<10;i++)
    {sir1[i]='A';       printf("%c%d ",sir1[i], i);
    sir2[i]='Z';        printf("%c%d ",sir2[i],i);}
delete sir1;
printf("\n");
sir3=new char[10];
for (i=0;i<10;i++)
   {sir3[i]=sir2[i]; printf("%c ",sir3[i]);}
printf("\n");
delete sir2;delete sir3;
return 0;}
```

**Varianta 2 in C++**

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
{char *sir1=new char[10];
char *sir2,*sir3;
int i;
sir2=new char[10];
for (i=0;i<10;i++)
   {sir1[i]='A';       cout<<sir1[i]<<i<<"\t";
    sir2[i]='Z';       cout<<sir2[i]<<i<<"\t";}
delete sir1;
cout<<endl;
sir3=new char[10];
for (i=0;i<10;i++)
   {sir3[i]=sir2[i]; cout<<sir3[i]<<"  ";}
cout<<endl;
delete sir2;delete sir3;
return 0;}
```

***Rezultate:***

```
A0 Z0 A1 Z1 A2 Z2 A3 Z3 A4 Z4 A5 Z5 A6 Z6 A7 Z7 A8 Z8 A9 Z9
Z Z Z Z Z Z Z Z Z Z
```

***Aplicație:***

Să se modifice programul astfel încât să se calculeze și afișeze elementele șirurilor din 2 în 2.

***Ex.6. Programul (în C++) alocă memorie dinamică pentru o matrice de nxm elemente intregi citite de la tastatura utilizand new si afiseaza matricea si matricea inmultita cu o constanta intreaga.***

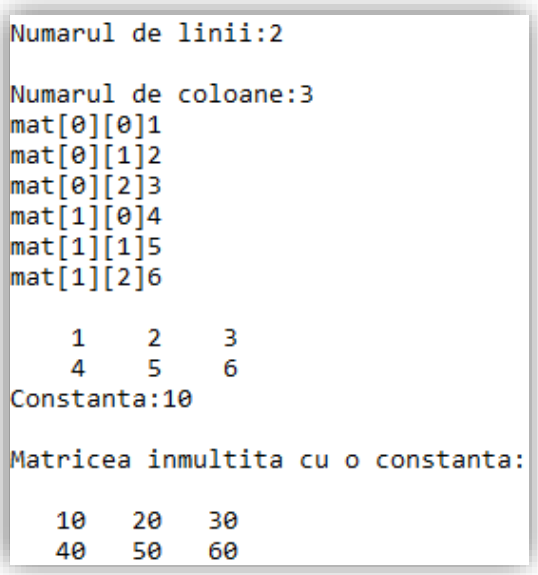
**Varianta 1 in C++**

```
#include <stdio.h>
int main()
{int n, m, ** mat, i, j, c;
printf("\nNumarul de linii:"); scanf("%d", &n);
printf("\nNumarul de coloane:");
scanf("%d", &m);
mat=new int*[n];
for(i=0 ; i<n ; i++) mat[i]=new int[m];
//citirea elementelor matricei
for(i=0 ; i<n ; i++)  for(j=0 ; j<m ; j++)
{ printf("mat[%d][%d] = ", i, j);
  scanf("%d", &mat[i][j]); }
//afişarea elementelor matricei
for(i=0 ; i<n ; i++)
 { printf("\n"); for(j=0 ; j<m ; j++)
   printf("%5d", mat[i][j]); }
printf("\nConstanta:"); scanf("%d", &c);
printf("\nMatricea inmultita cu o constanta:\n");
for(i=0 ; i<n ; i++)
 { printf("\n"); for(j=0 ; j<m ; j++)
   printf("%5d", c*mat[i][j]); }
for(i=0; i<n ; i++) // dealocarea memoriei
 delete [ ] mat[i];
 delete [ ] mat;
return 0;}
```

**Varianta 2 in C++**

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{int n, m, ** mat, i, j, c;
 cout <<"\nNumarul de linii:"; cin>>n;
 cout <<"\nNumarul de coloane:"; cin >>m;
 mat=new int*[n];
 for(i=0 ; i<n ; i++) mat[i]=new int[m];
//citirea elementelor matricei
 for(i=0 ; i<n ; i++)  for(j=0 ; j<m ; j++)
 { cout <<"mat"<<"["<<i<<"]["<<j<<"]"; cin>>mat[i][j];}
//afişarea elementelor matricei
 for(i=0 ; i<n ; i++)
 { cout<<"\n"; for(j=0 ; j<m ; j++)
cout<<setw(5)<<mat[i][j]; }
 cout<<"\nConstanta:"; cin>>c;
 cout<<"\nMatricea inmultita cu o constanta:\n";
 for(i=0 ; i<n ; i++)
  { cout<<"\n"; for(j=0 ; j<m ; j++)
    cout<<setw(5)<<c*mat[i][j]; }
for(i=0; i<n ; i++) // dealocarea memoriei
delete [ ] mat[i];
delete [ ] mat;
return 0;}
```

***Rezultate:***



```
Numarul de linii:2

Numarul de coloane:3
mat[0][0]1
mat[0][1]2
mat[0][2]3
mat[1][0]4
mat[1][1]5
mat[1][2]6

    1    2    3
    4    5    6
Constanta:10

Matricea inmultita cu o constanta:

   10   20   30
   40   50   60
```

***Aplicație:***

Să se modifice programul astfel încât să se calculeze și afișeze diagonala principala a matricii.

***Ex.7. Programul realizeaza căutarea unei litere într-un bloc de memorie în care s-au stocat valorile unui şir de caractere,***

**Varianta in C**

```
#include <stdio.h>
#include <string.h>
int main ()
{char *pch;  char str[] = "Exemplu sir pointeri";
pch = (char*) memchr (str, 'p', strlen(str));
printf("sirul este: %s\n", str);
if (pch!=NULL)
  printf ("'p' apare prima data pe pozitia %d din sir.\n", pch-
str+1);
   else printf (" 'p' nu a fost gasit in sir.\n");
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{char *pch;  char str[] = "Exemplu sir pointeri";
pch = (char*) memchr (str, 'p', strlen(str));
cout <<"sirul este:"<<str;
if (pch!=NULL)
   cout<<"\n'p' apare prima data pe pozitia "<< pch-
str+1<<" din sir.\n";
   else cout <<" 'p' nu a fost gasit in sir.\n";
   return 0;}
```

***Rezultate:***

```
sirul este:Exemplu sir pointeri
'p' apare prima data pe pozitia 5 din sir.
```

***Aplicație:***

Să se modifice programul astfel încât să se citeasca sirul de la tastatura si sa se afișeze pe ce pozitie apare litera i prima data in sir.

***Ex.8. Programul compara doua blocuri de memorie care contin fiecare cate un sir de caractere si afieaza un mesaj daca sunt identice.***

**Varianta in C**

```
#include <stdio.h>
#include <string.h>
int main ()
{char buffer1[] = "Sir text caractere";
 char buffer2[] = "Sir text caractere";
 int n;  n=memcmp( buffer1, buffer2, sizeof(buffer1) );
 if (n>0)
 printf ("'%s' este mai mare decat '%s'.\n",buffer1,buffer2);
 else if (n<0)
    printf ("'%s' este mai mic decat'%s'.\n",buffer1,buffer2);
    else printf ("'%s' este identic cu '%s'.\n",buffer1,buffer2);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h>
using namespace std;
int main ()
{char buffer1[] = "Sir text caractere";
 char buffer2[] = "Sir text caractere";
 int n;  n=memcmp( buffer1, buffer2, sizeof(buffer1) );
 if (n>0)
 cout<<buffer1<<" este mai mare decat "<<buffer2;
 else if (n<0)
   cout <<buffer1 <<" este mai mic decat"<<
   buffer2<<endl;
 else cout<<buffer1<< " este identic cu
"<<buffer2<<endl;
return 0;}
```

***Rezultate:***

```
'Sir text caractere' este identic cu 'Sir text caractere'.
```

***Aplicație:***

Să se modifice programul astfel încât să se caute in primul sir litera p si litera i si daca se gasesc sa se afiseze pozitia respectiva in sir, utilizand memchr()

***Ex.9. Programul realizează copierea unui bloc de memorie peste alt bloc (copiere şir peste alt şir)***

**Varianta in C**

```
#include <stdio.h>
#include <string.h>
int main ()
{char str1[] = "Test"; char str2[] = "PCLP2";
puts("str1 inainte de copiere cu memcpy ");
puts(str1);/* Copies contents of str2 to str1 */
memcpy (str1, str2, sizeof(str2));
puts("\nstr1 dupa copiere cu memcpy ");
puts(str1);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h>
using namespace std;
int main ()
{char str1[] = "Test"; char str2[] = "PCLP2";
cout<<"str1 inainte de copiere cu memcpy "<<endl;
cout<<str1;/* Copies contents of str2 to str1 */
memcpy (str1, str2, sizeof(str2));
cout<<"\nstr1 dupa copiere cu memcpy "<<endl;
cout<<str1;
return 0;}
```

***Rezultate:***

```
str1 inainte de copiere cu memcpy
Test

str1 dupa copiere cu memcpy
PCLP2
```

***Aplicație:***

Să se modifice programul astfel încât să se afiseze si str2 dupa copiere.

***Ex.10. Programul realizează mutarea unui bloc de memorie la adresa altui bloc (copiere şir peste alt şir)***

**Varianta in C**

```
#include <stdio.h>
#include <string.h>
int main ()
{char str1[] = "Test";
 char str2[] = "Programare PCLP2";
 puts("str1 inainte de memmove ");
 puts(str1);
 memmove(str1, str2, sizeof(str2));
 puts("\nstr1 dupa memmove ");
 puts(str1);
 return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h>
using namespace std;
int main ()
{char str1[] = "Test";
 char str2[] = "Programare PCLP2";
 cout<<"str1 inainte de memmove "<<endl;
 cout<<str1;
 memmove(str1, str2, sizeof(str2));
 cout<<"\nstr1 dupa memmove "<<endl;
 cout<<str1;
 return 0;}
```

***Rezultate:***

```
str1 inainte de memmove
Test

str1 dupa memmove
Programare PCLP2
```

***Aplicație:***

Să se modifice programul astfel încât să se afiseze str2 dupa copiere.

***Ex.11. Programul realizează iniţializarea unui bloc de memorie care conţine un şir de caractere, prin înlocuirea a 10 caractere cu '.' începând cu poziţia a 10 –a din şir***

**Varianta in C++**

```
#include <stdio.h>
#include <string.h>

int main()
{char str[50] = "Exemplu de utilizare memset";
printf("Inainte de memset(): %s\n", str);
// initializeaza 10 octeti cu '.' incepand cu str[10]
memset(str + 10, '.', 10*sizeof(char));
printf("Dupa memset(): %s", str);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{char str[50] = "Exemplu de utilizare memset";
cout<<"Inainte de memset(): "<< str<<endl;
// initializeaza 10 octeti cu '.' incepand cu str[10]
memset(str + 10, '.', 10*sizeof(char));
cout<<"Dupa memset(): "<< str;
return 0;}
```

***Rezultate:***

```
Inainte de memset(): Exemplu de utilizare memset
Dupa memset(): Exemplu de.......... memset
```

***Aplicaţie:***

Să se modifice programul astfel încât să inlocuiasca caracterul . cu *

## PROBLEME PROPUSE

1. ***Să se scrie programul care realizează alocarea dinamică a memoriei pentru un şir de n (citit de la tastatura) numere reale şi calculează şi afişează termenii şirului şi transpusa unei matrici utilizând alocarea dinamică a memoriei.***

2. ***Să se scrie programul care realizează inversa şi transpusa unei matrici utilizând alocarea dinamică a memoriei.***

3. ***Se consideră o matrice A cu m linii şi n coloane, cu elemente reale. Să se scrie programul care calculează şi afişează expresia de mai jos utilizând alocarea dinamică a memoriei:***

$$E = \sqrt{\prod_{i=1}^{m}\left(\sum_{j=1}^{n} a_{i,j}^2\right)}$$ ***unde*** $a_{i,j}$ ***sunt elementele matricii.***

4. ***Se consideră matricile pătratice A şi B cu n linii şi coloane. Se cere să se scrie un program pentru determinarea matricii*** $C=A^{-1}-B^{T}$ ***utilizând alocarea dinamică a memoriei.***

# Capitolul 6
# Linia de comandă.Argumentele funcției main()

În acest capitol sunt prezentate consideraţii teoretice privind utilizarea liniei de comandă şi argumentele funcţiei main() .

## CONSIDERAŢII TEORETICE

Programele executabile rezultate în urma compilării şi link-editării fişierelor text într-un mediu de programare sunt fişiere care se pot executa direct din linia de comandă într-o fereastră de tip Command Prompt.

**Linia de comandă** este compusă din numele programului ce urmează a fi executat şi argumentele liniei de comandă.

**Argumentele liniei de comandă** sunt informaţiile ce urmează numelui de program care va fi executat. Argumentele sunt despărţite prin câte un singur spaţiu şi reprezintă **argumentele funcţiei main** din programul executabil specificat în linia de comandă.

Headerul funcţiei main() este :

**int main(int argc, char *argv[]) { … }**

unde:

**argc** = este de tip întreg şi memorează numărul de argumente din linia de comandă. Are valoarea 1 implicit (cel puţin numele programului este dat în linia de comandă).

**argv** = este un tablou (şir) de pointeri către şiruri de caractere. Fiecare element al tabloului indică spre un argument din linia de comandă. Toate argumentele liniei de comandă sunt şiruri de caractere.

Astfel:

- **argv[0]** indică spre numele programului,
- **argv[1]** indică spre primul argument din linia de comandă

  după numele programului, etc…
- **argv[ ]** parantezele drepte fără conţinut indică faptul că şirul are lungimea nedeterminată.

Argumentele funcţiei main() , argc și argv[] sunt folosite pentru a prelua argumentele liniei de comandă.

## PROBLEME REZOLVATE

***Ex.1 Programul preia din linia de comandă un nume specificat de utilizator și afișează mesajul "Hello <<nume>> ce mai faci? " Proiectul se va numi nume.c și va deveni în urma compilării și link-editării "nume.exe". Acest program executabil se va lansa de la prompterul sistem ca o linie de comandă, astfel :***

**>nume Mihai**

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{if (argc!=2)
    {printf("Ati uitat sa va scrieti numele: \n");
    exit(1);}
 printf("Hello %s ce mai faci?", argv[1]);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include<stdlib.h>
using namespace std;
int main (int argc, char *argv[])
{if (argc!=2)
    {cout<<"Ati uitat sa va scrieti numele: "<<endl;
    exit(1);}
 cout<<"Hello "<<argv[1]<<" ce mai faci?";
return 0;}
```

***Rezultate:***

```
D:\laura\codeblocks\nume\bin\Debug>nume Laura
Hello Laura ce mai faci?
```

```
D:\laura\codeblocks\nume\bin\Debug>nume
Ati uitat sa va scrieti numele:
```

***Aplicație:***

*Să se modifice programul astfel încât să se introducă din linia de comandă: numele, prenumele și vârsta.*

***Ex.2. Programul citește un număr din linia de comandă și realizează numărătoarea inversă pornind de la numărul specificat până la 0 (similar cu cronometru). La sfârșit se afișează textul "gata". Exemplu de linie de comandă:***
***>afiseaza 5 invers***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

int main (int argc, char *argv[])
{int afis, contor;
if (argc<2)
{printf("introduceti un numar intreg, mai incercati o data\n");
exit(1);}
if (argc==3 && !strcmp(argv[2], "invers")) afis=1;
else afis = 0;
for (contor=atoi(argv[1]);contor;--contor)
if(afis) printf("%d\n", contor);
printf("gata\n");
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
using namespace std;
int main (int argc, char *argv[])
{int afis, contor;
if (argc<2)
{cout<<"introduceti un numar intreg, mai incercati o
data"<<endl; exit(1);}
if (argc==3 && !strcmp(argv[2], "invers")) afis=1;
else afis = 0;
for (contor=atoi(argv[1]);contor;--contor)
if(afis) cout<<contor<<endl;
cout<<"gata"<<endl;
return 0;}
```

***Rezultate:***

```
D:\laura\codeblocks\afiseaza\bin\Debug>afiseaza
introduceti un numar intreg, mai incercati o data
```

```
D:\laura\codeblocks\afiseaza\bin\Debug>afiseaza 5 inv
5
4
3
2
1
gata
```

***Aplicație:***

*Să se modifice programul astfel încât să se intoducă din linia de comandă: un număr întreg și să se afișeze dacă numărul este în intervalul [5,100] sau nu.*

***Ex.3. Programul suma.c calculează și afișează suma a 2 numere preluate din linia de comandă. Exemplu de linie de comandă: >suma 200 350***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{int a,b;
if (argc!=3)
 {  printf("Trebuie sa introduceti 2 nr.intregi \n");
exit(1); }
//convertirea sirului de caractere la nr. intreg
a=atoi(argv[1]);
printf("a=%d ",a);
b=atoi(argv[2]);
printf("b=%d ",b); printf("s= %d ",a+b);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main (int argc, char *argv[])
{int a,b;
if (argc!=3)
 {  cout<<"Trebuie sa introduceti 2 nr.intregi
"<<endl;    exit(1); }
//convertirea sirului de caractere la nr. intreg
a=atoi(argv[1]); cout<<"a="<<a;
b=atoi(argv[2]);
cout<<" b="<<b; cout<<" s="<<a+b;
return 0;}
```

***Rezultate:***

```
D:\laura\codeblocks\suma\bin\Debug>suma
Trebuie sa introduceti 2 nr.intregi
```

```
D:\laura\codeblocks\suma\bin\Debug>suma 100 20
a=100 b=20 s= 120
```

***Aplicație:***

*Să se modifice programul astfel încât să se efectueze și alte operații aritmetice (scăderea, înmulțirea, împărțirea, etc) introduse în linia de comandă.*

***Ex.4. Programul calculează și afișează suma unui șir de numere preluate din linia de comandă. Exemplu de linie de comandă: >sum 1 2 4 6 8***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{int i,s=0, n;
if (argc==1)
{printf("Nu ati introdus nici un nr!\n");  exit(1); }
else
    {for (i=1;i<argc;i++)   {n=atoi(argv[i]);s+=n;  }
printf("suma argumentelor din linia de comanda
este=%d",s);}
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main (int argc, char *argv[])
{int i,s=0, n;
if (argc==1)
{cout<<"Nu ati introdus nici un nr!"<<endl;  exit(1);}
else
    {for (i=1;i<argc;i++)   {n=atoi(argv[i]);s+=n;  }
cout<<"suma argumentelor din linia de comanda este="<<s;}
return 0;}
```

***Rezultate:***

```
D:\laura\codeblocks\sum\bin\Debug>sum
Nu ati introdus nici un nr!
```

```
suma argumentelor din linia de comanda este=150
D:\laura\codeblocks\sum\bin\Debug>
```

***Aplicație:***

*Să se modifice programul astfel încât să se afișeze produsul numerelor și suma pătratelor numerelor introduse în linia de comandă.*

## PROBLEME PROPUSE

1. ***Să se scrie un program care să ordoneze un şir de nr. întregi introdus în linia de comandă.***

2. ***Să se scrie un program care inversează şirul de caractere introdus in linia de comandă.***

3. ***Să se scrie un program care concatenează două şiruri de caractere introduse în linia de comandă***

4. ***Să se scrie un program care determină minimul şi maximul precum şi elementele impare dintre elementele unui şir de numere întregi specificate în linia de comandă.***

5. ***Să se scrie un program care determină c.m.m.m.c a două numere specificate în linia de comandă***

6. ***Să se scrie programul care determină şi afişează valoarea funcţiei ln(1+x) într-un punct dat $x \in (-1,1]$ introdus in linia de comandă, utilizând dezvoltarea în serie :***

$$ln(1+x) = \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$$

***şi să se compare rezultatele obţinute astfel cu rezultatul obţinut prin apelarea funcţiei log() din biblioteca <math.h> sau <cmath>.***

$$ln(2) = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \ldots$$

***Caz particular:***

7. ***Să se scrie programul care determină şi afişează valoarea funcţiei sinus hiperbolic într-un punct dat x introdus în linia de comandă, utilizând dezvoltarea în serie şi realizând comparaţia între rezultatele obţinute astfel cu cele obţinute prin apelarea funcţiei sinh() din biblioteca <math.h>:***

$$sh(x) = \frac{x}{1!} + \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots$$

# Capitolul 7
# Definirea structurilor.Structuri și tablouri. Structuri și funcții.Pointeri la structuri

În acest capitol sunt prezentate consideraţii teoretice şi probleme rezolvate privind definirea şi utilizarea datelor de tip structură, legătura dintre tablouri şi structuri, definirea şi utilizarea tablourilor de structuri, legătura dintre structuri şi funcţii precum şi pointeri la structuri.

## CONSIDERAŢII TEORETICE

Tipurile de date definite de utilizator se clasifică în limbajul C/C++ în următoarele categorii principale:

- structuri
- uniuni
- enumerari
- typedef
- câmpuri de biți

**Definirea structurilor**

**Structura**: este o colecţie de variabile ce pot fi de tipuri diferite (spre deosebire de tablouri) reunite sub acelaşi nume. La declararea unei structuri se aloca automat memorie pentru fiecare membru al structurii.

**Elementele (membri, câmpurile):** sunt variabilele din componenta unei structuri care pot fi de orice tip (standard, pointeri, șiruri, alte structuri, etc) iar numele lor pot fi aceleași cu elementele altei structuri din acelaşi program.

**Declararea tipului structura** se realizează după formatul**:**

```
struct nume_tip_struct
        {       tip var1;
                tip var2;
        …
        } ;
```

Declararea variabilei de tip structura se realizează cu instructiunea:

```
struct nume_tip_struct   lista_variabile_struct;
```

Declararea structurii printr-o singură instrucțiune:

```
struct nume_tip_struct
        { tip var1;
          tip var2;
          …
         }  lista_variabile_struct;
```

unde nume_tip_struct și lista_variabile_struct sunt opționale, dar nu pot lipsi ambele simultan.

**Ex.1**: Declararea unei structuri de tip student, utilizând declaraţia de tip separat faţă de declaraţia de variabilă .

```
struct  student {         // declararea tipului structurii
            int  nume[20];
            int  age;
            int  cod_curs;
            int an_studiu;
            int id_stud; } ;
struct student stud1,stud2,stud3; //declararea variabilelor
```

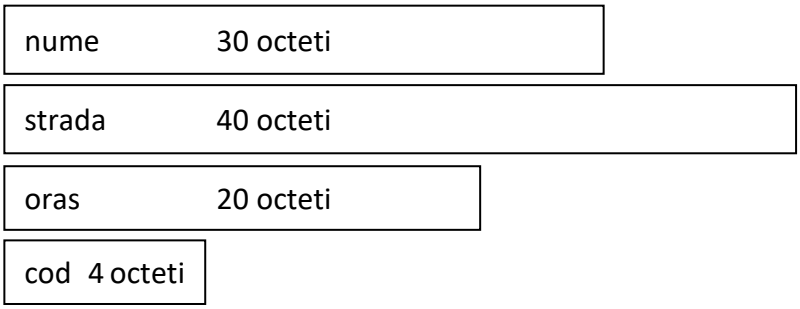
**Ex.2**: Declararea unei structuri de tip student, într-o singură instrucțiune.

```
struct  student {          // declararea tipului structurii
            int  nume[20];
            int  age;
            int  cod_curs;
            int an_studiu;
            int id_stud;
          } stud1,stud2,stud3;  //declararea variabilelor
```

**Ex.3**: Declararea unei structuri de tip adresă.

```
struct adrese          // adrese =numele tipului structurii
{   char nume[30];  //elementele sunt declarate intre acolade
              char strada[40];
              char oras[20];
              int cod;
};              // aici se termina declararea tipului structurii
struct adrese adr;    //declararea variabilei adr de tipul adrese
```

Pentru această variabilă adr (din Ex.3), de tip structură se vor rezerva 30+40+20+4=94 octeți de memorie, considerând că tipul char se reprezintă pe câte un octet iar tipul int pe 4 octeți.




**Accesul la elementele unei structuri** se realizează fie prin **operatorul punct (.), astfel**:

**nume_struct.nume_membru**

fie prin intermediul pointerilor la structuri ce vor fi definiți în cele ce urmează.

**Inițializarea elementelor unei structuri** se realizează similar cu inițializarea șirurilor, utilizând declararea de tip structură, constantele de inițializare fiind separate prin virgulă.

Informația conținută într-o structură poate fi atribuită altei structuri de acelaşi tip printr-o singură instrucțiune de atribuire. Astfel, dacă se declară două variabile de acelaşi tip structură, a și b , instrucțiunea de atribuire este b=a; și prin aceasta se înțelege că fiecare din câmpurile structurii a este atribuit câmpului corespunzător din structura b.

**Pointeri și structuri**

Formatul de **declarare a unui pointer la o structură:**

**struct tip_struct *adr_pointer**

unde ***adr_pointer** va fi un pointer către variabila de tip structură de tipul **tip_struct**

**Operatorul "săgeată" ->** permite accesul la elementele structurii folosind un pointer la această structură.

**Ex.** : Accesarea elementelor structurii folosind pointeri:

```
struct tip
        {  int i
           float bilant;
            } persoana;
struct tip *p;    //declara un pointer la structura
p=&persoana;  //preia adresa structurii persoana în pointerul p
p->bilant=0;    //permite accesul la membrul bilant din structura persoana
p->i++;         //incrementeaza membrul i al structurii
```

**Pointeri şi tablouri.**

a) **Tablourile ca elemente ale unei structuri**

Daca se declară mai multe variabile de tipul structură prezentat mai jos, și un pointer *p la una din aceste variabile, structi:

```
struct nume_tip_struct
        { tip var1;
          tip tab[max];          //tablou unidimensional
          tip mat[max][max];  //matrice
          …
         } struct1,struct2,…,structi,…, *p;
p=&structi
```

atunci accesul la tabloul tab[] se realizează în două moduri:

```
structi.tab[i]=expresie sau constantă;
p->tab[i]=expresie sau constantă; //cu pointeri
```

iar accesul la tabloul mat[][] se face astfel:

```
structi.mat[i][j]=expresie sau constantă
p->mat[i][j]=expresie sau constantă //cu pointeri
```

b) **Tablourile de structuri** se studiaza în Lab.7.

**Structuri imbricate**

Formatul de declarare a structurilor imbricate este :

```
struct tip_str1 {
                 tip var1;
                 tip var2;
                 …
                };
struct tip_str2 {
                 tip var1;
                 tip var2;
                 struct tip_str1 structa;
                  …
                 }struct1,struct2, …structi,…;
```

Accesul la elementele variabilei **structi** se realizează prin intermediul punctului, în mod analog cu adresarea structurii simple, astfel: **structi.structa.var1;**

**Ex.:** structuri imbricate (structura adr de tip adrese este imbricată în structura pers de tip angaj)

```
struct adrese {   char nume[30];
                  char strada[40];
                  char oras[20];
                  int cod; };
struct angaj {   struct adrese adr
                 float salar;} pers;
//pentru a atribui valoarea 3400 elementului cod din structura adr:
pers.adr.cod=3400;
```

**Structuri şi funcţii**

Transmiterea elementelor structurii către funcţii :

- transmitere element cu element către funcţii
- transmiterea structurilor întregi către funcţii

**Ex.**: transmiterea element cu element către funcţii :

```
struct tip
  {   char x;
      int y;
      float z;
      char s[10];
  } mihai;
…
func(mihai.x);  //preia valoarea de tip caracter din x
func1(mihai.y);          //preia valoarea intreaga din y
func2(mihai.z);          //preia valoarea de tip float din z
func3(mihai.s);          //preia adresa sirului s
func4(mihai.s[2]); //preia valoarea de tip caracter din s[2]
```

## PROBLEME REZOLVATE

***Ex.1.: Programul declară o variabilă de tip structură cu format de adresă. Se vor citi de la tastatură câmpurile nume, strada, ap, local și se atribuie prin program valori aleatoare câmpurilor nr și cod. Se afișează toate câmpurile acestei structuri.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>

int main()
{ struct adrese{
char nume[30];
        char strada[40];
        int nr;
        int ap;
        char local[20];
        int cod; } adr;
printf("Numele si prenumele: "); gets(adr.nume);
printf("Strada:"); gets(adr.strada); adr.nr=26;
printf("apartamentul:"); scanf("%d", &adr.ap);
printf("Localitate:"); scanf("%s", &adr.local); adr.cod=400118;
printf("\nAti introdus urmatoarele date:\n");
printf("%s\nAdresa:%s,%d,ap.%d\n",adr.nume,adr.strada,adr.
  nr,adr.ap);
printf("%s\nCod:%d\n", adr.local,adr.cod);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
{ struct adrese{
char nume[30];
        char strada[40];
        int nr;
        int ap;
        char local[20];
        int cod; } adr;
cout<<"Numele si prenumele: ";
cin.get(adr.nume,30); cin.ignore(); cout<<"Strada:";
cin.get(adr.strada,40); cin.ignore(); adr.nr=26;
cout<<"apartamentul:"; cin>>adr.ap;
cout<<"Localitate:"; cin>>adr.local; adr.cod=400118;
cout<<"\nAti introdus urmatoarele date:"<<endl;
cout<<adr.nume<<"\nAdresa:"<<adr.nume<<","
  <<adr.strada<<","<<adr.nr<<",ap."<<adr.ap<<endl;
cout<<adr.local<<endl<<"Cod:"<<adr.cod<<endl;
return 0;}
```

***Rezultate:***

```
Numele si prenumele: Popa Alessia
Strada:Baritiu
apartamentul:24
Localitate:Cluj-Napoca

Ati introdus urmatoarele date:
Popa Alessia
Adresa:Baritiu,26,ap.24
Cluj-Napoca
Cod:400118
```

***Aplicație:***

Să se modifice programul astfel încât să se declare 2 variabile de tip adrese: home_adr, office_adr care se inițializează cu date diferite și apoi se afișează.

***Ex.2.: Programul este un exemplu de atribuire a unei structuri altei structuri. Se definesc 2 variabile x și y de tip structură cu 2 câmpuri a și b. Se inițializează variabila x cu valori constante și se atribuie structurii y întreaga structură x, afișându-se câmpurile structurii y.***

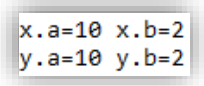
**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>

int main()
{struct { int a;  int b;
} x,y;    x.a=10; x.b=2;
y=x;   //atribuie intreaga structura (element cu element) x lui y
printf("x.a=%d x.b=%d\n", x.a, x.b);
printf("y.a=%d y.b=%d\n", y.a, y.b);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
{struct { int a;  int b;
} x,y;    x.a=10; x.b=2;
y=x; //atribuie intreaga structura (element cu
   element) x lui y
cout<<"x.a="<<x.a<<" x.b="<<x.b<<endl;
cout<<"y.a="<<y.a<<" y.b="<<y.b<<endl;
return 0;}
```

***Rezultate:***




***Aplicație:***
Să se modifice programul astfel încât să se afișeze x.a+y.a și y.a+y.b.

***Ex.3. Programul ilustrează operațiile de bază (adunare, scădere, înmulțire, împărțire) specifice numerelor complexe, și afișează modulul și argumentele acestora utilizând o variabilă de tip structură cu două câmpuri: partea reală și partea imaginară. Operațiile sunt descrise mai jos.***

- ***Adunarea/scăderea:*** $x+iy=(x_1+iy_1)\pm(x_2+iy_2)$
- ***Produsul:*** $x+iy=(x_1+iy_1)(x_2+iy_2)=(x_1x_2-y_1y_2)+i(x_1y_2+x_2y_1)$
- ***Câtul:*** $\boldsymbol{x+iy=\frac{x_1+iy_1}{x_2+iy_2}=\frac{x_1x_2+y_1y_2}{{x_2}^2+{y_2}^2}+i\frac{x_2y_1-x_1y_2}{{x_2}^2+{y_2}^2}}, {x_2}^2+{y_2}^2>0$
- ***Valoarea absolută:*** $\boldsymbol{|x+iy|=\sqrt{x^2+y^2}}$.
- ***Argumentul:***

$$arg(\boldsymbol{x+iy})=\begin{cases}\boldsymbol{arctg(\frac{y}{x}), arctg(\frac{y}{x})>=0}\\ \boldsymbol{arctg(\frac{y}{x})+2\pi, arctg(\frac{y}{x})<0}\end{cases}$$

**Varianta in C**

```
#include <stdio.h>
#include<conio.h>
#include <math.h>
#define PI 3.1415926
int  main()
{struct { double re; double im;} z,y;
//citirea nr.complexe
printf("Introduceti z=a+bj\n"); printf("a= "); scanf("%lf",&z.re); printf("b= "); scanf("%lf",&z.im);
//afisarea primului nr. complex
printf("z=%2.2lf+%2.2lfj\n", z.re, z.im);
//citirea celui de-al 2-lea nr.complex
printf("Introduceti y=c+dj\n");
printf("c= "); scanf("%lf",&y.re); printf("d= ");  scanf("%lf",&y.im);
//afisarea celui de-al 2-lea nr.complex
printf("y=%2.2lf+%2.2lfj\n", y.re, y.im);
//adunarea nr. complexe
printf("adunarea nr. complexe:\n");
printf("(%2.2lf+%2.2lfj)+(%2.2lf+%2.2lfj)=%2.2f+%2.2fj\n",z.re,z.im,y.re, y.im,z.re+y.re,z.im+y.im );
//inmultirea a nr. complexe
printf("inmultirea nr. complexe:\n");
printf("%2.2lf+%2.2lfj)*(%2.2lf+%2.2lfj)=%2.2f+%2.2fj\n",z.re,z.im, y.re,y.im,z.re*y.re-z.im*y.im, z.re*y.im+z.im*y.re);
//impartirea nr. complexe
printf("impartirea nr. complexe:\n");
if((z.im*z.im+y.im*y.im)>0)
{printf("%2.2lf+%2.2lfj)/(%2.2f+%2.2fj)= \n",z.re,z.im, y.re,y.im);
printf("=%2.2lf+%2.2lfj\n",(z.re*z.im+y.re*y.im)/(z.im*z.im+y.im*y.im),(z.im* y.re-z.re*y.im) /(z.im*z.im+y.im*y.im));} else
{printf("numarator =0\n") ;}
//modulul nr. complex
printf("modulul lui %2.2lf + %2.2lfj = %2.2f\n", z.re, z.im, sqrt(z.re*z.re +z.im*z.im));
printf("modulul lui %2.2lf + %2.2lfj = %2.2f\n", y.re, y.im, sqrt(y.re*y.re +y.im*y.im));
//argumentul nr.complex
if (atan2(z.im,z.re)>=0.) printf("arg(%2.2lf+%2.2lf j)=%2.2lf\n", z.re,z.im, atan2(z.im,z.re));
else  printf("arg(%2.2lf+%2.2lfj)=%2.2lf\n", z.re,z.im, atan2(z.im,z.re)+2*PI);
if (atan2(y.im,y.re)>=0.) printf("arg(%2.2lf+%2.2lf j)=%2.2lf\n", y.re,y.im, atan2(y.im,y.re));
else  printf("arg(%2.2lf+%2.2lfj)=%2.2lf\n", y.re,y.im, atan2(y.im,y.re) +2*PI);
 return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include<conio.h>
#include <math.h>
#define PI 3.1415926
using namespace std;
int  main()
{struct { double re;  double im;} z,y;
//citirea nr.complexe
cout<<"Introduceti z=a+bj"<<endl;
cout<<"a= "; cin>>z.re; cout<<"b= "; cin>>z.im;
//afisarea primului nr. complex
cout<<"z="<<z.re<<"+"<<z.im<<"j"<<endl;
//citirea celui de-al 2-lea nr.complex
cout<<"Introduceti y=c+dj"<<endl;
cout<<"c= "; cin>>y.re; cout<<"d= ";  cin>>y.im;
//afisarea celui de-al 2-lea nr.complex
cout<<"y="<<y.re<<"+"<<y.im<<"j"<<endl;
//adunarea nr. complexe
cout<<"adunarea nr. complexe:"<<endl;
cout<<"("<<z.re<<"+"<<z.im<<"j)+("<<y.re<<"+"<<y.im<<"j)="<<z.re+y.re<<"+"<<z.im+y.im<<"j"<<endl;
//inmultirea a nr. complexe
cout<<"inmultirea nr. complexe:"<<endl;
cout<<"("<<z.re<<"+"<<z.im<<"j)*("<<y.re<<"+"<<y.im<<"j)="<<z.re*y.re-
z.im*y.im<<"+"<<z.re*y.im+z.im*y.re<<"j"<<endl;  //impartirea nr. complexe
cout<<"impartirea nr. complexe:"<<endl;
if((z.im*z.im+y.im*y.im)>0)
{cout<<"("<<z.re<<"+"<<z.im<<"j)/("<<y.re<<"+"<<y.im<<"j)=";
cout<<(z.re*z.im+y.re*y.im)/(z.im*z.im+y.im*y.im)<<"+"<<(z.im* y.re-z.re*y.im) /(z.im*z.im+y.im*y.im)<<"j"<<endl;} else
{cout<<"numarator =0"<<endl ;}
//modulul nr. complex
cout<<"modulul lui "<<z.re<<"+"<< z.im<<"j = "<< sqrt(z.re*z.re +z.im*z.im)<<endl;
cout<<"modulul lui "<<y.re<<"+"<< y.im<<"j = "<< sqrt(y.re*y.re +y.im*y.im)<<endl;
//argumentul nr.complex
if (atan2(z.im,z.re)>=0.) cout<<"arg("<<z.re<<"+"<<z.im<<"j)="<<atan2(z.im,z.re)<<endl;
else  cout<<"arg("<<z.re<<"+"<<z.im<<"j)="<< atan2(z.im,z.re)+2*PI<<endl;
if (atan2(y.im,y.re)>=0.) cout<<"arg("<<y.re<<"+"<<y.im<<"j)="<<atan2(y.im,y.re)<<endl;
else  cout<<"arg("<<y.re<<"+"<<y.im<<"j)="<< atan2(y.im,y.re) +2*PI<<endl;
return 0;}
```

***Rezultate:***

```
Introduceti z=a+bj
a= 5.5
b= 4.8
z=5.50+4.80j
Introduceti y=c+dj
c= 1.2
d= 1.1
y=1.20+1.10j
adunarea nr. complexe:
(5.50+4.80j)+(1.20+1.10j)=6.70+5.90j
inmultirea nr. complexe:
5.50+4.80j)*(1.20+1.10j)=1.32+11.81j
impartirea nr. complexe:
5.50+4.80j)/(1.20+1.10j)=1.14+-0.01j
modulul lui 5.50 + 4.80j = 7.30
modulul lui 1.20 + 1.10j = 1.63
arg(5.50+4.80 j)=0.72
arg(1.20+1.10 j)=0.74
```

***Aplicație:***

*Să se rezolve aceeași problemă utilizând pointeri la variabilele structură.*

***Ex.4.: Programul este un exemplu de transmitere a structurilor catre funcții. Se definește funcția de tipărire numită f1(struct tip param) care are ca și parametru local o variabilă de tip structură.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>

//definirea unui tip de structura global ca sa poata fi
//"vizibil" si din functia main si din functia f1
struct tip  {int a,b;    char ch; };
void f1(struct tip param); //declarare prototip functie
int main()
{struct tip arg; arg.a=1000; arg.b=1500;
f1(arg); //transmiterea structurii catre functie
return 0;}
void f1(struct tip param)   //definire functie
{   printf("arg.a=%d, arg.b=%d\n", param.a, param.b);}
```

**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
using namespace std;
//definirea unui tip de structura global ca sa poata fi
//"vizibil" si din functia main si din functia f1
struct tip  {int a,b;    char ch; };
void f1(struct tip param); //declarare prototip functie
int main()
{struct tip arg; arg.a=1000; arg.b=1500;
f1(arg); //transmiterea structurii catre functie
return 0;}
void f1(struct tip param)   //definire functie
{ cout<<"arg.a="<<param.a<<",arg.b="<<param.b<<endl;}
```

***Rezultate:***

```
arg.a=1000, arg.b=1500
```

***Aplicație:***

*Să se modifice programul, astfel încât să se citească un caracter ch de la tastatură și dacă ch='y' să se apeleze o funcție care calculează rezultatul expresiei $\sqrt{a^2+b^2}$ unde a și b sunt componentele structurii arg.*

***Ex.5 Programul definește 2 variabile de tip (structură) student cu câmpurile de mai jos și se inițializează cu diverse valori care apoi vor fi afișate utilizând pointeri.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
int main()
{typedef struct {
char name[20];
        int age, year;
        long int id;
        double nota1, nota2, nota3, nota4;} student;
student s1={"Popescu Gheorghe",19,2,2021,10.,9.5,8.4,8.7},
s2={"Caprar Ioan",18,1,1018,8.5,5.6,5.4,7.},*p1,*p2;
p1=&s1;p2=&s2 ; printf("Tiparirea studentilor fara pointeri\n");
printf("nume student: %s\nvirsta: %d\nanul de studiu:%d\ncodul identificare: %ld\nmedia: %lf\n", s1.name, s1.age,
s1.year, s1.id,(s1.nota1+s1.nota2+s1.nota3+s1.nota4)/4); //atentie linie continuata!
printf("**********************************\n");
printf("nume student: %s\nvirsta: %d\nanul de studiu:%d\ncodul identificare: %ld\nmedia: %lf\n", s2.name, s2.age,
s2.year, s2.id,(s2.nota1+s2.nota2+s2.nota3+s2.nota4)/4); //atentie linie continuata!
printf("**********************************\n"); printf("Tiparirea studentilor cu pointeri\n");
printf("nume student: %s\nvirsta: %d\nanul de studiu: %d\ncodul identificare: %ld\nmedia: %lf\n", p1->name, p1->age,
p1->year,  p1->id,(p1->nota1+p1->nota2+p1->nota3+p1->nota4)/4);//atentie linie continuata!
printf("**********************************\n");
printf("nume student: %s\nvirsta: %d\nanul de studiu:%d\ncodul identificare: %ld\nmedia: %lf\n", p2->name, p2->age,
p2->year, p2->id, (p2->nota1+p2->nota2+p2->nota3+p2->nota4)/4); //atentie linie //continuata!
printf("**********************************\n");
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include<conio.h>
using namespace std;
int main()
{typedef struct {
char name[20];
        int age, year;
        long int id;
        double nota1, nota2, nota3, nota4;} student;
student s1={"Popescu Gheorghe",19,2,2021,10.,9.5,8.4,8.7}, s2={"Caprar Ioan",18,1,1018,8.5,5.6,5.4,7.},*p1,*p2;
p1=&s1;p2=&s2 ;
cout<<"Tiparirea studentilor fara pointeri"<<endl;
cout<<"nume student: "<<s1.name<<endl<<"virsta: "<<s1.age<<endl<<"anul de studiu: "<<s1.year<<endl<<"codul
identificare: "<<s1.id<<endl<<"media: "<<(s1.nota1+s1.nota2+s1.nota3+s1.nota4)/4<<endl; //atentie linie continuata!
cout<<"**********************************"<<endl;
cout<<"nume student: "<<s2.name<<endl<<"virsta: "<<s2.age<<endl<<"anul de studiu: "<<s2.year<<endl<<"codul
identificare: "<<s2.id<<endl<<"media: "<<(s2.nota1+s2.nota2+s2.nota3+s2.nota4)/4<<endl;  //atentie linie continuata!
cout<<"**********************************"<<endl;
cout<<"Tiparirea studentilor cu pointeri"<<endl;
cout<<"nume student: "<<p1->name<<endl<<"virsta: "<<p1->age<<endl<<"anul de studiu: "<<p1->year<<endl<<"codul
identificare: "<<p1->id<<endl<<"media: "<<(p1->nota1+p1->nota2+p1->nota3+p1->nota4)/4<<endl; //atentie linie
continuata!
cout<<"**********************************"<<endl;
cout<<"nume student: "<<p2->name<<endl<<"virsta: "<<p2->age<<endl<<"anul de studiu: "<<p2->year<<endl<<"codul
identificare: "<<p2->id<<endl<<"media: "<<(p2->nota1+p2->nota2+p2->nota3+p2->nota4)/4<<endl;//atentie linie
//continuata!
cout<<"**********************************"<<endl;
return 0;}
```

***Rezultate:***

```
Tiparirea studentilor fara pointeri
nume student: Popescu Gheorghe
virsta: 19
anul de studiu:2
codul identificare: 2021
media: 9.150000
**********************************
nume student: Caprar Ioan
virsta: 18
anul de studiu:1
codul identificare: 1018
media: 6.625000
**********************************
Tiparirea studentilor cu pointeri
nume student: Popescu Gheorghe
virsta: 19
anul de studiu: 2
codul identificare: 2021
media: 9.150000
**********************************
nume student: Caprar Ioan
virsta: 18
anul de studiu:1
codul identificare: 1018
media: 6.625000
**********************************
```

***Aplicație:***

*Să se modifice programul astfel încât să se inițializeze datele prin citire de la tastatură și să se afișeze doar studenții care au media peste 8.50.*

***Ex.6. Programul preia de la tastatură o dată calendaristică și afișează data din ziua următoare. Data calendaristică este declarată sub forma unei structuri cu trei câmpuri.***

**Varianta in C**

```
#include <stdio.h>

struct data { //variabile globale
      int zi, luna, an;};
int zi[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,30,31 };
struct data azi, maine;
void getdataazi( void ); //prototip functie
void getdataazi(void )
{  int valid = 0;
while( valid == 0 )
  {printf("Introduceti  anul (2000-2019)-->");
   scanf("%d", &azi.an);
   if( (azi.an < 1990) || (azi.an > 2019) )
            printf("Invalid an\n");
         else valid = 1;}
valid = 0;
while( valid == 0 ) {
printf("Introduceti  luna (1-12)-->"); scanf("%d", &azi.luna);
if( (azi.luna < 1) || (azi.luna > 12) )
printf("Invalid luna\n");
else  valid = 1;}
valid = 0;
while( valid == 0 )
{printf("Introduceti  ziua (1-%d)-->", zi[azi.luna-1]);
           scanf("%d", &azi.zi);
           if( (azi.zi < 1) || (azi.zi > zi[azi.luna-1]) )
                  printf("Invalid zi\n");
           else valid = 1;  } }
int main()
{getdataazi();
maine = azi; maine.zi++;
if( maine.zi > zi[maine.luna-1] )
{        maine.zi = 1; maine.luna++;
                  if( maine.luna > 12 )
{    maine.an++; maine.luna = 1; }
}
printf("Data de maine:%02d:%02d:%02d\n",maine.zi,
maine.luna, maine.an ); //linie continuata
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
struct data { //variabile globale
      int zi, luna, an;};
int zi[] = { 31, 28, 31, 30, 31, 30, 31, 31,30,31,30,31 };
struct data azi, maine;
void getdataazi( void ); //prototip functie
void getdataazi(void )
{  int valid = 0;
while( valid == 0 )
  {cout<<"Introduceti  anul (2000-2019)-->";
   cin>>azi.an;
   if( (azi.an < 1990) || (azi.an > 2019) )
            cout<<"Invalid an"<<endl;
         else valid = 1;}
valid = 0;
while( valid == 0 ) {
cout<<"Introduceti  luna (1-12)-->"; cin>>azi.luna;
if( (azi.luna < 1) || (azi.luna > 12) )
cout<<"Invalid luna"<<endl;
else  valid = 1;}
valid = 0;
while( valid == 0 )
{cout<<"Introduceti  ziua (1-"<<zi[azi.luna-1]<<")-->";
           cin>>azi.zi;
           if( (azi.zi < 1) || (azi.zi > zi[azi.luna-1]) )
                  cout<<"Invalid zi"<<endl;
           else valid = 1;  } }
int main()
{getdataazi();
maine = azi; maine.zi++;
if( maine.zi > zi[maine.luna-1] )
{        maine.zi = 1; maine.luna++;
                  if( maine.luna > 12 )
{    maine.an++; maine.luna = 1; }
}
cout<<"Data de maine:"<<maine.zi<<":"<<maine.luna<<
":"<< maine.an<<endl; //linie continuata
return 0;}
```

***Rezultate:***

```
Introduceti  anul (2000-2019)-->2019
Introduceti  luna (1-12)-->01
Introduceti  ziua (1-31)-->05
Data de miine:06:01:2019
```

***Aplicație:***

*Să se modifice programul astfel încât să se testeze dacă anul este bisect*

***Ex.7. Programul definește o structură de tip angajat al unei companii (angaj) care conține o altă structură imbricată de tip adresă (adrese). Programul citește de la tastatură datele angajaților și calculează impozitul pe baza salarului brut (considerat cunoscut), după formula: impozit =40% din salariu brut.***

**Varianta in C**

```
#include <stdio.h>
int main()
{struct adrese
        {   char strada[40];
            char oras[20];
            int cod; };
struct angaj
        {char nume[30];
        struct adrese adr ;
        float salarbrut;
        float impozit;
        float salarnet ;     };
struct angaj pers1;
printf("Introduceti nume angajat:"); scanf("%30s", pers1.nume);
printf("\nIntroduceti adresa angajat:"); printf("\nIntroduceti strada: ");
scanf("%s", pers1.adr.strada);  printf("Introduceti orasul: ");
scanf("%s", pers1.adr.oras); printf("Introduceti cod: ");
scanf("%d", &pers1.adr.cod); printf("\nSalar brut pentru angajat: ");
scanf("%f", &pers1.salarbrut);printf("\n");
pers1.impozit=pers1.salarbrut*0.4;
pers1.salarnet=pers1.salarbrut-pers1.impozit;
printf("Nume:%s\nAdresa:\n%s,%s\nCod numeric:%4d\n", pers1.nume, pers1.adr.strada,pers1.adr.oras, pers1.adr.cod);
printf("Salar brut: %.2f\nImpozit:%.2f\nRest de plata:%.2f\n", pers1.salarbrut, pers1.impozit,pers1.salarnet); return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{struct adrese
        {   char strada[40];
            char oras[20];
            int cod; };
struct angaj
        {char nume[30];
        struct adrese adr ;
        float salarbrut;
        float impozit;
        float salarnet ;     };
struct angaj pers1;
cout<<"Introduceti nume angajat:"; cin>>pers1.nume;
cin.ignore();
cout<<"\nIntroduceti adresa angajat:"<<endl; cout<<"Introduceti strada: ";
cin.get(pers1.adr.strada,40);
cin.ignore(); cout<<"Introduceti orasul: ";
cin.get(pers1.adr.oras,20);
cin.ignore(); cout<<"Introduceti cod: ";
cin>>pers1.adr.cod; cout<<"\nSalar brut pentru angajat: ";
cin>>pers1.salarbrut; cout<<endl;
pers1.impozit=pers1.salarbrut*0.4;
pers1.salarnet=pers1.salarbrut-pers1.impozit;
cout<<"Nume:"<<endl<<pers1.nume<<"Adresa:"<<endl<<pers1.adr.strada<<","<<pers1.adr.oras<<endl<<"Cod
numeric:"<< pers1.adr.cod<<endl;//linie continuata
cout<<"Salar brut: "<<pers1.salarbrut<<endl<<"Impozit:"<<pers1.impozit<<endl<<"Rest de plata:"<<pers1.salarnet<<endl;
return 0;}
```

***Rezultate:***

```
Introduceti nume angajat:Popescu

Introduceti adresa angajat:
Introduceti strada: G.Baritiu
Introduceti orasul: Cluj-Napoca
Introduceti cod: 400118

Salar brut pentru angajat: 4000

Nume:Popescu
Adresa:
G.Baritiu,Cluj-Napoca
Cod numeric:400118
Salar brut: 4000.00
Impozit:1600.00
Rest de plata:2400.00
```

***Aplicație:***

*Să se modifice programul astfel încât să se calculeze și afișeze salarul net dacă impozitul este calculat cu formula SalarBrut*30%-500 pentru SalarBrut <SalarBrutMinimEconomie și SalarBrut*30% +500 dacă SalarBrut ≥SalarBrut MinimEconomie (2080 lei/luna).*

## PROBLEME PROPUSE

1. ***Se consideră un magazin de echipamente electronice, în care fiecare echipament reprezintă un articol specificat prin următoarele câmpuri:***
   - ***Denumire (alfanumeric, max.30 caractere). Ex. : Video Recorder Panasonic SV500***
   - ***Cod (alfanumeric, max 6 caractere). Ex. A254G9***
   - ***Garanție (numeric întreg, max. 2 cifre) . Ex. 3***
   - ***Preț (numeric real , max. 10 cifre). Ex. 4 500 000***

   ***Să se scrie programul C/C++ în care să se citească 3 articole de la tastatură iar tipărirea acestora să se realizeze, utilizând numai câmpul denumire şi preţul calculat în EURO (1 EURO= aprox.40.000 lei).***

2. ***Se consideră un magazin de tehnică de calcul, în care fiecare produs reprezintă un articol specificat prin următoarele câmpuri:***
   - ***Denumire (alfanumeric, max.30 caractere). Ex. : PC Compaq P910***
   - ***Tip (alfabetic, max.10 caractere). Ex. calculatoare***
   - ***Caracteristici (alfabetic, max.30 caractere). Ex. 800MHz, 10 GB HDD,CDROM***
   - ***Garanție (numeric întreg, max. 2 cifre) . Ex. 3***
   - ***Preț (numeric real , max. 5 cifre). Ex. 950***

   ***Preţul e specificat în EURO şi se introduce de la tastatură.***

   ***Să se scrie programul C/C++ în care să se introducă de la tastatură 3 articole de tipul specificat mai sus, să se calculeze preţul echivalent în lei şi să se afişeze articolele, utilizând formatul de mai jos:***

| ***Denumire*** | ***Pret (LEI)*** |
|---|---|
| ***PC Compaq P910*** | ***28 500 000 lei*** |
| ***HP LaserJet 5000*** | ***15 000 000 lei*** |
| ***...*** | |

3. ***Se consideră o bibliotecă, în care fiecare carte sau revistă reprezintă un articol specificat prin următoarele câmpuri:***
   - ***Titlu (alfanumeric, max.30 caractere). Ex. : The Theory of Electric Circuits***
   - ***Autori (alfabetic, max 30 caractere). Ex. Michael Douglas***
   - ***Editura (alfanumeric, max.30 caractere). Ex. Teora***
   - ***ISBN (alfanumeric, max 14 caractere). Ex. 973-9204-98-8***
   - ***Pret (numeric real , max. 10 cifre). Ex. 350 000***

   ***Să se scrie programul C/C++ care realizează citirea a 2 articole de la tastatură iar tipărirea acstora să se realizeze utilizând numai câmpul denumire, prețul în lei și în EURO (1 EURO= aprox.40.000 lei).***

4. ***Se consideră o papetărie, în care fiecare produs reprezintă un articol specificat prin următoarele câmpuri:***
   - ***Denumire (alfanumeric, max.15 caractere). Ex. : Caiet A4***
   - ***Caracteristici (alfanumeric, max.20 caractere). Ex. File veline***
   - ***Pret (numeric real , max. 10 cifre). Ex. 55 000***

   ***Sa se scrie programul C/C++ care realizează citirea a 2 articole de tipul specificat mai sus , de la tastatură iar tipărirea articolelor să se realizeze utilizând toate câmpurile citite iar la sfârşit să se afișeze preţul total al celor 2 articole.***

# Capitolul 8
# Tablouri de structuri.Uniuni, Enumerări,Câmpuri de Biți

În acest capitol sunt prezentate consideraţii teoretice şi probleme rezolvate privind definirea, utilizarea şi sortarea tablourilor de stucturi.

## CONSIDERAŢII TEORETICE

***Tablourile de structuri*** *se definesc astfel:*

- se defineşte mai întâi un tip de structură
- se declară o variabilă tablou de structuri de tipul structurii definite

Indicii tablourilor de structuri sunt inițializați implicit cu 0, similar cu indicii tablourilor de date standard.

**Ex.**: definirea unui tablou unidimensional de structuri :

```
#include <stdio.h>
void main()
{struct adrese
        { char nume[30];
        char strada[40];
        char oras[20];
        int cod; } ;
//definire tablou de structuri cu 100 elemente de tip adrese
struct adrese adr[100];
adr[2].cod=3400;
printf("cod=%d",adr[2].cod);}
```

Sortarea tablourilor de structuri se poate realiza utilizând orice metodă de sortare dintre cele utilizate la sortarea tablourilor de date standard (cu elemente intregi, reale, caracter, etc). Cele mai des utilizate metode de sortare sunt: Metoda Bublle Sort, Metoda Quick Sort, Metoda selectiei si Metoda insertiei.

În continuare sunt prezentate modul de declarare şi utilizare a uniunilor, enumerarilor, câmpurilor de biţi şi a altor tipuri de date definite de utilizator utilizând typedef .

**I) Uniunea**: este prin definiție o variabilă care poata păstra , la momente diferite , obiecte de tipuri și mărimi diferite. Aceasta variabilă va ocupa suficientă memorie ca să poată stoca cel mai mare dintre tipurile enumerate în componența ei.

**Declararea** unor **variabile de tip uniune** se poate realiza în una sau două instructiuni distincte. Daca se utilizează două instrucţiuni distincte: se declara mai întâi tipul uniunii si apoi variabilele de tip uniune.

**Declararea tipului uniune**:

```
union nume_tip_uniune
        { tip var1;
          tip var2;
          …
        } ;
```

**Declararea variabilelor de tip uniune**: **union nume_tip_uniune lista_variabile_uniune;**

Declararea tipului uniunii si a variabilelor de tip uniune **într-o singură instrucțiune**:

```
union nume_tip_uniune
        { tip var1;
          tip var2;
          …
         } lista_variabile_uniune;
```

Alocarea memoriei se realizează în mod diferit pentru structuri și uniuni. Astfel pentru structuri: zona de memorie ocupată este mai mare sau egală cu suma mărimilor elementelor sale (vezi Exemplu a.). Pentru uniuni: zona de memorie ocupată este egală cu mărimea celui mai mare membru al său (Exemplu b.).

**Ex. a.** : Determinarea mărimii zonei de memorie ocupate de structura x

```
struct s
 {  char ch;     //1 octet
    int i;       //2 octeti
    float f;     //4 octeti
 } x;
// sizeof (x) >=7 (=1+2+4)
```

**Ex. b.** : Determinarea dimensiunii zonei de memorie ocupate de uniunea, y:

```
union u
 {  char ch;  //1 octet
    int i; //2 octeti
            float f;      //4 octeti
 } y;
// sizeof (y) =4
```

Accesul la elementele uniunii se realizează similar cu accesul la elementele unei structuri:

- **cu operatorul punct (.)** : nume_uniune.member
- **cu operatorul sageată (->)** : pointer_uniune-> member

**Ex:** acces la elementele unei uniuni

```
union tip
          { int i;
            char ch;
          } cont;
union tip *p;
cont.i=10;
p->i=10;
```

**II. Enumerare**: este prin definiție un set de constante care specifică toate valorile pe care le poate lua variabila de acel tip.

**Declarare variabilă de tip enumerare**: similar cu declararea structurilor

**enum [nume_enum] { lista enum } lista_variabile_enum ;**

**typedef enum [nume_enum] {lista enum} nume_tip;**

- **nume_ enum** este numele noului tip de date utilizator și este opțional
- **lista enum** este considerată listă de constante de tip întreg, primul element din listă are valoarea 0, al doilea valoarea 1, ș.a.m.d. , dacă nu se inițializează cu alte valori
- **lista_variabile_enum** este lista variabilelor de tipul nume_enum

În acest mod se declară variabile de tipul enumerare, tipul enumerare permițând definirea unei liste de constante întregi cu nume în vederea folosirii de nume sugestive pentru valori numerice.

**Ex.1:** declarare variabilă tip enumerare numită bani de tip monede

```
enum monede {dolar,marca,leu,yen,forint } ;
//declarare tip enumerare
enum monede bani;
//declarare variabila enumerare
```

**Ex, 2:** declarare variabilă tip enumerare numită logic de tip Boolean

```
enum Boolean {false, true} logic;
//false=0, true=1,
//se poate utiliza in expresii conditionale: logic==false sau logic == true
```

**Ex. 3:** declarare variabilă tip enumerare fără specificarea nume_enum

```
enum { ileg,ian,feb,mar,apr,mai,iun,iul,aug,sep,oct,nov,dec} luna ;
//expresii echivalente: luna =3;  luna=mar; (pentru ca mar=3)
//sau
enum {ian=1,feb,mar,apr,mai,iun,iul,aug,sep,oct,nov,dec} luna;
```

**Accesul la elementele** enumerării se poate realiza direct, utilizând numele și numărul de ordine din lista de enumerare .

**Ex.**: declarare variabila tip enumerare numită bani de tip monede

```
enum monede
        {dolar,marca,leu,yen,forint } ; //declarare tip  enumerare
enum monede bani;   //declarare variabila enumerare
//instructiuni permise
bani=leu;   //echivalent cu bani =2 pentru ca leu=2
if (bani==forint) printf("Banul este un forint")
printf("%d, %d", dolar,leu);   // va tipari valorile 0,2
```

**Inițializarea variabilelor de tip enumerare**

Implicit elementele din lista enum sunt inițializate cu valori pornind de la 0,1,… Inițializarea elementelor cu alte valori decât cele implicite se face utilizând semnul egal urmat de o valoare întreagă, modificându-se și valorile elementelor ce urmează după valoarea inițializată

**Ex.**: inițializare elemente enumerare:

```
enum monede
        {dolar,marca,leu=100,yen,forint } ;
enum monede bani;
printf("%d, %d, %d,%d,%d", dolar,marca,leu,yen,forint);
// va tipari valorile 0,1,100,101,102
```

Elementele din lista de enumerare nu sunt șiruri de caractere ci sunt o etichetă pentru valori întregi.

Operatorul **typedef** permite definirea unor tipuri particulare definite de utilizatori. Formatul de definire a unui tip nou de date este : **typedef tip nume_nou**

unde **tip** = orice tip de date existent
**nume_nou** = numele nou dat tipului tip

**Ex.**: declarație de tip float

```
typedef float bilant; //bilant este un alt nume pentru tipul float
bilant scadent;       //se declara variabila scadent de tipul bilant adica float
typedef bilant total;// total  este un alt nume pentru tipul bilant adica
                          //pentru tipul float;
```

Nu se creează de fapt nici un tip nou de date , ci numai un nou nume pentru un tip de date existent.

**III. CÂMPURI DE BIȚI**

**Câmpul de biți**: este un element al unei structuri care cuprinde unul sau mai multi biți adiacenți. Câmpurile de biți se pot accesa prin nume, unul sau mai multi biți dintr-un octet sau cuvânt și se pot grupa formând o structură .

**Formatul de declarare a unui câmp de biți** este :

```
struct nume_struct {
        tip nume1: lungime;
        tip nume2: lungime;
        …
        tip nume N: lungime;
        } lista_variabile;
```

unde **tip** = tipul câmpului de biți ce poate fi int,unsigned sau signed.
**lungime** = nr. de biți dintr-un câmp
Câmpul de biți permite accesul la un singur bit. Câmpul de biți cu lungimea 1 trebuie declarat de tip unsigned pentru că un singur bit nu poate avea semn. Câmpurile de biți sunt utilizate frecvent pentru analiza intrării de la un echipament hardware

Restricții de utilizare a variabilelor de tip câmp de biți:

- Nu se poate obține adresa unui câmp de biți cu operatorul &
- Nu pot fi utilizate într-o matrice
- Există restricții de rulare de la stânga la dreapta și invers care diferă de la echipament la echipament

**Ex.** : câmp de biți utilizat în cadrul unei structuri. Se definește o înregistrare într-o bază de date despre un angajat care folosește numai un octet pentru a păstra 3 informații:

- statutul angajatului,
- dacă a lucrat în luna respectivă si
- impozitul

```
struct angajat {
        struct adr adrese ;
        float salar ;
        unsigned activ:   1     //statut angajat: activ sau intrerupt
        unsigned orar:    1     //plata orara
        unsigned impozit:1      //impozit rezultat
};
```

Accesul la elementele câmpurilor de biți se realizează similar cu accesul la elementele unei structuri utilizând operatorul punct: **nume_struct.nume_camp**

## PROBLEME REZOLVATE

***Ex.1. Programul definește o variabilă tablou unidimensional de structuri, și realizează următoarele operații:***

- ***citește numărul studenților (dimensiunea n a tabloului), numele, prenumele și 2 note pentru fiecare student***
- ***calculează media aritmetică a notelor pentru fiecare student***
- ***afișează studenții sortați prin Metoda Bulelor în ordinea descrescătoare a mediilor.***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
struct student {
            char name[20];
            char prenume[20];
            double nota1;
            double nota2;
            double media;} s[100], aux[100];
int i=0,n,k;
int  main (void)
{printf("Introduceti numarul de studenti:");
scanf("%d", &n);
for (i=0;i<n;i++)
{            printf("\nIntroduceti numele studentului %d:", i+1);   scanf("%s", s[i].name);
             printf("Introduceti prenumele studentului %d:", i+1); scanf("%s",s[i].prenume);
             printf("Introduceti nota1 a studentului %d:",i+1); scanf("%lf", &s[i].nota1);
             printf("Introduceti nota2 a studentului %d:",i+1); scanf("%lf", &s[i].nota2);}
printf("\nLista studentilor:\n");
for (i=0;i<n;i++)
 {s[i].media = (s[i].nota1+s[i].nota2)/2;
printf("%8s %6s, nota1=%5.2lf, nota2=%5.2lf, media= %5.2lf \n", s[i].name, s[i].prenume,s[i].nota1, s[i].nota2, s[i].media);}
printf("\nStudentii sortati in ordinea descrescatoare a mediilor:\n");
do {k=0;
for (i=0;i<n-1;i++)
    {       if (s[i].media<s[i+1].media)
                        {           aux[i]=s[i] ;
                                    s[i]=s[i+1];
                                    s[i+1]=aux[i];k=1;}    } }
while (k);
for (i=0;i<n;i++)
{printf("%8s %6s, nota1=%5.2lf, nota2=%5.2lf, media=%5.2lf \n", s[i].name, s[i].prenume, s[i].nota1, s[i].nota2,s[i].media);}
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
using namespace std;
struct student {
            char name[20];
            char prenume[20];
            double nota1;
            double nota2;
            double media;} s[100], aux[100];
int i=0,n,k;
int  main (void)
{cout<<"Introduceti numarul de studenti:";
cin>>n;
for (i=0;i<n;i++)
{             cout<<"\nIntroduceti numele studentului "<< i+1<<":";           cin>>s[i].name;
              cout<<"Introduceti prenumele studentului "<< i+1<<":"; cin>>s[i].prenume;
              cout<<"Introduceti nota1 a studentului "<<i+1<<":"; cin>>s[i].nota1;
              cout<<"Introduceti nota2 a studentului "<<i+1<<":"; cin>>s[i].nota2;}
cout<<endl<<"Lista studentilor:"<<endl;
for (i=0;i<n;i++)
{s[i].media = (s[i].nota1+s[i].nota2)/2;
cout<<s[i].name<<", "<<s[i].prenume<<", nota1="<<s[i].nota1<<", nota2="<<s[i].nota2<<", media="<<  s[i].media<<endl;}
cout<<"\nStudentii sortati in ordinea descrescatoare a mediilor:"<<endl;
do {k=0;
for (i=0;i<n-1;i++)
    {       if (s[i].media<s[i+1].media)
                        {           aux[i]=s[i] ;
                                    s[i]=s[i+1];
                                    s[i+1]=aux[i];k=1;}     }  }
while (k);
for (i=0;i<n;i++)
{cout<<s[i].name<<", "<<s[i].prenume<<", nota1="<<s[i].nota1<<", nota2="<<s[i].nota2<<", media="<<  s[i].media<<endl;}
return 0;}
```

***Rezultate:***

```
Introduceti numarul de studenti:3

Introduceti numele studentului 1:Pop
Introduceti prenumele studentului 1:Ana
Introduceti nota1 a studentului 1:7.5
Introduceti nota2 a studentului 1:8.5

Introduceti numele studentului 2:Balc
Introduceti prenumele studentului 2:Alex
Introduceti nota1 a studentului 2:9
Introduceti nota2 a studentului 2:10

Introduceti numele studentului 3:Zahir
Introduceti prenumele studentului 3:Abdul
Introduceti nota1 a studentului 3:10
Introduceti nota2 a studentului 3:7.5

Lista studentilor:
     Pop    Ana, nota1= 7.50, nota2= 8.50, media=  8.00
    Balc   Alex, nota1= 9.00, nota2=10.00, media=  9.50
   Zahir  Abdul, nota1=10.00, nota2= 7.50, media=  8.75

Studentii sortati in ordinea descrescatoare a mediilor:
    Balc   Alex, nota1= 9.00, nota2=10.00, media= 9.50
   Zahir  Abdul, nota1=10.00, nota2= 7.50, media= 8.75
     Pop    Ana, nota1= 7.50, nota2= 8.50, media= 8.00
```

***Aplicație:***

*Să se realizeze sortarea studenților după medii , doar dacă au media mai mare decât 8.50.*

***Ex.2. Programul citește 3 date calendaristice de la tastatură care au anul peste 2000 și le memorează într-un tablou unidimensional de structuri. Se vor afișa datele pentru care anul aparţine intervalului (2004, 2019).***

**Varianta in C**

```
#include <stdio.h>
struct  data  {       //definire variabila  globala de tip structura
        int ziua, luna, an;   };
int main()
{ struct data date[3];
int i; printf("Introduceti 3 date calendaristice\n");
for( i = 0; i < 3; ++i )
{    printf("\nData calendaristica %d:(zz ll aa )",i+1 );
     scanf("%d %d %d", &date[i].ziua, &date[i].luna, &date[i].an );}
printf("Se verifica daca anul este in intervalul [2004,2019]\n");
for( i = 0; i < 3; ++i )
{    if (date[i].an>04 && date[i].an<19)
    printf("Data calendaristica %d este in interval\n",i+1);}
return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
struct  data  {       //definire variabila  globala de tip structura
        int ziua, luna, an;   };
int main()
{ struct data date[3];
int i; cout<<"Introduceti 5 date calendaristice"<<endl;
for( i = 0; i < 3; ++i )
{    cout<<"\nData calendaristica "<<i+1<<":(zz ll aa)";
    cin>>date[i].ziua>>date[i].luna>>date[i].an;}
cout<<"Se verifica daca anul este in intervalul [2004,2019]"<<endl;
for( i = 0; i < 3; ++i )
{    if (date[i].an>04 && date[i].an<19)
cout<<"Data calendaristica cu nr. "<<i+1<<" este in interval"<<endl;}
return 0;}
```

***Rezultate:***

```
Introduceti 5 date calendaristice

Data calendaristica 1:(zz ll aa)15 03 18

Data calendaristica 2:(zz ll aa)12 02 03

Data calendaristica 3:(zz ll aa)19 12 19
Se verifica daca anul este in intervalul [2004,2019]
Data calendaristica cu nr. 1 este in interval
```

***Aplicație:***

*Să se modifice programul astfel încât să se verifice dacă datele introduse sunt într-un interval citit de la tastatură.*

***Ex.3. Programul citește nr.de angajați și datele lor: nume,prenume, adresa,cod numeric, salar brut și calculează salarul net și impozitul fiecărui angajat afișând la sfârșit un tabel tip stat de plată.***

**Varianta in C**

```
#include <stdio.h>
int main()
{struct adrese
        {   char strada[40];
            char oras[20];
            int cod; };
struct angaj
{   char nume[30];
            char prenume[30];
            struct adrese adr ;
            float salarbrut;
            float impozit;
            float salarnet ;};
struct angaj pers[100]; int i,n;
float sumabrut=0.,sumanet=0.,sumaimp=0.;
printf ("Introduceti nr. de angajati: "); scanf("%d", &n);
for (i=0; i<n;i++){
        printf("Introduceti numele angajat %d:", i+1);
        scanf("%10s", pers[i].nume);
        printf("Introduceti prenume angajat %d:", i+1);
        scanf("%10s", pers[i].prenume);
        printf("Introduceti adresa angajat %d", i+1);
        printf("\nIntroduceti strada: ");
        scanf("%10s", pers[i].adr.strada);
        printf("Introduceti orasul: ");
        scanf("%s", pers[i].adr.oras);
        printf("Introduceti cod: ");
        scanf("%d", &pers[i].adr.cod);
        printf("Introduceti salarul brut: ");
        scanf("%f", &pers[i].salarbrut);   printf("\n");
        pers[i].impozit=pers[i].salarbrut*0.4;
        pers[i].salarnet=pers[i].salarbrut-pers[i].impozit;
        sumabrut+=pers[i].salarbrut;
        sumanet+=pers[i].salarnet;
        sumaimp+=pers[i].impozit;}
printf("*************************************************************************\n");
printf("                Tabel salarii                                  \n");
printf("*************************************************************************\n");
printf("Nr.%-8s%-6s %-14s %10s %10s %10s \n", "Nume", "Prenume", "Adresa", "Salar Brut", "Impozit", "Salar Net");
for (i=0; i<n;i++)
 { printf("%-2d.%-8s %-6s %4s,%4s,%4d %10.f %10.f %10.f\n",(i+1), pers[i].nume,  pers[i].prenume, pers[i].adr.strada,
pers[i].adr.oras, pers[i].adr.cod, pers[i].salarbrut, pers[i].impozit, pers[i].salarnet);
//atentie linie continuata !
printf("\n");} printf("%33s %10.f %10.f %10.f\n", "Total : ", sumabrut, sumaimp, sumanet);  return 0;}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{struct adrese
        {   char strada[40];
            char oras[20];
            int cod; };
struct angaj
{   char nume[30];
            char prenume[30];
            struct adrese adr ;
            float salarbrut;
            float impozit;
            float salarnet ;};
struct angaj pers[100]; int i,n;
float sumabrut=0.,sumanet=0.,sumaimp=0.;
cout <<"Introduceti nr. de angajati: "; cin>>n;
for (i=0; i<n;i++){
        cout<<"Introduceti numele angajat "<<i+1<<":";      cin>>pers[i].nume;
        cout<<"Introduceti prenume angajat "<<i+1<<":";     cin>>pers[i].prenume;
        cout<<"Introduceti adresa angajat "<<i+1<<":";
        cout<<"\nIntroduceti strada: ";      cin>>pers[i].adr.strada;
        cout<<"Introduceti orasul: ";        cin>> pers[i].adr.oras;
        cout<<"Introduceti cod: ";           cin>>pers[i].adr.cod;
        cout<<"Introduceti salarul brut: ";  cin>>pers[i].salarbrut;  cout<<endl;
        pers[i].impozit=pers[i].salarbrut*0.4;  pers[i].salarnet=pers[i].salarbrut-pers[i].impozit;
        sumabrut+=pers[i].salarbrut;
        sumanet+=pers[i].salarnet;
        sumaimp+=pers[i].impozit;}
cout<<"*************************************************************************"<<endl;
cout<<"              Tabel salarii                      "<<endl;
cout<<"*************************************************************************"<<endl;
cout<<"Nr.  Nume      Prenume      Adresa      Salar      Brut      Impozit      Salar Net"<<endl;
for (i=0; i<n;i++)
 { cout<<(i+1)<<"    "<<pers[i].nume<<"    "<<pers[i].prenume<<"    "<<pers[i].adr.strada<<","<<pers[i].adr.oras<<"
"<<pers[i].adr.cod<<"    "<<pers[i].salarbrut<<"    "<< pers[i].impozit<<"    "<<pers[i].salarnet;
//atentie linie continuata !
cout<<endl;}
cout<< "                                Total : " <<sumabrut<<"       "<<sumaimp<<"       "<<sumanet<<endl;
 return 0;}
```

***Rezultate:***

```
Introduceti nr. de angajati: 2
Introduceti numele angajat 1:Popa
Introduceti prenume angajat 1:Ioan
Introduceti adresa angajat 1
Introduceti strada: Cioran
Introduceti orasul: Cluj
Introduceti cod: 40020
Introduceti salarul brut: 4000

Introduceti numele angajat 2:Rusu
Introduceti prenume angajat 2:Anca
Introduceti adresa angajat 2
Introduceti strada: Cioran
Introduceti orasul: Cluj
Introduceti cod: 40020
Introduceti salarul brut: 6000
```

***Aplicație:***

*Să se modifice programul astfel încât să se realizeze sortarea în ordine alfabetică a angajaților după nume, sau oraș .*

***Ex.4. Programul citește numele, prenumele,data nașterii, codul personal pentru n persoane, calculează vârsta fiecărei persoane și apoi afișează toate datele, utilizând alocarea dinamică a memoriei.***

**Varianta in C**

```
#include <stdio.h>
#include <malloc.h>
struct data {
        int ziua;
        int luna;
        int an; };
struct pers {
char nume[15];
            char prenume[20];
            int cod;
            struct data datan;
            int varsta; };
struct pers p[20],*dp;
void citire(struct pers *);
void afisare(struct pers *);
int main()
{int i,n;
dp=&p[0];
printf("\n nr. angajati:");scanf("%d",&n);
dp=(struct pers *)malloc(n*sizeof(struct pers));
if (dp==NULL)
{ printf("nu exista suficienta memorie!\n");   return 0;}
printf("Introduceti datele fiecarui angajat:\n");
for (i=0;i<n;i++)
{printf("Datele persoanei %d:\n", i+1); citire(dp+i);}
printf("Afisarea datelor introduse:\n");
for (i=0;i<n;i++)
{printf("Datele persoanei %d:\n", i+1);  afisare(dp+i);}
if (dp) free(dp);
return 0;
}
void citire(struct pers *dp)
{           printf("Nume:"); scanf("%s", dp->nume);
            printf("Prenume:");  scanf("%s", dp->prenume);
            printf("Cod:"); scanf("%d", &dp->cod);
            printf("ziua nasterii:");  scanf("%d", &dp->datan.ziua);
            printf("luna nasterii:");  scanf("%d", &dp->datan.luna);
            printf("anul nasterii:");  scanf("%d", &dp->datan.an);
}
void afisare(struct pers *dp)
{           printf("Nume:%s",dp->nume);
            printf("\nPrenume:%s",dp->prenume);
            printf("\nCod: %d",dp->cod);
            printf("\ndata nasterii:%d/%d/%d",dp->datan.ziua,dp->datan.luna, dp-> datan.an ); //
            printf("\nVarsta: %d ani\n",2019-dp->datan.an );
}
```

**Varianta in C++**

```
#include <iostream>
#include <malloc.h>
using namespace std;
struct data {
        int ziua; //definitie globala de tip
        int luna;
        int an; };
struct pers {
char nume[15];
       char prenume[20];
       int cod;
       struct data datan;
       int varsta; };
struct pers p[20],*dp;
void citire(struct pers *);
void afisare(struct pers *);
int main()
{int i,n;
dp=&p[0];
cout<<"\n nr. angajati:";cin>>n;
dp=(struct pers *)malloc(n*sizeof(struct pers));
if (dp==NULL)
{ cout<<"nu exista suficienta memorie!"<<endl;   return 0;}
cout<<"Introduceti datele fiecarui angajat:"<<endl;
for (i=0;i<n;i++)
{cout<<"Datele persoanei"<< i+1<<":"; citire(dp+i);}
cout<<"Afisarea datelor introduse:"<<endl;
for (i=0;i<n;i++)
{cout<<"Datele persoanei"<<i+1<<":"<<endl;  afisare(dp+i);}
if (dp) free(dp);
return 0;}
void citire(struct pers *dp)
{           cout<<"Nume:"; cin>> dp->nume;
            cout<<"Prenume:";  cin>>dp->prenume;
            cout<<"Cod:"; cin>>dp->cod;
            cout<<"ziua nasterii:";  cin>>dp->datan.ziua;
            cout<<"luna nasterii:";  cin>>dp->datan.luna;
            cout<<"anul nasterii:";  cin>>dp->datan.an;
}
void afisare(struct pers *dp)
{           cout<<"Nume:"<<dp->nume<<endl;
            cout<<"Prenume:"<<dp->prenume<<endl;
            cout<<"Cod:"<<dp->cod<<endl;
            cout<<"data nasterii:"<<dp->datan.ziua<<"/"<<dp->datan.luna<<"/"<< dp-> datan.an<<endl ; //
            cout<<"Varsta:"<<2019-dp->datan.an<<"  ani"<<endl;
}
```

***Rezultate:***

```
 nr. angajati:2
Introduceti datele fiecarui a
Datele persoanei1:Nume:pop
Prenume:vasile
Cod:1212
ziua nasterii:02
luna nasterii:12
anul nasterii:1966
Datele persoanei2:Nume:rusu
Prenume:ana
Cod:2122
ziua nasterii:12
luna nasterii:12
anul nasterii:1971
Afisarea datelor introduse:
Datele persoanei1:
Nume:pop
Prenume:vasile
Cod:1212
data nasterii:2/12/1966
Varsta:53  ani
Datele persoanei2:
Nume:rusu
Prenume:ana
Cod:2122
data nasterii:12/12/1971
Varsta:48  ani
```

***Aplicație:***

*Să se modifice programul astfel încât să se verifice dacă data nașterii este introdusă corect și să se ordoneze persoanele în ordinea descrescătoare a vârstei.*

***Ex.5. Să se scrie un program în care se declară o uniune cu 3 câmpuri (elemente) cărora li se atribuie valori. (în memorie se pastrează la un moment dat un singur element al uniunii).***

**Varianta in C**

```
#include <stdio.h>
#include <string.h>

int main()
{union {char nume[10];  //declararea variabilei uniune
char prenume[20] ;
char lit;} uniune;
//atribuirea de valori elementelor unei uniuni
printf("Modificarea elementelor uniunii\n");
printf("prin atribuirea de valori elementelor\n\n");
printf(" nume\tprenume\tlitera\n");
strcpy(uniune.nume, "Popescu"); printf("%10s",
uniune.nume);
strcpy(uniune.prenume, "Maria");
printf(" %10s", uniune.prenume);
uniune.lit='T';  printf("\t%c\n",uniune.lit);
printf("%10s %10s\t%c\n",uniune.nume,uniune.prenume,
uniune.lit);
strcpy(uniune.prenume, "Mia");
printf("%10s %10s\t%c\n",uniune.nume,uniune.prenume,
uniune.lit);
uniune.lit='L';
printf("%10s %10s\t%c\n",uniune.nume,uniune.prenume,
uniune.lit); //linie continuata
return 0;
}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{union {char nume[10];  //declararea variabilei uniune
char prenume[20] ;
char lit;} uniune;
//atribuirea de valori elementelor unei uniuni
cout<<"Modificarea elementelor uniunii"<<endl;
cout<<"prin atribuirea de valori
elementelor"<<endl<<endl;
cout<<"nume\tprenume\tlitera"<<endl;
strcpy(uniune.nume, "Popescu  ");
cout<<uniune.nume;
strcpy(uniune.prenume, "Maria");
cout<< uniune.prenume;
uniune.lit='T';  cout<<"\t"<<uniune.lit<<endl;
cout<<uniune.nume<<"   "<<uniune.prenume<<"\t"<<
uniune.lit<<endl;
strcpy(uniune.prenume, "Mia");
cout<<uniune.nume<<"
"<<uniune.prenume<<"\t"<< uniune.lit<<endl;
uniune.lit='L';
cout<<uniune.nume<<"
"<<uniune.prenume<<"\t"<< uniune.lit<<endl;
return 0;}
```

***Rezultate:***

```
Modificarea elementelor uniunii
prin atribuirea de valori elementelor

      nume      prenume litera
   Popescu      Maria   T
     Taria      Taria   T
       Mia        Mia   M
       Lia        Lia   L
```

***Aplicație:***

*Să se modifice programul astfel încât să se introducă de la tastatură toate câmpurile uniunii.*

***Ex.6. În programul următor se declară un tablou unidimensional de maxim 10 uniuni, numit uniune[10], fiecare uniune conține 2 elemente care se inițializează cu valori introduse de la tastatură și ulterior se afișează. Se observă că nu se păstrează pentru fiecare element al uniunii decât ultima valoare de tip șir atribuită.***

**Varianta in C**

```
#include <stdio.h>
#include <string.h>
int main()
{union { char nume[10];
            char prenume[20] ;} uniune[10];
int i,n;
printf("n=");  scanf("%d", &n);
for(i=0;i<n;i++)
   {printf("\nNume %d:",i+1);
   scanf("%s",uniune[i].nume); printf("%s", uniune[i].nume) ;
   printf("\nPrenume %d:",i+1);scanf("%s",uniune[i].prenume);
   printf("%s",uniune[i].prenume);
   }
for(i=0;i<n;i++)
   {printf("\nNume %d: %s",i+1, uniune[i].nume);
   printf("\nPrenume %d: %s",i+1, uniune[i].prenume);
   }
return 0;
}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{union { char nume[10];
            char prenume[20] ;} uniune[10];
int i,n;
cout<<"n=";  cin>>n;
for(i=0;i<n;i++)
   {cout<<"\nNume " <<i+1<<":";
  cin>>uniune[i].nume; cout<<uniune[i].nume ;
  cout<<"\nPrenume " <<i+1<<":";
cin>>uniune[i].prenume;
  cout<<uniune[i].prenume;   }
for(i=0;i<n;i++)
   {cout<<"\nNume "
<<i+1<<":"<<uniune[i].nume;
   cout<<"\nPrenume "
<<i+1<<":"<<uniune[i].prenume;   }
return 0;
}
```

***Rezultate:***

```
n=2

Nume 1:Pop
Pop
Prenume 1:Ioan
Ioan
Nume 2:Rusu
Rusu
Prenume 2:Alina
Alina
Nume 1: Ioan
Prenume 1: Ioan
Nume 2: Alina
Prenume 2: Alina
```

***Aplicație:***

*Să se modifice programul astfel încât să se afișeze și inițiala numelui introdus pentru fiecare uniune*

***Ex.7. Programul este un exemplu de definire a unui tip de enumerare numit*** monede ***și a unei variabile de acest tip numite*** bani ***si ilustrează accesul la elementele variabilei și modul de tipărire a acestora.***

**Varianta in C**

```
#include <stdio.h>
int main()
{//declarare tip  enumerare
enum monede     {dolar,marca,leu,yen,forint } ;
//declarare variabila enumerare
enum monede bani;
bani=leu;
if (bani==forint) printf("Moneda este un forint\n");
printf("tiparirea unei variabile enumerare\n");
printf("are ca efect tiparirea valorilor numerice\nale
enumerarii ");
printf("si nu a numelor asociate \n");
printf("dolar:%d, marca:%d, leu:%d, yen:%d,
forint:%d\n",
dolar,marca,leu,yen,forint);   //atentie linie continuata!
return 0;
}
```

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{//declarare tip  enumerare
enum monede     {dolar,marca,leu,yen,forint } ;
//declarare variabila enumerare
enum monede bani; bani=leu;
if (bani==forint) cout<<"Moneda este un forint"<<endl;
cout<<"tiparirea unei variabile enumerare"<<endl;
cout<<"are ca efect tiparirea valorilor numerice\nale
enumerarii ";
cout<<"si nu a numelor asociate "<<endl;
cout<<"dolar:"<<dolar<<", marca:"<<marca<<",
leu:"<<leu<<", yen:"<<yen<<", forint:"<<forint<<endl;
//atentie linie continuata!
return 0;}
```

***Rezultate:***

```
tiparirea unei variabile enumerare
are ca efect tiparirea valorilor numerice
ale enumerarii si nu a numelor asociate
dolar:0, marca:1, leu:2, yen:3, forint:4
```

***Aplicație:***

*Să se modifice programul astfel încât declarația variabilei să fie următoarea:*

enum monede
{dolar,marca,leu=100,yen,forint } ;

*Ce se va afișa în acest caz?*

***Ex.8. Programul implementeaza o bază de date  cu calculatoare, fiecare articol cu câmpurile: denumire,  garanție, și preț. Se citesc articolele de la tastatură și  se afișeaza pe monitor.***

**Varianta in C**

```
#include <stdio.h>
#include <string.h>

int main()
{int i,n;
typedef struct {    char den[10];
                    int gar;   //garantie
                    double pret;}calculator;
calculator  j[10];
printf("Nr. de calculatoare:"); scanf("%d",&n);
printf("Introducerea calculatoarelor:");
for (i=0;i<n;i++)
{printf("\nCalculatorul nr. %d\n", i+1);
printf("Denumire:"); scanf("%s", j[i].den);
printf("Garantie:"); scanf("%d", &j[i].gar);
printf("Pret(lei):"); scanf("%lf", &j[i].pret);}
printf("\nAfisarea produselor introduse:\n");
for (i=0;i<n;i++)
{printf("%10s,%d,%5.2lf lei", j[i].den, j[i].gar, j[i].pret);}
printf("\n");
return 0;
}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{int i,n;
typedef struct {    char den[10];
                    int gar;   //garantie
                    double pret;}calculator;
calculator  j[10];
cout<<"Nr. de calculatoare:"; cin>>n;
cout<<"Introducerea calculatoarelor:";
for (i=0;i<n;i++)
{ cout<<"Calculatorul nr. "<<i+1<<endl;
cout<<"Denumire:"; cin>> j[i].den;
cout<<"Garantie:"; cin>>j[i].gar;
cout<<"Pret(lei):"; cin>>j[i].pret;}
cout<<endl<<"Afisarea produselor introduse:"<<endl;
for (i=0;i<n;i++)
{cout<<j[i].den<<", "<<j[i].gar<<","<<j[i].pret<<" lei";
cout<<endl;}
return 0;
}
```

***Rezultate:***

```
Nr. de calculatoare:2
Introducerea calculatoarelor:Calculatorul nr. 1
Denumire:HP
Garantie:2
Pret(lei):3000
Calculatorul nr. 2
Denumire:COMPAQ
Garantie:5
Pret(lei):4000

Afisarea produselor introduse:
HP, 2,3000 lei
COMPAQ, 5,4000 lei
```

***Aplicație:***

*Să se modifice programul astfel incat sa se afiseze doar articolele care au prețul in intervalul (1000,2500)*

***Ex.9. Programul realizează implementarea unei liste de tip STIVĂ (LIFO) prin alocare secvențială. Funcțiile implementate în program:***

- ***push() pentru punerea unui element pe stivă***
- ***pop() pentru scoaterea unui element de pe stivă***
- ***clear() pentru a goli o stivă***
- ***empty() pentru a determina dacă stiva e goală sau nu***
- ***full() pentru a determina dacă stiva e plină sau nu***
- ***top() permite accesul la elementul din vârful stivei fără a-l scoate din stivă, etc.***

***Stiva va fi implementată secvențial sub forma unui tablou unidimensional de tip int, numit stack, primul element din tablou stack[0] fiind elementul de la baza stivei.***

***Se utilizează un indice către primul element liber din stivă numit next, inițial stiva este goală, stack[0] este primul element liber din stivă, deci next=0***

**Varianta in C**

```
#include <stdio.h>
#  define MAX 100
int push(int x)      ;
int pop(void);
int top(void);
void clear(void);
int empty(void);
int full(void);
int stack[MAX];
int next=0;
int main()
{ int a=10,k=20,j;
puts("Introdu a=10 si k=20 pe stiva");
push(a); push(k);
printf("stiva are %d elemente\n", next);
puts("Scoate din lista ultimul element k=20\n");
j=pop();  printf("Elementul scos din stiva este %d\n", j);
printf("stiva are %d elemente\n", next);
j=top(); printf("Elementul din varful stivei este %d\n", j);
printf("stiva are %d elemente\n", next);
printf("golesc stiva!\n");
clear(); j=pop();
printf("stiva are %d elemente\n", next);return 0;}
int push(int x)
{if (next <MAX) {
stack[next++]=x; return(0);}
else return (1);}
int pop(void)
{if (next >0) return (stack[--next]);}
int top(void)
{if (next >0) return (stack[next-1]);}
void clear(void)
{next =0;}
int empty(void)
{return(!next) ;}
int full(void)
{return(next ==MAX);}
```

**Varianta in C++**

```
#include <iostream>
#  define MAX 100
using namespace std;
int push(int x)      ;
int pop(void);
int top(void);
void clear(void);
int empty(void);
int full(void);
int stack[MAX];
int next=0;
int main()
{ int a=10,k=20,j;
cout<<"Introdu a=10 si k=20 pe stiva";
push(a); push(k);
cout<<endl<<"stiva are "<<next<<"
elemente"<<endl;
cout<<"Scoate din lista ultimul element
k=20"<<endl;
j=pop();  cout<<endl<<"Elementul scos din stiva
este "<<j<<endl;
cout<<"stiva are "<<next<<" elemente"<<endl;
j=top(); cout<<"Elementul din varful stivei este
"<<j<<endl;
cout<<"stiva are "<<next<<" elemente"<<endl;
cout<<"golesc stiva!"<<endl;
clear(); j=pop();
cout<<"Stiva are "<<next<<" elemente"<<endl;
return 0;}
int push(int x)
{if (next <MAX) {
stack[next++]=x; return(0);             }
else return (1);}
int pop(void)
{if (next >0) return (stack[--next]);}
int top(void)
{if (next >0) return (stack[next-1]);}
void clear(void)
{next =0;}
int empty(void)
{return(!next) ;}
int full(void)
{return(next ==MAX);}
```

***Rezultate:***

```
Introdu a=10 si k=20 pe stiva
stiva are 2 elemente
Scoate din lista ultimul element k=20

Elementul scos din stiva este 20
stiva are 1 elemente
Elementul din varful stivei este 10
stiva are 1 elemente
golesc stiva!
stiva are 0 elemente
```

***Aplicație:***

*Să se adauge alte 3 elemente  pe stivă care ulterior să fie extrase pe rând din stivă, în mod similar cu programul de mai sus.*

## PROBLEME PROPUSE

1. ***Să se scrie un program care realizează citirea a 5 date calendaristice și memorarea acestora într-un tablou unidimensional de structuri. Se vor afișa datele pentru care anul este mai mare decât 2004.***
2. ***Să se scrie un program care citeşte numele, prenumele,data naşterii, codul personal pentru n persoane, calculează vârsta fiecarei persoane şi apoi afişează toate datele, utilizând alocarea dinamică a memoriei.***
3. ***Să se scrie un program în care se defineşte o variabilă tablou de structuri numită angajați de tip structură cu câmpurile: nume, adresa, cod numeric, salar net şi să se inițializeze tabloul cu n articole. Să se afişeze numai angajaţii care au salarul cuprins între 400 şi 1.000 RON.***
4. ***Să se scrie un program în care se defineşte o structură de tip catalog de carţi cu următoarele câmpuri: titlu, autor, editura, anul apariţiei. Să se definească o variabilă tablou de structuri de tipul catalog în care iniţializarea datelor şi afişarea lor să se realizeze atât prin intermediul câmpurilor cât şi prin intermediul unui pointer la această variabilă.***
5. ***Să se scrie un program în care se defineşte o variabilă tablou de structuri (magazin de tehnică de calcul) de tipul structură cu câmpurile:***
   - ***Denumire (alfanumeric, max.30 caractere). Ex. : PC Compaq P910***
   - ***Tip (alfabetic, max.10 caractere). Ex. calculatoare***
   - ***Caracterisitici (alfabetic, max.30 caractere). Ex. 800MHz, 10 GB HDD***
   - ***Garanție (numeric intreg, max. 2 cifre) . Ex. 3***
   - ***Preț în euro (numeric real , max. 5 cifre). Ex. 950***

   ***Se cere să se citească de la tastatură n articole cu formatul de structură de mai sus, să se calculeze preţul echivalent in RON și să se afișeze articolele, utilizând formatul de mai jos: Denumire Pret (RON) PC Compaq P910 2 850 RON .***

6. ***Se consideră o bază de date, de tipul unui magazin de echipamente electronice, în care fiecare echipament reprezintă un articol specificat prin următoarele câmpuri:***
   - ***Denumire (alfanumeric,max.30 caractere).Ex.:Video Recorder***
   - ***Cod (alfanumeric, max 6 caractere). Ex. A254G9***
   - ***Garanție (numeric intreg, max. 2 cifre) . Ex. 3***
   - ***Culoare (enumerare de maxim 5 culori) . Ex. alb, gri, negru, argintiu***
   - ***Preț (numeric real , max. 10 cifre). Ex. 450***

   ***Să se scrie programul în care să se citeasca n articole (n introdus de la tastatură) de tipul specificat mai sus, iar afişarea articolelor să se realizeze utilizând numai câmpul denumire şi preţul calculat în EURO la cursul oficial BNR din ziua respectivă (curs citit de la tastatură).***

# Capitolul 9
# Fișiere. Stream-uri

În acest capitol sunt prezentate consideraţii teoretice și probleme rezolvate privind definirea , clasificarea și utilizarea fișierelor respectiv a stream-urilor.

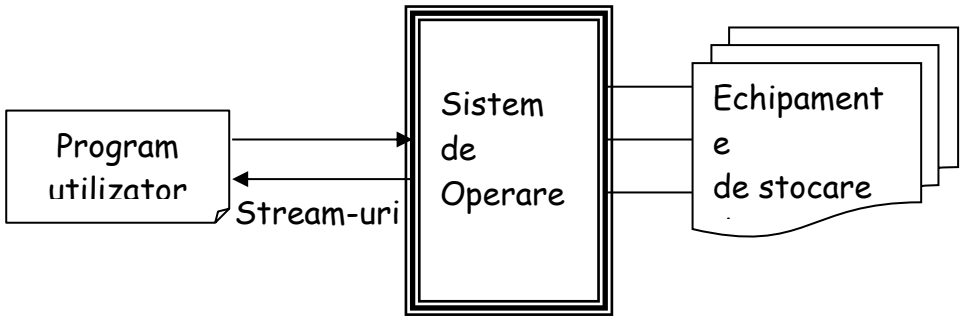
## CONSIDERAŢII TEORETICE

**Fișierul** este o colecție ordonată de înregistrări (articole) aflată pe un suport magnetic (instrument fizic efectiv) în care se stochează datele citite/scrise în programul utilizator. Fișierele în C++ au forma unei structuri numită FILE, care conține informații despre un anumit stream .

Mediile de programare C/C++ conțin funcțiile de I/O destinate manipulării structurii FILE, însă pentru utilizarea ei este necesară declararea unui pointer la această structură. Majoritatea funcțiilor specifice fișierelor sunt conținute în <stdio.h> și au un "f" adăugat în fața numelui . Ex. fscanf(), fprintf(), etc.

**Streamul** (flux) reprezintă o formă abstractă, independentă de tipul echipamentelor utilizate (terminale, drivere de disc, drivere de unități fizice, etc) care realizează legătura dintre programul utilizatorului și fișierele de I/O utilizate de acesta.

Prin intermediul stream-urilor se realizează operațiile de citire/scriere din/în fișiere. Stream-urile pot fi de tip text sau binare. Stream-urile realizează legătura dintre sistemul de operare și programul utilizator așa cum este ilustrat în figura de mai jos.




Fișierele (și streamurile) se clasifică în două mari categorii:

- **Fișiere (stream-uri) tip text** = secvență de caractere ASCII, organizată pe linii, terminată opțional de caracterul linie nouă. Ex. fiș. cu extensia .txt, .c, etc.
- **Fișiere (stream-uri) binare** = secvență ordonată de caractere (octeti). Ex. fiș. executabile cu extensia .exe, .bat, .com, etc.

Stream-urile standard utilizate în C si C++ sunt

- stdin: streamer de intrare (intrare standard) de tip text. Ex. tastatura
- stdout: streamer de ieșire (ieșire standard) de tip text. Ex.monitorul
- stderr: streamer de ieșire erori (ieșire standard erori) de tip text.

La lansarea în execuție a unui program C/C++ se deschid automat stream-urile specificate mai sus , iar la închiderea programului , acestea sunt închise.

Accesul la informația conținută în fișiere se obține prin intermediul următoarelor tipuri de operații:

- deschiderea fișierului respectiv
- citirea/scrierea informației

- căutarea unei anumite informații, alte operații
- închiderea fișierului

Indicatorul de poziție, la deschiderea fișierului se poziționează pe începutul de fișier, apoi se incrementează/decrementează pe parcursul parcurgerii fișierului iar în final este poziționat pe sfârșitul de fișier.

Modurile de tratare a fișierelor sunt:

**a. la nivel inferior (low level):**

- se utilizează mai rar pentru că depinde de sistemul de operare
- funcțiile utilizate la acest nivel sunt incluse în <io.h>, <stat.h>, <fcntl.h>. Funcții uzuale: open, creat, read, write, lseek, close

**b. la nivel superior (high level):**

- se utilizează mai frecvent
- funcțiile utilizate la acest nivel sunt incluse în <stdio.h> pentru C și <fstream.h> pentru C++ . Funcții uzuale: fopen, fclose, fputc, fgetc,fseek, fprintf, fscanf, feof, ferror, rewind, remove, fflush

**a) Funcții la nivel inferior:**

Funcția **open()** are următorul prototip:

**int open(const char *path, int access [,unsigned mode]);**

unde **path** = reprezintă calea spre fișier

**access** = variabila de tip întreg care indică modul de deschidere a fișierului.

Opțiunile și semnificația acestora sunt prezentate în tabelul de mai jos:

| access | Semnificatie |
|---|---|
| **O_RDONLY** | (numai) Permisiune de **citire** din fișier |
| **O_WRONLY** | (numai) Permisiune de **scriere** în fișier |
| **O_RDWR** | Permisiune de **citire și scriere** din/în fișier |
| **O_APPEND** | Permisiune de **adăugare** în fisier |
| **O_CREAT** | Permisiune de **creare** fișier. Dacă fișierul există nu are efect; dacă fisierul nu există este creat |
| **O_TRUNC** | Permisiune de **trunchiere** fișier. Dacă fișierul există lungimea sa e trunchiată la 0 |
| **O_BINARY** | Permisiune de **deschidere** a fișierului în mod **binar** |
| **O_TEXT** | Permisiune de **deschidere** a fișierului în mod **text** |

**mode** = este un parametru opțional și poate lua una dintre valorile specificate în tabelul de mai jos:

| mode | Semnificatie |
|---|---|
| **S_IWRITE** | Permisiunea de **scriere** în fișier este autorizată |
| **O_IREAD** | Permisiunea de **citire** din fișier este autorizată |
| **S_IREAD\|S_IWRITE** | Permisiune de **citire** și **scriere** din/în fișier autorizată |

Funcția **creat()** are următorul prototip:

**int creat(const char *path, int amode);**

unde **path** = calea spre noul fișier ce va fi creat

**amode** = variabila de tip întreg care indică modul de creare a fișierului și poate lua una din valorile specificate în tabelul de mai jos:

| amode | Semnificatie |
|---|---|
| **S_IWRITE** | Permisiunea de **scriere** în fișier este autorizată |
| **O_IREAD** | Permisiunea de **citire** din fișier este autorizată |
| **S_IREAD\|S_IWRITE** | Permisiune de **citire** și **scriere** din/în fișier autorizată |

Funcția **read()** are următorul prototip:

**int read(int handle, void *buf, unsigned len);**

unde **len** = nr de octeți ce se încearcă a fi citiți în bufferul de memorie **buf** din fișierul asociat cu **handle**

**Exemplu:**

```
void *buf;int handle, bytes;
...
handle=open("test.exe",O_RDONLY|O_BINARY,S_IWRITE|S_IREAD)
bytes=read(handle,buf,10);
...
```

Funcția **write()** are următorul prototip:

**int write(int handle, void *buf, unsigned len);**

unde **len** = nr de octeți ce se încearcă a fi scriși din bufferul de memorie **buf** în fișierul asociat cu **handle.**

Funcția **lseek()** are următorul prototip:

**int lseek(int handle, long offset, int fromwhere);**

Se poziționează cursorul în fișierul asociat cu **handle** la distanța **offset,** octeții începând de la **fromwhere.**

Funcția **close()** are următorul prototip:

**int close(int handle);**

Funcția close() închide fișierul asociat cu **handle**, returnând valoarea –1 în caz de eroare și 0 dacă operația s-a efectuat cu succes.

Un **pointer la fișier** este un pointer la informațiile despre fișierul respectiv: numele, starea și poziția curentă .

Declararea unui pointer la un fișier se realizează după formatul :

**FILE *fp ;**

unde: **FILE** este un cuvânt rezervat în C, și este numele unui tip structură prin care se definește un fișier, iar **fp** este numele variabilei pointer la fișierul respectiv.

**b) Funcții la nivel superior:**

Aceste funcții sunt prezentate în tabelul de mai jos:

| Funcție | Semnificație |
|---|---|
| **fopen()** | **deschiderea** unui fișier |
| **fclose()** | **închiderea** unui fișier |
| **fputc(),putc()** | **scrierea unui caracter** într-un fișier |
| **fputs()** | **scrierea unui șir de caractere** într-un fișier |
| **fprintf()** | **scrierea datelor formatate** într-un fișier |
| **fgetc(),getc()** | **citirea unui caracter** dintr-un fișier |
| **fgets()** | **citirea unui șir de caractere** dintr-un fișier |
| **fscanf()** | **citirea datelor formatate** dintr-un fișier |
| **fseek()** | **poziționarea cursorului** la un anumit octet într-un fișier |
| **feof()** | **determinarea sfârșitului de fișier**. Funcția returnează adevărat (≠0) dacă se determină sfârșit de fișier |
| **ferror()** | **determinarea unei erori** . Funcția returnează adevarat dacă a apărut o eroare |
| **rewind()** | **readucerea indicatorului de poziție la începutul fișierului** |
| **remove()** | **ștergerea** unui fișier |
| **fflush()** | **golirea streamului** asociat unui fișier |

Funcțiile uzuale utilizate pentru **deschiderea** și **închiderea fișierelor** sunt:

- deschiderea unui fișier **fopen()**
- închiderea unui fișier: **fclose()**

Funcțiile uzuale pentru **scrierea în fișiere** sunt:

- scrierea unui caracter în fișier: fputc
- scrierea unui sir de caractere în fișier: fputs
- scrierea datelor formatate în fișier: fprintf

Funcțiile utilizate pentru **citirea din fișiere** sunt:

- citirea unui caracter dintr-un fișier: fgetc
- citirea unui șir de caractere dintr-un fișier: fgets
- citirea datelor formatate dintr-un fișier: fscanf

Funcția fopen() realizează deschiderea fișierelor . Prototipul funcției este:

**FILE *fopen(const char *numefisier, const char *mod) ;**

- **numefisier** este numele fișierului ce va fi deschis în modul mod, și poate include opțional și o cale (director)
- **mod** este un șir de caractere care indică modul în care va fi deschis fișierul respectiv

| mod | Semnificație |
|---|---|
| **r** | deschiderea unui fișier text pentru **citire** |
| **w** | deschiderea/creearea unui fișier text pentru **scriere** |
| **a** | deschiderea unui fișier text pentru **adăugare** |
| **rb** | deschiderea unui **fișier binar** pentru **citire** |
| **wb** | deschiderea (creearea) unui **fișier binar** pentru **scriere** |
| **ab** | deschiderea unui **fișier binar** pentru **adăugare** |
| **r+** | deschiderea unui fișier text pentru **citire/scriere** |
| **w+** | deschiderea(creearea) unui fișier text pentru **citire/scriere** |
| **a+** | deschiderea (crearea) unui fișier text pentru **citire,scriere și adăugare** |
| **r+b** | deschiderea un **fișier binar** pentru **citire/scriere** |
| **w+b** | deschiderea (crearea) unui **fișier binar** pentru **citire/scriere** |
| **a+b** | deschiderea (crearea) unui **fișier binar** pentru **citire/scriere** |

Funcția **fopen()** returnează un pointer null în caz de eroare la deschiderea fișierului, și un pointer la fișierul deschis în caz de succes.

Funcția **fclose()** se utilizează pentru închiderea unui fișier (și a stream-ului asociat) deschis cu fopen() și are următorul prototip:

**int fclose(FILE *fp) ;**

unde **fp** este pointerul la fișierul deschis cu fopen()

Funcția returnează 0 în caz de succes, și EOF în caz de eroare și este inclusă în biblioteca <stdio.h>. Operația de închidere a fișierului trebuie efectuată după încheierea operațiilor de citire/scriere în fișier, în caz contrar pot rezulta pierderi de date din fișier și chiar distrugerea fișierului respectiv.

**Ex**.: deschiderea/închiderea unui fișier

```
FILE *fp;
…
fp=fopen(”test.txt”,”w”); //deschiderea fisierului
….                        //operatii de scriere in fisier
fclose(fp);               //inchiderea fisierului
```

Funcția **fputc()** este utilizată pentru scrierea unui singur caracter într-un fișier. Prototipul funcției este:

**int fputc(int ch, FILE *fp) ;**

unde : **fp** este pointerul returnat la deschiderea fișierului cu fopen() și specifică în ce fișier va fi scris caracterul **ch**

**ch** este caracterul scris în fișier, este de tip int dar se folosește numai octetul inferior

Funcția fputc este inclusă în biblioteca <stdio.h>.

**Ex.**: scrierea unui caracter în fișier

```
FILE *fp;
char ch=“A”;
…
```

```
fp=fopen(”test.txt”,”w”);       //deschiderea fisierului
ch=fputc(fp);                   //operatie de scriere in fisier
fclose(fp);                     //inchiderea fisierului
```

Funcția **fgetc()** este utilizată pentru citirea unui singur caracter dintr-un fișier. Prototipul funcției este:

**int fgetc(FILE *fp) ;**

unde **fp** este pointerul returnat la deschiderea fișierului cu fopen()

Funcția returnează un întreg (octetul superior) și EOF pentru eroare, fiind inclusă în biblioteca <stdio.h>.

Funcția **feof()** este utilizată pentru determinarea sfârșitului de fișier și are următorul prototip:

**int feof(FILE *fp) ;**

unde ***fp** este pointerul returnat la deschiderea fișierului cu fopen()

Funcția returnează 0 dacă s-a ajuns la sfârșitul fișierului sau o valoare pozitivă în caz contrar.

Funcția este inclusă în biblioteca <stdio.h>.

Funcția **fflush()** este utilizată pentru golirea streamului atașat unui fișier sau golirea tuturor fișierelor deschise pentru scriere și are următorul prototip:

**int fflush(FILE *fp);**

unde ***fp** pointer la fișier,

Funcția scrie conținutul datelor din stream(buffer) în fișierul asociat lui fp. Dacă fp este null atunci vor fi golite toate fișierele deschise pentru ieșiri.

Funcția returnează 0 în caz de succes, altfel returnează EOF, și este inclusă în biblioteca <stdio.h>.

Funcția **fputs()** se utilizează pentru scrierea unui șir de caractere într-un fișier și are următorul prototip:

**int fputs(const char *sir, FILE *fp) ;**

unde funcția scrie șirul de caractere pointat de ***sir** în fișierul specificat prin pointerul ***fp**
Funcția returnează o valoare pozitivă, sau EOF în caz de eroare și este inclusă în <stdio.h>.

**Ex.** :

```
...
char sir[80]=“orice text” ;
FILE *fp;
…
fputs(sir,fp);
…
```

Funcția **fgets()** este utilizată pentru citirea unui șir de caractere și are următorul prototip

**char *fgets(char *sir, int n, FILE *fp)**

unde fgets() citește un șir de lungime **n** caractere din fișierul specificat prin ***fp**, în șirul pointat de ***sir**.

Funcția returnează un pointer la șirul citit din fișier sau un pointer NULL în caz de eroare și este definită în biblioteca <stdio.h>.

Funcția **fprintf()** realizează scrierea în fișiere a datelor formatate și are următorul prototip:

**int fprintf(FILE *fp, const char *sir_control,…);**

unde ***fp** este un pointer de fișier, returnat de o apelare a funcției fopen() și operația de scriere se efectuează asupra acestui fișier.

Funcția este definită în <stdio.h>.

**Ex.** : scrierea într-un fișier a 2 variabile, tip șir și respectiv tip întreg

```
FILE *fp;
char s[80];
int t;
….
fprintf(fp,”Sirul este:%s, n=%d”, s,t ); //scrie in fisier
….
```

Funcția **fscanf()** este utilizată pentru citirea datelor formatate și are următorul prototip:

**int fscanf(FILE *fp, const char *sir_control,…);**

unde ***fp** este un pointer de fișier, returnat de o apelare a funcției fopen() și operația de citire se efectuează asupra acestui fișier.

Funcția este definită în <stdio.h>.

Funcția rewind() este utilizată pentru repoziționarea indicatorului de poziție la începutul fișierului și are următorul prototip:

**void rewind(FILE *fp);**

unde ***fp** pointer la fișierul în care se realizează repoziționarea indicatorului de poziție

Funcția este definită în <stdio.h>.

Funcția **ferror()** este utilizată pentru determinarea unei erori detectate la efectuarea unor operații asupra unui fișier și are următorul prototip:

**int ferror(FILE *fp);**

Funcția returnează o valoare pozitivă dacă a apărut o eroare în timpul ultimei operații asupra fișierului deschis prin pointerul *fp, și 0 în caz de reușita. Funcția este inclusă în biblioteca <stdio.h>

Funcția **remove()** este utilizată pentru ștergerea fișierelor din programul C/C++ și are următorul prototip:

**int remove(const char *numefisier);**

Funcția șterge fișierul specificat și returnează 0 în caz de succes al operației de ștergere, altfel o valoare diferită de 0.

Funcțiile specifice citirii/scrierii blocurilor de date din/în fisiere binare sunt: fwrite(), fread().

Funcția **fread()** are prototipul:

**size_t fread(void*buffer,size_t nrocteti,size_t numara,FILE *fp);**

unde ***buffer** este un pointer către o regiune de memorie în care se vor memora date primite de la fișier
**numara** corespunde numărului de elemente citite, cu dimeniunea egală cu **nrocteti**
**size_t** este definit în <stdio.h> și este aproximativ echivalent cu întreg fără semn

Funcția returnează nr. de elemente citite din fișier; această valoare poate fi mai mică decât **numara** dacă se ajunge la sfârșitul fișierului sau dacă apare o eroare.

Funcția **fwrite()** are prototipul:
**size_t fwrite(const void *buffer, size_t nrocteti, size_t numara, FILE *fp);**

unde ***buffer** este un pointer către datele care vor fi scrise în acel fișier
**numara** corespunde numărului de elemente scrise, având dimensiunea în octeți egală cu **nrocteti**

Funcția returnează nr. de elemente scrise în fișier; această valoare va fi egală cu **numara** dacă nu apare o eroare. Ambele funcții sunt definite în <stdio.h>.

Funcția **fseek()** realizează accesul aleator la un fișier și are prototipul:
**int fseek(FILE *fp, long numocteti, int origine);**

unde ***fp** este un pointer pentru fișier, returnat de o apelare a funcției fopen();
**numocteti** corespunde numărului de octeți de la origine care va deveni noua poziție curentă
**origine** este una din următoarele definiâii macro din <stdio.h>:
- Început fișier SEEK_SET
- Poziție curentă SEEK_CUR
- Sfârșit fișier SEEK_END

Funcția returnează 0 pentru operație încheiată cu succes, și valoare diferită de 0 pentru eroare. Funcția se utilizează pentru citire/scriere aleatoare și este definită în <stdio.h>.

Utilizarea intrării/ieșirii și erorii standard se realizează prin stream-urile standard: stdin,stdout,stderr. La lansarea în execuție a unui program C/C++ se deschid automat următoarele stream-uri:
- **stdin**: streamer de intrare (intrare standard) de tip text. Ex. tastatura
- **stdout**: streamer de ieșire (iesire standard) de tip text. Ex.monitorul
- **stderr**: streamer de ieșire erori (iesire standard erori) de tip text.

La închiderea programului C/C++ stream-urile sunt automat închise.

**Ex.**: scriere la ieșirea standard
```
putchar(char c)
{return putc(c, stdout);}
```

**Ex.** : citire de la intrarea standard
```
FILE *fp
char s[80]; int t;
fscanf(stdin, "%s%d",s,&t);
```

## PROBLEME REZOLVATE:

***Ex.1. Programul ilustrează crearea, deschiderea și scrierea într-un fișier numit "text.txt" utilizând funcțiile de nivel inferior: open, close, write. Să se verifice conținutul fișierului text.txt.***

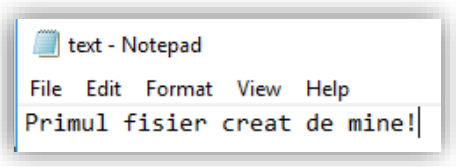
**Varianta in C**

```
#include <stdio.h>   //pentru printf
#include <string.h> //pentru strlen
#include <fcntl.h>   //pentru O_CREAT si O_RDWR
#include <io.h>      //pentru open, write si close
int main()
{int test;
char msg[]="Primul fisier creat de mine!";
if ((test=open("text.txt",O_CREAT|O_RDWR))==-1)
            {printf("eroare la deschidere fisier!\n");}
else {printf("Fisierul a fost creat. Verificati!\n");}
write(test,msg,strlen(msg)); close(test);
  return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <string.h> //pentru strlen
#include <fstream>
using namespace std;
int main()
{ofstream test("text.txt");
char msg[]="Primul fisier creat de mine!";
if (!test.is_open())
            {cout<<"eroare la deschidere fisier!"<<endl;}
else {cout<<"Fisierul a fost creat. Verificati!"<<endl;}
test<<msg;test.close();
return 0;}
```

***Rezultate:***




***Aplicație:***

*Să se modifice programul astfel încât să se adauge în fișier încă un șir de caractere introdus de la tastatură.*

***Ex.2. Programul realizează calculul factorialului: se citește n dintr-un fișier fis1.txt și tiparește n! într-un fișier fis2.txt. Atenție, fișierul de intrare în care se introduce valoarea lui n trebuie creat de către utilizator.După executarea programului să se verifice dacă fișierul de ieșire a fost creat și dacă conține rezultatul n! .***

Varianta in C

```
#include <stdio.h>
#include <stdlib.h>
int main (void)
{ FILE *fis1;
FILE *fis2;
char s1[12], s2[12];
int n,i;
double fact;
printf("Nume fisier intrare: \n");
scanf("%s",s1);
printf("Nume fisier iesire: \n");
scanf("%s",s2);
fis1=fopen(s1,"r");
if (fis1==NULL) {printf("ERROR la deschidere");exit(1);}
fscanf(fis1,"%d",&n);
fflush(stdin);
fis2=fopen(s2,"w");
if (fis2==NULL) {printf("ERROR");
exit(1);}
/* citeste n si calculeaza n!*/
fact=1.; i=2;
for (i=2;i<=n;i++) fact=fact*i;
printf("n=%d n!=%lf \n", n, fact);
fprintf(fis2,"n=%d n!=%lf \n", n, fact);
return 0;}
```

**Varianta in C++**

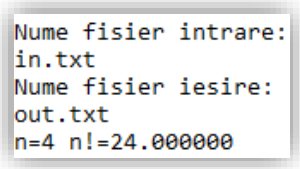
```
#include <iostream>
#include<fstream>
#include <stdlib.h>
using namespace std;
int main (void)
{ ifstream fis1;
 ofstream fis2;
char s1[12], s2[12];
int n,i;
double fact;
cout<<"Nume fisier intrare: "<<endl;cin.get(s1,12);
cin.ignore();
cout<<"Nume fisier iesire: "<<endl;
cin.get(s2,12);
fis1.open(s1);
if (fis1.fail()) {cerr<<"ERROR la deschidere"<<endl;exit(1);}
fis1>>n;
fis2.open(s2);
/* citeste n si calculeaza n!*/
fact=1.; i=2;
for (i=2;i<=n;i++) fact=fact*i;
printf("n=%d n!=%lf \n", n, fact);
fis2<<"n="<<n<<" n!="<<fact<<endl;
fis2.close();
return 0;}
```

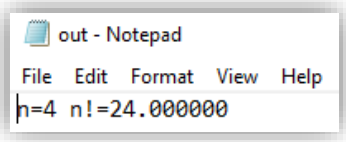
***Rezultate:***

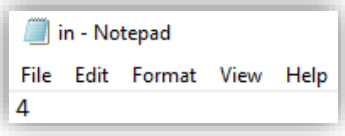
- *fișierul de intrare trebuie să se afle în rădăcina directorului care se creează pentru proiect (nu în bin, nu în bin/debug)*
- *dacă fișierul de intrare nu există execuția se termină cu eroare; fișierul de ieșire e creat automat la execuție*

***Aplicație:***

*Să se modifice programul astfel încât să se scrie în fișierul de ieșire rezultatul lui* $\sqrt{n^3}$.




***Ex.3.Programul realizează ștergerea unui fișier specificat în linia de comandă sub forma: >remove numefisier.extensie***

**Varianta in C**

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
int main(int argc,char *argv[])
{ char s[50];
  if (argc!=2){ printf("corect remove
fisier.extensie\n");exit(1);}
  printf("Sterg %s ? (Y/N):",argv[1]); gets(s);
  if(toupper(*s)=='Y')    {if(remove(argv[1])) {
          printf("fisierul nu se poate sterge\n"); exit(1);}
  else printf("Chiar l-am sters !");
  return 0; }}
```

**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
#include <stdlib.h>
#include <cstdio>//pentru remove
using namespace std;
int main(int argc,char *argv[])
{ char s[50];
  if (argc!=2){ cout<<"corect remove
fisier.extensie"<<endl;exit(1);}
  cout<<"Sterg "<<argv[1]<<"? (Y/N):"; cin.get(s,50);
  if(toupper(*s)=='Y')    { if(remove(argv[1])) {
          cout<<"fisierul nu se poate sterge"<<endl;
exit(1);}
  else cout<<"Chiar l-am sters !";} }
```

***Rezultate:***

Obs:

- fișierul executabil poate fi redenumit remove.exe
- execuția trebuie realizată dintr-o linie de comandă
- înainte de execuție trebuie creat un fișier text care va fi sters prin executia programului (editare in Notepad sau Codeblocks)

***Aplicație:***

*Să se verifice dacă fișierul a fost efectiv șters*

```
 Directory of D:\laura\codeblocks\ex83\bin\Debug

02/12/2019  02:18 PM    <DIR>          .
02/12/2019  02:18 PM    <DIR>          ..
02/12/2019  02:18 PM                19 in.txt
02/12/2019  02:16 PM            30,035 remove.exe
               2 File(s)         30,054 bytes
               2 Dir(s)  386,563,194,880 bytes free

D:\laura\codeblocks\ex83\bin\Debug>remove in.txt
Sterg in.txt ? (Y/N):y
Chiar l-am sters !
D:\laura\codeblocks\ex83\bin\Debug>dir
 Volume in drive D is Local Disk
 Volume Serial Number is 4E57-A59E

 Directory of D:\laura\codeblocks\ex83\bin\Debug

02/12/2019  02:19 PM    <DIR>          .
02/12/2019  02:19 PM    <DIR>          ..
02/12/2019  02:16 PM            30,035 remove.exe
               1 File(s)         30,035 bytes
               2 Dir(s)  386,563,194,880 bytes free
```

***Ex. 4. Programul realizează scrierea/citirea unor variabile tip int și double (nu caractere!) într-un /dintr-un fișier binar. Să se verifice dacă fișierului binar "test" a fost creat și dacă conține variabilele tipărite.***

Varianta in C

```
#include <stdio.h>
#include <stdlib.h>



int main(void)
{FILE *fp;
double d=10.5; int i=100;
if((fp=fopen ("test","wb+"))==NULL) {
   printf("nu se poate deschide fisierul\n"); exit(1); }
fwrite(&d, sizeof(double),1,fp);
fwrite(&i, sizeof(int),1,fp); rewind(fp);
fread(&d, sizeof(double),1,fp);
fread(&i, sizeof(int),1,fp);
printf("%lf %d ",d,i);
fclose(fp);
return 0;}
```

Varianta in C++

```
#include <iostream>
#include <cstdio>
#include<fstream>
using namespace std;
int main(void)
{FILE *fp;
double d=10.5; int i=100;
if((fp=fopen ("test","wb+"))==NULL) {
   cout<<"nu se poate deschide fisierul\n"; }
fwrite(&d, sizeof(double),1,fp);
fwrite(&i, sizeof(int),1,fp); rewind(fp);
fread(&d, sizeof(double),1,fp);
fread(&i, sizeof(int),1,fp);
cout<<d<<"  "<<i;
fclose(fp);
return 0;}
```

***Rezultate:***

```
10.500000 100
```

***Aplicație:***

*Să se modifice programul astfel încât să se citească de la tastatură două valori pentru variabilele d și i și apoi să se scrie pătratul lor în fișierul test*

***Ex.5: Programul citește pe rând câte o operație din fișierul de intrare a.txt, sub forma op1 op op2 și se tipărește rezultatul operației în fișierul ab.txt***

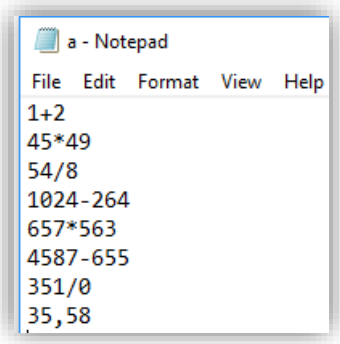
Varianta in C

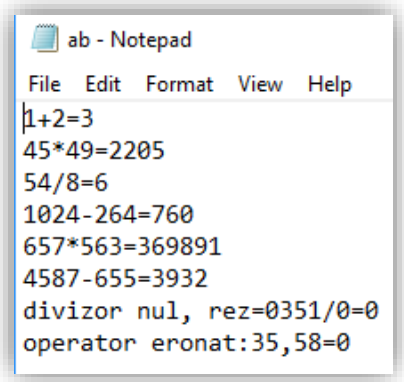
```
#include <stdio.h>
#include <stdlib.h>
int main (void)
{FILE *fis1, *fis2;
int op1,op2,rez; char op;
fis1=fopen("a.txt","r");
fis2=fopen("ab.txt","wa");
while (fscanf(fis1,"%d%c%d\n",&op1,&op,&op2)!=EOF)
{ switch (op) {
case '+': rez=op1+op2; break;
case '-': rez=op1-op2; break;
case '*': rez=op1*op2; break;
case '/': if (op2==0) { rez=0;fprintf(fis2,"divizor nul,
rez=%d", rez); }
else rez=op1/op2;  break;
default: rez=0; fprintf(fis2,"operator eronat:");}
fprintf(fis2,"%d%c%d=%d\n",op1,op,op2,rez);}
printf ("Succes");
return 0;}
```

Varianta in C++

```
#include <iostream>
#include <stdlib.h>
#include<fstream>
using namespace std;
int main (void)
{ifstream fis1;
ofstream fis2;
int op1,op2,rez; char op;
fis1.open("a.txt");
fis2.open("ab.txt");
if (fis1.fail()) {cerr<<"EROARE la deschidere"<<endl;exit(1);}
while (!fis1.eof())
{ fis1>>op1>>op>>op2;
   switch (op) {
case '+': rez=op1+op2; break;
case '-': rez=op1-op2; break;
case '*': rez=op1*op2; break;
case '/': if (op2==0) { rez=0;fis2<<"divizor nul, rez= "<<rez; }
else rez=op1/op2;  break;
default: rez=0; fis2<<"operator eronat:";}
fis2<<op1<<op<<op2<<"="<<rez<<endl;}
cout<<"Succes";
return 0;}
```

***Rezultate:***




***Aplicație:***

*Să se modifice programul astfel încât să se adauge și alte operații matematice (pow, sqrt,sin, log, etc ) utilizând funcții din <math.h>*

***Ex.6: Programul copiază un fișier text specificat în alt fișier specificat de către utilizator. Trebuie să creați fișierul de intrare și să salvați un text oarecare în acest fișier, iar după executarea programului să verificați dacă fișierul copiat a fost creat și dacă conține același text ca și fișierul de intrare.***

**Varianta in C**

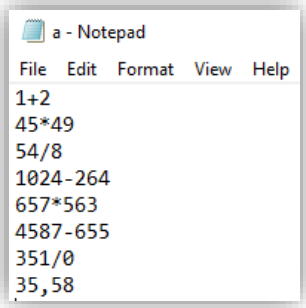
```
#include <stdio.h>
#include <stdlib.h>
int main()
{char in_name[25], out_name[25];
FILE *in_file, *out_file, *fopen (); int c;
printf("Numele fisierului ce trebuie copiat:\n");
scanf("%24s", in_name);
printf("Numele fisierului in care se copiaza:\n");
scanf("%24s", out_name);
in_file = fopen ( in_name, "r");
if( in_file == NULL )
    printf("Nu pot deschide %s pentru citire.\n", in_name);
     else { out_file = fopen (out_name, "w");
            if( out_file == NULL )
printf("Nu pot deschide %s pentru scriere.\n", out_name);
            else { while( (c = getc( in_file)) != EOF )
            putc (c, out_file);  putc (c, out_file);
           /* copy EOF */
 printf("Fisierul a fost copiat.\n");
 fclose (out_file); } fclose (in_file);   }
 return 0;}
```

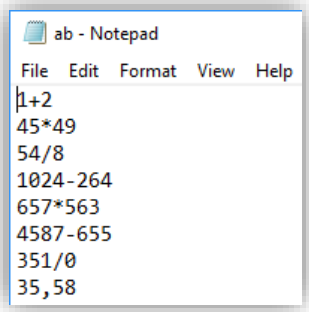
**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
#include<fstream>
using namespace std;
int main()
{char in_name[25], out_name[25];
ifstream in_file;
ofstream out_file; char c;
cout<<"Numele fisierului ce trebuie copiat:"<<endl;
cin.get(in_name,25);
cin.ignore();
cout<<"Numele fisierului in care se copiaza:"<<endl;
cin.get(out_name,25);
cin.ignore();
in_file.open(in_name);
if( in_file.fail())
    cerr<<"Nu pot deschide "<<in_name<<" pentru
citire."<<endl;
     else { out_file.open (out_name);
            if(out_file.fail())
cerr<<"Nu pot deschide "<<out_name<<" pentru
scriere."<<endl;
            else { while (in_file.get(c))   out_file<<c;
cout<<"Fisierul a fost copiat.";
in_file.close(); } out_file.close();      }
return 0;}
```

***Rezultate:***

```
Numele fisierului ce trebuie copiat:
a.txt
Numele fisierului in care se copiaza:
ab.txt
Fisierul a fost copiat.
```




***Aplicație:***

*Să se modifice programul astfel încât să se adauge la sfârșitul fișierului copiat textul "STOP".*

***Ex.7 Fișierul int.txt se inițializează prin program cu numere de la 0 la 1000 iar dublul acestor numere se tipărește în fișierul out.txt.***

**Varianta in C**

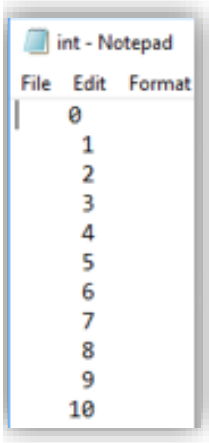
```
#include <stdio.h>
#include <stdlib.h>
#define max 2000


int main(void)
{FILE *fis1, *fis2;
int i,j=0,a[max], *p;
printf ("\nInitializarea sirului cu nr. de la 0 la 1000 folosind
un pointer si un index\n");
fis1=fopen("int.txt","w"); fis2=fopen("out.txt","w");
for (p=a,i=0;i<=1000;j++,i++)  p[i]=j;
for (p=a,i=0;i<=1000;i++)
{printf("%4d", p[i]);
fprintf(fis1,"%5d \n ", p[i]);
fprintf(fis2,"%5d \n", p[i]*2);}
fclose (fis1);fclose (fis2);
 return 0;}
```

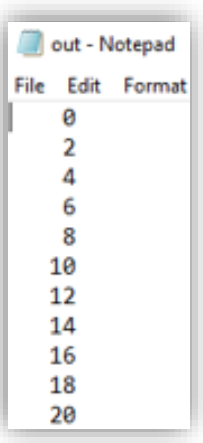
**Varianta in C++**

```
#include <iostream>
#include <stdlib.h>
#define max 2000
#include <fstream>
using namespace std;
int main(void)
{ofstream fis1, fis2;
int i,j=0,a[max], *p;
cout<<endl<<"Initializarea sirului cu nr. de la 0 la 1000
folosind un pointer si un index";
fis1.open("int.txt"); fis2.open("out.txt");
for (p=a,i=0;i<=1000;j++,i++)  p[i]=j;
for (p=a,i=0;i<=1000;i++)
{cout<< p[i]<<"  ";
fis1<<p[i]<<endl;
fis2<<p[i]*2<<endl;}
fis1.close();fis2.close();
 return 0;}
```

***Rezultate:***

secvență din pagina de rezultate:




***Aplicatie:***

*Modificați programul astfel încât fișierul de ieșire să conțină valoarea polinomului $2x^3+7x^2-x+5$, unde x sunt valorile din fișierul int.txt*

***Ex.8 Programul C/C++ :***

1. ***citeste trei valori de la tastatura reprezentand numarul de ore in care PC-ul, TV-ul si masina de spalat au fost conectate la reteaua de alimentare cu energie electrica***
2. ***calculeaza si afiseaza pe ecran consumul pe ziua respectiva pentru cele 3 echipamente, tinand cont de nr de ore cat sunt pornite dispozitivele , stiind ca un  PC consuma 250 W/h, un TV 33 W/h si o masina de spalat 650 W/h .***
3. ***scrie intr-un fisier out.txt consumul pe ziua respectiva pentru cele 3 echipamente.***

**Varianta in C**

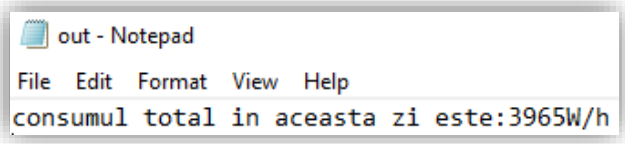
```
#include <stdio.h>
#include <stdlib.h>

int main()
{ float orePC, oreTV, oremasina,total=0;
FILE* fis1;
fis1=fopen("out.txt","w");
printf("orePC="); scanf("%f", &orePC);
printf("oreTV="); scanf("%f", &oreTV);
printf("oremasina="); scanf("%f", &oremasina);
if (orePC < 0 || orePC>24 || oreTV < 0 || oreTV>24 ||
oremasina < 0 || oremasina>24) printf("valorile citite
nu sunt corecte");
  else {total=orePC*250+oreTV*33+oremasina*650;}
printf("consumul total pe  zi este:%.0f W/h", total);
fprintf(fis1,"consumul total pe zi este:%f W/h", total);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{ float orePC, oreTV, oremasina,total=0;
ofstream fis1;
cout << "orePC="; cin>>orePC;
cout << "\noreTV="; cin>>oreTV;
cout << "\noremasina="; cin>>oremasina;
if (orePC < 0 || orePC>24 || oreTV < 0 || oreTV>24 ||
oremasina < 0 || oremasina>24) cout <<"valorile citite nu
sunt corecte\n";
   else { total = orePC * 250 + oreTV * 33 + oremasina * 650;}
cout<<"consumul total pe zi este:"<<total<<"W/h"<< endl;
fis1.open("out.txt");
fis1<<"consumul total pe zi este:"<<total<<"W/h"<< endl;
fis1.close();return 0;}
```

***Rezultate:***

```
orePC=10

oreTV=5

oremasina=2
consumul total in aceasta zi este:3965W/h
```




***Aplicație:***

*Să se modifice programul astfel încât să se adauge si alte dispozitive electrocasnice pentru care se calculeaza consumul pe zi.*

***Ex.9. Program C/C++:***

***a) citește de la tastatura urmatoarele valori pentru 2 materiale conductoare ,aluminiu si cupru: rezistivitate (ρ), permeabilitatea relativa ($\mu_r$), permeabilitatea vidului($\mu_0$) si frecventa (f)***

***b) calculeaza adancimea lor de patrundere($\delta$) cu formula de mai jos pentru ambele materiale si afiseaza rezultatul pe ecran***:

$$\delta = \sqrt{\frac{\rho}{\pi f_0 \mu_r \mu_0}}$$

***c) afiseaza intr-un fisier de iesire aceste rezultate.***

**Varianta in C**

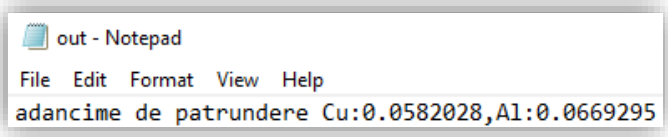
```
#include <stdio.h>
#include <math.h>
int main()
{float rezal,rezcu,miurcu,miural,permrelo,fcu;
float fal,adpcu,adpal;
FILE* fis1;
fis1=fopen("out.txt","w");
printf("rezal="); scanf("%f", &rezal);
printf("rezcu="); scanf("%f", &rezcu);
printf("miurcu="); scanf("%f", &miurcu);
printf("miural="); scanf("%f", &miural);
printf("fcu="); scanf("%f", &fcu);
printf("fal="); scanf("%f", &fal);
printf("permrelo="); scanf("%f", &permrelo);
adpcu=sqrt(rezcu/(3.14*fcu*miurcu*permrelo));
adpal=sqrt(rezal/(3.14*fal*miural*permrelo));
fprintf(fis1, "adancime de patrundere Cu:%f, Al:%f",
adpcu,adpal);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;
int main()
{ float rezal,rezcu,miurcu,miural,permrelo,fcu;
float fal,adpcu, adpal;
ofstream fis;
cout<<"rezal="; cin>>rezal; cout<<"rezcu="; cin>>rezcu;
cout<<"miurcu="; cin>>miurcu;
cout<<"miural="; cin>>miural;
cout<<"fcu="; cin>>fcu; cout<<"fal="; cin>>fal;
cout<<"permrelo="; cin>>permrelo;
adpcu=sqrt(rezcu/(3.14*fcu*miurcu*permrelo));
adpal=sqrt(rezal/(3.14*fal*miural*permrelo));
fis.open("out.txt");
cout<<"adancime de patrundere
Cu:"<<adpcu<<",Al:"<<adpal;
fis<<"adancime de patrundere Cu:"<<adpcu<<",Al:"<<adpal;
return 0;}
```

***Rezultate:***

```
rezal=2.65
rezcu=1.67
miurcu=1.25
miural=1.25
fcu=100
fal=120
permrelo=1.256
adancime de patrundere Cu:0.0582028,Al:0.0669295
```




***Aplicație:***

*Să se modifice programul astfel încât să se citeasca dintr-un fisier de intrare datele cunoscute in problema.*

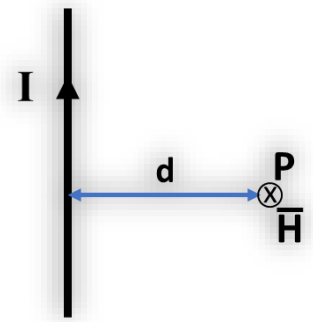
***Ex.10 Programul C/C++:***

***a) citește de la tastatură valoarea reală a curentului I ce parcurge un conductor rectiliniu, infinit lung și distanța d dintre un punct P și acest conductor.***

***b) calculeaza si afiseaza pe ecran intensitatea câmpului magnetic H în punctul P,produs de curentul I din conductor, după formula:***

$$H = \frac{I}{2 * \pi * d}$$

***c) scrie într-un fișier abc.txt rezultatele.***



**Varianta in C**

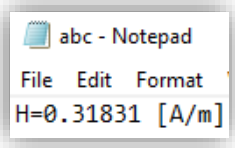
```
#include <stdio.h>
#include <stdlib.h>
#define pi 3.14159
int main()
{float I, d,H;
 FILE *f1;
 f1=fopen("abc.txt","w");
 //Citirea datelor introduse de utilizator
printf("Introduceti intensitatea curentului I=");
scanf("%f",&I);
printf("Introduceti distanta d=");scanf("%f",&d);
//Determinarea Intensitatii cp. magnetic H
H=I/(2*pi*d);
printf("Intensitatea campului magnetic H=%f [A/m]", H);
//scrierea rezultatului in fisier
fprintf(f1,"H=%f [A/m]", H);
fclose(f1);
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <fstream>
#define pi 3.14159
using namespace std;
int main()
{ float I, d,H;
 ofstream f1;
 f1.open("abc.txt");
 //Citirea datelor introduse de utilizator
 cout<<"Introduceti intensitatea curentului I= "; cin>>I;
 cout<<"Introduceti distanta d="; cin>>d;
 //Determinarea Intensitatii cp. magnetic H
 H=I/(2*pi*d);
cout<<"Intensitatea campului magnetic H="<<H<<" [A/m]"<<
endl;
//scrierea rezultatului in fisier
f1<<"H="<<H<<" [A/m]";
f1.close(); return 0;}
```

***Rezultate:***

```
Introduceti intensitatea curentului I= 20
Introduceti distanta d=10
Intensitatea campului magnetic H=0.31831 [A/m]
```




***Aplicație:***

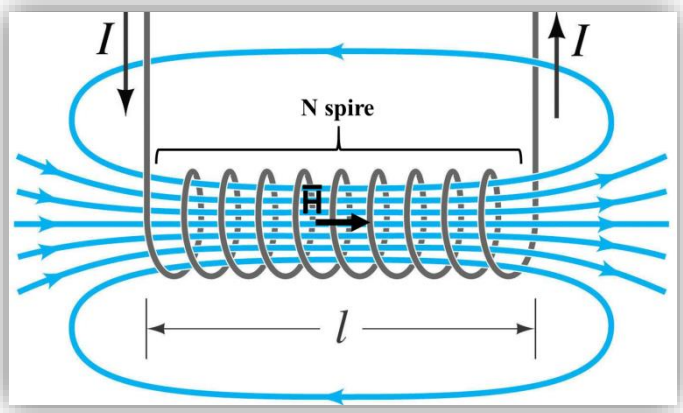
*Să se modifice programul astfel încât să se citească dintr-un fisier de intrare datele cunoscute in problema.*

***Ex. 11.Programul in C/C++:***

***a) citește de la tastatură valoarea întreagă a numărului de spire (N), valoarea reală a curentului I ce parcurge o bobină cilindrică L și lungimea acesteia l.***

***b) calculeaza si afiseaza intensitatea câmpului magnetic H în interiorul bobinei L, pe axa acesteia, după formula : $H = \frac{N*I}{l}$***

***c) scrie valorile calculate la punctul b) intr-un fisier rez.txt.***



**Varianta in C**

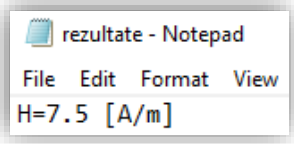
```
#include <stdio.h>
#include <stdlib.h>
int main()
{   float I,l,H;
 int N; FILE *f1;
 f1=fopen("rezultate.txt","w");
 //Citirea datelor introduse de utilizator
 printf("Introduceti intensitatea curentului I= ");
 scanf("%f",&I);
 printf("\nIntroduceti lungimea bobinei l=");
scanf("%f",&l);
printf("\nIntroduceti numarul de spire a bobinei N=");
scanf("%d",&N);
//Determinarea Intensitatii cp. magnetic H
H=N*I/l;
printf("Intensitatea campului magnetic H=%f [A/m]", H);
//scrierea rezultatului in fisier
fprintf(f1,"H=%f [A/m]", H);
fclose(f1); return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{   float I,l,H;int N;
ofstream f1;
f1.open("rezultate.txt");
//Citirea datelor introduse de utilizator
cout<<"Introduceti intensitatea curentului I= ";cin>>I;
cout<<"\nIntroduceti lungimea bobinei l=";
cin>>l;
cout<<"\nIntroduceti numarul de spire a bobinei N=";
cin>>N;
//Determinarea Intensitatii cp. magnetic H
H=N*I/l;
cout<<"Intensitatea campului magnetic H="<<H<<"
[A/m]"<<endl;
//scrierea rezultatului in fisier
f1<<"H="<<H<<" [A/m]";
f1.close(); return 0;}
```

***Rezultate:***

```
Introduceti intensitatea curentului I= 10

Introduceti lungimea bobinei l=20

Introduceti numarul de spire a bobinei N=15
Intensitatea campului magnetic H=7.5 [A/m]
```




***Aplicație:***

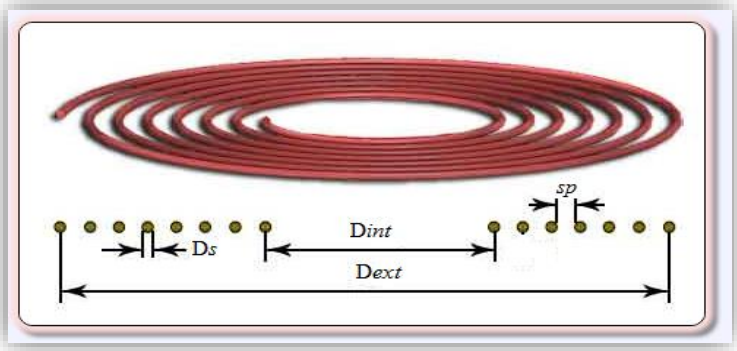
*Să se modifice programul astfel încât să se citească dintr-un fisier de intrare datele cunoscute in problema.*

***Ex. 12. Programul C/C++:***

***a) citește de la tastatura o valoarea întreagă a numărului de spire (N), si valori reale pentru: diametrul interior ($D_{int}$), diametrul spirei ($D_S$) si distanta dintre spire (sp) pentru spira din imagine***

***b) calculeaza si afișeaza pe ecran inductivitatea spirei pentru valorile de intrare ,calculate cu formula*** $L = \frac{N^2+A^2}{30 \cdot A - 11 \cdot D_{int}}$, ***unde*** $A = \frac{D_{int}+N \cdot (D_s+sp)}{2}$

***c) tipareste intr-un fisier de iesire N, Dint, DS, sp, si inductivitatea spirei L.***



**Varianta in C**

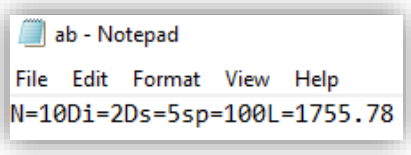
```
#include <stdio.h>
#include <stdlib.h>
#define pi 3.141593
int main()
{FILE *fis1;
int N; double A,Ds, Di,sp;
printf(" N="); scanf("%d",&N);
printf(" Ds="); scanf("%lf",&Ds);
printf(" Di="); scanf("%lf",&Di);
printf(" sp="); scanf("%lf",&sp);
A=(Di+N*(Ds+sp))/2;
printf("L=%lf",(N*N*A*A)/(30*A-11*Di));
fis1=fopen("ab.txt","w");
printf("N=%d Di=%lf Ds=%lf sp=%lf L=%lf",
N,Di,Ds,sp,(N*N*A*A)/(30*A-11*Di));
fprintf(fis1,"N=%d Di=%lf Ds=%lf sp=%lf L=%lf",
N,Di,Ds,sp,(N*N*A*A)/(30*A-11*Di));
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <string>
#include <fstream>
const double pi = 3.141593;
using namespace std;
int main()
{ ofstream fis1;
int N;double A,Ds, Di,sp;
cout <<" N="; cin>>N;  cout <<" Ds="; cin>>Ds;
cout <<" Di=";  cin>>Di; cout <<" sp=";  cin>>sp;
A=(Di+N*(Ds+sp))/2;
cout<<" L="<<(N*N*A*A)/(30*A-11*Di);
fis1.open("ab.txt");
fis1<<"N="<<N<<"Di="<<Di<<"Ds="<<Ds<<"sp="<<sp<<"L="<
<(N*N*A*A)/(30*A-11*Di);
fis1.close();
return 0;}
```

***Rezultate:***

```
N=10
Ds=5
Di=2
sp=100
L=1755.78
```




***Aplicație:***

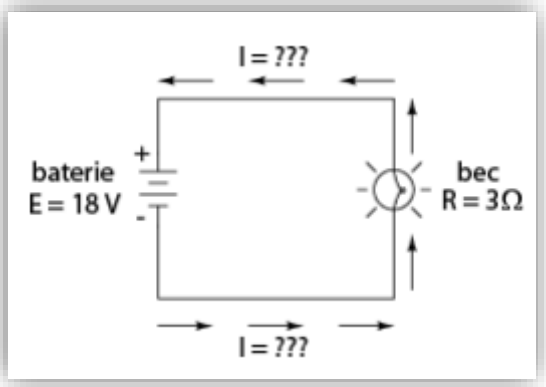
*Să se modifice programul astfel încât să se citeasca dintr-un fisier de intrare datele cunoscute in problema.*

***Ex. 13. Programul C/C++ :***

***a) citește de la tastatură un sir de valori reale ale lui E (18 V, 38 V, 56V) si o singura valoare pentru rezistenta R=3 Ohmi din montajul de mai jos***

***b) calculeaza si afiseaza valorile curentului I (Amperi) si a puterii becului (Wati) corespunzatoare valorilor citite: I=E/R , P=I E***

***c) scrie intr-un fisier de iesire a.txt valorile rezultate***



**Varianta in C**

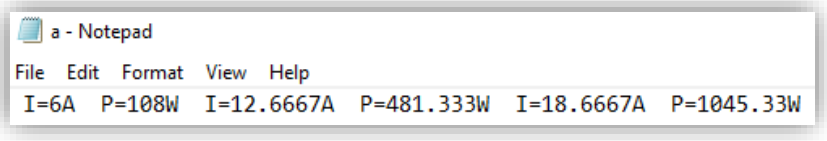
```
#include <stdio.h>
#include <stdlib.h>
#define pi 3.141593
int main()
{FILE *fis1;
float E[100], R=3; int i, n;
printf("n=");scanf("%d",&n);
fis1=fopen("a.txt", "w");
for(i=0; i<n; i++)
{ printf ("\nE[%d]=", i); scanf("%f",&E[i]);
printf("I=%.0f A",E[i]/R);
printf(" P=%.0f W", (E[i]/R)*E[i]);
fprintf(fis1," I=%.0f A",E[i]/R);
fprintf(fis1," P=%.0f W",(E[i]/R)*E[i]);   }
fclose(fis1);return 0;
return 0;}
```

**Varianta in C++**

```
#include <iostream>
#include <string>
#include <fstream>
const double pi = 3.141593;
using namespace std;
int main()
{ofstream fis1;
float E[100], R=3; int i, n;
cout<<"n=";   cin>>n;
fis1.open("a.txt");
for(i=0; i<n; i++)
{ cout << "\nE["<<i<<"]="; cin>>E[i];
cout <<" I="<<E[i]/R<< "A";
cout<<" P="<<  (E[i]/R)*E[i]<<"W";
fis1<<" I="<<E[i]/R<<"A ";
fis1<<" P="<<(E[i]/R)*E[i]<<"W ";   }
fis1.close();return 0;}
```

***Rezultate:***

```
n=3

E[0]=18
 I=6A P=108W
E[1]=38
 I=12.6667A P=481.333W
E[2]=56
 I=18.6667A P=1045.33W
```




***Aplicație:***

*Să se modifice programul astfel încât să se citeasca dintr-un fisier de intrare datele cunoscute in problema.*

***Ex.14 Programul realizează calculul mediilor studenților ale căror date sunt introduse în fișierul int.txt sub forma (tab între campuri):***

| | | | | | |
|---|---|---|---|---|---|
| Popa Mircea | 1111 | 10. | 9.5 | 9.5 | 8. |
| Munteanu Ioan | 1323 | 6.5 | 7.5 | 8.5 | 6.5 |
| Adam Mihai | 1121 | 7.5 | 6. | 7. | 8. |
| Cristea Mihaela | 1122 | 8.5 | 9. | 10. | 7.5 |
| Alisie Mirela | 2121 | 6.5 | 8.5 | 7.5 | 9. |
| Laza Sergiu | 2222 | 10. | 10. | 10. | 10. |
| Vaida Maria | 2525 | 9.5 | 7. | 6. | 8. |

***Fișierul care conține articolele de tip student cu mediile calculate este med.txt. Programul calculează mediile studenților în fișierul med.txt și apoi ordonează descrescător în ordinea mediilor studenții în fișierul sort.txt .***

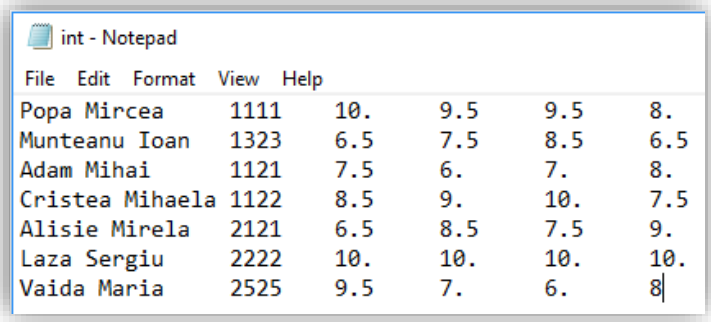
**Varianta in C**

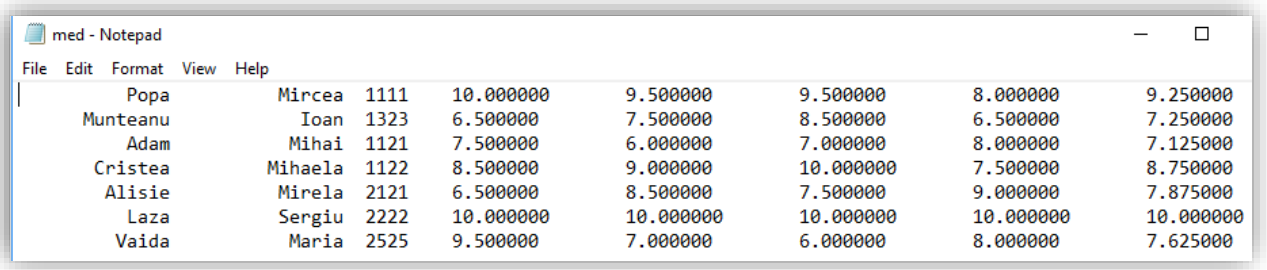
```
#include <stdio.h>
#include <stdlib.h>
struct student{
                        char name[20];
                        char prenume[20];
                        int id;       //cod grupa
                        double nota1, nota2, nota3, nota4, media;} s[100], aux[100];
FILE *fis1;                         //fisier I cu notele studentilor int.txt
FILE *fis2;                         //fisier O cu mediile studentilor med.txt
FILE *fis3;                         //fisier O cu studentii ordonati dupa medii sort.txt
int i=0,n=0, k=0;
char ch;
void sortmed(struct student s[100]);
int main()
{fis1=fopen("int.txt","r");
if (fis1==NULL) {printf("ERROR");  exit(1);}
while (fscanf(fis1, "\n%14s\t%14s\t%d\t%lf\t%lf\t%lf\t%lf", s[i].name, s[i].prenume,&s[i].id,
&s[i].nota1,&s[i].nota2,&s[i].nota3,&s[i].nota4)!=EOF)
{i++;}n=i;
printf("n=%d\n", n);
fis2=fopen("med.txt", "w+");
for (i=0;i<n;i++)
    {s[i].media = (s[i].nota1+s[i].nota2+s[i].nota3+s[i].nota4)/4;
    fprintf(fis2, "%14s\t%14s\t%d\t%lf\t%lf\t%lf\t%lf\t%lf\n", s[i].name, s[i].prenume,s[i].id,
s[i].nota1,s[i].nota2,s[i].nota3,s[i].nota4,s[i].media);}
printf("Fisierul sort.txt este ordonat descrecator in ordinea mediilor");
for (i=0;i<n;i++)  {sortmed(s);}
fis3=fopen("sort.txt","w+");
fprintf(fis3,"\nLista studentilor in ordinea mediilor\n");
for (i=0; i<n; i++)
{fprintf(fis3,"\n%d.\nNume : %-14s\nPrenume: %-14s\ncod grupa: %d\nmedia:%lf\n",(i+1),s[i].name,s[i].prenume,
s[i].id,s[i].media);
fprintf(fis3, "**********************************\n"); }
return 0;}
void sortmed(struct student s[100])
{int j;
for (j=0;j<=n;j++)
            {if (s[j].media<s[j+1].media)
                { aux[j]=s[j] ;  s[j]=s[j+1];  s[j+1]=aux[j];k++;printf("k=%d\n",k);}  }
}
```

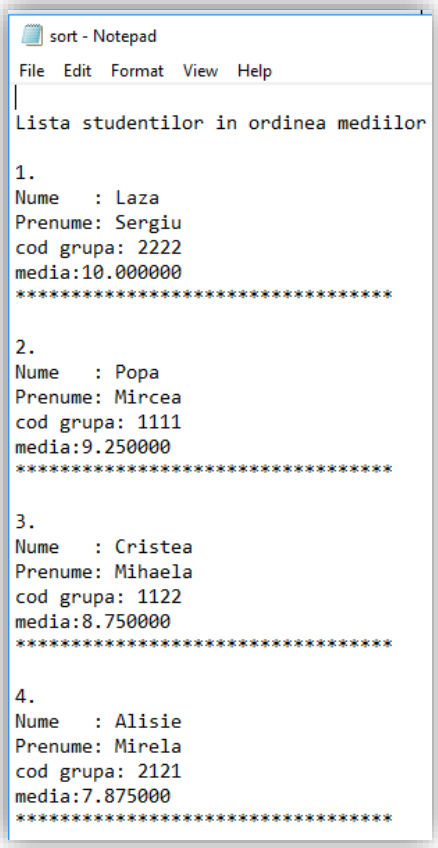
**Varianta C++**

```
#include <iostream>
#include <stdlib.h>
#include<fstream>
using namespace std;
struct student{
                    char name[20];
                    char prenume[20];
                    int id;      //cod grupa
                    double nota1, nota2, nota3, nota4, media;} s[100], aux[100];
ifstream fis1;                  //fisier I cu notele studentilor int.txt
ofstream fis2;                  //fisier O cu mediile studentilor med.txt
ofstream fis3;                  //fisier O cu studentii ordonati dupa medii sort.txt
int i=0,n=0, k=0;
char ch;
void sortmed(struct student s[100]);
int main()
{fis1.open("int.txt");
if( fis1.fail())  {cerr<<"eroare"<<endl; exit(1);}
while (!fis1.eof())
{   fis1>>s[i].name>>s[i].prenume>>s[i].id>>s[i].nota1>>s[i].nota2>>s[i].nota3>>s[i].nota4; i++;}
n=i; cout<<"n="<<n<<endl;
fis2.open("med.txt");
for (i=0;i<n;i++)
{s[i].media = (s[i].nota1+s[i].nota2+s[i].nota3+s[i].nota4)/4;
fis2<<s[i].name<<"\t"<<s[i].prenume<<"\t"<<s[i].id<<"\t"<<s[i].nota1<<"\t"<<s[i].nota2<<"\t"<<s[i].nota3<<"\t"<<
s[i].nota4<<"\t"<<s[i].media<<endl;}
cout<<"Fisierul sort.txt este ordonat descrecator in ordinea mediilor";
for (i=0;i<n;i++)  {sortmed(s);}
fis3.open("sort.txt");
fis3<<"\nLista studentilor in ordinea mediilor"<<endl;
for (i=0; i<n; i++)
{fis3<<"\n"<<i+1<<"\nNume : "<<s[i].name<<"\nPrenume: "<<s[i].prenume<<"\ncod grupa: "<<s[i].id<<"\nmedia:
"<<s[i].media<<endl;
 fis3<<"**********************************"<<endl; }
return 0;}
void sortmed(struct student s[100])
{int j;
for (j=0;j<=n;j++)
            {if (s[j].media<s[j+1].media)
                { aux[j]=s[j] ;s [j]=s[j+1]; [j+1]=aux[j];k++;cout<<"k="<<k<<endl;}  }
}
```

***Rezultate:***






***Aplicație:***

*Modificați programul astfel încât să sortați fișierul în ordine alfabetică.*

## PROBLEME PROPUSE

1. ***Să se scrie un program care permite introducerea liniilor de text de la tastatură într-un fișier specificat în linia de comandă. Dacă programul se numește scrie.c atunci în linia de comandă se introduce de ex: >scrie text.txt***
   ***și un text oarecare scris pe mai multe linii fiecare linie incheiată cu Enter și terminat prin caracterul "." Fișierul text.txt va conține liniile de text introduse de la tastatură după linia de comandă.***

2. ***Să se scrie programul care:***
   - ***citește datele angajaților dintr-o firmă de la tastatură sub forma: nume, prenume, adresa,cod numeric,salar brut***
   - ***creează un fișier în care se scriu aceste date***
   - ***calculează salarul net și impozitul (după formula impozit =0.4*salarbrut) pentru fiecare angajat***
   - ***crează un fișier în care sunt tipăriți într-un tabel tip stat de plată toți angajații cu sumele corespunzătoare salarului brut, salarului net și impozitului.***

# Capitolul 10
# Stream-urile cin și cout

În acest capitol sunt prezentate consideraţii teoretice și probleme rezolvate privind definirea și utilizarea stream-urilor cin și cout in C++.

## CONSIDERAŢII TEORETICE

Stream-urile cin și cout din <iostream.h> sunt utilizate in C++ în locul funcțiilor de citire scriere utilizate în C. Semnificația lor este :

- cin (“see in”) = input stream
- cout (“see out”)= output stream

cin și cout se utilizează cu operatorii << și >> care au următoarea semnificație:

- << (“put to”) = operator de inserție
- >> (“get from”) = operator de extracție

Formatul de utilizare al acestor streamuri este :

**cin >> var1 >> var2 …;**
**cout << var1 << var2 ...;**

unde **var1, var2**, … pot fi numai nume de variabile în cazul lui **cin** nu și constante sau expresii ca în cazul lui **cout.**

Efectul operației de inserție este descris pentru tipurile standard char, int și float în tabelul de mai jos:

| **Tipul variabilei din operația de inserție cin** | **Data considerată validă** |
|---|---|
| Char | Un singur caracter diferit de blank (spațiu) |
| Int | O constantă întreagă opțional precedată de un semn |
| Float | O constantă întreagă sau reală (eventual specificată în format științific- utilizând notația cu e), opțional precedată de semn |

**Ex.** : citirea unui întreg

```
int i;
cin >> i;
```

Efect: similar cu efectul instrucțiunilor:

```
int i;
scanf(“%d”, &i);
```

Exemple de utilizare a streamului cin:

| **Instrucțiune de inserție** | **Data intrare** | **Conținut variabilă după inserție** |
|---|---|---|
| 1. cin >> i; | 32 | i=32 |
| 2. cin >> i >> j; | 4 60 | i=4, j=60 |
| 3. cin >> i >> ch >> x; | 25 A 16.9 | i=25, ch=‘A’, x=16.9 |

| | | |
|---|---|---|
| 4. cin >> i >> ch >> x; | 25<br>A<br>16.9 | i=25, ch='A', x=16.9 |
| 5. cin >> i >> ch >> x; | 25A16.9 | i=25, ch='A', x=16.9 |
| 6. cin >> i >> j >> x; | 12 8 | i=12, j=8 –calculatorul așteaptă introducerea lui x |
| 7. cin >> i >> x; | 46 32.4 15 | i=46,x=32.4 –calculatorul reține ultimul nr. (15) pentru o inserție ulterioară; dacă aceasta nu se realizează aceste date se pierd. |

Caracterul linie nouă : "\n"( new line ) și endl se utilizează pentru trecerea pe linie nouă. În exemplele de mai jos se ilustrează modul de utilizare a streamului cout.

**Ex.:** operații de scriere (afisare) echivalente

```
cout << "Hello \n";
cout << "Hello" << endl;
```

**Ex. :** scrierea (afișarea) unui text

```
cout << "Hello student !\n";
```

**Efect:** similar cu efectul instrucțiunii:

```
printf("Hello student !\n");
```

Funcția **cin.get()** se utilizează pentru citirea datelor de tip caracter și are următorul format de apelare: **cin.get(caracter);** Se preia caracterul introdus de la tastatură indiferent dacă acesta este spațiu sau linie nouă ('\n')

Diferența dintre cin și cin.get(): cin ignoră spațiile cin.get() preia și spațiile sau caracterul de linie nouă introduse de la tastatură.

**Ex. :**

```
cin >>ch1 >>ch2;
```

Dacă se introduce de la tastatură:

```
>R 1
```

atunci lui ch1 i se atribuie caracterul "R" se sare peste spațiu și se atribuie lui ch2 , caracterul "1" Atenție: caracterul "1" este interpretat de către calculator în mod diferit de întregul "1"!

Dacă se dorește și citirea spațiului dintre R și 1 atunci se utilizează cin.get()

```
cin .get(ch1);
cin .get(ch2);
cin .get(ch3);
```

Funcția ignore() se utilizează pentru ignorarea unui număr de date citite și are următorul format de apelare:

**cin .ignore(intreg, caracter);**

unde **intreg** reprezintă un nr. intreg sau o expresie de tip întreg

**caracter** este o constantă de tip caracter

**Ex.:** ignorarea primelor 200 de caractere introduse de la tastatura sau ignorarea tuturor caracterelor până se introduce caracterul de linie nouă. Se realizează condiția care este îndeplinită prima .

```
cin.ignore(200, '\n');
```

**Ex.:** se ignoră primele 100 de caractere introduse de la tastatură sau se ignoră toate caracterele până se întâlnește litera "B" : cin.ignore(100, 'B');

**Ex.:** se ignoră primele 2 caractere introduse de la tastatură : cin.ignore(2, '\n');

**Probleme rezolvate**

***Ex.1 : Programul C++ calculează pătratul și cubul unui nr. întreg utilizând cin/cout.***

**Varianta in C++**

```
#include <iostream>
#include <math.h>
using namespace std;
int Patrat( int n );
int Cub( int n);
int main()
{int a, b;
cout << "a= " ;  cin >> a  ;
cout << "patratul lui " << a << " este " << Patrat(a) << endl;
cout << "iar cubul lui " << a << " este " << Cub(a) << endl;
cout << "b="; cin >> b;
cout << a << " la puterea " << b << " este " << pow(a,b) << endl;
return 0; }
int Patrat( int n ) {     return n * n;}
int Cub( int n )    {     return n * n * n; }
```

***Rezultate:***

```
a= 2
patratul lui 2 este 4
iar cubul lui 2 este 8
b=10
2 la puterea 10 este 1024
```

***Aplicație:***

*Scrieți programul care calculează și afișează valorile funcției* $f(x) = \begin{cases} 3x^2 + 2x - 1, x < 0 \\ 5x + 2, x \geq 0 \end{cases}$ *, pentru orice valoare x introdusă de la tastatură cu cin*

***Ex.2. Programul C++ calculează și afișează valoarea ipotenuzei unui triunghi dreptunghic utilizând cin și cout.***

**Varianta in C++**

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{  float catetaA, catetaB,  ipoten;
cout << "Introduceti  catetele: " << endl;
cout << "cateta A="; cin >> catetaA ;
cout << "cateta B=" ;cin >> catetaB ;
ipoten = sqrt(catetaA * catetaA+ catetaB*catetaB);
cout << "Ipotenuza este:  " << ipoten << endl;
return 0;}
```

***Rezultate:***

```
Introduceti  catetele:
cateta A=50
cateta B=60
Ipotenuza este:  78.1025
```

***Aplicație:***

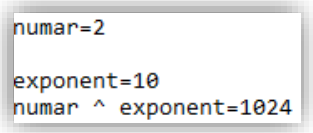
*Scrieți programul care rezolvă ec. de gradul I cu cin, cout*

***Ex.3 : Programul C++ calculează pătratului unui nr. întreg utilizând o funcție recursivă, și cin , cout.***

**Varianta in C++**

```
#include <iostream>
using namespace std;
int Power( int, int );
int main()
{  int number;           // Numarul  intreg
   int exponent;         // Exponentul intreg
   cout<<"numar=";cin >> number ;
   cout<<"\nexponent="; cin >> exponent;
   cout << "numar ^ exponent="<<Power(number, exponent);
return 0;}
int Power( int x,  int n )  // functia de ridicare la putere
// Presupunem :    n > 0
// Atentie la depasirea intervalului int pentru  exponent prea mare
{   if (n == 1)  return x;
    else return x * Power(x, n - 1); // apel recursiv
}
```

***Rezultate:***




***Aplicație:***

*Modificați programul astfel încât să calculați și afișati n! utilizând  cin, cout*

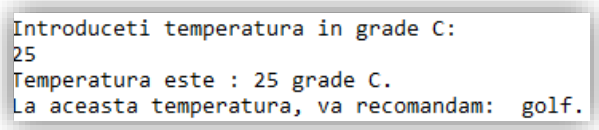
***Ex.4 : Programul C++ citește valoarea temperaturii (t) de afară și afișează o activitate recomandată conform tabelului de mai jos:***

| ***Temperatură*** | ***Activitate*** |
|---|---|
| *t>35˚C* | *inot* |
| *35>t>25* | *Tenis* |
| *25>t>20* | *golf* |
| *20>t>0* | *schi* |
| *t<0˚C* | *discoteca* |

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main()
{   int temperature;
cout << "Introduceti temperatura in grade C:" << endl; cin >> temperature;
cout << "Temperatura este : " << temperature << " grade C." << endl;
cout << "La aceasta temperatura, va recomandam:  "; if (temperature > 35)  cout << "inot." << endl;
else if (temperature > 25)  cout << "tenis." << endl;
else if (temperature > 20)  cout << "golf." << endl;
else if (temperature < 0)    cout << "schi." << endl;
else  cout << "discoteca." << endl;
return 0;}
```

***Rezultate:***




***Aplicație:***

Modificați programul astfel încât să calculați și afișați temperatura în grade Fahrenheit: [°C] = ([°F] − 32) × 5⁄9.

***Ex.5. Programul C++ calculează pentru orice x real, valoarea funcţiei f(x) definită prin expresia:***

$$f(x) = \begin{cases} 3x^2 + 2x - 10, x < 0 \\ x^2, x > 0 \\ 2, x = 0 \end{cases}$$

**Varianta in C++**

```
#include <iostream>
using namespace std;
int main(void)
{float x,f;
cout<<"introduceti valoarea lui x:";cin >>x;
if (x<0.) f=3.0*x*x+2.0*x-10.0;
  else if (x==0.) f=2.0;
    else f=x*x;
cout <<"f(x)=" << f <<endl;
return 0;}
```

***Rezultate:***

```
introduceti valoarea lui x:10.5
f(x)=110.25
```

***Aplicație:***

Modificați programul astfel încât să calculați și afișați valoarea funcției f(x) =$x^3$-1.

***Ex.6. Programul C++ calculează şi afişează valorile primilor n termeni ai şirului:*** $a_i = \left(1 + \frac{1}{i}\right)^i$***, calculându-se pentru fiecare din valorile lui*** $a_i$ ***şi abaterile acestora de la valoarea exactă a nr. e=2.7182818285. Problema se bazează pe următoarele limite cunoscute:*** $e = \lim_{n\to\infty}\left(1 + \frac{1}{n}\right)^n$ ***şi*** $e = \lim_{n\to\infty} \frac{n}{\sqrt[n]{n!}}$.

**Varianta in C++**

```
#include <iostream>
#include <math.h>
using namespace std;
int main  (void)
{int i, n;  double e,a;
cout << "n="; cin >>n; e=exp(1);
cout << "e=" << e << endl;
 for (i=1;i<=n;i++)  {a= pow((1.+1./i),i) ;
 cout << endl << a <<"\t" << ", abatere: "<< exp(1)-a ; }
 cout << endl;
 return 0;}
```

***Rezultate:***

```
n=10
e=2.71828

2       , abatere: 0.718282
2.25    , abatere: 0.468282
2.37037 , abatere: 0.347911
2.44141 , abatere: 0.276876
2.48832 , abatere: 0.229962
2.52163 , abatere: 0.196655
2.5465  , abatere: 0.171782
2.56578 , abatere: 0.152497
2.58117 , abatere: 0.137107
2.59374 , abatere: 0.124539
```

***Ex.7. Programul C++ calculează şi afişează rădăcinile ecuaţiei de gradul II: $ax^2+bx+c$, pentru a,b,c numere reale introduse de la tastatură. Se vor utiliza 2 funcţii: citire()-pentru citirea coeficienţilor ecuaţiei şi rezolv()-pentru rezolvarea ecuaţiei.***

**Varianta in C++**

```
#include <iostream>
#include <math.h>
using namespace std;
double a,b,c,y,y2,x1,x2,D,x;
char i;
void citire()
{ cout <<"Fie ecuatia de gradul 2: aX*X + bX + c = 0" << endl;
cout << endl <<"Introduceti coeficientii ecuatiei: " <<endl;
cout << endl << "a=";cin >> a;
cout << "b="; cin >> b;
cout << "c=";cin >> c;}
void rezolv()
{ if(a==0) { cout << endl <<"Ecuatia este de gradul I";
                        if(b!=0) {x=-c/b;
                                cout << endl <<"Solutia ecuatiei este: " << x;}
                        else cout <<"Eroare. Introduceti din nou coeficientii";   }
              else
             {cout << "Ecuatia este " << a <<"X*X +"<<b <<"X +" << c;
    cout  << "= 0" << endl;
D=b*b-4*a*c;
cout << endl <<"Discriminantul ecuatiei:" <<D;
if(D>=0) { x1=(-b+sqrt(D))/(2.*a);
              x2=(-b-sqrt(D))/(2.*a);
            cout <<endl << "Ecuatia are radacini reale: x1 = " << x1 <<", x2 = "<<x2;
           }
else { x1=-b/(a*2.);y=sqrt(-D)/(2.*a);
x2=-b/(a*2.);y2=sqrt(-D)/(2.*a);
cout << endl << "Ecuatia are radacini complexe:" << endl;
cout << "z1 = " << x1 << "+ " <<y <<"*i, z2 =" << x2 << " -" << y2;
cout << "*i" << endl;}
}}
int main()
{citire();
rezolv();
return 0;}
```

***Rezultate:***

```
Fie ecuatia de gradul 2: aX*X + bX + c = 0

Introduceti coeficientii ecuatiei:

a=2
b=5.5
c=-1.4
Ecuatia este 2X*X +5.5X +-1.4= 0

Discriminantul ecuatiei:41.45
Ecuatia are radacini reale: x1 = 0.234542, x2 = -2.98454
```

***Ex.8. Programul C++ determină şi afişează n termeni ai şirului lui Fibonacci utilizând 3 metode diferite:***

***a) Metoda recursivă ţinând cont de relaţiile:*** $f_k=f_{k-1}+f_{k-2}$, $f_0=0$, $f_1=1$, $k\geq 2$

***b) O metodă nerecursivă***

***c) O altă metoda nerecursivă ţinând cont de relaţiile:*** $f_k = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k\right]$,

$$\Phi^{k-2} \leq f_k \leq \Phi^{k-1}$$

$\Phi = \frac{1+\sqrt{5}}{2} \approx 1.6180339887$ *= raportul de aur*

**Varianta in C++**

```
#include <iostream>
#include <math.h>
using namespace std;
int n, k;
int main ()
{int n, a=1,b=0,c,f1[100], f2[100],f3[100];
double fi1, fi2;
f1[0]=f2[0]=f3[0]=0;
f1[1]=f2[1]=f3[1]=1;
fi1=(1+sqrt(5))/2;
fi2=(1-sqrt(5))/2;
cout <<"Introduceti n>=0, n="; cin >>n;
if (n==0)
{cout <<"f1[ 0]=" << f1[0]; cout <<"f2[ 0]=" << f2[0]; cout <<"f3[ 0]=" << f3[0];}
else if (n==1)
{cout <<"f1[ 1]=" << f1[1]; cout <<"f2[ 1]=" << f2[1]; cout <<"f3[ 1]=" << f3[1]; }
for (k=2;k<=n;k++)
//metoda recursiva 1
{f1[k]=f1[k-1]+f1[k-2]; cout <<"f1[ "<< k << "]=" << f1[k] << "\t";
//metoda nerecursiva 2
c=a; a+=b;b=c;f2[k]=a; cout <<"f2[ "<< k << "]=" << f2[k] << "\t";
//metoda nerecursiva 3
f3[k]=round((1/sqrt(5))*(pow(fi1,k)-pow(fi2,k))); cout << "f3[ "<< k <<"]=" << f3[k] << endl ;}
return 0;}
```

***Rezultate:***

```
Introduceti n>=0, n=15
f1[ 2]=1        f2[ 2]=1        f3[ 2]=0
f1[ 3]=2        f2[ 3]=2        f3[ 3]=2
f1[ 4]=3        f2[ 4]=3        f3[ 4]=3
f1[ 5]=5        f2[ 5]=5        f3[ 5]=5
f1[ 6]=8        f2[ 6]=8        f3[ 6]=8
f1[ 7]=13       f2[ 7]=13       f3[ 7]=13
f1[ 8]=21       f2[ 8]=21       f3[ 8]=21
f1[ 9]=34       f2[ 9]=34       f3[ 9]=34
f1[ 10]=55      f2[ 10]=55      f3[ 10]=55
f1[ 11]=89      f2[ 11]=89      f3[ 11]=89
f1[ 12]=144     f2[ 12]=144     f3[ 12]=144
f1[ 13]=233     f2[ 13]=233     f3[ 13]=233
f1[ 14]=377     f2[ 14]=377     f3[ 14]=377
f1[ 15]=610     f2[ 15]=610     f3[ 15]=610
```

## PROBLEME PROPUSE:

1. ***Să se scrie un program C++ utilizând cin și cout care să citească un șir de la tastatură și să:***
   - ***determine și afișeze, elementele minim și maxim ,***
   - ***calculeze și afișeze suma și produsul elementelor***
   - ***ordoneze șirul prin metoda bulelor***

2. ***Să se scrie un program C++ utilizând cin și cout care să citească un șir de caractere și să:***
   - ***afișeze șirul invers,***
   - ***numere vocalele și consoanele din șir***
   - ***afișeze toate literele cu majuscule***

3. ***Să se scrie un program C++ care citește de la tastatură un număr n întreg și valorile reale ale n rezistențe și calculezează și afișează valoarea rezistenței echivalente serie și paralel.***

4. ***Să se scrie un program C++, utilizând cin și cout, în care se definește o variabilă tablou de structuri numită angajați de tip structură cu câmpurile: nume, adresă, cod numeric, salar net și inițializați acest tablou cu n articole de acest tip (n introdus de la tastatură). Să se afișeze numai angajații care au salarul cuprins între 400 si 1.000 RON.***

5. ***Să se scrie un program C++, utilizând cin și cout, care realizează următoarele operații:***
   - ***citește de la tastatură un număr întreg reprezentând studenții unei grupe, numele, prenumele și 2 note pentru fiecare student***
   - ***calculează media aritmetică a notelor pentru fiecare student***
   - ***afișează studenții sortați în ordinea mediilor.***

# Capitolul 11
# Structurarea modulara a programelor C/C++ în mai multe fișiere

În acest capitol sunt prezentate consideraţii teoretice și probleme rezolvate privind structurarea programelor în mai multe fisiere/module.

## CONSIDERAŢII TEORETICE

**Reguli de editare ale fișierelor:**

- se alege un fișier care va fi **fișierul principal**, în acest fișier fiind implementată funcția **main();** în celelalte fișiere nu va mai fi implementată această funcție.
- În **fișierul antet** (cu extensia .h) se vor trece **definiția tipurilor de date** folosite **și declarațiile (prototipurile) funcțiilor**. **Nu se vor trece în fișierul antet: declarații de variabile și implementări de funcții**, altfel nu există modularizarea și încapsularea datelor
- În **fișierul auxiliar antetului** (cu același nume și **extensia .c**) se va trece implementarea funcțiilor definite în fișierul antet. **Nu se va implementa funcția main()** în acest fișier.
- Atât **fișierul principal** cât **și** cel **auxiliar vor include fișierul antet** $\Rightarrow$ încapsularea tipurilor de date și a funcțiilor în fișiere separate.
- Această manieră de structurare a programelor este folosită în scrierea unor biblioteci de funcții.

**Reguli de compilare a fișierelor:**

- se editează fișierele cu extensiile .h, .c
- se includ în același proiect fișierele cu extensia .c, se verifică dacă fișierele sunt incluse în același director sau dacă s-a specificat corect calea în care se găsesc fișierele
- se compilează și link-editează fișierele
- se corectează erorile de sintaxă și link-editare
- se verifică rezultatele.

## PROBLEME REZOLVATE

***Ex.1: Programul structurat pe mai multe fișiere ilustrează operațiile de adunare, scădere, înmulțire, împărțire și afișare a modulului și argumentelor numerelor complexe.***

- ***Adunarea/scăderea:*** $x+iy=(x_1+iy_1)\pm(x_2+iy_2)$

- ***Produsul:*** $x+iy=(x_1+iy_1)(x_2+iy_2)=(x_1x_2-y_1y_2)+i(x_1y_2+x_2y_1)$

- ***Câtul:***

$$x+iy=\frac{x_1+iy_1}{x_2+iy_2}=\frac{x_1x_2+y_1y_2}{{x_2}^2+{y_2}^2}+i\frac{x_2y_1-x_1y_2}{{x_2}^2+{y_2}^2},$$
$${x_2}^2+{y_2}^2>0$$

- ***Valoarea absolută:*** $|x+iy|=\sqrt{x^2+y^2}$.
- ***Argumentul:***

$$arg(x+iy)=\begin{cases} arctg(\frac{y}{x}), arctg(\frac{y}{x})>=0 \\ arctg(\frac{y}{x})+2\pi, arctg(\frac{y}{x})<0 \end{cases}$$

*Programul va fi structurat pe trei fișiere:*

- *main.c- care conține funcția main*
- *operatii.h- fișierul antet unde sunt declarate principalele tipuri de date și functii și*
- *operatii.c– fișierul auxiliar pentru implementarea funcțiilor.*

*Se editează mai întâi fișierul main.c, apoi operatii.h și operatii.c. Se include în același proiect și calcul.c și operatii .c și se execută programul calcul.c*

**Varianta in C**

**//main.c**

```
#include <stdio.h>
#include "operatii.h"
int main()
{ complex x,y;
  printf("introduceti primul numar complex: x\n");
  x=citire();
  printf("introduceti al 2-lea numar complex: y\n");
  y=citire();
  printf("\nsuma: x+y= ");
  afisare(suma(x,y));  printf("\n");
  printf ("scadere: x-y=");
  afisare (scadere(x,y)); printf ("\n");
  printf ("impartire: x/y=");
  afisare (cat(x,y)); printf ("\n");
  printf ("abs(x)=%lf,arg(x)=%lf", absolut(x), argument(x));
  printf ("\n");
  printf ("abs(y)=%lf,arg(y)=%lf", absolut(y), argument(y));
  printf ("\n");
  return 0;}
```

**//operatii.h**

```
#ifndef OPERATII_H_INCLUDED
#define OPERATII_H_INCLUDED
// prototipurilor de functii
/* definire tip de date complex */
typedef struct nr_complex
  {    double pr;
       double pi;
  } complex;
/* prototipuri functii exportate */
complex citire(void);
void afisare(complex);
complex suma(complex,complex);
complex scadere(complex, complex);
complex cat(complex, complex);
double absolut(complex);
double argument(complex);
#endif
```

**Varianta in C++**

**//main.cpp**

```
#include <iostream>
#include "operatii.h"
using namespace std;
int main()
{ complex x,y;
  cout<<"introduceti primul numar complex: x"<<endl;
  x=citire();
  cout<<"introduceti al 2-lea numar complex: y"<<endl;
  y=citire();
  cout<<endl<<"suma: x+y= ";
  afisare(suma(x,y));  cout<<endl;
  cout<<"scadere: x-y=";
  afisare (scadere(x,y)); cout<<endl;
  cout<<"impartire: x/y=";
  afisare (cat(x,y)); cout<<endl;
  cout<<"abs(x)="<<absolut(x)<<", arg(x)="<<argument(x);
  cout<<endl;
  cout<<"abs(y)="<<absolut(y)<<", arg(y)="<<argument(y);
  cout<<endl;
return 0;}
```

**//operatii.h**

```
#ifndef OPERATII_H_INCLUDED
#define OPERATII_H_INCLUDED
// prototipurilor de functii
/* definire tip de date complex */
typedef struct nr_complex
  {    double pr;
       double pi;
  } complex;
/* prototipuri functii exportate */
complex citire(void);
void afisare(complex);
complex suma(complex,complex);
complex scadere(complex, complex);
complex cat(complex, complex);
double absolut(complex);
double argument(complex);
#endif // OPERATII_H_INCLUDED
```

**Varianta in C**

```
//Fisierul operatii.c : definirea functiilor
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "operatii.h"
/* implementare functii exportate */
complex citire(void)
{complex nr;
  printf("Partea reala: ");  scanf("%lf",&nr.pr);
  printf("Partea imaginara: ");  scanf("%lf",&nr.pi);
  return nr;}
void afisare(complex nr)
{  if (nr.pi<0)
      printf("%.2lf%.2lfi\n",nr.pr,nr.pi);
    else
      printf("%.2lf+%.2lfi\n",nr.pr,nr.pi);}
complex suma(complex nr1, complex nr2)
{complex rez1;
  rez1.pr=nr1.pr + nr2.pr;
  rez1.pi=nr1.pi + nr2.pi;
  return rez1;}
complex scadere(complex nr1, complex nr2)
{complex rez2;
  rez2.pr=nr1.pr - nr2.pr;
  rez2.pi=nr1.pi - nr2.pi;
  return rez2;}
complex cat(complex nr1, complex nr2)
{complex rez3;
  double num;
  num= pow(nr2.pr,2)+pow(nr2.pi,2);
              if (num == 0.) {printf ("numarator =0!"); exit (1);}
  else {   rez3.pr=(nr1.pr*nr2.pr+nr1.pi*nr2.pi)/num;
             rez3.pi=(nr2.pr*nr1.pi-nr1.pr*nr2.pi)/num;  }
  return rez3;}
double absolut(complex nr)
{double abs;
  abs = sqrt(pow(nr.pr,2)+pow(nr.pi,2));
  return abs;}
double argument(complex nr)
{double  a;
  a=atan(nr.pi/nr.pr);
  if (a>=0.) return a;
  else  return (a+2.*3.14);}
```

**Varianta in C++**

```
//Fisierul operatii.cpp : definirea functiilor
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include "operatii.h"
using namespace std;
/* implementare functii exportate */
complex citire(void)
{complex nr;   cout<<"Partea reala: ";  cin>>nr.pr;
  cout<<"Partea imaginara: ";  cin>>nr.pi;
  return nr;}
void afisare(complex nr)
{  if (nr.pi<0)
      cout<<nr.pr<<nr.pi<<"i"<<endl;
    else
      cout<<nr.pr<<"+"<<nr.pi<<"i"<<endl;}
complex suma(complex nr1, complex nr2)
{complex rez1;
  rez1.pr=nr1.pr + nr2.pr;
  rez1.pi=nr1.pi + nr2.pi;
  return rez1;}
complex scadere(complex nr1, complex nr2)
{complex rez2;
  rez2.pr=nr1.pr - nr2.pr;
  rez2.pi=nr1.pi - nr2.pi;
  return rez2;}
complex cat(complex nr1, complex nr2)
{complex rez3;
  double num;
  num= pow(nr2.pr,2)+pow(nr2.pi,2);
              if (num == 0.) {cout<<"numarator =0!"; exit (1);}
  else {   rez3.pr=(nr1.pr*nr2.pr+nr1.pi*nr2.pi)/num;
             rez3.pi=(nr2.pr*nr1.pi-nr1.pr*nr2.pi)/num;  }
  return rez3;}
double absolut(complex nr)
{double abs;
  abs = sqrt(pow(nr.pr,2)+pow(nr.pi,2));
  return abs;}
double argument(complex nr)
{double  a;
  a=atan(nr.pi/nr.pr);
  if (a>=0.) return a;
  else  return (a+2.*3.14);}
```

***Rezultate:***

```
introduceti primul numar complex: x
Partea reala: 2.5
Partea imaginara: 3.2
introduceti al 2-lea numar complex: y
Partea reala: 1.5
Partea imaginara: 4.2

suma: x+y= 4.00+7.40i

scadere: x-y=1.00-1.00i

impartire: x/y=0.86-0.29i

abs(x)=4.060788,arg(x)=0.907593
abs(y)=4.459821,arg(y)=1.227772
```

***Aplicație:***

*Să se modifice programul astfel încât, utilizând pointeri, să se adauge o funcție care verifică egalitatea celor două numere complexe .*

***Ex.2. Programul structurat pe mai multe fișiere realizează următoarele operații:***

- ***citește numărul studenților (dimensiunea n a tabloului), numele, prenumele și 2 note pentru fiecare student***
- ***calculează media aritmetică a notelor pentru fiecare student, și***
- ***afișează studenții sortați prin Metoda Bulelor în ordinea descrescătoare a mediilor***

**Varianta in C**

**//fisier main.c**

```
#include <stdio.h>
#include "date.h"
int main()
{student s[100];
int i,n;
printf("Introduceti numarul de studenti:");
scanf("%d", &n);
for (i=0;i<n;i++)  s[i]=citire();
printf("\nLista studentilor:\n");
for (i=0;i<n;i++)
  {s[i].med=(s[i].nota1+s[i].nota2)/2; afisare(s[i]);}
printf("\nLista studentilor in ordinea descrescatoare a
mediilor\n");
for (i=0;i<n;i++)
   {sortare(s,n);afisare(s[i]);}
return 0;}
```

**//fisier date.h**

```
typedef struct stud {
            char name[20];
            char prenume[20];
            double nota1;
            double nota2;
            double med;} student;
/* prototipuri functii exportate */
student citire(void);
void afisare(student);
void sortare(student s[], int n);
```

**Varianta in C++**

**//fisier main.c**

```
#include <iostream>
#include "date.h"
using namespace std;
int main()
{student s[100];
int i,n;
cout<<"Introduceti numarul de studenti:";
cin>>n;
for (i=0;i<n;i++)  s[i]=citire();
cout<<endl<<"Lista studentilor:"<<endl;
for (i=0;i<n;i++)
  {s[i].med=(s[i].nota1+s[i].nota2)/2; afisare(s[i]);}
cout<<endl<<"\nLista studentilor in ordinea
descrescatoare a mediilor"<<endl;
for (i=0;i<n;i++)
  {sortare(s,n);afisare(s[i]);}
return 0;}
```

**//fisier date.h**

```
#ifndef DATE_H_INCLUDED
#define DATE_H_INCLUDED
typedef struct stud {
            char name[20];
            char prenume[20];
            double nota1;
            double nota2;
            double med;} student;
/* prototipuri functii exportate */
student citire(void);
void afisare(student);
void sortare(student s[], int n);
#endif // DATE_H_INCLUDED
```

**Varianta in C**

```
//functii.c
#include <math.h>
#include "date.h"
/* implementare functii exportate */
student citire(void)
{ student stud;
 printf("\nIntroduceti numele studentului: ");
 scanf("%s", stud.name);
 printf("Introduceti prenumele studentului: ");
 scanf("%s",stud.prenume);
 printf("Introduceti nota1 a studentului: ");
 scanf("%lf", &stud.nota1);
 printf("Introduceti nota2 a studentului: ");
 scanf("%lf", &stud.nota2);
 return stud;}
void afisare(student stud)
{printf("\n%6s %6s, nota1=%5.2lf, nota2=%5.2lf,
media=%5.2lf", stud.name,
stud.prenume,stud.nota1,stud.nota2,stud.med);
//atentie linie continuata!
}
void sortare(student st[], int n)
{           student aux[100];
            int i,k;
do
 {k=0;
  for (i=0;i<n-1;i++)
    {if (st[i].med<st[i+1].med)
            { aux[i]=st[i] ;  st[i]=st[i+1];   st[i+1]=aux[i];k=1;}
    }
  }
while (k);}
```

**Varianta in C++**

```
//functii.c
#include <math.h>
#include<iostream>
#include "date.h"
using namespace std;
/* implementare functii exportate */
student citire()
{  student stud;
 cout<<"\nIntroduceti numele studentului: ";
 cin>>stud.name;
 cout<<"Introduceti prenumele studentului: ";
 cin>>stud.prenume;
 cin.ignore();
 cout<<"Introduceti nota1 a studentului: ";
 cin>>stud.nota1;
 cout<<"Introduceti nota2 a studentului: ";
 cin>>stud.nota2;
 return stud;}
void afisare(student stud)
{cout<<endl<<stud.name<<"  "<<stud.prenume<<",
nota1="<<stud.nota1<<", nota2="<<stud.nota2<<",
media="<<stud.med;
//atentie linie continuata!
}
void sortare(student st[], int n)
{           student aux[100];
            int i,k;
do
 {k=0;
  for (i=0;i<n-1;i++)
    {if (st[i].med<st[i+1].med)
            { aux[i]=st[i] ;  st[i]=st[i+1];
st[i+1]=aux[i];k=1;}}}
while (k);}
```

***Rezultate:***

```
Introduceti numele studentului: Popa
Introduceti prenumele studentului: Ioan
Introduceti nota1 a studentului: 5.5
Introduceti nota2 a studentului: 7.5

Introduceti numele studentului: Rusu
Introduceti prenumele studentului: Larisa
Introduceti nota1 a studentului: 10
Introduceti nota2 a studentului: 9.5

Introduceti numele studentului: Boca
Introduceti prenumele studentului: Vlad
Introduceti nota1 a studentului: 8.5
Introduceti nota2 a studentului: 9.5

Lista studentilor:

  Popa   Ioan, nota1= 5.50, nota2= 7.50, media= 6.50
  Rusu Larisa, nota1=10.00, nota2= 9.50, media= 9.75
  Boca   Vlad, nota1= 8.50, nota2= 9.50, media= 9.00

Lista studentilor in ordinea descrescatoare a mediilor

  Rusu Larisa, nota1=10.00, nota2= 9.50, media= 9.75
  Boca   Vlad, nota1= 8.50, nota2= 9.50, media= 9.00
  Popa   Ioan, nota1= 5.50, nota2= 7.50, media= 6.50
```

***Aplicație:***

*Să se modifice programul astfel încât, să se introducă și un element cod grupă (întreg , ex.1123) și să se realizeaze sortarea pe grupe și medii.*

## PROBLEME PROPUSE

1. ***Să se scrie un program structurat pe mai multe fișiere care realizează următoarele operații:***
    - ***citește numele, media la bacalaureat și media la testul grilă a unui tablou de structuri de n studenți,***
    - ***calculează media aritmetică a notelor pentru fiecare student, și***
    - ***afișează studenții admiși (cei cu media peste 5) sortați în ordinea descrescătoare a mediilor***
    - ***afișeaza în ordine alfabetică studenții respinși (cu media sub 5).***

2. ***Să se scrie un program structurat pe mai multe fișiere care realizează operațiile specifice matricilor (adunare, înmulțire, înmulțire cu o constantă, transpusa, inversa, etc.)***

3. ***Să se scrie un program structurat pe mai multe fișiere care definește un tablou de structuri de tip bibliotecă cu câmpurile: titlu, autori, anul publicării, editura și:***
    - ***citește n articole de acest tip de la tastatură***
    - ***sortează alfabetic cărțile după titlu și autori***
    - ***afișează articolele ordonate după anul apariției și publicate în ultimii 5 ani.***

# Capitolul 12
# Diferențe între C și C++. Programarea orientată pe obiecte. Clase și obiecte. Moștenire,constructori,destructori.Funcții și clase prietene

În acest capitol sunt prezentate consideraţii teoretice și probleme rezolvate privind diferențele semnificative între limbajele C și C++, definirea și utilizarea noțiunilor specifice programării orientate pe obiecte. Este de asemenea ilustrat modul în care se definesc clasele și componentele lor, operatorul de rezoluție, constructorii și destructorii, funcțiile prietene, conceptul de moștenire, etc.

## CONSIDERAŢII TEORETICE

În general, nu există diferențe între fișierele sursă din C si C++. Ambele limbaje acceptă directivele de compilator (Ex. #include și #define). Fișierele program scrise în C++ vor fi salvate cu extensia .cpp pentru a le diferenția de cele scrise în limbajul C.

În cele ce urmează sunt prezentate foarte pe scurt câteva dintre diferențele semnificative dintre cele două limbaje **C** și respectiv **C++**.

**1.** În C++ sunt posibile **conversiile de tip** (type casting)
În C și C++ conversiile de tip se realizează diferit.

**Ex**.:

**in C:**
```
int n;
float x;
x = (float) n;
```

**in C++:**
```
int n;
float x;
x = float(n);
```

**2.** În C++ se utilizează **cin** >> și respectiv **cout** << în locul funcțiilor scanf() și printf() (detalii cap.10).

**3.** În C++ **declarațiile variabile** se pot face oriunde în program
În C, declarațiile de tip ale variabilelor se realizează de regulă imediat după acolada deschisă a unei funcții (main sau altă funcție utilizator).

În C++ declarațiile de tip se pot realiza oriunde în program.
**Ex**.: variabila i este declarată în corpul funcției for în C++.

**în C**
```
#include <stdio.h>
void main()
{int i;
for (i=0;i<10;i++)
printf("%d\n", i);}
```

**în C++**
```
#include <iostream.h>
void main()
{for (int i=0;i<10;i++)
cout << i << '\n';}
```

**Ex:** program în C++
```
int a;a = 10;
cout << a << endl; // afiseaza 10
{ double a, b = 2.0; a = 3.14 * b;
cout << a << endl;} // afiseaza 6.28
    a *= 3; cout << a << endl; // afiseaza 18.84
```

**4.** În C++ este posibil **apelul prin referința (&)**

Există două posibilități de transmitere a parametrilor actuali către o funcție în C++:

- prin valoare
- prin referință

Transferul prin referință este util și atunci când parametrul are dimensiune mare (struct, class) și crearea în stiva a unei copii a valorii lui reduce viteza de execuție .

**Ex**.: interschimbarea a valorilor a două variabile prin referință

```
void schimba(int &a, int &b)
{int  aux=a;
a=b; b=aux;}
```

**5.** În C++ se pot defini **funcții care returnează variabile**

**Ex.:**

```
float &var_max(float &x, float &y)
{if (x > y) return x;
else return y;
}
```

**6.** În C++ se pot utiliza **funcțiile inline**

În C++ se utilizează  funcții inline pentru a crea programe mai rapide .

**Ex. :**

```
inline double ipot(double a, double b)
{     return sqrt(a*a + b*b);};
```

**7.** În C++ se pot utiliza **parametri impliciți**

La apelarea unei funcții cu parametri impliciți se poate omite specificarea parametrilor efectivi pentru acei parametri formali care au declarate valori implicite și se transferă automat valorile respective. Se pot specifica mai multe argumente cu valori implicite pentru o funcție.

**Ex**.:

```
int fct(int x=7, int y=9)
{      return x+y; }
void main()
{cout << fct() << endl; // 16
cout << fct(2) << endl; // 11
cout << fct(2, 3) << endl;} // 5
```

**8.** În C++ este posibilă **supraîncărcarea funcțiilor**

Există posibilitatea de a atribui unui simbol mai multe semnificații, care pot fi deduse în funcție de context. În particular, majoritatea operatorilor C++ pot fi priviți ca nume de funcții și deci pot fi supraîncărcați.

**Ex**.:

```
void fct(int a)
{  cout << "functia 1" << a;}
void fct(char *a)
{    cout << "functia 2" << a;}
```

**9.** În C++ **alocarea dinamică a memoriei** se realizează utilizând operatorii new și delete
Operatorii **new** și **delete** sunt similari funcțiilor din grupul malloc() și free(), dar constituie o metoda nouă, superioară acestora și adaptată programării orientate pe obiecte.

Operatorul delete este complementarul lui new și înlocuiește funcția free() de dezalocare a memoriei dinamice alocate.

**Ex**:
```
int * ip1, *ip2, *ip3;
ip1=new int;      // variabila întreaga neinitializata
ip2=new int(2);   // variabila întreaga initializata cu 2
ip3=new int[100]; // tablou de 100 de întregi
```

**10.** În C++ se utilizează obiecte de tip **clasă** care stau la baza programării orientate pe obiecte.

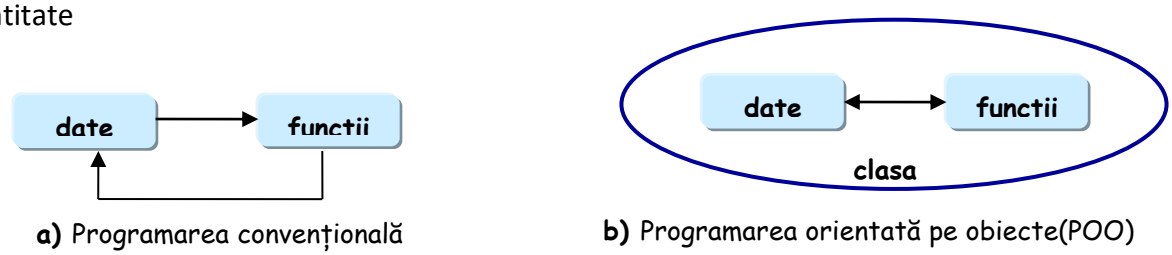
**Programarea orientata pe obiecte (POO)=** o metodă de implementare în care programele sunt organizate ca și colecții de obiecte ce cooperează între ele, fiecare obiect reprezentând instanța unei clase; Fiecare clasă aparține unei ierarhii de clase, clasele fiind unite prin relații de mostenire.

Proprietățile POO sunt:
- obiectele și nu algoritmii sunt blocurile logice fundamentale;
- fiecare obiect este o instanță a unei clase;
- clasele sunt legate între ele prin relații de moștenire.

**Limbaj de programare bazat pe obiecte =** Un limbaj de programare care oferă suport pentru utilizarea claselor și a obiectelor .
Caracteristicile POO:
- **Abstractizarea: O abstracțiune** exprimă **toate caracteristicile esențiale ale unui obiect**, care fac ca acesta să se distingă de alte obiecte;
- **Încapsularea:**este conceptul complementar abstractizării, și reprezintă **procesul de compartimentare a elementelor care formează structura și comportamentul unei abstracțiuni.** Ex.: încapsularea este un mecanism care leagă într-o variabilă numită clasa, date și funcții, restricționând accesul la acestea .
- **Modularitatea: Clasele și obiectele obținute în urma abstractizării și încapsulării** trebuie grupate și apoi stocate într-o formă fizică, denumită **modul**. Modularizarea constă în divizarea programului într-un număr de module care pot fi compilate separat, dar care sunt conectate între ele. Ex. în C++ modulele sunt fișiere ce pot fi compilate separat. Interfața modulului este plasata într-un fișier header (extensii uzuale sunt: ".h" , ".hpp", ".hh"), iar implementarea acestuia se va regăsi într-un fișier sursă (extensii uzuale sunt: ".cc", ".cpp", ".c").
- **Ierarhizarea:** reprezintă o **ordonare a abstracțiunilor**. Cele mai importante ierarhii de clase în paradigma obiectuală sunt: ierarhia de clase (relație de tip "is a") și ierarhia de obiecte (relație de tip "part of").
- **Moștenirea:** definește o **relație între clase** în care o clasa împărtășește structura și comportarea definită în una sau mai multe clase (după caz vorbim de moștenire simplă sau multiplă).

Cea mai semnificativă diferență între programarea convențională și POO este ilustrată în Fig. 1:

- în **programarea convențională** datele sunt preluate și retransmise de funcții
- în **programarea orientată pe obiecte** , clasele încapsulează datele și funcțiile într-o singură entitate




**a)** Programarea convențională **b)** Programarea orientată pe obiecte(POO)

**Fig.1.** Programarea convențională - POO

**Clasa = abstractizare** similară cu o structură**,** utilizată pentru declararea unui nou tip de date care încapsulează date și funcții;

**Obiect =**exemplar (**instanță**) a unei clase

**Elementele unei clase** = o clasă în C++ are asociate 4 tipuri de elemente:

- colecție de **date membre** (atribute)
- colecție de **funcții membre** (metode)
- **nivele de acces** ale programului (public, private, ...)
- **nume**

La **declararea clasei**, se specifică:

- **numele clasei**,
- **lista claselor de bază** din care e derivată clasa, dacă există, și
- **membrii clasei**, atât membrii de date cât și funcții.

**Implementarea clasei** constă în

- **definițiile funcțiilor** componente care indică comportamentul tipului de date reprezentat de clasa respectivă. Dacă funcțiile prezente în declararea clasei sunt corect adecvate scopului propus, atunci utilizatorul nu mai are nevoie de definirea lor și implementarea se poate face într-un fișier separat (.*cpp*).

**Formatul de declarare** a unei **clase** care **nu moștenește altă clasă:**

***class*** nume_clasă
{
**variabile** și funcții particulare
**specificator de acces**:
variabile și funcții
specificator de acces:
variabile și funcții
…
specificator de acces:
variabile și **funcții**
} [**Listă_de_obiecte**];

unde :

- **variabile** se declară cu tip: int,double,float,char,etc.
- **specificatori de acces** pot fi:
  - **public**: variabilele (datele) și funcțiile declarate cu acest specificator sunt vizibile (accesibile) din orice zonă a programului.
  - **private**: variabilele și funcțiile declarate cu acest specificator pot fi accesate doar de către membrii clasei.
  - **protected**: variabilele și funcțiile declarate cu acest specificator pot fi modificate în cadrul clasei sau în clasele derivate de ea
- Funcțiile se implementează astfel:
  - **funcții inline** (în interiorul clasei)
  - **operatorul de rezoluție** ::(în afara clasei)
- **Lista_de_obiecte** este opțională, iar declararea obiectelor se poate face și astfel:
  - nume_clasă Listă_de_obiecte;

Referitor la specificatorii de acces (vizibilitate) se pot face următoarele observații:

- implicit, variabilele și funcțiile declarate într-o clasă sunt proprii (private) acelei clase și numai membrii ei au access la ele.
- specificatorii de acces au **efect** până când se întâlnește un alt specificator de acces sau se ajunge la sfârșitul declarației de clasă.
- specificatorii de acces pot **alterna** în declararea unei clase. public, private,public…

Pentru exemplificarea modului în care se utilizează clasele, se consideră exemplul definirii unui **articol tip produs** care se poate realiza fie printr-o structură fie printr-o clasă:

**Ex**:
**a)** utilizând o structură

```
struct Produs {
        char comp_id[4];    // identificator companie max. 4 caractere.
        char prod_id[8];   // identificator produs max 8 caractere
        int  pret;       // pretul produsului in USD.
        int  stoc;  // cantitate existenta in stoc
      };
```

**b)** utilizând o clasă:

```
class Produs {
   public:
        char comp_id[4];    // 4 char code for the manufacturer.
        char prod_id[8];   // 8-char code for the product
        int  pret;       // price of the product in dollars.
        int  stoc;  // quantity on hand in inventory
      };
```

Definirea funcțiilor membre ale claselor se poate face fie în interior fie în exteriorul clasei.

**Ex.**
**a)** Definirea funcției membre în interiorul clasei

```
class adunare {
public:
        int suma(int a,int b) {
        int rez;
        rez=a+b;
        return rez;}};
```

**b)** Definirea funcției membre în exteriorul clasei

```
class adunare {
public:
        int suma(int a,int b) };
int adunare::suma(int a,int b) {
        int rez;
        rez=a+b;        return rez;}
```

**Operatorul de rezoluție ::** este un operator de specificare a domeniului și este utilizat pentru a specifica din ce domeniu (clasă) face parte o anumită funcție membră. Trebuie luat în considerare și faptul că mai multe clase diferite pot sa folosească același nume de funcție .

Accesul la componentele unei clase se realizează similar cu accesul la elementele unei structuri, prin:

- **Instanțierea** (declararea) unui obiect
  nume_clasă nume obiect;
- Accesarea componentelor clasei cu operatorul .
  nume_obiect.nume_funcție;
- Accesarea componentelor clasei prin pointeri cu operatorul "sageată" ->
  nume_pointer->nume_funcție;

Noțiunea de **moștenire** a fost introdusă în C++ pentru a permite construirea unei ierarhii de clase. Procesul de ierarhizare constă în :

- crearea unei **clase de baza (parinte)=** cea mai generală descriere care stabilește calitățile comune ale tuturor obiectelor ce vor deriva din această bază
- crearea **claselor derivate (copii)** din clasa de bază, care vor include :
  - toate **caracteristicile clasei de bază** și în plus
  - **calități proprii** clasei respective.

**Forma generală de declarare a claselor derivate prin moștenire** este :

```
class nume_nou_clasă: [specificator_acces] clasa_moștenită
 {   //corpul noii clase
 }
```

Modul de acces corespunzător fiecărui specificator de acces este prezentat în tabelul urmator:

| Specificator acces | Acces permis pentru |
|---|---|
| public | Toate celelalte clase |
| private | Nici o altă clasă |
| protected | Numai clasele derivate |

**Funcție constructor =** funcție specială care este membru al unei clase și **are același nume cu acea clasă,** nu poate să returneze valori și nu conțin nici tipul de returnat
Un **constructor al unui obiect** este apelat atunci când este creat acel obiect. Un **destructor al unui obiect** va produce distrugerea obiectului creat (pentru eliberarea de memorie, etc.)

**Ordinea de apelare a constructorilor și destructorilor**

- Constructori: clasă bază ⇒clasă derivat1 ⇒clasă derivat2
- Destructori: clasă derivat2 ⇒clasă derivat1 ⇒clasă bază

**Ex. :** Utilizarea constructorului și destructorului unei stive:

```
class stiva {
        int stiv[SIZE];
        int vis;
public:
         stiva() ;//constructor
        ~stiva() ;//Destructor
        void pune(int i);
        int scoate();
};
stiva::stiva()    //functia constructor pentru stiva
{  vis=0;  cout << "Stiva initializata\n" ;  }
stiva::~stiva()          //functia destructor pentru stiva
{  cout << "Stiva distrusa!\n";  }
```

**Funcție prietenă (friend) =** funcție care are acces la membrii private și protected ai clasei căreia ii este prietenă.

**Format de declarare:**

```
class nume_clasa {
        variabile și funcții private
public:
        friend prototip_fctie();
        variabile și funcții publice;};
```

**Funcțiile prietene sunt utile la:**

- supraîncărcarea operatorilor
- simplifică creearea funcțiilor de I/O
- scrierea mai eficientă a programelor

**Clasa prietenă (friend) =** clasa care are acces la variabilele particulare (nume de tipuri și enumerări de constante) definite în cadrul celeilalte clase.

## PROBLEME REZOLVATE

**Ex.1: Programul în C++ definește o clasă de bază numită clădire cu**

- **datele membre:** camere, etaje și supraf și
- **funcțiile membre:** nr_camere(),nr_etaje(), cate_etaje(), ce_supraf(), cat_supraf()

și 2 clase derivate, una numită casa cu

- **datele membre:** dormit, bai și
- **funcțiile membre:** nr_dormit(), cate_dormit(), nr_bai(), cate_bai()

și alta numită scoala cu

- **datele membre:** saliclasa, lab și
- **funcțiile membre:** nr_saliclasa(), cate_saliclasa(), nr_lab(), cate_lab();

Se cere să se declare obiecte de tipul casă și școală și utilizând funcțiile membre să se afișeze diferite valori ale datelor membre.

**Varianta in C++**

```
#include <iostream>
using namespace std;
class cladire {
        int camere;
        int etaje;
        int supraf;
public:
        void nr_camere(int num);
        int cate_camere();
        void nr_etaje(int num);
        int cate_etaje();
        void ce_supraf(int num);
        int cat_supraf();
};
//clasa casa derivata din cladire
class casa: public cladire {
        int dormit;
        int bai;
public:
        void nr_dormit(int num);
        int cate_dormit();
        void nr_bai(int num);
        int cate_bai();
};
//clasa scoala derivata din cladire
class scoala: public cladire {
        int saliclasa;
        int lab;
public:
        void nr_saliclasa(int num);
        int cate_saliclasa();
        void nr_lab(int num);
        int cate_lab();
};
//definirea functiilor din clasa de baza
void cladire::nr_camere(int num)
{camere=num;}
void cladire::nr_etaje(int num)
{etaje=num;}
```

```
void cladire::ce_supraf(int num)
{supraf=num;}
int cladire::cate_camere()
{return camere;}
int cladire::cate_etaje()
{return etaje ;}
int cladire::cat_supraf()
{return supraf ; }
//definirea functiilor din clasa derivata casa
void casa::nr_dormit (int num)
{dormit=num; }
void casa::nr_bai (int num)
{bai=num; }
int casa::cate_dormit ()
{return dormit; }
int casa::cate_bai ()
{return bai; }
//definirea functiilor din clasa derivata scoala
void scoala::nr_saliclasa(int num)
{saliclasa=num; }
void scoala::nr_lab(int num)
{lab=num;}
int scoala::cate_saliclasa()
{return saliclasa; }
int scoala::cate_lab()
{return lab; }
//functia main()
int main(void)
{casa c; //obiect de tip casa
scoala s; //obiect de tip scoala
c.nr_camere(12);    c.nr_etaje(3);
c.ce_supraf(4500); c.nr_dormit(5);  c.nr_bai(3);
cout << "Casa are suprafata:" <<c.cat_supraf() <<" mp," << c.cate_etaje() << " etaje, si "   ;
cout << c.cate_camere() << " camere, din care:\n";
cout << "- dormitoare: " << c.cate_dormit() <<endl;
cout << "- camere diferite de dormitoare: " << c.cate_camere() - c.cate_dormit() <<endl;
cout << "- bai: " << c.cate_bai() <<endl;
s.nr_camere(200);   s.nr_etaje(3);
s.nr_saliclasa(180); s.nr_lab(5);  s.ce_supraf(25000);
cout << "\nScoala are suprafata: " << s.cat_supraf() <<" mp," ;
cout<<s.cate_etaje()<<"etaje, si"<<s.cate_camere() << "incaperi, din care:\n"; //atentie linie continuata!
cout << "-sali de clasa: " << s.cate_saliclasa() << endl;
cout << "-laboratoare: " << s.cate_lab() <<endl;
cout << "-alte sali(magazii, cancelarie, etc) : " << s.cate_camere() - s.cate_saliclasa()- s.cate_lab() <<endl;
return 0;}
```

***Rezultate:***

```
Casa are suprafata:4500 mp,3 etaje, si 12 camere, din care:
- dormitoare: 5
- camere diferite de dormitoare: 7
- bai: 3

Scoala are suprafata: 25000 mp,3etaje, si200incaperi, din care:
-sali de clasa: 180
-laboratoare: 5
-alte sali(magazii, cancelarie, etc) : 15
```

***Aplicație:***

*Să se modifice programul astfel încât să se declare și alte obiecte de tipul casă și școală și utilizând funcțiile membre să se afișeze diferite valori ale datelor membre.*

***Ex. 2. Programul C++ implementează clasa student cu următoarele funcții membru:***

- ***funcție prin care să se citească numele, prenumele și 5 note obținute la o sesiune de examene de către fiecare student;***
- ***funcție pentru afișare;***
- ***funcție pentru calculul mediei de promovare.***

**Varianta in C++**

```
#include <iostream>
#include<conio.h>
#define N 25
using namespace std;
class student{
        char nume[25];
        char prenume[30];
        int nota[5];
public:
        void id_student();
        void afis();
        double media();};
int n;
void student::id_student(){
        cout<<"\nNume student:"; cin>>nume;cout<<"Prenume student:"; cin>>prenume;
        for(int k=1;k<=5;k++){ cout<<"Nota"<<k<<":";  cin>>nota[k];}}
void student::afis(){
        cout<<"\nNume student:";cout<<nume;cout<<"\nPrenume student:";  cout<<prenume<<"\n";
        for(int k=1;k<=5;k++){ cout<<"Nota:"<<k <<":" <<nota[k] <<endl;          }
        cout<<"Media:"<<media() << endl;}
double student::media(){
        double m=.0;
        for(int k=1;k<=5;k++)
                if(nota[k]<5){ cout<<"\n Student restantier"; return 0;}
                else m+=nota[k];  return m/5;}
int main(){
        student s[N]; cout<<"\n Numarul de studenti:"; cin>>n;
        for(int i=1;i<=n;i++){s[i].id_student(); s[i].afis();}
   return 0;}
```

***Rezultate:***

```
 Numarul de studenti:

Nume student:Popa
Prenume student:Petre
Nota1:10
Nota2:9
Nota3:5
Nota4:8
Nota5:7

Nume student:Popa
Prenume student:Petre
Nota:1:10
Nota:2:9
Nota:3:5
Nota:4:8
Nota:5:7
Media:7.8
```

***Aplicație:***

*Să se modifice programul astfel încât să calculeze și afișeze nr. studenților care au media peste 8.50 și numele acestora*

***Ex.3. Programul C++ implementează tipul de date complex utilizând clase și realizează calculul sumei a n numere complexe.***

**Varianta in C++**

```
#include <iostream>
using namespace std;
int n;
class complex{
        double re[20],im[20];
public:
        double cit();
        void afis();
        double suma_1();
        double suma_2();};
double complex::cit(){
        cout<<"n="; cin>>n;
        for(int i=1;i<=n;i++){
                cout<<"Partea reala a numarului "; cout<<"complex"<<" "<<i <<": "; cin>>re[i];
                cout<<"Partea imaginara a numarului "; cout<<"complex"<<" "<<i << ": "; cin>>im[i]; }
        return 1;}
double complex::suma_1(){
        double s1=0;    for(int i=1;i<=n;i++) s1+=re[i];        return s1;}
double complex::suma_2(){
        double s2=0;    for(int i=1;i<=n;i++)s2+=im[i];         return s2;}
void complex::afis(){
        cout<<"\n Suma celor"<<" "<<n <<" numere complexe:";
        if(suma_2()>0)  cout<<suma_1()<<"+"<<suma_2()<<"*i";
        else cout<<suma_1()<<suma_2()<<"*i" << endl;}
int main(){complex c;
        c.cit();  c.afis();  cout<<endl;return 0;}
```

***Rezultate:***

```
n=2
Partea reala a numarului complex 1: 5.5
Partea imaginara a numarului complex 1: 2.3
Partea reala a numarului complex 2: 1.5
Partea imaginara a numarului complex 2: 4.8

 Suma celor 2 numere complexe:7+7.1*i
```

***Aplicație:***

*Să se modifice programul astfel încât să calculeze și afișeze modulul fiecărui număr complex (valoarea absolută)*

## PROBLEME PROPUSE

1. ***Să se scrie un program în C++ care realizează un program de gestiune a unui magazin de echipamente electronice, pentru care se introduc de la tastatură datele: denumirea, cod, cantitate și preț și se cere ordonarea produselor după fiecare din aceste date.***

2. ***Să se implementeze o clasă numită Vector utilă pentru operaţiile cu tablourile de numere întregi, având ca date membru un tablou de numere întregi (int elem[20];) şi numărul efectiv de elemente ale acestuia (int n;). Să se includă în clasă metode pentru citirea unui vector de la tastatură, înmulţirea tuturor elementelor cu un scalar, adăugarea unui nou element în vector, eliminarea unui element din vector şi afişarea respectivului vector. În exteriorul clasei, să se definească o funcţie de adunare a două elemente de tip Vector. Să se scrie un program care să utilizeze diferite obiecte din această clasă.***

# Capitolul 13
# Aplicații WINDOWS în C# realizate cu Visual Studio. Partea I

În acest capitol este prezentat modul în care se utilizează mediul Microsoft Visual Studio, care include un set complet de instrumente de dezvoltare, pentru generarea de aplicații ASP.NET, Servicii Web XML, aplicații desktop și aplicații pentru dispozitive mobile.

## CONSIDERAŢII TEORETICE

Visual Basic, Visual C++, Visual C# și Visual J# folosesc același mediu de dezvoltare integrat (IDE) care le permite partajarea instrumentelor și facilitează crearea de soluții folosind mai multe limbaje de programare. Aceste limbaje şi caracteristicile .NET Framework oferă acces la tehnologii cheie care simplifica dezvoltarea de aplicații web ASP și XML Web Services prin Visual Web Developer.

C# este un limbaj de programare conceput de Microsoft la inceputul anilor 2000 care a fost conceput ca un concurent pentru Java ambele fiind derivate din limbajele de programare C/C++. Creatorii acestui limbaj au fost o echipa de la firma Microsoft condusa de Anders Hejlsberg si nu este destinat doar platformelor Microsoft ci si alteor sisteme de operare cum sr fi Linux sau Macintosh. Creat ca instrument de dezvoltare pentru arhitectura .NET, limbajul ofera o modalitate facila si eficienta de a scrie programe pentru sistemul Windows,aplicatii pentru internet, componente software etc.

Inainte de explozia Internetului, majoritatea programelor erau compilate si destinate utilizarii pe un anumit procesor si sub un anumit sistem de operare. Programele scrise in C si C++ se compilau intodeauna pana la cod masina executabil. Codul masina este legat de un anumit procesor si de un anumit sistem de operare. Dupa aparitia Internetului insa, la care sunt conectate sisteme cu procesoare si sisteme de operare diferite, problema portabilitatii a devenit foarte importanta. Limbajul Java a realizat portabilitatea prin transformarea codului sursa al programului intr-un cod intermediar numit bytecode. Acest format intermediar este executat apoi de asa numita Masina Virtuala Java (MVJ).

Desi Java a rezolvat cu succes problema portabilitatii, exista unele facilitati care ii lipsesc. Una dintre acestea este interoperabilitatea limbajelor diferite, sau programarea in limbaj mixt (capacitatea codului scris intr-un limbaj de a relationa cu codul scris in alt limbaj). Interoperabilitatea limbajelor diferite este esentiala la realizarea sistemelor software de dimensiuni mari. Ca parte a ansamblului strategiei .NET, dezvoltata de Microsoft, la finele anilor ’90 a fost creat limbajul C#. C# este direct inrudit cu C, C++ si Java. “Bunicul” limbajului C# este C-ul. De la C, C# mosteneste sintaxa, multe din cuvintele cheie si operatorii. De asemenea, C# este bazat pe modelul de obiecte definit in C++.

C# are o legatura deosebita cu mediul sau de rulare, arhitectura .NET. Pe de o parte, C# a fost dezvoltat pentru crearea codului pentru arhitectura .NET, iar pe de alta parte bibliotecile utilizate de C# sunt cele ale arhitecturii .NET. Arhitectura .NET defineste un mediu de programare care permite dezvoltarea si executia aplicatiilor indiferent de platforma. Aceasta permite programarea in limbaj mixt si ofera facilitati de securitate si portabilitate a programelor. Este disponibila deocamdata pentru platformele Windows.

Legat de C#, arhitectura .NET defineste doua entitati importante si anume biblioteca de clase (Fig. 1). NET si motorul comun de programare sau Common Language Runtime (CLR). C# nu are o biblioteca de clase proprie ci utilizeaza direct biblioteca de clase .NET.

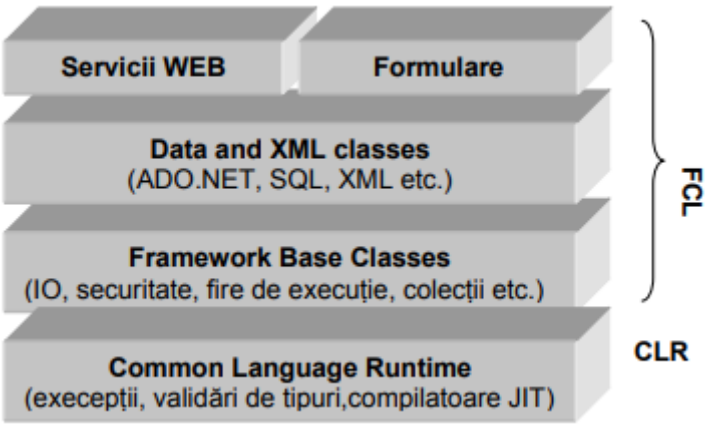



**Fig.1.** Arhitectura .NET Framework

Motorul comun de programare (CLR) se ocupa de executia programelor C#. Acesta asigura de asemenea programarea in limbaj mixt, securitatea si portabilitatea programelor. Atunci cand este compilat un program C#, sau un program in limbaj mixt, rezultatul compilarii nu este un cod executabil. In locul acestuia, se produce un fisier care contine un tip de pseudocod numit limbaj intermediar sau pe scurt IL (Intermediate Language). Acest fisier IL poate fi copiat in orice calculator care dispune de .NET CLR. Prin intermediul unui compilator denumit JIT (Just In Time), motorul comun de pogramare transforma codul intermediar in cod executabil.

Un program scris într-unul dintre limbajele .NET conform Common Language Specification (CLS) este compilat în Microsoft Intermediate Language (MSIL sau IL). Codul astfel obţinut are extensia "exe", dar nu este direct executabil, ci respectă formatul unic MSIL.

CLR include o maşină virtuală asemănătoare cu o maşină Java, ce execută instrucţiunile IL rezultate în urma compilării. Maşina foloseşte un compilator special JIT (Just In Time). Compilatorul JIT analizează codul IL corespunzător apelului unei metode şi produce codul maşină adecvat şi eficient. El recunoaşte secvenţele de cod pentru care s-a obţinut deja codul maşină adecvat, permiţând reutilizarea acestuia fără recompilare, ceea ce face ca, pe parcursul rulării, aplicaţiile .NET să fie din ce în ce mai rapide.

Faptul că programul IL produs de diferitele limbaje este foarte asemănător are ca rezultat interoperabilitatea între aceste limbaje. Astfel, clasele şi obiectele create într-un limbaj specific .NET pot fi utilizate cu succes în altul.

Exista nenumarate tutoriale care prezinta in detaliu tipurile de date si elementele de sintaxa, ale limbajului C# (ex. https://www.w3schools.com/cs/ ). O aplicaţie C# este formată din una sau mai multe clase, grupate în spaţii de nume (namespaces). Este obligatoriu ca doar una din aceste clase să conţină un „punct de intrare” (entry point), şi anume metoda (funcţia) Main.

Clasa (class), în termeni simplificaţi, reprezintă principalul element structural şi de organizare în limbajele orientate spre obiecte, grupând date cât şi funcţii care prelucrează respectivele date. Din raţiuni practice, programele mari, sunt divizate în module, dezvoltate separat, de mai multe persoane. Din acest motiv, există posibilitatea sa apara identificatori cu acelaşi nume. Pentru a evita astfel de erori, în 1955 limbajul C++ introduce noţiunea şi cuvântul cheie namespace. Fiecare mulţime de definiţii dintr-o librărie sau program este grupată într-un spaţiu de nume, existând astfel

posibilitatea de a avea într-un program definiţii cu nume identic, dar situate în alte spaţii de nume. În cazul în care, într-o aplicaţie, unele clase sunt deja definite, ele se pot folosi importând spaţiile de nume care conţin definiţiile acestora. De asemenea un spaţiu de nume poate conţine mai multe spaţii de nume.

**Cuvintele cheie** din limbajul C# sunt prezentate in tabelul de mai jos:

| | | | | |
|---|---|---|---|---|
| abstract | as | base | bool | break |
| byte | case | catch | char | checked |
| class | const | continue | decimal | default |
| delegate | do | double | else | enum |
| event | explicit | extern | false | finally |
| fixed | float | for | foreach | goto |
| if | implicit | in | int | interface |
| internal | is | lock | long | namespace |
| new | null | object | operator | out |
| override | params | private | protected | public |
| readonly | ref | return | sbyte | sealed |
| short | sizeof | stackalloc | static | string |
| struct | switch | this | throw | true |
| try | typeof | uint | ulong | unchecked |
| unsafe | ushort | using | virtual | void |
| volatile | while | | | |

Pentru a da semnificaţii specifice codului, în C# s-au definit şi **cuvintele cheie contextuale**:

| | | | | |
|---|---|---|---|---|
| ascending | by | descending | equals | from |
| get | group | into | join | let |
| on | orderby | partial | select | set |
| value | where | yield | | |

În C# există două modalităţi de declarare a constantelor: folosind const sau folosind modificatorul readonly. Variabilele pot să conţină fie o valoare a unui tip elementar, fie o referinţă la un obiect. Limbajul C# este „case sensitive”.

In C# sunt definiţi mai mulţi operatori prezentati in tabelul de mai jos cu ordinea de prioritate a executiei lor in expresii:

| Prioritate | Tip | Operatori |
|---|---|---|
| 0 | Primar | ( ) [ ] f() . x++ x-- new typeof sizeof checked unchecked -> |
| 1 | Unar | + - ! ~ ++x --x (tip) true false & sizeof |
| 2 | Multiplicativ | * / % |
| 3 | Aditiv | + - |
| 4 | De deplasare | << >> |
| 5 | Relațional | < > <= >= is as |
| 6 | De egalitate | == != |
| 7 | AND (SI) logic | & |
| 8 | XOR (SAU exclusiv) logic | ^ |
| 9 | OR (SAU) logic | \| |
| 10 | AND (SI) condițional | && |
| 11 | OR (SAU) condițional | \|\| |
| 12 | Condițional(ternar) | ?: |
| 13 | atribuire simplă<br>atribuire compusă | =<br>*= /= %= += -= ^= &= <<= >>= \|= |

Pentru a avea control asupra modului de afişare a informaţiei numerice, se poate folosi următoarea formă a lui WriteLine():

**WriteLine("sir",var1,var2,…, varn);**

unde „sir” este format din două elemente:

- caracterele afişabile obişnuite conţinute în mesaje
- specificatorii de format ce au forma generală {nr_var,width:fmt}

unde

- nr_var precizează numărul variabilei (parametrului) care trebuie afişată începând cu 0,
- width stabileşte lăţimea câmpului de afişare, iar fmt stabileşte formatul

Ex.

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Exemplu
{ class Program
 {  static void Main(string[] args)
 { int a, b, c = 5;
 a = c++;
 b = ++c;
 Console.WriteLine("a={0} b={1}", a,b);
}
 }
}
```

**Tipuri de date in C#:**

În C# există două categorii de tipuri de date:

- tipuri valoare
  - tipul simplu predefinit: byte, char, int, float etc.
  - tipul enumerare – enum
  - tipul structură - struct
- tipuri referinţă
  - tipul clasă – class
  - tipul interfaţă – interface
  - tipul delegat – delegate
  - tipul tablou - array

Toate tipurile de date sunt derivate din tipul System.Object .Toate tipurile valoare sunt derivate din clasa System.ValueType, derivată la rândul ei din clasa Object (alias pentru System.Object).

**Tipul valoare**

**Tipuri predefinite**

Limbajul C# conţine un set de tipuri predefinite (int, bool etc.) şi permite definirea unor tipuri proprii (enum, struct, class etc.).

| Tip | Descriere | Alias pentru tipul `struct` din spațiul de nume `System` |
|---|---|---|
| `object` | rădăcina oricărui tip | |
| `string` | secvență de caractere Unicode | `System.String` |
| `sbyte` | tip întreg cu semn, pe 8 biți | `System.Sbyte` |
| `short` | tip întreg cu semn, pe 16 biți | `System.Int16` |
| `int` | tip întreg cu semn pe, 32 biți | `System.Int32` |
| `long` | tip întreg cu semn, pe 64 de biți | `System.Int64` |
| `byte` | tip întreg fără semn, pe 8 biți | `System.Byte` |
| `ushort` | tip întreg fără semn, pe 16 biți | `System.Int16` |
| `uint` | tip întreg fără semn, pe 32 biți | `System.Uint32` |
| `ulong` | tip întreg fără semn, pe 64 biți | `System.Uint64` |
| `float` | tip cu virgulă mobilă, simplă precizie, pe 32 biți (8 pentru exponent, 24 pentru mantisă) | `System.Single` |
| `double` | tip cu virgulă mobilă, dublă precizie, pe 64 biți (11 pentru exponent, 53 pentru mantisă) | `System.Double` |
| `bool` | tip boolean | `System.Boolean` |
| `char` | tip caracter din setul Unicode, pe 16 biți | `System.Char` |
| `decimal` | tip zecimal, pe 128 biți (96 pentru mantisă), 28 de cifre semnificative | `System.Decimal` |

Domeniul de valori pentru tipurile numerice sunt prezentete in tabelul de mai jos:

| Tip | Domeniul de valori |
|---|---|
| sbyte | -128; 127 |
| short | -32768; 32767 |
| int | -2147483648; 2147483647 |
| long | -9223372036854775808; 9223372036854775807 |
| byte | 0; 255 |
| ushort | 0; 65535 |
| uint | 0; 4294967295 |
| ulong | 0; 18446744073709551615 |
| float | -3.402823E+38; 3.402823E+38 |
| double | -1.79769313486232E+308; 1.79769313486232E+308 |
| decimal | -79228162514264337593543950335; 79228162514264337593543950335 |

In limbajul C# instructiunile au fost preluare din C, C++ existand astfel instructiuni de atribuire, conditionale (if then else) , instructiuni de ciclare (for, while, do while) Sintaxa instructiunilor scrise in limbajul C# este similara cu cea din limbajele C/C++. O instrucţiune nouăcare nu exista in C/C++ este foreach. Această instrucţiune enumeră elementele dintr-o colecţie, executând o instrucţiune pentru fiecare element. Elementul care se extrage este de tip read-only, neputând fi transmis ca parametru şi nici aplicat un operator care să-i schimbe valoarea.

Ex.:
```
string[] nume = {"Ana", "Ionel", "Maria"} //se declara un tablou de siruri
for(int i=0; i<nume.Length; i++)
  Console.Write("{0} ", nume[i]);
//Acelaşi rezultat se obţine folosind instrucţiunea foreach:
foreach (string student in nume)
  Console.Write("{0} ", student);
```

Instrucţiunea goto poate fi folosită, în C#, pentru efectuarea unor salturi, în instrucţiunea switch :

Ex.:
```
switch (a)
{case 1:
 x = 0;
 y = 0;
 goto case 20;
case 2:
 x = 3;
 y = 1;
 goto default;
case 3:
 x = 5;
 y = 8;
 break;
default:
 x = 1;
 y = 0;
 break;}
```

Limbajul C# tratează tablourile într-o manieră nouă faţă de alte limbaje (Pascal, C/C++). La declararea unui tablou, se creează o instanţă a clasei .NET, System.Array. Compilatorul va traduce operaţiile asupra tablourilor, apelând metode ale System.Array.

Declararea unui **tablou unidimensional** se face astfel:

**Tip[] nume;**

Prin această declaraţie nu se alocă şi spaţiu pentru memorare. Pentru aceasta, tabloul trebuie instanţiat:

**nume = new Tip[NumarElemente];**

Se pot face în acelaşi timp operaţiile de declarare, instanţiere şi iniţializare:

Ex.:
int[] v = new int[] {1,2,3);

În cazul **tablourilor multidimensionale** se face distincţie între tablouri regulate şi tablouri neregulate (tablouri de tablouri) Declararea în cazul tablourilor regulate bidimensionale se face astfel:

**Tip[,] nume;**

iar instanţierea se face astfel:

**nume = new Tip[Linii,Coloane];**

Accesul la elementele tabloului

**nume[indice1,indice2]**

Ex.: Declararea, instanţierea şi iniţializarea
int[,] mat = new int[,] {{1,2,3},{4,5,6},{7,8,9}};
//sau:
int[,] mat = {{1,2,3},{4,5,6},{7,8,9}};

**Şiruri de caractere**

Pentru reprezentarea şirurilor de caractere, în limbajul C#, tipul de date utilizat este clasa System.String (sau aliasul string). Se definesc două tipuri de şiruri:

- regulate
- de tip „Verbatim”

Tipul regulat conţine între ghilimele zero sau mai multe caractere, inclusiv secvenţe escape.

Limbajul C# introduce, pe lângă şirurile regulate şi cele de tip verbatim. În cazul în care se folosesc mai multe secvenţe escape, se vot utiliza şirurile verbatim. Aceste şiruri se folosesc în special în cazul în care se fac referiri la fişiere, la prelucrarea lor, la regiştri. Un astfel de şir începe cu simbolul „@” înaintea ghilimelelor de început.

**Concatenarea şirurilor de caractere** se face cu operatorul “+”
Ex.:
```
string a = "Invat " + "limbajul " + "C#";
//sirul a contine = "Invat limbajul C#"
```

**Funcţii importante pentru şiruri:**

Clasa String pune la dispoziţia utilizatorului mai multe metode şi proprietăţi care permit prelucrarea şirurilor de caractere. Dintre acestea cele mai uzuale sunt:

- metode de comparare:
  - Compare
  - CompareOrdinal
  - CompareTo
- metode pentru căutare:
  - EndsWith
  - StartsWith
  - IndexOf
  - LastIndexOf
- metode care permit modificarea şirului curent prin obţinerea unui nou şir:
  - Concat
  - CopyTo
  - Insert
  - Join
  - PadLeft
  - PadRight
  - Remove
  - Replace
  - Split
  - Substring
  - ToLower
  - ToUpper
  - Trim
  - TrimEnd
  - TrimStart

Proprietatea Length returnează un întreg care reprezintă lungimea (numărul de caractere) şirului.

Tabelul de mai jos prezintă câteva dintre funcţiile (metodele) clasei String:

| Funcția (metodă a clasei Strig) | Descrierea |
|---|---|
| `string Concat(string u, string v)` | returnează un nou şir obținut prin concatenarea şirurilor `u şi v` |
| `int IndexOf(char c)` | returnează indicele primei apariții a caracterului `c` în şir |
| `int IndexOf(string s)` | returnează indicele primei apariții a subşirului `s` |
| `string Insert(int a, string s)` | returnează un nou şir obținut din cel inițial prin inserarea în şirul inițial, începând cu poziția a, a şirului s |
| `string Remove(int a, int b)` | returnează un nou şir obținut din cel inițial prin eliminarea, începând cu poziția a, pe o lungime de b caractere |
| `string Replace(string u, string v)` | returnează un nou şir obținut din cel inițial prin prin înlocuirea subşirului u cu şirul v |
| `string Split(char[] c)` | împarte un şir în funcție de delimitatorii `c` |
| `string Substring(int index)` | returnează un nou şir care este un subşir al şirului ini ial începând cu indicele index |
| `string Substring(int a, int b)` | returnează un nou şir care este un subşir al şirului inițial, începând de pe poziția a, pe lungimea b caractere |
| `string ToLower()` | returnează un nou şir obținut din cel inițial prin convertirea tuturor caracterelor la minuscule |
| `string ToUpper()` | returnează un nou şir obținut din cel inițial prin convertirea tuturor caracterelor la majuscule |
| `string Trim()` | returnează un nou şir obținut din cel inițial prin ştergerea spațiilor goale de la începutul şi sfârşitul şirului ini ial |
| `string TrimEnd()` | returnează un nou şir obținut din cel inițial prin ştergerea spațiilor goale de la sfârşitul şirului ini ial |
| `string TrimStart()` | returnează un nou şir obținut din cel inițial prin ştergerea spațiilor goale de la începutul şirului ini ial |

**Fisiere in C#**

Pentru accesarea unui fişier de pe disc se folosesc funcţii din spaţiul de nume System.IO. În acest spaţiu există mai multe clase: File, StreamWriter, BinaryReader şi BinaryWriter. Aceste clase sunt folosite pentru operaţiile de intrare-ieşire cu fişiere. Obiectul File este o reprezentare a unui fişier de pe disc, iar pentru a-l utiliza trebuie să îl conectăm la un flux (stream).

Pentru a scrie datele pe disc, se ataşează unui flux un obiect File. Astfel se face administrarea datelor. Limbajul C# (similar cu C/C++) realizeaza operatii cu două tipuri de fişiere: fişiere text şi fişiere binare.

**Scrierea şi citirea datelor din fişiere text si binare**

Fişierele de ieşire necesită utilizarea unui obiect StreamWriter. Funcţia CreateText(), ce face parte din clasa File, deschide un fişier şi creează obiectul StreamWriter. Dacă la fişierele text tipul de flux folosit era StreamWriter, la cele binare, pentru scrierea datelor programul creează un obiect FileStream, la care trebuie ataşat şi un obiect BinaryWriter.

**Programarea vizuală**

Visual Studio și C# sunt deosebit de utile pentru a crea interfețe/formulare de tip Windows Forms sau aplicații Web. O aplicație de tip Windows Form este concepută pentru a fi compilată și executată pe un calculator desktop , nu într-un browser web.

Problemele rezolvate incluse in acest capitol prezinta modul in care se crează aplicații de tip desktop utilizând Windows Forms. O aplicație de tip Windows Form poate fi compilată de pe un computer și poate integra o colecție de etichete, casete pentru text, casete pentru liste și multe alte tipuri de elemente.

O aplicaţie vizuală dispune de o interfaţă grafică sugestivă şi pune la dispoziţia utilizatorului instrumente specifice de utilizare (drag, clic, hint etc.)

O aplicaţie Windows conţine cel puţin o fereastră (Form) în care se poate crea o interfaţă cu utilizatorul aplicaţiei. Componentele vizuale ale aplicaţiei pot fi prelucrate în modul Designer (Shift+F7) pentru a plasa noi obiecte, a le stabili proprietăţile etc. Codul „din spatele" unei componente vizuale este accesibil în modul Code (F7).

Compilarea modulelor aplicaţiei şi asamblarea lor într-un singur fişier „executabil" se realizează cu ajutorul opţiunilor din meniul Build, uzuală fiind Build Solution (F6). Proiectarea vizuală a interfetei formularului se poate face inserând controale selectate din fereastra de instrumente (Toolbox) şi setând proprietăţile acestora.

**Fereastra Properties**

In Toolbox există toate tipurile de controale care sunt necesare pentru realizarea unei aplicaţii. Cele mai multe controale sunt obiecte de clase derivate din clasa System.Windows.Forms.Control. Datorită acestui fapt multe dintre proprietăţile şi evenimentele diverselor controale vor fi identice.

Fereastra Properties, din interfaţa mediului de programare, conţine atât proprietăţile cât şi evenimentele ataşate controalelor. Proprietăţile controalelor, sunt moştenite sau supraînscrise din clasa de bază Control.

Tabelul de mai jos prezintă proprietăţile comune controalelor, proprietăţi furnizate de către clasa Control:

| Proprietatea | Descrierea proprietății |
|---|---|
| Anchor | se referă la posibilitatea de a ancora controlul față de o margine (sau toate) |
| BackColor | permite stabilirea culorii de fundal a controlului |
| Bottom | permite stabilirea distanței dintre marginea de sus a ferestrei şi control |
| Dock | atașează controlul la una dintre marginile ferestrei |
| Enabled | permite controlului să recepționeze evenimente de la utilizator |
| ForeColor | permite stabilirea culorii textului |
| Height | permite definirea înălțimii controlului |
| Left | permite stabilirea distanței dintre marginea din stânga a ferestrei şi marginea stânga a controlului |
| Name | permite denumirea controlului pentru a-l putea mai uşor vizualiza şi manipula în codul sursă |
| Parent | părintele controlului |
| Right | permite stabilirea distanței dintre marginea din dreapta a ferestrei şi marginea din dreapta a controlului |
| TabIndex | prin numărul de ordine care i se ataşează se stabileşte ordinea activării controlului la apăsarea tastei TAB |
| TabStop | permite sau nu ca respectivul control să fie activat prin apăsarea tastei TAB |
| Tag | se referă la un şir de caractere pe care controlul îl poate stoca în interiorul său |
| Top | permite stabilirea distanței dintre marginea de sus a ferestrei şi marginea de sus a controlului |
| Visible | stabileşte dacă respectivul control, care exită în fereastră, este (TRUE) sau nu vizibil |
| Width | stabileşte lățimea controlului |

Tot clasa Control amintită mai sus, implementează şi o serie de evenimente la care controalele vor reacţiona:

| Evenimentul | Descrierea evenimentului |
|---|---|
| Clic | se generează când se dă clic asupra unui control |
| DoubleClic | se generează când se dă dublu clic asupra unui control. Excepție făcând Button asupra căruia nu se va putea face dublu clic, deoarece controlul acționează la primul clic |
| DragDrop | se generează la finalizarea lui drag and drop |
| DragEnter | se generează atunci când obiectul, printr-un drag and drop, ajunge în interiorul controlului |
| DragLeave | se generează atunci când obiectul, printr-un drag and drop, ajunge să părăsească controlului |
| DragOver | se generează atunci când obiectul, printr-un drag and drop, ajunge deasupra controlului |
| KeyDown | se generează atunci când o tastă este apăsată în timp ce controlul este activ. Se va furniza codul ASCII al tastei apăsate. Se generează înainte de evenimentele KeyPress şi KeyUp |
| KeyPress | se generează atunci când o tastă este apăsată în timp ce controlul este activ. Se va furniza codul de scanare al tastei apăsate. Se generează după KeyDown şi înainte de KeyUp |
| KeyUp | se generează când o tastă este eliberată în timp ce controlul este activ. Se generează după KeyDown şi KeyPress |
| GotFocus | se generează când controlul devine activ (se mai spune: când controlul primeşte input focusul) |
| LostFocus | se generează când controlul devine inactiv (se mai spune: când controlul pierde input focusul) |
| MouseDown | se generează când cursorul mouse-ului este deasupra controlului şi se apasă un buton al mouse-ului |
| MouseMove | se generează când trecem cu mouse-ul deasupra controlului |
| MouseUp | se geerează când mouse-ul este deasupra controlului şi eliberăm un buton al mouse-ului |
| Paint | se generează la desenarea controlului |
| Validated | se generează când un control este pe cale să devină activ. Se generează după terminarea evenimentului Validating, indicând faptul că validarea controlului este completă |
| Validating | se generează când un control este pe cale să devină activ |

**Construirea interfeţei utilizator**

**Ferestre**

Spaţiul Forms oferă clase specializate pentru: creare de ferestre sau formulare (System.Windows.Forms.Form), elemente specifice (controale) cum ar fi butoane (System.Windows.Forms.Button), casete de text (System.Windows.Forms.TextBox) etc. Proiectarea unei ferestre are la bază un cod complex, generat automat pe măsură ce se desemneaza componentele şi comportamentul acesteia.

Acest cod realizează: derivarea unei clase proprii din System.Windows.Forms.Form, clasă care este înzestrată cu o colecţie de controale (iniţial vidă). Constructorul ferestrei realizează instanţieri ale claselor Button, MenuStrip, Timer etc. (orice plasăm noi în fereastră) şi adaugă referinţele acestor obiecte la colecţia de controale ale ferestrei.

Dacă modelul de fereastră reprezintă ferestra principală a aplicaţiei, atunci ea este instanţiată automat în programul principal (metoda Main). Dacă nu, trebuie scris codul care realizează

instanţierea.

Clasele derivate din Form moştenesc o serie de proprietăţi care determină atributele vizuale ale ferestrei (stilul marginilor, culoare de fundal, etc.), metode care implementează anumite comportamente (Show, Hide, Focus etc.) şi o serie de metode specifice (handlere) de tratare a evenimentelor (Load, Click etc.).

O fereastră poate fi activată cu form.Show() sau cu form.ShowDialog(), metoda a doua permiţând ca revenirea în fereastra din care a fost activat noul formular să se facă numai după ce noul formular a fost închis (spunem că formularul nou este deschis modal).

In tabelul de mai jos este prezentata o listă cu controalele cele mai uzuale si descrierea lor:

| **Funcția controlului** | **Numele controlului** | **Descriere** |
|---|---|---|
| buton | **Button** | Sunt folosite pentru a executa o secvență de instrucțiuni în momentul activării lor de către utilizator |
| calendar | **MonthCalendar** | Afişează implicit un mic calendar al lunii curente. Acesta poate fi derulat şi înainte şi înapoi la celelalte luni calendaristice. |
| casetă de validare | **CheckBox** | Oferă utilizatorului opțiunile : da/nu sau include/exclude |
| etichetă | **Label** | Sunt folosite pentru afişarea etichetelor de text, şi a pentru a eticheta controalele. |
| casetă cu listă | **ListBox** | Afişează o listă de articole din care utilizatorul poate alege. |
| imagine | **PictureBox** | Este folosit pentru adăugarea imaginilor sau a altor resurse de tip bitmap. |
| pointer | **Pointer** | Este utilizat pentru selectarea, mutarea sau redimensionarea unui control. |
| buton radio | **RadioButton** | Este folosit pentru ca utilizatorul să selecteze un singur element dint-un grup de selecții. |
| casetă de text | **TextBox** | Este utilizat pentru afişarea textului generat de o aplicație sau pentru a primi datele introduse de la tastatură de către utilizator. |

Celelelate elemente utilizate in aplicatiile vizuale rezolvate vor fi detaliate in cele ce urmeaza.

## PROBLEME REZOLVATE

**Ex 1:** Implementarea aplicației *Hello World* cu ajutorul Windows Form Application. Etapele de realizare sunt următoarele:

- Primul pas presupune crearea unui nou proiect în Visual Studio. Astfel, după deschiderea Visual Studio se va alege din meniu *New-> Project* sau în versiunile mai noi se va accesa direct *Create a new project* (Fig.1).

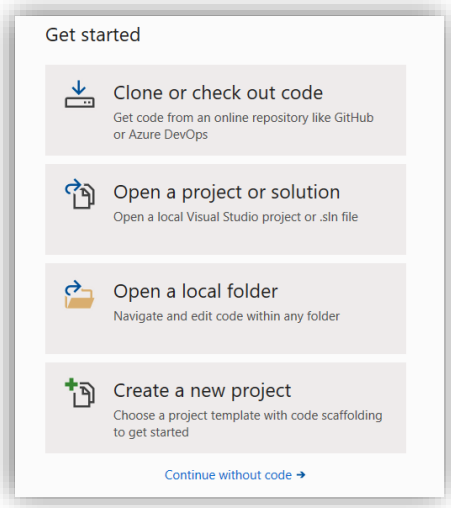



**Fig. 1.** Crearea unui nou proiect Visual Studio

- Următorul pas este alegerea tipului de proiect ca aplicație Windows Form. Se vor preciza de asemenea numele și locația unde se va salva noul proiect. În cazul versiunilor mai noi se va regăsi opțiunea Windows Form App ( .NET Framework) din multitudinea de diferite tipuri de proiecte ce pot fi create și se va da click pe *Next*, după care în următoarea fereastră se va alege denumirea și locația noului proiect (Fig.2).

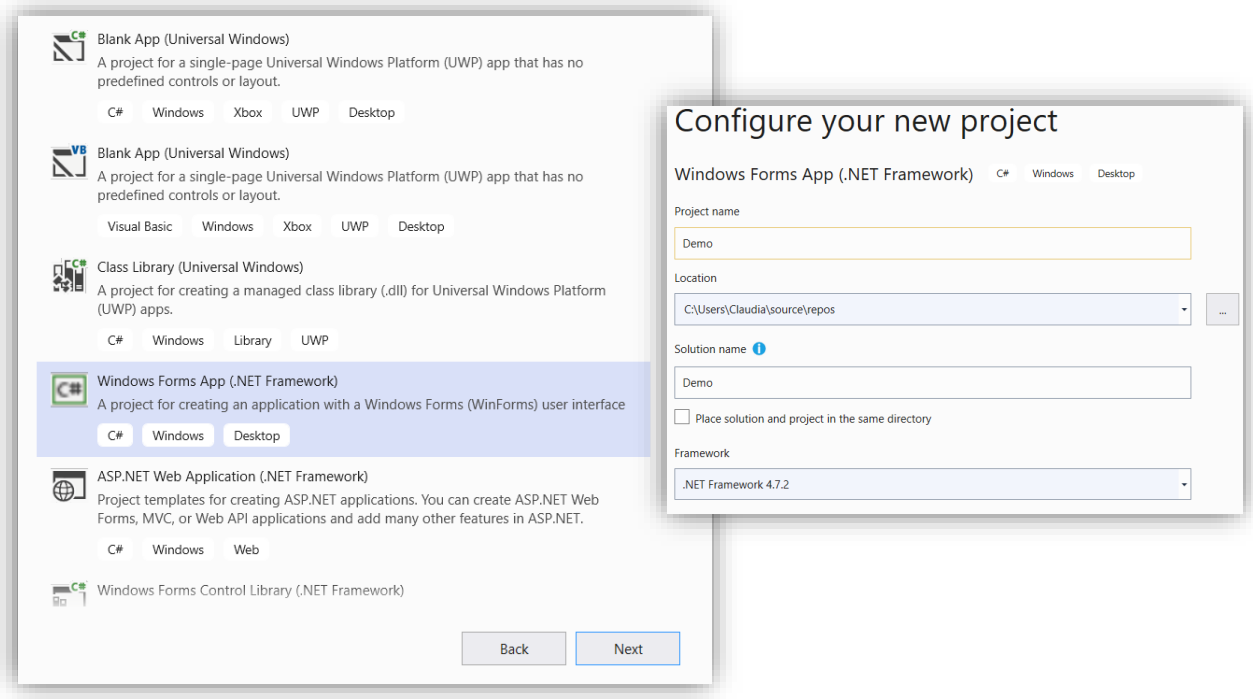



**Fig. 2.** Configurarea noului proiect Visual Studio

În urma execuţiei paşilor precedenți, se va deschide o fereastră conținând un formular (Form Designer).În fereastra Solution Explorer se va găsi proiectul creat cu denumirea Demo conținând 2 fișiere și anume o aplicație form numită ***Form1.cs*** care conține codul pentru aplicația de tip Windows Form și programul principal denumit ***Program.cs*** care este codul implicit creat când o nouă aplicație este definită în Visual Studio.

Acest cod va conține codul de startup pentru întreaga aplicație. De asemenea în interfață se va deschide și un meniu, ***Toolbox***, care integreaza toate butoanele/ elementele ce pot fi adăugate formularelor. În cazul în care Toolbox-ul nu este prezent se apasă combinația de taste Ctrl+Alt+x și acesta va fi afisat pe ecran.

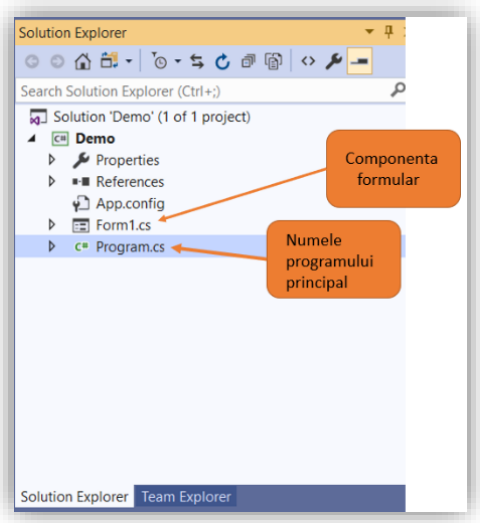



**Fig. 3.** Fereastra configurare Solution Explorer

- Se va adăuga o etichetă (***Label***) formularului deja creat care conține textul *Hello World,* sau orice alt text. Din meniul Toolbox se va alege Label și se va insera în formular cu drag and drop. (Fig.4)

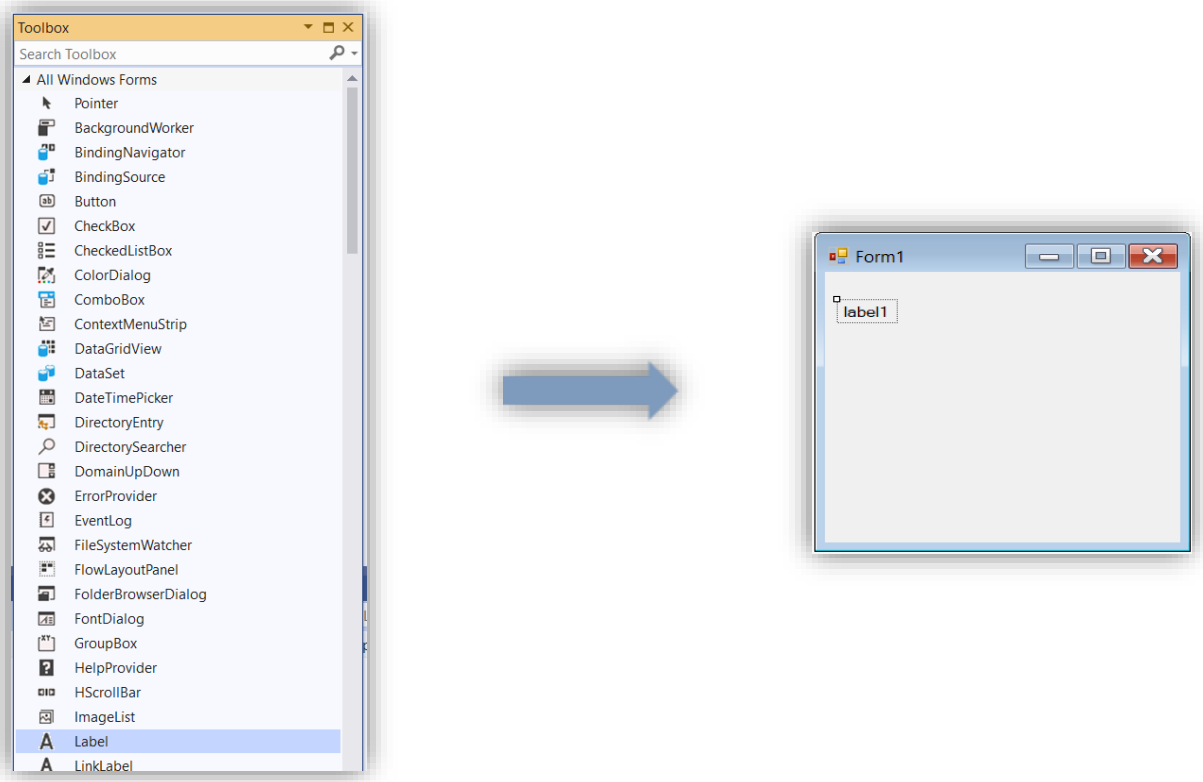



**Fig. 4.** Inserarea unei etichete într-un formular

- Următorul pas este accesarea proprietăților etichetei și modificarea acestora. Astfel, cu click dreapta pe label se alege Properties (proprietățile apar și în partea din dreapta jos a ferestrei din Visual Studio). În cadrul proprietăților se va selecta Text și se va introduce *Hello World* .

- Astfel, formularul va arata ca in Fig. 5 după parcurgerea pașilor de mai sus. După cum se poate observa, codul este cel care este creat la deschiderea unui Windows Form, fără a adăuga un nou cod pentru introducerea label-uri. Următorul pas este accesarea proprietăților etichetei și modificarea acestora. Astfel, cu click dreapta pe label se alege Properties (proprietățile apar și în partea din dreapta jos a ferestrei din Visual Studio). În cadrul proprietăților se va selecta Text și se va introduce *Hello World* .

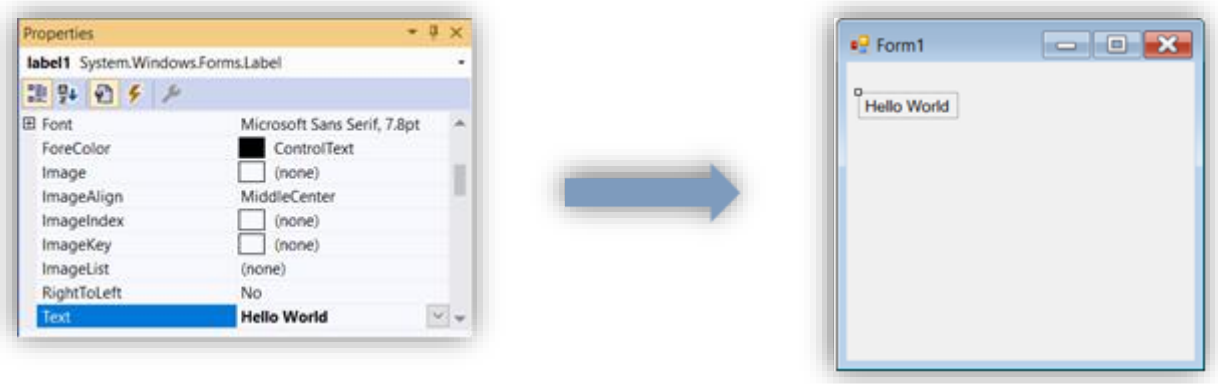
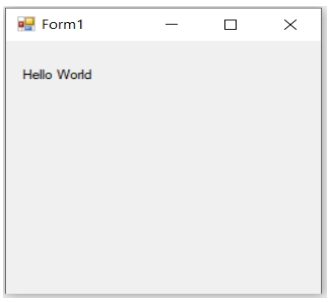



**Fig. 5.** Inserarea mesajului Hello World în label

| **Varianta in C#** |
|---|
| using System;<br>using System.Collections.Generic;<br>using System.ComponentModel;<br>using System.Data;<br>using System.Drawing;<br>using System.Linq;<br>using System.Text;<br>using System.Threading.Tasks;<br>using System.Windows.Forms;<br><br>namespace _Demo<br>{<br>  public partial class Form1 : Form<br>  {<br>  public Form1()<br>  {<br>  InitializeComponent();<br>  }<br>  }<br>} |

***Rezultate:***




***Aplicație:***

Să se modifice programul astfel încât să se declare și alte obiecte de tipul casă și școală și utilizând funcțiile membre să se afișeze diferite valori ale datelor membre.

***Ex. 2: Aplicatia explica implementarea și configurarea unui formular care conține datele unei persoane și o listă de cursuri, cu elemente din Toolbox .***

Se va creea un formular utilizând limbajul C# și Visual Studio, care să includă câmpurile: Nume, Adresa, Oras resedinta, Gen și Curs. Aceste câmpuri se vor creea prin introducerea în formular a unor elemente din ToolBox .

**GroupBox** este un element care grupează anumite componente ale formularului într-o secțiune. Un exemplu de folosire este utilizarea de elemente grupate pentru colectarea datelor personale ale unei persoane (numele și adresa).

Primul pas pentru implementarea acestei aplicații este introducerea cu drag and drop a unui GroupBox în formular. Odată ce Groupbox-ul a fost introdus, din fereastra de proprietăți se va introduce în tabul *Text* textul *Detalii utilizator*. Fereastra de configurare a formularului este prezentată în Fig.6.

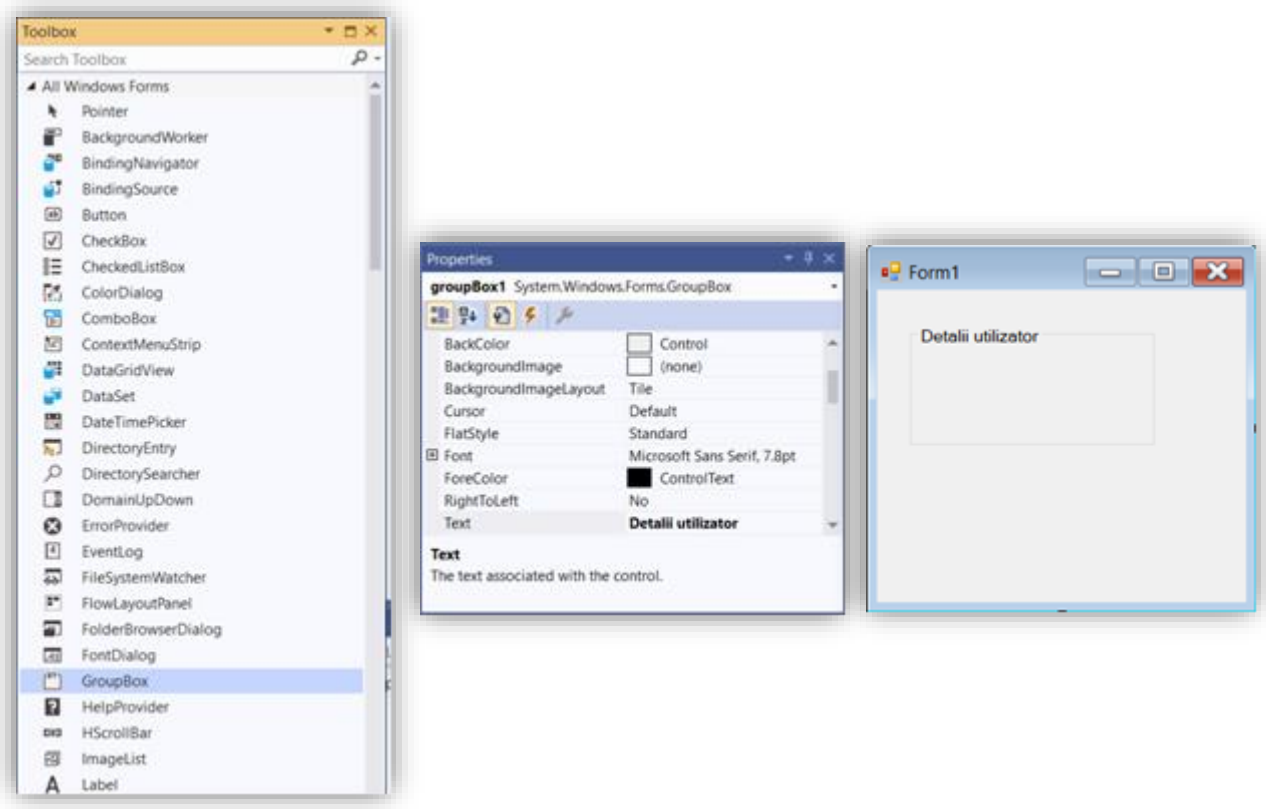



**Fig. 6.** Introducerea GroupBox-ului în formular

**Label**-urile (etichete) sunt utilizate pentru a afișa un text sau un mesaj către utilizator și se folosesc de obicei împreună cu alte elemente. De obicei aceste etichete se utilizează pentru a prezenta informația ce se introduce într-un TextBox.

În cazul acestui formular se vor introduce 2 label-uri în interiorul GroupBox-ului deja creat, prin metoda drag and drop. Se vor redenumi label-urile N*ume* și *Adresa* în fereastra Properties care apare când utilizatorul este poziționat pe label la secțiunea text. Modul în care va arăta formularul după inserarea etichetelor se poate observa în figura de mai jos.

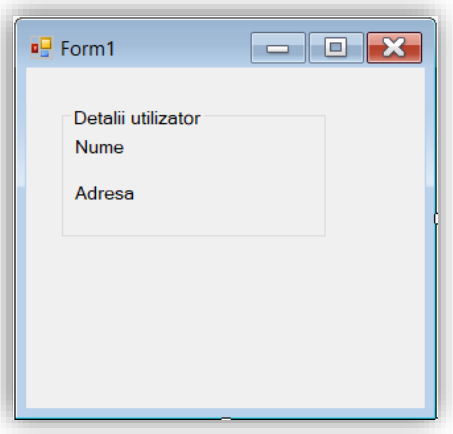



**Fig. 7.** Introducerea label-urilor în GroupBox

Un **TextBox** este folosit pentru a introduce un text în aplicația creată. În cadrul formularului se vor adăuga 2 căsuțe de tip TextBox, cu drag and drop, după care se va redimensiona GroupBox-ul pentru a include etichetele si textbox-urile adaugate. Odată introduse, se va modifica numele acestora în *numetext* și *adresatext* din fereastra de proprietăți a fiecăreia. Este utilă denumirea acestora deoarece este mai ușor de adăugat ulterior o altă funcționalitate pentru aceste elemente de control.

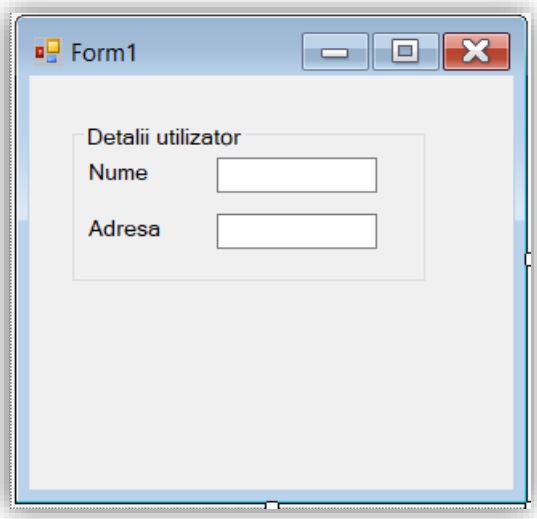



**Fig. 8.** Introducerea TextBox-urilor

**ListBox-ul** este o componentă din Toolbox utilă pentru afișarea unei liste de elemente într-un formular. O astfel de listă predefinită poate fi utilizată în cadrul formularului pentru a afișa o listă de orașe de reședință predefinită. Pentru acesta se adaugă, cu drag and drop, din *ToolBox* un *ListBox* în formular, după care se va selecta această componentă pentru a accesa fereastra de proprietăți.

Primul pas va fi modificarea denumirii ListBox-ului în opțiunile de *Design*, *Name*. În cazul opțiunilor pentru *Data* se regăsește proprietatea *Items*. Dacă se va da click pe această opțiune se deschide o fereastră unde poate fi introdusă lista predefinită (în cazul nostru „Cluj”, „Timisoara”, „Iasi”, „Bucuresti”), după care se apasă OK (Fig. 9, Fig.10).

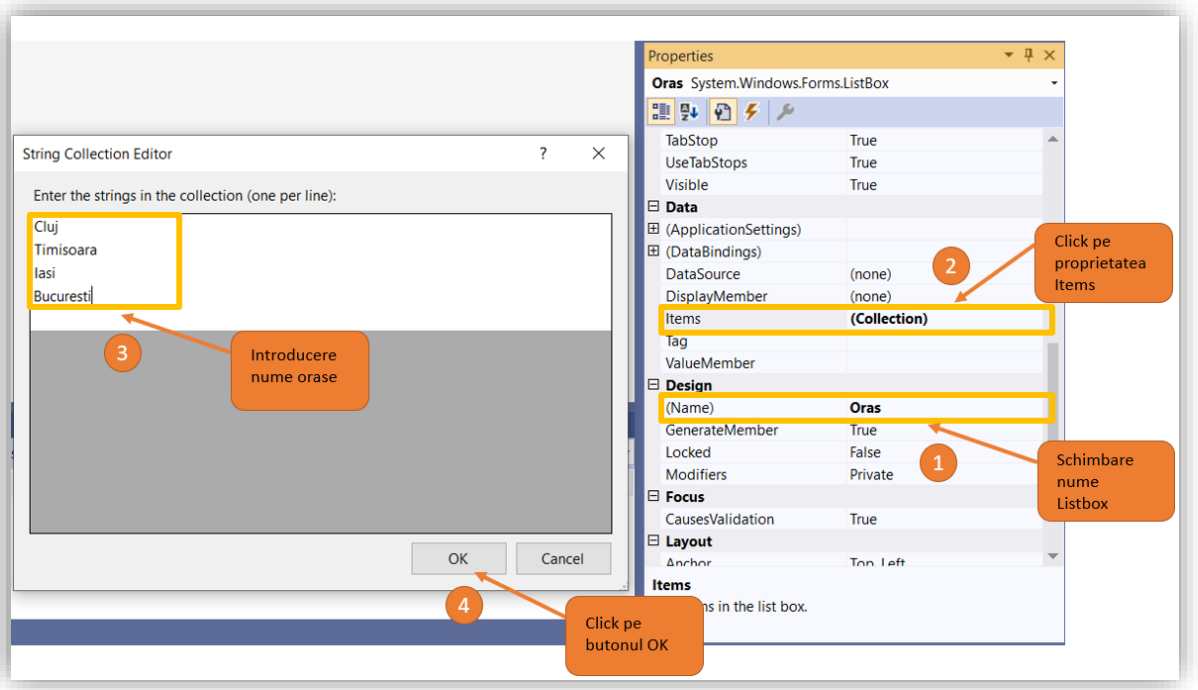



**Fig. 9.** Pașii necesari pentru implementarea ListBox-ului

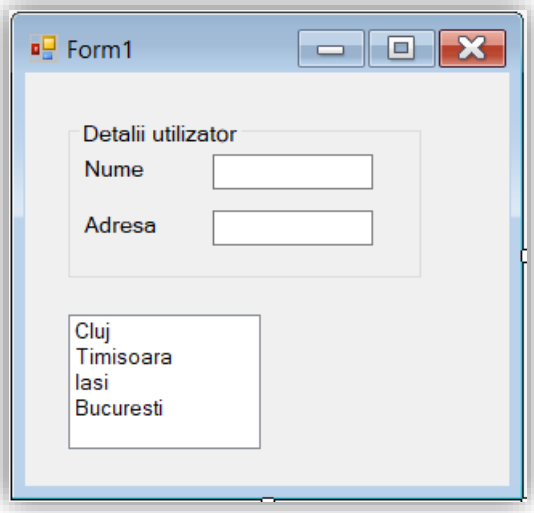



**Fig. 10.** Interfața formularului după parcurgerea pașilor anteriori

Un element (radio) de tip **RadioButton** este util pentru a crea o listă de elemente din care doar unul poate fi selectat. În formular se vor crea butoane radio astfel încât utilizatorul să poată selecta genul. Astfel se introduc 2 butoane radio , cu drag and drop și se vor modifica proprietățile acestora din fereastra *Properties,* schimbând textul lor în Masculin, respectiv Feminin.

Din proprietățile de Design se vor da nume acestor butoane radio m și f. Design-ul formularului creat până în acest moment este prezentat în Fig. 11.

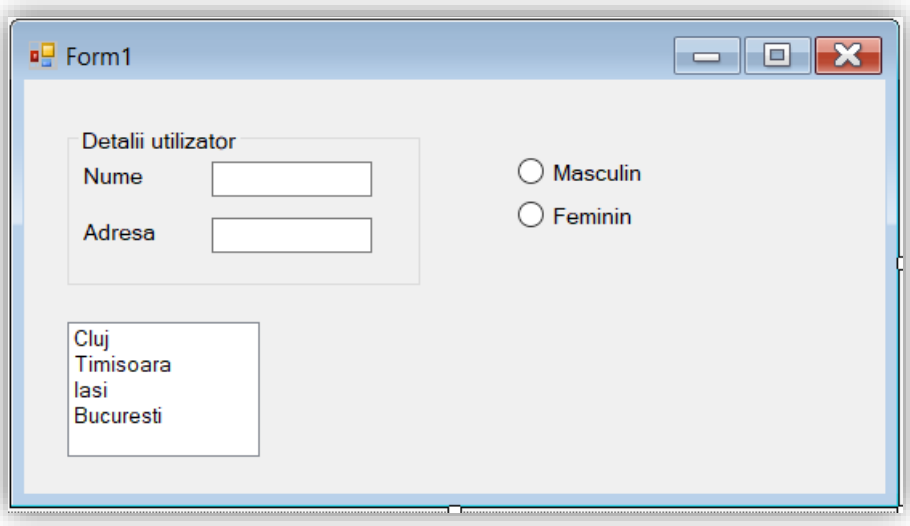



**Fig. 11.** Interfața formularului după inserarea butoanelor radio

Un element **CheckBox** este util pentru implementarea unei liste de opțiuni din care utilizatorul poate face o alegere multiplă. În cazul aplicației de față se vor introduce 2 comenzi de tip CheckBox prin care utilizatorul poate alege dacă dorește să participe la cursul de C# sau ASP.Net. Pentru aceasta se vor introduce elementele CheckBox cu drag and drop din ToolBox, iar apoi proprietățile acestora se vor modifica din fereastra Properties. Se va modifica textul aferent în *C#* și *ASP.Net*.

În mod similar se va modifica numele butoanelor în *chkC* și *chkASP*. Formularul creat până în acest stadiu se poate observa în Figura 12.

Butoanele (**Button**) sunt utilizate pentru a permite utilizatorului să activeze procesarea formularului. Astfel, în formularul creat se va introduce un buton *Proceseaza* care finalizează introducerea datelor. Se va introduce din ToolBox butonul și se va modifica textul în *Proceseaza* și denumirea în *Butonp*.

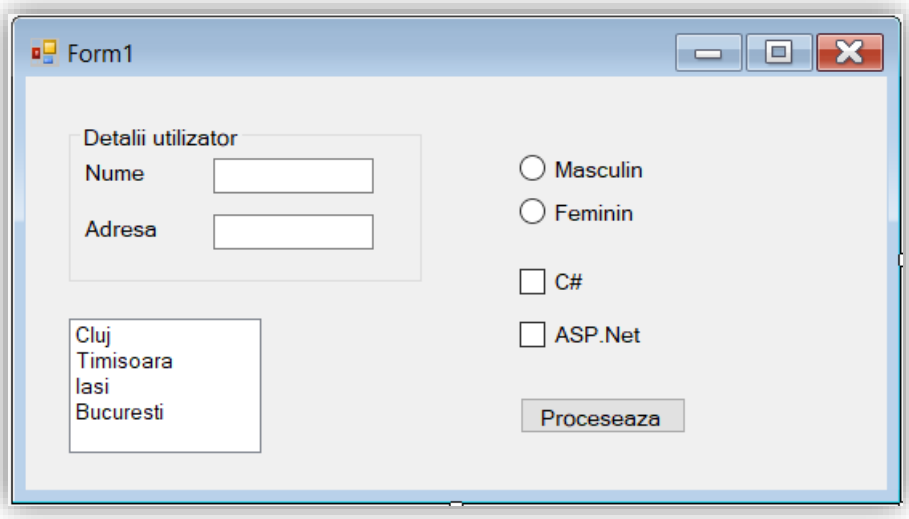



**Fig. 12.** Forma finală a formularului

**Gestionarea evenimentelor pentru elementele din Toolbox:** în formulare, se pot adăuga evenimente care apar la accesarea unui element, de exemplu unele procese sunt pornite la apăsarea unui buton.

Spre exemplu daca se dorește ca la alegerea unuia dintre orașele predefinite să apară un mesaj care să afișeze chiar orașul selectat, se va da dublu click pe ListBox-ul din form design pentru ca Visual Studio să deschidă codul din spatele formularului și să adauge o parte de cod pentru eveniment. Această parte de cod va fi accesată de fiecare dată cand un element din ListBox va fi ales.

```
private void Oras_SelectedIndexChanged(object sender, EventArgs e)
    {
    }
```

Codul care va trebui inserat între acolade este prezentat mai jos, funcția devenind:

```
private void Oras_SelectedIndexChanged(object sender, EventArgs e)
    {  string text = Oras.GetItemText(Oras.SelectedItem);
       MessageBox.Show(text);       }
```

Prima linie de cod definește o variabilă de tip string denumită *text* care primește denumirea orașului selectat. Se observa că *Oras* este denumirea specificata pentru ListBox și *GetItemText* este o funcție care preia denumirea selectată. MessageBox se utilizează și pentru afișarea valorii variabilei text.

După compilare și execuție, la alegerea unui oraș, acesta se va afișa într-un mesaj ca în Fig. 13.

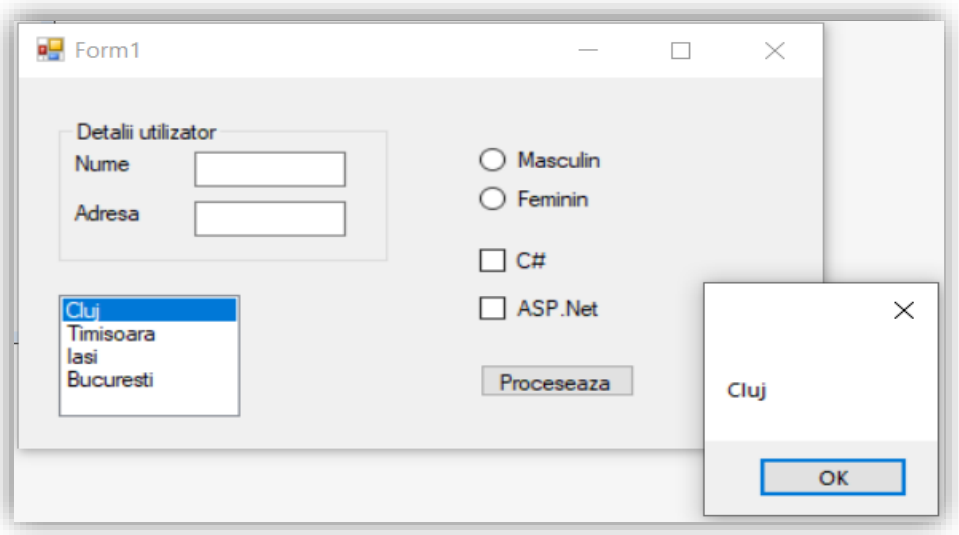



**Fig. 13.** Afișarea unei locație la accesarea ei din ListBox

Pentru a afișa numele și adresa pe rânduri diferite după ce acestea au fost introduse, se selecteaza butonul *Proceseaza* , si cu dublu click pe buton și se va scrie codul:

```
private void Butonp_Click(object sender, EventArgs e)
    {  string nume = textBox1.Text;
       string adresa = textBox2.Text;
       MessageBox.Show(nume+'\n'+adresa); }
```

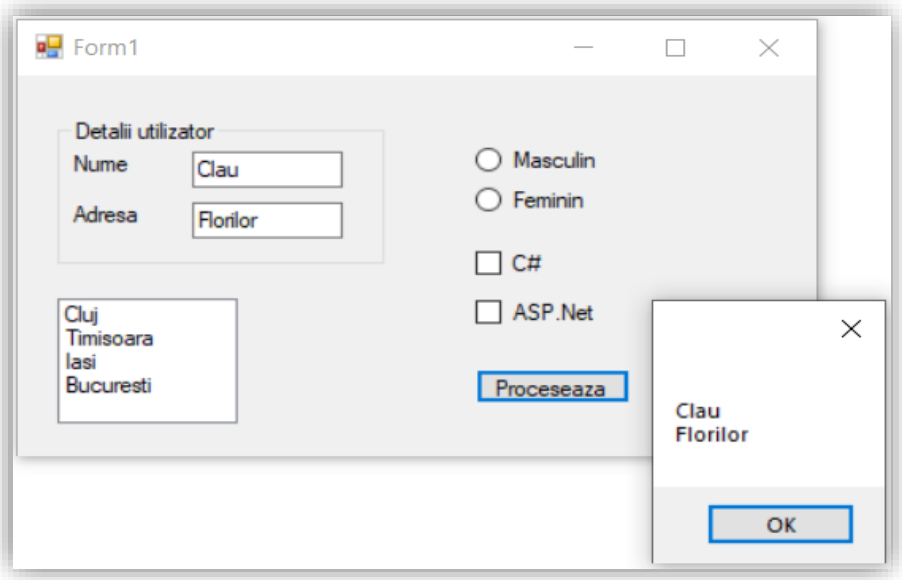



**Fig. 14.** Afișarea numelui și a adresei la apăsarea butonului Proceseaza

Pentru a introduce o imagine se folosește ToolBox-ul din Visual Studio de unde se selectează opțiunea **PictureBox** care se plasează în interiorul formularului cu drag and drop. Se pot adăuga imagini prin alegerea din fereastra Properties a proprietății *Image*. Se dă click pe butonul *Import* din fereastra ce s-a deschis și se va alege o imagine, după care se apasă butonul ok (Fig. 15). Design-ul final al formularului este prezentat în Fig. 16.

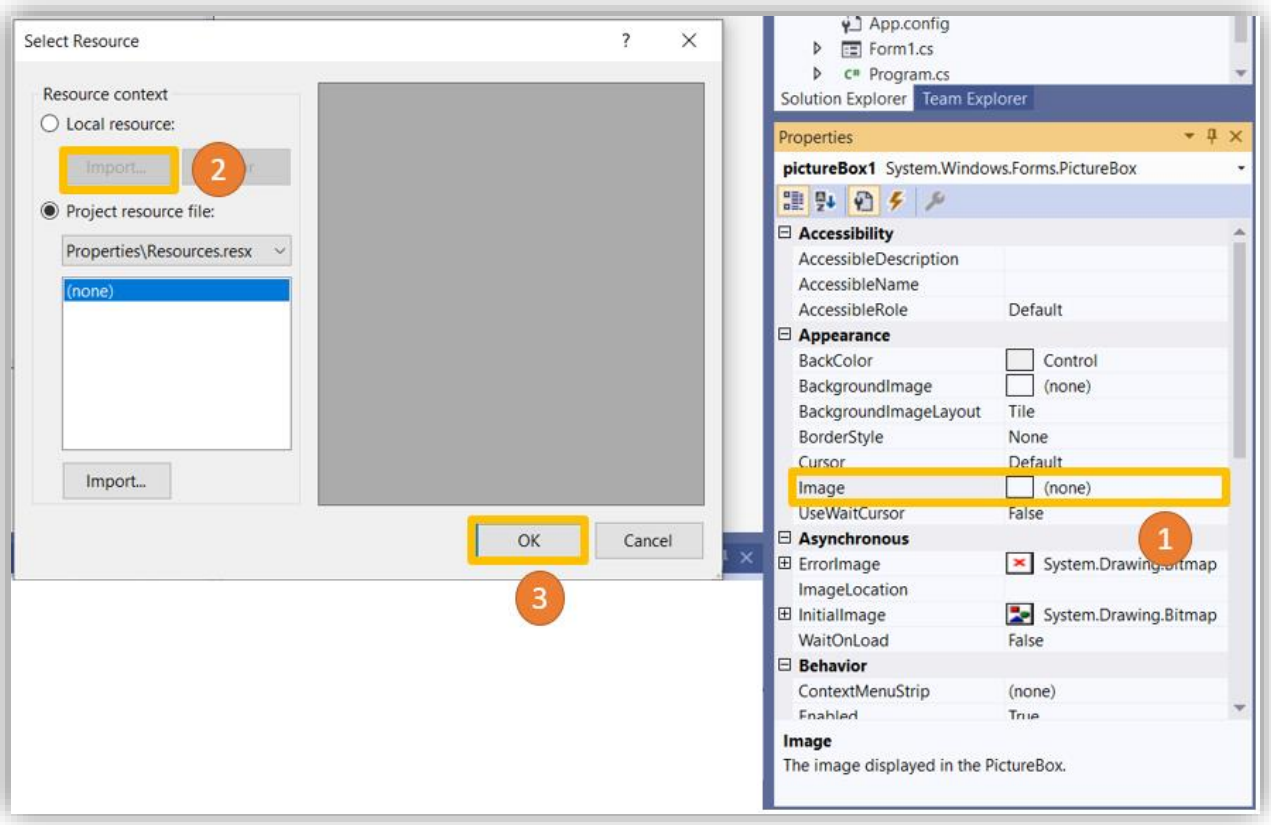



**Fig. 15.** Inserarea unei imagini într-un formular

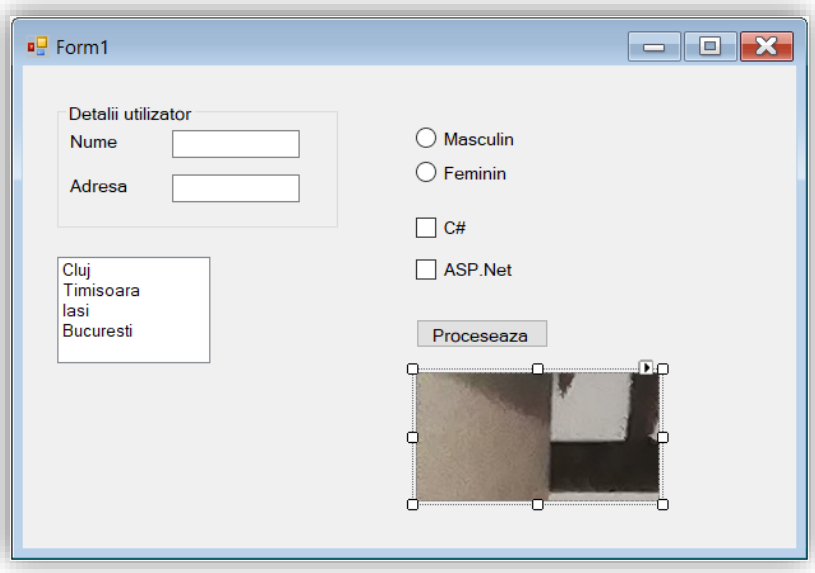
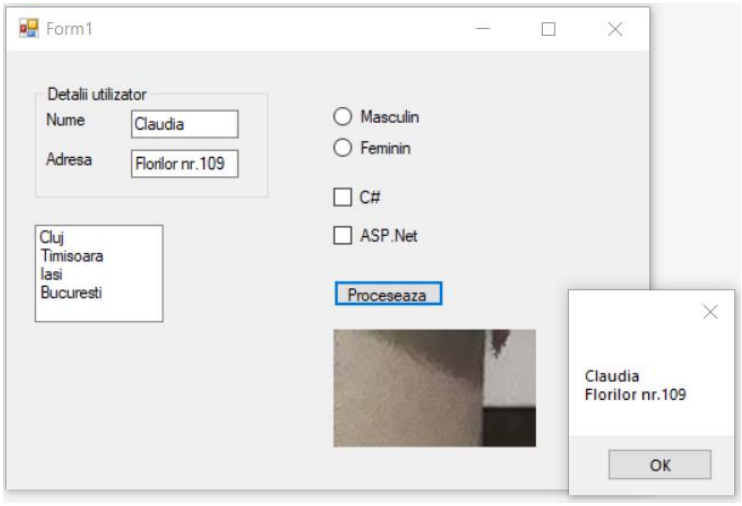



**Fig. 16.** Forma finală a formularului

**Varianta in C#**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Demo
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void Oras_SelectedIndexChanged(object sender, EventArgs e)
      {
         string text = Oras.GetItemText(Oras.SelectedItem);
         MessageBox.Show(text);
      }

      private void Butonp_Click(object sender, EventArgs e)
      {
         string nume = textBox1.Text;
         string adresa = textBox2.Text;
         MessageBox.Show(nume+'\n'+adresa);
      }
   }
}
```

***Rezultate:***




***Aplicație:***

Să se creeze un formular folosind C# pentru un cabinet de proteze auditive unde se vor introduce detalii de utilizator (nume, prenume, varsta, adresa) și un grad de deficiența de auz care duca la afisarea unui tip de proteză auditivă.

***Ex. 3: Adăugarea de elemente grafice formularelor. Această aplicație permite desenarea unor linii într-un formular***

Pentru a putea introduce elemente grafice într-un formular, este necesară crearea unui obiect de tip *Graphics* folosind metoda **CreateGraphics()**. Pentru a desena în formularul se poate folosi codul:

```
System.Drawing.Graphics gr = this.CreateGraphics();
```

Pentru a desena într-un **PictureBox** se utilizează:

```
System.Drawing.Graphics gr = PictureBox1.CreateGraphics();
```

De asemenea se poate folosi ca suprafață de desen un TextBox scriind codul:

```
System.Drawing.Graphics gr = TextBox1.CreateGraphics();
```

Obiectul Graphics care este creat nu desenează nimic până cand este apelată metoda obiectului Graphics. Pe lângă aceasta, un obiect de tip *Pen* trebuie creat ca element de desenare. Codul pentru crearea acestuia este prezentat în cele ce urmează.Un creion (Pen) se crează cu codul:

```
myPen = New Pen(Brushes.Color, LineWidth);
```

unde myPen este o variabilă de tip *Pen*. Primul argument al obiectului definește culoarea liniei de desenare, în timp ce cel de al doilea argument reprezintă lățimea liniei. De exemplu, codul următor va crea un creion de culoare *Blue* iar lățimea liniei va fi 10 pixeli:

```
System.Drawing.Pen albastru;
albastru = new System.Drawing.Pen(Drawing. Color.Blue, 20);
```

sau

```
albastru = new System.Drawing.Pen(Brushes.Blue, 20);
```

Implementarea aplicației începe prin crearea unui proiect de tip Windows Form App ( .NET Framework). Se introduce în formular un buton care se denumește *Desenează*. Codul aferent evenimentului ce se efectuează în urma apăsării butonului este:

```
        private void Button1_Click(object sender, EventArgs e)
            {  System.Drawing.Graphics gr = this.CreateGraphics();
               System.Drawing.Pen portocaliu;
               portocaliu = new System.Drawing.Pen(Brushes.OrangeRed, 25);
               gr.DrawLine(portocaliu, 60, 100, 150, 50);       }
```

Prima linie de cod din funcție crează obiectul de tip Graphics iar cea de a 2-a creionul cu denumirea *portocaliu*. A treia linie de cod nu face altceva decât să definească caracteristicile creionului și anume culoarea și dimensiunea acestuia.

Folosind instrucțiunea *DrawLine* se desenează linia. Primul argument reprezintă variabila de tip *Pen* și anume portocaliu, al doilea și al treilea argument definesc coordonatele punctului de start al liniei, iar ultimele 2 argumente definesc punctul final al liniei. Sintaxa instrucțiunii *Drawline* este:

```
        object.DrawLine(Pen, x1, y1, x2, y2)
```

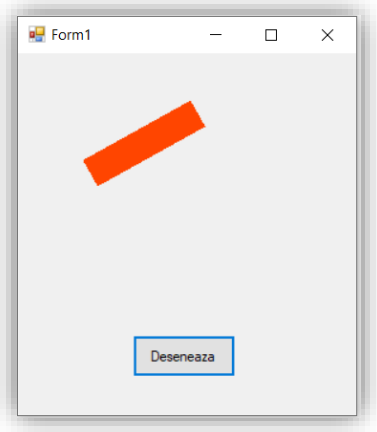



**Fig. 17.** Formularul cu o linie desenată

Forma finală a aplicației de tip formular care include elemente grafice este prezentată în Fig. 17.

**Varianta in C#**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace grafic
{   public partial class Form1 : Form
   {      public Form1()
      {         InitializeComponent(); }
      private void Button1_Click(object sender, EventArgs e)
      {
         System.Drawing.Graphics gr = this.CreateGraphics();
         System.Drawing.Pen portocaliu;
         portocaliu = new System.Drawing.Pen(Brushes.OrangeRed, 25);
         gr.DrawLine(portocaliu, 60, 100, 150, 50);
      }
   } }
```

***Rezultate:***


***Aplicație:***

Să se creeze un formular folosind C# care la apăsarea unui buton să afiseze desenat un asterix cu 3 linii de culori și lungimi diferite, dar aceeași grosime.

***Ex. 4: Aplicatie de desenare a unui dreptunghi, cerc, elipsă și poligon***

Prima modalitate de desenare preyzentată este desenarea directă a dreptunghiului folosind funcția *DrawRectangle* specificând coordonatele colțului din stânga sus și lățimea și înălțimea:

```
myGraphics.DrawRectangle(myPen, X, Y, width, height)
private void Button1_Click(object sender, EventArgs e)
{System.Drawing.Graphics gr = this.CreateGraphics();
System.Drawing.Pen albastru;
albastru = new System.Drawing.Pen(Brushes.Blue, 10);
gr.DrawRectangle(albastru, 60, 100, 150, 50);}
```

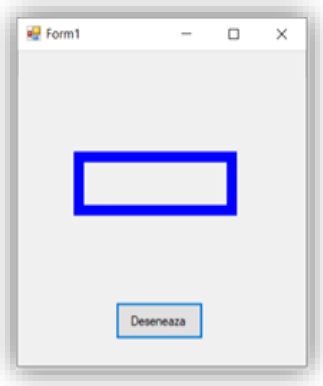



**Fig. 18.** Codul și formularul rezultat pentru desenarea unui dreptunghi

Linia cu care sunt desenate diferitele forme geometrice poate fi formatată pentru a fi punctată sau sub forma linie punct linie (Fig. 18) . Sintaxa pentru o astfel de modificare este:

```
myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dot
```

unde ultimul argument poate lua una din valorile *Dash, DashDot, DashDotDot* și *Solid*. Următorul cod desenează un dreptunghi cu o linie punctată albastră (Fig. 19).

```
private void Button1_Click(object sender, EventArgs e)
{ System.Drawing.Graphics gr = this.CreateGraphics();
System.Drawing.Pen albastru;
albastru = new System.Drawing.Pen(Brushes.Blue, 5);
albastru.DashStyle = System.Drawing.Drawing2D.DashStyle.Dot;
gr.DrawRectangle(albastru, 60, 100, 150, 50);       }
```

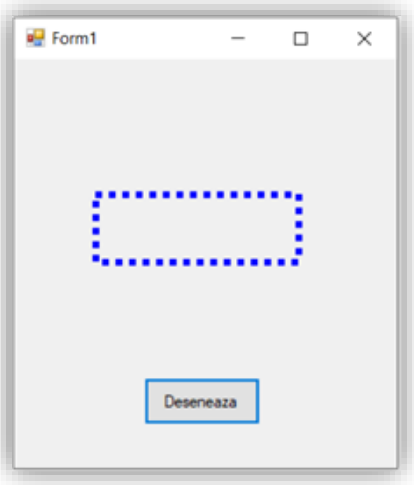



**Fig. 19.** Codul și formularul rezultat pentru desenarea unui dreptunghi cu linie punctată

În mod similar, pentru a desena o elipsă se folosește codul *gr.DrawEllipse(albastru, 60, 100, 150, 50)* ceea ce rezultă într-o elipsă înscrisă în dreptunghiului desenat cu argumentele funcției, iar pentru cerc codul *gr.DrawEllipse(albastru, 60, 100, 50, 50)* deoarece cercul este o elipsă cu dimensiunea razei constantă.

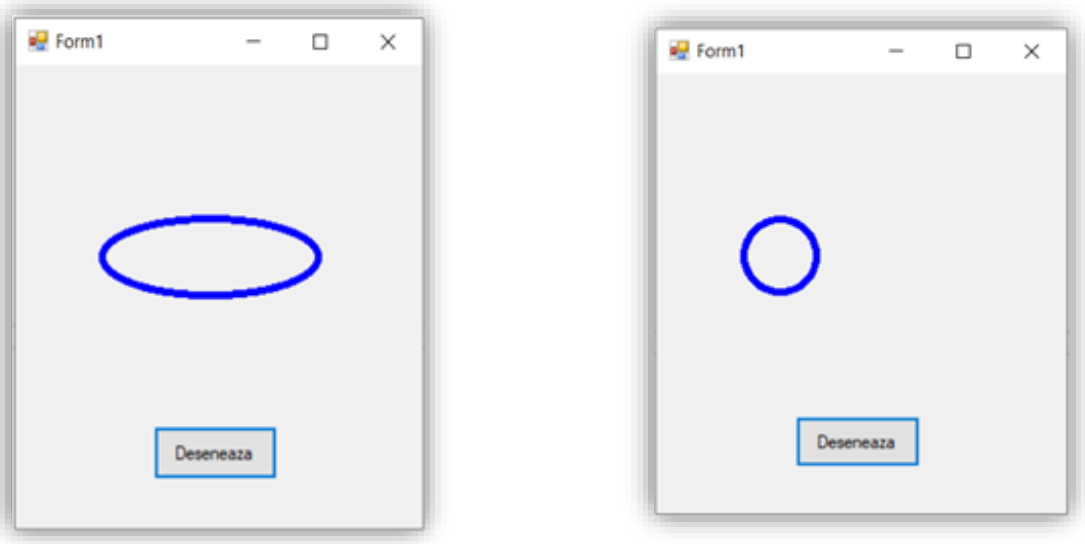



**Fig. 20.** Formularele rezultate pentru desenarea unei elipse și a unui cerc

Pentru a desena un poligon este necesară definirea a 3 sau mai multe puncte. Un punct se definește cu următorul cod :

```
Point point1 = new Point(x, y);
```

unde x și y sunt coordonatele punctului pe formular. După declararea punctelor, este necesară definirea unui șir de puncte care grupează toate punctele folosind sintaxa:

```
Point[] puncte = {point1, point2, point3 };
```

Desenarea unui poligon creat din 3 puncte  se realizeaza folosind sintaxa:

```
gr.DrawPolygon(creion, puncte);
private void Button1_Click(object sender, EventArgs e)
    {  System.Drawing.Graphics gr = this.CreateGraphics();
       System.Drawing.Pen albastru;
       albastru = new System.Drawing.Pen(Brushes.Blue, 5);
       Point point1 = new Point(10, 10);
       Point point2 = new Point(100, 50);
       Point point3 = new Point(60, 150);
      Point[] puncte = {point1, point2, point3 };
       gr.DrawPolygon(albastru, puncte);
    }
```

În Fig. 20 este ilustrata desenarea triunghiului (poligon) creat. Pentru mai multe laturi ale poligonului se vor defini mai multe puncte.

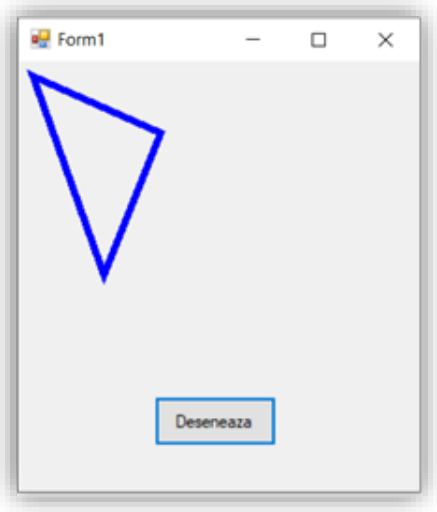



**Fig. 21.** Codul și formularul rezultat pentru desenarea unui poligon cu 3 laturi

Pentru a umple poligonul cu o anumită culoare se poate complete programul cu codul:

```
System.Drawing.Brush rosu;
rosu = new System.Drawing.SolidBrush(Color.IndianRed);
```

pentru crearea unei pensule și pentru umplerea poligonului *gr.FillPolygon(rosu, puncte)*. Codul final și rezultatul obținut în urma procesului se pot observa în Fig. 21. Pentru a umple alte forme geometrice se folosesc alte instrucțiuni specifice de exemplu pentru elipsă *FillEllipse*, pentru dreptunghi *FillRectangle*.

```
private void Button1_Click(object sender, EventArgs e)
    {  System.Drawing.Graphics gr = this.CreateGraphics();
       System.Drawing.Pen albastru;
       albastru = new System.Drawing.Pen(Brushes.Blue, 5);
       System.Drawing.Brush rosu;
       rosu = new System.Drawing.SolidBrush(Color.IndianRed);
       Point point1 = new Point(10, 10);
       Point point2 = new Point(100, 50);
       Point point3 = new Point(60, 150);
       Point[] puncte = {point1, point2, point3 };
       gr.DrawPolygon(albastru, puncte);
       gr.FillPolygon(rosu, puncte);
    }
```

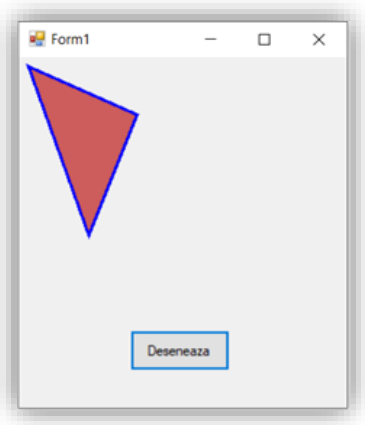


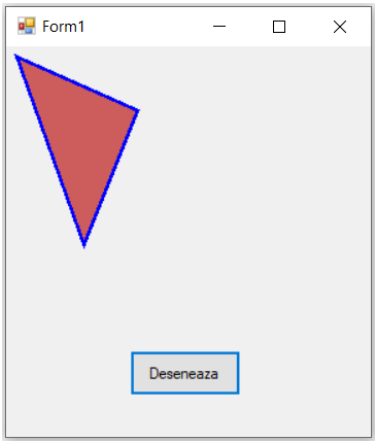

**Fig. 22.** Umplerea unei forme geometrice create cu formulare

Forma finala a aplicatiei de tip formular care include elemente grafice este prezentata in Fig. 17.

**Varianta in C#**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace grafic
{   public partial class Form1 : Form
   {       public Form1()
      {          InitializeComponent();}
      private void Button1_Click(object sender, EventArgs e)
      {
         System.Drawing.Graphics gr = this.CreateGraphics();
         System.Drawing.Pen albastru;
         albastru = new System.Drawing.Pen(Brushes.Blue, 5);
         System.Drawing.Brush rosu;
         rosu = new System.Drawing.SolidBrush(Color.IndianRed);
         Point point1 = new Point(10, 10);
         Point point2 = new Point(100, 50);
         Point point3 = new Point(60, 150);
        Point[] puncte = {point1, point2, point3 };
         gr.DrawPolygon(albastru, puncte);
         gr.FillPolygon(rosu, puncte);
      }
  }
 }
```

***Rezultate:***



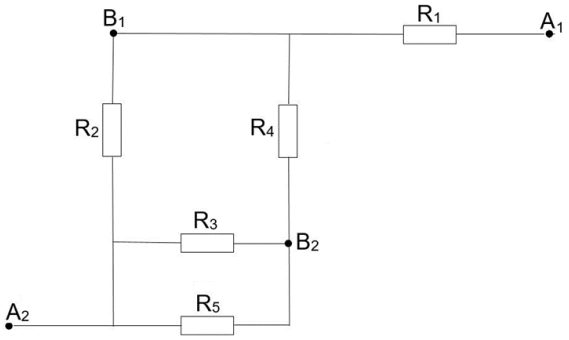

***Aplicație:***

Implementați un formular în C# folosind Visual Studio în care să se deseneze dreptunghiuri și cercuri concentrice. Modificați proprietățile de desenare astfel încât să aibă culori diferite

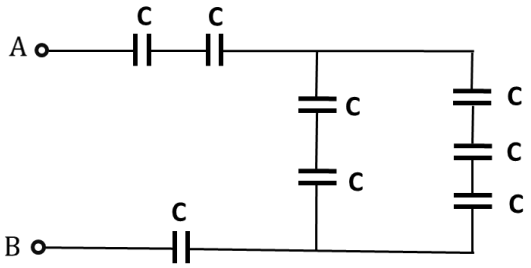
## PROBLEME PROPUSE

1. ***Să se creeze un formular cu ajutorul căruia utilizatorul, prin accesarea unor butoane să acceseze simbolurile predefinite ale elementelor din circuitele electrice împreună cu unele considerații teoretice despre acestea.***

2. ***Să se creeze un formular unde să se deseneze un circuit de forma celui de mai jos, iar utilizatorul să poată defini valorile pentru cele 5 rezistențe. Să ce calculeze rezistența echivalentă dintre bornele A1 și A2.***




3. ***2. Să se creeze un formular unde să se deseneze un circuit de forma celui de mai jos, iar utilizatorul să poată defini valorile pentru cele 8 condensatoare. Să se calculeze capacitatea echivalentă dintre bornele A și B.***



# Capitolul 14
# Aplicații WINDOWS în C# realizate cu Visual Studio. Partea a II-a

În acest capitol se continuă prezentarea modului de implementare a unor aplicații utilizând limbajul C# și mediul de programare Microsoft Visual Studio.

## PROBLEME REZOLVATE

***Ex. 1. Aplicația calculează prin intermediul unei interfețe de tip formular valorile unor funcții trigonometrice pentru valoare specificată de utilizator.***

Se creează un proiect Visual Studio Windows Form App (C#) în care se introduc în formular 8 elemente de tip *Label* pentru afișarea unor mesaje către utilizatori și pentru afișarea valorilor finale.

*Label*-urile din partea stângă vor corespunde funcțiilor matematice ce vor fi apelate pentru calcule și li se vor schimba denumirile din fereastra de Properties în *sin, cos, tg, ctg*, iar în partea dreaptă se va schimba mesajul din *labelx* în *val* sau se pot lăsa valorile implicite deoarece aici se vor afișa valorile calculate (Fig. 1).

Se va adăuga un buton de tip *NumericUpDown* din *Toolbox* pentru a introduce valoare pentru care utilizatorul dorește calcularea funcțiilor trigonometrice. În momentul în care este selectată comanda *NumericUpDown* se va modifica în fereastra *Properties* valoarea maximă la 10 și numărul de zecimale permise la 2 după cum se poate observa în Fig. 2. Se introduce de asemenea un *Label* în fața acestuia care să specifice că în căsuța *NumericUpDown* se va scrie o valoare.

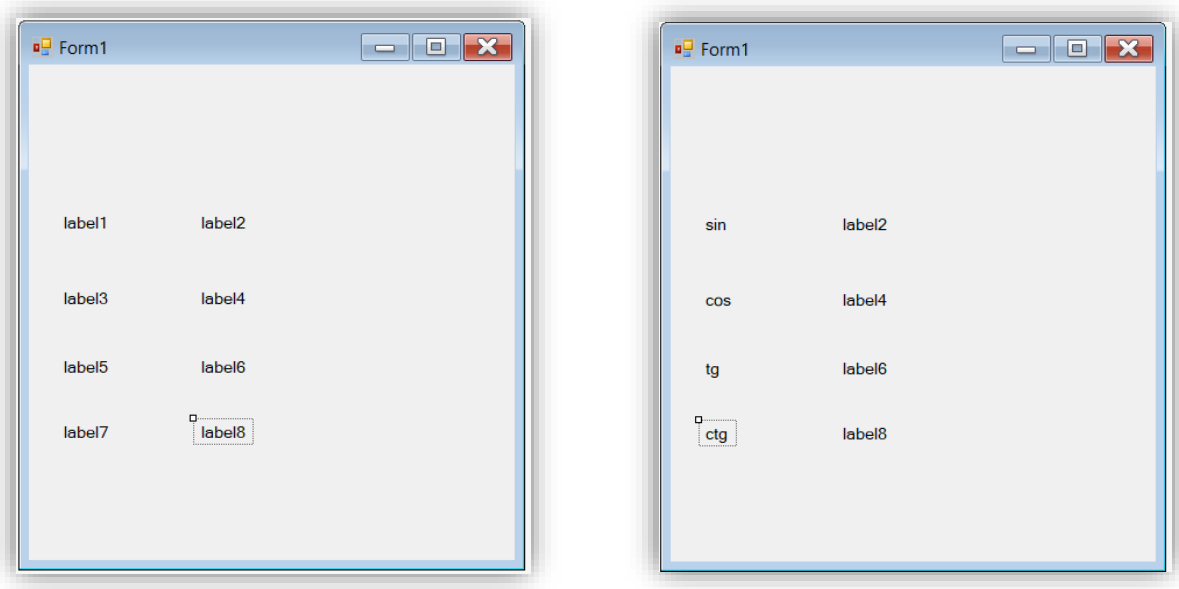



**Fig. 1.** Inserarea label-urilor și editarea acestora în formular

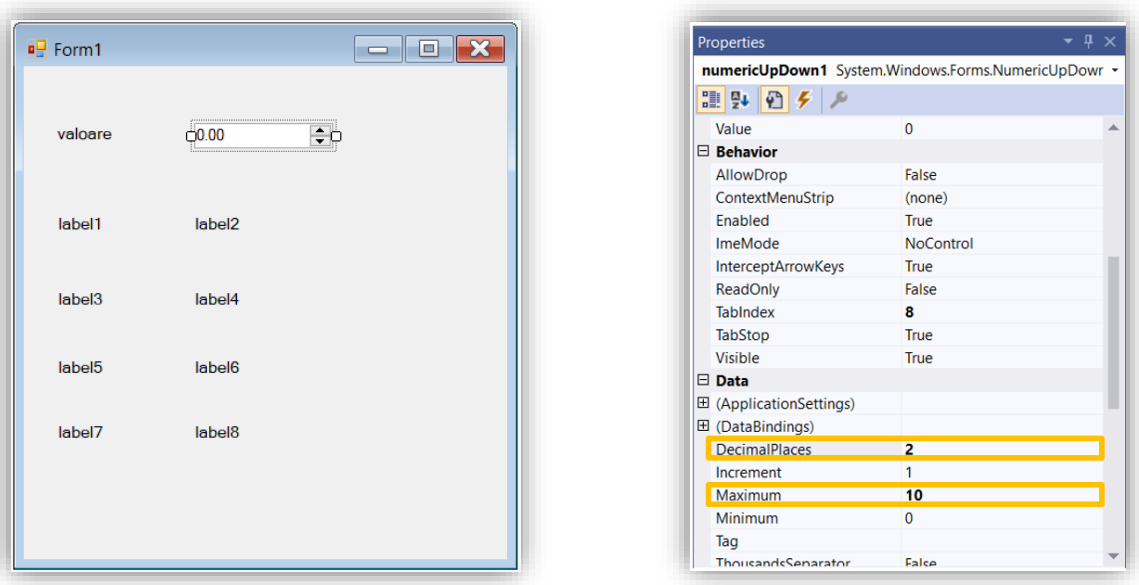



**Fig. 2.** Formatul final al formularului și definirea proprietăților pentru comanda *NumericUpDown*

Ultimul pas este scrierea codului pentru calcularea valorilor funcțiilor trigonometrice pentru valoarea dată de către utilizator.

Astfel, se dorește afișarea rezultatelor în momentul introducerii valorii inițiale în elementul *NumericUpDown*, deci cu dublu click pe această căsuță se scrie codul aferent prezentat mai jos.

Se definește în interiorul codului o variabilă de tip double rad2. În momentul în care se preia valoarea, aceasta este considerată de tip string și se convertește în double prin codul *rad2 = Convert.ToDouble(this.numericUpDown1.Value)*.

Pentru a afișa rezultatele obținute în interiorul label-urilor , acestea se vor transforma din nou în string cu instrucțiunea *Convert.ToString* iar în paranteză se specifica funcția, de exemplu *Math.Sin(rad2).*

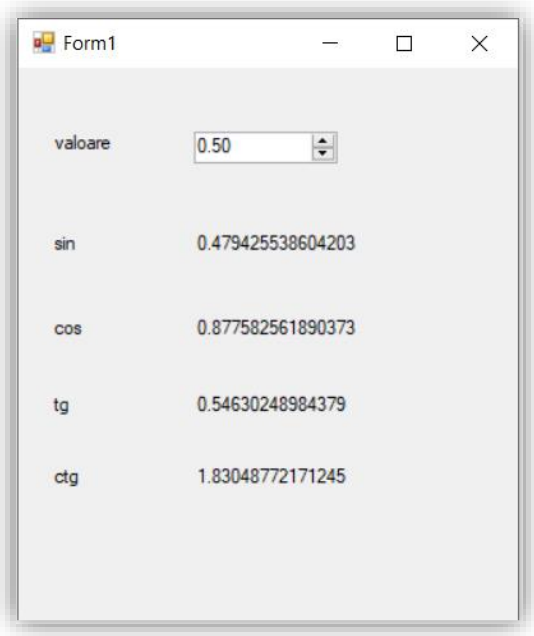
```
private void NumericUpDown1_ValueChanged(object sender, EventArgs e)
    { double rad2;
      rad2 = Convert.ToDouble(this.numericUpDown1.Value);
      this.label2.Text = Convert.ToString(Math.Sin(rad2));
      this.label4.Text = Convert.ToString(Math.Cos(rad2));
      this.label6.Text = Convert.ToString(Math.Tan(rad2));
      this.label8.Text = Convert.ToString(1/Math.Tan(rad2));
    }
```

Rezultatul compilării și execuției aplicației se poate observa în secțiunea Rezultate.

**Varianta în C#**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace functii_trigonometrice1
{   public partial class Form1 : Form
   {      public Form1()
      {       InitializeComponent();
      }
      private void NumericUpDown1_ValueChanged(object sender, EventArgs e)
      { double rad2;
         rad2 = Convert.ToDouble(this.numericUpDown1.Value);
         this.label2.Text = Convert.ToString(Math.Sin(rad2));
         this.label4.Text = Convert.ToString(Math.Cos(rad2));
         this.label6.Text = Convert.ToString(Math.Tan(rad2));
         this.label8.Text = Convert.ToString(1/Math.Tan(rad2));
      }
   }
}
```

***Rezultate:***




***Aplicație:***

Modificați formularul astfel încât să introduceți și alte funcții matematice.

***Ex. 2. Aplicația realizează operaţiile aritmetice cu numere complexe și calculează modulul şi argumentul acestora***

Pentru implementarea aplicatiei se va utiliza o alta aplicatie care rezolva aceeasi problema, prezentata intr-un capitol anterior si realizata in limbajele C si C++. În aplicația de fata însă, interfaţa cu utilizatorul, va fi de tip aplicație Windows si va fi realizata in C#.

Implementarea aplicatiei incepe cu creearea unui formular , pentru care se alege un proiect de tip Windows Form App (C#). Proiectul care se deschide are deja creat un formular pe care se pot adăuga diferite butoane și ferestre din Toolbox-ul oferit de mediul de programare Visual Studio.

Se vor adăuga 4 etichete de tip *Label* din Toolbox-ul oferit de Visual Studio și 4 elemente de tip *EditBox* în care utilizatorul va introduce valorile părților reale și imaginare a celor două valori complexe. Se continuă cu adăugarea unui *GrupBox* şi a 6 *butoane* în interiorul acestuia.

Se va schimba eticheta pentru GrupBox în *Operatii cu numere complexe* iar etichetele butoanelor şi ID-ul acestora se modifică astfel încât să identifice fiecare dintre operaţiile cu numere complexe (ca în Figura 3).Se va mai integra in formular un buton denumit *Iesire* pentru a ieși din formularul creat. Ultimul element introdus in formular va fi de tip *richTextBox* pentru afisarea rezultatelor.

Modificarea denumirii unei componente (de exemplu a Label-urilor se va face prin selectarea acestora și modificarea denumirii din fereastra Properties de unde se pot modifica și alte caracteristici ca de exemplu fontul sau culoarea scrisului).

În ferestrele de tip *Editbox* se vor scrie valorile pentru partea reală și imaginară a celor două numere complexe. Prin apăsarea unuia dintre butoanele din *GrupBox* se va efectua una dintre operații asupra acestor numere și rezultatul se va afișa în *richTextBox*.

Butonul de ieșire va face ca utilizatorul să iasă din aplicație. Codul scris pentru acest buton este:

```
private void Button1_Click(object sender, EventArgs e)
    { this.Close();      }
```

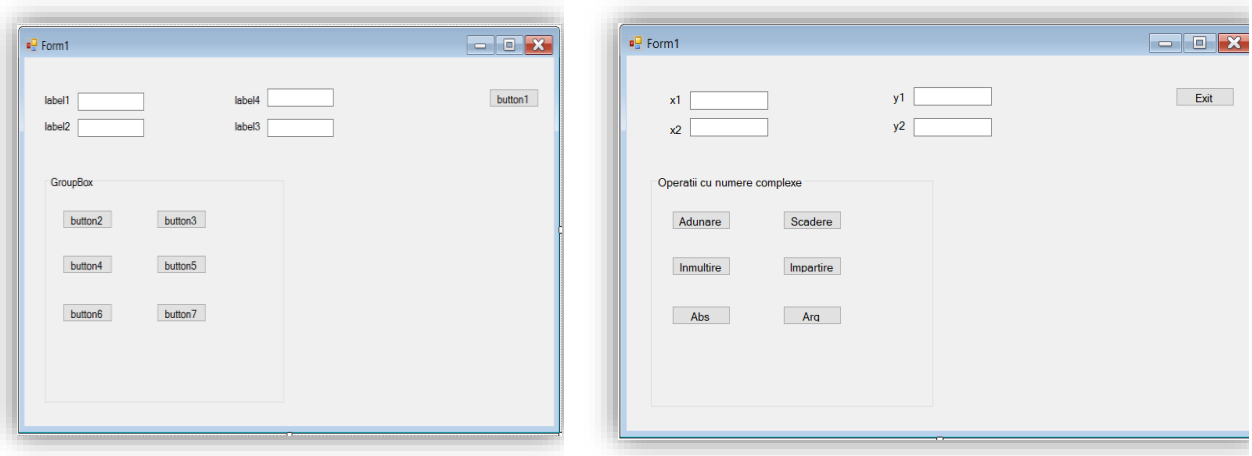



**Fig. 3.** Definirea formatului formularului și modificarea textului aferent butoanelor și al label-urilor

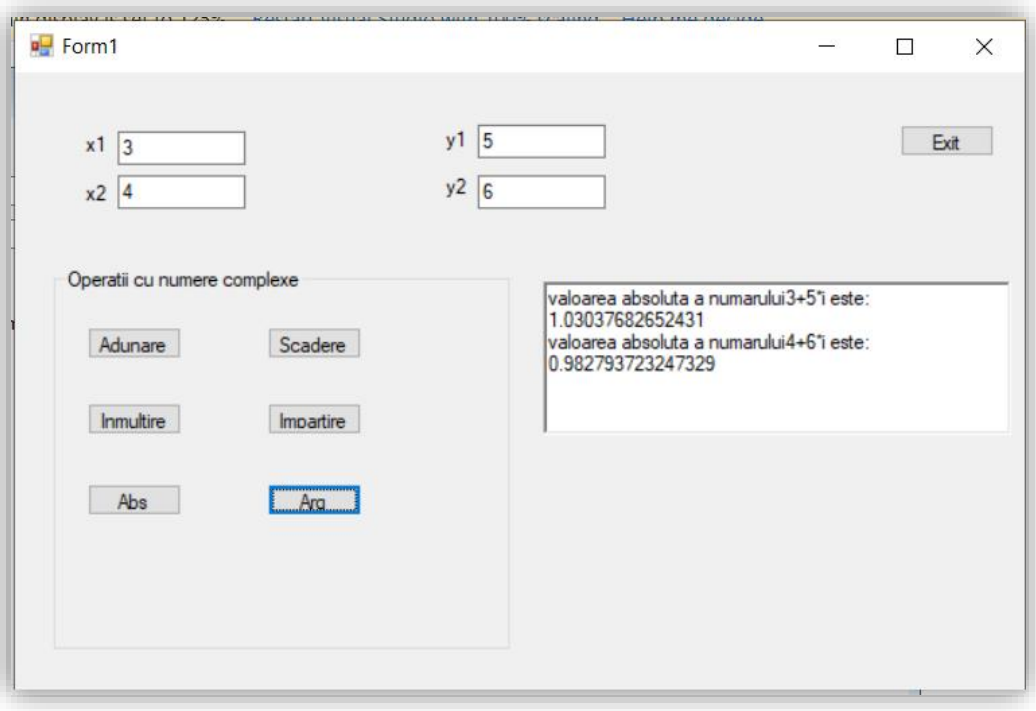
În urma creării formularului se va stabili funcționalitatea fiecărui buton în parte prin scrierea codului după ce s-a dat dublu click pe buton.

**Varianta in C#**

```
//pentru adunare
private void Button2_Click(object sender, EventArgs e)
      {  double resr1, resi1, resr2, resi2, resr,resi;
         resr1 = System.Convert.ToDouble(textBox1.Text);
         resi1 = System.Convert.ToDouble(textBox3.Text);
         resr2 = System.Convert.ToDouble(textBox2.Text);
         resi2 = System.Convert.ToDouble(textBox4.Text);
         resr = resr1 + resr2;
         resi = resi1 + resi2;
         richTextBox1.Text=resr.ToString()+"+"+resi.ToString()+"*i";
      }
//pentru scadere
private void Button3_Click(object sender, EventArgs e)
      {  double resr1, resi1, resr2, resi2, resr, resi;
         resr1 = System.Convert.ToDouble(textBox1.Text);
         resi1 = System.Convert.ToDouble(textBox3.Text);
         resr2 = System.Convert.ToDouble(textBox2.Text);
         resi2 = System.Convert.ToDouble(textBox4.Text);
         resr = resr1 -resr2;
         resi = resi1 - resi2;
         if (resi>0)
         richTextBox1.Text = resr.ToString() + "+" + resi.ToString() + "*i";
         else richTextBox1.Text = resr.ToString() + resi.ToString() + "*i";
      }
//pentru abs
private void Button6_Click(object sender, EventArgs e)
      {   double resr1, resi1, resr2, resi2, x, y;
         resr1 = System.Convert.ToDouble(textBox1.Text);
         resi1 = System.Convert.ToDouble(textBox3.Text);
         resr2 = System.Convert.ToDouble(textBox2.Text);
         resi2 = System.Convert.ToDouble(textBox4.Text);
         x = Math.Sqrt(resr1 * resr1 + resi1 * resi1);
         y = Math.Sqrt(resr2 * resr2 + resi2 * resi2);
         richTextBox1.Text ="modulul numarului"+ resr1.ToString() + "+" + resi1.ToString() + "*i este:
"+x.ToString()+"\n"+ "modulul numarului" + resr2.ToString() + "+"+  resi2.ToString() + "*i este: " + y.ToString() + "\n";
      }
//pentru inmultire
private void Button4_Click(object sender, EventArgs e)
      {  double resr1, resi1, resr2, resi2, resr, resi;
         resr1 = System.Convert.ToDouble(textBox1.Text);
         resi1 = System.Convert.ToDouble(textBox3.Text);
         resr2 = System.Convert.ToDouble(textBox2.Text);
         resi2 = System.Convert.ToDouble(textBox4.Text);
         resr = resr1*resr2-resi1*resi2;
         resi = resr1*resi2 - resr2*resi1;
         if (resi > 0)
         richTextBox1.Text = resr.ToString() + "+" + resi.ToString() + "*i";
         else richTextBox1.Text = resr.ToString() + resi.ToString() + "*i";
      }
//pentru impartire
private void Button5_Click(object sender, EventArgs e)
      {  double resr1, resi1, resr2, resi2, resr, resi;
         resr1 = System.Convert.ToDouble(textBox1.Text);
         resi1 = System.Convert.ToDouble(textBox3.Text);
         resr2 = System.Convert.ToDouble(textBox2.Text);
         resi2 = System.Convert.ToDouble(textBox4.Text);
         resr = (resr1 * resr2 - resi1 * resi2)/(resr2*resr2+resi2*resi2);
```

```
        resi = (resr2 * resi1 - resr1 * resi2)/ (resr2 * resr2 + resi2 * resi2);
        if (resi > 0)
           richTextBox1.Text = resr.ToString() + "+" + resi.ToString() + "*i";
        else richTextBox1.Text = resr.ToString() + resi.ToString() + "*i";
      }
//pentru absolut
private void Button7_Click(object sender, EventArgs e)
      {  double resr1, resi1, resr2, resi2, a, b;
         resr1 = System.Convert.ToDouble(textBox1.Text);
         resi1 = System.Convert.ToDouble(textBox3.Text);
         resr2 = System.Convert.ToDouble(textBox2.Text);
         resi2 = System.Convert.ToDouble(textBox4.Text);
         a= Math.Atan(resi1 / resr1);
         b=Math.Atan(resi2 / resr2);
         if ((a >= 0)&&(b>=0)) richTextBox1.Text = "valoarea absoluta a numarului" + resr1.ToString() + "+" +
resi1.ToString() + "*i este: " + a.ToString() + "\n" + "valoarea absoluta a numarului" + resr2.ToString() + "+" +
resi2.ToString() + "*i este: " + b.ToString() + "\n";
         else if ((a < 0) && (b >= 0)) richTextBox1.Text = "valoarea absoluta a numarului" + resr1.ToString() + "+" +
resi1.ToString() + "*i este: " + (a+2*3.14).ToString() + "\n" + "valoarea absoluta a numarului" + resr2.ToString() + "+" +
resi2.ToString() + "*i este: " + b.ToString() + "\n";
         else if ((a >= 0) && (b <0)) richTextBox1.Text = "valoarea absoluta a numarului" + resr1.ToString() + "+" +
resi1.ToString() + "*i este: " + a.ToString() + "\n" + "valoarea absoluta a numarului" + resr2.ToString() + "+" +
resi2.ToString() + "*i este: " + (b+2*3.14).ToString() + "\n";
         else  richTextBox1.Text = "valoarea absoluta a numarului" + resr1.ToString() + "+" + resi1.ToString() + "*i este:
" + (a+2*3.14).ToString() + "\n" + "valoarea absoluta a numarului" + resr2.ToString() + "+" + resi2.ToString() + "*i este:
" + (b+2*3.14).ToString() + "\n";       }
```

***Rezultate:***




***Aplicație:***

Modificați formularul astfel încât să se definească 4 numere naturale și să se calculeze operații aritmetice cu acestea.

***Ex. 3. Aplicația generează un grid la apăsarea unui buton și afișează graficul funcției f(x)=x² utilizând gridul deja definit.***

În cadrul acestei aplicații se creează în partea dreapta trei butoane (Fig. 4) utilizand elemente Toolbox, lasând loc în partea stânga a interfeţei aplicaţiei pentru reprezentarea grafică a funcţiei.

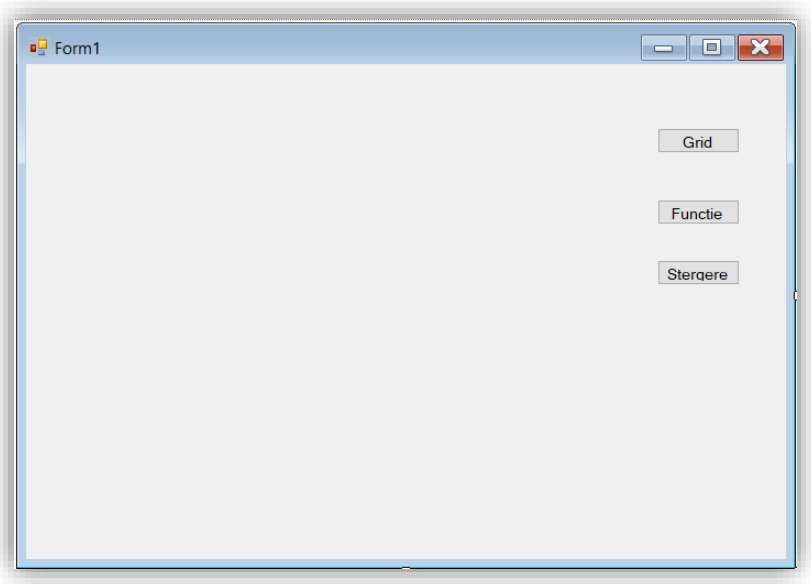



**Fig. 4.** Definirea formularului

Un buton se va numi *Grid* și va desena grid-ul, cel de al doilea se va numi *Funcție* și va genera graficul funcției definite în cod, iar cel de al treilea buton, *Stergere*, va șterge partea grafică apărută în formular în urma apăsării celor 2 butoane anterior amintite . În cauzul butonului *Grid* codul care va fi scris este:

```
private void Button1_Click(object sender, EventArgs e)
    {  int i;
       System.Drawing.Graphics gr = this.CreateGraphics();
       System.Drawing.Pen gri;
       gri = new System.Drawing.Pen(System.Drawing.Color.Gray);
       System.Drawing.Pen grid;
       grid = new System.Drawing.Pen(System.Drawing.Color.LightGray);
       for(i=0; i<this.Width; i+=10)
       { if (i % 50 == 0) gr.DrawLine(gri, i, 0, i, this.Height);
       else gr.DrawLine(grid, i, 0, i, this.Height);
       }
      for (i = 0; i < this.Height; i += 10)
       {  if (i % 50 == 0) gr.DrawLine(gri, 0, i, this.Width,i);
          else gr.DrawLine(grid, 0, i, this.Width, i);
       }
```

Pentru început se creează un spațiu grafic și se definesc 2 creioane, și anume unul gri deschis și unul gri închis. Urmează 2 instrucțiuni for. Prima dată se vor desena liniile verticale, de aceea se va merge cu o variabilă de la 0 la lățimea formularului desenându-se linii din 10 în 10 pixeli. Tot a 5-a linie se va desena cu un gri închis. Cel de al doilea for este creat pentru desenarea liniilor orizontale pe toată înălțimea formularului. Rezultatul obținut în urma apăsării butonului se poate observa în Fig. 5.

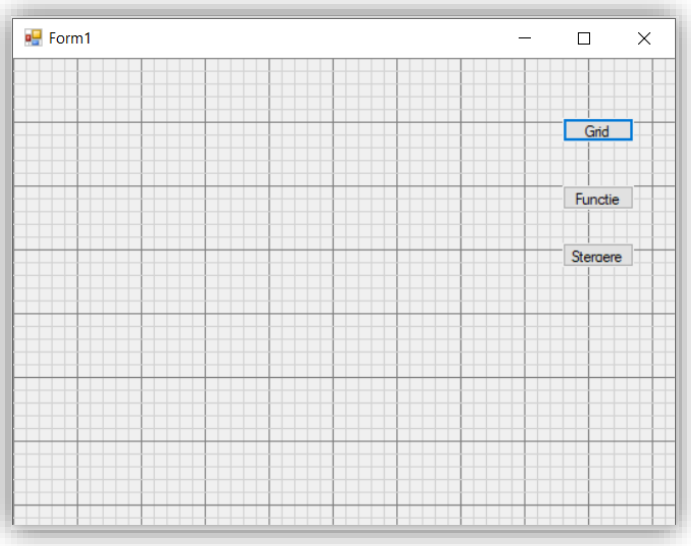



**Fig. 5.** Afișarea grid-ului în urma apăsării butonului *grid*

În continuare se va scrie codul pentru desenarea funcției **f=x²** dând dublu click pe butonul definit *Funcție*. Se va defini un spațiu grafic și un creion roșu după care se vor defini 2 variabile pentru înălțime și lățime astfel încât graficul să fie încadrat în formular.

Se desenează graficul funcției calculând puncte succesive și unindu-le prin linii. În Fig. 6 se poate observa graficul care se poate suprapune peste partea de grid care a fost desenată în prealabil.

```
private void Button2_Click(object sender, EventArgs e)
    {
      float h, i, lat, x, y = 0, yv = 0;
      System.Drawing.Graphics gr = this.CreateGraphics();
      System.Drawing.Pen rosu;
      rosu = new System.Drawing.Pen(System.Drawing.Color.Red);
      h = this.Height - 50;
      i = 0;
      lat = this.Width - 50;
      for(x=-lat/2;x<=lat/2;x++)
      { y = h - x * x / (lat / 4);
        gr.DrawLine(rosu, i - 1, yv, i, y);
        yv = y;
        i = i + 1;
      }
    }
```

Pentru a închide aplicatia de tip formular se definește un nou buton Inchide. Codul generat pentru acesta este:

```
private void Button3_Click(object sender, EventArgs e)
    {
      this.Close();
    }
```

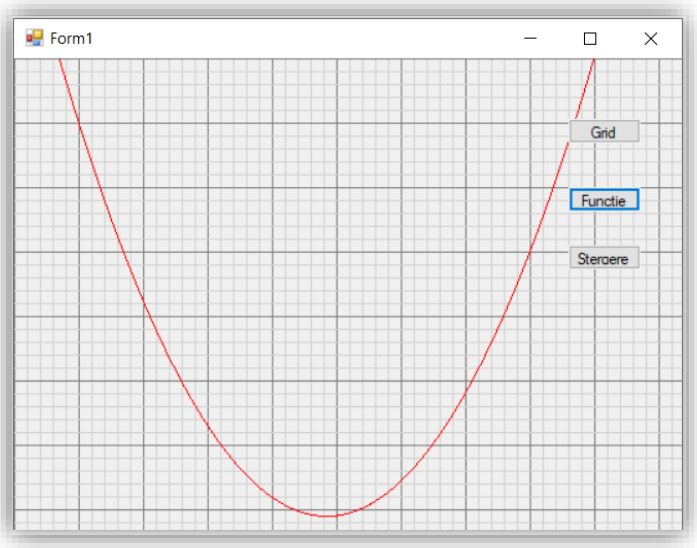


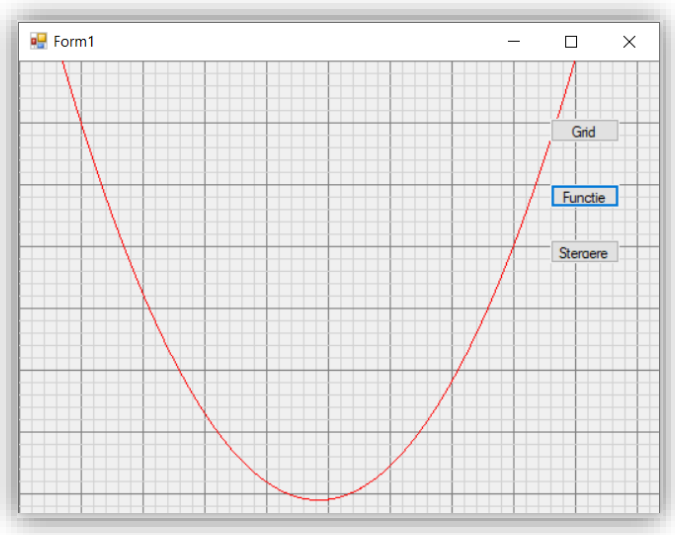

**Fig. 6.** Afișarea grid-ului și al graficului funcției $x^2$ în urma apăsării butoanelor *grid* și *funcție*

**Varianta in C#**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp3
{   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void Button1_Click(object sender, EventArgs e)
      {
         int i;
         System.Drawing.Graphics gr = this.CreateGraphics();
         System.Drawing.Pen gri;
         gri = new System.Drawing.Pen(System.Drawing.Color.Gray);
         System.Drawing.Pen grid;
         grid = new System.Drawing.Pen(System.Drawing.Color.LightGray);
         for(i=0; i<this.Width; i+=10)
         { if (i % 50 == 0) gr.DrawLine(gri, i, 0, i, this.Height);
         else gr.DrawLine(grid, i, 0, i, this.Height);
         }
        for (i = 0; i < this.Height; i += 10)
         {  if (i % 50 == 0) gr.DrawLine(gri, 0, i, this.Width,i);
            else gr.DrawLine(grid, 0, i, this.Width, i);
         }
      }
      private void Button2_Click(object sender, EventArgs e)
      {
```

```
        float h, i, lat, x, y = 0, yv = 0;
        System.Drawing.Graphics gr = this.CreateGraphics();
        System.Drawing.Pen rosu;
        rosu = new System.Drawing.Pen(System.Drawing.Color.Red);
        h = this.Height - 50;
        i = 0;
        lat = this.Width - 50;
        for(x=-lat/2;x<=lat/2;x++)
        { y = h - x * x / (lat / 4);
          gr.DrawLine(rosu, i - 1, yv, i, y);
          yv = y;
          i = i + 1;  }
    }
    private void Button3_Click(object sender, EventArgs e)
    {          this.Close();       }
  }
}
```

***Rezultate:***




***Aplicație:***

Modificați formularul astfel încât să se deseneze graficul pentru funcția f(x)=x3+2x-3

***Ex. 4. Aplicația de tip desktop realizată în C# este destinată calculului integralelor*** $\int x^n$, $\int \frac{1}{x+1}$ ***utilizând formulare.***

Din fereastra cu elemente *Toolbox*, opțiunea *Menus & Toolbars* , se adaugă un meniu de tip *Menu Strip* denumit *Tipul integralei*, care permite selectarea uneia dintre cele două integrale .

Se vor adăuga 2 opțiuni: *Integrala din x la n* și *Integrala din 1/(x+1)* după cum se poate observa în Fig. 7 sau adăugând elemente noi direct din formular unde apare scris *Type Here*.

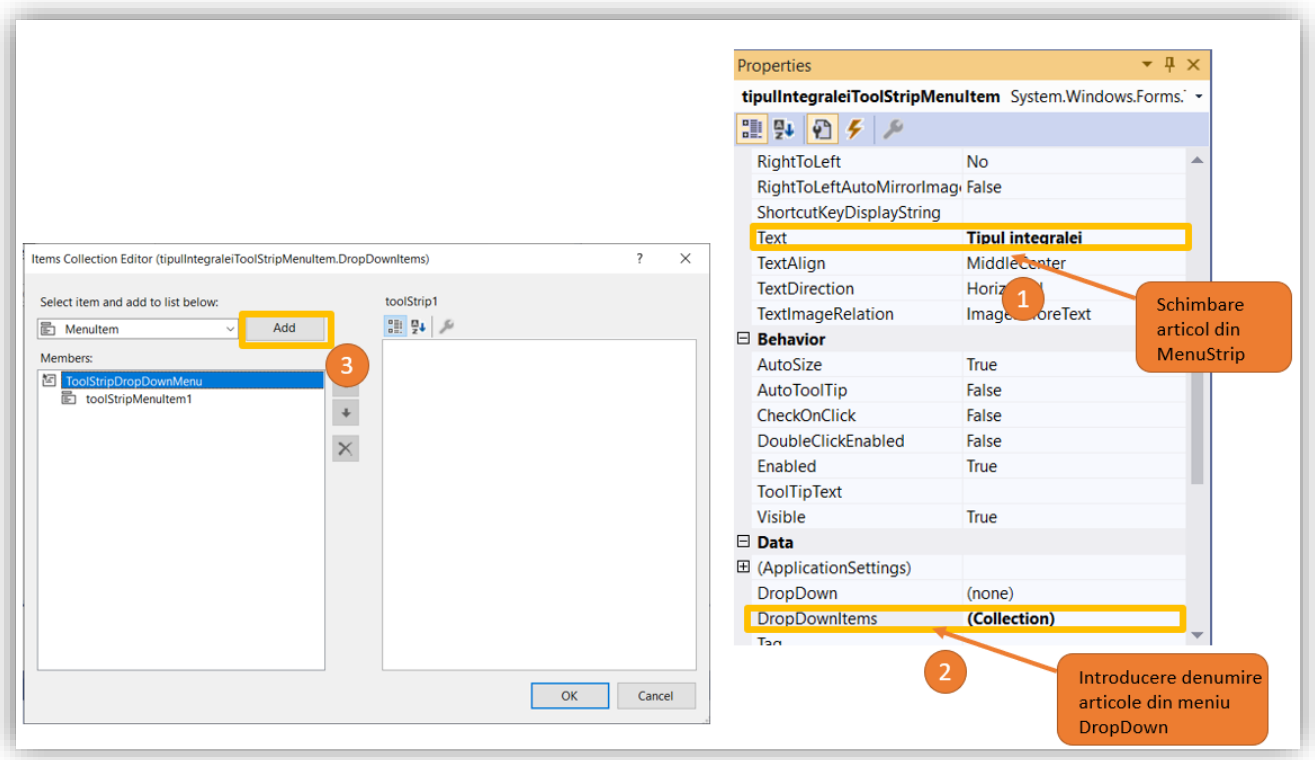



**Fig. 7.** Crearea toolbar-ului

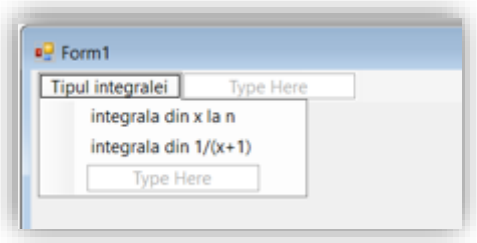



**Fig. 8.** Modul de selectare a integralei în aplicaţie

În fereastra *Solution Explorer* se da click dreapta pe numele proiectului Integrale și se selectează opțiunea *Add*, iar din meniul derulant se alege opțiunea *New Item*. Se selectează opțiunea *Windows Form*, iar la nume (Name) se trece numele formularului Form2.cs respectiv Form3.cs.

Ca urmare, în fereastra *Solution Explorer* se mai adaugă formularele create (Figura 9). Din fereastra *Properties* se pot modifica mai multe proprietăți ale formularelor: BackColor, Text, Font, FontColor, etc.

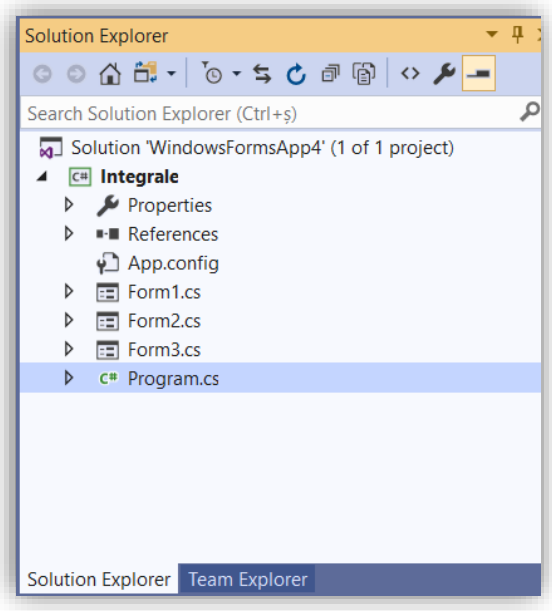



**Fig. 9.** Adăugarea unor noi elemente în cadrul proiectului

Dacă se dă dublu click pe una dintre opțiunile dim meniul de tip *DropDown* se va deschide partea de cod corespunzătoare ei. Astfel cu dublu click pe *Integrala din x* la n se scrie codul care va deschide un nou formular:

```
private void ToolStripMenuItem1_Click(object sender, EventArgs e)
    {  Form2 f2 = new Form2();
       f2.Show();       }
```

Pentru a defini designul celui de-al doilea formular se va da dublu click pe Form2.cs din fereastra *Solution Explorer*. Formularul 2 va fi definit ca în Fig. 10 . conținând label-uri, butoane și textbox-uri. Fiecare dintre aceste elemente vor fi configurate după cum se poate observa în aceeași figură.


**Fig. 10.** Definirea formularului pentru prima integrală

Codul care determină calcularea rezultatelor la apăsarea butonului *Calculati* este:

```
        private void Button1_Click(object sender, EventArgs e)
            {  double a, b, n;
               double x;
               if(textBox1.Text==""|| textBox2.Text == "" || textBox3.Text == "" )
               { MessageBox.Show("Nu ati completat toate campurile", "Atentie!", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning);  }
               else
               { a = Convert.ToInt32(this.textBox2.Text);
                 b = Convert.ToInt32(this.textBox3.Text);
                 n = Convert.ToInt32(this.textBox1.Text);
                 x = ((Math.Pow(b, n + 1) - Math.Pow(a, n + 1)) / (n + 1));
                 this.textBox4.Text = this.textBox4.Text + Convert.ToString(x);          }
```

La apăsarea butonului *Calcul Nou* se vor șterge valorile introduse și rezultatul obținut. Codul pentru această acțiune se va scrie dând dublu click pe acest buton și este:

```
        private void Button2_Click(object sender, EventArgs e)
            {  this.textBox4.Text = "";
               this.textBox2.Text = "";
               this.textBox3.Text = "";
               this.textBox1.Text = "";      }
```

Pentru butonul de ieșire care închide formularul cu ajutorul codului:

```
        private void Button3_Click(object sender, EventArgs e)
            {  this.Close();}
```

Cu dublu click pe *Integrala din 1/(x+1)* se scrie codul care va deschide un nou formular:

```
         private void ToolStripMenuItem2_Click(object sender, EventArgs e)
            {         Form3 f3 = new Form3();
              f3.Show();      }
```

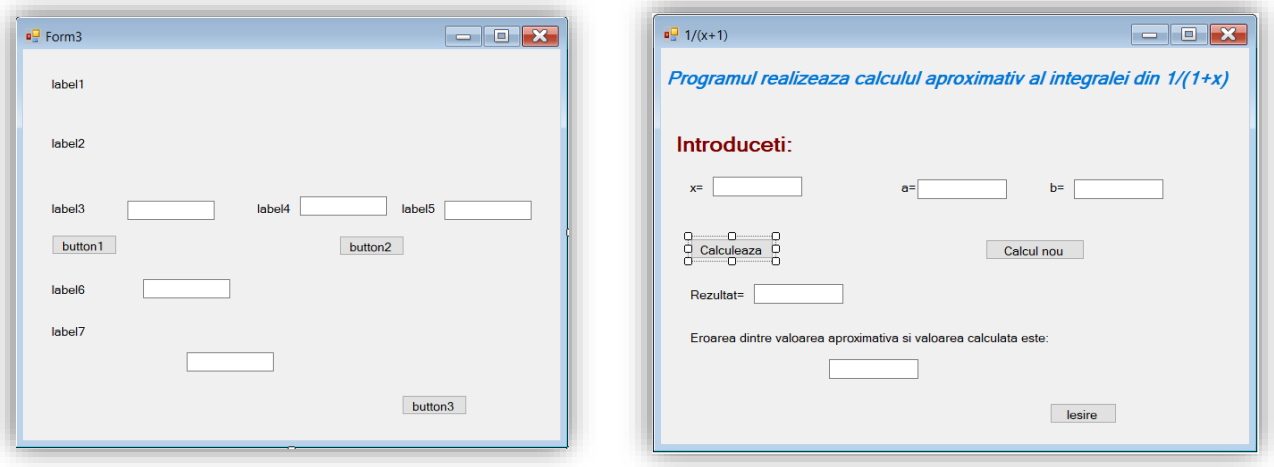



**Fig. 10.** Definirea formularului 3 pentru ce-a de-a doua integrală

Codul pentru butonul *Calculati* este:

```
        private void Button1_Click(object sender, EventArgs e)
            {
               int i, n;
               double a, b, h, rex, r;
               if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "")
               {
                  MessageBox.Show("Nu ati completat toate campurile", "Atentie!", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning);          }
               else
               {
                  a = Convert.ToInt32(this.textBox2.Text);
                  b = Convert.ToInt32(this.textBox3.Text);
                  n = Convert.ToInt32(this.textBox1.Text);
                  a = 0;b = 1;h = (b - a) / n;r = 0;rex = 0.6931471805;
                  for (i = 1; i <= n; i++)
                  { r += 1 / ((a - (h / 2) + i * h) + 1.0); }
                     r *= h;
                  this.textBox4.Text = this.textBox4.Text + Convert.ToString(rex);
                  this.textBox5.Text = this.textBox5.Text + Convert.ToString(rex-r);
               }
            }
```

Pentru butonul *Calcul Nou* codul va fi:

```
        private void Button2_Click(object sender, EventArgs e)
             {
                this.textBox4.Text = "";
                this.textBox2.Text = "";
                this.textBox3.Text = "";
                this.textBox1.Text = "";
                this.textBox5.Text = "";
            }
```

Butonul *Iesire* va avea codul:

```
        private void Button3_Click(object sender, EventArgs e)
            {         this.Close();
            }
```

**Varianta in C#**

***Pentru Form 1***

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp4
{   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
```

```
private void ToolStripMenuItem1_Click(object sender, EventArgs e)
      {
         Form2 f2 = new Form2();
         f2.Show();
      }
      private void ToolStripMenuItem2_Click(object sender, EventArgs e)
      {
         Form3 f3 = new Form3();
         f3.Show();
      }
   }
}
```

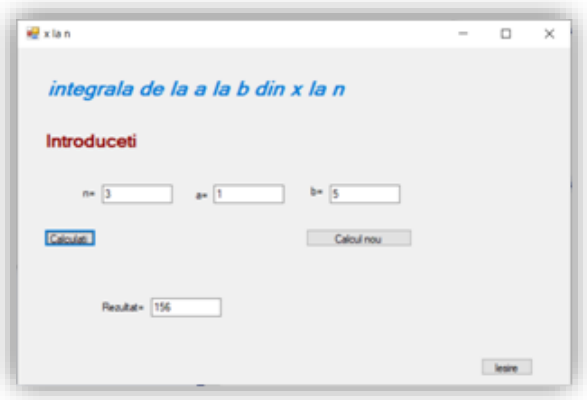
***Pentru Form 2***

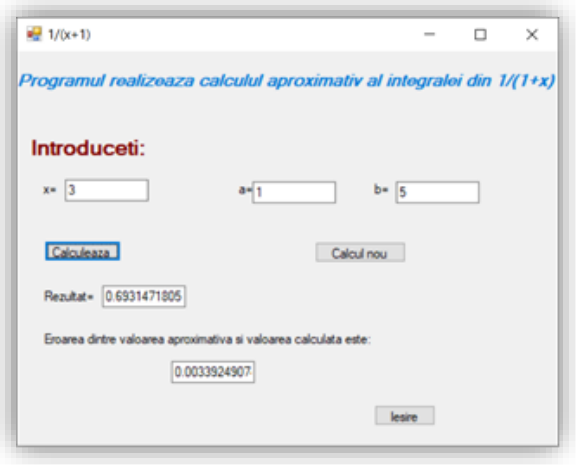
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp4
{    public partial class Form2 : Form
   {
      public Form2()
      {
         InitializeComponent();
      }
      private void Button1_Click(object sender, EventArgs e)
      {
         double a, b, n;
         double x;
         if(textBox1.Text==""|| textBox2.Text == "" || textBox3.Text == "" )
         { MessageBox.Show("Nu ati completat toate campurile", "Atentie!", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning);          }
         else
         { a = Convert.ToInt32(this.textBox2.Text);
           b = Convert.ToInt32(this.textBox3.Text);
           n = Convert.ToInt32(this.textBox1.Text);
             x = ((Math.Pow(b, n + 1) - Math.Pow(a, n + 1)) / (n + 1));
             this.textBox4.Text = this.textBox4.Text + Convert.ToString(x);
         }
      }
      private void Button2_Click(object sender, EventArgs e)
      {
         this.textBox4.Text = "";
         this.textBox2.Text = "";
         this.textBox3.Text = "";
         this.textBox1.Text = "";
      }
      private void Button3_Click(object sender, EventArgs e)
      {
         this.Close();
      }
   }
}
```

***Pentru Form 3***

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp4
{    public partial class Form3 : Form
   {       public Form3()
      {
         InitializeComponent();
      }
      private void Button1_Click(object sender, EventArgs e)
      {
         int i, n;
         double a, b, h, rex, r;
         if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "")
         {
            MessageBox.Show("Nu ati completat toate campurile", "Atentie!", MessageBoxButtons.OKCancel,
MessageBoxIcon.Warning);
         }
         else
         {
            a = Convert.ToInt32(this.textBox2.Text);
            b = Convert.ToInt32(this.textBox3.Text);
            n = Convert.ToInt32(this.textBox1.Text);
            a = 0;b = 1;h = (b - a) / n;r = 0;rex = 0.6931471805;
            for (i = 1; i <= n; i++)
            { r += 1 / ((a - (h / 2) + i * h) + 1.0); }
               r *= h;

            this.textBox4.Text = this.textBox4.Text + Convert.ToString(rex);
            this.textBox5.Text = this.textBox5.Text + Convert.ToString(rex-r);
         }
      }
      private void Button2_Click(object sender, EventArgs e)
        {
            this.textBox4.Text = "";
            this.textBox2.Text = "";
            this.textBox3.Text = "";
            this.textBox1.Text = "";
            this.textBox5.Text = "";
      }
      private void Button3_Click(object sender, EventArgs e)
      {
         this.Close();
      }
   }
}
```

***Rezultate:***




***Aplicație:***

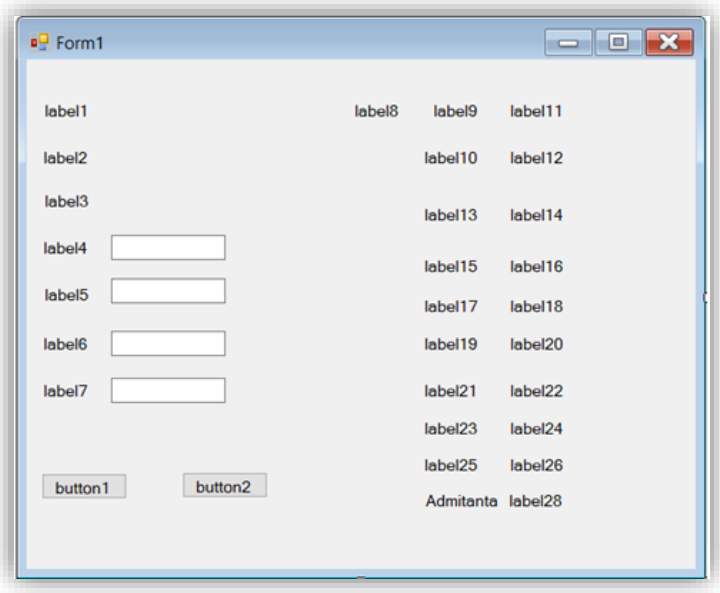
Modificați formularul astfel încât introduceți un nou tab pentru deschiderea unui nou formular de calcul pentru alte tipuri de integrale.

***Ex. 5. Aplicația calculează valorile caracteristice pentru un circuit RLC serie în regim sinusoidal.***

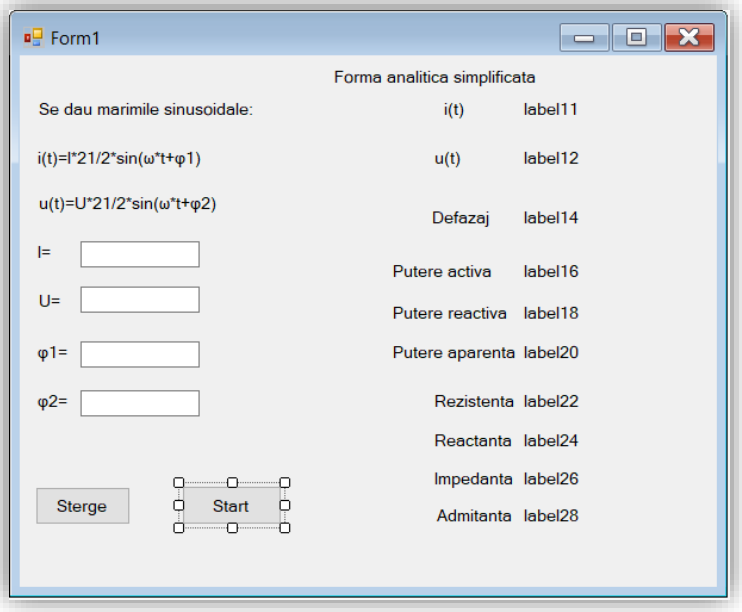
Aplicația calculează într-o aplicație de tip desktop: admitanța, reactanța, rezistența, puterea aparentă, puterea activă, puterea reactivă, defazajul și forma analitică simplificată a celor 2 ecuații sinusoidale introduse de utilizator. Cele două ecuații sinusoidale ale circuitului serie sunt scrise sub forma lor generală, utilizatorul trebuind să introducă variabilele.

Formularul conține în total 28 de label-uri. Primele trei afișează forma generală a ecuațiilor sinusoidale. Următoarele patru îi indică utilizatorului datele care trebuie introduse în TextBox-uri. 11 label-uri îi arată rezultatele care urmează a fi afișate. Ultimele 10 label-uri nu sunt vizibile utilizatorului până la apăsarea butonului “Start”. În aceste 10 label-uri sunt afișate cele 10 rezultate obținute în urma rulării aplicației.

Aplicația conține o singură Pagină principală, care conține toate elementele prezentate anterior. În partea dreaptă a Figurii 11 se poate observa modul în care s-au definit label-urile și formularul în versiunea lui finală. Pentru a putea scrie caractere speciale, formula a fost redactată inițial în word, după care a fost scrisă în fereastra *Properties* în loc de textul standard al label-ului.




a) Adaugare elemente din Toolbox in design-ul aplicatiei




b) Redenumirea elementelor si configurarea interfetei aplicatiei

**Fig. 11.** Definirea formularului

Pentru ca valorile calculate să apară doar în momentul în care utilizatorul dă click pe butonul *Start* se vor selecta label-urile unde apar rezultate și din meniul *Properties* la vizibilitate se alege *False*.

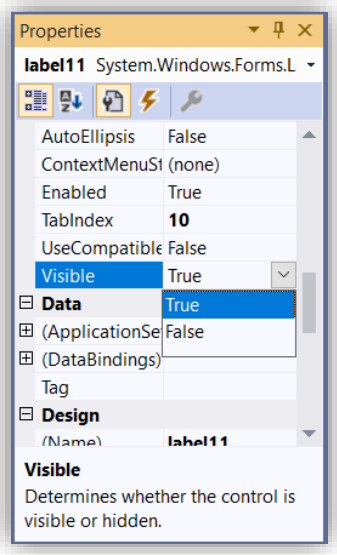



**Fig. 12.** Modificarea vizibilității unui label în momentul rulării aplicației

Butonul *Start* este cel care declanșează rularea programului. Odată ce butonul *Start* este apăsat, programul verifică dacă toate *TextBox*-urile au fost completate și în acest caz rulează și afișează rezultatele. În cazul în care nu toate *TextBox*-urile au fost completate utilizatorul va primi un mesaj care îi cere să introducă toate datele.

Se convertesc valorile pentru fi1, fi2, I și U în numere, după care cu ajutorul formulelor se calculează fiecare valoare în parte. Fiecare valoare este apoi transformată într-o valoare de tip string și afișată în label-urile destinate rezultatelor.
Pentru ca utililizatorul să poată vedea rezultatele, label-urile ascunse vor deveni vizibile prin o linie de cod de genul: *label11.Visible = true.*

```
private void Button2_Click(object sender, EventArgs e)
    {  if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "")
       {
          MessageBox.Show("Va rog completati toate datele");
       }
       else
       {  double f1, f2, i, u;
          f1 = Convert.ToDouble(textBox3.Text);
          f2 = Convert.ToDouble(textBox4.Text);
          i = Convert.ToDouble(textBox1.Text);
          u = Convert.ToDouble(textBox2.Text);
          double A = i * Math.Cos(f1); double A2 = Math.Round(A, 4);
          double B = i * Math.Sin(f1); double B2 = Math.Round(B, 4);
          double A1 = u * Math.Cos(f2); double A3 = Math.Round(A1, 4);
          double B1 = u * Math.Sin(f2); double B3 = Math.Round(B1, 4);
          double Fi = f2 - f1; double Fi1 = Math.Round(Fi, 4);
          double P = i * u * Math.Cos(Fi); double P1 = Math.Round(P, 4);
          double Q = i * u * Math.Sin(Fi); double Q1 = Math.Round(Q, 4);
          double R = P / (i * i); double R1 = Math.Round(R, 4);
          double X = Q / (i * i); double X1 = Math.Round(X, 4);
          double Z = u / i; double Z1 = Math.Round(Z, 4);
          double S = u * i; double S1 = Math.Round(S, 4);
          double Y = i / u; double Y1 = Math.Round(Y, 4);
          label11.Text = A2 + " + j " + B2;
          label12.Text = A3 + " + j " + B3;
```

```
                label14.Text = Fi1.ToString();
                label16.Text = P1.ToString();
                label18.Text = Q1.ToString();
                label20.Text = S1.ToString();
                label22.Text = R1.ToString();
                label24.Text = X1.ToString();
                label26.Text = Z1.ToString();
                label28.Text = Y1.ToString();
                label11.Visible = true;
                label12.Visible = true;
                label14.Visible = true;
                label16.Visible = true;
                label18.Visible = true;
                label20.Visible = true;
                label22.Visible = true;
                label24.Visible = true;
                label26.Visible = true;
                label28.Visible = true;
            }
```

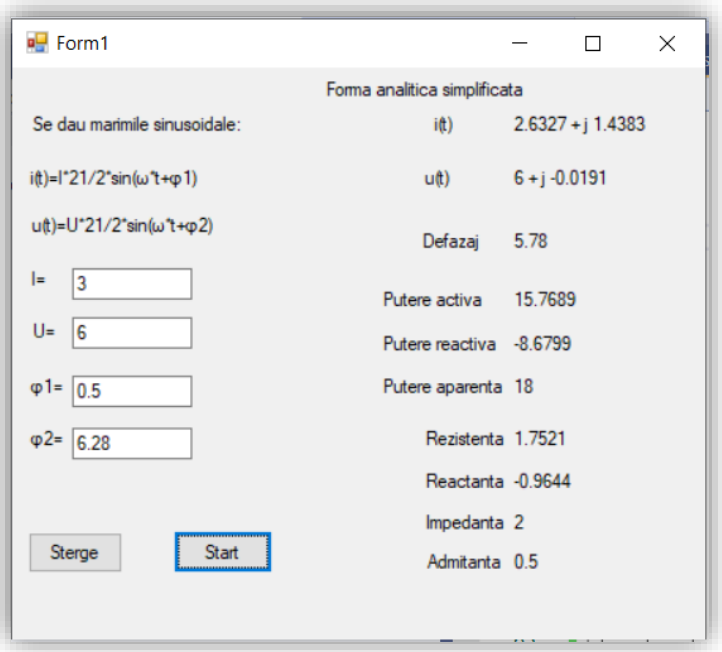
Butonul *Sterge* sterge toate datele din toate *TextBox*-urile și are codul:

```
        private void Button1_Click(object sender, EventArgs e)
            {    foreach (Control X in this.Controls)
                 {
                    if (X is TextBox)
                       X.Text = "";
                 }
          }
```

**Varianta in C#**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApp5
{   public partial class Form1 : Form
   {       public Form1()
      {          InitializeComponent();       }
      private void Button2_Click(object sender, EventArgs e)
      {     if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "")
         {  MessageBox.Show("Va rog completati toate datele"); }
         else
         {  double f1, f2, i, u;
            f1 = Convert.ToDouble(textBox3.Text);            f2 = Convert.ToDouble(textBox4.Text);
            i = Convert.ToDouble(textBox1.Text);             u = Convert.ToDouble(textBox2.Text);
            double A = i * Math.Cos(f1); double A2 = Math.Round(A, 4);
            double B = i * Math.Sin(f1); double B2 = Math.Round(B, 4);
            double A1 = u * Math.Cos(f2); double A3 = Math.Round(A1, 4);
            double B1 = u * Math.Sin(f2); double B3 = Math.Round(B1, 4);
            double Fi = f2 - f1; double Fi1 = Math.Round(Fi, 4);
            double P = i * u * Math.Cos(Fi); double P1 = Math.Round(P, 4);
            double Q = i * u * Math.Sin(Fi); double Q1 = Math.Round(Q, 4);
            double R = P / (i * i); double R1 = Math.Round(R, 4);
            double X = Q / (i * i); double X1 = Math.Round(X, 4);
            double Z = u / i; double Z1 = Math.Round(Z, 4);
            double S = u * i; double S1 = Math.Round(S, 4);
            double Y = i / u; double Y1 = Math.Round(Y, 4);
            label11.Text = A2 + " + j " + B2;
            label12.Text = A3 + " + j " + B3;
            label14.Text = Fi1.ToString();
            label16.Text = P1.ToString();
            label18.Text = Q1.ToString();
            label20.Text = S1.ToString();
            label22.Text = R1.ToString();
            label24.Text = X1.ToString();
            label26.Text = Z1.ToString();
            label28.Text = Y1.ToString();
            label11.Visible = true;
            label12.Visible = true;
            label14.Visible = true;
            label16.Visible = true;
            label18.Visible = true;
            label20.Visible = true;
            label22.Visible = true;
            label24.Visible = true;
            label26.Visible = true;
            label28.Visible = true;          }      }
      private void Button1_Click(object sender, EventArgs e)
      {    foreach (Control X in this.Controls)
            {                if (X is TextBox)                    X.Text = "";              }          }
   }
}
```

***Rezultate:***




***Aplicație:***

Să se proiecteze un formular cu un design mai atractiv.

***Ex. 5. Aplicația calculează valoarea rezistenței echivalente a unui număr de maxim 3 rezistențe legate în serie, considerând rezistențele ca având aceeași valoare***

Formularul realizează calcularea rezistenței serie echivalente pentru una, două sau trei rezistențe (la alegerea utilizatorului) având aceeași valoare și va fi compus din 4 label-uri pentru scrierea unor texte ajutătoare, un meniu de tip *NumericDropDown* unde se va scrie valoarea rezistențelor și 3 butoane de unde utilizatorul va alege câte rezistențe dorește să conțină circuitul său: 1, 2 sau 3. În Figura 13 se poate observa versiunea finală a formularului. La adaugarea butonului de tip *NumericUpDown* s-a specificat că numărul poate avea maxim 5 zecimale din fereastra *Properties,* la *Data – DecimalPlaces*.

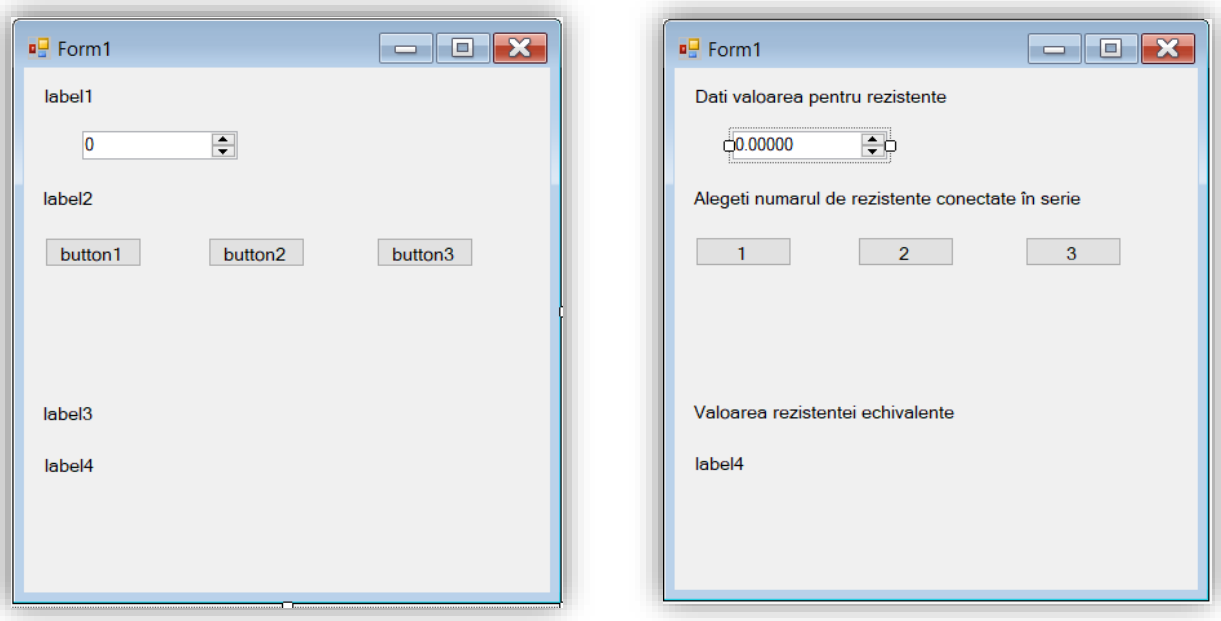



**Fig. 13.** Formularul creat pentru aplicatie

Cu dublu click pe butonul 3 se deschide fereastra de editare unde se completeaza codul care generează atât circuitul cât și soluția numerică:

```
private void Button3_Click(object sender, EventArgs e)
    {
       double x, result;
       System.Drawing.Graphics gr = this.CreateGraphics();
       gr.Clear(this.BackColor);
       System.Drawing.Rectangle dr = new System.Drawing.Rectangle(70, 120, 50, 20);
       gr.DrawRectangle(System.Drawing.Pens.Red, dr);
       Pen cr = new Pen(Color.Red);
       cr.Width = 1;
       Point p1 = new Point(120, 130);
       Point p2 = new Point(130, 130);
       gr.DrawLine(cr, p1, p2);
       System.Drawing.Rectangle dr2 = new System.Drawing.Rectangle(130, 120, 50, 20);
       gr.DrawRectangle(cr, dr2);
       Point p3 = new Point(180, 130);
       Point p4 = new Point(190, 130);
       gr.DrawLine(cr, p3, p4);
       System.Drawing.Rectangle dr3 = new System.Drawing.Rectangle(190, 120, 50, 20);
       gr.DrawRectangle(cr, dr3);
       x = System.Convert.ToDouble(numericUpDown1.Value);
       result = x + x + x;
       label4.Text = result.ToString();
    }
```

Se poate observa că prima parte din cod este scrisă pentru a creea un spațiu grafic și pentru a desena 3 dreptunghiuri și liniile care le unesc, alcătuind astfel circuitul analizat. Ultima parte din secventa de cod reprezinta  calculul efectiv al valorii rezistenței echivalente.

Pentru cel de al doilea buton diferențele în cod sunt minore:

```
private void Button2_Click(object sender, EventArgs e)
    {
       double x, result;
       System.Drawing.Graphics gr = this.CreateGraphics();
       gr.Clear(this.BackColor);
       System.Drawing.Rectangle dr = new System.Drawing.Rectangle(70, 120, 50, 20);
       gr.DrawRectangle(System.Drawing.Pens.Red, dr);
       Pen cr = new Pen(Color.Red);
       cr.Width = 1;
       Point p1 = new Point(120, 130);
       Point p2 = new Point(130, 130);
       gr.DrawLine(cr, p1, p2);
       System.Drawing.Rectangle dr2 = new System.Drawing.Rectangle(130, 120, 50, 20);
       gr.DrawRectangle(cr, dr2);
        x = System.Convert.ToDouble(numericUpDown1.Value);
       result = x + x ;
       label4.Text = result.ToString();
    }
```

Pentru butonul 1 codul este:

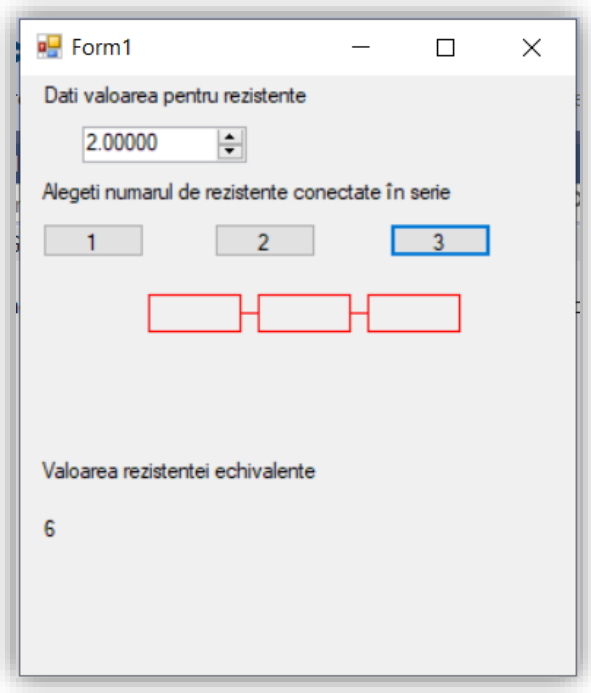
```
private void Button1_Click(object sender, EventArgs e)
      {
          double x, result;
          System.Drawing.Graphics gr = this.CreateGraphics();
          gr.Clear(this.BackColor);
          System.Drawing.Rectangle dr = new System.Drawing.Rectangle(70, 120, 50, 20);
          gr.DrawRectangle(System.Drawing.Pens.Red, dr);
          Pen cr = new Pen(Color.Red);
          cr.Width = 1;
          x = System.Convert.ToDouble(numericUpDown1.Value);
          result = x ;
          label4.Text = result.ToString();
       }
```

**Varianta in C#**

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace WindowsFormsApp6
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void Button3_Click(object sender, EventArgs e)
      {   double x, result;
         System.Drawing.Graphics gr = this.CreateGraphics();
         gr.Clear(this.BackColor);
         System.Drawing.Rectangle dr = new System.Drawing.Rectangle(70, 120, 50, 20);
         gr.DrawRectangle(System.Drawing.Pens.Red, dr);
         Pen cr = new Pen(Color.Red);
         cr.Width = 1;
         Point p1 = new Point(120, 130);
         Point p2 = new Point(130, 130);
         gr.DrawLine(cr, p1, p2);
         System.Drawing.Rectangle dr2 = new System.Drawing.Rectangle(130, 120, 50, 20);
         gr.DrawRectangle(cr, dr2);
         Point p3 = new Point(180, 130);
         Point p4 = new Point(190, 130);
         gr.DrawLine(cr, p3, p4);
         System.Drawing.Rectangle dr3 = new System.Drawing.Rectangle(190, 120, 50, 20);
         gr.DrawRectangle(cr, dr3);
         x = System.Convert.ToDouble(numericUpDown1.Value);
         result = x + x + x;
         label4.Text = result.ToString();
      }
      private void Button2_Click(object sender, EventArgs e)
      {   double x, result;
         System.Drawing.Graphics gr = this.CreateGraphics();
         gr.Clear(this.BackColor);
         System.Drawing.Rectangle dr = new System.Drawing.Rectangle(70, 120, 50, 20);
         gr.DrawRectangle(System.Drawing.Pens.Red, dr);
         Pen cr = new Pen(Color.Red);
```

```
    cr.Width = 1;
    Point p1 = new Point(120, 130);
    Point p2 = new Point(130, 130);
    gr.DrawLine(cr, p1, p2);
    System.Drawing.Rectangle dr2 = new System.Drawing.Rectangle(130, 120, 50, 20);
    gr.DrawRectangle(cr, dr2);
     x = System.Convert.ToDouble(numericUpDown1.Value);
    result = x + x ;
    label4.Text = result.ToString();       }
private void Button1_Click(object sender, EventArgs e)
{
      double x, result;
      System.Drawing.Graphics gr = this.CreateGraphics();
      gr.Clear(this.BackColor);
      System.Drawing.Rectangle dr = new System.Drawing.Rectangle(70, 120, 50, 20);
      gr.DrawRectangle(System.Drawing.Pens.Red, dr);
      Pen cr = new Pen(Color.Red);
      cr.Width = 1;
      x = System.Convert.ToDouble(numericUpDown1.Value);
      result = x ;
      label4.Text = result.ToString();          }      }   }
```

***Rezultate:***




***Aplicație:***

Să se creeze un formular pentru calculul rezistenței paralel echivalente a maxim 4 rezistențe de aceeași valoare si respective de valori diferite.

**PROBLEME PROPUSE**

***1. Sa se implementeze în C# o aplicație de tip Windows desktop care realizează conversia unităților de măsură pentru lungime, masă și energie***

***2. Să se creeze un formular pentru calculul capacității echivalente serie și paralel a maxim 4 condensatoare de valori diferite, valorile fiind definite de utilizator.***

# Capitolul 15
# Realizarea aplicațiilor în C/C++ utilizând Arduino

În acest capitol este prezentat modul de proiectare şi implementare a unor aplicaţii realizate cu placi Arduino şi mediul de programare Arduino IDE care are la bază limbajele C/C++.

## CONSIDERAȚII TEORETICE

Arduino este o placă de circuite programabilă open source care poate fi integrată într-o mare varietate de proiecte simple sau de complexitate mai ridicată. Această plăcuță conține un microcontroler care se poate programa să simtă și să controleze obiecte. Datorită faptului că răspunde la senzori și date de intrare, Arduino poate să interacționeze cu o multitudine de elemente care dau date de ieșire precum leduri, motoare, și display-uri. Datorită flexibilității și a costurilor reduse, Arduino a devenit o alegere populară printre producătorii sau designerii care doresc să creeze proiecte hardware interactive.

Plăcuțele Arduino au fost introduse prima dată în Italia în 2005 de către Massimo Banzi ca o modalitate prin care de a creea astfel de proiecte hardware cu un cost redus. Deoarece plăcuța este open source, oricine poate creea o clonă a acesteia, dar doar plăcuțele care conțin în nume Arduino sunt cele oficiale.

Una dintre cele mai populare plăcuțe este Arduino Uno. Chiar dacă nu a fost prima plăcuță creată, rămâne cea mai folosită și mai documentată de pe piață și este ușor de utilizat. Pentru programare și dezvoltare de programe este asigurat un mediu de dezvoltare integrat (IDE), asigurând suport pentru limbajul C și C++. În Fig. 1 se pot observa componentele de bază ale unei astfel de plăcuțe numerotate de la 1-15.

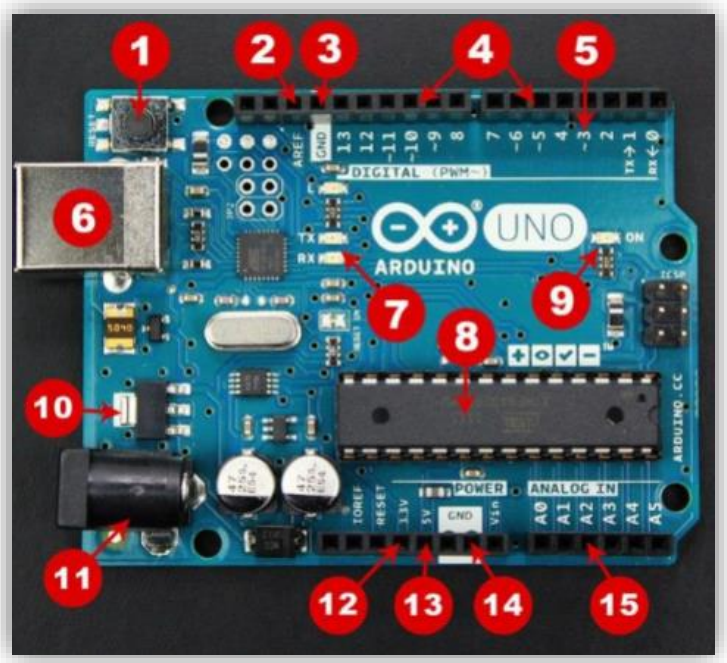



**Fig. 1.** Componentele unei plăcuțe Arduino Uno

1. **Butonul de Reset** – Acest buton va restarta orice cod care este încărcat pe plăcuța Arduino.
2. **AREF** – denumirea vine de la referință analogică și este folosită pentru a seta o tensiune externă de referință
3. **Pini de pământare** – sunt mai mulți pini de pământare în cadrul Arduino și toți funcționează la fel
4. **Intrare/ieșire digitală** – Pinii 0-13 sunt pini digitali de intrare-ieșire
5. **PWM** – Pinii care sunt marcați cu simbolul (~) pot simula intrarea analogică
6. **Conexiunea USB** – folosită pentru alimentarea plăcuței Arduino și pentru a uploada programele
7. **TX/RX** – LED-uri pentru indicarea transmiterii și recepționării datelor
8. **Microcontroler ATmega** – acesta este "creierul" plăcuței și locul unde toate programele sunt stocate
9. **LED-ul care indică alimentarea** – acest LED se aprinde de fiecare data când plăcuța este legată la o sursă de alimentare
10. **Regulator de tensiune** – acesta controlează tensiunea care intră în placa Arduino
11. **Power barrel jack de curent continuu** - acesta este folosit pentru a alimenta plăcuța Arduino cu o alimentare
12. **Pin de 3.3 V** – acest pin alimentează proiectele cu 3.3 V
13. **Pin de 5 V** – acest pin alimentează proiectele cu 5 V
14. **Pini de pământare** – sunt mai mulți pini de pământare în cadrul Arduino și toți funcționează la fel
15. **Pini analogici** – acești pini pot să citească semnalul de la un senzor analogic și să îl convertească înnt-un semnal digital

Plăcuța Arduino Uno are nevoie de o sursă de alimentare pentru a funcționa și poate fi alimentat în mai multe moduri. Majoritatea utilizatorilor conectează plăcuța direct la calculator/laptop via un cablu USB. Dacă se dorește ca proiectul să fie mobil, se consider folosirea unei baterii de 9 V. Ultima metodă este folosirea unei alimentări de curent alternative de 9 V. Toate aceste tipuri de alimentare se pot observa în Fig. 2.

**Fig. 2.** Tipuri de alimentare pentru plăcuțele Arduino

Un alt element foarte important atunci când se lucrează cu plăcuțe Arduino sunt breadboard-urile. Aceste dispozitive permit crearea de proiecte Arduino fără a fi necesară sudarea permanentă a circuitului. Folosirea breadboard-urilor permmite crearea de prototipuri temporare și experimentarea cu circuite diferite.

În interiorul orificiilor din carcasa de plastic sunt clipsuri metalice care sunt conectate unele cu celelalte prin fâșii de material conductor. Breadboard-ul nu este alimentat și are nevoie de alimentare de la plăcuța Arduino folosind fire de legătură. Aceste fire sunt de asemenea folosite pentru a creea un circuit pri conectarea de rezistoare, butoane sau alte componente.

Odată ce circuitul a fost creat pe un breadbrd, se va dori uploadarea programului (denumit sketch). Acest program este un set de instrucțiuni care spun plăcuței ce instrucțiuni sunt necesare pentru a funcționa. Un Arduino poate să funcționeze doar cu un program o dată. Software-ul folosit pentru a creea programe Arduino este denumit IDE(Mediu de Dezvoltare Integrat). Acest software se poate downloada gratis de la adresa https://www.arduino.cc/en/Main/Software. Orice program Arduino are două părți principale:

**void setup()** -setează lucrurile care trebuie făcute o singură dată
**void loop()** – conține instrucțiuni care sunt repetate până când este oprită plăcuța.

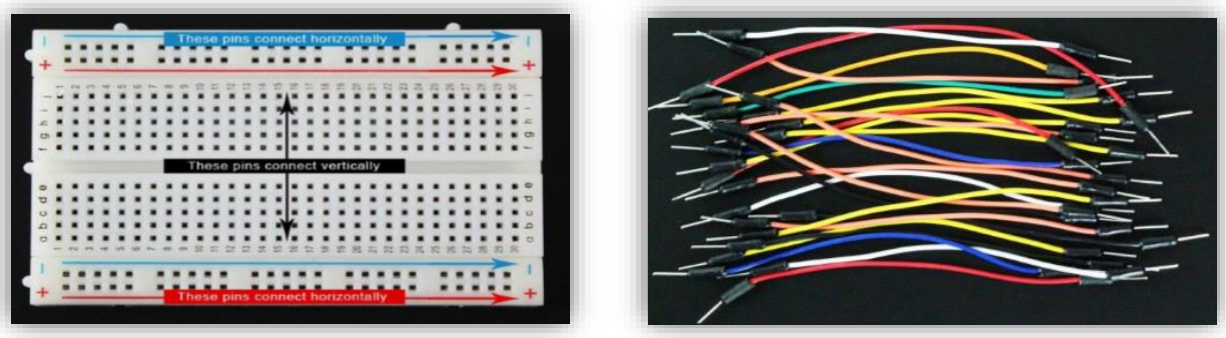



**Fig. 3.** Breadboard și fire de legătură

Se găsesc o gamă largă de senzori compatibili cu plăcuțele Arduino. Dintre cei mai uzuali sunt: senzori de distanță, senzori de mișcare PIR, senzor de luminozitate, senzori de proximitate și de presiune, senzori de accelerație, senzori pentru detecția sunetului, senzori pentru determinarea gesturilor, senzori de umiditate și temperatură, etc.

Odată ce componentele necesare pentru proiecte sunt disponibile, se instalează software-ul Arduino IDE care se poate descarca gratuit de pe site-ul: http://www.arduino.cc/en/main/software. Interfața acestui mediu de programare este prezentată în Fig.6 iar componentele sunt descrise mai jos:

1. **Bara de meniu** – Oferă acces la uneltele necesare pentru crearea și salvarea programelor Arduino
2. **Butonul de verificare** – compilează codul și îl verifică de erori de ortografie și sintaxă
3. **Butonul de upload(încărcare)** – trimite codul plăcuței care este conectată. Ledurile de pe plăcuță vor clipi rapid cand se încarcă un cod
4. **Program (sketch) nou** – deschide o nouă fereastră goală
5. **Numele programului** – când programul este salvat, numele lui este scris aici
6. **Deschide un program existent** – permite deschiderea unui program existent sau a unui exemplu dintre cele oferite de program
7. **Salvare program** – acest buton salvează programul care este deschis

8. **Serial monitor** – când plăcuța este conectată, acesta va prezenta informația oferită de Arduino
9. **Zona de cod** – această zonă este folosită pentru editarea codului care spune plăcuței ce are de făcut
10. **Zona de mesaje** – această zonă oferă informații despre salvarea programului, compilarea codului, erori și altele
11. **Consolă pentru text** – oferă detalii despre un mesaj de eroare, dimensiunea programului care a fost compilat și alte informații
12. **Plăcuța și port serial** – spune ce plăcuță este folosită în montaj și la ce port serial este conectată.

Odată ce plăcuța este conectată la calculator, se va alege tipul de plăcuță utilizat prin accesarea ***Tools->Board->Arduino Uno.***

Următorul pas este alegerea portului care este utilizat pentru conectarea plăcuței. Pentru a selecta acest port se va accesa ***Tools->Port*** și se alege portul dorit.

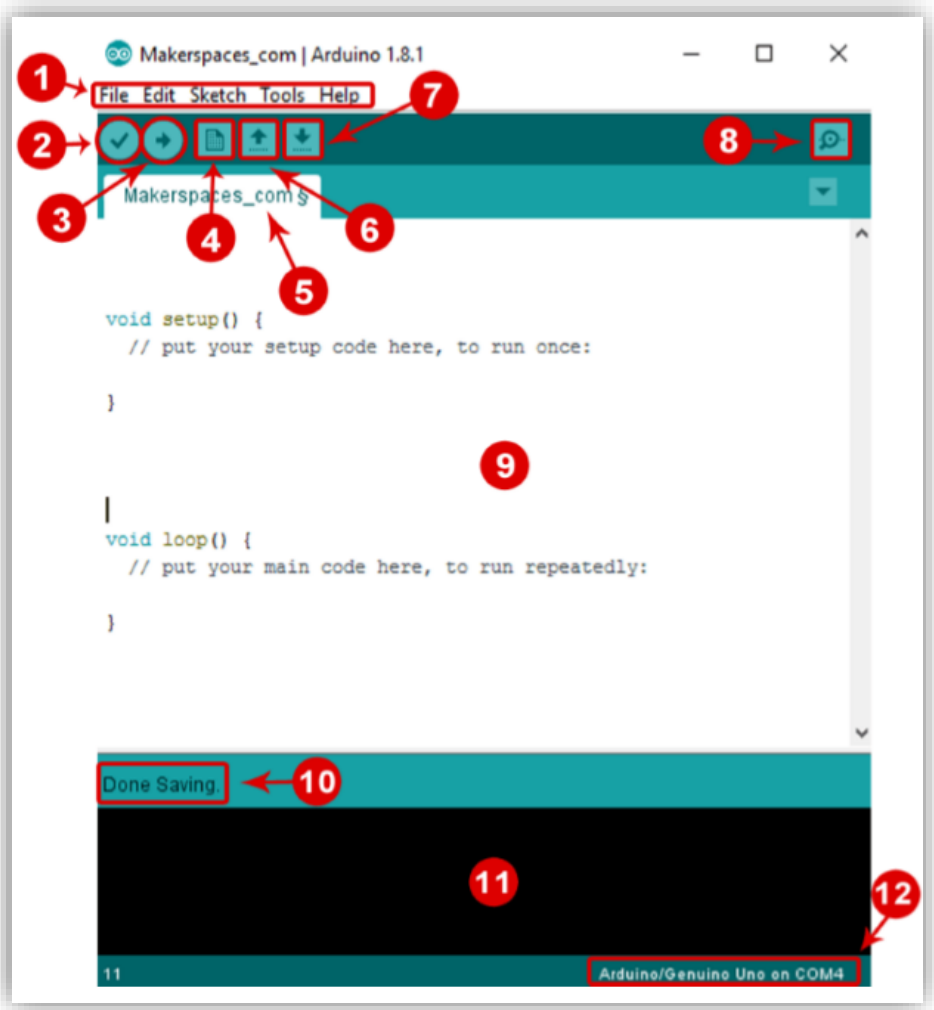



**Fig. 4.** Interfața software-ului Arduino IDE

## PROBLEME REZOLVATE

**Ex. 1**. Aplicația prezintă crearea unui circuit simplu care realizează pornirea și oprirea succesivă a unui led. Circuitul creat este inclus alături de alte exemple utile pentru creearea de aplicații similare pe site-ul https://www.tinkercad.com/learn/circuits . Legăturile realizate sunt evidențiate în Fig. 5. Deoarece LED-ul are un anod și un catod se va plasa piciorul mai scurt al LED-ului înspre rezistență.

Pentru acest proiect sunt necesare:

- ✓ plăcuță Arduino Uno
- ✓ Breadboard
- ✓ Fire
- ✓ Cablu USB
- ✓ LED
- ✓ Rezistență de 220 Ohmi

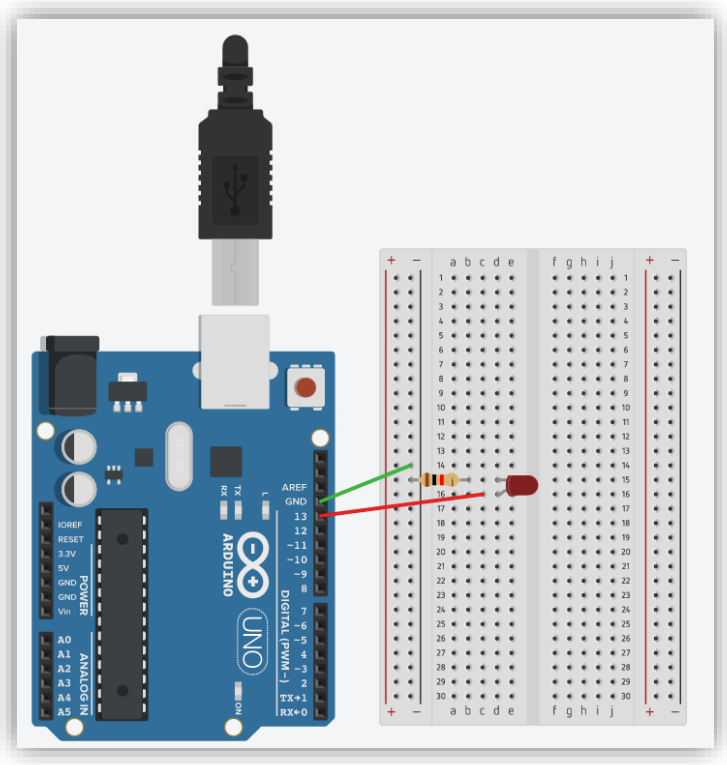


**Fig. 5.** Circuitul creat pentru proiectul 1

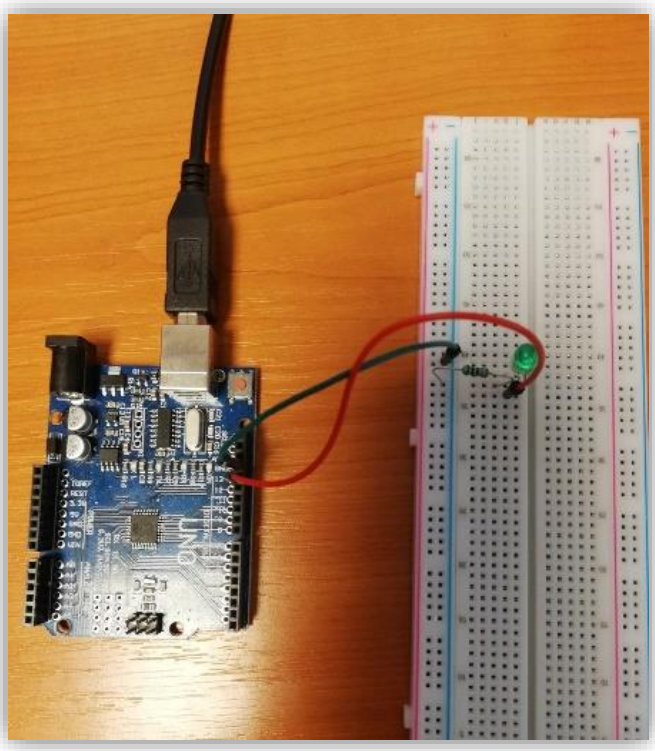
Pentru a deschide aplicația pentru acest proiect se accesează fișierul predefinit ca exemplu: ***File->Examples->Basics->Blink***. Acesta conţine codul de mai jos care va fi afişat pe ecran. Se va apăsa butonul de verificare situat în partea din stânga sus a ferestrei IDE pentru a verifica dacă sunt erori.

Dupa compilare se va afişa mesajul “Done compiling” si se va apăsa butonul de încărcare al programului. Astfel LED-ul va clipi, pornindu-se pentru o secundă apoi oprindu-se tot pentru o secundă într-o buclă care se va termina la deconectarea plăcuței. Arduino măsoară timpul în milisecunde și 1000 milisecunde=1 secundă.

**Cod Arduino**

```
void setup()
 {//  se inițializează pinul digital  LED_BUILTIN ca o variabilă de ieșire
  pinMode(LED_BUILTIN, OUTPUT);
}
// funcția buclă care funcționează repetitiv
void loop()
{  digitalWrite(LED_BUILTIN, HIGH);   // pornește LED-ul (HIGH este nivelul de tensiune)
  delay(1000);                  // se așteaptă o secundă
  digitalWrite(LED_BUILTIN, LOW);    // oprește LED-ul prin setarea tensiunii la LOW
  delay(1000);                  // se așteaptă o secundă
}
```

***Rezultate:***



***Aplicație:***

*Modificați codul astfel încât delay-ul LED-ului să fie 200.*

**Ex.2.** Aplicația continuă aplicația anterioară prin adăugarea unui buton în circuitul construit anterior. Acest buton este o componentă electrică care va condiționa funcționarea circuitului. Când butonul nu este apăsat, circuitul este oprit, altfel este pornit. Circuitul creat în cadrul acestui proiect se poate observa în Fig. 6.

Pentru acest proiect sunt necesare:

- ✓ O plăcuță Arduino Uno
- ✓ Breadboard
- ✓ Fire
- ✓ Cablu USB
- ✓ LED
- ✓ buton
- ✓ Rezistență de 220 Ohmi
- ✓ Rezistență de 10 kOhmi

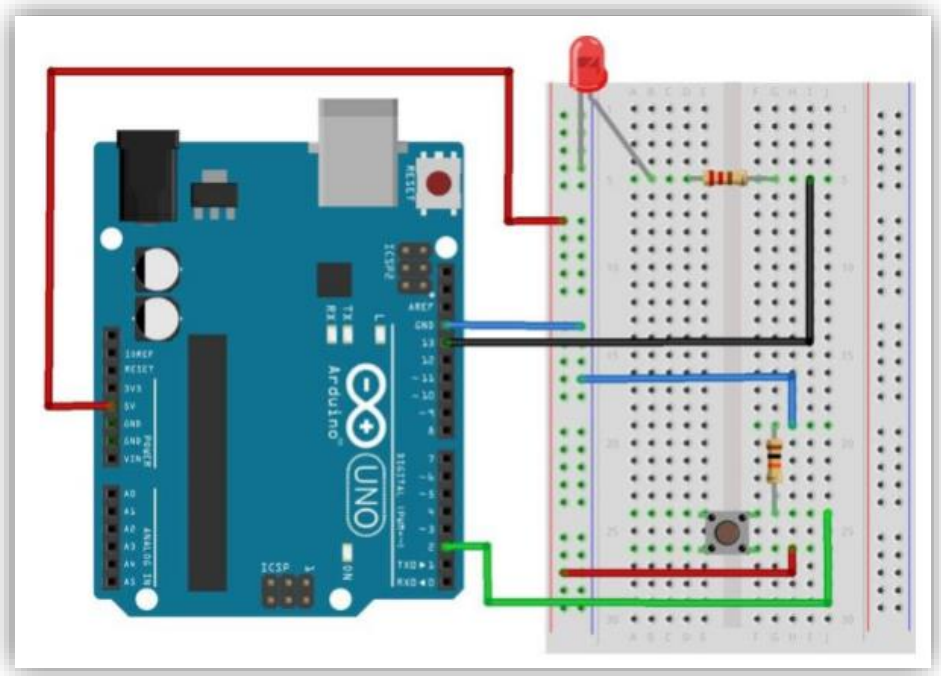


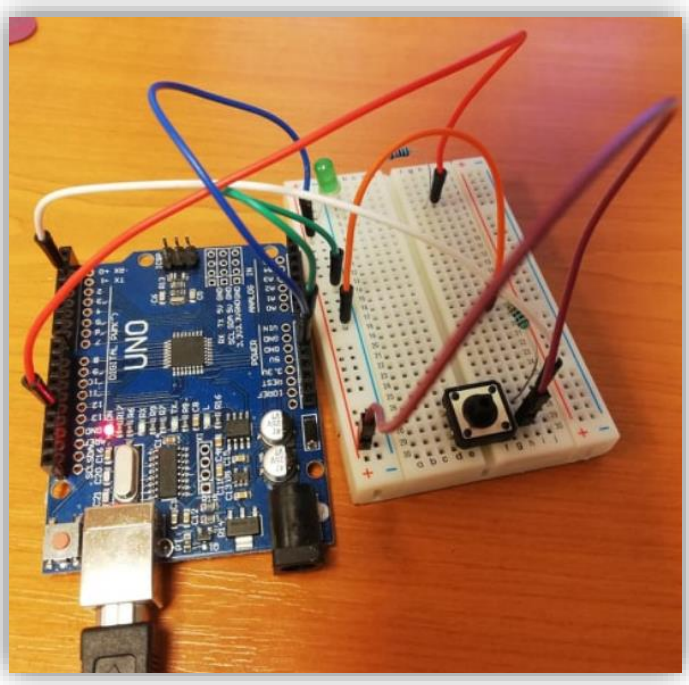
**Fig. 6.** Circuitul creat pentru aplicatia Ex.2

Codul pentru acestasta aplicaţie este predefinit în ***File->Examples->Digital->Button***.

**Cod Arduino**

```
const int buttonPin = 2;     // numărul pinului pentru buton
const int ledPin =  13;      // numărul pinului pentru LED
// variabilele se vor schimba:
int buttonState = 0;         // variabila pentru citirea statusului butonului
void setup() {
  // initializarea pinului pentru LED ca variabilă de ieșire:
  pinMode(ledPin, OUTPUT);
  // initializarea pinului pentru buton ca variabilă de intrare:
  pinMode(buttonPin, INPUT);}
void loop()
{  // citirea valorii corespunzătoare stării butonului:
  buttonState = digitalRead(buttonPin);
  // se verifică dacă butonul este apăsat. Dacă este, atunci buttonState este HIGH:
  if (buttonState == HIGH) {
    // pornește LED-ul:
    digitalWrite(ledPin, HIGH);
  } else {
    // oprește LED-ul:
    digitalWrite(ledPin, LOW);  } }
```

***Rezultate:***




***Aplicație:***

*Modificați pinii care sunt atribuiți butonului și LED-ului și rescrieți codul.*

În cazul în care se întâmpină anumite probleme se vor verifica următoarele:

- Se verifică dacă LED-ul este funcțional. Se folosește o baterie de 3V și se conectează piciorul lung al LED-ului la (+) și piciorul scurt la (-)
- Se verifică dacă piciorul lung al LED-ului este conectat corect. Piciorul lung la (+) și cel scurt la (-)
- Se verifică dacă Arduino IDE arată corect plăcuța folosită. Aceasta se selectează accesând *Tools->Board*
- Se verifică dacă Arduino IDE arată corect portul utilizat. Acesta se selectează accesând *Tools->Port*
- Se verifică dacă conexiunea componentelor este una sigură.

**Ex.3.** Aplicația descrie implementarea unui dispozitiv de măsurare a capacității condensatoarelor.

Determinarea valorii capacității unor condensatoare poate fi o sarcină dificilă dacă nu exista un multimetru digital pentru a le testa. Orice dispozitiv de măsurare a capacității construit prin utilizarea placuţelor Arduino se bazează pe o proprietate a circuitelor rezistor-capacitor (RC) – constanta de timp.

Constanta de timp a circuitului RC este definită ca timpul necesar pentru ca tensiunea din condensator să atingă 63,2% din tensiunea sa când este complet încărcat. Condensatoarele mai mari necesită mai mult timp pentru a se încărca și, prin urmare, vor crea constante de timp mai mari.

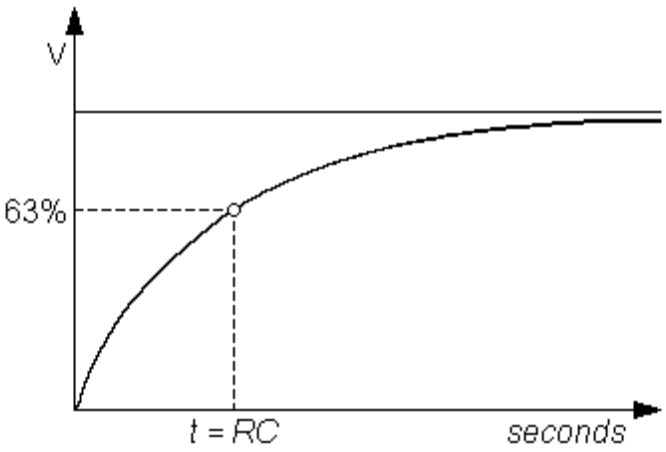



**Fig. 7.** Definirea constantei de timp

Capacitatea într-un circuit RC este legată de constanta de timp prin ecuația :

$$TC=R*C \quad (1)$$

unde:

TC-constanta de timp pentru condensator (sec)
R-rezistența din circuit (Ohmi)
C-capacitatea unui condensator (F)

Prin rearanjarea ecuației, capacitatea se calculează astfel:

$$C = \frac{TC}{R} \quad (2)$$

Fiecare dispozitiv de calcul a capacității care folosește Arduino are integrat un circuit RC cu valori ale rezistenței cunoscute și o valoare a necunoscută a condensatorului. Arduino va măsura tensiunea la condensator și va înregistra timpul necesar pentru a atinge 63,2% din tensiunea când este complet încărcat (constanta de timp).

Deoarece valoarea rezistenței este deja cunoscută, formula de mai sus se va implementa într-un program care calculează capacitatea necunoscută. Acuratețea rezultatelor obținute este mai bună pentru condensatoarele cu valori între 0,1 și 3900 µF.

Circuitul creat pentru aceasta aplicație este ilustrat în Fig.8. Pentru a calcula capacitatea condensatorului se realizează un circuit alcătuit din următoarele componente:

- ✓ O plăcuță Arduino Uno
- ✓ Breadboard
- ✓ Fire
- ✓ Cablu USB
- ✓ Rezistență de 220 Ohmi
- ✓ Rezistență de 10 kOhmi
- ✓ Un condensator cu capacitate necunoscută

Codul pentru acest circuit este prezentat în continuare, iar rezultatul din Serial Monitor-ul de care dispune software-ul Arduino IDE este ilustrat la sectiunea Rezultate. Prima coloană afişată conține constanta de timp a condensatorului, iar cea de a doua-a coloană reprezintă capacitatea măsurată.

Unitățile se vor schimba automat din microFarad în nanoFarad. Se va observa că va fi necesar un timp mai îndelungat ca Arduino să afișeze o citire dacă valoarea capacității e mai mare. Acest lucru se întâmplă deoarece condensatoarele cu capacități mai mari au nevoie de un timp mai îndelungat pentru a ajunge la 63.2% din tensiunea lor când sunt încărcate.

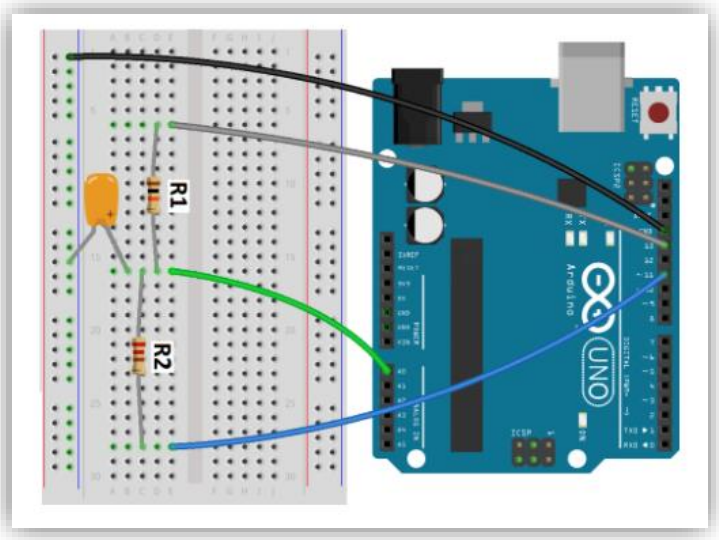
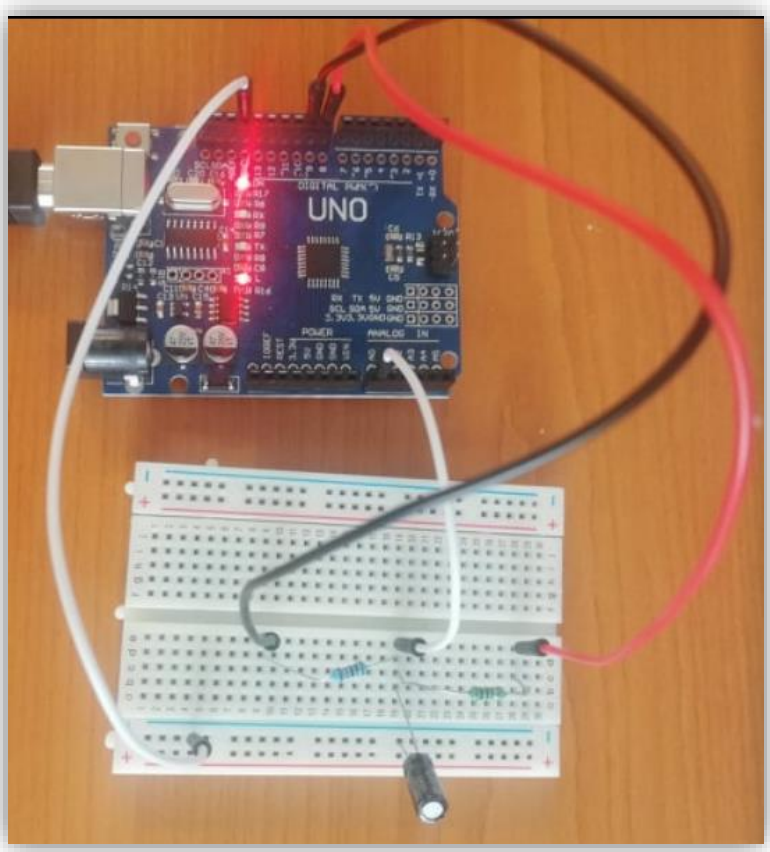



**Fig. 8.** Circuitul creat pentru aplicație

**Cod Arduino**

```
#define analogPin 0 // pinul analogic pentru măsurarea tensiunii condensatorului
#define Pinincarcare 9 // pinul pentru încărcarea condensatorului – conectat la un capăt a rezistenței de încărcare
#define Pindescarcare   8 // pinul de descărcare a condensatorului
#define resistorValue  10000.0F  // se schimbă la valoarea rezistenței care este folosită
                                 // F spune compilatorului ca este un număr real cu zecimale
unsigned long timpdeinceput;
unsigned long timpscurs;
float microFarads;
float nanoFarads;
void setup(){
  pinMode(Pinincarcare, OUTPUT);  // se setează Pinincarcare ca și OUTPUT
  Serial.begin(9600);  }// inițializarea transmisiei serial pentru debugging
void loop(){
  digitalWrite(Pinincarcare, HIGH);   // setează Pinincarcare HIGH și condensatorul se încarcă
  timpdeinceput = millis(); //returnează numărul în millisecunde care au trecut de cand plăcuța Arduino a pornit programul curent
  while(analogRead(analogPin) < 648){      // 647 este 63.2% din 1023, care corespunde unei tensiuni complete
  }
timpscurs= millis() - timpdeinceput; // convertește millisecundele în secunde ( 10^-3 ) și Farad în microFarad ( 10^6 )
  microFarads = ((float)timpscurs / resistorValue) * 1000;
  Serial.print(timpscurs);
  Serial.print(" mS    ");
  if (microFarads > 1){
    Serial.print((long)microFarads);
    Serial.println(" microFarads");       }
  else{
 // dacă valoarea este mai mica decât 1 microFarad, convertește în nanoFarads (10^-9 Farad).
    nanoFarads = microFarads * 1000.0;      // multiplică cu 1000 pentru a concerti în nanoFarads (10^-9 Farads)
    Serial.print((long)nanoFarads);
    Serial.println(" nanoFarads");
    delay(500);   }
  digitalWrite(Pinincarcare, LOW);          // se setează Pinincarcare  LOW
  pinMode(Pindescarcare, OUTPUT);          // se setează Pindescarcare ca OUTPUT
  digitalWrite(Pindescarcare, LOW);         // se setează Pindescarcare LOW
  while(analogRead(analogPin) > 0){        // se așteaptă descărcarea complete a condensatorului
  }  pinMode(Pindescarcare, INPUT);          // se setează Pindescarcare  înapoi ca INPUT
}
```

***Rezultate:***




***Aplicație:***

*Să se modifice aplicaţia astfel încât să se aleagă condensatoare de diferite capacități și să se observe acuratețea cu care se măsoară valoarea lor.*

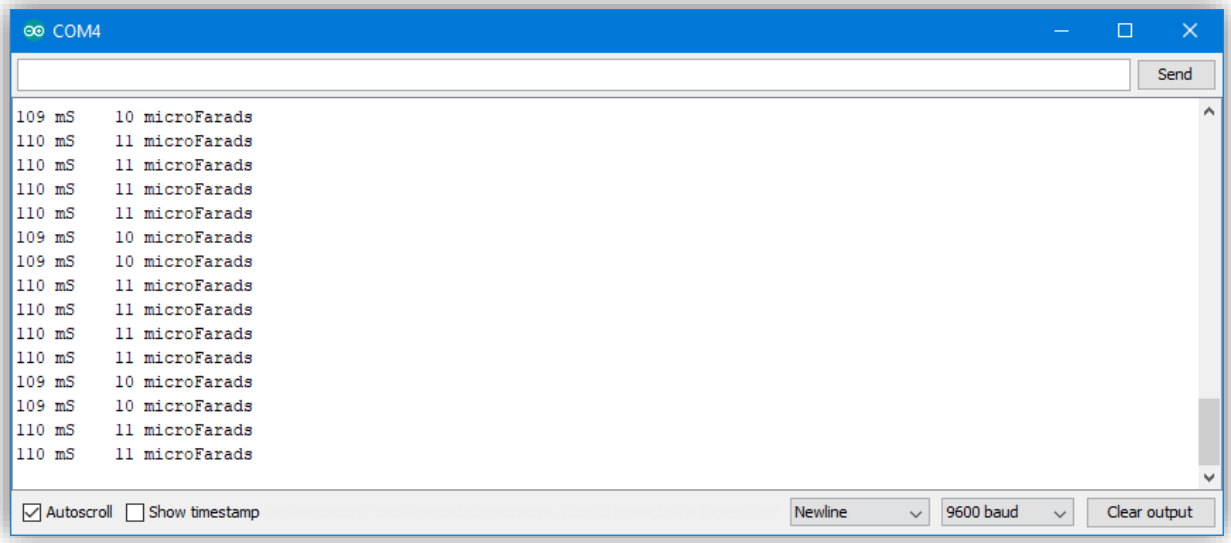



**Fig. 9.** Rezultatele afișate de Monitorul Serial pentru afișarea capacității unui condensator de 10 µF

**Ex. 4 .** Aplicația realizează afișarea temperaturii și a umidității dintr-o încăpere cu ajutorul unei plăci Arduino. În montaj (Fig.10) se va integra un senzor de temperatură și umiditate și valorile se vor afișa pe monitorul serial oferit de Arduino IDE. Astfel, pentru circuit sunt necesare:

- ✓ O plăcuță Arduino Uno
- ✓ Breadboard
- ✓ Fire
- ✓ Cablu USB
- ✓ DHT 11 senzor de temperatură și umiditate

Pentru a citi informațiile furnizate de senzor se va încărca codul de mai jos si după aceea se va accesa monitorul serial oferit de Arduino IDE prin ***Tools->Serial Monitor*** sau Ctrl+Shift+M.

Este necesară încărcarea bibliotecii pentru senzor care se descarcă de pe internet și se încarcă astfel ***Sketch -> Include Library -> Add .ZIP Library***.

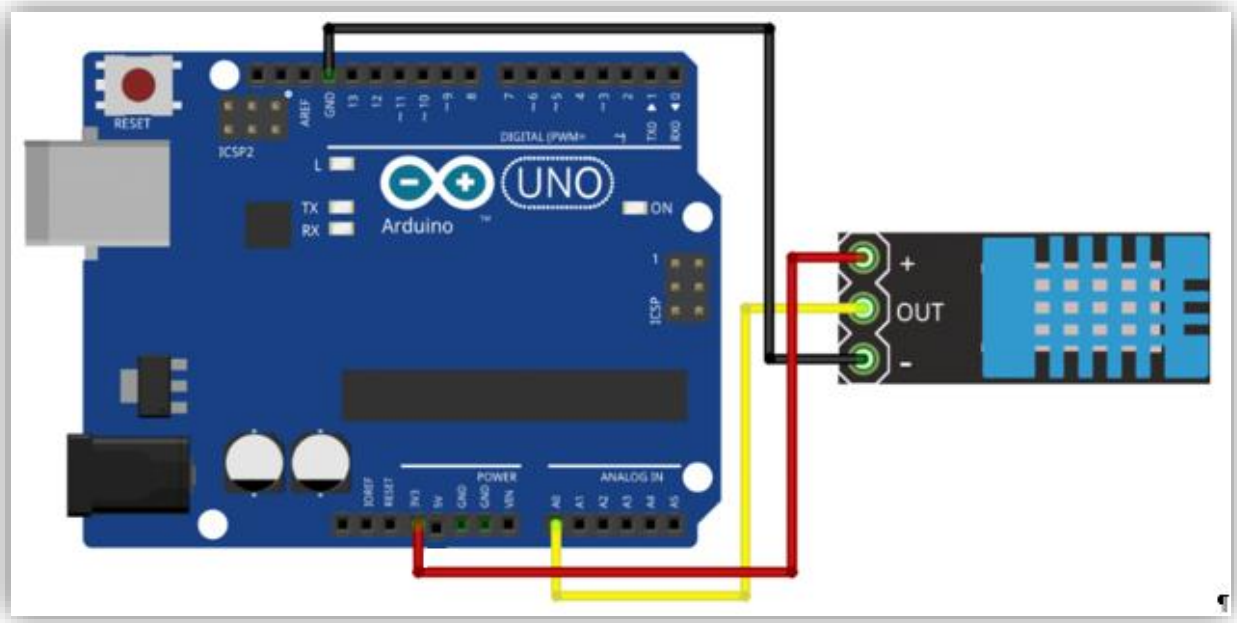


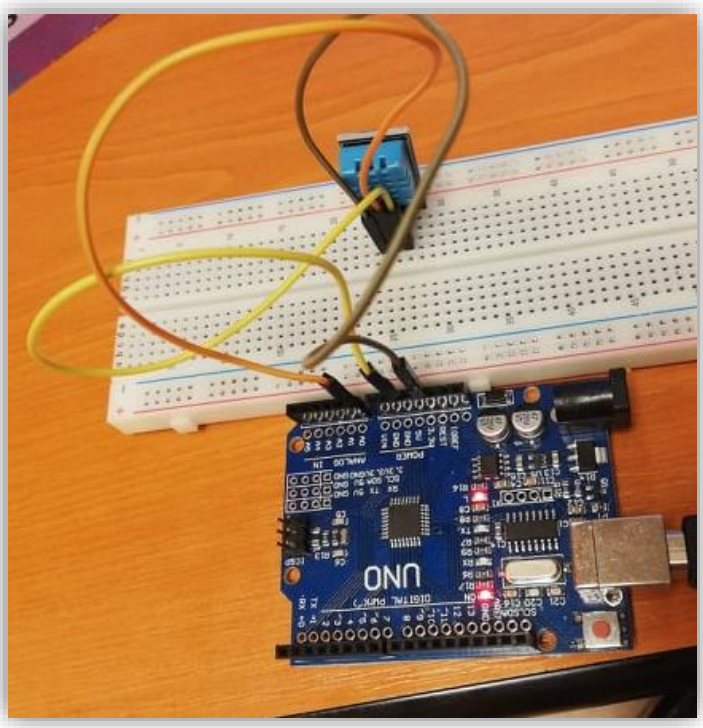

**Fig. 10.** Circuitul creat pentru aplicatie

Codul pentru citirea valorilor returnate de către senzor:

**Cod Arduino**

```
#include "dht.h"
#define dht_apin A0 // Pinul analigic la care este conectat senzorul
dht DHT;
void setup()
{  Serial.begin(9600);
  delay(500);
  Serial.println("DHT11 Senzor de umiditate si temperatura\n\n");
  delay(1000);//se așteaptă până la accesarea senzorului
 }
void loop()
{    DHT.read11(dht_apin);
    Serial.print("Umiditate = ");
    Serial.print(DHT.humidity);
    Serial.print("%  ");
    Serial.print("Temperatura = ");
    Serial.print(DHT.temperature);
    Serial.println("C  ");
       delay(5000);//se așteaptă 5 secunde până la a accesa din nou senzorul.
   //timpul ar trebui să fie mai mare de 2 secunde
 }
```

***Rezultate:***




***Aplicație:***

*Suflați asupra senzorului pentru a observa diferențele în funcționare*

Rezultatul obținut se poate observa pe Monitorul serial după cum se poate observa în Fig. 11.

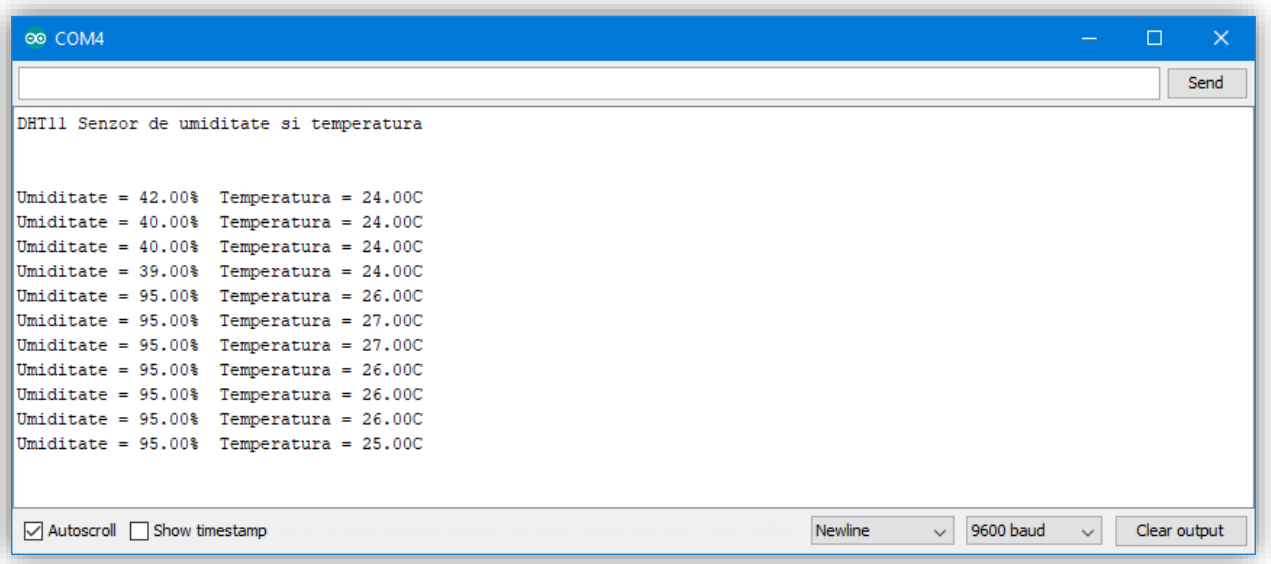


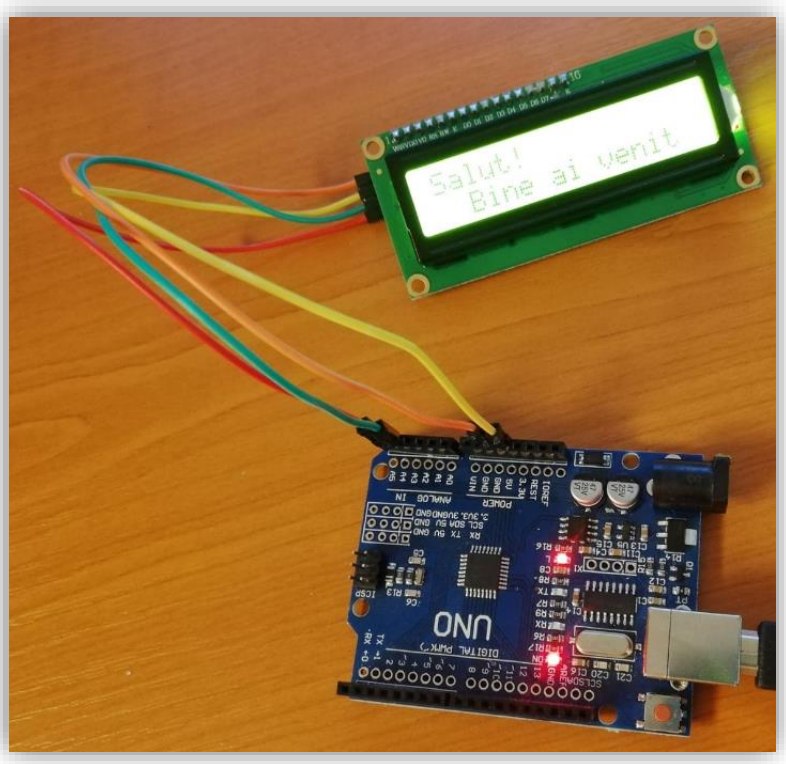

**Fig. 11.** Rezultatele obținute pentru senzorul de temperatură și umiditate

Se va conecta în continuare un LCD 16x2 Ic2 la montajul realizat anterior. Acesta se conectează așa cum este ilustrat în figura prezentată la Rezultate. Pentru a afișa mesajul “Salut! Bine ai venit” Se va scrie codul:

**Cod Arduino**

```
// Se include librăriile:
#include <Wire.h> // Librărie pentru comunicarea cu I2C
#include <LiquidCrystal_I2C.h> // Librărie pentru LCD
LiquidCrystal_I2C lcd (0x27,2,1,0,4,5,6,7,3,POSITIVE);// Se setează adresa pentru LCD-ul I2C
void setup() {
  // Initiate the LCD:
lcd.begin(16,2);
delay(500);
  lcd.clear();}
void loop()
  lcd.setCursor(0, 0); // Setează cursorul pe prima linie și prima coloană
  lcd.print("Salut!");
  lcd.setCursor(2, 1); //Setează cursorul pe a 3-a coloană și al doilea rând
  lcd.print("Bine ai venit");}
```

***Rezultate:***




***Aplicație:***

*Să se modifice mesajul afișat și să se alinieze la dreapta*

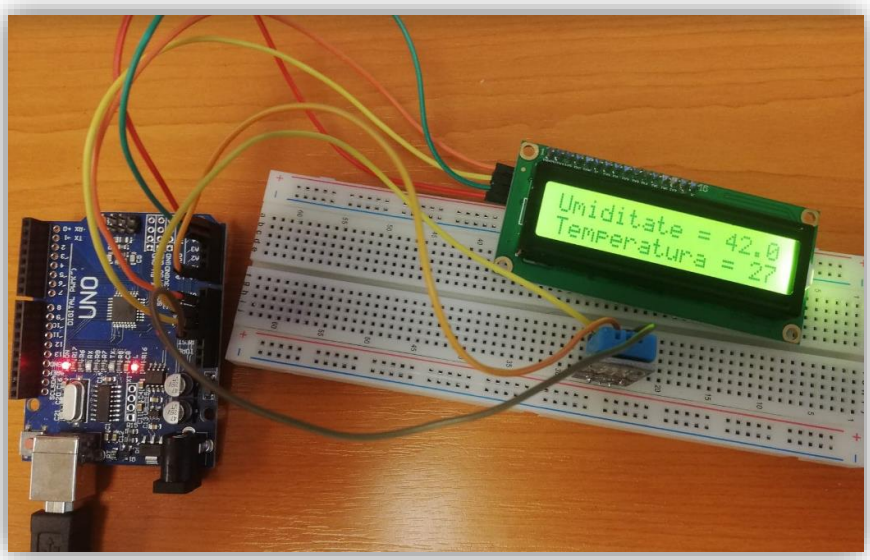
Cu ajutorul codului de mai jos se vor putea observa valorile obținute pentru temperatură și umiditate pe LCD:

**Cod Arduino**

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "dht.h"
#define dht_apin A0
 dht DHT;
 LiquidCrystal_I2C lcd (0x27,2,1,0,4,5,6,7,3,POSITIVE);
void setup(){
 lcd.begin(16,2);
 delay(500);//Delay to let system boot
 lcd.print("DHT11 umid/temp");
  delay(5000);//Wait before accessing Sensor
 lcd.clear();
}
void loop(){
DHT.read11(dht_apin);
 lcd.setCursor(0,0);
 lcd.print("Umiditate = ");
 lcd.print(DHT.humidity);
 lcd.print("%  ");
  lcd.setCursor(0,1);
     lcd.print("Temperatura = ");
    lcd.print(DHT.temperature);
    lcd.println("C  ");
  delay(8000);}
```

***Rezultate:***

***Aplicație:***




*Să se verifice și compare rezultatele obținute în diferite zone ale încăperii*

## PROBLEME PROPUSE

1. ***Să se utilizeze plăcuța Arduino într-un montaj (ce conține o fotorezistență) care determină luminozitatea dintr-o încăpere.***

2. ***Să se realizeze o aplicație care să creeze un joc de lumini utilizând Arduino și LED-uri de diferite culori.***