Programarea calculatoarelor

Seminar 1

Profesor: Maria Guțu

Noţiuni fundamentale de programare

Programarea este disciplina informatică ce are ca scop realizarea de programe care să constituie soluţiile oferite cu ajutorul calculatorului unor probleme concrete.

Prezentare generală a limbajului C

- C este un limbaj de programare creat la începutul anilor ’70 (1969 - 1973) de către Ken Thompson și Dennis Ritchie;
- Deși este un limbaj de nivel înalt, care respectă principiile programării structurate, arhitectura sa conține elemente foarte asemănătoare instrucțiunilor cod-mașină;
- Această proprietate a făcut ca C să fie folosit pe scară largă pentru crearea de software de bază, incluzând sisteme de operare, browsere, drivere etc.;
- C nu este numai unul dintre cele mai folosite limbaje de programare din toate timpurile, ci și cel mai influent: C#, Java, Objective C, PHP, Python și multe alte limbaje au preluat construcțiile sale de bază.

Etapele realizării unui program C

- Limbajul C este unul compilat, adică programatorul scrie instrucțiuni specifice într-un fișier text, numit fișier sursă, cu extensia .c;
- Apoi un alt program, numit compilator ”traduce” textul respectiv (numit ”cod sursă”, sau pur și simplu ”sursă”);
- Se obține un alt fișier care poate fi înțeles și executat de către sistemul de operare;
- Nu orice cod sursă compilat poate fi executat; există fișiere cu extensia .c care sunt doar folosite de către alte fișiere .c.

Structura unui program C

- Prin ”program C” vom înțelege un fișier sursă care poate fi executat;
- În general programele C conțin o primă zonă, cu așa numitele instrucțiuni de ”precompilare” – în cazul programului alăturat:

  #include <stdio.h>
- Aceste instrucțiuni indică secvențe de cod care vor fi inserate în codul sursă (fără ca programatorul să vadă codul inserat) sau înlocuiri care urmează să fie făcute în etapa de compilare;
- Biblioteca stdio.h conține funcții speciale pentru citirea și afișarea datelor – prin urmare instrucțiunea de includere a acesteia va fi practic prezentă în orice program pe care îl vom scrie în C.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello world!\n");
6      return 0;
7  }
8
```

Structura unui program C

- **#include <stdio.h>**

Ce biblioteci (librării) standart cunoașteți?

- **#include "nume_fisier.c„**

- **Exemple:**

 # include "factorial.c"
 # include "D:\\Code-Blocks\\nrPare.c"

Structura unui program C

- **Sintaxa generală #include < nume_fisier_antet.h>**
- **Exemplu: #include < stdio.h >**

Nume fișier antet	Descrierea
stddef.h	definește mai multe tipuri și macro-uri utile;
stdint.h	definește tipurile de întregi exacte de lățime;
stdio.h	definește funcțiile de intrare și ieșire de bază;
stdlib.h	definește funcțiile de conversie numerică, generatorul de rețea pseudo-aleatoriu, răspunde de alocarea memoriei;
string.h	definește funcțiile de manipulare a șirurilor;
math.h	definește funcțiile matematice comune.

Structura unui program C

- Orice program C este de fapt o colecție de module, numite ”funcții” sau ”subprograme” care interacționează;
- Fiecare funcție este formată din antet și blocul de instrucțiuni, delimitat de acolade;
- Antetul conține tipul rezultatului returnat, numele funcției și lista parametrilor (care poate fi vidă), delimitată de paranteze;
- Funcția **int main()** este punctul de pornire al oricărui program;
- În lipsa acesteia, compilarea fișierului sursă nu are ca efect obținerea unui program executabil;
- Prin urmare, fiecare program pe care îl vom scrie va conține funcția **int main()** și blocul de instrucțiuni corespunzător.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello world!\n");
6      return 0;
7  }
8
```

Structura unui program C

- Programul alăturat conține două instrucțiuni;
- Prima dintre ele:

  printf(”Hello world!\n”);

  are ca efect afișarea pe ecran a mesajului de salut corespunzător;

  A doua instrucțiune:

- **return 0;**
- este obligatorie la sfârșitul funcției **int main()**, prin urmare va apărea în toate programele pe care urmează să le scriem.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello world!\n");
6      return 0;
7  }
8
```

Citirea datelor în C

- Pentru citirea datelor, în C se folosește funcția scanf:

  scanf(”%format”, &variabila);

Exemplu:

```
int x;
scanf(”%d”, &x);
```

- Pot fi citite mai multe variabile cu un singur apel al funcției scanf, chiar dacă variabilele sunt de tipuri diferite:

  scanf(”%format1%format2”, &var1, &var2);

Exemplu:

```
int x;
double y; // variabila de tip real cu formatul lf
scanf(”%d%lf”, &x, &y);
```

- Lista completă a formatelor pentru citire și afișare cu funcții din biblioteca stdio.h urmează să fie studiate ulterior, odată cu tipurile de date din limbajul C.

Afișarea datelor în C

- Pentru afișarea datelor, în C se folosește funcția printf:
  printf(”%format”, expresie);

Exemplu:

```
int x;
...
printf(”%d”, x+2);
```

- Pot fi afișate expresii complexe, care să conțină și mesaje de tip text cu un singur apel al funcției printf:
  printf(”text0 %format1 text1 %format2 text2”, expr1, expr2);

Exemplu:

```
int x;
double y; // variabila de tip real cu formatul lf
...
printf(”x este egal cu %d, iar y+1 =%lf”, x, y+1);
```

Declararea variabilelor în C

- Spre deosebire de scheme logice și pseudocod, în C fiecare variabilă trebuie să fie declarată înainte de a fi folosită;
- Declararea presupune precizarea tipului și numelui variabilei:

  tip numeVariabila;

- Declararea are ca efect rezervarea unei zone de memorie (în memoria internă) care va fi identificată cu ajutorul numelui variabilei;
- Tipul variabilei va determina dimensiunea memoriei alocate și operațiile permise (de exemplu, pentru o variabilă de tip real, nu va fi permisă operația ”%” care determină restul împărțirii);
- Tipul poate fi unul simplu, predefinit (int, char, float, double etc.) sau unul definit anterior în cadrul programului;
- Numele variabilei poate conține litere, cifre și caracterul underscore (_) și **NU** trebuie să înceapă cu o cifră.

Instrucțiunea de atribuire în C

- Atribuirea are în C următoarea formă:

  variabila = expresie;

Exemplu:

```
int x;
x = 18;
```

- De fapt în C atribuirea nu este o instrucțiune propriu-zisă ci este o expresie.

Exemplu:

```
#include <stdio.h>

int main()
{
    int x = 1, y, z = 3,t;
    x += y = z -= t = 5;
    printf("x = %d\n y = %d\n z = %d\n t = %d",x, y, z, t);
    return 0;
}
```

Operatorii aritmetici în C

Operatorii aritmetici folosiţi în limbajul C sunt:

+ adunarea a două numere;

- scăderea a două numere;

***** înmulţirea a două numere;

/ împărţirea a două numere (rezultatul împărţirii pentru numere reale, câtul împărţirii pentru numere întregi);

% modulo (restul împărţirii a două numere întregi);

++ incrementarea (mărirea unei valori cu o unitate);

-- decrementarea (micşorarea unei valori cu o unitate).

Operatorii aritmetici în C

Ce va afișa următorul program?

```c
#include <stdio.h>

int main (void)
{
int a = 54;
int mod;
mod = a % 10;
printf ("Restul impartirii lui %d la 10 este %d\n",a,mod);
return 0;
}
```

Operatorii aritmetici în C

Operatorii de incrementare - decrementare sunt folosiţi pentru mărirea respectiv micşorarea unei valori cu o unitate.

Sunt de forma:

v++ - incrementare;

--v – decrementare.

Se pot folosi şi instrucţiuni de pre/pos incrementare/decrementare:

- post-incrementare: x = a++;

  - este echivalentă cu: x = a; a = a + 1;

- pre-incremenatare: x = ++a;

  - este echivalentă cu: a = a + 1; x = a;

Pentru decrementare se procedează în mod analog.

Operatorii aritmetici în C

Ce va afișa următorul program?

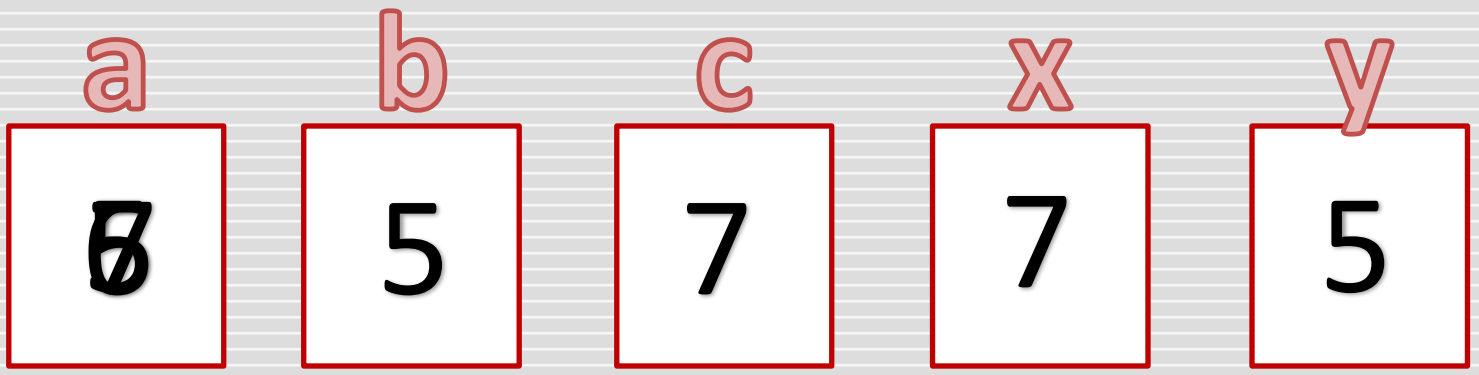
```c
#include <stdio.h>

int main (void)
{
    int a = 5, b, c, x, y;
    b = a++;
    c = ++a;
    x = a--;
    y = --a;
    printf ("a = %d\nb = %d\nc = %d\nx = %d\ny = %d\n",a,b,c,x,y);
    return 0;
}
```

Operatorii aritmetici în C

```c
#include <stdio.h>

int main (void)
{
    int a = 5, b, c, x, y;
    b = a++;
    c = ++a;
    x = a--;
    y = --a;
    printf ("a = %d\nb = %d\nc = %d\nx = %d\ny = %d\n",a,b,c,x,y);
    return 0;
}
```



Operatorii aritmetici în C

Ce va afișa următorul program?

```c
#include <stdio.h>

int main (void)
{
    int a = 5, b, c, x, y;
    b = a++; a = 5;
    c = ++a; a = 5;
    x = a--; a = 5;
    y = --a; a = 5;
    printf ("a = %d\nb = %d\nc = %d\nx = %d\ny = %d\n",a,b,c,x,y);
    return 0;
}
```

Operatorii aritmetici în C

Ce va afișa următorul program?

```c
#include <stdio.h>

int main()
{
    int x = 1, y, z = 3,t;
    x += y = z -= t = 5;
    printf("x = %d\n y = %d\n z = %d\n t = %d",x, y, z, t);
    return 0;
}
```

x = x-5 y

xy+= y

x	y	z	t
-1	-2	-2	5

Operatorii aritmetici în C

Ce va afișa următorul program?

```c
#include <stdio.h>

int main()
{
    int x = 2, y, z = 3, t;
    x += y = z *= t = 5;
    printf("x = %d\n y = %d\n z = %d\n t = %d",x, y, z, t);
    return 0;
}
```

```c
#include <stdio.h>

int main()
{
    int x = 2, y, z = 3, t;
    x += y /= z *= t = 5;
    printf("x = %d\n y = %d\n z = %d\n t = %d",x, y, z, t);
    return 0;
}
```

Operatori relaţionali şi logici

Termenul „relaţional” se referă la relaţiile care se pot stabilii între diverse valori.

Termenul logic se referă la felul în care se pot lega relaţiile existente.

Limbajul C nu implementează explicit tipul de date Boolean. C folosește tipul de date **int**, iar expresii precum i>j returnează valorea 1 pentru adevărat și 0 pentru fals.

Operatori relaţionali şi logici

Operatori relaționali

Operator	Acțiune
>	Mai mare decât
>=	Mai mare sau egal
<	Mai mic
<=	Mai mic sau egal
==	Egal
!=	Diferit

Operatori logici

Operator	Acțiune
&&	ȘI
||	SAU
!	NU

Operatori relaţionali şi logici

Tabelul de adevăr pentru operatorii logici, prezentat folosind cifrele 1 şi 0.

a	b	a && b	a || b	!a
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Operatori relaţionali şi logici

Ce va afișa următorul program?

```
#include <stdio.h>

int main()
{
    int a=3,b=4,c=0,rez;
    rez=!(a < c) && (a<b) || (b<=c);
    printf("%d\n",rez);
    return 0;
}
```

Exemple de programe

1. **Se citesc a,b,c coeficienţii reali a unei ecuaţii de gradul II. Să se afişeze soluţile reale ale ecuaţiei.**

Descrierea algoritmului:

- ecuaţia de gradul II este de forma $ax^2 + bx + c = 0$;

-presupunând că a != 0 (a<>0),calculăm determinantul ecuatiei delta=b*b-4*a*c;

$$\Delta = b^2 - 4ac$$

$$\Delta > 0: \exists x_1, x_2 \in \mathbb{R}, x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a}$$

$$\Delta = 0: \exists x_1, x_2 \in \mathbb{R}, x_1 = x_2 = \frac{-b}{2a}$$

$$\Delta < 0: \nexists x \in \mathbb{R}.$$

Exemple de programe

```c
#include <stdio.h>

int main()
{
    float a, b, c, delta;
    printf("Introdu coeficientii:\n");
    scanf("%f%f%f", &a, &b, &c);
    if (a != 0){
        delta = pow(b,2) - 4*a*c;
        if (delta > 0){
            printf("x1 = %f\nx2 = %f\n",(-b + sqrt(delta))/(2*a), (-b - sqrt(delta))/(2*a));
        }
        if (delta = 0){
            printf("x1 = %f\nx2 = %f\n",-b/(2*a), -b/(2*a));
        }
        if (delta < 0){
            printf("Ecuatia nu are sol. reale");
        }
    } else {
        printf("Ecuatia nu are solutii!");
    }
    return 0;
}
```

Probleme spre rezolvare

1. **Se consideră două numere întregi distincte. Să se scrie un program care determină numărul mai mare.**
2. **Se consideră un cerc de raza *R* și un pătrat cu latura *a*. Să se scrie un program care determină dacă cercul încape în pătrat.**
3. **Se consideră un număr natural citit de la tastatură. Să se scrie un program care afișează la ecran dacă numărul este par sau impar.**
4. **Se consideră trei numere reale. Să se scrie un program care determină dacă aceste numere reprezintă lungimile laturilor unui triunghi și, în caz afirmativ, să se:**
   - **Calculeze perimetrul triunghiului;**
   - **Determine natura triunghiului în dependență de lungimile laturilor(isoscel, echilateral, scalen).**