UNIVERSITATEA TEHNICĂ A MOLDOVEI

Programarea calculatoarelor

Fundamentele limbajului C

Profesor: Maria Guțu

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Structura unui program C

Header/antet	#include <stdio.h>
main()	int main() {
Partea declarativă	int a = 15;
Corpul programului	printf(“%d”, a);
Return	return 0; }

Fișiere antet

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.

Sintaxa de includere a fișierelor antet

```
#include <nume_fisier_antet.h>
```

Declararea metodei main()

Sintaxa funcției main() este:

```
int main()
{ …
}
```

Declararea variabilelor

```
int main()
{  int x;
float y = 12.3, z;
char ch;   …
}
```

Corpul programului

```
int main()
{  int x = 15;
printf("%d", x);
…
}
```

Declarația de returnare

```
int main()
{  int x = 15;
printf("%d", x);
    return 0;
}
```

Specificatori de format

Specificatori format	Descriere
%d	întreg zecimal cu semn
%i	întreg zecimal, octal (0) sau hexazecimal (0x, 0X)
%o	întreg în octal, fără 0 la inceput
%u	întreg zecimal fără semn
%x, %X	întreg hexazecimal, fără 0x/0X; cu a-f pt. %x, A-F pt. %X
%c	caracter
%s	şir de caractere, până la '\0' sau nr. de caractere dat ca precizie

Specificatori de format

%f, %F	real fără exp.; precizie implicită 6 poz.; la precizie 0: fără punct real (posibil cu exponent)
%e, %E	numere reale cu mantisă şi exponent (al lui 10); precizie implicită 6 poz.; la precizie 0: fără punct
%g, %G	numere reale în format %f sau %e, funcție de valoare real, ca %e, %E dacă exp. < -4 sau precizia; altfel ca %f. Nu tipăreşte zerouri sau punct zecimal în mod inutil
%p	pointer, în formatul tipărit de printf
%ld, %li	numere întregi lungi
%lf, %le, %lg	numere reale în precizie dublă (*double*)
%Lf, %Le, %Lg	numere reale de tip *long double*
%%	caracterul procent

Tipuri de date

char – un singur caracter;

int – numere întregi;

float – numere zecimale;

double – numere reale cu dublă precizie;

Tipuri de date

_Bool – stochează 0 sau 1, <stdbool.h>;

int – numere întregi;

float – numere zecimale;

double – numere reale cu dublă precizie;

Tipuri de date

Tip de Date	Memoria	Valorile	Format
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld

Tipuri de date

Tip de Date	Memoria	Valorile	Format
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4		%f
double	8		%lf
long double	12		%Lf

Afișarea datelor în C

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

Exemplu:

```
int x = 5;
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 = 3;
double y = 1; // variabila de tip real cu formatul lf
…
printf(”x este egal cu %d, iar y+1 =%lf”, x, y+1);
```

Exemple de program C
Specificatorul de format: %c

```
#include <stdio.h>
int main()
{
        char ch = 'A';
        printf("%c\n", ch);
        return 0;
}
```

Exemple de program C
Specificatorul de format de tip întreg: %d, %i

```
#include <stdio.h>
int main()
{
        int x = 25;
        printf(„%d\n", x);
        return 0;
}
```

Exemple de program C
Specificatorul de format de tip float: %f, %e, %E

```
#include <stdio.h>
int main()
{
        float nota = 9.89;
        printf("%f\n", nota);
        return 0;
}
```

Exemple de program C
Specificatorul de format de tip float: %f

```
#include <stdio.h>
int main()
{
        float z = 12.345673;
        printf("%10f\n", z);
        return 0;
}
```

```
#include <stdio.h>
int main()
{
        float z = 12.345623;
        printf("%-10f\n", z);
        return 0;
}
```

Exemple de program C
Specificatorul de format de tip float: %f

```
#include <stdio.h>
int main()
{
        float z =12.345823;
        printf("%+10f\n", z);
        return 0;
}
```

```
#include <stdio.h>
int main()
{
        float z = 12.347823;
        printf("%10.3f\n", z);
        return 0;
}
```

Exemple de program C
Specificatorul de format a unui șir: %s

```
#include <stdio.h>
int main()
{
        char a[] = "Sunt student";
        printf("%s\n", a);
        return 0;
}
```

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);
```

Exemple de program C
Citirea de la tastatură a unui char

```
#include <stdio.h>
int main()
{       printf(“Intrudu un caracter:  ”);
        char ch;
        scanf(“%c“, &ch);    //input
        printf(" Caracterul introdus este: %c\n", ch);
        return 0;
}
```

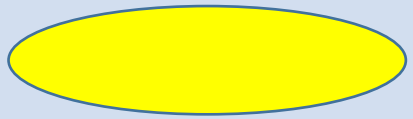
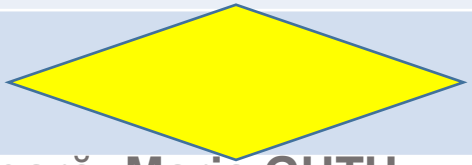
Exemple de program C
Citirea de la tastatură a unui nr. întreg

```
#include <stdio.h>
int main()
{       printf(“Intrudu un numar:  ”);
        int x;
        scanf(“%d“, &x);    //input
        printf(" Numarul introdus este: %d\n", x);
        return 0;
}
```

Exemple de program C
Citirea de la tastatură a unui string

```
#include <stdio.h>
int main()
{       printf(“Introdu sirul:  \n”);
        char  str[10];
        scanf(“%s“, str);    //input
        printf(" Sirul introdus este: %s\n", str);
        return 0;
}
```

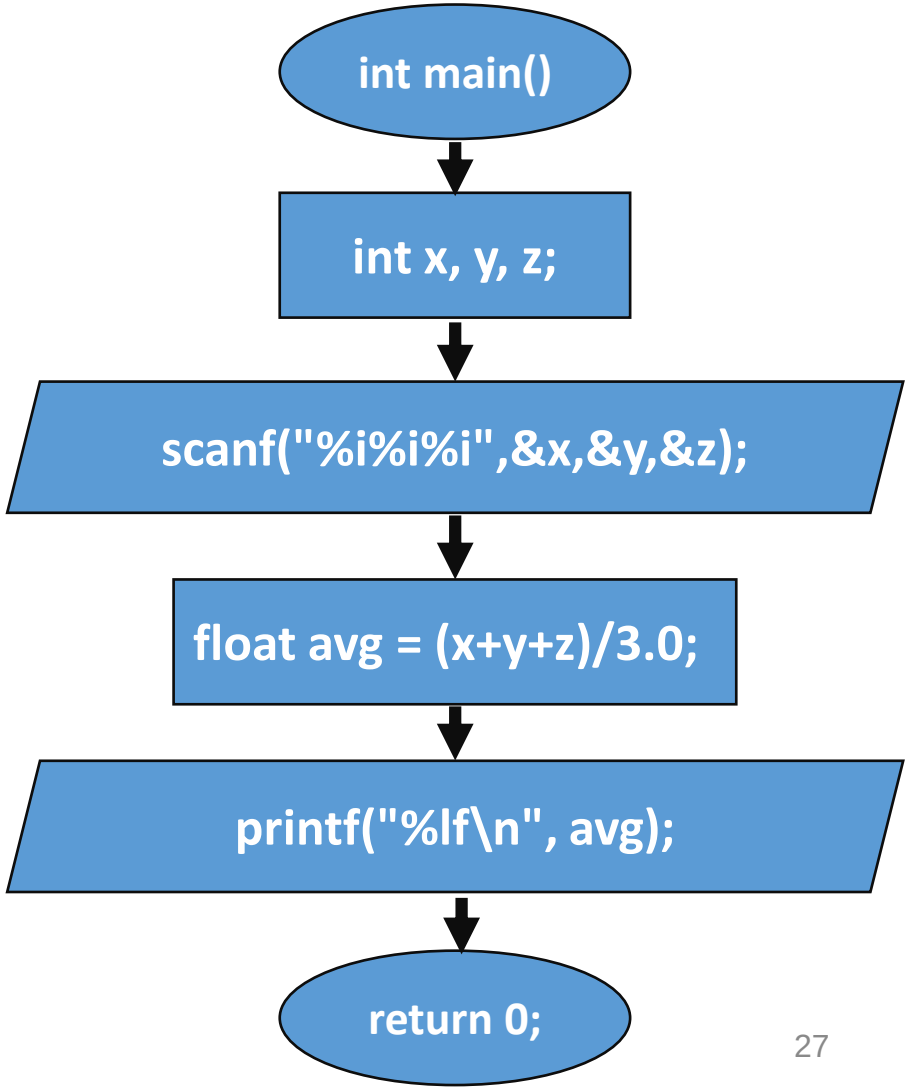
Reprezentarea grafică / organiograma soluției algoritmice / Flowchart

Simbol	Denumirea	Funcționalitatea
	Start/End	Reprezintă începutul și sfârșitul unei execuții.
	Săgeți	Linii conectoare care indică relația dintre figuri.
	Input/Output	Introducerea/Afișarea datelor
	Procesarea	Procesarea datelor
	Decizia	Condiție decizională

Exemplu

Se consideră trei numere întregi. Să se scrie un program care calculează media lor aritmetică.

```
#include <stdio.h>
int main() {
        int x, y, z;
        scanf("%i%i%i",&x,&y,&z);
        float avg = (x+y+z)/3.0;
        //float avg = (float)(x+y+z)/3;
        printf("%lf\n", avg);
        return 0; }
```



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;
}
```

Operatori de atribuire în C

Operator	Acțiune
=	Atribuire simplă: x = a+b;
+=	Adaugă și operator de atribuire: x +=5; => x = x + 5;
-=	Extrage și operator de atribuire: x -=a; => x = x - a;
*=	Înmulțit și operator de atribuire: x *=y; => x = x * y;
/=	Împărțit și operator de atribuire: x /=2; => x = x / 2;
%=	Restul împărțirii și operator de atribuire: x %=3; => x = x % 3;

Instrucțiunea de atribuire în C

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

Atribuirea are în C următoarea formă: variabila = expresie;

Exemplu: int x; x = 18;

```
#include <stdio.h>
int main()
{
    int x = 1, y, z = 3, t;
    x += y = z -= t = 5;
    printf("x=%d; y=%d; \nz=%d; t=%d", x, y, z, t);
  return 0;
}
```

Operatorii aritmetici în C

Ce va afișa următorul program?

```
#include <stdio.h>
void main()
{
    int x = 2, y=5, z = 3, t;
    x += y *= z %= t = 12;
    printf("x=%d; y=%d; \nz=%d; t=%d", x, y, z, t);
}
```

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?

```
#include <stdio.h>
int main()
{
    int a =3, b, c, d, e;
    b = a++;
    c = ++a;
    d = a--;
    e = --a;
    printf("a=%i\nb=%i\nc=%i\nd=%i\ne=%i\n", a, b, c, d, e);
 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;
}
```

Alți operatori

Operator	Acțiune
sizeof()	Returnează lungimea variabilei: sizeof(x);
&	Returnează adresa variabilei: &x;
*	Returnează pointerul unei variabile: *x;
?:	Operator condițional: Condiție? True : False

Operator condițional

```
#include <stdio.h>
int main()
{
    int x, y;
    printf("introdu numere\n");
    scanf("%d %d", &x, &y);
    int rez = x > y ? x : y;
    printf("max=%d", rez);
    return 0;
}
```

```
#include <stdio.h>
int main()
{
    int x, y;
    printf("introdu numere\n");
    scanf("%d %d", &x, &y);
    x > y ? printf("max=%d", x):
            printf("max=%d", y);
    return 0;
}
```

Funcții matematice
#include <math.h>

int abs (int x);	valoarea absoluta a argumentului.
double fabs (double x);	valoarea absoluta a argumentului real.
double floor(double x);	rotunjire prin lipsa.
double ceil(double x);	rotunjire prin adaos.
double sin (double x);	sin(x), unde x este dat in radiani. Numărul real returnat se afla in intervalul [-1, 1].
double cos (double x);	cos(x), unde x este dat in radiani. Numărul real returnat se afla in intervalul [-1, 1].
double tan(double x);	tg(x), unde x este dat in radiani.
double exp (double x);	e la puterea x.

Funcții matematice
#include <math.h>

double log(double x);	logaritmul natural al argumentului (ln(x).
double log10(double x);	logaritmul zecimal al argumentului (lg(x).
double pow (double baza, double exponent);	Baza la puterea exponent.
double pow10(int x);	10 la puterea x.
double fmod(double x, double y);	restului de la împărțirea lui x la y.
double sqrt(double x);	rădăcina pătrata a argumentului x.
double cbrt(double x);	rădăcina de ordinal 3 a argumentului x.
M_E și M_PI	Valoarea constantelor Euler și PI.

Random & Limits

Un program C pentru a printa valoarile INT_MAX și INT_MIN, trebuie să includă fișierul antet **<limits.h>**

```
#include <limits.h>
#include <stdio.h>
int main()
{
    printf("%d\n", INT_MAX);
    printf("%d\n", INT_MIN);
    printf("%ld\n", LONG_MAX);
    printf("%ld\n", LONG_MIN);
}
```

Random & Limits

Funcția **rand()** este utilizată pentru generarea numerelor aleatoare din diapazonul [0, RAND_MAX).

Pentru a folosi această funcție trebuie de importat fișierul-antet **#include <stdlib.h>.**

Reține: Dacă numerele aleatoare sunt generate cu **rand()** fără a apela mai întâi **srand()**, ***programul dvs. va crea aceeași succesiune de numere de fiecare dată când rulează***.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
      printf(" %d ", rand());
      printf(" %d ", rand());
      return 0;
}
```

Random & Limits

Funcția ***srand()*** stabilește punctul de plecare pentru producerea unei serii de numere întregi pseudo-aleatorii, prin urmare, funcția ***srand()*** setează generatorul la un punct de plecare diferit.

Notă: Generatorul de numere pseudo-aleatorii ar trebui să fie inițializat o singură dată, înainte de orice apeluri către ***rand()*** și începutul programului. Practica standard este de a folosi rezultatul unui apel către ***srand(time (0))*** ca inițializare. Cu toate acestea, ***time()*** returnează o valoare ***time_t*** care variază de fiecare dată și, prin urmare, numărul pseudo-aleator variază pentru fiecare apel de program.

Random & Limits

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

int main(void){
      srand(time(0));
      // srand(time(NULL));
      printf(" %d ", rand());
      printf(" %d ", rand());
      return 0;
}
```

NOTĂ: Acest program va crea secvențe diferite de numere aleatorii la fiecare rulare a programului.

Generarea numerelor aleatorii între un interval predefinit

Input:
Lower = 50,
Upper = 100
Deoarece C nu are o funcție încorporată pentru generarea unui număr în interval, dar are o funcție rand() care generează un număr aleator de la 0 la RAND_MAX. Cu ajutorul funcției rand() se poate genera un număr în interval ca:
num = (rand() % (upper – lower + 1)) + lower.

NOTĂ: Folosim operatorul % în programul nostru pentru orice set de valori a și b pentru a genera numere între zero și valoarea indicate după operatorul %.

Generarea numerelor aleatorii între un interval predefinit

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){
int lower = 2, upper = 35;
      srand(time(0));
int num = (rand() % (upper - lower + 1)) + lower;
printf("%d ", num);
return 0;
}
```

Generarea numerelor aleatorii între un interval predefinit

```
#include <stdio.h>
#include <stdlib.h>
int main() {
      int n;
      printf("Numar in intervalul [1,100]\n");
      srand(time(NULL));
      n = rand() % 100 + 1;
      printf("%d\n", n);
  return 0;
}
```

Aplicații Practice!!!

(maria.gutu@iis.utm.md)