Programarea calculatoarelor
Limbajul C

CURS 2

Ş.l. Carmen Odubăşteanu

Cuprins

Limbajul C - Scurt istoric

- 1972, *Dennis Ritchie (AT&T Bell Laboratories)*
  - pentru programe de sistem
- 1973, sistemul de operare UNIX este în totalitate scris în C
- cartea de referință care defineşte un standard minim: Brian W. Kernighan, Dennis Ritchie - "The C Programming Language" - Prentice Hall 1978
- a fost dezvoltat un standard internațional (1983-1989)- ANSI C (ANSI - American National Standards Institute)
- sunt dezvoltate medii de programare C performante sub UNIX şi DOS, care contribuie la utilizarea masivă a limbajului.

Limbajul C - Scurt istoric

- gruparea structurilor de date cu operațiile care prelucrează respectivele date - o*biect* sau *clasa*.
- 1980, Bjarne Stroustrup: "*C with Classes*“
- 1983, C-with-classes a pătruns şi în lumea academică şi a instituțiilor de cercetare.
- Denumirea finală a acestui limbaj a fost *C++*.
- Succes: a extins cel mai popular limbaj al momentului, C.
- Programele scrise în C funcționează şi în C++, şi ele pot fi transformate în C++ cu eforturi minime.
- Cea mai recentă etapă în evoluția acestui limbaj - limbajul *JAVA (SUN)*

Caracteristicile limbajului

- limbaj de nivel înalt, portabil, structurat, flexibil
- permite atat programarea la nivel înalt cât şi la nivel scăzut.
- produce programe eficiente (lungimea codului relativ scăzută, viteza de execuție mare)
- set bogat de operatori
- multiple facilități de reprezentare şi prelucrare a datelor
- utilizare extensivă a apelurilor de funcții şi a pointerilor
- verificare mai scăzută a tipurilor -*loose typing* - spre deosebire de PASCAL
- limbaj procedural
  - definirea structurilor de date
  - definirea funcțiilor pentru lucrul cu aceste structuri

Utilizare

Este utilizat în multiple aplicații, în care nu predomină caracterul numeric:

- programe de sistem
- proiectare asistată de calculator
- grafică
- prelucrare de imagini
- aplicații de inteligență artificială
- etc

Mediul de dezvoltare DevC++

- Scrierea unui program simplu
  - Editare
    - File - New
  - Compilare : se rezolvă directivele preprocesor prin expandarea în codul sursă şi apoi se transpune programul sursă în program obiect (.obj)
    - Compile
  - Link-editare : legarea programului obiect obținut la pasul anterior cu bibliotecile de sistem şi transformarea lui într-un program executabil
    - Compile
    - Build
  - Lansare în execuție
    - Run

Primul program C

```
#include<stdio.h>
void main(void)
{
  printf(“Hello World!”);
}
```

Sau:

```
#include<stdio.h>
void main()
{
  printf(“Hello World!”);
}
```

Primul program C – fără warning la compilare în DevC++

```
#include<stdio.h>
#include<stdlib.h>
int main(void)
{
  printf(“Hello World!”);
  system(“pause”);
  return 1;
}
```

Structura unui program C

- Un program C este compus dintr-o ierarhie de funcții, orice program trebuind să conțină cel puțin funcția main, prima care se execută la lansarea programului C.
- Un program C are, în principiu, următoarea *structura*:
  - directive preprocesor
  - definiții de tipuri
  - prototipuri de funcții - tipul unei funcții (tipul valorii returnate) şi tipurile parametrilor transmişi funcției
  - definiții de variabile globale
  - definiții de funcții

Observații

- execuția programului începe cu prima linie din main()
- cuvintele cheie sunt scrise cu litere mici
- instrucțiunile se termină cu ';'
- şirurile de caractere sunt incluse între ghilimele duble
- limbajul C este case sensitive
- \n poziționează cursorul la începutul liniei următoare
- printf() poate fi utilizată pentru afişare pe ecran
- {…} delimitează începutul şi sfârşitul unui bloc-program
- #include: directivă de preprocesare (includerea unor funcții de bibliotecă)

Elemente de bază ale limbajului C

- Alfabetul şi atomii lexicali
- Identificatorii
- Cuvintele cheie
- Tipurile de date
- Constantele şi variabilele
- Comentariile
- Operatorii şi expresiile

Alfabetul limbajului

- Caracterele se codifică conform *codului ASCII* (American Standard Code for Information Interchange)
  - Codificare pe 8 biți (un octet);
  - sunt 256 (0 - 255) de caractere în codul ASCII.
  - alfabetul cuprinde simboluri grafice şi simboluri fără corespondent grafic
  - spațiul are codul mai mic decat simbolurile grafice (32)
  - cifrele (în ordine crescătoare), literele mari şi literele mici (în ordine alfabetică) ocupă cate trei zone compacte.

Program C

- *Atomi lexicali:*
  - identificatori
  - constante (explicite) - numerice, caracter, şir
  - operatori
  - semne de punctuatie.
- Separati de *separatori*:
  - spațiul
  - caracterul de tabulare orizontală
  - terminatorul de linie
  - comentariul
    - orice text aflat între combinațiile de caractere /* şi */
    - textul început cu // până la sfârşitul liniei

Identificatori

- Identificatorii (nume) - primul caracter o literă sau _ :
  - suma
  - produs
  - x1
  - X1
  - PI
  - functia_gauss
- Cuvintele cheie:

int	extern	else	char	register	for	
float		typedef	do	double	static	while
struct		goto	switch	union	return	case
long	sizeof	default	short	break	if	
unsigned	continue		auto			

- Standardul ANşi C a mai adăugat :

enum	const	signed	void	volatile

Tipuri de date

***Fundamentale*:**

- caracter (char – 1 octet)
- întreg (int – 2 octeti)
- virgulă mobilă (float – 4 octeti)
- virgulă mobilă dublă precizie (double – 8 octeti)
- nedefinit (void)

- ***Derivate:***
  - tipuri structurate (tablouri, structuri)
  - tipul pointer

Tipuri aritmetice

- dimensiuni şi specificatori (signed/unsigned, short/long)

Tip	Lungime	Domeniu
char	**8**	**-127÷127**
int	**16**	**-32.767÷32.767**
float	**32**	**10^{-37}÷10^{37} (şase zecimale exacte)**
double	**64**	**10^{307}÷10^{307} (zece zecimale exacte)**

Tipuri aritmetice

Tip	Lungime	Domeniu
unsigned char	8	0÷255
signed char	8	-127÷127
unsigned int	16	0÷65.535
signed int	16	-32.767÷32.767
short int	16	-32.767÷32.767
unsigned short int	16	0÷65.535
signed short int	16	-32.767÷32.767
long int	32	-2.147.483.647÷2.147.483.647
signed long int	32	-2.147.483.647÷2.147.483.647
unsigned long int	32	0÷4.294.967.295
long double	80	zece zecimale exacte

Valori

- Valori de tip întreg (implicit int)
  123
  int numar_octal=045;
  int numar_hexa=0x25;
  1234L
  12345ul

- Valori de tip real (implicit double)
  -1,5e-5
  2.e-4
  12.4
  12.3f

- Valori de tip caracter -se reprezintă pe un octet şi au ca valoare codul ASCII al caracterului respectiv
  'A' -> 65

Şiruri şi caractere

- Caractere speciale–se foloseşte \
  Corect: ‘\’’ - pentru apostrof
  ‘\\’ - pentru backslash
  Greşit: ‘’’ sau ‘\’

- Secvențe escape:
  \n \t \a \b

- Valori şir de caractere - succesiune de octeti ce contin codurile ASCII ale caracterelor din şir şi la sfârşit octetul nul (termină orice şir de caractere)
  char s[]=“abc”;
  “A” -> 65 00
  “abc01”

Tipul void

- nu are constante (valori)
- utilizat atunci când funcțiile nu întorc valori

```
void f(int a)
{
   if (a) a =a/2;
}
```

- sau când funcțiile nu au parametri

```
void f(void)
/*echivalent cu void f()*/
int f(void)
/*echivalent cu int f()*/
```

Constante, variabile, comentarii

- Constante

```
const double PI=3.1415;
const int LIM=10000;
```

- Variabile – declarare, inițializare:

  tip lista_variabile;

```
char a,b,c='x';
int d;
double d;
float f=1.2;
```

- *Comentarii*

```
/* Acesta este
un comentariu în C */
// acesta este un alt comentariu C (C++)
```

Operatori, operanzi, expresii

- *Operatorii:* simboluri utilizate pentru precizarea operațiilor care trebuie executate asupra operanzilor.
- *Operanzii:* constante, variabile, nume de funcții, expresii.
- *Expresiile:* entități construite cu ajutorul operanzilor şi operatorilor, respectând *sintaxa (regulile de scriere)* şi *semantica (sensul, înțelesul)* limbajului.
- Cea mai simplă expresie este cea formată dintr-un singur operand.

Clasificarea operatorilor

- după numărul operanzilor prelucrați:
  - unari
  - binari
  - ternari - cel condițional;
- după ordinea de succedare a operatorilor şi operanzilor:
  - prefixati
  - infixati
  - postfixati
- după tipul operanzilor şi al prelucrării:
  - aritmetici
  - relaționali
  - logici
  - la nivel de bit.
- Operatorii se împart în *clase de precedență*, fiecare clasă având o *regulă de asociativitate*, care indică ordinea aplicării operatorilor consecutivi de aceeaşi precedență (prioritate).

Operatori

- Operatori aritmetici

++ --	incrementare, decrementare
-	minus unar
* / %	înmulțire, împărțire, rest împărțire întregi
+ -	adunare, scădere

- Operatori relaționali şi logici

!	*not*
> >= < <=	
== !=	egal, diferit de
&&	şi logic
||	sau logic

0 –fals în C!

Orice valoare diferită de 0 este considerată ca având valoarea adevărat!

Operatori

- Operatorul de atribuire

  nume_variabila=expresie;

  a = b = … = x =expresie;

  v=v op expresie echivalent cu : v op = expresie.

  Exemplu:

  ```
  int a;
  a+=4;
  ```

- Operatorul de conversie explicită (cast)

(tip) operand

```
int a=10, b=4;
double c;
c=a/b;
c=(float)a/b
```

Operatori

- Operatorul dimensiune: sizeof(tip/variabila)
- Operatorul condițional:
  exp1 ? exp2 : exp3;

```
int a=5, b=3,c;
c=a>b ? b : a;
```

- Operatorul virgulă: expr1, expr2, … , exprn

Expresii

- La evaluarea expresiilor se ține cont de precedență şi asociativitatea operatorilor!

Exemple:

```
int a,v=0;
a=++v;        //după executarea Instrucțiunii v=1 şi a=1
a=v++;        //a=1 şi v=2
a=v--;        //a=2 şi v=1
a=--v;        //a=0 şi v=0;
float b=5;
b>5 && b<10
int a, b,c,d;
a=b=c=d=1;
```

Precedența şi asociativitatea operatorilor C

OPERATORI	ASOCIERE
( ) [ ] . -> ++ -- (postfix)	stânga
++ -- (prefix) ! ~ & (adresa) * (deferentiere) + - (unari) sizeof()	dreapta
*** / %**	stânga
+ -	stânga
<< >>	stânga
< <= > >=	stânga
== !=	stânga
&	stânga
^	stânga
|	stânga
&&	stânga
||	stânga
?:	dreapta
= += -= *= /= %= >>= <<= &= ^= |=	dreapta
, (operatorul virgula)	stânga

Conversii implicite

- La evaluarea expresiilor în expresia variabila=expresie
  - Se evaluează prima data expresia, fără a ține cont de tipul variabilei; dacă tipul rezultatului obținut este diferit de cel al variabilei, se realizează conversia implicită la tipul variabilei
  - Dacă o expresie are doar operanzi întregi, ei se convertesc la int
  - Dacă o expresie are doar operanzi reali sau întregi, ei se convertesc la double.

Directive de preprocesare

- Substituția lexicală – constante simbolice

#define identificator [text]
#define void
#define then
#define begîn {
#define end }
#define N 100

- Includerea fişierelor la compilare

#include<nume_fisier>
#include ”nume_fisier”

Funcții de intrare/ieşire

- Funcția printf
- Funcția scanf

Exemple:

```
int i;
float r;
double d;
scanf(“%d%f%lf”,&i, &f, &d);

char c;
printf(“rezultatul este: %c\n”,ch);
```

Specificatori format *scanf*

- %d: întreg zecimal cu semn
- %i: întreg zecimal, octal (0) sau hexazecimal (0x, 0X)
- %o: întreg în octal, precedat sau nu de 0
- %u: întreg zecimal fără semn
- %x, %X: întreg hexazecimal, precedat sau nu de 0x, 0X
- %c: orice caracter; nu sare peste spații (doar " %c")
- %s: şir de caractere, până la primul spațiu alb. Se adaugă '\0'.
- %a, %A, %e, %E, %f, %F, %g, %G: real (posibil cu exponent)
- %p: pointer, în formatul tipărit de printf
- %[…]: şir de caractere din mulțimea indicată între paranteze
- %[^…]: şir de caractere exceptând mulțimea indicată între paranteze
- %%: caracterul procent

Specificatori format *printf*

- %d, %i: întreg zecimal cu semn
- %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
- %f, %F: real fără exp.; precizie implicită 6 poz.; la precizie 0: fără punct
- %e, %E: real, cu exp.; precizie implicită 6

Alte funcții de intrare/ieşire

int getchar(void);

- returnează codul unui caracter citit de la tastatura sau valoarea EOF (constantă simbolică definită în stdio.h, având valoarea -1) dacă s-a tastat Ctrl/Z.

int putchar(int c);

- tipăreşte pe ecran caracterul transmis ca parametru;
- returnează codul caracterului sau EOF în cazul unei erori.

int puts(const char * şir);

- tipareste şirul primit ca parametru, apoi un NewLine, cursorul trecand la începutul rândului următor.
- returnează codul ultimului caracter din şir.

Alte funcții de intrare/ieşire

int getche(void);

- Citeşte un caracter (aşteaptă apăsarea unei taste, chiar dacă în buffer-ul de intrare mai sunt caractere neprelucrate)
- afişează caracterul pe ecran
- returnează codul; returneaza EOF la tastarea lui Ctrl/Z, respectiv CR ('\r', cu codul 13) la tastarea lui Enter.

int getch(void);

- Analog cu funcția de mai sus, dar caracterul nu se transmite în ecou (nu se afişează pe ecran).

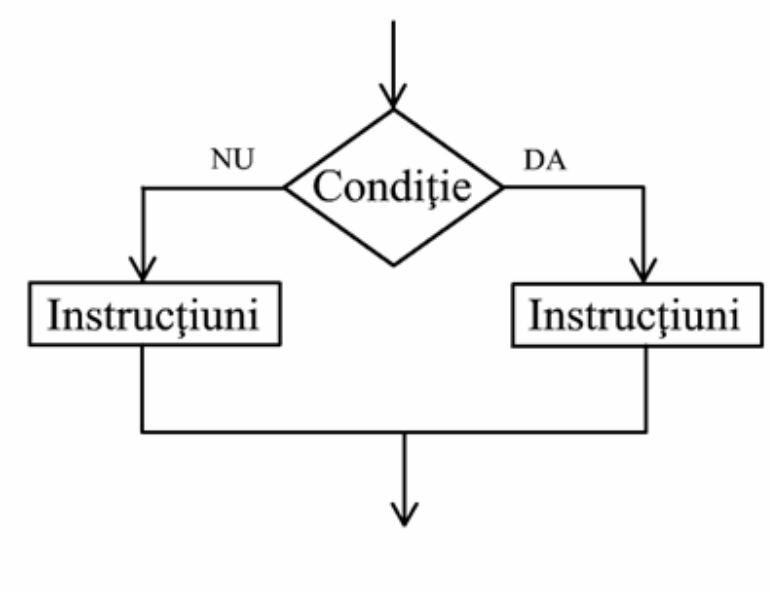
Instrucțiuni C

- Instrucțiunea vidă ;
- Instrucțiunea de atribuire a = 3
- Instrucțiunea compusă (bloc)

  { instr1; instr2; etc }

- Instrucțiunea if

  if (expresie) Instrucțiune1;
  else Instrucțiune2



Instrucțiunea *if*

- Exemple:

```
if(x)
  if(y) printf(“1”);
  else printf(“2”);

if(x){
    if (y) printf(“1”);
}
else printf(“2”);

If (i=0) printf(“Variabila i are valoarea 0”);
else printf(“Variabila i are o valoare diferita de 0”);
```

Probleme propuse

1. Interschimbul valorilor a două variabile a şi b.
2. Rezolvarea ecuației de grad 2: $ax^2+bx+c=0$.
3. Se citesc de la tastatură 3 numere întregi reprezentând lungimile laturilor unui triunghi. Să se calculeze şi să se afişeze aria triunghiului.
4. Să se afişeze în ordine crescătoare valorile a 3 variabile a, b şi c.