Programarea calculatoarelor
Limbajul C

CURS 1

Carmen Odubăşteanu

Cuprins

- Introducere
- Prezentare limbaj C
- Elemente de bază ale limbajului C
  - Tipuri de date şi constante
  - Variabile şi operatori
  - Expresii
  - Directive de preprocesare
  - Funcții I/O
  - Instrucțiuni
- Vectori

Cuprins

- Funcții
- Pointeri
- Vectori şi pointeri
- Funcții şi pointeri
- Şiruri de caractere
- Structuri
- Alocare dinamică
- Programare generică
- Fişiere text şi fişiere binare

- Programe complexe. Compilări separate. Fişiere proiect.
- Convenții de programare

Algoritm

- Succesiune de etape ce se poate aplica mecanic pentru rezolvarea unei clase de probleme
- Redactare
  - Scheme logice
  - Pseudocod
  - Mental – cine îşi permite?
- Cerințe
  - Claritate – fără ambiguități
  - Generalitate – pentru o întreagă clasă de probleme
  - Finitudine – furnizare rezultat în timp finit

Obs: O problemă poate avea mai mulți algoritmi de rezolvare – cel mai bun?

Program

- Descriere precisă şi concisă a unui algoritm într-un anumit limbaj de programare

Limbaje de programare

- Limbaje de nivel coborât, dependente de calculator:
  - Limbaj maşină
  - Limbaj de asamblare
    - mnemonice pentru operațiuni
    - simboluri pentru adrese
  - greu, dar interesant!

Limbaje de programare

- Limbaje de nivel înalt, independente de structura calculatorului:
  - Fortran (FORmula TRANslation) – 1955, IBM, probleme tehnico-ştiințifice
  - Cobol – 1959, probleme economice
  - Programare structurată – ‘70
  - Programare orientată pe obiecte – ‘80

Limbajul C - Scurt istoric

- 1972, *Dennis Ritchie (Laboratoarele AT&T Bell)*
  - 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)*

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

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 Code::Blocks

```
#include<stdio.h>
int main()
{
  printf(“Hello World!”);
  return 0;
}
```

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
- 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

1. 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.

2. 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

3. 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 ANSI C a mai adăugat :

  enum const signed void volatile

4. 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()*/
```

5. 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++)
```

6. 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

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