1 Elemente introductive în limbajul C

BREVIAR TEORETIC

Proiectat și implementat de Dennis Ritchie în 1972 la AT&T Bell Laboratories, pentru programe de sistem (dezvoltate doar în limbaje de asamblare). C-ul este un succesor al limbajului B, creat de Ben Thompson în 1973. Sistemul de operare UNIX este în totalitate scris în limbajul C. Cartea de referință care definește un standard minim: Brian W. Kernighan, Dennis Ritchie - "The C Programming Language" - Prentice Hall 1978. Dezvoltarea unui standard internațional (1983-1989) -- ANSI C (ANSI - American National Standards Institute). Dezvoltate medii de programare C performante sub UNIX si DOS.

Caracteristicile limbajului C, care i-au determinat popularitatea, sunt prezentate pe scurt mai jos și vor fi analizate pe parcursul cursului:

- limbaj de nivel mediu, portabil, structurat, flexibil;
- produce programe eficiente (lungimea codului scăzută, viteză de execuție mare);
- de dimensiune relativ scăzută;
- set bogat de operatori;
- multiple facilități de reprezentare și prelucrare a datelor;
- utilizare extensivă a apelurilor de funcții și a pointerilor;
- verificare mai scăzută a tipurilor *loose typing* - spre deosebire de PASCAL;
- permite programarea la nivel scăzut - *low level* , apropiat de hardware.

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.

Cel mai simplu program scris în limbajul C are următoarea structură[1]:

```
1  /*
2   * hello_world
3   *
4   *  Created on: 2015-02-11
5   *      Author: Ionela Rusu
6   */
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main()
11 {
12     printf("Hello world!\n");
13     return 0;
14 }
15
```

- Liniile 1-6 reprezintă comentarii (linii care vor fi ignorate la execuție);
- Liniile 7 și 8 sunt directive preprocesor;
- Linia 10 definește funcția main() prin care este indicat începutul programului;
- Linia 12 permite afișarea textului "Hello world!" pe ecran;
- Linia 13 marchează sfârșitul programului/funcției main.

[1] Program scris în mediul de dezvoltare Code::Blocks 13.12

Observație: includerea fișierelor prin directiva `#include` se poate face prin două moduri și anume:

a) `#include <fișier.h>`

b) `#include "fișier.h"`

Diferența dintre cele două moduri o reprezintă locația unde aceste fișiere sunt căutate. Prin includerea fișierului prin modul a), acesta este căutat în directoarele standard specificate prin variabilele de mediu sau opțiunile mediului de dezvoltare folosit. Prin modul b), fișierul este căutat în directorul curent, apoi, în cazul în care nu este găsit se va căuta în directoarele standard.

1.1 Variabile, constante și expresii

Algoritmii/programele scrise în limbajul C utilizează variabile, constante și expresii.

Variabilele sunt locații din memorie care sunt identificate prin nume și memorează valori ce se pot schimba pe parcursul execuției programului.

```
<tip> <lista_declaratori>;
<listă_declaratori> <declarator> [, <declarator> ]
Ex.: int a,b,c;
```

Constantele sunt valori (numerice, caracter sau șir de caractere) într-un program care NU se mai modifică pe durata execuției programului. Declararea unei astfel de constante se face utilizând cuvântul cheie "***const***".

```
const <tip> <declarator>;
<declarator> - nume_variabilă = constantă
Ex.: const int SIZE = 10;
```

Constantele simbolice se declară utilizând directiva "`#define`". Utilizarea acestor constante permite o mai bună înțelegere a codului sursă oferind lizibilitate codului.

```
#DEFINE <nume_const> <valoare_const>
Ex.: #DEFINE PI 3.141592653589
```

Expresiile reprezintă combinații de date (operanzi) și operatori. În funcție de tipul variabilei rezultat în limbajul C întâlnim expresii aritmetice, logice, relaționale, enumerare sau șir.

```
Ex.:    int a,b;
        a = (b + 1) * PI;
```

1.2 Tipurile fundamentale (predefinite) în limbajul C

Tabel 1 Tipuri fundamentale

Tip de dată	Exemple
Tipuri întregi	`int a, oVariabilă = 0;` `char litera_A = 'A';`
Tipuri reale	`float element;` `double sir[20], mat[20][20];`
Tipul enumerare	`enum zile {Luni, Marti, Miercuri, Joi, Vineri, Sambata, Duminica};`
Tipul vid (void)	`void functie();`

1.3 Tabela operatorilor în limbajul C

Tabel 2 Tabela operatorilor

Operatori	Asociativitate	Descriere
`( ) [ ] . -> -- ++`	→	Operatori de indexare, selecție, postdecrementare, respectiv postincrementare
`- + -- ++ ! ~ * & sizeof() (tip)`	←	Operatori unari, predecrementare/preincrementare, negare logică, complement față de 1, dereferențiere, adresare, mărime în octeți, conversie de tip
`* / %`	→	Operatori aritmetici: înmulțire, împărțire, restul împărțirii
`+ -`	→	Operatori aritmetici: adunare, scădere
`<< >>`	→	Operatori logici: deplasare pe biți stânga/dreapta
`< <= > >=`	→	Operatori relaționali: mai mic, mai mic sau egal, mai mare, mai mare sau egal
`== !=`	→	Operatori relaționali: egal, diferit
`& ^ | && ||`	→	Operatori logici: ȘI pe biți, SAU-EXCLUSIV pe biți, SAU pe biți, ȘI logic, SAU logic
`? :`	←	Operator condițional
`= *= /= %= += -= &= ^= |= <<= >>=`	←	Operatori de atribuire: atribuire, atribuire cu înmulțire/împărțire/modulo/adunare/scădere/SI/SAU-EXCLUSIV/ SAU/deplasare pe biți
`,`	→	Operatorul virgulă

1.4 Afișarea pe consolă

Pentru transferul datelor de la dispozitivele de intrare sau către dispozitivele de ieșire limbajul C lucrează cu așa numitele fluxuri de intrare/ieșire (en. *stream*) care reprezintă dispozitivele logice asociate dispozitivelor fizice.

Pentru afișarea informațiilor pe consola de ieșire (ecranul este considerat fișierul standard de ieșire - *stdout*) se utilizează funcția "***`printf()`***".

Prototipul funcției printf():

```
int printf ( const char * format, ... );
```

unde:

format – `%[indicatori][lățimea][.precizie][lungime]specificator`

`...` – argumente adiționale. Numărul de argumente trebuie să fie cel puțin egal cu numărul de valori specificate în șirul de formatare (format).

Specificatorii de format reprezintă componenta cea mai importantă într-un șir de formatare. Acești specificatori (Tabel 3) definesc tipul și interpretarea lor în cadrul argumentelor corespondente. Conțin de asemenea o serie de sub-specificatori opționali precum: indicatori, număr minim de caractere (lățimea câmpului), precizie, lungime, cu ajutorul cărora informația este afișată sub diverse forme.

Tabel 3 Specificatori de format

Specificator	Descriere	Exemplu
`d sau i`	***Întreg în zecimal cu semn***	`415`
`u`	***Întreg în zecimal fără semn***	`8420`
`o`	***Întreg în octal fără semn***	`104`
`x, X`	***Întreg fără semn în hexazecimal***	`8ff, 8FF`

f, F	***Real în zecimal***	415.56
e, E	***Real în notație științifică (mantisa/exponent)***	4.1556e+2 4.1556E+2
g, G	***Afișarea cu un număr minim de cifre a numerelor reale reprezentate cu %e, %E, %f și %F***	415.56
a, A	***Număr în hexazecimal în virgulă mobilă***	-0xc.90fep-2 (minuscule) -0XC.90FEP-2 (majuscule)
c	***caracter***	'A'
s	***Șir de caractere***	"Acesta este un text."
p	***Adresă memorată în pointer***	0020a244
n	***Nu se va afișa nimic pe ecran***	
%	***„%” urmat de un alt „%” va avea ca efect afișarea unui singur „%” pe ecran***	%
lf, le	***double***	200.8
Lf, Le	***long double***	200000L

Tabel 4 Indicatori de formatare a datelor afișate pe consolă

Indicator	Descriere	Exemplu
+	***Rezultatul afișat este precedat de semnul minus sau plus chiar și pentru numerele pozitive (implicit doar numere negative sunt precedate de semnul -)***	`int a = 200;` `printf("a = %+d");  // a = +200`

-	***Rezultatul este afișat aliniat la stânga în funcție de câmpul*** `nr_minim_caractere` ***(implicit alinierea este la dreapta)***	`int a = 200;`
		`printf("a = %-10d"); // a = 200`
		`printf("a = %10d");  // a =`
		`200`

spațiu	***Se va insera un spațiu înainte de rezultat***	`int a = 200;` `printf("a = % d");  // a =  200`

#	***Atunci când este utilizat cu specificatorii o, x, X, rezultatul care este diferit de 0 va fi precedat de 0, 0x sau 0X.***
	
	Atunci când este utilizat cu specificatorii a, A, e, E, f, F, g, G, se afisează rezultatul cu virgulă.
		`int a = 200;`
		`printf("a = %x");  // a = c8`
		`printf("a = %#x");  // a = 0xc8`
		
		`float x = 10.50;`
		`printf("\nx = %#.3e", x);// x =`
		`1.050e+001`
		`printf("\nx = %#.3g", x);// x =`
		`10.5`

0		

Pentru formatarea ieșirii se mai utilizează așa numitele *secvențe de evitare* (en. *escape sequences*) care sunt precedate de caracterul ”\” (en. *backslash*) ce conțin caractere neafișabile. Aceste secvențe sunt redate în tabelul de mai jos:

Secvență	Caracter	Descriere
\a	(alarmă)	Sună alarma
\b	BS	Backspace (se va șterge caracterul ce precede \b)
\f	FF	Form feed (se trece la următoarea ”pagină” / secvență)
\n	LF	Line feed (se trece la următoarea linie pe prima poziție – linie nouă)
\r	CR	Carriage return (cursorul se va plasa pe linia curentă, pe prima poziție)
\t	TAB	Tab orizontal
\v	VT	Tab vertical
****	\	Backslash
\’	’	Apostrof
\”	”	Ghilimele

\?	?	Semnul întrebării
\o	caracter	Caractere în notație octală
\xH	caracter	Caractere în notație hexazecimală

1.5 Citirea de pe consolă

Pentru citirea informațiilor de pe consola de intrare (tastatura este considerată fișierul standard de intrare - *stdin*) se utilizează funcția "***scanf()***".

Prototipul funcției scanf():

```
int scanf ( const char * format, ... );
```

unde:

format – `%[*][lățime][lungime]specificator`

`...` – argumente adiționale care indică către obiecte deja alocate conform tipului de dată indicat prin specificatorul de format. Funcția trebuie să conțină cel puțin tot atâtea argumente la fel ca numărul de valori din șirul de formatare. Pentru a memora un rezultat într-o variabilă, numele variabilei trebuie precedat de caracterul "**&**".

Specificatorii de formatare pentru funcția de citire "scanf()" sunt aceiași ca la funcția de afișare "printf()" (vezi Tabel 3). Acești specificatori definesc ce fel de caractere sunt extrase din fluxul de intrare, interpretarea lor și corespondența în cadrul argumentelor.

Exemplu: Rezolvarea ecuației de gradul al II-lea

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

int main(void)
{
      float a, b, c, d, delta;
      printf("Introduceţi coeficienţii ecuaţiei"
             "(aX*2 + bX + c = 0) \n") ;
      printf("\na = "); scanf("%f", &a );
      printf("\nb = "); scanf("%f", &b );
      printf("\nc = "); scanf("%f", &c );
      /*
      *      [ REZOLVARE ]
      */
      if ( a == 0 ) {   //separa de cazul ecuatiei de gr. I
       if ( b == 0 ) {     // evita / la ZERO
        if ( c == 0 )
            printf(     "\nO inf. de solutii ( %d     = 0 )", c);
        else printf("\nFara solutii ( %5.2f =   0 ).", c);
       }
        else
            printf (" \nSolutie ... X = %5.2f", -c/b ) ;
      }
      else  // ... sigur nu este ec. de gr. I
      {
        delta = b*b - 4*a*c;
        if( delta >= 0 ) //Solutii REALE
           printf ("\nX1 = %5.2f\nX2 = %5.2f",
                (-b + sqrt( delta ))/2/a, (-b - sqrt( delta ))/2/a ) ;
        else{      //Solutii COMPLEXE
```

```
39              printf("\nX1 = %5.2f +i %5.2f",-b/2/a, sqrt( -delta )/2/a );
40              printf("\nX1 = %5.2f -i %5.2f",-b/2/a, sqrt( -delta )/2/a );
41            }
42          }
43
44
45          return 0;
46  }//end main()
```

ÎNTREBĂRI

1. *Care sunt tipurile de date fundamentale utilizate în limbajul C?*
2. *Indicaţi regulile de care trebuie să se ţină cont în denumirea variabilelor.*
4. *În ce constă declararea implicită a variabilelor?*
5. *Explicaţi ce presupune declararea explicită a variabilelor.*
6. *Care este operatorul de conversie explicită?*
7. *Dați exemplu de specificatori de formatare pentru funcția de citire de la tastatura?*
8. *Care este secvența de cod pentru a afișa pe ecran textul "Specificatorii de format sunt precedați de caracterul '%'."*

PROBLEME PROPUSE SPRE REZOLVARE

1. Să se scrie un program care să citească de la tastatura un număr întreg 'r' și să afișeze pe ecran lungimea și aria cercului definit de raza 'r'.

Indicație: Lungimea și aria cercului se calculează cu formulele:
$L = 2\pi r$; $A = \pi r^2$

2. Să se afișeze pe ecran mărimea în octeți a următoarelor variabile:

```
char un_caracter='a';
int un_numar=2, vector[10] = {5};
float un_numar_real = 3.14;
double un_alt_numar_real = 5.654;
long int x;
long long y;
```

3. Să se studieze/testeze câteva din funcțiile standard definite în "math.h" (vezi http://www.cplusplus.com/reference/cmath/)

4. Să se scrie un program care să citească de la tastatura un număr real 'x' și să se afișeze rezultatul expresiei |x-4|+|5-x|.

Indicație: Modulul (valoarea absolută) unui număr real în limbajul C este definit prin funcția *fabs*. Pentru numere întregi valoarea absolută se calculează cu funcția *abs*.

5. Să se scrie un program care să afișeze pe ecran următorul text:

```
CARNET NOTE
Data si ora curenta: Tue Mar 10 14:24:48 2015.
Student: [nume student]
```

```
Nr. Crt.        Disciplina      Nota examen
-----------------------------------------
1.              PCLP            10
2.              GAC             7,50
3.              AM              7,75

Media pentru semestrul I este: 8,42.
```

Indicații:

a) Pentru preluarea datei și orei curente se va utiliza funcția `time_t current_time()` definită în fișierul header `"time.h"`, iar pentru convertirea ca șir de caractere se va folosi funcția `ctime(&current_time)` (vezi http://www.cplusplus.com/reference/ctime/);

b) Numele studentului și notele vor fi preluate de la tastatură.