CAPITOLUL 4

4. Tipuri de date predefinite

4.1. Date numerice

Limbajul C este prevăzut cu un set de date predefinite. Aceste tipuri de date sunt concretizate prin folosirea unui număr de locații de memorie, pentru stocarea, conform unui algoritm bine definit funcție de tipul de dată, a unui tip de date existente. În cazul tipului de date char, deşi a fost introdus pentru stocarea datelor de tip alfanumeric, el poate fi folosit în calcule matematice şi logice în intervalul de valori numerice pe care le poate stoca.

Pentru tipurile de date ce stochează valori numerice întregi (prescurtat int) sau , după cum am spus char, numărul este stocat în memorie direct sub forma rezultată la conversia în binar.

În cazul datelor de tip real, float, sau double la conversia directă a numărului în baza doi s-ar fi pierdut prea multe locații de memorie şi de aceea ele se păstrează sub forma ştiințifică. Această formă se referă la faptul că există o zonă de memorie care ține mantisa şi o zonă de memorie care ține exponentul. Un exemplu de reprezentare ştiințifică din punct de vedere matematic ar fi:

$$123344445455454=1.23344445455454 * 10^{14} =1.23344445455454E+14$$

unde mantisa=1,23344445455454 iar exponentul = 14.

Mai jos prezentăm o serie de tipuri de date predefinite, dimensiunile lor şi domeniul valorilor care pot fi stocate. Se observă că s-au folosit modificatori de tip. Prim modificatori de tip se înțeleg cuvinte cheie care au ca efect, la compilare, ca dimensiunea (în octeți) pe care este reprezentat un tip de date să crească (cu un factor care este 2 sau o putere a lui doi)).

char ∈ {–128, 127}	1 octet
unsigned char ∈ {0, 255} fără semn	1 octet
signed char ∈ {–128, 127} semn	1 octet cu
int ∈ {–32768, 32767 } cuvânt (2 octeti)	1
unsigned int ∈ {0, 65535} semn	1 cuvânt fără
signed int ∈ {–32768, 32767} semn	1 cuvânt cu
short int ∈ {–32768, 32767} semn	1 cuvânt cu
unsigned short int ∈ {0, 65535} semn	1 cuvânt fără

signed short int –32768 to 32767 1 cuvânt cu semn
long int ∈ {–2147483648, 2147483647} 2 cuvinte
signed long int ∈ {–2147483648, 2147483647} 2 cuvinte cu semn
float ∈ {–3.4E–38, 3.4E+38} 2 cuvinte
double ∈ {–1.7E–308, 1.7E+308} 4 cuvinte
long double ∈ {–3.4E–4932, 1.1E+4932} 5 cuvinte

Observație: Aceste intervale sunt de obicei specifice sistemului de operare sub care lucrează compilatorul şi maşinii pe care acesta este executat. Domeniile de mai sus sunt valabile pentru compilatorul de C++ realizat de Borland.

4.1.1 Operatorul virgulă

a,b ⇔ evaluează a.evaluează b
Acest operator permite evaluarea a două sau mai multe expresii distincte in locurile in care o singură expresie este admisă a fi folosită(asta e rațiunea introducerii lui)
De ex:

for (j=0,k=100 ; k-j>0 ; j++,k--) este chivalent cu

```
j=0 ; k=100;          j=0 , k=100
{                     {
…                      …
j++;          sau     j++;
k--;                  k--;
}                     }
```

Un alt exemplu funcția break_line:

```
break_line (int interval)
    { int c,j;
      for (c=getchar(),j=0 ; c!=EOF ; c=getchar (),putchar (c),j++)
      if (j % interval = = 0 ) printf( "\n" ) ;
    }
```

4.2. Date alfanumerice

În cadrul proiectării unei aplicații pot apare pe lângă cerințe legate de calcule numerice complexe şi cerințe pentru procesarea de texte sau a unor baze de date.

Totalitatea caracterelor ce pot fi afişate sau listate în cazul lucrului sub un sistem de operare clasic (UNIX sau DOS), se numesc caractere alfanumerice (alfabetice şi numerice). Fiecare caracter are o imagine pe ecran. Această imagine este stocată în memoria nevolatilă a calculatorului şi are ataşat un cod unic la care se face apel când se doreşte afişarea sau listarea ei.

Datorită necesității ca un text editat pe un sistem fabricat de o anume firmă să-şi păstreze înțelesul s-a trecut la standardizarea codării imaginilor caracterelor. Astfel a rezultat standardul ASCII folosit la ora actuală de toate calculatoarele din seria X86.

Există 255 de caractere în acest standard. Nu vom face referiri la seturile extinse de caractere specifice editoarelor de teste sau sistemelor de operare de tip Windows sau XWindows.

Pentru stocarea unui şir de caractere limbajul C foloseşte un tablou monodimensional de caractere care poate stoca codurile a maximum 255 de caractere o dată în cazul declarării statice.

Observația 1: Dacă s-a definit variabila s ca având capacitatea de stocare de 25 de caractere iar la citirea ei sunt introduse mai mult de 25, atunci numai pentru primele 25 nu există riscul pierderii lor pe parcursul execuției programului. În acest caz pot apărea erori mascate în cod.

Observația 2: În cazul în care nu am definit numărul de caractere care poate fi stocat în respectiva variabilă, funcție de modul de construcție al compilatorului, fie se alocă dimensiunea maxim posibilă, fie la prima inițializare se alocă exact cât este necesar dar nu peste maximul definit.

Observația 3: Nu există nici o legătură între valoarea numerică x şi caracterul x care apare pe ecran. De exemplu 2 ≠ '2'.

Inițializarea unei date cu o valoare utilă poate avea loc în următoarele moduri:

- la declararea ei;
- la citirea datelor de intrare;
- oriunde în program .

4.2.1. Caractere speciale de control

Pe lângă caracterele direct afişabile mai există în limbajul C o serie de caractere de control. Acestea provin din necesitatea de control a unei imprimante şi simultan au fost folosite şi la afişarea pe terminalele alfanumerice.

Secvența	Cod	Caracter
\a	07h	BEL (alert produce un semnal sonor)
\b	08h	BS (backspace)
\f	0Ch	FF (form feed mută cursor la următoarea pagină)
\n	0Ah	LF
\r	0Dh	CR (carriage return, salt la inceput de linie nouă)
\t	09h	HT (horizontal tab)
\v	0Bh	VT (vertical tab)
\\	5Ch	\ (backslash)
\'	27h	' (apostrof)
\"	22h	" (ghilimele)
\?	3Fh	? (semnul întrebării)
\ooo	ooo	Caracterul cu codul ooo în octal
\xhh	hh	Caracterul cu codul hh în hexa

Pentru justificarea introducerii acestor caractere de control se va prezenta următorul caz: era inutilă trimiterea uni număr fix de caractere identice de fiecare dată când se dorea un tabulator (un anumit număr de spații goale) la începutul unui paragraf şi atunci s-a convenit să se folosească un caracter de control, urmând ca imprimanta să-l reconvertească intern în spații goale. Anterior s-a prezentat un tabel ce conține aceste caractere.

4.3. Operațiuni simple de citire sau afişare a datelor

Limbajul C are la dispoziție o serie de funcții pentru citirea datelor de la tastatură sau pentru afişarea lor pe ecran. Cele mai folosite dintre ele sunt grupate în două biblioteci conio.h (CONsoleInputOutput.Header) şi (StandardInputOutput.Header) stdio.h.

4.3.1. Funcții de intrare sau ieşire cu utilizatorul

Pentru a putea discuta primele programe este necesar studierea modului în care un program poate schimba date dispozitivele standard de intrare sau ieşire (monitor, tastatură). Din acest punct de vedere există două clase de funcții:

- **Funcții de intrare:** cu ajutorul cărora sistemul de calcul primeşte de la tastatură valori numerice, comenzi şi oricare alt tip de informație ce poate fi furnizată prin intermediul acelui dispozitiv.
- **Funcții de ieşire:** cu ajutorul cărora sistemul de calcul livrează rezultatele (afişează) folosind un dispozitiv standard de ieşire (monitor sau imprimantă).

Pentru a citi o dată sau un şir de date de la tastatură avem o funcție specializată care poate citi orice tip predefinit de date. Funcția are prototipul prezentat mai jos

*int scanf (const char *format [,address, ...]);*

Tipul de date este specificat de către programator prin modificarea unor parametri ai funcției. Mai jos este dat un scurt exemplu:

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

void main (void)
    {
    int a;
    char c;
    char b[20];
    float d;
    clrscr(); // şterge ecranul
    scanf("%d", &a); // citeşte un întreg şi-l depozitează în
    // variabila a
    scanf("%c", &c); // citeşte un caracter
    scanf("%s", b); // citeşte un şir de caractere
    scanf("%f", &d); // citeşte un număr zecimal
    getch();
    }
```

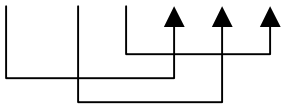
Această funcție, la apelare, preia informația introdusă de la tastatură după apăsarea tastei ENTER. Ca efect utilizatorul vede doar că programul îşi opreşte derularea şi

aşteaptă ceva, dar fără a primi un mesaj care să îi comunice ce trebuie să facă sau informații despre natura datelor care trebuiesc livrate.

Mai jos este prezentat prototipul unei funcții des folosită în afişarea diverselor tipuri de date.

*int printf (const char *format [, argument, ...]);*

Ex: printf("Afişează cele 3 numere: %d %d %d",nr1,nr2,nr3);



Vom prezenta un prim exemplu despre unele din posibilităție de afişare ale funcției printf .

```
#include <stdio.h>

main()
    {
    int a;          /*întreg simplu              */
    long int b;       /* întreg lung               */
    short int c;      /* întreg scurt              */
    unsigned int d;    /* întreg fără semn          */
    char e;          /* caracter simplu           */
    float f;         /*număr zecimal în  simplă precizie          */
    double g;         /*număr zecimal în dubă precizie     */
      a = 1023;
      b = 2222;
      c = 123;
      d = 1234;
      e = 'X';
      f = 3.14159;
      g = 3.1415926535898;
      printf("a = %d\n",a);  /* afişare folosind baza 10 de numerație      */
      printf("a = %o\n",a);     /* afişare folosind baza 8 de numerație */
      printf("a = %x\n",a);     /* afişare folosind baza 16 de numerație */
      printf("b = %ld\n",b);    /* afişare specifică pentru întreg lung      */
      printf("c = %d\n",c);     /* afişare specifică pentru întreg scurt */
      printf("d = %u\n",d);    /* afişare specifică pentru întregfără semn */
      printf("e = %c\n",e);     /* afişare specifică pentru caracter */
      printf("f = %f\n",f);     /* afişare specifică pentru număr în simplă precizie */
      printf("g = %f\n",g);     /* afişare specifică pentru număr în dublă precizie */
      printf("\n");                  /*salt la line nouă*/
      printf("a = %d\n",a);     /* afişare specifică pentru întreg */
      printf("a = %7d\n",a);    /* afişare specifică pentru un  întreg dar alocând 7
    caractere în dreapta */
```

```
   printf("a = %-7d\n",a);   /* afişare specifică pentru un  întreg dar
                              alocând 7 caractere în stânga */
   c = 5;
   d = 8;
   printf("a = %*d\n",c,a);   /* folosim un câmp de  5 caractere     */
   printf("a = %*d\n",d,a);   /* folosim un câmp de  8 caractere */
   printf("\n");
   printf("f = %f\n",f);  /* afişare pentru număr în simplă precizie       */
   printf("f = %12f\n",f);    /* folosim un câmp de  12 caractere */
   printf("f = %12.3f\n",f);  /* folosim un câmp de  3 caractere după
                              virgulă */
   printf("f = %12.5f\n",f); /* folosim un câmp de  5 caractere după
                              virgulă */
printf("f = %-12.5f\n",f); /* afişare specifică pentru un  număr în
                 simplă precizie dar alocând 7 caractere în stânga */
}
```

Să reluăm exemplul prezentat la citirea datelor introducând de această dată şi mesajele pentru utilizator.

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

void main (void)
    {
    int a;
    char c;
    char b[20];
    float d;
    clrscr(); // şterge ecranul
    printf("Introduceti o valoare intreaga")
    scanf("%d", &a); // citeşte un întreg şi-l depozitează în //   variabila a
    printf("\nValoarea citita este %d", a);
    printf("\nDati un caracter ")
    scanf("%c", &c); // citeşte un caracter
    printf("\nCaracterul citit este %c",c);
    prinf("\nDati un sir de caractere ");
    scanf("%s", b); // citeşte un şir de caractere
    printf("\nS-a citit urmatoarea propozitie %s",b);
    printf("\nDati un numar zecimal");
    scanf("%f", &d); // citeşte un număr zecimal
    printf("\n Sa citi numarul %f",d)
   printf("\nApasa orice tasta pentru a continua")
   getch();
    }
```