2. TIPURI DE DATE. FUNCȚIILE DE CITIRE ŞI DE SCRIERE A DATELOR

2.1. Constante şi variabile

Datele de diverse tipuri (întregi, reale, caractere sau şir de caractere) folosite în cadrul programului de calcul sunt clasificate în constante şi variabile. O *constantă* este un spațiu din memoria internă a calculatorului destinat memorării unui anumit tip de date al cărui conținut nu poate fi schimbat pe parcursul execuției programului de calcul, pe cînd o *variabilă* reprezintă tot un spațiu din memoria calculatorului, destinat memorării unui anumit tip de date dar al cărui conținut poate fi modificat în timpul execuției programului. Pentru a putea fi uşor identificate, variabilelor li se asociază un nume şi un tip, care se referă la tipul valorilor pe care le memorează. În programul 2.1 este folosită funcția *printf* pentru a tipări constanta întreagă 10 şi constanta reală 3.14159.

Exemplu 2.1.

```
/* Programul ex_2_1*/
#include <stdio.h>
void main (void)
{
        printf("\n Aceasta este constanta intreaga %d",10);
        printf("\n Aceasta este constanta reala %f",3.14159);
}
```

Această variantă nu este utilizată, deoarece acelaşi rezultat se obține mai uşor cu programul din exemplu 2.1.a.

Exemplu 2.1.a.

```
/* Programul ex_2_1_a */
#include <stdio.h>
void main (void)
{
        printf("\n Aceasta este constanta intreaga 10");
        printf("\n Aceasta este constanta reala 3.14159");
}
```

Nici această variantă nu este avantajoasă. În exemplu 2.1.b. este scris programul anterior utilizînd variabile în locul constantelor. În prima parte se creează 2 variabile (una de tip întreg, alta de tip real) care capătă pentru început valorile 10, respectiv 3.14159. Acestea vor fi afişate cu ajutorul a 2 apeluri ale funcției *printf*. Apoi, valorile variabilelor sunt modificate, noile valori fiind afişate cu ajutorul celor 2 apeluri ale funcției *printf* din finalul programului.

Exemplu 2.1.b.

```
/* Programul ex_2_1_b */
#include <stdio.h>
void main (void)
{
        int intreg;     float real;
        intreg =10;     real =3.14159;
        clrscr( );
        printf("\nVariabila intreg are valoarea %d",intreg);
        printf("\nVariabila real are valoarea %f",real);
        intreg = intreg + 5;
        real = real - 3;
        printf("\n Noua valoare a variabilei intreg %d",intreg);
        printf("\n Noua valoare a variabilei real %f",real);
        getch( );
}
```

Prin execuția acestui program, pe ecran se va afişa următorul text:

Variabila intreg are valoarea 10
Variabila real are valoarea 3.14159
Noua valoare a lui intreg este 15

Noua valoare a lui real este 0.14159

în primele 2 instrucțiuni se folosesc cuvintele cheie int şi float pentru a crea variabila de tip întreg şi respectiv, variabila de tip real, al cărei nume este *real*. Efectul execuției acestor instrucțiuni este de a rezerva zonele de memorie în care se vor stoca valorile ce vor fi atribuite variabilelor în timpul execuției programului şi de a atribui acestor zone numele respective. Pentru definirea numelor variabilelor se poate folosi orice combinație de cifre şi litere, primul caracter fiind literă. De obicei, variabilelor li se atribuie nume semnificative, putîndu-se utiliza despărțirea în silabe cu ajutorul caracterului _(underbar). în exemplul 2.1.b mai semnificativ ar fi fost utilizarea în locil lui *real* a numelui *pi*.

Următoarele 2 instrucțiuni din cadrul programului ex_2_1_b atribuie variabilei *intreg* valoarea 10, iar variabilei *real* valoarea 3.14159, folosind operatorul de atribuire "=". Valorile sunt apoi afişate pe ecran folosind funcția printf, iar în locul constantelor - numele variabilelor. în continuare, valorile celor două variabile sunt modificate folosind operatorul de atribuire şi operatorii aritmetici "+" şi "-". Noile valori sunt afişate cu ajutorul funcțiilor printf.

2.2. Declararea variabilelor. Tipuri de date

O declarație de variabilă cuprinde două părți. Prima parte conține tipul variabilei şi este un cuvînt care face parte din setul de cuvinte cheie ale limbajului C. Acestea nu pot fi folosite ca nume de variabile. în exemplul prezentat am folosit pentru a desemna tipurile de variabile întregi şi reale, cuvintele cheie int şi float. A doua parte, separată de prima prin unul sau mai multe spații, conține numele variabilei. Dacă există mai multe variabile de acelaşi tip, acestea pot fi scrise pe o singură linie, separate prin virgulă. Exemplu:

float real, pi, real_z, imag_z;

Precizăm faptul, că în limbajul C nu există tipuri de variabile predefinite. Deci, înainte de a fi utilizate, toate variabilele vor fi declarate, pentru a specifica tipul şi numele lor şi pentru a rezerva spațiul de memorie necesar memorării valorilor.

2.2.1. Variabile caracter (tipul char)

O variabilă de tip caracter este definită memorării unui singur caracter într-un octet de memorie. Declararea variabilelor caracter se face cu ajutorul cuvîntului cheie char. Astfel, pentru a declara 2 variabile de tipul caracter *car1, car2* vom folosi instrucțiunea: *char car1, car2;*.

Folosind un octet de memorie, o variabilă de tipul caracter poate memora valori întregi cuprinse între 0 şi 255. în funcție de contextul în care este folosită o variabilă de tipul caracter, conținutul acesteia poate fi interpretat ca un întreg în sine sau ca un întreg reprezentînd codul ASCII al unui caracter. Pentru exemplificare, considerăm următorul program:

Exemplul 2.2

```
/* Programul ex_2_2 */
#include <stdio.h>
void main (void)
 {
        char car;
        car = 65;
        printf("\nContinutul variabilei car este %3d si reprezinta caracterul %c",car,car);
        car = car+2;
        printf("\nNoul continut al variabilei car este %3d si reprezinta caracterul %c",car,car);
 }
```

în cadrul acestui program variabilei *car i*-am atribuit valoarea 65 (codul literei A). în apelul funcției printf conținutul variabilei *car* va fi interpretat mai întîi ca un întreg, apoi ca un caracter. în continuare, valoarea variabilei *car* este modificată, noul conținut fiind afişat. în urma execuției programului ex_2_2 pe ecran va fi afişat textul:

Continutul variabilei car este 65 si reprezinta caracterul A

Noul continut al variabilei este 67 si reprezinta C.

Pentru atribuirea de valori unei variabile de tipul caracter se pot folosi constante întregi (în exemplul nostru 65) sau constantele caracter, care constau din caracterul propriu-zis încadrat de 2 caractere apostrof. Astfel, în locul instrucțunii *car=65;* se poate folosi instrucțiunea *car='A';* rezultatul obținut fiind acelaşi.

2.2.2. Variabile întregi (tipul int)

O variabilă de tipul întreg este destinată memorării unei valori întregi cuprinsă între -32768 şi 32767, folosind 2 O de memorie. Declararea variabilelor de tipul întreg se face cu ajutorul cuvîntului cheie int urmat de

numele variabilelor. Dacă valorile ce urmează a fi memorate sunt nenegative în instrucțiunea de declarare se foloseşte prefixul unsigned, care are ca efect translatarea domeniului de valori ce pot fi memorate în intervalul 0-65535. Exemplu: *unsigned int ival;*. în cazurile, în care cei 2 octeți rezervați pentru memorarea valorilor în variabilele de tip întreg sunt insuficienți (valori mai mari decît 32767 sau mai mici decît -32768) în instrucțiunea de declarare se foloseşte prefixul long. O declarație de forma: *long int ivall;* sau *long ival1;* declară variabila *ival1* de tipul întreg rezervîndu-i un spațiu de 4 O în memorie. în acest fel, pot fi utilizate valori întregi, cuprinse între -2147483648 şi 2147483647.

2.2.3.Variabile reale (tipul float şi tipul double)

O variabilă de tipul real este destinată memorării valorilor reale în formă exponențială. Exprimarea numerelor reale în formă exponențială constă în transformarea numărului într-o valoare, numită mantisă, avînd o singură cifră în stînga punctului zecimal urmată de litera e şi valoarea exponenrului. De exemplu: 54231 se scrie sub forma 5.4231e4, în care numărul 4 indică puterea lui 10 cu care trebuie înmulțită mantisa pentru a obține valoarea adevărată a numărului. Exponentul poate fi şi negativ: *3.25e-3;* Utilizînd exprimarea exponențială, în variabile reale pot fi stocate valori mult mai mari decît în variabile întregi sau mult mai mici (ordinele de mărime cuprinse între 10^{-38} şi 10^{38}). Operațiile cu numerele reale se desfăşoară mai încet decît cu numerele întregi şi sunt aproximative.

În limbajul C există 2 modalități de a declara varebelile reale: float şi duble.

O declarație de tipul:*float fval;* declară variabila fval de tipul real, rezervînd 4 O de memorie pentru memorarea valorilor: 3 O sunt folosiți pentru memorarea mantisei, cu o precizie de 7 cifre, iar unul pentru memorarea exponentului.

A doua modalitate de declarare a variabilelor de tip real o constitue utilizarea cuvîntului cheie double. O declarație de tipul *double dval;* declară variabila reală *dval*, rezervînd 8 O pentru memorarea variabelilor. Creşterea numărului de O destinați memorării valorilor reale de tipul double are ca efect pe de o parte creşterea domeniului de valori memorate(de la 10^{-306} pînă la 10^{306}), iar pe de altă parte creşterea preciziei de reprezentare exponențială la 15 cifre exacte.

2.3. Adresele şi inițializarea variabilelor

Memoria internă a calculatorului este organizată pe celule numite octeți, care sunt notate de la 0 la capacitatea maximă a memoriei. Aceste numere ataşate celululelor poartă denumirea de adrese. La declararea variabilelor, pe lîngă faptul că se precizează numele şi tipul acestora li se rezervă şi un spațiu în memorie, deci li se ataşază o adresă. Pentru variabilele de tipul caracter, adresa este chiar adresa octetului, în timp ce pentru celelalte tipuri, adresa este adresa primului octet din cele 2, 4 sau 8 pentru memorarea valorilor în funcție de tipul variabilei.

Din cele prezentate putem concluziona că o variabilă este:

- o valoare care poate fi modificată în timpul execuției programului;
- un tip care determină mărimea zonei de memorare rezervată pentru memorarea valorilor;
- o adresă care reprezintă locul în memoria internă, unde urmează a fi memorate valorile.

în limbajul C toare cele trei elemente asociate unei variabile sunt accesibile programatorului. Astfel, valoarea variabilei este furnizată de numele acesteia, adresa se obține folosind în fața numelui variabilei caracterul "&", numit operator de adresă, iar numărul de octeți se obține prin apelul funcției sizeof.

Inițializarea variabilelor este operația prin care acstora li se atribuie valori inițiale. Aceasta se poate face în cadrul unor instrucțiuni separate, folosind operatorul de atribuire, cum s-a procedat în exemplele 2.1.b şi 2.2,sau chiar în cadrul instrucțiunilor de declarare, aşa cum se poate constata din exemplul 2.3.

Exemplul 2.3.

```
/*programul ex_2_3*/
# include <stdio.h>
# include <conio.h>
void main (void)
 {
        char car='A';           int ival=1230;
        long lval=45123;        float fval=7235.1259; /*notație obişnuită*/
        double dval=7.2351259e3;                      /*notație exponențială*/
        clrscr ( );       gotoxy(34,6);  printf("Exemplul 2.3");
        gotoxy (18,8); printf ("Tipurile fundamentale de date ale limbajului C");
        gotoxy(8,10);  printf ("Tip: char continut: %c Adr.: %p Nr. O %d",car,&car,sizeof(char));
        gotoxy (8,11); printf ("Tip: int continut: %d Adr.: %p Nr. O %d", ival,&ival,sizeof(int));
```

```
        gotoxy (8,12); printf (”Tip: long continut:%d Adr.:%p Nr. O %d”,lval,&lval,sizeof(long));
        gotoxy (8,13); printf (”Tip: float continut:%d Adr.:%p Nr. O %d”,fval,&fval,sizeof(float));
        gotoxy (8,14); printf (”Tip: double continut:14.10e Adr.:%p Nr. O %d”,dval,&dval,sizeof(double));
    }
```

Orice program de calcul pe lîngă variabile şi instrucţiuni poate conţine şi comentarii. în limbajul C orice şir de caractere încadrat de /*, respectiv */, reprezintă un comentariu şi poate să apară pe o linie separată sau în continuarea unei instrucţiuni. O altă modalitate de introducere a comentariilor, recunoscută de Turbo C++/Borland C, o reprezintă utilizarea caracterelor //, urmate de şirul de caractere ce desemnează comentariul.

Variabilele trebuie iniţializate înainte de a fi utilizate. în program s-au folosit funcţiile clrscr şi gotoxy, care sunt funcţii ale mediului de programare Turbo C++/Borland C. Acestea sunt funcţii, specifice modului de lucru text, care au ca efect ştergerea ecranului(clrscr), respectiv poziţionarea cursorului într-o poziţie dorită pe ecran (gotoxy).

în urma execuţiei programului pe ecran va fi afişat următorul text:

Exemplul 2.3.

TIPURIL;E FUNDAMENTALE DE DATE ALE LIMBAJULUI C

Tip: char Continut: A Adr.: 0FBD Nr. O 1
Tip: int Continut: 1230 Adr.: 0FBE Nr. O 2
Tip: long Continut: 45123 Adr.: 0FC0 Nr. O 4
Tip: real Continut: 7.235125977e+3 Adr.: 0FC4 Nr. O 4
Tip: doyble Continut: 7.235125900e+3 Adr.: 0FC8 Nr. O 8

în ceea ce priveşte conţinutul variabilelor *fval* şi *dval*, se constată că, deşi au fost utilizate cu aceleaşi valori, la afişarea ele diferă prin utimile cifre zecimale. Acest lucru este datorat faptului că, aşa cum au spus, precizia de reprezentare a variabilelor de tipul float este mai mică decît precizia variabililor de tipul double.

2.4. Citirea şi scrierea datelor

Operaţiile de citire şi scriere a datelor constau dintr-un ansamblu de activităţi prin care valorile ce urmează a fi prelucrate de programul de calcul sunt introduse în memoria internă, respectiv rezultatele obţinute sunt extrase şi prezentate utilizatorului. Aceste operaţii poartă denumirea de operaţii de citire/scrise sau intrare/ieşire (input/output).

Introducerea datelor (input) se poate efectua fie de la tastieră, prin acţiuni ale utilizatorului, fie din memoria externă, unde datele sunt păstrate pe termen lung sub formă de fişiere.

Rezultatele obţinute într-un program sunt extrase (output) din memoria internă, pentru a fi prezentate utilizatorului pe ecranul monitorului sau /şi pentru a fi salvate în memoria externă în vederea unor afişări sau prelucrări ulterioare.

2.4.1. Afişarea datelor - printf

în limbajul C funcţia printf este funcţia standard C prin intermediul căreia rezultatele obţinute în urma procesului de calcul sunt afişate pe ecranul monitorului. Forma generală de apel a ei:

printf(constantă şir de caractere,lista de valori);

Tabelul 2

Descritor de format	Tipul de date pentru care se foloseşte
%c	Caracter
%s	şir de caractere
%d	întreg zecimal cu semn
%f	real (notaţie zecimală)
%e	real (notaţie exponenţială)
%g	real (selectează care notaţie este mai scurtă)
%u	întreg zecimal fără semn
%x	întreg hexazecimal fără semn
%o	întreg octal fără semn
%p	adresa hexazecimală

O constantă şir de caractere este o succesiune de litere, cifre, spaţii albe şi caractere speciale delimitate de o pereche de ghilimele. Primul parametru al funcţiei printf este o construcţie de acest tip şi conţine textul ce va fi afişat pe ecran, precum şi descriptorii de format. Fiecare descriptor de format poate fi plsat oriunde în şirul de caractere şi constă din caracterul % urmat de una sau două litere. După cum se ştie, în memoria internă valorile sunt reprezentate în formă binară. Rolul descriptorilor de format este de a specifica pe de o parte

conversia de efectuare asupra valorilor, preluate din memoria internă, înainte ca acestea să fie inserate în textul ce va fi afişat pe ecran, iar pe de altă parte - locul şi modul în care valorile astfel convertite vor fi inserate în text. Spunem că după conversie valorile se substituie descriptorilor de format în şirul de caractere care va fi afişat pe ecran. Acest mecanism impune ca între descriptorii de format şi lista de valori să existe o corespondență biunivocă, adică fiecărei valori să-i corespundă un descriptor de format. în tabelul 2 sunt prezentați descriptorii de format ce pot fi utilizați în apelul funcției printf pentru afişarea diverselor tipuri de valori.

Prefixul l este utilizat cu descriptoriii de format %d, %u ,%x, %o pentru Întregii de tip long (de exemplu %ld), respectiv cu descriptorii de format pentru realii de tip double.

Lista de valori care constituie al doilea parametru al funcюiei printf, poate fi alcrtuitr dun constante şi variabile de orice tip sau chiar din expresii aritmetice sau relaюionale. Pentru a explica modul de lucru al funcюiei printf analizrm instrucюiunea *printf("Aceasta este constanta %d",10);* care are rolul de a afişa pe ecran textul: Aceasta este constanta 10. Mai ÎntÎi, valoarea constantei este convertitr conform descriptului de format %d specific valorilor de tip Întreg, valoarea obюinutr este inseratr În text În locul descriptorului de format si apoi textul rezultant este afişat pe ecran.

În afara faptului,cr permite afişarea tuturor tipurilor de date, funcюia printf oferr programatorului multe facilitrюi de control pentru afişarea pe ecran. Astfel, pentru valorile de tipul Întreg sau real, În descriptorii de format, dupr caracterul % se poate specifica mrrimea cÎmpului (numrrul de celule de pe ecran rezervat pentru afişarea valorii), precum şi numrrul de zecimale cu care se vor afişa valorile reale. Fie cr e nevoie de afişat pe ecran un tabel care sr conюinr dour coloane. Prima coloanr conюine valori intregi (numrrul Însrsi) şi are lrюimea de 6 spaюii. A doua conюine valori reale care vor fi afişate cu trei zecimale şi cu lrюimea 12. Acest lucru poate fi realizat cu ajutorul urmrtorului program:

Exemplu 2.4

```
/*programul ex_2_4*/
#include <stdio.h>
void main(void)
 {
        int i;    float fval=5.025;
        for (i=1; i<=10; i++)
                printf("\n%6d\t%12.3f",i,fval+(i-1)*1.5);
 }
```

Elementul de bazr În cadrul acestui program Îl constituie apelarea repetatr a funcюiei printf pentru afişarea celor dour coloane. În acest sens am folosit instrucюiunea ciclicr for. Funюia printf va fi apelatr de 10 ori pentru i=1,...,10. Remarcrm, cr În lista de valori pe lÎngr variabila *i* s-a folosit şi expresia *fval+(i-1)*1.5*. Pentru a construi textul ce se va afişa pe ecran funcюia printf substituie valoarea variabilei *i* descriptorului %6d , apoi evaluează valoarea exprisiei şi o substituie descriptorului de format %12.3f.

În urma rulrri acestui program, pe ecran va fi afişat urmrtorul tabel :

1 5.025
2 6.525
3 8.025
. . . . . .
9 17.025
10 18.525

Tabelul 3

Secvența	Semnificația
\n	Salt la o linie nouă (new line)
\t	Deplasare la dreapta (tab)
\b	Deplasare la stînga cu o poziție (backslash)
\r	întoarcere la cap de rînd (CR)
\f	Trecere pe linia următoare (formfeed)
\’	apostroful
\”	Ghilimelele
\\	Backslash
\xdd	Codul ASCII în cifre hexazecimale

După terminarea ciclului valoarea atribuitr inirial variabiei *fval* nu se va modifica. Se constatr deasemenea cr valorile din cele 2 coloane sunt aliniate la dreapta În cadrului cîmpului de afişare. Prin utilizarea semnului minus În faroa mrrimii cÎmpului ,valorile vor fi aliniate la stÎnga În cele dour coloane.

În şirul de caractere care constituie primul parametrul funcroiei printf, pe lÎngr descriptorii de format se folosesc comenzi de control, numite secvenroe escape. Pentru aceasta se foloseşte simbolul \, numit backslash, care determinr o semnificaroie diferitr de cea obisnuitr a caracterelor ce Îl urmeazr. Astfel secvenroa \n are ca efect trecerea pe linia urmrtoare (new line), iar \t deplaseazr la dreapta cu un numrr de coloane (tabulare).

UtilizÎnd secvenroele escape se pot afişa caracterele care În mod obişnuit au altr semnificaroie. Este cazul apostrofului utilizat pentru definirea constantelor caracteriale, a gilimelelor, folosite pentru constantele şiruri de caractere şi backslash-ului. În tabelul 3 sunt prezentate secvenroele escape şi semnificaroiia lor.

Fiecare caracter este reprezentat În calculator printr-un numrr. Numerele de la 0 la 127 formeazr contururile tuturor literilor mari şi mici, ale cifrelor de la 0 la 9 şi alte caractere de control. Calculatoarele IBM compatibile folosesc 128 de caractere. Acestea constau din caractere grafice şi simboluri, folosite În limbaje strrine sau În matematicr, avÎnd codurile cuprinse Între 128 şi 255. Pentru aceasta se utilizeazr secvenroa \xdd. Utilizarea secvenroei \xdd În cadrul funcroiei printf este o modalitate de a crea imagini grafice folosind codurile hexazecimale din extensia IBM.

Astfel, în prgramul din exemplul 2.5 este prezentat modul cum poate fi trasat un dreptunghi pe ecran.

Exemplu 2.5

```
/*programul ex_2_5*/
#include <stdio.h>
#include <conio.h>
void main (void)
 {
        clrscr( );
        gotoxy(20,5);  printf("\xC9\xCD\xCD\xCD\xCD\xCD\xBB");
        gotoxy(20,6);  printf("\xC8\xCD\xCD\xCD\xCD\xCD\xBC");
 }
```

2.4.2. Citirea datelor - scanf

Funcția scanf este perechea funcției printf în cadrul operațiilor de intrare/ieşire, facilitînd introducerea datelor de la tastieră în memoria internă. Pentru exemplificare considerăm următorul program în care se citeşte o valoare reală de la tastieră, care exprimă temperatura în grade Celsius şi pe care vrem să o afişăm în grade Kelvin.

Exemplu 2.6

```
/*programul ex_2_6*/
#include <stdio.h>
void main (void)
 {
        float temp;
        printf("\n Introduceti temperatura in grade Celsius");        scanf"%f",&temp);
        printf("\n Temperatura in grade Kelvin este %f",temp+273.15);
 }
```

în primul apel al funcției printf nu am folosit descrriptorul de format şi nici lista de variabile. Efectul unui astfel de apel constă în afişarea şirului de caractere pe ecran şi constituie modalitatea prin care programul transmite mesajul utilizatorului.

Aşa cum se observă, forma funcției scanf este asemănătoare cu cea a funcției printf. Astfel, primul parametru este tot o constantă şir de caractere ca re reprezintă descriptorul de format. Deosebirea esențială dintre cele 2 funcții constă în faptul că, spre deosebire de funcția printf care utilizează numele variabilelor (conținutul acestora), scanf foloseşte adresa variabilelor. Acest lucru este normal deoarece scanf primeşte de la tastieră valoarea introdusă conform descriptorului de format şi o depune în zona de memorie rezervată variabilei.

Funcția scanf poate citi în cadrul unui singur apel, a cărui formă generală este:

scanf(constantă şir de caractere,lista adreselor variabilelor);

mai multe valori pentru variabile de diverse tipuri. Considerăm programul, în cadrul căruia se citesc 3 valori (un caracter, un întreg şi un real) printr-un singur apel al funcției scanf.

Exemplu 2.7

```
/*programul ex_2_7*/
```

```
#include <stdio.h>
void main (void)
 {
        char car;       int ival;               float fval;
        printf("\n Introduceti un caracter, un intreg si un real ");    scanf"%c %d %e",&car,&ival,&fval);
        printf("\n Valorile introduse sunt: %c\t%d\t%f",car,ival,fval);
```

Tabelul 4

Tipul variabilei	Descritori de format	
	printf	scanf
Caracter	%c	%c
şir de caractere	%s	%s
întreg zecimal cu semn	%d	%d
real în notație zecimală	%f	%f
real în notație exponențială	%e	%e
întreg zecimal fără semn	%u	%u
întreg hexazecimal fără semn	%x	%x
întreg octal fără semn	%o	%o

```
}
```

Cînd introducem datele, acestea vor fi separate între ele printr-un spațiu. Aceste spații sunt în concordanță cu spațiile ce separă descriptorii de format şi constituie modalitatea prin care funcția scanf detectează sfîrşitul unei valori, respectiv începutul altei valori. Ca spațiu de separare la introducerea datelor se poate folosi orice spațiu alb recunoscut în limbajul C, adică spațiul obişnuit (blank), tastele <Tab> sau <Enter>. Indiferent de varianta aleasă pentru introducerea valorilor, ultima acțiune constă în apăsarea tastei <Enter>.

Descriptorii de format folosiți în cadrul funcției scanf sunt practic aceiaşi cu cei folosiți de funcția printf. Există unele deosebiri, care sunt prezentate în tabelul 4.

în acest tabel nu apare descriptorul *%g*, care este specific funcției printf. în apelul funcției scanf pentru întregii de tipul lorg se foloseşte descriptul *%ld*, iar pentru realii de tip double, descriptorul *%lf*.

La testarea valorilor variabilelor, funcția scanf le preia şi în acelaşi timp afişează pe ecran caracterele asociate tastelor apăsate. Spunem că apăsarea tastelor se face cu ecou. Există situații în care se doreşte citirea unui caracter de la tastieră fără a se face ecoul tastei apăsate. Această operație se realizează prin apelul funcției getch. De ezemplu, într-o instrucțiune de forma *car = getch( );* variabila de tipul caracter *car* primeşte codul tastei apăsate fără ca pe ecran să apară caracterul asociat acesteia. Dacă totuşi este necesar a se citi un caracter cu ecou, atunci se foloseşte funcția getche, similară cu funcția getch, cu deosebirea că afişează pe ecran caracterul asociat tastei apăsate.