IV. Limbajul C

1. Construcții de bază
2. Tipuri de date
3. Expresii
4. Instrucțiuni
5. Operaţii de intrare/ieşire cu tastatura/monitorul
6. Tipuri dinamice de date
7. Subprograme

IV.5. Operații de intrare/ieșire

1. Descriptori de format
2. Funcții de citire/scriere cu fomat
3. Funcții de intrare/ieșire fără format

1. Descriptori de format

Forma generală a șirului descriptor: *%[cadraj][lăţime[.precizie]]cod_conversie*

Pentru scrierea datelor pe ecran:

- *Cadraj*:
  - – (minus) determină alinierea datelor la stânga
  - + (plus) determină alinierea la dreapta
- *Lăţime*: lăţimea câmpului (în număr de caractere) în care se va scrie valoarea
- *Precizie*: indică precizia cu care se va scrie data
  - pentru numere reale, indică numărul de zecimale afişate
  - pentru un şir de caractere, indică numărul maxim de caractere care se vor afişa
- *Cod_conversie*: este format din unul sau două caractere

Pentru citirea datelor de la tastatură (parametri opţionali):

- *lăţime*: lăţimea maximă a câmpului din care se va citi valoarea respectivă;
- *un caracter asterisc* (se va citi de la tastatură valoarea respectivă, dar nu va fi atribuită nici unei variabile).

Semnificaţia codurilor de conversie

Cod conversie	Semnificaţie	Observaţii
d	zecimal cu semn (int)	întregi
i	zecimal cu semn (int)	întregi
o	octal fără semn	întregi
u	zecimal fără semn	întregi
x	printf: hexazecimal fără semn (int) litere mici scanf: hexazecimal (int)	întregi
X	printf: hexazecimal fără semn (int) litere mari scanf: hexazecimal (int)	întregi
f	virgulă mobilă [-]dddd.ddd (format matematic)	virgulă mobilă
e	virgulă mobilă [-]d.ddd.e[+/-]dd (format ştiinţific)	virgulă mobilă
g	ca şi f sau e, depinde de precizie	virgulă mobilă
E	ca şi e, utilizează litera E pentru exponent	virgulă mobilă
G	ca şi g, utilizează litera E pentru exponent	virgulă mobilă
c	imprimă un singur caracter	întregi
s	imprimă caractere până la '\0' (NULL) sau până la .prec	şiruri
l sau L	poate să preceadă descriptorii d, o, x sau u, caz în care se face conversia din int spre long; dacă precede descriptorul f, se face conversie din float spre double	întregi
%	caracterul %	%% => %

2. Funcţii de scriere/citire cu format

Funcţii definite în biblioteca standard *stdio.h*:

- printf(<sir_descriptor>,<expresie1>,<expresie2> ...);
- scanf(<sir_descriptor >,<adresa1>,<adresa2> ...);

- **Funcţia *printf***
  - execută scrierea datelor pe unitatea standard de ieşire
  - returnează numărul de octeţi (câmpuri) afişate în caz de succes sau -1 în caz de eroare
  - *parametrul* <*sir_descriptor*> conţine caractere care se afişează şi descriptori de format

- **Funcţia *scanf***
  - execută citirea datelor de la unitatea standard de intrare
  - returnează numărul de câmpuri corect procesate
  - parametrul <*sir_descriptor*> poate conţine (în afara codurilor de conversie), caractere care vor trebui introduse ca atare de la tastatură

Exemple - printf

1. `printf("abcde");`
   `printf("%s","abcde");`

Ambele apeluri au acelaşi efect şi anume afişarea şirului de caractere *abcde*.

2. `printf("#%-4c#", 'A');`

Apelul are ca efect afişarea şirului #A # .

3. `printf("#%10s#","abcde");`
   `printf("#%15.10s#","Limbajul C++");`

Apelurile au ca efect afişarea pe ecran a şirurilor:

abcde#

Limbajul C#

4. `printf("a=%5.2f", 123.456);`
   `printf("a=%5.2f", 12.341);`

Apelurile au ca efect afişarea pe ecran:

a=123.46

a= 12.34

Exemple - scanf

1. Citirea a două numere întregi

```
int x,y;
scanf("%i %i", &x,&y); /*separate prin spaţiu */
scanf("%i,%i", &x,&y); /* separate prin virgulă */
```

2. Citirea unui caracter

```
char a;
scanf("%c",&a);
```

3. Citirea unui caracter dintr-un șir de caractere

```
char den[20];
printf("caracter: ");
scanf("%c", &den[0]);/* den[0] nu este pointer*/
```

4. Citirea unui șir de caractere

```
char den[20];
printf("nume: ");
scanf("%s", den);   /* den este pointer */
```

3. Funcţii de intrare/ieşire fără format

Funcţii definite în biblioteca standard *stdio.h*:

- Funcţia *gets*
  - gets(masiv_de_caractere);
  - introducerea de la tastatură a unui şir de caractere
  - se returnează adresa unde a fost depus acest şir
  - şirul citit poate conţine şi spaţii
- Funcţia *puts*
  - puts(sir);
  - afişarea unui şir de caractere pe ecran,
  - se returnează codul ASCII al ultimului caracter afişat în caz de succes, sau –1 în caz de eroare

Exemple:

1. char sir[80];

printf("Introduceti un text:");

gets(sir);

printf("Sirul introdus este: %s\n", sir);

2. char sir[] = "C este un limbaj de programare\n";

puts(sir);

Funcţii de intrare/ieşire specifice caracterelor

Funcţii definite în biblioteca standard *conio.h*:

- Funcţiile *getch şi getche*
  - getch(); getche();
  - returnează codul ASCII al caracterului
  - *getch* se realizează fără ecou pe ecran (caracterul corespunzător tastei apăsate nu este afişat pe ecran)
  - *getche*, pe ecran apare caracterul corespunzător tastei apăsate
- Funcţia *putch*
  - are ca parametru o valoare întreagă
  - afişează caracterul ASCII corespunzător

Macro-uri definite în biblioteca standard *stdio.h*

- *Macro-ul getchar*
  - permite citirea cu ecou pe ecran a caracterelor de la tastatură
  - se pot citi numai caractere asociate codurilor ASCII direct afişabile
- *Macro-ul putchar*
  - are ca parametru o expresie al cărei rezultat reprezintă codul ASCII al caracterului care se doreşte să fie afişat
  - se returnează codul ASCII al caracterului afişat sau -1 în caz de eroare

Exemple

1. Citirea unui caracter fără ecou

```
unsigned char c;
c=getch();
printf("%d",c);
```

2. Afișarea unui caracter

```
int i,j;
putch(i*8+j);
```

3. Pe ecran se va afişa primul caracter introdus de la tastatură

```
putchar(getchar());
putchar('\n');
```