Laborator 4 – Operaţii de intrare-ieşire standard 2

Funcţia gets( )

Cu ajutorul acestei funcţii se citeşte cu ecou de la terminal o succesiune de caractere ASCII care este terminată prin apăsarea tastei <ENTER>. Funcţia plasează aceste caractere la adresa memorată de argument. Caracterul ENTER nu face parte din şir ci la sfârşitul acestuia este plasat un caracter de terminare NUL şi funcţia revine. Greşelile de tastare pot fi corectate prin apăsarea tastei <BACKSPACE> înainte de a apăsa tasta <ENTER>. Având în vedere că numele unui tablou are ca valoare adresa de început a zonei de memorie alocate, înseamnă că numele unui tablou poate fi folosit ca parametru în apelul funcţiei gets().
Prototipul funcţiei gets( ) se află în fişierul standard <stdio.h> (figura 1).

SINTAXĂ	#include <stdio.h> char *gets(*str);

Fig.1

str reprezintă un tablou care primeşte caracterele introduse de utilizator.
Caracterele citite se păstrează în str.

Funcţia puts( )

Cu ajutorul acestei funcţii se afişează la terminal un şir de caractere ASCII. După afişarea şirului, cursorul trece automat în coloana întâi din linia următoare. Prototipul funcţiei puts( ) se află în fişierul standard <stdio.h> (figura 2).

SINTAXĂ	include <stdio.h> int puts(const char *str);

Fig. 2.

str reprezintă numele tabloului unidimensional de tip char în care se află şirul de caractere de afişat.
Funcţia puts() returnează o valoare non-negativă, care este codul ultimului caracter al şirului de caractere afişat sau -1 în caz de eroare.

Observaţie:
Apelul funcţiei puts( ) necesită mai puţine resurse decât apelul funcţiei printf( ), deoarece puts( ) poate genera numai un şir de caractere şi nu poate genera numere sau conversii de format. De aceea funcţia puts( ) ocupă mai mult spaţiu şi rulează mai rapid decât printf(). Când sunt necesare coduri de mare eficienţă care nu necesită conversii se utilizează funcţia puts( ).

Exemple.
1.Să se citească un şir de caractere din tabloul *sir* şi să se afişeze şirul şi lungimea acestuia. Pentru operaţiile de intrare/ieşire se vor utiliza funcţiile gets( ) şi puts( ).

Pentru a calcula lungimea şirului introdus se va utiliza funcţia strlen( ) din biblioteca standard a limbajului C. Aceasta are prototipul în fişierul antet <string.h>.

```
/*P05_04.C*/ /*Citirea unui sir si afisarea sirului si a lungimii acestuia*/
#include <stdio.h>
#include<conio.h>
#include <string.h>
void main(void)
{
char sir[50];
printf("\nIntroduceti sirul:");
gets(sir);
printf("\nSirul este:");
puts(sir);
printf("Lungimea sirului este:%d\n",strlen(sir));
printf("Actionati orice tasta pentru continuare\n"),
getch();
}
```

2. Să se scrie un program care citeşte de la intrarea standard numele şi prenumele unei persoane şi apoi afişează iniţialele acesteia pe un rând, fiecare iniţială fiind urmată de un punct.

```
/*P05_05.C*/ /*Utilizarea functiilor gets() si puts()*/
#include<stdio.h>
#include<conio.h>
void main(void)
{
char nume[200]; char prenume[200];
puts("Introduceti numele persoanei:");
gets(nume);
puts("Introduceti prenumele persoanei:");
gets(prenume);
puts(Initialele persoanei sunt:”);
putch(nume[0]);
putch('.');
putch(prenume[0]);
putch('.');
puts("\nApasati orice tasta pentru continuare");
getch();
}
```

Funcţia getchar( )

Funcţia getchar( ) citeşte un caracter ASCII cu ecou de la terminal. Ea memorează într-o zonă tampon caracterele tastate până la apăsarea tastei <ENTER>. Această zonă se numeşte intrare cu rând tampon (line-buffered input). La acţionarea tastei <ENTER> se introduce caracterul

de rând nou şi se continuă execuţia lui getchar( ).Funcţia returnează codul ASCII al caracterului curent din zona tampon. La fiecare nou apel al funcţiei se va returna codul ASCII al caracterului următor din zona tampon. La întâlnirea sfârşitului de fişier funcţia getchar() va returna valoarea constantei simbolice EOF care este –1. Prototipul acestei funcţii se găseşte în <stdio.h> (figura 3).

SINTAXĂ	include <stdio.h> int getchar(void);

Fig. 3.

Funcţia putchar( )

Această funcţie scrie un caracter pe ecran la poziţia curentă a cursorului. Prototipul acestei funcţii se găseşte în <conio.h> (figura 4)

SINTAXĂ	include <conio.h> int putchar(int c);

Fig. 4.

c reprezintă codul ASCII al caracterului care se va afişa. Funcţia returnează codul ASCII al caracterului afişat sau EOF (-1) dacă survine o eroare.
Exemplu.
Să se citească de la tastatură litere şi să se afişeze cu majuscule dacă sunt scrise normal şi invers. Programul se va realiza utilizând funcţiile getchar() şi putchar().

```
/*P05_06.C*/ /*Utilizarea functiilor getchar() si putchar()*/
#include <stdio.h>
#include <conio.h>

void main (void)
{
int litera;
printf(" \nIntroduceti litera:");
litera=getchar();
if (litera<=122 &&litera>=97)
  putchar(litera-'a'+'A');
else
  if( litera<=90 &&litera>=65)
    putchar(litera+'a'-'A');
  else
    printf("\nCaracterul introdus nu e litera");
printf("\nApasati orice tasta pentru continuare");
```

```
getch();
}
```

Funcţia printf( )

Această funcţie realizează tipărirea pe ecran (fişierul stdout) sub controlul formatelor.
Prototipul acestei funcţii se află în fişierul standard <stdio.h> (figura 5).

SINTAXĂ	include <stdio.h> int printf(const char *sir_control, par1,par2,…parn);

Fig. 5.

sir_control se compune din două tipuri de articole: primul articol se compune din caracterele care se vor afişa pe ecran; al doilea articol conţine specificatorii de format care indică modul cum vor fi tipărite argumentele par1, par2, …parn care urmează. Specificatorul de format este compus din caracterul % urmat de codul formatului. Aceşti specificatori de format definesc conversiile datelor din format intern în format extern. Corespondenţa dintre specificatorii de format şi argumente se face în ordine de la stânga la dreapta iar numărul de argumente este obligatoriu să fie identic cu numărul specificatorilor de format.
Observaţie:
Cele două articole ale sir_control nu sunt întotdeauna ambele prezente.
Funcţia printf( ) returnează numărul de caractere scrise sau –1 în caz de eroare.

În tabelul 1. sunt prezentaţi specificatorii de format acceptaţi de funcţia printf( ) şi semnificaţia pe care o au aceştia.

Tab. 1

Cod	Format	Semnificaţie
%c	Caracter	Se afişează un caracter. Valoarea argumentului corespunzător este codul ASCII al caracterului care se afişează.
%d	Întreg zecimal cu semn	Datele de tip int se convertesc şi se afişează în zecimal.
%i	Întreg zecimal cu semn	Datele de tip int se convertesc şi se afişează în zecimal.
%e	Notaţie ştiinţifică (e mic)	Datele flotante de tip float sau double se convertesc şi se afişează în forma **notaţie ştiinţifică**: parte întreagă.cifre zecimale exponent.
%E	Notaţie ştiinţifică (E mare)	Datele flotante de tip float sau double se convertesc şi se afişează în forma **notaţie ştiinţifică**: parte întreagă.cifre zecimale exponent.
%f	Virgulă mobilă	Datele flotante de tip float sau double se convertesc spre o **reprezentare zecimală**: parte întreagă.cifre zecimale.
%g	Foloseşte %e sau %f,	Datele de tip float sau double se convertesc şi se afişează

	cel mai scurt	fie ca în cazul specificatorului %f fie ca în cazul lui %e, astfel încât rezultatul să fie afişat pe un număr minim de caractere.
%G	Foloseşte %E sau %f, cel mai scurt	Datele de tip float sau double se convertesc şi se afişează fie ca în cazul specificatorului %f fie ca în cazul lui %E, astfel încât rezultatul să fie afişat pe un număr minim de caractere.
%o	Octal fără semn	Datele de tip int sau unsigned se convertesc şi se afişează în octal (baza de numeraţie 8).
%s	Şir de caractere	Se afişează un şir de caractere pe ecran până la întâlnirea caracterului nul. Argumentul corespunzător este adresa de început a zonei de memorie care conţine şirul.
%u	Întreg zecimal fără semn	Datele binare de tip unsigned se convertesc şi se afişează în zecimal.
%x	Hexazecimal fără semn (litere mici)	Datele de tip int sau unsigned se convertesc şi se afişează în hexazecimal (baza de numeraţie 16).
%X	Hexazecimal fără semn (litere mari)	Datele de tip int sau unsigned se convertesc şi se afişează în hexazecimal.
%%	Afişează simbolul procent	Dacă după caracterul % urmează un alt caracter decât codurile de format, % se ignoră şi se afişează caracterul care urmează după el.

Alinierea datelor la ieşire
Datele se încadrează implicit la dreapta câmpului în care ele se scriu. Dacă după caracterul % apare semnul "-" atunci data este încadrată la stânga.

Modificatori de format
Modificatorii de format au rolul de a modifica uşor semnificaţia specificatorilor de format. Aceştia sunt:
- specificatorul lăţimii minime de format;
- specificatorul de precizie.
- modificatorii de tip long şi short.

Specificatorul lăţimii minime de format (minimum field width specifier) reprezintă un şir de cifre zecimale opţional plasat între semnul % şi codul formatului şi defineşte dimensiunea minimă a câmpului în care se afişează datele corespunzătoare formatului indicat.
Întâlnim următoarele situaţii:
a. Dacă şirul sau numărul este mai lung decât dimensiunea precizată, va fi afişat integral;
b. Dacă şirul sau numărul este mai scurt decât dimensiunea precizată, se va scrie în câmpul respectiv iar restul câmpului se completează cu spaţii.

Specificatorul de precizie (precision specifier) se scrie imediat după specificatorul lăţimii minime de format (dacă acesta există). Este format dintr-un punct urmat de un întreg. Semnificaţia sa depinde de tipul de date căruia i se aplică. Dacă data este în virgulă mobilă care foloseşte unul din specificatorii %f, %e sau %E, atunci specificatorul de precizie reprezintă numărul de zecimale care se scriu. Dacă specificatorul de precizie se aplică formatelor %g sau %G, acesta reprezintă numărul de caractere semnificative. Dacă se aplică

la şiruri, specificatorul de precizie indică numărul maxim de caractere care se scriu. Dacă şirul este mai lung decât dimensiunea maximă, caracterele vor fi trunchiate. Dacă se aplică întregilor, specificatorul de precizie indică numărul minim de cifre care apare pentru fiecare număr. Pentru a realiza numărul de cifre indicat, la începutul numărului se adaugă zerouri.

Modificatorii de tip long şi short

Tab 2.

Tipul de modificator	Specificatorii la care poate fi aplicat	Tipul de dată obţinută
long (l)	d, i, o, u, x, X e, E, f, g, G	long int double
short (h)	d, i, o, u, x, X	short int
long (L)	e, E, f, g, G	long double

Exemple.
1.Să se citească un caracter imprimabil şi să se afişeze:
a.precedat şi urmat de apostrof într-un câmp de 3 caractere, aliniat la dreapta;
b.precedat şi urmat de apostrof într-un câmp de 3 caractere, aliniat la stânga.

```
/*P05_07.C*/ /*Afisarea de caractere imprimabile*/
#include <stdio.h>
#include <conio.h>
void main (void)
{
int caract1;
printf(" \nIntroduceti caracterul:");
caract1=getche();
printf(" \nAfisare caracter aliniat la dreapta: '%3c'\n",caract1);
printf("Afisare caracter aliniat la stanga: '%-3c'\n",caract1);
printf("Apasati orice tasta pentru continuare");
getch();
}
```

2.Să se afişeze textul “Programarea in limbajul C” folosind specificatori de format.

```
/*P05_08.C*/
/*Afisarea sirurilor de caractere*/
#include <stdio.h>
#include <conio.h>
void main (void)
{ char sir[]="Programarea in limbajul C";
printf("\n!%s!",sir);
printf("\n!%30s!",sir);
printf("\n!%-30s!",sir);
printf("\n!%15s!",sir);
printf("\n!%15.10s!",sir);
printf("\n!%-15.10s!",sir);
printf("\n!%10.21s!",sir);
printf("\nApasati orice tasta pentru continuare");
getch();
```

```
}
```

3.Să se afişeze constanta 18648 cu diferiţi specificatori de format: %d, %8d, %-8d, %08d, %o, %x, %X.

```
/*P05_09.C*/
/*Afisarea unei constante cu diferiti specificatori de format*/
#include <stdio.h>
#include <conio.h>
#define C 18648
void main (void)
{
clrscr();
printf("\nAfisare in zecimal:!%d!",C);
printf("\nAfisare in zecimal:!%8d!",C);
printf("\nAfisare in zecimal:!%-8d!",C);
printf("\nAfisare in zecimal:!%08d!",C);
printf("\nAfisare in octal:!%o!",C);
printf("\nAfisare in hexazecimal:!%x!",C);
printf("\nAfisare in HEXAZECIMAL:!%X!",C);
printf("\nApasati orice tasta pentru continuare");
getch();
}
```

Funcţia scanf( )

Funcţia are rolul de a citi toate tipurile de date încorporate şi de a converti în mod automat numerele din format extern în formatul intern corespunzător. Ea este oarecum asemănătoare cu inversul funcţiei printf( ). Prototipul funcţiei se găseşte în <stdio.h> (figura 6).
Funcţia returnează numărul articolelor de date cărora li s-a atribuit o valoare. În caz de eroare se returnează –1.

SINTAXĂ	include <stdio.h> int scanf(const char *sir_control, par1,par2,…parn);

Fig. 6.

sir_control defineşte formatele datelor şi a eventualelor texte aflate la intrarea de la tastatură.
par1, par2, …parn reprezintă adresele zonelor în care se păstrează datele citite după ce au fost convertite în formatul intern. Adresa unei zone de memorie se exprimă utilizând operatorul unar adresă &.
sir_control constă din 3 categorii de caractere: specificatori de format; caractere spaţiu; caractere non-spaţiu.

Specificatorul de format este compus din caracterul % urmat de codul formatului. Aceşti specificatori de format definesc conversiile datelor din format extern în format intern. Specificatorii de format acceptaţi de funcţia scanf sunt prezentaţi în tabelul 3.

Tab. 3.

Cod	Format	Semnificaţie
%c	Caracter	Se citeşte caracterul curent din zona tampon corespunzătoare datelor introduse de la tastatură
%d	Întreg zecimal	Se citeşte un întreg în baza 10 şi se converteşte spre tipul int.
%i	Întreg zecimal	Se citeşte un întreg în baza 10 şi se converteşte spre tipul int.
%e	Număr în virgulă mobilă	Se citeşte un număr în virgulă mobilă şi se converteşte spre tipul float.
%f	Număr în virgulă mobilă	Se citeşte un număr în virgulă mobilă şi se converteşte spre tipul float.
%g	Număr în virgulă mobilă	Se citeşte un număr în virgulă mobilă şi se converteşte spre tipul float.
%o	Număr în octal	Se citeşte un număr în octal (baza de numeraţie 8).
%s	Şir de caractere	Se citeşte un şir de caractere. Caracterele citite se pot păstra într-un tablou de caractere. Argumentul corespunzător specificatorului de format este numele acestui tablou. După ultimul caracter citit se păstrează în mod automat caracterul nul.
%u	Întreg zecimal fără semn	Citeşte un întreg fără semn şi se converteşte spre tipul unsigned.
%x	Hexazecimal	Se citeşte un număr în hexazecimal (baza de numeraţie 16).

Caracterele spaţiu (albe) din şirul de control sunt neglijate. Un caracter spaţiu alb determină funcţia scanf( ) să citească fără a stoca orice număr (inclusiv zero) de caractere spaţiu alb, până la întâlnirea unui caracter de alt tip.

Caracterele non-spaţiu din şirul de control sunt de asemenea neglijate. Un caracter non-spaţiu determină scanf( ) să citească şi să omită caracterele corespunzătoare din fluxul de intrare.

Modificatori de format

Ca şi în cazul funcţiei printf( ), funcţia scanf( ) permite aplicarea unor modificatori asupra unor specificatori de format. Aceştia sunt: modificatorul de lungime maximă de câmp; modificatorii de tip long şi short.

Modificatorul de lungime maximă de câmp este un întreg plasat între % şi specificatorul de format şi limitează numărul de caractere care pot fi citite pentru acel câmp.

Modificatorii de tip long şi short – vezi printf

Ignorarea datelor de intrare

Ignorarea anumitor date de intrare este utilă când se doreşte prelucrarea doar a unei părţi din datele de intrare. Funcţia scanf( ) poate citi un câmp, dar să nu-l atribuie nici unei variabile folosind înaintea codului de format caracterul "*". Când este necesară prelucrarea doar a unei părţi din datele de intrare se foloseşte această ignorare a atribuirii.
Exemplu.
scanf("%d%*c%d",&x,&y);
Se introduce perechea de coordonate 15, 15. Virgula va fi corect citită, dar nu va fi atribuită nici unei variabile.

Exemple.
1.Să se citească un şir de caractere din tabloul sir şi să se afişeze şirul şi lungimea acestuia. Pentru operaţiile de intrare/ieşire se vor utiliza funcţiile scanf( ) şi printf( ).

```
/*P05_13.C*/
/* citirea unui sir si afisarea sirului si a lungimii acestuia*/
#include <stdio.h>
#include<conio.h>
#include <string.h>
void main(void)
{
char sir[50];
printf("Introduceti sirul:");
scanf("%s", sir);
printf("\nSirul este:%s", sir);
printf("\nLungimea sirului este:%d\n",strlen(sir));
printf("Actionati orice tasta pentru continuare\n"), getch();
}
```

Astfel acest program prezintă deosebiri faţă de exemplul de program din cadrul funcţiilor gets( ) și puts( ), în care cu ajutorul funcţiei gets( ) s-a introdus întreg şirul şi s-a calculat lungimea acestuia.

2. Să se citească un număr întreg zecimal, unul octal şi unul hexazecimal, să se afişeze corespunzător cele trei numere şi apoi să se afişeze numărul zecimal în octal şi hexazecimal.

```
/*P05_14.C*/
/*Citeste si afiseaza numere intregi in diferite baze de numeratie*/
#include<stdio.h>
#include<conio.h>
void main(void)
{
int zecim, octal, hexazec;
printf("Introduceti numarul zecimal:");
scanf("%d",&zecim);
printf("Introduceti numarul in baza 8:");
scanf("%o",&octal);
printf("Introduceti numarul in baza 16:");
scanf("%x",&hexazec);
printf("nr. zecimal=%d, nr. octal=%o, nr. hexazecimal=%x\n", zecim, octal, hexazec);
printf("Numarul zecimal %d este %o in octal \n",zecim, zecim);
printf("Numarul zecimal %d este %x in hexazecimal \n",zecim,zecim);
```

```
printf("Numarul zecimal %d este %X in HEXAZECIMAL \n",zecim,zecim);
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

3. Să se citească în mod conversaţional elementele unui şir având elemente numere reale şi apoi să se tipărească şirul.

```
/*P05_15.C*/ /*Citirea si tiparirea elementelor unui sir*/
#include<stdio.h>
#include<conio.h>
#define MAX 50
void main(void)
{
int n,i;
double a[MAX];
printf ("Introduceti dimensiunea sirului n<50:");
scanf("%d",&n);
while(n<0||n>MAX)
{
  printf("Dimensiune eronata>%d\n");
  printf("Introduceti alt n:");
  scanf("%d",&n);
}
/*se citesc  elementele sirului*/
for (i=0;i<n;i++){
printf("Introduceti elem a(%d) al sirului:",i);
scanf("%lf",&a[i]);
                    }

/*se tiparesc  elementele sirului*/
printf("Elementele sirului sunt:\n");
for(i=0;i<n;i++)
        printf("a(%d)=%lf\n",i,a[i]);

printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

Probleme propuse:

1. Calculaţi media aritmetică a două numere întregi.
2. Să se citească două numere reale de la tastatură. Să se calculeze suma, diferenţa, produsul şi valoarea exactă a împărţirii lor. Atenţie la cazurile de nedeterminare.
   - Să se afişeze suma aliniată la stânga pe 30 de câmpuri.
   - Sa se afişeze diferenţa aliniată la dreapta pe 10 câmpuri cu 2 zecimale.
   - Sa se afişeze: Produsul numerelor * si * este *. (*- reprezintă numerele si produsul) Produsul se afişează în format ştiinţific.
   - Sa se afişeze rezultatul împărţirii cu 4 zecimale.