Laborator 8 – Instrucţiuni de bază 2
(Instrucţiuni de ciclare)

Instrucţiunile de ciclare se împart în:
a. instrucţiuni care realizează o structură repetitivă **condiţionată anterior**:
while, for.
b. instrucţiuni care realizează o structură repetitivă **condiţionată posterior**:
do-while.

Instrucţiunea while

Această instrucţiune realizează o structură ciclică condiţionată anterior .

while(conditie)
instrucţiune;

conditie poate fi orice expresie care dacă este adevărată este diferită de zero. **instrucţiune** poate fi o singură instrucţiune sau o instrucţiune compusă.

Instrucţiunea while se execută în felul următor:
1. Se evaluează valoarea conditie din paranteză;
2. Dacă conditie are valoare de adevăr adică diferită de zero, atunci se execută instrucţiune şi se reia pasul 1, altfel se trece la instrucţiunea imediat următoare instrucţiunii while.

In cadrul instrucţiunii pot să existe alte instrucţiuni while. În acest caz instrucţiunile se numesc instrucţiuni while imbricate.

Instrucţiune din corpul lui while poate să realizeze terminarea instrucţiunii while fără a se mai ajunge la evaluarea expresiei logice.

Observaţie:
1. Dacă conditie are valoarea fals de la început, atunci instrucţiune nu se execută niciodată. Această facilitate oferită de instrucţiunea while poate elimina necesitatea efectuării unui test condiţional separat înainte de ciclu.
2. In interiorul ciclului while nu este necesară prezenţa instrucţiunilor.

Instrucţiunea for

Această instrucţiune este o instrucţiune foarte flexibilă care se utilizează de asemenea pentru realizarea unei structuri repetitive condiţionate anterior .
for(initializare;conditie;actualizare)
instrucţiune;

Antetul ciclului este definit de rândul:
for(initializare,conditie,actualizare)
initializare, conditie, actualizare sunt expresii.
instrucţiune care se execută în mod repetat formează corpul ciclului.

initializare – reprezintă partea de iniţializare a ciclului, reprezentată prin una sau mai multe expresii de atribuire;

condiţie – reprezintă o expresie relaţională care determină când se încheie ciclul;

actualizare – reprezintă partea de reiniţializare a ciclului, reprezentată printr-o listă de expresii de atribuire. Cu ajutorul acestora, unele din datele prelucrate ciclic sunt actualizate (de regulă cele ce intervin în expresia testată).

Instrucţiunea for se execută conform următorilor paşi:

1. Se execută secvenţa de iniţializare definită de expresia initializare.
2. Se evaluează conditie. Dacă conditie are valoarea zero, atunci se iese din ciclu, adică se trece la instrucţiunea imediat următoare instrucţiunii for. Altfel se execută instrucţiune din corpul ciclului.
3. După executarea corpului ciclului, se execută expresia de reiniţializare definită de actualizare. Apoi se reia secvenţa de la pasul 2.

Pentru a mări flexibilitatea şi domeniul de aplicabilitate al ciclului for, limbajul C permite existenţa a două sau mai multe variabile de control ale ciclului. Se utilizează în acest sens operatorul virgulă între expresiile ciclului for.

Observaţii:

1. Instrucţiune din corpul ciclului for poate să nu se execute niciodată dacă conditie are valoarea zero încă de la început.
2. Expresiile din antetul funcţiei for pot fi şi vide. Totuşi caracterele punct şi virgulă vor fi întotdeauna prezente.
3. Se recomandă ca instrucţiunea de ciclare for să se folosească cu prioritate în ciclurile care au pas. Aceasta nu înseamnă că nu putem folosi instrucţiunea for şi în alte cazuri.
4. Instrucţiunea for de forma următoare este de asemenea validă:

```
for(; ;)
        instrucţiune;
```

Această instrucţiune defineşte un ciclu infinit din care se iese prin alte mijloace decât cele obişnuite, cum ar fi de exemplu revenirea dintr-o funcţie sau un salt la un anumit punct al programului etc.

5. Conţinutul ciclului for poate fi de asemenea vid. Acest lucru poate fi utilizat pentru a îmbunătăţi eficienţa unor algoritmi şi pentru a crea cicluri fără întârziere.

Instrucţiunile de ciclare for şi while care urmează au acelaşi efect:

```
for(exp1; exp2; exp3)
        instrucţiune

exp1;
while(exp2){
        instrucţiune
        exp3;
            }
```

Reciproc orice instrucţiune while poate fi scrisă folosind o instrucţiune for cu exp1 şi exp3 vide.

Astfel instrucţiune while de mai jos:

```
while(exp)
        instrucţiune
```

este echivalentă cu următoarea instrucţiune for:

```
for(; exp ;)
        instrucţiune
```

Instrucţiunea do-while

Această instrucţiune realizează structura repetitivă condiţionată posterior. Ea poate fi realizată cu ajutorul celorlalte instrucţiuni definite de ciclare. Cu toate acestea, prezenţa ei în limbaj măreşte flexibilitatea în programare.

Ea are următorul format:

```
do
        instructiune
while(conditie);
```

Instrucţiunea do-while se execută în felul următor:

1. Se execută **instrucţiune**
2. Se evaluează **conditie** din paranteza rotundă.
3. Dacă valoarea conditie este nulă, se trece la execuţia instrucţiunii aflată după instrucţiunea do-while. Altfel se revine şi se execută întâi instrucţiune şi apoi se testează condiţia de repetare a execuţiei ei.

Observaţie:
În cazul instrucţiunii do-while, corpul ciclului se execută cel puţin odată, spre deosebire de instrucţiunile for şi while, unde este posibil să nu se execute niciodată.

Instrucţiunea do-while se poate echivala prin secvenţa while de mai jos:

```
instrucţiune
while(conditie)
        instrucţiune
```

Exemple

1. Să se scrie un program care să afişeze cele 26 litere mari ale alfabetului latin împreună cu codurile lor ASCII.

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
char caract1,caract2 ;
caract1='A';
caract2=caract1+13;
printf("CARACTERELE ALFABETULUI LATIN SI CODURILE LOR ASCII\n");
while(caract2<='Z')
{
```

```
printf("\t %c: %d\t\t%c:  %d\n",caract1,caract1,caract2,caract2);
caract1++; caract2++;
}
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

2. Să se scrie un program care citeşte un număr întreg din fişierul de intrare, apoi calculează şi tipăreşte “n!”.

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
int n,i;
double fact;
printf("Introduceti valoarea lui n:");
scanf("%d",&n);
i=1;
fact=1;
while(i<=n)
        {
        fact*=i;
        i++;
        }
printf("n=%d   n!=%g\n",n,fact);
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

3. Să se scrie un program care calculează și afișează valoarea polinomului $p(x) = 2x^2 - 7x + 1$ pentru următorul șir de valori: x=0.5, 1, 1.5, ..., 10.

```
#include <stdio.h>
void main(void)
{
float x;
x=0.5;
while(x<=10){
        printf("Valoarea polinomului in punctul %.1f este: %.1f\n",x, 2*x*x-7*x+1);
        x=x+0.5;
        }
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

Exerciţiu propus:

4. Să se scrie un program care calculează $C_n^k = \frac{n!}{(n-k)!\cdot k!}$.

5. Să se calculeze pătratul unui număr real introdus de la tastatura în mod repetat.

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
float x,y;
char caract;

for(;;){
  printf("\nIntroduceti un numar real x:");
  scanf("%f",&x);
  y=x*x;
  printf("(x)^2=%f\n",y);
  printf("\nDoriti reluarea calculelor? (Y/N)");
  caract=getche();
  if(caract!='y' && caract!='Y')
  break;
        }
printf("\nApasati orice tasta pentru continuare\n");
getch();
}
```

6. Să se scrie un program care citeşte un întreg pozitiv şi îl testează dacă este prim.

```
#include <stdio.h>
#include <conio.h>
main()
{
long i,n;
int j,t;
printf("\nIntroduceti intregul pozitiv n=");
while(scanf("%ld",&n)!=1||n<=0){
          printf("n nu este intreg pozitiv:\n");
          printf("Introduceti alt n:");
                                                }
for(j=1,i=2;i*i<=n&&j;i++){
   if(n%i==0)
   j=0;
                                          }
if(j==0)
   printf("Numarul %ld nu este prim\n",n);
else
   printf("Numarul %ld este prim\n",n);
getch();
}
```

7. Să se calculeze valoarea funcţiei f(x)=y, pe intervalul [a,b] parcurs cu pasul p, a căror valori sunt introduse de la tastatură.

$$f(x)=\begin{cases} x-5, & \text{daca } x<-7 \\ 12, & \text{daca} -7\le x<3 \\ \dfrac{x^2+x+3}{x-5}, & \text{daca } x\ge 3 \end{cases}$$

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
float x,y,a,b,p;
printf("Introduceti capetele intervalului:\n");
printf("a=");
scanf("%f",&a);
printf("b=");
scanf("%f",&b);
printf("Introduceti pasul p=");
scanf("%f",&p);
if(a>b){
        a=a+b;
        b=a-b;
        a=a-b;
}
for(i=1;i<=(b-a)/p+1;i++){
        x=a+(i-1)*p;
        if (x<-7) y=x-5;
        else
                if (x<3) y=12;
        else
                if (x==5) printf("In punctul x=%.2f functia nu se poate calcula\n",x);
        else y=(x*x+x+3)/float(x-5);
        if (x!=5) printf("Valoarea functiei f(%.2f) este y=%.2f\n",x,y);
}
getch();
}
```

8. Să se creeze un program care citește două numere reale de la tastatură și aplică unul din operatorii: adunare/scădere/înmulţire/împărțire asupra acestora funcţie de alegerea utilizatorului, de câte ori dorește acesta.

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
long float x,y;
char i, caract;
```

```
for(;;){
printf("Introduceti cele doua numere:\n");
printf("x="); scanf("%lf", &x);
printf("y="); scanf("%lf", &y);
printf("Pentru adunare introduceti +, pentru scadere introduceti -, pentru inmultire introduceti
*, pentru impartire introduceti /\n");
i=getche();
putch(10);
if (i=='+'|| i=='-'|| i=='*'|| i=='/')
        switch (i) {
        case '+':
                printf("Suma numerelor este: %lf\n",x+y);
                break;
        case '-':
                printf("Diferenta numerelor este: %lf\n",x-y);
                break;
        case '*':
                printf("Produsul numerelor este: %lf\n",x*y);
                break;
        case '/':
                if (y==0)
                        printf("Catul impartirii lui %lf la 0 nu se poate calcula", x);
                else
printf("Catul numerelor este: %lf\n",x/y);
                break;
        }
else
        printf("Nu ati introdus un operator corect\n");
printf("Doriti sa continuati? (apasati y sau Y daca da)\n");
caract=getch();
if(caract!='y' && caract!='Y')
break;
      }
}
```

Exerciţiu propus

9. Rezolvaţi problema 2 folosind un ciclu for.

10. Să se calculeze $\sqrt{a}$ cu cinci zecimale exacte, folosind următoarea relaţie iterativă: $x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n} \right)$, convergentă, iar $x_0 = 1$.

```
#include<stdio.h>
#include<math.h>
void main(void)
{
double a,x1,x2,y,eps=0.0000099;
```

```
printf("Introduceti a(>0)=");
scanf("%lf",&a);
if (a<0)
printf("Nu ati introdus un numar pozitiv");
else
{
x2=1;
do {
        x1=x2;
        x2=0.5*(x1+a/x1);
        //y=x2-x1;
} while (abs(x2-x1)>=eps);
printf("Radical din %lf este: %lf\n",a,x2);
}
getch();
}
```