MINISTERUL EDUCAȚIEI AL REPUBLICII MOLDOVA

UNIVERSITATEA TEHNICĂ A MOLDOVEI

**RAPORT**

LUCRAREA DE LABORATOR NR. 2

LA PROGRAMAREA CALCULATOARELOR

Elaborat: st. gr. TI – 211, Popa Ecaterina

Verificat: Stratulat Ștefan

UTM, Chișinău, 2021

**Scopul lucrării**

Scopul lucrării este de a familiariza studentul cu instrucțiunile de ramificare și ciclice a limbajului C.

**Probleme pentru instrucțiuni de ramificare:**

1. Scrieți un program C pentru a găsi maxim între două numere.

#include <stdio.h>

int main () {

int a, b;

printf ("Introduceti a = ");

scanf ("%d", &a);

printf ("Introduceti b = ");

scanf ("%d", &b);

if (a>b) {

printf ("Max = %d", a);

}

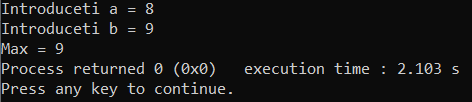
else {

printf ("Max = %d", b);

}

return 0;

}



1. Scrieți un program C pentru a găsi maxim între trei numere.

#include <stdio.h>

int main () {

int a, b, c;

printf ("Introduceti a = ");

scanf ("%d", &a);

printf ("Introduceti b = ");

scanf ("%d", &b);

printf ("Introduceti c = ");

scanf ("%d", &c);

if ((a>b) && (a>c)) {

printf ("Max = %d", a);

}

else if ((a>b) && (a>c)){

printf ("Max = %d", b);

}

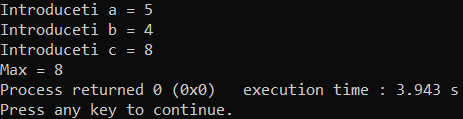
else {

printf ("Max = %d", c);

}

return 0;

}



1. Scrieți un program C pentru a verifica dacă un număr este negativ, pozitiv sau zero.

#include <stdio.h>

int main () {

int a, b, c;

printf ("Introduceti a = ");

scanf ("%d", &a);

if (a>0) {

printf ("a este pozitiv");

}

else if (a=0){

printf ("a este zero");

}

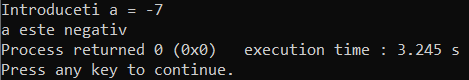
else {

printf ("a este negativ");

}

return 0;

}



1. Scrieți un program C pentru a verifica dacă un număr este divizibil cu 5 și 11 sau nu.

#include <stdio.h>

int main () {

int a, b, c;

printf ("Introduceti a = ");

scanf ("%d", &a);

if (a%5==0 && a%11==0) {

printf ("a este divizibil cu 5 si 11");

}

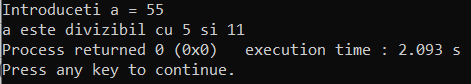
else {

printf ("a nu este divizibil cu 5 si 11");

}

return 0;

}



1. Scrieți un program C pentru a verifica dacă un număr este par sau impar.

#include <stdio.h>

int main () {

int m;

printf ("Introduceti m = ");

scanf ("%d", &m);

if (m%2==0) {

printf ("m este par");

}

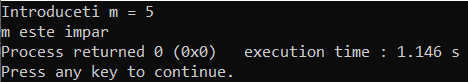
else {

printf ("m este impar");

}

return 0;

}



1. Scrieți un program C pentru a verifica dacă un an este bisect sau nu.

#include <stdio.h>

int main () {

int an;

printf ("Anul = ");

scanf ("%d", &an);

if (an%4==0) {

printf ("%d este an bisect", an);

}

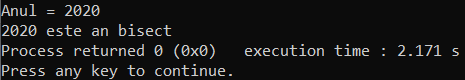
else {

printf ("%d nu este an bisect", an);

}

return 0;

}



1. Scrieți un program C pentru a verifica dacă un caracter este sau nu alfabet.

#include <stdio.h>

int main () {

char a;

printf ("Introduceti caracterul ");

scanf ("%c", &a);

if ((a>='a')&&(a<='z')||(a>='A')&&(a<='Z')) {

printf ("Caracterul este alfabet"); }

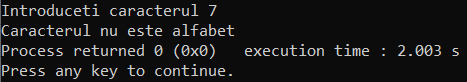
else {

printf ("Caracterul nu este alfabet");

}

return 0;

}



1. Scrieți un program C pentru a introduce orice alfabet și verificați dacă este vocală sau consoană.

#include <stdio.h>

int main () {

char a;

printf ("Introduceti caracterul ");

scanf ("%c", &a);

if ((a=='a')||(a=='e')||(a=='i')||(a=='o')||(a=='u')||(a=='A')||(a=='E')||(a=='I')||(a=='O')||(a=='U')) {

printf ("Caracterul este vocala"); }

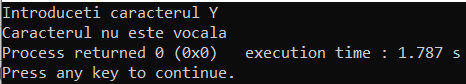
else {

printf ("Caracterul nu este vocala");

}

return 0;

}



1. Scrieți un program C pentru a introduce orice caracter și verificați dacă este alfabet, cifră sau caracter special.

#include <stdio.h>

int main () {

char a;

printf ("Introduceti caracterul ");

scanf ("%c", &a);

if ((a>='a')&&(a<='z')||(a>='A')&&(a<='Z')) {

printf ("Caracterul este alfabet");

}

else if ((a>='1')&&(a<='9')) {

printf ("Caracterul este cifra");

}

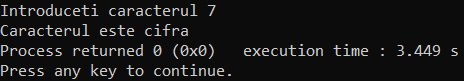
else {

printf ("Caracterul este caracter special");

}

return 0;

}



1. Scrieți un program C pentru a verifica dacă un caracter este cu majuscule sau minuscule.

#include <stdio.h>

int main () {

char a;

printf ("Introduceti caracterul ");

scanf ("%c", &a);

if ((a>='a')&&(a<='z')) {

printf ("Caracterul este minuscula");

}

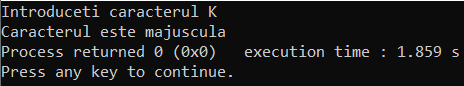
else {

printf ("Caracterul este majuscula");

}

return 0;

}



1. Scrieți un program C pentru a introduce numărul săptămânii și a imprima ziua săptămânii.

#include <stdio.h>

int main () {

int i;

printf ("Introduceti numarul zilei din saptamana: ");

scanf ("%d", &i);

if (i==1) {

printf ("Luni");

}

else if (i==2) {

printf ("Marti");

}

else if (i==3) {

printf ("Miercuri");

}

else if (i==4) {

printf ("Joi");

}

else if (i==5) {

printf ("Vineri");

}

else if (i==6) {

printf ("Sambata");

}

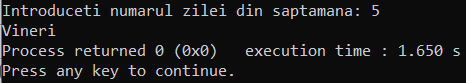
else {

printf ("Duminica");

}

return 0;

}



1. Scrieți un program C pentru a introduce numărul lunii și a imprima numărul de zile din luna respectivă.

#include <stdio.h>

int main () {

int luna;

printf ("Introduceti numarul lunii: ");

scanf ("%d", &luna);

if ((luna==1)||(luna==3)||(luna==5)||(luna==7)||(luna==8)||(luna==10)||(luna==12)) {

printf ("Luna are 31 de zile");

}

else if ((luna==4)||(luna==6)||(luna==9)||(luna==11)) {

printf ("Luna are 30 de zile");

}

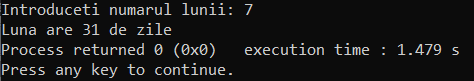
else {

printf ("Luna are 28 de zile");

}

return 0;

}



1. Scrieți un program C pentru a număra numărul total de cifre într-un număr.

#include <stdio.h>

int main () {

int n, nr;

printf ("Introduceti numarul: ");

scanf ("%d", &n);

while (n!=0) {

n=n/10;

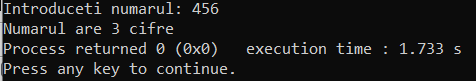
nr++;

}

printf ("Numarul are %d cifre", nr);

return 0;

}



1. Scrieți un program C pentru a introduce unghiurile unui triunghi și verificați dacă triunghiul este valid sau nu.

#include <stdio.h>

int main () {

double a, b, c;

printf ("Introduceti unghiul a: ");

scanf ("%lf", &a);

printf ("Introduceti unghiul b: ");

scanf ("%lf", &b);

printf ("Introduceti unghiul c: ");

scanf ("%lf", &c);

if (a+b+c==180) {

printf ("Triunghiul este valid");

}

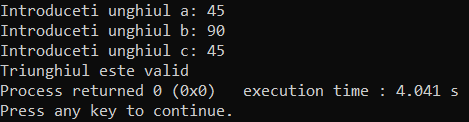
else {

printf ("Triunghiul nu este valid");

}

return 0;

}



1. Scrieți un program C pentru a introduce toate laturile unui triunghi și verificați dacă triunghiul este valid sau nu.

#include <stdio.h>

int main () {

int a, b, c;

printf ("Introduceti latura a: ");

scanf ("%d", &a);

printf ("Introduceti latura b: ");

scanf ("%d", &b);

printf ("Introduceti latura c: ");

scanf ("%d", &c);

if ((a+b>=c)||(a+c>=b)||(b+c>=a)) {

printf ("Triunghiul este valid");

}

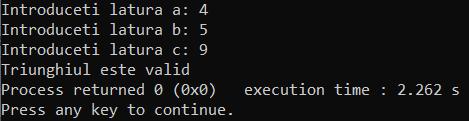
else {

printf ("Triunghiul nu este valid");

}

return 0;

}



1. Scrieți un program C pentru a verifica dacă triunghiul este echilateral, isoscel sau triunghi scalen.

#include <stdio.h>

int main () {

int a, b, c;

printf ("Introduceti latura a: ");

scanf ("%d", &a);

printf ("Introduceti latura b: ");

scanf ("%d", &b);

printf ("Introduceti latura c: ");

scanf ("%d", &c);

if ((a==b)&&(a==c)) {

printf ("Triunghiul este echilateral");

}

else if ((a==b)||(b==c)||(a==c)) {

printf ("Triunghiul este isoscel");

}

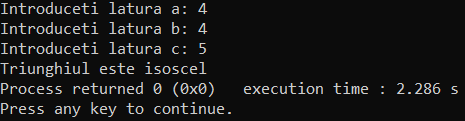
else {

printf ("Triunghiul este scalen");

}

return 0;

}



1. Scrieți un program C pentru a găsi toate rădăcinile unei ecuații pătratice.

#include <stdio.h>

#include <math.h>

int main () {

double a, b, c, delta, x, x1, x2;

printf ("Introduceti numarul a:");

scanf ("%lf", &a);

printf ("Introduceti numarul b:");

scanf ("%lf", &b);

printf ("Introduceti numarul c:");

scanf ("%lf", &c);

delta = pow(b,2)-4\*a\*c;

if (delta > 0) {

x1 = (- b - sqrt(delta))/(2 \* a);

x2 = (- b + sqrt(delta))/(2 \* a);

printf ("Solutiile sunt %.2f si %.2f", x1, x2);

} else if (delta == 0) {

x = (- b )/ (2 \* a);

printf ("Solutia este %.2f", x);

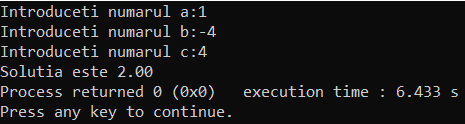
} else {

printf ("Ecuatia nu are solutii");

}

return 0;

}



**Probleme pentru instrucțiuni ciclice:**

1. Scrieți un program C pentru a imprima toate numerele naturale de la 1 la n. - folosind bucla while.

#include <stdio.h>

int main () {

int n, a=1;

printf ("n = ");

scanf ("%d", &n);

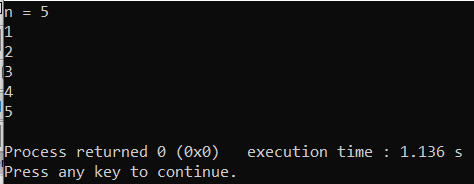
while (a<=n) {

printf ("%d \n", a++);

}

return 0;

}



1. Scrieți un program C pentru a imprima toate numerele naturale invers (de la n la 1). - folosind bucla while.

#include <stdio.h>

int main () {

int n;

printf ("n = ");

scanf ("%d", &n);

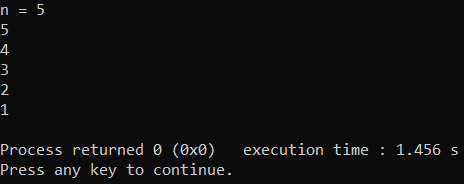
while (n>=1) {

printf ("%d \n", n--);

}

return 0;

}



1. Scrieți un program C pentru a imprima toate alfabetele de la a la z. - folosind bucla while

#include <stdio.h>

int main () {

int n;

printf ("n = ");

scanf ("%d", &n);

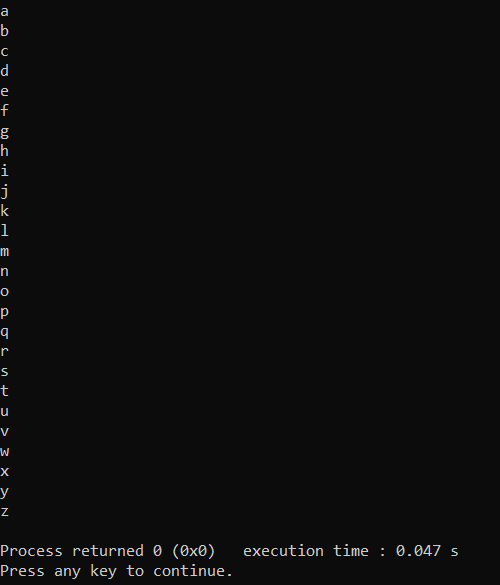
while (n>=1) {

printf ("%d \n", n--);

}

return 0;

}



1. Scrieți un program C pentru a imprima toate numerele pare între 1 și 100. - utilizând bucla while

#include <stdio.h>

int main () {

int i=1;

while (i<=100){

if(i%2==0) {

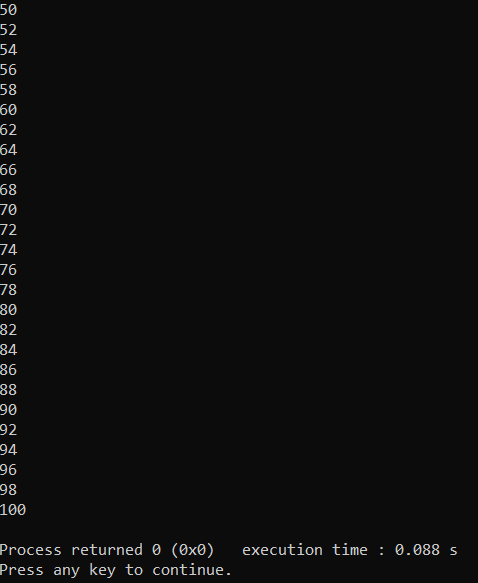
printf ("%d \n", i);

}

i++;}

return 0;

}



1. Scrieți un program C pentru a imprima toate numerele impare între 1 și 100.

#include <stdio.h>

int main () {

int i=1;

while (i<=100){

if(i%2!=0) {

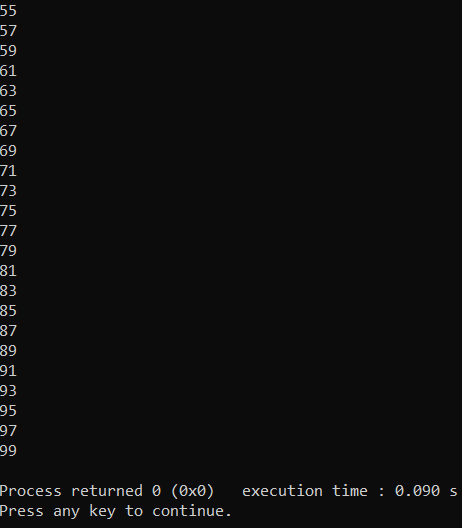
printf ("%d \n", i);

}

i++;}

return 0;

}



1. Scrieți un program C pentru a găsi suma tuturor numerelor naturale cuprinse între 1 și n.

#include <stdio.h>

int main () {

int sum=0, n, i=1;

printf ("n = ");

scanf ("%d", &n);

while (i<=n){

sum=i+sum;

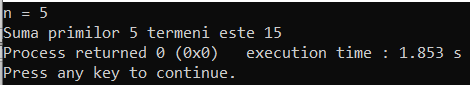
i++;

}

printf ("Suma primilor %d termeni este %d ", n, sum);

return 0;

}



1. Scrieți un program C pentru a găsi suma tuturor numerelor pare între 1 și n.

#include <stdio.h>

int main () {

int sum=0, n, i=1;

printf ("n = ");

scanf ("%d", &n);

while (i<=n){

if (i%2==0) {

sum=i+sum;

}

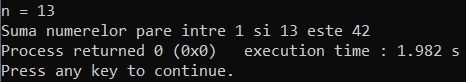
i++;

}

printf ("Suma numerelor pare intre 1 si %d este %d ", n, sum);

return 0;

}



1. Scrieți un program C pentru a găsi suma tuturor numerelor impare cuprinse între 1 și n.

#include <stdio.h>

int main () {

int sum=0, n, i=1;

printf ("n = ");

scanf ("%d", &n);

while (i<=n){

if (i%2!=0) {

sum=i+sum;

}

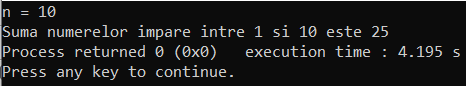
i++;

}

printf ("Suma numerelor impare intre 1 si %d este %d ", n, sum);

return 0;

}



1. Scrieți un program C pentru a imprima tabelul de înmulțire a oricărui număr.

#include <stdio.h>

int main () {

int i, n, p=1, z;

printf ("Numarul: ");

scanf ("%d", &n);

printf ("Tastati de cate ori doriti ca nr sa fie inmultit = ");

scanf ("%d", &z);

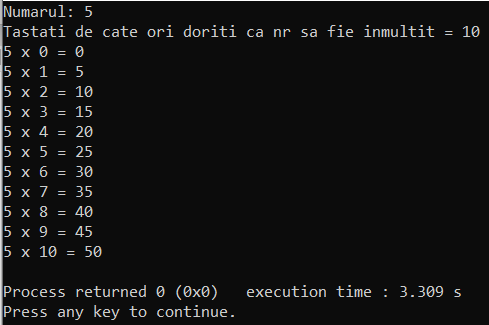
for (i=0;i<=z;i++){

printf ("%d x %d = %d \n", n, i,n\*i);

}

return 0;

}



1. Scrieți un program C pentru a număra numărul de cifre dintr-un număr.

#include <stdio.h>

int main () {

int n, nr;

printf ("Introduceti numarul: ");

scanf ("%d", &n);

while (n!=0) {

n=n/10;

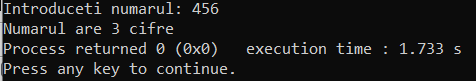
nr++;

}

printf ("Numarul are %d cifre", nr);

return 0;

}



1. Scrieți un program C pentru a găsi prima și ultima cifră a unui număr.

#include <stdio.h>

int main () {

int n, p;

printf ("n = ");

scanf ("%d", &n);

p=n;

while (n>10) {

n=n/10;

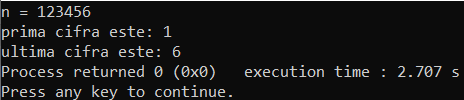
}

printf ("prima cifra este: %d \n", n);

printf ("ultima cifra este: %d", p%10);

return 0;

}



1. Scrieți un program C pentru a găsi suma primei și ultimei cifre a unui număr.

#include <stdio.h>

int main () {

int n, p;

printf ("n = ");

scanf ("%d", &n);

p=n;

while (n>10) {

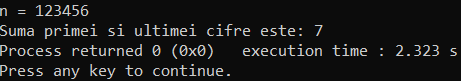
n=n/10;

}

printf ("Suma primei si ultimei cifre este: %d", n+(p%10));

return 0;

}



1. Scrieți un program C pentru a schimba prima și ultima cifră a unui număr.

#include <stdio.h>

#include <math.h>

int main () {

int p, u, i=0, n, num,num1;

printf ("Inainte de schimbare: ");

scanf ("%d", &n);

p=n;

num1=n;

u=n%10;

while (num1>0) {

num1=num1/10;

i++;

}

while (p>10){

p=p/10;

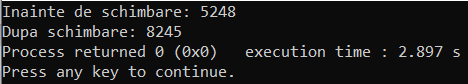
}

num=n-(p\*pow(10,i-1))+(u\*pow(10,i-1))-u+p;

printf("Dupa schimbare: %d", num);

return 0;

}



1. Scrieți un program C pentru a calcula suma cifrelor unui număr.

#include <stdio.h>

#include <math.h>

int main () {

int n, i, sum=0;

printf ("n: ");

scanf ("%d", &n);

while (n>0) {

i=n%10;

sum=i+sum;

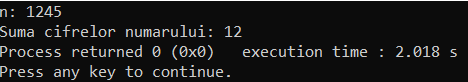
n=n/10;

}

printf ("Suma cifrelor numarului: %d", sum);

return 0;

}



1. Scrieți un program C pentru a calcula produsul cifrelor unui număr.

#include <stdio.h>

int main () {

int n, i, prod=1;

printf ("n: ");

scanf ("%d", &n);

while (n>0) {

i=n%10;

prod=i\*prod;

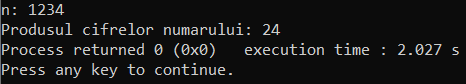
n=n/10;

}

printf ("Produsul cifrelor numarului: %d", prod);

return 0;

}



1. Scrieți un program C pentru a introduce un număr și imprimați invers.

#include <stdio.h>

int main () {

int n, i=0, temp;

printf ("n: ");

scanf ("%d", &n);

while (n!=0) {

temp=n%10;

i=temp+i\*10;

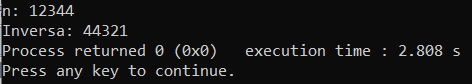
n=n/10;

}

printf ("Inversa: %d", i);

return 0;

}



1. Scrieți un program C pentru a verifica dacă un număr este palindrom sau nu.

#include <stdio.h>

int main () {

int n, i=0, initial, temp;

printf ("n: ");

scanf ("%d", &n);

initial=n;

while (n!=0) {

temp=n%10;

i=temp+i\*10;

n=n/10;

}

if (i==initial) {

printf ("Numarul este palindrom");

}

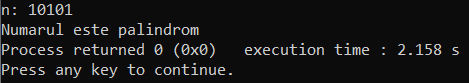
else {

printf ("Numarul nu este palindrom");

}

return 0;

}



1. Scrieți un program C pentru a calcula factorialul unui număr.

#include <stdio.h>

int main () {

int n, fact=1, i;

printf ("n = ");

scanf ("%d", &n);

for(i=1; i<=n; i++) {

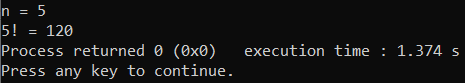
fact=fact\*i;

}

printf ("%d! = %d", n, fact);

return 0;

}



1. Scrieți un program C pentru a verifica dacă un număr este sau nu numărul Armstrong.

#include <stdio.h>

#include <math.h>

int main () {

int n, a, b, sum=0, temp;

printf ("n = ");

scanf ("%d", &n);

temp=n;

while (n>0) {

a=n%10;

sum=sum+pow(a,3);

n=n/10;

}

if (temp==sum){

printf ("%d este un numar Armstrong", sum);

}

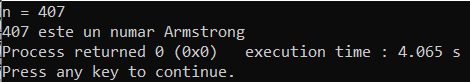
else {

printf ("%d nu este un numar Armstrong", sum);

}

return 0;

}



1. Scrieți un program C pentru a imprima seria Fibonacci până la n termeni.

#include <stdio.h>

int main () {

int n,i, n1,n2,n3;

printf ("n = ");

scanf ("%d", &n);

n1=0;

n2=1;

i=1;

printf ("Seria lui Fibonacci pana la %d termeni: ", n);

while (i<=n) {

printf ("%d ",n1);

n3=n1+n2;

n1=n2;

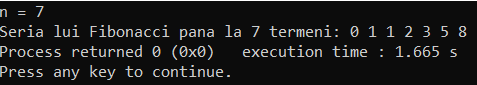
n2=n3;

i++;

}

return 0;

}



**Rezultatul lucrării:**

* Ce instrucțiuni de ramificare sunt în limbajul C?

În limbajul C cunoaștem instrucțiunile de ramificare *if* și *if-else*.

* Ce instrucțiuni ciclice sunt în limbajul C?

În limbajul C distingem următoarele instrucțiuni ciclice:

* instrucțiunea ciclică cu parametru (*for*);
* instrucțiunea ciclică precedată de condiție (*while*);
* instrucțiunea ciclică cu postcondiție (*do-while*)
* Care este diferența între while și do while?

1. Ciclul *while* este utilizat atunci când nu este cunoscut numărul de repetări ale ciclului și nu există necesitatea ca ciclul să fie executat măcar o dată; pe când instrucțiunea *do-while* se utilizează când cazul numărul de repetări nu este cunoscut, dar este necesar ca ciclul să fie îndeplinit cel puțin odată.
2. Ciclul *while* inițial verifică valoarea de adevăr a condiției, apoi îndeplinește instrucțiunea, iar apoi testează din nou instrucțiunea; pe când ciclul *do-while* mai întâi citește instrucțiunea, apoi calculează valoarea expresiei, ciclul continuându-se doar dacă valoarea expresiei este adevărată.

* Care este diferența între for și while?

Numărul de repetări ale ciclului *for* este cunoscut de la începutul executării lui, pe când ciclul *while* este folosit ăn cazul când nu este cunoscut numărul de repetări ale ciclului.

* Care este diferența între if și switch?

Ambele instrucțiuni sunt destinate pentru selectarea unei variante dintr-o mulțime posibilă de ramificări, însă în cazurile când sunt prezente mai mult de 3 condiții este mai rațională folosirea lui *switch*.

* Ce înseamnă break, continue?

Instrucțiunea *break* este una de întrerupere, executarea căreia provoacă întreruperea unui ciclu.

Instrucțiunea *continue* este una de continuare pentru a întrerupe o iterație a ciclului și de a trece la îndeplinirea iterației următoare a ciclului.