1. **CICLUL FOR şi INSTRUCŢIUNILE REPETITIVE DE TIP WHILE ŞI DO-WHILE**

**– CICLUL FOR -**

* 1. **SCOPUL LUCRĂRII**

Scopul lucrării este de a înţelege şi a exersa cele două tipuri de instrucţiuni repetitive din cadrul limbajului C/C++, şi anume instrucţiunile repetitive cu test iniţial (implementate prin instrucţiunile for – structura ciclică cu număr cunoscut de paşi – şi while) şi instrucţiunile repetitive cu test final, implementate prin instrucţiunea do while.

Se utilizează şi instrucţiunile adiacente – break şi continue.

* 1. **BREVIAR TEORETIC**

3.2.1 Instrucţiunea ***for***

este o structură ciclică cu test iniţial şi implementează ***structura ciclică cu număr cunoscut de paşi***

Sintaxa:

**for** (expresie1; expresie2; expresie3)

instructiune;

Funcţionarea:

evaluare ***expresie1***

ATÂTA TIMP CÂT ***expresie2* este *TRUE*** REPETĂ

begin

instrucţiune

evaluare ***expresie3***

end

Nu este obligatorie prezenţa expresiilor, ci doar a instrucţiunilor vide.

3.2.2 Instrucţiunea ***while***

este o structura ciclică cu test iniţial

Sintaxa:

**while**(<expresie>)

instr1;

La întâlnirea acestei instrucţiuni, se evaluează ***expresi***e. Dacă aceasta are valoarea ***TRUE*** (diferită de ZERO), se execută instrucţiunea ***instr1***. Se reevaluează valoarea ***expresie***-i. Dacă ea este tot 1, se repetă ***instrucţiune***, etc. Astfel, instrucţiunea (corpul ciclului) se repetă atât timp cât ***expresie*** are valoarea de adevăr ***TRUE***.

În momentul în care <***expresie***> are valoarea de adevăr ***FALSE*** (egală cu ZERO), se iese din ciclu şi se trece la următoarea instrucţiune din afara buclei ***while***.

În cazul în care la prima evaluare a expresiei, aceasta are valoarea de adevăr ***FALSE***, corpul instrucţiunii ***while*** nu va fi executat niciodată.

***instr1*** din corpul ciclului ***while*** poate fi compusă (un bloc).

Instrucţiunea/instrucţiunile din corpul ciclului while trebuie să modifice valoarea expresiei, altfel va fi un „ciclu infinit”.

*3.2.3* Instrucţiunea***do while***

Sintaxa:

***do*** instr1;

***while***(<expresie>)

Funcţionarea:

Se execută instrucţiunea ***instr1*** sau blocul de instrucţiuni. Se evaluează apoi <***expresie***>. Dacă aceasta are valoarea ***TRUE*,** se execută din nou ***instr1***, altfel se iese din buclă. Se testează din nou valoarea expresiei. Se repetă execuţia instrucţiunii ***instr1*** atâta cât timp valoarea expresiei este ***TRUE***. În cazul instrucţiunii ***do while*,** corpul ciclului se execută cel puţin o dată.

Instrucţiunea ***break***

forţează ieşirea din interiorul unei bucle, fără a se mai ţine seama de condiţia de menţinere în buclă. Instrucţiunile situate în corpul buclei după instrucţiunea ***break*** nu vor mai fi executate.

Intrucţiunea ***continue***

duce la ignorarea instrucţiunilor din buclă, situate după aceasta, şi testarea din nou a expresiei de menţinere în buclă. În cazul buclelor ***for,*** se realizează şi evaluarea celei de *a treia expresii* , responsabilă cu incrementarea contorilor.

* 1. **EXEMPLE**

1. ***Determinarea faptului dacă un număr n este sau nu prim***

*#include <conio.h>*

*#include <iostream.h>*

*#include <math.h>*

*void main() {*

*clrscr();*

*unsigned long n, d;*

*cout<<"n=?";cin>>n;*

*int prim=1; // presupunem ca numarul n este prim*

*for(d=2;d<=sqrt(n);d++) if(n%d==0) prim=0; //daca se divide la d, nu poate fi prim*

*if(prim) cout<<n<<" este prim";*

*else cout<<n<<" nu este prim";*

*getch();*

*}*

1. ***Media numerelor impare introduse de la tastatura***

*#include <conio.h>*

*#include <iostream.h>*

*void main(){*

*clrscr();*

*int n, i, x, nrimp=0; // nrprim – contor pentru numerele impare*

*float s=0, media;*

*cout<<"n=? "; cin>>n;*

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

*cout<<"x=? "; cin>>x;*

*if(x%2!=0){*

*nrimp++;*

*s=s+x;*

*}*

*}*

*media=s/nrimp;*

*cout<<"media="<<media;*

*getch();*

*}*

1. ***Calculul sumei***

*#include <conio.h>*

*#include <iostream.h>*

*void main(){*

*clrscr();*

*int n, i;*

*unsigned long int s=0;*

*cout<<"n=? "; cin>>n;*

*for(i=1; i<=n; i++) s=s+i\*(i+1);*

*cout<<"suma este "<<s<<endl;*

*getch();*

*}*

1. ***Calculul cmmdc al două numere, utilizând algoritmul lui Euclid***

*#include<iostream.h>*

*#include<conio.h>*

*void main(){*

*clrscr();*

*int a, b, p=0;*

*cout<<"a=";cin>>a;*

*cout<<"b=";cin>>b;*

*if(a==b)cout<<"cmmdc="<<a;*

*else while(a!=b){if(a>b)a=a-b;else b=b-a;p++;}*

*cout<<"cmmdc="<<a<<" in "<<p<<" pasi";*

*getch();*

*}*

*#include<iostream.h>*

*#include<conio.h>*

*void main(){*

*clrscr();*

*int a, b, r, p;*

*cout<<"a=";cin>>a;*

*cout<<"b=";cin>>b;*

*while(b){*

*r=a%b;*

*a=b;*

*b=r;*

*p++;*

*}*

*cout<<"cmmdc="<<a<<" in "<<p<<" pasi";*

*getch();*

*}*

* 1. **TEME DE LABORATOR**

1. Se introduce de la tastatură numărul natural ***n***; se citesc ***n*** numere întregi. Să se calculeze şi afişeze produsul numerelor cuprinse în intervalul [-5,5) , suma celor mai mari decât 20 şi numărul celor mai mici decât 100.
2. Aplicând faptul că n%10 reprezintă ultima cifră a unui număr natural n, iar operaţia n=n/10 realizează “scurtarea” cu o cifră a lui n, să se afişeze pe monitor cifrele constituente ale unui număr natural ***n*** de exact 5 cifre, introdus de la tastatură
3. De câte ori apare o cifră dată ***c***, în numarulnatural ***n*** de exact 7 cifre, introdus de la tastatură?
4. Să se calculeze suma cifrelor unui număr natural ***n*** de exact 6 cifre, introdus de la tastatură
5. Şirul lui Fibonacci este definit recursiv ca ai = ai-2 + ai-1, cu a0 = 0 şi a1 = 1; să se afişeze primii 20 de termeni din şir.
6. Pentru un număr ***n***, introdus de la tastarură, să se calculeze sumele primelor ***n*** termeni din
7. **Se citeşte de la tastatură o valoare naturală *n*. Să se afişeze numărul divizorilor numarului citit.**
8. **Scrieţi un program care afişează numerele divizibile cu 3 din intervalul [a,b]. Valorile *a* şi *b* se citesc de la tastatură.**
9. Se citeşte un număr întreg ***n*** şi n valori pentru parametrul real ***x***. Să se afişeze maximul şi minimul valorilor introduse.