

CAPITOLUL 2

Subprograme Definite de Utilizator

2.1. Subprograme implementate în manieră iterativă

2.1.1 Teste cu alegere multiplă și duală

1. Ce se va afișa în urma executării programului următor:

```
var x,y,z:byte;
procedure P(a,b:byte;var c:byte);
var aux:byte;
begin
  aux:=a; a:=b-a;
  c:=c-aux; b:=aux;
end;
begin
  x:=10; y:=100; z:=200;
  p(x,y,z);
  write(x,' ', y,' ',z,' ');
  p(x,y,z);
  write(x,' ',y,' ',z);
end.
```

```
#include <iostream.h>
int x,y,z;
void P(int a,int b, int &c)
{ int aux;
  aux=a; a=b-a;
  c=c-aux ; b=aux;
}

void main ()
{ x=10; y=100; z=200;
  P(x,y,z);
  cout<<x<<" " <<y<<" "<<z<<" ";
  P(x,y,z);
  cout<<x<<" " <<y<<" "<<z<<" ";
}
```

- a) 90 10 190 176 90 100
b) 10 10 190 10 10 180

- c) 10 100 190 10 100 190
d) 10 100 190 10 100 180

2. Ce se va afișa în urma executării programului următor:

```
var a:byte;
procedure P(x:byte; var y:byte);
begin
  y:=y*x; x:=x+y;
  write(x,' ',y,' ')
end;
begin
  a:=11;
  p(a,a);
  writeln(a);
end.
```

```
#include <iostream.h>
int a =11;
void P(int x,int &y)
{
  y=y*x; x=x+y;
  cout<<x<<" "<<y<<" ";
}

void main(){
  P(a,a);
  cout<<a;
}
```

- a) 11 121 11
b) 132 121 121

- c) 242 242 242
d) 11 121 121

3. În urma apelului $P(n,x)$ se dorește afișarea cifrei maxime a unui număr natural prin intermediul apelului $write(x)$, respectiv $cout<<x$. Identificați antetul corect al subprogramului P .

a)
function $P(n:integer):byte;$

b)
procedure $P(var\ a,b:integer);$

c)
procedure $P(n,x:integer);$

d)
procedure $P(x:integer;var\ n:byte);$

a)
int $P(int\ n)$

b)
void $P(int\ \&x,int\ \&y)$

c)
void $P(int\ n,\ int\ x)$

d)
void $P(int\ x,\ int\ \&n)$

4. Subprogramul P efectuează ștergerea dintr-un vector, transmis ca parametru, a tuturor elementelor egale cu o valoare cunoscută. Subprogramul primește, prin intermediul altor doi parametri întregi, lungimea tabloului și valoarea elementelor ce vor fi șterse. Identificați antetul corect al subprogramului P . Pentru varianta Pascal, se consideră următoarea definiție de tip: *type sir=array[1..100] of integer*.

a)
function $P(a:sir;$
 $n,x:integer;):sir;$

b)
procedure $P(var\ a:sir;$
 $n,x: integer);$

c)
procedure $P(var\ a:sir;var\ n:byte;$
 $x:integer);$

d)
procedure $P(a:sir; n,x:integer);$

a)
int $P(int\ a[100],\ int\ n,\ x)$

b)
void $P(int\ a,\ int\ \&x,\ int\ \&y)$

c)
void $P(int\ a[100],int\ \&n,\ int\ x)$

d)
void $P(int\ a[100],\ int\ x,\ int\ n)$

5. Ce valori vor fi afișate în urma executării următorului program?

```
var x:integer;

procedure p1(y:integer);
var x:integer;
begin
x:=y; x:=2; write(x,' '); y:=x;
end;
```

```
procedure p2(var y:integer);
begin
x:=y; x:=3; write(x,' '); y:=x;
end;
```

```
#include <iostream.h>
int x;
void p1(int y)
{ int x;
  x=y; x=2;
  cout<<x<<' '; y=x ;
}
```

```
void p2(int &y)
{ x=y; x=3;
  cout<<x<<' '; y=x;
}
```

```

begin
  x:=1;
  p1(x); write(x, ' ');
  p2(x); writeln(x);
end.

```

```

void main()
{ x=1;
  p1(x); cout<<x<<' ';
  p2(x); cout<<x<<endl;
}

```

a) 2 2 3 3

b) 2 2 3 2

c) 2 1 3 3

d) 2 1 3 2

6. Ce valori vor fi afișate în urma executării următorului program?

```

var x:integer;

procedure p1(x:integer);
var y:integer;
begin
  y:=10; x:=x + y; write(x, ' ');
end;

```

```

procedure p2(var x:integer);
var y:integer;
begin
  y:=x; x:=x + 3;
  x:=x + y; write(x, ' ');
end;

```

```

begin
  x:=20;
  p1(20); write(x, ' ');
  p2(x); write(x, ' ');
end.

```

```

#include <iostream.h>
int x;
void p1(int x)
{ int y=10;
  x=x+y;
  cout<<x<<' ';
}

```

```

void p2(int &x)
{int y;
  y=x; x+=3; x+=y;
  cout<<x<<' ';
}

```

```

void main()
{ x=20;
  p1(20); cout<<x<<' ';
  p2(x); cout<<x<<endl;
}

```

a) 20 20 43 20

b) 30 20 43 43

c) 30 20 43 20

d) 20 20 46 46

7. Ce valori vor fi afișate în urma executării următorului program?

```

var x,y:integer;

procedure p(var a,b:integer);
var x:integer;
begin
  x:=a*b; a:=a+x; b:=a+x;
  write(a, ' ',b, ' ',x, ' ');
end;

```

```

begin
  x:=5;y:=10;
  p(x,y); write(x, ' ',y);
end.

```

```

#include <iostream.h>
int x,y;

```

```

void p(int &a,int &b)
{ int x;
  x=a*b; a+=x; b=a+x;
  cout<<a<<' '<<b<<' '<<x<<' ';
}

```

```

void main()
{ x=5; y=10;
  p(x,y); cout<<x<<' '<<y;
}

```

a) 55 105 50 105 105

b) 55 55 50 55 55

c) 55 105 50 55 55

d) 55 105 50 55 105

8. Ce valori vor fi afișate în urma executării următorului program?

```
var n:integer;
function cif(var x:integer):byte;
var c:byte;
begin
  c:=x mod 10; x:=x div 10;
  cif:=(c+x mod 10)mod 10;
end;
begin
  n:=21987;
  writeln(cif(n)+cif(n))
end.
```

```
#include <iostream.h>
int n=21987;
int cif(int &x)
{
  int c;
  c=x%10; x/=10;
  return (c+ x%10)%10;
}
void main()
{ cout << cif(n)+cif(n);
}
```

- a) 12; b) 10; c) 32; d) 30.

9. Indicați care dintre următoarele antete de funcții sunt corecte sintactic:

a)
function F1(z byte);

b)
function F2(x,y):real;

c)
function F3(y:byte):text;

d)
function 12F(c:char):real;

e)
function F4(y:byte;z:byte):real;

a)
int F1(int z;)

b)
int F2(int x,y)

c)
FILE* F3(int y)

d)
float 12F(char c)

e)
float F4(int x, int y)

10. Identificați care dintre funcțiile următoare returnează un număr real, reprezentând media aritmetică dintre câtul și restul la împărțirea a două numere întregi, transmise ca parametri:

a)
function M(x,y:integer):byte;
begin
 M := (x div y + x mod y)/2;
end;

a)
int M(int x,y)
{
 return (x/y + x*y)/2;
}

b)
function M(x,y:integer):real;
var medie:real;
begin
 medie := x div y + x mod y;
 M := medie/2;
end;

b)
float M(int x,int y)
{
 float medie;
 medie = x/y + x*y;
 return medie/2;
}

c)
function M(x,y:integer):real;
begin
 M := x div y + x mod y;
 M := M/2;
end;

c)
float M(int x,int y)
{
 return x/y + x*y;
}

```

d)
function M(x,y:integer):real;
begin
  x := x div y;
  y := x mod y;
  M := (x + y)/2;
end;

```

```

d)
float M(int x,int y)
{
  x = x/y ;
  y = x%y ;
  return (x+y)/2;
}

```

11. Considerăm definită funcția *prim*, care returnează *true* în varianta Pascal, respectiv 1 în varianta C++, dacă valoarea întreagă transmisă ca parametru este un număr prim și *false* / 0, în caz contrar. Considerăm un tablou unidimensional *a* ce conține *n* elemente de valori întregi. Identificați care dintre următoarele funcții returnează indicele din *a* al primului element număr prim. Dacă acesta nu conține nici un element prim, se returnează valoarea -1.

Pentru varianta Pascal, se consideră următoarea definiție de tip: *type sir=array[0..100] of integer*, deci primul element este *a[0]*.

```

a)
function p(a:sir;n:byte):integer;
var i:integer;
begin
  p:=-1;
  for i:=0 to n-1 do
    if prim(a[i]) then p:=i;
  end;

```

```

b)
function p(a:sir;n:byte):integer;
var x:integer;
begin
  x:=-1;
  repeat
    inc(x)
  until (prim(a[x])) or (x=n);
  if x=n then p:=-1
    else p:=x;
end;

```

```

c)
function p(a:sir;n:byte):integer;
var x,i:integer;
begin
  x:=-1;
  while x<n-1 do begin
    inc(x);
    if prim(a[x])=true then x:=n;
  end;
  p:=x;
end;

```

```

a)
int p1(int a[101], int n)
{
  int i,x=-1;
  for(i=0;i<n;i++)
    if (prim(a[i])) x=i;
  return x;
}

```

```

b)
int p2(int a[101], int n)
{
  int x=-1;
  do x++;
  while (!prim(a[x]) && (x<n));
  if (x==n) return -1;
  else return x;
}

```

```

c)
int p3(int a[101], int n)
{
  int i,x=-1;
  while (x<n-1) {
    x++;
    if (prim(a[x])) return x;
  }
  return x-n-1;
}

```

```

d)
function p(a:sir;n:byte):integer;
var i:integer;
begin
  p:=-1;
  for i:=0 to n-1 do
    if prim(i) then
      begin
        p:=i;
        exit;
      end
    end
  end;
end;

```

```

d)
int p4(int a[101], int n)
{
  int i,x=0;
  for(i=0;i<n;i++)
    if (prim(i)) x=i;
  return x;
}

```

12. Considerând următorul program, ce valori vor fi afișate în urma executării acestuia?

```

type sir=array[1..10] of integer;
var a:sir; i,n:integer;

procedure x(var a:sir;
             var n:integer);
var i,j:integer;
begin
  i:=1;
  while i<=n do
    if a[i]<0 then begin
      for j:=i to n-1 do
        a[j]:=a[j+1];
      dec(n);
    end
    else inc(i);
  end;

begin
  n:=6;
  a[1]:=-3;
  a[2]:=2; a[3]:=-1;
  for i:=1 to n div 2 do
    a[n-i+1]:=a[i];
  x(a,n);
  for i:=1 to n do write(a[i], ' ');
end.

```

```

#include<iostream.h>
int i,n=6,a[10];

void x(int a[10], int& n)
{int i=0,j;
  while (i<n) {
    if (a[i]<0){
      for(j=i;j<n-1;j++)
        a[j]=a[j+1];
      n--;
    }
    else i++;
  }
}

void main()
{
  a[0]=-3;
  a[1]=2;
  a[2]=-1;
  for(i=n/2;i<n;i++)
    a[i]=a[n-i-1];
  x(a,n);
  for(i=0;i<n;i++)
    cout<<a[i]<<' ';
}

```

- a) -3 -1 -1 -3
b) -2 -2

- c) 2 2
d) 3 3 3 3

13. Considerăm un tablou unidimensional a , ce conține n elemente de valori întregi. Identificați care dintre următoarele funcții returnează indicele din a al primului element de valoare 0. Dacă acesta nu conține nici un element nul, se returnează valoarea -1. Pentru varianta Pascal, se consideră următoarea definiție de tip: *type sir=array[0..100] of integer*, deci primul element este $a[0]$.

```

a)
function p(a:sir;n:byte):integer;
var i:integer;
begin
  p:=-1;
  for i:=0 to n-1 do
    if a[i]=0 then p:=i;
  end;

```

```

b)
function p(a:sir;n:byte):integer;
var x:integer;
begin
  x:=-1;
  repeat
    inc(x)
  until (a[x]=0) or (x=n);
  p:=x; end;

```

```

c)
function p(a:sir;n:byte):integer;
var x,i:integer; ok:boolean;
begin
  x:=-1; ok:=true;
  while (x<n-1) and ok do begin
    inc(x);
    if (a[x]=0) then ok:=false;
  end;
  if ok then p:=-1 else p:=x;
end;

```

```

d)
function p(a:sir;n:byte):integer;
var x:integer;
begin
  x:=-1;
  while ((a[x+1]<>0) and (x<n-1)) do
    inc(x);
  p:=x+1;
end;

```

```

a)
int p(int a[101], int n)
{
  int i,x=0;
  for(i=0;i<n;i++)
    if (a[i]==0) x=i;
  return x;
}

```

```

b)
int p(int a[101], int n)
{
  int x=-1;
  do x++;
  while ((a[x]!=0)&&(x<n));
  return x;
}

```

```

c)
int p(int a[101], int n)
{
  int i,x=-1;
  while (x<n-1)
  {
    x++;
    if (a[x]==0)
      return x;
  }
  return x-n;
}

```

```

d)
int p(int a[101], int n)
{
  int x=-1;
  while ((a[x+1]!=0)&&(x<n))
    x++;
  return x+1;
}

```

14. Ce se va afișa în urma executării următorului program:

```

var a:string; var b:char;

procedure x(var s:string;
            var c:char);
begin
  while c=s[length(s)] do
    delete(s,length(s),1);
  if length(s)>0 then
    c:=upcase(s[length(s)])
  else c:= '.';
end;

```

```

#include <iostream.h>
#include <string.h>
char *a, b;

void x(char *s, char &c) {
  while (c==s[strlen(s)-1])
    s[strlen(s)-1]=0;
  if (strlen(s)>0)
    c=s[strlen(s)-1],
    c=c>='a'&&c<='z'?c+'A'-'a':c;
  else c='.';
}

```

```

begin
  a:='copiii'; b:='i';
  x(a,b);
  writeln(a,b);
end.

```

```

void main() {
  a = "copiii"; b = 'i';
  x(a,b);
  cout<<a<<b<<endl;
}

```

a) copP

b) copiii.

c) copiiii

d) copiiiP

15. Care dintre următoarele funcții returnează cea mai mare putere a lui 2 care este mai mică sau egală cu o valoare naturală (<15), transmisă ca parametru?

a)

```

function F(n:byte):integer;
var x:integer;
begin
  x:=-1;
  repeat
    inc(x);
  until 1 shl x>n;
  F:=1 shl (x-1);
end;

```

b)

```

function F(n:byte):integer;
var x:integer;
begin
  x:=1;
  while (x<n) do x:=x*2;
  F:=x;
end;

```

c)

```

function F(n:byte):integer;
var x:integer;
begin
  x:=1;
  repeat
    x:=x*2;
  until x>=n;
  F:=x-1;
end;

```

d)

```

function F(n:byte):integer;
var i,x:integer;
begin
  x:=1;
  for i:=1 to n do begin
    x:=x*2;
    if x<=n then begin
      F:=x; exit;
    end;end;
  F:=x-1;
end;

```

a)

```

int F(int n)
{
  int x=-1;
  do
    x++;
  while (1 << x <= n );
  return 1 << (x-1);
}

```

b)

```

int F(int n)
{
  int x=1;
  while (x<n) x*=2;
  return x;
}

```

c)

```

int F(int n)
{
  int x=1;
  do
    x*=2;
  while (x<n);
  return x-1;
}

```

d)

```

int F(int n)
{
  int i,x;
  x=1;
  for(i=1;i<=n;i++)
  {
    x*=2;
    if (x<=n) return x;
  }
  return x-1;
}

```


16. Ce se va afișa în urma executării următorului program:

```
var a:real;
function F(var x:real):real;
begin
  x:=abs(x*10);
  F:=int(x/10) + frac(x);
end;
begin
  a:=15.25;
  write(F(a):0:2, ' ');
  write(F(a):0:2, ' ');
  writeln(a:0:2);
end.
```

```
#include <stdio.h>
float F(float &x)
{ x=(x > 0 ? x: -(x));
  x*=10;
  return int(x/10) + x - int(x);
}
void main()
{float a=15.25;
 printf("%0.2f ",F(a));
 printf("%0.2f ",F(a));
 printf("%0.2f ",a);
}
```

- a) 15.50 15.50 1525.00
b) 15.50 15.50 152.50

- c) 15.50 152.00 1525.00
d) 15.50 152.00 152.50

17. Câte elemente divizibile cu 10 se vor afișa în urma executării programului următor:

```
var x,y:byte;
function F(var y:byte;x:byte):byte;
begin
  y:=y div 10 + x;
  F:= x+y ;
end;
begin
  x:=101; y:=10;
  write(F(x,y), ' ');
  write(x, ' ', y, ' ');
  writeln(F(x,y));
end.
```

```
#include <iostream.h>
int F(int &y, int x)
{
  y = y/10 + x;
  return x + y;
}
main()
{int x = 101, y = 10;
 cout<<F(x,y)<<" ";
 cout << x <<" "<< y<<" ";
 cout<<F(x,y);
}
```

- a) 1 b) 2 c) 3 d) 4

18. În urma apelului $write(P(n, x))$, respectiv $cout<<P(n, x)$, se dorește afișarea numărului de apariții al cifrei x în scrierea zecimală a numărului întreg n . Identificați antetul corect al subprogramului P .

- a)
procedure $P(n,x:integer)$
- b)
function $P(x,c:integer):integer$
- c)
function $P(n:real;x:byte):byte$
- d)
function $P(x:integer;n:char):byte$

- a)
void $P(int\ n, int\ x)$
- b)
int $P(int\ x, int\ c)$
- c)
int $P(int\ n[10], int\ x)$
- d)
void $P(int\ x, float\ n)$

19. Considerăm subprogramul următor:

```
function P(var x:integer):byte;
var s:byte;
begin
  s:=0;
  while x>0 do begin
    s:=s+x mod 2;  x:=x div 2;
  end;
  P:=s;
end;
```

```
int P(int &x)
{ int s=0;
  while (x!=0)
  {
    s+=x%2;
    x/=2;
  }
  return s;
}
```

Considerăm că variabila întreagă n , declarată global, are valoarea 32. Ce se va afișa în urma apelului $write(P(n)+P(n)+1)$, respectiv $cout<<P(n)+P(n)+1$?

- a) 2; b) 3; c) 4; d) 1.

20. Considerăm secvența de declarații următoare:

```
var x:array[1..10]of byte;

procedure P1(n:integer);
var x:integer;
begin
  ...
end;

procedure P2(n,x:real);
begin
  ...
end;

begin
  ...
end.
```

```
int x[100];

void P1(int n)
{ char x;
  ...
}

void P2(float n,float x)
{
  ...
}

void main ()
{
  ...
}
```

Care dintre următoarele afirmații este adevărată:

- a) Elementul $x[1]$ poate fi referit oriunde în program, deci în oricare subprogram $P1$ sau $P2$;
 b) Programul sursă conține erori de sintaxă;
 c) Elementul $x[1]$ poate fi referit doar în programul principal (varianta Pascal), respectiv numai în funcția $main$ (varianta C++);
 d) Elementul $x[1]$ poate fi referit doar în subprogramele $P1$ și $P2$.

2.1.2 Probleme rezolvate

1. Se consideră un vector cu n ($1 \leq n \leq 40$) componente numere naturale mai mici sau egale cu 60000. Să se realizeze un subprogram care, primind prin intermediul a doi parametri tabloul respectiv și lungimea sa, întoarce printr-un alt parametru, cel mai mare element prim. Dacă nu există, se va returna valoarea -1. Considerăm că există definită funcția *prim* care verifică dacă valoarea naturală primită ca parametru este un număr prim. Ea returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului este sau nu număr prim.

Exemplu: Pentru $n=5$ și tabloul (1, 6, 7, 55, 19), subprogramul va returna prin intermediul unui parametru valoarea "19".

Soluție:

Subprogramul va avea trei parametri, dintre care unul referință. Datorită faptului că valoarea întoarsă poate fi -1, tipul parametrului nu poate aparține un tip de date fără semn. Pentru varianta Pascal este definit tipul de date *sir*, iar subprogramul realizat este o procedură, datorită faptului că rezultatul este returnat prin intermediul unui parametru. Tot din acest motiv, în varianta C++, funcția realizată are tipul *void*.

<pre>1 type sir=array[1..40] of word; 2 procedure P(x:sir; n:byte; 3 var y:longint); 4 var i:byte; 5 begin 6 y:=-1; 7 for i:=1 to n do 8 if (prim(x[i]))and(y<x[i]) 9 then y:=x[i]; 10 end;</pre>	<pre>void P(unsigned int x[40], int n, long &y) { int i; y = -1; for(i = 0; i < n; i++) if (prim(x[i])&&(x[i]>y)) y = x[i]; }</pre>
---	---

2. Se consideră un vector cu n ($1 \leq n \leq 50$) componente numere naturale, mai mici sau egale cu 30000. Să se realizeze un subprogram care, primind prin intermediul a doi parametri tabloul respectiv și lungimea sa, afișează toate elementele care conțin un număr maxim de cifre în reprezentarea în baza 10. Considerăm că există definită funcția *nrc* care returnează, pentru o valoarea naturală primită printr-un parametru, numărul de cifre din reprezentarea acesteia în baza 10.

Exemplu: Pentru $n=4$ și tabloul (231, 54, 809, 2), subprogramul va afișa 231, 809.

Soluție:

Subprogramul va avea doi parametri valoare. Pentru varianta Pascal, subprogramul realizat este o procedură, datorită faptului că operația pe care trebuie să o realizeze este cea de afișare, fără să fie necesar transferul unor rezultate. Tot din acest motiv, în varianta C++, funcția realizată are tipul *void*. Pentru varianta Pascal se consideră definit tipul de date *sir=array[1..50] of word;*.

```

1 procedure P(x:sir; n:byte);
2   var m,i:byte;
3   begin
4     m:=0;
5     for i:=1 to n do
6       if (nrc(x[i])>m) then
7         m:=nrc(x[i]);
8     for i:=1 to n do
9       if (nrc(x[i])=m) then
10        write(x[i], ' ');
11   end;

```

```

void P(int x[50],int n)
{
  int i,m=0;
  for (i=0; i<n;i++)
    if (nrc(x[i])>m)
      m=nrc(x[i]);
  for (i=0; i<n;i++)
    if (nrc(x[i])==m)
      cout<<x[i]<<' ';
}

```

3. Se citesc de la tastatură n ($1 \leq n \leq 50$) intervale de numere întregi de forma $[a, b]$, introduse prin limitele a și b ($-100 \leq a \leq b \leq 100$). Să se realizeze un subprogram care efectuează operația de citire a intervalelor și returnează numărul maxim de valori palindrom incluse într-unul dintre intervalele citite, fără a folosi vreun parametru. Considerăm că există definită funcția *pal* care verifică dacă valoarea naturală primită printr-un parametru reprezintă un număr palindrom. Ea returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului este sau nu palindrom.

Exemplu: Pentru $n=4$ și intervalele $[32, 44]$, $[2, 4]$, $[5, 12]$, $[56, 62]$, subprogramul va returna valoarea 6, deoarece intervalul $[5, 12]$ conține 6 numere palindrom.

Soluție:

În enunț se specifică faptul că rezultatul nu va fi transferat prin intermediul parametrilor referință. Deoarece tipul valorii rezultat aparține unui tip de date simplu, este posibilă realizarea unei funcții al cărei rezultat va fi de tip *byte* / *unsigned int*.

Funcția nu va realiza transfer de date prin parametri valoare, deoarece limitele intervalelor citite și numărul acestora vor reprezenta variabile locale.

```

1 function citește:integer;
2   var i,j,nr,m,n,a,b:byte;
3   begin
4     m:=0; readln(n);
5     for i:=1 to n do begin
6       readln(a,b); nr:=0;
7       for j:=a to b do
8         if pal(j) then inc(nr);
9       if nr>m then m:=nr;
10    end;
11    citește:=m;
12  end;

```

```

unsigned int citește()
{
  int i,j,nr,m,n,a,b;
  m=0; cin>>n;
  for (i=1;i<=n;i++){
    cin>>a; cin>>b; nr=0;
    for (j=a;j<=b;j++)
      if (pal(j)) nr++;
    if (nr>m) m=nr;
  }
  return m;
}

```

4. Subprogramul *L* permite înlocuirea elementelor unui vector *a*, transmis ca parametru, cu suma factorialelor cifrelor lor, dacă această sumă nu se află într-un interval $[x,y]$. Subprogramul va primi prin intermediul a trei parametri întregi, lungimea vectorului n ($n \leq 50$) și limitele intervalului. El va face apel la funcția *S*,

care returnează suma factorialelor cifrelor unui număr natural transmis printr-un parametru. Scrieți definițiile complete ale subprogramelor L și S . Inițial, elementele vectorului sunt numere întregi mai mici ca 10.000.

Exemplu: Pentru $n=5$, $x=5$ $y=35$ și tabloul $a=(10, 234, 21, 148, 34)$ la finalul execuției subprogramului L , elementele lui a vor fi $(2, 234, 3, 40345, 34)$.

Soluție:

Subprogramul va avea patru parametri, dintre care unul referință (tabloul). Pentru varianta Pascal este definit tipul de date *sir*, iar subprogramul realizat este o procedură, datorită faptului că rezultatele sunt returnate prin intermediul unui parametru (ce nu aparține unui tip de date simplu). Tot din acest motiv, în varianta C++, funcția realizată are tipul *void*.

Valoarea returnată de funcția S este preluată în variabila locală v , pentru a se evita apelul repetat pentru o singură valoare a parametrului.

<pre> 1 type 2 sir=array[1..50]of longint; 3 4 function S(x:integer):longint; 5 var sm,p,i:longint; 6 begin 7 sm:=0; 8 while x<>0 do begin 9 p:=1; 10 for i:=1 to x mod 10 do 11 p:=p*i; 12 sm:=sm+p; x:=x div 10; 13 end; 14 s:=sm; 15 end; 16 procedure L(var a:sir; 17 n,x,y:integer); 18 var i,v:longint; 19 begin 20 for i:=1 to n do begin 21 v:=S(a[i]); 22 if (v<x)or(v>y)then a[i]:=v 23 end; 24 end;</pre>	<pre> long S (int x) { long s,p,i; s=0; while (x!=0) { p=1; for (i=1;i<=x%10;i++) p*=i; s+=p; x/=10; } return s; } void L(long a[50],int n, int x,int y) { long i,v; for (i=0;i<n;i++){ v=S(a[i]); if ((v<x) (v>y)) a[i]=v; } }</pre>
--	--

5. Realizați un subprogram S , care să permită ordonarea elementelor unui vector a de numere întregi, transmis ca parametru. Subprogramul va primi, prin intermediul parametrilor întregi x și y , indicii elementelor între care se va face sortarea. Tipul ordonării care se dorește a fi realizată (ascendentă-descendentă) va fi transmis codificat prin parametrul întreg k . Dacă la apel, acestuia i se transmite valoarea 1, atunci se va realiza o ordonare ascendentă, iar dacă valoarea transmisă acestuia este -1 , sortarea va fi descendentă.

Exemplu: Considerăm vectorul $a=(10, 234, 21, 1, 34)$. Subprogramul S apelat de două ori, pentru ordonarea crescătoare a primelor trei elemente și descrescătoare a următoarelor două, modifică elementele tabloului astfel: $(10, 21, 234, 34, 1)$.

Soluție:

Subprogramul va avea patru parametri, dintre care unul referință (tabloul a). Pentru varianta Pascal este definit tipul de date *sir=array[1..50] of integer*, iar subprogramul realizat este o procedură, datorită faptului că rezultatele sunt returnate prin intermediul unui parametru (ce nu aparține unui tip de date simplu). Tot din acest motiv, în varianta C++ funcția realizată are tipul *void*. Inegalitatea care intervine în procesul de ordonare $k*a[i]>k*a[j]$ își schimbă semnul când valoarea transmisă lui k este -1, realizându-se astfel o ordonare descrescătoare.

```
1  procedure S(var a:sir;  
2      k,x,y:integer);  
3  var i,j,t:integer;  
4  begin  
5      for i:=x to y-1 do  
6          for j:=i+1 to y do  
7              if k*a[i]>k*a[j] then  
8                  begin  
9                      t:=a[i];  
10                     a[i]:=a[j];  
11                     a[j]:=t  
12                 end;  
13  end;
```

```
void S(int a[50],int k,  
      int x,int y)  
{  
    int i,j,t;  
    for (i=x;i<y;i++)  
        for (j=i+1;j<=y;j++)  
            if (k*a[i]>k*a[j]){  
                t=a[i];  
                a[i]=a[j];  
                a[j]=t;  
            }  
}
```

6. Funcția S verifică dacă un tablou unidimensional, primit prin intermediul parametrului a , reprezintă o permutare, fără puncte fixe, a mulțimii numerelor de la 1 la n ($n \leq 50$, transmis ca parametru întreg). Subprogramul returnează, în Pascal, valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum tabloul primit reprezintă sau nu o permutare fără puncte fixe.

Spunem că o permutare $(a_1, a_2 \dots a_n)$ are puncte fixe dacă există cel puțin un element $a_i = i$ ($1 \leq i \leq n$). De exemplu, pentru $n=4$, permutarea $(2,4,3,1)$ are puncte fixe deoarece $a_3=3$.

Subprogramul S va apela funcția nr , care determină numărul de apariții al unei valori într-un vector. Atât valoarea căutată, cât și tabloul și lungimea acestuia sunt primite de către funcția nr prin intermediul parametrilor. Scrieți definițiile complete ale subprogramelor S și nr .

Exemplu: Pentru $n=5$ și oricare dintre tablourile $(2, 4, 5, 1, 2)$ sau $(2, 1, 5, 4, 3)$ se va returna valoarea *False/0*.

Soluție:

Funcția S va avea doi parametri: tabloul a și n , lungimea acestuia. Funcția nr va fi apelată pentru a verifica dacă tabloul a reprezintă o permutare a mulțimii numerelor de la 1 la n . Pentru varianta Pascal este definit tipul de date *sir=array[1..50] of byte*.

```

1 function nr(a:sir;
2             n,x: byte):byte;
3 var m:byte;
4 begin
5   m:=0;
6   for i:=1 to n do
7     if a[i]=x then inc(m);
8   nr:=m;
9 end;
10
11 function S(a:sir;n:byte):
12           boolean;
13 var i:byte;
14 begin
15   S:=true;
16   for i:=1 to n do
17     if (nr(A,n,i)<>1) or (a[i]=i)
18   then S:=false;
19 end;

```

```

int nr(int a[50],int n,int x)
{
  int m=0;
  for (i=0;i<n;i++)
    if (a[i]==x)
      m++;
  return m;
}

int S(int a[50],int n)
{
  int i;
  for (i=1;i<=n;i++)
    if ((nr(a,n,i)!=1) ||
        (a[i-1]==i))
      return 0;
  return 1;
}

```

7. Subprogramul *X* primește, prin intermediul a doi parametri, un tablou unidimensional de numere întregi și numărul de elemente al acestuia (≤ 100). El returnează, prin intermediul altor doi parametri întregi, cifra maximă ce apare în scrierea din baza 10 a elementelor vectorului și numărul de apariții al acesteia.

În cadrul subprogramului *X* se apelează subprogramul *D*, care determină, pentru o valoare întreagă, transmisă ca parametru, cifra maximă a sa și numărul de apariții al acesteia. Aceste două valori sunt returnate de *D* prin intermediul a doi parametri întregi.

Scrieți definițiile complete ale celor două subprograme.

Exemplu: Pentru tabloul ce conține 5 elemente: (22, 405, 5125, 102, 53), subprogramul *X* va returna două valori 5 și 4, reprezentând cifra maximă a elementelor din vector și numărul de apariții al acesteia.

Soluție:

Pentru varianta Pascal, ambele subprograme vor fi implementate ca proceduri, rezultatele fiind returnate, pentru fiecare, prin intermediul a doi parametri referință. Pentru varianta C++, funcțiile vor fi, din același motiv, de tip *void*. Subprogramul *D* va determina, pentru fiecare element din vector, cifra maximă și numărul de apariții al acesteia.

Pentru varianta Pascal considerăm definit tipul de date *sir=array[1..100] of integer*.

```

1 procedure D(x:integer;
2             var c,m:byte);
3 begin
4   m:=0; c:=0;
5   while x<>0 do begin
6     if c<x mod 10 then begin
7       c:=x mod 10; m:=1;
8     end

```

```

void D(int x,int &c, int &m)
{
  m=0; c=0;
  while (x!=0){
    if (c<x%10){
      c=x%10;
      m=1;
    }

```

```

9  else
10     if c=x mod 10 then inc(m);
11     x:=x div 10;
12 end;
13 end;
14
15 procedure X(a:sir; n:byte;
16             var cm,nm:byte);
17 var i,c,m:byte;
18 begin
19     cm := 0; nm := 0;
20     for i:=1 to n do begin
21         D(a[i],c,m);
22         if c=cm then nm:=nm+m
23         else if c>cm then begin
24             cm:=c;
25             nm:=m
26         end
27     end
28 end;

```

```

else
    if (c=x % 10) m++;
    x=x / 10;
}
}

void X(int a[100],int n,
      int &cm,int &nm)
{
    int i,c,m;
    cm = 0; nm = 0;
    for (i=0;i<n;i++){
        D(a[i],c,m);
        if (c==cm) nm+=m;
        else
            if (c>cm)
                {cm=c; nm=m;}
    }
}

```

8. Considerăm o matrice pătratică cu n linii și n coloane ($n \leq 10$), cu elemente valori strict pozitive. Funcția L determină numărul de ordine al liniei ce conține cele mai multe elemente care sunt termeni ai șirului lui Fibonacci. Matricea și numărul n de linii sunt transmise funcției prin intermediul a doi parametri.

În cadrul funcției L se va face apel la funcția Fib care verifică dacă o valoare naturală transmisă ca parametru reprezintă un termen al șirului lui Fibonacci. Ea returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului este sau nu termen al șirului.

Scrieți definițiile complete ale subprogrameelor L și Fib .

Exemplu: Pentru $n=3$ și tabloul :

```

2 14 8
3 5 21
21 4 4

```

funcția L va returna valoarea 2.

Soluție:

Funcția Fib verifică dacă o valoare naturală x reprezintă un termen al șirului lui Fibonacci folosind trei variabile locale de tip întreg. Ea este apelată de funcția L pentru fiecare element din matrice, contorizându-se în același timp numărul de valori corecte de pe fiecare linie. Pentru varianta Pascal, considerăm definit tipul de date $mat=array[1..10,1..10]$ of integer.

```

1  function Fib(x:integer):boolean;
2  var a,b,c:integer;
3  begin
4      a := 1; b := 1;
5      while a+b<=x do begin
6          c := a + b; a := b; b := c;
7      end;
8      fib := (c = x) or (x = 1);
9  end;

```

```

int Fib(int x)
{
    int a,b,c;
    a = 1; b = 1;
    while (a + b<=x){
        c = a + b; a = b; b = c;
    }
    return (c==x) || (x==1);
}

```



```

10 Function L(a:mat; n:byte):byte;
11 var i,j,m,mx:byte;
12 begin
13   mx := 0;
14   for i:=1 to n do begin
15     m := 0;
16     for j:=1 to n do
17       if Fib(a[i,j]) then inc(m);
18     if m > mx then begin
19       L := i; mx := m;
20     end
21   end;
22 end;

```

```

int L(int a[10][10],int n)
{int l,i,j,m,mx;
mx=0;
for (i=0;i<n;i++){
m=0;
for (j=0;j<n;j++){
if (Fib(a[i][j])) m++;
if (m > mx){
l = i; mx = m;
}
}
}
return l+1;
}

```

9. Fișierul text *in.txt* conține, pe o singură linie, mai multe numere naturale mai mici ca 30000, separate prin câte un spațiu. Funcția Z permite citirea tuturor valorilor din fișier și returnează numărul de zerouri în care se termină produsul valorilor citite.

În cadrul ei este apelat subprogramul *Exp* care returnează, prin intermediul unui parametru întreg, exponentul la care apare un număr prim în descompunerea unui număr natural. Ambele valori sunt primite de subprogram prin intermediul a doi parametri întregi.

Scrieți definițiile complete ale subprogramelor Z și *Exp*.

Exemplu: Considerând că fișierul *in.txt* conține valorile 24, 25, 15, 10, 150, atunci funcția Z va returna valoarea 5.

Soluție:

Numărul de zerouri în care se termină produsul unor numere este egal cu cel mai mic exponent al factorilor de 2 și de 5, care apar în descompunerea lor.

Funcția Z va apela subprogramul *Exp* pentru a determina acești exponenți. În varianta Pascal, subprogramul *Exp* va fi implementat ca procedură deoarece se impune ca valoarea exponentului să fie returnată lui X prin intermediul unui parametru referință. Din același motiv, în varianta C++, funcția *Exp* va avea tipul *void*.

```

1 procedure Exp(x,f:integer;
2               var e:integer);
3 begin
4   while x mod f=0 do begin
5     inc(e);
6     x:=x div f;
7   end;
8 end;
9
10 function Z:integer;
11 var f:text;e5,e2,x:integer;
12 begin
13   e5:=0; e2:=0;
14   assign(f,'in.txt'); reset(f);

```

```

void Exp(int x,int f,int &e)
{
while (x%f==0)
{
e++;
x/=f;
}
}

int Z()
{
FILE *f; int e5,e2,x;
e5=e2=0;
f=fopen("in.txt","r");

```

```

15 while not eof(f) do begin
16   read(f,x);
17   Exp(x,5,e5); Exp(x,2,e2);
18 end;
19 close(f);
20 if e5<e2 then Z:=e5
21 else Z:=e2;
22 end;

```

```

while (fscanf(f,"%d",&x)==1)
{
  Exp(x,5,e5); Exp(x,2,e2);
}
fclose(f);
return e5<e2 ? e5 : e2;
}

```

10. Se consideră un număr natural x , citit de la tastatură. Să se realizeze un program care determină cel mai apropiat număr de x care reprezintă factorialul unei valori. Programul va conține două subprograme:

- funcția P , care verifică dacă o valoare transmisă ca parametru poate fi scris sub forma $k!$ ($1 \leq k$)
- funcție A , care are doi parametri întregi x și t . Ea determină cel mai mic număr mai mare ca x sau cel mai mare număr mai mic decât x de forma $k!$, după cum valoare unui parametru t este 1 sau -1 la apel.

Exemplu: Pentru $x=10$ se va afișa 6 ($6=3!$) iar pentru $x=22$ se va afișa 24 ($24=4!$).

Soluție:

Funcția A va mări sau micșora valoarea inițială a parametrului x , până când acesta va avea o valoare de forma $k!$. Valoarea primită la apel de parametru valoare t va reprezenta valoarea cu care se va modifica x în cadrul fiecărei iterații.

Funcția P returnează, în Pascal, valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului reprezintă sau nu factorialul unei valori.

```

1  var x:integer;
2  function P(x:integer):boolean;
3  var i,y:integer;
4  begin
5    y:=1; i:=1;
6    while y<x do begin
7      y:=y*i; inc(i)
8    end;
9    p:= y=x;
10 end;
11
12 function A(x,t:integer):integer;
13 begin
14   while (not P(x)and(x>0)) do
15     x:=x+t;
16   A:=x;
17 end;
18
19 begin
20   readln(x);
21   if x-A(x,-1)<A(x,1)-x then
22     write(A(x,-1))
23   else write(A(x,1));
24 end.

```

```

#include <iostream.h>
int x;

int P(int x)
{ int i,y;
  y=1; i=1;
  while (y<x) {
    y*=i; i++;
  }
  return (y==x);
}

int A(int x,int t)
{
  while (!P(x)&&(x>0))  x+=t;
  return x;
}

void main ()
{ cin>>x;
  if (x-A(x,-1)<A(x,1)-x)
    cout<<A(x,-1);
  else cout<<A(x,1);
}

```

11. Considerăm o matrice pătratică cu n linii și n coloane ($n \leq 10$), cu elemente valori strict pozitive. Funcția L determină numărul de ordine al liniei ce conține cele mai multe elemente care sunt valori autopomorfe. Numim număr autopomorfic o valoare care este egală cu unul dintre sufixele pătratului său. Exemplu de valoare autopomorfică este 25 deoarece $25^2 = 625$.

Matricea și numărul n de linii ale sale sunt transmise funcției prin intermediul a doi parametri.

În cadrul funcției L se va face apel la funcția A , care verifică dacă o valoare naturală transmisă ca parametru este autopomorfică. Ea returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului este sau nu număr autopomorfic.

Scrieți definițiile complete ale subprogramelor L și A .

Exemplu: Pentru $n=3$ și tabloul :

```
2 5 8
3 5 25
6 4 4
```

funcția L va returna valoarea 2.

Soluție

Funcția A verifică dacă o valoare naturală x este număr autopomorfic folosind o singură variabilă locală care este egală inițial cu pătratul lui x .

Ea este apelată de funcția L pentru fiecare element din matrice, contorizându-se în același timp numărul de valori corecte de pe fiecare linie.

Pentru varianta Pascal, considerăm definit tipul de date $mat=array[1..10,1..10]$ of integer.

```
1 function A(x:integer):boolean;
2 var y:integer;
3 begin
4   y := x * x; A:=true;
5   while x>0 do begin
6     if y mod 10<>x mod 10 then
7       begin
8         A:=false; exit;
9       end;
10    x:=x div 10; y:=y div 10;
11  end; end;
12 Function L(b:mat; n:byte):byte;
13 var i,j,m,mx:byte;
14 begin
15   mx := 0;
16   for i:=1 to n do begin
17     m := 0;
18     for j:=1 to n do
19       if A(b[i,j]) then inc(m);
20     if m > mx then begin
21       L := i; mx := m;
22     end
23   end;
24 end;
```

```
int A(int x)
{ int y;
  y = x * x;
  while (x>0){
    if (y % 10!=x % 10)
      return 0;
    x/=10;
    y/=10;
  }
  return 1;}

int L(int b[10][10],int n)
{ int l,i,j,m,mx;
  mx = 0;
  for (i=0;i<n;i++){
    m = 0;
    for (j=0;j<n;j++){
      if (A(b[i][j])) m++;
      if (m > mx){
        l = i; mx = m;
      }
    }
  }
  return l+1;
}
```

12. Funcția S primește, prin intermediul parametrului a , un vector de numere întregi cu 10 de elemente și, prin intermediul parametrului k , un număr natural nenul ($k \leq 10$). Funcția returnează suma tuturor elementelor pozitive $a[i]$ cu proprietatea că $k \leq i \leq 10$. Realizați un program care citește de la intrarea standard un șir de 10 de valori întregi și două numere naturale nenule x și y ($x < y \leq 10$) și afișează suma elementelor pozitive din șir cu numerele de ordine cuprinse între x și y , folosind apeluri la funcția S .

Exemplu: Pentru șirul 2, 5, 3, -7, -8, -9, 4, 2, 3, 3, $x=3$ și $y=8$ programul va afișa valoarea 9. ($3+4+2=9$)

Soluție:

Pentru a determina suma elementelor pozitive din șir cu numerele de ordine cuprinse între x și y (inclusiv x și y), se va apela funcția S pentru $k = x$ și $k = y + 1$. Programul va afișa diferența celor două valori returnate de funcție. Ambii parametri ai funcției realizează un transfer prin valoare.

<pre> 1 type 2 sir= array[1..10]of integer; 3 var i,x,y:integer; v:sir; 4 5 function S(a:sir;k:byte):word; 6 var t,i:integer; 7 begin 8 t:=0; 9 for i:=k to 10 do 10 if a[i]>0 then t:=t+a[i]; 11 S:=t; 12 end; 13 14 begin 15 for i:=1 to 10 do read(v[i]); 16 readln(x,y); 17 writeln(S(v,x)-S(v,y+1)); 18 end.</pre>	<pre> #include <iostream.h> int i,x,y, v[11]; int S(int a[11], int k) {int t,i; t=0; for (i=k;i<=10;i++) if (a[i]>0) t+=a[i]; return t; } void main() { for (i=1;i<=10;i++) cin>>v[i]; cin>>x>>y; cout<<(S(v,x)-S(v,y+1)); }</pre>
--	---

13. Realizați un program care să verifice dacă o valoare naturală x ($2 < x < 1000$), citită de la tastatură, reprezintă un număr perfect (egal cu suma divizorilor săi, strict mai mici decât el). Vor fi definite și apelate două subprograme:

- funcția D , care determină suma divizorilor unei valori transmise printr-un parametru întreg k , divizori strict mai mici decât k .
- funcția F , care determină numărul de valori perfecte dintr-un interval $[a, b]$ transmis prin intermediul a doi parametri întregi ($1 < a < b$).

Exemplu: Pentru $x=28$ programul va afișa mesajul “Da” ($28=1+2+4+7+14$), iar pentru $x=29$ se va afișa mesajul “Nu”.

Soluție:

Funcția D va fi apelată în cadrul subprogramului F , pentru fiecare din valorile întregi din intervalul $[a, b]$, contorizându-se numărul de valori perfecte.

Condiția ca numărul x să fie perfect este echivalentă cu expresia $F(2, x-1) \neq F(2, x)$. În cadrul ambelor funcții se efectuează doar transfer prin valoare.

<pre> 1 var x: word; 2 3 function D(k:integer):word; 4 var s,i:word; 5 begin 6 s:=0; 7 for i:=1 to k div 2 do 8 if k mod i=0 then s:=s+i; 9 D:=s; 10 end; 11 12 function F(a,b:integer):word; 13 var s,i:integer; 14 begin 15 s:=0; 16 for i:=a to b do 17 if D(i)=i then inc(s); 18 F:=s; 19 end; 20 21 begin 22 readln(x); 23 if F(2, x-1)<>F(2, x) then 24 write('DA') 25 else write('NU'); 26 end.</pre>	<pre> # include<iostream.h> int x; long int D(int k) {long int s,i; s=0; for (i=1;i<=k/2;i++) if (k%i==0) s+=i; return s; } int F(int a,int b) {int s,i; s=0; for (i=a;i<=b;i++) if (D(i)==i) s++; return s; } void main() { cin>>x; if (F(2, x-1)!=F(2, x)) cout<<"DA"; else cout<<"NU"; }</pre>
--	--

14. Considerăm un vector a cu n ($n < 100$) elemente numere naturale distincte (< 9999) și o valoare k naturală ($k < n$). Se dorește determinarea celui de-al k -lea element din vectorul ordonat. Subprogramul P determină acest element fără a sorta efectiv elementele vectorului. Acesta are ca parametri pe a , n și k , iar valoarea determinată o returnează prin intermediul unui parametru întreg x .

În cadrul subprogramului se face apel la funcția Nr care determină câte elemente dintr-un vector sunt mai mici decât o valoare dată. Vectorul, numărul de elemente ale acestuia și valoarea respectivă sunt transmise prin parametri.

Scrieți definițiile complete ale subprogramelor P și Nr .

Exemplu: Pentru tabloul ce conține 5 elemente: (220, 45, 5125, 102, 53) și $k=3$, se va determina valoarea 102.

Soluție:

În varianta Pascal, subprogramul P va fi implementat ca procedură, deoarece rezultatul este returnat printr-un parametru. Din același motiv, în C++, funcția P are tipul *void*.

Subprogramul va identifica prin apeluri la funcția Nr , elementul din vector pentru care se regăsesc $k-1$ elemente de valori strict mai mici decât el.

<pre> 1 function Nr(a:sir; 2 n,x:integer):byte; 3 var t,i:integer; 4 begin 5 t:=0; 6 for i:=1 to n do 7 if a[i]<x then inc(t); 8 Nr:=t; 9 end; 10 11 procedure P(a:sir;n,k:integer; 12 var x:integer); 13 var i:integer; 14 begin 15 i:=1; 16 while (Nr(a,n,a[i])<>k-1) and 17 (i<=n) do inc(i); 18 x:=a[i]; 19 end; </pre>	<pre> int Nr(int a[100],int n,int x) { int t,i; t=0; for (i=0;i<n;i++) if (a[i]<x) t++; return t; } void P(int a[100],int n, int k,int &x) { int i; i=0; while ((Nr(a,n,a[i])!=k-1)&& (i<=n)) i++; x=a[i]; } </pre>
---	---

15. Realizați un program care descompune un număr natural n ($3 \leq n \leq 250$), în sumă de minimum doi termeni distincți ai șirului lui Fibonacci ($f_1=1$, $f_2=1$, $f_3=2$, $f_n=f_{n-1}+f_{n-2}$).

În cadrul programului va fi definită o funcție *Fib* care determină cel mai mare termen din șirul Fibonacci, strict mai mic decât o valoare naturală, transmisă prin intermediul unui parametru.

Exemplu: Pentru $n=21$ se va afișa $13+5+2+1$, iar pentru $n=150$ se va afișa $144+5+1$

Soluție:

Funcția *Fib* va avea un singur parametru, transmis prin valoare, și va returna o valoare dintr-un tip întreg. Ea va fi apelată inițial pentru valoarea n , apoi pentru diferența dintre n și valoarea returnată anterior, ș.a.m.d. Toate valorile obținute în urma apelurilor la *Fib*, vor reprezenta termeni din scrierea lui n ca sumă de elemente ale șirului lui Fibonacci.

<pre> 1 var n,x:byte; 2 3 function fib(n:byte):byte; 4 var a,b,c:byte; 5 begin 6 a:=1;b:=1; 7 while a+b<n do begin 8 c:=a+b; 9 a:=b; 10 b:=c; 11 end; 12 fib:=b; 13 end; 14 </pre>	<pre> #include <iostream.h> int n,x; int fib(int n) {int a,b,c; a=1; b=1; while (a+b<n) { c=a+b; a=b; b=c; } return b; } </pre>
--	---

```

15 begin
16   readln(n);
17   while n>2 do begin
18     x:=fib(n);
19     write(x, ' ');
20     n:=n-x;
21   end;
22   writeln(n)
23 end.

```

```

void main()
{cin >>n;
  while (n>2){
    x=fib(n);
    cout<<x<<' ';
    n=n-x;
  }
  cout<<n<<' ';
}

```

16. Se consideră un șir de n numere scrise în baza 16. Să se realizeze un program care determină cel mai mare divizor comun al acestor numere, afișându-l în reprezentarea din baza 10. În cadrul programului, vor fi definite și apelate două subprograme:

- funcția *Conv*, care primește un șir de caractere reprezentând un număr exprimat în baza 16 și returnează numărul obținut în urma conversiei acestuia în baza 10.
- funcția *Cmmdc*, care returnează cel mai mare divizor comun a două valori naturale transmise prin intermediul a doi parametri.

Cifrele din baza 16 sunt reprezentate prin cifrele zecimale 0,...,9 și majusculile A,B,C,D,E,F (corespunzătoare resturilor de la 10 la 15).

Exemplu: Pentru $n=3$ și numerele 1A, 27 și 41 afișează valoarea 13.

Soluție:

Ambele funcții vor realiza transfer de date prin intermediul unor parametri valoare. Funcția *Conv* va fi apelată pentru fiecare număr hexazecimal. Valorile returnate de ea vor constitui parametri actuali(efectivi) în cadrul apelurilor la funcția *Cmmdc*.

```

1  var s:string; n,i,a,b:integer;
2
3  function Conv(s:string):integer;
4  var i, nr:integer;
5  begin
6    nr:=0;
7    for i:=1 to length(s) do
8      if s[i] in ['0'..'9'] then
9        nr:=nr*16 + ord(s[i])-48
10     else
11       nr:=nr*16 + ord(s[i])-55;
12     Conv:=nr;
13   end;
14
15   function Cmmdc(a,b:integer):integer;
16   begin
17     while a<>b do
18       if a>b then a:=a-b
19         else b:=b-a;
20     cmmdc:=a;
21   end;
22

```

```

#include <iostream.h>
#include <string.h>
char s[100]; int n,i,a,b;

int Conv(char s[100])
{int i, nr=0;
 for (i=0;i<strlen(s);i++)
  if (s[i]>='0' && s[i]<='9')
    nr=nr*16 + s[i]-48;
  else
    nr=nr*16 + s[i]-55;
  return nr;
}

int Cmmdc(int a,int b)
{
  while (a!=b)
    if (a>b) a=a-b;
    else b=b-a;
  return a;
}

```

```

23 begin
24   readln(n); readln(s);
25   a:=Conv(s);
26   for i:=2 to n do begin
27     readln(s); b:=Conv(s);
28     a:=Cmmdc(a,b);
29   end;
30   writeln(a);
31 end.

```

```

void main(){
  cin>>n; cin>>s;
  a=Conv(s);
  for (i=2; i<=n ;i++){
    cin >>s; b=Conv(s);
    a=Cmmdc(a,b);
  }
  cout<<a ;
}

```

17. Considerăm o matrice pătratică a cu n linii și n coloane ($n \leq 100$). Se dorește ordonarea elementelor diagonalei principale prin interschimbări de linii și coloane. Scrieți definițiile următoarelor subprograme:

- subprogramul *interl*, care interschimbă elementele a două linii, ale căror indici sunt transmiși ca parametri.
- subprogramul *interc*, care interschimbă elementele a două coloane, ale căror indici sunt transmiși ca parametri.
- subprogramul *S* care ordonează elementele diagonalei secundare prin apeluri la *interl* și *interc*.

Toate cele trei subprograme vor avea matricea a și numărul n de linii transmise prin parametri.

Exemplu: Pentru $n=3$ și tabloul

```

3 2 1
2 1 3
3 1 2

```

se va afișa tabloul

```

1 3 2
1 2 3
2 1 3

```

Soluție:

În varianta Pascal, toate subprogramele vor fi implementate ca proceduri având ca parametru referință tabloul bidimensional. Din același motiv, în C++ toate cele trei funcții au tipul *void*, numele tabloului transmis ca parametru fiind el însuși un pointer.

Pentru varianta Pascal considerăm definit tipul de date *mat=array[1..100,1..100] of byte*.

```

1 procedure interl(var a:mat;
2                   n,x,y:byte);
3 var i,t:byte;
4 begin
5   for i:=1 to n do begin
6     t:=a[x,i];
7     a[x,i]:=a[y,i];
8     a[y,i]:=t;
9   end;
10 end;
11
12 procedure interc(var a:mat;
13                  n,x,y:byte);
14 var i,t:byte;

```

```

void interl(int a[100][100],
            int n,int x,int y)
{
  int i,t;
  for (i=0;i<n;i++){
    t=a[x][i];
    a[x][i]=a[y][i];
    a[y][i]=t;
  }
}

void interc(int a[100][100],
            int n,int x,int y)
{ int i,t;

```


<pre> 15 begin 16 for i:=1 to n do begin 17 t:=a[i,x]; a[i,x]:=a[i,y]; 18 a[i,y]:=t; 19 end; 20 end; 21 22 procedure S(var a:mat;n:byte); 23 begin 24 for i:=1 to n-1 do 25 for j:=i+1 to n do 26 if a[i,i]>a[j,j] then begin 27 interl(a,n,i,j); 28 interc(a,n,i,j); 29 end; 30 end; </pre>	<pre> for (i=0;i<n;i++){ t=a[i][x]; a[i][x]=a[i][y]; a[i][y]=t; } } void S(int a[100][100],int n) { for (i=0;i<n-1;i++) for (j=i+1;j<n;j++) if (a[i][i]>a[j][j]) { interl(a,n,i,j); interc(a,n,i,j); } } </pre>
---	--

18. Considerăm o funcție S care returnează suma tuturor cifrelor a două valori naturale transmise prin intermediul a doi parametri.

Se dorește determinarea unei perechi de elemente distincte, ale unui tablou unidimensional care au proprietatea că suma *tuturor* cifrelor lor este maximă.

Vectorul se consideră a avea maximum 100 de elemente pozitive cu valori mai mici decât 2.000.000.000.

Subprogramul P , primește prin doi parametri, vectorul și numărul lui de elemente și returnează, prin intermediul altor doi parametri, elementele tabloului ce au proprietatea cerută. Acesta va face apel la funcția S .

Scrieți definițiile celor două subprograme.

Exemplu: Pentru $n=5$ și tabloul (86, 15, 13, 86, 112), subprogramul P va returna valorile 86 și 15 ($8+6+1+5=20$ =suma maximă).

Soluție:

În varianta Pascal, subprogramul P va fi implementat ca procedură, având doi parametri referință întregi. Aceștia vor returna elementele tabloului, pentru care suma tuturor cifrelor este maximă. Din același motiv, în C++ funcția P are tipul *void*.

Tabloul și numărul de elemente ale acestuia vor reprezenta, pentru subprogramul P , parametri valoare. Pentru varianta Pascal considerăm definit tipul de date *sir=array[1..100] of longint*.

<pre> 1 function S(x,y:longint):byte; 2 var su:byte; 3 begin 4 su:=0; 5 while (x>0)or(y>0)do begin 6 su:=su+x mod 10+y mod 10; 7 x:=x div 10; y:=y div 10; 8 end; 9 S:=su; 10 end; </pre>	<pre> int S(long x,long y) { int su; su=0; while (x>0 y>0){ su += x % 10 + y % 10; x /= 10; y /= 10; } return su; } </pre>
---	---

```

11 procedure P(a:sir;n:byte;
12             var x,y:longint);
13 var max,i,j:byte;
14 begin
15   max:=0;
16   for i:=1 to n-1 do
17     for j:=i+1 to n do
18       if (max<S(a[i],a[j]))and
19         (a[i]<>a[j]) then begin
20         max:=S(a[i],a[j]);
21         x:=a[i]; y:=a[j];
22       end;
23   end;

```

```

void P(long a[100],int n,
      long & x,long & y)
{int max,i,j;
  max=0;
  for (i=0;i<n-1;i++)
    for (j=i+1;j<n;j++)
      if (max<S(a[i],a[j])
        && a[i]!=a[j])
      {
        max=S(a[i],a[j]);
        x=a[i]; y=a[j];
      }
}

```

19. Fie un tablou unidimensional cu n elemente întregi. Se cere să se realizeze un subprogram S care ordonează descrescător elementele vectorului, după numărul de cifre distincte pe care le conțin. În cazul elementelor cu același număr de cifre distincte, ordonarea se va face descrescător după valorile lor. Subprogramul va apela funcția Nr care determină numărul de cifre distincte ale unui număr întreg primit prin intermediul unui parametru.

Exemplu: Primind $n=7$ și tabloul (334, 124, 21, 34, 222, 1, 9) subprogramul S va transmite, în urma execuției, tabloul (124, 334, 34, 21, 222, 9, 1).

Soluție:

În varianta Pascal, subprogramul S va fi implementat ca procedură, având ca parametru referință tabloul unidimensional și ca parametru valoare numărul de elemente ale acestuia. Din același motiv, în C++ funcția S are tipul *void*, numele tabloului transmis ca parametru fiind el însuși un pointer.

Pentru varianta Pascal considerăm definit tipul de date *sir=array[1..100] of integer*.

```

1 function Nr(x:integer):byte;
2 var m:byte; c:set of byte;
3 begin
4   m:=0;
5   c:=[];
6   while (x>0) do begin
7     if not (x mod 10 in c) then
8       begin
9         inc(m);
10        c:=c+[x mod 10];
11      end;
12    x:=x div 10;
13  end;
14  Nr:=m;
15 end;
16
17 procedure S(var a:sir;n:byte);
18 var i,j:byte;x:integer;
19 begin

```

```

int Nr(int y)
{
  int x,i,m,c;
  m=0;
  for (i=0;i<10;i++){
    c=0;
    x=y;
    while (x>0) {
      if (x % 10 == i) c=1;
      x /= 10;
    }
    if (c) m++;
  }
  return m;
}

void S (int a[100], int n)
{
  int i,j,x;

```

```

20  for i:=1 to n-1 do
21    for j:=i+1 to n do
22      if (Nr(a[i])<Nr(a[j]))or
23        ((Nr(a[i])=Nr(a[j])) and
24          (a[i]<a[j]))
25      then begin
26        x:=a[i]; a[i]:=a[j];
27        a[j]:=x;
28      end;
29  end;

```

```

for (i=0;i<n-1;i++)
  for (j=i+1;j<n;j++)
    if (Nr(a[i])<Nr(a[j]) ||
      Nr(a[i])==Nr(a[j])&&
      a[i]<a[j])
    {
      x=a[i]; a[i]=a[j];a[j]=x;
    }
}

```

20. Se consideră un vector ce conține n elemente întregi. În fața oricărui element precedat de un element de semn contrar se înserează un element pozitiv, a cărui valoare este obținută prin alipirea cifrelor celor două numere de semne contrare, în ordine. Să se scrie definițiile următoarelor trei subprograme:

- funcția L , care returnează numărul natural obținut prin alipirea, în ordine, a cifrelor a două valori primite prin intermediul unor parametri întregi.
- subprogramul Ins , care permite inserarea într-un vector, pe o poziție dată, a unei valori date. Vectorul, indicele pe care urmează să se efectueze inserarea și valoarea care se va insera sunt transmise subprogramului prin intermediul parametrelor.
- subprogramul Rez , care în urma apelurilor la subprogramele Ins și A , efectuează prelucrarea cerută prin enunț, asupra unui vector pe care îl primește pînă-un parametru.

Exemplu: După execuția subprogramului Rez , tabloul (3,-1, 73, 5, -9, 2) va conține elementele (3, 31,-1, 173, 73, 5, 59, -9, 92, 2).

Soluție :

În varianta Pascal, subprogramele Ins și Rez vor fi implementate ca proceduri, ambele având transmiși, prin parametri referință, tabloul unidimensional și lungimea acestuia. În C++, funcțiile Ins și Rez au tipul *void*, transferul tabloului și a lungimii acestuia realizându-se tot prin parametri referință.

Funcția L va fi apelată pentru oricare pereche de elemente consecutive de semne contrare. Valorile returnate de ea, vor constitui parametri actuali la apelul subprogramului Ins .

Pentru varianta Pascal, considerăm definit tipul de date $sir=array[1..100]$ of *integer*.

```

1  function L(x,y:integer):integer;
2  var z:integer;
3  begin
4    z:=y;
5    while (z>0) do begin
6      x:=x*10;
7      z:=z div 10;
8    end;
9    L:=x+y;
10 end;

```

```

int L(int x,int y)
{
  int z;
  z=y;
  while (z>0) {
    x *=10;
    z/=10;
  }
  return x+y;
}

```

```

11 procedure Ins(var a:sir;
12     var n:byte; x,p:integer);
13 var i,j:byte;
14 begin
15     for i:=n downto p do
16         a[i+1]:=a[i];
17     a[p]:=x; inc(n)
18 end;
19
20 procedure Rez(var a:sir;
21     var n:byte);
22 var x,y:integer;
23 begin
24     i:=1;
25     while i<n do
26         if a[i]*a[i+1]<0 then begin
27             x:=abs(a[i]);y:=abs(a[i+1]);
28             Ins(a,n,L(x,y),i+1);
29             inc(i,2);
30         end
31         else inc(i)
32     end;

```

```

void Ins(int a[100],
        int & n,int x,int p)
{
    int i,j;
    for (i=n-1;i>=p;i--)
        a[i+1]=a[i];
    a[p]=x; n++;
}

void Rez(int a[100], int &n)
{
    int x,y;
    i=0;
    while (i<n-1)
        if (a[i]*a[i+1]<0){
            x=abs(a[i]);y=abs(a[i+1]);
            Ins(a,n,L(x,y),i+1);
            i += 2;
        }
        else i++;
}

```

2.1.3 Probleme propuse

1. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere naturale. Se cere realizarea unui program care înlocuiește fiecare element din vector cu cel mai apropiat număr prim față de acesta. Programul va conține următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- funcția *Prim*, care verifică dacă valoare primită printr-un parametru întreg este număr prim. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- Funcția *Det*, care va determina cel mai apropiat număr prim față de o valoare întreagă primită printr-un parametru, folosindu-se de apeluri la funcția *Prim*. Rezultatul funcției *Det* va fi de tip întreg.
- subprogramul *Scrie*, care permite afișarea elementelor unui vector. Tabloul și numărul de elemente ale acestuia sunt primite de subprogram prin intermediul a doi parametri.

Exemplu: Pentru $n=4$ și tabloul $a=(5, 16, 33, 24)$ se va afișa: 5 17 31 23

2. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere întregi. Să se realizeze un program care ordonează crescător elementele situate în prima jumătate a vectorului și descrescător pe cele situate în a doua jumătate. Programul va conține următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- subprogramul *Sortare*, care ordonează la alegere elementele unui vector, între doi indici transmiși prin intermediul a doi parametri întregi. (vezi problema 5, secțiunea 2.1.2). Vectorul va fi transmis subprogramului printr-un parametru.
- subprogramul *Scrie*, care permite afișarea elementelor unui vector. Tabloul și numărul de elemente ale acestuia sunt primite de subprogram prin intermediul a doi parametri.

Exemplu: Pentru $n=5$ și tabloul $a=(153, 16, 8, 33, 124)$ se va afișa:
16 153 124 33 8

3. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere întregi. Să se realizeze un program care identifică un subtablou a lui a , format din k elemente, subtablou care conține cele mai multe valori cuburi perfecte.

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor variabilelor n , k și a elementelor vectorului a .
- funcția *Cub*, care verifică dacă valoare primită printr-un parametru întreg este cub perfect. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- funcția *Poz*, care determină poziția (indicele) de început a subtabloului de k elemente care conține cele mai multe cuburi perfecte. Funcția va face apel la subprogramul *Cub*. Vectorul, numărul de elemente ale acestuia și valoarea lui k vor fi primite de subprogram prin intermediul parametrilor.
- subprogramul *Scrie*, care permite afișarea unei secvențe de k elemente, începând cu o poziție p , dintr-un vector transmis prin parametru. Valorile lui k și p vor fi primite de subprogram prin intermediul a doi parametri întregi.

Exemplu: Pentru $n=7$, $k=4$ și tabloul $a=(1, 9, 4, 27, 8, 1, 12)$ se va afișa: 4, 27, 8, 1

4. Se consideră două numere naturale, n și b ($n \leq 100$, $b \leq 10$). Să se construiască un tablou unidimensional ale cărui elemente vor fi primele n numere naturale, care au proprietatea că în baza b se reprezintă numai cu cifre de 0 și 1.

În cadrul programului, vor fi definite următoarele subprograme:

- funcția *Baza*, care verifică dacă valoare primită printr-un parametru întreg se reprezintă într-o bază b doar prin cifre de 0 și 1. Baza b va fi primită tot prin intermediul unui parametru întreg. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- subprogramul *Det*, care construiește un vector cu n elemente. Acestea reprezintă primele n valori naturale ce au proprietatea că în baza b sunt scrise numai cu cifre de 0 și 1. Tabloul, valoarea lui n și a lui b sunt transmise subprogramului prin intermediul parametrilor.

- subprogramul *Scrie*, care permite afișarea elementelor unui vector. Tabloul și numărul de elemente ale acestuia sunt primite de subprogram prin intermediul a doi parametri.

Exemplu: Pentru $n=7$ și $b=3$ tabloul va conține valorile 0, 1, 3, 4, 9, 10, 12.

5. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere naturale, mai mici ca 2.000.000.000. Să se realizeze un program care afișează pe linii, în fișierului *Nr.txt*, elementele din vectorul a , grupate după cifra dominantă (prima în scrierea zecimală). O linie va conține doar elemente cu aceeași cifră dominantă. Scrierea în fișier se va face în ordinea crescătoare a primei cifre.

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- funcția *Cif*, care determină cifra dominantă a unei valori primite printr-un parametru întreg.
- subprogramul *Scrie*, care afișează pe linii, în fișierul *Nr.txt*, elementele din a , după regula prezentată în enunț. Se vor folosi apeluri la funcția *Cif*. Tabloul și numărul de elemente ale acestuia sunt transmise subprogramului prin intermediul parametrilor.

Exemplu: Pentru $n=7$ și tabloul $a=(334, 124, 71, 34, 122, 1, 39)$ fișierul *Nr.txt* va conține:

```
124 122 1
334 34 39
71
```

6. Se consideră două tablouri unidimensionale a și b ce conțin n , respectiv m elemente numere întregi ($m \leq n \leq 50$). Să se realizeze un program care determină de câte ori toate elementele lui b apar pe poziții consecutive în tabloul a .

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a numărului de elemente ale unui vector și a elementelor acestuia. Transferul datelor citite va fi realizat prin parametri.
- funcția *Ok*, care verifică dacă elementele unui vector b se regăsesc în vectorul a , pe poziții consecutive, începând de la un indice dat. Funcția va avea 5 parametri: cele două tablouri, numărul de elemente ale fiecăruia și indicele începând cu care se face verificarea. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- funcția *Nr*, care determină numărul de apariții cerut în enunț. Cei doi vectori și numărul de elemente ale fiecăruia sunt transmise subprogramului prin intermediul parametrilor. Se vor folosi apeluri la funcția *Ok*.

Exemplu: Pentru $n=7$, $m=3$ și tablourile $a=(33, 3, 71, 3, 71, 3, 39)$, $b=(3, 71, 3)$, funcția *Nr* va returna valoarea 2.

7. Se consideră un tablou unidimensional a cu $n(<100)$ elemente întregi. Să se determine toate secvențele de elemente situate pe poziții consecutive care au suma egală cu S . Fiecare secvență de elemente va fi afișată pe câte o linie pe ieșirea standard.

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor variabilelor n , S și ale elementelor vectorului a .
- funcția *Sum*, care determină suma elementelor unui vector, situate între două poziții date. Tabloul și cei doi indici vor fi transmiși subprogramului prin intermediul parametrilor.
- subprogramul *Scrie*, care permite afișarea tuturor secvențelor de elemente dintr-un tablou care respectă cerința din enunț. Vectorul și numărul de elemente ale acestuia vor fi primiți de subprogram prin intermediul parametrilor.

Exemplu: Pentru $n=7$, $S=9$ și vectorul (3, 2, 3, 4, 5, 11, -7) se va afișa:

2 3 4

4 5

5 11 -7

8. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere naturale, mai mici ca 20.000. Să se realizeze un program care determină un subtablou al său, cu proprietatea că suma elementelor este divizibilă cu n .

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- funcția *Rest*, care determină restul la împărțirea cu n a sumei primelor k elemente ale unui vector. Tabloul, n și k vor fi trimise subprogramului prin intermediul parametrilor.
- subprogramul *Det*, determină doi indici din vectorul a , care delimitează subtabloul cu proprietatea prezentată în enunț. Se vor folosi apeluri la funcția *Rest*. Subprogramul va primi, prin intermediul a doi parametri, vectorul și numărul de elemente ale acestuia și va returna, prin intermediul altor doi parametri întregi, cei doi indici determinați.
- subprogramul *Scrie*, care permite afișarea unei secvențe de elemente dintr-un tablou, situate între două poziții date. Vectorul și cei doi indici vor fi primiți de subprogram prin intermediul parametrilor.

Exemplu: Pentru $n=7$ și tabloul $a=(4, 5, 1, 3, 2, 0, 9)$, se va afișa 3 2 0 9.

9. Se consideră două mulțimi reținute în doi vectori. Să se realizeze un program care determină reuniunea, intersecția și diferența lor. Pentru fiecare dintre operații se cere definirea unui subprogram. Acestea vor apela funcția *Ap* care verifică dacă o valoare dată aparține unei mulțimi. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++. Transferul datelor între subprograme va fi realizat prin intermediul parametrilor.

Exemplu : $A=(2, 4, 1, 6, 7)$, $B=(3, 4, 8, 9)$ se va afișa:

Reuniunea= $(2, 4, 1, 6, 7, 3, 8, 9)$; Intersectia= (4) ; Diferenta= $(2, 1, 6, 7)$

10. Considerăm un vector a ce reprezintă o permutare a mulțimii numerelor de la 1 la n ($n \leq 50$). Se cere determine toate permutările circulare ale lui a care nu conțin puncte fixe.

Spunem că o permutare $(a_1, a_2 \dots a_n)$ are puncte fixe dacă există cel puțin un element $a_i = i$ ($1 \leq i \leq n$). De exemplu, pentru $n=4$, permutarea $(2,4,3,1)$ are puncte fixe, deoarece $a_3=3$.

În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- funcția *Pfix*, care verifică dacă un vector, ce reprezintă o permutare, are puncte fixe. Vectorul și lungimea acestuia vor fi primite prin intermediul parametrilor. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- subprogramul *Perm*, care va genera o permutare circulară cu o poziție spre stânga a unui vector primit printr-un parametru.
- subprogramul *Scrie*, care permite afișarea permutărilor circulare ale unui vector, permutări ce nu conțin puncte fixe. Acest subprogram va face apel la *Pfix* și *Perm*. Tabloul și numărul de elemente ale acestuia sunt primite de subprogramul *Scrie* prin intermediul a doi parametri.

Exemplu: Pentru $n=4$ și tabloul $a=(5, 1, 3, 2, 4)$ se va afișa:

2 4 5 1 3

4 5 1 3 2

11. Se consideră un tablou bidimensional pătratic cu n linii ($n \leq 10$). Să se determine elementele care sunt situate pe linii și coloane de sumă egală. Un element $a[i,j]$ va fi afișat dacă suma pe linia i este egală cu suma pe coloana j .

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor tabloului a .
- funcția *SumL*, care calculează suma elementelor dintr-o matrice, situate pe o linie, al cărui număr de ordine este transmis printr-un parametru întreg.
- funcția *SumC*, care calculează suma elementelor dintr-o matrice, situate pe o coloană, al cărui număr de ordine este transmis printr-un parametru întreg.
- subprogramul *Scrie*, care permite afișarea tuturor elementelor dintr-o matrice ce respectă cerința din enunț. Tabloul și numărul de linii/coloane, sunt primite de subprogram prin intermediul parametrilor.

Exemplu: Pentru $n=4$ și tabloul

0 5 0 0

3 7 2 4

-20 6 20 10

0 -2 0 0

se vor afișa elementele 7 și 6.

12. Se consideră un tablou bidimensional pătratic cu n linii ($n \leq 20$). Să se determine toate elementele ce reprezintă puncte 'șa' (element minim pe linie și maxim pe coloana pe care este situat).

În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor tabloului a .
- funcția *MinL*, care determină cea mai mică valoare situată în matricea a pe o linie, al cărui număr de ordine este transmis printr-un parametru întreg.
- funcția *MaxC*, care primește două valori întregi, prin intermediul a doi parametri x și c . Ea verifică dacă x reprezintă maximum pe coloana c a matricii a . Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- subprogramul *Scrive*, care permite afișarea tuturor punctelor „șa” dintr-o matrice. Tabloul și numărul de linii/coloane, sunt primite de subprogram prin intermediul parametrilor.

Exemplu: Pentru $n=4$ și tabloul

9 8 6 7

8 3 5 4

9 9 6 7

5 1 4 3

se vor afișa elementele 6 și 6.

13. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente întregi. Să se identifice numerele de ordine ale coloanelor care conțin cele mai multe elemente în afara intervalului $[x, y]$.

În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n și m și ale elementelor tabloului a .
- funcția *Nr*, care primește trei valori întregi prin intermediul parametrilor x , y și c . Ea determină câte elemente situate pe coloana c , în matricea a , se află în afara intervalului $[x, y]$.
- subprogramul *Det*, care afișează indicii coloanelor cu proprietatea cerută în enunț. În cadrul acestuia se va face apel la funcția *Nr*.

Exemplu: Pentru $n=4$, $m=5$, $x=2$, $y=7$ și tabloul

2 8 5 8 3

4 3 6 7 5

8 8 8 8 8

9 9 9 9 9

se va afișa 2 4 (coloana a doua și a patra).

14. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$) cu componente întregi. Folosind interschimbări de coloane, să se ordoneze crescător elementele pare situate pe ultima linie din tabloul a .

În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n și m și ale elementelor tabloului a .
- subprogramul *InterC*, care primește două valori întregi prin intermediul parametrilor x, y . Acesta interchimbă elementele coloanei x cu cele situate pe coloana y în matricea a .
- subprogramul *Sortare*, care efectuează ordonarea crescătoare a elementelor pare situate pe ultima linie în matricea a folosind apeluri la *InterC*.
- subprogramul *Scrie*, care efectuează afișarea pe ecran a elementelor tabloului bidimensional a .

Exemplu: Pentru $n=3, m=4$ și tabloul a :

1 2 3 4	se va afișa:
1 2 3 4	1 4 2 3
1 2 3 4	1 4 2 3
1 4 6 2	1 2 4 6

15. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente întregi. Să se identifice numerele de ordine ale liniilor care conțin elemente ordonate crescător și, în plus, diferența dintre oricare două elemente alăturate, este constantă în cadrul liniei. În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale elementelor tabloului a .
- funcția *Ok*, care primește o valoare naturală prin intermediul parametrului l și verifică dacă elementele liniei de indice l , din matricea a , respectă regula din enunț. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- subprogramul *Scrie*, care efectuează afișarea pe ecran a numerelor de ordine determinate.

Exemplu: Pentru $n=4$ și tabloul a :

```
1 3 5 7
4 3 2 1
2 4 6 8
9 9 9 8
```

se va afișa 1 3 (prima linie și a treia).

16. Se consideră un tablou bidimensional pătratic a cu n linii și n coloane ($n \leq 50$) cu componente întregi. Să se permute circular cu k poziții, spre dreapta, toate liniile tabloului. În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui k și ale elementelor tabloului a .
- subprogramul *Perm*, care primește o valoare naturală prin intermediul parametrului l și permută toate elementele liniei de indice l , cu o poziție spre dreapta.

- subprogramul *Solve*, care permută cu k poziții spre dreapta, toate liniile unei matrici pătratică, prin apeluri la *Perm*.
- subprogramul *Scrie*, care efectuează afișarea pe ecran a numerelor de ordine ale liniilor lui a care au proprietatea impusă prin enunț.

Tabloul bidimensional pătratic și numărul de linii ale acestuia sunt transmise subprogramelor prin intermediul a doi parametri.

Exemplu: Pentru $n=4$, $k=2$ și tabloul a :

1 2 3 4	3 4 1 2
1 2 3 4	3 4 1 2
1 4 6 2	6 2 1 4
2 4 5 3	5 3 2 4

17. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente numere naturale ≤ 60000 . Să se afișeze, pentru fiecare linie, numărul de zerouri în care se termină produsul elementelor lor. În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale elementelor tabloului a .
- funcția *Exp*, care returnează exponentul la care apare un număr prim în descompunerea unui număr natural. Ambele valori sunt primite de subprogram prin intermediul a doi parametri întregi.
- funcția *Nrz*, care primește o valoare naturală prin intermediul parametrului l și determină numărul de zerouri în care se termină produsului elementelor situate pe linia l , în matricea a . Se vor folosi apeluri la funcția *Exp*.
- subprogramul *Scrie*, care efectuează afișarea pe ecran, pentru fiecare linie, a numărului de zerouri în care se termină produsul elementelor lor. Se vor folosi apeluri la funcția *Nrz*.

Exemplu: Pentru $n=4$, $m=3$ și tabloul a

2 8 5
4 3 6
10 10 10
25 25 4

se va afișa 1, 0, 3, 2.

18. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente cifre zecimale (0..9). Considerăm că fiecare linie reprezintă cifrele unui număr. Se cere identificarea celei mai mici baze comune tuturor numerelor reprezentate în matrice și afișarea acestora în urma conversiei din baza minimă determinată, în baza 10.

În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale elementelor tabloului a .
- funcția *Baza*, care determină cea mai mică bază comună tuturor numerelor reprezentate în matricea a .

- funcția *Conv*, care primește două valori naturale prin intermediul parametrilor b și l . Ea determină valoarea din baza 10 obținută în urma conversiei numărului reprezentat în baza b pe linia l .
- subprogramul *Scrie*, care efectuează afișarea pe ecran, pentru fiecare linie, a numărului returnat în urma apelului la funcția *Conv*.

Exemplu: Pentru $n=3$, $m=3$ și tabloul a

2 1 5

4 3 6

1 0 1

se va afișa 110, 223, 50 (baza minimă 7).

19. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente cifre zecimale (0..9). De la tastatură este introdus un șir de $m-1$ caractere $+$ și $-$, având semnificația operatorilor aritmetici cunoscuți. Ele vor fi "plasate" în ordine între valorile de pe fiecare linie. Se cere să se determine care este valoarea maximă care se obține în urma evaluării expresiilor obținute în cadrul fiecărei linii.

În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale elementelor tabloului a .
- funcția *Eval*, care primește o valoare naturală printr-un parametru l și un șir de caractere printr-un parametru s . Aceasta evaluează expresia obținută prin plasarea în ordine a operatorilor $+$ și $-$ din care este format s , între elementele situate pe linia l a matricei a .
- funcția *Max*, care determină valoarea maximă care se obține în urma evaluării expresiilor obținute în cadrul fiecărei linii, prin apeluri la funcția *Eval*.

Exemplu: Pentru $n=3$, $m=4$, $s=,+-$ și tabloul a

2 1 5 2

4 3 6 3

1 0 1 2 se va afișa 10 ($4+3+6-3$)

20. Se consideră un număr natural n ($n \leq 100$). Să se creeze o matrice ale cărei elemente sunt numerele de la 1 la n^2 . Valorile sunt plasate în matrice, consecutiv în cadrul unei linii, dar alternând monotonia, după cum reiese și din tabloul următor:

1 2 3 4

8 7 6 5

9 10 11 12

16 15 14 13

În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *linieC*, care primește două valori naturale prin intermediul parametrilor x și l . Acesta plasează, crescător, valori consecutive pe linia l începând cu valoarea x .

- subprogramul *linieD*, care primește două valori naturale prin intermediul parametrilor x și l . Acesta plasează, descrescător, valori consecutive pe linia l începând cu valoarea x .
- subprogramul *Solve*, care construiește o matrice pătratică de ordin n , ale cărei elemente respectă regula din enunț. Acesta va face apel la subprogramele *linieC* și *linieD*. Matricea și numărul n de linii este transmis subprogramului *Solve* prin intermediul a doi parametri.
- subprogramul *Scrie*, care efectuează afișarea pe ecran elementelor unui tablou bidimensional pătratic transmis printr-un parametru. Numărul de linii al său este transmis prin al doilea parametru.

21. Se consideră un șir de cel mult 100 de caractere, format din litere mici ale alfabetului englez. Se numește bâlăbă o secvență de caractere care apare în șir de cel puțin două ori una după alta. De exemplu șirul "abddcabddabcabc" conține bâlbele "d" și "abc".

a) Să se realizeze funcția *MaxB*, care determină cea mai lungă "bâlăbă" dintr-un șir de caractere primit printr-un parametru.

b) Să se realizeze funcția *Cod*, care codifică un șir de caractere primit ca parametru. Regula de codificare este următoarea: fiecare caracter este înlocuit cu ultima cifră a numărului 2^x , unde x este codul ASCII asociat literei respective. Rezultatul returnat de funcție este un șir de caractere.

Exemplu: Pentru șirul "abbcabab", funcția *MaxB* va returna șirul "ab", iar funcția *Cod* va returna șirul de caractere "24482424"

22. Considerăm un vector a , care conține n elemente numere reale ($n \leq 100$). Se dorește construirea unui vector b cu n elemente numere întregi, creat pe baza elementelor lui a după următoarea regulă: dacă elementul $a[i]$ are partea întreagă un număr par, atunci $b[i]$ va reprezenta cel mai apropiat întreg de $a[i]$ divizibil cu 10; în caz contrar, $b[i]$ va fi egal cu cel mai apropiat întreg de $a[i]$ divizibil cu 100. În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor reale ale vectorului a .
- funcția *Div10*, care rotunjește o valoare reală, primită printr-un parametru, la cel mai apropiat întreg divizibil cu 10.
- funcția *Div100*, care rotunjește o valoare reală, primită printr-un parametru, la cel mai apropiat întreg divizibil cu 100.
- subprogramul *Solve*, care primind prin parametrul a un vector cu elemente reale, returnează printr-un alt parametru b , un vector ale cărui elemente întregi respectă regula din enunț. Se vor folosi apeluri la funcțiile *Div10* și *Div100*.
- subprogramul *Scrie* care permite afișarea elementelor întregi ale unui vector. Tabloul și numărul de elemente ale acestuia sunt primite de subprogram prin intermediul a doi parametri.

Exemplu: Pentru $n=3$ și $a=(1327.3, 234.67, 487.34)$, se va afișa 1300, 230, 500.

23. Se consideră un tablou bidimensional pătratic a cu n linii ($n \leq 50$), cu componente numere întregi. Să se identifice o zonă pătratică în matrice, cu latura strict mai mică decât n , care conține cele mai multe valori egale cu numărul întreg k . Dacă există mai multe astfel de zone se va afișa cea de arie minimă. Programul va conține definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui k și ale elementelor tabloului a .
- funcția *SubT*, care primește trei valori întregi prin parametrii x , y , l . Aceasta determină numărul de valori egale cu k din zona care are colțul stânga-sus în coordonatele x , y și este formată din l linii și l coloane.
- subprogramul *Solve*, care determină zona pătratică din matricea a care conține cele mai multe valori egale cu k . Rezultatul este returnat prin trei parametri $x1$, $y1$, lm reprezentând coordonatele colțului stânga-sus ($x1$, $y1$) și numărul lm de linii și coloane ale zonei determinate.

Exemplu: Pentru $n=5$, $k=3$ și tabloul a

2 1 5 6 3

4 3 3 3 3

1 0 1 2 3

2 4 5 6 7

2 3 3 3 6

se va afișa 2 2 4 (colțul stânga sus de coordonate 2, 2 având 4 linii și 4 coloane).

24. Numim matrice rară, o matrice ale cărei elemente sunt în majoritate egale cu 0. O astfel de matrice poate fi memorată într-un vector de înregistrări în care fiecare element nenul este memorat împreună cu numărul de ordine al său, obținut în urma liniarizării matricei. Pornind de la un astfel de vector, și cunoscând numărul de linii și de coloane (n , $m \leq 50$) ale matricei rare pe care o codifică, să se genereze în memorie tabloul bidimensional corespunzător acestuia.

Programul va conține definite subprogramele:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale tuturor celor nr înregistrări ale vectorului a .
- funcția *L*, care primind trei valori întregi prin parametrii x , y și t , determină linia pe care este situat elementul cu numărul de ordine t , obținut la liniarizarea unei matrice cu x linii și y coloane.
- funcția *C*, care primind trei valori întregi prin parametrii x , y și t , determină coloana pe care este situat elementul cu numărul de ordine t , obținut la liniarizarea unei matrice cu x linii și y coloane.
- subprogramul *Solve*, care construiește în memorie tabloul bidimensional corespunzător matricei rare codificate într-un vector de înregistrări. Șirul este primit ca parametru, împreună cu numărul de elemente al său. Matricea construită este transmisă printr-un alt parametru.

Exemplu: Pentru $n=3$, $m=4$, $nr=4$ și vectorul de înregistrări ((2,3) (7,5) (5,6) (4,9) (8,11)) se va construi matricea:

```
0 0 2 0
7 5 0 0
4 0 8 0
```

25. Toate cele n ($n \leq 50$) camere ale unui hotel au formă dreptunghiulară. Pentru mochetarea acestora, patronul apelează la o firmă care îi trimite o ofertă de prețuri și îl asigură de o reducere de $x\%$ din valoarea totală a produselor cumpărate. Patronul dispune de s de lei și se hotărăște să achiziționeze un sortiment de mochetă care costă y lei pe m^2 .

Realizați un program care determină numărul maxim de camere care pot fi mochetae complet, cu suma de bani de care dispune patronul și prețul efectiv pe care acesta îl va plăti. În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *Citeste*, care preia, de la tastatură, valorile n , x , y , s și n perechi de numere naturale, reprezentând lungimile laturilor fiecărei camere. Subprogramul va returna, prin parametrul a , un vector ale cărui elemente reprezintă șirul ariilor camerelor.
- subprogramul *Ordon*, care efectuează ordonarea crescătoare a elementelor unui vector primit printr-un parametru.
- subprogramul *Calcul*, care primește șirul ariilor prin parametrul a și trei valori întregi prin parametrii s , x și y . Subprogramul returnează, prin parametrul nr , numărul maxim de camere care pot fi mochetae complet cu suma de bani s . Valoarea finală de plată având o reducere de $x\%$ va fi returnată prin parametrul întreg p .

Exemplu:

Pentru $n=4$, $x=3$, $y=15$, $s=1000$ și dimensiunile camerelor (5, 8), (3, 6), (3, 7), (4, 5) se va afișa: 3 camere; Preț final=858.

2.2. Subprograme implementate în manieră recursivă

2.2.1 Teste cu alegere multiplă și duală

1. Considerăm următoarea funcție recursivă:

```
function F(x:real;n:byte):real;
begin
  if n<=2 then F:=2.0
    else F:=2*x + F(x-1,n-1);
end;
```

```
float F(float x, int n)
{
  if (n<=2) return 2.0;
  else return 2*x + F(x-1,n-1);
}
```