Tipuri de date structurate

Fişiere in C++

Un fişier este o colecţie de date indicat printr-un nume şi o extensie. Numele este despărţit de extensie prin punct.
Avantajul lucrului cu fisiere este evident, datele rezultate în urma execuţiei unui program putând fi salvate. În esenţă, există doua tipuri de fisiere: ***fisiere text*** si ***fisiere binare.*** Un fişier text conţine text (cifre si caractere) iar un fisier binar poate conţine si imagini, baze de date, etc.
Vom studia lucru cu fisere text.
Pentru lucrul cu fişiere text in C++ se adaugă o bibliotecă standard si anume <fstream.h>. Această bibliotecă lucrează cu fluxuri (stream-uri) :

ifstream – flux de intrare
ofstream – flux de iesire

Opreaţiile care se efectuează, în general, cu fisiere text sunt:

- deschidrea unui fisier text
- închiderea unui fisier text
- citirea datelor dintr-un fisier text
- scrierea datelor intr-un fisier text
- adăugarea datelor într-un fişier text

Pentru o mai bună înţelegere ne propunem să exemplificăm rezolvarea următoarei situaţii: *Fie fişierul text fis.in în care sunt scrise numere pe mai multe rânduri. Se cere:*
- citirea datelor din fişierul text fis.in si scrirea lor in fisierul text fis.out;

Pentru aceasta vom parcurge următoarele etape:

I. Deschiderea unui fişier text

Pentru a putea efectua operaţii cu un fişier text acesta trebuie mai intâi deschis. Astfel, se poate folosi unul din obiectele ifstream(pentru citire) sau ofstream(pentru scriere) din/în fisier.

```
ifstream f("fis.in"); //s-a deschis fisierul pentru citire
ofstream f("fis.in"); //s-a deschis fisierul pentru scriere
```

II. Citirea datelor dintr-un fişier text

```
f>>var;   //in loc de cin se scrie numele fisierului din care se face citirea - f
```

III. Scrierea datelor într-un fişier text

```
g<<var/const/expresie; //in loc de cout se scrie numele fisierului in care se face scrierea - g
```

Fisierul în care dorim să scriem textul va fi suprascris la fiecare execuţie a programului.

Programul este următorul:

```
#include<fstream.h>
int x;
int main()
{
while(f>>x)       // cât timp nu s-a ajuns la sfârşitul fisierului text
      g<<x<<" ";
return 0;
}
```

Un alt exemplu: Se citesc din fișierul "date.in" doua numere intregi. Sa se afiseze in fisierul "date.out" suma si produsul celor doua numere.

```
#include<fstream.h>
ifstream f("date.in");
ofstream g("date.out");
int a,b;
int main()
{
      f>>a>>b;
      g<<"suma este:"<<a+b;
      g<<'\n';
      g<<"produsul este:"<<a*b;
      return 0;
}
```

Tablouri

Tabloul este o lista de elemente de acelasi tip plasate succesiv intr-o zona de memorie.

Tablourile por fi : **simple (vector)** sau **multiple (matrice).**

Tablouri unidimensionale

Tablourile unidimensionale sunt tablouri cu un singur indice (vectori). Daca tabloul contine nr_elem elemente, indicii elementelor au valori intregi din intervalul [0, nr_elem-1].

Declararea unui vector:

tip nume**[**nr_elem**]**;

Exemplu:

- int v[10] ; am declarat un vector **v** cu **10** componente de **tip intreg** care au indici intre **0** si **9 , v[0], v[1],.....v[9]**

- float a[10], b[20] ; am declarat doi vectori **a** si **b** care au **10** respectiv **20** de componente de tip **real**

Un element al unui tablou poate fi utilizat ca orice alta variabila. Se pot efectua operatii asupra fiecarui element al tabloului, nu asupra intregului tablou.

Citirea si afisarea elementelor unui vector

Exemplu. Se introduc valorile componentelor unui vector **a[100]** si se atribuie aceste valori componentelor vectorului **b[100]**.

```
#include<iostream.h>
int main()
{
int n,i,a[100],b[100];
cout<<"Introduceti numarul de componente n="<<" ";cin>>n;
for(i=1;i<=n;i++)
                  { cout<<"a["<<i<<"]=";cin>>a[i];}
for(i=1;i<=n;i++)  b[i]=a[i];
cout<<'\n';
for(i=1;i<=n;i++)  cout<<b[i]<<" ";
return 0;
}
```

Un tablou unidimensional poate fi **initializat** cu un set de valori astfel:

int a[5]={-2,4,8,1,9} ;

Valorile din lista de valori sunt separate prin virgula, iar intreaga lista este inclusa intre acolade.

La declararea unui vector cu initializarea elementelor sale, numarul maxim de elemente ale tabloului poate fi omis, caz in care compilatorul determina automat marimea tabloului, in functie de numarul elementelor initializate.

int a[]={-2,4,8,1,9} ; //vectorul **a** va avea 5 componente alocate, de la 0 la 4

Tablouri bidimensionale

Sunt tablouri cu doua dimensiuni: una pentru linie si una pentru coloana. Se mai numesc matrici.
Exemplu:
- int a[10][20] ; am declarat o matrice cu **10 linii** si **20 coloane** cere se adreseaza astfel:

a[0][0], a[0][1], a[0][2],.....a[9][19].

- int b[3][4]={ {11,12,13,14}, {21,22,23,24}, {31,32,33,34} } ;

Deci, pentru a parcurge elementele unei matrici, avem nevoie de doua instructiuni for.

Exemplul1. Afisarea unei matrici cu componentele declarate initial.

```
#include<iostream.h>
void main()
{
int a[3][3]={11,12,13,21,22,23,31,32,33};
int i,j;
for(i=0;i<3;i++){
      for(j=0;j<3;j++)
                  {
             cout<<a[i][j]<<' ';
             }
                  cout<<endl;
             }
return 0;
}
```

Rezultatul programului va fi afişarea următoarei matrici:

11 12 13

21 22 23

31 32 33

Exemplul2. Se introduce numărul de linii **m** si numarul de coloane **n** ale unei matrici, se intoduc elementele matricii apoi se afiseaza matricea.

```
#include<iostream.h>
int i,j,m,n,a[10][10];
int main()
{
cout<<"Introduceti numarul de linii"<<" "<<"m=";cin>>m;
cout<<"Introduceti numarul de coloane"<<" "<<"n=";cin>>n;
cout<<"Intoduceti elementele"<<endl;
for(i=1;i<=m;i++)
                  for(j=1;j<=n;j++) {
                                       cout<<"a["<<i<<j<<"]=";
                                       cin>>a[i][j];
                                     }

for(i=1;i<=m;i++){
```

```
                        for(j=1;j<=n;j++)
                              cout<<a[i][j]<<' ';

                        cout<<'\n';
                       }
        return 0;
        }
```

ALGORITMI FUNDAMENTALI CARE LUCREAZA CU VECTORI

1.MAXIM, MINIM

Se da un vector si se cere sa se determine cea mai mare/cea mai mica valoare a sa.

Rezolvare: O variabila preia continutul primei componente a vectorului (**max=v[0]** sau **min=v[0]**), apoi o compara pe rând cu celelalte componente ale vectorului, iar in functie de conditia care se pune in program, variabila va retine componenta cu valoare maxima sau componente cu valoare minima.

Pentru **maxim** :

max=v[0] ; if(v[i]>max) max=v[i];

Pentru **minim** ;

min=v[0] ; if(v[i]<min) min=v[i];

Exemplu. Se introduc valorile componentelor unui vector si se afiseaza valoarea maxima si valoarea minima.

```
#include<iostream.h>
int main()
{
int max,min,n,i,v[100];
cout<<"Introduceti numarul de elemente n="; cin>>n;
for(i=1;i<=n;i++)
             {cout<<"v["<<i<<"]=";cin>>v[i];};
max=v[0];
for(i=1;i<=n;i++)   if(v[i]>max) max=v[i] ;
cout<<"Valoarea maxima citita este"<<" "<<max<<'\n';
```

```
min=v[0];
for(i=1;i<=n;i++) if(v[i]<min) min=v[i];
cout<<"Valoarea minima citita este"<<" "<<min;
return 0;
}
```

2.ELEMENTE DISTINCTE

Se citeste un vector cu **n componente** si se decide daca numerele citite sunt distincte (nu exista doua numere egale) sau daca nu sunt distincte (exista doua numere egale).

Pentru a rezolva problema se procedeaza astfel:

- o variabila **i** retine indicele **primei componente**

- o variabila **j** retine indicele **urmatoarelor componente**

Ex: cand **i=1 j=2,3,.....n**

cand **i=2 j=3,4,.....n**

cand **i=n j=n-1**

- se initializeaza o variabila **gasit** cu valoarea logica **0**

-- daca sunt gasite doua valori egale variabilei **gasit** i se atribuie vloarea logica **1**

Exemplu.

```
#include<iostream.h>
int main()
{
int v[10],i,j,n,gasit;
cout<<"introduceti numarul de elemente n="<<" "; cin>>n;
for(i=1;i<=n;i++)
           { cout<<"v["<<i<<"]="; cin>>v[i]; }
 gasit=0;
for(i=1;i<=n ;i++)
    for(j=i+1;j<=n ;j++)
       if(v[i]==v[j]) gasit=1;
if(gasit) cout<<"Numerele nu sunt distincte";
else cout<<"Numerele sunt distincte";
 return 0; }
```

3.MULTIMI

In cadrul unei multimi, un element **apare o singura data** (o multime nu poate avea 2 valori egale). Elementele unei multimi sunt memorate intr-o variabila de tip vector.

Aplicatii:

Exemplul 1. Se citeste o multime **A** care contine **n** elemente numere intregi , se citeste un numar intreg **e ,** se verifica daca numarul **e** apartine multimii **a.**

```
#include<iostream.h>
int main()
{
int A[10],n,e,i,j,gasit;
cout<<"Introduceti numarul de elemente n a multimii"<<" "<<"n=" ; cin>>n;
for(i=1;i<=n;i++) { cout<<"A["<<i<<"]="; cin>>A[i]; }
cout<<"Introduceti numarul considerat"<<" "<<"e="; cin>>e;
gasit=0;
for(i=1;i<=n;i++)
for(j=i+1;j<=n;j++)
if(A[i]==e) gasit=1;
if(gasit) cout<<"Numarul"<<" "<< e<<" apartine multimii";
else cout<<"Numarul"<<" "<<e<<" nu apartine multimii";
return 0;
}
```

Exemplul2. Se citesc multimile **A** si **B** si se afiseaza multimea **C** unde **C = A- B**

```
#include<iostream.h>
int main()
{
int A[10],B[10],C[10],m,n,i,j,z,k,gasit;
cout<<"Specificati numarul de elemente a multimii A"<<" "<<"m="; cin>>m;
cout<<"Specificati numarul de elemente a multimii B"<<" "<<"n="; cin>>n;
cout<<"Introduceti elementele multimii A"<<'\n';
for(i=1;i<=m;i++) { cout<<"A["<<i<<"]="; cin>>A[i];};
cout<<"Introduceti elementele multimii B"<<'\n';
for(j=1;j<=n;j++) { cout<<"B["<<j<<"]="; cin>>B[j];};
k=0;
for(i=1;i<=m;i++)
      {
      gasit=0;
      for(j=1;j<=n;j++)
       if(A[i]==B[j])gasit=1;
      if(!gasit) C[k++]=A[i];
      }
cout<<"A-B"<<" "<<"={"<<" ";
for(i=0;i<k;i++) cout<<C[i]<<" "; cout<<"}" ;
return 0;;
```

```
}
```

Algoritmul de rezolvare este urmatorul:

Pentru fiecare element din multimea **A** se face testul daca apartine sau nu multimii **B**. Daca **nu apartine** este adaugat unei multimi **C** care initial este vida (variabila **k** cu valoare initiala **0** retine indicele componentei din **C** care va memora urmatorul element ce se adauga multimii **C**. In final se tipareste multimea **C**.

4. Sortarea crescatoarea a unui tablou unidimensional in C++

```
#include<iostream.h>
typedef int sir[25];
sir v;
int i, n, ok, aux;
void main()
{
cout<<”n=”;
cin>>n;
for(i=1;i<=n;i++)
{
cout<<”v["<<i<<"]=”;
cin>>v[i];
}
for(i=1;i<=n;i++)
cout<<v[i]<<” “;
//sortarea crescatoare
do{
ok=1;
for(i=1;i<=n-1;i++)
if(v[i]>v[i+1])
{
// interschimbare
aux=v[i];
v[i]=v[i+1];
v[i+1]=aux;
ok=0;
}
}while(ok!=1);
cout<<”————–”<<endl;
for(i=1;i<=n;i++)
cout<<v[i]<<” “;
}
```

5. Cautarea secventiala a unei valori intr-un tablou unidimensional - C++

Se da un vector cu n elemente si o valoare x. Se cere sa se verifice daca x se afla printre elementele vectorului. Daca se afla, se va afisa pozitia pe care a fost gasit, iar in caz contrar, se va afisa un mesaj corespunzator.

Rezolvare: se parcurge vectorul secvential (adica element cu element), comparand fiecare element cu valoarea lui x. In cazul in care x==v[i], variabila *gasit*, care a fost initializata cu valoarea -1, va retine pozitia pe care s-a gasit elementul. Daca *gasit* ramane -1, inseamna ca valoarea nu se afla in vector.

```
#include<iostream.h>
typedef int sir[25];
sir v;
int i,n,gasit=-1;

int main()
{
cout<<”n=”;
cin>>n;

cout<<”Cauta elementul =”;
cin>>x;

for(i=1;i<=n;i++)
{
cout<<”v["<<i<<"]=”;
cin>>v[i];
}

for(i=1;i<=n;i++)
if(v[i]==x)
      {
            gasit=i;
            break;
      }
if(gasit!=-1)cout<<”Elementul a fost gasit pe pozitia <<gasit;
else
      cout<<”Elementul nu s-a gasit in sir”;
return 0;
}
```

6. Cautarea binara intr-un tablou unidimensional - C++

Se da un vector cu n elemente, ordonate crescator (descrescator) si o valoare x. Se cere sa se verifice daca x se afla printre elementele vectorului. Daca se afla, se va afisa pozitia pe care a fost gasit, iar in caz contrar, se va afisa un mesaj corespunzator.

Metoda de rezolvare consta in cautarea valorii intre doua limite li si ls, care initial sunt 1, respectiv n. Se injumatateste intervalul si se verifica cum este valoarea cautata fata de valoarea din mijloc (fata de v[lm]). Daca este mai mica, se muta cautarea in jumatatea din stanga a intervalului si se actualizeaza capatul din dreapta al acestuia (ls=lm), daca este mai mare se muta cautarea in jumatatea din dreapta si se actualizeaza capatul din stanga (li=lm+1), iar daca este egala, inseamna ca s-a gasit pe pozitia data de lm (mijlocul intervalului). Acest mecanism de rezolvare este permis tocmai pentru ca elementele vectorului sunt ordonate.

```
#include<iostream.h>
typedef int sir[25];
sir v;
int i,n,li,ls,lm, gasit;

int main()
{
cout<<”n=”;
cin>>n;

cout<<”Cauta elementul =”;
cin>>x;

for(i=1;i<=n;i++)
{
cout<<”v["<<i<<"]=”;
cin>>v[i];
}

li=1;
ls=n;
gasit=0;

while((li<=ls)&&(!gasit))
{

      lm=(li+ls)/2;
      if(v[lm]<x)
      {
      li=lm+1;
      }
      else if(v[lm]==x)
      {
      cout<<”l-am gasit pe pozitia”<<lm;
      gasit=1;
      }
```

```
      else ls=lm-1;
      }

      if(!gasit)cout<<”Nu apare in vector”;

return 0;}
```

7. Interclasarea a doua tablouri unidimensionala in C++

Se dau doi vectori cu m, respectiv n elemente, ordonate crescator. Se cere sa se obtina un al treilea vector care sa contina elementele celor doi vectori cititi, in ordine crescatoare.

Rezolvare: Se parcurg cei doi vectori simultan si se compara elementele lor curente (a[i] cu b[j]), selectandu-se in vectorul c cel mai mic dintre ele. La selectarea unei componente, se incrementeaza indicele cu care se parcurge sirul din care s-a selectat elementul. Procedeul continua pana cand cel putin unul din cei doi vectori a fost parcurs in intregime. La final, se copiaza in vectorul c toate elementele vectorului care nu a fost terminat de parcurs, aceste elemente fiind ordonate crescator, sirul obtinut va fi crescator.

```
#include<iostream.h>
typedef int sir[25];
sir a,b,c;
int n,m,i,j,k;

int main()
{
cout<<”Introduceti lungimile celor 2 vectori a si b= “;
cin>>m>>n;

// cititi elementele celor 2 vectori ordonati crescator
for(i=1;i<=m;i++)
{
cout<<”a["<<i<<"]=”;
cin>>a[i];
}

for(i=1;i<=n;i++)
{
cout<<”b["<<i<<"]=”;
cin>>b[i];
}

// interclasarea

i=1;
j=1;
k=0;

while((i<=m)&&(j<=n))
{
```

```
if(a[i]<b[j])
{
k++;
c[k]=a[i];
i++;
}

else {
k++;
c[k]=b[j];
j++;
}

}

// adaugam elementele ramase neparcurse

if(i<=m)
{
while(i<=m)
{
k++;
c[k]=a[i];
i++;
}

}

if(j<=n)
{
while(j<=n)
{
k++;
c[k]=b[j];
j++;
}
}

// afisez vectorul interclasat
for(i=1;i<=k;i++)
cout<<c[i]<<"  ";

return 0;

}
```

APLICATII CARE LUCREAZA CU MATRICI

1.Interschimbarea a doua linii intre ele sau a doua coloane

Pentru a interschimba 2 variabile intre ele utilizam o a treia variabila de manevra care am denumit-o **temp** si inca doua variabile **x** si **y** carora le atribuim ca valori numerele liniilor sau ale coloanelor pe care dorim sa le interschimbam.

a) Interschimbarea a **2 linii**

```
for(j=1;j<=n;j++) {
                   temp=a[x][j];
                   a[x][j]=a[y][j];
                   a[y][j]=temp ;
                  }
```

b) Interschimbarea a **2 coloane**

```
for(i=1;i<=n;i++) {
                   temp=a[i][x];
                  a[i][x]=a[i][y];
                  a[i][y]=temp ;
                   }
```

Exemplu. Schimbarea a **2 coloane** ale unei matrici

```
#include<iostream.h>
int main()
{
int i,j,m,n,a[10][10],x,y,temp;
cout<<"Introduceti numarul de linii"<<" "<<"m=";cin>>m;
cout<<"Introduceti numarul de coloane"<<" "<<"n=";cin>>n;

cout<<"Introduceti elementele\n";
for(i=1;i<=m;i++) {
for(j=1;j<=n;j++) { cout<<"a["<<i<<j<<"]=", cin>>a[i][j];}}

cout<<"Matricea introdusa are forma:\n";
for(i=1;i<=m;i++){
for(j=1;j<=n;j++)  {cout<<a[i][j]<<' '; } cout<<'\n'; }
cout<<'\n';
```

```
cout<<"Introduceti numerele coloanelor care doriti sa le
interschimbati"<<endl;
cout<<"x=";cin>>x;cout<<"y=";cin>>y;
for(i=1;i<=n;i++) { temp=a[i][x]; a[i][x]=a[i][y]; a[i][y]=temp ; }
cout<<'\n';

cout<<"Noua matrice  are forma:\n";
for(i=1;i<=m;i++){
for(j=1;j<=n;j++){ cout<<a[i][j]<<' '; } cout<<'\n';}
return 0;
}
```

2. Afisarea elementelor de pe perimetrul matricei

```
#include<iostream.h>
int main()
{
int i,j,m,n,a[10][10];
cout<<"Introduceti numarul de linii"<<" "<<"m=";cin>>m;
cout<<"Introduceti numarul de coloane"<<" "<<"n=";cin>>n;

cout<<"Introduceti elementele\n";
for(i=1;i<=m;i++) {
for(j=1;j<=n;j++) { cout<<"a["<<i<<j<<"]=", cin>>a[i][j];}}

cout<<"Matricea introdusa are forma:\n";
for(i=1;i<=m;i++){
for(j=1;j<=n;j++)  {cout<<a[i][j]<<' '; } cout<<'\n'; }
cout<<'\n';
cout<<"Elementele de pe perimetru:\n";
for(j=1;i<=n;j++)cout<<a[1][j]<<" ";
for(i=2;i<=m;i++)cout<<a[n][i]<<" ";
for(j=n-1;j>=1;j--)cout<<a[m][j]<<" ";
for(i=m-1;i>=2;i--)cout<<a[i][1]<<" ";
return 0;
}
```

3.Afisarea elementelor de pe diagonalele unei matrici patratice

Elementele aflate pe diagonala principala au j=i, iar cele de pe diagonala secundara au j=n-i+1.

```
#include<iostream.h>
int main()
{
int i,j,m,n,a[10][10];
cout<<"Introduceti numarul de linii si de coloane"<<" "<<"n=";cin>>n;

cout<<"Introduceti elementele\n";
```

```
for(i=1;i<=n;i++) {
for(j=1;j<=n;j++) { cout<<"a["<<i<<j<<"]=", cin>>a[i][j];}}

cout<<"Matricea introdusa are forma:\n";
for(i=1;i<=n;i++){
for(j=1;j<=n;j++)  {cout<<a[i][j]<<' '; } cout<<'\n'; }
cout<<'\n';
cout<<"Elementele de pe diagonala principala:\n";

for(i=1;i<=n;i++)cout<<a[i][i]<<" ";

cout<<'\n';

cout<<"Elementele de pe diagonala secundara:\n";

for(i=1;i<=n;i++)cout<<a[i][n-i+1]<<" ";

return 0;

}
```

BIBLIOGRAFIE

1. DANA LICA, *Informatică: Manual pentru clasa a IX-a*, Ed. L&S Infomat, Bucureşti, 1999
2. DORU POPESCU ANASTASIU, MIRCEA BALAN, *Tehnologia informaţiei si comunicării pentru gimnaziu*, Ed. L&S Info-mat , Bucuresti, 2008
3. LUMINITA RIPEANU, *Ghid de Tehnologia Informaţiei pentru Gimnaziu, Editura LVS Crepuscul, 2010*
4. RODICA MATEI, DORINA MATEIAŞ, *Tainele informaticii, manual de informatica clasele V - VIII*, Ed. PARALELA 45, Bucuresti, 2009
5. ***Internet

CUPRINS