STRUCTURI DE DATE - TABLOURI

Tablouri unidimensionale
VECTORI

1. NOȚIUNI INTRODUCTIVE
A. NOȚIUNEA DE VECTOR

- În practică, apare deseori necesitatea de a prelucra un set de valori de același tip, așezate într-o anumită ordine.
- O astfel de structură se numește **șir**, iar valorile respective se numesc **elementele șirului.**
- **De exemplu**, cu mediile generale ale elevilor unei clase putem alcătui un șir de numere reale.

- Este ineficient să declarăm câte o variabilă pentru fiecare element al şirului, deoarece numărul acestor elemente poate fi foarte mare.
- Limbajul C++ oferă posibilitatea de a memora toate elementele unui şir într-o singură *variabilă indexată*, în care *elementele sunt dispuse într-o anumită ordine, ocupând locaţii de memorie succesive, bine-determinate*.
- O astfel de variabilă se numeşte **tablou unidimensional** sau **vector.**

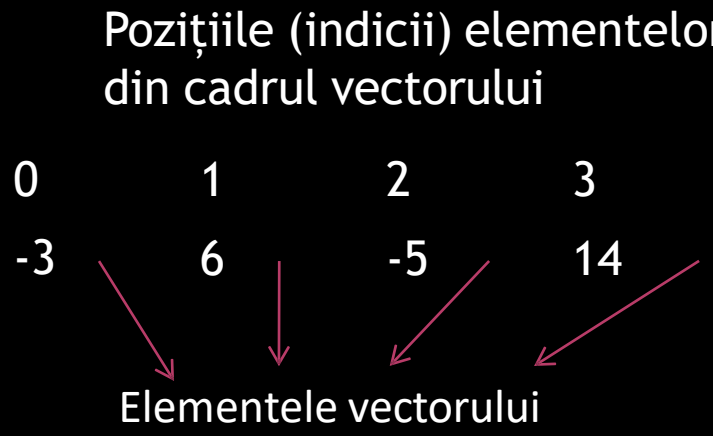
- Elementele șirului memorat într-un vector se numesc generic ”**elemente ale vectorului**”.
- Fiecare element este identificat prin locul pe care îl ocupă în cadrul vectorului; adică prin ceea ce numim "**poziţia**" sau "**indicele**" sau "**rangul**" elementului.
- Poziţiile (indicii) elementelor în vector sunt numerotate succesiv începând cu **0 (0, 1, 2, 3,...)**.
- *Pentru a referi un anumit element al vectorului, trebuie să scriem numele variabilei-vector urmat de poziţia elementului cuprinsă între paranteze pătrate.*

- **Exemplu:** dacă numele variabilei-vector este v, atunci prin **v[3]** înțelegem elementul de pe poziția 3, sau altfel spus elementul de indice 3, în vector.
- Poziția (indicele) unui element NU arată al câtelea este elementul respectiv în vector, deoarece numerotarea pozițiilor începe de la 0.
- Astfel, în exemplul anterior, elementul **v[3]**, de pe poziția 3 (sau de indice 3), nu este al treilea element al vectorului, ci al patrulea, înaintea sa aflându-se trei elemente, și anume **v[0]**, **v[l]** și **v[2].**

- Lucrările de specialitate asimilează complet noțiunea de **poziție** cu cea de **indice**.
- Dacă vrem să păstrăm concordanța avem o soluție foarte simplă: nu folosim elementul **v[0].** Dacă am utiliza elementele doar începând cu **v[1],** atunci **v[1]** ar fi primul element, **v[2]** ar fi al doilea, ș.a.m.d., iar poziția fiecărui element ar indica într-adevăr al câtelea este el în vector.
- Vom adopta această soluție în unele aplicații, acolo unde ne va ajuta să urmărim mai ușor parcurgerea și prelucrarea vectorului.

EXEMPLU:

- Considerăm șirul de numere întregi: -3, 6, -5, 14.
- Notăm variabila vector cu **v** și pozițiile elementelor în vector cu **0,1,2,3.**



- Elementul de pe poziția **0** în vector este primul element al șirului adică -3, și se notează **v[0]**; elementul de pe poziția**1** în vector este al doilea element al șirului, deci **v[1l]=6** analog, **v[2]=-5, v[3]=14.**

B. DECLARAREA UNUI VECTOR

- Un vector trebuie declarat, la fel ca orice variabilă, în secțiunea de declarații a programului.
- Variabila-vector aparține unui tip de date nou, numit "tipul **array**". În declarația unui vector trebuie să apară:
  - **identificatorul vectorului**, (numele variabilei- vector)
  - **tipul elementelor.**

Sintaxa este:

<t_elem> <nume>[nr_elem];

unde>

- **<t_elem>** —> tipul de date al elementelor;
- **<nume>** —> numele (identificatorul) variabilei-vector;
- **<nr_elem>** —> numărul maxim al elementelor vectorului.

- Orice vector se caracterizează printr-un **număr maxim de elemente,** reprezentând *cel mai mare număr de elemente care s-ar putea memora in vectorul respectiv* (o valoare constantă).
- Acesta se precizează **la declararea vectorului**, între **paranteze pătrate**.
- Poziţiile elementelor vor fi numerotate începând de la 0 până la "numărul maxim de elemente minus 1".

EXEMPLU:

int v[20] ;

- Am declarat un vector v cu maxim 20 de elemente numere întregi;
- numele variabilei-vector este v;
- tipul elementelor vectorului este **int,** adică elementele sunt numere întregi.
- Elementele vectorului sunt v[0], v[1], …, **v[19]** având indicii 0,1,2,...,19.
- *Tipul elementelor poate fi orice tip predefinit sau definit de utilizator.*

DECLARAȚII CORECTE DE VECTORI:

- Care dintre variantele de mai jos reprezintă o declarație corectă a unui vector **v** cu 20 de elemente numere întregi ?
  a) v[20]: integer;
  b) v[20] int;
  c) int v[20];
  d) int: v[20];
  e) integer v[20] ;
- **Varianta corectă este c.**
- În varianta *b)* sunt inversate sintagmele **"int"** și **"v[20]"**,
- În varianta *d)* apare separatorul ":" între tipul elementelor și numele vectorului
- În variantele *a)* și *e)* identifică greșit tipul întreg al elementelor (denumirea acestuia este **int** și nu **integer**).

2. PARCURGEREA, CITIREA ŞI AFIŞAREA UNUI VECTOR

A. NUMĂRUL REAL (EFECTIV) DE ELEMENTE

- In urma declarării unui vector, compilatorul rezervă pentru elementele sale o zonă fixă de memorie, care să poată memora numărul maxim de elemente (precizat la declarare).
- În general, într-un program în care am declarat un vector nu vom folosi toate elementele acestuia. Pentru ca programul să fie cât mai general şi eficient, ar trebui ea la"fiecare execuţie a sa să putem memora în vector un alt şir alcătuit din alte elemente.

- Procedăm astfel:
  - Definim un așa-numit **număr real (efectiv) de elemente**, reprezentând *numărul elementelor care vor fi efectiv folosite la o anumită execuție a programului*.
  - Numărul real de elemente precum și elementele vectorului se vor citi de la tastatură în timpul execuției.

EXEMPLU:

- Considerăm vectorul declarat astfel: **int v[25]** ;
  - în urma declarării, se rezervă memorie pentru 25 de elemente, în speță pentru elementele **v[0], v[1], …, v[24]** (reamintim că pozițiile elementelor se numerotează începând cu 0 !).
  - Dacă notăm cu **n** numărul real (efectiv) de elemente ale vectorului, atunci elementele (componentele) efectiv folosite, vor **fi v[0], v[1],..., v[n-2], v[n-1],** și ocupă în cadrul vectorului pozițiile (indicii) 0,1,..., n-2, n-1.
  - Atunci când dorim să asigurăm concordanța între poziția unui element și al câtelea este el în vector, putem să nu utilizăm elementul **v[0]**, caz în care elementele efectiv folosite vor fi **v[1], v[2],..., v[n-1], v[n]**.
  - **În** general, prin **v[i]** se înțelege elementul de pe poziția **i** în vector, unde **i** parcurge pe rând valorile **0**, **1**, ... , **n-1**, sau **1**,**2**, …, **n.**

- Elementele unui vector se comportă ca niște **variabile simple**, deci pot fi *citite*, *afișate* și *folosite* în expresii și atribuiri.
- În continuare, considerăm vectorul

v= (v[0],v[1],. . . ,v[n-1]), unde n este numărul real de elemente.

- **Prin** parcurgerea vectorului se înțelege "vizitarea" elementelor pe rând, și prelucrarea acestora. Pentru a parcurge primele **n** elemente ale vectorului **v** putem proiecta un ciclu, astfel:
  - *Contorul ciclului **i*** va *lua pe rând valorile 0, 1,..., n-1, reprezentând pozițiile elementelor în vector;*
  - *La fiecare pas, pentru fiecare valoare a lui i, "vizităm" si prelucrăm elementul **v[i]**.*

B. PARCURGEREA CU CICLU FOR

pentru (i←0,1, …,n-1)
<prelucrează>

Pașii ciclului:

Pasul 1: i:=0, prelucrează **v[0]**

Pasul 2: i:=1, prelucrează **v[1]**

…….

Pasul n: **i**:=n-1, prelucrează **v[**n-1**]**

În cazul în care nu am fi dorit să folosim elementul **v[0]**, s-ar fi prelucrat succesiv elementele **v[1]**, **v[2] , ... ,v[n]**, iar ciclul de parcurgere ar fi:

pentru (i←1, 2, …,n)
<prelucrează>

1. CITIREA UNUI VECTOR CU N ELEMENTE, FOLOSIND UN CICLU FOR

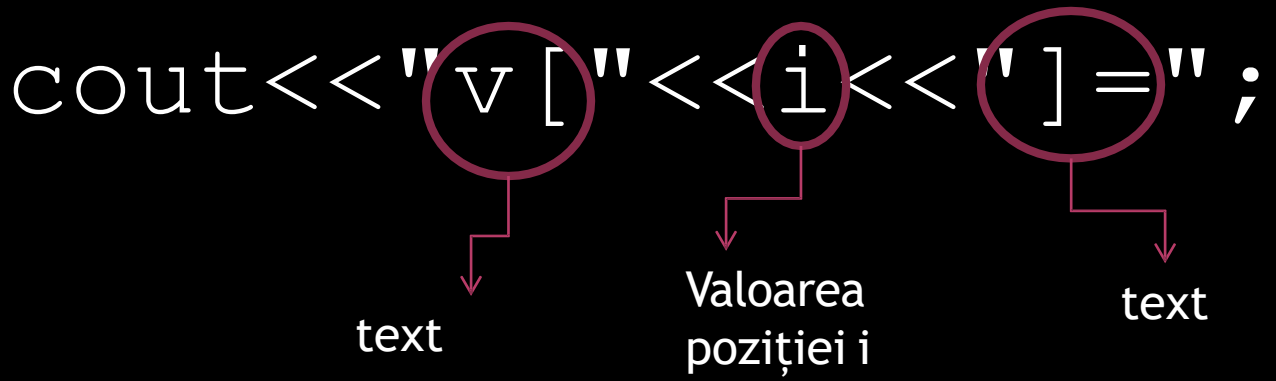
- Nu putem citi "dintr-o dată" toate elementele vectorului printr-o instrucțiune de genul (cin>>v);, pentru că **v** este o variabilă indexată (compusă) care înglobează mai multe valori.
- Vom citi mai întâi numărul real de elemente **n.**
- ***Parcurgem într-un ciclu pozițiile elementelor i=0,1, … ,n-l și pentru fiecare valoare a lui i, citim elementul de pe poziția i, adică elementul v[i].***
- Astfel, pentru i=0 se citește elementul **v[0]**, pentru **i=1** are loc citirea lui **v[1]**, ș.a.m.d.

```
#include<iostream.h>
int main(void)
{
int i,n;
int v[20];
cout<<"dati numarul de
  elemente ale
  vectorului:";
cin>>n;
for(i=0;i<=n-1;i++)
   {

        cout<<"v["<<i<<"]="
   ;
        cin>>v[i];
   }
}
```

- Condiția de trecere la pasul următor "i<=n-1" din linia **for,** putea fi scrisă și sub forma " i<n" (echivalentă din punct de vedere matematic).
- În secvența de program de mai sus, înainte de citirea elementului **v[i]** (cu **cin**), apare linia, `cout<<"v["<<i<<"]=";` care afișează un mesaj, cu scopul de a ne spune ce element urmează să citim la pasul respectiv al ciclului.
- O astfel de comandă nu este obligatorie, dar este recomandabilă.

DE EXEMPLU, ÎNAINTE DE A AŞTEPTA INTRODUCEREA UNEI VALORI PENTRU ELEMENTUL **V[2]**, CALCULATORUL VA AFIŞA:



De exemplu, înainte de a aştepta introducerea unei valori pentru elementul **v[2]**, calculatorul va afişa:
v[2]= _

cursor în așteptare

- Desigur că se putea afișa un mesaj mai simplu, dar mai puțin elegant, de genul:
- Astfel, înainte de citirea elementului **v[2]**, se va afişa textul:

```
COUT<<"\N DAŢI ELEMENTUL DE PE
            POZIŢIA "<<I<< ":";
```

```
Daţi elementul de pe poziţia 2:
```

- Așa cum am precizat la început, odată ce am declarat un vector, nu este obligatoriu să folosim toate "căsuțele" acestuia.
- Astfel, dacă vrem să folosim **n** elemente din vector, acestea pot fi foarte bine **(v[1], v[2], …, v[n])**, în loc de **(v[0], v[1], …, v[n-1]**).
- Singura modificare pe care ar suferi-o algoritmul de citire al vectorului în acest caz, se referă la parcurgerea pozițiilor cu contorul i: acesta va lua valori de la 1 la n. Ciclul de citire arată astfel:

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

2. AFIȘAREA UNUI VECTOR CU N ELEMENTE, FOLOSIND UN CICLU FOR

- Nu putem afișa "dintr-o dată" toate elementele vectorului printr-o instrucțiune `cout<<v;`.
- Folosim același ciclu de parcurgere a vectorului: vom parcurge pozițiile elementelor din vector i=0,1. . .n-1 și pentru fiecare valoare a lui i, afișăm elementul de pe poziția i, adică **v[i]**.

- Se tipăresc toate elementele pe un singur rând (cu **cout**).
- La fiecare pas se afișează un element **v[i]**, urmat de un caracter *"spațiu* ".

```
for(i=0;i<=n-1;i++)
  cout<<v[i]<<" ";
```

sau

- Se tipărește fiecare element al vectorului pe rând nou.
- La fiecare pas înainte de afișarea un element **v[i]**, cursorul sare pe un rând nou.

```
for(i=0;i<=n-1;i++)
  cout<<"\n"<<v[i];
```