

Apunts 3r parcial Pro1

Vectors

Inicialització de vectors

- Per declarar un vector primer de tot hem d'incloure la llibreria vector.
- Un vector es declara de la següent forma:
 - `vector <tipus> nom_vector (tamany)`
- Existeix un segon tipus de declaració d'un vector al qual indiquem a quin valor el volem inicialitzar. Recordem el següent: quan creem una variable i aquesta no és inicialitzada no sabem el que tenim allà dins, el mateix pot passar amb els vectors. Si volem inicialitzar un vector amb totes les posicions a 0 podem fer el següent:
 - `vector <int> nom_vector (5 , 0)` . Amb aquesta declaració he inicialitzat el vector de mida 5 amb totes les posicions el valor 0.
- Aquesta inicialització també es pot fer amb variables booleanes per exemple entre d'altres
 - `vector <bool> nom_vector(n , false)`
- A continuació es mostra un exemple de dos tipus de declaracions de vectors.

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int tamany;
    cin >> tamany;
    vector <int> vec(tamany , 0);
    vector <bool> vec(tamany , false);
}
```

- **Nota :** Recordem que les posicions d'un vector van de 0 a $n - 1$.
 - **Exemple :** Si tinc un vector de mida 4 ,la seva primera posició serà l'índex 0 i per altre banda la seva última serà l'índex 3.
- **Nota 2 :** Un vector sempre s'ha de inicialitzar amb un tamany . No es pot inicialitzar cap vector sense mida exceptuant aquells vectors que s'inicialitzen a partir d'un vector que és obtingut per una funció. (Això ho veurem a classe)

Lectures i escriptures d'un vector

- Tant per poder llegir com un vector com per escriure és important saber el seu tamany. Una forma fàcil i ràpida és utilitzant la funció de la llibreria vector.
 - Sigui **nom_vec** el nom del nostre vector podrem saber el seu tamany fent **nom_vec.size();**

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int tamany;
    cin >> tamany;
    vector<int> vec(tamany , 0);

    // omplir un vector

    for(int i = 0; i < vec.size(); ++i) cin >> vec[i];

    //escriure un vector per pantalla

    for(int i = 0; i < vec.size(); ++i) cout << vec[i] << " ";

}
```

- La lectura i escriptura és tan senzill com fer un for des de la posició 0 fins a la $n - 1$ si el volem imprimir en seqüència i des de la posició 0.

- Si per altre banda el volem imprimir des de el final però volem utilitzar el mateix for començant des de 0 fins a $n - 1$ es pot aplicar la següent formula.
 - Sigui $n - 1$ la posició final del vector $\Rightarrow \text{vec}[n - 1 - i]$. Això en la primera iteració com la nostre i val 0 anirem a la posició $n - 1$. Conforme vagi creixent la i anirem arribant cap a índexs més baixos.

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int tamany;
    cin >> tamany;
    vector<int> vec(tamany , 0);

    // omplir un vector

    for(int i = 0; i < vec.size(); ++i) cin >> vec[i];

    // Lectura d'un vector comensant per el índex més alt

    for(int i = 0; i < vec.size(); ++i) cout << vec[vec.size() - 1 - i] << " ";

}
```

Pas per referència

- En vectors al ser variables molt grans és important entendre què és el pas per referència i quina diferència hi ha de fer per referència i no fer-lo.

Observem el següent gif :



El pas per valor el que fa es crea una copia d'aquella variable dins de la funció. Tot el que nosaltres efectuem sobre aquella variable que passem per valor un cop sortim de la funció la original no es veurà afectada.

Per altre banda si observem el gif de sobre veurem que conforme es va omplint la tassa de la funció la nostre principal també.

Llavors per explicar-ho de forma fàcil i sense entrar en detalls concrets (això s'explica a EC) podem dir doncs que el que fem quan passem un valor per referència és crear un enllaç directa amb la variable original. Tot el que nosaltres modifiquem en la funció, si aquesta variable ha sigut passada per referència, fora de la funció també es veurà afectada. Altrament no.

- Per què fem servir el pass per referència en vectors quan no el vui modificar dins la funció?
 - El pas per referència serveix per evitar fer la còpia de tot el vector. Hem de començar a pensar en codi eficient tant a nivell de memòria com de velocitat. Al ser una variable molt gran aquesta ocupa molta memòria, per tant fer la còpia de tot el vector a la memòria costa molt de temps. Fent el pas per referència simplement estem passant un enllaç . Finalment si aquest vector no el modifiquem declarem el vector com a const per evitar que es pugui modificar.

```
#include <iostream>
#include <vector>
using namespace std;

void fa_algo(const vector<int> &v) {
    // ..... //
}

int main() {
    int tamany;
    cin >> tamany;
    vector<int> vec(tamany, 0);

    // omplir un vector

    for(int i = 0; i < vec.size(); ++i) cin >> vec[i];

    // crido a la funció

    fa_algo(vec);
}
```

- Tal i com es pot observar el pas per referència s'indica amb el símbol **&** i es posa entre la declaració del vector i el seu nom.

Matrius

- Una matriu en programació no és més que un vector de vectors . Sigui v1 el primer vector per cada posició de v1 a dins i tenim un segon vector que anomenem v2 de mida n.

mat

Columns

Rows

50	5	27	400	7
0	67	90	6	97
30	14	23	251	490

- Per cada fila de la matriu (vector 1) tenim n columnes on n columnes és igual al tamany del vector 2 i cada fila és una posició de v1. (per comprendre millor es realitzarà un dibuix a la pissarra que posteriorment serà adjuntat aqui)

Declaració de matrius

- Tal i com hem dit una matriu no és res més que un vector de vectors. Per tant caldrà declarar un vector dins d'un altre vector

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;

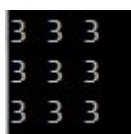
    // Declaració de la matriu

    vector <vector<int> > matriu (n , vector <int> (4 , 3));

    // escriptura de matriu per pantalla

    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) {
            cout << matriu[i][j] << " ";
        }
        cout << endl;
    }
}
```

- Tal i com observem per declarar una matriu cal seguir els següents passos.
 1. Declarar vector de la forma **vector < tipus > nom_vector (tamany)**
 2. En el tipus del vector declarem dins un altre vector escrivint sols fins el seu tipus
 3. A nom vector indiquem el nom de la nostre Matriu.
 4. Un cop dins del vector indiquem la seva mida.
 5. Recordem que si posem una ' , ' el següent valor indicava a quin valor s'inicialitzava el vector en la seva declaració. Allà doncs declarem novament el vector amb la seva mida i si volem amb la seva inicialització . En la foto ha sigut inicialitzat a 3 per lo que la seva sortida és la que es mostra a continuació amb n = 3.



```
3 3 3
3 3 3
3 3 3
```

Esriptures i lectures d'una matriu

- Per escriure i llegir dins d'una matriu s'ha de fer obligatòriament amb dos bucles for amb un dins de l'altre, ja que com hem mencionat una matriu en c++ és un vector de vectors. Per tant hem de recórrer dos vectors.
- **Nota :** Per convenció per recorre vectors s'utilitza la lletra “*i*” per files i la lletra “*j*” per columnes.

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;

    // Declaració de la matriu

    vector <vector<int> > matriu (n , vector <int> (n));

    // Escriptura de matriu

    for(int i = 0; i < n ; ++i) {
        for(int j = 0; j < n; ++j) {
            cin >> matriu[i][j];
        }
    }

    // Lectura de matriu per pantalla

    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) {
            cout << matriu[i][j] << " ";
        }
        cout << endl;
    }
}
```

- Tal i com es pot observar per accedir a una posició concreta d'una matriu és similar a vectors afegint uns nous “`[]`” que ens permeten seleccionar la columna .
- En la imatge anterior s'ha declarat una matriu quadrada de $n \times n$ i s'ha omplert amb valors introduïts per pantalla per després ser escrits per pantalla.

Aprenent a moure's per una matriu

A continuació s'exposen les formes per saber moure's per una matriu en funció de n .

Nota : *Aprenere a realitzar això és molt important ja que sol caure en molts casos exercicis on hagi de moure's per una matriu des de posicions concretes per qualsevol n en EXÀMENS.*

Donat que a nivell teòric és un conyas ho veurem amb el següent exemple que és més dinàmic

Exercici 1 :

Donat una matriu de 4 x 4 es desitja imprimir les dues diagonals de la matriu en:

- Des de l'esquerra a la dreta de forma descendent
- Des de la dreta a l'esquerra de la forma descendent
- Des de l'esquerra a la dreta de forma ascendent
- Des de la dreta a l'esquerra de la forma descendent

En la següent taula es mostra la matriu d'exemple sobre la qual es desitja imprimir les seves diagonals.

1	3	4	1
7	3	2	5
0	3	6	4
1	2	0	9

La sortida hauria de ser :

- 1 3 6 9
- 1 2 3 1
- 1 3 2 1
- 9 6 3 1

Nota : *S'ha de utilitzar per totes les diagonals el for amb inicialitzacions de $i = 0$ i $j = 0$.*

.

Solució pas a pas :

1. Es calcula com ens mourem per cada diagonal (En l'examen utilitzeu paper i boli abans que posar-vos a escriure codi i feu proves sobre paper si funciona per qualsevol tamany de matriu)
 - a. Diagonal d'esquerra a dreta descendent $[i][j]$ per $i == j$
 - b. Diagonal de dreta a esquerra de forma descendent $[i][n - i - j]$ per $i == j$
 - c. Diagonal esquerra a dreta de forma ascendent $[n - 1 - i][j]$ per $i == j$
 - d. Diagonal de dreta a esquerra de forma ascendent $[n - 1 - i][n - 1 - j]$ per $i == j$
2. Ara sols queda ja escriure codi quedant de la següent forma:

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<vector<int>> > matriu (n , vector<int> (n));
    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) cin >> matriu[i][j];
    }

    // primera diagonal
    cout << "Diagonal esquerra dreta descendent : ";
    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) {
            if(i == j) cout << matriu[i][j] << " ";
        }
    }
    cout << endl;

    // segona diagonal
    cout << "Diagonal Dreta esquerra descendent : ";

    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) {
            if(i == j) cout << matriu[i][n - 1 - j] << " ";
        }
    }

    cout << endl;

    // tercera diagonal
    cout << "Diagonal esquerra a dreta de forma ascendent : ";
    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) {
            if(i == j) cout << matriu[n - 1 - i][j] << " ";
        }
    }

    cout << endl;

    // quarta diagonal
    cout << "Diagonal Dreta a esquerra de forma ascendent : ";
    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) {
            if(i == j) cout << matriu[n - 1 - i][n - 1 - j] << " ";
        }
    }

    cout << endl;
}
```

3. I per pantalla es mostra el següent :

```
4
1 3 4 1
7 3 2 5
0 3 6 4
1 2 0 9
Diagonal esquerra dreta descendent : 1 3 6 9
Diagonal Dreta esquerra descendent : 1 2 3 1
Diagonal esquerra a dreta de forma ascendent : 1 3 2 1
Diagonal Dreta a esquerra de forma ascendent : 9 6 3 1
```