Apuntes OIFem II Nivel 1

Listas y Ordenamientos

Vector

Un vector almacena varios valores del mismo tipo (int, string, char, double, etc.)bajo una misma variable. Podríamos visualizarlo como la lista de la compra, la lista de los alumnos de clase, etc.

Código- Inicializar un vector:

```cpp
#include <vector> // biblioteca para usar los vectores
#include <iostream>

using namespace std;

int main(){
    vector<int>lista; //inicializamos un vector vacío
    vector<int>listade3 (3); //inicializamos un vector con 3 elementos (se pueden borrar o cambiar o añadir más elementos)
    int N;
    cin>>N;
    vector<int>listadeN (N); //inicializamos un vector con N elementos
vector<int>listadeceros(N,0); // tenemos un vector de N elementos y todos son ceros

    return 0;
}
```

Código- Añadir y borrar elementos:

```cpp
int main(){
    vector<int>lista;// lista vacía
    lista.push_back(1);// en nuestra lista tenemos el elemento 1, es decir lista[0]=1
    lista.push_back(2);// hemos añadido el dos lista[1]=2;
    lista.pop_back();// hemos eliminado el último elemento de la lista, en este caso el 2 (lista [1])
}
```

Acceder a elementos:

Los elementos dentro de un vector de tamaño N están numerados del 0 al N-1. Por ejemplo tenemos el siguiente vector de 5 amigas:

0 Ana

1 Marta

2 Isabel

3 Teresa

4 Carlota

Si queremos saber quien es la tercera amiga en la lista, buscamos en amigas[2]. Si queremos saber quien es la iesima amgiga en la lista, buscamos en amigas[i-1].

Cambiar valores:

Si se enfadan todas con Carlota y se buscan una nueva amiga, Carla, entonces pueden cambiar la lista diciendo amiga[4]=Carla. De este modo cambiará automáticamente.

En un caso general si tenemos una lista(n) y queremos cambiar el elemento x, podremos hacer esto diciendo lista[x]=nuevoelemento;

Otros comandos importantes

-Vaciar un vector:

lista.clear();

(Con esto borramos toda la información gurdada previamente en el vector)

-Varias dimensiones:

No solo podemos crear listas, si no que también tablas, vectores de dos dimensiones o más. Normalemente usamos vectores de una o dos dimensiones, en algún problema más complejo se usarán más.

Ejemplo uso de vector dos dimensiones:

```
int main(){
    vector<vector<int> >tabla(n);//Le damos tamaño n
    vector<int>auxiliar(t);//Para dar tamaño t a todos los vectores dentro de la
    ↪  tabla creamos un vector auxiliar de ese tamaña:
    for(int i=0;i<n;i++)tabla[i]=auxiliar;
    int columna,fila;
    cin>>fila>>columna;
    cout<<tabla[fila][columna];//para leer elementos
}
```

-Tamaño de un vector:

int tamaño=lista.size();

-Darle otro tamaño al vector:

lista.resize(n);

-Darle otro tamaño y valores a todos los elementos del vector(ejemplo con todos los valores = 0):

lista.assign(n,0);

Ejemplo práctico: LA FIESTA ABURRIDA

Tinín es un tipo bastante peculiar. No le gustan nada las fiestas, ni las celebraciones, ni las cenas familiares o con amigos. Nada. Es sorprendente que, a pesar de sus pocas ganas de interactuar con gente, haya encontrado novia.

Cuando ésta consigue sacarle de casa para llevarle a algún tipo de celebración y la gente le dice "Hombre, Tinín, ¡pero si has venido!" él siempre suelta su mítica frase "Yo soy más traído que venido.".

Una de las cosas que más le molesta es tener que ir saludando a todos los presentes, sobre todo si no le conocen. Eso de que se le acerquen y le digan "Soy Lotario" y que él tenga que saludar no lo soporta. ¿Puedes ayudarle?

ENTRADA: La entrada comienza con un número que indica la cantidad de gente a la que Tinín no conoce en la fiesta en la que está. A continuación viene una línea por cada una de esas personas en la que se presenta indicando su nombre: "Soy Lotario". La línea siempre tendrá el mismo formato; una primera palabra "Soy" seguida del nombre de la persona que será simple (no será un nombre compuesto por varias palabras) y formado por como mucho 100 letras del alfabeto inglés.

SALIDA: Para cada persona que se le aproxima, se debe escribir la cadena "Hola, [nombre].".

Link al problema: La fiesta aburrida

Solución- Opción 1:

Paso1: Crear vector¡string¿desconocidos

Paso2: Añadir nombre de desconocidos al vector

Paso3: Imprimir los nombres

```
#include <vector>
#include <iostream>

using namespace std;

int main(){
        int numerodesconocidos;
        cin>>numerodesconocidos;
        vector<string>desconocidos; // Paso1
        string soy;
        string nombre;
        for(int i=0;i<numerodesconocidos;i++){
                cin>>soy>>nombre;
                desconocidos.push_back(nombre);// Paso2
```

```
        }
        for(int i=0;i<numerodesconocidos;i++)
        cout<<"Hola, "<<desconocidos[i]<<"."<<"\n";// Paso3
        return 0;
}
```

Solución- Opción 2:

Paso1: Crear vector¡string¿desconocidos de tamaño 2*número de desconocidos

Paso2: Cambiar los valores del vector a los strings de entrada (”Soy” nombre), es decir en los números impares estarán los nombres

Paso3: Imprimir los nombres, es decir los strings en posiciones impares del vector

```
#include <vector>
#include <iostream>

using namespace std;

int main(){
        int numerodesconocidos;
        cin>>numerodesconocidos;
        vector<string>desconocidos(2*numerodesconocidos);//Paso1
        for(int i=0;i<2*numerodesconocidos;i++){
                cin>>desconocidos[i];//Paso2
        }
        for(int i=1;i<2*numerodesconocidos;i+=2)
        cout<<"Hola, "<<desconocidos[i]<<"."<<"\n";//Paso3
        return 0;
}
```

Ordenar un vector:

Se puede ordenar un vector. Hay una función llamada sort, que ordena los elementos de tu lista de menor a mayor.

```
#include <vector>
#include <iostream>
#include <algorithm>// bibloteca para el sort
using namespace std;

int main(){
    int n;
    cin>>n;
    vector<int>lista(n);
    for(int i=0;i<n;i++){
        cin>>lista[i]; // añadimos números a la lista
    }
    sort(lista.begin(),lista.end());// ordenamos la lista de menor a mayor
    for(int i=0;i<n;i++)cout<<lista[i]<<" ";//imprimimos la lista ordenada
    return 0;
}
```

Ordenación personalizada:

También podemos ordenar los vectores por otros criterios, escribiendo nuestra propia función bool micomparador. [sort(lista.begin(),lista.end(),micomparador)].

¿Qué hace la función micomparador?

A está se le dan dos elementos por ejemplo dos números número 1 y número 2. Devuelve TRUE si el número 1 debe ir antes del número 2 y FALSE si el número 2 tiene que ir antes del número 1.

Ejemplo de programa para ordenar números de mayor a menor:

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

bool micomparador(int elemento1,int elemto2){
    if(elemento1>elemto2)return true;
    else return false;
}

int main(){
    int n;
    cin>>n;
    vector<int>lista(n);
    for(int i=0;i<n;i++){
        cin>>lista[i];
    }
    sort(lista.begin(),lista.end(),micomparador);
    for(int i=0;i<n;i++)cout<<lista[i]<<" ";
    return 0;
}
```

Errores comunes:

Intentar acceder a un elemento del vector inexistente. Por ejemplo en el código de arriba si decimos lista[0]=1 nos dará error, ya que no existe dicho elemento lista[0]. En cambio si que podemos decir listade3[0]=1, ya listade3[0] es un elemento existente.