Vector

El tipo de variable vector permite almacenar varios valores bajo una sola variable y realizar operaciones complejas con esta. Todos los valores dentro del vector han de ser del mismo tipo (todas **int** o todas **string** o todas **char** o todas **bool**, ...).

Lo primero que se ha de hacer para poder utilizar vectores es añadir al principio del código esta línea:

```
#include <vector>
```

Una variable de tipo vector se declara así:

```
vector < tipo_de_los_valores > nombre_de_la_variable( tamanio );
```

Por ejemplo:

```
vector<int> puntos_equipos(15); // Esto declara un vector de 15 valores
```

Podemos dar un valor inicial a todas las posiciones del vector. Por ejemplo, inicialmente queremos que todos los equipos tengan 0 puntos porque la competición todavía no ha comenzado:

```
vector<int> puntos_equipos(15, 0);
```

Los valores se pueden leer y asignar igual que con los arrays:

```
puntos_equipos[0], puntos_equipos[1], ...
```

Los dos parámetros pueden ser variables:

```
int numEquipos = 15;
int puntosIniciales = 0;
vector<int> puntos_equipos(numEquipos, puntosIniciales);
```

Podemos mirar cuantos valores tiene este vector con su subsintrucción **.size()**:

```
cout << puntos_equipos.size();
```

Esto mostrará por pantalla “15”. No importa si todas las 15 posiciones del vector tienen valor o no, te dirá el tamaño total del vector.

Podemos añadir un elemento nuevo a un vector con su subinstrucción **.push_back(valor)**. Por ejemplo si ahora hacemos `puntos_equipos.push_back(10)` el vector pasaría a ser de 16 posiciones. **push_back()** siempre añade el valor al final del vector (al final de su tamaño, aunque tenga posiciones sin valor). Por lo tanto incluso podríamos crear un vector sin tamaño (tamaño 0) e ir añadiendo con **push_back()** uno a uno los elementos que leemos del cin:

```
vector<int> puntos_equipos;
int numEquipos; cin >> numEquipos;
while(numEquipos > 0) {
      int puntos; cin >> puntos;
      puntos_equipos.push_back(puntos);
      numEquipos = numEquipos - 1;
                        }
```

Vector ofrece muchas más subinstruccions como **pop_back()**, **insert()**, **erase()**, **sort()**, que pueden ser útiles en muchos momentos.

Cuándo quieres enviar como parámetro un vector a una función, a la función receptora se le debe poner un "&" antes del nombre del parámetro:

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

```
using namespace std;
void muestraPosicion(vector<int> &parametro, int index) {
                                                cout << parametro[index]; }
int main() {
      vector<int> lista;
      lista = { 6, 8, 7, 3, 2, 8, 9, 3 };
      int posicion;
      cin >> posicion;
      muestraPosicion(lista, posicion);
      }
```

Esto es porque normalmente cuando pasamos un parámetro lo que hace C++ realmente es hacer una copia del valor y pasar esta copia a la función. Pero con los vectores casi siempre lo que queremos es que la función receptora modifique el vector y nos lo devuelva con las modificaciones, por lo tanto no queremos que reciba una copia sino que queremos que reciba el original y trabaje directamente sobre nuestra variable.

Esto, evidentemente, también se puede hacer con cualquier otro tipo de variable como **int**, **string**, **char**, etc:

```
#include <iostream> using namespace std;
void dobla(int &n) { n = n * 2; }
int main() { int num; cin >> num; dobla(num); cout << num; }
```