Vectores en C++

Club de Programacion Competitiva Pu++

Abril del 2020

Los **vectores** en c++ son arreglos de **tamaño dinámico**.

Funcionan de manera muy parecida a los arreglos, tenemos acceso a alguno de sus elementos en tiempo constante (es decir la 'velocidad' de acceso no depende del numero de elementos en el vector) y también podemos tener vectores de distintos tipos de dato cono int, char, string, etc.

¿Dinámico?

Así es, no te tienes que preocupar por el tamaño de tu arreglo (vector mejor dicho), c++ se ocupa de eso. Para saber el tamaño de un vector ocupanos *size()*.

```
vector<char> v;  // definiendo el vector v
v.push_back('n');  // insertando al final
v.push_back('o');  // insertando al final
v.push_back('b');  // insertando al final

cout << v.size() << endl;  // 3
```

Igual que en los arreglos, accedemos a los elementos de acuerdo a su **índice**. Comenzando por el cero por supuesto.

```
vector<char> v = {'b', 'o', 'c', 'a',
    'j', 'u', 'n', 'i', 'o', 'r', 's'};

cout << v[0] << endl; // b
cout << v[2] << endl; // c
cout << v[0] << v[1] << v[3] << endl // boa
```

¿Podemos cambiar un elemento?

Claro. Continuando con nuestro ejemplo anterior:

```
v[2] = 'l';

for (int i=0; i<v.size(); i++) {  // notar v.size()
    cout << v[i];
}
cout << endl;

// se imprime:
// bolajuniors
```

Uff, joya pero ¿se puede quitar un elemento?

Sí. Claro quitemos el **último** elemento.

```
// quitando el elemento con indice n - 1
v.pop_back();

for (int i=0; i<v.size(); i++) {
    cout << v[i];
}
cout << endl;

// se imprime:
// bolajunior
```

Esa función solo quita el ultimo elemento, si se desea quitar un elemento distinto a este, se ocupa *erase* pero eso no se necesitará por ahora.

¿Vectores de vectores?

En efecto, se pueden declarar matrices con vectores.

```
// definiendo el tamaño desde su creación
// un vector de vectores de tamaño 2
// eso no define el tamaño de sus vectores 'hijo'
vector<vector<int>> mat(2);
for (int i=0; i<2; i++) {
    mat[i].push_back(8+i);
    mat[i].push_back(3+i);
}
```

La matriz queda:

8	3
9	4

Arreglo de vectores

Muy útiles.

```
const int MAXN = 1e3;  // definiendo una constante
// arreglo de vectores de int
vector<int> g[MXN];  // foreshadowing de gráficas

// agregandole un 20 al vector en la posición 2
g[2].push_back(20);

cout << g[2].back() << endl; // 20
```

La función *back()* regresa el valor del ultimo elemento en el vector.

¿V-vectores de arreglos?

Jaja sí owo... **Pero pasemos a otra cosa.** Sobrevivirás sin vectores de arreglos.

Valor default

El valor default de un vector de enteros es cero no importando dónde se defina. Si se desea llenar de un valor distinto se puede:

```
vector<int> v(3);
```


```
// Inicializando al vector con 1's
vector<int> v(3, 1);
```

Ordenando

```
vector<int> v = {3, 9, 0};
sort(v.begin(), v.end());
```

No te apures por *begin()* ni por *end()*, solo recuerda que eso necesitas para referenciar **todo** el vector en alguna función que te pida un rango.

Limpiando

A veces necesitamos limpiar el contenido de un vector para volver a llenarlo, como cuando un problema tiene muchos casos independientes.

```
vector<int> v;
for (int t=1; t<=T; t++) {
    // usamos el vector
    // ...

    // lo limpiamos
    v.clear();
}
```

Toma en cuenta que el tamaño del vector regresa a ser cero.

Más a fondo

Documentación de *vector*:

`https://en.cppreference.com/w/cpp/container/vector`