

Vectores

<https://en.cppreference.com/w/cpp/container/vector>

#include <vector>

Sintaxis:

```
vector<tipo> nombre;
```

Constructores



```
1  vector<tipo> v0; // {}  
2  vector<tipo> v1 = { 4, 3, 5, 1 };  
3  vector<tipo> v2(tamano); // { 0, 0, ... }  
4  vector<tipo> v3(tamano, -1); // { -1, -1, ... }
```

Librería útil

<https://raw.githubusercontent.com/louisdx/cxx-prettyprint/master/prettyprint.hpp>

#include “prettyprint.hpp”

Te permite hacer *cout* a diferentes estructuras, entre ellas está vectores:



```
1  #include <iostream>
2  #include <vector>
3  #include "prettyprint.hpp"
4
5  using namespace std;
6
7  int main(int argc, char const *argv[]) {
8      vector<int> numeros = { 4, 3, 5, 1 };
9
10     cout << numeros << endl; // [4, 3, 5, 1]
11
12     return 0;
13 }
```

Métodos útiles



at(indice)


Sirve para acceder a un elemento en la posición indicada pero da error si la posición es inválida:



```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  cout << numeros.at(2) << endl; // 5  
4  cout << numeros.at(-1) << endl; // Da error
```


push_back(elemento)

AGREGA el elemento dado al **FINAL** del vector:



```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  numeros.push_back(10); // { 4, 3, 5, 1, 10 }
```

pop_back()

ELIMINA el ÚLTIMO elemento:



```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  numeros.pop_back(); // { 4, 3, 5 }
```

size()

Retorna el **TAMAÑO** del vector:



```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  cout << numeros.size() << endl; // 4
```

clear()


ELIMINA TODOS los elementos del vector:



```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  numeros.clear(); // {}  
4  cout << numeros.size() << endl; // 0
```

empty()

Retorna **true** si el vector está vacío y **false** caso contrario:



```
1  vector<int> numeros = { 4, 3, 5, 1 };
2
3  cout << numeros.empty() << endl; // 0: false
4  numeros.clear();
5  cout << numeros.empty() << endl; // 1: true
```

emplace_back(elemento)

Puede servir más adelante:

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

Iteradores



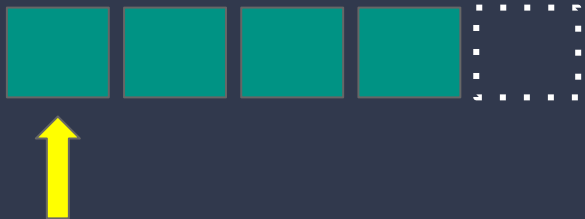
¿Qué es un iterador?

En términos simples, es un **PUNTERO**.

Se usa para apuntar a la dirección de memoria de un ***container*** de la librería **STD**.

Para acceder a su contenido se usa *****.

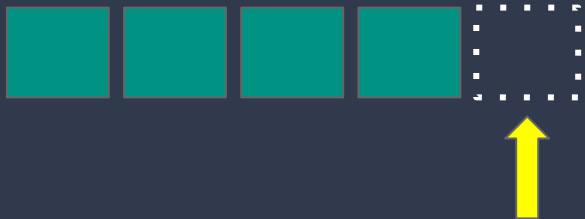
begin()



Retorna un iterador que apunta al inicio.

```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  auto it = numeros.begin();  
4  cout << it << endl; // error  
5  cout << *it << endl; // 4
```

end()



AGREGA el elemento dado en la **POSICIÓN** solicitada:



```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  auto it = numeros.end();  
4  cout << *it << endl; // Puede dar 0 o error  
5  cout << *(it - 1) << endl; // 1, ultimo elemento
```

Puedes mover el iterador



```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  auto it = numeros.begin();  
4  cout << *it << endl; // 4  
5  cout << *(it + 1) << endl; // 3  
6  cout << *(it + 2) << endl; // 5  
7  cout << *(it + 3) << endl; // 1
```

insert(iterador, elemento)

AGREGA el elemento dado en la **POSICIÓN** solicitada:



```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  numeros.insert(numeros.begin() + 1, 10);  
4  // { 4, 10, 3, 5, 1 }
```

erase(iterator)

ELIMINA el elemento dado en la **POSICIÓN** solicitada:



```
1  vector<int> numeros = { 4, 3, 5, 1 };  
2  
3  numeros.erase(numeros.begin() + 2);  
4  // { 4, 3, 1 }
```