Vectores

Los vectores en C++ son una parte fundamental de la STL (Standard Template Library) y ofrecen una manera flexible de manejar colecciones de datos de tamaño dinámico. A continuación, te explicaré cómo trabajar con vectores en C++, incluyendo su creación, adición de elementos, eliminación, modificación, y más.

1. ¿Qué es un Vector en C++?

Un **vector** es una estructura de datos que se asemeja a un arreglo dinámico. A diferencia de los arreglos estáticos, los vectores pueden cambiar de tamaño automáticamente cuando se añaden o eliminan elementos.

2. Creación de un Vector

Para usar vectores, debes incluir el encabezado <vector> en tu programa:

```
#include <vector>
```

Declaración de un Vector:

```
std::vector<int> numeros; // Un vector de enteros
```

Este código crea un vector numeros que puede almacenar elementos de tipo int.

3. Añadir Elementos a un Vector

Puedes añadir elementos a un vector usando el método push_back().

Ejemplo:

```
numeros.push_back(10);
numeros.push_back(20);
numeros.push_back(30);
```

Esto añade los valores 10, 20, y 30 al final del vector numeros.

4. Acceso y Modificación de Elementos

Puedes acceder y modificar los elementos de un vector utilizando el operador de índice [] o el método at().

Acceso:

```
int primero = numeros[0]; // Accede al primer elemento (10)
int segundo = numeros.at(1); // Accede al segundo elemento (20)
```

Modificación:

```
numeros[0] = 100; // Cambia el primer elemento a 100
numeros.at(1) = 200; // Cambia el segundo elemento a 200
```

5. Eliminar Elementos de un Vector

Los vectores ofrecen varios métodos para eliminar elementos.

Eliminar el Último Elemento:

```
numeros.pop_back(); // Elimina el último elemento
```

Eliminar un Elemento en una Posición Específica:

```
numeros.erase(numeros.begin() + 1); // Elimina el segundo elemento
```

numeros.begin() devuelve un iterador al primer elemento, por lo que numeros.begin() + 1 es un iterador al segundo elemento.

Eliminar Todos los Elementos:

```
numeros.clear(); // Vacía el vector, eliminando todos los elementos
```

6. Tamaño y Capacidad de un Vector

Los vectores tienen métodos para consultar y modificar su tamaño y capacidad.

Obtener el Tamaño del Vector:

```
int tamano = numeros.size(); // Devuelve el número de elementos en el vector
```

Redimensionar un Vector:

```
numeros.resize(5); // Cambia el tamaño del vector a 5 elementos
```

Si el nuevo tamaño es mayor, se añaden elementos con el valor predeterminado (0 para int).
Si es menor, se eliminan los elementos adicionales.

7. Iterar Sobre un Vector
Puedes utilizar bucles para iterar sobre los elementos de un vector.

Uso de un Bucle for:

```
for (int i = 0; i < numeros.size(); i++) {
        std::cout << numeros[i] << " ";
}
```

Uso de un Bucle for de Rango:

```
for (int num : numeros) {
        std::cout << num << " ";
}
```

8. Insertar Elementos en una Posición Específica
Puedes insertar elementos en cualquier posición del vector utilizando el método insert().

Ejemplo:

```
numeros.insert(numeros.begin() + 1, 50); // Inserta 50 en la segunda posición
```

9. Verificar si un Vector está Vacío
Puedes comprobar si un vector está vacío utilizando el método empty().

Ejemplo:

```
if (numeros.empty()) {
        std::cout << "El vector está vacío" << std::endl;
}
```

10. Copiar y Asignar Vectores
Puedes copiar vectores simplemente asignándolos.

Ejemplo:

```
std::vector<int> copia = numeros; // Crea una copia del vector numeros
```

Resumen:

Creación: std::vector<int> v;
Añadir elementos: v.push_back(valor);
Acceso/modificación: v[indice] o v.at(indice);
Eliminar elementos: v.pop_back();, v.erase(it);, v.clear();
Redimensionar: v.resize(nuevo_tamano);
Insertar: v.insert(it, valor);
Iterar: for (auto& elem : v) { ... }