Manual 2 - 2do Torneo de Programación Competitiva

Lions R.C.

Junio 2019

Contents

1 Estructuras de datos sencillos

Muchas veces, nos encontramos en medio de un problema que ocupa el uso de algo mas flexible que un arreglo. Si queremos borrar datos de un arreglo, tendriamos que desplazar todos los demás datos que estan enfrente hacia atras, y nuestros resultados se vuelven mas ineficientes o más complicados.

Una solución es utilizar otros estructuras de datos, que almacenan y manejan datos de distintas maneras. En esta sección, solo se explicaran algunas de las muchas estructuras de datos, y la siguiente semana se darán a conocer los demás.

1.1 Vectores

Un vector no es nada más que un arreglo dinámico. Esto significa que el vector no tiene un tamaño fijo y puedes agregar y quitar elementos sin problema.

Para crear un vector, es necesario agregar una libreria especifico a esta estructura, llamado **vector**.

Para incluir esta libreria, se debe de escribir **#include <vector>** en las primeras lineas de tu programa.

Listing 1: Vectores

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

using namespace std;

int main() {

}
```

Como un vector se parece a un arreglo, debes definir el tipo de dato que se almacenará a la hora de declararlo. Se puede declarar el vector con el sintaxis **vector<dato> nombre;** Como se puede observar, el vector no requiere que le des un tamaño predeterminado.

Listing 2: Vectores

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

using namespace std;

int main() {
    vector<int> enteros;
}
```

Aqui se ha declarado un vector de enteros, y para agregarle elementos a este vector, se debe escribir el nombre del vector seguido por un punto y la función **push_back()** con el valor del entero entre los paréntesis.

Listing 3: Agregando valores

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

using namespace std;

int main() {
    vector<int> enteros;
    enteros.push_back(1);
    enteros.push_back(4);
    enteros.push_back(9);
}
```

Aqui, nuestro vector tendrá los valores de 1, 4 y 9 guardados.

Para ver o modificar el valor en algún indice, se puede utilizar el mismo sintaxis de un arreglo. Si en el ejemplo queremos cambiar el 9 a 7, podemos modificarlo y ver sus cambios con el siguiente código:

Listing 4: Modificando valores

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

using namespace std;

int main() {
    vector<int> enteros;
    enteros.push_back(1);
    enteros.push_back(4);
    enteros.push_back(9);
    for(int i = 0; i < 3; i++) {
        cout << enteros[i] << endl;
    }
    enteros[2] = 7;
```

```
    for(int i = 0; i < 3; i++) {
        cout << enteros[i] << endl;
    }
}
```

Primero se imprimiran los valores de 1, 4 y 9, luego se verán los valores 1, 4 y 7 en la consola.

Existen mas funciones de los vectores que son utiles, como **insert ()** que inserta elementos en ciertos indices, **erase** que elimina ciertos elementos, **clear ()** que borra todos los datos en un vector, **pop_back()** que elimina el último valor y finalmente **size ()**, que indica el tamaño de un vector.

Listing 5: Jugando con vectores

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

using namespace std;

int main() {
    vector<int> enteros;
    for(int i = 0; i < 10; i++) {
        enteros.push_back(i);
    }
    enteros.insert(enteros.begin() + 5, 25);
    enteros.erase(enteros.begin() + 7);
    cout << "Cantidad: " << enteros.size() << endl;
    for(int i = 0; i < enteros.size(); i++) {
        cout << enteros[i] << endl;
    }
    enteros.clear();
    cout << "Cantidad: " << enteros.size() << endl;
}
```

Liga al código

Como se puede ver en el código de arriba, se crea un vector de enteros y se llena de los valores de 0 a 9, luego se inserta el valor 25 en el índice 5 y se elimina el valor en el índice 7. Luego se imprime el tamaño del arreglo (12 en ese momento), se imprimen todos los valores, se limpia el vector y finalmente se imprime el tamaño final (cero porque se limpió).

Se debe notar que para las funciones **insert** y **erase**, se ocupa llamar a la función begin para ese vector y luego se debe sumar el índice a ese valor. Esta suma luego determina el lugar en la memoria donde esta guardado el valor con ese índice.