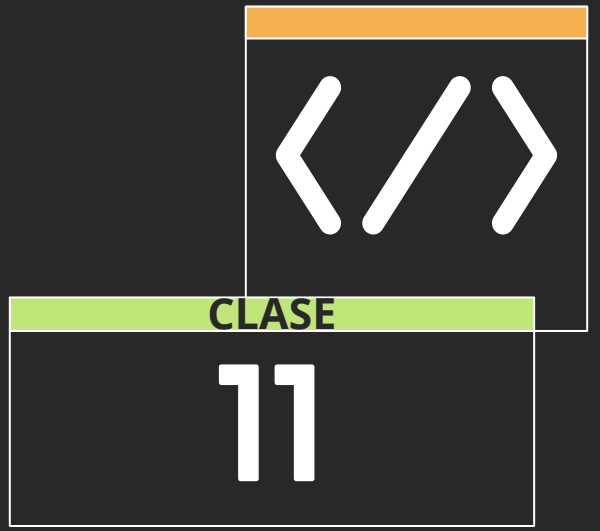
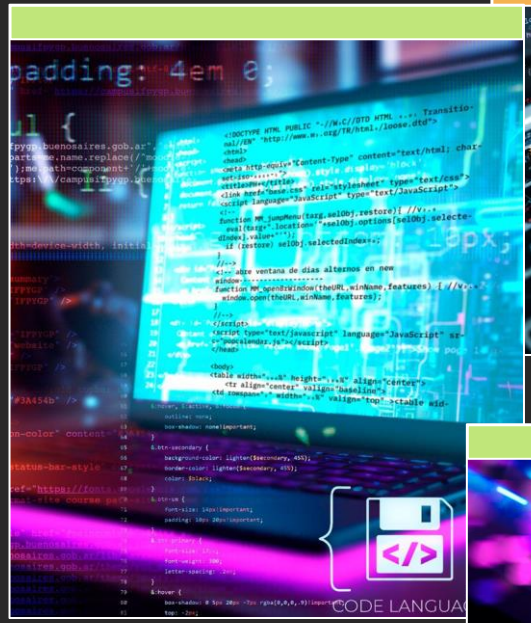
Programacion II [PRACTICA]



1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0

1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0

1010101100011110



Algoritmos

1100101001011011110

Algoritmos

Los algoritmos en la STL son funciones que operan sobre contenedores, proporcionando operaciones útiles como búsqueda, ordenación, modificación y más.

- std::sort: Ordena los elementos de un contenedor.
- std::find: Busca un elemento en un rango de un contenedor.
- std::reverse: Invierte el orden de los elementos en un rango.
- std::count: Cuenta el número de ocurrencias de un valor en un rango.
- std::accumulate (de la librería <numeric>): Suma los elementos de un rango.
- std::for_each: Aplica una función a cada elemento en un rango.
- std::remove_if: Elimina elementos que cumplen una condición específica.
- std::transform: Aplica una función a un rango y guarda el resultado en otro rango

Algoritmos

```
// Ordena el vector `vec` en orden ascendente
std::sort(vec.begin(), vec.end());

// Busca `valor` en el vector `vec`
std::find(vec.begin(), vec.end(), valor);

// Invierte la lista `list`
std::reverse(list.begin(), list.end());

// Cuenta las ocurrencias de `valor` en `vec`
std::count(vec.begin(), vec.end(), valor);

// Suma todos los elementos de `vec`, comenzando desde 0
std::accumulate(vec.begin(), vec.end(), 0);
```

Algoritmos

```
// Multiplica cada elemento de `vec` por 2
void multiplicarPorDos(int& n)
{    n *= 2;
}
std::for_each(vec.begin(), vec.end(), multiplicarPorDos);
// Elimina números pares de `vec`
bool esPar(int n)
{    return n % 2 == 0;
}
std::remove_if(vec.begin(), vec.end(), esPar);
// Eleva al cuadrado cada elemento de `vec`
int elevarAlCuadrado(int n)
{    return n * n;
}
std::transform(vec.begin(), vec.end(), vec.begin(), elevarAlCuadrado);
```

Retos de Programacion

01 Gestión de tareas con std::vector

Crear una pequeña aplicación de lista de tareas (to-do list) usando std::vector para almacenar las tareas ingresadas. La interfaz debe permitir agregar y eliminar tareas.

1001010100101010

Instrucciones:

- Usa un QListWidget para mostrar las tareas.
- Al agregar una tarea con un QLineEdit y botón "Agregar", esta se añade al std::vector.
- Al seleccionar una tarea y presionar "Eliminar", se elimina del std::vector y se actualiza la lista.

1010010010110111

02 Ordenar y buscar el mayor número en un vector

Crea una aplicación que permita al usuario ingresar números en un vector, ordenarlos, y luego mostrar el número mayor después de ordenar.

Frontend (QtCreator):

- ✓ 1 QLineEdit para ingresar un número.
- ✓ 1 QPushButton con el texto "Agregar al Vector«.
- ✓ 1 QPushButton con el texto "Ordenar y Mostrar Mayor".
- ✓ 1 QLabel para mostrar el número más alto.
- ✓ 1 QLabel para mostrar el vector ordenado.

Lógica en C++:

Al presionar "Agregar al Vector", los números se agregan a un std::vector<int>. Al presionar "Ordenar y Mostrar Mayor", el vector se ordena usando std::sort, se muestra en una etiqueta, y el número mayor (último elemento del vector) se muestra en otra etiqueta.

03 Buscar varios valores en un vector

Permite al usuario agregar una lista de números y luego buscar múltiples valores en el vector. Los resultados de las búsquedas se muestran en un área de texto.

Frontend (QtCreator):

- ✓ 1 QTextEdit para mostrar los números ingresados.
- ✓ 1 QLineEdit para ingresar un número.
- ✓ 1 QPushButton con el texto "Agregar al Vector".
- ✓ 1 QLineEdit para ingresar un número a buscar.
- ✓ 1 QPushButton con el texto "Buscar en el Vector".
- ✓ 1 QLabel para mostrar el resultado de la búsqueda (si fue encontrado o no).

Lógica en C++:

El usuario agrega números al vector, que se muestran en el QTextEdit. Al buscar un número, se utiliza std::find para verificar si está presente, y se muestra el resultado en la etiqueta.

04 Invertir una lista de palabras

El usuario ingresa varias palabras en una lista, y luego puede invertir el orden de las palabras y mostrar la lista invertida.

1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0

Frontend (QtCreator):

- ✓ 1 QLineEdit para ingresar palabras.
- ✓ 1 QPushButton con el texto "Agregar Palabra".
- ✓ 1 QPushButton con el texto "Invertir Lista".
- ✓ 1 QTextEdit para mostrar la lista invertida.

Lógica en C++:

El usuario ingresa palabras en una lista (std::list<std::string>). Al hacer clic en "Invertir Lista", se utiliza std::reverse para invertir el orden de las palabras y se muestra el resultado en un QTextEdit.

1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1

05 Contar ocurrencias de palabras en una lista

El usuario puede agregar palabras a una lista y luego buscar cuántas veces una palabra específica aparece en la lista.

1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0

Frontend (QtCreator):

- ✓ 1 QLineEdit para ingresar una palabra.
- ✓ 1 QPushButton con el texto "Agregar Palabra".
- ✓ 1 QLineEdit para ingresar la palabra a contar.
- ✓ 1 QPushButton con el texto "Contar Ocurrencias".
- ✓ 1 QLabel para mostrar el número de ocurrencias.

1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1

Lógica en C++:

Se utiliza std::count para contar cuántas veces una palabra específica aparece en la lista de palabras agregadas por el usuario.