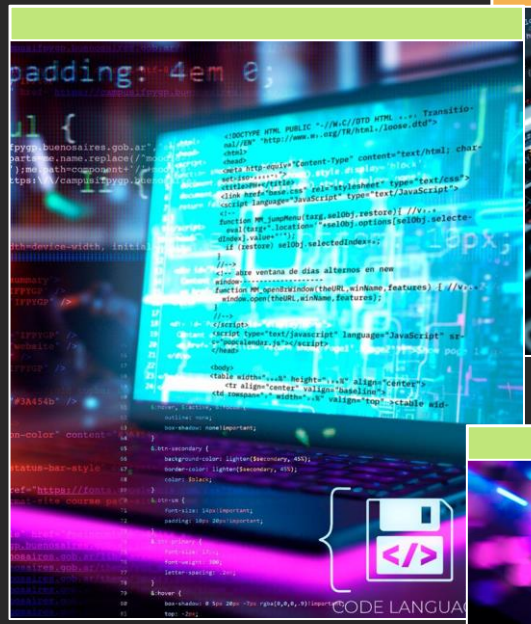
Programacion II [PRACTICA]

1 0 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

Biblioteca de Plantillas Estándar (S.T.L)



1010101100011110

1100101010101110

Plantillas

Una plantilla es una construcción que genera un tipo o función normal en tiempo de compilación en función de los argumentos que proporciona el usuario para los parámetros de la plantilla.

```cpp
template <typename T>
T max(T a, T b) {
    return (a > b) ? a : b;
}

int main() {
    int x = 10, y = 20;
    std::cout << "Mayor: " << max(x, y) << std::endl;

    double a = 5.5, b = 3.7;
    std::cout << "Mayor: " << max(a, b) << std::endl;

    return 0;
}
```

La función max es genérica y puede aceptar cualquier tipo de dato (int, double, etc.) al usar la palabra clave template.

Contenedores

Los contenedores son estructuras de datos que se utilizan para almacenar objetos y datos según los requisitos. Cada contenedor se implementa como una clase de plantilla que también contiene los métodos para realizar operaciones básicas en él. Cada contenedor STL se define dentro de su propio archivo de encabezado.

Contenedores de secuencia:
array

- vector
- deque
- List
- forward_list

Contenedores

Array

```cpp
std::array<int, 5> arr = {1, 2, 3, 4, 5};
for (int i : arr) std::cout << i << " ";
```

- std::array<int, 5>: Es una estructura de datos estática que almacena elementos en una secuencia continua de memoria. Aquí, hemos declarado un array de 5 enteros.
- for (int i : arr): Este es un ciclo "range-based for", que recorre cada elemento del array y lo asigna a la variable i.
- std::cout << i << " ": Imprime cada elemento del array seguido de un espacio.Este ejemplo inicializa el array con los valores {1, 2, 3, 4, 5} y luego los imprime uno por uno.

Vector

```cpp
std::vector<int> vec = {1, 2, 3, 4, 5};
vec.push_back(6);
for (int i : vec) std::cout << i << " ";
```

- std::vector<int>: Es un contenedor dinámico que puede redimensionarse automáticamente cuando se agregan o eliminan elementos. Al igual que el array, almacena elementos de tipo int en una secuencia continua de memoria.
- vec.push_back(6): Añade el valor 6 al final del vector. El vector ahora contiene {1, 2, 3, 4, 5, 6}.
- for (int i : vec): Un ciclo "range-based for" que recorre el vector y asigna cada valor a la variable i.
- std::cout << i << " ": Imprime los valores del vector uno por uno.

Contenedores

Deque (double-ended queue)

```cpp
std::deque<int> dq = {1, 2, 3};
dq.push_front(0);
dq.push_back(4);
for (int i : dq) std::cout << i << " ";
```

- std::deque<int>: Es un contenedor dinámico que permite la inserción y eliminación de elementos en ambos extremos (frontal y posterior).
- dq.push_front(0): Añade el valor 0 al frente del deque. Ahora el deque contiene {0, 1, 2, 3}.
- dq.push_back(4): Añade el valor 4 al final del deque. El deque contiene ahora {0, 1, 2, 3, 4}.
- for (int i : dq): Recorre el deque e imprime sus elementos.

List (doble lista enlazada):

```cpp
std::list<int> lst = {1, 2, 3, 4};
lst.push_back(5);
lst.push_front(0);
for (int i : lst) std::cout << i << " ";
```

- std::list<int>: Es una lista enlazada doble, lo que significa que cada elemento apunta tanto al siguiente como al anterior. Es eficiente para inserciones y eliminaciones en cualquier parte de la lista, pero no permite acceso aleatorio (a diferencia de vector o deque).
- lst.push_back(5): Añade el valor 5 al final de la lista. Ahora la lista contiene {1, 2, 3, 4, 5}.
- lst.push_front(0): Añade el valor 0 al principio de la lista. La lista contiene ahora {0, 1, 2, 3, 4, 5}.
- for (int i : lst): Recorre la lista e imprime cada elemento.

Contenedores

Forward_list (lista enlazada simple)

```
1  std::forward_list<int> fl = {1, 2, 3};
2  fl.push_front(0);
3  for (int i : fl) std::cout << i << " ";
```

- std::forward_list<int>: Es una lista enlazada simple, donde cada elemento solo apunta al siguiente. Es más ligera que std::list pero solo permite inserción eficiente al frente y no admite acceso aleatorio ni inserción en el medio de la lista de manera directa.
- fl.push_front(0): Añade el valor 0 al frente de la lista. La lista ahora contiene {0, 1, 2, 3}.
- for (int i : fl): Recorre la lista enlazada simple e imprime sus elementos.

Stack (pila)

```
1  std::stack<int> stk;
2  stk.push(10);
3  stk.push(20);
4  std::cout << stk.top();  // Muestra el tope (20)
```

- std::stack<int>: Es una estructura LIFO (Last In, First Out), lo que significa que el último elemento añadido es el primero en salir.
- stk.push(10): Añade 10 a la pila.
- stk.push(20): Añade 20 a la pila, que ahora contiene {10, 20}.
- stk.top(): Devuelve el elemento en la parte superior de la pila, que es 20.

Contenedores

Queue (cola)

```cpp
std::queue<int> q;
q.push(1);
q.push(2);
std::cout << q.front();  // Muestra el primer elemento (1)
```

- std::queue<int>: Es una estructura FIFO (First In, First Out), lo que significa que el primer elemento que entra es el primero que sale.
- q.push(1): Añade 1 a la cola.
- q.push(2): Añade 2 a la cola, que ahora contiene {1, 2}.
- q.front(): Devuelve el primer elemento de la cola, que es

Priority_queue (cola de prioridad)

```cpp
std::priority_queue<int> pq;
pq.push(10);
pq.push(5);
pq.push(20);
std::cout << pq.top();  // Muestra el elemento mayor (20)
```

- std::priority_queue<int>: Es una cola donde los elementos se organizan automáticamente de acuerdo con su prioridad. Por defecto, la prioridad es que el valor más alto se encuentre al principio.
- pq.push(10), pq.push(5), pq.push(20): Añade los valores a la cola de prioridad. Internamente, los elementos se ordenan de forma que el valor más alto (20) tenga la prioridad más alta.
- pq.top(): Devuelve el valor con mayor prioridad, que en este caso es 20.

Retos de Programacion

01 Ejercicio de array

Crea un programa con un formulario que permita al usuario ingresar 5 números en un array, luego mostrarlos en un QLabel cuando se presione un botón.

Frontend (QtCreator):

- ✓ 5 QLineEdit para ingresar números.
- ✓ 1 QPushButton con el texto "Mostrar Array".
- ✓ 1 QLabel para mostrar el contenido del array

Lógica en C++:

Al hacer clic en el botón, el programa debe tomar los valores de las cajas de texto, almacenarlos en un std::array<int, 5>, y luego mostrarlos en la etiqueta.

02 Ejercicio de vector

Crea un programa que permita al usuario agregar números a un vector dinámico y mostrar los números almacenados.

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

Frontend (QtCreator):

- ✓ 1 QLineEdit para ingresar un número.
- ✓ 1 QPushButton con el texto "Agregar al Vector".
- ✓ 1 QPushButton con el texto "Mostrar Vector".
- ✓ 1 QLabel para mostrar los elementos del vector.

Lógica en C++:

El primer botón agregará el número ingresado al vector, y el segundo botón mostrará el contenido del vector en la etiqueta

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

03 Ejercicio de deque

Crea un programa donde el usuario pueda agregar números al frente o al final de un deque, y luego mostrarlos en la pantalla.

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

Frontend (QtCreator):

- ✓ 1 QLineEdit para ingresar un número.
- ✓ 2 QPushButton ("Agregar al Frente" y "Agregar al Final").
- ✓ 1 QPushButton para mostrar el deque.
- ✓ 1 QLabel para mostrar el contenido.

Lógica en C++:

El usuario podrá agregar números al frente o al final del deque usando los botones, y mostrarlos en el QLabel al hacer clic en "Mostrar deque".

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

04 Ejercicio de list

Permite al usuario agregar números al principio o al final de una lista y mostrarlos.

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

Frontend (QtCreator):

- ✓ 1 QLineEdit para ingresar un número.
- ✓ 2 QPushButton ("Agregar al Frente" y "Agregar al Final").
- ✓ 1 QPushButton para mostrar la lista.
- ✓ 1 QLabel para mostrar el contenido de la lista.

Lógica en C++:
Al presionar "Agregar al Frente" o "Agregar al Final", el número se insertará en la lista. El botón "Mostrar Lista" mostrará todos los números en el QLabel.

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

05 Ejercicio de stack

Crea un programa donde el usuario pueda apilar números en una pila (stack) y luego ver cuál es el número en la cima.

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

Frontend (QtCreator):

- ✓ 1 QLineEdit para ingresar un número.
- ✓ 1 QPushButton con el texto "Apilar Número".
- ✓ 1 QPushButton con el texto "Ver Cima".
- ✓ 1 QLabel para mostrar el número en la cima de la pila.

Lógica en C++:

El botón "Apilar Número" añadirá un número a la pila, y "Ver Cima" mostrará el valor más alto de la pila en el QLabel.

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