

Instructor: Ernesto Enrique García Ramos

Contacto: egarcia97.r@gmail.com

gr15i04001@usonsonate.edu.sv

## Guía 5: Librería list

C++ está diseñado para la programación orientada a objetos (POO), y en este paradigma, todas las entidades que podemos manejar son objetos.

### Objetivos:

- Conocer e implementar librería list

La librería list es una clase template (plantilla) en la biblioteca estándar C++. Podemos crear listas que contengan cualquier tipo de objetos.

List contiene muchas operaciones, incluyendo: **push\_back()**, **push\_front**, **pop\_front**, **pop\_back()**, **begin()**, **end()**, **size()**, y **empty()**

### Ejemplo de sintaxis:

```
#include <iostream>
#include <list>

using namespace std;

int main()
{
    list<string> ejemplo;

    ejemplo.push_back("Solis");
    ejemplo.push_back("Garcia");
    ejemplo.push_back("Ramos");
    ejemplo.push_front("Peniate");
    cout<<ejemplo.size()<<endl;

    ejemplo.pop_back();
    cout<<ejemplo.size();

    return 0;
}
```

### Iteradores:

Un iterador (iterator) es un puntero que se puede mover a través de la lista y provee acceso a elementos individuales.

El operador referencia (\*) es usado cuando necesitamos obtener o fijar el valor de un elemento de la lista.

### Ejemplo de sintaxis:

```
//Declarando un iterador
list<string>::iterator pos;
```

# Programación 3

Facultad de Ingeniería y Ciencias Naturales

```
pos = ejemplo.begin();  
cout << *pos << endl;           // "Peniate"  
  
*pos = "Ernesto";                //Reemplazo lo que contiene  
  
pos = ejemplo.begin();  
cout << *pos << endl;           // "Ernesto"
```

<Ejemplo> En una lista de elementos aleatorios especifica qué tipo de datos recibirá.

Podemos usar los operadores ++ y -- para manipular iteradores. El siguiente código recorre la lista y despliega los ítems usando un iterador

```
//Recorriendo la lista  
list<string>::iterator pos;  
pos=ejemplo.begin();  
while( pos != ejemplo.end())  
{  
    cout << *pos << endl;  
    pos++;  
}
```

## Eliminación

La función miembro erase() remueve el nodo de la posición del iterador. El iterador es no válido después de la operación

### Ejemplo de sintaxis:

```
list<string>::iterator pos=ejemplo.begin();  
ejemplo.erase(pos);  
  
list<string>::iterator prueba=ejemplo.begin();  
while( prueba != ejemplo.end())  
{  
    cout << *prueba << endl;  
    prueba++;  
}
```

Existen otros métodos para la eliminación

- **pop\_back** borra el último elemento de la lista
- **pop\_front** borra el primer elemento de la lista
- **erase** borra uno o más elementos de la lista
- **remove** borra un elemento de la lista
- **clear** borra todos los elementos de la lista

## Ejercicio

Implementar librería list

