
	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 1

INFORME DE LABORATORIO

INFORMACION BASICA					
ASIGNATURA:	Tecnología de Objetos				
TITULO DE LA PRACTICA:	Templates en C++				
NUMERO DE LA PRACTICA:	06	AÑO LECTIVO:	2025 - B	N° SEMESTRE:	VI
FECHA DE PRESENTACION:	25 / 10 / 2025	HORA DE PRESENTACION:	--:-- PM		
INTEGRANTE (s): ■ Huayhua Hillpa, Yourdyy Yossimar				NOTA:	
DOCENTE (s): ■ Mg. Escobedo Quispe, Richart Smith					



1. Tarea

1.1. Problema propuesto:

- [1] Usar clase template para implementar estructuras de lista enlazada simple que permita armar secuencia de edades secuencia de palabras.
- [2] Transcribir y analizar el siguiente código, describir el comportamiento.
- [3] Crear una interface gráfica (que implemente señales y slots) que muestre una lista de países, al dar clic sobre alguno que se muestre un Label o Text con el idioma y capital correspondiente.

2. Equipos, materiales y temas utilizados

- Subsistema de Windows para Linux (WSL) con Ubuntu.
- Sistema operativo: Microsoft Windows [Versión 10.0.26100.6584]
- TeX Live 2025
- Helix 25.01.1 (e7ac2fcd)
- Visual Studio Code 1.104.0 x64
- Git version 2.41.0.windows.1
- Cuenta activa en GitHub para la gestión de repositorios remotos.
- Plantillas
- Leguaje de programación C++

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 2</p>

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/yhuayhuahi/Teo.git>
- URL para el laboratorio (06) en el Repositorio GitHub.
- <https://github.com/yhuayhuahi/Teo/tree/main/laboratorios/lab06>

4. Desarrollo de las actividades

4.1. Actividad 01: Implementación en C++ para una lista enlazada simple

4.1.1. Función Main en C++

A continuación se muestra la implementación de la función main en C++ para probar la clase template de una lista enlazada simple que permite armar secuencias de enteros para este caso.

Listing 1: Función Main en cpp



```

1  #include <iostream>
2  #include "cola.h"
3
4  using namespace std;
5
6  int main() {
7      // Probar con enteros
8      Cola<int> cola;
9
10     cola.insertarValor(10);
11     cola.insertarValor(20);
12     cola.insertarValor(30);
13
14     cout << "Cantidad: " << cola.accederCantidad() << endl;
15     cola.mostrarCola();
16
17     // Buscar y eliminar
18     cola.eliminarValor(20);
19     cout << "Después de eliminar: " << cola.accederCantidad() << endl;
20     cola.mostrarCola();
21
22     return 0;
23 }
```

La implementación de la clase template Cola se encuentra en el archivo cola.h que se adjunta en el repositorio del laboratorio.

4.1.2. Prueba de ejecución

Se realizó la prueba de ejecución del código en C++ y se obtuvo el siguiente resultado:

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 3

```
WSL at ~ / ... /laboratorios/lab06/ejer01
17:33:54 g++ main.cpp -o main
WSL at ~ / ... /laboratorios/lab06/ejer01
17:34:08 ./main
Cantidad: 3
Cola: 10 → 20 → 30 → NULL
Después de eliminar: 2
Cola: 10 → 30 → NULL
WSL at ~ / ... /laboratorios/lab06/ejer01
```

Figura 1: Prueba de ejecución de la lista enlazada simple

4.2. Actividad 02



Se tiene el siguiente código en C++, brindado en la guía de laboratorio:

Listing 2: Código en C++ brindado en la guía de laboratorio

```
1 #include <iostream>
2
3 using namespace std;
4
5 template <class T>
6 class Contenedor {
7     T elemento;
8 public:
9     Contenedor(T arg) { elemento = arg; }
10    T add() { return ++elemento; }
11 };
12
13 template <>
14 class Contenedor <char> {
15     char elemento;
16 public:
17     Contenedor(char arg) { elemento = arg; }
18     char uppercase ()
19     {
20         if ((elemento >= 'a') && (elemento <= 'z')) {
21             elemento += 'A' - 'a';
22         }
23         return elemento;
24     }
25 };
26
27 int main() {
28     Contenedor<int> cint(5);
29     Contenedor<char> cchar('t');
30     std::cout << cint.add() << std::endl;
31     std::cout << cchar.uppercase() << std::endl;
32     return 0;
33 }
```

4.2.1. Análisis del código

El código presentado define una clase plantilla 'Contenedor' que puede almacenar un elemento de cualquier tipo. Se proporciona una especialización para el tipo 'char' que incluye un método para

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 4</p>

convertir el carácter a mayúsculas. En la función 'main', se crean instancias de 'Contenedor' para un entero y un carácter, y se muestran los resultados de las operaciones.

Al ejecutar el código, se observa que la instancia de 'Contenedor<int>' incrementa el valor entero de 5 a 6, mientras que la instancia de 'Contenedor<char>' convierte el carácter 't' a su equivalente en mayúscula 'T'. Por lo tanto, la salida del programa será:

```
WSL at ~ / ... /laboratorios/lab06/ejer02
08:27:14 g++ contenedor.cpp -o contenedor
WSL at ~ / ... /laboratorios/lab06/ejer02
08:27:28 ./contenedor
6
T
WSL at ~ / ... /laboratorios/lab06/ejer02
08:27:33
```

Figura 2: Salida del programa al ejecutar el código proporcionado



Esto demuestra el uso de plantillas en C++ para crear clases genéricas y la capacidad de especializar comportamientos para tipos específicos.

4.3. Actividad 03

El siguiente código proporcionado en la guía de laboratorio implementa una clase template con parámetros por defecto en C++:

Listing 3: Código en C++ con parámetros por defecto

```
1 #include <iostream>
2
3 using namespace std;
4
5 template <class T = char, int N = 8>
6 class Comun1 {
7     T bloque[N];
8 public:
9     void set(int num, T tval);
10    T get(int num);
11 };
12
13 template <class T, int N>
14 void Comun1<T, N>::set(int num, T tval) {
15     bloque[num] = tval;
16 }
17
18 template <class T, int N>
19 T Comun1<T, N>::get(int num) {
20     return bloque[num];
21 }
22
23 int main() {
24     Comun1 <int, 5> objInt;
25     Comun1 <double, 5> objFloat;
26     Comun1 <> obj;
27     objInt.set(0, 10);
28     objFloat.set(2, 3.1);
```

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 5</p>

```

29
30     std::cout << objInt.get(0) << std::endl;
31     std::cout << objFloat.get(2) << std::endl;
32     std::cout << obj.get(4) << std::endl;
33
34     return 0;
35 }
```

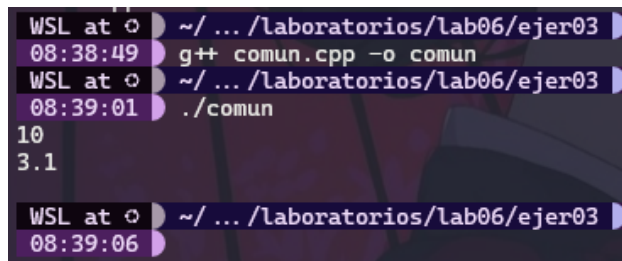
4.3.1. Análisis del código

El código define una clase plantilla 'Comun1' que utiliza parámetros de plantilla con valores pre-determinados: 'T' (tipo de dato) y 'N' (tamaño del bloque). La clase contiene un arreglo 'bloque' de tamaño 'N' y proporciona métodos para establecer ('set') y obtener ('get') valores en el arreglo.

En la función 'main', se crean tres instancias de 'Comun1': una para enteros con tamaño 5, otra para dobles con tamaño 5, y una tercera que utiliza los valores predeterminados (carácter y tamaño 8). Se establecen valores en las primeras dos instancias y se imprimen los valores obtenidos.

Al ejecutar el código, se observa que la instancia 'objInt' almacena el valor entero 10 en la posición 0, y la instancia 'objFloat' almacena el valor doble 3.1 en la posición 2. La tercera instancia 'obj', que utiliza los valores predeterminados, no tiene valores establecidos, por lo que al intentar obtener un valor de la posición 4, se obtiene un valor no inicializado (comportamiento indefinido).

La salida del programa será:



```

WSL at ~ / ... /laboratorios/lab06/ejer03
08:38:49 g++ comun.cpp -o comun
WSL at ~ / ... /laboratorios/lab06/ejer03
08:39:01 ./comun
10
3.1
WSL at ~ / ... /laboratorios/lab06/ejer03
08:39:06
```

Figura 3: Salida del programa al ejecutar el código proporcionado

4.4. Commits realizados

4.4.1. Primer Commit

- En este primer commit se subió el código completo del ejercicio la implementación de la clase template para una lista enlazada simple en C++.
- Se incluyó el archivo main.cpp para probar la funcionalidad de la clase template

```
WSL at C:\Users\user\WSL> cd ~/.../laboratorios/lab06/ejer01
15:10:24 ls
cola.h main.cpp
WSL at C:\Users\user\WSL> git add .
15:10:25 git add .
WSL at C:\Users\user\WSL> git commit -m "Se termino el ejercicio 01 del laboratorio"
15:10:32 git commit -m "Se termino el ejercicio 01 del laboratorio"
[main 3514cf8] Se termino el ejercicio 01 del laboratorio
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100755 laboratorios/lab06/ejer01/main
WSL at C:\Users\user\WSL> git push
15:10:55 git push
```

Figura 4: Primer Commit - Ejercicio 01 completo

4.4.2. Segundo Commit

- En este segundo commit se subió el análisis del código brindado en la guía de laboratorio para el ejercicio 02.
- Se incluyó la explicación del comportamiento del código y la salida obtenida al ejecutarlo.

```
WSL at C:\Users\user\WSL> cd ~/.../laboratorios/lab06/ejer02
15:27:05 git add .
WSL at C:\Users\user\WSL> git commit -m "Ejercicio 02 comprobado"
15:27:10 git commit -m "Ejercicio 02 comprobado"
[main 8b9eef0] Ejercicio 02 comprobado
1 file changed, 33 insertions(+)
create mode 100644 laboratorios/lab06/ejer02/contenedor.cpp
WSL at C:\Users\user\WSL> git push
15:27:27 git push
```



Figura 5: Segundo Commit - Ejercicio 02

4.4.3. Tercer Commit

- En este tercer commit se subió el análisis del código brindado en la guía de laboratorio para el ejercicio 03.
- Se incluyó la explicación del comportamiento del código y la salida obtenida al ejecutarlo.

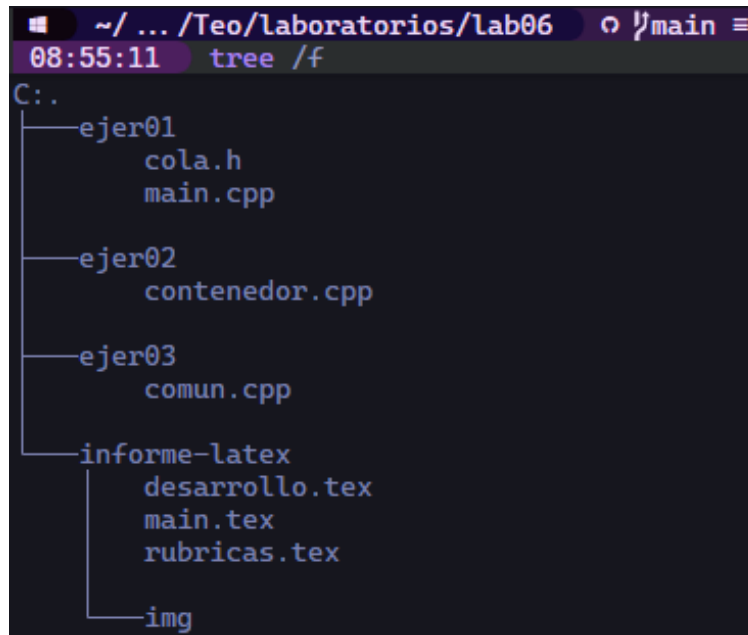
```
WSL at C:\Users\user\WSL> cd ~/.../laboratorios/lab06/ejer03
15:37:16 git add .
WSL at C:\Users\user\WSL> git commit -m "Ejercicio 03 comprobado"
15:37:21 git commit -m "Ejercicio 03 comprobado"
[main 1840f6f] Ejercicio 03 comprobado
1 file changed, 35 insertions(+)
create mode 100644 laboratorios/lab06/ejer03/comun.cpp
WSL at C:\Users\user\WSL> git push
15:38:18 git push
```

Figura 6: Tercer Commit - Ejercicio 03

	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 7

4.5. Estructura del laboratorio

A continuación se muestra la estructura de archivos y carpetas del laboratorio realizado: Claramente los archivos de compilación y otros que se pudieron generar no se subieron al repositorio.



```

~/.../Teo/laboratorios/lab06 08:55:11 tree /f
C:.\
├── ejer01
│   ├── cola.h
│   └── main.cpp
├── ejer02
│   └── contenedor.cpp
├── ejer03
│   └── comun.cpp
├── informe-latex
│   ├── desarrollo.tex
│   ├── main.tex
│   └── rubricas.tex
└── img
  
```

Figura 7: Estructura de archivos y carpetas del laboratorio

5. Cuestionario



5.1. Revisar y encontrar diferencias, ventajas y desventajas entre plantillas y polimorfismo.

5.1.1. Diferencias

- Las plantillas permiten la creación de funciones y clases genéricas que pueden trabajar con cualquier tipo de dato, mientras que el polimorfismo permite que una función o método se comporte de manera diferente según el objeto que lo invoque.
- Las plantillas se resuelven en tiempo de compilación, generando código específico para cada tipo utilizado, mientras que el polimorfismo se resuelve en tiempo de ejecución mediante el uso de punteros o referencias a clases base.

5.1.2. Ventajas

- Las plantillas permiten reutilizar código y evitar la duplicación, ya que una sola definición puede manejar múltiples tipos de datos.
- El polimorfismo permite la extensibilidad del código, ya que se pueden agregar nuevas clases derivadas sin modificar el código existente.

	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 8

5.1.3. Desventajas

- Las plantillas pueden aumentar el tamaño del código generado debido a la instanciación de múltiples versiones de la misma función o clase.
- El polimorfismo puede introducir una sobrecarga en tiempo de ejecución debido a la resolución dinámica de métodos.

5.2. Revisar sobre funciones amigas, cómo se integra a las plantillas.

Las funciones amigas en C++ son funciones que tienen acceso a los miembros privados y protegidos de una clase, incluso si no son miembros de esa clase. Para integrar funciones amigas con plantillas, se puede declarar una función amiga dentro de una clase plantilla utilizando la palabra clave 'friend'. Esto permite que la función amiga pueda acceder a los miembros privados de cualquier instancia de la clase plantilla, independientemente del tipo de dato utilizado.

Aquí se presenta un ejemplo de cómo declarar una función amiga en una clase plantilla:

Listing 4: Función amiga en una clase plantilla

```

1 template <typename T>
2 class MiClase {
3     T dato;
4 public:
5     MiClase(T val) : dato(val) {}
6     friend void mostrarDato<>(const MiClase<T>& obj);
7 };
8 template <typename T>
9 void mostrarDato(const MiClase<T>& obj) {
10     std::cout << "Dato: " << obj.dato << std::endl;
11 }



```

6. Rúbricas

6.1. Entregable Informe

Cuadro 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.



	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 9

6.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

Cuadro 2: Niveles de desempeño

Nivel				
Puntos	Insatisfactorio 25%	En Proceso 50%	Satisfactorio 75%	Sobresaliente 100%
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 10

Cuadro 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Check- klist	Estudian- te	Profe- sor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	25	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
Total		20		175	

7. Referencias

[1] <https://es.cppreference.com/w/cpp/language/friend>

[2] <https://www.geeksforgeeks.org/friend-function-in-cpp/>