# PROGRAMACIÓN ORIENTADA A OBJETOS (parte 4)

Elaborado por: Juan Miguel Guanira Erazo PUCP

# PLANTILLAS DE CLASES

# Definición:

Adaptándola a las clases, es una herramienta del lenguaje C++ que permite implementar una clase y que a la hora de compilarla el sistema genere varias versiones de esta clase de forma automática de modo que se adapten a los diferentes tipos de datos que se coloquen como atributos de la clase.

[3]

Al igual que con las funciones, la plantilla de clases requiere implementar un tipo de dato “genérico”. Este tipo de dato será el que cambiará en cada implementación.

**template < typename TIPO>**

Como se verá más adelante este encabezado deberá estar presente inmediatamente antes de la declaración de la clase, así como en la implementación de cada uno de los métodos.

# RESTRICCIONES:

Al igual que con las funciones, el código de una plantilla de clases no puede estar condicionado a un tipo de dato en especial. Esto es, no puede haber una pregunta que pretenda hacer algo diferente si el dato es de un tipo especial.

**El usuario de la plantilla de clases debe adaptase a la plantilla, la plantilla no puede adaptarse al tipo de dato que quiere emplear el usuario.**

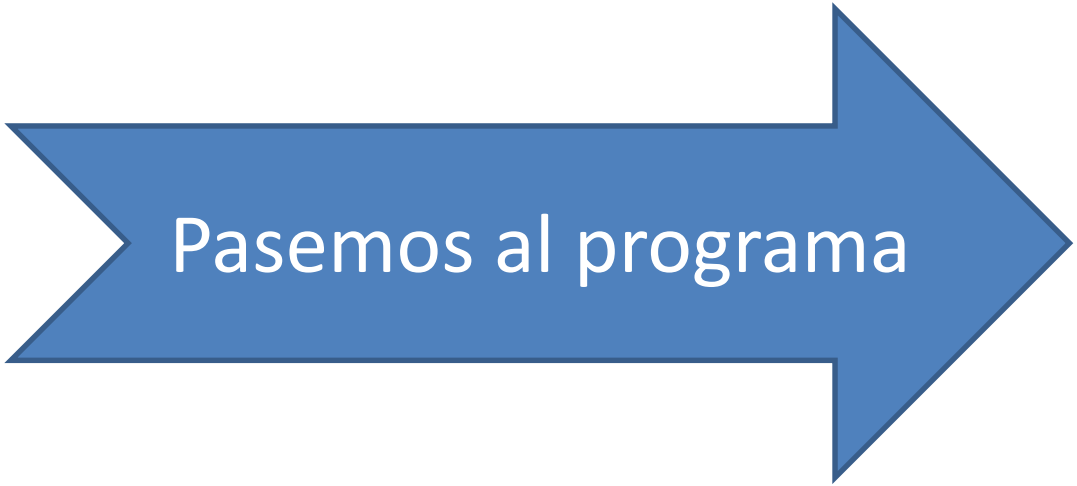
**Plantilla**

```
template <typename T>
class  Plantilla {
private:
      T  atibuto;
public:
     void metodo(T);
}

template <typename T>
void class Plantilla<T>::metodo(T v){
      …
}
```

**Proyecto**

```
#include <Plantilla.h>
#include <ClaseX.h>
int main(){
   class Plantilla <int>obj1;
   obj1.metodo(5);
   …
   class Plantilla <double>obj2;
   obj2.metodo(37.91);
   …
   class aseX  objX;
   class Plantilla < class ClaseX >obj3;
   obj3.metodo(objX);
   …
}
```


Pasemos al programa

# BIBLIOTECA ESTÁNDAR DE PLANTILLAS

## “Standard Template Library” (STL)

# Definición:

Contrario a su nombre, la denominada Biblioteca estándar de plantillas, no se trata de una sola biblioteca, sino de un conjunto de bibliotecas en las que se definen distintas estructuras de datos o como también se les llama **"contenedores"**.

Estos **"contenedores"** son genéricos, por lo que pueden almacenar cualquier tipo de dato.

Son muchos los contenedores que se definen en la STL, en el curso nos centraremos en tres de ellas.

## Biblioteca vector

Este contenedor simulará el trabajo con un arreglo dinámico. La asignación de memoria la hace por incrementos y maneja índices como cualquier arreglo.

## Biblioteca list

El contenedor simulará una lista ligada. Puede colocarse los datos al inicio o al final de la lista y se le puede recorrer en las dos direcciones.

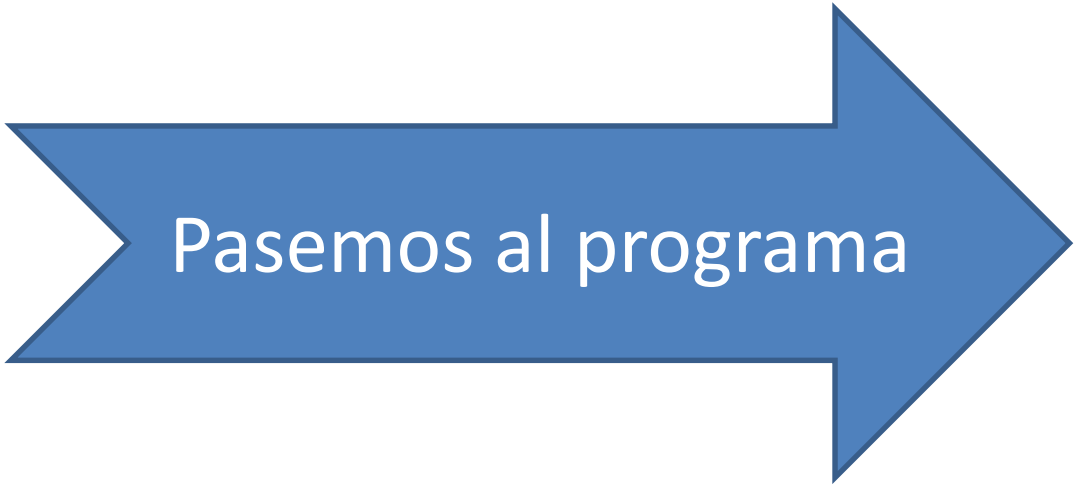
## Biblioteca map

Su principal utilidad es que se puede manejar como una tabla Hash muy eficiente.

# ITERADORES

Algunos contenedores como los “vector” se maneja a través de índices o de métodos apropiados.

Otros como los “list”, por su naturaleza no pueden manejarse con esos índices, por lo que se ha definido otra biblioteca denominada “iterator” que define elementos similares a los punteros para recorrerlos y manipularlos.


Pasemos al programa

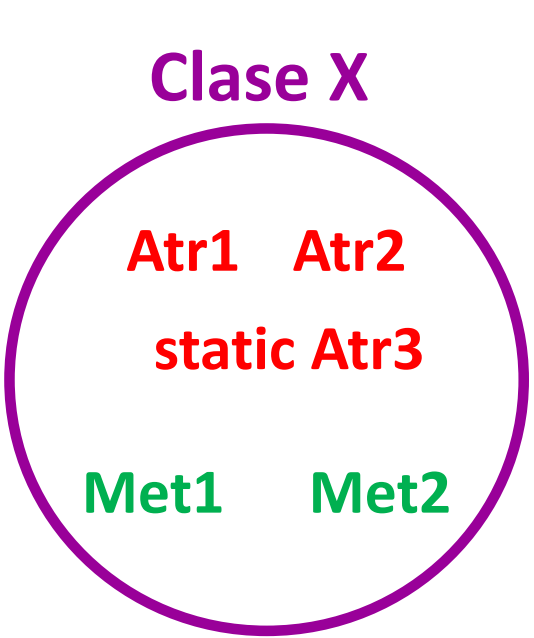
# ATRIBUTOS Y MÉTODOS ESTÁTICOS

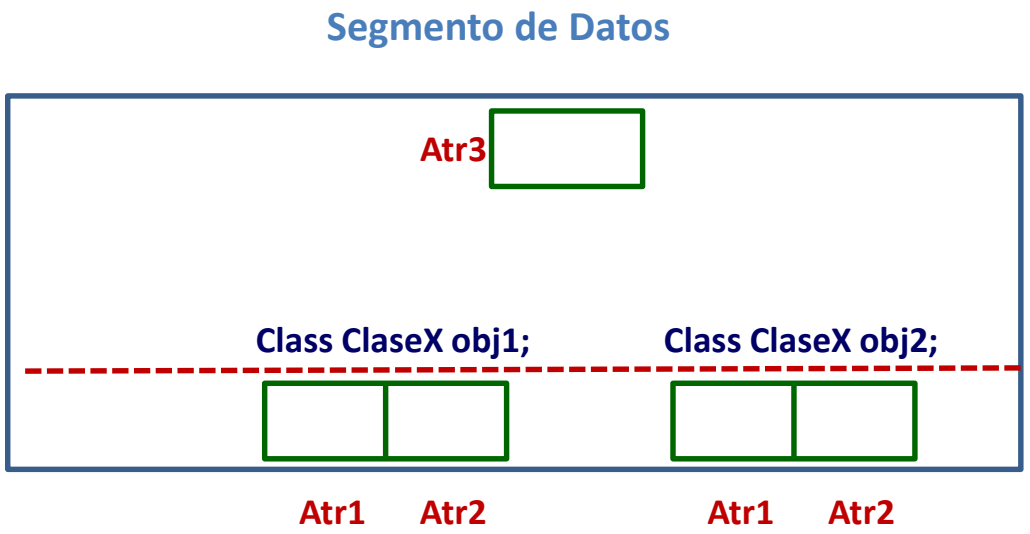
# Implementación:

Un atributo estático se define como:

```
class ClaseX{
    private:
        static int valor;
        ...
    public:
        ...
};
```

# Definición en memoria:



# Utilidad:

# Implementación:

Un método estático se define como:

```
class ClaseX{
    private:
        ...
    public:
        static int metodo();
        ...
};
```

# Utilidad:

Se crean para modificar los atributos estáticos.