**Manejo de plantillas (Clases Genéricas)**

**1. Introducción**

El manejo de plantillas en C++ permite la creación de clases genéricas que pueden trabajar con diferentes tipos de datos. Esto se logra utilizando plantillas de clases, que son clases parametrizadas por uno o más tipos de datos. Las plantillas permiten escribir código que se adapta automáticamente a diferentes tipos de datos, lo que mejora la reutilización del código y la legibilidad del mismo.

**2. Definiciones:**

* **Plantillas (Templates):**

Las plantillas son un mecanismo en C++ que permite definir funciones y clases que funcionan con tipos de datos genéricos.

Permiten escribir código que se puede reutilizar con diferentes tipos de datos sin tener que escribir múltiples versiones de la misma función o clase.

* **Clases Genéricas:**

Las clases genéricas, o plantillas de clases, son clases que pueden trabajar con tipos de datos genéricos.

Se definen utilizando la palabra clave template, seguida de la lista de parámetros de tipo entre <>.

* **Tipos de Datos Abstractos (TDA):**

Los TDAs son una abstracción en programación que encapsula datos y operaciones en una estructura cohesiva.

Permiten ocultar los detalles de implementación y proporcionan una interfaz clara para interactuar con los datos.

**3. Importancia de las plantillas:**

Las plantillas permiten escribir código genérico que puede funcionar con múltiples tipos de datos sin necesidad de escribir implementaciones específicas para cada tipo. Esto promueve la reutilización del código y reduce la duplicación.

También permiten la creación de abstracciones de datos y algoritmos independientes del tipo, lo que facilita la creación de estructuras de datos y algoritmos genéricos que pueden adaptarse a diferentes necesidades y tipos de datos.

Las plantillas proporcionan flexibilidad en el diseño del código al permitir la parametrización por tipos y valores. Esto permite adaptar el comportamiento de las funciones y clases a las necesidades específicas del usuario sin tener que cambiar la implementación.

**4. Ejemplo:**

En este ejemplo, crearemos una clase genérica llamada Par que representa un par de valores de cualquier tipo. La clase tendrá métodos para establecer y obtener los valores del par.

***Par.h***

#ifndef PAR\_H

#define PAR\_H

template <typename T1, typename T2>

**class** **Par** {

private:

T1 primero;

T2 segundo;

public:

Par(const T1& p, const T2& s) : primero(p), segundo(s) {}

T1 obtenerPrimero() const {

**return** primero;

}

T2 obtenerSegundo() const {

**return** segundo;

}

void establecerPrimero(const T1& p) {

primero = p;

}

void establecerSegundo(const T2& s) {

segundo = s;

}

};

#endif

***main.cpp***

#include <iostream>

#include "Par.h"

**int** **main**() {

Par<**int**, **double**> miPar(**5**, **3.14**);

std::cout << "Primer valor: " << miPar.obtenerPrimero() << std::endl;

std::cout << "Segundo valor: " << miPar.obtenerSegundo() << std::endl;

miPar.establecerPrimero(**10**);

miPar.establecerSegundo(**6.28**);

std::cout << "Primer valor actualizado: " << miPar.obtenerPrimero() << std::endl;

std::cout << "Segundo valor actualizado: " << miPar.obtenerSegundo() << std::endl;

**return** **0**;

}

**5. Conclusiones**

En conclusión, el uso de plantillas genéricas en C++ ofrece una manera poderosa de escribir código flexible y reutilizable al permitir la definición de clases y funciones que pueden trabajar con múltiples tipos de datos.

Esto promueve la reutilización del código al tiempo que proporciona flexibilidad y parametrización, lo que permite adaptar el comportamiento del código a diferentes necesidades sin cambiar su implementación. Además, las plantillas facilitan la abstracción y generalización al permitir la creación de estructuras de datos y algoritmos independientes del tipo, lo que simplifica el desarrollo y promueve una mayor modularidad y eficiencia en el diseño del software.

**6. Referencias**

* Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to Algorithms (4th ed.). MIT Press.

<https://dl.ebooksworld.ir/books/Introduction.to.Algorithms.4th.Leiserson.Stein.Rivest.Cormen.MIT.Press.9780262046305.EBooksWorld.ir.pdf>

* Drozdek, Adam. "Data Structures and Algorithms in C++." 5th Edition, Cengage Learning, 2019.

<https://itlectures.ro/wpcontent/uploads/2016/04/AdamDrozdek__DataStructures_and_Algorithms_in_C_4Ed.pdf>