Grado en Ingeniería Información

PROGRAMACIÓN II - Sesión 3

Tema 1.

Introducción a la programación orientada a objetos

Curso 2022-2023

Marta N. Gómez

UNIVERSIDAD NEBRIJA

UNIVERSIDAD NEBRIJA

T1. Introducción a la Programación Orientada a Objetos

ÍNDICE

Constructores

El ***constructor*** de una **Clase** es un **método/función especial** que **se ejecuta automáticamente cuando se crea/declara un objeto**.

El ***constructor*** permite **inicializar** los atributos/variables miembro de la clase.

Este método tiene las siguientes características:

- Tiene el **mismo nombre** que la Clase.
- **No** tiene valor de **retorno, ni *void***.
- Puede tener parámetros o no tenerlos.
- Una **clase** puede tener **varios constructores alternativos**, con diferentes parámetros (***sobrecarga*** de funciones).
- **Siempre** debe ser **público**.

Constructor por defecto o SIN parámetros es un método constructor que **no tiene parámetros**. Los atributos del objeto se inicializan con **valores por defecto**.

Se ejecuta cuando al declarar el objeto **no se especifican valores de inicialización con parámetros**.

Constructores sin parámetros

```
#include <iostream>

class Prueba{
  public:

  // Constructor sin parámetros
  Prueba();
};

int main(){
  Prueba unaPrueba; // se crea un objeto y se ejecuta el constructor
  return 0;
}

Prueba::Prueba(){
    std::cout << "\n\n\tConstructor de la clase Prueba\n";
  }
```

Constructor de oficio es el método **constructor sin parámetros** creado por el compilador automáticamente si la clase no tiene definido ningún constructor.

El **constructor de oficio** es un **constructor por defecto** creado por el **compilador automáticamente** para inicializar los atributos.

Este constructor **inicializa las variables** pero puede producir **resultados incorrectos** e incluso a **errores**.

Constructores: Inicializadores

Los **inicalizadores** de C++ permiten dar valor a los atributos de forma más eficiente.

Sintaxis:

nombre_constructor (tipo1 ***par1****, tipo2* ***par2****, ...)* **:**
*atributo1(**par1**), atributo2(**par2**), ...* ***// inicializadores***
{ ***// En este caso el cuerpo del constructor está vacío*** *}*

Constructores: Inicializadores

```
#include <iostream>
#include <array>

class Matriz2x2  {
  public:
    Matriz2x2():datos{4,4,4,4}{};    // Constructor con inicializadores

    void mostrar() const;
  private:
    std::array<float, 4> datos;
};

int main(){
  Matriz2x2 miMatriz;
  std::cout << "\n\n\tLa matriz miMatriz es:" << std::endl;
  miMatriz.mostrar();

  std::cout << "\n\n\t";
  return 0;
}
```

Constructores con parámetros

```
#include <iostream>
#include <array>

class Matriz2x2  {
  public:
    Matriz2x2();      // Constructor sin parámetros
    Matriz2x2(std::array<float, 4> const & v);   // Constructor con parámetros
    void mostrar() const;
  private:
    std::array<float, 4> datos;
};

int main(){
  Matriz2x2 miMatriz;
  std::cout << "\n\n\tLa matriz miMatriz es:" << std::endl;
  miMatriz.mostrar();

  Matriz2x2 newMatriz{{1,2,3,4}};
  std::cout << "\n\n\tLa matriz newMatriz es:" << std::endl;
  newMatriz.mostrar();

  std::cout << "\n\n\t";
  return 0;
}
```

Constructores con parámetros

```
//Constructor SIN parámetros
Matriz2x2::Matriz2x2()  {
  datos = {0,0,0,0};
}

//Constructor CON parámetros
Matriz2x2::Matriz2x2(std::array<float, 4> const & v)  {
  datos = v;
}

//Método que visualiza
void Matriz2x2::mostrar() const  {
  std::cout << "\n\t|" << datos.at(0) << "\t" << datos.at(1) << "|";
  std::cout << "\n\t|" << datos.at(2) << "\t" << datos.at(3) << "|";
}
```

Constructor copia

Constructor con un único parámetro que es una **referencia constante a un objeto de la misma clase**. Es decir, se inicializa un objeto con el estado de otro objeto de la misma clase.

```
nombre_Clase (const nombre_Clase& nombre_Objeto); {
        // operaciones necesarias para copiar el objeto
}
```

El compilador también proporciona un Constructor Copia de oficio cuando no ha sido definido que hace una copia **literal del objeto**.

Constructor copia

```
#include <iostream>
#include <array>

class Matriz2x2  {
  public:
    Matriz2x2();      // Constructor sin parámetros
    Matriz2x2(std::array<float, 4> const & v);   // Constructor con parámetros
    Matriz2x2(Matriz2x2 const &M);  // Constructor copia
    void mostrar() const;
  private:
    std::array<float, 4> datos;
};

int main(){
  Matriz2x2 M2x2{newMatriz};
  std::cout << "\n\n\tLa matriz M2x2 es:" << std::endl;
  M2x2.mostrar();

  std::cout << "\n\n\t";
  return 0;
}

//Constructor copia
Matriz2x2::Matriz2x2(Matriz2x2 const &M)  {
  datos = M.datos;
}
```

Punteros a objetos

```cpp
#include <iostream>
#include <array>

class Matriz2x2  {
  public:
    Matriz2x2():datos{4,4,4,4}{};    // Constructor con inicializadores

    void mostrar() const;
  private:
    std::array<float, 4> datos;
};

int main(){
  std::array<float, 4> v{5,5,5,5};
  std::shared_ptr<Matriz2x2> pM=std::make_shared<Matriz2x2>(v);
  std::cout << "\n\n\tLa matriz M2x2 es:" << std::endl;
  pM->mostrar();

  std::cout << "\n\n\t";
  return 0;
}
```

Destructores

El **destructor** es un método miembro de una clase que se ejecuta cuando el **objeto deja de existir:**

- Si es **un objeto (local) definido dentro de un bloque**, el destructor se llama/ejecuta cuando el programa llega **al final de ese bloque**.
- Si **el objeto es global** su duración es la misma que la del programa, y el destructor se llama/ejecuta **al terminar la ejecución del programa**.

Destructores

El **destructor** siempre es **único** (no puede estar sobrecargado).

No tiene parámetros ni valor de retorno.

Su nombre es igual que la clase precedido por el **carácter tilde** (**~**), el carácter se consigue tecleando **Alt+126**.

```
~nombreClase () {
// operaciones necesarias para liberar recursos
}
```

Si el programador no lo define, el compilador proporciona un **destructor de oficio**, que normalmente es adecuado.

Ejercicios

Escribir un programa en C++ con una clase **Circulo** que tenga como atributos el **radio** y el **area** del círculo. Además, debe tener los siguientes métodos:

- Constructor por defecto con radio que valga 10
- Constructor con inicializadores
- Constructor con parámetros
- Constructor copia
- Los métodos ***get*** y **set** para obtener y dar valor a los atributos
- Método para calcular el valor de área en función del radio.

El programa creará cuatro objetos, C1, C2, C3 y C4 utilizando los constructores anteriores. Se debe conseguir tener círculos de 10, 5 y 15 cm de radio, calcular el área para cada uno de ellos y mostrar los datos de cada uno por pantalla.

Ejercicios

Escribir un programa en C++ con una clase **vectorElementos** que tenga un atributo que sea un **array de 10 enteros**.

La clase debe permitir modificar los valores de los elementos del array mediante un método **cambiaElemento** y calcular la suma de los elementos del array mediante un método **calculaSuma**.

Además, cuando se crea un objeto de esta clase el array contendrá los valores del 1 al 10.

Después, implementar un programa en C++ que cree un objeto de esta clase y muestre por pantalla el resultado de la suma de los elementos del array, antes de modificarlos y después de hacer varias modificaciones.

Repetir el proceso de modificación de valores mientras quiera el usuario.