CS1112: Programación II

Unidad 5: POO

Sesión de Laboratorio - 8B

Profesores:
María Hilda Bermejo **mbermejo@utec.edu.pe**
Estanislao Contreras **econtreras@utec.edu.pe**
Jorge Villavicencio **jvillavicencio@utec.edu.pe**
Edson Mendiolaza **emendiolaza@utec.edu.pe**
Ian Paul Brossard **ibrossard@utec.edu.pe**
Jose Chavez **jchaveza@utec.edu.pe**
Julio Yarasca **jyarascam@utec.edu.pe**
Percy Quevedo **pquevedo@utec.edu.pe**
Wilder Nina **wnina@utec.edu.pe**
José Fiestas **jfiestas@utec.edu.pe**

Material elaborado por:
Maria Hilda Bermejo

5.2

Unidad 5: POO
Constructores y destructores

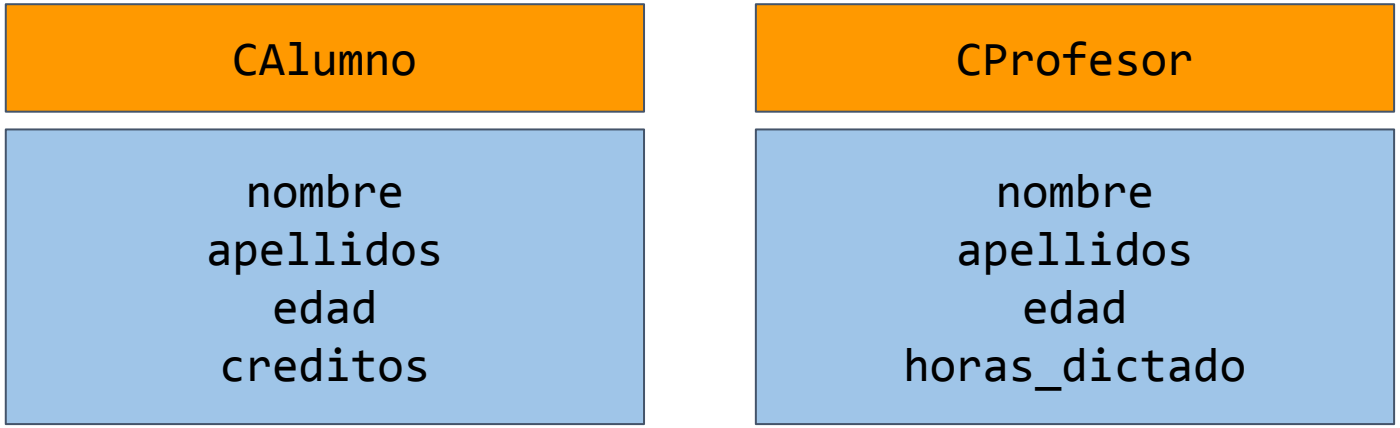
Logro de la sesión:

Al finalizar la sesión, los alumnos se familiarizan con el paradigma de la programación orientada a Objetos.

- Clase – Objeto
- Métodos de acceso (setter y getters)
- Constructores y destructores

Ejercicio 1

En este ejercicio se le pide implementar dos clases CAlumno y CProfesor con los atributos que se muestra en el gráfico siguiente.



También debe implementar los métodos de acceso a datos “setter” y “getter” para cada atributo. Luego, en el archivo “main.cpp” implementar un método genérico para imprimir los datos básicos tanto para el alumno como para el profesor, es decir: nombre, apellidos y edad.

Ejercicio 1

Main.cpp

```cpp
#include <iostream>
#include "CAlumno.h"
#include "CProfesor.h"

//*************************************************//
// implementar aquí la función mostrarDatosBasicos //
//*************************************************//

int main() {
    CAlumno a1;
    a1.setNombre("Felipe");
    a1.setApellidos("Carranza Arriola");
    a1.setEdad(19);

    CProfesor p1;
    p1.setNombre("Maria");
    p1.setApellidos("Avalos Sanchez");
    p1.setEdad(35);

    mostrarDatosBasicos<CAlumno>(a1);
    mostrarDatosBasicos<CProfesor>(p1);

    return 0;
}
```

Output

```
-------
Nombre: Felipe
Apellidos: Carranza Arriola
Edad: 19
-------
Nombre: Maria
Apellidos: Avalos Sanchez
Edad: 35
```

Ejercicio 1

Main.cpp

```cpp
#include <iostream>
#include "CAlumno.h"
#include "CProfesor.h"

template <typename T>
void mostrarDatosBasicos(T &objeto)
{
    cout<<"--------\n";
    cout<<"Nombre: "<<objeto.getNombre()<<endl;
    cout<<"Apellidos: "<<objeto.getApellidos()<<endl;
    cout<<"Edad: "<<objeto.getEdad()<<endl;
}
int main() {
    CAlumno a1;
    a1.setNombre("Felipe");
    a1.setApellidos("Carranza Arriola");
    a1.setEdad(19);

    CProfesor p1;
    p1.setNombre("Maria");
    p1.setApellidos("Avalos Sanchez");
    p1.setEdad(35);

    mostrarDatosBasicos<CAlumno>(a1);
    mostrarDatosBasicos<CProfesor>(p1);

    return 0;
}
```

Output

```
--------
Nombre: Felipe
Apellidos: Carranza Arriola
Edad: 19
--------
Nombre: Maria
Apellidos: Avalos Sanchez
Edad: 35
```

Ejercicio 2

Crear una clase Inmueble con atributos para el área, y precio por metro cuadrado, así como 3 constructores. Genere un procedimiento de compra/venta con un menú que permita comprar un inmueble (departamento, casa u hotel). Realice la compra de los siguientes inmuebles:

3 departamentos, cada uno con un área aleatoria entre 100 y 200 m^2
4 casas, cada uno con un área aleatoria entre 500 y 800 m^2
2 hoteles, cada uno con un y área aleatoria entre 1000 y 1500 m^2

Ejercicio 2

El **precio por metro cuadrado** es un aleatorio entre 2000 y 2500 Soles.

El pago será a plazos, para lo que es necesario calcular el **tiempo de pago** si se decide por una **cuota mensual**, si se tiene un **interés** del 0.2% mensual.

Almacene los datos de los inmuebles comprados en un *vector*, y genere una función externa a la clase para imprimir, en cada caso, el nombre del inmueble, su precio por metro cuadrado, y el tiempo de pago (en años).

Ejercicio 3

Defina la clase **NumeroBinario** que representa números binarios de 64 bits, por ejemplo con un array de dimensión 64. La clase debe contener los siguientes métodos:

a) Un constructor, que **inicialice** todos los bits a cero

b) Un método que **imprima** el valor del número binario

c) Un método que pida un número entero y permita **asignar** el valor a un bit a 1, en esa posición. La función retorna el valor **true** si el valor del bit se cambió a 1, y **false** en caso no se pudo cambiar porque el entero no representa ninguna posición del binario.

d) Un método que permita **incrementar** el valor de **un bit** de 0 a 1. Para ello se empieza a recorrer los bits de menor a mayor (derecha a izquierda). Si el bit es 0, se convierte a 1, y termina la ejecución. Si el bit es 1, se convierte a 0, y se sigue con el siguiente bit.

Ejercicio 3

El programa debe ejecutar la siguiente secuencia:

- **imprimir** el valor inicial del número binario
- usar el método **c)** para pedir un número n e intentar **asignar** el bit a 1 en esa posición
- Asimismo, **asignar** a 1 los bits en las posiciones 2 a 20, 61 y 62
- **imprimir** el número binario
- llamar el método **d) incrementar**, 30 veces e **imprimir** el número binario

Ejercicio 4

En versiones modernas de C++ se ha introducido la clase “vector” para operar con arreglos unidimensionales de datos.
En este ejercicio se le pide realizar una implementación propia de dicha estructura de datos considerando las siguientes características :

- La clase Vector debe gestionar los elementos en un arreglo dinámico.
- Implementar al menos tres constructores: constructor vacío, constructor para crear e inicializar el arreglo dinámico, y un constructor que copie los datos de otro vector.
- Se debe también implementar los métodos básicos para insertar elementos al arreglo dinámico y para obtener sus valores.
- Y por último, implementar en el destructor la liberación de memoria del arreglo dinámico.

Ejercicio 4

Main.cpp

```cpp
#include "CVector.h"

int main()
{
    //vector sin inicializar
    CVector vector1();

    //vector inicializado con tamaño 5 y elementos iniciali
zados con 1
    CVector vector2(5, 1);

    //vector inicializado a partir de otro vector
    CVector vector3(vector2);//[1,1,1,1,1]

    //reemplazar el valor de un elemento
    vector3.setElemento(20, 0);//[20,1,1,1,1]
    //insertar elemento en una posicion especifica
    vector3.insertar(25, 4);//[20,1,1,1,25,1]
    //agregar elemento al final del vector
    vector3.agregar(10);//[20,1,1,1,25,1,10]

    //obtener el tamaño actual del vector
    entero n = vector3.getTamanio();//n = 7
    for(int i=0; i<n; i++)
        cout<<vector3.getElemento(i)<<endl;

    return 0;
}
```

CVector.h

```cpp
#include <iostream>
using namespace std;
typedef int entero;
typedef int* array_entero;

class CVector
{
private:
    entero tamanio;
    array_entero elementos;
    entero maximo;
    void redimensionar();
public:
    //constructor vacio
    CVector();
    //constructor inicilizador
    CVector(entero _tamanio, entero _valor);
    //constructor copia
    CVector(const CVector &otro_vector);
    //destructor
    virtual ~CVector();
    //reemplazar elemento
    void setElemento(entero elemento, entero posicion);
    //insertar elemento
    void insertar(entero elemento, entero posicion);
    //agregar elemento
    void agregar(entero elemento);
    //getters
    entero getTamanio();
    entero getElemento(entero posicion);
};
```

¡Nos vemos en la siguiente clase!

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA