Laboratorio 06

Clases

1. Competencias

1.1. Competencias del curso

Conoce, comprende e implementa programas usando clases en el lenguaje de programación C++.

1.2. Competencia del laboratorio

Conoce, comprende e implementa programas usando clases en el lenguaje de programación C++.

2. Equipos y Materiales

- Un computador.
- IDE para C++.
- Compilador para C++.

3. Marco Teórico

3.1. Introducción a Clases en C++

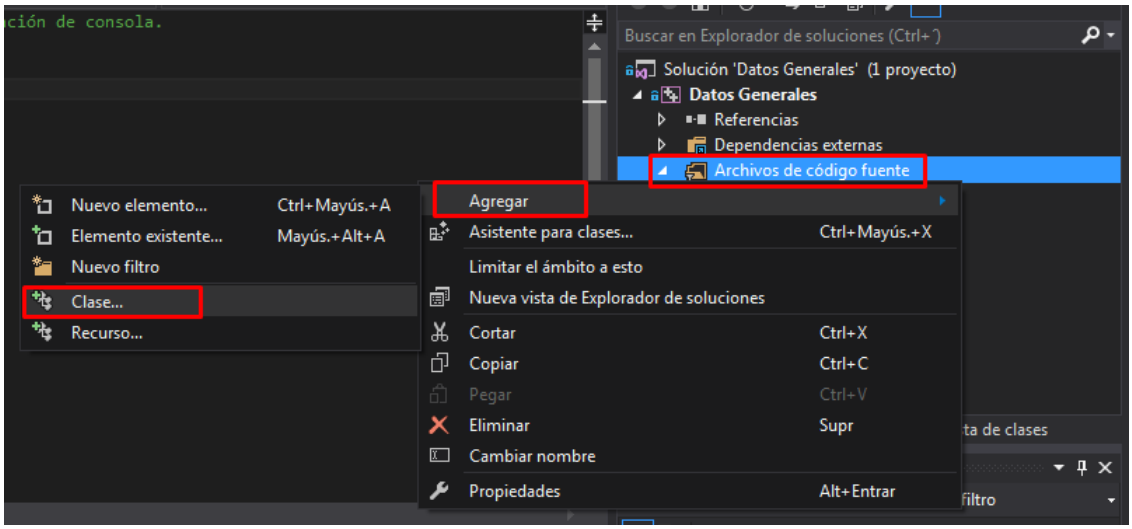
En C++ se suelen definir e implementar las clases en sus propios archivos y no en el mismo donde se encuentra la función principal, se hace de esta manera con el fin de lograr una buena organización en los proyectos y hacer un tipo de encapsulamiento externo porque la implementación de la clase queda fuera de vista del programa principal.
Cada clase irá programada en dos archivos:

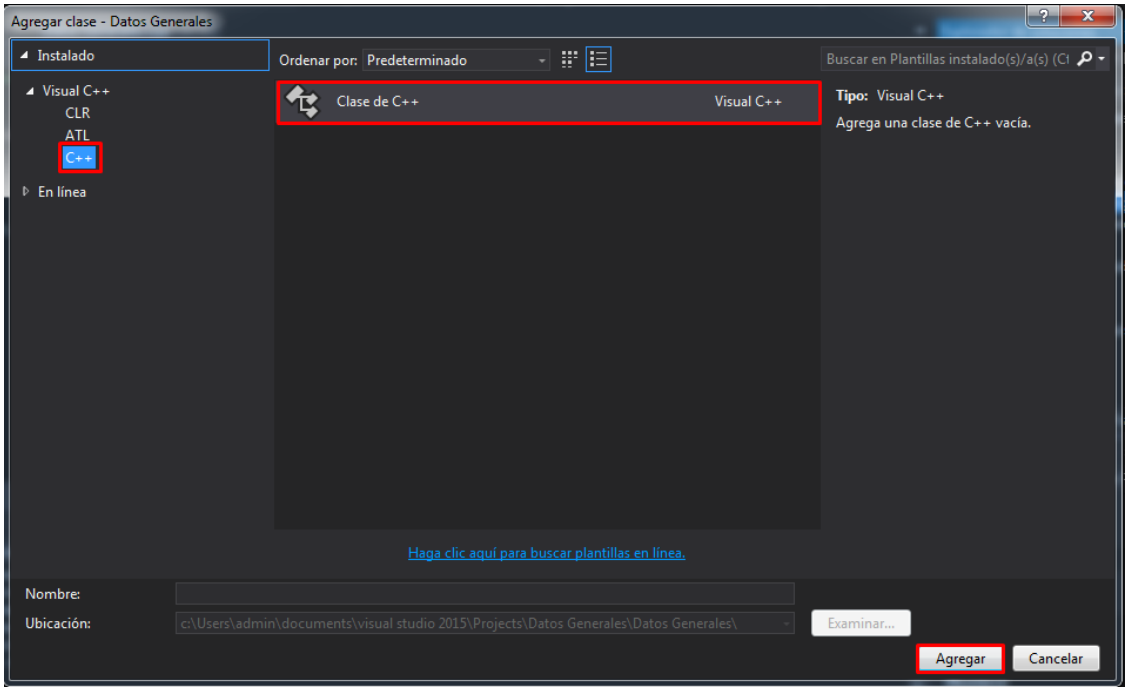
Un archivo de cabecera (.h)
En donde se declara la clase y se definen sus atributos y métodos, en este archivo solamente se declaran los miembros de la clase, para los métodos se escriben los prototipos de función y si es pertinente, para aquellas funciones cortas su implementación como inline.

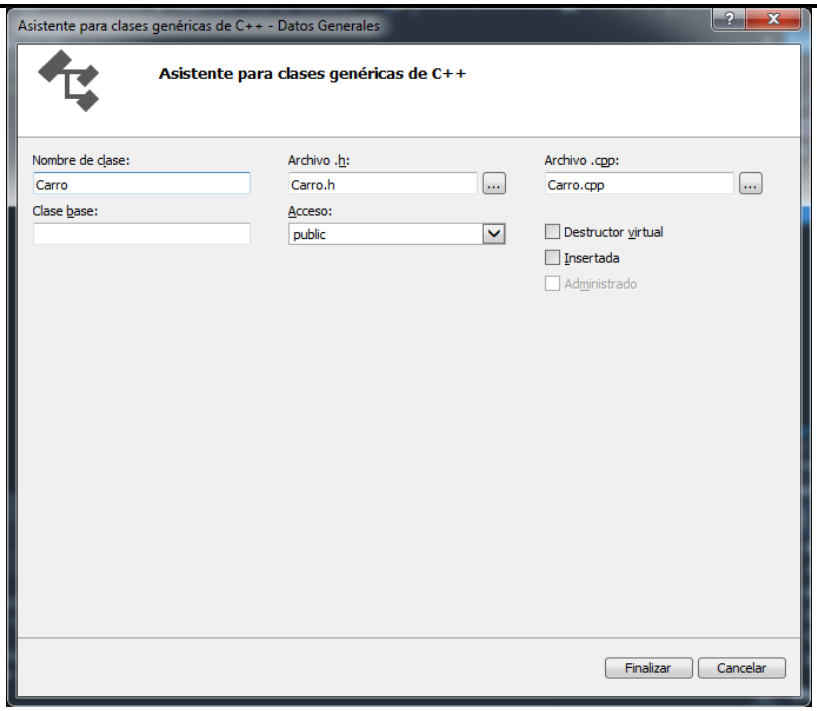
Un archivo fuente (.cpp)
En donde se realiza la implementación de aquellos métodos de los que declaramos su prototipo en la definición de la clase.

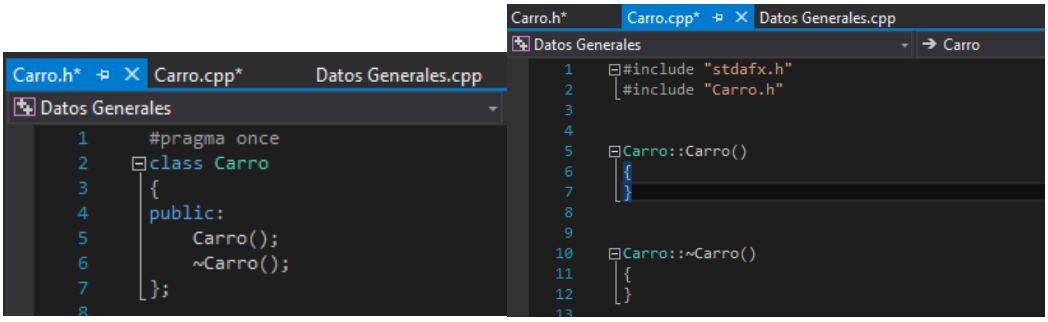
De esta manera podremos tener a la vista, en el archivo de cabecera, las características de la clase y saber qué es lo que tenemos en un objeto de dicha clase; así mismo dejamos de lado los detalles más profundos de la implementación en el archivo fuente.

Los pasos para crear una clase:



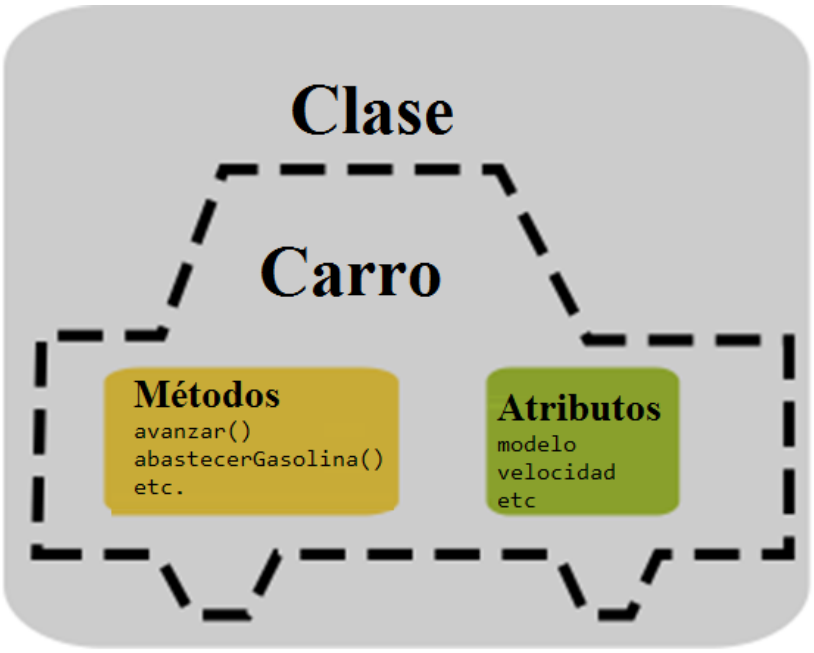






3.2. Definición de la clase

Lo que debe hacerse primero es la declaración de la clase, es decir, dar nombre a la clase y establecer cuáles serán sus atributos y métodos, así como la visibilidad de estos dos últimos ante el exterior.
Una clase en C++ puede pensarse como un Tipo de Dato Abstracto (TDA), es decir, un conjunto de datos y las operaciones asociadas a esos datos. En términos de objetos, el conjunto de datos equivale a los atributos de la clase y las operaciones o algoritmos que se aplican a dichos datos serían los métodos.



La sintaxis para declarar una clase es la siguiente:

```
class nombre
{
especificador_de_acceso_1:
   miembro1;
   miembro2;
   ...
   miembroN;

especificador_de_acceso_2:
   miembroN+1;
   ...
   miembroM;
...
};
```

Se utiliza la palabra reservada class seguida por un identificador que sirva como nombre a la clase. Entre llaves escribiremos el cuerpo de la clase que se conformará por declaraciones de los miembros de la clase, ya sean datos (atributos) o funciones (métodos) precedidos opcionalmente por un especificador de acceso a manera de etiqueta, una de las palabras reservadas: public, private o protected. Según el especificador se establece el acceso a los miembros de la clase:

private:

Los miembros privados son accesibles solamente desde adentro de la clase, únicamente los miembros de esa clase podrán verlos.

protected:
Los miembros protegidos únicamente pueden ser visibles por los demás miembros de la clase y de sus clases hijas o derivadas.

public:
Los miembros públicos son accesibles desde cualquier lugar donde el objeto al que pertenecen sea visible.

Veamos un ejemplo:

```
//Definición de la clase carro -> archivo "Carro.h"
#pragma once
#include <iostream>
using namespace std;
class Carro
{
	// Atributos: caracteristicas del carro
private:
	string modelo;
	int gasolina;
	int velocidad;

	// Metodos: Acciones que realiza la clase
public:
	Carro(string, int, int);   // Constructor
	~Carro();     // Destructor
	void avanzar();
	void abastecerGasolina();
	};
```

Los carros que modelamos tendrán sus datos privados, por lo tanto, solamente podrán ser leídos o modificados por los métodos de la misma clase.
La clase posee algunos métodos públicos, lo que significa que todo el comportamiento de estos objetos estará visible desde otros objetos.
La clase tiene un constructor definido, este método se llamará al momento de instanciar un carro. Nótese que el constructor no lleva un tipo de dato de retorno en la declaración y tiene el mismo identificador que la clase.

3.3. Implementación de los métodos

Una vez que ya tenemos nuestra definición de la clase podemos trabajar en el archivo fuente en donde escribiremos la implementación de los métodos, esta parte del código normalmente solo está disponible para quien escribe la clase, en el momento de estarla escribiendo o depurándola; una vez terminado suele ser consultado únicamente el archivo de cabecera que nos da una visión general de la clase.

El archivo de implementación tendrá una estructura como la siguiente:

```
// Archivo fuente de implementación de una clase
#include "Archivo_cabecera.h"

tipo_retorno NombreClase::nombreMetodo1(argumento1,argumento2,...)
{
   //Código del método
}

tipo_retorno NombreClase::nombreMetodo2(argumento1,argumento2,...)
{
   //Código del método
}
```

El archivo fuente debe tener la directiva del preprocesador para incluir la definición de la clase que se encuentra en el archivo de cabecera (.h). Después podremos hacer la implementación de cada método en el resto del archivo; los escribiremos como una función cualquiera: colocando el tipo de dato de retorno (con excepción del constructor) seguido por el nombre de la función miembro (método), que está conformado por el nombre de la clase a la que pertenece, el operador de ámbito "::" y el nombre de la función, después entre paréntesis la definición de los parámetros y por último el cuerpo de la función que es donde escribiremos finalmente el algoritmo que define al método.

Veamos el ejemplo de la clase Carro:

```
// Clase Carro.cpp
#include "stdafx.h"
#include "Carro.h"
#include "iostream"
#include "string"

// Constructor
Carro::Carro(string _modelo,int _gasolina,int _velocidad)
{
    modelo = _modelo;
    gasolina = _gasolina;
    velocidad = _velocidad;
}

Carro::~Carro()
{
}

void Carro::avanzar() {
    cout << "El auto es de modelo " << modelo << " llega a una velocidad de " <<
velocidad <<endl;
}

void Carro::abastecerGasolina() {
```

```
    cout << " El modelo de auto " << modelo << " usa gasolina de " << gasolina <<
endl;
}
```

Con este archivo finalizado ya tenemos una clase implementada y podremos usarla para crear objetos en cualquier programa que escribamos posteriormente, si bien es cierto que nuestra clase es muy básica y se queda bastante corta para modelar a los carros, sabemos que cualquier clase puede crecer; le podemos añadir más características y hacerla tan compleja como lo deseemos. A continuación se presente el contenido del main():

```
#include "stdafx.h"
#include <iostream>
#include "Carro.h"


int main()
{
    Carro c1 = Carro("Yaris", 90, 200);
    c1.avanzar();
    system("pause");
     return 0;
}
```

4. Ejercicios

Resolver los siguientes ejercicios planteados:

1. Implemente un programa con clases que calcule el área de un rectángulo y perímetro.
2. Se conoce de un alumno de la Universidad Nacional de San Agustín: CUI, nombre completo y tres notas parciales (nota1, nota2, nota3). El programa con clases debe imprimir: CUI, el primer nombre, el promedio de las tres notas e indique con un mensaje si el alumno aprobó (nota final >= 10.5) o no aprobó (nota final < 10.5) la asignatura de Ciencias de la Computación II.
3. Implemente un programa con clases que lea la fecha de nacimiento y la fecha de hoy y muestre por pantalla el nombre y la edad de la persona.
4. Implemente una aplicación con clases donde una clase contiene un arreglo en el que se pueden almacenar como máximo 5 datos. Debe tener un nuevo dato para almacenarlo, debe existir un método que verifique que el arreglo tenga espacio disponible para guardar el dato. En caso contrario se debe mostrar un mensaje indicando que el arreglo está lleno. Asimismo, debe tener un método para retirar un dato del arreglo, este método debe verificar que, si haya algo para sacar del arreglo, es decir que el arreglo no vaya a estar vacío, en cuyo caso debe desplegar un mensaje diciendo que no hay nada para sacar de arreglo.
5. Implementar un programa con clases que haga la búsqueda de un dato almacenado en una matriz de 3x3 que tiene los números de 1 al 9 ingresados aleatoriamente y debe indicar la posición donde se encuentra el dato.

5. Entregables

Al final estudiante deberá:

1. Compactar el código elaborado y subirlo al aula virtual de trabajo. Agregue sus datos personales como comentario en cada archivo de código elaborado.
2. Elaborar un documento que incluya tanto el código como capturas de pantalla de la ejecución del programa. Este documento debe de estar en formato PDF.
3. El nombre del archivo (comprimido como el documento PDF), será su LAB06_GRUPO_A/B/C_CUI_1erNOMBRE_1erAPELLIDO.

   (Ejemplo: LAB06_GRUPO_A _2022123_PEDRO_VASQUEZ).
4. Debe remitir el documento ejecutable con el siguiente formato:

   LAB06_GRUPO_A/B/C_CUI_ EJECUTABLE_1erNOMBRE_1erAPELLIDO

   (Ejemplo: LAB06_GRUPO_A_EJECUTABLE_2022123_PEDRO_VASQUEZ).

En caso de encontrarse trabajos similares, los alumnos involucrados no tendrán evaluación y serán sujetos a sanción.