

Universidad Mariano Gálvez de Guatemala

Facultad de ingeniería en sistemas

Campus Villa nueva, Guatemala

Curso: Programación I

Docente: Inge. Carlos Arias



Actividad: Laboratorio 7

Nombre: Stephanie Cristina Sabán Cárcamo

Carnet: 5090- 23-11167

Sección: "A"

Fecha: 08/04/24

## **Introducción**

En este trabajo se incluye el manejo de archivos a través de la biblioteca `fstream`, que permite leer y escribir datos en archivos de texto y binarios.

C++ también es un lenguaje orientado a objetos, basado en conceptos clave como la encapsulación, herencia, polimorfismo y abstracción. La abstracción se logra mediante el uso de clases y funciones, y la instanciación es el proceso de crear objetos a partir de estas clases.

Las clases definen los datos que vienen siendo los atributos y los comportamientos que vendrían siendo los métodos de los objetos, mientras que las funciones son bloques de código reutilizables.

Otras bibliotecas importantes en C++ son `string` e `iostream`, que proporcionan herramientas para trabajar con cadenas de texto y realizar operaciones de entrada/salida. Finalmente, los algoritmos de ordenamiento, como el ordenamiento por burbuja, inserción, son procedimientos fundamentales para reorganizar elementos en una lista o array.

## Ejercicio 1 - Manipulación de Archivos

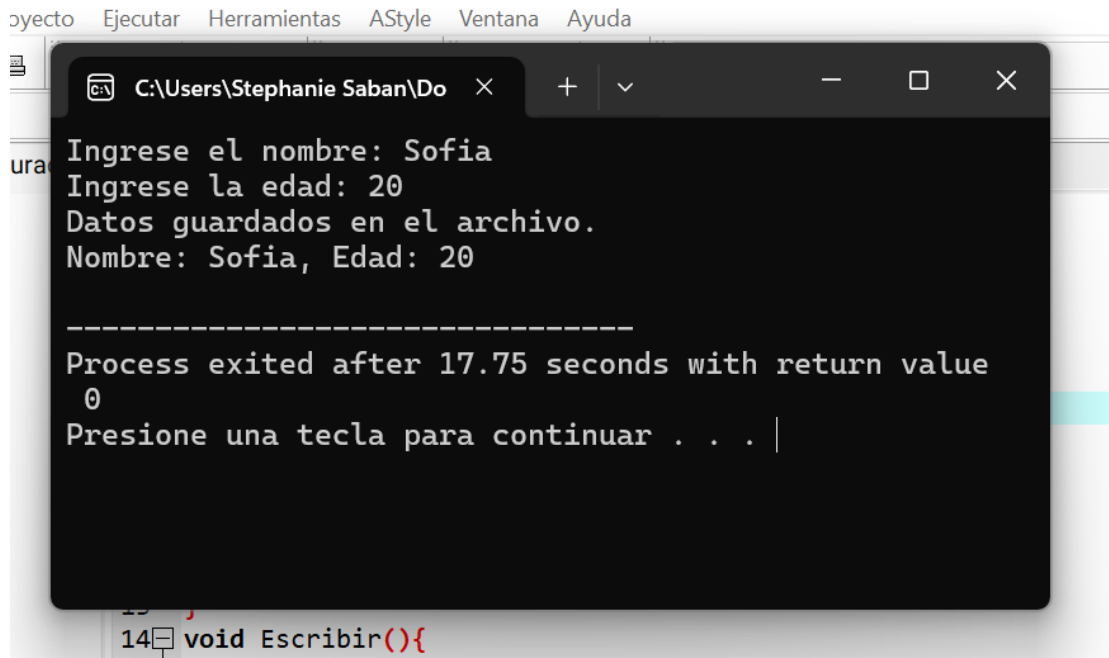
```
Ej1.cpp
1 #include <iostream>
2 #include <stdlib.h>
3 #include <fstream>
4 #include <string>
5 using namespace std;
6 void Escribir();
7 void LeerMostrar();
8
9 int main() {
10     Escribir();
11     LeerMostrar();
12     return 0;
13 }
14 void Escribir(){
15     // Abrir un archivo en modo de escritura
16     ofstream archivito("exercise1.txt", ios::out | ios::binary);
17     if (archivito.is_open()) {
18         string nombre;
19         int edad;
20         cout << "Ingrese el nombre: ";
21         getline(cin, nombre);
22         cout << "Ingrese la edad: ";
23         cin >> edad;
24         cin.ignore(); // Ignorar el salto de línea después de Leer la edad
25
26         archivito << nombre << "," << edad << endl; // Escribir los datos en el archivo
27         archivito.close(); // Cerrar el archivo
28         cout << "Datos guardados en el archivo." << endl;
29     } else {
30         cout << "No se pudo abrir el archivo." << endl;
31     }
32 }
33 void LeerMostrar(){
34     ifstream archivito("exercise1.txt"); // Abrir el archivo en modo de Lectura
35     if (archivito.is_open()) {
36         string linea;
37         while (getline(archivito, linea)) {
38             size_t posicionComa = linea.find(",");
39             string nombre = linea.substr(0, posicionComa);
40             int edad = stoi(linea.substr(posicionComa + 1));
41             cout << "Nombre: " << nombre << ", Edad: " << edad << endl;
42         }
43         archivito.close(); // Cerrar el archivo
44     } else {
45         cout << "No se pudo abrir el archivo." << endl;
46     }
47 }
```

Historial de Compilación Depuración Resultados Console

Este código demuestra el uso de la biblioteca `fstream` para leer y escribir datos en un archivo de texto. Comienza con la inclusión de las bibliotecas necesarias: `iostream` para entrada/salida, `stdlib.h` para funciones de utilidad, `fstream` para manejo de archivos y `string` para trabajar con cadenas de texto.

La función `Escribir()` abre un archivo en modo de escritura, solicita al usuario que ingrese un nombre y una edad, y luego escribe esos datos en el archivo en formato de texto separado por comas. La función `LeerMostrar()` abre el mismo archivo en modo de lectura, lee línea por línea, separa el nombre y la edad, y los muestra en pantalla.

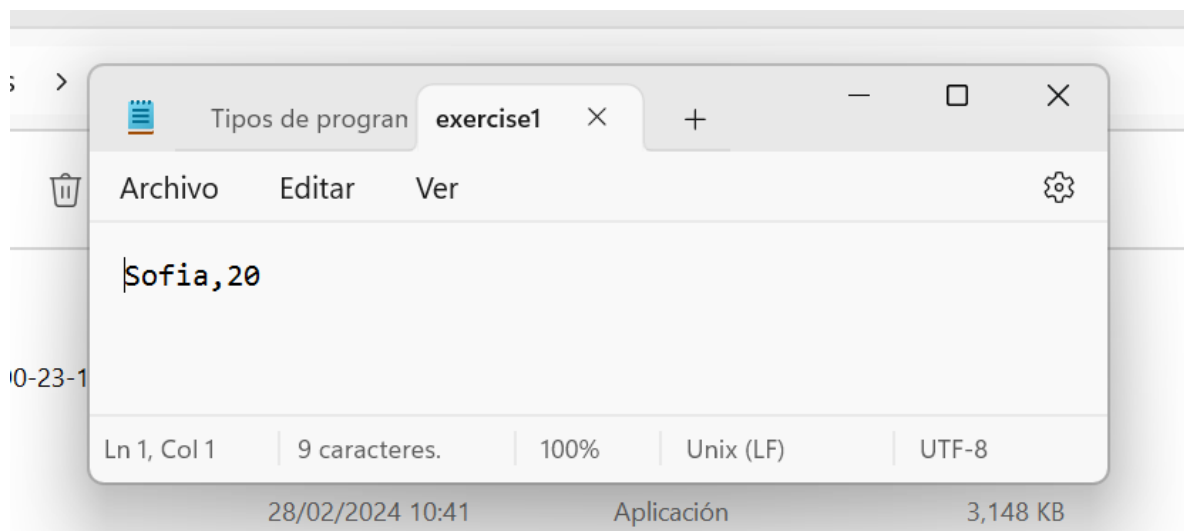
## Ejercicio 1 en ejecución



```
C:\Users\Stephanie Saban\Do x + - □ ×
Ingrese el nombre: Sofia
Ingrese la edad: 20
Datos guardados en el archivo.
Nombre: Sofia, Edad: 20

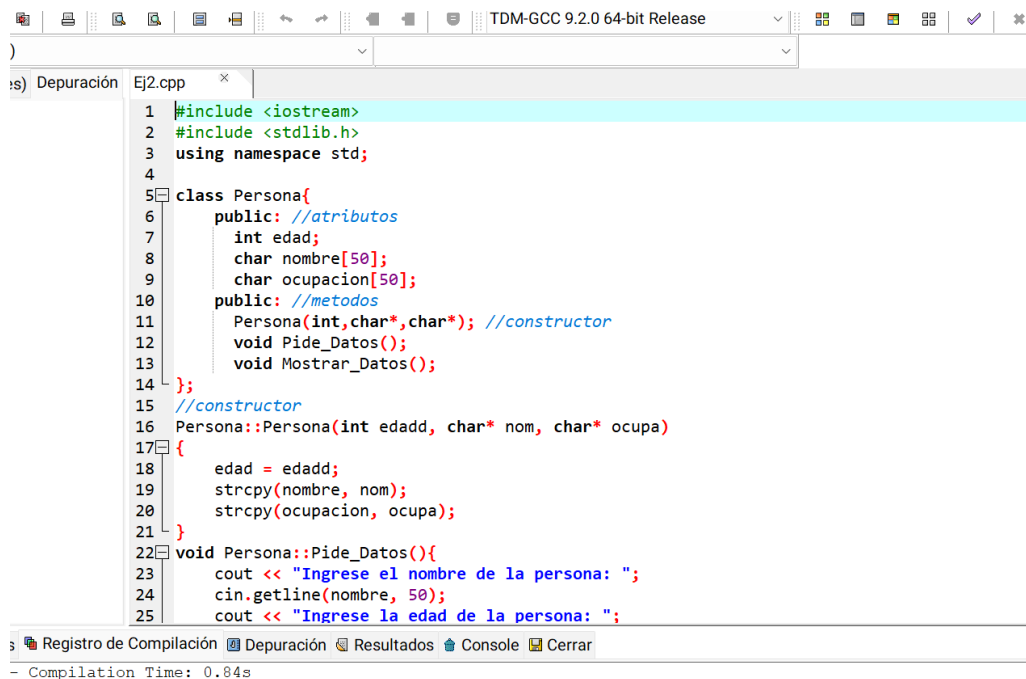
-----
Process exited after 17.75 seconds with return value
0
Presione una tecla para continuar . . . |
```

14 void Escribir(){



```
Tipos de program exercise1 x + - □ ×
Archivo Editar Ver
Sofia,20
Ln 1, Col 1 | 9 caracteres. | 100% | Unix (LF) | UTF-8
28/02/2024 10:41 Aplicación 3,148 KB
```

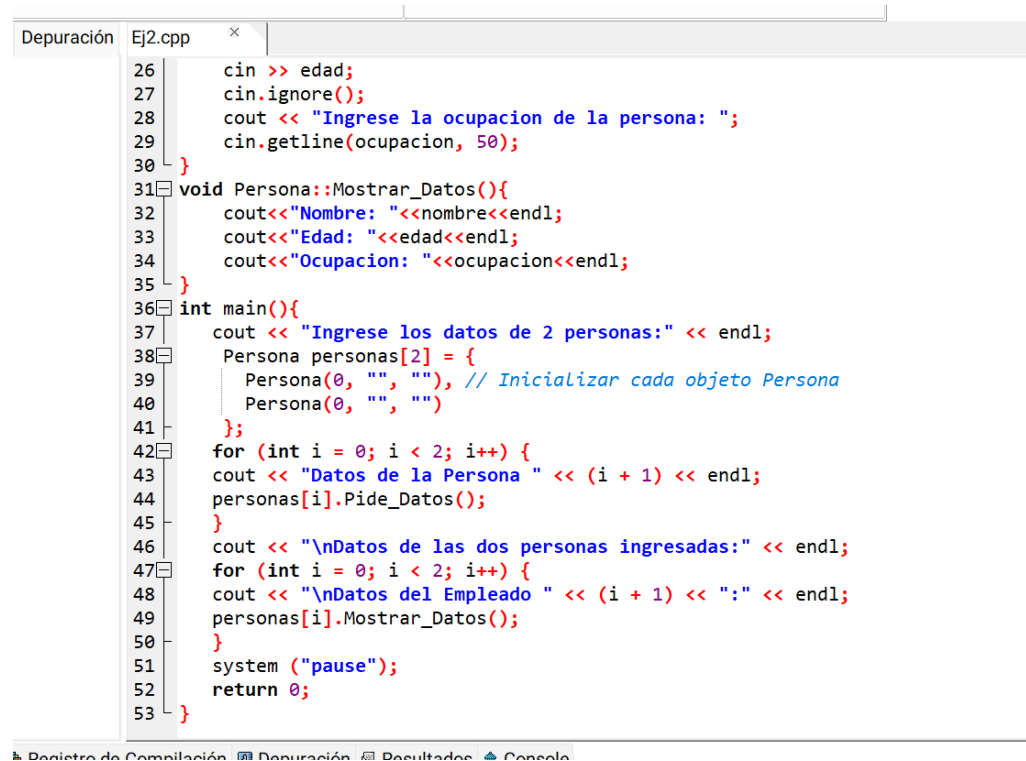
## Ejercicio 2 - Programación Orientada a Objetos



```
1 #include <iostream>
2 #include <stdlib.h>
3 using namespace std;
4
5 class Persona{
6 public: //atributos
7     int edad;
8     char nombre[50];
9     char ocupacion[50];
10 public: //metodos
11     Persona(int, char*, char*); //constructor
12     void Pide_Datos();
13     void Mostrar_Datos();
14 };
15 //constructor
16 Persona::Persona(int edadd, char* nom, char* ocupa)
17 {
18     edad = edadd;
19     strcpy(nombre, nom);
20     strcpy(ocupacion, ocupa);
21 }
22 void Persona::Pide_Datos(){
23     cout << "Ingrese el nombre de la persona: ";
24     cin.getline(nombre, 50);
25     cout << "Ingrese la edad de la persona: ";
```

Registro de Compilación Depuración Resultados Console Cerrar

Compilation Time: 0.84s



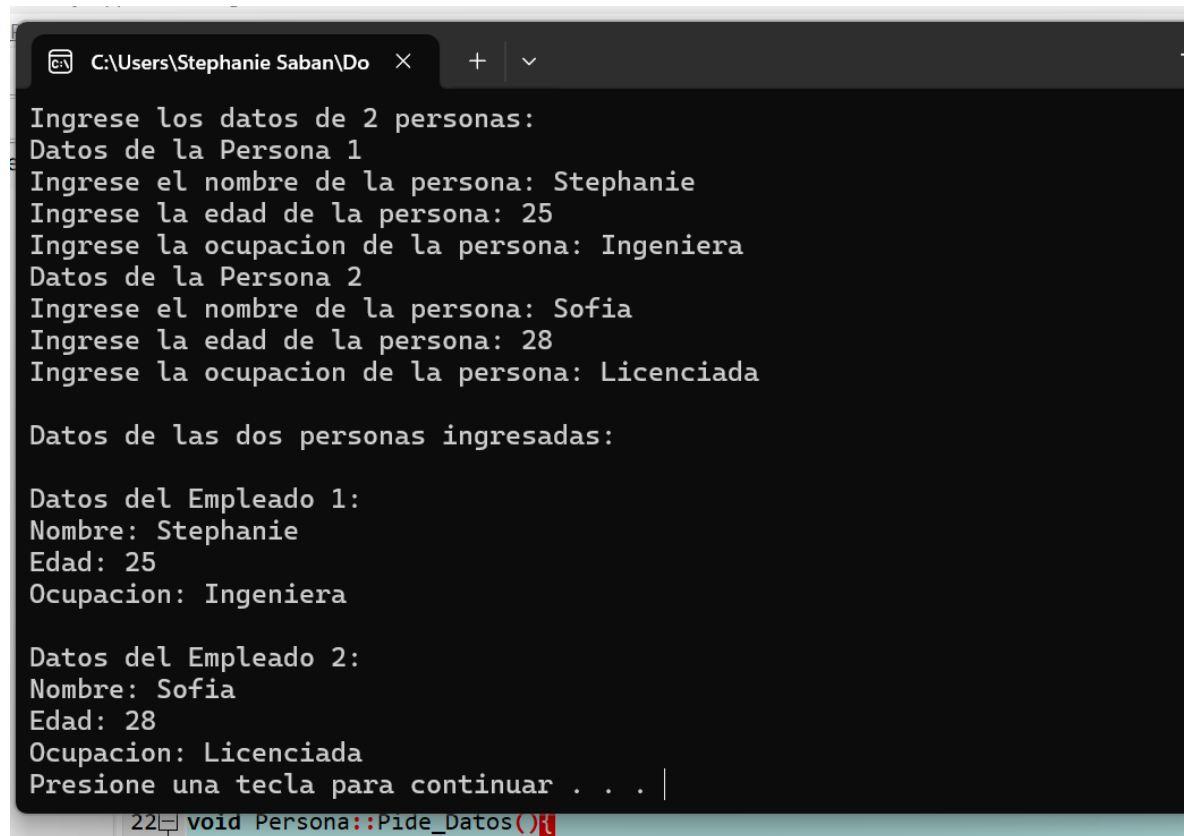
```
26     cin >> edad;
27     cin.ignore();
28     cout << "Ingrese la ocupacion de la persona: ";
29     cin.getline(ocupacion, 50);
30 }
31 void Persona::Mostrar_Datos(){
32     cout<<"Nombre: "<<nombre<<endl;
33     cout<<"Edad: "<<edad<<endl;
34     cout<<"Ocupacion: "<<ocupacion<<endl;
35 }
36 int main(){
37     cout << "Ingrese los datos de 2 personas:" << endl;
38     Persona personas[2] = {
39         Persona(0, "", ""), // Inicializar cada objeto Persona
40         Persona(0, "", "")
41     };
42     for (int i = 0; i < 2; i++) {
43         cout << "Datos de la Persona " << (i + 1) << endl;
44         personas[i].Pide_Datos();
45     }
46     cout << "\nDatos de las dos personas ingresadas:" << endl;
47     for (int i = 0; i < 2; i++) {
48         cout << "\nDatos del Empleado " << (i + 1) << ":" << endl;
49         personas[i].Mostrar_Datos();
50     }
51     system ("pause");
52     return 0;
53 }
```

Registro de Compilación Depuración Resultados Console

Para este programa definí una clase llamada Persona que tiene tres atributos: edad, nombre y ocupacion. La clase también incluye un constructor que inicializa estos atributos y dos métodos: Pide\_Datos() para solicitar al usuario que ingrese los datos de una persona, y Mostrar\_Datos() para imprimirlos en pantalla.

En el main(), se crea un arreglo de dos objetos de la clase Persona, inicializados con valores predeterminados. Luego, el programa solicita al usuario que ingrese los datos de las dos personas, y finalmente muestra los datos ingresados. Este código demuestra el uso de la programación orientada a objetos en C++, donde se define una clase con sus respectivos atributos y métodos, y se instancian objetos de esa clase para almacenar y procesar la información de las personas.

### Ejercicio 2 en ejecución



```
C:\Users\Stephanie Saban\Do x + v
Ingrese los datos de 2 personas:
Datos de la Persona 1
Ingrese el nombre de la persona: Stephanie
Ingrese la edad de la persona: 25
Ingrese la ocupacion de la persona: Ingeniera
Datos de la Persona 2
Ingrese el nombre de la persona: Sofia
Ingrese la edad de la persona: 28
Ingrese la ocupacion de la persona: Licenciada

Datos de las dos personas ingresadas:

Datos del Empleado 1:
Nombre: Stephanie
Edad: 25
Ocupacion: Ingeniera

Datos del Empleado 2:
Nombre: Sofia
Edad: 28
Ocupacion: Licenciada
Presione una tecla para continuar . . . |
22 void Persona::Pide_Datos()
```

## Ejercicio 3 - Abstracción e Instanciación

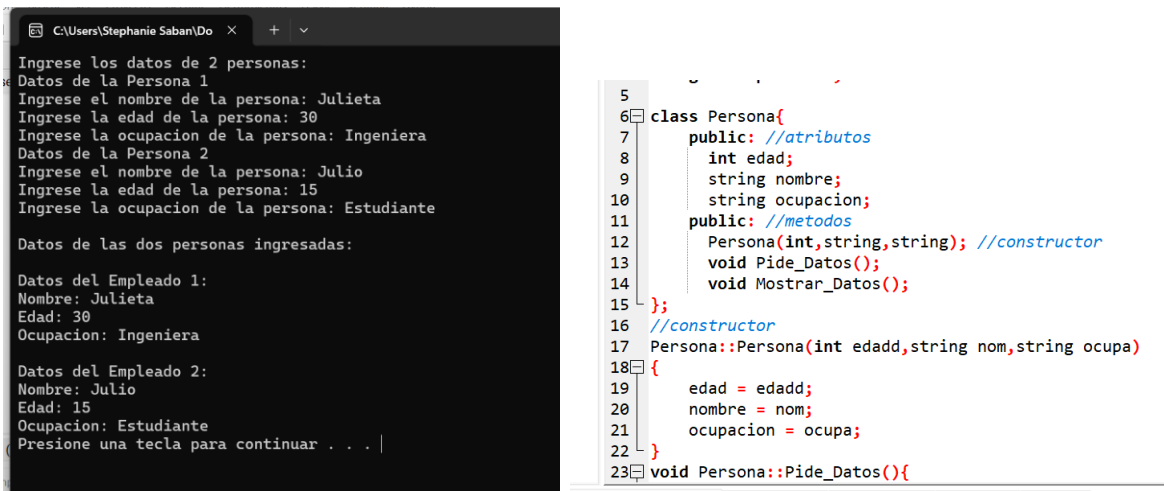
### Explicar el concepto de abstracción en la programación orientada a objetos y cómo se relaciona con la definición de clases.

La abstracción permite crear objetos que interactúan entre sí, es el proceso de identificar los aspectos esenciales de un objeto y representar esos aspectos de manera simplificada en el código. Esto nos ayuda a manejar la complejidad de los sistemas de software al dividirlos en partes más simples y manejables.

Cuando definimos una clase, estamos abstrayendo las características esenciales de un conjunto de objetos similares. Por ejemplo, si estamos en un sistema de gestión de empleados, podríamos definir una clase "Empleado" que tenga atributos como nombre, edad, salario, etc., y métodos para realizar acciones como calcular el salario o imprimir los detalles del empleado.

### Demostrar la instanciación de objetos utilizando la clase Persona creada en la actividad anterior.

Para empezar, cuando creamos una instancia de una clase, estamos creando un objeto específico que tiene sus propias variables y métodos, pero comparte la estructura y el comportamiento definidos por la clase. Para ejemplificarlo usaré el código anterior.



```

C:\Users\Stephanie Saban\Do x + v
Ingrese los datos de 2 personas:
Datos de la Persona 1
Ingrese el nombre de la persona: Julieta
Ingrese la edad de la persona: 30
Ingrese la ocupacion de la persona: Ingeniera
Datos de la Persona 2
Ingrese el nombre de la persona: Julio
Ingrese la edad de la persona: 15
Ingrese la ocupacion de la persona: Estudiante

Datos de las dos personas ingresadas:

Datos del Empleado 1:
Nombre: Julieta
Edad: 30
Ocupacion: Ingeniera

Datos del Empleado 2:
Nombre: Julio
Edad: 15
Ocupacion: Estudiante
Presione una tecla para continuar . . . |

5
6 class Persona{
7     public: //atributos
8         int edad;
9         string nombre;
10        string ocupacion;
11    public: //metodos
12        Persona(int,string,string); //constructor
13        void Pide_Datos();
14        void Mostrar_Datos();
15    };
16    //constructor
17    Persona::Persona(int edadd,string nom,string ocupa)
18    {
19        edad = edadd;
20        nombre = nom;
21        ocupacion = ocupa;
22    }
23    void Persona::Pide_Datos(){
```

Veamos cómo se instancian los objetos Persona en este ejemplo:

**Persona persona(30, "Julieta", "Ingeniera"); // imaginándose por el ejemplo**

Aquí se crea un objeto Persona llamado persona y se inicializa utilizando el constructor parametrizado Persona(int, string, string).

Los valores 30, "Julieta" y "Ingeniera" se pasan como argumentos al constructor, y se asignan a los atributos edad, nombre y ocupacion del objeto persona, respectivamente.

## Ejercicio 4 - Clases, Objetos y Métodos

```
ianie Saban\Documents\Ej4.cpp - Embarcadero Dev-C++ 6.3
Buscar Ver Proyecto Ejecutar Herramientas AStyle Ventana Ayuda
TDM-GCC 9.2.0 64-bit Release
(globals)
(Ej4.cpp) Depuración Ej4.cpp
1 #include<iostream>
2 #include<stdlib.h>
3 using namespace std;
4
5 class Persona{
6     private:
7         string nombre;
8         int edad;
9     public:
10         Persona(string, int);//constructor
11         void PedirDatos();
12         void MostrarDatos();
13 };
14 class Estudiante: public Persona{
15     private:
16         int numEstudiante;
17         float promNotas;
18     public:
19         Estudiante(string, int, int, float);//constructor de la clase Estudiante
20         void PedirDatos1();
21         void MostrarDatos1();
22 };
23 //constructor de la clase Persona(clase padre)
24 Persona::Persona(string nom,int edadd){
25     nombre=nom;
26     edad=edadd;
27 }
28 //constructor de la clase Estudiante (clase heredada)
29 Estudiante::Estudiante(string nom,int edadd, int num, float prom):Persona(nom, edadd){
30     numEstudiante = num;
31     promNotas= prom;
32 }
33 void Persona::PedirDatos(){
34     cout << "Ingresa el nombre de la Persona: ";
35     getline(cin,nombre);
36     cout << "Ingresa la edad de la Persona: ";
37     cin >> edad;
38     cin.ignore();
39 }
40 void Persona::MostrarDatos(){
41     cout<<"Nombre: "<<nombre<<endl;
42     cout<<"Edad: "<<edad<<endl;
43 }
44 void Estudiante::PedirDatos1(){
45     PedirDatos();
46     cout << "Ingresa el numero del estudiante: ";
```

Recursos Registro de Compilación Depuración Resultados Console Cerrar

Compilación

Compilation results...

-----

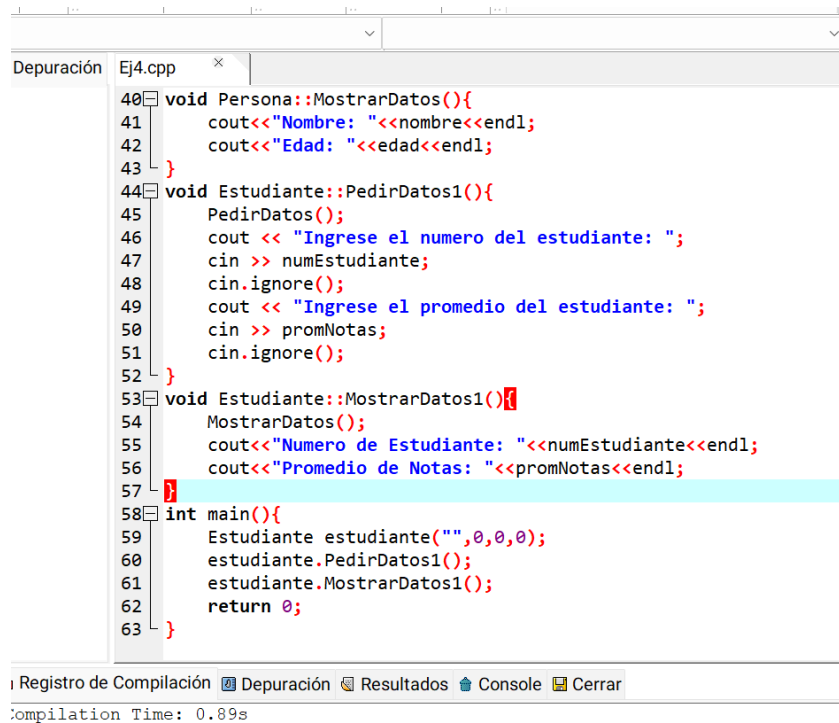
- Errors: 0

Ej4.cpp

```
22 };
23 //constructor de la clase Persona(clase padre)
24 Persona::Persona(string nom,int edadd){
25     nombre=nom;
26     edad=edadd;
27 }
28 //constructor de la clase Estudiante (clase heredada)
29 Estudiante::Estudiante(string nom,int edadd, int num, float prom):Persona(nom, edadd){
30     numEstudiante = num;
31     promNotas= prom;
32 }
33 void Persona::PedirDatos(){
34     cout << "Ingresa el nombre de la Persona: ";
35     getline(cin,nombre);
36     cout << "Ingresa la edad de la Persona: ";
37     cin >> edad;
38     cin.ignore();
39 }
40 void Persona::MostrarDatos(){
41     cout<<"Nombre: "<<nombre<<endl;
42     cout<<"Edad: "<<edad<<endl;
43 }
44 void Estudiante::PedirDatos1(){
45     PedirDatos();
46     cout << "Ingresa el numero del estudiante: ";
```

Registro de Compilación Depuración Resultados Console Cerrar





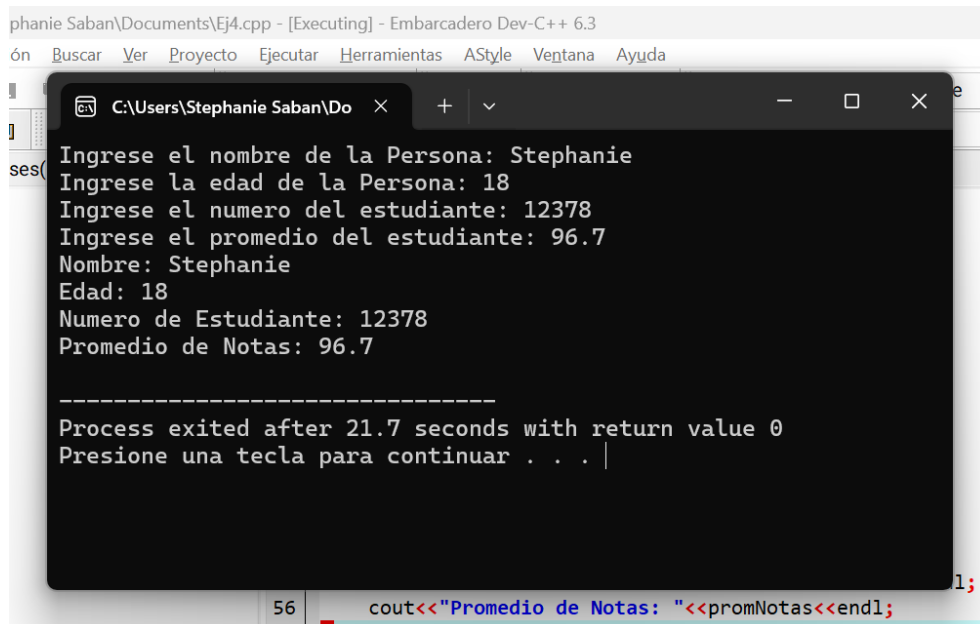
```
40 void Persona::MostrarDatos(){
41     cout<<"Nombre: "<<nombre<<endl;
42     cout<<"Edad: "<<edad<<endl;
43 }
44 void Estudiante::PedirDatos1(){
45     PedirDatos();
46     cout << "Ingrese el numero del estudiante: ";
47     cin >> numEstudiante;
48     cin.ignore();
49     cout << "Ingrese el promedio del estudiante: ";
50     cin >> promNotas;
51     cin.ignore();
52 }
53 void Estudiante::MostrarDatos1(){
54     MostrarDatos();
55     cout<<"Numero de Estudiante: "<<numEstudiante<<endl;
56     cout<<"Promedio de Notas: "<<promNotas<<endl;
57 }
58 int main(){
59     Estudiante estudiante("",0,0,0);
60     estudiante.PedirDatos1();
61     estudiante.MostrarDatos1();
62     return 0;
63 }
```

Registro de Compilación Depuración Resultados Console Cerrar

Compilation Time: 0.89s

En este código primero creamos las clases tanto la clase padre que en este caso es la clase Persona, le agregamos todos sus atributos y métodos y luego creamos la clase heredada que en este caso es la de Estudiante, donde ponemos sus propios atributos y hacemos referencia a los atributos de la clase padre, hacemos sus respectivos constructores, y empezamos con sus métodos para mostrar y pedir datos, en el main solo mandamos a llamar a las clases y creamos los objetos.

### Ejercicio 4 en ejecución



```
phanie Saban\Documents\Ej4.cpp - [Executing] - Embarcadero Dev-C++ 6.3
ón Buscar Ver Proyecto Ejecutar Herramientas AStyle Ventana Ayuda
C:\Users\Stephanie Saban\Do x + v - e
ses(
Ingrese el nombre de la Persona: Stephanie
Ingrese la edad de la Persona: 18
Ingrese el numero del estudiante: 12378
Ingrese el promedio del estudiante: 96.7
Nombre: Stephanie
Edad: 18
Numero de Estudiante: 12378
Promedio de Notas: 96.7

-----
Process exited after 21.7 seconds with return value 0
Presione una tecla para continuar . . . |
1;
56 cout<<"Promedio de Notas: "<<promNotas<<endl;
57 }
```

## Ejercicio 5 - Bibliotecas Estándar

En este código solo reemplazamos los arreglos de caracteres por el string del código anterior

```
Ej5.cpp
1 #include <iostream>
2 #include <stdlib.h>
3 #include <string>
4 using namespace std;
5
6 class Persona{
7     public: //atributos
8         int edad;
9         string nombre;
10        string ocupacion;
11    public: //metodos
12        Persona(int,string,string); //constructor
13        void Pide_Datos();
14        void Mostrar_Datos();
15 };
16 //constructor
17 Persona::Persona(int edadd,string nom,string ocupa)
18 {
19     edad = edadd;
20     nombre = nom;
21     ocupacion = ocupa;
22 }
23 void Persona::Pide_Datos(){
24     cout << "Ingrese el nombre de la persona: ";
25     getline(cin,nombre);
26     cout << "Ingrese la edad de la persona: ";
27     cin >> edad;
28     cin.ignore();
29     cout << "Ingrese la ocupacion de la persona: ";
30 }
31
32 void Persona::Mostrar_Datos(){
33     cout<<"Nombre: "<<nombre<<endl;
34     cout<<"Edad: "<<edad<<endl;
35     cout<<"Ocupacion: "<<ocupacion<<endl;
36 }
37 int main(){
38     cout << "Ingrese los datos de 2 personas:" << endl;
39     Persona personas[2] = {
40         Persona(0, "", ""), // Inicializar cada objeto Persona
41         Persona(0, "", "")
42     };
43     for (int i = 0; i < 2; i++) {
44         cout << "Datos de la Persona " << (i + 1) << endl;
45         personas[i].Pide_Datos();
46     }
47     cout << "\nDatos de las dos personas ingresadas:" << endl;
48     for (int i = 0; i < 2; i++) {
49         cout << "\nDatos del Empleado " << (i + 1) << ":" << endl;
50         personas[i].Mostrar_Datos();
51     }
52     system("pause");
53     return 0;
54 }
```

o de Compilación Depuración Resultados Console

0 Lines: 54 Length: 1389 Insertar Done parsing in 0.703 seconds

ars\Stephanie Saban\Documents\Ej5.cpp - Embarcadero Dev-C++ 6.3

Edición Buscar Ver Proyecto Ejecutar Herramientas AStyle Ventana Ayuda

TDM-GCC 9.2.0 64-bit Release

(globals)

o Clases(Funciones) Depuración Ej5.cpp

```
27     cin >> edad;
28     cin.ignore();
29     cout << "Ingrese la ocupacion de la persona: ";
30     getline(cin,ocupacion);
31 }
32 void Persona::Mostrar_Datos(){
33     cout<<"Nombre: "<<nombre<<endl;
34     cout<<"Edad: "<<edad<<endl;
35     cout<<"Ocupacion: "<<ocupacion<<endl;
36 }
37 int main(){
38     cout << "Ingrese los datos de 2 personas:" << endl;
39     Persona personas[2] = {
40         Persona(0, "", ""), // Inicializar cada objeto Persona
41         Persona(0, "", "")
42     };
43     for (int i = 0; i < 2; i++) {
44         cout << "Datos de la Persona " << (i + 1) << endl;
45         personas[i].Pide_Datos();
46     }
47     cout << "\nDatos de las dos personas ingresadas:" << endl;
48     for (int i = 0; i < 2; i++) {
49         cout << "\nDatos del Empleado " << (i + 1) << ":" << endl;
50         personas[i].Mostrar_Datos();
51     }
52     system("pause");
53     return 0;
54 }
```

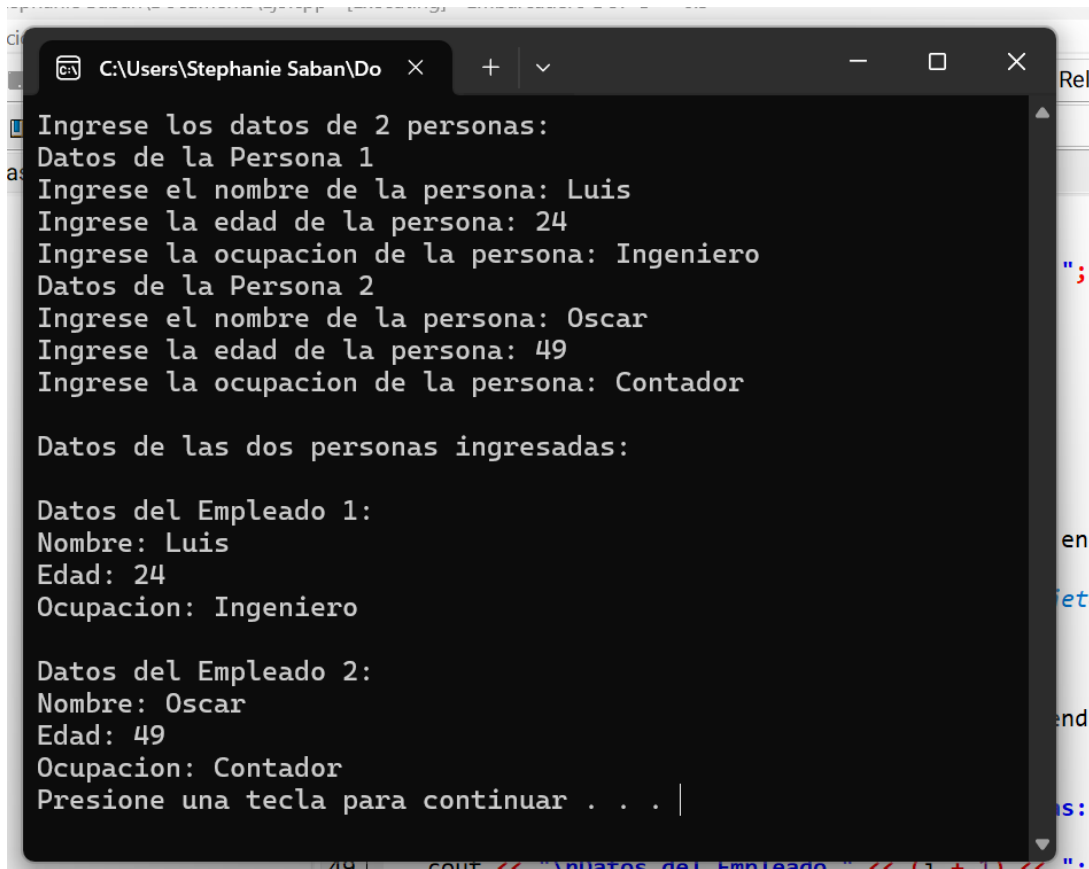
ilador Recursos Registro de Compilación Depuración Resultados Console

Col: 1 Sel: 0 Lines: 54 Length: 1389 Insertar Done parsing in 0.703 seconds

Algunas diferencias entre este reemplazo:

- Declaración de los atributos: En el código con arreglos de caracteres, los atributos nombre y ocupacion se declaran como arreglos de char de tamaño 50. En el código con string, los atributos nombre y ocupacion se declaran como objetos de la clase string.
- Lectura de datos: En el código con arreglos de caracteres, se utiliza `cin.getline()` para leer las cadenas de caracteres. En el código con string, se utiliza `getline(cin, nombre)` y `getline(cin, ocupacion)` para leer directamente en los objetos string.

### Ejercicio 5 en ejecución



```
C:\Users\Stephanie Saban\Do >
Ingrese los datos de 2 personas:
Datos de la Persona 1
Ingrese el nombre de la persona: Luis
Ingrese la edad de la persona: 24
Ingrese la ocupacion de la persona: Ingeniero
Datos de la Persona 2
Ingrese el nombre de la persona: Oscar
Ingrese la edad de la persona: 49
Ingrese la ocupacion de la persona: Contador

Datos de las dos personas ingresadas:

Datos del Empleado 1:
Nombre: Luis
Edad: 24
Ocupacion: Ingeniero

Datos del Empleado 2:
Nombre: Oscar
Edad: 49
Ocupacion: Contador
Presione una tecla para continuar . . . |
```

## Ejercicio 6 - Algoritmo de Ordenamiento

```
Ej6.cpp
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string nombre1, nombre2, nombre3; // Declaramos las variables para los nombres
7
8     // Pedimos el ingreso de los tres nombres y los guardamos en las variables respectivas
9     cout << "Ingrese el primer nombre: ";
10    getline(cin, nombre1);
11    cout << "Ingrese el segundo nombre: ";
12    getline(cin, nombre2);
13    cout << "Ingrese el tercer nombre: ";
14    getline(cin, nombre3);
15
16    // Empezamos con las condicionales para ordenar los nombres
17    if (nombre1 < nombre2 && nombre2 < nombre3) {
18        // Si nombre1 < nombre2 && nombre2 < nombre3
19        if (nombre2 < nombre3)
20            cout << "El orden correcto es (ascendente): " << nombre1 << ", " << nombre2 << ", " << nombre3 << endl;
21        else
22            cout << "El orden correcto es (ascendente): " << nombre1 << ", " << nombre3 << ", " << nombre2 << endl;
23    } else if (nombre2 < nombre1 && nombre2 < nombre3) {
24        // Si nombre2 < nombre1 y nombre2 < nombre3
25        if (nombre1 < nombre3)
26            cout << "El orden correcto es (ascendente): " << nombre2 << ", " << nombre1 << ", " << nombre3 << endl;
27        else
28            cout << "El orden correcto es (ascendente): " << nombre2 << ", " << nombre3 << ", " << nombre1 << endl;
29    } else if (nombre3 < nombre1 && nombre3 < nombre2) {
30        // Si nombre3 < nombre1 y nombre3 < nombre2
31        if (nombre1 < nombre2)
32            cout << "El orden correcto es (ascendente): " << nombre3 << ", " << nombre1 << ", " << nombre2 << endl;
33        else
34            cout << "El orden correcto es (ascendente): " << nombre3 << ", " << nombre2 << ", " << nombre1 << endl;
35    }
36
37    return 0;
38 }
```

Para este programa se solicita al usuario que ingrese tres nombres, los almacena en variables de tipo string, y luego utiliza una serie de condicionales para determinar y mostrar el orden ascendente de los nombres ingresados. Primero, compara los tres nombres y determina cuál es el menor o queda hasta atrás por orden de abecedario, luego compara los otros dos nombres para determinar el orden correcto. Finalmente, imprime en pantalla el orden ascendente de los nombres. Este programa demuestra el uso de variables de tipo string para manipular y comparar cadenas de texto, así como el uso de estructuras de control if-else para tomar decisiones y ordenar los nombres de forma adecuada.

### Ejercicio 6 en ejecución

```
C:\Users\Stephanie Saban\Do
Ingreso el primer nombre: Zendaya
Ingreso el segundo nombre: Abigail
Ingreso el tercer nombre: Grecia
El orden correcto es (ascendente): Abigail, Grecia, Zendaya

-----
Process exited after 20.11 seconds with return value 0
Presione una tecla para continuar . . . |
```

## Conclusión

En resumen, el manejo de archivos, la programación orientada a objetos, la abstracción, la instanciación, las clases y objetos, los métodos y funciones, así como el uso de bibliotecas como `string` e `iostream`, y los algoritmos de ordenamiento, son conceptos clave en la programación en C++.

Estos elementos forman la base para crear aplicaciones complejas y robustas en este lenguaje. El dominio de estos temas es esencial para cualquier desarrollador que quiera trabajar eficazmente con C++ y aprovechar al máximo sus capacidades.

Entender cómo interactúan estos diferentes aspectos de la programación en C++ permite a los programadores diseñar soluciones más estructuradas, modularizadas y eficientes. En conjunto, estos conceptos proporcionan a los desarrolladores las herramientas necesarias para abordar una amplia variedad de problemas y crear aplicaciones de alto rendimiento.

La definición de clases y objetos, junto con el uso de métodos y funciones, permite una mejor organización del código y una mayor reutilización. Como también el empleo de bibliotecas como `string` e `iostream` proporciona funcionalidades esenciales de una manera estandarizada y eficiente, ahorrando tiempo y esfuerzo a los programadores. Finalmente, el conocimiento de algoritmos de ordenamiento es fundamental para el manejo de grandes volúmenes de datos, optimizando el rendimiento y la usabilidad de las aplicaciones.

## Referencias

Repositorio con código fuente de cada uno de los programas

<https://github.com/StephanieSabann/Lab7StephanieSaban.git>