

Universidad Mariano Gálvez de Guatemala

Facultad de ingeniería en sistemas

Campus Villa nueva, Guatemala

Curso: Programación I

Docente: Inge. Carlos Arias



Actividad: Segundo Parcial

Nombre: Stephanie Cristina Sabán Cárcamo

Carnet: 5090- 23-11167

Sección: "A"

Fecha: 17/04/24

Introducción

En este informe para el parcial se incluye el manejo de archivos a través de la biblioteca `fstream`, que permite leer y escribir datos en archivos de texto y binarios, además del uso de POO, programación orientada a objetos, como las clases, objetos, abstracción, herencia etc.

La programación orientada a objetos (POO) y el manejo de archivos son dos conceptos fundamentales en el lenguaje de programación C++.

La POO permite encapsular datos y funcionalidades en objetos, facilitando la organización, reutilización y mantenimiento del código. Por otro lado, el manejo de archivos es crucial para almacenar y recuperar información de manera persistente. 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.

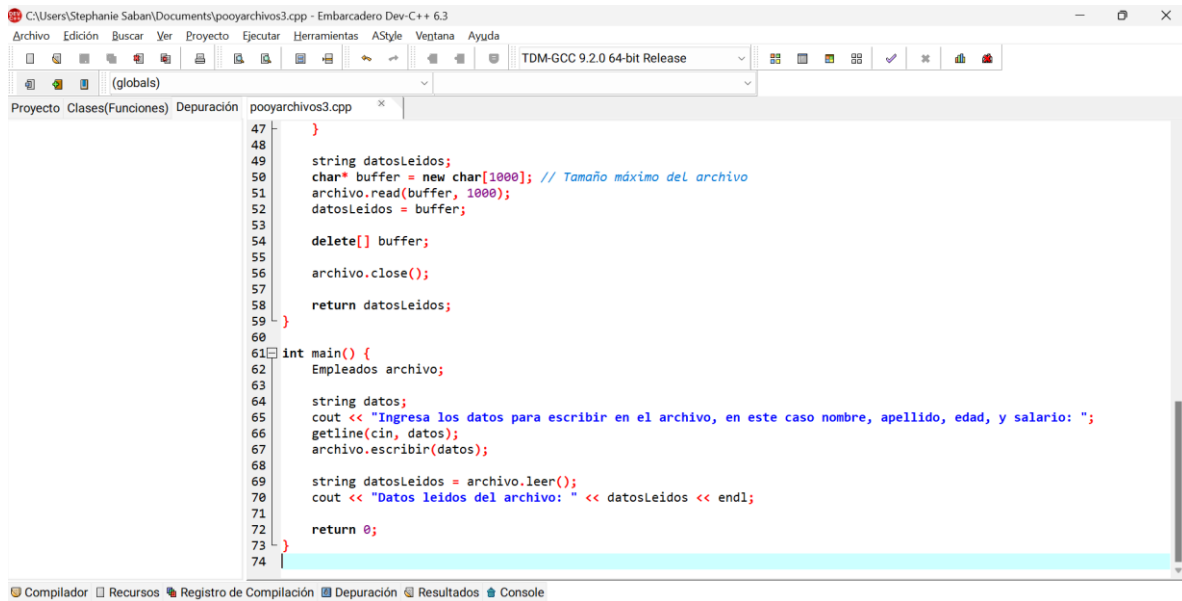
Programas

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4
5 using namespace std;
6
7 class Empleados {
8 public:
9     Empleado(string nom, string ape, int ed, int curs) {
10         nombre = nom;
11         apellido = ape;
12         edad = ed;
13         salario = sal;
14     }
15     void escribir(string datos);
16     void sobrescribir(int posicion, string datos);
17     string leer();
18
19 private:
20     fstream archivo;
21     string nombre;
22     string apellido;
23     int edad;
24     int salario;
25 };
26
27 void Empleados::escribir(string datos) {
28     archivo.open("Empleados.txt", ios::out | ios::binary);
29 }
```

En este programa primero se realiza la clase llamada empleados que tiene atributos privados como nombre, apellido, edad, salario, cada uno con sus respectivos tipos de datos, luego como publico puse el constructor de la clase, donde definimos cada atributo con otro subnombre, y declaramos los métodos en este caso escribir y leer.

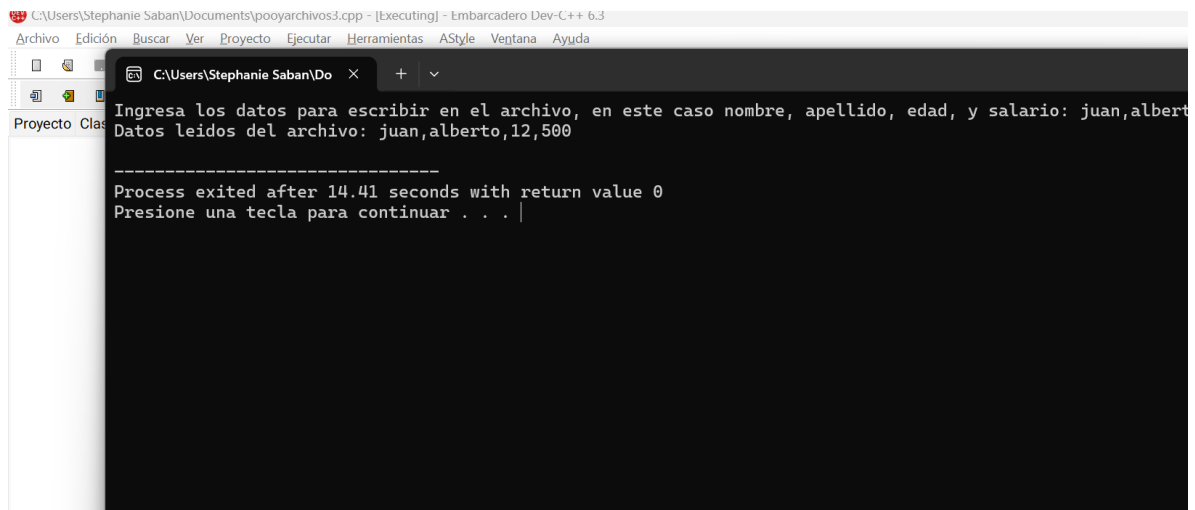
```
26 void Empleados::escribir(string datos) {
27     archivo.open("Empleados.txt", ios::out | ios::binary);
28
29     if (!archivo) {
30         cerr << "No se pudo abrir el archivo." << endl;
31         return;
32     }
33
34     archivo.write(datos.c_str(), datos.size());
35
36     archivo.close();
37 }
38
39
40
41 string Empleados::leer() {
42     archivo.open("Empleados.txt", ios::in | ios::binary);
43
44     if (!archivo) {
45         cerr << "No se pudo abrir el archivo." << endl;
46         return "";
47     }
48
49     string datosLeidos;
50     char* buffer = new char[1000]; // Tamaño máximo del archivo
51     archivo.read(buffer, 1000);
52     datosLeidos = buffer;
53     delete[] buffer;
54 }
```

Empezamos con el método o función para escribir los datos, donde abre el archivo con nombre “Empleados.txt” y luego escribe todo los datos que se ingresaron en main con un “write”, luego tenemos el método para leer todos los datos escritos en el archivo usando un “read” y luego cerrando el archivo.



```
47 }
48
49 string datosLeidos;
50 char* buffer = new char[1000]; // Tamaño máximo del archivo
51 archivo.read(buffer, 1000);
52 datosLeidos = buffer;
53
54 delete[] buffer;
55
56 archivo.close();
57
58 return datosLeidos;
59 }
60
61 int main() {
62     Empleados archivo;
63
64     string datos;
65     cout << "Ingresa los datos para escribir en el archivo, en este caso nombre, apellido, edad, y salario: ";
66     getline(cin, datos);
67     archivo.escribir(datos);
68
69     string datosLeidos = archivo.leer();
70     cout << "Datos leídos del archivo: " << datosLeidos << endl;
71
72     return 0;
73 }
74
```

En el main implementamos el objeto empleado archivo, y ingresamos todos los datos y los almacenamos en “datos” luego mandamos a llamar las funciones en este caso escribir y leer y nos muestra los datos escritos en el archivo.



```
C:\Users\Stephanie Saban\Documents\pooyarchivos3.cpp - [Executing] - Embarcadero Dev-C++ 6.3
Ingresa los datos para escribir en el archivo, en este caso nombre, apellido, edad, y salario: juan,alberto,12,500
Datos leídos del archivo: juan,alberto,12,500
-----
Process exited after 14.41 seconds with return value 0
Presione una tecla para continuar . . . |
```

Conclusión

El enfoque de POO ha demostrado ser sumamente valioso en el desarrollo de programas que trabajan con archivos. Al encapsular las operaciones de lectura, escritura y manipulación de archivos en clases, se logra un código más organizado, reutilizable y fácil de mantener. Esto se evidencia en los programas realizados, donde la clase permite abstraer y simplificar el acceso a los archivos, sin que el código principal se vea sobrecargado con detalles de bajo nivel.

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 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.

En conclusión, la combinación de POO y manejo de archivos en C++ ha demostrado ser una estrategia eficaz para desarrollar aplicaciones robustas, escalables y fáciles de mantener. Los programas realizados han puesto en evidencia los beneficios de esta integración, brindando una sólida base para la construcción de soluciones de software más sofisticadas y versátiles.

Referencias

Repositorio con código fuente de cada uno de los programas

<https://github.com/StephanieSabann/Parcial2Progra.git>