

Universidad Mariano Gálvez de Guatemala

Campus Villa Nueva, Guatemala

Ingeniería en Sistemas

Ing. Carlos Arias

Curso: Programación I

Laboratorio 7

Sección: “A”

Carné: 5090-23-1407

Nombre: Angelyn Judith Díaz Zeceña

Introducción

Como podemos observar, en este tipo de programa se desarrolló muchos temas, uno de ellos, es el manejo adecuado sobre manipulación de archivos, a utilizar bibliotecas como: `fstream`, `string`, a implementar un algoritmo de ordenamiento entre los nombres de personas ingresados por el usuario. Entre otros temas que, más adelante podrá observar a continuación.

Contenido

```
1 //Angelyn Judith Diaz Zeceña 5090-23-1407
2 //Laboratorio 7
3 #include <iostream>
4 #include <fstream>
5 #include <string>
6 #include <vector>
7 #include <algorithm>
8
9 using namespace std;
10
11 // aqui se define la clase Persona
12 class Persona {
13 private:
14     string nombre;
15     int edad;
16     string ocupacion;
17
18 public:
19     Persona(string n, int e, string o) : nombre(n), edad(e), ocupacion(o) {}
20
21     // Métodos para establecer y obtener valores de los atributos
22     void setNombre(string n) { nombre = n; }
23     void setEdad(int e) { edad = e; }
24     void setOcupacion(string o) { ocupacion = o; }
25
26     string getNombre() const { return nombre; }
27     int getEdad() const { return edad; }
28     string getOcupacion() const { return ocupacion; }
29 };
30
```

Como podemos observar en esta parte del código, empieza con las bibliotecas para caracteres da la definición de la clase Persona, llamada: class “persona” que incluye atributos como nombre, edad y ocupación, acompañados de un string, así como métodos para establecer y obtener los valores de estos atributos.

```

30
31 // Función para guardar datos en un archivo de texto
32 void guardarDatos(const vector<Persona>& personas) {
33     ofstream archivo("datos.txt");
34     if (archivo.is_open()) {
35         for (const auto& persona : personas) {
36             archivo << persona.getNombre() << " " << persona.getEdad() << " " << persona.getOcupacion() << endl;
37         }
38         archivo.close();
39         cout << "Datos guardados correctamente en el archivo 'datos.txt'" << endl;
40     } else {
41         cout << "Error al abrir el archivo 'datos.txt'" << endl;
42     }
43 }
44
45 // Función para leer datos desde un archivo de texto
46 void leerDatos() {
47     ifstream archivo("datos.txt");
48     if (archivo.is_open()) {
49         string nombre, ocupacion;
50         int edad;
51         while (archivo >> nombre >> edad >> ocupacion) {
52             cout << "Nombre: " << nombre << ", Edad: " << edad << ", Ocupación: " << ocupacion << endl;
53         }
54         archivo.close();
55     } else {
56         cout << "No se pudo abrir el archivo 'datos.txt'" << endl;
57     }
58 }

```

En esta otra parte del código, se abre un objeto de tipo ofstream llamado archivo con el nombre del archivo como argumento. Esto crea un nuevo archivo de texto llamado "datos.txt" para escribir en él, y verifica si el archivo se ha abierto correctamente utilizando el método is_open() de la clase ofstream. Si el archivo se abre correctamente, el proceso continúa y se itera a través del vector de personas usando un bucle for.

```

// Aquí se hereda Clase Estudiante a Persona
class Estudiante : public Persona {
private:
    string numEstudiante;
    float promedioCalificaciones;

public:
    Estudiante(string n, int e, string o, string ne, float pc)
        : Persona(n, e, o), numEstudiante(ne), promedioCalificaciones(pc) {}

    void setNumEstudiante(string ne) { numEstudiante = ne; }
    void setPromedioCalificaciones(float pc) { promedioCalificaciones = pc; }

    string getNumEstudiante() const { return numEstudiante; }
    float getPromedioCalificaciones() const { return promedioCalificaciones; }
};

// Función para ordenar una lista de nombres de estudiantes
void ordenarNombres(vector<string>& nombres) {
    sort(nombres.begin(), nombres.end());
}

int main() {
    vector<Persona> personas;
    vector<string> nombres;
}

```

Como podemos ver, en esta parte del código, define una clase Estudiante que hereda de la clase Persona e incluye atributos y métodos adicionales específicos de los estudiantes, como el número de estudiante y el promedio de calificaciones. También proporciona una función para ordenar una lista de nombres de estudiantes. Gracias a la ayuda de un void, un vector y string.

```
        cout << "Edad: ";
        cin >> edad;
        cout << "Ocupación: ";
        cin >> ocupacion;

        // Crear objeto Persona y agregarlo al vector de personas
        personas.push_back(Persona(nombre, edad, ocupacion));
        nombres.push_back(nombre); // Para el algoritmo de ordenamiento
    }

    guardarDatos(personas);

    // Aquí Lee los datos del archivo y mostrarlos en pantalla
    cout << "\nDatos almacenados en el archivo 'datos.txt':" << endl;
    leerDatos();

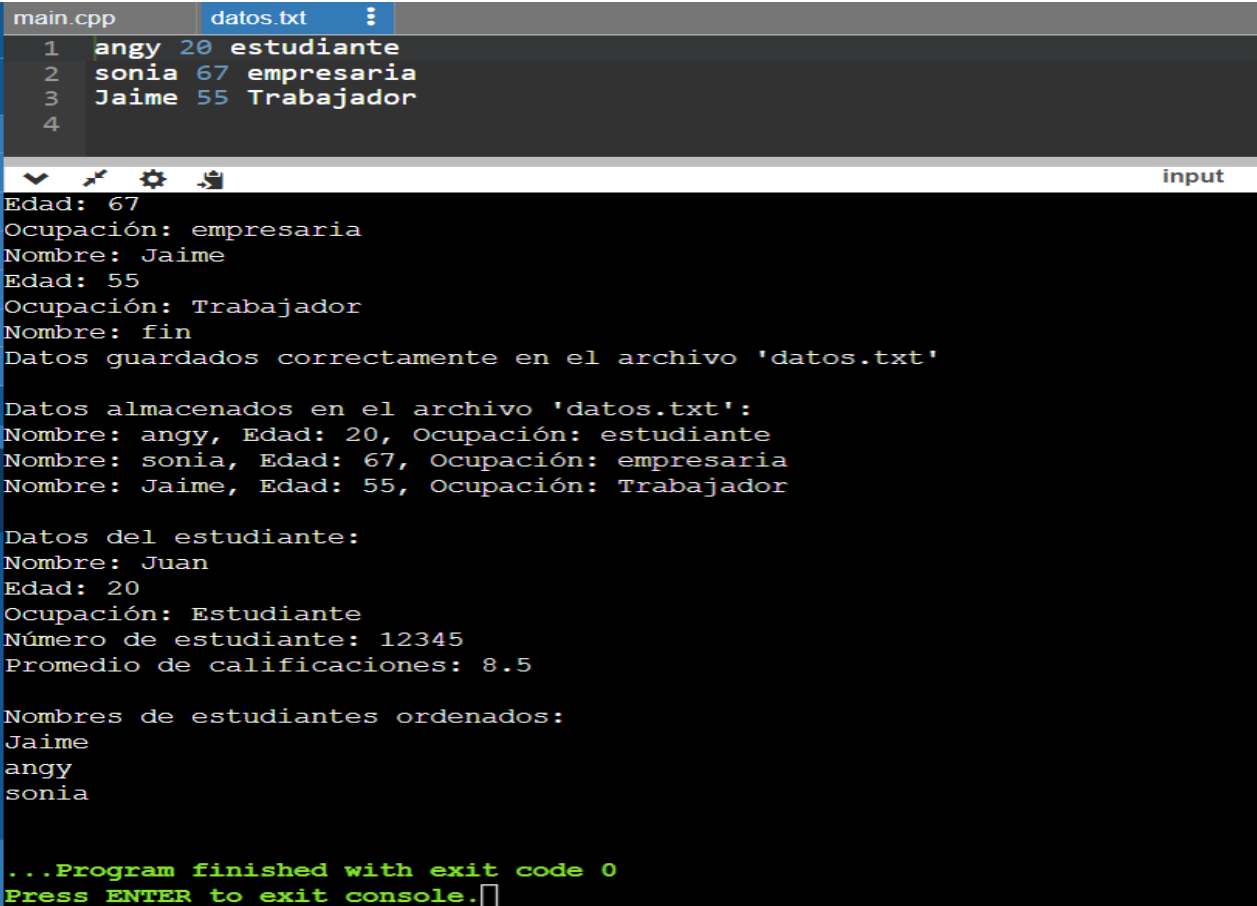
    // Aquí esta la instancia objetos de la clase Estudiante
    Estudiante estudiante("Juan", 20, "Estudiante", "12345", 8.5);
    cout << "\nDatos del estudiante:" << endl;
    cout << "Nombre: " << estudiante.getNombre() << endl;
    cout << "Edad: " << estudiante.getEdad() << endl;
    cout << "Ocupación: " << estudiante.getOcupacion() << endl;
    cout << "Número de estudiante: " << estudiante.getNumEstudiante() << endl;
    cout << "Promedio de calificaciones: " << estudiante.getPromedioCalificaciones() << endl;

    ordenarNombres(nombres);
    cout << "\nNombres de estudiantes ordenados:" << endl;
    for (const auto& nombre : nombres) {
        cout << nombre << endl;
    }

    return 0;
}
//Aquí finaliza el programa
```

En esta última parte del código, podemos observar que se le permite al usuario ingresar datos de personas, los almacena en un archivo de texto, los lee desde el archivo y los muestra en la pantalla. Además, instancia un objeto de la clase Estudiante con datos predefinidos y muestra los nombres de los estudiantes ordenados alfabéticamente. Inclusive, tomando la ayuda de un bucle while, para permitir al usuario ingresar múltiples conjuntos de datos de personas.

Finalizando con un return 0. Que quiere decir que el programa ha sido ejecutado exitosamente.



```
main.cpp | datos.txt |  
1 angy 20 estudiante  
2 sonia 67 empresaria  
3 Jaime 55 Trabajador  
4  
  
input  
Edad: 67  
Ocupación: empresaria  
Nombre: Jaime  
Edad: 55  
Ocupación: Trabajador  
Nombre: fin  
Datos guardados correctamente en el archivo 'datos.txt'  
  
Datos almacenados en el archivo 'datos.txt':  
Nombre: angy, Edad: 20, Ocupación: estudiante  
Nombre: sonia, Edad: 67, Ocupación: empresaria  
Nombre: Jaime, Edad: 55, Ocupación: Trabajador  
  
Datos del estudiante:  
Nombre: Juan  
Edad: 20  
Ocupación: Estudiante  
Número de estudiante: 12345  
Promedio de calificaciones: 8.5  
  
Nombres de estudiantes ordenados:  
Jaime  
angy  
sonia  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Como podemos observar en esta captura de pantalla, se ingresaron los nombres de las personas, su edad y que ocupación tenían. Y al finalizar el ingreso de datos, se colocó fin, para que el programa creyera automáticamente que ya no se iban a ingresar más datos. Y en donde fueron guardados.

Después, se ingresa datos del estudiante, y ya finaliza la captura, mostrando los nombres ordenados en forma de listado.

Conclusión

En conclusión, en este programa, en donde se utilizaron muchos tipos de funciones relacionadas a objetos. Como, proporcionar una estructura básica para interactuar con datos de personas y estudiantes en C++. Utiliza clases para modelar entidades como Persona y Estudiante, lo que permite organizar y manipular los datos de manera eficiente. Además, emplea funciones para guardar y leer datos desde un archivo de texto, lo que facilita la persistencia de la información entre diferentes ejecuciones del programa.

Referencias

<https://github.com/Ashe122/LABORATORIO-7.git>