Universidad Mariano Gálvez de Guatemala

Carrera de Ingeniería en Sistemas – 5090

Curso: Programación I

Docente: Ing. Carlos Alejandro Arias

Laboratorio 7

Archivos y POO

Pablo Javier Roldán Vásquez

Carné: 5090-23-13164

Fecha 08/04/2024

Introducción

En el ámbito educativo de la informática, la práctica activa y el fortalecimiento de los conocimientos adquiridos son fundamentales para el desarrollo de habilidades sólidas en programación. Este laboratorio fue asignado con el propósito claro de proporcionar a nosotros los estudiantes actividades para fortalecer nuestra comprensión sobre los temas vistos en clase, a través de la manipulación de archivos, la implementación de conceptos de programación orientada a objetos y el dominio de las bibliotecas estándar.

1. Manipulación de archivos (Primer programa)

Función para almacenar los datos en un archivo

```
// Función para almacenar los datos en un archivo
void guardarDatos(const string& nombreArchivo) {
    ofstream archivo(nombreArchivo); // Crea un objeto de archivo de salida para escribir en el archivo especificado

    if (!archivo) { // Verifica si el archivo se abrió correctamente
        cerr << "Error al abrir el archivo " << nombreArchivo << endl;
        return;
    }
```

Cantidad de personas para agregar

```
 int cantidadPersonas;
 cout << "Ingrese la cantidad de personas: "; // Solicita al usuario la cantidad de personas
 cin >> cantidadPersonas;

 for (int i = 0; i < cantidadPersonas; ++i) { // Bucle para ingresar los datos de cada persona
     string nombre;
     int edad;
     system("cls");
     cout << "Ingrese el nombre del Estudiante " << i + 1 << ": ";
     cin >> nombre;
     cout << "Ingrese la edad del Estudiante " << i + 1 << ": ";
     cin >> edad;

     archivo << nombre << " " << edad << endl; // Ingresa el nombre y la edad en el archivo
 }

 archivo.close(); // Cierra el archivo después de escribir los datos
 cout << "Datos guardados correctamente" << nombreArchivo << endl;
 system("pause");
```

Función para mostrar los datos

```
// Función para leer los datos del archivo y mostrarlos en la pantalla
void mostrarDatos(const string& nombreArchivo) {
    ifstream archivo(nombreArchivo);

    if (!archivo) { // Verifica si el archivo se abrió correctamente
        cerr << "Error al abrir el archivo " << nombreArchivo << endl;
        return;
    }

    string nombre;
    int edad;
    system("cls");
    cout << "Datos almacenados" << nombreArchivo << ":" << endl;
    while (archivo >> nombre >> edad) { // Bucle para leer los datos del archivo
        cout << "Nombre: " << nombre << ", Edad: " << edad << endl; // Muestra el nombre y la ed
    }

    archivo.close();
}
```

2. Programación orientada a objetos (Segundo programa)

Definición de la clase persona

```
// Definición de la clase Persona
class Persona {
private:
    string nombre; // Atributo privado para almacenar el nombre de la persona
    int edad; // Atributo privado para almacenar la edad de la persona
    string ocupacion; // Atributo privado para almacenar la ocupación de la persona
```

Constructor que inicia los atributos de la clase y los métodos para get, set al igual que la impresión de datos de la persona

```
public:
    // Constructor que inicializa los atributos de la clase
    Persona(string nombre, int edad, string ocupacion) : nombre(nombre), edad(edad), ocupacion(ocupacion) {}

    // Métodos para acceder a los atributos (get)
    string getNombre() const { return nombre; }
    int getEdad() const { return edad; }
    string getOcupacion() const { return ocupacion; }

    // Métodos para modificar los atributos (set)
    void setNombre(string nombre) { this->nombre = nombre; }
    void setEdad(int edad) { this->edad = edad; }
    void setOcupacion(string ocupacion) { this->ocupacion = ocupacion; }

    // Método para imprimir los datos de la persona
    void mostrarDatos() const {
        cout << "Nombre: " << nombre << endl;
        cout << "Edad: " << edad << endl;
        cout << "Ocupacion: " << ocupacion << endl;
    }
};
```

Menu del programa

```
int main() {
    vector<Persona> personas;
    int opcion;

    do {
        system("cls");
        cout << "Menu:" << endl;
        cout << "1. Agregar persona" << endl;
        cout << "2. Modificar datos de persona" << endl;
        cout << "3. Leer datos de personas" << endl;
        cout << "4. Salir" << endl;
        cout << "Seleccione una opcion: ";
        cin >> opcion;
```

Código para agregar a una persona

```
switch (opcion) {
    case 1: {
        system("cls");
        string nombre, ocupacion;
        int edad;
        cout << "Ingrese el nombre: ";
        cin >> nombre;
        cout << "Ingrese la edad: ";
        cin >> edad;
        cout << "Ingrese la ocupacion: ";
        cin >> ocupacion;
        personas.push_back(Persona(nombre, edad, ocupacion));
        cout << "Persona agregada exitosamente." << endl;
        system("pause");
        break;
    }
```

Código para modificar los datos de una persona

```
case 2: {
    int indice;
    system("cls");
    cout << "Ingrese el indice de la persona a modificar (0,1,2... segun el orden de la persona): ";
    cin >> indice;
    if (indice >= 0 && indice < personas.size()) { // Verifica si el índice es válido
        string nombre, ocupacion;
        int edad;
        cout << "Ingrese el nuevo nombre: ";
        cin >> nombre;
        cout << "Ingrese la nueva edad: ";
        cin >> edad;
        cout << "Ingrese la nueva ocupacion: ";
        cin >> ocupacion;
        personas[indice].setNombre(nombre); // Modifica el nombre de la persona
        personas[indice].setEdad(edad); // Modifica la edad de la persona
        personas[indice].setOcupacion(ocupacion); // Modifica la ocupación de la persona
        cout << "Datos de persona modificados exitosamente." << endl;
        system("pause");
    } else {
        cout << "Índice inválido." << endl;
    }
    break;
```

Código para mostrar en pantalla los datos de las personas

```
case 3: {
    system("cls");
    cout << "Datos de personas:" << endl;
    for (const auto& persona : personas) { // Itera
        persona.mostrarDatos();
        cout << endl;
    }
    system("pause");
    break;
```

3. **Abstracción e Instanciación**

Abstracción en la programación orientada a objetos (POO): Es un principio fundamental que permite a los desarrolladores representar entidades del mundo real como objetos de software, se trata de simplificar la representación de un objeto para capturar solo lo que es importante para el problema que se está resolviendo.

¿Como se relaciona la abstracción con la definición de clases?: La relación entre abstracción y definición de clases es que las clases son la herramienta principal que utilizamos para aplicar la abstracción en la POO. Al definir una clase, estamos creando una abstracción de los objetos que queremos modelar en nuestro sistema. La abstracción nos permite manejar la complejidad al enfocarnos en los aspectos importantes de los objetos y sus interacciones, lo que facilita el diseño, la implementación y el mantenimiento del código.

```
case 1: {
    system("cls");
    string nombre, ocupacion;
    int edad;
    cout << "Ingrese el nombre: ";
    cin >> nombre;
    cout << "Ingrese la edad: ";
    cin >> edad;
    cout << "Ingrese la ocupacion: ";
    cin >> ocupacion;
    personas.push_back(Persona(nombre, edad, ocupacion));
    cout << "Persona agregada exitosamente." << endl;
    system("pause");
    break;
}
```

La instanciación de objetos de la clase “Persona” se realiza en la parte del código donde se agregan personas al vector “personas” en la función main(). En este bloque de código, se crea una nueva instancia de Persona utilizando el constructor de la clase Persona, pasando los valores ingresados por el usuario para el nombre, la edad y la ocupación.

4. Clases, Objetos y Métodos

Definición de la clase persona

```
// Definición de la clase Persona
class Persona {
protected:
    string nombre; // Atributo protegido para
    int edad; // Atributo protegido para alma
```

Definición de la clase Estudiante que hereda de persona

```
// Definición de la clase Estudiante que hereda de Persona
class Estudiante : public Persona {
private:
    int numeroEstudiante; // Atributo privado para almacenar el número de estudiante
    float promedioCalificaciones; // Atributo privado para almacenar el promedio de calificaciones
public:
    Estudiante(string _nombre = "", int _edad = 0, int _numeroEstudiante = 0, float _promedioCalificaciones = 0.0)
        : Persona(_nombre, _edad), numeroEstudiante(_numeroEstudiante), promedioCalificaciones(_promedioCalificaciones) {}

    // Métodos para establecer y obtener el número de estudiante y el promedio de calificaciones
    void setNumeroEstudiante(int _numeroEstudiante) {
        numeroEstudiante = _numeroEstudiante;
    }

    int getNumeroEstudiante() {
        return numeroEstudiante;
    }

    void setPromedioCalificaciones(float _promedioCalificaciones) {
        promedioCalificaciones = _promedioCalificaciones;
    }

    float getPromedioCalificaciones() {
        return promedioCalificaciones;
    }
```

Conclusión

Este laboratorio ha representado una valiosa oportunidad para que podamos profundizar en nuestras habilidades de programación en C++. La combinación de la manipulación de archivos, la implementación de conceptos orientados a objetos y la comprensión de las bibliotecas estándar ha proporcionado un entorno de aprendizaje integral que va más allá de la mera comprensión teórica.

Referencias

Github: https://github.com/PabloRoldan2/Laboratorio-7.git