Universidad Mariano Gálvez De Guatemala

Facultad de ingeniería, Matemática y ciencias Físicas

Campus Villa Nueva Guatemala

Ingeniería en Sistemas de Información y ciencias de la computación

Curso: programación 1

Código de curso: 012

Código de la carrera: 5090

Estudiante: COOSEMANS Spiegeler Max Eduardo

Sección: A

Numero de carné: 5090-23-3282

Fecha: 08/04/2024

**INTRODUCCION:**

En este laboratorio, exploramos varios conceptos clave de la programación en C++, incluida la manipulación de archivos, la programación orientada a objetos, la abstracción, la instanciación, el uso de bibliotecas estándar y la implementación de algoritmos básicos como el ordenamiento. A través de una serie de actividades prácticas, pudimos aplicar estos conceptos y mejorar nuestras habilidades de programación en C++.En este ejemplo, utilizamos la clase std::string de la biblioteca estándar de C++ para manejar cadenas de caracteres de manera más flexible y segura que los arreglos de caracteres tradicionales. Demostramos cómo ingresar, concatenar, obtener la longitud, acceder a caracteres individuales, verificar si una cadena está vacía y recorrer una cadena utilizando bucles.

Contenido:

Manipulación de Archivos: Aprendimos a usar la biblioteca fstream para leer y escribir datos desde y hacia archivos en C++. Implementamos un programa que permite al usuario ingresar datos y almacenarlos en un archivo de texto, luego leímos esos datos desde el archivo y los mostramos en la pantalla.

Programación Orientada a Objetos: Definimos una clase Persona con atributos como nombre, edad y ocupación, y métodos para establecer y obtener los valores de estos atributos. Creamos objetos de esta clase y utilizamos los métodos para modificar y mostrar la información de cada persona.

Abstracción e Instanciación: Exploramos el concepto de abstracción en la programación orientada a objetos y cómo se relaciona con la definición de clases. Demostramos la instanciación de objetos utilizando la clase Persona.

Clases, Objetos y Métodos: Desarrollamos una clase Estudiante que hereda de la clase Persona y tiene atributos adicionales como número de estudiante y promedio de calificaciones. Implementamos métodos para establecer y obtener los valores de estos atributos y utilizamos los métodos heredados y propios.

Bibliotecas Estándar: Utilizamos la biblioteca string para manipular cadenas de caracteres en lugar de arreglos de caracteres en nuestros programas anteriores, lo que nos proporcionó una interfaz más segura y fácil de usar.

Algoritmo de Ordenamiento: Implementamos el algoritmo de ordenamiento de burbuja para ordenar una lista de nombres de estudiantes, lo que nos permitió comprender mejor los algoritmos de ordenamiento y su aplicación práctica.

1. Manipulación de Archivos en C++

#include <iostream>

#include <fstream>

#include <string>

void guardarDatosEnArchivo(const std::string& nombreArchivo) {

std::ofstream archivo(nombreArchivo);

if (archivo.is\_open()) {

std::string nombre;

int edad;

std::cout << "Ingrese el nombre y la edad (separados por espacio, escriba 'fin' para terminar):\n";

while (true) {

std::cin >> nombre;

if (nombre == "fin") break;

std::cin >> edad;

archivo << nombre << " " << edad << std::endl;

}

archivo.close();

} else {

std::cerr << "No se pudo abrir el archivo.\n";

}

}

void mostrarDatosDeArchivo(const std::string& nombreArchivo) {

std::ifstream archivo(nombreArchivo);

if (archivo.is\_open()) {

std::string nombre;

int edad;

std::cout << "Contenido del archivo:\n";

while (archivo >> nombre >> edad) {

std::cout << "Nombre: " << nombre << ", Edad: " << edad << std::endl;

}

archivo.close();

} else {

std::cerr << "No se pudo abrir el archivo.\n";

}

}

int main() {

const std::string nombreArchivo = "datos.txt";

guardarDatosEnArchivo(nombreArchivo);

mostrarDatosDeArchivo(nombreArchivo);

return 0;

}

Texto

Descripción generada automáticamente

2. Programación Orientada a Objetos en C++

#include <iostream>

#include <string>

class Persona {

private:

std::string nombre;

int edad;

std::string ocupacion;

public:

Persona(const std::string& nombre, int edad, const std::string& ocupacion)

: nombre(nombre), edad(edad), ocupacion(ocupacion) {}

void setNombre(const std::string& nombre) {

this->nombre = nombre;

}

std::string getNombre() const {

return nombre;

}

void setEdad(int edad) {

this->edad = edad;

}

int getEdad() const {

return edad;

}

void setOcupacion(const std::string& ocupacion) {

this->ocupacion = ocupacion;

}

std::string getOcupacion() const {

return ocupacion;

}

void mostrarInformacion() const {

std::cout << "Nombre: " << nombre << ", Edad: " << edad << ", Ocupación: " << ocupacion << std::endl;

}

};

int main() {

Persona persona("Juan", 30, "Ingeniero");

persona.mostrarInformacion();

return 0;

}

Captura de pantalla de computadora

Descripción generada automáticamente

**3. Abstracción e Instanciación**

#include <iostream>

#include <fstream>

#include <string>

// Definición de la clase Persona

class Persona {

private:

std::string nombre;

int edad;

public:

// Constructor

Persona(const std::string& nombre, int edad) : nombre(nombre), edad(edad) {}

// Métodos para establecer y obtener el nombre y la edad

void setNombre(const std::string& nombre) {

this->nombre = nombre;

}

std::string getNombre() const {

return nombre;

}

void setEdad(int edad) {

this->edad = edad;

}

int getEdad() const {

return edad;

}

// Método para mostrar información de la persona

void mostrarInformacion() const {

std::cout << "Nombre: " << nombre << ", Edad: " << edad << std::endl;

}

};

// Función para guardar datos de personas en un archivo

void guardarDatosEnArchivo(const std::string& nombreArchivo, const std::vector<Persona>& personas) {

std::ofstream archivo(nombreArchivo);

if (archivo.is\_open()) {

for (const auto& persona : personas) {

archivo << persona.getNombre() << " " << persona.getEdad() << std::endl;

}

archivo.close();

std::cout << "Datos guardados en el archivo correctamente.\n";

} else {

std::cerr << "No se pudo abrir el archivo.\n";

}

}

// Función para leer datos de personas desde un archivo

void leerDatosDeArchivo(const std::string& nombreArchivo) {

std::ifstream archivo(nombreArchivo);

if (archivo.is\_open()) {

std::string nombre;

int edad;

std::cout << "Contenido del archivo:\n";

while (archivo >> nombre >> edad) {

Persona persona(nombre, edad);

persona.mostrarInformacion();

}

archivo.close();

} else {

std::cerr << "No se pudo abrir el archivo.\n";

}

}

int main() {

// Crear instancias de personas

std::vector<Persona> personas;

personas.push\_back(Persona("Juan", 30));

personas.push\_back(Persona("Maria", 25));

personas.push\_back(Persona("Pedro", 40));

// Guardar datos en un archivo

guardarDatosEnArchivo("datos.txt", personas);

// Leer datos desde un archivo y mostrarlos

leerDatosDeArchivo("datos.txt");

return 0;

}

Captura de pantalla de computadora

Descripción generada automáticamente

4. Clases, Objetos y Métodos

#include <iostream>

#include <string>

class Persona {

private:

std::string nombre;

int edad;

std::string ocupacion;

public:

Persona(const std::string& nombre, int edad, const std::string& ocupacion)

: nombre(nombre), edad(edad), ocupacion(ocupacion) {}

void setNombre(const std::string& nombre) {

this->nombre = nombre;

}

std::string getNombre() const {

return nombre;

}

void setEdad(int edad) {

this->edad = edad;

}

int getEdad() const {

return edad;

}

void setOcupacion(const std::string& ocupacion) {

this->ocupacion = ocupacion;

}

std::string getOcupacion() const {

return ocupacion;

}

void mostrarInformacion() const {

std::cout << "Nombre: " << nombre << ", Edad: " << edad << ", Ocupación: " << ocupacion << std::endl;

}

};

class Estudiante : public Persona {

private:

int numeroEstudiante;

float promedioCalificaciones;

public:

Estudiante(const std::string& nombre, int edad, const std::string& ocupacion, int numeroEstudiante, float promedioCalificaciones)

: Persona(nombre, edad, ocupacion), numeroEstudiante(numeroEstudiante), promedioCalificaciones(promedioCalificaciones) {}

void setNumeroEstudiante(int numeroEstudiante) {

this->numeroEstudiante = numeroEstudiante;

}

int getNumeroEstudiante() const {

return numeroEstudiante;

}

void setPromedioCalificaciones(float promedioCalificaciones) {

this->promedioCalificaciones = promedioCalificaciones;

}

float getPromedioCalificaciones() const {

return promedioCalificaciones;

}

void mostrarInformacion() const {

Persona::mostrarInformacion();

std::cout << "Número de Estudiante: " << numeroEstudiante << ", Promedio de Calificaciones: " << promedioCalificaciones << std::endl;

}

};

int main() {

Estudiante estudiante("Maria", 20, "Estudiante", 12345, 9.5);

estudiante.mostrarInformacion();

return 0;

}

Captura de pantalla de computadora

Descripción generada automáticamente

**5. Bibliotecas Estándar**

#include <iostream>

#include <string>

int main() {

// Declarar una cadena de caracteres utilizando la clase string de la biblioteca estándar

std::string nombre;

// Pedir al usuario que ingrese su nombre

std::cout << "Ingrese su nombre: ";

std::cin >> nombre;

// Mostrar el nombre ingresado por el usuario

std::cout << "¡Hola, " << nombre << "!" << std::endl;

// Concatenar cadenas utilizando el operador '+'

std::string apellido = "Gómez";

std::string nombreCompleto = nombre + " " + apellido;

std::cout << "Tu nombre completo es: " << nombreCompleto << std::endl;

// Obtener la longitud de una cadena utilizando el método size()

std::cout << "La longitud de tu nombre completo es: " << nombreCompleto.size() << " caracteres" << std::endl;

// Acceder a caracteres individuales de una cadena utilizando el operador de corchetes '[]'

std::cout << "El primer carácter de tu nombre es: " << nombreCompleto[0] << std::endl;

// Comprobar si una cadena está vacía utilizando el método empty()

if (nombre.empty()) {

std::cout << "La cadena 'nombre' está vacía." << std::endl;

} else {

std::cout << "La cadena 'nombre' no está vacía." << std::endl;

}

// Bucle para iterar sobre cada carácter de una cadena

std::cout << "Cada carácter de tu nombre completo es: ";

for (char c : nombreCompleto) {

std::cout << c << " ";

}

std::cout << std::endl;

return 0;

}

Texto

Descripción generada automáticamente

**6. Algoritmo de Ordenamiento**

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

int main() {

std::vector<std::string> nombres = {"Juan", "Maria", "Pedro", "Ana", "Luis"};

std::cout << "Nombres antes de ordenar:\n";

for (const auto& nombre : nombres) {

std::cout << nombre << std::endl;

}

std::sort(nombres.begin(), nombres.end());

std::cout << "\nNombres después de ordenar:\n";

for (const auto& nombre : nombres) {

std::cout << nombre << std::endl;

}

return 0;

}

Captura de pantalla de computadora

Descripción generada automáticamente

**Conclusión:**

Este laboratorio nos brindó una excelente oportunidad para aplicar y consolidar nuestros conocimientos en programación en C++. A través de actividades prácticas, pudimos profundizar nuestra comprensión de conceptos fundamentales como la manipulación de archivos, la programación orientada a objetos y el uso de bibliotecas estándar. Además, la implementación de algoritmos básicos como el ordenamiento nos permitió mejorar nuestras habilidades de resolución de problemas y pensamiento algorítmico. En general, esta experiencia fue valiosa para nuestro desarrollo como programadores en C++.

Enlace de GitHub :