Algoritmos en C++

Uso del tipo de datos char

Ejercicio 1: Comparar dos caracteres

En este ejercicio, se pide al usuario que ingrese dos caracteres y se compara cuál es mayor alfabéticamente.

#include <iostream>

using namespace std;

int main() {

char caracter1, caracter2;

bool datoValido = false;

do {

cout << "Ingrese el primer caracter: ";

cin >> caracter1;

cout << "Ingrese el segundo caracter: ";

cin >> caracter2;

if(caracter1 < 'A' || (caracter1 > 'Z' && caracter1 < 'a') || caracter1 > 'z' || caracter2 < 'A' || (caracter2 > 'Z' && caracter2 < 'a') || caracter2 > 'z') {

cout << "Error: los caracteres ingresados son invalidos." << endl;

} else {

datoValido = true;

if(caracter1 < caracter2) {

cout << "El caracter " << caracter2 << " es mayor que el caracter " << caracter1 << "." << endl;

} else if(caracter1 > caracter2) {

cout << "El caracter " << caracter1 << " es mayor que el caracter " << caracter2 << "." << endl;

} else {

cout << "Los caracteres son iguales." << endl;

}

}

} while(!datoValido);

return 0;

}

En este programa, se utiliza un bucle do-while para solicitar la entrada del usuario y comprobar si es válida. Si se ingresa un caracter inválido, se muestra un mensaje de error y se solicita que se ingrese un caracter nuevamente. Si se ingresan caracteres válidos, se compara cuál es mayor alfabéticamente y se muestra el resultado en la pantalla.

Ejercicio 2: Convertir un caracter a su valor ASCII

En este ejercicio, se pide al usuario que ingrese un caracter y se muestra su valor en la tabla ASCII.

#include <iostream>

using namespace std;

int main() {

char caracter;

bool datoValido = false;

do {

cout << "Ingrese un caracter: ";

cin >> caracter;

if(caracter < 'A' || (caracter > 'Z' && caracter < 'a') || caracter > 'z') {

cout << "Error: el caracter ingresado es invalido." << endl;

} else {

datoValido = true;

cout << "El valor ASCII del caracter " << caracter << " es " << int(caracter) << "." << endl;

}

} while(!datoValido);

return 0;

}

En este programa, se utiliza un bucle do-while para solicitar la entrada del usuario y comprobar si es válida. Si se ingresa un caracter inválido, se muestra un mensaje de error y se solicita que se ingrese un caracter nuevamente. Si se ingresa un caracter válido, se muestra su valor en la tabla ASCII utilizando la función int() para convertir el caracter en su valor entero correspondiente.

Ejercicio 3: Convertir un caracter de minúscula a mayúscula

En este ejercicio, se pide al usuario que ingrese un caracter en minúscula y se convierte el caracter a mayúscula.

#include <iostream>

using namespace std;

int main() {

char caracter;

bool datoValido = false;

do {

cout << "Ingrese un caracter en minuscula: ";

cin >> caracter;

if(caracter < 'a' || caracter > 'z') {

cout << "Error: el caracter ingresado es invalido." << endl;

} else {

datoValido = true;

caracter = caracter - 32; // Convertir a mayuscula

cout << "El caracter en mayuscula es " << caracter << "." << endl;

}

} while(!datoValido);

return 0;

}

En este programa, se utiliza un bucle do-while para solicitar la entrada del usuario y comprobar si es válida. Si se ingresa un caracter inválido, se muestra un mensaje de error y se solicita que se ingrese un caracter nuevamente. Si se ingresa un caracter válido, se convierte el caracter a mayúscula restando 32 de su valor en la tabla ASCII. El resultado se muestra en la pantalla.

Uso del tipo de dato char y cadenas

Ejercicio 1: Invertir una cadena

Escribir un programa que lea una cadena de caracteres por teclado y la imprima invertida en la pantalla. Se debe utilizar un arreglo de tipo char para almacenar la cadena.

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char cadena[100];

bool datoValido = false;

do {

cout << "Ingrese una cadena: ";

cin.getline(cadena, 100);

int longitud = strlen(cadena);

if(longitud == 0) {

cout << "Error: la cadena no puede estar vacía." << endl;

} else {

datoValido = true;

cout << "La cadena invertida es: ";

for(int i = longitud-1; i >= 0; i--) {

cout << cadena[i];

}

}

} while(!datoValido);

return 0;

}

Ejercicio 2: Contar las vocales

Escribir un programa que lea una cadena de caracteres por teclado y cuente el número de vocales que contiene. Se debe utilizar un arreglo de tipo char para almacenar la cadena.

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char cadena[100];

bool datoValido = false;

do {

cout << "Ingrese una cadena: ";

cin.getline(cadena, 100);

int longitud = strlen(cadena);

if(longitud == 0) {

cout << "Error: la cadena no puede estar vacía." << endl;

} else {

datoValido = true;

int contador = 0;

for(int i = 0; i < longitud; i++) {

if(cadena[i] == 'a' || cadena[i] == 'e' || cadena[i] == 'i' || cadena[i] == 'o' || cadena[i] == 'u' ||

cadena[i] == 'A' || cadena[i] == 'E' || cadena[i] == 'I' || cadena[i] == 'O' || cadena[i] == 'U') {

contador++;

}

}

cout << "El numero de vocales es: " << contador;

}

} while(!datoValido);

return 0;

}

Ejercicio 3: Convertir una cadena a mayúsculas

Escribir un programa que lea una cadena de caracteres por teclado y la convierta a mayúsculas. Se debe utilizar un arreglo de tipo char para almacenar la cadena.

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char cadena[100];

bool datoValido = false;

do {

cout << "Ingrese una cadena: ";

cin.getline(cadena, 100);

int longitud = strlen(cadena);

if(longitud == 0) {

cout << "Error: la cadena no puede estar vacía." << endl;

} else {

datoValido = true;

for(int i = 0; i < longitud; i++) {

if(cadena[i] >= 'a' && cadena[i] <= 'z') {

cadena[i] = cadena[i] - 'a' + 'A';

}

}

cout << "La cadena en mayusculas es: " << cadena;

}

} while(!datoValido);

return 0;

}

Ejercicio 4: Contar la cantidad de caracteres en una cadena

En este ejercicio, se pide al usuario que ingrese una cadena de caracteres y se cuenta cuántos caracteres tiene la cadena.

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char cadena[100];

bool datoValido = false;

do {

cout << "Ingrese una cadena: ";

cin.getline(cadena, 100);

if(strlen(cadena) == 0) {

cout << "Error: la cadena no puede estar vacía." << endl;

} else {

datoValido = true;

int contador = 0;

while(cadena[contador] != '\0') {

contador++;

}

cout << "La cadena tiene " << contador << " caracteres." << endl;

}

} while(!datoValido);

return 0;

}

Uso del tipo de dato int

Ejercicio 1: Realizar un programa que pida al usuario ingresar dos números enteros y muestre por pantalla la suma de dichos números.

#include <iostream>

using namespace std;

int main() {

int num1, num2, suma;

bool datosValidos = false;

do {

cout << "Ingrese dos numeros enteros: ";

cin >> num1 >> num2;

if(cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Error: los datos ingresados son invalidos." << endl;

} else {

datosValidos = true;

suma = num1 + num2;

cout << "La suma de " << num1 << " y " << num2 << " es " << suma << "." << endl;

}

} while(!datosValidos);

return 0;

}

En este programa, se utiliza un bucle do-while para solicitar la entrada del usuario y comprobar si es válida. Si se ingresan datos inválidos, se muestra un mensaje de error y se solicita que se ingresen los datos nuevamente. Si los datos son válidos, se calcula la suma y se muestra el resultado en la pantalla.

Ejercicio 2: Realizar un programa que pida al usuario ingresar un número entero y muestre por pantalla si dicho número es par o impar.

#include <iostream>

using namespace std;

int main() {

int num;

bool datoValido = false;

do {

cout << "Ingrese un numero entero: ";

cin >> num;

if(cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Error: el dato ingresado es invalido." << endl;

} else {

datoValido = true;

if(num % 2 == 0) {

cout << num << " es un numero par." << endl;

} else {

cout << num << " es un numero impar." << endl;

}

}

} while(!datoValido);

return 0;

}

En este programa, se utiliza un bucle do-while para solicitar la entrada del usuario y comprobar si es válida. Si se ingresa un dato inválido, se muestra un mensaje de error y se solicita que se ingrese el dato nuevamente. Si el dato es válido, se verifica si es par o impar y se muestra el resultado en la pantalla.

Ejercicio 3: Calculadora de operaciones aritméticas

En este ejercicio, se le pedirá al usuario que ingrese dos números enteros y una operación aritmética (suma, resta, multiplicación o división). El programa realizará la operación seleccionada y mostrará el resultado.

#include <iostream>

int main() {

int num1, num2;

char operacion;

std::cout << "Ingrese el primer número: ";

std::cin >> num1;

std::cout << "Ingrese el segundo número: ";

std::cin >> num2;

std::cout << "Ingrese la operación a realizar (+, -, \*, /): ";

std::cin >> operacion;

int resultado;

switch (operacion) {

case '+':

resultado = num1 + num2;

break;

case '-':

resultado = num1 - num2;

break;

case '\*':

resultado = num1 \* num2;

break;

case '/':

resultado = num1 / num2;

break;

default:

std::cout << "Operación inválida. Inténtelo de nuevo.\n";

return 0;

}

std::cout << "El resultado es: " << resultado << "\n";

return 0;

}

Ejercicio 4: Factorial de un número

En este ejercicio, se le pedirá al usuario que ingrese un número entero positivo y el programa calculará su factorial. El factorial de un número n se define como el producto de todos los números enteros positivos desde 1 hasta n.

#include <iostream>

int main() {

int num;

do {

std::cout << "Ingrese un número entero positivo: ";

std::cin >> num;

} while (num < 0);

int factorial = 1;

for (int i = 1; i <= num; i++) {

factorial \*= i;

}

std::cout << "El factorial de " << num << " es " << factorial << "\n";

return 0;

}

Uso de excepciones con try { } catch { }

Ejercicio 1:

Enunciado: Escribe un programa en C++ que solicite al usuario un número entero positivo y muestre por pantalla si es par o impar.

#include <iostream>

#include <stdexcept>

using namespace std;

int main() {

int num;

while (true) {

try {

cout << "Introduce un numero entero positivo: ";

cin >> num;

if (num < 0) {

throw invalid\_argument("El numero debe ser positivo.");

}

break;

}

catch (const exception& e) {

cout << "Error: " << e.what() << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

if (num % 2 == 0) {

cout << "El numero es par." << endl;

}

else {

cout << "El numero es impar." << endl;

}

return 0;

}

Ejercicio 2:

Enunciado: Escribe un programa en C++ que solicite al usuario dos números enteros y muestre por pantalla su suma.

#include <iostream>

#include <stdexcept>

using namespace std;

int main() {

int num1, num2;

while (true) {

try {

cout << "Introduce un numero entero: ";

cin >> num1;

cout << "Introduce otro numero entero: ";

cin >> num2;

break;

}

catch (const exception& e) {

cout << "Error: " << e.what() << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

int suma = num1 + num2;

cout << "La suma de " << num1 << " y " << num2 << " es " << suma << "." << endl;

return 0;

}

Ejercicio 3:

Enunciado: Escribe un programa en C++ que solicite al usuario un número entero y muestre por pantalla su tabla de multiplicar del 1 al 10.

Código:

#include <iostream>

#include <stdexcept>

using namespace std;

int main() {

int num;

while (true) {

try {

cout << "Introduce un numero entero: ";

cin >> num;

break;

}

catch (const exception& e) {

cout << "Error: " << e.what() << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

cout << "Tabla de multiplicar de " << num << ":" << endl;

for (int i = 1; i <= 10; i++) {

cout << num << " x " << i << " = " << num \* i << endl;

}

return 0;

}

Ejercicio 4: Calculadora de operaciones aritméticas

#include <iostream>

#include <stdexcept>

int main() {

int num1, num2;

char operacion;

while (true) {

try {

std::cout << "Ingrese el primer número: ";

std::cin >> num1;

std::cout << "Ingrese el segundo número: ";

std::cin >> num2;

std::cout << "Ingrese la operación a realizar (+, -, \*, /): ";

std::cin >> operacion;

if (std::cin.fail() || (operacion != '+' && operacion != '-' && operacion != '\*' && operacion != '/')) {

throw std::runtime\_error("Error: entrada inválida. Inténtelo de nuevo.");

}

break;

} catch (std::runtime\_error &e) {

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

std::cerr << e.what() << std::endl;

}

}

int resultado;

switch (operacion) {

case '+':

resultado = num1 + num2;

break;

case '-':

resultado = num1 - num2;

break;

case '\*':

resultado = num1 \* num2;

break;

case '/':

resultado = num1 / num2;

break;

default:

std::cout << "Operación inválida. Inténtelo de nuevo.\n";

return 0;

}

std::cout << "El resultado es: " << resultado << "\n";

return 0;

}

Uso de vectores con números enteros

Ejercicio 1: Calcular la suma de los elementos de un vector de enteros

Enunciado:

Escribe un programa en C++ que solicite al usuario que ingrese un vector de enteros y calcule la suma de sus elementos. El programa debe imprimir la suma en pantalla.

#include <iostream>

using namespace std;

int main() {

const int TAM = 10;

int vector[TAM], suma = 0;

// Solicitamos al usuario que ingrese los elementos del vector

cout << "Ingrese los elementos del vector:" << endl;

for (int i = 0; i < TAM; i++) {

cout << "Elemento " << i+1 << ": ";

cin >> vector[i];

}

// Calculamos la suma de los elementos del vector

for (int i = 0; i < TAM; i++) {

suma += vector[i];

}

// Imprimimos la suma en pantalla

cout << "La suma de los elementos del vector es: " << suma << endl;

return 0;

}

Ejercicio 2: Suma de elementos de un vector

Enunciado:

Escriba un programa que lea un vector de números enteros desde el teclado y calcule la suma de todos sus elementos.

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> vec;

int n, sum = 0;

cout << "Ingrese el numero de elementos del vector: ";

cin >> n;

vec.resize(n);

for(int i = 0; i < n; i++) {

bool error = true;

do {

try {

cout << "Ingrese el elemento " << i + 1 << " del vector: ";

cin >> vec[i];

if(cin.fail()) {

throw runtime\_error("Error: el valor ingresado no es un entero.");

}

error = false;

} catch(runtime\_error& e) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << e.what() << endl;

}

} while(error);

}

for(int i = 0; i < n; i++) {

sum += vec[i];

}

cout << "La suma de los elementos del vector es: " << sum << endl;

return 0;

}

Ejercicio 3: Ordenar un vector de enteros de menor a mayor

Enunciado:

Escribe un programa en C++ que solicite al usuario que ingrese un vector de enteros y lo ordene de menor a mayor utilizando el algoritmo de ordenamiento por burbuja. El programa debe imprimir el vector ordenado en pantalla.

#include <iostream>

using namespace std;

int main() {

const int TAM = 10;

int vector[TAM], aux;

bool cambio;

// Solicitamos al usuario que ingrese los elementos del vector

cout << "Ingrese los elementos del vector:" << endl;

for (int i = 0; i < TAM; i++) {

cout << "Elemento " << i+1 << ": ";

cin >> vector[i];

}

// Ordenamos el vector utilizando el algoritmo de ordenamiento por burbuja

do {

cambio = false;

for (int i = 0; i < TAM-1; i++) {

if (vector[i] > vector[i+1]) {

aux = vector[i];

vector[i] = vector[i+1];

vector[i+1] = aux;

cambio = true;

}

}

} while (cambio);

// Imprimimos el vector ordenado en pantalla

cout << "Vector ordenado: ";

for (int i = 0; i < TAM; i++) {

cout << vector[i] << " ";

}

cout << endl;

return 0;

}

Ejercicio 4: Ordenamiento de un vector dinámico con control de excepciones

Enunciado:

Escriba un programa que lea un vector de números enteros desde el teclado y los ordene de manera ascendente utilizando el algoritmo de ordenamiento burbuja.

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> vec;

int n;

cout << "Ingrese el numero de elementos del vector: ";

cin >> n;

vec.resize(n);

for(int i = 0; i < n; i++) {

bool error = true;

do {

try {

cout << "Ingrese el elemento " << i + 1 << " del vector: ";

cin >> vec[i];

if(cin.fail()) {

throw runtime\_error("Error: el valor ingresado no es un entero.");

}

error = false;

} catch(runtime\_error& e) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << e.what() << endl;

}

} while(error);

}

for(int i = 0; i < n - 1; i++) {

for(int j = 0; j < n - i - 1; j++) {

if(vec[j] > vec[j + 1]) {

int temp = vec[j];

vec[j] = vec[j + 1];

vec[j + 1] = temp;

}

}

}

cout << "El vector ordenado de manera ascendente es: ";

for(int i = 0; i < n; i++) {

cout << vec[i] << " ";

}

cout << endl;

return 0;

}

Ejercicio 5: Suma de vectores

En este ejercicio se pide al usuario que ingrese dos vectores de tamaño N y luego se realiza la suma de los mismos, almacenando el resultado en un tercer vector.

Enunciado:

1. Solicitar al usuario el tamaño N de los vectores.
2. Crear dos vectores de tamaño N.
3. Solicitar al usuario que ingrese los elementos del primer vector.
4. Solicitar al usuario que ingrese los elementos del segundo vector.
5. Sumar los elementos correspondientes de ambos vectores y almacenarlos en un tercer vector.
6. Mostrar el resultado de la suma.

#include <iostream>

#include <vector>

using namespace std;

int main() {

int N;

cout << "Ingrese el tamaño N de los vectores: ";

cin >> N;

// Crear dos vectores de tamaño N

vector<int> v1(N);

vector<int> v2(N);

// Solicitar los elementos del primer vector

cout << "Ingrese los elementos del primer vector:" << endl;

for (int i = 0; i < N; i++) {

while (true) {

try {

cout << "Elemento " << i << ": ";

cin >> v1[i];

break;

} catch (exception e) {

cout << "Error: el valor ingresado no es un número entero. Inténtelo de nuevo." << endl;

cin.clear();

cin.ignore(10000, '\n');

}

}

}

// Solicitar los elementos del segundo vector

cout << "Ingrese los elementos del segundo vector:" << endl;

for (int i = 0; i < N; i++) {

while (true) {

try {

cout << "Elemento " << i << ": ";

cin >> v2[i];

break;

} catch (exception e) {

cout << "Error: el valor ingresado no es un número entero. Inténtelo de nuevo." << endl;

cin.clear();

cin.ignore(10000, '\n');

}

}

}

// Sumar los elementos correspondientes de ambos vectores

vector<int> suma(N);

for (int i = 0; i < N; i++) {

suma[i] = v1[i] + v2[i];

}

// Mostrar el resultado de la suma

cout << "La suma de los dos vectores es: ";

for (int i = 0; i < N; i++) {

cout << suma[i] << " ";

}

cout << endl;

return 0;

}

Este programa también utiliza try y catch para manejar errores en la entrada de datos, asegurando que solo se ingresen números enteros.

Uso del tipo de dato float

Ejercicio 1

Realizar un programa que calcule el promedio de 3 números ingresados por teclado y muestre el resultado por pantalla.

#include <iostream>

using namespace std;

int main() {

float num1, num2, num3, promedio;

bool error;

do {

error = false;

cout << "Ingrese el primer numero: ";

if (!(cin >> num1)) {

cout << "Error: debe ingresar un numero." << endl;

cin.clear();

cin.ignore(10000, '\n');

error = true;

}

} while (error);

do {

error = false;

cout << "Ingrese el segundo numero: ";

if (!(cin >> num2)) {

cout << "Error: debe ingresar un numero." << endl;

cin.clear();

cin.ignore(10000, '\n');

error = true;

}

} while (error);

do {

error = false;

cout << "Ingrese el tercer numero: ";

if (!(cin >> num3)) {

cout << "Error: debe ingresar un numero." << endl;

cin.clear();

cin.ignore(10000, '\n');

error = true;

}

} while (error);

promedio = (num1 + num2 + num3) / 3;

cout << "El promedio es: " << promedio << endl;

return 0;

}

Ejercicio 2

Realizar un programa que convierta grados Celsius a Fahrenheit. El programa debe solicitar el valor en grados Celsius y mostrar el resultado en grados Fahrenheit por pantalla.

#include <iostream>

using namespace std;

int main() {

float celsius, fahrenheit;

bool error;

do {

error = false;

cout << "Ingrese los grados Celsius: ";

if (!(cin >> celsius)) {

cout << "Error: debe ingresar un numero." << endl;

cin.clear();

cin.ignore(10000, '\n');

error = true;

}

} while (error);

fahrenheit = (celsius \* 1.8) + 32;

cout << celsius << " grados Celsius son " << fahrenheit << " grados Fahrenheit." << endl;

return 0;

}

Ejercicio 3:

Escribir un programa que pida al usuario dos números flotantes y muestre por pantalla su división. Si el divisor es cero, el programa debe mostrar un mensaje de error.

#include <iostream>

using namespace std;

int main()

{

float dividendo, divisor, resultado;

cout << "Ingrese el dividendo: ";

cin >> dividendo;

bool divisor\_valido = false;

while (!divisor\_valido)

{

cout << "Ingrese el divisor: ";

cin >> divisor;

if (divisor == 0)

{

cout << "Error: el divisor no puede ser cero. Intente nuevamente." << endl;

}

else

{

divisor\_valido = true;

}

}

resultado = dividendo / divisor;

cout << "El resultado de la division es: " << resultado << endl;

return 0;

}

Explicación del código:

* En este programa, primero se declaran las variables necesarias: dividendo, divisor y resultado, todas del tipo float.
* Luego se pide al usuario que ingrese el dividendo, utilizando el operador de entrada >> con la función cin.
* A continuación, se inicia un bucle while que se ejecuta mientras el valor del divisor no sea válido. La variable divisor\_valido se inicializa en false, lo que hace que el bucle se ejecute al menos una vez.
* Dentro del bucle, se pide al usuario que ingrese el divisor utilizando cin.
* Luego se verifica si el divisor es cero. Si es así, se muestra un mensaje de error y se vuelve a pedir el divisor. Si no es cero, se establece divisor\_valido en true y se sale del bucle.
* Después de salir del bucle, se realiza la operación de división y se almacena el resultado en la variable resultado.
* Por último, se muestra el resultado en la pantalla utilizando el operador de salida << con la función cout.

Este programa incluye el control de errores para el caso en que el divisor sea cero. Si el usuario ingresa un divisor inválido, el programa le pedirá que ingrese un divisor válido hasta que lo haga.

Ejercicio 4

Enunciado: Escribir un programa en C++ que calcule la hipotenusa de un triángulo rectángulo dados sus dos catetos.

#include <iostream>

#include <cmath>

using namespace std;

int main() {

float cateto1, cateto2, hipotenusa;

bool entrada\_correcta = false;

while (!entrada\_correcta) {

try {

cout << "Ingrese el primer cateto: ";

cin >> cateto1;

if (cin.fail()) {

throw runtime\_error("El valor ingresado no es un número válido.");

}

cout << "Ingrese el segundo cateto: ";

cin >> cateto2;

if (cin.fail()) {

throw runtime\_error("El valor ingresado no es un número válido.");

}

entrada\_correcta = true;

}

catch (const exception& e) {

cout << "Error: " << e.what() << " Por favor, intente de nuevo." << endl;

cin.clear();

cin.ignore(10000, '\n');

}

}

hipotenusa = sqrt(pow(cateto1, 2) + pow(cateto2, 2));

cout << "La hipotenusa es: " << hipotenusa << endl;

return 0;

}

Este programa utiliza el teorema de Pitágoras para calcular la hipotenusa de un triángulo rectángulo dados sus dos catetos. Primero, se declaran tres variables de tipo float: cateto1, cateto2 y hipotenusa. Luego, se declara una variable booleana entrada\_correcta que se utilizará para controlar el bucle while que se encarga de solicitar los datos de entrada y controlar que sean válidos. Dentro del bucle, se utiliza un bloque try-catch para detectar si se ha introducido un valor no numérico y mostrar un mensaje de error. Si se produce un error, se llama a las funciones cin.clear() y cin.ignore() para borrar la entrada incorrecta del buffer y evitar un bucle infinito.

Una vez que se han introducido los valores correctos para cateto1 y cateto2, se utiliza la función pow() de la biblioteca cmath para elevar al cuadrado ambos catetos, se suman y se utiliza la función sqrt() para calcular la raíz cuadrada de la suma, que es la hipotenusa. Finalmente, se muestra el resultado en pantalla utilizando la función cout.

Uso del tipo de dato float con vectores

Ejercicio 1: Calculo del promedio de 5 números

#include <iostream>

using namespace std;

int main() {

const int TAM = 5;

float numeros[TAM];

float suma = 0, promedio = 0;

// Ingreso de datos

cout << "Ingrese " << TAM << " numeros:" << endl;

for (int i = 0; i < TAM; i++) {

bool input\_valido = false;

do {

cout << "Numero " << i+1 << ": ";

if (cin >> numeros[i]) {

input\_valido = true;

} else {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Entrada invalida. Intente de nuevo." << endl;

}

} while (!input\_valido);

suma += numeros[i];

}

// Calculo del promedio

promedio = suma / TAM;

// Impresion de los numeros ingresados y el promedio

cout << "\nNumeros ingresados: ";

for (int i = 0; i < TAM; i++) {

cout << numeros[i] << " ";

}

cout << "\nPromedio: " << promedio << endl;

return 0;

}

Este programa pide al usuario que ingrese 5 números en un vector y calcula el promedio de los mismos. Además, incluye el control de entrada de datos inválidos con un ciclo do-while y el uso de la función cin.clear() para limpiar el buffer de entrada de datos y evitar un loop infinito en caso de que el usuario ingrese una entrada inválida.

Ejercicio 2: Ordenamiento de números en un vector

En este ejercicio, se le pedirá al usuario que ingrese la cantidad de números que desea ingresar, luego se le pedirá que ingrese los números y finalmente se ordenarán y mostrarán en orden ascendente.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

int n;

float num;

vector<float> numeros;

cout << "Ingrese la cantidad de numeros: ";

cin >> n;

// Pedimos los numeros y los almacenamos en el vector

for (int i = 0; i < n; i++) {

while (true) {

cout << "Ingrese el numero " << i + 1 << ": ";

if (cin >> num) {

numeros.push\_back(num);

break;

} else {

cout << "Dato incorrecto. Intente nuevamente." << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

}

// Ordenamos los numeros en orden ascendente

sort(numeros.begin(), numeros.end());

// Mostramos los numeros ordenados

cout << "Los numeros en orden ascendente son: ";

for (int i = 0; i < n; i++) {

cout << numeros[i] << " ";

}

cout << endl;

return 0;

}

Ejercicio 3: Promedio de números en un vector

En este ejercicio, se le pedirá al usuario que ingrese la cantidad de números que desea ingresar, luego se le pedirá que ingrese los números y finalmente se calculará y mostrará el promedio.

#include <iostream>

#include <vector>

using namespace std;

int main() {

int n;

float num, suma = 0;

vector<float> numeros;

cout << "Ingrese la cantidad de numeros: ";

cin >> n;

// Pedimos los numeros y los almacenamos en el vector

for (int i = 0; i < n; i++) {

while (true) {

cout << "Ingrese el numero " << i + 1 << ": ";

if (cin >> num) {

numeros.push\_back(num);

break;

} else {

cout << "Dato incorrecto. Intente nuevamente." << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

}

}

// Calculamos el promedio

for (int i = 0; i < n; i++) {

suma += numeros[i];

}

float promedio = suma / n;

// Mostramos el resultado

cout << "El promedio es: " << promedio << endl;

return 0;

}

Uso de matrices

Ejercicio 1: Suma de matrices

En este ejercicio, se pide al usuario ingresar dos matrices y luego se suman sus elementos. El programa debe comprobar que las matrices tengan las mismas dimensiones antes de realizar la suma.

cpp

Copy code

#include <iostream>

using namespace std;

int main() {

const int ROWS = 2, COLS = 3;

float matrix1[ROWS][COLS], matrix2[ROWS][COLS], result[ROWS][COLS];

// Ingresar elementos de la primera matriz

cout << "Ingrese los elementos de la primera matriz (" << ROWS << "x" << COLS << "):\n";

for (int i = 0; i < ROWS; i++) {

for (int j = 0; j < COLS; j++) {

cout << "Ingrese el elemento [" << i << "][" << j << "]: ";

cin >> matrix1[i][j];

}

}

// Ingresar elementos de la segunda matriz

cout << "\nIngrese los elementos de la segunda matriz (" << ROWS << "x" << COLS << "):\n";

for (int i = 0; i < ROWS; i++) {

for (int j = 0; j < COLS; j++) {

cout << "Ingrese el elemento [" << i << "][" << j << "]: ";

cin >> matrix2[i][j];

}

}

// Sumar las matrices

for (int i = 0; i < ROWS; i++) {

for (int j = 0; j < COLS; j++) {

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

// Imprimir el resultado

cout << "\nLa suma de las matrices es:\n";

for (int i = 0; i < ROWS; i++) {

for (int j = 0; j < COLS; j++) {

cout << result[i][j] << " ";

}

cout << endl;

}

return 0;

}

Uso del tipo de datos enum

Ejercicio 1: Días de la semana

En este ejercicio, se utilizará un tipo de dato enum para representar los días de la semana. Se pedirá al usuario que ingrese un número entero entre 1 y 7, y el programa imprimirá el día de la semana correspondiente.

#include <iostream>

enum DiasSemana {

LUNES = 1,

MARTES,

MIERCOLES,

JUEVES,

VIERNES,

SABADO,

DOMINGO

};

int main() {

int dia;

std::cout << "Ingrese un número entre 1 y 7 para indicar el día de la semana: ";

std::cin >> dia;

while (dia < LUNES || dia > DOMINGO) {

std::cout << "Número ingresado inválido. Ingrese un número entre 1 y 7: ";

std::cin >> dia;

}

switch (dia) {

case LUNES:

std::cout << "Lunes" << std::endl;

break;

case MARTES:

std::cout << "Martes" << std::endl;

break;

case MIERCOLES:

std::cout << "Miércoles" << std::endl;

break;

case JUEVES:

std::cout << "Jueves" << std::endl;

break;

case VIERNES:

std::cout << "Viernes" << std::endl;

break;

case SABADO:

std::cout << "Sábado" << std::endl;

break;

case DOMINGO:

std::cout << "Domingo" << std::endl;

break;

default:

std::cout << "Número ingresado inválido." << std::endl;

break;

}

return 0;

}

Uso de constantes

Ejercicio 1: Área de un círculo con constante PI

En este ejercicio, se pide al usuario que ingrese el radio de un círculo y luego se calcula y se muestra por pantalla el área del mismo utilizando una constante PI predefinida.

#include <iostream>

using namespace std;

int main() {

const float PI = 3.14159;

float radio, area;

cout << "Ingrese el radio del circulo: ";

cin >> radio;

// Calculamos el area del circulo utilizando la formula A = pi \* r^2

area = PI \* radio \* radio;

cout << "El area del circulo es: " << area << endl;

return 0;

}

Ejercicio 2: Valor máximo con constante MAX

En este ejercicio, se pide al usuario que ingrese dos números y se muestra por pantalla cuál es el número máximo utilizando una constante MAX predefinida.

#include <iostream>

using namespace std;

int main() {

const int MAX = 100;

int num1, num2;

cout << "Ingrese el primer numero: ";

cin >> num1;

cout << "Ingrese el segundo numero: ";

cin >> num2;

// Comparamos ambos numeros y mostramos por pantalla el mayor de ellos

if (num1 > num2) {

cout << "El numero mayor es: " << num1 << endl;

} else {

cout << "El numero mayor es: " << num2 << endl;

}

return 0;

}

Ejercicio 3: Edad mínima con constante EDAD\_MINIMA

En este ejercicio, se pide al usuario que ingrese su edad y se verifica si cumple con la edad mínima requerida para realizar una determinada actividad utilizando una constante EDAD\_MINIMA predefinida.

#include <iostream>

using namespace std;

int main() {

const int EDAD\_MINIMA = 18;

int edad;

cout << "Ingrese su edad: ";

cin >> edad;

// Verificamos si la edad ingresada cumple con la edad minima requerida

if (edad >= EDAD\_MINIMA) {

cout << "Usted cumple con la edad minima requerida." << endl;

} else {

cout << "Usted no cumple con la edad minima requerida." << endl;

}

return 0;

}

En estos ejemplos, la constante es definida con la palabra clave "const" seguida del tipo de dato y el nombre de la constante. Luego, se puede utilizar esta constante en el código de la misma manera que se utilizaría cualquier otra variable.

El uso de #define en C++

#define se utiliza para definir constantes simbólicas en el código. Estas constantes se reemplazan por su valor correspondiente en tiempo de compilación.

Ejercicio 1: Conversión de grados Celsius a Fahrenheit

En este ejercicio se solicitará al usuario ingresar una temperatura en grados Celsius y se calculará la equivalente en grados Fahrenheit utilizando la fórmula: F = (C \* 1.8) + 32. Para ello, se definirá una constante simbólica con #define para el factor de conversión 1.8 y se solicitará la entrada de datos por teclado.

#include <iostream>

#define CONVERSION\_FACTOR 1.8

using namespace std;

int main() {

float celsius, fahrenheit;

// Pedir la entrada de temperatura en grados Celsius

cout << "Ingrese la temperatura en grados Celsius: ";

cin >> celsius;

// Convertir a grados Fahrenheit utilizando la constante simbólica

fahrenheit = (celsius \* CONVERSION\_FACTOR) + 32;

// Mostrar la temperatura equivalente en grados Fahrenheit

cout << celsius << " grados Celsius equivalen a " << fahrenheit << " grados Fahrenheit." << endl;

return 0;

}

Ejercicio 2: Cálculo del área de un círculo

En este ejercicio se solicitará al usuario ingresar el radio de un círculo y se calculará su área utilizando la fórmula: A = pi \* r^2. Para ello, se definirá una constante simbólica con #define para el valor de pi y se solicitará la entrada de datos por teclado.

#include <iostream>

#define PI 3.1416

using namespace std;

int main() {

float radio, area;

// Pedir la entrada del radio del círculo

cout << "Ingrese el radio del circulo: ";

cin >> radio;

// Calcular el área del círculo utilizando la constante simbólica

area = PI \* radio \* radio;

// Mostrar el área del círculo

cout << "El area del circulo es: " << area << endl;

return 0;

}

Ejercicio 3: Conversión de dólares a pesos mexicanos

En este ejercicio se solicitará al usuario ingresar una cantidad en dólares y se calculará su equivalente en pesos mexicanos utilizando un tipo de cambio fijo. Para ello, se definirá una constante simbólica con #define para el tipo de cambio y se solicitará la entrada de datos por teclado.

#include <iostream>

#define TIPO\_CAMBIO 21.10

using namespace std;

int main() {

float dolares, pesos;

// Pedir la entrada de la cantidad en dólares

cout << "Ingrese la cantidad en dolares: ";

cin >> dolares;

// Convertir a pesos mexicanos utilizando la constante simbólica

pesos = dolares \* TIPO\_CAMBIO;

// Mostrar la cantidad equivalente en pesos mexicanos

cout << dolares << " dolares equivalen a " << pesos << " pesos mexicanos." << endl;

return 0;

}

Uso del tipo de dato struct

Enunciado:

Crear un programa en C++ que permita almacenar información de varios estudiantes utilizando una estructura llamada "Estudiante" que contenga los siguientes campos: nombre, edad, promedio y género. El programa debe permitir al usuario ingresar los datos de varios estudiantes, almacenarlos en un array de estructuras y luego mostrarlos en pantalla.

Código fuente:

#include <iostream>

#include <string>

using namespace std;

const int MAX\_ESTUDIANTES = 10; // Definimos la cantidad máxima de estudiantes a almacenar

// Definimos la estructura Estudiante

struct Estudiante {

string nombre;

int edad;

float promedio;

char genero;

};

// Función para ingresar los datos de un estudiante

void ingresarEstudiante(Estudiante &estudiante) {

cout << "Ingrese el nombre del estudiante: ";

getline(cin, estudiante.nombre);

cout << "Ingrese la edad del estudiante: ";

cin >> estudiante.edad;

cout << "Ingrese el promedio del estudiante: ";

cin >> estudiante.promedio;

cout << "Ingrese el género del estudiante (M/F): ";

cin >> estudiante.genero;

cin.ignore(); // Ignoramos el salto de línea que queda en el buffer después de ingresar el género

}

// Función para mostrar los datos de un estudiante

void mostrarEstudiante(const Estudiante &estudiante) {

cout << "Nombre: " << estudiante.nombre << endl;

cout << "Edad: " << estudiante.edad << endl;

cout << "Promedio: " << estudiante.promedio << endl;

cout << "Género: " << estudiante.genero << endl;

}

int main() {

Estudiante estudiantes[MAX\_ESTUDIANTES]; // Declaramos el array de estructuras

int cantidadEstudiantes;

// Pedimos al usuario la cantidad de estudiantes a ingresar

cout << "Ingrese la cantidad de estudiantes a ingresar (maximo " << MAX\_ESTUDIANTES << "): ";

cin >> cantidadEstudiantes;

cin.ignore(); // Ignoramos el salto de línea que queda en el buffer después de ingresar la cantidad

// Ingresamos los datos de los estudiantes

for (int i = 0; i < cantidadEstudiantes; i++) {

cout << "Estudiante #" << i+1 << ":" << endl;

ingresarEstudiante(estudiantes[i]);

cout << endl;

}

// Mostramos los datos de los estudiantes

cout << "Datos de los estudiantes:" << endl;

for (int i = 0; i < cantidadEstudiantes; i++) {

cout << "Estudiante #" << i+1 << ":" << endl;

mostrarEstudiante(estudiantes[i]);

cout << endl;

}

return 0;

}

Este programa utiliza la estructura "Estudiante" para almacenar los datos de varios estudiantes, permitiendo al usuario ingresar los datos de cada estudiante y luego mostrarlos en pantalla. La función "ingresarEstudiante" se encarga de pedir al usuario los datos de un estudiante y almacenarlos en una estructura, mientras que la función "mostrarEstudiante" muestra en pantalla los datos de un estudiante. En el "main" se declara un array de estructuras "estudiantes" y se utiliza un ciclo "for" para ingresar y mostrar los datos de cada estudiante.

Uso de funciones

Ejercicio 1

Enunciado: Crear un programa en C++ que tenga un menú de tres opciones para calcular el área de un cuadrado, de un triangulo rectángulo y de un círculo. El usuario debe elegir una opción del menú y el programa debe pedir los datos de la opción seleccionada y realizar los cálculos. El programa debe usar funciones.

#include <iostream>

#include <cmath>

using namespace std;

// Prototipos de funciones

void menu();

float calcularAreaCuadrado();

float calcularAreaTriangulo();

float calcularAreaCirculo();

int main() {

int opcion;

do {

menu(); // Mostrar el menú

cin >> opción; // Leer la opción seleccionada por el usuario

cout << endl;

switch(opcion) {

case 1: // Calcular el área de un cuadrado

cout << "El área del cuadrado es: " << calcularAreaCuadrado() << endl;

break;

case 2: // Calcular el área de un triángulo rectángulo

cout << "El area del triangulo rectángulo es: " << calcularAreaTriangulo() << endl;

break;

case 3: // Calcular el área de un círculo

cout << "El área del circulo es: " << calcularAreaCirculo() << endl;

break;

case 4: // Salir del programa

cout << "Saliendo del programa..." << endl;

break;

default: // Opción inválida

cout << "Opción invalida. Intente nuevamente." << endl;

break;

}

cout << endl;

} while(opcion != 4);

return 0;

}

// Función que muestra el menú de opciones

void menu() {

cout << "MENÚ DE OPCIONES" << endl;

cout << "----------------" << endl;

cout << "1. Calcular el área de un cuadrado" << endl;

cout << "2. Calcular el área de un triangulo rectángulo" << endl;

cout << "3. Calcular el área de un circulo" << endl;

cout << "4. Salir del programa" << endl;

cout << "Seleccione una opción: ";

}

// Función que calcula el área de un cuadrado

float calcularAreaCuadrado() {

float lado;

cout << "Ingrese el lado del cuadrado: ";

cin >> lado;

return pow(lado, 2);

}

// Función que calcula el área de un triángulo rectángulo

float calcularAreaTriangulo() {

float base, altura;

cout << "Ingrese la base del triangulo rectángulo: ";

cin >> base;

cout << "Ingrese la altura del triangulo rectángulo: ";

cin >> altura;

return (base \* altura) / 2;

}

// Función que calcula el área de un círculo

float calcularAreaCirculo() {

float radio;

cout << "Ingrese el radio del circulo: ";

cin >> radio;

return M\_PI \* pow(radio, 2);

}

Este programa utiliza un bucle do-while para mantenerse en ejecución hasta que el usuario decida salir del programa seleccionando la opción "4" del menú. En cada iteración del bucle, se muestra el menú de opciones y se lee la opción seleccionada por el usuario con cin. Luego, se usa un switch para ejecutar el código correspondiente a la opción seleccionada.

Cada una de las tres opciones del menú se implementan en tres funciones separadas.