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.