Documentación parcial pensamiento algorítmico

Tema: Estaciones

Estudiante: David Santiago Trujillo Murillo

1.1 Descripción del problema

La Agencia Espacial Internacional está diseñando nuevas estaciones espaciales con forma triangular para optimizar la captación de energía solar. Los ingenieros necesitan un programa que analice las dimensiones de estas estructuras para garantizar su estabilidad y clasificarlas según su geometría. El sistema de análisis "TriStation" debe examinar tres medidas que representan las longitudes de los lados de la estación espacial en unidades espaciales (UE) y determinar su configuración:

-Si los tres lados son iguales, se clasifica como "Equilátero" - ideal para una distribución uniforme de paneles solares.

-Si dos lados son iguales, se clasifica como "Isósceles" - útil para estaciones con un área de observación privilegiada.

-Si todos los lados son diferentes, se clasifica como "Escaleno" - diseño especial para múltiples funciones.

-Si la suma de dos lados cualesquiera no es mayor al tercer lado, se clasifica como "No es un triángulo" - diseño estructuralmente inestable.

Propósito del proyecto:

Este proyecto consiste en el desarrollo de un sistema de análisis de estaciones espaciales triangulares para la Agencia Espacial Internacional. Su propósito es determinar la estabilidad y la clasificación geométrica de estas estructuras según las longitudes de sus lados.

El sistema está implementado en Python y C++ utilizando clases y funciones lambda para optimizar la validación y clasificación.

Identificación de requisitos:

Requisitos funcionales:

1.El sistema debe recibir tres valores enteros que representan las longitudes de los lados.

2.El programa debe verificar si los valores corresponden a un triángulo válido.

3.Clasificar la estructura según su tipo: Equilátero, Isósceles, Escaleno o "No es un triángulo".

4.Implementar la solución en dos lenguajes: Python y C++.

5.Utilizar clases, polimorfismo y funciones lambda.

Requisitos no funcionales**:**

1.El código debe ser eficiente y cumplir con los estándares de programación.

2.Debe ser fácilmente entendible y estar debidamente documentado.

Casos de Uso

Entradas:

1.Tres números enteros que representan las longitudes de los lados de la estructura.

Procesos:

1.Validar si los valores ingresados forman un triángulo válido.

2.Determinar el tipo de triángulo según la igualdad de sus lados.

3.Retornar el resultado de la clasificación.

Salidas:

"Equilátero": Los tres lados son iguales.

"Isósceles": Dos lados son iguales.

"Escaleno": Todos los lados son diferentes.

"No es un triángulo": No cumple con la desigualdad triangular.

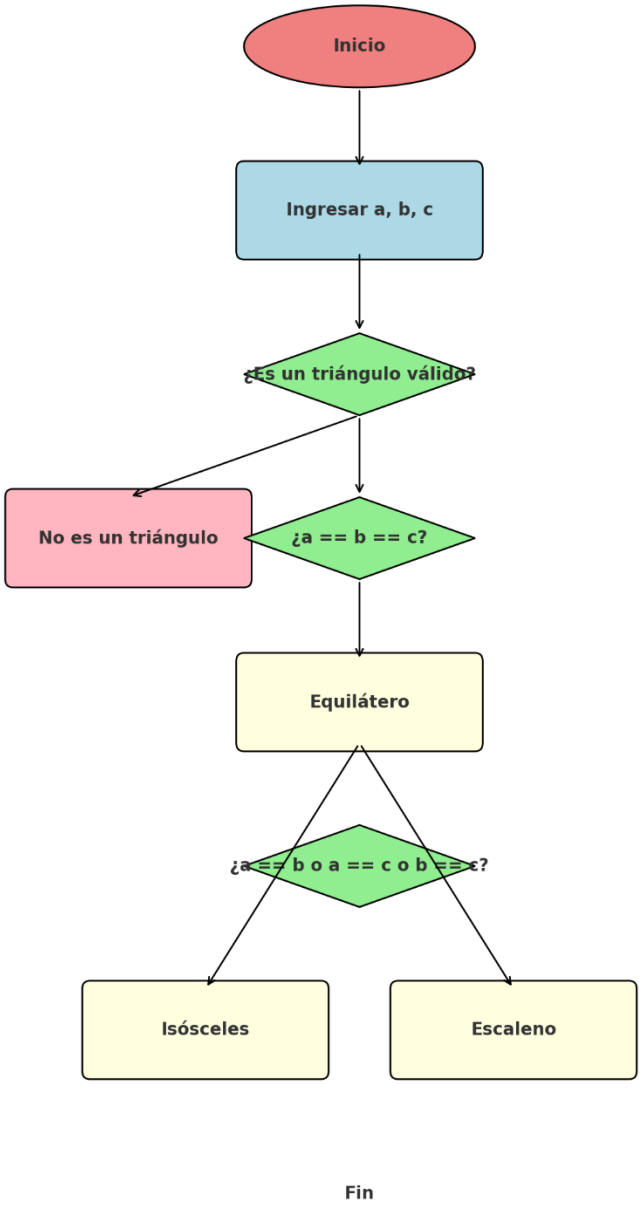
1.2 justificación de la solución

Para resolver el problema, se creó una clase Triangulo que contiene la lógica para verificar y clasificar los triángulos. Se utilizó una función lambda que extraje de los archivos de la biblioteca de Python y esta sirve para simplificar la comprobación de si los valores ingresados forman un triángulo válido. Esto hace que el código sea más compacto y fácil de entender.

Comparación con Otras Alternativas:

Se podría haber utilizado una serie de condicionales fuera de una clase, pero esto haría que el código fuera menos organizado y más difícil de mantener. Usar una clase permite estructurar mejor la solución y reutilizar el código con mayor facilidad. Además, la función lambda simplifica la validación del triángulo sin necesidad de escribir una función separada.

1.3 Diagrama UML



2. Documentación técnica

El proyecto está organizado en dos carpetas principales una la cual contiene el proyecto en Python y la otra en c++

Clases y Métodos

Clase Triangulo

La clase Triangulo es el núcleo del proyecto. Representa un triángulo y contiene la lógica para clasificarlo según las longitudes de sus lados.

Atributos de la Clase

a: Longitud del primer lado del triángulo.

b: Longitud del segundo lado del triángulo.

c: Longitud del tercer lado del triángulo.

Métodos de la Clase

1.Método Constructor:

Python: __init__(self, a, b, c)

C++: Triangulo(int a, int b, int c)

Descripción: Inicializa los atributos de la clase con las longitudes de los lados del triángulo. Se llama automáticamente cuando se crea una nueva instancia de la clase.

Método clasificar:

Python: def clasificar(self):

C++: string clasificar()

Descripción: Clasifica el triángulo en uno de los siguientes tipos:

"Equilátero": Si todos los lados son iguales.

"Isósceles": Si al menos dos lados son iguales.

"Escaleno": Si todos los lados son diferentes.

"No es un triángulo": Si las longitudes no cumplen con la desigualdad triangular.

Función Lambda (solo en Python):

Descripción: En la implementación de Python, se utiliza una función lambda para verificar si las longitudes pueden formar un triángulo. La función se define dentro del método clasificar y simplifica la lógica de verificación.

Organización del Código

1. Implementación en Python

Archivo estaciones.py:

-Contiene la clase Triangulo con su constructor y el método clasificar.

-La función lambda es_triangulo se utiliza para verificar la condición del triángulo.

Proceso para que realizarlo:

-Solicita al usuario que ingrese las longitudes de los lados del triángulo.

Crea una instancia de la clase Triangulo y llama al método clasificar.

Imprime la clasificación del triángulo.

Archivo README.md:

-Proporciona instrucciones para ejecutar el programa en Python.

-Incluye ejemplos de uso y detalles sobre la implementación.

2. Implementación en C++

Archivo estaciones.cpp:

-Contiene la clase Triangulo con su constructor y el método clasificar.

-Utiliza una función lambda para verificar la condición del triángulo.

Proceso para realizarlo:

-Solicita al usuario que ingrese las longitudes de los lados del triángulo.

-Crea una instancia de la clase Triangulo y llama al método clasificar.

-Imprime la clasificación del triángulo.

3. Instalar Python en Windows

● Instala Python desde python.org y marca la opción "Add Python to PATH".

● Instala Visual Studio Code desde code.visualstudio.com.

● Abre VS Code y ve a la pestaña Extensiones, busca "Python" e instálalo.

● Crea un archivo en VS Code con extensión .py

● Abre la terminal en Terminal > Nueva Terminal. ● Ejecuta el código escribiendo python conductividad.py y presiona Enter.

● Ingresa un número cuando lo pida el programa y obtendrás el resultado. 2. Instalar Python en Mac/Linux

● Verifica si Python está instalado escribiendo python3 --version en la terminal.

● Si no está, en macOS instala con brew install python, en Linux con sudo apt install python3 o sudo dnf install python3.

● Descarga e instala Visual Studio Code desde code.visualstudio.com.

● Mismos pasos de Windows después de instalar VS

Compilar y ejecutar en Windows c++

- Requisitos: - Compilador: Instala MinGW (Minimalist GNU for Windows) y Microsoft Visual Studio.

- Instala la extensión C++ en Visual Studio.

- Editor de texto: Puedes usar Notepad, Notepad++, o cualquier otro editor de tu preferencia.

- Pasos:

- Instalar MinGW:

- Descarga e instala MinGW desde su sitio oficial.

- Corre el instalador y sigue los pasos del mago de instalación.

- En el mago de instalación, escoge la carpeta deseada de instalación, guarda este directorio para más tarde.

- Cuando la instalación termine asegúrate de que marques “Ejecuta MSYS2” y da click en terminar. Una ventana terminal de MSYS2 se abrirá automáticamente.

- En la ventana terminal de MSYS2, instala la cadena de herramientas de “MinGW-w64” e ingresa el siguiente comando: pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain

- Te mostrarán la lista de paquetes disponibles

- Acepta la cantidad de paquetes por defecto en la cadena de herramientas presionando “Enter”.

- Ingresa “Y” cuando te lo pidan para continuar con la instalación.

- Agrega el camino de tu carpeta “bin” MinGW-w64 a los caminos de entorno de Windows usando las siguientes instrucciones:

- En la barra de búsqueda de Windows, escribe “Configuración” y abre la configuración de Windows.

- Busca “Editar variables de entorno” para tu cuenta.

- En las variables de tu usuario, selecciona variables de usuario y luego selecciona editar.

- Selecciona “Nuevo” y añade la carpeta de destino de “MinGW-w64” que guardaste durante el proceso de instalación.

- Selecciona “Aceptar” y luego selecciona “Aceptar” nuevamente en la ventana de variables del usuario para actualizar la variable de entorno de camino (Path), tienes que abrir una consola para que la variable de entorno de camino esté disponible para el uso.

- Crea una carpeta de proyectos desde la terminal para guardar tus proyectos de Visual Studio, luego crea una subcarpeta con el nombre de tu espacio, navega a ella y abre Visual Studio con el siguiente comando:

- mkdir proyectos 10

- cd proyectos

- mkdir espacio

- cd espacio - code .

- El comando code . abre Visual Studio en la carpeta de trabajo actual, lo que lo vuelve tu espacio de trabajo

- Siguiendo el tutorial verás que se crean 3 carpetas más en el folder .vscode de tu espacio de trabajo [tasks.json / launch.json / c_cpp_properties.json].

- Abre el código:

- Abre el código con extensión .cpp

- Compilar y correr el codigo en Visual Studio:

- En la parte superior derecha vas a encontrar un botón de play con un botón secundario en forma de flecha hacia abajo

- Selecciona el botón secundario y selecciona la opción “Ejecutar un archivo C++”

- En la configuración selecciona “C++: g++.exe construye y debug archivo activo”

- Solo tendrás que seleccionar el compilador la primera vez que corras un archivo .cpp, el compilador se guardará como opción por defecto en la carpeta tasks.json

- Una vez la construcción termine la salida de tu programa aparecerá en la terminal integrada en Visual Studio.

- El código solicitara un valor numérico, por favor ingrese este valor y oprima “enter”

4.Ejemplos de cada función

Código en Python

estaciones.py > ...

```python
class Triangulo:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def clasificar(self):
        triangulo = lambda x, y, z: x + y > z and x + z > y and y + z > x

        if not triangulo(self.a, self.b, self.c):
            return "No es un triángulo"
        if self.a == self.b == self.c:
            return "Equilátero"
        if self.a == self.b or self.a == self.c or self.b == self.c:
            return "Isósceles"
        return "Escaleno"

if __name__ == "__main__":
    a, b, c = map(int, input("Ingrese las longitudes de los lados (a, b, c): ").split())
    triangulo = Triangulo(a, b, c)
    print(triangulo.clasificar())

```

Código explicado a detalle

Class triangulo: Aquí se define una clase llamada Triangulo. Una clase es una plantilla para crear objetos en Python. En este caso, los objetos de la clase Triangulo representarán triángulos con tres lados.

__init__: El método __**init**__ es el constructor de la clase. Se ejecuta automáticamente cuando se crea un objeto de la clase. Este método recibe cuatro parámetros:

-self: Es una referencia al objeto que se está creando. Es obligatorio en todos los métodos de una clase.

-a, b, c: Representan las longitudes de los tres lados del triángulo.

Dentro del constructor, se asignan los valores de **a**, **b** y **c** a los atributos del objeto (**self.a**, **self.b**, **self.c**). Estos atributos almacenan las longitudes de los lados del triángulo.

Clasificar: El método clasificar se encarga de determinar el tipo de triángulo en función de las longitudes de sus lados. Dentro de este método, se define una función lambda llamada triangulo.

Función Lambda: Es una función que se define en una sola línea. En este caso, la función triangulo toma tres parámetros (x, y, z) y verifica si cumplen con la desigualdad triangular. La desigualdad triangular establece que la suma de las

longitudes de dos lados de un triángulo debe ser mayor que la longitud del tercer lado. Si esto no se cumple, las longitudes no pueden formar un triángulo.

verificación de desigualdad triangular: Aquí se utiliza la función triangulo para verificar si las longitudes de los lados (self.a, self.b, self.c) cumplen con la desigualdad triangular. Si no la cumplen, el método retorna el mensaje "No es un triángulo".

Clasificación del triángulo:

Si las longitudes de los lados sí forman un triángulo, se procede a clasificarlo:

-Equilátero: Si los tres lados son iguales (self.a == self.b == self.c), el triángulo es equilátero.

-Isósceles: Si al menos dos lados son iguales (self.a == self.b o self.a == self.c o self.b == self.c), el triángulo es isósceles.

-Escaleno: Si todos los lados son diferentes, el triángulo es escaleno.

Bloque Principal (if __name__ == "__main__":)

Este bloque se ejecuta solo si el script se ejecuta directamente (no cuando se importa como un módulo). Aquí se realiza lo siguiente:

Entrada de Datos:

-Se solicita al usuario que ingrese las longitudes de los tres lados del triángulo. La función input captura la entrada como una cadena de texto.

-La función split() divide la cadena en una lista de subcadenas, utilizando espacios como separadores.

-La función map(int, ...) convierte cada subcadena en un número entero (int). Esto es necesario porque input devuelve una cadena, y necesitamos trabajar con números. Por ejemplo, si el usuario ingresa "3 4 5", split() devuelve ["3", "4", "5"], y map(int, ...) convierte esto en [3, 4, 5].

Resumen del funcionamiento:

1.Entrada: El usuario ingresa las longitudes de los lados del triángulo.

2.Validación: Se verifica si las longitudes pueden formar un triángulo válido.

3.Clasificación: Se determina si el triángulo es equilátero, isósceles, escaleno o si no es un triangulo.

4.Salida: Se muestra el resultado en la consola.

Código en c++

estaciones c++.cpp > main()

```cpp
#include <iostream>
using namespace std;

class Triangulo {
private:
    int a, b, c;

public:
    Triangulo(int a, int b, int c) : a(a), b(b), c(c) {}

    string clasificar() {
        auto triangulo = [this]() {
            return (a + b > c) && (a + c > b) && (b + c > a);
        };

        if (!triangulo()) {
            return "No es un triángulo";
        }
        if (a == b && b == c) {
            return "Equilátero";
        }
        if (a == b || a == c || b == c) {
            return "Isósceles";
        }
        return "Escaleno";
    }
};

int main() {
    int a, b, c;
    cout << "Ingrese las longitudes de los lados (a, b, c): ";
    cin >> a >> b >> c;

    Triangulo triangulo(a, b, c);
    cout << triangulo.clasificar() << endl;

    return 0;
}
```

Inclusión de Bibliotecas y Espacio de Nombres

- `#include <iostream>`: Esta línea incluye la biblioteca estándar de entrada y salida de C++, que permite utilizar `cout` para imprimir en la consola y `cin` para leer datos de entrada.

- `using namespace std;`: Esto permite utilizar los elementos del espacio de nombres `std` sin necesidad de prefijarlos con `std::`. Por ejemplo, se puede usar `cout` en lugar de `std::cout`.

Definición de la Clase `Triangulo`

Aquí se define una clase llamada `Triangulo`. Dentro de la clase, se declaran tres variables privadas (`a`, `b`, `c`) que representan las longitudes de los lados del triángulo. Al ser privadas, no pueden ser accedidas directamente desde fuera de la clase.

Constructor de la Clase.

El constructor de la clase `Triangulo` se define aquí. Este constructor recibe tres parámetros (`a`, `b`, `c`) y los inicializa utilizando una lista de inicialización. Esto significa que las variables miembro de la clase (`this->a`, `this->b`, `this->c`) se inicializan con los valores proporcionados al crear un objeto de la clase.

Método `clasificar`

El método `clasificar` se encarga de determinar el tipo de triángulo en función de las longitudes de sus lados. Dentro de este método, se define una **lambda function** llamada `triangulo`:

Lambda Function.

Es una función que se puede definir en línea. En este caso, la función `triangulo` captura el contexto de la clase (`[this]`) y verifica si las longitudes de los lados cumplen con la desigualdad triangular. La desigualdad triangular establece que la suma de las longitudes de dos lados de un triángulo debe ser mayor que la longitud del tercer lado.

Verificación de la Desigualdad Triangular

Aquí se llama a la función `triangulo` para verificar si las longitudes de los lados (`a`, `b`, `c`) cumplen con la desigualdad triangular. Si no se cumple, el método retorna el mensaje `"No es un triángulo"`

Clasificación del Triángulo

Si las longitudes de los lados sí forman un triángulo, se procede a clasificarlo:

1. Equilátero: Si los tres lados son iguales (`a == b && b == c`), el triángulo es equilátero.

2. Isósceles: Si al menos dos lados son iguales (`a == b` o `a == c` o `b == c`), el triángulo es isósceles.

3. Escaleno: Si todos los lados son diferentes, el triángulo es escaleno.

Función Principal (`main´)

En la función `main`, se realiza lo siguiente:

Declaración de Variables: Se declaran tres variables enteras (`a`, `b`, `c`) para almacenar las longitudes de los lados del triángulo.

Entrada de Datos: Se solicita al usuario que ingrese las longitudes de los lados del triángulo utilizando cout para mostrar el mensaje: "Ingrese las longitudes de los lados (a, b, c): ".

Se utiliza cin para leer los valores ingresados por el usuario y almacenarlos en las variables a, b y c. cin separa automáticamente los valores ingresados por espacios.

Retorno de la Función main:

La función main retorna 0, lo que indica que el programa finalizó correctamente.

Resumen del funcionamiento:

1.Entrada: El usuario ingresa las longitudes de los lados del triángulo.

2.Validación: Se verifica si las longitudes pueden formar un triángulo válido.

3.Clasificación: Se determina si el triángulo es equilátero, isósceles, escaleno o si no es un triángulo.

4. Salida: Se muestra el resultado en la consola.

4.Errores comunes y soluciones

-Entrada invalida: cuando el usuario ingresa valores no validos ej: letras, números negativos o cero

Solución: verificar la entrada para asegurar que los valores sean números enteros positivos para que se ejecute correctamente el programa.

-Uso de decimales o valores flotantes: Si se permiten valores decimales (por ejemplo, **3.5**), las comparaciones de igualdad (a == b) pueden fallar debido a la precisión limitada de los números de punto flotante.

Solucion: En el código usar un “int” para tener en cuenta números enteros colocar un “double” o un “float” que permita recibir datos con números decimales.

5. Como pueden otras personas contribuir al proyecto

Pues al realizar de manera autónoma mi proyecto al momento de subirlo al repositorio se puede dejar abierto o publico para que otros lo analicen y encuentren una forma de hacerlo más sencilla sin extenderse tanto mas sin embargo si hubiese hecho este proyecto en grupo con una o más personas quizá podríamos haber hecho un análisis de los diferentes puntos de vista de cada uno y poder a llegar a una solución diferente o quizá a la ya hecha por mi en este trabajo

6. Licencia

7. Bibliografía

Algunas clases y funciones no vistas en clase fueron extraídas de los archivos de python para poder los códigos de manera eficiente sin necesidad de extender tanto el código Fuente:

https://docs.python.org/3/tutorial/controlflow.html#defining-functions.

https://code.visualstudio.com/docs/cpp/config-mingw#_prerequisites

https://code.visualstudio.com/docs/cpp/config-linux#_prerequisites

https://www.python.org/downloads/

https://marketplace.visualstudio.com/items?itemName=ms-python.python

https://brew.sh/

https://code.visualstudio.com/docs/cpp/config-mingw#_prerequisites