1. Informe escrito (30% de la calificación)
El informe debe contener:
1.1. Análisis del problema
Descripción clara del problema.
Identificación de los requisitos funcionales y no funcionales.
Análisis de casos de uso principales.
Identificación de entradas, procesos y salidas esperadas.
1.3. Diagrama UML
El diagrama debe ser claro, legible y seguir las convenciones UML estándar.

Problema:

Se debe clasificar la conductividad térmica de un material y determinar la categoría en la que se encuentra según las condiciones dadas:

Si k < 0.1: **"Aislante térmico"**
Si 0.1 ≤ k < 1: **"Baja conductividad"**
Si 1 ≤ k < 100: **"Buen conductor"**
Si k ≥ 100: **"Conductor excelente"**

Como el programa final será usado por los ciudadanos del reino de la ciencia esta tiene que ser simple y fácilmente entendible por ellos.

Requisito funcional:
1. Permite la entrada de un número decimal o entero para que cumpla la función.
2. El código debe brindar una salida que clasifique la categoría correcta usando el coeficiente de conductividad térmica (k).
3. El código debe de brindar una salida de número decimal o entero que sea el valor de coeficiente de conductividad térmica (k).
4. Emplear condicionales entre las variables de salida del código.

Requisito no funcional:
1. El código debe ser claro para que los usuarios del reino de la ciencia puedan comprenderla.
2. Debe cumplir con las convenciones de cada lenguaje de programación.
3. Interacción con el usuario.
4. Usar comparaciones simples para optimizar el rendimiento.

Análisis de casos de uso principales:

Caso de Uso	**Entrada**	**Proceso**	**Salida Esperada**
1	k < 0.1	Validar entrada, clasificar material.	“Aislante térmico”
2	0.1 ≤ k < 1	Validar entrada, clasificar material.	“Baja conductividad”
3	1 ≤ k < 100	Validar entrada,	“Buen conductor”

		clasificar material.	
4	k ≥ 100	Validar entrada, clasificar material.	“Excelente conductor”

Identificación de entradas, procesos y salidas esperadas.

Entrada:

- Número decimal o entero: k

Proceso:

1. Identificar si el valor ingresado es válido.
2. Identificar en qué rango se encuentra el material:
    a. Si k < 0.1: **"Aislante térmico"**
    b. Si 0.1 ≤ k < 1: **"Baja conductividad"**
    c. Si 1 ≤ k < 100: **"Buen conductor"**
    d. Si k ≥ 100: **"Conductor excelente”**

Salida:

- Un mensaje que indica la categoría del material.

1.2. Justificación de la solución

Explicación detallada de la estrategia elegida para resolver el problema.
Justificación de las estructuras de datos y algoritmos seleccionados.
Comparación con posibles soluciones alternativas y razones de la elección final.

Explicación:

Se diseña un algoritmo el cual debe recibir un valor (k) y utilizando condicionales lo comparamos con rangos previamente establecidos para asignar las clasificaciones. Luego al ejecutarlo nos debe salir un texto diciendo el resultado. Es una solución simple que es ideal para el contexto del reino de la ciencia, donde es necesario usar conceptos simples y eficientes.

Justificación:

Estructura de datos:

- Los datos que vamos a utilizar se dividen en datos de entrada y de salida.
- Clasificamos los datos en float o double ya que el coeficiente de conductividad es un un decimal.

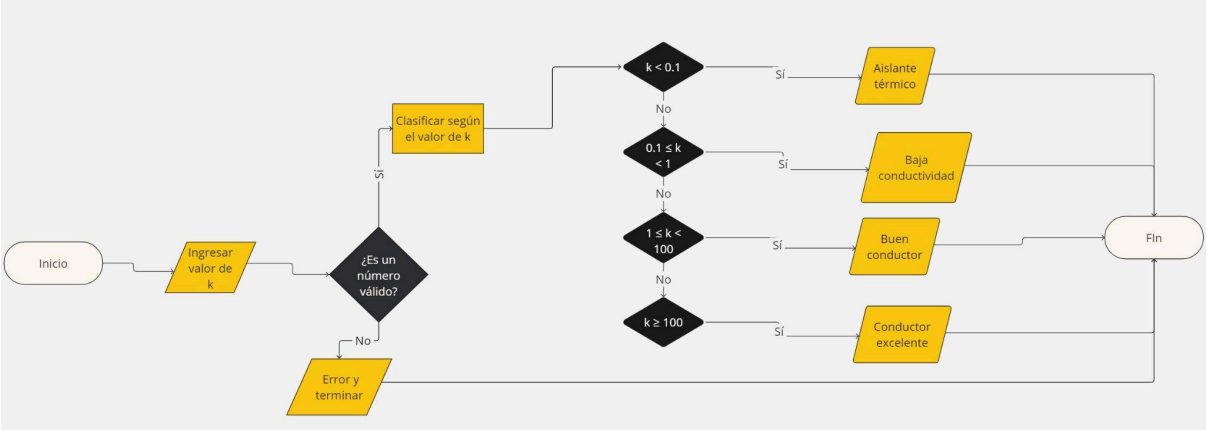
Algoritmo seleccionado:

- Hacemos uso de los condicionales if-elif-else, ya que tenemos varios rangos a evaluar, cuando uno de los condicionales se cumple el algoritmo finaliza y esto nos brinda nuestra respuesta (salida), esto nos ayuda a que nuestro algoritmo sea simple, únicamente se requiere que el usuario brinde una entrada y se brinda una única salida.

Comparación:

Podríamos hacer uso de una función de búsqueda de listas que requeriría recorrer una lista de rangos en cada consulta, lo que lo hace más complejo, por lo que elegimos una solución más sencilla y efectiva basándonos en el análisis del problema y el usuario final.

1.3 Diagrama UML



2. Implementación (50% de la calificación)
2.1. Código en Python
Implementación completa de la solución en Python.
El código debe seguir las convenciones de estilo PEP 8.
3.1. Documentación Python
Comentarios explicativos para secciones complejas.
Instrucciones claras para ejecutar el programa (En los distintos sistemas operativos).

1. Instalar Python en Windows

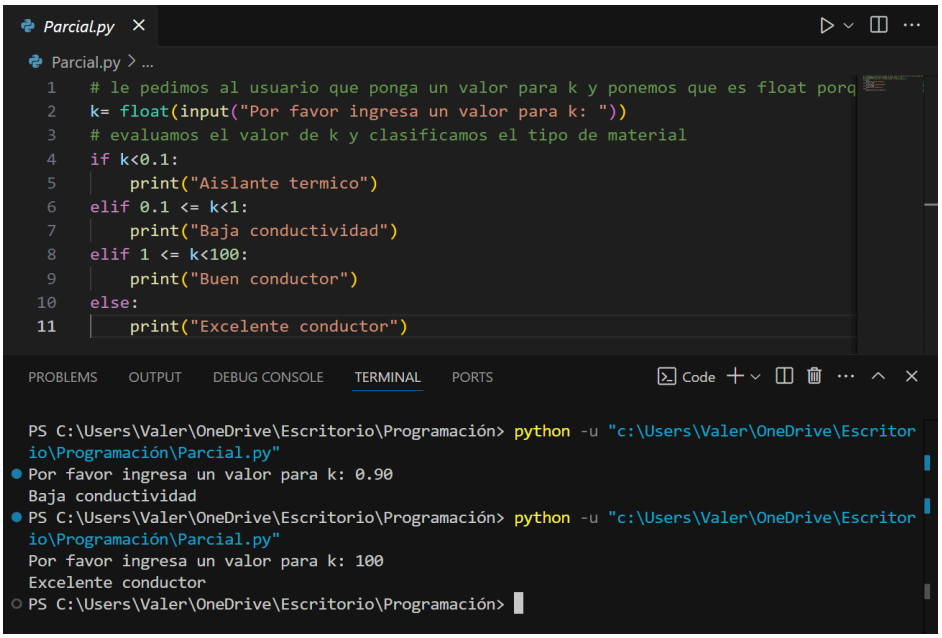
- Instala Python desde python.org y marca la opción "Add Python to PATH".
- Instala Visual Studio Code desde code.visualstudio.com.
- Abre VS Code y ve a la pestaña Extensiones, busca "Python" e instálalo.
- Crea un archivo en VS Code con extensión .py
- Abre la terminal en Terminal > Nueva Terminal.
- 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

Ejemplos de uso

Este código puede ser usado para clasificar valores numéricos en diferentes categorías, sus usos varían en múltiples sectores, financiero (clasificar valores de ingresos/pérdidas), médico (Clasificar valores de resultados de exámenes), deportivo (Clasificar resultados de ejercicios) y hasta Videojuegos (Clasificar valores de resultados de niveles).



2.2. Código en C++
Implementación completa de la solución en C++.

Instrucciones claras para compilar y ejecutar el programa (En los distintos sistemas operativos).

1. Compilar y Ejecutar en Windows
    - 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 sub-carpeta con el nombre de tu espacio, navega a ella y abre Visual Studio con el siguiente comando:
    - `mkdir proyectos`
    - `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”**

2. Compilar y Ejecutar en Linux

- Requisitos:
    - Compilador: Instala GCC (compilador g++ y debugger GDB) (GNU C++ Compiler) y Microsoft Visual Studio.
        - Instala la extensión C++ en Visual Studio.
    - Editor de texto: Puedes usar nano, vim, gedit, o cualquier otro editor.

- Pasos:
    - Instala GCC:
        - Abre una terminal y ejecuta el siguiente comando: `sudo apt-get update`
        - Luego utiliza el siguiente comando en la terminal para instalar las herramientas de compilador GNU y el debugger: `sudo apt-get install build-essential gdb`
        - Crea una carpeta de proyectos desde la terminal para guardar tus proyectos de Visual Studio, luego crea una sub-carpeta con el nombre de tu espacio, navega a ella y abre Visual Studio con el siguiente comando:
            - `mkdir proyectos`
            - `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++ 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”**

3. Compilar y Ejecutar en Mac OS:

    - Requisitos:
        - Compilador: Instala Clang/LLVM (compilador y debugger) y Microsoft Visual Studio.
            - Instala la extensión C++ en Visual Studio.
        - Editor de texto: Puedes usar nano, vim, TextEdit, o cualquier otro editor.

    - Pasos:
        - Instala clang:
            - Abre una terminal y ejecuta el siguiente comando: `xcode-select --install`
            - Crea una carpeta de proyectos desde la terminal para guardar tus proyectos de Visual Studio, luego crea una sub-carpeta con el nombre de tu espacio, navega a ella y abre Visual Studio con el siguiente comando:
                - `mkdir proyectos`
                - `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`].

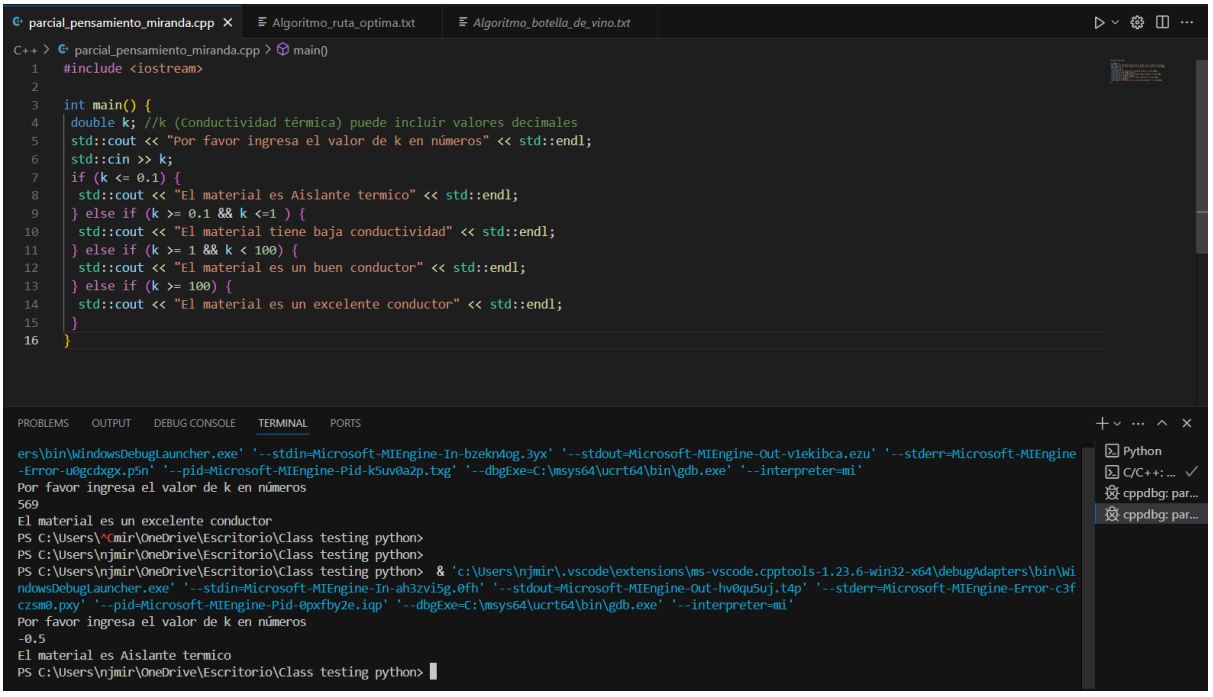
    - Abre el código:

- 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++: clang++ 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 consola debug integrada en Visual Studio.
  - **El código solicitara un valor numérico, por favor ingrese este valor y oprima “enter”**

Ejemplos de uso:

Este código puede ser usado para clasificar valores numéricos en diferentes categorías, sus usos varían en múltiples sectores, financiero (clasificar valores de ingresos/pérdidas), médico (Clasificar valores de resultados de exámenes), deportivo (Clasificar resultados de ejercicios) y hasta Videojuegos (Clasificar valores de resultados de niveles).



Bibliografía

Using GCC with MinGW
Using Clang in Visual Studio Code
Using C++ on Linux in VS Code