OSCILADOR

1. análisis del problema

1.1 Descripción

Este programa determina el tipo de radiación electromagnética dependiendo de la frecuencia que el usuario ingresa. Su propósito es identificar el tipo de radiación. Para esto, el usuario ingresa una base y un exponente, y el programa calcula el tipo de radiación siguiendo las siguientes condiciones:

·Si $f < 3\times10^{9}$ → "Radio"

·Si $3\times10^{9} \leq f < 3\times10^{12}$ → "Microondas"

·Si $3\times10^{12} \leq f < 4.3\times10^{14}$ → "Infrarrojo"

·Si $4.3\times10^{14} \leq f < 7.5\times10^{14}$ → "Visible"

·Si $7.5\times10^{14} \leq f < 3\times10^{17}$ → "Ultravioleta"

·Si $3\times10^{17} \leq f < 3\times10^{19}$ → "Rayos X"

·Si $f \geq 3\times10^{19}$ → "Rayos gamma".

1.2. Identificación de requisitos (Python)

1.2.1. Requisitos funcionales

- Permitir ingresar valores numéricos para una base y un exponente.
- Calcular la frecuencia ingresada por el usuario.
- Determinar el tipo de radiación mediante los valores de frecuencia definidos.
- Manejar mensajes de error cuando el usuario ingresa valores inválidos.
- Mostrar en pantalla el tipo de radiación.

1.2.2. Requisitos no funcionales

- La interfaz debe ser clara y fácil de utilizar.
- El programa debe ser capaz de manejar errores y notificarlos.
- Las respuestas deben mostrarse en tiempo real.
- El programa debe ser compatible con cualquier versión de Python.

2. análisis de casos de uso principales (Python)

Actor: Usuario

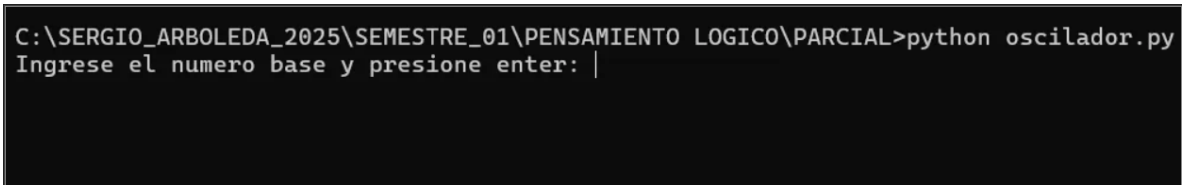
Entorno principal:

2.1. Ejecutar el programa:

El usuario debe ingresar a la carpeta donde está ubicado en programa desde un terminal, puede ser en el terminal del sistema operativo o el terminal del editor de Código y ejecutar el comando “python oscilador.py”.

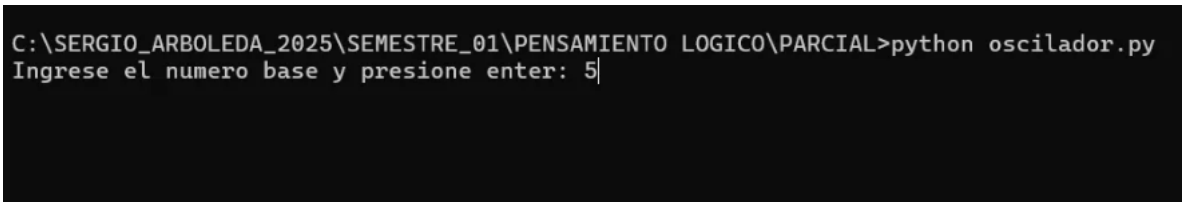
2.2. El programa solicita la base del número (entrada):

El usuario al ejecutar el programa se encontrará con la primera solicitud que es el número base del exponente.



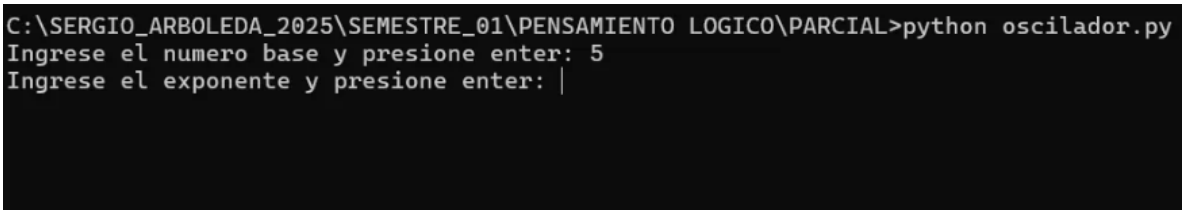
2.3. El usuario ingresa el valor base:

El usuario debe ingresar cualquier valor numérico para determinar la base, en este caso “5” y presionar enter.



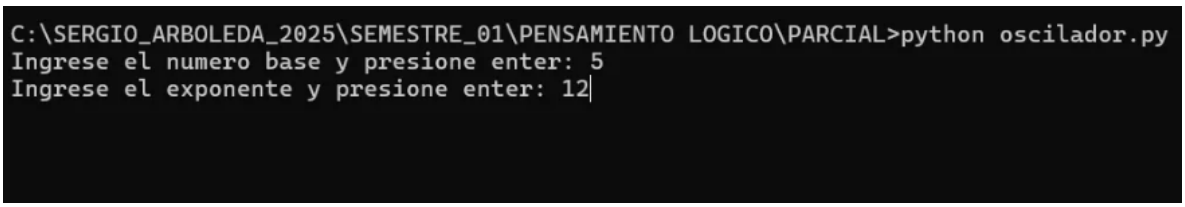
2.4. El programa solicita el exponente (entrada):

Después de presionar enter, el programa desplegará otra solicitud que será el valor del exponente.



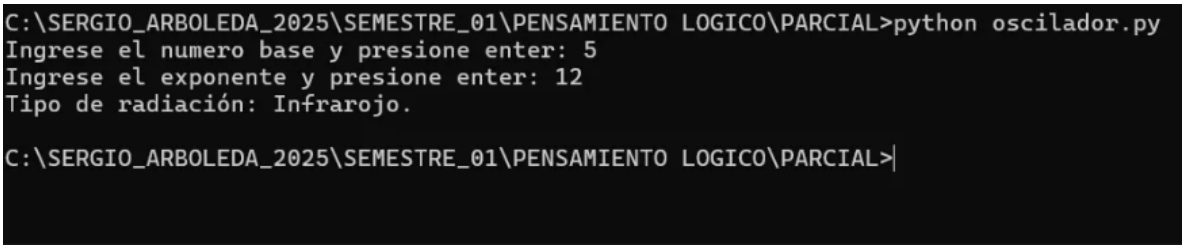
2.5. El usuario ingresa el exponente:

El usuario debe ingresar cualquier valor numérico para determinar el exponente, en este caso “12” y presionar enter.

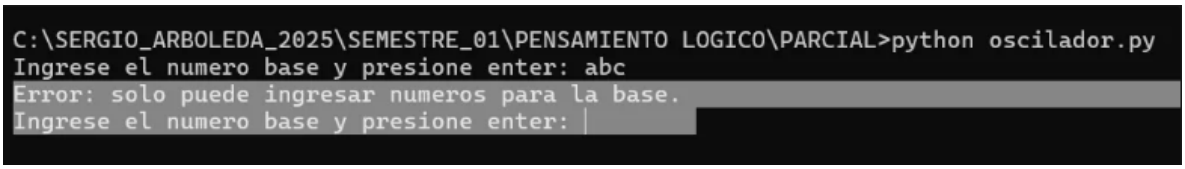


2.6. Resultado (proceso y salida):

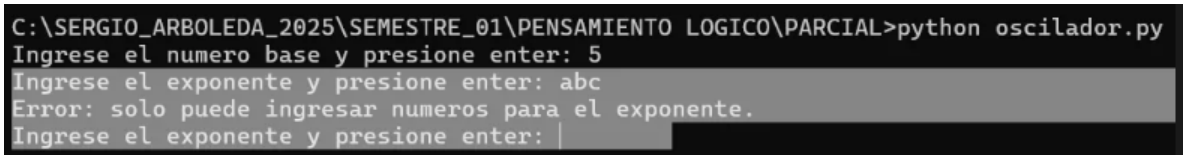
después de presionar enter, el programa realiza los cálculos, muestra el resultado y finaliza el programa.



- **excepciones:**
    - Si el usuario ingresa un valor diferente a un número, por ejemplo, una letra, el programa muestra un mensaje de error y vuelve a solicitar el dato.
        - Base:



        - Exponente:



3. Justificación de la solución

3.1 Estrategia elegida

Se determinó un programa que permitiera al usuario entenderlo visual y funcionalmente. Se utiliza recursos de entrada como **input ()** ‘python’ o **std::cin>> ‘C++’** y estructuras de gestión como bucles y condicionales para obtener una ejecución optima.

4. Justificación de estructuras de datos y algoritmos

4.1. Código completo en Python:

```python
# Bucle que solicita el número base hasta recibir un valor válido
while True:
    try: # Código que puede contener un error
        numero_base = float(input("Ingrese el numero base y presione enter: "))
        break # Rompe el bucle si la entrada es válida
    except ValueError: # Usamos except para clasificar el error
        print("Error: solo puede ingresar numeros para la base.")

# Bucle que solicita el exponente hasta recibir un valor válido
while True:
    try: # Código que puede contener un error
        exponente = float(input("Ingrese el exponente y presione enter: "))
        break # Rompe el bucle si la entrada es válida
    except ValueError: # usamos except para clasificar el error
        print("Error: solo puede ingresar numeros para el exponente.")

# Cálculo de la frecuencia
f = numero_base * 10**exponente

# Definir límites de cada tipo de radiación en Hercios
radio = 3e9
microondas = 3e12
infrarojo = 4.3e14
visible = 7.5e14
ultravioleta = 3e17
rayos_x = 3e19

```

```python
# Definir el tipo de radiación según la frecuencia
if f < radio:
    print("Tipo de radiación: Radio.")
elif f < microondas:
    print("Tipo de radiación: Microondas.")
elif f < infrarojo:
   print("Tipo de radiación: Infrarojo.")
elif f < visible:
    print("Tipo de radiación: Visible.")
elif f < ultravioleta:
    print("Tipo de radiación: Ultravioleta.")
elif f < rayos_x:
    print("Tipo de radiación: Rayos X.")
else:
    print("Tipo de radiación: Rayos gamma.")
```

4.1.1. Entrada para el número base y el exponente:

```python
# Bucle que solicita el número base hasta recibir un valor válido
while True:
    try: # Código que puede contener un error
        numero_base = float(input("Ingrese el numero base y presione enter: "))
        break # Rompe el bucle si la entrada es válida
    except ValueError: # Usamos except para clasificar el error
        print("Error: solo puede ingresar numeros para la base.")

# Bucle que solicita el exponente hasta recibir un valor válido
while True:
    try: # Código que puede contener un error
        exponente = float(input("Ingrese el exponente y presione enter: "))
        break # Rompe el bucle si la entrada es válida
    except ValueError: # usamos except para clasificar el error
        print("Error: solo puede ingresar numeros para el exponente.")

```

Input(): Para obtener los datos que ingresa el usuario se creó una variable que guarda un input() para almacenar el valor que el usuario ingresa, el input está dentro de un float () para que el usuario pueda escribir números decimales y realizar operaciones con esos valores.

Try-except: Este recurso nos permite controlar errores cuando una ejecución tiene un fallo, en este caso, para los input () lo almacenamos en try ya que si el usuario ingresa un valor que no sea numérico, por ejemplo una letra, este error lo controlará except mostrando un mensaje por pantalla. Esto evita que el programa falle y se cierre si detecta un error.

While y break: La función de while es ejecutar indefinidamente la entrada hasta que el usuario ponga un dato valido y el ciclo se termine con el break, si el usuario no inserta un numero en input, while mostrará el contenido de except y mostrará de nuevo el input para un nuevo valor.

4.1.2. Cálculo de la frecuencia y determinar los límites de cada tipo de radiación:

```python
# Cálculo de la frecuencia
f = numero_base * 10**exponente

# Definir límites de cada tipo de radiación en Hercios
radio = 3e9
microondas = 3e12
infrarojo = 4.3e14
visible = 7.5e14
ultravioleta = 3e17
rayos_x = 3e19

```

Ya obtenidos los valores del usuario se hace el respectivo cálculo para obtener la frecuencia en la variable f, también se clasificó la frecuencia de cada tipo de radiación en variables para que podamos manipular sus datos de manera más sencilla.

4.1.3. Condicional para determinar el tipo de radiación:

```
# Definir el tipo de radiación según la frecuencia
if f < radio:
    print("Tipo de radiación: Radio.")
elif f < microondas:
    print("Tipo de radiación: Microondas.")
elif f < infrarojo:
   print("Tipo de radiación: Infrarojo.")
elif f < visible:
    print("Tipo de radiación: Visible.")
elif f < ultravioleta:
    print("Tipo de radiación: Ultravioleta.")
elif f < rayos_x:
    print("Tipo de radiación: Rayos X.")
else:
    print("Tipo de radiación: Rayos gamma.")
```

Para obtener el tipo de radiación con los datos obtenidos, se usa una condicional que determina su resultado. En esta condicional nos dice que, si el valor de f es menor al valor máximo de cada variable de radiación, el resultado va a ser la radiación que f no alcanzó a superar.

5. Identificación de requisitos (C++)

5.1. Requisitos funcionales

- Permitir ingresar valores numéricos para una base y un exponente.
- Calcular la frecuencia ingresada por el usuario.
- Determinar el tipo de radiación mediante los valores de frecuencia definidos.
- Manejar mensajes de error cuando el usuario ingresa valores inválidos.
- Mostrar en pantalla el tipo de radiación.

5.2. Requisitos no funcionales

- La interfaz debe ser clara y fácil de utilizar.
- El programa debe ser capaz de manejar errores y notificarlos.
- Las respuestas deben mostrarse en tiempo real.
- El programa debe ser compatible con cualquier versión de C++.

6. análisis de casos de uso principales (C++)

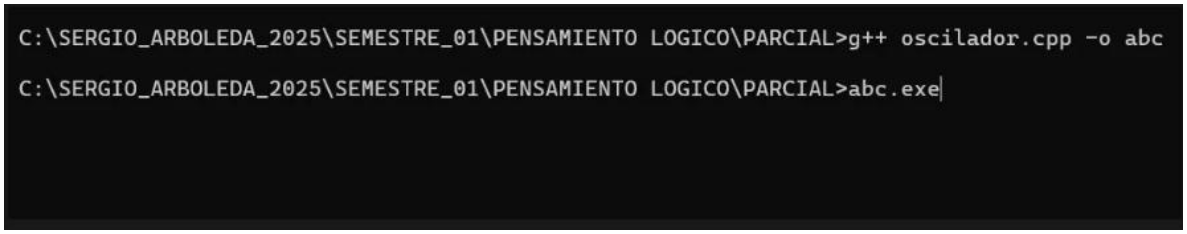
Actor: Usuario

Entorno principal:

6.1. Ejecutar el programa.

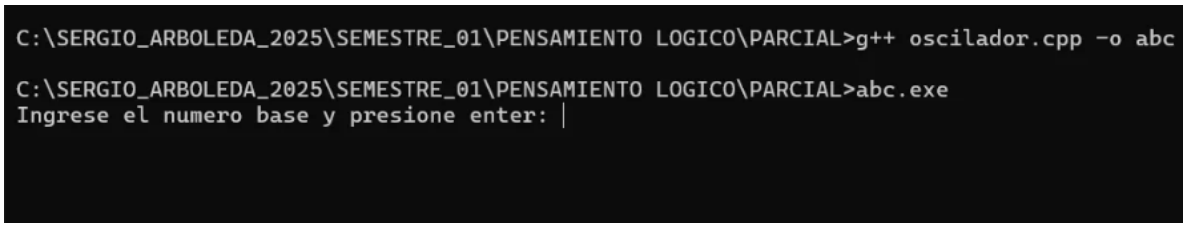
1. Instalar un compilador para ejecutar C++, preferiblemente Mingw64 si están en Windows, para descargar el compilador ver el siguiente tutorial: https://youtu.be/v3ENcQpoA5A

2. Abrir el terminal de Windows o del editor de Código y ubicar la carpeta donde está guardado el programa.
3. ejecutar en el terminal el siguiente comando y presionar enter: g++ oscilador.cpp -o abc.
4. Nos dará un archivo .exe con el nombre que le asignamos después de '-o', y escribimos ese archivo .exe y presionamos enter.



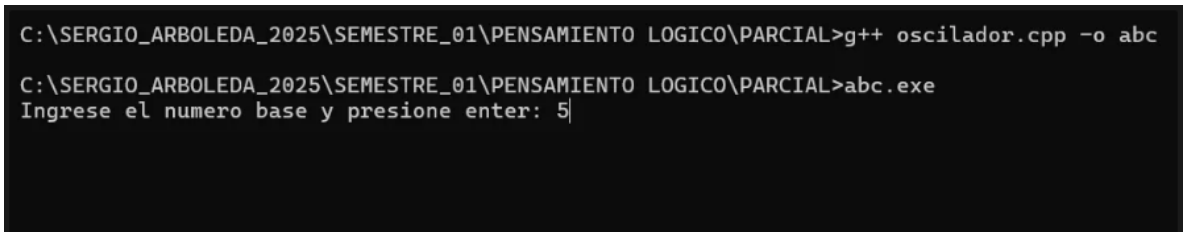
6.2. El programa solicita la base del número (entrada):

El usuario al ejecutar el programa se encuentra con la primera solicitud que es el número base del exponente.



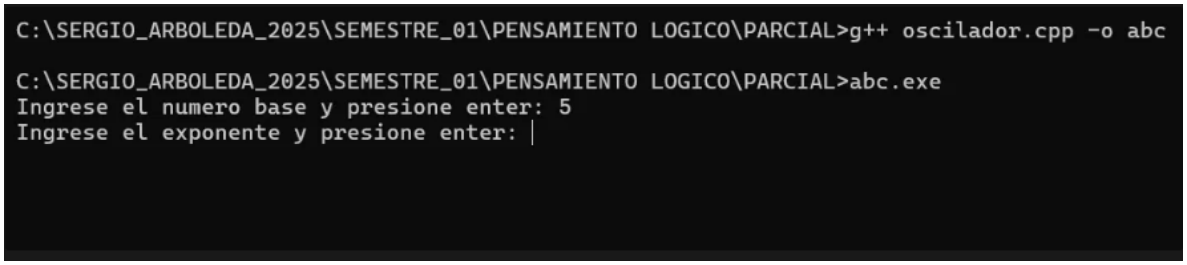
6.3. El usuario ingresa el valor base:

El usuario debe ingresar cualquier valor numérico para determinar la base, en este caso "5" y presionar enter.



6.4. El programa solicita el exponente (entrada):

después de presionar enter, el programa desplegará otra solicitud que será el valor del exponente.



6.5. El usuario ingresa el exponente:

El usuario debe ingresar cualquier valor numérico para determinar el exponente, en este caso “12” y presionar enter.

```
C:\SERGIO_ARBOLEDA_2025\SEMESTRE_01\PENSAMIENTO LOGICO\PARCIAL>g++ oscilador.cpp -o abc

C:\SERGIO_ARBOLEDA_2025\SEMESTRE_01\PENSAMIENTO LOGICO\PARCIAL>abc.exe
Ingrese el numero base y presione enter: 5
Ingrese el exponente y presione enter: 12
```

6.6. Resultado (proceso y salida):

después de presionar enter, el programa realiza los cálculos, muestra el resultado y finaliza el programa.

```
C:\SERGIO_ARBOLEDA_2025\SEMESTRE_01\PENSAMIENTO LOGICO\PARCIAL>g++ oscilador.cpp -o abc

C:\SERGIO_ARBOLEDA_2025\SEMESTRE_01\PENSAMIENTO LOGICO\PARCIAL>abc.exe
Ingrese el numero base y presione enter: 5
Ingrese el exponente y presione enter: 12
Tipo de radiacion: Infrarojo.

C:\SERGIO_ARBOLEDA_2025\SEMESTRE_01\PENSAMIENTO LOGICO\PARCIAL>
```

- **excepciones:**
    - Si el usuario ingresa un valor diferente a un número, por ejemplo, una letra, el programa muestra un mensaje de error y vuelve a solicitar el dato.
        - Base:

```
C:\SERGIO_ARBOLEDA_2025\SEMESTRE_01\PENSAMIENTO LOGICO\PARCIAL>g++ oscilador.cpp -o abc

C:\SERGIO_ARBOLEDA_2025\SEMESTRE_01\PENSAMIENTO LOGICO\PARCIAL>abc.exe
Ingrese el numero base y presione enter: abc
Error: Solo puede ingresar numeros para la base.
Ingrese el numero base y presione enter:
```

        - Exponente:

```
C:\SERGIO_ARBOLEDA_2025\SEMESTRE_01\PENSAMIENTO LOGICO\PARCIAL>g++ oscilador.cpp -o abc

C:\SERGIO_ARBOLEDA_2025\SEMESTRE_01\PENSAMIENTO LOGICO\PARCIAL>abc.exe
Ingrese el numero base y presione enter: 5
Ingrese el exponente y presione enter: abc
Error: Solo puede ingresar numeros para el exponente.
Ingrese el exponente y presione enter:
```

7. Justificación de la solución

7.1. Estrategia elegida

Se determinó un programa que permitiera al usuario entenderlo visual y funcionalmente. Se utiliza recursos de entrada como **input()** 'python' o **std::cin>> 'C++'** y estructuras de gestión como bucles y condicionales para obtener una ejecución optima.

8. Justificación de estructuras de datos y algoritmos

8.1. Código completo en C++

```cpp
#include <iostream>
#include <cmath> // Para usar pow()
#include <limits> // Para usar numeric_limits<streamsize>
using namespace std;

int main() {
    float numero_base;
    // Bucle que solicita el numero base hasta recibir un valor válido
    while (true) {
        std::cout << "Ingrese el numero base y presione enter: ";
        if (std::cin >> numero_base) {
            break; // Rompe el bucle si la entrada es válida
        } else {
            std::cerr << "Error: Solo puede ingresar numeros para la base." << std::endl;
            std::cin.clear(); // Limpia los datos guardados en cin
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Limpia el bufer de cin
        }
    }

    float exponente;
    // Bucle que solicita el exponente hasta recibir un valor válido
    while (true) {
        std::cout << "Ingrese el exponente y presione enter: ";
        if (std::cin >> exponente) {
            break; // Rompe el bucle si la entrada es válida
        } else {
            std::cerr << "Error: Solo puede ingresar numeros para el exponente." << std::endl;
            std::cin.clear(); // Limpia los datos guardados en cin
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Limpia el bufer de cin
        }
    }

    // Cálculo de la frecuencia y usando pow() para calcular el exponente
    double f = numero_base * pow(10, exponente);

    // Definir límites de cada tipo de radiación en hercios
    double radio = 3e9;
    double microondas = 3e12;
    double infrarojo = 4.3e14;
    double visible = 7.5e14;
    double ultravioleta = 3e17;
    double rayos_x = 3e19;

    // Definir el tipo de radiación según la frecuencia
    if (f < radio) {
        std::cout << "Tipo de radiacion: Radio." << std::endl;
    } else if (f < microondas) {
        std::cout << "Tipo de radiacion: Microondas." << std::endl;
    } else if (f < infrarojo) {
        std::cout << "Tipo de radiacion: Infrarojo." << std::endl;
    } else if (f < visible) {
        std::cout << "Tipo de radiacion: Visible." << std::endl;
    } else if (f < ultravioleta) {
        std::cout << "Tipo de radiacion: Ultravioleta." << std::endl;
    } else if (f < rayos_x) {
        std::cout << "Tipo de radiacion: Rayos X." << std::endl;
    } else {
        std::cout << "Tipo de radiacion: Rayos gamma." << std::endl;
    }

    return 0;
}
```

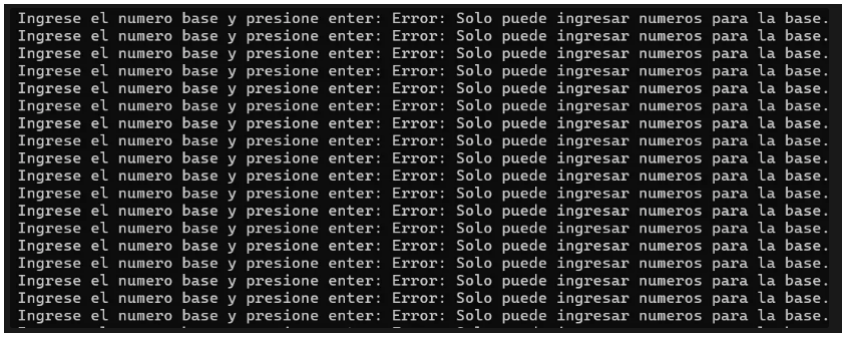
8.1.1. Entrada para el número base y el exponente:

```cpp
#include <iostream>
#include <cmath> // Para usar pow()
#include <limits> // Para usar numeric_limits<streamsize>
using namespace std;

int main() {
    float numero_base;
    // Bucle que solicita el numero base hasta recibir un valor válido
    while (true) {
        std::cout << "Ingrese el numero base y presione enter: ";
        if (std::cin >> numero_base) {
            break; // Rompe el bucle si la entrada es válida
        } else {
            std::cerr << "Error: Solo puede ingresar numeros para la base." << std::endl;
            std::cin.clear(); // Limpia los datos guardados en cin
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Limpia el bufer de cin
        }
    }

    float exponente;
    // Bucle que solicita el exponente hasta recibir un valor válido
    while (true) {
        std::cout << "Ingrese el exponente y presione enter: ";
        if (std::cin >> exponente) {
            break; // Rompe el bucle si la entrada es válida
        } else {
            std::cerr << "Error: Solo puede ingresar numeros para el exponente." << std::endl;
            std::cin.clear(); // Limpia los datos guardados en cin
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Limpia el bufer de cin
        }
    }

```

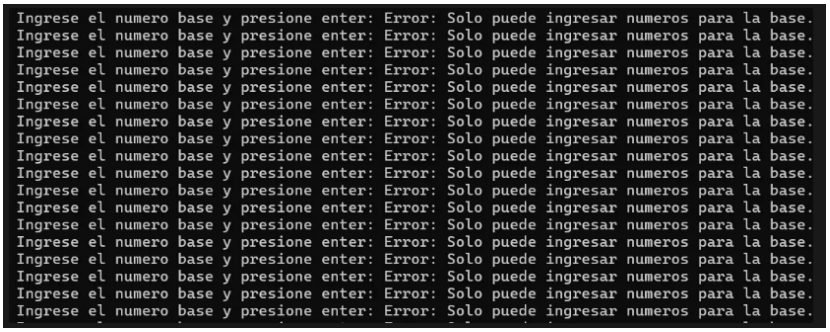
Para obtener los datos ingresados por el usuario se utilizaron los siguientes recursos:

- **‘float numero_base;’:** Esta variable está definida como float para que el usuario pueda insertar números decimales.
- **‘std::cin >> numero base’:** En esta entrada vamos a recibir el valor ingresado por el usuario y guardarlo en la variable numero_base.
- If-else: Si los valores que ingresa el usuario son válidos, el break rompe el bucle que hace while, pero si los datos ingresados son erróneos, por ejemplo, las letras, la condicional imprime un mensaje de error y lo clasificamos con ‘cerr’.

- std::cin.clear(): Permite restablecer los indicadores de error, si el usuario inserta un valor erróneo, la función cin.clear() borra esos indicadores para que std::sin vuelva a funcionar correctamente, si no se pone esta función, el while creará un ciclo repetitivo del mismo error con la misma entrada.



- **std::cin.ignore(std::numeric_limits<std::streamsize::max(), '\n'):** Funciona parecido a 'std::cin.clear()' pero su función es borrar los caracteres que están guardados en el buffer de entrada, también es importante implementar esta función porque si no lo agregamos va a pasar el mismo error con el std::cin.clear(). el carácter '\n' marca donde se acaba una línea y empieza la siguiente.



while y break: ya definidas las entradas, las condicionales y los errores, entra el ciclo a cumplir su función, si el usuario ingresa un valor valido en la entrada, el bucle se rompe con break, pero si ingresa datos erróneos, mostrará el mensaje de error y solicitará de nuevo el valor de la entrada.

8.1.2. Cálculo de la frecuencia y determinar los límites de cada tipo de radiación:

```
32
33      // Cálculo de la frecuencia y usando pow() para calcular el exponente
34      double f = numero_base * pow(10, exponente);
35
36      // Definir límites de cada tipo de radiación en hercios
37      double radio = 3e9;
38      double microondas = 3e12;
39      double infrarojo = 4.3e14;
40      double visible = 7.5e14;
41      double ultravioleta = 3e17;
42      double rayos_x = 3e19;
43
```

Se obtiene los valores de la base y el exponente escritas por el usuario y se realiza el cálculo para definir la frecuencia. Se utiliza pow () (extraído de la biblioteca <cmath>) para calcular la potencia de un número, es de vital importancia implementarlo para este ejercicio. Se definen los tipos de radio en variables mediante double para almacenar valores enteros y decimales, también se clasificaron de esta forma para manipular los valores de los tipos de radio de manera sencilla.

8.1.3. Condicional para determinar el tipo de radiación:

```
43
44      // Definir el tipo de radiación según la frecuencia
45      if (f < radio) {
46          std::cout << "Tipo de radiacion: Radio." << std::endl;
47      } else if (f < microondas) {
48          std::cout << "Tipo de radiacion: Microondas." << std::endl;
49      } else if (f < infrarojo) {
50          std::cout << "Tipo de radiacion: Infrarojo." << std::endl;
51      } else if (f < visible) {
52          std::cout << "Tipo de radiacion: Visible." << std::endl;
53      } else if (f < ultravioleta) {
54          std::cout << "Tipo de radiacion: Ultravioleta." << std::endl;
55      } else if (f < rayos_x) {
56          std::cout << "Tipo de radiacion: Rayos X." << std::endl;
57      } else {
58          std::cout << "Tipo de radiacion: Rayos gamma." << std::endl;
59      }
60
61      return 0;
62  }
```

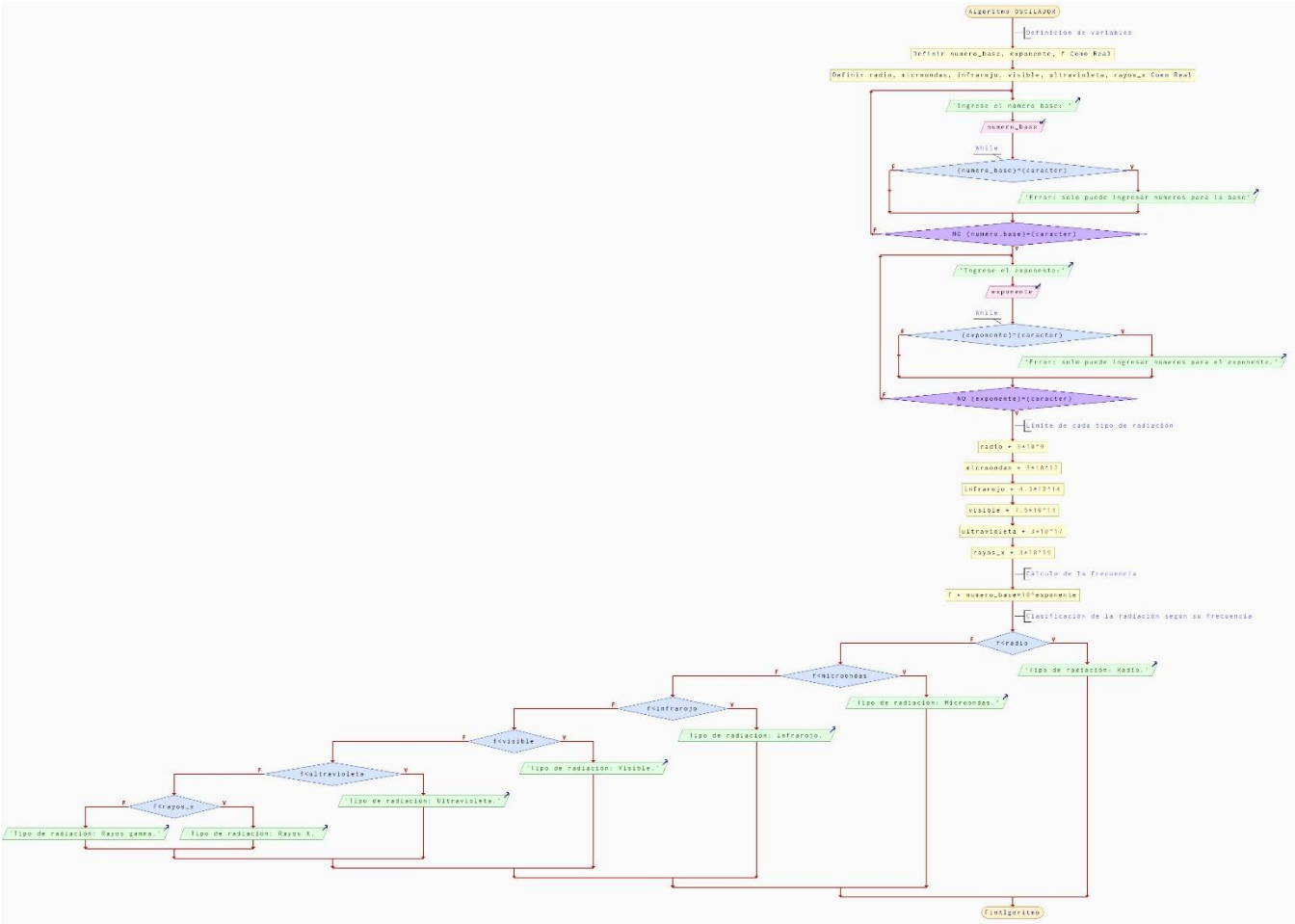
Con el valor de la frecuencia ya podemos calcular el tipo de radiación, se utilizó if-else para hacer el procedimiento, la variable f se compara con el valor de cada radiación y si no supera el valor máximo de la radiación se va a determinar por el nombre de la radiación que no superó.

9. Comparación con Posibles Soluciones Alternativas (aplica para los dos lenguajes):

A nivel de interfaz se pudo agregar más mensajes, una presentación visual más agradable para el usuario o una estructura de Código más compleja para tener resultados más específicos, pero el propósito del desarrollo de estos programas es mostrar de manera simple el resultado a partir de los datos ingresados por un usuario. Es posible añadir más cosas para que el programa tenga más funcionalidades y pueda resolver diferentes problemas.

10. Diagrama de flujo:

Este diagrama es el paso a paso de la ejecución del programa para ambos lenguajes, se desarrolló primero un Pseudocódigo para traducirlo a diagrama de flujo:



11. Pseudocódigo:

```
Algoritmo OSCILADOR
    // Definición de variables
    Definir numero_base, exponente, f Como Real
    Definir radio, microondas, infrarojo, visible, ultravioleta, rayos_x Como Real

    Repetir
        Escribir 'Ingrese el número base: '
        Leer numero_base // While
        Si (numero_base)=(caracter) Entonces
            Escribir 'Error: solo puede ingresar números para la base'
        FinSi
    Hasta Que  NO (numero_base)=(caracter)

    Repetir
        Escribir 'Ingrese el exponente:'
        Leer exponente // While
        Si (exponente)=(caracter) Entonces
            Escribir 'Error: solo puede ingresar números para el exponente.'
        FinSi
    Hasta Que  NO (exponente)=(caracter)

    // Limite de cada tipo de radiación
    radio ← 3*10↑9
    microondas ← 3*10↑12
    infrarojo ← 4.3*10↑14
    visible ← 7.5*10↑14
    ultravioleta ← 3*10↑17
    rayos_x ← 3*10↑19


    // Cálculo de la frecuencia
    f ← numero_base*10↑exponente
    // Clasificación de la radiación según su frecuencia

    Si f < radio Entonces
        Escribir 'Tipo de radiación: Radio.'
    SiNo
        Si f < microondas Entonces
            Escribir 'Tipo de radiación: Microondas.'
        SiNo
            Si f < infrarojo Entonces
                Escribir 'Tipo de radiación: Infrarojo.'
            SiNo
                Si f < visible Entonces
                    Escribir 'Tipo de radiación: Visible.'
                SiNo
                    Si f < ultravioleta Entonces
                        Escribir 'Tipo de radiación: Ultravioleta.'
                    SiNo
                        Si f < rayos_x Entonces
                            Escribir 'Tipo de radiación: Rayos X.'
                        SiNo
                            Escribir 'Tipo de radiación: Rayos gamma.'
                        FinSi
                    FinSi
                FinSi
            FinSi
        FinSi
    FinSi
FinAlgoritmo
```

Licencias:

Visual studio Code: Desarrollo de programa en Python y C++

Mingw64: Compilador para C++

Pseint: Desarrollo de diagrama de flujo

Python: lenguaje de programación descargado desde la página web oficial https://www.python.org/

Windows 11: Sistema operativo donde se trabajó todo el proyecto

ELABORADO POR:

SEBASTIAN PAEZ CASTAÑEDA