

Documentación Parcial 1 Pensamiento algorítmico

Tema: Radiación al Límite

Nombre: Juan Felipe Gonzalez Barrera

1. Descripción del problema

Se debe determinar la categoría de un isótopo para la supervivencia y para los avances de la ciencia en función de su vida media T (el tiempo que tarda en desintegrarse la mitad de su masa). Un isótopo con una vida media corta es altamente radiactivo, mientras que uno con vida media larga es más estable y seguro. Según se determinó los siguientes criterios:

- Si $T \leq 1$ segundo \rightarrow "Alta radiactividad"
- Si $1 < T \leq 86400$ segundos \rightarrow "Radiactivo moderado"
- Si $T > 86400$ segundos \rightarrow "Baja radiactividad"

Requisitos Funcionales:

1. El programa debe aceptar un número entero que represente la vida media del isótopo en nanosegundos.
2. El programa debe convertir la vida media de nanosegundos a segundos.
3. El programa debe clasificar el isótopo en una de las tres categorías basándose en el valor de los nanosegundos.
4. El programa debe mostrar la clasificación del isótopo.

Requisitos No Funcionales:

1. El programa debe ser eficiente en términos de tiempo de ejecución.
2. El código debe ser claro y fácil de entender.
3. El programa debe manejar correctamente los valores de entrada y no debe fallar en caso de entradas inesperadas.

Análisis de casos de uso:

- **Caso 1:** El usuario ingresa un valor de nanosegundos que es menor o igual a 1×10^9 nanosegundos (1 segundo). El programa debe clasificar el isótopo como de "Alta radiactividad".
- **Caso 2:** El usuario ingresa un valor de nanosegundos que está entre 1×10^9 y 86400×10^9 nanosegundos (1 segundo a 86400 segundos). El programa debe clasificar el isótopo como de "Radiactividad moderada".
- **Caso 3:** El usuario ingresa un valor de nanosegundos que es mayor a 86400×10^9 nanosegundos (más de 86400 segundos). El programa debe clasificar el isótopo como de "Baja radiactividad".

Identificación de Entradas, Procesos y Salidas:

- Entrada: Un número entero que representa la vida media del isótopo en nanosegundos.
- Procesos:
 1. Convertir la vida media de nanosegundos a segundos.
 2. Comparar el valor de nanosegundos con los umbrales definidos para determinar la categoría.
- Salida: Una cadena de texto que indica la clasificación del isótopo.

Estrategia Elegida:

La estrategia elegida para resolver el problema es simple y directa. Se toma la vida media en nanosegundos, se convierte a segundos y luego se compara con las condiciones definidas para determinar la categoría de radiactividad. Esta estrategia es eficiente y fácil de implementar.

Algoritmos Seleccionados:

- Algoritmos: Se utiliza una serie de condicionales (if-else) para comparar el valor de la variable nanosegundos con las condiciones y determinar la categoría. Este enfoque es adecuado dado que el problema tiene un número limitado de condiciones.

Comparación con Soluciones Alternativas:

- Alternativa 1: Uso de algoritmos mas simples tales como lo pueden ser el elif para dar solución al problema con menos líneas de código
- Alternativa 2: Uso de algoritmos los cuales son los utilizados en clase y darle una solución adecuada y más simple de entender al ser una estructura mas marcada.

Razones de la Elección Final:

La solución final se basa en condicionales simples porque es la más directa y fácil de entender para este problema específico. No se requieren estructuras de datos complejas ni algoritmos avanzados, lo que hace que el código sea claro y eficiente.

Conclusión

El problema de clasificación de isótopos radiactivos se resuelve de manera efectiva utilizando una estrategia simple basada en condicionales. La solución es eficiente, fácil de implementar y cumple con todos los requisitos funcionales y no funcionales.

2. Instalación

Requisitos Previos:

Python: Asegúrate de tener Python instalado en tu sistema. Puedes descargarlo desde python.org.

Entorno de Desarrollo: Puedes usar un editor de texto como Visual Studio Code o cualquier otro que prefieras.

Instalación:

No se requieren bibliotecas externas para este proyecto, ya que utiliza solo funciones básicas de Python.

Simplemente descarga o copia el archivo `tipos_de_radioactividad.py` en tu directorio de trabajo.

3. Guía de uso

Ejecución del Programa:

Al ejecutar el script, se te pedirá que ingreses la cantidad de nanosegundos.

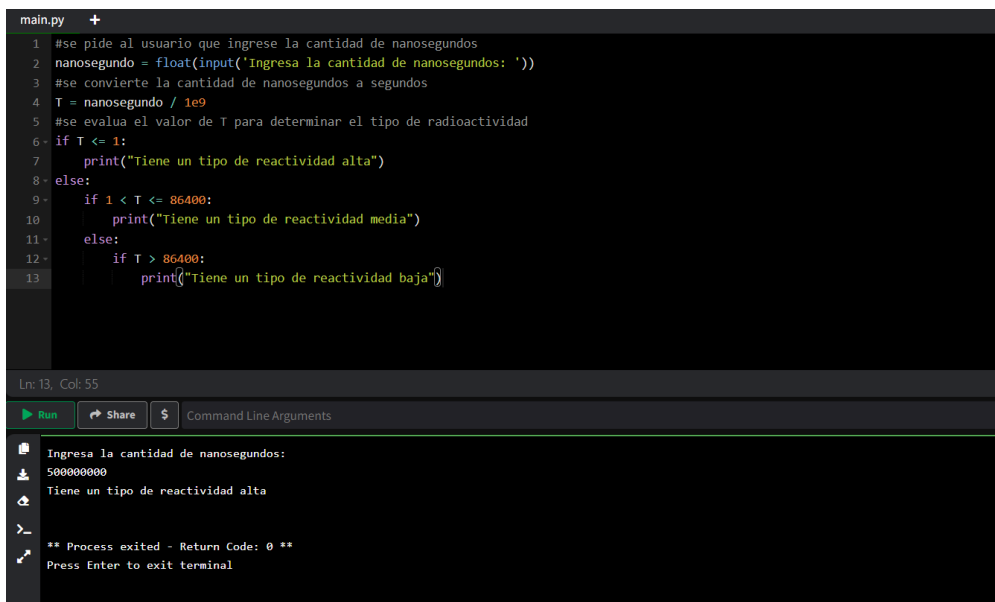
Ingresa un valor numérico entero (por ejemplo, 500000000 para 0.5 segundos).

Interacción:

El programa convertirá automáticamente los nanosegundos a segundos y determinará el tipo de reactividad.

Mostrará un mensaje indicando si la reactividad es alta, media o baja.

Ejemplo de Uso:



```
main.py +
1 #se pide al usuario que ingrese la cantidad de nanosegundos
2 nanosegundo = float(input('Ingresa la cantidad de nanosegundos: '))
3 #se convierte la cantidad de nanosegundos a segundos
4 T = nanosegundo / 1e9
5 #se evalua el valor de T para determinar el tipo de radioactividad
6 if T <= 1:
7     print("Tiene un tipo de reactividad alta")
8 else:
9     if 1 < T <= 86400:
10         print("Tiene un tipo de reactividad media")
11     else:
12         if T > 86400:
13             print("Tiene un tipo de reactividad baja")

Ln: 13, Col: 55
Run Share Command Line Arguments
Ingresa la cantidad de nanosegundos:
500000000
Tiene un tipo de reactividad alta
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

```
main.cpp +
1 #include <iostream>
2 using namespace std;
3 //En este caso se hace un programa que recibe una cantidad de nanosegundos y determina si es de alta, media o baja reactividad
4 //En la siguiente línea se hace el ingreso de las variables
5 int main() {
6     float t;
7     // se pide al usuario que ingrese la cantidad de nanosegundos
8     cout << "ingrese la cantidad de nanosegundo" << endl;
9     cin >> t;
10    //se hace la conversion del dato para darlo en segundos
11    float tmedio = t / 1e9;
12    //se hace la comparacion de la cantidad de segundos para determinar si es de alta, media o baja reactividad
13    if (tmedio <= 1) {
14        cout << "tiene una vida media de alta reactividad" << endl;
15    } else if (tmedio <= 86400) {
16        cout << "tiene una vida media de media reactividad" << endl;
17    } else {
```

Ln: 20, Col: 8

Run Share Command Line Arguments

```
ingrese la cantidad de nanosegundo
5000000000
tiene una vida media de media reactividad

** Process exited - Return Code: 0 **
```

4. Documentación Técnica

Descripción de la estructura interna del proyecto, clases, métodos, y cómo están organizados:

Estructura del Proyecto:

- El proyecto consiste en un solo archivo Python (tipos_de_radioactividad.py).
- No utiliza clases ni métodos adicionales, ya que es un script simple.

Flujo del Programa:

- Entrada: El usuario ingresa la cantidad de nanosegundos.
- Procesamiento:
 - Se convierte el valor de nanosegundos a segundos.
 - Se evalúa el valor de TT para determinar la categoría de reactividad.
- Salida: Se imprime el tipo de reactividad.

Variables:

- nanosegundo: Almacena la entrada del usuario en nanosegundos.
- T: Almacena el valor convertido a segundos.

5. Explicación del código

- Entrada de Datos python:

El programa comienza solicitando al usuario que ingrese un valor entero que representa la vida media de un isótopo radiactivo en nanosegundos.

La función `input ()` captura la entrada del usuario como una cadena de texto, y `int ()` la convierte a un número entero.

Ejemplo: Si el usuario ingresa 500000000, esto representa 500 millones de nanosegundos.

- Conversión de Unidades:

La vida media en nanosegundos se convierte a segundos dividiendo el valor entre 1×10^9 (es decir, $1e9$).

Ejemplo: 500000000 nanosegundos se convierten en 0.5 segundos.

- Evaluación de la Reactividad:

El programa utiliza una serie de condicionales if-else para determinar la categoría de reactividad del isótopo:

- Si $T \leq 1$ segundo \rightarrow "Alta radiactividad"
- Si $1 < T \leq 86400$ segundos \rightarrow "Radiactivo moderado"
- Si $T > 86400$ segundos \rightarrow "Baja radiactividad".

Anidación de Condicionales:

Aunque el código funciona correctamente, la anidación de if-else puede dificultar la lectura. Una alternativa más limpia sería usar elif, pero en este caso, se mantiene la estructura solicitada.

6. Errores comunes

- Error: Entrada no válida (no es un número entero):

Solución: Asegúrate de ingresar un número entero. Puedes agregar manejo de excepciones para evitar que el programa falle:

- Error: El programa no muestra ninguna salida:

Solución: Verifica que el valor de T

T esté dentro de los rangos esperados. Asegúrate de que el código de evaluación esté correctamente escrito.

- Error: El programa muestra un resultado incorrecto:

Solución: Revisa la conversión de nanosegundos a segundos. Asegúrate de que el valor de T

T se calcule correctamente.

1. Inclusión de Bibliotecas c++:

- El programa utiliza la biblioteca <iostream> para manejar la entrada y salida de datos.
- using namespace std; permite usar funciones como cout y cin sin necesidad de prefijarlas con std::.

2. Función Principal (main):

- El programa comienza con la función main(), que es el punto de entrada de cualquier programa en C++.

3. Entrada de Datos:

- Se declara una variable t de tipo float para almacenar la vida media en nanosegundos.
- El programa solicita al usuario que ingrese un valor usando cout y lo captura con cin.

4. Conversión de Unidades:

- La vida media en nanosegundos se convierte a segundos dividiendo el valor entre 1×10^9 (es decir, $1e9$).
- El resultado se almacena en la variable tmedio.

5. Evaluación de la Reactividad:

- El programa utiliza una estructura if-else if-else para determinar la categoría de reactividad:
 - Alta Reactividad: Si $tmedio \leq 1$.
 - Reactividad Media: Si $1 < tmedio \leq 86400$.
 - Baja Reactividad: Si $tmedio > 86400$.
- Cada condición se evalúa en orden, y el programa imprime el resultado correspondiente usando cout.

6. Retorno de la Función:

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

7. Contribuciones

Cómo pueden otras personas contribuir al proyecto:

1. Reportar Problemas:

- Si encuentras un error o tienes una sugerencia, abre un issue en el repositorio del proyecto.

2. Nuevas Funcionalidades:

- Si tienes ideas para nuevas funcionalidades, como agregar más categorías de reactividad o mejorar la interfaz de usuario.

8. Licencia

Detalles sobre la licencia bajo la cual se distribuye el proyecto:

- Este proyecto se distribuye bajo la licencia MIT.
- MIT License es una licencia de software libre que permite el uso, modificación y distribución del código con muy pocas restricciones.
- Puedes ver el texto completo de la licencia en el archivo LICENSE en el repositorio del proyecto.