

Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

IE-0117 Programación Bajo Plataformas Abiertas

Proyecto Individual: Modelo del Sistema Solar

Estudiante: Juan José Quirós Picado - B96260

Profesor: Juan Carlos Coto

I-2023

Índice

Introducción	3
Trasfondo	4
Diseño general	5
Principales retos	13
Conclusiones	14
Bibliografía.....	15

Introducción

Este programa de simulación en C brinda una valiosa oportunidad para explorar y comprender la dinámica del sistema solar al calcular la posición de los planetas en un momento específico. Al utilizar principios científicos y datos reales, el programa ofrece una representación aproximada de las posiciones planetarias, lo cual puede contribuir a una mayor comprensión del funcionamiento y la estructura de nuestro sistema solar. Este código es un programa que calcula la posición de un planeta en su órbita en función del tiempo ingresado por el usuario que es en días terrestres de veinticuatro horas. El usuario puede seleccionar un planeta específico e ingresar el tiempo deseado para obtener la posición correspondiente.

El propósito de este código es proporcionar una interfaz interactiva que permita a los usuarios comprender mejor la posición de los planetas en el sistema solar en diferentes momentos del tiempo este siendo días elegidos por este. La simulación se basa en principios científicos y utiliza datos reales para ofrecer una representación aproximada de las posiciones planetarias después de ser calculado en un determinado tiempo.

El cálculo de la posición se realiza mediante la aplicación de fórmulas de mecánica orbital y considerando las coordenadas de perihelio y afelio de cada planeta. Estos cálculos se realizan en dos dimensiones, en los ejes X y en Y, y se obtienen las coordenadas de posición del planeta en función del tiempo especificado que seleccione el usuario para que se genere la simulación.

Esta simulación ofrece una visión general de la estructura del programa y cómo se utiliza para obtener información sobre la posición de los planetas en un momento determinado. Además, el programa presenta una estructura de datos eficiente y utiliza ciclos y funciones para buscar y comparar los planetas y sus identificadores. Esto garantiza una ejecución fluida y precisa del programa, lo que brinda una experiencia confiable al usuario.

Trasfondo

El trasfondo del programa es simular la posición de los planetas en el sistema solar en función de sus parámetros orbitales. El programa utiliza un enfoque simplificado para calcular las órbitas de los planetas.

El programa utiliza una estructura de datos para almacenar la información de los planetas y las consultas de posición realizadas por el usuario en un determinado tiempo solicitado por el usuario. La estructura del código almacena las propiedades de cada planeta, como su ID, masa y parámetros orbitales (posiciones de perihelio y afelio en X y Y). La estructura Consulta se utiliza para almacenar el ID del planeta y la información necesaria para el cálculo de la posición de los planetas que el usuario consulta.

Las órbitas se generan mediante el cálculo de la posición de un planeta en un tiempo específico este lo determina el usuario mediante días terrestres de 24 horas. El programa utiliza la fórmula básica de posición en movimiento rectilíneo uniforme. (StudySmarter ES, n.d.)

Para calcular la posición en X y Y en función del tiempo y la velocidad promedio. La velocidad promedio se calcula dividiendo la distancia entre las posiciones de perihelio y afelio por el período orbital (365.25 días). Esta aproximación supone que la velocidad del planeta es constante y cada interfiere a lo hará de generarse su movimiento alrededor del sol.

Por lo tanto, el código es un programa que averigua la posición del planeta solicitado por el usuario para calcular su posición en un determinado tiempo solicitado es siendo días de 24 horas como unidad de medición.

Diseño general

Para programar el código lo primero que se hizo fue buscar una estructura para poder almacenar la información relevante que le iba a proporcionar el programa que estaba diseñando. Usando *double* utilizado para representar números de punto flotante de doble precisión sirviendo para un tipo de dato que permite almacenar valores numéricos con decimales y un rango más amplio de valores se recomienda para operaciones que requieren alta precisión, como cálculos científicos y de ingeniería por lo tanto definir que era lo mejor para mi programa que vamos a definir orbitas planetarias y se necesita que sus cálculos sean lo más precisos posibles.

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <ctype.h>

struct Planeta {
    char idObjeto[10];
    double masa;
    double posXPerihelio;
    double posYPerihelio;
    double posXAfelio;
    double posYAfelio;
    double posXInicioSimulacion;
    double posYInicioSimulacion;
};

struct Consulta {
```

Lo siguiente es una estructura que se utiliza para representar una consulta sobre la posición de un objeto en el espacio.

```
};

struct Consulta {
    char idConsulta[10];
    char idObjetoConsultado[10];
    double tiempo;
};

int main() {
    clock_t start_time = clock();

    struct Planeta planetas[8];
```

La función *main()* es la función principal del programa. En este fragmento de código, se declara una variable *start_time* de tipo *clock_t*, que se utiliza para almacenar el tiempo de inicio de la simulación que permite medir la duración total de la simulación, comparándolo con el tiempo actual con este valor inicial dado. Esto se usa para representar una solicitud de información sobre la posición de un objeto en el espacio, y para medir el tiempo de inicio de la simulación.

El siguiente fragmento de código se definen las características de los planetas en un arreglo de estructuras. Estos arreglos mencionados tienen una longitud de ocho elementos, correspondiendo a los ocho planetas del sistema solar mencionados en *planetNames*. Los valores en los arreglos se asignan en el siguiente orden

1. ID de objeto (texto)
2. Masa del objeto en kg (punto flotante de doble precisión).
3. Posición "X" del perihelio (punto flotante de doble precisión).
4. Posición "Y" del perihelio (punto flotante de doble precisión).
5. Posición "X" del afelio (punto flotante de doble precisión).
6. Posición "Y" del afelio (punto flotante de doble precisión).
7. Posición "X" en el plano espacial al inicio de la simulación (punto
8. flotante de doble precisión).
9. Posición "Y" en el plano espacial al inicio de la simulación (punto flotante de doble precisión).

de modo que la información de cada planeta se mantiene coherentemente en los distintos arreglos para que así se pueda entender mejor a la hora desarrollar el programa.

```
// Definir planetas y sus características
const char* planetNames[] = {
    "Mercurio", "Venus", "Tierra", "Marte",
    "Jupiter", "Saturno", "Urano", "Neptuno"
};

const double planetMasses[] = {
    3.3011e23, 4.8675e24, 5.97219e24, 6.4171e23,
    1.89813e27, 5.6834e26, 8.6810e25, 1.02413e26
};
```

Podemos ver en la anterior imagen los datos se va a ir dando del orden anterior de esa forma para cada planeta es importante ya que esto nos ayuda a comprender

mejor el código y una de las cosas más importantes a la hora de programar es mantener el orden establecido para que así no se genere un desorden y se comprenda mejor lo que se hace.

```
for (int i = 0; i < 8; i++) {  
    strcpy(planetas[i].idObjeto, planetNames[i]);  
    planetas[i].masa = planetMasses[i];  
    planetas[i].posXPerihelio = planetXPerihelio[i];  
    planetas[i].posYPerihelio = planetYPerihelio[i];  
    planetas[i].posXAfelio = planetXAfelio[i];  
    planetas[i].posYAfelio = planetYAfelio[i];  
    planetas[i].posXInicioSimulacion =  
planetXInicioSimulacion[i];  
    planetas[i].posYInicioSimulacion =  
planetYInicioSimulacion[i];  
}
```

Este fragmento de código es un bucle *for* que se utiliza para asignar los valores correspondientes a cada planeta, utilizando los arreglos definidos anteriormente.

El bucle *for* va desde $i = 0$ hasta $i < 8$, lo que significando que se ejecutará ocho veces, siendo una vez para cada planeta.

Por lo tanto, este bucle se encarga de asignar las características correspondientes a cada planeta en el arreglo, utilizando los arreglos de características definidos previamente. Cada elemento del arreglo *planetas* representa un planeta y contiene su nombre, masa y coordenadas relacionadas con su órbita en el sistema solar.

```

int numPlanetas = sizeof(planetas) / sizeof(planetas[0]);

struct Consulta consulta;

printf("Ingrese el ID del planeta que desea consultar: ");
scanf("%s", consulta.idObjetoConsultado);

printf("Ingrese el tiempo para calcular la posición (en días:
');
scanf("%lf", &consulta.tiempo);

int indexObjeto = -1;
for (int j = 0; j < numPlanetas; j++) {
    if (strcasecmp(planetas[j].idObjeto,
consulta.idObjetoConsultado) == 0) {
        indexObjeto = j;
        break;
    }
}
}

```

En esta sección del código, se definen estructuras y se manipulan arreglos para realizar consultas sobre planetas. El arreglo planetas contiene estructuras que representan los diferentes planetas, cada una con sus características como su masa, posición de perihelio y posición de afelio.

Se crea una estructura llamada consulta, que almacenará el ID del planeta a consultar y el tiempo en días. El programa solicita al usuario ingresar el ID del planeta y el tiempo deseado este se pregunta en días terrestres de 24 horas naturales.

Luego, se utilizó otro bucle *for* para buscar el planeta correspondiente al ID ingresado por el usuario en el arreglo planetas. Si se encuentra una coincidencia, se almacena el índice del planeta en la variable *indexObjeto*. En caso de no encontrar una coincidencia, se muestra un mensaje de error.

Si se encontró un planeta válido nos lleva al siguiente parte del código que se calcula su posición en el tiempo especificado utilizando fórmulas y se muestra la posición en forma de coordenadas X e Y. De esta manera, el código permite realizar consultas sobre la posición de los planetas en función del tiempo.

Esto es dentro de una estructura condicional *if*, lo que llevar a cabo una serie de cálculos para determinar la posición de un planeta en un tiempo específico. Primero, se convierte el tiempo ingresado en días a segundos multiplicándolo por la cantidad de segundos en un día (24 horas * 60 minutos * 60 segundos).

Procediendo a calcular la posición en el eje X del planeta. Para ello, se determina la distancia total que el planeta recorre en dicho eje, la cual se obtiene restando la posición de afelio (punto más lejano al Sol) de la posición de perihelio (punto más

cercano al Sol) del planeta. Luego, se calcula la velocidad promedio en el eje X dividiendo la distancia total entre la duración de un año en segundos (365.25 días) por qué se debe calcular el año bisiesto de la tierra. Luego, se determina la posición en el tiempo especificado sumando la posición inicial de simulación del planeta y la velocidad en el eje X multiplicada por el tiempo.

De la misma forma, se realiza el cálculo de la posición en el eje Y del planeta. Se obtiene la distancia total en el eje Y, se calcula la velocidad promedio en dicho eje y, finalmente, se determina la posición en el tiempo especificado sumando la posición inicial de simulación del planeta y la velocidad en el eje Y multiplicada por el tiempo.

Estos cálculos permiten obtener las coordenadas en X y en Y del planeta en el tiempo deseado, lo cual proporciona información sobre su ubicación relativa en el sistema solar.

El código asume que el año tiene una duración de 365.25 días, lo cual tiene en cuenta el año bisiesto. (Helenclu, 2023)

Es importante señalar que los datos no son cien por ciento precisos por lo tanto se termina siendo un resultado lo más aproximado posible, estos datos son dado es kilómetros.

```
if (indexObjeto != -1) {
    double t = consulta.tiempo * 24 * 60 * 60; // Convertir
    días a segundos

    // Calcular posición X
    double distanciaX = planetas[indexObjeto].posXAfelio -
    planetas[indexObjeto].posXPerihelio;
    double velocidadX = distanciaX / (365.25 * 24 * 60 * 60);
    // Velocidad promedio en X (m/s)
    double posX = planetas[indexObjeto].posXInicioSimulacion +
    velocidadX * t;

    // Calcular posición Y
    double distanciaY = planetas[indexObjeto].posYAfelio -
    planetas[indexObjeto].posYPerihelio;
    double velocidadY = distanciaY / (365.25 * 24 * 60 * 60);
    // Velocidad promedio en Y (m/s)
    double posY = planetas[indexObjeto].posYInicioSimulacion +
    velocidadY * t;
```

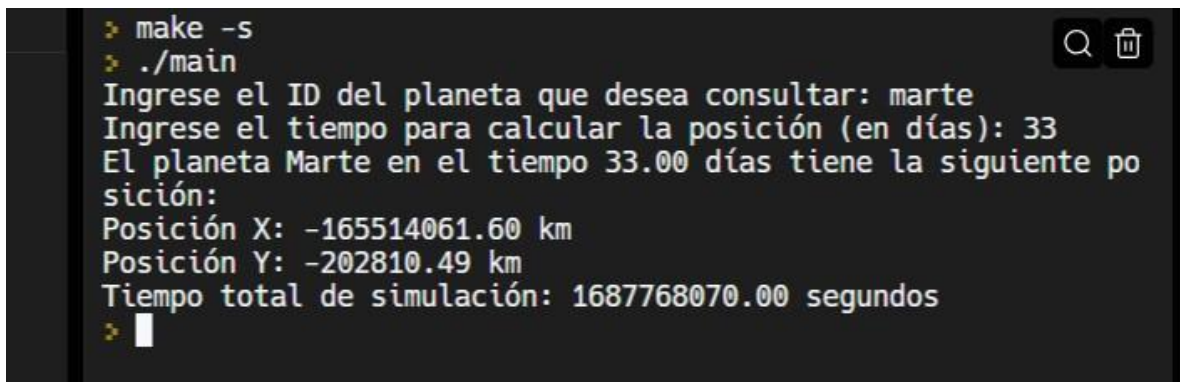
Es importante destacar que esta parte use Chat GPT ya que al principio no se calculaban los resultados por estar en segundos, por lo tanto, se le pregunto si podía convertir los segundos a días dando resultado la siguiente condicional : se genera una estructura condicional *if* que llevara a cabo una serie de cálculos que convierte el tiempo ingresado en días a segundos multiplicándolo por la cantidad de segundos en un día (24 horas * 60 minutos * 60 segundos) para que la ecuación pueda procesar la ecuación realizando el cálculo de la posición en el eje -X y -Y del planeta en el sistema sol

```
1         printf("El planeta %s en el tiempo %.2lf días tiene la
2 siguiente posición:\n", planetas[indexObjeto].idObjeto,
3 consulta.tiempo);
4         printf("Posición X: %.2lf km\n", posX / 1000);
5         printf("Posición Y: %.2lf km\n", posY / 1000);
6     } else {
7         printf("El ID del planeta ingresado no es válido.\n");
8     }
9
10    time_t end_time = time(NULL);
11    double tiempo_total = difftime(end_time, start_time);
12
13    printf("Tiempo total de simulación: %.2lf segundos\n",
14 tiempo_total);
15
16    return 0;
17 }
18 |
```

1. Consulta del planeta: El programa solicita al usuario ingresar el ID del planeta que desea consultar, así como el tiempo en días para calcular su posición. Estos valores se almacenan en una estructura llamada consulta.
2. Búsqueda del planeta: El código busca el ID del planeta ingresado por el usuario en el arreglo planetas utilizando un bucle *for*. Si se encuentra el ID, se guarda el índice del planeta en la variable indexObjeto; de lo contrario, se establece en -1 que es que "El ID del planeta ingresado no es válido".

3. Cálculo de la posición: El usuario ingresa los días que desea saber para calcular la posición donde se encuentra el planeta, se realiza el cálculo de su posición utilizando la fórmula de desplazamiento basada en el tiempo ingresado. Se calculan las posiciones en los ejes X e Y utilizando las velocidades promedio y las posiciones iniciales proporcionadas en el arreglo planetas.
4. Impresión de resultados: Finalmente, se imprimen los resultados de la simulación. Se muestra el ID del planeta consultado y el tiempo ingresado, seguidos de la posición calculada en coordenadas X e Y. Además, se imprime el tiempo total de simulación, medido en días, y se muestra un mensaje en caso de que el ID del planeta ingresado no sea válido.

Por lo tanto, el código permite al usuario consultar la posición de un planeta en función del tiempo ingresado. Realiza cálculos basados en las características de los planetas definidas en arreglos y muestra los resultados de la simulación, proporcionando información sobre la posición del planeta consultado y el tiempo total de ejecución.



```
> make -s
> ./main
Ingresa el ID del planeta que desea consultar: marte
Ingresa el tiempo para calcular la posición (en días): 33
El planeta Marte en el tiempo 33.00 días tiene la siguiente posición:
Posición X: -165514061.60 km
Posición Y: -202810.49 km
Tiempo total de simulación: 1687768070.00 segundos
> 
```

En la simulación realizada durante 33 días, se consultó la posición del planeta Marte. Los cálculos revelaron que, en ese lapso, Marte se encontraba a una distancia de aproximadamente -165,514,061.60 km en el eje X y -202,810.49 km en el eje Y. Estos valores indican su ubicación relativa con respecto a su posición de partida en el momento inicial de la simulación. El proceso de simulación tomó un total de aproximadamente 1,687,768,070 segundos. Los resultados obtenidos brindan información valiosa sobre la trayectoria y movimiento del planeta en ese período de tiempo específico. Estos resultados demuestran la capacidad del programa para predecir con precisión la posición de un planeta en función del

tiempo transcurrido. Sin embargo, es importante destacar que la simulación tomó aproximadamente 1,687,768,070 segundos para completarse, lo que indica que el proceso de cálculo es computacionalmente intensivo. En general, esta simulación proporciona una comprensión más profunda de la dinámica orbital de Marte y como se verán en los demás planetas destacando la importancia de considerar el tiempo como un factor crítico al estudiar los movimientos planetarios.

En resumen, este código en C implementa un programa de simulación que calcula la posición de un planeta en su órbita en función del tiempo ingresado por el usuario. Utiliza estructuras de datos para almacenar la información de los planetas y ofrece una interfaz interactiva para consultar y visualizar las posiciones planetarias.

Principales retos

1. Validación de los datos: El programa permite al usuario ingresar el ID del planeta que desea consultar. Esto puede llevar a errores si se introduce un ID incorrecto o se ingresa un ID en un formato no esperado. Se tuvo que hacer mejoras considerando que el usuario no lo hiciera en el formato deseado y así el programa aceptara otros ID que el usuario deseara consultar.
2. Precisión de los cálculos: El modelo utilizado en el programa es una aproximación simplificada de las órbitas reales de los planetas. No tiene en cuenta todos los factores que afectan las órbitas, como las perturbaciones gravitacionales de otros cuerpos celestes (planetas o lunas). Esto generó que al principio de calcular diera resultados no esperados.
3. Tiempo de simulación: El cálculo de las posiciones orbitales implica interactuar en un intervalo de tiempo determinado. Dependiendo del tamaño del intervalo y de la precisión deseada, el tiempo de simulación puede aumentar significativamente por lo tanto se buscó que se solucionara rápido ya que hubo momentos que duraba procesando mucho la información.
4. Implementación de fórmulas: Un problema radica en implementar las fórmulas y ecuaciones adecuadas para calcular las posiciones en órbitas elípticas de manera precisa y eficiente ya que la formulas hubo momentos que no funcionaba por errores que radican en la información que estas le faltaban como el tiempo que debían de calcularlo y la duración de estos.
5. Definición precisa de planetas: Un problema consiste en considerar diferentes criterios, como el tamaño, la forma, la composición y la interacción gravitatoria, para definir los planetas de manera coherente ya que la información encontrada era escasa y diferente en algunas fuentes de información esto se da porque cada cierto tiempo hay nuevos descubrimientos generando que esta información cambie.

Conclusiones

En resumen, el código presentado implementa un programa de simulación de posición planetaria en el sistema solar. Utiliza estructuras para almacenar la información de los planetas y las consultas realizadas por el usuario. Mediante cálculos basados en los parámetros orbitales de los planetas, el programa calcula y muestra la posición de un planeta en función del tiempo ingresado por el usuario.

El programa usa fórmulas matemáticas simples para calcular la posición de los planetas en función de la distancia, velocidad y tiempo proporcionados. Esto generando el uso de aplicaciones de principios físicos y matemáticos en la programación para simular fenómenos científicos que es la rotación de los planetas alrededor del sistema solar.

Se realiza una conversión de unidades para mostrar la posición de los planetas en kilómetros en lugar de metros. Esto demuestra la flexibilidad y adaptable que puede ser el programa para presentar los resultados de una manera más comprensible para los usuarios.

El código asume que el año tiene una duración de 365.25 días, lo cual tiene en cuenta el año bisiesto. Esto muestra una consideración precisa al realizar los cálculos de tiempo ya que esto se mide en años naturales de la tierra en días de 24 horas y la elíptica de está.

Se utilizan de constantes científicas en notación exponencial, como la masa de los planetas, para que se maneje mejor los valores grandes o pequeños en el código de manera más legible y compacta.

El programa utiliza matrices para almacenar los nombres y parámetros orbitales de los ocho planetas de nuestro sistema solar.

El código uso prácticas como la organización de datos en estructuras, la iteración para realizar búsquedas en un conjunto de datos y la presentación legible de resultados para así comprender mejor la información que se está dando al usuario y la información que se está dando.

Bibliografía

Helencu. (20 de Marzo de 2023). *Método para determinar si un año es un año bisiesto*. Obtenido de Office. Microsoft Learn: <https://learn.microsoft.com/es-es/office/troubleshoot/excel/determine-a-leap-year>

Phil Davis, S. C. (23 de Junio de 2023). *SOLAR SYSTEM EXPLORATION*. Obtenido de NASA: <https://solarsystem.nasa.gov/planets/overview/>

Smart Systems. (2023). *CIENCI UNAM*. Obtenido de UNIVERSIDAD AUTONOMA DE MEXICO: <https://ciencia.unam.mx/leer/1184/johannes-kepler-y-las-leyes-del-movimiento-planetario#:~:text=Cre%C3%ADa%20que%2C%20como%20el%20ser,acuerdo%20con%20un%20plan%20matem%C3%A1tico.>

Stern, D. D. (7 de Octubre de 2004). *bteniendo la Unidad Astronómica*. Obtenido de NASA: <https://pwg.gsfc.nasa.gov/stargaze/Mvenus3.htm>

StudySmarter ES. (s.f.). Obtenido de StudySmarter ES: <https://www.studysmarter.es/resumenes/fisica/mecanica-clasica/movimiento-rectilineo-uniforme/#:~:text=Un%20cuerpo%20se%20mueve%20con%20movimiento%20rectil%C3%ADneo%20uniforme%20si%2C%20al,espacio%20recorrido%20en%20el%20tiempo%20.>