Universidad de Costa Rica

Escuela de Estudios Generales

IE-0117

Proyecto : Modelo del Sistema Solar

Profesor: Juan Carlos Coto

Grupos 003

Estudiante:

Isabella Con Vilela C22277

I Semestre 2023

San José, Costa Rica

Indice:

Introduccion:

El documento presente tiene como finalidad brindar una descripción minuciosa del proyecto final del curso de Programación bajo plataformas abiertas (IE-0117), titulado Modelo del Sistema Solar. En el presente se pretende ilustrar dar seguimiento del proyecto y su alcance, presentar los recursos utilizados y los resultados obtenidos, consigo un análisis de los retos a través del proceso de desarrollo y sus soluciones. Esto con finalidad de informar una clara representación concisa de las partes y elementos utilizados para el proyecto, para así entonces documentar y analizar su avance.

El trabajo se ha realizado con el objetivo de desarrollar un modelo del sistema solar. En el desarrollo de este documento se presentan los principales aspectos del proyecto incluyendo su diseño, partes, trasfondo, y retos, a estos se les añade la respectiva información investigada y trasfondo científico del modelo de Kepler.

Este informe tiene como objetivo servir como nota de referencia para los usuarios o donde se ofrece una visión completa con detalles de los desafíos del proyecto y las actividades y pasos para poder unir las diferentes partes del modelo y obtener los resultados requeridos.

A lo largo de este documento se ofrece una explicación de los logros relevantes y conclusiones que provienen de la experiencia y el conocimiento adquirido durante el desarrollo del proyecto. Así entonces, con este documento se pretende proporcionar una generalidad del Modelo del Sistema Solar para el cálculo de órbitas.

Transfondo:

De manera concisa se puede decir que el proyecto de Modelo del Sistema Solar, está desarrollado con el fin de calcular órbitas alrededor de un foco, con esto calcular la posición de un cuerpo gravitando alrededor de la órbita después de un plazo de tiempo predeterminado.

El programa inicia recibiendo el nombre de un archivo que se abre para ser procesado e interpretado. Este archivo de entrada está dividido en dos partes, una donde contiene los cuerpos, seguidos de sus especificaciones de posición y masa, y una segunda parte donde se delimitan las consultas por un ID y el dato predeterminado desde inicio de la simulación.

Al procesar los datos para interpretación, la primera parte de los datos de los cuerpos, se procesan separando los datos y se interpretan para ser asignados como propiedades de los objetos. En la segunda parte, de delimitación de consultas se procesan los datos para interpretación e identificar con cuál de los cuerpos y sus datos usar como referencia, además se debe de interpretar el dato de tiempo que se brinda en esta sección por este dato pre delimitado es necesario para el uso de las fórmulas para el cálculo de la posición final del objeto en la órbita.

Ahora bien, al procesar separando los datos, e interpretar ubicando los datos como propiedades donde deberían de ser utilizados, seguimos con el cálculo de la posición de las órbitas. En esta parte es donde se debe de introducir el trasfondo científico del programa.

Para la parte mencionada anteriormente se utiliza como base el modelo de Kepler. El modelo de Kepler hace referencia al estudio de un Astrónomo aleman Johannes Kepler en el siglo XVII donde establece las leyes del movimiento planetario que hacen referencia a la descripción del movimiento de los planetas alrededor del sol.

La primera ley de Kepler, se establece como la Ley de las órbitas. En esta se ilustra el movimiento de los planetas alrededor del Sol en órbitas con forma de elipse.

En la segunda ley de Kepler titulada la Ley de áreas, establece que hay una línea que hace conexión entre el planeta en la órbita y el foco de la elipse que es el solo, donde esta barre áreas iguales en tiempos iguales. Lo dicho anteriormente describe que los cuerpos en una

órbita se mueven más rápido cuando están más cerca del sol y más lento cuando están más alejados.

La tercera y última ley de Kepler, la ley de los periodos aclara que en el cuadrado del periodo orbital de un cuerpo en una órbita es proporcional al cubo de la longitud del semieje mayor de su misma órbita. Esto hace referencia a que existe una relación entre el periodo orbital (tiempo que se tarda el cuerpo en completar una órbita alrededor del sol) y la distancia promedio al sol.

Para poder entender este modelo y las fórmulas necesarias, además se debe entender dos términos importantes, el perihelio y el afelio. Estos son términos utilizados en el área de astronomía, que sirven para describir la posición de un objeto en su órbita. El perihelio hace referencia a un cuerpo en la órbita cuando está en su posición más cercana posible al sol, es decir que es el punto donde hay máxima aproximación al sol en la trayectoria predeterminada orbital, donde además es donde este cuerpo está a su mayor velocidad. En cuanto al afelio, se podría decir que es lo opuesto, es la ubicación de la órbita donde el objeto está más lejano del sol, entonces es en la posición donde su velocidad es más baja.

El archivo de entrada brinda las coordenadas cartesianas (x,y) del perihelio y el afelio de la órbita del cuerpo que se está evaluando, y además las coordenadas (x,y) también cartesianas del objeto al inicio de la simulación. Se establece en este programa predeterminadamente, el foco donde se supone el sol o el cuerpo alrededor del cual se órbita como si estuviera posicionado en el centro del plano cartesiano, el origen (0,0), y la elipse como una horizontal.

Ahora bien, con los datos de la posición del afelio y el perihelio se debe de calcular la distancia desde el origen (0,0), hasta estos puntos que se podrían además considerar los vértices de la elipse horizontal. Gracias a la predeterminación del foco en el origen, se sabe que el afelio y el perihelio deben de tener coordenadas en formato de (x,0), por lo que es fácil determinar las distancias del foco a estos ya que es el valor absoluto de x. La suma del valor absoluto de sus valores de x corresponden al eje mayor, y la mitad de este el semieje mayor. El valor del eje mayor se puede denominar **a**, donde el semieje sería a/2.

Para encontrar el valor del centro “c” de la elipse se debe utilizar el valor del semieje Mayor y dividirlo entre dos, sin embargo además se le debe de restar el valor absoluto de la distancia que se encuentre en el lado negativo, o bien al lado izquierdo del foco, para encontrar el valor x

de las coordenadas del centro, donde x=c. Con estos valores se debe encontrar el valor de la excentricidad describe que tan alargada o achatada se encuentra una elipse, por lo que una elipse con excentricidad de 0 describe un círculo, esta se calcula con la fórmula $e = c/(a/2)$, a pesar que no es necesaria para el cálculo del punto que estamos buscando, este da una idea de la forma de la elipse.

Es esencial buscar el valor del eje menor, y semieje menor, representado como **"b"**. Se puede despejar la b de la fórmula $C^2 = a^2 - b^2$. por lo que $b = \sqrt{a^2 - c^2}$. En este caso la b es el semieje menor y el eje menor es 2b.

Por otro lado, es difícil calcular el periodo orbital, ya que este por lo elaborado anteriormente respecto a los cambios de la velocidad de un cuerpo en órbita. Este cálculo dependen valores de masas y de gravedad, donde la G es la constante de gravedad 6.67×10^{-11}, además de la A. El periodo orbital se denomina con una **"T"**, $T = 2\pi\sqrt{A^3/(G \times M)}$. Esto donde Au es una unidad astronómica, para este elegimos ser constante como Au=1, el cual corresponde al planeta tierra.

Si encontramos el periodo orbital de este cuerpo podemos saber la duración de recorrido. En el archivo de entrada en la parte de consulta hay un dato de tiempo, con este dato podemos visualizar el porcentaje de cuanto de la órbita se tuvo que haber movido. Para poder calcular entonces el Ángulo de recorrido, calculamos (segundos/T) el cual corresponde al porcentaje que se ha movido para sacar el angulo que ha recorrido este $(segundos/T) * 360$ nos va a dar el ángulo que se ha movido, sin embargo hay que añadir el ángulo que hay desde 0 hasta la posición inicial, para este ocupamos transformar las coordenadas del punto inicial (x,y) a coordenadas polares (r, θ). Por lo que $r = \sqrt{x^2 + y^2}$ y ya que $rcos(\theta) = x$ podemos hacer un despeje para dejar $\theta = arcos(x/r)$. De esta manera el $angulo\ recorrido\ \theta = ángulo\ del\ punto\ final.$

Finalmente, es esencial tener presente las fórmulas paramétricas de una elipse. Mientras que la fórmula cartesiana de esta es $\frac{(x-x0)}{a^2} + \frac{(y-y0)}{b^2} = 1$. Sin embargo, las coordenadas paramétricas están establecidas de tal manera en las que $x - x0 = semieje\ mayor \times cos(\theta)$ y $y - y0 = semieje\ menor \times sen(\theta)$, donde (x0,y0) representan el punto del centro de la elipse, con esto dicho y0=0 siempre, ya que en este modelo solar está predefinido el foco en

(0,0) y es una elipse horizontal. Así entonces al despejar estas fórmulas podemos dejar establecido que las coordenadas del punto final están calculadas por la siguiente fórmula. $x = ((a/2) \times cos(angulo\ del\ punto\ final) + x0)$
$y = ((b/2) \times sen(ángulo\ del\ punto\ final) + y0)$ o de igual manera, podríamos decir que el punto final se encuentra en las coordenadas cartesianas: ($([a/2) \times cos(angulo\ del\ punto\ final) + x0],[(b/2) \times sen(ángulo\ del\ punto\ final) + y0]$).

Diseño general:

El modelo de sistemas solar, es un modelo programado en lenguaje “C”, capaz de simular un sistema solar a partir de un archivo de entrada, de esta forma es capaz de interpretar y procesar los datos de planetas y las consultas que se le solicitan. De esta manera el programa debe calcular la posición final de un cuerpo en órbita elíptica alrededor de un centro, en este caso el sol, posicionado en un foco de la elipse, con predeterminadas coordenadas centradas en el origen del plano cartesiano, coordenadas (0,0).

El programa del proyecto está compuesto por un file solo, 2 folders y 2 documentos de texto. El file aislado “main.c”, un folder de clases, donde se definen las consultas, las coordenadas, los objetos, y las respuestas. Otro folder de útils donde se separan la entrada y las fórmulas necesarias para hacer los cálculos matemáticos. El archivo de entrada, que es un archivo de texto, “configuración”, que brinda la información necesaria para hacer los cálculos. El segundo documento de texto, que se crea dentro del programa de respuestas donde se escriben las respuestas a las consultas.

Iniciando con el archivo “main.c”. El archivo de main tiene pocas líneas de código, sin embargo en este archivo es donde se inicializa el programa. Seguido de inicializar el código, en este bloque se le pide al usuario ingresar el nombre del archivo de entrada que desea utilizar como archivo de configuración. Esto permite que el usuario utilice el mismo programa en diferentes archivos de configuración de manera eficiente, para poder personalizarlo. El nombre de este archivo ingresado se utiliza en una función “abrir_archivo” definida en el documento de “entrada.c”.

Para entender el código de este modelo de sistema solar, se deben de describir las estructuras definidas. En el archivo “classes”, se definen las 4 estructuras necesarias, ”coordenadas”,”objeto”,“consulta”, ”respuesta”. La estructura de coordenadas es simple, esta fue creada con el propósito de almacenar los dos datos que componen una coordenada (x,y) y estos como sus propiedades. “Objeto” es una estructura que describe un cuerpo o un planeta en la órbita, este tiene 8 propiedades, el ID, la masa, coordenadas (x,y) del perihelio, coordenadas (x,y) del afelio, coordenadas (x,y) del cuerpo al inicio de la simulación. La estructura de consulta, está definida con 3 propiedades, el id_consulta, el id_objeto, y los segundos que está solicitando después del inicio de la simulación. La estructura “Respuesta”, está compuesta por el ID de la respuesta que debe estar acorde al ID de la consulta, y las coordenadas cartesianas (x,y) que responde a la consulta.

El documento de entrada, o documento de configuración está compuesto por dos partes, descripción y consultas. Este inicia con un encabezado “[descripción]”, seguido por cuerpos o planetas en cada línea, y los datos el ID, la masa, coordenadas (x,y) del perihelio, coordenadas (x,y) del afelio, coordenadas (x,y) del cuerpo al inicio de la simulación separados por comas. Luego un encabezado “[consultas]” y una consulta por línea con los datos id_consulta, el id_objeto, y los segundos que está solicitando después del inicio de la simulación separados por una coma “,”.

En el documento de “entrada.c” se guardan la mayoría de funciones. Se inicia pre definiendo un arreglo para almacenar cada estructura, lo que refiere a crear un arreglo para objetos, consultas, y respuestas. Se debe iniciar por explicar la función de “abrir_archivo” donde se define una línea y un contador para los objetos y las consultas por separado. Se abre el archivo y se almacena la cabecera en un arreglo “seccion_actual” con un tamaño predefinido para no tener que usar malloc y calloc. Se usa un ciclo while para pasar de línea a línea del archivo. Para interpretar en qué sección del archivo está, se analiza el primer char de la línea. Si el primer char de la línea es ‘[’ significa que está posicionado en una cabecera, luego si el primero es ‘d’

está en la sección de descripción, mientras que si está en ‘c’ está en la sección de consulta.

Luego la función de “Interpretador_objeto”, como su nombre interpreta cada objeto para separar sus datos. Para eso utilizamos una función “strtok”, la cual separa cada línea en datos separados por “, ” en este caso. Se utiliza un ciclo while y condicional if con un contador, para interpretar los datos y procesarlos para poder ingresar cada uno de ellos dentro de las propiedades de una estructura de “objeto”, gracias a que los datos están organizados en las líneas del documento de configuración en un orden específico. Después se cierra el archivo y se llama la función “generar_salida”.

El interpretador de consultas está definido en una función “interpretador_consultas”. Esta funciona exactamente igual que la función interpretadora de consultas pero es llamada cuando la función de “abrir_archivos” identifica que se encuentra en la sección de consultas. Esta de igual manera separa los datos y los identifica en las propiedades de la estructura creada “consulta”.

Es importante definir la función de “procesador_respuestas”. En esta función se generan dos ciclos “for” anidados, esto para que se responda cada consulta con respecto a su cuerpo respectivo. Luego se introducen las funciones que contienen las fórmulas matemáticas para el cálculo de las coordenadas finales. Se calculan los radios del afelio y perihelio con la función “calcularDistancia”, luego se calcula el eje mayo con la suma del radio del afelio y el perihelio. Luego se calcula el centro “c” con la función calcular centro el cual se calcula a la mitad del eje mayor, sin embargo para obtener su coordenada de x se debe restar el valor que esté en la parte negativa del plano cartesiano y esto depende de si el foco de la elipse alrededor del cual se está orbitando es el lado izquierdo o derecho. Con el “c” centro, se calcula el eje menor con la función “calcularEjeMenor” la cual hace referencia a la fórmula $c^2 = a^2 - b^2$. Con los datos del semieje Mayor, la masa del objeto y las constantes necesarias, se calcula el periodo orbital en la función “calcularPeriodoOrbital” . Con este se calcula la fracción de movimiento que ha experimentado el cuerpo para poder calcular el ángulo de la

posición final en la función de “calcularAnguloRecorrido”. Finalmente a base de las ecuaciones parametricas de una elipse, se logra calcular la posición final de un punto, con respecto al ángulo y al centro de la elipse, y esto se hace en la función de “calcularCoordenadasFinales”, donde este resultado se guarda como una estructura “Coordenadas”, en un parámetro de respuesta.

Finalmente en el documento de entrada se utiliza una función llamada “generar salida”. En esta función se crea un archivo nuevo, archivo “respuestas” en el que se puede escribir. Luego con un ciclo “for” se recorren las consultas y sus respuestas asociadas para así imprimir su ID y las coordenadas de respuesta final. Para finalizar se cierra el archivo.

Principales Retos:

1. **Separación de los datos de entrada y formato de entrada:**

   Recibir el archivo de entrada y abrirlo no fue un gran reto, sin embargo el separar la información para procesar los datos de la entrada en el formato dado, y así interpretarlos como propiedades dentro de los objetos que se evalúan después. Para resolverlo cree un ciclo while que se encarga de detectar en cuál sección del documento está: Descripción o Consultas. Posteriormente, se ingresa a otro ciclo while se evalúa que hayan datos, a su vez se utiliza la función strtok, para separar un dato de la línea que está siendo leída al llegar a una coma, y luego evaluarlos por medio de un condicional if según su posición en la línea para almacenarlos como propiedades de un objeto "Objeto".

2. **Almacenamiento de los objetos, las consultas y las respuestas:**

   Como en este caso se está utilizando el lenguaje de programación C y el mismo no cuenta con bibliotecas de vectores, listas, o colas almacenar los objetos, la consultas, y las respuestas presentó una complicación importante. Para solucionarlo tuve que utilizar arreglos estáticos en memoria, estos arreglos tienen un tamaño predeterminado. De esta manera no se debe utilizar malloc y calloc.

3. **Transcribir las fórmulas del modelo de kepler a programación:**

   Para poder usar simbología matemática, y trigonometría, se debe importar una biblioteca de matemática para "C" <math.h>. A pesar de entender la lógica matemática de lo que debería de pasar, pasarlo a código o ciclos es complicado. A Veces hacer un ejemplo de la matemática a mano ayuda a aclarar la manera en la que debería de estar programada la función.

4. **Fórmulas matemáticas:**

   El modelo de kepler como explicado en la sección del trasfondo compone 3 reglas, sin embargo en ninguna de estas reglas realmente está especificado

como sacar lo que el programa necesita calcular. Para esto se tuvo que llevar a cabo una extensa investigación, para esto se tuvo que hacer un profundo análisis del formato de las coordenadas y una cascada de fórmulas donde hay dependencia de la final de las demás para usar los datos que se devuelven. Para esto hubo mucha prueba y error con diferentes posibles casos, donde algunas fórmulas no estaban claras, o se estaban usando datos que estaban siendo mal interpretados. Para esto utilice prints en varias secciones del código para poder ver donde estaba el error, cuáles resultados si tenían sentido y cuáles no.

5. **Contador como puntero:**

A lo largo del proceso de desarrollo de mi proyecto tuve un problema persistente debido a que los datos se estaban solapando en memoria, la razón de este error fue que estaba “pasando” una variable “contador” de manera desreferenciada. Por lo tanto al incrementar el contador, éste se incrementó solamente dentro de la función, por ende al utilizarlo seguía en el valor inicial. Para solucionarlo opte por “pasar” el parámetro por medio de un puntero, y realizar los cambios pertinentes para que no hubiera problemas.

Conclusiones:

Finalmente, luego de todo lo expuesto podría decir que el hecho de que el problema del proyecto que está planteado de manera ambiguo permite abrir la interpretación del programador, promueve la investigación del tema que rodea el proyecto, y amplía mucho la manera en la que se programa el modelo requerido.

El proyecto final de programación ha sido una experiencia enriquecedora, esta me ha permitido practicar, y desarrollar mis habilidades y conocimientos de este lenguaje que para mi es nuevo. De la misma forma podría decir que me ayudó a encontrar cierta comodidad programando en este lenguaje. A lo largo del desarrollo de este programa he aprendido muchas cosas, desde nuevas funciones y maneras de resolver problemas en este lenguaje de programación, hasta contenido científico de física y astronomía.

Definitivamente mi dominio del lenguaje de programación “C” definitivamente se ha profundizado, he logrado alcanzar una comprensión más sólida de las funciones, los bucles, las estructuras de control, esto me ha permitido investigar y hacker prueba y error de diferentes versiones del código.

Este proyecto definitivamente ha retado mi capacidad de resolver problemas complejos pacientemente, sin embargo me ha enseñado separar los problemas en partes, para buscar soluciones eficientes, a usar estructuras y datos adecuados también. Esta habilidad de descomponer y romper los problemas en partes más pequeñas, fue esencial para poder solucionar el problema planteado y finalizar el proyecto.

Además aprendí y he llegado a comprender lo fundamental que es mantener el orden y un diseño constante y uniforme en el código. La separación de archivos, documentos, folders es esencial para poder facilitar la lección del programador al momento de resolver y debug, igualmente del lector para comprension. Definitivamente ha mejorado la legibilidad y mi práctica de mantener una buena y limpia estructura del código.

En síntesis, el proyecto final del curso de Programación bajo plataformas abiertas, ha sido una buena experiencia, que sin duda alguna ha permitido aclarar muchas dudas del lenguaje, y fomentado la paciencia para resolver problemas difíciles, mi dominio y la fluidez del uso de este lenguaje de programación definitivamente han mejorado exponencialmente.

Referencias:

1. Ley de Newton de la gravitación universal. (s.f.). En Wikipedia. Recuperado el 26 de junio de 2023, de https://es.wikipedia.org/wiki/Ley_de_Newton_de_la_gravitaci%C3%B3n_universal
2. Leyes de Kepler. (s.f.). En Enciclopedia Britannica. Recuperado el 26 de junio de 2023, de https://www.britannica.com/science/Keplers-laws-of-planetary-motion
3. Biblioteca estándar de C. (s.f.). En GNU Project. Recuperado el 26 de junio de 2023, de https://www.gnu.org/software/libc/