Evaluación de la unidad 2 de teoría de la programación

Integrante 1: Anthony Yaguana

Correo: anthony.yaguana@unl.edu.ec

Integrante 2: Narcisa Pinzón

Correo: narcisa.pinzon@unl.edu.ec

Integrante 3: Camila Chimbo

Correo: camila.chimbo@unl.edu.ec

Integrante 4: Santiago Villamagua

Correo: Santiago.villamagua@unl.edu.ec

Integrante 6: Darwin Correa

Correo: darwin.correa@unl.edu.ec

Definición del problema:

Debido a que en el presente periodo académico se nos ha asignado crear una batería que se cargue con energía solar hemos decidido utilizar un panel fotovoltaico, pero, al momento de querer implementarlo nos damos cuenta de que el sol varía su ubicación constantemente y esto se debe al lugar en que lo coloquemos o debido a que el sol se mueve conforme transcurre el día , entonces este panel solar constantemente debe ajustar su posición automáticamente para así optimizar la captación de luz solar a lo largo del transcurso del día, de esta manera conseguiremos que nuestro cargador solar maximice su eficiencia.

Objetivo del algoritmo:

Crear un algoritmo que nos permita calcular la altitud y orientación solar en la que se debe colocar el panel solar utilizando los siguientes datos: altitud y latitud de nuestra posición actual, la fecha y hora actual del sistema.

Análisis de variables implementadas en pseudocódigo

Para el presente algoritmo se han identificado varias variables que se utilizaran en el transcurso en el que se desarrolla el algoritmo.

Variables principales:

-Lati_tud y Longi_tud: Las cuales son fáciles de obtener, simplemente hay que dirigirse a Google Maps ya sea desde una computadora o un celular y pinchar en la ubicación que se haya encontrado para que proporcione los datos correspondientes.
-Fecha y Hora actual: Estos datos que cambian constantemente se pueden calcular de una manera automática, solo se debe leer la fecha y hora del sistema para usarlas de forma conveniente.
Dentro de la fecha actual se tienen en cuenta variables como: dia, mes y anio.
Y en la hora actual sólo se utilizaron dos variables para guardar la hora actual y los minutos actuales de estas variables: hora y minutos.

Variables requeridas a lo largo del algoritmo: Estas variables se utilizaron para guardar los datos obtenidos de los diferentes cálculos realizados en el algoritmo.
-N: Guarda el número de días que han transcurrido desde que inició el año hasta el día actual.
-Declinacion: Guarda el valor calculado con la ecuación de la declinación solar por lo tanto es la declinación solar.
-EoT: Aquí se almacena el valor del tiempo solar verdadero calculado con la ecuación del tiempo
-TSV: Guarda el valor del tiempo solar verdadero calculado con su fórmula respectiva.
-hora_solar: Guarda solo la hora del TSV.
-minutos_solar: Aquí se almacena los minutos respectivos del TSV.
-H: Guarda el valor del ángulo horario calculado también con una fórmula.
-AS: Esta variable guarda el valor final de la altura del sol en radianes.
-AS_Grados: Es la variable que almacena la altura final del sol en grados.
-Azimut: Aquí se almacena la orientación solar en radianes.
-Azimut_Grados: Almacena el valor de la orientación solar en grados.

Anexo 1

Explicación del pseudocódigo del algoritmo: Este es el funcionamiento general del algoritmo para calcular la altitud solar y orientación solar en la que debe colocarse el panel fotovoltaico.
Primero se obtienen los datos necesarios para hacer los respectivos cálculos y así obtener los ángulos necesarios, primero se tienen en cuenta los datos de longitud y altitud, después la fecha actual y hora actual del sistema, separándolas en sus respectivas variables.
Con estos datos iniciales ya obtenidos se procede a hacer los cálculos con apoyó de las funciones antes ya definidas que se explicarán a continuación del documento.
Cabe recalcar que la forma en que se obtiene cada dato es secuencial, es decir, calculamos un dato necesario para después usarlo en otra ecuación y así obtener los datos necesarios.
Entonces los pasos del algoritmo son:

1.Definir la longitud y latitud del lugar en que se encuentra.

2.Recuperación de la fecha y hora actual del sistema.

3. Calcular el número de días que han transcurrido (N) desde que inició el año hasta el día actual usando las variables de mes, dia y anio.

4. Con los días totales calculados se procede a hacer el respectivo cálculo de la declinación solar (Declinación).

5. Hacer uso también de los días totales transcurridos (N) para proceder a resolver la ecuación del tiempo (EoT).

6. Calcular el tiempo solar verdadero (TSV), utilizando la hora actual (hora), los minutos actuales (minutos), la longitud de nuestra ubicación (Longi_tud) y el resultado de la ecuación del tiempo (EoT).

7. Hacer el cálculo de la hora solar (hora_solar) y los minutos solares (minutos_solar), para la hora solar utilizando la parte entera del tiempo solar verdadero (TSV) y para los minutos solares.

8. Hacer el cálculo del ángulo horario (H), con ayuda del resultado del tiempo solar verdadero (TSV).

9. Acontecer al cálculo de la declinación solar (Declinación) y la latitud (Lati_tud) a radianes usando la fórmula general de conversión a radianes que sería: variable * (PI/180).

10. Con todos los datos obtenidos anteriormente se procede a la realización de cada uno de los dos datos finales de nuestro algoritmo, la altitud solar (AS), que se denota como los grados de inclinación que debe tener nuestro panel solar obtenido con la declinación solar (Declinación), la latitud (Latitud) y el ángulo horario (H). Cabe recalcar que todos los datos usados para calcular la altitud solar deben estar en radianes.
11. Finalmente se realiza la obtención de la orientación solar (Azimut) que son los grados en que debe estar dirigido el panel fotovoltaico con respecto al norte, para calcular la orientación solar (Azimut) usamos los datos respectivos de la declinación solar (Declinación), la altitud solar (AS), la latitud con respecto a la ubicación actual (Latitud) y el ángulo horario (H). En este caso se utilizó todos los ángulos en radianes. **Anexo 2.**

Funciones definidas en el algoritmo

Función Calcular_numero_dias (N): En esta función se pasan como parámetros el mes actual (mes), el día actual(dia) y el año actual (anio), tomando un bucle "Para" y un "Según" para hacer una iteración desde el mes 1 que sería enero hasta un mes anterior al actual esto debido a que el mes actual no está totalmente cursado para determinar el número de mes y por cada número de mes se suma la cantidad de días que tiene cada mes almacenando esta cantidad de días en la variable (N).
Al finalizar se realiza un análisis para verificar si el año actual es un año bisiesto, de ser así se suma un 1 dia a la cantidad de días calculado anteriormente ya que los años bisiestos cuentan con un dia mas que los años normales.
Anexo 3

Función Calcular_angulo_declinación (Declinacion): Esta función recibe como parámetro el número de días (N) que han transcurrido desde que inició el año en el mes de enero hasta el día actual mediante la formula:

$$\delta = -23.45^\circ \times \cos\left(\frac{360}{365} \times (d + 10)\right)$$

A esta fórmula se le complementan ciertos parámetros que se manifiestan de la siguiente manera:

Término _1: Este valor se debe a que es el ángulo de inclinación máximo de la tierra respecto al plano de su órbita alrededor del sol.

Término_2: Se establece una división de 360° entre el número de días del año multiplicado al número de días que han pasado más 10, para compensar cierto tiempo que se pudo haber perdido
Finalmente, al realizar estos cálculos se obtiene el ángulo de la declinación solar para el día N.
Anexo 4
Función Calcular_ecuacion_tiempo (EoT): Para llevar a cabo este cálculo se tiene como referencia la Ecuación del Tiempo (EoT) en minutos denotada como una ecuación que corrige la excentricidad de la órbita de la tierra y la inclinación del eje de la tierra donde:

$$EoT = 9.87 \cdot \sin(2B) - 7.53 \cdot \cos(B) - 1.5 \cdot \sin(B)$$

[imagen 1]
Esta fórmula se utiliza para calcular el ángulo del horario del sol restando al día 81 para ajustar el día del año para que se centre alrededor del solsticio de invierno.

```
B = 360 * (N - 81) / 365
```

[imagen 2]
Luego se convierte el ángulo del horario del sol de grados a radianes multiplicando (B * PI/180) y finalmente se utiliza el ángulo calculado para determinar la Ecuación del tiempo
Anexo 5
Función Calcular_TSV (TSV): Esta función se encarga de calcular la hora solar verdadera (TSV) en función de la hora local, longitud, zona horaria y ecuación del tiempo (EoT).

$$TSV = \text{Hora local} + \frac{4 \cdot (\text{Longitud local} - \text{Longitud estándar}) + EoT}{60}$$

[imagen 3]
Anexo 6
Función Calcular_angulo_horario (H): Pasamos como parámetros el tiempo solar verdadero (TSV) calculado en la anterior función, para calcular este ángulo horario nos ayudamos de la fórmula:

$$H = 15^\circ \times (TSV - 12)$$

[imagen 4]
Luego convertimos en radianes con la fórmula general de conversión a radianes para así poder trabajar este dato adecuadamente.
Anexo 7
Función Calcular_AS: En esta función se usan como parámetros los valores anteriormente calculados, para hacer uso de la declinación solar (Declinacion), la latitud de nuestra ubicación actual (Lati_tud) y el ángulo horario (H).

Con los parámetros mencionados se procede a utilizar la siguiente fórmula para poder calcular la altura solar (AS):

De esta manera obtenemos la inclinación correspondiente del panel solar (AS).

$$\alpha = \sin^{-1}[\sin\delta \sin\varphi + \cos\delta \cos\varphi \cos(HRA)]$$

[imagen 5]

Función Calcular_Azimut: Con esta función se procede a calcular la orientación (Azimut) que debe tener nuestro panel solar con respecto al norte, pasando los parámetros correspondientes que serían: la declinación solar (Declinacion), la altitud solar (AS), la latitud de nuestra ubicación actual (Lati_tud) y el ángulo horario (H). De la misma manera que en las anteriores funciones, tomando la fórmula correspondiente para calcular el azimut:

$$\text{Azimut} = \cos^{-1}\left(\frac{\sin(\delta) - \sin(\alpha) \cdot \sin(\phi)}{\cos(\alpha) \cdot \cos(\phi)}\right)$$

[imagen 6]

Anexo 8

Implementación del algoritmo empleado en el lenguaje de programación C.

El programa realizado a continuación tiene como objetivo optimizar la orientación de un panel solar para maximizar la captación de energía solar. La captación eficiente de energía solar depende de la orientación precisa del panel respecto al sol en cualquier momento del día y del año.

Inicio del Programa en C.

Definición de bibliotecas y constantes utilizadas en el código:

```c
#include <stdio.h>
#include <math.h>
#include <time.h>
#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif
```

<stdio.h> .- Librería empleada para la entrada y salida de datos.
<math.h>.- Librería utilizada para resolver funciones matemáticas.
<time.h>.- Librería para obtener la fecha y hora en en tiempo real.
#define .- Se implementa para tomar una variable como constante en el programa.

Anexo 9

Inicialización del contador de días a 0 en la variable de (N_Dias).
Se inicializa el **Bucle For** el cual se encarga de:

- Recorrer los meses desde 1 hasta el valor de meses.
- Para cada iteración se usa un switch, para sumar los días en (i).
- Despues de completar las iteraciones se las añade en la variable N_Dias.

Suma del día del mes actual:

- **(N_Dias = N_Dias + dias)** Se suma los días del mes actual a N_Dias.

Dentro del mismo se incorpora un ajuste para saber si el año es o no Bisiesto:

- El primer if verifica si (anios) es divisible para 4.
- El segundo if verifica si (anios) es divisible para 400.
- Esto permitirá saber si el (anio) es Bisiesto y si lo es se sumará un día adicional.

Retorno total:

- **(return N_Dias)**retorna el número total de días.

Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular el ángulo de Declinación incluyendo la variable (int N_Dias).

Esta función calcula la declinación solar para un día específico del año, representado por (**N_Dias)**, considerando el efecto de la inclinación del eje terrestre y la variación anual del ángulo solar.

Anexo 10

Inicialización las variables a usar como double en este caso: **(double Termino1, Termino2, DS_Radianes, Coseno, Declinacion_Solar).**

- **Termino1** se le establece -23.44 el cual representa la máxima declinación solar en grados.
- **Termino2** se calcula con ayuda de una fórmula**(360.0/365.0)*(N_Dias+10)** la cual ayudará a convertir el número de días a una fracción del año a grados y la cual se ajusta sumando 10 días para compensar el desfase entre el inicio del año y el punto en la cual la declinación es 0.

Conversión del **termino2** de grados a radianes:

- La conversión se efectúa de la siguiente manera **(M_PI/180.0)**.

Cálculo del coseno en radianes **(DS_Radianes):**

- Se calcula usando la función cos() la cual proporciona el ángulo correspondiente al día del año en radianes.

Cálculo de la declinación solar:.

- Se calcula multiplicando **(Termino1*coseno del ángulo)** el resultado representa la declinación solar en grados para el día especificado.

Retorno del valor calculado:

- La función devuelve el valor de la declinación solar en grados para (N_Dias).

Función con una variable (double) esta variable maneja decimales, esta función nos ayudará para calcular la Ecuación del tiempo incluyendo la variable (int N_Dias).

- Calcula la ecuación del tiempo teniendo en cuenta las variaciones a la excentricidad de la órbita e inclinación de su eje

Anexo 11

Definición de la función y declaración de variables:

- La función toma como entero **(N_dias).**
- Se declaran dos variables de tipo double **(B y EoT)**.

Cálculo de valor B en grados:

- Se calcula **((360.0/365.0)*(N_Dias-81))** esta convierte el número de días en una fracción del año en grados, restando 81 para ajustar el solsticio de primavera, cuando la ecuación del tiempo es aproximadamente 0.

Conversión de B de grados a radianes:

- Se convierte multiplicando por **(M_PI/180.0)**.

Cálculo de la ecuación del tiempo **(EoT)**:

- Se calcula de la siguiente manera **9.87*sin(2*B)-7.53*cos(B)-1.5*sin(B)**. Esta combina funciones trigonométricas para modelar la variación de la ecuación del tiempo a lo largo del año. Mide la diferencia entre el tiempo solar aparente y el tiempo solar medio, la cual varía debido a la excentricidad de la órbita y la inclinación del eje terrestre.

Retorno del valor calculado:

- La función devuelve el valor de EoT, en minutos para el día del año especificado en N_Dias.

Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular el Tiempo Solar.

Esta función permitirá ajustar la orientación de los paneles solares de manera más precisa y eficiente, siguiendo el movimiento del sol en el cielo a lo largo del día.

Anexo 12

Conversión de Hora Local a Formato Decimal:

- La hora local, se convierte a un valor decimal.

Cálculo de la Longitud Estándar de la Zona Horaria:

- Se calcula multiplicando el valor de la zona horaria por 15 grados. Esto se debe a que cada zona horaria cubre 15 grados de longitud.

Cálculo del Tiempo Solar Verdadero **(TSV)**:

- Se calcula la diferencia entre la longitud geográfica de la ubicación y la longitud de la zona horaria.
- Esta diferencia se la convierte en minutos ya que la Tierra gira 1 grado cada cuatro minutos aproximadamente.
- La ecuación del tiempo se añade a este valor en minutos para ajustar el desfase debido a la órbita elíptica y su inclinación axial.
- El total se convierte a horas y se añade a la hora local en formato decimal y obtener el tiempo solar verdadero.

Retorno del **(TSV):**

- Se obtiene en horas decimales, representando en un reloj solar la ubicación específica y momento dado.

Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular el Ángulo .

Esta función calcula el ángulo horario se utiliza para ajustar la inclinación de los paneles de manera que estén orientados hacia el sol para captar la máxima cantidad de radiación solar.

Anexo 13

Definición y Declaración de variables dentro de la función:

- Toma como entrada **(TSV).**

Cálculo del Ángulo Horario:

- Se calcula de la siguiente manera **(Angulo_Horario = 15*(TSV–12)).** Esto ajustará el ángulo para que sea 0 al mediodía solar y si es positivo o negativo dependiendo si es antes o después del mediodía.

Conversión a Radianes:

- Se convierte de la siguiente manera **(Angulo_Horario*(M_PI/180.0)).** Lo que convertirá el ángulo horario de grados a radianes.

Retorno del Valor Calculado:

- Devolverá el Angulo_Horario en radianes para el **(TSV)** dado.

Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular el Altitud.

Con esta función podremos obtener la altitud solar que se utiliza para calcular la intensidad de la radiación solar incidente en los paneles y ajustar la inclinación de estos paneles para maximizar la captación de energía solar.

Anexo 14

Definición y Declaración de variables dentro de la función:

- Toma como entrada los siguientes parametros, **(Declinacion_Solar, Latitud y Angulo_Horario)** estos deberán estar en radianes.

Cálculo de la Altitud Solar:

- Se usa la función asin() para el cálculo del arcoseno, el cual representa el angulo de elevacion del sol de la siguiente manera:
  Altitud_Solar=asin(sin(Declinacion_Solar)*sin(Latitud)+cos(Declinacion_Solar)*cos(Latitud)*cos(Angulo_Horario).

1. **sin(Declinacion_Solar)*sin(Latitud)** determina el componente vertical.
2. **cos(Declinacion_Solar)*cos(Latitud)*cos(Angulo_Horario)** determina la componente horizontal.

Retorno del valor calculado:

- La función devolverá la Altitud_Solar que es el ángulo de elevación del sol sobre el horizonte en radianes.

Función con una variable (double) esta variable maneja decimales esta función nos ayudará para calcular la Azimut.

En sistemas de seguimiento solar, el azimut solar se utiliza para ajustar continuamente la orientación de los paneles solares a lo largo del día y del año.

Anexo 15

Definición y Declaración de variables dentro de la función.

- La función toma cuatro parametros como entrada: **(Declinacion_Solar, Altitud_Solar, Latitud y Angulo_Horario)** estos parámetros deben estar en radianes.

Cálculo del Azimut Solar:

- Para calcular se usa la función acos() para calcular el arcocoseno, el cual representa el ángulo azimut. De la siguiente manera: **Azimut=acos((sin(Declinacion_Solar)-sin(Altitud_Solar)*sin(Latitud))/(cos(Altitud_Solar)*cos(Latitud))).**

1. **sin(Declinacion_Solar)** componente vertical.
2. **sin(Altitud_Solar)*sin(Latitud)** componente horizontal.
3. **cos(Altitud_Solar)*cos(Latitud)** factor de normalización para convertir las coordenadas a un sistema cartesiano.

Ajuste del Azimut segun el Angulo Horario:

- Si el ángulo horario es positivo se ajusta el azimut agregando **2*M_PI** y restando el valor calculado.

Retorno del valor calculado:

- Devuelve el Azimut en radianes.

Anexo 16

Ingreso de datos
Se declara una variable de tipo doule denominada como Pedir_Datos que recibe una constante de tipo char llamada mensaje y tiene un paso de parámetros por referencia.

Variables definidas
float valor;
Int resultado;
Una vez definidas las variables se hace uso de un bucle while el cual se va a ejecutar siempre que el resultado sea igual a cero, es decir cuando el resultado ingresado sea invalido mientras que si el valor ingresado es correcto la variable resultado va a ser igual a uno y va a retornar el valor ingresado por el usuario.

Anexo 17

Variables utilizadas:

- int horas, minutos, segundos.
- int anios, meses, dias.
- int N_Dias.
- double Longitud, Latitud.
- double Declinacion_Solar, EoT, TSV, Angulo_Horario, Altitud_Solar, Orientacion_Solar, Azimut, Hora_Solar, Minutos_Solar.

Obtención de Fecha y Hora exactas.

Declaración de variable:

time_t tiempoahora.

Nos ayuda a almacenar la Fecha y Hora como un número entero.

Obtencion de Fecha y Hora actuales:

time(&tiempoahora).

Se llama a la función time() de la librería estándar time.h, que obtiene la cantidad de segundos desde el 1 de enero de 1970 y la guarda en la variable tiempoahora.

Conversión a tiempo local.

struct tm *mitiempo = localtime(&tiempoahora);

localtime(&tiempoahora); es una función que toma el valor de time_t almacenado en tiempoahora y lo convierte en una estructura tm que contiene los componentes de fecha y hora (como horas, minutos, segundos, año, mes, día, etc.) en la zona horaria local.

Asignación de componentes:

- horas = mitiempo->tm_hour;
- minutos = mitiempo->tm_min;
- segundos = mitiempo->tm_sec;

Estas son las variables de la estructura tm que representan la hora, minutos y segundos actuales del sistema, respectivamente.

- anios = mitiempo->tm_year + 1900;

Contiene el número de años transcurridos desde 1900. Al sumarle 1900, obtenemos el año actual.

- meses = mitiempo->tm_mon;

Representa el mes actual, dónde enero es 0, febrero es 1, y así sucesivamente.

- dias = mitiempo->tm_mday;

Es el día del mes actual.

En resumen, este código obtiene la fecha y hora actuales del sistema y almacena cada componente (horas, minutos, segundos, año, mes, día) en variables separadas para su posterior uso en el programa.

Anexo 18

Discusión para la implementación del código en un sistema de paneles solares reales.

El software utilizado para realizar el panel solar puede extenderse ampliamente beneficiando a todo aquel que lo necesite y sea un amante de los recursos naturales que ofrece la naturaleza como fuente de energía propiciando al ahorro de la energía eléctrica, además no solo puede un panel solar puede tener múltiples usos, empezando como un cargador solar hasta como un implemento de alumbrado en las viviendas, como aire acondicionado o inclusive como una fuente de energía para duchas de agua caliente, el software puede ser modificado para infinidad de usos y no solo destinarse a uno solo. Pero para lograrlo debemos considerar algunos aspectos como:

- **Interfaz con hardware**: El código proporcionado debe comunicarse eficientemente con los controladores de los paneles solares.
- **Gestión de datos**: Debe manejar los datos de manera eficiente y segura. Esto incluye el almacenamiento local en el sistema de control y la transmisión segura de datos a través de redes.
- **Control y monitoreo en tiempo real**: El sistema debe ser capaz de monitorear constantemente el rendimiento de los paneles, la energía generada, y otros parámetros relevantes.
- **Seguridad y redundancia**: Se deben implementar medidas de seguridad robustas para proteger el sistema contra accesos no autorizados y para garantizar que el sistema pueda manejar situaciones de emergencia de manera adecuada.

- **Optimización de eficiencia**: El código debe ser optimizado para minimizar el consumo de recursos computacionales y energéticos, asegurando que el sistema funcione de manera eficiente tanto en términos de hardware como de energía.

Mejoras futuras:

- **Optimización del sistema de almacenamiento de energía:** Mejorar los algoritmos de gestión de la batería para optimizar la carga y descarga, considerando patrones de consumo y generación.

- **Automatización avanzada:** Implementar lógica de control avanzada para ajustar automáticamente la orientación de los paneles solares y optimizar el seguimiento del sol.
- **Monitoreo remoto y gestión de mantenimiento predictivo:** Utilizar análisis de datos avanzados para detectar anomalías en el rendimiento antes de que causen problemas mayores, permitiendo intervenciones de mantenimiento predictivo.

Anexo 1

DIAGRAMA DE FLUJO GENERAL DEL ALGORITMO:

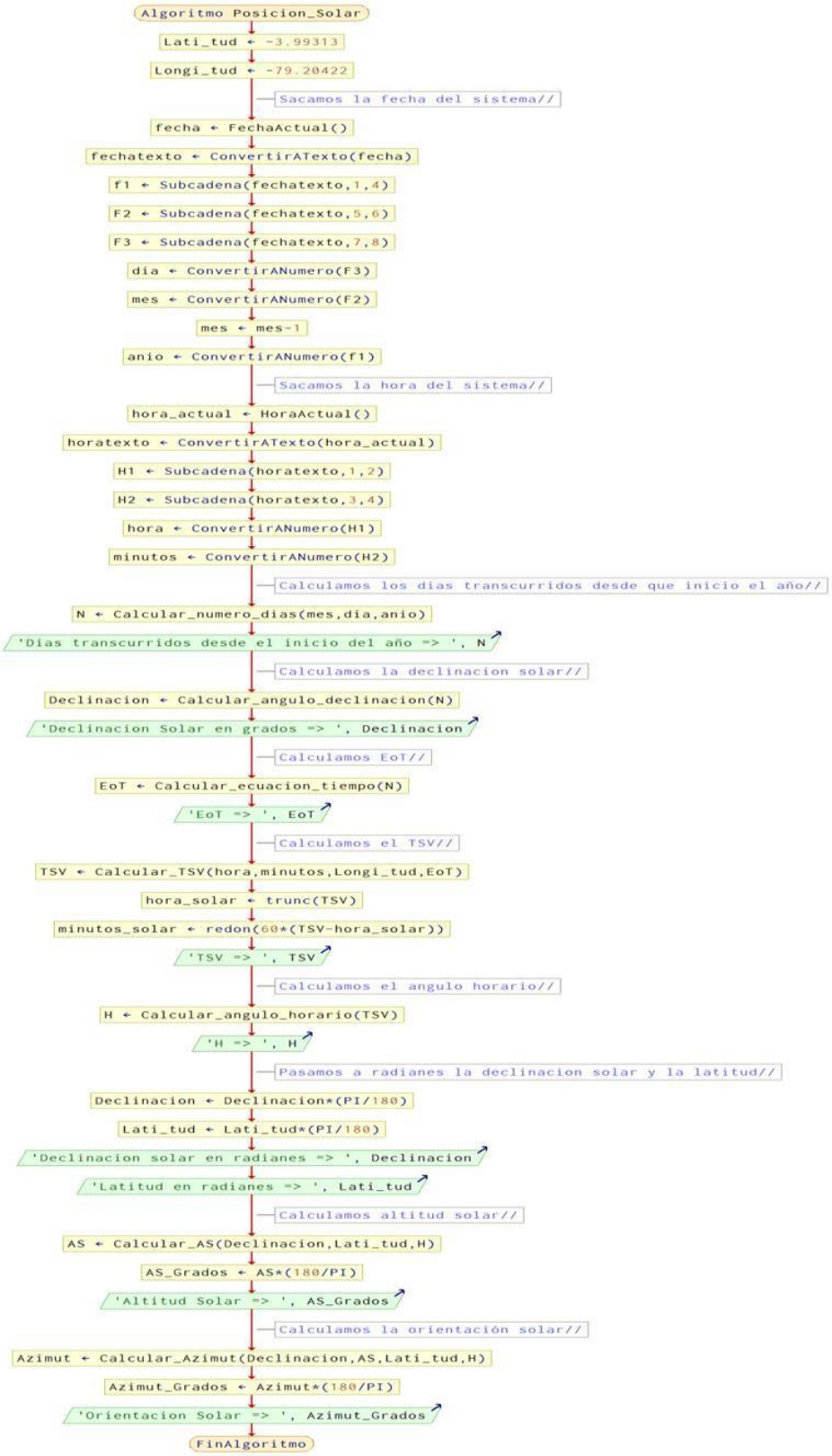


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR LOS DÍAS TRANSCURRIDOS DESDE EL INICIO DEL AÑO HASTA EL DIA ACTUAL:

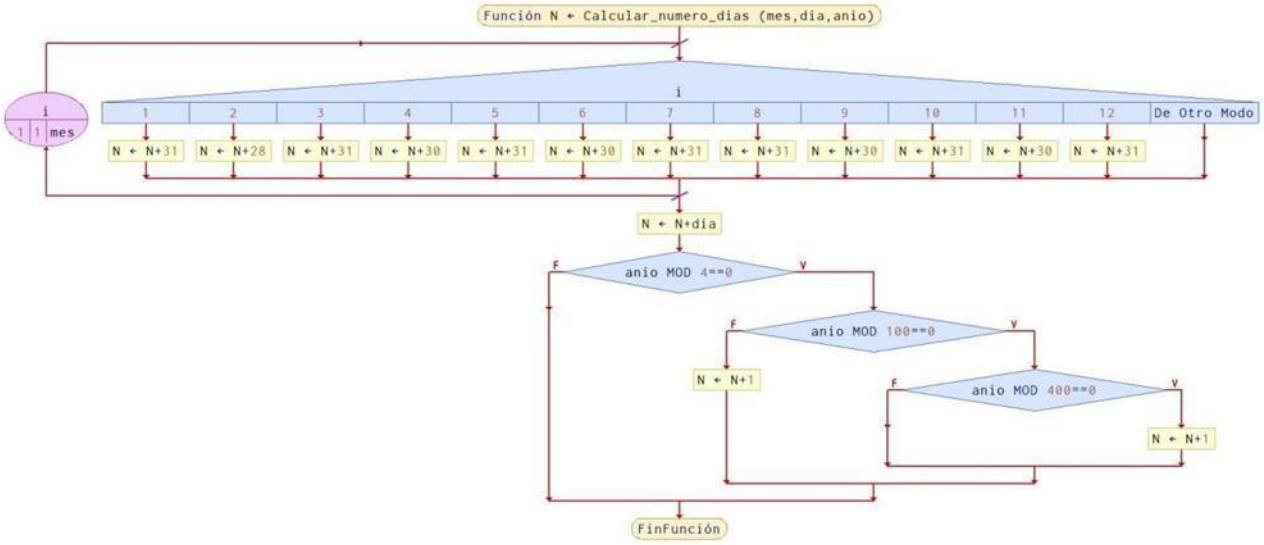


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR EL ÁNGULO DE DECLINACIÓN:

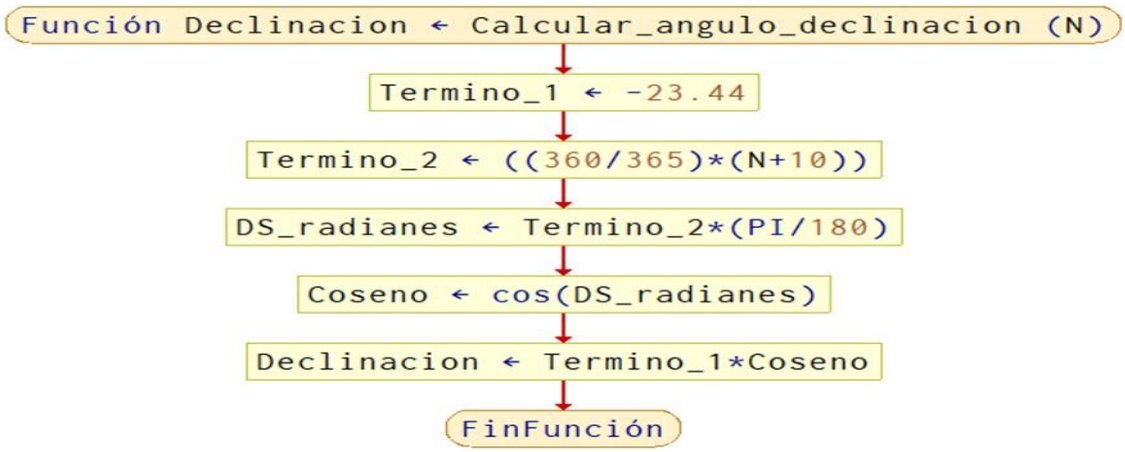


DIAGRAMA DE FLUJO DE LA FUNCIÓN DE LA ECUACIÓN DEL TIEMPO:

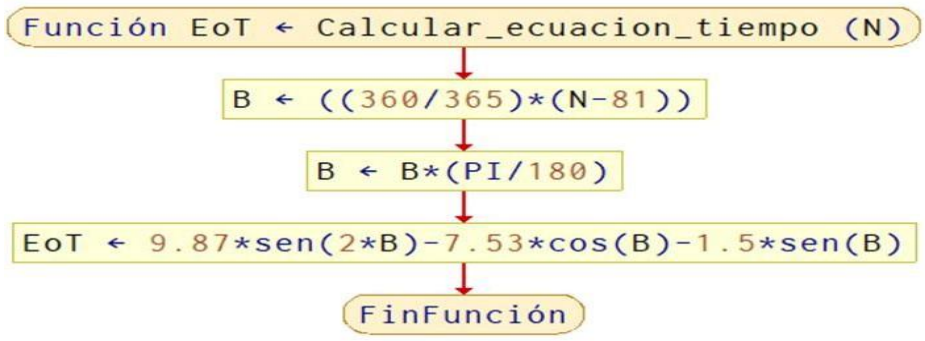


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR EL TSV:

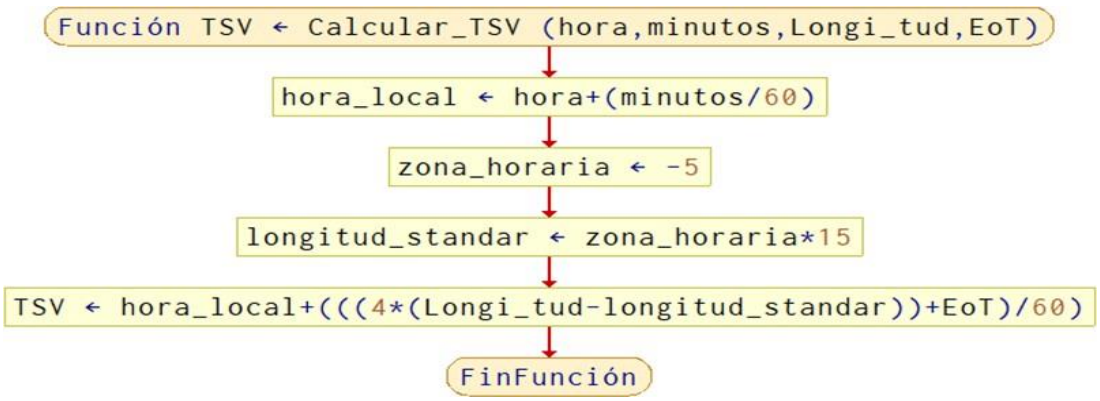


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR EL ÁNGULO HORARIO:

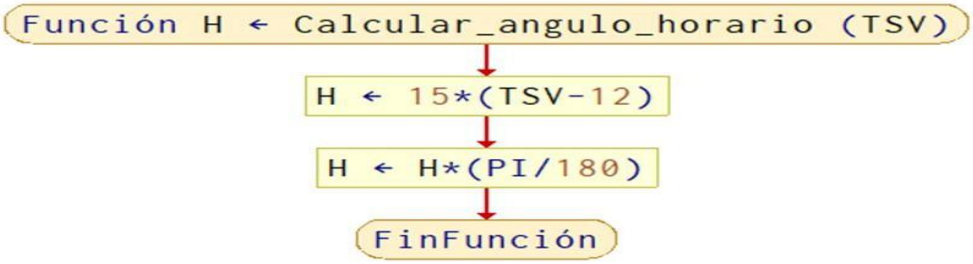


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR LA ALTITUD SOLAR:

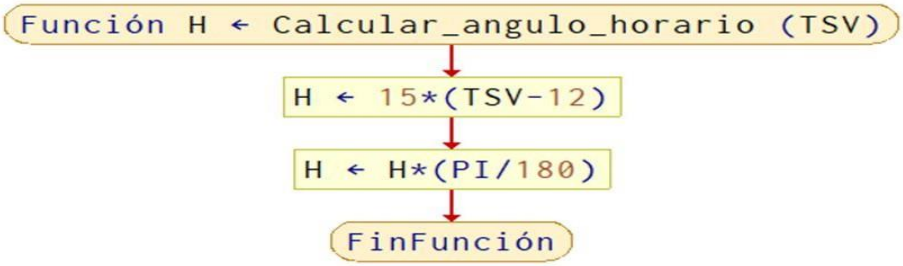
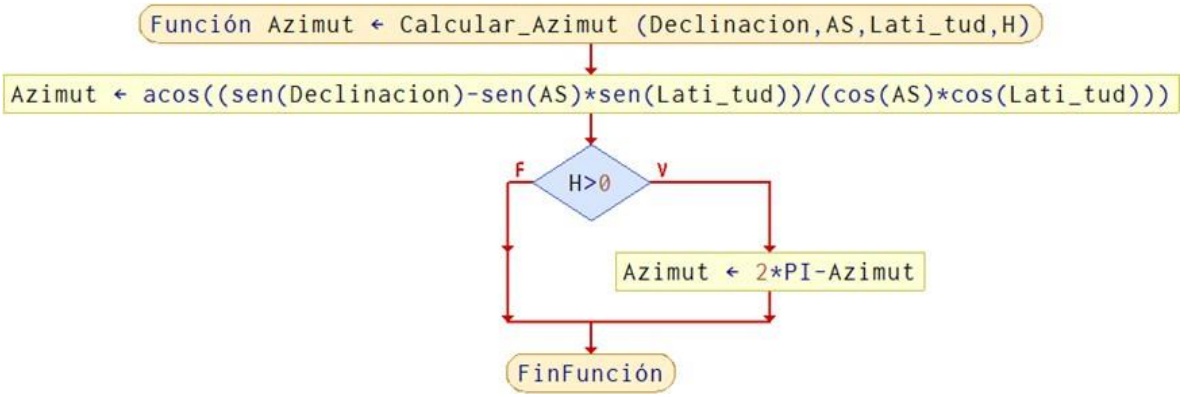


DIAGRAMA DE FLUJO DE LA FUNCIÓN PARA CALCULAR LA ORIENTACIÓN SOLAR:



Pseudocódigo principal:

```
Algoritmo Posicion_Solar
    Lati_tud ← -3.99313
    Longi_tud ← -79.20422
    // Sacamos la fecha del sistema//
    fecha ← FechaActual()
    fechatexto ← ConvertirATexto(fecha)
    f1 ← Subcadena(fechatexto,1,4)
    F2 ← Subcadena(fechatexto,5,6)
    F3 ← Subcadena(fechatexto,7,8)
    dia ← ConvertirANumero(F3)
    mes ← ConvertirANumero(F2)
    mes ← mes-1
    anio ← ConvertirANumero(f1)
    // Sacamos la hora del sistema//
    hora_actual ← HoraActual()
    horatexto ← ConvertirATexto(hora_actual)
    H1 ← Subcadena(horatexto,1,2)
    H2 ← Subcadena(horatexto,3,4)
    hora ← ConvertirANumero(H1)
    minutos ← ConvertirANumero(H2)
    // Calculamos los dias transcurridos desde que inicio el año//
    N ← Calcular_numero_dias(mes,dia,anio)
    Escribir 'Dias transcurridos desde el inicio del año => ', N
    // Calculamos la declinacion solar//
    Declinacion ← Calcular_angulo_declinacion(N)
    Escribir 'Declinacion Solar en grados => ', Declinacion
    // Calculamos EoT//
    EoT ← Calcular_ecuacion_tiempo(N)
    Escribir 'EoT => ', EoT
    // Calculamos el TSV//
    TSV ← Calcular_TSV(hora,minutos,Longi_tud,EoT)
    hora_solar ← trunc(TSV)
    minutos_solar ← redon(60*(TSV-hora_solar))
    Escribir 'TSV => ', TSV
    // Calculamos el angulo horario//
    H ← Calcular_angulo_horario(TSV)
    Escribir 'H => ', H
    // Pasamos a radianes la declinacion solar y la latitud//
    Declinacion ← Declinacion*(PI/180)
    Lati_tud ← Lati_tud*(PI/180)
    Escribir 'Declinacion solar en radianes => ', Declinacion
    Escribir 'Latitud en radianes => ', Lati_tud
    // Calculamos altitud solar//
    AS ← Calcular_AS(Declinacion,Lati_tud,H)
    AS_Grados ← AS*(180/PI)
    Escribir 'Altitud Solar => ', AS_Grados
    // Calculamos la orientación solar//
    Azimut ← Calcular_Azimut(Declinacion,AS,Lati_tud,H)
    Azimut_Grados ← Azimut*(180/PI)
    Escribir 'Orientacion Solar => ', Azimut_Grados
FinAlgoritmo
```

```
Funcion N ← Calcular_numero_dias (mes, dia, anio)
    Para i←1 Hasta mes Con Paso 1 Hacer
        Segun i Hacer
            1:
                N = N + 31
            2:
                N = N + 28
            3:
                N = N + 31
            4:
                N = N + 30
            5:
                N = N + 31
            6:
                N = N + 30
            7:
                N = N + 31
            8:
                N = N + 31
            9:
                N = N + 30
            10:
                N = N + 31
            11:
                N = N + 30
            12:
                N = N + 31
        Fin Segun
    Fin Para
    N = N + dia
    si anio % 4 == 0 Entonces
        si anio % 100 == 0 Entonces
            si anio % 400 == 0 Entonces
                N = N + 1
            Fin si
        SiNo
            N = N + 1
        FinSi

    FinSi
Fin Funcion
```

```
Funcion Declinacion ← Calcular_angulo_declinacion  ( N )
    Termino_1 = -23.44
    Termino_2 = ((360/365) * (N + 10))
    DS_radianes = Termino_2 * (PI/180)
    Coseno = cos(DS_radianes)
    Declinacion = Termino_1 * Coseno
Fin Funcion
```

Anexo 5

```
Funcion EoT ← Calcular_ecuacion_tiempo ( N )
    B = ((360/365) * (N - 81))
    B = B * (PI/180)
    EoT = 9.87 * sen(2 * B) -7.53 * cos(B) -1.5 * sen(B)
Fin Funcion
```

Anexo 6

```
Funcion TSV ← Calcular_TSV (hora, minutos, Longi_tud, EoT)
    hora_local = hora + (minutos/60)
    zona_horaria = -5
    longitud_standar = zona_horaria * 15
    TSV = hora_local + (((4*(Longi_tud - longitud_standar)) + EoT)/60)
Fin Funcion
```

Anexo 7

```
Funcion H ← Calcular_angulo_horario (TSV)
    H = 15 * (TSV - 12)
    H = H * (PI/180)
Fin Funcion
```

Anexo 8

```
Funcion Azimut ← Calcular_Azimut (Declinacion, AS, Lati_tud, H)
    Azimut = acos((sen(Declinacion) - sen(AS) * sen(Lati_tud)) / (cos(AS) * cos(Lati_tud)))
    si H > 0 Entonces
        Azimut =2 * PI - Azimut
    FinSi
Fin Funcion
```

Anexo 9

```
int Calcular_Numero_Dias(int meses, int dias, int anios)
{
    int N_Dias = 0;
    for (int i = 1; i <= meses; i++)
    {
        switch (i)
        {
        case 1:
            N_Dias = N_Dias + 31;
            break;
        case 2:
            N_Dias = N_Dias + 28;
            break;
        case 3:
            N_Dias = N_Dias + 31;
            break;
        case 4:
            N_Dias = N_Dias + 30;
            break;
        case 5:
            N_Dias = N_Dias + 31;
            break;
        case 6:
            N_Dias = N_Dias + 30;
            break;
        case 7:
            N_Dias = N_Dias + 31;
            break;
        case 8:
            N_Dias = N_Dias + 31;
            break;
        case 9:
            N_Dias = N_Dias + 30;
            break;
        case 10:
            N_Dias = N_Dias + 31;
            break;
        case 11:
            N_Dias = N_Dias + 30;
            break;
        case 12:
            N_Dias = N_Dias + 31;
            break;
        }
    }
    N_Dias = N_Dias + dias;
    if (anios % 4 == 0)
    {
        if (anios % 100 == 0)
        {
            if (anios % 400 == 0)
            {
                N_Dias = N_Dias + 1;
            }
        }
        else
        {
            N_Dias = N_Dias + 1;
        }
    }
    return N_Dias;
```

Anexo 10

```
double Calcular_Angulo_Declinacion_Solar(int N_Dias)
{
    double Termino1, Termino2, DS_Radianes, Coseno, Declinacion_Solar;
    Termino1 = -23.44;
    Termino2 = (360.0 / 365.0) * (N_Dias + 10);
    DS_Radianes = Termino2 * (M_PI / 180.0);
    Coseno = cos(DS_Radianes);
    Declinacion_Solar = Termino1 * Coseno;
    return Declinacion_Solar;
}
```

Anexo 11

```
double Ecuacion_Del_Tiempo(int N_Dias)
{
    double B, EoT;

    B = ((360.0 / 365.0) * (N_Dias - 81));
    B = B * (M_PI / 180.0);
    EoT = 9.87 * sin(2 * B) - 7.53 * cos(B) - 1.5 * sin(B);
    return EoT;
}
```

Anexo 12

```
double Calcular_Tiempo_Solar_Verdadero(int horas, int minutos, double Longitud, double EoT)
{
    int Zona_Horaria;
    double Hora_Local, Longitud_Estandar, TSV;

    Hora_Local = horas + (minutos / 60.0);
    Zona_Horaria = -5;
    Longitud_Estandar = Zona_Horaria * 15;
    TSV = Hora_Local + (((4 * (Longitud - Longitud_Estandar)) + EoT) / 60.0);
    return TSV;
}
```

Anexo 13

```
double Calcular_Angulo_Horario(double TSV)
{
    double Angulo_Horario;
    Angulo_Horario = 15 * (TSV - 12);
    Angulo_Horario = Angulo_Horario * (M_PI / 180.0);
    return Angulo_Horario;
}
```

Anexo 14

```
double Calcular_Altitud_Solar(double Declinacion_Solar, double Latitud, double Angulo_Horario)
{
    double Altitud_Solar;
    Altitud_Solar = asin(sin(Declinacion_Solar) * sin(Latitud) + cos(Declinacion_Solar) * cos(Latitud) * cos(Angulo_Horario));
    return Altitud_Solar;
}
```

Anexo 15

```
double Calcular_Azimut(double Declinacion_Solar, double Altitud_Solar, double Latitud, double Angulo_Horario)
{
    double Azimut;
    Azimut = acos((sin(Declinacion_Solar) - sin(Altitud_Solar) * sin(Latitud)) / (cos(Altitud_Solar) * cos(Latitud)));
    if (Angulo_Horario > 0)
    {
        Azimut = 2 * M_PI - Azimut;
    }
    return Azimut;
}
```

Anexo 16

```
double Pedir_Datos(const char *mensaje)
{
    float valor;
    int resultado;

    while (1)
    {
        printf("%s", mensaje);
        resultado = scanf("%lf", &valor);

        // Limpiar el dato de entrada para manejar las entradas incorrectas
        while (getchar() != '\n')
            ;

        if (resultado == 1)
        {
            return valor;
        }
        else
        {
            printf("*** Dato invalido, por favor ingrese bien el dato que se a solicitado ****\n");
        }
    }
}
```

Anexo 17

Presentación y llamado de funciones en el (main)

```c
int main()
{
    // Definimos todas las variables necesarias para nuestro programa //
    int horas, minutos, segundos;
    int anios, meses, dias;
    int N_Dias;
    double Longitud, Latitud;
    double Declinacion_Solar, EoT, TSV, Angulo_Horario, Altitud_Solar, Orientacion_Solar, Azimut, Hora_Solar, Minutos_Solar;
    /* Recuperamos la fecha y hora actual del sistema con ayuda de la libreria "time.h" y metemos cada uno de los datos en su respectiva
    variable*/
    time_t tiempoahora;
    time(&tiempoahora);
    struct tm *mitiempo = localtime(&tiempoahora);
    horas = mitiempo->tm_hour;
    minutos = mitiempo->tm_min;
    segundos = mitiempo->tm_sec;
    anios = mitiempo->tm_year + 1900;
    meses = mitiempo->tm_mon;
    dias = mitiempo->tm_mday;
    printf("================================================\n");
    printf("// FECHA Y HORA ACTUAL DEL SISTEMA //\n");
    printf("Horas => %d\n", horas);
    printf("Minutos => %d\n", minutos);
    printf("Segundos => %d\n", segundos);
    printf("Anio => %d\n", anios);
    printf("Mes => %d\n", meses);
    printf("Dia => %d\n", dias);
```

Anexo 18

```
    Latitud = -3.99313;
    Longitud = -79.20422;

    // Iniciamos el Programa llamando y asando los valores a cada una de las funciones antes definidas //
    printf("=====================================================\n");
    printf("// VERIFICACION DE DATOS //\n");
    N_Dias = Calcular_Numero_Dias(meses, dias, anios);
    printf("> Dias transcurridos desde que inicio el año => %d\n", N_Dias);

    // Calculamos la Declinacion Solar //
    Declinacion_Solar = Calcular_Angulo_Declinacion_Solar(N_Dias);
    printf("> Declinacion Solar => %f\n", Declinacion_Solar);

    // Calculamos la ecuacion del tiempo (EoT) //
    EoT = Ecuacion_Del_Tiempo(N_Dias);
    printf("> Ecuacion del tiempo => %f\n", EoT);

    // Calculamos el tiempo solar verdadero //
    TSV = Calcular_Tiempo_Solar_Verdadero(horas, minutos, Longitud, EoT);
    Hora_Solar = trunc(TSV);
    Minutos_Solar = round(60 * (TSV - Hora_Solar));
    printf("> Tiempor solar verdadero => %f\n", TSV);

    // Calculamos el angulo horario //
    Angulo_Horario = Calcular_Angulo_Horario(TSV);
    printf("> Angulo Horario => %f\n", Angulo_Horario);

    // Pasamos la Declinacion Solar y la Latitud de grados a radianes //
    Declinacion_Solar = Declinacion_Solar * (M_PI / 180.0);
    Latitud = Latitud * (M_PI / 180.0);
    printf("> Declinacion Solar en radianes => %f\n", Declinacion_Solar);
    printf("> Latitud en radianes => %f\n", Latitud);

    // Calculamos la Altitud Solar en radianes //
    Altitud_Solar = Calcular_Altitud_Solar(Declinacion_Solar, Latitud, Angulo_Horario);
    // Pasamos la Altitud Solar a grados //
    Altitud_Solar = Altitud_Solar * (180.0 / M_PI);

    // Calculamos la orientacion Solar o Azimut en radianes//
    Azimut = Calcular_Azimut(Declinacion_Solar, Altitud_Solar, Latitud, Angulo_Horario);
    // Pasamos la orientacion solar a grados //
    Azimut = Azimut * (180.0 / M_PI);

    printf("=====================================================\n");
    printf("// RESULTADOS FINALES DE ALTITUD DEL SOL Y SU ORIENTACION //\n");
    printf("> La posicion de la altitud del sol es de => %f grados de elevacion desde el piso.\n", Altitud_Solar);
    printf("> La posicion de la orientación del sol es de => %f grados desde el norte.\n", Azimut);
    printf("=====================================================\n");
    return 0;
}
```

Llamado de funciones y presentación en pantalla.

Conclusiones:

- El uso de funciones y procedimientos es importante para proporcionar un estilo más claro en la realización del programa.
- Para mantener una ubicación precisa del lugar en donde se encuentra el sol fue necesario la aplicación de distintas fórmulas para calcular la inclinación y orientación a la que se va a dirigir el panel solar.
- La aplicación de un código eficiente proporciona resultados adecuados y óptimos para su debida implementación en el panel solar.

Recomendaciones:

- Definir de una manera clara y precisa el propósito y que problema que resolverá el programa.
- Dividir el programa en pequeños módulos manejables y con una función específica.
- Tener en cuenta que las funciones a usar deben cumplir con tareas específicas las cuales ayudan a la resolución del problema.
- Realiza pruebas a cada función de manera individual para asegurar su funcionamiento.
- Mantener una documentación clara en tu código con pequeñas notas claves para que sea más fácil identificar algún error y su mantenimiento sea más fácil de realizar.

Referencias:

[1] Tutorialspoint. (s.f.). C Programming - Tutorialspoint. Recuperado de https://www.tutorialspoint.com/cprogramming/index.htm

[2] PV Education. (s.f.). Ángulo de Declinación Solar. [imagen 1] Recuperado de https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/angulo-de-declinación

[3] Algoritmos. (s.f.). Tiempo Solar. Recuperado de https://algoritmos9511.gitlab.io/_downloads/e1ac04d57c11925f0283040c533417bb/tiempo.pdf

[4] Universidad Santo Tomás, "Título del documento," en *Repositorio Institucional de la Universidad Santo Tomás*, año. [En línea]. Disponible en: https://repository.usta.edu.co/handle/11634/16519. [Consultado: 16 junio 2024].

[5] M. Álvarez Álvarez, D. Ramírez Fonseca, y L. Pérez Caballero, "Fotovoltaic technology as an alternative energy source," *Ingeniería Energética*, vol. 36, no. 2, pp. 138-145, 2015. [En línea]. Disponible en: http://scielo.sld.cu/scielo.php?pid=S1815-59012015000200008&script=sci_arttext&tlng=en. [Consultado: 17 junio 2024].

imagen 2: https://susdesign.com/popups/sunangle/eot.php

imagen 3: https://www.e-education.psu.edu/eme810/node/531

imagen 4: https://solarsena.com/solar-hour-angle-calculator-formula/

imagen 5:

https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/posici%C3%B3n-del-sol

imagen 6: https://www.pveducation.org/es/fotovoltaica/2-propiedades-de-la-luz-del-sol/%C3%A1ngulo-acimut