

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESTUDIOS GENERALES CIENCIAS

1INF01 - FUNDAMENTOS DE PROGRAMACIÓN

Guía de laboratorio #6

Estructuras Algorítmica Selectiva Múltiple e Iterativa con Salida Controlada



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

26 de octubre de 2021

Índice general

Historial de revisiones	1
Siglas	2
1. Guía de Laboratorio #6	3
1.1. Introducción	3
1.2. Materiales y métodos	3
1.3. Estructura Selectiva Múltiple	3
1.3.1. Representación de la Estructura Selectiva Múltiple	4
1.3.2. Representación en pseudocódigo	4
1.3.3. Representación en diagrama de flujo	5
1.3.4. Implementación en C	5
1.3.5. Uso de la instrucción break	6
1.3.6. Probando las opciones	6
1.3.7. Calculadora	8
1.3.8. Convertir notas numéricas a notas cualitativas aproximadas	10
1.3.9. Días en un mes	12
1.4. Estructura Iterativa con Salida Controlada	13
1.4.1. Representación de la Estructura Iterativa con Salida Controlada	14
1.4.2. Representación en pseudocódigo	14
1.4.3. Representación en diagrama de flujo	14
1.4.4. Implementación en lenguaje C	14
1.4.5. Cálculo del número e	15
1.4.6. Cálculo del seno	17
1.4.7. Cálculo del número combinatorio	17
2. Ejercicios propuestos	19
2.1. Nivel básico	19
2.1.1. Fórmula de Stokes	19
2.1.2. Radio de Van Der Waals	20

2.1.3. Promedio Ponderado	21
2.1.4. Calcular el consumo de energía eléctrica en su hogar	21
2.1.5. Números poligonales	21
2.2. Nivel Intermedio	23
2.2.1. Conversión de base 16 a base 10	23
2.2.2. Ecuaciones de Movimientos en Física	24
2.2.3. Calculadora de operaciones lógicas	24
2.2.4. Números Armstrong	25
2.2.5. Números Palíndromos	25
2.2.6. Identificar el tipo de un número	26
2.2.7. Área y Perímetro de un polígono	26
2.2.8. Probabilidad de Poisson	26
2.2.9. Magnitudes eléctricas básicas	28
2.2.10. Series de Taylor	29
2.3. Nivel avanzado	30
2.3.1. Calculadora de números complejos	30
2.3.2. Manipulación de números enteros	31
2.3.3. Cálculo de las raíces ecuaciones cuadráticas con una variable	32
2.3.4. Suma de fracciones	32
2.3.5. Dentro del cuadrado y fuera de la circunferencia (Adaptado del Laboratorio 6 – 2020-2)	32
2.3.6. Rectas Paralelas (Adaptado del Laboratorio 6 – 2020-2)	34
2.3.7. Transformando un número (Adaptado del Laboratorio 6 – 2020-2)	37
2.3.8. Dígito Encubierto (Adaptado del Laboratorio 6 – 2020-2)	39
2.3.9. Sucesión de Fibonacci (Adaptado del Laboratorio 6 – 2020-2)	40
2.3.10. Lectura de circunferencias (Adaptado del Laboratorio 6 – 2020-2)	42
2.3.11. Rectas no paralelas (Adaptado del Laboratorio 6 – 2020-2)	43
2.3.12. Perforación de triángulo (Adaptado del Laboratorio 6 – 2020-2)	45
2.3.13. Cubos y Prismas (Adaptado del Laboratorio 6 – 2021-1)	47
2.3.14. Pirámide cuadrangular y pentagonal regular (Adaptado del Laboratorio 6 - 2021-1)	54
2.3.15. Pirámide triangular y hexagonal regular (Adaptado del Laboratorio 6 - 2021-1)	59
2.3.16. Prisma pentagonal y hexagonal regular (Adaptado del Laboratorio 6 - 2021-1)	64
2.3.17. Tetraedro y hexaedro (Adaptado del Laboratorio 6 - 2021-1)	70
3. Referencias	76

Historial de Revisiones

Revisión	Fecha	Autor(es)	Descripción
1.0	03.10.2019	J.Zárate, S.Vargas	Versión inicial.
2.0	31.05.2021	D.Allasi	Reorganización e incremento de problemas.
3.0	21.10.2021	D.Allasi, S.Ponce	Reorganización e incremento de problemas.

Siglas

ASCII	American Standard Code for Information Interchange
ANSI	American National Standards Institute
EEGGCC	Estudios Generales Ciencias
IDE	Entorno de Desarrollo Integrado
OMS	Organización Mundial de la Salud
PUCP	Pontificia Universidad Católica del Perú
RAE	Real Academia Española

Capítulo 1

Guía de Laboratorio #6

1.1. Introducción

Esta guía ha sido diseñada para que sirva como una herramienta de aprendizaje y práctica para el curso de Fundamentos de Programación de los Estudios Generales Ciencias (**EEGGCC**) en la Pontificia Universidad Católica del Perú (**PUCP**). En particular se focaliza en el tema “Estructuras Algorítmica Selectiva Múltiple e Iterativa con Salida Controlada”.

Se busca que el alumno resuelva paso a paso las indicaciones dadas en esta guía contribuyendo de esta manera a los objetivos de aprendizaje del curso, en particular en el diseño de programas con estructuras selectivas múltiples e iterativas con salida controlada, usando el paradigma imperativo. Al finalizar el desarrollo de esta guía y complementando lo que se realizará en el correspondiente laboratorio, se espera que el alumno:

- Comprenda el funcionamiento de la estructura algorítmica selectiva múltiple.
- Construya programas usando estructuras algorítmicas selectivas múltiple.
- Comprenda el funcionamiento de la estructura algorítmica iterativa con salida controlada.
- Construya programas usando estructuras algorítmicas iterativas con salida controlada.

1.2. Materiales y métodos

Como herramienta para el diseño de pseudocódigos y diagramas de flujo se utilizará **PSeInt**¹. El **PSeInt** deberá estar configurado usando el perfil **PUCP** definido por los profesores del curso. Como lenguaje de programación imperativo se utilizará el lenguaje C. Como Entorno de Desarrollo Integrado (**IDE**) para el lenguaje C se utilizará **Dev C++**². No obstante, es posible utilizar otros **IDEs** como **Netbeans** y **Eclipse**.

1.3. Estructura Selectiva Múltiple

Como ya se comentó sesiones anteriores, los algoritmos siguen un flujo de ejecución el cual se puede modificar a través de estructuras de control de flujo. Las estructuras de control de flujo pueden ser de dos tipos: estructuras algorítmicas selectivas y estructuras algorítmicas iterativas.

Las estructuras selectivas evalúan una condición y si es que se cumple permiten que se ejecute un conjunto de instrucciones una sola vez. Por otra parte, las estructuras iterativas permiten que un conjunto de instrucciones se ejecute tantas veces como sea necesario, dependiendo de la evaluación de la condición.

¹<http://pseint.sourceforge.net/>

²<http://sourceforge.net/projects/orwelldevcpp>

Las estructuras algorítmicas selectivas se pueden clasificar en: estructuras selectivas simples y estructuras selectivas dobles. Dentro de las estructuras selectivas se pueden tener estructuras selectivas dobles anidadas que constan de estructuras selectivas dentro de otras estructuras selectivas. Existen ciertos casos en los que es necesario evaluar múltiples opciones y una estructura selectiva doble o una doble anidada no es necesariamente la solución más sencilla, en estos casos utilizamos las estructuras selectivas múltiples.

Recordar que:

La estructura selectiva simple permite que se ejecute un conjunto de instrucciones si se cumple determinada condición. Si la condición no se cumple, el conjunto de instrucciones no se ejecuta. La estructura selectiva doble complementa a la estructura selectiva simple porque permite ejecutar un grupo de instrucciones si es que no se cumple la condición. Esto último es lo que diferencia a la selectiva doble de la selectiva simple. En la selectiva doble se cuentan con dos conjuntos de instrucciones que dependiendo de la condición se ejecuta uno o se ejecuta el otro, por eso el nombre de selectiva doble. La estructura selectiva doble es muy usada en la programación imperativa, gracias a ella se pueden tomar decisiones sobre cómo realizar determinado procesamiento.

Las estructuras selectivas anidadas resultan de colocar una o más estructuras selectivas dentro del bloque de instrucciones a realizarse, permitiendo así el evaluar más de dos opciones o caminos posibles. Por otro lado, con las estructuras selectivas múltiples conseguimos lo mismo de una manera más directa dependiendo del modo de representación. El detalle con la implementación de estas estructuras selectivas es que la condición que se evalúa es en realidad una comparación de igualdad entre una **variable_numerica** o expresión y los valores de las opciones o casos (valores discretos).

Muchas situaciones requieren que se realice el cálculo de diferente manera dependiendo de la condición. Por ejemplo, para traducir notas numéricas a notas cualitativas aproximadas. Si la nota es 0, 1, 2 o 3 entonces la nota cualitativa es Mal. Una nota de 4 o 5 es Regular. Una nota de 6 o 7 es Bueno. Muy Bueno con una nota de 8. Excelente con 9 o 10. Como los valores a evaluar son discretos podemos utilizar estructuras selectivas múltiples. También se podrían usar estructuras selectivas dobles anidadas, pero su implementación sería más compleja. Diferente sería el caso si quisiéramos hacer la conversión de notas en Bélgica. Donde entre 0 – 9,9 es insuficiente, 10 – 11, 9 es aprobado, 12 - 13, 9 es notable 14 – 15, 9 es muy bueno, y 16 – 20 es excelente. En este caso en particular es necesario utilizar estructuras selectivas anidadas, y no es posible utilizar las selectivas múltiples, ya que en el caso de las estructuras selectivas múltiples los valores a evaluar no pueden ser valores continuos o rangos de valores.

1.3.1. Representación de la Estructura Selectiva Múltiple

A continuación, se revisará como se representa la estructura selectiva múltiple tanto en pseudocódigo como en diagrama de flujo, así como su implementación en lenguaje C.

1.3.2. Representación en pseudocódigo

En la figura ?? se puede apreciar la representación de la estructura selectiva múltiple en pseudocódigo. La estructura inicia con el identificador *Segun* y finaliza con el identificador *Fin Segun*. La *variable_numerica* es la variable que será comparada con cada una de las opciones, como, por ejemplo: la nota. Luego de la *variable_numerica* se coloca el identificador *Hacer*.

La estructura selectiva múltiple permite comparar la *variable_numerica* con una opción y si son iguales entonces se ejecuta el conjunto de instrucciones. Si son diferentes compara dicha *variable_numerica* con la siguiente opción, si son iguales ejecuta el conjunto de instrucciones correspondiente. Así se evalúan todas las opciones, para cualquier otro caso se utiliza la instrucción *De otro Modo* y ejecuta el conjunto de instrucciones correspondientes.

Como *variable_numerica* se puede colocar cualquier expresión o variable, pero esta debe guardar relación de tipo con las opciones. En las opciones se pueden colocar valores específicos o expresiones que darán como resultado un valor en particular. En las opciones no se pueden colocar rangos de valores.

```

Segun variable_numerica Hacer
    opcion_1:
        conjunto de instrucciones
    opcion_2:
        conjunto de instrucciones
    opcion_3:
        conjunto de instrucciones
    De Otro Modo:
        conjunto de instrucciones
Fin Segun

```

Figura 1.1: Pseudocódigo: Selectiva Múltiple

1.3.3. Representación en diagrama de flujo

En la figura ?? se puede apreciar la representación de la estructura selectiva múltiple en diagrama de flujo. La estructura inicia con la parte superior de la figura en donde se coloca la *variable_numerica* y en la parte inferior se coloca cada uno de los valores a comparar y de igual modo el *De Otro Modo* y si la opción es igual al valor de la variable numérica se ejecutarán el conjunto de instrucciones correspondiente.

Como *variable_numerica* se puede colocar cualquier expresión o variable, pero esta debe guardar relación de tipo con las *opciones*. En las *opciones* se pueden colocar valores específicos o expresiones, separados por comas, que darán como resultado un valor en particular. En las *opciones* no se pueden colocar rangos de valores.

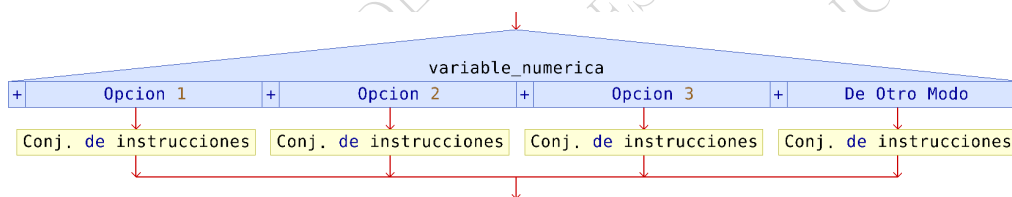


Figura 1.2: Diagrama de Flujo: Selectiva múltiple

1.3.4. Implementación en C

En el lenguaje C la estructura selectiva múltiple se implementa a través de la instrucción *switch*. La representación del *switch* se puede apreciar en el programa 1.1. La expresión es la que se compara a la expresión constante de cada caso *case*. Existe el *default*, que nos permite evaluar cualquier otra opción que no cumpla ninguno de los casos anteriores. El conjunto de instrucciones en lenguaje C se delimita por los símbolos *{ y }*. Esta delimitación es opcional cuando el conjunto de instrucciones está formado por una sola instrucción.

Recordar que:

El lenguaje C no contempla el tipo de dato lógico (*bool o boolean*), asume que cuando una condición falla es igual a 0. El comportamiento es muy similar al *falso* de una expresión lógica. Por ello, un valor diferente de 0 hará que se cumpla la condición.

Por ejemplo: si se tienen las siguientes definiciones *int suma = 0, contador = 10;*, las siguientes expresiones serán consideradas *verdaderas*: *suma <= 100, contador == 10, suma < contador*. Por otro lado, las siguientes expresiones serán consideradas *falsas*: *suma >= 100, contador == 20, suma > contador*.

Programa 1.1: C: Estructura selectiva múltiple

```

1  ...
2  switch (expresion) {
3      case <expresion constante>:
4          conjunto de instrucciones a;
5      case <expresion constante>:
6          conjunto de instrucciones b;
7      case <expresion constante>:
8          conjunto de instrucciones c;
9      default:
10         conjunto de instrucciones d;
11 }
12 ...

```

1.3.5. Uso de la instrucción break

Una instrucción de ruptura de flujo es la instrucción *break*. Esta instrucción hace que termine la ejecución completa. Luego de un *break* no se ejecutará ninguna otra instrucción que esté dentro del *switch* independientemente si la comparación es verdadera o no.

1.3.6. Probando las opciones

Pruebe la siguiente implementación en lenguaje C 1.2

Programa 1.2: C: Lectura de dos dígitos enteros

```

1  #include <stdio.h>
2  int main(){
3      int dig1, dig2;
4      printf("Ingrese un dígito del 0 al 10\n"); scanf("%d",&dig1);
5      printf("Ingrese un dígito del 0 al 10\n"); scanf("%d",&dig2);
6      printf("El primer dígito es %d , el segundo dígito es %d",dig1,dig2);
7      return 0;
8  }

```

Para poner en práctica

- Realice los cambios necesarios en 1.2 para leer dos caracteres en lugar de dos dígitos enteros.
- Pruebe el programa con los valores 'A' y 'B' respectivamente, ¿nota algún problema en la lectura?
- Implemente las modificaciones necesarias para que la lectura se realice correctamente.

Pruebe la siguiente implementación en lenguaje C 1.3 y ejecute el programa.

Programa 1.3: C: Lectura de dos dígitos enteros

```

1  #include <stdio.h>
2  int main(){
3      int dig;
4      printf("Ingrese un dígito del 0 al 10\n"); scanf("%d",&dig);
5      switch(dig){
6          case 1:
7              printf("El valor ingresado es 1\n");
8          case 2:
9              printf("El valor ingresado es 2\n");
10         case 3:
11             printf("El valor ingresado es 3\n");
12         case 4:
13             printf("El valor ingresado es 4\n");
14         case 5:
15             printf("El valor ingresado es 5\n");
16         default:

```

```

17     printf ("El valor ingresado es 6 o 7 o 8 o 9 \n");
18 }
19 return 0;
20 }

```

Para poner en práctica

- ¿Existe algún problema al ejecutar el programa?.
- Agregue las llaves de bloque donde lo considere necesario, ¿persiste el problema?.

Pruebe la siguiente implementación en lenguaje C 1.4 y ejecute el programa.

Programa 1.4: C: Selección múltiple de un dígito

```

1  #include <stdio.h>
2  int main(){
3      int dig;
4      printf("Ingrese un dígito del 0 al 10\n"); scanf(" %d",&dig);
5      switch(dig){
6          case 1:
7              {
8                  printf ("El valor ingresado es 1\n");
9              }
10         case 2:
11             {
12                 printf ("El valor ingresado es 2 \n");
13             }
14         case 3:
15             {
16                 printf ("El valor ingresado es 3\n");
17             }
18         case 4:
19             {
20                 printf ("El valor ingresado es 4 \n");
21             }
22         case 5:
23             {
24                 printf ("El valor ingresado es 5 \n");
25             }
26         default:
27             {
28                 printf ("El valor ingresado es 6 o 7 o 8 o 9 \n");
29             }
30     }
31     return 0;
32 }

```

Para poner en práctica

- ¿Existe algún problema al ejecutar el programa?.
- Agregue los break que considere necesario para que se obtenga la salida esperada.

Pruebe la siguiente implementación en lenguaje C 1.5 y ejecute el programa.

Programa 1.5: C: Selección múltiple de una expresión

```

1  #include <stdio.h>
2  int main(){
3      int dig;
4      printf("Ingrese un dígito del 0 al 10\n"); scanf(" %d",&dig);
5      switch(dig){
6          case 1:
7              printf ("El valor ingresado es 1\n");
8          case 2:
9              printf ("El valor ingresado es 2 \n");

```

```

10     case 3:
11         printf ("El valor ingresado es 3\n");
12     case 4:
13         printf ("El valor ingresado es 4 \n");
14     case 5:
15         printf ("El valor ingresado es 5 \n");
16     default:
17         printf ("El valor ingresado es 6 o 7 o 8 o 9 \n");
18 }
19 return 0;
20 }

```

Para poner en práctica

- ¿Existe algún problema al ejecutar el programa?.
- Agregue los break que considere necesario para que se obtenga la salida esperada.
- Observe que un valor numérico se puede obtener como resultado de una operación matemática.

1.3.7. Calculadora

Un ejemplo clásico visto en la guía anterior es el de la implementación de una calculadora. Dado dos números que representan a los operandos de una operación y un carácter que representa a una operación (+, -, *, /), se busca que retorne el resultado de aplicar el operador a los operandos. En caso se ingrese una operación no esperada, deberá emitir un mensaje de error.

Una alternativa de solución a este problema utilizando selectivas múltiples y diagramas de flujo se puede apreciar en la Figura 1.4. El correspondiente pseudocódigo se puede apreciar en la Figura 1.3.

```

1  Algoritmo Calculadora
2      Escribir "Ingrese dos operandos"
3      Leer numero1, numero2
4      Escribir "Ingrese operación (+, -, *, /):"
5      Leer operacion
6      Segun operacion Hacer
7          '+':
8              resultado<-numero1+numero2
9          '-':
10             resultado<-numero1-numero2
11         '*':
12             resultado<-numero1*numero2
13         '/':
14             resultado<-numero1/numero2
15         De Otro Modo:
16             Escribir "Operacion no soportada"
17     Fin Segun
18     Escribir "El resultado es:",resultado
19 FinAlgoritmo

```

Figura 1.3: Pseudocódigo: Calculadora de operadores aritméticos

Para poner en práctica

- ¿Es posible agregar a la calculadora la operación de potencia (^) y residuo (%)?. Implemente el pseudocódigo correspondiente.

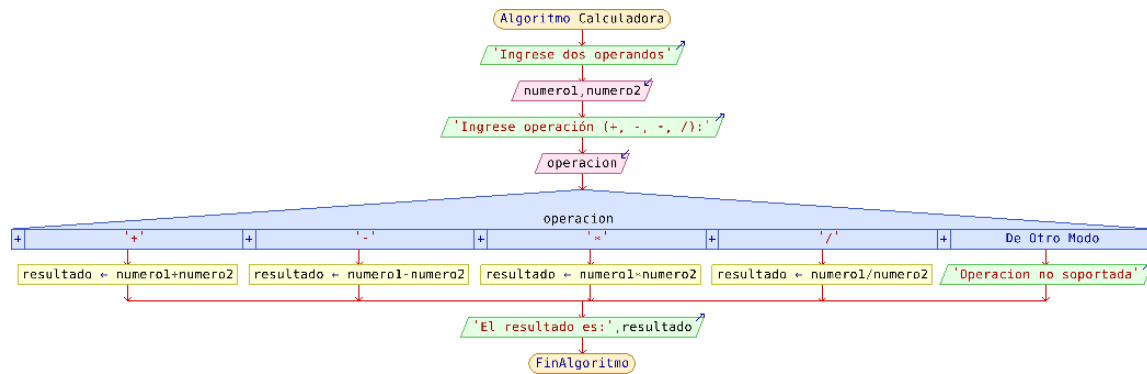


Figura 1.4: Diagrama de flujo: Calculadora de operadores aritméticos

Para poner en práctica

- ¿Cuál es la salida si para el programa 1.6 se usan los datos 20.7 16.3 / ?
- ¿Cómo se puede asegurar que el resultado sea el adecuado? Implemente el código correspondiente.

Para poner en práctica

- ¿Cómo se puede asegurar que el mensaje de la línea 27 del programa 1.6 se muestre únicamente cuando la operación es válida? Implemente la solución de la manera más eficiente posible, sin repetición innecesaria de código.
- ¿Es posible agregar a la calculadora la operación de potencia $^$ y residuo $\%$. Implemente el pseudocódigo correspondiente.

La representación del algoritmo se puede apreciar en el programa 1.6. Observe en la línea 8 como luego de `%d %d` se ha colocado `%*c` con el fin de leer el cambio de línea o enter. Esta tecla se representa internamente con un caracter especial, y este caracter especial será leído por el `%*c`. Por ese motivo es necesario leer antes ese cambio de línea para que el programa funcione correctamente.

Programa 1.6: C: Calculadora de operadores aritméticos

```

1 #include <stdio.h>
2
3 int main() {
4     int numero1, numero2; double resultado; char operacion;
5     printf("Ingrese dos operandos: \n");
6     scanf("%d %d %*c", &numero1, &numero2);
7     printf("Ingrese operación (+, -, *, /): \n");
8     switch(operacion){
9         case '+':
10             resultado=numero1+numero2;
11             break;
12         case '-':
13             resultado=numero1-numero2;
14             break;
15         case '*':
16             resultado=numero1*numero2;
17             break;
18         case '/':
19             resultado=numero1/numero2;
20             break;
21         default:
22             printf("Operación no soportada\n");
23     }
24     printf("El resultado es: %lf", resultado);
25     return 0;
26 }
```

1.3.8. Convertir notas numéricas a notas cualitativas aproximadas

Se tienen notas expresadas en números y dependiendo del valor, se le asigna una nota cuantitativa. Si la nota es 0, 1, 2 o 3 entonces la nota cualitativa es Insuficiente. Una nota de 4 o 5 es Regular. Una nota de 6 o 7 es Bueno. Distinguido con una nota de 8. Excelente con 9 o 10 (ver figura 1.5).

Nota numérica	Nota cualitativa
0 o 1 o 2 o 3	Insuficiente (I)
4 o 5	Regular(R)
6 o 7	Bueno(B)
8	Distinguido(D)
9 o 10	Excelente(E)

Figura 1.5: Escala notas cualitativas

Dado el valor de la nota se solicita que elabore un algoritmo expresado en diagrama de flujo y pseudocódigo así como un programa en lenguaje C que traduzca la nota numérica a una nota cualitativa. Recuerde que en lenguaje C el uso de cadenas no se cubre en el curso, por lo que se debe utilizar únicamente el carácter relacionado o un mensaje escrito que permita evitar el uso de cadenas.

```

1  Algoritmo Traducir_Nota_Cualitativa
2      Escribir 'Ingrese la nota'
3      Leer nota
4      Segun nota Hacer
5          0,1,2,3:
6              notaCualitativa <- 'Insuficiente'
7              Escribir 'Su calificacion es: ',notaCualitativa
8          4,5:
9              notaCualitativa <- 'Regular'
10             Escribir 'Su calificacion es: ',notaCualitativa
11          6,7:
12             notaCualitativa <- 'Bueno'
13             Escribir 'Su calificacion es: ',notaCualitativa
14          8:
15             notaCualitativa <- 'Distinguido'
16             Escribir 'Su calificacion es: ',notaCualitativa
17          9,10:
18             notaCualitativa <- 'Excelente'
19             Escribir 'Su calificacion es: ',notaCualitativa
20      De Otro Modo:
21          notaCualitativa <- 'invalida'
22          Escribir 'Su calificacion es: ',notaCualitativa
23      FinSegun
24  FinAlgoritmo

```

Figura 1.6: Pseudocódigo: Traducción a nota cualitativa

En la figura 1.6 se aprecia el pseudocódigo correspondiente a la estructura en la que se evalúa si la nota es igual a 0 o es igual a 1 o es igual 2 o es igual a 3 y de ser cierta alguna de esas comparaciones se procede a asignar el valor de la nota cualitativa y posteriormente a escribir el mensaje correspondiente. Como se puede observar si una de las opciones se cumple entonces se ejecutan todas las instrucciones anteriores a la siguiente opción.

Para poner en práctica

- ¿Es posible simplificar de un modo más eficiente el pseudocódigo 1.6? ¿Qué modificaciones haría? Implemente el pseudocódigo correspondiente
- Implemente el pseudocódigo de 1.6 utilizando estructuras selectivas dobles anidadas.

En la figura 1.7 se puede apreciar el diagrama de flujo que diseña la alternativa de solución propuesta. A diferencia que en la selectiva simple o doble, la comparación se hace de manera interna y lo que evalúa es: $\text{nota} = 0$ O $\text{nota} = 1$ O $\text{nota} = 2$ O $\text{nota} = 3$ para la primera opción y así sucesivamente para los otros juegos de opciones.

Para poner en práctica

- ¿Es posible simplificar de un modo más eficiente el diagrama de flujo 1.7? ¿Qué modificaciones haría? Implemente el diagrama de flujo correspondiente
- Implemente el diagrama de flujo de 1.7 utilizando estructuras selectivas dobles anidadas.

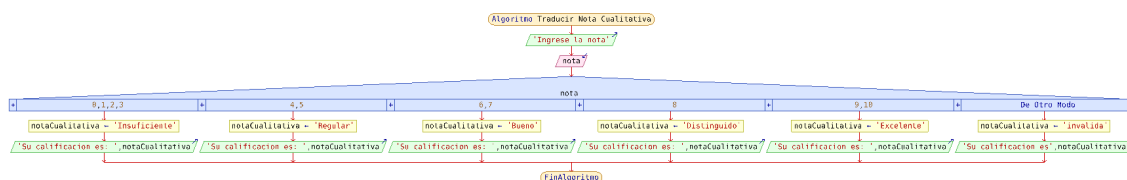


Figura 1.7: Diagrama de flujo: Traducción a nota cualitativa

Recordar que:

La indentación es una sangría que se incluye en un conjunto de instrucciones para mejorar la legibilidad del código fuente, es decir, el conjunto de instrucciones se mueve unos caracteres a la derecha. Si bien es cierto a las herramientas que ejecutan código no les afecta la indentación de código, a los humanos sí. Es mucho más fácil entender un código bien indentado que un código sin indentación. Esto afecta mucho la etapa de corrección de errores y al mantenimiento de software.

En el programa 1.7 se puede apreciar la alternativa de solución en lenguaje C. La estructura selectiva múltiple inicia en la línea 10 y finaliza en la línea 34. El bloque de instrucciones que se ejecuta cuando la nota es igual a 0 o 1 o 2 o 3 se encuentra en la línea 15 y con el fin de que no se ejecuten las siguientes instrucciones se usa la instrucción `break` en la línea 16. El `break` permite interrumpir la estructura.

Recordar que:

Un aspecto importante de la sintaxis de lenguaje C es que la expresión que representa el valor a comparar siempre va entre paréntesis. Los paréntesis no son necesarios para el caso del diagrama de flujo y pseudocódigo.

Programa 1.7: C: Traducir notas cualitativas

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <stdlib.h>
4
5 int main() {
6     int nota;
7     char notaCualitativa;

```

```
8   printf("Ingrese la nota ");
9   scanf("%d", &nota);
10  switch (nota) {
11      case 0:
12      case 1:
13      case 2:
14      case 3:
15      printf("Su calificación es: Insuficiente\n");
16      break;
17      case 4:
18      case 5:
19      printf("Su calificación es: Regular\n");
20      break;
21      case 6:
22      case 7:
23      printf("Su califica calificación es: Bueno\n");
24      break;
25      case 8:
26      printf("Su calificación es: Distinguido\n");
27      break;
28      case 9:
29      case 10:
30      printf("Su calificación es:Excelente\n");
31      break;
32      default:
33      printf("Su calificación es inválida\n");
34  }
35  return 0;
36 }
```

Para poner en práctica

- Implemente el programa de 1.7 utilizando estructuras selectivas dobles anidadas.

1.3.9. Días en un mes

Se desea diseñar un algoritmo en diagrama de flujo e implementar un programa en lenguaje C que, dado un año y un mes expresado en formato numérico, identifique cuántos días tendrá dicho mes en particular. En caso el usuario ingrese un mes erróneo, el programa deberá emitir un mensaje de error.

Para solucionar este problema es necesario recordar que:

- Tienen 31 días: enero, marzo, mayo, julio, agosto, octubre y diciembre.
- Tienen 30 días: abril, junio, septiembre y noviembre.
- Tiene 29 días: febrero (si el año es bisiesto).
- Tiene 28 días: febrero (si el año no es bisiesto).
- Son bisiestos todos los años múltiplos de 4, excepto aquellos que son múltiplos de 100 pero no de 400.

Para poner en práctica

- Implemente el pseudocódigo correspondiente al diagrama de flujo de la figura 1.8 utilizando estructuras selectivas dobles anidadas
- Implemente el programa de 1.8 utilizando estructuras selectivas dobles anidadas.

Para poner en práctica

- Implemente el pseudocódigo correspondiente de 1.8 utilizando estructuras selectivas múltiples.

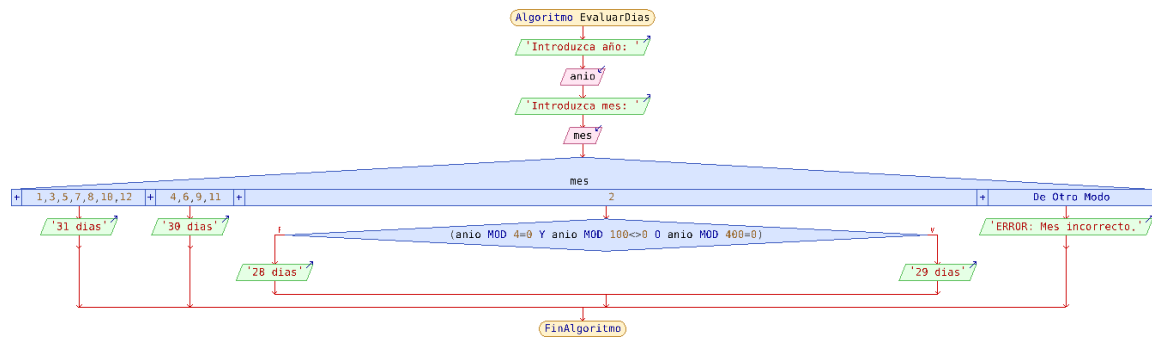


Figura 1.8: Diagrama de flujo: Días del mes en un año en particular

Programa 1.8: C: Días del mes en un año en particular

```

1 #include <stdio.h>
2 int main(){
3     int anio, mes;
4     printf("\n Introduzca anho: ");
5     scanf("%d", &anio);
6     printf("\n Introduzca mes: ");
7     scanf("%d", &mes);
8     switch ( mes ){
9         case 1 :
10        case 3 :
11        case 5 :
12        case 7 :
13        case 8 :
14        case 10 :
15        case 12 : printf( "\n 31 dias" );
16                    break;
17        case 4 :
18        case 6 :
19        case 9 :
20        case 11 : printf( "\n 30 dias" );
21                    break;
22        case 2 : /* Inicio del anidamiento */
23                if ( anio % 4 == 0 && anio % 100 != 0 || anio % 400 == 0 )
24                    printf( "\n 29 dias" );
25                else
26                    printf( "\n 28 dias" );
27                break;
28        /* Fin del anidamiento */
29        default : printf( "\n ERROR: Mes incorrecto." );
30    }
31    return 0;
32 }

```

Para poner en práctica

- Ejecute el programa 1.8 utilizando un valor negativo para el año. ¿Qué modificaciones debería hacer al código de 1.8 ?
- Implemente el programa 1.8 considerando las modificaciones que propone para solucionar la validación de un año inválido.

1.4. Estructura Iterativa con Salida Controlada

Como ya se ha revisado anteriormente, se sabe que las estructuras algorítmicas iterativas permiten que un conjunto de instrucciones se repita tantas veces como sea necesario y son muy útiles para resolver problemas que requieren realizar cálculos de forma repetitiva. Por ejemplo, se puede recorrer un rango de valores y ejecutar

un mismo conjunto de instrucciones, generar los términos de una sucesión, hallar el valor de sumatorias y productorias, ejecutar cálculos reiterativos sobre un conjunto de datos, entre otros.

La clasificación de las estructuras algorítmicas iterativas es:

- Ciclo iterativo con entrada controlada.
- Ciclo iterativo con salida controlada.

La decisión de repetir el bloque de instrucciones se realiza a partir de una condición que se puede evaluar antes o después de ejecutar la iteración. En el ciclo iterativo con entrada controlada, la evaluación de la condición del ciclo se realiza antes de la ejecución del conjunto de instrucciones y dependiendo del valor de la condición, puede que el conjunto de instrucciones no se ejecute, es decir, si antes de iniciar el ciclo la condición es falsa, entonces el bloque de instrucciones se ejecutará cero veces. Mientras que, en el ciclo iterativo con salida controlada, la evaluación de la condición del ciclo se realiza después de la ejecución del conjunto de instrucciones, por lo cual el conjunto de instrucciones se ejecuta por lo menos una vez.

1.4.1. Representación de la Estructura Iterativa con Salida Controlada

La representación de la estructura iterativa con salida controlada tanto en Pseudocódigo como en Diagrama de Flujo se verá a continuación, así como su implementación en lenguaje C.

1.4.2. Representación en pseudocódigo

En la figura 1.9 se puede apreciar la representación del ciclo iterativo con salida controlada en pseudocódigo. El ciclo inicia con el identificador Repetir y finaliza con el identificador Mientras Que. La condición es una expresión lógica, por ejemplo: $i < 100$ y $i < > 0$. En el conjunto de instrucciones se pueden colocar asignaciones ($i \leftarrow 0$), expresiones matemáticas ($\text{suma} \leftarrow \text{suma} + \text{termino}$), estructuras selectivas (Si $i=0$ Entonces) e inclusive otras estructuras iterativas.

Repetir
conjunto de instrucciones
Mientras Que condición

Figura 1.9: Pseudocódigo: Ciclo iterativo con salida controlada

1.4.3. Representación en diagrama de flujo

En la figura 1.10 se observa la representación del ciclo iterativo con salida controlada en diagrama de flujo y para ello se inicia con el bloque de instrucciones a repetir seguidamente de la condición y si la condición es verdadera, se regresa a ejecutar nuevamente el bloque de instrucciones.

En el diagrama, el control del flujo se gestiona a través de las líneas que conectan los bloques. Como se muestra, primero se ejecuta el bloque de instrucciones e seguidamente el control se dirige hacia el bloque condición. Si la condición se cumple (es verdadera), el flujo continúa por la línea V nuevamente al bloque de instrucciones para repetirse. Para evitar que el flujo se repita de forma indefinida, la condición tiene que fallar en algún momento (hacerse falsa). Cuando la condición falla, el flujo continúa por la línea F.

1.4.4. Implementación en lenguaje C

En el lenguaje C el ciclo iterativo con salida controlada se representa según especificación presentada en el programa 1.9.

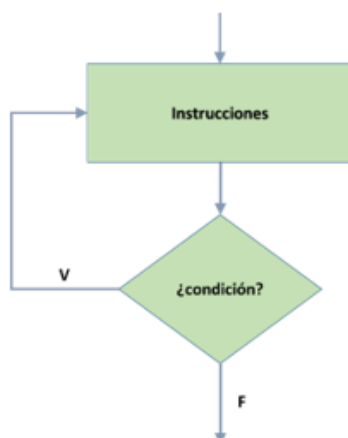


Figura 1.10: Diagrama de Flujo: Ciclo iterativo con salida controlada

Programa 1.9: C: Ciclo iterativo con salida controlada

```

1
2  do {
3      conjunto de instrucciones;
4  } while (expresion);

```

1.4.5. Cálculo del número e

El número e es un número muy conocido, usado e importante en las ciencias ya que constituye el cálculo de la función exponencial. Las primeras referencias a este número datan del año 1618 en un trabajo de John Napier sobre logaritmos. Al igual que π , el número e es un número irracional y tiene infinitas cifras decimales por lo que no se puede conocer su valor. El valor aproximado de este número es 2,71828 (aproximado a 5 decimales).

Existen numerosas definiciones del número e, la más común es a través del $(1 + \frac{1}{n})^n$. Este límite puede aproximarse con la siguiente serie $1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \frac{1}{6!} + \dots$

A continuación, se desarrollará el cálculo aproximado del número e usando la fórmula antes descrita realizando la implementación usando un ciclo iterativo con salida controlada. El control se realizará por medio de una cantidad de términos constante, para nuestro ejemplo 10. Se realizará la implementación del pseudocódigo en PSeInt.

El objetivo de esta guía es que usted la siga paso a paso, implementando cada etapa de la solución y que experimente las diversas situaciones que se comentan en ella. Se ha puesto énfasis en la solución en el ciclo iterativo con salida controlada, podrá encontrar mayor detalle de la solución de este problema en la guía #4 cuando se implementó el mismo problema usando la estructura iterativa con entrada controlada.

- Listado de n números.

Lo primero que se realizará será listar los 10 primeros números usando una estructura con salida controlada, controlando la cantidad de números a imprimir. Una alternativa de solución a este problema se puede apreciar en la figura 1.11.

- Acumulación de los términos de la serie.

Una vez que se ha conseguido generar con una estructura iterativa con salida controlada se procede a generar el término de la serie, acumular la suma y presentar el resultado al usuario. Esto se puede apreciar en la figura 1.12. El lector podrá encontrar mayor detalle de la solución a este problema en la guía #4.

- Implementación en lenguaje C - Iterativa de salida controlada por contador.

```

1  Algoritmo Calculo_Numero_e
2      numero_termino <- 1
3      Repetir
4          Escribir numero_termino
5          numero_termino <- numero_termino+1
6      Mientras Que numero_termino<=10
7  FinAlgoritmo

```

Figura 1.11: Pseudocódigo: Listado de n números

```

1  Algoritmo Calculo_Numero_e
2      numero_termino <- 1
3      suma <- 1
4      factorial <- 1
5      Repetir
6          factorial <- factorial*numero_termino
7          termino <- 1/factorial
8          suma <- suma+termino
9          numero_termino <- numero_termino+1
10     Mientras Que numero_termino<=10
11     Escribir 'El valor de número e para ',numero_termino-1,' términos es ',suma
12 FinAlgoritmo

```

Figura 1.12: Pseudocódigo: Cálculo del Número e

En el programa 1.10 se muestra la implementación en lenguaje C de acuerdo al pseudocódigo presentado anteriormente. Se pone énfasis en los aspectos de implementación propios del lenguaje de programación.

Programa 1.10: C: Ciclo iterativo con salida controlada

```

1  #include <stdio.h>
2
3  int main(){
4      int numero_termino=1;
5      double suma=1,factorial=1;
6      double termino;
7      do{
8          factorial=factorial*numero_termino;
9          termino=1/(double)factorial;
10         suma=suma+termino;
11         numero_termino++;
12     }while (numero_termino<=10);
13     printf("El valor del número e para %d términos es %.15lf\n", numero_termino-1,suma);
14     return 0;
15 }

```

Para poner en práctica

En el problema se indica que se requiere una precisión de 5 decimales, debido a que la impresión por defecto de un tipo de dato double incluye 6 decimales, no se ha realizado ninguna modificación al formato %lf. Pero si requeriría mejorar la precisión del cálculo:

- ¿Qué cambios realizaría al programa?
- ¿La precisión del cálculo es la misma si se calcula para 10, 15 o 20?

- Implementación en lenguaje C - Iterativa de salida controlada por el valor del término de la serie.

A continuación, se presenta la implementación en lenguaje C, para calcular el número e con una precisión de 7 decimales, en el programa 1.11.

Programa 1.11: C: Cálculo del número e con salida controlada por el valor del término

```

1  #include <stdio.h>
2
3  int main(){

```

```

4   int numero_termino=1;
5   double suma=1,factorial=1;
6   double termino=1;
7   do{
8       factorial=factorial*numero_termino;
9       termino=1/((double)factorial);
10      suma=suma+termino;
11      numero_termino++;
12      printf("termino %d %.15lf, valor e: %.15lf\n",numero_termino,termino,suma);
13  }while (termino>=0.00000001);
14  printf("El valor del número e para una precisión de 0.00000001 es %.15lf\n", numero_termino-1,suma);
15  return 0;
16  }

```

1.4.6. Cálculo del seno

El seno es una de las funciones trigonométricas más conocidas y usadas en las ciencias. Se utiliza por ejemplo para calcular la trayectoria de un proyectil.

El seno de un ángulo α se define como la razón entre el cateto opuesto a dicho ángulo α y la hipotenusa. Se puede calcular de diversas maneras, entre ellas, la serie de Taylor. Usando esta serie el seno de un grado x , expresado en radianes, se puede calcular de la siguiente manera: $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \frac{x^{13}}{13!} + \dots$

A continuación, se presenta la implementación en lenguaje C que calcula el valor aproximado del seno usando la serie de Taylor descrita previamente. El algoritmo deberá incluir solamente términos que en su valor absoluto tengan hasta 6 dígitos significativos.

Programa 1.12: C: Cálculo del seno

```

1  #include <stdio.h>
2  #include <math.h>
3
4  int main(){
5      int i=1,signo=1;
6      double x,suma=0,factorial=1,termino=1;
7      printf("Ingrese el angulo en radianes:\n");
8      scanf("%lf",&x);
9      do{
10         factorial=factorial*i;
11         if (i%2==1){
12             termino=signo*pow(x,i)/factorial;
13             suma=suma+termino;
14             signo=signo*(-1);
15         }
16         i++;
17     }while(fabs(termino)>=0.0000001);
18     printf("El seno calculado es %.10lf\n",suma);
19     return 0;
20 }

```

Para poner en práctica

- Considerando que el problema inicial se modifique y le solicite calcular el seno, pero con una precisión en base a la cantidad de decimales del último término que se ingresa, ¿qué cambios debería realizar?.
- Se sugiere implementar dos subprogramas, el primero que reciba el número de decimales y calcule la precisión del último término y el segundo que recibiendo la precisión calcule el seno.

1.4.7. Cálculo del número combinatorio

Se define número combinatorio como el valor numérico de las combinaciones ordinarias (sin repetición) de un conjunto de p elementos tomados en grupos de r , siendo p y r dos números enteros y positivos tales que $p \geq r$.

r. Matemáticamente, un número combinatorio se expresa como:

A continuación, se presenta la implementación en lenguaje C que calcule el combinatorio de x sobre y.

Programa 1.13: C: Número combinatorio

```
1 #include <stdio.h>
2
3 int main(){
4     int x,y,i=1;
5     int fact_x=1,fact_xmenosy=1,fact_y=1;
6     int combinatorio;
7     printf("Ingrese los valores para calcular el combinatorio de x sobre y:\n");
8     scanf("%d %d", &x,&y);
9     do{
10         fact_x*=i;
11         if (i<=y)
12             fact_y*=i;
13         if (i<=(x-y))
14             fact_xmenosy*=i;
15         i++;
16     }while (i<=x);
17     combinatorio=fact_x/(fact_xmenosy*fact_y);
18     printf("El combinatorio de %d sobre %d es %d\n",x,y,combinatorio);
19     return 0;
20 }
```

Capítulo 2

Ejercicios propuestos

Para cada uno de los ejercicios propuestos se solicita que elabore el correspondiente algoritmo representado tanto en diagrama de flujo como en pseudocódigo así como la implementación de un programa en C conforme a los temas revisados en las guías preliminar, #1, #2, #3, #4 y #5 del curso Fundamentos de Programación. Se recomienda que las soluciones en C incluyan funciones definidas por el usuario.

2.1. Nivel básico

2.1.1. Fórmula de Stokes

La ley de Stokes se refiere a la fuerza de fricción experimentada por objetos esféricos moviéndose en el seno de un fluido viscoso en un régimen laminar de bajos números de Reynolds. En general la ley de Stokes es válida en el movimiento de partículas esféricas pequeñas moviéndose a velocidades bajas. La ley de Stokes puede escribirse como¹: $F_d = 6 * \pi * R * n * v$, donde Donde R es el radio de la esfera, v su velocidad y n la viscosidad del fluido.

La condición de bajos números de Reynolds implica un flujo laminar lo cual puede traducirse por una velocidad relativa entre la esfera y el medio inferior a un cierto valor crítico. En estas condiciones la resistencia que ofrece el medio es debida casi exclusivamente a las fuerzas de rozamiento que se oponen al deslizamiento de unas capas de fluido sobre otras a partir de la capa límite adherida al cuerpo. La ley de Stokes se ha comprobado experimentalmente en multitud de fluidos y condiciones.

Si las partículas están cayendo verticalmente en un fluido viscoso debido a su propio peso puede calcularse su velocidad de caída o sedimentación igualando la fuerza de fricción con el peso aparente de la partícula en el fluido.

$$V_s = \frac{2}{9} \frac{r^2 g (\rho_p - \rho_f)}{\eta}$$

Figura 2.1: Velocidad de caída

- V_s es la velocidad de caída de las partículas
- (velocidad límite) g es la aceleración de la gravedad
- ρ_p es la densidad de las partículas
- ρ_f es la densidad del fluido.

¹<https://books.google.com.pe>

- n es la viscosidad del fluido.
- r es el radio equivalente de la partícula.

Elabore el diseño en pseudocódigo e implemente el programa en lenguaje C que permita calcular el valor de F_d y el valor de la V_s en base al material seleccionado por el usuario para la esfera, el radio de la esfera y el fluido. Debe utilizar estructuras selectivas múltiples y dobles anidadas.

Material de la esfera	Densidad (g/cm ³)
Hierro (H)	7.88
Aluminio (A)	2.70
Cobre (C)	8.93
Plomo (P)	11.35
Volframio (V)	19.34

Figura 2.2: Densidad de esfera según el material

Fluido	Densidad (g/cm ³)	Viscosidad (kg/m·s)
Agua (A)	1.0	0.00105
Glicerina (G)	1.26	1.3923
Benceno (B)	0.88	0.000673
Aceite (C)	0.88	0.391

Figura 2.3: Densidad y viscosidad

2.1.2. Radio de Van Der Waals

El radio de Van der Waals es el radio de una esfera sólida imaginaria utilizada para modelizar el átomo.

Los gases reales no se comportan exactamente como predice el modelo de gas ideal pudiendo ser la desviación considerable en algunos casos. Así, por ejemplo, los gases ideales no presentan transiciones de fase líquida o sólida, independientemente del descenso de temperatura o incremento de presión al que sean sometidos.

Una de las modificaciones de la ley de los gases ideales propuesta es la ecuación de estado de Van der Waals, que introduce dos parámetros a y b obtenidos experimentalmente y que dependen de la naturaleza del gas. El factor de corrección b denominado volumen de exclusión, hace referencia tanto al volumen propio de los átomos, como al volumen circundante en el que no puede haber otros porque a esa distancia predominan las fuerzas de repulsión entre los átomos del gas (fuerzas de Van der Waals)².

Código una letra	Aminoácido	radio VDW
A	Alanina	67
R	Arginina	148
N	Asparagina	96
D	Ácido aspártico	91
C	Cisteína	86
Q	Glutamina	114
E	Ácido glutámico	109
G	Glicina	48
H	Histidina	118

Figura 2.4: Valores de radio de Van Der Waals

²<https://esacademic.com/>

Elabore el diseño en pseudocódigo e implemente el programa en lenguaje C que permita identificar el nombre del aminoácido y el valor del radio de Van Der Waals considerando el código de una letra. Debe utilizar estructuras selectivas múltiples y dobles anidadas.

Sugerencia:

Considere el código de una letra para cada uno de los aminoácidos

Recordar que:

La constante de Avogadro (símbolos: L, N) es el número de partículas constituyentes (usualmente átomos o moléculas) que se encuentran en la cantidad de sustancia de un mol. Por tanto, es el factor proporcional que relaciona la masa molar de una sustancia a la masa de una muestra. Su valor es igual a $6,022 \times 10^{23} \text{ mol}^{-1}$.

2.1.3. Promedio Ponderado

Se solicita que lea un listado de códigos de cursos con su respectivo número de créditos y la nota obtenida y determine el promedio ponderado. Los códigos de cursos están formados por dígitos y la nota obtenida solo se restringe a números enteros. El flujo de lectura finalizará cuando se registre como código del curso un valor negativo.

Variación del problema

Dado el listado de cursos, créditos y notas descrito anteriormente, determine las notas con mayor y menor contribución al promedio ponderado. Debe verificar que las notas no sean mayores a 20 ni menores a 0.

2.1.4. Calcular el consumo de energía eléctrica en su hogar

Diariamente tenemos en nuestro hogar una gran cantidad de electrodomésticos conectados a la energía eléctrica sin conocer cuánto consume c/u de ellos. El consumo de energía eléctrica afecta directamente a nuestra economía, al uso de nuestros recursos naturales no renovables y produce contaminación ambiental. Por ello es importante controlar el uso de los electrodomésticos en nuestro hogar.

Se le pide que lea un listado de electrodomésticos, su consumo en watts(W) y la cantidad de horas al mes que está encendido c/u de ellos; y calcule el total de la energía eléctrica que consume mensualmente en kilowatts(kW).

Adicionalmente se le solicita que identifique al electrodoméstico que consume mayor energía y el pago mensual que realiza, solicitando que se ingrese la tarifa por kW consumido.

2.1.5. Números poligonales

Un número poligonal es aquel que puede representarse como puntos dispuestos en forma de un polígono regular (ver figura 2.5).

La fórmula para hallar el n-ésimo número poligonal de l lados es: $n = \frac{n*((l-2)*n*(l-4))}{2}$

Se le pide que lea la cantidad de lados del polígono y calcule los primeros n números poligonales correspondientes.



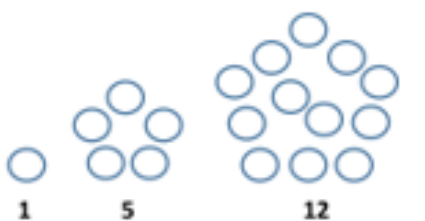
Polígono	Representación
Triangular	 1 3 6
Cuadrangular	 1 4 9
Pentagonal	 1 5 12

Figura 2.5: Números Poligonales

Casos de prueba

Utilice los siguientes datos para probar su solución.

- Si la cantidad de lados es 3 y n es 10, se debe mostrar: 1 3 6 10 15 21 28 36 45 55.
- Si la cantidad de lados es 5 y n es 15, se debe mostrar: 1 5 12 22 35 51 70 92 117 145 176 210 247 287 330.
- Si la cantidad de lados es 12 y n es 8, se debe mostrar: 1 12 33 64 105 156 217 288.

Variación del problema

- Dado un número determine si es un número triangular, pentagonal u octagonal.
- Sugerencia: Desarrolle un subprograma que verifique si un número es poligonal, recibiendo como parámetros un número y el número de lados del polígono.
- Si el número es 21, se debe mostrar:
 - Se puede representar como número triangular.
 - Se puede representar como número octogonal.
- Si el número es 145, se debe mostrar:
 - Se puede representar como número pentagonal.
- Si el número es 100, se debe mostrar:
 - No se puede representar como número triangular, pentagonal ni octagonal.

2.2. Nivel Intermedio

2.2.1. Conversión de base 16 a base 10

Elabore el diseño en pseudocódigo e implemente el programa en lenguaje C que permita convertir un número de dos dígitos en base 16 a base 10. Para ello el usuario debe ingresar dígito por dígito dicho número.

Sugerencia:

- Recuerde que el código ASCII del caracter 1 es 48 y del caracter 0 es 47.
- Un número de dos dígitos en base 16, tiene el dígito de más a la izquierda multiplicado por 16 elevado a la 1 y el dígito de más a la derecha multiplicado por 16 elevado a la 0.
- Recuerda que los dígitos válidos en esta base son del 0 al 9 y de la A que representa 10 hasta la F que representa 15.
- Recuerde utilizar estructuras selectivas múltiples para su solución.

El siguiente ejemplo muestra un caso de ejecución.

Ingrese el número de tres dígitos en base 16, dígito por dígito.

Ingrese el primer dígito de más a la izquierda: A

Ingrese el segundo dígito de más a la derecha: C

El valor AC en base 10 es: 172

El siguiente ejemplo muestra un caso de ejecución.

Ingrese el número de tres dígitos en base 16, dígito por dígito.

Ingrese el primer dígito de más a la izquierda: 1

Ingrese el segundo dígito de más a la derecha: 7

El valor 17 en base 10 es: 23

2.2.2. Ecuaciones de Movimientos en Física

El movimiento es uno de los fenómenos más evidentes, al ser fácilmente observable. La rama de la Física que se encarga del estudio de este fenómeno es la cinemática, que estudia las leyes del movimiento sin tener en cuenta las causas que lo han producido. Elabore el diseño en pseudocódigo e implemente el programa en lenguaje C que permita calcular la posición final de un cuerpo en un determinado tiempo y dependiendo del tipo de movimiento que está realizando. Para ello se considera:

- A: Movimiento Rectilíneo Uniforme (M.R.U.).
- B: Movimiento Rectilíneo Uniforme Variado (M.R.U.V.).
- C: Caída libre

Debe solicitar al usuario que ingrese la letra asociada al movimiento que desea calcular y, dependiendo del movimiento se debe de solicitarle las variables necesarias para calcular la posición final de un cuerpo. Si ingresa una letra incorrecta, debe mostrar un mensaje indicando que la opción ingresada es inválida.

Recordar que:

- En M.R.U. la fórmula de posición final de un cuerpo es la siguiente: $X = X_o + v * t$, donde X_o = Posición Inicial, v = velocidad y t =tiempo.
- En M.R.U.V. la fórmula de posición final de un cuerpo es la siguiente: $X = X_o + v * t + (1/2) * a * t^2$, donde X_o = Posición Inicial, v = velocidad, t =tiempo y a =aceleración.
- En Caída Libre la fórmula de posición final de un cuerpo es la siguiente: $Y = H - (1/2) * g * t^2$, donde H = La altura desde la que se deja caer el cuerpo, g = aceleración de la gravedad que es 9.8 m/s^2 y t =tiempo.

2.2.3. Calculadora de operaciones lógicas

Se desea implementar una calculadora de operaciones lógicas. Para ello, se cuenta con dos caracteres que representan el valor de una proposición lógica (V o F) y un caracter que representa una operación lógica a realizar, considerar C que referencia a la operación de conjunción, D a la disyunción, K a la operación de condicional y B que representa a la operación bicondicional. Se pide que implemente la calculadora y retorne el resultado de aplicar la operación lógica a los valores dados. En caso que los valores dados y la operación ingresada no corresponda con las antes mencionadas, deberá emitirse un mensaje de advertencia y no realizar las operaciones.

Casos de prueba

Utilice los siguientes datos para probar su solución.

- Si $p = V$, $q = V$ y operación = C se debe imprimir El resultado es V.
- Si $p = V$, $q = F$ y operación = C se debe imprimir El resultado es F.
- Si $p = F$, $q = F$ y operación = D se debe imprimir El resultado es F.
- Si $p = V$, $q = F$ y operación = D se debe imprimir El resultado es V.
- Si $p = V$, $q = F$ y operación = K se debe imprimir El resultado es F.
- Si $p = F$, $q = F$ y operación = B se debe imprimir El resultado es V.

Se desea implementar una calculadora de operaciones lógicas. Para ello, se cuenta con dos caracteres que representan el valor de una proposición lógica (V o F) y un caracter que representa una operación lógica a

realizar, considerar C que referencia a la operación de conjunción, D a la disyunción, K a la operación de condicional y B que representa a la operación bicondicional. Se pide que implemente la calculadora y retorne el resultado de aplicar la operación lógica a los valores dados. En caso que los valores dados y la operación ingresada no corresponda con las antes mencionadas, deberá emitirse un mensaje de advertencia y no realizar las operaciones.

Casos de prueba:

- Si $p = V$, $q = V$ y operación = C se debe imprimir El resultado es V.
- Si $p = V$, $q = F$ y operación = C se debe imprimir El resultado es F.
- Si $p = F$, $q = F$ y operación = D se debe imprimir El resultado es F.
- Si $p = V$, $q = F$ y operación = D se debe imprimir El resultado es V.
- Si $p = V$, $q = F$ y operación = K se debe imprimir El resultado es F.
- Si $p = F$, $q = F$ y operación = B se debe imprimir El resultado es V.

2.2.4. Números Armstrong

Un número Armstrong, también llamado número narcisista, es todo aquel número que es igual a la suma de cada uno de sus dígitos elevado al número total de dígitos. Dado un número entero determine si es un número de Armstrong, no debe solicitar el número de dígitos del número.

Caso de prueba

- Si el número es 4210818, se debe mostrar: El número es Armstrong.
- Si el número es 7842456, se debe mostrar: El número NO es Armstrong.
- Si el número es 92727, se debe mostrar: El número es Armstrong.

Variación del problema

- ¿El número a evaluar debe validarse?
- ¿Cuál es la máxima cantidad de dígitos que almacena un dato tipo int? Revise la librería limits.h y añada la validación al problema.

2.2.5. Números Palíndromos

Un número es palíndromo si se lee igual de izquierda a derecha y al revés (de derecha a izquierda).

Dado un número entero determine si es un número palíndromo, no debe solicitar el número de dígitos del número.

Casos de prueba

- Si el número es 1234321, se debe mostrar: El número es Palíndromo.
- Si el número es 992456, se debe mostrar: El número NO es Palíndromo.
- Si el número es 978676879, se debe mostrar: El número es Palíndromo.

Variación del problema

- ¿El número a evaluar debe validarse?.
- ¿Cuál es la máxima número que almacena un dato tipo int? Revise la librería limits.h y añada la validación al problema.

2.2.6. Identificar el tipo de un número

De acuerdo a las siguientes definiciones de un número, evalúe un número y determine el tipo de número que corresponde lista de números enteros positivos y determine para cada uno de ellos el tipo de número que corresponde.

- Número primo: Solo se puede dividir entre el mismo número y el uno.
- Número perfecto: La suma de sus divisores, sin considerar el mismo número, es igual al número.
- Número abundante: La suma de sus divisores, sin considerar el mismo número, es mayor al número.
- Número deficiente: La suma de sus divisores, sin considerar el mismo número, es menor al número.

Sugerencia

- Desarrolle un subprograma que devuelva la suma de sus divisores sin considerar el mismo número.
- Compare la suma con el número y determine qué tipo de número es.

2.2.7. Área y Perímetro de un polígono

Para un polígono de n lados en el plano cartesiano, calcule su perímetro y área. Para el perímetro utilice la distancia euclidiana y para el área el algoritmo de Lazada. Los algoritmos respectivos están descritos en las guías anteriores del curso.

2.2.8. Probabilidad de Poisson

La teoría de colas es una disciplina que mediante métodos matemáticos analíticos analiza el comportamiento de un sistema de colas. Un sistema de colas se puede describir cómo sigue. Un conjunto de “clientes” llega a un sistema buscando un servicio, esperan si este no es inmediato, y abandonan el sistema una vez han sido atendidos. En algunos casos se puede admitir que los clientes abandonan el sistema si se cansan de esperar.

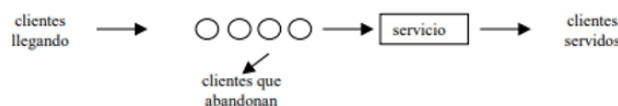


Figura 2.6: Un sistema de cola básico. Imagen tomada de Tomado de “Aplicando Teoría de Colas en Dirección de Operaciones- José Pedro García Sabater”

En este ámbito, se utiliza la distribución de Poisson para estimar la probabilidad que lleguen n clientes a la cola en un intervalo de tiempo t : Donde:

- n : número de clientes que llegan por minuto,
- λt : número promedio de llegadas por minuto

$$P(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}$$

Utilizando la distribución de Poisson, determine la probabilidad de llegada de una cantidad de clientes a una cola en un banco. Los valores de la cantidad de clientes y el promedio de clientes por minuto se ingresan al programa. Debe validar que ambos sean números mayores a 0.

Ejemplos de ejecución del programa:

Ejemplo 1

Ingrese hasta que cantidad de clientes desea evaluar: 5

Ingrese el promedio de clientes por minuto que llegan al banco: 2

Probabilidad para que llegue 1 clientes 0.270671

Probabilidad para que llegue 2 clientes 0.270671

Probabilidad para que llegue 3 clientes 0.180447

Probabilidad para que llegue 4 clientes 0.090224

Probabilidad para que llegue 5 clientes 0.036089

Ejemplo 2

Ingrese hasta que cantidad de clientes desea evaluar: 5

Ingrese el promedio de clientes por minuto que llegan al banco: -2

Los datos ingresados no son válidos

Ejemplo 3

Ingrese hasta que cantidad de clientes desea evaluar: 10

Ingrese el promedio de clientes por minuto que llegan al banco: 3

Probabilidad para que llegue 1 clientes 0.149361

Probabilidad para que llegue 2 clientes 0.224042

Probabilidad para que llegue 3 clientes 0.224042

Probabilidad para que llegue 4 clientes 0.168031

Probabilidad para que llegue 5 clientes 0.100819

Probabilidad para que llegue 6 clientes 0.050409

Probabilidad para que llegue 7 clientes 0.021604

Probabilidad para que llegue 8 clientes 0.008102

Probabilidad para que llegue 9 clientes 0.002701

Probabilidad para que llegue 10 clientes 0.000810

Sugerencia

Para hallar el número e, desarrolle un subprograma utilizando el algoritmo descrito anteriormente en esta guía.

2.2.9. Magnitudes eléctricas básicas

Las magnitudes eléctricas básicas son la Tensión (V), Intensidad de corriente (I) y Resistencia (R). Estas magnitudes se relacionan por medio de la Ley de Ohm, que establece que la tensión (V) que se aplica sobre los extremos de un conductor es proporcional a la intensidad de la corriente (I) que circula por el conductor e inversamente proporcional a la resistencia (R) conectada al circuito (ver figura 2.7).

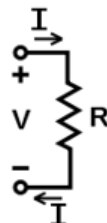


Figura 2.7: Ley de Ohm

Adicionalmente otra magnitud importante es la Potencia eléctrica (P). Estas cuatro magnitudes se relacionan entre ellas a través de las fórmulas descritas en la siguiente tabla (ver figura 2.8):

Magnitud	Descripción	Unidad de Medida	Fórmulas
Tensión	Diferencia de potencial entre dos elementos eléctricos.	voltio (V)	$V = RI$ $V = \frac{P}{I}$ $V = \sqrt{PR}$
Intensidad de corriente	Cantidad de electricidad que atraviesa un conductor en una unidad de tiempo	amperio (A)	$I = \frac{V}{R}$ $I = \frac{P}{V}$ $I = \sqrt{\frac{P}{R}}$
Resistencia	Dificultad que opone un circuito al paso de una corriente	ohmio (Ω)	$R = \frac{V}{I}$ $R = \frac{V^2}{P}$ $R = \frac{P}{I^2}$
Potencia	Cantidad de energía eléctrica entregada o absorbida por un elemento eléctrico	watt (W)	$P = VI$ $P = RI^2$ $P = \frac{V^2}{R}$

Figura 2.8: Tabla de Magnitudes

Se pide que reciba una lista de dos de las magnitudes (tensión, intensidad de corriente, resistencia o potencia) y calcule las otras dos faltantes.

Ejemplo de ejecución del programa:

El orden de ingreso de las magnitudes es Tensión Intensidad de Corriente Resistencia Potencia. Coloque 0 en las magnitudes que desea calcular, solo 2 magnitudes pueden ser diferente de 0. Para terminar, ingrese -1 para todas las magnitudes.

0 0 10 6

La tensión es 7.745967 V.

La intensidad de corriente es 0.774597 A.

12 15 0 0

La resistencia es 0.800000 Ohmio.

La potencia es 180.000000 W.

12 0 16 0

La intensidad de corriente es 0.750000 A.

La potencia es 9.000000 W.

10 0 0 2

La intensidad de corriente es 0.200000 A.

La resistencia es 50.000000 Ohmio.

0 0 0 2

Solo dos magnitudes pueden tener el valor de cero. No se cumplen con el formato de datos solicitados.

10 0 3 0

La intensidad de corriente es 3.333333 A.

La potencia es 33.333333 W.

Sugerencia

Desarrolle los siguientes subprogramas:

- Validar datos.
- Calcular la tensión, recibiendo como parámetros las otras 3 magnitudes.
- Calcular la intensidad de corriente, recibiendo como parámetros las otras 3 magnitudes.
- Calcular la resistencia, recibiendo como parámetros las otras 3 magnitudes.
- Calcular la potencia, recibiendo como parámetros las otras 3 magnitudes.

2.2.10. Series de Taylor

Dado un ángulo cualquiera (en grados sexagesimal), calcule, usando las series de Taylor, el seno y coseno hiperbólico correspondiente y compare este cálculo con el resultado de las funciones implementadas en math.h. Solicite la cantidad de términos que desea utilizar en la serie, el ángulo en sexagesimal, la cantidad de decimales para evaluar la precisión con el resultado de la función math.h y presente el valor del seno y coseno hiperbólicos de la serie de Taylor y el resultado de la función de math.h.

Considere el primer término de las series para $n=0$. Puede apoyarse en la siguiente tabla:

Serie de Taylor	math.h
$\text{seno hiperbólico}(x) = \sum_{n=0}^{\infty} \frac{(x)^{2n+1}}{(2n+1)!}, \quad -\infty < x < \infty$	$\sinh(x)$
$\text{coseno hiperbólico}(x) = \sum_{n=0}^{\infty} \frac{(x)^{2n}}{(2n)!}, \quad -\infty < x < \infty$	$\cosh(x)$

Ejemplos de ejecución del programa:

Ejemplo 1

Ingrese el ángulo en sexagesimal: 180

Ingrese la cantidad de términos: 6

Ingrese la cantidad de decimales de precisión: 4

Precisión 0.000100

Seno hiperbólico según Taylor 11.547361

Seno hiperbólico según math 11.547850

El cálculo de Taylor para el seno hiperbólico NO coincide con la función de la librería.

Coseno hiperbólico Taylor es 11.589029

Coseno hiperbólico math.h es 11.591067

El cálculo de Taylor para el coseno hiperbólico NO coincide con el de la librería.

Ejemplo 2

Ingrese el ángulo en sexagesimal: 60

Ingrese la cantidad de términos: 3

Ingrese la cantidad de decimales de precisión: 4

Precisión 0.000100

Seno hiperbólico según Taylor 1.249048

Seno hiperbólico según math 1.249326

El cálculo de Taylor para el seno hiperbólico NO coincide con la función de la librería.

Coseno hiperbólico Taylor es 1.598387

Coseno hiperbólico math.h es 1.600255

El cálculo de Taylor para el coseno hiperbólico NO coincide con el de la librería.

Sugerencia

Desarrolle los siguientes subprogramas:

- Convertir grados sexagesimales a radianes.
- Calcular el seno hiperbólico.
- Calcular el coseno hiperbólico.
- Calcular la precisión para evaluación.

2.3. Nivel avanzado

2.3.1. Calculadora de números complejos

Dado dos números complejos representados en forma binomial (i.e., $z = a + bi$) y un caracter que representa una operación aritmética (+, -, *, y /) se solicita que retorne el resultado de aplicar la operación en los dos números complejos dados.

Recordar que:

Dado dos números complejos $a + bi$ y $c + di$. Se definen las siguientes operaciones:

- suma: $(a + bi) + (c + di) = (a + c) + (b + d)i$
- resta: $(a + bi) - (c + di) = (a - c) + (b - d)i$
- multiplicación: $(a + bi) (c + di) = (ac - bd) + (ad + bc)i$
- división: $\frac{(a + bi)}{(c + di)} = \frac{(ac + bd)}{(c^2 + d^2)} + \frac{(bc - ad)}{(c^2 + d^2)}i$

Sugerencia:

- Utilice 2 variables para leer un número complejo, una para representar la parte real y otra para representar la parte imaginaria. Por ejemplo, para el primer número podría utilizar las variables a y b y para el segundo número las variables c y d .
- Verifique que no exista división entre 0.
- Si el resultado de una operación retorna la parte imaginaria negativa, entonces deberá imprimirse la parte imaginaria con el signo negativo (e.g., abi), en caso contrario deberá imprimirse el signo positivo (e.g., $a + bi$).

2.3.2. Manipulación de números enteros

Dado un número n que pertenece a los números naturales y un caracter o que representa un caracter, se desea que se retorne la inversa del número n cuando el caracter o es igual a 'I', retorne la suma de los dígitos elevado al cubo cuando el caracter o es igual a 'A' y retorne si el número es capicúa cuando el caracter o es igual a 'C'. La cantidad de dígitos del número deberá ser de exactamente 3 caracteres. Si el caracter no es ni 'A' ni 'I' ni 'C', deberá retornar el mensaje "Opción inválida". Si la cantidad de dígitos no es la indicada, deberá retornar el mensaje "Debe ingresar un número de 3 dígitos".

El siguiente ejemplo muestra un caso de ejecución cuando se desea invertir un número.

Ingrese número n: -153
Ingrese opción o: I
El número invertido es: -351

El siguiente ejemplo muestra un caso de ejecución cuando se desea obtener la suma de los dígitos al cubo del número.

Ingrese número n: 121
Ingrese opción o: A
La suma de dígitos al cubo es: 10

El siguiente ejemplo muestra un caso de ejecución cuando se desea retornar si el número es capicúa.

Ingrese número n: 121
Ingrese opción o: C
El número 121 SI es capicúa.

El siguiente ejemplo muestra un caso de ejecución cuando la opción ingresada es incorrecta.

Ingrese número n: 146
Ingrese opción o: W
Opción inválida

El siguiente ejemplo muestra un caso de ejecución cuando la cantidad de dígitos del número no es la adecuada.

Ingrese número n: 4578
Ingrese opción o: A
Debe ingresar un número de 3 dígitos

2.3.3. Cálculo de las raíces ecuaciones cuadráticas con una variable

Dada una ecuación cuadrática con una variable, realice el cálculo de las raíces y presente la ecuación correspondiente. Deberá solicitar al usuario ingresar la letra B si desea expresar las raíces complejas usando la representación binómica o la letra P si desea visualizar la representación polar. Si ingresa una letra incorrecta, debe mostrar un mensaje indicando que la opción ingresada es inválida.

Recordar que:

La representación binómica de un número complejo es de la forma $x + yi$ Donde:

- x es la parte real.
- y es la parte imaginaria.

Recordar que:

La representación polar de un número complejo es de la forma $|z|(\cos(\theta) + i\sin(\theta))$ Si se tiene el número de la forma $x + yi$ se puede calcular:

- $|z| = \sqrt{x^2 + y^2}$
- $\cos(\theta) = \frac{x}{\sqrt{x^2 + y^2}}$
- $\sin(\theta) = \frac{y}{\sqrt{x^2 + y^2}}$

2.3.4. Suma de fracciones

Se pide que, dada una lista de fracciones, calcule e imprima la suma de ellas. El flujo de lectura de datos finalizará cuando se ingrese una fracción que tenga como numerador 0 o un número negativo.

Sugerencia:

- Recuerde que para sumar dos fracciones:

$$\frac{a}{b} + \frac{c}{d} = \frac{(m.c.m(b, d) \div b) \times a + (m.c.m(b, d) \div d) \times c}{m.c.m(b, d)}$$

- Recuerde que:

$$m.c.m(a, b) = \frac{a \times b}{m.c.d(a, b)}$$

- Utilice un ciclo iterativo controlado por un centinela. El centinela podría ser una variable como encontrado que cuando se ingresa un número 0 se le asigna el valor de 1. Esta variable la podría inicializar en 0.

2.3.5. Dentro del cuadrado y fuera de la circunferencia (Adaptado del Laboratorio 6 – 2020-2)

Se solicita implementar un programa en lenguaje C que permita al usuario un punto (x,y) y, en seguida, solicite repetidas veces el ingreso de un cuadrado y una circunferencia. Se deberá validar que el cuadrado y la circunferencia ingresadas cumplan con la condición de que el punto (x,y) se encuentre dentro del cuadrado pero fuera de la circunferencia. En caso no se cumpla alguna de estas condiciones, deberá imprimir los mensajes

“El punto se encuentra fuera del cuadrado” y/o “El punto se encuentra dentro de la circunferencia, según corresponda”.

Una vez ingresado el punto (x,y), el programa deberá proceder de la siguiente manera:

- Leer los 4 vértices del cuadrado. Se deberá asumir que el usuario siempre ingresa un cuadrado matemáticamente correcto y los vértices del cuadrado se ingresan de tal forma que el punto A es el vértice superior izquierdo, el punto B es el vértice superior derecho, el punto C es el vértice inferior derecho y el punto D es el vértice inferior izquierdo.
- Leer el centro de la circunferencia (h,k) y el perímetro de esta. Se deberá asumir que el usuario siempre ingresa una circunferencia matemáticamente correcta y que el perímetro es positivo.
- Se deberá validar que el cuadrado y la circunferencia ingresados permitan que el punto (x,y) ingresado al inicio del ejercicio, esté dentro del cuadrado pero fuera de la circunferencia (el punto no debe estar exactamente en el borde del cuadrado ni exactamente sobre la circunferencia).
- Se deberá repetir este proceso hasta que se ingrese un cuadrado y una circunferencia que no hagan posible que el punto se encuentre dentro del cuadrado, pero fuera de la circunferencia.

El programa deberá hacer uso de los siguientes módulos:

- Un módulo llamado **leer_cuadrado** que reciba como parámetros cuatro puntos (x1, y1), (x2, y2), (x3, y3) y (x4, y4) a ser leídos. Para la lectura de cada uno de estos puntos, el módulo **leer_cuadrado** deberá hacer un llamado a otro módulo llamado **leer_punto**, el cual recibirá como parámetros las coordenadas del punto que se desea leer, es decir, una variable x y una variable y, así como también una letra A, B, C o D que permitirá emitir los mensajes Ingrese el punto A, Ingrese el punto B, Ingrese el punto C o Ingrese el punto D según corresponda. Se deberá asumir que el usuario siempre ingresa un cuadrado matemáticamente correcto y los vértices del cuadrado se ingresan de tal forma que el punto A es el vértice superior izquierdo, el punto B es el vértice superior derecho, el punto C es el vértice inferior derecho y el punto D es el vértice inferior izquierdo.
- El módulo **leer_punto** mencionado en el párrafo anterior.
- Un módulo llamado **leer_circunferencia** que reciba como parámetro los 3 valores a leer, es decir, las coordenadas del punto (h, k) y el perímetro. Se deberá asumir que el usuario siempre ingresa una circunferencia matemáticamente correcta y que el perímetro es positivo.
- Un módulo llamado **validar_dentro_cuadrado** que retorne 1 si es que el punto está dentro del cuadrado y retorne 0 si es que está fuera del cuadrado o exactamente en el borde de este.
- Un módulo llamado **validar_fuera_circunferencia** que retorne 1 si es que el punto ingresado se encuentra fuera de la circunferencia, y retorne 0 si es que se encuentra dentro de la circunferencia o exactamente en la circunferencia. Recuerde que el radio de una circunferencia se obtiene con la siguiente fórmula: $\frac{\text{perímetro}}{2\pi}$

La solución natural de este ejercicio es mediante el uso de una estructura algorítmica iterativa con salida controlada, pues se debe realizar por lo menos 1 iteración para ingresar la primera pareja de cuadrado y circunferencia. Resolver este ejercicio con una estructura algorítmica iterativa con entrada controlada será considerado como error.

A continuación, se muestran algunos ejemplos de ejecución:

Ejemplo 1

Ingrese un punto (x,y): 0 0

Iteración 1:

Ingrese el punto A: -5 5
Ingrese el punto B: 5 5
Ingrese el punto C: 5 -5
Ingrese el punto D: -5 -5
Ingrese el centro (h,k): 4 4
Ingrese el perímetro: 1

Iteración 2:
Ingrese el punto A: -5 5
Ingrese el punto B: 5 5
Ingrese el punto C: 5 -5
Ingrese el punto D: -5 -5
Ingrese el centro (h,k): 1 1
Ingrese el perímetro: 3

Iteración 3:
Ingrese el punto A: -5 5
Ingrese el punto B: 5 5
Ingrese el punto C: 5 -5
Ingrese el punto D: -5 -5
Ingrese el centro (h,k): 1 1
Ingrese el perímetro: 10
El punto se encuentra dentro de la circunferencia

Ejemplo 2

Ingrese un punto (x,y): 0 0

Iteración 1:
Ingrese el punto A: -5 5
Ingrese el punto B: 5 5
Ingrese el punto C: 5 -5
Ingrese el punto D: -5 -5
Ingrese el centro (h,k): 0 0
Ingrese el perímetro: 0.5
El punto se encuentra dentro de la circunferencia

Ejemplo 3

Ingrese un punto (x,y): 6 6

Iteración 1:
Ingrese el punto A: -5 5
Ingrese el punto B: 5 5
Ingrese el punto C: 5 -5
Ingrese el punto D: -5 -5
Ingrese el centro (h,k): 0 0
Ingrese el perímetro: 4
El punto se encuentra fuera del cuadrado

2.3.6. Rectas Paralelas (Adaptado del Laboratorio 6 – 2020-2)

Se solicita implementar un programa en lenguaje C que permita al usuario ingresar dos puntos (x_1, y_1) y (x_2, y_2) y calcular la recta que pasa por esos 2 puntos. En seguida, el programa deberá imprimir la pendiente y la ordenada en el origen de dicha recta.

Luego de esto, el programa deberá permitirle al usuario ingresar múltiples rectas paralelas a dicha recta. Estas deben estar siempre por encima de la recta anterior, o siempre por debajo de la anterior, según el usuario lo desee. El programa deberá preguntarle al usuario si desea ingresar rectas por encima (carácter 'E' o 'e') o rectas por debajo (carácter 'D' o 'd'). Si el usuario ingresa un carácter diferente, el programa emitirá el mensaje “Debe ingresar D/E y finalizará inmediatamente”.

Una vez decidido si se desea ingresar rectas siempre por encima de las anteriores, o rectas siempre por debajo de las anteriores, el programa deberá proceder de la siguiente manera:

- Solicitar al usuario que ingrese un punto (x, y) . El programa deberá calcular una recta que pase por dicho punto y, además, sea paralela a la recta ingresada anteriormente.
- Se deberá imprimir la pendiente y la ordenada en el origen de la nueva recta obtenida.
- Repetir el proceso tomando como referencia esta nueva recta ingresada (es decir, el programa debe olvidar la recta original con la que se dio inicio a este ejercicio, y volver a pedir un nuevo punto, el cual pasará por una recta paralela a la nueva recta obtenida).
- Se deberá repetir el proceso hasta que el usuario ingrese un punto inválido. Se dice que un punto es inválido en cualquiera de las siguientes situaciones: 1) cuando el usuario eligió ingresar rectas paralelas siempre por debajo de las rectas anteriores, y el usuario ingresa un punto ubicado encima de la última recta obtenida, o exactamente en la última recta obtenida, y 2) cuando el usuario eligió ingresar rectas paralelas siempre por encima de la recta anterior, y el usuario ingresa un punto ubicado debajo de la última recta obtenida, o exactamente en la última recta obtenida.
- En caso se ingrese un punto inválido, el programa deberá emitir los mensajes “Solo debe ingresar rectas por debajo de la recta anterior” o “Solo debe ingresar rectas por encima de la recta anterior”, según corresponda.
- Asumir que el usuario nunca ingresa rectas verticales (siempre ingresa rectas ascendentes, descendente u horizontales).

La solución deberá contener los siguientes módulos:

- El módulo principal (main).
- Un módulo que permita obtener la pendiente y la ordenada en el origen de la primera recta de todas, a través de 2 puntos (x_1, y_1) y (x_2, y_2) .
- Un módulo que permita obtener la ordenada en el origen de otra recta. Este módulo deberá ser llamado múltiples veces, teniendo en cuenta que todas las nuevas rectas obtenidas tendrán siempre la misma pendiente de la primera recta original. Este módulo recibirá como parámetro el nuevo punto ingresado por el usuario y la pendiente de la recta original.
- Un módulo que permita verificar que la dirección ingresada por el usuario es válida. Esta dirección se refiere a la dirección hacia la que el usuario desea ingresar nuevas rectas: Encima de las rectas anteriores, o debajo de las rectas anteriores. Este módulo retornará 1 si es que la dirección es válida y 0 si es que no es válida. Debe contemplar mayúsculas y minúsculas para los valores de la dirección (D/d/E/e).

La solución natural de este ejercicio es mediante el uso de una estructura algorítmica iterativa con salida controlada, pues se debe realizar por lo menos 1 iteración para ingresar la primera recta paralela. Resolver este ejercicio con una estructura algorítmica iterativa con entrada controlada será considerado como error.

A continuación, se muestran algunos ejemplos de ejecución:

Ejemplo 1

Ingrese el primer punto: 1 4
Ingrese el segundo punto: 5 2
La pendiente es: -0.500000

La b es: 4.500000

Ingrese dirección (D = debajo, E = encima): Q

Debe ingresar D/E

Ejemplo 2

Ingrese el primer punto: 1 4

Ingrese el segundo punto: 5 2

La pendiente es: -0.500000

La b es: 4.500000

Ingrese dirección (D = debajo, E = encima): D

Iteración 1:

Ingrese un punto: 4 7

La pendiente es: -0.500000

La b es: 9.000000

Solo debe ingresar rectas por debajo de la recta anterior

Ejemplo 2

Ingrese el primer punto: 1 4

Ingrese el segundo punto: 5 2

La pendiente es: -0.500000

La b es: 4.500000

Ingrese dirección (D = debajo, E = encima): d

Iteración 1:

Ingrese un punto: 4 7

La pendiente es: -0.500000

La b es: 9.000000

Solo debe ingresar rectas por debajo de la recta anterior

Ejemplo 3

Ingrese el primer punto: 4 7

Ingrese el segundo punto: 8 5

La pendiente es: -0.500000

La b es: 9.000000

Ingrese dirección (D = debajo, E = encima): e

Iteración 1:

Ingrese un punto: 5 2

La pendiente es: -0.500000

La b es: 4.500000

Solo debe ingresar rectas por encima de la recta anterior

Ejemplo 4

Ingrese el primer punto: 1 4

Ingrese el segundo punto: 5 2

La pendiente es: -0.500000

La b es: 4.500000

Ingrese dirección (D = debajo, E = encima): E

Iteración 1:

Ingrese un punto: 4 7

La pendiente es: -0.500000

La b es: 9.000000

Iteración 2:

Ingrese un punto: 5 7

La pendiente es: -0.500000

La b es: 9.500000

Iteración 3:

Ingrese un punto: 6 7

La pendiente es: -0.500000

La b es: 10.000000

Iteración 4:

Ingrese un punto: 5 7

La pendiente es: -0.500000

La b es: 9.500000

Solo debe ingresar rectas por encima de la recta anterior

2.3.7. Transformando un número (Adaptado del Laboratorio 6 – 2020-2)

Implementar un programa en lenguaje C que solicite un número, y lo transforme usando el siguiente método:

- Si el número tiene una cantidad impar de dígitos, aumentarle el dígito 0 a la derecha.
- Partir el número en 2 números nuevos llamados Parte 1 y Parte 2: si el número original tenía 'N' dígitos, los nuevos números tendrán '(N-2)' dígitos cada uno. Por ejemplo, el número 123456 se deberá partir en 123 y en 456.
- Invertir los dígitos de Parte 1 (parte que contiene la primera mitad del número original, es decir, la mitad con las cifras más significativas del número original).
- Si la Parte 1 invertida es igual a la Parte 1 antes de su inversión, entonces se deberá invertir también la parte 2.
- Concatenar la Parte 1 invertida con la Parte 2 o con la parte 2 invertida, según corresponda.
- Imprimir el número resultante. Por ejemplo, el número 123456 pasaría a ser 321456, pues es la concatenación de 321 (inversión de 123) con 456.

Se deberá asumir que el usuario nunca ingresa números que contienen el dígito 0. Se deberá hacer uso de los siguientes módulos:

- Un módulo llamado **contar_digitos** que cuente la cantidad de dígitos que tiene el número ingresado. Este módulo recibirá como parámetro el número a evaluar y retornará la cantidad de dígitos que tiene dicho número.
- Un módulo llamado **partir_numero** que reciba como parámetro un número a dividir y calcule el valor para dos parámetros llamados **partel1** y **parte2** los cuales contendrán las particiones del número. Cada una de estas 2 particiones tendrá la mitad de cifras del número original. La primera partición contendrá a las cifras más significativas del número original, la segunda partición contendrá a las cifras menos significativas del número original. Por ejemplo, para el número 8425, la primera parte será 84 y la segunda parte será 25.

- Un módulo llamado **invertir_numero**, que reciba como parámetro un número a invertir, y retorne la inversión de dicho número. Este módulo deberá ser llamado con el objetivo de invertir la parte 1 del número original.
- Un módulo llamado **concatenar**, que reciba como parámetros 2 números a ser concatenados y retorne el número concatenado. Por ejemplo, si recibe el número 25 y el número 79, este módulo retornará 2579, en cambio, si recibe 412 y 795, este módulo retornará 412795.

La solución natural de este ejercicio requerirá un total de dos estructuras algorítmicas iterativas con salida controlada (una en el módulo **contar_digitos** y otra en el módulo **invertir_numero**). Resolver este ejercicio con una estructura algorítmica iterativa con entrada controlada será considerado como error.

A continuación, se muestran algunos ejemplos de ejecución: Ejemplo 1

Ingrese un número: 12345678
El numero a transformar es: 12345678
Parte 1 = 1234
Parte 2 = 5678
La parte 1 invertida es: 4321
El número final es: 43215678

Ejemplo 2

Ingrese un número: 12345
El numero a transformar es: 123450
Parte 1 = 123
Parte 2 = 450
La parte 1 invertida es: 321
El número final es: 321450

Ejemplo 3

Ingrese un número: 2
El numero a transformar es: 20
Parte 1 = 2
Parte 2 = 0
La parte 1 invertida es: 2
La parte 2 invertida es: 0
El número final es: 20

Ejemplo 4

Ingrese un número: 1221567
El numero a transformar es: 12215670
Parte 1 = 1221
Parte 2 = 5670
La parte 1 invertida es: 1221
La parte 2 invertida es: 0765
El número final es: 12210765

2.3.8. Dígito Encubierto (Adaptado del Laboratorio 6 – 2020-2)

Implementar un programa en lenguaje C que solicite un número, una cifra a encubrir, y un carácter que encubrirá la cifra a encubrir. El programa deberá reemplazar la cifra a encubrir y mostrar el resultado. Por ejemplo, si ingresa el número 12345 y decide encubrir la cifra 2 con una X, el programa imprimirá 1X345.

El programa, deberá funcionar tanto para números positivos como para negativos. Por ejemplo, si se ingresa el número -542841 y se desea encubrir la cifra 4 con el carácter F, el programa deberá imprimir -5F28F1.

Se deberá asumir que el usuario siempre ingresa números que no tienen el dígito 0.

Se deberá hacer uso de los siguientes módulos:

- Un módulo que verifique si el número ingresado es positivo. Este módulo retornará 1 si el número es positivo y retornará 0 si no lo es.
- Un módulo llamado **contar_digitos** que cuente la cantidad de dígitos que tiene el número ingresado. Este módulo recibirá como parámetro el número a evaluar y retornará la cantidad de dígitos que tiene dicho número.
- • Un módulo llamado **imprimir_camouflado** que reciba como parámetro un número positivo, la cifra a encubrir, y el carácter con el que se quiere camuflar la cifra. Este módulo deberá imprimir el número encubierto final. Para poder hacer este proceso, el módulo **imprimir_camouflado** deberá recorrer dígito por dígito comparando cada dígito del número con la cifra que se desea encubrir, si algún dígito es igual a la cifra que se quiere encubrir, este se deberá reemplazar por el valor del carácter que sirve de camuflaje. Para poder recorrer todas las cifras del número será necesario saber cuántos dígitos tiene dicho número, para lo cual será necesario hacer un llamado al módulo **contar_digitos**.

La solución natural de este ejercicio requerirá un total de dos estructuras algorítmicas iterativas con salida controlada (una en el módulo **contar_digitos** y otra en el módulo **imprimir_camouflado**). Resolver este ejercicio con una estructura algorítmica iterativa con entrada controlada será considerado como error.

A continuación, se muestran algunos ejemplos de ejecución: Ejemplo 1

Ingrese un número positivo: 12345678
Ingrese el dígito a encubrir: 3
Ingrese carácter de encubrimiento: a
El valor encubierto es: 12a45678

Ejemplo 2

Ingrese un número positivo: 45625857
Ingrese el dígito a encubrir: 5
Ingrese carácter de encubrimiento: 4
El valor encubierto es: 44624847

Ejemplo 3

Ingrese un número positivo: -68432
Ingrese el dígito a encubrir: 2
Ingrese carácter de encubrimiento: F
El valor encubierto es: -6843F

Ejemplo 4

Ingrese un número positivo: 456789
Ingrese el dígito a encubrir: 6
Ingrese carácter de encubrimiento: 6
El valor encubierto es: 456789

2.3.9. Sucesión de Fibonacci (Adaptado del Laboratorio 6 – 2020-2)

La Sucesión de Fibonacci es una conocida sucesión que empieza con 0 y 1, y a partir de ese momento, cada término que sigue es la suma de los dos anteriores. Aplicando esta lógica, tendríamos la siguiente sucesión:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597...

Existen muchos métodos que permiten obtener esta sucesión. Algunos métodos más eficientes que otros (eficiencia medida en el número de pasos o en el número de operaciones que se debe realizar para encontrar un término de dicha sucesión).

A continuación, se explica un método que permite encontrar un término de esta sucesión en muy pocos pasos. La posición del término está dada por un índice que empieza en 0 (es decir, el primer término de todos es el término en la posición 0, el segundo término es el término en la posición 1, y así). El término en la posición 0 en esta sucesión tiene el valor de 0, y el término en cualquier otra posición (es decir, desde la posición 1 en adelante) se obtiene con el siguiente método:

- Inicializar el punto (a, b) en (1,0) y el punto (c, d) en (0,1).
- Inicializar una variable auxiliar i en la posición del término menos 1 (por ejemplo, si se está buscando el término 7, la variable auxiliar i se inicializará en 6, asimismo, si se está buscando el término 1, la variable auxiliar i se inicializará en 0).
- Si esta variable auxiliar i es mayor que 0, entonces, deberá realizar el siguiente proceso iterativo:
 - Si la variable auxiliar i es impar, sobrescribir el punto (a, b) con los valores $(bd + ac, bd + ad + bc)$.
 - Sobrescribir el punto (c, d) con los valores $(c^2 + d^2, 2cd + d^2)$.
 - Sobrescribir a la variable auxiliar i por $i/2$, para esta operación, se debe realizar la división entera, es decir, 5 entre 2 será 2, 7 entre 2 será 3, y así.
 - Seguir iterando mientras que la nueva i sea positiva. Cuando la i calculada deje de ser positiva, entonces el término de la Sucesión Fibonacci será a+b.

Se pide:

Implementar un programa en lenguaje C que permita obtener un término de la Sucesión Fibonacci. El programa deberá iniciar solicitando al usuario la posición del término que desea calcular. Se debe tener en cuenta que el primero de los términos es el término en posición 0, el segundo de los términos es el término en posición 1, y así sucesivamente.

Se deberá hacer uso de los siguientes módulos:

- El programa deberá validar que la posición ingresada sea mayor o igual a cero. En caso se ingrese una posición menor que cero, el programa deberá imprimir el mensaje “Debe ingresar una posición mayor o igual a cero” y finalizar la ejecución inmediatamente. La validación se hará mediante un módulo llamado **validar_termino**, el cual retornará 1 si es que el término es válido y 0 si no lo es.
- Si la posición del término es válida, el programa deberá obtener el término de la sucesión Fibonacci correspondiente, mediante el método expuesto párrafos arriba. Este término se deberá calcular mediante un módulo llamado **calcular_termino** que reciba como parámetro la posición del término a calcular, y retorne 0 si la posición es 0, pero retorne a+b en cualquier otro caso (nota: el valor de a+b que se retorna debe ser el valor a+b obtenido después de haber realizado el método expuesto). Este módulo, además,

deberá contar cuántas iteraciones se realizaron al calcular el término de la serie Fibonacci. Por ejemplo, si el término es 0 no se realizó ninguna iteración, porque en ese caso, se retorna 0, asimismo, si el término es 1, tampoco se realiza ninguna iteración, porque la variable auxiliar i no sería mayor que 0 y en ese caso solo se retorna $a+b$ calculado con los valores iniciales de a y de b . En cambio, si el término es 2 o más, entonces la variable auxiliar i sería superior a 0 y habría por lo menos 1 iteración. Al terminar la última iteración, este módulo deberá imprimir el número de iteraciones que fueron necesarios para obtener el término en la posición solicitada, y retornar el valor de dicho término.

- Un módulo que permita determinar si i es positivo. Este módulo se llamará **verificar_positivo** y retornará 1 si es que la i es positiva y retornará 0 si es que no lo es. Este módulo debe ser usado en la condición de la estructura iterativa con salida controlada usada en la función **calcular_termino**.

Al final del programa, se deberá imprimir el término de la sucesión Fibonacci calculado.

La solución natural de este ejercicio es mediante el uso de una estructura algorítmica iterativa con salida controlada, pues en caso de que la variable auxiliar i sea mayor que cero, se debe realizar por lo menos 1 iteración para obtener un valor para (a, b) a ser usado en el término $a+b$ (en cambio, para los términos 0 o 1 no es necesario realizar ninguna iteración, según lo señalado en los párrafos anteriores. Resolver este ejercicio con una estructura algorítmica iterativa con entrada controlada será considerado como error.

A continuación, se muestran algunos ejemplos de ejecución: Ejemplo 1

Ingrese posición del término: -5
Debe ingresar una posición mayor o igual a cero

Ejemplo 2

Ingrese posición del término: 0
Número de pasos: 0
El término 0 de la sucesión Fibonacci es: 0

Ejemplo 3

Ingrese posición del término: 1
Número de pasos: 0
El término 1 de la sucesión Fibonacci es: 1

Ejemplo 4

Ingrese posición del término: 8
Número de pasos: 3
El término 8 de la sucesión Fibonacci es: 21

Ejemplo 5

Ingrese posición del término: 15
Número de pasos: 4
El término 15 de la sucesión Fibonacci es: 610

2.3.10. Lectura de circunferencias (Adaptado del Laboratorio 6 – 2020-2)

Se solicita implementar un programa en lenguaje C que permita al usuario ingresar dos puntos (x_1, y_1) y (x_2, y_2) y, en seguida, solicite el ingreso de circunferencias hasta que los 2 puntos ingresados inicialmente no estén dentro de la circunferencia ingresada. Cada vez que se ingresan circunferencias que contengan solo a uno de los 2 puntos, se deberá imprimir el mensaje indicando cuál es el punto que está fuera de ella.

Una vez ingresados los 2 puntos iniciales, el programa deberá proceder de la siguiente manera:

- Solicitar al usuario que ingrese el centro de la primera circunferencia (h,k) y el área comprendida en el interior de dicha circunferencia. Se deberá asumir que el usuario siempre ingresa áreas positivas.
- Se deberá calcular el radio de la circunferencia, a partir del área comprendida dentro de ella.
- Se considerará una circunferencia como válida, si por lo menos uno de los dos puntos ingresados al inicio del ejercicio se encuentra comprendido dentro de ella (esto excluye a los puntos comprendidos exactamente en la circunferencia).
- Si solamente uno de los dos puntos se encuentra fuera de la circunferencia, se deberá emitir los mensajes “El punto 1 está fuera de la circunferencia ingresada” o “El punto 2 está fuera de la circunferencia ingresada, según corresponda”.
- Se deberá seguir ingresando más y más circunferencias mediante centro (h,k) y área comprendida dentro de ella, hasta que se ingrese una circunferencia que haga que los 2 puntos ingresados al inicio del ejercicio no se encuentren dentro de ella. En ese caso, se deberá emitir el mensaje “Los 2 puntos están fuera de la circunferencia ingresada” y finalizar la ejecución del programa inmediatamente.

Se deberá hacer uso de los siguientes módulos:

- Un módulo para la lectura de los datos de una circunferencia, es decir, las coordenadas del centro (h,k) , y el área.
- Un módulo que permita calcular el radio de una circunferencia que reciba el centro (h,k) y el área y retorne el radio calculado mediante la siguiente fórmula: $\sqrt{\left(\frac{A}{\pi}\right)}$, donde A es el área comprendida dentro de la circunferencia y pi es el valor 3.1415926536.
- Un módulo que establezca si una circunferencia es válida o no; este módulo se llamará **validar_circunferencia** y retornará 1 si es que la circunferencia es válida y 0 si es que no lo es (se debe tener en cuenta que no se está validando que exista la circunferencia, sino que contenga por lo menos a 1 de los 2 puntos ingresados).
- Si se cree conveniente calcular la distancia entre 2 puntos, se podrá hacer uso de un módulo llamado **calcular_distancia** que reciba como parámetros las coordenadas de dos puntos (x_1, y_1) y (x_2, y_2) y retorne la distancia entre dichos puntos.

La solución natural de este ejercicio es mediante el uso de una estructura algorítmica iterativa con salida controlada, pues se debe realizar por lo menos 1 iteración para ingresar la primera circunferencia. Resolver este ejercicio con una estructura algorítmica iterativa con entrada controlada será considerado como error.

A continuación, se muestran algunos ejemplos de ejecución: Ejemplo 1

Ingrese el primer punto: 0 0
 Ingrese el segundo punto: 5 5

Iteración 1:
 Ingrese el centro de la circunferencia h,k: 5 5
 Ingrese área: 0.5
 El punto 1 está fuera de la circunferencia ingresada

Iteración 2:

Ingrese el centro de la circunferencia h,k: 2.5 2.5

Ingrese área: 100

Iteración 3:

Ingrese el centro de la circunferencia h,k: 90 90

Ingrese área: 5

Los 2 puntos están fuera de la circunferencia ingresada.

Ejemplo 2

Ingrese el primer punto: -10 -10

Ingrese el segundo punto: 10 10

Iteración 1:

Ingrese el centro de la circunferencia h,k: -11 -11

Ingrese área: 2

Los 2 puntos están fuera de la circunferencia ingresada.

2.3.11. Rectas no paralelas (Adaptado del Laboratorio 6 – 2020-2)

Se solicita implementar un programa en lenguaje C, que permita al usuario ingresar dos puntos (x,y) y calcular la recta que pasa por esos 2 puntos. En seguida, el programa deberá volver a pedir otros dos puntos y volver a calcular la nueva recta que pasa por estos 2 puntos nuevos. Repetir el proceso hasta que el usuario ingrese rectas 2 paralelas (es decir, el programa terminará cuando se ingresen 2 puntos que correspondan a una recta paralela a la recta que pasaba por los 2 puntos ingresados anteriormente).

Asumir que el usuario nunca ingresa rectas verticales (siempre ingresa rectas ascendentes, descendentes u horizontales).

Solo se debe trabajar con rectas consecutivas, es decir, solo comparará la recta ingresada, con la recta anterior (y no con todas las rectas ingresadas en el pasado).

Si se ingresan 2 puntos que pertenecen a la misma recta que pasa por los 2 puntos ingresados anteriormente, entonces no es se trata de rectas paralelas, sino de la misma recta. En ese caso, el programa no debe finalizar.

Se deberán usar los siguientes módulos:

- Usar un módulo para hallar una recta a través de 2 puntos. Este módulo recibirá como parámetros dos puntos (x1, y1) y (x2, y2) y permitirá obtener la pendiente de dicha recta, así como también su ordenada en el origen.
- Un módulo para determinar si 2 rectas son paralelas o no. Este módulo deberá recibir dos pendientes m1 y m2 y dos puntos de intersección en el eje de ordenadas b1 y b2 y retornar 1 si es que son paralelas y 0 si es que no lo son. Debido a la imprecisión de decimales, para evaluar si 2 rectas son paralelas no se deberá comparar directamente las pendientes de ambas rectas, sino que se deberá hacer uso de un error ingresado por teclado (por ejemplo, 0.01, 0.0001, 0.000001, etc.). Este error se deberá solicitar al comienzo de la ejecución del programa y deberá ser usado en todas las comparaciones (leer siguiente viñeta para más detalle).
- Se deberá validar que el error sea un número positivo. Si se ingresa un valor no positivo, el programa deberá imprimir el mensaje “El error debe ser positivo y finalizar su ejecución inmediatamente”. Dicha validación se realizará mediante un módulo que retorne 1 si es que el error es positivo y retorne 0 si es que no lo es.

La solución natural de este ejercicio es mediante el uso de una estructura algorítmica iterativa con salida controlada, pues se debe realizar por lo menos 1 iteración para ingresar el primer par de rectas a comparar. Resolver este ejercicio con una estructura algorítmica iterativa con entrada controlada será considerado como error.

A continuación, se muestran algunos ejemplos de ejecución: Ejemplo 1

Ingrese el error: 0.001

Iteración 1:

Ingrese el primer punto: 1 4

Ingrese el segundo punto: 5 2

La pendiente es: -0.500000

La b es: 4.500000

Iteración 2:

Ingrese el primer punto: 4 7

Ingrese el segundo punto: 8 5

La pendiente es: -0.500000

La b es: 9.000000

Se ingresaron dos rectas paralelas

Ejemplo 2

Ingrese el error: 0.01

Iteración 1:

Ingrese el primer punto: 1 4

Ingrese el segundo punto: 5 2

La pendiente es: -0.500000

La b es: 4.500000

Iteración 2:

Ingrese el primer punto: 1 4

Ingrese el segundo punto: 5 2

La pendiente es: -0.500000

La b es: 4.500000

Iteración 3:

Ingrese el primer punto: 1 4

Ingrese el segundo punto: 5 2

La pendiente es: -0.500000

La b es: 4.500000

Iteración 4:

Ingrese el primer punto: 2 2

Ingrese el segundo punto: 4 4

La pendiente es: 1.000000

La b es: 0.000000

Iteración 5:

Ingrese el primer punto: 3 2

Ingrese el segundo punto: 5 4

La pendiente es: 1.000000

La b es: -1.000000

Se ingresaron dos rectas paralelas

Ejemplo 3

Ingrese el error: -5

El error debe ser positivo.

2.3.12. Perforación de triángulo (Adaptado del Laboratorio 6 – 2020-2)

Implementar un programa en lenguaje C que solicite al usuario el ingreso de los tres vértices (x,y) de un triángulo equilátero ABC, y calcule las distancias AB, BC y AC (es decir, la longitud de los lados del triángulo).

Si las 3 distancias son iguales (para esto se deberá comparar las distancias de dos en dos, calculando si la diferencia entre 2 de los lados es menor o igual a 0.001, tomando como distancias iguales a aquellas cuya diferencia es menor o igual a 0.001), entonces se deberá proceder de la siguiente manera:

- Calcular al área del triángulo equilátero y, en seguida, perforarlo en su interior quitándole el triángulo medial (el triángulo medial de un triángulo ABC es aquel triángulo cuyos vértices están situados en los puntos medios de los lados AB, BC y AC).
- El término perforar, se refiere a eliminar toda el área contenida dentro del triángulo medial, y mantener únicamente el área correspondida dentro del triángulo ABC, pero fuera del triángulo medial. Es decir, quedarán las siguientes 3 áreas:
 - El área comprendida en el nuevo triángulo formado entre vértice A, el punto medio del lado AB y el punto medio del lado AC.
 - El área comprendida en el nuevo triángulo formado entre vértice B, el punto medio del lado AB y el punto medio del lado BC.
 - o El área comprendida en el nuevo triángulo formado entre vértice C, el punto medio del lado AC y el punto medio del lado BC.
- Esto quiere decir que el área comprendida en triángulo formado por los puntos medios de AB, BC y AC ha sido retirada de la figura.
- Con cada uno de los nuevos 3 triángulos formados, repetir el mismo proceso (perforándoles el triángulo medial a cada uno de ellos).
- Repetir este proceso hasta que el área total de los triángulos conservados sea un valor muy cercano a cero (más adelante se detallará cómo verificar si un valor es cercano a cero o no). Con esto se demuestra que al perforar repetidas veces un triángulo equilátero, el área resultante tiende a ser 0.

Para calcular el área de cada uno de los triángulos conservados, se deberá usar la siguiente fórmula:

$$area = \frac{lado^2 \times \sqrt{3}}{4}$$

Para determinar cuándo se dice que el área total de los triángulos mantenidos (es decir, la suma de las áreas de todos los triángulos mantenidos, o, lo cual es lo mismo que el área total del triángulo inicial menos las áreas de todos los triángulos mediales retirados) es cercana a cero, se deberá hacer uso de un error ingresado por teclado, por ejemplo, 0.01, 0.0001, 0.000005, etc. Este error deberá ser ingresado al inicio del ejercicio. El error debe ser positivo. En caso no lo sea, el programa deberá emitir el mensaje “El error debe ser positivo” y finalizar su ejecución inmediatamente.

Se deberá hacer uso de los siguientes módulos:

- Un módulo que valide que el error sea positivo. Este módulo retornará 1 si el error es positivo y retornará 0 si no lo es.
- Un módulo para calcular la distancia entre dos puntos. Este módulo deberá ser llamado tres veces para poder calcular las tres distancias AB, BC y AC, antes de verificar que el triángulo ABC sea equilátero.
- Un módulo que verifique que un triángulo es equilátero. Este módulo recibirá como parámetro los 3 lados de un triángulo, y validará que los tres lados sean casi iguales, para esto, hará comparará los lados de dos en dos, y verificará que la diferencia entre cada pareja de lados sea menor o igual a 0.001. Si las 3 diferencias son menores o iguales a 0.001, este módulo retornará 1, si alguna de las diferencias es superior a 0.001, este módulo retornará 0.
- Un módulo que, calcule el área total de los triángulos conservados. Este módulo deberá primero calcular el área del triángulo inicial ABC y, luego, mediante el uso de una estructura iterativa con salida controlada, calcular el área total resultante después de realizar las perforaciones de la siguiente manera:
 - En la primera iteración, existe un solo triángulo, por lo tanto, se hará 1 sola perforación (tener en cuenta que el triángulo medial perforado tiene $1/3$ del área del triángulo ABC). Al final de esta iteración, habrá 3 triángulos.
 - En la segunda iteración, se realizarán 3 perforaciones, pues hay 3 triángulos:
 - El triángulo formado entre vértice A, el punto medio del lado AB y el punto medio del lado AC.
 - El triángulo formado entre vértice B, el punto medio del lado AB y el punto medio del lado BC.
 - El triángulo formado entre vértice C, el punto medio del lado AC y el punto medio del lado BC.Las áreas de cada una de estas 3 perforaciones corresponderán a $1/3$ del área de cada uno de estos 3 triángulos. Al final de esta iteración, habrá 9 triángulos.
- En la tercera iteración se harán 9 perforaciones, pues hay 9 triángulos (es decir, a los 9 triángulos se les borrará su triángulo medial, formando 3 triángulos nuevos cada uno). Cada uno de los 9 triángulos mediales eliminados tienen $1/3$ del área de cada uno de los 9 triángulos que existían al comienzo de esta iteración.

Nota:

Si cada uno de los 9 triángulos mediales tiene $1/3$ del área de cada uno de los 9 triángulos completos, entonces, la suma del área de los 9 triángulos mediales borra dos será $1/3$ de la suma de las áreas de los 9 triángulos completos.

Seguir iterando hasta que el área resultante sea casi cero (es decir, hasta que sea menor o igual al error ingresado). **Mensajes de error:**

- Si los tres lados del triángulo no son casi iguales, se deberá imprimir el mensaje “El triángulo debe ser equilátero”, y finalizar la ejecución inmediatamente.
- Si el error ingresado no es positivo, se deberá imprimir el mensaje “El error debe ser positivo”, y finalizar la ejecución inmediatamente.

Al final, el programa deberá imprimir el mensaje “El área es lo suficientemente cercana a cero:” y se deberá imprimir el área con 16 decimales.

La solución natural de este ejercicio es mediante el uso de una estructura algorítmica iterativa con salida controlada, pues se debe realizar por lo menos 1 iteración para realizar la primera perforación al triángulo equilátero. Resolver este ejercicio con una estructura algorítmica iterativa con entrada controlada será considerado como error.

A continuación, se muestran algunos ejemplos de ejecución: Ejemplo 1

Ingrese el punto A: 0 0
Ingrese el punto B: 3 0
Ingrese el punto C: 3 4
El triángulo debe ser equilátero

Ejemplo 2

Ingrese el punto A: 1 1
Ingrese el punto B: 5 1
Ingrese el punto C: 3 4.465
Ingrese el error: -5
El error debe ser positivo

Ejemplo 3

Ingrese el punto A: 1 1
Ingrese el punto B: 5 1
Ingrese el punto C: 3 4.465
Ingrese el error: 0.0000001
El área es lo suficientemente cercana a cero: 0.0000000932137999

Ejemplo 4

Ingrese el punto A: 1 2
Ingrese el punto B: 3 4
Ingrese el punto C: 0.268 4.732
Ingrese el error: 0.00001
El área es lo suficientemente cercana a cero: 0.0000082669912060

2.3.13. Cubos y Prismas (Adaptado del Laboratorio 6 – 2021-1)

En geometría, un prisma es un poliedro cuya superficie está formada por dos caras iguales y paralelas llamadas bases y por caras laterales (tantas como lados tienen las bases) que son paralelogramos. Además, un poliedro regular es aquella figura geométrica de tres dimensiones cuyas caras son todas iguales y, además, son polígonos regulares. Esto quiere decir que un poliedro regular está conformado por polígonos idénticos, cada uno de los cuales, a su vez, cumple con la condición de ser regular. Es decir, todos sus lados y sus ángulos interiores miden lo mismo.

Existen 5 tipos de poliedros regulares y diversos tipos de prismas, pero en esta ocasión trabajaremos con los cubos y con los prismas cuadrangulares:

- **Cubo.-** Es una figura de seis lados formada por cuadrados idénticos. Cabe recordar que un cuadrado es un cuadrilátero regular, específicamente, un paralelogramo. Se caracteriza porque sus cuatro lados miden igual y sus ángulos interiores también son todos iguales y rectos (miden 90°).
- **Prisma cuadrangular regular.-** Es aquel que tiene como bases dos cuadrados. Sus caras laterales son 4 rectángulos iguales. Para recordar, un cuadrado es un polígono con 4 lados y ángulos iguales (todos sus ángulos son de 90°).

En esta ocasión se pide elaborar un programa en lenguaje C que permita evaluar cubos y prismas cuadrangulares regulares, para ello debe leer un carácter que representa a un tipo de poliedro (Cubo (C) y Prisma Cuadrangular (P)) y permita evaluar si el poliedro a procesar es un cubo o un prisma cuadrangular regular.

Para verificar si se trata de un poliedro regular, en el caso del cubo, debe solicitar que ingrese por cada cara del poliedro, los cuatro puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ y $D(x_4, y_4)$ del plano cartesiano que conforman el cuadrado que se encuentra en dicha cara y verificar que en verdad se trate de un cuadrado. Considerar que la validación solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD y DA.

Para verificar si se trata de un prisma cuadrangular regular debe tener en cuenta lo siguiente:

- Debe solicitar que ingrese por cada cara lateral del prisma los cuatro puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ y $D(x_4, y_4)$ del plano cartesiano que conforman el rectángulo que se encuentra en dicha cara lateral y verificar que se trate de un rectángulo.
- Considerar que la validación del rectángulo solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD y DA. Debe considerar como lados paralelos el AB y CD, así como el BC y DA. Además recuerde que en un rectángulo los lados paralelos tienen la misma medida.
- Debe solicitar que ingrese, por cada base del prisma, los 4 puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ y $D(x_4, y_4)$ del plano cartesiano que forman la base y verificar si se trata de un cuadrado.
- Considerar que la validación del cuadrado solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD y DA.

En esta pregunta se deben mostrar mensajes específicos ante las siguientes situaciones:

- Al ingresar el tipo de poliedro, debe verificar que se ingresen solamente las letras C y P en mayúsculas o minúsculas. En caso no se cumpla, se deberá emitir el siguiente mensaje **La opción de poliedro ingresado no corresponde a un Cubo o un prisma cuadrangular regular** y el programa debe terminar.
- Al calcular los lados del cuadrado que pertenecen a una cara del cubo, se debe validar que los cuatro lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un cuadrado** y se continúa evaluando la siguiente cara.
- Al finalizar la validación de si se trata de un cubo, debe mostrar el siguiente mensaje **El cubo ingresado es válido** en caso el cubo sea válido. En caso el cubo sea inválido, debe mostrar el siguiente mensaje **El cubo ingresado no es válido** y además debe indicar la cantidad de caras cuyos puntos ingresados no formaron un cuadrado durante la validación.
- Al calcular los lados del rectángulo que pertenecen a una cara lateral del prisma cuadrangular regular, se debe validar que efectivamente se trate de un rectángulo. En caso no se cumpla, deberá emitir el siguiente mensaje **Los puntos ingresados no forman un rectángulo** y se continúa evaluando la siguiente cara.
- Al calcular los lados del cuadrado que pertenecen a una base del prisma cuadrangular, se debe validar que los cuatro lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un cuadrado** y se continúa evaluando la siguiente base.
- Al finalizar la validación de si se trata de un prisma cuadrangular regular, debe mostrar el siguiente mensaje **El prisma cuadrangular es válido** en caso el prisma sea válido. En caso que el prisma regular sea inválido, debe mostrar el siguiente mensaje **El prisma cuadrangular no es válido** y además debe indicar el motivo por el cual no es un prisma cuadrangular regular válido (laterales y/o base).

El programa debe estar desarrollado bajo el paradigma del programación modular, por lo que debe desarrollar y considerar únicamente los siguientes módulos dentro de su solución:

- Un módulo main
- Un módulo que permita validar si un prisma cuadrangular regular es válido o no. Para ello, el módulo debe devolver, como parámetros que se modifican luego de la invocación, si el prisma es válido, si sus caras laterales son correctas y si sus bases son correctas.

- Un módulo que permita procesar las bases de un prisma y determine si el prisma, en sus bases es válido o no. Para ello, el módulo debe recibir como parámetro el número de bases que tiene el prisma y debe devolver si todas las bases procesadas son válidas o no. En este módulo debe utilizar **una** estructura iterativa con salida controlada para procesar las bases del prisma.
- Un módulo que permita procesar las caras laterales de un prisma regular y determine si el prisma, en sus caras laterales es válido o no. Para ello, el módulo debe recibir como parámetro la cantidad de caras laterales que va a procesar y debe devolver si todas las caras laterales procesadas son válidas o no. En este módulo debe utilizar **una** estructura iterativa con salida controlada para procesar las caras laterales del prisma .
- Un módulo que permita determinar si cuatro puntos forman un rectángulo o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los cuatro puntos ingresados en una cara lateral y debe devolver si dichos puntos forman un rectángulo.
- Un módulo que permita determinar si cuatro puntos forman un cuadrado o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los cuatro puntos ingresados en una base y debe devolver si dichos puntos forman un cuadrado.
- Un módulo que permita procesar las caras de un cubo y determine si el cubo es válido o no, además debe determinar la cantidad de caras cuyos puntos ingresados no formaron un cuadrado. Para ello, el módulo debe devolver como parámetros que se modifican luego de la invocación, si el cubo es válido o no y la cantidad de caras cuyos puntos ingresados no formaron un cuadrado. En este módulo debe utilizar una estructura iterativa con salida controlada para procesar las caras del cubo.
- Un módulo que permita calcular la distancia que existe entre dos puntos del plano cartesiano. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de dos puntos y debe devolver la distancia que existe entre ellos. Recuerde que la fórmula para calcular la distancia entre dos puntos es la siguiente:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Comparación de números reales

Muchas veces el resultado de la comparación de números reales a través de la igualdad no es el deseado. Esto sucede por la forma en que se representa internamente los reales, basta que exista una pequeña diferencia de precisión para que no se de la igualdad. En este caso es recomendable usar el valor absoluto de la diferencia de los números que se desean comparar. Para este problema, si esta diferencia es menor o igual a 0.01, se puede asumir que son iguales.

Caso de prueba 1

Evalutando Cubos y Prismas Cuadrangulares

Ingrese el tipo de poliedro a evaluar (C-Cubo, P-Prisma Cuadrangular): j

La opción del poliedro ingresado no corresponde a un cubo o un prisma cuadrangular regular

Caso de prueba 2

Evalutando Cubos y Prismas Cuadrangulares

Ingrese el tipo de poliedro a evaluar (C-Cubo, P-Prisma Cuadrangular): c

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 1

Coordenada x e y del punto A: 1.5 4.5

Coordenada x e y del punto B: 6.5 4.5

Coordenada x e y del punto C: 6.5 -0.5

Coordenada x e y del punto D: 1.5 -0.5

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 2

Coordenada x e y del punto A: 1 4

Coordenada x e y del punto B: 6 4

Coordenada x e y del punto C: 6 0

Coordenada x e y del punto D: 1 0

Los puntos ingresados no forman un cuadrado

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 3

Coordenada x e y del punto A: 1 4

Coordenada x e y del punto B: 6 4

Coordenada x e y del punto C: 6 -1

Coordenada x e y del punto D: 1 -1

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 4

Coordenada x e y del punto A: 5.2 8.5

Coordenada x e y del punto B: 2.5 3.6

Coordenada x e y del punto C: 4.5 2.1

Coordenada x e y del punto D: 1 0

Los puntos ingresados no forman un cuadrado

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 5

Coordenada x e y del punto A: 5.2 8.5

Coordenada x e y del punto B: 2.5 3.6

Coordenada x e y del punto C: 4.5 2.1

Coordenada x e y del punto D: 3 0

Los puntos ingresados no forman un cuadrado

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 6

Coordenada x e y del punto A: 4.2 7.5

Coordenada x e y del punto B: 3.5 4.6

Coordenada x e y del punto C: 2.5 3.1

Coordenada x e y del punto D: 3 0

Los puntos ingresados no forman un cuadrado

El cubo ingresado no es válido.

La cantidad de caras que no son cuadrados es 4

Caso de prueba 3

Evalutando Cubos y Prismas Cuadrangulares

Ingrese el tipo de poliedro a evaluar (C-Cubo, P-Prisma Cuadrangular): C

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 1

Coordenada x e y del punto A: 1.5 4.5

Coordenada x e y del punto B: 6.5 4.5

Coordenada x e y del punto C: 6.5 -0.5

Coordenada x e y del punto D: 1.5 -0.5

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 2

Coordenada x e y del punto A: 1 4

Coordenada x e y del punto B: 6 4

Coordenada x e y del punto C: 6 -1

Coordenada x e y del punto D: 1 -1

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 3

Coordenada x e y del punto A: 3 6

Coordenada x e y del punto B: 8 6

Coordenada x e y del punto C: 8 1

Coordenada x e y del punto D: 3 1

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 4

Coordenada x e y del punto A: 4.5 7.5

Coordenada x e y del punto B: 9.5 7.5

Coordenada x e y del punto C: 9.5 2.5

Coordenada x e y del punto D: 4.5 2.5

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 5

Coordenada x e y del punto A: 5 8

Coordenada x e y del punto B: 10 8

Coordenada x e y del punto C: 10 3

Coordenada x e y del punto D: 5 3

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 6

Coordenada x e y del punto A: 4.2 7.2

Coordenada x e y del punto B: 9.2 7.2

Coordenada x e y del punto C: 9.2 2.2

Coordenada x e y del punto D: 4.2 2.2

El cubo ingresado es válido.

Caso de prueba 4

Evalutando Cubos y Prismas Cuadrangulares

Ingrese el tipo de poliedro a evaluar (C-Cubo, P-Prisma Cuadrangular): P

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 2

Ingrese las coordenadas x e y del punto B: 3 2

Ingrese las coordenadas x e y del punto C: 3 0

Ingrese las coordenadas x e y del punto D: 1 0

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 1.5 2.5

Ingrese las coordenadas x e y del punto B: 3.5 2.5

Ingrese las coordenadas x e y del punto C: 3.5 0.5

Ingrese las coordenadas x e y del punto D: 1.5 0.5

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 6 1

Ingrese las coordenadas x e y del punto C: 6 -1

Ingrese las coordenadas x e y del punto D: 4 -1

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 4.5 1.5

Ingrese las coordenadas x e y del punto B: 6.5 1.5

Ingrese las coordenadas x e y del punto C: 6.5 -1.5

Ingrese las coordenadas x e y del punto D: 4.5 -1.5

Validaremos la base 1

Ingrese las coordenadas x e y del punto A: 4.2 7.2

Ingrese las coordenadas x e y del punto B: 9.2 7.2

Ingrese las coordenadas x e y del punto C: 9.2 2.2

Ingrese las coordenadas x e y del punto D: 4.2 2.2

Validaremos la base 2

Ingrese las coordenadas x e y del punto A: 5 8

Ingrese las coordenadas x e y del punto B: 10 8

Ingrese las coordenadas x e y del punto C: 10 3

Ingrese las coordenadas x e y del punto D: 5 3

El prisma cuadrangular regular es válido

Caso de prueba 5

Evalutando Cubos y Prismas Cuadrangulares

Ingrese el tipo de poliedro a evaluar (C-Cubo, P-Prisma Cuadrangular): p

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 3

Ingrese las coordenadas x e y del punto B: 3 2

Ingrese las coordenadas x e y del punto C: 3 0

Ingrese las coordenadas x e y del punto D: 1 0

Los puntos ingresados no forman un rectángulo

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 1.5 2.5

Ingrese las coordenadas x e y del punto B: 3.5 4.5

Ingrese las coordenadas x e y del punto C: 3.5 0.5

Ingrese las coordenadas x e y del punto D: 1.5 0.5

Los puntos ingresados no forman un rectángulo

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 6 1

Ingrese las coordenadas x e y del punto C: 6 -1

Ingrese las coordenadas x e y del punto D: 4 -1

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 4.5 1.5

Ingrese las coordenadas x e y del punto B: 6.5 1.5

Ingrese las coordenadas x e y del punto C: 6.5 -1.5

Ingrese las coordenadas x e y del punto D: 4.5 -1.5

Validaremos la base 1

Ingrese las coordenadas x e y del punto A: 4.2 7.2

Ingrese las coordenadas x e y del punto B: 9.2 7.2

Ingrese las coordenadas x e y del punto C: 9.2 2.2

Ingrese las coordenadas x e y del punto D: 4.2 2.2

Validaremos la base 2

Ingrese las coordenadas x e y del punto A: 5 8

Ingrese las coordenadas x e y del punto B: 10 8

Ingrese las coordenadas x e y del punto C: 10 3

Ingrese las coordenadas x e y del punto D: 5 3

El prisma cuadrangular regular no es válido debido a:

Al menos un rectángulo lateral no es correcto

2.3.14. Pirámide cuadrangular y pentagonal regular (Adaptado del Laboratorio 6 - 2021-1)

En geometría, una pirámide es un poliedro cuya superficie está formada por una base que es un polígono cualquiera y caras laterales triangulares que confluyen en un vértice que se denomina ápice (o vértice de la pirámide). Las pirámides tienen tantos triángulos en las caras laterales como lados tiene la base.

Las pirámides se pueden clasificar según el número de lados que tiene su base, pero en esta ocasión solo veremos 2 de ellas, las cuales son:

- **Pirámide cuadrangular regular.-** Es aquella que tiene como base un cuadrado. Sus caras laterales son 4 triángulos isósceles e iguales entre sí. Para recordar, un triángulo es isósceles cuando dos de sus lados son iguales y el otro desigual.
- **Pirámide pentagonal regular.-** Es aquella que tiene como base un pentágono regular. Sus caras laterales son 5 triángulos isósceles e iguales entre sí. Para recordar, un pentágono regular es un polígono con cinco lados y ángulos iguales (todos sus ángulos son de 108°).

En esta ocasión se le pide elaborar un programa en lenguaje C que permita evaluar pirámides cuadrangulares y pentagonales regulares, para ello debe leer un carácter que representa a un tipo de pirámide regular (Cuadrangular (C) y Pentagonal (P)) y permita evaluar si la pirámide es regular.

Para verificar si se trata de una pirámide cuadrangular regular debe tener en cuenta lo siguiente:

- Debe solicitar que ingrese por cada cara lateral de la pirámide los tres puntos $A(x_1, y_1)$, $B(x_2, y_2)$ y $C(x_3, y_3)$ del plano cartesiano que conforman el triángulo que se encuentra en dicha cara lateral y verificar que se trate de un triángulo isósceles.
- Considerar que la validación del triángulo isósceles solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC y CA.
- Debe solicitar que ingrese la base de la pirámide los 4 puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ y $D(x_4, y_4)$ del plano cartesiano que conforman el cuadrado y verificar que se trate de un cuadrado.
- Considerar que la validación del cuadrado solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD y DA.

Para verificar si se trata de una pirámide pentagonal regular debe tener en cuenta lo siguiente:

- Debe solicitar que ingrese por cada cara lateral de la pirámide los tres puntos $A(x_1, y_1)$, $B(x_2, y_2)$ y $C(x_3, y_3)$ del plano cartesiano que conforman el triángulo que se encuentra en dicha cara lateral y verificar que se trate de un triángulo isósceles.
- Considerar que la validación del triángulo isósceles solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC y CA.
- Debe solicitar que ingrese la base de la pirámide los 5 puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$, $D(x_4, y_4)$ y $E(x_5, y_5)$ del plano cartesiano que conforman el pentágono y verificar que se trate de un pentágono regular.
- Considerar que la validación del pentágono regular solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD, DE y EA.

En esta pregunta se deben mostrar mensajes específicos ante las siguientes situaciones:

- Al ingresar el tipo de pirámide regular, debe verificar que se ingresen solamente las letras C y P en mayúsculas. En caso no se cumpla, se deberá emitir el siguiente mensaje La opción de pirámide regular ingresada no corresponde a un cuadrangular o pentagonal y el programa debe terminar.

- Al calcular los lados del triángulo isósceles que pertenecen a una cara lateral de la pirámide, se debe validar que efectivamente se trate de un triángulo isósceles. En caso no se cumpla, deberá emitir el siguiente mensaje **Los puntos ingresados no forman un triángulo isósceles** y se continúa evaluando la siguiente cara.
- Al calcular los lados del cuadrado que pertenece a la base de la pirámide cuadrangular regular, se debe validar que los cuatro lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un cuadrado**.
- Al calcular los lados del pentágono que pertenece a la base de la pirámide pentagonal regular, se debe validar que los cinco lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un pentágono regular**.
- Al finalizar la validación de si se trata de una pirámide regular, debe mostrar el siguiente mensaje **La pirámide cuadrangular regular es válida** o **La pirámide pentagonal regular es válida** en caso la pirámide regular sea válida. En caso que la pirámide regular sea inválida, debe mostrar el siguiente mensaje **La pirámide cuadrangular regular no es válida** o **La pirámide pentagonal regular no es válida** según corresponda y además debe indicar el motivo por el cuál no es una pirámide regular válida (laterales y/o base).

El programa debe estar desarrollado bajo el paradigma de programación modular, por lo que debe desarrollar y considerar únicamente los siguientes módulos dentro de su solución:

- Un módulo **main**
- Un módulo que permita procesar las caras laterales de una pirámide regular y determine si la pirámide, en sus caras laterales es válida o no. Para ello, el módulo debe recibir como parámetro el tipo de pirámide y devolver, como parámetros que se modifican luego de la invocación, si las caras laterales de la pirámide son válidas y la cantidad de laterales incorrectas. En este módulo debe utilizar **una** estructura iterativa con salida controlada para procesar las caras laterales de la pirámide regular.
- Un módulo que permita procesar la base de una pirámide y determine si la pirámide, en su base es válida o no. Para ello, el módulo debe recibir como parámetro el tipo de pirámide y devolver si la base es válida o no.
- Un módulo que permita leer las coordenadas x e y de un punto del plano cartesiano. Para ello, el módulo debe devolver como parámetros que se modifican luego de la invocación, los valores de las coordenadas x e y del punto.
- Un módulo que permita determinar si cuatro puntos forman un cuadrado o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los cuatro puntos ingresados en la base de la pirámide cuadrangular y debe devolver si dichos puntos forman un cuadrado.
- Un módulo que permita determinar si cinco puntos forman un pentágono regular o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los cinco puntos ingresados en la base de la pirámide pentagonal regular y debe devolver si dichos puntos forman un pentágono regular.
- Un módulo que permita determinar si tres puntos forman un triángulo isósceles o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los tres puntos ingresados en una cara lateral y debe devolver si dichos puntos forman un triángulo isósceles.
- Un módulo que permita calcular la distancia que existe entre dos puntos del plano cartesiano. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de dos puntos y debe devolver la distancia que existe entre ellos. Recuerde que la fórmula para calcular la distancia entre dos puntos es la siguiente:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Comparación de números reales

Muchas veces el resultado de la comparación de números reales a través de la igualdad no es el deseado. Esto sucede por la forma en que se representa internamente los reales, basta que exista una pequeña diferencia de precisión para que no se de la igualdad. En este caso es recomendable usar el valor absoluto de la diferencia de los números que se desean comparar. Para este problema, si esta diferencia es menor o igual a 0.01, se puede asumir que son iguales.

Caso de prueba 1

Evalutando Pirámides Cuadrangulares y Pentagonales regulares

Ingrese el tipo de pirámide a evaluar (C-Cuadrangular, P-Pentagonal): c

La opción de la pirámide ingresada no corresponde a una cuadrangular o pentagonal.

Caso de prueba 2

Evalutando Pirámides Cuadrangulares y Pentagonales regulares

Ingrese el tipo de pirámide a evaluar (C-Cuadrangular, P-Pentagonal): C

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 1

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 6 -4

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 2 1

Ingrese las coordenadas x e y del punto B: 6 3

Ingrese las coordenadas x e y del punto C: 7 -4

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 3 1

Ingrese las coordenadas x e y del punto B: 7 3

Ingrese las coordenadas x e y del punto C: 8 -4

Validaremos la base

Ingrese las coordenadas x e y del punto A: 1.5 4.5

Ingrese las coordenadas x e y del punto B: 6.5 4.5

Ingrese las coordenadas x e y del punto C: 6.5 -0.5

Ingrese las coordenadas x e y del punto D: 1.5 -0.5

La pirámide cuadrangular regular es válido.

Caso de prueba 3

Evalutando Pirámides Cuadrangulares y Pentagonales regulares

Ingrese el tipo de pirámide a evaluar (C-Cuadrangular, P-Pentagonal): C

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 3 1

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 6 -4

Los puntos ingresados no forman un triángulo isósceles

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 5.5 1

Ingrese las coordenadas x e y del punto B: 6 3

Ingrese las coordenadas x e y del punto C: 7 -4

Los puntos ingresados no forman un triángulo isósceles

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 3 1

Ingrese las coordenadas x e y del punto B: 7 3

Ingrese las coordenadas x e y del punto C: 8 -4

Validaremos la base

Ingrese las coordenadas x e y del punto A: 1.5 4.5

Ingrese las coordenadas x e y del punto B: 6.5 4.5

Ingrese las coordenadas x e y del punto C: 6.5 -0.5

Ingrese las coordenadas x e y del punto D: 1.5 -0.5

La pirámide cuadrangular regular no es válida debido a:

Hay 2 caras laterales incorrectas.

Caso de prueba 4

Evalutando Pirámides Cuadrangulares y Pentagonales regulares

Ingrese el tipo de pirámide a evaluar (C-Cuadrangular, P-Pentagonal): P

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 1

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 6 -4

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 2 1

Ingrese las coordenadas x e y del punto B: 6 3

Ingrese las coordenadas x e y del punto C: 7 -4

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 3 1

Ingrese las coordenadas x e y del punto B: 7 3

Ingrese las coordenadas x e y del punto C: 8 -4

Validaremos la cara lateral 5

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 8 3

Ingrese las coordenadas x e y del punto C: 9 -4

Validaremos la base

Ingrese las coordenadas x e y del punto A: 4.2 6.75

Ingrese las coordenadas x e y del punto B: 8.14 9.73

Ingrese las coordenadas x e y del punto C: 12.19 6.91

Ingrese las coordenadas x e y del punto D: 10.76 2.18

Ingrese las coordenadas x e y del punto E: 5.82 2.08

La pirámide pentagonal regular es válida.

Caso de prueba 5

Evalutando Pirámides Cuadrangulares y Pentagonales regulares

Ingrese el tipo de pirámide a evaluar (C-Cuadrangular, P-Pentagonal): P

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 1

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 6 -4

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 2 1

Ingrese las coordenadas x e y del punto B: 6 3

Ingrese las coordenadas x e y del punto C: 7 -4

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 5 1

Ingrese las coordenadas x e y del punto B: 7 3.5

Ingrese las coordenadas x e y del punto C: 8 -4

Los puntos ingresados no forman un triángulo isósceles

Validaremos la cara lateral 5

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 8 3

Ingrese las coordenadas x e y del punto C: 9 -4

Validaremos la base

Ingrese las coordenadas x e y del punto A: 4.2 6.75

Ingrese las coordenadas x e y del punto B: 8.14 9.73

Ingrese las coordenadas x e y del punto C: 12.19 6.91

Ingrese las coordenadas x e y del punto D: 12.76 1.18

Ingrese las coordenadas x e y del punto E: 5.82 2.08

Los puntos ingresados no forman un pentágono regular

La pirámide pentagonal regular no es válida debido a:

Hay 1 laterales incorrectos

La base de la pirámide es incorrecta

2.3.15. Pirámide triangular y hexagonal regular (Adaptado del Laboratorio 6 - 2021-1)

En geometría, una pirámide es un poliedro cuya superficie está formada por una base que es un polígono cualquiera y caras laterales triangulares que confluyen en un vértice que se denomina ápice (o vértice de la pirámide). Las pirámides tienen tantos triángulos en las caras laterales como lados tiene la base.

Las pirámides se pueden clasificar según el número de lados que tiene su base, pero en esta ocasión solo veremos 2 de ellas, las cuales son:

- **Pirámide triangular regular.-** Es aquella que tiene como base un triángulo equilátero. Sus caras laterales son 3 triángulos isósceles e iguales entre sí. Para recordar, un triángulo es isósceles cuando dos de sus lados son iguales y el otro desigual.
- **Pirámide hexagonal regular.-** Es aquella que tiene como base un hexágono regular. Sus caras laterales son 6 triángulos isósceles e iguales entre sí. Para recordar, un hexágono regular es un polígono con seis lados y ángulos iguales (todos sus ángulos son de 108°).

En esta ocasión se le pide elaborar un programa en lenguaje C que permita evaluar pirámides triangular y

hexagonal regulares, para ello debe leer un caracter que representa a un tipo de pirámide regular (Triangular (T) y Hexagonal (H)) y permita evaluar si la pirámide es regular.

Para verificar si se trata de una pirámide triangular regular debe tener en cuenta lo siguiente:

- Debe solicitar que ingrese por cada cara lateral de la pirámide los tres puntos $A(x_1, y_1)$, $B(x_2, y_2)$ y $C(x_3, y_3)$ del plano cartesiano que conforman el triángulo que se encuentra en dicha cara lateral y verificar que se trate de un triángulo isósceles.
- Considerar que la validación del triángulo isósceles solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC y CA.
- Debe solicitar que ingrese la base de la pirámide triangular los 3 puntos $A(x_1, y_1)$, $B(x_2, y_2)$ y $C(x_3, y_3)$ del plano cartesiano que conforman el triángulo equilátero y verificar que se trate de un triángulo equilátero.
- Considerar que la validación del triángulo equilátero solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC y CA.

Para verificar si se trata de una pirámide hexagonal regular debe tener en cuenta lo siguiente:

- Debe solicitar que ingrese por cada cara lateral de la pirámide los tres puntos $A(x_1, y_1)$, $B(x_2, y_2)$ y $C(x_3, y_3)$ del plano cartesiano que conforman el triángulo que se encuentra en dicha cara lateral y verificar que se trate de un triángulo isósceles.
- Considerar que la validación del triángulo isósceles solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC y CA.
- Debe solicitar que ingrese la base de la pirámide los 6 puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$, $D(x_4, y_4)$, $E(x_5, y_5)$ y $F(x_6, y_6)$ del plano cartesiano que conforman el hexágono y verificar que se trate de un hexágono regular.
- Considerar que la validación del hexágono regular solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD, DE, EF y FA.

En esta pregunta se deben mostrar mensajes específicos ante las siguientes situaciones:

- Al ingresar el tipo de pirámide regular, debe verificar que se ingresen solamente las letras T y H en mayúsculas. En caso no se cumpla, se deberá emitir el siguiente mensaje **La opción de pirámide regular ingresada no corresponde a un triangular o hexagonal y el programa debe terminar.**
- Al calcular los lados del triángulo isósceles que pertenecen a una cara lateral de la pirámide, se debe validar que efectivamente se trate de un triángulo isósceles. En caso no se cumpla, deberá emitir el siguiente mensaje **Los puntos ingresados no forman un triángulo isósceles y se continúa evaluando la siguiente cara.**
- Al calcular los lados del triángulo que pertenece a la base de la pirámide triangular regular, se debe validar que los tres lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un triángulo equilátero.**
- Al calcular los lados del hexágono que pertenece a la base de la pirámide hexagonal regular, se debe validar que los seis lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un hexágono regular.**
- Al finalizar la validación de si se trata de una pirámide regular, debe mostrar el siguiente mensaje **La pirámide triangular regular es válida o La pirámide hexagonal regular es válida** en caso la pirámide regular sea válida. En caso que la pirámide regular sea inválida, debe mostrar el siguiente mensaje **La pirámide triangular regular no es válida o La pirámide hexagonal regular no es válida** según corresponda y además debe indicar el motivo por el cual no es una pirámide regular válida (laterales y/o base).

El programa debe estar desarrollado bajo el paradigma de programación modular, por lo que debe desarrollar y considerar únicamente los siguientes módulos dentro de su solución:

- Un módulo main
- Un módulo que permita procesar las caras laterales de una pirámide regular y determine si la pirámide, en sus caras laterales es válida o no. Para ello, el módulo debe recibir como parámetro el tipo de pirámide y devolver, como parámetros que se modifican luego de la invocación, si las caras laterales de la pirámide son válidas y la cantidad de laterales incorrectas. En este módulo debe utilizar **una** estructura iterativa con salida controlada para procesar las caras laterales de la pirámide regular.
- Un módulo que permita procesar la base de una pirámide y determine si la pirámide, en su base es válida o no. Para ello, el módulo debe recibir como parámetro el tipo de pirámide y devolver si la base es válida o no.
- Un módulo que permita leer las coordenadas x e y de un punto del plano cartesiano. Para ello, el módulo debe devolver como parámetros que se modifican luego de la invocación, los valores de las coordenadas x e y del punto.
- Un módulo que permita determinar si tres puntos forman un triángulo equilátero o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los tres puntos ingresados en la base de la pirámide triangular y debe devolver si dichos puntos forman un triángulo equilátero.
- Un módulo que permita determinar si seis puntos forman un hexágono regular o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los seis puntos ingresados en la base de la pirámide hexagonal regular y debe devolver si dichos puntos forman un hexágono regular.
- Un módulo que permita determinar si tres puntos forman un triángulo isósceles o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los tres puntos ingresados en una cara lateral y debe devolver si dichos puntos forman un triángulo isósceles.
- Un módulo que permita calcular la distancia que existe entre dos puntos del plano cartesiano. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de dos puntos y debe devolver la distancia que existe entre ellos. Recuerde que la fórmula para calcular la distancia entre dos puntos es la siguiente:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Comparación de números reales

Muchas veces el resultado de la comparación de números reales a través de la igualdad no es el deseado. Esto sucede por la forma en que se representa internamente los reales, basta que exista una pequeña diferencia de precisión para que no se de la igualdad. En este caso es recomendable usar el valor absoluto de la diferencia de los números que se desean comparar. Para este problema, si esta diferencia es menor o igual a 0.01, se puede asumir que son iguales.

Caso de prueba 1

Evalutando Pirámides Triangulares y Hexagonales regulares

Ingrese el tipo de pirámide a evaluar (T-Triangular, H-Hexagonal): t

La opción de la pirámide ingresada no corresponde a una triangular o hexagonal.

Caso de prueba 2

Evalutando Pirámides Triangulares y Hexagonales regulares

Ingrese el tipo de pirámide a evaluar (T-Triangular, H-Hexagonal): T

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 1

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 6 -4

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 2 1

Ingrese las coordenadas x e y del punto B: 6 3

Ingrese las coordenadas x e y del punto C: 7 -4

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la base

Ingrese las coordenadas x e y del punto A: 1 3

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 3 6.464101

La pirámide triangular regular es válido.

Caso de prueba 3

Evalutando Pirámides Triangulares y Hexagonales regulares

Ingrese el tipo de pirámide a evaluar (T-Triangular, H-Hexagonal): T

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 3 1

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 6 -4

Los puntos ingresados no forman un triángulo isósceles

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 5.5 1

Ingrese las coordenadas x e y del punto B: 6 3

Ingrese las coordenadas x e y del punto C: 7 -4

Los puntos ingresados no forman un triángulo isósceles

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la base

Ingrese las coordenadas x e y del punto A: 1 3

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 3 6.464101

La pirámide triangular regular no es válida debido a:

Hay 2 caras laterales incorrectas.

Caso de prueba 4

Evalutando Pirámides Triangulares y Hexagonales regulares

Ingrese el tipo de pirámide a evaluar (T-Triangular, H-Hexagonal): H

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 1

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 6 -4

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 2 1

Ingrese las coordenadas x e y del punto B: 6 3

Ingrese las coordenadas x e y del punto C: 7 -4

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 3 1

Ingrese las coordenadas x e y del punto B: 7 3

Ingrese las coordenadas x e y del punto C: 8 -4

Validaremos la cara lateral 5

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 8 3

Ingrese las coordenadas x e y del punto C: 9 -4

Validaremos la cara lateral 6

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la base

Ingrese las coordenadas x e y del punto A: -2 3.46

Ingrese las coordenadas x e y del punto B: 0 6.93

Ingrese las coordenadas x e y del punto C: 4 6.93

Ingrese las coordenadas x e y del punto D: 6 3.46

Ingrese las coordenadas x e y del punto E: 4 0

Ingrese las coordenadas x e y del punto F: 0 0

La pirámide hexagonal regular es válida.

Caso de prueba 5

Evalutando Pirámides Triangulares y Hexagonales regulares

Ingrese el tipo de pirámide a evaluar (T-Triangular, H-Hexagonal): H

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 1

Ingrese las coordenadas x e y del punto B: 5 3

Ingrese las coordenadas x e y del punto C: 6 -4

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 2 1

Ingrese las coordenadas x e y del punto B: 6 3

Ingrese las coordenadas x e y del punto C: 7 -4

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 5 1

Ingrese las coordenadas x e y del punto B: 7 3.5

Ingrese las coordenadas x e y del punto C: 8 -4

Los puntos ingresados no forman un triángulo isósceles

Validaremos la cara lateral 5

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 8 3

Ingrese las coordenadas x e y del punto C: 9 -4

Validaremos la cara lateral 6

Ingrese las coordenadas x e y del punto A: 2.5 1

Ingrese las coordenadas x e y del punto B: 6.5 3

Ingrese las coordenadas x e y del punto C: 7.5 -4

Validaremos la base

Ingrese las coordenadas x e y del punto A: 4.2 6.75

Ingrese las coordenadas x e y del punto B: 8.14 9.73

Ingrese las coordenadas x e y del punto C: 12.19 6.91

Ingrese las coordenadas x e y del punto D: 12.76 1.18

Ingrese las coordenadas x e y del punto E: 5.82 2.08

Ingrese las coordenadas x e y del punto F: 7.82 2.08

Los puntos ingresados no forman un hexágono regular

La pirámide hexagonal regular no es válida debido a:

Hay 1 laterales incorrectos

La base de la pirámide es incorrecta

2.3.16. Prisma pentagonal y hexagonal regular (Adaptado del Laboratorio 6 - 2021-1)

En geometría, un prisma es un poliedro cuya superficie está formada por dos caras iguales y paralelas llamadas bases y por caras laterales (tantas como lados tienen las bases) que son paralelogramos.

Los prismas se pueden clasificar según el número de lados que tienen sus bases, pero en esta ocasión solo veremos 2 de ellos, los cuales son:

- **Prisma pentagonal regular.-** Es aquel que tiene como bases dos pentágonos regulares. Sus caras laterales son 5 rectángulos iguales. Para recordar, un pentágono regular es un polígono con cinco lados y ángulos iguales (todos sus ángulos son de 108°).
- **Prisma hexagonal regular.-** Es aquel que tiene como bases dos hexágonos regulares. Sus caras

laterales son 6 rectángulos iguales. Para recordar, un hexágono regular es un polígono con seis lados y ángulos iguales (todos sus ángulos son de 120°).

En esta ocasión se le pide elaborar un programa en lenguaje C que permita evaluar prismas pentagonales y hexagonales regulares, para ello debe leer un caracter que representa a un tipo de prisma regular (Pentagonal (P) y hexagonal (H)) y permita evaluar si el prisma es regular.

Para verificar si se trata de un prisma pentagonal regular debe tener en cuenta lo siguiente:

- Debe solicitar que ingrese por cada cara lateral del prisma los cuatro puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ y $D(x_4, y_4)$ del plano cartesiano que conforman el rectángulo que se encuentra en dicha cara lateral y verificar que se trate de un rectángulo.
- Considerar que la validación del rectángulo solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD y DA. Debe considerar como lados paralelos el AB y CD, así como el BC y DA. Además recuerde que en un rectángulo los lados paralelos tienen la misma medida.
- Debe solicitar que ingrese por cada base del prisma los 5 puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$, $D(x_4, y_4)$ y $E(x_5, y_5)$ del plano cartesiano que conforman el pentágono y verificar que se trate de un pentágono regular.
- Considerar que la validación del pentágono regular solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD, DE y EA.

Para verificar si se trata de un prisma hexagonal regular debe tener en cuenta lo siguiente:

- Debe solicitar que ingrese por cada cara lateral del prisma los cuatro puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ y $D(x_4, y_4)$ del plano cartesiano que conforman el rectángulo que se encuentra en dicha cara lateral y verificar que se trate de un rectángulo.
- Considerar que la validación del rectángulo solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD y DA. Debe considerar como lados paralelos el AB y CD, así como el BC y DA. Además recuerde que en un rectángulo los lados paralelos tienen la misma medida.
- Debe solicitar que ingrese por cada base del prisma los 6 puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$, $D(x_4, y_4)$, $E(x_5, y_5)$ y $F(x_6, y_6)$ del plano cartesiano que conforman el hexágono y verificar que se trate de un hexágono regular.
- Considerar que la validación del hexágono regular solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD, DE, EF y FA.

En esta pregunta se deben mostrar mensajes específicos ante las siguientes situaciones:

- Al ingresar el tipo de prisma regular, debe verificar que se ingresen solamente las letras P y H en mayúsculas o minúsculas. En caso no se cumpla, se deberá emitir el siguiente mensaje **La opción de prisma ingresado no corresponde a un pentagonal o hexagonal** y el programa debe terminar.
- Al calcular los lados del rectángulo que pertenecen a una cara lateral del prisma, se debe validar que efectivamente se trate de un rectángulo. En caso no se cumpla, deberá emitir el siguiente mensaje **Los puntos ingresados no forman un rectángulo** y se continúa evaluando la siguiente cara.
- Al calcular los lados del pentágono que pertenecen a una base del prisma pentagonal, se debe validar que los cinco lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un pentágono regular** y se continúa evaluando la siguiente base.
- Al calcular los lados del hexágono que pertenecen a una base del prisma hexagonal, se debe validar que los seis lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un hexágono regular** y se continúa evaluando la siguiente base.

- Al finalizar la validación de si se trata de un prisma regular, debe mostrar el siguiente mensaje **El prisma hexagonal es válido** o **El prisma pentagonal es válido** en caso el prisma regular sea válido. En caso que el prisma regular sea inválido, debe mostrar el siguiente mensaje **El prisma hexagonal no es válido** o **El prisma pentagonal no es válido** según corresponda y además debe indicar el motivo por el cual no es un prisma regular válido (laterales y/o base).

El programa debe estar desarrollado bajo el paradigma del programación modular, por lo que debe desarrollar y considerar únicamente los siguientes módulos dentro de su solución:

- Un módulo main
- Un módulo que permita validar si un prisma (pentagonal o hexagonal) es válido o no. Para ello, el módulo debe recibir como parámetro el tipo de prisma y devolver, como parámetros que se modifican luego de la invocación, si el prisma es válido, si sus caras laterales son correctas y si sus bases son correctas.
- Un módulo que permita procesar las bases de un prisma y determine si el prisma, en sus bases es válido o no. Para ello, el módulo debe recibir como parámetro el tipo de prisma y devolver si todas las bases procesadas son válidas o no. En este módulo debe utilizar **una** estructura iterativa con salida controlada para procesar las bases del prisma.
- Un módulo que permita procesar las caras laterales de un prisma regular y determine si el prisma, en sus caras laterales es válido o no. Para ello, el módulo debe recibir como parámetro la cantidad de caras laterales que va a procesar y devolver si todas las caras laterales procesadas son válidas o no. En este módulo debe utilizar **una** estructura iterativa con salida controlada para procesar las caras laterales del prisma.
- Un módulo que permita leer las coordenadas x e y de un punto del plano cartesiano. Para ello, el módulo debe devolver como parámetros que se modifican luego de la invocación, los valores de las coordenadas x e y del punto.
- Un módulo que permita determinar si cuatro puntos forman un rectángulo o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los cuatro puntos ingresados en una cara lateral y debe devolver si dichos puntos forman un rectángulo.
- Un módulo que permita determinar si cinco puntos forman un pentágono regular o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los cinco puntos ingresados en una base y debe devolver si dichos puntos forman un pentágono regular.
- Un módulo que permita determinar si seis puntos forman un hexágono regular o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los seis puntos ingresados en una base y debe devolver si dichos puntos forman un hexágono regular.
- Un módulo que permita calcular la distancia que existe entre dos puntos del plano cartesiano. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de dos puntos y debe devolver la distancia que existe entre ellos. Recuerde que la fórmula para calcular la distancia entre dos puntos es la siguiente:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Comparación de números reales

Muchas veces el resultado de la comparación de números reales a través de la igualdad no es el deseado. Esto sucede por la forma en que se representa internamente los reales, basta que exista una pequeña diferencia de precisión para que no se de la igualdad. En este caso es recomendable usar el valor absoluto de la diferencia de los números que se desean comparar. Para este problema, si esta diferencia es menor o igual a 0.01, se puede asumir que son iguales.

Caso de prueba 1

Evalutando Prismas Pentagonales y Hexagonales

Ingrese el tipo de prisma a evaluar (P-Pentagonal, H-Hexagonal): j

La opción del prisma ingresado no corresponde a un pentagonal o hexagonal.

Caso de prueba 2

Evalutando Prismas Pentagonales y Hexagonales

Ingrese el tipo de prisma a evaluar (P-Pentagonal, H-Hexagonal): p

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 2

Ingrese las coordenadas x e y del punto B: 3 2

Ingrese las coordenadas x e y del punto C: 3 0

Ingrese las coordenadas x e y del punto D: 1 0

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 1.5 2.5

Ingrese las coordenadas x e y del punto B: 3.5 2.5

Ingrese las coordenadas x e y del punto C: 3.5 0.5

Ingrese las coordenadas x e y del punto D: 1.5 0.5

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 6 1

Ingrese las coordenadas x e y del punto C: 6 -1

Ingrese las coordenadas x e y del punto D: 4 -1

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 4.5 1.5

Ingrese las coordenadas x e y del punto B: 6.5 1.5

Ingrese las coordenadas x e y del punto C: 6.5 -1.5

Ingrese las coordenadas x e y del punto D: 4.5 -1.5

Validaremos la cara lateral 5

Ingrese las coordenadas x e y del punto A: 7 3

Ingrese las coordenadas x e y del punto B: 9 3

Ingrese las coordenadas x e y del punto C: 9 1

Ingrese las coordenadas x e y del punto D: 7 1

Validaremos la base 1

Ingrese las coordenadas x e y del punto A: 4.2 6.75

Ingrese las coordenadas x e y del punto B: 8.14 9.73

Ingrese las coordenadas x e y del punto C: 12.19 6.91

Ingrese las coordenadas x e y del punto D: 10.76 2.18

Ingrese las coordenadas x e y del punto E: 5.82 2.08

Validaremos la base 2

Ingrese las coordenadas x e y del punto A: 7.2 6.75

Ingrese las coordenadas x e y del punto B: 11.14 9.73

Ingrese las coordenadas x e y del punto C: 15.19 6.91

Ingrese las coordenadas x e y del punto D: 13.76 2.18

Ingrese las coordenadas x e y del punto E: 8.82 2.08

El prisma pentagonal es válido.

Caso de prueba 3

Evalutando Prismas Pentagonales y Hexagonales

Ingrese el tipo de prisma a evaluar (P-Pentagonal, H-Hexagonal): P

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 3

Ingrese las coordenadas x e y del punto B: 3 2

Ingrese las coordenadas x e y del punto C: 3 0

Ingrese las coordenadas x e y del punto D: 1 0

Los puntos ingresados no forman un rectángulo

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 1.5 2.5

Ingrese las coordenadas x e y del punto B: 3.5 4.5

Ingrese las coordenadas x e y del punto C: 3.5 0.5

Ingrese las coordenadas x e y del punto D: 1.5 0.5

Los puntos ingresados no forman un rectángulo

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 6 1

Ingrese las coordenadas x e y del punto C: 6 -1

Ingrese las coordenadas x e y del punto D: 4 -1

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 4.5 1.5

Ingrese las coordenadas x e y del punto B: 6.5 1.5

Ingrese las coordenadas x e y del punto C: 6.5 -1.5

Ingrese las coordenadas x e y del punto D: 4.5 -1.5

Validaremos la cara lateral 5

Ingrese las coordenadas x e y del punto A: 7 3

Ingrese las coordenadas x e y del punto B: 9 3

Ingrese las coordenadas x e y del punto C: 9 1

Ingrese las coordenadas x e y del punto D: 7 1

Validaremos la base 1

Ingrese las coordenadas x e y del punto A: 4.2 6.75

Ingrese las coordenadas x e y del punto B: 8.14 9.73

Ingrese las coordenadas x e y del punto C: 12.19 6.91

Ingrese las coordenadas x e y del punto D: 10.76 2.18

Ingrese las coordenadas x e y del punto E: 5.82 2.08

Validaremos la base 2

Ingrese las coordenadas x e y del punto A: 7.2 8.75

Ingrese las coordenadas x e y del punto B: 11.14 9.73

Ingrese las coordenadas x e y del punto C: 15.19 6.91

Ingrese las coordenadas x e y del punto D: 13.76 2.18

Ingrese las coordenadas x e y del punto E: 8.82 2.08

Los puntos ingresados no forman un pentágono regular

El prisma pentagonal no es válido debido a:

Al menos un rectángulo lateral no es correcto

Al menos una base no es correcta

Caso de prueba 4

Evalutando Prismas Pentagonales y Hexagonales

Ingrese el tipo de prisma a evaluar (P-Pentagonal, H-Hexagonal): h

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 3

Ingrese las coordenadas x e y del punto B: 3 2

Ingrese las coordenadas x e y del punto C: 3 0

Ingrese las coordenadas x e y del punto D: 1 0

Los puntos ingresados no forman un rectángulo

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 1.5 2.5

Ingrese las coordenadas x e y del punto B: 3.5 4.5

Ingrese las coordenadas x e y del punto C: 3.5 0.5

Ingrese las coordenadas x e y del punto D: 1.5 0.5

Los puntos ingresados no forman un rectángulo

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 6 1

Ingrese las coordenadas x e y del punto C: 6 -1

Ingrese las coordenadas x e y del punto D: 4 -1

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 4.5 1.5

Ingrese las coordenadas x e y del punto B: 6.5 1.5

Ingrese las coordenadas x e y del punto C: 6.5 -1.5

Ingrese las coordenadas x e y del punto D: 4.5 -1.5

Validaremos la cara lateral 5

Ingrese las coordenadas x e y del punto A: 7 3

Ingrese las coordenadas x e y del punto B: 9 3

Ingrese las coordenadas x e y del punto C: 9 1

Ingrese las coordenadas x e y del punto D: 7 1

Validaremos la cara lateral 6

Ingrese las coordenadas x e y del punto A: 7.5 3.5

Ingrese las coordenadas x e y del punto B: 9.5 3.5

Ingrese las coordenadas x e y del punto C: 9.5 1.5

Ingrese las coordenadas x e y del punto D: 7.5 1.5

Validaremos la base 1

Ingrese las coordenadas x e y del punto A: -2 3.46

Ingrese las coordenadas x e y del punto B: 0 6.93

Ingrese las coordenadas x e y del punto C: 4 6.93

Ingrese las coordenadas x e y del punto D: 6 3.46

Ingrese las coordenadas x e y del punto E: 4 0

Ingrese las coordenadas x e y del punto F: 0 0

Validaremos la base 2

Ingrese las coordenadas x e y del punto A: -1 3.46

Ingrese las coordenadas x e y del punto B: 1 6.93

Ingrese las coordenadas x e y del punto C: 5 6.93

Ingrese las coordenadas x e y del punto D: 7 3.46

Ingrese las coordenadas x e y del punto E: 5 0

Ingrese las coordenadas x e y del punto F: 1 0

El prisma hexagonal no es válido debido a:

Al menos un rectángulo lateral no es correcto

Caso de prueba 5

Evalutando Prismas Pentagonales y Hexagonales

Ingrese el tipo de prisma a evaluar (P-Pentagonal, H-Hexagonal): H

Validaremos la cara lateral 1

Ingrese las coordenadas x e y del punto A: 1 2

Ingrese las coordenadas x e y del punto B: 3 2

Ingrese las coordenadas x e y del punto C: 3 0

Ingrese las coordenadas x e y del punto D: 1 0

Validaremos la cara lateral 2

Ingrese las coordenadas x e y del punto A: 1.5 2.5

Ingrese las coordenadas x e y del punto B: 3.5 2.5

Ingrese las coordenadas x e y del punto C: 3.5 0.5

Ingrese las coordenadas x e y del punto D: 1.5 0.5

Validaremos la cara lateral 3

Ingrese las coordenadas x e y del punto A: 4 1

Ingrese las coordenadas x e y del punto B: 6 1

Ingrese las coordenadas x e y del punto C: 6 -1

Ingrese las coordenadas x e y del punto D: 4 -1

Validaremos la cara lateral 4

Ingrese las coordenadas x e y del punto A: 4.5 1.5

Ingrese las coordenadas x e y del punto B: 6.5 1.5

Ingrese las coordenadas x e y del punto C: 6.5 -1.5

Ingrese las coordenadas x e y del punto D: 4.5 -1.5

Validaremos la cara lateral 5

Ingrese las coordenadas x e y del punto A: 7 3

Ingrese las coordenadas x e y del punto B: 9 3

Ingrese las coordenadas x e y del punto C: 9 1

Ingrese las coordenadas x e y del punto D: 7 1

Validaremos la cara lateral 6

Ingrese las coordenadas x e y del punto A: 7.5 3.5

Ingrese las coordenadas x e y del punto B: 9.5 3.5

Ingrese las coordenadas x e y del punto C: 9.5 1.5

Ingrese las coordenadas x e y del punto D: 7.5 1.5

Validaremos la base 1

Ingrese las coordenadas x e y del punto A: -2 3.46

Ingrese las coordenadas x e y del punto B: 0 6.93

Ingrese las coordenadas x e y del punto C: 4 6.93

Ingrese las coordenadas x e y del punto D: 6 3.46

Ingrese las coordenadas x e y del punto E: 4 0

Ingrese las coordenadas x e y del punto F: 0 0

Validaremos la base 2

Ingrese las coordenadas x e y del punto A: -1 3.46

Ingrese las coordenadas x e y del punto B: 1 6.93

Ingrese las coordenadas x e y del punto C: 5 6.93

Ingrese las coordenadas x e y del punto D: 7 3.46

Ingrese las coordenadas x e y del punto E: 5 0

Ingrese las coordenadas x e y del punto F: 1 0

El prisma hexagonal es válido

2.3.17. Tetraedro y hexaedro (Adaptado del Laboratorio 6 - 2021-1)

En geometría, un poliedro regular es aquella figura geométrica de tres dimensiones cuyas caras son todas iguales y, además, son polígonos regulares. Esto quiere decir que un poliedro regular está conformado por polígonos idénticos, cada uno de los cuales, a su vez, cumple con la condición de ser regular. Es decir, todos sus lados y

sus ángulos interiores miden lo mismo.

Existen 5 tipos de poliedros regulares, pero en esta ocasión solo veremos 2 de ellos, los cuales son:

- **Tetraedro regular.-** Es una figura de cuatro caras que son triángulos equiláteros. Cabe recordar que en un triángulo equilátero, sus tres lados miden lo mismo, al igual que sus ángulos interiores que son de 60° (la suma de los ángulos interiores de un triángulo siempre es 180°).
- **Hexaedro regular.-** Es una figura de seis lados formada por cuadrados idénticos. Cabe recordar que un cuadrado es un cuadrilátero regular, específicamente, un paralelogramo. Se caracteriza porque sus cuatro lados miden igual y sus ángulos interiores también son todos iguales y rectos (miden 90°).

En esta ocasión se le pide elaborar un programa en lenguaje C que permita evaluar tetraedros y hexaedros, para ello debe leer un carácter que representa a un tipo de poliedro (Tetraedro (T) y Hexaedro (H)) y permita evaluar si el poliedro es regular.

Para verificar si se trata de un poliedro regular, en el caso del tetraedro, se debe solicitar que ingrese por cada cara del poliedro, los tres puntos $A(x_1, y_1)$, $B(x_2, y_2)$ y $C(x_3, y_3)$ del plano cartesiano que conforman el triángulo que se encuentra en dicha cara y verificar si en realidad se trata de un triángulo equilátero. Considerar que la validación solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC y AC.

En el caso del hexaedro, debe solicitar que ingrese por cada cara del poliedro, los cuatro puntos $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ y $D(x_4, y_4)$ del plano cartesiano que conforman el cuadrado que se encuentra en dicha cara y verificar que en verdad se trate de un cuadrado. Considerar que la validación solo se debe realizar a nivel de igualdad de lados y debe considerar los lados formados por los puntos AB, BC, CD y DA.

En esta pregunta se deben mostrar mensajes específicos ante las siguientes situaciones:

- Al ingresar el tipo de poliedro, debe verificar que se ingresen solamente las letras T y H en mayúsculas o minúsculas. En caso no se cumpla, se deberá emitir el siguiente mensaje **La opción de poliedro ingresado no corresponde a un Tetraedro o Hexaedro** y el programa debe terminar.
- Al calcular los lados del triángulo que pertenecen a una cara del tetraedro, se debe validar que los tres lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un triángulo equilátero** y se continúa evaluando la siguiente cara.
- Al calcular los lados del cuadrado que pertenecen a una cara del hexaedro, se debe validar que los cuatro lados sean iguales. En caso no se cumpla, se deberá emitir el siguiente mensaje **Los puntos ingresados no forman un cuadrado** y se continúa evaluando la siguiente cara.
- Al finalizar la validación de si se trata de un tetraedro, debe mostrar el siguiente mensaje **El tetraedro ingresado es válido** en caso el tetraedro sea válido. En caso el tetraedro sea inválido, debe mostrar el siguiente mensaje **El tetraedro ingresado no es válido** y además debe indicar la cantidad de caras cuyos puntos ingresados no formaron un triángulo equilátero durante la validación.
- Al finalizar la validación de si se trata de un hexaedro, debe mostrar el siguiente mensaje **El hexaedro ingresado es válido** en caso el hexaedro sea válido. En caso el hexaedro sea inválido, debe mostrar el siguiente mensaje **El hexaedro ingresado no es válido** y además debe indicar la cantidad de caras cuyos puntos ingresados no formaron un cuadrado durante la validación.

El programa debe estar desarrollado bajo el paradigma de programación modular, por lo que debe desarrollar y considerar únicamente los siguientes módulos dentro de su solución:

- Un módulo main
- Un módulo que permita validar si la opción de poliedro ingresada es válida. Para ello, el módulo debe recibir como parámetro la opción de poliedro ingresada y debe devolver si dicha opción es válida o no.

- Un módulo que permita procesar las caras de un tetraedro y determine si el tetraedro es válido o no, además debe determinar la cantidad de caras cuyos puntos ingresados no formaron un triángulo equilátero. Para ello, el módulo debe devolver como parámetros que se modifican luego de la invocación, si el tetraedro es válido o no y la cantidad de caras cuyos puntos ingresados no formaron un triángulo equilátero. En este módulo debe utilizar una estructura iterativa con salida controlada para procesar las caras del tetraedro.
- Un módulo que permita procesar las caras de un hexaedro y determine si el hexaedro es válido o no, además debe determinar la cantidad de caras cuyos puntos ingresados no formaron un cuadrado. Para ello, el módulo debe devolver como parámetros que se modifican luego de la invocación, si el hexaedro es válido o no y la cantidad de caras cuyos puntos ingresados no formaron un cuadrado. En este módulo debe utilizar una estructura iterativa con salida controlada para procesar las caras del hexaedro.
- Un módulo que permita leer las coordenadas x e y de un punto del plano cartesiano. Para ello, el módulo debe devolver como parámetros que se modifican luego de la invocación, los valores de las coordenadas x e y del punto.
- Un módulo que permita determinar si un triángulo es equilátero o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los tres puntos ingresados en una cara y debe devolver si dichos puntos forman un triángulo equilátero.
- Un módulo que permita determinar si cuatro puntos forman un cuadrado o no. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de los cuatro puntos ingresados en una cara y debe devolver si dichos puntos forman un cuadrado.
- Un módulo que permita calcular la distancia que existe entre dos puntos del plano cartesiano. Para ello, el módulo debe recibir como parámetros las coordenadas x e y de dos puntos y debe devolver la distancia que existe entre ellos. Recuerde que la fórmula para calcular la distancia entre dos puntos es la siguiente:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Comparación de números reales

Muchas veces el resultado de la comparación de números reales a través de la igualdad no es el deseado. Esto sucede por la forma en que se representa internamente los reales, basta que exista una pequeña diferencia de precisión para que no se de la igualdad. En este caso es recomendable usar el valor absoluto de la diferencia de los números que se desean comparar. Para este problema, si esta diferencia es menor o igual a 0.00001, se puede asumir que son iguales.

Caso de prueba 1

Evalutando tetraedros y hexaedros

Ingresa el poliedro a evaluar (T-Tetraedro y H-Hexaedro): j

La opción de poliedro ingresado no corresponde a un Tetraedro o Hexaedro.

Caso de prueba 2

Evalutando tetraedros y hexaedros

Ingrese el poliedro a evaluar (T-Tetraedro y H-Hexaedro): T

Vamos a evaluar un Tetraedro, por ello validaremos las 4 caras que deben ser triángulos equiláteros

Ingrese las coordenadas de los puntos A, B y C del triángulo de la cara 1

Coordenada x e y del punto A: -1 1

Coordenada x e y del punto B: 3 1

Coordenada x e y del punto C: 1 4.464101

Ingrese las coordenadas de los puntos A, B y C del triángulo de la cara 2

Coordenada x e y del punto A: 3 5

Coordenada x e y del punto B: 3.2 9

Coordenada x e y del punto C: 5 4.464101

Los puntos ingresados no forman un triángulo equilátero.

Ingrese las coordenadas de los puntos A, B y C del triángulo de la cara 3

Coordenada x e y del punto A: 1 3

Coordenada x e y del punto B: 5 3

Coordenada x e y del punto C: 3 7.464101

Los puntos ingresados no forman un triángulo equilátero.

Ingrese las coordenadas de los puntos A, B y C del triángulo de la cara 4

Coordenada x e y del punto A: 1 3

Coordenada x e y del punto B: 5 3

Coordenada x e y del punto C: 3 6.464101

El tetraedro ingresado no es válido.

La cantidad de caras que no son triángulos equiláteros es 2.

Caso de prueba 3

Evalutando tetraedros y hexaedros

Ingrese el poliedro a evaluar (T-Tetraedro y H-Hexaedro): t

Vamos a evaluar un Tetraedro, por ello validaremos las 4 caras que deben ser triángulos equiláteros

Ingrese las coordenadas de los puntos A, B y C del triángulo de la cara 1

Coordenada x e y del punto A: -1 1

Coordenada x e y del punto B: 3 1

Coordenada x e y del punto C: 1 4.464101

Ingrese las coordenadas de los puntos A, B y C del triángulo de la cara 2

Coordenada x e y del punto A: -1 1

Coordenada x e y del punto B: 3 1

Coordenada x e y del punto C: 1 -2.464101

Ingrese las coordenadas de los puntos A, B y C del triángulo de la cara 3

Coordenada x e y del punto A: 1 3

Coordenada x e y del punto B: 5 3

Coordenada x e y del punto C: 3 6.464101

Ingrese las coordenadas de los puntos A, B y C del triángulo de la cara 4

Coordenada x e y del punto A: -3 -1

Coordenada x e y del punto B: 1 -1

Coordenada x e y del punto C: -1 2.464101

El tetraedro ingresado es válido.

Caso de prueba 4

Evalutando tetraedros y hexaedros

Ingrese el poliedro a evaluar (T-Tetraedro y H-Hexaedro): H

Vamos a evaluar un Hexaedro, por ello validaremos las 6 caras que deben ser cuadrados

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 1

Coordenada x e y del punto A: 1.5 4.5

Coordenada x e y del punto B: 6.5 4.5

Coordenada x e y del punto C: 6.5 -0.5

Coordenada x e y del punto D: 1.5 -0.5

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 2

Coordenada x e y del punto A: 1 4

Coordenada x e y del punto B: 6 4

Coordenada x e y del punto C: 6 0

Coordenada x e y del punto D: 1 0

Los puntos ingresados no forman un cuadrado

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 3

Coordenada x e y del punto A: 1 4

Coordenada x e y del punto B: 6 4

Coordenada x e y del punto C: 6 -1

Coordenada x e y del punto D: 1 -1

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 4

Coordenada x e y del punto A: 5.2 8.5

Coordenada x e y del punto B: 2.5 3.6

Coordenada x e y del punto C: 4.5 2.1

Coordenada x e y del punto D: 1 0

Los puntos ingresados no forman un cuadrado

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 5

Coordenada x e y del punto A: 5.2 8.5

Coordenada x e y del punto B: 2.5 3.6

Coordenada x e y del punto C: 4.5 2.1

Coordenada x e y del punto D: 3 0

Los puntos ingresados no forman un cuadrado

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 6

Coordenada x e y del punto A: 4.2 7.5

Coordenada x e y del punto B: 3.5 4.6

Coordenada x e y del punto C: 2.5 3.1

Coordenada x e y del punto D: 3 0

Los puntos ingresados no forman un cuadrado

El hexaedro ingresado no es válido. La cantidad de caras que no son cuadrados es 4

Caso de prueba 5

Evalutando tetraedros y hexaedros

Ingrese el poliedro a evaluar (T-Tetraedro y H-Hexaedro): h

Vamos a evaluar un Hexaedro, por ello validaremos las 6 caras que deben ser cuadrados

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 1

Coordenada x e y del punto A: 1.5 4.5

Coordenada x e y del punto B: 6.5 4.5

Coordenada x e y del punto C: 6.5 -0.5

Coordenada x e y del punto D: 1.5 -0.5

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 2

Coordenada x e y del punto A: 1 4

Coordenada x e y del punto B: 6 4

Coordenada x e y del punto C: 6 -1

Coordenada x e y del punto D: 1 -1

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 3

Coordenada x e y del punto A: 3 6

Coordenada x e y del punto B: 8 6

Coordenada x e y del punto C: 8 1

Coordenada x e y del punto D: 3 1

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 4

Coordenada x e y del punto A: 4.5 7.5

Coordenada x e y del punto B: 9.5 7.5

Coordenada x e y del punto C: 9.5 2.5

Coordenada x e y del punto D: 4.5 2.5

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 5

Coordenada x e y del punto A: 5 8

Coordenada x e y del punto B: 10 8

Coordenada x e y del punto C: 10 3

Coordenada x e y del punto D: 5 3

Ingrese las coordenadas de los puntos A, B, C y D del cuadrado de la cara 6

Coordenada x e y del punto A: 4.2 7.2

Coordenada x e y del punto B: 9.2 7.2

Coordenada x e y del punto C: 9.2 2.2

Coordenada x e y del punto D: 4.2 2.2

El hexaedro ingresado es válido.

Capítulo 3

Referencias

- Paul Deitel and Harvey Deitel. C How to program. Pearson –Addison Wesley, seventh edition, 2011.
- Guía de Laboratorio #5 de Fundamentos de Programación - 2019
- Guía de Laboratorio #8 de Fundamentos de Programación - 2019