

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESTUDIOS GENERALES CIENCIAS

1INF01 - FUNDAMENTOS DE PROGRAMACIÓN

Guía de laboratorio #3

Ciclo Iterativo con Entrada Controlada



**PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ**

18 de septiembre de 2021

Índice general

Historial de revisiones	1
Siglas	2
1. Guía de Laboratorio #3	3
1.1. Introducción	3
1.2. Materiales y métodos	3
1.3. Ciclo Iterativo con Entrada Controlada	3
1.3.1. Representación del Ciclo Iterativo con Entrada Controlada	4
1.4. Cálculo del número e	5
1.4.1. Implementación del pseudocódigo en PSeInt	6
1.4.2. Implementación del diagrama de flujo en PSeInt	12
1.4.3. Implementación en Lenguaje C	13
1.5. Cálculo del seno	21
1.5.1. Listando números impares	21
1.5.2. Leyendo X	22
1.5.3. Sumando los números impares	23
1.5.4. Calculando el factorial	24
1.5.5. Formando el término de la suma	24
1.5.6. Gestionando las potencias en el término	25
1.5.7. Gestionando el signo en el término	26
1.5.8. Controlando el ciclo por el valor del término	26
2. Ejercicios propuestos	28
2.1. Nivel básico	28
2.1.1. Mayor y menor número de una lista	28
2.1.2. Conteo de frecuencias	29
2.1.3. Medidas de tendencia central y de dispersión	30
2.1.4. Clasificación de un triángulo - adaptado del examen parcial 2019-0	32
2.1.5. Cuadrado de un número	33

2.1.6. Convergencia de series	34
2.2. Nivel intermedio	35
2.2.1. Cálculo de $\ln(1 + x)$	35
2.2.2. Cálculo de e^x	37
2.2.3. Cálculo de la potencia de un número	38
2.2.4. Funciones trigonométricas: coseno	39
2.2.5. Distancia de Hamming	39
2.2.6. Distancia de Chebyshev	40
2.2.7. Distancia de Manhattan	41
2.2.8. El Algoritmo de la Lazada	42
2.2.9. Algoritmo KNN	44
2.2.10. Conjetura de Collatz	46
2.2.11. Cálculo de integrales: la suma de Riemann	47
2.2.12. Rebotes sucesivos	48
2.2.13. Cálculo numérico: El método de bisección	48
2.2.14. Regresión lineal simple: suma de las distancias al cuadrado	49
2.2.15. Trayectoria de un proyectil	50
2.3. Nivel avanzado	51
2.3.1. La serie Gregory-Leibniz	51
2.3.2. La serie de Nilakantha	52
2.3.3. Rotar un número	53
2.3.4. Cocientes de la serie Fibonacci	53
2.3.5. El número Anaprimo	54
2.3.6. Calcular el número de triangulaciones para un polígono	55
2.3.7. La Regla de Simpson	56
2.3.8. Calcular área	57
2.3.9. Número Narcisista	58
2.3.10. Cálculo del arcotangente de un número real (adaptado del laboratorio 3 2020-2)	59
2.3.11. Cálculo del logaritmo natural (adaptado del laboratorio 3 2020-2)	60
2.3.12. Cálculo del arcoseno de un número (adaptado del laboratorio 3 2020-2)	61
2.3.13. Cálculo del seno con la serie de Taylor (adaptado del laboratorio 3 2020-2)	62
2.3.14. Aproximación de series para el cálculo de e (adaptado del laboratorio 3 2020-2)	63
2.3.15. Cálculo de la serie Maclaurin (adaptado del laboratorio 3 2020-2)	64
2.3.16. Aproximación de series para el cálculo de e (adaptado del laboratorio 3 2020-2)	65
2.3.17. Cálculo de la cosecante de un ángulo con la serie de Taylor (adaptado del laboratorio 3 2020-2)	66
2.3.18. Cálculo del arcosecante de un número (adaptado del laboratorio 3 2020-2)	67
2.3.19. Cálculo de e en función a aproximación de series (adaptado del laboratorio 3 2020-2)	68

2.3.20. Cálculo de la función racional (adaptado del laboratorio 3 2020-2)	69
2.3.21. Rectas paralelas y perpendiculares (adaptado del laboratorio 3 2021-1)	70
2.3.22. Rectas tangentes a la circunferencia (adaptado del laboratorio 3 2021-1)	73
2.3.23. Rectas tangentes y rectas normales a la circunferencia (adaptado del laboratorio 3 2021-1)	76
2.3.24. Los segmentos en el plano cartesiano (adaptado del laboratorio 3 2021-1)	78
2.3.25. La Circunferencia (adaptado del laboratorio 3 2021-1)	80

FUNDAMENTOS DE PROGRAMACIÓN
ESTUDIOS GENERALES CIENCIAS
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Historial de Revisiones

Revisión	Fecha	Autor(es)	Descripción
1.0	06.05.2018	A.Melgar	Versión inicial.
1.1	10.05.2018	A.Melgar	Versión inicial revisada por Layla Hirsh.
1.2	29.09.2018	A.Melgar	Se mejoró la representación del ciclo iterativo con entrada controlada.
2.0	10.05.2019	A.Melgar	Se incrementó la cantidad de problemas propuestos, se añadió color al código en el lenguaje C y se completaron los casos de prueba de los problemas propuestos.
2.0	30.05.2019	L. Hirsh	Revisión de la versión 2.0.
2.1	06.07.2019	A.Melgar	Revisión de la redacción.
2.2	20.09.2020	J.Zárate	Clasificación de los ejercicios propuestos en nivel básico, intermedio y avanzado. Se agregaron ejercicios del nivel avanzado
3.0	20.04.2021	D.Allasi	Se incrementó la cantidad de problemas propuestos y se actualizó los problemas para no manejar centinela
3.1	18.09.2021	S.Ponce	Se incluyen nuevos ejercicios

Siglas

EEGGCC	Estudios Generales Ciencias
IDE	Entorno de Desarrollo Integrado
PUCP	Pontificia Universidad Católica del Perú

Capítulo 1

Guía de Laboratorio #3

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 “Estructura Iterativa con Entrada 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 con la comprensión de las estructuras algorítmicas iterativas. 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 iterativa con entrada controlada.
- Comprenda el funcionamiento de la iteración controlada por contador.
- Construya programas usando la estructura algorítmica iterativa con entrada 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. Ciclo Iterativo con Entrada Controlada

Los programas escritos en el paradigma imperativo se basan en el cambio de estado de las variables definidas en los programas. Todo programa sigue un flujo el cual puede ser cambiado a partir 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 algorítmicas selectivas, vistas anteriormente, permiten que los programas ejecuten un conjunto de instrucciones si es que se cumple determinada condición. Por otra parte, las estructuras iterativas permiten que los programas ejecuten un conjunto de instrucciones tantas veces como sea necesario, dependiendo de determinada condición.

¹<http://pseint.sourceforge.net/>

²<http://www.bloodshed.net/dev/devcpp.html>

Las estructuras algorítmicas iterativas se pueden clasificar en dos: i) ciclo iterativo con entrada controlada y ii) ciclo iterativo con salida controlada. La diferencia entre ambas radica en el momento en que se realiza la verificación de la condición. Si la condición se evalúa antes de la ejecución del conjunto de instrucciones, se estará ante un ciclo iterativo con entrada controlada. Si la condición se evalúa luego de la ejecución del conjunto de instrucciones, se estará ante un ciclo iterativo con salida controlada.

En el ciclo iterativo con entrada controlada, como se ha mencionado anteriormente, la evaluación de la condición se realiza antes de la ejecución del conjunto de instrucciones. Por esta razón, es posible que el conjunto de instrucciones no se ejecute si es que la condición de entrada no se cumple. Por otro lado, si la condición se cumple, el conjunto de instrucciones se ejecutará mientras la condición sea verdadera. Por este motivo es importante que dentro del conjunto de instrucciones de esta estructura, existan algunas instrucciones que lleven a cambiar el estado de la condición hasta que la condición falle, sino el ciclo nunca terminará (ciclo infinito).

Los ciclos iterativos son muy usados en la programación imperativa, gracias a ellos se pueden procesar instrucciones de forma reiterativa. Muchos cálculos matemáticos requieren de ciclos iterativos para hallar sus valores como por ejemplo: la serie de Fibonacci, el cálculo del factorial, el cálculo del número e , las funciones trigonométricas como seno, coseno, tangente, cotangente, secante y cosecante, el cálculo de integrales, entre otros.

1.3.1. Representación del Ciclo Iterativo con Entrada Controlada

A continuación se revisará cómo se representa el ciclo iterativo con entrada controlada tanto en pseudocódigo como en diagrama de flujo, así como su implementación en el lenguaje C.

Representación en pseudocódigo

En la figura 1.1 se puede apreciar la representación del ciclo iterativo con entrada controlada en pseudocódigo. El ciclo inicia con el identificador **Mientras** y finaliza con el identificador **Fin Mientras**. La condición es una expresión lógica como por ejemplo: $i < 100$, $nota \leq 10$, $i \leq max$ y $i < > 0$. En el conjunto de instrucciones se pueden colocar asignaciones ($i \leftarrow 0$), expresiones matemáticas ($suma \leftarrow suma + termino$), estructuras selectivas (Si $i=0$ Entonces) e inclusive otras estructuras iterativas.

```

Mientras condición Hacer
| conjunto de instrucciones;
Fin Mientras

```

Figura 1.1: Pseudocódigo: Ciclo iterativo con entrada controlada

Representación en diagrama de flujo

En la figura 1.2 se puede apreciar la representación del ciclo iterativo con entrada controlada en diagrama de flujo. El ciclo inicia con el bloque **condición** y si la condición es verdadera, se ejecuta el **conjunto de instrucciones**. La letra V representa *verdadero* y la letra F representa *falso* e indican la dirección por donde pasa el flujo teniendo en cuenta el valor de la condición. En el diagrama, el control del flujo se gestiona a través de las líneas que conectan los bloques. Como se aprecia, inmediatamente después de ejecutar el **conjunto de instrucciones**, el flujo se dirige nuevamente hacia el bloque **condición**. Si la condición se sigue cumpliendo, el flujo se dirige, otra vez, hacia el **conjunto de instrucciones**. 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 se dirige por la línea F .

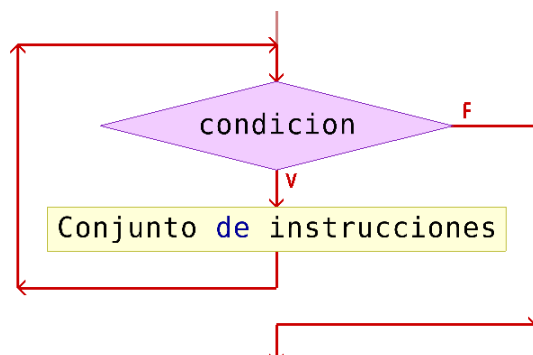


Figura 1.2: Diagrama de Flujo: Ciclo iterativo con entrada controlada

Implementación en el lenguaje C

En el lenguaje C existen varias formas de implementar el ciclo iterativo con entrada controlada. En este guía se revisará la instrucción `while` cuya representación se puede apreciar en el programa 1.1. El funcionamiento de la instrucción `while` es muy similar al *Mientras* del pseudocódigo, pero hay que tener ciertas consideraciones para su uso.

Como se sabe, el lenguaje C no implementa nativamente el tipo de dato lógico (`bool` o `boolean`). Entonces, ¿cómo hace el lenguaje C para evaluar la condición de la instrucción `while`? El lenguaje C asume que todo valor igual a 0 falla una condición. El comportamiento es muy similar al *falso* de una expresión lógica. Por lo contrario, una valor diferente de 0 hará que se cumpla la condición. Comportamiento similar al *verdadero* de una expresión lógica. Por ejemplo: si se tienen las siguientes definiciones `int suma=0, i=10;`, las siguientes expresiones serán consideradas *verdaderas*: $\text{suma} \leq 100$, $i == 10$, $\text{suma} < i$, i . Por otro lado, las siguientes expresiones serán consideradas *falsas*: $\text{suma} \geq 100$, $i == 20$, $\text{suma} > i$, suma .

Programa 1.1: Lenguaje C: Ciclo iterativo con entrada controlada

```

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

```

Se debe tener especial cuidado en no confundir el operador relacional de igualdad (`==`) con la asignación (`=`). Ambos son operadores, por lo tanto retornan un valor luego de evaluarse en la expresión, consecuentemente ambos pueden ser usados dentro de la condición en la instrucción `while`. La expresión `i == 10` lo que hace es realizar una comparación entre los dos operandos, si los operandos son iguales retorna el valor de 1 y si los operandos son diferentes retorna el valor de 0. Según la definición anterior, la expresión `i == 10` retornaría el valor de 1 si es que la variable `i` contiene el valor de 10. ¿Qué pasa cuando se tiene la expresión `i = 10` en la condición del `while`? Como se comentó anteriormente, `=` es el operador de asignación, hace básicamente dos cosas: i) asigna el valor del operando de la derecha al operando que está a la izquierda (por esta razón el operando de la izquierda debe ser siempre una variable); y ii) retorna como valor de la expresión, el valor realizado en la asignación. En este caso puntual el valor de la expresión `i = 10` sería 10. En conclusión, si colocásemos `i = 10` como condición de la instrucción `while`, no nos reportaría error, estaríamos frente a una condición que siempre se evaluaría como *verdadera*.

1.4. Cálculo del número e

El número e es un número muy usado en las ciencias³, jugando un papel importante en 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 del cual no podemos conocer su valor

³Para entender un poco más a profundidad qué es el número e , se recomienda el siguiente video <https://www.youtube.com/watch?v=Z5czpA-fyMU>.

exacto porque tiene infinitas cifras decimales. El valor aproximado de este número es 2.71828 (aproximado a 5 decimales).

Existen varias definiciones del número e siendo la más común el $\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$. 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 aproximando el número a 6 decimales.

1.4.1. Implementación del pseudocódigo en PSeInt

A continuación 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.

Listado de n números

Como se puede apreciar en la fórmula para el cálculo del número e , en el denominador aparece una secuencia de números (1, 2, 3, 4, 5, 6, ...). El patrón que sigue esta secuencia es bastante simple: el sucesor de un número es igual al predecesor incrementado en 1. Intentaremos en primer lugar imprimir esta serie de números. No sabemos *a priori* cuantos términos tendrá la serie para el cálculo del número e . Para efectos prácticos listaremos los 10 números naturales incluyendo el cero, luego nos preocuparemos de que la serie contenga la cantidad de términos correctos.

¿Cómo hacemos para imprimir los 10 primeros números naturales? Este es un típico ejemplo de uso de una estructura iterativa. Usaremos la instrucción **Mientras**. Hay que tener siempre en consideración 3 aspectos al usar la estructura iterativa: i) inicialización de las variables de control, ii) definición de la condición usando las variables de control, y iii) asegurarse que la condición en algún momento llegue a finalizar.

Si se desea imprimir los 10 primeros números naturales, y dado que estamos frente a la programación imperativa, se debe tener una variable que cambie de estado desde 0 hasta el valor requerido. Usaremos como nombre de variable i . ¿Con qué valor inicializamos esta variable i ? Bueno se requiere ir desde 0, lo más natural es que esta variable i sea inicializada con el valor de 0. ¿Cómo se define la condición en la estructura iterativa? Se desea parar cuando ya se tengan impreso los 10 primeros números naturales, esto quiere decir que i no puede llegar a ser igual a 10 por lo que una condición válida sería $i < 10$. ¿Cómo se asegura que la condición en algún momento llegue a finalizar? Si la variable i inicia en 0 y la condición $i < 10$ tiene que hacerse falsa alguna vez, entonces en algún momento i debe llegar a tener el valor de 10, debe paulatinamente ir incrementándose su valor. ¿Cómo se hace ese incremento? El incremento dependerá de cada problema en específico, en este caso el patrón indica que debe incrementarse en 1. Este incremento por lo general se realiza en la última instrucción del bloque de instrucciones del **Mientras**.

En la figura 1.3 se puede apreciar un pseudocódigo que lista los 10 primeros números naturales incluyendo el cero.

```

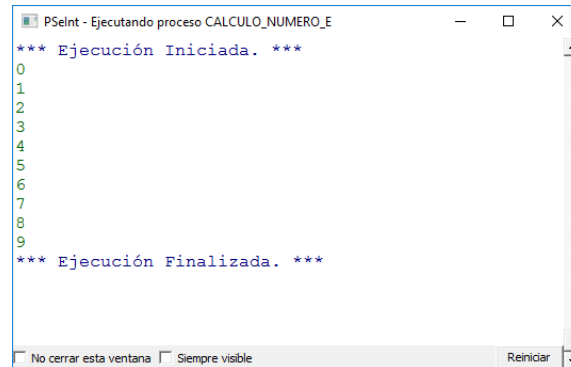
1  Algoritmo calculo_Numero_e
2      i <- 0
3      Mientras i<10 Hacer
4          Escribir i
5          i <- i+1
6      FinMientras
7  FinAlgoritmo

```

Figura 1.3: Pseudocódigo: Listado de n números

Se puede observar claramente los 3 elementos mencionados anteriormente: i) inicialización de la variable de

control (figura 1.3-línea 2), ii) definición de la condición usando la variable de control (figura 1.3-línea 3), y iii) aseguramiento que la condición en algún momento llegue a finalizar (figura 1.3-línea 5). Lo que se hace en cada iteración es imprimir el valor de la variable i (figura 1.3-línea 4) que se va incrementando de 1 en 1 hasta llegar a tener el valor de 10. Cuando llega a tener el valor de 10 la condición falla y no se ejecuta el conjunto de instrucciones asociados a la instrucción **Mientras**. Por esta razón solo se imprimen los números del 0 al 9 tal como se puede apreciar en la figura 1.4.



```
*** Ejecución Iniciada. ***
0
1
2
3
4
5
6
7
8
9
*** Ejecución Finalizada. ***
```

Figura 1.4: Salida: Listado de n números

Para poner en práctica

- ¿Qué tendría que variar en el pseudocódigo anterior si es que se quisiera listar los números entre determinado rango? Intente imprimir los números en los siguientes rangos: $[5..10]$, $[5..10[$, $]5..10]$, $]5..10[$.
- ¿Qué tendría que variar en el pseudocódigo anterior si es que se quisiera listar los números pares? Intente imprimir los números pares en el rango $[2..20]$
- ¿Qué tendría que variar en el pseudocódigo anterior si es que se quisiera listar los números impares? Intente imprimir los números pares en el rango $[1..13]$.

Acumulando la suma

Se ha conseguido imprimir los números del 0 al 9, pero para calcular el número e debemos de sumar los términos de la serie. Vamos a centrarnos en acumular la suma de los números que estamos generando en cada iteración, posteriormente veremos cómo lo adecuamos para calcular el número e . ¿Cómo se acumula la suma? Para poder obtener la suma acumulada luego del ciclo iterativo, se debe usar otra variable para ese fin. La llamaremos *suma*. Todo acumulador debe ser inicializado, en este caso lo inicializaremos con el valor de 0 ($suma \leftarrow 0$) dado que 0 es el elemento neutro para la suma. En cada iteración procederemos a actualizar su valor ($suma \leftarrow suma + i$), de esta forma cuando finalice el ciclo iterativo, la variable *suma* tendrá el valor acumulado de todos los números que hemos ido generando. Al final *suma* tendrá el valor de $0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9$. Es importante notar que este valor solo se obtendrá al final de todas iteraciones, por este motivo se debe imprimir al terminar el ciclo iterativo. En la figura 1.5 se puede apreciar el pseudocódigo propuesto para acumular la suma. En la 1.6 se puede apreciar la salida obtenida por este pseudocódigo.

```

1  Algoritmo calculo_Numero_e
2      i <- 0
3      suma <- 0
4      Mientras i<10 Hacer
5          Escribir i
6          suma <- suma + i
7          i <- i+1
8      FinMientras
9      Escribir "suma = ", suma
10 FinAlgoritmo

```

Figura 1.5: Pseudocódigo: Acumulando la suma

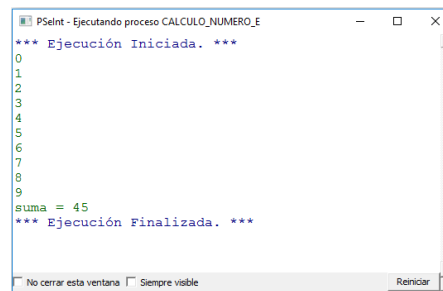


Figura 1.6: Salida: Acumulando la suma

Para poner en práctica

- ¿Qué pasa si es que la impresión de la variable *suma* se realiza dentro de la instrucción **Mientras**? Pruebe moviendo la impresión de la variable *suma* (figura 1.5-línea 9) dentro de la iteración, luego de la acumulación (insértela después de la figura 1.5-línea 6).
- Luego de experimentar con el cambio, reflexione sobre la siguiente pregunta: ¿existe alguna diferencia entre colocar la impresión de la variable *suma* dentro del ciclo iterativo y colocarla fuera del ciclo iterativo?, ¿por qué?

Hallando el término de la serie

Ahora que ya se sabe cómo acumular la suma de n números, intentaremos ir dándole forma al término para que sea lo más cercano a lo que se nos pide en la serie del número e . Intentaremos acumular primero la siguiente serie $1 + \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \dots$. Se ha retirado el factorial que se verá posteriormente. Si con el pseudocódigo anterior acumulábamos la suma de los números, bastará que en lugar de acumular i acumulemos $\frac{1}{i}$ y resolvemos lo pedido. En la figura 1.7 se presenta el pseudocódigo con el cambio indicado. Intente ejecutarlo con la herramienta PSeInt.

Para analizar

¿Qué error ocurre con el pseudocódigo?, ¿por qué no funciona?

Verificando el primer término

Si ha analizado al detalle el pseudocódigo en el paso anterior, se habrá dado cuenta que el valor inicial de la variable i es 0, entonces la primera vez que se ejecuta el bloque iterativo intentará realizar lo siguiente: $suma \leftarrow suma + \frac{1}{0}$. Como se sabe, en las matemáticas no es posible la división entre 0 por lo que se debe

```

1  Algoritmo calculo_Numero_e
2      i <- 0
3      suma <- 0
4      Mientras i<10 Hacer
5          Escribir i
6          suma <- suma+1/i
7          i <- i+1
8      FinMientras
9      Escribir 'suma = ',suma
10 FinAlgoritmo

```

Figura 1.7: Pseudocódigo: Hallando el término de la serie

evitar este tipo de operaciones. Recordemos que deseamos obtener el valor de $1 + \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \dots$. Nuestro pseudocódigo funcionaría bien para todos los casos, menos para el primero. ¿Cómo hacer para que solamente en el primer caso incremente en 1 a la suma y en los siguientes incremente en $\frac{1}{i}$? El algoritmo debe tomar una decisión (incrementar la suma en 1 o incrementar la suma en $\frac{1}{i}$) dada determinada condición ($i = 0$). Es una situación en la que se puede usar una estructura de control de flujo ya estudiada: la estructura selectiva. En la figura 1.8 se puede apreciar el pseudocódigo que se obtiene luego de incorporarle una estructura selectiva doble. Cuando se esté procesando el primer número (condición $i = 0$) la suma se incrementará en 1 en las demás situaciones se incrementará en $\frac{1}{i}$. En la 1.9 se puede apreciar la salida obtenida por este pseudocódigo.

```

1  Algoritmo calculo_Numero_e
2      i <- 0
3      suma <- 0
4      Mientras i<10 Hacer
5          Escribir i
6          Si i=0 Entonces
7              suma <- suma + 1
8          SiNo
9              suma <- suma+1/i
10         Fin Si
11         i <- i+1
12     FinMientras
13     Escribir 'suma = ',suma
14 FinAlgoritmo

```

Figura 1.8: Pseudocódigo: Verificando el primer término

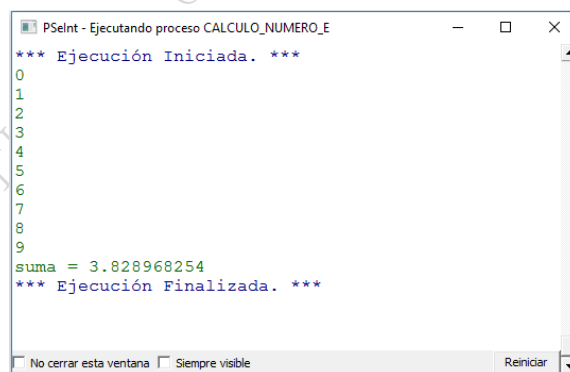


Figura 1.9: Salida: Verificando el primer término

Para pensar

Existe una manera de evitar el uso de la estructura selectiva en el caso anterior y obtener al final el mismo resultado. ¿Qué cambios debería hacerse en el pseudocódigo para lograr esto? Recuerde que i no debería tener el valor de 0 pues ejecutaría nuevamente división entre 0.

Calculando el factorial de i

Hasta el momento se ha podido acumular la suma, verificando el primer término para que no realice división entre 0. Pero para hallar el valor del número e debemos dividir por el factorial del número. Entonces, ¿cómo calculamos el factorial?

Recordar que:

El factorial de número n se puede definir como:

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ 1 \times 2 \times 3 \times \dots \times (n-1) \times n & \text{si } n \text{ es } > 0 \end{cases}$$

El cálculo del factorial de i es similar a la suma, pero en este caso no se tendrá una sumatoria ($\sum_{i=1}^n i$), sino una productoria ($\prod_{i=1}^n i$). Se usará otra variable para este cálculo la cual se llamará *factorial*. ¿Con qué valor inicializamos esta variable? Como es una productoria, no es posible inicializarla con 0 dado que cualquier número multiplicado por 0 retornaría 0. Dado que el factorial de 0 es 1 y dado que el primer número de la serie también inicia en 0, inicializaremos *factorial* con 1. ¿Cómo se actualiza la variable *factorial*? En cada iteración se debe ir actualizando esta variable, multiplicándola por el valor actual de i . Nuevamente se debe tener cuidado con el primer término de la serie el cual es 0. En este caso no debemos realizar la multiplicación pues obtendríamos como resultado 0. En los casos que la variable i tenga valores diferentes de 0 la variable *factorial* deberá ser actualizada de la siguiente manera $factorial \leftarrow factorial \times i$. Cuando i tenga el valor de 1 la variable factorial tendrá el valor 1×1 , cuando i tenga el valor de 2 la variable factorial tendrá el valor $1 \times 1 \times 2$, cuando i tenga el valor de 3 la variable factorial tendrá el valor $1 \times 1 \times 2 \times 3$, y así sucesivamente. De esta forma se logra que la variable *factorial* tenga en cada iteración el valor correcto del factorial de i .

En la figura 1.10 se puede apreciar el pseudocódigo que se obtiene luego de incorporarle el cálculo del factorial. En la 1.11 se puede apreciar la salida obtenida por este pseudocódigo.

Para poner en práctica

- Si usted observa bien la salida verá que no se ha impreso el factorial de 0, ¿qué cambios debería hacer en el pseudocódigo de forma tal que se imprima también el factorial de 0 pero se utilice una sola vez la instrucción Escribir i , “ ”, factorial?
- ¿Qué ocurre cuando el cálculo del factorial se realiza fuera de la instrucción selectiva? Mueva las líneas 10 y 11 de la figura 1.10 e insértelas entre la línea 12 y la línea 13.

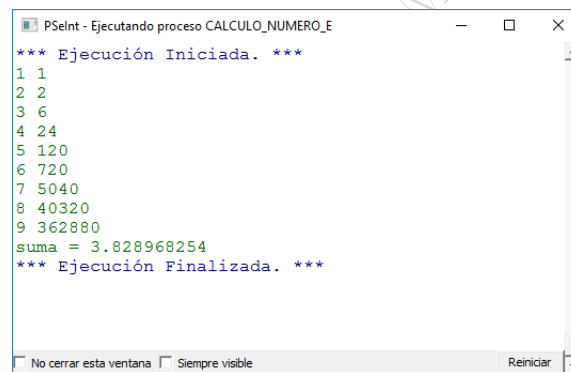
Incluyendo el factorial en el término de la serie

Ahora que se ha calculado el factorial de i procederemos a incluirlo en el término de la serie. Recordemos que por el momento se ha calculado la suma como $suma \leftarrow suma + \frac{1}{i}$. Para tener el término correcto, se debe cambiar el i por el *factorial* (recordemos que la variable i debe ser reemplazada por el factorial de i). Realizado este cambio el término correcto quedaría como $suma \leftarrow suma + \frac{1}{factorial}$. Para que esta expresión

```

1  Algoritmo calculo_Numero_e
2      i <- 0
3      suma <- 0
4      factorial <- 1
5      Mientras i<10 Hacer
6          Si i=0 Entonces
7              suma <- suma+1
8          SiNo
9              suma <- suma+1/i
10             factorial <- factorial*i
11             Escribir i, " ", factorial
12         FinSi
13         i <- i+1
14     FinMientras
15     Escribir 'suma = ', suma
16 FinAlgoritmo

```

Figura 1.10: Pseudocódigo: Calculando el factorial de i Figura 1.11: Salida: Calculando el factorial de i

realice el cálculo correctamente, el cálculo del factorial debe realizarse antes de su utilización. En la figura 1.12 se puede apreciar el pseudocódigo incluyendo el cálculo del factorial, además se está acumulando en la variable *suma* el término correcto, es decir $\frac{1}{i!}$. Se ha dejado solamente la impresión final dado que es lo que se desea mostrar. En la 1.13 se puede apreciar la salida obtenida por este pseudocódigo.

```

1  Algoritmo calculo_Numero_e
2      i <- 0
3      suma <- 0
4      factorial <- 1
5      Mientras i<10 Hacer
6          Si i=0 Entonces
7              suma <- suma+1
8          SiNo
9              factorial <- factorial*i
10             suma <- suma+1/factorial
11         FinSi
12         i <- i+1
13     FinMientras
14     Escribir 'suma = ', suma
15 FinAlgoritmo

```

Figura 1.12: Pseudocódigo: Incluyendo el factorial en el término de la serie

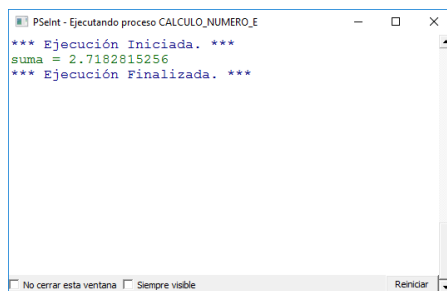


Figura 1.13: Incluyendo el factorial en el término de la serie

Para poner en práctica

- ¿Qué ocurre si el cálculo del factorial se realiza luego de acumular *factorial* en la *suma*? Inter-cambie las líneas 9 y 10 de la figura 1.12 e verifique que sucede.
- ¿Importa el orden de ejecución de las instrucciones?

Controlando el flujo por el valor del término de la serie

Hasta el momento se ha iterado el ciclo 10 veces pero el problema no indica que éste sea el número de iteraciones que se deba usar. Se colocó 10 por practicidad pero no se puede dejar ese valor pues no estaríamos resolviendo el problema. El problema indica que debemos calcular el número e con una aproximación de 6 decimales. ¿Cómo hacemos para controlar entonces el ciclo iterativo?

Si analizamos a la serie $1 + \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \dots$ podemos notar que la misma es decreciente. Esto quiere decir que los términos de la serie se van haciendo cada vez más pequeños. Existirán términos que serán muy pequeños por lo que podemos parar de acumular cuando el término que se calcule sea menor a 0.0000001 dado que se desea una precisión de 6 dígitos decimales.

¿Cómo realizar el control por el término de la serie? Para esto se utilizará una nueva variable que llamaremos *termino*. En esta variable calcularemos el término de la serie y se comparará este valor en la condición. El ciclo iterativo se ejecutará si y solo si el *termino* $>$ 0.0000001, si se llega a calcular un valor menor a este, el ciclo iterativo debe parar. Pero estamos frente a un ciclo iterativo con entrada controlada, esto significa que antes de calcular el término, se ejecutará la condición de entrada a la instrucción *Mientras*. Si se desea usar término como condición de la iterativa, debe tener un valor inicial. ¿Con qué valor se recomienda inicializar esta variable? Con cualquier valor que haga que la condición *termino* $>$ 0.0000001 se haga *verdadera* la primera vez. Por ejemplo podemos usar el valor de 1. En la figura 1.14 se puede apreciar una alternativa de solución al problema planteado del cálculo del número e . La salida se puede visualizar en la figura 1.15.

Para pensar

¿Qué cambios se le debería realizar al pseudocódigo para que el ciclo iterativo termine cuando encuentre el número e con una aproximación de 4 decimales o cuando el número de términos de la serie llegue a 15?

1.4.2. Implementación del diagrama de flujo en PSeInt

A continuación se presentará una propuesta de solución utilizando la implementación de diagrama de flujo de PSeInt. La salida que se obtiene por este diagrama de flujo es la misma que se obtiene en la sección anterior (1.4.1). Se recomienda realizar las alteraciones solicitadas en la sub sección anterior usando ahora diagramas de flujo. El diagrama de flujo se puede apreciar en la figura 1.16. Se puede apreciar fácilmente la iteración pues la flecha de control de flujo se dirige nuevamente hacia el inicio de la condición (*termino* $>$ 0000001).


```

1  Algoritmo calculo_Numero_e
2      i <- 0
3      suma <- 0
4      factorial <- 1
5      termino <- 1
6      Mientras termino>0.0000001 Hacer
7          Si i=0 Entonces
8              suma <- suma+1
9          SiNo
10             factorial <- factorial*i
11             termino <- 1/factorial
12             suma <- suma + termino
13         FinSi
14         i <- i+1
15     FinMientras
16     Escribir 'El valor del número e es = ', suma
17 FinAlgoritmo

```

Figura 1.14: Pseudocódigo: Controlando el flujo por el valor del término de la serie

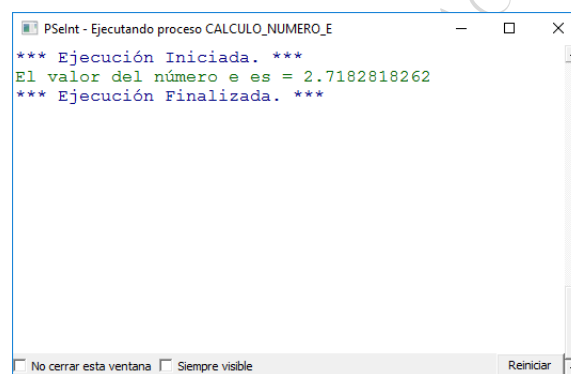


Figura 1.15: Salida: Controlando el flujo por el valor del término

1.4.3. Implementación en Lenguaje C

En esta sección se presenta una propuesta de solución al problema del cálculo del número e utilizando el lenguaje C. Se podrá énfasis en los aspectos de implementación propios del lenguaje de programación. Se recomienda que siga paso a paso cada etapa de la solución utilizando un IDE de C, como por ejemplo Netbeans o Dev C++, y que experimente los diversos casos particulares que se comentan en esta guía.

Listado de n números

En el programa 1.2 se presenta la implementación en el lenguaje C para el listado de los 10 primeros números naturales. Como ya se ha visto previamente, cuando se ejecuta un programa escrito en C, las primeras instrucciones que se procesan son las que se encuentra en la función `main`.

Programa 1.2: Listado de n números

```

1  #include <stdio.h>
2
3  int main() {
4      int i = 0;
5      while (i<10){
6          printf("%d\n", i);
7          i++;
8      }
9      return 0;
10 }

```

Otro detalle importante de implementación que se debe notar en el programa 1.2 se encuentra en la línea

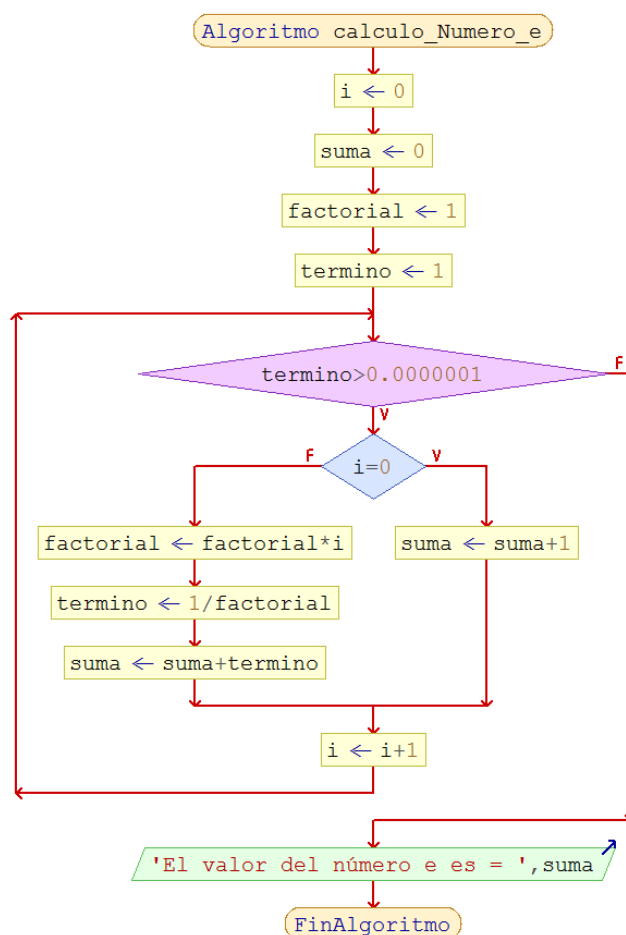


Figura 1.16: Diagrama de Flujo: Controlando el flujo por el valor del término

7. En ella se puede apreciar la instrucción $i++$; . El operador $++$ es un operador unario que lo que hace es incrementar en 1 el operando. Debido a esto, el operando debe ser una variable. En este caso en particular, la instrucción $i++$; sería equivalente a $i=i+1$; . Lo más común en el mundo de la programación en C y otros lenguajes imperativos es la utilización del operador $++$ en lugar del incremento vía asignación $i=i+1$; .

La salida que genera este programa es la siguiente:

0
1
2
3
4
5
6
7
8
9

Para poner en práctica

- ¿Qué ocurre si se cambia la línea 1 por `void main()`?, ¿el programa sigue funcionando?, ¿debe corregirse alguna otra instrucción?
- ¿Qué ocurre si se cambia la línea 7 por `++i`?, ¿el programa sigue funcionando?, ¿el operador `++` se utiliza en la notación prefija, posfija o en ambas?

Acumulando la suma

Ahora se procederá a acumular la suma. Para esto se creará una variable de tipo `double`. Se utiliza el tipo de datos `double` dado que el valor que se desea almacenar es un número real y este tipo de datos ofrece una gran precisión. En el programa 1.3 se puede apreciar la acumulación de la suma. Note que el formato de impresión de un tipo de dato `double` es `"%lf"`.

Programa 1.3: Acumulando la suma

```

1 #include <stdio.h>
2
3 int main() {
4     int i = 0;
5     double suma = 0;
6     while (i<10){
7         printf("%d\n", i);
8         suma = suma + i;
9         i++;
10    }
11    printf("Suma = %lf", suma);
12    return 0;
13 }
```

La salida que genera este programa es la siguiente:

```

0
1
2
3
4
5
6
7
8
9
Suma = 45.000000
```

Para poner en práctica

¿Qué ocurre si se cambia la línea 8 por `suma += i`?, ¿El programa sigue funcionando?

Verificando el primer término de la serie

En el programa 1.4 se presenta la implementación en el lenguaje C para la verificación del primer término de la serie. Observe en particular la línea 8: `if (i==0)`. En este caso se está realizando una comparación, sería incorrecto colocar como condición `(i=0)`. Si colocásemos `(i=0)` en la condición, lo que se estaría haciendo es asignar el valor de 0 a la variable `i`.

Programa 1.4: Verificando el primer término de la serie

```

1 #include <stdio.h>
2
3 int main() {
4     int i = 0;
5     double suma = 0;
6     while (i < 10) {
7         printf("%d\n", i);
8         if (i == 0)
9             suma = suma + 1;
10        else
11            suma = suma + 1 / i;
12        i++;
13    }
14    printf("Suma = %lf", suma);
15    return 0;
16 }

```

La salida que genera este programa es la siguiente:

```

0
1
2
3
4
5
6
7
8
9
Suma = 2.000000

```

Para poner en práctica

¿Qué ocurre si se cambia la línea 8 por `if (i==0)?`, ¿el programa sigue funcionando?, ¿funciona como lo esperado?, ¿por qué?

Convirtiendo enteros en reales

Si ha analizado la salida del programa anterior, verá que se imprime como suma el valor de 2.000000. Al parecer no está sumando las partes decimales a pesar de estar usando una variable de tipo `double`. ¿Qué está ocurriendo? Sucede que el operador de división (`/`) funciona de una manera en particular: cuando los operandos son enteros, realiza una división entera, si al menos un operador es real, realiza una división real. Es decir $5/2$ retornaría 2 y no 2.5, por otro lado $5.0/2$ retornaría 2.5. Este funcionamiento del operador de división (`/`) se realiza de forma independiente al tipo de dato de la variable que recepcionará el resultado de la división, es decir que en términos prácticos no interesa el tipo de dato de la variable `suma` y sí los operandos de la división $1/i$. Como se puede notar en el programa anterior (1.4), los operandos de la división $1/i$ son ambos enteros, por lo tanto se realizará siempre una división entera. ¿Qué se puede hacer para que en la división uno de los operandos se convierta en `double` para que se pueda ejecutar una división real? El lenguaje C proporciona una solución elegante para esto a través de un mecanismo llamado *cast*. Basta anteponer a la variable que queremos transformar, el tipo de datos deseado. El tipo de dato a transformar se coloca entre paréntesis. En el programa 1.5 se puede observar este mecanismo en la línea 11.

Programa 1.5: Convirtiendo enteros en reales

```

1 #include <stdio.h>
2
3 int main() {
4     int i = 0;

```

```

5  double suma = 0;
6  while (i < 10) {
7      printf("%d\n", i);
8      if (i == 0)
9          suma = suma + 1;
10     else
11         suma = suma + (double)1 / i;
12     i++;
13 }
14 printf("Suma = %lf", suma);
15 return 0;
16 }

```

La salida que genera este programa es la siguiente:

```

0
1
2
3
4
5
6
7
8
9
Suma = 3.828968

```

Para poner en práctica

¿Qué ocurre si se cambia la línea 11 por `suma = suma + 1.0/i;`?, ¿el programa sigue funcionando?, ¿funciona como lo esperado?, ¿por qué?

Calculando el factorial de i

El cálculo del factorial se realiza de forma muy similar a lo realizado en el pseudocódigo. A nivel de lógica es en esencia la misma solución. La única diferencia que se debe tener en consideración es que cuando se realiza un programa en C, las variables tienen asociadas un tipo de dato. Dada la definición de factorial presentada anteriormente, lo más natural es elegir como tipo de dato para la variable *factorial* a `int`. En el programa 1.6 se puede observar la implementación propuesta en C definiendo la variable factorial como `int`.

Programa 1.6: Calculando el factorial de i

```

1  #include <stdio.h>
2
3  int main() {
4      int i = 0;
5      double suma = 0;
6      int factorial = 1;
7      while (i < 10) {
8          if (i == 0)
9              suma = suma + 1;
10         else {
11             suma = suma + (double) 1 / i;
12             factorial = factorial*i;
13             printf(" %d %d\n", i, factorial);
14         }
15         i++;
16     }
17     printf("Suma = %lf", suma);
18     return 0;
19 }

```

La salida que genera este programa es la siguiente:

```
1 1
2 2
3 6
4 24
5 120
6 720
7 5040
8 40320
9 362880
Suma = 3.828968
```

Para poner en práctica

Modifique el programa para que también se imprima el factorial de 0. Recuerde que en un buen programa se procura no repetir código por lo que se requiere una solución con una sola instrucción `printf("%d %d", i, factorial);`

Incrementando el número de términos

Analicemos con un poco más de profundidad el uso del tipo de dato `int` para la variable *factorial*. Vamos a realizar un cambio simple en el programa para ampliar los términos de la serie e ir hasta 20. En el programa 1.7 se puede observar el cambio. Básicamente se ha modificado la condición de la instrucción `while`, ahora la condición es `(i < 20)`.

Programa 1.7: Incrementando el número de términos

```
1 #include <stdio.h>
2
3 int main() {
4     int i = 0;
5     double suma = 0;
6     int factorial = 1;
7     while (i < 20) {
8         if (i == 0)
9             suma = suma + 1;
10        else {
11            suma = suma + (double) 1 / i;
12            factorial = factorial*i;
13            printf(" %d %d\n", i, factorial);
14        }
15        i++;
16    }
17    printf("Suma = %lf", suma);
18    return 0;
19 }
```

La salida que genera este programa es la siguiente:

```
1 1
2 2
3 6
4 24
5 120
6 720
7 5040
8 40320
9 362880
```

```

10 3628800
11 39916800
12 479001600
13 1932053504
14 1278945280
15 2004310016
16 2004189184
17 -288522240
18 -898433024
19 109641728
Suma = 4.547740

```

Para pensar

- Observe detalladamente la salida. ¿Se está calculando correctamente el valor del factorial para todos los números?
- ¿El factorial de 13 es el correcto?
- ¿Por qué el factorial de 17 retorna un número negativo?

Cambiando el tipo de dato de la variable de factorial

Como se habrán podido dar cuenta, existen algunos problemas en seleccionar `int` como tipo de dato para la variable *factorial*. En particular los errores comienzan con el factorial de 13. El factorial de 13 es 6227020800 pero el programa 1.7 retorna 1932053504. ¿Qué ha pasado? Resulta que el tipo de dato `int`, como todos los tipos de datos, posee un límite. Este límite dependerá del compilador que se esté usando. El compilador usado para esta guía definía como límite para el `int` el valor de +2147483647. Es por este motivo que el factorial de 13 no se puede representar usando una variable `int` pues supera este límite.

Es importante notar que el lenguaje C no emite ningún mensaje de error en estos casos. La responsabilidad es trasladada al programador, por eso es importante tener presente los límites de los tipos de datos sobre todo cuando se van a procesar números grandes.

¿Cómo se puede solucionar este problema? Se debe buscar un tipo de datos que posea un límite mayor. Para el ejemplo que se está desarrollando, se propone como alternativa la utilización del `double` como tipo de dato para la variable *factorial*. Si bien es cierto el factorial es un valor entero, para este problema en específico, conviene representarlo en un `double` dado que este tiene un mayor límite.

En el programa 1.8 se puede apreciar la propuesta de solución con la variable *factorial* declarada como `double`.

Programa 1.8: Cambiando el tipo de dato de la variable de factorial

```

1  #include <stdio.h>
2
3  int main() {
4      int i = 0;
5      double suma = 0;
6      double factorial = 1;
7      while (i < 20) {
8          if (i == 0)
9              suma = suma + 1;
10         else {
11             suma = suma + (double) 1 / i;
12             factorial = factorial*i;
13             printf(" %d %lf\n", i, factorial);
14         }
15         i++;
16     }
17     printf("Suma = %lf", suma);
18     return 0;
19 }

```

La salida que genera este programa es la siguiente:

```
1 1.000000
2 2.000000
3 6.000000
4 24.000000
5 120.000000
6 720.000000
7 5040.000000
8 40320.000000
9 362880.000000
10 3628800.000000
11 39916800.000000
12 479001600.000000
13 6227020800.000000
14 87178291200.000000
15 1307674368000.000000
16 20922789888000.000000
17 355687428096000.000000
18 6402373705728000.000000
19 121645100408832000.000000
Suma = 4.547740
```

Para pensar

- ¿Qué pasaría si en lugar de utilizar un `double` se utiliza el tipo de dato `unsigned int`? Cambie la línea 6 por `unsigned int factorial = 1;`.
- ¿Qué pasaría si en lugar de utilizar un `double` se utiliza el tipo de dato `long int`? Cambie la línea 6 por `long int factorial = 1;`.
- ¿Qué pasaría si en lugar de utilizar un `double` se utiliza el tipo de dato `unsigned long int`? Cambie la línea 6 por `unsigned long int factorial = 1;`.

Incluyendo el factorial en el término

A continuación se procede a incluir el factorial en el cálculo del término de la serie. En el programa 1.9 se visualiza una propuesta de solución incluyendo el factorial como denominador del término. Recuerde que el cálculo del factorial se debe realizar antes de la acumulación de la variable *suma*.

Programa 1.9: Incluyendo el factorial en el término

```
1 #include <stdio.h>
2
3 int main() {
4     int i = 0;
5     double suma = 0;
6     double factorial = 1;
7     while (i < 20) {
8         if (i == 0)
9             suma = suma + 1;
10        else {
11            factorial = factorial*i;
12            suma = suma + 1 / factorial;
13        }
14        i++;
15    }
16    printf("Suma = %lf", suma);
17    return 0;
18 }
```


La salida que genera este programa es la siguiente:

Suma = 2.718282

Controlando el flujo por el valor del término

Finalmente se realiza el control por el valor del término que se va calculando de la serie. La propuesta final se puede preciar en el programa 1.10.

Programa 1.10: Controlando el flujo por el valor del término

```

1 #include <stdio.h>
2
3 int main() {
4     int i = 0;
5     double suma = 0;
6     double factorial = 1;
7     double termino=1;
8     while (termino>=0.0000001) {
9         if (i == 0)
10             suma = suma + 1;
11         else {
12             factorial = factorial*i;
13             termino = 1 / factorial;
14             suma = suma + termino;
15         }
16         i++;
17     }
18     printf("El valor del número e es = %1.6lf", suma);
19     return 0;
20 }
```

La salida que genera este programa es la siguiente:

El valor del número e es = 2.718282

1.5. 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 implementará un programa en lenguaje C que calcule el valor aproximado del seno usando la serie de Taylor descrita previamente. El programa deberá incluir solamente términos que en su valor absoluto tengan hasta 6 dígitos significativos.

Recordar que:

Un radián (rad) es la medida de un ángulo central que subtiende un arco cuya longitud es igual al del radio de la circunferencia. Puesto que la longitud de una circunferencia de radio r es igual a 2π , se tiene que $360^\circ = 2\pi$ radianes.

1.5.1. Listando números impares

La serie que se utilizará para calcular el valor del seno de x es una serie que se basa en una sucesión de números impares que se utilizan como exponente y para calcular el factorial. Se iniciará la propuesta de

solución listando los números impares. Al igual que en la solución anterior, no nos preocuparemos en este momento por la cantidad de término, asumiremos por practicidad que se imprimirán los números impares en el rango $[1..10[$.

El programa 1.11 realiza justamente esta labor. Para determinar si un número es impar, se utiliza el operador de módulo (%). El operador % es un operador binario que retorna el resto de la división entera entre los dos operandos. Los operandos deben ser número enteros. Para determinar si un número es impar, basta preguntar si el resto que se obtiene de dividir el número entre el valor 2 es igual a 1, pues esta es la característica de los números impares (no son divisibles por 2). Esto se puede apreciar en la línea 6 del programa 1.11.

Programa 1.11: Listando números impares

```

1 #include <stdio.h>
2
3 int main() {
4     int i = 1;
5     while (i < 10) {
6         if (i % 2 == 1)
7             printf("%d\n", i);
8         i++;
9     }
10    return 0;
11 }
```

La salida que genera este programa es la siguiente:

```

1
3
5
7
9
```

Para poner en práctica

- ¿Qué cambios debería hacer en el programa si en lugar de imprimir los números impares se solicita imprimir los números pares?
- ¿Qué cambios debería hacer en el programa si en lugar de imprimir los números impares se solicita imprimir los números múltiplos de 3?
- ¿Qué sucede si eliminamos la línea 6 y en lugar de incrementar i en 1 lo incrementamos en 2?, ¿se obtiene el mismo resultado?

1.5.2. Leyendo X

El siguiente paso es leer el valor del ángulo x en radianes. Se definirá la variable x como tipo `double` dado que los valores radianes son números reales. Para la lectura se utilizará la función `scanf`. Recuerde que el primer parámetro de la función `scanf` debe contener el formato del tipo de dato que se desea leer, en este caso al ser un `double` el formato es “%lf”. En el segundo parámetro se debe indicar la dirección de memoria de la variable que se desea leer. Para obtener la dirección de memoria de una variable, se utiliza el operador de dirección (&). Para el ejemplo, la dirección de memoria de la variable x se puede obtener con la siguiente expresión `&x`.

En el programa 1.12 se puede apreciar la inclusión de la lectura del ángulo. Al ejecutar el programa solicitará que se ingrese el ángulo en radianes. Por el momento no interesa el valor que se ingrese dado que no se usará todavía para el cálculo del término. Para efectos de prueba puede colocar cualquier valor para el ángulo x .

Programa 1.12: Leyendo X

```

1 #include <stdio.h>
2
```

```

3 int main() {
4     int i = 1;
5     double x;
6     printf("Ingrese ángulo en radianes: ");
7     scanf("%lf", &x);
8     while (i<10){
9         if (i%2==1)
10            printf("%d\n", i);
11        i++;
12    }
13    return 0;
14 }

```

Asumiendo que se ingresó como valor de x el número 3, la salida que genera este programa sería la siguiente:

Ingrese ángulo en radianes: 3

1
3
5
7
9

Para pensar

¿Qué pasa si en lugar de colocar `scanf("%lf", &x);` se coloca `scanf("%lf", x);`?, ¿qué intenta hacer el compilador?, ¿el programa puede ejecutarse?

1.5.3. Sumando los números impares

A continuación se procederá a acumular los números impares en la variable *suma*. Definiremos esta variable como *double* dado que los valores que se incluirán posteriormente serán reales. En el programa 1.13 se puede apreciar la acumulación de los números impares en la línea 11. Al ejecutar el programa solicitará que se ingrese el ángulo en radianes. Por el momento no interesa el valor que se ingrese dado que no se usará todavía para el cálculo del término. Para efectos de prueba puede colocar cualquier valor para el ángulo x .

Programa 1.13: Sumando los números impares

```

1 #include <stdio.h>
2
3 int main() {
4     int i = 1;
5     double x, suma=0;
6     printf("Ingrese ángulo en radianes: ");
7     scanf("%lf", &x);
8     while (i<10){
9         if (i%2==1){
10            printf("%d\n", i);
11            suma = suma + i;
12        }
13        i++;
14    }
15    printf("suma = %lf\n", suma);
16    return 0;
17 }

```

Asumiendo que se ingresó como valor de x el número 3, la salida que genera este programa sería la siguiente:

Ingrese ángulo en radianes: 3

1
3

```

5
7
9
suma = 25.000000

```

1.5.4. Calculando el factorial

A continuación se procederá a calcular el factorial. Definiremos la variable *factorial* como *double* por el motivo ya comentado en el ejercicio anterior. En el programa 1.14 se puede apreciar el cálculo del factorial en la línea 12. Al ejecutar el programa solicitará que se ingrese el ángulo en radianes. Por el momento no interesa el valor que se ingrese dado que no se usará todavía para el cálculo del término. Para efectos de prueba puede colocar cualquier valor para el ángulo x .

Programa 1.14: Calculando el factorial

```

1 #include <stdio.h>
2
3 int main() {
4     int i = 1;
5     double x, suma=0, factorial=1;
6     printf("Ingrese ángulo en radianes: ");
7     scanf("%lf", &x);
8     while (i<10){
9         if (i%2==1){
10             suma = suma + i;
11         }
12         factorial = factorial * i;
13         printf("%d %lf\n", i, factorial);
14         i++;
15     }
16     printf("suma = %lf\n", suma);
17     return 0;
18 }

```

Asumiendo que se ingresó como valor de x el número 3, la salida que genera este programa sería la siguiente:

```

Ingrese ángulo en radianes: 3
1 1.000000
2 2.000000
3 6.000000
4 24.000000
5 120.000000
6 720.000000
7 5040.000000
8 40320.000000

```

Para poner en práctica

- ¿Qué cambios deberían hacerse en el programa si se deseara calcular el factorial dentro de la estructura selectiva?
- Si en lugar de incrementar i en 1 incrementamos i en 2. ¿Qué cambios se deberían hacer al programa para que siga funcionando correctamente? Analice el cálculo del factorial y la estructura selectiva.

1.5.5. Formando el término de la suma

A continuación se procederá a ir formando el término de la serie para acumular correctamente la sumatoria. Básicamente dividimos x entre el *factorial* calculado. Note que en este caso no es necesario realizar *cast* pues

los operandos son reales, lo que produce una división real. En el programa 1.15 se puede apreciar como se va formando el término en la línea 11.

Programa 1.15: Formando el término de la suma

```

1  #include <stdio.h>
2
3  int main() {
4      int i = 1;
5      double x, suma=0, factorial=1;
6      printf("Ingrese ángulo en radianes: ");
7      scanf("%lf", &x);
8      while (i<10){
9          factorial = factorial * i;
10         if (i%2==1){
11             suma = suma + x/factorial;
12         }
13         i++;
14     }
15     printf("suma = %lf\n", suma);
16     return 0;
17 }

```

Asumiendo que se ingresó como valor de x el número 3.14, la salida que genera este programa sería la siguiente:

```

Ingrese ángulo en radianes: 3.14
suma = 3.690132

```

1.5.6. Gestionando las potencias en el término

Ahora analizaremos el numerador del término. Como se puede apreciar en la serie de Taylor para el seno, el numerador de cada término de la serie es de la forma x^i . Pero, ¿cómo realizar potencias en lenguaje C? Existe una función llamada `pow` que se puede usar para estos fines. `pow` es el diminutivo de *power* que en español se puede traducir como potencia. Para usar la función `pow` debe incluirse el archivo de cabecera `math.h` pues en este archivo se encuentran las declaraciones necesarias para invocar a la librería matemática. `pow` recibe dos parámetros de tipo `double` y en términos prácticos `pow(x, i)` es equivalente a x^i . En el programa 1.16 se puede apreciar el cálculo del término usando la función `pow(x, i)` para gestionar las potencias de la serie.

Programa 1.16: Gestionando las potencias en el término

```

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

```

Asumiendo que se ingresó como valor de x el número 3.14, la salida que genera este programa sería la siguiente:

```

Ingrese ángulo en radianes: 3.14
suma = 11.522477

```

1.5.7. Gestionando el signo en el término

Se debe ahora gestionar el signo de la serie. Si se analiza detalladamente la serie podrá darse cuenta de que el signo inicia con $+$ luego cambia a $-$ luego a $+$ y así sucesivamente. ¿Cómo se puede gestionar el signo en el lenguaje C? Se utiliza en realidad un artilugio matemático más que un elemento de programación. Se sabe que el número 1 es el elemento neutro de la multiplicación. Además también se sabe que 1×-1 retorna -1 y que -1×-1 retorna 1. Entonces lo que se hace para alternar el signo es utilizar una variable para que almacene 1 e ir multiplicándola en cada iteración por -1 .

En el programa 1.17 se presenta una propuesta para gestionar el signo. Se ha creado una variable llamada *signo* para este fin (ver línea 5), esta variable se multiplica por -1 luego de usarla para calcular el término (ver línea 13). Con esta alteración, el programa ya podrá aproximar el valor del seno para algún valor conocido de x .

Al ejecutar el programa solicitará que se ingrese el ángulo en radianes. Ingresaremos el valor de 1.57 que es el valor aproximado de $\frac{\pi}{2}$. Como se sabe, el valor del seno de $\frac{\pi}{2}$ es igual a 1. Por lo tanto se espera como respuesta un valor aproximada a 1.

Programa 1.17: Gestionando el signo en el término

```

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;
7      printf("Ingrese ángulo en radianes: ");
8      scanf("%lf", &x);
9      while (i<100){
10         factorial = factorial * i;
11         if (i%2==1){
12             suma = suma + signo*pow(x,i)/factorial;
13             signo = signo*(-1);
14         }
15         i++;
16     }
17     printf("suma = %lf\n", suma);
18     return 0;
19 }
```

Asumiendo que se ingresó como valor de x el número 1.57, la salida que genera este programa sería la siguiente:

```

Ingrese ángulo en radianes: 1.57
suma = 1.000000
```

1.5.8. Controlando el ciclo por el valor del término

Para finalizar el problema se debe controlar la iteración mediante el valor del término. Se requiere que el ciclo iterativo pare cuando el término que se está calculando tenga menos de 6 dígitos de precisión. La situación es muy similar al caso del número e visto anteriormente. En esencia es la misma solución. La única diferencia es que se debe tomar el valor absoluto dado que algunos términos serán negativos. Para esto se puede usar la función `fabs` cuyo prototipo también está declarado en el archivo de cabecera `math.h`. En el programa 1.18 se presenta una propuesta para el problema planteado.

Programa 1.18: Controlando el ciclo por el valor del término

```

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 ángulo en radianes: ");
8  scanf("%lf", &x);
9  while (fabs(termino)>=0.0000001){
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 }
18 printf("suma = %lf\n", suma);
19 return 0;
20 }
```

Para pensar

Otra forma de implementar esta serie de Taylor es a través de la siguiente sumatoria: $\sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$.

Proponga un programa que realice tal implementación.

Capítulo 2

Ejercicios propuestos

2.1. Nivel básico

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 el lenguaje C conforme a los temas revisados en las guías *preliminar*, #1 y #2 del curso de Fundamentos de Programación.

2.1.1. Mayor y menor número de una lista

La radiación solar es la energía emitida por el sol en forma de radiación electromagnética que llega a la atmósfera. Para medirla se utiliza un instrumento denominado piranómetro. Los piranómetros por lo general registran la radiación solar en vatios o watt por metro cuadrado (W/m^2).

Se solicita que lea una cantidad de ciudades, luego ingrese un listado de ciudades con su respectiva radiación solar y determine las dos ciudades que poseen mayor radiación solar. Por comodidad las ciudades estarán representadas por un carácter.

A continuación se presentan unos ejemplos de ejecución ¹:

```
Ingrese la cantidad de ciudades: 1
Ingrese número ciudad: M
Ingrese valor de radiación: -1

No se ingresaron valores de radiación para ninguna ciudad.
```

```
Ingrese la cantidad de ciudades: 1
Ingrese nombre de la ciudad: M
Ingrese valor de radiación: 5.7

La primera ciudad con mayor radiación es M (5.70).
Solo se registró una ciudad.
```

```
Ingrese la cantidad de ciudades: 5
Ingrese nombre de la ciudad: M
Ingrese valor de radiación: 5.7
```

¹Los datos han sido obtenidos a través de Atlas de la Energía Solar del Perú disponible a través de la URL http://cedecap.org.pe/uploads/biblioteca/80bib_arch.pdf.

Ingrese nombre de la ciudad: A
Ingrese valor de radiación: 4.06

Ingrese nombre de la ciudad: S
Ingrese valor de radiación: 5.92

Ingrese nombre de la ciudad: L
Ingrese valor de radiación: 7.03

Ingrese nombre de la ciudad: B
Ingrese valor de radiación: 4.56

La primera ciudad con mayor radiación es L (7.03).
La segunda ciudad con mayor radiación es S (5.92).

Variación al problema

Dado el listado de ciudades con su respectiva radiación solar como la descrita anteriormente, determine las dos ciudades que poseen menor radiación solar. Usando los datos del último caso de prueba, su solución deberá mostrar:

La primera ciudad con menor radiación es A (4.06).
La segunda ciudad con menor radiación es B (4.56).

2.1.2. Conteo de frecuencias

Las variables categóricas se caracterizan por representar una cantidad finita de posibles valores a diferencia de las variables numéricas que pueden contener, en principio, un número infinito de posibilidades. Las variables categóricas se usan típicamente en las encuestas de opinión para recopilar datos relativos a determinado tema. Por ejemplo, ante la pregunta ¿Cuál es su región de residencia?, las posibles opciones de respuesta podrían ser:

- Región Norte
- Región Sur
- Región Centro
- Región Oriente

Se le pide que ingrese la cantidad de respuestas a procesar, luego lea un conjunto de respuestas a la pregunta ¿Cuál es su región de residencia?, calcule e imprima la frecuencia de respuestas de cada región así como el porcentaje de las mismas. Por facilidad asuma que para la Región Norte se utilizará el carácter N, para la Región Sur se utilizará el carácter S, para la Región Centro se utilizará el carácter C y para la Región Oriente se utilizará el carácter O.

A continuación se presenta un ejemplo de ejecución:

Ingrese la cantidad de respuestas a procesar: 8

Ingrese la región de residencia del encuestado 1: C
Ingrese la región de residencia del encuestado 2: N
Ingrese la región de residencia del encuestado 3: N
Ingrese la región de residencia del encuestado 4: S
Ingrese la región de residencia del encuestado 5: O

Ingrese la región de residencia del encuestado 6: S
 Ingrese la región de residencia del encuestado 7: O
 Ingrese la región de residencia del encuestado 8: N

Cantidad de residentes en Región Norte: 3
 Cantidad de residentes en Región Sur: 2
 Cantidad de residentes en Región Centro: 1
 Cantidad de residentes en Región Oriente: 2
 Cantidad de blancos o nulos: 0

Porcentaje de residentes en Región Norte: 37.500%
 Porcentaje de residentes en Región Sur: 25.000%
 Porcentaje de residentes en Región Centro: 12.500%
 Porcentaje de residentes en Región Oriente: 25.000%
 Porcentaje de blancos o nulos: 0.00%

2.1.3. Medidas de tendencia central y de dispersión

En la estadística, las medias de tendencia central indican dónde se sitúa un dato dentro de una distribución. Por otro lado, las medidas de dispersión, indican si esos datos están próximos entre sí o si están dispersos. Ambos tipos de medidas ayudan en la caracterización de una distribución de datos.

Se le solicita que ingrese una cantidad de números a procesar y luego, dada una lista de números, determine las principales medidas de tendencia central y de dispersión. No se sabe de antemano cuántos números existen en la lista por lo que deberá controlar el flujo de ingreso de datos a través de un centinela. El valor del centinela deberá ser solicitado al usuario.

Entre las principales medidas de tendencia central que deberá calcular se encuentran:

- Media aritmética.
- Media geométrica.
- Media armónica.

Entre las principales medidas de dispersión que deberá calcular se encuentran:

- Rango.
- Desviación estándar.
- Varianza.

Medidas de tendencia central: media aritmética

$$media\ aritmética = \overline{x_a} = \frac{\sum_{i=1}^n x_i}{n} = x_1 + x_2 + x_3 + \cdots + x_{n-1} + x_n$$

Medidas de tendencia central: media geométrica

$$media\ geométrica = \overline{x_g} = \sqrt[n]{\prod_{i=1}^n x_i} = \sqrt[n]{x_1 \times x_2 \times x_3 \times \cdots \times x_{n-1} \times x_n}$$

Medidas de tendencia central: media armónica

$$\text{media armónica} = H = \frac{n}{\sum_{i=1}^n \frac{1}{x_i}} = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_3} + \cdots + \frac{1}{x_{n-1}} + \frac{1}{x_n}}$$

Medidas de dispersión: varianza

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i - \bar{x})^2$$

Medidas de dispersión: desviación estándar

$$\sigma = \sqrt{\sigma^2}$$

A continuación se presentan un ejemplo de ejecución:

Ingrese la cantidad de números a procesar: 6

Ingrese valor 1 de la distribución: 3

Ingrese valor 2 de la distribución: 2

Ingrese valor 3 de la distribución: 1

Ingrese valor 4 de la distribución: 4

Ingrese valor 5 de la distribución: 1

Ingrese valor 6 de la distribución: 2

La media aritmética es: 2.166666667

La media geométrica es: 1.906368586

La media armónica es: 1.674418605

Rango: [1..4]

La varianza es: 1.366666667

La desviación estándar es: 1.169045194

Sugerencia para el cálculo de la varianza

Como podrá notar en la fórmula del cálculo de la varianza, se requiere la media aritmética \bar{x} para poder calcularla. ¿Cómo realizar el cálculo de la varianza si es que no se sabe *a priori* la media aritmética?

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i - \bar{x})^2$$

Básicamente lo que se hará es el desarrollo de los cuadrados. Recuerde que $(a + b)^2 = a^2 + 2ab + b^2$.

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i^2 - 2x_i\bar{x} + \bar{x}^2)$$

Separando los términos en diferentes sumatorias se obtiene:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i^2) + \frac{1}{N} \sum_{i=1}^n (-2x_i\bar{x}) + \frac{1}{N} \sum_{i=1}^n (\bar{x}^2)$$

Analicemos el término $\frac{1}{N} \sum_{i=1}^n (\bar{x}^2)$: en esta sumatoria el término \bar{x}^2 se suma n veces y luego se divide entre n . Esto equivale a $\frac{n \times \bar{x}^2}{n}$ que simplificando equivale a \bar{x}^2 . Reemplazando esta equivalencia en la fórmula se tiene:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i^2) + \frac{1}{N} \sum_{i=1}^n (-2x_i\bar{x}) + \bar{x}^2$$

Analicemos ahora el término $\frac{1}{N} \sum_{i=1}^n (-2x_i\bar{x})$: esto equivale a $-2\bar{x}(\frac{1}{N} \sum_{i=1}^n (x_i))$. Note además que $\frac{1}{N} \sum_{i=1}^n (x_i)$ equivale a \bar{x} . Luego de realizar este análisis se concluye que $\frac{1}{N} \sum_{i=1}^n (-2x_i\bar{x})$ equivale a $-2\bar{x}^2$. Reemplazando esta equivalencia en la fórmula se tiene:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i^2) - 2\bar{x}^2 + \bar{x}^2$$

Operando se obtiene finalmente:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i^2) - \bar{x}^2$$

Esta forma de calcular σ^2 es interesante en términos computacionales pues no requiere de la \bar{x} al momento de la sumatoria. De esta manera se puede calcular la suma de los cuadrados de forma independiente de la \bar{x} haciendo viable el cálculo de la varianza en una simple iteración.

2.1.4. Clasificación de un triángulo - adaptado del examen parcial 2019-0

Los triángulos se pueden clasificar de acuerdo a la magnitud de sus ángulos y de acuerdo a la magnitud de sus lados tal como se muestra en la tabla 2.1. Se le pide que lea un caracter que represente el criterio de clasificación que requiere el usuario (A para clasificar por ángulos, L para clasificar por lados), luego lea la cantidad de conjunto de datos a procesar, luego, por cada conjunto de datos lea los 3 datos (ángulos o lados según sea el criterio) del triángulo y usando estos datos clasifique cada triángulo según el criterio ingresado. Asuma que los ángulos se ingresan usando grados sexagesimales.

A continuación se presentan algunos ejemplos de ejecución:

Ingrese criterio de calificación: A
 Ingrese la cantidad de conjuntos de datos a procesar: 4

Tipos de triángulos según sus ángulos		Tipos de triángulos según sus lados	
Triángulo acutángulo	Sus tres ángulos son menores a 90 sexagesimales.	Triángulo equilátero	Tiene los tres lados iguales.
Triángulo rectángulo	Uno de sus ángulos mide 90 sexagesimales.	Triángulo isósceles	Tiene dos lados iguales y el tercero distinto.
Triángulo obtusángulo	Uno de sus ángulos es mayor a 90 sexagesimales.	Triángulo escaleno	Tiene sus tres lados distintos.

Tabla 2.1: Clasificación del triángulo.

```

Ingrese ángulos (a, b, c) del triángulo 1: 60 60 60
El triángulo es acutángulo
Ingrese ángulos (a, b, c) del triángulo 4: 60 90 90
Los ángulos ingresados no corresponden a un triángulo.
Ingrese ángulos (a, b, c) del triángulo 2: 30 90 60
El triángulo es rectángulo
Ingrese ángulos (a, b, c) del triángulo 3: 110 40 30
El triángulo es obtusángulo

```

```

Ingrese criterio de calificación: L
Ingrese la cantidad de conjuntos de datos a procesar: 4

Ingrese lados (l1, l2, l3) del triángulo 1: 5 5 5
El triángulo es equilátero
Ingrese lados (l1, l2, l3) del triángulo 2: 6 3 6
El triángulo es isósceles
Ingrese lados (l1, l2, l3) del triángulo 3: 5 3 7
El triángulo es escaleno
Ingrese lados (l1, l2, l3) del triángulo 4: 10 1 1
Los lados ingresados no corresponden a un triángulo.

```

Recordar que:

- Según el teorema de la desigualdad del triángulo “La suma de las longitudes de cualesquiera de los lados de un triángulo es mayor que la longitud del tercer lado”.
- La suma de los ángulos internos de un triángulo es igual a 180 grados sexagesimales.

2.1.5. Cuadrado de un número

Una forma de calcular el cuadrado de un número n es sumando los n primeros números impares. De esta forma:

$$n^2 = \sum_{i=1}^n (2i - 1)$$

Se le pide que lea un número entero n cualquiera e imprima su cuadrado utilizando para el cálculo, la suma de los n primeros números impares.

A continuación se presentan unos ejemplos de ejecución:

Ingrese número n: 3

La suma es $S = 1 + 3 + 5 = 9$

3 al cuadrado es 9

Ingrese número n: -4

La suma es $S = 1 + 3 + 5 + 7 = 16$

-4 al cuadrado es 16

2.1.6. Convergencia de series

Se sabe que una serie es convergente si la sucesión de sumas parciales de dicha serie tiene un límite finito. Dada la siguiente serie:

$$\sum_{n=1}^{\infty} \frac{1}{2^n}$$

Se le pide que verifique si efectivamente esta serie converge a 1. Para esto deberá solicitar la cantidad de términos que desea utilizar en la serie, calcular y mostrar todas las sumas parciales para dicha serie, calcular y mostrar el error absoluto de cada suma parcial. Asuma que la serie converge si es que el error absoluto es menor que 0.001.

A continuación se presenta un ejemplo de ejecución:

Ingrese cantidad de términos: 10

Sumas parciales

S1=0.5000000000, error parcial=0.5000000000

S2=0.7500000000, error parcial=0.2500000000

S3=0.8750000000, error parcial=0.1250000000

S4=0.9375000000, error parcial=0.0625000000

S5=0.9687500000, error parcial=0.0312500000

S6=0.9843750000, error parcial=0.0156250000

S7=0.9921875000, error parcial=0.0078125000

S8=0.9960937500, error parcial=0.0039062500

S9=0.9980468750, error parcial=0.0019531250

S10=0.9990234375, error parcial=0.0009765625

Suma total = 0.9990234375, error absoluto=0.0009765625

La serie converge.

Recordar que:

Las sumas parciales S_k de una serie se define como la suma de la sucesión a_n desde a_1 hasta a_k .

$$S_k = \sum_{i=1}^k a_i = a_1 + a_2 + a_3 + \cdots + a_k$$

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

# términos	Suma total
1	≈ 0.5000000000
2	≈ 0.7500000000
9	≈ 0.9980468750
10	≈ 0.9990234375
14	≈ 0.9999389648
26	≈ 0.9999999851
32	≈ 0.9999999998

Variación al problema

Se le pide que solicite el máximo error absoluto permitido y teniendo este valor en consideración calcule e imprima el valor de todas las sumas parciales hasta encontrar una en donde el error sea menor o igual que el máximo error absoluto permitido. Deberá indicar además la cantidad de términos que fueron necesarios. En caso el usuario ingrese un error absoluto ≤ 0 o ≥ 1 , el programa deberá emitir el siguiente mensaje de error El error absoluto debe encontrarse en el rango $]0,1[$.

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

error absoluto máximo	suma parcial	error absoluto calculado	# términos
0.500000	0.500000	≈ 0.500000	1
0.100000	0.937500	≈ 0.062500	4
0.005000	0.996093	≈ 0.003906	8
0.000080	0.999938	≈ 0.000061	14

Recuerde que:

El error absoluto se define como la diferencia entre el valor real y el valor aproximado o calculado, en valor absoluto.

$$e = |\text{valor real} - \text{valor calculado}|$$

2.2. Nivel intermedio

2.2.1. Cálculo de $\ln(1+x)$

Dado un número real x tal que $|x| < 1$, se le pide que calcule el $\ln(1+x)$ usando la serie de Taylor. Se le pide que solicite la cantidad de términos que desea utilizar en la serie, solicite el número x , muestre todos los términos de la serie en cada iteración y presente al final el valor de $\ln(1+x)$ usando para este fin la serie de Taylor. Deberá verificar que el número x cumpla que $|x| < 1$.

Cálculo de $\ln(1+x)$ usando la serie de Taylor:

$$\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \dots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}$$

A continuación se presentan unos ejemplos de ejecución:

Ingrese el valor de x: 1.6

No se puede realizar el cálculo. Debe cumplirse que $|x| < 1$

Ingrese el valor de x: 0.5

Ingrese cantidad de términos: 5

Término 1: 0.5000000000

Término 2: -0.1250000000

Término 3: 0.0416666667

Término 4: -0.0156250000

Término 5: 0.0062500000

El $\ln(1+0.50) = 0.4072916667$

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

x	# términos	$\ln(1+x)$
0.70	100	≈ 0.5306282511
0.23	750	≈ 0.2070141694
0.40	3	≈ 0.3413333333
0.40	3000	≈ 0.3364722366

Variación al problema #1

Se le pide que solicite el número real x y un número real denominado error > 0 y < 1 . Deberá imprimir los términos de la serie hasta encontrar un término que en valor absoluto sea menor o igual al error ingresado. Deberá indicar además la cantidad de términos usados en la serie. En caso el usuario ingrese un error ≤ 0 o ≥ 1 , el programa deberá emitir el siguiente mensaje de error El error debe encontrarse en el rango $]0,1[$.

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

x	error	i-ésimo término	$\ln(1+x)$	# términos
0.4	0.005	0.0020480000	≈ 0.0020480000	5
0.39	0.0001	-0.0000669001	≈ 0.3292865684	8
-0.25	0.000005	-0.0000019073	≈ -0.2876815251	8

Variación al problema #2

Realice las adecuaciones correspondientes para usar la serie de Taylor para calcular el valor de $\ln\left(\frac{1}{1-x}\right)$, considerando que:

$$\ln\left(\frac{1}{1-x}\right) = x + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{1}{4}x^4 + \cdots = \sum_{i=1}^{\infty} \frac{x^i}{i}$$

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

x	# términos	$\ln\left(\frac{1}{1-x}\right)$
0.70	100	≈ 1.2039728043
0.23	750	≈ 0.2613647641
0.40	3	≈ 0.5013333333

x	error	i-ésimo término	$\ln\left(\frac{1}{1-x}\right)$	# términos
0.4	0.005	0.0020480000	≈ 0.5097813333	5
0.39	0.0001	0.0000669001	≈ 0.4942604882	8
-0.25	0.000005	0.0000019073	≈ -0.2231432052	8

2.2.2. Cálculo de e^x

Dado un número real x cualquiera, se le pide que calcule el valor de e^x usando la serie de Taylor. Se le pide que solicite la cantidad de términos que desea utilizar en la serie, el número real x y presente al final el valor de e^x usando para este fin la serie de Taylor. Deberá validar que el número de términos a usar sea ≤ 23 .

Cálculo de e^x usando la serie de Taylor:

$$e^x = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

A continuación se presentan unos ejemplos de ejecución:

Ingrese el valor de x : 1.6

Ingrese la cantidad de términos: 25

No se puede realizar el cálculo. La cantidad de términos debe ser menor o igual a 23.

Ingrese el valor de x : 2.3

Ingrese la cantidad de términos: 15

$e^{2.3} = 9.9741822171$

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

x	# términos	e^x
1.7	5	≈ 5.3118375000
2.9	10	≈ 18.1585497247
-1.8	15	≈ 0.1652988929
-2.1	20	≈ 0.1224564283

2.2.3. Cálculo de la potencia de un número

Dado un número real x cualquiera y un número $n \geq 0$, se desea calcular el valor de x^n usando para ello la serie de Taylor.

Cálculo de $(1+x)^n$ usando la serie de Taylor:

$$(1+x)^n = \sum_{i=0}^{\infty} \binom{n}{i} x^i = \frac{n!}{0!(n)!} x^0 + \frac{n!}{1!(n-1)!} x^1 + \frac{n!}{2!(n-2)!} x^2 + \frac{n!}{3!(n-3)!} x^3 + \frac{n!}{4!(n-4)!} x^4 + \dots$$

La misma serie puede ser vista como:

$$(1+x)^n = \sum_{i=0}^{\infty} \binom{n}{i} x^i = 1 + \frac{n(n-1)!}{1!(n-1)!} x^1 + \frac{n(n-1)(n-2)!}{2!(n-2)!} x^2 + \frac{n(n-1)(n-2)(n-3)!}{3!(n-3)!} x^3 + \dots$$

$$(1+x)^n = \sum_{i=0}^{\infty} \binom{n}{i} x^i = 1 + \frac{n}{1!} x^1 + \frac{n(n-1)}{2!} x^2 + \frac{n(n-1)(n-2)}{3!} x^3 + \frac{n(n-1)(n-2)(n-3)}{4!} x^4 + \dots$$

Implemente la solución usando la serie antes mencionada. Finalice el cálculo cuando encuentre un término con el valor de cero. En cada iteración deberá imprimir el número de la iteración, el valor del numerador del término, el valor del factorial (el denominador) y el valor de x^i .

Recordar que:

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 $p \geq r$. Matemáticamente, un número combinatorio se expresa como:

$$C_r^p = \binom{p}{r} = \frac{p!}{r! \times (p-r)!}$$

A continuación se presenta un ejemplo de ejecución:

Ingrese el valor de x: 3
Ingrese el valor de n: 5

Iteración i=1, numerador=5, 1!=1, x^1=2.000000
Iteración i=2, numerador=20, 2!=2, x^2=4.000000
Iteración i=3, numerador=60, 3!=6, x^3=8.000000

```
Iteración i=4, numerador=120, 4!=24, x^4=16.000000
Iteración i=5, numerador=120, 5!=120, x^5=32.000000

3^5 = 243.0000000000
```

Restricciones

- Para la implementación en PSeInt no podrá usar el operador $^$.
- Para la implementación en el lenguaje C no podrá usar la función `pow`.

2.2.4. Funciones trigonométricas: coseno

Dado un número real x que representa un ángulo en radianes y un número entero n que representa la cantidad de términos a usar, se solicita que calcule el valor del coseno de dicho ángulo usando la siguiente serie de Taylor:

Taylor: $\sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$. Deberá verificar que la cantidad de términos sea ≤ 11 .

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

x	# términos	$\cos(x)$
0.50	11	≈ 0.877583
-0.20	6	≈ 0.980067
-0.90	8	≈ 0.621610
3.14	10	≈ -0.999999

Variación al problema

Se le pide que solicite el valor del ángulo x en radianes y un número real denominado error > 0 y < 1 . Deberá calcular el valor del coseno hasta encontrar un término que en valor absoluto sea menor o igual al error ingresado. Deberá indicar además la cantidad de términos usados en la serie. En caso el usuario ingrese un error ≤ 0 o ≥ 1 , el programa deberá emitir el siguiente mensaje de error El error debe encontrarse en el rango $]0,1[$.

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

x	error	$\cos(x)$	# términos
0.50	0.01	≈ 0.877604	4
-0.20	0.3	≈ 0.980000	2
-0.90	0.2	≈ 0.622337	4
3.14	0.5	≈ -0.976140	8

2.2.5. Distancia de Hamming

En 1950 Richard Hamming, un matemático americano, creó una métrica muy utilizada en el campo de la teoría de la información la denominada actualmente distancia de Hamming. La distancia de Hamming es una

métrica que expresa la distancia entre dos objetos calculada por el número de diferencias entre sus pares. Se utiliza principalmente para análisis de cadenas de caracteres y de representación binaria, pero también puede ser útil para variables numéricas.

Cálculo de la distancia de Hamming para números enteros:

Si n_1 y n_2 son dos números enteros positivos que poseen una cantidad de dígitos nd , la distancia de Hamming d se puede calcular como sigue:

$$d(n_1, n_2) = \sum_{k=1}^{nd} \text{Dígito}(n_1, k) \neq \text{Dígito}(n_2, k)$$

Donde $\text{Dígito}(n, k)$ representa el k -ésimo dígito del número n .

Se le pide que dados dos números enteros n_1 y n_2 , verifique si ambos poseen la misma cantidad de dígitos y si cumplen con esta condición, deberá calcular e imprimir la distancia Hamming que existe entre ambos números.

A continuación se presentan algunos ejemplos de ejecución:

Ingrese número n1: 101

Ingrese número n2: 4568

Los números no poseen la misma cantidad de dígitos

Ingrese número n1: 467

Ingrese número n2: 477

La distancia de Hamming entre 467 y 477 es 1.

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

n_1	n_2	distancia Hamming
123456	123456	0
32567	42568	2
2147483641	2145473651	3
29111976	25071978	4

2.2.6. Distancia de Chebyshev

Debe leer primero la cantidad de pares de puntos a procesar y luego por cada conjunto de pares de puntos $x = (x_1, x_2)$ e $y = (y_1, y_2)$ que pertenecen a un plano cartesiano, se le pide que calcule y muestre para cada par de puntos, la distancia Chebyshev que existe entre ellos.

A continuación se presentan algunos ejemplos de ejecución:

Ingrese la cantidad de pares de puntos a procesar: 3

Ingrese punto 1 (x1, x2): 1 2

Ingrese punto 2 (y1, y2): 1 5

La distancia de Chebyshev es 3.

Ingrese punto 1 (x_1 , x_2): 2 3

Ingrese punto 2 (y_1 , y_2): 4 7

La distancia de Chebyshev es 4.

Ingrese punto 1 (x_1 , x_2): -1 2

Ingrese punto 2 (y_1 , y_2): -1 2

La distancia de Chebyshev es 0.

Cálculo de la distancia de Chebyshev:

Dados dos puntos $x = (x_1, x_2)$ e $y = (y_1, y_2)$ que $\in \mathbb{R}^2$, la distancia de Chebyshev se puede calcular como:

$$d(x, y) = \max\{|x_1 - y_1|, |x_2 - y_2|\}$$

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta. En la siguiente tabla el punto 1 está representado por los valores x_1 e x_2 mientras que el punto 2 por los valores y_1 e y_2 .

x_1	x_2	y_1	y_2	distancia Chebyshev
4	6	8	9	4
7	5	8	2	3
5	-3	6	-2	1
2	1	-6	7	8
3	-5	4	6	11

2.2.7. Distancia de Manhattan

Dado un punto origen P_0 y una serie de puntos adicionales en un espacio n dimensional de hasta 3 dimensiones, se le pide que calcule la menor y la mayor distancia que existe desde el punto origen P_0 hacia los demás. Para esto deberá leer en primer lugar la cantidad de dimensiones que posee el punto, leer el punto de origen P_0 , leer los puntos adicionales para finalmente calcular y mostrar la menor y la mayor distancia.

- Considere para este problema que los puntos solamente se podrán representar en 2 o 3 dimensiones.
- Considere que para la lectura de los puntos adicionales primero debe ingresar la cantidad de puntos adicionales a ingresar.
- Para el cálculo de la distancia utilice la distancia de Manhattan.

A continuación se presentan algunos ejemplos de ejecución:

Ingrese dimensiones (2 o 3): 2

Ingrese número punto origen (x , y): -1 5

Ingrese la cantidad de puntos adicionales a ingresar: 3

Ingrese punto adicional 1 (x1, y1): 1 6

Ingrese punto adicional 2 (x2, y2): 3 5

Ingrese punto adicional 3 (x3, y3): 2 3

La menor distancia al punto (-1.00, 5.00) es de 3.00 que corresponde con el punto (1.00, 6.00)

La mayor distancia al punto (-1.00, 5.00) es de 5.00 que corresponde con el punto (2.00, 3.00)

Ingrese dimensiones (2 o 3): 3

Ingrese número punto origen (x, y, z): 0.4 0.9 2.7

Ingrese la cantidad de puntos adicionales a ingresar: 4

Ingrese punto adicional 1 (x1, y1, z1): 2.50 6.70 1.70

Ingrese punto adicional 2 (x2, y2, z2): 6.70 8.50 8.50

Ingrese punto adicional 3 (x3, y3, z3): 2.50 2.50 4.70

Ingrese punto adicional 4 (x4, y4, z4): 3.30 4.40 5.50

La menor distancia al punto (0.40, 0.90, 2.70) es de 5.70 que corresponde con el punto (2.50, 2.50, 4.70)

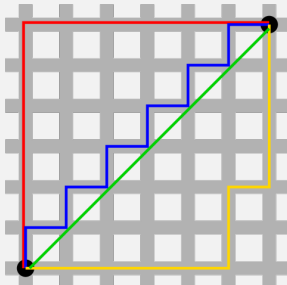
La mayor distancia al punto (0.40, 0.90, 2.70) es de 19.70 que corresponde con el punto (6.70, 8.50, 8.50)

Cálculo de la distancia de Manhattan:

La distancia de Manhattan entre dos puntos $x = (x_1, x_2, \dots, x_n)$ e $y = (y_1, y_2, \dots, y_n)$ en un espacio n dimensional es la suma de las distancias en cada dimensión.

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

En la siguiente imagen^a podrá notar la diferencia entre la distancia Euclidiana y la distancia de Manhattan. Las líneas rojas, azul y amarillas tienen una distancia de 12 tanto en la geometría Euclidiana como en la de Manhattan. En la geometría Euclidiana la línea verde tiene una longitud de $6 \times \sqrt{2} \approx 8.4852$ que corresponde con el camino más corto. Usando la distancia de Manhattan, la longitud de la línea verde es de 12, por lo que no es la distancia más corta. La distancia de Manhattan es muy utilizada para calcular la distancia que recorren los taxistas en la ciudades.



^aImagen de dominio público disponible en la URL https://commons.wikimedia.org/wiki/File:Manhattan_distance.svg

2.2.8. El Algoritmo de la Lazada

Una forma de calcular el área de un polígono simple cuyos vértices están representados como coordenadas en un plano es mediante el algoritmo de la Lazada. Este algoritmo es bastante versátil pues permite calcular el área del polígono sin importar la cantidad de vértices que tenga éste. Este algoritmo calcula el área de la siguiente manera:

Algoritmo de la Lazada

$$area = \frac{1}{2} \left| \sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n \right|$$

donde:

(x_i, y_i) corresponden a las coordenadas del vértices i

Se le pide que solicite al usuario la información relacionada a un polígono y en base a esto dados calcule el área. Se deberá solicitar la cantidad de vértices que posee y los valores de los puntos (abscisa y ordenada) de cada vértice. Deberá validar que la cantidad de vértices sea mayor que 2. En caso sea menor que 2, deberá emitir un mensaje de advertencia al usuario.

A continuación se presentan unos casos de prueba de ejecución:

Ingrese cantidad de vértices: 2

Debe ingresar un número de vértices mayor a 2.

Ingrese cantidad de vértices: -1

Debe ingresar un número de vértices mayor a 2.

Ingrese cantidad de vértices: 5

Ingrese punto 1 (x,y): 3 4

Ingrese punto 2 (x,y): 5 11

Ingrese punto 3 (x,y): 12 8

Ingrese punto 4 (x,y): 9 5

Ingrese punto 5 (x,y): 5 6

El área del polígono es: 30.000000

Ingrese cantidad de vértices: 4

Ingrese punto 1 (x,y): -2 -4

Ingrese punto 2 (x,y): 6 -2

Ingrese punto 3 (x,y): 7 4

Ingrese punto 4 (x,y): -8 2

El área del polígono es: 74.000000

Ingrese cantidad de vértices: 3

Ingrese punto 1 (x,y): 2 4

Ingrese punto 2 (x,y): 3 -8

Ingrese punto 3 (x,y): 1 2

El área del polígono es: 7.000000

¿Cómo funciona el algoritmo de la Lazada?

Imagine que se tiene el polígono cuyos vértices son $P_1(2, 4)$, $P_2(3, -8)$ y $P_3(1, 2)$. Entonces:

- $(x_1, y_1) = (2, 4)$
- $(x_2, y_2) = (3, -8)$
- $(x_3, y_3) = (1, 2)$

Aplicando la fórmula:

$$\begin{aligned} \text{area} &= \frac{1}{2} \left| \sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n \right| \\ \text{area} &= \frac{1}{2} |x_1 y_2 + x_2 y_3 + x_3 y_1 - x_2 y_1 - x_3 y_2 - x_1 y_3| \\ \text{area} &= \frac{1}{2} |(2 \times -8) + (3 \times 2) + (1 \times 4) - (3 \times 4) - (1 \times -8) - (2 \times 2)| \\ \text{area} &= \frac{1}{2} |-16 + 6 + 4 - 12 - (-8) - 4| = \frac{1}{2} |-6 - 8| = \frac{1}{2} |-14| = 7 \end{aligned}$$

Variación al problema

En el problema planteado, se solicita primero al usuario que ingrese la cantidad de vértices que tendrá el polígono. Altere su solución para que no se solicite este dato. El flujo de lectura finalizará cuando el usuario ingrese un vértice que es igual que el anterior.

A continuación se presenta un caso de prueba de ejecución:

```
Ingrese punto 1 (x,y): 2 4
Ingrese punto 2 (x,y): 3 -8
Ingrese punto 3 (x,y): 1 2
Ingrese punto 4 (x,y): 1 2
El último punto no se considerará para el cálculo del área del polígono.

El área del polígono es: 7.000000
```

2.2.9. Algoritmo KNN

El Aprendizaje de Máquina (*Machine Learning* en inglés, también conocido como Aprendizaje Automático) es un campo de las Ciencias de la Computación que se basa en la idea de que los sistemas pueden aprender a partir de los datos, identificando patrones y tomando decisiones con una mínima intervención humana. Según el Aprendizaje de Máquina los paradigmas de aprendizaje se pueden clasificar en dos: el aprendizaje supervisado y el aprendizaje no supervisado. Dentro del aprendizaje supervisado se pueden realizar diversas tareas, dentro de ellas la clasificación.

La clasificación, dentro del Aprendizaje de Máquina, se puede definir como una tarea que dado un conjunto de clases conocidas *a priori*, intenta predecir a qué conjunto de clases pertenece determinado individuo. Para llevar a cabo una tarea de clasificación se cuentan con diversos algoritmos, dentro de ellos se tiene el KNN.

El algoritmo KNN proviene de las siglas en inglés *K-Nearest Neighbours*, en español se traduce como k vecinos más cercanos. Para facilitar el entendimiento del algoritmo KNN, se simplificarán las condiciones del problema. En primer lugar se asumirá que solo existen 2 clases para predecir, la clase *A* y la clase *B*. En segundo lugar se asumirá que los datos de cada clase son puntos en el plano cartesiano. En tercer lugar se asumirá que el valor de *K* para este problema será el valor de 3. Una configuración de estas condiciones se puede apreciar en la figura 2.1.

Para determinar si un individuo (en la figura 2.1 se puede apreciar al individuo etiquetado con un ?) pertenece

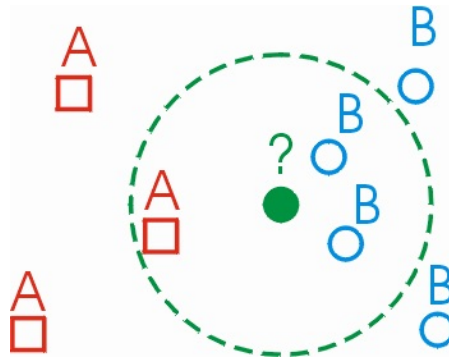


Figura 2.1: Algoritmo KNN. Imagen tomada de la URL <http://www.fsanchezcv.com/2012/06/>.

a determinada clase, se calcula los k individuos más cercanos, en esta caso los 3 puntos más cercanos. Para este cálculo se puede utilizar la distancia euclidiana entre dos puntos. Una vez que se tiene los k puntos más cercanos, se cuentan cuántos de ellos son de la clase A y cuántos de ellos son de la clase B . La clase que tenga más puntos cercanos será la predicción que realizará el algoritmo. En el caso del ejemplo de la figura 2.1, el algoritmo predecirá que el individuo pertenece a la clase B pues se tienen dos puntos, la clase A solo un punto.

Se pide que implemente el algoritmo KNN, para esto deberá:

- Asumir que solo se cuentan con dos clase, la clase A y la clase B .
- Asumir que los datos se representarán a través de puntos en el plano cartesiano.
- Asumir que el valor de K será igual a 3.
- La distancia entre los puntos se calculará a través de la distancia euclidiana.

Deberá realizar como mínimo lo siguiente:

- Solicitar al usuario el punto que se desea clasificar.
- Solicitar al usuario un conjunto de puntos con sus respectivas clases. El flujo de lectura terminará cuando se ingrese el punto (0,0).
- Identificar los 3 puntos más cercanos con sus respectivas clases.
- Determinar la clase a la que pertenece el punto que se quiere clasificar.
- Deberá verificar que no pueda terminar si es que no se ingresan por lo menos 3 puntos.
- Deberá verificar que solo permita el ingreso de las clases A y B .

A continuación se presenta algunos ejemplos de ejecución:

```
Ingrese el punto a clasificar (x,y): 4 4
Ingrese los datos del punto (x,y, clase): 1 1 A
Ingrese los datos del punto (x,y, clase): 2 2 B
Ingrese los datos del punto (x,y, clase): 3 3 A
Ingrese los datos del punto (x,y, clase): 0 0 X
```

El punto ingresado corresponde a la clase A

```
Ingrese el punto a clasificar (x,y): 4 4
Ingrese los datos del punto (x,y, clase): 0 0 X
```

Debe ingresar por lo menos tres puntos antes de finalizar el algoritmo.

Ingrese los datos del punto (x,y, clase): 1 1 A

Ingrese los datos del punto (x,y, clase): 2 2 C

La clase es inválida, no se considerará este punto.

Ingrese los datos del punto (x,y, clase): 2 2 B

Ingrese los datos del punto (x,y, clase): 3 3 B

Ingrese los datos del punto (x,y, clase): 0 0 0

El punto ingresado corresponde a la clase B

Ingrese el punto a clasificar (x,y): 4 4

Ingrese los datos del punto (x,y, clase): 2 3 A

Ingrese los datos del punto (x,y, clase): 2 4 A

Ingrese los datos del punto (x,y, clase): 5 3 B

Ingrese los datos del punto (x,y, clase): 5 2 B

Ingrese los datos del punto (x,y, clase): 3 4 A

Ingrese los datos del punto (x,y, clase): 1 1 A

Ingrese los datos del punto (x,y, clase): 4 3 B

Ingrese los datos del punto (x,y, clase): 2 4 B

Ingrese los datos del punto (x,y, clase): 0 0 Z

El punto ingresado corresponde a la clase B

Recuerde que:

La distancia euclidiana entre dos puntos se calcula como sigue:

$$d(P_1(x_1, y_1), P_2(x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Sugerencia para el lenguaje C

En caso lo requiera, en el archivo de cabecera `float.h` se encuentra definida la constante simbólica `DBL_MAX` que define el máximo valor que se puede almacenar en un `double`.

2.2.10. Conjetura de Collatz

En 1937 el matemático Lothar Collatz enunció la siguiente conjetura. Dado un número natural n , aplicar la siguiente función:

$$f(n) = \begin{cases} \frac{n}{2} & \text{si } n \text{ es par} \\ 3n + 1 & \text{si } n \text{ es impar} \end{cases}$$

con el número obtenido, repetir el proceso (es decir se aplica nuevamente la función) de manera sucesiva. En algún instante del tiempo se llega al número 1.

Aunque la conjetura no se ha demostrado de forma matemática, computacionalmente se ha comprobado que hasta el número 2^{58} la secuencia acaba siempre en 1.

Se le pide a usted que dado un número entero mayor que 1 y menor que 2^{58} imprima la secuencia que se genera, indique la cantidad de términos que componen la serie así como el mayor número.

Casos de prueba para verificación de solución

Use los siguientes casos para verificar si su solución está correcta.

- Si el número es 4 se obtiene la siguiente secuencia: 4, 2, 1.
- Si el número es 5 se obtiene la siguiente secuencia: 5, 16, 8, 4, 2, 1.
- Si el número es 13 se obtiene la siguiente secuencia: 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

2.2.11. Cálculo de integrales: la suma de Riemann

En matemáticas, la suma de Riemann permite determinar el valor de una integral definida, es decir, el área bajo una curva. Consiste en trazar un número finito de rectángulos dentro de un área irregular, calcular el área de cada uno de ellos y sumarlos. El área bajo la curva será la suma de todos los rectángulos. Mientras más rectángulos se tengan, más precisa será el valor del área calculada. En la figura 2.2 se presenta la gráfica de la función $1 + x^2$ en el rango $[0, 5]$. Siguiendo lo enunciado por Riemann, el área bajo la curva sería la suma de los rectángulos verdes, los que se encuentran bajo la curva.

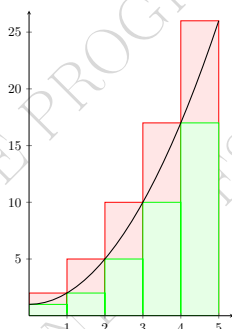


Figura 2.2: Suma de Riemann

Dada una función $f(x)$ continua en el rango $[a, b]$ se le pide que calcule el valor de la suma usando la siguiente sumatoria $\sum_{i=1}^n f(y_i)(x_i - x_{i-1})$. $(x_i - x_{i-1})$ representa la base de un rectángulo y $f(y_i)$ representa la altura del mismo. Se debe cumplir que: $x_{i-1} \leq y_i \leq x_i$. La elección del valor de y_i es arbitraria pero por lo general toman dos valores: o x_{i-1} (suma de Riemann por la izquierda) o x_i (suma de Riemann por la derecha). Los valores de x se toman de tal forma que $a = x_0 < x_1 < x_2 < \dots < x_n = b$.

Sugerencia

- Para calcular los valores de x se sugiere iterar, iniciando en a
 1. Defina una variable *incremento* con un valor pequeño, recuerde que mientras más pequeño el rectángulo, más precisión se tendrá.
 2. Cree dos variables, una para almacenar el valor de x_i y otra para x_{i+1} . Inicialmente a x_i asígnele el valor de a y a x_{i+1} asígnele el valor de $x_i + \text{incremento}$.
 3. Para la siguiente iteración, actualice la variable x_i con el valor actual de x_{i+1} y a la variable x_{i+1} súmele el valor de *incremento*.
 4. Finalice la iteración cuando x_i llegue a ser mayor o igual que b .

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

1. $f(x) = (x - 1)^2 + 2$, rango $[-1, 2]$, debe retornar un número cercano a 9.
2. $f(x) = x^2 - 2x$, rango $[1, 3]$, debe retornar un número cercano a $\frac{2}{3}$.

2.2.12. Rebotes sucesivos

Se tiene una pelota que se encuentra a una determinada altura h . La pelota cae y rebota hasta $\frac{1}{x}$ de la altura h . Se desea calcular el espacio que recorre la pelota hasta que esta queda en reposo. Indique además cuántas veces rebotó la pelota.

Sugerencia

- A pesar que este problema se puede solucionar fácilmente aplicando la suma de los infinitos términos de una sucesión geométrica ($S_n = \frac{a_1}{1-r}$) se pide resolver el problema aplicando fuerza bruta computacional, es decir usando una estructura iterativa para ir acumulando la suma de las alturas alcanzadas en cada rebote.
- En teoría la cantidad de rebotes es infinita, asuma que puede terminar de contar cuando la altura alcanzada por la pelota es menor a 10^{-3} .

Casos de prueba para verificación de solución

Use los siguiente casos para verificar si su solución está correcta.

- Si $h = 18$ metros y $x = 2$, el espacio que recorre la pelota es 36 metros.
- Si $h = 18$ metros y $x = 3$, el espacio que recorre la pelota es 27 metros.
- Si $h = 18$ metros y $x = 4$, el espacio que recorre la pelota es 24 metros.

2.2.13. Cálculo numérico: El método de bisección

El objetivo del método de bisección es encontrar la raíz de una ecuación. Recuerde que x será una raíz de una función f si es que x pertenece al dominio de la función y además $f(x) = 0$.

Se parte de una función $f(x)$ continua en un intervalo $[a, b]$ y además se debe cumplir que $f(a) * f(b) < 0$. Entonces por el teorema del valor intermedio se garantiza que existe un $r \in]a, b[$ tal que $f(r) = 0$. Es decir, se garantiza que existe una raíz.

Según este método, se deben seguir los siguiente pasos para calcular la raíz:

- Se divide el intervalo por la mitad m . Se calcula el valor de m de la siguiente manera: $m = \frac{a+b}{2}$
- Si $f(m) = 0$, entonces m es la raíz buscada y termina el cálculo.
- Caso contrario
 - si $f(a)$ y $f(m)$ tienen signos diferentes, entonces la raíz se encontrará en el intervalo $]a, m[$
 - si $f(m)$ y $f(b)$ tienen signos diferentes, entonces la raíz se encontrará en el intervalo $]m, b[$
 - con el nuevo intervalo se repiten los pasos descritos anteriormente hasta encontrar la raíz.

- El algoritmo termina cuando se ha encontrado una raíz ($f(m) = 0$) o cuando $|b_n - a_n| < \varepsilon$. Siendo ε la precisión aceptada.

Se solicita que calcule la raíz de una determinada función, en un determinado rango y con una determinada precisión usando el método de bisección. Para cada iteración imprima: el número de iteración, el valor de a , el valor de b , el valor de m , el valor de $f(a)$, el valor de $f(m)$, el valor de $f(b)$ y el valor de $|b - a|$.

Casos de prueba

Use los siguientes datos para probar su solución:

- $f(x) = x^4 + 3x^3 - 2$, $a = 0$, $b = 1$, *precisión* = 0.01
- $f(x) = x^3 + 4x^2 - 10$, $a = 1$, $b = 0.5$, *precisión* = 0.001
- $f(x) = x^5 - x + 3$, $a = -2$, $b = -1$, *precisión* = 0.0001

2.2.14. Regresión lineal simple: suma de las distancias al cuadrado

El objetivo de un modelo de regresión es tratar de explicar la relación que existe entre una variable dependiente (variable respuesta) Y , y un conjunto de variables independientes (variables explicativas) x_1, x_2, \dots, x_n . Es una técnica muy utilizada para hacer predicciones, es una de las técnicas clásicas usadas en la minería de datos para hacer predicción numérica.

En la regresión lineal simple el modelo se representa a través de una recta de la forma $Y = bX + a$. En la figura 2.3 se presenta un ejemplo de un modelo de regresión lineal simple que trata de explicar la dependencia entre el peso y la altura. Este modelo se expresa a través de la recta $\text{altura} = 0.6 \times \text{peso} + 130.2$. Al ser una predicción, algunos datos quedarán fuera de la recta (puede apreciar los puntos en la figura 2.3). Se busca que la distancia entre los datos reales (observaciones) y la recta (estimación/predicción) sea la mínima posible.

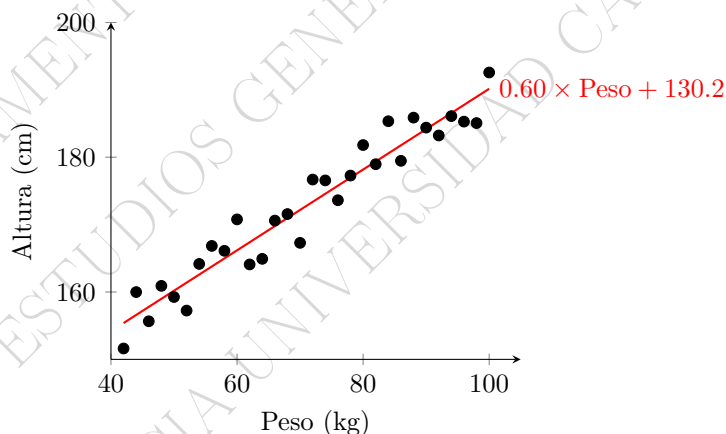


Figura 2.3: Regresión lineal

Se pide que dada una recta que representa una regresión lineal simple de la forma $Y = bX + a$ y un conjunto de pares de valores (x, y) que representan observaciones reales, calcule la suma de las distancias al cuadrado del valor observado y el valor estimado. Es decir $\sum_{i=1}^n (y_i - \hat{y}_i)^2$. Donde y_i representa al valor observado y \hat{y}_i representa al valor estimado que se calcula usando el valor de x observado y la recta que representa el modelo de regresión lineal ($Y = bX + a$).

Sugerencia

Siga los siguientes pasos:

- Solicite que el usuario ingrese el valor de la cantidad de datos observados, en una variable n .
- Luego itere n veces. En cada iteración solicite la lectura de los datos observados (x, y) . Utilice una variable para x_i y otra para y_i .
- Haga el cálculo del valor estimado \hat{y}_i en otra variable.
- Con los datos obtenidos, haga el cálculo solicitado $(y_i - \hat{y}_i)^2$ y acumúlelos en una variable llamada *suma*.

Casos de prueba

Use los siguientes datos para probar su solución:

- Como modelo de regresión lineal utilice la siguiente recta: $\text{producción} = 3.47 \times \text{horas} + 32.01$.
- Como observaciones utilice los siguientes datos:

horas	producción
80	300
79	300
83	315
84	330

2.2.15. Trayectoria de un proyectil

La trayectoria de un proyectil se puede calcular a través del estudio del movimiento parabólico. Un proyectil que se mueve en un medio que no ofrece resistencia al avance y que está sujeto a un campo gravitatorio uniforme describe una parábola como se puede apreciar en la figura 2.4.

Típicamente al momento de lanzar un proyectil se conocen la velocidad inicial de lanzamiento (v_o), el ángulo de lanzamiento (θ) y la gravedad ($g \approx 9.8 \text{ m/s}^2$). Con estos datos y analizando el movimiento en el eje horizontal y el eje vertical se puede determinar, la altura máxima de elevación (H), el rango o alcance horizontal (A), el tiempo total de vuelo y el tiempo medio para llegar al punto más alto. El alcance horizontal se puede determinar a través de la siguiente fórmula: $A = \frac{v_o^2 \times \sin(2\theta)}{g}$.

Perú se encuentra jugando con Dinamarca en el estadio de Ekaterimburgo en Rusia. El marcador está 0 – 0. A los 45 minutos del segundo tiempo, Christian Cueva, centrocampista de la selección peruana, percibe que Paolo Guerrero, delantero peruano, está en una posición inmejorable para anotar. Él se encuentra a 25 metros de Paolo Guerrero y sabe que con una velocidad de 20 m/s las chances de acertar el pase son bien altas. Sabiendo que es la última jugada del partido debe calcular rápidamente el ángulo con el que debe patear el balón. Se pide que escriba un programa que le permita recomendar cuál debería ser el ángulo con el que Christian Cueva deberá patear el balón.

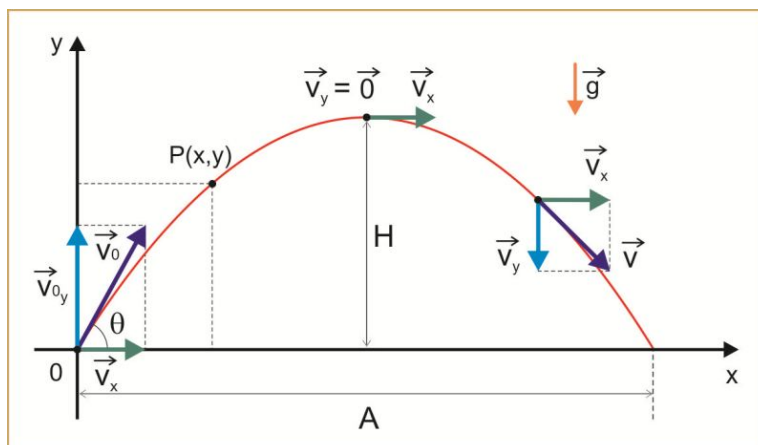


Figura 2.4: Trayectoria de un proyectil. Imagen tomada de la URL http://osfundamentosdafisica.blogspot.pe/2011/01/resolucao-de-preparando-se-para-as_22.html.

Sugerencia

- Dada la naturaleza del problema, los posibles valores de ángulo oscilan entre $[0..90]$. Se sugiere que itere en este rango y calcule el alcance del “disparo”. Inicie con el valor de 0 y haga pequeños incrementos hasta llegar a 90.
- En el lenguaje C utilice la función `sin` cuyo prototipo se encuentran en el archivo de cabecera `math.h`. La función `sin`, cuyo prototipo es `double sin(double x)`, retorna el seno del parámetro x . El parámetro debe indicar un ángulo en radianes.
- En PSeInt utilice la función `sen`. La función `sen`, retorna el seno del parámetro x . El parámetro debe indicar un ángulo en radianes.
- Es probable que el alcance no llegue a ser igual a lo requerido (exactamente 25 metros). Para determinar el “mejor” ángulo calcule la menor diferencia entre la distancia de Paolo y el alcance del “disparo”. Use en este caso el valor absoluto.
- Sería interesante además calcular el tiempo que se demora en llegar la pelota a Paolo. Para esto puede usar la fórmula $t_v = 2 \times \frac{v_o \times \sin(\theta)}{g}$. Si con dos ángulos se obtiene la menor distancia, el tiempo podría ser una característica que permite tomar la mejor decisión.

2.3. Nivel avanzado

2.3.1. La serie Gregory-Leibniz

Una forma simple para obtener el resultado de la aproximación de Pi es utilizando la siguiente suma de series (Gregory-Leibniz): (ver Figura 2.16):

$$Pi = 4 \times (1 - 1/3 + 1/5 - 1/7 + 1/9 - \dots \infty)$$

Realizar un programa en el lenguaje C, que permita evaluar los resultados de la serie anterior y compararlos con un valor de Pi conocido que tiene 10 dígitos decimales. El programa le permitirá comparar los valores obtenidos conforme va aumentando términos a la serie. (ver Figura 2.17).

Es importante indicar que el programa deberá validar que la cantidad de términos a evaluar sea mayor e igual a 1. En caso no sea así, debe mostrar un mensaje de “Error”.

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = 1 - \frac{1}{3} + \frac{1}{5} - \dots = \frac{\pi}{4}$$

Figura 2.5: Sumatoria de la serie

Valores que va tomando Pi	
Error	con menos de 1 término
$Pi = 4 \times (1)$	con 1 término
$Pi = 4 \times (1 - \frac{1}{3})$	con 2 términos
$Pi = 4 \times (1 - \frac{1}{3} + \frac{1}{5})$	con 3 términos
Y así según corresponda.	

Figura 2.6: Valores que va tomando Pi

Donde: fabs(x) = Valor Absoluto de x. Valor de referencia de PI=3.1415926536

El programa debe indicar el valor calculado de Pi y la diferencia absoluta respecto al valor de referencia de PI.

Casos de prueba

Use los siguientes casos para verificar si su solución está correcta:

Cant. términos	Aprox. de Pi	Diferencia
1	4.0000000000	0.8584073464
2	2.6666666667	0.4749259869
3	3.4666666667	0.3250740131
5	3.3396825397	0.1980898861
10	3.0418396189	0.0997530347

2.3.2. La serie de Nilakantha

Una forma simple para obtener el resultado de la aproximación de Pi es utilizando la siguiente suma de series (Nilakantha): (ver Figura 2.16):

$$Pi = 3 + 4/(2 \times 3 \times 4) - 4/(4 \times 5 \times 6) + 4/(6 \times 7 \times 8) - 4/(8 \times 9 \times 10) - \dots \infty$$

$$\pi = 3 + \sum_{n=1}^{\infty} \frac{4(-1)^{n+1}}{(2n) \times (2n+1) \times (2n+2)}$$

Figura 2.7: Sumatoria de la serie

Realizar un programa en el lenguaje C, que permita evaluar los resultados de la serie anterior y compararlos con un valor de Pi conocido que tiene 10 dígitos decimales. El programa le permitirá comparar los valores obtenidos conforme va aumentando términos a la serie.

Es importante indicar que el programa deberá validar que la cantidad de términos a evaluar sea mayor e igual a 1. En caso no sea así, debe mostrar un mensaje de "Error".

Donde: fabs(x) = Valor Absoluto de x.

El programa debe indicar el valor calculado de Pi y la diferencia absoluta respecto al valor de referencia de PI.

Casos de prueba

Use los siguientes casos para verificar si su solución está correcta:

Cant. términos	Aprox. de Pi	Diferencia
1	3.1666666666	0.0250740130
2	3.1333333333	0.0082593203
3	3.1452380952	0.0036454416
5	3.1427128427	0.0011201891
10	3.1414067185	0.0001859351

2.3.3. Rotar un número

Llamemos rotación de un número al proceso por el cual tomamos un número y corremos una posición todos sus dígitos (para el caso del dígito que quede fuera, esta toma el lugar de la posición que quedó vacía al hacer el cambio). Por ejemplo, el número 5678 puede rotar hacia la derecha, moviendo sus primeros 3 dígitos una posición hacia la derecha, y el cuarto dígito (el número 8) a la posición inicial, es decir, quedaría 8567. Asimismo, este número podría rotar hacia la izquierda si corremos sus últimos 3 dígitos una posición hacia la izquierda, y movemos el primer dígito a la posición de la unidad, de la siguiente manera: 6785.

Realizar un programa en el lenguaje C, que reciba como datos un número a rotar de cuatro dígitos, el número de veces que se quiere hacer la rotación, y un número que permita al usuario indicar hacia qué dirección desea hacer la rotación: rotando hacia la derecha si el usuario ingresa el número 1 y hacia la izquierda si el usuario ingresa el número 2.

El programa debe validar que el **número** a rotar sea de 4 dígitos, que el número de **veces** sea mayor a cero y que el valor del sentido solo sea 1 o 2. En los demás casos debe mostrar un mensaje de error.

Casos de prueba

Use los siguientes casos para verificar si su solución está correcta:

Número	Veces	Sentido de rotación	Resultado	Mensaje
7894	2	1	9478	
3468	1	2	4683	
897	1	1		Error de alguno de los datos
3567	2	3		Error de alguno de los datos
4599	-1	2		Error de alguno de los datos
5629	3	1	6295	

2.3.4. Cocientes de la serie Fibonacci

Si examinamos el cociente de los términos formados a partir de la sucesión de Fibonacci dividiendo un término por el término que lo precede (ver Figura Secuencias de divisores), descubrimos otra notable propiedad de los números de Fibonacci. Hagámoslo con unos cuantos términos: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.

1/1	2/1	3/2	5/3	8/5	13/8	21/13	34/21	55/34
1,000	2,000	1,500	1,333	1,600	1,625	1,615	1,619	1,617

Figura 2.8: Secuencias de divisiones

Muy pronto los cocientes se aproximan a un valor conocido como la proporción áurea designado por la letra griega Φ y tiene el valor exacto:

$$\phi = \frac{1+\sqrt{5}}{2}$$

Figura 2.9: Fórmula de Φ

y esto se puede aproximar al decimal 1,618033988.

Realizar un programa en el lenguaje C, que permita imprimir los cocientes de los valores de la serie Fibonacci pero controlado por la cantidad de iteraciones y la diferencia del cociente y la proporción **áurea**. El dato que debe ingresar el usuario para resolver el problema es el número **n** de iteraciones y la aproximación **dif** ambos permiten controlar la iteración. La iteración se detiene si se cumplen todas las iteraciones o la aproximación es menor que el valor de **dif**.

El programa debe validar que el valor de **n** sea mayor a cero para realizar el cálculo. Si **n** es menor o igual a cero se muestra el mensaje “n debe ser mayor que 0”. También debe validar que el valor de **dif** debe ser mayor que 0.

Casos de prueba

Use los siguientes casos para verificar si su solución está correcta:

n	dif	Mensaje
10	0.01	Nro. fracción distancia
10	0.01	:1 :0.000000 : 1.618034
10	0.01	:1 :1.000000 : 0.618034
10	0.01	:2 :1.000000 : 0.618034
10	0.01	:3 :2.000000 : 0.381966
10	0.01	:5 :1.500000 : 0.118034
10	0.01	:8 :1.666667 : 0.048633
10	0.01	:13 :1.600000 : 0.018034
10	0.01	:21 :1.625000 : 0.006966
10	0.05	Nro. fracción distancia
10	0.05	:1 :0.000000 : 1.618034
10	0.05	:1 :1.000000 : 0.618034
10	0.05	:2 :1.000000 : 0.618034
10	0.05	:3 :2.000000 : 0.381966
10	0.05	:5 :1.500000 : 0.118034
10	0.05	:8 :1.666667 : 0.048633
10	0.05	:13 :1.600000 : 0.018034
10	0.05	:21 :1.625000 : 0.006966
10	0.05	:34 :1.615385 : 0.002649
10	0.05	:55 :1.619048 : 0.001014
-5		El número debe ser mayor que cero

2.3.5. El número Anaprimo

Un número **Omipr** es un número primo que al tomar sus cifras en orden contrario también son primos pero diferentes a él, no es capicúa. De ahí su nombre: primo escrito al revés.

Ahora se dice que un número es **Anaprimo**, si al combinar sus cifras en órdenes distintos obtenemos por lo menos otro número primo.

Se pide realizar esta evaluación para números de 3 dígitos. Por ejemplo:

- 113 es primo y el inverso es 311 que primo también, entonces 113 es omipr.

- Luego 131 también es primo. Por lo tanto, el número 113 es anaprímo

Realizar un programa en el lenguaje C, que permita identificar si un número es **omirp** o **anaprímo**. El dato que debe ingresar el usuario para resolver el problema es el número **n** por evaluar.

El programa debe validar que el valor de **n** sea mayor a cero para realizar el cálculo. Si **n** es menor o igual a cero se muestra el mensaje “n debe ser mayor que 0”. También debe validar que el número debe tener la cantidad de dígitos establecido.

Casos de prueba

Use los siguientes casos para verificar si su solución está correcta:

n	n	n inverso	Mensaje
113	Es primo	Es omirp	Es anaprímo
117	Es primo	Es omirp	No es anaprímo
397	Es primo	Es omirp	Es anaprímo
99			El número 99 no es de tres dígitos
-5			El número debe ser mayor que cero

2.3.6. Calcular el número de triangulaciones para un polígono

Tenemos que C_n es el número de formas distintas de dividir un polígono convexo de $n + 2$ lados en triángulos conectando vértices con diagonales sin que ninguna se corte. La siguiente figura ilustra el caso de las $C_4 = 14$ posibles triangulaciones para un polígono de 6 lados:

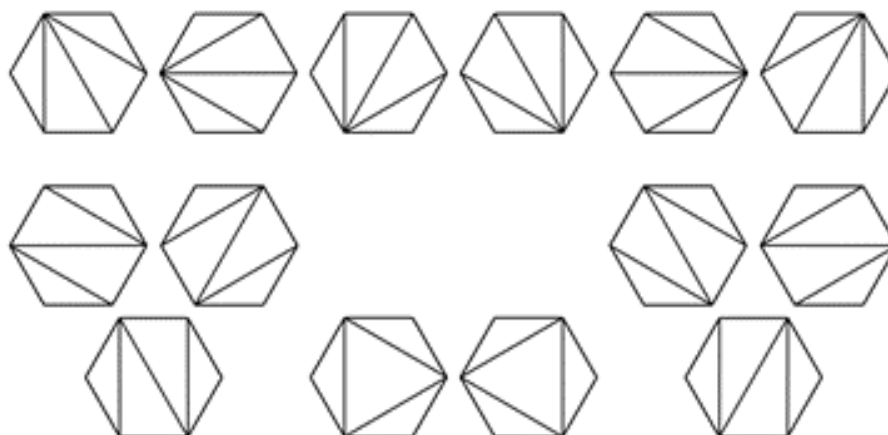


Figura 2.10: Triangulaciones de un polígono de 6 lados

Los números de Catalan forman la sucesión cuyo término general es.

$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)! \cdot n!} \wedge n \geq 0$$

Figura 2.11: Fórmula para obtener los números

Los primeros números de catalán son 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786. Los números de Catalan satisfacen la siguiente relación de recurrencia:

Realizar un programa en el lenguaje C, que permita identificar el número de triangulaciones para un polígono de n lados. El dato que debe ingresar el usuario para resolver el problema es el valor de los n lados del polígono.

$$C_n = \begin{cases} \text{si } n = 0 & \Rightarrow 1 \\ \text{si } n > 0 & \Rightarrow \frac{2(2n-1)}{n+1} C_{n-1} \end{cases}$$

Figura 2.12: Fórmula de recurrencia

El programa debe validar que el valor de n sea mayor a cero para realizar el cálculo. Si n es menor o igual a cero se muestra el mensaje “ n debe ser mayor que 0”.

Casos de prueba

Use los siguientes casos para verificar si su solución está correcta:

n	Mensaje
6	La cantidad de posibles triangulaciones es 14
8	La cantidad de posibles triangulaciones es 32
2	Verificar la cantidad de lados del polígono
4	La cantidad de posibles triangulaciones es 2
-5	El número debe ser mayor que cero

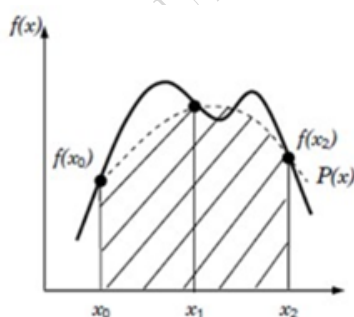
2.3.7. La Regla de Simpson

La regla de Simpson es un método para calcular área plana bajo una curva de una función continua y acotada utilizando trapecios curvilíneos a partir una interpolación con una función cuadrática como se muestra a continuación:

$$I \cong (b-a) \frac{f(x_0) + 4f(x_1) + f(x_2)}{6}$$

Figura 2.13: Fórmula de Simpson

Esta ecuación es conocida como la Regla de Simpson de 1/3 simple. La especificación 1/3 surge del hecho de que h está dividida en 3 (ver Figura 2.17).



$$h = (b - a)/2.$$

$$\int_a^b f(x) dx \cong \frac{h}{3} (f(x_0) + 4f(x_1) + f(x_2))$$

Figura 2.14: Gráfica de la curva para Simpson

Se puede ampliar si subdividimos el intervalo $[a, b]$ en “ n ” subintervalos, todos de la misma longitud.

En la regla de Simpson se toman tres puntos como referencia. Estos puntos se representan como la función inicial para el primer punto, la función del punto final, para el tercer punto y la función del promedio del punto final e inicial para el segundo punto.

$$h = \frac{b - a}{n}$$

Figura 2.15: Fórmula de subintervalos

Realizar un programa en el lenguaje C que permita calcular el área debajo de curva descrita por la función $f(x) = x^3 + x^2 \cdot 5x + 3$ entre a y b aplicando el método explicado y muestre el resultado redondeado a dos decimales y un mensaje dependiendo de la respuesta.

El dato que debe ingresar el usuario para resolver el problema es el intervalo (límite inferior “ a ” y límite superior “ b ”) y la cantidad de subintervalos “ n ” con el siguiente formato “[a , b]; n ”.

El programa debe validar que el valor de b sea mayor que el de a y que n sea mayor a cero para realizar el cálculo. Si b es menor que a se muestra el mensaje “ b debe ser mayor o igual que a ”, si n es menor o igual a cero se muestra el mensaje “ n debe ser mayor que 0”. Si el área es mayor o igual a cero, el mensaje será “ El área es mayor o igual a 0”, sino el mensaje será “ El área es menor a 0”.

Casos de prueba

Use los siguientes casos para verificar si su solución está :

Inicio	Fin	n	Área	Mensaje
1.1	10.1	10000	2719.1339999992	El área es mayor o igual a cero
2.2	1	100		b debe ser mayor o igual que a
1.1	10.1	2	2719.1340000000	El área es mayor o igual a cero

2.3.8. Calcular área

La integral definida entre los puntos a y b de una función continua y acotada $f(x)$ representa el área comprendida debajo de esa función. En ocasiones es necesario calcular integrales (áreas) de modo numérico, es decir, sin conocer la integral de la función $f(x)$. Un método consiste en sustituir el área por un conjunto de n trapecios elementales, cada uno de altura h y bases $f(a + i \cdot h)$ y $f(a + (i + 1) \cdot h)$ (ver Figura 1), o lo que es equivalente, por n rectángulos de base h y altura $(f(a + i \cdot h) + f(a + (i + 1) \cdot h)) / 2$. Es obvio que con este método los errores van a ser más pequeños. En este caso, si llamamos f_i a $f(a + i \cdot h)$, la formula resulta ser:

$$\int_a^b f(x) dx = \sum_{i=0}^{n-1} (f_i + f_{i+1})h / 2 = \frac{f_0 + f_n}{2} h + \sum_{i=1}^{n-1} f_i h$$

Figura 2.16: Fórmula de trapecios

La representación gráfica de esta forma de aproximar la integral se presenta en la Fig. 2. Resulta que si n se hace muy grande (h muy pequeño) el error será pequeño.

Realizar un programa en el lenguaje C, que permita calcular el área debajo de curva descrita por la función $f(x) = x^3 + x^2 \cdot 5x + 3$ entre a y b aplicando el método explicado y muestre el resultado redondeado a dos decimales y un mensaje dependiendo de la respuesta.

El dato que debe ingresar el usuario para resolver el problema es el intervalo (límite inferior “ a ” y límite superior “ b ”) y la cantidad de subintervalos “ n ” con el siguiente formato “[a , b]; n ”.

El programa debe validar que el valor de b sea mayor que el de a y que n sea mayor a cero para realizar el cálculo. Si b es menor que a se muestra el mensaje “ b debe ser mayor o igual que a ”, si n es menor o igual a cero se muestra el mensaje “ n debe ser mayor que 0”. Si el área es mayor o igual a cero, el mensaje será “ El área es mayor o igual a 0”, sino el mensaje será “ El área es menor a 0”.

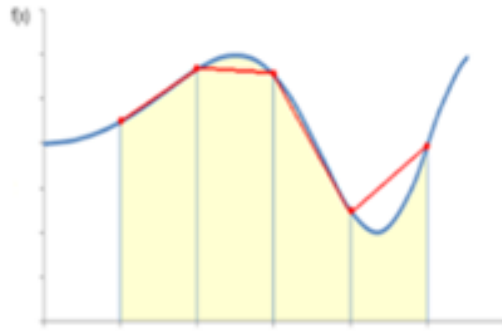


Figura 2.17: Gráfico de la curva con trapecios

Casos de prueba

Use los siguientes casos para verificar si su solución está correcta:

a	b	n	Área	Mensaje
1	10	10000	2612.250	El área es mayor o igual a cero
2.2	1	100		b debe ser mayor o igual que a
1.1	10.1	2	2719.134	El área es mayor o igual a cero
12	14	3	4634.667	El área es mayor o igual a cero
16	18	3	10274.667	El área es mayor o igual a cero

2.3.9. Número Narcisista

Existe muchos números especiales y uno de ellos son los números **narcisistas**, que son aquellos números de k dígitos que cumple que es igual a la suma de las potencias k de sus dígitos. Por ejemplo: 153 es un "número narcisista" porque $153 = 1^3 + 5^3 + 3^3$.

Por eso se desea estudiarlos a profundidad. Se pide evaluar los números que posean exactamente 3 cifras y determinar cuáles números son narcisistas.

Realizar un programa en el lenguaje C, permita calcular e imprimir los números narcisistas de 3 cifras que se encuentren en un intervalo (límite inferior "a" y límite superior "b").

El programa debe validar que el valor de **b** sea mayor que el de **a**. Si **b** es menor que **a** se muestra el mensaje "b debe ser mayor o igual que a". Si el número no tiene 3 cifras, el mensaje será "número inválido".

Casos de prueba

Use los siguientes casos para verificar si su solución está correcta:

a	b	Salida	Mensaje
100	250	153	La cantidad de números narcisista es 1
250	500	370,371,407	La cantidad de números narcisista es 3
500	999	–	La cantidad de números narcisista es 0
100	999	152,379,371,407	La cantidad de números narcisista es 4
999	100		b debe ser mayor o igual que a
1010	1200		Número inválido

2.3.10. Cálculo del arcotangente de un número real (adaptado del laboratorio 3 2020-2)

Dado un valor de x ingresado por el usuario que cumpla:

$$|x| < 1$$

Se le pide calcular e imprimir el valor del arcotangente de x , en grados sexagesimales, de acuerdo a una cantidad de iteraciones indicada también por el usuario. Tenga en cuenta que para ello, deberá utilizar la Serie de Taylor que permite realizar el cálculo aproximado:

$$\text{arcotangente}(x) = \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} x^{2i+1}$$

También deberá calcular e imprimir el valor del arcotangente, en grados sexagesimales, a través de la función correspondiente en lenguaje C. Luego de realizar ambos cálculos deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.0005) entre la serie y el cálculo de la función.” si la diferencia entre los valores es menor a 0.0005 sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual que 0.0005) entre la serie y el cálculo de la función.”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 13

Ingrese el valor de x a evaluar: -0.577350269

iteración	Valor de X	Valor del arcotangente
0	-0.58	-33.079741
1	-0.58	-29.404214
2	-0.58	-30.139319
3	-0.58	-29.964294
4	-0.58	-30.009671
5	-0.58	-29.997295
6	-0.58	-30.000786
7	-0.58	-29.999778
8	-0.58	-30.000074
9	-0.58	-29.999986
10	-0.58	-30.000012
11	-0.58	-30.000004
12	-0.58	-30.000007

El cálculo final del arcotangente con 13 iteraciones es: -30.000007

El cálculo del arcotangente utilizando la función es: -30.000006

Margen de error despreciable (menor que 0.0005) entre la serie y el cálculo de la función.

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 19

Ingrese el valor de x a evaluar: -1

iteración	Valor de X	Valor del arcotangente
0	-1.00	-57.295791
1	-1.00	-38.197194
2	-1.00	-49.656353
3	-1.00	-41.471240
4	-1.00	-47.837439
5	-1.00	-42.628730
6	-1.00	-47.036099
7	-1.00	-43.216379
8	-1.00	-46.586720
9	-1.00	-43.571152
10	-1.00	-46.299523
11	-1.00	-43.808402
12	-1.00	-46.100233
13	-1.00	-43.978167
14	-1.00	-45.953884
15	-1.00	-44.105633
16	-1.00	-45.841869
17	-1.00	-44.204846
18	-1.00	-45.753381

El cálculo final del arcotangente con 19 iteraciones es: -45.753381

El cálculo del arcotangente utilizando la función es: -45.000009

Margen de error considerable (mayor o igual que 0.0005) entre la serie y el cálculo de la función.

2.3.11. Cálculo del logaritmo natural (adaptado del laboratorio 3 2020-2)

Dado un valor de x ingresado por el usuario que cumpla:

$$|x| < 1$$

Se le pide calcular e imprimir el valor del logaritmo natural de $(1+x)$ de acuerdo a una cantidad de iteraciones indicada también por el usuario. Tenga en cuenta que para ello, deberá utilizar la siguiente Serie de Taylor que permite realizar el cálculo aproximado:

$$\ln(1+x) = \sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i} x^i$$

También deberá calcular e imprimir el valor del logaritmo natural de $(1+x)$ a través de la función correspondiente en lenguaje C. Luego de realizar ambos cálculos, deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.001) entre la serie y el cálculo de la función.” si la diferencia (positiva) entre los valores es menor a 0.001 sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual que 0.001) entre la serie y el cálculo de la función.”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 9

Ingrese el valor de x a evaluar: 0.6

iteración	Valor de X	Valor del Log Natural
1	0.60	0.600000
2	0.60	0.420000
3	0.60	0.492000
4	0.60	0.459600
5	0.60	0.475152
6	0.60	0.467376
7	0.60	0.471375
8	0.60	0.469276
9	0.60	0.470395

El cálculo final del logaritmo con 9 iteraciones es: 0.470395

El cálculo utilizando la función de logaritmo es: 0.470004

Margen de error despreciable (menor que 0.001) entre la serie y el cálculo de la función.

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 4

Ingrese el valor de x a evaluar: 0.4

iteración	Valor de X	Valor del Log Natural
1	0.40	0.400000
2	0.40	0.320000
3	0.40	0.341333
4	0.40	0.334933

El cálculo final del logaritmo con 4 iteraciones es: 0.334933

El cálculo utilizando la función de logaritmo es: 0.336472

Margen de error considerable (mayor o igual que 0.001) entre la serie y el cálculo de la función.

2.3.12. Cálculo del arcoseno de un número (adaptado del laboratorio 3 2020-2)

Dado un valor de x ingresado por el usuario que cumpla:

$$|x| < 1$$

Se le pide calcular e imprimir el valor del arcoseno de x, en grados sexagesimales, de acuerdo a una cantidad de iteraciones indicada también por el usuario. Tenga en cuenta que para ello, deberá utilizar la Serie de Taylor que permite realizar el cálculo aproximado:

$$\text{arcoseno}(x) = \sum_{i=0}^{\infty} \frac{(2i)!}{4^i (i!)^2 (2i+1)} x^{2i+1}$$

También deberá calcular e imprimir el valor del arcoseno, en grados sexagesimales, a través de la función correspondiente en lenguaje C. Luego de realizar ambos cálculos deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.005) entre la serie y el cálculo de la función.” si la diferencia (positiva) entre los valores es menor a 0.005 sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual que 0.005) entre la serie y el cálculo de la función.”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 5

Ingrese el valor de x a evaluar: 0.707106781

iteración	Valor de X	Valor del arcoseno
0	0.71	40.514243
1	0.71	43.890430
2	0.71	44.650072
3	0.71	44.876156
4	0.71	44.953087

El cálculo final del arcoseno con 5 iteraciones es: 44.953087

El cálculo del arcoseno utilizando la función es: 45.000009

Margen de error considerable (mayor o igual que 0.005) entre la serie y el cálculo de la función.

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 5

Ingrese el valor de x a evaluar: 0.5

iteración	Valor de X	Valor del arcoseno
0	0.50	28.647896
1	0.50	29.841558
2	0.50	29.975845
3	0.50	29.995828
4	0.50	29.999228

El cálculo final del arcoseno con 5 iteraciones es: 29.999228

El cálculo del arcoseno utilizando la función es: 30.000006

Margen de error despreciable (menor que 0.005) entre la serie y el cálculo de la función.

2.3.13. Cálculo del seno con la serie de Taylor (adaptado del laboratorio 3 2020-2)

Dado un ángulo en grados sexagesimales ingresado por el usuario, se le pide calcular e imprimir el valor aproximado del seno del ángulo hallado a través de la siguiente Serie de Taylor.

$$\text{seno}(x) = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)!} x^{2i+1}$$

La cantidad de iteraciones de la serie será indicada por el usuario y el ángulo x se encuentra en radianes

También deberá calcular e imprimir el valor del seno del ángulo, a través de la función correspondiente en lenguaje C. Luego de realizar ambos cálculos deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.00001) entre la serie y el cálculo de la función.” si la diferencia entre los valores es menor a 0.00001; sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual 0.00001) entre la serie y el cálculo de la función.”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 2

Ingrese el ángulo a evaluar en grados sexagesimales: 60

Iteración	Ángulo	Valor del seno
0	60	1.047197
1	60	0.855801

El cálculo final del seno con 2 iteraciones es: 0.855801

El cálculo del seno utilizando la función es: 0.866025

Margen de error considerable (mayor o igual que 0.00001) entre la serie y el cálculo de la función.

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 6

Ingrese el ángulo a evaluar en grados sexagesimales: 45

Iteración	Ángulo	Valor del seno
0	45	0.785398
1	45	0.704653
2	45	0.707143
3	45	0.707106
4	45	0.707107
5	45	0.707107

El cálculo final del seno con 6 iteraciones es: 0.707107

El cálculo del seno utilizando la función es: 0.707107

Margen de error despreciable (menor que 0.00001) entre la serie y el cálculo de la función.

2.3.14. Aproximación de series para el cálculo de e (adaptado del laboratorio 3 2020-2)

El valor de e puede hallarse a través de distintas sumatorias que permiten llegar a aproximaciones de su valor. Para ello, se le pide calcular e imprimir el valor de la siguiente serie que equivale a una ecuación en función de e . El usuario ingresará la cantidad de iteraciones hasta la que llegará la sumatoria.

$$\sum_{i=1}^{\infty} \frac{(i^2 + 5i + 1)}{(i)!} = 8e - 1$$

También deberá calcular e imprimir el valor de la ecuación en función de e . Luego de realizar ambos cálculos, deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.000015) entre la serie y el cálculo de la fórmula de e .”, si la diferencia (positiva) entre los valores es menor a 0.000015; sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual que 0.000015) entre la serie y el cálculo de la fórmula de e .”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 4

iteración	Valor de la suma
1	7.000000
2	14.500000
3	18.666667
4	20.208333

El cálculo final de la sumatoria usando 4 iteraciones es: 20.208333

El cálculo utilizando la fórmula de e es: 20.746240

Margen de error considerable (mayor o igual que 0.000015) entre la serie y el cálculo de la fórmula de e.

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 10

iteración	Valor de la suma
1	7.000000
2	14.500000
3	18.666667
4	20.208333
5	20.633333
6	20.726389
7	20.743254
8	20.745858
9	20.746208
10	20.746250

El cálculo final de la sumatoria usando 10 iteraciones es: 20.746250

El cálculo utilizando la fórmula de e es: 20.746240

Margen de error despreciable (menor que 0.000015) entre la serie y el cálculo de la fórmula de e.

2.3.15. Cálculo de la serie Maclaurin (adaptado del laboratorio 3 2020-2)

Dado un valor de x ingresado por el usuario que cumpla:

$$|x| < 1$$

Se le pide calcular e imprimir el valor de la sumatoria de acuerdo a una cantidad de iteraciones indicada también por el usuario. Tenga en cuenta que para ello, deberá utilizar la siguiente Serie de Taylor que permite realizar el cálculo aproximado:

$$\frac{1}{(1-x)^3} = \sum_{i=2}^{\infty} \frac{(i-1)i}{2} x^{i-2}$$

También deberá calcular e imprimir el valor de la derecha de la igualdad aplicando la fórmula en lenguaje C. Luego de realizar ambos cálculos, deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.001) entre la serie y el cálculo de la función.” si la diferencia (positiva) entre los valores es menor a 0.001 sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual que 0.001) entre la serie y el cálculo de la función.”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 12

Ingrese el valor de x a evaluar: 0.2

iteración	Valor de X	Resultado
1	0.20	1.000000
2	0.20	1.600000
3	0.20	1.840000
4	0.20	1.920000
5	0.20	1.944000
6	0.20	1.950720
7	0.20	1.952512
8	0.20	1.952973
9	0.20	1.953088
10	0.20	1.953116
11	0.20	1.953123
12	0.20	1.953125

El cálculo final de la sumatoria con 12 iteraciones es: 1.953125

El cálculo utilizando la fórmula es: 1.953125

Margen de error despreciable (menor que 0.001) entre la serie y el cálculo de la función.

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 8

Ingrese el valor de x a evaluar: 0.5

iteración	Valor de X	Resultado
1	0.50	1.000000
2	0.50	2.500000
3	0.50	4.000000
4	0.50	5.250000
5	0.50	6.187500
6	0.50	6.843750
7	0.50	7.281250
8	0.50	7.562500

El cálculo final de la sumatoria con 8 iteraciones es: 7.562500

El cálculo utilizando la fórmula es: 8.000000

Margen de error considerable (mayor o igual que 0.001) entre la serie y el cálculo de la función.

2.3.16. Aproximación de series para el cálculo de e (adaptado del laboratorio 3 2020-2)

El valor de e puede hallarse a través de distintas sumatorias que permiten llegar a aproximaciones de su valor. Para ello, se le pide calcular e imprimir el valor de la siguiente serie que equivale a una ecuación en función de e . El usuario ingresará la cantidad de iteraciones hasta la que llegará la sumatoria.

$$\sum_{i=1}^{\infty} \frac{(2i^2 + 3i + 1)}{(i + 5)!} = 23e - \frac{7501}{120}$$

También deberá calcular e imprimir el valor de la ecuación en función de e . Luego de realizar ambos cálculos, deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.00005) entre la serie

y el cálculo de la fórmula de e ”, si la diferencia (positiva) entre los valores es menor a 0.00005; sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual que 0.00005) entre la serie y el cálculo de la fórmula de e .”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 3

iteración	Valor de la suma
1	0.008333
2	0.011310
3	0.012004

El cálculo final de la sumatoria usando 3 iteraciones es: 0.012004

El cálculo utilizando la fórmula de e es: 0.012107

Margen de error considerable (mayor o igual que 0.00005) entre la serie y el cálculo de la fórmula de e .

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 10

iteración	Valor de la suma
1	0.008333
2	0.011310
3	0.012004
4	0.012128
5	0.012146
6	0.012148
7	0.012149

El cálculo final de la sumatoria usando 7 iteraciones es: 0.012149

El cálculo utilizando la fórmula de e es: 0.012107

Margen de error despreciable (menor que 0.00005) entre la serie y el cálculo de la fórmula de e .

2.3.17. Cálculo de la cosecante de un ángulo con la serie de Taylor (adaptado del laboratorio 3 2020-2)

Dado un ángulo en grados sexagesimales ingresado por el usuario, se le pide calcular e imprimir el valor aproximado de la cosecante del ángulo hallado a través de la siguiente Serie de Taylor.

$$\text{cosecante}(x) = \frac{1}{\sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)!} x^{2i+1}}$$

La cantidad de iteraciones de la serie será indicada por el usuario y tenga en cuenta que el ángulo x se encuentra en radianes.

También deberá calcular e imprimir el valor de la cosecante del ángulo, a través de la función correspondiente en lenguaje C. Luego de realizar ambos cálculos deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.000001) entre la serie y el cálculo de la función.” si la diferencia entre los valores es menor a 0.000001; sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual 0.000001) entre la serie y el cálculo de la función.”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 4

Ingrese el ángulo a evaluar en grados sexagesimales: 30

Iteración	Ángulo	Valor de la cosecante
0	30	1.909860
1	30	2.001305
2	30	1.999992
3	30	2.000000

El cálculo final de la cosecante con 4 iteraciones es: 2.000000

El cálculo de la cosecante utilizando la función es: 2.000000

Margen de error despreciable (menor que 0.000001) entre la serie y el cálculo de la función.

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 4

Ingrese el ángulo a evaluar en grados sexagesimales: 60

Iteración	Ángulo	Valor de la cosecante
0	60	0.954930
1	60	1.168496
2	60	1.154341
3	60	1.154706

El cálculo final de la cosecante con 4 iteraciones es: 1.154706

El cálculo de la cosecante utilizando la función es: 1.154701

Margen de error considerable (mayor o igual que 0.000001) entre la serie y el cálculo de la función.

2.3.18. Cálculo del arcosecante de un número (adaptado del laboratorio 3 2020-2)

Dado un valor de x ingresado por el usuario que cumpla:

$$|x| < 1$$

Se le pide calcular e imprimir el valor del arcosecante de x , en grados sexagesimales. Tenga en cuenta que para ello, deberá utilizar la siguiente fórmula:

$$\text{arcosecante}(x) = \frac{\pi}{2} - \text{arcoseno}\left(\frac{1}{x}\right)$$

Asimismo, debe considerar que la Serie de Taylor permitirá realizar el cálculo aproximado del arcoseno de acuerdo a una cantidad de iteraciones indicada también por el usuario:

$$\text{arcoseno}(y) = \sum_{i=0}^{\infty} \frac{(2i)!}{4^i (i!)^2 (2i+1)} y^{2i+1}$$

También deberá calcular e imprimir el valor del arcosecante, en grados sexagesimales, a través de la función correspondiente en lenguaje C. Luego de realizar ambos cálculos deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.000001) entre la serie y el cálculo de la función.” si la diferencia (positiva) entre los valores es menor a 0.000001 sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual que 0.000001) entre la serie y el cálculo de la función.”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 8

Ingrese el valor de x a evaluar: 1.154700538

iteración	Valor de X	Valor del arcosecante
0	1.15	40.380389
1	1.15	34.177938
2	1.15	32.084610
3	1.15	31.150089
4	1.15	30.673094
5	1.15	30.409663
6	1.15	30.256416
7	1.15	30.255060

El cálculo final del arcosecante con 8 iteraciones es: 30.255060

El cálculo del arcosecante utilizando la función es: 29.999987

Margen de error considerable (mayor o igual que 0.00001) entre la serie y el cálculo de la función.

Caso de prueba

la cantidad de iteraciones a realizar: 8

Ingrese el valor de x a evaluar: 1.414213562

iteración	Valor de X	Valor del arcosecante
0	1.41	49.485757
1	1.41	46.109570
2	1.41	45.349928
3	1.41	45.123844
4	1.41	45.046913
5	1.41	45.018588
6	1.41	45.007603
7	1.41	45.007539

El cálculo final del arcosecante con 8 iteraciones es: 45.007539

El cálculo del arcosecante utilizando la función es: 44.999991

Margen de error considerable (mayor o igual que 0.00001) entre la serie y el cálculo de la función.

2.3.19. Cálculo de e en función a aproximación de series (adaptado del laboratorio 3 2020-2)

Se le pide calcular e imprimir el valor de la siguiente serie que equivale a una ecuación en función de e . Para ello, el usuario ingresará la cantidad de iteraciones hasta la que llegará la sumatoria.

$$\sum_{i=1}^{\infty} \frac{(3i^2 + 2i + 6)}{(i + 2)!} = 10e - 24$$

También deberá calcular e imprimir el valor de la ecuación en función de e . Luego de realizar ambos cálculos, deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.0001) entre la serie y el cálculo de la fórmula de e .”, si la diferencia (positiva) entre los valores es menor a 0.0001; sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual que 0.0001) entre la serie y el cálculo de la fórmula de e .”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para

verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 5

iteración	Valor de la suma
1	1.833333
2	2.750000
3	3.075000
4	3.161111
5	3.179167

El cálculo final de la sumatoria usando 5 iteraciones es: 3.179167

El cálculo utilizando la fórmula de e es: 3.182800

Margen de error considerable (mayor o igual que 0.0001) entre la serie y el cálculo de la fórmula de e.

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 12

iteración	Valor de la suma
1	1.833333
2	2.750000
3	3.075000
4	3.161111
5	3.179167
6	3.182292
7	3.182752
8	3.182811
9	3.182818
10	3.182818
11	3.182818
12	3.182819

El cálculo final de la sumatoria usando 12 iteraciones es: 3.182819

El cálculo utilizando la fórmula de e es: 3.182800

Margen de error despreciable (menor que 0.0001) entre la serie y el cálculo de la fórmula de e.

2.3.20. Cálculo de la función racional (adaptado del laboratorio 3 2020-2)

Dado un valor de t ingresado por el usuario que cumpla:

$$|t| < 1$$

Se le pide calcular e imprimir el valor de la serie de acuerdo a una cantidad de iteraciones indicada también por el usuario y denotada por n . Tenga en cuenta que para ello, deberá utilizar la siguiente Serie de Taylor que permite realizar el cálculo aproximado:

$$\frac{1}{1+t^2} = \sum_{i=0}^{n-1} (-1)^i t^{2i} + \frac{(-1)^n t^{2n}}{1+t^2}$$

También deberá calcular e imprimir el valor de la derecha de la igualdad aplicando la fórmula en lenguaje C. Luego de realizar ambos cálculos, deberá compararlos e imprimir el mensaje “Margen de error despreciable (menor que 0.00005) entre la serie y el cálculo de la función.” si la diferencia (positiva) entre los valores es

menor a 0.00005 sino deberá imprimir el mensaje “Margen de error considerable (mayor o igual que 0.00005) entre la serie y el cálculo de la función.”

Deberá imprimir los valores según las tablas indicadas en los casos de pruebas. Use los siguientes casos para verificar si su solución está correcta:

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 6

Ingrese el valor de t a evaluar: 0.4

iteración	Valor de t	Resultado
0	0.40	1.000014
1	0.40	0.840029
2	0.40	0.865643
3	0.40	0.861562
4	0.40	0.862232
5	0.40	0.862141

El cálculo final de la sumatoria con 6 iteraciones es: 0.862141

El cálculo utilizando la fórmula es: 0.862069

Margen de error considerable (mayor o igual que 0.00005) entre la serie y el cálculo de la función.

Caso de prueba

Ingrese la cantidad de iteraciones a realizar: 10

Ingrese el valor de t a evaluar: 0.5

iteración	Valor de t	Resultado
0	0.50	1.000001
1	0.50	0.750002
2	0.50	0.812502
3	0.50	0.796878
4	0.50	0.800785
5	0.50	0.799809
6	0.50	0.800054
7	0.50	0.799994
8	0.50	0.800010
9	0.50	0.800007

El cálculo final de la sumatoria con 10 iteraciones es: 0.800007

El cálculo utilizando la fórmula es: 0.800000

Margen de error despreciable (menor que 0.00005) entre la serie y el cálculo de la función.

2.3.21. Rectas paralelas y perpendiculares (adaptado del laboratorio 3 2021-1)

La ecuación general de una recta es una expresión de la forma $Ax + By + C = 0$ donde A, B y C son números reales. La pendiente de la recta (m) está determinada por el coeficiente de la variable x , la misma que al ser despejada de la ecuación da como resultado: $m = -A/B$.

Cabe precisar que la pendiente de una recta también se puede calcular si conocemos las coordenadas de dos puntos que pasen por la recta de la siguiente manera: Sean $P(x_1, y_1)$ y $Q(x_2, y_2)$ los puntos que pasan por una recta, la pendiente (m) se puede calcular con la siguiente fórmula: $m = (y_2 - y_1)/(x_2 - x_1)$.

Dentro de la geometría analítica, existen diversos tipos de rectas, para este problema vamos a trabajar con las rectas paralelas y perpendiculares.

Las *rectas paralelas* son dos rectas en un plano que nunca se intersectan. A nivel de pendientes, dos rectas se consideran paralelas si tienen la misma pendiente.

Las *rectas perpendiculares* son dos rectas en un plano que se intersectan formando un ángulo de 90 grados. A nivel de pendientes, dos rectas se consideran perpendiculares si la multiplicación de sus pendientes da como resultado -1.

Se le pide que elabore un programa en lenguaje C que realice lo siguiente:

- Lea los coeficientes A, B y C de la ecuación general de una recta y con estos datos calcule su pendiente.
- Lea la cantidad de pares de puntos de rectas que se van a procesar.
- Por cada par de puntos realice lo siguiente:
 - Lea 2 puntos, $P(x_1, y_1)$ y $Q(x_2, y_2)$, que representan a los puntos que pasan por una recta.
 - Determine si esta recta formada por los dos puntos leídos es perpendicular o paralela con respecto a la recta ingresada en la ecuación general de la recta (primer paso).
 - Si la recta formada por los dos puntos leídos no es perpendicular ni paralela con respecto a la recta ingresada en la ecuación general de la recta (primer paso), la recta formada por dichos puntos se considera inválida.
- Al finalizar debe mostrar un resumen de los datos procesados con la siguiente información:
 - La cantidad de rectas paralelas encontradas.
 - La mayor distancia formada por los puntos para una recta paralela, de ser el caso.
 - La cantidad de rectas perpendiculares encontradas.
 - La cantidad de rectas inválidas encontradas.

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

Ingrese los valores de A, B y C de la ecuación general de la recta: 0.333 -1 4

La pendiente de la recta formada por la ecuación general de la recta ingresada es: 0.333

Ingrese la cantidad de pares de puntos a procesar: 3

Coordenadas del par de puntos 1

Ingrese las coordenadas x e y del punto P: -1 5

Ingrese las coordenadas x e y del punto Q: 2 7

La pendiente de la recta formada por los 2 puntos ingresados es: 0.667

La puntos ingresados no forman una recta paralela o perpendicular respecto a la ecuación $0.333x + -1.000y + 4.000 = 0$

Coordenadas del par de puntos 2

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

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

La pendiente de la recta formada por los 2 puntos ingresados es: -3.000

La pendiente de la recta formada por los puntos (1.000 , 2.000) y (0.000 , 0.500) es perpendicular a la recta de la ecuación $0.333x + -1.000y + 4.000 = 0$

Coordenadas del par de puntos 3

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

Ingrese las coordenadas x e y del punto Q: 2 -5

La pendiente de la recta formada por los 2 puntos ingresados es: -3.000

La pendiente de la recta formada por los puntos (0.000 , 1.000) y (2.000 , -5.000) es perpendicular a la recta de la ecuación $0.333x + -1.000y + 4.000 = 0$

La cantidad de rectas paralelas encontradas fue 0

La cantidad de rectas perpendiculares encontradas fue 2

La cantidad de rectas inválidas fue 1

Caso de prueba

Ingrese los valores de A, B y C de la ecuación general de la recta: 2 -1 1
 La pendiente de la recta formada por la ecuación general de la recta ingresada es: 2.000
 Ingrese la cantidad de pares de puntos a procesar: 4
 Coordenadas del par de puntos 1
 Ingrese las coordenadas x e y del punto P: 0 1
 Ingrese las coordenadas x e y del punto Q: -0.5 0
 La pendiente de la recta formada por los 2 puntos ingresados es: 2.000
 La pendiente de la recta formada por los puntos (0.000 , 1.000) y (-0.500 , 0.000) es paralela a la recta de la ecuación $2.000x + -1.000y + 1.000 = 0$
 Coordenadas del par de puntos 2
 Ingrese las coordenadas x e y del punto P: 1 0
 Ingrese las coordenadas x e y del punto Q: -1 1
 La pendiente de la recta formada por los 2 puntos ingresados es: -0.500
 La pendiente de la recta formada por los puntos (1.000 , 0.000) y (-1.000 , 1.000) es perpendicular a la recta de la ecuación $2.000x + -1.000y + 1.000 = 0$
 Coordenadas del par de puntos 3
 Ingrese las coordenadas x e y del punto P: 2 0
 Ingrese las coordenadas x e y del punto Q: 0 1
 La pendiente de la recta formada por los 2 puntos ingresados es: -0.500
 La pendiente de la recta formada por los puntos (2.000 , 0.000) y (0.000 , 1.000) es perpendicular a la recta de la ecuación $2.000x + -1.000y + 1.000 = 0$
 Coordenadas del par de puntos 4
 Ingrese las coordenadas x e y del punto P: -1 5
 Ingrese las coordenadas x e y del punto Q: -0.5 6
 La pendiente de la recta formada por los 2 puntos ingresados es: 2.000
 La pendiente de la recta formada por los puntos (-1.000 , 5.000) y (-0.500 , 6.000) es paralela a la recta de la ecuación $2.000x + -1.000y + 1.000 = 0$
 La cantidad de rectas paralelas encontradas fue 2
 La mayor distancia de puntos formada por una de las rectas paralelas fue 1.118034
 La cantidad de rectas perpendiculares encontradas fue 2
 La cantidad de rectas inválidas fue 0

2.3.22. Rectas tangentes a la circunferencia (adaptado del laboratorio 3 2021-1)

La circunferencia es una línea curva cerrada cuyos puntos están todos a la misma distancia de un punto fijo llamado centro. El radio es el segmento que une el centro de la circunferencia con un punto cualquiera de la misma.

El diámetro es un segmento que pasa por el centro de la circunferencia. El diámetro mide el doble del radio.

Si conocemos las coordenadas de dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$ que pertenecen a la circunferencia y además, dichos puntos forman parte del diámetro de la misma, el centro de la circunferencia $C(x_r, y_r)$ se puede calcular de la siguiente manera: $x_r = (x_1 + x_2)/2$ y $y_r = (y_1 + y_2)/2$

La ecuación general de una recta es una expresión de la forma $Ax + By + C = 0$ donde A, B y C son números reales. La pendiente de la recta (m) está determinada por el coeficiente de la variable x, la misma que al ser despejada de la ecuación da como resultado: $m = -A/B$.

La recta tangente a una circunferencia es perpendicular al radio correspondiente al punto de tangencia, el cual pertenece a la circunferencia.

La distancia de un punto a una recta es la longitud de un segmento que, partiendo del punto del plano, sea perpendicular a la recta. Para que la longitud de ese segmento sea la mínima, el segmento y la recta deben de ser perpendiculares.

La fórmula para hallar la distancia de un punto $P(x_p, y_p)$ a una recta $Ax + By + C = 0$ es la siguiente:

$$d = \left| \frac{Ax_p + By_p + C}{\sqrt{A^2 + B^2}} \right|$$

Se le pide que elabore un programa en lenguaje C que realice lo siguiente:

- Lea las coordenadas de los puntos P y Q que pertenecen a una circunferencia y además, forman parte del diámetro de la misma.
- Calcule el centro, el radio y el área de la circunferencia a la cuál pertenecen los puntos P y Q.
- Imprima los valores del centro, radio y área calculados en el punto anterior.
- Lea la cantidad de rectas que se van a procesar.
- Por cada recta realice lo siguiente:
 - Lea los coeficientes A, B y C de la ecuación de la recta $Ax + By + C = 0$ que validaremos si es tangente o no a la circunferencia.
 - Validar si esta recta es tangente a la circunferencia.
- Al finalizar debe mostrar un resumen de los datos procesados con la siguiente información:
 - La cantidad de rectas tangentes a la circunferencia.
 - La cantidad de rectas con pendientes positivas que existe en las rectas tangentes a la circunferencia, de ser necesario.
 - La cantidad de rectas con pendientes negativas que existe en las rectas tangentes a la circunferencia, de ser necesario.
 - La cantidad de rectas NO tangentes a la circunferencia.

Nota.- Recuerde que la distancia entre dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$ es igual a: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

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

Ingrese las coordenadas del punto P: 1 1.192

Ingrese las coordenadas del punto Q: 1 -4.192

El centro de la circunferencia formado por el punto medio de P(1.000,1.192) y Q(1.000,-4.192) es (1.000,-1.500)

El radio de la circunferencia al cual pertenecen los puntos P(1.000,1.192) y Q(1.000,-4.192) es 2.692

El área de la circunferencia al cual pertenecen los puntos P(1.000,1.192) y Q(1.000,-4.192) es 22.766

Ingrese la cantidad de puntos a procesar: 3

Ingrese los coeficientes A, B y C de la ecuación de la recta: 3 -1 6

La distancia del centro de la circunferencia a la recta ingresada es 3.320

La recta con ecuación $3.000x + -1.000y + 6.000 = 0$ NO es una recta tangente a la circunferencia con centro C(1.000,-1.500) y radio 2.692

Ingrese los coeficientes A, B y C de la ecuación de la recta: 3 -1 4.01

La distancia del centro de la circunferencia a la recta ingresada es 2.691

La recta con ecuación $3.000x + -1.000y + 4.010 = 0$ es una recta tangente a la circunferencia con centro C(1.000,-1.500) y radio 2.692

Ingrese los coeficientes A, B y C de la ecuación de la recta: 3 -1 -13.01

La distancia del centro de la circunferencia a la recta ingresada es 2.691

La recta con ecuación $3.000x + -1.000y + -13.010 = 0$ es una recta tangente a la circunferencia con centro C(1.000,-1.500) y radio 2.692

La cantidad de rectas tangentes encontradas fue: 2

La cantidad de pendientes positivas de las rectas tangentes encontradas fue: 2

La cantidad de pendientes negativas de las rectas tangentes encontradas fue: 0

La cantidad de rectas NO tangentes encontradas fue: 1

Caso de prueba

Ingrese las coordenadas del punto P: 1 1.192

Ingrese las coordenadas del punto Q: 1 -4.192

El centro de la circunferencia formado por el punto medio de P(1.000,1.192) y Q(1.000,-4.192) es (1.000,-1.500)

El radio de la circunferencia al cual pertenecen los puntos P(1.000,1.192) y Q(1.000,-4.192) es 2.692

El área de la circunferencia al cual pertenecen los puntos P(1.000,1.192) y Q(1.000,-4.192) es 22.766

Ingrese la cantidad de puntos a procesar: 3

Ingrese los coeficientes A, B y C de la ecuación de la recta: 3 -1 6

La distancia del centro de la circunferencia a la recta ingresada es 3.320

La recta con ecuación $3.000x + -1.000y + 6.000 = 0$ NO es una recta tangente a la circunferencia con centro C(1.000,-1.500) y radio 2.692

Ingrese los coeficientes A, B y C de la ecuación de la recta: 3 2 5

La distancia del centro de la circunferencia a la recta ingresada es 1.387

La recta con ecuación $3.000x + 2.000y + 5.000 = 0$ NO es una recta tangente a la circunferencia con centro C(1.000,-1.500) y radio 2.692

Ingrese los coeficientes A, B y C de la ecuación de la recta: 3 2 -3

La distancia del centro de la circunferencia a la recta ingresada es 0.832

La recta con ecuación $3.000x + 2.000y + -3.000 = 0$ NO es una recta tangente a la circunferencia con centro C(1.000,-1.500) y radio 2.692

La cantidad de rectas tangentes encontradas fue: 0

La cantidad de rectas NO tangentes encontradas fue: 3

2.3.23. Rectas tangentes y rectas normales a la circunferencia (adaptado del laboratorio 3 2021-1)

La circunferencia es una línea curva cerrada cuyos puntos están todos a la misma distancia de un punto fijo llamado centro. El radio es el segmento que une el centro de la circunferencia con un punto cualquiera de la misma.

El diámetro es un segmento que pasa por el centro de la circunferencia. El diámetro mide el doble del radio.

Si conocemos las coordenadas de dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$ que pertenecen a la circunferencia y además, dichos puntos forman parte del diámetro de la misma, el centro de la circunferencia $C(x_r, y_r)$ se puede calcular de la siguiente manera: $x_r = (x_1 + x_2)/2$ y $y_r = (y_1 + y_2)/2$

La ecuación general de una recta es una expresión de la forma $Ax + By + C = 0$ donde A, B y C son números reales. La pendiente de la recta (m) está determinada por el coeficiente de la variable x, la misma que al ser despejada de la ecuación da como resultado: $m = -A/B$.

Si se conocen dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$ que pasan por una recta, también se puede calcular la pendiente de la misma de la siguiente manera: $m = (y_2 - y_1)/(x_2 - x_1)$. Además, la ecuación de la recta sería: $y = mx + b$.

La recta tangente a una circunferencia es perpendicular al radio correspondiente al punto de tangencia, el cual pertenece a la circunferencia.

La recta perpendicular a la tangente en el punto de tangencia se llama recta normal.

Se le pide que elabore un programa en lenguaje C que realice lo siguiente:

- Lea las coordenadas de los puntos P y Q que pertenecen a una circunferencia y además, forman parte del diámetro de la misma.
- Calcule el centro y el radio de la circunferencia a la cuál pertenecen los puntos P y Q.
- Imprima los valores del centro y radio calculados en el punto anterior.
- Lea la cantidad de juego de datos que se van a procesar.
- Por cada juego de datos realice lo siguiente:
 - Lea los coeficientes A, B y C de la ecuación de la recta $Ax + By + C = 0$ que validaremos si es tangente o no a la circunferencia.
 - Lea las coordenadas del punto de tangencia T.
 - Validar si esta recta es tangente a la circunferencia en el punto de tangencia T ingresado.
 - Si la recta es tangente, los datos ingresados se consideran válidos y además, debe calcular la ecuación de la recta normal que es perpendicular a esta recta tangente, que pasa por el centro de la circunferencia y por el punto de tangencia.
- Al finalizar debe mostrar un resumen de los datos procesados con la siguiente información:
 - La cantidad de datos que dieron un resultado válido.
 - La cantidad de rectas normales con pendientes positivas encontradas, de ser necesario.
 - La cantidad de rectas normales con pendientes negativas encontradas, de ser necesario.
 - La cantidad de datos que dieron un resultado inválido.

Nota.- Recuerde que la distancia entre dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$ es igual a: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

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

Ingrese las coordenadas del punto P: 8 4

Ingrese las coordenadas del punto Q: 2 0

El centro de la circunferencia formado por el punto medio de P(8.000,4.000) y Q(2.000,0.000) es (5.000,2.000)

El radio de la circunferencia al cual pertenecen los puntos P(8.000,4.000) y Q(2.000,0.000) es 3.606

Ingrese la cantidad de puntos a procesar: 3

Ingrese los coeficientes A, B y C de la ecuación de la recta: 1.5 1 -3

Ingrese los coordenadas del punto de tangencia T: 2 0

La distancia del centro de la circunferencia a la recta ingresada es 3.606

La recta con ecuación $1.500x + 1.000y + -3.000 = 0$ es una recta tangente a la circunferencia con centro C(5.000,2.000) y radio 3.606

La recta con ecuación $y = 0.667X + -1.333$ es una recta normal a la circunferencia con centro C(5.000,2.000) y radio 3.606

Ingrese los coeficientes A, B y C de la ecuación de la recta: 1 2 3

Ingrese los coordenadas del punto de tangencia T: 5 3

La distancia del centro de la circunferencia a la recta ingresada es 2.500

La recta con ecuación $1.000x + 2.000y + 3.000 = 0$ NO es una recta tangente a la circunferencia con centro C(5.000,2.000) y radio 3.606

Ingrese los coeficientes A, B y C de la ecuación de la recta: 2.5 3 6

Ingrese los coordenadas del punto de tangencia T: 1 3

La distancia del centro de la circunferencia a la recta ingresada es 4.031

La recta con ecuación $2.500x + 3.000y + 6.000 = 0$ NO es una recta tangente a la circunferencia con centro C(5.000,2.000) y radio 3.606

La cantidad de datos que dieron un resultado válido fue: 1

La cantidad de pendientes positivas de las rectas normales encontradas fue: 1

La cantidad de pendientes negativas de las rectas normales encontradas fue: 0

La cantidad de datos que dieron un resultado inválido fue: 2

Caso de prueba

Ingrese las coordenadas del punto P: 8 4

Ingrese las coordenadas del punto Q: 2 0

El centro de la circunferencia formado por el punto medio de $P(8.000,4.000)$ y $Q(2.000,0.000)$ es $(5.000,2.000)$

El radio de la circunferencia al cual pertenecen los puntos $P(8.000,4.000)$ y $Q(2.000,0.000)$ es 3.606

Ingrese la cantidad de puntos a procesar: 2

Ingrese los coeficientes A, B y C de la ecuación de la recta: 1 2 3

Ingrese los coordenadas del punto de tangencia T: 5 3

La distancia del centro de la circunferencia a la recta ingresada es 2.500

La recta con ecuación $1.000x + 2.000y + 3.000 = 0$ NO es una recta tangente a la circunferencia con centro $C(5.000,2.000)$ y radio 3.606

Ingrese los coeficientes A, B y C de la ecuación de la recta: 2.5 3 6

Ingrese los coordenadas del punto de tangencia T: 1 3

La distancia del centro de la circunferencia a la recta ingresada es 4.031

La recta con ecuación $2.500x + 3.000y + 6.000 = 0$ NO es una recta tangente a la circunferencia con centro $C(5.000,2.000)$ y radio 3.606

La cantidad de datos que dieron un resultado válido fue: 0

La cantidad de datos que dieron un resultado inválido fue: 2

2.3.24. Los segmentos en el plano cartesiano (adaptado del laboratorio 3 2021-1)

Un segmento es la porción de recta limitada por dos puntos, llamados extremos. Los segmentos se nombran por los puntos que lo limitan o por una letra minúscula.

La longitud o distancia del segmento viene dado por la distancia que existe entre los dos puntos que conforman el segmento. La distancia entre dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$ es igual a: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Se le pide que elabore un programa en lenguaje C que realice lo siguiente:

- Lea las coordenadas de los puntos P y Q que forman el segmento PQ.
- Calcule la distancia (longitud) del segmento PQ.
- Lea la cantidad de segmentos que se va a procesar.
- Por cada segmento realice lo siguiente:
 - Lea las coordenadas de los puntos, $R(x_3, y_3)$ y $S(x_4, y_4)$, que representan a los extremos del segmento a procesar.
 - Calcule la distancia (longitud) del segmento RS.
 - Determine si la distancia del segmento RS es igual a la distancia del segmento PQ.
- Al finalizar debe mostrar un resumen de los datos procesados con la siguiente información:
 - La cantidad de segmentos RS de igual distancia que el segmento PQ.
 - La cantidad de segmentos RS de diferente distancia que el segmento PQ.
 - La menor distancia de un segmento RS con diferente distancia que el segmento PQ, de ser el caso.
 - La mayor distancia de un segmento RS con diferente distancia que el segmento PQ, de ser el caso.

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

Ingrese las coordenadas del punto P: 4 -3

Ingrese las coordenadas del punto Q: 3 0

La distancia del segmento PQ es: 3.162

Ingrese la cantidad de segmentos a procesar: 3

Coordenadas del par de puntos del segmento 1

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

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

La distancia del segmento RS ingresado es: 3.162

El segmento RS formado por los puntos (3.000,0.000) y (0.000,1.000) tiene la misma distancia que el segmento PQ.

Coordenadas del par de puntos del segmento 2

Ingrese las coordenadas x e y del punto R: 4 -3

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

La distancia del segmento RS ingresado es: 5.657

El segmento RS formado por los puntos (4.000,-3.000) y (0.000,1.000) No tiene la misma distancia que el segmento PQ.

Coordenadas del par de puntos del segmento 3

Ingrese las coordenadas x e y del punto R: -2.5 -5

Ingrese las coordenadas x e y del punto S: 3.2 4.5

La distancia del segmento RS ingresado es: 11.079

El segmento RS formado por los puntos (-2.500,-5.000) y (3.200,4.500) No tiene la misma distancia que el segmento PQ.

La cantidad de segmentos de igual distancia que el segmento PQ fue 1

La cantidad de segmentos de diferente distancia que el segmento PQ fue 2

La menor distancia de un segmento con diferente distancia que el segmento PQ fue 5.657

La mayor distancia de un segmento con diferente distancia que el segmento PQ fue 11.079

Caso de prueba

Ingrese las coordenadas del punto P: 4 -3
 Ingrese las coordenadas del punto Q: 3 0
 La distancia del segmento PQ es: 3.162
 Ingrese la cantidad de segmentos a procesar: 4
 Coordenadas del par de puntos del segmento 1
 Ingrese las coordenadas x e y del punto R: 3 0
 Ingrese las coordenadas x e y del punto S: 0 1
 La distancia del segmento RS ingresado es: 3.162
 El segmento RS formado por los puntos (3.000,0.000) y (0.000,1.000) tiene la misma distancia que el segmento PQ.
 Coordenadas del par de puntos del segmento 2
 Ingrese las coordenadas x e y del punto R: 4 1
 Ingrese las coordenadas x e y del punto S: 1 2
 La distancia del segmento RS ingresado es: 3.162
 El segmento RS formado por los puntos (4.000,1.000) y (1.000,2.000) tiene la misma distancia que el segmento PQ.
 Coordenadas del par de puntos del segmento 3
 Ingrese las coordenadas x e y del punto R: 4.5 1.5
 Ingrese las coordenadas x e y del punto S: 1.5 2.5
 La distancia del segmento RS ingresado es: 3.162
 El segmento RS formado por los puntos (4.500,1.500) y (1.500,2.500) tiene la misma distancia que el segmento PQ.
 Coordenadas del par de puntos del segmento 4
 Ingrese las coordenadas x e y del punto R: 6 3
 Ingrese las coordenadas x e y del punto S: 3 4
 La distancia del segmento RS ingresado es: 3.162
 El segmento RS formado por los puntos (6.000,3.000) y (3.000,4.000) tiene la misma distancia que el segmento PQ.
 La cantidad de segmentos de igual distancia que el segmento PQ fue 4
 La cantidad de segmentos de diferente distancia que el segmento PQ fue 0

2.3.25. La Circunferencia (adaptado del laboratorio 3 2021-1)

La circunferencia es una línea curva cerrada cuyos puntos están todos a la misma distancia de un punto fijo llamado centro. El radio es el segmento que une el centro de la circunferencia con un punto cualquiera de la misma.

El diámetro es un segmento que pasa por el centro de la circunferencia. El diámetro mide el doble del radio.

Si conocemos las coordenadas de dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$ que pertenecen a la circunferencia y además, dichos puntos forman parte del diámetro de la misma, el centro de la circunferencia $C(x_r, y_r)$ se puede calcular de la siguiente manera: $x_r = (x_1 + x_2)/2$ y $y_r = (y_1 + y_2)/2$

Se le pide que elabore un programa en lenguaje C que realice lo siguiente:

- Lea las coordenadas de los puntos P y Q que pertenecen a una circunferencia y además, forman parte del diámetro de la misma.
- Calcule el centro, el radio y el área de la circunferencia a la cuál pertenecen los puntos P y Q.
- Imprima los valores del centro, radio y área calculados en el punto anterior.
- Lea la cantidad de puntos que se van a procesar.
- Por cada punto realice lo siguiente:

- Lea las coordenadas del punto $Z(x, y)$ que representa a un punto que puede o no pertenecer a la circunferencia ingresada en el primer paso.
 - Determine si este punto pertenece o no a la circunferencia ingresada en el primer paso.
 - Si el punto pertenece a la circunferencia, debe ir calculando el punto que tenga el mayor valor para la coordenada x . En caso existan dos o más puntos con la mayor coordenada x , debe considerar como el mayor al último punto ingresado.
- Al finalizar debe mostrar un resumen de los datos procesados con la siguiente información:
- La cantidad de puntos que pertenecen a la circunferencia.
 - Las coordenadas del punto, que pertenece a la circunferencia, que tiene la mayor coordenada x , de ser necesario.
 - La cantidad de puntos que no pertenecen a la circunferencia.

Nota.- Recuerde que la distancia entre dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$ es igual a: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

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

Ingrese las coordenadas del punto P: 2 -1
 Ingrese las coordenadas del punto Q: -4 7
 El centro de la circunferencia formado por el punto medio de P(2.000,-1.000) y Q(-4.000,7.000) es (-1.000,3.000)
 El radio de la circunferencia al cual pertenecen los puntos P(2.000,-1.000) y Q(-4.000,7.000) es 5.000
 El área de la circunferencia al cual pertenecen los puntos P(2.000,-1.000) y Q(-4.000,7.000) es 78.538
 Ingrese la cantidad de puntos a procesar: 3
 Ingrese las coordenadas del punto Z: -2 -1
 El radio es 4.123106
 El punto Z(-2.000,-1.000) NO pertenece a la circunferencia con centro C(-1.000,3.000) y radio 5.000
 Ingrese las coordenadas del punto Z: 1 8
 El radio es 5.385165
 El punto Z(1.000,8.000) NO pertenece a la circunferencia con centro C(-1.000,3.000) y radio 5.000
 Ingrese las coordenadas del punto Z: -4 3
 El radio es 3.000000
 El punto Z(-4.000,3.000) NO pertenece a la circunferencia con centro C(-1.000,3.000) y radio 5.000
 La cantidad de puntos que pertenecen a la circunferencia fue: 0
 La cantidad de puntos que no pertenecen a la circunferencia fue: 3

Caso de prueba

Ingrese las coordenadas del punto P: 2 -1

Ingrese las coordenadas del punto Q: -4 7

El centro de la circunferencia formado por el punto medio de P(2.000,-1.000) y Q(-4.000,7.000) es (-1.000,3.000)

El radio de la circunferencia al cual pertenecen los puntos P(2.000,-1.000) y Q(-4.000,7.000) es 5.000

El área de la circunferencia al cual pertenecen los puntos P(2.000,-1.000) y Q(-4.000,7.000) es 78.538

Ingrese la cantidad de puntos a procesar: 4

Ingrese las coordenadas del punto Z: -1 8

El radio es 5.000000

El punto Z(-1.000,8.000) pertenece a la circunferencia con centro C(-1.000,3.000) y radio 5.000

Ingrese las coordenadas del punto Z: 4 -3

El radio es 7.810250

El punto Z(4.000,-3.000) NO pertenece a la circunferencia con centro C(-1.000,3.000) y radio 5.000

Ingrese las coordenadas del punto Z: 4 3

El radio es 5.000000

El punto Z(4.000,3.000) pertenece a la circunferencia con centro C(-1.000,3.000) y radio 5.000

Ingrese las coordenadas del punto Z: 2 6

El radio es 4.242641

El punto Z(2.000,6.000) NO pertenece a la circunferencia con centro C(-1.000,3.000) y radio 5.000

La cantidad de puntos que pertenecen a la circunferencia fue: 2

Las coordenadas del punto que pertenece a la circunferencia con mayor coordenada x fue (4.000,3.000)

La cantidad de puntos que no pertenecen a la circunferencia fue: 2