**Instituto Politécnico Nacional**

**ESCOM**

**Practica 1**

**Algoritmos**

**Algoritmia y Programación Estructurada**

**Docente: Mtro. Miguel Ángel Rodríguez Castillo**

**Alumno: García King Ricardo Neftaly**

**1CV4**

Índice

Planteamiento del Problema 2

Diseño y Funcionamiento de la Solución 3

Implementación de la Solución 8

Funcionamiento 11

Conclusiones y Bibliografía 29

Anexos 31

Planteamiento del Problema

1.- Dados 2 números enteros positivos, retornar el Máximo Común Divisor entre ellos

2.- Identifica el número menor de diez números positivos ingresados por el usuario, considera que los datos se ingresan de uno a uno.

3.- Determinar si 2 rectas son perpendiculares. Considera que el usuario ingresa las coordenadas de 2 puntos de cada una de ellas.

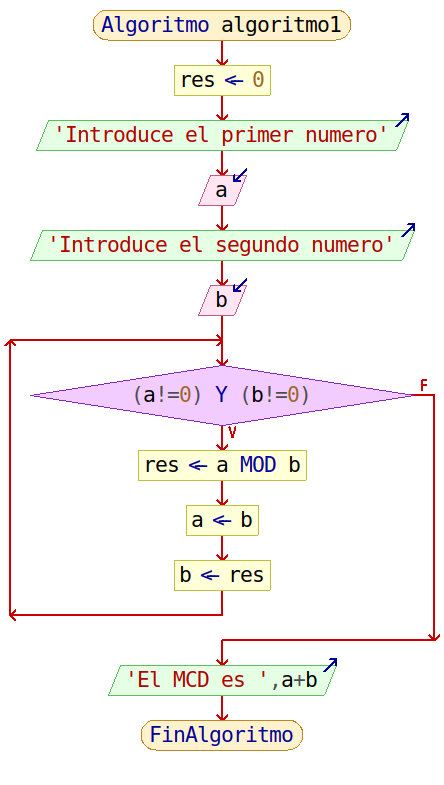
4.- Analiza la siguiente serie numérica, desarrolla su algoritmo y muestra los siguientes 7 números de la sucesión.

3 5 4 7 6 \_\_ \_\_ \_\_ \_\_ \_\_ \_\_ \_\_

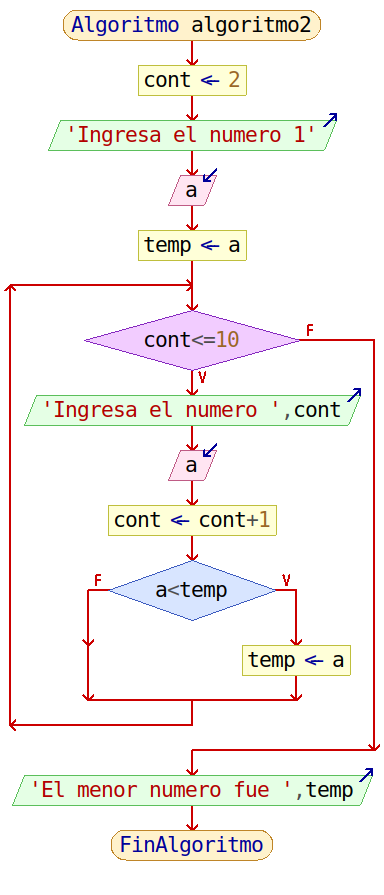
5.- Calcule la media de una serie de números positivos, suponiendo que los datos se leen uno a uno.

Un valor de cero como entrada indicará que se ha alcanzado al final de la serie de números positivos ingresados por el usuario.

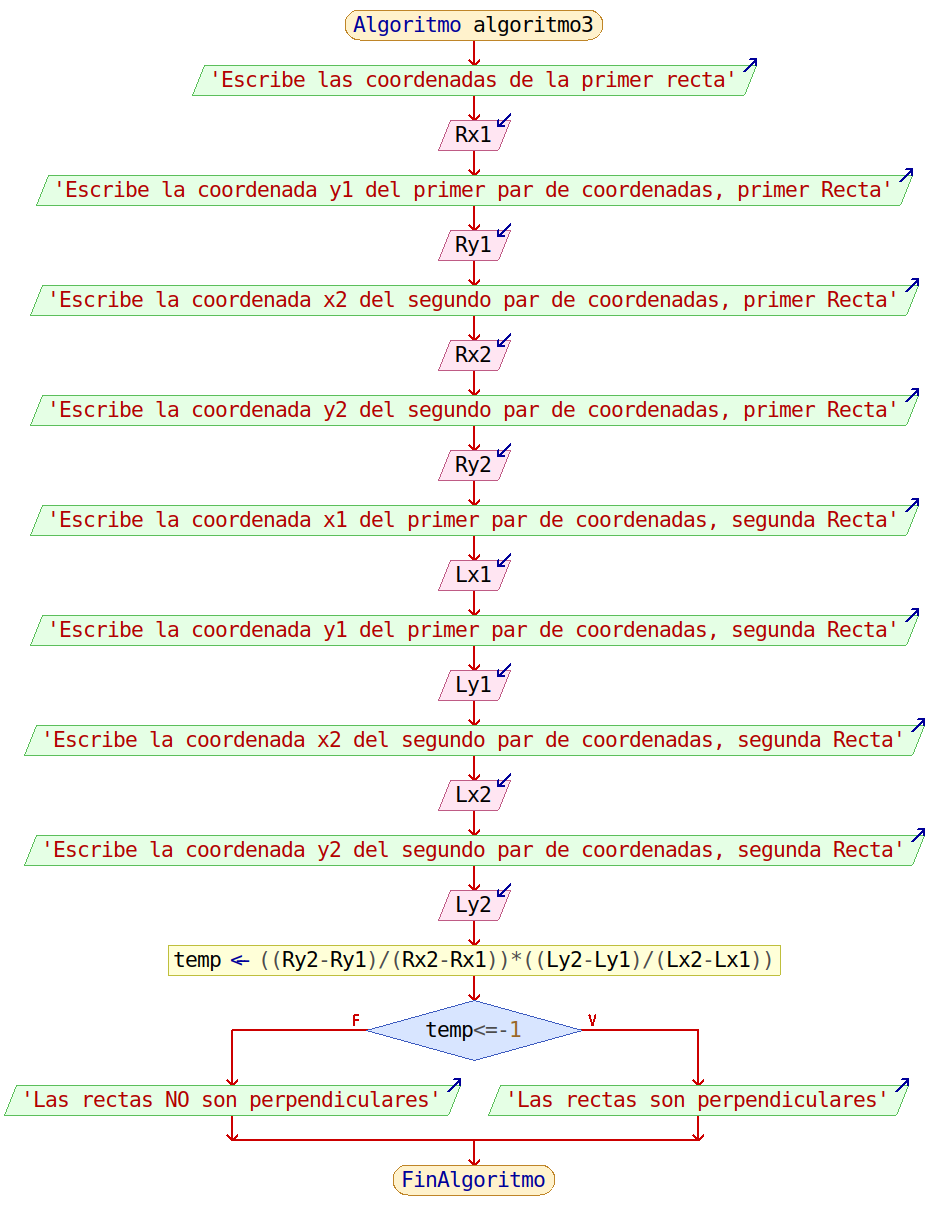
Diseño y funcionamiento de la solución

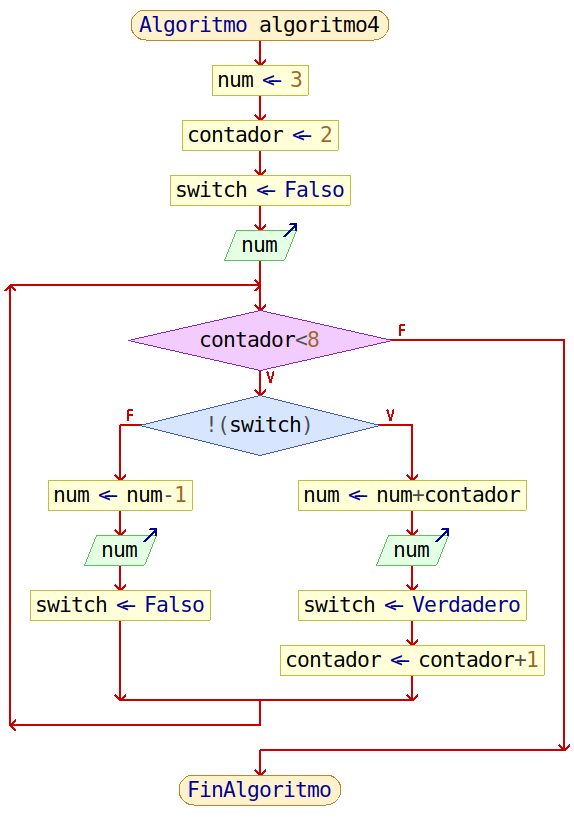


1.-

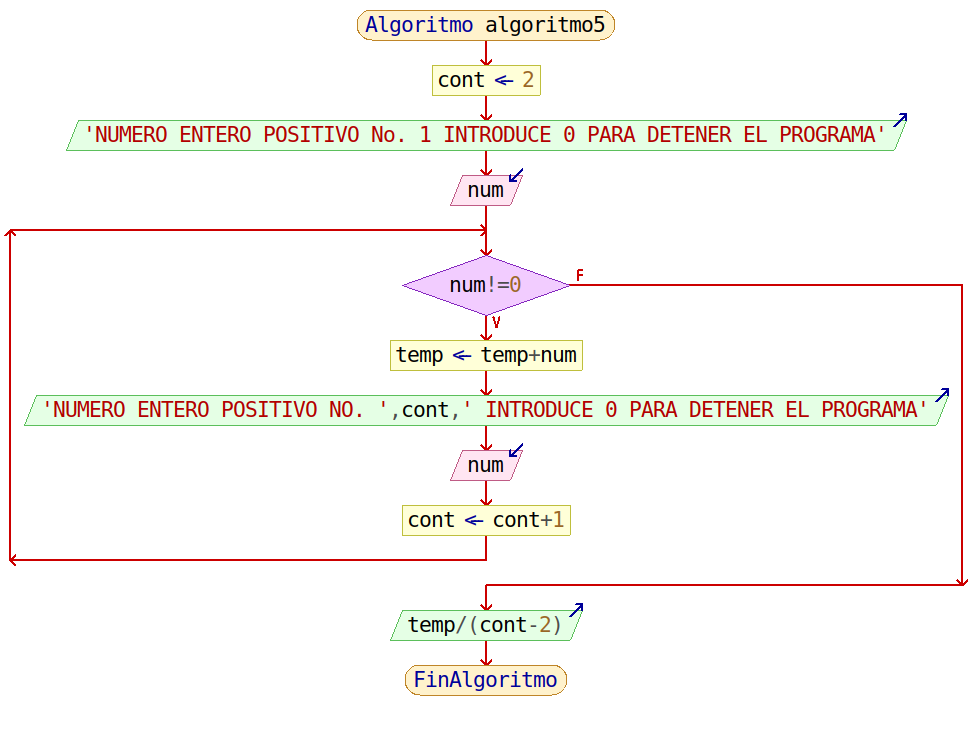


2.-

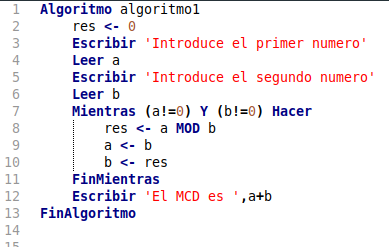
3.-



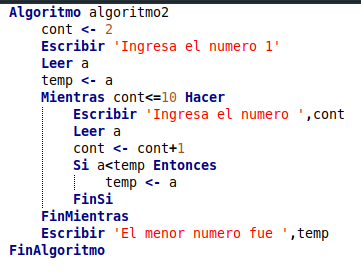
4.-

5.-

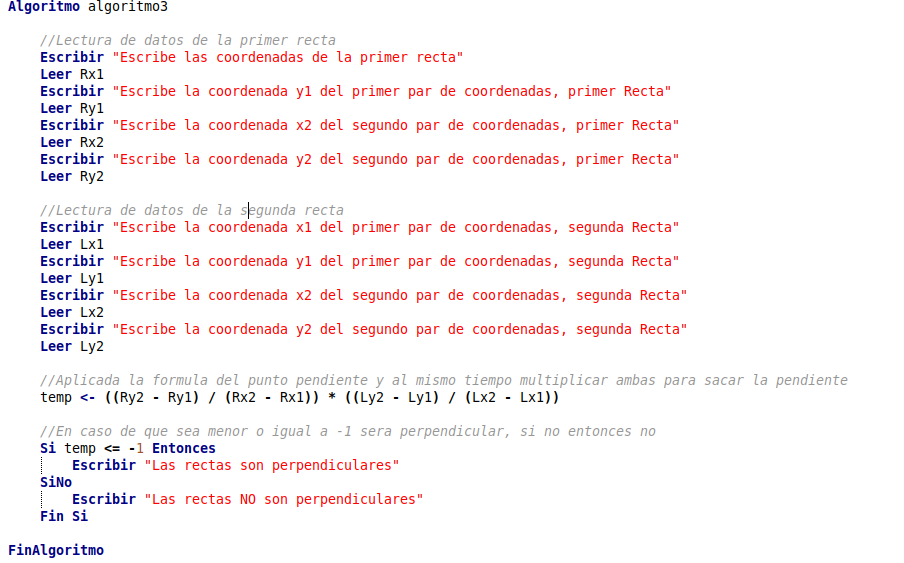
Implementación de la Solución

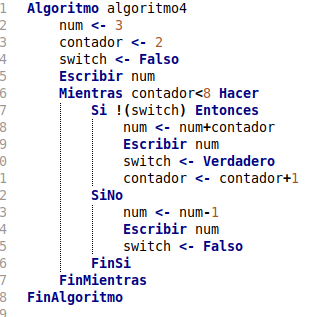


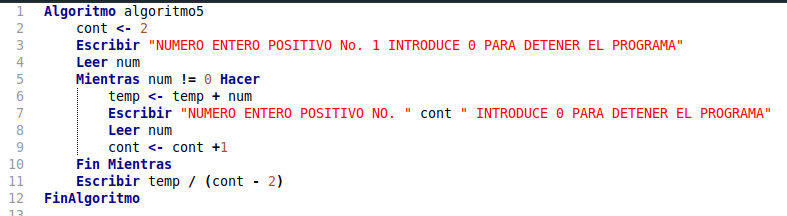
1.-



2.-

3.-

4.-

5.-

Funcionamiento

Algoritmo 1.-

* Prueba de Escritorio

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **res** | **A + B (MCD)** |
| 90 | 0 | 0 | - |
| 90 | 0 | 0 | 90 |

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **res** | **A + B (MCD)** |
| 270 | 192 | 0 | - |
| 192 | 78 | 78 | - |
| 78 | 36 | 36 | - |
| 36 | 6 | 6 | - |
| 6 | 0 | 0 | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **res** | **A + B (MCD)** |
| 48 | 60 | 0 | - |
| 60 | 48 | 48 | - |
| 12 | 0 | 0 | 12 |

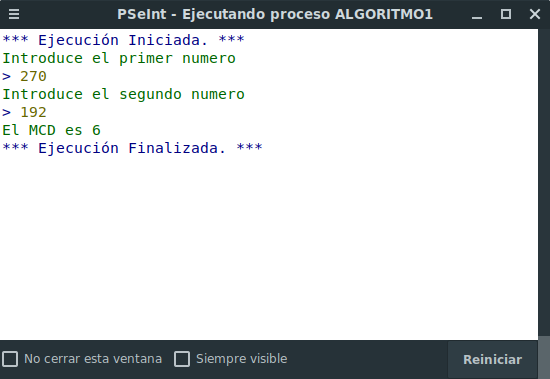
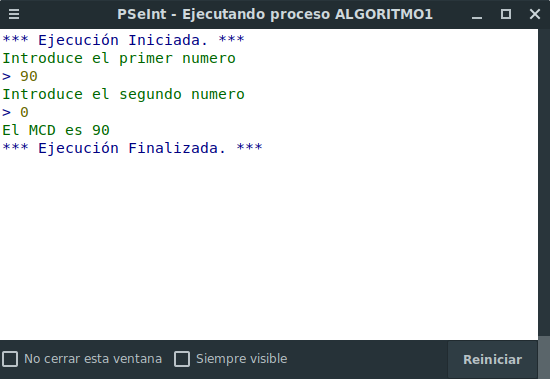
* Resultados de Salida

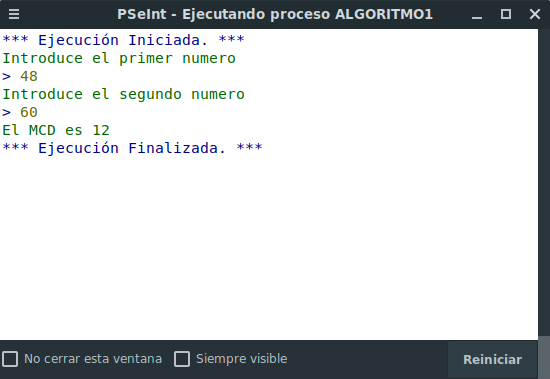
|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **PROCESO** | **SALIDA** |
| a <- 90  b <- 0  res <- 0 | Mientras (a != 0) && (b != 0) Hacer  res <- a % b  a <- b  b <- res | 90 |

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **PROCESO** | **SALIDA** |
| a <- 270  b <- 192  res <- 0 | Mientras (a != 0) && (b != 0) Hacer  res <- a % b  a <- b  b <- res | 6 |

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **PROCESO** | **SALIDA** |
| a <- 48  b <- 60  res <- 0 | Mientras (a != 0) && (b != 0) Hacer  res <- a % b  a <- b  b <- res | 12 |

* Captura de Pantalla





Algoritmo 2.-

* Prueba de Escritorio

|  |  |  |
| --- | --- | --- |
| **cont** | **a** | **temp** |
| 2 | -5 | -5 |
| 2 | 0 | -5 |
| 3 | 90 | -5 |
| 4 | -40 | -40 |
| 5 | 49 | -40 |
| 6 | 89 | -40 |
| 7 | 4 | -40 |
| 8 | 9 | -40 |
| 9 | 9 | -40 |
| 10 | 1 | -40 |

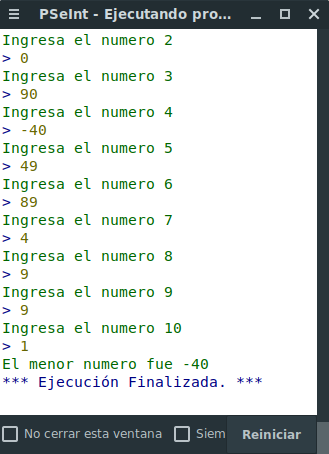
|  |  |  |
| --- | --- | --- |
| **cont** | **a** | **temp** |
| 2 | 4 | 4 |
| 2 | 2 | 2 |
| 3 | 0 | 0 |
| 4 | -40 | -40 |
| 5 | 100 | -40 |
| 6 | 2 | -40 |
| 7 | -432 | -432 |
| 8 | 9 | -432 |
| 9 | 9 | -432 |
| 10 | 1 | -432 |
|  |  |  |

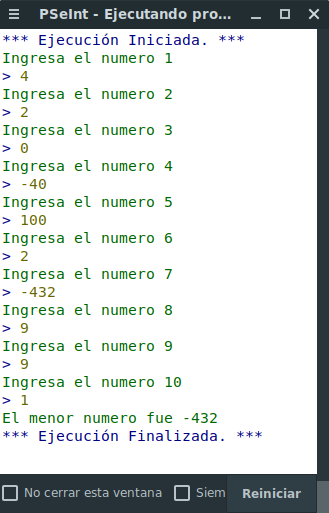
* Resultado de Salida

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **PROCESO** | **SALIDA** |
| a <- (-5)  a <- 0  a <- 90  a <- (-40)  a <- 49  a <- 89  a <- 4  a <- 9  a <- 9  a <- 1 | temp <- a  Mientras cont <= 10 Hacer  Si a < temp Entonces  a <- temp  cont ++  Sino  cont ++ | temp |

|  |  |  |
| --- | --- | --- |
| **ENTRADA** | **PROCESO** | **SALIDA** |
| a <- 4  a <- 2  a <- 0  a <- (-40)  a <- 100  a <- 2  a <- (-432)  a <- 9  a <- 9  a <- 1 | temp <- a  Mientras cont <= 10 Hacer  Si a < temp Entonces  a <- temp  cont ++  Sino  cont ++ | temp |

* Captura de pantalla





Algoritmo 3.-

* Prueba de Escritorio

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Rx1** | **Ry1** | **Rx2** | **Ry2** | **Lx1** | **Ly1** | **Lx2** | **Ly2** | **temp** |
| 3 | 5 | 7 | -1 | 0 | 0 | 12 | 8 | - |
| 3 | 5 | 7 | -1 | 0 | 0 | 12 | 8 | -1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Rx1** | **Ry1** | **Rx2** | **Ry2** | **Lx1** | **Ly1** | **Lx2** | **Ly2** | **temp** |
| 3 | 5 | 7 | -1 | 5 | 5 | 12 | 8 | - |
| 3 | 5 | 7 | -1 | 5 | 5 | 12 | 8 | -0.6 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Rx1** | **Ry1** | **Rx2** | **Ry2** | **Lx1** | **Ly1** | **Lx2** | **Ly2** | **temp** |
| 4 | 4 | 7 | -1 | 5 | 5 | 12 | 8 | - |
| 4 | 4 | 7 | -1 | 5 | 5 | 12 | 8 | -0.71 |

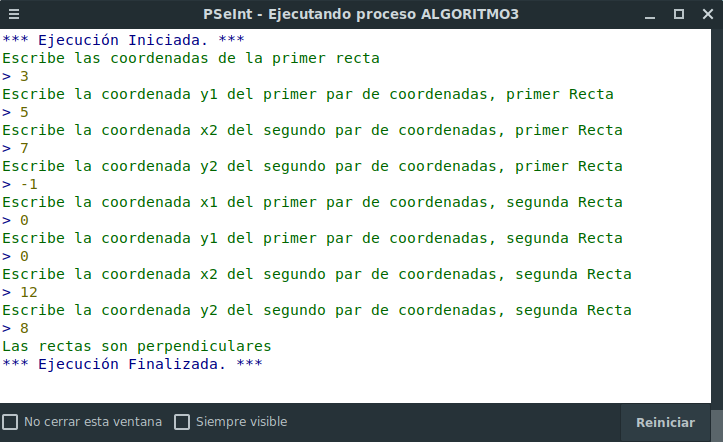
* Resultados de Salida

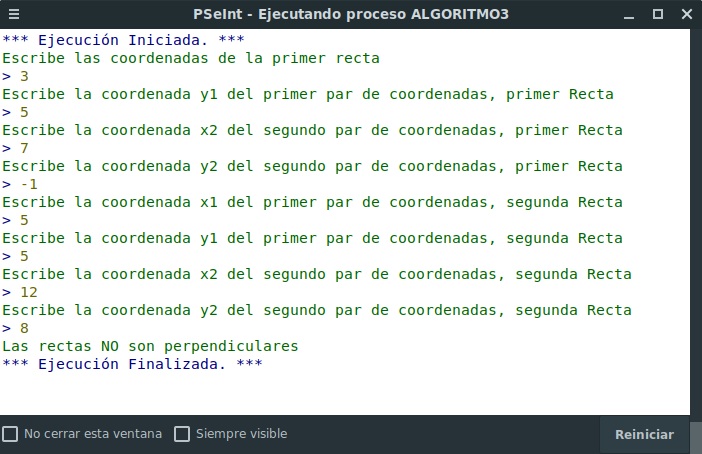
|  |  |  |
| --- | --- | --- |
| **Entrada** | **Proceso** | **Salida** |
| (3,5),(7,-1)  (0,0),(12,8) | temp <- ((Ry2-Ry1)/(Rx2-Rx1))\*((Ly2-Ly1)/(Lx2-Lx1))  Si temp <= -0.9 && temp >= -1.9 Entonces  Escribir 'Las rectas son perpendiculares'  SiNo  Escribir 'Las rectas NO son perpendiculares'  FinSi | Las Rectas son perpendiculares |

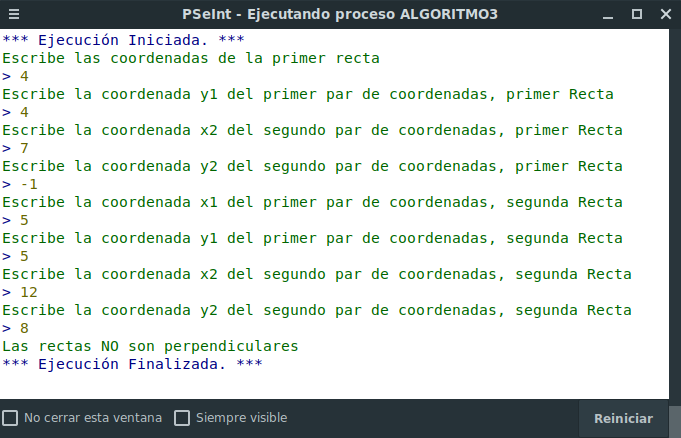
|  |  |  |
| --- | --- | --- |
| **Entrada** | **Proceso** | **Salida** |
| (3,5),(7,-1)  (5,5),(12,8) | temp <- ((Ry2-Ry1)/(Rx2-Rx1))\*((Ly2-Ly1)/(Lx2-Lx1))  Si temp <= -0.9 && temp >= -1.9 Entonces  Escribir 'Las rectas son perpendiculares'  SiNo  Escribir 'Las rectas NO son perpendiculares'  FinSi | Las Rectas NO son perpendiculares |

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Proceso** | **Salida** |
| (4,4),(7,-1)  (0,0),(12,8) | temp <- ((Ry2-Ry1)/(Rx2-Rx1))\*((Ly2-Ly1)/(Lx2-Lx1))  Si temp <= -0.9 && temp >= -1.9 Entonces  Escribir 'Las rectas son perpendiculares'  SiNo  Escribir 'Las rectas NO son perpendiculares'  FinSi | Las Rectas NO son perpendiculares |

* Capturas de Pantalla







Algoritmo 4.-

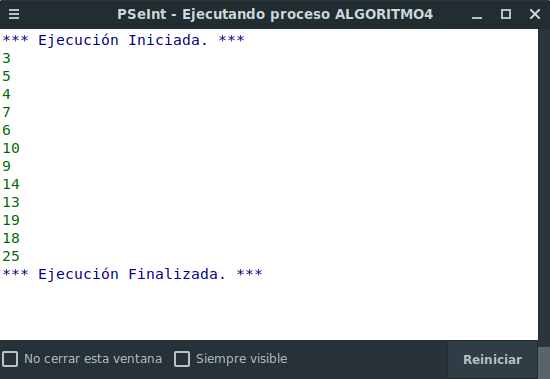
* Prueba de Escritorio

|  |  |  |
| --- | --- | --- |
| **num** | **contador** | **switch** |
| 3 | 2 | F |
| 5 | 3 | V |
| 4 | 3 | F |
| 7 | 4 | V |
| 6 | 4 | F |
| 10 | 5 | V |
| 9 | 5 | F |
| 14 | 6 | V |
| 13 | 6 | F |
| 19 | 7 | V |
| 18 | 7 | F |
| 25 | 8 | V |

* Resultado de Salida

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Proceso** | **Salida** |
| n/a | Escribir num  Mientras contador<8 Hacer  Si !(switch) Entonces  num <- num+contador  Escribir num  switch <- Verdadero  contador <- contador+1  SiNo  num <- num-1  Escribir num  switch <- Falso  FinSi  FinMientras | 3  5  4  7  6  10  9  14  13  19  18  25 |

* Captura de Pantalla



Algoritmo 5.-

* Prueba de Escritorio

|  |  |  |
| --- | --- | --- |
| **num** | **temp** | **cont** |
|  |  | 2 |
| 8 | 8 | 2 |
| 6 | 14 | 3 |
| 7 | 21 | 4 |
| 0 | 7 | - |

|  |  |  |
| --- | --- | --- |
| **num** | **temp** | **cont** |
|  |  | 2 |
| 5 | 3 | 2 |
| 10 | 15 | 3 |
| 2 | 17 | 4 |
| 0 | 5.66 | - |

|  |  |  |
| --- | --- | --- |
| **num** | **temp** | **cont** |
|  |  | 2 |
| 4 | 4 | 2 |
| 4 | 8 | 3 |
| 10 | 18 | 4 |
| 0 | 6 | - |

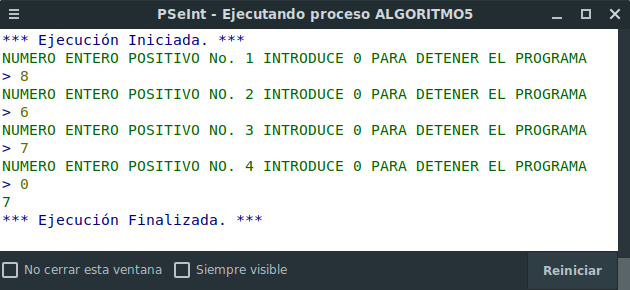
* Resultados de Salida

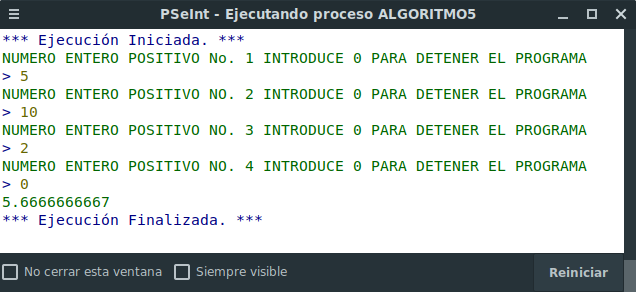
|  |  |  |
| --- | --- | --- |
| **Entrada** | **Proceso** | **Salida** |
| 8, 6, 7, 0 | cont <- 2  Escribir "NUMERO ENTERO POSITIVO No. 1 INTRODUCE 0 PARA DETENER EL PROGRAMA"  Leer num  Mientras num != 0 Hacer  temp <- temp + num  Escribir "NUMERO ENTERO POSITIVO NO. " cont " INTRODUCE 0 PARA DETENER EL PROGRAMA"  Leer num  cont <- cont +1  Fin Mientras  Escribir temp / (cont - 2) | 7 |

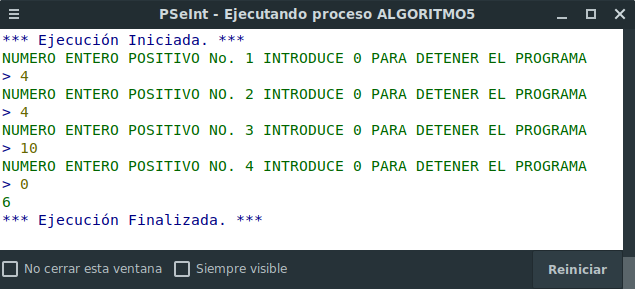
|  |  |  |
| --- | --- | --- |
| **Entrada** | **Proceso** | **Salida** |
| 5, 10, 2, 0 | cont <- 2  Escribir "NUMERO ENTERO POSITIVO No. 1 INTRODUCE 0 PARA DETENER EL PROGRAMA"  Leer num  Mientras num != 0 Hacer  temp <- temp + num  Escribir "NUMERO ENTERO POSITIVO NO. " cont " INTRODUCE 0 PARA DETENER EL PROGRAMA"  Leer num  cont <- cont +1  Fin Mientras  Escribir temp / (cont - 2) | 4 |

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Proceso** | **Salida** |
| 4, 4, 10, 0 | cont <- 2  Escribir "NUMERO ENTERO POSITIVO No. 1 INTRODUCE 0 PARA DETENER EL PROGRAMA"  Leer num  Mientras num != 0 Hacer  temp <- temp + num  Escribir "NUMERO ENTERO POSITIVO NO. " cont " INTRODUCE 0 PARA DETENER EL PROGRAMA"  Leer num  cont <- cont +1  Fin Mientras  Escribir temp / (cont - 2) | 6 |

* Captura de Pantalla







Conclusiones

1.- El problema plantea resolver un Máximo Común Divisor, para ello se recurrió primero a buscar más información sobre este en el sitio web de Wikipedia[[1]](#footnote-0).

Tras revisar la información, se encontró el algoritmo de Euclides, lo cual llevó a realizar otra investigación en la web hasta encontrar un artículo que explica a detalle su uso, el artículo de Khan Academy[[2]](#footnote-1).

El algoritmo de Euclides ayudo bastante, ya que se considera óptimo sin tanto desborde de recursos.

Finalmente con la experiencia programando en otros lenguajes de programación, se implementó el algoritmo como pseudocódigo dentro del programa PSeInt.

Se encontraron dificultades a lo largo del desarrollo por la sintaxis un tanto limitante del pseudocódigo, además de un posible error lógico en el operador AND (&) y el operador OR (||), PSeINT teóricamente está. invirtiendo la lógica de estos dos operadores puesto que el AND realiza la acción del OR y viceversa.

2.- El problema plantea escribir 10 números uno por uno, y finalmente determinar el menor del conjunto ingresado. Ya que no se pueden usar arreglos para acomular los valores y despues realizar un algoritmo de búsqueda lineal, se optó por usar 2 variables, una que se rellene con el nuevo número en curso, y el otro que se le asigne al inicio del programa esta misma variable para después mediante una iteración ir comparando con la primer variable, y en caso de ser menor que rellene la segunda variable.

3.- Se intentó optimizar el código al realizar un ciclo que lea las variables de las coordenadas, pero no se encontró ninguna manera, asi que finalmente se tuvo que pedir variable por variable y finalmente en una sola variable temp realizar las operaciones

4.- Se observa de primera instancia que no hay datos de entrada, así que se procede directamente al proceso, se encontró que hay 2 secuencias dentro de la serie, una que resta -1 siempre y otra que suma un número cada ocasión que se adiciona al número a imprimir, se decidió hacer un if que vaya intercalando entre restar -1 y usar el contador de n+1 usando un booleano para cambiar 1 y 1 hasta terminar los 7 espacios faltantes de la serie.

5.- Se decidió usar la misma técnica usada en el algoritmo 2, usar 2 variables una para ir leyendo los datos del usuario y en otra acomular los datos e irlos sumando, esto hasta que el usuario detenga con un 0, se reciclo una variable (contador textual) aprovechando también el contador que determina entre qué debe dividirse

Anexos

* Todos los códigos pueden mejorarse implementando validaciones
* El algoritmo 3 en específico puede reducir su repetición de lectura de variables usando una iteración, pero por falta de tiempo no se logró reducir el código.
* Los códigos fuente están documentados, pero las capturas no, a excepción del código 3.
* Solo se consultó información para el primer algoritmo, la bibliografía de esta información se encuentra en la sección de conclusiones como pie de página.
* Como dato curioso, el algoritmo 2 primero se desarrollo en C++, y este no funcionó, pero copiando el algoritmo tal cual pero traducido a pseudocodigo en PSeInt, este funciono sin problemas.

1. <https://es.wikipedia.org/wiki/M%C3%A1ximo_com%C3%BAn_divisor> [↑](#footnote-ref-0)
2. <https://es.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/the-euclidean-algorithm> [↑](#footnote-ref-1)