|  |  |  |
| --- | --- | --- |
| Imagen que contiene dibujo, taza  Descripción generada automáticamente | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ingeniería Karina García Morales |
| *Asignatura:* | Fundamentos de Programación |
| *Grupo:* | 1120 |
| *No de Práctica:* | Solución de problemas y Algoritmos.  *3º* |
| *Integrante:* | Yushua Haza Jibaja |
| *No. de Equipo de cómputo empleado:* |  |
| *No. de Lista o Brigada:* | 22 |
| *Semestre:* | 2022 - 1 |
| *Fecha de entrega:* | 28 de septiembre de 2021 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Objetivo:**

El alumno elaborará algoritmos correctos y eficientes en la solución de problemas

siguiendo las etapas de Análisis y Diseño pertenecientes al Ciclo de vida del software.

**Actividades:**

• A partir del enunciado de un problema, identificar el conjunto de entrada y el

conjunto de salida.

• Elaborar un algoritmo que resuelva un problema determinado (dado por el

profesor), identificando los módulos de entrada, de procesamiento y de salida.

**Introducción**

Un problema informático se puede definir como el conjunto de instancias al cual

corresponde un conjunto de soluciones, junto con una relación que asocia para cada

instancia del problema un subconjunto de soluciones (posiblemente vacío).

Para poder solucionar un problema nos apoyamos en la Ingeniería de Software que de

acuerdo con la IEEE se define como “La aplicación de un enfoque sistemático,

disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del

software". Por lo que el uso y establecimiento de principios de ingeniería sólidos, son

básicos para obtener un software que sea económicamente fiable y funcione

eficientemente.

La Ingeniería de Software provee métodos que indican cómo generar software. Estos

métodos abarcan una amplia gama de tareas:

• Planeación y estimación del proyecto.

• Análisis de requerimientos del sistema y software.

• Diseño de la estructura de datos, la arquitectura del programa y el procedimiento

algorítmico.

• Codificación.

• Pruebas y mantenimiento (validación y verificación).

**Ciclo de vida del software**

“El ciclo de vida del desarrollo del software (también conocido como SDLC o Systems Development Life Cycle) contempla las fases necesarias para validar el desarrollo del software y así garantizar que este cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo, asegurándose de que los métodos usados son apropiados.

Su origen radica en que es muy costoso rectificar los posibles errores que se detectan tarde en la fase de implementación. Utilizando metodologías apropiadas, se podría detectar a tiempo para que los programadores puedan centrarse en la calidad del software, cumpliendo los plazos y los costes asociados.

Aunque existen diferentes ciclos de desarrollo de software, la normativa ISO/IEC/IEEE 12207:2017 establece:

“Un marco común para los procesos del ciclo de vida de los programas informáticos, con una terminología bien definida, a la que pueda remitirse la industria del software. Contiene procesos, actividades y tareas aplicables durante la adquisición, el suministro, el desarrollo, el funcionamiento, el mantenimiento o la eliminación de sistemas, productos y servicios informáticos. Estos procesos del ciclo de vida se llevan a cabo mediante la participación de los interesados, con el objetivo final de lograr la satisfacción del cliente”.” (Intelequia news, 2020)

El ciclo de vida básico de un software consta de los siguientes procedimientos:

Definición de objetivos: define la finalidad del proyecto y su papel en la estrategia global.

Análisis de los requisitos y su viabilidad: recopila, examina y formula los requisitos del cliente y examina cualquier restricción que se pueda aplicar.

Diseño general: requisitos generales de la arquitectura de la aplicación.

Diseño en detalle: definición precisa de cada subconjunto de la aplicación.

Programación (programación e implementación): implementación de un lenguaje de programación para crear las funciones definidas durante la etapa de diseño.

Prueba de unidad: prueba individual de cada subconjunto de la aplicación para garantizar que se implementaron de acuerdo con las especificaciones.

Integración: garantiza que los diferentes módulos se integren con la aplicación. Este es el propósito de la prueba de integración que está cuidadosamente documentada.

Prueba beta (o validación): garantiza que el software cumple con las especificaciones originales.

Documentación: sirve para documentar información necesaria para los usuarios del software y para desarrollos futuros.

Implementación

Mantenimiento: comprende todos los procedimientos correctivos (mantenimiento correctivo) y las actualizaciones secundarias del software (mantenimiento continuo).

El orden y la presencia de cada uno de estos procedimientos en el ciclo de vida de una aplicación dependen del tipo de modelo de ciclo de vida acordado entre el cliente y el equipo de desarrolladores.

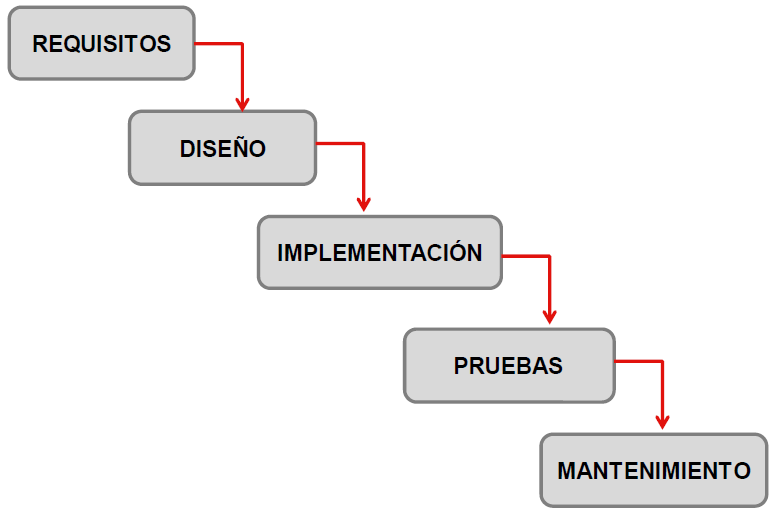


Figura 1 Ciclo de vida del software

**Solución de problemas**

En el ciclo de vida del software, el paso más esencial para poder elaborar el código es el análisis del problema a resolver, en este paso se deben comprender todos los aspectos del problema a solucionar

Se definen las necesidades y requerimientos del programa o algoritmo, para ello tenemos que comprender que tendremos 2 conjuntos de dentro del programa u algoritmo, el conjunto de Entrada de datos y el conjunto de Salida de datos.

El conjunto de Entrada de datos son todos los datos que el usuario puede proporcionar para que a través de la manipulación del programa se generen resultados en el conjunto de Salida de datos

El conjunto de Salida de datos son todos los datos resultantes del programa y son variables dependientes de los datos del conjunto de Entrada de datos

La unión de estos 2 conjuntos se conoce como dominio del problema, es decir, los valores que el problema puede manejar.

**Algoritmos**

“La palabra algoritmo proviene del sobrenombre de un matemático árabe del siglo IX, Al-Khwarizmi, que fue reconocido por enunciar paso a paso las reglas para las operaciones matemáticas básicas con decimales (suma, resta, multiplicación y división)…

Un algoritmo informático es una secuencia de instrucciones finitas que llevan a cabo una serie de procesos para dar respuesta a determinados problemas. Es decir, un algoritmo informático resuelve cualquier problema a través de unas instrucciones y reglas concisas, mostrando el resultado obtenido.

Los algoritmos son muy importantes en el mundo de la informática ya que permiten al programador resolver el problema antes de escribirlo en un lenguaje de programación que entienda la máquina u ordenador. Antes de escribir el código de un programa hay que resolver con un algoritmo el problema que se nos plantea.” (Carlos López Jurado, 2021)

La creación del algoritmo se encuentra en la etapa del diseño del ciclo de vida del software, es durante este paso que se deben abstraer todas las posibles soluciones o metodologías de solución para el problema en cuestión, tomando en cuenta que la mejor solución es la más simple y eficiente en resolver dicho problema

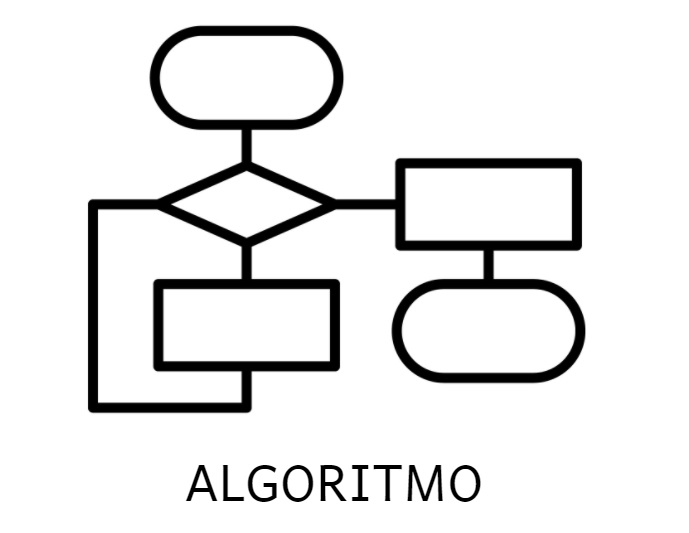


Figura 2 “Descripción visual de un algoritmo”

Las principales características con las que debe cumplir un algoritmo son:

• Preciso: Debe indicar el orden de realización de paso y no puede tener

ambigüedad.

• Definido: Si se sigue dos veces o más se obtiene el mismo resultado.

• Finito: Tiene fin, es decir tiene un número determinado de pasos.

• Correcto: Cumplir con el objetivo.

• Debe tener al menos una salida y ésta debe de ser perceptible

• Debe ser sencillo y legible

• Eficiente: Realizarlo en el menor tiempo posible

• Eficaz: Que produzca el efecto esperado

**Módulos del algoritmo**

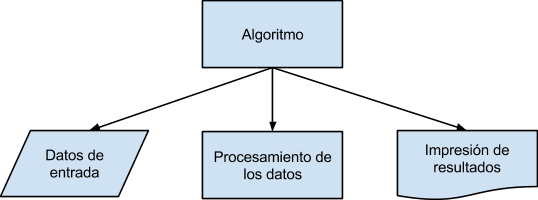


Figura 3 “Módulos del algoritmo informático”

**Variables**

En informática y computación, se defina a variable como aquel dato que puede ser varios valores numéricos, pero solamente un valor simultaneo, se usan estas

“Una variable es donde se guarda (y se recupera) datos que se utilizan en un programa. Cuando escribimos código, las variables se utilizan para:

Guardar datos y estados.

Asignar valores de una variable a otra.

Representar valores dentro de una expresión matemática.

Mostrar valores por pantalla.

Todas las variables deben ser de un tipo de datos, ya sea un dato de tipo primitivo, como un número o texto, o un dato abstracto, como un objeto que se ha creado.

Así que básicamente podemos decir que una variable es

Tipo de dato → identificador variable → valor almacenado.” (Lenguaje de programación.net, 2020)

**Actividades**

Analizamos los problemas presentados en la práctica, el primer problema presentado fue el siguiente:

1-Determinar si un número dado es positivo o negativo

Análisis: Para poder definir una solución primero se deben encontrar las restricciones o requerimientos del solicitador del programa, en este caso la única restricción presente es que no podemos evaluar el valor 0 puesto que simboliza vacío, una vez definida la restricción podemos predecir nuestros datos de salida, los cuales pueden obtener un valor de positivo o negativo sea el caso del valor ingresado.

Numero positivo o negativo

Conjunto de Números Reales

Programa o algoritmo

Datos de entrada:

Restricciones: El valor a designar “N” no puede adoptar el valor de 0

Datos de salida: Determinación de “N” como negativo o positivo

Algoritmo propuesto:

1-Inicio

2-Solicitar valor para N

3-Si N=0, ir al paso previo, sino continuar

4-Si N<0, imprimir “”N” es negativo”

5-SI N>0, imprimir “”N” es positivo”

6-Fin

Prueba de escritorio

|  |  |  |
| --- | --- | --- |
| Iteración | N | Datos de salida |
| 1 | 66 | Positivo |
| 2 | -66 | Negativo |
| 3 | -5 | Negativo |
| 4 | 7 | Positivo |

2-Obtener el mayor de 2 números diferentes dados

Análisis: Para poder definir una solución primero se deben encontrar las restricciones o requerimientos del solicitador del programa, en este caso la única restricción presente es que ambos números dados deben tener valores distintos, no pueden ser iguales, una vez definida la restricción podemos predecir nuestros dato de salida que será la definición ordenada de cual número es mayor que el otro

Conjunto de Números reales

Programa o algoritmo

Num 1:

Num 2:

Rest: Números ≠

Dato de entrada: Núm. Real

Restricción: Los núms. Reales de entrada no pueden ser iguales

Datos de salida: Impresión del núm. Real más grande

Dominio: Todos los núms. Reales (R)

Algoritmo

1-Inicio

2-Solicitar 2 números reales

3-Si ambos números son iguales, regresar al primer paso

4-Si el número 1 es mayor que el segundo, imprimir “Número 1” es mayor, ir al paso final o fin del algoritmo

5-Si el número 2 es mayor que el primero, imprimir “Número 2” es mayor

6-Final

Prueba de escritorio

|  |  |  |  |
| --- | --- | --- | --- |
| Iteración | Número 1 | Número 2 | Dato de salida |
| 1 | 30 | 19 | 30 es mayor |
| 2 | 9 | 9 | Valor invalido |
| 3 | -34 | -75 | -34 es mayor |

3-Obtener el factorial de un número dado. El factorial de un número está dado por el producto de ese numero por cada uno de los números anteriores hasta llegar a 1. El factorial de 0 (0!) es 1:

N! = n\* (n-1)!

Análisis: Para poder definir una solución primero se deben encontrar las restricciones o requerimientos del solicitador del programa, en este caso la única restricción presente es que “n” no puede adoptar valor negativo, una vez definida la restricción podemos predecir nuestros dato de salida que será el factorial del “n” ingresado.

Conjunto de Números naturales

Programa o algoritmo

Conjunto de números naturales mayores o igual a 0

Dato de entrada: Núm. Entero

Restricción: N no puede adoptar valores negativos (-)

Datos de salida: Impresión del factorial de N

Algoritmo

1-Inicio

2-Solicitar valor para N (entero)

3-Si el valor de N es menor a 0, regresar al paso anterior

4.Si el valor de N es mayor a 0, definir como variable entera contador

5-Si la var contadora es menor que 0 pero igual al núm de entrada, entonces

5.1- Multiplicar el valor de var contadora con el valor de N. El resultado se guarda en la var factorial

5.2- Incrementar el valor de var contadora en una unidad (1)

5.3 Regresar al paso 4

6-Si var contadora no incumple en ser menor o igual al valor de N, se imprime el resultado almacenado en la var factorial.

7-Final

Pruebas de escritorio

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteración | N | Factorial | Contadora | Dato de salida |
| 1 | 0 | 1 | 2 | Factorial de 0 es 2 |
| 2 | 4 | 24 |  | Factorial de 4 es 24 |
| 3 | 7 |  |  | Factorial de 7 es |
| 4 | -1 | 1 | 2 | Invalido |

4-Solicitar un valor al usuario y seguirlo solicitando hasta que este sea múltiplo de 5

**Restricciones:** El número debe pertenecer al conjunto de los números reales

**Datos de entrada:** El conjunto de datos de entrada “Ent” este en el conjunto de números reales

Ent pertenece a Reales donde Num pertenece a R (-infinito, infinito)

**Datos de salida:** EL conjunto de salida (SA), esta conformado por, todos los números reales que son múltiplos de 5

Conjunto de Números naturales

Conjunto de números naturales mayores o igual a 0

Programa o algoritmo

Problema: Como encontrar Novi@

Restricciones: Que la persona no comparta el sentimiento de enamoramiento, que la persona se encuentre con una pareja o varias, que la persona simplemente no acepte por alguna circunstancia o característica

Datos de entrada: datos de la persona (complexión física, personalidad, gustos, motivaciones, ideología, situación emocional, etc..)

Datos de salida: Relación saludable

Programa o algoritmo

Relación amorosa saludable

Datos o características deseadas en la otra persona

**Tarea**

**Describe las salidas de los ejercicios finales y la comparación entre ambos algoritmos., realicen una propuesta para un óptimo algoritmo.**

**Problema:** Seguir el algoritmo para obtener una figura

Datos de entrada: Hoja tamaño carta, regla y pluma

Datos de salida: Figura

Algoritmo

1-Dibuja una V rotada a 180 grados empezando del lado izquierdo al derecho

2-En el último vértice donde finaliza la V rotada, dibujar una línea en un ángulo aproximado de poco más de 45 grados, de modo que corte a la mitad la primera línea de la V rotada y que tenga aproximadamente el mismo largo

3-dibujar una línea paralela en perspectiva del dibujante de modo que esta parezca paralela al contorno inferior de la hoja, de aproximadamente el largo de la arista de la V rotada

4-Unir el vértice final de la línea paralela con el vértice inicial de la V rotada con una línea recta

5-Fin, has dibujado correctamente una estrella de 5 picos

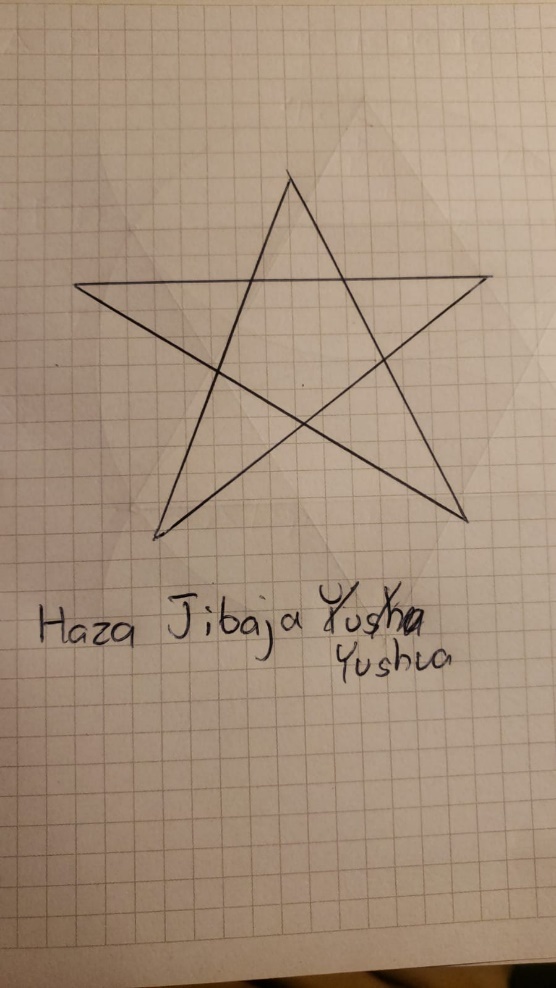


Figura “Algoritmo para estrella de 5 picos”

**Problema:** seguir el algoritmo para obtener una figura

Datos de entrada: Hoja carta, regla y pluma

Datos de salida: Figura

Algoritmo:

1-Dibuja una circunferencia con un compás

2-Traza un vértice en el extremo inferior y otro en el extremo superior de la circunferencia

3-Pon el origen de tu compás en uno de los vértices en los extremos de la circunferencia

4-dibuja una “media circunferencia” dentro de la primera circunferencia

5-Repite el paso anterior pero ahora con el otro vértice extremo

6-En los puntos donde intersecan las medias circunferencias con la circunferencia principal traza vértices

7-Traza un triángulo cuyos vértices sean el vértice superior y los 2 vértices más inferiores a la circunferencia

8-Repite el proceso anterior pero ahora cambiando los vértices inferiores por los superiores y el vértice extremo superior por el inferior, los triángulos no pueden compartir vértices

9-Fin

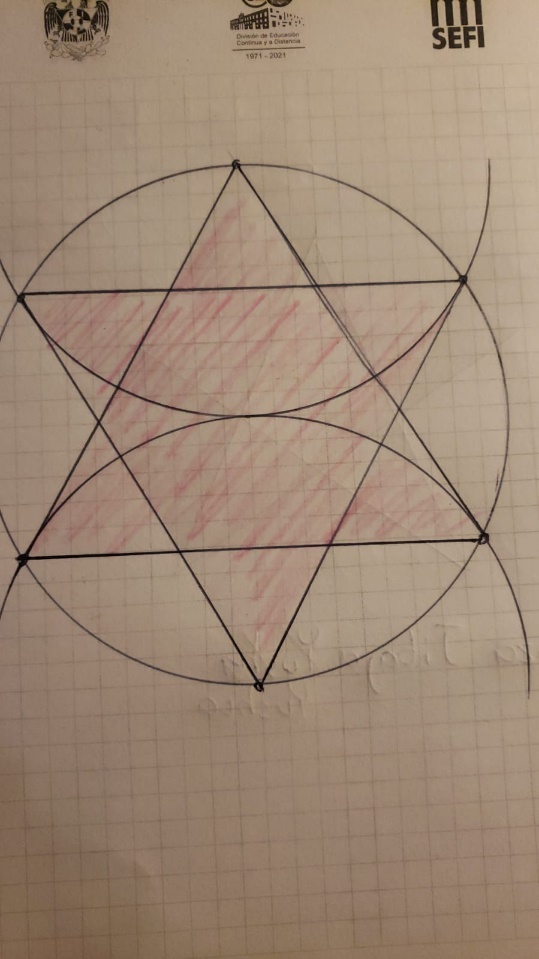


Figura “Algoritmo para estrella de 6 picos”

**Problema:** Dibuja un círculo

Datos de entrada: Herramientas compas, pluma, hoja carta, regla de medir

Restricción: tamaño de la hoja para poder dibujar un círculo entero

Dato de salida: figura círculo

Algoritmo

1-Inicio

2-Abrir el compas de modo que las puntas tengan la distancia acorde al valor del radio del círculo deseado que no sobrepase el límite de tamaño de dibujo de la hoja

3-Colocar la hoja sobre una superficie plana, limpia y seca

4-Colocar la punta de origen del compás en algún punto de la hoja donde se pueda trazar el circulo sin salirse del área del círculo

5-Girar el compas de modo que se trace un círculo

6-Soltar el compas de la hoja, si se desea se puede colorear el interior del área del círculo-circunferencia

7-Fin

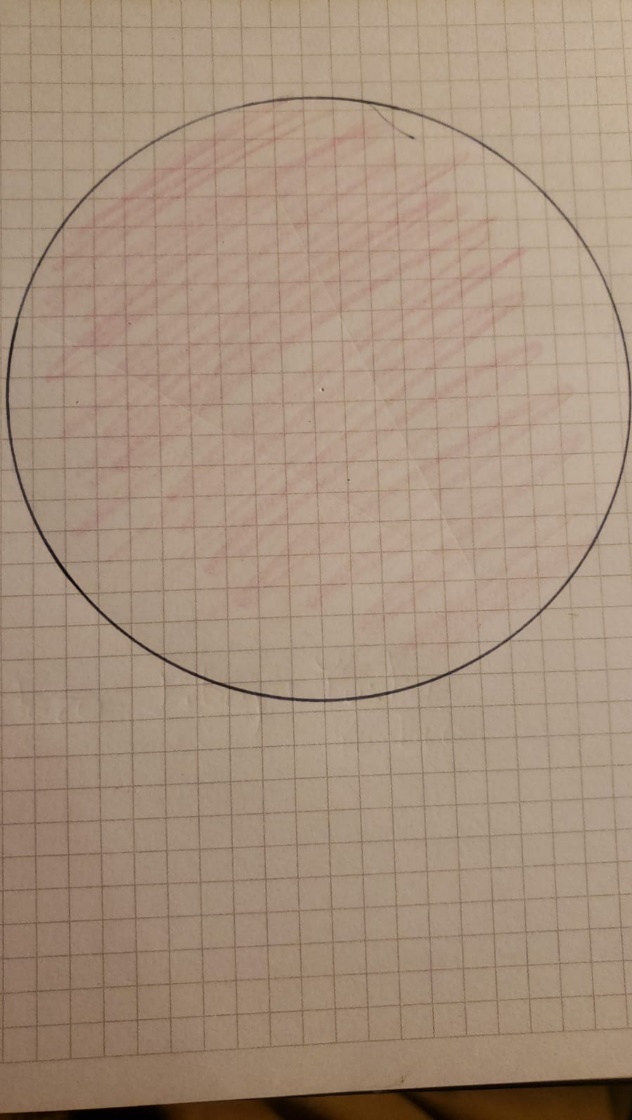


Figura “Algoritmo para círculo”

**2. ¿Qué se necesita para dar solución a un problema?**

1-Definir el problema de la manera más concreta y clara posible

2-Abstraer sus restricciones

3-Elaborar posibles soluciones a ese problema ya sea con posibles soluciones de problemas semejantes al del caso o abstrayendo una solución concreta que dé el resultado solución deseada

4-Evaluar cual solución es más eficiente para solucionar el problema en cuestión

5-Obtener Valores de salida que sean acordes a los resultados esperados, por ejemplo, si a un algoritmo se le solicita obtener los primeros 23 números impares, se espera que la solución sean 23 valores con terminaciones 1,3,5,7,9 que corresponden a números impares.

**3. Describe las fases del ciclo de vida del desarrollo de software explicadas en clase e ilustra**

**Requisitos:** son las características o resultados deseados que deben salir en el conjunto de Salida de datos después de pasar por el programa o algoritmo

Esta etapa requiere se cierto conocimiento para poder entender la idea que el solicitante propone, además de que regularmente debes tomar nota con cada uno de los puntos importantes, de este modo puedes hacer una planificación al momento y llegar incluso a determinar los tiempos de desarrollo que te llevará, antes de proceder a entregar el producto final. El solicitante puede imaginar el producto final de una forma tan abstracta, que se necesita poner los pies en el suelo para obtener resultados que se acerquen más a la realidad.

**Análisis:** Este paso es posiblemente el más crucial para la elaboración del programa o algoritmo, pues se depende de un entendimiento completo de los requisitos y del problema mismo para poder abstraer la solución a dicho problema, puesto que si no se entiende completamente el problema o no

Una vez que el sistema se va desarrollando, es importante para el ciclo de vida del desarrollo del software, que se realicen ciertas pruebas conforme se vaya avanzando. La idea es que no se termine el desarrollo para poder hacer pruebas, si no que mucho antes, durante el proceso de creación, estas ya se puedan ir ejecutando. Las pruebas nos van a permitir ver si el sistema que se está desarrollando es funcional, si tiene algunos errores, si le faltan ciertas cosas para funcionar correctamente, pues básicamente para avanzar al siguiente punto del ciclo de desarrollo de software, será necesario haber pasado las pruebas correctamente.

Mantenimiento: Este es un paso fundamental para mantener funcionando al software de manera óptima, se refiere a la acción de ir evaluando el desempeño del software con las experiencias y retroalimentaciones dadas por los usuarios que usan el software,

1-Si aparece un fallo que no se pudo probar en la etapa de prueba, se pueda solucionar a través de un parche o modificación al software

2-Ingresar nuevas características, funcionalidades o elementos al softare

**Pruebas:** Una vez desarrollado el programa o algoritmo, este se debe probar haciendo uso de este, definiendo los valores de las variables y comprobando externamente del programa si es que este último se está ejecutando correctamente o si está proporcionando los datos de salida deseados y correctos

**4. Analizar las siguientes problemáticas, desglosando y dando solución al problema identificando claramente su conjunto de entrada (datos de entrada), el conjunto de salida (datos de salida) y restricciones, si es que tiene.**

a) En una playa el estacionamiento cobra $ 2.5 por hora o fracción. Determinar cuánto debe pagar un cliente por el estacionamiento de su vehículo, conociendo el tiempo de estacionamiento en horas y minutos.

Datos de entrada: se cobra 2.5 la hora o fracción, una hora es 60 minutos, se convierten 60 minutos a =1 hora

Restricción: No se puede modificar el valor de las horas (60mn=1hora) y no se pueden ingresar cantidades de minutos negativas (aunque estas no existan por naturalidad)

Datos de salida: Total de cobro por estacionamiento

Algoritmo:

1-Inicio

2-Preguntar al usuario la cantidad de minutos que usó el servicio de estacionamiento

3-Leer cantidad de minutos

4-Si la cantidad de minutos es menor que 60, imprimir “total de cobro es 2.5”-fin

5-Si la cantidad de minutos es mayor a 60, entonces se divide la cantidad de minutos entre 60, no importa que la cifra tenga un sobrante indivisible entre 60, puesto que esos minutos se redondearán al valor de una hora

6-Leer como horas

6-Si cantidad horas<1, entonces multiplicar cantidad horas por 2.5

8-Imprimir resultado del paso previo como “Total de cobro es “cantidad horas \* 2.5

9-Fin

Pruebas de escritorio

|  |  |  |  |
| --- | --- | --- | --- |
| Iteración | Cantidad horas | Operación de cobro | Total de cobro |
| 1 | 3 | 2.5\*3 | 7.5 |
| 2 | 4 | 2.5\*4 | 10 |
| 3 | 6 | 2.5\*6 | 15 |
| 4 | 42mn | 2.5\*1 | 2.5 |

b) Dados dos números realizar suma, resta, producto y división (considerar el dividendo ≠ 0).

Datos de entrada: Los números ingresados deben ser enteros

Restricciones: Los números ingresados no pueden adoptar el valor de 0, esto porque si el valor del divisor en la división es 0 se crea una indeterminación o valor invalido

Dato de salida: Resultados de suma, resta, división y multiplicación

Algoritmo:

1-Inicio

2-Solicitar un número (para simplificación se les denominará N)

3-Leer N

4-Solicitar un número (para simplificación se les denominará M) este valor al ser el segundo y el que será el divisor, no puede adoptar el valor de 0

5-Leer M

6-Suma N+M

7-Imprimir el valor de la suma de N+M

7-Resta N-M

9-Imprimir el valor de la resta de N-M

8-Multiplicación N\*M

9-Imprimir el valor de la multiplicación de N\*M

12-División N/M

13-Imprimir el valor de la división de N/M

14-FIN

Pruebas de escritorio

Suma

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteración | Núm. 1 | Núm. 2 | Operación | Dato de salida |
| 1 | 3 | 4 | (3+4) | 7 |
| 2 | 66 | 44 | (66+44) | 110 |

Resta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteración | Núm. 1 | Núm. 2 | Operación | Dato de salida |
| 1 | 3 | 4 | (14-7) | 7 |
| 2 | 66 | 44 | (66-44) | 22 |

Multiplicación

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteración | Núm. 1 | Núm. 2 | Operación | Dato de salida |
| 1 | 3 | 4 | (3\*4) | 12 |
| 2 | 66 | 44 | (66+44) | 2904 |

División

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteración | Núm. 1 | Núm. 2 | Operación | Dato de salida |
| 1 | 3 | 4 | (3/4) | Invalido en enteros, en racional es el valor 0.75 |
| 2 | 66 | 44 | (66/44) | 1 en entero, en irracional es 1.5 |

c)Dados tres números; si el primero es negativo, deberá realizar el producto de los tres, y en caso contrario realizará la suma.

Datos de entrada: 3 núms. enteros

Restricciones: Los números deben ser forzosamente enteros

Datos de salida: suma o multiplicación de los 3 números

Algoritmo:

1. Inicio

2. Solicitar valor para N (primer número)

3. Leer N

4. Solicitar valor para M (segundo número)

5. Leer M

6. Solicitar valor para P (tercer número)

7. Leer P

8. Si N es menor a 0 entonces

8.1. Multiplicar (N\* M\* P)

8.2. Leer “el producto es igual a (N\*M\*P)”

8.3. Ir al paso 10

9. Si N no es menor a 0, entonces

9.1. Sumar (N+M+P)

9.2. Leer “la suma es igual a (N+M+P)”

9.3. Ir al paso 10

10. Fin del algoritmo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Iteración | N | M | P | Operación | Dato de salida |
| 1 | 2 | 3 | 4 | (2\*3\*4) | 24 |
| 2 | -1 | 6 | 6 | (-a\*6\*6) | -36 |
| 3 | -4 | 7 | -4 | (-4\*7\*-4) | 112 |
| 4 | -5 | -5 | -5 | (-5\*-5\*-5) | -125 |

# Referencias

Carlos López Jurado. (29 de enero de 2021). *¿Qué es el ciclo de vida del software y cuáles son sus etapas?* Obtenido de https://es.ccm.net/contents/223-ciclo-de-vida-del-software: https://es.ccm.net/contents/223-ciclo-de-vida-del-software

Flores, L. A. (sin fecha). *Algoritmos*. Obtenido de https://www.uaeh.edu.mx/scige/boletin/prepa4/n10/e1.html: https://www.uaeh.edu.mx/scige/boletin/prepa4/n10/e1.html

Intelequia news. (28 de noviembre de 2020). *Ciclo de vida del software: todo lo que necesitas saber*. Obtenido de https://intelequia.com/blog/post/2083/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber: https://intelequia.com/blog/post/2083/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber

Lenguaje de programación.net. (2020). *¿Qué es una variable en programación?* Obtenido de https://lenguajesdeprogramacion.net/diccionario/que-es-una-variable-en-programacion/: https://lenguajesdeprogramacion.net/diccionario/que-es-una-variable-en-programacion/

Marker, G. (2020). *¿Qué es un algoritmo? Caracteristicas y Tipos*. Obtenido de https://www.tecnologia-informatica.com/algoritmo-definicion/: https://www.tecnologia-informatica.com/algoritmo-definicion/

OK HOSTING. (2021). *El Ciclo de Vida del Software*. Obtenido de https://okhosting.com/blog/el-ciclo-de-vida-del-software/: https://okhosting.com/blog/el-ciclo-de-vida-del-software/

Robledano, Á. (18 de junio de 2019). *Qué es un algoritmo informatico*. Obtenido de https://intelequia.com/blog/post/2083/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber: https://intelequia.com/blog/post/2083/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber

Link Github: https://github.com/YushuaHaza/Practica-3.git