Un diagrama de flujo de datos (DFD por sus siglas en español e inglés) es una representación gráfica del "flujo" de datos a través de un sistema de información. Un diagrama de flujo de datos también se puede utilizar para la visualización de procesamiento de datos (diseño estructurado). Es una práctica común para un diseñador dibujar un contexto a nivel de DFD que primero muestra la interacción entre el sistema y las entidades externas. Este contexto a nivel de DFD se "explotó" para mostrar más detalles del sistema que se está modelando.

Principios de Programación (DFD)

Algoritmo. Un algoritmo es un método para resolver un problema mediante una serie de pasos precisos (indicar el orden de realización en cada paso), definidos (si se sigue dos veces, obtiene el resultado cada vez) y finitos (tiene fin; un número determinados de pasos). Proviene de un nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX. Un algoritmo debe producir un resultado en un tiempo finito. Los métodos que utilizan algoritmos se llaman métodos algorítmicos, en oposición a los métodos que implican algún juicio o interpretación que se denominan métodos heurísticos. Los primeros se pueden implementar en la computadora, los segundos no han sido convertidos fácilmente en las computadoras. Ejemplos de algoritmos son: instrucciones para montar una bicicleta, hacer una receta de cocina, obtener el máximo común divisor de dos números, etc. Pueden expresarse mediante fórmulas, diagramas de flujo o N-S y pseudocódigos

EJEMPLO DE ALGORITMO:

A continuación se detallan las instrucciones básicas para poder andar en bicicleta:
1. Asegúrese de tener una bicicleta.
2. Si ha comprobado que la tiene, súbase en ella pasando una de sus piernas por encima del asiento dejándolo justo en medio de ambas bajo su cola.
3. Para el buen uso de la bicicleta es importante que se haya subido de manera tal que el manubrio quede frente a usted y no a sus espaldas.
4. Tómese del manubrio.
5. Identifique el pedal que se halle más cercano al volante y coloque su pie sobre él (es necesario que sea el pie que se encuentre del mismo lado de la bicicleta que dicho pedal).
6. Empuje el pedal hacia adelante y hacia abajo para comenzar a andar.
7. Cuando comience a moverse pose su otro pie sobre el otro pedal y vaya acompañando con sus pies el movimiento circular de los pedales. Es importante que vaya alternando la pierna que empuja de manera tal que, la que realice la fuerza, sea siempre la que este arriba y avanzando.
8. Mantenga el equilibrio.
9. Si bien muchos de los pasos anteriores pueden realizarse colocando la bicicleta en cualquier posición, el uso apropiado de la misma (y el único modo en que se consigue que cumpla su objetivo de avanzar) es manteniéndola con sus ruedas sobre el suelo.
10. Continúe pedaleando mientras desee seguir andando.

Si ha realizado los pasos anteriores con éxito entonces puede sonreír y disfrutar: usted está andando en bicicleta.

Conceptos

Programa. Un programa de computadora o un programa de 'software' (usualmente abreviado como un programa), es una lista de instrucciones paso a paso escritas para una arquitectura de computadora en un lenguaje de computación en particular. Estos bloques de instrucciones, una vez compilados (por lo general), son cargados a memoria para su ejecución.

Programación. Proceso de escribir un programa o software. Cuando un programa se ejecuta se deben de teclear datos de entrada, esta información será procesada por la computadora.

Lenguaje de programación. Es una técnica de comunicación estandarizada para describir las instrucciones a una computadora. Cada lenguaje de programación tiene sus propias reglas de sintaxis y semántica. usadas que definen un programa de computadora. Un lenguaje habilita al programador a especificar de forma precisa que datos son dados a la computadora, como son almacenados y transmitidos estos datos, y que acciones bajo que circunstancias van a ser tomados. Los lenguajes de programación son del tipo de los lenguajes de computadora, excluyendo al pseudocódigo el cual es exclusivamente para la comunicación humana. Es decir, sirven para escribir programas que permitan la comunicación usuario/máquina.

Lenguaje máquina. El código máquina o lenguaje máquina es un sistema de instrucciones y datos directamente entendibles por un CPU de una computadora. Las "palabras" del lenguaje máquina son llamadas instrucciones, las cuales tienen una acción elemental por el CPU. Cada CPU tiene su propio conjunto de instrucciones de lenguaje máquina.

Compilador. Un compilador es un programa de computadoras que traslada el texto escrito en un lenguaje de computación (fuente) en otro lenguaje de computación (destino). La secuencia original por lo general se conoce como código fuente, y la salida es llamada código objeto. La razón más común en que se traslada el código fuente, es en un programa ejecutable el cual es una serie de representaciones binarias de instrucciones máquina

Intérprete. Lenguaje de programación el cual ejecuta los programas desde su código fuente. Aún se tienen algunas buenas características sobre los compiladores.

Pseudocódigo. Es una descripción de un algoritmo de programación en computadora, el cual usa las convenciones estructurales de los lenguajes de programación, pero omite detallar las subrutinas o la sintaxis de un lenguaje específico.

Identificador. Es un nombre que se les da a las diferentes entidades que puede tener un programa, como lo son las variables, tipos, etiquetas, subrutinas, paquetes, etc. Nombrar las entidades hace posible referenciarlos, lo cual es esencial para cualquier tipo de procesamiento. Este concepto es análogo al "nombre", como por ejemplo el que se le da a una variable. Los lenguajes de computación usualmente definen las restricciones de como debe ser un identificador, como por ejemplo que debe de estar compuesto por letras, números y/o guiones bajos.

Variable. Una variable es un símbolo que denota una cantidad o representación simbólica. En los lenguajes de computación, una variable puede ser tomada como un lugar donde se almacena un valor en la memoria de la computadora. Más precisamente, una variable está asociada a un nombre (llamado algunas veces identificador) dentro de una localidad; el valor se almacena como un objeto de datos en una localidad, de tal manera que este objeto puede ser después accesado mediante una variable, tal como se le refiere a alguien mediante su nombre.

Constante. Una constante es una variable inmutable la cual es referenciada a un lugar donde el valor no puede ser cambiado. El valor de una constante se asigna una sola vez, por ejemplo, HorasPorDia=24. La constante puede ser referenciada múltiples veces en un programa. Usar una constante puede simplificar el mantenimiento del código, disminuyendo el cambio de su valor y suplir un significado a esta constante y consolidándola, como por ejemplo al inicio del programa.

Definición De Lenguaje

Lenguaje

Medio de comunicación entre los seres humanos a través de signos orales y escritos que poseen un significado. También podría decirse que es cualquier procedimiento que sirve para comunicarse, representado mediante símbolos y caracteres específicos

Desde El Punto De Vista Informático El Lenguaje Es:

La representación por medio de signos, símbolos y caracteres que existe entre la comunicación de la PC.

Lenguajes De Programación

Son los lenguajes utilizados para escribir programas de computadoras que puedan ser entendidos por ellas.

Los lenguajes de programación se clasifican en tres grandes categorías:

- Máquina
- Bajo nivel (ensamblador) y
- Alto nivel

Lenguaje Máquina

Es el lenguaje propio de la computadora, basado en la lógica binaria, de ceros y unos (00010111). Este lenguaje resulta difícil de utilizar para las personas; ya que el programador debe introducir todos y cada uno de los comandos y datos en forma binaria, y una operación sencilla como comparar el contenido de un registro con los datos situados en una ubicación del chip de memoria puede tener el siguiente formato: 11001010 00010111 11110101 00101011.

La programación en lenguaje máquina es una tarea tan tediosa y consume tanto tiempo que muy raras veces lo que se ahorra en la ejecución del programa justifica los días o semanas que se han necesitado para escribir el mismo.

Lenguaje De Bajo Nivel (Ensamblador)

Como vimos anteriormente la programación en lenguaje máquina es difícil por ello se necesitan lenguajes que faciliten este proceso. Por este motivo han sido diseñados los lenguajes de bajo nivel.

Estos lenguajes dan a cada instrucción un mnemónico, como por ejemplo STORE, ADD o JUMP. Los lenguajes de bajo nivel permiten crear programas muy rápidos, pero que son a menudo difíciles de

aprender. Esta abstracción da como resultado un lenguaje de muy bajo nivel que es específico de cada microprocesador:

El Lenguaje Ensamblador

Al asignar un código mnemotécnico (por lo general de tres letras) a cada comando en lenguaje máquina, es posible escribir y depurar o eliminar los errores lógicos y de datos en los programas escritos en lenguaje ensamblador, empleando para ello sólo una fracción del tiempo necesario para programar en lenguaje máquina.

En el lenguaje ensamblador, cada comando mnemotécnico y sus operadores simbólicos equivalen a una instrucción de máquina. Un programa ensamblador traduce el código fuente, (una lista de códigos de operación mnemotécnicos y de operadores simbólicos), a código objeto (es decir, a lenguaje máquina) y, a continuación ejecuta el programa, todo esto gracias a un intérprete o a un compilador, los cuáles veremos más adelante

Sin embargo, el lenguaje ensamblador puede utilizarse con un solo tipo de chip de CPU o microprocesador, por lo que los programas escritos en un bajo nivel son prácticamente específicos para cada procesador.

Si se quiere ejecutar el programa en otra máquina con otra tecnología, será necesario rescribir el programa desde el principio.

Así que los programadores necesitaban un método abreviado en el que un enunciado simbólico pudiera representar una secuencia de numerosas instrucciones en lenguaje máquina, y un método que permitiera que el mismo programa pudiera ejecutarse en varios tipos de máquinas. Estas necesidades llevaron al desarrollo de lenguajes de alto nivel.

Lenguaje De Alto Nivel

Los llamados lenguajes de alto nivel son los que se emplean con mayor frecuencia como lenguajes de programación, porque permiten expresar los algoritmos de una manera y con un estilo fácilmente reconocible por parte de diversos programadores y usuarios; debido a que están formados por elementos de lenguajes naturales, como el inglés utilizando términos del tipo LIST, PRINT u OPEN como comandos.

En Basic, el lenguaje de alto nivel más conocido, los comandos se introducen desde el teclado, desde un programa residente en la memoria o desde un dispositivo de almacenamiento, y son interceptados por un programa que los traduce a instrucciones en lenguaje máquina.

Asimismo, presentan una ventaja fundamental: la facilidad de poder ser transportados de una máquina a otra sin necesidad de realizar grandes cambios en ellos, por lo que se dice que son independientes de la máquina empleada.

A este grupo pertenecen los lenguajes más conocidos, tales como el APL, FORTRAN, PASCAL, COBOL, LISP, PROLOG, C, ADA, PL/I.

Sin embargo, tanto los lenguajes de alto nivel como los de bajo nivel, no son entendibles directamente por la máquina, sino que necesitan ser traducidos a instrucciones en lenguaje máquina que entiendan las computadoras por lo que es necesario disponer de una interface con el lenguaje máquina para que el programa sea ejecutable. Al respecto existen dos tipos fundamentales de interface, que son:

a) Compiladores

b) Intérpretes

Un compilador es:

Un traductor que facilita la comunicación entre el programador y la máquina, por medio de un proceso de transformación llamado compilación.

De esta manera traduce un programa íntegro a lenguaje máquina antes de su ejecución, por lo cual se ejecutan con tanta rapidez como si hubiesen sido escritos directamente en lenguaje máquina.

El compilador es el más eficaz para la mayor parte de las máquinas, puesto que presenta la ventaja de de que cada una de las sentencias del programa es interpretada y traducida al lenguaje máquina solo una vez.

Un compilador crea una lista de instrucciones de código máquina, el código objeto, basándose en un código fuente.

El código objeto resultante es un programa rápido y listo para funcionar, pero que puede hacer que falle el ordenador si no está bien diseñado.

Un intérprete es:

Es un programa que se traduce línea por línea bajo la misma plataforma.

Es Un traductor pero más lento que los compiladores ya que no producen un código objeto, sino que recorren el código fuente una línea cada vez. Cada línea se traduce a código máquina y se ejecuta. Cuando la línea se lee por segunda vez, como en el caso de los programas en que se reutilizan partes del código, debe compilarse de nuevo. Aunque este proceso es más lento, es menos susceptible de provocar fallos en la computadora

Variables y constantes

Identificar las reglas para la asignación de nombres a variables y constantes.

Los identificadores que representan los nombre de módulos, subprogramas, funciones, tipos, variables y otros elementos, debe ser elegidos apropiadamente para conseguir programas legibles. El objetivo es usar interfaces significativos que ayuden al lector a recordar el propósito de un identificador sin tener que hacer referencia continua a declaraciones o listas externas de variables. Hay que evitar las abreviaturas crípticas

Identificadores largos se deben de utilizar para la mayoría de los objetos significativos de un programa, así como los objetos utilizados en muchas posiciones, tales como, por ejemplo, el nombre de un programa usado frecuentemente. Identificadores más cortos se utilizarán estrictamente para los objetos locales: así i, j, k son útiles para índices de arrays en un bucle, variables, contadores de bucle, etc., y son más expresivos que indice, VariableDeControl, etc.

Los identificadores deben utilizar letras mayúsculas y minúsculas. Cuando un identificador consta de dos o más palabras, cada palabra debe de comenzar con una letra mayúscula. Una excepción son los tipos

de datos definidos por el usuario, que suelen comenzar con una letra minúscula. Así, identificadores idóneos son: SalarioMes, Nombre, MensajeUsuario, MensajesDatosMal.

Usar nombres para nombras objetos de datos, tales como variables, constantes y tipos. Utilizar Salario mejor que APagar o Pagar.

Utilizar verbos para nombrar procedimientos. LeerCaracter, LeerSigCar, CalcularSigMov son procedimientos que realizan estas acciones mejor que SigCar o SigMov (siguiente movimiento).

Utilizar formas del verso "ser" o "estas" para funciones lógicas. SonIguales, EsCero, EsListo y EsVacio se utilizan como variables y funciones lógicas.

Los nombres de los identificadores de objetos deben sugerir el significado del objeto al lector del programa. Ser nemónicas.

Es conveniente, también, definir constantes con nombres y evitar las explícitas siempre que sea posible. Por ejemplo, no utilizar 7 para el dìa de la semana ó 3.141592 para representar el valor de la constante.

Clasificar los operadores. Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales. Una expresión consta de operandos y operadores. Según sea el tipo de objetos que manipulan. El resultado de una expresión aritmética es de tipo numérico, el resultado de la expresión relacional y de una expresión lógica, es de tipo lógico; el resultado de una expresión carácter es de tipo carácter. Las expresiones se clasifican en:

- Aritméticas. Son análogas a las fórmulas matemáticas. Las variables y constantes son numéricas (reales o enteras) y las operaciones son las aritméticas. Los operadores aritméticos son:

+	suma
-	resta
*	multiplicación
/	división
**, ^	exponenciación
div	división entera
mod	módulo (resto)

En la expresión **5 + 3** los valores **5** y **3** se denominan *operandos*. El valor de la expresión 5 + 3 se conoce como *resultado* de la expresión. **div** es la división entera, es decir, 19 **div** 3 da como resultado 6. El operador **mod** representa el resto de la división entera, algunos lenguajes lo representan con el **%** o el \. 15 **mod** 6 es igual a 3.

- Relacionales. Permiten realizar comparaciones de valores de tipo numérico o carácter. Sirven para expresar las condiciones en los algoritmos, los operadores de relación son:

<	menor
>	mayor
=	igual
<=	menor o igual que
>=	mayor o igual que
<>	distinto de (diferente a)

El resultado de la operación será verdadero o falso. Los operadores de relación se pueden aplicar a cualquiera de los cuatro tipos de datos estándar: *enteros, reales, lógicos* y *carácter*. La aplicación a valores numéricos es evidente. Para realizar comparaciones de datos de tipo carácter, se requiere una secuencia de ordenación de los caracteres similar al orden creciente y decreciente. Esta ordenación suele ser alfabética, tanto mayúsculas como minúsculas, y numérica considerándolas de modo independiente.

- Lógicos. Este es el segundo tipo de expresiones también llamado de tipo *booleano* (se denomina así en honor del matemático británica George Boole, que desarrolló el Álgebra lógica de Boole). El valor resultante de la expresión siempre es *verdadero* (true) o *falso* (false). Las expresiones lógicas se forman combinando constantes lógicas, variables lógicas y otras expresiones lógicas, utilizando los *operadores lógicos* y los *operadores relacionales*. Los operadores lógicos son:

and	y
or	o
not	no

La definición de las operaciones **no**, **y** y **o** se resumen en las tablas conocidas como *tablas de la verdad*:

a	**no** a
verdadero	falso
falso	verdadero

a	b	a **y** b
verdadero	verdadero	verdadero
verdadero	falso	falso
falso	verdadero	falso
falso	falso	falso

a	b	a **o** b
verdadero	verdadero	verdadero
verdadero	falso	verdadero
falso	verdadero	verdadero
falso	falso	falso

Solución de problemas en la computadora

Explicar lar siete fases para la solución de problemas en la computadora. El proceso de resolución de un problema en una computadora conduce a las escritura de un programa y a la ejecución en la misma. Aunque el proceso de diseñar un programa es -esencialmente- un proceso creativo, se puede considerar una fase o pasos comunes, que generalmente deben de seguir todos los programadores. :

- **Definición del problema y análisis de los datos**. Esta fase requiere una clara definición, donde se contemple exactamente lo que debe hacer el programa y el resultado o solución deseada. La computadora requiere especificaciones detalladas de entrada y salida. ¿Qué entradas se requieres? (tipo y cantidad)

¿Cuál es la salida deseada? (tipo y cantidad) ¿Qué método produce la salida deseada?.

- **Diseño de la solución**. En el análisis se determina qué hace el programa. En el diseño se determina cómo hace el programa la tarea solicitada. Los métodos más eficaces para el proceso de diseño se basan en el conocido por *divide y vencerás*. Es decir, la resolución de un problema complejo se realiza dividiendo el problema en subproblemas (módulos). Cualquier programa consta de un programa principal, que a su vez puede llamar a otros subprogramas.
- **Herramientas de programación**. Las más comunes son los *diagramas de flujo* (flowchart) y *pseudocódigo*. Los primeros es una representación gráfica de un algoritmo. Los símbolos más utilizados están normalizados; los segundos son donde las instrucciones se escriben en palabras similares al inglés o español, que facilitan tanto la escritura como la lectura de programas. En esencia, el pseudocódigo se puede definir como *un lenguaje de especificaciones de algoritmos*.
- **Codificación**. Es la escritura en un lenguaje de programación de la representación de algoritmos desarrollada en las etapas precedentes. Dado que el diseño de un algoritmo es independiente del lenguaje de programación utilizado para su implementación, el código puede ser escrito con igual facilidad en un leguaje o en otro. Para realizar la conversión del algoritmo en programa, se debe de sustituir las palabras reservadas en español por sus homónimos en inglés, y las operaciones/instrucciones indicadas en lenguaje natural expresarlas en el lenguaje de programación correspondiente. La documentación interna es la que se incluye dentro del código del programa mediante comentarios que ayudan a la comprensión del código.
- **Compilación.** Una vez que el algoritmo es convertido en programa fuente, es preciso introducirlo en memoria mediante el teclado y almacenarlo posteriormente en un disco. Esta operación se realiza con el programa editor, posteriormente el programa fuente se convierte en un *archivo de programa* que se guarda (graba) en disco. El *programa fuente* debe ser traducido a lenguaje máquina, este proceso se realiza con el compilador y el sistema operativo que se encarga prácticamente de la compilación. Si tras la compilación, el programa presenta errores (*errores de compilación*) en el programa fuente, es preciso volver a editar el programa, corregir los errores y compilar de nuevo.
- **Prueba (verificación) y depuración**. La verificación de un programa es el proceso de ejecución
- **Documentación y mantenimiento**.

Identificar los datos de entrada y salida en el planteamiento de un problema, detallando el proceso de solución

Reconocer las características de los algoritmos. *Preciso* (indicar el orden de realización de cada paso), *definido* (si se sigue dos veces, se obtiene el mismo resultado cada vez), y *finito* (tiene fin; tiene número determinado de pasos).

Describir las herramientas de representación algorítmica más comunes

Utilizar la simbología para la representación de diagramas de flujo, diagramas Nassi-Shneiderman y reglas para la elaboración de PseudoCódigo.

Aplicar la jerarquía de los operadores aritméticos, relacionales y lógicos en la resolución de expresiones. Las reglas de prioridad en las operaciones aritméticas son primero los exponenciales, luego la multiplicación y división, despues el **mod** y el **div**, y por último la suma y la resta.

GUIA PARA INVESTIGAR

Distinguir los tipos de datos: simples (enteros, reales, caracteres, lógicos, enumerados y subrango) y estructurados (cadena de caracteres, arreglos y registros)

Realizar un ejemplo de algoritmo como el descrito en el procedimiento de montar bicicleta.

Entregar por correo electrónico a oswaldolag@gmail.com con el asunto GUIA1. CONCEPTOS DFD. Y en el mensaje se debe colocar el nombre y el código del alumno.