

Índice

1. Introducción y Objetivos.....	1
2. Indique cuáles son las fases de resolución de un problema con computadora...	2
3. Define que es algoritmo.....	3
4. DESCRIBE LAS CARACTERISTICAS DE UN ALGORITMO.....	3
5. Describe el diseño descendente (<i>top-down</i>) o modular.....	3
6. Describe Las dos herramientas más utilizadas comúnmente para diseñar algoritmos.....	4
7. ¿Cuáles son los métodos usuales para representar un algoritmo?.....	4
8. Describe que es un Diagramas de flujo.....	4
9. Describe cuales son los Símbolos más utilizados en los diagramas de flujo.....	5
10. Describe que es la Codificación de un programa.....	7
11. Describe que es la <i>Documentación interna</i> (en un programa).....	7
12. Describe que es programa fuente, programa objeto y programa ejecutable.....	7
13. Describe que es la Verificación y depuración de un programa.....	8
14. Describe que es PROGRAMACIÓN MODULAR.....	8
15. Describe que es PROGRAMACIÓN ESTRUCTURADA Y CUAL ES LA DIFERENCIA ENTRE PROGRAMACION ESTRUCTURADA Y PROGRAMACION NO ESTRUCTURADA.....	8,9
16. ¿Qué son las Estructuras de control en la programación estructurada? Describa cada una.....	9
17. Bibliografía.....	10
18. Conclusión.....	11

Introducción

En esta investigación conoceremos punto por punto los elementos y herramientas de la programación, llegando a desarrollar un conocimiento básico pero impórtate para el crecimiento de nuestro entendimiento de la programación por computadora.

Seguiremos cada punto como si estuviéramos subiendo una escalera, comprendiendo que paso a paso cada uno de los escalones se entrelaza.

Esta investigación se desarrollara como un cuestionario de preguntas, para que con esto podamos conocer la pregunta y entender mejor la solución.

Objetivos

El objetivo principal de esta investigación es analizar, desarrollar y comprender los elementos iniciales del mundo de la programación.

**Universidad Interamericana de Panamá
FACULTAD DE INGENIERÍA
PROGRAMACION I
INVESTIGACIÓN 1 – CONCEPTOS BÁSICOS**

Fecha de entrega: JUEVES 1 DE OCTUBRE de 2015

Desarrolle cada punto indicado abajo, sea lo más objetivo posible.

1. Indique cuáles son las fases de resolución de un problema con computadora

R/. La fases de resolución de un problema de un computador serian:

- Análisis del problema: El problema se analiza teniendo presente la especificación de los requisitos dados por el cliente de la empresa o por otra persona que encarga el programa.
- Diseño del algoritmo: una vez analizado el problema, se diseña una solución que conducirá a un algoritmo que resuelva el problema.
- Codificación (implementación): la solución se escribe en la sintaxis del lenguaje de alto nivel (por ejemplo, C) y se obtiene un programa.
- Ejecución, verificación y depuración: el programa se ejecuta, se comprueba rigurosamente y se elimina todos los errores (denominados “bugs”, en inglés) que puedan aparecer.
- Mantenimiento: El programa se actualiza y modifica, cada vez que sea necesario, de modo que se cumplan todas las necesidades de cambio de sus usuarios.
- Documentación: Escritura de las diferentes fases del ciclo de vida del software, esencialmente el análisis, diseño y codificación, unidos a manuales de usuario y referencia, así como normas para el mantenimiento.

Las dos primeras fases conducen a un diseño detallado escrito en forma de algoritmo. Durante la tercera etapa (codificación) se implementa el algoritmo en un código escrito en un lenguaje de programación, reflejando las ideas desarrolladas en las fases de análisis y diseño.

La fase de ejecución y compilación traduce y ejecuta el programa. En las fases de verificación y depuración el programador busca errores de las etapas anteriores y los elimina. Comprobará que mientras más tiempo se gaste en la fase de análisis y diseño, menos se gastara en la depuración del programa. Por último, se debe realizar la documentación del programa.

2. ¿Define que es algoritmo?

R/. Un algoritmo es un método para resolver un problema mediante una serie de pasos precisos, definidos y finitos.

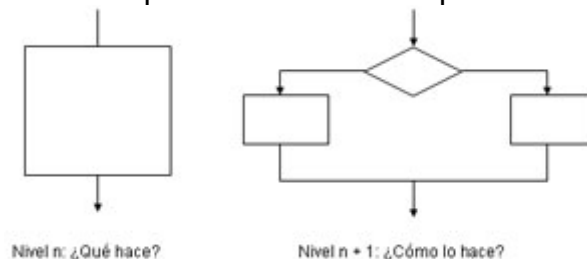
3. DESCRIBE LAS CARACTERISTICAS DE UN ALGORITMO

R/. Las Características de un Algoritmos serian

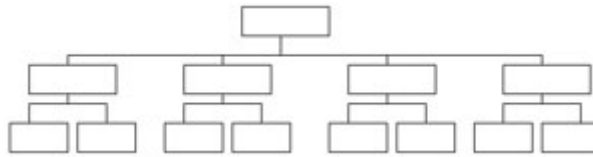
- Finitos: Debe acabar en algún momento.
- Eficientes: Deben ocupar la mínima memoria y minimizar el tiempo de ejecución.
- Legibles: El texto que lo describe debe ser claro, tal que permita entenderlo y leerlo fácilmente.
- Modificables: Estarán diseñados de modo que sus posteriores modificaciones sean fáciles de realizar, incluso por programadores diferentes a sus propios autores.
- Modulares: La filosofía utilizada para su diseño debe favorecer la división del problema en módulos pequeños.
- Único punto de entrada, único punto de salida: A los algoritmos y a los módulos que lo integran se entra por un sólo punto, inicio, y se sale por un sólo punto también, fin.

4. Describe el diseño descendente (top-down) o modular

R/. La programación descendente o "Top - Down" este es el proceso mediante el cual un problema se descompone en una serie de niveles o pasos sucesivos de refinamiento (stepwise). La metodología descendente consiste en efectuar una relación entre las sucesivas etapas de estructuración de modo que exista una relación entre ellas mediante entradas y salidas de información. El problema se descompone en varias estructuras jerárquicas, de forma que se pueda considerar cada estructura desde dos puntos de vista: ¿qué hace? y ¿cómo lo hace? Las estructuras desde los dos puntos de vista se representan de la siguiente forma:



El diseño descendente se representa así:



5. Describe Las dos herramientas más utilizadas comúnmente para diseñar algoritmos.

R/ Los métodos más usuales para representar un algoritmo serian:

1. PSeInt = Pseudocódigo

Es un lenguaje simplificado para describir un algoritmo utilizando una mezcla de frases en lenguaje común, y palabras claves que indican el inicio y el fin del algoritmo y las instrucciones específicas a realizar.


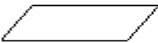
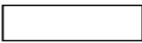
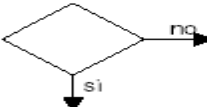
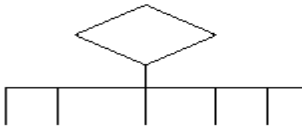



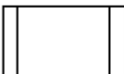
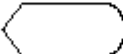
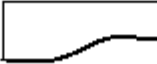
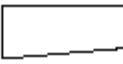

2. RAPTOR = Diagrama de flujo

El diagrama de flujo es la representación gráfica de un algoritmo; para ello se utiliza un conjunto de estándares mundialmente utilizados y desarrollados por tales como ANSI (American National Institute) e ISO (International Standard Organization) para la elaboración de diagramas de flujo

6. Describe que es un Diagramas de flujo

R/. Un diagrama de flujo (flowchart) es una de las técnicas de representación de algoritmo más antigua y a la vez más utilizada, aunque se empleo ha disminuido considerablemente, sobre todo desde la aparición de lenguajes de programación estructurados. Un diagrama de flujo es un diagrama que utiliza los símbolos (cajas) estándar mostrados en la figura 1 y que tiene los pasos del algoritmo escritos en esas cajas unidas por flechas, denominadas líneas de flujo, que indican la secuencia en que se deben ejecutar.

7. Describe cuales son los Símbolos más utilizados en los diagramas de flujo.
R/

Símbolos Principales	Función
	Terminal (representa el comienzo, "inicio" y el final, "fin", de un programa. Puede representar también una parada o interrupción programada que sea necesario realizar en un programa).
	Entrada/Salida (cualquier tipo de introducción de datos en la memoria desde los periféricos "entrada", o registro de la información procesada en un periférico, "salida").
	Proceso (cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transferencia, etc.).
	Decisión (indica operaciones lógicas o de comparación entre datos —normalmente dos— y en función del resultado de la misma determina cual de los distintos caminos alternativos de programa se debe seguir; normalmente tiene dos salidas —respuestas SI o NO—, pero puede tener tres o mas, según los casos).
	Decisión múltiple (en función del resultado de la comparación se seguirá uno de los diferentes caminos de acuerdo con dicho resultado).
	Indicador de dirección o línea de flujo (indica el sentido de ejecución de las operaciones).
	Línea conectora (sirve de unión entre dos símbolos).
	Conector (conexión entre dos puntos del organigrama situado en paginas diferentes).
	Llamada a subrutina o a un proceso predeterminado (una subrutina es un modulo independiente del programa principal, que recibe una entrada procedente de dicho programa, realiza una tarea determinada y regresa, al terminar, al programa principal).
Símbolos Secundarios	Función
	Pantalla (se utiliza en ocasiones, en lugar del símbolo de E/S).
	Impresora (se utiliza en ocasiones en lugar del símbolo DE/S).
	Teclado (se utiliza en ocasiones en lugar del símbolo de E/S).
	Comentarios (se utiliza para añadir comentarios clasificadores a otro símbolos del diagrama de flujo. Se puede dibujar a cualquier lado del símbolo)

9. Describe que es la Codificación de un programa

R/.La Codificación de un programa es la conversión de un algoritmo en programa, utilizando un lenguaje de programación.

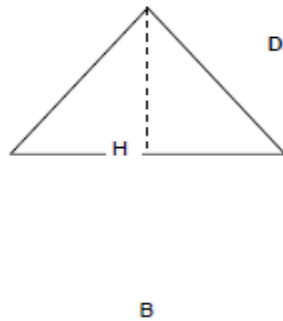
Las instrucciones expresadas en lenguaje natural deben ser expresadas en lenguaje de programación correspondiente.

Si tras la compilación se presenta errores en el programa fuente, es preciso volver a editar el programa, corregir los errores y copilar nuevamente. Este proceso se repite hasta que no se producen errores, obteniéndose el programa objeto que todavía no es ejecutable directamente. Suponiendo que no existe errores en el programa fuente, se debe instruir al sistema operativo para que realice la fase de montaje o enlace (link), es decir, la carga del programa objeto con las librerías del programa de compilador. El proceso de montaje produce un programa ejecutable.

Cuando el programa ejecutable se ha creado, se puede ya ejecutar desde el sistema operativo con solo teclear su nombre, desde sistema operativo DOS, o bien haciendo doble clic, en Windows.

Ejemplo:

Realizar un programa, para calcular el Área de un triángulo e imprimir el resultado.
(Utilizar todas las etapas para resolver un programa)

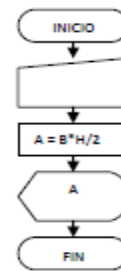


DATOS:

H = Altura

B = Base

DIAGRAMA DE FLUJO:



ANALISIS

Fórmula:

Area = $B * H / 2$

Datos de Entrada:

A, B

Datos de Salida:

H

ALGORITMO:

1. Inicio
2. Leer: A, B
3. Calcular: Area = $B * H / 2$
4. Imprimir Area
5. Fin

CODIFICACIÓN:

```

void main(void)
{
    int b,h;
    float a;

    printf("Ingresar la base: ");
    scanf("%d",&b);
    printf("Ingresar la altura: ");
    scanf("%d",&h);
    a=b*h/2;
    printf("es area es: %f",a);
}
  
```

10. Describe que es la *Documentación interna (en un programa)*

R/. La Documentación Interna en un programa es la documentación que va incluida con el código y estos pueden ser de dos tipos

- Complementaria Esta permite comprender mejor su funcionamiento
- Ayuda Interactivas normalmente destinadas al usuario se puede considerar también de usuario.
- Las ayudas Interactivas sofisticadas se pueden considerar como una función más de software.

11. Describe que es programa fuente, programa objeto y programa ejecutable

R/. El Programa fuente es un lenguaje de alto nivel (texto ordinario que contiene las sentencias del programa en un lenguaje de programación). Necesita ser traducido a código máquina para poder ser ejecutado.

Programa (o código) objeto: Es el programa fuente traducido (por el compilador) a código máquina. Aún no es directamente ejecutable.

Programa Ejecutable: Traducción completa a código máquina, realizada por el enlazador, del programa fuente y que ya es directamente ejecutable.

12. Describe que es la Verificación y depuración de un programa

R/. La verificación de un programa: es el proceso de ejecución del programa con una amplia variedad de datos de entrada, llamados datos de test o prueba, que determinarán si el programa tiene errores("bugs"). Para realizar la verificación se debe desarrollar una amplia gama de datos de test: calores normales de entrada, valores extremos de entrada que comprueben los límites del programa y valores de entrada que comprueben aspectos especiales del programa.

La depuración: es el proceso de encontrar los errores del programa y corregir o eliminar dichos errores

13. Describe que es PROGRAMACIÓN MODULAR

R/. **Una** Programación modular consta de varias secciones divididas de forma que interactúan a través de llamadas a procedimientos, que integran el programa en su totalidad.

En la programación modular, el programa principal coordina las llamadas a los módulos secundarios y pasa los datos necesarios en forma de parámetros.

A su vez cada módulo puede contener sus propios datos y llamar a otros módulos o funciones.

14. Describe que es PROGRAMACIÓN ESTRUCTURADA Y CUAL ES LA DIFERENCIA ENTRE PROGRAMACION ESTRUCTURADA Y PROGRAMACION NO ESTRUCTURADA.

R/. La programación estructurada es una teoría de programación que consiste en construir programas de fácil comprensión, es especialmente útil, cuando se necesitan realizar correcciones o modificaciones después de haber concluido un programa o aplicación. Al utilizar la programación estructurada, es mucho más

sencillo entender la codificación del programa, que se habrá hecho en diferentes secciones.

Se basa en una metodología de desarrollo de programas llamada refinamientos sucesivos: Se plantea una operación como un todo y se divide en segmentos más sencillos o de menor complejidad, una vez terminado todos los segmentos del programa, se procede a unificar las aplicaciones realizadas por el grupo de programadores. Si se ha utilizado adecuadamente la programación estructurada, esta integración debe ser sencilla y no presentar problemas al integrar la misma, y de presentar algún problema, será rápidamente detectable para su corrección.

A diferencia de la no estructurada, no se puede bifurcar el programa. Es decir, sólo puedes ejecutar el programa por secciones. Para realizar una bifurcación, tendrás que recurrir a instrucciones condicionales que ejecutarán una sección del programa sólo si se cumple una determinada condición. Aquí radica la diferencia fundamental entre ambas formas de programación. El lenguaje no estructurado permite la bifurcación desde y hacia cualquier línea del programa. Ejemplos de lenguajes no estructurados: BASIC, FORTRAN, Assembler.

19. ¿Qué son las Estructuras de control en la programación estructurada? Describe cada una.

En las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa.

Con las estructuras de control se puede:

- De acuerdo a una condición, ejecutar un grupo u otro de sentencias (If-Then-Else)
- De acuerdo al valor de una variable, ejecutar un grupo u otro de sentencias (Select-Case)
- Ejecutar un grupo de sentencias mientras se cumpla una condición (Do-While)
- Ejecutar un grupo de sentencias hasta que se cumpla una condición (Do-Until)
- Ejecutar un grupo de sentencias un número determinado de veces (For-Next)

Todas las estructuras de control tienen un único punto de entrada. Las estructuras de control se puede clasificar en:

Secuenciales, iterativas y de control avanzadas.

Esto es una de las cosas que permite que la programación se rija por los principios de la programación estructurada.

1. Ejecución secuencial

Pero por lo general las instrucciones se ejecutan una después de la otra, en el orden en que están escritas, es decir, en secuencia. Este proceso se conoce como ejecución secuencial.

2. Estructuras de control iterativas

Las estructuras de control iterativas o de repetición, inician o repiten un bloque de instrucciones si se cumple una condición o mientras se cumple una condición.

3. Estructuras anidadas

Las estructuras de control básicas pueden anidarse, es decir pueden ponerse una dentro de otra.

Bibliografía

1. <http://www.iqcelaya.itc.mx/~vicente/Programacion/ResProb.pdf>
2. http://ing.unne.edu.ar/pub/informatica/Alg_diag.pdf
3. http://ing.unne.edu.ar/pub/informatica/Alg_diag.pdf
4. <http://www.mailxmail.com/curso-introduccion-lenguaje-pascal/programacion-descendente-top-down>
5. http://www.gayatlaacomulco.com/tutorials/pascal/u1_1_3.html
6. http://www.gayatlaacomulco.com/tutorials/pascal/u1_1_3.html
7. <http://macabremoon0.tripod.com/id6.html>
8. <http://macabremoon0.tripod.com/id6.html>
9. http://www.nebrija.es/~abustind/Informatica/Metodologia/Elementos_basicos_C.pdf
10. http://www.nebrija.es/~abustind/Informatica/Metodologia/Elementos_basicos_C.pdf
11. http://www.nebrija.es/~abustind/Informatica/Metodologia/Elementos_basicos_C.pdf
12. http://www.nebrija.es/~abustind/Informatica/Metodologia/Elementos_basicos_C.pdf
13. <http://teleformacion.edu.aytolacoruna.es/PASCAL/document/modular.htm>
14. <http://www.desarrolloweb.com/articulos/2477.php>
15. https://es.wikipedia.org/wiki/Estructuras_de_control

Conclusión

Conociendo ya los procesos y las herramientas necesarias para la programación denotamos que en principio es esencial manejar un orden importante para la estructuración del programa , puesto que con el manejo de las secuencia de códigos y una buena estructura basa en un diagrama de flujo se pueden llegar a mimetizar los errores del programador y del usuario. Sabiendo que aprendimos el proceso inicial del algoritmo sus formas de trabajar y sus características principales entendimos que es solo una base inicial para el conocimiento inmenso de la programación por ende seguimos avanzando hasta encontrarnos con la codificación de un programa centrado en las bases de lenguaje c y c++ lo que nos explicó que con la ase del algoritmo podemos desarrollar los conceptos estructurados y plasmarlos en la codificación de un lenguaje más utilizado los cuales anteriormente mencione. Luego de todo esta increíble investigación llegamos a la base de la programación estructurada la cual es más que todo el procedimiento evolucionado y más completo del algoritmo utilizando lenguaje c y c++. Por ende concluyó que es importante conocer las bases y afianzarlas para luego tener una estructura maciza y completa el conocimiento es de todos y la alegría de tenerlo es indiscutible.