

MODULO
CURSO ALGORITMOS

IVAN ARTURO LOPEZ ORTIZ
Ivan.lopez@UNAD.edu.co
Ivan.lopezortiz@gmail.com

**UNIVERSIDAD NACIONAL ABIERTA Y A
DISTANCIA – UNAD
FACULTAD DE CIENCIAS BÁSICAS E
INGENIERÍA
PROGRAMA INGENIERIA DE SISTEMAS
BOGOTÁ D.C., 2005**

PROTOCOLO ACADEMICO Y GUÍA DIDÁCTICA
CURSO: ALGORITMOS

@CopyRigth
Universidad Nacional Abierta y a Distancia

ISBN

2005
Centro Nacional de Medios para el Aprendizaje

TABLA DE CONTENIDO

<u>1.-Ficha Tecnica.....</u>	<u>5</u>
<u>Planificación De Las Unidades Didácticas.</u>	<u>6</u>
<u>Introduccion.....</u>	<u>7</u>
<u>1. Primera Unidad</u>	<u>5</u>
GENERALIDADES, CONCEPTOS, ANTECEDENTES Y DESARROLLO DE LOS ALGORITMOS	
.....	5
INTRODUCCIÓN	5
1.1 INTENCIONALIDADES FORMATIVAS:	6
1.2 GENERALIDADES.....	7
1.2.1 LECTURA 1. INTRODUCCIÓN A LA INFORMÁTICA	8
1.3 PROGRAMACIÓN DE COMPUTADORAS.....	18
1.3 PROGRAMACIÓN DE COMPUTADORAS.....	18
1.3.1 LENGUAJES DE PROGRAMACIÓN	18
Lectura # 2: Lenguajes De Programación	19
1.3.2 LA LÓGICA DE LA PROGRAMACIÓN	22
1.3.3 TÉCNICAS DE PROGRAMACIÓN	23
1.4 TIPOS DE DATOS Y OPERADORES	28
1.4.1. VARIABLES Y CONSTANTES	28
1.3.2 EJERCICIOS DE VERIFICACIÓN	31
<u>2. Segunda Unidad:.....</u>	<u>33</u>
ESTRUCTURA GENERAL DE UN ALGORITMO.....	33
INTRODUCCIÓN	33
2.1 INTENCIONALIDADES FORMATIVAS:	33
2.2 DIAGRAMAS DE FLUJO	34
2.2.1 EJERCICIOS DE VERIFICACIÓN.....	40
2.3 ALGORITMOS	42
2.3.1 ESTRUCTURAS DE SELECCIÓN.....	45
2.3.1.1 Ejercicios De Verificación	47
2.3.2 ESTRUCTURAS DE SECUENCIA CICLOS O BUCLES.....	48
2.3.2.1 Ciclo Para O Desde	49
2.3.2.1.1 Ejercicios De Verificación	52
2.3.2.2 Ciclo Mientras.....	54
2.3.2.3 Ciclo Repita Hasta Que	55
2.3.2.4 Ejercicios De Verificación	56
2.3.3 SUBPROGRAMA O MODULO	58

3. Tercera Unidad.....	61
LENGUAJE DE PROGRAMACIÓN C++	61
INTRODUCCIÓN	61
3.2 LENGUAJE DE PROGRAMACIÓN C++	62
3.2.2 EJECUCIÓN DEL PROGRAMA	64
3.2.3 EJERCICIOS DE VERIFICACIÓN	71
3.2.4 C++, CICLOS.....	75
3.2.4.1 Ciclo For	75
3.2.4.1.1 Ejercicios De Verificación	77
3.2.4.2 Ciclos While Y Do While	77
3.2.4.2.1 Ejercicios De Verificación	79
3.1.5.1 Funciones Incorporadas.....	82
3.1.5.2.-Nuestras Propias Funciones	83
Fuentes Documentales	89
Sitios Web	89
Anexos	91
ANEXO 1 – MAPA CONCEPTUAL	92
ANEXO2 GUIA PARA LA CONSTRUCCION DE PORTAFOLIO.....	95
ANEXO 3 FICHA DE SEGUIMIENTO	100
ANEXO 4 FORMATO PARA LA AUTOEVALUACIÓN DEL GRUPO COLABORATIVO	102
ANEXO 5 COMAPAR Y CONTRASTAR	103
ANEXO 7 HABILIDAD DE OBSERVAR	104
ANEXO 8 ÍNDICE DE FUNCIONES	105

1.-FICHA TECNICA

<i>NOMBRE DEL CURSO</i>	ALGORITMOS.
<i>PALABRAS CLAVE</i>	Computadora, Hardware, Software, Informática, Código binario, Periféricos, Memoria, Programación de computadoras, programador, Programación estructurada, Lenguaje de programación, Interprete, Compilador, Variable, Constante, Diagrama de flujo, Diagramas de flujo, Algoritmos, Toma de decisión, Ciclos, Funciones
<i>INSTITUCION</i>	UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA UNAD
<i>CIUDAD</i>	SANTA FE DE BOGOTÁ
<i>AUTOR DEL PROTOCOLO ACADEMICO</i>	IVAN ARTURO LOPEZ ORTIZ ivan.lopez@UNAD.edu.co ivan.lopezortiz@gmail.com
<i>AÑO</i>	2005
<i>UNIDAD ACADEMICA</i>	FACULTAD DE CIENCIAS BASICAS E INGENIERIA
<i>CAMPO DE FORMACION</i>	PROFESIONAL
<i>AREA DE CONOCIMIENTO</i>	INGENIERIA DE SISTEMAS Y AFINES
<i>CREDITOS ACADEMICOS</i>	TRES (3)
<i>TIPO DE CURSO</i>	TEORICO PRÁCTICO
<i>DESTINATARIOS</i>	Estudiantes de diversos programas de la UNAD,
<i>COMPETENCIA GENERAL DE APRENDIZAJE</i>	El estudiante se inicia en los fundamentos esenciales de las técnicas y destrezas del diseño, análisis y construcción de algoritmos y programas informáticos
<i>METODOLOGIA DE OFERTA</i>	A DISTANCIA
<i>FORMATO DE CIRCULACION</i>	Documentos impresos en papel con apoyo en Web; CD-ROM.
<i>DENOMINACION DE LAS UNIDADES DIDACTICAS</i>	1. Conceptos, antecedentes y desarrollo de algoritmos 2. Estructura general de un algoritmo 3. Herramienta de programación

PLANIFICACIÓN DE LAS UNIDADES DIDÁCTICAS.

Unidades didácticas, capítulos, temas, secciones, fragmentos

Unidades Didácticas	capítulos	temas	secciones
Unidad 1: Conceptos, antecedentes y desarrollo de los algoritmos	<ul style="list-style-type: none"> Generalidades 	<ul style="list-style-type: none"> Características 	Introducción a la informática
	<ul style="list-style-type: none"> Programación DE COMPUTADORAS 	<ul style="list-style-type: none"> Técnicas Programación modular Programación estructurada 	
	<ul style="list-style-type: none"> Lenguajes de programación 	<ul style="list-style-type: none"> Lenguajes de máquina, bajo y alto nivel 	
		<ul style="list-style-type: none"> Traductores 	<ul style="list-style-type: none"> Interpretes compiladores
	<ul style="list-style-type: none"> tipos de datos y operaciones 	<ul style="list-style-type: none"> Númericos Alfanuméricos Lógicos 	Enteros Reales Carácter Booleano
	<ul style="list-style-type: none"> Variables y constantes 	<ul style="list-style-type: none"> concepto identificadores Manipulación 	
	<ul style="list-style-type: none"> Expresiones 	<ul style="list-style-type: none"> Aritméticas Lógicas(booleanas) Operadores lógicos 	<ul style="list-style-type: none"> Reglas de prioridad
Unidad 2: Estructura General de un Algoritmo	<ul style="list-style-type: none"> Conceptos generales 	Diagramas de flujo Algoritmos	Simbología Conceptualización
	<ul style="list-style-type: none"> Sentencias en Algoritmos 	<ul style="list-style-type: none"> De asignación De lectura(entrada) De escritura(salida) Decisión 	
	<ul style="list-style-type: none"> Elementos básicos 	<ul style="list-style-type: none"> Bucles (ciclos) 	<ul style="list-style-type: none"> Desde (para) Mientras que Repetir - hasta que
		<ul style="list-style-type: none"> Contadores Acumuladores 	
	<ul style="list-style-type: none"> Subprogramas 	<ul style="list-style-type: none"> Subalgoritmos Variables locales y globales Comunicación 	
Unidad 3: Herramienta de programación	<ul style="list-style-type: none"> Introducción al lenguaje de programación C o C++ 	<ul style="list-style-type: none"> Conceptos Generales Instalación Estructura de un programa Estructuras de control Funciones 	

INTRODUCCION

El curso de Algoritmos, esta adscrito a la facultad de Ciencias Básicas e Ingeniería de la UNAD y corresponde al programa de Ingeniería de Sistemas, esta constituido por tres créditos académicos, correspondientes a 38 de acompañamiento y 106 de estudio independiente, de acuerdo al contenido programático establecido por la facultad de ciencias básicas e ingeniería, esta dirigido inicialmente a alumnos de la UNAD de segundo semestre o periodo académico, sin que esto implique que lo puedan tomar otros participantes deseados de adquirir conocimientos en el arte de la programación de computadoras. Este curso corresponde a la formación básica del programa y no requiere que el participante posea conocimientos iniciales para el desarrollo de los temas planteados; el temario pretende que los participantes adquieran y apliquen conocimientos básicos necesarios para la construcción de soluciones informáticas, utilizando para ello diversas estrategias de aprendizaje, propias del modelo de educación a distancia, permitiendo activar las habilidades cognitivas y metacognitivas en el estudiante.

El presente modulo no pretende ser un libro especializado en la construcción de algoritmos; es un material de consulta que pretende llevar al estudiante al aprendizaje de los conceptos básicos necesarios para adquirir conocimientos previos en la programación de computadoras, los cuales le ayudarán a enfrentarse a la solución de supuestos problemicos y de problemáticas reales en su entorno.

Toma la premisa de Luis Joyanes Aguilar¹, pues construye y recopila una cantidad de ejercicios prácticos que brindan la posibilidad de adquirir destreza y habilidad en el abordaje, análisis y resolución de los mismos.

Esta dividido en tres unidades didácticas, que van desde la adquisición de conocimientos previos en el diseño de algoritmos, hasta la solución de los mismos mediante el lenguaje de programación C++.

La primera Unidad comprende, una introducción a hardware, software, técnicas de programación, lenguajes de programación, tipos de datos, operadores, variables, constantes, expresiones (aritméticas, Lógicas, carácter);

La segunda inicia con una conceptualización de los diagramas de flujo, simbología y utilidad, luego se realiza el abordaje de la estructura como tal de los algoritmos, su evolución, sentencias de asignación, entrada, salida de datos, instrucciones de decisión, cada uno de los ciclos empleados en la programación de computadoras, así mismo se trabajara con contadores y acumuladores; finalizaremos la unidad con un vistazo a los subprogramas.

¹ Madrid, 1996 "*Todos los cursos de programación deben apoyarse en la resolución de gran número de problemas que permitan al alumno adquirir práctica que le facilite el aprendizaje.*"

La tercera unidad, procura llevar a la práctica lo realizado con cada algoritmo, en esta unidad se utilizara el lenguaje de programación C++, como herramienta de programación, al igual que en las anteriores unidades, ésta inicia con una fundamentación teórica, continúa con la estructura de un programa, las estructuras de control y finaliza con la practica de funciones.

Cada una de las unidades con sus correspondientes temas y secciones se abordara mediante recopilación de lecturas, complementadas con diferentes talleres para ser abordados en forma individual, grupo colaborativo y gran grupo. Evidenciada permanentemente en las fichas de seguimiento que se llevan en el portafolio.

Es importante destacar que para este curso los estudiantes tengan algunas habilidades de dominio del computador, las cuales se dieron en el curso de herramientas informáticas, al igual se sugiere tomar el curso de mantenimiento y ensamble de computadoras, que aportara grandes referentes para entender la arquitectura básica de un computador.

1. PRIMERA UNIDAD

Generalidades, Conceptos, antecedentes y desarrollo de los algoritmos

Introducción

La primera unidad del curso de algoritmos, esta dirigida esencialmente a la conceptualización de términos básicos necesarios para el abordaje del presente curso.

Entre los aspectos fundamentales se encuentran, las generalidades de la programación, las técnicas de programación, evolución histórica de la programación, lenguajes de programación, traductores (compiladores e interpretes), tipos de datos (numéricas, alfanuméricas, lógicas) y operaciones, manejo de variables y constantes lo mismo que expresiones aritméticas lógicas, todo esto acompañado de procesos pedagógicos, propios del modelo de la educación a distancia apropiada en el uso de las nuevas tecnologías

Esta unidad se trabaja mediante lecturas que permiten la apropiación del conocimiento y se evidencian en diferentes productos que le ofrecen la oportunidad de aplicar las diferentes herramientas, adquiridas en los primeros cursos académicos; también se desarrollarán una serie de ejercicios prácticos, tendentes a adquirir habilidades en la resolución de problemáticas supuestas, para pasar a problemáticas reales.

Igualmente están implícitas diferentes estrategias de pensamiento de orden superior que el estudiante irá descubriendo gracias al apoyo permanente del tutor, quien en es el mediador del proceso de aprendizaje.

Al final de la unidad, se plantean una serie de actividades que buscan determinar el grado de apropiación del conocimiento, además de dar soporte valorativo a la nota definitiva

**

1.1 INTENCIONALIDADES FORMATIVAS:

Propósitos de la unidad

Motivar al estudiante en el abordaje de los temas referentes a la evolución y desarrollo de los algoritmos

Realizar lecturas que permitan conceptualizar lo referente a hardware y software

Objetivos de la unidad

Conceptualizar los aspectos fundamentales referentes a los antecedentes, desarrollo y evolución de los algoritmos.

Determinar las técnicas de programación, lo mismo que los lenguajes de programación

Conocer tipos de operadores

Diferenciar y aplicar variables y constantes

Jerarquizar las expresiones mediante las reglas de prioridad

Conocer diferentes tipos de lenguajes de programación

Competencias de la unidad:

El estudiante domina los conceptos previos necesarios para el desarrollo de algoritmos

Metas de aprendizaje

El estudiante mediante lecturas y acompañamiento tutorial mediado es capaz de comprender los conceptos fundamentales para el desarrollo de un algoritmo.

Unidades Didácticas:

Palabras claves:

Hardware

Software

Informática

Código binario

Periféricos

Memoria

Programación de computadoras

programador

Programación estructurada

Lenguaje de programación

Interprete

Compilador

Variable

Constante

Diagrama de flujo

1.2 GENERALIDADES

Es importante que los estudiantes del curso de algoritmos, adquieran conocimientos básicos en temas referentes a la informática como el funcionamiento básico de los computadores, sus dispositivos hardware, las características esenciales del software y esencialmente se adquiera conocimiento en lenguajes de programación, características, evolución, las tendencias y técnicas de programación y en fin todo aquello que aporte al desarrollo habilidades que permitan un conocimiento inicial necesario para abordar temas primordiales en la programación de computadoras, para lo cual se propone la siguientes lecturas

1.2.1 Lectura 1. Introducción a la Informática²

Es necesario que el alumno determine y diferencie claramente aspectos básicos sobre Informática, hardware, software, para lo cual se sugiere al estudiante se apoye en la lectura número 1, 2 y 3, a partir de ella debe realizar la actividades programadas para estas lecturas.

“

INTRODUCCIÓN A LA INFORMÁTICA

Desde sus orígenes, el hombre ha tenido necesidad de la información. Esta información, que en principio se recogía de forma oral, con el surgimiento de la escritura, comenzó a almacenarse en medios que evolucionaron desde las tablillas hasta el papel, pasando por los papiros y los pergaminos. También los medios de transmisión han ido evolucionando desde la transmisión oral, buena para distancias cortas, hasta la transmisión a grandes distancias por cables utilizando código Morse o la propia voz mediante el teléfono.

Se puede decir que el tratamiento de la información es tan antiguo como el hombre y se ha ido potenciando y haciendo más sofisticado con el transcurso del tiempo hasta llegar a la era de la electrónica. El hombre no ha parado a lo largo de la historia de crear máquinas y métodos para procesar la información. Para facilitar esta tarea, en especial en el mundo actual, donde la cantidad de información que se procesa a diario es ingente, surge la **informática**.

DEFINICIONES BÁSICAS.

El término **Informática** proviene de la unión de las palabras **información** y **automática**. De una forma muy general podemos decir que la informática se ocupa del tratamiento automático de la información. Concretando más, podemos definir Informática como la ciencia o conjunto de conocimientos científicos que permiten el tratamiento automático de la información por medio de ordenadores.

Como se puede observar, en la definición anterior de Informática, intervienen dos palabras

clave:

• Información, y • ordenador.

Por **información** se entiende cualquier conjunto de símbolos que represente hechos, objetos o ideas.

¿Qué es un **ordenador**? Un ordenador o computadora es básicamente una máquina compuesta de una serie de circuitos electrónicos que es capaz de recoger unos datos de entrada, efectuar con ellos ciertos cálculos, operaciones lógicas y operaciones aritméticas y devolver los datos o información resultante por medio de algún medio de salida. Todas estas acciones las realiza la

² Documento tomado de: <http://www.di.ujen.es/asignaturas/fundTopo/TEMA1.pdf>

computadora sin necesidad de intervención humana y por medio de un programa de instrucciones previamente introducido en ella.

Si tenemos en cuenta esta definición de computadora podemos redefinir el concepto de Informática como la ciencia que abarca todos los aspectos del diseño y uso de las computadoras.

El ordenador se diferencia del resto de la máquina con capacidad de tratar información (por ejemplo, una calculadora básica o una máquina de escribir) en lo siguiente:

- Gran velocidad de tratamiento de la información.
- Gran potencia de cálculo aritmético y lógico.
- Capacidad para memorizar los programas y datos necesarios para resolver cualquier problema técnico o de gestión.
- Capacidad de comunicación con las personas y con otras máquinas y dispositivos para recibir o transmitir datos.
- Posibilidad de tratamiento de datos en tiempo real.
- Actúa **sin** intervención de un operador humano y bajo el control de un programa previamente almacenado en la propia computadora.

Desde el punto de vista informático, existen dos tipos de información: Datos e instrucciones.

Los **datos** son conjuntos de símbolos que utilizamos para expresar o representar un valor

Numérico, un hecho, un objeto o una idea, en la forma adecuada para su tratamiento. Como se puede ver, este concepto es bastante más amplio que el utilizado en otras disciplinas como la Física o las Matemáticas, ya que en Informática un dato no es sólo una temperatura o una longitud, sino que también se entiende como dato una matrícula, una dirección, un nombre, etc. Estos datos los puede obtener el ordenador directamente mediante mecanismos electrónicos (detectar sonidos, Temperaturas, contornos, imágenes,...) o pueden ser introducidos mediante grafismos (letras y números) que es el medio más utilizado (lenguaje escrito). Cualquier información (datos e instrucciones) se puede introducir al ordenador mediante caracteres (letras, dígitos, signos de puntuación, ...). Generalmente el ordenador devolverá la información utilizando también esta forma Escrita.

Las **instrucciones** le indican a la computadora qué es lo que debe realizar y los datos son

los elementos sobre los que actúan o que generan las instrucciones. Visto esto, una computadora la podemos ver como un sistema que tiene como entradas datos e instrucciones y produce en función de éstos unos determinados resultados. El funcionamiento básico de un ordenador se podría describir así:



¿Cuáles son las razones que de alguna forma han obligado a la automatización del tratamiento de la información?

Las principales son:

1. A veces es necesario realizar funciones que el hombre puede abordar por sí mismo, pero que le llevarían mucho tiempo, como por ejemplo, cálculos complejos como los necesarios para el seguimiento y control de naves espaciales (cálculos en tiempo real).
2. Es necesario realizar funciones que el hombre, por sí solo no puede cubrir, como por ejemplo, las comunicaciones a larga distancia.
3. Es necesario obtener seguridad en algunas tareas, sobre todo en las de tipo repetitivo en las que el hombre es más propenso a cometer errores. Sin embargo, las máquinas, una vez que se les ha enseñado cómo realizar las tareas correctamente, repiten el proceso una y otra vez sin cometer ningún error.
4. Se puede sustituir al hombre en las tareas monótonas. Este tipo de tareas no implican el desarrollo de su actividad intelectual, con lo que al automatizarlas, el hombre puede dedicar su esfuerzo a funciones más decisivas e importantes.

REPRESENTACIÓN DE LA INFORMACIÓN.

Debido a las características de las computadoras, la información se almacena dentro de ellas de forma codificada. La *codificación* es una transformación que representa los elementos de un conjunto mediante los de otro, de tal forma que a cada elemento del primer conjunto le corresponde uno distinto del segundo.

ejemplos de códigos: - código de barras

- código de circulación

- carnet de identidad

Dos características importantes de los códigos son que nos permiten comprimir y estructurar la información.

Dentro de la computadora la información se almacena y se transmite en base a un código que sólo usa dos símbolos, el 0 y el 1, y a este código se denomina *código binario*. En la entrada y en la salida de la computadora se realizan automáticamente los cambios de código que sean necesarios, de forma que la información pueda ser entendida fácilmente por los usuarios.

El **BIT** (**B**inary **I**gital, dígito binario) es la unidad elemental de información que equivale a un valor binario (0 ó 1) y constituye, dentro de una computadora la capacidad mínima de información. El bit, como unidad de información mínima, representa la información correspondiente a la ocurrencia de un suceso de entre dos posibilidades distintas.

La información se representa por caracteres (letras, números, ...), internamente estos caracteres se representan utilizando el código binario, es decir, con bits; esto quiere decir que a cada posible carácter le corresponde una secuencia de

bits. Un **byte** es el número de bits necesarios para almacenar un carácter. Este número va a depender del código concreto usado por la computadora, Aunque generalmente se usan 8, esto es, podemos asumir que un byte equivale a 8 bits.

La capacidad de almacenamiento de una computadora o de un soporte físico (como un disco)

se suele dar en bytes o en unidades superiores (múltiplos), ya que el byte es una unidad relativamente pequeña. Los principales múltiplos del byte son:

1 **K**byte (KB): 2¹⁰ bytes.

1 **M**egabyte (MB): 2¹⁰ KB = 2²⁰ bytes.

1 **G**igabyte (GB): 2¹⁰ MB = 2³⁰ bytes.

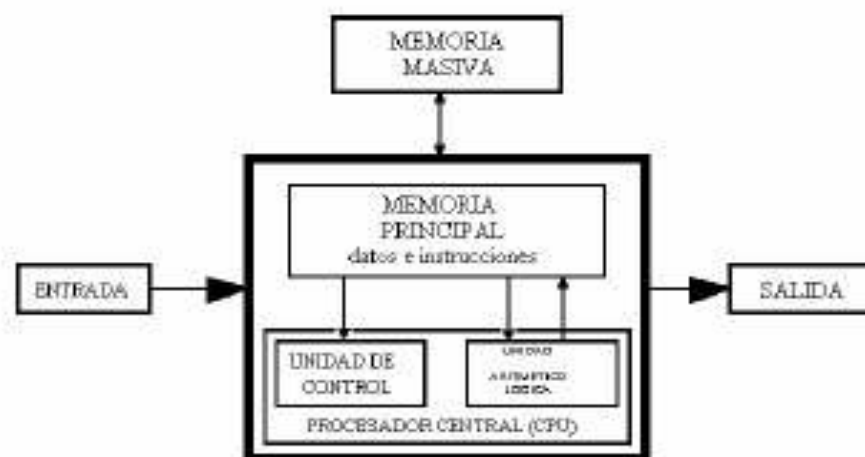
1 **T**erabyte (TB): 2¹⁰ GB = 2⁴⁰ bytes.

Estos múltiplos (K,M,G y T) no solo se pueden utilizar con bytes, sino también con bits. Por

Ejemplo, 1Mb equivale a 2²⁰ bits. (B=byte y b=bit).

ESTRUCTURA DE UN ORDENADOR.

En la figura puede observarse el diagrama de bloques de una computadora básica:



Una computadora se compone de las siguientes unidades funcionales:

- **Unidad de Entrada:** es el dispositivo por donde se introducen en la computadora tanto datos como instrucciones. La información de entrada se transforma en señales binarias de naturaleza eléctrica. Una misma computadora puede tener distintas unidades de entrada. ej.:

- teclado
- scanner
- una unidad de disco
- ...

- **Unidad de Salida:** es el dispositivo por donde se obtienen los resultados de los programas que se están ejecutando en la computadora. En la mayoría de los casos se transforman las señales binarias eléctricas en caracteres escritos o visualizados. ej.:

- monitor
- impresora

- plotter
- una unidad de disco
- ...

La acción combinada de estos dos tipos de unidades -de entrada y de salida-, hace que el usuario de un ordenador sea ajeno a la forma en que éste representa la información.

De manera genérica, tanto a las unidades de entrada como a las de salida, se les denomina

Periféricos.

- **Memoria:** es la unidad donde se almacenan los datos y las instrucciones. En función de la velocidad y también de la capacidad de almacenamiento podemos distinguir dos tipos básicos de memorias

• **Memoria principal o central**, es la más rápida y está estrechamente ligada a las unidades funcionales más rápidas de la computadora (UC y ALU). Es la unidad donde se almacenan tanto los datos como las instrucciones durante la ejecución de un programa.

La memoria está constituida por una serie de posiciones numeradas correlativamente, cada una de las cuales es capaz de almacenar un número determinado de bits. A cada una de estas celdas se le denomina *posición o palabra de memoria*. Cada palabra de memoria se identifica por un número, su *dirección*, que indica la posición que ocupa en el conjunto. Si queremos leer o escribir en una posición de memoria debemos dar su dirección. Por eso se suele decir que la memoria principal es una *memoria de acceso directo* ya que accedemos de forma directa al dato que necesitamos sin más que dar su dirección. Por tanto, el tiempo de acceso a cualquier palabra de memoria es independiente de la dirección o posición a la que se accede.

Dentro de la memoria principal podemos distinguir entre la memoria **ROM** (Read Only Memory) y la **RAM** (Random Acces Memory). La memoria ROM sólo permite leer la información que contiene, pero no se puede escribir en ella. Las memorias ROM no se borran cuando se les deja de suministrar corriente. La memoria RAM es la memoria de lectura y escritura en la que deben estar cargados nuestros programas y sus datos para poder ejecutarse. En la memoria RAM se puede escribir y leer, pero la información que contiene se pierde al dejarle de suministrar corriente (memoria *volátil*).

• **Memoria auxiliar o secundaria.** En contraste con la memoria principal, la memoria auxiliar tiene una alta capacidad de almacenamiento, aunque la velocidad de acceso es notoriamente inferior (es más lenta). Los soportes típicos de memoria auxiliar son los discos y cintas magnéticas, CD-ROM, unidades ZIP, etc. Normalmente los programas y los datos se guardan en disco, evitando el tener que volver a introducirlos (por un dispositivo de entrada) cada vez que queramos utilizarlos. La información almacenada en la memoria auxiliar permanece indefinidamente mientras no deseemos borrarla.

- **Unidad Aritmético-Lógica (ALU):** como su nombre indica se encarga de realizar las operaciones aritméticas (sumas, restas, etc.) y las operaciones lógicas (comparación, operaciones del álgebra de Boole binaria, etc).

- **Unidad de Control (UC):** Esta unidad se encarga de controlar y coordinar el conjunto de operaciones que hay que realizar para dar el oportuno tratamiento a la información. Su función obedece a las instrucciones contenidas en el programa en ejecución: detecta *señales de estado* que indican el estado de las distintas unidades, y en base a estas señales y a las instrucciones que capta de

la memoria principal, genera las *señales de control* necesarias para la correcta ejecución de la instrucción actual. Dentro de la UC existe un *reloj* o *generador de pulsos* que sincroniza todas las operaciones elementales de la computadora. El periodo de esta señal se le denomina tiempo de ciclo. La frecuencia del reloj, medida en MegaHercios (**MHz**), es un parámetro que en parte determina la velocidad de funcionamiento de la computadora.

El esquema de interconexión representado en la figura puede variar dependiendo de la computadora. La *computadora central* está constituida por la UC, la ALU y la memoria principal. Al conjunto formado por la UC y la ALU se le conoce con las siglas **CPU** (Central Processing Unit).

Otra unidad de información ligada a la computadora es la *palabra*. Una palabra está formada por un número entero de bits (8, 16, 32, 64 ...) e indica el tamaño de los datos con los que opera la ALU (*palabra de CPU*) o de los datos transferidos entre CPU y memoria (*palabra de memoria*). La longitud de palabra determina, entre otras cosas, la precisión de los cálculos y la variedad del repertorio de instrucciones. La longitud de palabra, el tiempo de ciclo del reloj y la capacidad de memoria, son factores determinantes para establecer la potencia de una computadora. Aunque actualmente las computadoras son bastante más complejas, conceptualmente el esquema visto sigue siendo válido. Hace unas décadas cada una de las distintas unidades representadas equivalía a un armario independiente conectado por mangueras de cables al resto de las unidades. Actualmente, y debido fundamentalmente al desarrollo de la microelectrónica, varias unidades funcionales pueden estar en una misma tarjeta de circuitos integrados e incluso en un mismo circuito integrado. Por ejemplo, actualmente, la Unidad de Control, Unidad Aritmético Lógica y los registros (de la CPU) están físicamente unidos en un chip al que se denomina *microprocesador*.

El microprocesador es el verdadero cerebro del ordenador. Desde el punto de vista externo, un microprocesador es un chip cuadrado con un tamaño superior al del resto de los chips de la placa base. Un *microordenador* (o microcomputador) es un computador cuyo procesador central es un microprocesador. Conviene destacar el hecho de que el prefijo *micro* en este caso hace referencia al tamaño de la CPU y no a las prestaciones de la misma.

¿Cuál es el funcionamiento básico de un ordenador?

La filosofía general del ordenador es muy simple: Recibe datos del usuario a través de las unidades de entrada, los procesa con la CPU y presenta el resultado mediante las unidades de salida. Pero la CPU no recibe los datos directamente de la unidad de entrada ni los envía directamente a la unidad de salida. Existe una zona de almacenamiento temporal, la memoria RAM, que sirve como lugar de paso obligatorio para acceder a la CPU. Dentro de la CPU, el funcionamiento es el siguiente: Una vez almacenado el programa a ejecutar y los datos necesarios en la memoria principal, la Unidad de Control va decodificando (analizando) instrucción a instrucción. Al decodificar una instrucción detecta las unidades (ALU, dispositivos de entrada, salida o memoria) implicadas, y envía señales de control a las mismas con las cuales les indica la acción a realizar y la dirección de los datos implicados. Las unidades implicadas a su vez, cuando terminen de operar sobre los datos,

enviarán señales a la UC indicando que la acción se ha realizado o bien el problema que ha imposibilitado que se haga. En líneas generales podríamos decir que el funcionamiento del ordenador se rige por dos principios:

- La CPU es la única que puede procesar los datos (lo cual implica que los datos tienen que llegar de alguna forma a la CPU para ser procesados), y
- la CPU sólo puede acceder a los datos almacenados en memoria RAM. Estos dos principios tienen un corolario muy claro que ya fue señalado anteriormente: *Todos los datos, absolutamente todos, tiene que pasar por la memoria RAM para que desde allí puedan ser leídos por la CPU.*

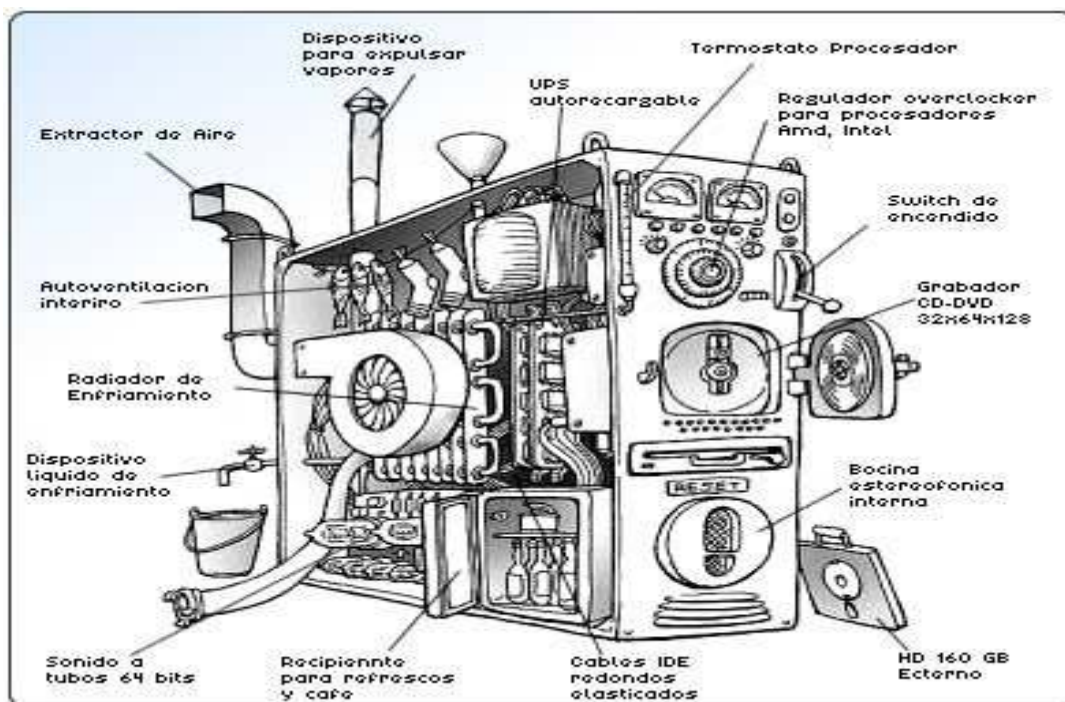
CLASIFICACIÓN DE LOS ORDENADORES

Los ordenadores se pueden clasificar atendiendo a distintos criterios.

En función del uso, podemos distinguir entre:

1. *Computador de uso general*: Puede utilizarse para distintos tipos de aplicaciones, tales como gestión administrativa, cálculos científicos o técnicos. Que realice una aplicación u otra depende del programa que el usuario quiera ejecutar.

2. *Computador de uso específico*: Es aquel que únicamente puede utilizarse para una aplicación concreta. Ej.: Un videojuego, el computador que contiene un robot, el que contiene un misil para seguir su trayectoria o un computador para control del tráfico. Estos computadores se caracterizan porque ejecutan uno o muy pocos programas y porque las unidades de entrada y salida están perfectamente adaptados a la aplicación específica del ordenador. Por ejemplo, en un robot las unidades de entrada suelen ser sensores (de proximidad, de sonido, de formas, etc.) y las unidades de salida, motores.



• **En función de la potencia, capacidad o el tamaño del computador²**, podemos distinguir entre:

1. *Supercomputadores*: Este tipo de computadores se caracterizan por su rapidez. Son de longitud de palabra grande y están constituidas por varios procesadores trabajando en paralelo.

Son multiusuario.

Se utilizan para realizar cálculos a gran velocidad con grandes cantidades de datos. Por ejemplo, predicciones meteorológicas.

2. *Macrocomputadores (mainframes)*: Son computadores de uso general con amplias posibilidades

de procesamiento, memoria y E/S. Son -al igual que los supercomputadores- multiusuario, es decir, que puede haber cientos de usuarios conectados y trabajando a la vez con el mismo ordenador. Cada uno de estos usuarios puede trabajar en un microordenador (que tiene a su vez capacidad de proceso) o simplemente en un terminal, que es una consola sin capacidad de proceso, destinado simplemente a enviar y recibir los mensajes en el mainframe. Suelen tener velocidad y longitud de palabra inferior a la de los supercomputadores.

Son ordenadores típicos para la gestión de grandes empresas.

3. *Minicomputadores*: Son máquinas muy potentes pero de menor potencia, tamaño, velocidad y fiabilidad que los mainframes. Son también multiusuario, pero el número de usuarios admisibles suele ser menor que para los macrocomputadores. Su longitud de palabra puede ser igual o inferior a la de los anteriores. En definitiva, son sistemas similares a los mainframes pero a escala reducida de prestaciones y precio. Son ordenadores usados por empresas medianas.

4. *Estaciones de trabajo (workstations)*: Son un tipo de ordenadores a medio camino entre los minicomputadores y los ordenadores personales. Suelen utilizarse en forma monousuario y tienen como CPU un microprocesador de gran potencia (de 32 o 36 bits). Generalmente trabajan bajo el S.O. UNIX (marca registrada de los laboratorios Bell).

5. *Computadores profesiones / personales (Pc's)*: Son microcomputadores de longitud de palabra

de 16 o 32 bits. Su memoria principal suele tener una capacidad del orden de 4 MBytes. Normalmente se utilizan en forma monousuario. Se caracterizan por la gran cantidad de programas disponibles para ellos y la gran compatibilidad entre unos y otros. Son, en definitiva, ordenadores de pequeño tamaño y precio, cada vez más potentes y fiables, que han hecho posible que el ordenador se introduzca casi como electrodoméstico imprescindible.

6. *Nanocomputadores*: (Este tipo de ordenadores prácticamente tiende a desaparecer). Son microcomputadores de bajo coste y capacidad limitada, ideados para uso doméstico con programas tales como videojuegos, contabilidad familiar o tratamiento de textos. Frecuentemente usan como pantalla de salida un monitor de TV.

7. *Calculadores programables de bolsillo*: Se caracterizan por su pequeño tamaño y alimentación a pilas. Su unidad de entrada es un teclado sencillo que incluye teclas específicas para cada instrucción y su unidad de salida está constituida por un visualizador (display). Disponen de una capacidad de memoria y lenguaje de programación muy limitados.

Hay que insistir en que los límites de esta clasificación son muy imprecisos. Los avances tecnológicos han conseguido, por ejemplo, que actualmente algunos ordenadores personales tengan más capacidad de proceso que algunos mainframes antiguos.

PROGRAMAS E INSTRUCCIONES.

La computadora necesita los programas de igual forma que éstos requieren de computadoras para poder ejecutarse. Un programa es un conjunto de sentencias que se dan a una computadora indicándole las operaciones que se desea que realice. Es un conjunto de sentencias que ha de procesar un ordenador con el objetivo de obtener unos resultados o datos de salida a partir de unos datos iniciales o de entrada. Las sentencias son conjuntos de símbolos y se clasifican en dos tipos:

- Sentencias imperativas o instrucciones: Representan una orden para el ordenador.
- Sentencias declarativas: Proporcionan información sobre los datos que maneja el programa.

Las instrucciones se pueden clasificar en los siguientes tipos:

- instrucciones de transferencia de datos, como pueden ser instrucciones para llevar datos de memoria a la ALU o de memoria a un dispositivo de salida, etc.
- instrucciones de tratamiento, como instrucciones para sumar dos datos, para compararlos, es decir, todo tipo de instrucciones aritmético-lógicas.
- instrucciones de bifurcación y saltos. Este tipo de instrucciones son necesarias ya que las computadoras ejecutan las instrucciones secuencialmente (una detrás de otra) y en determinados momentos podemos necesitar instrucciones para realizar bifurcaciones o saltos que nos permitan alterar el orden de ejecución. Dentro de este tipo de instrucciones cabe resaltar las instrucciones que nos permiten interrumpir la ejecución de un programa y saltar a otro programa (llamado *rutina* o subalgoritmo) para una vez finalizado éste volver al anterior en el punto en que se dejó.

Existen otras instrucciones como esperar a que se pulse una tecla o rebobinar una cinta, etc.

Las sentencias se construyen con unos símbolos determinados y siguiendo unas reglas precisas, es decir, siguiendo un *lenguaje de programación*.

Los circuitos electrónicos de la CPU de la computadora sólo pueden ejecutar instrucciones del lenguaje propio de la computadora, conocido como *lenguaje o código máquina*. Estas instrucciones están formadas por palabras de bits (ceros y unos) que usualmente tienen dos partes diferenciadas, el *código de operación*, que indica cuál es de entre las posibles instrucciones; y el *campo de dirección*, que almacena la dirección de memoria del dato con/sobre el que opera la instrucción.

Programar en código máquina presenta inconvenientes obvios ya que es un lenguaje dependiente de la máquina y si queremos programar otra máquina

deberemos aprender su código máquina particular; además el repertorio de instrucciones suele ser muy pequeño y elemental y sólo se programa usando números, lo que puede resultar muy engorroso. Para paliar estos problemas surgieron los *lenguajes de alto nivel* (PASCAL, C, COBOL,...) que no dependen de la computadora y están pensados para facilitar la programación. Existen unos programas llamados *traductores* que toman como datos de entrada programas escritos en un lenguaje de alto nivel y devuelven como datos de salida el programa en lenguaje máquina de esa computadora equivalente, de tal forma que ya si puede ser ejecutado por la CPU de la computadora.

Existen dos tipos de traductores: *compiladores* e *intérpretes*. Los compiladores funcionan básicamente como se explicó antes, es decir, cogen en conjunto el programa escrito en lenguaje de alto nivel, *programa fuente*, lo traducen y generan un programa en código máquina, *programa objeto*. Los intérpretes, sin embargo, van analizando, interpretando y ejecutando instrucción a instrucción del programa fuente y por tanto no generan programa objeto.

Además de estos programas, el constructor debe suministrar junto con la computadora otra serie de programas que permitan controlar y utilizar de forma eficiente y cómoda la computadora. A este conjunto de programas se le conoce con el nombre de **sistema operativo**. Para utilizar el sistema operativo se utiliza un lenguaje de control constituido por *órdenes o comandos*, que nos sirven, por ejemplo, para grabar los programas y datos en nuestro disquete, o para editar y ejecutar programas, etc. Actualmente los sistemas operativos más difundidos son: Windows, Linux, Unix, entre otros

SOPORTE FÍSICO (HARDWARE) Y SOPORTE LÓGICO (SOFTWARE) DE LOS ORDENADORES.

El soporte físico o *hardware* de una computadora está constituido por la máquina en sí, esto es, por el conjunto de circuitos electrónicos, cables, armarios, dispositivos electromecánicos y cualquier otro componente físico. Por extensión, también se considera hardware todo lo relacionado con la máquina, como son las disciplinas relativas al diseño y construcción de ordenadores.

El soporte lógico o *software* de la computadora es el conjunto de programas (sistema operativo, utilidades y del usuario) ejecutables por la computadora. Por extensión también se considera software a las materias relacionadas con el diseño y construcción de programas.

No hay ordenador sin máquina (hardware) o sin programa (software).

Para que un ordenador funcione es necesario usar programas; es decir, una computadora con sólo soporte físico no funciona. Tan necesario es el hardware como el software.

1.3 PROGRAMACIÓN DE COMPUTADORAS

1.3.1 Lenguajes De Programación

Se puede definir un lenguaje de computadora como una secuencia de elementos, los cuales tienen un significado concreto y entendible. Estos elementos son: las palabras reservadas, los identificadores y los símbolos, la sintaxis del lenguaje define cómo se combinarán todos estos para producir un código ejecutable por la máquina.

El concepto de *programación estructurada* como un enfoque científico a la programación de computadoras lo introdujeron E.W.Dijkstra (izquierda) y C.A.R.Hoare (derecha) a fines de los años sesentas. Mediante el análisis matemático de la estructura de los programas, ellos mostraron que podemos evitar muchos errores de diseño de programas mediante un enfoque sistemático a la programación. Es fundamental en la programación estructurada el diseño adecuado de los algoritmos y el manejo de las estructuras de datos.



En la siguiente lectura se pueden describir mejor algunos de los lenguajes de programación que se han utilizado a lo largo de la historia de la informática, siendo claridad que existen muchos más, los cuales se pueden consultar o profundizar en diferentes sitios de internet o en la biblioteca

Lectura # 2: Lenguajes de programación

“Algunos lenguajes de Programación³

[“FORTRAN clic pra ver ejemplo](#)

El lenguaje fortran es uno de los lenguajes que forman el grupo de lenguajes de computador orientados a procedimientos, los cuales están fundamentados en la estructura del lenguaje usado originalmente para describir el problema, como también en el procedimiento empleado para resolverlo. Tiene por objeto descargar al programador de la tarea de reducir todos los cálculos y toma de decisiones a los pasos elementales requeridos por el repertorio limitado de operaciones ofrecido a nivel de lenguaje de maquina. FORTRAN es un acrónimo de FORMula TRANslation (traducción de formulas), diseñado especialmente para la manipulación de formulas científicas y la aplicación de métodos numéricos a la solución de problemas.

[COBOL clic para ejemplo](#)

En mayo de 1959 mediante una reunión realizada en Estados Unidos por una comisión denominada CODASYL (Conference On Data Systems Languages) integrada por fabricantes de ordenadores, empresas privadas y representantes del Gobierno; un lenguaje de programación fue diseñado expresamente para el procesamiento de datos administrativos. Es un lenguaje de alto nivel y como tal generalmente es independiente de la maquina. Una versión preliminar de COBOL (Common Business Oriented Lenguaje) apareció en diciembre de 1959. Esta versión fue seguida en 1961 por la versión COBOL-61, que constituyó la base para el desarrollo de versiones posteriores. En 1968 se aprobó una versión estándar del lenguaje por lo que ahora se denomina ANSI y una versión revisada se aprobó por ANSI en 1974. El COBOL en cualquiera de sus versiones es el lenguaje apropiado para las aplicaciones administrativas del computador.

[LISP clic para ejemplo](#)

Es el lenguaje para aplicaciones como la inteligencia artificial. Es un lenguaje funcional que ha desempeñado un papel especial en la definición de lenguajes. La definición de un lenguaje debe estar escrita en alguna notación, llamada *metalenguaje o lenguaje de definición*, y los lenguajes de definición tienden a ser funcionales. De hecho la primera implantación de LISP se produjo, casi por accidente, cuando se uso LISP para definirse a sí mismo. De esta manera los conceptos básicos de programación funcional se organizaron con LISP, diseñado por Jhon McCarthy en 1968, el cuál es lenguaje con mayor edad después de Fortran. LISP significa (Lots of Silly Parenthese “montones de tontos parentesis”). A comienzos de 1960 Lisp fue “ultralento” para aplicaciones numéricas. Ahora hay buenas implantaciones disponibles.

[PASCAL clic para ejemplo](#)

PACAL es un lenguaje de programación de alto nivel de propósito general; esto es, se puede utilizar para escribir programas para fines científicos y comerciales. Fue diseñado por el profesor Niklaus (Nicolás) Wirth en Zurich, Suiza, al final de los años 1960 y principios de los

³ [1]

<http://www.itq.edu.mx/vidatec/espacio/aisc/ARTICULOS/leng/LENGUAJESDEPROGRAMACI%D3N.htm>

[2] 1993-2003 Microsoft Corporation. Reservados todos los derechos.

70's. Wirth diseñó este lenguaje para que fuese un buen lenguaje de programación para personas comenzando a aprender a programar. Pascal tiene un número relativamente pequeño de conceptos para aprender a denominar. Su diseño facilita escribir programas usando un estilo que esta generalmente aceptado como práctica estándar de programación buena. Otra de las metas del diseño de Wirth era la implementación fácil.

[PROLOG clic ejemplo](#)

Es un lenguaje de programación de computadoras que fue inventado alrededor de 1970 por Alain Colmerauer y sus colegas de la Universidad de Marcella. A finales de 1970 comenzaron a aparecer versiones de Prolog para microcomputadoras fue el micro-prolog y se dedicaron muchos libros de prolog a él. Pero el micro-prolog no ofrece la riqueza de predicados que ofrece un lenguaje como él turbo prolog. No existió mucho interés en el prolog, hasta que los científicos, japoneses lanzaron su famoso proyecto de la quinta generación con el objetivo de diseñar nuevas computadoras y software, los cuales no tendrían rivales en los años 1990 y posteriores. A las principales implementaciones de prolog le falta la habilidad para mejorar problemas sobre "números" o "procesamiento de texto", en su lugar, prolog está diseñado para manejar "problemas lógicos" (es decir problemas donde se necesita tomar decisiones de un a forma ordenada). Prolog intenta hacer que la computadora razone la forma de encontrar una solución.

[SMALLTALK clic ejemplo](#)

Alan Kay creó SMALLTALK es principalmente un lenguaje *interpretado*, es decir smalltalk es un lenguaje compilado en forma incremental: tanto el compilador como el lenguaje son parte del ambiente de programación smalltalk, cuando se utiliza smalltalk nunca se sale del ambiente de programación (incluyendo el apoyo de biblioteca, las clases y los métodos), usted puede probar incluso el fragmento más pequeño del programa con él interprete, o compilar solo una sección de código. Usted podría utilizar el intérprete smalltalk como una calculadora muy compleja, para evaluar expresiones matemáticas.

OBJECT PASCAL

Es un lenguaje de programación muy poderoso que está si dudas a la altura de C++ y que incluso lo supera en algunos aspectos. Este lenguaje surge a partir del desarrollo de Borland Pascal 7.0, un lenguaje que ocupa un lugar muy importante en la programación de ordenadores personales. El Object Pascal es totalmente compatible con el Borland Pascal 7.0, lo que permite que programas desarrollados con este último puedan ser convertidos a Delphi. Nuevos aspectos en el Object Pascal en relación a sus predecesores son el Excepción – Handling (tratamiento y canalización de errores de run-time), un manejo más sencillo de los punteros con reconocimiento automático y referenciación, las llamados propiedades de objetos que pueden ser asignados como las variables, etc.

[DELPHI clic ejemplo](#)

Es una potente herramienta de desarrollo de programas que permite la creación de aplicaciones para Windows 3.x, Windows 95 y Windows NT. De hecho, aunque el programa ANÁLOGA.EXE corre perfectamente en cualquier tipo de Windows, fue desarrollado sobre una plataforma Windows NT Workstation. Dispone de un compilador muy rápido (más que la mayoría de los compiladores de C++, como ya era tradicional en Turbo Pascal), y potentes herramientas para la creación visual de aplicaciones, completas herramientas para la creación y manejo de bases de datos, aplicaciones multimedia, enlace DDE, creación de DLLs, VBX, etc. Cubre muchos temas de programación bajo Windows: se incluye entre los mismos un completo centro de control para la creación de aplicaciones multimedia, así como una gran variedad de componentes que actúan "debajo" del entorno, como tipos de listado muy variados y contenedores generales de datos. Las aplicaciones terminadas están disponibles en archivos ejecutables (EXE) que pueden utilizarse sólo con bibliotecas adicionales.

JAVA clic ejemplo

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems, una compañía reconocida por sus estaciones de trabajo UNIS de alta calidad en 1991 como parte de un proyecto de investigación para desarrollar software para dispositivos electrónicos (televisores, video cassetes, tostadores y otros de aparatos que se pueden comprar en cualquier tienda departamental). Fundamentado en C++, el lenguaje Java se diseñó para ser pequeño, sencillo y portátil a través de plataformas y sistemas operativos, tanto a nivel de código fuente como binario, lo que significa que los programas en Java (applets y aplicaciones) pueden ejecutarse en cualquier computadora que tenga instalada una máquina virtual de Java. Es un lenguaje ideal para distribuir programas ejecutables vía World Wide Web, además de un lenguaje de programación de propósito general para desarrollar programas que sean fáciles de usar y portables en una gran variedad de plataformas.” [1]

“C clic ejemplo

Lenguaje de programación desarrollado en 1972 por el estadounidense Dennis Ritchie en los Laboratorios Bell. Debe su nombre a que su predecesor inmediato había sido llamado lenguaje de programación B. Aunque muchos consideran que C es un lenguaje ensamblador más independiente de la máquina que un lenguaje de alto nivel, su estrecha asociación con el sistema operativo UNIX, su enorme popularidad y su homologación por el American National Standards Institute (ANSI) lo han convertido quizá en lo más cercano a un lenguaje de programación estandarizado en el sector de microordenadores o microcomputadoras y estaciones de trabajo. C es un lenguaje compilado que contiene un pequeño conjunto de funciones incorporadas dependientes de la máquina. El resto de las funciones de C son independientes de la máquina y están contenidas en bibliotecas a las que se puede acceder desde programas escritos en C. Estos programas están compuestos por una o más funciones definidas por el programador.

C++.

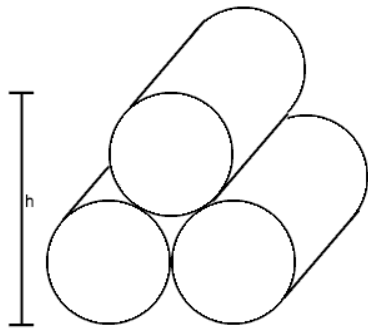
Una versión orientada a objetos derivada del lenguaje de programación de aplicación general denominado C, desarrollada por Bjarne Stroustrup en los Bell Laboratories de la compañía American Telephone and Telegraph (AT&T); en un principio también fue conocido como C with Classes (C con clases, alusión a las clases de la programación orientada a objetos). Comenzó a desarrollarse en 1980 y se nombró C++ en 1983; el primer manual y su primera implementación como producto comercial aconteció en 1985. Versiones sucesivas se publicaron en 1989 y 1990, siendo sus referencias oficiales, además de las publicaciones de su versión estandarizada, las obras The C++ Programming Language (El lenguaje de programación C++, 1985) y Annotated C++ Reference Manual (Manual de referencia comentado de C++, 1990).” [2]

Y por supuesto todos los lenguajes [visuales como Visual Basic, Visual Fox](#); también las últimas tendencias como los .net y lenguajes libres como PHP, entre otros

1.3.2 La lógica de la programación

Uno de los aspectos importantes en la programación de computadoras, es la lógica de la programación, es por eso que esta actividad pretende, a parte de relajar, activar la capacidad de analizar y encontrar los métodos adecuados de solucionar diferentes problemas

Un acertijo geométrico muy simple. Casi para resolver a golpe de vista. Tenemos tres cilindros iguales, de 1 metro de diámetro cada uno, apilados como se ve en la figura.



¿Cuál es la altura de los tres cilindros así colocados?
Como siempre, lo interesante no es tanto el resultado sino cómo lo resolvieron

2.-Mi pequeño sobrino estudiaba geometría, pero le costaba entender la diferencia entre superficie y perímetro.

Tratando de explicarle le pregunté:

-¿Qué superficie tiene tu habitación?-

-Mi habitación es cuadrada y tiene 5 metros más de superficie que de perímetro-

Me costó hacerle entender, pero...

¿Cuánto mide de lado la habitación de mi sobrino?

3.-Sospechando que me estaban haciendo trampa les pregunté:

¿Cuántas cartas tienen ustedes?

Julio: -Si tuviera el doble de las que tengo y Verne me diera dos de las suyas, entonces tendría el cuádruple de las que le quedarían a Verne-

Verne: -Si Julio me diera tres de las suyas, tendría el cuádruple de las que entonces tendría Julio-

¿Cuántas cartas tenía cada uno?

4.-En la ya famosa reunión, Emilio propuso un Juego. Nos puso en la frente unas etiquetas auto adhesivas con un número de forma tal que cada uno podía ver los números de los otros dos pero no el propio.
 -Los números- nos explicó Emilio *-son o tres pares consecutivos, o tres impares consecutivos o tres enteros consecutivos-*
 Yo podía ver que Julio y Verne tenían un 4 y un 6 respectivamente.
 Razoné que, si eran pares consecutivos yo podía tener un 2 o un 8; y si eran enteros consecutivos, podía tener un 5.
 Como no podía deducir nada, me quedé esperando. Después de unos momentos en los que nadie dijo nada, me dí cuenta de cual era mi número.

¿Cuál era?

5.-Dos parejas que se encontraban de viaje estuvieron comprando algunos recuerdos. Al final, cada pareja compró 12 postales. Horacio compró tres postales más que Juana. Karla compró solo dos.

¿Cuántas postales compró Ignacio?

Si usted es un apasionado de los acertijos o los juegos de lógica, lo invito a que ingrese a la página www.markelo.f2o.org, de pronto se convierta en un adicto a este tipo de juegos.

1.3.3 Técnicas de Programación

A partir de este momento nos adentramos en el mundo de la programación, para tal objeto, se propone la realización de la lectura # 3 referente a las diferentes técnicas de programación, propuesta por Justo Méndez (camus_x@yahoo.com),

Lectura #3

“El estudio de los lenguajes de programación agrupa tres intereses diferentes; el del programador profesional, el del diseñador del lenguaje y del Implementador del lenguaje.

Además, estos tres trabajos han de realizarse dentro de las ligaduras y capacidades de la organización de una computadora y de las limitaciones fundamentales de la propia "calculabilidad". El termino "el programador" es un tanto amorfo, en el sentido de que camufla importantes diferencias entre distintos niveles y aplicaciones de la programación. Claramente el programador

que ha realizado un curso de doce semanas en COBOL y luego entra en el campo del procesamiento de datos es diferente del programador que escribe un compilador en Pascal, o del programador que diseña un experimento de inteligencia artificial en LISP, o del programador que combina sus rutinas de FORTRAN para resolver un problema de ingeniería complejo, o del programador que desarrolla un sistema operativo multiprocesador en ADA.

En esta investigación, intentaremos clarificar estas distinciones tratando diferentes lenguajes de programación en el contexto de cada área de aplicación diferente. El "diseñador del lenguaje" es también un termino algo nebuloso. Algunos lenguajes (como APL y LISP) fueron diseñados por una sola persona con un concepto único, mientras que otros (FORTRAN y COBOL) son el producto de desarrollo de varios años realizados por comités de diseño de lenguajes.

El "Implementador del lenguaje" es la persona o grupo que desarrolla un compilador o interprete para un lenguaje sobre una maquina particular o tipos de maquinas. Mas frecuentemente, el primer compilador para el lenguaje Y sobre la maquina X es desarrollada por la corporación que manufactura la maquina X . Por ejemplo, hay varios compiladores de Fortran en uso; uno desarrollado por IBM para una maquina IBM, otro desarrollado por DEC para una maquina DEC, otro por CDC, y así sucesivamente. Las compañías de software también desarrollan compiladores y también lo hacen los grupos de investigación de las universidades. Por ejemplo, la universidad de Waterloo desarrolla compiladores para FORTRAN Y PASCAL, los cuales son útiles en un entorno de programación de estudiantes debido a su superior capacidad de diagnostico y velocidad de compilación.

Hay también muchos aspectos compartidos entre los programadores, diseñadores de un lenguaje implementadores del mismo. Cada uno debe comprender las necesidades y ligaduras que gobiernan las actividades de los otros dos.

Hay, al menos, dos formas fundamentales desde las que pueden verse o clasificarse los lenguajes de programación: por su nivel y por principales aplicaciones. Además, estas visiones están condicionadas por la visión histórica por la que ha transcurrido el lenguaje. Además, hay cuatro niveles distintos de lenguaje de programación.

Los "Lenguajes Declarativos" son los más parecidos al castellano o ingles en su potencia expresiva y funcionalidad están en el nivel más alto respecto a los otros. Son fundamentalmente lenguajes de ordenes, dominados por sentencias que expresan "Lo que hay que hacer" en ves de "Como hacerlo". Ejemplos de estos lenguajes son los lenguajes estadísticos como SAS y SPSS y los lenguajes de búsqueda en base de datos, como NATURAL e IMS. Estos lenguajes se desarrollaron con la idea de que los profesionales pudieran asimilar mas rápidamente el lenguaje y usarlo en su trabajo, sin necesidad de programadores o practicas de programación.

Los lenguajes de "Alto Nivel" son los mas utilizados como lenguaje de programación. Aunque no son fundamentalmente declarativos, estos lenguajes permiten que los algoritmos se expresen en un nivel y estilo de escritura fácilmente legible y comprensible por otros programadores. Además, los lenguajes de alto nivel tienen normalmente las características de "Transportabilidad". Es decir, están implementadas sobre varias maquinas de forma que un programa puede ser fácilmente "Transportado" (Transferido) de una maquina a otra sin una revisión sustancial. En ese sentido se llama "Independientes de la maquina". Ejemplos de estos lenguajes de alto nivel son PASCAL, APL y FORTRAN (para aplicaciones científicas), COBOL (para aplicaciones de procesamiento de datos), SNOBOL (para aplicaciones de procesamiento de textos), LISP y PROLOG (para aplicaciones de inteligencia artificial), C y ADA (para aplicaciones de programación de sistemas) y PL/I (para aplicaciones de propósitos generales).

Los "Lenguajes Ensambladores" y los "Lenguajes Maquina" son dependientes de la maquina. Cada tipo de maquina, tal como VAX de digital, tiene su propio lenguaje maquina distinto y su lenguaje ensamblador asociado. El lenguaje Ensamblador es simplemente una representación simbólica del lenguaje maquina asociado, lo cual permite una programación menos tediosa que con el anterior. Sin embargo, es necesario un conocimiento de la arquitectura mecánica subyacente para realizar una programación efectiva en cualquiera de estos niveles lenguajes.

Los siguiente tres segmentos del programa equivalentes exponen las distinciones básicas entre lenguajes maquina, ensambladores de alto nivel:

Como muestra este ejemplo, a más bajo nivel de lenguaje más cerca esta de las características de un tipo e maquina particular y más alejado de ser comprendido por un humano ordinario. Hay también una estrecha relación (correspondencia 1:1) entre las sentencias en lenguaje ensamblador y sus formas en lenguaje maquina codificada. La principal diferencia aquí es que los lenguajes ensambladores se utilizan símbolos (X,Y,Z,A para "sumar", M para "multiplicar"), mientras que se requieren códigos numéricos (OC1A4, etc.) para que lo comprenda la maquina.

La programación de un lenguaje de alto nivel o en un lenguaje ensamblador requiere, por tanto, algún tipo de interfaz con el lenguaje maquina para que el programa pueda ejecutarse. Las tres interfaces mas comunes: un "ensamblador", un "compilador" y un "interprete". El ensamblador y el compilador traduce el programa a otro equivalente en el lenguaje X de la maquina "residente" como un paso separado antes de la ejecución. Por otra parte, el intérprete ejecuta directamente las instrucciones en un lenguaje Y de alto nivel, sin un paso de procesamiento previo.

La compilación es, en general, un proceso mas eficiente que la interpretación para la mayoría de los tipos de maquina. Esto se debe principalmente a que las sentencias dentro de un "bucle" deben ser reinterpretadas cada vez que se ejecutan por un intérprete. Con un compilador. Cada sentencia es interpretada y luego traducida a lenguaje maquina solo una vez.

Algunos lenguajes son lenguajes principalmente interpretados, como APL, PROLOG y LISP. El resto de los lenguajes -- Pascal, FORTRAN, COBOL, PL/I, SNOBOL, C, Ada y Modula-2 -- son normalmente lenguajes compilados. En algunos casos, un compilador estará utilizable alternativamente para un lenguaje interpretado (tal como LISP) e inversamente (tal como el interprete SNOBOL4 de los laboratorios Bell). Frecuentemente la interpretación es preferible a la compilación en un entorno de programación experimental o de educación, donde cada nueva ejecución de un programa implicado un cambio en el propio texto del programa. La calidad de diagnosis y depuración que soportan los lenguajes interpretados es generalmente mejor que la de los lenguajes compilados, puesto que los mensajes de error se refieren directamente a sentencias del texto del programa original. Además, la ventaja de la eficiencia que se adjudica tradicionalmente a los lenguajes compilados frente a los interpretados puede pronto ser eliminado, debido a la evolución de las maquinas cuyos lenguajes son ellos mismos lenguajes de alto nivel. Como ejemplo de estos están las nuevas maquinas LISP, las cuales han sido diseñadas recientemente por Symbolics y Xerox Corporations.

Los lenguajes de Programación son tomados de diferentes perspectivas. Es importante para un programador decidir cuales conceptos emitir o cuales incluir en la programación. Con frecuencia el programador es osado a usar combinaciones de conceptos que hacen al lenguaje "DURO" de usar, de entender e implementar. Cada programador tiene en mente un estilo particular de programación, la decisión de incluir u omitir ciertos tipos de datos que pueden tener una significativa influencia en la forma en que el Lenguaje es usado, la decisión de usar u omitir conceptos de programación o modelos.

Existen cinco estilos de programación y son los siguientes:

1. Orientados a Objetos.
2. Imperativa: Entrada, procesamiento y salidas de Datos.
3. Funcional: "Funciones", los datos son funciones, los resultados pueden ser un valor o una función.
4. Lógico: {T,F} + operaciones lógicas (Inteligencia Artificial).
5. Concurrente: Aún esta en proceso de investigación.

El programador, diseñador e implementador de un lenguaje de programación deben comprender la evolución histórica de los lenguajes para poder apreciar por que presentan características diferentes. Por ejemplo, los lenguajes "mas jóvenes" desaconsejan (o prohíben) el uso de las sentencias GOTO como mecanismo de control inferior, y esto es correcto en el contexto de las filosofías actuales de ingeniería del software y programación estructurada. Pero hubo un tiempo en que la GOTO, combinada con la IF, era la única estructura de control disponible; el programador no dispone de algo como la construcción WHILE o un IF-THEN-ELSE para elegir. Por tanto, cuando se ve un lenguaje como FORTRAN, el cual tiene sus raíces en los comienzos de la historia de los lenguajes de programación, uno no debe sorprenderse de ver la antigua sentencia GOTO dentro de su repertorio.

Lo más importante es que la historia nos permite ver la evolución de familias de lenguajes de programación, ver la influencia que ejercer las arquitecturas y aplicaciones de las computadoras sobre el diseño de lenguajes y evitar futuros defectos de diseño aprendiendo las lecciones del pasado. Los que estudian se han elegido debido a su mayor influencia y amplio uso entre los programadores, así como por sus distintas características de diseño e implementación. Colectivamente cubren los aspectos más importantes con los que ha de enfrentarse el diseño de lenguajes y la mayoría de las aplicaciones con las que se enfrenta el programador. Para los lectores que estén interesados en conocer con más detalle la historia de los lenguajes de programación recomendamos las actas de una reciente conferencia (1981) sobre este tema, editadas por Richard Wexelblat. Vemos que FORTRAN I es un ascendente directo de FORTRAN II, mientras que FORTRAN, COBOL, ALGO 60, LISP, SNOBOL y los lenguajes ensambladores, influyeron en el diseño de PL/I.

También varios lenguajes están prefijados por las letras ANSI. Esto significa que el American National Standards Institute ha adoptado esa versión del lenguaje como el estándar nacional. Una vez que un lenguaje está estandarizado, las máquinas que implementan este lenguaje deben cumplir todas las especificaciones estándares, reforzando así el máximo de transportabilidad de programas de una máquina a otra. La policía federal de no comprar máquinas que no cumplan la versión estándar de cualquier lenguaje que soporte tiende a "fortalecer" el proceso de estandarización, puesto que el gobierno es, con mucho, el mayor comprador de computadoras de la nación.

Finalmente, la notación algebraica ordinaria, por ejemplo, influyó fuertemente en el diseño de FORTRAN y ALGOL. Por otra parte, el inglés influyó en el desarrollo del COBOL. El cálculo λ de Church dio los fundamentos de la notación funcional de LISP, mientras que el algoritmo de Markov motivó el estilo de reconocimiento de formas de SNOBOL. La arquitectura de computadoras de Von Neumann, la cual fue una evolución de la máquina más antigua de Turing, es el modelo básico de la mayoría de los diseños de computadoras de las últimas tres décadas. Esta máquina no solo influyó en los primeros lenguajes sino que también suministraron el esqueleto operacional sobre el que evolucionó la mayoría de la programación de sistemas.

Una discusión más directa de todos estos primeros modelos no están entre los objetivos de este texto. Sin embargo, es importante apuntar aquí debido a su fundamental influencia en la evolución de los primeros lenguajes de programación, por una parte, y por su estado en el núcleo de la teoría de la computadora, por otra. Mas sobre este punto, cualquier algoritmo que pueda describirse en inglés o castellano puede escribirse igualmente como una máquina de Turing (máquina de Von Neumann), un algoritmo de Markov o una función recursiva. Esta sección, conocida ampliamente como "tesis de Church", nos permite escribir algoritmos en distintos estilos de programación (lenguajes) sin sacrificar ninguna medida de generalidad, o potencia de programación, en la transición.

1.4 TIPOS DE DATOS Y OPERADORES

Como se ha podido ver a lo largo de las lecturas, para que una computadora tenga una razón de ser, se hace necesario la programación de las mismas, es decir realizar software que permita el ingreso de datos (estos datos se representan a nivel de maquina como una secuencia de dígitos binarios (0 o 1) denominados bits) para ser transformada en información. Los datos que se ingresan a una computadora pueden ser:

Numéricos (enteros y reales)

Lógicos (boléanos – verdazo / falso)

Carácter (Char y cadena de caracteres)

Existen lenguajes de programación que admiten una serie de datos complejos, pero para nuestro caso estos van a ser los tipos principales.

Datos Numéricos: están representados por dos tipos principales

- ❖ *Enteros:* Representan los números que no posee componente fraccionaria y pueden ser tanto positivos como negativos
- ❖ *Reales:* Representan todos los números que poseen componente fraccionaria y también pueden ser positivo o negativo

Datos Lógicos (booleano), este tipo de dato solo puede tomar uno de dos valores (verdadero o falso)

Datos tipo Carácter: Representan datos alfanuméricos que pueden ser

- ❖ *Cadena* de caracteres (string), que es una sucesión de caracteres numéricos, letras, símbolos, etc; esta cadena inicia y termina con apostrofes o comillas, dependiendo del lenguaje que se este utilizando, para este caso la representaremos con comillas “Este es un Ejemplo.”
- ❖ *Carácter:* (char), contiene solo un carácter y también se incluye las comillas para su asignación “l”

1.4.1. Variables Y Constantes

Una variable es un espacio reservado en el computador para contener valores que pueden cambiar durante el desarrollo del algoritmo. Los tipos de variables (Numéricas, carácter, lógicas) determinan cómo se manipulará la información contenida en esas. Una variable que se ha definido de un cierto tipo solo puede tomar valores de ese tipo, es el caso de la variable entera x, solo podrá recibir número enteros,

Una Constante: es un espacio reservado para contener valores que no cambian a lo largo de la ejecución de un algoritmo,

Es necesario distinguir que existen variables locales y variables globales:

Variables locales: es aquella que afecta únicamente el subprograma (ver semana: 9)

Variable Global: variable que afecta a un programa en todo su contexto, programa principal y modulo (ver semana 9)

En ambos casos existen una serie de reglas, las características de los nombres de las variables o constante, entre estas están:

Normalmente deben iniciar con una letra

No deben contener símbolos ni signos de puntuación (#,(,?...)

Do deben contener espacios en blanco

Ejemplo:

Contador → correcto

44444 → in correcto

Mi contador → in correcto

Pedro →correcto

#k →incorrecto

K →Correcto

Como podemos observar las variables o constantes se declaran utilizando nombres o letras

Las operaciones que se realicen sobre estas variables y/o constantes, están definidas por una serie de operadores, entre los cuales se encuentran:

Operadores: Aritméticos.

- | | | | |
|------------|---|-----|-----|
| •Potencia. | ^ | ** | |
| •Producto. | * | | |
| •División. | / | Div | Mod |
| •Suma. | + | | |
| •Resta. | - | | |

Operadores: Alfanuméricos.

- Concatenación. +

Ejm.

'UN' + 'AD'

↓

'UNAD'

Operadores: Relacionales.

- Igual a. =
- Menor que. <
- Menor o igual que. <=
- Mayor que. >
- Mayor o igual que. >=
- Distinto a. <>

Operadores: Lógicos.

- | | | |
|-----------------------|-----|----|
| •Negación. | Not | no |
| •Conjunción/producto. | And | y |
| •Disyunción/suma. | Or | o |

Operadores: Paréntesis.

- El paréntesis Permite alterar el orden en que realizan las diferentes operaciones

Ejm. A / (2 * B)

En la ejecución de un programa o algoritmo se hace cumplir una serie de reglas de prioridad que permiten determinar el orden de las operaciones

Orden de evaluación de los operadores

- Paréntesis.
- Cambio de signo.
- Potencias.
- Productos y divisiones.
- Sumas y restas.
- Concatenación.
- Relacionales.
- Negación.

- Conjunción.
- Disyunción.

Observación

El operador MOD, permite obtener el residuo de una división

El operador DIV, Permite obtener la parte entera de una división

1.3.2 Ejercicios de Verificación

Ejercicio 1.0 Asociar la definición con el termino adecuado

- | | |
|---------------------------|---|
| 1. Computador | a)Scanner |
| 2. Informática | b)Maquina Electrónica |
| 3. Unidad de Entrada | d)Sistema Operativo |
| 4. Unix | e)SAS |
| 5. Pascal | e)Compiladores |
| 6. Hardware | f) Tratamiento Automático de la información |
| 7. Lenguajes declarativos | g)Lenguaje de Programación |
| 8. Software | g)Disco Duro |

Ejercicio 2.0 Definir los Siguietes Términos

1. Lenguaje de maquina
2. Interprete
3. Compilador
4. Lenguaje de Alto Nivel
5. Programador

Ejercicio 3.0

1. ¿Porque el procesador es una parte importante del computador?
2. ¿Cuales son las funciones que debe cumplir la memoria Ram?
3. Si usted va a adquirir una computadora en este momento, Cuales serán los criterios necesarios para su elección
4. Considera que es necesario el conocimiento hardware, para poder desarrollar Programas informáticos. por que?
5. Linux es un sistema operativo libre, esto quiere decir que no hay que pagar para su uso, que conoce acerca de este tipo de software?
6. Es usted partidario del software con licencia GNU
7. Los estudiantes del programa de Ingeniería de Sistemas de la Unad, realizan variedad de productos (software), como proyecto de curso o de grado, lo invito a que se acerque a la biblioteca y revise dos proyectos, luego haga un breve comentario de su usabilidad.

2. SEGUNDA UNIDAD:

Estructura General de un Algoritmo

Introducción

En la segunda unidad nos enfocaremos ya a la solución de supuestos problémicos, a pesar de que el curso esta enfocado esencialmente en el trabajo a través de algoritmos, es importante que los estudiantes conozcan de manera sintetizada la importancia que tienen, la solución de estos problemas, utilizando diagramas de flujo, pues esta herramienta no solo es utilidad en ingeniería de sistemas, si no en muchas actividades profesionales, donde se requieren representar problema estructurales mediante diagramas, es por ello, la importancia que los estudiantes de ingeniería de sistemas, conozcan y profundicen en este tipo de instrumentos de análisis y representación grafica. Esta unidad contiene gran cantidad de talleres con ejercicios que ayudarán al estudiante a desarrollar habilidades para saber cuando y como se deben emplear las estrategias de la solución de los mismos; cada planteamiento demostrativo, esta seguido por un pequeño análisis, la solución al mismo y una descripción de la manera como fue solucionado

Se sugiere que los estudiantes no solucionen únicamente los ejercicios propuestos en cada uno de los talleres, sino que traten de solucionar diversidad de ejercicios presentados en los textos que se sugieren como bibliografía

2.1 INTENCIONALIDADES FORMATIVAS:

Propósitos de la unidad

Realizar lecturas y ejercicios que permitan el desarrollo de ejercicios
Conocer dos herramientas que permiten solución de problemas de información, los diagramas y los algoritmos

Objetivos de la unidad

Realizar ejercicios que permitan adquirir habilidades, utilizando diagramas de flujo
Realizar ejercicios que permitan adquirir habilidades, utilizando diagramas de algoritmos
Identificar, variables, constantes, prioridades y reglas de su uso
Identificar los proceso y/o toma de dediciones con diagramas de flujo
Aprender a escribir instrucciones sencillas de toma de decisiones y ciclos

Competencias de la unidad:

El estudiante desarrolla ejercicios básicos utilizando diagramas de flujo y algoritmos

Metas de aprendizaje

El estudiante mediante lecturas y acompañamiento tutorial mediado es capaz de comprender, analizar, desarrollar y proponer ejercicios que permiten evidenciar su aprendizaje

Unidades Didácticas:

Palabras claves:

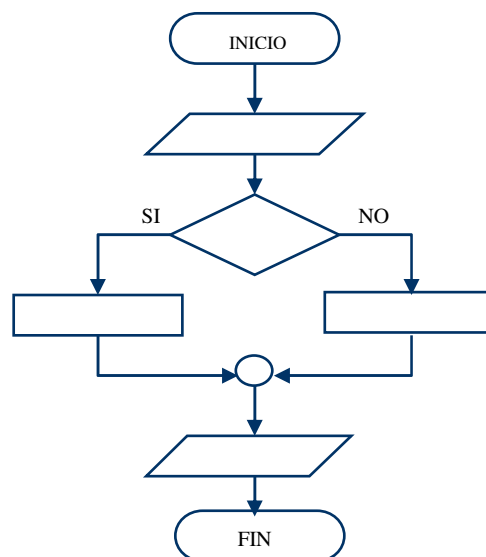
Diagramas de flujo
Algoritmos
Toma de decisión
Ciclos
Funciones

2.2 DIAGRAMAS DE FLUJO

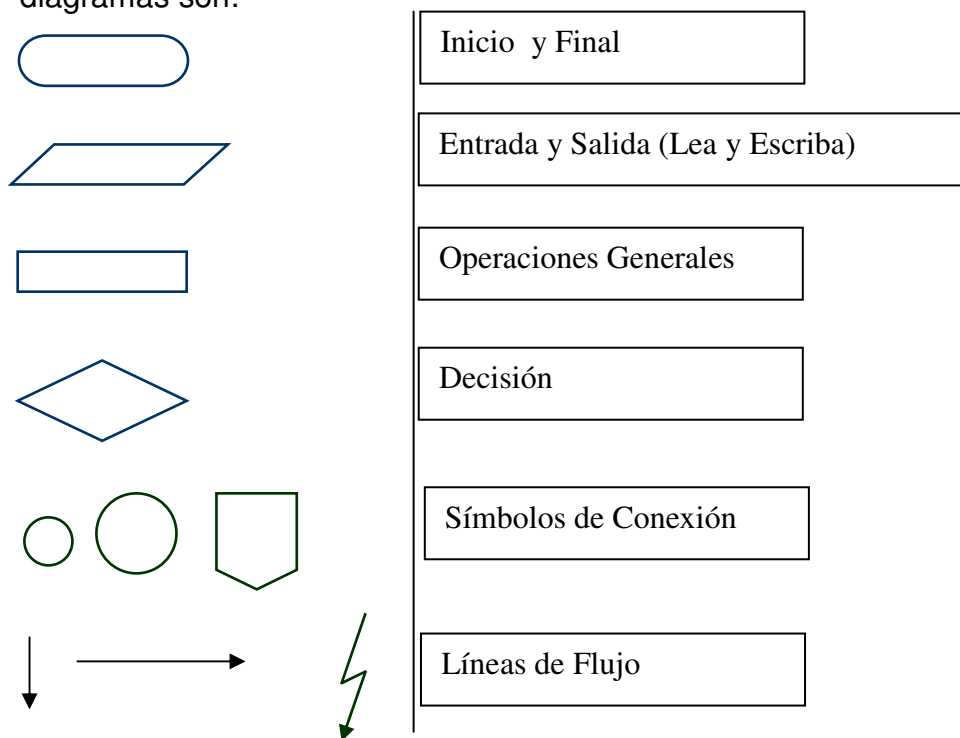
Antes de iniciar en el análisis y la construcción de algoritmos es importante apoyarnos en una herramienta útil en la programación de computadoras como lo es el diagrama de flujo, cuyas características, hace que se aplique no solo en la informática si no en todos los procesos que llevan una secuencia lógica, entre sus aspectos fundamentales están:

- Sencillez. Construcción fácil.
- Claridad. Fácil reconocimiento de sus elementos.
- Utilización de normas en la construcción de algoritmos.
- Flexibilidad. Facilidad en las modificaciones.

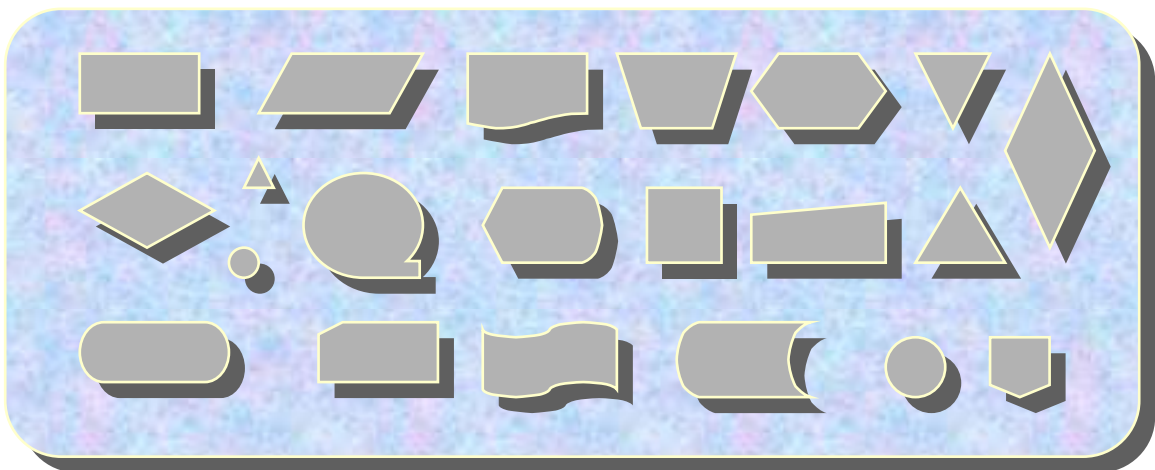
Entonces un diagrama Un diagrama de flujo es la representación gráfica del flujo de datos o de operaciones de un programa.



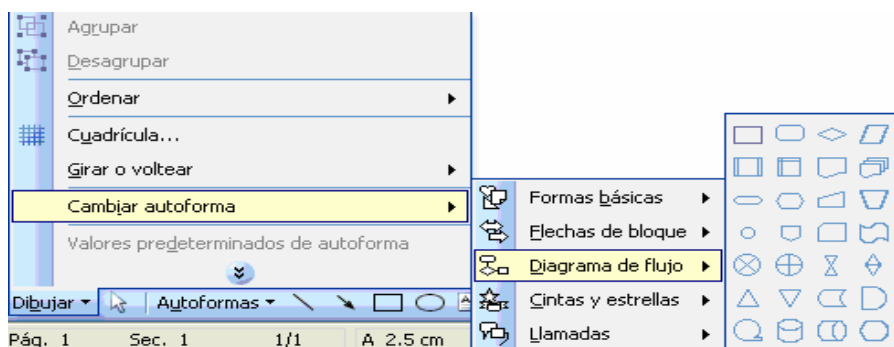
Los símbolos de mayor utilización en la representación grafica por medio de diagramas son:



Para realizar estos gráficos existen plantillas o herramientas que mejoran la presentación



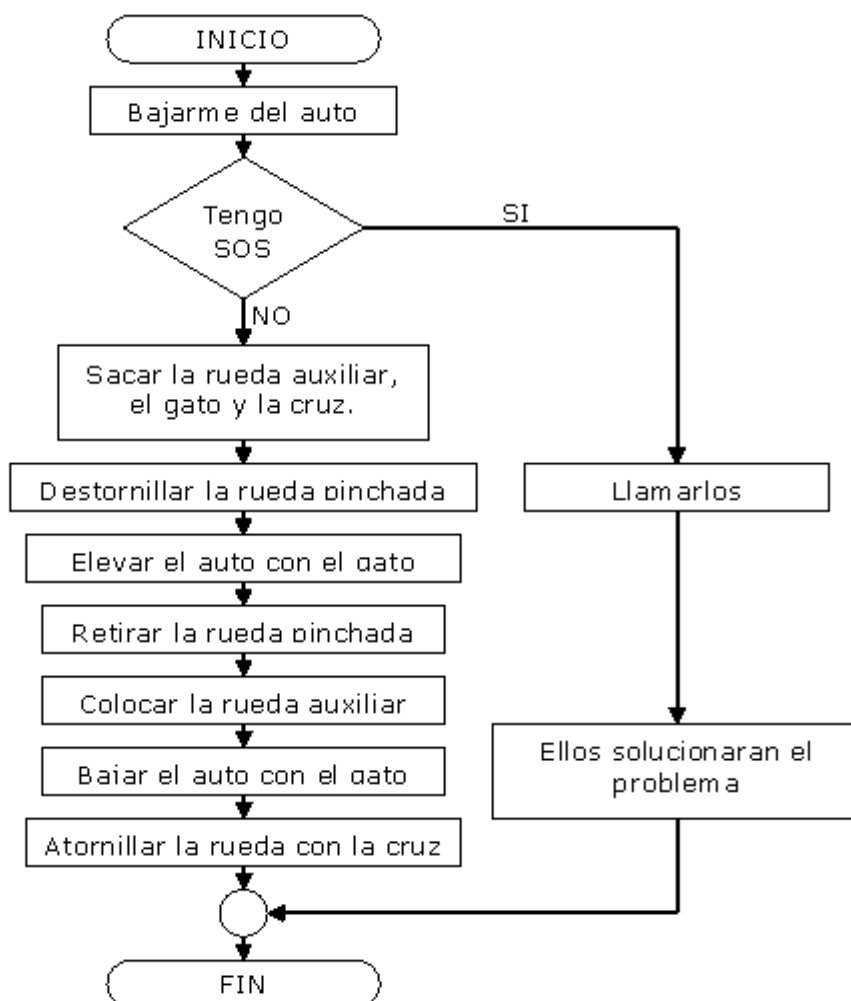
También se pueden encontrar en los procesadores de texto barras que permiten realizar estos gráficos



Este es uno de los cursos en los que se necesita realizar muchos ejercicios para poder lograr un aprendizaje exitoso.

Algunos ejemplos

Supongamos el siguiente problema, viajamos en nuestro auto y este se pincha. Lo primero que debemos hacer es preguntarnos Que?, en nuestro caso la respuesta sería, cambiar la rueda. Luego nos tenemos que preguntar Como?, aquí se establecen los pasos a seguir, podemos optar por la resolución mediante diagrama de flujo, una posible solución sería la solución nos quedaría de esta forma:

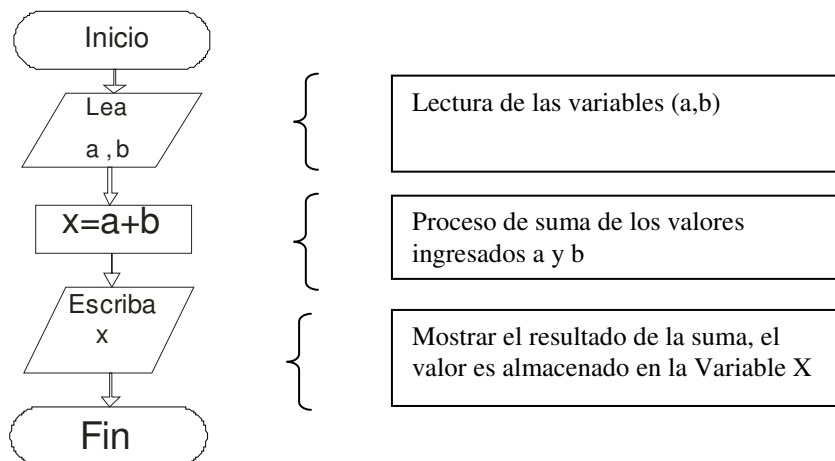


Ejemplo 1: realizar un diagrama que permita leer dos números, sumarlos y mostrar el resultado

Análisis

- 1.- leer cuidadosamente el planteamiento del ejercicio
- 2.-Análisis del Problema
- 3.-Que información debe ser necesaria para la solución del problema
- 3.-Que datos *no conocemos* y son necesarios para darle solución.

Para el ejercicio que nos compete, debemos prestar mucha atención en las variables necesarias para su solución, en este caso no conocemos los dos números y tendremos que captarlos en variables, luego sumarlos (las variables), para luego mostrar el resultado,

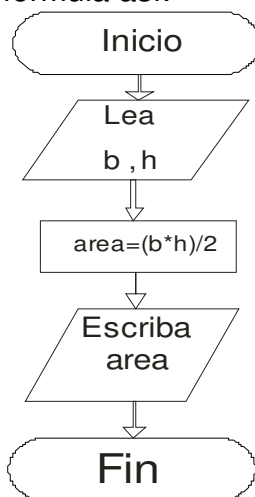


Ejemplo # 2

Encontrar el área de un triángulo y mostrar su resultado

Análisis

Para la realización de este ejercicio es indispensable conocer la fórmula de un triángulo $(b \cdot h)/2$, si nos damos cuenta en la fórmula, existen dos valores que no conocemos, la base y la altura (b, h), por lo tanto esas dos variables se deben pedir y el dos es una constante que no se debe leer, simplemente aplicar en la fórmula así:



Prueba de Escritorio		
La prueba de escritorio se realiza para verificar con datos reales, la correcta construcción del diagrama, para este caso:		
b	h	area
5	2	$(5 \cdot 2)/2 = 5$
<hr/>		
Otros valores		
b	h	area
20	4	40
<hr/>		
-4	-1	?

Avancemos

Ahora vamos a utilizar condicionales, un condicional es un parámetro que permite tomar una decisión, para el caso de la programación estructurada solo existe dos alternativas, un Si o un No

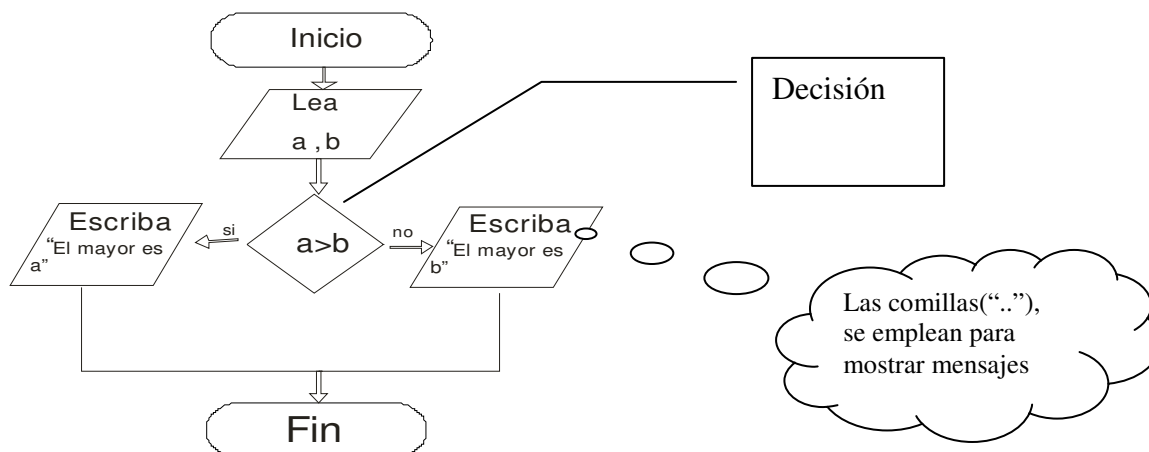
Para entender mejor el concepto lo haremos mediante un ejercicio

Ejemplo # 3

Realizar un diagrama que permita determinar cual es el mayor de 2 números

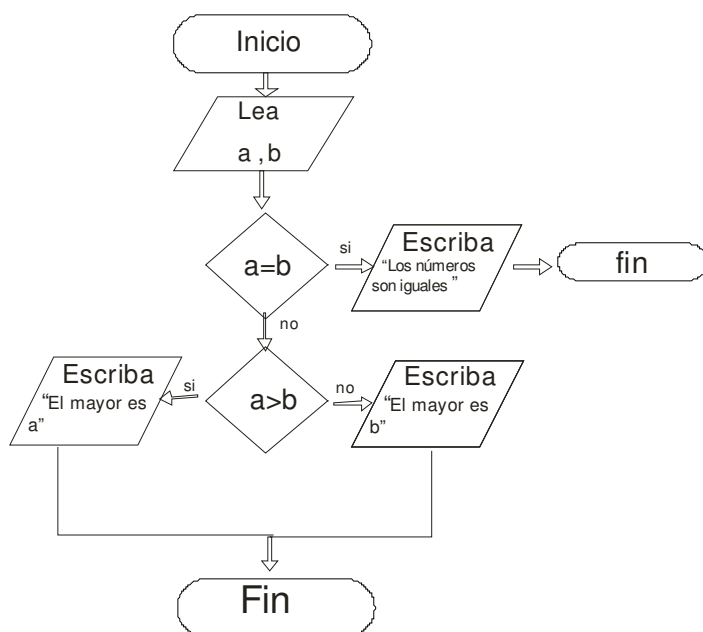
Análisis

Para determinar cual es el mayor de dos números, debemos primero conocer los números, para el caso se deben leer (A,B), luego realizar la comparación, si $a > b$, entonces el mayor es A, en caso contrario el mayor es B:



Una pregunta que nace del ejercicio anterior es, ¿qué pasa cuando son iguales?

Para ese caso necesitamos de un segundo condicional que verifique si las dos variables son iguales ($a=b$)



Se recomienda utilizar la herramienta **dfd**, desarrollada por el grupo Smart de la universidad del Magdalena, la cual la puede descargar de:

http://www.programas-gratis.net/php/descarga.php?id_programa=1808&descargar=DFD%201.0, esta herramienta permite desarrollar y ejecutar los diagramas.

A demás de consultar el anexo "Tutorial de DFD" propuesto por el ingeniero Félix Javier Villero Maestre⁴ y desarrollado en Unicesar

⁴ Coordinador Zona Caribe FCBI - UNAD

2.2.1 ejercicios de verificación

Realizar el análisis, diagrama de flujo y prueba de escritorio, para los siguientes planteamientos

1.-Realizar un diagrama de flujo que permita determinar lo pasos para ir al cine con el novio o la novia

2.-Realizar un diagrama donde se indique los pasos para realizar un plato típico de la Región

3.-Realizar un diagrama que permita esquematizar los pasos para bañar un elefante

4-determinar cual es el cuadrado, de un determinado número

5.-Elaborar un diagrama de flujo que permita determinar si un número es Positivo o Negativo

6.-Ejercicio de conversión, desarrollar un diagrama, que permita ingresar una cantidad en metros y la convierta a Centímetros, Kilómetros, Pies, pulgadas..

7.-Leer una determinada temperatura en grados centígrados y convertirla a Fahrenheit

8.-una persona es apta para prestar el servicio militar obligatorio (presente), cuando: es mayor de 18 años, menor de 25 años, nacionalidad Colombiana y género masculino. Realizar un diagrama que permita determinar si una persona es apta o no par prestar el servicio militar

9.- Elabore un diagrama de flujo que teniendo como datos de entrada el radio y la altura de un cilindro calcule el área total y el volumen del cilindro

10 Una persona recibe un préstamo de un banco por un año y desea saber cuánto pagará de interés al terminar el préstamo si se sabe que el banco le cobra una tasa del 1.8% mensual.

Realice un diagrama de flujo que permita determinar este monto

11.- Elaborar un diagrama de flujo, que permita ingresar 3 valores y los imprima en forma descendente

12.- Una empresa desea conocer el monto de comisión correspondiente a una venta realizada por un vendedor bajo las siguientes condiciones. Si la venta es menor a \$1,000.00, se le otorga el 3% de comisión. Si la venta es de \$1,000.00 o más, el vendedor recibe el 5% de comisión

13.-Una empresa ha decidido, realizar aumentos de salario a sus trabajadores de acuerdo a las siguientes categorías

Sindicalizado	20%
De confianza	10%
Alto directivo	5%
Ejecutivo	0%

Usted debe desarrollar un diagrama que permita ingresar la categoría, el salario actual y calcular el nuevo salario.

14.-Desarrollar una diagrama que permita con dos números, simular una calculadora (+,-,/,*), se debe leer los números y la operación a realizar

15.- Dado un valor de x calcular el valor de y según la siguiente función:

$$y = f(x) = \begin{cases} 3x+36 & \text{si } x \leq 11 \\ x^2 - 10 & \text{si } 11 < x \leq 33 \\ x+6 & \text{si } 33 < x \leq 64 \\ 0 & \text{para los demás valores de } x \end{cases}$$

16. Se recomienda realizar ejercicios básicos planteado en los textos, utilizados como bibliografía de este módulo, a demás de los propuestos por el tutor del curso

2.3 ALGORITMOS

Como podemos observar en las actividades anteriores, los diagramas se convierten en una herramienta básica fundamental, en la resolución de problemas informáticos, pero este módulo lo que pretende es dar a conocerlos, sin profundizar mucho sobre ellos, es trabajo de los estudiantes ahondar en los conceptos y técnicas en el desarrollo de diagramas.

Lectura #4 Sobre el origen de la palabra algoritmo

“

:

MUHAMMAD BIN MUSA AL-KHWARIZMI (Algorizm)

(770 - 840 C. E.)



A Portrait of Al-Khwarizmi

by
Dr. A. Zahoor

Abu Abdullah Muhammad Ibn Musa al-Khwarizmi was born at Khwarizm (Kheva), a town south of river Oxus in present Uzbekistan. (Uzbekistan, a Muslim country for over a thousand years, was taken over by the Russians in 1873.) His parents migrated to a place south of Baghdad when he was a child. The exact date of his birth is not known. It has been established from his contributions that he flourished under Khalifah (Calif) Al-Mamun at Baghdad during 813 to 833 C.E. and died around 840 C.E. He is best known for introducing the mathematical concept Algorithm, which is so named after his last name.

Al-Khwarizmi was one of the greatest mathematicians ever lived. He was the founder of several branches and basic concepts of mathematics. He is also famous as an astronomer and geographer. Al-Khwarizmi influenced mathematical thought to a greater extent than any other medieval writer. He is recognized as the founder of Algebra, as he not only initiated the subject in a systematic form but also developed it to the extent of giving analytical solutions of linear and quadratic equations. The name Algebra is derived from his famous book *Al-Jabr wa-al-Muqabilah*. He developed in detail trigonometric tables containing the sine functions, which were later extrapolated to tangent functions. Al-Khwarizmi also developed the calculus of two errors, which led him to the concept of differentiation. He also refined the geometric representation of conic sections

The influence of Al-Khwarizmi on the growth of mathematics, astronomy and geography is well established in history. His approach was systematic and logical, and not only did he bring together the then prevailing knowledge on various branches of science but also enriched it through his original contributions. He synthesized Greek and Hindu knowledge and also contained his own contribution of fundamental importance to mathematics and science. He adopted the use of zero, a numeral of fundamental importance, leading up to the so-called arithmetic of positions and the decimal system. His pioneering work on the system of numerals is well known as "Algorithm," or "Algorizm." In addition to introducing the Arabic numerals, he developed several arithmetical procedures, including operations on fractions.

In addition to an important treatise on Astronomy, Al-Khwarizmi wrote a book on astronomical tables. Several of his books were translated into Latin in the early 12th century by Adelard of Bath and Gerard of Cremona. The treatises on Arithmetic, *Kitab al-Jam'a wal-Tafreeq bil Hisab al-Hindi*, and the one on Algebra, *Al-Maqala fi Hisab-al Jabr wa-al-Muqabilah*, are known only from Latin translations. It was this later translation which introduced the new science to the West "unknown till then." This book was used until the sixteenth century as the principal mathematical text book of European universities. His astronomical tables were also translated into European languages and, later, into Chinese.

The contribution of Al-Khwarizmi to geography is also outstanding. He not only revised Ptolemy's views on geography, but also corrected them in detail. Seventy geographers worked under Khwarizmi's leadership and they produced the first map of the globe (known world) in 830 C.E. He is also reported to have collaborated in the degree measurements ordered by khalifah (Caliph) Mamun al-Rashid were aimed at measuring of volume and circumference of the earth. His geography book entitled "*Kitab Surat-al-Ard*," including maps, was also translated. His other contributions include original work related to clocks, sundials and astrolabes. He also wrote *Kitab al-Tarikh* and *Kitab al-Rukhmat* (on sundials). “

Existen muchas definiciones referentes a algoritmos, entre las cuales tenemos:

1. un algoritmo es un conjunto de instrucciones las cuales le dicen a la computadora cómo ejecutar una tarea específica
2. Un algoritmo es un conjunto ordenado y finito de instrucciones que conducen a la solución de un problema
3. “Una lista de instrucciones donde se especifica una sucesión de operaciones necesarias para resolver cualquier problema de un tipo dado”.

Un algoritmo esta compuesto por tres elementos esenciales

- a- Cabecera -> donde se da el nombre del algoritmo y se declaran las variables
- b- Cuerpo -> donde se realizan todas las acciones del programa
- c- Final -> donde se da finalización, porque debe ser finito

Ejemplo:

Retomado el primer ejercicio de los diagramas, Leer dos números, sumarlos y obtener su resultado

Como el análisis del ejercicio ya se realizó, pasamos a su solución mediante un algoritmo

1. Algoritmo Suma; 2. Var 3. a,b,suma: entero; 4. inicio 5. escriba("por favor ingrese un número"); 6. lea (a); 7. escriba("por favor ingrese otro número"); 8. lea (b); 9. suma = a+b; 10. escriba ("el resultado es: ",suma) 11. fin
--

Prueba de escritorio		
a	b	suma
4	4	8(4+4)
a	b	suma
20	5878	5898(20+5878)
a	b	suma
-3587	-4578	?_____

Explicación

Sección encabezado (líneas 1,2,3)

Línea 1 ->definición del nombre, todo algoritmo debe ser identificado con un nombre

Línea 2 -> Palabra para identificar las variables a utilizar

Línea 3 -> variables utilizadas y el tipo de las mismas, para este caso de tipo entero (recordemos que los datos numéricos se dividen en enteros y reales)

Línea 4 -> Damos inicio al cuerpo del algoritmo

Línea 5 ->La palabra *escriba* es una directiva de salida, es decir todo lo que se ingresa dentro de esta instrucción son comentarios o valores, que serán visualizados por el usuario, para este caso, estamos solicitando que ingrese un número.

Línea 6 -> La palabra *lea* es una directiva de entrada, significa que todo lo que se escriba dentro, será ingresado, en este caso estamos ingresando un número cualquiera

Línea 7 y 8 -> repite la secuencia de las líneas 5 y 6

Línea 9 -> Proceso, en este segmento se le asigna el valor de la suma de dos variables a una tercera (suma)

Línea 10 -> podemos observar una salida doble, es decir la primera parte un comentario y luego la variable de resultado

Línea 11-> final del algoritmo

Para tener en cuenta

Con los algoritmos, resulta más cómodo, presentar mensajes de claridad al usuario, por ejemplo cuando decimos ("por favor ingrese un número"), en ese momento somos explícitos con lo que queremos hacer,

Resumiendo

Escriba ->sirve para dar a conocer mensajes o resultados

Lea -> permite capturar información

Entrada -----> proceso----->Salida
Lea Escriba

2.3.1 Estructuras de Selección

Las estructuras de selección son estructuras de control utilizadas para la toma de decisiones dentro de un programa. A estas estructuras se conocen también como estructuras selectivas o estructuras de decisión y son las siguientes:

- La estructura de selección simple (SI).
- La estructura de selección doble (SI-SINO).
- Estructura de Selección doble en Cascada SI-SINO-SI

La estructura de **selección simple** permite ejecutar una acción o un grupo de acciones sólo si se cumple una determinada condición.

```
Si (condicional)
    Sentencia 1
    Sentencia 2
    ...
Fin si
```

La estructura de **selección doble** permite seleccionar una ruta de dos rutas posibles en base a la verdad o falsedad de una condición.

```
Si (condicional)
    Sentencia 1
    Sentencia 2
    ...
Si_no
    Sentencia 1
    Sentencia 2
    ...
Fin fi
```

La estructura de **selección doble en cascada** esta formada por varias estructuras de selección doble SI-SINO puestas una a continuación de otra de forma que a un SI-SINO le sigue otro SI-SINO.

En la estructura de selección doble en cascada, las condiciones se evalúan orden descendente, pasando de una condición a otra si la condición anterior resulta falsa. En el momento que se encuentra una condición verdadera, se efectúa acción correspondiente a dicha condición se corta el resto de la estructura. Si todas las condiciones resultan falsas, se efectuará acciones correspondientes al SINO, que se considera como la acción por defecto.

SI(condicional1)
 accion1
SINO SI(condicional2)
 accion2

```

SINO SI( condicional3 )
    accion3
    .
    .
    .
SINO
    ....

```

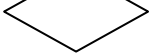
El siguiente ejemplo ilustra el manejo de condicionales

Ejemplo 2

Determinar cual de 2 números es mayor

1. algoritmo mayor
2. var
3. a,b: entero;
4. inicio
5. Escriba("Por Favor entre un número");
6. Lea(a);
7. Escriba("por favor entre el segundo número");
8. lea (b);
9. si (a>b)
10. escriba("El mayor de los números es: ",a);
11. sin_no
12. escriba("El mayor de los números es: ",b);
13. fin_si
14. fin

Comentarios

Línea 9: los condicionales que se representaban  como, ahora lo expresamos como un *Si()*,

Línea 11: para este ejercicio, la sentencia *sin_no* representa caso contrario

Línea 13: Toda instrucción *Si*, debe terminar con un *fin_si*, para indicar hasta donde va ese condicional.

Ejemplo 3

Variación del ejercicio anterior, que pasa si los números son iguales

Entonces una posible solución es la siguiente

1. Algoritmo mayor_v1
2. var
3. a,b:entero
4. inicio
5. Escriba("Por Favor entre un número");
6. Lea(a);
7. Escriba("por favor entre el segundo número");
8. Lea (b);
9. Si (a=b)

10. escriba ("los Números son Iguales");
11. fin_si
12. Si ($a > b$)
13. escriba ("El mayor es :", a);
14. fin_si
15. Si ($a < b$)
16. escriba("el mayor es :",b);
17. fin_si
18. fin

Por favor discuta y analice con su compañeros de grupo, porque en este ejercicio se emplearon 3 condicionales?, Existen otras formas de resolverlo?

2.3.1.1 Ejercicios de Verificación

1. desarrollar mediante algoritmos los ejercicios propuestos en el ítem. 2.2(Ejercicios de Verificación con diagramas de flujo)
2. Profundización en los temas :
 - Tipos de instrucciones
 - Instrucciones de Asignación
 - Instrucciones de Entrada
 - Instrucciones de Salida
 - Instrucciones de Decisión

2.3.2 Estructuras de secuencia Ciclos o Bucles

Este tipo de estructuras marcan como orden de ejecución la reiteración de una serie de acciones basándose en un bucle.

“*Un BUCLE* (loop, en inglés) es un trozo de algoritmo cuyas instrucciones son repetidas un cierto número de veces, mientras se cumple una cierta condición que ha de ser claramente especificada. La condición podrá ser verdadera o falsa, y se comprobará en cada paso o iteración del bucle.

Básicamente, existen tres tipos de estructuras repetitivas:

Desde o Para: este tipo de ciclo es ideal cuando se conoce la cantidad de veces que se desea ejecutar una acción

```
...
Para (Contador=ValorInicial Hasta ValorFinal)
    ....
    Instrucción(es)
    ....
Fin para
...
```

Mientras que: este tipo de ciclo se ejecuta mientras se cumpla una determinada condición, en este caso la condición se evalúa al inicio del ciclo

```
...
Mientras (Exp. booleana) hacer
    ....
    Instrucción(es)
    ....
Fin mientras
....
```

Repita... hasta que: este ciclo es similar al anterior, solo que, en este tipo de ciclo, la condición se evalúa al final, permitiendo que el bucle se ejecute por lo menos una vez

```
....
Repita
    ....
    Instrucción(es)
    ...
Hasta que (exp. Booleana)
...
```

También se hace necesario conocer otros conceptos que se manejan mucho cuando hablamos de ciclos o bucle ellos son:

- Decisión: donde se evalúa la condición y, en caso de ser cierta, se ejecuta.
- Cuerpo del bucle: son las instrucciones que queremos ejecutar repetidamente un cierto número de veces.
- Salida del bucle: es la condición que dice cuándo saldremos de hacer repeticiones (caminar mientras encuentro un sillón, en el momento de encontrar un sillón dejo de caminar y he salido del ciclo).

Una forma de controlar un bucle es mediante una variable llamada **CONTADOR**, cuyo valor se incrementa o decrementa en una cantidad constante en cada repetición que se produzca.

También, en los bucles suele haber otro tipo de variables llamadas **ACUMULADOR**, cuya misión es almacenar una cantidad variable resultante de operaciones sucesivas y repetidas. Es como un contador, con la diferencia que el incremento/decremento es variable.

2.3.2.1 Ciclo para o desde

La estructura repetitiva PARA ó DESDE permite que las instrucciones las cuales contiene en su ámbito, se ejecuten un número de veces determinado. Una variable de control, que llamamos contador, se incrementa o decrementa desde un valor inicial hasta un valor final. Supondremos un incremento de uno en uno y decrementos de menos uno en menos uno.

Bueno para ser explícito, lo mejor es realizar un ejercicio donde se comprenda todos los conceptos mencionados hasta el momento.

Ejercicio

Realizar un algoritmo que sume los 10 primeros números naturales e imprima su resultado.

Análisis

1. los números naturales se tomarán del 1 al 10
2. como se conoce la cantidad de veces que hay que sumar los números, lo mejor es utilizar el ciclo desde o para
3. como se necesita sumar los números, lo mejor es utilizar un acumulador
4. En este caso, los números se conocen, por lo tanto no habría que capturar ningún valor

Solución

1. algoritmo suma10
2. var
3. k,suma : entero;
4. inicio
5. suma=0;
6. Para (k=1 hasta 10 hacer)
7. suma=suma+k;
8. fin_para
9. Escriba("la suma es: ",suma);
10. fin

Prueba de escritorio	
k	suma
-	0
1	1
2	3
3	6
4	10
5	15
6	21
7	28
8	36
9	45
10	55

Explicación

Línea 5 -> es importante inicializar las variables que serán utilizadas como *acumuladores*

Línea 6 -> instrucción del ciclo básico desde el número 1 hasta el 10, en este caso la variable k se comporta como un *contador*

Línea 7-> Al valor anterior que tiene la variable *suma*, le acumulamos un nuevo valor, como puede observarse en la prueba de escritorio

Línea 8-> todo ciclo al igual que todo condicional se debe cerrar

Línea 9-> esta línea, se hubiese podido colocar antes de cerrar el ciclo, pero se imprimiría 10 un resultado

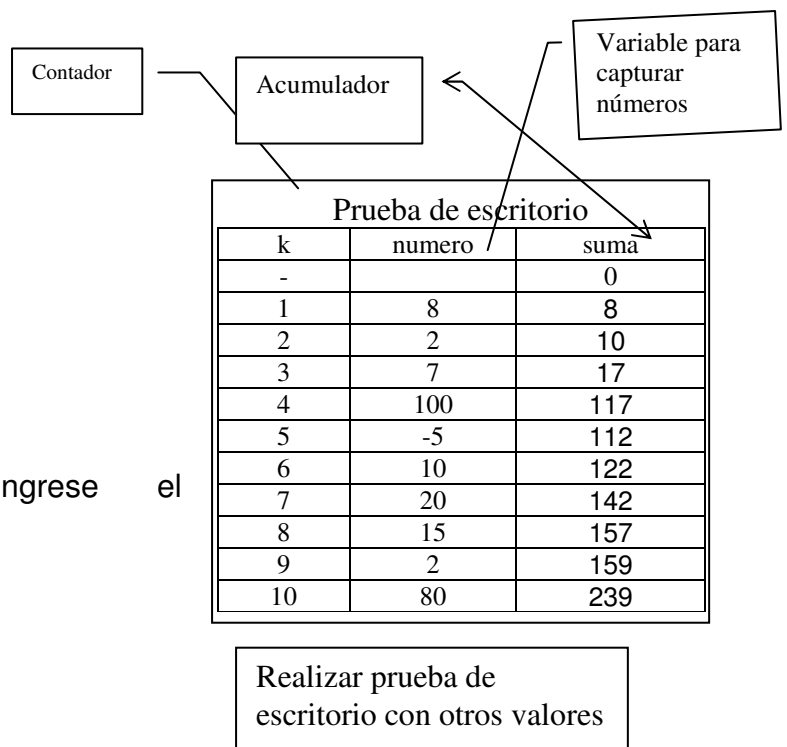
Ejercicio 2

Una pequeña variación al ejercicio anterior

Realizar la suma de 10 números cualesquiera e imprimir su resultado

Análisis

- 1.- Para este caso se saben cuantos números serán los sumados, por lo cual se recomienda utilizar nuevamente el ciclo para
- 2.- No se sabe cuales son los números, por lo tanto hay que capturarlos
- 3.- se aumentará una variable, la que captura los números



Solución

```

1. algoritmo suma10_1
2. var
3. k, numero, suma : entero;
4. inicio
5. suma=0;
6. Para (k=1 hasta 10 hacer)
7.   Escriba("por favor ingrese el
   número");
8.   lea(numero)
9.   suma=suma+numero;
10. fin_para
11. Escriba("la suma es: ",suma);
12. fin
  
```

Explicación

Línea 7-> Es un mensaje para que el usuario sepa que tiene que hacer

Línea 8-> Es necesario capturar el número, en este caso 10 veces porque son 10 números diferentes (ver prueba de escritorio)

Línea 9-> En este caso no acumulamos *k* a la variable *suma*, por *k* es simplemente un contador y lo que piden es sumar los números, por lo que hay que acumular es el valor de la variable *numero*

Ejercicio

Realizar un algoritmo que permita ingresar 10 números, de los cuales se debe sumar aquellos que son positivos y contar los que son negativos, imprimir los resultados

Análisis

- 1.-Al igual que los ejercicios anteriores, se sabe cuantos números son
- 2.-Nuevamente el ciclo recomendado es el ciclo para o desde (desde $i=1$ hasta 10)
- 3.-Se complica un poco, pues se pide sumar y contar dependiendo el número
- 4.-Tendremos que utilizar una variable para contar y otra para acumular
- 5.-es necesario dentro del ciclo tomar una decisión

Solución

```

1. algoritmo suma10_2
2. var
3. k, numero, suma, kn : entero;
4. inicio
5.   suma=0;
6.   kn=0;
7.   Para (k=1 hasta 10 hacer)
8.     Escriba("por favor ingrese el número")
9.     lea(numero)
10.    si (numero >= 0)
11.      suma=suma+numero;
12.    si_no
13.      kn=kn+1;
14.    fin_si
15.  fin_para
16.  Escriba("la sumatoria de positivos es: ",suma);
17.  Escriba ("la cantidad de # negativos es: ",kn);
18. fin

```

Prueba de escritorio			
k	kp	numero	suma
-	0		0
1		8	8
2		2	10
3	1	-7	
4		100	110
5	2	-5	
6		10	120
7	3	-20	
8		15	135
9	4	-2	
10		80	215

Explicación

Línea 6-> inicializamos en cero la variable que utilizamos como contador de números negativos

Línea 10-> condicionamos el número leído, para determinar si es positivo o negativo, en el caso de ser positivos lo sumamos en la línea 11, en el caso de ser negativo lo contamos en la línea 13

2.3.2.1.1 EJERCICIOS DE VERIFICACIÓN

Desarrollar el siguiente taller

1.-Realizar un algoritmo que permita leer 20 temperaturas (grados C°) diferentes durante un día, se debe indicar cual fue la temperatura el promedio de ese día

2.-Una empresa que cuenta con n empleados desea realizar algunos cálculos para la nueva nómina. Los datos con que cuenta son los sueldos de los n empleados:

$$n, s_1, s_2, s_3, \dots, s_n.$$

Elabore un algoritmo de para leer los datos y contestar a las siguientes preguntas:

- a) ¿Cuál es el aumento correspondiente a cada empleado según el siguiente criterio?
- 17% si el sueldo es inferior a \$5,000
 - 10% si el sueldo está entre \$5,000 y \$15,000
 - 5% si el sueldo es superior a \$15,000
- b) ¿Cuál es el nuevo sueldo para cada empleado?
- c) ¿Cuál es el total de la nueva nómina?
- d) ¿Cuál es el incremento en la nómina?
- e) ¿Cuál es el máximo sueldo nuevo?
- f) ¿Cuál es el mínimo sueldo nuevo?

Se sugiere leer la variable k para determinar la cantidad de empleados antes de hincar el ciclo

3.-se desea desarrollar un algoritmo que permita, desarrollar la tabla de multiplicar de un determinado número (la tabla básica va de 1 a9);

4.-Variación del ejercicio anterior, se debe desarrollar un algoritmo que permita mostrar las tablas del 1 al 9

5.-Se desea construir un algoritmo que permita imprimir el resultado del factorial de un número dado (factorial =n!);

6.- una empresa con 20 empleados desea saber cuantos ganan menos de un salario mínimo, cuantos tienen un salario entre uno y dos salarios mínimos y cuantos ganan más de tres salarios mínimos, además cual es el valor actual de la nomina de la empresa, cuanto aumentará la nomina mensual si se hace incrementos así; 20% a aquellos que gana menos de un salario mínimo, 10% a los que ganan entre 1 y dos salarios mínimos y 5% a quienes gana más de 3 salarios mínimos.

Se deben realizar los cálculos, teniendo en cuenta el valor del salario mínimo legal vigente

7.-Se desea desarrollar un algoritmo que permita determinar e imprimir el promedio de las edades de los alumnos de el curso algoritmos de este Cead, a demás se debe aprovechar la consulta para determina cuantos son casados, cuanto viven en unión libre, cuantos solteros, cuantos son viudos, cuantos vienen de otras ciudades, cuantos conocen la misión de la Unad, si lo desean lo pueden clasificar por géneros.

La realización de este ejercicio con datos reales podrá conocer un poco más a sus compañeros de curso

8.-Consultar información referente a los ciclos mientras y el ciclo Repita

2.3.2.2 Ciclo Mientras

Controla la ejecución de un conjunto de instrucciones de tal forma que éste se ejecuta mien-tras se cumpla la condición de control que aparece al comienzo de la instrucción. Es decir funciona siempre y cuando la condición sea verdadera.

Ejemplo

Se debe desarrollar un algoritmo que este permita ingresar las notas del curso de algoritmos, el programa debe terminar cuando la nota ingresada es cero (cero), luego mostrar el promedio de las notas ingresadas

Análisis

1. En este ejercicio no se sabe la cantidad de notas
- 2.-Se sabe que se termina cuando una de las notas ingresada es cero
- 3.-Se debe utilizar un contador, para determinar el número de notas ingresadas
- 4.-también se debe utilizar un acumulador para sumar cada nota
- 5.-se utilizara una variable extra si es del caso para obtener el promedio.

Solución

1. Algoritmo notas
2. Var
3. k : entero
4. suma, nota, promedio: real;
5. inicio
6. suma=0;k=0;
7. **mientras (nota <>0)**
8. escriba("entre la nota");
9. lea (nota)
10. si (nota >0)
11. suma=suma+nota;
12. k=k+1;
13. fin_si
14. fin_mientras
15. promedio=suma/k;
16. escriba("la cantidad de notas ingresadas son: ",k);
17. escriba("el promedio de las notas es de :",promedio);
18. fin.

Prueba de escritorio			
k	suma	nota	promedio
0	0	-	-
1	3.5	3.5	
2	7.5	4.0	
3	9.5	2.0	
4	14.5	5.0	
-	-	0	3.625

Explicación

Línea 4-> no olvidemos que en algunos casos no se debe emplear números enteros

Línea 6 -> se puede emplear una línea para realizar varias actividades

Línea 7 -> se da inicio al ciclo con el condicional, para este caso que la nota sea diferente a 0, la sentencia diferente también se puede representar como (!=)

Línea 10. Es importante el condicional para controlar que solo recibamos números positivos;

Línea 14->todo ciclo se debe cerrar

2.3.2.3 Ciclo Repita hasta que

Controla la ejecución de un conjunto de instrucciones de tal forma que éste se ejecuta hasta que se cumpla la condición de control que aparece al final de la instrucción.

Ejercicio

Se realiza una variación al ejercicio anterior

Las notas permitidas son únicamente, números positivos comprendidos entre 0.1 y cinco.

Con este ejercicio se pretende abordar un tema de mucha importancia, como son los *filtros*, los cuales solo permiten el ingreso de unos determinados valores, en caso de que esos valores no sean validos se generara mensajes de error,

Con este planteamiento, podemos involucrar los dos ciclos

Análisis

- 1.-El análisis prácticamente es el mismo anterior, solo que se debe tener en cuenta las notas permitidas (0.1 a 5.0)
- 2.-En caso de no ser una nota permitida el algoritmo emitirá un mensaje y pedirá un nuevo número

```

1. Algoritmo notas_v1
2. Var
3. k : entero
4. suma, nota, promedio: real;
5. inicio
6.   suma=0; k=0;
7.   mientras (nota <>0)
8.     repita
9.       escriba("entre la nota");
10.      lea(nota);
11.      si (nota <0) o (nota >5)
12.        escriba("Error. intentelo nuevamente");
13.      fin_si
14.    hata que (nota<0 ) o (nota >5);
15.    if (nota <>0 )
16.      suma=suma+nota;
17.      k=k+1;
18.    fin_si
19.  fin_mientras
20.  promedio=suma/k;
21.  escriba("la cantidad de notas ingresadas son: ",k);
22.  escriba("el promedio de las notas es de :",promedio);
23. fin
  
```

Explicación

De la línea 8 a la 14 esta controlada por el ciclo *repita ..hasta que*

Línea 11-> este condicional permite generar un mensaje de error, si la nota esta fuera del rango establecido

Línea 14, termina el ciclo repita con una condición de verificación

Línea 15->, por qué validar que la nota sea diferente de 0,?. Bueno la respuesta es que el cero es nuestro valor *CENTINELA* o *BANDERA* y no lo estamos teniendo en cuenta dentro del ciclo repita ni en el condicional

Observación importante, en este ejercicio, hemos empleado los conectores lógicos, para este caso el conector o lo podemos observar en las líneas 11 y 14, se sugiere consultar más referente a estos conectores.

2.2.3.4 ejercicios de verificación

Evidencia: documento con informe de la actividad, y registro en el portafolio

1.-determine cual es la diferencia entre:

- a) contador y acumulador
- b) ciclo para y ciclo mientras
- d) ciclo mientras y ciclo repita
- e) condicional y ciclo

2.-Realizar los siguientes ejercicios

a) Los cubos de Nicómaco. Considera la siguiente propiedad descubierta por Nicómaco de Gerasa: Sumando el primer impar, se obtiene el primer cubo. Sumando los dos siguientes se obtiene el segundo cubo. Sumando los tres siguientes, se obtiene el tercer cubo y así sucesivamente

Es decir:

$$1 = 1^3,$$

$$3 + 5 = 2^3 = 8,$$

$$7 + 9 + 11 = 3^3 = 27,$$

$$13 + 15 + 17 + 19 = 4^3 = 64.$$

Elabore un algoritmo que dado un número n entero positivo, imprima los n primeros cubos utilizando esta propiedad.

b)la serie fibonacci es un ejercicio interesante, el cual se construye a partir de los dos primeros números que son el 0 y 1, y apartir de ahí se construye la serie ejemplo: 0,1, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34.55.. Desarrollar un algoritmo que permita, calcular n números de esta serie

c) En un periodo académico existen 45 alumnos y cada uno de ellos tiene 4 calificaciones, correspondientes a 5 diferentes cursos. Se requiere encontrar:

- ❖ El promedio de cada estudiante.
- ❖ el promedio general del curso (=promedio de los promedios).

d) se debe realizar una mejora al algoritmo anterior, permitiendo controlar para cada alumno n cursos, se debe imprimir los mismos ítems

e) Para las elecciones presidenciales que se realizarán en Colombia, existen tres partidos políticos aspirando con sus candidatos (1, 2,3). Uno de estos ha decidido realizar una consulta (encuesta) a un cierto número de personas, para determinar las preferencias de los electores

A cada persona se le pregunta:

Si va a votar,
En caso de que la respuesta se afirmativa, se le preguntará por
qué partido votará.

Elaborar un algoritmo, para llevar un control de la información y así obtener unos resultados con prontitud

Nota: el dato *partido* solamente se lee si la persona entrevistada ha contestado que sí votará.

El algoritmo imprimirá la siguiente información:

- ❖ ¿Cuál es el partido que esta repuntando?
- ❖ ¿cuál es % de abstención?
- ❖ ¿Cuál es % a favor de cada partido, teniendo en cuenta, las entrevistas validas?
- ❖ ¿cual es el % de personas que SI votaran?

f) Realizar un algoritmo que permita determinar si un número es par o impar, teniendo en cuenta las siguientes condiciones:

Solo se admiten números positivos

Solo se evalúan números entre 1000 y 147890, cualquier otro número genera una advertencia de error

El algoritmo, termina cuando el usuario decide no ingresar más números ("Desea continuar S/N")

g) Un almacén desea obtener una serie de informes diarios a partir de las ventas realizadas en un día. Elabore un algoritmo que:

Solicite el código del artículo

El valor unitario

La cantidad de artículos

El código para terminar es -999

Se desconoce el número de ventas que se realizan en un día, por lo que el final de los datos se indica con un -1. Suponga que el IVA es del 15%.

2.3.3 SUBPROGRAMA O MODULO

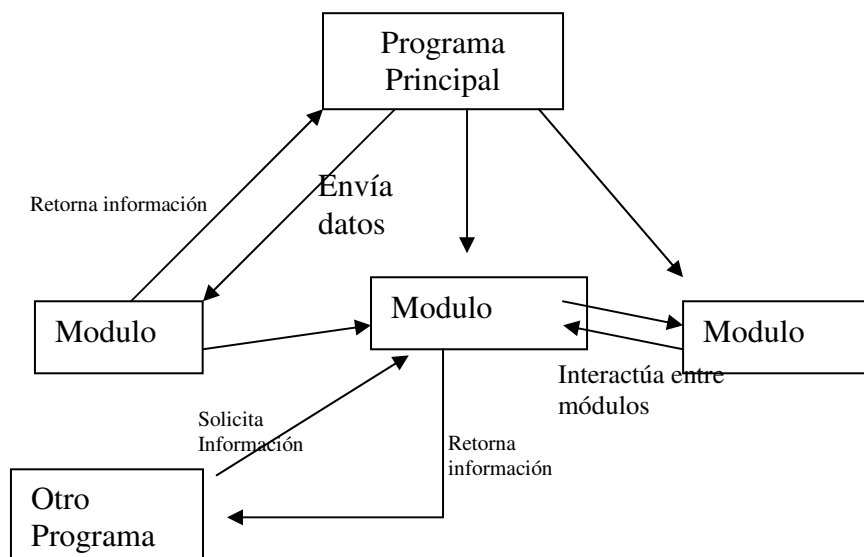
Hay una expresión que dice “*Divide y Vencerás*”, precisamente los módulos lo que permiten es dividir los programas grandes, en fragmentos pequeños.

- Un subprograma o modulo es un trozo de código que tiene
 - Entradas
 - Salidas
 - Instrucciones
 - Es otro programa “prendido” al programa principal
- Un subprograma puede ser utilizado por el programa principal o por otros subprogramas

Al dividir un programa en módulos este permite

- Permite mayores niveles de abstracción, por ende capacidad de abordar problemas más complejos
- Diseñar programas más claros, y por ende más fáciles de mantener
- Permite la reutilización de código, y por ende evita duplicidad de trabajo y aumenta productividad
- La detección de errores y corrección se hace fácil

En esquema un programa dividido en módulos será:



Puesto que nuestra herramienta de programación va a ser el lenguaje C o C++, se hará énfasis en el uso de funciones.

Para comprender mejor el concepto, lo realizaremos a través de un ejemplo, práctico

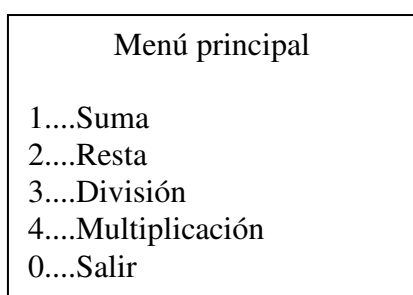
Ejercicio: Realizar un algoritmo que mediante un menú, permita realizar la cuatro operaciones básicas, suma, resta, división y multiplicación, terminando, permitiendo salir con el número 0.

Análisis

- 1.- ya sabemos como se hacen cada una de las operaciones,
- 2.- sabemos como se comportan los ciclos

3.- manos a la obra con funciones

Se pide un menú⁵



Solución

```

1 Algoritmo menú
2 Var
3 dato1, dato2, op: entero;
4 inicio
5   op=1;
6   Mientras (op<>0)
7     Escriba ("1...Suma");
8     Escriba ("2....Resta");
9     Escriba ("3....División");
10    Escriba ("4....Multiplicación");
11    Escriba ("0....Salir");
12    Lea (op);
13    Si (op=1)
14      Escriba ("Entre el primer numero");
15      Lea (dato1);
16      Escriba ("Entre el segundo número");
17      Lea (dato2);
18      Escriba ("El resultado de la suma es", suma (dato1, dato2);
19      Fin _ si
20    Si (op=2)
21      Escriba ("Entre el primer numero");
22      Lea (dato1);
23      Escriba ("Entre el segundo número");
24      Lea (dato2);

```

Llamado a la función suma, a la cual se le pasan dos **parámetros**

El resultado viene en el nombre de la función y se imprime de unas ves

⁵ Menú:


```

25     Escriba ("El resultado de la resta es", resta (dato1, dato2);
26     Fin _ si
27 Si (op=3)
28     Escriba ("Entre el primer numero");
29     Lea (dato1);
30     Escriba ("Entre el segundo número");
31     Lea (dato2);
32     Escriba ("El resultado de la división es", div(dato1,dato2);
33     Fin _ si

34     Si (op=4)
35     Escriba ("Entre el primer numero");
36     Lea (dato1);
37     Escriba ("Entre el segundo número");
38     Lea (dato2);
39     Escriba ("El resultado de la multiplicación es", mult (dato1, dato2);
40     Fin _ si
41 Fin _ mientras
42 Fin

43 Entero función Suma (entero: dato1, dato2)
44 Var
45 Respuesta: entero;
46 Inicio
47     Respuesta= (dato1+dato2);
48     Devolver (respuesta);
49 Fin.
50 Entero función resta (entero: dato1, dato2)
51 Var
52 Respuesta: entero;
53 Inicio
54     Respuesta= (dato1-dato2);
55     Devolver (respuesta);
56 Fin.
57 Entero función div (entero: dato1, dato2)
58 Var
59 Respuesta: entero;
60 Inicio
61     Respuesta= (dato1 div dato2);
62     Devolver (respuesta);
63 Fin.
64 Entero función Mult (entero: dato1, dato2)
65 Var
66 Respuesta: entero;
67 Inicio
68     Respuesta= (dato1 * dato2);
69     Devolver (respuesta);
70 Fin.

```

Se crea la función suma, esta es de tipo entero por que devuelve, un valor entero; recibe 2 datos, también de tipo entero que son los que se envían del programa principal, estos pueden tener o no el mismo nombre

Retorna la respuesta al programa principal

1.-Nota importante: hay dos tipos de paso de parámetros entre el programa y las funciones o entre las funciones: por parámetro o por valor

2.-Nota: en muchos lenguajes de programación existen funciones y procedimientos, invitamos a los estudiantes a que profundicen más en estos conceptos, mediante consulta en biblioteca o en sitios web,

3. TERCERA UNIDAD

Lenguaje de Programación C++

Introducción

Para finalizar el curso de algoritmos es importante llevar la teoría a la práctica, es decir convertir los algoritmos en verdaderos programas de computador, para lo cual se utilizará C++ como lenguaje base de nuestro trabajo. Para esta unidad es necesario que el estudiante tenga presente todos los conceptos tratados, a demás que los ejercicios realizados hasta el momento. Se ha tomado el lenguaje C++ por que? Porque incluye al lenguaje C y muchas otras funciones, a demás c++ permite el trabajo orientado a objetos, que se trabajarán en otros cursos del mismo programa, a demás se encuentran todas las estructuras de los demás lenguajes de programación, y se queda listo para migrar a programas basados en el mismo C++ como lo es Java. Para abordar esta unidad es importante que el estudiante ya haya tenido contacto con la computadora, sobretodo con editores de texto, lo que permitirá utilizar acciones de borrado, búsqueda, copias, pegar, que hacen que la digitación de los programas sean más rápidas y en consecuencia se dedicará más tiempo para el análisis de los mismos.

Los temas tratados a lo largo de esta unidad son:

1. Conceptualización
2. Ejecución
3. Estructura de un programa
4. Estructuras recontrol
5. Funciones

3.1. INTENCIONALIDADES FORMATIVAS:

Propósitos de la unidad

Desarrollar habilidades para realizar programas utilizando C++ como lenguaje de programación

Objetivos de la unida

Entender el desarrollo de un programa en C++

Aprender a codificar programas en C++

Manejar instrucciones básicas como condicionales, bucles y funciones

Competencias de la unidad:

El estudiante desarrolla habilidades de análisis, deducción, corrección de errores entre otras

Metas de aprendizaje

El estudiante es capaz de resolver pequeños problemas utilizando C++ como lenguaje de programación.

Motivar al estudiante para que explore nuevos entornos de programación

3.2 LENGUAJE DE PROGRAMACIÓN C++

Conceptualización: Para esta unidad se selecciono el lenguaje de programación C++, por ser uno de los más difundidos, a demás por su gran bibliografía, esto no quiere decir que no se pueda utilizar otros *compiladores*, como es el caso de C estándar , o turbo C; a demás se trabajará, bajo el supuesto que el sistema operativo es Windows 9x ; lo que no significa que no se puede trabajar bajo Linux, (C++ bajo Linux), donde se darán unas pautas para su trabajo.

Para iniciar tomaremos la siguiente lectura que servirá de base para esta unidad:

lectur

“Desde 1980 se utilizaron versiones anteriores del lenguaje C++ que se denominaban a sí mismas ‘C con clases’. Estas primeras versiones surgieron debido a que Bjarne Stroustrup⁶ tuvo necesidad de realizar simulaciones manejadas por eventos y el lenguaje con el que podría haber resuelto su problema (Simula 67) no era lo suficientemente eficiente por lo que decidió crear un lenguaje ad hoc con sus necesidades.



En este punto el diseñador tuvo que elegir cuál sería la manera de generar el nuevo lenguaje de programación:

Si utilizando como base un lenguaje conocido o empezando desde cero la implementación del mismo, por lo que decidió utilizar como base el lenguaje C debido a las siguientes características que lo hicieron atractivo: Es un lenguaje versátil, conciso y de nivel relativamente bajo, lo que lo hace adecuado para la mayoría de las tareas de desarrollo de sistemas además de que es un lenguaje muy portable y tiene cabida en el ambiente de programación UNIX; el C era un lenguaje ampliamente utilizado por muchos programadores y ya existían muchísimos sistemas implementados en él, además de que era un lenguaje lo suficientemente estudiado para que en ese momento se tuviese un conocimiento amplio sobre sus fortalezas y debilidades. Al mismo tiempo, como debía ser un lenguaje que pudiese utilizarse para la simulación de sistemas, incorpora muchas características de Simula 67 como el concepto de clase, las clases derivadas y las funciones virtuales.

⁶ Consultar más sobre Bjarne Stroustrup : <http://www.research.att.com/~bs/homepage.html>

C++, no solamente incorpora características de C y Simula 67. Por ejemplo:

De Algol68: se copia la capacidad para sobrecargar operadores y la libertad para poder hacer declaraciones en cualquier parte del código.

De Ada y parcialmente de ML se tomó el mecanismo para resolver excepciones y el recurso de patrones.

Por otra parte, el lenguaje C++ incorpora la reutilización de código, la cual consiste en que teniendo bloques de código (clases), estos pueden utilizarse en varias partes de un sistema o en distintos sistemas. Esto lo hace mucho mas conveniente que el lenguaje C y a su vez posibilita a los programadores el poder realizar sistemas de mayor tamaño y complejidad con menor esfuerzo. Inicialmente C++ nace como una herramienta generada para que el autor y sus amigos no tuviesen que programar en ensamblador o C, la cual debía permitirles hacer mas fácil y agradable la escritura de programas de buena calidad para el programador individual. De tal manera que no surge como un proyecto en forma, ni se generó un grupo de trabajo para diseñar C++, básicamente el lenguaje se enriquecía y se transformaba de acuerdo a las necesidades y sugerencias que los amigos del autor y algunos usuarios le hacían llegar.

Al inicio existieron muchas versiones 'no oficiales' del C++ y no fue sino hasta 1987 que se advirtió la necesidad de estandarizarlo de manera formal y para ello se hizo un esfuerzo de establecer comunicación entre los realizadores de compiladores de C++ y los principales usuarios mediante correo electrónico, correo convencional y reuniones en conferencias a cerca de C++. Dicho esfuerzo culmina con la creación del grupo X3J16 de ANSI que en 1989 se reunió para crear el estándar de C++ tal y como lo conocemos hoy en día.

3.2.1 Características Del Lenguaje C++

El lenguaje C++ incorpora todas las características de C en cuanto a:

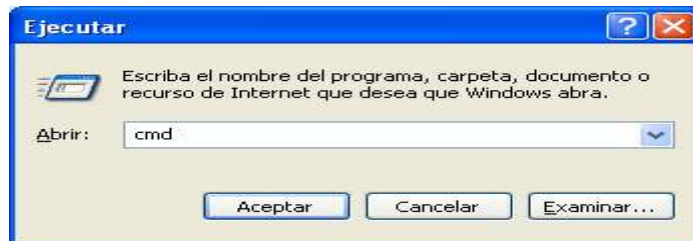
- Tipos de datos básicos
- Estatutos de control
- Arreglos
- Apuntadores
- Estructuras y Uniones
- Funciones definidas por el usuario
- Funciones integradas que permiten el fácil manejo de los recursos del sistema en bajo nivel
- Y además incorpora las siguientes características que permiten la programación orientada a objetos:
 - Abstracción de datos:
 - Encapsulamiento:
 - Herencia
 - Polimorfismo
 - Sobrecarga de funciones y operadores” 7

⁷ Artículo completo: <http://www.itlp.edu.mx/posgrado/lengprog/c.htm>

3.2.2 Ejecución del programa

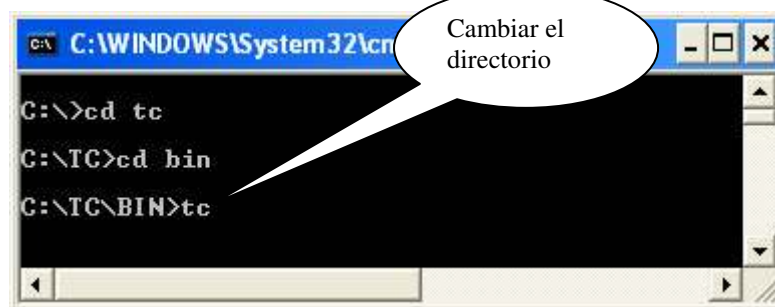
Una vez el programa este instando en su computadora, la ejecución es sencilla:

1.



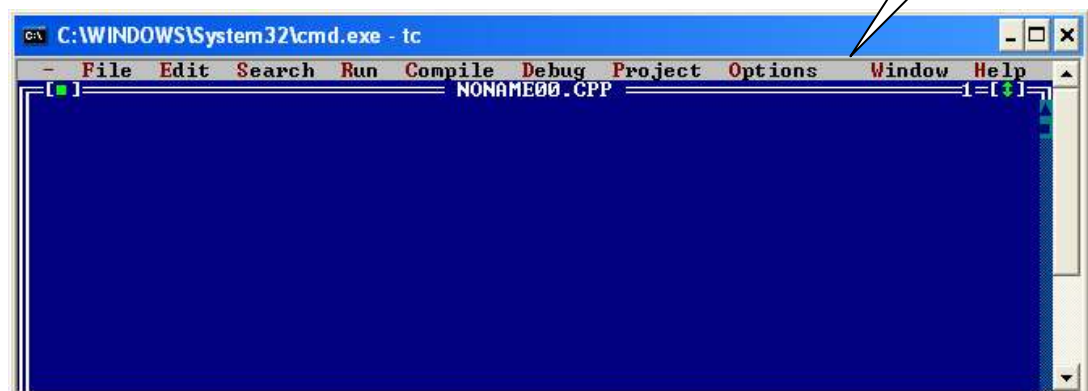
Desde Windows

2.



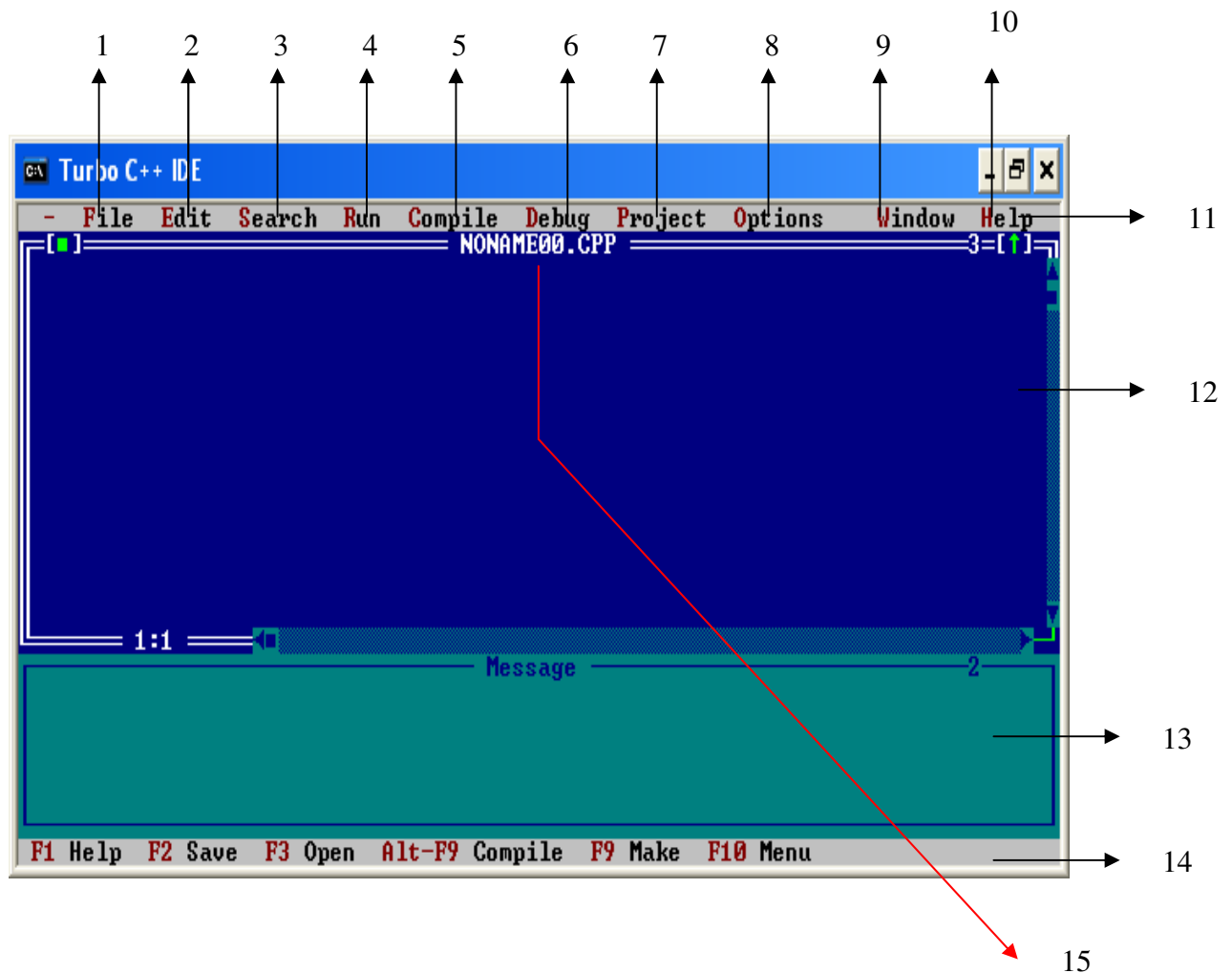
Editor de turbo
C++ ver. 3.0

3.



Si desean utilizar un compilador con múltiples funciones visuales, se recomienda visitar la página http://www.lcc.uma.es/~pedre/LP_DevC.htm y descargar el compilador **Dev-c++**

El Editor de turbo C++



- .- procesos de apertura cierre grabado....
- 2.- Procesos de edición Copiar, cortar, pegar....
- 3.-realizar búsquedas por diferentes criterios en un texto de programa
- 4.-opciones para correr los programas
- 5.-opciones para compilar los programas, una de las más importantes
- 6.-permite tener diversidad de parámetros para depurar programas
- 7.-opciones necesarias para crear proyectos desde cero
- 8.-permite configurar todo el entorno e inclusive opciones de trabajo en modo grafico
- 9.-cada programa se puede trabajar en ventanas independientes
- 10.-importante toda una ayuda de comando, funciones con ejemplos
- 11.-barra de menús
- 12.-espacio para escribir los programas, el editor propiamente dicho
- 13.-espacio donde aparecen o se configuran diversidad de ventas de apoyo al programa
- 14.-barra de ayudas y accesos rápidos
- 15.-nombre que toman los archivos

Más información ver anexo 10

Manos a la obra

Vamos a realizar un ejercicio practico y sobre el se explicaran cada una de las acciones

Ejercicio: tomaremos como base nuestro ejercicio clásico de la suma de dos números e imprimir su resultado

El análisis de este ejercicio y se conoce

Procedemos

- ❖ Se carga el editor del C++, para proceder a ingresar el programa, tomando como referente el algoritmo

The screenshot shows a C++ editor window titled "C:\WINDOWS\System32\cmd.exe - tc" with a menu bar (File, Edit, Search, Run, Compile, Debug, Project, Options, Window, Help) and a toolbar. The code in the editor is as follows:

```
#include<iostream.h>
#include<conio.h>

void main()
{
    int a,b,suma;
    clrscr();
    cout<<"por favor entre el primer Numero";
    cin>>a;
    cout<<"Por favor entre el Segundo Numero";
    cin>>b;
    suma=a+b;
    cout<<"el resultado de la suma es : "<<suma;
    cout<<"\n presione una tecla para continuar";
    getch();
}
```

Red boxes with labels point to the following parts of the code:

- Llamado a las Librerías:** Points to the `#include<iostream.h>` and `#include<conio.h>` lines.
- Llamado a la funcion Principal Main():** Points to the `void main()` line.
- Declaración de variables, en este caso de tipo entero:** Points to the `int a,b,suma;` line.
- Función para limpiar o clarear pantalla:** Points to the `clrscr();` line.
- Cut<< = Escriba Cin>> = lea:** Points to the `cout<<"por favor entre el primer Numero";` and `cin>>a;` lines.
- Función de tipo carácter que espera se presione una tecla:** Points to the `getch();` line.

Pasó 2: ejecución del programa:

Procedemos a compilar (ver su Glosario de términos), para este caso, hacemos clic en el menú compile y damos clic, si el programa no contiene errores, presionamos una tecla y precedemos a ejecutarlo (ver su Glosario: Interprete) Para lo cual damos clic en el menú Run y



seleccionamos la primera opción y ya podemos ingresar los datos solicitados

```

C:\WINDOWS\System32\cmd.exe - tc
por favor entre el primer Numero4
Por favor entre el Segundo Número4
el resultado de la suma es : 8
presione una tecla para continuar
  
```

dar

Explicación:

Como se puede cuenta, una vez el algoritmo este

bien diseñado es fácil llevarlo a un lenguaje de programación,

Cambios a tener en cuenta:

1.- en C++ es necesario hacer llamado a librerías o cabeceras de programa, para este caso se utilizan dos `<iostream.h>`, que permite el manejo de entrada y/o salida mediante dos objetos de flujo de datos `cout<<`⁸ y `cin>>`⁹ y `<conio.h>`, quien trae las funciones básicas como posicionamiento o limpieza de pantalla entre otras

2.- En el lenguaje de programación C++, trabaja de manera modular, es decir el programa principal también es un modulo, que inicia con la instrucción el llamado a la función `main()`¹⁰; en este caso utilizamos la directiva `void` para especificar que esta función no retorna valor, en caso que se la coloque este parámetro, al finalizar el programa tenemos que utilizar la directiva de retorno **`return 0.`**

3.-**`int`** palabra reservada que indica que las variables son de tipo entero, otros rangos utilizados comúnmente son:

Cuadro tipo de datos simples en C++

Tipo	Rango mínimo	Rango máximo
Char	0	255
Short	-128	127
Int	-32768	32767
Unsigned int	0	65535
Long	-2147483648	2147483637
Float	$3.4 \cdot (10^{-38})$	$3.4 \cdot (10^{38})$
Double	$1.7 \cdot (10^{-308})$	$1.7 \cdot (10^{308})$
Long double	$1.7 \cdot (10^{-308})$	$1.7 \cdot (10^{308})$

4.-`clrscr()`¹¹, llamado a función que permite limpiar pantalla

5.-`cout<<` y `cin >>`, mirar pie de pagina

6.-`getch()`: captura una entrada de tipo carácter, en el caso de emplearse sola permite realizar una espera, por eso el mensaje de presione una tecla para continuar

⁸ `<<console output>>` Objeto de flujo estándar de salida, que normalmente es por la pantalla de la computadora

⁹ `cin>>`: objeto de flujo de captura de datos

¹⁰ Función principal de todo programa en c++

¹¹ `Clrscr`: (clear scream)

7.-Existe una serie de códigos secuenciales o de escape, por ejemplo el utilizado en la línea **cout<<"\n Presione una tecla para continuar"**; en este caso indica que esa línea se ubica en una nueva posición, es decir una nueva línea

Otros caracteres secuenciales¹²

Código	Significado
\n	Nueva Línea
\r	Retorno de carro
\t	Tabulación
\v	Tabulación vertical
\a	Alerta sonora
\b	Retroceso de espacio
\f	Avance de pagina
\\	Barra inclinada inversa
\'	Comillas simple
\"	Comillas dobles
\?	Signo de interrogación
\000	Número octal
\xhh	Número hexadecimal

8.- Las llaves permiten abrir y cerrar segmentos de programas,

9.- Se emplea punto y coma al finalizar cada línea, para indicarle al compilador, que ahí termina la instrucción, se aconseja no emplear punto y coma, al inicio de las funciones, ni en los condicionales y tampoco en los ciclos

10.- es muy común que al momento de compilar el ejercicio, genere una serie de errores, el indica la línea y el tipo de error, el más común es el olvido del punto y coma

/*****

Ejercicio 2

Ahora vamos a realizar un ejercicio que involucre condicionales

Retomemos nuestro viejo compañero: realizar un programa que lea dos números y determine cual de ellos es mayor.

Análisis: el análisis de este ejercicio ya lo hemos realizado en temas anteriores

Solución

```
#include<iostream.h>
#include<conio.>
void main()
{
    int a,b;
    clrscr();
    cout <<"por favor ingrese un número ";
    cin>>a;
```

¹² Es necesario que se de una mirada a los codigos ASCII

```

    cout<<"por favor entre otro número"
    cin>>b;
    if (a>b)
        cout<<"el número Mayor "<<a
    else
        cout<<"El número mayor es "<<b;
    getch();
}

```

Explicación

1.-los condicionales se determinan por la palabra **if** que significa si y la instrucción **else** que significa en caso contrario, en este caso bajo el condicional solo existe una línea, por consiguiente no se hace necesario ni abrir ni cerrar la instrucción { }, lo que pasaría en el caso de que hubiera, mas de una instrucción ej:

```

If (a>b)
{
    ....
    ....
    ...
}

```

/*****

Ejemplo 2

Tomamos es mismo ejemplo pero agregamos la condición en el caso de que sean iguales

Solución

```

#include<iostream.h>
#include<conio.h>
void main()
{
    int a,b;
    clrscr();
    cout <<"por favor ingrese un número ";
    cin>>a;
    cout<<"por favor entre otro número"
    cin>>b;
    if (a==b)
        cout<<" Los números son Iguales";
    if (a>b)
        cout<<"el número Mayor "<<a;
    if (a<b)
        cout<<"El número mayor es "<<b;
    getch();
}

```

Explicación

Para comparar una igualdad hacemos uso del doble signo de igualdad (==), en el ejemplo if(a==b), *Razón*: un solo signo de igualdad significa asignación.

Si es del caso determinar un *diferente* lo hacemos if(a!=b) con el signo de admiración

```
//*****
```

Ejemplo 3

Leer dos número (a,b), si el número a es mayor que b, realizar la división de a entre b, y mostrar su resultado, en el caso de que sea el número a menor que b, entonces realizar su producto y mostrar su resultado, y en el caso de que sean iguales simplemente indicar que son iguales

Solución

```
#include<iostream.h>
#include<conio.>
void main()
{
    int a,b,r;
    clrscr();
    cout<<"por favor ingrese un número ";
    cin>>a;
    cout<<"por favor entre otro número"
    cin>>b;
    if (a==b)
        cout<<" Los números son Iguales";
    if (a>b)
    {
        r=a/b;
        cout<<"El número mayor es a y el resultado de la división es "<<r;
    }
    if (a<b)
    {
        r=a*b;
        cout<<" El número mayor es b y el resultado del producto la es "<<b;
    }
    getch();
}
```

Comentarios

Para este caso en los condicionales que tienen más de una línea se emplea llaves de apertura y de cierre lo que indica hasta donde va el condicional

3.2.3 ejercicios de verificación

1.-consultar: en sitios Web o en la bibliografía sugerida para este modulo, los siguientes ítems:

- Palabras reservadas(que son y para que se utilizan)
 - Mínimo 20 palabras reservadas
- Signos de Puntuación
- Librerías o archivos de cabecera
 - Mínimo 6
- Sentencias de control *switch*
- Errores frecuentes de Programación

2.- Analizar y codificar en C++ los siguientes ejercicios

- Diseñe un programa para la conversión una medida de metros a pies y pulgadas.
- Dado un carácter alfabético en mayúsculas, elabore un programa que imprima en pantalla su equivalente en minúscula (consulte el código **ASCII**).
- Hacer un programa para calcular el **IVA** de un valor digitado por el teclado, mostrar este resultado y el de sumar el **IVA** al valor digitado.
- Un banco ha solicitado se diseñe un programa que permita encriptar la información de las contraseñas (4 números) digitada por teclado hasta el servidor principal, utilizando el siguiente criterio, el primer numero se envía de ultimo, el segundo, de penúltimo, el tercer numero pasa a la segunda posición, el último pasa a ser primero: ejemplo

Ejemplo: Sea 7458, se debe enviar como 8547

- Haga un programa que convierta una medida de longitud en kilómetros a metros, centímetros, milímetros, pulgadas, yardas, millas y pies.
- Elabore un programa que convierta una medida de masa en toneladas a kilogramos, quintales, gramos, libras.
- Realice un programa que convierta unidades de fuerza en newtons a dinas.
- Elabore un programa que convierta una unidad de presión en pascales a bares.
- diseñe un programa que calcule el área de una cara de un cubo y su volumen.
- Elabore un programa que convierta una unidad de volumen en metros cúbicos m^3 a litros y centímetros cúbicos.
- Diseñe un programa que Lea dos puntos (x, y) y calcule la distancia entre ellos
-

- Elabore un preprograma que lea la hora y muestre por pantalla la hora un segundo después ejemplo

1:20:21 debe mostrar 1:20:22

1:59:59 debe mostrar 2:00:00

- Elabore un programa que lea tres valores diferentes y determine el mayor, el menor y el promedio.
- Elabore un programa que valide mediante un mensaje si una pareja (x, y) pertenece o no a la siguiente función: $y = 3x - 4$.

Ejemplo: la pareja $(2,2)$ si pertenece a esta función.

- Escribir un programa que permita determinar cuál es el ganador de la matricula de honor de entre 4 estudiantes. El algoritmo deberá hallar la nota definitiva de c/u de ellos (4 materias.) Si es mayor que 4.5 el alumno podrá aspirar a la matricula de honor, de lo contrario no.
- Diseñe un programa que determine si un año leído por el teclado es o no bisiesto.
- Escribir un programa para calcular la fecha del siguiente día a partir de una fecha digitada desde el teclado por el usuario (dd, mm, aaaa) e imprimirla. (tenga en cuenta los años bisiestos.)
- Escriba un algoritmo para la resolución de una ecuación de primer grado ($ax + b = 0$).
- Lea dos números por teclado y determine si uno es divisor del otro.
- Se lee un número de máximo tres dígitos (verifique que efectivamente sea de máximo tres dígitos) y se debe determinar si es un número capicúa, es decir, que leído de izquierda a derecha es igual que leído de derecha a izquierda. Por ejemplo: 727, 343, etc.
- Usted debe realizar un programa para un cajero automático, que dispone de billetes de todas las denominaciones existentes (2000, 5000, 10000, 20000,50000), de forma que se le indique una cantidad a pagar y determine cual es la combinación apropiada de billetes para formarla. Las cantidades que no se puedan lograr con estos billetes deben aproximarse adecuadamente.
- En un colegio se ha variado el sistema de calificaciones, por tanto se requiere un algoritmo que indique la valoración en letras cuando se tiene la nota en números, siguiendo la tabla mostrada a continuación

Nota Numérica	Valoración en letras
0.0 – 5.9	E
6.0 – 6.9	D

7.0 – 7.9	C
8.0 – 8.9	B
9.0 – 10.0	A

- En una multinacional se cuenta con tres departamentos de ventas, en los cuales los empleados devengan el mismo salario, sin embargo se tiene un incentivo de acuerdo al cual, si un departamento vende más del 50% del total de ventas se da una bonificación del 20% del salario a los empleados. Considerando el total de ventas como la suma de las ventas de los tres departamentos, indique cuánto devengarán los empleados de cada uno de los tres departamentos este mes.
- En una organización se tiene a los empleados agrupados por categoría, los de categoría 1 ganan \$20.000, los de categoría 2, \$15.000, los de categoría 3, \$10.000 y los de categoría 4, \$7.500. Se quiere un algoritmo que permita determinar cuanto debe pagarse a un empleado si se conoce el número de horas que trabajó durante el mes y la categoría a la que pertenece. Se sabe que a todos se les descuenta un 7.2% por concepto de salud, y si el salario total devengado (mensual) es menos de 1'000.000, se le da un subsidio del 15% sobre su salario mensual (sin descuentos).
- Se debe leer un número y determinar en que categoría se encuentra; se sabe que la categoría A, son los números entre 0 y 2 inclusive, la categoría B son los números entre 3 y 6 inclusive, la categoría C, los números 7 y 8, y la categoría D el número 9. (Adivinó, los números validos son entre 0 y 9).
- Se quiere determinar el valor de depreciación de un artículo en una empresa, se sabe que el valor de depreciación anual se determina dividiendo el valor de compra del mismo, entre el número de años de vida útil; la vida útil se determina de acuerdo a la clase de artículo, los edificios tienen 20 años, la maquinaria, muebles y enseres, 10 años, los vehículos 5 años y los computadores 3.
- En un concesionario de vehículos, se pagan las comisiones a los vendedores según el valor de la venta (ver tabla). Al final del mes se desea saber ¿Cuánto ganó un vendedor en total por todas las comisiones, si se sabe que hizo 4 ventas?

Valor de Venta	Comisión para el Vendedor
Hasta 10.000.000	2%
Más de 10 y Menos de 15 millones	4%
Mas de 15 millones	10%

- El encargado del planetario desea que se diseñe un programa para que al digitar el nombre del día indique el astro que dio origen a ese nombre. Recuerde los astros:

Nombre día	del Astro
Domingo	Sol
Sábado	Saturno
Viernes	Venus
Jueves	Júpiter
Miércoles	Mercurio
Martes	Marte
Lunes	Luna

- Realice un programa que calcule si un triángulo es isósceles, equilátero o escaleno dados sus tres lados A , B y C
 - Isósceles \Rightarrow dos lados iguales
 - Escaleno $\Rightarrow A \neq B \neq C$
 - Equilátero $\Rightarrow A = B = C$
- Con relación a sus ángulos un triángulo puede ser:
 - Rectángulo \Rightarrow Un ángulo recto
 - Acutángulo \Rightarrow 3 ángulos agudos
 - Obtusángulo \Rightarrow 1 ángulo obtuso

Elabore un programa que calcule si un triángulo es rectángulo, acutángulo u obtusángulo.

- Elabore un algoritmo que seleccione personal para un empleo con las siguientes características: mujeres adultas, solteras y que practiquen algún deporte.
- Elabore un programa que muestre el dígito que más se repite en un número de 5 cifras, en caso de no repetirse ninguno imprimir un mensaje que diga "no hay dígitos repetidos".
- El recargo por trabajar horas nocturnas en una empresa es del 70%, el recargo por trabajar festivos es del 100%, haga un programa que lea los días laboradas por un empleado, las horas nocturnas el valor de la hora normal laborada y calcule e imprima el sueldo a pagar junto con el nombre del empleado.
- Elabore un programa que tenga cuatro niveles de seguridad para un programa, si el usuario logra ingresar imprimir el mensaje "Bienvenido", en caso contrario imprimir "Error clave" y el nivel del error.
- A los profesores de cierta universidad se les paga por horas cátedra dictadas de 50 minutos, elabore un programa que lea el número de horas dictadas en un semestre siendo estas horas de 60 minutos y calcule el pago del semestre para el profesor

teniendo en cuenta que a los profesores se les cancela según su categoría:

A \$12.400=

B \$11.200=

C \$10.000=

D \$ 8.500=

Al final al profesor se le resta el 10% de retención en la fuente. El pago debe tomar en cuenta las fracciones de hora

3.2.4 c++, ciclos

Es necesario dar una al capítulo anterior donde se trabaja los algoritmos

3.2.4.1 Ciclo for

Mediante un *ejercicio* se explicara el funcionamiento del ciclo for,

Ejercicio

Realizar un programa que sume los 10 primeros números naturales e imprima su resultado.

Este ejercicio, esta resuelto en algoritmo, por ende no se realiza el análisis correspondiente

Solución

```
#include<iostream.h>
#include<conio.>
void main()
{
    Int k,suma=0;
    clrscr();
    for (k=1;k<=10;k++)
        suma=suma+k;
    cout<<"el resultado de la suma de los 10 números es " << suma;
    getch();
}
```

Explicación

for(k=1;k<=10;k++)-> este ciclo se divide en res partes principales

- 1.-la variable k tomoa un valor inicial de arranque, aunque c++, permite definir las variables en el mismo ciclo.

- 2.- $k \leq 10$; condición, de parada, para este caso que llegue a 10
- 3.- $k++$; incremento, decimos que queremos incrementar la variable k en pasos de 1; se puede utilizar en sentido inverso $k--$, es decir decrementos
4. Como dentro del ciclo, no hay sino una instrucción, entonces no se requiere apertura ni cierre de llaves

Ejercicio 2

Una pequeña variación al ejercicio anterior

Realizar la suma de 10 números cualesquiera e imprimir su resultado

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int k, numero, suma=0;
    clrscr();
    for (k=1;k<=10;k++)
    {
        cout<<"Por favor entre un número";
        cin>>numero;
        suma=suma+numero;
    }
    cout<<"el resultado de la suma de los 10 números es " << suma;
    getch();
}
```

Explicación:

En este caso, el ciclo si abre llaves, por tener más de una instrucción

Ejercicio

Realizar un programa que permita ingresar 10 números, de los cuales se debe sumar aquellos que son positivos y contar los que son negativos, imprimir los resultados

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int k, numero, suma=0, kn=0;
    clrscr();
    for (k=1;k<=10;k++)
    {
        cout<<"Por favor entre un número";
        cin>>numero;
```

```

        if (numero >=0)
            suma=suma+numero
        else
            kn++;

    }
    cout<<"el resultado de la suma de los números positivos es : "<< suma <<"\n";
    cout<<"la cantidad de números negativos ingresados es : "<<kn;
    getch();
}

```

Explicación

La sentencia kn++, remplace a k=k+1;

/*****

3.2.4.1.1 ejercicios de verificación

- 1.-codificar los algoritmos del taller propuesto en el ítem 2.3.2.1.1
- 2.-Profundizar y realizar ejemplos con sentencias de incrementos y decrementos
- 3.-Consultar la directiva de posicionamiento gotoxy(x,y), para darle ubicación y presentación a los programas

3.2.4.2 Ciclos while y do while

Sentencia **while**: esta sentencia de ciclo o bucle es muy sencilla pero muy potente, su estructura.

```

while (<condición>) <sentencia>
Puede ser también
While (condición)
{
    ----
    ----
}

```

Sentencia **do while**, este ciclo es muy utilizado cuando queremos realizar filtros¹³ y cuando deseamos que se permita el ingreso al ciclo al menos una vez

Ejemplo

¹³ Filtro: permitir el ingreso de datos dentro de un rango especificado

Se debe desarrollar un programa que permita ingresar las notas del curso de algoritmos, el programa debe terminar cuando la nota ingresada es cero (0), luego mostrar el promedio de las notas ingresadas, las notas ingresadas no deben ser negativos ni superiores a cinco

Análisis:

Mucho de análisis se realizó en la semana referente a ciclos, trabajado con algoritmos.

Con este ejemplo se utilizarán *dos tipos de ciclos*, uno para controlar las entradas de datos hasta que estas sean diferentes de cero y el otro ciclo que permita entrar únicamente valores mayores a cero y menores o iguales a cinco

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int k;
    float suma, nota, promedio;
    clrscr();
    suma=0; k=0;
    while (nota !=0)
    {
        do
        {
            cout<<"entre una nota";
            cin>>nota;
            if (nota <0) || (nota >5)
            {
                cout<<"Error. Inténtelo nuevamente";
            }
        }
        while (nota<0 ) || (nota >5);
        if (nota !=0 )
        {
            suma=suma+nota;
            k++;
        }
    }
    promedio=suma/k;
    cout<<"la cantidad de notas ingresadas son: "<<k<<"\n";
    cout<<"el promedio de las notas es de : "<<promedio;
    getch();
}
```

Explicación

1.- Los conectores lógicos en c++ se representan así

- El conector **O** con ||
- El conector **Y** con &&

2.-al finalizar el ciclo **do** se cierra llaves l con un **while**, el cual termina con punto y coma

3.2.4.2.1 ejercicios de verificación

1.-Cada grupo debe codificar el taller número 4, propuesto para ser resuelto mediante algoritmos

2.-la siguiente es una recopilación de ejercicios para ser codificados por los grupos colaborativos

- Realice un programa que imprima en pantalla el conjunto de los (n) múltiplos de un número entero (x) digitado por el usuario.
- Haga un programa que imprima en pantalla el conjunto de los divisores de un número entero (x) digitado por el usuario.
- Elabore un programa que calcule el mínimo común múltiplo (*m.c.m*) de dos números A y B , recuerde el *m.c.m.* como su nombre lo indica es el menor múltiplo común de dos o mas números. Ejemplo: sean los números 8 y 6.

$$m.c.m. (8, 6) = 24$$

- Al divisor común que es mayor que todos los divisores comunes de dos números (A, B) se le llama máximo común divisor (*m.c.d.*). Elabore un programa para calcular el *m.c.d.* de dos números. Ejemplo: sea 8 y 12 (investigue el algoritmo de Euclides).

$$m.c.d. (8,12) = 4$$

- Dos números son amigos, si cada uno de ellos es igual a la suma de los divisores del otro. Ejemplo: 220 y 284 son amigos por que,

284	220
1	1
2	2
4	4
71	5
142	10
	11
	20
	22
	44
	55
	110
220	284

- Elabore un programa que calcule si dos números son amigos o no.
- Elabore un programa que calcule el número de días que existen entre dos fechas. Tenga en cuenta que existen meses de 30 y 31 días y los años bisiestos.
- Calcule usando cada uno de los tres ciclos el valor de X^n .
- Calcule e imprima las tablas de multiplicar del 1 al 9 usando el ciclo while ().
- Calcule e imprima las tablas de multiplicar del 1 al 9 usando el ciclo haga..do .. while().
- Calcule e imprima las tablas de multiplicar del 1 al 9 usando el ciclo for(.....).
- Desarrolle un programa que impriman las siguientes series:
 - 1, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44.
 - 1, 4, 9, 16, 25, 36, 49, 64, 81.
 - 2, 4, 6, 8, 10,100.
 - -2, +4, -6, +10,100.
- Escriba un programa para calcular si un número es primo o no, recuerde que los números primos son aquellos que solo son divisibles por la unidad y por ellos mismos: ejemplo 5, 11, 17, etc..
- Calcular mediante un programa cuantos números primos existen entre 1 y un número **M** dado por el usuario.
- Escriba un programa que muestre el cuadrado de los números del 1 al **n**.
- Diseñar un programa para determinar la cantidad de mujeres y hombres que hay en un grupo de N estudiantes. Además se debe hallar el promedio de edad y de estatura del grupo. (el usuario digitará para cada integrante del grupo, su sexo, edad y estatura).
- Desarrolle un programa que permita seleccionar personal para un empleo de un total de **N** aspirantes. Los aspirantes deben cumplir las siguientes condiciones para ser aceptados:
 - Mayores de edad
 - Ser ingeniero titulado
 - Tener experiencia laboral

Al final el programa debe mostrar el *total* de aspirantes aceptados.

- Desarrolle un programa que permita calcular el valor de la *tangente* de un ángulo dado en grados usando la serie de *Taylor* del seno y del coseno.
- Diseñe un programa que calcule e imprima la suma de los números pares e impares comprendidos entre 1 y N.
- Leer N números y calcular el mayor sin importar que se repita.
- Leer N números y calcular el menor sin importar que se repita.
- Leer una serie de M números y mostrar al final cuantos son positivos.
- Calcular la suma de los cuadrados de los números comprendidos entre 1 y N.
- Leer N números y al final imprimir el promedio de estos.

El número de términos debe ser dado por el usuario (entre mayor sea el

- Calcular suma de los primeros 100 términos de la serie de Fibonacci.

- Elaborar un programa que convierta un número entero positivo, menor a 257 a sistema binario
- Elabore un programa que permita convertir un número entero positivo (máximo cuatro cifras) en su equivalente en sistema octal.
- Escribir un programa que halle el número de años bisiestos en un intervalo dado por el usuario (año bisiesto – sí es múltiplo de 4, pero sí es múltiplo de 100 deberá ser también múltiplo de 400).
- Dada tu fecha de nacimiento (mes, día, año) indicar cuantos días (exactos) han transcurrido desde ese año.
- Se deben leer números hasta que se digite 99 (el 99 no se debe contar), y determinar cuantos primos hay, y cuantos pares. (recuerde que estas dos condiciones no son exclusivas).
- Elabore un programa que genere un número aleatorio y que les dé la posibilidad a dos jugadores de adivinar dicho número, el algoritmo debe pedir el número de partidas, intercalar los turnos para adivinar, mostrar el ganador por partida y el ganador final. El número debe estar entre 0-100. (Use la función random.)
- Elabore un programa que lea las ventas de (n) número de vendedores, para los productos (A, B, C, D y C), si los precios de los productos son (\$1000, \$2345, \$3876, \$1235 y \$550) respectivamente, calcule el número individual y total de productos vendidos, las ventas totales por producto, el promedio total de ventas, el producto mas vendido, el menos vendido, el vendedor que más ventas realizó.
- Realice un programa que calcule la suma de (n) números, el producto de estos y cuantos de estos son negativos y cuantos positivos. Estos datos deben ser mostrados por pantalla.

3.1.5. Funciones

3.1.5.1 Funciones incorporadas

Las Funciones se incorporan al lenguaje de programación C o C++ por medio de la Librerías: La principal estrategia de la programación estructurada al resolver un problema complejo es la de dividirlo en subproblemas (divide y vencerás) cuya resolución sea mucho más sencilla. Estos subproblemas se pueden dividir a su vez en otros más pequeños, y así sucesivamente, según la conveniencia. Esta estrategia también se llama *diseño descendente*, debido a que se parte de lo general y se diseñan soluciones específicas para sus subproblemas. Estos subproblemas los podemos implementar en el lenguaje C o C++ mediante la codificación de funciones. (ver anexo índice de Funciones),

El siguiente programa emplea funciones trigonométricas contenidas en el archivo de cabecera "math.h"

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main(){
    double angulo = 0.0;

    //real de doble precisión, 8 bytes = 64 bits

    cout << "Pi = " << M_PI;

    cout.precision(7); // se formatean los números con 7 decimales
    cout.setf(ios::fixed); // se utiliza notación fija en números
    cout << "\n\nSeno    (" << angulo << "°) = " << sin(angulo * M_PI/180);
    angulo += 30.0;
    cout << "\nCoseno   (" << angulo << "°) = " << cos(angulo * M_PI/180);
    angulo += 30.0;
    cout << "\nTangente (" << angulo << "°) = " <<
        sin(angulo*M_PI/180) / cos(angulo*M_PI/180);
    angulo += 30.0;
    cout << "\nCotangente(" << angulo << "°) = " <<
        cos(angulo*M_PI/180)/sin(angulo*M_PI/180);
    angulo -= 30.0;
    cout << "\nSecante   (" << angulo << "°) = " << 1/cos(angulo * M_PI/180);
    angulo -= 30.0;
    cout << "\nCosecante (" << angulo << "°) = " << 1/sin(angulo * M_PI/180);
    cout << "\nPi = " << M_PI;
    cout.precision(1); // se formatean los números con 1 decimal
    cout.setf(ios::scientific); // se utiliza notación científica
    cout << "\nPi = " << M_PI;
    cout << "\n\n Digite cualquier tecla y terminar...";
    getch();
}
```

El siguiente programa permite oír notas musicales entre 260 Hz y 520 Hz, a través de la utilización de funciones incorporadas en <dos.h>: sound() y nosound().

```
#include <iostream.h>
#include <dos.h>
```

```
const Tempo=1000; // aproximadamente 1000 milisegundos
```

```
void main(){
```

```
    sound(260); cout << "Do "; delay(Tempo);
    sound(290); cout << "Re "; delay(Tempo);
    sound(322); cout << "Mi "; delay(Tempo);
    sound(342); cout << "Fa "; delay(Tempo);
    sound(390); cout << "Sol "; delay(Tempo);
    sound(440); cout << "La "; delay(Tempo);
    sound(494); cout << "Si "; delay(Tempo);
    sound(520); cout << "Do "; delay(Tempo);
```

```
    nosound();
    cout << "\nSilencio\n";
    delay(Tempo);
```

```
    sound(260); cout << "Do "; delay(Tempo/2);
    sound(290); cout << "Re "; delay(Tempo/2);
    sound(322); cout << "Mi "; delay(Tempo/2);
    sound(342); cout << "Fa "; delay(Tempo/2);
    sound(390); cout << "Sol "; delay(Tempo/2);
    sound(440); cout << "La "; delay(Tempo/2);
    sound(494); cout << "Si "; delay(Tempo/2);
    sound(520); cout << "Do "; delay(Tempo/2);
```

```
    nosound();
    cout << "\nSilencio\n";
    delay(Tempo/2);
```

```
    sound(520); cout << "Do "; delay(Tempo/4);
    sound(494); cout << "Si "; delay(Tempo/4);
    sound(440); cout << "La "; delay(Tempo/4);
    sound(390); cout << "Sol "; delay(Tempo/4);
    sound(342); cout << "Fa "; delay(Tempo/4);
    sound(322); cout << "Mi "; delay(Tempo/4);
    sound(290); cout << "Re "; delay(Tempo/4);
    sound(260); cout << "Do "; delay(Tempo/4);
    nosound();
```

```
}
```

Observación: funciona con el speaker del computador, no con la tarjeta de sonido

3.1.5.2.-Nuestras Propias Funciones

Las funciones son bloques de instrucciones que tienen por objeto el alcanzar un resultado que sustituirá a la función en el punto de invocación (las funciones devuelven un resultado).

Cada función se evoca utilizando su nombre en una expresión con los argumentos actuales o reales encerrados entre paréntesis.

Para hacer una referencia a una función se invoca mediante un nombre y en caso de existir, una lista de parámetros actuales necesarios (argumentos). Los

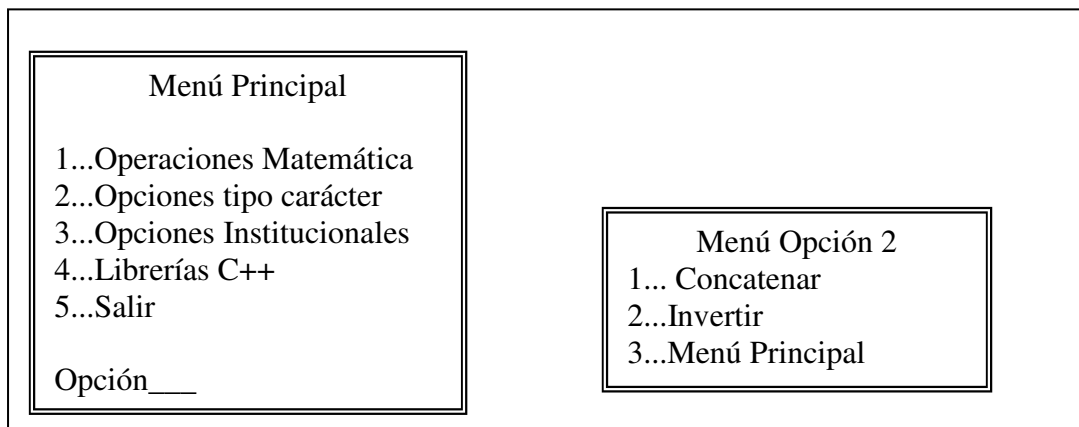
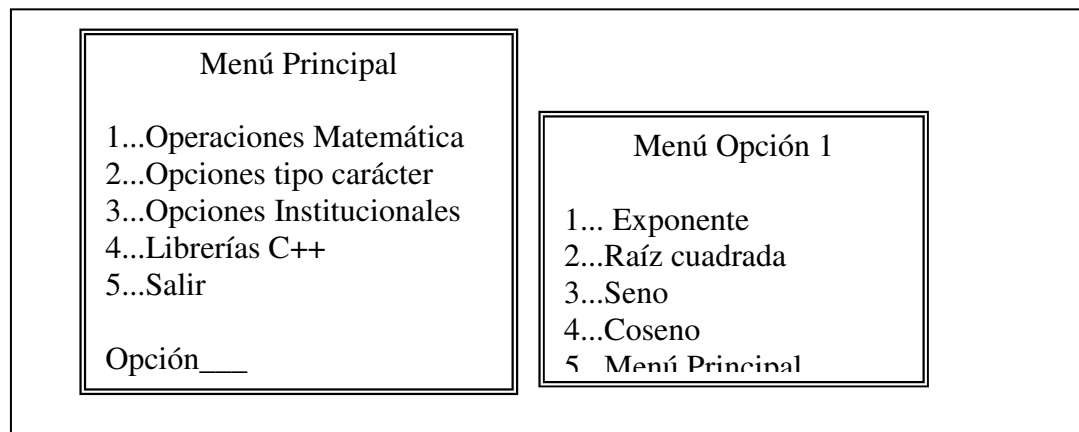
argumentos deben coincidir en cantidad, tipo y orden con los de la función que fue definida. La función devuelve un valor único.

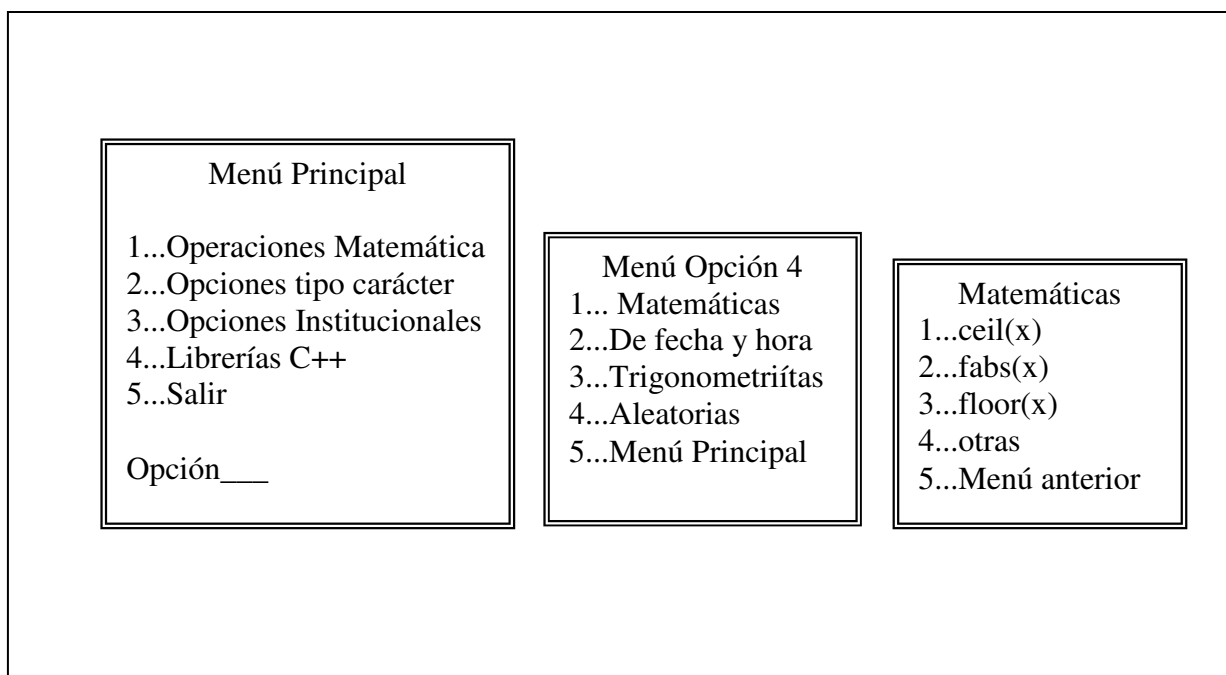
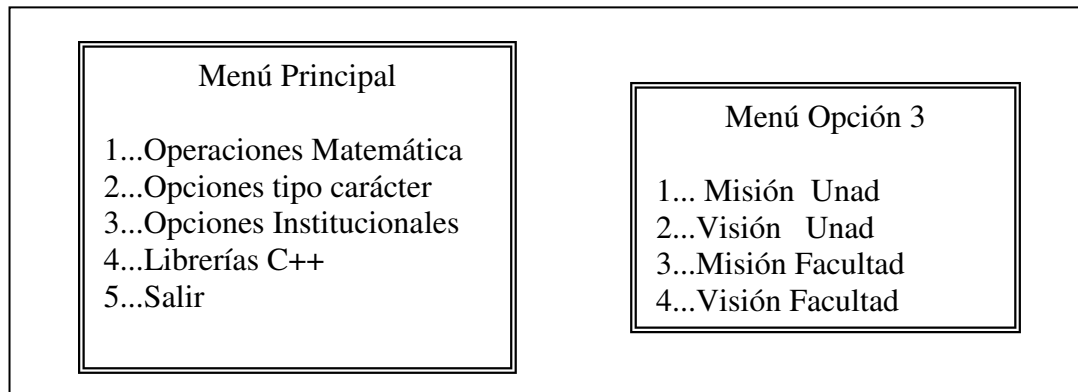
Las funciones a que se hace referencia, se conocen como funciones de usuario puesto que son definidas por él mismo y permiten su uso en forma idéntica a las funciones estándares. Para coordinar e iniciar el procesamiento, se utiliza un módulo principal que es colocado al final del algoritmo.

Para entender mejor la construcción de funciones, se propone un ejemplo práctico

Ejercicio

Realizar un menú que permita el manejo de submenú, para lo cual se realizara la primer parte del ejercicio y usted debe complementar el resto, para tener una idea de lo que se pretende, se visualizara los menús así



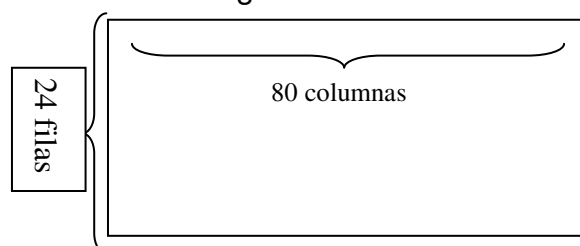


Análisis

Se puede observar que es un programa con un mayor grado de complejidad, por manejar una serie de menús y submenús

1.-a simple vista se puede observar que cada uno de los menús y submenús está contenido dentro de un marco, por consiguiente se puede pensar que una de las primeras funciones a realizar es el cuadro o marco contenedor, por lo cual se debe tener en cuenta que:

Un monitor normal en formato texto está dividido en filas y columnas, 24 filas por 80 columnas, por consiguiente es importante saber posicionar el cursor en estos rangos



- 2.-En el programa principal únicamente se controlara el menú principal
- 3.-que cada función de acuerdo a lo establecido en el estándar c++, se debe declarar como prototipo (en la cabecera se declara igual a como se realizara)

Para este ejemplo se creará únicamente la función de cuadro, el menú principal y el submenú 1, para que se pueda posicionar en cualquier lado de la pantalla

Solución

```
#include<iostream.h>
#include<conio.h>
void cuadro(int,int,int,int);//Prototipo función cuadro
void opciones1();//prototipo primer submenú
void main()
{
    int op=0;
    while (op!=5)// condicional menú principal
    {
        clrscr();
        cuadro(5,5,40,15);//llamado a la función cuadro en determinadas posiciones
        gotoxy(12,6);
        cout<<"MENU PRINCIPLA";
        gotoxy(6,8);
        cout<<"1...Operaciones Matematicas";
        gotoxy(6,9);
        cout<<"2.. Opciones tipo carácter";
        gotoxy(6,14);
        cout<<"5...Salir";
        cin>>op;
        if (op == 1)
        {
            opciones1 ();// llamado a la opción del primer submenú
        }
    }
}

// Final programa o función principal

void cuadro(int x,int y,int x1,int y1) //Construcción de la funcion cuadro
{
    int i,j;
    for(i=x; i <=x1;i++)// ciclo para mover el eje de las x
    {
        gotoxy(i,y);
        cout<<"="; // carácter ASCII 205
        gotoxy(i,y1);
        cout<<"=";
    }

    for(i=y; i<=y1;i++)// ciclo para mover ele eje de las y
    {
        gotoxy(x,i);
        cout<<"|"; //carácter ASCII 186
        gotoxy(x1,i);
        cout<<"|";
    }
}
```

Variables que toman los valores
constantes enviados de diferentes
lugares del programa

```

}
void opciones1()
{
    int op1;
    while (op1!=5)
    {
        cuadro(45,10,70,19);
        gotoxy( 50,11);
        cout<<"Menú opción 1";
        gotoxy(46,12);
        cout<<"1...Exponente";
        gotoxy(46,13);
        cout<<"Raiz cuadrada";
        if (op1==1)
        {
            //sentencias para ser complementadas por usted.
        }
    }
    getch();
}

```

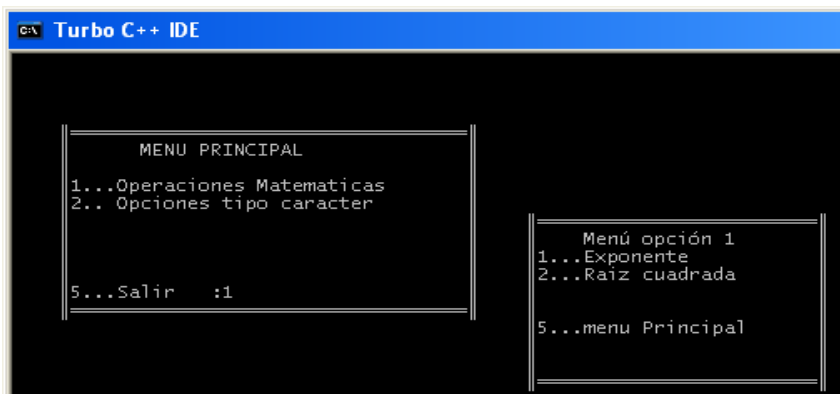
Explicación

- 1.-Es importantes que se lean los comentarios que se escriben al frente de cada línea del programa, estos comentarios enastan resaltados con negrita
- 2.- Con un poco de ingenio se puede mejorar el cierre de la líneas de los cuadros (utilizar códigos ASCII)

3.1.5.3 ejercicios de verificación

1- Complementar el ejercicio

Posible salida



2.-

- Diseñe una función para calcular X^n donde X sea un número real y n un número entero positivo o negativo.
- Desarrolle un programa que mediante funciones pueda calcular la suma, resta, multiplicación y división de dos números reales **A**, **B** y **C**. El algoritmo debe tener el siguiente menú de opciones.

Menú
1. Suma 2. Resta 3. Multiplicación 4. División 5. Salir

Cada función debe tener como parámetros tres números reales y debe retornar el resultado en un número real.

- Elabore un programa que calcule mediante dos funciones el seno y el coseno de un ángulo dado en grados usando la serie de *Taylor*.
- Diseñe una función que lea **N** notas y que calcule el promedio de estas. (el parámetro que se pasa a la función es **N** y regresa el promedio).
- Desarrolle un algoritmo que use una función para calcular el mayor de tres números.
- Diseñe un programa para calcular el volumen de un cilindro usando una función que recibe como parámetros el valor del radio y la altura y retorna el volumen.
- Diseñe un programa que tenga un menú de opciones para realizar conversiones de monedas usando funciones para efectuar los cálculos.

Menú
1. Pesos a Bolívares 2. Pesos a Dólares 3. Dólares a Euros 4. Pesos a Euros 5. Salir

- Diseñe un Programa para calcular el máximo común divisor de dos números enteros positivos usando una función que reciba como parámetros los dos números enteros y que retorne el **mcd**.
- Determine en una función el valor de **PI**, donde la función recibe el número de términos deseados.
- . Empleando el combinatorio para realizar un programa que genere el triángulo de Pascal

3.- Si el grupo colaborativo así lo desea puede consultar a cerca de las funciones graficas en C++, (es un buen momento para hacerlo)

Evidencia: el proyecto se sustentará en la semana 18

FUENTES DOCUMENTALES

INSUASTY R, Luis Delfín, Guía "A","B","C","D" de aprendizaje autonomo. Bogotá Colombia, UNAD- Cafan

MAURREN, Priestley . Tecnicas y estrategias del pensamiento critico. Mexico D.F. 1996 (reimp .2000). Trillas.

ARCEO B, Frida y Otro. Estrategias Decentes Para un Aprendizaje Significativo. Mexico D,F 1999. McGRAW-HILL

KENNETH C, louden . Lenguajes de programación (segunda edición). Mexico D.F 2004. Thomson

AGUILAR, Luis. Fundamentos de programación, algoritmos y estructura de datos (segunda edición). España. McGRAW-HILL.

AGUILAR, Luis. Fundamentos de programación, algoritmos, estructura de datos y Objetos (tercera edición). España. 2003. McGRAW-HILL.

DEYTEL Y DEYTEL. Como programa C++(segunda Edición). Mexico D.F. 1999. Prentice Hall. McGRAW-HILL

FARREL, Joyce, introducción a la programación lógica y diseño. Mexico D.F 2000. Thomson

BROOKSHEAR, J. Glenn , Introducción a las ciencias de la Computación (Cuarta Edición). Edición Española 1995. Addison-Wesley Iberoamericana

Sitios WEB

http://www.geocities.com/david_ees/Algoritmia/curso.htm (Curso de algoritmia)

<http://www.ilustrados.com/publicaciones/EpZVVEZpyEdFpAKxjH.php>
(Lenguajes de Programación)

<http://www.ilustrados.com/buscar.php> (Algoritmos)

<http://www.inf.utfsm.cl/~mcloud/iwi-131/diapositivas.html> (Algoritmos)

<http://www.ucsm.edu.pe/rabarcaf/vonuep00.htm> (Diccionario académico)

<http://www.funlam.edu.co/bired/index.asp-offset=0.htm> (Aprendizaje Autonomo)

www.markelo.f2o.org (acertijos)

http://www.programas-gratis.net/php/descarga.php?id_programa=1808&descargar=DFD%201.0, (Descarga de dfd)
<http://www.itq.edu.mx/vidatec/espacio/aisc/ARTICULOS/leng/LENGUAJESDEPROGRAMACION.htm>

www.microsoft.com (Productos de programación)

www.borland.com (Productos de programación)

ANEXOS

ANEXO 1 – MAPA CONCEPTUAL

El Mapa Conceptual es una herramienta cognitiva que permite representar el conocimiento (ideas y asociaciones) de una manera gráfica y sintética, orientado al aprendizaje eficiente y Significativo.

Este instrumento educativo fue ideado por Joseph Novak en la década del 60, como una forma de poner en práctica las teorías de David Ausubel sobre Aprendizaje Significativo, es por ello que en la construcción de mapas conceptuales se enfatiza la importancia del conocimiento anterior para ser capaz de aprender nuevos conceptos en forma de proposiciones. Novak concluyó que "el aprendizaje Significativo implica la asimilación de nuevos conceptos y proposiciones en las estructuras cognitivas existentes".

La elaboración de mapas conceptuales permite la utilización de ambos hemisferios del cerebro, potenciando con ello los procesos del pensamiento abstracto y los psicomotrices, de manera que se complementan, sin olvidar que éstos fomentan también el desarrollo de la memoria, la reflexión, el espíritu crítico y la creatividad.

La construcción de mapas conceptuales permite diseñar un ambiente de aprendizaje donde se estimula no sólo la representación del conocimiento, sino también información textual y / o adicional que se organiza jerárquicamente. De esta forma el mapa conceptual puede ser utilizado con diferentes propósitos¹⁴

Como hacer un mapa conceptuales¹⁵

1. – En la medida que se lea debe identificarse las ideas o conceptos principales e ideas secundarias y se elabora con ellos una lista.

¹⁴ <http://www.educarchile.cl/eduteca/todounmundo/acti/mapa.htm>

¹⁵ <http://www.monografias.com/trabajos10/mema/mema.shtml>

2. - Esa lista representa como los conceptos aparecen en la lectura, pero no como están conectadas las ideas, ni el orden de inclusión y derivado que llevan en el mapa. Hay que recordar que un autor puede tomar una idea y expresarla de diversas maneras en su discurso, para aclarar o enfatizar algunos aspectos y en el mapa no se repetirán conceptos ni necesariamente debe seguirse el orden de aparición que tienen en la lectura.

3: - Seleccionar los conceptos que se derivan unos de otros.

4. - Seleccionar los conceptos que no se derivan uno del otro pero que tienen una relación cruzada

5.- Si se consiguen dos o más conceptos que tengan el mismo peso o importancia, estos conceptos deben ir en la misma línea o altura, es decir al mismo nivel y luego se relacionan con las ideas principales.

6. - Utilizar líneas que conecten los conceptos, y escribir sobre cada línea una palabra o enunciado (palabra enlace) que aclare porque los conceptos están conectados entre sí.

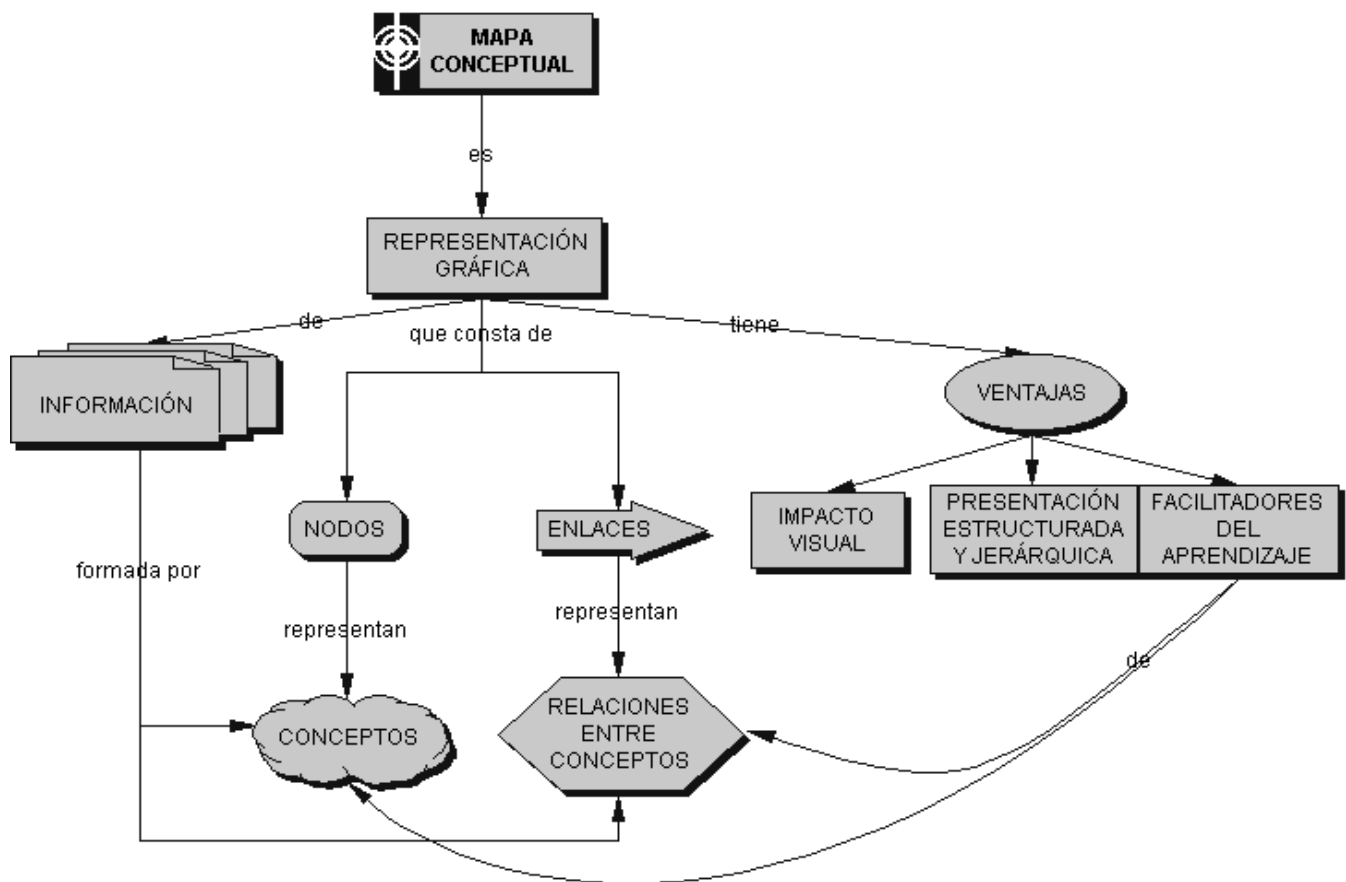
7. _ Ubicar las imágenes que complementen o le dan mayor significados a los conceptos o proposiciones

8. -. Diseñar ejemplos que permitan concretar las proposiciones y /o conceptos

9- Seleccionar colores, que establezcan diferencias entre los conceptos que se derivan unos de otros y los relacionados (conexiones cruzadas)

10. - Seleccionar las figuras (óvalos, rectángulos, círculos, nubes) de acuerdo a la información a manejar.

11. - El siguiente paso será construir el mapa, ordenando los conceptos en correspondencia al conocimiento organizado y con una secuencia instruccional. Los conceptos deben ir representados desde el más general al más específico en orden descendente y utilizando las líneas cruzadas para los conceptos o proposiciones interrelacionadas.



Observación: Existen también software que permite desarrollar mapas conceptuales, que permiten explorar nuevas alternativas, se sugiere visitar realizar consultas en Internet y descargar, diferentes versiones.

ANEXO2 GUIA PARA LA CONSTRUCCION DE PORTAFOLIO

Se tomará compilación realizada por Julio Roberto Sanabria. De la universidad Católica del norte www.ucn.edu.co

1. Concepto:¹⁶

El Portafolio ha sido concebido como la colección sistemática y organizada del material que un estudiante produce durante la semana de acuerdo con las metas establecidas por él y las especificaciones propias del curso que esté realizando, con el propósito de monitorear su progreso en cuanto a conocimientos, habilidades cognitivas e interpersonales, actitudes y motivaciones; de poner en evidencia la calidad de los procesos, los aprendizajes y los productos realizados; y, de evaluar tanto los procesos, los aprendizajes y los productos llevados a cabo como los materiales producidos con el fin de identificar avances y necesidades, y proponer las acciones de seguimiento pertinentes.¹⁷

En síntesis el Portafolio es una carpeta personal, dedicada al archivo de todas las evidencias de aprendizaje, referidas a las facilidades, dificultades o simplemente el registro de experiencias asociadas al acto de aprender (resúmenes, mapas conceptuales, cuadros, gráficas). Es una técnica que permite a la persona aprender a evaluar su desempeño en término de progresos o errores.

Los OBJETIVOS del Portafolio son: Facilitar el ejercicio de sistematización de experiencias. Aumentar la capacidad de reflexión sobre los textos leídos, así como un monitoreo de su trayectoria como estudioso(a). Desarrollar habilidades de escritura. Abrir un lugar para ejercitar su creatividad.

2. Atributos:

- ❑ “Es un regulador de procesos de aprendizaje”.
- ❑ “Contiene material básico para la Investigación Formativa”.
- ❑ “Permite la autoevaluación como elemento fundamental en el proceso formativo del aprendizaje autónomo”.

¹⁶. Ideas tomadas de la documentación orientadora para la Especialización en Pedagogía para el Desarrollo de Aprendizaje Autónomo y los aportes de los miembros del Grupo Medellín 2. (25-Jun-00)

¹⁷. Guía de Aprendizaje Autónomo “A”. UNAD-CAFAM. Pág. 31.

- ❑ “Es un dispositivo pedagógico de acompañamiento y regulación de disciplina”.
- ❑ “Es una herramienta de reflexión sobre los logros y los aspectos a mejorar en un proceso de aprendizaje”.
- ❑ “Evidencia el crecimiento personal e intelectual de su autor”.
- ❑ “Permite precisar y jerarquizar el conocimiento”.
- ❑ “Como dispositivo pedagógico facilita el proceso de metacognición y la construcción de conocimientos”.
- ❑ “Es un espacio para la articulación de la teoría y la práctica”.
- ❑ “Sirve como medio para realizar la evaluación del cambio cognitivo del estudiante”.
- ❑ “Permite obtener conciencia del proceso metacognitivo realizado por el estudiante”.
- ❑ “Es un elemento de consulta permanente que facilita la reflexión sobre los procesos de aprendizaje”.
- ❑ “Es la evidencia de la investigación educativa centrada en la experimentación, evaluación y planes de acción como resultado de la práctica”.
- ❑ “Permite establecer conductas de trabajo individual y cooperativo a partir de las actitudes individuales”
- ❑ “Es facilitador del aprendizaje a través de procesos formativos que posibilitan construcciones propias del aprendizaje”.
- ❑ “Es un medio integrador de la teoría y la práctica pedagógica en el escenario particular del aprendiente”

3. Organización del Portafolio:

La organización del portafolio comprende cuatro tareas, a saber:

- ❑ La definición del propósito del mismo.
- ❑ La selección del material de acuerdo con el propósito.
- ❑ La reflexión de los resultados intrínsecos y extrínsecos del trabajo de la semana y
- ❑ La proyección de metas futuras de aprendizaje.

La decisión sobre el material que se ha de coleccionar depende en gran medida de la claridad con que se defina su propósito.

4. ¿Qué guardar en el Portafolio?

- Los registros de lectura de cada semana.
- El Proyecto Personal de Aprendizaje (PPA) desarrollado para cada ocasión que deba ejercer el papel de SUSTENTADOR.
- Las notas de lo que discuta con sus compañeros en el Pequeño grupo.
- La página de autoevaluación que realizará cada semana. Es un pequeño texto con su respectivo título.

- Los trabajos o pruebas que se hagan en individualmente o en Pequeño Grupo.
- Anotaciones opcionales a manera de Diario de Procesos.
- Las observaciones del asesor

El Portafolio promueve un clima de reflexión. Surgirá el conflicto entre lo que ve y lo que realmente desearía que fuera. Piense, escoja y tome el riesgo de realizar preguntas sobre lo que ha causado una disonancia cognoscitiva. Realice preguntas que no haya podido resolver sobre el tema de la semana sobre sus procesos de aprendizaje o de todo lo que crea que permite una mayor comprensión de los procesos, someta todas estas inquietudes al trabajo en el pequeño grupo.

Es recomendable diligenciar un Portafolio por cada asignatura que se esté cursando, con el claro propósito de:

- ❑ Compilar los materiales producidos individualmente o en forma colectiva.
- ❑ Poner en evidencia el progreso en términos de calidad de los materiales producidos.
- ❑ Demostrar el avance en su desarrollo personal e interpersonal.

En principio se seleccionarán y archivarán:

- ❑ Los materiales que el autor del Portafolio produce semanalmente como requisito del curso o actividad académica en que se encuentre. Estos pueden ser manuscritos, pero claros.
- ❑ Los materiales que resulten de la acción pedagógica del autor durante la aplicación y el ensayo del aprendizaje autónomo.
- ❑ Los exámenes y otras pruebas.
- ❑ Las instrucciones relacionadas con los trabajos especiales de la semana.

La esencia de un portafolio es la reflexión que hace su autor sobre cada elemento que lo constituye. *“Una compilación de materiales sin reflexión es un archivo, más no un Portafolio”*¹⁸

La preparación de un Portafolio, tal como se acaba de describir, se convierte en una valiosa estrategia de desarrollo de la metacognición, por cuanto los participantes toman conciencia de sus propios procesos de pensamiento y aprendizaje y se convierten en gestores de su propio desarrollo personal y profesional.

Varios autores sugieren que la organización del Portafolio, refleja hasta cierto grado la personalidad y carácter de su autor. Una tabla de contenido, al comienzo de cada semana, le ayudará a poner las cosas en orden.

¹⁸ . Ibid. Pág. 32

El éxito del Portafolio depende del cuidado con que se produzcan, organicen, consulten y comparen los materiales semana tras semana, a fin de descubrir patrones de desarrollo y de proyectar líneas de acción para aprendizajes futuros.

Cuando los procesos de aprendizaje, se acompañan de un dispositivo pedagógico —como el Portafolio— se permite el monitoreo del curso de acción del pensamiento y resultan mejores posibilidades para facilitar la reflexión y la autorregulación de la experiencia.

5. Valoración del Portafolio:

La evaluación y calificación del Portafolio es responsabilidad de su autor. Lo importante no es calificar la compilación, sino la dedicación y calidad con que se ha procedido en cada semana. En verdad, lo que el participante debe valorar en el Portafolio es la responsabilidad consigo mismo y el respeto por lo demás, antes que el número de páginas archivadas.

La calificación del Portafolio depende de tres criterios:

- ❑ ¿Está completo? Es decir, ¿contiene todos los materiales y artefactos que deben coleccionarse?
- ❑ ¿Está organizado? Es decir, ¿tiene una estructura y todos sus componentes están debidamente identificados, ordenados, categorizados y limpios, de suerte que un lector pueda recorrer fácilmente todas sus secciones y obtener la información que necesita?
- ❑ Al observar los contenidos de la semana actual y compararlos con los de la semana anterior, ¿ayudó a medir su progreso y a producir materiales más imaginativos y más creativos? ¿Cuáles?

Cada estudiante es el veedor del Portafolio de un Compañero quien a su vez será el veedor del primero. Esta función es coevaluar el Portafolio de su compañero y corresponsabilizarse de la calificación que se asigne.

6. Una Guía para autoevaluar el Portafolio:¹⁹

El Portafolio puede ser autoevaluado de muchas maneras: con guías estructuradas o semiestructuradas, con esquemas que preguntan qué tiene o de qué carece.

¹⁹ . Un aporte de Jairo Gutiérrez, Emerson Mosquera y Julio Roberto Sanabria. UNAD. Medellín. 1999

Para efectos de este documento se presentan algunas preguntas que pretenden propiciar la reflexión alrededor de dos grandes inquietudes para su autor: “*Para qué se hace*” y “*Para quién se hace*”. Suponemos que de sus respuestas dependen las características del “*Cómo hacerlo*”.

Consideremos entonces, los siguientes cuestionamientos:

1. ¿En qué forma el Portafolio ha sido útil para el ejercicio práctico y pedagógico del Aprendizaje Autónomo?
2. ¿Qué enseñanzas le han aportado los Portafolios de otros compañeros?
3. ¿En qué consisten los registros que Usted ha llevado al Portafolio y para qué?
4. ¿Cómo podría demostrar su capacidad para articular sus experiencias y las teorías mediante el uso y diligenciamiento del Portafolio?
5. ¿Cómo se presenta o se desarrolla la reflexión que usted hace sobre los contenidos de su Portafolio?
6. ¿En qué forma ha realizado las llamadas “Biografías de los trabajos”, para su Portafolio?
7. ¿Cómo se ve reflejado el Portafolio en su proyecto de vida?
8. ¿Cómo se evidencia la continuidad y la pertinencia temática del Portafolio?
9. ¿Qué nuevos significados se han derivado a partir de la elaboración del Portafolio?
10. ¿Cómo ha contribuido la elaboración del Portafolio a su crecimiento Intelectual y Pedagógico?

De acuerdo con las respuestas a los anteriores cuestionamientos, usted puede calificar la calidad de su Portafolio utilizando la escala de 0 a 5 y explicando las causas, motivos o justificaciones.

FUENTES DE CONSULTA:

UNAD-CAFAM. Especialización en Pedagogía para el desarrollo de Aprendizaje Autónomo. Guía de aprendizaje “A”.

GRUPO MEDELLIN 02. Principios Teóricos alrededor del Portafolio. Aportes Escritos de los asistentes al Encuentro Mensual de Semana 24. Medellín. 25-Jun-2000

ANEXO 3 FICHA DE SEGUIMIENTO

Según la metodología a distancia y el sistema de créditos académicos, Comprende el Estudio independiente y el Acompañamiento tutorial.

Estudio independiente

Es el fundamento de la formación y del aprendizaje. Se desarrolla a través del Trabajo personal y del trabajo en pequeños grupos colaborativos de Aprendizaje. Por cada crédito académico el estudiante debe dedicar en Promedio 32 horas al trabajo académico en estudio independiente.

Trabajo personal

Es la fuente básica del aprendizaje y de la formación e implica Responsabilidades específicas del estudiante con respecto al estudio en cada Curso académico del plan analítico, guía didáctica, módulo, lecturas Complementarias, consultas en biblioteca, consultas de sitios especializados a Través de Internet, desarrollo de actividades programadas en la guía didáctica, Elaboración de informes, realización de ejercicios de autoevaluación, Presentación de evaluaciones²⁰.

Se presenta una guía que permite reflexionar sobre la tarea

Nombre: _____

Grupo: _____ **Semana(1..18)** _____

Curso Académico: _____
 fecha: _____

1. Puntos de de referencia para reflexionar

- a. Claridad en la concepción de la actividad propuesta (¿lo que estoy haciendo es lo que piden las instrucciones?)
- b. Duración: ¿El tiempo empleado es el adecuado? Si / no me donde me excedí, con que rapidez debí hacerla
- c. ¿Tenia claridad en los conocimientos y habilidades necesarios para llevar a cabo la tarea?
- d. ¿Conocía los métodos requeridos para realizar el trabajo?, ¿comprendía las reglas del juego pertinentes?

²⁰ **EL MATERIAL DIDÁCTICO y el acompañamiento tutorial en el contexto de la formación a distancia y el sistema de créditos académicos Roberto J. Salazar Ramos**

- e. ¿Visualicé la complejidad de la tarea?, ¿Cuáles eran los puntos difíciles?
- f. ¿Preví los recursos necesarios para el desarrollo de la actividad?
- g. ¿tuve claro de exactitud y precisión con que debía realizar la tarea?
- h. ¿Preparé un plan de la tarea con el fin de distribuir el trabajo en el tiempo y ejercer el control?

2. complete los siguientes núcleos de conclusiones de acuerdo a sus reflexiones

- a. La tarea fue significativa para mí porque:_____
- b. La tarea demuestra mi comprensión sobre:_____
- c. Estoy muy orgulloso de está tarea porque:_____
- d. No estoy satisfecho con esta tarea porque :_____
- e. Algo que yo quiero que los demás vean en está tarea es:_____
- f. Una cuestión que quiero profundizar como resultado de esta tarea es:_____
- g. Esta tarea muestra mi progreso hacia el logro de mi meta porque:_____
- h. Esta tarea demuestra un desafío porque:_____

ANEXO 4 FORMATO PARA LA AUTOEVALUACIÓN DEL GRUPO COLABORATIVO²¹

		SI	NO
1	Trabajamos siguiendo un plan		
2	Trabajamos todos juntos		
3	Intentamos resolver la actividad de diferentes maneras		
4	Resolvimos la actividad		
5	Repasamos nuestro trabajo para asegurarnos que todos Estamos de acuerdo		
6	Le asignamos responsabilidades a cada miembro		
	Responsabilidad	Responsable	
7	Usamos los siguiente materiales o bibliografía		
8	Aprendimos:		
9	Resolvimos la actividad con la siguiente estrategia:		
10	Lo aprendido lo podemos aplicar en los siguientes contexto		

²¹ Técnicas y estrategias del pensamiento critico pag.166-168

ANEXO 5 COMAPAR Y CONTRASTAR

“Consiste en examinar los objetos con la finalidad de reconocer los atributos que los hacen tanto semejantes como diferentes. Contrastar es oponer entre si los objetos o compáralos haciendo hincapié en sus diferencias.”²²

- “Determine las características intrínsecas o criterios externos alrededor de los cuales los dos o mas elementos se van a compara de acuerdo con el pensamiento del auto o de acuerdo con su pensamiento, si discrepa del pensamiento del autor
- En una matriz de tres o más columnas, presente los resultados de la evaluación de cada elemento o conjunto de cada elemento o conjunto de elementos de acuerdo con los criterios o características y determine en qué son semejantes y en qué son diferentes los elementos

CARACTERISTICAS	ELEMENTO A	ELEMENTO B
1	Si la posee (+)	No la posee (+)
2		
3		
CONCLUSION		

- Existen otras formas de presentar los resultados de la comparación y contraste. Consúltelas y ensáyelas”²³

²² Técnicas y estrategias del pensamiento critico pag.116

²³ Especialización APRA el desarrollo del aprendizaje autónomo, guía C pag 80

ANEXO 7 HABILIDAD DE OBSERVAR

Observar se entiende en el sentido de advertir o estudiar algo con atención, cualquiera que sean los sentidos que en ello se emplean, Es lo que nos permite obtener información para identificar la cualidad, cantidad, textura, color, forma, número, Posición, entre otras.

Con esta habilidad es importante que usted se convierta en un detectives y observe todo aquello que lo rodea, preferiblemente, teniendo como referente los programas informáticos que utilizan las computadoras de diferentes entidades, teniendo en cuenta diseño, facilidad, usted puede diseñar un cuadro con anotaciones sobre cada una de la observaciones.

ANEXO 8 Índice de funciones

-A-

Función	Librería	Fichero de cabecera C
arc	graphics	graphics.h

-B-

Función	Librería	Fichero de cabecera C
bar	graphics	graphics.h
bar3d	graphics	graphics.h

-C-

Función	Librería	Fichero de cabecera C
cgets	conio	conio.h
circle	graphics	graphics.h
cleardevice	graphics	graphics.h
clearviewport	graphics	graphics.h
closegraph	graphics	graphics.h
clreol	conio	conio.h
clrscr	conio	conio.h
cprintf	conio	conio.h
cputs	conio	conio.h
cscanf	conio	conio.h

-D-

Función	Librería	Fichero de cabecera C
delline	conio	conio.h
detectgraph	graphics	graphics.h
drawpoly	graphics	graphics.h

-E-

Función	Librería	Fichero de cabecera C
ellipse	graphics	graphics.h

-F-

Función	Librería	Fichero de cabecera C
fillellipse	graphics	graphics.h
fillpoly	graphics	graphics.h
floodfill	graphics	graphics.h

-G-

Función	Librería	Fichero de cabecera C
getarccoords	graphics	graphics.h
getaspectratio	graphics	graphics.h
getbkcolor	graphics	graphics.h
getch	conio	conio.h
getche	conio	conio.h
getcolor	graphics	graphics.h
getdefaultpalette	graphics	graphics.h
getdrivename	graphics	graphics.h
getfillpattern	graphics	graphics.h
getfillsettings	graphics	graphics.h
getgraphmode	graphics	graphics.h

getimage	graphics	graphics.h
getline settings	graphics	graphics.h
getmaxcolor	graphics	graphics.h
getmaxmode	graphics	graphics.h
getmaxx	graphics	graphics.h
getmaxy	graphics	graphics.h
getmodename	graphics	graphics.h
getmoderange	graphics	graphics.h
getpalette	graphics	graphics.h
getpalettesize	graphics	graphics.h
getpass	conio	conio.h
getpixel	graphics	graphics.h
gettext	conio	conio.h
gettextinfo	conio	conio.h
gettext settings	graphics	graphics.h
getview settings	graphics	graphics.h
getx	graphics	graphics.h
gety	graphics	graphics.h
gotoxy	conio	conio.h
graphdefaults	graphics	graphics.h
grapherrormsg	graphics	graphics.h
graphfreemem	graphics	graphics.h
graphgetmem	graphics	graphics.h
graphresult	graphics	graphics.h

-H-

Función	Librería	Fichero de cabecera C
highvideo	conio	conio.h

-I-

Función	Librería	Fichero de cabecera C
imagesize	graphics	graphics.h
initgraph	graphics	graphics.h
inport	conio	conio.h
insline	conio	conio.h
installuserdriver	graphics	graphics.h
installuserfont	graphics	graphics.h

-K-

Función	Librería	Fichero de cabecera C
kbhit	conio	conio.h

-L-

Función	Librería	Fichero de cabecera C
line	graphics	graphics.h
linereel	graphics	graphics.h
lineto	graphics	graphics.h
lowvideo	conio	conio.h

-M-

Función	Librería	Fichero de cabecera C
moverel	graphics	graphics.h
movetext	conio	conio.h

moveto	graphics	graphics.h
--------	----------	------------

-N-

Función	Librería	Fichero de cabecera C
normvideo	conio	conio.h

-O-

Función	Librería	Fichero de cabecera C
outport	conio	conio.h
outtext	graphics	graphics.h
outtextxy	graphics	graphics.h

-P-

Función	Librería	Fichero de cabecera C
pieslice	graphics	graphics.h
putch	conio	conio.h
putimage	graphics	graphics.h
putpixel	graphics	graphics.h
puttext	conio	conio.h

-R-

Función	Librería	Fichero de cabecera C
rectangle	graphics	graphics.h
registerbgidriver	graphics	graphics.h
registerbgifont	graphics	graphics.h
restorecrtmode	graphics	graphics.h

-S-

Función	Librería	Fichero de cabecera C
sector	graphics	graphics.h
setactivepage	graphics	graphics.h
setallpalette	graphics	graphics.h
setaspectratio	graphics	graphics.h
setbkcolor	graphics	graphics.h
setcursortype	conio	conio.h
setfillpattern	graphics	graphics.h
setfillstyle	graphics	graphics.h
setgraphbufsize	graphics	graphics.h
setgraphmode	graphics	graphics.h
setlinestyle	graphics	graphics.h
setpalette	graphics	graphics.h
setrgbpalette	graphics	graphics.h
settextjustify	graphics	graphics.h
settextstyle	graphics	graphics.h
setusercharsize	graphics	graphics.h
setviewport	graphics	graphics.h
setvisualpage	graphics	graphics.h
setwritemode	graphics	graphics.h

-T-

Función	Librería	Fichero de cabecera C
textattr	conio	conio.h
textbackground	conio	conio.h
textcolor	conio	conio.h
textheight	graphics	graphics.h
textwidth	graphics	graphics.h

-U-

Función	Librería	Fichero de cabecera C
ungetch	conio	conio.h

-W-

Función	Librería	Fichero de cabecera C
wherex	conio	conio.h
wherey	conio	conio.h
window	conio	conio.h

Anexo 9

Tutorial de DFD

Tutorial de DFD Por: Mauricio Vargas Garro

Tutor Semillero LogicalSoft

Asignatura: Algoritmos y Fundamentos de Programación

Profesor: Ing Eliecer Suárez Serrano

UNICESAR 2005

1. Conceptos básicos para trabajar en DFD:

a. Que es DFD:

Dfd es un software diseñado para construir y analizar algoritmos . Usted puede crear diagramas de flujo de datos para la representación de algoritmos de programación estructurada a partir de las herramientas de edición que para éste propósito suministra el programa. Después de haber ingresado el algoritmo representado por el diagrama, podrá ejecutarlo, analizarlo y depurarlo en un entorno interactivo diseñado para éste fin. La interfaz gráfica de Dfd, facilita en gran medida el trabajo con diagramas ya que simula la representación estándar de diagramas de flujo en hojas de papel.

b. Que es un algoritmo:

Un algoritmo es un procedimiento para la resolución de problemas de cualquier tipo por medio de determinada secuencia de pasos simples y no ambiguos. El concepto fue utilizado originalmente para el cálculo matemático pero ahora es ampliamente usado en programación de computadoras.

c. Diagrama de Flujo de Datos

Un diagrama de flujo de datos es una descripción gráfica de un procedimiento para la resolución de un problema. Son frecuentemente usados para describir algoritmos y programas de computador. Los diagramas de flujo de datos están conformados por figuras conectadas con flechas. Para ejecutar un proceso descrito por un diagrama de flujo de datos se comienza por el INICIO y se siguen las flechas de figura a figura, ejecutándose las acciones indicadas por cada figura; el tipo de figura indica el tipo de paso que representa.

Los diagramas de flujo son frecuentemente usados debido a que pueden suprimir detalles innecesarios y tener un significado preciso, si son usados correctamente.

d. Tipos de Datos

Real: Valores numéricos que van desde -1×10^{2000} hasta 1×10^{2000} . Los valores más cercanos a 0 que se pueden manejar son 1×10^{-2000} y -1×10^{-2000} .

Ejemplo: 1998, 1.0007, 0, 328721, -3242781

Cadena de Caracteres: Secuencia de caracteres encerrada entre comillas simples.

Ejemplo: 'Diagramar es fácil' , 'París' , '1955'

Lógico: La letra V ó F encerrada entre puntos, para indicar verdadero ó falso respectivamente.

Ejemplo: .V. , .F. , .v. , .f.

e. Campos de Datos

Constantes: Con su nombre muestran su valor y éste no se puede cambiar.

Ejemplo: 1996 , 'Los algoritmos son útiles' , .V.

Variables: Es posible modificar su valor. El nombre de una variable debe comenzar por una letra seguida de letras, números o el caracter (_).

Ejemplo: Valor , Contador , año , Valor_1

No se tiene en cuenta la diferencia entre mayúsculas y minúsculas para el nombre de una variable; es decir, CASA equivale a casa. Cuando una variable recibe un valor por primera vez, el tipo de dato de ésta será igual al tipo de dato del valor.

f. Arreglos

Dfd soporta arreglos n-dimensionales de cualquier tipo de dato. El nombre de un arreglo debe comenzar por una letra seguida de letras, números o el caracter (_).

Ejemplo: Vector (2) , Matriz (i , j) , v (1, j, ñ, p)

No se tiene en cuenta la diferencia entre mayúsculas y minúsculas para el nombre de un vector; es decir, VECTOR(2) equivale a vector(2).

g. Interfaz de Usuario

Dfd posee una ventana principal que proporciona el ambiente de trabajo en donde se pueden construir y analizar algoritmos. Los componentes básicos de la ventana principal son: La barra de menú, barras de herramientas, barras de desplazamiento y el área de trabajo.

h. Acción Actual

Es el estado en el que se encuentra Dfd.

La acción actual puede ser:

Edición: Es el estado en el que un diagrama de flujo puede ser creado o modificado utilizando las herramientas de edición de Dfd. En este modo el diagrama también se puede imprimir, guardar y abrir.

Ejecución: Es la ejecución del algoritmo representado por el diagrama con el que se está trabajando. En tiempo de Ejecución pueden presentarse errores en el algoritmo, en tal caso se suspende la ejecución y se muestra el mensaje de error correspondiente.

Depuración: En este estado se puede observar con detalle el comportamiento del algoritmo, facilitando la detección y eliminación de errores. En Dfd las herramientas de depuración permiten realizar depuración /paso a paso y depuración/ejecutar hasta.

En depuración/paso a paso, la ejecución del algoritmo se realiza objeto por objeto haciendo uso del comando Paso simple.

En depuración/Ejecutar hasta, la ejecución del algoritmo se realiza deteniéndose en el objeto seleccionado haciendo uso del comando Ejecutar Hasta. Después de esto la acción actual será depuración/Paso a paso.

La barra de estado ubicada ubicada en la parte inferior de la ventana de Dfd muestra la acción actual.

i. Subprograma Actual

En Dfd, solo un subprograma (incluyendo el principal) puede ser visualizado a la vez, considerándose éste el Subprograma Actual.

j. Errores de Sintaxis

Estos errores son detectados en tiempo de revisión cuando se intenta ejecutar un algoritmo que contiene expresiones incorrectas. El mensaje de error correspondiente será mostrado y se indicara el objeto en el que se produjo el error.

Revisión del Diagrama: Cuando se intenta cambiar la acción actual de edición a cualquier otro modo, se realiza primero una revisión del diagrama para detectar errores de sintaxis, errores en los atributos de los objetos, entre otros. Si un error es detectado se muestra el mensaje de error correspondiente y se resalta el objeto en el cual se produjo el error.

2. Sistema de menus:

a. Archivo | Nuevo

El comando Nuevo inicia la sesión de trabajo con un nuevo diagrama.

Otras formas de acceder al comando:

Teclado: CTRL + N

Dfd da como nombre temporal al nuevo diagrama "Sin nombre.dfd", hasta que éste sea guardado con un nombre de archivo único. Al ejecutar este comando quedará seleccionada la opción Angulos en Grados del menú Opciones.

b. Archivo | Abrir

Inicia la sesión de trabajo con un diagrama ya existente, con este comando puede abrir un archivo de Dfd y comenzar a trabajar sobre él.

Otras formas de acceder al comando:

Teclado: CTRL + A

Al abrir un archivo de Dfd, las opciones del menú Opciones, tomarán el estado que tenían en el momento en que fue guardado el archivo.

c. Archivo | Guardar

Guarda en disco el diagrama que se está editando(principal y subprogramas) y el estado del menú Opciones, como un archivo de extensión "dfd".

A medida que Usted trabaja va haciendo cambios en el diagrama original, por lo cual es conveniente guardar con frecuencia el diagrama. Otras formas de acceder al comando: Teclado: CTRL + G.

d. Archivo | Guardar Como

El comando Guardar Como guarda en disco permite colocar un nombre al diagrama en edición. Se despliega un cuadro de diálogo donde se selecciona el nombre y la ubicación (unidad y directorio) del archivo en cual se va a guardar el diagrama.

Otra forma de acceder el comando:

Teclado: ALT + A , C

e. Archivo | Imprimir

Este comando despliega el cuadro de diálogo de impresión del sistema, el tamaño del diagrama a imprimir será proporcional al tamaño del diagrama que se visualiza en pantalla.

Otras formas de acceder al comando:

Teclado: CTRL + P

f. Archivo | Salir

El comando Salir termina una sesión de trabajo con Dfd .

Otras formas de acceder al comando:

Teclado: ALT + A , S

Si el diagrama en edición no ha sido guardado desde la última modificación, Dfd le preguntará si desea guardar antes de salir.

g. Edición | Cortar

Este comando se usa para eliminar un objeto seleccionado de un diagrama y colocarlo en el portapapeles de Dfd . El comando Cortar estará disponible cuando un objeto eliminable se encuentre seleccionado y la acción actual sea Edición.

Otras formas de acceder el comando:

Teclado: CTRL + X

Cuando se cortan objetos, estos reemplazan el contenido del portapapeles de Dfd . Los objetos que conforman estructuras de control serán cortados junto con su cuerpo.

h. Edición | Copiar

Este comando se usa para obtener una copia del objeto seleccionado en el portapapeles de Dfd. El objeto seleccionado queda intacto; es decir, no se remueve del diagrama. El comando Copiar estará disponible cuando exista un objeto eliminable seleccionado y la acción actual sea Edición.

Otras formas de acceder el comando:

Teclado: CTRL + C

Cuando se copian objetos, estos reemplazan el contenido del portapapeles de Dfd. Los objetos que conforman estructuras de control serán copiados juntos con su cuerpo.

i. Edición | Pegar

Use este comando para insertar una copia del contenido del portapapeles de Dfd a continuación del objeto seleccionado. El comando Pegar estará disponible cuando el portapapeles de Dfd no esté vacío, exista un objeto seleccionado y la acción actual sea Edición.

Otras formas de acceder al comando:

Teclado: CTRL + V

Después de haber sido pegado, el objeto permanece en el portapapeles de Dfd, de manera que puede pegarlo las veces que desee.

j. Edición | Eliminar

Este comando elimina el objeto seleccionado del diagrama sin colocarlo en el portapapeles de Dfd . Se encontrará disponible cuando un objeto eliminable se encuentre seleccionado y la acción actual sea Edición.

Otras formas de acceder al comando:

Teclado: SUPR

Los objetos que conforman estructuras de control (Son estructuras que ejercen control sobre la ejecución de bloques de objetos de acuerdo a una condición.) serán eliminados junto con su cuerpo. En caso de que el objeto seleccionado sea de tipo subprograma, entonces se ejecutará el comando Eliminar Subprograma.

k. Edición | Eliminar Subprograma

Este comando se usa para eliminar todos los objetos que conforman un subprograma. El comando estará disponible cuando esté visualizado un subprograma (no el principal) y la acción actual sea Edición.

Otras formas de acceder el comando:

Teclado: ALT + E, S

l. Edición | Insertar Objeto

Este comando se utiliza para insertar a continuación del objeto seleccionado un objeto del tipo que indique el ítem seleccionado en el menú Objeto; es decir, el último objeto seleccionado en la barra de herramientas.

El comando estará disponible cuando exista un objeto seleccionado, el ítem seleccionado en el menú Objeto sea diferente de Cursor y la acción actual sea Edición.

Otra forma de acceder al comando:

Teclado: INS

Mouse : Clic sobre la zona de inserción

m. Edición Objeto | Editar

Este comando se utiliza para editar el contenido de un objeto seleccionado.

Estará disponible cuando se encuentre seleccionado un objeto editable y la acción actual sea Edición.

Otra forma de acceder al comando:

Teclado: ENTER

Mouse : Doble clic sobre el objeto

n. Objeto | Cursor

Este comando selecciona el cursor normal del Mouse, el cual se puede usar para:

- Seleccionar y quitar la selección de objetos.
- Abrir los cuadros de diálogo para la edición de objetos.

Otras formas de acceder al comando:

Teclado: ALT + O, C

Cuando la acción actual es diferente de Ejecución, el cursor normal puede cambiar dependiendo de la posición del apuntador del Mouse.

Es la flecha de cursor que se presenta cuando el apuntador del Mouse no está sobre ningún objeto. La forma de este puntero depende de las propiedades del Mouse que maneja el sistema.

El cursor en forma de mano señalando se presenta cuando el apuntador del Mouse se sitúa sobre un objeto que se puede seleccionar, éste indica que se puede seleccionar, quitar la selección de otro objeto ó editar el objeto

Hacer clic con el botón izquierdo del Mouse dentro de un objeto selecciona el objeto y quita la selección a cualquier otro que se encuentre seleccionado en el subprograma actual. Hacer clic con el botón izquierdo del Mouse sobre un área vacía del diagrama quita la selección del objeto. Hacer doble clic con el botón izquierdo del Mouse sobre un objeto editable invoca al correspondiente cuadro de diálogo para la edición.

o. Objeto | Asignación

Este comando se utiliza para indicar que el siguiente objeto a ser insertado en el diagrama es de tipo Asignación.

Otras formas de acceder al comando:

Teclado: ALT + O, A

p. Objeto | Ciclo Mientras

Este comando se utiliza para indicar que el siguiente objeto a ser insertado en el diagrama es de tipo Ciclo Mientras.

Otras formas de acceder al comando:

Teclado: ALT + O, M

q. Objeto | Ciclo Para

Este comando se utiliza para indicar que el siguiente objeto a ser insertado en el diagrama es de tipo Ciclo Para.

Otras formas de acceder al comando:

Teclado: ALT + O, P

r. Objeto | Decisión

Este comando se utiliza para indicar que el siguiente objeto a ser insertado en el diagrama es de tipo Decisión.

Otras formas de acceder al comando:

Teclado: ALT + O, D

s. Objeto | Lectura

Este comando se utiliza para indicar que el siguiente objeto a ser insertado en el diagrama es de tipo Lectura.

Otras formas de acceder al comando:

Teclado: ALT + O, E

t. Objeto | Llamada

Este comando se utiliza para indicar que el siguiente objeto a ser insertado en el diagrama es de tipo Llamada.

Otras formas de acceder al comando:

Teclado: ALT + O, L

u. Objeto | Salida

Este comando se utiliza para indicar que el siguiente objeto a ser insertado en el diagrama es de tipo Salida.

Otras formas de acceder al comando:

Teclado: ALT + O, S

v. Objeto | Nuevo Subprograma

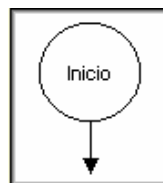
El comando Nuevo Subprograma crea un nuevo subprograma y lo deja como el subprograma actual. Este comando estará disponible cuando la acción actual sea Edición.

Otras formas de acceder al comando:

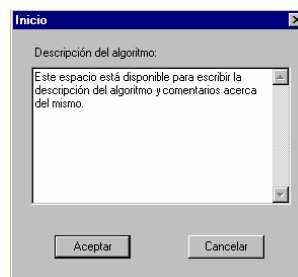
Teclado: ALT + O, N

3. Objetos que utiliza DFD

a. Objeto Inicio

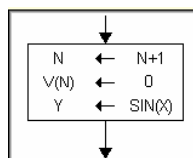


Es el primer objeto a ejecutar en cualquier algoritmo. Al ser ejecutado, el objeto Inicio transfiere el control al siguiente objeto.

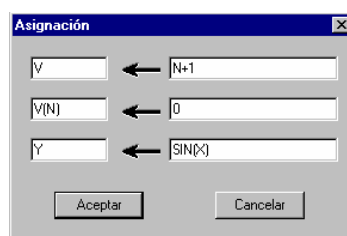


El cuadro de dialogo del objeto Inicio contiene un espacio para la descripción o comentarios acerca del algoritmo.

b. Objeto Asignación

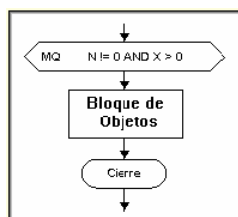


El objeto Asignación asigna valores a campos variables. Al ser ejecutado, puede realizar hasta tres asignaciones.



El cuadro de dialogo del objeto Asignación contiene espacio para tres asignaciones, cada asignación consta de un espacio para el campo variable situado siempre a la izquierda, el símbolo de asignación y un espacio para la expresión situada siempre a la derecha. Esto indica que al campo variable se le asigna el resultado de la evaluación de la expresión. Debe realizarse por lo menos una asignación.

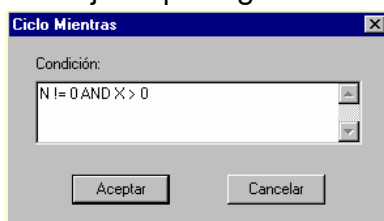
c. Objeto Ciclo Mientras



El objeto Ciclo Mientras tiene como función el ejecutar un bloque de objetos mientras que una condición sea verdadera. La condición debe ser siempre una expresión que al ser evaluada de como resultado un valor de tipo de dato Lógico.

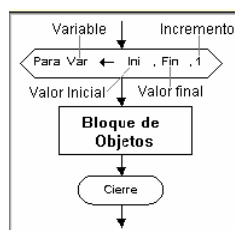
Ejemplo : $3 < W$, $x > 0$ AND $Sw = .V.$, $Valor * 15 < 300 * Contador$.

Si al evaluar la condición se obtiene el valor .F. la ejecución del algoritmo continuará a partir del objeto que sigue al Cierre.



El cuadro de dialogo del objeto Ciclo Mientras contiene espacio para la expresión que conforma la condición.

d. Objeto Ciclo Para

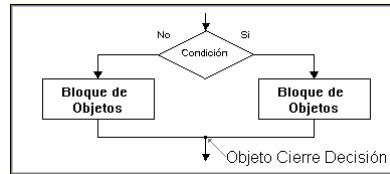


Su función es ejecutar un bloque de objetos mientras que la variable contadora no alcance el límite establecido por el valor final. El contador es siempre una variable de tipo de dato Real. Contiene además un valor inicial que será asignado al contador al iniciar la ejecución del ciclo, un valor final y un valor de incremento. Si el contador excede el valor final, la ejecución continuará a partir del objeto que sigue al Cierre. En caso contrario, se ejecutará el cuerpo del ciclo y el contador será incrementado en el valor indicado por el incremento.



El cuadro de diálogo del objeto Ciclo para contiene espacio para la variable contador, valor inicial, valor final y el valor de incremento en su respectivo orden.

e. Objeto Decisión

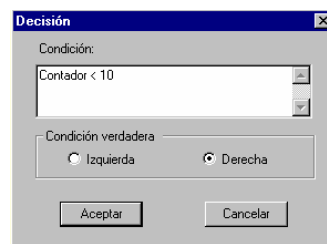


El objeto decisión selecciona el flujo a seguir de acuerdo al valor lógico de una condición. La condición debe ser siempre una expresión que al ser evaluada de como resultado un valor de tipo de dato Lógico.

Ejemplo : $3 < w$, $x > 0$ AND $sw = .V.$, $valor * 15 < 300 * contador$.

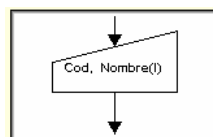
El objeto Decisión esta asociado a dos bloques de objetos ubicados a lado y lado de este, y un objeto Cierre Decisión ubicado a continuación de ambos bloques.

Si al evaluar la condición se obtiene el valor lógico .V., se ejecuta el bloque rotulado con la palabra Si, en caso contrario se ejecuta el bloque rotulado con No. En ambos casos la ejecución continua en el objeto Cierre Decisión.

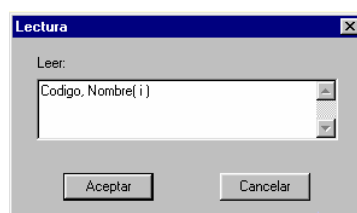


El cuadro de dialogo del objeto Decisión contiene espacio para la expresión que conforma la condición, y dos casillas por medio de las cuales se puede especificar por cual lado continuara el flujo en caso de que la condición sea verdadera.

f. Objeto Lectura

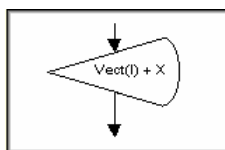


El objeto Lectura permite la entrada de valores constantes desde el teclado y se los asigna a campos variables . Podrá ser leída cualquier cantidad de variables utilizando un objeto Lectura. Al ejecutarse, el objeto despliega un cuadro de diálogo por cada variable presente en la lista, este cuadro de diálogo espera que el usuario introduzca un valor constante que será asignado a la respectiva variable.

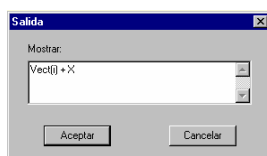


El cuadro de diálogo para la edición del objeto contiene un espacio para ingresar una lista de variables separadas por comas. Debe existir por lo menos una variable.

g. Objeto Salida

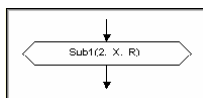


El objeto Salida muestra valores por pantalla. Puede ser visualizada cualquier cantidad de valores utilizando un objeto Salida. Al ejecutarse, este objeto evalúa cada una de las expresiones que contiene y despliega un cuadro de diálogo que muestra el valor obtenido en cada una de las expresiones en su respectivo orden.



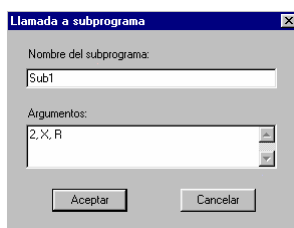
El cuadro de diálogo para la edición del objeto contiene un espacio para ingresar una lista de expresiones separadas por comas. Debe existir por lo menos una expresión.

h. Objeto Llamada



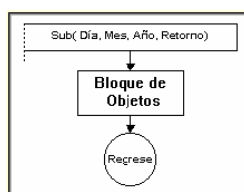
La función de este objeto es realizar una llamada a un subprograma, el cual debe encontrarse en el diagrama en edición. En la llamada deben encontrarse los argumentos que han de ser pasados al subprograma, la cantidad, el orden y el tipo de los argumentos deben coincidir con los parámetros del subprograma.

Una vez que el subprograma haya sido ejecutado la ejecución continuará en el objeto siguiente a la llamada.



El cuadro de diálogo para la edición de este objeto contiene el espacio para el nombre del subprograma a llamar y el espacio para la lista de argumentos. Dichos argumentos deben estar separados por comas.

i. Objeto Subprograma



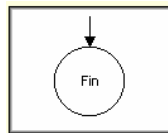
Es el primer objeto a ser ejecutado cuando un subprograma es llamado. Al ser ejecutado, el objeto Subprograma transfiere el control al siguiente objeto.

El cuadro de diálogo del objeto Subprograma contiene un espacio para la descripción o comentarios acerca del mismo ; contiene un espacio para el nombre del subprograma y un espacio para los parámetros. Estos parámetros (si existen) deben estar separados por comas. El nombre de un subprograma debe comenzar por una letra seguida de letras, números ó el carácter (_).

Ejemplo: Factorial , Leer , Sub1 , sub_programa.

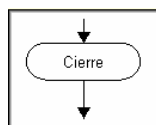
No se tiene en cuenta la diferencia entre mayúsculas y minúsculas para el nombre de un subprograma, es decir , SUB equivale a sub.

j. Objeto Fin



Este objeto junto con el objeto Inicio, delimita el cuerpo del procedimiento principal. Solo existe un objeto Fin en el diagrama ; la ejecución de este objeto finaliza la ejecución del algoritmo.

k. Objeto Cierre Ciclo

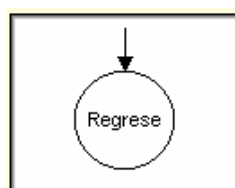


Este objeto delimita el cuerpo de un ciclo, al culminar la ejecución del ciclo el control se transfiere al objeto que sigue al objeto Cierre Ciclo.

l. Objeto Cierre Decisión

Este objeto delimita el cuerpo de una estructura de decisión, al culminar la ejecución de dicha estructura el control se transfiere al objeto que sigue al objeto Cierre Decisión.

m. Objeto Regrese



Este objeto junto con el Objeto Subprograma,

delimita el cuerpo de un subprograma. La ejecución de este objeto transfiere el control al objeto que realizó la llamada.

Anexo 10

DESCRIPCIÓN DE LOS ELEMENTOS DEL MENÚ PRINCIPAL

Elemento	Opciones
=	Muestra el número de versión, limpia o restaura la pantalla y ejecuta varios programas de utilidades proporcionados por Turbo C
File	Carga y graba archivos, gestiona directorios, invoca al DOS y sale de Turbo C
Edit	Realiza funciones de edición
Search	Realiza diversas búsquedas y reemplazamientos
Run	Compila, enlaza y ejecuta el programa cargado actualmente en el entorno
Compile	Compila el programa actual del entorno
Debug	Establece diversas opciones del depurador, incluido el establecimiento de puntos de ruptura
Project	Gestiona proyectos con varios archivos
Options	Establece diversas opciones del compilador, enlazador y del entorno
Window	Controla la forma en la que se muestran diversas pantallas
Help	Activa el sistema de ayuda sensible al contexto

FILE

Resaltando la opción **File**, se activa un menú desplegable con las opciones:

Opción	Descripción
Open	Pide un nombre de archivo y lo carga en el editor. Si el archivo no existe se crea. Esta opción también muestra una lista de archivos entre los que se puede elegir.
New	Abre otra ventana de edición y le permite crear un archivo nuevo. El archivo se llama por defecto NONAME n .C, donde n es un valor entre 0 y 99. Se puede renombrar el archivo cuando se graba
Save	Graba el archivo de la ventana activa
Save as	Le permite grabar el archivo con un nombre diferente
Save all	Graba los archivos de todas las ventanas abiertas
Change dir	Cambia el directorio por defecto a uno que se especifique
Get info	Muestra información sobre el archivo de la ventana activa
DOS shell	Carga el intérprete de órdenes de DOS y le permite ejecutar órdenes de DOS. Se debe escribir <i>EXIT</i> para volver a Turbo C
Quit	Salida de Turbo C

EDIT

Resaltando la opción **Edit**, se activa un menú desplegable con las opciones:

Opción	Descripción
Undo	Deshacer la última operación de edición
Redo	Rehacer la última operación de edición
Cut	Corta el bloque seleccionado
Copy	Copia el bloque seleccionado
Paste	Pega en la ventana activa del editor, en la posición actual del cursor, el bloque más recientemente copiado o cortado
Clear	Borra el bloque seleccionado
Copy example	Cuando se activa el sistema de ayuda y se pide información sobre una característica de C, se incluye con frecuencia un ejemplo. Si este es el caso, se puede copiar automáticamente el código del ejemplo seleccionando esta opción

Show clipboard	Muestra los bloques anteriormente cortados o copiados
-----------------------	---

RUN

Resaltando la opción **Run**, se activa un menú desplegable con las opciones:

<i>Opción</i>	<i>Descripción</i>
Run	Ejecuta el programa actual. Si el programa no ha sido compilado todavía, Run lo compila y luego lo ejecuta
Program reset	Termina el programa cuando se está ejecutando en modo de depuración
Go to cursor	Ejecuta el programa hasta que alcanza la línea de código donde está colocado el cursor
Trace into	Ejecuta el programa sentencia a sentencia. Si la siguiente sentencia incluye una llamada a una subrutina, se traza la subrutina
Step over	Ejecuta el programa paso a paso. Ejecuta la siguiente línea de código, pero no traza las subrutinas a las que pueda llamar
Arguments	Se usa para pasar argumentos de la línea de órdenes a un programa que se ejecuta desde el entorno

COMPILE

Resaltando la opción **Compile**, se activa un menú desplegable con las opciones:

<i>Opción</i>	<i>Descripción</i>
Compile to OBJ	Compila el archivo actual del editor a un archivo .OBJ, un archivo .OBJ es un archivo de código objeto reubicable que está preparado para ser enlazado en un archivo .EXE que se pueda ejecutar
Make EXE file	Compila directamente el programa en un archivo ejecutable
Link EXE file	Permite enlazar el programa actual
Build all	Construir todo, recompila todos los archivos relacionados con el programa actual
Remove messages	Limpia la ventana de mensajes

OPTIONS

Resaltando la opción **Options**, se activa un menú desplegable con las opciones:

<i>Opción</i>	<i>Descripción</i>
Compiler	Permite cambiar aspectos sobre la forma en la que Turbo C genera el código
Transfer	Permite añadir programas al menú de sistema de Turbo C. Los programas mostrados en el menú de sistema de Turbo C pueden ejecutarse desde dentro de Turbo C sin dejar el IDE
Make	Cambia las opciones de cómo compilar el programa
Linker	Cambia las opciones de cómo enlazar el programa
Directories	Permite establecer los directorios que usa Turbo C por defecto
Debugger	Permite establecer varias opciones de depuración
Environment	Permite cambiar la forma en la que opera el IDE
Save	Permite determinar cuántas opciones del IDE se graban para un uso futuro de Turbo C

WINDOW

Resaltando la opción **Window**, se activa un menú desplegable con las opciones:

Opción	Descripción
Size/Move	Se puede cambiar el tamaño de la ventana activa o moverla a otra posición de la pantalla
Zoom	Aumenta el tamaño de la ventana activa de modo que llene completamente la pantalla. Una vez que una ventana ha sido ampliada, si se selecciona Zoom por segunda vez, la ventana vuelve a su tamaño normal
Tile	A cada ventana se le da una parte reducida de la pantalla
Cascade	Cada vez que se crea una ventana nueva ésta queda parcialmente sobre una o más de las otras ventanas
Next	Si se tienen varias ventanas abiertas, con esta opción se puede saltar progresivamente de una a la siguiente
Close	Para cerrar una ventana de la pantalla
Message	Ventana usada por Turbo C para dar información
Output	Muestra la salida generada cuando se ejecuta un programa dentro de la pantalla IDE
Watch	Se utiliza en depuración
User screen	Muestra la pantalla completa de salida de un programa. Si se selecciona esta opción, se debe pulsar F5 para volver a la pantalla del IDE
Register	Muestra el contenido de cada registro de la CPU
Project	Estas ventanas están relacionadas con los proyectos
List all	Para hacer un listado de todas las ventanas abiertas. Se puede activar una ventana seleccionándola en esta lista

HELP

Resaltando la opción **Help**, se activa un menú desplegable con las opciones:

Opción	Descripción
Contents	Muestra la tabla de contenidos del sistema de ayuda
Index	Activa un índice de temas tratados por el sistema de ayuda. Para hacer una selección: 1. se mueve el resaltado al tema deseado y se pulsa <ENTER>. 2. se digita la palabra relativa al tema deseado y se pulsa <ENTER>. Entonces se ve la información relativa al tema seleccionado. Para salir del sistema de ayuda se pulsa <ESC>
Topic search	Se muestra información sobre la palabra clave en la que está colocado actualmente el cursor
Previous topic	Para ver el tema anterior
Help on help	Para ver ayuda sobre el sistema de ayuda