UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA

ESCUELA DE CIENCIAS BASICAS TECNOLOGIA E INGENIERIA

PROGRAMA INGENIERIA DE SISTEMAS

301303 - ALGORITMO

IVAN ARTURO LOPEZ

ivan.lopez@unad.edu.co
(Director Nacional)

Acreditador

POPAYAN
Febrero de 2010

ASPECTOS DE PROPIEDAD INTELECTUAL Y VERSIONAMIENTO

El presente módulo fue diseñado en el año 2004 por el Ing. Iván Arturo López Ortiz, docente de la UNAD, adscrito a la Escuela de Ciencias Básicas Tecnología e Ingeniería y ubicado en el CEAD de Popayán zona centro sur, el Ing. Iván L. es Ingeniero de sistemas, Especialista en pedagogía para el desarrollo del aprendizaje autónomo y maestro en Educación Tics, está vinculado con la universidad desde febrero de 2002.

El presente módulo ha tenido tres actualizaciones, todas desarrolladas por el mismo ingeniero.

En este momento se encuentra apoyando la revisión y certificación de material el ing.?

INTRODUCCIÓN

El curso de Algoritmos, está adscrito a la Escuelas de Ciencias Básicas tecnología e Ingeniería de la UNAD y corresponde al programa de Ingeniería de Sistemas, esta constituido por tres créditos académicos, correspondientes a 36 actividades de acompañamiento y 106 de estudio independiente, de acuerdo al contenido programático establecido por la Escuela de ciencias básicas tecnología e ingeniería, esta dirigido inicialmente a estudiante s de la UNAD de segundo semestre o periodo académico, sin que esto implique que lo puedan tomar otros participantes deseosos 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 iníciales para el desarrollo del 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 Aguilar[1], 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# de Microsoft.

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,

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

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.

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 práctica 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.

INDICE DE CONTENIDO

UNIDAD I

UNIDAD II

LISTADO DE TABLAS

LISTADO DE GRÁFICOS Y FIGURAS

UNIDAD 1

Nombre de la Unidad	Generalidades y antecedentes
Introducción	La primera unidad del curso de algoritmos, está 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, 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 una recopilación de lecturas que permiten la apropiación del conocimiento y se evidencian en diferentes productos que le ofrecen la oportunidad de aplicar las diversas herramientas adquiridas en los primeros cursos académicos; también se plantean y 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
Justificación	Una de las principales razones para que los estudiantes del los programas de ingeniería de la UNAD aprendan y

	conceptualicen sobre las técnicas y lenguajes de programación radica en que la computadora se convierte en un elemento esencial y de uso cotidiano, pedo deben existir personas que como ustedes escriban esos programas (software), que respondan a los requerimientos cada vez más estrictos de los usuarios “Normales”. Uno de los aspectos fundamentales de esta unidad es dar una contextualización por los elementos fundamentales y de contexto general que se debe tener antes de iniciar la construcción de algoritmos y programas de cómputo. Por ende los capítulos de esta unidad conceptualizarán en los elementos históricos contextuales de conocimiento básico que los participantes deben adquirir.
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** Al finalizar la primer Unidad:

	El estudiante es capaz de comprender los conceptos fundamentales de la programción de computadoras. El estudiante conocerá la evolución histórica de las computadoras acompañada de los lenguajes de programación El Estudiante estará en capacidad de diferenciar los diferentes tipos de datos y operadores necesarios en la construcción de programas informáticos **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
Denominación de capítulos	**CAPITULO 1: GENERALIDADES.** **CAPITULO 2: CLASIFICACIÓN DE LOS COMPUTADORES “ORDENADORES”.** **CAPITULO 3: TIPOS DE DATOS Y OPERADORES**

1. CAPITULO 1: GENERALIDADES.

Introducción

Es importante que los estudiantes del curso de algoritmos, adquieran o recuerden los conocimientos básicos en temas referentes a la informática como el funcionamiento elemental de los computadores, sus dispositivos hardware, las características esenciales del software y básicamente 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 lograr esto se utilizará una serie de lecturas apoyadas en presentaciones y videos

Lección 1: Introducción a la informática

Introducción a la informática. Introducción a la informática

El hombre desde sus orígenes siempre ha tenido la necesidad de comunicarse con sus semejantes y sobre todo la de transmitir información, que en sus inicios se realizó y se mantuvo por muchas generaciones de forma oral, con las consecuencias que esto conlleva (perdida u olvido de información, tergiversación de la misma...), posteriormente con el surgimiento de la escritura (jeroglíficos), está se pudo almacenar desde las rocas, hasta llegar al papel y los medios digitales que hoy conocemos , más información en: http://centros5.pntic.mec.es/ies.arzobispo.valdes.salas/alumnos/escri/presen.html

A su vez se hizo imprescindible que los medios de comunicación se fuesen perfeccionando desde la comunicación oral en cortas distancias hasta lograr el cableado de “toda la tierra”, si desea ampliar más esta información en: http://es.wikipedia.org/wiki/Medio_de_comunicaci%C3%B3n

Con esto se puede afirmar que el hombre desde que tuvo la necesidad de comunicarse, guardar y transmitir información también ha tenido la necesidad de idear nuevos mecanismos que permitan agilizar y potenciar estos mecanismos a demás por el incremento permanente de información que se procesa nace la ***informática.***

Lección 2: Algunas definiciones*:*

"El término ***Informática*** proviene de la unión de las palabras **infor**mación y auto**mática**. 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 (computadores).

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.

Ordenador o computador: 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 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:

Figura1: Entrada y salida de datos

¿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.

Lección 3: 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

- Carné 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.

Se puede tener más información en:

http://www.slideshare.net/gugaslide/representacion-de-informacion-en-computadoras-presentation

Lección 4: Las diferentes unidades de medida

BIT Un BIT es una manera "binaria " de presentar información; es decir, expresa una de solamente dos alternativas posibles. Se expresa con un 1 o un 0, con un sí o no, verdadero o falso, blanco o negro, algo es o no es, voltaje o no voltaje, un nervio estimulado o un nervio inhibido. (Sabemos que no todo lo que se encuentra en nuestro universo es blanco o negro, pero aún así podemos utilizar esta forma binaria de representación para expresar estados intermedios logrando la precisión deseada).

BYTE Es la unidad de información formada por ocho bits (01011101). Según cómo estén combinados los bits (ceros o unos), formaran un bytes dependiendo de la cantidad de bytes, formarán kilobytes, un megabytes, gigabytes, etc. Relacionados: Nibble que equivale a medio bytes; DBCS: es el conjunto de caracteres que necesitan dos bytes para aparecer**.**

KILOBYTE Unidad de medida de la cantidad de información en formato digital. Un byte consiste de 8 bits. Un BIT es un cero (0) o un uno (1). Por lo tanto un ejemplo de un byte es 01001001. Esa secuencia de números (byte) pueden simbolizar una letra o un espacio. Un kilobytes (Kb) son 1024 bytes y un Megabytes (Mb) son 1024 Kilobytes

MEGABYTE El Megabytes (MB) es una unidad de medida de cantidad de datos informáticos. Es un múltiplo binario del byte, que equivale a 220 (1 048 576) bytes, traducido a efectos prácticos como 106 (1 000 000) bytes.

GYGABYTE Es una unidad de almacenamiento. Existen dos concepciones de gigabytes (GB). (Debemos saber que un byte es un carácter cualquiera) Un gigabytes, en sentido amplio, son 1.000.000.000 bytes (mil millones de bytes), ó también, cambiando de unidad, 1.000 megas (MG o megabytes). Pero si somos exactos, 1 GB son 1.073.741.824 bytes ó 1.024 MB.

TERABYTE Una unidad de almacenamiento tan desorbitada que resulta imposible imaginársela, ya que coincide con algo más de un trillón de bytes. Un uno seguido de dieciocho ceros.

Tabla No 1: Conversiones

File Storage Capacity by Bits and Bytes					
	Bit	**byte**	**Kilobyte**	**Megabyte**	**Gigabyte**
bit	1	8	8,192	8,388,608	8,589,934,592
byte	8	1	1,024	1,048,576	1,073,741,824
Kilobyte	8,192	1,024	1	1,024	1,048,576
Megabyte	8,388,608	1,048,576	1,024	1	1,024
Gigabyte	8,589,934,592	1,073,741,824	1,048,576	1,024	1
Terabyte	8,796,093,022,208	1,099,511,627,776	1,073,741,824	1,048,576	1,024
Petabyte	9,007,199,254,740,990	1,125,899,906,842,620	1,099,511,627,776	1,073,741,824	1,048,576
Exabyte	9,223,372,036,854,780,000	1,152,921,504,606,850,000	1,125,899,906,842,620	1,099,511,627,776	1,073,741,824
Zettabyte	9,444,732,965,739,290,000,000	1,180,591,620,717,410,000,000	1,152,921,504,606,850,000	1,125,899,906,842,620	1,099,511,627,776

Estructura De Un Ordenador.

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

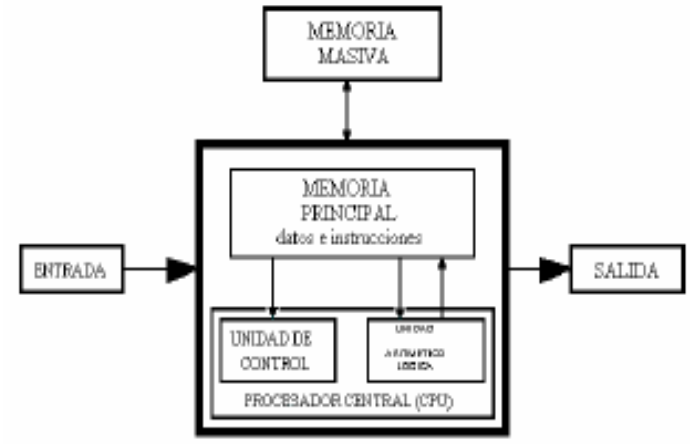


Figura2: Estructura de un ordenador

Una computadora se compone de las siguientes unidades funcionales

Unidades funcionales

Los computadores se caracterizan por tener una diversidad de dispositivos que se pueden distinguir entre las diferentes unidades, como las unidades de entrada de salida, de almacenamiento y muchas más

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

- 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:

Figura3: dispositivos de salida y/o entrada

- 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.

Lección 5: 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 necesitemos 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 las memorias:

ROM (Read Only Memory)

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

RAM (Random Acces Memory).

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.

Otras Unidades

Existen otras unidades de gran importancia en el funcionamiento del computador y que normalmente no están visibles para el usuario normal, algunas de estas unidades vienen incluidas en el mismo dispositivo

Unidad Aritmético-Lógica (ALU):

Como su nombre indica se encarga de realizar las operaciones aritméticas (suma, resta, 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 período 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 microelectrónico, variasunidades 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.

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.*"[2]

Mirar video en: http://blip.tv/file/982668/

[2] Tomado de: Documento tomado de: http://wwwdi.ujaen.es/asignaturas/fundTopo/TEMA1.pdf

2. CAPITULO 2: CLASIFICACIÓN DE LOS COMPUTADORES “ORDENADORES”.

Introducción

Se ha tenido una diversidad de criterios para clasificar los computadores, pero en este apartado se realizará en función de su capacidad y potencia, en este sentido esta es clasificación

Lección 6: Según su Capacidad y potencia

Superordenadores

“Supercomputadora o Superordenador es una computadora con capacidades de cálculo muy superiores a las comúnmente disponibles de las máquinas de escritorio de la misma época en que fue construida.

Hoy se utiliza en ingeniería, medicina y ciencia. Eso incluye el desarrollo de biocombustibles y el diseño de vehículos que gasten menos combustible. Ingenieros de IBM y del laboratorio de Los Álamos trabajaron seis años en la tecnología de la computadora. Algunos elementos de Roadrunner tienen como antecedentes videojuegos populares, de acuerdo con David Turek, vicepresidente del programa de supercomputadoras de IBM. En cierta forma, se trata "de una versión superior de Sony PlayStation 3*, indicó. "Tomamos el diseño básico del chip (de PlayStation) y mejoramos su capacidad*, informó Turek.

Sin embargo, la supercomputadora Roadrunner difícilmente pueda asemejarse a un videojuego. El sistema de interconexión ocupa 557 m² de espacio. Cuenta con 91,7 km de fibra óptica y pesa 226,8 **Supercomputadora** o **Superordenador** es una computadora con capacidades de cálculo muy superiores a las comúnmente disponibles de las máquinas de escritorio de la misma época en que fue construida.

Hoy se utiliza en ingeniería, medicina y ciencia. Eso incluye el desarrollo de biocombustibles y el diseño de vehículos que gasten menos combustible. Ingenieros de IBM y del laboratorio de Los Álamos trabajaron seis años en la tecnología de la computadora. Algunos elementos de Roadrunner tienen como antecedentes videojuegos populares, de acuerdo con David Turek, vicepresidente del programa de supercomputadoras de IBM. En cierta forma, se trata "de una versión superior de Sony PlayStation 3*, indicó. "Tomamos el diseño básico del chip (de PlayStation) y mejoramos su capacidad*, informó Turek.

Sin embargo, la supercomputadora Roadrunner difícilmente pueda asemejarse a un videojuego. El sistema de interconexión ocupa 557 m² de espacio. Cuenta con 91,7 km de fibra óptica y pesa 226,8 t. La supercomputadora está en el laboratorio de investigaciones de IBM en Poughkeepsie, Nueva York y fue

trasladada en julio del 2008 al Laboratorio Nacional Los Alamos, en Nuevo México."[3], se pude ampliar más en: http://es.wikipedia.org/wiki/Supercomputadora

Mainframes o Macroordenadores:

"Es una computadora grande, potente y costosa usada principalmente por una gran compañía para el procesamiento de una gran cantidad de datos; por ejemplo, para el procesamiento de transacciones bancarias.

Figura4: Macroordenadores

La capacidad de una computadora central se define tanto por la velocidad de su CPU como por su gran memoria interna, su alta y gran capacidad de almacenamiento externo, sus resultados en los dispositivo E/S rápidos y considerables, la alta calidad de su ingeniería interna que tiene como consecuencia una alta fiabilidad y soporte técnico caro pero de alta calidad. Una computadora central puede funcionar durante años sin problemas ni interrupciones y las reparaciones del mismo pueden ser realizadas mientras está funcionando. Los vendedores de computadoras centrales ofrecen servicios especiales; por ejemplo, si se rompe la computadora, el vendedor ejecutará las aplicaciones de su cliente en sus propias computadoras sin que los usuarios lo noten mientras que duran las reparaciones. La independencia interna de estas computadoras es tan grande que, por lo menos, en un caso conocido, los técnicos pudieron cambiar las computadoras centrales de sitio desmontándolas pieza a pieza y montándolas en otro lugar, dejando, mientras tanto, dichas computadoras funcionando; en este ejemplo, el cambio de las computadoras centrales de un sitio a otro se produjo de manera transparente."[4]

Servidores

Un **servidor** es una computadora que, formando parte de una red, provee servicios a otras computadoras denominadas clientes.

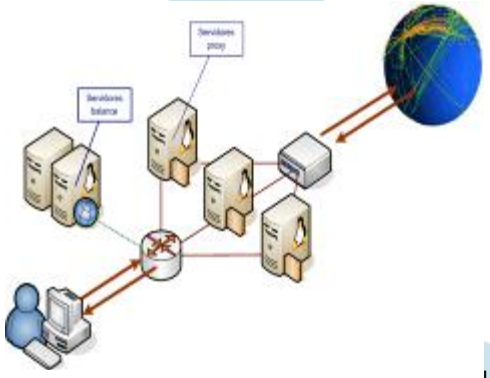


Figura5: Servidor

También se suele denominar con la palabra servidor a:

Una aplicación informática o programa que realiza algunas tareas en beneficio de otras

upercomputadora

Tomado de : http://es.wikipedia.org/wiki/Computadora_central

aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final. Este es el significado original del término. Es posible que un ordenador cumpla simultáneamente las funciones de cliente y de servidor.

Una computadora en la que se ejecuta un programa que realiza alguna tarea en beneficio de otras aplicaciones llamadas clientes, tanto si se trata de un ordenador central (*mainframe*), un miniordenador, un ordenador personal, una PDA o un sistema integrado; sin embargo, hay computadoras destinadas únicamente a proveer los servicios de estos programas: estos son los servidores por antonomasia.

Un servidor no es necesariamente una máquina de última generación de grandes proporciones, no es necesariamente un superordenador; un servidor puede ser desde una computadora vieja, hasta una máquina sumamente potente (ej.: servidores web, bases de datos grandes, etc. Procesadores especiales y hasta varios gigabytes de memoria). Todo esto depende del uso que se le dé al servidor. Si usted lo desea, puede convertir al equipo desde el cual usted está leyendo esto en un servidor instalando un programa que trabaje por la red y a la que los usuarios de su red ingresen a través de un programa de servidor web como Apache."[5]

Computadoras Personales

"Tiene tres significados:

La gama de computadoras personales de IBM que originaron el uso del término: IBM PC.

Figura6: Computadora Personal

Término genérico utilizado para referirse a microcomputadoras que son compatibles con las especificaciones de IBM.

Término genérico utilizado a veces para referirse a todas las microcomputadoras.

Una computadora personal es una microcomputadora, diseñada en principio para ser usada por una sola persona a la vez, y que es compatible con el PC de IBM (aunque en el lenguaje corriente se puede referir también a equipos incompatibles). Una computadora personal es generalmente de tamaño

[5] Tomado de : http://es.wikipedia.org/wiki/Servidor

medio y es usado por un sólo usuario (aunque hay sistemas operativos que permiten varios usuarios simultáneamente, lo que es conocido como multiusuario).

Una computadora personal suele estar equipada para cumplir tareas comunes de la informática moderna, es decir permite navegar por Internet, escribir textos y realizar otros trabajos de oficina además de escuchar música, ver vídeos, jugar, estudiar, etc."[6]

Lección 7: Software

Programas e instrucciones Las computadoras ates mencionadas no podrían cumplir ninguna función si estas no cuentan con su complemento el "Software", los cuales se construyen mediante líneas de condigo (de las cuales hablaremos más adelante como objeto primordial del curso), en esencia un software es un programa que está constituido por una serie de sentencias o instrucciones que son una serie de símbolos que permiten el ingreso datos a la computadora y esta a su vez entrega una información, estas sentencias se clasifican en sentencias e instrucciones:

Sentencias imperativas o instrucciones:

Representan una orden para el ordenador. : Sentencias declarativas: Proporcionan información sobre los datos que maneja el programa." En este sentido se puede decir que: 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 y 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

[6] Tomado de: http://es.wikipedia.org/wiki/Computadora_personal

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..” [7]

Lección 8: Programación de computadoras:

La programación de computadoras es un proceso en el cual y por medio de una serie de instrucciones se que se le ingresan a una computadoras se le dice que debe realizar ante una situación en particular, en este sentido el nombre que adquiere la persona que programa las computadoras se denomina programador y para realizar esto el necesita de una serie de herramientas software (que las han realizado otros programadores), denominados lenguajes de programación.

Y ya como tal l concepto de programación *estructurada* (hay otros tipos de programación de los cuales me referiré más adelante), 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.

Figura 7:E.W.Dijkstra

Figura 8: C.A.R.Hoare

[7] Tomado de: http://wwwdi.ujaen.es/~mcdiaz/docencia/cur04_05/fi/teoria/01_Introduccion.pdf

Los lenguajes de programación se les ha dado una clasificación para poder tener un mejor entendimiento que pueden ser según su nivel de abstracción y según el paradigma de programación

Nivel de Abstracción:

"Generalmente, en entornos científicos, cuando se estudia un problema demasiado complejo se utilizan mecanismos que permitan simplificarlo pero que a la vez se muestren sus aspectos más relevantes. A este proceso de simplificación, en el que ciertos aspectos se ocultan reduciendo así su complejidad, se le denomina *abstracción*

Para ilustrar el proceso de abstracción se puede tomar como ejemplo el problema de buscar un edificio en cualquier lugar del mundo mediante su dirección de correo postal. Las direcciones de correo postal incluyen el país en el que se encuentra el destinatario. En un primer nivel de búsqueda, se toma la información referida a todos los países del mundo y se selecciona el país especificado en la dirección. En una segunda fase, se obtiene la información sobre la división interna de un país (estas divisiones pueden tener diferentes nombres tales como comunidades, provincias, departamentos, estados, etc.) Una vez localizada la unidad territorial se debe seleccionar una ciudad. Para ello, tan sólo es preciso obtener la información de ciudades incluidas en la unidad territorial bajo consideración y localizar la del destinatario. Finalmente, dentro de la ciudad se debe obtener la información sobre las calles y sus números para conocer la posición exacta del edificio que se busca"[8], según este nivel, los lenguajes se clasifican en lenguajes de: máquina, bajo nivel, medio nivel y alto nivel

[8] Tomado de : http://ocw.uc3m.es/ingenieria-telematica/arquitectura-de-ordenadores/lecturas/html/int.html#id2642122

Lenguaje de máquina:

El lenguaje de maquina son aquellas cadenas de código directamente legibles por la máquina que en esencia son cadena de dígitos 0 y 1 que si bien recordamos son la representación lógica del manejo electrónico en la computadora, la ventaja de esto es la rapidez de la ejecución de los programas, sin embargo se complica un poco cuando se pretende escribir estos programas y la inserción de estos a la computadora.

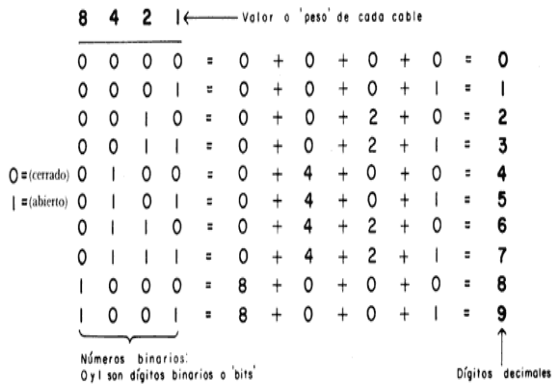


Figura9: Representacion código binario

Lenguaje de bajo nivel:

"Los lenguajes de bajo nivel son lenguajes de programación que se acercan al funcionamiento de una computadora. El lenguaje de más bajo nivel por excelencia es el código máquina. A éste le sigue el lenguaje ensamblador, ya que al programar en ensamblador se trabajan con los registros de memoria de la computadora de forma directa. Ejemplo:

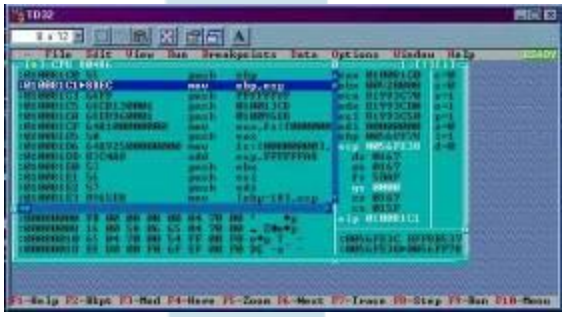


Figura10: Editor de lenguaje de maquina

```
;Lenguaje ensamblador, sintaxis Intel para procesadores x86
mov eax,1 ;mueve a al registro eax el valor 1
xor ebx, ebx ;pone en 0 el registro ebx
int 80h ;llama a la interrupción 80h (80h = 128 sistema decimal)"[9]
```

Lenguaje de medio nivel:

"Hay lenguajes de programación que son considerados por algunos expertos como lenguajes de medio nivel (como es el caso del lenguaje C) al tener ciertas características que los acercan a los lenguajes de bajo nivel pero teniendo, al mismo tiempo, ciertas cualidades que lo hacen un lenguaje más cercano al humano y, por tanto, de alto nivel. Ejemplo:

```
/*Lenguaje C*/
/*declaración de las funciones estandars de entrada y salida*/
```

[9] Tomado de http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n#Lenguajes_de_M.C3.A1quina

```
#include <stdio.h>
int main(int argc, char **argv)
{
   char *p; /*creamos un puntero a un byte*/
   if(argc == 1){
     printf("\nIngrese un argumento al programa\n");/*imprimimos el texto*/
     return 1;
   }
   p = 0x30000 /*el puntero apunta a 0x30000 */
   *p = argv[1][0] /*el primer caracter del primer argumento lo copiamos a la posición 0x30000 */
   return 0;
}
```

El ejemplo es muy simple y muestra a los punteros de C, éstos no son muy utilizados en lenguajes de alto nivel, pero en C sí."[10]

Lenguaje de alto nivel

Figura11: biblioteca de texto

"Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés. En BASIC, uno de los lenguajes de alto nivel más conocidos, los comandos como "IF CONTADOR = 10 THEN STOP" pueden utilizarse para pedir a la computadora que pare si el **CONTADOR** es igual a **10**. Esta forma de trabajar puede dar la sensación de que las computadoras parecen comprender un lenguaje natural; en realidad lo hacen de una forma rígida y sistemática, sin que haya cabida, por ejemplo, para ambigüedades o dobles sentidos. Ejemplo:

```
{Lenguaje Pascal}
program suma;

var x,s,r:integer; {declaración de las variables}
begin {comienzo del programa principal}
   writeln('Ingrese 2 números enteros');{imprime el texto}
   readln(x,s);  {lee 2 números y los coloca en las variables x y s}
   r:= x + s; {suma los 2 números y coloca el resultado en r}
   writeln('La suma es  ',r);  {imrpime el resultado}
```

[10] Tomado de: http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n#Lenguajes_de_medio_nivel

```
    readln;
end.{termina el programa principal}
```

Ese es el lenguaje Pascal, muy utilizado por principiantes al aprender a programar.” 11

Paradigma de programación

El paradigma de programación es la filosofía como se estructura las sentencias de programación, en este sentido los lenguajes de programación se los puede clasificar como: imperativos, funcionales, lógicos, orientado a objetos

Imperativos

Un lenguaje imperativo programa mediante una serie de comandos, agrupados en bloques y compuestos de órdenes condicionales que permiten al programa retornar a un bloque de comandos si se cumple la condición. Estos fueron los primeros lenguajes de programación en uso y aún hoy muchos lenguajes modernos usan este principio.

No obstante, los lenguajes imperativos estructurados carecen de flexibilidad debido a la secuencialidad de las instrucciones[12].

Funcionales

Un lenguaje de programación funcional(a menudo llamado lenguaje procedimental) es un lenguaje que crea programas mediante funciones, devuelve un nuevo estado de resultado y recibe como entrada el resultado de otras funciones. Cuando una función se invoca a sí misma, hablamos de recursividad.13, este curso trabajaremos sobre este tipo de categorización, dado que el enfoque es instrucciones y procedimental.

Lógicos

“La computación lógica direcciona métodos de procesamiento basados en el razonamiento formal. Los objetos de tales razonamientos son "hechos" o reglas "if then". Para computar lógicamente se utiliza un conjunto de tales estamentos para calcular la verdad o falsedad de ese conjunto de estamentos. Un estamento es un hecho si sus tuplas verifican una serie de operaciones.

[11] Tomado de: http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n#Lenguajes_de_alto_nivel

[12] Tomado de: http://es.kioskea.net/contents/langages/langages.php3

[13] Tomado de: http://es.kioskea.net/contents/langages/langages.php3

Un hecho es una expresión en la que algún objeto o conjunto de objetos satisface una relación específica. Una tupla es una lista inmutable. Una tupla no puede modificarse de ningún modo después de su creación.

Un regla if then es un estamento que informa acerca de conjuntos de tuplas o estamentos relacionados que pueden predecir si otras tuplas satisfacerán otras relaciones.

Un estamento que es probado verdadero como resultado de un proceso se dice que es una inferencia del conjunto original. Se trata por tanto de una descripción de cómo obtener la veracidad de un estamento dado que unas reglas son verdaderas."[14]

La computación lógica está por tanto relacionada con la automatización de algún conjunto de métodos de inferencia.

Orientado a objetos

Figura12: Programacion Orientada a Objetos

"Los lenguajes de programación orientados a objetos tratan a los programas como conjuntos de objetos que se ayudan entre ellos para realizar acciones. Entendiendo como objeto a las entidades que contienen datos. Permitiendo que los programas sean más fáciles de escribir, mantener y reutilizar.

Los objetos tienen toda la información (atributos) que los diferencia de otros pertenecientes a otra clase. Por medio de unos métodos se comunican los objetos de una misma o diferente clase produciendo el cambio de estado de los objetos. Esto hace que a los objetos se les trate como unidades indivisibles en las que no se separan la información ni los métodos usados en su tratamiento.

Este lenguaje tiene su origen en un lenguaje que fue diseñado por los profesores Ole-Johan Dahl y Kristen Nygaard en Noruega. Este lenguaje de programación orientado a objetos fue el "Simula 67" que fue un lenguaje creado para hacer simulaciones de naves.
Son lenguajes dinámicos en los que estos objetos se pueden crear y modificar sobre la marcha. Esta programación orientada a objetos (POO) tomo auge a mediados de los años ochenta debido a la propagación de las interfaces gráficas

[14] Tomado de: http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n#Lenguajes_L.C3.B3gicos

de usuarios, para lo que los lenguajes de programación orientados a objetos están especialmente dotados.

Entre los principales lenguajes de este tipo tenemos:

Ada, C++, C#, VB.NET, Clarion, Delphi, Eiffel, Java, Lexico (en castellano), Objective-C, Ocaml, Oz, PHP, PowerBuilder, Python, Ruby y Smalltalk."[15]

"Algunos lenguajes de Programación[16]

A continuación se referenciaran algunos de los lenguajes de programación más comunes y conocidos en la actualidad.

"FORTRAN clic para 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 máquina. 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

[15] Tomado de: http://www.articulandia.com/premium/article.php/25-09-2006Lenguajes-de-programacion-orientada-a-objetos.htm

[16] [1] http://www.itq.edu.mx/vidatec/espacio/aisc/ARTICULOS/leng/LENGUAJESDEPROGRAMACI%D3N.htm

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, Zuiza, al final de los años 1960 y principios de los 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. Ala 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 Exceptión – 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 Wokstation. Dispone de un compilador muy rápido (más que la mayorìa de los compìladores 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 peden 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 UNÍS de alta calidad en 1991 como parte de un proyecto de investigación para desarrollar software para dispositivos electrónicos (televisiones, video casseteras, tostadores y otros de aparatos que se pueden comprar en cualquier tienda departamental). Fundamentado en C++, el lenguaje Java se diseño 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 (apples y aplicaciones) pueden ejecutarse en cualquier computadora que tenga instalada una maquina 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

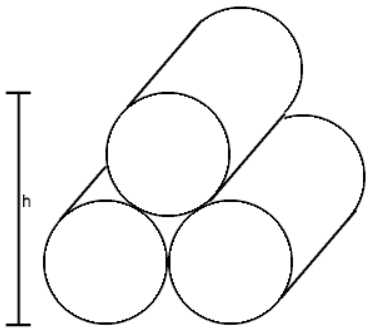
Lección 9: Lógica de la programación

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

Acertijos

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?

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 http://acertijos-y-enigmas.com.ar/, de pronto se convierta en un adicto a este tipo de juegos.

Lección 10: 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

Esto es una adaptación de la lectura de técnicas de programación
"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 término "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 órdenes, 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 maquinan 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 mismos1lenguajes 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.

Estilos de programación :

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 prohiben) 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 aprendido 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 implementacion. Colectivamente cubren los aspectos más importantes con los que ha de enfrentarse el diseñado 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 recién 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 ANS. Esto significa que el American National Standards Institute ha adoptado esa versión del lenguaje como el estándar nacional. Una vez que un lenguaje esta estandarizado, las maquinas que implementan este lenguaje deben cumplir todas las especificaciones estándares, reforzando así el máximo de transportabilidad de programas de una maquina a otra. La policía federal de no comprar maquinas que no cumplan la versión estándar de cualquier lenguaje que soporte tiende a "fortalecer" el proceso de estandarizacion, puesto que el gobierno es, con mucho, el mayor comprador de computadoras de la nación.

Finalmente, la notación algebraica ordinaria, por ejemplo, influyo fuertemente en el diseño de FORTRAN y ALGOL. Por otra parte, el ingles influyo en el desarrollo del COBOL. El lambda cálculo de Church dio los fundamentos de la notación funcional de LISP, mientras que el algoritmo de Markov motivo el estilo de reconocimiento de formas de SNOBOL. La arquitectura de computadoras de Von Neumann, la cual fue una evolución de la maquina mas antigua de Turing, es el modelo básico

de la mayoría de los diseños de computadoras de las ultimas tres décadas. Esta máquina no solo influyeron en los primeros lenguajes sino que también suministraron el esqueleto operacional sobre el que evoluciono 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 ingles o castellano puede escribirse igualmente como una maquina de Turing (maquina 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.

3. CAPITULO 3: TIPOS DE DATOS Y OPERADORES

Introducción

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 de diferente tipo de dato:

Lección 11. Tipos de datos

En la actualidad la mayoría de los lenguajes de programación y por lo tanto des la construcción de los algoritmos es indispensable definir con claridad qué tipo de dato tendrá una determina variable (tema que se estudiará más adelante), los tipos de datos que se emplean son:

- *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:
Permiten representar valores escalares de forma numérica, esto incluye a los números enteros y los reales. Este tipo de datos permiten realizar operaciones aritméticas comunes

Enteros:
Representan los números que no posee componente fraccionaria y pueden ser tanto positivos como negativos, Ejemplo 2345 , 4567 , -3451

***Reales*:**
Representan todos los números que poseen componente fraccionaria y también pueden positivo o negativo, ejemplo: 2345.20 , 4567.10 , -3451.01

Datos Lógicos **(booleano),**
Este tipo de dato solo puede tomar uno de dos valores verdadero o falso (true or false). Este tipo de datos se utiliza para representar las opciones (si/no) a

determinadas preguntas, es el caso: cuando se pide si un valor entero es positivo, la respuesta ser verdadera o falsa, según sea positivo o negativo.

Datos tipo Carácter:

Representan datos alfanuméricos que pueden ser reconocidos por la computadora y estos 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 esté 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”

Lección 12: Variables Y Constantes

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,

Variables locales:

Es aquella que afecta únicamente el subprograma, es decir solo un bloque de programa bien definido, un ejemplo de la vida cotidiana puede ser aquellas cosas que afectan únicamente lo que se encuentra en la casa, por es el caso si no pago el recibo de energía la suspenden, pero esto no afecta a las casas de los vecinos dado que ellos no se afectan directamente de la suspensión de la energía en mi casa.

Variable Global:

Variable que afecta a un programa en todo su contexto, programa principal y modulo, lo cual se entenderá mejor cuando se mire el concepto de funciones, sin embargo y para continuar con el ejemplo anterior, si es el municipio quien no cancela los recibos de alumbrado público y este es suspendido, mire que afecta a muchas personas y entidades incluida la seguridad publica etc,

Reglas para la definición de variables

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

- Deben iniciar con una letra (a…z), excepto la ñ
- No deben contener símbolos ni signos de puntuación como estos: # $ % & / ( ) = ? ¡ ¿ +.
- Do deben contener espacios en blanco, esto es en caso de una palabra esta no debe ser compuesta

Ejemplo:

Variable	**Estado**
Contador	→ correcto (cumple con las reglas)
44444	→ in correcto (no debe iniciar con un número)
Mi contador	→ in correcto (contiene espacio en blanco)
Pedro	→correcto (cumple con las reglas)
#k	→incorrecto (Inicia con un carácter diferente a una letra)
K	→Correcto (cumple con las reglas)
Kkk1	→Correcto (cumple con las reglas)

Constantes:
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:

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.

Operadores:

Los operadores se dividen en operadores Aritméticos, que se encargan de las operaciones aritméticas como sumas restas.. y las operaciones con cadena de carácter como es el caso de la concatenación.

Aritméticos.

Los operadores aritméticos nos permiten realizar cualquier operación aritmética básica que necesitemos como: suma, resta, multiplicación, división y otras que las revisaremos a lo largo del módulo. En la siguiente tabla se muestran los operadores más comunes y que se emplean en la mayoría de los lenguajes con algunas variaciones.

Tabla No 2: Operadores

Operador	Acción	ejemplo
^ o **	Potencia	X = 2^3 // x = 8
*****	Multiplicación	X = 2*3 // x = 6
/	División	X = 12/3 // x = 4
+	Suma	X = 2+3 // x = 5
-	Resta	X = 6-3 // x = 3
Div	División entera	X = 7 div 3 // x = 2

Mod	Modulo (residuo)	X = 9 mod 3 // x = 0
		X = 9 Mod 2 // x = 1

Operadores: Alfanuméricos.

Existe una cantidad de operaciones que se pueden realizar con alfanuméricos, pero en realidad estas son funciones específicas de cada lenguaje, para este caso solo se trabaja la concatenación igual a la suma

•Concatenación. **+**

Ejm.

'UN' + 'AD'

↓

'UNAD'

Lección 13: Operadores

Existen diversos dos grandes grupos de operadores definidos en la programación de computadores, los operadores relacionales y los operadores lógicos

Operadores: Relacionales.

También denominados operadores binarios lógicos y de comparación, se utilizan para comprobar la veracidad o falsedad de determinadas propuestas de relación. Las expresiones que los contienen se denominan **expresiones relacionales**. Aceptan diversos tipos de argumentos, y el resultado, que es la respuesta a la pregunta, es siempre del tipo verdadero o falso, es decir, producen un resultado **booleano** que se miro en un apartado anterior

Tabla No 3: Condicionales

Denominación	Símbolo
Igual a.	=
Menor que.	<
Menor o igual que.	<=
Mayor que.	>
Mayor o igual que.	>=
Distinto a.	< > o !=

Operadores: Lógicos.

Como operadores lógicos designamos a aquellos operadores que nos permiten ``conectar'' más de una propiedades, sugiero revisar el modulo de lógica matemática para recordar este tema tan interesante

Tabla No 4: Operadores lógicos

Denominación	Ingles	español
Negación.	Not	No
Conjunción/producto.	And	Y
Disyunción/suma.	Or	Or

Nota El paréntesis se puede considerar como un operador dado que 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

Lección 14: Prioridad en la evaluación de operadores

En la programación de computadoras es indispensable tener en cuenta el orden como se escriben las operaciones aritméticas con sus operadores, dado que la operación no se evalúa de izquierda a derecha como lo hace una calculadora, si no como un todo y por tano analiza la regla de prioridad

Tabla No 5: Prioridad de operadores

Prioridad	Nombre	Sigo
1	Paréntesis.	( )
2	Cambio de signo.	+ -
3	Potencias.	^
4	Productos y divisiones.	* /
5	División entera	Div
6	Modulo	Mod
7	Sumas y restas.	+

Observación

El operador MOD, permite obtener el residuo de una división

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

Ejemplo:

X= 2 + 3 * 5 + (2*3)

X=2+3*5+6

X=2+15+6

X=23

Es importante tener en cuenta que cuando existen operadores del mismo nivel, este se evalúa de izquierda a derecha Ejemplo.

X= 1-2+3*5+(2*3)

X=4-2+3*5+6

X=4-2+15+6

X= 2+15+6

X=23

Lección 15: 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	c) SAS
5.Pascal	e) Compiladores
6.Hardware	f) Tratamiento Automático de la información
7.Lenguajes declarativos	g) Lenguaje de Programación
8.Software	h) Disco Duro

Ejercicio 2.0 Definir los Siguientes 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.

Ejercicio 4.0

De los siguientes identificadores de variables cuales no son validos, y cuál será la opción correcta

a) Pedro si__ no___ Porque?
b) Xpedro si__ no___ Porque?
c) Contador5 si__ no___ Porque?
d) 8contador si__ no___ Porque?
e) #suma si__ no___ Porque?
f) Con tador si__ no___ Porque?

Ejercicio 5.0

Obtener el resultado de la variable X

a) X=5+3*2

X=?

b) X=5+3*2

X=X+X

X=?

c) X=4+(3*2)+7↑2 +4/2

X=?

d) A=5

B=10

C=8

X=A+B*C+(B-C)

X=X-A

X=?

e) X=35 DIV 4

X=X MOD 2

X= X+X

X=?

UNIDAD 2

Nombre de la Unidad	**Estructura General de un Algoritmo**
Introducción	En la segunda unidad nos enfocaremos ya a la solución de supuestos polémicos, a pesar de que el curso está 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 muchos 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 cuándo y cómo 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
Justificación	La segunda unidad es quizá una de las más relevantes para los estudiantes que se inician en el mundo de la programación, dado que se da nociones de la lógica de la programación, desde la construcción e interpretación de diagramas de flujo, con todas sus representaciones graficas hasta llegar a la construcción de algoritmos con todos los requerimientos y normativas que estos requieren. Como se podrá apreciar el estudiante necesita de una gran dedicación dado que como yo lo afirmo “la construcción de diagramas y de algoritmos no se aprende en ningún texto, esto se hace a basé de ejercicios, como las matemáticas”.

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
Denominación de capítulos	**CAPITULO 4: DIAGRAMAS DE FLUJO** **CAPITULO 5: ALGORITMOS** **CAPITULO 6: SUBPROGRAMA O MODULO**

4. CAPITULO 4: DIAGRAMAS DE FLUJO

Introducción

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:

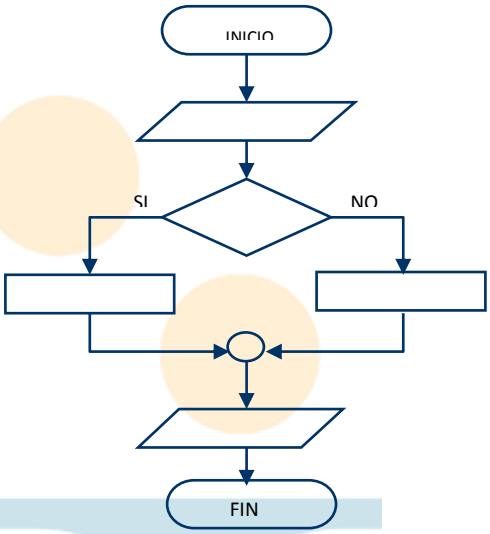


Figura13:Diagrama de flujo

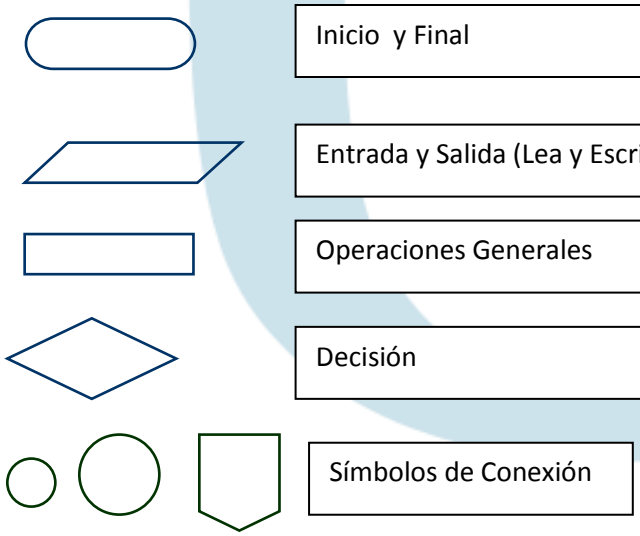
Lección 1: Características de los Diagramas

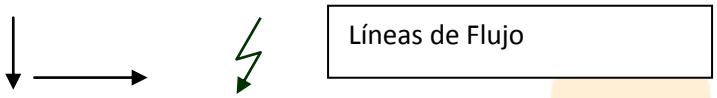
Entre sus aspectos fundamentales están:

- 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:





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

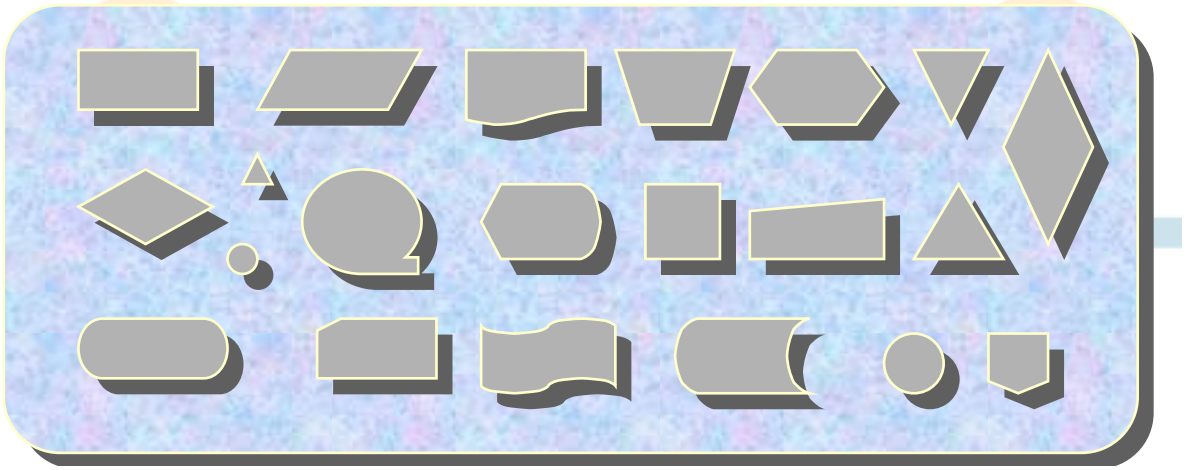


Figura14:simbolos diagrama de flujo

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

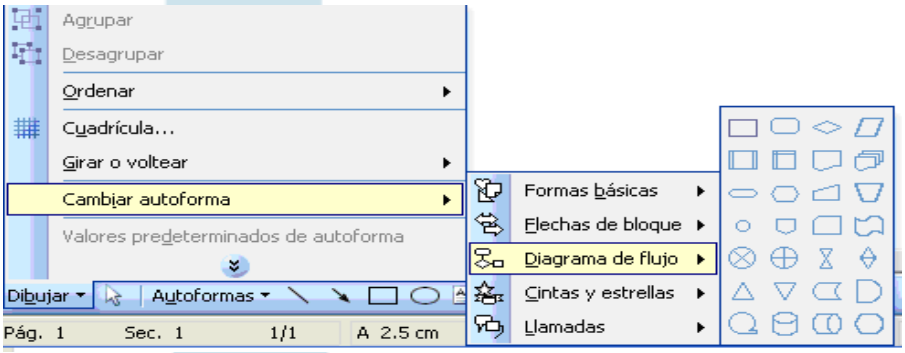
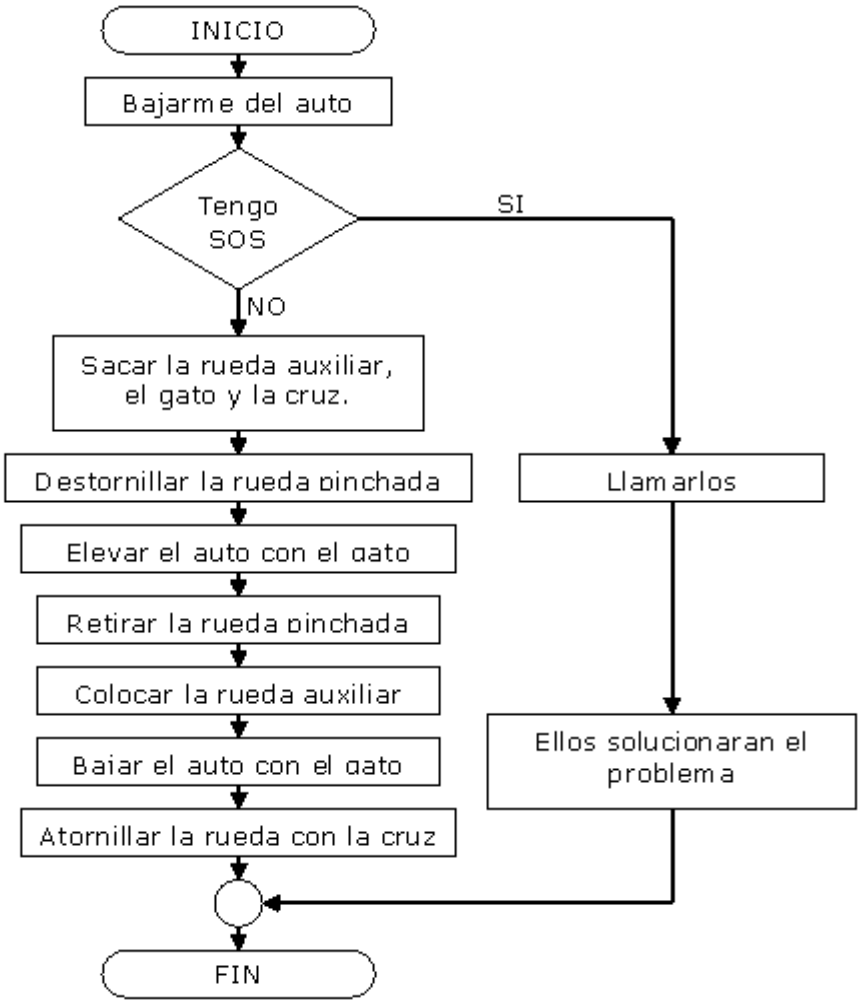


Figura 15:Graficos procesador de texto

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

Lección 2: Ejemples prácticos

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 seria, cambiar la rueda . Luego nos tenemos que preguntar Cómo?, 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

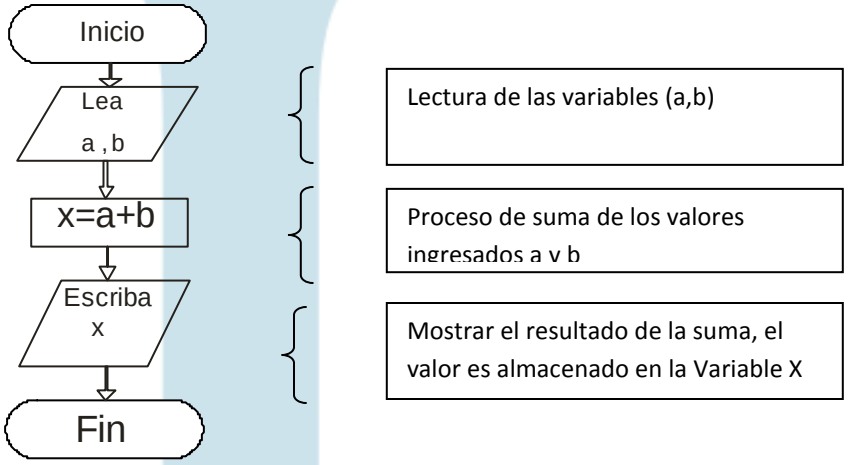
1.- leer cuidadosamente el planteamiento del ejercicio

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 sumarlas (las variables), para luego mostrar el resultado,



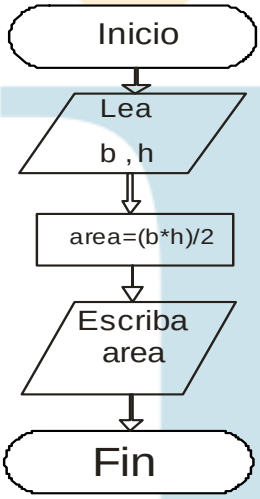
En algunos textos el símbolo escriba se representa cómo este símbolo pero para nuestro caso se utilizara el mismo como se menciono antes

Ejemplo # 2

Encontrar el área de un triangulo y mostrar su resultado

Análisis

Para la realización de este ejercicio es indispensable conocer la formula de un triangulo (b*h)/2, si nos damos cuenta en la formula, 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 formula 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*2)/2=5

Otros valores

b	h	area
20	4	40

Avancemos

Ahora vamos a utilizar condicionales.

Lección 3: Condicionales

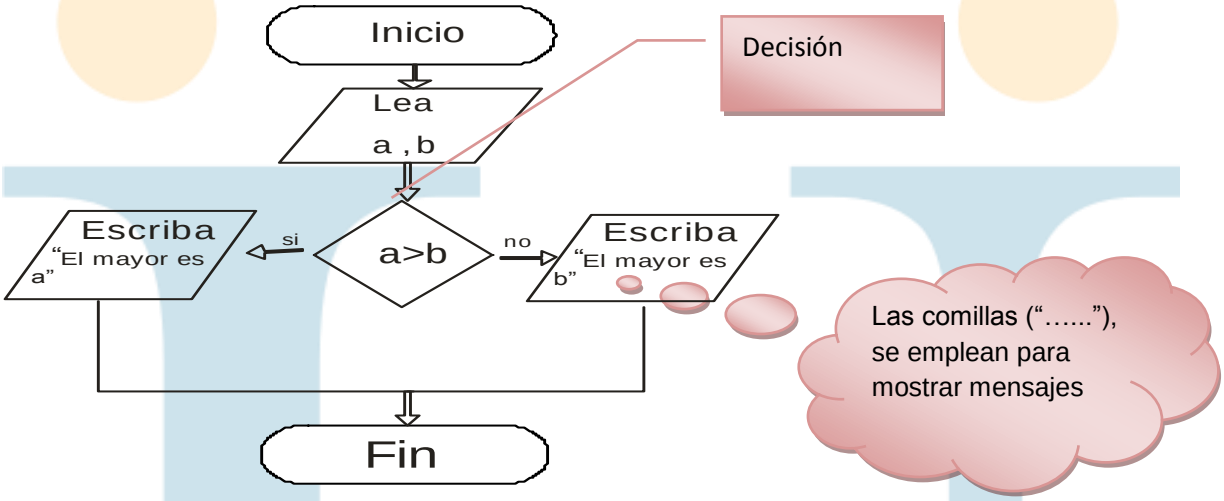
Es un parámetro que permite tomar una decisión, para el caso de la programación estructurada solo existe dos caminos a seguir cuando se evalúa un condicional, por un lado puede ser **si** y por otro puede ser **no** Para entender mejor el concepto lo haremos mediante un ejercicio, en este tipo de programación no existe “quien sabe” o “el tal vez”, propios del la vida diaria, por ejemplo a la pregunta, ¿Usted tiene hambre?, se puede responder Si o No, pero también podría decir si tengo un poco, en el caso de la programación si pregunto tienen hambre la respuesta es Si o No.

Ejemplo # 3

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

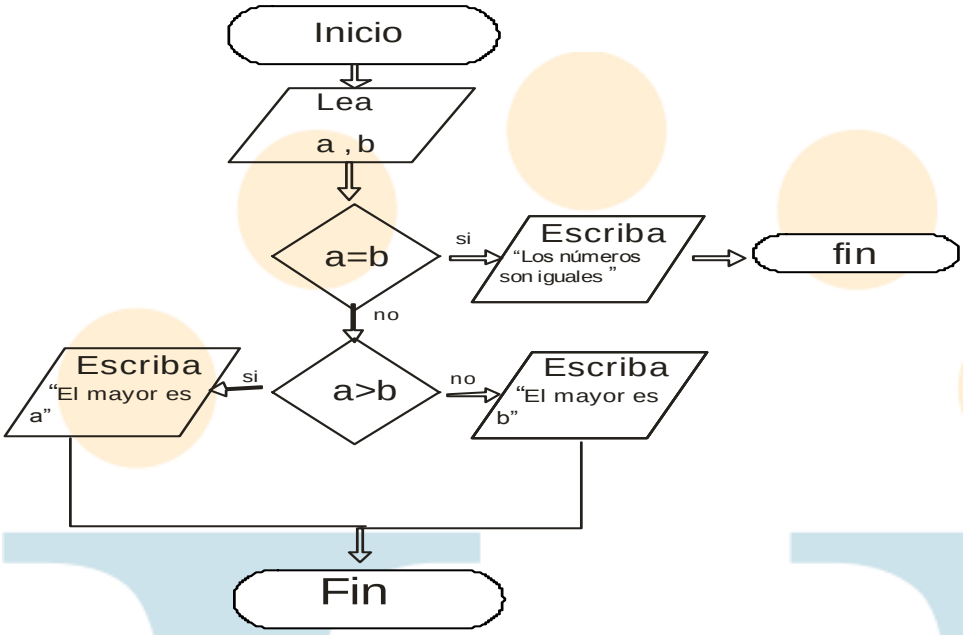
Análisis

Para determinar cuál 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 A y B son iguales?

Para ese caso necesitamos de un segundo condicional que verifique si las dos variables son iguales (A = B), a continuación se propone la solución.



Se recomienda utilizar la herramienta **dfd, desarrollada por el grupo Smart** de la universidad del Magdalena, la cual la puede descargar de: http://ivan.lopezortiz.googlepages.com/algoritmos

Esta herramienta permite utilizar las representaciones graficas del diagrama de flujo directamente en un programa y obtener los resultados. A continuación presento una adaptación desarrollada con la colaboración del Ing Javier Villero Maestre[17] y desarrollado en Unicesar, donde se muestra todo el potencial de esta herramienta. También y si es del caso lo invito a revisar un corto video del funcionamiento de esta herramienta que también esta publicada en la dirección antes mencionada.

[17] Coordinador Zona Caribe ECBTI - UNAD

Lección 4: Dfd

Tutorial de DFD Por: Mauricio Vargas Garro
Tutor Semillero LogicalSoft
Asignatura: Algoritmos y Fundamentos de Programación
Profesor: Ing Eliecer Suárez Serrano
UNICESAR 2005

Conceptos básicos para trabajar en DFD:

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.

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.

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.

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.

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.

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).

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.

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 esta 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.

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

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.

Sistema de menus:
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.

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.

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.

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

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

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.

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.

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.

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.

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.

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

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

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

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.

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

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

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

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

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

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

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:

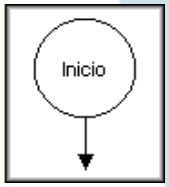
Teclado: ALT + O, S

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.

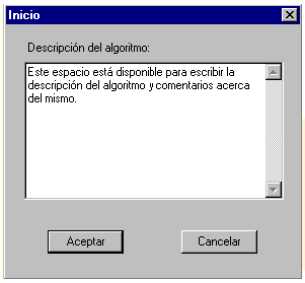
Otras formas de acceder al comando:

Teclado: ALT + O, N

Objetos que utiliza DFD
Objeto Inicio

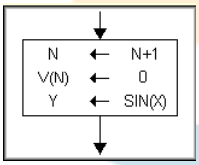


Es el primer objeto a ejecutar en cualquier algoritmo. Al ser ejecutado, el objeto Inicio transfiere el control al siguiente objeto.



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

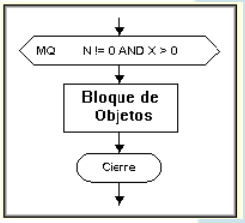
Objeto Asignación



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

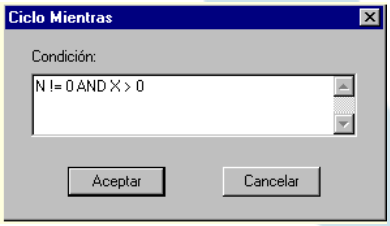
El cuadro de dialogo del objeto Asignación contiene espacio para tres asignaciones, cada asignación consta de un espacio para el campo variable situado siempre a la izquierda, el símbolo de asignación y un espacio para la expresión situada siempre a la derecha. Esto indica que al campo variable se le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

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.

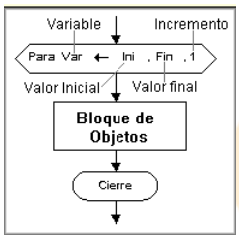
Ejemplo : 3<W , x >0 AND Sw = .V. , Valor*15 < 300*Contador.



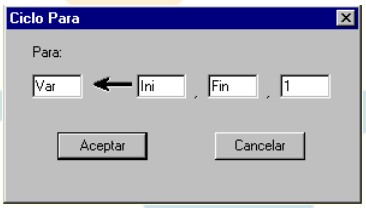
Si al evaluar la condición se obtiene el valor .F. la ejecución del algoritmo continuará a partir del objeto que sigue al Cierre.

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

Objeto Ciclo Para

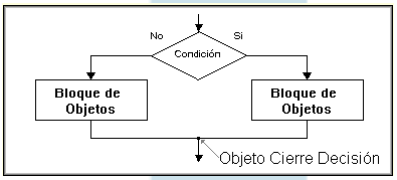


Su función es ejecutar un bloque de objetos mientras que la variable contadora no alcance el límite establecido por el valor final. El contador es siempre una variable de tipo de dato Real. Contiene además un valor inicial que será asignado al contador al iniciar la ejecución del ciclo, un valor final 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.

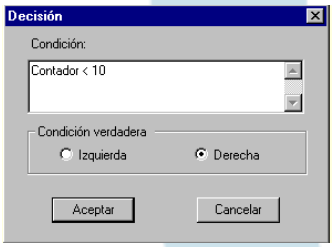
Objeto Decisión



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.

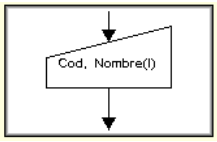
El objeto Decisión esta asociado a dos bloques de objetos ubicados a lado y lado de este, y un objeto Cierre Decisión ubicado a continuación de ambos bloques.



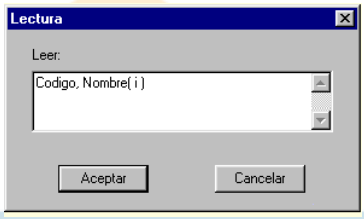
Si al evaluar la condición se obtiene el valor lógico .V., se ejecuta el bloque rotulado con la palabra Si, en caso contrario se ejecuta el bloque rotulado con No. En ambos casos la ejecución continua en el objeto Cierre Decisión.

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.

Objeto Lectura

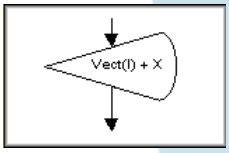


El objeto Lectura permite la entrada de valores constantes desde el teclado y se los asigna a campos variables. Podrá ser leída cualquier cantidad de variables utilizando un objeto Lectura. Al ejecutarse, el objeto despliega un cuadro de diálogo por cada variable presente en la lista, este cuadro de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

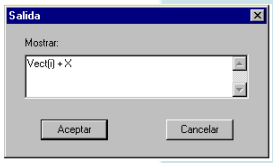


El cuadro de diálogo para la edición del objeto contiene un espacio para ingresar una lista de variables separadas por comas. Debe existir por lo menos una variable.

Objeto Salida

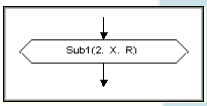


El objeto Salida muestra valores por pantalla. Puede ser visualizada cualquier cantidad de valores utilizando un objeto Salida. Al ejecutarse, este objeto evalúa cada una de las expresiones que contiene y despliega un cuadro de diálogo que muestra el valor obtenido en cada una de las expresiones en su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

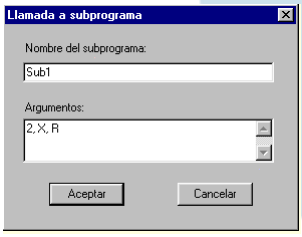


El cuadro de diálogo para la edición del objeto contiene un espacio para ingresar una lista de expresiones separadas por comas. Debe existir por lo menos una expresión.

Objeto Llamada

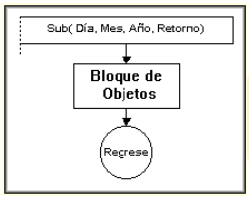


La función de este objeto es realizar una llamada a un subprograma, el cual debe encontrarse en el diagrama en edición. En la llamada deben encontrarse los argumentos que han de ser pasados al subprograma, la cantidad, el orden y el tipo de los argumentos deben coincidir con los parámetros del subprograma.

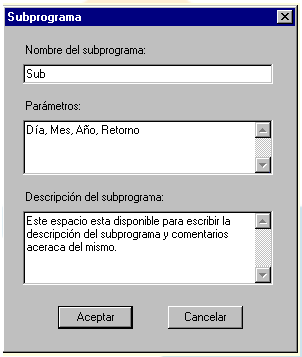


Una vez que el subprograma haya sido ejecutado la ejecución continuará en el objeto siguiente a la llamada.El cuadro de diálogo para la edición de este objeto contiene el espacio para el nombre del subprograma a llamar y el espacio para la lista de argumentos. Dichos argumentos deben estar separados por comas.

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.

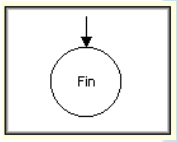


El cuadro de diálogo del objeto Subprograma contiene un espacio para la descripción o comentarios acerca del mismo ; contiene un espacio para el nombre del subprograma y un espacio para los parámetros. Estos parámetros (si existen) deben estar separados por comas. El nombre de un subprograma debe comenzar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

Ejemplo: Factorial , Leer , Sub1 , sub_programa.

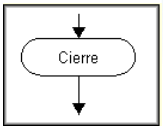
No se tiene en cuenta la diferencia entre mayúsculas y minúsculas para el nombre de un subprograma, es decir, SUB equivale a sub.

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.

Objeto Cierre Ciclo

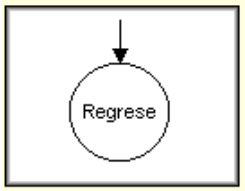


Este objeto delimita el cuerpo de un ciclo, al culminar la ejecución del ciclo el control se transfiere al objeto que sigue al objeto Cierre Ciclo.

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.

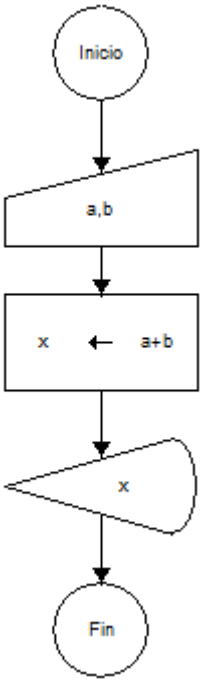
Objeto Regrese



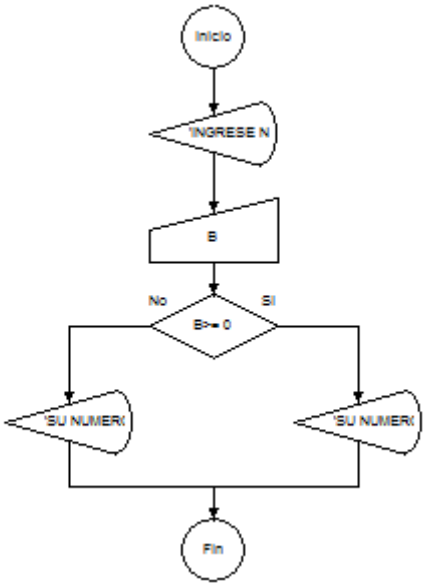
Este objeto junto con el Objeto Subprograma, delimita el cuerpo de un subprograma. La ejecución de este objeto transfiere el control al objeto que realizó la llamada.

Imágenes de ejercicios realizados con DFD

Vista ejercicio suma de 2 numero



Ejercicio para determinar si un número es positivo o negativo



Lección 5: Ejercicios de verificación

Realizar el análisis, diagrama de flujo, prueba de escritorio y codificarlos con la herramienta DfD , 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 cuál 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 perita 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 perita 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:

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

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.

5. CAPITULO 5: ALGORITMOS

Introducción

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

Lección 6: Origen de los Algoritmos

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 l2th 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). “

Lección 7: Definición de Algoritmos

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 está 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 realizo, 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)

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 cuero 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 cometario 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 cundo decimos (“por favor ingrese un número”), en ese momento somos explícitos con lo que queremos hacer,
- ✓ Cuando se realiza el planteamiento de los algoritmos es **importante** definir las variables y el tipo de dato que estas tienen, en el ejemplo anterior se puede apreciar este punto en la línea 3, donde se definen 3 variables de tipo entero (mirar apartado de variables unidad 1).

Resumiendo

Escriba ->sirve para dar a conocer mensajes o resultados

Lea -> permite capturara información

Entrada ------→ proceso-----→Salida

Lea Escriba

PSEINT

Al igual como se trabajo en el capitulo referente a diagramas de flujo, en este apartado incluiremos una herramienta que nos permite codificar los algoritmos que se propongan como parte de los talleres incluidos los realizados en el apartado anterior. Esta herramienta se denomina PSEINT, desarrollada por Pablo Noarra, su correo es zas k ar_@hotmail.com , se puede descargar de su sitio oficial http://pseint.sourceforge.net/ donde se encuentra los manuales o de http://ivan.lopezortiz.googlepages.com/algoritmos

En este mismo sitio puede mirar dos videos de cómo se codifica y ejecuta un programa.
El primero hace referencia a la suma de 2 números y el segundo a determinar cuál de 2 números es el mayor.

Nótese: que para este programa no es necesario realizar la definición de variables ni el tipo de ellas

Lección 8: Estructuras de selección

Las estructuras de selección son estructuras de control utilizadas para la toma 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
```

Ejemplo Determinar si un número es positivo

```
1. algoritmo Positivo  // nombre del Algoritmo
2. var
3. a: entero            //  Capturar el valor del número desconocido
4. inicio
5.    Escriba("Por Favor entre un número")   //  Mensaje
6.    Lea(a)                                 //   Captura Valor
7.    si (a>0)        // Condicional para determinar si el # es positivo (todos los > 0)
8.       escriba("El Número es positivo ")//  Mensaje de aviso que
9.     fin_si // Toda condición termina con un fin de condición
10. Fin   // todo algoritmo tiene un fin porque una de las características es que es
    finito
```

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
```

El siguiente ejemplo ilustra el manejo de condicionales

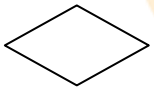
Ejemplo 2

Determinar cuál 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 ejercio, la sentencia si_no representa caso contrario

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

La estructura de **selección doble en cascada** está 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
   ….
```

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?

Existe otra forma de representar las sentencias de selección en cascada, es mediante la utilización de la sentencia CASE (Caso), esto es para cada condicional se representa mediante un caso, retomemos el ejerció anterior y analicemos como se construye con esta instrucción

Ejemplo empleado Case

```
1.  Algoritmo mayor_v1_case
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.    Case 1 (a=b)
10.       escriba ("los Números son Iguales");
11.       Case 2 (a>b)
12.       escriba ("El mayor es :", a);
13.       Case 3 a<b
```

```
14.     escriba(“el mayor es :”,b);
15.   fin_case
16. fin
```

Como se puede observar la sentencia case permite evaluar en cascada las condiciones requeridas. Ahora es decisión de cada uno su utilización.

Ejercicios de Verificación

Desarrollar mediante algoritmos los ejercicios propuestos en el apartado de verificación de los diagramas de flujo. (Ejercicios de Verificación diagramas de fluido).

Profundización en los temas :

Tipos de instrucciones

Instrucciones de Asignación

Instrucciones de Entrada

Instrucciones de Salida

Instrucciones de Decisión

Lección 9: 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:

1.Ciclo 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
```

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 explicito, lo mejor es realizar un ejercicio donde se comprenda todos los conceptos mencionados hasta el momento.

Ejercicios

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

Análisis

1. los números naturales se tomaran 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

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

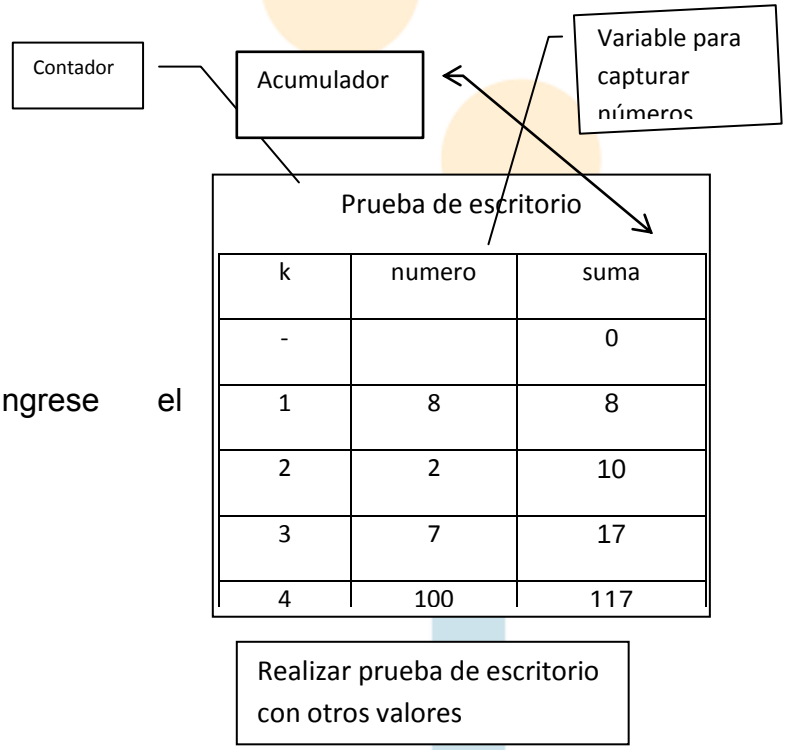
Análisis

1.- Para este caso se saben cuántos números serán los sumados, por lo cual se recomienda utilizar nuevamente el ciclo para

2.- No se sabe cuáles 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)

1ínea 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	kn	numero	suma
-	0		0
1		8	8
2		2	10
3	1	-7	
4		100	110

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

Ejercicios De Verificación

Desarrollar el siguiente taller y codificarlo en **PSeint**

1.-Realizar un algoritmo que permita leer 20 temperaturas (grados Cº) 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, …, 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 estudiante s 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. **Ciclo 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
```

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 eneros

Línea 6 -> se pude 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

....

3. **Ciclo 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 sillon dejo de caminar y he salido del ciclo).

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 realza 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.

Una forma de controlar un bucle es mediante una variable llamada **CONTADOR**,

Lección 10 .Contador y Acumuladores

Contador:
Es una variable cuyo valor se incrementa o decremento en una cantidad constante en cada repetición que se produzca.

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.

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

1.-determine cuál 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 estudiante s 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 estudiante 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 está 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%.

6. CAPITULO 6: SUBPROGRAMA O MODULO

Introducción

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

- Un subprograma o modulo es un trozo de código que tiene
  - Entradas
  - Salidas
  - Instrucciones

Es otro programa "prendido" al programa principal

Lección 11: Funciones

Una función se considera como 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.

Puesto que nuestra herramienta de programación va a ser el lenguaje C# se hará una introducción al uso de de funciones aunque este tema será estudiado en profundidad en el curso de introducción a la programación

En esquema un programa dividido en módulos será:

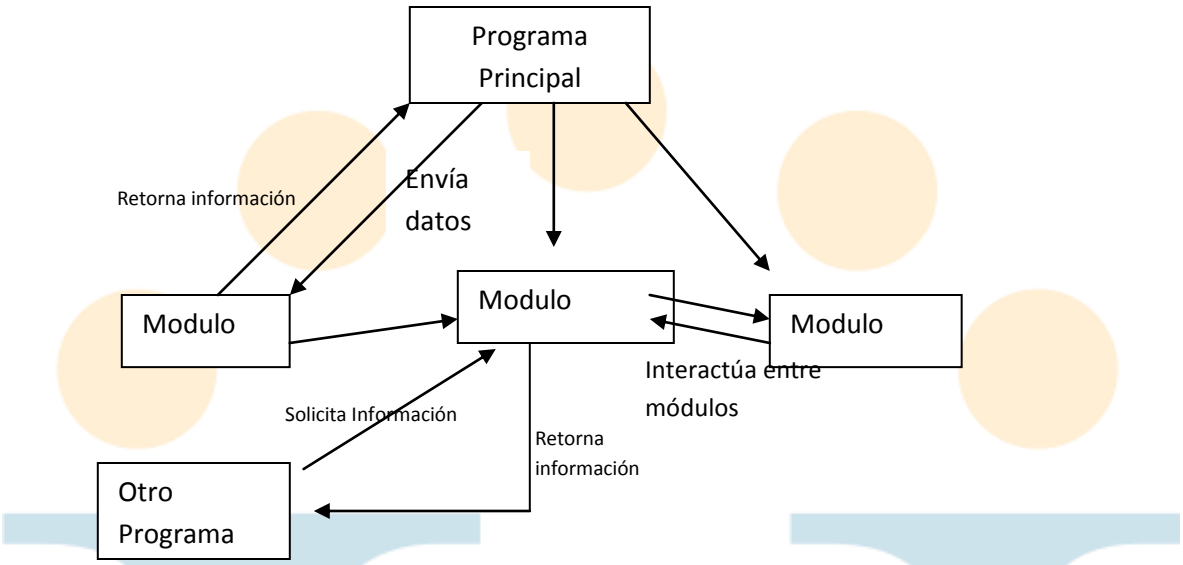


Figura16: Funciones

ograma principal que haga el llamado a cada una de las funciones que este necesite y a su vez un modulo o función puede llamar a otro. Es de destacar que en algunos casos de la figura se puede apreciar que existen flechas en doble sentido es decir se envía información de un lado o del programa principal y este envía información de retorno.

Lección 12: procedimientos

A pesar que muchos de los lenguajes de programación o ara ser mas exactos todos aquellos derivados de C o C++, emplean solamente funciones, existen otros que también pueden utilizar *procedimientos*, la diferencia esencial entre una función y un procedimiento radica en que este último es un subprograma que realiza una o varias tarea específicas y Puede recibir cero o mas valores del programa que llama y devolver cero o mas valores a dicho programa, en el caso de una función, esta puede recibir varios argumentos o valores y retornar cero o máximo un valor.

La declaración de de los procedimientos es similar a las funciones

Procedimiento nombre (lista de parámetros)

Instrucciones

……

Fin procedimiento

Los parámetros tienen el mismo significado que en las funciones

En ambos casos es importante tener en cuenta la forma de pasar los datos, que pueden ser por valor o por referencia

Lección 13: Paso de parámetros por valor

El paso de de parámetros por valor es el más utilizado por los lenguajes de programación, la gran ventaja de esto es que los valores se transforma en variables para ser trabajadas en los procedimiento o funciones, por ejemplo los valores 5,41 y 2 se transforman en X,Y,Z respectivamente cuando se ejecuta el procedimiento, en este sentido es de desatacar que los cambios que se produzcan dentro de la función invocada, esta no tiene cambio directo en los argumentos originales

Ejemplo

A=5, b=7;

Llamar a función suma (a, 20, b*2+6)

Función suma (R entero: X, Y ,Z)

Entonces la variable X toma el valor de a (5)

La variable Y toma el valor de 20

Y la variable Z toma el valor de 20 (7*2+6)

Y todo lo que se haga dentro de la función suma no afectara para nada las variables A,B

Lección 14: Paso De Parámetros Por Referencia

Para el caso de los parámetros por referencia, esta característica existe también en muchos de los lenguajes que manejan memoria dinámica, la característica de este radica en la idea lógica de las variable tienen una posición de memoria asignada desde la cual se puede asesar o actualizar los valores para este caso los parámetros se denominan parámetros variables.

La definición de los procedimientos se realiza de acuerdo al lenguaje de programación, para el caso de los algoritmos, también se hace una definición previa en la cabecera y luego su construcción al final del programa pricipal

Lección 15: Construcción de un proyecto

Para comprende 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 numero 0.

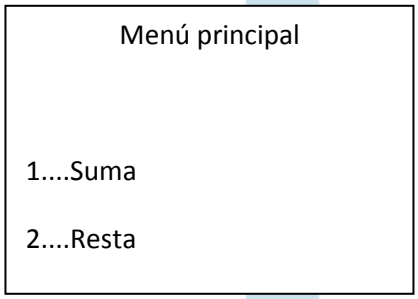
Análisis

1.- ya sabemos cómo se hacen cada una de las operaciones,

2.-sabemos cómo se comportan los ciclos

3.-manos a la obra con funciones

Se pide un menú[18]



Solución

1 Algoritmo menú

2 Var

3 dato1, dato2, op: entero;

4 inicio

5 op=1;

[18] Menú: Serie de opciones de donde se debe tomar una para realizar una acción

```
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);
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”);
```

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

```
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);
```

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

Retorna la respuesta al programa principal

```
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.
```

Notas finales

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

UNIDAD 3

Nombre de la Unidad	**PROGRAMACIÓN EN 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# de Microsoft 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# porque, porque incluye al lenguaje el fundamento de la programación basada en 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 y la programación .net. Para abordar esta unidad es importante que el estudiante ya haya tenido contacto con la computadora, sobre todo 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. Es de aclara que el lenguaje es completamente visual, sin embargo gran parte de la unidad se desarrollara a modo de consola.
Justificación	Bueno, es hora de pasar de la teoría a la práctica, es decir de llevar los ejercicios a un verdadero lenguaje de programación, que para este caso se selecciono
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 y sobretodo nuevos lenguajes orientados a la programación web
Denominación de capítulos	**CAPITULO 7: LENGUAJE DE PROGRAMACIÓN C#** **CAPITULO 8: EJECUCIÓN DE UN PROGRAMA** **CAPITULO 9: INTRODUCCIÓN A APLICACIONES WINDOWS**

7. CAPITULO 7: LENGUAJE DE PROGRAMACIÓN C#

Introducción

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, C++ , o visual C++; a demás se trabajará, bajo el supuesto que el sistema operativo es Windows xp o superior ; lo que no significa que no se pueda explorar entornos de programación bajo Linux, (C++ bajo Linux.)

Es importante mencionar que se anexa un apartado de dedicado a la programación en entorno MsDos con la versión 3.3 de Turbo C que corre sin ningún problema en maquinas con sistema operativo windos

Lección 1: Contextualización

Lectura Historia C#

Para esta parte se tomara como apartes un documento publicado por José Antonio González Seco (josanguapo@hotmail.com), en su sitio web http://www.josanguapo.com/

"Origen y necesidad de un nuevo lenguaje

C# (leído en inglés "C Sharp" y en español "C Almohadilla") es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth (imagen superior izquierda) y Anders Hejlsberg (imagen inferior izquierda), éste último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi. Puede encontrar la bibliografia de estos personajes en:

Figura17: Wiltamuth - Anders Hejlsberg

http://www.elavefenix.net/biografias.aspx

Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el **lenguaje nativo de .NET**

La sintaxis y estructuración de C# es muy parecida a la de C++ o Java, puesto que la intención de Microsoft es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son comparables con los de Visual Basic.

Un lenguaje que hubiese sido ideal utilizar para estos menesteres es Java, pero debido a problemas con la empresa creadora del mismo -Sun-, Microsoft ha tenido que desarrollar un nuevo lenguaje que añadiese a las ya probadas virtudes de Java las modificaciones que Microsoft tenía pensado añadirle para mejorarlo aún más y hacerlo un lenguaje orientado al desarrollo de componentes.

En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la BCL usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el *.NET Framework SDK*

Lección 2: Características Del Lenguaje C++

Estas son algunas de las características y fortalezas del lenguaje C#, tomadas de la página oficial:

http://www.microsoft.com/spanish/msdn/vstudio/Express/VCS/Docs/Top10/top10.mspx

- ✓ **Documentación para principiantes** Aprende lo básico utilizando los tutoriales "Cómo se hace" incluidos que te conducen mientras creas tus primeras aplicaciones de Consola y para Windows.
- ✓ **Formularios Windows arrastrar y soltar.** Diseña fácilmente aplicaciones sencillas pero muy vistosas utilizando el diseñador de interfaz de usuario intuitivo arrastrar y soltar.
- ✓ **Kits de inicio.** Los kits de inicio RSS Screensaver y Movie Collection completamente integrados hacen que sea fácil y divertido el aprendizaje.
- ✓ **Snippets de código IntelliSense.**Los Snippets de código IntelliSense son porciones de código que rellenan los espacios en blanco. Simplifican la cantidad de código que debes escrcibir a mano al proporcionarte plantillas para porciones de código comunes como añadir una clase, propiedad o foreach loop.
- ✓ **Refactoring**Refactoring permite a los desarrolladores automatizar muchas de las tareas mas comunes a la hora de re-estructurar código.
- ✓ **ClickOnce** Publica y comparte automáticamente tus aplicaciones completas en Internet, tu red de área local o en CDs utilizando los ayudantes de implementación ClickOnce.

- ✓ **Depurador simplificado.** Usa Editar y Continuar para aplicar cambios dinámicamente al código mientras se está ejecutando. Utiliza consejos de datos (datatips) y visualizadores de depuración para ver el contenido de tus estructuras de datos.
- ✓ **Aplicaciones preparadas para el uso de datos.** Construye aplicaciones preparadas para el uso de datos que se conectan con SQL Server 2005 Express Edition. Usa ayudantes que te ahorran tiempo para conectar a servicios Web, objetos o bases de datos.
- ✓ **Totalmente personalizable.** Visual C# 2005 ofrece una configuración personalizable que te permite controlar cada aspecto de tu código, incluyendo las opciones de formato de nuevas líneas, espaciado, envolturas y atajos de teclado. Podrás incluso personalizar el colorido de nuevos tipos .NET como tipos de valor, delegados, enumeradores e interfaces.
- ✓ **Mejoras en el lenguaje de programación C#.** Aprovecha al máximo la potencia del lenguaje C# incluyendo nuevas construcciones como genéricos, tipos anulables, iteradores y métodos anónimos.

Otras características interesantes del lenguaje se pueden consultar en: http://www.clikear.com/manuales/csharp/c10.aspx

Lección 3: Instalación C#

Lo primero es descargar el programa, esta descarga se debe realizar de:

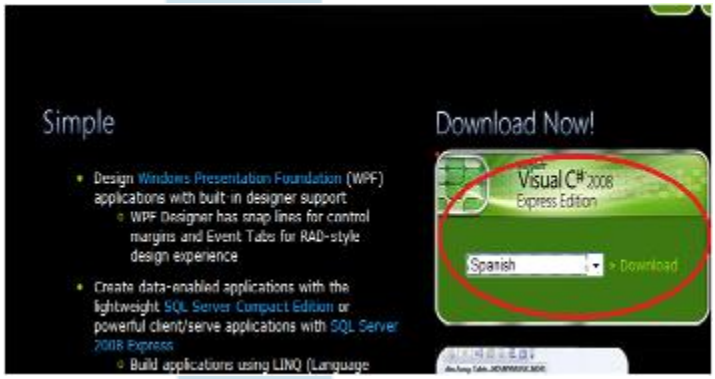


Figura18: Sitio descarga C#

http://www.microsoft.com/express/vcsharp/#webInstall para la versión 2008 aunque ya permiten descargar una versión 2010.

Una vez realizada la descarga el proceso de instalación es muy simple y similar a todos los paquetes desarrollados para Windows. Para tener en cuenta, la versión funciona por un periodo de 21 días, sin embargo para no tener problemas con esto lo único que se debe hacer es registrar el producto estos paso son sencillos, solo basta con tener una cuenta de correo en Hotmail, en caso de no tenerla recomiendo crear una y con esto ya se puede registrar y se tiene la versión completamente funcional y legal para ser utilizada

Una vez instalado ya se puede ejecutar el programa

A continuación se presenta la pantalla inicial de c# ver 2005, aunque es momento de aclara que también se puede trabajar con la versión 2008, sin ningún inconveniente.

No nos detendremos explicando el contenido de la pantalla de ingreso, por eso procedemos a crear nuestro primer proyecto

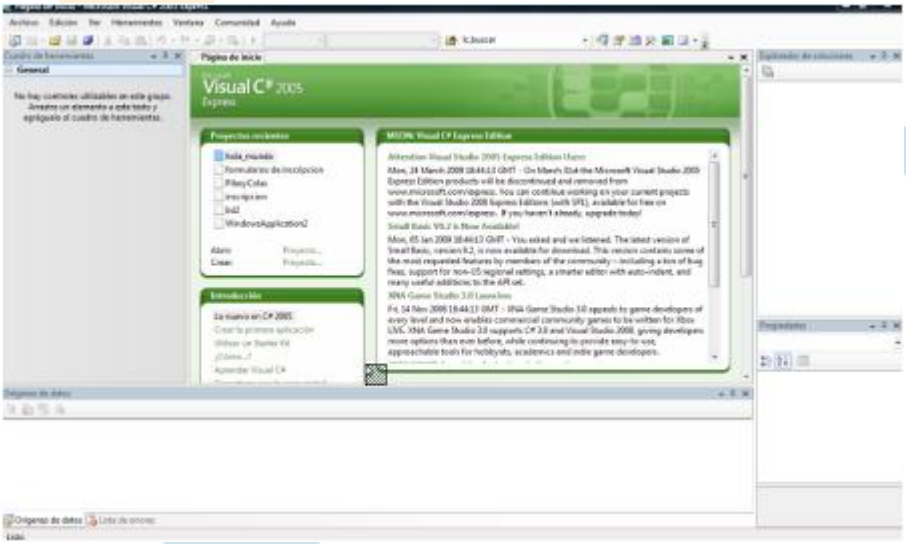


Figura 19: Administrador de proyectos

Lección 4: Primeros pasos

Nuestro primer proyecto será el saludo al mundo “Hola Mundo”, por tanto el nombre del proyecto será hola mundo. A demás utilizaremos este ejercicio para explicar un poco como está compuesto el editor de C#. Utilizando para ello la imagen de la parte inferior

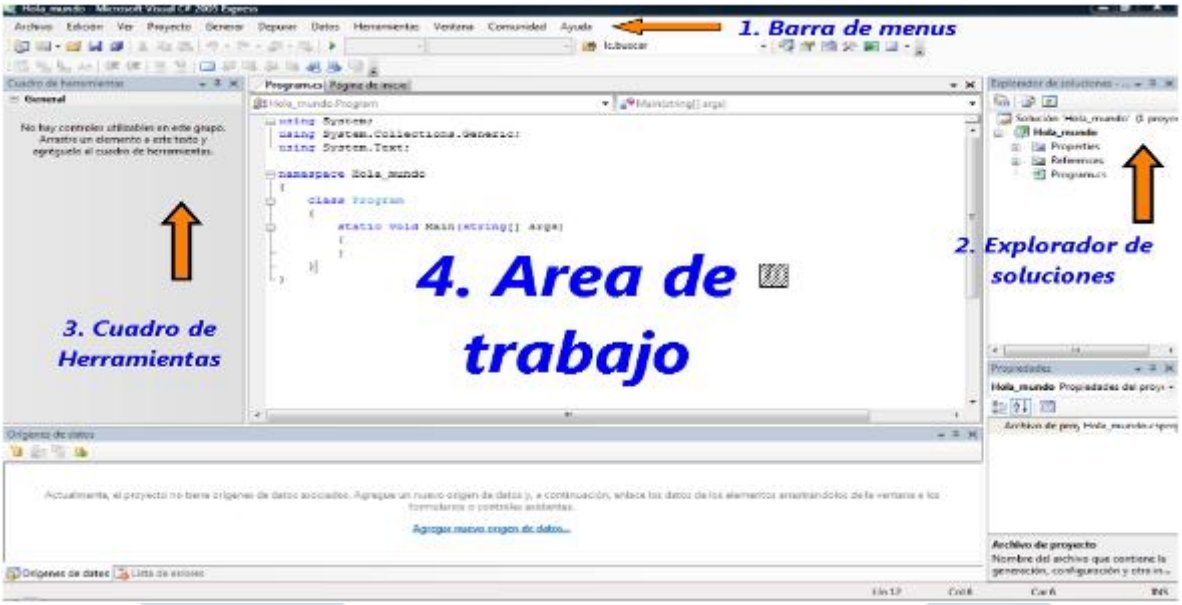


Figura 20: área de trabajo C#

1.- Barra de Menús: En esta área encontrara todas las opciones del editor, es similar a un procesador de texto como WORD, salvo que tiene unas opciones propias para configurar el compilador que las revisaremos más adelante.

2.- Explorador de Soluciones: en este espacio se encuentra disponible todo los archivos y formas (para programas visuales), que se generan con el proyecto

3.- Cuadro de herramientas: en este espacio se despliegan todas las opciones que tiene C# para la construcción de programas visuales, para el caso que nos atañe, esta opción no está disponible dado que la aplicación es tipo consola

4.- Area de trabajo: Es el espacio donde se llevará acabo toda la programación o la escritura del código necesario para obtener los respectivos resultados.

Volviendo a nuestro programa “hola mundo”, se puede apreciar que el área de trabajo ya trae incorporado parte del código

Las sentencias que están por defecto son:

```
1. using System;
2. using System.Collections.Generic;
3. using System.Text;
4. namespace Hola_mundo
```

```
5. {
6.     class Program
7.     {
8.         static void Main(string[] args)
9.         {

10.        }
11.    }
12. }
```

Explicación:

Las líneas 1, 2, y 3: se refiere a las librerías básicas necesarias para que nuestro programa funcione, se puede decir que son rutinas que contienen las instrucción necesarias como por ejemplo para mostrar lo mensajes, capturar valores entre otras posteriormente se revisara un poco más estas librerías
Línea 4: Namespace se utiliza para declarar un nombre que permite organizar el código en forma global permitiendo al compilador identificar el contenido de este. Para nuestro caso no lo vamos a modificar, solo queda con el nombre que le dimos por defecto “hola_ mundo”
Línea 5: { este da inicio a nuestro programa
Línea 6: Class Program: estructura propia de la programación orientada a objetos que será estudiada en el respectivo curso
Línea 7: Inicio a la clase
Línea 8 : inicio a nuestro programa propiamente dicho, teniendo en cuenta que la filosofía de todos los compiladores basados en C, son mediante funciones, esta será nuestra función principal
Línea 10,11,12, Es importante cerrar todas las directivas de inicio abiertas, esto lo trae por defecto, pero no para el caso de condicionales y ciclos que se revisará en uno de los apartados siguientes

Lección 5: Primer proyecto

Programa Hola_mundo :

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Hola_mundo
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
            Console.Write("Hola Mundo");
            Console.ReadKey();
        }
    }
}
```

Análisis: a la explicación dada anteriormente solo se ha agregado dos nuevas líneas:

```
Console.Write("Hola Mundo");
Console.ReadKey();
```

Console.Write : son instrucciones propias del lenguaje que le informan al compilador que se enviará un mensaje al monitor, para el primer las palabras HOLA MUNDO, ojo esta instrucción se debe encerrar entre comillas. Y la siguiente instrucción Console.ReadKey(); lo que hace es esperar que el usurio presione una tecla para continuar, lo que permite o da tiempo para mirar el resultado.

Observación adicional: es importante destacar que toda instrucción termina en punto y coma (;).

8. CAPITULO 8: EJECUCIÓN DE UN PROGRAMA

Introducción

En este capítulo se mira de una manera breve como se compila y ejecuta un programa, la definición de variables, el trabajo con condicionales y la utilización de los ciclos que como ya se menciono anteriormente son estructuras que permiten repetir secuencialmente una variedad de instrucciones. Los ciclos que se verán en este apartado son: for, while, do..while. Todo esto con la propuesta y realización de ejercicios básicos que permitan la conceptualización de los elementos fundamentales.

Lección 6: Compilación y Ejecución

Son dos procesos que están de la mono en la puesta a punto de un programa

Depuración

El primer paso es la depuración del programa, que cumple dos funciones uno encontrar posibles errores de sintaxis y por otro lado ralizar la compilación del código fuente. Una vez terminado la codificación del programa se procede a realizar la compilación del mismo, esto lo realizamos utilizando el menú depurar y la opción iniciar depuración que aparte de compilar permite encontrar los posibles errores de programación (ojo errores de sintaxis no detecta errores lógicos)

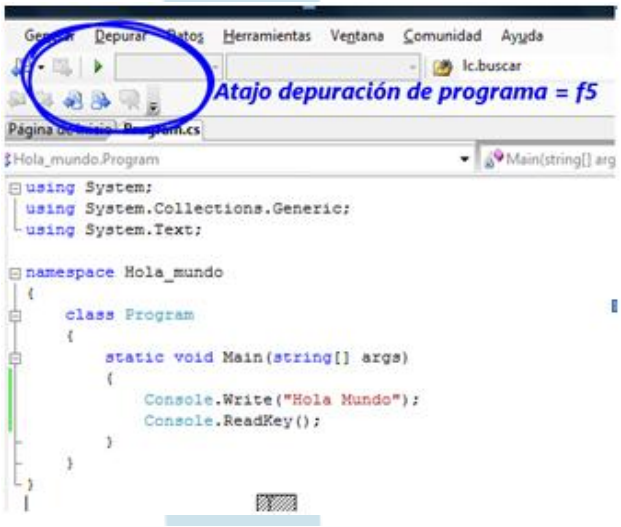


Figura21:Ejecución

Ejecución:

Con el mismo proceso mencionado anteriormente se realiza la ejecución del programa y se obtiene el resultado, para este caso será en una pantalla "negra", (y esta es una de las razones por las que estamos trabajando en forma de consola), con el resultado del programa, solo resta presionar una tecla para regresar al editor y realizar los ajustes necesarios si es del caso.

Figura22: vista compilado

Definición de variables

La definición o declaración de variables en C#, tienen las mismas consideraciones de las vistas al inicio de este modulo, por consiguiente es de interés recordarlas o volver a revisar

C# puede declara variables en cualquier lugar del programa pero se recomienda realizar su declaración a continuación de la definición de la función principal. Es importante tener en mente cuales son los diferentes tipos de variables que reconoce C#, para esto puede mirar el siguiente cuadro

Tabla No 6: Rango de valores

C# Tipo	Rango
sbyte	-128 a 127
short	-32768 a 32767
int	-2147483648 a 2147483647
long	-9223372036854775808 a 9223372036854775807
byte	0 a 255
ushort	0 a 65535
uint	0 a 4294967295
ulong	0 a 18446744073709551615
float	Aprox. $\pm 1.5 \times 10^{-45}$ a $\pm 3.4 \times 10^{38}$ con 7 decimales
double	Aprox. $\pm 5.0 \times 10^{-324}$ a $\pm 1.7 \times 10^{308}$ con 15 o 16 decimales
decimal	Aprox. $\pm 1.0 \times 10^{-28}$ a $\pm 7.9 \times 10^{28}$ con 28 o 29 decimales
char	Cualquier carácter Unicode
bool	true o false

Para identificar como se definen las variables presentemos el siguiente programa Ejemplo

Realizar la suma de 2 números cualesquiera

Bueno este ejemplo lo hemos realizado con anterioridad, por lo tanto ahora solo procedemos a codificar en c# y explicar algunas líneas en particular

```
namespace suma
{
    class Program
    {
        static void Main(string[] args)
        {
            1  int a, b, suma;
            2  Console.Write("Por favor entre un número");
            3  a = int.Parse(Console.ReadLine());
            4  Console.Write("Por favor entre un número");
            5  b = int.Parse(Console.ReadLine());
            6  suma = a + b;
            7  Console.Write("La suma de los números es {0}",suma);
            8  Console.ReadKey();

        }
    }
}
```

Explicación

Línea 1: se definió tres variables de tipo entero (int), es en este espacio donde se deben definir las variables con el tipo correspondiente y de acuerdo al cuadro antes mencionado

Línea 2 y 4: Mensajes por pantalla donde se pide el ingreso de un valor

Línea 3 y 5: Se captura el valor de la variable, es importante detenernos en este punto dado a la forma como se capturan los valores de tipo numérico primero a la variable se le asigna un valor que es necesario convertir (int.Parse), dado que la instrucción Console.ReadLine solo permite leer datos de tipo carácter, y de ahí es importante tener claro como convertir cualquier tipo de dato.

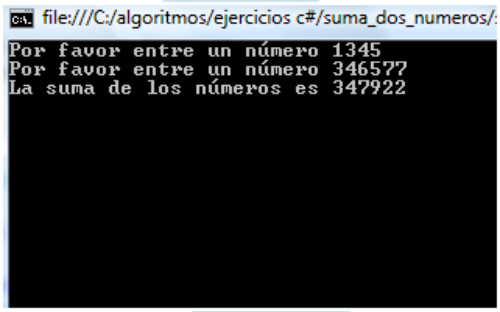


Figura23:vista compilado

Línea 7: muestra por pantalla el resultado de la operación, pero es importante observar que para desplegar un valor es importante utilizar la directiva { 0 } , que le indica al compilador que se mostrara un valor numérico, posteriormente se miraran otras directivas

Ejercicios de verificación

Es hora de desarrollar unos pequeños ejercicios para determinar el grado conceptualización

- Desarrollar un programa que permita encontrar el área de un triangulo
- Escribir un programa que le pida a un usuario dos números y que muestre la suma de los números, el producto, la diferencia del primero con el segundo, y el cociente de ambos.
- Realizar un programa que permita convertir de grados centígrados a farenheit y a kelvin
- Realizar un programa que permita ingresar 3 números y como resultado retorne su promedio
- Realizar un programa que permita encontrar el area de un cuadrado

Lección 7: Condicionales

Son instrucciones que permiten ejecutar bloques de instrucciones sólo si se cumple una determinada condición.

La **instrucción if** permite ejecutar ciertas rutina sólo si da una determinada condición como verdadera.

```
if(<condición>){
  <instruccionesIf>}
else{
   <instruccionesElse>}
```

"El significado de esta instrucción es el siguiente: se evalúa la expresión <condición>, que ha de devolver un valor lógico. Si es cierta (devuelve **true**) se ejecutan las <instruccionesIf>, y si es falsa (**false**) se ejecutan las <instruccionesElse> La rama **else** es opcional, y si se omite y la condición es falsa se seguiría ejecutando a partir de la instrucción siguiente al **if**. En realidad, tanto <instruccionesIf> como <instruccionesElse> pueden ser una única instrucción o un bloque de instrucciones."

Manos a la obra

Vamos a realizar un ejercicio práctico y sobre él se explicaran cada una de las acciones y condiciones que se deben tener en cuenta en la utilización de condicionales

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

Solución, a continuación se muestra parte del código

```
        static void Main(string[] args)
        {
            int a, b;
            Console.Write("Por favor entre un número");
            a = int.Parse(Console.ReadLine());
            Console.Write("Por favor entre otra un número");
            b = int.Parse(Console.ReadLine());
1           if (a > b)
2           {
3               Console.Write("El mayor de los números es {0}", a);
4           }
5           else
6           {
7               Console.Write("El mayor de los números es {0}", b);
        }
            Console.ReadKey();
    }
```

Analicemos las líneas caso de estudio

Línea 1: las estructuras condicionales son simples y similares a lo visto en el tema de algoritmos

Línea 2: después de cada condicional es importante abrir una llave { que indique el espacio de código que abarca la condición (es importante tener en cuenta que si después de la condición solo hay una instrucción como es el caso del presente programa no es necesario la llaves, pero tampoco es un error).

Línea 4: se debe cerrar en su debido lugar cada apertura que se haga con una llave

Línea 5: Else = en caso contrario, es decir si no es blanco es negro y nos evitamos un condicional.

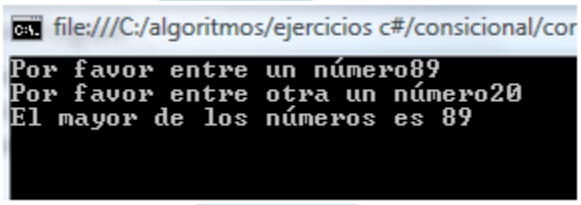


Figura24:Vista compilado

Una vez se ejecute el programa obtenemos el resultado esperado como se puede ver en la imagen de la izquierda

Ahora nos responderemos la pregunta que nos realizamos permanentemente: que pasa si los números son iguales

Ejercicio: desarrollar un programa que permita determinar cuál es el mayor de dos números o si estos son iguales

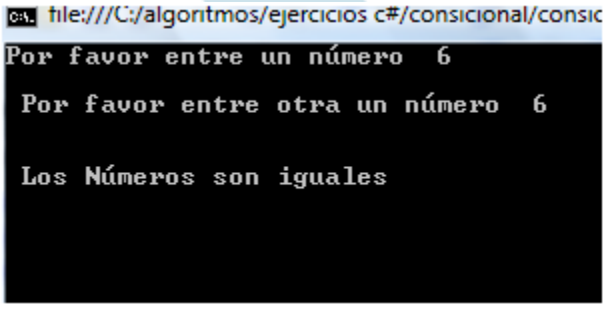
Solución:

```
int a, b;
          Console.Write("Por favor entre un número  ");
          a = int.Parse(Console.ReadLine());
          Console.Write("\n Por favor entre otra un número  ");
          b = int.Parse(Console.ReadLine());
1          if (a == b)
2          {
3            Console.Write("\n\n Los Números son iguales");
4          }
5          else
6          {
7            if (a > b)
8            {
9               Console.Write("\n\n El mayor de los números es {0}", a);
10           }
11           else
12           {
13              Console.Write("\n\n El mayor de los números es {0}", b);
14           }
15         }
            Console.ReadKey();
        }
    }
```

Explicación:
Linea 1: observese que se agrego una nueva etructura condicional que permite evaluar si las varibales son iguales, es importante tener encuenta que se utiliza el doble igual == en el caso de realizar comparación y cuando solo exista un sigono = es asignación, este caso se utiliza para asignar un valor a una variable (x=5). Si es del caso determinar un *diferente l*o hacemos if (a!=b) con el signo de admiración

Línea 2 y 4, se incluyo un inicio y un final para que ejecute únicamente esa instrucción

Línea 5: se incluyo un nuevo else que permite evitar que al ejecutar el programa, este evalué las instrucciones siguientes (como ejercicio usted puede retirar este else con sus respectivas llaves líneas 6 y 15)



Observación: nótese que en algunas instrucciones se ha agregado unos comodines \n , lo que hace esto es un salto de línea para trabajar de una manera más ordenada.

Figura25: Vista compliado

En la siguiente tabla se muestra otra serie de comodines que se pueden utilizar para mejora la presentación.

Tabla de códigos secuenciales c#
Estos son algunos de los comodines o códigos secuenciales que pueden ser utilizados en C#

Tabla No 7: Códigos secuenciales

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

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 o case*
- 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 la sentencia que permite hacer esto).
- 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 número 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 m3 a litros y centímetros cúbicos.
- Diseñe un programa que Lea dos puntos (x, y) y calcule la distancia entre ellos
- Elabore un programa 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 matrícula 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 estudiante podrá aspirar a la matrícula 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 cuánto 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 del día	**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 => dos lados iguales
  - Escaleno => $A \neq B \neq C$
  - Equilátero => $A = B = C$

- Con relación a sus ángulos un triángulo puede ser:

  - Rectángulo => Un ángulo recto
  - Acutángulo => 3 ángulos agudos
  - Obtusángulo => 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

Lección 8: Ciclo for

En C# y en casi todos los lenguajes de programación este ciclo es uno de los más usados para repetir una secuencia de instrucciones sobre todo cuando se conoce su inicio y final o la cantidad exacta de instrucciones a repetir

Su formato general es:

for (inicialización; condición; incremento)

{

Instrucciones

} // fin del ciclo

En su forma sencilla y la más utilizada la inicialización se hace con una instrucción de asignación que carga una variable de control de ciclo con un valor inicial. Luego se condiciona indicando las veces que se repetirá

Por último El incremento es el que dice en que en “pasos” de cuando se harán los incrementos.

Es de destacar que tanto la variable de inicio como la condición y el incremento deben estar separados por punto y coma (;).

La mejor forma de entender esto es mediante un ejemplo

Ejercicio 1

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

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

Solución

```
  namespace ciclofor1
{
    class Program
    {
        static void Main(string[] args)
        {
  1           int k, suma = 0;
  2           for (k = 1; k <= 10; k++)
  3           {
  4               suma = suma + k;
  5           }
```

```
6           Console.WriteLine("\n\n el resultado de la suma de los 10
                              números es {0} ", suma);
8           Console.ReadKey();
9       }
10   }
11 }
```

Explicación

Línea 2: **for(k=1;k<=10;k++)**-→ este ciclo se divide en tres partes principales

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

La condición k < = 10; condición, de parada, para este caso que llegue a 10

El incremento k++; incremento, decimos que queremos incrementar la variable k en pasos de 1; se puede utilizar en sentido inverso k--, es decir decrementos

Línea 3 y 5 Como dentro del ciclo, no hay sino una instrucción, entonces no se requiere apertura ni cierre de llaves, pero para este caso lo vamos a utilizar lo cual no es un error

Línea 4: Es la representación básica de un acumulador.

En la página http://ivan.lopezortiz.googlepages.com/algoritmos Puede ver el video del funcionamiento paso a paso del ejercicio (video ciclo for1)

Ejercicio 2

Una pequeña variación al ejercicio anterior

Realizar la suma de 10 números cualesquiera e imprimir su resultado

```
static void Main(string[] args)
        {
            int numero,k, suma = 0;
            for (k = 1; k <= 10; k++)
            {
                Console.Write("por favor entre el {0} núemro ", k, " ");
                numero = int.Parse(Console.ReadLine());
                suma = suma + numero;
            }
            Console.WriteLine("\n\n el resultado de la suma de los 10
                              números es {0} ", suma);
            Console.ReadKey();
```

```
        }
```

Explicación:

En este caso, el ciclo si abre llaves, por tener más de una instrucción, a demás es importante ir ingresando cada uno de los números e irlos acumulado

Ejercicio

Es importante combinar los siclos con otras instrucciones, por ejemplo con condicionales, para lo cual se propone el siguiente 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

```
static void Main(string[] args)
      {
          int k, numero, suma=0, kn=0;

        for (k=1;k<=10;k++)
        {
            Console.Write("Por favor entre un el {0}° número  ",k);
                numero = int.Parse(Console.ReadLine());
              if (numero >=0)
                  suma=suma+numero;
              else
                  kn++;
        }
        Console.WriteLine("el resultado de la suma de los números
                           positivos es : {0}", suma);
        Console.WriteLine("la cantidad de números negativos ingresados es
                           :{0} ",kn);
        Console.ReadKey();
    }
```

Explicación

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

Ejercicios de verificación

1.-codificar los algoritmos del taller propuesto en el ítem relacionado con el ciclo para

2.-Profundizar y realizar ejemplos con sentencias de incrementos y decrementos

3.-Consultar la directiva de posicionamiento en la pantalla y para la limpieza de la misma, para darle ubicación y presentación a los programas

Lección 9: Ciclos while

En este ciclo el cuerpo de instrucciones se ejecuta mientras la condición permanezca como verdadera y en el momento en que la condición se convierte en falsa el ciclo termina, es importante mencionar que para que se ejecute las instrucciones este debe ser verdadero por lo menos la primera vez

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

while (<condición>) <sentencia>

Puede ser también

While (condición)

{

}

Lección 10: Ciclo do while :

Este ciclo es muy utilizado cundo queremos realizar filtros[19] y cuando deseamos que se permita el ingreso al ciclo al menos una vez

Ejemplo

Para entender mejor el funcionamiento de esos ciclos, lo mejor es demostarlo con un programa práctico.

Planteamiento: 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:

Gran parte de este análisis se realizo en el tema referente a ciclos, trabajado con algoritmos.

[19] Filtro: permitir el ingreso de datos dentro de un rango especificado

Con este ejemplo se utilizaran *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

```
1       int k;
2       float suma,nota,promedio;
3       suma = 0; k = 0; nota = 5; ;
4        while (nota != 0)
5        {
6             do
7           {
8             Console.WriteLine("entre una nota");
9             nota= float.Parse(Console.ReadLine());
10             if  (nota <0 || nota >5)
11               {
12                Console.WriteLine("Error. Inténtelo nuevamente");
13               }
14          }
15           while (nota<0  || nota >5);
16             if (nota !=0 )
17           {
18               suma=suma+nota;
19               k++;
20             }
21        }
22         promedio=suma/k;
23         Console.WriteLine("la cantidad de notas ingresadas son: {0}
                             ",k);
24         Console.WriteLine("El promedio de las notas es de : {0}
                             ",promedio);
25         Console.ReadKey();
```

Explicación General

Es el primer ejercicio de los realizados donde utilizamos un tipo de dato diferente al int, en este caso el float, con esto nos damos cuenta lo fácil de utilizar. También es de destacar la utilización de los conectores lógicos y se representan así

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

Para finalizar es importante mencionar queel ciclo **do** se cierra llaves l con un **while**, el cual termina con punto y coma

9. CAPITULO 9: INTRODUCCIÓN A APLICACIONES WINDOWS

Introducción

Hasta el momento y como lo mencionamos antes, solo hemos utilizado la consola para el desarrollo de aplicaciones, pero es monto de dar un vistazo a aplicaciones desarrolladas para Windows, no sin antes enfatizar que esta será solo una pequeña muestra con algunos ejercicios sin profundizar en muchos de los conceptos

Lección 11: Primeros Pasos

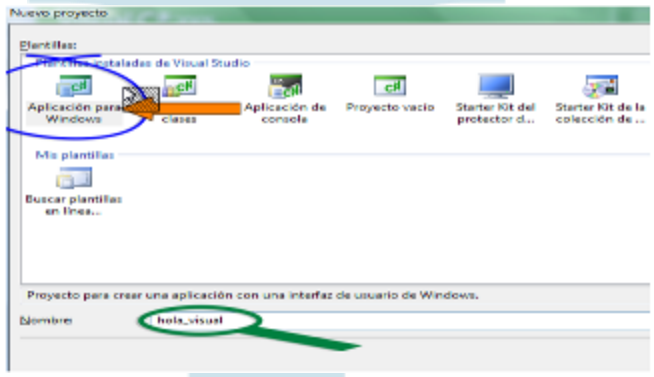


Figura26: Selección nuevo proyecto

Lo primero es iniciar un nuevo proyecto, pero en este caso seleccionamos un nuevo proyecto, pero ya no seleccionamos la opción consola si no que ejecutamos la opción “aplicación para Windows”, como lo podemos ver en la imagen de la izquierda. Es importante recordar la importancia de dar el nombre al proyecto

El ingreso

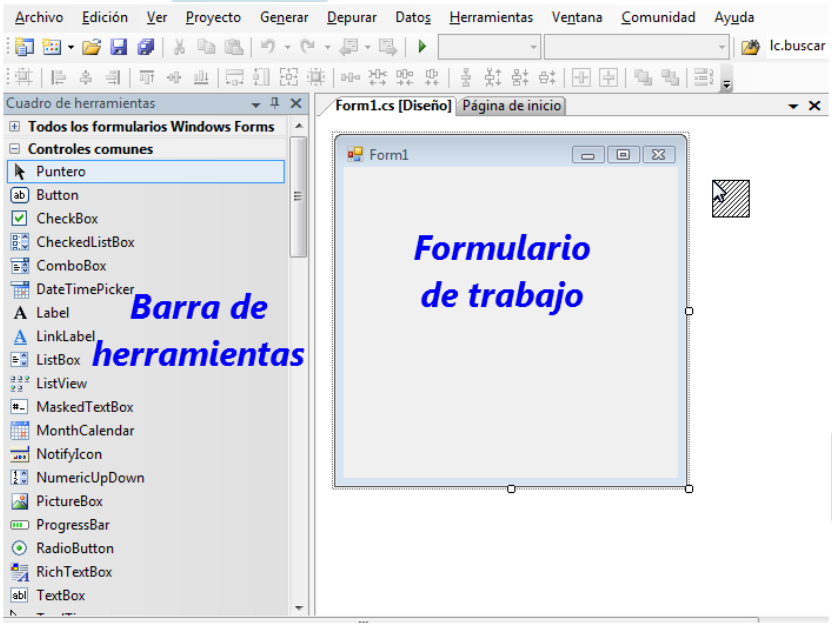
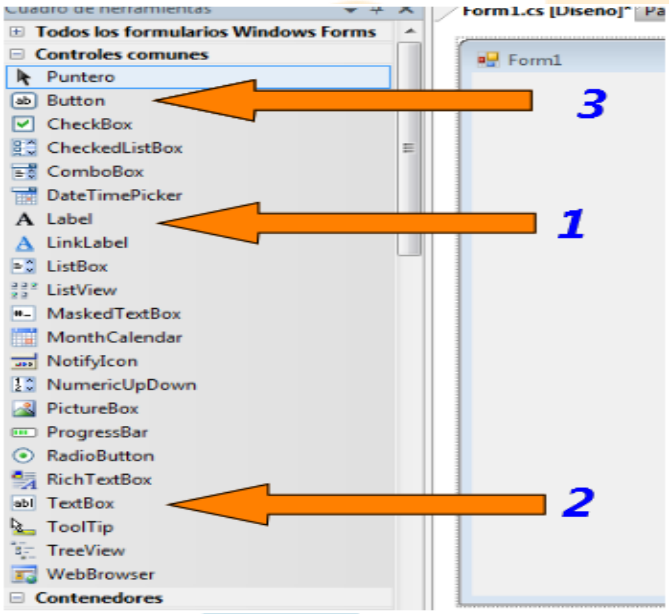


Figura27: opciones visuales

Una vez seleccionamos el proyecto entramos a nuestro proyecto con algunas diferencias que las aplicaciones tipo consola, en este caso ya tenemos habilitado la barra de herramientas y un formulario donde podemos pegar los “objetos” que requiere la aplicación. Para

explicar mejor las opciones lo mejor es realizar un ejercicio,

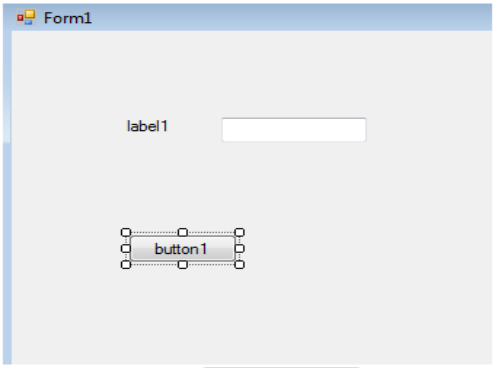
Ejercicio:



Se realizara un programa que pida su nombre y el programa genere un saludo de bienvenida.
Lo primero es seleccionar varios controles de la ventana de herramientas, para nuestro caso utilizamos

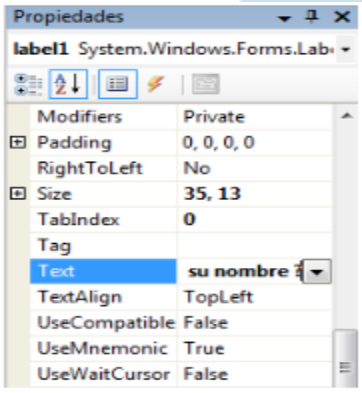
1- Label -> permite escribir mensajes
2- Texbox -> permite capturar información
3- Button -> permite ejecutar una acción

Figura28: Herramientas comunes



En la imagen de la izquierda se ve como queda la el diseño del formulario de trabajo, Posteriormente solo se debe hacer unos cambios básicos de nombre a los objetos y listo, en la presente imagen se presenta la ventana de propiedades donde puede cambiar las “propiedades de los objetos”,

Figura29: Diseño de formulario



en este caso se cambio el texto del label y del button. Como es puede ver en la figura (barra de propiedades)

Figura30:barra de propiedades

En la siguiente imagen se puede apreciar el diseño final

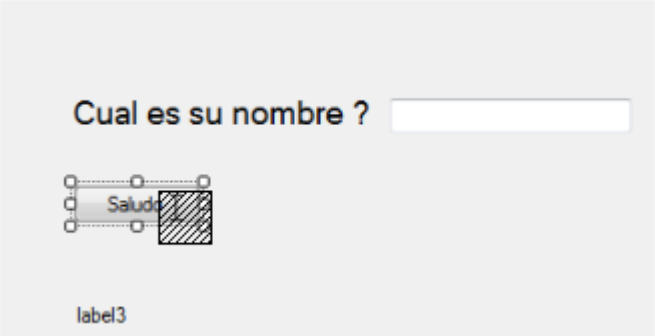


Figura 31: vista del diseño

Nótese que se agrego un nuevo label a nuestro proyecto, esto con el fin de mostrar la respuesta.

Ahora solo queda escribir el código, este se debe ejecutar como respuesta a una acción, por consiguiente el código se incluirá para este caso en el botton, esto se hace dando doble clic sobre este botton, y agregamos el código como podemos apreciar en la siguiente imagen

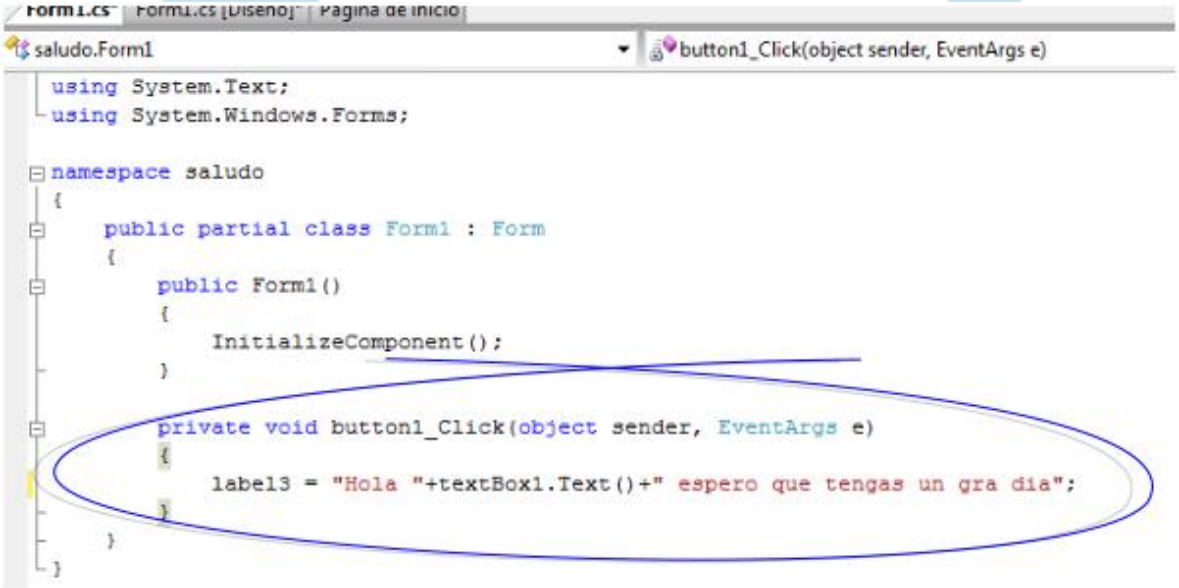


Figura 32:codigo saludo

Al momento de dar el doble clic el crea una función para este objeto, y solo agregamos el código para unir dos cadenas de caracteres, una con el nombre y otra con el saludo, tal como se aprecia en la imagen anterior

Por último solo resta ejecutar la aplicación y este proceso ya lo conocemos, simplemente con la tecla F5 y el botón Saludo y obtendremos el resultado final que se ve en la imagen que esta a continuación.

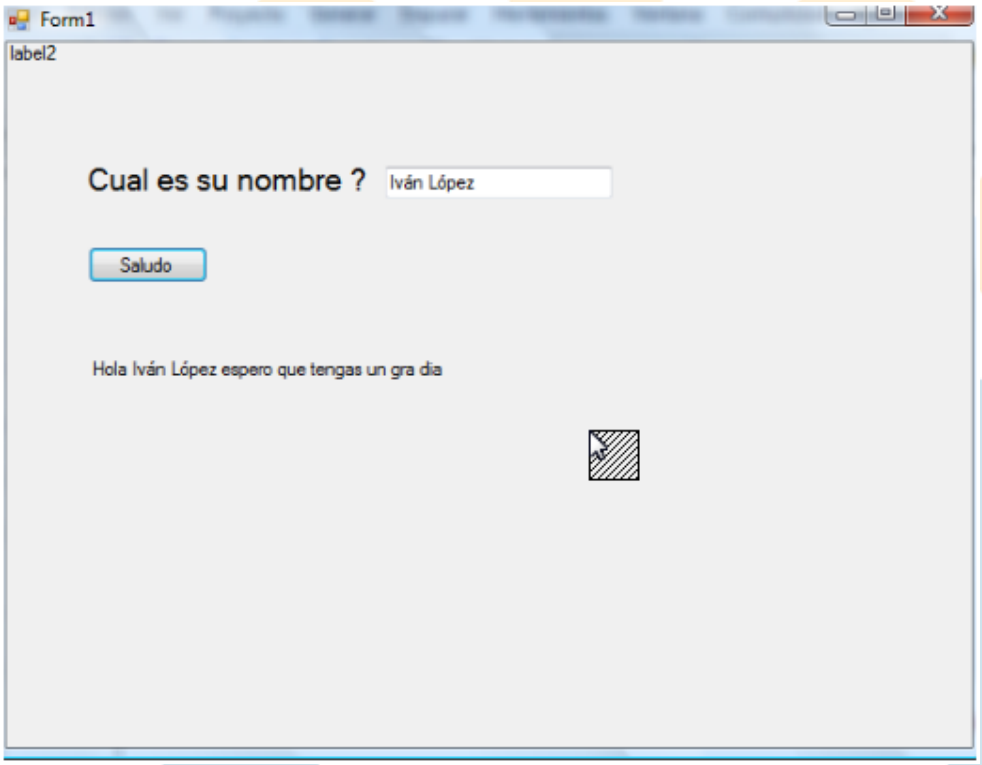


Figura 33:vista programa ejecutado

También puede mirar todo el proceso de creación en la página http://ivan.lopezortiz.googlepages.com/algoritmos , Video programa visual, si lo desea y espero que así sea puede profundizar mucho más en http://msdn.microsoft.com/es-es/library/kx37x362%28VS.80%29.aspx

Y las consultas que pueda realizar en internte.

Espero que el material aquí presentado haya sido de su interés y despierte la motivación para la construcción de futuros proyectos.

Lección 12: Descripción de los controles comunes

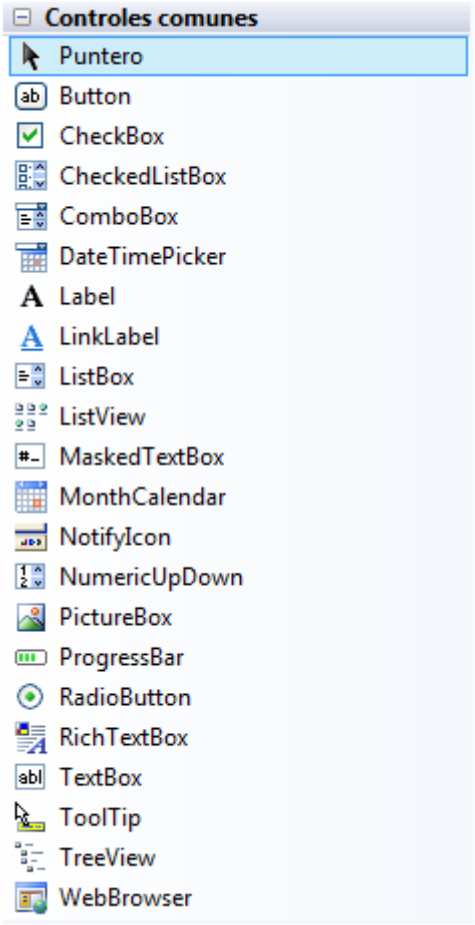


Figura34:Controles comunes

1-. Puntero: Permite seleccionar los objetos del formulario
2-. Button: Permite desencadenar eventos al presionar (dar clic)
3-. CheckBox. Permite Seleccionar o quitar una selección a una determinada opción
4.-CheckedListBox: Permite crear una lista de elementos para seleccionar uno de ellos o vario de ellos
5.-ComboBox: Crea una lista mostrar una serie de elementos
6.-DateTimerpicture: permite diseñar un calendario en varios formatos
7.-Label: Permite la creación de textos informativos
8.- LinkLabel: lo mismo que el anterior pero con la opción de link a una dirección web
9.- lo mismo que el #5, pero siempre permanece visible la lista
10.- ListView: permite crear una colección de elementos
11.-MaskedTextBox: permite crear una máscara para el ingreso o salida de datos, la máscara se refiere a un formato Ejemplo Fechas, Monedas, télefonos…
12.- MonthCalendar: Muestra calendario mensual
13.- NotifyIcon: permite crea un icono en la barra de tares de Windows, cuando este se está ejecutando
14.-NumericUpDown: permite crear una lista de números
15.-PictureBox: permite adicionar una imagen al formulario
17.-ProgressBar: permite crear un diseño para mirar el % de ejecución de un programa
18.- RadioButton: se diseña para que el usuario solo pueda seleccionar uno de varios elementos
19.-RichTextBox: Para crear objetos topo texto
20.-TextBox : Permite la entrada de datos

21.-Tooltip: permite crear ayuda cuando se pasa el puntero sobre un objeto
22.-TrreView: genera estructuras de árbol tipo directorio de Windows
23.-WebBrowser: Se puede generar páginas web, desde el formulario
Ahora realicemos un ejercicio aplicando alguno de estos controles.

Lección 13: Condicionales

Al igual como se ha tratado en todos los apartados anteriores, los condicionales se sutilizan de la misma forma, solo que ahora se pueden utilizar para seleccionar una "opción", de una serie de elementos que permita pasar el valor de grados centígrados a Kelvin o Fahrenheit

Lo primero es crear un nuevo proyecto, con los siguientes elementos:

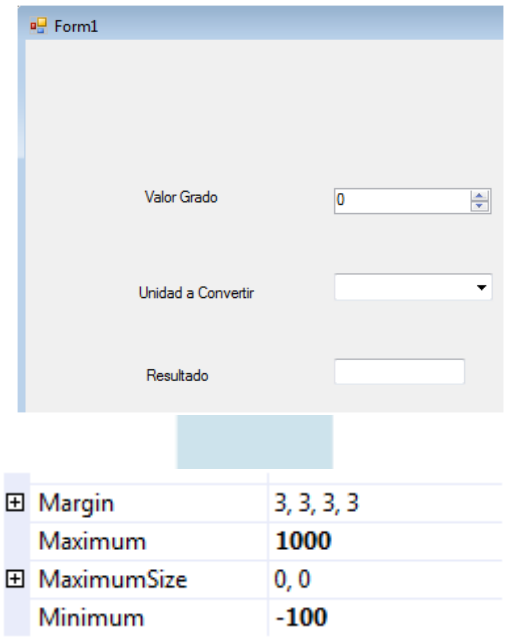


Figura35: Vista de diseño

Tres etiquetas label

Un NumericUpDown

Un ComboBox

Un TexBox

En la barra de herramientas cambiamos lo valores por defecto (textos) de los label a los que están en la figura, a demás modificamos lo valores del NumericUpDown que por defecto están en el rango de 0 y 100, para esto seleccionamos las opciones maximun y lo incrementamos a 1000 y el opción minimun a -100

Con esto se procede a agregar el código necesario (nótese), que no estamos utilizando un **button**, por consiguiente el código lo incluimos en la opción indexchange del ComboBox1, esto se hace dando doble clic sobre este objeto tal como se ve en la siguiente imagen

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{


}
```

Ahora se procede a ingresar el código respectivo,

```
float grados;

            if (comboBox1.Text == "Kelvin")
            {
                grados = float.Parse(numericUpDown1.Value.ToString());
                grados = grados + 273;
                textBox1.Text = (grados.ToString());
            }
            else
            {
                grados = float.Parse(numericUpDown1.Value.ToString());
                grados = (grados * 9 / 5) + 32;
                textBox1.Text = (grados.ToString());
    }
```

Lo primero se define una variable “grados”, para poder realizar las formulas

Luego se procede a preguntar por el valor que esta tomando el el objeto comboBox1, para coonpararlo con un valor base (Kelvi)

Nota importante, estos valores se incluyen en la lista desplegrable de comboBox1, dando clic en el objeto como se ve en la siguiente figura.

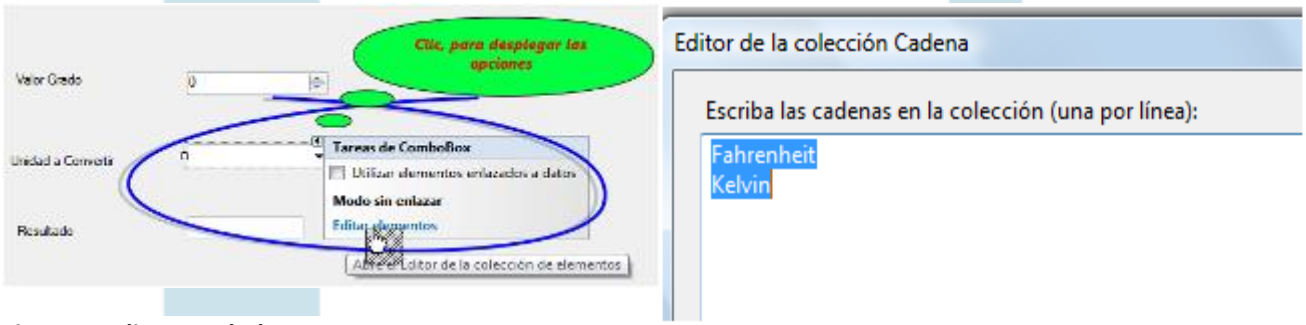


Figura36:Editor Combobox

Ahora se selecciona la opción “Editar elementos”, y se escribe la opciones que se desea aparezcan en el cuadro desplegable

Para este ejemplo solo se utilizan dos elementos, usted puede utilizar más elemento u opciones como se ve en la siguiente imagen

Fin de nota

Se continúa con el código y se ejecuta el programa como ya se ha explicado y el resultado es (ver fig):

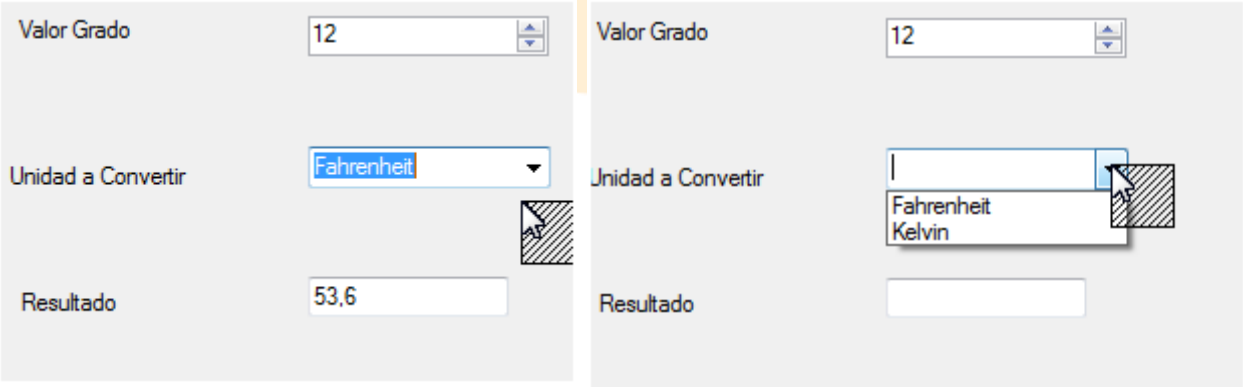


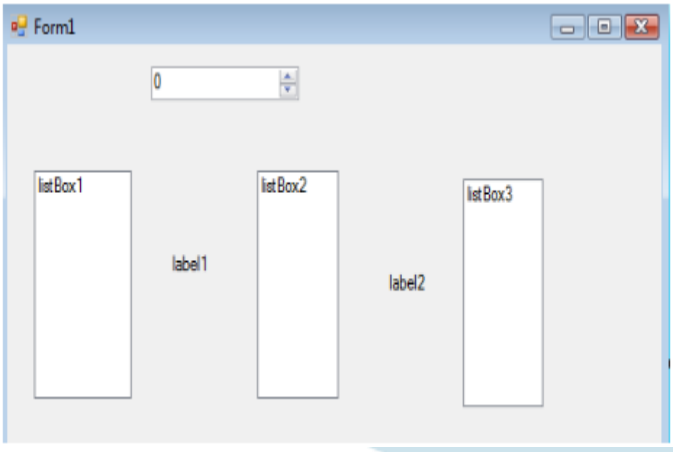
Figura37: Seleccion con comboBox

Ejercicios:

1. Realizar el mismo ejercicio incluyendo Grados Celsius
2. Cambiar la opción TextBox por RadioButton
3. Realizar ejercicios para encontrar áreas de diferentes figuras.

Lección 14: Ciclos

Al igual que en los apartados de este tema este se tratará mediante la codificación de un ejercicio que consiste en: desarrollar una aplicación que permita mostrar en forma visual una tabla de multiplicar



Para esto es necesario agregar los siguientes objetos

Dos etiquetas label

Un NumericUpDown

Tres ListBoxt1

Figura38: Varios objetos

Ahora un poco de diseño y el formulario debe quedar como se ve en la siguiente figura:

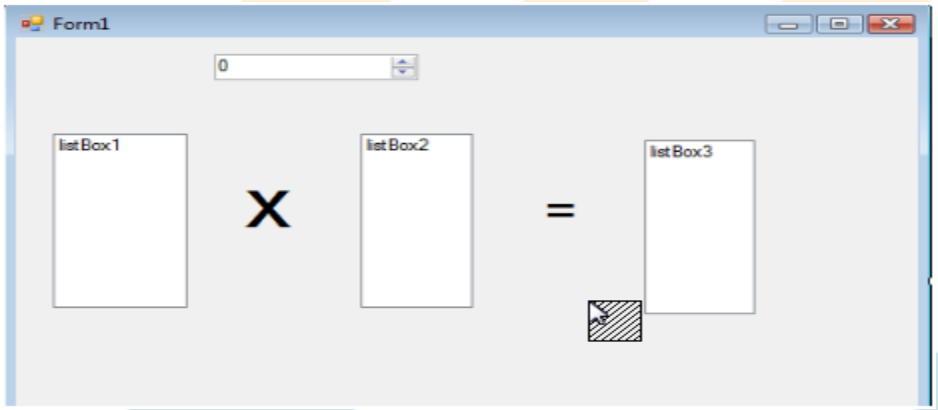


Figura 39:Vista de diseño

Como se puede apreciar este ejercicio tampoco tiene vinculado un Button, por lo tanto el código se realizo en el objeto NumericUpDown, que es el que esta cambiado con un evento (clic). Para hacer esto damos doble clic al ese objeto

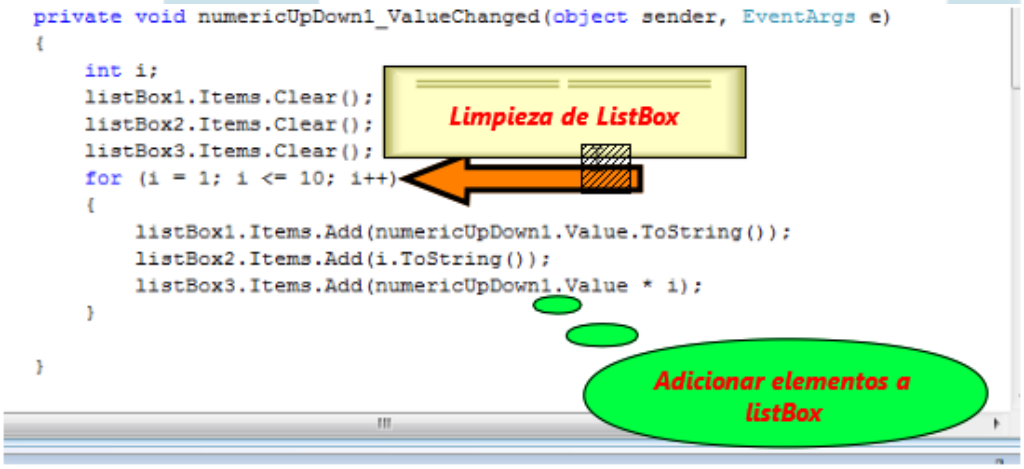
```
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {


        }

    }
```

Ahora se procede a agregar el código y para esto utilizamos el ciclo for tal y como se ve en la siguiente imagen



Ejercicios: realizar la implementación del ejerció anterior utilizando el ciclo while.

2.- crear un nuevo proyecto, adicionar una imagen y mediante la utilización de un ciclo permitir que esta imagen se desplace de izquierda a derecha y al contrario dentro de un formulario.

Lección 15: Trabajo con varios formularios

Al momento de realizar aplicaciones se hace necesario la utilización de varios formularios, a continuación se plantea un ejercicio que permita mediante la creación de un menú llamar tantos formularios como opciones tenga.

Nuevamente para explicar esta lección se planteará un supuesto problemico

Ejercicio: Realizar una aplicación que mediante la construcción de un menú, permita encontrar el área de diferentes figuras (triangulo, cuadrado, circulo), teniendo en cuenta que los resultados solo pueden ser positivos.

Se requiere:

4 formularios, los cuales se agregan buscando la opción agregar un nuevo elemento la opción agregar win forms (imágenes inferiores)

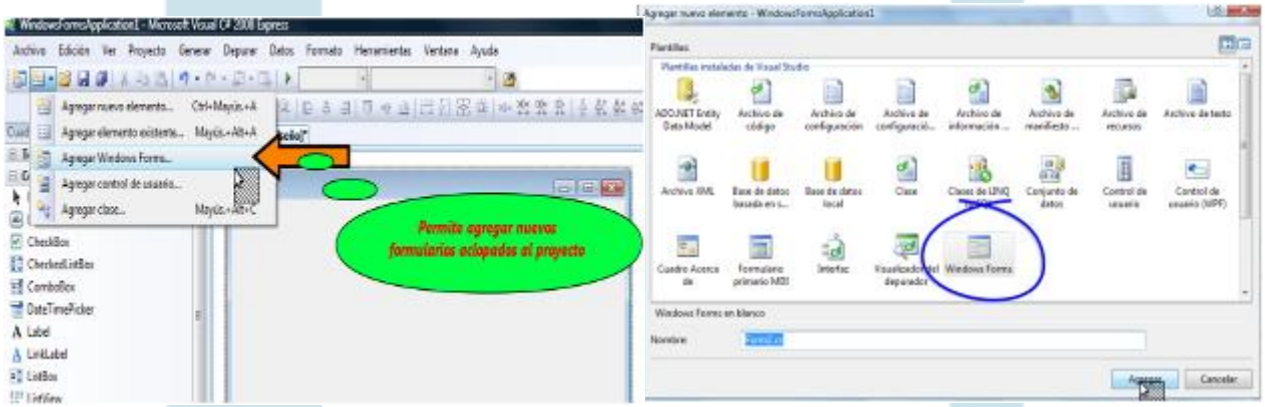


Figura 40: trabajo con varios formularios

Primer formulario

Dos label (imagen izquierda) y un MenuStrip(para la creación de de los menús imagen derecha)

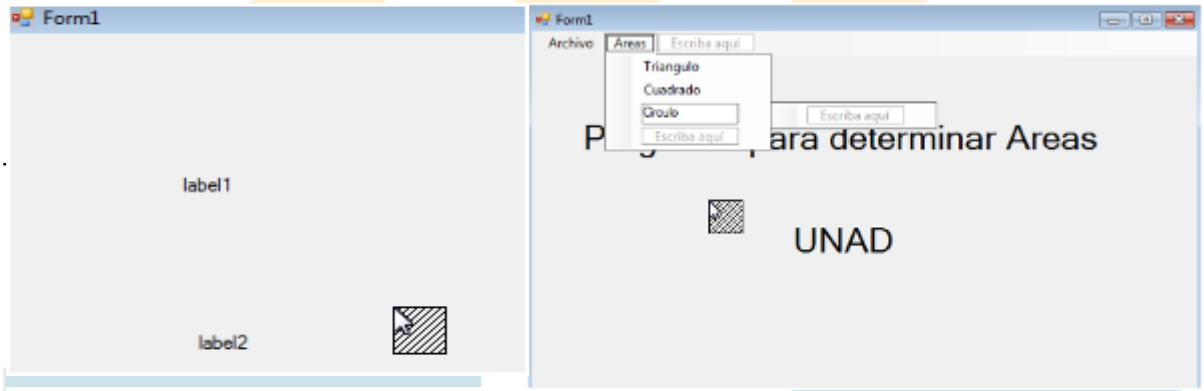


Figura41:Diseñando

En este momento ya deben estar activos los cuatro formularios como se puede ver en la siguiente imagen

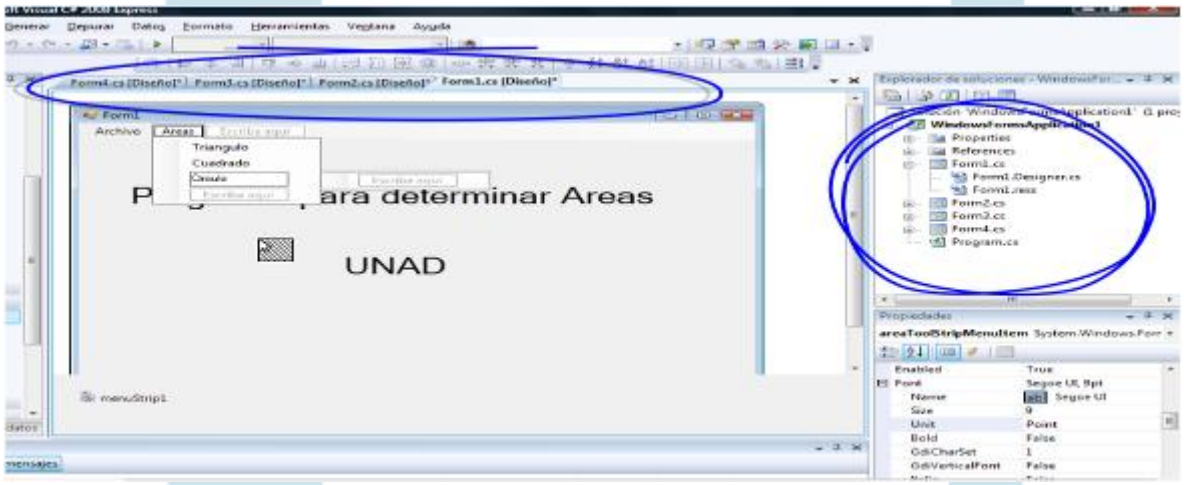


Figura 42: vista 4 formularios

Y es momento de ver el código para llamar a los formularios creados esto se hace dando doble clic en la opción del menú correspondiente, y se agrega un código muy simple

```
private void trianguloToolStripMenuItem_Click(object sender, EventArgs e)
{
    Form2 frm2 = new Form2();
    frm2.Show();

}
```

Con esto es suficiente para llamar el segundo formulario y así con cada uno de los formularios si se ejecuta se puede ver en la siguiente imagen

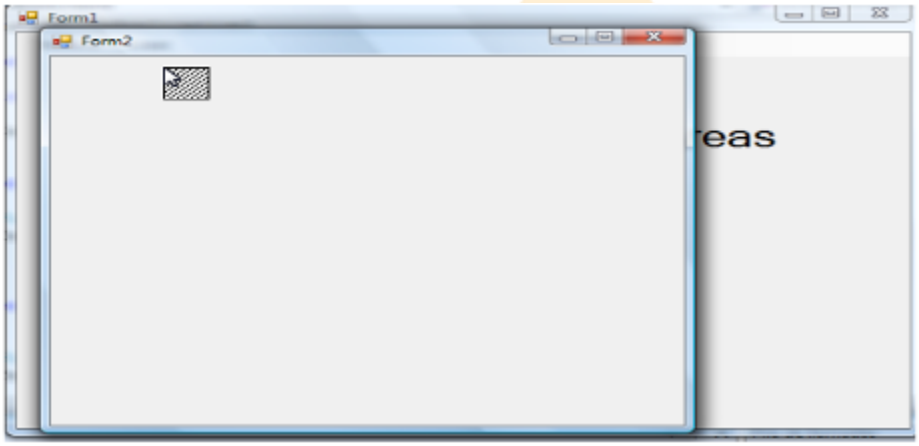


Figura 43:llamado a un formulario

Ahora se debe realizar cada una de las opciones como se ve en n la siguiente imagen, en este caso para el segundo formulario

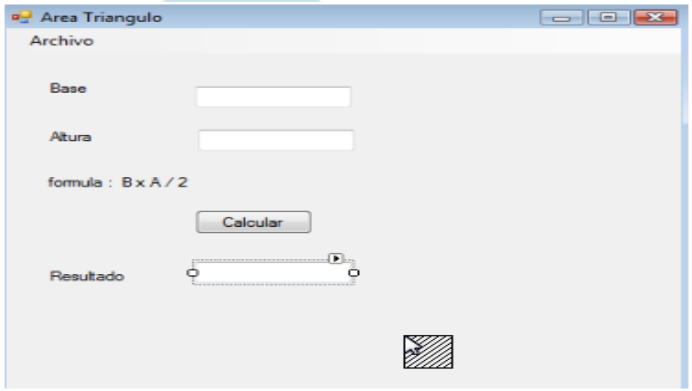


Figura 44:Diseño formulario 2

Con esto solo queda agregar el código necesario para calcular el área del triangulo dando doble clic en el button1 (Calcular)

```
private void button1_Click(object sender, EventArgs e)
        {
            float al,a, b;

            b = float.Parse( textBox1.Text);
            al = float.Parse(textBox2.Text);
            a = (b * al);
            if (a <= 0)
                MessageBox.Show(" cometio una gran falta");
            else
```

```
textBox3.Text = (a.ToString());
```

En el anterior código se puede observar como se realice el cálculo del área del triangulo a demás de eso permite generara un mensaje de error cuando el resultado es negativo, como se había solicitado

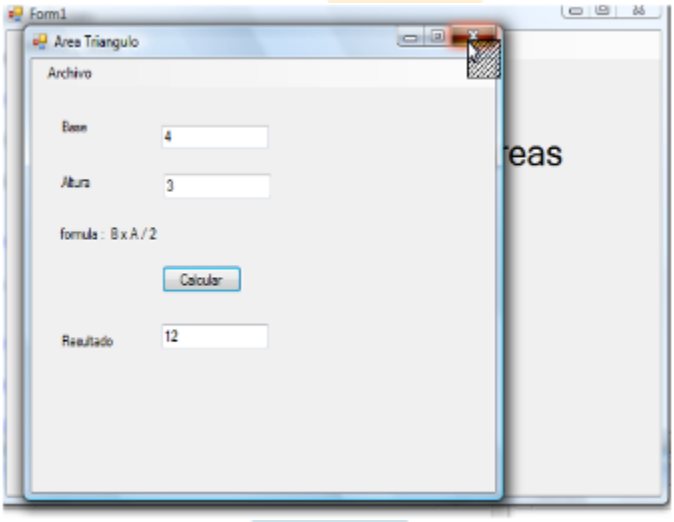


Figura46: Formulario 2 funcional

Figura45: mensaje de error

La construcción de esta aplicación también la puede mirar en http://ivan.lopezortiz.googlepages.com/algoritmos video multiples formularios

INSUASTY R, Luis Delfín, Guia “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 Edicón). Edición Española 1995. Addison-Wesley Iberoamericana

Sitios WEB

David Elías Espinoza Sandoval (2004). Curso Básico de Algoritmia. Consultado en 08,08,2008 en http://www.geocities.com/david_ees/Algoritmia/curso.htm.

Justo Mendez (2004). Las tendencias en los lenguajes de Programación. Consultado en 08,08,2008 en http://www.ilustrados.com/publicaciones/EpZVVEZpyEdFpAKxjH.php.

Alvarez, delia y otros: (2008). historia de la escritura. Consultado en 1/02?2009 en http://centros5.pntic.mec.es/ies.arzobispo.valdes.salas/alumnos/escri/presen.html.

Desconocido (2008). Medio de Comunicación. Consultado en 12/12/2008 en http://es.wikipedia.org/wiki/Medio_de_comunicaci%C3%B3n.

Desconocido (2006). introducción a la informatica. Consultado en 06/12/2007 en http://wwwdi.ujaen.es/~mcdiaz/docencia/cur04_05/fi/teoria/01_Introduccion.pdf.

Desconocido (2006). lengueje de programación . Consultado en 06/12/2007 en http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n#Lenguajes_de_M.C3.A1quina.

Desconocido (2008). lengueje de programación . Consultado en 02/12/2009 en http://es.kioskea.net/contents/langages/langages.php3.

ANEXOS

ANEXO

1. 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:

Lectura

"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 Stroustrup20 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.

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

[20] Consultar más sobre Bjarne Stroustrup : http://www.research.att.com/~bs/homepage.html

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 lo 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.

1.1 Características Del Lenguaje C++

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

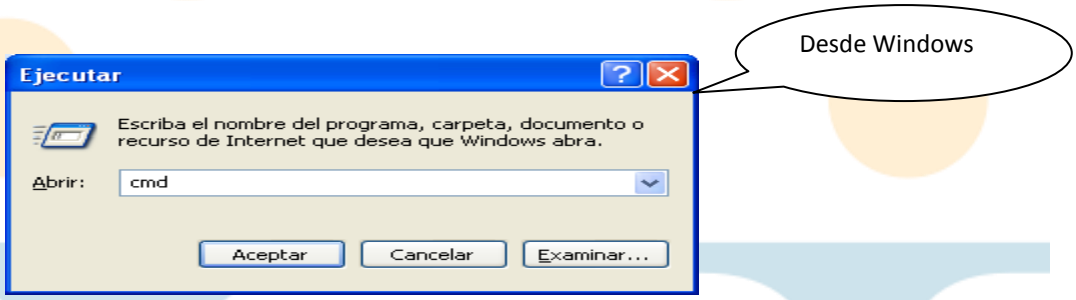
- Tipos de datos básicos
- Estatutos de control
- Arreglos
- Apuntadores
- Estructuras y Uniones
- Funciones definidas por el usuario
- Funciones integradas que permiten el fácil manejo de los recursos del sistema en bajo nivel
- Y además incorpora las siguientes características que permiten la programación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

- Herencia
- Polimorfismo
- Sobrecarga de funciones y operadores" 21

1.2 Ejecución del programa

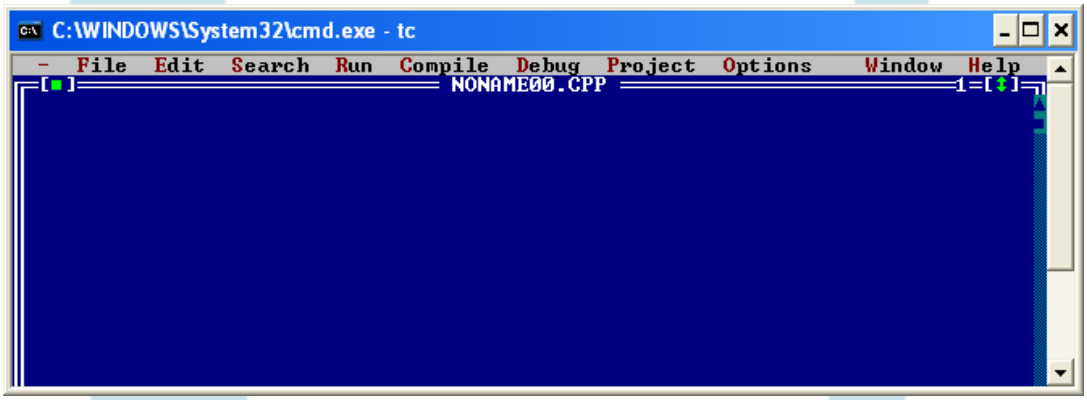
Una vez el programa este instando en su computadora, la ejecución es sencilla:

1.



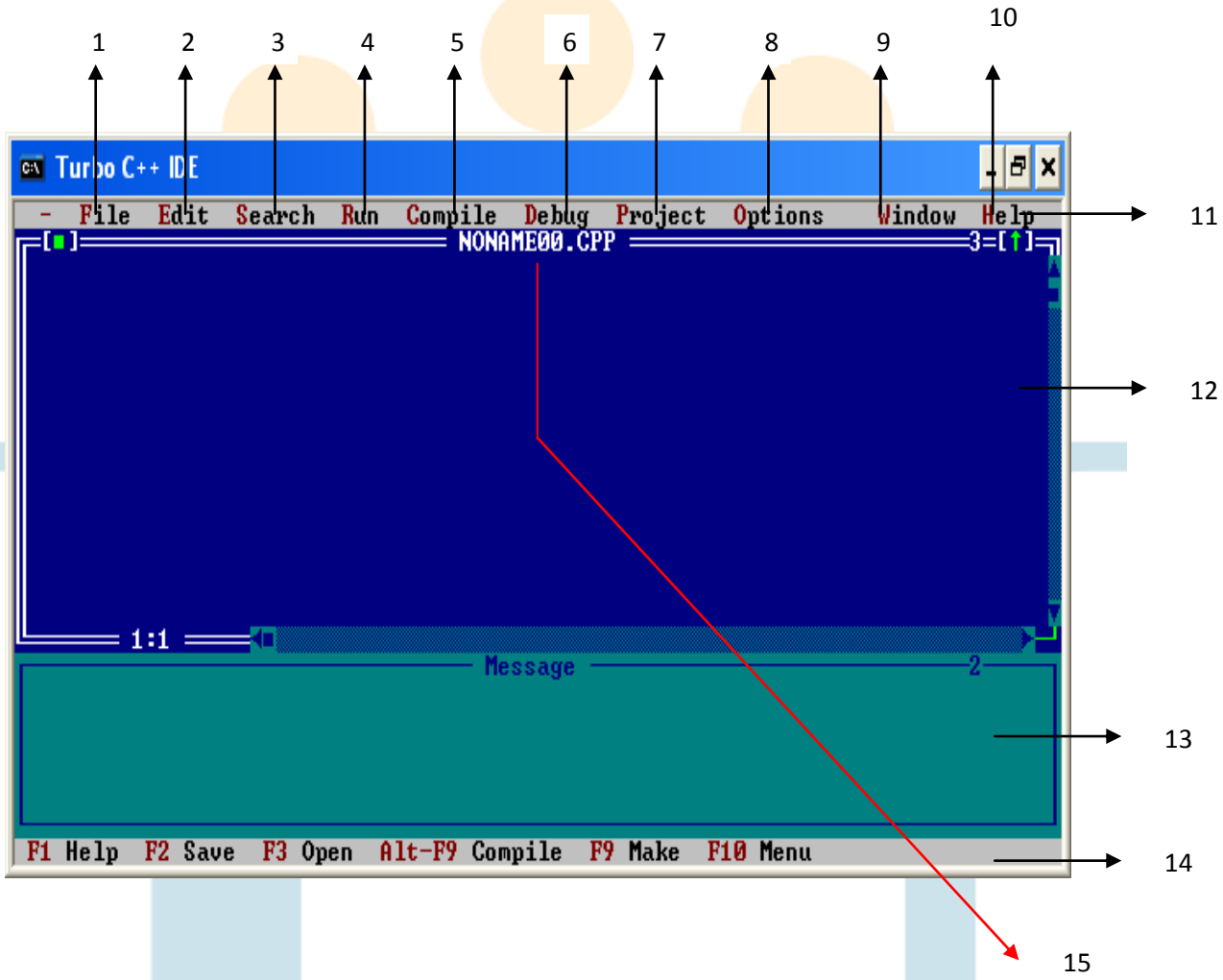
2.

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++**

[21] Articulo completo: http://www.itlp.edu.mx/posgrado/lengprog/c.htm



l 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

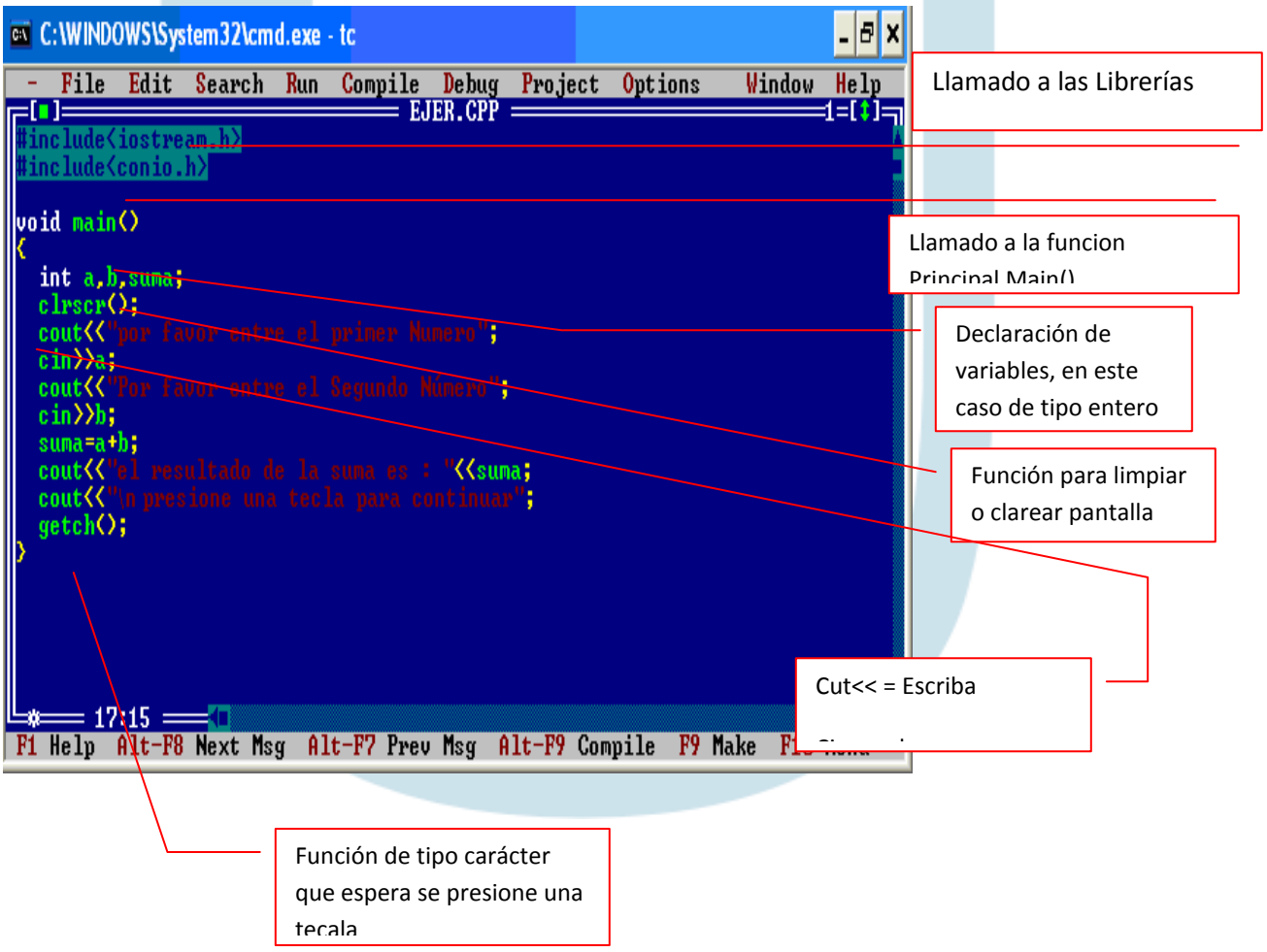
Vamos a realizar un ejercicio practico y sobre el se explicaran cada una de las acciones

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

El análisis de este ejercido y se conoce

Procedemos

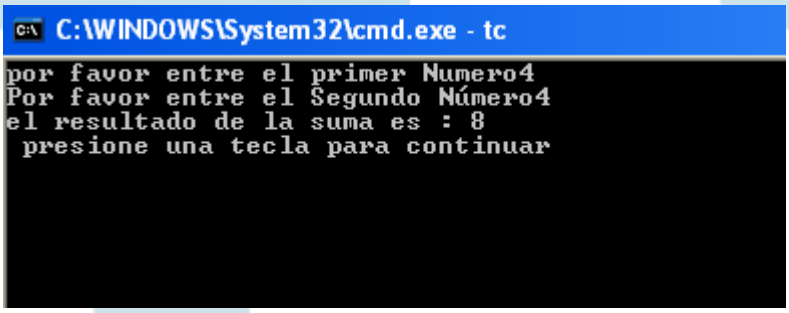
- Se carga el editor del C++, para proceder a ingresar el programa, tomando como referente el algoritmo



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



Explicación:

Como se puede dar 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<<[22] y cin>>[23] 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

[22] <<console output>> Objeto de flujo estándar de salida, que normalmente es por la pantalla de la computadora

[23] cin>>: objeto de flujo de captura de datos

main()[24]; 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	$3.4*(10^{-38})$	$3.4*(10^{38})$
Double	$1.7*(10^{-308})$	$1.7*(10^{308})$
Long double	$1.7*(10^{-308})$	$1.7*(10^{308})$

4.-clrscr()[25], 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

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 línia se ubica en una nueva posición, es decir una nueva línea

Otros caracteres secuénciales[26]

[24] Función principal de todo programa en c++

[25] Clrscr: (clear scream)

[26] Es necesario que se de una mirada a los codigos ASCII

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
\\	Barre 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;
        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.>
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 igualad significa asignación.

Si es del caso determinar un *diferente l*o 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

1.2.3 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 aprograma 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 estudiante 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 del día	**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 => dos lados iguales
  - Escaleno => $A \neq B \neq C$
  - Equilátero => $A = B = C$

- Con relación a sus ángulos un triángulo puede ser:

  - Rectángulo => Un ángulo recto
  - Acutángulo => 3 ángulos agudos
  - Obtusángulo => 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

1.2.4 c++, ciclos

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

1.2.4.1 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;

  gtch();

}
```

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<=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.>
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;
  gtch();
}
```

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.>
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;
  gtch();
}
```

Explicación

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

/******

1.2.4.1.1 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

1.2.4.2 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 cundo queremos realizar filtros[27] y cuando deseamos que se permita el ingreso al ciclo al menos una vez

[27] Filtro: permitir el ingreso de datos dentro de un rango especificado

Ejemplo

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 realizo en la semana referente a ciclos, trabajado con algoritmos.

Con este ejemplo se utilizaran *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.>
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 conecto **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

1.2.4.2.1 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.
- Calcular 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.

1.1.5. Funciones

1.1.5.1 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

1.1.5.2.-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.

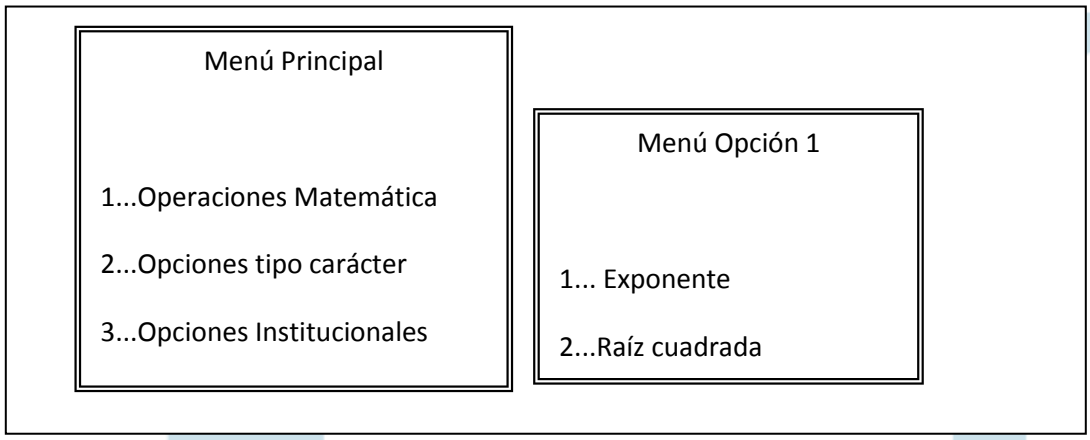
Para hacer una referencia a una función se invoca mediante un nombre y en caso de existir, una lista de parámetros actuales necesarios (argumentos). Los argumentos deben coincidir en cantidad, tipo y orden con los de la función que fue definida. La función devuelve un valor único.

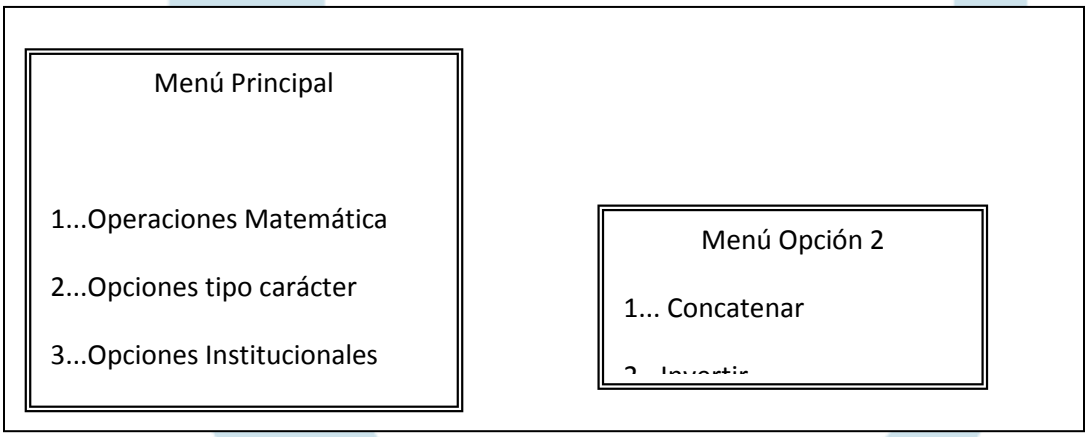
Las funciones a que se hace referencia, se conocen como funciones de usuario puesto que son definidas por él mismo y permiten su uso en forma idéntica a las funciones estándares. Para coordinar e iniciar el procesamiento, se utiliza un módulo principal que es colocado al final del algoritmo.

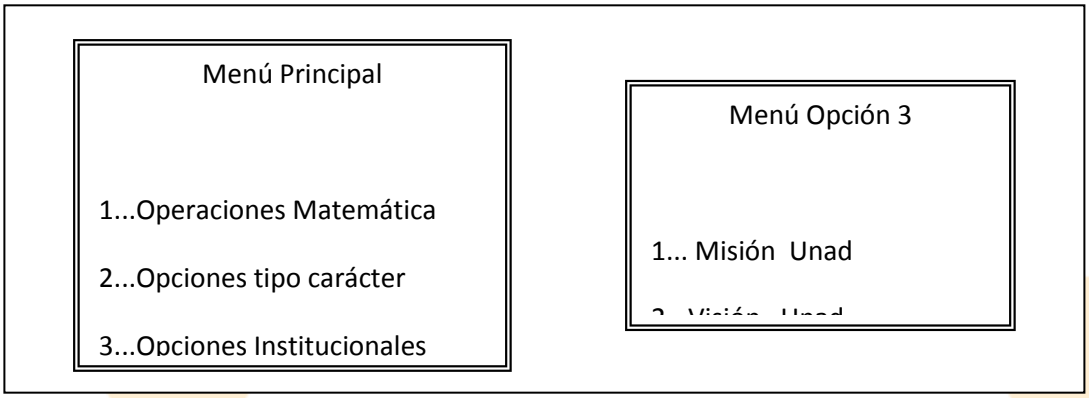
Para entender mejor la construcción de funciones, se propone un ejemplo práctico

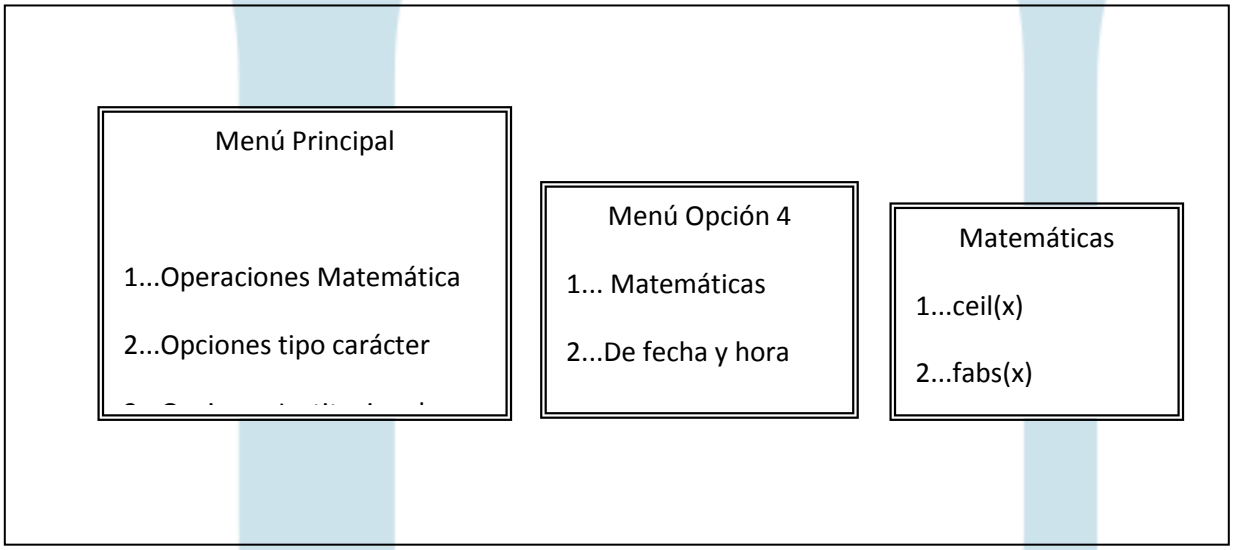
Ejercicio

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







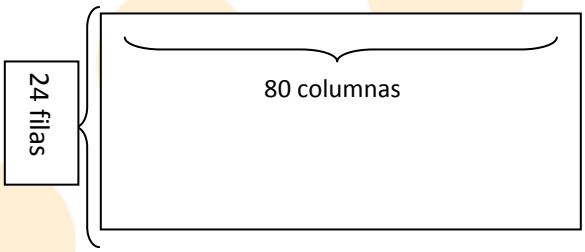


Análisis

Se pude 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 esta contenido dentro de un marco, por consiguiente se puede pensar que una de las primeras funciones a realizar es el cuadro o marco contenedor, par lo cual se debe tener en cuenta que:

Un monitor normal en formato texto esta 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<<"║";
 }

}
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();
}
```

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

Explicación

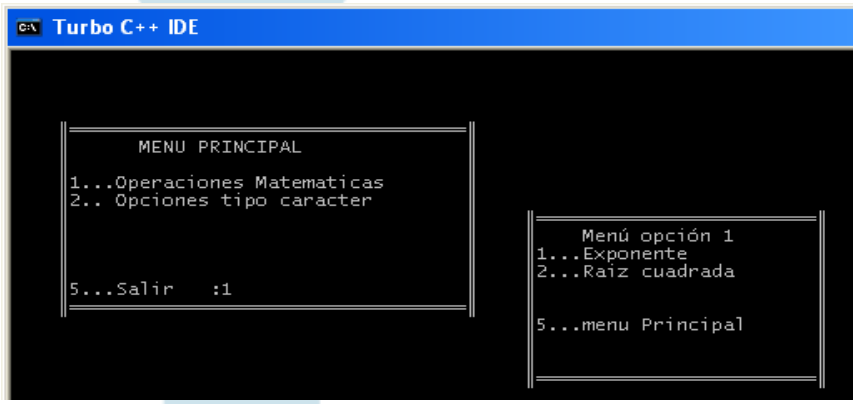
1.-Es importantes que se lean los comentarios que se escriben al frente de cada línea del programa, estos comentarios enastan resaltados con negrita

2.- Con un poco de ingenio se puede mejorar el cierre de la líneas de los cuadros (utilizar códigos ASCII)

1.1.5.3 ejercicios de verificación

1- Complementar el ejercicio

Posible salida



2.-

- Diseñe una función para calcular Xn 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)

ANEXO 1 FORMATO PARA LA AUTOEVALUACIÓN DEL GRUPO COLABORATIVO[28]

<table>
<tr><th></th><th></th><th>SI</th><th>NO</th></tr>
<tr><td>1</td><td colspan="2">Trabajamos siguiendo un plan</td><td></td><td></td></tr>
<tr><td>2</td><td colspan="2">Trabajamos todos juntos</td><td></td><td></td></tr>
<tr><td>3</td><td colspan="2">Intentamos resolver la actividad de diferentes maneras</td><td></td><td></td></tr>
<tr><td>4</td><td colspan="2">Resolvimos la actividad</td><td></td><td></td></tr>
<tr><td>5</td><td colspan="2">Repasamos nuestro trabajo para asegurarnos que todos
Estamos de acuerdo</td><td></td><td></td></tr>
<tr><td rowspan="3">6</td><td colspan="2">Le asignamos responsabilidades a cada miembro</td><td></td><td></td></tr>
<tr><td>Responsabilidad</td><td>Responsable</td><td colspan="2" rowspan="2"></td></tr>
<tr><td></td><td></td></tr>
<tr><td>7</td><td colspan="4">Usamos los siguiente materiales o bibliografía</td></tr>
</table>

[28] Técnicas y estrategias del pensamiento critico pag.166-168

8	Aprendimos:
9	Resolvimos la actividad con la siguiente estrategia:
10	Lo aprendido lo podemos aplicar en los siguientes contexto

ANEXO 2 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. "[29]

- "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"[30]

[29] Técnicas y estrategias del pensamiento critico pag.116

[30] Especialización APRA el desarrollo del aprendizaje autónomo, guía C pag 80