Programación en C
Marco introductorio

Lenguaje de Programación Estructurado C

Operadores, Expresiones y Estructuras

Ciclos

Funciones en C

Estructuras de Datos

Presentación

El Presente documento ha sido elaborado con la finalidad, de proveer a los estudiantes, de un recurso ágil y fácil de comprender por aquellos que inician su estudio en la programación del lenguaje C.

Espero que, este pequeño manual les sirva de mucho y sobre todo, le saquen mucho provecho.

CAPITULO I "MARCO INTRODUCTORIO"

Marco Conceptual

Sé muy bien, que usted, querido lector; está muy ansioso por comenzar a programar, pero considero que es importante; conocer un poco del lenguaje C, tanto de sus orígenes como de sus ventajas, pero no se preocupen, seré breve en esta introducción teórica. Además que es importante conocer o recordar algunos conceptos que, son importantes al momento de programar.

Computadora

Componentes de Una Computadora

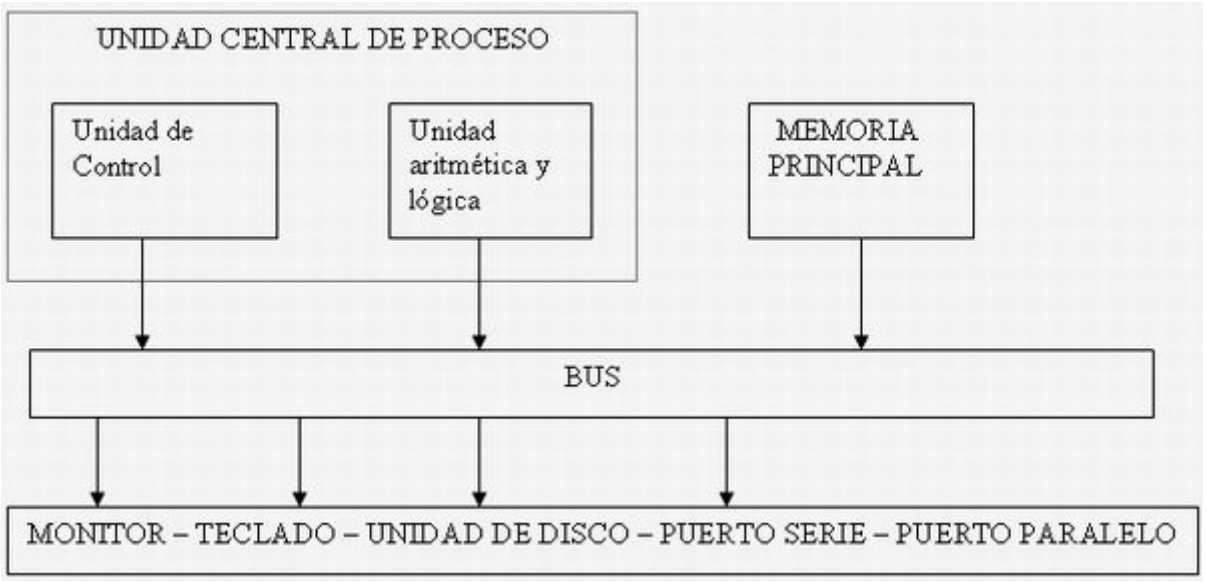
Hemos definido una, computadora como una máquina que recibe datos y ordenes, que al ejecutarlas produce cierta información; dicha información se presenta en un "idioma" codificado; por que ha de saberse que las computadoras no entienden nuestro idioma, o cualquier otro en el mundo. Dicho "idioma" está compuesto únicamente por dos elementos los ceros y los unos. Mejor conocido como código Binario, con el cual se representan los datos, que arroja la computadora.

En una forma más general, y creo que más sencilla, una computadora se comprende por dos grandes grupo: El Hardware y el Software.

Hardware

El Hardware de un computador es un conjunto de elementos físicos, que la componen.

Veámoslo gráficamente:



Podríamos entrar en detalle de cada una de las partes descritas anteriormente, pero ese, no es el objetivo de estas insignificantes páginas; sino que esto es una mera introducción teórica, por ello sólo daré una breve explicación.

En la Unidad Central de Proceso (o CPU, por sus siglas en ingles –Central Proccessing Unit-) se contiene la Unidad de Control, que su función es organizar y clasificar las instrucciones recibidas; mientras que la Unidad Aritmética y Lógica, Se encarga de ejecutar dichas instrucciones. Los Buses, son los mecanismos de interconexión en el CPU.

La memoria Principal, Es el lugar donde se cargan todas las instrucciones, programas, etc que se están ejecutando.

Software

Debemos entender el software como la parte lógica de la computadora... ¿un poco difícil de comprender, verdad?; es decir, que el software, es lo que dota a los componentes físicos de poder realizar tareas determinadas. Ejemplo, para poder utilizar una computadora, esta debe tener instalado un sistemas operativo. Para poder imprimir algún trabajo, aparte de poseer un impresor, en la computadora, debo tener un software que me permita imprimir dicha acción (generalmente las impresoras traen un cd, son su respectivo software de instalación).

Aburrido?...

Es necesario, que empiece con tanta palabrería; por que es necesario, para el lector tener en claro estos conceptos.

Ahora vamos a hablar de algo un poco más interesante, como lo es el lenguaje de programación

Un lenguaje de Programación Es un conjuntos de palabras, reglas, con las cuales se le indica a la computadora las funciones que debe realizar. Un lenguaje de programación puede ser:

Lenguajes Máquinas: se trata de lenguaje cuyas instrucciones son directamente comprendidas por el ordenador o computador en el que se ejecuta el programa.

Lenguaje de Bajo Nivel: este tipo de lenguajes, al igual que sucede con los lenguajes máquinas, existe una gran dependencia con el equipo en el que se va a ejecutar. No obstante son algo más fáciles de escribir, quedando ubicados por tanto, según su grado de complejidad; en un nivel intermedio entre el lenguaje máquina y el de alto nivel.

Programación en C

Lenguaje de Alto Nivel: Disponen de una sintaxis en lenguaje más natural, y un amplio conjunto de funciones internas, que ayudan al programador en distintas situaciones, así como un número determinado de utilidades y asistentes que ahorran tiempo y trabajo al programador. Dentro de estos lenguajes tenemos: Visual Foxpro, Visual Basic. NET.

Programa:

Es un conjunto de instrucciones que se le dan a la computadora, para que ésta realice una determinada tarea.

Lenguaje C

El lenguaje C, fue diseñado por Dennies Ritchie en 1970, en los laboratorios Bell de Estados Unidos.

Este lenguaje presenta varias características, entre las cuales están:

Lenguaje de programación de propósitos generales

Permite la Programación Estructurada

Abundancia de Operadores y Tipos de Datos

No está asociado a ningún sistema operativo ni a ninguna máquina

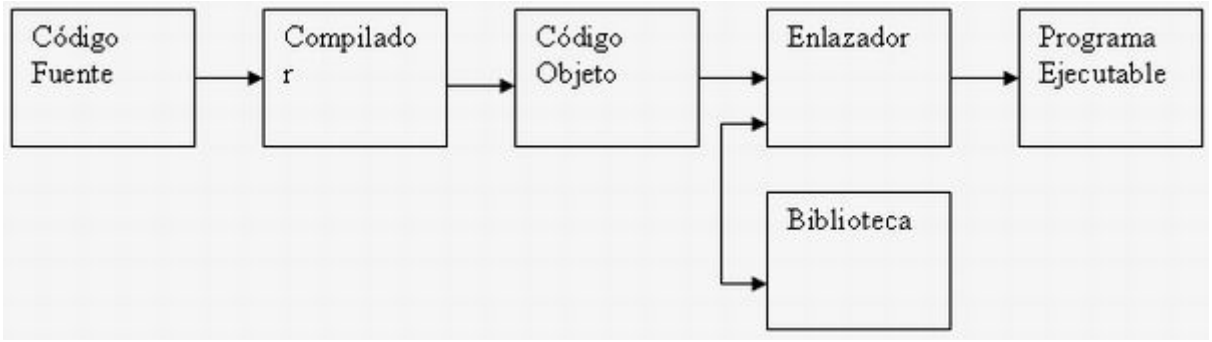
Popular y Eficaz

Permite el desarrollo de Sistemas Operativos y programas de aplicación

Portabilidad

Existen las librerías en las bibliotecas

tiene sólo 32 palabras reservadas



*bibliotecas: es el archivo que contiene código objeto de una colección de rutinas o funciones que realizan tareas determinadas y se pueden utilizar en los programas.

*Enlazador: Programa que convierte las funciones de la biblioteca estándar de C, con el código que ha traducido el compilador .

Estructura de Un programa en C

Ya estamos apunto de entrar a lo más interesante, a la programación en sí; pero es necesario, primero; mencionar algunos de los errores típicos al programar, para que el lector sepa cómo identificarlos y así los pueda corregir.

ERROR DE SINTAXIS: Estos errores son producidos, cuando se hace mal uso de las reglas del lenguaje de programación, y se violan las normas de sintaxis, de ese lenguaje (en nuestro caso C); estos errores son fáciles de detectar por que generalmente es el compilador, que los identifica (Y hasta muestra la línea donde se encuentra dicho error, pero eso depende de la versión del compilador que estemos usando). En este curso he usado Turbo C, en su versión 2 y 3.

ERRORES DE EJECUCIÓN: Estos errores se producen, cuando le indicamos a la computadora, realizar una determinada acción, y esta la comprende, pero no puede ejecutarla. Por ejemplo, indicarle a la computadora una división entre cero, sumar dos variables a las cuales no se les ha signado valor alguno, etc.

ERRORES DE LÓGICA: Muchas veces, cuando estamos programando, el compilador no nos indica errores de sintaxis, ni de lógica; pero el resultado de nuestro programa, esta fuera del rango esperado, esto es producto de un error de lógica en el código de nuestro programa. Este tipo de errores son muy difíciles de identificar y por supuesto de corregir, ya que generalmente hay que revisar línea por línea de nuestro programa. Ejemplo: El sueldo negativo de un empleado, etc.

La estructura de un programa en C, consta de algunas partes esenciales: las cuales son uno o más módulos llamadas funciones, siendo main() la primera función que es llamada cuando empieza la ejecución del programa .

Cada función debe contener:

>Directivas de pre-procesador (instrucciones que se le dan al compilador

#include antes de compilar)

#define

ejemplo:

#include <stdio.h>

Lo que se le esta indicando, es que de las librerías, "Incluya" en nuestro programa la directiva stdio.h, la cual contiene las funciones de entrada y salida de datos (standar input output, en inglés). Si necesitamos las funciones matemáticas, debemos especificarlo con la declaratoria:

#include <math.h>

Si necesitamos las funciones de cadenas:

#inlcude <stlib.h>

Es necesario aclarar que esto se hace al inicio del programa, y las declaratorias deben llevar el símbolo de numeral (#) seguido de la sentencia "include", y entre signos de mayor y menor que (<>) el nombre de la directiva.

>Declaraciones Globales

pueden ser:

*Prototipos de Funciones: También llamadas declaraciones de funciones, lo cual se tratará más adelante

*Declaraciones de Variables

cabe destacar, que esto se hace seguido de los #include y los #define.

>Función Principal main()

Esta es la función principal de nuestro programa, su cuerpo, por ello NUNCA debe faltar, ya que en ella van contenidas todas las instrucciones de nuestro programa.

```
main()

{

declaraciones locales /*Comentarios */

sentencias

}
```

la función main() va al inicio, luego abrimos llaves y dentro de ellas van las declaraciones de variables, las sentencias de lectura, cálculos, asignaciones e impresiones, y con la última llave (}), le indicamos el final del programa.

Ejemplo 1.1

Programa que a partir del radio, calcula el área de un circulo

```
#include <stdio.h>

#include <conio.h>

main()

{

float radio, area;

printf("Radio=\n");

scanf("%f", &radio);

area=3.14159*radio*radio;

printf("El Area es %f\n\n", area);

getch();

return 0;

}
```

Explicación:

Le indicamos al compilador, que usaremos las bibliotecas <stdio.h> y <conio.h>, ¿por qué <conio.h>?, por que esta biblioteca, contiene las funciones getche(), getch(), etc, y de una de ellas hacemos uso en este pequeño ejemplo.

Luego, le indicamos a nuestro programa el incio de nuestro programa (función main()).

Declaramos, como valores reales, las variables radio y area (de esto se hablará más adelante). Luego, con la instrucción printf(), mostramos en pantalla el mensaje (Radio=) y scanf se encarga de leer el valor digitado por el usuario. Posteriormente area, es igual al la multiplicación de pi (3.14159), el radio al cuadrado. Se muestra en pantalla ese resultado, luego el programa espera que se presiones cualquier tecla (getch()) y no retorna ningún valor (return 0).

Programación en C
Ejercicios

Defina los siguientes conceptos:

Programa:_____

CPU:_____

Software:_____

Memoria Principal:_____

Lenguaje de
Programación:_____

Indique que tipo de error (Error de sintaxis, error de ejecución o error lógico), en cada uno de los siguientes enunciados

Utilizar una variable antes de asignarle un valor:_____

asignarle un valor real a una variable declarada como entero:_____

al cometer este error, los resultados arrojados por el programa no son los que se
esperaban:_____

Un programa no puede ser ejecutado por el computador, mientras tenga este tipo de
errores:_____

estos errores no son detectados por el compilador, ni tampoco son errores de
ejecución:_____

Mencione y Explique, la estructura general de un programa en C:

Capitulo II: " Lenguaje de Programación Estructurado C"

¿Por qué programación estructurada?

Si el lector recuerda, en el capítulo anterior, se hablaba de las características del lenguaje C, y en una de ellas se decía que, el Lenguaje de Programación C, permite la programación estructurada. Esto implica que, haremos uso de una técnica llamada Lógica Estructurada, y esto no es más ni menos que una de las técnicas básicas y fundamentales de la programación estructurada, su objetivo es diseñar soluciones "correctas" y confiables a los problemas, ignorando al principio consideraciones de eficiencia como la minimización del uso de memoria y el tiempo de su respuesta.

Lo que significa que, haremos uso de esa técnica para crear programas correctos; esta es una técnica que ayuda al programador (un tanto a la fuerza), a ser ordenado, al momento de programar.

Los frutos de ésta técnica se reflejan cuando, queremos darle mantenimiento al programa, es más fácil hacerlo ya que hemos programado de una manera lógica y ordenada. Al igual que al momento de corregir errores de sintaxis y lógica, esta técnica nos facilita el trabajo.

Ahora iniciemos, de una vez por todas, lo que el lector está esperando:

como su nombre lo indica, estos son los nombres, con los que identificamos las variables, constantes, funciones, vectores, etc, de nuestro programa. Para ello debemos tener presente algunas reglas:

>pueden tener de 1 hasta un máximo de 31 caracteres

>Debe de iniciar con una letra o subrayado

Ejemplo:

(Correctos)

c2

_c2

(Incorrectos)

2c

2 c

>No es lo mismo una minúscula que una mayúscula, ya que c distingue de entre ellas. Ejemplo: BETA ¹ Beta ¹ beta ¹ BeTa

>No son válidos los identificadores de palabras reservadas. En un inicio hablamos que c posee 32 palabras reservadas, entre ellas están:

float char while

int else return

Estas palabras no pueden ser utilizadas para identificar variables, constantes, funciones etc

identificadores:

En todo programa que estemos diseñando en C (o en cualquier otro lenguaje de programación); es necesario insertar ciertos comentarios en el código, para que en posteriores modificaciones y cuando se realice el mantenimiento, podamos recordar cosas importantes ya que, en los comentarios, podemos incluir aspectos importantes del programas, explicaciones del funcionamiento de las sentencias, etc.

El formato de los comentarios en C, es el siguiente:

/*este es un comentario en C */

/*Podemos colocar mucha información importante

de nuestro Programa */

Comentarios

Permite que, el pre-procesador, incluya funciones proporcionadas por el fabricante, a nuestro programa. Ejemplo:

#include <stdio.h> /* le decimos al compilador que incluya la librería

Programación en C
stdio.h */

La Directiva #include

permite definir constantes simbólicas. Pero hasta ahora ha sido poco lo que hemos hablado acerca de las constantes, es por ello que en aprovechando, este especio; dedicaré unas cuantas líneas para aclarar ello.

Las variables pueden cambiar de valor, durante la ejecución del programa, por eso es que se llaman variables. Y las constantes como su nombre lo indica, son valores que permanecen constantes durante toda la ejecución del programa, un ejemplo de ello, es el valor de p (pi) que equivale a 3.14159....

En C existen diferentes tipos de variables, entre ellas tenemos:

1. Constates Numéricas:

Son valores numéricos, enteros o de reales (de punto flotante). Se permiten también constantes octales y hexadecimales.

2. Constantes Simbólicas:

las constantes simbólicas tiene un nombre (identificador), y en esto se parecen las variables. Sin embargo, no pueden cambiar de valor a lo largo de la ejecución del programa. En C, se pueden definir mediante el preprocesador.

(Tomado del Manual "Aprenda Lenguaje ANSI C como si estuviera en Primero" Escuela superior de Ingenieros Industriales. Universidad de Navarra. Febrero de 1998).

Ejemplo:

#define N 100

#define PI 3.1416

#define B 45

Esta directiva (#define) va, inmediatamente después de los #include. Se escribe la directiva, se deja un espacio y se escribe el identificador de la constante, otro espacio y su valor.

la directiva #define

/ ! % ^ & * () - + {} [] \ ; : <> ¿ .

Signos de Puntuación y de Separación

Al momento de programar en C, esta es una regla de oro, y la causa por la cual nuestro programa puede darnos muchos errores de sintaxis, cuando se omite, al final de cada sentencia un punto y coma (;). Ya que con ello le indicamos al compilador que ha finalizado una sentencia.

NOTA: el lector no debe confundirse, las directivas: #include, #define. Main(), no llevan punto y coma, por que no son sentencias.

Recordemos el ejemplo 1.1, y vea que al final de cada sentencia lleva su correspondiente punto y coma:

#include <stdio.h>

```
#include <conio.h>

main()

{

float radio, area;

printf("Radio=\n");

scanf("%f", &radio);

area=3.14159*radio*radio;

printf("El Area es %f\n\n", area);

getch();

return 0;

}
```

Todas las Instrucciones o sentencias del programa terminan con un punto y coma (;)

Esta consideración toma mayor auge, cuando veamos las instrucciones anidadas en condiciones, ciclos, etc.

Ejemplo:

```
{

...

printf("Hola\n\b");

...

}
```

Todo Bloque de Instrucciones debe ir entre llaves

En una línea se pueden escribir más de una instrucción separada por un punto y coma

Esto es posibles, por que con el punto y coma, le estamos indicando al compilador el fin de una sentencia o instrucción.

Ejemplo:

```
b = c + d; d = 2*k;
```

Tipos de Datos en C

Un tipo de dato, se define como un conjunto de valores que puede tener una variables, junto con ciertas operaciones que se pueden realizar con ellas.

*TIPOS DE DATOS PREDEFINIDOS

TABLA CON LOS TIPOS DE DATOS PREDEFINIDOS EN C			
>ENTEROS: numeros completos y sus negativos			
Palabra reservada:	Ejemplo	Tamaño (byte)	Rango de valores
int	-850	2	-32767 a 32767
VARIANTES DE ENTEROS			
short int	-10	1	-128 a 127
unsigned int	45689	2	0 a 65535
long int	588458	4	-2147483648 a 2147483647
unsigned long	20000	4	0 a 4294967295
>REALES: números con decimales o punto flotante			
Palabra reservada:	Ejemplo	Tamaño (byte)	Rango de valores
float	85	4	3.4x10-38 a 3.4x1038
VARIANTES DE LOS REALES			
double	0.0058	8	1.7x10-308 a 1.7x10308
long double	1.00E-07	10	3.4x10-4932 a 1.1x104932
>CARÁCTER: letras, digitos, símbolos, signos de puntuación.			
Palabra reservada:	Ejemplo	Tamaño (byte)	Rango de valores
char	'O'	1	0 255

TABLA 2.1

NOTA: El tipo de dato string y bolean NO existen en C. Sin embargo más adelante veremos una forma de cómo hacer uso de las cadenas de texto.

Declaración de Variables

Una Variable, como su nombre lo indica, es capaz de almacenar diferentes valores durante la ejecución del programa, su valor varía. Es un lugar en la memoria el cual, posee un nombre (identificador), y un valor asociado.

La declaración de variables en C, se hace en minúsculas.

Formato:

Tipo_de_dato nombre_de_la_variable;

Ejemplos:

*Declare una variable de tipo entero y otra de tipo real, una con el nombre de "x" y otra con el identificador "y":

int x;

float y;

*Declare una variable de tipo entero llamada moon, e inicialícela con un valor de 20

int x = 20;

*Declare una variable de tipo real, llamada Pi, e inicialícela con una valor de 3.1415

float pi=3.1415;

*Declare una variable de tipo caracrter y asígnele el valor de "M"

char car = ’M’;

*Declare una variable llamada nombre, que contenga su nombre:

char nombre[7]="Manuel";

Explicación:

En el apartado anterior, se explicó, que C, no tiene el tipo de dato llamado string, o mejor conocido como cadenas de texto, pero nosotros podemos hacer uso de ellas, por medio de un arreglo, (de lo cual hablaremos con más detalle, posteriormente); pero para declarar este tipo de datos colocamos el tipo de datos, es decir la palabra reservada char luego el nombre, e inmediatamente abrimos, entre corchetes, va el número de letras, que contendrá dicha variable. Es muy importante que al momento de declarar el tamaño, sea un número mayor, al verdadero número de letras; por ejemplo, la palabra "Manuel", solo tiene 6 letras, pero debemos declararlo para 7 letras ¿Por qué?.

Veámoslo gráficamente, en la memoria, C crea un variable llammada nombre y esta posee la palabra Manuel, así:

M	a	n	u	e	l	\0
0	1	2	3	4	5	6

en realidad, hay 7 espacios, pero la cuanta llega hasta 6, por que c, toma la primera posición como la posición cero, y para indicar el final de la cadena lo hace con un espacio en blanco.

Declaración de Constantes

Las constantes, como su nombre lo indica, son valores que se mantiene invariables durante la ejecución del programa.

Su formato es el siguiente:

const tipo_de_dato nombre= valor;

donde const, es una palabra reservada, para indicarle al compilador que se esta declarando una constante.

Ejemplo:

const int dia=7;

const float pi=3.14159;

const char caracter= ‘m’;

const char fecha[]="25 de diciembre";

Caso Especial Constantes Simbólicas

Las constantes simbólicas, se declaran mediante la directiva #define, como se explicó anteriormente. Funcionan de la siguiente manera, cuando C, encuentra el símbolo que representa a la constante, lo sustituye por su respectivo valor.

Ejemplo:

#define N 150

#define PI 3.1416

#define P 50

NOTA: El lector debe comprender algunas diferencias fundamentales entre la declaratoria const y #define; la primera, va dentro del programa, es decir, dentro de la función main() o alguna función definida por el usuario, mientras que #define va en el encabezado, después de los #include, por eso estas no llevan al final el punto y coma (;).

Entrada y Salida Por Consola

Entrada y Salida por consola: se refiere a las operaciones que se producen en el teclado y en la pantalla de la computadora. En C no hay palabras claves para realizar las acciones de Entrada/Salida, estas se hacen mediante el uso de las funciones de la biblioteca estándar (stadio.h).

Para utilizar las funciones de E / S debemos incluir en el programa el archivo de cabecera stdio.h, mediante la declaratoria:

#include <stdio.h>

Las Funciones de E / S más simples son getchar() que lee un carácter del teclado, espera un retorno de carro (¿), es decir un enter y el eco aparece. Es decir la tecla presionada.

*putchar(): Imprime un carácter en la pantalla, en la posición actual del cursor.

Algunas variaciones:

*getche(): Aparece el Eco

*getch(): No aparece el eco

estas instrucciones se encuentran en la biblioteca conio.h

Veamos un ejemplo:

Programa que espera que se presiona una tecla, la muestra en pantalla, y además muestra el carácter siguiente:

Ejemplo 2.1:

```
#include <stdio.h>

#include <conio.h>

main()

{

char car;

clrscr(); /*Se encarga de borrar la pantalla por eso se llama claer screen*/

car=getchar();

putchar(car+1);

getch();

return 0;

}
```

Ejemplo 2.2:

```
#include <stdio.h>

#include <conio.h>

main()

{

char x; /*Declaramos x como caracter*/

printf("Para Finalizar Presione cualquier Tecla:");

x= getchar();/*Captura y muestra el caracter presionado*/

getch();/*Espera a que se presione cualquier otra tecla para finalizar*/
```

return 0;

}

Entrada / Salida de Cadenas

Una Cadena, es una frase, compuesta por varias palabras. En C, podemos hacer uso de las cadenas, mediante, la sentencia:

*gets(): Lee una cadena de carácter introducido por el teclado. Se puede introducir caracteres hasta que se de un retorno de carro, (enter); el cual no es parte de la cadena; en su lugar se coloca un terminador nulo \0.

*puts(): Imprime en pantalla, el argumento guardado en la variable que se manda a impresión.

Ejemplo 2.3

Diseñe un programa en C, que lea su nombre; lo salude y mande a impresión su nombre, usando gets e y puts

```
#include <stdio.h>

#include <conio.h>

main()

{

char nombre[40];

puts("digite su nombre:");

gets(nombre);

puts("BIENVENIDO:");

puts(nombre);

getch();

return 0;

}
```

NOTA: No haré mucho énfasis en estas instrucciones, ya que más adelante, veremos las instrucciones scanf() y printf(), que son mucho más completas.

Entrada / Salida Por Consola con Formato

Las funciones gets, puts, getch, etc; son utilizadas, en una forma un poco rudimentaria, sin embargo; C posee otra serie de funciones, que son más completas, las cuales nos permiten leer e imprimir (en pantalla), datos con un formato determinado, el cual ha sido definido por el programador.

Salida Hacia Pantalla [printf()]

Se utiliza para imprimir en pantalla cadenas de texto solas, o mandar a pantalla el valor de alguna variable, o constante, o una combinación de las anteriores. Su formato es el siguiente:

Printf("cadena de control", nombre_de_variables);

En donde:

Cadena de control: contiene códigos de formato que se asocian con los tipos de datos contenidos en las variables.

Código	Formato
%d	Un entero
%i	Un entero
%c	Una caracter
%s	Una cadena
%f	Un real
%ld	Entero largo
%u	Decimal sin signo
%lf	Doble posición
%h	Entero corto
%o	Octal
%x	Hexadecimal
%e	Notación Científica
%p	Puntero
%%	Imprime Porcentaje

TABLA 2.2

Ejemplo:

Int suma=10;

Printf("La suma es %d", suma);

Explicación:

Declaramos primero la variable como entero, con un valor de 10, luego la función printf, el mensaje va entre comillas dobles, luego en el lugar que queremos que aparezca el valor, colocamos el formato de la variable, cerramos comillas, luego una coma y el nombre de la variable. Es importante recalcar, que en la posición que coloquemos el formato es donde aparecerá el valor de la variable en este caso, 10.

Ejemplo:

Char nombre[7]="Manuel";

printf("%s es en creador de este manual", nombre);

NOTA: el número de argumentos que tendrá la función printf() es indefinido, por lo que se puede transmitir cuantos datos sean necesarios.

Ejemplo:

Int x=12, y=15;

char z='D';

float v=10.2563;

printf("Estos son números %d %d %f; y esta es una letra %c", x,y,v,z);

También podemos hacer algunos arreglos, al formato de salida, por ejemplo, si deseamos imprimir un número real justificado a la izquierda podemos colocar:

printf("%-f", z);

para justificar colocarle signo: %+f

%20f >> Longitud numérica del campo

%.2f >>Imprime el valor con sólo dos decimales

Secuencias de Escapes

Indica que debe ejecutar algo extraordinario.

Carácter de Escape	Explicación
\n	Simula un Enter. Se utiliza para dejar una línea de por medio
\t	Tabulador horizontal. Mueve el cursor al próximo tabulador
\v	Tabulador vertical.
\a	Hace sonar la alarma del sistema
\\	Imprime un carácter de diagonal invertida
\?	Imprime el carácter del signo de interrogación
\"	Imprime una doble comilla

TABLA 2.3

Ejemplos:

Programación en C

1) printf("Manuel \n Antonio \n Ortez\n\n);

2) int x=15;

printf("El Valor de la variable es %d\n\n", x);

3) float x=8.5689, pi=3.1416;

printf("El valor de x es %.2f\t\n",x);

printf("\t Y el valor de pi es %.2f\n\n", pi);

Entrada Desde Teclado

Se realiza mediante la función scanf(), su formato es:

scanf("Cadena de control", Dirección y nombre de la variable);

Ejemplo 2.4

Diseñe un programa que guarde y muestre la nota del examen final de 3 alumnos

```
#include <stdio.h>

#include <conio.h>

main()

{

float n1, n2, n3;

char nom1[10], nom2[10], nom3[10];

printf("Introduzca el Nombre del Primer alumno:\n");

scanf("%s", nom1);

printf("Introduzca la nota de este alumno:\n");

scanf("%f", &n1);

printf("Digite el nombre del segundo alumno:\n");

scanf("%s", nom2);

printf("Su nota es:\n");

scanf("%f", &n2);

printf("Finalmente el ultimo alumno es:\n");

scanf("%s", nom3);

printf("Y su nota es:\n");
```

```
scanf("%f", &n3);

getch();

return 0;

}
```

Explicación:

Primero, iniciamos con las directivas del preprocesador:

```
#include <stdio.h>

#include <conio.h>
```

Con la cual le indicamos al compilador, que de su librería añada a nuestro programa las funciones estándar de entrada y salida; así como las entradas y salidas por consola (stadio.h y conio.h, respectivamente).

Luego declaramos la variables, que contendrán las notas como reales (o de punto flotante:

```
float n1, n2, n3;
```

Ya que, las notas pueden ser deciamales, por ejemplo 9.6, 8.5; etc.

Luego declaramos las variables, que contendrán las notas, caba aclarar que al momento de las declaraciones las podemos hacer en el orden que deseemos, pueden ser primeros los tipo char y luego los float, o viceversa, pero teniendo el cuidado que las variables que contendrán las nombres lleven la longitud máxima entre corchetes, para nuestro caso, 10. ([10]).

Posteriormente, mostramos en pantalla, un mensaje con el cual le indicamos al usuario que introduzca los datos respectivos:

```
printf("Introduzca el Nombre del Primer alumno:\n");
```

A continuación, va la función scanf, primero y entre comillas el tipo de dato que va a leer:

```
scanf("%s", nom1);
```

como puede notarse, va a leer la cadena de texto que contendrá la variable nom1. cabe aclarar, que cuando se van a leer cadenas de texto, no es necesario colocar la dirección (&), lo cual no sucede con los otros tipos de datos:

```
scanf("%f", &n1);
```

Después de haber leído los datos, espera a que se presiones cualquier tecla para finalizar la ejecución del programa.

Ejemplo 2.5

Programa que imprime dos veces, la cadena de texto que se ha introducido:

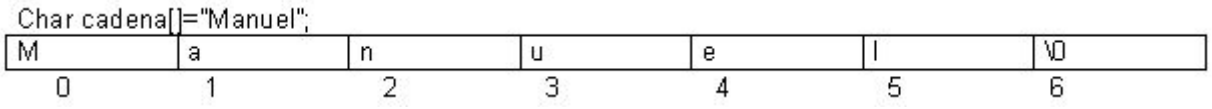
```
#include <stdio.h>

#include <conio.h>
```

```
main()

{

char cadena[15];

printf("Digite la cadena:\n\n");

scanf("%s", cadena);

printf("\n\t LA CADENA ES LA SIGUIENTE:\n\n");

printf("************************************************\n");

printf("%s\n", cadena);

printf("%s\n", cadena);

printf("************************************************\n");

getch();

return 0;

}
```

Es importante, que el lector, intente correr, en su máquina estos ejemplos, para que comprenda con mayor facilidad.

NOTA: Cuando la entrada, es una cadena de carácter, no es necesario el operador direccional (&). El nombre de la cadena contiene la dirección.

Ejemplo:



scanf(), finaliza la captación de la cadena al encontrar un espacio en blanco o fin de línea.

Ejemplo:

```
char cadena[15];

printf("Digite la cadena:\n\n");

scanf("%s", cadena);
```

Casos Especiales

*JUEGO DE INSPECCIÓN: Define Un conjunto de caracteres que puede leerse utilizando scanf().

Así:

%[ABC]s: A, B y C son los únicos caracteres que puede leer al encontrar uno diferente, finaliza con un valor nulo.

%[A-Z]s: También pueden ser rangos de carácter en este caso sólo acepta mayúsculas.

*JUEGO INVERSO: Aquí se declaran que caracteres NO puede tomar, la función scanf(), se utiliza el circunflejo (^), que acepta cualquiera menos...

Ejemplo:

%[^\n]s: Acepta cualquier carácter menos un salto de línea.

%[^0-9]s: Acepta cualquier carácter menos del 0 al 9.

Ejemplo:

Scanf("%[0-9]s", &edad);

Cuestionario

Mencione y Explique que es la lógica estructurada:_____

Para que sirven las funciones getch() y putchar():_____

Menciones las diferencias fundamentales entre las funciones de entrada y salida por consola, con las funciones de entrada y salida por consola con
formato:_____

Escriba algunas restricciones que deben cumplir los
Identificadores:_____

¿Cuál es la siferencia entre el tipo de dato %c, y el tipo de dato
%s?:_____

Para que sirve la directiva <stdio.h>:_____

¿Y la directiva <conio.h>?_____

¿Para que sirve a declaratoria #define?:_____

Para que sirve el punto y coma (;) en C:_____

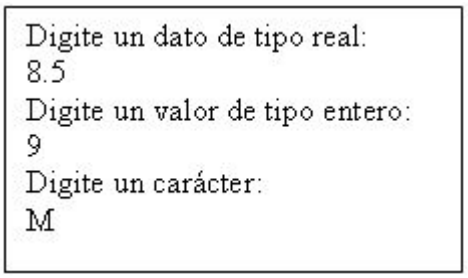
En C, no existe el tipo de dato string; sin embargo, podemos hacer uso de las cadenas de texto, ¿Por qué?.
Explique:_____

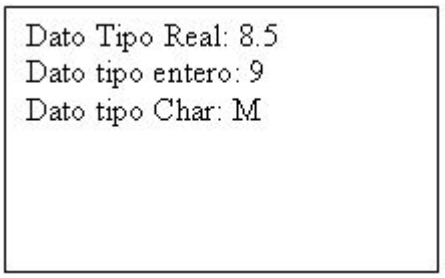
Ejercicios:

Haciendo uso de las funciones gets y puts, diseñe un programa en C, que se lea el nombre del usuario y lo muestre en pantalla junto con un saludo.

Diseñe un programa en C, que lea y muestre en pantalla el valor de tres variables de tipo Entero.

Diseñe un programa que muestre, los diferentes tipos de datos, usados en C. Primero, debe indicársele al usuario que introduzca un valor, de un tipo dado; luego y después de haber introducido valores en todas las variables, debe imprimirse el contenido de ellas, junto con un mensaje que indique, el tipo de dato:





4. Diseñe un programa, en el cual se introduzcan el nombre y el peso y de un alumno, y luego la muestre en pantalla. El Nombre debe incluir el apellido, y en el campo del peso, solo deben incluir valores numéricos.

5. Diseñe un programe en C, en el cual después de haber introducido, una tabla de multiplicación cualquiera, imprima ésta en forma de tabla:

2x2=4

2x3=6

2x4=8

. .

.

2x10=20

6. Realice el ejercicio 2.5, tal como se muestra, luego ejecútalo, nuevamente, pero quitándole al código las sentencias: getch() y return 0. ¿Qué observas? Realiza tus propias conclusiones de ello y de la importancia de estas dos funciones.

Capitulo III "Operadores, Expresiones y Estructuras"

Hasta ahora, prácticamente hemos visto, como el protocolo esencial, para realizar un programa en C; y algunas funciones muy importantes, como son las funciones de lectura e impresión (scanf y printf, respectivamente).

Ahora veremos, otros aspectos fundamentales, como lo son los operadores, que pueden ser: lógicos, matemáticos, relacionales, etc. Las expresiones, y las estructuras: de secuenciación, de selección y de iteración.

Operadores

Un operador, es un símbolo que indica al compilador que se lleve a cabo ciertas manipulaciones matemáticas o lógicas.

Operadores Aritméticos

Operador	Propósito
+	Suma
-	Resta
*	Multiplicación
/	División

%	Resto de la división entera

TABLA 3.1

Todos estos operadores se pueden aplicar a constantes, variables y expresiones. El resultado es el que se obtiene de aplicar la operación correspondiente entre los dos operandos. (Tomado de "Aprenda Lenguaje ANSII C, como si estuviera en primero". Pag. 25).

Los operandos sobre los que actúan los operadores aritméticos deben ser valores Numéricos, es decir datos enteros, punto flotante o de carácter (Int, float y char, respectivamente).

Una aclaración especial, merece el operador "%", que indica el resto de la división entera. Veámoslo con un ejemplo:

Si dividimos 30/3, su cociente es 10, y su residuo es 0. Si dividimos 25/3, su cociente es 8, y tiene un residuo de 1. Entonces de lo que se encarga, este operador, es de devolvernos el valor del residuo de una división. Cabe aclarar que los datos deben de ser tipo entero, y su sintaxis es la siguiente:

25%3

NOTA: Este Operador, NO puede aplicarse a los datos de tipo float.

Una Expresión, Es un conjunto de variable, constantes y otras expresiones más sencillas, relacionadas por algún tipo de operador. De las cuales hablaremos con más detalle, posteriormente.

Operadores de Relaciónales, Lógicos y Unarios

Estos Operadores, los podemos dividir, en varios tipos, entre los cuales están:

OPERADORES UNARIOS: C, incluye una clase de operadores que actúan sobre un solo operador para producir un nuevo valor. Por eso el nombre de unarios, por que para poder funcionar solo necesitan de un operador.

Operador	Propósito
-	Menos Unario: Es el signo menos que va delante de una variable, constante o expresión.
++	Operador Incremento: Hace que la variable, constante o expresión se aumente en uno.
--	Operador Decremento: Hace que su variable, constante o expresión disminuya en uno.

TABLE 3.2

Ejemplo:

```
Int i=1, x=5;

Printf("%d", ++i);

Printf("%d", - -i);
```

Estos operadores, el incremento y el decremento, pueden utilizarse de dos maneras, eso depende del orden de aparición de los mismos:

-Si el operador precede al operando el valor del operando se modifica antes de ser utilizado.

-Si el operador aparece después del operando, este se modifica después de ser utilizado.

Ejemplo 3.1:

Utilizando los operadores Unarios:

```
#include <stdio.h>
#include <conio.h>
main()
{
int x=5;
printf("\tPRIMERO OBSERVAREMOS EL RESULTADO DE ++X\n\n");
printf("%d\n", ++x);
printf("%d\n", ++x);
printf("%d\n", ++x);
printf("\tAHORA OBSERVAREMOS EL RESULTADO DE --X\n\n");
printf("%d\n", --x);
printf("%d\n", --x);
printf("%d\n", --x);
printf("\tEL RESULTADO DE X++ ES:\n\n");
printf("%d\n", x++);
printf("%d\n", x++);
printf("\tY EL DE X-- ES:\n\n");
printf("%d\n", x--);
printf("%d\n", x--);
getch();
return 0;
}
```

- OPERADORES RELACIONALES O DE COMPARACIÓN:

Operador	Significado
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que

==	Igual que (Para las comparaciones)
! =	No igual a

TABLA 3.3

Estos Operadores se encuentran dentro del mismo grupo de procedencia, que es menor que la de los Operadores Unarios y aritméticos.

La Asociatividad de éstos es de izquierda a derecha. Cabe mencionar la diferencia entre los operadores = y ==, el primero (=), se utiliza para asignaciones de valores, mientras que el otro (==), se usa para comparaciones. Ejemplo: Si x>5, entonces x==6.

3. OPERADORES LÓGICOS: Estos son los que nos permiten unir varias comparaciones: 10>5 y 6==6. Los operadores lógicos son: AND (&&), OR (||), NOT(!).

Operador && (AND, en castellano Y): Devuelve un 1 si se cumplen dos condiciones.

printf("Resultado: %i", (10==10 && 5>2);

Operador || (OR, en castellano O): Devuelve un 1 si se cumple una de las dos condiciones.

Operador ! (NOT, negación): Si la condición se cumple NOT hace que no se cumpla y viceversa.

Ver el capítulo Sentencias, sección Notas sobre las condiciones para más información. (Tomado de "Curso de C" por Gorka Urrutia).

Operadores de Asignación

Los Operadores de Asignación, como su nombre lo indica, se encargan de atribuirle, asignarle, confinarle, etc a una variable, el resultado de una expresión o el valor de otra variable.

Se utilizan en forma de expresiones de asignación en los que se asigna en el valor de una expresión a un identificador. El operador de asignación más utilizado es "=" y su formato es:

identificador = expresión;

Donde el identificador representa por lo general una variable y una constante, una variable o una expresión más compleja.

Si los dos operandos de la expresión de asignación son de tipo de datos diferentes el valor de la expresión de la derecha se convertirá automáticamente al tipo de identificador de la izquierda de ésta forma la expresión de asignación será del mismo tipo de datos.

Ejemplo:

*Un valor en coma flotante puede ser truncado, se asigna a un identificador entero.

*Un valor de doble precisión puede ser redondeado si se asigna a un identificador de coma flotante.

En C, están permitidas las asignaciones múltiples, así:

Identificador1 = identificador2 = identificador3.....= identificadorn=expresión

C, posee además los siguientes operadores de asignación:

Operador	Explicación
+=	Expresión1+=expresión2. Equivale a: expresión1=expresión1 + expresión2
-=	i-=1. equivale a: i=i-1
*=	J*=2. Equivale a: j=j*2

/=	K/=m, equivale a: k=k/m
%=	P%n. Equivale a: p=p%n

TABLA 3.4

Los Operadores de asignación tiene menos procedencia que el resto de los operadores y tienen asociatividad de izquierda a derecha.

Ejemplo 3.2

Programa que calcula el valor de la expresión X^2+X+1

```
#include <stdio.h>
#include <conio.h>
main()
{
float x, y, z;
clrscr();
printf("\tPROGRAMA QUE CALCULA EL VALOR DE LA ECUACION X^2+X+1\n\n");
printf("Introduzaca el valor de x:\n");
scanf("%f", &x);
y=x*x;
z=y+x+1;
printf("**************************************\n");
printf("**El valor de la expresi¢n es: %.2f**\n", z);
printf("**************************************\n");
getch();
return 0;
}
```

Jerarquía de Operadores

Categoría del Operador	Operador
1. Operadores Unarios	-, ++, --, !
2.Operadores Aritméticos: 1. Multiplicación, división y Resto entero 2. Suma y Resta	*, /, % +,-
3. Operadores Relacionales	<, <=, >, >=
4. Operadores de Igualdad	==, ! =
5. Operadores Lógicos	&& (Y Lógico), || (NO Lógico)
6. Operadores de Asignación	=, +=, -=, *=, /?, %=,

TABLA 3.5

REGLAS DE JERARQUÍA:

1. Se ejecuta primero el operador de más alta jerarquía
2. Operadores que tienen igual jerarquía se evalúan de izquierda a derecha
3. si existen expresiones encerradas entre paréntesis, estas se evalúan primero.
4. si existen paréntesis anidados se evalúan primero los paréntesis más internos.

EXPRESIONES

(Tomado de "Aprenda ANSII C como si estuviera en Primero", Universidad de Navarra. 1998).

Ya han aparecido algunos ejemplos del lenguaje C en las secciones precedentes. Una *Expresión* es una combinación de variables y/o constantes, y operadores. La expresión es equivalente al resultado que proporciona al aplicar sus operadores a sus operandos. Por ejemplo 1 + 5 es una expresión formada por dos operandos (1 y 5)y el operador (el +); esta expresión es equivalente al valor 6, por lo cual quiere decir que allí donde esta expresión aparece en el programa, en el momento de la ejecución es evaluada y sustituida por su resultado. Una expresión puede estar formada por otras expresiones más sencillas, y puede contener paréntesis de varios niveles agrupando distintos términos. En C, existen diferentes tipos de expresiones. El cual depende del tipo de operadores que se estén utilizando. Por ejemplo: Expresiones lógicas, aritméticas, etc

Se debe hacer hincapié en que, si existen algunas expresiones encerradas entre paréntesis, estas se evalúan primero. Ejemplo:

9*(8+5)

primero sumamos 8+5, cuyo resultado es 13, y este lo multiplicamos por nueve, con lo que la expresión anterior, da cómo resultado: 117.

Si existen expresiones en paréntesis anidadas, es decir, que uno se encuentra dentro de otros paréntesis, se evalúan los más internos. Ejemplo:

2*((20/(12-2))+5)

se evalúa la operación 12-2, que da como resultado 10, luego se divide 20, entre el resultado anterior, es decir 10. el resultado es 2, y a este número se le suma 5, obteniendo 7. ahora se multiplica por dos, para determinar así que la expresión anterior es igual a 14.

Estructuras

Estructuras Secuenciales

Se les denomina así, por que; son estructuras en un programa, que después de ejecutar una instrucción o sentencia, continúan con la otra, hasta llegar al final del programa. Los ejemplos que hemos visto anteriormente, son ejemplos de estructuras secuenciales. Veamos otros ejemplos:

Ejemplo 3.3

Diseñe un programa que calcula el cuadrado y el cubo de tres números introducidos por el usuario.

```
#include <stdio.h>

#include <conio.h>

main()
```

```
{

int x, x1, x2, y, y1, y2, z, z1, z2;

clrscr();

printf("\tPROGRAMA QUE CALCULA EL CUADRADO Y EL CUBO DE 3 NUMEROS\n\n");

printf("Introduzaca el primer n£mero:\n");

scanf("%d", &x);

printf("Ahora ingrese el siguiente n£mero:\n");

scanf("%d", &y);

printf("Y el tercer n£mero es:\n");

scanf("%d", &z);

x1=x*x;

x2=x*x*x;

y1=y*y;

y2=y*y*y;

z1=z*z;

z2=z*z*z;

printf("**********************************\n");

printf("**Numero****Cuadrado*****Cubo****\n");

printf("**%d **** %d ***** %d ****\n", x, x1, x2);

printf("**%d **** %d ***** %d ****\n", y, y1, y2);

printf("**%d **** %d ***** %d ****\n", z, z1, z2);

printf("**********************************\n");

getch();

return 0;

}
```

Ejemplo 3.4

Una empresa necesita conocer el sueldo neto a pagar a un empleado. Teniendo como entrada el salario produzca una salida de sueldo neto. Los descuentos a aplicar son: ISSS 5%, AFP 7% y Renta 10%, estos descuentos son sobre el salario, y es sueldo neto es la diferencia entre el salario y el total de las retenciones:

```
#include <stdio.h>

#include <conio.h>

main()

{

float sueldo, afp, isss, renta, sn;

char nombre[50];

clrscr();

printf("Introduzca el Nombre del empleado:\n");

scanf("%s", nombre);

printf("Su sueldo es:\n");

scanf("%f", &sueldo);

afp=sueldo*0.07;

isss=sueldo*0.05;

renta=sueldo*0.10;

sn=sueldo-(afp+isss+renta);

printf("El empleado %s\n", nombre);

printf("Posee un sueldo neto de %.2f\n", sn);

getch();

return 0;

}
```

Ejemplo 3.5

Diseñe un programa que calcule el promedio y la suma de tres números ingresados por el usuario:

```
#include <stdio.h>

#include <conio.h>

main()

{

float x, y, z, sum, prom;

clrscr();
```

```
printf("El Primer n£mero es:\n");

scanf("%f", &x);

printf("Ahora el segundo n£mero:\n");

scanf("%f", &y);

printf("El Ultimo numero es:\n");

scanf("%f", &z);

sum=x+y+z;

prom=sum/3;

printf("****************************************\n");

printf("**La suma es %.2f y el promedio es %.2f*\n", sum, prom);

printf("****************************************\n");

getch();

return 0;

}
```

Estructuras Selectivas

Los pequeños programas que hemos diseñada hasta el momento, han sido del tipo secuencial, es decir, una sentencia se ejecuta después de otra, hasta el final del programa.

Pero en la vida diaria muchas veces debemos elegir entre un camino y otro para llegar a nuestro destino. Lo mismo pasa en programación, al realizar alguna actividad, nuestro programa debe ser capaz de elegir uno u otro camino, a seguir dependiendo del valor de alguna condición evaluada.

Para ello C, dispone de tres tipos de 3 tipos de estructuras selectivas, la cuales son:

Estructura Selectiva Simple

Estructura Selectiva Doble

Estructura Selectiva Múltiple

ESTRUCTURA SELECTIVA SIMPLE

Funciona de la siguiente manera: se evalúa una condición, de ser cierta efectúa una acción, de lo contrario, continúa con la ejecución normal del programa.

Su sintaxis es la siguiente:

If(condición) Acción;

O también:

If(Condición)

Acción;

Donde:

Condición: Es una expresión lógica que es evaluada por el compilador

Acción: es la Acción o Acciones que realizará el programa de resultar cierta la condición

NOTA: En C, no existe la sentencia "End If", como en otros lenguajes de programación para indicar que ha terminado el bloque de selección, sino que este se especifica con el punto y coma al final. Además que, después de la condición NO se escribe un punto y coma. Si son varias acciones, estas deben ir dentro de llaves {}, para indicarle al compilador que son un solo bloque de acciones que deben ejecutarse.

Ejemplo 3.6

En una tienda se venden artículos de primera necesidad, a los cuales se les aplica un descuento del 20%, de la compra total, si esta es igual o mayor a $50. Diseñe un programa en C, que a partir del importe total de la compra muestre lo que debe pagar el cliente.

```
#include <stdio.h>

#include <conio.h>

main()

{

float compra;

clrscr();

printf("Introduzca el valor de la compra:\n");

scanf("%f", &compra);

if(compra>=50)

compra=compra*0.8;

printf("El Importe de la compra es %.2f\n\n", compra);

getch();

return 0;

}
```

ESTRUCTURA SELECTIVA DOBLE

Esta estructura, se caracteriza por el hecho que ofrece dos caminos a seguir, dependiendo si al evaluar la condición resulta cierta o falsa. Su sintaxis es la siguiente:

if(Condición)

Acción 1;

else

Acción 2;

Funciona, de la siguiente manera si condición, al evaluarla resulta cierta, realiza la acción 1. de lo contrario, es decir; si al evaluar la condición resulta falsa, realiza la acción 2.

Se debe tener en cuenta la condición puede ser compuesta, es decir haciendo uso de los operadores && y || (Y lógico y No lógico), además que cuando tenemos más de una sentencia por ejecutar ya sea del lado del cierto o del falso, estas van dentro de llaves.

Ejemplo 3.7

Se desea saber si un número es par o impar. Diseñe un programa en el cual el usuario, ingrese el número y el programa muestre con un mensaje, si éste es par o no.

```
#include <stdio.h>

#include <conio.h>

main()

{

int num;

printf("Ingrese el n£mero:\n");

scanf("%d", &num);

if(num%2==0)

printf("ES PAR\n\n");

else

printf("ES IMPAR\n\n");

getch();

return 0;

}
```

Ejemplo 3.8

Diseñe un programa, que dada la nota de alumno, imprima en la pantalla un comentario sobre esa nota. El criterio para los comentarios es el siguiente:

Si nota es mayor o igual a 9 "Excelente"

Si nota es mayor o igual a 8 "Muy Bueno"

Si nota es mayor o igual a 7 "Bueno"

Si nota es mayor o igual a 6 "Regular"

Si nota es menor que 6 "Necesita Mejorar"

```
#include <stdio.h>

#include <conio.h>

main()

{

float nota;

printf("Digite la nota, porfavor:\n");

scanf("%f", &nota);

if(nota >= 9.0)

printf("EXCELENTE\n\n");

else

if(nota >= 8.0)

printf("MUY BUENO\n\n");

else

if(nota >= 7.0)

printf("BUENO\n\n");

else

if(nota >=6.0)

printf("REGULAR\n\n");

else

printf("NECESITA MEJORAR\n\n");

getch();

return 0;

}
```

Este ejemplo, muestra que C, permite hacer anidamientos, es decir, una selección dentro de otra, ya sea del lado del cierto, o del falso o de ambos.

El lector, puede tratar de hacer sus propias conclusiones, además de buscar otras posibles solucione para este mismo problema. Por ejemplo, ¿que pasaría si iniciamos con la condición del 6.0?. ¿Qué pasaría si el usuario digita una neta negativa? ¿Cómo podrías darle solución a este problema?. Como programadores, debemos hacernos muchas preguntas al momento de diseñar nuestros programas, ya que estos No serán usados por nosotros, sino por otras personas.

Ejemplo 3.9

Dada el peso, la altura y el sexo, de unos estudiantes. Determinar la cantidad de vitaminas que deben consumir estos estudiantes, en base al siguiente criterio:

>> Si son varones, y su estatura es mayor a 1.60, y su peso es mayor o igual a 150 lb, su dosis, serán: 20% de la estatura y 80% de su peso. De lo contrario, la dosis será la siguiente: 30% de la estatura y 70% de su peso.

>> Si son mujeres, y su estatura es mayor de a 1.50 m y su peso es mayor o igual a 130 lb, su dosis será: 25% de la estatura y 75% de su peso. De lo contrario, la dosis será: 35% de la estatura y 65% de su peso. La dosis debe ser expresada en gramos.

```
#include <stdio.h>

#include <conio.h>

main()

{

float peso, estatura, dosis;

char sexo;

printf("Introduzca el sexo del alumno(a)<H/M>:\n");

scanf("%c", &sexo);

printf("Peso:\n");

scanf("%f", &peso);

printf("La estatura es de:\n");

scanf("%f", &estatura);

if(sexo=='H' || sexo=='h')

{

if(estatura>1.60 && peso >=150)

{

dosis=(0.20*estatura)+(0.8*peso);

printf("La dosis de este alumno ser : %.2f gramos\n\n", dosis);

}
```

```
else

{

dosis=(0.3*estatura)+(0.7*peso);

printf("La dosis de este alumno sera %.2f gramos\n\n", dosis);

}

}

else

{

if(estatura>1.50 && peso >=130)

{

dosis=(0.25*estatura)+(0.75*peso);

printf("La dosis de esta alumna debe ser de %.2f gramos\n\n", dosis);

}

else

{

dosis=(0.35*estatura)+(0.65*peso);

printf("La dosis de esta alumna debe ser de %.2f gramos\n\n", dosis);

}

}

getch();

return 0;

}
```

SELECCIÓN MÚLTIPLE

Como su nombre lo indica, permite seleccionar entre varios caminos para llegar al final. En este caso se pueden elegir un camino o acción a ejecutar de entre varios posibles que se debe de evaluar, llamada selector. Sintaxis:

```
switch(selector)

{

case Etiqueta A:
```

```
Acción A;

break;

case Etiqueta B:

Acción B;

break;

case Etiqueta n:

Acción n;

break;

default:

Excepción;

break;

}
```

En donde:

Selector: Variables, expresiones simples de tipo ordinal, (enteros y caracteres –int y char-)

Etiqueta: Tiene que ser del mismo tipo de datos de selecto. Estas deber ser constantes únicas y diferentes de otras.

Excepción: Es opcional.

Ejemplo 3.10

Diseñe un programa en C, que dado un número del 1 al 3, muestre en pantalla y en letras, el mismo número:

```
#include <stdio.h>

#include <conio.h>

main()

{

int n;

clrscr();

printf("El N£mero es:\n");

scanf("%d", &n);

switch(n)

{
```

```
case 0: puts("Cero");

break;

case 1: puts("Uno");

break;

case 2: puts("Dos");

break;

case 3: puts("Tres");

break;

default: puts("Dato No valido");

break;

}

getch();

return 0;

}
```

Cuestionario

Mencione las diferencias entre las expresiones y los operadores:_____

Que tipo de datos son válidos para los operadores aritméticos:_____

Explique, el resultado de los operadores incremento y decremento, dependiendo de su posición:_____

¿Qué son y para que sirven los operadores unarios?:_____

Explique, el funcionamiento de los operadores de asignación:_____

Ejercicios:

Diseñe un programa que dados tres números indique cual es el mayor de ellos.

Diseñe un programa que dados tres números indique cual de ellos es el menor.

En un cine se exhiben, películas para mayores de edad, diseñe un programa que dada la edad, indique si la persona puede o no ver la película.

En un supermercado, se realizan descuentos por las compras a partir de unas bolitas de colores. Si el cliente saca una bolita color azul, tiene un descuento del 20%, si la bolita es roja, se aplica un descuento del 30% y si saca una bolita color blanca, no se aplica ningún descuento. Diseñe un programa que a partir del importe de la compra y el color de la bolita, muestre lo que debe pagar dicho cliente.

Se procesan las notas de 5, alumnos, de las cuales se desea saber cual es el promedio de esas 5 notas, y cual fue la nota mayor y menor, además de imprimir al final el nombre y la nota de cada alumno en forma de tabla.

un estudiante desea saber cuál fue su promedio en matemática I, para ello dispone de la siguiente información: tiene 3 exámenes, con una ponderación del 20% cada uno y 2 laboratorios con una ponderación del 30% cada uno. Diseñe un programa que dadas las notas calcule el promedio del alumno y muestre en pantalla si el alumno esta reprobado o no (para aprobar esta materia se requiere de una nota mayor o igual a 6.00).

En un estacionamiento, se cobra de la siguiente manera: los primeros 10 minutos son gratis, los siguientes 30 minutos tiene un valor de $0.30 y la hora $0.60. diseñe un programa que reciba tanto minutos como horas y muestre lo que debe cancelar el cliente. Tomando en cuenta que si es Martes y Sábado se hace un descuento del 12.56% sobre el monto total.

Diseñe un programa que al introducir un dígito del 0 a 9, muestre como se lee.

Diseñe un pequeña calculadora que, al digitar un código realice una operación específica: si el código es 1, la operación es la suma, si es 2, Resta. 3, multiplicación y 4 división. Si el usuario a escrito otro código inválido, mostrar un mensaje de error.

Construya un programa que dado el salario de un empleado, permita aplicarle un aumento de 10% si el salario es inferior a $500, si es mayor se le aumentará un 8%. Luego debe aplicar una retención del 0.96% en concepto de Renta a ambos casos.

Se desea calcular el sueldo de un trabajador, a partir de las horas trabajadas en la semana y la clase a la que pertenece: Trabajadores Clase "A", se les paga $7 por hora. Trabajadores clase "B", se paga $5 por hora. Trabajadores clase "C", se les paga $4 por hora y los de clase "D", $3.5 por hora.

Un comerciante se dedica a la venta de sillas únicamente. Vende tres tipos de sillas: tipo A, tipo B y Tipo C los precios son $5.00, $7.00 y $10.00 respectivamente. Por cada cinco sillas compradas del tipo A, del tipo B o del tipo C los clientes reciben un descuento de 3%, 5% y 7%, las demás se cobran a precio normal. Diseñe un programa que imprima en forma de factura, con el nombre, precio unitario, precio total, nombre de la tienda, etc lo que debe cancelar cada cliente en concepto de la compra.

Descubre donde está el error.

El siguiente código, es de un programa que a partir de una nota determina si un alumno esta o no reprobado, y este puede presentar algunos errores de lógica, de sintaxis o de ejecución. ¿Puedes descubrirlos y modificarlos?

```
#Include <stdio.h>

#incluide <conio.h>

main()
```

```
{

float nota;

printf("Digite la nota:\n")

scanf("%f", nota);

if(nota>=6.00)

printf("Aprobado\n\n);

else

printf(Reprobado\n\n);

getch();

return 0;

}
```

Capitulo IV "Ciclos"

Introducción**

Es muy común encontrar en los programas operaciones que se deben ejecutar un número repetido de veces en períodos más o menos espaciados. Si bien las instrucciones son las mismas, los datos sobre los que operan varían. A nuestro alrededor, encontramos problemas que presentan esas características, por ejemplo: el cálculo de la nota final de los estudiantes de Programación I, se realizará tantas veces como alumnos hayan inscritos en dicha asignatura, el cálculo del salario de los empleados de una empresa, etc. En estos casos la solución que se diseñe para un solo grupo de datos se debe repetir tantas veces como sea necesario (de acuerdo al número de estudiantes y de empleados para los ejemplos anteriores).

Los cálculos simples o la manipulación de pequeños conjuntos de datos se pueden realizar fácilmente a mano, pero las tareas grandes o repetitivas son realizadas con mayor eficacia por una computadora, ya que estas están especialmente preparadas para ello.

Para repetir varias veces un proceso determinado haremos uso de los ciclos repetitivos, a los cuales se les conoce con el nombre de estructura repetitiva, estructura iterativa, lazo o bucle.

(Tomado de Los guiones de clase de Introducción a la Informática. Universidad de El Salvador. Año 2005)

En C, podemos encontrar tres tipos de ciclos:

Entrada Asegurada (while)

Ciclo Controlado Por Contador (for)

Hacer Mientras (do.. while)

Este ultimo, no está lógicamente estructurado, por tanto no haremos mucho hincapié en él.

Conceptos Generales

Funcionamiento de Un Ciclo

Un ciclo, funciona de la siguiente manera: Evalúa una condición de resultar cierta, realiza una acción o bloque de acciones, luego vuelve a evaluar la condición y si nuevamente resulta cierta, realiza la (s) acción (es). Cuando la condición de cómo resultado falso, se sale del ciclo y continúa con la ejecución normal del programa.

Acumulador:

Es una variable, que , como su nombre lo indica se encarga de acumular valores. Esto se vuelve muy útil, por ejemplo, cuando queremos encontrar la suma de los números del 0 al 9, en el acumulador, vamos guardando los valores de dichas cifras. Puede ser tanto real como entera. Su valor inicial, en la mayoría de los casos es cero.

Contador:

Es una variable de tipo entero, que nos ayuda, en el programa a contabilizar el número de ejecuciones de una misma acción, de un grupo de alumnos etc. Un acumulador tiene tres valores distintos:

Valor Inicial: es el valor con el cual iniciamos nuestro contador. Generalmente es cero. Esta asignación puede hacerse cuando se declara la variable.

Valor Final: después de la ejecución del ciclo, el valor del contador, será distinto a su valor inicial, este puede ser mayo o menor que el mismo, todo depende si fue una cuenta creciente o decreciente.

Valor de Cambio: Es el valor Constante, en el cual se irá incrementando nuestro contador, este puede ser positivo o negativo; es decir, si la cuanta se realiza de manera ascendente o descendente.

NOTA: el lector no debe confundirse entre las variables tipo acumulador y tipo contador, estas se diferencian unas de otras en que: los contadores, su valor de cambio es una constante, ya que aumenta y disminuyen en el mismo valor, mientras que los acumuladores su valor de cambio no es constante. Un acumulador necesariamente lo inicializamos con cero (o al menos en la mayoría de los casos). Un contador puede iniciar con cualquier valor.

Bandera:

Las variables tipo bandera son aquellas que sólo admiten dos valores: cierto o falso, true o false, hombre o mujer... etc

Ciclo de Entrada Asegurada

La sintaxis es la siguiente:

while(condición)

Acción;

Funciona de la siguiente manera: primero evalúa la condición, si da como resultado cierta realiza la acción, luego vuelve a evaluar la condición, si su resultado es falso, se sale del ciclo y continúa con la ejecución del programa.

Hay que tener mucho cuidado, cuando trabajamos con ciclos, ya que podemos caer en un ciclo infinito, es decir que nunca se sale de él. Lo cual no es un error de sintaxis sino de lógica. Por lo cual en las acciones debemos siempre colocar algo que haga que se modifique el resultado de la condición, lo cual puede ser una bandera, un contador o un acumulador.

Ejemplo 4.1

Programación en C
Diseñe un Programa que imprima los primeros 10 números.

```
#include <stdio.h>

#include <conio.h>

main()

{

int i=1; /*Declaramos nuestro contador con su Valor Inicial*/

while(i<=10) /*Mientras i sea menor o igual a 10:*/

{

printf("%d\t", i);/*Imprimir el valor de i*/

i+=1;/*Aumentar el contador en 1*/

}

getch();

return 0;

}
```

Ejemplo 4.2

Se desea conocer el promedio de los números mayores que cero, en una serie de números ingresados por el usuario. De los cuales no se sabe la cantidad, haciendo uso de una bandera, diseñe un programa en el cual el usuario ingrese los números que desee.

```
#include <stdio.h>

#include <conio.h>

main()

{

int i=0, sum=0, ban=1, n;

float prom;

while(ban==1)

{

printf("Ingrese un n£mero por Favor:\n");

scanf("%d", &n);

if(n>0)
```

```
{

i=i+1;

sum+=n;

}

printf("Desea Ingresar Otro N£mero? (Si=1 y No=0)\n");

scanf("%d", &ban);

}

prom=sum/i;

printf("************************************************************\n");

printf("*** El Promedio de los numeros mayores que cero es: %.2f ***\n", prom);

printf("************************************************************\n");

getch();

return 0;

}
```

Ejercicio 4.3

En un salón se tienen las notas de 14, alumnos; de los cuales se desea saber cual fue el promedio de todas las notas, cual fue la nota mayor y la nota menor. Así como la cantidad de aprobados en el curso (Para Aprobar la asignatura se requiere de una nota mayor o igual a 6.0)

```
#include <stdio.h>

#include <conio.h>

main()

{

float suma=0, prom, menor=11, mayor=-1, nota;

int i=1,j=0;

while(i<=14)

{

printf("Ingrese la Nota del alumno %d:\n", i);

scanf("%f", &nota);

while(nota<0.00 || nota >10.00)
```

```
{

printf("ERROR, la nota debe estar entre 0 y 10\n");

scanf("%f", &nota);

}

if(nota>=6.00)

j=j+1;

if(nota>mayor)

mayor=nota;

if(nota<menor)

menor=nota;

i=i+1;

suma=suma+nota;

}

prom=suma/14;

printf("El Promedio es %.2f\n\n", prom);

printf("El total de Aprobados es %d\n", j);

printf("La Mayor nota fue %.2f\n", mayor);

printf("%.2f corresponde a la nota menor\n", menor);

getch();

return 0;

}
```

Ciclo Controlado por contador.

En algunas ocasiones, sabemos a ciencia cierta el número de veces que se tiene que repetir una misma acción o bloque de acciones. Y para ello es que nos sirve, esta estructura. Su sintaxis es la siguiente:

```
for( valor inicial; condición; incremento)

accion;
```

Donde:

Valor inicial: es el valor con el cual inicializamos nuestra variable de control.

Condición: si la cumple, ejecuta la acción o acciones e incrementa o decrementa la variable de control, sino la cumple la condición, se sale del ciclo.

Incremento; que puede ser positivo o negativo (decremento).

Veamos un ejemplo sencillo:

Ejemplo 4.4:

Diseñe un programa que imprima los primeros 10 números:

```
#include <stdio.h>

#include <conio.h>

main()

{

int i;

for(i=1; i<=10; i++)

printf("%d\t", i);

getch();

return 0;

}
```

Ejemplo 4.5

Diseñe un programa en C, que calcule las compras totales, realizadas por un grupo de 20 amas de casa. Luego con esa información obtenga la media.

```
#include <stdio.h>

#include <conio.h>

main()

{

int i;

float compra, desvia, prom, varinza, sum=0;

for(i=1; i<=10; i++)

{

printf("Ingrese la cantidad que gast¢ la ama de casa %d:\n", i);

scanf("%f", &compra);
```

```
while(compra<0)

{

printf("ERROR, la compra debe ser mayor que cero, vuelva a intentarlo:\n");

scanf("%f", &compra);

}

sum=sum+compra;

}

prom=sum/12;

printf("El promedio de las compras es %.2f\n\n\a", prom);

getch();

return 0;

}
```

Cabe, mencionar que, en el ciclo for, podemos hacer cuentas decrecientes, es decir asignarle un valor grande a nuestra variable de control y luego irla disminuyendo hasta un valor determinado.

Ejemplo 4.6

En un cine, se tienen 3 diferentes clases de boletos. Se pide que diseñe un programa en el cual:

se lea el precio de las 3 clase de boletos

Se lea el numero de boletos vendidos de cada tipo

Calcular cual boleto es el que se vendió menos

El total recaudado en taquilla

Además se sabe que durante el día se realizaron un total de n ventas.

```
#include <stdio.h>

#include <conio.h>

main()

{

float preciob1, preciob2, preciob3, sum=0, sum1=0, sum2=0, sum3=0;

int n, i, boletos1, boletos2, boletos3, boleto;

clrscr();
```

```
printf("\t\tBIENVENIDO(A)\n\n\n");

printf("Ingrese el precio de los boletos 1:\n");

scanf("%f", &preciob1);

while(preciob1<0)

{

printf("ERROR\n");

scanf("%f", &preciob1);

}

printf("Ingrese el precio de los boletos 2:\n");

scanf("%f",&preciob2);

while(preciob2<0)

{

printf("ERROR\n");

scanf("%f", &preciob2);

}

printf("Ingrese el precio de los boletos 3:\n");

scanf("%f",&preciob3);

while(preciob3<0)

{

printf("ERROR\n");

scanf("%f", &preciob3);

}

printf("&uml;Cu ntas ventas se realizaron este d¡a?:\n");

scanf("%d", &n);

while(n<0)

{

printf("ERROR\n");

scanf("%d", &n);
```

```
}

for(i=1; i<=n; i++)

{

printf("Ingrese el Boleto:\n");

scanf("%d", &boleto);

switch(boleto)

{

case 1: printf("Ingrese la cantidad de boletos vendidos:\n");

scanf("%d", &boletos1);

sum1+=boletos1;

sum=sum+(boletos1*preciob1);

break;

case 2: printf("Ingrese la cantidad de boletos vendidos:\n");

scanf("%d", &boletos2);

sum2+=boletos2;

sum=sum+(boletos2*preciob2);

break;

case 3: printf("Ingrese la cantidad de boletos vendidos:\n");

scanf("%d", &boletos3);

sum3+=boletos3;

sum=sum+(boletos3*preciob3);

break;

default: printf("ERROR, Vuelva a intentarlo\n\n");

break;

}

}

clrscr();

if(sum3<sum2 && sum3<sum1)
```

```
printf("Los Boletos que se vendieron menos fueron los boletos numero UNO\n\n");

if(sum2<sum3 && sum2<sum1)

printf("Los Boletos que se vendieron menos fueron los boletos numero DOS\n\n");

if(sum1<sum2 && sum1<sum3)

printf("Los Boletos que se vendieron menos fueron los boletos numero TRES\n\n");

printf("El total recaudado en taquilla, durante este dia fue: %.2f\n\n", sum);

getch();

return 0;

}
```

Ciclo Do... while

Es te ciclo funciona de la siguiente manera, realiza la acción o conjunto de acciones, luego evalúa una condición de resultar cierta vuelve a realizar la/s accion/es. Cuando sea falsa, se sale del ciclo. Esta estructura, no está lógicamente, estructurada, por ello, no hablaremos mucho, sin embargo realizaremos un par de ejemplos, de este ciclo.

Formato :

```
do {

sentencia;

.

} while(<expL>);
```

La diferencia fundamental, entre el ciclo while y do...while, es que en este ultimo, las sentencias se realizarán por lo menos una vez, en cambio, con while, solo se cumplirán mientras se cumpla la condición, lo cual puede ser nunca.

Ejemplo 4.7

Programa que determina si un año es bisiesto o no. Y un año es bisiesto si es múltiplo de cuatro, pero excluyendo aquellos que son múltiplos de 100 pero no de 400

```
#include <stdio.h>

#include <conio.h>

void main()

{

int anio;

char respuesta;

printf("\n\n\nINICIO DEL PROGRAMA\n\n\n");
```

```
printf("\n\nEl programa te pide un anio y te dice exactamente si es bisiesto o no");

do

{

/*ENTRADA DE DATOS*/

printf("\n\nIntroduzca un anio determinado ");

scanf("%d",&anio);

/*PROCESO Y SALIDA DE DATOS*/

if ((anio%4==0 && anio%100!=0)||(anio%400==0)) printf("\n\nEl anio es bisiesto");

else printf("\n\nEl anio no es bisiesto");

printf("\n\nDesea introducir mas datos\n\n");

respuesta=getch();

} while(respuesta=='S' || respuesta=='s');

printf("\n\n\nFIN DEL PROGRAMA\n\n\n");

}
```

NOTA: este código ha sido tomado de "Practicas de Programación en C", de Fernando Muñoz Ledesma. Practica 3, ejercicio 5.

Cuestionario

¿qué es y cómo funciona un ciclo?_____

Cuál es la diferencia entre un contador y un acumulador:_____

¿cuál es la mejor manera de validar datos?:_____

¿cómo se evita un ciclo infinito?:_____

¿Qué diferencia existe entre un ciclo de entrada asegurada y el do... while?:_____

Descubre donde está el error.

Programación en C
El siguiente código muestra la serie:

1^2+2^2+3^2....n^2

en el cual hay errores de lógica, de sintaxis o hasta de ejecución, puedes descubrirlos y corregirlos?

```
#include <stdio.h>

#include <conio.h>

main()

{

int n i, x, sum=0;

printf("Inrtroduzca el valor de n:\n");

scanf("%d", &n);

while(n<0)

{

printf("Error, vuelva a digitar el valor de n:\n");

scanf("%d", n);

}

for(i=1; i<=n, i++)

x=i*i;

sum+=n;

printf("El valor de la suma es:%d\n\n", sum)

getch();

return 0;

}
```

Ejercicios

Se desea conocer la suma de los números enteros, positivos menores que n, el cual es un dato dado por el usuario.

Muestre un programa en c, que imprima en pantalla los números desde un valor inicial, hasta un valor final, ingresados por el usuario, tanto en forma descendente como ascendente.

Diseñe un programa que imprima la serie de Fugonacci, así: 0 1 1 2 3 5 8 13.... hasta un número n dado por el usuario.

Calcule el promedio de edades de un grupo de estudiantes, de los cuales no se conoce la cantidad.

Diseñe un programa que obtenga, la calificación mayor y la calificación menor, de un grupo de 40 estudiantes, además de los nombres de dichos alumnos.

En un país hubieron elecciones para elegir al presidente. El país consta de 7 provincias o regiones, de las cuales se han levantado actas que contiene el total de votos obtenidos por los 4 partidos políticos en dicha región. Diseñe un programa en c, que lea las actas de las 7 provincias, muestre que partido ganó las elecciones y en caso de empate, lo especifique con un mensaje.

en un supermercado, hay 3 departamentos (de ropa, comestibles y perfumería), en lo cuales se realizan un descuento de 5%, 3.5% y 8% respectivamente, por las compras totales mayores de $100.00. diseñe un programa que dado el monto de la compra, realice los descuentos pertinentes por departamento, le indique al usuario a cuanto asciende su nuevo monto e indique, cuanto fue lo recaudado al final del día.

La Empresa, el porvenir s.a de c.v desea conocer lo que debe pagar en concepto de horas extras aun grupo de n empleados. Se sabe que una hora extra diurna, se paga el doble que una hora normal. Y una hora extra nocturna se paga el doble de una hora normal más el 25%. Además que todos los empleados tiene sueldos diferentes, muestre el nuevo sueldo de cada uno de ellos y lo que tendrá que pagar la empresa en concepto de horas extra.

Una compañía de teléfonos, cobra $0.03 por minuto la llamada nacional local, $0.06 por la llamada de larga distancia nacional y $0.10 la llamada de larga distancia internacional. Diseñe un programa que calcule las facturas mensuales de los clientes, sabiendo que, si las llamadas fueron realizadas por la mañana tienen un doble valor, y si los 10 primeros minutos de llamadas locales son gratis, en cualquier horario.

Capitulo V: Funciones en C

La modularización, es una técnica usada por los programadores para hacer sus códigos más cortos, ya que consiste en reducir un gran problema complejo, en pequeños problemitas más sencillos, concentrándose en la solución por separado, de cada uno de ellos.

En C, se conocen como funciones aquellos trozos de códigos utilizados para dividir un programa con el objetivo que, cada bloque realice una tarea determinada.

En las funciones juegan un papel muy importe las variables, ya que como se ha dicho estas pueden ser locales o globales.

Variables Globales: Estas se crean durante toda la ejecución del programa, y son globales, ya que pueden ser llamadas, leídas, modificadas, etc; desde cualquier función. Se definen antes del main().

Variables Locales: Estas, pueden ser utilizadas únicamente en la función que hayan sido declaradas.

La sintaxis de una función es la siguiente:

Tipo_de_datos nombre_de_la_funcion(tipo y nombre de argumentos)

{

acciones

}

donde:

Tipo_de_datos: Es el tipo de dato que devolverá esa función, que puede ser real, entera, o tipo void(es decir que no devolverá ningún valor).

Nombre_de_la_funcion: Es el identificador que le damos a nuestra función, la cual debe cumplir las reglas que definimos en un principio para los identificadores.

Tipo y nombre de argumentos: son los parámetros que recibe la función. Los argumentos de una función no son más que variables locales que reciben un valor. Este valor se lo enviamos al hacer la llamada a la función. Pueden existir funciones que no reciban argumentos.

Acciones: Constituye el conjunto de acciones, de sentencias que cumplirá la función, cuando sea ejecutada. Entre ellas están:

Asignaciones

Lecturas

Impresiones

Cálculos, etc

Una función, termina con la llave de cerrar, pero antes de esta llave, debemos colocarle la instrucción return, con la cual devolverá un valor específico. Es necesario recalcar que si la función no devuelve ningún valor, es decir, es tipo void, no tiene que ir la sentencia return, ya que de lo contrario, nos dará un error.

Pero, es válido que nos hagamos la siguiente pregunta:

¿Cómo es que funcionan los Subprogramas?

A menudo, utilizamos el adjetivo de "Subprogramas", para referirnos a las funciones, así que, el lector debe familiarizarse también con este término.

Los subprogramas se comunican con el programa principal, que es el que contiene a las funciones, mediante parámetros, que estos pueden ser: Parámetros Formales y Parámetros Actuales.

Cuando se da la comunicación los parámetros actuales son utilizados en lugar de los parámetros formales.

Paso de Parámetros

Existen dos formas de pasar parámetros, las cuales son:

Paso por Valor

También conocido como parámetros valor. Los valores se proporcionan en el orden de cálculos de entrada.

Los parámetros se tratan como variables locales y los valores iniciales se proporcionan copiando los valores de correspondientes argumentos.

Los parámetros formales-Locales de una función reciben como inicilaes los valores de los parámetros actuales y con ellos se ejecutan las acciones descritas en el subprograma.

Ejemplo:

A=5;

B=7;

C=proc1(A, 18, B*3+4);

Proc1(X, Y, Z)

Explicación:

Donde, se encuentra c, se está llamando la función, denominada proc1, en la cual se están enviando como parámetros el valor de A, que es cinco; el cual es recibido por la variable X, en la definición de la función proc1; en la misma función, Y tendrá el valor de 18; por que ese es el valor del parámetro formal, mientras que Z, tendrá un valor inicial de 25, ya que ese es el resultado del tercer parámetro que resulta ser una expresión aritmética.

Funciones Definidas Por El Usuario en C

Una función, como ya se ha dicho, es un bloque de código dentro del programa que se encarga de realizar una tarea determinada. Por lo tanto un programa en c debe constar de una o más funciones, y por su puesto no puede faltar la función principal main().

Un viejo adagio dice: Separa y vencerás, lo cual se acopla perfectamente cuando tenemos un programa que es bastante grande; podemos separarlos en pequeños subprogramas (funciones), y concentrarnos en la solución por separados de cada uno de ellos y así resolver un gran problemas, en unos cuantos problemitas más pequeños.

Si un programa, está constituido por más de una función, las llamadas a la misma, pueden realizarse desde cualquier parte del programa, y la definición de ellas debe ser independiente unas de otras.

Por lo tanto sería un grave error el tratar de definir una función dentro de otra.

Una función puede ser llamada desde cualquier parte del programa no sólo una vez, y cuando es llamada, empieza a ejecutar las acciones que están escritas en código.

Para mayor comodidad del lector vamos a ver varios ejemplos, del uso de funciones y a medida que vayamos avanzando se volverán más complejos.

El orden será el siguiente:

Funciones que no devuelven ningún valor

Funciones que devuelven un valor entero

Funciones que devuelven un valor Real

Funciones combinadas

Funciones en las que usamos Menú.

Funciones que no devuelven ningún valor.

Cómo se ha dicho las funciones pueden o no devolver algún valor, para mi parecer, este tipo de funciones son las más sencillas, ya que cuando se llama la función, esta realiza lecturas, asignaciones, cálculos o impresiones, finaliza la ejecución de la función y el programa continúa normalmente.

Ejemplo 5.1

Diseñe un programa que dados dos números enteros determine la suma y cual de ellos es mayor, usando dos funciones diferentes.

```
#include <stdio.h>

#include <conio.h>

void suma (int a, int b); /*Declaraci¢n de la funci¢n*/

void mayor (int a, int b); /*Tipo de dato, nombre de la funci¢n y el tipo y nombre de los argumentos*/

main()

{

int a, b;

printf("Ingrese el valor de a:\n");

scanf("%d", &a);

printf("Ingrese el valor de b:\n");

scanf("%d", &b);

suma(a,b); /*Llamado de la funci¢n*/

mayor(a,b); /*Unicamente el nombre de la funci¢n y de los par metros*/

getch();

return 0;
```

```
}

void suma(int a, int b) /*Definici¢n de la funci¢n*/

{ /*Abrimos llaves al inicio de la definici¢n*/

int sum; /*Declaraci¢n de las variables locales*/

sum=a+b;

printf("El valor de la suma es %d:\n\n", sum);

} /*Fin de la funci¢n suma*/

void mayor(int a, int b)

{

if(a==b)

printf("Son iguales\n\n");

else

{

if(a>b)

printf("El valor de a es mayor que el de b\n\n");

else

printf("El valor de b es mayor que el de a\n\n");

}

}
```

Definición de la Función

La función ha sido declarada, ha sido llamada y por lo tanto deber haber sido definida. Lo cual consta de dos partes, las cuales son:

Que como su nombre lo indica, es la primera línea de la definición de la función y con ella le indicamos al compilador que está en presencia de una función. Su formato es el siguiente:

Tipo_de_dato nombre_de_la_función (tipo y nombre de los argumentos)

La Primera Línea

Cuerpo de la función

Se inicia con una llave "{", y en ella, se pueden realizar asignaciones, cálculos, impresiones, así como la declaración de las variables locales. Puede estar constituidas por estructuras secuenciales, selectivas, iterativas, anidamientos, se pueden llamar otras funciones, etc; finaliza con "}". Puede devolver uno o ningún valor.

Programación en C
Ejemplo 5.2

Diseñe un Programa en C, que Dado un número entero y mayor que cero, Determine si es o no un número Primo. Ojo, los números primos sólo son divisibles por el mismo y por la unidad (1).

```
#include <stdio.h>

#include <conio.h>

void primo (int numero);

main()

{

int numero, ban=1;

clrscr();

while(ban==1)

{

printf("Introduzca el n£mero por favor:\n");

scanf("%d", &numero);

while(numero<0)

{

printf("ERROR, el valor del n£mero debe ser mayor que cero\n");

scanf("%d", &numero);

}

primo(numero);

printf("&uml;Otro n£mero (si=1 y No=0)?\n");

scanf("%d", &ban);

}

getch();

return 0;

}

void primo (int numero)

{

int div, primo=1;
```

```
for(div=2; div<numero; div++)

{

if(numero%div==0)

{

primo=0;

printf("%d NO es primo\n\n\n", numero);

return 0;

}

else

primo=1;

}

if(primo!=0)

printf("%d es primo\n\n\n", numero);

}
```

Las funciones que devuelven algún valor, se les llama PROTOTIPOS DE FUNCIONES:

Antes de usar una función C debe tener conocimiento acerca del tipo de dato que regresara y el tipo de los parámetros que la función espera.

El estándar ANSI de C introdujo una nueva (mejor) forma de hacer lo anterior respecto a las versiones previas de C.

La importancia de usar prototipos de funciones es la siguiente:

Se hace el código mas estructurado y por lo tanto, más fácil de leer.

Se permite al compilador de C revisar la sintaxis de las funciones llamadas.

Lo anterior es hecho, dependiendo del alcance de la función. Básicamente si una función ha sido definida antes de que sea usada (o llamada), entonces se puede usar la función sin problemas.

Si no es así, entonces la función se debe declarar. La declaración simplemente maneja el tipo de dato que la función regresa y el tipo de parámetros usados por la función.

Es una práctica usual y conveniente escribir el prototipo de todas las funciones al principio del programa, sin embargo esto no es estrictamente necesario.

Para declarar un prototipo de una función se indicara el tipo de dato que regresará la función, el nombre de la

función y entre paréntesis la lista del tipo de los parámetros de acuerdo al orden que aparecen en la definición de la

función. Por ejemplo:

int longcad(int n); Lo anterior declara una función llamada longcad que regresa un valor entero y acepta otro valor entero como parámetro.

(Tomado de "Manual de C" de Héctor Tejada Villela)

Ejemplo 5.3

Diseñe un programa, que dado un número entero y mayor que cero, muestre su factorial. (El factorial de 5 es 120; 5x4x3x2x1=120)

```
#include <stdio.h>

#include <conio.h>

int factorial (int num);

main()

{

int num, ban=1;

clrscr();

while(ban==1)

{

printf("Ingrese el valor del n£mero por favor:\n");

scanf("%d", &num);

while(num<0)

{

printf("ERROR, el valor del n£mero debe ser mayor que cero:\n");

scanf("%d", &num);

}

printf("El valor del factorial es %d\n\n", factorial (num));

printf("¨Desea Realizar otro calculo?Si=1 y No=0\n");

scanf("%d", &ban);

}
```

```
getch();

return 0;

}

int factorial (int num)

{

int sum=1, i;

for(i=2; i<=num; i++)

{

sum=sum*i;

}

return (sum);

}
```

Explicación:

Quizá, lo único nuevo, e importante de explicar, radica en la llamada y la definición de la función. Cuando una función nos devolverá un valor entero, al identificador de dicha función debe precederle el tipo de dato. En el lugar, donde llamamos la función, es que aparecerá el valor que nos devuelva, como valor de retorno. En nuestro ejemplo, en una impresión. Y al momento de definirla, no se nos debe olvidar, colocarle la sentencia return(); ya que, mediante esta declaratoria, está retornando el valor calculado.

Pero, que sucede cuando se está trabajando, con valores bastante grandes, al utilizar solamente el int, se producirá un error lógico; ya que como valor de retorno podría ser un cero o una cifra negativa. Por tanto debemos usar el tipo de dato "long int".

Ejemplo 5.4

Diseñe un programa, que dada una cifra entera y mayor que cero, sea elevada a una potencia introducida por el usuario, la cual. (Ejemplo: 5^2=25).

```
#include <stdio.h>

#include <conio.h>

long int potencia (int base, int exponente);

main()

{

int base, exponente;

clrscr();
```

```
printf("La Base es:\n");

scanf("%d", &base);

while (base<0)

{

printf("ERROR, el dato debe ser mayor que cero:\n");

scanf("%d", &base);

}

printf("El Exponente es:\n");

scanf("%d", &exponente);

printf("%d ^ %d es %ld\n\n", base, exponente, potencia(base,exponente));

getch();

return 0;

}

long int potencia (int base, int exponente)

{

long int sum=0, i,x;

for(i=1; i<exponente; i++)

{

x=base*base;

sum=sum+x;

}

return (sum);

}
```

Este método es un poco complejo y puede realizarse de manera más fácil, haciendo uso de las funciones predefinidas en C, de las cuales hablaremos a continuación.

Funciones que devuelven un valor entero

Funciones que Devuelven un Valor Real

Antes que nada, trataremos las funciones predefinidas en C. Ya que C, posee ciertas funciones que nos ayudan hacer nuestros programas más fáciles y utilizar menos código.

NOTA: El que haya decido hablar de las funciones predefinidas en C, en este apartado, no significa que únicamente se utilizan con este tipo de funciones, todo lo contrario. Pero decidí tratarlas hasta aquí, por cuestiones didácticas.

El lenguaje c, cuenta con una serie de funciones de bibliotecas que realizan operaciones y cálculos de uso frecuente.

Para acceder a una función, se realiza mediante el nombre seguido de los argumentos que le servirán a la función a realizar la tarea específica.

Nombre(arg1, arg2,...argn);

*Funciones Matemáticas

Para acceder a ellas, se debe colocar la directiva #include <math.h> en el encabezado del programa.

Función (Sintaxis)	Tipo de Dato	Propósito
acos(d)	double	Devuelve el arco coseno de d
asin(d)	double	Devuelve el arco seno de d
atan(d)	double	Devuelve el arco tangente de d
atan(d1, d2)	double	Devuelve el arco tangente de d1/d2
ceil(d)	double	Devuelve el valor redondeado por exceso, al siguiente entero mayor
cos(d)	double	Devuelve el coseno de d
cosh(d)	double	Devuelve coseno hiperbólico de d
exp(d)	double	Eleva a la potencia d
fabs(d)	double	Devuelve el valor absoluto de d
floor(d)	double	Devuelve el valor redondeado por defecto al entero menor más cercano
log(d)	double	Devuelve el logaritmo natural de d
log10(d)	double	Devuelve el lo. (base10) de d
pow(d1, d2)	double	Devuelve d1 elevado a la potencia d2

sin(d)	Double	Devuelve el seno de d
sinh(d)	double	Seno hiperbólico de d
sqrt(d)	double	Raíz cuadrada de d
Tan(d)	double	Devuelve la tangente de d
tanh(d)	double	Devuelve la tangente hiperbólica de d

Las siguientes funciones se encuentran en las librerías: stdid.h ó stdlib.h:

Función (sintaxis)	Tipo	Propósito
abs(i)	int	Devuelve el valor absoluto de i
ran()	int	Devuelve un entero aleatorio
srand(u)	void	Inicializa el generador de números aleatorios
div(d1/d2)	Double/ int	Devuelve el cociente y el resto de la división
atuf(s)	Double	Convierte la cadena a una cantidad de doble precisión
atoi(s)	int	Convierte cadenas a un entero
atol(s)	long	Convierte cadenas a un entero largo

Hay muchas otras funciones, pero para ahondar más, debes saber cuál es la versión de C, instalada en tu máquina y así verificar cuáles funcionan correctamente; pero por lo general, estas funciones son muy estándar para la mayoría de compiladores.

A continuación, pasaremos a desarrollar una serie de ejercicios, en los cuales haremos uso de la funciones predefinidas en c, así como la modularización, es decir; el uso de funciones definidas por el usuario.

Ejemplo 5.5

Se desea conocer el resultado de las siguientes operaciones:

Ö a+b

|a-b|

Las variables a y b, son de tipo real, y pueden ser positivas o negativas.

```
#include <stdio.h>

#include <conio.h>
```

```
#include <math.h>

double raiz(float a, float b);

double valor_absoluto(float a, float b);

double exponente (float a, float b);

main()

{

float a, b;

clrscr();

printf("\t\tBIENVENIDO\n\n");

printf("Ingrese el valor de a, por favor:\n");

scanf("%f", &a);

printf("Ahora el valor de b:\n");

scanf("%f", &b);

printf("El resultado de la ra¡z cuadrada de %.2f + %.2f es %.2f\n\n", a,b,raiz(a,b));

printf("|%.2f-%.2f| es igual a %.2f\n\n", a,b,valor_absoluto(a,b));

printf("%.2f^%.2f es igual a %f\n\n", a,b,exponente(a,b));

getch();

return 0;

}

double raiz(float a, float b)

{

float x;

double y;

x=a+b;

y=sqrt(x);

return (y);

}

double valor_absoluto(float a, float b)
```

```
{

float x;

double y;

x=a-b;

y=fabs(x);

return (y);

}

double exponente (float a, float b)

{

double x;

x=pow(a,b);

return (x);

}
```

Supongo que, este ejemplo no requiere mayor explicación. Pero me gustaría que el lector, comprenda la gran cantidad de usos que podemos darle, a aquellas funciones matemáticas, junto con las funciones definidas por el usuario, esta es una gran ayuda, ya que ¿se imaginan la cantidad de código que deberíamos colocar, para determinar cosas tan elementales como el valor absoluto?; con estas funciones matemáticas, C, nos ahorra mucho trabajo y código.

NOTA: el lector, NO debe de caer en el error que únicamente, podemos hacer uso de las funciones definidas por el usuario, del mismo tipo de datos. Podemos utilizar varias funciones de diferentes tipos de datos en el mismo programa, lo cual no constituye ningún error, y tampoco una violación a la programación estructurada

ab

A continuación veremos un ejemplo de un programa en el cual utilizamos dos funciones de diferente tipo de dato.

Ejemplo 5.5

El valor del número e se puede aproximar sumando n términos de la serie: e = 1 + 1/1! + 1/2! + 1/3! + ... Escribir un programa que solicite el número de términos de la serie a sumar e informe del valor aproximado de e. Téngase en cuenta que el termino i de la anterior serie se obtiene dividiendo por (i-1). (La exclamación es el factorial).

```
#include <stdio.h>

#include <conio.h>
```

```
#include <stdlib.h>

void calculodatos(int numero);

double factorial(int dato);

void main(void)

{

int numero;

char respuesta;

printf("\n\n\nINICIO DEL PROGRAMA\n\n\n");

printf("\n\nEl programa te calcula el valor del numero e.");

do {

do {

printf("\n\nIntroduzca un numero de terminos de la serie: ");

scanf("%d",&numero);

} while (numero<0);

calculodatos(numero);

printf("\n\n¨Desea introducir mas datos\?\n\n");

respuesta=getch();

system("cls");

} while (respuesta=='s' || respuesta=='S');

printf("\n\n\n\t\tÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ");

printf("\n\t\tÛÛÛ FIN DEL PROGRAMA ÛÛÛ");

printf("\n\t\tÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ\n\n\n");

}

void calculodatos(int numero)

{

register int i;

register double e=1.;

for (i=1;i<=numero;i++) {
```

```
e=e+1./factorial(i);

}

printf("\n\nEl valor de e para %d terminos es %f.",numero,e);

}

double factorial(int dato)

{

register int i;

register double resultado=1;

for (i=dato;i>0;i--) resultado=resultado*i;

return resultado;

}
```

El ejemplo anterior ha sido tomado de "Practicas de C", de Fernando Muñoz Ledesma.
ledesmafernando[arroba]msn.com

Y así como este ejemplo, podemos realizar muchas otras combinaciones de funciones, según necesitemos y lo solicite nuestro programa.

Funciones Combinadas

Funciones en las que usamos Menú

En la práctica, muchas veces debemos diseñar programas, que nos permitan elegir la acción o acciones a realizar, es decir haciendo uso de un menú. El cual, no es más ni menos que la aplicación de un selector múltiple. Un switch.

Veamos un ejemplo.

Ejemplo 5.6

Diseñe un programa, que dado un ángulo, muestre su seno, coseno o tangente; según lo desee el usuario.

```
#include <stdio.h>

#include <conio.h>

#include <math.h>

void seno (float angulo);

void coseno (float angulo);

void tangente (float angulo);

main()
```

```
{

float angulo;

int opcion, ban=1;

clrscr();

while(ban==1)

{

printf("\t\tBIENVENIDO/A\n\n");

printf("Introduzca el valor del angulo, por favor:\n");

scanf("%f", &angulo);

printf("¨Que desea hacer?:\n\n");

printf("*********************************************\n");

printf("**** 1. seno del angulo ****\n");

printf("**** 2. coseno del angulo ****\n");

printf("**** 3. tangente del angulo ****\n");

printf("*********************************************\n");

scanf("%d", &opcion);

while(opcion<0 || opcion>3)

{

printf("ERROR, la opcion debe estar entre 0 y 3:\n");

scanf("%d", &opcion);

}

clrscr();

switch(opcion)

{

case 1: seno (angulo);

break;

case 2: coseno (angulo);

break;
```

```
case 3: tangente (angulo);

break;

}

printf("¨Hay mas datos? (si=1 y no=0)\n");

scanf("%d",&ban);

}

getch();

return 0;

}

void seno (float angulo)

{

float y;

y=sin (angulo);

printf("El seno de %f es %f\n\n", angulo, y);

}

void coseno (float angulo)

{

float y;

y=cos(angulo);

printf("El coseno de %f es %f\n\n", angulo, y);

}

void tangente (float angulo)

{

float y;

y=tan(angulo);

printf("La tangente de %f es %f\n\n", angulo, y);

getch();

}
```

Programación en C
Cuestionario

Mencione y explique, las parte en las que se componen las funciones definidas por el usuario en C:_____

¿Cuál es la diferencia entre las funciones predefinidas en c y las funciones definidas por el usuario?_____

¿En que consiste el paso de parámetros?:_____

¿Cuál es la diferencia entre parámetros formales y actuales?:_____

En que se diferencias las variables locales a las globales:_____

Ejercicios

Realice una pequeña calculadora, utilizando funciones

Diseñe un programa que permita calcular la serie ½ + 1/3 + ¼+ ... 1/n.

Diseñe un programa, que muestre el mayor y el menor de tres introducidos por el usuario.

Se desea conocer el logaritmo natural y el logaritmo base 10 de una serie de números. Así como la suma de dichos valores

Se desea conocer la permutación de dos números distintos. Usando funciones. Diseñe un programa que resuelva dicho problema. (NOTA: 5P3=5!/(5-3)!)

Se desea conocer la equivalencia de dólares a colones (un dólar = 8.75 de colón), la equivalencia de un kilogramos a libras (1kg=2.2lb) y la conversión de kilómetros a millas (1km=0.62millas). realice esta solución mediante un menú.

Calcule lo que debe pagar cada cliente en un almacén; si por cada compra el cliente tiene derecho a sacar un papelito, y dependiendo del color, se efectúan diferentes descuentos. Si el color es blanco, se realiza un descuento del 2.63% sobre la cuenta, si es verde, un descuento de 4.85% y si es rojo, un descuento de 5.02%. se sabe además que si es día lunes o viernes, el porcentaje de descuento es el doble.

El seno de un ángulo, puede aproximarse, de la siguiente manera: sin(x) = x - x^3/3! + x^5/5! - x^7/7! + ..., determine este valor, y usando la función sin(d), luego muestre la diferencia entre estos valores.

En una empresa de electricidad, se cobrar las facturas correspondientes al consumo de kwh, de la siguiente manera: los primeros 100 kwh, se cobran $2.5, lo siguientes 200 kwh, son a $5.00, los 300kwh, siguientes, son cobrados a razón de $7.5, los kwh siguientes se cobran a $7.00. diseñe un programa que permita determinar lo que debe pagar un grupo de clientes al final del mes.

En una empresa de repuestos de automóvil, poseen 10 tipos de repuestos identificados con los números de 1 al 10. durante la semana se realizan diferentes ventas de los repuestos. Se desea saber la cantidad de repuestos que se deben comprar, para actualizar el inventario. El cual se realiza cada cinco días, y se procede de la siguiente manera: cada día se contabilizan el total de facturas, en las cuales se muestran la cantidad de artículos vendidos así como el total de la venta. Con esos datos, indique al usuario cuantos y de que tipo, son los repuestos que se deben comprar así como la ganancia.

Capitulo VI "Estructuras de Datos"

Programación en C

Un array es un identificador que referencia un conjunto de datos del mismo tipo. Imagina un tipo de dato int; podremos crear un conjunto de datos de ese tipo y utilizar uno u otro con solo cambiar el índice que lo referencia. El índice será un valor entero y positivo. En 'C' los arrays comienzan por la posición 0.

Vectores

Un vector es un array unidimensional, es decir, solo usa un índice para referenciar a cada uno de los elementos.

Su declaración será: tipo nombre [tamaño];

El tipo puede ser cualquiera de los ya conocidos y el tamaño indica el número de elementos del vector (se debe indicar entre corchetes [ ]). En el ejemplo puedes observar que la variable i es utilizada como índice, el primer for sirve para rellenar el vector y el segundo para visualizarlo. Como ves, las posiciones van de 0 a 9 (total 10 elementos).

(Tomado de "Introducción al lenguaje de programación de C/C++". Sergio Pacho)

Ejemplo:

```
int num[100]; /*Arreglo de tipo entero compuesto de 100 posiciones*/

char nom[80]; /*Texto de 80 caracteres*/

float x[12]; /*arreglo de 12 elementos punto flotantes */
```

Constante Simbólica

Hace más sencillo o más fácil modificar un programa que utiliza arreglos. Ya que todas las referencias al tamaño del arreglo pueden ser alteradas, cambiando el valor de la constante simbólica.

Ejemplo 6.1

Diseñe un programa que lea un vector de 10 posiciones, luego determine si la quinta posición es positiva, si la primera posición es negativa y si la ultima posición es cero.

```
#include <stdio.h>

#include <conio.h>

#define N 10

main()

{

float x[N];

int i;

for(i=0; i<N; i++)

{

printf("Ingrese el valor %d:\n", i);

scanf("%f", &x[i]);
```

```
}

if(x[4]>0)

{

printf("La quinta Posici¢n es Positiva\n\n");

}

if(x[0]<0)

{

printf("La 1ø Posici¢n es Negativo\n\n");

}

if(x[N-1]==0)

{

printf("La Ultima Posici¢n es cero\n\n");

}

getch();

return 0;

}
```

Explicación

En este ejemplo estamos haciendo uso de la constante simbólica, de cuyos beneficios ya se habló. La definimos con 10 posiciones, recordando que C, empieza a contar desde cero. Luego definimos el vector llamado x, como punto flotante, y ojo, que éste va acompañado de su dimensión. Luego haciendo uso de un for, empezamos a llenar el vector. Luego preguntamos si la posición cuatro es positiva. El lector se preguntará el por que, la instrucción es x[4] y no x[5], ya que; lo que queremos es averiguar, si la posición cinco es la positiva. Pues bien, la posición identificada con el número cuatro, es en efecto la que contiene el quinto número.

Así:

-2	3	5	6	8	4	-6	1	6	7
0	1	2	3	4	5	6	7	8	9

Esta es una versión gráfica, de lo que sucedería al llenar nuestro vector con los valores indicados. Como podemos ver, C empieza a enumerar las casillas desde el cero, hasta el 9. totalizando de esa manera 10 posiciones. Así que, x[4]=8.

Es por ello, que el for, lo inicializamos con cero, hasta un valor menor que el de la constante, ya que de lo contrario nos daría un error.

Una particularidad con los vectores de tipo char (cadena de caracteres), es que deberemos indicar en que elemento se encuentra el fin de la cadena mediante el carácter nulo (\0). Esto no lo controla el compilador, y tendremos que ser

nosotros los que insertemos este carácter al final de la cadena. Por tanto, en un vector de 10 elementos de tipo char podremos rellenar un máximo de 9, es decir, hasta vector[8].Si solo rellenamos los 5 primeros, hasta vector[4], debemos asignar el carácter nulo a vector[5]. Es muy sencillo: vector[5]='\0';

Ahora veremos un ejemplo de como se rellena un vector de tipo char. Podemos ver que en el for se encuentran dos condiciones:

1.-Que no se hayan rellenado todos los elementos (i<19).

2.-Que el usuario no haya pulsado la tecla ENTER, cuyo código ASCII es 13.

(cadena[x-i]!=13).

Uso de Vectores dentro de las Funciones

Un vector, solo puede ser argumento formal, es decir; por el momento, no podemos enviarlo como valor de retorno., digo por el momento por que cuando hablemos de punteros, veremos que si se pueden enviar.

Y dicho proceso se realiza de la siguiente manera:

Declaración o Prototipo:

Tipo_de_dato nombre de la funcion (tipo_de_dato[]);

Llamado de la Función

Nombre_de_la_funcion(nombre del vector);

Definición de la función

Tipo_de_dato nombre de la funcion (tipo_de_dato nombre[])

Ejemplo 6.2

Diseñe un programa en C, que lea un vector de un máximo de 20 posiciones, y luego determine:

-La suma de todos los valores

-El mayor de los valores, así como la posición del mismo.

se sabe que dichos datos son de tipo entero

```
#include <stdio.h>

#include <conio.h>

#define N 20

int suma (int [N]); /*Declaración de la función*/

void mayor (int [N]); /*Ojo, los argumentos que son vectores solo necesitan el tipo de dato y la dirección*/

main()

{
```

```
int numeros[N], i;

printf("Ingrese la Cantidad de Valores:\n");

scanf("%d", &limite);

while(limite<0 || limite >N)

for(i=0; i<N; i++)

{

printf("Ingrese el elemento %d del vector:\n", i);

scanf("%d", &numeros[i]);

}

printf("La suma de todos los elementos del vector es: %d\n", suma(numeros));

mayor(numeros); /*Llamado de la función */

getch();

return 0;

}

int suma (int numeros [N]) /*Definición de la función */

{

int sum=0, i;

for(i=0; i<N; i++)

sum=sum+numeros[i];

return (sum);

}

void mayor (int numeros [N])

{

int pos=0, mayor=numeros[0], i;

for(i=1; i<N; i++)

{

if(numeros[i]>mayor)

{
```

```
mayor=numeros[i];

pos=i;

}

}

printf("El valor mayor es %d y esta en la posici¢n %d\n\n", mayor, pos);

}
```

El lector, debe preguntarse, que pasaría si existen dos valores exactamente iguales que sean los valores máximos y que por ende, estén en diferentes posiciones, que solución le darías como programador?... este tipo de preguntas debe hacerse siempre que ha finalizado un programa, y nunca dejar nada sin resolver, por que recordemos que los programas que diseñamos son para que otras personas los usen.

Matrices

Las matrices se declaran de forma análoga, con corchetes independientes para cada

subíndice. La forma general de la declaración es:

tipo nombre[numero_filas][numero_columnas];

donde tanto las filas como las columnas se numeran también a partir de 0. La forma de

acceder a los elementos de la matriz es utilizando su nombre, seguido de las expresiones

enteras correspondientes a los dos subíndices, entre corchetes.

En C tanto los vectores como las matrices admiten los tipos de las variables escalares

(char, int, long, float, double, etc.),

Las matrices en C se almacenan por filas, en posiciones consecutivas de memoria. En

cierta forma, una matriz se puede ver como un vector de vectores-fila. Si una matriz tiene N

filas (numeradas de 0 a N-1) y M columnas (numeradas de 0 a la M-1), el elemento (i, j)

ocupa el lugar:

posición_elemento(0, 0) + i * M + j

A esta fórmula se le llama fórmula de direccionamiento de la matriz.

(Tomado de "Aprenda Lenguaje ANSI C como si estuviera en Primero". Universidad de Navarra).

Ejemplo 6.3

Diseñe un programa que lea un matriz de 6*6 y luego determine la suma de cada una de las filas y la almacene en un vector llamado suma.

```
#include <stdio.h>
```

```
#include <conio.h>

#define F 6

#define C 6

main()

{

int matriz[F][C], i,j, vector [F]={0,0,0,0,0,0};

for(i=0; i<F; i++)

for(j=0; j<C; j++)

{

printf("Ingrese el elemento F=%d y Columna=%d de la matriz:\n", i,j);

scanf("%d", &matriz[i][j]);

vector[i]=vector[i]+matriz[i][j];

}

printf("La Matriz generada es:\n\n");

for(i=0; i<F; i++)

{

for(j=0; j<C; j++)

{

printf("*%d*", matriz[i][j]);

}

printf("\n");

}

printf("Y el vector suma de las filas es:\n\n");

for(i=0; i<F; i++)

printf("%d\t", vector[i]);

getch();

return 0;

}
```

Creo que no hay mucho por explicar, el uso de una matriz en C, es bastante parecido al de un vector, pero con las diferencias que en un vector tenemos únicamente una dimensión y en las matrices tenemos dos.

A continuación desarrollaremos un ejemplo, el cual es bastante significativo para mi, ya que fue mi primer proyecto, cuando cursé la Materia de Programación I en la Universidad, espero que les guste:

Ejemplo 6.4

Escriba un programa que visualice un cuadro mágico de orden impar N, comprendido entre 3 y 11; el usuario debe elegir el valor de N. Un cuadro mágico se compone de números enteros entre 1 y N, la suma de los números que figuran en cada fila, columna y diagonal son iguales.

Ejemplo:

8	1	6
3	5	7
4	9	2

Un método de generación consiste en situar en el centro de la primera fila, el número siguiente en la casilla situada por encima y a la derecha, y así sucesivamente ... el cuadro es cíclico, la línea encima de la primera, es de hecho, la última y la columna a la derecha de la ultima es la primera. En caso de que el número generado caiga en una casilla ocupada, se elige la casilla situada de bajo del número que acaba de ser situado.

(Un poco complicado de entender... ¿verdad?... no te preocupes, a mi me costó un poco de tiempo entenderlo, para darle solución)

```
#include <stdio.h>

#include <conio.h>

#define N 11

#define M 11

int comprueba (int [N][M], int dim);

void impresion (int [N][M], int dim);

main()

{

int cuadrado[N][M];

int dim, f, c, i;

clrscr();

printf("Introduzca la dimenci¢n por favor:\n");

scanf("%d", &dim);
```

```
while(dim<3 || dim>11)

{

printf("ERROR, el valor de la dimenci¢n debe estar entre 3 y 11:\n");

scanf("%d", &dim);

}

while((dim%2)!=1)

{

printf("ERROR el valor de la dimenci¢n debe ser Impar:\n");

scanf("%d", &dim);

}

for(f=0; f<dim; f++)

for(c=0; c<dim; c++)

cuadrado[f][c]=0;

f=0;

c=dim/2;

cuadrado[f][c]=1;

for(i=2; i<=dim*dim; i++)

{

f--;

c++;

if(f<0 && c==dim)

{

f=1;

c=dim-1;

}

if(f<0)

f=dim-1;

if(c==dim)
```

```
c=0;


if(cuadrado[f][c]!=0)

{

c--;

f=f+2;

}

cuadrado[f][c]=i;

}

printf("La constante m gica es: %d\n\n", comprueba (cuadrado, dim));

impresion(cuadrado, dim);

getch();

return 0;

}

int comprueba (int cuadrado [N][M], int dim)

{

int magic=1, f,c, consmagic, sum=0, i, j=-1;

consmagic=((dim*dim*dim)+dim)/2;

for(f=0; f<dim; f++)

{

sum=0;

for(c=0; c<dim; c++)

sum=sum+cuadrado[f][c];

if(sum!=consmagic)

magic=0;

}

for(c=0; c<dim; c++)

{
```

```
sum=0;

for(f=0; f<dim; f++)

sum=sum+cuadrado[f][c];

if(sum!=consmagic)

magic=0;

}

sum=0;

for(i=0; i<dim; i++)

sum=sum+cuadrado[i][i];

if(sum!=consmagic)

magic=0;

sum=0;

for((i=dim-1); i>=0; i--)

{

j=j+1;

sum=sum+cuadrado[i][j];

}

if(sum!=consmagic)

magic=0;

if(magic==0)

consmagic=0;

return (consmagic);

}

void impresion (int cuadrado[N][M], int dim)

{

int f, c;

printf("\tEL CUADRO GENERADO ES:\n\n");

for(f=0; f<dim; f++)
```

```
{

for(c=0; c<dim; c++)

printf("*%d*", cuadrado[f][c]);

printf("\n");

}

}
```

Cuestionario

¿Qué es una array o arreglo?_____

¿Cuál es la diferencia entre un vector y una matriz?_____

¿Cómo se define y se declara una función cuyos parámetros son vectores o matrices?_____

¿Cuáles son los tipos de datos admitidos para los arreglos?:_____

¿Cuáles son las diferencias fundamentales entre un arreglo y una variable simple?_____

Ejercicios

En una escuela se tiene el listado de 30 alumnos con sus respectivas notos, diseñe un programa que muestre las notas de los alumnos que tuvieron una nota mayor que el promedio.

Diseñe un programa que dado un vector de magnitud X, busque y muestre la posición en la que se encuentra un valor N, dentro del vector

Se tiene dos vectores A y B, diseñe una solución, en la cual, en un tercer vector se guarde la multiplicación de los vectores A y B, y luego se impriman los tres vectores, uno a la par del otro, en forma vertical

Diseñe un programa en C, en el cual guarde un vector de 100 posiciones, determine la media y la desviación estándar.

Almacenar 50 números en un vector, elevar al cuadrado cada valor almacenado en el vector, almacenar el resultado en otro vector. Imprimir el vector original y el vector resultante

Diseñe un algoritmo y programa que lea dos vectores A y B de 20 elementos cada uno y sume el primer elemento de A con el ultimo elemento de B y luego el segundo elemento de A por el diecinueveavo elemento de B y así sucesivamente hasta llegar al veinteavo elemento de A por el primer elemento de B. El resultado de la suma almacenarlo en un vector C.

Se desea conocer la suma de la diagonal mayor y la diagonal menor de una matriz de F*C, e indique la diferencia matemática entre ambos resultados.

En una tiemda, hay 8 departamentos y se tiene el registro de las ventas del año pasado de cada departamento por mes. Se desea conocer: el departamento que tuvo mayores ventas a lo largo de año. El departamento que tuvo menores

ventas en el año. El mes en que se vendió más en el departamento número 3 y los meses y el departamento que superó las ventas promedios así como el total de lo vendido a lo largo de año.

Se tienen dos Matrices de tamaño 4x4, se pide escriba un programa en el cual,. Mediante un menú, se puedan sumar, multiplicar o dividir las matrices.

10. El departamento de policía de la ciudad de San Salvador ha acumulado información referente a las infracciones de los límites de velocidad durante un determinado periodo de tiempo. El departamento ha dividido la ciudad en cuatro cuadrantes y desea realizar una estadística de las infracciones a los límites de velocidad en cada uno de ellos. Para cada infracción se ha preparado una tarjeta que contiene la siguiente información:

· numero de registro del vehículo;

· cuadrante en el que se produjo la infracción

· limite de velocidad en milla por hora

Diseñe un programa para producir 2 informes; el 1o. Que contiene una lista de la multa de velocidad recolectadas, donde la multa se calcula como la suma del costo de la corte ($20,000) mas $ 1,250 por cada mph que exceda la velocidad limite. Prepare una tabla con los siguientes resultados:

INFRACCIONES A LOS LIMITES DE VELOCIDAD			
Registro del vehículo	Velocidad registrada (MPH)	Velocidad limite	Multa

Este informe debe ser seguido de un segundo en el cual se proporcione un análisis de las infracciones por cuadrante. Para cada uno de los 4 cuadrantes mencionados, debe darse el numero de infracciones y la multa promedio.