|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Rodriguez Espino Claudia |
| *Asignatura:* | Fundamentos de Programacion |
| *Grupo:* | 1102 |
| *No de Práctica(s):* | Practica numero 7 |
| *Integrante(s):* | Gutierrez Orozpe Luis Fernando |
|  |  |
|  |  |
| *Semestre:* | 1er Semestre. |
| *Fecha de entrega:* | 29 de Septiembre de 2017 |
| *Obervaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

OBJETIVO

Elaborar porgramas en lenguaje C utilizando las instrucciones de control de tipo secuencia, para realizar la declaración de variables de diferentes tipos de datos, asi como efectuar llamadas a funciones externas de entrada y salida para asginar y mostrar valores de variables y expresiones.

ACTIVIDADES

Durante el desarrollo de esta practica, al igual que en la anterior, pudimos seguir conociendo mas acerca del lenguaje de programación C. En esta ocasión aprendimos a poder agregar comentarios dentro del código de nuestro programa, conocimos más a fondo la manera en la que se deben declarar las variables y utilizamos operadores como él (cast) que nos sirven para cambiar el resultado de una variable. A continuación, se hablará un poco acerca de lo visto y posteriormente se anexarán los programas u actividades a realizar.

**COMENTARIOS**

Como primer punto tenemos a los comentarios que pueden ser de dos tipos: el comentario por línea y el comentario por bloque. El comentario por línea inicia cuando se insertan los símbolos ‘//’ y termina con el salto de línea (hasta que termine el renglón)

El comentario por bloque inicia cuando se insertan los símbolos ‘/\*’ y termina cuando se encuentran los símbolos ‘\*/’. Cabe resaltar que el comentario por bloque puede abarcar varios renglones.

Para efectos de la esta práctica y de las actividades que se mostraran a continuación, utilice el tipo de comentarios por bloque para especificar en que consiste cada programa, así como sus condiciones.

**TIPOS DE DATOS**

En el lenguaje de programación C existen muchos tipos de datos, sin embargo, los datos básicos son:

**Caracteres:** codificación definida por la máquina. Este tipo de dato tiene como especificador de formato la siguiente sintaxis; **%c, %d, %i, %o, %x.**

**Enteros:** números sin punto decimal. Este tipo de dato tiene como especificador de formato la siguiente sintaxis; **%d, %i, %ld, %li, %o, %x.**

**Flotantes:** números reales de precisión normal. Este tipo de dato tiene como especificador de formato la siguiente sintaxis; **%f, %lf, %e, %g**

**IDENTIFICADOR**

También pudimos conocer que un identificador es el nombre con el que se va a almacenar en memoria un tipo de datos. Los indicadores deben iniciar con una letra (a-z) y pueden contener letras mayusculas (A-Z), minúsculas (a-z), números (0-9) y el carácter guion bajo (\_).

**PROGRAMA NUMERO 1: SUMA DE DOS NUMEROS.**

Para comprobar los conocimientos ya adquiridos durante las clases de teoría y en el laboratorio se nos pide como actividades realizar una serie de programas con ciertas características las cuales se describirán con cada programa. En este primer programa se pide que se pueda hacer la suma de dos números, por lo que su código fuente es el siguiente:

#include <stdio.h>

#include <math.h>

/\*

Este programa realiza la

suma de dos numeros ya sean

reales o enteros

\*/

int main ()

{

float a,b,s;

printf("Escribe la primera cantidad a sumar\n");

scanf("%f",&a);

printf("Escribe la otra cantidad\n");

scanf("%f",&b);

s=a+b;

printf("El resultado de la suma es: %f \n", s);

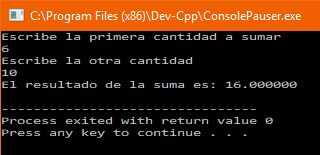
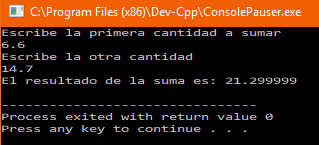
return 0;

}

Como ya sabemos, antes de iniciar cualquier programa en lenguaje C es necesario consultar las librerías adecuadas para el programa, por lo tanto se llama a la librería <stdio.h> y <math.h> aunque esta última no es muy necesaria para este programa.

Se puede apreciar que antes de iniciar con la función “main” se anexaron comentarios del tipo bloque que nos está especificando lo que el programa va a realizar.

Posteriormente está el código en el que utilizamos variables de tipo flotante leídas debidamente con su especificador de formato “%f”

Como en los comentarios se especificó que el programa podía realizar sumas tanto de números reales como enteros, al ejecutarlo se comprueba la validez del comentario.

**PROGRAMA 2: AREA DEL CIRCULO CON “PI” DECLARADA COMO CONSTANTE**

En este programa, a diferencia del anterior, se nos especifica que el valor de “pi” debe ser declarado como una constante para que el programa solo nos pida el valor del radio y no necesite que nosotros le especifiquemos el valor de “pi”, el código fuente es el siguiente:

#include <stdio.h>

/\*

Este programa realiza la operacion para

obtener el area de un circulo pero con la

condicion de que el valor de "pi" sea una

constante

\*/

int main ()

{

const double PI = 3.141592653589;

float radio,area;

printf("Dame el valor del radio: \n");

scanf("%f",&radio);

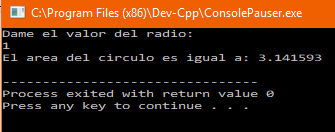
area = PI\*radio\*radio;

printf("El area del circulo es igual a: %f \n", area);

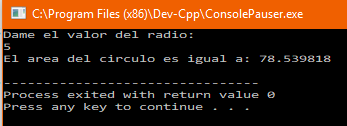
return 0;

}

Se puede observar que se llamó a la librería <stdio.h> y se insertaron comentarios de tipo bloque que nos están especificando que el valor de “pi” debe ser declarado como una constante por lo que en el código se declara como “const doublé Pi = 3.141592653589”. De esta manera, cuando el programa se esté ejecutando, no será necesario especificarle el valor de PI ya que ya lo sabe. Así mismo se utilizan variables del tipo flotante para definir al radio y a el área.



Para comprobar que el valor de “PI” fue declarado como una variable se ejecutó el programa dándole como valor de radio = 1, por lo que el programa debe devolvernos el valor declarado de “PI” como constante y efectivamente, ese fue el resultado.



Por otro lado, se introdujo el valor de radio = 5 y al ejecutarse nos dio el resultado correcto. Es importante señalar que las variables “radio” y “área” se declararon como flotantes ya que de haberse declarado como enteras, no nos mostraría el resultado correcto ya que el valor de la constante “PI” es de modo flotante.

**PROGRAMA 3: PROMEDIO DE 3 NUMEROS CON MOLDEADO DE RESULTADO**

En este programa se pide que se utilice el moldado de resultado o (cast) el cual permite que al hacer operaciones entre números, el resultado no nos de un tipo de dato diferente, obligando al programa a que si el resultado es de modo flotante, nos de el resultado de tipo entero, el código de fuente es el siguiente;

#include <stdio.h>

/\*

Este programa nos ayuda a obtener

el promedio de tres numeros con

la condicion de que se debe utilizar

el moldeado de resultado

\*/

int main ()

{

float a,b,c;

float suma;

float prom;

int promedio;

printf("Dame el primer numero: \n");

scanf("%f",&a);

printf("Dame el segundo numero: \n");

scanf("%f",&b);

printf("Dame el tercer numero: \n");

scanf("%f",&c);

suma=a+b+c;

prom=suma/3;

promedio = (int)prom;

printf("El promedio es igual a: = %d \n", promedio);

return 0;

}

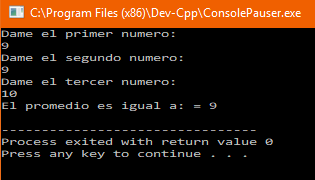
Al igual que en los programas anteriores, se utilizó la librería <stdio.h> y se puso un comentario de tipo bloque en el que nos esta especificando que en el programa se esta utilizando el moldeo de resultado o cast.

Como se puede observar, se declararon las variables “a,b,c” de tipo flotante, estas variables serán las 3 cantidades que el usuario va a ingresar a modo de saber el promedio de ellas tres.

Posteriormente, también se declaró una variable de tipo flotante llamada “suma” que es la que se encargará de almacenar el resultado de la suma de los tres valores que el usuario haya ingresado.

De la misma forma se declaró una variable llamada “prom” de tipo flotante que es la que se encargara de dividir el resultado de la suma de las cantidades sobre 3, que es el número de cantidades ingresadas, debido a esto, si le digo al programa que me imprima la variable “prom” al ejecutar el programa, me dará una variable de tipo flotante y sin embargo, yo quiero que el resultado sea entero.

Por lo tanto, para evitar que esto suceda, se hace función de (cast) ya que esta convierte mi variable “prom” de tipo flotante a una variable “promedio” de tipo entero. Por lo tanto, al querer sacar el promedio de 3 números, me dará el resultado sin decimales.



Al ejecutar el programa y darle los valores 9, 9 y 10, la suma de estos valores es igual a 28 y estos divididos sobre 3 debería dar el resultado de 9.3, sin embargo, como le pedimos a nuestro programa que convirtiera esa variable de tipo flotante a una entera, el resultado del promedio de esos tres números es igual a 9.

CONCLUSIONES

Creo que conforme vamos avanzando en el curso y en las practicas me gusta mucho mas la materia, ya que nos pide que realicemos programas que, aunque no tienen un grado de dificultad muy alto, si nos exige que cumplamos con ciertas condiciones para que el programa se ejecute como debe de hacerlo. Esto, a mi parecer, opino que nos servirá como experiencia para que cuando entremos al mundo laboral, podamos cumplir con las especificaciones de programa que el cliente nos pida.