	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Adrian Ulises Mercado Martínez

Asignatura: Fundamentos de programación

Grupo: 07

No de Práctica(s): 06

Integrante(s): Arredondo Granados Gerardo
Casillas Herrera Leonardo Didier
Diaz Gonzalez Rivas Angel Iñaqui
Galván Castro Martha Selene

No. de Equipo de cómputo pleado: 20

No. de Lista o Brigada: 04

Semestre: 2022-1

Fecha de entrega: 09 de Noviembre del 2021

Observaciones:

CALIFICACIÓN:_____

Práctica 6
Entorno y fundamentos de Lenguaje C

Índice

Objetivo: 3

Introducción 3

Desarrollo 8

Conclusiones 17

Conclusión Grupal 17

Conclusiones individuales 17

Referencias 18

Objetivo:

El alumno elaborará programas en lenguaje C utilizando las instrucciones de control de tipo secuencia, para realizar la declaración de variables de diferentes tipos de datos, así como efectuar llamadas a funciones externas de entrada y salida para asignar y mostrar valores de variables y expresiones.

Introducción

Como hemos visto a lo largo de este curso, el escribir un código tiene varias etapas, no solo es en sí escribir dicho código, primero se tiene que pasar por un proceso de identificación de los datos de entrada y los datos de salida, se tiene que haber diseñado previamente un algoritmo el cual dará respuesta a nuestro problema, y se necesita representar todo esto de alguna manera gráfica o escrita, donde nosotros podemos hacer el diagrama de flujo o pseudocódigo, después de que pasamos por todas estas etapas anteriores, ahora nos es posible escribir el código.

Entorno de C

En este curso estamos aprendiendo el lenguaje C, para poder ejecutar el programa deseado, se requiere escribir el código de acuerdo a las reglas que se tienen para este lenguaje, teniendo el código correcto, es necesario un compilador para lograr compilar el código. Un compilador en pocas palabras, toma nuestro código, y lo transforma en algo que la máquina pueda entender, y lo más importante, ejecutar, el compilador nos devolverá un archivo que es posible ejecutar para la computadora.

Editores

Un programa en lenguaje C primero debe ser escrito en un editor de texto para que después lo podamos compilar, y como vimos en la sección anterior, generar un archivo ejecutable para poder correr el programa. Es por esto que los editores de texto son tan importantes, es la base en donde nosotros literalmente escribiremos el código. Cabe aclarar que un editor de texto no es lo mismo que un procesador de texto, este último permite dar un formato más libre al texto, a la hoja en donde estemos escribiendo el texto, permite insertar imágenes, entre otras cosas.
A continuación, se presentar algunos de los editores de texto más conocidos:

Editor Visual Interface de GNU/Linux (vi)

Este es el editor más común en cualquier distribución de sistemas operativos con núcleo basado en UNIX. Éste se encuentra disponible en la línea de comandos, si se tiene un sistema operativo con entorno gráfico se puede acceder a este desde la terminal de la computadora.

Para iniciar *vi*, debe teclearse desde la línea de comandos:

vi nombre_archivo[.ext]

En este ejemplo, **nombre_archivo** es el nombre del archivo que buscamos editar o el nombre del archivo que queremos crear, .ext se refiere a la extensión que tiene nuestro archivo, significa que el texto es un programa escrito en algún lenguaje o es texto plano.

Modo comando:

Dentro de este editor, nosotros como usuarios tenemos varios comandos a nuestra disposición, al teclear ciertas teclas se ejecutan diversas acciones predeterminadas, cabe aclarar que los comandos son sensitivos a las mayúsculas y minúsculas. Algunos de estos comandos son:

- ↑ o *k* mueve el cursor hacia arriba.
- ↓ o *j* mueve el cursor hacia abajo.
- ← o *h* mueve el cursor hacia la izquierda.
- → o *l* mueve el cursor hacia la derecha.
- *1G* lleva el cursor al comienzo de la primera línea.
- *G* lleva el cursor al comienzo de la última línea.
- *x* borra el carácter marcado por el cursor.
- *dd* borra o corta la línea donde está el cursor.
- *ndd* donde *n* es la cantidad de líneas que se borrarán o cortarán después del cursor.
- *D* borra o corta desde la posición de cursor hasta el final de la línea.
- *dw* borra o corta desde la posición del cursor hasta el final de una palabra.
- *yy* copia la línea donde está el cursor.
- *p* pega un contenido copiado o borrado.
- *u* deshace el último cambio.

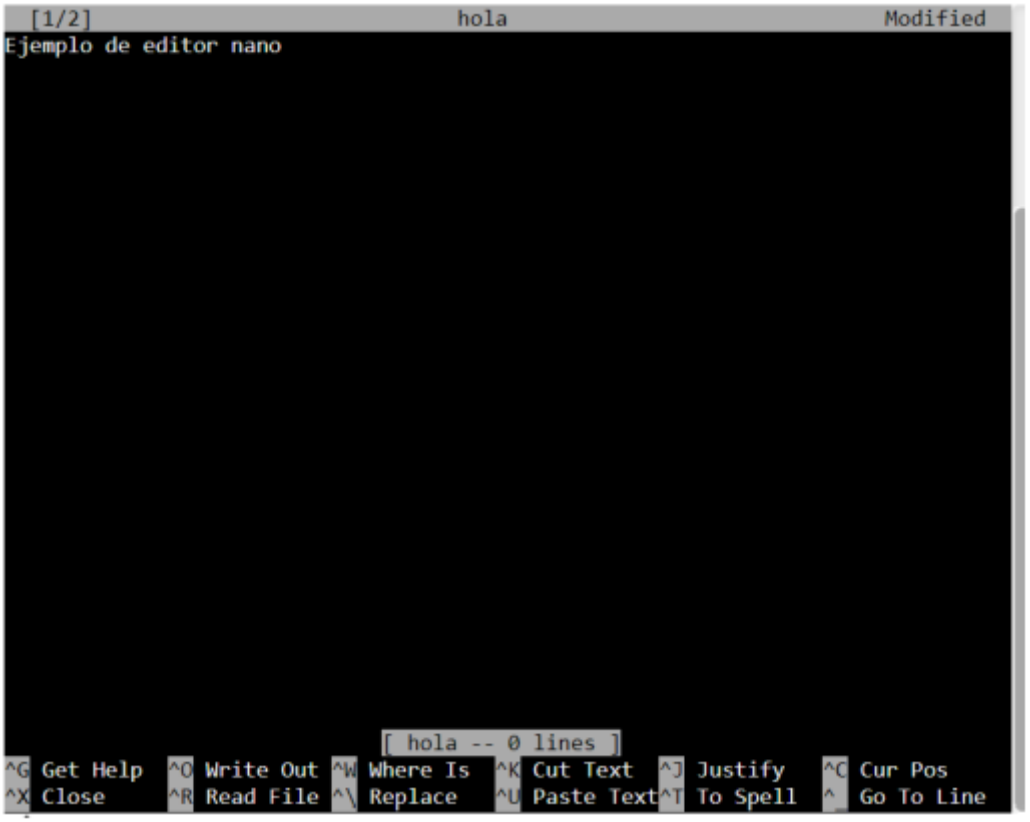
GNU nano

Este editor está disponible para los sistemas operativos que son basados en UNIX en línea de comandos. Es posible acceder a este en un entorno gráfico desde la terminal. Para algunos principiantes dentro de los editores de texto, podría ser que vi se viera un poco complicado, es aquí donde este editor resalta, se trata de un editor más intuitivo que vi, pero cabe aclarar que de igual manera será menos potente que vi.

Para iniciar nano, debe teclearse desde la línea de comandos:

nano nombre_archivo[.ext]

Al abrir este editor, se puede apreciar en la parte inferior algunos de los comandos básicos, al presionar la tecla **F1,** logramos visualizar la ayuda con la lista de todos los comandos en este editor.



Editor de texto nano

Otros

Para editores de texto tenemos muchos para escoger como usuarios, algunas de las opciones no mencionadas anteriormente son:

- Visual Studio Code
- Atom
- Sublime Text
- Notepad
- Etc.

Compiladores

Una vez que tenemos el código escrito, es necesario hacer uso de un compilador para crear un archivo ejecutable y que de esta manera la computadora lo pueda comprender. Un estándar muy común dentro de los compiladores para lenguaje C es **ANSI C**, de este existen diferentes extensiones como **ISOC99 y GNU C**. Cuando nosotros generamos un programa dentro de estos estándares, significa que garantiza se correcta ejecución en cualquier máquina, siempre y cuando exista un compilador para ella.

Si el código fuente contiene errores, el compilador no generará un archivo ejecutable, dirá en qué línea del código se encuentra el error, y solo quedará corregir dicho error en nuestro editor de textos.

Estos son algunos de los compiladores más comunes para el lenguaje C:

gcc (GNU Compiler Collection)
Como su nombre lo indica, este compilador es una colección de compiladores para sistemas operativos basados en UNIX. Al ser una colección de compiladores, contiene varios, entre estos existe el que sirve para programas en lenguaje C. Este compilador viene por defecto en diversas distribuciones de GNU/Linux, pero igual existe una versión modificada que puede ejecutar y crear programas para plataformas Windows en el paquete llamado **MinGW.**

Suponiendo que se tiene un programa escrito en C y se le llamó *calculadora.c,* la manera de compilarlo es localizándose mediante la línea de comandos en la ruta donde el archivo se encuentra y ejecutando el comando:

```
gcc calculadora.c
```

Esto creará un archivo *a.out* (en Windows *a.exe*) que es el programa ejecutable resultado de la compilación.

Si se desea que la salida tenga un nombre en particular, debe definirse por medio del parámetro *-o* de *gcc.* Por ejemplo, para que se llame *calculadora.out* (en Windows *calculadora.exe*), se escribe en la línea de comandos:

```
gcc calculadora.c -o calculadora.out
```

A veces, para realizar un programa más complejo, se necesitan bibliotecas que se instalaron en el equipo previamente y se definió su uso en el programa escrito en C pero al momento de compilar es necesario indicar a GCC que se están usando bibliotecas que no se encuentran en su repertorio de bibliotecas estándar. Para ello es necesario utilizar el parámetro *-l* seguido inmediatamente por el nombre de la biblioteca, sin dejar espacio alguno:

```
gcc calculadora.c -o calculadora -lnombre_biblioteca
```

Entorno de Desarrollo Integrado (IDE por sus siglas en inglés)
Un IDE es una sola aplicación en donde se llevan a cabo todas las etapas de desarrollo de software, son muy utilizados para maximizar la productividad del programador. Estos programas normalmente ofrecen características para la creación, modificación, compilación, implementación y depuración de software.
Algunos de los IDE's más comunes son los siguientes:

Dev-C++ : Un IDE que usa el compilador MinGW, un software libre, sencillo y eficiente, es utilizado en la plataforma Windows.

Code Blocks: Un IDE desarrollado con el lenguaje C++. Es muy popular este los programadores nuevos.

Xcode: El IDE más conocido para programar en el sistema operativo Mac con el compilador gcc e Interface Builder.

Ejecución

Una vez compilado el programa, el siguiente paso naturalmente es ejecutarlo. En el sistema operativo de Windows se puede ejecutar el programa haciendo clic sobre el programa ya compilado, el archivo que genera el compilador, pero es muy recomendable ejecutarlo desde símbolo de sistema.

Considerando que se tiene un programa compilado en un sistema base Unix cuyo nombre es *calculadora.out*, para ejecutar debe teclearse en línea de comandos:

```
./calculadora.out
```

Si el programa realizado necesita tener una entrada de información por medio de argumentos, éstos se colocan así:

```
./calculadora.out argumento1 argumento2
```

Para los programas básicos dentro del lenguaje C, será lógico que sólo funcionarán en modo línea de comandos, para que tenga una interfaz gráfica es necesario de conocimiento avanzado del lenguaje, y del sistema operativo para el que se diseña el programa.

Muchas veces nuestro código no tendrá errores en cuanto al código fuente, pero es posible que algunas veces se ejecute el programa dando un resultado que no es el deseado, es por esto que aún en la fase de ejecución se hagan diversas pruebas para verificar que el programa es correcto.

Desarrollo

Coma.c

Practica6 > C coma.c > ...

```c
#include <stdio.h>

//el operador coma sirve para poner de manera secuencial varias expresiones sin necesidad de usar punto y coma
int main(){

    int x,y,v,z;
    x=(v=3, v*5);
    printf("x=%d, v=%d\n",x,v);

    x=(v+=5, v%3);
    printf("x=%d, v=%d\n",x,v);

    x=(y=(15>10), z=(2>=y), y && z);
    printf("x=%d, y=%d, z=%d\n", x,y,z);

    return 0;
}
```

Captura de pantalla del programa Coma.c

Comentarios

Aquí podemos ver la forma de utilizar el operador coma para las variables y así no tener que hacerlo renglón por renglón, de este modo queda más ordenado y es más entendible. Este operador es muy sencillo de usar, solo consta de colocar una coma después de las variables al momento de identificarlas, por ejemplo cómo se usa en la línea 8, al momento de imprimir algún resultado.

Decremento1.c

Practica6 > C decremento1.c > ...

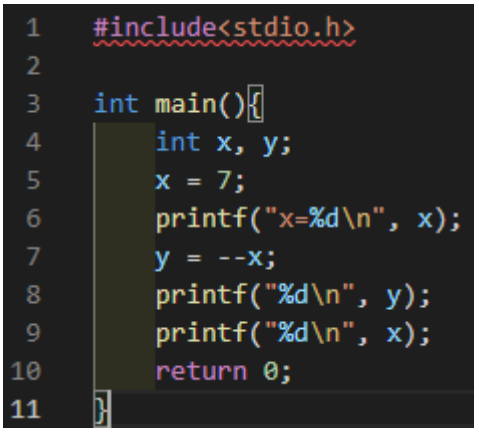
```c
#include<stdio.h>

int main(){

    int x,y;
    x=6;
    printf("x= %d\n", x);


    //este operador es de post decremento, hasta el final es cuando resta
    y=x--;
    printf("y= %d\n", y);
    printf("x= %d\n", x);


    return 0;
}
```

Captura de pantalla del programa Decremento1.c

Comentarios

Aquí utilizamos el decremento de una variable para obtener el valor de una variable, cabe aclarar que este operador es el operador de post decremento, en donde usamos “--” después de la variable, de aquí el nombre identificador de “post”. Lo que éste operador hace es restar 1 a la variable a la que se le aplica.

Decremento2.c



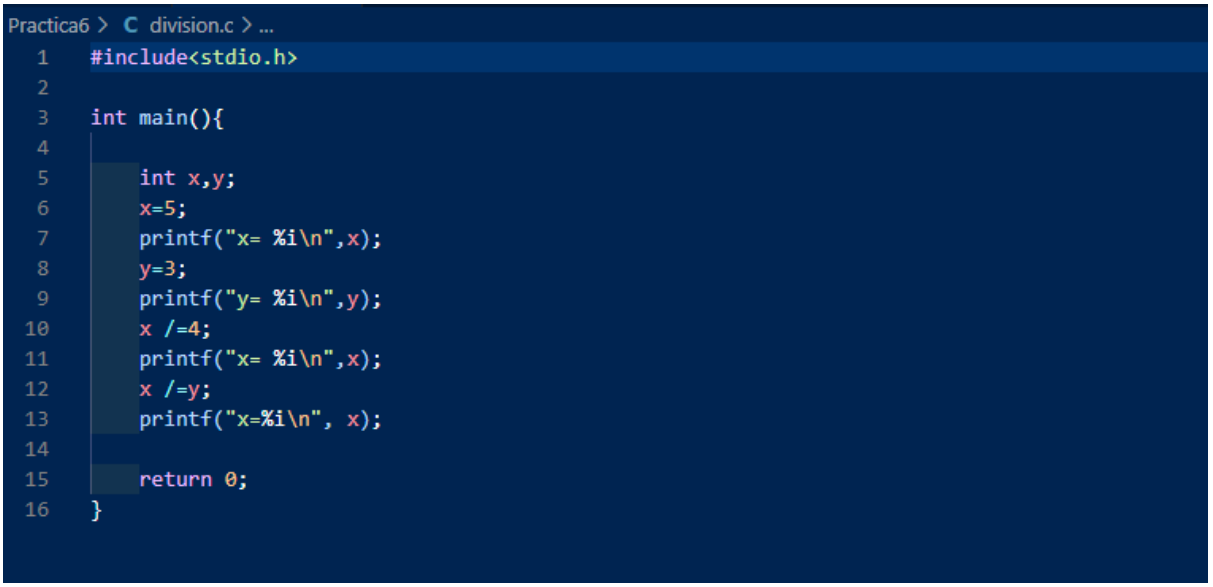
```c
#include<stdio.h>

int main(){
    int x, y;
    x = 7;
    printf("x=%d\n", x);
    y = --x;
    printf("%d\n", y);
    printf("%d\n", x);
    return 0;
}
```

Captura de pantalla del programa Decremento2.c

Comentario

Este programa es pre decremento, el pre decremento permite realizar el decremento antes de cualquier operación, es por eso que se escribe antes de la variable x. Este operador de decremento tiene como nombre más específico “Pre decremento”, ya que los símbolos de menos se encuentran antes de la variable.

división.c



Practica6 > C division.c > ...

```c
#include<stdio.h>

int main(){

    int x,y;
    x=5;
    printf("x= %i\n",x);
    y=3;
    printf("y= %i\n",y);
    x /=4;
    printf("x= %i\n",x);
    x /=y;
    printf("x=%i\n", x);

    return 0;
}
```

Captura de pantalla del programa division.c

Comentarios

Aquí utilizamos el operador / para realizar una división con una variable entera y así obtener el resultado de una división. Ya que en este programa se está trabajando con enteros solamente, algunos de los resultados de las divisiones serán incorrectos, ya que no todas las divisiones darán un entero como resultado.

división 2.c

Practica6 > C division2.c > ...

```c
#include <stdio.h>

int main(){

    float x,y;
    x=25;
    printf("x= %f\n",x);
    y=3;
    printf("y= %f\n",y);
    x /=4;
    printf("x= %f\n",x);
    x /=y;
    printf("x=%f\n", x);

    return 0;
}
```

Captura de pantalla del programa division 2.c

Comentarios

Aquí utilizamos el operador / para realizar una división con una variable flotante y así obtener el resultado de una división. Gracias a que nosotros en este programa estamos trabajando con flotantes, el resultado será una división exacta con los decimales que debe de llevar, a diferencia de la anterior que solo nos podía devolver números enteros.

incremento.c

Practica6 > C incremento.c > ...

```c
#include<stdio.h>

int main(){

    int x,y;
    x=7;
    printf("x= %d\n", x);

    //este operador se llama post incremento, y se incrementa hasta el final
    y=x++;
    printf("y= %d\n", y);
    printf("x= %d\n", x);


    return 0;
}
```

Captura de pantalla del programa incrementó.c

Comentarios

Aquí utilizamos el incremento de una variable para obtener el valor de otra variable. Este operador de incremento en específico se llama “Post incremento”, y se denomina con ‘++’. A diferencia del otro tipo de incremento, los símbolos de más, van después de la variable, es la principal y única diferencia entre el otro incremento.

incremento 2.c

Practica6 > C incremento2.c > ...

```c
#include<stdio.h>

int main(){

    int x,y;
    x=7;
    printf("x= %d\n", x);

    //aqui el operador es de pre incremento, y este suma primero
    y=++x;
    printf("y= %d\n", y);
    printf("x= %d\n", x);


    return 0;
}
```

Captura de pantalla del programa incremento2.c

Comentarios

En este programa se utiliza el operador de incremento de la variable x antes de realizar una suma con el operador =++. A diferencia del otro operador de incremento, en este encontramos que los símbolos '++' se encuentran antes de la variable, se puede observar en la línea de código 10, es por esto que recibe su nombre de "pre incremento". Para finalizar al programa se le pide que al final imprima el valor de la variable "y" y de la variable "x".

lógicos.c

Practica6 > C logicos.c > ...

```c
#include <stdio.h>
#define btoa(x) (x)? "true": "false"


int main(){

    printf("%s\n", btoa(!(7>15)));
    printf("%s\n", btoa(!0));
    printf("%s\n", btoa((35>20) && (20<=23)));
    printf("%s\n", btoa(0 && 1));
    printf("%s\n", btoa((35 > 20) || (20<=18)));
    printf("%s\n", btoa(0 || 1));


    return 0;

}
```

Captura de pantalla del programa lógicos.c

Comentarios

El programa lógicos evalúa los valores falso(false) y verdadero(true), con 7>15 y 0, además se utilizan los operadores AND(&&) en 35>20 y 20<=23 y en 0 y 1. Y el operador OR(||) en 35>20 y 20<=18 y en 0 y 1.

multiplicación.c

Practica6 > C multiplicacion.c > ...

```c
#include<stdio.h>

int main(){

    int x,y;
    x=5;
    printf("x= %d\n",x);
    y=3;
    printf("y= %d\n",y);
    x *=4;
    printf("x= %d\n",x);
    x *=y;
    printf("x=%d\n", x);

    return 0;
}
```

Captura de pantalla del programa multiplicación.c

Comentarios

En este programa se realiza la multiplicación con el operador *= de variables. Lo primero en el programa fue darle valor a las variables, e imprimimos este valor inicial, para después hacer la multiplicación en la línea de código 10 del valor de la variable x por 4, este resultado igualmente lo vamos a imprimir, y para terminar el programa, pedimos que realice la operación de x por la variable y.

Programa2.c

practica6_2 > C programa2.c > main()

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

int main(){

    printf("El tamaño en memoria de un entero es %d bytes\n", sizeof(int));
    printf("El tamaño en memoria de un entero corto es %d bytes\n", sizeof(short));
    printf("El tamaño en memoria de un entero largo es %d bytes\n", sizeof(long));
    printf("El tamaño en memoria de un flotante es %d bytes\n", sizeof(float));
    printf("El tamaño en memoria de un doble precisión es %d bytes\n", sizeof(double));
    printf("El tamaño en memoria de un doble precisión largo es %d bytes\n", sizeof(long double));
    printf("El tamaño en memoria de un doble caracter es %d bytes\n", sizeof(char));
    printf("El tamaño en memoria de un booleano es %d bytes\n", sizeof(bool));
    printf("El tamaño en memoria de un void es %d bytes\n", sizeof(void));

    printf("El tamaño en memoria de un entero con signo es %d bytes\n", sizeof(signed int));
    printf("El tamaño en memoria de un entero sin signo es %d bytes\n", sizeof(unsigned int));

    return 0;
}
```

Captura de pantalla del programa2.c

Comentarios

El programa realizado durante la práctica sirve para medir el tamaño de los diferentes tipos de datos(enteros, flotantes, doble, carácter,booleano y void) en bytes. Para poder utilizar el booleano se hizo uso de <stdbool.h>. El funcionamiento correcto de este programa se basa en la función **sizeof(),** que como su nombre nos indica, nos devolverá el valor de la variable que nosotros coloquemos, es una función muy simple y le podemos dar muchos usos en varios programas.

Programa3.c

practica6_2 > programa3.c > main()

```c
#include <stdio.h>
#include <stdbool.h>
#define btos(x) ((x)?"true":"false")

int main(){
    int ent;
    long largo;
    float flotante;
    double doble;
    long double doblelargo;
    char car;
    signed int Sint;
    unsigned int _uint;
    short corto;
    bool b;

    ent = 14; largo = -2147483648;
    _uint = 65535;
    Sint = 32768;
    flotante = 3.4E+38f;
    doble = 1.7e308;
    doblelargo = 1.7e308;
    car = 127;
    corto = 12;
    b = true;

    printf(" Valor entero: %d\n", ent);
    printf(" Valor entero con signo: %i\n", Sint);
    printf(" Valor entero sin signo: %u\n", _uint);
    printf(" Valor entero sin signo en octal: %o\n", _uint);
    printf(" Valor entero sin signo en hexadecimal: %x\n", _uint);
    printf(" Valor entero; %ld\n", largo);
    printf(" Valor cadena: %s\n", "esto es una cadena");

    printf(" Valor entero: %d\n", corto);
    printf(" Valor entero con signo: %i\n", corto);
    printf(" Valor entero sin signo: %u\n", corto);
    printf(" Valor entero sin signo en octal: %o\n", corto);
    printf(" Valor entero sin signo en hexadecimal: %x\n", corto);

    printf(" Valor flotante: %lf\n", flotante);
    printf(" Valor flotante: %5.5f\n", flotante);
    printf(" Valor flotante: %5.2f\n", flotante);
    printf(" Valor flotante: %ef\n", flotante);
    printf(" Valor flotante: %g\n", flotante);

    printf(" Valor flotante: %lf\n", doble);
    printf(" Valor flotante: %5.5f\n", doble);
    printf(" Valor flotante: %5.2f\n", doble);
    printf(" Valor flotante: %ef\n", doble);
    printf(" Valor flotante: %g\n", doble);

    printf(" Valor flotante: %lf\n", doblelargo);
    printf(" Valor flotante: %5.5f\n", doblelargo);
    printf(" Valor flotante: %5.2f\n", doblelargo);
    printf(" Valor flotante: %ef\n", doblelargo);
    printf(" Valor flotante: %g\n", doblelargo);

    printf("Valor caracter: %c\n", car);
    printf("Valor caracter: %d\n", car);

    printf("Valor bool: %s\n", btos(b));
    printf("Valor bool: %d\n", b);
    return 0;
}
```

Captura de pantalla del programa3.c

Comentarios

Realizamos un programa que lee los valores enteros con signo, sin signo, en octal y en hexadecimal. Puede leer también la cadena, valores flotantes largo y doble largo el valor carácter y el valor bool o lógico. Lo primero que se necesitó hacer en este programa es identificar las variables, son las primeras líneas en el código, después les dimos un valor, este valor no va a cambiar a menos que nosotros lo cambiemos desde el código fuente, después de esto, imprimimos los valores de diferentes variables.

Programa4.c

practica6_2 > C programa4.c > ...

```c
#include <stdio.h>

int main(){
    int x;
    float v;

    x= 4.5 +3;
    v = 4.5 +3;
    printf("El valor de x es: %d\n",x);
    printf("El valor de v es: %f\n",v);

    x= 4.5 -3;
    v = 4.5 -3;
    printf("El valor de x es: %d\n",x);
    printf("El valor de v es: %f\n",v);

    x= 4.5*3;
    v = 4.5*3;
    printf("El valor de x es: %d\n",x);
    printf("El valor de v es: %5.2f\n",v);
    v=4*3;
    printf("El valor de v es: %5.2f\n",v);

    x= 4/3;
    printf("El valor de x es: %d\n",x);
    x=4.0/3.0;
    v = 4/3;
    printf("El valor de x es: %d\n",x);
    printf("El valor de v es: %5.2f\n",v);
    v=4.0/3;
    printf("El valor de v es: %5.2f\n",v);
    v= (float)4/3;
    printf("El valor de v es: %5.2f\n",v);
    v=((float)5+3)/6;
    printf("El valor de v es: %5.2f\n",v);

    x= 15%2;
    v = (15%2)/2;
    printf("El valor de x es: %d\n",x);
    printf("El valor de v es: %5.2f\n",v);
    v=((float)(15%2))/2;
    printf("El valor de v es: %5.2f\n",v);

    return 0;

}
```

Captura de pantalla del programa4.c

Comentarios

Realizamos un programa que pudiera realizar adición, sustracción, multiplicación, división y módulo con valores enteros(int) y flotantes (float). La diferencia radica en que el primero no incluye decimales y el segundo si. Este programa a grandes rasgos es lo que hemos aprendido con los demás programas.

Programa 5.c

```c
#include <stdio.h>

// es para el +=
int main(){
    int x, y;
    x = 6;
    printf("x=%d\n",x);
    y = 4
    printf("x=%d\n",y);
    x += 5; // x = x+5
    printf("x=%d\n",x);
    x += y;
    printf("x=%d\n",x);
    return 0;
}
```

Captura de pantalla del programa5.c

Comentario

En este programa se hace la suma de de dos números enteros x,y, utilizando el operador para suma "+=". Lo primero que se hace en el programa es asignarle un valor a la variable “x” y “y”, teniendo ese valor, llegamos a la línea de código 10, en donde pedimos que al valor asignado de x se le agregue 5 y se imprima el resultado, en la línea de código 13 es muy similar, solo que ahora a la variable x, se le sumará el valor que tenía la variable y, e imprimimos el valor.

Programa 6.c

practica6_2 > C programa6.c > main()

```c
#include <stdio.h>


int main(){
    int x, y;
    x = 10;
    printf("x=%d\n", x);
    y = 5;
    printf("x=%d\n", y);
    x -= 3;
    printf("x=%d\n", x);
    x -= y;
    printf("x=%d\n", x);

    return 0;

}
```

Captura de pantalla del programa6.c

Comentarios

En este programa que realizamos se utiliza el operador -= para poder realizar la resta de x-3 y x-y, tomando en cuenta que a x se le asigna 10 y a y 5, las operaciones que se realizan son las siguientes: 10-3=7 y 7-5 porque ahora x tiene el valor de 7.

Programa8.c

practica6_2 > C programa8.c > ...

```c
#include <stdio.h>

int main(){
    int x, y;
    x = 25;
    printf("x=%i\n", x);
    y = 3;
    printf("y=%i\n", y);
    x /= 4;
    printf("x=%i\n", x);
    x /= y;
    printf("x=%i\n", x);

    return 0;

}
```

Captura de pantalla del programa8.c

Comentarios

En este programa se utiliza el operador /=, el cual hace referencia a una división entre variables, en el programa se puede ver que primero se le asigna un valor a la variable “x” y a la variable “y”, en la línea de código 9, se pide que se divida el valor que se asignó a x sobre 4 y se imprima el resultado, lo mismo pasa en la línea 11, solo que ahora se pide que x sea dividida por la variable y.

Rel.c

```c
#include<stdio.h>
#define btoa(x) (x)? "true": "false"

int main(){
    printf("%s\n", btoa('h'=='p'));
    printf("%s\n", btoa ('a'!='p'));
    printf("%s\n", btoa(7<15));
    printf("%s\n",btoa (22<11));
    printf("%s\n",btoa (15<=2));
    printf("%s\n",btoa (35>=20));
    printf("%s\n",btoa ((7>8) > (9>6)));
    return 0;
}
```

Captura de pantalla del programa rel.c

Comentarios

En este programa utilizamos “btoa”, el cual nos va a servir para identificar si la condición que colocamos después, es verdadera o falsa, por ejemplo en la línea del código 5, lo que se piensa verificar es si el carácter ‘h’ es igual al caracter ‘p’, nosotros sabemos que eso no es verdad, ambos caracteres son diferentes entre sí, entonces lo que nosotros esperamos que el programa devuelva es un “falso”, en caso de la línea de código 10, 35 es mayor que 20 por lo que 35>=20 es verdadera(true). Y en la línea 11, 9>6 es verdadera pero 7>8 no lo es, además de que 7 no es mayor que 9, por lo que 7>9 es falsa(false).

Conclusiones

Conclusión Grupal

En está práctica pasamos a la etapa de codificación y pruebas dentro del ciclo de vida del software, el cual realizamos con el lenguaje de programación en C. Se utilizó el editor de texto y el compilador para hacer declaraciones de variables de datos como (int, char, float, etc). A partir de instrucciones de control de tipo secuencial. Y por último, utilizamos las funciones con diferentes tipos de datos para poder tener la entrada y salida de los datos requeridos en el programa.

Conclusiones individuales

Arredondo Granados Gerardo: Los objetivos en esta práctica fueron realizados correctamente, nosotros como alumnos logramos hacer varios programas en lenguaje C utilizando diferentes funciones, dentro de estos programas fue indispensable la declaración de variables en cada programa. Para dar fin a los programas, en la mayoría de estos, fue necesario imprimir el resultado que se estaba buscando, dentro de este punto fue donde se concluyó otro objetivo de la práctica, que era el mostrar los valores finales de las diferentes variables utilizadas.

Casillas Herrera Leonardo Didier: En la práctica pudimos utilizar un editor como Vim para aprender sus comandos y la forma de utilizarlo en la edición de los programas en C, además realizamos la declaración de distintas variables, les asignamos valores y finalmente obtuvimos la salida de los datos de esas variables, cumpliendo con el objetivo de la práctica y utilizando distintos operadores para ejemplificar su uso.

Diaz Gonzalez Rivas Angel Iñaqui: Durante la realización de la práctica se utilizaron las herramientas Vim y visual studio para poder realizar la asignación de las variables a los diferentes tipos de datos con los que cuenta C. Y se hizo uso de las funciones para que el programa a partir de la entrada estándar (teclado) se pudieran ingresar los datos requeridos para poder presentar en la salida estándar (monitor) los datos necesitados.

Galvan Castro Martha Selene: En conclusión en esta práctica aprendimos un poco más acerca del lenguaje C ya que fuimos aprendimos el funcionamiento de los editores de código fuente, además por medio de la realización de unos ejemplos se fue analizando y comprendiendo su uso de manera práctica ya que a los programas se le fue haciendo pequeñas modificacion para ver qué cambios le ocurrían al funcionamiento de los programas por todo esto se puede decir que se cumplio con el objetivo de practica.

Referencias

- El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.