



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Karina Morales Garcia

Asignatura: Fundamentos de programacion

Grupo: 20

No. de práctica(s): 6

Integrante(s): Avila Pineda Samuel David

No. de lista o brigada: 06

Semestre: 2023-1

Fecha de entrega: 09 de noviembre del 2022

Observaciones:

CALIFICACIÓN: _____

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.

Desarrollo

La etapa de la codificación surge después de que ya se haya analizado el problema, se haya diseñado el algoritmo y que se haya representado de manera gráfica o escrita el algoritmo.

¿Qué es la codificación?

Básicamente es un proceso en el cual se usa lenguajes de programación con el fin de dar instrucciones a una computadora. Esto impulsan a los sitios web el software y las aplicaciones que la gente usa todos los días. Dicha codificación se puede realizar en cualquier lenguaje de programación estructurada.

A continuación, se mostrará el ciclo de vida del software.



Como podemos ver la implementación de un algoritmo se encuentra en la etapa de codificación de un problema.

Entorno de C

El lenguaje C es muy poderoso debido a que combina las características de un lenguaje de alto nivel, con uno de bajo nivel, por lo que se han creado variantes que permiten programar miles de dispositivos electrónicos en el mundo con sus respectivos compiladores.

¿Qué es lo que hace el lenguaje en C?

Se elabora describiendo cada una de las instrucciones de acuerdo con las reglas definidas en este lenguaje en un archivo de texto para después ser procesadas por el compilador.

¿Qué es un compilador?

Es un programa que toma como entrada un archivo de texto y tiene como salida un programa ejecutable, contiene instrucciones que pueden ser procesadas por el hardware.

Algo muy importante es que si un programa se escribe en lenguaje C puede ejecutarse en cualquier maquina siempre y cuando exista un compilador de C, se le conoce como multiplataforma.

Editores

Un programa en C debe ser escrito en un editor de texto esto con el fin de que después se pueda generar un programa ejecutable en la computadora por medio de un compilador.

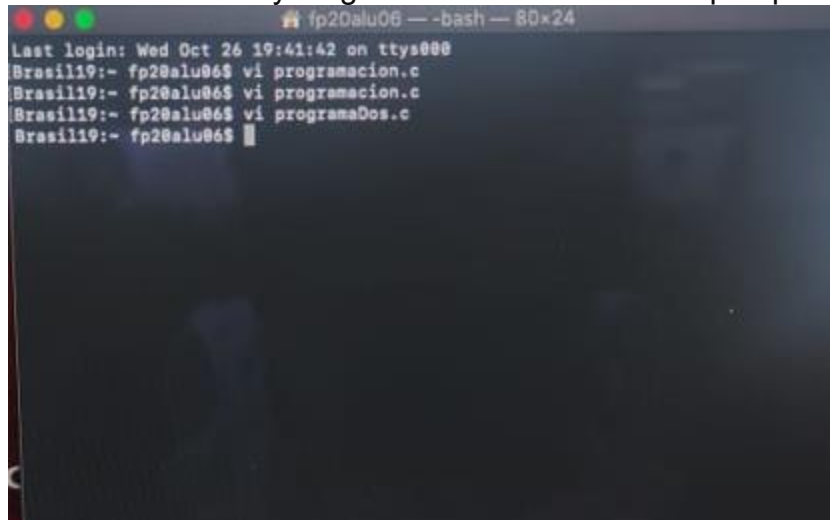
Editor Visual Interface de GNU/Linux (vi)

Es el más común, este puede resultar difícil al inicio.

`vi nombre_archivo[.ext]`

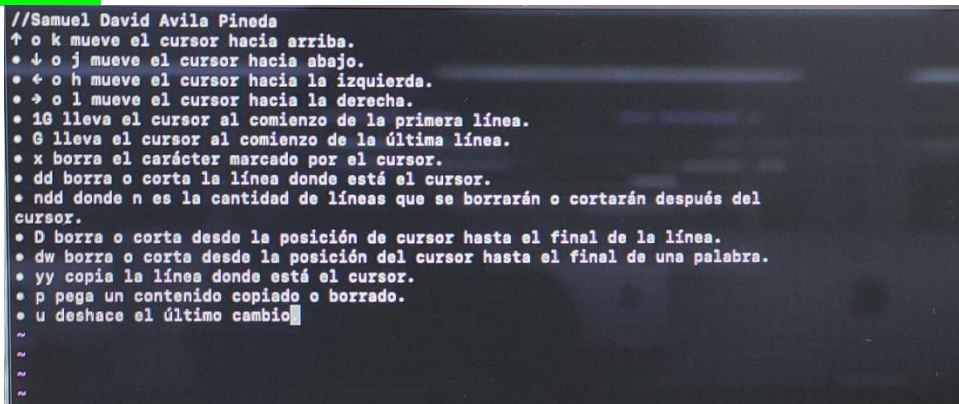
En donde nombre. -archivo es el nombre del archivo a editar o el de un archivo nuevo [.ext] se refiere a la extensión que indica que el texto es un programa escrito en algún lenguaje.

A continuación, se muestra los diferentes archivos que se abrieron. Como se puede observar primero se escribe vi y luego el nombre del archivo que queremos abrir.



```
fp20alu06 -bash - 80x24
Last login: Wed Oct 26 19:41:42 on ttys000
Brasil19:- fp20alu06$ vi programacion.c
Brasil19:- fp20alu06$ vi programacion.c
Brasil19:- fp20alu06$ vi programaDos.c
Brasil19:- fp20alu06$
```

Modo comando



```
//Samuel David Avila Pineda
↑ 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.
GG 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.
n dd 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
```

Primero lo que hicimos fue abrir un archivo llamado programa.c, posteriormente pasamos a editar, colocamos al principio nuestro nombre completo y después colocamos el modo comando que se activa por defecto cuando se abre “vi”. Se fueron haciendo pruebas, para ver si en realidad funcionaba.

Modo última línea

Se puede acceder a el utilizando el modo comando, siempre y cuando se presione la tecla Enter. S puede cancelar el comando utilizando la tecla Esc.

Algunos ejemplos son los siguientes.

/texto donde la cadena texto será buscada hacia delante de donde se encuentra el cursor.

- /texto donde la cadena texto será buscada hacia atrás de donde se encuentra el cursor.

- :q para salir de vi sin haber editado el texto desde la última vez que se guardó.

- :q! para salir de vi sin guardar los cambios.

- :w para guardar los cambios sin salir de vi.

- :w archivo para realizar la orden “guardar como”, siendo archivo el nombre donde se guardará el documento.

- :wq guarda los cambios y sale de vi.

Compiladores.

Una vez que se ha codificado un programa en C en algún editor de texto, este debe ser leído por un programa que produzca un archivo ejecutable. A este programa se le conoce como compilador.

Un programa en C debe de respetar una serie de reglas para que el compilador pueda entenderlas y realizar su función.

Cabe recalcar que es muy común cometer algún error a la hora de elaborar un programa en C y esto ocasionara que no se pueda generar el programa ejecutable y muestre en la línea de comandos de que error se trata y en donde se pudo haber producido. Un mínimo error puede desencadenar muchos otros y al corregir los demás dejen de ser errores.

Cuando el compilador señala un error no cabe más que invocar algún editor de texto, revisar cuidadosamente el programa y corregir.

El compilador del C es el siguiente:

gcc (GNU Compiler Collection)

Este es un conjunto de compiladores de uso libre para sistemas operativos basados en UNIX.

El gcc tiene diferentes opciones de ejecución para usuarios más avanzados.

Suponiendo que se tiene un programa escrito en C y se le llamo calculadora.c

`gcc calculadora.c`

Posteriormente creara un archivo a.out que es el programa ejecutable resultado de la compilación.

Si se desea que la salida tenga un nombre en particular puede definirse como:

`gcc calculadora.c -o calculadora.out`

Comentarios.

Existen dos tipos de comentarios en C: el comentario por línea y el comentario por bloque.

El primero inicia cuando se insertan símbolos // y terminan con el salto de línea, lo que quiere decir es hasta donde termina.

El segundo inicia cuando se insertan los símbolos /* t y termina cuando se encuentran los símbolos */

ProgramaDos.c

```
#include <stdio.h>

int main() {
    // Comentario por línea
    /* Comentario por bloque
       que puede ocupar
       varios renglones */
    // Este código compila y ejecuta
    /* pero no muestra salida alguna
       debido a que un comentario
       ya sea por línea o por bloque */
    // no es tomado en cuenta al momento
    // de compilar el programa,
    /* sólo sirve como documentación en el */
    /*
       código fuente
    */
    return 0;
}
```



Palabras reservadas.

Son las palabras que tienen un significado predefinido estándar y solo se pueden utilizar para su propósito ya establecido. En el lenguaje C son los siguientes:

<i>auto</i>	<i>double</i>	<i>int</i>	<i>struct</i>
<i>break</i>	<i>else</i>	<i>long</i>	<i>switch</i>
<i>case</i>	<i>enum</i>	<i>register</i>	<i>typedef</i>
<i>char</i>	<i>extern</i>	<i>return</i>	<i>union</i>
<i>const</i>	<i>float</i>	<i>short</i>	<i>unsigned</i>
<i>continue</i>	<i>for</i>	<i>signed</i>	<i>void</i>
<i>default</i>	<i>goto</i>	<i>sizeof</i>	<i>volatile</i>
<i>do</i>	<i>if</i>	<i>static</i>	<i>while</i>

Tipos de datos

int - cantidad entera

char - carácter

float - numero en punto flotante

double – numero en punto flotante de doble precisión

Entrada y salida de datos

La mayoría de las versiones de C incluyen una colección de archivos de cabecera que proporcionan la información necesaria para las distintas funciones de la biblioteca.

La biblioteca stdio.h contiene diversas funciones, tanto para imprimir en la salida estándar (monitor) como para leer de la entrada estándar (teclado).

Se pueden escribir datos en el dispositivo de salida estándar, utilizando la función de biblioteca printf.

printf(cadena de control, arg1, arg2, argn)

EJEMPLO:

```
#include <stdio.h>

int main() {
    //Declaración de variables
    int entero;
    float flotante;
    double doble;
    char caracter;
    //Asignación de variables
    entero = 14;
    flotante = 3.5f;
    doble = 6.8e10;
    caracter = 'A';

    //Funciones de salida de datos en pantalla
    printf("La variable entera tiene valor: %i \n", entero);
    printf("La variable flotante tiene valor: %f \n", flotante);
    printf("La variable doble tiene valor: %f \n", doble);
    printf("La variable caracter tiene valor: %c \n", caracter);
    printf("Entero como octal: %o \n Como Hexadecimal %X \n", entero,
           entero);
    printf("Flotante con precisión: %5.2f \n", flotante);
    printf("Doble con precisión: %5.2f \n", doble);
    printf("Carácter como entero: %d \n", caracter);

    return 0;
}
```


Programa3.c

```
fp20alu06 — vi programa3.c — 80x60
#include <stdio.h>
int main() {
    //Declaración de variables
    int entero;
    float flotante;
    double doble;
    char caracter;
    //Asignación de variables
    entero = 14;
    flotante = 3.5f;
    doble = 4.8e10;
    caracter = 'A';
    //Funciones de salida de datos en pantalla
    printf("La variable entera tiene valor: %i \n", entero);
    printf("La variable flotante tiene valor: %f \n", flotante);
    printf("La variable doble tiene valor: %f \n", doble);
    printf("La variable caracter tiene valor: %c \n", caracter);
    printf("Entero como octal: %o \n Como Hexadecimal %X \n", entero,
        entero);
    printf("Flotante con precisión: %5.2f \n", flotante);
    printf("Doble con precisión: %5.2f \n", doble);
    printf("Carácter como entero: %d \n", caracter);
    return 0;
}
```

En la práctica se colocó el programa 3 en una terminal, como lo hicimos anteriormente.

```
fp20alu06 — -bash — 80x60
Brasil19:~$ fp20alu06$ vi programa3.c
Brasil19:~$ fp20alu06$ gcc programa3.c -o programa.out
programa3.c:12:12: error: non-ASCII characters are not allowed outside of
    literals and identifiers
    caracter = 'A';
                  ^
programa3.c:12:14: error: non-ASCII characters are not allowed outside of
    literals and identifiers
    caracter = 'A';
                  ^
programa3.c:12:18: error: use of undeclared identifier 'A'
    caracter = 'A';
                  ^
programa3.c:14:18: error: non-ASCII characters are not allowed outside of
    literals and identifiers
    printf("La variable entera tiene valor: %i \n", entero);
                  ^
programa3.c:14:11: error: use of undeclared identifier 'La'
    printf("La variable entera tiene valor: %i \n", entero);
            ^
programa3.c:14:48: error: non-ASCII characters are not allowed outside of
    literals and identifiers
    printf("La variable entera tiene valor: %i \n", entero);
                  ^
programa3.c:15:18: error: non-ASCII characters are not allowed outside of
    literals and identifiers
    printf("La variable flotante tiene valor: %f \n", flotante);
                  ^
programa3.c:15:11: error: use of undeclared identifier 'La'
    printf("La variable flotante tiene valor: %f \n", flotante);
            ^
programa3.c:15:60: error: non-ASCII characters are not allowed outside of
    literals and identifiers
    printf("La variable flotante tiene valor: %f \n", flotante);
                  ^
programa3.c:16:18: error: non-ASCII characters are not allowed outside of
    literals and identifiers
    printf("La variable doble tiene valor: %f \n", doble);
                  ^
programa3.c:16:11: error: use of undeclared identifier 'La'
    printf("La variable doble tiene valor: %f \n", doble);
            ^
programa3.c:16:47: error: non-ASCII characters are not allowed outside of
    literals and identifiers
    printf("La variable doble tiene valor: %f \n", doble);
                  ^
programa3.c:17:18: error: non-ASCII characters are not allowed outside of
    literals and identifiers
    printf("La variable caracter tiene valor: %c \n", caracter);
                  ^
```

A la hora de compilar, mostro muchos errores, todos eran por el uso de las comillas, las teníamos que cambiar, se hizo el cambio y...

```

Brasil19:~ fp20alu06$ gcc programa3.c -o programa3.out
Brasil19:~ fp20alu06$ ./programa3.out
La variable entera tiene valor: 14
La variable flotante tiene valor: 3.500000
La variable doble tiene valor: 68000000000.000000
La variable caracter tiene valor: A
Entero como octal: 16
Como Hexadecimal E
Flotante con precisión: 3.50
Doble con precisión: 68000000000.00
Carácter como entero: 65
Brasil19:~ fp20alu06$

```

Se volvió a compilar y posteriormente a ejecutar y así quedo.

Para leer datos desde el dispositivo se utiliza la función de biblioteca scanf.

scanf(cadena de control, arg1, arg2, ... , argn)

EJEMPLO:

```

#include <stdio.h>

int main()
{
    int entero;
    float flotante;

    printf("Ingresa el valor entero: ");
    scanf("%i", &entero);
    printf("El valor ingresado es: %d\n", entero);

    printf("Ingresa el valor float: ");
    scanf("%f", &flotante);
    printf("El valor ingresado es: %f\n", flotante);

    return 0;
}

```

Programa4.c

Este ejemplo de programa se debía de hacer igual en la práctica, sin embargo, ya no teníamos tanto tiempo, por lo que la profesora solo nos mostró cómo hacerlo y nosotros debíamos de elaborarlo en la casa.


```
1 #include <stdio.h>
2 int main()
3 {
4     int entero;
5     float flotante;
6     printf("Ingresa el valor entero: ");
7     scanf("%i", &entero);
8     printf("El valor ingresado es: %d\n", entero);
9     printf("Ingresa el valor float: ");
10    scanf("%f", &flotante);
11    printf("El valor ingresado es: %f\n", flotante);
12    return 0;
13 }
14
```

```
main.c:7:11: error: use of undeclared identifier 'i'
scanf("%i", &entero);
      ^
main.c:8:8: error: non-ASCII characters are not allowed outside of
literals and identifiers
printf("El valor ingresado es: %d\n", entero);
      ^
main.c:8:11: error: use of undeclared identifier 'El'
printf("El valor ingresado es: %d\n", entero);
      ^
main.c:8:38: error: non-ASCII characters are not allowed outside
of literals and identifiers
printf("El valor ingresado es: %d\n", entero);
      ^
main.c:9:8: error: non-ASCII characters are not allowed outside of
literals and identifiers
printf("Ingresa el valor float: ");
      ^
main.c:9:11: error: use of undeclared identifier 'Ingresa'
printf("Ingresa el valor float: ");
      ^
main.c:9:35: error: non-ASCII characters are not allowed outside
of literals and identifiers
printf("Ingresa el valor float: ");
      ^
main.c:10:7: error: non-ASCII characters are not allowed outside
of literals and identifiers
scanf("%f", &flotante);
      ^
main.c:10:10: error: expected expression
scanf("%f", &flotante);
      ^
main.c:10:12: error: non-ASCII characters are not allowed outside
of literals and identifiers
scanf("%f", &flotante);
      ^
main.c:10:11: error: use of undeclared identifier 'f'
scanf("%f", &flotante);
      ^
main.c:11:8: error: non-ASCII characters are not allowed outside
of literals and identifiers
printf("El valor ingresado es: %f\n", flotante);
      ^
```

Se puede mostrar que del lado izquierdo esta el programa y a la hora de compilarnos marca diferentes errores, por lo que se tuvo que corregir cada uno de ellos.

```
1 #include <stdio.h>
2 int main()
3 {
4     int entero;
5     float flotante;
6     printf("Ingresa el valor entero: ");
7     scanf("%d", &entero);
8     printf("El valor ingresado es: %d\n", entero);
9     printf("Ingresa el valor float: ");
10    scanf("%f", &flotante);
11    printf("El valor ingresado es: %f\n", flotante);
12    return 0;
13 }
14
```

Ya se puede observar que se corrigieron los errores, en este caso era cambiar todas las comillas y cambiar el %i por %d

```
❯ clang-7 -pthread -lm -o main main.c
❯ ./main
Ingresa el valor entero: 12
El valor ingresado es: 12
Ingresa el valor float: -4
El valor ingresado es: -4.000000
❯
```

Este es el resultado final una vez que ya está compilado y ejecutado.

Tarea:

1.- Investigar cual es el dato que se encuentra por default en lenguaje C (signed o unsigned)

El que se encuentra por default es el Signed.

Signed: indica a la variable que va a llevar signo. Es el utilizado por defecto.

Unsigned: indica a la variable que no va a llevar signo

2.- Indicar que sucede cuando en una variable tipo carácter se emplea el formato %d, %i, %o, %x

Esto no se puede llevar a cabo debido a que es una variable de tipo carácter, si utilizamos %d, %i, %o, %x es para variables con número entero decimal, octal, hexadecimal. Si queremos una variable de tipo carácter se utiliza %c.

3.- Mencionar las características con las que debe crearse una variable

Debe tener 6 conceptos principales.

- El nombre: identificador para referirse a la variable durante el programa.

- La dirección: posición de memoria donde se almacena el dato.

- El tipo: tipo de datos que se almacenará en la variable.

- El alcance: es la sección del programa en la cual se puede usar en expresiones o enunciados.

- Tiempo de vida: segmento de tiempo durante la ejecución del programa durante la cual la variable está presente en la memoria del computador.

4.- ¿Cuál es la diferencia entre variable estática y constante?

Variable estática, quiere decir que no cambia de lugar y la encontraremos en una parte determinada, mientras que la variable constante es la que no cambia su valor

5.- Menciona en que momento empleas los dos tipos de diferentes (< > !=)

"< y >" se emplean a la hora de mencionar una condición en el programa, el primero significa "menor que" y el otro "mayor que"

Ejemplo: $x < y$ lo que quiere decir que x es menor que y

$x > y$ lo que quiere decir que x es mayor que y

"!=" se emplea para mencionar una condición en el programa y quiere decir que "es distinto de"

Ejemplo: $x != y$ lo que quiere decir que x es distinto de y

6.- Crea un programa en el que declares 4 variables haciendo uso de las reglas signed/unsigned, las cuatro variables deben ser solicitadas al usuario (se emplea scanf) y deben mostrarse en pantalla (emplear printf)

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

```
int W, X, Y, Z
```

```
main () {  
    printf ("Ingresa 4 variables");  
    scanf ("%d, %d, %d, %d" &W, &X, &Y, &Z)  
    {  
        printf ("W");}  
    {  
        printf ("X");}  
    {  
        printf ("Y");}
```

```

    {
    printf ("Z");}
    return 0
}

```

7.- Crea un programa que le solicite su edad al usuario, leer el dato (emplear scanf) y mostrarlo en pantalla

```

#include<stdio.h>
int EDAD
main () {
    printf ("¿Cuál es tu edad?");
    scanf ("%d" & EDAD)

    printf ("Tu edad es: %d", EDAD)
    return 0
}

```

8.- Revisar y colocar cuando se emplea MOD y cuando se emplea % (pseudocódigo y codificación) agrega un ejemplo de su uso.

Se utiliza en las expresiones aritméticas para obtener el modulo del primer partido por el segundo. Ejemplo: 8 MOD 5 devolverá 3 ya que ese es el residuo.

9.- Comparación entre Editor de Texto y Procesador de Texto (Realizar una tabla comparativa)

Procesador de texto	Editor de texto
En esta permite crear los documento de texto en una computadora	Se encarga de permitir la escritura y la modificación de un archivo el cual está compuesto por texto.

10.- Indica los comandos utilizados para compilar y para ejecutar un programa en iOS o Linux.ls

Los comandos son los siguientes:

```

gcc nombre del programa.c -o programa.out    - sirve para compilar
./nombre del programa.out                    - sirve para ejecutar

```

11.- Compilación y prueba del programa antes mostrado en DEV C++ u otro IDE en sistema operativo Windows.

Dev-C++: Es un software libre, sencillo y eficiente, para la plataforma Windows. Emplea el compilador MinGW.

12.- Genera un programa que solicite dos variables enteras al usuario y realice las 4 operaciones básicas, compila y ejecuta el programa utilizando terminal y los comandos indicados para cada instrucción.

```

1  #include<stdio.h>
2  main ()
3  {
4      int op;
5      float a,b,res;
6      printf("\n\t favor de seleccionar una opcion");
7      printf("\n\t opcion de suma 1\n");
8      printf("\n\t opcion de resta 2\n");
9      printf("\n\t opcion de multiplicacion 3\n");
10     printf("\n\t opcion de division 4\n");
11     scanf("%d",&op);
12     if(op>0 && op<5)
13     {
14         printf("\n\t teclear el valor de a\t");
15         scanf("%f",&a);
16         printf("\n\t teclear el valor de b\t");
17         scanf("%f",&b);
18     }
19     switch(op)
20     {
21         case1:
22             res=a+b;
23             printf("\n\t la suma es:%f\n",res);
24             break;
25         case2:
26             res=a-b;
27             printf("\n\t la resta es:%f\n",res);
28             break;
29         case3:
30             res=a*b;
31             printf("\n\t la multiplicacion es:%f\n",res);
32             break;
33         case4:
34             if(b!=0)

```

```

21         case1:
22             res=a+b;
23             printf("\n\t la suma es:%f\n",res);
24             break;
25         case2:
26             res=a-b;
27             printf("\n\t la resta es:%f\n",res);
28             break;
29         case3:
30             res=a*b;
31             printf("\n\t la multiplicacion es:%f\n",res);
32             break;
33         case4:
34             if(b!=0)
35             {
36                 res=a/b;
37                 printf("\n\t la division:%f\n");
38             }
39             else
40                 printf("\n\t ERROR!!!");
41             break;
42         default:
43             printf("\n\t OPCION NO VALIDA\n");
44     }
45     return 0;
46 }
47

```

```
2 warnings generated.  
➤ ./main  
  
    favor de seleccionar una opcion  
opcion de suma 1  
  
opcion de resta 2  
  
opcion de multiplicacion 3  
  
opcion de division 4  
1  
  
teclear el valor de a 1  
teclear el valor de b 1  
  
OPCION NO VALIDA  
➤ 
```

A la hora de ejecutar me sale Opción no valida, trate de ver si había errores y al parecer no, no sé por qué no funciona.

Conclusiones

Puedo concluir que esta práctica no estuvo muy complicada, básicamente si sabes cómo hacer un pseudocódigo, podrás programar con el lenguaje c, solo es acordarse de que comandos debemos de utilizar en cada situación y que variables. Se puede llegar a complicar a la hora de compilar y ejecutar, ya que si hay un mínimo error en la redacción del programa, nos puede llegar a dar otra cosa que no queremos, por lo que esto se debe de hacer con calma.

Referencias

Laboratorio Salas A y B. (s.f). <http://lcp02.fi-b.unam.mx/>

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