|  |  |  |
| --- | --- | --- |
|  | **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 Programación |
| *Grupo:* | 1102 |
| *No de Práctica(s):* | Practica número 12 |
| *Integrante(s):* | Gutierrez Orozpe Luis Fernando |
|  |  |
|  |  |
| *Semestre:* | 2018-1 |
| *Fecha de entrega:* | 19 de Noviembre de 2017 |
| *Obervaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

OBJETIVO

Elaborar programas en C donde la solución del problema se divida en funciones. Distinguir lo que es el prototipo o firma de una función y la implementación de ella, así como manipular parámetros tanto en la función principal como en otras.

ACTIVIDADES

Para el buen aprendizaje del tema funciones la practica nos pide una calculadora hecha a base de funciones en la que el usuario pueda elegir cual operación realizar.

**FUNCIONES**

La sintaxis básica para definir una función es la siguiente:

valorRetorno nombre (*parámetros*){

// bloque de código de la función

}

El nombre de la función se refiere al identificador con el cual se ejecutará la función; se debe seguir la notación de camello.

Una función puede recibir parámetros de entrada, los cuales son datos de entrada con los que trabajará la función, dichos parámetros se deben definir dentro de los paréntesis de la función, separados por comas e indicando su tipo de dato, de la siguiente forma:

(tipoDato nom1, tipoDato nom2, tipoDato nom3…)

El tipo de dato puede ser cualquiera de los vistos hasta el momento (entero, real, carácter o arreglo) y el nombre debe seguir la notación de camello. Los parámetros de una función son opcionales.

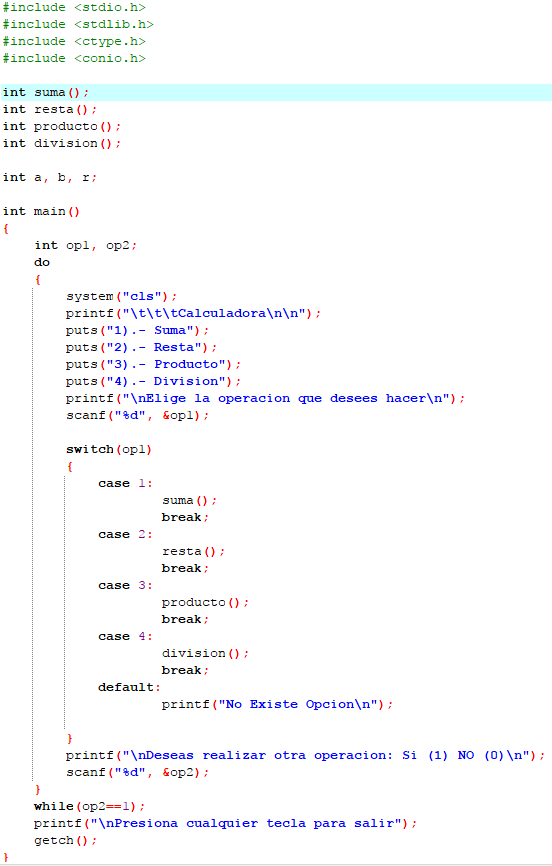
El valor de retorno de una función indica el tipo de dato que va a regresar la función al terminar el bloque de código de la misma. El valor de retorno puede ser cualquiera de los tipos de datos vistos hasta el momento (entero, real, carácter o arreglo), aunque también se puede regresar el elemento vacío (void).

El compilador C revisa que las funciones estén definidas o declaradas antes de ser invocadas. Por lo que una buena práctica es declarar todas las funciones al inicio del programa. Una declaración, prototipo o firma de una función tiene la siguiente sintaxis:

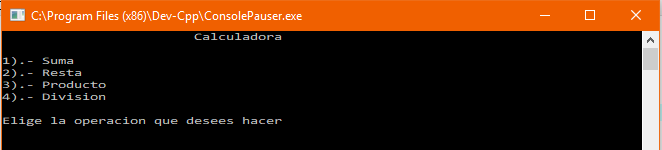
valorRetorno nombre (*parámetros*);

La firma de una función está compuesta por tres elementos: el nombre de la función, los parámetros que recibe la función y el valor de retorno de la función; finaliza con punto y coma (;).

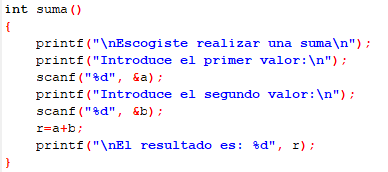
**PROGRAMA: CALCULADORA**



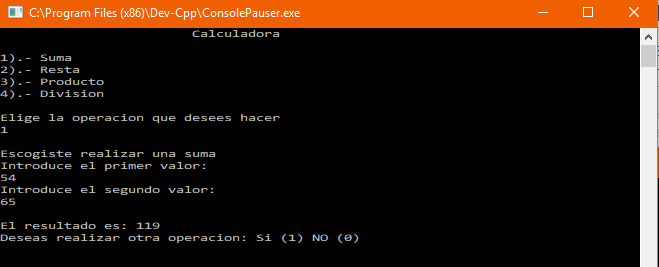
Este programa contiene 4 funciones que fueron declaradas al principio (antes de la función principal). Las 4 funciones declaradas son de tipo de dato “int” (entero) y cada una de ellas es una operación distinta. Solo se incluyen las operaciones básicas: suma, resta, producto y división.



La función principal (main) es la que se encarga de mostrarle al usuario el Menú de las distintas operaciones que puede realizar el programa. Cada operación se ejecutará gracias a su función correspondiente que se explicaran a continuación.

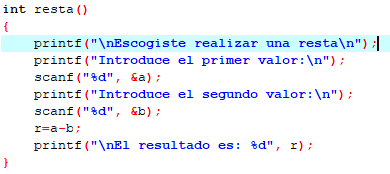


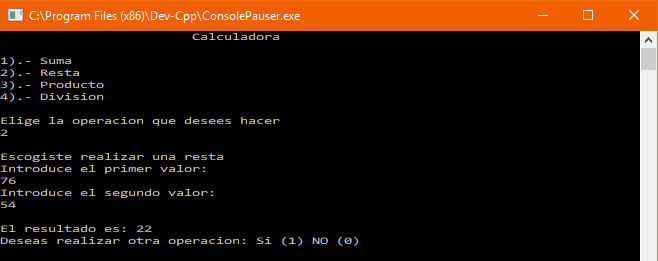
Cuando el usuario selecciona la opción 1 (suma), la función principal entra al “case 1” que en este caso, manda a llamar a la función suma. Una vez llamada la funcion suma, se ejecuta el código de dicha función y cuando las líneas de código de la función terminan, el programa continua con el código de la función principal, en este caso correspondería el “break” del “case 1” que es el que indica que el bloque de código del caso a llegado a su fin.



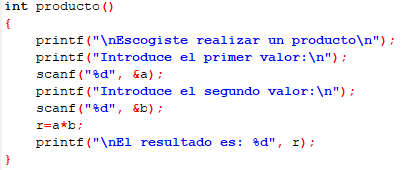
Una vez que se ejecutó la función suma, el programa continua con las líneas de código de la función principal main, la cual se encarga de preguntarle al usuario si desea realizar otra operación. En caso de que el usuario desee realizar otra operación debe presionar el número 1 en su teclado y regresara al menú principal, de lo contrario continuara con las líneas del código y terminara el programa (esto se logra gracias al ciclo Do-While que tiene la función principal)

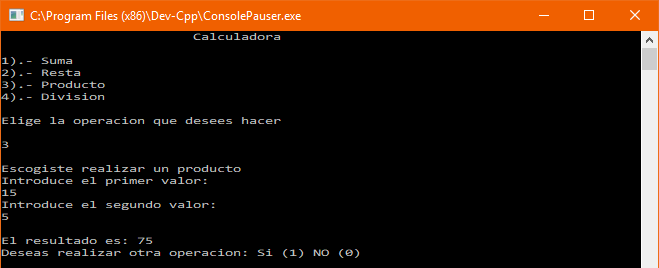
Al igual que en el caso anterior, lo mismo pasa cuando el usuario selecciona la opción 2, 3 o 4. Dependiendo de la opción que éste seleccione, se llamará a la función que corresponda, por lo tanto, si el usuario selecciona la opción 2, se llamara a la función “resta” que está escrita en el “case 2” de la función principal.



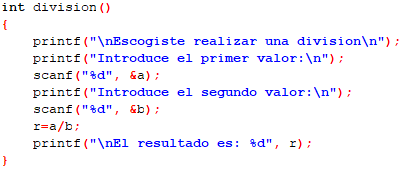


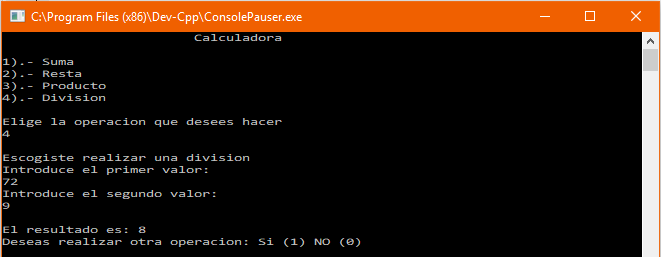
Si se selecciona la opción 3, la función principal manda a llamar a la función “producto” que se encuentra escrita en el “case 3”.





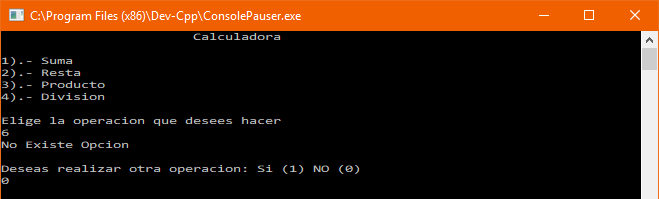
Finalmente, si se selecciona la opción 4, la función principal manda a llamar a la función “división” que se encuentra escrita en el “case 4”.





De esta manera se puede cumplir con el objetivo de la practica al realizar un programa que involucre llamar a diferentes funciones para lograr la ejecución correcta del programa.

Sin embargo, si el usuario por error introduce una opción que el programa no contiene, el programa detectará el error gracias al “default” preguntándole si desea regresar al menú principal, de lo contrario, el programa terminará.



CONCLUSION

Pienso que el tema de funciones fue el más interesante. A mi parecer las funciones dentro del lenguaje de programación en C son muy importantes ya que gracias a ellas podemos hacer programas de una mayor extensión con una mejor organización. Un claro ejemplo de lo antes mencionado es el programa que realizamos en el examen. Debido a que cada función del programa principal demandaba una considerable cantidad de líneas de código, es más practico realizar cada función por separado y declararlas al principio para llamarlas solo cuando el programa así lo requiera. De esta manera se agiliza el trabajo y se le da una mejor estética y organización al código fuente.