**LENGUAJE C**

Una función es un subprograma que devuelve un único valor, un conjunto de valores o realiza alguna tarea específica, tal como entrada / salida. La función principal en un programa escrito en C, es la función “main”.

Todos los programas se construyen a partir de una o mas funciones que se integran para crear una aplicación. El usuario integra una o mas sentencias para realizar una única tarea, como imprimir pantalla, escribir un archivo, cambiar el color de pantalla, etc.

**TIPOS DE DATOS**

Los **tipos** de datos más comunes son:

* ENTEROS: números completos y sus negativos, de tipo INT
* VARIANTES ENTERIOS: tipos SHORT, LONG, INT, UNSIGNED.
* REALES: números decimales, tipos FLOAR, DOUBLE, O LONG DOUBLE.
* CARACTERES: letras, dígitos, símbolos y signos de puntuación, tipo CHAR.
  + Char 🡪 carácter ASCII (solo un carácter)
  + Char[30] 🡪 vector (lo uso como string)

Los **rangos** que cubren los distintos tipos de datos son:

|  |  |
| --- | --- |
| **TIPO** | **RANGO (Min … Max)** |
| Char | 0 … 2 |
| Short | -128 … 127 |
| Int (2 bytes) | -32768 … 32767 |
| Unsigned int | 0 … 65535 |
| Long (8 bytes) | -2147483648 … 2147483637 |
| Float (4 bytes) | 3.4 \* (10 -308) … 3.4 \* (10 308) |
| Double (8 bytes) | 1.7 \* (10 -308) … 1.7 \* (10 308) |
| Long double | Igual que double |

El formato de la **declaración** de datos es: <tipo de dato> <nombre de la variable> = <valor inicial>

Las **expresiones** más comunes son:

+ suma - resta \* multiplicación / división

Z = √X 🡪 Z = SQRT(X)

Z = XY 🡪 Z = POW (X, Y)

Las **máscaras** de entrada y salida son:

%d 🡪 Entero decimal

%o 🡪 Octal

%x 🡪 Hexadecimal

%u 🡪 Entero sin signo

%c 🡪 Carácter

%e 🡪 Float con notación científica

%f 🡪 Float con notación decimal

%g 🡪 Float con notación científica o decimal

%s 🡪 Cadena de caracteres

%lf 🡪 Double

Los **operadores** **lógicos** son:

NOT ! AND && OR ||

Ejemplo función If 🡪If((var1 > var2) && (var3>0))

Los **comparadores lógicos** son:

Z = (var1 > var2) 🡪 mayor

Z = (var1 < var2) 🡪 menor

Z = (var1 > = var2) 🡪 mayor e igual

Z = (var1 > = var2) 🡪 menor e igual

Z = (var1 = = var2) 🡪 igual

Z = (var1 ! = var2) 🡪 distinto

Los operadores especiales aritméticos son:

* a ++ 🡪 a = a + 1
* a -- 🡪 a = a - 1

Las sentencias que se dan dentro de las estructuras de selección y de iteración son:

* if
* if else
* switch
* case
* break
* default
* while
* for

**FUNCIONES**

Los tipos de funciones son:

* Funciones de carácter
* Funciones numéricas
  + Funciones matemáticas
  + Funciones trigonométricas
  + Funciones logarítmicas y exponenciales
* Funciones aleatorias
* Funciones de fecha y hora
* Funciones de utilidad

Las declaraciones globales indican el compilador que las funciones definidas por el usuario son comunes a todas las funciones de su programa. Estas, se sitúan antes de la función main ().

La zona de declaraciones globales de un programa puede incluir declaraciones de variables además de declaraciones de función. Las declaraciones de función se denominan prototipos.

**VARIABLES**

Existen 4 tipos de ámbitos de una variable:

* PROGRAMA 🡪 Conocidas como globales, se las define antes de la función main() y fuera de cualquier función. Esta función será visible para todo el programa y para cualquier función.
* ARCHIVO FUENTE 🡪 procedidas en su declaración por la palabra “static” y se escriben fuera de cualquier función
* FUNCION 🡪 son locales a la función y solo son visibles dentro de ella; fuera NO pueden utilizarse. Solo existen en memoria cuando la función esta activada, por lo que sus nombres pueden no ser únicos.
* BLOQUE 🡪 son solo visibles en él.

Si declaro una variable antes del main(), se considera global, por lo que es común o todas las funciones del programa, cualquier función del programa puede acceder a esa variable.

En cambio, las variables situadas en el interior del cuerpo de la función del main o cualquier otra función se consideran locales y se declaran y definen en la cabecera del bloque o cuerpo de la función. Solo son visibles dentro de ellas, no se las puede utilizar fuera de ellas ya que solo existen en memoria cuando la función esta activada, por lo que puede tener el mismo nombre que una variable declarada global.

La variable descripta en el main() es local a ella y puede ser usada en el transcurso de la función main(); mientras que las variables declaradas en una función, son locales a ella, por lo que si es usada en otra función, esta no la reconocería por no haber sido declarada dentro de ella. Es decir, dos variables pueden llegar a tener el mismo nombre siempre y cuando sean locales a sus funciones distintas ya que cada función se remitirá a su declaración.

**PROGRAMA EN C**

Un programa en C puede incluir:

* Directivas del preprocesador
  + #INCLUDE 🡪 archivos de cabecera (<studio.h>)
  + #DEFINE 🡪 macros del procesador
* Declaraciones globales
  + Prototipos de funciones
  + Variables
* Función principal main (cabecera de función)

*main()*

*{*

*declaraciones locales*

*sentencias*

*}*

* Definiciones de otras funciones

*Tipo función (…)*

*{*

*..*

*}*

* Comentarios del programa

**TIPOS DE ERRORES**

Los errores mas comunes en C son:

* ERRORES DE COMPILACION

Se producen normalmente por uso incorrecto del lenguaje de programación y suelen ser errores de sintaxis

* ERRORES DE EJECUCION

Se producen por instrucciones que la computadora puede comprender, pero no ejecutar

* ERRORES LOGICOS

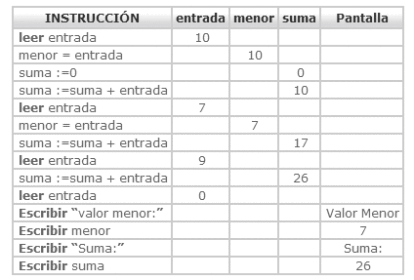
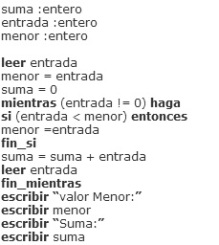
Se producen en la lógica del programa y la fuente de error suele ser el diseño del algoritmo. Son los más difíciles de detectar, ya que solo se advierten por la obtención de datos incorrectos.

**PRUEBAS DE ESCRITORIO**

Es una de las etapas más importantes en el desarrollo de un programa, por cuanto el realizar la prueba de escritorio nos permite saber:

1. Si el programa hace lo que debería hacer
2. Si no hace lo que debería hacer, nos permitirá detectar errores como ser:
   * Si algún paso o instrucción no está en el orden correcto
   * Si falta algo
   * Si algo esta demás
   * Si los pasos o instrucciones que se repiten lo hacen más o menos veces de lo debido
   * Si las instrucciones están en un orden apropiado
   * Otros errores que pueden presentarse
3. Elegir los datos apropiados para la prueba

La prueba de escritorio no es más que efectuar un proceso de simulación con el algoritmo desarrollado (ver que haría la computadora). Este trabajo se realiza en base a una tabla cuyos encabezados son las variables que se usan en el algoritmo y debajo de cada una de ellas se van colocando los valores que van tomando, paso a paso y siguiendo el flujo indicado por el algoritmo, hasta llegar al final.

[](https://brendabalderas.files.wordpress.com/2012/09/2.png)[](https://brendabalderas.files.wordpress.com/2012/09/11.jpg)Por ejemplo:

**CUESTIONARIO**

1. ¿Qué es un algoritmo?
2. ¿Cuáles son las fases de resolución de un problema?
3. Pasos del análisis del problema
4. ¿Qué dice el teorema de la programación estructurada?
5. ¿Qué función no falta nunca en un programa escrito en C?
6. ¿Qué es una función?
7. ¿Qué es una función definida por el usuario?
8. Tipos de datos más comunes en C y cuáles son sus máscaras de entrada y salida
9. Rango que cubren distintos tipos de datos
10. ¿Cómo deben nombrarse a las variables en c?
11. ¿Cuáles son los operadores especiales aritméticos?
12. Tipos de estructuras en lenguaje C
13. Sentencias que se dan en cada una
14. Tipos de funciones
15. Partes a declarar de una función
16. Ámbito de una variable
17. ¿En que difiere si declaro una variable antes del mail, dentro del main o dentro de una función creada por el usuario?
18. ¿La variable descripta en el main es la misma que la descripta dentro de una función?
19. Indique los pasos a desarrollar para escribir y hacer funcionar un programa en C
20. Indique los errores más comunes en C
21. Dentro de un programa, ¿a qué se llama una constante?
22. Dentro de un programa, ¿a qué se llama una variable?
23. ¿Qué es una variable global y local?
24. Duración de una variable
25. Defina los operadores aritméticos, el uso de paréntesis y su asociatividad
26. Defina los operadores de incremento y decremento
27. Defina los operadores relacionales
28. Defina los operadores lógicos
29. Defina los operadores de manejo desplazamiento de bits
30. ¿Qué es una prueba de escritorio?
31. Indique los conversores de tipo de datos