	
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación

Profesor: Saavedraa Hernández Honorato

Asignatura: Fundamentos de programación

Grupo: 01

No de Práctica(s): 12 "Funciones"

Integrante(s): Luis Salinas Ludwig

Semestre: 2018-1

Fecha de entrega: 20/11/2017

Observaciones:

CALIFICACIÓN: _____

salas A y B

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.

Desarrollo

Comenzamos recordando que es una función y donde es que la utilizamos que para este caso se utiliza dentro de los programas. La sintaxis básica para el uso adecuado para definir la función es la siguiente:

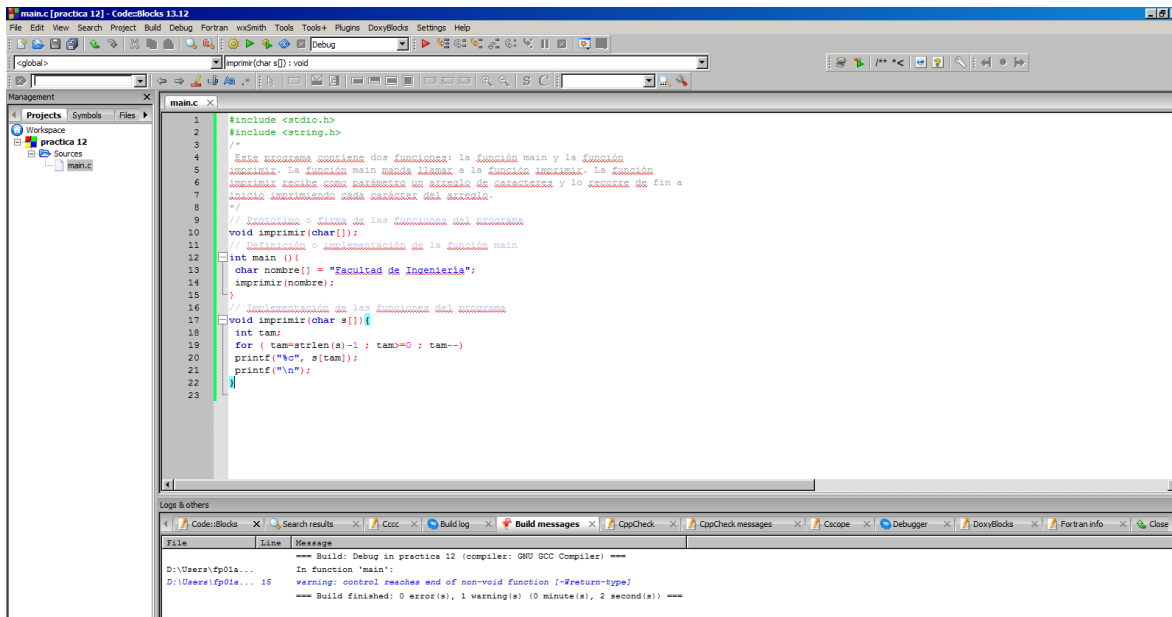
valorRetorno nombre (parámetros)

```
{  
    // bloque de código de la función  
}
```

Seguimos con el Código (funciones)

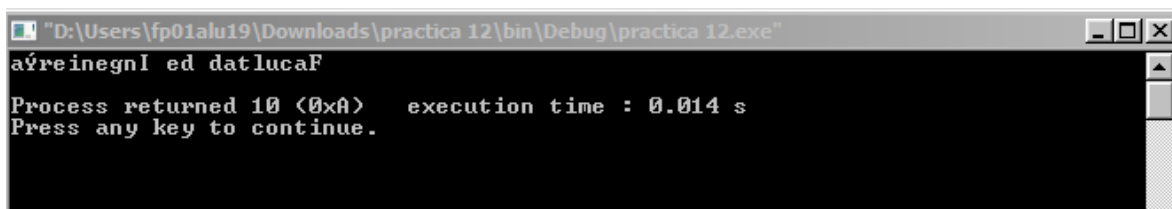
```
#include <stdio.h>  
#include <string.h>  
  
/*  
    Este programa contiene dos funciones: la función main y la función  
    imprimir. La función main manda llamar a la función imprimir. La función  
    imprimir recibe como parámetro un arreglo de caracteres y lo recorre de fin a  
    inicio imprimiendo cada carácter del arreglo.  
*/  
  
// Prototipo o firma de las funciones del programa  
void imprimir(char[]);  
  
// Definición o implementación de la función main  
int main (){  
    char nombre[] = "Facultad de Ingeniería";  
    imprimir(nombre);  
}  
  
// Implementación de las funciones del programa  
void imprimir(char s[]){  
    int tam;  
    for ( tam=strlen(s)-1 ; tam>=0 ; tam-- )  
        printf("%c", s[tam]);  
    printf("\n");  
}
```

En la parte de void imprimir (char[]) nos dice que tenemos una función que más adelante podremos definir por eso es que es un prototipo.



Se es un arreglo y "Facultad de ingeniería" se guarda en s aquí veos que el parámetro nos sirve para recibir información. Como tenemos 22 caracteres C crea un arreglo de 23 caracteres e decir que el arreglo tiene 23 elementos, pero no se imprime el fin de cadena porque es el número 0 que no representa ningún carácter hay que aclarar que esto es solamente en C, y vemos que el primer elemento es 0 y el último elemento es el 22 donde se encuentra un 0 guardado, y en el elemento 21 está guardado una "a", strlen representa el tamaño de un arreglo, y la cadena nombre mide 2 y como nombre lo pasamos a la cadena s ahora s mide 22, y con esto entramos al for que iremos evaluado ahí dentro y ahí tendremos el ciclo repetitivo hasta que ya no se cumpla. Como imprimimos %C nos imprime un carácter de esta manera vemos que dependiendo del tamaño de tam imprimir cierto carácter.

Como resultado nos imprime lo siguiente:



Proseguimos con el Ámbito o alcance de las variables si es local se declara dentro de nuestra función pero si es global esta se declara fuera de las funciones y estas no se pierden, de igual manera se pueden utilizar en cualquier función.

```

void sumar() {
    int x;
    // ámbito de la variable x
}

```

```
#include <stdio.h>

int resultado;

void multiplicar() {
    resultado = 5 * 4;
}
```

Y para poder entenderlo mejor vimos el siguiente ejemplo del Código (Ámbito de las variables)

```
#include <stdio.h>

/*
    Este programa contiene dos funciones: la función main y la función incremento. La
    función main manda llamar a la función incremento dentro de un ciclo for. La función
    incremento aumenta el valor de la variable enteraGlobal cada vez que es invocada.
*/

void incremento();

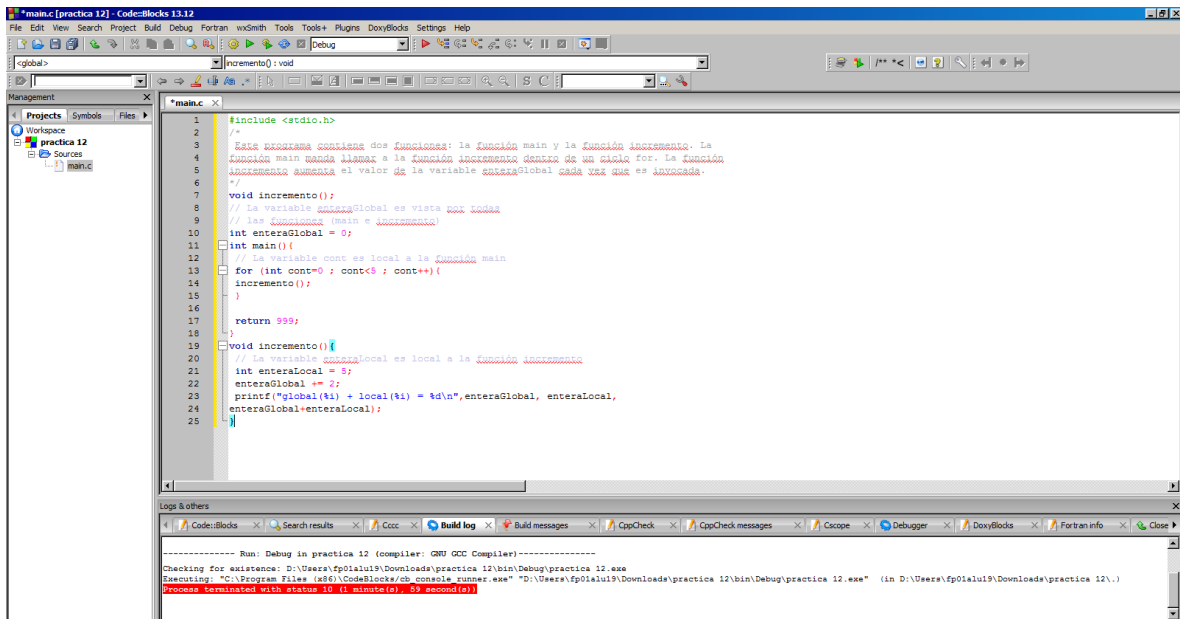
// La variable enteraGlobal es vista por todas
// las funciones (main e incremento)
int enteraGlobal = 0;

int main(){
    // La variable cont es local a la función main
    for (int cont=0 ; cont<5 ; cont++){
        incremento();
    }

    return 999;
}

void incremento(){
    // La variable enteraLocal es local a la función incremento
    int enteraLocal = 5;
    enteraGlobal += 2;
    printf("global(%i) + local(%i) = %d\n", enteraGlobal, enteraLocal,
    enteraGlobal+enteraLocal);
}
```

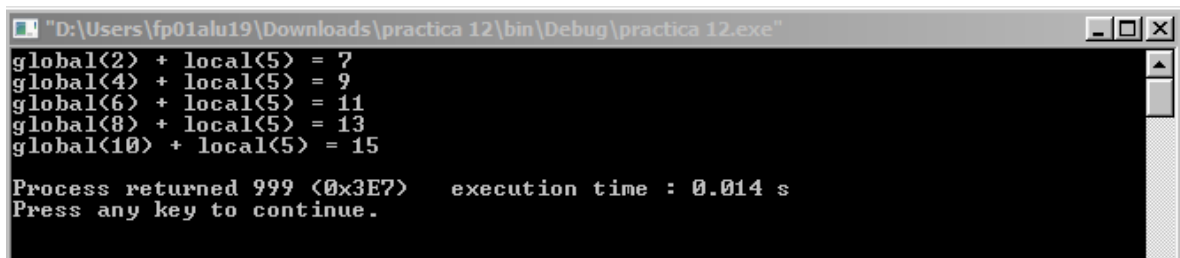
Para enteroGlobal al inicio vale 2 pero después va valiendo de 2 en 2 aquí vemos que no se pierde. En cambio, para enteraLocal ese si se perderá dejar de existir.



```
1 #include <stdio.h>
2 /*
3  Este programa contiene dos funciones: la función main y la función incremento. La
4  función main manda llamar a la función incremento dentro de un ciclo for. La función
5  incremento aumenta el valor de la variable enteraGlobal cada vez que es invocada.
6 */
7 void incremento();
8 // La variable enteraGlobal es vista por todos
9 // las funciones (esta es una variable global)
10 int enteraGlobal = 0;
11
12 int main()
13 // La variable cont es local a la función main
14 for (int cont=0 ; cont<5 ; cont++){
15     incremento();
16 }
17 return 999;
18 }
19
20 void incremento()
21 // La variable enteraLocal es local a la función incremento
22 int enteraLocal = 5;
23 printf("global(%i) + local(%i) = %d\n", enteraGlobal, enteraLocal,
24        enteraGlobal+enteraLocal);
25 }
```

Logs & others

```
----- Run: Debug in practica 12 (compiler: GNU GCC Compiler) -----
Checking for existence: D:\Users\fp01alu19\Downloads\practica 12\bin\Debug\practica 12.exe
Executing: "C:\Program Files (x86)\CodeBlocks\cb_console_runner.exe" "D:\Users\fp01alu19\Downloads\practica 12\bin\Debug\practica 12.exe" (in D:\Users\fp01alu19\Downloads\practica 12\.)
Process terminated with status 10 (1 minute(s), 59 second(s))
```



```
"D:\Users\fp01alu19\Downloads\practica 12\bin\Debug\practica 12.exe"
global(2) + local(5) = 7
global(4) + local(5) = 9
global(6) + local(5) = 11
global(8) + local(5) = 13
global(10) + local(5) = 15

Process returned 999 (0x3E7)   execution time : 0.014 s
Press any key to continue.
```

Continuamos con Argumentos para la función main.

Se vio como ejemplo:

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

/*
 Este programa permite manejar los argumentos enviados al ejecutarlo.
 */

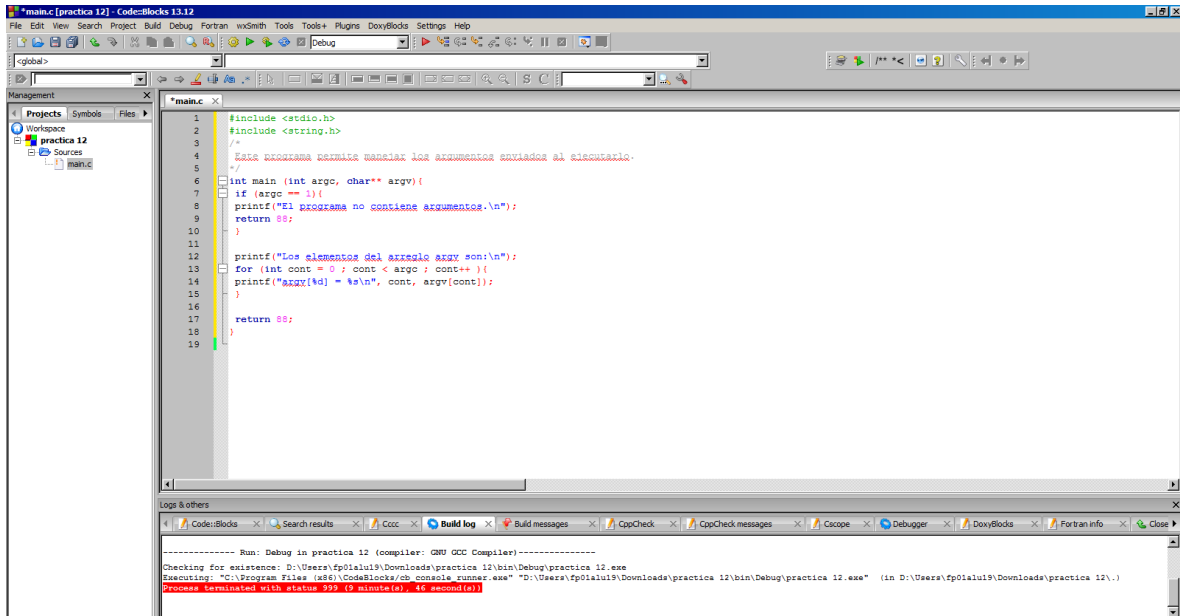
int main (int argc, char** argv){
    if (argc == 1){
        printf("El programa no contiene argumentos.\n");
        return 88;
    }

    printf("Los elementos del arreglo argv son:\n");
    for (int cont = 0 ; cont < argc ; cont++){
        printf("argv[%d] = %s\n", cont, argv[cont]);
    }

    return 88;
}
```

Al final crea un arreglo de arreglos es decir un arreglo de cadenas y en cada cadena vendrá el argumento que se ingresó. Hay que decir que el nombre del programa ahora será el primer parámetro.

Hay dos cadenas la primera será el nombre del programa y medirá dependiendo del nombre del programa por los parámetros, mientras que la segunda será `-s` que medirá 2



```
1 #include <stdio.h>
2 #include <string.h>
3
4 /* Este programa permite manejar los argumentos enviados al ejecutable. */
5
6 int main (int argc, char** argv) {
7     if (argc == 1) {
8         printf("El programa no contiene argumentos.\n");
9         return 0;
10    }
11
12    printf("Los elementos del arreglo argv son:\n");
13    for (int cont = 0 ; cont < argc ; cont++) {
14        printf("argv[%d] = %s\n", cont, argv[cont]);
15    }
16
17    return 0;
18 }
```

Log & others

Run: Debug in practica 12 (compiler: GNU GCC Compiler)

Checking for existence: D:\Users\fp01alul9\Downloads\practica 12\bin\Debug\practica 12.exe

Executing: "C:\Program Files (x86)\CodeBlocks\cb_console_runner.exe" "D:\Users\fp01alul9\Downloads\practica 12\bin\Debug\practica 12.exe" (in D:\Users\fp01alul9\Downloads\practica 12\.)

Process terminated with status 0 (1 minute 17.46 seconds)

Como estamos en codeBlocks no sabremos los parámetros por ello lo correremos en una terminal.

Conclusiones

Las funciones son de gran utilidad para nuestro programa y hay diferencias en las variables cuando las declaramos dentro de una función o fuera de ella directamente en el programa, aquí es donde vemos la gran diferencia ya que la variable global afecta a todas las funciones si es que son utilizadas en las funciones, en cambio en las variables locales solo afecta a la función en la que fue declarada y utilizada. Gracias a las practicas anteriores donde se utilizaron las funciones en conjunto con esta práctica quedo más claro cuál es la manera en que trabaja una función dentro de nuestro programa.