



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorios de
docencia

Laboratorio de Computación

Salas A y B

Profesor(a): César Fabián Domínguez Velasco

Asignatura: Fundamentos de Programación

Grupo: 15

No de Práctica(s): 11

Integrante(s): Camacho Duarte Héctor Enrique

Gutiérrez Esquivel Giovani Emiliano

Flores Jiménez Diego

Moreno Chapan Amilet

Reyes García Raúl de Jesús

No. de lista 07,11,15,16,25,35.

o brigada:

Semestre: 2024-02

Fecha de entrega: 23/04/24

Observaciones:

CALIFICACIÓN: _____

Introducción

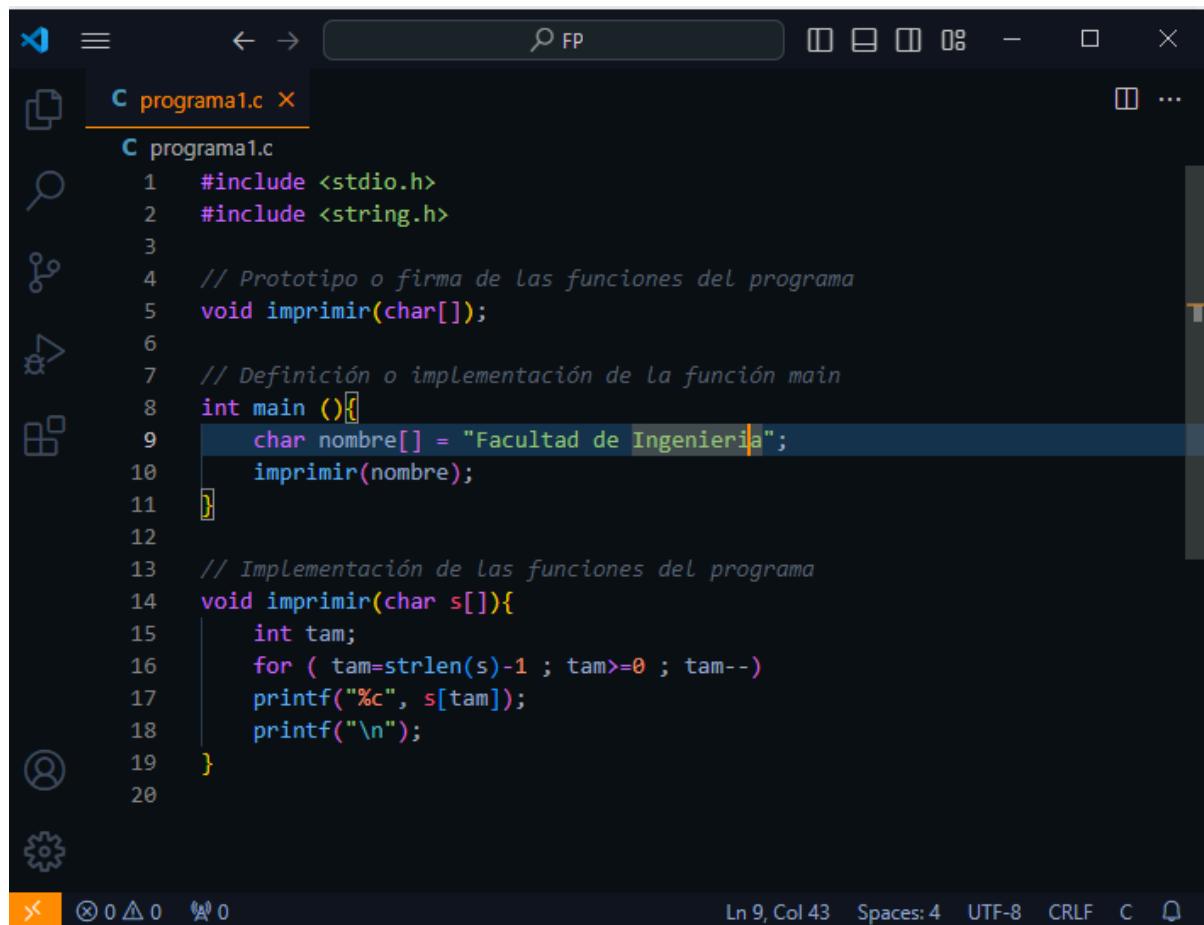
En esta práctica, nos enfocaremos en el uso y la implementación de funciones en C. Exploraremos cómo definir y llamar funciones, pasárselas argumentos, y retornar valores. También analizaremos la importancia de las funciones para la modularidad y reutilización del código, así como para facilitar la depuración y el mantenimiento de programas más complejos.

A través de ejercicios prácticos, aprenderemos a crear funciones personalizadas y a integrarlas en nuestros programas, comprendiendo mejor cómo estas contribuyen a un desarrollo más eficiente y estructurado en el lenguaje C.

Objetivos

El alumno elaborará programas en C donde la solución del problema se divide 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.

Programa 1.c



```
C programa1.c
C programa1.c
1 #include <stdio.h>
2 #include <string.h>
3
4 // Prototipo o firma de las funciones del programa
5 void imprimir(char[]);
6
7 // Definición o implementación de la función main
8 int main (){
9     char nombre[] = "Facultad de Ingeniería";
10    imprimir(nombre);
11 }
12
13 // Implementación de las funciones del programa
14 void imprimir(char s[]){
15     int tam;
16     for ( tam=strlen(s)-1 ; tam>=0 ; tam-- )
17         printf("%c", s[tam]);
18     printf("\n");
19 }
```

The screenshot shows a code editor window with a dark theme. On the left is a sidebar with various icons: file, search, file tree, refresh, and settings. The main area displays a C program named 'programa1.c'. The code includes standard library includes, a prototype for the 'imprimir' function, the 'main' function which calls 'imprimir' with a string literal, and the implementation of 'imprimir' which prints the string character by character. The code editor interface includes a top bar with file operations and status indicators at the bottom.

```
AMILET@DESKTOP-T56HH6F ~
$ cd /cygdrive/c/fp/
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ gcc programa1.c -o programa1.exe
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ ./programa1
Facultad de ingeniería
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ |
$ rm programa1.exe
```

En este programa en c, a través de tres funciones una de tipo int y dos tipo void con apoyo de la biblioteca stdio.h y string.h manda llamar a la función imprimir con la función “main”.

La función imprimir recibe como parámetro un arreglo de caracteres y lo recorre de fin a inicio imprimiendo cada carácter del arreglo en este caso “ Facultad de ingeniería” pero de forma inverso, al inicio nos mandaba error pero se arreglo usando una firma de función.

Programa 2.c

```
#include <stdio.h>
void sumar(){
    int x=5, y=10, z; //variables locales
    z=x+y;
    printf("%i", z);
}
int main (){
    sumar(); //Llamado de la función suma
}
```

```
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ gcc programa2.c -o programa2.exe
programa2.c: In function 'main':
programa2.c:4:5: warning: implicit declaration of function 'sumar' [-Wimplicit-function-declaration]
  4 |     sumar(); //llamado de la funcion suma
   |     ^
programa2.c: At top level:
programa2.c:7:6: warning: conflicting types for 'sumar'; have 'void()'
  7 | void sumar(){
   |     ^
programa2.c:4:5: note: previous implicit declaration of 'sumar' with type 'void()'
  4 |     sumar(); //llamado de la funcion suma
   |     ^
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ gcc programa2.c -o programa2.exe

AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ ./programa2
15
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$
```

Se aplican declaraciones con el uso de variables locales, así también se puede identificar que hubo funciones de variables globales.

Programa3.c

```
#include <stdio.h>
int resultado; // variable global
void multiplicar();

int main (){
    multiplicar(); // Llamado de la funcion
    printf("%i", resultado);
}

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

```
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ gcc programa3.c -o programa3.exe
programa3.c: In function 'multiplicar':
programa3.c:13:12: warning: 'return' with a value, in function returning void
  13 |     return resultado;
      |     ^
programa3.c:11:6: note: declared here
  11 | void multiplicar() { // funcion multiplicar
      |     ^
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ gcc programa3.c -o programa3.exe

AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ ./programa3
20
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ gcc programa3.c -o programa3.exe

AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ ./programa3
20
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ |
```

Se hace el uso de comandos como int main, void.

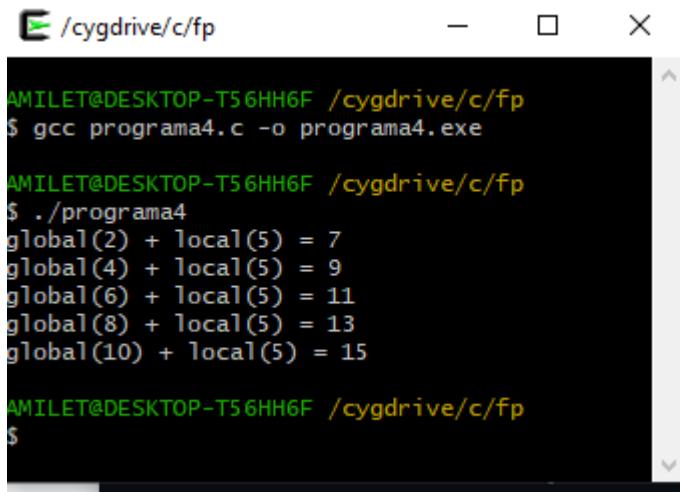
Se utilizaron declaraciones de variables globales resultantes.

Programa4.c

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, ..., back arrow, forward arrow, search icon, file type dropdown (set to C), save icon, close icon.
- Left Sidebar:** Icons for file operations (New, Open, Save, Find, Settings, Help).
- Tab Bar:** programa1.c, programa2.c, programa3.C, **programa4.c** (highlighted in orange), Settings.
- Code Area:** The code for **programa4.c** is displayed. It includes a global variable `enteraGlobal` and a function `incremento()`. The variable is initialized to 0 and increased by 2 in each iteration of the loop. The final output is printed as "global(0) + local(5) = 10".

```
#include <stdio.h>
void incremento();
int enteraGlobal;
int main (){
    int cont;
    enteraGlobal = 0;
    for(cont=0 ; cont<5; cont++){
        incremento();
    }
    return 999;
}
void incremento(){
    int enteraLocal = 5;
    enteraGlobal+=2;
    printf("global(%i) + local(%i) = %d\n", enteraGlobal, enteraLocal, enteraGlobal+enteraLocal);
}
```



```
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ gcc programa4.c -o programa4.exe

AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ ./programa4
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

AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$
```

Se aplican dos funciones como main, incrementó, la función main manda a llamar al incremento en un ciclo for.

Programa5:c

```
C programa5.c
1 #include <stdio.h>
2 #include <string.h>
3
4 int main (int argc, char** argv){
5     if (argc == 1 ){
6         printf("El programa no contiene argumentos.\n");
7         return 88;
8     }
9     printf("Los elementos del arreglo argv son:\n");
10    for(int cont=0; cont< argc; cont++){
11        printf("argv[%d] = %s\n", cont, argv[cont]);
12    }
13 }
14
```

```
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ gcc programa5.c -o programa5.exe

AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ ./programa5
El programa no contiene argumentos.

AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ |
```

La función main puede también recibir parámetros.

Se utilizan comandos con los que se pueden iniciar parámetros, main, char,Argv.

Programa6.c

```
C programa5.c X C programa6.c
C programa5.c
1 #include <stdio.h>
2 #include <string.h>
3
4 int main (int argc, char** argv){
5     if (argc == 1 ){
6         printf("El programa no contiene argumentos.\n");
7         return 88;
8     }
9     printf("Los elementos del arreglo argv son:\n");
10    for(int cont=0; cont< argc; cont++){
11        printf("argv[%d] = %s\n", cont, argv[cont]);
12    }
13    return 88;
14 }
```

```
AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ gcc programa6.c -o programa6.exe

AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$ ./programa6
Esta función se ha llamado 1 veces.
Esta función se ha llamado 2 veces.
Esta función se ha llamado 3 veces.
Esta función se ha llamado 4 veces.
Esta función se ha llamado 5 veces.

AMILET@DESKTOP-T56HH6F /cygdrive/c/fp
$
```

Fread permite leer uno o varios elementos de la misma longitud a partir de una dirección

de memoria determinada (apuntador)

El código fread muestra un contenido de texto y el nombre de este es considerado como el argumento.

Programa7.c

The screenshot shows a code editor interface with a dark theme. On the left is an Explorer sidebar listing various C files and other documents. The main area displays the content of the 'estatica.c' file. The code defines four functions: suma, resta, producto, and cociente, all operating on integers.

```
estatica.c — Documents
estatica.c x calculadora.c

C estatica.c
1 #include <stdio.h>
2 int suma(int,int);
3 static int resta(int,int);
4
5 int producto(int,int);
6 static int cociente (int,int);
7
8 int suma (int a, int b)
9 {
10     return a + b;
11 }
12 static int resta (int a, int b)
13 {
14     return a - b;
15 }
16 int producto (int a, int b)
17 {
18     return (int)(a*b);
19 }
20 static int cociente (int a, int b)
21 {
22     return (int)(a/b);
23 }
```

```
C calculadora.c
1 #include <stdio.h>
2 int suma(int,int);
3 //static int resta(int,int);
4 int producto(int,int);
5 //static int cociente (int,int);
6 int main()
7 {
8     printf("5 + 7 = %i\n",suma(5,7));
9     //printf("9 - 77 = %d\n",resta(9,77));
10    printf("6 * 8 = %i\n",producto(6,8));
11    //printf("7 / 2 = %d\n",cociente(7,2));
12 }
```

Código `fwrite` realiza una copia exacta de dos archivos, el origen y destino de reciben como el argumento de la función principal.

Conclusión

En conclusión, la práctica sobre funciones en el lenguaje C nos ha permitido comprender y aplicar uno de los conceptos fundamentales de la programación estructurada. Hemos aprendido cómo definir, declarar y llamar funciones, así como la manera en que estas pueden interactuar entre sí para realizar tareas complejas de manera más manejable y modular.

A través de los ejercicios realizados, pudimos observar las ventajas de dividir un programa en funciones más pequeñas y especializadas, como la firma de una función que es una herramienta útil, o definir elemento estáticos, incluso aprendimos a compilar dos programas en gcc siempre y cuando estos programas tengan una relación.

Bibliografía

- Laboratorio Salas A y B. (n.d.). <http://lcp02.fi-b.unam.mx/>
- El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, Segunda edición, USA, Pearson Educación 1991.