



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: MC. Alejandro Esteban Pimentel Alarcón

Asignatura: Fundamentos de Programación

Grupo: 3

No de Práctica(s): Práctica 7

Integrante(s): Martínez Marcelino Dalila

*No. de Equipo de
cómputo empleado:* No. de cuenta: 313080119

No. de Lista o Brigada:

Semestre: 2020-1

Fecha de entrega: Lunes 30 de septiembre de 2019

Observaciones: Excelente

CALIFICACIÓN: 10

Práctica No. 7

Introducción:

En esta práctica el estudiante conocerá los diferentes tipos de variable que se maneja para el Lenguaje C. Después de ello se comenzará a desarrollar programas en Lenguaje C que, si bien su lenguaje es parecido a los pseudocódigos, ahora el lenguaje incluirá la utilización de instrucciones y la declaración de variables, los cuales serán aún más específicos que en los pseudocódigos de las practicas pasadas. Estos programas serán desarrollados en algunos de los editores de texto que se conocieron y utilizaron la práctica pasada.

Objetivo:

Elaborar 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.

Actividad 1.

Conociendo las expresiones para los diferentes tipos de variables. Vemos que ahora se le asigna un rango para la memoria que se va a utilizar dentro del programa.

| DATA TYPE | MEMORY (BYTES) | RANGE |
|------------------------|----------------|------------------------------------|
| short int | 2 | -32,768 to 32,767 |
| unsigned short int | 2 | 0 to 65,535 |
| unsigned int | 4 | 0 to 4,294,967,295 |
| int | 4 | -2,147,483,648 to 2,147,483,647 |
| long int | 4 | -2,147,483,648 to 2,147,483,647 |
| unsigned long int | 4 | 0 to 4,294,967,295 |
| long long int | 8 | $-(2^{63})$ to $(2^{63})-1$ |
| unsigned long long int | 8 | 0 to 18,446,744,073,709,551,615 |

Para los reales, también se tienen diferentes tipos de variables que asignas más bits para tener mayor rango y mayor precisión. Las variables reales siempre poseen signo.

| <i>Tipo</i> | <i>Bits</i> | <i>Valor Mínimo</i> | <i>Valor Máximo</i> |
|--------------------|-------------|-------------------------|-------------------------|
| <i>float</i> | 32 | 3.4 E-38 | 3.4 E38 |
| <i>double</i> | 64 | 1.7 E-308 | 1.7 E308 |
| <i>long double</i> | 80 | 3.4 E-4932 | 3.4 E4932 |

| <i>Tipo de dato</i> | <i>Especificador de formato</i> |
|-----------------------------|---------------------------------|
| <i>Entero</i> | %d, %i, %ld, %li, %o, %x |
| <i>Flotante</i> | %f, %lf, %e, %g |
| <i>Carácter</i> | %c, %d, %i, %o, %x |
| <i>Cadena de caracteres</i> | %s |

Se sigue haciendo uso de los operadores lógicos, a través de su simbología adecuada.

| <i>Operador</i> | <i>Operación</i> | <i>Uso</i> | <i>Resultado</i> |
|-----------------|------------------|---------------|------------------|
| + | Suma | 125.78 + 62.5 | 188.28 |
| - | Resta | 65.3 - 32.33 | 32.97 |
| * | Multiplicación | 8.27 * 7 | 57.75 |
| / | División | 15 / 4 | 3.75 |
| % | Módulo | 4 % 2 | 0 |

| <i>Operador</i> | <i>Operación</i> | <i>Uso</i> | <i>Resultado</i> |
|-----------------|------------------|------------|------------------|
| == | Igual que | 'h' == 'H' | Falso |
| != | Diferente a | 'a' != 'b' | Verdadero |
| < | Menor que | 7 < 15 | Verdadero |
| > | Mayor que | 11 > 22 | Falso |
| <= | Menor o igual | 15 <= 22 | Verdadero |
| >= | Mayor o igual | 20 >= 35 | Falso |

| <i>Operador</i> | <i>Operación</i> |
|-----------------|------------------|
| ! | No |
| && | Y |
| | O |

```

#include <stdio.h>

int main() {
    //Declaramos variables a leer
    //Esto es solo un comentario que no aparecera en el programa
    int numeroEntrada;
    double realEntrada;

    //Asignamos variables
    int numeroEntero = 32768;
    char caracter = '0';
    float numeroReal = 89.8;

    //Mostramos texto y valores
    printf("Primero texto solo\n");
    printf("Luego podemos poner un entero: %i\n", numeroEntero);
    printf("Tambien podemos poner un caracter: %c\n", caracter);
    printf("Y un numero real: %.2f\n", numeroReal);

    // Leemos valores
    scanf("%i", &numeroEntrada);
    scanf("%lf", &realEntrada);

    //Y ahora podemos mostrarlos tambien
    printf("Tu entero: %i\n", numeroEntrada);
    printf("Tu real: %.3lf\n", realEntrada);

    return 0;
}

```

Y su respectiva compilación, si el desarrollo del programa fue correcto, entonces saldrá esto:



```
DELL@DESKTOP-N3V5QTU ~/progra
$ ls
ejemplo1.exe  ejemplo2.exe  ejemplo3.exe  prueba.c  vim-tutor.txt
ejemplo2.c    ejemplo3.c    primerprog.c  prueba.c~
ejemplo2.c~   ejemplo3.c~   primerprog.c~ prueba.exe

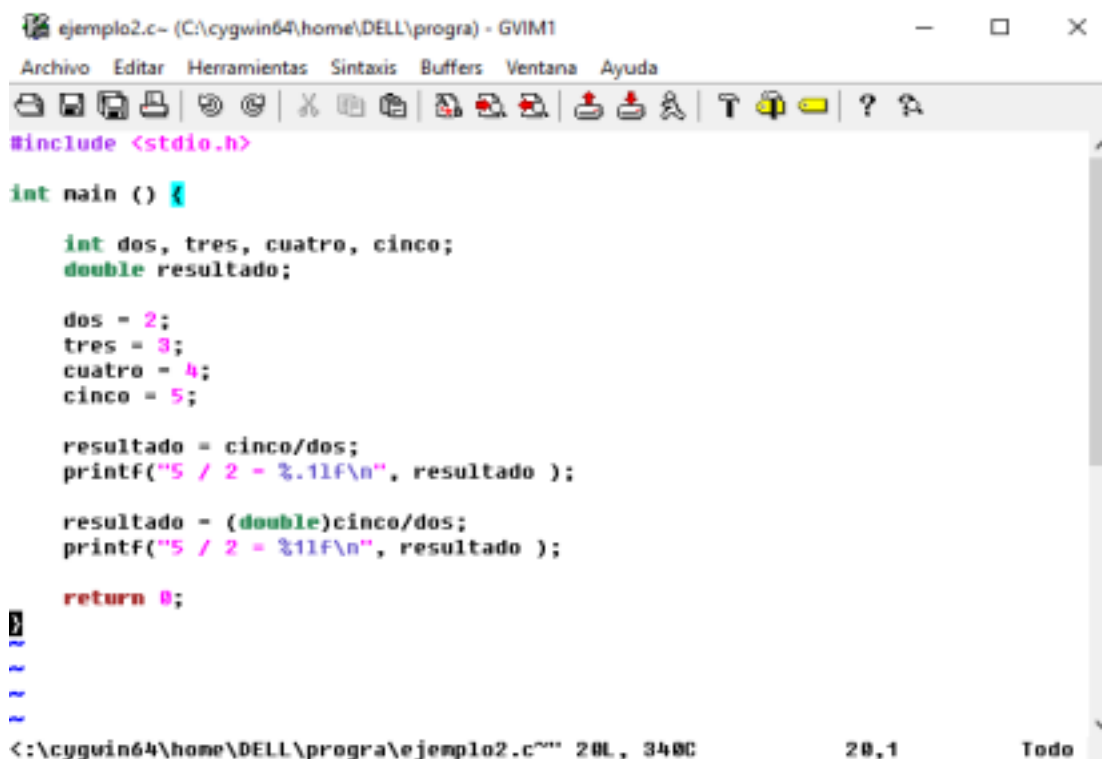
DELL@DESKTOP-N3V5QTU ~/progra
$ gcc primerprog.c -o ejemplo1

DELL@DESKTOP-N3V5QTU ~/progra
$ ./ejemplo1
Primero texto solo
Luego podemos poner un entero: 32768
Tambien podemos poner un caracter: B
Y un numero real: 89.80
10
18.8
Tu entero: 10
Tu real: 18.800

DELL@DESKTOP-N3V5QTU ~/progra
$
```

Ejemplo 2.

En esta se hace uso de los operadores para hacer una operación de tipo división, y vemos que el especificar el dato de salida es necesario.



```
ejemplo2.c~ (C:\cygwin64\home\DELL\progra) - GVIM1
Archivo  Editar  Herramientas  Sintaxis  Buffers  Ventana  Ayuda
#include <stdio.h>

int main () {

    int dos, tres, cuatro, cinco;
    double resultado;

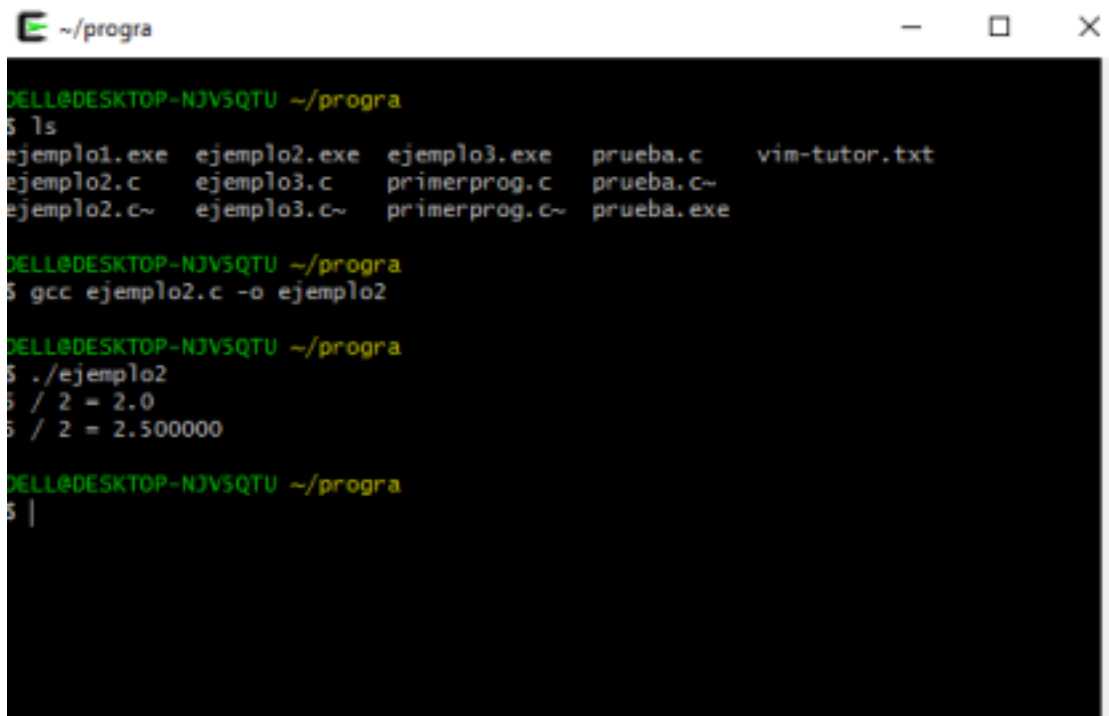
    dos = 2;
    tres = 3;
    cuatro = 4;
    cinco = 5;

    resultado = cinco/dos;
    printf("5 / 2 = %.11f\n", resultado );

    resultado = (double)cinco/dos;
    printf("5 / 2 = %11f\n", resultado );

    return 0;
}
```

En su compilación vemos que muestra dos resultados de los cuales uno es entero y el otro es real y esto es debido a la especificación en el tipo de variable.



```
DELL@DESKTOP-NJVSQTU ~/progra
$ ls
ejemplo1.exe  ejemplo2.exe  ejemplo3.exe  prueba.c  vim-tutor.txt
ejemplo2.c    ejemplo3.c    primerprog.c  prueba.c~
ejemplo2.c~   ejemplo3.c~   primerprog.c~ prueba.exe

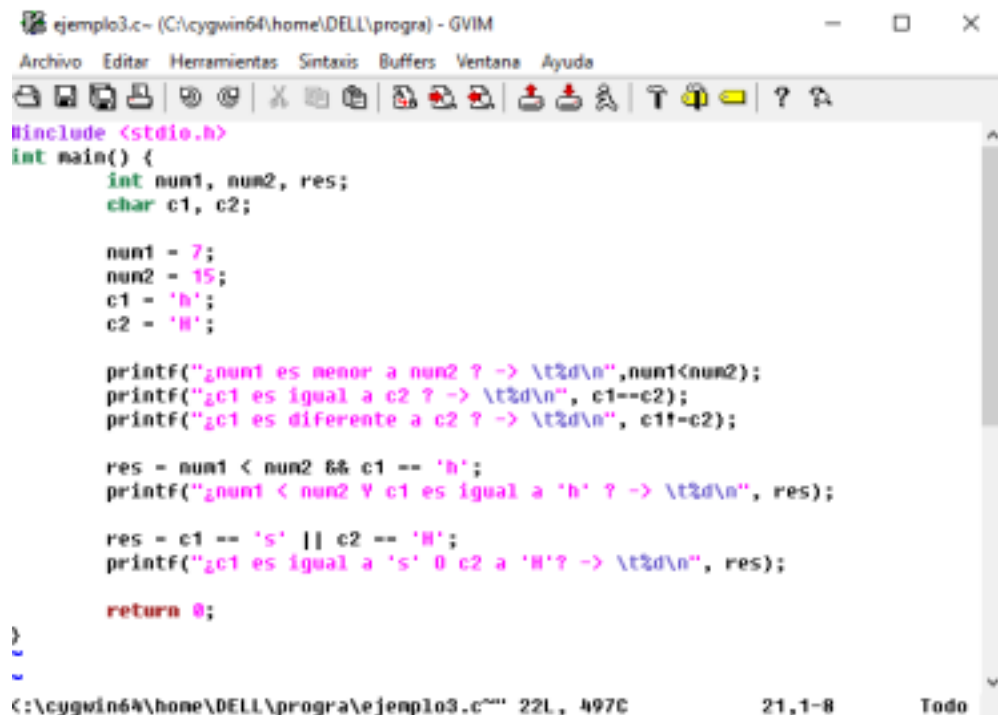
DELL@DESKTOP-NJVSQTU ~/progra
$ gcc ejemplo2.c -o ejemplo2

DELL@DESKTOP-NJVSQTU ~/progra
$ ./ejemplo2
/ 2 = 2.0
/ 2 = 2.500000

DELL@DESKTOP-NJVSQTU ~/progra
$ |
```

Ejemplo 3.

En este ejemplo se hace uso de operadores lógicos en el desarrollo del programa, para hacer comparaciones, y el resultado que se obtiene es de verdadero o falso expresado en código binario.



```
ejemplo3.c~ (C:\cygwin64\home\DELL\progra) - GVIM
Archivo  Editor  Herramientas  Sintaxis  Buffers  Ventana  Ayuda

#include <stdio.h>
int main() {
    int num1, num2, res;
    char c1, c2;

    num1 = 7;
    num2 = 15;
    c1 = 'h';
    c2 = 'H';

    printf("¿num1 es menor a num2 ? -> %td\n", num1 < num2);
    printf("¿c1 es igual a c2 ? -> %td\n", c1 == c2);
    printf("¿c1 es diferente a c2 ? -> %td\n", c1 != c2);

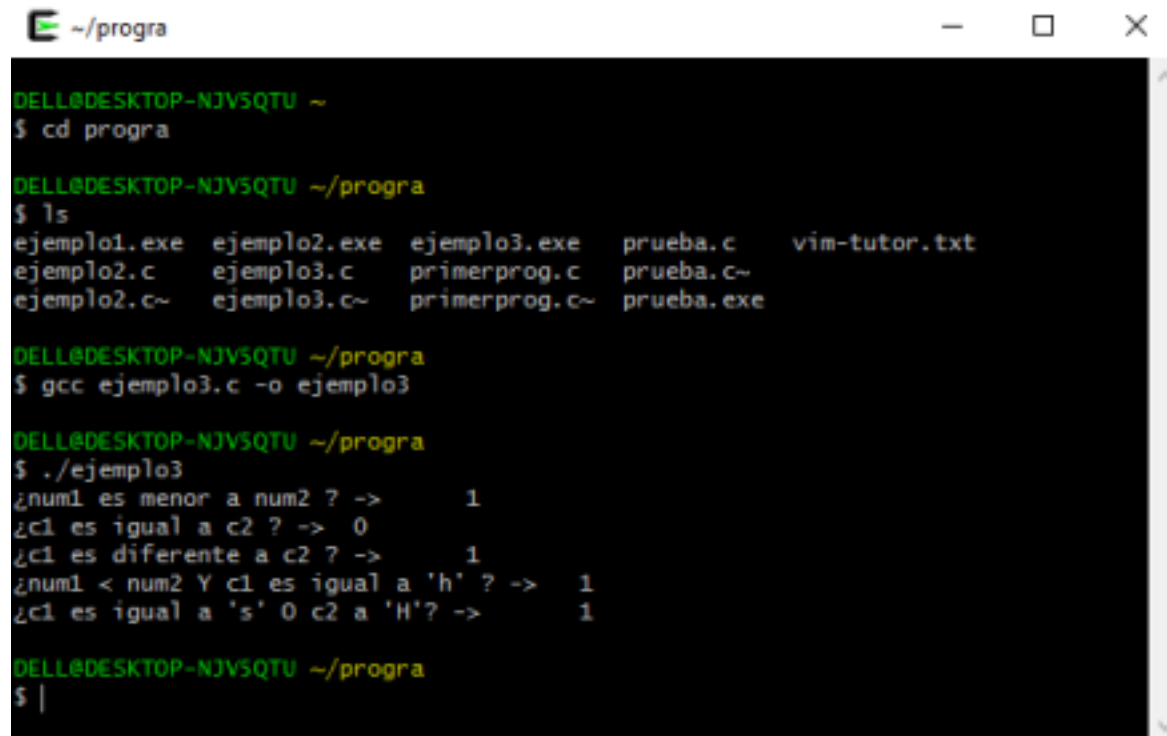
    res = num1 < num2 && c1 == 'h';
    printf("¿num1 < num2 y c1 es igual a 'h' ? -> %td\n", res);

    res = c1 == 's' || c2 == 'H';
    printf("¿c1 es igual a 's' o c2 a 'H' ? -> %td\n", res);

    return 0;
}

C:\cygwin64\home\DELL\progra\ejemplo3.c~ 22L, 497C 21,1-8 Todo
```

En su compilación podemos ver estos resultados.



```
DELL@DESKTOP-NJVSQTU ~  
$ cd progra  
  
DELL@DESKTOP-NJVSQTU ~/progra  
$ ls  
ejemplo1.exe  ejemplo2.exe  ejemplo3.exe  prueba.c  vim-tutor.txt  
ejemplo2.c    ejemplo3.c    primerprog.c  prueba.c~  
ejemplo2.c~   ejemplo3.c~   primerprog.c~ prueba.exe  
  
DELL@DESKTOP-NJVSQTU ~/progra  
$ gcc ejemplo3.c -o ejemplo3  
  
DELL@DESKTOP-NJVSQTU ~/progra  
$ ./ejemplo3  
¿num1 es menor a num2 ? ->      1  
¿c1 es igual a c2 ? ->    0  
¿c1 es diferente a c2 ? ->    1  
¿num1 < num2 Y c1 es igual a 'h' ? ->    1  
¿c1 es igual a 's' O c2 a 'H'? ->    1  
  
DELL@DESKTOP-NJVSQTU ~/progra  
$ |
```

Conclusión:

Podemos ver que desarrollar programas en Lenguaje C, es parecido de los que utilizábamos cuando desarrollábamos pseudocódigos, sin embargo, en este lenguaje si se tiene que ser mas específico, en cuanto al tipo de variable, cuanto espacio le queremos dar dentro de la memoria de programa, de igual manera el lenguaje se vuelve extenso debido a que en los resultados y en el procedimiento se tiene que seguir especificando que tipo de variable estamos ocupando. Además, ahora si se tiene que precisar la sintaxis, ya que, si hay un error de dedo, por así decirlo, al momento de compilarlo en la terminal el programa no se ejecutara y si se llegara a ejecutar no hará lo que estamos esperando.