



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: M.I. MARCO ANTONIO MARTINEZ QUINTANA

Asignatura: FUNDAMENTOS DE PROGRAMACIÓN

Grupo: 3

No de Práctica(s): 7

Integrante(s): CARRILLO CERVANTES IVETTE ALEJANDRA

*No. de Equipo de
cómputo empleado:* NO APLICA

No. de Lista o Brigada: 7

Semestre: PRIMER SEMESTRE

Fecha de entrega: 23 NOVIEMBRE 2020

Observaciones:

CALIFICACIÓN: _____

Fundamentos del lenguaje C

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.

Actividades:

- Crear un programa en lenguaje C que tenga definidas variables de varios tipos, se les asigne valores adecuados (por lectura o asignación directa) y muestre su valor en la salida estándar.
- En un programa en C, asignar valores a variables utilizando expresiones aritméticas; algunas con uso de cambio de tipo (cast)
- Elaborar expresiones relacionales/lógicas en un programa en C y mostrar el resultado de su evaluación.

Introducción:

Por otro lado, C es un lenguaje compilado, es decir, existe un programa (llamado compilador) que, a partir de un código en lenguaje C, genera un código objeto (ejecutable). Para crear un programa en C se siguen tres etapas principales:

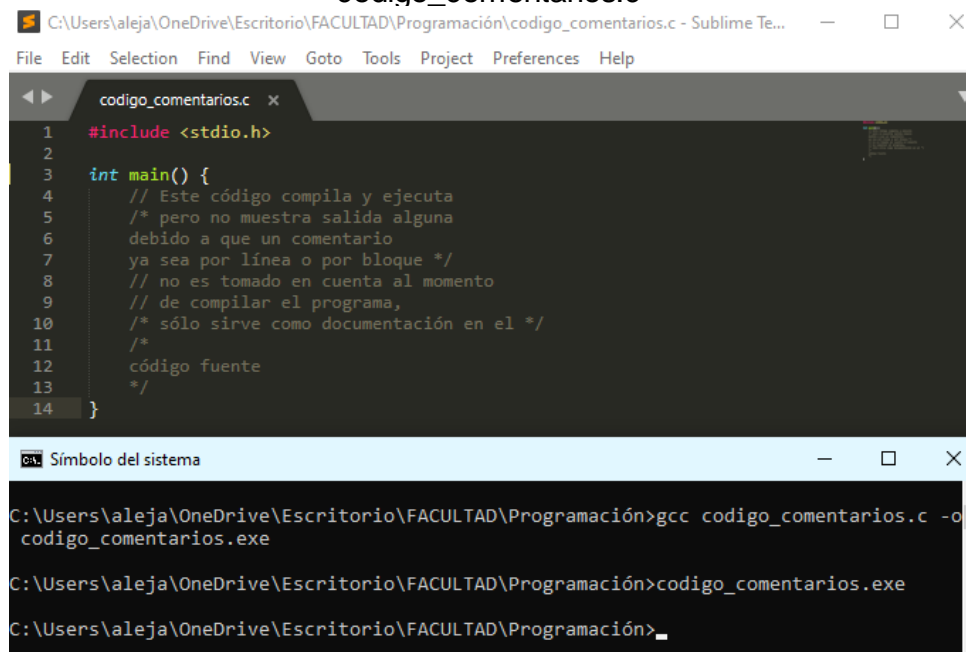
- Edición: Se escribe el código fuente en lenguaje C desde algún editor de textos.
- Compilación: A partir del código fuente (lenguaje C) se genera el archivo en lenguaje máquina (se crea el programa objeto o ejecutable).
- Ejecución: El archivo en lenguaje máquina se puede ejecutar en la arquitectura correspondiente.

Al momento de ejecutar un programa objeto (código binario), se ejecutarán únicamente las instrucciones que estén definidas dentro de la función principal.

Comentarios:

//Realizar comentarios para documentar el programa.

codigo_comentarios.c



```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\codigo_comentarios.c - Sublime Te...
File Edit Selection Find View Goto Tools Project Preferences Help

codigo_comentarios.c x
1 #include <stdio.h>
2
3 int main() {
4     // Este código compila y ejecuta
5     /* pero no muestra salida alguna
6     debido a que un comentario
7     ya sea por línea o por bloque */
8     // no es tomado en cuenta al momento
9     // de compilar el programa,
10    /* sólo sirve como documentación en el */
11    /*
12    código fuente
13    */
14 }
```

```
cmd Símbolo del sistema
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>gcc codigo_comentarios.c -o
codigo_comentarios.exe
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>codigo_comentarios.exe
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>_
```

Declaración de variables

Para declarar variables en C se sigue la siguiente sintaxis:

[modificadores] tipoDeDato identificador [= valor];

Típos de datos

Los tipos de datos básicos en C son:

- Caracteres: codificación definida por la máquina.
- Enteros: números sin punto decimal.
- Flotantes: números reales de precisión normal.
- Dobles: números reales de doble precisión.

Identificador

Un identificador es el nombre con el que se va a almacenar en memoria un tipo de dato.

Los identificadores siguen las siguientes reglas:

- Debe iniciar con una letra [a-z].
- Puede contener letras [A-Z, a-z], números [0-9] y el carácter guión bajo (_).

codigo_declaraciondeVariables.c

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación\codigo_declaraciondeVariables.c - S...
File Edit Selection Find View Goto Tools Project Preferences Help

codigo_declaraciondeVariables.c x
1  #include <stdio.h>
2  /*
3     Este programa muestra la manera en la que se declaran y asignan variables
4     de diferentes tipos: numéricas (enteras y reales) y caracteres.
5  */
6  int main() {
7     // Variables enteras
8     short enteroNumero1 = 32768;
9     signed int enteroNumero2 = 55;
10    unsigned long enteroNumero3 = -789;
11    char caracterA = 65; // Convierte el entero (ASCII) a carácter
12    char caracterB = 'B';
13    // Variables reales
14    float puntoFlotanteNumero1 = 89.8;
15    // La siguiente sentencia genera un error al compilar
16    // porque los valores reales siempre llevan signo
17    // unsigned double puntoFlotanteNumero2 = -238.2236;
18    double puntoFlotanteNumero2 = -238.2236;
19    return 0;
20 }
```

Símbolo del sistema

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>gcc codigo_declaraciondeVariables.c -o codigo_declaraciondeVariables.exe

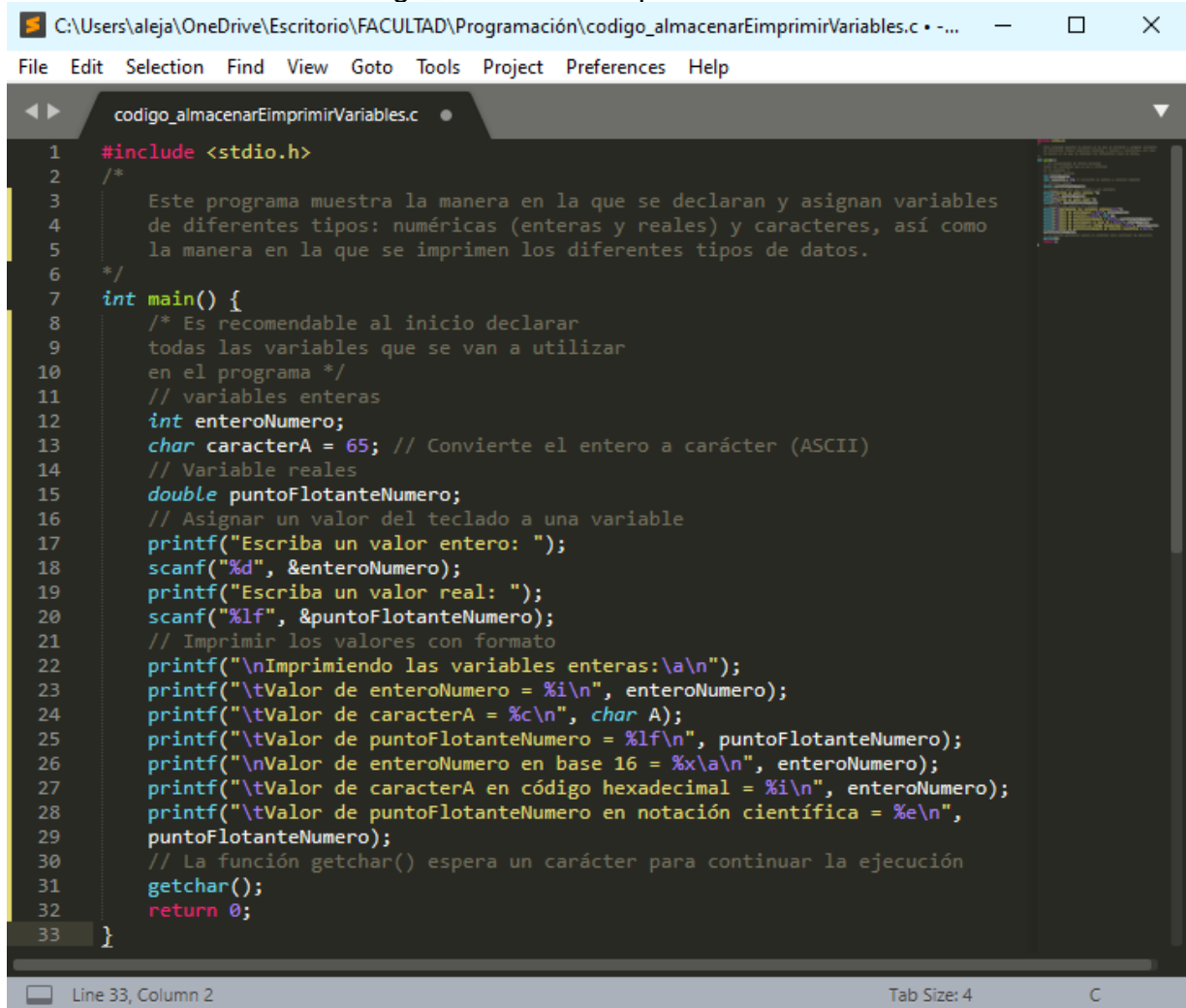
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>codigo_declaraciondeVariables.exe

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>
```

Para imprimir con formato también se utilizan algunas secuencias de caracteres de escape, C maneja los siguientes:

- \a carácter de alarma
- \b retroceso
- \f avance de hoja
- \n salto de línea
- \r regreso de carro
- \t tabulador horizontal
- \v tabulador vertical
- \0 carácter nulo

codigo_almacenarEimprimirVariables.c



```
1  #include <stdio.h>
2  /*
3   Este programa muestra la manera en la que se declaran y asignan variables
4   de diferentes tipos: numéricas (enteras y reales) y caracteres, así como
5   la manera en la que se imprimen los diferentes tipos de datos.
6  */
7  int main() {
8      /* Es recomendable al inicio declarar
9       todas las variables que se van a utilizar
10      en el programa */
11      // variables enteras
12      int enteroNumero;
13      char caracterA = 65; // Convierte el entero a carácter (ASCII)
14      // Variable reales
15      double puntoFlotanteNumero;
16      // Asignar un valor del teclado a una variable
17      printf("Escriba un valor entero: ");
18      scanf("%d", &enteroNumero);
19      printf("Escriba un valor real: ");
20      scanf("%lf", &puntoFlotanteNumero);
21      // Imprimir los valores con formato
22      printf("\nImprimiendo las variables enteras:\n");
23      printf("\tValor de enteroNumero = %i\n", enteroNumero);
24      printf("\tValor de caracterA = %c\n", caracterA);
25      printf("\tValor de puntoFlotanteNumero = %lf\n", puntoFlotanteNumero);
26      printf("\nValor de enteroNumero en base 16 = %x\n", enteroNumero);
27      printf("\tValor de caracterA en código hexadecimal = %i\n", enteroNumero);
28      printf("\tValor de puntoFlotanteNumero en notación científica = %e\n",
29             puntoFlotanteNumero);
30      // La función getchar() espera un carácter para continuar la ejecución
31      getchar();
32      return 0;
33 }
```

Modificadores de alcance

Los modificadores que se pueden agregar al inicio de la declaración de variables son const y static.

El modificador const impide que una variable cambie su valor durante la ejecución del programa, es decir, permite para crear constantes. Por convención, las constantes se escriben con mayúsculas y se deben inicializar al momento de declararse.

codigo_variablesestaticasYconstantes.c

```
1 #include <stdio.h>
2 /*
3 Este programa muestra la manera en la que se declaran y asignan
4 las variables estáticas y las constantes.
5 */
6 int main() {
7     const int constante = 25;
8     static char a = 'a';
9     printf("Valor constante: %i\n", constante);
10    printf("Valor estático: %c\n", a);
11    // El valor de la variable declarada como constante no puede cambiar.
12    // La siguiente línea genera un error al compilar si se quita el comentario:
13    // constante = 30;
14    // las variables estáticas sí pueden cambiar de valor
15    a = 'b';
16    printf("\nValor estático: %c\n", a);
17    return 0;
18 }
```

Simbolo del sistema

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>codigo_variablesestaticasYconstantes.exe
Valor constante: 25
Valor estático: a
Valor estático: b
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>
```

Moldeo o cast

El resultado de una operación entre dos tipos de datos iguales puede dar como resultado un tipo de dato diferente, en esos casos es necesario moldear el resultado. A este proceso se le conoce como cast.

Ejemplo7.c

```
1 // Si se tienen 2 enteros
2 int cinco = 5, dos = 2;
3 // La operación de división entre dos enteros
4 // genera un valor real, en este caso hay que
5 // moldear (cast) el resultado del lado derecho del
6 // igual para que corresponda con el lado izquierdo
7 // y se pueda asignar.
8 double res = (double)cinco/dos;
9 // Si no se hiciese el cast el resultado se truncaría.
```

codigo_operadores.c

```
1 #include <stdio.h>
2 /*
3 Este programa muestra la manera en la que se realiza un moldeo o cast.
4 También muestra como manipular números a nivel de bits:
5 - Corrimiento de bits a la izquierda y a la derecha
6 - Operador AND a nivel de bits
7 - Operador OR a nivel de bits
8 */
9 int main(){
10     short ocho, cinco, cuatro, tres, uno;
11     // 8 en binario: 0000 0000 0000 1000
12     ocho = 8;
13     // 5 en binario: 0000 0000 0000 0101
14     cinco = 5;
15     // 4 en binario: 0000 0000 0000 0100
16     cuatro = 4;
17     // 3 en binario: 0000 0000 0000 0011
18     tres = 3;
19     // 2 en binario: 0000 0000 0000 0010
20     dos = 2;
21     // 1 en binario: 0000 0000 0000 0001
22     uno = 1;
23     printf("Operadores aritméticos\n");
24     double res = (double)cinco/dos; // Cast
25     printf("5 / 2 = %f\n", res);
26     printf("5 modulo 2 = %d\n", cinco%dos);
27     printf("Operadores lógicos\n");
28     printf("8 >> 2 = %d\n", ocho>>2);
29     printf("8 << 1 = %d\n", ocho<<1);
30     printf("5 & 4 = %d\n", cinco&cuatro);
31     printf("3 | 2 = %d\n", tres|dos);
32     printf("\n");
33     return 0;
34 }
```

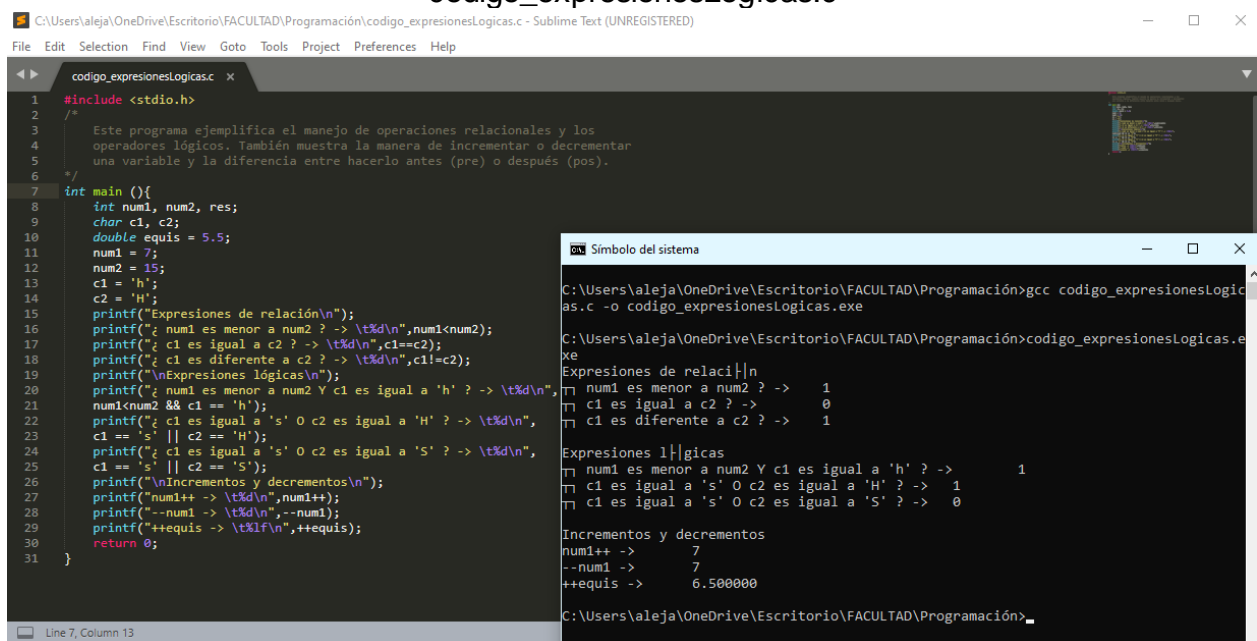
Simbolo del sistema

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>codigo_declaraciondeVariables.exe
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>gcc codigo_operadores.c -o codigo_operadores.exe
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>codigo_operadores.exe
Operadores aritméticos
5 / 2 = 2.500000
5 modulo 2 = 1
Operadores lógicos
8 >> 2 = 2
8 << 1 = 16
5 & 4 = 4
3 | 2 = 3
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>
```

Expresiones lógicas

Las expresiones lógicas están constituidas por números, caracteres, constantes o variables que están relacionados entre sí por operadores lógicos. Una expresión lógica puede tomar únicamente los valores verdadero o falso.

codigo_expresionesLogicas.c



```
1 #include <stdio.h>
2
3 /*
4  Este programa ejemplifica el manejo de operaciones relacionales y los
5  operadores lógicos. También muestra la manera de incrementar o decrementar
6  una variable y la diferencia entre hacerlo antes (pre) o después (pos).
7 */
8 int main (){
9     int num1, num2, res;
10    char c1, c2;
11    double equis = 5.5;
12    num1 = 7;
13    num2 = 15;
14    c1 = 'h';
15    c2 = 'H';
16    printf("Expresiones de relación\n");
17    printf("num1 es menor a num2 ? -> %d\n", num1 < num2);
18    printf("c1 es igual a c2 ? -> %d\n", c1 == c2);
19    printf("c1 es diferente a c2 ? -> %d\n", c1 != c2);
20    printf("\nExpresiones lógicas\n");
21    printf("num1 es menor a num2 Y c1 es igual a 'h' ? -> %d\n",
22           num1 < num2 && c1 == 'h');
23    printf("c1 es igual a 's' O c2 es igual a 'H' ? -> %d\n",
24           c1 == 's' || c2 == 'H');
25    printf("c1 es igual a 's' O c2 es igual a 'S' ? -> %d\n",
26           c1 == 's' || c2 == 'S');
27    printf("\nIncrementos y decrementos\n");
28    printf("num1++ -> %d\n", num1++);
29    printf("--num1 -> %d\n", --num1);
30    printf("++equis -> %lf\n", ++equis);
31 }
```

Símbolo del sistema

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>gcc codigo_expresionesLogicas.c -o codigo_expresionesLogicas.exe
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>codigo_expresionesLogicas.exe
Expresiones de relación
num1 es menor a num2 ? -> 1
c1 es igual a c2 ? -> 0
c1 es diferente a c2 ? -> 1
Expresiones lógicas
num1 es menor a num2 Y c1 es igual a 'h' ? -> 1
c1 es igual a 's' O c2 es igual a 'H' ? -> 1
c1 es igual a 's' O c2 es igual a 'S' ? -> 0
Incrementos y decrementos
num1++ -> 7
--num1 -> 7
++equis -> 6.500000
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\Programación>
```

Depuración de programas

Cuando un programa falla (no termina su ejecución de manera correcta) y la información enviada por el compilador es muy general, se puede ejecutar el programa en un contexto controlado para saber, exactamente, dónde está fallando. Se revisará este tema en la guía práctica de estudio “Depuración de programas” para conocer las diferentes herramientas que nos ayudan a encontrar los errores de un programa.

Conclusión.

En esta práctica aprendí acerca lo básico acerca del lenguaje de programación C, algunas cosas se me hicieron conocidas, puesto que en en CCH vi Java y son muy similares las estructuras; me percaté de algunas de sus diferencias, al igual que aprendí cosas nuevas acerca de C.

Referencias.

Manual de prácticas del Laboratorio de Fundamentos de programación. 23 Noviembre 2020, de Facultad de Ingeniería Sitio web: <http://lcp02.fi-b.unam.mx/>