	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): Karina García Morales

Asignatura: Fundamentos de programación

Grupo: 22

No de Práctica(s): 6

Integrante(s): Vicente Martínez Edilberto

No. de lista o brigada: 50

Semestre: 1

Fecha de entrega: 16 de octubre del 2025

Observaciones: _____

_____

_____

CALIFICACIÓN: _____

Practica 06: Entorno y fundamentos del lenguaje C

Objetivo:
El alumnado elaborará 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.

Desarrollo:
Una vez que un problema dado ha sido analizado (se identifican los datos de entrada y la salida deseada), que se ha diseñado un algoritmo que lo resuelva de manera eficiente (procesamiento de datos), y que se ha representado el algoritmo de manera gráfica o escrita (diagrama de flujo o pseudocódigo) se puede proceder a la etapa de codificación.

La codificación se puede realizar en cualquier lenguaje de programación estructurada. En este curso se aprenderá el uso del lenguaje de programación C.

Entorno de C

Un lenguaje de programación permite expresar una serie de instrucciones que podrán ser realizadas por una computadora. Uno de los lenguajes de programación mayormente difundidos es el lenguaje C.

Una característica importante del lenguaje C es que es muy poderoso ya que combina las características de un lenguaje de alto nivel (facilidad de programación), con uno de bajo nivel (manejo más preciso de una máquina); por lo que se han creado variantes que permiten programar miles de dispositivos electrónicos en el mundo con sus respectivos compiladores.

Un programa en C se elabora describiendo cada una de las instrucciones de acuerdo con las reglas definidas en este lenguaje en un archivo de texto para después ser procesadas en un compilador. Un compilador es un programa que toma como entrada un archivo de texto y tiene como salida un programa ejecutable, éste tiene instrucciones que pueden ser procesadas por el hardware de la computadora en conjunto con el

sistema operativo que corre sobre ella. Se tiene como ventaja que un programa escrito en lenguaje C, siguiendo siempre su estándar, puede ejecutarse en cualquier máquina siempre y cuando exista un compilador de C. A esto también se le conoce como multiplataforma.

Editores
Un programa en C debe ser escrito en un editor de texto para después generar un programa ejecutable en la computadora por medio de un compilador. Tanto el editor de texto como el compilador van de la mano con el sistema operativo y si posee o no interfaz gráfica, por lo que son factores que se deben tomar en cuenta a la hora de elegir el entorno para desarrollar programas en C.

Es importante señalar que no es lo mismo un editor de texto que un procesador de texto.

El primero edita un texto plano que puede tener muchas utilidades como guardar una configuración, tener escrito un programa, etcétera, y será interpretado hasta que se haga una lectura de éste. Un procesador de texto permite dar formato al texto, a la hoja donde está escrito, incrustar imágenes, entre otros, su salida puede ser un archivo de texto plano que contiene etiquetas que señalan el formato que se le dio al texto o algo un poco más complejo.

Editor Visual Interface de GNU/Linux (vi)
El editor vi (visual interface) es el editor más común en cualquier distribución de sistemas
operativos con núcleo basado en UNIX. vi es un editor que puede resultar difícil de usar en un inicio. Aunque existen editores más intuitivos en su uso; en muchas ocasiones vi es el único disponible.

Para iniciar vi, debe teclearse desde la línea de comandos:
vi nombre_archivo [.ext]

Donde nombre_archivo es el nombre del archivo a editar o el nombre de un archivo nuevo que se creará con vi, y [.ext] se refiere a la extensión que indica que el texto es una programa escrito en algún lenguaje o es texto plano.

Modo de última línea:
Se puede acceder a él utilizando el modo comando pero los comandos no tendrán efecto
hasta que se presiona la tecla Enter además de que se visualizará el comando en la última línea del editor. Es posible cancelar el comando con la tecla Esc. Los comandos de última línea se caracterizan porque inician con /, ? o :. Algunos ejemplos son:
● /texto donde la cadena texto será buscada hacia delante de donde se encuentra el cursor.
● ?texto donde la cadena texto será buscada hacia atrás de donde se encuentra

el cursor.
● :q para salir de vi sin haber editado el texto desde la última vez que se guardó.
● :q! para salir de vi sin guardar los cambios.
● :w para guardar los cambios sin salir de vi.
● :w archivo para realizar la orden “guardar como”, siendo archivo el nombre donde se guardará el documento.
● :wq guarda los cambios y sale de vi.

Modo Edición
Para poder editar un texto dentro del editor, solamente tendremos que oprimir la tecla <i> desde el modo comando y podremos insertar el texto en donde se encuentre el cursor y comienza a escribir justo en el carácter, con <a> podremos insertar el texto en donde se encuentre el cursor y comenzar a escribir justo después del carácter; si oprimimos la tecla <I> podremos insertar texto desde el inicio de la línea donde se encuentre el cursor y con <A> podremos insertar texto al final de la línea donde se encuentre el cursor.

GNU nano
Es un editor de texto disponible para sistemas operativos basados en UNIX en línea de comandos. Se puede acceder en un entorno gráfico desde la aplicación de terminal.

nano es un editor clon de otro editor llamado pico.
Para iniciar nano, debe teclearse desde la línea de comandos:
nano nombre_archivo[.ext]
Donde “nombre_archivo” es el nombre del archivo a editar o el nombre de un archivo nuevo.
Una vez en el editor, en la parte inferior se pueden observar los comandos básicos. Si se presiona la tecla F1 es posible visualizar la ayuda con la lista de todos comandos que
existen.
Los atajos de teclado pueden corresponder a:
● ^ que es la tecla Ctrl.
● M- que es la tecla Esc o bien Alt.

Compiladores
Una vez codificado un programa en C en algún editor de texto, éste debe ser leído por un programa que produzca un archivo ejecutable. A este programa se le conoce como compilador. Para el caso del lenguaje C el compilador traduce el código fuente del programa a código ejecutable.
Un programa en C tampoco puede ser escrito de manera arbitraria debe respetar una serie de reglas para que el compilador pueda entenderlas y realizar su función. Un estándar muy común es ANSI C y existen diferentes extensiones como ISOC99 y GNU C que representan mejoras para el estándar original. Realizar un programa en dicho estándar garantiza que puede ejecutarse en cualquier máquina siempre y cuando exista

un compilador hecho para ella. A veces, el programador no sigue un estándar o lo desconoce, usando características no estándar que, a la hora de usar el mismo programa para otra máquina no funciona, teniendo que realizar adaptaciones que se reflejan en costos. Por ejemplo, es muy común empezar a desarrollar programas en C en plataforma Windows en procesadores x86 usando características propias. Al trasladar, por alguna necesidad, dicho programa a plataforma GNU/Linux con procesador ARM, el programa no funcionará porque no se siguió el estándar que garantiza universalidad para el lenguaje C.

gcc (GNU Compiler Collection)

Es un conjunto de compiladores de uso libre para sistemas operativos basados en UNIX.

Entre sus compiladores existe el que sirve para programas escritos en C. Se encuentra por defecto en diversas distribuciones de GNU/Linux. El compilador trabaja en línea de comandos.

Existe también una versión modificada que puede ejecutar y crear programas para plataformas Windows en un paquete llamado MinGW (Minimalist GNU for Windows). Al compilar un programa en C el compilador genera diversos archivos intermedios que corresponden a las distintas fases que realiza. Éstas no son de interés por el momento y son eliminadas una vez obtenido el archivo ejecutable. gcc tiene diferentes opciones de ejecución para usuarios más avanzados.

Suponiendo que se tiene un programa escrito en C y se le llamó calculadora.c, la manera de compilarlo es localizándose mediante la línea de comandos en la ruta donde el archivo se encuentra y ejecutando el comando:
gcc calculadora.c

Esto creará un archivo a.out (en Windows a.exe) que es el programa ejecutable resultado de la compilación.

Si se desea que la salida tenga un nombre en particular, debe definirse por medio del parámetro -o de gcc. Por ejemplo, para que se llame calculadora.out (en Windows calculadora.exe), se escribe en la línea de comandos:
gcc calculadora.c -o calculadora.out

A veces, para realizar un programa más complejo, se necesitan bibliotecas que se instalaron en el equipo previamente y se definió su uso en el programa escrito en C pero al momento de compilar es necesario indicar a GCC que se están usando bibliotecas que no se encuentran en su repertorio de bibliotecas estándar. Para ello es necesario utilizar el parámetro -l seguido inmediatamente por el nombre de la biblioteca, sin dejar espacio alguno:

gcc calculadora.c -o calculadora -lnombre_biblioteca
Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés)

Los IDE están diseñados para maximizar la productividad del programador proporcionando componentes muy unidos con interfaces de usuario similares. Los IDE presentan un único programa en el que se llevan a cabo las diversas etapas de desarrollo de software. Generalmente, este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de software. Esto contrasta con el desarrollo de software utilizando herramientas no relacionadas, como vi, GNU Compiler Collection (gcc) o make.
Dev-C++: Este emplea el compilador MinGW. Se trata de un software libre, sencillo, ligero y eficiente, para la plataforma Windows.
Code Blocks: Este es un software libre, multiplataforma. Code Blocks es una alternativa a Dev-C++ y desarrollada mediante el propio lenguaje C++. Sus capacidades son bastante buenas y es muy popular entre los nuevos programadores. Se puede encontrar
separado del compilado o la versión “mingw” que incluye g++ (gcc para C++).

Ejecución

La ejecución es la etapa que sigue después de haber compilado el programa. Una vez compilado el programa, se puede distribuir para equipos que ejecuten el mismo sistema operativo y tengan la misma plataforma de hardware (tipo de procesador, set de instrucciones y arquitectura en general). Los pasos para realizar la ejecución dependen del sistema operativo y del entorno.
En Windows se puede ejecutar el programa haciendo doble clic sobre el programa ya compilado, pero se recomienda exhaustivamente que se haga desde símbolo de sistema.
Considerando que se tiene un programa compilado en un sistema base Unix cuyo nombre es calculadora.out, para ejecutar debe teclearse en línea de comandos:
./calculadora.out
Si el programa realizado necesita tener una entrada de información por medio de argumentos, éstos se colocan así:
./calculadora.out argumento1 argumento2

Lenguaje de programación C
C es un lenguaje de propósito general basado en el paradigma estructurado. El teorema del programa estructurado, demostrado por Böhm-Jacopini, dicta que todo programa puede desarrollarse utilizando únicamente 3 instrucciones de control: Secuencia, Selección e Iteración.
Un programa en C consiste en una o más funciones, de las cuales una de ellas debe llamarse main( ) y es la principal.
Al momento de ejecutar un programa objeto (código binario), se procesarán las instrucciones que estén definidas dentro de la función principal. Dicha función puede

contener sentencias, estructuras de control y comentarios. Dentro de las sentencias se encuentran la declaración y/o asignación de variables, la realización de operaciones básicas, y las llamadas a funciones.

El conjunto de caracteres en C

Del mismo modo que en nuestro lenguaje habitual utilizamos un conjunto de caracteres para construir instrucciones que tengan significado, los programas que se realicen en C se escriben utilizando un conjunto de caracteres formado por lo siguiente:

• Las 26 letras minúsculas del alfabeto inglés (a b c d e f g h i j k l m n o p q r s t u v w x y z).
• Las 26 letras mayúsculas (A B C D E F G H I J K L M N O P Q R S T U V W X Y Z).
• Los 10 dígitos (1 2 3 4 5 6 7 8 9 0).
• Los símbolos especiales (@ ^ { } [ ] ( ) & $ % # ~ ‘ “ / ? ; : _ . , = / * - +).
• El espacio en blanco o barra espaciadora.

Nota: Lenguaje C es sensible a minúsculas y mayúsculas.

Palabras reservadas

Las palabras reservadas (Tabla 1) son palabras que tienen un significado predefinido estándar y sólo se pueden utilizar para su propósito ya establecido; no se pueden utilizar
como identificadores definidos por el programador. En lenguaje C son las siguientes:

Tabla 1: Palabras reservadas

auto	*double*	*int*	*struct*
break	*else*	*long*	*switch*
case	*enum*	*register*	*typedef*
char	*extern*	*return*	*union*
const	*float*	*short*	*unsigned*
continue	*for*	*signed*	*void*
default	*goto*	*sizeof*	*volatile*
do	*if*	*static*	*while*

Tipos de datos

El lenguaje C ofrece distintos tipos de datos (Tabla 2), cada uno de los cuales se puede encontrar representado de forma diferente en la memoria de la computadora.

Tabla 2: Tipos de datos

Tipo	Descripción	Espacio en Memoria	Rango
int	Cantidad entera	2 bytes o una palabra* (varía de un compilador a otro)	-32767 a 32766
char	Carácter	1 byte	-128 a 127
float	Número en punto flotante (un número que incluye punto decimal y/o exponente)	1 palabra* (4 bytes)	$-3.4E^{38}$ a $3.4\ E^{38}$
double	Número en punto flotante de doble precisión (más cifras significativas y mayor valor posible del exponente)	2 palabras* (8 bytes)	$-1.7E^{308}$ a $1.7E^{308}$

Tarea:

1. ¿Cuál es el dato que se encuentra por default en el lenguaje C(signed o unsigned)?

   En el lenguaje C, el dato por default para los tipos de datos enteros (como int) es signed (con signo). Esto significa que puede almacenar tanto valores positivos como negativos.

2. Crea un programa en el que declares 4 variables haciendo uso de las reglas signed/unsigned,

3. las cuatro variables deben ser solicitadas al usuario(se emplea scanf) y deben mostrarse en

   pantalla (emplear printf)

   #include <stdio.h>

   int main() { signed int num1; // Variable con signo (puede ser negativa o positiva) unsigned int num2; // Variable sin signo (solo positiva) signed short num3; // Variable short con signo unsigned long num4; // Variable long sin signo

```
    // Pedir los valores al usuario     printf("Ingresa un número
entero con signo (signed int): ");     scanf("%d", &num1);

    printf("Ingresa un número entero sin signo (unsigned int): ");
scanf("%u", &num2);
    printf("Ingresa un número corto con signo (signed short): ");
scanf("%hd", &num3);

    printf("Ingresa un número largo sin signo (unsigned long): ");
scanf("%lu", &num4);

    // Mostrar los valores ingresados
printf("\n--- Valores ingresados ---\n");
printf("Signed int: %d\n", num1);
printf("Unsigned int: %u\n", num2);
printf("Signed short: %hd\n", num3);
printf("Unsigned long: %lu\n", num4);

    return 0;
}
```

3. Comparación entre Editor de Texto y Procesador de Texto:

Características	Editor de texto	Procesador de texto
Funciones principales	Permite crear y modificar archivos de texto plano	Permite crear, editar y dar formato a documentos con texto, como tablas
Formato de los contenidos	Solo texto, sin ningún tipo de formato como negritas o color	Este si permite textos con formato, como negritas y colores

Extensiones más comunes	.txt, .c, .html, .py, .cpp	.doc, .docx, .dot, .rft, .pdf

4. Indica los comandos utilizados para compilar y para ejecutar un programa en iOS o Linux:

   1- gcc programa.c -o programa

   2- ./programa

5. Indicar qué sucede cuando en una variable tipo carácter se emplea el formato %d, %i, %o, %x:

   %c: Muestra la variable en carácter.

   %d o %i: Muestra su código ASCII en decimal.

   %o: Lo muestra en sistema octal.

   %x: Lo muestra en sistema hexadecimal.

6. Genera un programa y ejecútalo en la interfaz que elijas con el número binario de tu letra inicial del nombre y realiza un corrimiento a la izquierda y uno a la derecha del bit más significativo.

   **********Notas sobre corrimientos**********

   Adjunto ejemplos de corrimiento a la izquierda y a la derecha <<   >>

   Corrimiento a la izquierda

   Al tener el número en binario se hace un corrimiento del bit más significativo a la izquierda

   ```
   #include <stdio.h>

   int main() {     unsigned char letra = 'V';  // Cambia 'V' por la letra inicial
   de tu nombre     unsigned char izquierda, derecha;

      printf("Letra: %c\n", letra);
   printf("Valor decimal: %d\n", letra);     //
   Mostrar en binario (manual, solo para
   ```

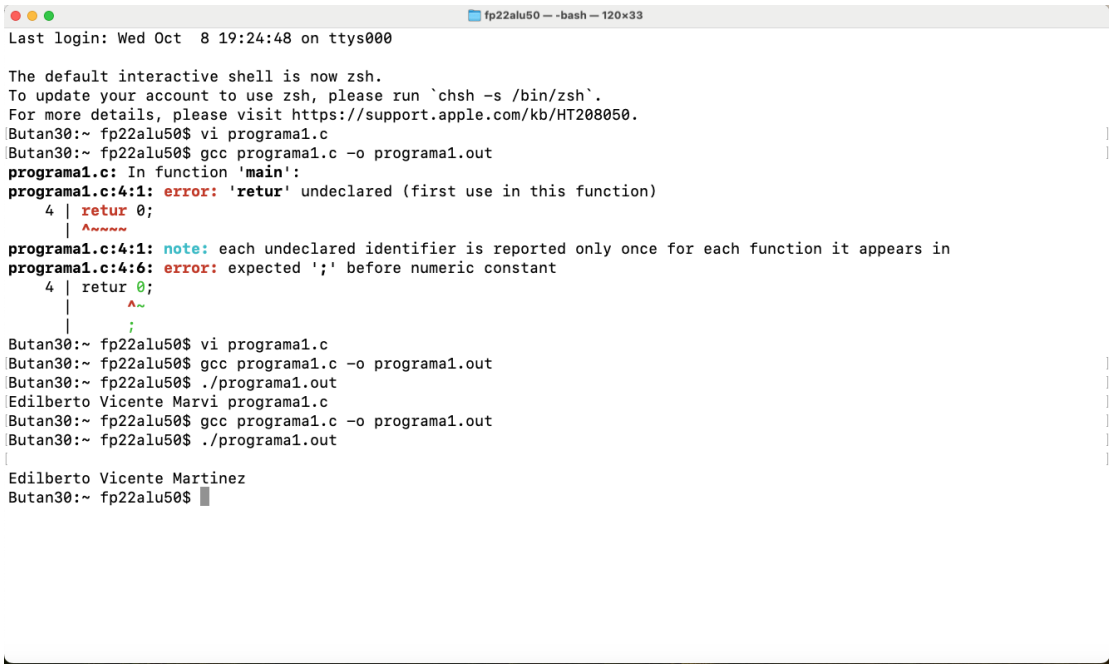
```
visualización)     printf("Valor binario
original: ");     for (int i = 7; i >= 0; i--) {
printf("%d", (letra >> i) & 1);
    }

    // Corrimiento     izquierda = letra << 1;  //
Corrimiento a la izquierda     derecha = letra >> 1;
// Corrimiento a la derecha

    // Mostrar resultados     printf("\n\nCorrimiento a la izquierda
(<< 1): %d\n", izquierda);     printf("Valor binario: ");     for (int i =
7; i >= 0; i--) {         printf("%d", (izquierda >> i) & 1);
    }

    printf("\n\nCorrimiento a la derecha (>> 1): %d\n",
derecha);     printf("Valor binario: ");     for (int i = 7; i >= 0; i--)
{         printf("%d", (derecha >> i) & 1);
    }
printf("\n");
return 0;
}
```

EVIDENCIAS DEL LABORATORIO



fp22alu50 — vi programa3.c — 87×24

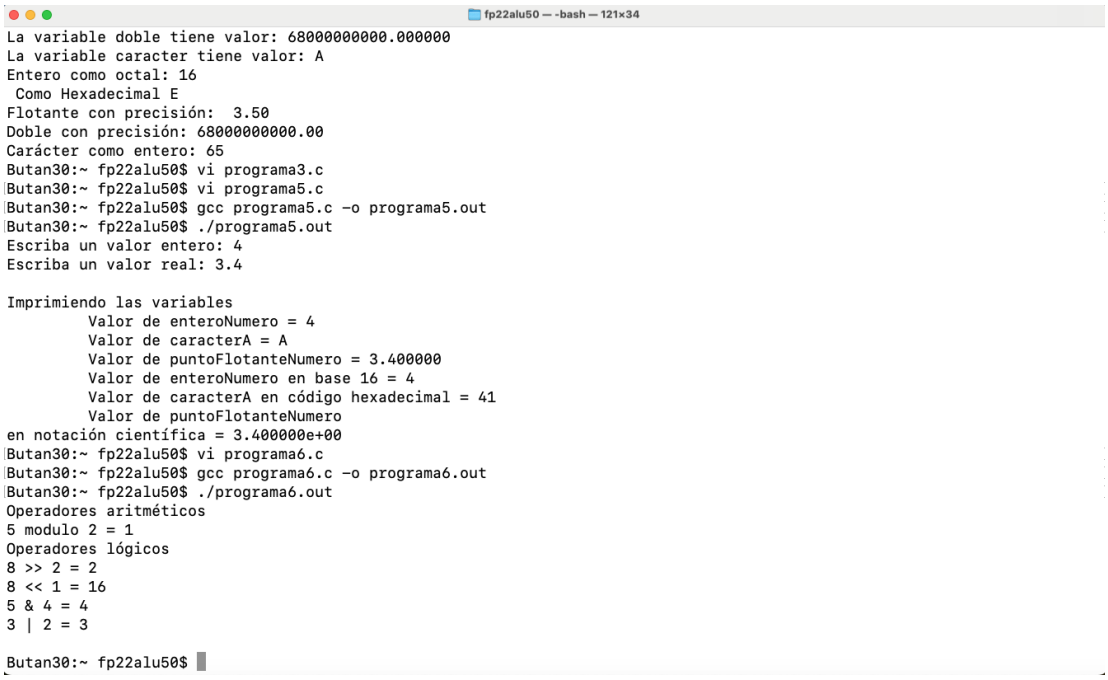
```
#include <stdio.h>
int main() {
 //Declaración de variables
int entero;
float flotante;
double doble;
char caracter;
//Asignación de variables
entero = 14;
flotante = 3.5f;
doble = 6.8e10;
caracter ='A';
//Funciones de salida de datos en pantalla
printf("La variable entera tiene valor: %i \n", entero);
printf("La variable flotante tiene valor: %f \n", flotante);
printf("La variable doble tiene valor: %f \n", doble);
printf("La variable caracter tiene valor: %c \n", caracter);
printf("Entero como octal: %o \n Como Hexadecimal %X \n", entero, entero);
printf("Flotante con precisión: %5.2f \n", flotante);
printf("Doble con precisión: %5.2f \n", doble);
printf("Carácter como entero: %d \n", caracter);
return 0;
}
"programa3.c" 23L, 724B
```

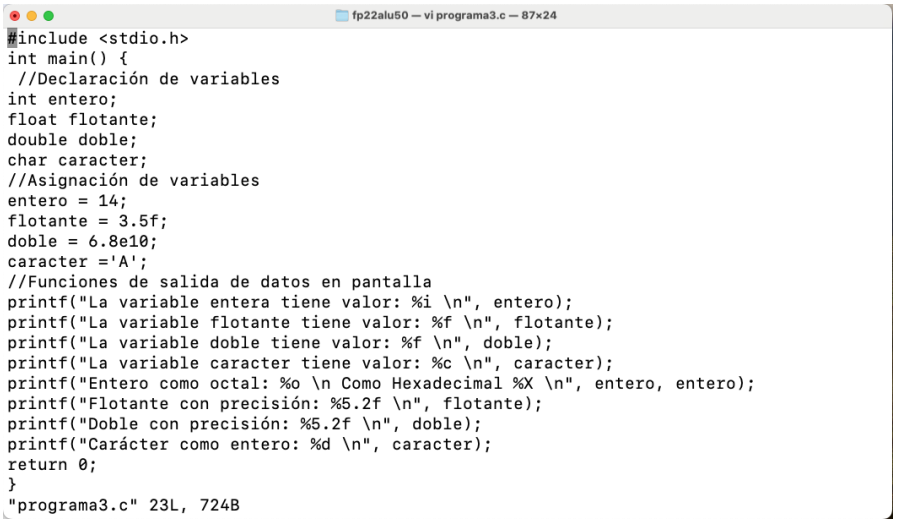
fp22alu50 — vi programa6.c — 121×34

```
#include <stdio.h>
int main()
{
 short ocho, cinco, cuatro, tres, dos, uno;

 // 8 en binario: 0000 0000 0000 1000
 ocho = 8;
 // 5 en binario: 0000 0000 0000 0101
 cinco = 5;
 // 4 en binario: 0000 0000 0000 0100
 cuatro = 4;
 // 3 en binario: 0000 0000 0000 0011
 tres = 3;
 // 2 en binario: 0000 0000 0000 0010
 dos = 2;
 // 1 en binario: 0000 0000 0000 0001
 uno = 1;
 printf("Operadores aritméticos\n");
 printf("5 modulo 2 = %d\n",cinco%dos);
 printf("Operadores lógicos\n");
 printf("8 >> 2 = %d\n",ocho>>dos);
 printf("8 << 1 = %d\n",ocho<<1);
 printf("5 & 4 = %d\n",cinco&cuatro);
 printf("3 | 2 = %d\n",tres|dos);

 printf("\n");
 return 0;
}
~
~
~
~
~
-- INSERT --
```





Conclusión:

En esta práctica pude ver el como es que empleamos lo que habíamos visto las clases anteriores solución de problemas y diagramas de flujo para ahora sí poder llevarlo a cabo en nuestro lenguaje a utilizar, el cual es el lenguaje tipo C, en lo personal, aunque en algunos comandos estoy un poco perdido, creo que más o menos ya se como llevar a cabo programas un poco más complejos pero en la computadora.

Bibliografía

http://lcp02.fi-b.unam.mx/