ÍNDICE

4 Elementos de Programación Imperativa
(Codificación, Compilación, Ejecución, Pruebas y Documentación)

En la fase de **Análisis** se determinar ***que és*** lo que el programa a desarrollar deberá de hacer, planteando y contestando las siguientes preguntas: ¿Cuáles son los datos de entrada? Y ¿Cuáles son los resultados esperados?; posteriormente en la fase de **Diseño** se debe determinar ***cómo*** se debe desarrollar un proceso que permita resolver el problema haciendo uso de diagramas de flujo y pseudocódigos.

Al finalizar la fase de **Diseño**, se debe de considerar un factor importante para el desarrollo del programa: la selección de un lenguaje de programación. Para fines prácticos y bajo cuestiones académicas, en esta versión de las notas del curso se hará uso del **lenguaje de programación C**.

Una vez seleccionado el lenguaje de programación a utilizar es posible continuar con la fase de **Codificación** del modelo en cascada. La codificación implica una traducción del pseudocódigo o diagrama de flujo a un lenguaje en específico, considerando las características propias del lenguaje.

Una observación importante es que la fase de Análisis y Diseño se desarrollan sin considerar el lenguaje de programación a utilizar, es antes del inicio de la fase de Codificación en donde se determina el tipo de lenguaje a utilizar.

Este capitulo se comenzará con una introducción al lenguaje de programación seleccionado, considerando su origen, características, la estructura general de un programa, las fase de Codificación, Compilación, Ejecución, Pruebas y Documentación en el desarrollo de un programa, así como los distintos IDE con los que es posible trabajar. Posteriormente se comenzara a mapear cada una de las características definidas en el capitulo 3 al lenguaje seleccionado, es decir, tipos de identificadores, datos, constantes y variables, estructuras de control, etc.

4.1 Introducción al Lenguaje de Programación

Lenguaje C: Historia y Características

El lenguaje de programación C es un **lenguaje de alto nivel** y de **propósito general**, conocido como un **lenguaje de programación de sistemas** por excelencia [5] [7]. Es resultado de la evolución de dos lenguajes: BCPL y B, desarrollados por Martin Richards y Ken Thompson respectivamente. Fue entonces con la publicación del libro "The C Programming Lenguaje" por Brian Kernighan y Dennos Ritchie en 1978 cuando se dio origen al Lenguaje de Programación C [5] [7].

En 1983 se desarrollo un comité (internacional) con el objetivo de hacer "una definición no ambigua del lenguaje C y que sea independiente de máquina", dicho comité lleva el nombre de **American National Standards Institute (ANSI)**. En dicho comité se determino una definición moderna y comprensible al lenguaje C, dando origen en 1988 al estándar "**ANSI**" o "**ANSI C"** [7].

El lenguaje de programación C ofrece una serie de características, entre las cuales se describen a continuación. El lenguaje C:

- Es poderoso y flexible, considera el uso de instrucciones, operaciones y funciones de biblioteca que se pueden utilizar para escribir programas.
- Es utilizado para el desarrollo de sistemas operativos, compiladores, sistemas de tiempo real y aplicaciones de comunicaciones.
- Presenta un nivel de portabilidad, es decir, un programa escrito en un Sistema Operativo (o aún en una computadora) en específico, puede ser trasladado a otros Sistema Operativo (o computadora) con pocas o ninguna modificación.

Posteriormente al origen del Lenguaje C, surgieron nuevos enfoques de programación, como por ejemplo el lenguaje orientado a objetos C++ (C con clase), o el lenguaje C# que es una evolución de C++ con propiedades de Java. Al mismo tiempo se fueron desarrollando compiladores e IDEs como por ejemplo: Visual C++, Builder C++, Borland C++, gcc de Linux (GNU C Compiler), etc.

Estructura General de un Programa en Lenguaje C

Un programa escrito usando el lenguaje C es llamado **Código Fuente**, puede hacerse uso de un editor de texto o bien de un IDE para la codificación del código que permitirá el desarrollo de un programa ejecutable. La extensión en que debe de guardarse dicho código fuente será de la siguiente manera <***NombreArchivos***>**.c**.

Un código fuente o programa fuente contiene características particulares del lenguaje C, considerando aspectos que el compilador (se encarga del análisis semántico y sintáctico del código fuente) deberá de verificar.

Las características básicas de un código fuente utilizando el lenguaje C se describen a continuación:

- ***Documentación del encabezado*** del código fuente, que debe de incluirse entre otras cosas el nombre del archivo incluyendo la extensión, una breve descripción del programa, los nombres completos del autor o autores y la fecha de creación del programa.
- ***Inclusión de librerías*** a utilizar dentro del código fuente, dichas librerías cuentan con instrucciones, funciones y valores que permiten ser utilizados a lo largo del código fuente.
- ***Función principal*** del programa fuente es nombrada **main**, dentro de esta función se codifica las instrucciones del programa a desarrollar. En el capitulo 6 se dará detalle el uso de funciones y procedimientos.

Existen otras características que a lo largo de los temas de los apuntes del curso se irán agregando a la estructura de un programa en lenguaje C.

Ejemplo "Hola Mundo con el Lenguaje C"

El objetivo de este ejemplo es mostrar como se presentan cada una de las características principales de un programa escrito en el Lenguaje C. Se muestra el pseudocódigo del programa para determinar que se ha elaborado las fases de Análisis y Diseño.

Pseudocódigo. "***HolaMundoenC***"
comienza

Escribir ("Hola Mundo con el Lenguaje C")
Escribir ("…. Esto es todo: Adiós!!!.")
termina

Código Fuente "*HolaMundo.c*"

```
/*
  Archivo: HolaMundo.c
  Descripcion: <<¨Qu, es lo que hace el programa?>>
                Este programa envia un mensaje al usuario.
  Autor: <<Nombres de los autores>>
                Luis Franco Aguilar
  Fechar: <<Fecha de creacion del archivo>>
                09 de abril de 2008
*/

/*Incluir librerias a utilizar dentro del programa*/
#include <stdio.h>  //Libreria "Standar Input Output"

/* Funcion Principal*/
int main()
{
        /*Declaracion de variables*/

        /*Cuerpo del programa*/
        printf("Hola Mundo con el Lenguaje C \n") ;
        printf("... Esto es todo: Adios!!!.");
```

```
        //Funcion que permite pausar el programa hasta que
        //el usuario presione una tecla
        getch();

        //Se¤al de finalizacion correcta del programa
        return (0);
}
```

El código presentado anteriormente presenta algunas características que se irán tratando a lo largo de las notas del curso. Cada una de las líneas del código tienen un significado lexicografico, sintáctico, semántico para el compilador, de ahí la importancia de que el código fuente cumpla con las características necesarias.

Los **comentarios** que utiliza el lenguaje C son de dos tipos, el objetivo de estos comentarios es explicar conceptos claves o proporcionar información importante sobre el código.

-Para comentarios de más de una línea de código, se hace uso de '/*' al inicio y '*/' al final del comentario. Ejemplo:

```
/* Este es un comentario de
   varias
   líneas  */
```

-Para comentarios de una sola línea de código, se hace uso de '//' al inicio y todo lo que se encuentre después de estos símbolos será considerado como comentario hasta el fin de línea o retorno de carro, muchas veces este tipo de comentario precede a una instrucción particular. Ejemplo:

```
// Este es un comentario de una sola linea
Resultado = 20 + ValorX ;     //Sumatoria de los valores….
```

La **función principal** (main) esta delimitado mediante las llaves '{' y '}', indicando el cuerpo que engloba una serie de instrucciones. El comienza esta indicado por '{' y el termina por '}'.

Las **librerías**, también conocidas como bibliotecas o archivos de cabecera (o de inclusión), son archivos con extensión .h que contiene código en C y que constituyen una serie de instrucciones predefinidas que se pueden utilizar dentro de un programa.

La librería que es utilizada por la mayoría de los programas fuentes, es **stdio.h.** Esta librería contiene información para las funciones de biblioteca que realizan operaciones de entrada y salida. Algunas otras librerías son: **math.h** para funciones matemáticas, **string.h** para el manejo de cadenas, **stdlib.h** una serie de funciones para el manejo de procesos y memoria, además de la conversión entre tipos de datos, **conio.h** para el manejo interfaz, y otras funciones de entrada y salida, etc.

Es necesario distinguir las **instrucciones** de las otras características que ofrece el lenguaje C, cada instrucción debe de estar definida en una línea de código y debe de ser finalizada con **;** (punto y coma)

En los siguientes temas del curso se irán desarrollando una serie de ejemplos y ejercicios para identificar cada una de las características del lenguaje C, tanto como las anteriormente descritas así como de nuevas características describiendo la utilidad de las mismas.

Ambientes de Desarrollo Integrado (IDE) – Turbo C

Actualmente existen una serie de IDEs que permiten desarrollar de manera eficiente programas utilizando el lenguaje de programación C, algunos de estos son: DevC++, Borlan C, TurboPascal, BuilderC, Eclipse –C, etc.

El IDE Turbo C permite la integración de algunas herramientas de programación que permiten el desarrollo adecuado de programas o aplicaciones. Una de las características importantes del Turbo C es la de ser un programa que se ejecuta en un ambiente de MS-DOS, aunque actualmente existan versiones que se ejecuten en un entorno de Windows, para fines académicos como lo es la introducción a los lenguajes de programación, el uso de Turbo C sigue siendo adecuado para el método de enseñanza-aprendizaje.

Turbo C cuenta con una serie de herramientas integradas que permiten el desarrollo de programas de manera eficiente, entre dichas herramientas se encuentran las siguientes:

- **Editor de Texto**: que permite codificar los programas siguiendo la sintaxis de lenguaje C.
- **Compilador**: verifica que los programas codificados cumplan con las reglas de sintaxis y de semántica. Una vez que el programa fuente este libre de errores de compilación, se genera un archivo con extensión ***obj***, que es un archivo que es generado por el compilador para posteriormente ser convertido a un archivo ejecutable.
- **Enlazador**: obtiene todos aquellos archivos con extensión ***obj*** que fueron generados durante la compilación y los une para formar un solo archivo ejecutable con extensión ***exe***.
- **Ambiente de Ejecución**: Es posible mandar a ejecutar ('correr') el programa generado después del enlazado. Los programas elaborados con Turbo C, que tienen una extensión exe, son ejecutados en un ambiente MS-DOS, es decir desde la línea de comandos que ofrece MS-DOS.
- **Depurador**: El depurador permite al programador identificar en donde se encuentran errores de compilación y darles el seguimiento adecuado. El depurador es una de las herramientas que el programador tendrá la necesidad de aprender a utilizarla adecuadamente, ya que con su ayuda permite rápidamente identificar errores que impiden el desarrollo completo de los programas.

4.2 Datos

4.2.1 Identificadores

Un **identificador** es una secuencia de caracteres, letras, dígitos y guiones bajos. Se definen las siguientes reglas básicas para indicar si un conjunto de caracteres es un identificador o no.

1. El primer carácter puede ser: letra o guión bajo ('_').
2. Los siguientes caracteres pueden ser letras, números o guiones bajos.
3. Los identificadores no pueden ser palabras reservadas.
4. En el lenguaje C los identificadores son sensibles a mayúsculas o minúsculas.

Una **palabra reservada** son todos aquellos conjuntos de caracteres que son predefinidas por el lenguaje de programación, en este caso C. Algunos ejemplos de palabras reservadas son: main, return, int, float, if, else, for, etc. Dichas palabras no pueden ser utilizadas como identificadores ya que presentarían inconsistencias al efectuar la compilación.

Ejemplos

"***Identificadores validos***"

Variable, contador, i, _valor_X, X_1, Y_2, A, B, C.

"***Identificadores validos***"

4_valores, **#**cinco, cin**&**co, variable x, main.

"***Identificadores sensibles a minúsculas o mayúsculas***"

El identificador **Variable** es diferente al identificador **VARIABLE**, y a su vez estos son diferentes al identificador ***VARiable***.

Aun que un identificador compuesto por un sólo carácter es valido, es importante y es considerado como buenos principios de programación, utilizar identificadores que indiquen el significado o utilidad del identificador dentro de un programa, con esto permitiría que otros programadores tengan un rápido análisis de los programas.

4.2.2 Constantes y Variables

Una **variable** en C es una localidad de memoria donde se almacena un valor (temporalmente) de un cierto tipo de dato en particular. Todas las variables dentro de un programa deben ser declaradas previamente antes de ser utilizadas, la **declaración de variables** se efectúa al inicio del bloque de la función, es decir, antes de cualquier otra sentencia o instrucción.

El **nombre de la variable** debe de ser un identificador valido, con un número de caracteres corto y que haga referencia al contexto del problema en donde se utilizara dicha variable.

Sintaxis de la declaración de variables con o sin valor inicial, utilizando el lenguaje de programación C. Los distintos tipos de datos que se definen en el lenguaje C se describen en la sección "Tipos de datos simples".

- **Declaración** de una la variable de un tipo de dato determinado, se requiere de colocar ';' al final de a instrucción.
  - Estructura:
    < tipo de variable> <identificador>;
  - Ejemplo:
    int valor_X;
    char carácter;

- **Declaración e inicialización** de una variable de tipo de dato determinado, colando el símbolo '=' y el valor inicial que será asignado a la variable; al finalizar la instrucción colocar el símbolo ';'
  - Estructura:
    < tipo de variable> <identificador> = <valor inicial> ;
  - Ejemplo:
    int valor_X = 0;
    char carácter = 'A';
    float raiz = 2.1416 ;
- **Declaración** de más de una variable de un sólo tipo de dato determinado, cada identificador de la variable deberá estar separado por ',' y el último identificador deberá de colocarse el símbolo ';'
  - Estructura:
    < tipo de variable> <identificador>, <identificador>, …. , <identificador> ;
  - Ejemplo:
    int valor_X, valor_Y, valor_Z;
    char carácter, letra, vocal;
    float raiz, cuadrado, potencia, base ;

Ejemplo "***Declaración de variables dentro del un fragmento de un código fuente***"

```
/* Función Principal*/
int main()
{
        /*Declaración de variables*/
        int valor_X = 0;
        int resultado = 0;

        /*Cuerpo del programa: sentencias o instrucciones que hacen uso de las variables*/
        /* anteriormente declaradas, NO es posible hacer uso de variables sin haberlas
declarado */
        valor_X = 15;

        resultado = valor_X * valor_X  + 3 *valor_x -30;

        //Función que permite pausar el programa hasta que
        //el usuario presione una tecla
        getch();

        //Señal de finalización correcta del programa
        return (0);
}
```

Las **constantes** son identificadores que se asocian con valores de manera estática, tienen una similitud a las variables, pero el valor de las constantes no pueden ser modificadas

En el lenguaje C se hace uso de constantes conocidas como simbólicas, utilizando la directiva **#define**. La declaración de las constantes se encuentra entre la documentación del encabezado del programa y la definición de la función principal (**main**). Para estandarizar su uso, el identificador de la constante deberá ser completamente con mayúsculas utilizando guiones bajos para separar cada palabra.

La estructura para la declaración de una constante se desarrolla de la siguiente manera:

#define <NOMBRE_MAYUSCULAS> <Valor de la constante>

En el siguiente fragmento de código se presenta la declaración de constantes considerando su valor inicial.

```
#define CADENA "HOLA MUNDO"
#define PI 3.141592
#define NUEVA_LINEA '\n'

int main ()
{
   printf("CADENA: %s \n", CADENA);
   printf("PI: %f\n", PI);
   printf("NUEVA LINEA: %c ....", NUEVA_LINEA);

   return (0);
}
```

En las sección 4.4.2 se describen cuales son los distintos formatos de impresión a pantalla utilizando la función **printf()**.

4.2.3 Tipos de Datos - Definición

El **tipo de dato** permite representar un subconjunto finito de valores, que depende del lenguaje de programación, la representación de los valores y del tamaño de espacio usado.

Los **tipos de datos simples o básicos** que se presentan en el lenguaje C son números enteros, reales (punto flotante) y caracteres.

El lenguaje C permite la creación de nuevos tipos de datos, por ejemplo el uso de **estructuras**, una estructura es una colección de uno o más tipos de elementos denominados miembros.

4.3 Tipos de Datos Simples

Los tipos de datos simples o básicos definidos para el lenguaje C son:

- **Enteros**: Incluyen tanto negativos y positivos.
- **Reales**: Números de punto flotante negativos y positivos.
- **Caracteres**: Letras, dígitos, símbolos, caracteres de escape y signos de puntuación.

4.3.1 Enteros

Se almacenan en 2 bytes (16 bits) de memoria, y de acuerdo a los modificadores de **unsigned** y **short** presenta un rango de longitud de acuerdo a la siguiente tabla:

Tipo de dato	Rango Inferior	Rango superior
int	-32,768 (-2^{15})	32,767 ($2^{15}-1$)
unsigned int	0	65,535 ($2^{16}-1$)
short int	-128 (2^{7})	127 ($2^{7}-1$)

Tabla 1.- Tabla de rango para los enteros.

4.3.2 Reales o de Punto Flotante

Se dividen en dos: **float** y **doueble**, los ***float*** se almacenan en 4 bytes (32 bits) con 7 dígitos de precisión y los ***double*** se almacenan en 8 bytes (64 bits) con 15 dígitos de precisión.

Tipo de dato	Rango Inferior	Rango superior
float	-3.4×10^{-38}	3.4×10^{-38}
double	-1.7×10^{-308}	1.7×10^{-308}

Tabla 2.- Tabla de rango para los flotantes.

4.3.3 Caracteres

Un carácter es cualquier elemento de un conjunto de carácter predefinido o alfabeto. En el lenguaje de programación C se utiliza el conjunto de caracteres ASCII (American Standard Code for Information Interchange).

Los caracteres requieren de 1 byte (8 bits) de almacenamiento en memoria, internamente los caracteres se almacenan como números:

A ➔ 65, B➔ 66, …. A➔ 97, … etc.

Al ser considerados como números es posible efectuar sumas o restas con los caracteres, sin embargo es importante considerar que el resultado de la operación se encuentre dentro del rango establecido para los caracteres. Como por ejemplo:

‘A’ + 32 = 65 +32 = 92 = ‘a’ o ‘a – 32 = 92 -32 = 65 = ‘A’

Los caracteres se manejan dentro del lenguaje como en Pseudocódigo entre comillas simples, ejemplo: ‘a’, ‘A’, ‘#’, ‘@’, etc.

Tipo de dato	Rango Inf/Sup	ASCII
char	-128 a 127	0 a 255

Tabla 3.- Tabla de rango para los caracteres.

4.3.4 Boléanos

A diferencia de otros lenguajes de programación, C **no** incorpora este tipo de dato, es decir, no existe un tipo de dato que se le pueda asignar el valor de **true** o **false**.

Sin embargo, en algunas estructuras de control como el **if** o **while** (que se verán en la sección 4.6) hacen uso de un dato de tipo **int** donde:

- **Verdadero** (true): para cualquier entero con valor distinto de 0.
- **Falso** (false): para un entero con valor igual a 0.

4.4 Sentencias Simples

4.4.1 Asignación

La asignación de un valor a una variable se desarrolla mediante el símbolo '←' en Pseudocódigo y el símbolo '=' en código. La declaración de una variable no indica que contiene un valor por default, es por ello que se requiere de una asignación para inicializar el contenido de la variable.

Pseudocódigo: <nombre de la variable> ← <valor a asignar>
Código: <nombre de la variable> = <valor a asignar del mismo tipo de la variable>;

En el lenguaje C al combinar la asignación con la declaración de variables da origen a la inicialización de la variable, la estructura de la inicialización de la variable se describe de la siguiente forma.

<tipo de dato> <nombre variable> = <valor inicial > ;
<tipo de dato> <nombre variable> = <valor inicial > , <nombre variable 2> = <valor inicial > ;

Notar que para la asignación de mas de una variable del mismo tipo, cada asignación se separa mediante una coma (,) y se finaliza con un punto y coma (;).

Definición 1.- Inicialización y Asignación

Pseudocódigo	Código
//En Pseudocódigo no hay declaración como tal. //La inicialización se desarrolla así valor_f ← 10.2325 //Inicialización de una variable. X1← 10 X2 ← 201 X3 ← 365 //La asignación de una variable previamente declarada // es semejante a la inicialización de una variable.	//Declaración de variables int valor_1, valor_2; //Inicialización float valor_F = 10.2325; //Inicialización de más de una variable del mismo tipo. int X1 = 10, X2 = 201, X3 = 365; //Asignación a un variable previamente declarada valor_1 = 66; X2 = 12214; falor_F = 3.1415681;

4.4.2 Entrada y Salida

Una de las bibliotecas definidas en C es **stdio.h** (Standard input output), es una librería que proporciona funciones para la entrada y salida de datos (información).

Salida
La salida de datos de un programa se puede dirigir a diversos dispositivos de la computadora como por ejemplo la impresora, memoria o a pantalla. También puede ser dirigida a archivos, flujos de datos por medio de canales conocidos como sockets, a elementos de persistencias como son las bases de datos, etc.

La función de la librería ***stdio.h*** que permite la salida a pantalla es llamada **printf()**. Esta función tiene como argumentos ya sea cadenas como mensajes, valores de variables o una combinación entre ellas.

Definición 2: Salida a pantalla

Pseudocódigo	Código
//Escribiendo mensajes **Escribir**(“Mensaje: Hola”) //Escribiendo contenido de variables **Escribir**(varEntera) **Escribir**(varFlotante) **Escribir**(varCaracter) //Escribir un mensaje con el contenido de una variable **Escribir**(“El valor es: ”valor) //El mensaje van entre comillas dobles, y las variables se colocan sin símbolos especiales **Escribir**(“Los valores son: ” valor1 “ , ” valor2 “ , ” valor3 “.”) **Escribir**(“Valor Cinco:” 5) **Escribir**(“Carácter:” ‘A’)	//Escribiendo mensajes **printf**(“Mensaje: Hola”); //El %d, %f y %c son formatos de salida para entero, flotante y carácter respectivamente. **printf**(“%d”, varEntera); **printf**(“%f”, varFlotante); **printf**(“%c”, varCaracter); //Mensajes completo entre comillas y seguido de las variables separadas por comas. **printf**(“El valor es: %d”, valor); //Cada formato va emparejado con la variable **printf**(“Los valores son: %d, %d, %d . ”, valor1, valor2, valor3); **printf**(“Valor Cinco: %d”, 5); **printf**(“Carácter: %c”, ‘A’);

La función **printf()** utiliza una serie de caracteres que permiten dar presentación visual de la información que se envía a la pantalla, impresora o archivo, estos caracteres son llamados **caracteres de secuencia de escape**. En la siguiente tabla se muestran los caracteres de escape utilizados en la función **printf()**.

Secuencia de escape	Significado
\a	Alarma
\b	Retroceso de espacio
\f	Avance de página
\n	Retorno de carro y avance de línea
\r	Retorno de carro
\t	Tabulación
\v	Tabulación vertical
\\	Barra inclinada
\?	Signo de interrogación
\”	Comillas dobles
\000	Número octal
\xhh	Número hexadecimal
\0	Cero, nulo (ASCII 0)

Tabla 1.- Caracteres de secuencia de escape [6].

Entrada

La entrada de datos o información que requiera utilizarse dentro de un programa puede ser de dispositivos como el teclado, un lector de códigos, dispositivos de almacenamientos, etc. Pero también puede leerse de archivos, de flujos de datos, bases de datos, etc.

La función de la librería ***stdio.h*** que permite la manipulación de datos de entrada desde teclado es llamada **scanf()**. Esta función permite la entrada formateada desde el teclado.

Definición 3: Entrada desde teclado

Pseudocódigo	Código
//Leer un sólo dato **Leer** (numEntero) **Leer**(numFlotante) **Leer**(Caracter) //Leer más de un sólo dato **Leer** (valor1 valor2)	//Leer un sólo dato. Es indispensable utilizar el & **scanf** ("%d", **&**numEntero); **scanf**("%f", **&**numFlotante); **scanf**("%c", **&**Caracter); //Leer mas de un solo dato **scanf** ("%d %f", **&**valor1, **&**valor2);

El uso del símbolo **&** (ampersand) dentro de la función **scanf**() indica que la variable se pasa por referencia, y al momento de ejecutarse la función, el dato leído se almacena dentro de la variable referenciada con el &.

El formato de salida para desplegar el contenido de variables utilizando la función printf() o bien para la lectura de la información utilizando la función **scanf()**, depende del tipo de dato en que fue declarada la variable y el formato seleccionado. En la siguiente tabla se muestran algunos de los formatos utilizados para el desplegado y lectura de información.

Formato	Descripción
%d	El datos se considera un entero
%o	El dato se considera un octal
%X	El dato se considera un hexadecimal
%u	El dato se considera un entero sin signo
%c	El dato se considera un carácter
%e	El dato se considera un flotante (float) en notación científica.
%f	El dato se considera un flotante (float) en notación decimal, con una parte entera y con signos de precisión.
%g	El dato se considera un flotante (float).
%s	El dato se considera una cadena de caracteres.
%lf	El dato se considera un double.

Tabla 2.- Formato para entrada y salida de datos [6].

Ejemplo *"Conversión de Celsius a Fahrenheit"*

Dada una temperatura en grados Celsius, efectuar la conversión de grados Celsius a Fahrenheit.

Entrada: Temperatura en grados Celsius (tempC).
Salida: Temperatura en grados Fahrenheit (tempF).
Procedimiento:

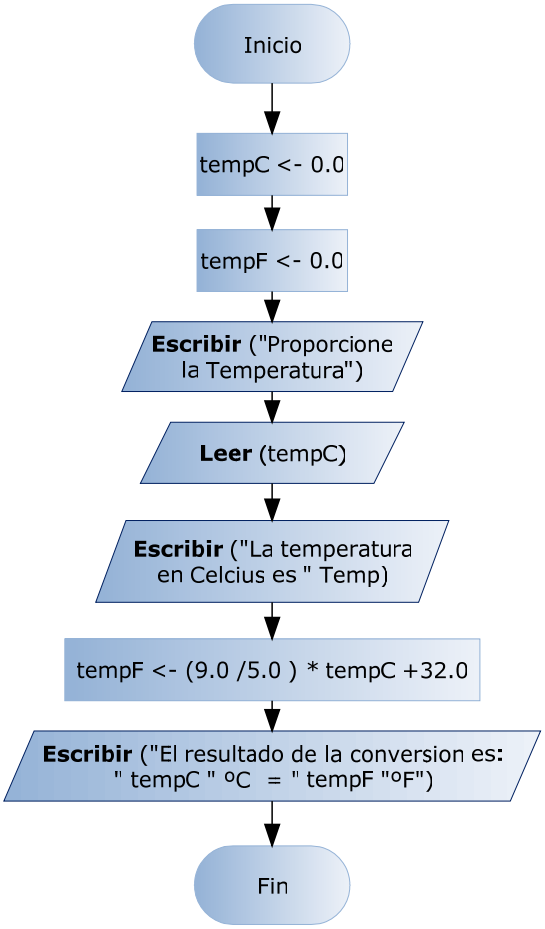


Figura 34.- Diagrama de flujo para la conversión de temperaturas.

Pseudocódigo. "ConversiónTemperatura"
comienza

tempC ← 0.0
tempF ← 0.0

Escribir ("Proporcione la temperatura:")
Leer (tempC)
Escribir ("La temperatura es:"tempC)

tempF ← (9/5)*tempC + 32

Escribir ("El resultado de la conversión es::"tempC " ºC = " tempF "ºF")
termina

Código en C

```
/*
  Archivo: ConverTemperatura.cpp
  Descripción: Este programa efectúa la conversión de temperaturas de
       Celsius a Fahrenheit. La lectura de la temperatura se efectúa
       desde teclado.
      La salida del resultado se efectúa a consola.
      Objetivo:
        Utilización de funciones de entrada y salida: printf() y scanf()
  Autor: Basurto Páez Gustavo
  Fecha: 15 de abril de 2008
*/

/*Incluir librerías utilizadas en el programas*/
#include <stdio.h>

int main(){
    /*Declaración (e inicialización) de variables*/
    float tempC = 0.0;
    float tempF = 0.0;

    printf(" Programa de Conversión de Temperatura \n");
    printf("Proporcione la temperatura: " );

    //Lectura de la temperatura, con formato de flotante : %d
    scanf("%f", &tempC);

    printf(" La temperatura en Celcius es: %f ºC y en Fahrenheit %f
             ºF\n", tempC, tempF);

    //Calculo de la temperatura
    tempF = (9.0/5)*tempC + 32;
      //NOTA: si la división no es flotante genera otro resultado


    printf(" El resultado de la conversión es: %f §C = %f \n", tempC,
tempF);

    getch();
    return (0);
}
```

4.5 Construcción de expresiones aritméticas y lógicas.

Expresiones aritméticas

*	Multiplicación	Observaciones:
/	División	-Evaluación de Izquierda a derecha.
+	Suma	-Los paréntesis no son productos:
-	Resta	$(7)(6) \neq (7)*(6)$
%	Modulo	
( )	Agrupar	

Ejemplo: Expresiones aritméticas:

Expresión Matemática	En C
$\frac{1}{3+4^2}$	1 / (3 + 4 * 4)
$3\left(\frac{4}{5}\right)$	3 * 4/5 3 * (4/5)
3(4)+5(-6)+(8)(3+5)	(3*4) + 5*(-6) + (8)*(3+5)
$(20-5) \bmod 5 + 2(4) \bmod 3 - \frac{1}{10}(8 \bmod 7)$	(20-5) % 5 + 2 * 4 % 3 – 1/10 * (8 % 7)

Ejemplo *"Evaluar Formula"*

Evaluar la expresión en C, el resultado de la siguiente fórmula:

$$\text{resultado} = \frac{(x^2 - a^2)(b + c)}{z^2 + 1} \quad \text{donde } a=6.0,\ b=0.5,\ c=b,\ x=-1.0,\ z=a+c.$$

Codigo en C

```
/****
*/
#include <stdio.h>

int main(){
    /*Declaración e inicialización de variables*/
    float a, b, c, x, z;
    float resultado = 0.0; //resultado de la ecuación

    a = 6.0;
    b = 0.5;
    c = b;
    x = -1.0;
    z = a + b;

    /*Evaluación del resultado*/
    resultado = ( x*x - a*a )* ( b + c) / ( z*z - 1);

    printf(" \t El resultado es: %f ", resultado);
    getch();

    return (0);
}
```

Operadores Relacionales y Operadores Lógicos

Operadores Relacionales		
Descripción	**Pseudocódigo**	**En C**
Menor que	<	<
Mayor que	>	>
Igual	=	== "dos = juntos"
Menor igual que	≤	< =
Mayor igual que	≥	> =
No igual (distinto)	≠	! =

Operadores Lógicos		
Descripción	**Pseudocódigo**	**En C**
Conjunción (and, ∧)	y	&& (ampersand)
Disyunción (or, ∨)	o	|| (pipe)
Negación (not, ¬)	no	!

Tabla de Verdad				
Valor	**Valor**	**Conjunción**	**Disyunción**	**Negación**
p	Q	p ∧ q	p ∨ q	¬ p
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

Ejemplo

Referente a de operadores lógicos y operadores relacionales.

Pseudocodigo "***OperLogicosyRelacionales***"

comienza

a ← 3

b ← 4

Si (a ≤3 **Y** b ≠ 2) **entonces** //El resultado de la condición es "Verdadero"

Escribir ("es verdadero")

Si ((a ≤3 **Y** b ≠ 2)**Y** a = b) **entonces** //El resultado de la condición "Falso"

Escribir ("es verdadero")

Si ((a ≤3 **Y** b ≠ 2) **O** a = b) **entonces** //El resultado de la condición "Verdadero"

Escribir ("es verdadero????")

Otro

Escribir ("es falso???")

termina

4.6 Estructuras de Control.

Las instrucciones básicas y comunes a casi todos los lenguajes de programación se pueden condensar en cuatro grupos:

1. **Instrucciones de entrada y salida**. (de transferencia de información y datos).
2. **Instrucciones aritméticas y lógicas.**
3. **Instrucciones selectivas.**
4. **Instrucciones repetitivas.**

La clasificación de las estructuras de control se define en **tres estructuras básicas**:

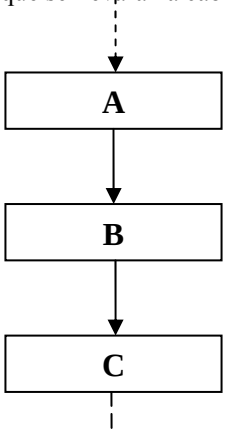
- **Secuenciación**: Ejecución sucesiva de pasos, tareas, acciones o instrucciones.
- **Selección**: Dependiendo de una condición dada, se determina si se efectúa una serie de acciones o se efectúa otra serie distinta.
- **Iteración**: Repetición de una serie de instrucciones u operaciones, mientras se cumpla una condición proporcionada.

Pseudocódigo

Es una forma de expresar los algoritmos, de una forma 'natural', utilizando un lenguaje específico, y que utiliza las 3 estructuras básicas de la programación estructurada.

4.6.1 Secuenciación.

La secuenciación puede notarse con el uso de diagramas de flujo, por ejemplo para las instrucciones A, B y C, la secuenciación indica el orden en que se llevaran a cabo las instrucciones:



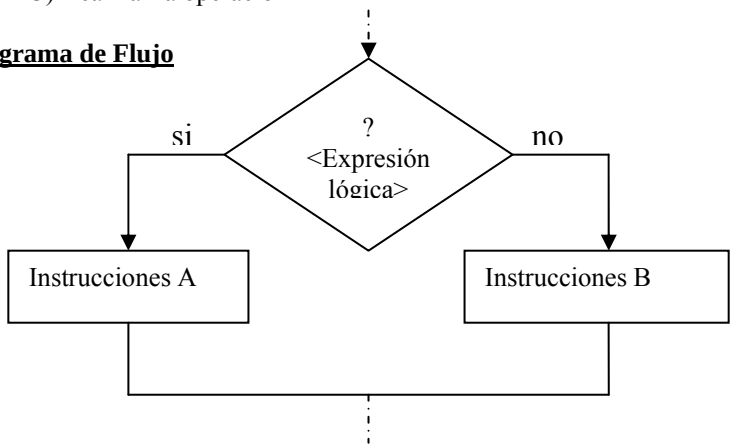
4.6.2 Selección condicional simple, doble y múltiple.

Selección condicional simple y doble

Algotirmo

1) Evaluar **Si** condición **entonces** ir a paso 2 **otro** ir a paso 3
2) Realizar la operación A
3) Realizar la operación B

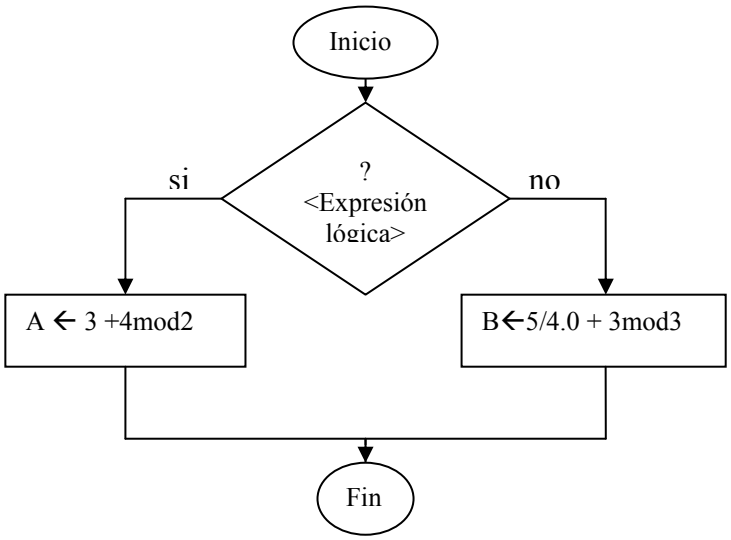
Diagrama de Flujo



Definición 3: Selección condicional (Si… entonces….. otro)

Pseudocódigo	En C
Si <condición > **entonces** comienza <instrucciones A> termina **otro** comienza <instrucciones B> termina	**if** (<condición >) { /**instrucciones A**/ } **else** { /**instrucciones B**/ }

Ejemplo *"Uso de la selección condicional*"



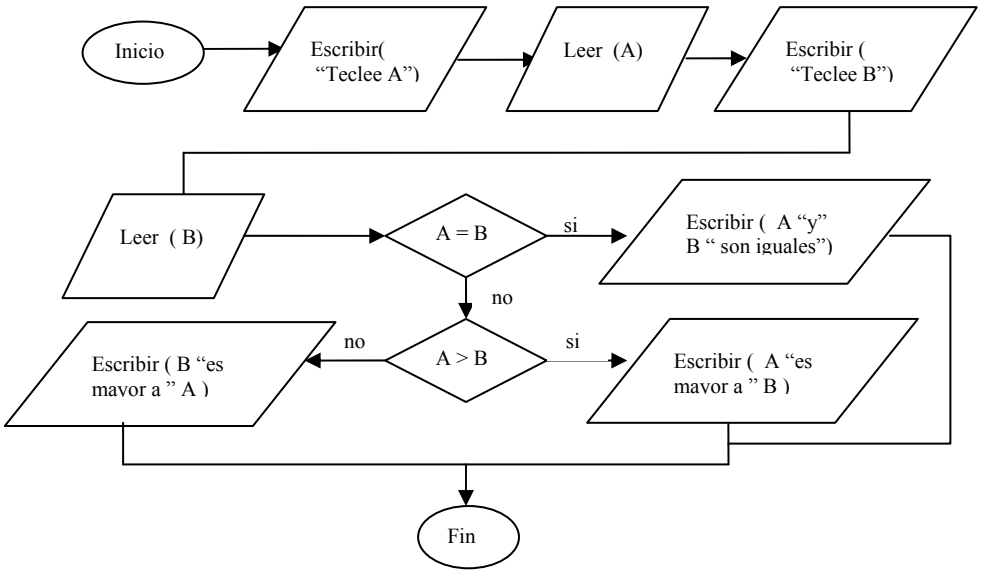
Pseudocódigo	Código C
Pseudocódigo "Condicional" **comienza** **Si** (<expresión lógica >) **entonces** **com** A ← 3 + 4 mod 2 Escribir ("Instrucción A:" A) **term** **Otro** **com** B ← 5 / 4.0 + 3 mod 3 Escribir ("Instrucción B:" B) **term** **termina**	/**"Condicional.c"**/ #include <stdio.h> int main() { **if**(<expresión lógica>) **{** A = 3 +1 %2; printf("Instrucciones A %d", A); **}** **else** **{** B = 5 /4.0 + 3 % 3; printf("Instrucciones B %d", B); **}** }

Ejemplo "*Uso de la selección condiciona anidada*"
Dado dos números enteros positivos (Numa y NumB), determinar si son **IGUALES** o en otro caso cual es el **MAYOR** de los dos

a) **Algoritmo** identificando las entradas y salidas, no es necesario desarrollar el procedimiento.

Entrada: Dos números enteros positivos (Numa y NumB),
Salida: Mensaje indicando si son iguales, o en otro caso el mayor de los dos números.

b) **Diagrama de flujo**.



c) **Pseudocódigo**.

Pseudocódigo "MayorOIgual"
Comienza

```
Escribir (" Teclee valor de A")
Leer ( valA )
Escribir (" Teclee valor de B")
Leer (valB )

Si ( A =  B ) entonces
com
    Escribir (valA "y:" valB "son iguales.")
term
Otro
com
        Si ( A >  B ) entonces
        com
          Escribir (valA " es mayor a" valB )
        term
        Otro
        com
          Escribir (valB " es mayor a" valA)
        term
term
```

Termina

d) **Código en C.**

```
/*
  Archivo: MayorOIgual.c
  Descripción: Este programa efectúa la comparación de dos números  enteros positivos,
       envía a consola un mensaje indicando si los dos números son iguales, o cual es el
       mayor.
```

```
  Autor: Basurto Páez Gustavo
  Fecha: 10 de octubre de 2007
*/
/*Incluir librerias */
#include <stdio.h>

//Función principal
int main()
{
    //Declaraci¢n de variables
    int valorA, valorB;

    printf(" Teclee el valor de A: ");
    scanf("%d", &valorA );
    printf(" Teclee el valor de B: ");
    scanf("%d", &valorB );

    if( valorA == valorB )
    {
       printf(" %d y %d son iguales.", valorA, valorB );
    }
     else
     {
       if ( valorA >  valorB ) //if anidado
       {
              printf ( "%d es mayor a %d ", valorA, valorB );
       }
       else
       {
              printf ( "%d es mayor a %d ", valorB, valorA );
       }
    }
    getch();
    return (0);
 }
```

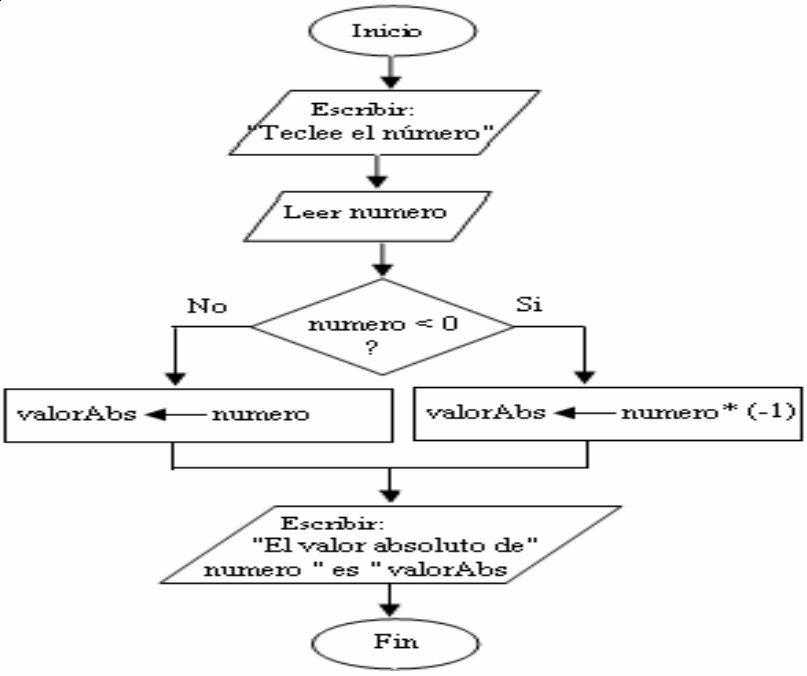
Ejemplo "Valor absoluto de un n*úmero entero*"

Calcular el valor absoluto de un número entero (incluye al cero). Ejemplo: |-10 | = 10, | 5 | = 5

Entrada: Un número entero numero.

Salida: Mensaje indicando el valor absoluto de numero.

Diagrama de Flujo



Pseudocodigo "***ValorAbsoluto***"
Comienza
Escribir ("Teclee el número entero: ")
Leer (numero)

Si (numero < 0) **entonces**
com
valorAbs ← numero * (– 1)
term
Otro
com
valorAbs ← numero
term
Escribir ("El valor absoluto de "numero " es " valorAbs)
Termina

Código C

```
/*
  Archivo: ValorAbsoluto.c
  Descripcion: Este programa calcula el valor absoluto de un numero entero.
  Autor: Basurto Páez Gustavo
  Fecha: 10 de octubre de 2007
*/
/*Incluir librerias utilizadas en el programas*/
#include <stdio.h>

int main()
{
    /*Declaracion e inicializacion de variables*/
    int numero;
    int valorAbs = 0 ;

    printf("\n Teclee el numero entero: ");
    scanf("%d", &numero);

    if ( numero < 0  )
    {
       valorAbs = numero * (-1);
    }
    else
    {
       valorAbs = numero;
    }
    printf(  "\n El valor absoluto de %d  es %d .\n", numero , valorAbs );

    getch();
    return (0);
}
```

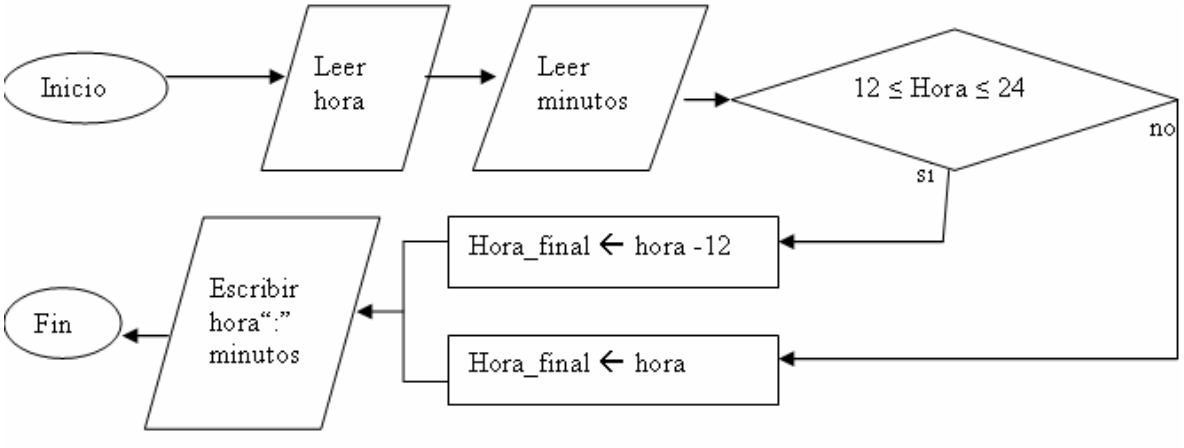
Es posible simplificar la resolución del problema de la obtención del valor absoluto de un número entero. A continuación se describe el pseudocódigo.

Pseudocodigo "ValorAbsSimpli"
Comienza
Escribir ("Teclee el número entero: ")
Leer (numero)
valorAbs ← numero
Si (numero < 0) **entonces**
com
valorAbs ← numero * (– 1)
term
Escribir ("El valor absoluto de "numero " es " valorAbs)
Termina

Ejemplo "Conversión de formatos de horas"

Convertir una hora dada en formato de 24 hrs., al formato de 12 hrs. Ejemplo: 23:10 hrs. → 11:10 hrs.

Entrada: Hora en formato de 24 hrs. hora:minutos (ambos enteros). hora y minutos.
Salida: Hora en formato de 12 hrs. hora:minutos
Diagrama de Flujo



Pseudocodigo "Conversionhoras"
Comienza

Escribir ("Teclee Hora: ")
Leer (hora)
Escribir ("Teclee minutos: ")
Leer (minutos)

Si (12 < hora ≤ 24) **entonces**
com
Hora_final ← hora – 12
term
Otro
com
Hora_final ← hora
term
Escribir ("la hora es:" Hora_final":"minutos)

Termina

Código C

```
/*
  Archivo: ValorAbsoluto.c
  Descripcion: Este programa calcula el valor absoluto de un numero entero.
  Autor: Basurto Páez Gustavo
  Fecha: 10 de octubre de 2007
*/
/*Incluir librerias utilizadas en el programas*/
#include <stdio.h>

int main()
{
    /*Declaracion e inicializacion de variables*/
    int numero;
    int valorAbs = 0 ;

    printf("\n Teclee el numero entero: ");
    scanf("%d", &numero);

    if ( numero < 0  )
    {
```

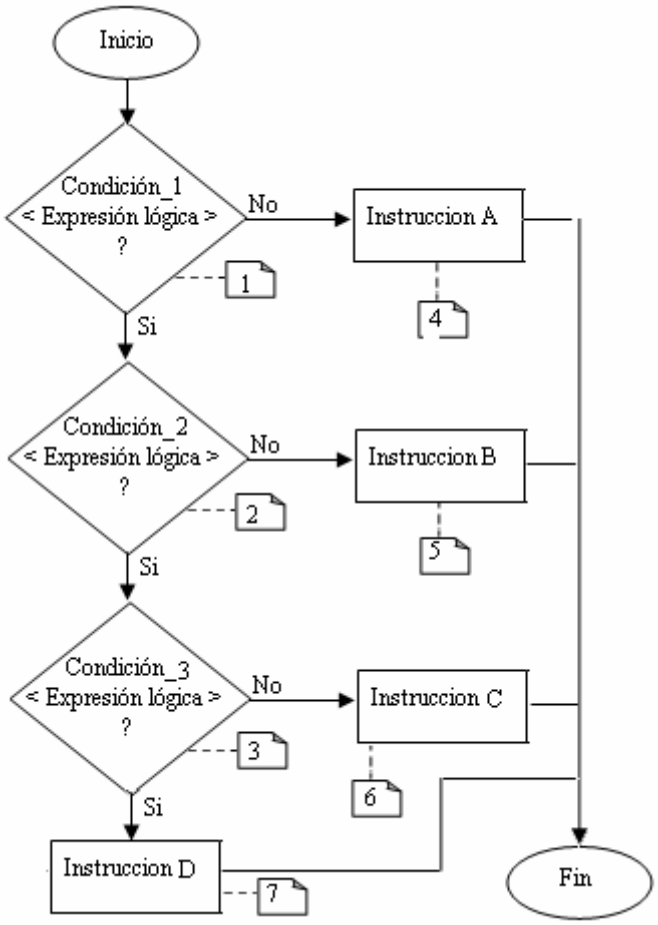
```
      valorAbs = numero * (-1);
   }
   else
   {
      valorAbs = numero;
   }
   printf(  "\n El valor absoluto de %d  es %d .\n", numero , valorAbs );

   getch();
   return (0);
}
```

Selección condicional múltiple (estructuras anidadas - 'if-anidados')

Algotirmo

1) Evaluar **Si** condición_1 **entonces** ir a paso 2 **otro** ir a paso 4
2) Evaluar **Si** condición_2 **entonces** ir a paso 3 **otro** ir a paso 5
3) Evaluar **Si** condición_3 **entonces** ir a paso 6 **otro** ir a paso 7
4) Realizar la operación A.
5) Realizar la operación B.
6) Realizar la operación C.
7) Realizar la operación D.

Diagrama de Flujo



Definición 4: Selección condicional múltiple (sentencias anidadas)

Pseudocódigo	En C
Si (<condición_1 >) **entonces** comienza **Si** (<condición_2 >) **entonces** comienza **Si** (<condición_3 >) **entonces** comienza <instrucción D> termina **otro** comienza < instrucción C> termina termina **otro** comienza < instrucción B> termina **otro** comienza < instrucción A> termina	**if** (<condición_1>) { **if** (<condición_2>) { **if** (<condición_3 >) { /*instrucción D*/ } **else** { /*instrucción C*/ } } **else** { /*instrucción B*/ } **else** { /*instrucción A*/ }

Ejemplo "Cinco primeros números primos"

Dado un número entero positivo determinar si se encuentra entre los 5 primeros números primos (2, 3, 5, 7, 11).

Entrada: Un numero entero positivo.

Salida: Mensaje si pertenece o no pertenece a los primeros 5 números primos.

Pseudocodigo "***AnalisiNumPrimos***"

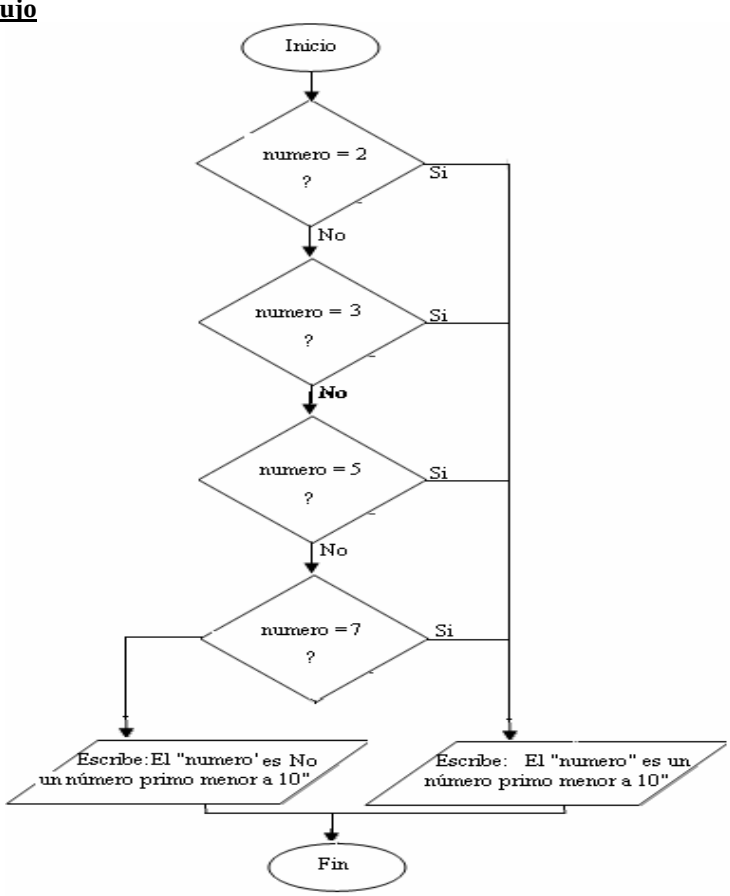
```
Comienza
        Escribir ("Teclee numero a determinar si es primo o no: ")
        Leer (numero)
        Si ( numero = 2) entonces
        com
                Escribir ("El número " numero " es primo")
        term
        Otro
        com
                Si ( numero = 3) entonces
                com
                        Escribir ("El número " numero " es primo")
                term
                Otro
                com
                        Si ( numero = 5) entonces
                        com
                                Escribir ("El número " numero " es primo")
                        term
                        Otro
                        com
                                Si ( numero = 7) entonces
                                com
                                   Escribir ("El número " numero " es primo")
                                term
                                Otro
                                com
                                   Escribir ("El número " numero " NO es primo menor a 10")
                                term
                        term
```

term

term

termina

Diagrama de Flujo



Código C

```
/*
 Archivo: AnalisiNumPrimos.c
 Descripción: Este programa que determina si un número entero positivo
              pertenece a los primos, considerando solo los valores siguientes:  2, 3, 5,
7
  Autor: Basurto Páez Gustavo
  Fecha: 10 de octubre de 2007
*/

/*Incluir librerias utilizadas en el programas*/
#include <stdio.h>
int main()
{
    /*Declaracion e inicializacion de variables*/
    int numero;

    printf("\n Teclee el numero entero positivo: ");        scanf("%d", &numero);

    if ( numero == 2)
    {
       printf ("El numero %d es primo", numero );
    }
    else
    {
       if (numero == 3)
       {
              printf( "El numero %d es primo", numero);
       }
       else
```

```
    {
          if ( numero == 5)
          {
                printf("El numero %d  es primo", numero);
          }
          else
          {
                if ( numero ==  7)
                 {
                      printf("El numero %d  es primo",       numero);
                 }
                else
                 {
                      printf("El numero %d NO es primo menor a 10", numero);
                 }
          }
       }
  }
  getch();
  return (0);
}
```

Selección condicional múltiple (switch)

La sentencia **swicth** es una sentencia en C que es utilizada para seleccionar entre múltiples alternativas, principalmente cuando la selección se basa en el valor de una variable (o una expresión de control o selector). El valor de la expresión deberá ser de tipo carácter o entero (de tipo **char** o **int** en C, respectivamente), pero no del tipo flotante (**float** o **double**).

Definición 5: Selección condicional múltiple (switch)

Pseudocódigo	En C
Si (<condición_1>) **entonces** comienza <instrucción A> termina **Otro** comienza **Si** (<condición_2>) **entonces** comienza <instrucción B> termina **Otro** comienza **Si** (<condición_3>) **entonces** comienza <instrucción C> termina **Otro** comienza **Si** (<condición_4>) **entonces** comienza <instrucción D> termina **Otro** comienza <instrucción E> termina termina termina termina	//Variable puede ser **char** o **int** **switch**(<Contenido de Variable>) { **case** (<valor_1>) : { /*instrucción A */ **break**; //Se va a **Z** } **case** (<valor_2>) : { /*instrucción B */ **break**; //Se va a **Z** } **case** (<valor_3>) : { /*instrucción C */ **break**; //Se va a **Z** } **case** (<valor_4>) : { /*instrucción D */ **break**; //Se va a **Z** } **default** : { /*instrucción E */ } ////Se va a **Z** } **/**instrucción Z****/

La sentencia **case** indica el 'caso' del resultado posible de la valor evaluado en el **switch**. Cuando ninguno de los **case**s se cumple, la sentencia **default** se ejecuta, es decir, ya no se considera el valor y se ejecuta cuando no se cumpla en ninguno de los **case**s anteriores.

La instrucción **break**, es utilizada en C para indicar la finalización de la sentencia del **case**, y del **switch.** Del ejemplo de código en C descrito en la tabla anterior, después de ejecutarse el las *instrucción X (A, B, C ó D)*, la instrucción **break** finaliza la acción y se pasa inmediatamente a ejecutar la instrucción Z (después del **switch**).

Ejemplo "Cinco primeros números primos con switch"

Del problema descrito anteriormente se resolvió con selección condicional múltiple, con sentencias anidadas: "Dado un número entero positivo determinar si se encuentra entre los 5 primeros números primos (2, 3, 5, 7, 11)."

NOTA: El Pseudocódigo se queda intacto, **switch** es una sentencia en los lenguajes de programación y NO se debe mapearse con pseudocódigo.

Código C

```
/*
  Archivo: AnalisiNumP_switch.c
  Descripción: Programa similar a AnalisiNumPrimos.c, haciendo uso de la sentencia
switch.
  Autor: Basurto Páez Gustavo
  Fecha: 10 de octubre de 2007
*/
#include <stdio.h>

int main()
{
    /*Declaración e inicialización de variables*/
     int numero;
     printf("\n Teclee el numero entero positivo: ");
     scanf("%d", &numero);

     /*La estructura de control switch acepta valores de tipo int o char */
     switch( numero )
     {
             case (2) :
                 {
                      printf ("El numero %d es primo", numero );
                      break;
                 }
             case (3) :
                 {
                      printf ("El numero %d es primo", numero );
                      break;
                 }
             case (5) :
                 {
                      printf ("El numero %d es primo", numero );
                      break;
                 }
             case (7) :
                 {
                      printf ("El numero %d es primo", numero );
                      break;
                 }
             default :
                {
                printf("El numero %d NO es primo menor a 10", numero);
                }
      } //fin switch
    getch();
    return (0);
}
```

4.6.3 Iteración Condicional.

Un **ciclo** (bucle) representa una sección del programa que **repite** una serie de instrucciones (o una sola instrucción), por un número determinado de veces de acuerdo a la validación de una **condición**. La condición es determinada por una ***expresión lógica o booleana***, que permite controlar la secuencia de repetición de la estructura.

Las dos partes principales de un ciclo, son:

-**Cuerpo del ciclo**: representa el conjunto de instrucciones que se repetirán de acuerdo a una condición. Es posible considerar dentro del cuerpo del ciclo otro tipo de estructuras, como las de sección condicional o de iteración),

-**Iteración**: representa cada una de las repeticiones del cuerpo del ciclo.

De acuerdo a la condición de la estructura condicional, se determina el número de iteraciones que se efectuaran sobre el cuerpo del ciclo.

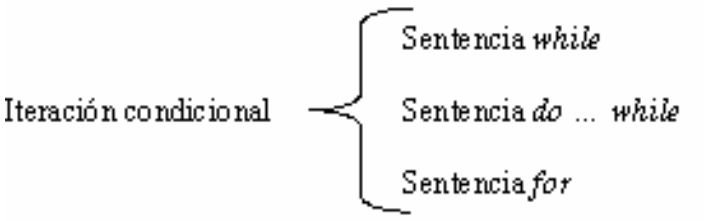
Es posible que se efectúe:

-***Ninguna iteración***: en caso de que la condición de la iteración no se cumpla por completo.

-***Una sola vez:*** considerando que se efectúa la primera iteración y en la siguiente ya no se cumpla la condición.

-***Por un tiempo indeterminado (ciclo infinito)***: esto ocurre cuando siempre se cumple la condición y no se especifica como finalizar las iteraciones.

La iteración condicional se divide en tres estructuras de control iterativo, mismas que se mientras en el diagrama y que se explicarán en las siguientes secciones del capítulo.



4.6.3.1 Sentencia While (mientras)

Definición 6: Sentencia iterativa While (Mientras)

Pseudocódigo	En C
mientras <condición_ciclo > **comienza** <instrucciones A> ….. <instrucciones N> **finMientras**	**while** (< condición_ciclo >) { /**instrucciones A**/ ……. /**instrucciones N**/ } /*Fin Mientras*/

Ejemplo "Número de Positivos Tecleados"

El usuario tecleara 10 números positivos y el programa deberá de ir contando la cantidad de números positivos tecleados.

Entrada: Una serie de números enteros (**NEntero**).

Salida: Mensaje indicando el números de valores positivos tecleados por el usuario (**NPositivos**).

Pseudocódigo *"ContPositivos"*

```
comienza
        NEntero ← 0
        NPositivos ← 0
        Cont ← 0

        mientras ( Cont < 10 )
        comienza

            Escribir ("Teclee un número entero positivo")
            Leer ( NEntero)

            Si ( NEntero > 0 ) entonces
            comienza
                NPositivos ← Npositivos + 1
            termina

            Cont ← Cont +1
        termina
        Escribir ("El número de valores positivos tecleados es:" NPositivos)
termina
```

Código en C

```
/*
  Archivo: NumerosPositivos.c
  Descripción: Este programa lee desde teclado un numero entero  y manda a pantalla los
       siguientes 10 numeros enteros.
      Objetivo:
      Utilización de la estructura de control - Iteracion Condicional
      -Sentencia while (mientras)
      Observación:
      Si es necesario haga una "Tabla de seguimiento" para verficar que es validad la
       condicion numEntero < fin, y no numEntero <= fin
  Autor: Basurto Páez Gustavo
  Fecha: 24 de abril de 2008
*/
/*Incluir librerias utilizadas en el programas*/
#include <stdio.h>

int main()
{
    /*Declaracion (e inicializacion) de variables*/
```

```
    int NEntero = 0 ;
    int NPositivos = 0 ;
    int Cont = 0;

    printf("\t Programa Números Positivos \n\n ");


    //Uso de Iteracion condicional - while (mientras)
    while (  Cont < 10)
    {
       printf(" Proporcione un número entero: " );
       scanf("%d", &NEntero);

       if(NEntero > 0 )
       {
             NPositivos = NPositivos  + 1;
       }
        Cont = Cont + 1;

    }//fin while

    printf("El número de valores positivos tecleados es: %d", NPositivos);
    getch();
    return (0);
}
```

Evidencias (Ejecución del programa)

Programa Numeros Positivos
Proporcione un número entero: 10
Proporcione un número entero: -2
Proporcione un número entero: 0
Proporcione un número entero: -36
Proporcione un número entero: 5
Proporcione un número entero: -45
Proporcione un número entero: 2
Proporcione un número entero: 0
Proporcione un número entero: -2
Proporcione un número entero: -10
El número de valores positivos tecleados es: 3

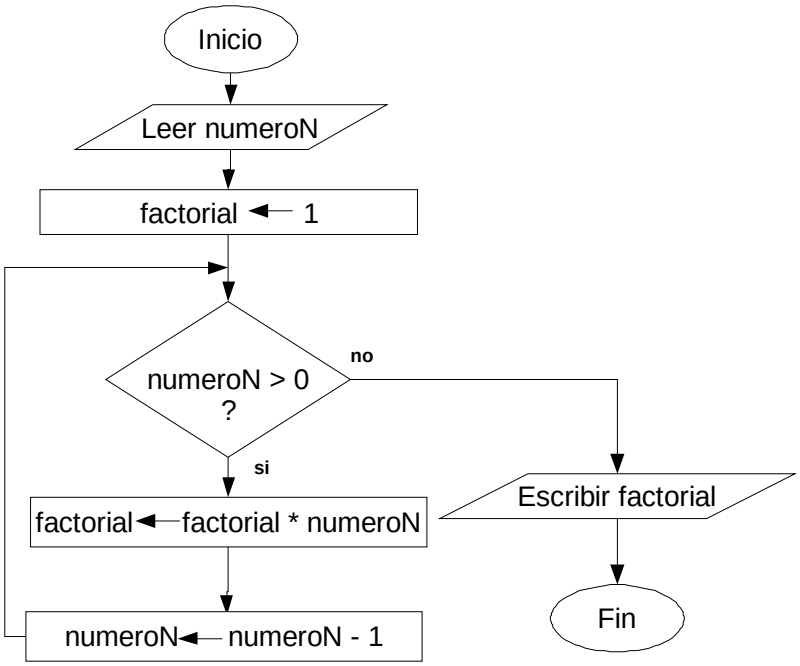
Ejemplo "Factorial de un Número"

Calcular el factorial de un número entero positivo .Considerar los casos iniciales factorial (0) = 1 y factorial (1) = 1. Ejemplo: factorial (2) = 2*1, factorial (3) = 3*2*1 = 6, factorial (4) = 4*3*2*1 =12, etc.

Entrada: Número entero (numeroN).
Salida: Mensaje indicando el factorial de numeroN.
Procedimiento:

1) Leer un entero positivo asignándole a la variable: numeroN.
2) Inicializar la variable factorial a uno.
3) Evaluar si numeroN es mayor a cero, si es así ir al paso 4, en otro caso ir al paso 7.
4) Asignar a factorial el resultado de multiplicar factorial por numeroN.
5) Decrementar en una unidad el contenido de la variable numeroN.
6) Regresar el paso 3
7) Escribir el mensaje indicando el factorial.

Diagrama de Flujo



Pseudocódigo. "***factorial***"

comienza

Escribir ("Proporcione un número entero positivo:")
Leer (numeroN)
factorial ← 1

mientras (numeroN > 0)
comienza
factorial ← factorial * numeroN
numeroN ← numeroN - 1
finMientras
Escribir (factorial)

termina

Código en C

```
/*
  Archivo: FactorialDecrem.c
  Descripción: Este programa lee desde teclado un número entero positivo
               efectúa el calculo el factorial del numero tecleado.
               Esta versión hace el calculo decrementando el numero inicial
                    n * (n -1 ) .... 5 * 4 * 3 * 2 * 1.
             Objetivo:
              Utilización de la estructura de control - Iteración Condicional
              -Sentencia while (mientras)
  Autor: Basurto Páez Gustavo
  Fecha: 13 de octubre de 2007
*/

/*Incluir librerías utilizadas en el programas*/
#include <stdio.h>

int main()
{
    /*Declaración (e inicialización) de variables*/
    int numeroN = 0 ;
    int factorial  = 1;

    printf("\t Programa : Factorial \n\n ");
    printf(" Proporcione un numero entero positivo: " );
```

```
    scanf("%d", &numeroN);

    printf("\n\n Se efectuara el calculo del factorial del numero %d, factorial ( %d )=
             ", numeroN ,numeroN );

    //Uso de Iteracion condicional - while (mientras)
    while (  numeroN > 0 )
    {
          factorial = factorial * numeroN;
          numeroN = numeroN - 1;
    }//fin while
    printf(" %d ", factorial);

    getch();
    return (0);
}
```

Evidencias

```
         Programa : Factorial

  Proporcione un numero entero positivo: 6
 Se efectuara el calculo del factorial del numero 6, factorial ( 6 )=  720
```

Ejemplo "Series de Fibonacci"

Escribir la serie de Fibonacci especificando el número (entero positivo) de iteraciones a escribir. A continuación se especifica la regla para especificar la serie de Fibonacci.

- $f_i = f_{i-2} + f_{i-1}$
- $f_0 = 0$
- $f_1 = 1$

Ejemplo: Analizar las siguientes series de Fibonaci.

$f_2 = f_0 + f_1 = = 0 + 1 = 1$, $f_3 = f_1 + f_2 = 1 + 1 = 2$, $f_4 = 3$, $f_5 = 5$, $f_6 = 8$, …

0, 1, 1, 2, 3, 5, 8, 13, …

Entrada: Número entero (numeroN), indicando el tamaño de la serie de Fibonacci.
Salida: Mensaje indicando la serie de Fibonacci.

Diagrama de Flujo (por desarrollar)

Pseudocódigo. "***Fibonacci***"

comienza

Escribir ("Proporcione el tamaño de la serie de Fibonnaci")
Leer (numeroMax)

valor_i ← 2
f_imenos1 ← 1
f_imenos2 ← 0

Escribir (f_imenos2 "," f_imenos1)

mientras (valor_i < numeroMax)
comienza
f_i ← f_imenos2 + f_imenos1

Escribir ("," f_i)

f_imenos2 ← f_imenos1
f_imenos1 ← f_i
valor_i ← valor_i + 1
finMientras
Escribir (factorial)

termina

Código en C

```
/*
  Archivo: Fibonacci.c
  Descripción: Este programa despliega la serie de Fibonacci, de un tamaño
               leido desde teclado, dicho valor es un entero positivo. Valores:
                 -    fi = fi - 2 + fi - 1
                 -    f0 = 0
                 -    f1 = 1
                  Ejemplo: Analizar las siguientes serie de Fibonaci
            f2 = f0 + f1 =  = 0 + 1 = 1,     f3 = f1 + f2 =  1 + 1 = 2  ,       f4 = 3 ,
               f5 = 5, f6 = 8, …
                      0, 1, 1, 2, 3, 5, 8, 13, …
               Objetivo:
                Utilizacion de la estructura de control - Iteración Condicional
                -Sentencia while (mientras)
  Autor: Basurto Páez Gustavo
  Fecha: 14 de octubre de 2007
*/

/*Incluir librerias utilizadas en el programas*/
#include <stdio.h>

int main()
{
    /*Declaracion (e inicializacion) de variables*/
    int numeroMax = 0 ;
    int valor_i = 2;
    int f_imenos_1 = 1;
    int f_imenos_2 = 0;
    int f_i = 0;

    printf("\t Programa : Serie de Fibonacci \n\n ");
    printf("Proporcione el tamaño de de la serie de fibonaci (numero entero positivo):");
    scanf("%d", &numeroMax);

    printf("\n\n La Serie de Fibonacci de tamaño %d se describe a continuación: \n",
            numeroMax);

    printf (" %d, %d", f_imenos_2, f_imenos_1);
```

```
    //Uso de Iteracion condicional - while (mientras)
    while (  valor_i < numeroMax )
    {
          f_i = f_imenos_2 +  f_imenos_1;
          printf(", %d", f_i);

          f_imenos_2 = f_imenos_1;
          f_imenos_1 = f_i;
          valor_i =  valor_i + 1;
    }//fin while
    getch();
    return (0);
}
```

Evidencias

```
        Programa : Serie de Fibonacci

Proporcione el tama±o de de la serie de fibonaci (numero entero positivo): 10

 La Serie de Fibonacci de tama±o 10 se describe a continuación:
 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

4.6.3.2 Sentencia Do…. While (Hacer … Mientras)

La sentencia ***do-while*** presenta una similitud con el ciclo condicional – ***while*** (mientras), con la diferencia que **se ejecuta siempre al menos una vez**. La condición del ciclo ***while*** se evalúa al inicio del ciclo, mientras que la condición de ***do-while*** se evalúa al final. Es útil cuando se requiere que se desarrolle al menos una vez el cuerpo del ciclo.

Definición 7: Sentencia iterativa do… while (hacer … Mientras)

Pseudocódigo	En C
hacer **comienza** <instrucciones A> ….. <instrucciones N> **termina** **mientras** <condición_ciclo >	**do** { /**instrucciones A**/ ……. /**instrucciones N**/ } **while** (< condición_ciclo >) **;**

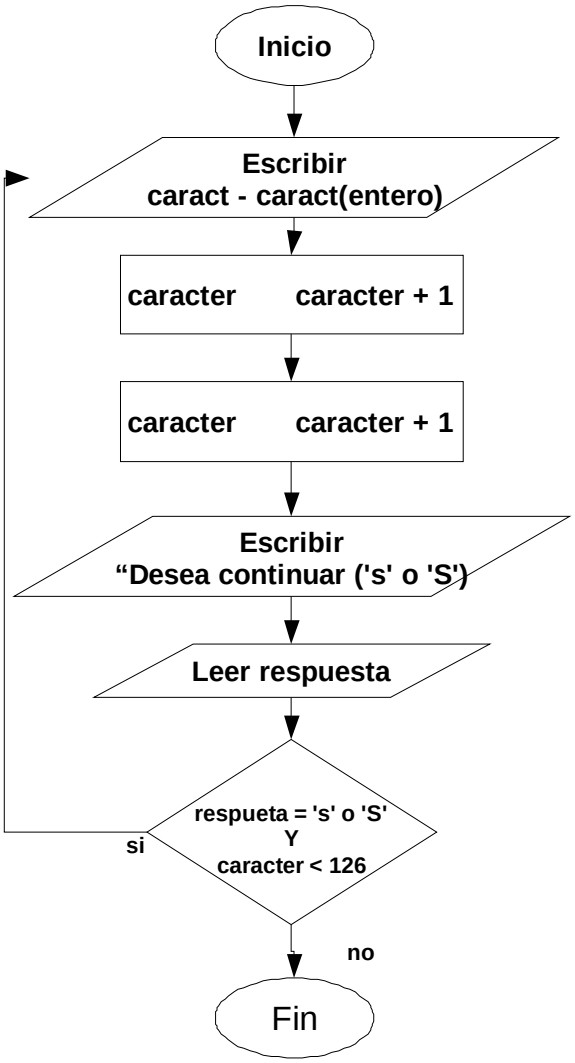
Ejemplo "Pregunta Continuación"

Elaborar un programa que imprima los caracteres ASCII junto con el valor entero (entre el rango del 32 ' ' al 126 '~'). Para cada escritura en pantalla, preguntar si se desea continuar. El usuario tecleara un de los caracteres de acuerdo a las decisión que tome: 's' ('S') o 'n' ('N').

Entrada: caracteres que representan las respuestas del usuario 's' o 'n', 'S' o 'N'.
Salida: Los caracteres ASCII.
Procedimiento:

1) Escribir le primer carácter ASCII y en entero.
2) Incrementar en uno el valor del carácter_ascii en uno.
3) Escribir al usuario si desea continuar 's' o 'S'.
4) Evaluar si respuesta es igual a 's' **O** 'S', **Y** que el valor de carater_ascii a imprimir no sobrepase de 126, si es así ir al **paso 1**, en otro caso terminar.

Diagrama de Flujo



Pseudocódigo. “CaracteresASCII ”

comienza

hacer

comienza

respuesta ← ‘N’

Escribir (caracter - caracter (entero))

caracter ← caracter + 1

Escribir (“Desea continuar S (s) o N (n):”)

Leer (respuesta)

t**ermina**

mientras ((respuesta = ‘S’ o respuesta ‘s’) Y caracter < 126)

termina

Código en C (version 1)

```
/*
  Archivo: CaracteresASCII.c
  Descripción: Este programa despliega en pantalla los caracteres ASCII y su
               representación en entero. Este proceso se continua hasta que el
               usuario tecle 'N' o 'n', o bien no se sobrepase de 126.
              Objetivo:
               Utilización de la estructura de control - Iteración Condicional
                  -Sentencia do-while (hacer-mientras)
               Lectura de caracteres desde teclado con: scanf() o con getche()
  Autor: Basurto Páez Gustavo
  Fecha: 14 de octubre de 2007
```

```
*/
/*Incluir librerias utilizadas en el programas*/
#include <stdio.h>
int main()
{
    /*Declaración (e inicialización) de variables*/
    int respuesta = 0 ;
    char caracter = ' ';

    printf("\t Programa : Caracteres ASCII \n\n ");

    //Uso de Iteración condicional - do - while (hacer  - mientras)
    do
    {
              respuesta = 'n';
              printf("\n Caracter   Entero \n    (%c)  -  (%d) ", caracter, caracter);
              caracter ++; // caracter = caracter + 1

              printf(" \n Desea continuar (S/N): "  );
              //La funcion limpia el buffer de entrada STD IN
              fflush( stdin );

              //Utilizacion de la funcion scanf para la lectura de caracteres
              scanf("%c", &respuesta);

    }
    while ( ( respuesta == 's' || respuesta  == 'S') &&  caracter <= 126 );

    printf("\n\nPresione una tecla para finalizar el programa...");
    getch();
    return (0);
}
```

Codigo C (version 2)

```
/*Incluir librerias utilizadas en el programas*/
#include <stdio.h>
int main()
{
    /*Declaración (e inicialización) de variables*/
    int respuesta = 0 ;
    char caracter = ' ';

    printf("\t Programa : Caracteres ASCII \n\n ");

    //Uso de Iteracion condicional - do - while (hacer  - mientras)
    do
    {
              respuesta = 'n';
              printf("\n Caracter   Entero \n    (%c)  -  (%d) ", caracter, caracter);
              caracter ++; // caracter = caracter + 1

              printf(" \n Desea continuar (S/N): "  );

             //Es posible hacer uso de la función getche()
              //la cual lee 'un' caracter de pantalla inmeditamente (no es
              // necesario presionar enter.
              //Es similar a getch() sin embargo la 'e' es de echo

              respuesta = getche();

    }
    while ( ( respuesta == 's' || respuesta  == 'S') &&  caracter <= 126 );

    printf("\n\nPresione una tecla para finalizar el programa...");
    getch();
    return (0);
}
```

Evidencias

```
         Programa : Caracteres ASCII
 Caracter   Entero
    ( )  -  (32)
 Desea continuar (S/N): s
 Caracter   Entero
    (!)  -  (33)
 Desea continuar (S/N): s
 Caracter   Entero
    (")  -  (34)
 Desea continuar (S/N): n
Presione una tecla para finalizar el programa...
-Otra corrida
         Programa : Caracteres ASCII
Caracter   Entero
    (})  -  (125)
 Desea continuar (S/N): s
 Caracter   Entero
    (~)  -  (126)
 Desea continuar (S/N): s
Presione una tecla para finalizar el programa...
```

Ejemplo *"Juego de Apuestas"*

En el juego, el usuario comienza con 100 pesos y le apuesta a un dado. Si sale el número que apostó, gana lo que apostó, sino se le retira de su dinero. Se requiere que:

-Al leer la cantidad de dinero apostada, se debe verificar que el jugador tenga suficiente.
-Al leer el número al que se le apuesta, se debe verificar que este en el rango del dado (1-6).
-Mostrar el resultado del dado y el número que aposto el usuario.
-Mostrar si gano o perdió, y cuanto le queda
-Preguntar si quiere salir o no
-Si el usuario quiere continuar, hacer otra ronda sólo si tiene dinero.
-Para sacar un número aleatorio se utiliza la función ***aleatorio***(6) en pseudocódigo que se traduce a ***rand() %6 + 1*** en C. (incluir **stdlib.h** y también hacer uso de la función ***srand(100)*** al principio, justo después de la declaración de variables).
-Al salir mostrar cuanto dinero quedo.

Esto es lo que debe ver el usuario:

```
                Programa : Juego de Apuesta con lanzamiento de Dado
                Inicias con la cantidad de $100.
       Apuesta una cantidad menor y se efectua el lanzamiento de dado. Si el dado
    que da en el numero que seleccionaste es ganas la cantidad apostada, caso
    contrario pierdes.
       Comienza!!!
       Cuanto apuestas (max=100)?:50
       Selecciona un numero (1-6)?:6
       El dado salio:6
       Tu le apostaste al:6
       Ganaste 50 pesos, tu total es:150
       Deseas salir (s/n)?:N
       Cuanto apuestas (max=150)?:50
       Selecciona un numero (1-6)?:2
       El dado salio:5
       Tu le apostaste al:2
       Perdiste 50 pesos, te quedan:100
       Deseas salir (s/n)?:n
       Cuanto apuestas (max=100)?:50
       Selecciona un numero (1-6)?:2
       El dado salio:4
       Tu le apostaste al:2
       Perdiste 50 pesos, te quedan:50
       Deseas salir (s/n)?:n
       Cuanto apuestas (max=50)?:50
       Selecciona un numero (1-6)?:4
```

```
El dado salio:1
Tu le apostaste al:4
Perdiste 50 pesos, te quedan:0
Deseas salir (s/n)?:n
Te quedaron 0 pesos... Hasta luego!
Presione una tecla para finalizar el programa...
```

Entrada: Cantidad apostar, un número al que se apuesta y la respuesta de continuar.
Salida: El resultado del dado, la cantidad gaste, la opción de continuar o salir del programa.
Pseudocódigo. "***ApuestaDado***"

```
comienza
    dinero ← 100
    hacer
    comienza
        hacer
        comienza
            Escribir ( "Cuanto apuestas?")
            Leer (apuesta)
        termina
        mientras (apuesta > dinero)

        hacer
        comienza
            Escribir ("Selecciona un numero (1-6)?:")
            Leer (numero)
        termina
        mientras (numero < 1 O numero > 6)

        dado ← aleatorio (6)
        Escribir ("El dado salio:", dado)

        Si (dado = numero) entonces
        comienza
            dinero ← dinero + apuesta
            Escribir ("Ganaste $" apuesta " ahora tu total es: $" dinero)
        termina
        otro
        comienza
            dinero ← dinero – apuesta
            Escribir (" Perdiste $" apuesta ", te quedan: $" dinero)
        termina

        hacer
        comienza
            Escribir ("Deseas Salir (s/n)?:")
            Leer (respuesta)
        termina
        mientras ( (respuesta ≠ 's' Y respuesta ≠ 'S' ) Y (respuesta ≠ 'n' Y respuesta ≠ 'N' ) )

    termina
    mientras (dinero > 0 Y respuesta ≠ 's' Y respuesta ≠ 'S')

    Escribir ("Te quedaron $" dinero )
termina
```

Código en C

```
/*
Archivo: ApuestaDado.c
Descripción: Este programa representa el juego de azar de tiradas del dado. Se comienza
       con un cantidad de dinero inicial.
       El usuario selecciona la cantidad a apostar y un numero al cual se apuesta (1-6).
       El programa genera un número aleatorio y se limita en el rango de 1-6.
       Si se es igual al numero seleccionado entonces se gana la apuesta, en caso
       contrario se disminuye dicha cantidad.
       El juego termina cuando el usuario desea salir o bien cuando ya no tenga dinero
       para apostar.
       Objetivo:
          Utilización de la estructura de control - Iteración Condicional -Sentencia do-
              while (hacer-mientras)
          Utilización de funciones para generar números aleatorios.
          Utilización del operador modulo para delimitar rangos de valores
       Especificaciones:
          #define RAND_MAX 0x7FFF (32767)
          REF http://www.conclase.net/c/librerias/macro.php?mac=RAND_MAX
Autor: Basurto Páez Gustavo
Fecha: 15 de octubre de 2007
*/

/*Incluir librerias utilizadas en el programas*/
#include <stdio.h>
#include <stdlib.h> //Proporciona las funciones srand() y rand()

int main()
{
    int dado, apuesta, deseado;
    int dinero=100;
    char c;
    //inicializacion del ramdom, se le pasa como semilla un entero positivo
    srand ( 100 );

    printf("\t Programa : Juego de Apuesta con lanzamiento de Dado\n\n ");
    printf("\t Inicias con la cantidad de $%d. \n Apuesta una cantidad menor y se efectua
             el lanzamiento ", dinero);
    printf(" de dado. Si el dado queda en el numero que seleccionaste es ganas la
             cantidad apostada, caso contrario pierdes. \nComienza!!!");

    do
    {
      //Solicitud de dinero de apuesta
         do
         {
             printf("\n\nCuanto apuestas (max=%d)?:",dinero);
             scanf("%d",&apuesta);
          }while(apuesta>dinero);

          //Solicitud de numero a apostar
          do
          {
             printf("Selecciona un numero (1-6)?:");
             scanf("%d",&deseado);
          }while(deseado < 1 || deseado > 6);


       //la funcion regresa un numero aleatorio de acuerdo a las semilla, entre 0 y 32767
       // (65535)
       //se le aplica el modulo para que genere un valor entre (0 - 5), por eso se le
       //suma 1 (1 - 6)

          dado=(rand() % 6 )+1;

          printf("El dado salio:%d\n",dado);
          printf("Tu le apostaste al:%d\n",deseado);
```

```
        if(deseado==dado)
        {
            dinero = dinero + apuesta;
            printf("Ganaste $%d, tu total es:$%d\n",apuesta,dinero);
        }
        else
        {
            dinero = dinero - apuesta;
            printf("Perdiste $%d, te quedan:$%d\n",apuesta,dinero);
        }

        //Solicitud de continuar con el programa
        do
        {
            printf("\nDeseas salir (s/n)?:");
            c=getche();
        }while((c!='s' && c != 'S' ) && ( c!='n' && c !='N') );

        printf("\n");
    }while((c!='s' && c!='S')&& dinero>0);
    printf("\n\nTe quedaron $%d ... Hasta luego!\n",dinero);
    printf("\n\nPresione una tecla para finalizar el programa...");
    getch();
    return (0);
}
```

Evidencias

Ver a detalle en el planteamiento del problema

4.6.3.3 Sentencia For (para ….. hasta ….. hacer)

Recordando que un **ciclo** (bucle) representa una sección del programa que **repite** una serie de instrucciones, por un número determinado de veces de acuerdo a la validación de una **condición**, el ciclo ***para*** representa una estructura de iteración condicional. La condición que presenta es determinada por una ***expresión lógica***, que permite controlar la secuencia de repetición de la estructura.

La sentencia ***para*** es un ciclo para ejecutar un bloque de sentencias (instrucciones), un número fijo de veces. Es decir, cuando se conoces la cantidad de veces que se va a repetir el cuerpo del ciclo.

Sintaxis de la sentencia ***para***, en el procedimiento en un algoritmo es:

"**Para** cada valor de variable_contador de un rango dado **hacer** pasos n, n+1, …, n+m."

La sentencia para ***for*** esta constituida por las siguientes partes:

-**Cabecera de la sentencia:**

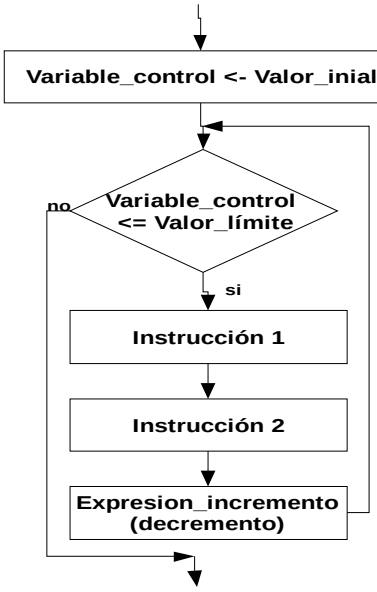
-***Inicialización***: inicialización de las variables de control del ciclo.

-***Condición de iteración***: expresión lógica para determinar si se continua con el cuerpo del ciclo, siempre que esta se verdadera.

-***Incremento/decremento***: para la(s) variable(s) de control del ciclo.

-**Cuerpo del ciclo**: representa el conjunto de instrucciones.

Diagrama de Flujo de la sentencia ***para… hasta… hacer***



"Si la condición es falsa (aun desde el inicio), no se efectúan las intrusiones del cuerpo del ciclo. Se continua con la siguiente instrucción del programa".

Una vez finalizada la ultima instrucción del cuerpo del ciclo, el flujo continua para verificar nuevamente si se cumple la condición. Considerando el incremento o decremento de la(s) variable(s) de control.

Definición 8: Sentencia iterativa *for* (para … hacer… hasta)

Pseudocódigo

para (valor_control ← <valor_inicial> **hasta** <condición sobre valor_control> **con** <inc o dec>) **hacer**
comienza
<instrucciones A>
…..
<instrucciones N>
finPara

En Código C

```
for (valor_control = valor_inicial ; var_control '<=' valor_limite ; exp_incremento_decremento )
{
 /**instrucciones A**/
        …….
 /**instrucciones N**/
 }   /*Fin For*/
```

Operadores de incremento y decremento

El lenguaje de programación C, proporciona los operadores de incremento y decremento, mismo que son representados como abreviatura, del incremento o decremento en una unidad entero.

En C se utilizan de la siguiente forma:

```
var_incremento++;          /* var_incremento = var_incremento + 1 */
var_decremento--;          /* var_decremento  = var_decremento -1 */

var_incremento++;          /*postincremento*/
++ var_incremento;         /*preincremento*/
var_decremento--;          /*postdecremento*/
-- var_incremento;         /*predecremento*/
```

Ejemplo

Dada la variable ***cont*** efectuar una serie de incrementos y decrementos, y el resultado se asigna a la variable ***var***. El objetivo es verificar que en cada instrucción al ser ejecutada se modifica el contenido en cada variable.

En C

```
cont  = 10;
var = cont ++;     //var vale 10 y cont vale 11, post-incremento
var = ++cont;      //var vale 12 y cont vale 12, pre-incremento
var = cont--;      //var vale 12 y cont vale 11, post-decremento
var = --cont;      //var vale 10 y cont vale 10, pre-decremento
```

Ejemplo: "***Mensaje Hola Mundo***"

Leer un número entero y escribir en pantalla ese número de veces el mensaje de 'Hola Mundo'.

Entrada: Número entero (numEntero).
Salida: numEntero el mensaje de "Hola Mundo.

Pseudocódigo "***Mensaje Hola Mundo***"

Comienza

Numentero ← 0
Escribir ("Proporcione un número entero:")
Leer (numEntero)

para (cont ← 0 **hasta** cont < numEntero **con** cont ← cont + 1) **hacer**
comienza
Escribir (cont " Hola Mundo")
finPara

Termina

Código en C

```
/*
  Archivo: HolaMundoN.cpp
  Descripción: Este programa lee desde teclado un número entero
               y manda a pantalla ese numero de veces el mensaje de
               "Hola Mundo"
             Objetivo:
                    Utilización de la estructura de control - Iteración Condicional -
             Sentencia for ( )(para... hasta... hacer)
             Observación:
             Si es necesario haga una "Tabla de seguimiento" para verificar que es
             validad la condición cont < numEntero.
  Autor: Basurto Páez Gustavo
  Fecha: 22 de octubre de 2007
*/

/*Incluir librerías utilizadas en el programas*/
#include <stdio.h>

int main()
{
     /*Declaración (e inicialización) de variables*/
    int numEntero = 0 ;
    int cont = 0;

    printf("\t Programa Hola Mundo N-Veces \n\n ");
    printf(" Proporcione un numero entero: " );
    scanf("%d", &numEntero);

    printf(" Se imprimirán %d veces el mensaje:'Hola Mundo'\n\n ", numEntero);

    //Inicio del ciclo for, el contador empieza desde 0
    //se valida hasta que cont sea menor a numEntero
    //por lo tanto se imprimirá  diez veces: desde el 0 hasta el 9
    for(cont =0 ; cont < numEntero; cont++)
    {
         printf( " %d.- Hola Mundo con el ciclo For....   \n", cont);
    } //fin for

    getch();
    return (0);
}
```

Ejemplo *"Solución de una función"*

Dada una función cuadrática $f(x) = x^2 + 4x + 4$, se solicita al usuario un intervalo en donde se encuentre la solución (intervalo inferior e intervalo superior). El programa desplegará el valor de X y el resultado generado al evaluar la función.

Entrada: Rango Inferior, Rango Superior.
Salida: Mensajes indicando la evaluación de la función en cada una de las variables x.

Pseudocódigo. "***EvalFuncion***"

```
Comienza
        Escribir ("Evaluación de la función en un rango determinado [inf –sup]")
        Escribir ("Defina el rango inferior:")
        Leer (rango_inf)
        Escribir ("Defina el rango superior:")
        Leer (rango_sup)

        var_x ← rango_inf
        para ( cont ← rango_inf  hasta  cont < rango_sup con cont ← cont + 1  ) hacer
        comienza
                resultado ← var_x * var_x + 4*var_x + 4
                Escribir ("EL resultado de la evaluación con " var_x " es: " resultado)

                si ( resultado = 0) entonces
                comienza
                        Escribir ("Se encontró una solución, con el valor de :" var_x)
                termina

                var_x ← var_x + 1
        finPara
Termina
```

Código en C

```
/*
  Archivo: EvalFuncion.c
  Descripción: Este programa permite la evaluación de una función cuadrática
               considerando que se introduce un rango de posibles soluciones.
               Mediante el uso de una ciclo se recorre uno a uno de estos valores y se va
             evaluando la función. En caso de encontrar una solución, es decir, si el
             resultado de la evaluación es cero, entonces se envía a pantalla el mensaje
             indicando que se ha encontrado una solución.
  Autor: Basurto Paez Gustavo
  Fecha: 30 de octubre de 2007
*/

/*Incluir librerías utilizadas en el programas*/
#include <stdio.h>

int main()
{
     /*Declaración (e inicialización) de variables*/
    int var_x  = 0 ;
    int resultado  = 0;
    int cont = 0;
    int rango_inf =-20;
    int rango_sup = 20;

    printf("\t Programa De Búsqueda de Solución  \n\n ");
    printf(" Dada la función f(x) =  x*x + 4*x + 4,  solicitar el un rango para ");
    printf(" definir en donde esta la aplicación. \n\n\n Tecle el rango inferior: " );
    scanf("%d", &rango_inf);

    printf(" Tecle el rango superior: " );
```

```
    scanf("%d", &rango_sup);

    //Asignacion para definir  contexto
    var_x = rango_inf;

    for( cont=rango_inf; cont < rango_sup; cont++)
    {
         resultado  =  var_x*var_x + 4*var_x + 4;
         printf(" \n\t  Valor de X:%d, F(x):%d", var_x, resultado );
         if ( resultado == 0)
         {
            printf(" \n\t\t ***Se encontro una solucion con x = %d **\n ", var_x);
         }
         var_x++;
    } //Fin for


    getch();
    return (0);
}
```

Ejemplo "Tablas de multiplicar del 1 al 10"

El programa desplegará las tablas de multiplicar del 1 al 10.

Entrada: Ningún datos.
Salida: Mensaje indicando las tablas de multiplicación.

Pseudocódigo. "***Tabla10***"

Comienza

Escribir (" Tablas de multiplicar del 1 al 10:")

para (cont_1 ← 0 **hasta** cont_1 < 10 **con** cont_1 ← cont_1 + 1) **hacer**
comienza

para (cont_2 ← 0 **hasta** cont_2 < 10 **con** cont_2 ← cont_2 + 1) **hacer**
comienza
producto ← cont_1 * cont_2
Escribir (con_1"X" cont_2 "=" producto)
finPara

finPara

Termina

Código en C

```
/*
  Archivo: Tabla10.c
  Descripcion: Este programa despliega cada una de las tablas desde
               1 al 10.
              Objetivo: Utilización de la estructura de control - Iteración Condicional -
              Sentencia for ( )(para... hasta... hacer) ANIDADOS
              Observación:  Si  es  necesario  haga  una  "Tabla  de  seguimiento"  para
              verificar el uso de los ciclos anidados.
  Autor: Basurto Páez Gustavo
  Fecha: 22 de octubre de 2007
*/
/*Incluir librerías utilizadas en el programas*/
#include <stdio.h>

int main()
{
     /*Declaración (e inicialización) de variables*/
```

```
    int cont_1 = 0 ;
    int cont_2 = 0 ;
    int producto = 0;

    printf("\n\t Programa de las Tablas de Multiplicación (1-10)\n\n ");

     //CICLO FOR EXTERNO
    //Inicio del ciclo for, el contador empieza desde 1 hasta que cont_1 sea //menor a 10
    for(cont_1 =1 ; cont_1 <= 10; cont_1++)
    {
        printf("\n   ******************************** \n");
        printf("\tTabla de multiplicar del %d \n", cont_1);
        printf("   ******************************** \n");

        //CICLO FOR ANIDADO
        //Uso de otro contador por la independencia entre ambos
        for(cont_2 =1 ; cont_2 <= 10; cont_2++)
        {
                  producto = cont_1 * cont_2;
                  printf( " \t %d X %d = %d \n", cont_1, cont_2, producto);
        }//fin for anidado

    } //fin for

    getch();
    return (0);
}
```

4.6.4 *Ejercicios de Unidad 4*

Los siguientes ejercicios permiten ejercitar aspectos prácticos de la unidad 4. Algunos ejercicios son propuestas del capitulo 3 en donde se sugirió el desarrollo del Algoritmo, Diagrama de Flujo y/o Pseudocódigo.

Para el desarrollo de los siguientes ejercicios se requiere efectuar previamente las fases de Análisis y Diseño, ya que se contempla desarrollar las siguientes fases: Codificación, Compilación, Ejecución y Pruebas, Documentación y Mantenimiento.

I. Sentencias Simples, Estructura de Control Condicional Simple, Doble y Múltiple.

1. "***Conversión de kilómetros a metros y a centímetros***"
   Elabore un programa que dada una cantidad en kilómetros, sea convertida en metros y en centímetros. Desarrolle algoritmo, diagrama de flujo y pseudocódigo. Ejemplo: 1.250 Km. equivale a 1,250 metros y a 125000 centímetros.
2. "***Mayor de dos números***"
   Elabore un programa que dado dos números enteros se determine el mayor de los dos.

3. "***Convertir horas minutos y segundos a sólo segundos***"
   Elabore un programa que efectué la conversión de horas con minutos a segundos. Ejemplo: 2 horas y 35 minutos y 5 segundos son 9305 segundo.

4. "***Convertir segundos a horas, minutos y segundos***"
   Elabore un programa que efectué la conversión de segundos a horas y minutos. Ejemplo: 10000 segundos equivale a 2 horas y 46 minutos (y 40 segundos).

5. "***Mayor de tres números***"
   Elabore un programa que dado tres números enteros se determine el mayor de los tres.

6. "***Agenda – Switch***"
   Elabore un programa que pida el día de la semana como un número del 1 al 7, y muestre los mensajes siguientes dependiendo del día. Si se introduce un día distinto de 1 a 7 se marca un error.
   1.- Lunes: Regresar al estudio, hacer tareas, entregar reportes, etc.
   2.- Martes: Estudiar para la clase Introducción a la Programación.
   3.- Miércoles: Hacer ejercicio e ir al Cine 2X1.
   4.- Jueves: Estudiar para el examen.
   5.- Viernes: Hacer el examen, y salir a 'cenar' con los amigos.
   6.- Sábado: Ir a comprar cosas y después descansar todo el día.
   7.- Domingo: Comer con la familia y a listarse para la siguiente semana.

II. Estructura de Control Iterativa – *While*.

1. "***Suma de 10 valores - Mientras***"
   Elabore un programa que lea 10 números enteros y que los vaya sumando, al finalizar despliegue la suma total de los números ingresados. (¿Qué haría si en lugar de leer 10 números fuesen 100? ¿En qué partes del diseño – diagrama de flujo, pseudocódigo y código modificaría?).

2. "***Solo enteros positivos - Mientras***"
   Elabore un programa que lea tantos números enteros como sea posible hasta que el usuario teclee un valor negativo o cero. (¿Se requiere el uso de un contador? Es decir, ¿Se necesita un contador en el condicional?).

3. ***"Promedio de N exámenes***"
   Elaborar un programa que calcule el promedio de los N exámenes (flotantes positivos), considerando que el valor de N es introducido por el usuario y cada una de las N calificaciones. Se espera que como respuesta se escriba el promedio de las calificaciones y el mensaje si aprobó o no el curso (el promedio es aprobatorio si es mayor o igual a 6.0).

4. "**Reporte de números ingresados – *While***"
Desarrolle un programa que reciba un número determinado de números enteros (N), el usuario ingrese cada uno de los N valores, y al final el programa deberá de generar un reporte indicando:

i. La cantidad de números positivos.
ii. La cantidad de números positivos.
iii. La cantidad de números pares.
iv. La cantidad de números impares.
v. La cantidad de números igual a cero.

Sugerencia: Utilice un contador para cada uno de los incisos.

III. Estructura de Control Iterativa – *Do … while*.

1. "***Imprimir Contador***"
Elabore un programa que utilice un contador (inicializado en 0) para imprimir su contenido cada vez que el usuario teclea la letra S. Ejemplo:

```
0
Desea continuar (S/N): S
1
Desea continuar (S/N): S
2
Desea continuar (S/N): S
3
Desea continuar (S/N): N
Hasta pronto!!!
```

Sugerencia: El contador no es considerado dentro de la condición, sin embargo la letra que tecleo el usuario 'S' o 'N', es importante para condición.

2. "***Suma de 10 valores - Hacer Mientras***"
Diseñe un programa similar al ejercicio "Suma 10 valores – Mientras", considerando el uso de la sentencia ***Hacer... Mientras***. (¿Qué modificaciones se tendría que efectuar?. En este ejercicio, ¿Cuál estructura de control iterativa (condicional) es adecuada utilizar: ***Mientras*** o ***Hacer... Mientras***?).

3. "***Solo enteros positivos – Hacer Mientras*** "
Diseñe un programa similar al ejercicio "Solo enteros positivos - Mientras", considerando el uso de la sentencia ***Hacer... Mientras***. (¿Qué modificaciones se tendría que efectuar?. En este ejercicio, ¿Cuál estructura de control iterativa (condicional) es adecuada utilizar: ***Mientras*** o ***Hacer... Mientras***?).

IV. Estructura de Control Iterativa – *For*.

1. "***Suma de N valores***"
Elabore un programa que lea N números enteros y que los vaya sumando, al finalizar despliegue la suma total de los números ingresados. El valor de N es proporcionado por el usuario antes de ingresar los valores a sumar.

2. "***Solo enteros positivos - Para***"
Elabore un programa que lea tantos números enteros como sea posible hasta que el usuario teclee un valor negativo o cero. (¿Cuál es la estructura de control iterativa adecuada para el desarrollo de esté programa?).

3. "***Aproximación a la solución de una función***"
Considerar el ejemplo "Solución de una función" en la sección 4.6.3.3 con la diferencia que se utilizaran valores flotantes para la aproximación. Se solicitará al usuario además de los intervalos inferiores y superiores, un rango en punto flotante (float) para el incremento del valor de X. Esto permitirá obtener una mayor aproximación a la solución de la función.

4. "**Promedio de N exámenes**"
Calcular el promedio de los N exámenes (el valor de N es introducido por el usuario, cada promedio es un flotantes positivos), considerando que el valor de N es introducido por el usuario y cada uno de las N calificaciones. Se espera que como respuesta se escriba el promedio de las calificaciones y el mensaje si aprobó o no el curso (el promedio es aprobatorio si es mayor o igual a 6.0).

5. "**Tabla de Multiplicar NXN**"
Desarrolle un programa que efectué las tablas demultiplicar de acuerdo al valor ingresado por el usuario. Comenzando con la Tabla del 1, Tabla del 2, etc.

6. "**Reporte de números ingresados – *For***"
Desarrolle un programa que reciba un número determinado de números enteros (N), el usuario ingrese cada uno de los N valores, y al final el programa deberá de generar un reporte indicando:

i. La cantidad de números positivos.
ii. La cantidad de números positivos.
iii. La cantidad de números pares.
iv. La cantidad de números impares.
v. La cantidad de números igual a cero.

Sugerencia: Utilice un contador para cada uno de los incisos.