UNIDAD 2.- ESTRUCTURA ELEMENTAL DE DATOS. DISEÑO DE ALGORITMOS.

OBJETIVOS:

- Que el alumno comprenda el concepto de dato y las nociones básicas más importantes para la construcción de algoritmos, la forma de realizar cálculos y la noción de acción.
- Que el alumno comience a formularse y resolver problemas, diseñando las estrategias correspondientes de manera clara, sistémica y por sobre todo sencilla, mediante el diseño de algoritmos.
- Que el alumno logre aumentar la capacidad de reflexión del alumno, reforzando las conductas logradas mediante la Unidad 1.-

TEMAS:

2.1. Estructuración de un programa: encabezamiento, bloque de declaraciones, bloque de acciones.

2.2. Representación de datos elementales.

2.3. Operación de asignación.

2.4. Sentencias de entrada y salida.

2.5. Las instrucciones simples y compuestas.

2.6. La implementación de las estructuras secuenciales, condicionales.

2.1. ESTRUCTURACIÓN DE UN PROGRAMA: ENCABEZAMIENTO, BLOQUE DE DECLARACIONES, BLOQUE DE ACCIONES.

Todas las herramientas para la elaboración de algoritmos estudiadas hasta ahora, constituyen una ayuda para la codificación final del mismo. Esto significa escribir el algoritmo, en un lenguaje de alto nivel.

Todo programa algorítmico consta de diversos elementos, algunos de los cuales pasaremos a analizar.

PALABRAS RESERVADAS

Son identificadores que el lenguaje se reserva para su uso propio, y no pueden ser usadas como identificadores declarados por el usuario. Es decir, no podemos usar estas palabras como nombres de variables, constantes, funciones, etc.
EJEMPLO: auto, break, case, char, const, else, int, return etc.

Los identificadores deben cumplir también ciertas reglas impuestas por el lenguaje, tales como:

- Comenzar con una letra mayúscula o minúscula o cualquier símbolo siempre que este no represente una operación algebraica o lógica, o carácter especial.
  EJEMPLO

Válidos:	A123456	No válidos:	+sumar
	ecuenta		&letra
	b_numero		evalua*cion

  Se recomienda usar identificadores con nombres simples, que estén asociados con la acción de la variable o constante que representan.

- Los identificadores escritos con letras mayúsculas y minúsculas son considerados diferentes. Así por ejemplo la variable ***a*** no es igual a la variable ***A***. Ambas son diferentes.

- Generalmente todas las directivas en un programa en C se escriben con letra minúscula. Algunas definiciones especiales que corresponden al programa, y las variables, constantes, funciones y otras declaraciones usadas por el usuario, pueden ser escritas con mayúscula.

Desde el punto de vista de la escritura de un algoritmo, el mismo puede dividirse en tres partes:

- Cabecera del Programa
- Sección de Declaraciones
- Cuerpo del Programa

CABECERA DEL PROGRAMA

Los programas comienzan con un **nombre del programa** o palabras reservadas que el lenguaje especifica. En el caso del lenguaje C que se va a estudiar, se explicita los denominados archivos de cabecera en los que se indica que archivos de inclusión se usarán en el programa a construir.

en lenguaje C: **# include <stdio.h>**

Observación: *La directiva* **# *include <stdio.h>*** *no es estrictamente un nombre de programa. Es la declaración de los archivos de inclusión que se usarán en el programa. Con esta y otras directivas,* ***comienza todo programa en C.***
El símbolo # es también imposición de la sintaxis del lenguaje.

SECCIÓN DE DECLARACIONES

Es el sector del programa donde se especifican las variables, constantes, funciones, tipos de datos, etc., que se usan en el mismo. En el caso de las variables y constantes, se especifica el tipo de dato al que pertenecen. En forma general, la declaración será:

identificador < **tipo de dato >**

EJEMPLO en lenguaje C:

```
int a, b;
float c;
```

En esta sección se especifica el identificador y tipo de dato de cada variable.
En el caso del lenguaje adoptado, se lista a continuación los diferentes tipos de datos aceptados por el mismo y la especificación de rango que pueden tomar las variables de cada tipo.

Nombre	Tipo	Rango
unsigned char	Carácter	0 hasta 255
char	**Carácter**	± **127**
unsigned int	Numero	0 hasta 65,535
short int	Numero	-32.767 hasta 32.768
int	**Numero**	**-32.767** hasta **32.768**
unsigned long	Numero	0 hasta 4.294.967.295
long	Numero	± 2.147.483.647
float	**Numero**	**$1 * (10^{\pm 37})$**
double	Numero	$\pm 1*(10^{\pm 37})$
long double	Numero	$\pm 1*(10^{\pm 37})$

Observación: *Se escriben en* ***negrita*** *los tipos de datos más usados y a los que generalmente nos referiremos en este curso.*

Declaración de constantes

Constantes numéricas:

Se escribe el identificador precedido de la palabra reservada **const,** el signo igual y a continuación el tipo de dato y valor numérico que representa.

EJEMPLO en lenguaje C:

```
const  float  numeropi = 3.14 ;
const int  total = 100 ;
const  char letra = ‘ A ‘ ;
```

Observación: *Las constantes declaradas también ocupan un lugar en memoria como las variables. Solo que su valor no cambia durante la ejecución de un programa.*

Documentación del programa

Un programa adecuadamente escrito debe contener comentarios a fin de facilitar su comprensión y posterior mantenimiento y corrección.

Para la escritura de comentarios, se usa la siguiente nomenclatura:

En lenguaje C:
/ *** *este es un comentario* * /***

Las declaraciones de variables y constantes en el lenguaje **C** pueden ser escritas en esta **sección de declaraciones** o bien en el cuerpo del programa. Debe destacarse que su *significado es diferente*. También las variables pueden declararse en cualquier lugar del programa, pero en este curso al principio adoptaremos el hábito de declararlas en esta sección. (Antes del programa principal o main())

CUERPO DEL PROGRAMA

Cada acción o grupos de acciones en diagrama de flujo, serán escritos de acuerdo a las reglas sintácticas del lenguaje.

- En el caso del lenguaje C, todo cuerpo del programa se inicia con la denominada función **main()**, que es en donde se escribe la sucesión de acciones para la resolución del problema planteado. Es lo que previamente se denominará **programa principal.**

- A continuación de la función main() o programa principal, se escribe el símbolo delimitador de la misma. Este símbolo consiste en la apertura de una llave que indica el comienzo de la misma.(**{**)**.**

- Después de la finalización de la escritura de todas las instrucciones, se pone el delimitador de cierre que una llave de terminación (**}).**

Para la codificación del caso anterior, deberemos escribir :

```
# include <stdio.h>          [Cabecera del programa o archivos de inclusión ]

[ declaraciones de variables,  constantes y prototipos de funciones ]

main ( )                     [Comienzo del programa principal o función main ]
{                            [Delimitador de inicio del programa principal o fcion. main]
        sentencia1 ;
        sentencia2 ;
        . . . . . .
        sentencia N;

        return (0);          [Retorno de la función main y fin del programa ]
}                            [Delimitador de cierre de la función main]
```

Luego de escribir la sucesión de acciones (sentencias) que conforman el programa, y antes de colocar el indicador de terminación del programa o llave de cierre, se escribe la directiva **return (0),** la cual representa el retorno de la función main().

Las instrucciones que se usan en un programa computacional en **C** pueden clasificarse de la siguiente manera:

- Instrucciones de comienzo y finalización
- Instrucciones de asignación
- Instrucciones de lectura
- Instrucciones de escritura
- Instrucciones de bifurcación o selección
- Instrucciones de repetición

.

- **Instrucciones de comienzo y finalización**
  Son las llaves de apertura y cierre.

- **Instrucciones de asignación**
  Se usa el operador **=**
  Por ejemplo para asignar 10 a la variable **a** se escribe **a = 10;**
  equivalente a **a ← 10.**
  Después de cada terminación de sentencia, se escribe un **;** (punto y coma). Esto es para indicar la finalización de la directiva. La no escritura de la misma, da lugar a un error de sintaxis.

- **Instrucciones de lectura**
  Son las que permiten ingresar valores a las variables mediante el teclado. La directiva que se usa es**:**

  scanf (*especificador de formato* **,** ***&****variable cuyo valor se quiere ingresar* **) ;**

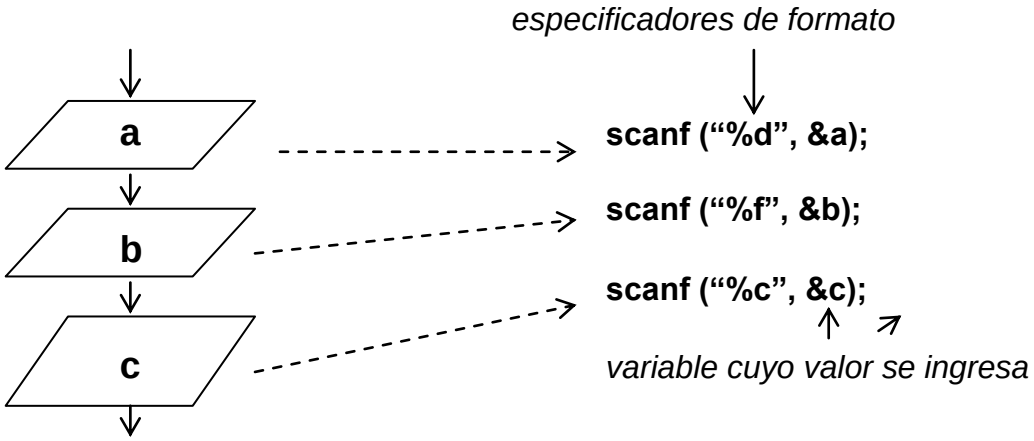
  - *especificador de formato:* definen el tipo de dato cuyo valor se ingresara. Se lo utiliza precedido por el símbolo **%.** La siguiente lista muestra las cadenas de formato para diferentes tipos de variables.

    %c carácter
    %d enteros decimales con signo
    %f números flotantes
    %s cadena de caracteres
    %x hexadecimales sin signo
    %p puntero o dirección de memoria

    El especificador de formato se escribe entre los símbolos **“ “.**

  - *variable cuyo valor se desea ingresar:* Se escribe la variable a la que se quiere ingresar un valor, precedida por el símbolo **&** cuando se trata de variables simples.

    EJEMPLO
    Ingresar mediante el teclado, los valores de tres variables:
    a (entera), b (flotante) y c (caracter).

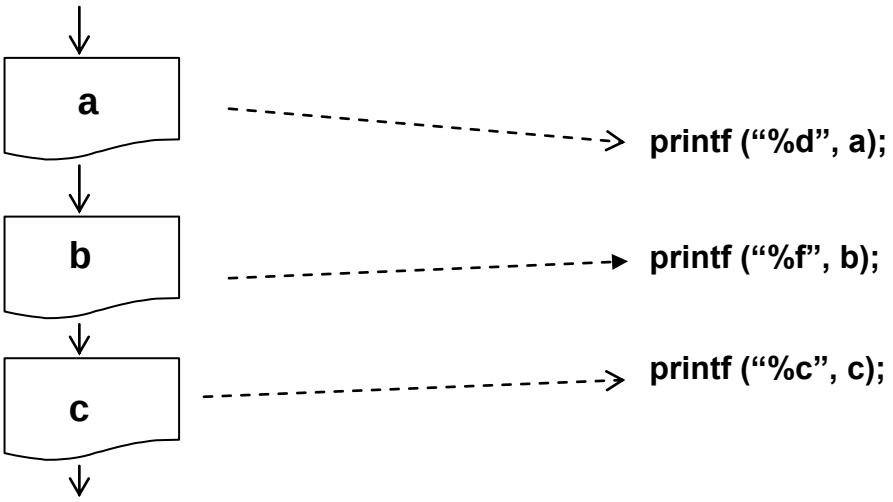


- **Instrucciones de escritura**
  Permiten mostrar el valor de variables, en el monitor. La función que se usará es:

  printf (*especificador de formato* **,** *variable cuyo valor se quiere mostrar* **) ;**

Esta directiva o función, incluye el uso del mismo especificador de formato como el caso de **scanf ( )**. La variable cuyo valor se quiere mostrar, **no se escribe precedida por el símbolo &.**

EJEMPLO
Mostrar las variables a (entera), b(flotante) y c (caracter) ingresadas en el ejemplo anterior.



Con los elementos expuestos, se codifica un programa en C donde se utilizan las partes explicitadas hasta ahora:

EJEMPLO

Escribir un programa en C en el que se ingrese por teclado el valor de una variable entera, otra flotante y un carácter y se muestre su valor por el monitor:

```
#include <stdio.h>                    } Cabecera del programa
int a;
float b;                              } Declaración de variables
char c;

main()
{     scanf("%d", &a);
      scanf ("%f", &b);
      scanf ("%c", &c);

      printf ("%d", a);               } Cuerpo del programa
      printf ("%f", b);                 o función main
      printf ("%c", c);
}
```

Figura 1

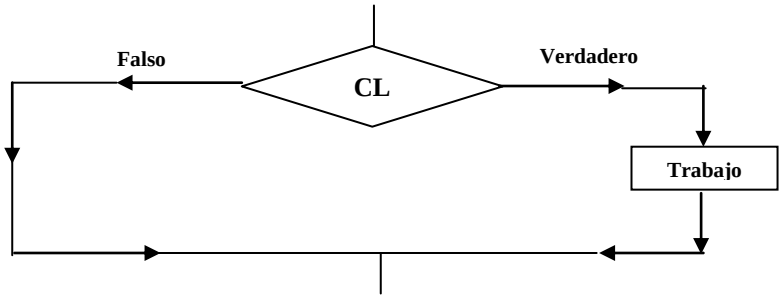
ESTRUCTURAS SECUENCIALES

Son aquellas donde las instrucciones están escritas de forma que una acción continúa a la otra, sin posibilidades de recorridos alternativos o diferentes.
Estas acciones o instrucciones son generalmente sentencias de entrada, salida, asignación, etc. El ejemplo de la FIGURA 1, muestra solo estructuras secuenciales.

ESTRUCTURAS DE SELECCIÓN

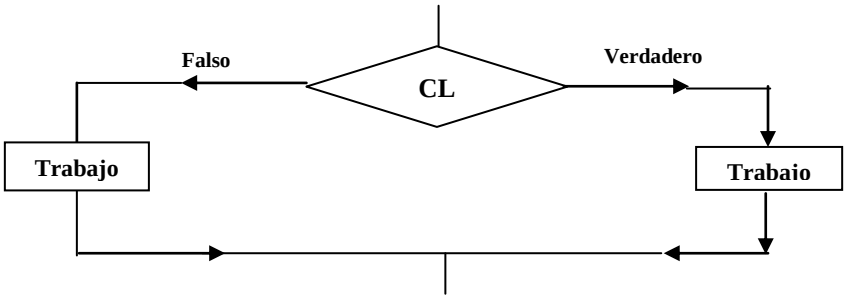
La codificación en lenguaje C de las estructuras de selección es la siguiente:

- ***Estructura de Selección Simple***




```
if (cl)
{
   Trabajo;
}
```

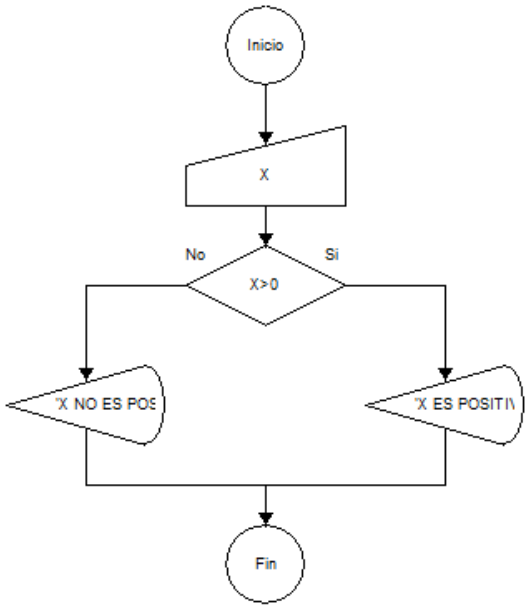
- ***Estructura de Selección Doble***




```
if (cl)
{
   Trabajo;
}
else
{
   Trabajo;
}
```

En este tipo de estructura, cuando hay una sola se la escribe a continuación de la sentencia **if (..)** o de la sentencia **else** sin usar delimitadores (**{ }**) .

Ejemplo: Determinar si un número ingresado por teclado es positivo o negativo (se incluirán en éstos últimos, los ceros que se ingresen). Se imprimirá una leyenda indicando que tipo de número es el ingresado.

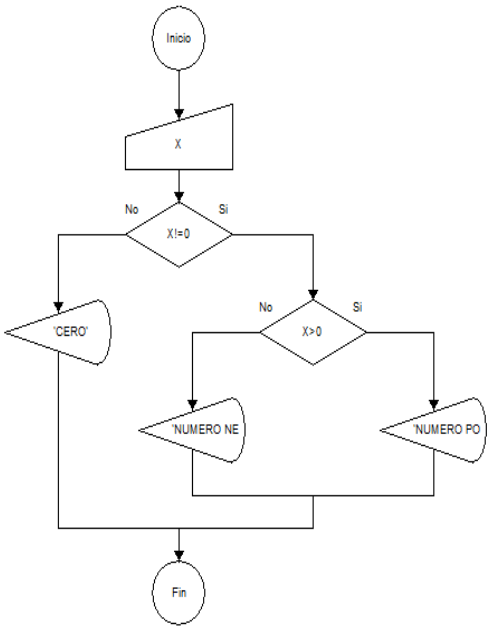


CODIFICACIÓN

```
# include <stdio.h>

int x;

main()
{
        scanf("%d",&x);

         if ( x > 0)
        {
                printf ("X ES POSITIVO");
        }
        else
        {
                printf("X ES POSITIVO");
        }
}
```

- **Estructuras de selección múltiple**

Ejemplo: Ingresar un número y determinar si este es cero, positivo o negativo. Mostrar mensajes indicadores de su condición.



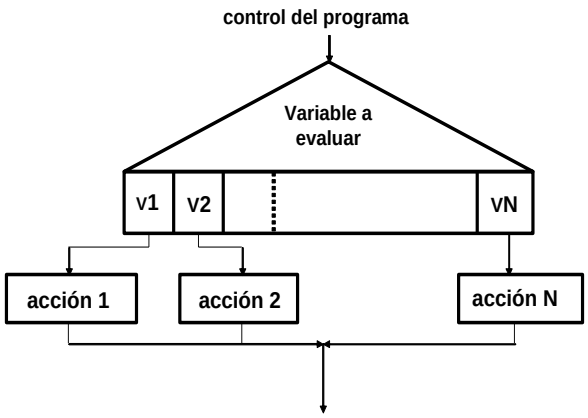
CODIFICACIÓN

```
# include <stdio.h>
int x;
main()
{
        scanf("%d",&x);
         if ( x != 0)
        {
              if (x>=)
              {
                 printf ("NUMERO POSITIVO");
              }
              else
              {
                 printf ("NUMERO NEGATIVO");
              }
        }
        else
        {
                printf("CERO");
        }
}
```

Los dos **else** seguidos pueden llevar a una incorrecta interpretación.
Se debe pensar que el primer **else** que se encuentre (especialmente cuando son varios) corresponde al primer **if** cuando se recorre el programa hacia arriba; el segundo con el siguiente y así sucesivamente.

Existe en la mayoría de los lenguajes de programación, una estructura que reemplaza a los if anidados en un único diagrama con su correspondiente codificación.
Se trata de la estructura **switch** en **C (**en otros lenguajes tienen denominaciones como **case of, etc),** con la particularidad que la variable que controla este tipo de estructura debe ser de tipo ordinal.
Se evalúa la variable de control, y según el valor que tome, se ejecuta la acción correspondiente.

DIAGRAMA DE FLUJO



La sintaxis de la estructura es:

```
switch (v)      //v es la variable cuyo valor se evalúa
{               //v1, v2, v3 son los diferentes valores de v

      case  v1 : <acción 1>
      case  v2: <acción 2>
      ……

      case vn : <acción N>

      default : <acción M>

}
```

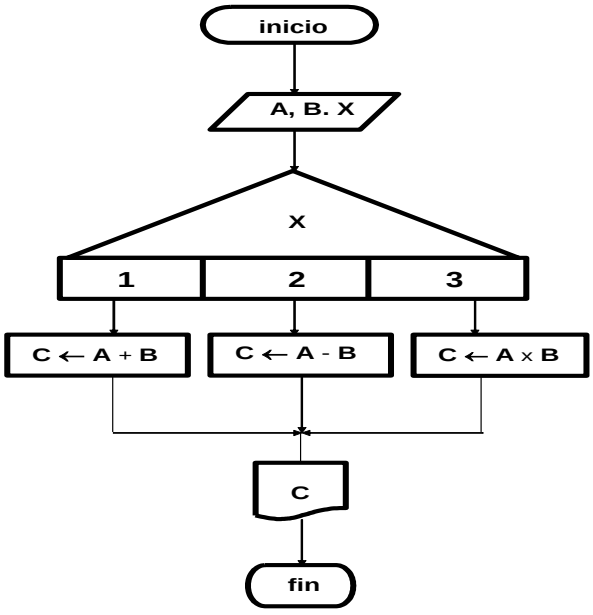
El caso de la directiva **default** indica que esa acción se ejecutará cuando no se hayan seleccionado ninguna de las otras opciones listadas.

EJEMPLO

Ingresar tres números enteros

- realizar la suma de los dos primeros, si el tercer número ingresado es 1;
- realizar la resta, si el tercer número es 2
- multiplicarlos si el tercer número es un 3.

En el presente ejemplo se incluyen los mensajes correspondientes antes del ingreso de datos.



CODIFICACIÓN

```
# include <stdio.h>

int A, B, X, C;

main()
{
  printf("INGRESE EL VALOR DE A:");
  scanf ("%d",&A);
  printf("INGRESE EL VALOR DE B:");
  scanf ("%d",&B);
  printf("INGRESE EL VALOR DE X (1) SUMAR (2) RESTAR (3) MULTIPLICAR:");
  scanf ("%d",&X);
  switch (X)
  {
        case 1 :  C = A + B;
                  printf("RESULTADO: %d",C);
                  break;
        case 2 :  C = A - B;
                  printf("RESULTADO: %d",C);
                  break;
        case 3 :  C = A * B;
                  printf("RESULTADO: %d",C);
                  break;
        default :  printf("No hay operaciones a realizar");
  }
}
```

Las sentencias break se las debe incluir porque de lo contrario se ejecutaran las otras opciones listadas en los siguientes case.
La sentencia break interrumpe la ejecución y sale de la estructura switch.