CIMAT

Informática I para Bachillerato

Introducción a la programación en C/C++
Estructura de Control IF/SWITCH

José Luis Alonzo Velázquez

CIMAT

Sesión 5

¿Que es un diagrama de flujo?

Un diagrama de flujo es una representación gráfica de un algoritmo. Se utiliza en disciplinas como la programación, la economía, los procesos industriales y la psicología cognitiva. Estos diagramas utilizan símbolos con significados bien definidos que representan los pasos del algoritmo, y representan el flujo de ejecución mediante flechas que conectan los puntos de inicio y de término.

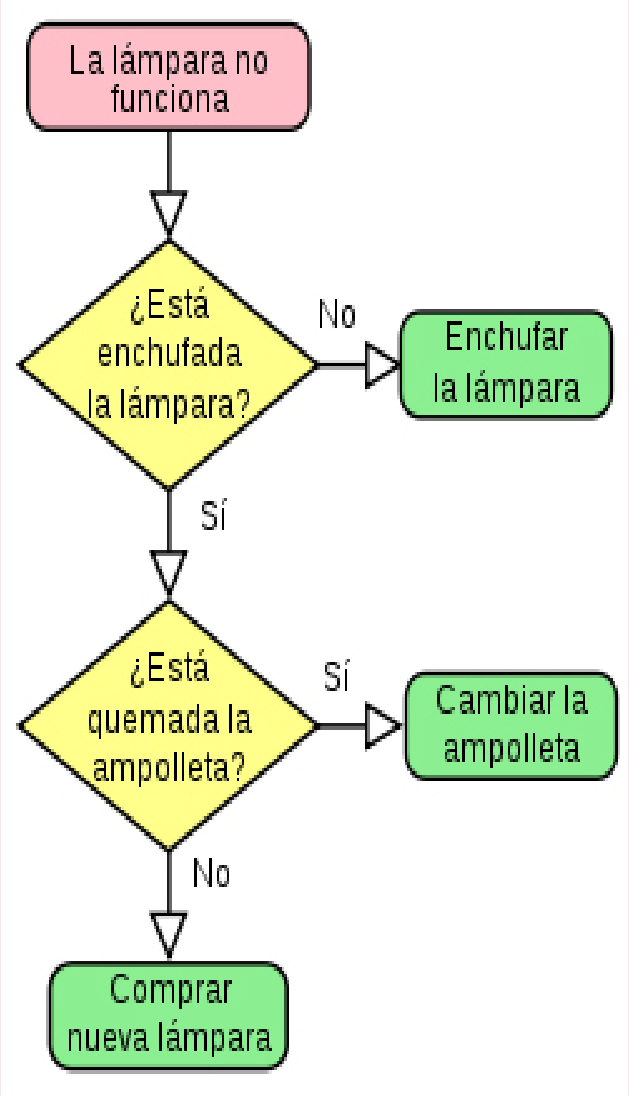
Tipos de Diagramas de Flujo

Tipos de diagramas de flujo
Formato vertical: es la secuencia de las operaciones va de arriba hacia abajo es una lista ordenada de las operaciones de un proceso con toda la información que se considere necesaria según su propósito.
Formato horizontal: en el flujo o la secuencia de las operaciones va de izquierda a derecha.

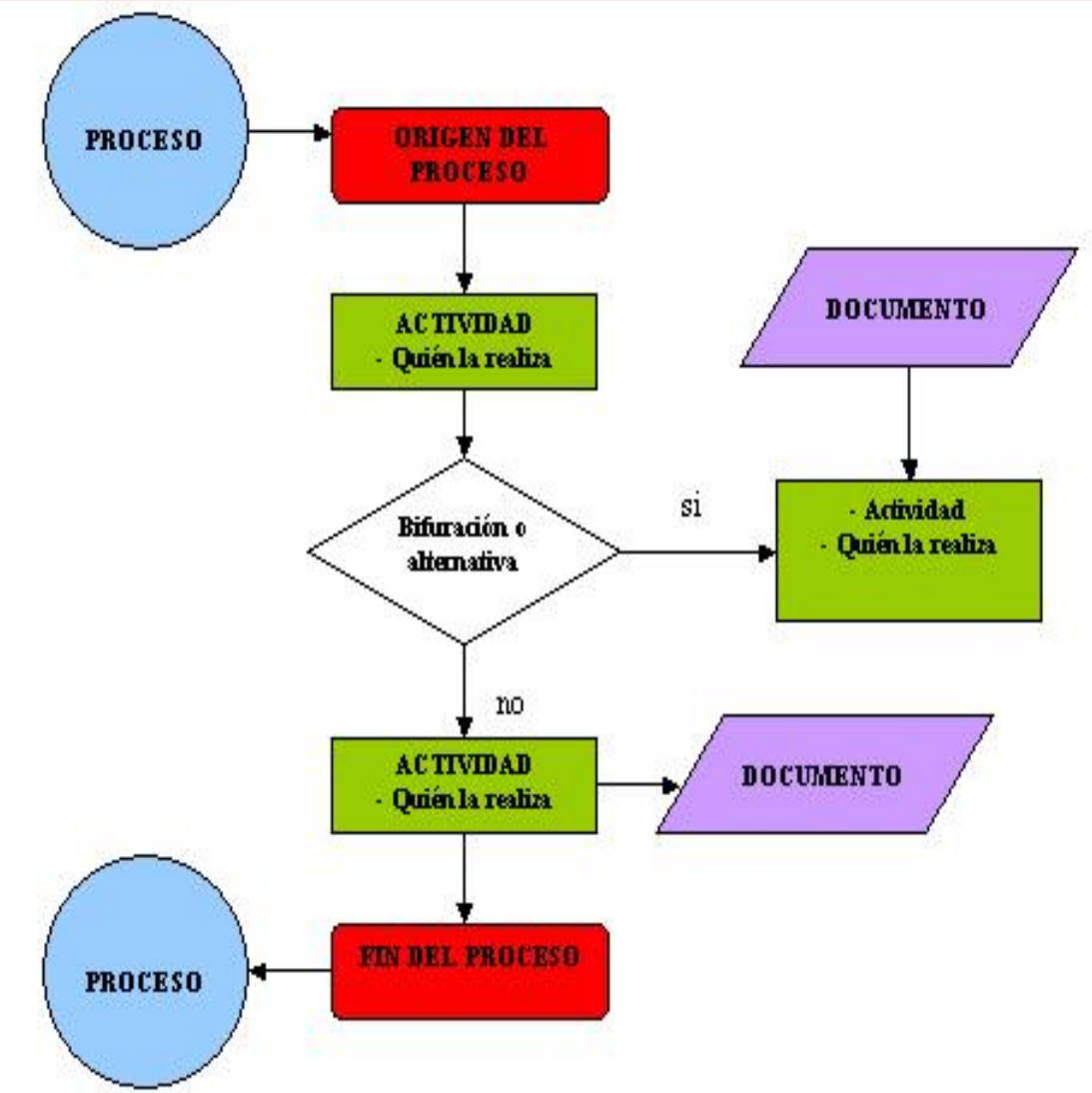
Significado de los símbolos

1. Óvalo: Inicio y término (Abre y/o cierra el diagrama).
2. Rectángulo: Actividad (Representa la ejecución de una o más actividades o procedimientos).
3. Rombo: Decisión (Formula una pregunta o cuestión).
4. Círculo: Conector (Representa el enlace de actividades con otra dentro de un procedimiento).
5. Triángulo boca abajo: Archivo definitivo (Guarda un documento en forma permanente).
6. Triángulo boca arriba: Archivo temporal (Proporciona un tiempo para el almacenamiento del documento).

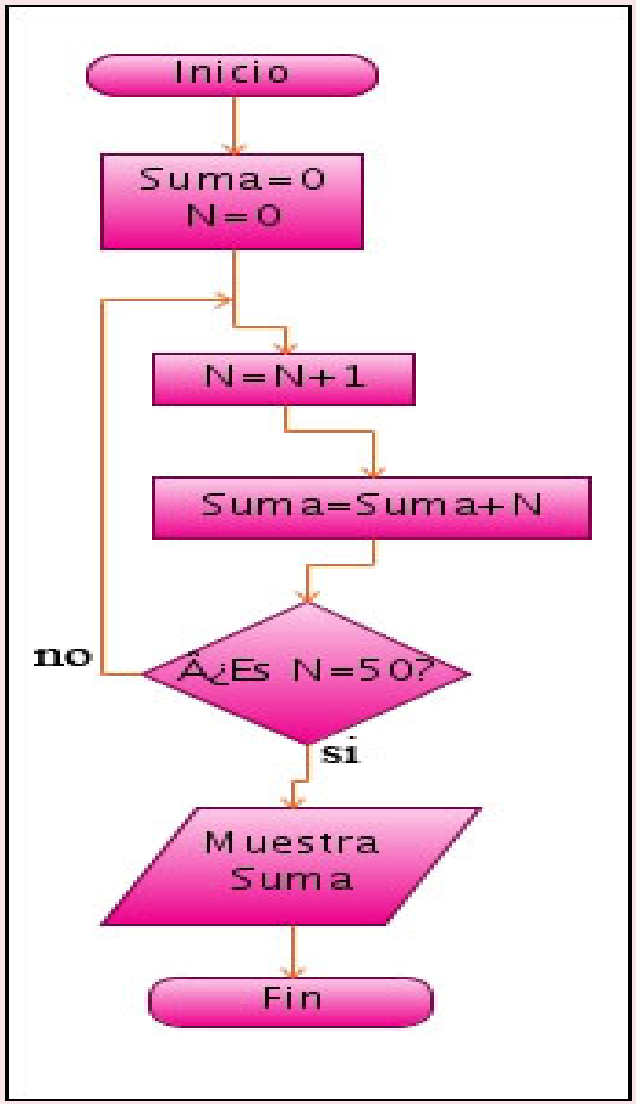
Ejemplo


La lámpara no funciona
¿Está enchufada la lámpara?
No
Enchufar la lámpara
Sí
¿Está quemada la ampolleta?
Sí
Cambiar la ampolleta
No
Comprar nueva lámpara

Estructura Básica



Ejemplo



La instrucción if-else

Es común tener problemas en los cuales se debe proceder de un modo u otro dependiendo de un condicional. Para esto tenemos la instrucción if-else, la sintaxis es la siguiente:

```
if(<condición>)
        {
        <instrucción sección 1>
        <instrucción sección 1>
        ⋮
        <instrucción sección 1>
        }
else
        {
        <instrucción sección 2>
        <instrucción sección 2>
        ⋮
        <instrucción sección 2>
        }
```

Problema:

Crear un programa que pida al usuario la calificación de un estudiante, y a continuación mande un mensaje de aprobado si a calificación es mayor o igual a 7.0 y reprobado si no lo es.

Ejemplo

```
1  #include <stdio.h>
2
3  int main (){
4      float x;
5      printf("Dame tu calificación: ");
6      scanf ("%f",&x);
7      if(x>=7){
8          printf("Aprobado!!!\n");
9      }else{
10         printf("Reprobado!\n");
11     }
12   return 0;
13 }
```

¿Como sería el diagrama de flujo del if-else?

if(<condición>)
{
<instrucción sección 1>
<instrucción sección 1>
⋮
<instrucción sección 1>
}
else
{
<instrucción sección 2>
<instrucción sección 2>
⋮
<instrucción sección 2>
}

Invirtiendo Condicionales

```
if(⟨condicion⟩)
        {
        instrucción 1();
        }
else
        {
        instrucción 2();
        }
```

Invirtiendo Condicionales

```
if(⟨NOcondicion⟩)
        {
        instrucción 2();
        }
else
        {
        instrucción 1();
        }
```

```
if(Condicional X)
        {
        has_nada();
        }
else
        {
        <instruciones>;
        }
```

```
if(condicional inversa a X)
        {
        <instruciones>;
        }
```

Factorización Superior

```
if(Condicional X)
        {
        <instrucion 1>;
        <instrucion 2>;
        }
else
        {
        <instrucion 1>;
        <instrucion 3>;
        }
```

Factorización Superior

```
        <instrucion 1>;
if(Condicional X)
        {
        <instrucion 2>;
        }
else
        {
        <instrucion 3>;
        }
```

Factorización Inferior

```
if(Condicional X)
        {
        <instrucion 1>;
        <instrucion 3>;
        }
else
        {
        <instrucion 2>;
        <instrucion 3>;
        }
```

Factorización Inferior

```
if(Condicional X)
        {
        <instrucion 1>;
        }
else
        {
        <instrucion 2>;
        }
        <instrucion 3>;
```

Doble Condicional Anidada

```
if(Condicional X)
        {
        <instrucion 1>;
        if(Condicional X)
                {
                <instrucion 2>;
                }
        }
```

Se reduce a en muchos casos

```
if(Condicional X)
        {
        <instrucion 1>;
        <instrucion 2>;
        }
```

En que casos no ocurre esto???

Un uso más

```
1  if (opcion == 1)
2      resultado = A + B;
3  else if (opcion == 2)
4      resultado = A - B;
5  else if (opcion == 3)
6      resultado = A * B;
7  else if (opcion == 4)
8      resultado = A / B;
9  else
10     printf("Opcion invalida\n");
```

Estructuras de selección

Aunque la sentencia if de C/C++ es muy potente, en ocasiones su escritura puede resultar tediosa, sobre todo en casos en los que el programa presenta varias elecciones después de chequear una expresión: selección múltiple o multialternativa. En situaciones donde el valor de una expresión determina qué sentencias serán ejecutadas es mejor utilizar una sentencia switch en lugar de una if.

Sintaxis de la estructura de control **switch**

```
switch (selector){
        case ⟨opcion 1⟩:
                ⟨bloque de instrucciones⟩
                break;
        case ⟨opcion 2⟩:
                ⟨bloque de instrucciones⟩
                break;
                        ⋮
        case ⟨opcion n⟩:
                ⟨bloque de instrucciones⟩
                break;
        default:
                ⟨bloque de instrucciones⟩
}
```

Una sentencia **switch** contiene un selector (en el ejemplo, operador), cuyo tipo debe ser int, char o enumerado. Cuando una sentencia **switch** se ejecuta, el valor del selector se compara con las etiquetas case. Si alguna de ellas concuerda con ese valor se ejecutará la correspondiente secuencia de sentencias. Si queremos que varias alternativas tengan el mismo conjunto de sentencias a ejecutar.

Ejemplo

```
switch (selector){
    case 1:
    case 2:
        printf("Salida para los casos 1 y 2\n");
        break;
    case 3:
        printf("Salida para el caso 3\n");
        break;
    default:
        printf("Salida para los restantes casos\n");
}
```

En este ejemplo, si el selector se evalúa y su valor es 1 ó 2, se ejecuta, en ambos casos, la instrucción printf("Salida para los casos 1 y 2"); En este caso particular puede apreciarse la utilidad de **break** a la hora de detener el flujo del programa. La sentencia **switch** puede incluir la opción **default** para establecer la secuencia de sentencias a ejecutar en el caso de que ninguna etiqueta concuerde con el valor de la expresión case. El tipo de esta expresión case y el de las etiquetas tiene que ser el mismo.

Ejemplo

```
1  if (operador == +)
2      resultado = A + B;
3  else if (operador == -)
4      resultado = A - B;
5  else if (operador == *)
6      resultado = A * B;
7  else if (operador == /)
8      resultado = A / B;
9  else
10     printf("Operador invalido");
```

Ejemplo

```
switch (operador) {
    case + :
        resultado = A + B;
        break;
    case - :
        resultado = A - B;
        break;
    case * :
        resultado = A * B;
        break;
    case / :
        resultado = A / B;
        break;
    default :
        printf("Operador invalido");
}
```

La palabra reservada **break** permite que el flujo de programa se detenga justo después de la ejecución de la sentencia anterior a ese **break**, impidiendo que se ejecuten las sentencias correspondientes a las siguientes alternativas del **switch**. Por tanto, debemos obligatoriamente acabar cada bloque de sentencias correspondiente a cada alternativa con una sentencia **break**.

Por otro lado, la alternativa **default** es opcional y engloba un conjunto de sentencias (que puede ser vacío, contener una sola sentencia o varias) que se ejecutan en caso de que ninguna de las alternativas del switch tenga un valor coincidente con el resultado de evaluar la expresión del selector.

Problema para clase

Hacer un menu que despliegue lo siguiente en pantalla:
Escoja una opción:
a)opcion 1
b)opcion 2
c)opcion 3
s)salir
si se escoge la opcion 1 imprima "Se eligio la opcion 1",
analogamente 2 y 3.

 Como Programar en C/C++, Deitel (Prentice Hall), 2da Edición.

 Programming Principles and Practice Using C++, Bjarne Stroustrup.

 http://www.codeblocks.org

 http://www.wxwidgets.org