Violeta Ocegueda
Profesor-Investigador
Facultad de Ciencias Químicas e Ingeniería
Universidad Autónoma de Baja California
Campus Tijuana

Programación

Tronco Común de Ingeniería

FCQI

Contenido

Estructuras de selección

Estructuras de selección

Regulan el flujo de ejecución de un programa, permiten combinar instrucciones o sentencias individuales en una simple unidad lógica con un punto de entrada y un punto de salida.

L Joyanes Aguilar (2014). Programacion En C/C++ Java Y Uml (2da ed.). México, D.F.: McGrawHill.

Estructuras de selección

Las estructuras algorítmicas selectivas que se utilizan para la toma de decisiones lógicas las podemos clasificar de la siguiente forma:

- **SI - ENTONCES:** selección sencilla.
- **SI - ENTONCES / SINO:** selección doble.
- **SI MULTIPLE:** selección múltiple.

Selección sencilla

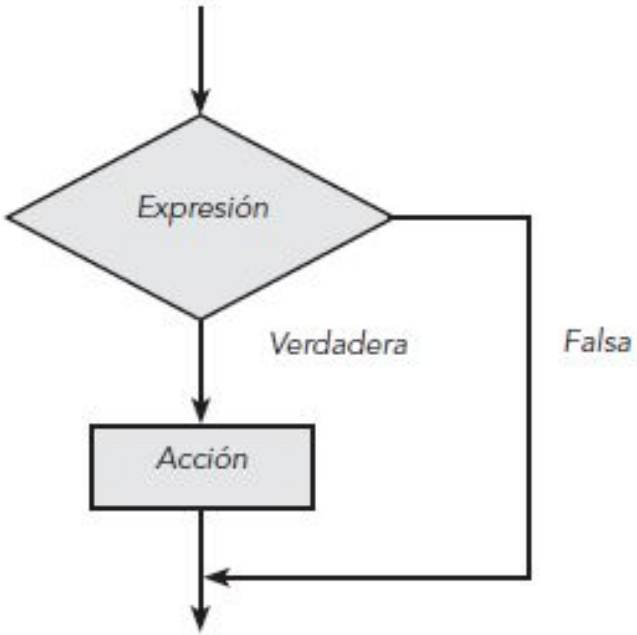
En C, la estructura de control de selección principal es una sentencia **if**. El formato tiene la sintaxis siguiente:

```
if(expresion)
{
    /*Acción*/
}
```

expresion: Expresión lógica que determina si la acción se ha de ejecutar.
Acción: Se ejecuta si la expresión lógica es verdadera.

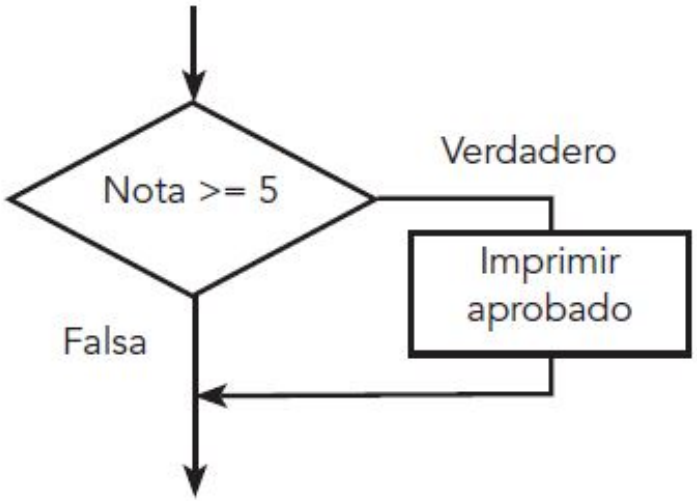
Selección sencilla

Diagrama de flujo de una sentencia básica *if*



Selección sencilla

Ejemplo: Representar la superación de un examen (Nota >= 5, Aprobado).



Selección sencilla

```c
#include <stdio.h>

void main()
{
    float nota;
    printf("Introduzca la nota obtenida (0-10): ");
    scanf("%f", &nota);
    /* compara nota con 5 */
    if (nota >= 5)
        printf("Aprobado");
}
```

Selección doble

En este formato la sentencia **if** tiene la siguiente sintaxis:

_if_ (Expresión)

Expresión lógica que determina la acción a ejecutar

Accion$_1$

Acción que se realiza si la expresión lógica es verdadera.

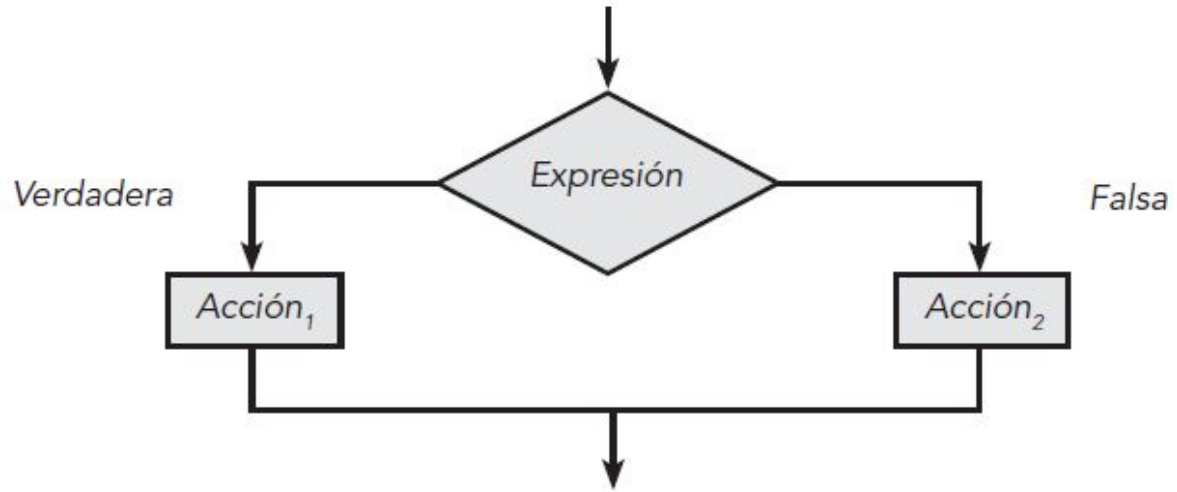
else Accion$_2$

Acción que se ejecuta si la expresión lógica es falsa.

Selección doble

Acción$_1$ y Acción$_2$, son individualmente, o bien una única instrucción que termina en un punto y coma (;) o un grupo de instrucciones encerradas entre llaves.

Si *Expresión* es verdadera, se ejecuta *Acción*$_1$ y en caso contrario se ejecuta *Acción*$_2$



Selección doble

Ejemplo: Calcular el mayor de dos números leídos del teclado y visualizarlo en pantalla.

```c
#include <stdio.h>
int main()
{
    int valor1, valor2;
    printf("Ingrese el valor 1: ");
    scanf("%d", &valor1);
    printf("Ingrese el valor 2: ");
    scanf("%d", &valor2);
    if(valor1>valor2)
        printf("El mayor es: %d", valor1);
    else
        printf("El mayor es: %d", valor2);
    return 0;
}
```

Selección Multiple

Una sentencia **if** es anidada cuando la sentencia de la rama verdadera o la rama falsa es a su vez una sentencia if. Una sentencia **if** anidada se puede utilizar para implementar decisiones con varias alternativas o multialternativas.

SINTAXIS:

```
if(condición_1)
    sentencia_1
else if(condicion_2)
    sentencia_2
    .
    .
else if(condicion_n-1)
    sentencia_n-1
else
    sentencia_n
```

Selección Multiple

La sentencia **switch** es una sentencia que se utiliza para seleccionar una de múltiples alternativas. Es especialmente útil cuando la selección se basa en el valor de una variable simple o de una expresión simple denominada ***expresión de control*** o ***selector***. El valor de esta expresión puede ser de tipo ***int*** o ***char***, pero no de tipo float ni double.

Selección Multiple

SINTAXIS

```
switch (selector)
{
    case etiqueta1 : sentencias1;
    case etiqueta2 : sentencias2;
    .
    .
    case etiquetan : sentenciasn;
    default: sentencias; /* opcional */
}
```

Selección Multiple

La expresión de control o **selector** se evalúa y se compara con cada una de las etiquetas de **case**. Cada etiqueta es un valor único, constante y cada etiqueta debe tener un valor diferente de los otros.

Si el valor de la expresión selector es igual a una de las etiquetas case, por ejemplo, etiqueta1, entonces la ejecución comenzará con la primera sentencia de la secuencia y continuará hasta que se encuentre el final de la sentencia de control switch, o hasta encontrar la sentencia **break**.

Típicamente después de cada bloque se sitúa la sentencia **break** como última sentencia del bloque. La sentencia **break** hace que siga la ejecución en la siguiente sentencia al switch.

Selección Multiple

SINTAXIS CON break

```
switch (selector)
{
    case etiqueta1 : sentencias1;
    break;
    case etiqueta2 : sentencias2;
    break;
    .
    .
    .
    case etiquetan :sentenciasn;
    break;
    default: sentenciasd; /* opcional */
}
```

Selección Multiple

Elección de tres opciones y un valor en forma predeterminada.

```
switch (opcion)
{
    case 0:
        printf("Cero!");
        break;
    case 1:
        printf("Uno!");
        break;
    case 2:
        printf("Dos!");
        break;
    default:
        printf("Fuera de rango");
}
```

Selección Multiple

Selección múltiple, tres etiquetas ejecutan la misma sentencia.

```
switch (opcion)
{
    case 0:
    case 1:
    case 2:
        printf("Menor de 3");
        break;
    case 3:
        printf("Igual a 3");
        break;
    default:
        printf("Mayor que 3");
}
```

Selección Multiple

Comparación de las sentencias if-else-if y switch. Se necesita saber si un determinado carácter car es una vocal. Solución con if-else-if.

```
if((car=='a')||(car=='A'))
   printf("%c es una vocal\n",car)
         ;
else if((car=='e')||(car=='E'))
   printf("%c es vocal\n",car);
else if((car=='i')||(car=='I'))
   printf("%c es vocal\n",car);
else if((car=='o')||(car=='O'))
   printf("%c es vocal\n",car);
else if((car=='u')||(car=='U'))
   printf("%c es vocal\n",car);
else
   printf("%c no es vocal\n",car);
```

```
switch (car)
{
     case 'a': case 'A':
     case 'e': case 'E':
     case 'i': case 'I':
     case 'o': case 'O':
     case 'u': case 'U':
          printf("%c es una vocal\n",car);
          break;
     default:
        printf("%c no es una vocal\n",car);
}
```

Gracias