Toma de decisiones en C

Estructuras de Control Selectivas

En un programa la ejecución es **secuencial**, es decir, las instrucciones se ejecutan **una después de la otra** hasta que finalice el programa.

¿Cómo hacemos si queremos tomar una decisión en tiempo de ejecución?

Las **estructuras selectivas** se utilizan para realizar la "toma de decisiones" y luego *ramificar* el flujo del programa en función del resultado de la toma de decisiones.

Condición para tomar la decisión

Acción 1

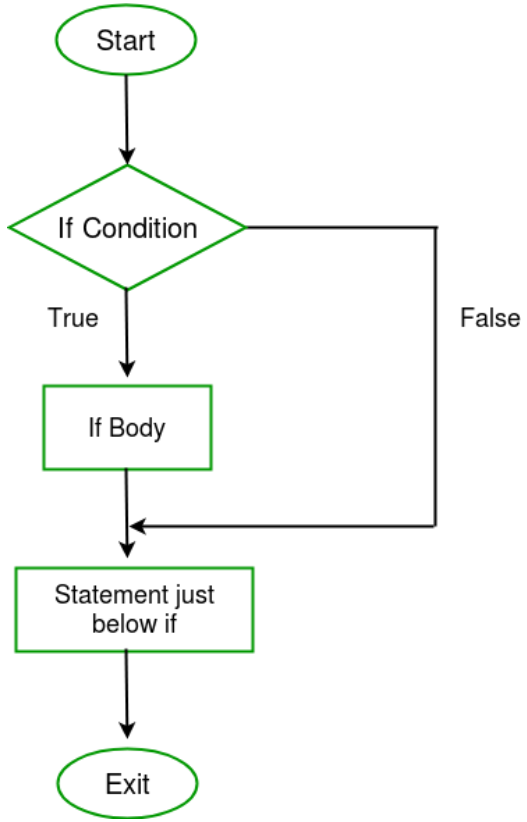
.

.

.

Acción n

Si la expresión se evalúa como **verdadera** las instrucciones dentro del cuerpo de if se ejecutan. De lo contrario, se omite y continua con la próxima instrucción.




```
if(condición){
    // Instrucciones a ejecutar
    // si la condición se evalúa
    // como verdadera
}

// Instrucciones posteriores
```

La condición es una **expresión lógica**, por lo tanto, después de la evaluación será verdadera o falsa.

Una ***expresión lógica*** es aquella en la que intervienen los operadores relacionales (==,>=,!=,etc) o lógicos (!,&&, ||).

Ejemplos

```
int edad = 20;

if(edad >= 18){
    printf(“Es adulto”);
}
```

Salida:

```
Es adulto
```

```
int edad = 16;

if(edad >= 18){
    printf(“Es adulto”);
}
```

No muestra nada porque la condición es falsa

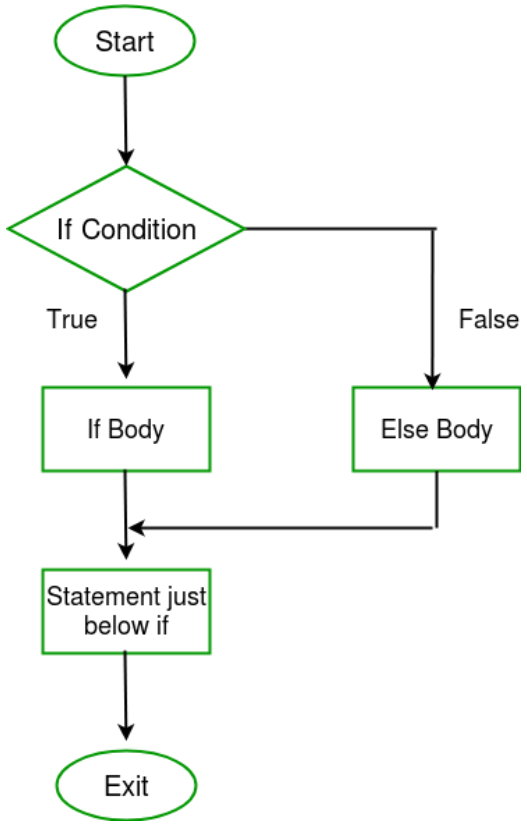
¡Problema!

¿Qué pasa si queremos realizar determinada acción si la expresión resulta falsa?

La declaración if es la declaración de toma de decisiones mas simple.

if
else

Si la expresión se evalúa como **verdadera** las instrucciones dentro del cuerpo de if se ejecutan. Si la expresión se evalúa como **falsa**, las declaraciones dentro del cuerpo de else se ejecutan.




```
if(condición){
    // Instrucciones a ejecutar
    // si la condición se evalúa
    // como verdadera
}
else{
    // Instrucciones a ejecutar
    // si la condición previa
    // se evalúa como falsa
}

// Instrucciones posteriores
```

La instrucción ***else*** no necesita evaluar una condición.

Si la condición previa se evalúa como verdadera, se omiten las declaraciones dentro del cuerpo de else.

Ejemplo

```
int edad = 17;

if(edad >= 18){
    printf(“Es adulto”);
}
else{
    printf(“Es adolescente”);
}
```

Salida: `Es adolescente`

¡Problema!

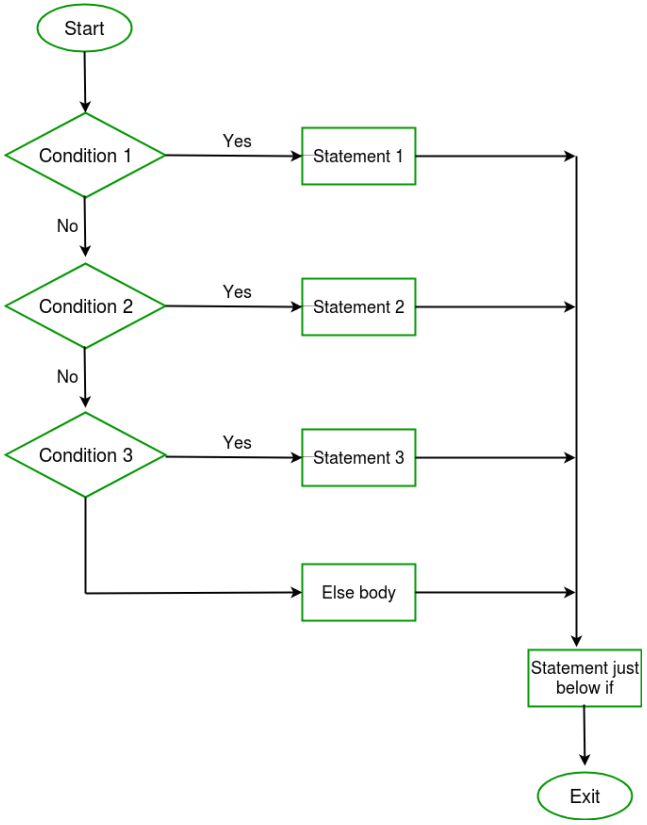
¿Y qué pasa si queremos decidir entre múltiples opciones?

La instrucción if else solo permite ejecutar dos códigos diferentes, dependiendo de si la expresión es verdadera o falsa.

if
else if
else

La instrucción if else if es una extensión de la instrucción if else. Si una condición es **verdadera**, las instrucciones definidas en el bloque if se ejecutarán; de lo contrario, si alguna otra condición es verdadera, las instrucciones definidas en el bloque else if se ejecutará. Si ninguna de las condiciones es verdadera, se ejecutarán las declaraciones definidas en el bloque else.

```
if(condición)
    // Instrucciones a ejecutar
    // si la condición se evalúa
    // como verdadera
}
else if(condición){
    // Instrucciones a ejecutar
    // si la condición se evalúa
    // como verdadera
}
.
.
.
else{
    // Instrucciones a ejecutar
    // si ninguna de las
    // condiciones se evalúa como
    // verdadera
}
```



Ejemplo

```
int edad = 8;

if(edad >= 18){
    printf(“Es adulto”);
}
else if(edad =< 12){
    printf(“Es infante”);
}
else{
    printf(“Es adolescente”);
}
```

Salida: Es infante

¡Problema!

¿Y qué pasa si tomar una decisión nos lleva a tener que tomar otra?

if anidado

Siempre es legal anidar declaraciones if, if else if else o if else, lo que significa que se puede usar una de las instrucciones anteriores dentro de otra instrucción if o else if.

```
if(condición){
    if(condición){
        // Instrucciones
    }
}
```

```
if(condición){
    if(condición){
        // Instrucciones
    }
    else if(condición){
        // Instrucciones
    }
    else{
        // Instrucciones
    }
}
```

Ejemplo

```
int edad = 8;

if(edad < 18){
    if(edad =< 12){
        printf(“Es infante”);
    }
    else{
        printf(“Es adolescente”);
    }
}
else{
    printf(“Es adulto”);
}
```

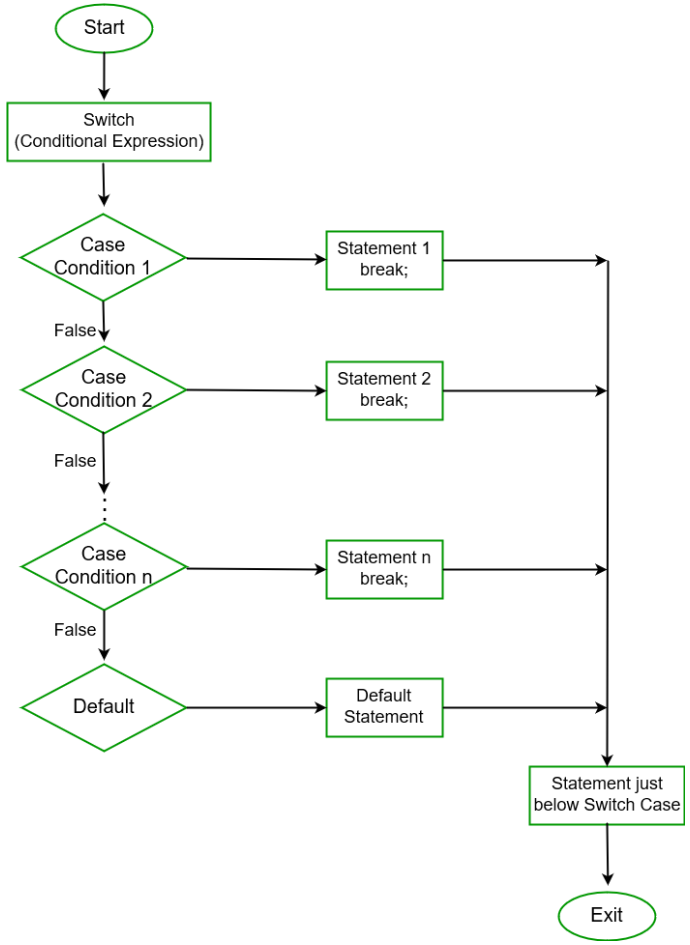
Salida: Es infante

Alternativa para realizar comparaciones: instrucción switch

switch ⟹

El valor de una variable se compara con los valores de cada caso. Si hay una coincidencia, se ejecutan las instrucciones correspondientes después del caso coincidente. Si no hay coincidencia, se ejecutan las instrucciones por defecto

```
switch(variable){
    case 1:
        // Código
        break;
    case 2:
        // Código
        break;
    .
    .
    .
    default:
        // Código
}
```



Ejemplo

Salida: a+b=15.

Si el operador no hubiera coincidido con ninguno de los tres casos se mostraría a*b=50.

```
char operador = '+';
int a = 10;
int b = 5;

switch(operador){
    case '+':
        printf("a+b=%i", a+b);
        break;
    case '-':
        printf("a-b=%i", a-b);
        break;
    case '/':
        printf("a/b=%i", a/b);
        break;
    default:
        printf("a*b=%i", a*b);
}
```

Analogía entre switch e if

```
char operador = ‘+’;
int a = 10;
int b = 5;

if(operador == ‘+’){
    printf(“a+b=%i”, a+b);
}
else if(operador == ‘-’){
    printf(“a-b=%i”, a-b);
}
else if(operador == ‘/’){
    printf(“a/b=%i”, a/b);
}
else{
    printf(“a*b=%i”, a*b);
}
```

Reglas

- No se permiten valores de caso duplicados.
- Lo que se evalúa debe ser el valor de una variable. No se pueden evaluar, a diferencia del if, expresiones lógicas.
- El valor de un caso debe ser del mismo tipo de dato del valor en cuestión, es decir, el de la variable que se esta evaluando.
- El valor de un caso debe ser constante o literal. No se permiten variables.
- La declaración de interrupción (break) se utiliza dentro de un caso para terminar una secuencia de instrucciones.
- El break es opcional, pero si se omite, la ejecución continuará en el siguiente caso.
- La declaración predeterminada (default) es opcional.
- Puede haber como máximo una declaración predeterminada.
- Es útil cuando se deben realizar muchas comparaciones. Es una cuestión de legibilidad de código.