

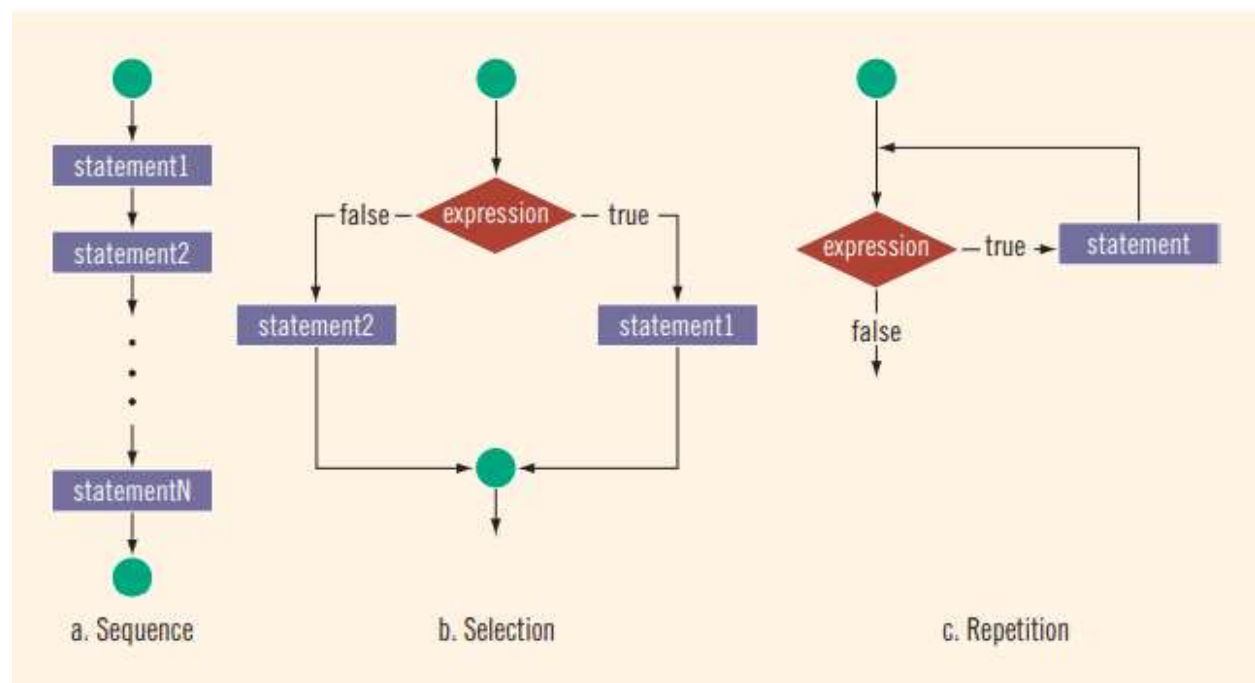


Estructuras de Control en C++

Una computadora puede procesar un programa de una de las siguientes maneras:

- En secuencia;
- Selectivamente, haciendo una elección;
- Repetidamente, ejecutando una declaración una y otra vez, usando una estructura llamada bucle;
- O llamando a una función.

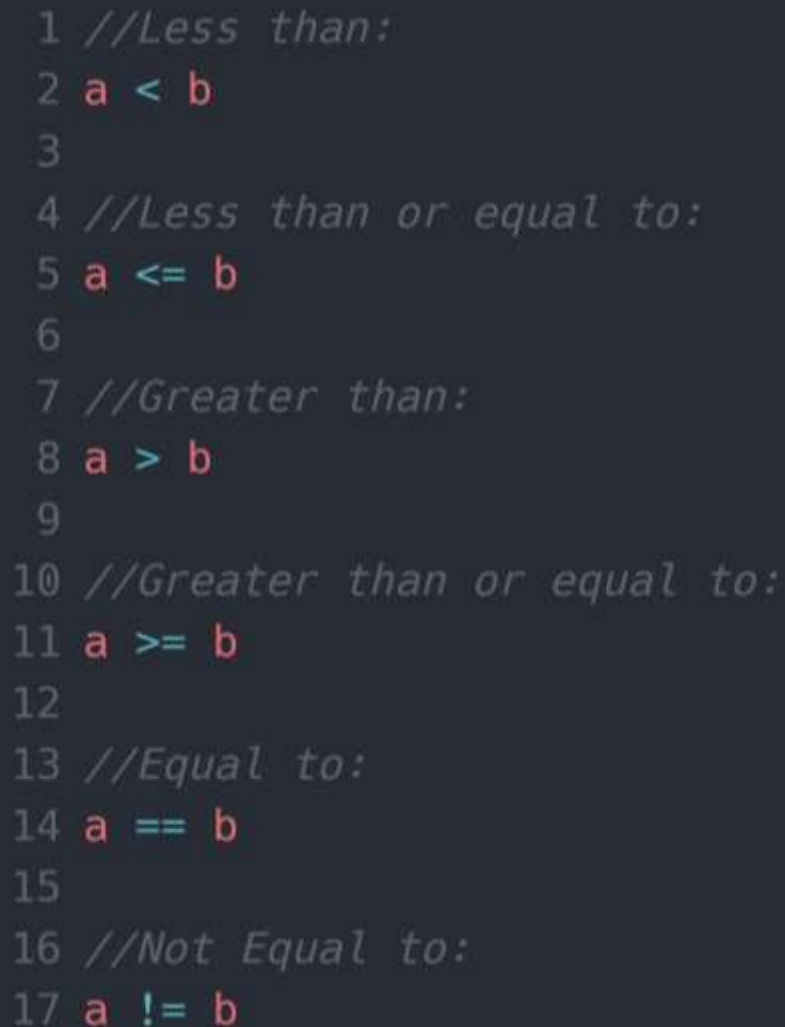
Las clases de programación anteriores incluían secuencias secuenciales simples en los programas. Con un programa de este tipo, la computadora comienza por el principio y sigue las declaraciones en orden. No se toman decisiones; no hay repetición. Las Estructuras de Control proporcionan alternativas a la ejecución secuencial del programa y se utilizan para alterar la secuencia flujo de ejecución. Las dos estructuras de control más comunes son la selección y la repetición. En la selección, el programa ejecuta declaraciones particulares dependiendo de algunas condiciones. En repetición, el programa repite declaraciones particulares un cierto número de veces basándose en alguna condición (es).



Estructuras de Control (Selección o Condicional)

En las Estructuras Condicionales de C++ tenemos a nuestra disposición el IF y el SWITCH:


C++ admite las operaciones de comparación habituales de las matemáticas:



```
1 //Less than:
2 a < b
3
4 //Less than or equal to:
5 a <= b
6
7 //Greater than:
8 a > b
9
10 //Greater than or equal to:
11 a >= b
12
13 //Equal to:
14 a == b
15
16 //Not Equal to:
17 a != b
```

Puede utilizar estas condiciones para realizar diferentes acciones para diferentes decisiones.

Para evaluar más de una condición lógica es necesario usar los operadores lógicos:



```
1 //not == ~  
2 !  
3  
4 //and == Y  
5 &&  
6  
7 //or == 0  
8 ||
```

Para trabajar con expresiones lógicas complejas, debe haber algún esquema de prioridad para los operadores evaluados. La siguiente tabla muestra el orden de precedencia de algunos operadores de C ++, incluyendo la aritmética.

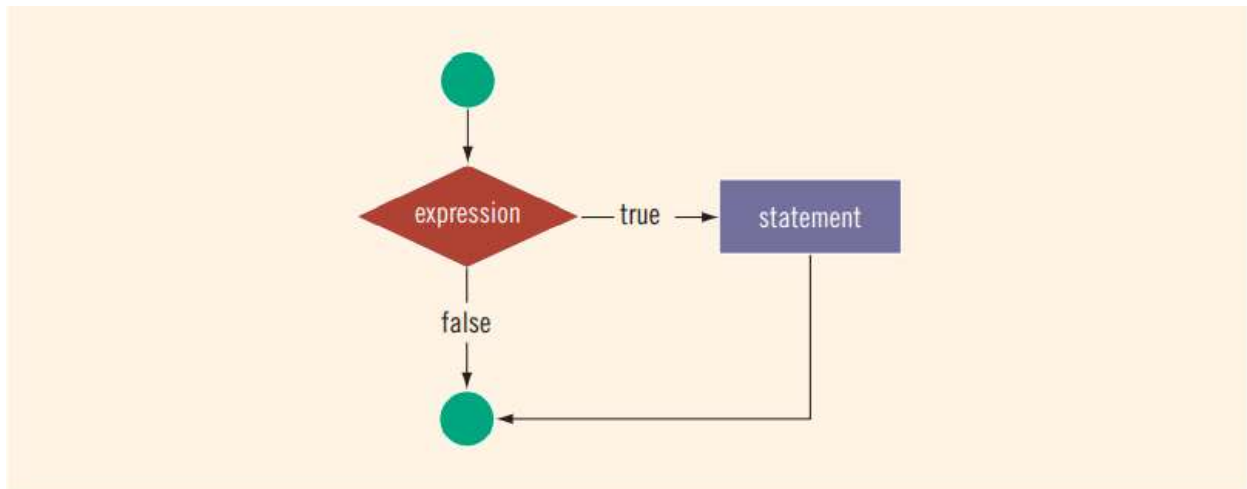
Operators	Precedence
!, +, - (unary operators)	first
*, /, %	second
+, -	third
<, <=, >=, >	fourth
==, !=	fifth
&&	sixth
	seventh
= (assignment operator)	last

Usando las reglas de precedencia en una expresión, se evalúan los operadores lógicos y relacionales de izquierda a derecha. Debido a que los operadores lógicos y relacionales se evalúan de izquierda a derecha, Se dice que la asociatividad de estos operadores es de izquierda a derecha.

La declaración IF

C++ tiene las siguientes declaraciones condicionales en IF:

- Use **if** para especificar un bloque de código que se ejecutará, si una condición especificada es verdadera.
- Use **else** para especificar un bloque de código que se ejecutará, si la misma condición es falsa.
- Use **else if** para especificar una nueva condición para probar, si la primera condición es falsa.



Selección unidireccional (One-Way Selection)

```
1 //one-way selection
2 if (expresion)
3     //code
```

Selección bidireccional (Two-Way Selection)



```
1 //two-way selection
2 if (expresion)
3     //code
4 else
5     //code
```

Bloque compuesto de declaraciones

Las estructuras if y if ... else controlan solo una declaración a la vez. Suponga, sin embargo, que desea ejecutar más de una instrucción si la expresión en un if o if ... else declaración se evalúa como verdadera. Para permitir declaraciones más complejas, C ++ proporciona una estructura llamada declaración compuesta o bloque de declaraciones.

Es decir, un enunciado compuesto consta de una secuencia de enunciados encerrados en rizado llaves, { y }. En una estructura if o if ... else, una sentencia compuesta funciona como si fue una sola declaración.

Selecciones múltiples: IF anidados

Puede incluir varias rutas de selección en un programa utilizando un if ... else si la declaración de acción en sí es una declaración if o if ... else. Cuando una instrucción de control se encuentra dentro de otra, se dice que está anidada.

Emparejamiento de un else con un if en una instrucción if anidada, C ++ asocia un else con el if reciente incompleto, es decir, el más reciente si no se ha emparejado con otro.

Selección de múltiples vías (Multiple-Way Selection)



```
1 //multiple-way selection
2 if (condition1) {
3     //code
4 } else if (condition2) {
5     //code
6 } else {
7     //code
8 }
```

Operador ternario (Ternary Operator)



```
1 //Ternary Operator
2 variable = (condition) ? expressionTrue : expressionFalse;
```