Fundamentos de Programación 101

By Ernie

Ernesto José Canales Guillén

Círculos de estudio UCA

Ciclo Virtual 01/2021

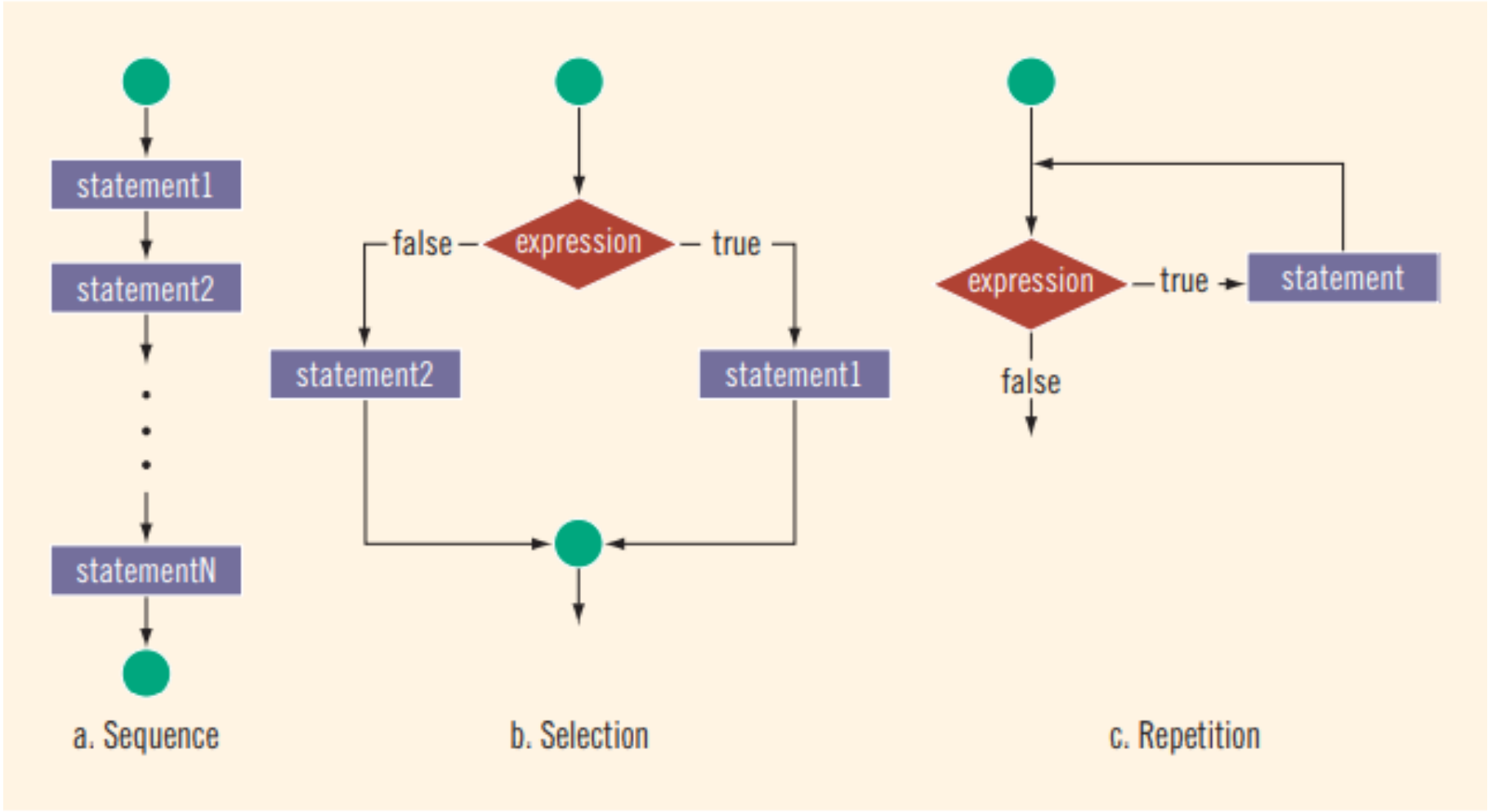
Estructuras de Control en C++

Selección y Repetición

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 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.


statement1
statement2
statementN
a. Sequence
false
expression
true
statement2
statement1
b. Selection
expression
true
statement
false
c. Repetition

Estructuras de Control (Selección o Condicional)

Estructuras de Selección o Condicional

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

Operadores de comparación

Se utilizan para comparar dos valores. El valor de retorno de una comparación es verdadero (1) o falso (0).

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Operadores lógicos

Se utilizan para determinar la lógica entre variables o valores.

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Trabajando con expresiones lógicas

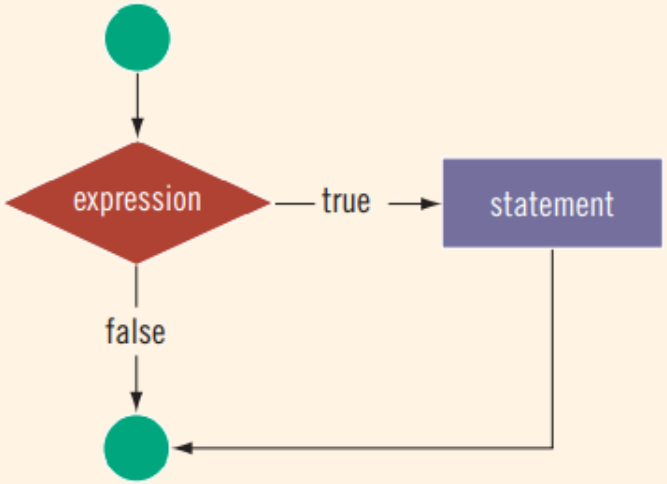
Para trabajar con expresiones lógicas complejas, debe haber algún esquema de prioridad para los operadores evaluados. La siguiente tabla muestra el orden de precedencia de algunos operadores de C ++, incluyendo la aritmética.

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

El *if*

if, if … else, if … else if … else

- 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. Cuándo 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;
```

Bibliografía

- L. J. Aguilar, Programación en C++. Algoritmos, estructuras de datos y objetos, Aravaca (Madrid): McGRAW-HILL, 2006.

- D. Malik, C++ Programming: From Problem Analysis to Program Design, Boston, MA: Cengage Learning, 2003.