# Chapter 6. Conditional operator

## 6.1 The concept of a branching algorithm

Branching algorithms are algorithms that have several alternative paths, the choice of which depends on the fulfillment of certain conditions. Branching is an algorithmic construct in which, depending on the result of a condition check (true or false), one of two consecutive commands, called branches, is executed [17].

The if statement is designed to implement a branching algorithm. A conditional construct in C++ is always written in parentheses after the if operator. The body of the condition is specified inside the curly brackets. If the condition is true, the commands between the curly braces (in the body of the condition) are executed. Otherwise, the operations following the word else are executed. If only one operator is to be executed after the word if or after the word else, the curly braces can be omitted. The syntax of the branching structure is shown in figure 6.1 [25].

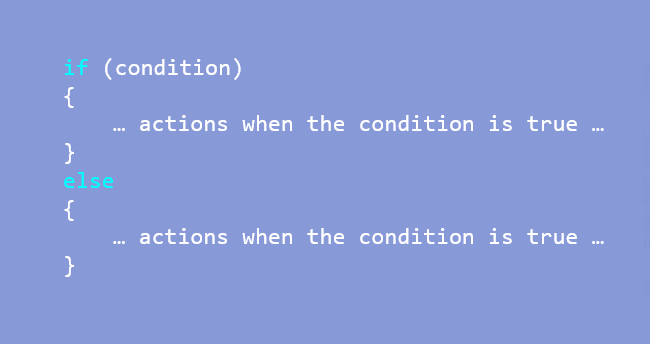


Figure 6.1 – Syntax of the if statement branching structure

The structure of an if statement can represent either full or incomplete branching. A branching is called incomplete if there is no else branch, then in case of a false condition the action passes to the operator located after the branching. The structure of full and incomplete branching is shown in block diagrams (figure 6.2).

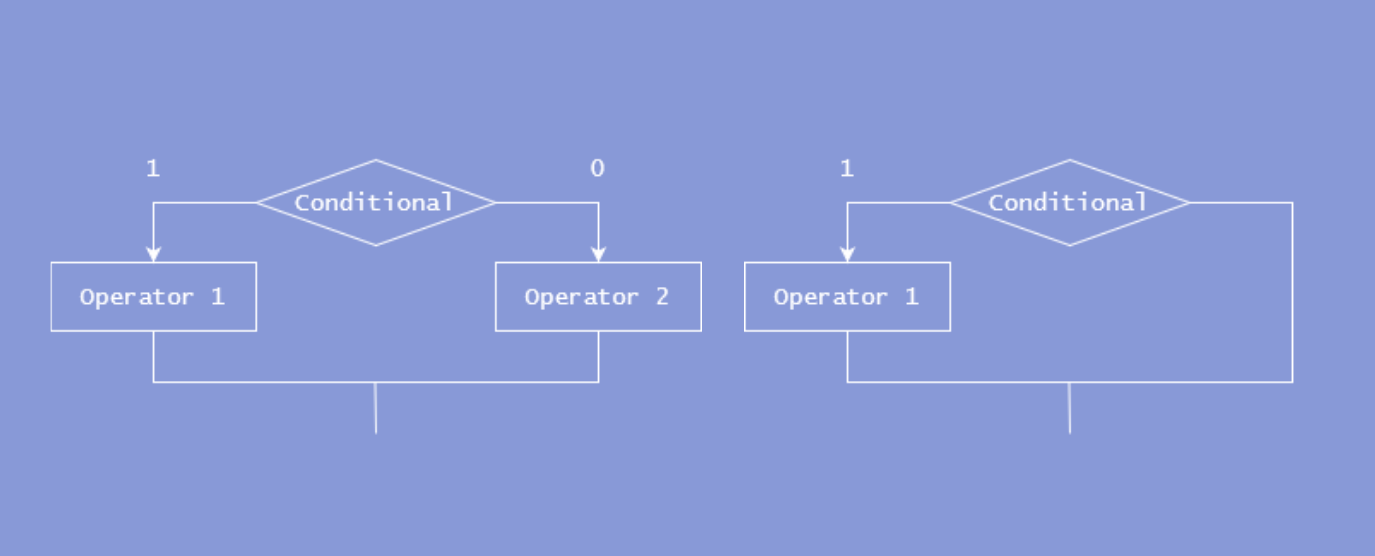


Figure 6.2 – Block diagrams of full and incomplete branching

The condition in the if statement does not have to be an expression. It can be any integer variable, and if the value of the variable is different from zero, the execution of the algorithm will follow the true branch, otherwise it will follow the false branch. It can also be a logical variable that has the bool data type.

To consolidate the material it is suggested to perform the tasks below.

Task 1:

The program is executed twice with the following input data:

1. value = -10;

2. value = 9;

It is necessary to explain what will be output as a result of the program execution (figure 6.3).

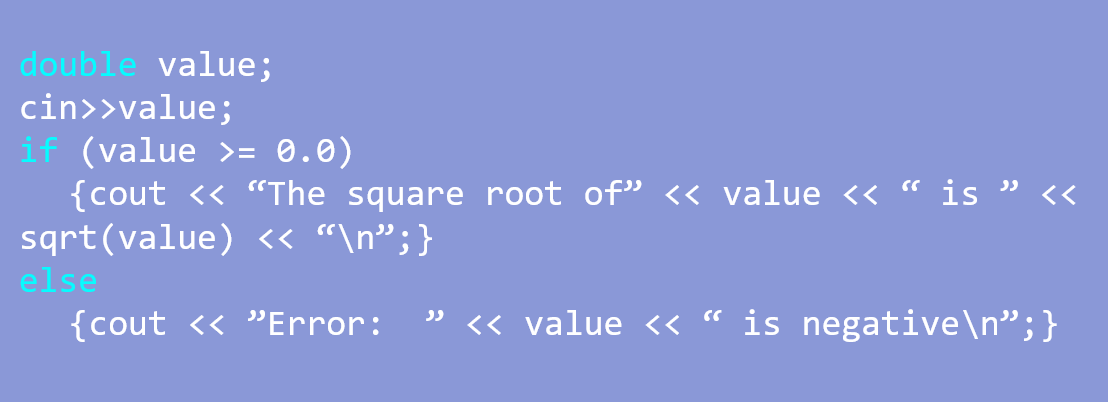


Figure 6.3 – Program code fragment of task №1

Task 2:

The program is executed twice with the following input data:

1) a = 5;

2) a = 16;

It is necessary to explain what will be output as a result of the program execution (figure 6.4).

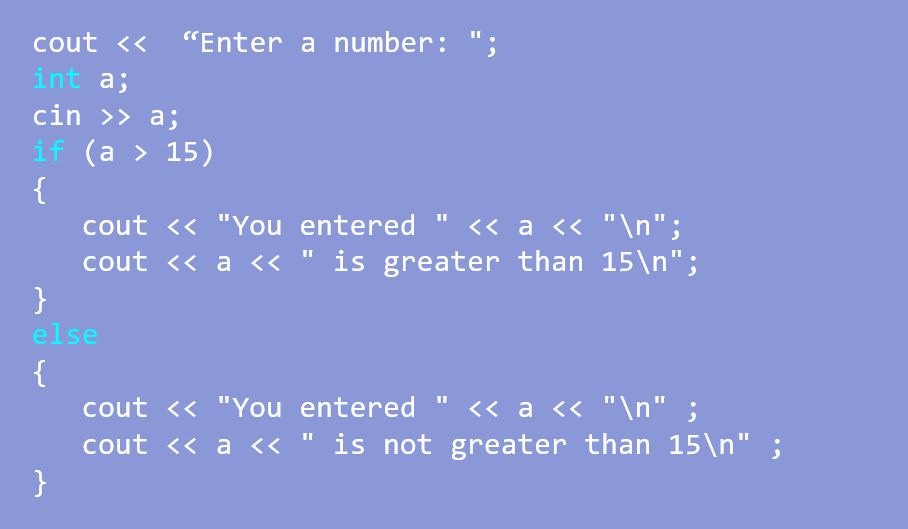


Figure 6.4 – Program code fragment of task №2

## 6.2 The concept of a nested conditional operator

A construct in which another condition must be checked within one condition is called a nested conditional statement. Nested conditional statements help to make a program more flexible. An example of using the nested conditional operator construct is finding the minimum number of three numbers (figure 6.5).

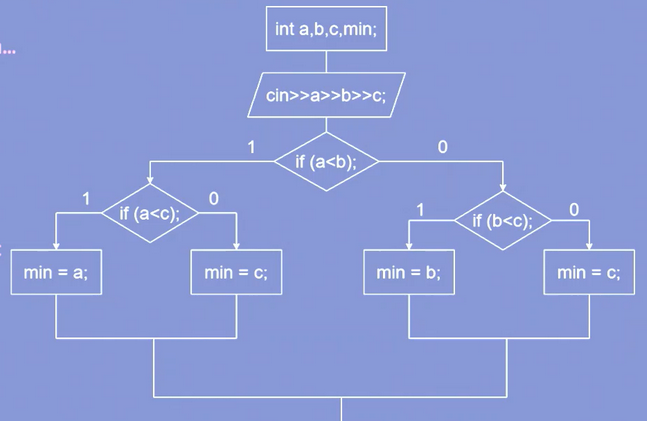


Figure 6.5 – Block diagram of the algorithm for finding the minimum number using a nested conditional operator

Task 3:

The program is executed twice with the following input data:

1. num = 10.5;

2. num = 10;

It is necessary to explain what will be output as a result of the program execution (figure 6.6).

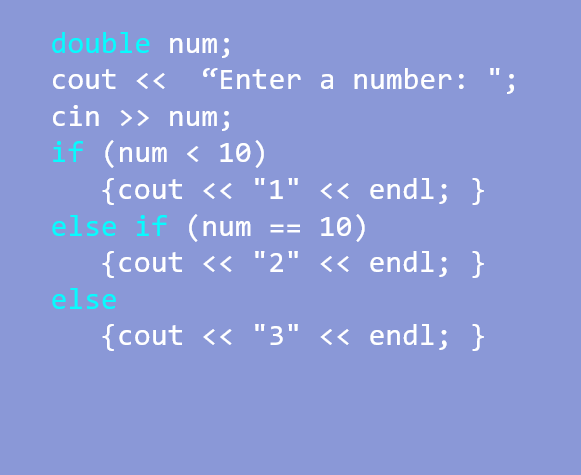


Figure 6.6 – Program code fragment of task №3

Task 4:

The program is executed twice with the following input data:

1. a = 6;

2. a = 15;

3. a = 17;

It is necessary to explain what will be output as a result of the program execution (figure 6.7).

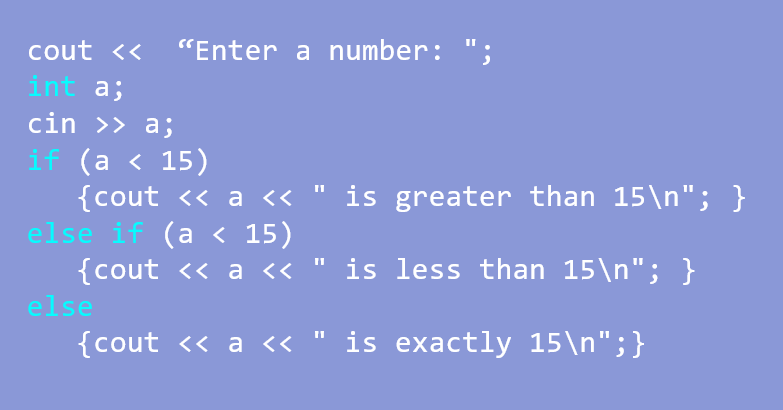


Figure 6.7 – Program code fragment of task №4