conditional expression: An expression consisting of two operators and three operands.

dangling *else*: A *if-else* statement in which there are more *if* statements than *else* statements.

De Morgan's law: A rule used to complement a logical expression.

default case: The case label that matches any unmatched value.

equality expression: An expression that determines if two entities are equal or not equal.

if statement: A statement that executes a block of code based on the value of an expression. If the statement is true, the block is executed; otherwise, the block is ignored.

if-else statement: A statement that executes one of two blocks of code depending on the truth or falsity of an expression.

logical-and expression: An expression that is *true* only if all of its conditions are *true*.

logical expression: An expression that uses one of the following three logical operator (not, and, or).

logical-or expression: An expression that is *true* if any of its conditions are *true*

multiway selection: A selection statement that is capable of evaluating more than two alternatives. In C++, the switch statement. Contrast with *two-way selection*.

one-wat selection: A selection *if* statement with only a true clause.

relational expression: An expression that uses one of the four relational operators: less than, less than or equal, greater than, and greater than or equal.

short-circuit behaviour: The design of the logical-and and logical-or expressions that allows the program to determine the result of the expressions without evaluating all of the possible alternatives.

switch statement: A conditional statement that evaluates an expression and executes labelled statements based on the value of the expression.

ternary expression: An expression that contains three operands (the conditional expression (?:) is the only ternary expression in C++).

two-way selection: A selection statement that is capable of evaluating only two alternatives.

To solve some problems, we need to make a decision based on the test of a true-false condition. This is referred to as **selection**.

The most common structure for making a decision is the **one way selection**, accomplished in C++ by the *if* statement.

The second statement we discussed for selection is the **two-way selection** using the *if-else* statement.

Sometimes decision-making problems are too complex to be solved by using the relational and equality expressions. We can combine relational and equality expressions with **logical expressions** to achieve a selection. We discussed three logical operators: NOT, AND, and OR.

Another **multiway decision** construct in C++ is the *switch* statement in which the decision is based on specific values.

Another construct that can be used for decision making is called the **conditional expression**. It is the only **ternary expression** in C++. It uses two operators and three operands.

RQ-1. Which expression has a higher precedence: *relational* or *equality*?

Relational

RQ-2. Which expression has a higher precedence: *logical-and* or *logical-or*?

logical-and

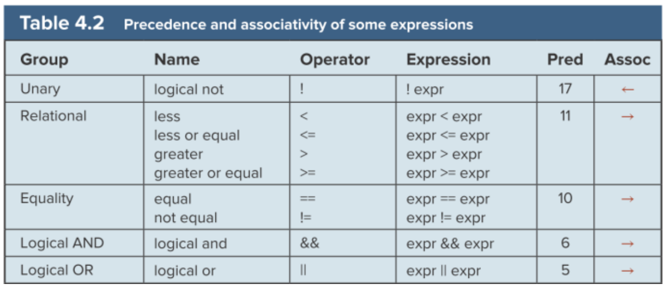


Table 4.2 Precedence and associativity of some expressions

Group	Name	Operator	Expression	Pred	Assoc
Unary	logical not	!	! expr	17	←
Relational	less less or equal greater greater or equal	< <= > >=	expr < expr expr <= expr expr > expr expr >= expr	11	→
Equality	equal not equal	== !=	expr == expr expr != expr	10	→
Logical AND	logical and	&&	expr && expr	6	→
Logical OR	logical or	||	expr || expr	5	→

RQ-3. List three selection statements discussed in this chapter.

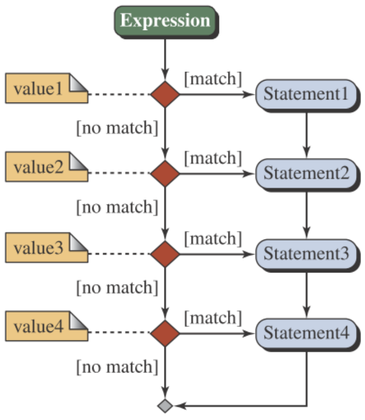
1. *if* statement,
2. *if-else* statement (+conditional expression),
3. *switch* statement,

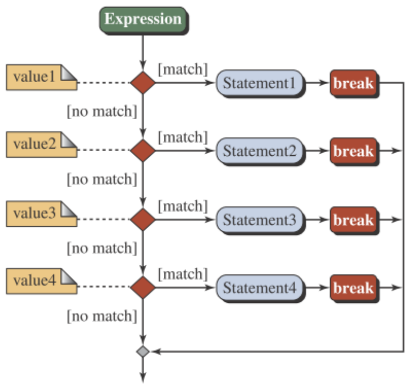
RQ-4. What is the dangling-else problem and how we can avoid it?

It is nested *if-else* statement problem. Occurs when we have more *if* branches than *else* branches. Use compound statements to avoid this problem.

RQ-5. Do we need *break* statements in a fall-through switch statement? Explain.

No, we do not need *break* statement because it terminates the statement (it is used to execute only one case). The fall through switch statement, execute all the rest cases when it find the first matched case.





RQ-6. Explain why the default case should be the last case in the switch statement. What happens if it is the first? What happens if it is the in middle?

The default case it used to execute the code when none of cases matches, so it has to be placed at the end of the switch statement otherwise the rest cases after will be ignored.

RQ-7. Describe the purpose of the conditional expression.

```
condition ? expression1 : expression2
```

Can be used anywhere an expression can be used. Also it is used to achieve tow-way decision and is written in a compact form.

```
x = condition ? statement1 : stateemnt2;          // As part of another statement
condition ?  statement1 : statement2;             // As an expression statement
```

(c) Using a condtional expression