Learning Objectives

At the end of this chapter, student should be able to:

- ✓ Use the relational operator (>, >=, <, <=, ==, !=)
- ✓ Use the logical operator (!, &&, ||)
- ✓ Understand the Boolean expression
- ✓ Write the pseudocode for the selection structure
- ✓ Create flowchart for the selection structure
- ✓ Implement decision using one-way, two-ways, and multiple-ways selection structure (if, if-else)
- ✓ Recognize the correct ordering of decisions in multiple branches
- ✓ Program simple and complex decision

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

A. Comparison and Expression

- To make decisions, you must be able to make comparisons and express the conditions that are **true** or **false**.
- Comparison is done by using relational operator.

  i.e. `i > j, x == y`, etc.
- An expression that has a value either **true** or **false** is called a logical (Boolean) expression.
- Condition is represented by a logical (Boolean) expression.

B. Relational and Logical Operator

1. Relational Operator

- There are 6 relational operators that allow to state conditions and make comparisons.
- They are all binary operations that accept 2 operands and compare them.
- The result is logical data, that is, it is always **true** (1) or **false** (0).

Relational Operator	Meaning
<	Less than
<=	Less or equal
>	Greater
>=	Greater or equal
==	Equal
!=	No equal

2. Logical Operator

- Logical operators enable you to combine logical expressions.
- C++ has three logical operators as below:

Operator	Description
!	not
&&	and
||	or

- Logical operators take only logical values as operands and produce only logical values as result.
- The operator `!` is unary, so it has only one operand.
- The operator `&&` and `||` are binary operators, so it has more than one operand.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

C. Simple Boolean Expression

- An expression in which two numbers (arithmetic values) are compared using a single relational operator.
- Each Boolean expression has the Boolean value true or false according to the arithmetic validity of the expression:
- Syntax of Boolean expression:

```
<arithmetic value> relational operator <arithmetic value> → Boolean value
```

- Example of Boolean expression using relational operator:

Expression	Meaning	Value
`8 < 15`	8 is less than 15	True
`6 != 6`	6 is not equal to 6	False
`2.5 > 5.8`	2.5 is greater than 5.8	False
`5.9 <= 7.5`	5.9 is less than or equal to 7.5	True

1. The ! (not) Operator

- Expression:

Expression	!(Expression)
true (1)	false (0)
false (0)	true (1)

- Example:

Expression	Value	Explanation
`! ('A' > 'B')`	true	Because 'A' > 'B' is **false**, ! ('A' > 'B') is **true**
`! (6 <= 7)`	false	Because 6 <= 7 is **true**, ! (6 <= 7) is **false**

2. The && (and) operator

- Expression:

Expression1	Expression2	Expression1 && Expression2
true (1)	true (1)	true (1)
true (1)	false (0)	false (0)
false (0)	true (1)	false (0)
false (0)	false (0)	false (0)

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

- Example:

Expression	Value	Explanation
`(14 > = 5) && (‘A’ < ‘B’)`	true	Because (14 > = 5) is **true**, (‘A’ < ‘B’) is **true**, and **true** && **true** is **true**, the expression evaluates to **true**.
`(24 >= 35) && (‘A’ < ‘B’)`	false	Because (24 >= 35) is **false**, (‘A’ < ‘B’) is **true**, and **false** && **true** is **false**, the expression evaluates to **false.**

3. The || (or) operator

- Expression:

Expression1	Expression2	Expression1 || Expression2
true (1)	true (1)	true (1)
true (1)	false (0)	true (1)
false (0)	true (1)	true (1)
false (0)	false (0)	false (0)

- Example:

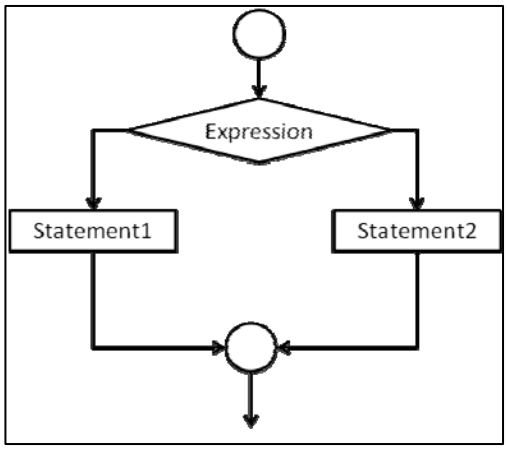
Expression	Value	Explanation
`(14 > = 5) || (‘A’ > ‘B’)`	true	Because (14 > = 5) is **true**, (‘A’ > ‘B’) is **false**, and **true** || **false** is **true**, the expression evaluates to **true**.
`(24 >= 35) || (‘A’ >‘B’)`	false	Because (24 >= 35) is **false**, (‘A’ > ‘B’) is **false**, and **false** && **false** is **false**, the expression evaluates to **false**.
`(‘A’ <=‘a’) || (7 !=7)`	true	Because (‘A’ <=‘a’) is **true**, (7 !=7) **false**, and **true** || **false** is **true**, the expression is evaluates to **true**.

D. **Precedence of Operator**

Operators	Precedence
++, --	First
!, +, - (unary operators)	Second
*, /, %	Third
+, - (binary operator)	Fourth
<, <=, >=, >	Fifth
==, !=	Sixth
&&	Seventh
||	Eighth
= (assignment operator)	Last

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

E. Decision Control Structure



- The program executes particular statements depending on some condition(s).
- This structure represents decision-making capabilities of the computer.
- You can use selection control structure in pseudocode/flowchart to illustrate a choice between two or more actions, depending on whether a condition is true or false.
- There a number of variations of the selection structures, as follows:

1. **One-way selection**
    a. Simple selection with null false branch (null ELSE statement)
2. **Two-ways selection**
    a. Simple selection (if-else statement)
    b. Combined selection
        i. && (AND)
        ii. || OR
3. **Multi-ways (multiple) selection**
    a. Linear nested if statement
    b. Non-linear nested if statement
    c. Switch statement

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

1. One Way Selection

- The null ELSE structure is a variation of the simple IF structure. It used when a task is performed only when a particular condition is true.
- If the condition is false, then no processing will take place and the IF statement will be passed.
- Syntax:

```
if (expression)
        Statement
```

- Example:

```
#include <iostream.h>
#include <conio.h>
#include <math.h>

void main()
{       int absoluteValue, negativeNumber;

        cout << "Enter negative number : ";
        cin >> negativeNumber;

        if (negativeNumber < 0)
                absoluteValue = fabs(negativeNumber);

        cout << "Absolute value for " << negativeNumber << " is "
             << absoluteValue;

getch();
}
```

2. Two Way Selection

a. Simple selection (if-else)

- Simple selection occurs when a choice is made between two alternative paths, depending on the result of a condition being true or false.
- Only one of the selection will be followed, depending on the result of the condition in the IF clause.
- Syntax:

```
if (expression)
        statement1
else
        statement2
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

- Example:

```
#include <iostream.h>
#include <conio.h>

void main()
{       double  wages, rate, hours;

        cout  << "Enter working hours and rate ";
        cin >> hours >> rate;

        if (hours > 40.0)
                wages = 40.0 * rate + 1.5 * rate * (hours – 40.0);
        else
                wages = hours * rate;

        cout <<  "The wages are RM " << wages;

getch();
}
```

b. Combined selection

- A combined IF statement is one which contains multiple conditions, each connected with the logical operators AND and OR.
- If the condition are combined using the connector AND, then both conditions must be true for the combined condition to be true.
- If the connector OR is used to combine any two conditions, then only one of the conditions needs to be true for the combined condition to be considered true.
- If neither condition is true, then the combined condition is considered false.

i. Example of program using && (AND) selection

```
#include <iostream.h>
#include <conio.h>

int number = 2;
void main ()
{       if (number > 0 && number % 2 == 0)
                cout << "Number is positive and even";
        else
                cout <<"Number is  negative and odd"     ;
getch();
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

ii. Example of program using || (OR)

```
#include <iostream.h>
#include <conio.h>

int number = 2;
void main ()
{       if (number > 0 || number % 2 == 0)
                cout << "Number is positive or even";
        else
                cout << "Number is negative and odd";
getch();
}
```

3. Multiway Selection

a. Linear nested if statement

- Linear IF statement is used when a field is being tested various values and a different action is to be taken for each value.
- This form of nested IF is called linear because each ELSE immediately follows the IF condition to which it corresponds,
- Comparisons are made until next ELSE statement is reached.
- There are an equal number of IF and ELSE statements.
- Syntax:

```
if (condition1)
        Statement1
else if (condition2)
        Statement2
else
        Statement3
```

- Example program:

```
#include <iostream.h>
#include <conio.h>

void main()
{       double balance, interestRate;

        cout << "\n What is your balance? ";
        cout << "\n Balance RM ";
        cin >> balance;

        if (balance > 5000.00)
                dividendRate = 0.07;
        else if (balance >= 2500.00)
                 dividendRate = 0.05;
        else if (balance >= 1000.00)
                 dividendRate = 0.03;
        else
                 dividendRate = 0.00;

        cout << "\n Your interest rate is RM " << dividendRate ;

getch();
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

b. Non-linear nested if statement

- A non-linear nested IF occurs when a number of different conditions need to be satisfied before a particular action can occur,
- It called non-linear because the ELSE statement may be separated from the IF statement with which it is paired.
- Each ELSE statement should be aligned with the IF condition to which it corresponds.
- Syntax:

```
if (condition1)
        if (condition2)
                Statement1
        else
                Statement2
```

- Example of program:

```
#include <iostream.h>
#include <conio.h>

void main()
{       char gender;
        int age;
        double policyRate;

        cout << "\n What is your gender and age? ";
        cout << "\n gender (M|F) ";
        cin >> gender;
        cout << "\n age ";
        cin >> age;

        if (gender == 'M' || gender == 'm')
                if (age < 21)
                        policyRate = 0.05;
                else
                        policyRate = 0.035;
        else  if (gender == 'F' || gender == 'f')
                if (age < 21)
                        policyRate = 0.04;
                else
                        policyRate = 0.03;
        cout << "\n Your policy rate is RM " << policyRate;

getch();
}
```

4. Compound (Block of) Statement for Selection Structure

a. Why compound statement

- The if and if…else structures control only one statement at a time.
- Suppose you want to execute more than one statement if the expression in an **if** or **if…else** statement evaluates to **true**, C++ provides a structure called compound statement or a block of statements.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

- A compound statement consist of sequence of statements enclosed in curly braces, { and }.
- In an in **if** or **if…else** structure, a compound statement functions as if it was a single statement.
- A compound statements takes the following syntax form:

```
if (condition1)
{       statement1
        statement2
        …
        statementn
}
```

b. Example of program for compound statement

```
#include <iostream.h>
#include <conio.h>

void main()
{       char gender;
        int age;
        double policyRate;

        cout << "\n What is your gender and age? ";
        cout << "\n gender (M|F) ";
        cin >> gender;
        cout << "\n age ";
        cin >> age;

        if (gender == 'M' || gender == 'm')
        {
                if (age < 21)
                {       policyRate = 0.05;
                        cout << "\n Your policy rate is RM " << policyRate;
                }
                else
                {       policyRate = 0.035;
                        cout << "\n Your policy rate is RM " << policyRate;
                }
        }
        else  if (gender == 'F' || gender == 'f')
         {
                if (age < 21)
                {       policyRate = 0.04;
                        cout << "\n Your policy rate is RM " << policyRate;
                }
                else
                {       policyRate = 0.03;
                        cout << "\n Your policy rate is RM " << policyRate;
        }
        else
        {       cout << "Unidentified code";
        }

getch();
}
```

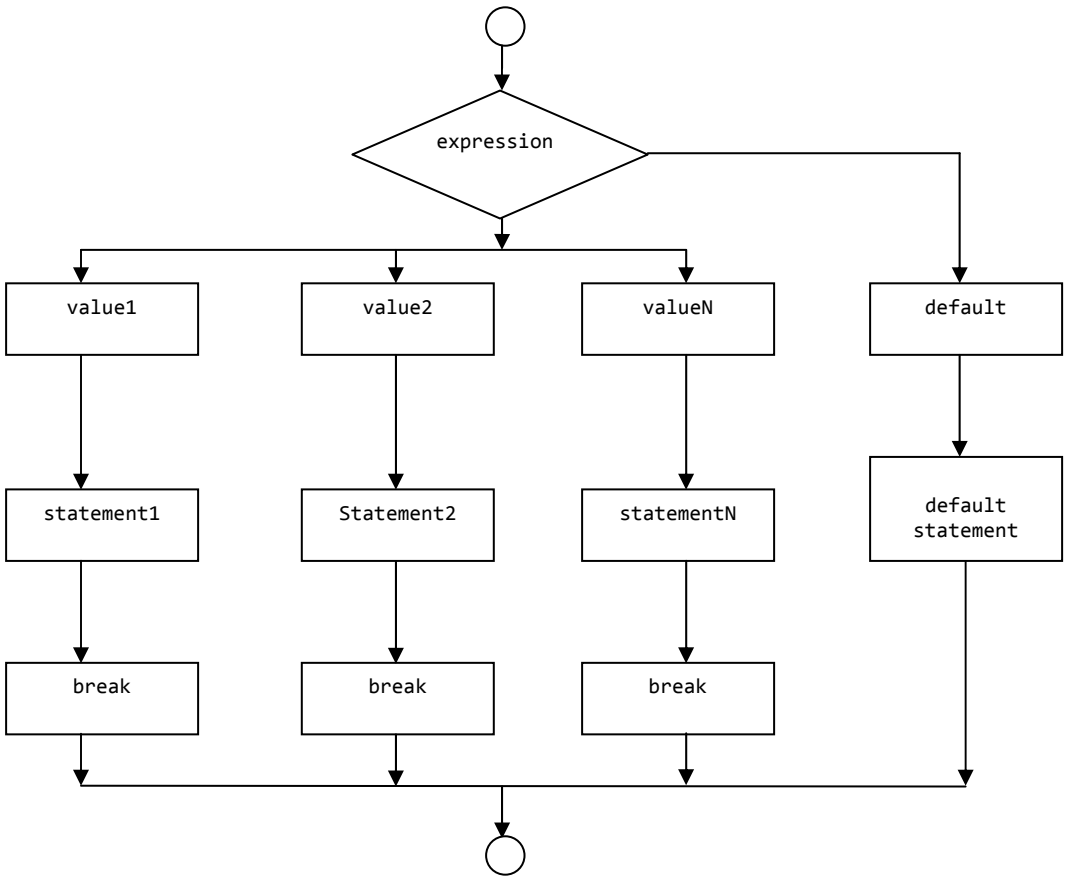

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Multiway Selections (Switch statement)

- `switch` statement is an alternative method for multiple selections.
- It is often used to replace the nested `if` statement to avoid confusion of deeply nested `if`s.
- `switch` statement only evaluates an integer expression or a character constant value.
- Syntax form:

```
switch (expression)
{
       case value1: statement1;
              break;
       case value2: statement2;
              break;
       …
       case valueN: statementN;
              break;
       default: default statement;  //optional
}
```

- Flowchart



References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Comparison of `if` and `switch` statement

1. Using integer expression

if-else statement

```
#include <iostream.h>
#include <conio.h>

void main()
{   int code;
    cout << "Enter code (1, 2, 3 or 4):"
    cin >> code;

    if (code == 1)
        cout << "Diploma in Computer Science";
    else if (code == 2)
        cout << "Diploma in Accountancy";
    else if (code == 3)
        cout << "Diploma in Business Study";
    else if (code == 4)
        cout << "Diploma in Banking";
    else
        cout << "Error in filling code";
    cout << "\n End of Program.";

getch();
}
```

switch statement

```
#include <iostream.h>
#include <conio.h>

void main()
{   int code;
    cout << "Enter code (1, 2, 3 or 4):"
    cin >> code;

    switch (code)
    {
        case 1 : cout << "Diploma in Computer Science";
                 break;
        case 2 : cout << "Diploma in Accountancy";
                 break;
        case 3 : cout << "Diploma in Business Study";
                 break;
        case 4 : cout << "Diploma in Banking";
                 break;
        default : cout << "Error in filling code";
    }
    cout << "\n End of Program.";

getch();
}
```

2. Using character constant

if-else statement

```
#include <iostream.h>
#include <conio.h>

void main()
{   char code;
    cout << "Enter code (G, Y, or R):"
    cin >> code;

    if (code == 'G'|| code == 'g')
        cout << "GREEN";
    else if (code == 'Y' || code == 'y')
        cout << "YELLOW";
    else if (code == 'R' || code == 'r' )
        cout << "RED";
    else
        cout << "Error in filling code";
    cout << "\n End of Program.";

getch();
}
```

switch statement

```
#include <iostream.h>
#include <conio.h>

void main()
{   char code;
    cout << "Enter code (G, Y, or R):"
    cin >> code;

    switch (code)
    {
        case 'G' :
        case 'g' : cout << "GREEN";
                   break;
        case 'Y' :
        case 'y' : cout << "YELLOW";
                   break;
        case 'R' :
        case 'r' : cout << "RED";
                   break;
        default : cout << "Error in filling code";
    }
    cout << "\n End of Program.";

getch();
}
```

String Comparison

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

- In C++, c-strings are compared character-by-character using the system's collating sequence.
- Predefined function that can be used in C++ to compare c-string is strcmp().

Format	Effect
`strcmp(s1, s2)`	• Returns a value < 0 (-) if `s1` less than `s2` • Returns 0 if `s1` and `s2` are the same • Returns a value > 0 (+) if `s1` is greater than `s2`

- Assume that you use the ASCII character set.
    i. The c-string "**Air**" is less than the c-string "boat" because the first charter of "**Air**" is less than the first character of "**Boat**".
    ii. The c-string "**Air**" is less than c-string "**An**" because he first character of both string are the same, but the second character '**i**' of "**Air**" is less than the second character '**n**' of "**An**".
    iii. The c-string "**Bill**" is less than the c-string "**Billy**" because the first four characters of "**Bill**" and "**Billy**" are the same, but the fifth character of "**Bill**", which is '\0' (the null character), is less than the fifth character of "**Billy**", which is '**y**'.
    iv. The c-string "**Hello**" is less than "**hello**" because the first character '**H**' of the c-string "**Hello**" is less than the first character '**h**' of the c-string "**hello**".
- Example

```
#include <iostream.h>
#include <string.h>

int main()
{
    char str1[20], str2[20];

    cout << "\nEnter first string ";
    cin >> str1;
    cout << "\nEnter second string ";
    cin >> str2;

    if (strcmp(str1, str2) == 0)
        cout << "Both string are identical";
    else if (strcmp(str1, str2) > 0)
        cout << "First string is greater than the second string";
    else if (strcmp(str1, str2) < 0)
        cout << "First string is less than the second string";

return 0;
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam