



CSC 128

TOPIC 3: SELECTION CONTROL STRUCTURE

COURSE OUTLINE

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

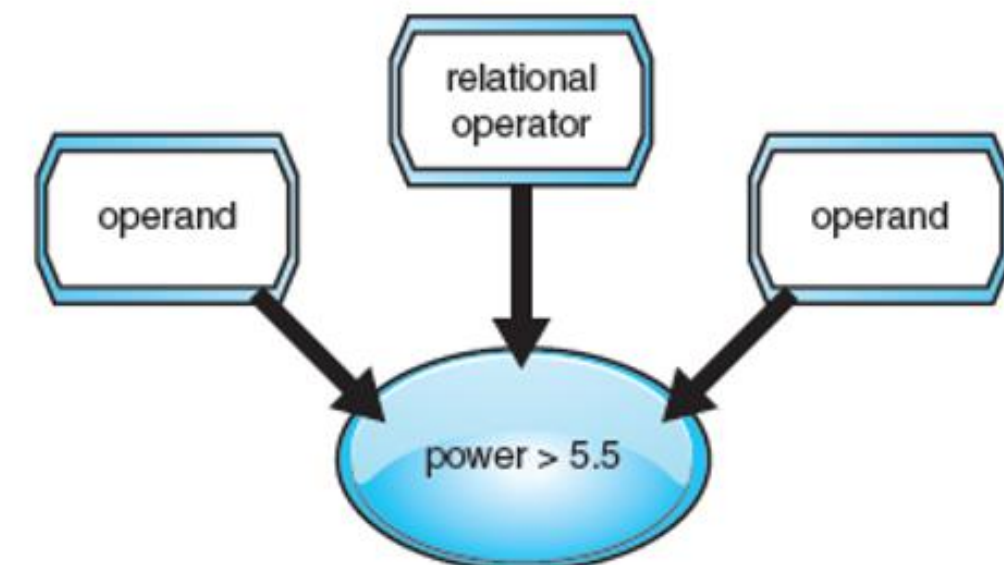
- Interpret the concept of relational and logical operators.
- Differentiate between three types of selection.
- Produce programs using selection control.
- Introduce the nested if problem, explain the output based on the various inputs from the user.
- Solve a problem using multiway selection.

BOOLEAN EXPRESSION

- A **Boolean expression** is an expression that produces the result **TRUE** (non-zero value) or **FALSE** (0).
- The computer will make decisions based on the Boolean expression.
- An example of a Boolean expression is **$a < b$** .
- The condition of a is tested against the condition of b. If a is in fact less than b, the test result is true.
- However, if the value of a is greater than the value of b, the test is false.
- The relational operator and/or logical operator is used to test the condition within a program.
- The value zero (0) is considered to be false by C++. Any positive or negative value is considered to be true. C++ evaluates any non-zero value to be true.

RELATIONAL OPERATORS

- In order to evaluate a comparison between two expressions, we can use the **Relational operators**.
- As specified by the ANSI-C++ standard, the result of a relational operation is a **Boolean** value that can only be **true** or **false**, according to the result of the comparison.
- Relational operators allow two quantities to be compared. Normally, when the expressions have two operands or identifiers, either similar or dissimilar, smaller than or bigger than.



RELATIONAL OPERATORS

- The six common relational operators available in C++ are listed in the table below.

Relational Operator	Meaning	Example
<code>==</code>	Equal	<code>respond == 'y'</code>
<code>!=</code>	Not Equal to	<code>result != 6.5</code>
<code><</code>	Less than	<code>count < 10</code>
<code><=</code>	Less than or equal to	<code>Count <= 10</code>
<code>></code>	Greater than	<code>average > 155.5</code>
<code>>=</code>	Greater than or equal to	<code>average >= 155.5</code>

RELATIONAL OPERATORS

- The **relational operator** can be used to **compare** between integer, character, double, float or string data types.
- If mixed data types are compared, for example integer data type compared with double data type, the computer will convert the number into double first, then it will compare the numbers and produce the value either true or false.

EXAMPLE 3.1

Boolean expression	Result
$(3 == 3)$	would return true
$(7 > 10)$	would return false
$(4.5 != 5.4)$	would return true
$(180 >= 180)$	would return true
$(6 < 6)$	would return false

RELATIONAL OPERATORS

```
#include <iostream>
using namespace std;
int main( )
```

```
{
```

```
    cout<<(20-8==6)<<endl; //12==6  FALSE 0
```

```
    cout<<(40 > 50)<<endl; //40>50  FALSE 0
```

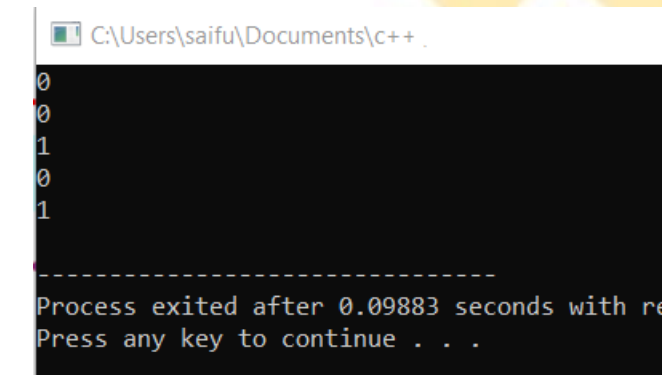
```
    cout<<(15.5 >= 15.5)<<endl; // TRUE 1
```

```
    cout<<(12 + 8 < 20)<<endl; //20<20  FALSE 0
```

```
    cout<<(4*10%2-5 <= 25/5+3)<<endl; //-5<=8  1
```

```
    return 0;
```

```
}
```



```
C:\Users\saifu\Documents\c++
0
0
1
0
1
-----
Process exited after 0.09883 seconds with re
Press any key to continue . . .
```

RELATIONAL OPERATORS

- We can also compare the numbers using variables. Example 3.3 shows examples of using variables of integer and double data types.

Example 3.3 : Suppose that int **x=7**, **y=6**, **z=3** ; double **i = 3.25**, **j= 5.6**;

Boolean Expression	Results
$(x == 7)$	would return true.
$(y * z \geq x)$	would return true since $(6 * 3 \geq 7)$
$(x + 3 > y * z)$	would return false since $(7 + 3 > 6 * 3)$
$((y = 7) == x)$	would return true. $(7 == 7)$
$i \leq j$	would return true $(3.25 \leq 5.6)$
$i == j$	would return false $(3.25 == 5.6)$

RELATIONAL OPERATORS

- Another example is to compare variables using the char data type.
- In ASCII code, letters are stored in alphabetical order, so the value of character 'A' is lower than value of character 'B'.
- If uppercase and lowercase letters are compared, the computer will check the sequence of uppercase then lowercase letters.

EXAMPLE 3.4

Boolean expression	Result
'a' < 'b'	would return true
'a' > 'b'	would return false
'A' < 'B'	would return true
'A' > 'B'	would return false
'A' < 'b'	would return true
'A' > 'b'	would return false

Note: Under the ASCII Collating Sequence : numbers < upper case letters < lower case letters. Please check ASCII chart table in Chapter 1.

RELATIONAL OPERATORS

- The comparison of string data type follows the same rule as the char data type.
- In the string data type, comparison is done character by character, beginning with the first character. The comparison continues until either a mismatch is found, or all the characters are found to be equal.
- If two strings of different lengths are compared, and the result of the comparison is equal to the last character of the shorter string, the shorter string is evaluated as less than the longer string.

RELATIONAL OPERATORS

EXAMPLE 3.5

```
string fruit1 = "Durian";  
string fruit2 = "Duku";  
  
cout<<(fruit1 < fruit2) <<endl;  
cout<<(fruit1 > fruit2) <<endl;  
cout<<(fruit1 < "Rambutan") <<endl;
```

OUTPUT

```
0 // false : third character 'r' greater than 'k'  
1 // true : third character 'k' less than 'r'  
1 // true : first character 'D' less than 'R'
```


LOGICAL OPERATORS

- In order to evaluate a comparison between two logical expressions, we can use the **Logical Operators**.
- Table 3.2 lists the logical operators and their descriptions, while Table 3.3 shows possible conditions for operand a and b.

Symbol	Meaning	Description
&&	AND	Both conditions must be TRUE
	OR	Either one of the conditions must be TRUE
!	NOT	Reverse condition

Table 3.2

Logical operators and their descriptions

First operand a	Second operand b	Result a && b	Result a b
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Table 3.3

Possible conditions for operand a and b

THE HIERARCHY OF RELATIONAL AND LOGICAL OPERATORS

- The table below shows the hierarchy of relational and logical operators, from the highest to the lowest.

Operator	Precedence
!	first
< <= > >=	second
== !=	third
&&	fourth
	fifth

COMPOUND BOOLEAN EXPRESSION

- The combination of relational and logical operator will be evaluated based on the hierarchy of execution similar to the arithmetic operators.

EXAMPLE 3.6

```
#include<iostream>
using namespace std;
int main()
{
    int y=2, x=-4 , z = 300 , w=2;

    cout<< ((y > x) && ! (y >= z)) <<endl;
    cout<< ((z > 100) || (z < w)) <<endl;
    cout<< ((y * 100 - x + 100 ) < z) <<endl;
    cout<< ((y == w && z > x) || ! (x < 10)) <<endl;

    return 0;
}
```

OUTPUT

```
1
1
0
1
```


ONE-WAY SELECTION

- The *if* statement tests for a particular condition (expressed as a Boolean expression) and only executes the following statement(s) if the condition is true.

Syntax

```
if (boolean-expression)
    statement is executed when condition is true;
```

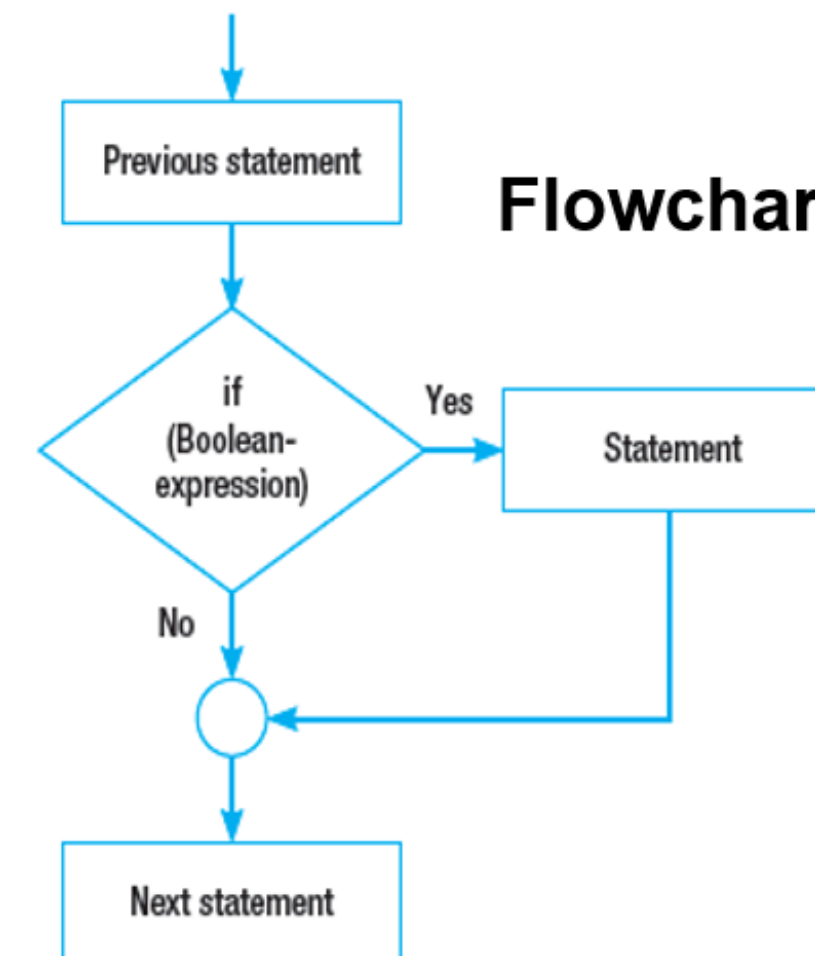
EXAMPLE 3.7

```
int x= 5, b = 10
if (x < b)
    cout<<"The integer number is less than 10";
```

EXAMPLE 3.8

```
double gpa;
cin>>gpa;
if (gpa > 2.5)
    cout<<"Congratulations, you PASSED!";
```

Flowchart:



Note: We can use the value Yes/No or True/False

ONE-WAY SELECTION

- If a sequence of statements are to be executed, this can be done by making a **compound statement** or **block** by enclosing the group of statements in braces.

```
if (boolean-expression)
{
    Statement 1;
    Statement 2;
    :
    Statement n;
}
```

EXAMPLE 3.9

```
int x = 5, b = 10
if (x < b)
{
    cout<<"The integer number is less than 10"<<endl;
    cout<<"Thank you"<<endl;
}
```

EXAMPLE 3.10

```
double gpa;
cin>>gpa;
if (gpa > 2.5)
{
    cout<<"Congratulations, you PASSED!"<<endl;
    cout<<"You have graduated ";
    cout<<"and received the certificated."<<endl;
}
```

TWO-WAY SELECTION

- A two-way selection will check a Boolean expression, and if the result of the Boolean expression is true, it will display the statement for the true condition.
- Otherwise, it will display the statement for the false condition. Compare two selections which are written in a statement of the form:

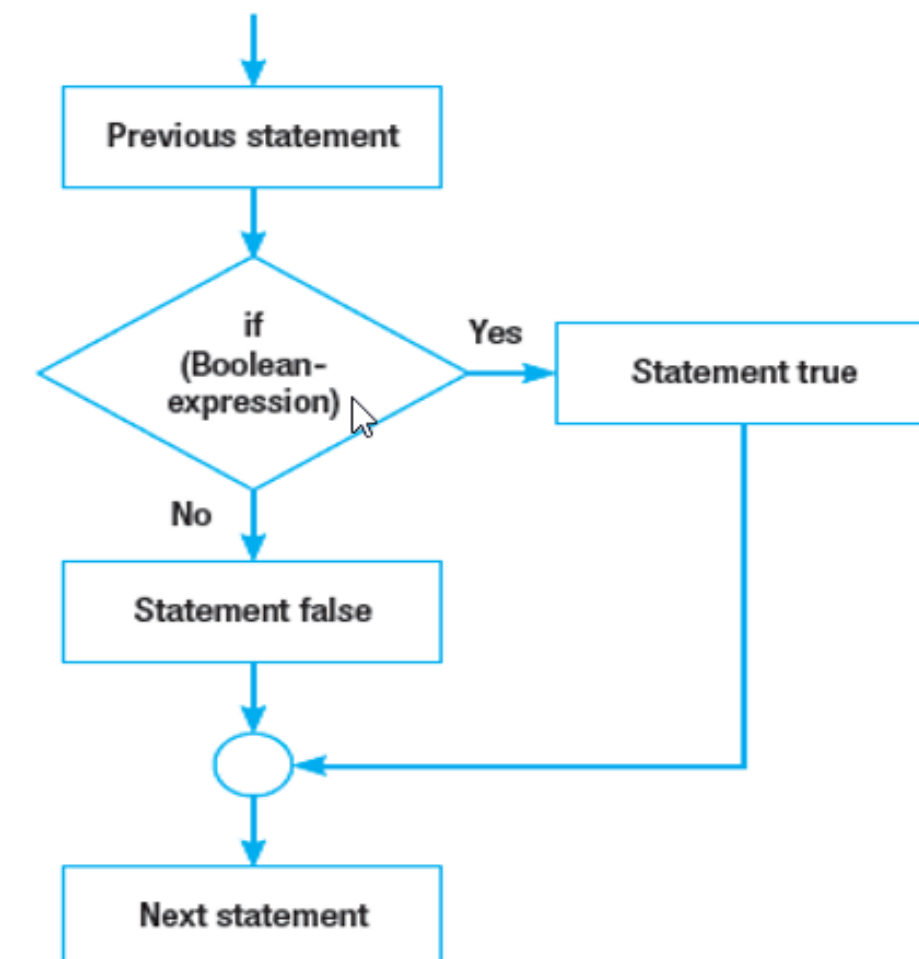
```

if (boolean-expression)
    statement for true condition;
else
    statement for false condition;
  
```

EXAMPLE 3.11

```

int x = 5, b = 10;
if (x < b)
    cout<<"The integer number is less than 10"<<endl;
else
    cout<<"The integer number is greater than 10"<<endl;
  
```



TWO-WAY SELECTION

- If a sequence of statements is to be executed, this is done by making a compound statement by using braces to enclose the sequence:

```
if (boolean-expression)
{
    statements;
}
else
{
    statements;
}
```

EXAMPLE 3.13

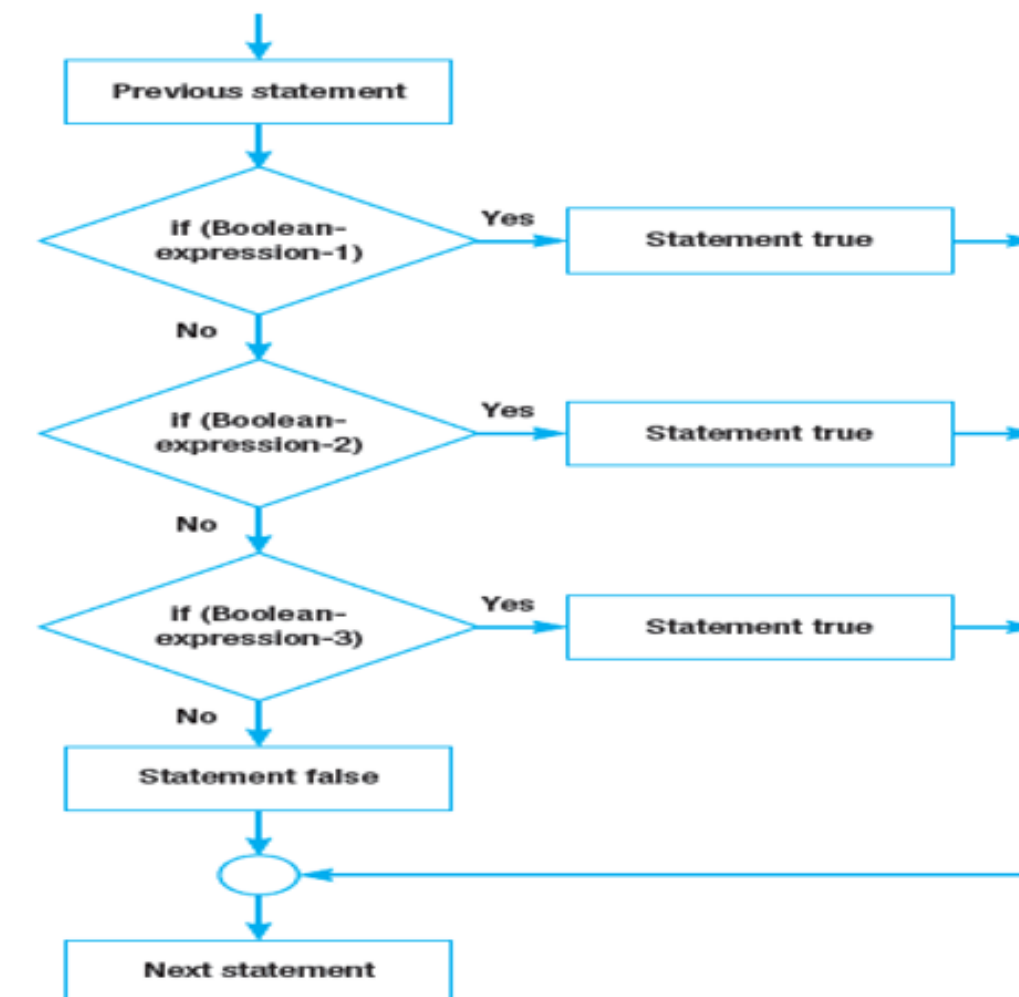
```
double gpa;
cin>>gpa;
if (gpa > 2.5)
{
    cout<<"Congratulations, you PASSED!"<<endl;
    cout<<"You have graduated ";
    cout<<"and will receive the certificate."<<endl;
}
else
{
    cout<<"Sorry, you FAILED."<<endl;
    cout<<"You cannot graduate.";
}
```


MULTIPLE SELECTION

- Multiple selections are selections that **compare more than two selections**. This type of selection will be executed for true conditions.
- The program will check the Boolean expressions in sequential order.
- The syntax for multiple selection is as follows:

```

if (boolean-expression-1)
    statement-1;
else if (boolean-expression-2)
    statement-2;
.
.
.
else
    statement when none of the conditions is true;
    
```



MULTIPLE SELECTION

EXAMPLE 3.16

```
/* The purpose of this program is to identify the
   classes of Bachelor's Degree (Honours) */
#include <iostream>
using namespace std;
int main ()
{
    double cgpa;
    cout<<"Please enter CGPA : ";
    cin>>cgpa;
    if (cgpa >= 3.50 && cgpa <= 4.00)
        cout<<"You get a first class with honours."<<endl;
    else if (cgpa >=3.00 && cgpa <= 3.49)
        cout<<"You get a upper second class with honours."<<endl;
    else if (cgpa >=2.20 && cgpa <= 2.99)
        cout<<"You get a lower second class with honours."<<endl;
    else if (cgpa >=2.00 && cgpa <= 2.19)
        cout<<"You get a third class with honours."<<endl;
    else
        cout<<"Sorry, you are not able to graduate."<<endl;

    cout<<"***** Thank You *****"<<endl;

    return 0;
}
```


DECISION CONTROL STRUCTURE

- Until now, we have learnt how to write Boolean expressions using relational and logical operators.
- Additionally, to write the instructions to enable the computer to make a decision, we need to use the **if statement** or **switch statement** method to accomplish the task.
- The if statement is a method that is commonly used to write conditional statements in programming language. There are four ways to write conditions using if statements—one-way, two-way, multiple and nested statement.
- The switch statement is one of the **alternatives** to the if statement. Normally, we use the switch statement to replace the if statement if the condition does **not involve relational or logical operators**.

IF STATEMENT

EXAMPLE 3.17

```
/* Ask user to enter three integer numbers and find the highest value among
the three numbers */
#include<iostream>
using namespace std;
int main()
{
    int num1, num2, num3;

    cout<<"Enter three integers:";
    cin>>num1>>num2>>num3;

    if(num1>=num2 && num1>=num3)
        cout<<"Highest value is "<< num1<<endl;
    else if(num2>=num1 && num2>=num3)
        cout<<" Highest value is "<< num2<<endl;
    else if(num3>= num1 && num3>= num2)
        cout<<" Highest value is "<< num3<<endl;

    return 0;
}
```

IF VS SWITCH STATEMENT

If statement	Switch statement
Valid for int , double , float , char and string	Valid for int and char only
Can use for range For example if (marks>=50 && marks<=100)	Cannot used for range
Using else for last option	default for last option

SWITCH STATEMENT

- The switch statement is valid for **integer numbers and single characters** only.
- The syntax for the switch statement is as follows.

```
switch (selector variable)
{
    case value for case 1:
        Statement 1;
        .....;
        break; //to stop at case 1
    case value for case 2:
        Statement 2;
        .....;
        break; //to stop at case 2
    .
    .
    .
    case value for case n:
        Statement n;
        .....;
        break; //to stop at case n
    default:
        statement;
        break; //to stop at default statement
}
```


SWITCH STATEMENT

- The value of the variable given into the switch is compared to the value following each of the cases, and when one value matches the value of the variable, the computer continues executing the program from that point.
- The **break** is used to **break out of the case statements**. If the value for case 1 is not a match, the program will check the next case value until it finds a matching value.
- The n value means we can have as many cases as we want in a program.
- The **default keywords are optional**—it means the value does not equal the values following any of the cases—it is similar to the else in the if statement.

SWITCH STATEMENT

- For more than one option in every case (example: capital and small letters), the following example shows how the code can be written.

EXAMPLE 3.19

```
#include<iostream>
using namespace std;
int main(){
    char code_size;
    cout<<"Enter your size:";
    cin>> code_size;
    switch (code_size) {
        case 's':
        case 'S':
            cout<<"Small Size"<<endl;
            break;
        case 'm':
        case 'M':
            cout<<"Medium Size"<<endl;
            break;
        case 'l':
        case 'L':
            cout<<"Large Size"<<endl;
            break;
        default :
            cout<<"Invalid Code"<<endl;
            break;
    }
    cout<<"Hope you have identified your size";
    return 0;
}
```


SWITCH STATEMENT

- The break statement causes the program to proceed to the first statement after the switch structure.
- Note that the switch control structure is different to the others in that braces are not required around multiple statements.
- What will happen if you do not put the break statement in every case?

EXAMPLE 3.22

```
#include<iostream>
using namespace std;
int main()
{
    char grade;
    cout<<"Enter your grade:";
    cin>>grade;

    switch (grade)
    {
        case 'A' :
            cout<<"Excellent";
        case 'B' :
            cout<<"Average";
        case 'C' :
            cout<<"Poor";
        case 'F' :
            cout<<"Try Again";
    }
    cout<<"\n Hope you are satisfied with your results!"
    return 0;
}
```

MULTI-WAY SELECTION (NESTED)

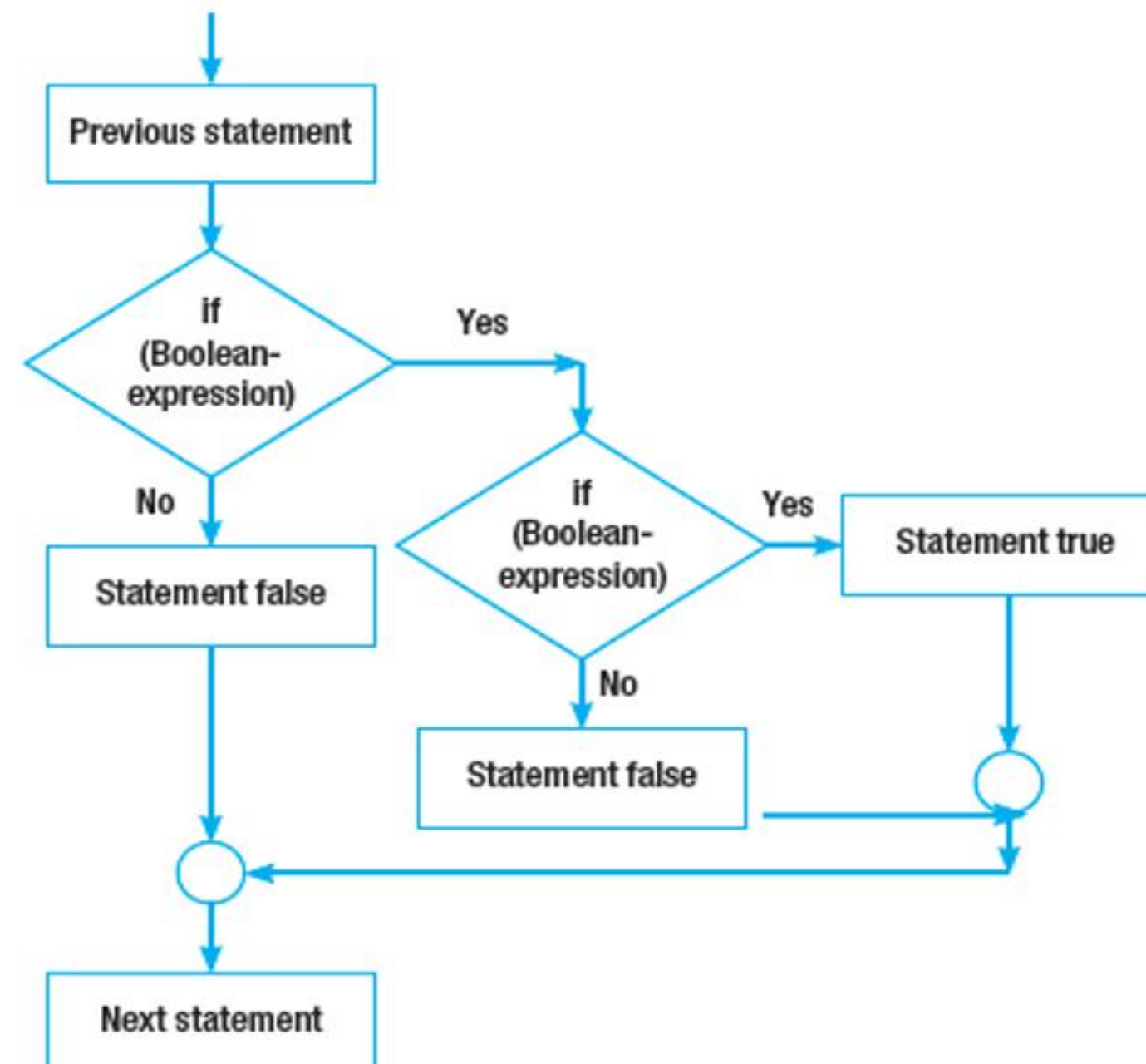
- Nested selections will occur when there are **two conditions** and we have to fulfil the first condition before we continue to the next condition.
- We can say that a nested statement is an if statement which is written inside another if statement.
- Usually, we must write in indentation style to make the multiple selection functionality much clearer.
- Nested selection can also be applied using the switch statement.

MULTI-WAY SELECTION (NESTED)

- The syntax for nested selection is as follows:

```

if (boolean-expression-1)
    if (boolean-expression-1)
        Statement-1;
    else
        Statement-2;
else
    Statement-3;
    
```



MULTI-WAY SELECTION (NESTED)

Gender	Waist	Status
Male	<37	Low Risk
	<40	Moderate Risk
	>40	High Risk
Female	<31.5	Low Risk
	<35	Moderate Risk
	>35	High Risk

MULTI-WAY SELECTION (NESTED)

EXAMPLE 3.23

```

/* The purpose of this program is to find the status of health risk by waist
measurement */

#include<iostream>
using namespace std;
int main()
{
    double waist;
    char gender;

    cout<<"Please enter your gender :";
    cin>>gender;

    cout<<"Please enter your waist measurement in inches :";
    cin>>waist;

    if (gender == 'M' || gender == 'm')
    {
        if(waist < 37)
            cout<<"Low Risk "<<endl;
        else if(waist < 40)
            cout<<"Moderate Risk "<<endl;
        else
            cout<<"High Risk "<<endl;
    }
    else if (gender == 'F' || gender == 'f')
    {
        if(waist < 31.5)
            cout<<"Low Risk "<<endl;
        else if(waist < 35)
            cout<<"Moderate Risk "<<endl;
        else
            cout<<"High Risk "<<endl;
    }
    else
        cout<<"INVALID CODE ENTERED"<<endl;
    return 0;
}

```

outside if - gender

inside if - waist

CONCLUSION

- The real intelligence of a computer comes from its ability to **make decisions**.
- The computer makes a decision based on Boolean expressions. The value of a Boolean expression is **1 or non-zero number (true)** or **0 (false)**.
- Relational operators and logical operators are used in Boolean expressions to make decisions.
- There are two methods used to give instruction to computers to make decisions - **if statement** and **switch statement**.
- There are three types of `if` statement—one way, two way and multiple way.
- Compound statement is a list of statements located between a pair of braces in an `if` statement.

CONCLUSION

- **Switch** statement is used when the value in the cases are integers and single characters only.
- **Break** statement is used for every case to stop cases when a match has been found in a switch statement.
- Nested selection is selection within selection. It is used when we have multiple conditions, when the first condition yields true, then it will check another condition to produce the result.

EXERCISE

EXERCISE 1

- Write an if statement which receives *monthSalary* from the user. The program will display the job based on the following table.

Salary	Job
1000-2000	Clerk
2001-4000	Executive
4001-8000	Manager

EXERCISE 1 - ANSWER

```
if (monthSalary >1000 && monthSalary<=2000)
cout << “\n Clerk”;
else if (monthSalary >2000 && monthSalary<=4000)
cout << “\n Executive”;
else if (monthSalary >4000 && monthSalary<=8000)
cout << “\n Manager”;
else
cout << “\n Out of range”;
```


EXERCISE 2

- What is the output of following program.

```
{  
  int number = 13;  
  
  if (number < 10)  
    cout<<number<<" is lesser than 10."<<endl;  
  else //  
    cout<<number<<" is greater than 10."<<endl;  
  cout<<"close."<<endl;  
  return 0;  
}
```

EXERCISE 2 - ANSWER

- What is the output of following program.

```
{  
  int number = 13;  
  
  if (number < 10) //FALSE  
  cout<<number<<" is lesser than 10."<<endl;  
  else //TRUE  
  cout<<number<<" is greater than 10."<<endl;  
  
  cout<<"close."<<endl;  
  return 0;  
}
```

OUTPUT

13 is greater than 10.
close.

EXERCISE 3 – NESTED IF / SWITCH

- Trace the output for the given input samples:

cin>>a;

```
if (a>0)
```

```
    switch(a){
    case 1: a=a+3;
    case 3: a++;
    break;
    case 6: a=a+6;
    case 8: a=a*8;
    break;
    default: a--;
    }
```

```
else
```

```
a= a+2;
```

```
cout<<a;
```

Input (a)	Output
6	
0	
2	
1	

EXERCISE 3 --ANSWER

- Trace the output for the given input samples:

```
cin>>a;
```

```
if (a>0)
```

```
    switch(a)
```

```
    {
```

```
    case 1: a=a+3;
```

```
    case 3: a++;
```

```
    break;
```

```
    case 6: a=a+6;
```

```
    case 8: a=a*8;
```

```
    break;
```

```
    default: a--;
```

```
    }
```

```
else
```

```
a= a+2;
```

```
cout<<a;
```

Input (a)	Output
6	96
0	2
2	1
1	5

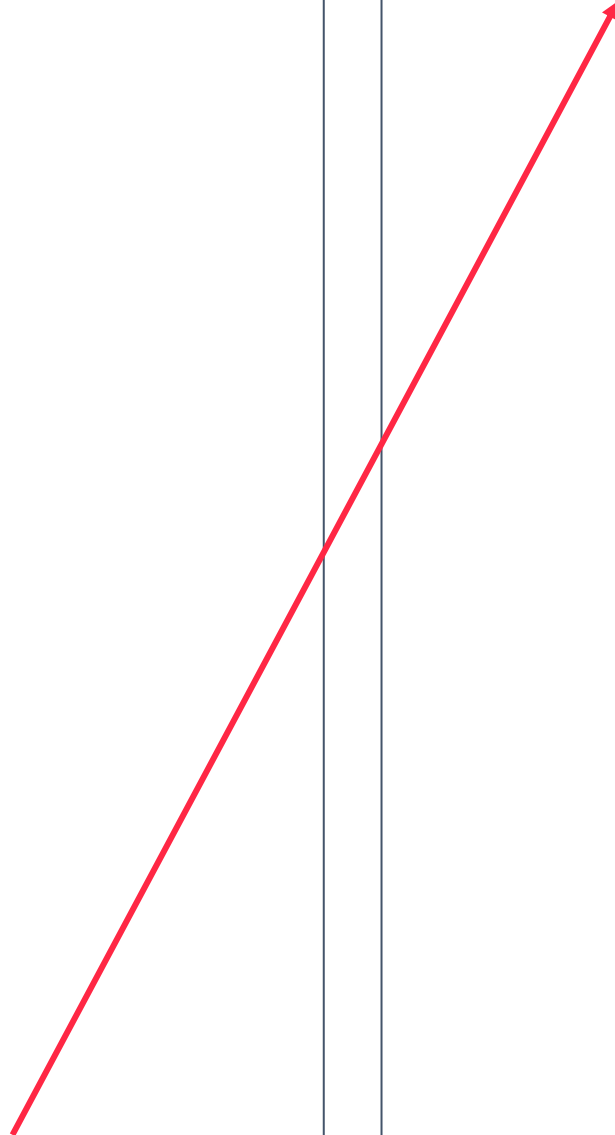
EXERCISE 4

- Write C++ program segment to accomplish the following task:
 - i. Prompt user to enter only one character. The character are as follows {A,a,B,b}. Then display whether it is uppercase or lowercase letter by using switch..case statement. If the user enters incorrect character, display the sentence “ incorrect input”.

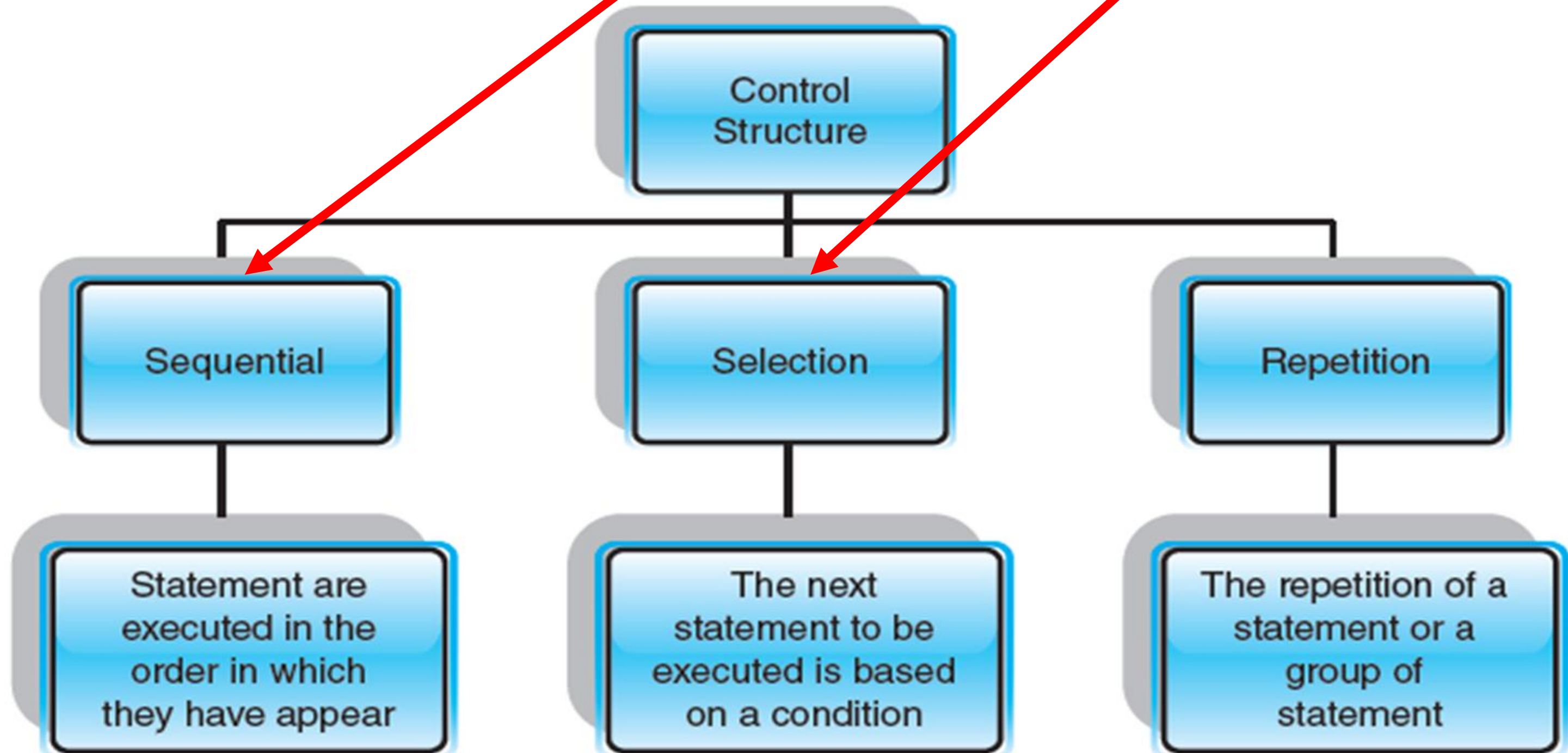
EXERCISE 4

```
char letter;  
cout<<"please enter a character";  
cin>>letter  
  
switch (letter)  
{  
    case 'A':  
        cout<<"Uppercase letter";  
        break;  
    case 'a':  
        cout<<"Lowercase letter";  
        break;
```

```
    case 'B':  
        cout<<"Uppercase letter";  
        break;  
    case 'b':  
        cout<<"Lowercase letter";  
        break;  
  
    default:  
        cout<<"Incorrect Input";  
        break;  
}
```



COMBINE SEQUENTIAL (TOPIC 2) AND SELECTION (TOPIC 3) IN ONE PROGRAM



Exercise 1 [Sequence + Selection]

Calculate a student Body Mass Index (BMI) and identify their weight status based on the result.

BMI	Weight Status
Below 18.5	Underweight
18.5 – 24.9	Normal or Healthy Weight
25.0 – 29.9	Overweight
30.0 and Above	Obese

Step by Step

SEQUENCE STRUCTURE

1. Calculate bmi first – ask user to input weight (kg) and height (m)
2. Identify formula
3. The bmi result **OUTPUT**

SELECTION STRUCTURE

1. From the bmi result – use **if statement** to make decision
2. Create **4 conditions** representing each status
3. Add else if possible

SEQUENCE STRUCTURE

```
double weight,height,bmi;

cout<<"Please enter your weight in KG :"; // input
cin>>weight;
cout<<"Please enter your height in METERS :"; // input
cin>>height;

bmi=weight/pow(height,2); // process calculation

cout<<"BMI result : "<<bmi<<endl; //output
```

SELECTION STRUCTURE

```
if (bmi>=0 && bmi<18.5) // condition 1
cout<<"Weight Status : Underweight ";

else if (bmi>=18.5 && bmi<=24.9) // condition 2
cout<<"Weight Status : Normal ";

else if (bmi>=25.0 && bmi<=29.9) // condition 3
cout<<"Weight Status : Overweight";

else if (bmi>=30) // condition 4
cout<<"Weight Status : Obese";

else // condition 5
cout<<"Out of range";
```

```

double weight,height,bmi;

cout<<"Please enter your weight in KG :"; // input
cin>>weight;
cout<<"Please enter your height in METERS :"; // input
cin>>height;

bmi=weight/pow(height,2); // process calculation

cout<<"BMI result : "<<bmi<<endl; //output

if (bmi>=0 && bmi<15) // condition 1
cout<<"Weight Status : Underweight ";
else if (bmi>=15 && bmi<=24.9) // condition 2
cout<<"Weight Status : Normal ";
else if (bmi>=25.0 && bmi<=29.9) // condition 3
cout<<"Weight Status : Overweight";
else if (bmi>=30) // condition 4
cout<<"Weight Status : Obese";

else // condition 5
cout<<"Out of range";

```

```

C:\Users\8boxc\Documents\c++\exercise1.exe
Please enter your weight in KG :75
Please enter your height in METERS :1.67
BMI result : 26.8923
Weight Status : Overweight
-----
Process exited after 10.34 seconds with return va
Press any key to continue . . .

```

Exercise 2 [Selection+ Sequence]

Create a program to calculate total ticket price to be paid, user need to choose rate of walk in or online and enter number of adult, child and senior citizen.

Walk-In rate:

Admission Ticket	Rates (MYR)
Adult (13 years old and above)	RM138
Child (12 years old and below)	RM113
Senior Citizen (60 years old and above)	RM113

SAVE RM5 when you buy ONLINE!

Admission Ticket	Rates (MYR)
Adult (13 years old and above)	RM133
Child (12 years old and below)	RM108
Senior Citizen (60 years old and above)	RM108



Step by Step

INPUT STATEMENTS

1. Ask user to select type of rate – walk in or online
2. In order to make it easier – you can put code for type of rate such as (A) for walk in and (B) for online , (1) for walk in and (2) for online
3. Ask user to enter how many adult, child and senior citizen to purchase ticket.

Step by Step

SELECTION STRUCTURE - SELECTION

4. Create 2 conditions for type of rate - walk in (1) or online (2)
5. For each rate – assign rate for adult, child and senior citizen.

SEQUENCE STRUCTURE - CALCULATE

6. Calculate total price for each category and total it all to get total fee.

DECLARE VARIABLE

```
int typeRate;  
int numAdult,numChild,numSenior;    //numbers of adult  
double adultRate,childRate,seniorRate; //rate for each category  
double totalPrice; //total price for all category
```

INPUT STATEMENTS

```
//INPUT  
cout<<"Please enter your ticket rate 1 - Walk in 2 - Online :"; // input  
cin>>typeRate;  
cout<<"Please enter how many adult :"; // input  
cin>>numAdult;  
cout<<"Please enter how many children :"; // input  
cin>>numChild;  
cout<<"Please enter how many senior citizen:"; // input  
cin>>numSenior;
```

CHECK CONDITION USING IF FOR DATA TYPE INTEGER

```
//SELECTION PART
if (typeRate==1)    //Condition 1 for Walk In Customer
{
    adultRate=138.00;    //assign rate
    childRate=113.00;
    seniorRate=113.00;
}

else if(typeRate==2)    //Condition 2 for OnLine Customer
{
    adultRate=133.00;    //assign rate
    childRate=108.00;
    seniorRate=108.00;
}

else
cout<<"Invalid Rate";    //Condition 3 for Invalid Code
```

Common Error By STUDENT

Example

adultRate=RM138.00;



ALTERNATIVE IF CHECK CONDITION USING IF FOR TYPERATE CATEGORY CHAR

```
//SELECTION PART
if (typeRate=='A' || typeRate=='a') //Condition 1 for Walk In Customer
{
    adultRate=138.00; //assign rate
    childRate=113.00;
    seniorRate=113.00;
}

else if(typeRate=='B' || typeRate=='b') //Condition 2 for OnLine Customer
{
    adultRate=133.00; //assign rate
    childRate=108.00;
    seniorRate=108.00;
}

else
cout<<"Invalid Rate"; //Condition 3 for Invalid Code
```

Common Error By STUDENT

Example

if (typeRate==A || typerate==a); 

CALCULATE TOTAL PRICE

```
//CALCULATE PART
totalPrice = (numAdult*adultRate)+(numChild*childRate)+(numSenior*seniorRate);

//OUTPUT 2 decimal place - DONT FORGET #include <iomanip>
cout.setf(ios::fixed);
cout.precision(2);

cout<<"Total Price for All Tickets = RM "<<totalPrice;

return 0;
```

CALCULATE TOTAL PRICE – MAKE IT BETTER OUTPUT

```
//CALCULATE PART
totalPrice = (numAdult*adultRate)+(numChild*childRate)+(numSenior*seniorRate);

//OUTPUT 2 decimal place - DONT FORGET #include <iomanip>
cout.setf(ios::fixed);
cout.precision(2);
cout<<"SUNWAY LAGOON TICKET\n";
cout<<"Total Adult = "<<numAdult<<endl;
cout<<"Total Ticket Price Adult = RM "<<numAdult*adultRate<<endl;
cout<<"Total Child = "<<numChild<<endl;
cout<<"Total Ticket Price Children = RM "<<numChild*childRate<<endl;
cout<<"Total Senior = "<<numSenior<<endl;
cout<<"Total Ticket Price Senior Citizen = RM "<<numSenior*seniorRate<<endl;

return 0;
```