Lesson 5: Program control structures in C++

5.1. Introduction

In our previous lesson, we learnt about tokens, variables, data types and constants in C++. In this lesson, we will discuss program control structures in C++. As discussed in programming methodology class there are three program control structures: sequence, selection and repetition structures.

5.2. Lesson objectives

By the end of this lesson, you will be able to:

- Describe sequence structures
- Describe selection structures
- Describe the repetition structures

5.3. Lesson outline

This lesson is structured as follows:

5.4. Sequence structure

By default all programs follow sequence structure. This means that the statements are executed one after another.

See the example below: A closer look at this program shows that the value of var will displayed will be changing from line 8 to line 12.

```
#include<iostream>
using namespace std;
void main()
{
        float var;
        cout << "Enter value of var:" << endl;
        cin >> var;
        cout << "The value of var: " << var << endl;
        var = var + 10;
        cout << "The value of var: " << var << endl;
        var = var * 2;
        cout << "The value of var: " << var << endl;
}
```

5.5. Selection structure

This is a program control structure that allows the program to execute one or more statements based on a condition. Examples of selection structure:

a) Single if statement: This statement allows one or more statements to be executed if the condition is true. See the example below:

#include<iostream> using namespace std;

```
void main()
{
        float age;      cout << "Enter you age:" << endl;    cin >> age;
                        if (age >= 0) //one condition tested
                        {
                                cout << "Age is above 0" << endl;
                        }
}
```

b) Nested if statement

This statement allows more than one condition to be tested. The first condition is tested .If the condition is true then some statements are executed but if the condition is false, then the next condition is tested. This continues until all the conditions are tested. Finally, the keyword else is used.

```
}#include<iostream> using namespace std; void main()
{
        float age;      cout << "Enter you age:" << endl;
        cin >> age;     if (age >= 0&&age<=18) //first condition tested
                        {
                                cout << "Minor:" << endl;
                        }
                        else if (age >= 19 && age <= 35) //second  condition tested
                        {
                                cout << "Young adult:" << endl;
                        }
                        else if (age >= 36) //last condition tested
                        {
                                cout << "Adult:" << endl;
                        }
                        else  //after you have tested all the condition
                        {
                                cout << "Invalid value try again:" << endl;
                        }
}
```

c) Switch case statement

This statement is similar to if statement but can allow several conditions to be tested. It uses the keyword switch.

```
#include<iostream> using namespace std; void main()
{
 int floor;  cout << "Enter the floor number:" << endl;  cin >> floor;  switch (floor)
 {  case 0:
                        cout << "This is the ground floor:" << endl;
        break;  case 1:
                        cout << "This is the first floor:" << endl;
        break;  case 2:
                        cout << "This is the second floor:" << endl;
        break;  default:
```

```
                                        cout << "The value you have entered is wrong try
again:" << endl;
                                                }
}
```

5.6. Repetition structures

The repetition structure forces the program to execute a code repeatedly until a certain condition is fulfilled. In every repetition/loop structure you must initialize the loop control variable usually before the loop starts. Then you must test the loop control variable usually done within the parenthesis of the while statement or for statement. Finally, update the loop control variable somewhere inside the loop, usually as the last statement in the loop. This structure is very common and is used in performing repetitive work in programming. There three types of repetition structure:

[i]. For loop

It is one of the most common types of loops. The initial expression is initialized only once at the beginning of the loop. Then the test expression is checked by the program. If the test expression is false, for loop is terminated. If the test expression is true, then the codes are executed and update expression is updated. The test expression is checked again. If it is false, the loop is terminated and if true the same process is repeated until test expression is false.

Syntax

```
For (variable initialization; condition; variable update )
{
  Code to execute while the condition is true
}
```

The variable initialization: Declare and initialize the counter variable that will be used in the loop.

Test loop: The conditional expression to be tested before execution. If it is true the loop should continue to repeat itself.

Variable update: It can increment or decrement the variable (counter).

NB: Notice that a semicolon separates each of these sections. Also note that every single one of the sections may be empty, though the semicolons still have to be there. If the condition is empty, it is evaluated as true and the loop will repeat until something else stops it.

Example: A program to print a string. The iterations are known.

```
#include<iostream> using namespace std; void main()
{
            int i;
        cout <<"This  program will print this is crazy 10 times!:"<<endl;
          for (i = 0; i < 10; i++)
            {
                    cout << "This is crazy:" << endl;
            }
}
```

[ii]. while loop

It is a repetition structure that works in much the same way as for loop. The while loop can be used if we do not know how many times a loop must run. First you must always initialize the counter before the while loop starts. Then, the expression is tested. If it is true, the codes inside the body of the while loop are/ is executed. Again the test expression is tested and the process is repeated until the test expression is false. Update the counter inside the loop otherwise the loop becomes infinitive.

Syntax

```
Variable initialization
while ( test expression )
{
statemen(s)//body of the loop update variable;
}
```

Variable initialization: The counter is given initial value e.g. counter=0.

Condition (test expression): Condition to be tested if true or false .It can be any combination of Boolean statements that are legal.

Body of the loop: Consists of the statement(s) that are executed while the condition is true.

Update variable: To increment/decrement the counter variable in order to avoid infinite loop.

Example: A program to print integers from 1 to 9 using while. The number of times not known only a condition is used.

```
#include<iostream> using namespace std; void main()
{
        int num;        num = 1;
            while (num<10)
            {
                    cout<<num<<endl;
                    num++;
            }
}
```

[iii]. do…while loop //number of iterations not known in advance. Only condition is used.

This program control structure is related to the while loop. The difference is that in whiledo loop, the test expression is checked first but in do-while loop the code is executed first then the test expression is checked last. Do while will execute statement(s) first and then test expression. The result is that the loop always runs at least once. Notice there is a semicolon after the while statement. This loop is useful for things that must loop at least once. **Syntax**

```
Do
 {
Statement(s);
Update variable;
 }
while ( test expression );
```

Example: Program to print numbers between 0 and 9.

```
#include<iostream> using namespace std; void main()
{
        int num;        num = 0;
            do
            {
                    cout<<num<<endl;
                    num++;
            } while (num<10);
}
```

Study the above program, compile, build and run it.

5.7. Revision questions

a) Using examples explain the difference between a while and do-while loop.

b) Give the syntax of :

[i]. For loop

[ii]. Do…while loop

[iii]. While loop

5.8. Summary

In this lesson you have learnt about three program control structures: sequence, selection and loop control structures. You have noted that sequence structure is a default structure of a program. You have also learnt that selection structure requires a condition to be tested and then the flow will depend on this test.

5.9. Suggested reading

[1]. Object oriented programming with C++ by E Balagurusamy 3rd ed; publisher: Tata Mcraw Hill

[2]. Sams teach yourself c++ in 24 hours by Jesse Liberty and Rogers Cadenhead. [3]. Object oriented programming in c++ by Joyce Farrel [4]. Object-oriented programming with c++ by Sourav Sahay.