Discipline Courses-I
Semester-I
Paper: Programming Fundamentals
Unit-II
Lesson: Selection Structures

Lesson Developer: Tarang Jain
College/Department: Moti Lal Nehru College, University of Delhi

Table of Contents

Chapter 6. Selection Structure-I

Welcome to Chapter 6! In the previous chapter, you learned about the conversion of variables form one data type to another. In this chapter, you will learn about the selection structures. It is a very important concept. By now, all the programs were sequential in nature. That means we started from the first statement of the program and moved to the end of the program executing all the statements sequentially only once. But now we will be able to learn about the selection of statements in a program and how to write programs using selection structures.

Learning Objectives

After reading this chapter you should be able to change the flow of control of your program from sequential to selection.

- Define what do you mean by selection?
- Discuss the “if” statement used for selection.
- Discriminate between “if” and “if-else” statements.
- Write C++ programs using these selection structures.

6. Selection Structure-I

By now whatever programs we have discussed were sequential in nature. It means that we start from the very first line/statement of the program. One by one, we move to the last line of the program.

Sometimes we want to execute statement/s only if some of the conditions are true. Otherwise we want to skip these statement/s. Now this selection of the statements will depend upon the values given by the user. For instance, if there is rain, then you will have to open an umbrella otherwise not. If you are thirsty, then you will have to drink something to end your thirst otherwise if you are hungry you will have to eat something. To handle these kinds of problems, we use selection structures. There are basically two types of selection structures used in C++. These are:

1. "if" structure
2. "switch" structure

In this chapter, we are going to discuss about the "if" structure only. The "switch" structure will be discussed later in the next chapter.

6.1 The "if" Statement

It is used for selective execution of a program segment. The general syntax for this statement is:

Syntax*:* *if (expression)*

```
if (expression)
{
    Statement/s
}
Statement 2;
```

Here, ***if*** is a reserved word, ***expression*** gives a Boolean output. There can be one or more statements in the block. If the Boolean output of the expression is true, then the block of statements will be executed and then the control will go to statement 2. Otherwise these statements will be skipped and the control will directly go to statement 2.

Let us try to understand it using some examples.

```
/* 6.1 write a program to calculate the bonus given to an employee. if the
salary of the employee is greater than Rs. 5000 then 5% is given as bonus
otherwise 0% bonus is given.*/

#include <iostream>                              //header file

using namespace std;

int main()                                       // main function
{
                int salary, bonus=0;             //variables declaration
                float totSal;
```

```
                    cout<<"Enter the salary of the employee  ";
                    cin>>salary;                    //input statement

                    if (salary>5000)                //if statement to find the bonus
                            bonus=5;

                    totSal=salary + salary*bonus/100;
                    cout<<" The new salary is  "<<totSal;

                    return(0);
}
```

The output of the above program will be:

```
Enter the salary of the employee  4000
The new salary is  4000

Enter the salary of the employee  9000
The new salary is  9450
```

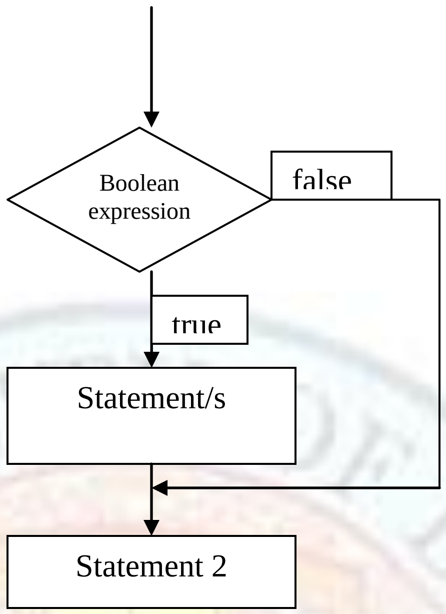
Value addition: Note

Heading text :if statement

Body text:

As it is clear fro the above program, if there is only one statement in the “if” block, then the curly brackets are not compulsory. You can still put curly brackets; it will not give any error.

An equivalent flowchart for the above syntax of “if” is:



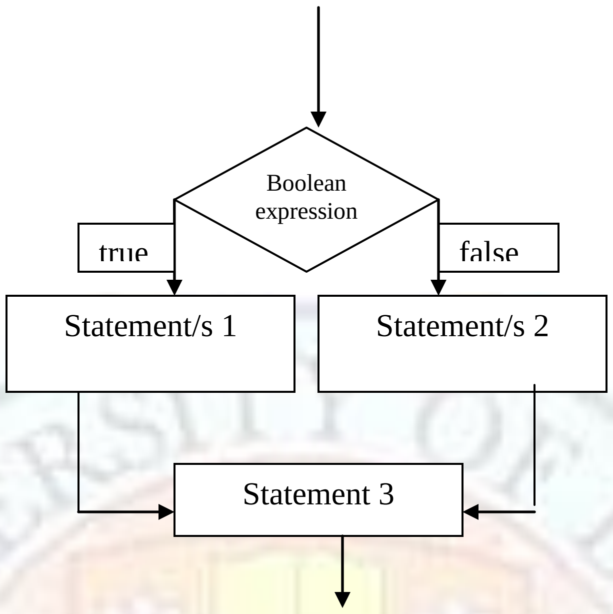
6.2 The “if-else” Statement

Another form of ”if” statement is “if-else” statement. The syntax for it is:

Syntax: *if (expression)*

```
          {
                statement/s 1
          }
          else
          {
                statement/s 2
          }
          Statement 3
```

Here ***if*** and ***else*** , both are reserved words. Now if the Boolean expression is true, then *statement/s 1* will be executed and then the control will move to *statement 3*. otherwise, if the Boolean expression is false then the *statement/s 2* will be executed and then the control will move to *statement 3*. An equivalent flowchart for the above syntax is:



Here, the Boolean expression is tested. If the output of the Boolean expression is true then statement/s 1 is executed and then control will transfer to statement 3. Statement/s 2 will not be executed in this case. But if the output of the Boolean expression is false, then statement/s 2 will be executed and then the control will move to statement 3. Now the statement/s 1 will not be executed. So depending upon the output of the Boolean expression, either of the statement/s 1 or statement/s 2 will be executed.
Let us try to understand it using an example:

```
/* 6.2 Write a program to find largest of two integer numbers */

#include <iostream>                         //header file

using namespace std;

int main ()                                         // main function
{
          int firstNum, secondNum, largest;  //variables declaration


          cout<<"Enter two numbers ";
          cin>>firstNum>>secondNum;                  //input statement

          if (firstNum>secondNum)
                         //if-else statement to find the largest
               largest=firstNum;
          else
               largest=secondNum;
```

```
                cout<<" The largest number is  "<<largest;

                return (0);
}
```

The output of the above program will be:

```
Enter two numbers 2345 8746
The largest number is  8746
```

6.2.1 The Nested if: Nested *if* is very commonly used while writing programs. Here the “if” statement includes another “if”. This “if” can be in the “if” part or the “else” part. Let us try to understand it using a program.

```
/* 6.2.1  Write a program to find largest of three integer numbers */
#include <iostream>                          //header file
using namespace std;

int main ()                                          // main function
{
        int firstNum, secondNum, thirdNum, largest;  //variables declaration


        cout<<"Enter three numbers ";
        cin>>firstNum>>secondNum>>thirdNum;          //input statement

        if (firstNum>secondNum)
                if(firstNum>thirdNum)
                        largest=firstNum;
                else
                        largest=thirdNum;
        else
                if(secondNum>thirdNum)
                        largest=secondNum;
                else
                        largest=thirdNum;

        cout<<" The largest number is  "<<largest;

        return (0);
}
```

The output of the above program will be:

```
Enter three numbers 2345 8746 3456
The largest number is  8746
Enter three numbers 9345 8746 3456
The largest number is  9345
Enter three numbers 2345 8746 9456
The largest number is  9456
```

6.2.2 The if-else-if ladder: It is also known as "*if*-*else*-*if staircase*". The syntax for this ladder is:

```
if (expression)
        {
                statement/s 1
        }
        else
                if (expression)
                {
                        statement/s 2
                }
                …
                else
                        statement 3
```

Here the *expression* is evaluated from the top of the ladder. As soon as the condition meets, the corresponding statements are evaluated and rest of the ladder is bypassed. If no *expression* evaluates to true value, then the final *else* statements (statement 3) are evaluated (If there is no *else* part in the ladder, then no statements of the ladder will be executed.). Let us try to understand it using a program:

```
/*6.2.2  Write a program to find roots of a quadratic equation using "if"*/
#include <iostream>                          //header file
#include <math.h>                            // for sqrt function
using namespace std;
int main()                                   // main function
{
                float a,b,c,d,root1,root2;   //variables declaration

                cout<<"Enter the coefficients a,b,c of a quadratic equation ";
                cin>>a>>b>>c;                        //input statement
                d=b*b-4*a*c;

                if(d==0)               //if-else-if ladder
                {
                   cout<<"The roots are real and equal"<<endl;
                   root1=root2=-b/(2*a);
                   cout<<"roots are "<<root1<<"\tand "<<root2<<endl;
                }
                else if(d>0)
                {
                   cout<<"The roots are real and unequal"<<endl;
                   d=sqrt(d);
                   root1=(-b+d)/(2*a);
                   root2=(-b-d)/(2*a);
                   cout<<"roots are "<<root1<<"\tand "<<root2<<endl;
                }
                else
                {
                   cout<<"The roots are complex and unequal"<<endl;
                   d=sqrt(-d);
                   float rpart=-b/(2*a);
                   float ipart=d/(2*a);
                   cout<<"Roots are "<<rpart<<"+i"<<ipart<<"\tand "<<rpart<<"-
i"<<ipart<<endl;
                }
                return(0);
}
```

The output of the above program would be:

```
Enter the coefficients a,b,c of a quadratic equation 2 2 2
The roots are complex and unequal
Roots are -0.5+i0.866025    and  -0.5-i0.866025

Enter the coefficients a,b,c of a quadratic equation 2 6 2
```

```
The roots are real and unequal
Roots are -0.381966 and  -2.61803


Enter the coefficients a,b,c of a quadratic equation 2 4 2
The roots are real and equal
Roots are -1 and -1
```

Value addition: Very Important Note

Heading text: Nested if statement

Body text:

One thing has to be remembered always that the *else* is associated with the nearest unmatched "if". For example:

```
if(x>10)
    if(y>10)
       x++;
    else
       y++;
```

Here the "else" is associated is innermost "if". If you want to associate this "else" with outer "if", then you will have to use brackets as shown below:

```
if(x>10)
{
    if(y>10)
       x++;
}
else
   y++;
```

6.3 The Logical Operator

You already know about operators. We have already discussed about arithmetic and relational operators. Now we will discuss about logical operators. As is the case with relational operators, the output of the logical operators is also a Boolean value (true/false). When we use logical operators, the operands should be Boolean values. If these operands are not Boolean values but these are some numeric values, then these will be converted into Boolean values by following the rule of converting no-zero values to true. We can say that a logical operator is used to connect relational or logical expressions. In C++, three logical operators are allowed.

6.3.1 AND-OR-NOT Operator

The three logical operators allowed in C++ are *AND (&&)*, *OR (||)*, and *NOT (!)*. AND and OR are binary operators while NOT is an unary operator. It means, for AND and OR we need two operands, while for NOT we need only one operand. In case of AND operator, the output will be true only when all the input are true. In case of OR operator, the output will be false only when all the inputs are false. In case of NOT operator, the output will be reversed. The following table will clear the picture.

A	B	A&&B	A||B	!A
F	F	F	F	T
F	T	F	T	T
T	F	F	T	F
T	T	T	T	F

Let us write a program using logical operators:

```
/* 6.3 Write a program to check whether a year is a leap year or not */

#include <iostream>                         //header file

using namespace std;

int main()                                          // main function
{
            int year;       //variables declaration


            cout<<"Enter the year to be checked ";
            cin>>year;                  //input statement

            if ((year%400==0)||((year%100!=0)&&(year%4==0)))
      //logical operators in if statement
                  cout<<year<<" is a leap year";
            else
                  cout<<year<<" is not a leap year";


            return(0);
}
```

```
The output of the above program would be:

Enter the year to be checked  1900
1900 is not a leap year

Enter the year to be checked  2004
```

```
2004 is a leap year
```

6.3.2 Precedence of Operators Revisited

The precedence of *logical AND* and *logical OR* operator is lower than all operators discussed till now except the assignment operators as shown in the following table. The precedence of *logical not* operator is same as that of other unary operators.

Highest precedence	()
Unary operators	+ - ! ++ --
Multiplicative operators	* / %
Additive operators	+ -
Relational operators	< <= > >=
Equality operators	== !=
Logical operators	&& ||
Assignment operators	= += -= *= /=
lowest precedence	

TABLE 6.1: Precedence of operators

Chapter 7. Selection Structure - II

Welcome to Chapter 7! In the previous chapter, you learned “if” and “if-else” which are used for selective execution of a program. In this chapter, you will learn about “switch” which is also used for selection. You will also be able to learn about the conditional operator which is a short form of the “if-else” structure. You will also learn to write C++ programs using this conditional operator.
This will help you understand the concept of “switch” statement and conditional operator used in a C++ program.

Learning Objectives

After reading this chapter you should be able to:

1. Recognize sequential and selection statements in a C++ program.
2. Define when a “switch” statement is used
3. Discriminate between “if-else” and “switch” statements.
4. Discuss the use of conditional operator.
5. Write simple C++ programs using “switch” statement and conditional operator.

7. Selection Structures-II

In the previous chapter we have discussed about "if" statement and "if-else" statement. Sometimes we need to select one alternative from many different alternatives available depending upon the condition. In that case either we can use multiple alternative "if" statement (if-else-if ladder), or "switch" statement.

7.1 "switch" statement

It is used when one alternative has to be selected from many alternatives. If the expression to be tested is of type integer or character, then this statement is much more useful than "if-else-if" ladder. The syntax of this statement is:

Syntax:

```
switch (expression)
{
        case constant 1:
                statement/s;
                break;
        case constant 2:
                statement/s;
                break;
        .
        .
        .
        default:
                statement/s;
}
statement n;
```

Here *switch, case, default* and *break* are reserved words.

The value of the expression is matched with the constant values of the *case.* If a match is found then the execution of a statement will start from that *case* and it will continue till either a *break* statement is found or end of the switch statement comes. Then it will go to *statement n*. If no match is found then the control will jump to *default*. If there is no *default* in a switch statement then the control will come out of the switch statement and will go to *statement n*.

Let us write the program to find the roots of a quadratic equation using switch statement. We have already done this program using "if". So it will make you clear the difference between the "if" and the "switch" structure.

```
/*7.1 Write a program to find roots of a quadratic equation */

#include <iostream>                         //header file
#include <math.h>                           // for sqrt function
using namespace std;
int main()                                  // main function
{
            int choice;                     //variables declaration
            float a,b,c,d,root1,root2;

            cout<<"Enter the coefficients a,b,c of a quadratic equation ";
            cin>>a>>b>>c;                           //input statement

            d=b*b-4*a*c;

            if (d==0)                       //to set the value of the variable choice
                  choice=1;
            else if (d>0)
                  choice=2;
            else
                  choice=3;

            switch (choice)                 //switch statement
            {
             case 1:
                  cout<<"The roots are real and equal"<<endl;
                  root1=root2=-b/(2*a);
                  cout<<"roots are "<<root1<<"\tand "<<root2<<endl;
                  break;
             case 2:
                  cout<<"The roots are real and unequal"<<endl;
```

```
                        d=sqrt(d);
                        root1=(-b+d)/(2*a);
                        root2=(-b-d)/(2*a);
                        cout<<"roots are "<<root1<<"\tand "<<root2<<endl;
                        break;
                case 3:
                        cout<<"The roots are complex and unequal"<<endl;
                        d=sqrt(-d);
                        float rpart=-b/(2*a);
                        float ipart=d/(2*a);
                        cout<<"Roots are "<<rpart<<"+i"<<ipart<<"\tand
"<<rpart<<"-i"<<ipart<<endl;
                        break;
               }
               return(0);
}
```

The output of the above program would be:

```
Enter the coefficients a,b,c of a quadratic equation 2 2 2
The roots are complex and unequal
Roots are -0.5+i0.866025    and  -0.5-i0.866025

Enter the coefficients a,b,c of a quadratic equation 2 6 2
The roots are real and unequal
Roots are -0.381966 and  -2.61803

Enter the coefficients a,b,c of a quadratic equation 2 4 2
The roots are real and equal
Roots are -1 and -1
```

As it is clear by now that "switch" structure can compare for equality only. While "if" can be used with all relational or logical operators.

Value addition: break

Heading text: switch statement

Body text:

break is used to break the flow of control. If we will not use break in the "case" of a "switch" structure, then the execution of statements of all cases will be done after a match is found. But since we want to execute the statements of that case only where a match has found, we have to use "break" statement to break the flow of control and to come out of the "switch" structure.

7.2 Menu Driven Program

"switch" statement is very useful in menu driven programs. Let us discuss it using an example.

```
/*7.2 Write a program to design a simple calculator which can do simple arithmetic
calculations */

#include <iostream>                                //header file
using namespace std;
int main()                                         // main function
{
            int choice;                            //variables declaration
            float num1,num2,result;
            cout<<"Enter two integer values ";
            cin>>num1>>num2;                       //input statement
            cout<<"1. addition\n2. subtraction\n3. multiplication\n";
            cout<<"4. multiplication\n\nEnter your choice";

            cin>>choice;

            switch (choice)
            {
             case 1:
                  result=num1+num2;
                  cout<<num1<<"+ "<<num2<<"="<<result<<endl;
                  break;
             case 2:
```

```
                          result=num1-num2;
                          cout<<num1<<"- "<<num2<<"="<<result<<endl;
                          break;
                case 3:
                          result=num1*num2;
                          cout<<num1<<"* "<<num2<<"="<<result<<endl;
                          break;
                case 4:
                          result=num1/num2;
                          cout<<num1<<"/ "<<num2<<"="<<result<<endl;
                default:
                          cout<<"Enter a proper choice ";
              }
              return(0);
}
```

The output of the above program would be:

```
Enter two integer values 23 56
1.  addition
2.  subtraction
3.  multiplication
4.  division

enter your choice 2
23-56=-33
Enter two integer values 23 56
1. addition
2. subtraction
3. multiplication
4. division

Enter your choice 3
23*56=1288


Enter two integer values 23 56
```

```
1. addition
2. subtraction
3. multiplication
4. division

Enter your choice 4
23/56=0.410714

Enter two integer values 23 56
1. addition
2. subtraction
3. multiplication
4. division

Enter your choice 5
Enter a proper choice
```

7.3 The conditional operator

By now we have discussed unary and binary operators. Now we will discuss a ternary operator i.e. the conditional operator. This operator is known as ternary operator because it works on three operands. It is used for performing some simple conditional operations. It is used in place of "if" statement. Using this operator, the size of the program will be reduced. Let us try to understand this operator using the same example program as we have discussed in case of "if" statement.

```
/* 7.3 Write a program to calculate the bonus given to an employee. if the
salary of the employee is greater than Rs. 5000 then 5% is given as bonus
otherwise 0% bonus is given.*/

#include <iostream>                         //header file

using namespace std;

int main()                                  // main function
{
                int salary,bonus;           //variables declaration
                float totSal;
                cout<<"Enter the salary of the employee  ";
                cin>>salary;                    //input statement

                bonus=(salary>5000)?5:0;            //conditional operator
                totSal=salary + salary*bonus/100;
                cout<<" The new salary is  "<<totSal;

                return(0);
}
```

```
The output of the above program will be:

Enter the salary of the employee  4000
The new salary is  4000

Enter the salary of the employee  9000
The new salary is  9450
```

Let us use this conditional operator for the program we have discussed for “if-else” statement.

```
/*7.4  Write a program to find largest of two integer numbers */

#include <iostream>                              //header file

using namespace std;

int main()                                       // main function
{
                int firstNum, secondNum, largest;   //variables declaration


                cout<<"Enter two numbers ";
                cin>>firstNum>>secondNum;                //input statement

                        //conditional operator to find the largest
                largest=(firstNum>secondNum)?firstNum:secondNum;
                cout<<" The largest number is  "<<largest;

                return(0);
}
```

The output of the above program will be:

```
Enter two numbers 2345 8746
The largest number is  8746
```

Summary

- The flow of control of a program is not always sequential.
- For selection of statements, “if” or “if-else” statements can be used.
- The logical operators are applied on relational or logical expressions.
- There are three logical operators. These are &&, || and !.
- Using selection structures, the statements are executed either zero time or one time.
- In some cases, “switch” statement and “if-else” statement are interchangeable.
- “switch statement is more useful than “if-else” statement when there is equality comparison with integer or character.
- “switch” statement is very useful for menu driven programs.
- The conditional operator provides a short form for “if-else” statement.

Exercises

6.1 Differentiate between the following:

a. "if" statement and "if-else" statement

b. Relational operators and Logical operators

c. AND operator and OR operator

6.2 Write a program in C++ to find the largest of three numbers.

6.3 Write a program in C++ to check whether a number is even or odd.

6.4 Evaluate the following expression for the given values of x, y and z:

(x>=y)||(!z==y)&&(z<x)

1. x=5, y=8, z=9
2. x=7, y=10, z=2
3. x=9, y=9, z=9

6.5 What will be displayed when the following code is executed?

```
#include <iostream>
using namespace std;
int main()
{
        int i=0;
        if (i=5)
                cout<<i;
        cout<<i;
        return 0;
}
```

Output : 55

7.1 ✓ Differentiate between the following:

a. ✓ Conditional operator and "if-else" construct

b. ✓ "if-else" and "switch"

7.2 Write a program in C++ to check whether a number is even/odd using conditional operator.

7.3 Write a program in C++ to find the largest of three numbers using conditional operator.

7.4 Write a menu driven program in C++ using "switch" statement. The choices are:

a. Largest of two numbers

b. Largest of three numbers

c. Number is even/odd

7.5 ✓ What will be displayed when the following code is executed?

```
#include <iostream>
using namespace std;
int main()
{
        int i, j=5;
        i=(j!=5)?5;0;
        cout<<i<<j;
        return 0;
}
```

0 5

Glossary

Conditional Operator- It is a ternary operator equivalent to "if-else".

Flow Chart- It is a graphical representation of a flow of a program using some specific symbols.

Hardware- It is that part of the computer which can be seen and touched. For example, monitor, mouse, printer, CPU, hard disk etc.

High-level Language- A computer language which is not dependent on the hardware. For example, C, C++, Java, Pascal etc.

if- It is a reserved word. It is used for selective execution of a program.

Program- It is a set of instructions written in high level language to solve a problem.

Selection – Selection means to choose from the alternatives available.

Statement- It is an expression ended with a semicolon.

Ternary Operator- In a ternary operator, there are three operands.

References

1. A.K.Sharma, Introductory Computer Science (Volume II), Dhanpat Rai Pub.
2. B. A. Forouzan and R. F. Gilberg, Computer Science, A structured Approach using C++, Cengage Learning, 2004
3. E. Balaguruswamy, Object Oriented Programming with C++, Tata Mcgraw Hill
4. H. Schildt, C++: The Complete Reference Book, Tata Mcgraw Hill