Lab 3
Branching

When instructions within a program are executed one after the other sequentially that program is said to have a linear structure. Decision making after examining all available options is very important in life as well as in programming. For example, it is the law that all males 18 or older should register with the selective service. If you are writing a program to send out reminders to enforce this law, the decision to send the letter should be based on if a person is male and if he is 18 or older. In this chapter you will learn how to write statements that make decisions. A simple program to look at an average grade of a student and display if that student passed or failed would look like this (Program 3-1):

There are several enhancements to this program. I added #include <iomanip> which allows the output to be formatted. There are several stream manipulators in the "iomanip". I used setw, fixed, showpoint and setprecission manipulators to give field width, fixed-point notation (rather than scientific notation), decimal point, and significant digits respectively. I also used cin.ignore() before the getchar(). The cin.ignore() will clear the keyboard buffer of the newline character.

Program 3-1.

```
/**************************************
Accept three grades, find the average
   and display Passed or Failed.
Teach objective  - making decisions
By Dr. John Abraham
Created for 1380 students
**************************************/

#include <iostream>
#include <iomanip>  //to format input and output. Here setw and endl require it
#include <fstream>
using namespace std;
void printToFile(int, int, int, double);

int main ()
{
 int one, two, three;
 double average;


       cout << "Enter three grades ";
       cin >> one >> two >> three;
       average= (one+ two+ three)/3.0; // the total of three grades are first converted
float
       cout << "Three grades are: "  <<one <<setw(4)<<two <<setw(4)<<three <<endl;
         //endl inserts line feed and carriage return
       cout <<fixed<<showpoint<<setprecision(2); //What does this line do?
       cout << "The average is :  " <<average <<endl;
       if (average >= 70) cout << "Student passed the course!";
       else cout << "Student failed the course!";
       printToFile(one,two,three, average);
```

```
        cin.ignore();
        getchar();

 return (0);

}


void printToFile(int one, int two, int three, double average)
{
       ofstream outfile;
       outfile.open("ifThenElse.txt");

       outfile << "Three grades are: "  <<one <<setw(4)<<two <<setw(4)<<three <<endl;
       outfile <<fixed<<showpoint<<setprecision(2);
       outfile << "The average is :  " <<average <<endl;
        if (average >= 70) outfile << "Student passed the course!";
        else outfile<< "Student failed the course!";
        outfile.close();
}
```

Program Run 3-1

```
Enter three grades 88 44 99
Three grades are: 88  44  99
The average is :  77.00
Student passed the course!
```

The decision was made using the if-else statements:

```
if (average >= 70) cout << "Student passed the course!";
else cout << "Student failed the course!";
```

We could add compound statements under the if condition or the else condition or both like this:

```
if (average >=70)
        {
          cout << “Student Passed the Course!\n”;
          cout << “The student is allowed to take the next course.”;
        }
else
        {
          cout << “Student failed the course!\n”;
          cout << “This course must be repeated before taking the next course!”;
        }
```

The operator symbols we used here >= are called **Relational Operators. Relational operators are == (equal to), > (greater than), < (less than), != (not equal to), >= (greater than or equal to), and <= (less than or equal to).** The result of a relational operation is either **false** or **true**. Variables that hold **false** or **true are** called **bool** variables.

There were only two alternatives in the above situation, either passed or failed. In actual grading we want to go beyond pass/fail; we want to assign a letter grade of A, B, C, D, or F. Let us rewrite the above program to handle the multiple alternatives. See Program 3-2.

Program 3-2.

```
/**************************************
Accept three grades, find the average
   and display Passed or Failed.
Teach objective  - making decisions
By Dr. John Abraham
Created for 1380 students
**************************************/

#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;
char getgrade (double);//to find letter grade
void printToFile(int, int, int, double, char);

int main ()
{
 int one, two, three;
 double average;
 char grade;


       cout << "Enter three grades ";
       cin >> one >> two >> three;
       average= (one+ two+ three)/3.0;
       cout << "Three grades are: "  <<one <<setw(4)<<two <<setw(4)<<three <<endl;
       cout <<fixed<<showpoint<<setprecision(2);
       cout << "The average is :  " <<average <<endl;
       grade = getgrade(average);
       cout << "The letter grade is : "<< grade << "\n";

       printToFile(one,two,three, average, grade);
       cin.ignore();
       getchar();

 return (0);

}

char getgrade (double average)
 {
   if (average >=90)return ('A');
   else if (average>= 80) return('B');
```

```
        else if (average >=70) return ('C');
             else if (average >= 60) return ('D');
                  else return('F');
 }

void printToFile(int one, int two, int three, double average, char grade)
{
      ofstream outfile;
      outfile.open("ifThenElse.txt");

      outfile << "Three grades are: "  <<one <<setw(4)<<two <<setw(4)<<three <<endl;
      outfile <<fixed<<showpoint<<setprecision(2);
      outfile << "The average is :  " <<average <<endl;
      outfile << "The letter grade is : "<< grade << "\n";
      outfile.close();
}
```

In the getgrade function if the average is 90 or above the function returns an A and ends the function; it only continues if the average is not 90 or above. You may want to modify the program as follows to avoid many **return** statements. The program reads better when you only have one **return**.

```
char getgrade (double average)
{
 char grade;

   if (average >=90) grade = 'A';
   else if (average>= 80) grade = 'B';
        else if (average >=70) grade ='C';
             else if (average >= 60) grade ='D';
                  else grade ='F';
  return (grade);
 }
```

Program Run 3-2.

```
Enter three grades 90 93 89
Three grades are: 90  93  89
The average is :  90.67
The letter grade is : A

Enter three grades 66 58 52
Three grades are: 66  58  52
The average is :  58.67
The letter grade is : F

Enter three grades 77 82 75
```

```
Three grades are: 77  82  75
The average is :  78.00
The letter grade is : C
```

Any time you write a program for multiple alternatives, the program should be run to check every alternative. In program Run 3-2 only three alternatives are tested. If you were to turn this program in for a grade, you should include all the alternatives.

Logical operators work with boolean values or results of relational operations. Logical operators are: **AND (&&)**, **OR (||)**, and **NOT (!)**. For each of this operation we can obtain a **truth table**.

T && T => T	**T || T => T**	**!T => F**
T && F => F	**T || F => T**	**!F =>T**
F && F => F	**F || F => F**	

Suppose the average score is 95. Let us try this statement:

```
If (average >=90 && average <= 100)
        cout << "Your grade is A \n";
```

The first relational operation of average >= 90 yields a T. The second operation of average <= 100 also yields a T. T && T gives a T. Since the entire operation yields a true then "Your grade is A" is displayed. If, on the other hand, the average score is 88, the first operation will yield a false and the second operation will yield a true (88 is less than 100); F && T is False, and the output will not be displayed. The concept of logical and relational operators will become clearer in the next chapter when we deal with repetitions.

Suppose you created a menu to choose one of the items from a list you may have to write some thing like this:

```
If (choice==1)  AddClient ();
else if (choice == 2) EditClientInfo ();
     else if (choice==3) LookUpClient();
and so on…
```

If the menu has many items there is a lot of coding you have to do and the code is hard to follow. There is alternative to the multiple if/else statements. We can use the **switch statement** as shown below.

```
Switch (choice)
{
  case 1: AddClient();
          break;
  case 2: EditClientInfo();
          break;
```

```
 case 3: LookUpCleint();
         break;
 and so on..
 default :
         cout << “invalid response”;

}
```

We are essentially telling c++ to do something in case of choice is 1, 2, or 3 and so on. If a match is not found then it falls to the default and carries out that instruction. You need to remember that you cannot use any relational operators with the case statement such as case >3. What happens if you do not include the break statements? Every case statement will be executed until it finds the break statement or until the end of the block. Try deleting the break statements and see what happens.
Here is a complete example of a program. Let us write a program to recieve three grades, find its average, determine the letter grade and write a brief comment about the grade.

Program Three-3.

```
/*************************************
Accept three grades, find the average
   and display Passed or Failed.
Teach objective  - making decisions
By Dr. John Abraham
Created for 1380 students
*************************************/

#include <iostream>
#include <iomanip>  //to format input and output. Here setw and endl require it
#include <fstream>
using namespace std;
char getgrade (double);//to find letter grade
void Message (int, int, int, double, char);
void printToFile(int, int, int, double, char);

int main ()
{
 int one, two, three;
 double average;
 char grade;


       cout << "Enter three grades ";
       cin >> one >> two >> three;
       average= (one+ two+ three)/3.0; // the total of three grades are first converted
to float
       grade = getgrade(average);
       Message (one,two,three,average,grade);

       printToFile(one,two,three, average, grade);
       cin.ignore();
       getchar();

 return (0);
```

```
}

char getgrade (double average)
 {
   if (average >=90)return ('A');
   else if (average>= 80) return('B');
        else if (average >=70) return ('C');
             else if (average >= 60) return ('D');
                  else return('F');
 }
void Message (int one, int two, int three, double average, char grade)
    {
      cout << "Three grades are: "  <<one <<setw(4)<<two <<setw(4)<<three <<endl;
    cout <<fixed<<showpoint<<setprecision(2);
      cout << "The average is :  " <<average <<endl;
      cout << "\nThe letter grade is : "<< grade << endl;


      switch (grade)
      {
      case 'A' : cout << "Very impressive grade indeed!\n";break;
      case 'B' : cout << "A solid performance, congratulations!\n"; break;
      case 'C' : cout << "C++ is a tough course, but YOU MADE IT!\n";break;
      case 'D' : cout << "Made it eh? \n";break;
      case 'F' : cout << "Don't give up. Try keeping up with all the homework!\n";
      }

 }


void printToFile(int one, int two, int three, double average, char grade)
{
      ofstream outfile;
      outfile.open("ifThenElse.txt");

      outfile << "Three grades are: "  <<one <<setw(4)<<two <<setw(4)<<three <<endl;
      outfile <<fixed<<showpoint<<setprecision(2);
        outfile << "The average is :  " <<average <<endl;
        outfile << "The letter grade is : "<< grade << "\n";
switch (grade)
      {
      case 'A' : outfile << "Very impressive grade indeed!\n";break;
      case 'B' : outfile << "A solid performance, congratulations!\n"; break;
      case 'C' : outfile << "C++ is a tough course, but YOU MADE IT!\n";break;
      case 'D' : outfile << "Made it eh? \n";break;
      case 'F' : outfile << "Don't give up. Try keeping up with all the homework!\n";
      }

       outfile.close();
}
```

Program Run Three-3

Enter three grades 80 85 99
Three grades are: 80 85 99
The average is : 88.00

The letter grade is : B
A solid performance, congratulations!

Assignment

Write a program to display the name of the month, given its number. For example if you enter 4 for the month, it should display April. Write it using if/else and then modify it to use the case statement.

Explain purpose the else statement in the above program? What happens if you do not use else.