Decision Making

Often it is required to decide whether a certain block of code be executed or not, based on some predefined conditions. This decision making requires the use of relational, equality and logical operators. Let us first introduce these operators.

Relational & Equality Operators

Relational operators and equality operators are used to compare numerical data. Both the equality and relational operators check for a particular relationship between two numeric values and results if the relationship holds or not if the relationship holds, then it returns a Boolean true (1) otherwise it returns a Boolean false (0).

Here is a list of Relational and Equality Operators

Equality Operators			
S. No	Operator	Name	Description and Example
1	==	Equal	This operator tests if the given operands are equal. In case if they are equal it yields a Boolean true, otherwise false `2 == 5;` This will be false because 2 is not equal to 5
2	!=	Not Equal	This operator tests if the given operands are not equal. In case if they are not equal it yields a Boolean true, otherwise false `2 != 5;` This will be true because 2 is not equal to 5
Relational Operators			
3	<	Less than	This operator tests if the first operand is less than the second operand. If so it yields a Boolean true, otherwise false `2 < 5;` This will be true because 2 is less than 5
4	>	Greater than	This operator tests if the first operand is greater than the second operand. If so it yields a Boolean true, otherwise false `2 > 5;` This will be false because 2 is not greater than 5
5	<=	Less than or equal to	This operator tests if the first operand is less than or equal to the second operand. If so it yields a Boolean true, otherwise false `2 <= 5;` This will be true because 2 is less than 5
6	>=	Greater than or equal to	This operator tests if the first operand is greater than or equal to the second operand. If so it yields a Boolean true, otherwise false `2 >= 5;` This will be false because 2 is not greater than 5

les_04_code_01.cpp

```
1. #include<iostream>

2. using namespace std;

3. int main ()

4. {
5.        int num1 = 10, num2 = 5, num3 = -9, num4 = 5;

6.        cout<<(num1>num2)<<endl;
7.        cout<<(num2<num3)<<endl;
8.        cout<<(num2>=num4)<<endl;
9.        cout<<(num3<=num4)<<endl;
10.       cout<<(num2==num4)<<endl;
11.       cout<<(num1!=num2)<<endl;


12.      return 0;
13.      }
```

Logical Operators

In cases where more than one relational and equality operators simultaneously decide the final result, logical operators help in getting the simultaneous relation. Logical operators work on Binary Operands, their result depends on the combined states of operands.

There are two logical operators (of our interest right now).

i. OR (||)
ii. AND (&&)

Logical operators work on Binary Operands, their result depends on the combined states of operands. For the two operands x and y (both Boolean), the OR (||) operator works according to the following truth table.

x	y	x||y
0	0	0
0	1	1
1	0	1
1	1	1

AND operator (&&) works according to the following truth table.

x	y	x&&y
0	0	0
0	1	0
1	0	0
1	1	1

Where 0 is false and 1 is true

les_04_code_02.cpp

```
1. #include<iostream>

2. using namespace std;

3. int main ()

4.       {
5.            int num1 = 10, num2 = 5, num3 = -9, num4 = 5;

6.            bool res1, res2, res3, res4;

7.            res1 = (num1<num2)&&(num2>num3);
8.            cout<<"\t(num1<num2)&&(num2>num3)\t"<<res1;

9.            res2 = (num1<num2)||(num2>num3);
10.           cout<<"\n\t(num1<num2)||(num2>num3)\t"<<res2;

11.           res3 = (num1>=num3)&&(num2<=num4);
12.           cout<<"\n\t(num1>=num3)&&(num2<=num4)\t"<<res3;


13.      return 0;
14.      }
```

```
(num1<num2)&&(num2>num3)        0
(num1<num2)||(num2>num3)        1
(num1>=num3)&&(num2<=num4)      1
```

At this point it is important to discuss the associativity and precedence order of these new operators introduced. The following screenshot from textbook page 55 will help you to understand. Precedence decrease from top to bottom.

Operators				Associativity	Type
()				*[See caution in Fig. 2.10]*	grouping parentheses
*	/	%		left to right	multiplicative
+	-			left to right	additive
<<	>>			left to right	stream insertion/extraction
<	<=	>	>=	left to right	relational
==	!=			left to right	equality
=				right to left	assignment

For more details please read section 2.7 of textbook (Dietel & Dietel)

`if` statement

`if` statement is used to execute a certain block of a program, based on predefined decision.

```
------------------------------syntax----------------------------------------

if (relational expression)
{

//body of if statement

statement to be executed if relational expression is true

}

--------------------------------------------------------------------------------
```

les_04_code_03.cpp

```
1. #include<iostream>

2. using namespace std;

3. int main()

4. {
5. int hoursWorked;
6. double rate, grossPay;
7. cout << "Enter the hours worked: ";
8. cin >> hoursWorked;
9. cout << "Enter the rate of pay: ";
```

```
10.      cin >> rate;
11.      if (hoursWorked <= 40)
12.      {
13.      grossPay = hoursWorked * rate;
14.      }
15.      if (hoursWorked > 40)
16.      {
17.      grossPay = (40 * rate) + ((hoursWorked-40) * (rate * 1.5));
18.      }
19.      cout << "Gross pay is: " << grossPay << endl;
20.      return 0;

21.      }
```

Output Run 1

```
Enter the hours worked: 27
Enter the rate of pay: 100
Gross pay is: 2700
```

Output Run 2

```
Enter the hours worked: 135
Enter the rate of pay: 100
Gross pay is: 18250
```

`if else` statement

if else statement is used to choose between certain blocks based on the relational expression if it is true or false. Both have a separate block and only one of them will be executed.

Note : There will be only one relational expression for if else

```
------------------------------syntax----------------------------------------

if (relational expression)
{
//body of if statement
statements to be executed if relational expression is true
}
else
{
//body of else statement
statements to be executed if relational expression is false
}

--------------------------------------------------------------------------------
```

les_04_code_04.cpp

```
1. #include<iostream>

2. using namespace std;

3. int main()

4. {
5.       int hoursWorked;
6.       double rate, grossPay;

7.       cout << "Enter the hours worked: ";
8.       cin >> hoursWorked;
9.       cout << "Enter the rate of pay: ";
10.      cin >> rate;

11.      if (hoursWorked <= 40)
12.      {
13.      grossPay = hoursWorked * rate;
14.      }

15.      else
16.      {
17.      grossPay = (40 * rate) + ((hoursWorked-40) * (rate * 1.5));
18.      }

19.      cout << "Gross pay is: " << grossPay << endl;
20.      return 0;

21.      }
```

Nested `if else`

les_04_code_05.cpp

```
1. #include<iostream>
2. using  namespace std;

3. int main()
4.       {
5.       int age;
6.       cout<<"Please enter your age : ";
7.       cin>>age;

8.       if(age<18)
9.       {
10.         cout<<"You can't vote."<<endl;
11.         if (age>=16)
12.            {
```

```
13.             cout<<"But you can drive."<<endl;
14.             }
   15.   else
16.      {
17.      cout<<"You can't drive either."<<endl;
18.      }

19.      }

20.      else

21.      {
22.      cout<<"You can vote."<<endl;
23.      cout<<"You can drive."<<endl;
24.      if(age>=21)
25.          {
26.          cout<<"You can also participate in elections."<<endl;
27.          }
28.      else
29.          {
30.          cout<<"But you can't participate in elections."<<endl;
31.          }

32.      }
33.      return 0;
34.      }
```

Output Run 1

```
Please enter your age : 15
You can't vote.
You can't drive either.
```

Output Run 2

```
Please enter your age : 17
You can't vote.
But you can drive.
```

Output Run 3

```
Please enter your age : 19
You can vote.
You can drive.
But you can't participate in elections.
```

Output Run 4

```
Please enter your age : 27
You can vote.
You can drive.
You can also participate in elections.
```

`if else if` construct

Sometimes it is required to decide among several blocks based on certain conditions e.g. To decide the grade of a student based on marks obtained. This can be achieved by the nested if - else construct. But it is not a good programming practice. Better way is to use if else if construct.

SYNTAX

```
if (test condition 1)
{
body of if
}

else if (test condition 2)
{
body of else if
}

else if (test condition 3)
{
body of else if
}
.
.
.
.

else if (test condition n)
{
body of else if
```

```
}

else
{
body of else // this is optional and will only be executed if none of the
test
             // conditions are true
}
```

EXAMPLE

Let's implement the following example
Based on these given conditions, our program will decide the letter grade of a student provided his/her marks

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^GradingScale^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

90 - 100	A
80 - 89	B
70 - 79	C
60 - 69	D
00 - 59	F

les_04_code_06.cpp

```
1. #include<iostream>

2. using namespace std;

3. int main ()

4. {
5. // suitable variables declaration
6. int marks;
7. string letterGrade;

8. //data input phase
9. cout<< "Enter Marks Obtained : ";
10.      cin>>marks;

11.      // now decision making for multiple selection

12.      if(marks>=90)
13.      {
14.      letterGrade = "A";
15.      }

16.      else if (marks >= 80)
17.      {
```

```
18.      letterGrade = "B";
19.      }

20.      else if (marks >= 70)
21.      {
22.      letterGrade = "C";
23.      }

24.      else if (marks >= 60)
25.      {
26.      letterGrade = "D";
27.      }
28.      else
29.      {
30.      letterGrade = "F";
31.      }

32.      cout<<"For a score of "<<marks<<" marks letter grade is
   "<<letterGrade;
33.      return 0;
34.      }
```

Using Logical Operators

EXAMPLE

Let's implement the following example
Based on these given conditions, our program will decide the letter grade of a student
provided his/her marks

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Grading Scale^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

97 - 100	A+
93 - 96	A
90 - 92	A-
87 - 89	B+
83 - 86	B
80 - 82	B-
77 - 79	C+
73 - 76	C
70 - 72	C-
67 - 69	D+
63 - 66	D
60 - 62	D-
00 - 59	F

Note this distribution can be programmed similarly like les_04_code_06.cpp
But for learning purpose we are implementing with the help of
logical operator

---

les_04_code_07.cpp

```
1. #include<iostream>

2. using namespace std;

3. int main ()

4. {
5. // suitable variables declaration
6. int marks;
7. string letterGrade;

8. //data input phase
9. cout<< "Enter Marks Obtained : ";
10.      cin>>marks;

11.      // now decision making for multiple selection

12.      if((marks>=97)&&(marks<=100))
13.      {
14.      letterGrade = "A+";
15.      }
16.      else if ((marks >= 93)&&(marks <= 96))
17.      {
18.      letterGrade = "A";
19.      }
20.      else if ((marks >= 90)&&(marks <= 92))
21.      {
22.      letterGrade = "A-";
23.      }
24.      else if((marks>=87)&&(marks<=89))
25.      {
26.      letterGrade = "B+";
27.      }
28.      else if ((marks >= 83)&&(marks <= 86))
29.      {
30.      letterGrade = "B";
31.      }
32.      else if ((marks >= 80)&&(marks <= 82))
33.      {
34.      letterGrade = "B-";
35.      }
```

```
36.      else if((marks>=77)&&(marks<=79))
37.      {
38.      letterGrade = "C+";
39.      }
40.      else if ((marks >= 73)&&(marks <= 76))
41.      {
42.      letterGrade = "C";
43.      }
44.      else if ((marks >= 70)&&(marks <= 72))
45.      {
46.      letterGrade = "C-";
47.      }
48.      else if((marks>=67)&&(marks<=69))
49.      {
50.      letterGrade = "D+";
51.      }
52.      else if ((marks >= 63)&&(marks <= 66))
53.      {
54.      letterGrade = "D";
55.      }
56.      else if ((marks >= 60)&&(marks <= 62))
57.      {
58.      letterGrade = "D-";
59.      }
60.      else
61.      {
62.      letterGrade = "F";
63.      }


64.      cout<<"For a score of "<<marks<<" marks letter grade is
   "<<letterGrade;
65.      return 0;
66.      }
```

Task

In the lesson_04 folder, there is a file game.exe, run this file and observe how it works (the correct answer is Watson). Your assignment is to write a code to replicate the same program.

Bonus Program

les_04_code_08.cpp

```
1. #include <iostream>

2. using namespace std;

3. int main()
4. {
```

```
5. double operand1, operand2, result;
6. char op;
7. cout << "Enter an expression (operand1 operator operand2): ";
8. cin >> operand1 >> op >> operand2;

9. if (op == '*')
10.      {
11.      result = operand1 * operand2;
12.      cout << "result: " << result << endl;
13.      }
14.      else if (op == '/')
15.      {
16.      result = operand1 / operand2;
17.      cout << "result: " << result << endl;
18.      }
19.      else if (op == '+')
20.      {
21.      result = operand1 + operand2;
22.      cout << "result: " << result << endl;
23.      }
24.      else if (op == '-')
25.      {
26.      result = operand1 - operand2;
27.      cout << "result: " << result << endl;
28.      }
29.      else
30.      cout << "Bad expression." << endl;



31.      return 0;
32.      }
```