Boolean Expressions

Sometimes a programmer would like one statement, or group of statements to execute only if certain conditions are true. There may be a different statement, or group of statements that are to be executed when the condition is false. An expression that can be evaluated to true or false is called a Boolean expression. *(Named after a mathematician named Boole.)*

In C++, the number 0 (zero) is considered to be false, all other numbers are true.

The Boolean operators

The Boolean operators are a bit different than the ones used in algebra because we have to be able to type them.

`==` equal ***(Remember, a single = is used to assign a value!)***
`<` less than
`>` greater than
`>=` greater than or equal to
`<=` less than or equal to
`!=` not equal

Boolean Expressions

(`x`, `y`, and `z` are variables with the values shown below)

```
x = 0;    y=5;    z=8;
```

Each expression below is **TRUE**:

`x < y` 0 is less than 5
`z > y` 8 is greater than 5
`y < z` 5 is less than 8
`y <= z` 5 is less than or equal to 8
`x != z` 0 is not equal to 8
`z >= y` 8 is greater than or equal to 5

In C and C++, 0 (zero) is considered false; everything else is true. Therefore, a variable all by itself is also a Boolean expression. For the variables shown above `x` is false, `y` is true, and `z` is true.

Continuing with the same values for x, y, and z, each expression below is **FALSE**:

`x > y` 0 is NOT greater than 5
`z < y` 8 is NOT less than 5
`y == z` 5 is NOT equal to 8
`y <= z` 5 is less than or equal to 8
`y != y` 8 is not equal to 8 is false
`x != x` 0 is not equal to 0 is also false

The if Statement

The **`if`** statement is used to execute a statement only under certain conditions.

The format is: **`if (`**_____**`)`** _____**`;`**

Boolean expression *Statement to execute if True*

Another way to show this format is:

`if (<`*`Boolean exp.`***`>) <`***`true statement`***` >;`**

else: To execute a statement when the Boolean expression is False, use **else**. The format is:

`if (<`*`Boolean exp.`***`>) <`***`true statement`***`>; else <`***`false statement`***`>;`**

(This can be on one or more lines.)

Example: the program below asks the user for a number. It prints out whether the number is true or false.

```
 0 // test if
 1 #include <iostream.h>
 2 int main()
 3 { double num;  // this will also work with int
 4   cout<<"Enter number to test:";
 5   cin>>num;
 6   if (num)   // 0 is false all other numbers are true
 7     cout<<num<<" is true\n";
 8   else
 9     cout<<num<<" is false\n";
10   return 0;
11 } //main
```

Sample Output:

```
Enter number to test:5
5.0 is true

Enter number to test:0
0.0 is false

Enter number to test:-6.3
-6.300000 is true
```

Use = = not =

Every statement in C++ returns a value. The statement **`x = 5;`** returns the last value assigned: a 5. You can take advantage of this in an assignment statement to give several variables the same value. Example: The statement **`x=y=0;`** assigns a value of 0 to y first and returns that value to be assigned to x. The statement **`if (x = 5) cout<<x;`** assigns a value of 5 to x first, then tests if 5 is true, it is. The value **`5`** will print every time, regardless of the original value of x.!

Blocks

To execute more than one statement for the true or false code, a pair of braces encloses the block. It is a good practice to label each **}** with a comment and use indentation for the statements inside the block. Anywhere one statement is allowed, a *block* can be used to group statements. A block will also be used for statements within a loop, and other situations.

The next program asks the user for two numbers, then prints them in order. Blocks are used to execute two statements when true and two statements when false:

```
 0 // Input two numbers print out in order
 1 #include <iostream.h>
 2 int main()
 3 { int num1,num2;
 4   cout<<"Enter a number:";
 5   cin>>num1;
 6   cout<<"Enter another number:";
 7   cin>>num2;
 8   if (num1 < num2)
 9   { cout<<"The smaller number is "<<num1<<"\n";
10     cout<<"The larger number is "<<num2<<"\n";
11   } // num1 < num2 is true
12   else
13   { cout<<"The smaller number is "<<num2<<"\n";
14     cout<<"The larger number is "<<num1<<"\n";
15   } // num2 < num1 is true
16   return 0;
17 } //main
```

Please note the indentation in the program above. A good programming practice is to indent all of the statements within the same set of braces the same amount. This will make your program much clearer, and easier to read, debug, and modify. About 2 spaces is plenty.

Sample Output:

This example executed the true block:

```
Enter a number:5
Enter another number:6
The smaller number is 5
The larger number is 6
```

This example executed the false block:

```
Enter a number:12
Enter another number:3
The smaller number is 3
The larger number is 12
```

What happens if you enter a double instead of an integer? Don't be afraid to experiment. Save your program before you run it. If you are working in Windows, save work in other applications before writing C programs in case the system crashes.

This example enters a double as the second number:

```
Enter a number:4
Enter another number:1.8
The smaller number is 1
The larger number is 4
```

This example enters a double as the first number:

```
Enter a number:3.8
Enter another number:The smaller number is 0
The larger number is 3
```

Explanation: When an integer is read in, the processor stops when it reads a decimal or anything that is not part of the integer. In the first example, the .8 was left in the input stream and never used. In the second example, the 3 was read in as the first number. The .8 was left in the input stream. When the `scanf` for the second number was executed, the .8 was in the buffer, so the processor did not have to wait for the user to enter a number. Because the user did not press enter, the output is on the same line.

This example enters two numbers with a space between them as the first number:

```
Enter a number:3 5
Enter another number:The smaller number is 3
The larger number is 5
```

Exchange 2 Numbers

Another way to print the numbers in order is to read them in, then rearrange them so that num1 is less than or equal to num2. Print num1 and num2 after they are in order.

```
 0 // Input two numbers, exchange so that num1 <= num2
 1 #include <iostream.h>
 2 int main()
 3 { int num1,num2, temp;
 4   cout<<"Enter a number:";
 5   cin>>num1;
 6   cout<<"Enter another number:";
 7   cin>>num2;
 8   if (num1 > num2)  // exchange the values
 9   { temp = num1;
10     num1 = num2;
11     num2 = temp;
12   } // exchange
13   cout<<"The smaller number is "<<num1<<"\n";
14   cout<<"The larger number is "<<num2<<"\n";
15   return 0;
16 } //main
```

3: In order for the exchange to work a third variable is needed. The statements num1=num2; num2=num1; won't work. Try it, afterwards both numbers will be the same. Which of the original numbers will they both equal?

8: The numbers only need to be exchanged if num1 is greater than num2.

9-11: The illustration below shows the exchange:

		num1	num2	temp
	sample values at start	5	4	?
9:	`temp = num1;`	5	4	5
10:	`num1 = num2;`	4	4	5
11:	`num2 = temp;`	4	5	5

13,14: At this point the numbers were in order to start with or have been exchanged.

Else IF

The program below uses a nested **else if** statement. This program asks the user his grade, then prints the letter grade.

```
 0 // Prints the letter grade
 1 #include <iostream.h>
 2 int main()
 3 { int grade;
 4   cout<<"Enter your numeric grade:";
 5   cin>>grade;
 6   if (grade < 65)
 7     cout<<grade<<" is F\n";
 8   else if (grade < 70)
 9     cout<<grade<<" is D\n";
10   else if (grade < 80)
11     cout<<grade<<" is C\n";
12   else if (grade < 90)
13     cout<<grade<<" is B\n";
14   else
15     cout<<grade<<" is A\n";
16   return 0;
17 } //main
```

Note the indentation: Usually it is best to put every else directly underneath it, but when there are nested if-else statements this would create a diagonal line going off the page. The method shown above is an accepted way to indent nested if-else statements.

Explanation:

14: There is no need to test if this is an A, is the program is at this point, all of the other conditions have failed.

Sample Output:

```
Enter your numeric grade:80
80 is B
```

Testing: A programmer should test every possible statement in a program**.** Run this program several times. Test for less than 65, exactly 65, between 65 and 70, exactly 70, and so on. Test it with the attitude that it was written by your worst enemy and you are determined to prove that it doesn't work!

The program below does exactly the same thing, but stores the grade in a character and prints at the end instead of printing immediately. This is a better method because it is easier to change the format of the output if it is just one statement.

```
 0 // Prints the letter grade
 1 #include <iostream.h>
 2 int main()
 3 { int grade;
 4   char letter; // the grade will be stored, printed at end
 5   cout<<"Enter your numeric grade:";
 6   cin>>grade;
 7   if (grade < 65) letter = 'F';
 8   else if (grade < 70) letter = 'D';
 9   else if (grade < 80) letter = 'C';
10   else if (grade < 90) letter = 'B';
11   else letter = 'A';
12   cout<<grade<<" is "<<letter<<"\n"; // print at end
13   return 0;
14 } //main
```

Note: Single quotes are used to surround a character, double quotes are used for strings. (Memory trick: A character is a *single* letter use a *single* quote.)

Exercise: Rewrite this program to test for an "A" first, then on down to "F" last.

Summary of IF Statements

`If` statements can have any of the formats below:

- One statement to execute when True:

```
if (<Boolean expression>) <statement>;
```

- One statement to execute when True, one when false:

```
if (<Boolean expression>) <statement> else <statement>;
```

- Using blocks: one or more statements to execute when true, one or more when false

```
if (<Boolean expression>)
{
   <statements>;
} // end of true block
else
{
   <statements>;
} // end of false block
```

- If Else Group: more than one condition to test

```
if (<Boolean expression1>)
{  <statements>
} // end of block for first condition
else if <Boolean expression2>
{  <statements>
} // end of block for second condition
// as many else if blocks as needed
else // block to execute if none of the other conditions is met
{
   <statements>  {the else block is optional}
} // end of block for last condition
```

Multiple Conditions: && and ||

A Boolean expression can test more than one condition using && for `AND, || for OR:`

```
if (<Boolean expression1> || <Boolean expression2>) <statement>
```

The <*statement*> is executed if expression1, expression2, or <u>both</u> are true.

```
if (<Boolean expression1> && <Boolean expression2>) <statement>
```

The <*statement*> is executed if <u>both</u> expression1 and expression2 are true.

Note: Each side of the **&&** or || must be a complete Boolean expression. The statement

```
if (x < 5 || > 100)...
```

is not allowed: it must be written

```
if (x < 5 || X > 100)...
```

The next program asks the user for the number of a month, then prints a message telling if the number is a valid month (between 1 and 12).

```
 0 // Input a valid month
 1 #include <iostream.h>
 2 int main()
 3 { int month;
 4   cout<<"Enter the number of a month:";
 5   cin>>month;
 6   if (month >=1 && month <=12)
 7     cout<<month<<" is a valid month.\n";
 8   else
 9     cout<<month<<" is NOT a valid month.\n";
10   return 0;
11 } // main
```

The next program does exactly the same thing as the program above, except that it checks for an invalid month using || instead of checking for a valid month with &&.

```
 0 // Input a valid month
 1 #include <iostream.h>
 2 int main()
 3 { int month;
 4   cout<<"Enter the number of a month:";
 5   cin>>month;
 6   if (month < 1 || month > 12)
 7     cout<<month<<" is NOT a valid month.\n";
 8   else
 9     cout<<month<<" is a valid month.\n";
10   return 0;
11 } // main
```

Rats in Mazes

In the diagrams below, the letters represent gates that can be open (1 is true) or closed (0 is false). The goal of the program is to determine if the rat escapes or is locked up.
Note: The rat is brilliant! If there is a way to escape she will find it!

Maze 1: The program to determine if the rat can escape from maze 1 is shown below:

```
 0 // Determine if rat escapes from maze
 1 #include <iostream.h>
 2 int main()
 3 { char answer;
 4   int A,B;
 5   cout<<"Is door A open? Answer Y or N:";
 6   cin>>answer;
 7   if (answer == 'Y' || answer == 'y')
 8      A = 1; // door A open
 9   else
10      A = 0; // door A closed
11   cout<<"Is door B open? Answer Y or N:";
12   cin>>answer;
13   if (answer == 'Y' || answer == 'y')
14      B = 1; // door B open
15   else
16      B = 0; // door B closed
17   if (A || B)  // either door A or Door B is open
18     cout<<"The rat can escape.\n";
19   else
20     cout<<"The rat can NOT escape.\n";
21   return 0;
22 } // main
```

Explanation:

7: Check if the user answered either uppercase or lowercase Y.

Maze 2: The program for maze 2 would change the **if** statement in line 14 as shown below:


```
if (A && B)  // both door A and Door B are open
```

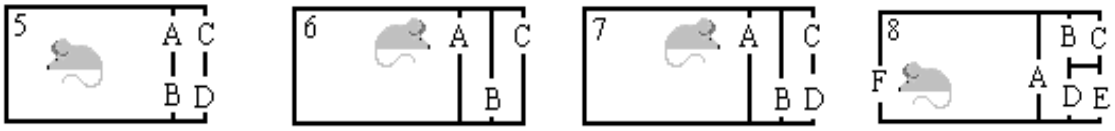
Maze 3: When both && and || appear in an expression, && is done first unless there are parenthesis to change to order.

The program for maze 3 must add the lines to read in whether door C is open or not (including reading the line feed after reading the answer for door B. The if statement in line 14 would be changed as shown below:

```
if ((A || B) && C)  // both door A and Door B are open
```

The statement `if (A || B && C)` would imply maze 4 as shown on the right:

Exercise: Write the **if** statement for each maze shown below. (Assume all of the doors have been set to 0 or 1.)



Word Problems: Write C++ programs for each problem below. Use the variable names in bold. Use appropriate prompts for inputting each value.

1. A person pays half **price** if their **age** is less than 12 or is 65 or more. Change the value of **price** accordingly.

2. A salesman receives a **bonus** if they had **sales** of at least $1000 or has at least 50 new **customers**. The salesman also gets the bonus if they had at least $800 in **sales** with 25 new **customers** or more. Assign a value to bonus: 0 (false) or 1 (true).

3. The **tuition** is $50 a **credit** if the student is a **resident**, or in the **military**, otherwise **tuition** is $120 per credit. Calculate tuition.

4. A typist is **level** 1 if they can not type fewer than 30 **wpm** (words per minute); level 2 if they type between 30 and 50 **wpm**; level 3 if they type more than 50 **wpm**.

5. A package is **accepted** if its **weight** is between 10 and 11 ounces. Assign a value to **accepted**: 0 (false) or 1 (true).

Switch

The `switch` construct offers an alternative to nested if statements. Switch can be used when a variable has a small number of possible values.

The sample program below asks the user for the number of a month, then prints the season.

```
 0 // Input a month, print the season
 1 #include <iostream.h>
 2 int main()
 3 { int month;
 4   cout<<"Enter the number of a month:";
 5   cin>>month;
 6   switch (month)
 7   { case 1: case 2: case 12:  cout<<"Winter\n"; break;
 8     case 3: case 4: case 5:   cout<<"Spring\n"; break;
 9     case 6: case 7: case 8:   cout<<"Summer\n"; break;
10     case 9: case 10: case 11: cout<<"Fall\n";   break;
11     default: cout<<month<<" is not a valid month\n";
12   } // switch month
13   return 0;
14 } // main
```

Explanation:

6: The variable to test is in parenthesis after the word **`switch`**.

7: The value of the variable is compared to each **`case`**. When a match is found, statements are executed until a **`break`** is encountered. If no case matches, the statements after the word **`default`** is executed.

8: C++ does not permit an ellipsis: **`3..5`** is not allowed.

12: The switch block requires braces at the beginning and end.

Sample output:

```
Enter the number of the month:5
Spring

Enter the number of the month:13
13 is not a valid month
```

A simple calculator

The sample program below allows the user to enter an equation such as 12*5 or 28/6 and calculates the result. Special care must be taken to avoid an attempt to divide by zero.

```
 0 // a simple calculator
 1 int main()
 2 { int x,y;
 3   char op;
 4   cout<<"Enter an equation (Example 12*5):";
 5   cin>>x>>op>>y;   //use cin to read all 3 variables
 6   switch (op)
 7   { case '+': cout<<x+y; break;
 8     case '-': cout<<x-y; break;
 9     case '*': cout<<x*y; break;
10     case '/': // don't try to divide by zero!!!
11       if (y==0) cout<<"Divide by zero not allowed.\n";
12       else cout<<x/(y*1.0); //force result as double
13       break;
14     case '%': cout<<x%y; break;
15     default: cout<<op<<" is not a valid operator.\n";
16   } //switch op
17   return 0;
18 } // main
```

Explanation:

6: The cin command allows more than one variable to be read in.

11: Dividing by zero is not allowed. We want to avoid errors from occurring that cause the program to crash.

12: The expression **`x/y`** will return an integer. It is forced to return a double by multiplying one of the operands by 1.0, which simply changes the operand from an integer to a double without changing the value.

Sample output:

```
Enter an equation (Example 12*5):3+21
24

Enter an equation (Example 12*5):23%7
2

Enter an equation (Example 12*5):12/5
2.400000

Enter an equation (Example 12*5):4x7
x is not a valid operator.
```

The program below is a nonsense program to illustrate what happens when the break statement is missing. The user enters a character. The switch is based on the character entered.

```
 0 // a nonsense program to illustrate break
 1 #include <iostream.h>
 2 int main()
 3 { char alpha;
 4   cout<<"Enter a letter from a to c: ";
 5   cin>>alpha;
 6   switch (alpha)
 7   { case 'A':
 8     case 'a': cout<<"A for angst\n"; break;
 9     case 'b':
10     case 'B': cout<<"B for break that is missing\n";
11     case 'c':
12     case 'C': cout<<"C for computer\n"; break;
13     default:  cout<<"No, no, no! Just A, B or C!\n";
14   } //switch alpha
15   return 0;
16 } // main
```

Explanation:

7: There is no statement after case ‘A’, however once this match is made statements are executed until a break. The `cout` statement after case ‘a’ executes for both ‘a’ and ‘A’.

9: When a ‘b’ is entered all of the statements from 9 to 12 are executed because there is no break for the letter B.

13: This statement is executed if the letter entered is not an A, a, B, b, C or c.

Sample output:

```
Enter a letter from a to c: x
No, no, no! Just A, B or C!\

Enter a letter from a to c: A
A for angst

Enter a letter from a to c: b
B for break that is missing
C for computer

Enter a letter from a to c: C
C for computer
```

Speedy or Reliable?

Two shipping companies are in fierce competition against each other. Their most recent ads are shown below. The shipping department needs a program that asks the weight of the package and tells which company to ship with. The program must be easy to modify if one of the companies offers new rates.

Speedy We get it there fast!
Next day:
$10 for any package up to 30 pounds.
$20 for any package up to 70 pounds.
No next day on packages over 70 pounds.
Whenever: $.50 per pound, any size

Reliable: You can depend on us!
Overnight:
$7 plus 20 cents per pound.

No Hurry:
$5 plus 10 cents per pound.

```
 0 // Decide which company to ship with
 1 #include <iostream.h>
 2 #include <iomanip.h>
 3 int main()
 4 { int pounds;
 5   double speedy, reliable;
 6   char rush;
 7   cout<<"Send overnight? Y or N:";
 8   cin>>rush;
 9   cout<<"Enter the weight of package in pounds:";
10   cin>>pounds;
11   // calculate speedy cost: rates eff. 5/22/96
12   if (rush == 'Y' || rush == 'y')
13   {  if (pounds <= 30) speedy = 10.00;
14      else if (pounds <= 70) speedy = 20.00;
15      else speedy = 0.00; // signal not accepted
16   } // calculate next day for speedy
17   else speedy = 0.50 * pounds;
18
19   // calculate reliable: rates eff. 5/14/96
20   if (rush == 'Y' || rush == 'y')
```

```
21      reliable = 7.00 + 0.20 * pounds;
22    else
23      reliable = 5.00 + 0.10 * pounds;
24
25    // print who to ship with
26    cout<<setprecision(2);  //print  dollar  with  2  decimal
27  places
28    if (!speedy && reliable)
29      cout<<"Ship with Reliable for $"<<reliable;
30    else if (speedy == reliable)
31      cout<<"Ship with either for $"<<reliable;
32    else if (speedy < reliable)
33      cout<<"Ship with Speedy for $"<<speedy;
34    else
35      cout<<"Ship with Reliable for $"<<reliable;
36    return 0;
  } //main
```

Programming Exercises:

1. A store is having a sale. Red tags are half price. Blue tags are 30% off. Green tags are 25% off. Ask the user the original price and the first letter of the tag color. (Use N for no tag.) Print out the sale price.
2. Compute and print regular pay, overtime pay and gross pay. Ask the hourly rate and the number of hours worked for one week. Employees get "time and a half" for overtime. Overtime is anything over 40 hours.
3. A new company is competing with Speedy and Reliable. Modify the program to consider the new shipping company.

Express We handle with care!
Next day:
$15 for any package up to 50 pounds.
Over 50 pounds, add 50 cents for each pound over 50.
No Hurry: $5 for any package up to 50 pounds.
Over 50 pounds, add 20 cents for each pound over 50.