Lab 3: Basic Control Flow

Your name:	Zach Evans
Your email address:	evansz2@students.wwu.edu
Your student ID number:	W01006113

Lab Objectives

To gain experience in

- implementing simple decisions
- writing expressions with relational operators
- input validation
- simple loops
- using Boolean variables

R1. The `if` statement

In C++, decisions to execute or not execute a statement or group of statements can be made by using the statement `if (`*`condition`*`)`. The *`condition`* must be an expression that can be evaluated as either `true` or `false`. If it evaluates to `true`, a succeeding statement or group of statements enclosed in `{ . . . }`, called a *block statement* will be executed.

`How much money will be added to your account when the initial starting amount is $100`?

$115

`How much money will be added to your account when the initial starting amount is $85?` $90

```
#include <iostream>

int main()
{

  double initial_value, final_value;

  cout <<"Please enter the initial balance: ";

  cin >> initial_value;

    if (inital_value > 90 )
    {

      final_value = final_value + 10;

    }

    final_value = final_value + 5;
    cout <<"Final Value" <<final_value;
    return 0;
}
```

R2. Relations and Relational Operators

The relational operators in C++ are `== != < > <= and >=`

- Formulate the following conditions in C++:

x is positive	x > 0
x is zero or negative	x <= 0
x is at least 8	x >= 8
x is less than 8	x < 8
x and y are both zero	x == 0 && y == 0

- Formulate the following conditions where `name` is a `string` and `salary` is a `double`.

`name` starts with the letter H	name(0) == 'H'
`salary` is at least $75,000	salary >= 75000

- Formulate the following conditions on the variables `day`, `month`, and `year`, of type `int` which represent Harry's birth date.

Harry is at least 21 years old.	thisYear - year >= 21
Harry was born in August.	month == 7 (if Jan == 0)
Harry was born in a leap year (assume his birthday was between 1901 and 2099).	(year % 4) == 0

P1. Input validation

- This exercise consists of three files: ccc_shap.h, ccc_shap.cpp (links on moodle) and the code below. Put all three files in a common directory. To build and run the following program simply compile the file below. Describe what happens when the two points have the same x coordinate?

```
The slope is printed as "inf".
```

```
#include <iostream>
using namespace std;
#include "ccc_shap.cpp"
main()
{  double slope;

   double xcoord, ycoord;
   Point p1, p2;

   cout << "Input x coordinate of the first point" << "\n";
   cin >> xcoord;
   cout << "Input y coordinate of the first point" << "\n";
   cin >> ycoord;

   p1 = Point(xcoord, ycoord);

   cout << "Input x coordinate of the second point" << "\n";
   cin >> xcoord;
   cout << "Input y coordinate of the second point" << "\n";
   cin >> ycoord;

   p2 = Point(xcoord, ycoord);

   slope = (p2.get_y() - p1.get_y()) / (p2.get_x() - p1.get_x() );

   cout << "The slope of the line between Points 1 and 2 is " << slope << "\n";
   return 0;
}
```

- Correct and rebuild the program to disallow a vertical line (denominator = 0).

```
#include <iostream>
using namespace std;
#include "ccc_shap.cpp"
int main()
{  double slope;

    double xcoord, ycoord;
    Point p1, p2;

    cout << "Input x coordinate of the first point" << "\n";
    cin >> xcoord;
    cout << "Input y coordinate of the first point" << "\n";
    cin >> ycoord;

    p1 = Point(xcoord, ycoord);

    cout << "Input x coordinate of the second point" << "\n";
    cin >> xcoord;
    cout << "Input y coordinate of the second point" << "\n";
    cin >> ycoord;

    p2 = Point(xcoord, ycoord);

    slope = (p2.get_y() - p1.get_y()) / (p2.get_x() - p1.get_x() );

    if (p2.get_x() - p1.get_x() == 0)
        {
            cout << "Points must have different x coordinates.";
        }
    else
        {
            cout << "The slope of the line between Points 1 and 2 is " << slope << "\n";
        }
    return 0;
}
```

- What are the results when point1=(4,2) and point2 = (4,2) ?

```
The output prints "Points must have different x coordinates."
```

- What are the results when when point1=(4,2.5) and point2 = (3,1.5) ?

```
The output prints "The slope of the line between Points 1 and 2 is 1"
```

P2. Additional Input Validation Exercise

- Build and run the following program. Describe what happens when the salaries from the first and second years are equal.

```
The percentage of difference is given as inf%.
```

```
#include <iostream>
using namespace std;
int main()
{
double fyear_sal, syear_sal,tyear_sal;

cout << "Input the first year's salary:" << "\n";
cin >> fyear_sal;;
cout <<  "Input the second year's salary:" <<  "\n";
cin >> syear_sal;
cout << "Input the third year's salary:" << "\n";
cin >> tyear_sal;
cout <<  "The difference between the third and second year salary and second and first is:"
cout << ((tyear_sal-syear_sal)/(syear_sal-fyear_sal))*100 << "%";
return 0;
}
```

- Correct and rebuild the program to not allow for case where both the first and second year salaries are the same.

```
#include <iostream>
using namespace std;
int main()
{
    double fyear_sal, syear_sal,tyear_sal;

    cout << "Input the first year's salary:" << "\n";
    cin >> fyear_sal;;
    cout <<  "Input the second year's salary:" <<  "\n";
    cin >> syear_sal;
    cout << "Input the third year's salary:" << "\n";
    cin >> tyear_sal;


    if (fyear_sal == syear_sal)
    {
        cout << "First and second year salaries must be different.";
    }
    else
    {
        cout <<  "The difference between the third and second year salary and
second and first is:";
        cout << ((tyear_sal-syear_sal)/(syear_sal-fyear_sal))*100 << "%";
    }
    return 0;
}
```

- What are the results when fyear_salary = 100000 and syear_salary = 100000?

```
The output prints "First and second year salaries must be different."
```

P3. The `if/else` statement

In the previous examples, your program probably responded to user input by ignoring cases that would result in a divide by zero. Instead, you can use the `if/else` format to explicitly specify the action to be taken.

```
if (condition)
   /* do something ... */
else
   /* do something different ... */
```

The electric company gives a discount on electricity based upon usage. The normal rate is $.60 per Kilowatt Hour (KWH). If the number of KWH is above 1000, then the rate is $.45 per KWH. Write a program that prompts the user for the number of Kilowatt Hours used and then calculates and prints the total electric bill.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{

    cout << "How may kilowatt-hours do you consume? ";

    int kilowattHours;
    cin >> kilowattHours;

    double totalCost;

    if (kilowattHours <= 1000)
    {
        totalCost = .6 * kilowattHours;
    }
    else
    {
        totalCost = .45 * kilowattHours;
    }

    cout << "The total cost is: " << setprecision(2) << fixed << totalCost;


    return 0;
}
```

According to your program, how much will it cost for:

900 KWH?	$540.00
1754 KWH?	$789.30
10000 KWH?	$4500.00

P4. Simple Loops

Frequently, a decision needs to be made whether or not to do something again. Here is a program that computes the number of digits needed to represent a number in base 10. It uses multiple `if` statements.

```
/* PURPOSE:  Count number of digits needed to express an integer in base 10
             using multiple if statements
*/

#include <iostream>
using namespace std;

int main()
{  int input;

   cout << "Input an integer between 1 and 9999: ";
   cin >> input;

   int temp = input;
   int d = 1;
   assert ( input >= 1 and input <= 9999 );

   if (temp > 9)
   {  temp = temp / 10;
      d++;
   }

   if (temp > 9)
   {  temp = temp / 10;
      d++;
   }

   if (temp > 9)
   {  temp = temp / 10;
      d++;
   }

   if (temp > 9)
   {  temp = temp / 10;
      d++;
   }

   cout << input << " can be expressed in " << d << " digits" << "\n";

   return 0;
}
```

But having to write

```
   if (temp > 9)
   {   temp = temp / 10;
       d++;
   }
```

four times, even using copy/paste, is clearly repetitive! It also only works for input <= 9999. One would like to have a way of testing that the input is still greater than 1, and executing the succeeding control block if it is. Replacing `if` with `while` does it.

```
/* PURPOSE:  Count number of digits needed to express an integer in base 10
             using while loop
*/

#include <iostream>
using namespace std;

int main()
{  int input;

   cout << "Input an integer: ";
   cin >> input;
   int d = 1;
   int temp = input;

   while (temp > 9)
   {  temp = temp / 10;
      d++;
   }

   cout << input << " can be expressed in " << d << " digits" << "\n";
}
```

The fractions 1/2, 1/4, 1/8, ... get closer and closer to 0. Change the previous program to count the number of divisions by two needed to be within 0.0001 of zero.

```
#include <iostream>
using namespace std;

int main()
{  double input;

    cout << "Input an integer between 1 and 9999: ";
    cin >> input;

    double temp = input;
    int d = 0;


    while (temp > .0001)
    {
        temp = temp / 2;
        d++;
    }


    cout << input << " must be halved " << d << " times to be smaller than .0001" <<
"\n";

    return 0;
}
```

P5. Loop Termination

Which values of `nyear` cause the following loops to terminate?

```
/* PURPOSE: Count number of year between a user-input year and the
            year 2000.
*/

int main()
{  int nyear;
   const int millennium = 2000;

   cout << "Please enter the current year";
   cin >> nyear;

   while (nyear != millennium)
   {  nyear++;
   }

   cout << " Another "<< millenium - nyear << "years to the millenium." << "\n";
   return 0;
}
```

```
Any integer equal to or below 2000 will allow the loop to terminate.
```

Re-write the preceding program so that the while loop will terminate for any integer input.

```
#include <iostream>
using namespace std;

int main()
{  int nyear;
    const int millennium = 2000;

    cout << "Please enter the current year: ";
    cin >> nyear;

    while (nyear != millennium)
    {
        if (nyear > millennium)
        {
            break;
        }

        nyear++;

    }

    cout << " Another "<< millennium - nyear << " years to the millenium." << "\n";
    return 0;
}
```

P6. Processing a Sequence of Inputs

Modify the electric rate program to allow the user to enter data for any number of months. The user should be able to enter a -1 when finished. The program should then compute the cost using the same formulas as above.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{

    cout << "How may kilowatt-hours do you consume? Enter your consumption for
each month ending with \"-1\": ";

    int kilowattHours;
    double totalCost = 0.00;
    cin >> kilowattHours;

    while (kilowattHours >= 0)
    {
        if (kilowattHours <= 1000)
        {
            totalCost = totalCost + .6 * kilowattHours;
        }
        else
        {
            totalCost = totalCost + .45 * kilowattHours;
        }

        cin >> kilowattHours;

    }

    cout << "The total cost is: " << setprecision(2) << fixed << totalCost;


    return 0;
}
```

R3. Using Boolean variables

According to the following program, what color is the resulting mixture under the following inputs?

Y N Y Red (due to uppercase)

Y Y N Red (due to uppercase)

N N N Red (due to uppercase)

```
#include <iostream>
#include <string>
using namespace std;

int main()
{  string mixture;
   bool red, green, blue;
   string string_bool;

   cout << "Include red in mixture? (y/n) " << "\n";
   cin >> string_bool;
   if (string_bool == "y")
      red = true;
   cout << "Include green in mixture? (y/n) " << "\n";
   cin >> string_bool;
   if (string_bool == "y")
      green = true;
   cout << "Include blue in mixture? (y/n) " << "\n";
   cin >> string_bool;
   if (string_bool == "y")
      blue = true;

   if (not blue and not green)
      mixture = "RED";
   else if (not red and not blue)
      mixture = "GREEN";
   else if (not red and not green)
      mixture = "BLUE";
   else if (red)
   {  if (green or blue)
      {  if (green and blue)
            mixture = "BLACK";
         else if (green)
            mixture = "YELLOW";
         else
            mixture = "PURPLE";
      }
   }
   else
   {  if (blue and green)
         mixture = "CYAN";
      else
         mixture = "WHITE";
   }
   cout << "Your mixture is " << mixture << "\n";

   return 0;
}
```

Don't forget to print this page to a pdf file when you're finished and submit the pdf file to moodle.