CIS121 Laboratory Four

Objectives of this lab:

1. Introduce three of the primitive data types in C++.
2. Discuss some of the elementary operations that can be performed on these types.
3. Introduce the if… else… control statement.
4. Investigate the storage systems used for three primitive data types (part 1-int & float).

Normally, instructions in a C++ program are executed in the order in which they appear. While this may be sufficient for simple programs, it is not flexible enough for more complex tasks. Thus, C++ provides a variety of control statements for redirecting the flow of execution. The order of execution for statements in your program is referred to as **flow of control**.

Part 1: The if…else… Control Statement

Let's look at one example. Suppose you are organizing an event that costs $12 for everyone older than 8 and $6 for any one 8 years or younger. One way to do this is to say the ticket is $12, unless you are 8 or younger, then it is $6. In this case, you can write:

Pseudo Code (English like statement)	C++ equivalent

ticket = 12;
if(age is 8 or younger)
  ticket = 6;
	double ticket = 12.00, age;
	if(age <= 8)
	  ticket = 6;

There is another way to do this.

Pseudo Code (English like statement)	C++ equivalent

if(age is 8 or younger)
  ticket = 6;
else
  ticket = 12;
	if(age <= 8)
	  ticket = 6;
	else
	  ticket = 12;

Both of these do the same thing. In both cases, you will change the flow of execution when you reach the statement; "age is 8 or younger". If that statement happens to be true, i.e., age is 8 or younger, and then the value of ticket will change to $6, otherwise, you will go with its initialized value of $12.

In general, the statement in the parentheses is either TRUE or FALSE. Depending on that being true or false, you will change the flow of execution of the statements in the program.

In order for your program to decide about the flow of the execution, it uses a comparison operator. Examples of **comparison operators** are:

>	Greater than
<	Less than
> =	Greater than or equal to
< =	Less than or equal to
= =	Equal to
! =	Not equal to

Consider the following code:

```
int years;
years = 6; // assignment statement years is assigned the value of 6
years == 5; // relational expression, not an assignment statement
years = years - 1; // assignment statement
years == 5; // relational expression
```

In this sequence the first occurrence of years == 5 is a false statement whereas the second occurrence is true.

Sometimes we may only want a portion of code executed under certain conditions. To do so, we use **conditional statements**. For example, if you are writing a payroll program to compute wages, then the program should only compute overtime pay *if* the employee worked more than 40 hours in a given week. Otherwise, when the program is executed the overtime portion of the code should be bypassed. An **if statement** is one kind of conditional statement.

Now that you have learned about changing the flow of control, let's write a program called ticket.cpp that asks users to enter an age and displays the cost of the ticket based on the criteria that were given above. Use both methods to make sure

Experiment 4.1

```
// ticket.cpp - This program asks for an age and displays the cost of the ticket
#include<iostream>
using namespace std;

int main( )
{
    double age, ticket = 12;

    cout << "Please enter the age \n";
    cin >> age;
    if(age <= 8)
        ticket = 6;
```

```
    cout << "Your ticket costs " << ticket << endl;

    return 0;
}
```

Step 1. Change the program by using if…else… statement.

Step 2. Change the ticket.cpp program such that it displays $6 for people who are 8 years old or younger OR 65 years or older.

Part 2: C++ Basics

Data of Type Integer

In the C++ system, the maximum and minimum integer values are represented by the system defined constants INT_MAX and INT_MIN, respectively.

Experiment 4.2

In this experiment you will investigate the storage of integer values on your own particular machine.

Step 1. Execute the following program, and record the value of INT_MIN.

```
#include <iostream>
using namespace std;
int main()
{
        cout << "The most negative value of type int \n";
        cout << "represented on this C++ system is ";
        cout << INT_MIN << endl;
        return 0;
}
```

Step 2. Modify the program in Step 1 to find the value for INT_MAX. Record what you learn.

Data of Type Real

As with the storage of integers, only a finite number of values of type real can be represented in a computer's memory. Other values must be rounded to a value that can be represented. Thus, a value stored as type double is often merely an approximation of the desired value. The header file float provides the predefined constants FLT _MIN and FLT_MAX to identify the boundaries of representation for floating-point storage.

Experiment 4.3

Step 1. Execute the following program, and record the value of FLT_MAX.

```
#include <cfloat>
#include <iostream>
using namespace std;
int main()
{
      cout << "The largest floating-point value that can \n";
      cout << "be represented is " << FLT_MAX << endl;
      return 0;

}
```

Step 2. Note that this program contains the statement

```
#include <cfloat>
```

This directive tells the compiler to include the header file cfloat during compilation. This file contains the information required for the compiler to understand references to FLT_MIN. Try to compile and execute the program without this statement, and record what happens.

Shorthand notation for Assignment Statement

A common operation is that of incrementing a value as accomplished by the statement

```
x = x + 1;
```

where x is an integer variable. C++ provides a shorthand notation for this assignment statement. It has the form

```
x++;
```

which consists of the name of the variable to be incremented followed by two plus signs. A similar expression

```
x--;
```

is a shorthand notation for the statement

```
x = x - 1;
```

that decrements the value of x by one.

Expressions such as x++ and x-- can be used in more complex expressions such as in the assignment statement

```
y = 5 + x++;
```

Here, y is assigned the result of adding 5 to x and then x is incremented by 1. In particular, the combination

```
x = 2;
y = 5 + x++;
```

would result in y being assigned the value 7 and x being assigned 3. Note that the value of x is not incremented until its original value has already been used in the computation. If we had wanted the incremented value to be used in the computation, we would have used the expression ++x rather than x++. In short, ++x means to increment the value of x and then use this new value in the remaining computation, whereas x++ means to increment the value of x after its original value has been used. Thus, the sequence

```
x = 2;
y = 5 + ++x;
```

would result in y being assigned 8 and x being assigned 3. The expression x-- has a similar variation.

The C++ language recognizes a close relationship between data of type integer and character. Indeed, it is often convenient to think of the bit pattern representing a symbol (type character) as an integer value. (The printable characters in ASCII are represented by the integer values 32 through 126.) In particular, if sym is of type char and num is of type int, then statements such as

```
sym = 99;
num = 'a';
```

and

```
sym++;
```

are permitted (although not always considered good programming practice). Assuming that characters are represented in ASCII, the first of these statements assigns the character c to the variable sym since the symbol c is represented by 99 in ASCII. The second statement assigns the value 97 to the variable num since the character a is represented by 97 in ASCII. The third statement is often used to convert a letter in the alphabet to the next sequential letter (except for the letter z).

Experiment 4.5

Assuming that x and y are variables of type integer. Write a short program to discover what actually happens when the following statements are executed.

```
y = x + x + x++;
y = x++ + x + x;
```

Experiment 4.6

1) Execute the following program. The operator % is called the modulus operator. What does it do?

```
#include <iostream>
using namespace std;
int main()
{
  int a, b, c;
  a = 7 % 3;
  b = 8 % 3;
  c = 8 % 4;
  cout << a << " " << b << " " << c << endl;
  return 0;
}
```

2) Change the statement a = 7 % 3 to a = 7.0 % 3.0 and try to execute the modified program. What can you conclude?

Part 3: Programming Exercises

There are two problems descripted below. Select one problem to complete for your Lab 4. Complete extra program can gain extra credit.

1. Basic Operations

Write a program that prompts the user to enter three integers, and finish the following steps:

Step 1: prompts the user to enter the first integer

a. Determine if the number is an odd or even number.
b. Determine if the number is positive or negative.
c. Determine whether the number is –
    i. divisible by 3 **and** 4
    ii. divisible by 3 **or** 4
    iii. divisible by either 3 or 4, but not both

Step 2: prompts the user to enter the second integer

a. Determine whether the first number is divisible by the second. If the second number is zero, your program should not do division.
b. Outputs the remainder of two numbers.
c. Compare the two integers, displays the integers in non-decreasing order.

Step 3: Sort three integers

Prompts the user to enter the third integer

a. Calculate the average of the three integers.
b. Output the largest number among three values.
c. Display the three numbers from the smallest to the largest. (Optional)

2. Pay Stub

Write a program that will print out a weekly paycheck stub for an hourly worker.

The employee is paid at a rate of $16.78 per hour for regular hours worked in a week. Any hours over 40 are paid at the overtime rate of one and one half times the base pay. From the worker's gross pay, 6% is withheld for social security tax, 14% is withheld for federal income tax, 5% is withheld for state income tax, and $10 per week is withheld for union dues. If the worker has three or more dependents, then an additional $35 is withheld to cover the extra cost of health insurance beyond what the employer pays.

Write a program that will read in the number of hours worked in a week and the number of dependents as input. It will then output the worker's gross pay, each withholding amount, and the net take-home pay for the week.

Prefer to line up all of the dollar amounts in a single column.

Example Output:

```
Enter the number of hours worked this week: 40
Enter the number of dependents: 2
```

```
Your gross pay is:          671.20

Deductions
  Social Security Tax:       40.27
  Federal Income Tax:        93.97
  State Income Tax:          33.56
  Union Dues:                10.00

Your net pay is:            493.40
```

Required Test Cases:

```
33, 2
40, 3
47, 4
```

Submission:

- Submit the .cpp files and the output screen shots to Lab 4 DropBox on D2L.
- Comments your programs and add file header.