CIS121 Laboratory Two

Objectives of this lab:

- Learn about problem solving and programming
- Learn about software life cycle
- Illustration of the problem solving and programming
- Three types of program errors
- A simple C++ program

Part 1: Problem Solving

Problem: A grocery store sells many cases of soft drink everyday. In each case, there are 12 bottles and the store profits 20 cents per bottle. We want to compute the profit that the store has every day of selling soft drink. We also want to know the profit for selling soft drink in a year. Assume a year is 365 days.

1) Problem Definition:

Compute the profit that a store has in one day for selling soft drink?
Compute the profit that a store has in one year for selling soft drink?

2) Program Design - Algorithm

Before we attempt to write the program, let's develop an algorithm for solving the problem.

Design the algorithm for this problem. On a piece of paper draw a diagram or write in English the steps. Remember your algorithm must be precise. This algorithm must be translated to C++ to obtain the program.

Test your algorithm on paper. Your algorithm should work correctly. Make sure it produces the correct results.

3) Implementing the Algorithm in C++

This is where you will translate the algorithm to C++. Here is a program that is designed based on the algorithm that is given in the previous part. Check to make sure you find everything is translated preciously.

```
//L2.cpp - This C++ Program will compute the profit of selling soft drinks

#include <iostream>
using namespace std;

int main( )
{
    int cases_per_day, bottles_per_day;
    int bottles_per_case = 12;
    double profit_per_bottle = 0.2;  // 20 cents per bottle profit
    double profit_per_day, profit_per_year;

   cout << "Press enter after entering each number \n";
   cout << "Enter number of cases \n";
   cin >> cases_per_day;

   profit_per_day = cases_per_day * bottles_per_case * profit_per_bottle;
   profit_per_year = 365 * profit_per_day;

   cout << "The store has made : ";
   cout << profit_per_day;
   cout << "  per day. \n";
   cout << "That means the profit for one year will be:  ";
   cout << profit_per_year << endl;

   cout << "Good business?! \n";
   return 0;
}
```

Create the lab2 directory under the directory that you are using for this course. Change to that directory. Open a blank file, Lab2.cpp, and either cut and paste or carefully type the above program in that file, save the file, and exit.

4) Test the Program

The last thing you need to do is to test the program to make sure it produces the correct results. Assuming the store sells 10 cases per day, what would be the profit per day and per year? You can confirm the answer by hand.

Modify program L2.cpp so that it uses 22 Cents profit per bottle in the calculations. This time in your output, display both the number of bottles sold and the profit for one day, one year, and 10 years.

Par 2: About a C++ program

What is he Simple C++ Program?

On the first line of the program you have:

```
// This C++ Program will compute the profit of selling soft drinks
```

The // tells the compiler that the line is only a comment and do not participate in the computing. Comments are added for readability and/or to describe parts of a program.

On the next line you have:

```
#include <iostream>
```

This is called an include directive. It tells the compile where to find information about some of the items that are used in your program. For example: cout, <<, >>, and cin in the above program. There are other libraries that you will use to include other items. Note that the directive always begin with #.

```
using namespace std;
```

This line will tell the compiler that the names defined in iostream are to be interpreted in and "standard way". Note that your program should work without this, as your default set up is to use a global namespace. We will discuss the namespace in more details in Lab 10.

```
 int main( )
{
```

Let's just say that this marks the beginning of your main program. Actually, int is used for type "integer", main is a function name and ( ) will mark the boundary of parameters. We promise that you will learn about all this very soon. Note that the { marks the beginning of the main function and } marks the end of it. In general, remember that for every open {, you should have a corresponding }.

In the next four lines:

```
     int cases_per_day, bottles_per_day;
     int bottles_per_case = 12,
     double profit_per_bottle = 0.2;  // 20 cents per bottle profit
    double profit_per_day, profit_per_year;
```

We will declare the variables that we plan to use and we also define their type. We look at different variable types in the future labs. In the above four lines int is used to declare variables of type "integer". On the second line we not only define the bottles_per_case as an integer, we also initialized that to 12. On the 3rd and 4th lines, we have defined variables of type double. Note that each instruction is ended with a ;.

On the following two lines:

```
cout << “Press enter after entering each number \n”;
cout << “Enter number of cases \n”;
```

we have used cout to display a message on the screen. The cout statement will allow us to direct data from a variable out to the screen.

On the other hand, in the following line the cin directs data from the keyboard into a variable.

```
cin >> cases_per_day;
```

It is important but very easy to remember what the direction of << and >> is. Note that when we send data to the screen, we send it to cout so the direction must be <<, and when we send data from the keyboard to a variable using cin the direction must be toward the variable >>.

In the sample program, we have perform some calculations:

```
profit_per_day = cases_per_day * bottles_per_case * profit_per_bottle;
profit_per_year = 365 * profit_per_day;
```

In the first line we have multiplied, *, cases_per_day by bottles_per_case by profit_per_bottle, and stored the result, =, into profit_per_day. We will learn about different arithmetic operators in future labs. Explain what you have done in the second line.

Congratulations, in this lab, you learned about some the basic steps you need to take to solve a problem using a computer program. Also, you learned some of the syntax in C++. In the last part, we briefly discuss the type of error you may run into when you write a computer program.

Part 3: Program Errors

When you use a program to solve a problem, you may have one of the following three errors:

1) Syntax Error
2) Logic Error
3) Run_time Error.

1) Syntax Error

Syntax error will be the result of violation of the syntax (the grammar rules) of the programming language that you use. For instance, if you forget to put ; at the end of a C++ instruction, your compiler will not correctly compile and will display and error on the screen. To see the type of error, you can remove one of the ; a try to compile the program. Another example of this type of error is if you do not have a paired open { and close } set of braces.

2) Logic Error

The logic error will be the result of incorrect translation of your algorithm when you were writing the program. This error will not be detected by the computer and the only way to find it is to test the program carefully after it is completed. For instance, if in the following statement:

```
profit_per_day = cases_per_day * bottles_per_case * profit_per_bottle; //correct
```

which is the correct statement for computing the profit_per_day, but if by mistake, we use + instead of *, then we will have:

```
profit_per_day = cases_per_day +bottles_per_case * profit_per_bottle;  //wrong
```

We will get an answer, but that answer is not correct. The error is the result of the mistake in the translation of our algorithm. Instead of * we have used +.

3) Run_time Error

A run_time error is detected when we run a program. This type of error is mostly related to numeric calculations. For example, a computer cannot compute the square root of a negative number.

Copy or cut and paste the following program into a file, save the file as PS2.cpp, and then compile the program. You will get some syntax errors, pay attention to type of error you will get and the line number in which the error has occurred. Fix them one-by-one or as many as you can in each try, compile it until there is no more error in the program. Run the program and make sure it produces an output.

```
include<iostream.h>

int main( )
}
    cout < "This is the second program of the lab 2 . \n";
    cout << "There were some syntax errors in it that I fix them. \n';
    count >> "Syntax errors are due to the violation of the grammar of C++ \n"

    return0;
}
```

Part 4 Programming Exercise

Temperature Conversions

Write a program that converts a temperature in degrees Celsius to the corresponding temperature in degrees Rankin.

The program must read in a temperature (in degrees Celsius) from the keyboard. The program must display the converted temperature to the console. The program should handle the temperatures as floating point numbers (doubles).

Note:

The following equation is used to convert Degrees Celsius to Degrees Fahrenheit:

TF = TC * (9.0/ 5.0) + 32

The following equation is used to convert Degrees Fahrenheit (TF) to Degrees Ranking (TR):

TF = TR – 459.67

Example Output:

```
Please enter the temperature in Degrees C: 22

That is 531.27 Degrees Rankin.
```

Submission

Submit sources codes and output screenshots for Part 4 Programming exercises

– Temperature Conversion