Computer Science 2510 - Lab 1

Readings

- Class Notes
- Textbook: Chapters 1-4

Objectives

- To become familiar with the lab environment, including aliasing, editing, compiling, etc.
- To gain experience with fundamental concepts in C++, such as variables, types, input/output, if statements, loops, vectors, etc.

Notes

- Most of the exercises in this lab were taken from the "Drill" section of Chapters 2, 3 and 4 of the textbook (Bjarne Stroustrup, *Programming - Principles and Practice Using C++*, Second edition, Addison-Wesley, 2014, ISBN 978-0-321-99278-9.)

Lab Exercises

1. Create a directory for the course, called `comp2510`, and subdirectories for each of the labs, called `lab01`, `lab02`, etc.

2. Download the modified header file which accompanies the textbook, `std_lib_facilities.h` here: std_lib_facilities.h.

   Put this file in your `comp2510` directory. Any C++ files contained in one of your `lab` directories can then reference it by using: `#include "../std_lib_facilities.h"`

3. Go to the link Setting up the C++ compiler to use C++14 and follow the instructions given there.

4. **Chapter 2 Drills**

4.1. Open your favourite text editor (for example, `gedit`, `kate`, `gvim/vim/vi`, `emacs`, `nedit`, etc.) to create a file called: `hello_world.cpp`

4.2. Type in the contents of `hello_world.cpp`, as specified below (indicating the correct location for the `std_lib_facilities.h` file that you downloaded above), and save it in an appropriate directory (folder), e.g., `lab01`.

```
#include "../std_lib_facilities.h"
int main() // C++ programs start by executing the function main
{
    cout << "Hello, World!\n"; // output "Hello, World!"
    return 0;
}
```

4.3. Compile and run the "Hello, World!" program using the command "`c++14 -o hello_world hello_world.cpp`". Quite likely, something didn't work quite right. It very rarely does in a first attempt to use a new programming language or a new programming environment. Find the problem and fix it!

4.4. Now is the time to get a bit better acquainted with your compiler's error-detection and error-reporting facilities! Try the six errors from Section 2.3 to see how your programming environment reacts.

5. Chapter 3 Drills (modified)
After each step of this drill, run your program to make sure it is really doing what you expect it to.

5.1. This drill is to write a program that produces a simple form letter based on user input. Begin by typing the code from Section 3.1 prompting a user to enter his or her first name and writing "Hello, `first_name`" where `first_name` is the name entered by the user. Then modify your code as follows: change the prompt to "Enter the name of the person you want to write to" and change the output to "Dear `first_name`,". Don't forget the comma.

5.2. Add an introductory line or two, like "How are you? I am fine. I miss you." Be sure to indent the first line. Add a few more lines of your choosing -- it's your letter.

5.3. Now prompt the user for the name of another friend, and store it in `friend_name`. Add a line to your letter: "Have you seen `friend_name` lately?"

5.4. Declare a `char` variable called `fav_sport` and initialize its value to `s`. Prompt the user to enter an `h` if the friend's favourite sport is hockey and a `b` if the friend's favourite sport is basketball. Assign the value entered to the variable `fav_sport`. Then use two `if`-statements to write the following:
If the friend's favourite sport is hockey, write "Wrong, `friend_name`! I think you'll find it's soccer."
If the friend's favourite sport is basketball, write "`friend_name`, we are no longer friends. That's even worse than hockey."

5.5. Prompt the user to enter the age of the recipient and assign it to an `int` variable `age`. Have your program write "I hear you just had a birthday and you are `age` years old." If `age` is 0 or less or 110 or more, call `simple_error("you're kidding!")` using `simple_error()` from `std_lib_facilities.h`.

5.6. Add this to your letter:
If your friend is under 12, write "Next year you will be `age`+1."
If your friend is 17, write "Next year you will be able to vote."
If your friend is over 70, write "I hope you are enjoying retirement."
Check your program to make sure it responds appropriately to each kind of value.

5.7. Add "Yours sincerely," followed by two blank lines for a signature, followed by your name.

6. **Chapter 4 Drills**
Go through this drill step by step. Do not try to speed up by skipping steps. Test each step by entering at least three pairs of values -- more values would be better.

6.1. Write a program that consists of a `while`-loop that (each time around the loop) reads in two ints and then prints them. Exit the program when a terminating '|' is entered.

6.2. Change the program to write out "the smaller value is:" followed by the smaller of the numbers and "the larger value is:" followed by the larger value.

6.3. Augment the program so that it writes the line "the numbers are equal" (only) if they are equal.

6.4. Change the program so that it uses `double`s instead of `int`s.

6.5. Change the program so that it writes out "the numbers are almost equal" after writing out which is the larger and the smaller if the two numbers differ by less than 1.0/100.

6.6. Now change the body of the loop so that it reads just one `double` each time around. Define two variables to keep track of which is the smallest and which is the largest value you have seen so far. Each time through the loop write out the value entered. If it's the smallest so far, write "the smallest so far" after the number. If it is the largest so far, write "the largest so far" after the number.

6.7. Add a unit to each `double` entered; that is, enter values such as `10cm`, `2.5in`, `5ft`, or `3.33m`. Accept the four units: `cm`, `m`, `in`, `ft`. Assume conversion factors 1m == 100cm, 1in == 2.54cm, 1ft == 12in. Read the unit indicator into a string. You may consider `12 m` (with a space between the number and the unit) equivalent to `12m` (without a space).

6.8. Reject values without units or with "illegal" representations of units, such as `y`, `yard`, `meter`, `km`, and `gallons`.

6.9. Keep track of the sum of values entered (as well as the smallest and the largest) and the number of values entered. When the loop ends, print the smallest, the largest, the number of values, and the sum of values. Note that to keep the sum, you have to decide on a unit to use for that sum; use meters.

6.10. Keep all the values entered (converted into meters) in a `vector`. At the end, write out those values.

6.11. Before writing out the values from the `vector`, sort them (that'll make them come out in increasing order).

Author: Department of Computer Science, MUN (BE220531)