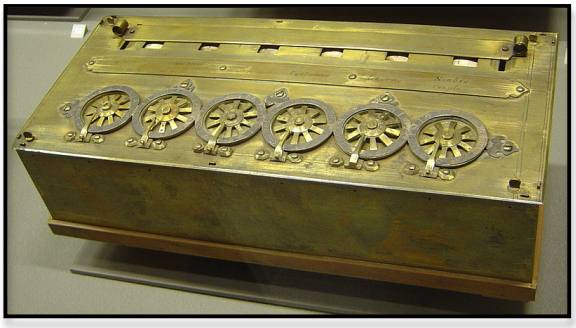
ECE2036 **Fall Semester, 2013**

Lab 1 – Pascal Calculator

Assigned: Aug 29, 2013

Section C -*Due: Sept 11, 2013 11:59pm*
Section B -*Due: Sept 12, 2013 11:59pm*
Section A -*Due: Sept 13, 2013 11:59pm*

A key feature of C++ is the ability to create classes, which are central to object-oriented programming (OOP). In this lab, you will create a single object in software that represents a simple adding machine that is, at a high level, generally similar to the basic calculator created by Blaise Pascal in 1642, which is called the Pacaline (see figure below).



For this first lab, you will need to help create the class `Pascaline`. Although this is not how the Pascaline exactly operated, for your Pascaline object you will need three private integer data members, `register1`, `register2`, and `resultRegister`. Also you will need the following member functions for your object, which are `setRegister1( int )`, `setRegister2 (int)`, `setRegisterResult(int)`, `getRegister1()`, `getRegister 2()`, `getRegisterResult()`, `addRegisters()`, `clearRegisters()`, `getInputValues()`, and `displayOutputValue()`.

A **hardcopy** of skeleton code for this lab was provided to you during class time to help you get started. As a part of this lab, it is your responsibility to manually type in this skeleton code! Do NOT get a softcopy from another source!

You will include the implementation of the member functions OUTSIDE the class definition; however, in this lab your entire program will be in one text file, which you should call `pascaline.cc` . As explained in section 3.6 in the textbook, we will start to separate files in a more sophisticated way to include a class interface in a header file with a separate implementation file for class member functions.

Programming Requirements

Your job in this lab is to create the following:

1. Create a member function `clearRegisters()` that sets `register1`, `register2`, and `registerResult` to zero. This function has no values passed to it and has a void return type.

2. Create a constructor that initializes the data members of the `Pascaline` object to zero.

3. Create a member function `addRegisters()`, which will add the value of the two registers, `register1` and `register2`, and set the `resultRegister` to the appropriate value. This function has no values passed to it and has a void return type.

4. You will need to create the member function `getInputValues()`, which should prompt the user for the inputs in the following way.

```
Please input the contents of register 1:

Please input the contents of register 2:
```

This function has no values passed to it and has a void return type.

5. You will need to create the member function `displayOutputValues()`, which displays the results of the calculation in the following way. This output example below assumes that the content of `register1` is 3 and the content of `register2` is 4. This function has no values passed to it and has a void return type.

```
Pascaline Result:  3 + 4 = 7
```

6. There are seven places marked in the code that you must insert a missing line or correct an intentionally placed syntax error. These places are marked by a comment before the error or missing line as:

```
//Missing line
```

or

```
//Correct syntax error
```

Sample Sessions

```
Please input the value of register 1: 5

Please input the value of register 2: 6

Pascaline Result: 5 + 6 = 11

Please input the value of register 1: -10

Please input the value of register 2: 8

Pascaline Result: -10 + 8 = -2

Please input the value of register 1: -9

Please input the value of register 2: -8

Pascaline Result: -9 + -8 = -17

Please input the value of register 1: 22

Please input the value of register 2: -12

Pascaline Result: 22 + -12 = 10
```

Accessing Jinx System

Instructions on accessing the jinx system can be found at:

http://users.ece.gatech.edu/~riley/ece2036/handouts/AccessingJinx.pdf

Source Code Text File

On the Jinx system you will have to create a text file that contains your C++ code. We would recommend that you use emacs, vi, or pico to create your file.

Please make a directory (i.e. "folder") called `Lab1` where you keep your files. To make this directory use the following command in your home directory:

```
mkdir Lab1
```

To go into this directory from your home directory, you can use the following command:

```
cd Lab1
```

To get back to your home directory you can use the command:

```
cd
```

Compiling Source Code

On the jinx system we will be using the gnu g++ compiler. At the command prompt in your `Lab1` directory use the following command to compile your source code and create an exectuable file called `pascaline`.

```
g++ pascaline.cc -o pascaline
```

To run your program at the command prompt, type in the executable file name

```
./pascaline
```

Turning in Lab1

The system administrator for the jinx cluster has created a script that you are to use to turn in your project. The scripts are found in /usr/local/bin, which should be in the search path for everyone. From your home directory enter one of the following at the commands at your prompt depending on your section.

```
riley-turnin Lab1
davis-turnin Lab1
hamblen-turnin Lab1
```

This automatically copies everything in your Lab1 directory to a place that we can access (and grade) it.