COSC 220 - Computer Science II
Lab 2

Dr. Joe Anderson

10 September 2019

1 Objectives

1. Develop a familiarity with the GNU/Linux operating system
2. Use the GCC compiler to compile more complex C++ programs
3. Practice using C++ classes, constructors, and arrays
4. Practice using objects and references to objects

2 Tasks

1. Use three separate files to write the code for this lab: `payroll.h`, `payroll.cpp`, and `main.cpp`
2. In `payroll.h`, design a `PayRoll` class that has private data members for an employees name, hourly pay rate, and number of hours worked.
   (a) Use the following header file template to make use of precompiler macros to avoid double declaration errors:

```
#ifndef PAYROLL_H
#define PAYROLL_H

class PayRoll{
  // your class prototypes here
};

#endif
```

   The `#ifndef`, `#define`, and `#endif` commands are the *pre-compiler macros* that get "executed" at compile time, and will only produce the inner `c++` code once, no matter how many times a project uses `#include "payroll.h"`.

   (b) Leave all methods undefined in the header file. Use `payroll.cpp` to define the class methods.
      i. You will need to use `#include "payroll.h"` at the top to include the prototypes.
      ii. Your methods will need to be defined with appropriate scope, e.g.

```
/*
 *  Calculate and return the total pay for
 *  this employee
 */
```

```
double PayRoll::getPay() {
  // implementation here
}
```

   (c) Add a constructor to take the name and pay rate (dollars per hour) as arguments.

   (d) Add public methods to:

      i. Set the number of hours worked

      ii. Calculate and return the total pay, based on the employees rate and the number of hours worked

   (e) Add a public method to print all the data for the employee in a nice format, including total pay

3. In the `main.cpp` file, declare and fill an array of seven PayRoll objects (with hard-corded name and pay rate)

   (a) You will need to `#include` your `PayRoll` header file at the top.

4. The program should ask the user for the number of hours each employee has worked and will then display the amount of gross pay each has earned.

5. To compile the code, you will need to need to pass *both* `payroll.cpp` and `main.cpp` to `g++` in some fasion.

   (a) Example 1: This will compile and link the code all at once into the binary called `main`

```
g++ main.cpp payroll.cpp -o main
```

   (b) Example 2: This will compile each file separately into *object* files, then link them into a binary called `main`

```
g++ -c payroll.cpp
g++ -c main.cpp
g++ main.o payroll.o -o main
```

   (c) As your programs become more complex, the second option will save lots of time when you only need to re-compile some parts of the code, but not all of it. We will see later how to gain this advantage without having to use so many commands!

6. Input Validation: Do not accept values greater than 60 for the number of hours worked.

7. Once all hours have been entered, display each employee's name and total wages for that week.

8. Repeat the above four steps, but using an array of *pointers* to PayRoll objects.

9. Use the `sizeof` operator to compare the sizes of the arrays, what is the difference and why?

10. Use the addresses of the allocated data to make some claims about where in memory your various different PayRoll objects are with respect to each other (there will be fourteen total objects created by the end).

11. Be sure to comment your code appropriate, and answer the above questions in a separate text document or in a block comment in `main.cpp`.

3 Submission

Upload your project files to MyClasses in a single `.zip` file.

Turn in (stapled) printouts of your source code, properly commented and formatted with your name, course number, and complete description of the code.

Also turn in printouts reflecting several different runs of your program (you can copy/past from the terminal output window). Be sure to test different situations, show how the program handles erroneous input and different edge cases.