

Assignment 1
CS 135: Computer Science I
Spring 2008

Objectives

1. Learn and demonstrate use of C/C++ data types
2. Learn and demonstrate use of C/C++ data stream abstractions for I/O
3. Learn and demonstrate prototyping and calling simple void functions

Use the `prettyPrint` functions in the file `prettyPrint.cpp` on the class webpage to ensure that your program output follows the required specifications. Your Instructor will go over a sample program that uses those functions in class and/or lab. There are two parts to the assignment.

1 Part 1: Game Scoring (10 points)

Design, code, and test a complete C/C++ program that gets information about game players and computes their game scores for a game like "Dance, Dance, Revolution (DDR)." The program should read (from standard input):

- Your first name (string)
- Your game handle (string)
- The number of correct dance steps (int)
- The number of missed dance steps (int)

In addition to your information, your program should also read your dance partner's information. That is, the program should read:

- Your partner's first name (string)
- Your partner's game handle (string)
- Your partner's number of correct dance steps (int)
- Your partner's number of missed dance steps (int)

The program should then print both your names, handles, score information, and **scores**. A player's score can be computed and printed by **calling** the function whose prototype is:

```
void computePrintScore(int correct, int missed);
```

This function takes two formal, integer parameters. The first is the number of correct dance steps and the second is the number of missed dance steps. Remember that you will have to declare this function's **prototype** before you **call** the function in your program.

You will always declare function prototypes before the **main** function. Annotated code that defines **computePrintScore** follows:

```
void computePrintScore(int right, int wrong)
{
```

The line above starts the definition of the function and its formal parameters, and the open curly brace begins the function. Subsequently we declare a variable of type **double** for holding the value of the computed score.

```
    double computedScore;
```

We then compute the score using the formula $1000 * \frac{c}{c+i}$

where **c** is the number of correct dance steps, and **i** is the number of incorrect dance steps. The computed score value is then assigned to the variable **computedScore**. All this is accomplished by the line:

```
    computedScore = 1000.0 * ((double) right / (right + wrong));
```

Finally, the function **computePrintScore** calls another function **prettyPrintDouble** to print the computed score. Note that functions should normally not be written to accomplish more than one task in order to support modularity and reusability. However in this case, **computePrintScore** does two things (i.e., compute and print) which saves you some actions you will be learning about next week.

```
    prettyPrintDouble(computedScore);
```

The function ends at the close curly brace.

```
}
```

Copy the above code (retype the 6 lines of code) and use it. Here is sample interaction with your program:

```
Enter your first name : John
Enter your game handle: sjl
Enter the number of correct dance steps you made: 100
Enter the number of dance steps you missed: 20
Enter your partner's first name : Jill
Enter your partner's game handle: JJ
Enter the number of correct dance steps your partner made: 120
Enter the number of dance steps your partner missed: 0
```

```
John    sjl    100    20    833.33
Jill    JJ    120    0    1000.00
```

Your output must look as neat and as lined up as it does above and you must use the following filename for writing your code: `ddrScore.cpp`

2 Part 2: Need for Speed (10 points)

Design, code, and test a complete C/C++ program that gets information about baseball pitchers and the speed of their fastball (typically the fastest a pitcher can throw the ball) and computes the time taken for the ball to travel from the pitcher's mound to home plate. The batter must react and bring the baseball bat around to try hit the ball within this time or get a strike. Your program should read the following information from standard input.

- Pitcher's name
- Pitcher's fastball speed

Your program should then print the pitcher's name, speed, and time taken for the ball to travel to home plate. For comparison, your program should also print the name, speed, and the computed time for an average person's (named "Chris") fastball. Assume that the average person can pitch fastballs at 55 feet per second. Given the speed, the time for the ball to travel between the mound and home plate can be computed and printed by calling the function whose prototype is:

```
void computePrintTime(double s);
```

which takes a single formal parameter of type `int`.

Remember that you will have to declare this function's prototype before you call the function. Annotated code for the definition of `computePrintTime` follows:

```
void computePrintTime(double s)
{
```

This function only takes a single formal parameter, `s`, the pitch speed. Now, an identifier like `s` could mean anything to someone reading your function's definition. A much better identifier name for a variable used to hold a baseball's speed is `speed` or `baseballSpeed`. There's more to type but the code is much, much easier to understand! You should modify this code and use a different and better variable name.

The rest of the function is similar to `computePrintScore` from part 1 and uses the well known $distance = speed \times time$ formula to compute the time taken. 66.5 feet is the distance from the pitching mound to home plate.

```
    double timeTaken = 66.5/s;
    prettyPrintDouble(timeTaken);
    return;
}
```

Copy the above code and use it. Here is sample interaction with your program.

```
Enter pitcher's name : Clemens
Enter pitch speed: 143.7333
```

```
    Clemens    143.73    0.46
      Chris     55.00    1.21
```

```
Enter pitcher's name : smolz
Enter pitch speed: 136.4
```

```
    smolz     136.40    0.49
      Chris     55.00    1.21
```

Your output must look as neat and as lined up as it does above and you must use the following filename for writing your code: **bbTimer.cpp**.

3 Turning in your lab assignment

Assume that this format will be used for all your laboratory assignments throughout the semester unless otherwise specified.

Turn in a Folder (Binder) containing:

- 1) Cover sheet with:
 - a) Assignment Number
 - b) Your Section Number
 - c) Your name and **your email address**
 - d) Your TA's name

- 2) Source code and executables on a rewritable CD/USB stick with your name and section number written on the CD/USB stick. This rewritable CD/USB stick should contain:
 - i) Source code files (bbTimer.cpp, ddrScore.cpp)
 - ii) Executables for each of the sections of this assignment.

- 3) For each of the sections in this assignment, hardcopy showing the running of your program on test cases.

Ask an instructor or TA if you have questions.