CSE 332 Studio Session on C++ Data: Variables and Basic Types

These studio exercises are intended to gain familiarity with basic C++ data concepts and techniques, some of which may be somewhat familiar from your previous programming experience, and to cover a few details specific to C++ (and especially the C++11 standard) that may not be obvious at first glance.

In this studio you will again work in groups of 2 or 3 people. Students who are more familiar with the material are encouraged to help those who are less familiar with it and asking questions of your instructors and teaching assistants during the studio sessions is highly encouraged as well.

If you are present in class, please create a new email to the course account (cse332@seas.wustl.edu). CC all teammates in the email. As you work, record your answers directly in the email. As class ends, please send your email with the subject line **"Data Studio"** for full credit on the studio. You may not finish all exercises in class, however the material covered in any exercise may be covered on an exam. So even though it is not required to receive full credit on the studio, it may be a good idea to finish the required exercises outside of class.

If you are not present in class, you are required to complete all required exercises in the studio for full credit. The enrichment exercises are optional but are a good way to dig into the material a little deeper, especially if you breeze through the required ones. Finish the full studio on your own time and send an email with your answers to the course email account with the subject line **"Data Studio"**.

PART I: REQUIRED EXERCISES

1. Form a team of 2 or 3 people of your choice and write down the names of the team members as the answer to this exercise.

2. Log into one of the studio lab's Windows machines, open up Visual Studio 2015, and create a new Visual C++ Win32 Console Application project (for example, named **`DataStudio`** or something similar that identifies which studio this is for). Change the signature of the main function to be **`int main (int argc, char * argv[])`** and add a line that prints out **`argv[0]`** (position **`0`** of **`argv`**, so named because it holds the program's **arg**uments **v**ector) to **`cout`** (the standard output stream). Add any necessary precompiler and compiler directives, build your project, and fix any remaining warnings or errors.

Once the program builds correctly open up a console window as you did in the previous studio, change to the directory where the executable program for the project was created, and run it. As the answer to this exercise please say (1) what output the program produced, and (2) what that tells you about what is always contained in position **`0`** of a program's arguments vector.

3. In your main function, declare a pointer to const character (of type **`const char *`**) as the loop variable of a **`for`** loop, and using only pointer operators like **`++`** and **`*`** (i.e., without using the square bracket array indexing operator **`[]`**) do the following: (1) initialize the pointer to point to the first character of **`argv[0]`**; (2) at each new position test whether the pointer still points to a non-zero

character (anything other than `'\0'`) and if so print out the character at that position and then an end of line; and (3) after each step move the pointer to the next character position in the program name, using the **++** operator on the pointer, until the end of the string is reached. As the answer to this exercise give what your program printed.

4. What happens if you try to repeat exercise 3 using a reference instead of a pointer? Please give three ways in which references differ from pointers in C++, and say which one of those three explains what happens when you try to use a reference that way in C++, as the answer to this exercise.

5. Repeat exercise 3 using the **`auto`** keyword instead of **`const char *`** to declare the pointer variable. Build and run your program, and as the answer to this exercise please indicate whether you observed any difference in how the program behaved for this exercise, compared to how it behaved in exercise 3 (and if you did see any differences what they were).

6. Again without using array indexing operators, modify your solution to exercise 5 so that instead of only iterating through the characters in the program name, it iterates through all of the characters of all of the strings that were passed as arguments to the program. As the answer to this exercise, please identify all of the pointer operators (not array operators) that you used, and describe briefly what each one does.

PART II: ENRICHMENT EXERCISE (optional, feel free to skip some and do ones that interest you)

7. Repeat exercise 5 using **`decltype`** in place of **`auto`** (but still initialize the pointer variable). Build and run your program. Then, try the same thing without initializing the variable. As the answer to this question please indicate what you observed in each of those cases.