

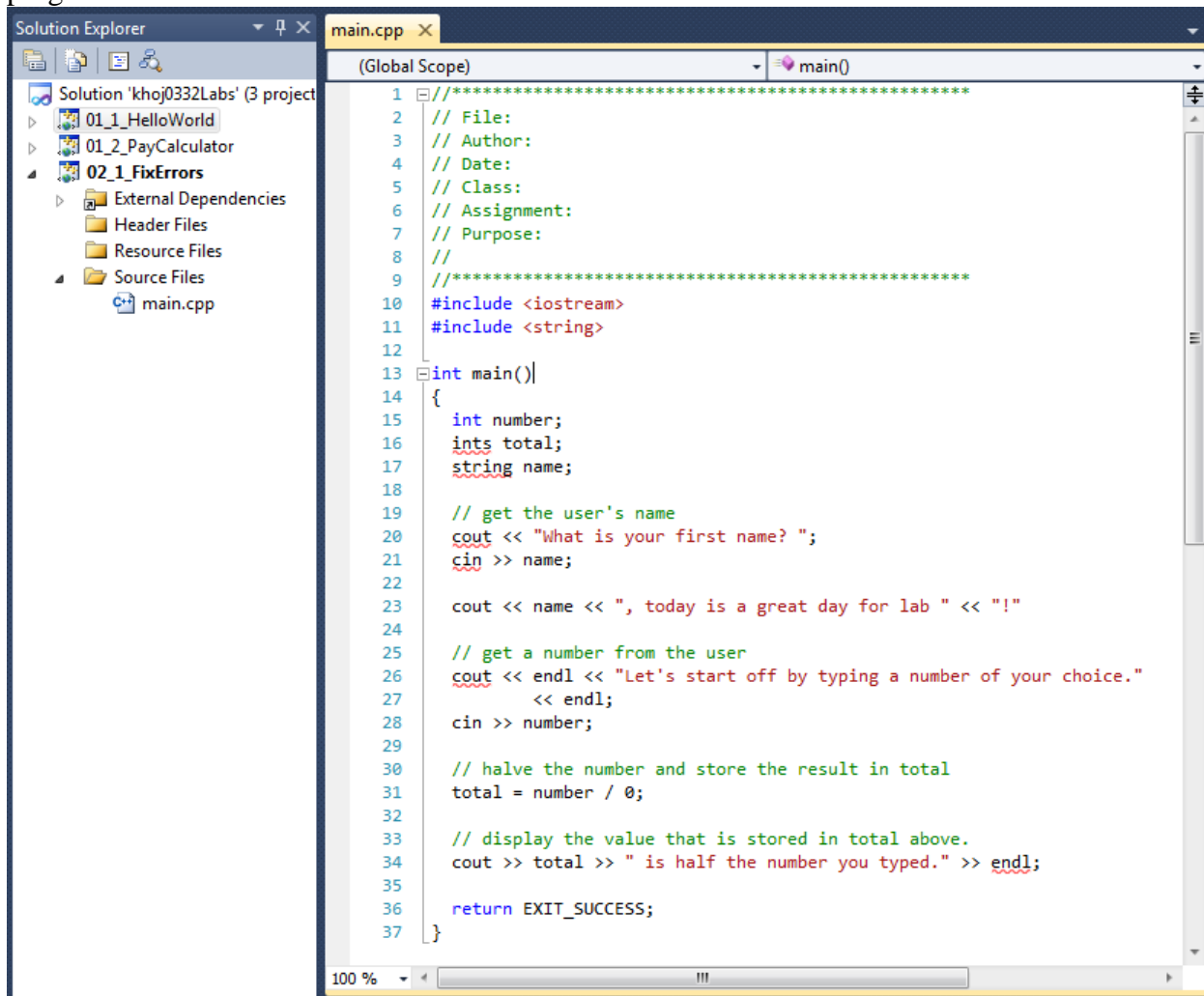
## CS 150 Lab 2

### Introduction to Compiler Errors, Variables, Assignments and Output

The purpose of today's lab session is to allow you to gain experience using primitive data types, assignment statements and output statements. Each activity today will consist of a small number of computations and a set of output to display the results, but first you will correct the syntax errors in a C++ program.

#### Lab 2.1: Fixing a C++ program

A programmer was assigned the task of writing a C++ program that allows the user the ability to enter a name and a number. Then half of the number entered is displayed. The programmer is also charged with making the program user friendly. The programmer produced the following program:



```
1 //*****
2 // File:
3 // Author:
4 // Date:
5 // Class:
6 // Assignment:
7 // Purpose:
8 //
9 //*****
10 #include <iostream>
11 #include <string>
12
13 int main()
14 {
15     int number;
16     ints total;
17     string name;
18
19     // get the user's name
20     cout << "What is your first name? ";
21     cin >> name;
22
23     cout << name << ", today is a great day for lab " << "!"
24
25     // get a number from the user
26     cout << endl << "Let's start off by typing a number of your choice."
27         << endl;
28     cin >> number;
29
30     // halve the number and store the result in total
31     total = number / 0;
32
33     // display the value that is stored in total above.
34     cout >> total >> " is half the number you typed." >> endl;
35
36     return EXIT_SUCCESS;
37 }
```

## **Lab 2.1: Syntax & Runtime errors**

### **Create a project:**

The following instructions are for creating a C++ project called **02\_1\_FixErrors** in your solution **PUNetIdLabs**.

- 1. Open up your solutions folder PUNetIDLabs that currently contains two projects 01\_1>HelloWorld and 01\_2\_PayCalculator created from the last lab.**
- 2. Create a new project called 02\_1\_FixErrors**
- 3. Add a new source file main.cpp to the Source Files folder in the project 02\_1\_FixErrors.**
- 4. Copy the program from fixerrors.cpp in the CS150Public to your Desktop. You can open the file fixerrors.cpp using Studio and then copy the contents into main.cpp.**
- 5. Set 02\_1\_FixErrors as the Startup Project and then build the project.**

### **► STOP – Show the instructor or TA**

The built program will give you errors produced by the compiler. You are likely to encounter many compiler errors as you progress through this course, and it's important for you to learn to recognize & fix these errors.

Each error message shows a line number in the file that contains the error. You can double click on the error message and Visual Studio will mark that line in the editor with a blue bar on the left of the line numbers. Take a look at the marked lines and fix the errors.

**Q. How many total errors did the compiler list? A. \_\_\_\_\_**

**Q. Which line numbers contained an error? A. \_\_\_\_\_ (Notice that a single line might have multiple errors)**

**Q. Explain what caused the error on line 20:**

**A. \_\_\_\_\_**

**Q. Explain what caused the very last error:**

**A. \_\_\_\_\_**

### **► STOP – Show the instructor or TA**

Eliminate ALL compiler errors. Do not fix anything that is not a compiler error.

Build your program which should produce a warning. You need to scroll up in the Output window to find the warning message.

**Q. What is the warning message?**

A. \_\_\_\_\_

**Note: When you turn in a working program, there are to be no compiler errors AND no warning messages or you will lose points.**

**Q. What happens if you run your program without eliminating the warning message?**

A. \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

This error is called a **runtime** error, meaning the error occurs when you run the software. The program builds but one or more of the operations the programmer asked the computer to perform is incorrect. Think about the error that you see and then look in the code to try to find the line of code that contains the **runtime** error. Fix the error, rebuild the software, and test it again.

**Run your program and test your program with the following testcases:**

**Testcase #1: John 10**

**Testcase #2: Sara 9**

**Q. Do you feel that you are getting valid output? Why or why not?**

A. \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

**► STOP – Show the instructor or TA**

## Lab 2.2

For this lab you will need to write a new program!

**Creat your fourth project called 02\_2\_NumberCruncher in your solutions folder PUNetIDLabs.**

**What does this program need to do?**

This program must ask the user for two **integers**. The program must then display the sum of the integers and the difference of the integers.

Example input and output is shown below. Your program's output must look EXACTLY like the output below. The large, bold text is data entered by the user from the keyboard.

```
*****  
Number Cruncher 1.0  
*****  
  
Please enter an integer: 9  
Please enter an integer: 100  
  
Sum: 9 + 100 = 109  
Difference: 9 - 100 = -91
```

**BEFORE YOU START PROGRAMMING answer these questions:**

**Q: What data does your program need to get from the user?**

**A:** \_\_\_\_\_

**Q: What declaration statements do you need for these data?**

**A:** \_\_\_\_\_

**► STOP – Show the instructor or TA**

**Write a completely documented program to produce the above output.**

**► STOP – Show the instructor or TA**

Once both projects are completed, place your solution PUNetIDLabs into the CS150-02 Drop folder on grace. Your solution is to have all four projects completely working and correct.

## Optional Challenge:

You do not need to submit the optional challenge!

For this lab you will create a new Visual Studio Project named **02\_3\_GeometryHelper** in your solutions folder PUNetIDLabs.

The goal of this program is to calculate the area and perimeter of a rectangle and then display both of these values on the screen. When displaying each value to the screen, display the formula used to calculate that value as well. You may use the previous program as a guide. Ask the user for the length and width of the rectangle as well as the unit of measurement used. Allow the user to provide a number with a **decimal point**. Make sure your program works with a length of 4.1 and width of 1.9. Example input and output follows.

```
*****
Geometry Helper 1.0
*****

What is the length and width of the rectangle? 4.1 1.9
What is the unit of measurement? meters

Perimeter:
4.1 + 4.1 + 1.9 + 1.9 = 12 meters
Area
4.1 * 1.9 = 7.79 square meters
```

**Q: What data does your program need to get from the user?**

**A:** \_\_\_\_\_

**Q: What declaration statements do you need for these data?**

**A:** \_\_\_\_\_

\_\_\_\_\_

Be sure to add comments to your code to detail how each variable is used and the purpose of each calculation.