

# CS 150 Lab 12

## if statements and the Debugger.

The main objective of today's lab is to practice arrays and function and learn to use the Visual Studio debugger.

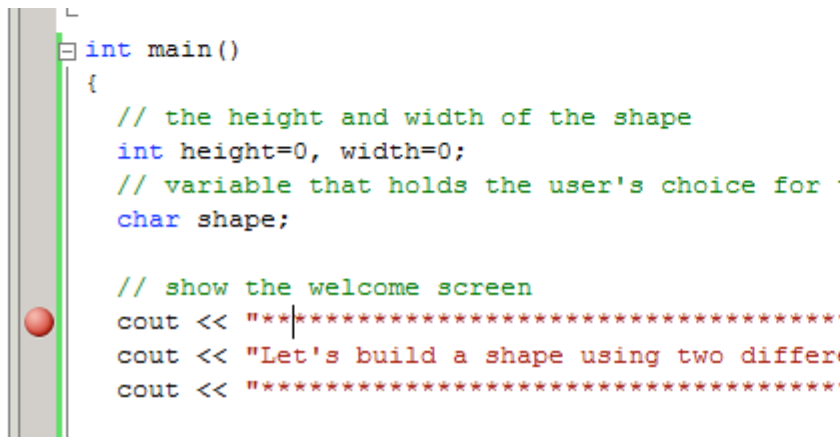
### Lab 12.1

This part of the lab will focus on using the Visual Studio debugger to find problems in the source code. Copy the directory **12BrokenCode** from "CS150-01 Public\Lab Projects" to your Desktop. This project contains a number of errors that you need to find using the debugger. Read the comments carefully to understand what the program needs to do to operate correctly.

#### To use the Debugger:

The debugger allows you to execute the program one statement at a time, and examine the value stored in each of the variables at each statement.

The first thing you need to do is set a *breakpoint* near the beginning of the main() function. A breakpoint is a spot in the code where the debugger will stop, or break, the execution of the program and wait for your input. To set a breakpoint using the debugger, click in the gray bar to the left of the statement where you want to set the breakpoint. A red sphere will appear. The image below shows a breakpoint that has been set. Click the red sphere to remove the breakpoint.



```
int main()
{
    // the height and width of the shape
    int height=0, width=0;
    // variable that holds the user's choice for
    char shape;

    // show the welcome screen
    cout << "*****\n";
    cout << "Let's build a shape using two differ\n";
    cout << "*****\n";
```

To start your program using the debugger, use the Debug | Start Debugging menu option. The program will run until it hits the first breakpoint. When it comes to the breakpoint, execution will stop and you will be able to interact with the debugger. A yellow arrow will appear on the red sphere to show you where the program has stopped. That is shown below.

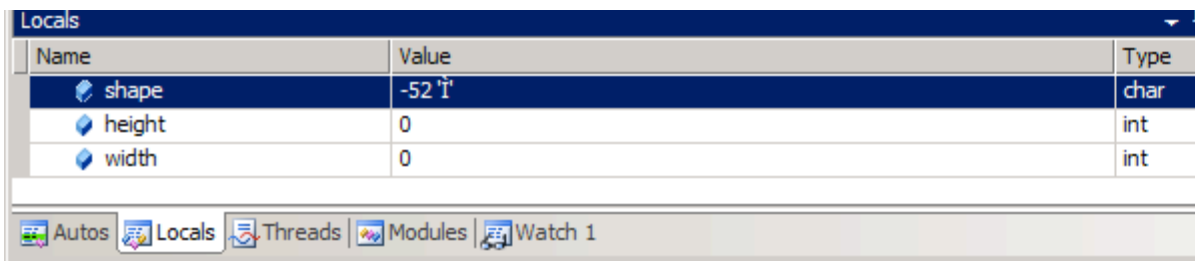
```

int main()
{
    // the height and width of the shape
    int height=0, width=0;
    // variable that holds the user's choice for
    char shape;

    // show the welcome screen
    cout << "*****"
    cout << "Let's build a shape using two differ
    cout << "*****"

```

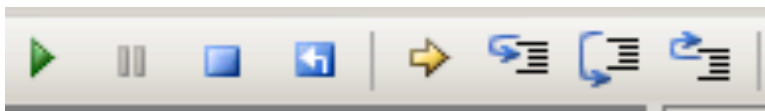
At the bottom left of Visual Studio you should see a list of variables defined in main(). The variables height and width should have a value of 0 and shape will have a random value since it has not been initialized. The Autos tab shows a set of variables Visual Studio guesses you may be interested in depending on the statement you are executing. The Locals tab shows all the variables defined in main().



To step through the execution, one statement at a time, use the stepping buttons in the icon bar at the top of Visual Studio.

The long blue arrow that jumps from the top to the bottom of the black lines will step to the next statement. The shorter blue arrows that jump into or out of the black lines will cause the debugger to try to jump into or out of function calls. We will use those more when we start writing our own functions. The green triangle runs the program until the next breakpoint is reached. The blue square stops the debugger and ends the running program.

### Debug the Code:



Run the code. What is it doing wrong? Use the debugger to step through the code and examine the values stored in the variables as the program runs. Be sure to see what happens when you step through a switch statement or a loop statement. When you fix a bug, write a comment in the code describing what you needed to fix.

► You do not need to turn this project in to the Lab drop folder, but be sure you show the instructor or the TA that you have fixed the bugs in the program and can use the debugger to step through the program's execution.

## Lab 12.2

For this part of the lab, you need to create a new Visual Studio project and implement the following program. Name your project 12\_2ArraysXXXXXX where XXXXX is your PUNetID.

**Be sure to put this assignment in the “CS150-01 Lab” folder when you are done! You have until 6 pm on Friday, November 16<sup>th</sup>.**

For this lab, you will need to use at least **one array** and at least three functions that take an array as an argument. You may use a one dimensional or two dimensional array, and you may use more than one array. **Be sure to answer all of the following questions before you start!**

Write a program that will read in 10 integers from the user in groups of 5. **For each group of integers:**

- print to the screen all the even integers
- print to the screen all the odd integers
- print to the screen all 5 integers in the order in which they were entered
- print to the screen all 5 integers in the reverse of the order in which they were entered
- print the sum of all the integers

After doing the above for each group, print the sum of all 10 integers and the pairwise sums of all the integers. A pairwise sum is the sum of the first integers in each group, the sum of the second integers in each group, etc. Finally, ask the user for a file name and store all ten integers, separated by tabs, to that file. Put the integers from Group one on the first line and the integers from Group two on the second line.

### **Sample Input**

```
*****
*****The Integer-O-Matic*****
*****
Please enter 5 integers: 1 3 4 9 2
Please enter 5 integers: -8 42 99 -1 7

Group One
Even integers: 4 2
Odd integers: 1 3 9
All integers: 1 3 4 9 2
Reverse integers: 2 9 4 3 1
Sum of the integers: 19

Group Two
Even integers: -8 42
Odd integers: 99 -1 7
All integers: -8 42 99 -1 7
Reverse integers: 7 -1 99 42 -8
Sum of the integers: 139

Sum of all integers: 158
Pairwise sums: -7 45 103 8 9
What file would you like to save the integers to: ints.dat
```

