# Computer Science II — Lab 1

## Overview

There are three points associated with each lab. These are "checkpoints" for completion of work. When you have completed each checkpoint, raise your hand and one of the TAs will quickly check your work.

## Organization and Rules

Here are a few significant organizational issues and rules about labs. The rules will be enforced in all labs.

- No IM, no email, no network! With the exception of downloading lab files provided by the instructor at the start of lab, you are not allowed to use the network at all, in any lab. Unplug your network connection, and remove/disable your wireless ethernet card. Anyone caught using the network during lab will be given an immediate 0 for that lab. There will be no exceptions.
- Get to know the other students in the lab. Introduce yourself to your neighbors. You may ask your fellow students questions about the lab. This will help reduce the burden on the TAs and will reduce your waiting time in lab. **Each student must produce his/her own solution.**
- Part of earning a checkpoint for a lab may involve answering a short question that the TA asks you. If you have done the checkpoint and understood it, you should have no trouble earning this credit. If you have relied on help from other students too much, you may find the question hard to answer.

## Checkpoints

1. The first checkpoint involves getting started with the Microsoft Visual C++ Development environment and compiling programs. It is a fairly quick introduction and it is specific to the *.net* version of the environment. Depending on when you purchased your laptop, you may have different versions of *.net*, but this shouldn't matter. You are welcome to use other development environments, especially if you have a Mac.

(a) **Create a folder** on your laptop to hold CS II files. Create a sub-folder to hold CS II labs. Create a sub-folder for Lab 1.

(b) **Start up Developer Studio.** Spend some time exploring the user interface, especially the online help. Help can be easily accessed through the 'Help/Search' menu. Also notice that if you leave the mouse over an icon for a second or two, a "tool tip" will pop up telling you what the icon does.

Work within Developer Studio is organized around projects and solutions. Projects are collections of files used to create an executable program; a solution is a collection of one or more projects. Browse the online help on solutions and projects.

(c) **Create a workspace.** Use the `File->New->Project` menu and click the `Visual C++ Projects` folder. There are multiple kinds of projects, but you only need to worry about the one named: `Win32 Project`. Make this choice, select the folder you created for your CS II labs as the Location, and then give your project a name such as Lab 1 (as you type, Visual Studio will confirm what the actual path of your project will be on your computer, which will make finding the executable program much easier). Click OK.

You will then see a very short wizard dialog that makes sure the project that is being created fits all of your needs — click on Applications Settings and change the program from a Windows Application to a Console Application using the row of check boxes. You should also check the Empty Project option to make sure that you are starting from a completely clean slate. Click OK.

(d) Using a web browser, **copy the following file** to your Lab 1 folder:

`http://www.cs.rpi.edu/courses/fall04/cs2/lab01/frame.cpp`

**From this point on, you may not use the network during this lab.**

(e) **Add your file to the project.** Use the `Project->Add Existing Item` menu and browse to the location where you made your local copies of the lab file (`frame.cpp`). Select the file that you want to add (note you can select multiple files by holding down the Ctrl key while clicking) and click OK. Then go to the Solution Explorer window (if it does not appear on the screen, you can open it from View menu) of the workspace, click the "+" next

to your project's Source Files folder, and you should see the file listed.

(f) Attempt to **build the project.** Go to the `Build` menu and select `Build Lab 1`. This is the Developer Studio term for **compiling** and **linking**. The process of compiling a program translates the high-level C++ code into machine-level, "object" code. The compiler itself is an extremely complicated program and is the most important part of the Developer Studio. The linking step gathers the object code from your program together with other code ("libraries") that your program needs which has already been compiled ("pre-compiled") and stored elsewhere. For your program this is just the standard library, which includes code for i/o streams and strings. Once the linking step is complete, you have an executable program.

**Important note: you are expected to know the above definitions of compiling and linking!**

We have introduced a number of errors into this program so that it will not compile correctly to produce an executable program. Compilation and linking errors are directed to the status pane along the bottom of your screen. If you double click on an error, the status pane will scroll to the error, and the position of the cursor in the editing pane will move to the file and line number where the error occurred.

(g) **Completing checkpoint 1.** Show one of the TAs the compiler errors that you obtained to demonstrate that you have reached this point.

**Note:** Learn how to save all of your work (look under the File menu). It is a good idea to save your work frequently, so that if the computer crashes you do not lose too much.

2. This checkpoint requires that you fix the compiler errors and then run the program.

   We have introduced only four errors in the program, but these cause many apparent errors in the compilation.

   - The first of these will refer to `gretin`, which is mistyped. The error message itself may be significantly more complicated; it might refer to `basic_string<char,...`, which at first looks confusing.

Objects of type `string` are really special cases of `basic_string` objects. The compiler immediately replaces the name `string` with this much longer name; hence the error message.

- The second error actually causes most of the remaining errors. We have left off `std::` in front of `cout`. Without this, the compiler doesn't know that `cout` exists. This produces the first message. The other messages are caused by the compiler being confused about what to do with the unknown `cout` in the context of the ostream operation.
- You should be able to figure out the last errors on your own, but don't try yet. Instead, read on.

The second error may be used to illustrate a good rule-of-thumb in finding and fixing compiler errors. In general, it is not a good idea to try to fix all errors at once. Sometimes the first or second error causes many subsequent errors. Therefore, you should make a habit of fixing only the first few errors (maybe even just the first) before recompiling. Stop trying to fix them at the first one that either you don't understand or you think might possibly be caused by an earlier error.

With the above discussion in mind, fix all of the compiler errors, recompiling and linking as needed to produce an executable program.

Now, you are ready to execute the program. You can do this by clicking on the `Debug` menu and selecting the "!" option. This will produce a new pane on which you will have to type the input to the program.

**Completing Checkpoint 2:** Show the TA that you have successfully fixed the compiler errors, compiled and executed the program.

Modify the code so that it gets both a first name and a last name, frames them, and outputs them. Here is a sample run:

```
Please enter your first name: Andrew
Please enter your last name: Smith

************************
*                      *
* Hello, Andrew Smith! *
*                      *
************************
Press any key to continue
```

You need to do very small changes; I only added four lines to the given code. Show the changes and a sample run to the TA.

3. Now that you have some initial experience with the Visual C++ environment, here's a bit of a chance to write your own program, working with strings.

   Write a program that reads the first, middle and last name of an individual into three different strings, concatenates them together with an intervening blank character to create a fourth string, and outputs the full name of the person. You should also output the lengths of the first name, the last name and the full name. Here is an example of what should appear on your screen when you are done running your program

```
Please enter your first name: Andrew
Please enter your middle name: John
Please enter your last name: Smith
Your full name is Andrew John Smith.
The length of the first name is 6 characters.
The length of the last name is 5 characters.
The length of the full name is 17 characters.
```

   To get started, you will need to

   - Create a new project within the current solution.
   - Create a new C++ source file. To do this, click

     `Project -> Add New Item`

     and select a C++ source file (.cpp file). Be sure it is being added to the new project (it should be) and then give it a name (something like `checkpt3.cpp`).

   Now, type in the source code, save it, compile it (fixing any compiler errors you make), and run it.

   Show the resulting program and its output to a TA to complete this checkpoint.

   **Note:** It is boring to write `std::cout` or `std::string` all the time; it also leads to many compiler errors. Instead, you can insert the following line in your code:

```
using namespace std;
```

You should insert this line after the `#include ...` statements. Now you could use any object from the namespace of the standard library without preceding it by `std::`

**TRY IT!!**