

Programming in C++ with Eclipse

Introduction

by Derek Schuurman

This set of introductory exercises are intended to introduce you to programming C++ in the Eclipse integrated development environment.

PART 1: Using an Editor and Compiling Your First Program

1. Log on to a Linux Workstation and open a **terminal** window. You can open a terminal windows by selecting “Terminal” from the “System Tools” menu. You may want to make a shortcut for a terminal on your dektop.
2. Type “**mkdir csc121**” to create a new directory for your CSC121 assignments. Remember, Linux is case-sensitive! Type “**ls**” to list all the files in your home directory. Change directories to the new one you created by typing “**cd csc121**”.
3. Start the **gedit** program by selecting it from the program menu or by typing “**gedit**” from within a terminal window. **Gedit** is a general purpose text editor (like Notepad) that you can use to edit the source code of your programs.
4. After you start **gedit** click the *New* icon to open a new file. Next, click *Save* to give your file a name. Select the **csc121** directory and then type a name for your source file such as **program1.cpp** (note that C++ program source files typically end with a **.cpp**). This will create a new file that you can begin editing.
5. Type the following source code for your C program. Note that all C programs should start with some header comments that identify the programmer and a description of what the program does. You should also observe that the “**.cpp**” extension indicates to **gedit** that your file is a C++ program and syntax highlighting is provided to make your code more readable.

```

/* My first CSC121 Assignment
   This program prints a friendly greeting */

#include <iostream>
#include <string>
using namespace std;

int main()
{
    cout << "Please enter your first name: " << endl;
    string firstName;
    cin >> firstName;
    cout << "Welcome to the world of C++, " << firstName << endl;

    return 0;
}

```

When you have finished editing save your file. Don't worry if you don't understand what you have just entered.

6. To compile your program, return to the terminal window and type the compile command at the prompt as shown below:

```
g++ -Wall program1.cpp -o program1
```

Notes:

- Before typing this command, ensure you are in the same directory as your source file
- `g++` invokes the GNU C++ compiler
- `-Wall` turns all warnings on
- `program1.cpp` is the name of the source file
- `-o program1` specifies the name of the compiled output file

If the compiler detects any errors or warnings, they will be reported. If there are any errors or warnings, you will have to return to the editor to make any corrections as necessary. Be careful - one forgotten semi-colon is enough to cause an error!

7. Once your editing and compiling are complete and no warnings or errors are generated, a new program file will appear in your directory (type `"ls"` to see a list of files). You can run your program in a terminal window by typing:

```
./program1
```

Homework submissions require a **log** of your program output. One way to do this is to use the **script** utility. To begin logging your output to a file, return to a terminal window and type `"script"`. The **script** program will log all the following output to a file called **typescript** by default. Run your program again and then type **exit** to end the **script** program and stop logging. To view the contents of the script file type:

```
cat typescript
```

Note that if you run **script** again it will overwrite the contents of the previous **typescript** file. To prevent the file from being overwritten, simply copy the **typescript** file to another filename using the **cp** (copy) command:

```
cp typescript part1script
```

Include a printout of the script output for this program when you hand in the assignment. You can print the script output by opening the file in `gedit` or copy and paste it into a word processing document. For more information on the **script** program type: **man script**

8. Recompile the program without the `-o program1` option. Use the `ls` command to view your directory.

Question A: What is the default name given to the output program if the `"-o"` option is not specified?

9. Edit the source file to remove the semi-colon from the `cout` statement and then recompile. Next, replace the semi-colon and remove one of the curly braces `"{"` or `"}"`.


Question B: What error messages does the compiler report due to a missing semi-colon? What error messages are reported with a missing curly brace?

10. Expand the program by adding lines to print out your name and Redeemer e-mail address. Recompile and run the program and log the output once again and submit this with your homework.

Part 2: Using the Eclipse Integrated Development Environment (IDE)

Modern software development is often done using an *Integrated Development Environment* (IDE). IDEs provide a graphical environment for editing, compiling and debugging code. In this tutorial you will use the **Eclipse** IDE.

11. Launch **Eclipse** by typing `"eclipse"` inside a terminal window or by clicking in the Eclipse icon in the menu.

12. To start coding a new program, start a new *project* by selecting **File/New/Project** from the menu. From the wizard menu under C++, select “**Managed Make C++ Project**”. Click “Next” and enter “Assignment1a” as the project name. *You should avoid project names that contain spaces or other non-alphanumeric characters!* Click “Next” and ensure that the project type is “Executable (Gnu)”. Click “Next” and Click the **C/C++ Indexer** tab, ensure “Enable C/C++ Indexing” is selected. Click “Finish” to finish setting up the project.
13. Create a new C++ source file in the project by clicking on the following icon:  A dialog box will open prompting you to enter a source file name. Enter the name “**addition.cpp**” (recall that C++ source files should have a “.cpp” extension). An editor window will appear where you may enter the following source code. Note that the editor uses syntax highlighting and that it automatically ensures that your code is indented properly.

```

/* My first CSC121 Assignment
   Name: your name
   This program sums two numbers and prints the result to the display.
*/

#include <iostream>
using namespace std;



int main()
{
    int x,y,sum;

    cout << "Enter the first number: " << endl;
    cin >> x;
    cout << "Enter the second number: " << endl;
    cin >> y;
    sum = x+y;
    cout << x << " + " << y << "is equal to " << sum << endl;
    return 0;
}

```






When you are done entering the program, be sure to click the save icon which will also *compile* your project.

Question C: Try introducing errors in your program (such as removing a semi-colon) and click “Save”. How does Eclipse help you to identify and locate errors in your program?

14. Compile and run the program by selecting **Run** from the **Run** menu. In the wizard dialog which appears, select “**C/C++ Local Application**” from the list of **Configurations**, then click the **New** icon. Click on the “Assignment1a” project and then click on “**Search Project**” to locate your executable file. Next, click on the **Debugger** tab and select the “**gdb Debugger**” from the drop-down menu. Click on “**Apply**” and then click “**Run**”. Once the run configuration has been setup properly, you can run the program again by simply clicking the run icon . Note that the output from the program will appear in a console window which appears *below* the edit window. If your program prompts user input, you will need to click in this window to enter any program input. **Hint:** the resulting output can be copied using the right mouse button and pasted into a document. This can be used to provide a copy of your runtime output which must be included as part of your homework submission.
15. Debugging is a part of the process of programming. One way to debug code is to use a special debugging tool that allows you to see what is going on inside your program while it executes. The **Eclipse** IDE uses a powerful *debugger* called **gdb** (the GNU debugger) which can be used to step through your code to find errors. Practice using the debugger with the code you wrote in the previous step. Instead of clicking the **run** icon, select **Run/Debug** or click the debug icon  and a debugging *perspective* will open in **Eclipse**. Upon starting, the debugger will launch your program and stop at the first instruction. Various windows indicate the status of the program as the debugger is run. One of the windows in the middle of the screen shows your program, highlighting the next line to be executed. Breakpoints may be set by clicking in the margin beside the line of code where you

want the debugger to stop. Another window shows the contents of any variables that have been declared. A window on the bottom shows the output of the program as it runs.

To begin debugging the program, use the following icons:

-  - Resume execution until a breakpoint is encountered
-  - “Step over” will single-step by executing the next instruction
-  - “Step into” will wingle-step into the next instruction, entering into a function if one is present
-  - Restart execution from the beginning of the program
-  - Terminate program execution

Start practicing using the debugger by using the “step over” function to single-step through your program and trace the execution line-by-line. Examine the contents of each of the *variables* as the program proceeds and runs to completion. Note that before the *variables* are initialized, they contain “garbage” values. Set some breakpoints and run the debugger again. **Hint:** When done with the debugger, you can return to the editing perspective by clicking the “Open Perspective” icon in the top right corner and select “C/C++”.

Question D: What is a variable? (Consult your textbook if necessary)

16. Create a new program that modifies the previous program so that the user is asked to enter 3 numbers (x , y , z), which are then multiplied together (note that the symbol for multiplication in C++ is ‘*’). Note that every new program will require its own project. After creating a new project, make a source file called **multiply.cpp** and enter the code. Run the new program and record the program output. Refer to Chapter 1 in your text for an introduction to variables and the **cin** and **cout** statements.

Remember: You should create a new C++ project for each new program that you write!

Question E: Using the debugger, record the values of the x , y and z variables when program execution begins and before the data is input from the keyboard. What are the initial values of these variable and where do they come from?