# Dalhousie University
# CSCI 2132 — Software Development
# Winter 2017
# Lab 5, February 16/17

In this lab, you will first learn an option of `gcc`. Next you will learn some basic commands of the tool `gdb`, which is a debugger. A program using if and for statements will be used as an example. After this, you will learn what options to use if you want `gcc` to use different C language standards. Finally, you will be asked to write a program to use a loop and if statements to solve a problem, to familiarize yourself with C statements.

Be sure to get help from teaching assistants whenever you have any questions.

1. Special note: If you have not completed Lab 4 due to the storm closure last Friday, please complete the work for Lab 4 first. Please also note that, due to the storm days, control structures (`if`, `switch`, `for`, `while` statements) have not been covered by Wednesday. However, as mentioned in class, the syntax for these statements is the same as that in Java, so you can still complete the work for this lab without attending this Friday's lecture first.

2. First, perform the following steps to get started:

   (i) Login to server bluenose.cs.dal.ca via SSH from a CS Teaching Lab computer or from your own computer.

   (ii) Change your current working directory to the `csci2132` directory created in Lab 1.

   (iii) Create a subdirectory named lab5.

   (iv) Change your current working directory to this new directory.

3. We first learn one useful option of `gcc`. Copy the following C source file to your current working directory.

   `/users/faculty/prof2132/public/hello.c`

   Edit this source file by removing the line that contains `return 0;`. Compile it. This will still work and you will not see any warning message regarding this.

   We then use the following command to compile:

   ```
   gcc -Wall -o hello hello.c
   ```

This -Wall option enables all the warnings messages about constructions that some users consider questionable, and that are easy to avoid. Here we will see a warning message about the missing return statement. Fix your program to avoid the warning.

It is often a good idea to use the -Wall option, and edit your program to eliminate the warning messages, since sometimes these warning messages are caused by errors in your programs.

4. Debugging is a methodical process of finding and reducing the number of bugs, or defects, in a computer program. Thus it is important in software development. Later in a lecture, we will discuss more about debugging and testing strategies. In this lab, we focus on learning the very basic commands of a debugger called `GDB`, the GNU project debugger, which is the standard debugger for the GNU software system.

   To use `gdb` to debug a C program, we need use a special option when we compile this program using `gcc`, which is the `-g` option. Copy the following file into your current working directory:

   ```
   /users/faculty/prof2132/public/numbers.c
   ```

   Compile it using the following command:

   ```
   gcc -g -o numbers numbers.c
   ```

   Now the executable file `numbers` is ready to be debugged by `gdb`.

5. To run `gdb`, we simply enter `gdb program_name`.

   Try the command `gdb numbers`. After you run this command, you will see the command-line interface of `gdb`.

   You can run the program `numbers` inside gdb. To do this, enter the `gdb` command `run`, or simply `r`. The program will run inside `gdb`, and prompts you to enter a positive integer. Enter 8. After that, `gdb` will finish running `numbers`, and you will see a line of text saying "exited normally".

6. The main approach of debugging with `gdb` is to set breakpoints, which are intentional pausing places in a program. When you set a break point at a certain line in `gdb` and then run this program using `gdb`, the program will pause when this line is executed. You can then use `gdb` commands to print the current values of variables, to check if their values are correct. If there is a bug in the program, you can keep on doing this until you find the line of code that causes the bug. This will help you locate the error.

   The syntax of the `gdb` command that sets a breakpoint is `break linenumber`, which sets a break point at a certain line number. You can find out the number of any line by looking at the bottom status bar of the emacs screen, which gives you the line number of the line that has your cursor.

In this lab, we will not debug an incorrect program. Instead, we will run a correct program using `gdb` to learn how to set breakpoints, to print values of variables and to execute a program line by line. This will help you learn `gdb` commands. You can then start using `gdb` to debug your own code after this.

Assume that you have not quit `gdb`. If you have, repeat Question 4.

Perform the following steps:

(a) Set a breakpoint at line 8 by entering `break 8` or `b 8`.

(b) Set a breakpoint at line 20.

(c) Run the program under `gdb` by entering `run` or `r`.

(d) Print the value of the variable `value` when the debugger stops at the first breakpoint. You can use the command `print value` or `p value`.

(e) Execute the next two lines by entering the `gdb` command `step` or `s` twice.

(f) When the program asks you to input a positive integer, enter `7`.

(g) Enter a `gdb` command to check whether the value of the variable `k` has been successfully read from stdin.

(h) Continue the execution of the program until it stops at the next breakpoint by entering `continue`, `cont` or `c`.

(i) Print the value of the variable `value`.

(j) Repeat the previous two steps until the program finishes execution.

(k) Quit the debugger by entering `quit` or `q`.

There are more `gdb` commands than those given above. You can type `help` in the `gdb` command line to see them. They are nicely organized into categories. Try this.

7. Record all the values of variable `value` that you have printed during the above process, in the order that they were printed. Then, enter the sequence of integers at the On-line Encyclopedia of Integer Sequences: `http://oeis.org/`

   What is the name of the integer sequence that you find? From this, can you tell what the program computes?

   Now read your program again to understand it. Treat it as an example of using if and for statements in C to familiarize yourself with C syntax.

8. There is one minor detail about for loops in C. In C99, we can declare the loop iterator in the for statement. For example, in the `numbers.c` program, remove i from the list of variables in the first line inside the main function body. Then, change the first line of for loop to:

```
for (int i = 1; i < k; i++) {
```

This modification will make the variable i usable in this for loop only.

This is supported in C99, but not in C89 or older, which requires declarations to precede all the statements in any function body.

However, when we compile the modified program using `gcc` commands that we learned before, we will get an error. Try this to see the error message.

Why did we get such an error message? This is because by default, the `gcc` C compiler uses a C language standard called gnu89, which is a GNU dialect of the C language. It contains some of the C99 features only.

How can we compile a C program with C99 features that are not in gnu89 then? To do so, use the `-std=c99` option. Enter the following command:

```
gcc -std=c99 -o numbers numbers.c
```

Try this. You can also use other options such as -g and -Wall at the same time, and the order in which `gcc` options appear does not matter.

For any assignment that gives you the command that we will use to compile your program, make sure that your program will compile using that command.

9. Now, work on the programming project 8 on page 123 of the C textbook. This question asks you to write a program that prints a one-month calendar. The program asks the user to input how many days there are in this month, and what is the day of the first day in this month (1 means Sunday, 2 means Monday, etc.). It then prints a calendar in the following format (assume that this month has 30 days and the first day is a Tuesday):

```
        1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

   See the textbook for hints. The solution is also available on the textbook's website, but do not simply read the solution: work on it on your own first.