# WEEK 1 - BASIC C PROGRAMMING

In this course, we will use **Code::Block** for developing your programs. Please refer to the uploaded documents on “Setting up Code Blocks and MINGW” [http://www.cprogramming.com/code_blocks/] and Creating a New Project in C [http://www.dummies.com/programming/c/how-to-create-a-new-codeblocks-project-in-c/] for the setting up of Code::Blocks and creation of your first program. Please refer to online resources for additional materials on using Code::Blocks.

You are required to do the following:

1. Lab Questions – please do the lab questions during the lab session.
2. Assignment Questions – please do the assignment questions online using the automated programming assignment submission system (APAS) (see Appendix A). You do not need to submit your code for grading for this assignment.

**Lab Tutor**: For this lab-tutorial session, please discuss about the solution for each question in the lab. You may allocate about 30 minutes for each question. No need to discuss about the assignment questions.

## Lab Questions

**1.** **(temperature)** Write a C program that reads the user input on temperature in degrees Fahrenheit, and then converts the temperature from degrees Fahrenheit into degrees Celsius. The relevant formula is given as follows: Celsius = (5/9)*(Fahrenheit – 32).

Sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the temperature in degree F:
45
Converted degree in C: 7.22
```

(2) Test Case 2:
```
Enter the temperature in degree F:
-12
Converted degree in C: -24.44
```

A program template is given below.

```
#include <stdio.h>
int main()
{
   float fahrenheit, celsius;  // declare variables

   printf("Enter the temperature in degree F: \n");
   scanf("%f", &fahrenheit);

    /* Write your program code here */

   printf("Converted degree in C: %.2f\n", celsius);
   return 0;
}
```

**2.** **(linearSystem)** Write a C program that computes the solutions for x and y in the linear system of equations:

$$a_1x + b_1y = c_1$$
$$a_2x + b_2y = c_2$$

The solutions for x and y are given by:

$$x = \frac{b_2c_1 - b_1c_2}{a_1b_2 - a_2b_1} \qquad \text{and} \qquad y = \frac{a_1c_2 - a_2c_1}{a_1b_2 - a_2b_1}$$

The program reads in $a_1$, $b_1$, $c_1$, $a_2$, $b_2$ and $c_2$, and then computes and prints the solutions. In your program, you may assume that the denominator ($a_1b_2$ - $a_2b_1$) of the above equations is not zero. Therefore, there is no need to check whether the denominator is zero or not.

Sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the values for a1, b1, c1, a2, b2, c2:
1 1 1 5 7 9
x = -1.00 and y = 2.00
```

(2) Test Case 2:
```
Enter the values for a1, b1, c1, a2, b2, c2:
1 1 2 2 3 3
x = 3.00 and y = -1.00
```

A program template is given below.

```
#include <stdio.h>
#include <math.h>
int main()
{
   /* Write your program code here */
   return 0;
}
```

## PROGRAM DEBUGGER

In addition, you will learn how to use Program Debugger in this lab. To develop a program to solve a problem, the program development process generally consists of 6 steps. They are Problem Definition, Problem Analysis, Program Design, Implementation, Program Testing and Documentation.

- Step 1: Problem Definition. This step determines the objective of the program, and writes a problem statement or paragraph describing the purpose of the program. The problem statement is a broad statement of the requirements of the program, in user terms.
- Step 2: Problem Analysis. This step analyzes the problem and produces a set of clear statements about the way the program is to work. This requires a clear understanding of the underlying concepts and principles of the problem. These statements define how the user uses the program (*program input*), what output the program will generate (*program output*), and the *functionality* of the program. The functionality may be expressed in terms of mathematical formulas or equations for specifying the transformation from input to output.
- Step 3: Program Design. This step is to formulate the program logic or *algorithm*. An algorithm is a series of actions in a specific order for solving a problem. There are two basic methods that can be used to design the program logic: *pseudocode* and *flowcharts*. The technique that is used for developing a program is the top-down stepwise refinement technique.
- Step 4: Implementation. This step is to convert the program logic into C statements forming the program. If the program has been designed properly, then it is a straightforward task to map the program design into the corresponding C code.
- Step 5: Program Testing. This step is to test the program code by running the program. This aims to determine whether the program does carry out the intended functionality. Program testing involves the use of test data that the correct answers are known beforehand, and the use of different test data to test the different computational paths of a program. Therefore, it is

important to design test cases such that all conditions that can occur in program inputs are tested. However, it is also necessary to ensure that the tests are not too exhaustive.
- Step 6: Documentation. This step is to document the programs. Documentation of computer code is useful for the understanding of the program's design and logic. This is important for the maintenance and future modification of the programs. In addition, documentation such as user manuals can also help users to understand on how to use the program.

Programs are bound to contain errors when they are first written. There are mainly three types of programming errors: *syntax errors*, *run-time errors* and *logic errors*.

- Syntax errors. These errors are due to violations of syntax rules of the programming language. They are detected by the compiler during the program compilation process. They are also called compilation errors. The compiler will generate diagnostic error messages to inform the programmer about the locations in the program where the errors have occurred. There are two types of diagnostic messages: warning diagnostic messages and error diagnostic messages. A warning diagnostic message indicates a minor error that might cause problems in program execution. However, the compilation is not terminated. Error diagnostic messages are serious syntax errors, which stop the compilation process. The nature and locations of the errors are given in the messages. If the programmer cannot locate the error according to the location given by the compiler error message, then the programmer should also look at the statements preceding the stated error. If this is unsuccessful, the programmer should also check all the statements related to the stated error statement. Examples of this type of errors include illegal variable names, unmatched parentheses, undefined variable names, etc.
- Run-time errors. These errors are detected during the execution of the program. They are caused by incorrect instructions that perform illegal operations on the computer. Examples of such illegal operations include division by zero, storing an inappropriate data value, etc. When run-time errors are detected, run-time error messages are generated and program execution is terminated.
- Logic errors. These errors occur due to incorrect design of the algorithm to the problem. Such errors are usually difficult to detect and correct since no error messages are given out. Debugging of logic errors requires a thorough review of the program development process on problem analysis, algorithm design and implementation. Tracing of program logic and testing of individual modules of the program are some of the techniques, which can be used to fix logic errors.

Since errors can occur during program development, program debugging is necessary to locate and correct errors. A number of techniques are available for program debugging. The most common approach is to trace the progress of the program. This approach is especially useful for debugging *logic errors*. Other techniques such as isolation of sections of program code and hand simulation can also be used to debug program errors.

Program tracing is to write code during program development to produce extra printout statements to monitor the progress of a program during program execution. This can give us an idea how the program works during the course of execution. Intermediate results can be printed and we can check to see whether the results are the intended ones. It is quite frustrating when a program runs but nothing is on the screen. The program could be running in a loop repeatedly, or it could be just waiting for user input. If printout statements are displayed on the screen, we should then know what the program is currently doing.

To do this, debugging printout statements such as **printf()** may be placed at several strategic locations inside the program. They can be used to print the input data read from the user, or to print computation results at different stages during program execution. Debugging printout statements are also very useful for printing data values inside a loop to show changes of these values in the loop. In addition, printout statements may also be inserted at the beginning of each function, so that the exact execution path of the program can be traced.

Apart from tracing the program, program code can also be separated into different sections, and then each section can be tested separately on its correctness. Once the errors are isolated, you may correct the errors or rewrite the code for the errorneous section. Hand simulation can also be used by pretending that we run the program in sequence on the computer. All the values for variables and program outputs are recorded. However, this method is only suitable for small programs.

In addition, program debugger can also enable programmers to inspect a program during its execution. A debugger can run with a program. We can interact with the debugger to interpret the program instructions in a step-by-step manner. We may also see the results of computations, and monitors the progress of the program. As each compiler's debugger is different, it is necessary to consult the compiler's documentation on how to use it. Learning how to use the debugger will help to speed up your program development process. Debugger is especially useful for debugging programs using data structures such as linked lists, stacks, queues anfd queues. Some important features and functions of debuggers are listed below:

- Setting breakpoints and stepping program pxecution. Breakpoints are the places where you want a program execution to stop and provide you a chance to enter debugger instructions, for example, examine the values of some variables or step the program execution. There are two ways of stepping a target program execution. Stepping means advancing target program execution one source code statement at a time.
- Examining Program Variables. You may also inspect the values of program variables including pointer variables.
- The More You Explore, the More You Find. For this lab, you just need to learn on how to use the debugger.

**3. Using Program Debugger**

In this question, the program code is given below (please refer to the uploaded document "Using C Debugger" [http://www.dummies.com/programming/c/how-to-use-the-codeblocks-debugger-with-c-programming/]. Create a source file called **Q3.c** to save the program code, and then compile and execute the program. Correct the error in the code.

```
#include <stdio.h>
int main()
{
  char loop;
  puts("Presenting the alphabet:");
  for (loop='A';loop<='Z';loop++);
    putchar(loop);
  return 0;
}
```

Follow the steps stated in the document to learn how to use the debugger in Code::Blocks to identify the error in the code.

## Assignment Questions

You may submit your code to the Automated Programming Submission and Assessment System (APAS) for testing.

**1. (power)** Write a C program that reads the user input on the current and resistance, and displays the power loss of the cable on the screen. The relevant formula is $P = I^2R$, where P is the power loss in watts, I is the current in amperes and R is the resistance in ohms.

Sample input and output sessions are given below:

(1) Test Case 1:
```
Enter the current:
12
Enter the resistance:
1.5
The power loss: 216.00
```

(2) Test Case 2:

```
Enter the current:
5
Enter the resistance:
0.5
The power loss: 12.50
```

A program template is given below.

```
#include <stdio.h>
int main()
{
   /* Write your program code here */
   return 0;
}
```

**2.** **(cylinder)** Write a C program that computes the volume and surface area of a cylinder. The program reads the user input on the radius and height of the cylinder, and then computes the volume and surface area of the cylinder. The relevant formulas are volume = $\pi r^2 h$ and surface area = $2\pi rh + 2\pi r^2$, where r is the radius and h is the height.

Sample input and output sessions are given below:

(1) Test Case 1:

```
Enter the radius:
5
Enter the height:
8
The volume is: 628.32
The surface area is: 408.41
```

(2) Test Case 2:

```
Enter the radius:
1
Enter the height:
5
The volume is: 15.71
The surface area is: 37.70
```

A program template is given below.

```
#include <stdio.h>
#define PI 3.1416
int main()
{
   /* Write your program code here */
   return 0;
}
```

**3.** **(speed)** Write a C program to read the time (in seconds) and distance (in kilometers) covered by a moving object, calculate the speed in kilometers per second, and display the speed on the screen. The relevant formula is: speed = distance/time. The output should be in the following format: "`The speed is [the speed value] km/sec.`"

Sample input and output sessions are given below:

(1) Test Case 1:

```
Enter distance (in km):
1
Enter time (in sec):
10
The speed is 0.10 km/sec
```

(2) Test Case 2:
```
Enter distance (in km):
10
Enter time (in sec):
5
The speed is 2.00 km/sec
```

A program template is given below.

```
#include <stdio.h>
int main()
{
   /* Write your program code here */
   return 0;
}
```

**4.** **(distance)** The distance between two points ($x_1$, $y_1$) and ($x_2$, $y_2$) is given by

$$\text{distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Write a C program that reads the (x, y) coordinates for two points, computes the distance between the two points.

Sample input and output sessions are given below:

(1) Test Case 1:
```
Enter first point x1 y1:
1 5
Enter second point x2 y2:
2 7
The distance is 2.24
```

(2) Test Case 2:
```
Enter first point x1 y1:
1 1
Enter second point x2 y2:
5 5
The distance is 5.66
```

A program template is given below.

```
#include <stdio.h>
#include <math.h>
int main()
{
   /* Write your program code here */
   return 0;
}
```

## Appendix A - Automated Programming Submission and Assessment System (APAS)

To use the online Automated Programming Submission and Assessment System (APAS), please follow the following steps:

- Open the Chrome web browser, and type in the following link (to be provided during lecture and lab).
- Enter your student account (Network Account) in **uppercase**, and password (will be given during the lab session).

- The user menu will be displayed and you may then use the system according to the user menu.
- You may use Code::Blocks to edit, compile and run your code for each of the questions. For online submission, the **template file** is provided in the system. Please do not change the contents in the main() function. Just copy and paste your own code in the missing code location <`/* Write your program code here */`> in the program template.
- After entering the code, you may:
  1. Test the code with sample input – the system will compile your code, and run your code against the sample test cases.
  2. Try compilation – you may compile the source code to see whether there are any compilation errors in your code.
  3. Run Input – you may run the source code with your input data in the system. To do this, you will need to type in all the input data into the Data Input Box, and after program execution, you will see all the output data in the Data Output Box.
- **Save** the code of a question whenever you have successfully solved the question and submit the code when you have completed all the questions.
- The APAS will conduct automated checking on your source code by using pre-defined test cases.

The link and the related information about the APAS system will be given in your lab session.