A guide for "Fundamental Exercise on Computer and Information Engineering 1B"

Masaomi Kimura

April 9, 2015

Chapter 1

Introduction

This guide is a material for the class "Fundamental Exercise on Computer and Information Engineering 1B". This class is aimed at 2nd grade level.

We assume that all students in this class have learned basics of C language. Based on the knowledge and the skill, you are expected to be able to design and implement

any program

you want.

However,

I have heard that there are not a small number of students who feel some difficulty to write programs by themselves. In fact, I see some 3rd grade students have a hard time to create programs.

The aim of this class is to provide the chance to have experiences to create practical programs and to develop skills to have a good command of C language. Especially, if you have a motivation,

"I want to understand how to create a program!"

and/or

"I found that I need knowledge to create this program...",

this exercise should be suitable for you.

The agenda of this exercise class is as follows:

- C language brush-ups (Session 1 to Session 3),
- Creation of a schedule management application program (Session 4 to Session 7),
- A simulation program and an image processing program (Session 8 to Session 11),
- A game program (Session 12 to Session 15).

Now it is the time to get it started.

We expect you to access linux server via a terminal emulator called "teraterm" and to compile your program by means of gcc command. Use emacs to make a source program.

Again, the aim of this course is to establish your basic programming skills. Moreover, for the students who are familiar with C language programming, we will provide advance exercises.

Please be sure that drinking and eating is prohibited in PC class room.

The students who are absent more than 5 times will not get credit for this class.

Chapter 2

C language brush-up (1), Basics in basics

In this chapter, let us review C languge grammer that you should be familiar with. I can hear your voice, "hey, practical program?". Be patient. Basic is Basic. It is the first hurdle you must clear.

Remember that a program should both input something and output other. Let us review standard input via a keyboard, standard output via a monitor, the way to deal variables and arrays, and repetition statements.

Create programs that satisfy the following requirements.

Until next session, complete programs listed in Section2.1 and Section2.2. You are required to present your source codes and their output as a report. We collect your reports at the starting time of Session 2 [1]. Section 2.3 is for advance learners. Inclusion of its answers in the report adds additional score. Please be sure to write a minuites paper about what is your weak point about C language and questions related to this class.

2.1 Standard Input/Output = keyboard input / Display output

1. Create a program that displays "1 plus 1 makes 2". (The name of the source program should be prog1-1.c)

[1]The report should be in A4 paper size. The first page should have only a title, your student number and your name. Your report should be bind with a staple at its left top corner.

- Use the function, **printf**'. This is the simplest program of C language and is a start point in this class.Do not forget to include the header file "stdio.h". Hereafter we will not mention this, but recommend to include the header file if needed. [2]

2. Prepare int-type variables, i and j. Use **scanf** twice to read keyboard input and to insert the read data into i and j. Display "A plus B makes C", where A has the value of i, B has the value of j and C is their sum. (Name the source file as prog1-2.c)

   - Remember that we use not the variable i but its reference $\&i$ in order to read an input value by means of **scanf**.

3. Prepare int-type variables, i and j. Use **scanf** twice to read keyboard input and to insert the read data into i and j. Display "A divided by B equals C", where A has the value of i, B has the value of j and C is the quotient. (Name the source file as prog1-3.c)

   - For example, if $i = 3$ and $j = 2$, then the program should display "3 divided by 2 equals 1.5". In the case that $j = 0$, the program should display "Cannot divide".

2.2 Arrays and repetition

1. Prepare int-type variables, i and j. Use **scanf** twice to read keyboard input and to insert the read data into i and j. If i or j are less or equal to zero, display "Please input positive integer values". If i and j are positive, display a rectangle obtained by repetition of the letter #, vertically i times and horizontally j times. (Name the source file as prog1-4.c)

   - For example, in the case that 8 and 6 were input as i and j, display

```
######
######
######
######
######
```

[2] *stdio* is an abbreviation of "standard io".

```
######
######
######
```

2. Prepare an int-type variable, i. Use **scanf** to read keyboard input and to insert the read data into i. If i is less or equal to zero, display "Please input positive integer values". If i is positive, display a right triangle obtained by repetition of the letter #, whose legs are i. (Name the source file as prog1-5.c)

   - In the case that 6 was input as i, display

   ```
   ######
   #####
   ####
   ###
   ##
   #
   ```

3. Create a program to read 10 integer values input from keyboard and to display their total sum. (Name the source file as prog1-6.c)

   - Use an array a[ ], which has 10 elements. Use **scanf** to read keyboard input.

2.3 A challenge for advanced users

- Improve the program prog1-2.c to read integer values not by **scanf** but by runtime arguments of **main** function. (Name the source file as prog1-2a.c)

  – Let prog1-2a be an executable file of the program prog1-2a.c. If we execute

  ```
  ./prog1-2a  10  7
  ```

  then the program should display "10 plus 7 makes 17".

  – Remember that you need to use **main** function written as

```
int main(int argc, char[][] argv){
          ...
}
```

- The program prog1-6.c assumed the number of input integers was 10. Improve it to let a user specify the number of input integers as an execution argument. (Name the source file as prog1-6a.c)
  - Let prog1-6a be an executable file of the program prog1-6a.c. If we execute

    ```
    ./prog1-6a  16
    ```

    the program should read 16 integers and display their sum.
- Create a program to calculate the expression in Reversed Polish Notation given as an execution argument (Name the source file as prog1-rpol.c)
  - Let prog1-rpol be an executable file of the program prog1-rpol.c. If we execute

    ```
    ./prog1-pol  4 5 + 2 3 + -
    ```

    the program should return the result of (4+5)-(2+3) in normal notation. Let us assume that there are space letters between numbers and symbols (like + and -).