Programming and Data Structure Laboratory

Problem Solving using C

Sourav Sen Gupta
sg.sourav@gmail.com

Indian Statistical Institute

Programming Environment

Login to the CSSC Matlab server, where `CS15xx` is your roll number:

`ssh mtc15xx@192.168.54.156`	if you use *vi* or *emacs*
`ssh -X mtc15xx@192.168.54.156`	if you use *gedit*

Set up the basic programming environment for today (Day 5):

`cd`	go to your home directory
`mkdir -p pdslab/day5`	create a directory for today
`cd pdslab/day5`	go to the directory for today

Create the classwork programs for today (e.g., in *gedit*):

`gedit cs15xx-day5-progy.c`	where `y` = program number

Programming Environment

First few lines of *any* classwork or assignment program:

```
/*-------------------------------------
Name:
Roll number:
Date:
Program description:
Acknowledgements:
-------------------------------------*/
```

Compiling and running your program:

```
gcc -g -Wall -o progy cs15xx-day5-progy.c
./progy
```

Solving a Problem

1. Clearly understand and *define* the given problem in own words.
2. Take a few examples, if necessary, to clarify the problem better.
3. Create an *implementation-independent* solution first!
4. Look at the problem from *every* possible point-of-view.
5. Try to find similarities with other problems, and devise a strategy.
6. Identify sub-problems and choose appropriate data structure(s).
7. Identify iterative and conditional relationships to connect pieces.

"the sooner you start coding, the longer it is going to take"

Problems to Solve

Problem 1

Write a program to determine the ranges of the generic C data types: `char`, `int`, `float`, `double` – both for unsigned and signed cases. Consider `short` and `long` data types too, wherever appropriate.

Problem 2

Swap the values of two variables. How many ways of swapping do you know? For each method of swap, determine the pros and cons.

Problems to Solve

Problem 3

Take n integers $\{a_i\}$ from the user, and compute $\sum_{i=1}^{n} a_i$. How long does it take with naive repeated additions? Can you do any better?

Problem 4

Take a positive integer x from the user, and compute the value of $x!$. How long does it take with naive multiplications? Can you do any better? Can you compute for arbitrary sized inputs?

Problems to Solve

Problem 5

Take positive integers x, n from the user, and compute x^n. How long does it take with naive multiplications? Can you do any better?

Problem 6

Take three positive integers – x, n, m – from the user, and compute the value of $x^n \bmod m$. Can you compute for arbitrary sized inputs?

Problem 7

Take two positive integers – x, p – from the user, and compute an approximate value of $\sqrt{x}$, correct upto p bits after the decimal point.

Problem 8

Special triangles with integer sides and integer area are called *Heron triangles*. Take three integers – a, b, c – from the user, and determine if they form the sides of a Heron triangle. Do the same when the user inputs three vertices of the triangle – (x_1, y_1), (x_2, y_2), (x_3, y_3).

Problem 9

Take two positive integers – x, p – from the user, and compute an approximate value of e^x, correct upto p bits after the decimal point.

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Problem 10

Compute the ratio between two successive Fibonacci numbers $\frac{F_{n+1}}{F_n}$ in a limiting sense (as $n \to \infty$), correct upto 20 bits after decimal point.

Problem 11

Write a program that prints numbers in LCD display style.

Input: a digit $(0 - 9)$, and size s

Output: Print the digit in an LCD style, using s '-' signs for each horizontal segment and s '|' signs for each vertical one.

Each digit should occupy $s + 2$ columns and $2s + 3$ rows.

Example: some digits for $s = 2$

```
 --   --
|  |    |  |  |    |
|  |    |  |  |    |
 --   --   --
|  |  |         |    |
|  |  |         |    |
 --   --
```

Problem 12

Take two points from the user – (x_1, y_1), (x_2, y_2) – and construct a path joining them together. Run a loop to take n more points from the user, one at a time, and determine whether there was a *left turn*, a *right turn* or a *straight walk* in relation to the previous two points.

Input: $(0, 0)$ – $(2, 2)$ – $(3, 2)$ – $(4, 2)$ – $(5, 1)$ – $(6, 4)$

Output: n.a. – n.a. – Right – Straight – Right – Left