Mathematics 1261 (C Programming and computing) Michaelmas 2013

December 8, 2013

1 First assignment, due 4/10/13

Write a C program to print out the following (**EXACTLY**). The string contains double quotes, which need special treatment (see below). The single quotes (there are two kinds) can be typed as they appear below.

```
"Good morning, madam," to Eve said Adam,
‘Good morning, sir,’ to him said she.
```

The double quotes need special treatment, because they interfere the double quotes surrounding the string. They should be ‘escaped,’ with a backslash:

```
printf("\"Good morning, madam,\" .....
```

Edit, compile, and run your program to make sure it works. When it works correctly, type in

```
submit-work 1261:1
```

The submit-work program will help you to submit your completed program, call it hello.c or whatever. The `1261:1` means the first assignment in course 1261.

Although this almost the simplest assignment possible, you should **INDENT** your program correctly, that is, arrange for it to be readable. The style is:

```
#include <stdio.h>   /* at start of line  */

main ()              /* at start of line */
{                    /* at start of line, nothing else on line */
  program begins here
                     /* indent two places, for readability */

  program ends here  /* indent two places, for readability */
}                    /* at start of line */
```

Motivation: compiling, running, and submitting, a C program. Although the words are not serious, there is some point in repeating them *exactly*: some marks may be deducted for deviation. The same goes for indentation.

2 Second assignment, due 11/10/13

Write a program to read integers from the command line, and print their number and their sum. It should produce output like this:

```
%a.out 2 -3 4 5 -6 7 8
7 numbers were read. The total is 17.
```

Motivation: For-loops, command-line arguments, `atoi()`.

3 Third assignment, due 18/10/13

Write a program to read *double-precision* values from the command line, and to calculate their average and standard deviation.

You can use `atof()` to convert command-line arguments into double-precision format.

If one declares a variable n and sets it equal to `argc-1`, then n is the number of values to be read.

The average

$$\overline{x} = \frac{\sum_i x_i}{n}$$

should be calculated along with

$$s = \sum_i x_i^2$$

and the following formulae for variance v and standard deviation σ used:

$$v = \frac{(\sum_i x_i^2) - n * \overline{x}^2}{n-1}$$
$$\sigma = \sqrt{v}$$

The C `sqrt` function has that name. Thus

```
sdev = sqrt ( v )
```

will do the trick, if one include

```
#include <math.h>
```

However `gcc` needs more information to locate the sqrt function:

```
gcc -lm myprog.c
```

(The -lm, with an 'ell' not a '1', means 'use maths library.')

```
EXAMPLE:
a.out 3.80 4.63 3.23 4.33 3.70
5 numbers, average 3.938000, standard deviation 0.549882
```

Motivation: Using doubles and atof(), printing double-precision values, applying the formulae for mean and standard deviation.

4 Fourth assignment, due 25/10/13

The assignment, like the previous one, is to calculate the mean and standard deviation of a list of numbers.

There are three differences.

- First, the numbers will be stored in an array of **double**s. Declare the array as follows.

  ```
  double x[1000];
  ```

  (That's big enough for the test data).

- The numbers are read using `scanf()`. A file `big` will be made available for testing.

- Second, you must use a 'while' loop and use the return value of `scanf()` to count the numbers read in.

  Be careful to use the format type for doubles, not floats.

- Third, you must compute two different versions of the variance. The first is as in Assignment 3. The second is to use the direct formula

$$\frac{1}{n-1}\sum_{i=0}^{n-1}(x_i-\overline{x})^2.$$

- Print the number n of numbers read in, the average $\overline{x}$, and the two different estimates of variance and standard deviation.

- The point is that last week's method can be inaccurate.

Motivation. Storing data in arrays and operating on arrays. Correct usage of `scanf()`. Usage of a while-statement.

5 Fifth assignment, due 15/11/13

Two matrices are stored in a data file `data/mats`, as follows:

```
m_1 n_1
a[0][0] ..... a[m_1][n_1]
m_2 n_2
a[0][0] ..... a[m_2][n_2]
```

Your program should read in two matrices (you may assume that the dimensions are at most 10×10), using `scanf()` from standard input. If $n_1 \neq m_2$, your program should stop with an error message. Otherwise it should compute and print the matrix product.

In printing the product matrix, use `%8g` format — this will handle integers gracefully.

Motivation. Practice with matrix operations.

6 Sixth assignment, due 22/11/13

Refer to the incomplete program **gausspp.template** stored on the web page.

The assignment is to add some code to convert the template into a fully-running program. Less than a dozen lines of code need be added. You must complete

```
void back_substitute ( int n, double u[10][10], double y[10], double x[10])
```

and add input statements to the main routine so that the program works. There are data files to test it: `aug2x3, aug3x4,` and `aug5x6`. For example,

```
aug2x3
2 3
  2 -2 -2
  1 -2 -4
0
sample output
Input matrix
    2.00000   -2.00000     -2.00000
    1.00000   -2.00000     -4.00000
Reduced matrix
    2.00000   -2.00000     -2.00000
    0.00000   -1.00000     -3.00000
Answer (transposed)  2.000000 3.000000
```

Motivation. Writing subroutines, and using them; further linear algebra computation.

7 Seventh assignment, due 29/11/13

Write a program which includes two routines

```
void del_nl ( char * s )
        as done in class
void copy_reversed ( char * a, char * b )
        for example, if a holds "buffer" then b
        should contain "reffub".
```

The main program should read lines of text from input, delete the newline chars, make a reversed copy and print that copy. For example,

```
input...............

We know that you highly esteem the kind of Learning taught in those
Colleges, and that the Maintenance of our young Men, while with
```

```
output..............

esoht ni thguat gninraeL fo dnik eht meetse ylhgih uoy taht wonk eW
htiw elihw ,neM gnuoy ruo fo ecnanetniaM eht taht dna ,segelloC
```

Motivation. Programming with textual data.

8 Eighth assignment, due 6/12/13

Refer to the incomplete program **matrix.template** stored on the web page. The assignment is to write the functions `zero_matrix()` and `zero_vector()` and the routine `back_substitute()`. This is a variant of assignment 6. Test it on the same data files as that assignment.

Motivation. Dynamic storage allocation and the usage of structured data-types `MATRIX, VECTOR`.

9 Ninth assignment, due 13/12/13

The assignment is an easy modification of assignment 8. The difference is in file usage.

- Use named input and output files, named in `argv[1]` and `argv[2]`.
- Remember that the output file must be closed at the end of the program.
- Use multi-file compilation, with a file `ninth.c` containing the main program and a file `matrix9.c` containing the other routines.
- Files `matrix9.template`, `ninth.template`, and `matrix9.h` are on the web.

Motivation. Using files.