C Programming, Autumn 2013, Exercises for the First Week

Introduction

In the first exercise we learn to compile programs, to write simple programs with various input and output, and to use types and control statements *if, for, while.*

1. Suppose that a program is in the file prog.c. To compile and link it, use command

   ```
   gcc prog.c
   ```

   The compiled result is in the file *a.out*. To run it, just start it by *./a.out*.

   Usually we want the compiler to check more. One possibility is to use an option *pedantic*:

   ```
   gcc -pedantic prog.c
   ```

   The other option is to *Wall*:

   ```
   gcc -Wall prog.c
   ```

   Remember the format of the printf command:

   ```
   printf( string, expr_1, expr_2, ...);
   ```

   For example:

   ```
   printf("%d",123);
   ```

   Here "d"determines the type of the output, in this case it is a signed decimal number. Other possibilities:

   - d, i: signed decimal
   - ld: long decimal
   - u: unsigned decimal
   - o: unsigned octal
   - x, X: unsigned hexadecimal
   - f: floating point in the form [−] ddd.ddd
   - e: floating point [−] d.ddddde(sign)dd
   - E: floating point [−] d.dddddE(sign)dd

- g: shorter of f and e
- G: shorter of f and E
- c: character
- s: character string

It is also possible the define the width of the field where the output is printed:

```
printf("%5d",i);
```

prints the value of i into a field of five digits. Furthermore, you can define the length of the decimal part and the position of the output. See the following examples by making a command line C program and compiling and running it:

a) `printf("%6d, %4d", 86, 1040);`

b) `printf("%12.5e", 30.253);`

c) `printf("%.4f", 83.162);`

d) `printf("%-6.2g", .0000009979);`

Then fill out the function in the program for the first exercise the NetBeans IDE that should print "Hello World! It's now year <insert current year here> ", and a line break afterwards.

2. Make a program that prints the following numbers and in the way it is explained. One number per one line (a linebreak is assumed after each printed number)

   a) 12345.6789 - The total length of the field is 12 and only the integer part and two decimals are printed. The number should be on the left side of the field.

   b) 100.00000002 - the total length of the field is 15, the number should be in the exponential form, and the number is on the right side of the field.

   c) 32 - Print the number in an hexadecimal and octal form separated by a space.

3. You can read characters or numbers from a keyboard using the function scanf(). For example,

```
int i, j;
scanf("%d %d", &i, &j);
```

   reads two integers into i and j. The function returns the number of items that have been successfully read, and EOF if no items have been read and end-of-file has been encountered.

   a) Make a program that reads two integers separated by a space from a keyboard and prints them onto the screen followed by a line break.

b) Make a program that reads five characters separated by a space from a keyboard and prints them onto the screen followed by a line break.

4. It is possible to read numbers and characters from a file using fscanf(). First it is necessary to open a file. For example

```
FILE * f;
double x, y, z;

if ((f = fopen("test/TEST1.txt", "r")) == NULL) {
    fprintf(stderr, "can't read %s\n", "TEST1.txt");
{
```

   Note that error messages are printed to to standard error stream `stderr`, instead of the standard out tream, `stdout`. This is the preferred method, because the standard output stream may have been redirected, which would result in any output written to it having been redirected also.

   Now let's read three decimal numbers from the file:

```
if (fscanf(f, "%lf%lf%lf", &x, &y, &z) != 3) {
    fprintf(stderr, "File read failed\n");
    return EXIT_FAILURE;
}
```

   Suppose the file TEST1.txt contains an integer, a decimal number (read in %lf format), an integer in the hexadecimal form, and a character. These entries in the file are separated by a space. Make a program that reads all those items with fscanf and prints them onto the screen followed by a line break.

5. Write the following functions. You may need the Math library. Required libraries are already included within exercise template.

   a) `double sum_of_absolutes(double x, double y)` that returns the sum of the absolute values of x and y.

   b) `int sum_of_rounded(double x, double y)` that returns the sum of the rounded values. The variables x and y are rounded up to the nearest integer before the sum is calculated.

   c) `int sum_of_characters(char a, char b)` that returns the sum of two characters. (C is not strongly typed, so you can use various types in arithmetical expressions.)

6. Write a recursive function that returns the greatest common divisor of two integers. Use the Euclidean algorithm:

```
Parameters:  Two positive integers, a and b.
Returns:     The greatest common divisor, g, of a and b.
```

```
Method:
     1. If a<b, exchange a and b.
     2. Divide a by b and get the remainder, r.
        If r=0, report b as the GCD of a and b.
     3. Replace a by b and replace b by r.
        Return to the previous step.
```

7. Write also an iterative version of the previous function (greatest common divisor).
8. Remember how to add fractions:
   $$\frac{a}{c} + \frac{b}{d} = \frac{ad + bc}{cd}.$$
   Usually the result is simplified by dividing the numerator and denumerator by the greatest common divisor. Write a function

   ```
   int add_fraction(int a, int c, int b, int d)
   ```

   which prints the sum of a/c and b/d as a simplified fraction. For example, if the fractions are 11/30 and 1/42, then $1/30 + 1/42 = 12/210 = 6/105 = 2/35$. The fraction 2/35 should be printed followed by a line break in the following format:

   ```
   Result: 2/35\n
   ```

   Test your program also with negative values. The function returns 1, if the operation succeeded, otherwise 0. Note that for each fraction operation there are situation where the operation is not defined, leading to an unsuccessful operation.
9. Write a function

   ```
   int sub_fraction(int a, int c, int b, int d)
   ```

   which prints the subtraction of a/c and b/d as a simplified fraction followed by a line break in the same format as above.
10. Write a function

    ```
    int mul_fraction(int a, int c, int b, int d)
    ```

    which prints the multiplication of a/c and b/d as a simplified fraction followed by a line break in the same format as above.
11. Write a function

    ```
    int div_fraction(int a, int c, int b, int d)
    ```

    which prints the division of a/c and b/d as a simplified fraction followed by a line break in the same format as above.