C Programming, Exercises for the first week

Introduction

In the first exercise we learn to compile programs, to write simple programs with various 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

   Make a program that prints your name. Write your program into a file, compile it with gcc and run it.

2. Make, compile and run the previous program in the NetBeans environment. After this, you can make your programs using either NetBeans or traditional method.

3. 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.ddded(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:

\[
\text{printf("%5d", i);}\]

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 below what output the following calls of printf produce:

a)\text{printf("%6d, %4d", 86, 1040);}\]

b)\text{printf("%12.5e", 30.253);}\]

c)\text{printf("%.4f", 83.162);}\]

d)\text{printf("%-6.2g", .0000009979);}\]

4. C is not a strongly typed language. For that reason it is possible to write expressions using various types. For example, you can add an integer and character and interpret the result as an integer or character.

a) Print 5%2, -5%2, 5%-2, and -5%-2.

b) Let i be an integer variable with value 7, f a floating-point variable with value 5.5 and c a character-type variable with value w. Print i + f as a double-precision type, i + c as an integer, i + c \( -'0' \) as an integer and \( (i + c) - (2 \times f / 5) \) as an double precision value.

c) Suppose f is a floating-point variable whose value is 5.5 Print the value of ((int)f)%2 as an integer.

d) Suppose an integer variable i has a value 1. What is printed below:

\[
\text{printf("i = %d \n", i);}\]

\[
\text{printf("i = %d \n", ++i);}\]

\[
\text{printf("i = %d \n", i);}\]

If the print commands had been

\[
\text{printf("i = %d \n", i);}\]

\[
\text{printf("i = %d \n", i++);}\]

\[
\text{printf("i = %d \n", i);}\]

2
what would have been printed?

e) Suppose \( i \) is an integer variable, \( x \) a floating-point variable and \( c \) a character-type variable. What is printed below:

```c
printf("integer: %d \n", sizeof i);
printf("float: %d \n", sizeof x);
printf("double: %d \n", sizeof d);
printf("character: %d \n", sizeof c);
```

5. A C program contains the following declarations and initial assignments:

```c
int i = 8, j = 5;
double x = 0.005, y = -0.01;
char c = 'c', d = 'd';
```

Determine the value of each of the following expressions and explain the meaning of the library function:

- a) \( \text{abs}(i - 2 \times j) \)
- b) \( \text{fabs}(x+y) \)
- c) \( \text{isprint}(c) \)
- d) \( \text{isdigit}(c) \)
- e) \( \text{toupper}(d) \)
- f) \( \text{ceil}(x) \)
- g) \( \text{ceil}(x+y) \)
- h) \( \text{floor}(x) \)
- i) \( \text{floor}(x+y) \)
- j) \( \text{islower}(c) \)
- k) \( \text{isupper}(j) \)
- l) \( \text{exp}(x) \)
- m) \( \text{log}(x) \)
- n) \( \text{log}(\text{exp}(x)) \)
- o) \( \text{sqrt}(x \times x + y \times y) \)
- p) \( \text{isalnum}(10 \times j) \)
- q) \( \text{isalpha}(10 \times j) \)
- r) \( \text{isascii}(10 \times j) \)
- s) \( \text{tolower}(65) \)
- t) \( \text{fmod}(x,y) \)
- u) \( \text{strlen}("hello\0") \)
- v) \( \text{sqrt}(x \times x + y \times y) \)
6. A C program consists of functions and a main program. Write a function that takes an integer as a parameter and returns its absolute value. Test your function in a main program.

7. 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.

8. Write also an iterative version of the previous function (greatest common divisor).

9. 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 denominator by the greatest common divisor. Write a function

   int addFraction(int a, int c, int b, int d)

   which prints the sum of \(a/c\) and \(b/d\) as a simplified fraction. Test your program also with negative values. The function returns 1, if the operation succeeded, otherwise 0.

10. Write a function

    int subFraction(int a, int c, int b, int d)

    which prints the subtraction of \(a/c\) and \(b/d\) as a simplified fraction.

11. Write a function

    int mulFraction(int a, int c, int b, int d)

    which prints the multiplication of \(a/c\) and \(b/d\) as a simplified fraction.

12. Write a function

    int divFraction(int a, int c, int b, int d)

    which prints the division of \(a/c\) and \(b/d\) as a simplified fraction.