

C programming: exercise sheet

L2-STUE (2011-2012)

Algorithms and Flowcharts

Exercise 1: comparison

Write the flowchart and associated algorithm that compare two numbers a and b.

Exercise 2: 2nd order polynomials

Work out the algorithm that output the solutions of a 2nd order polynomial $ax^2+bx+c=0$, given the parameters a, b and c. Only real solutions will be treated. The cases with 1 or 2 solutions will be separated.

Exercise 3: XOR

Write the flowchart of the XOR logical gate defined by

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Exercise 4: Minimum of three numbers

Find the flowchart and algorithm of a program that finds the minimum of three values a, b and c.

Exercise 5: Guess a number

Write the flowchart and algorithm of a program where the user is asked to find an integer number between 1 and 50.

Exercise 6: sum, average and root-mean-square

Define the flowchart of a program where the user supplies integer values between 1 and 9 and the program returns the sum, average and RMS of the values. The program will exit when 0 is entered. Values outside of the bounds will be discarded.

Write the corresponding algorithm.

Exercise 7: father and son (or mother and daughter)

Define an algorithm that returns the number of years until a father will have an age double of its son's age.

Variables and types

Exercise 8: first step.

Write the C code that prints in the shell the sentence: “hello word!”. Use the *printf* function of the `<stdio.h>` library to print on the screen. In the shell, you will use the *gcc* compiler to produce an executable code (the instruction *man gcc* will print the manual of the *gcc* compiler).

Exercise 9: reading from the keyboard.

Write a C code that asks the user to supply two integer numbers and then prints the two numbers and their sum on the shell. You will use the *scanf* function to read the two numbers.

Exercise 10: divisions

Starting from the following declarations

```
int a=2,b=3,c;  
float x=5.0,y;
```

Predict the outcome of the operations

```
y=a*b;  
c=a*b;  
y=a/b;  
c=a/b;  
y=a/b*x;  
c=a/b*x;  
y=a*x/b;  
c=a*x/b;
```

Write the corresponding C code to check your predictions. You will use the *printf* command (`<stdio.h>`) to output the results on the screen. Note that you will have to use `%i` to print an integer and `%f` to print a float.

Exercise 11: casts

In exercise 8, use cast operations to output the decimal result. In which case does it not work?

Exercise 12: time conversion

Given a time in seconds (integer), print to the screen the corresponding time in hours, minutes and seconds. The output will be formatted like: “XXXX seconds is equivalent to XX hours, XX minutes and XX seconds”.

Exercise 13: constants and pi

Write a C code that computes the surface of a disk given its radius (use *scanf*). The number pi will be declared as a constant in the file.

In a second step, you will call the *<math.h>* library, which defines pi. Look where the library is located on the disk then find how pi is defined (use the UNIX command *locate*). Modify your code to use this library instead of the constant defined in the previous version.

Exercise 14: switch

Write a C code that switches the values of two variables A and B and prints the result on the screen. How many variables do you need?

Exercise 15: addition and memory

Write a C code that sums 4 integer numbers (entered using *scanf*) using:

- 1) 5 variables (numbers are kept in variables)
- 2) 2 variables (no memory of the entries kept)

Exercise 16: distance

Write a C code that calculates the distance between two points whose coordinates in the [X,Y] plane are read from the keyboard as integer.

Handling conditions

Exercise 17: XOR

Write the C code that corresponds to exercise 3 and test it.

Exercise 18: 2nd order polynomial

Write the C code that corresponds to exercise 2 and test it.

Exercise 19: maximum of three numbers

Write a program that reads three integers (A, B and C) and prints the largest of the values using:

- 1) the if-else structure and a temporary variable
- 2) if-else structures without a temporary variable
- 3) conditional operations

Exercise 20: sign of a multiplication

Write a C code that returns the sign of a multiplication of A and B without doing the multiplication.

Exercise 21: parity

Write the C code that returns the parity of a number. Define the proper mathematical test and the associated C condition. The number will be read from the keyboard using the *scanf* function of the *<stdio.h>* library.

Exercise 22: sign of

Using the *if-else* condition then the *switch* condition and finally the ternary operator, write a C code that prints the sign of an integer. Sign returns 1 for a positive number, -1 for a negative number and 0 if the value is 0.

Exercise 23: day of the week

Write a program that outputs the day of the week given a date expressed as *j* (day) *m* (month) *a* (year). You will use the following formula:

$$m_1 = \begin{cases} m-2 & \text{si } m \geq 3 \\ m+10 & \text{si } m < 3 \end{cases} \quad a_1 = \begin{cases} a & \text{si } m \geq 3 \\ a-1 & \text{si } m < 3 \end{cases}$$

and with n_s being the two first digits of a_1 and a_s the two last digits of a_1

$$f = j + a_s + \frac{a_s}{4} - 2n_s + \frac{n_s}{4} + \frac{26m_1 - 2}{10}$$

The day of the week will then be given by the modulo of f and 7 (0 is Sunday, 1 Monday etc).

Using loops

Exercise 24: prime numbers

Write a program that tests if a positive integer is a prime number using a straightforward approach. You will assume that your number is smaller than 1000.

Exercise 25: factorial

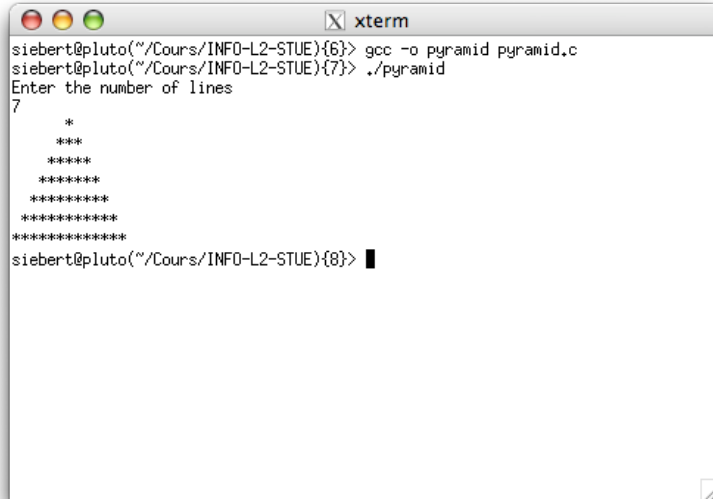
Write a C code that computes the factorial of an integer n . When does the code fail (upper limit on n for the result to be correct) and why?

Exercise 26: x^n

Write the C code that computes x^n using the three loop functions (*for*, *while* and *do-while*). We will consider n to be an integer.

Exercise 27: pyramid

Write a C code that permits the user to specify a number of lines and then prints on the shell a pyramid consisting of stars. The pyramid must be centered. Hints: all characters are printed separately and both the number of stars per line and the number of spaces per line must be calculated. The output will look like:



```

siebert@pluto("~/Cours/INFO-L2-STUE"){6}> gcc -o pyramid pyramid.c
siebert@pluto("~/Cours/INFO-L2-STUE"){7}> ./pyramid
Enter the number of lines
7
  *
 ***
****
*****
*****
*****
*****
*****
siebert@pluto("~/Cours/INFO-L2-STUE"){8}> █

```

Exercise 28: min, max, average

Using a loop, ask the user to enter positive integer numbers. The program will output the number of values entered, the minimum value, the maximum value and the average of all numbers. The code will exit once a negative integer is entered.

Exercise 29: counting entries.

Write a program, which counts the number of time an integer between 0 and 9 has been entered by the user. The program will ask the user to enter a number between 0 and 9, exit when a negative number is entered (values outside of the range will be discarded). Upon exit, it will print the number of 0, 1... entered.

Exercise 30: zero of a function

Write a C code that finds the zero of a function $y=ax+b$ (eg x such as $y=0$) without solving the equation. The zero will be found within a precision limit $eps=10^{-3}$. You will start from a fixed point at $x=0$ and move x in the proper direction until $abs(y)<eps$.

Tables and arrays

Exercise 31: are two vectors orthogonal?

Consider 2 three-dimensional vectors X and Y whose components are specified by the user and stored in 1D arrays. Write the C code that calculates if the two vectors are orthogonal.

Exercise 32: reversing an array

Consider an array X (type float) of length n . Write the C code that prints the array Y, which is X in reverse order.

Exercise 33: minimum and position

Consider an array X of length n randomly initialized. Write a program that returns the position and value of the minimum.

Exercise 34: polynomial

Given a polynomial of degree n defined as follow

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$$

- 1) find the algorithm that allows you to calculate $f(x)$ without the use of the power function
- 2) write the associated C code assuming $n < 50$. The user will be asked to supply the degree of the polynomial as well as the coefficients a_i . The coefficients will be stored in an array.

Exercise 35: sorting an array (simple sort)

Consider an array of integers of size n , whose values are set randomly using the *rand()* function. Write a C code that sorts the elements from the smallest to the largest value. One will proceed as follow:

- 1) search for the minimum of the elements 0 to $n-1$. Swap the minimum and first element.
- 2) Search for the minimum of the elements 1 to $n-1$. Swap the minimum and second element.
- 3) ...
- 4) repeat until the array is sorted.

Exercise 36: matrix product

Write the C code that computes the product of two 3x3 matrices. The code must print the result on the screen.

Exercise 37: concatenation of two tables

Consider X and Y, two sorted arrays of integers. Write a C code that concatenates the two tables into one table Z (sorted) which contains the elements of X and Y.

Functions and recursive programming

Exercise 38: simple functions

Write two C functions that compute $f(x)=2.3*x$ and $g(x,y)=x*y$.

Exercise 39: factorial function

Write a C function *facto* that computes the factorial of a number n (integer) as in exercise 25. The factorial function is a typical recursive problem. Rewrite the equation defining the factorial function in a recursive form. Write the recursive version of the code and call the function *facto_rec*.

Exercise 40: x^n

Rewrite x^n in its recursive form and write the associate C function that you will call *pow_rec*.

Exercise 41: exponential function

The Taylor expansion of the exponential function is given by

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Using the previous exercises, write a C functions that calculates the exponential for integer values of x . Compare its results to the results of the C exponential function defined in `<math.h>`.

Exercise 42: polynomial and derivatives

Assuming that the coefficients of a polynomial are stored in an array a (we will assume its dimension n is at max 50), write a C code that computes its derivative at a point x specified by the user.

Exercise 43: simple integration

Same as exercise 42 but now the function returns the integral of the polynomial between two points. You will use two different methods:

- 1- analytical method: using the coefficients you will compute the integral from a to b of the polynomial
- 2- numerical method: use the trapeze method fixing dx .

Discuss the impact of dx on the accuracy of the result.

Exercise 44: is_prime

Write a function that returns 1 if a integer number is prime, 0 otherwise.

Pointers and structures

Exercise 45: pointer algebra

Using the following declarations:

```
int A[]={2,6,5,1,3};
int *p;
p=A;
```

what is the result of the instructions:

1) *p	6) p	11) p+(*p+4)-3
2) *p+2	7) A	
3) *(p+2)	8) &A[0]	
4) &p	9) &A[0]+3	
5) &p+1	10) A+3	

Exercise 46: complex number and structure

Create a structure to store a complex number and write three functions (for addition, multiplication and norm) that handle this new structure.

Exercise 47: structures (cont.)

Write a program that will stores the personal information of a student (such as name, date of birth, address) in a structure that you will define and then prints the record on the screen. A table of characters can be defined using pointers such as *char *myline*.

Exercise 48: pointers and functions

Write a function of type void that takes as input a table of integers and returns the minimum maximum and sum in three variables. We will assume that the table contains 50 numbers.

Exercise 49: conundrum

What does this function do?

```
int f(int n, int (*f)()) {
    return n>0?n*f(n-1,f):1;
}
```

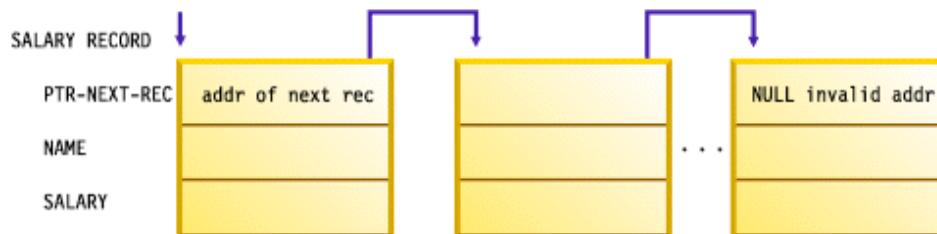

Exercise 50: conundrum 2

What are the outputs of this code? Explain.

```
int t1[] = {0, 0, 1, 1, 1}, t2[] = {0, 0, 1, 1, 1};
main() {
    int *p1 = t1, *p2 = t2;
    while ( !*p1++ || !*p2++ );
    printf("%d %d", p1 - t1, p2 - t2);
}
```

Exercise 51: chained lists

Using pointers and structures, create a chained list in a C program. The elements of the list will describe the record of an employee. The global structure will follow this scheme:



The C program will provide two functions to add a new record and remove the last record.

Strings

Exercise 52: storing and printing

Write a small C program that prints on the screen a text supplied by the user. The string will not be more than 80 characters in size.

Exercise 53: handling strings

Write a small C program that prints each word of a user-supplied text on a new line.

Exercise 54: lower to upper case

Using the integer representation of a character, write a function that replaces all the lower case characters in a string by upper case characters.

Exercise 55: strcpy

Write three version of a function that takes one string as input and copies it on a second string. You will first use the tabular structure of a string then pointers and finally pointer arithmetic.

Exercise 56: palindrome

Write a program that tests if a user-supplied sentence is a palindrome. The program should not be case sensitive.

Exercise 57: characters and strings

Given a string as input, write a function that counts the numbers, the lower case, upper case and special characters.

Exercise 58: extracting characters

From a user-supplied string, write a program that extracts the sentences one by one.

Input/Output and files

For these exercises we assume that the files contain ASCII characters (not binary files).

Exercise 59: reading from a file

Write a program that reads the content of a file and prints it on the screen.

Exercise 60: writing in a file

Write a program that prints x and x^2 in a file for $x=1$ to 10 by steps of 0.1.

Exercise 61: reading and writing

Using the structure student (ex 47) that contains the personal information of a student, write a program that asks the user to enter his/her information and stores it in an existing file named student.txt. If the file does not exist, it will be created. After the user has supplied his information, the program will print on the screen all the student information available in the file.

Putting things together

Exercise 62: direct Monte-Carlo and the value of pi.

Using random numbers, design a simple method to estimate the value of pi. Write the corresponding code and print in a file the convergence of the solution.

Hint: compare the surface of a square and a circle.

Exercise 63: tic-tac-toe

Write the C code that permits to play the tic-tac-toe game. The winner will be saved in a file.

Exercise 64: towers of Hanoi

The towers of Hanoi is a mathematical game where a number of disks of decreasing size from bottom to top are set on the left of a set of three rods. The goal is to move the stack of disks to another rod, obeying the following rules:

- only one disk may be moved at a time
- each move consists of taking the upper disk from one rod and sliding it onto another rod
- no disk may be placed on top of a smaller disk.

Write a C program that solves this problem and prints to the screen the different moves.

Hint: the solution is purely recursive.