MSC - INF101B
C Programming Language
Travaux Pratiques (TP) 1

Panagiotis PAPADAKIS

1 Introduction

The practical session TP 1 proposes coding exercises that will allow you to strengthen your understanding of the notions presented during Course 1. In the beginning of each TP, you should create a separate folder that will contain your work. For the first TP, create a folder named TP1 from a terminal and go into this directory.

NOTES: an integral part of learning any programming language is by encountering mistakes. The TPs may intentionally make you encounter certain common mistakes that may occur. Your purpose is to try to understand what caused the mistake and find the solution, so that you can avoid doing it in the future or know how to deal with it the next time it happens.

Finally, ask your tutor if you do not understanding what is asked by the exercise or if you are blocked by a problem that you do not manage to resolve.

2 Exercises

2.1

Create a file using a system editor (for example gedit, nano) with the name “myfirstprogram”.c . Write a program that prints “Hello ” followed by your name and last name, i.e. “Hello Name SURNAME”. Compile and link the program so that the executable file is called “myprog.hello”.

2.2

Write a new program which prints to the screen the size of bytes for the types of variables that you were taught during the course (you will need to use the function `sizeof()`). Do the values shown by your program confirm those given in the slides?

2.3

2.3.1

Write a new C source file called "logical_use.c" which should print on the screen three distinct tables that explain the result of the logical operators AND, OR and XOR respectively[1], between two variables "a" and "b". Since each variable can take two values (0 or 1), each table should contain $2^2 = 4$ possible result combinations (you will need to use logical operators `&&` and `||`).

2.3.2

1. Change the previous program by creating a function declared as:
   `void logical_operator_use(char x);`

   that prints to the screen the usage of only one logical operator at a time, depending on its argument. Specifically, the function should print the usage of AND operator if x='A', of OR operator if x='O' and of XOR operator if x='X'. Make your program print on the screen the usage of all three operators by calling the function that you have created[2].

2. In this part, you have to separate your function from the C source file that contains your `main` function. To do that, you need to create two new files, namely, a C *header* file called "logical.h" that will only contain the declaration of the function and a C source file called "logical.c" that will contain the definition of the function. Try to first compile the file "logical.c" and afterwards link to perform the executable. What do you notice?

3. To be able to use the function in your program, you will need to include the header file "logical.h" into your main program. Afterwards, try to only compile your program by using the "-c" option of the gcc compiler. If no errors occur, then *link* the object code files "logical.o" and "logical_use.o" using the "-o" option. Execute the produced program and verify that it works as intended.

4. If everything went well until this point, you have successfully created a minimal example of a C library, whose functionality is within the object file "logical.o". Create a copy of file "logical.o" as "logical_yourLASTNAME.o" and of file "logical.h" as "logical_yourLASTNAME.h".Send by e-mail the new object file (not the source) along with the new header file to the person/team in front of you (or to the first person/team of the desks' column where you are belong) and notify them.

   When yourself receive the respective files from the corresponding person/team, modify your program to use their files instead of your own. You can go own with the next exercises until you wait.

[1]see `https://en.wikipedia.org/wiki/Truth_table`

[2]Alternatively, you could try to code this using a `switch` instead of `if else` commands.

2.4

2.4.1

Write a program that prints on the screen, line by line, every possible printable character along with the corresponding integer that represents the character (you will need to use the functions `isprint` and `putchar`[3]). If a character is not printable, then your program should not print anything.

How many printable and non-printable characters are they? What happens if you try to print 1000 characters? Is it normal?

2.4.2

Write a program that performs currency conversions among US dollars \$, Euros € and English pounds £. Your program should show the trading balances that it knows/uses (for example, 1 US \$ = 0.9 Euro €) and ask the user if he/she wants to use the predefined balances or input new balances (given by the keyboard)[4].

In what ways could we increase the accuracy of calculations? Can you think of a way to program the conversions which could guarantee that no rounding errors will happen?

[3]You can obtain documentation of these functions or other, by typing in the linux terminal: man 3 the-name-of-the-function

[4]To print the currency characters \$, € and £ using `printf` function, you can use *escape* characters \u0024, \u20AC and \u00A3