Programming 2

Ayman Makki (ayman.makki@uni.lu)
Semen Yurkov (semen.yurkov@uni.lu)

Lab 4 – Pointers

Exercise 12 – Square Root

#c :Pointer #c :PointerOperator #programming :CallByValue #programming :CallByReference #c :PreprocessorIncluc

Functions in C can at most return a single value. If a change to a certain variable and some status information is required, the typical usage is to pass a pointer to the variable to be changed as an argument to the function, which in turn returns the status information. This can be seen in the very `scanf` function, which expects a pointer to the variable a value shall be read in, but returns the number of values read in.

Write a function `mysqrt` which calculates the square root of a given `double`. The function shall have two parameters : the number for which the square root must be calculated and a pointer to a `double` to store the result of the calculation. It shall return an `int` value equal to 0, if the square root cannot be calculated and equal to 1 if it can be calculated (recall that $\sqrt{x}$ is only defined for $x \geq 0$). You can use the function `double sqrt(double)` which is defined in `math.h` to calculate the square root.

Finally write a program, which tests your function. It shall read a `double` from standard input and display the result.

Exercise 13 – Where are you pointing at ? (based on Exercise 8.3 in *C : From Theory to Practice* by G.S. Tselikis)

#c :Pointer #c :PointerOperator #c :scanf

The following program uses a pointer to read and display a decimal number. Is there any programming bug ?

```c
#include <stdio.h>
int main() {
  double *ptr, i;
  printf("i = ");
  scanf("%lf", ptr);
  printf("i = %lf\n", *ptr);
  return 0;
}
```

Exercise 14 – Array Addresses

#c :PreprocessorMacroDefinition #c :Typedef #c :Pointer #c :ArrayType

Write a C program that declares an array of fixed size. Run over the elements of the array and print the address of each element. Also, print the number of bytes to the next element.

Hints :

— The size and element type of the array shall be easily modifiable in your code.
— To get the address of the slot `i` in an array `arr`, write `&arr[i]`.
— The difference of two pointers results in a numerical value of type `size_t` (which depends on the architecture)
— Use `sizeof()` to get the size in bytes of a given type.

Show that the distance between each element is constant (depending on the chosen element type) and that the array is allocated in memory in a contiguous way. Try your program for different data types, such as `short`, `float` or `long double`.

Ayman Makki (ayman.makki@uni.lu)
Semen Yurkov (semen.yurkov@uni.lu)

For an array of 4 integers, the output would be :

```
Element 0 at 0x7ffee3de7060   Distance to element 1: 4
Element 1 at 0x7ffee3de7064   Distance to element 2: 4
Element 2 at 0x7ffee3de7068   Distance to element 3: 4
Element 3 at 0x7ffee3de706c
```

Exercise 15 – Challenges with pointers

The goal of this exercise is to let you play with pointers and discover their power as well as their weaknesses. You don't **have to** do all of the below exercises, but I would recommend you to try as much as you can. Pointers training is indeed what can make your strength in the following years : do not be afraid of them, learn how to make them useful for your problems.
Try to write a C program with the following functions :

— A function that swaps two numbers using pointers
— A function to construct an array from standard input
— A function to display an array and/or a matrix
— A function that returns the transpose of a matrix using pointers
— A function to reverse an array using pointers
— A function to swap two arrays (please take care of the case where arrays aren't of the same size)
— A function to multiply two matrices (does the memory layout of the matrix influence the speed of it ?)
— A function find the length of a string using pointers
— A function to concatenate two strings using pointers
— A function that returns both an integer, a float and a string using pointers

White book of programming

Please follow this general workflow for the challenges :

1° Think before you start coding !
2° Check the main lines of your programs using a pencil and a paper.
3° Start coding..
4° While (your program bugs & you have energy) : try debugging.
5° If (no more energy) : look up online.
6° Never give up, never surrender !