Programming 2

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

Lab 5 – Arrays & Pointers, Pointer Arithmetic & Strings

Exercise 15 – Array–Pointer Duality

#c :ArrayType #c :Pointer #c :PointerOperator

Write a C console application using only pointers when working with an array. The program must at least implement the following functionalities :

- user input of 21 integer values which are stored in an array (the size should be easily adjustable in your code)
- a function printing the resulting array to standard output
- a function printing only the values present in the array to standard output which are equal to or greater than, e.g., 42. If there are no such values, a corresponding message must be displayed.
- a function which, based on an integer value given by the user :
  - deletes all occurrences of that value from the array
  - additionally, all elements following a deleted value must be shifted left and a 0 written as the last element of the array
  - this procedure must be repeated for all occurrences of the value
  - print the resulting array to standard output using the already implemented function
- a function sorting the array in ascending order
  - you may choose your favorite sorting algorithm
  - print the resulting array to standard output using the already implemented function

All functions must be self-written and put in a separate file.

```
The resulting array: 95 32  78  44  88  54  74  20  38  19  34  74  51  41  97  26  74  5 73  85  55

Values greater than or equal to 42: 95  78  44  88  54  74  74  51  97  74  73  85  55

Enter a value to be deleted: 74
After deleting 74, the resulting array is: 95 32  78  44  88  54  20  38  19  34  51  41  97  26  5 73  85  55  0 0 0

After sorting: 0  0 0 5 19  20  26  32  34  38  41  44  51  54  55  73  78  85  88  95  97
```

Exercise 16 – Size matters

#c :ArrayType #c :Pointer

Demonstrate that using the `sizeof` operator on an array passed as a function parameter will give erroneous results. Why ? What is the solution ?

Exercise 17 – Pointer Arithmetic

#c :PointerArithmetic #c :Pointer #c :Expression #c :Operator

Write a C console application iterating over an integer array using only pointer arithmetic.

1° Generate an integer array with 21 values
- the first value is set to 1
- the following values are calculated by adding 1 to the value of its predecessor
- print the array to standard output

2° Using the applicable post-, pre-, increment-, and decrement-operations for pointers as discussed in the lecture :
- begin with the first value in the array
- write **a single expression** incrementing the current value a pointer points to by 1 and afterwards resetting the pointer to point to the next value in the array
- in addition, use this expression to calculate the sum of all array values while iterating through the array
- print the resulting array and the calculated sum to standard output

```
[1] [2] [3] [4] [5] [6] [7] [8] [9]   [10]  [11]  [12]  [13]  [14]  [15]  [16]  [17]  [18]  [19]  [20]  [21]

[2] [3] [4] [5] [6] [7] [8] [9] [10]  [11]  [12]  [13]  [14]  [15]  [16]  [17]  [18]  [19]  [20]  [21]  [22]

The sum of all array elements is: 252
```

Exercise 18 – *Drei Chinesen mit dem Kontrabass* **#c :String**

Write a C console application which enables the user to change specific characters within a predefined string. For the predefined string, you may use this German children's song :

Drei Chinesen mit dem Kontrabass
sassen auf der Strasse und erzaehlten sich was.
Da kam die Polizei, fragt : "Was ist denn das ?"
Drei Chinesen mit dem Kontrabass.

1° Print the original sentence to standard output.

2° Write a function that counts the number of characters in the sentence. Do *not* count any special characters like blank space or commas.

3° Write a function which replaces a character in the sentence with another character
- — ask the user for the character which should be replaced as well as the one it should be replaced with
- — print the changed sentence to standard output

4° Write a function that replaces all vowels in the sentence with a vowel defined by the user
- — ask the user for the replacement vowel
- — check whether the entered character is indeed a vowel
- — if a vowel was entered, print the changed sentence to standard output

Only use pointer arithmetic to manipulate the strings. In addition, pay attention to correctly replace letters with respect to their case. For instance, if a `d` shall be replaced by a `q`, a `D` in the predefined string shall also become a `Q`. Some functions from the `ctype.h` header file might come in handy.

```
Drei Chinesen mit dem Kontrabass sassen auf der Strasse und erzaehlten sich was. Da kam die Polizei, fragt: 'Was ist denn
      das?' Drei Chinesen mit dem Kontrabass.

Non-special characters in this sentence: 128

Enter the character to replace and the character to be replaced with, separated by a space:
n y

Drei Chiyesey mit dem Koytrabass sassey auf der Strasse uyd erzaehltey sich was. Da kam die Polizei, fragt: 'Was ist deyy
      das?' Drei Chiyesey mit dem Koytrabass.

Enter a vowel the other vowels should be replaced with.
r

No vowel. Enter a vowel the other vowels should be replaced with.
a

Draa Chanasan mat dam Kantrabass sassan aaf dar Strassa and arzaahltan sach was. Da kam daa Palazaa, fragt: 'Was ast dann
      das?' Draa Chanasan mat dam Kantrabass.
```

Exercise 19 – String Comparison #c :String

Write a C console application which enables comparing two strings read from standard input.

1° Read two separate strings with a predefined maximum length from standard input. To do so, implement and call the following function in `main()` :

2° Write two functions to compare the two strings :
- The first one implements this functionality using pointer notation, while the second one uses the subscript operator (`array[0]`).
- Both functions take two arguments of type `char *` and return :
  - -1 if the first string is strictly smaller than the second one
  - 1 if the first string is strictly bigger than the second one
  - 0 if both strings are exactly the same
- Two strings can be compared using the ASCII codes of the individual characters.

3° Print the result of the comparisons to standard output. Also compare the results of your functions with the result of the `strcmp` function. Latter is defined in `string.h` and also takes two arguments of type `char *`.

```
Enter s1: Don't Panic
Enter s2: Don't Pass
Result of mystrcmp1: -1
Result of mystrcmp2: -1
Result of strcmp: -5
```