Programming in C II, Group A
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: JTSK-320112
February 16th, 2018

# Assignment 4 - Makefiles and Function Pointers

- The problems of this assignment must be solved in C.
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules:
  https://grader.eecs.jacobs-university.de/courses/320112/2018_1gA/Grading-Criteria-C2.pdf

## Problem 4.1 *Makefile* (2 points)

**Presence assignment, due by 18:30 h today**
Continue with your solution for **Problem 3.4** in the following manner: write and upload a Makefile called "`Makefile.txt`" which has multiple targets and can be used to: 1) generate all object files corresponding to the source files, 2) generate executable code from the object files and 3) delete all files except source files.
Submit the three source files and the makefile called "`Makefile.txt`".
*You can assume that the input will be valid.*

## Problem 4.2 *Simple function pointers* (2 points)

Write a program that reads a string and then repeatedly reads a command (one character) from the standard input.
If you enter '`a`', then the string should be printed in uppercase on the standard output.
If you enter '`b`', then the string should be printed lowercase on the standard output.
If you enter '`c`', then lowercase characters should be printed uppercase and uppercase characters are printed lowercase on the standard output.
If you enter '`d`', then the program should quit the execution of the program.
Your `main` function or any other function where you read the commands may not contain any `if` or `switch` statements for mapping the command to which function should be called. Your `main` function should contain an endless `while` loop.
Implement the solution using function pointers. The original string should not be changed.
*You can assume that the input will be valid.*

### Testcase 4.2: input

```
This is a String
a
b
c
b
a
d
```

### Testcase 4.2: output

```
THIS IS A STRING
this is a string
tHIS IS A sTRING
this is a string
THIS IS A STRING
```

## Problem 4.3 *Quicksort with function pointers* (2 points)

Write a program that sorts an array of `n` characters. After reading `n` and the values of the array from the standard input, the program reads an integer number and if this number is 1 then the sorting should be ascending, if the number is 2 then the sorting should be descending and if the number is 3 then the program should quit execution.
Your `main` function or any other function where you read the characters and the numbers may not contain any `if` or `switch` statements, but an endless `while` loop for repeated input.
Your program should use function pointers and for sorting you should use the function `qsort()` from `stdlib.h`.
*You can assume that the input will be valid.*

**Testcase 4.3: input**

```
5
b
d
a
e
c
2
1
3
```

**Testcase 4.3: output**

```
e d c b a
a b c d e
```

**Problem 4.4** *Bubblesort with function pointers* (4 points)

Write a program that reads an array of the following structure and sorts the data in ascending order by title or publication year using the *bubblesort algorithm*.

```
struct book {
        char title[45];
        int year;
};
```

Your program should read the number of books from the standard input followed by the array of data corresponding to the books. You should print the lists of sorted books in ascending order with respect to their title (alphabetical order) and with respect to their publication year. Within the sorting according to the title, note that if multiple books have the same title, then they should be sorted ascending with respect to their publication year. Within the sorting according to year, note that if multiple books have the same publication year, then they should be sorted alphabetically with respect to their name.
Instead of writing two sorting functions use function pointers such that you can implement a generic `bubblesort` function able to sort according to different criteria.
*You can assume that the input will be valid.*
The pseudocode of the bubblesort algorithm is the following:

```
repeat
  swapped = false
  for i = 1 to length(A) - 1 inclusive do:
  /* if this pair is out of order */
    if A[i-1] > A[i] then
      /* swap them and remember something changed */
      swap( A[i-1], A[i] )
      swapped = true
    end if
  end for
until not swapped
```

**Testcase 4.4: input**

```
3
Algorithms and Data Structures
1993
Java Programming
2001
C Programming
1987
```

**Testcase 4.4: output**

```
By title:
{Algorithms and Data Structures, 1993}
{C Programming, 1987}
{Java Programming, 2001}
By year:
{C Programming, 1987}
{Algorithms and Data Structures, 1993}
{Java Programming, 2001}
```

## How to submit your solutions

- Your source code should be properly indented and compile with `gcc` without any warnings (You can use `gcc -Wall -o program program.c`). Insert suitable comments (not on every line . . . ) to explain what your program does.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   JTSK-320112
   a4_p1.c
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  **`https://cantaloupe.eecs.jacobs-university.de`**.
  If there are problems (but **only** then) you can submit the programs by sending mail to `k.lipskoch@jacobs-university.de` **with a subject line that begins with JTSK-320112**.
  **It is important that you do begin your subject with the coursenumber, otherwise I might have problems to identify your submission.**
- Please note, that after the deadline it will not be possible to submit any solutions. It is useless to send late solutions by mail, because they will not be accepted.

## This assignment is due by Wednesday, February 21$^{st}$, 10:00 h.