BERGISCHE UNIVERSITÄT WUPPERTAL

University of Wuppertal
School of Mathematics and Natural Sciences
Dr. Holger Arndt

Modern Programming

Winter Term 2021/2022

Exercise Sheet 1

How solutions to programming exercises should be handed in: Exercise sheets are handed out on Mondays, all solutions to exercises must be handed in one week afterwards just before the lecture, i. e. by 12:00 (exception: this and next sheet).

There will be two types of exercises: Occasionally, you will be required to hand in sheets or hard-copies, e. g. when drawing UML diagrams. Please hand in your sheets at the beginning of the lecture. As an alternative, you can scan in or photograph a hand-written solution or use some drawing program and send the file by email.

Most of the time, solutions to exercises will consist of C++ source code files.

- Send your source code files via email to `modprog@studs.math.uni-wuppertal.de`.
- Source code files must be plain text files with correct file extension, i. e. `.cc`, `.cpp`, or `.cxx` for C++ files.
- Do not use `.txt` as extension.
- Do not use spaces in filenames.
- Do not send source code files as inline texts—please send your files as attachments.
- Do not send in compiled executables, only the source code.
- Your solution must be compilable, otherwise points will be deducted.
- Please send your solution only once. If you find an error after already sending in your solution, please send it in again and write a short note why you send your solution again.

As explained in the lecture, you can gain bonus points to improve your exam result by handing in solutions to the exercises. Therefore, handing in solutions in groups is not allowed:

- Solutions must be *unique*!
  - Every student writes her/his own solution.
  - Absolutely no copying from each other!
  - If solutions are identical, you will lose points.
- Please write your name *and* matriculation number into your solutions (email and sheets).

Exercise 1 (Programming, 4 p.)

Write a program that sorts an array $a = (a_0, \dots, a_{n-1})$ using sorting by selection. Requirements:

- read an integer number n from the command line
- create an array $a = (a_0, \dots, a_{n-1})$
- read the array values
- sort the array (preferably in a separate function)
- print the elements of the sorted array

The algorithm “sorting by selection” performs the following operations:

```
for $i = 0, \dots, n-2$
    find a minimal number $a_j$ in $\{a_i, \dots, a_{n-1}\}$
    swap $a_i$ and $a_j$
```

Depending on your programming language the array elements might be numbered $a = (a_1, \dots, a_n)$, in that case you have to adapt the loops.

You can choose any programming language. Some hints how to use different languages on the IT-cluster:

language	source file	compile command	run command
C	`selsort.c`	`gcc -Wall -std=c11 selsort.c -o selsort`	`./selsort`
C++17	`selsort.cc`	`g++ -Wall -std=c++17 selsort.cc -o selsort`	`./selsort`
D	`selsort.d`	`CC=gcc dmd selsort.d -ofselsort`	`./selsort`
Fortran 77	`selsort.f`	`gfortran -Wall selsort.f -o selsort`	`./selsort`
Fortran 95	`selsort.f90`	`gfortran -Wall selsort.f90 -o selsort`	`./selsort`
Java	`Selsort.java`	`javac Selsort.java`	`java Selsort`
C# (Mono)	`selsort.cs`	`mcs selsort.cs`	`mono selsort.exe`

Hand in by: Wed., 27.10.2021 until 12:00 by email to `modprog@studs.math.uni-wuppertal.de`