Aufgabe 1.a.:

Durch wiederholtes Vertauschen der Werte eines Positionspaares in Abhängigkeit ihrer Größenrelation sortiert der Algorithmus eine Zahlenreihe so um, dass die enthaltenen Zahlen, beginnend mit der Kleinsten, aufsteigend von links nach rechts angeordnet werden.

Schrittnr.

Zustand vor Algorithmus-Schritt

Algorithmus Schritt

01

11,5,1

Der Wert auf Position 1 (11) wird ausgewählt. [1]

02

11,5,1

Die Werte der Positionen 1 (11) und 2 (5) werden verglichen. [2]

03

11,5,1

1 (11)>2 (5), die Werte der Positionen 1 und 2 werden getauscht. [3a]

04

5,11,1

Der Wert auf Position 1 (5) wird ausgewählt. [1]

05

5,11,1

Die Werte der Positionen 1 (5) und 2 (11) werden verglichen. [2]

06

5,11,1

1 (5)<2 (11), Position 2 wird ausgewählt. [3b]

07

5,11,1

Die Werte der Positionen 2 (5) und 3 (1) werden verglichen. [2]

08

5,11,1

2 (11) > 3 (1), Die Werte der Positionen 2 und 3 werden getauscht. [3a]

09

5,1,11

Der Wert auf Position 1 (5) wird ausgewählt. [1]

10

5,1,11

Die Werte der Positionen 1 (5) und 2 (1) werden verglichen. [2]

11

5,1,11

1 (5)>2 (1), die Werte der Positionen 1 und 2 werden getauscht. [3a]

12

1,5,11

Der Wert auf Position 1 (1) wird ausgewählt. [1]

13

1,5,11

Die Werte der Positionen 1 (1) und 2 (5) werden verglichen. [2]

14

1,5,11

1 (1)<2 (5), Position 2 wird ausgewählt. [3b]

15

1,5,11

Die Werte der Positionen 2 (5) und 3 (11) werden verglichen. [2]

16

1,5,11

2 (5)<3 (11), Position 3 wird ausgewählt. [3b]

17

1,5,11

Es gibt keine weitere Position mit deren Wert der Wert der Position 3 verglichen werden kann.

Die Zahlenreihe ist zu Ende, somit ist der Algorithmus beendet. [4]

Aufgabe 1.b.:

Der Algorithmus arbeitet für alle Zahlenreihen Korrekt. Es können drei Fälle unterschieden werden:

• Alle in der Reihe enthaltenen Zahlen sind gleich.

Durch die Arbeitsanweisung, die Werte zweier Positionen zu tauschen sofern der Wert der vorderen Position größer ist und im anderen Fall (Wert ist kleiner oder gleich) mit der nächsten Position weiter zu verfahren durchläuft der Algorithmus alle Positionen.

• Alle in der Reihe enthaltenen Zahlen sind verschieden.

Wie in 1. kann der Algorithmus alle Positionen ohne Widersprüche durchlaufen.

• In der Reihe sind sowohl verschiedene als auch gleiche Zahlen enthalten.

Durch die in 1. geschilderte Eigenschaft sortiert der Algorithmus Positionen mit gleichen Werten hintereinander und verfährt im Anschluss widerspruchslos weiter bis alle Positionen durchlaufen sind.

Aufgabe 2.a.:

Eine Variable im Sinne der Informatik ist ein mit einem Namen versehener Speicherbereich dessen Größe über seinen Typ vorgegeben ist und in welchem ein Wert abgelegt werden kann.

Zwei unterschiedlich benannte Variablen können demnach typabhängig den selben Wert annehmen.

Eine Funktion die sich demnach auf eine der beiden Variablen beziehen kann würde mit dem selben Wert arbeiten, welcher jedoch in unterschiedlichen Speicherbereichen abgelegt ist.

Aufgabe 2.b.:

Eine Zuweisung ist ein Statement einer Programmiersprache, das eine Operation mit Variablen beschreibt. Einem benannten Speicherbereich wird demnach ein Wert zugewiesen.

Aufgabe 3.a.:

Das Erstellen einer Liste mit beispielhaften Ein- und Ausgaben erleichtert bei komplexen zu implementierenden Algorithmen den Prüfschritt. Vielschrittige, schwer zu überblickende Berechnungen können fälschlicher Weise als Korrekt berechnet angenommen werden. Eine Liste mit Beispielhaften Ein- und Ausgaben führt somit zu einer erleichterten Fehlererkennung während der Implementierung und im Test/Revisions-Schritt.

Aufgabe 3.b.:

Durch die Definition von Konstanten, „nicht veränderbaren“ Variablen macht man das Programm für sämtliche Varianten der vorgesehenen Berechnung einsetzbar.

Blablabla final int in java

Aufgabe 3.c.:

Ich fand die in Absatz 2.1.9 behandelten Entscheidungsregeln wann eine Help-Function zu bilden ist am hilfreichsten. So werde ich zukünftig meinen Workflow besser streamlinen können.

Ich fand den Aufbau des Skripts sehr angenehm. Es wurde in jeden schritt sehr gut dargelegt was getan würd uns aus welchem Grund, was einem gut ermöglicht hat der code Entwicklung zu folgen.

Ich fand generell interessant wie man an den Aufbau eines Programmes heran gehen kann und wie man an die jeweiligen Entwiekelungsphasen (wie z.B. in 2.1.9) heran gehen kann.

Aufgabe 4.:

Version Redou:

Using built-in specs.

COLLECT\_GCC=gcc

COLLECT\_LTO\_WRAPPER=c:/mingw/bin/../libexec/gcc/mingw32/4.8.1/lto-wrapper.exe

Target: mingw32

Configured with: ../gcc-4.8.1/configure --prefix=/mingw --host=mingw32 --build=mingw32 --without-pic --enable-shared --enable-static --with-gnu-ld --enable-lto --enable-libssp --disable-multilib --enable-languages=c,c++,fortran,objc,obj-c++,ada --disable-sjlj-exceptions --with-dwarf2 --disable-win32-registry --enable-libstdcxx-debug --enable-version-specific-runtime-libs --with-gmp=/usr/src/pkg/gmp-5.1.2-1-mingw32-src/bld --with-mpc=/usr/src/pkg/mpc-1.0.1-1-mingw32-src/bld --with-mpfr= --with-system-zlib --with-gnu-as --enable-decimal-float=yes --enable-libgomp --enable-threads --with-libiconv-prefix=/mingw32 --with-libintl-prefix=/mingw --disable-bootstrap LDFLAGS=-s CFLAGS=-D\_USE\_32BIT\_TIME\_T

Thread model: win32

gcc version 4.8.1 (GCC)

Berk Gülgec:

Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-gxx-include-dir=/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.11.sdk/usr/include/c++/4.2.1

Apple LLVM version 7.0.0 (clang-700.1.76)

Target: x86\_64-apple-darwin15.0.0

Thread model: posix