Programmierkurs
Winter 2025/26

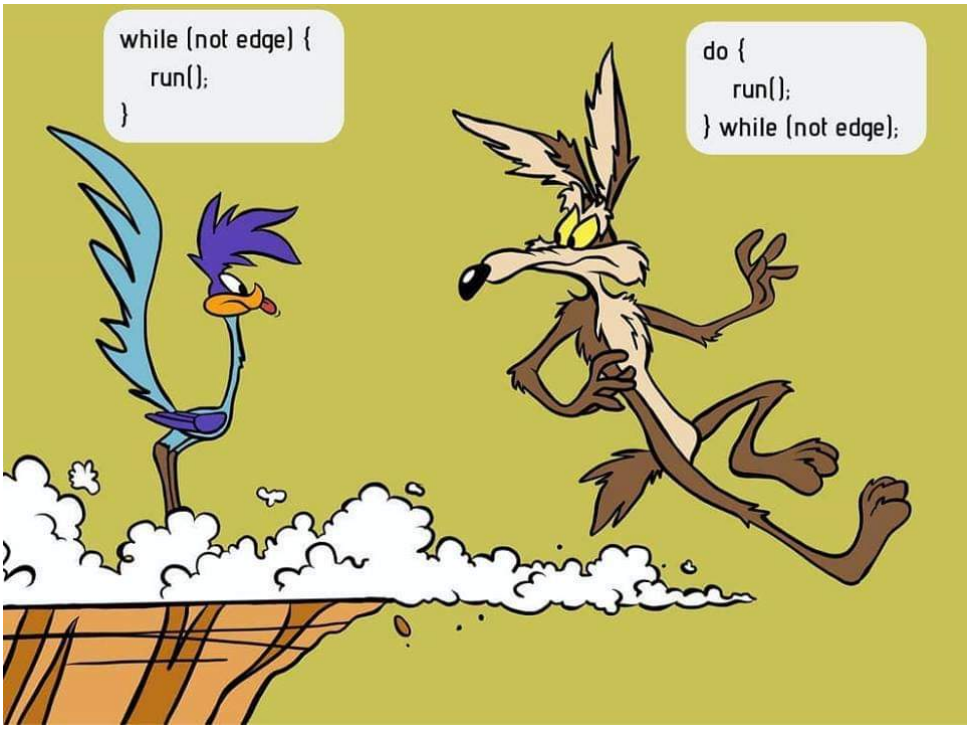
Department Mathematik / Informatik
Dr. Stefan Zellmann
Simon Wolf

Übungen 17.11.-21.11.2024
Blatt 5

Abgabe der Hausaufgaben bis spätestens **Freitag, 21.11.2025 um 23:59 Uhr** via git.
Besprechung der Hausaufgaben in der nächsten Woche.

Hinweise:

- Zur Erinnerung: Erstellen Sie ein neues Repository für jedes Übungsblatt. Folgen Sie dabei den Anweisungen aus der git-Starthilfe.
- Die Bewertung jeder Aufgabe erfolgt unabhängig; ein Fehler in Hausaufgabe 3 beeinflusst beispielsweise nicht die Bewertung von Hausaufgabe 1.
- Um die Hausaufgaben zu testen, weisen wir den Variablen diverse Werte zu. Stellen Sie also sicher, dass Ihre Lösung auch unabhängig von den Werten funktioniert, mit denen sie Sie initialisieren.
- Denken Sie daran den SSH und nicht den https Link zu Ihrem Repository in Ilias hochzuladen:
  `git@github.com:<USERNAME>/<REPOSITORY_NAME>.git`.
- In einigen Aufgaben sollen Sie ein **Array** deklarieren und initialisieren. Beachten Sie, dass bei unseren Tests sowohl die Werte als auch die Länge der Arrays variieren können, sofern nicht anders in der Aufgabenstellung angegeben. Wenn nicht anders gesagt, dürfen Sie stets davon ausgehen, dass die von uns zum Testen verwendeten Arrays immer mindestens eine Zelle haben.



Präsenzaufgabe 1 [Schleifen: Umformungen]

Erstellen Sie im Repository ein Package **p1** und fügen Sie diesem Package die Klasse **P1_main** hinzu, welche die **main**-Methode enthält.

Betrachten Sie folgenden Quellcode:

```
int x=0;
while ( x <= 10 ) {
 x = x + 1;
 System.out.println(x);
}
```

(a) Geben Sie zu der oben angegebenen Schleife eine **for**-Schleife an, welche die gleiche Ausgabe bewirkt. Das heißt, ersetzen Sie die fünf Zeilen mit einer **for**-Schleife. Dabei soll die Ausgabe Ihres Quellcodes mit der Ausgabe, welche durch die obigen fünf Zeilen verursacht wird, übereinstimmen.

(b) Geben Sie zu der oben angegebenen Schleife eine **do-while**-Schleife an, welche die gleiche Ausgabe bewirkt. Dabei soll die Ausgabe Ihres Quellcodes mit der Ausgabe, welche durch die obigen fünf Zeilen verursacht wird, übereinstimmen.

Präsenzaufgabe 2 [Maximum und Zellenindex bestimmen]

Erstellen Sie im Repository ein Package **p2** und fügen Sie diesem Package die Klasse **P2_main** hinzu, welche die **main**-Methode enthält.

- Deklarieren und initialisieren Sie ein **int**-Array namens **numbers**. Weiterhin deklarieren Sie zwei **int**-Variablen namens **max** und **minIndex** und weisen beliebige Werte zu.
- Schreiben Sie ein Programm, welches in **max** den größten aller in **numbers** enthaltenen Werte speichert. In **minIndex** soll der kleinste Index einer Zelle, die den Wert aus **max** enthält, gespeichert werden.

Beispiele:

- Beispiel 1: Ist **numbers = {4,2,3,4,1}**, dann wird in **max** der Wert 4 und in **minIndex** der Wert 0 gespeichert.
- Beispiel 2: Ist **numbers = {1,-8000,5}**, dann wird in **max** der Wert 5 und in **minIndex** der Wert 2 gespeichert.

Präsenzaufgabe 3 [Zweidimensionales Array: Matrix-Transformation]

Erstellen Sie im Repository ein Package **`p3`** und fügen Sie diesem Package die Klasse **`P3_main`** hinzu, welche die **`main`**-Methode enthält.

Erstellen Sie ein Programm, das eine 3×3-Matrix mit Hilfe eines zweidimensionalen Arrays von ganzen Zahlen initialisiert und dann folgende Operationen durchführt:

(a) Erhöhen Sie jedes Element der Matrix um 1 und geben Sie die Matrix aus.

(b) Ermitteln Sie die Summe aller Elemente in der Matrix und geben Sie den Wert auf der Konsole aus.

(c) Drehen Sie die Matrix um 90 Grad im Uhrzeigersinn und geben Sie die gedrehte Matrix auf der Konsole aus.

Beispiel:
Gegeben die Matrix:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Ausgabe nach (a):

$$\begin{bmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{bmatrix}$$

Ausgabe nach (b):

Summe der Elemente: 45

Ausgabe nach (c):

$$\begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

Hausaufgabe 1 [Reihenfolge umdrehen] — 15 Punkte

Erstellen Sie im Repository ein Package **`h1`** und fügen Sie diesem Package die Klasse **`H1_main`** hinzu, welche die **`main`**-Methode enthält.

- Deklarieren und initialisieren Sie ein **`int`**-Array namens **`myArray`**.
- Schreiben Sie ein Programm, welches die Reihenfolge der Werte in **`myArray`** umdreht. Am Ende des Programms muss also der erste Wert in **`myArray`** genau dem Wert entsprechen, der bei der Initialisierung der letzte Wert in **`myArray`** gewesen ist.

Beispiel:

- Aus | 5 | 6 | 7 | 10 | soll | 10 | 7 | 6 | 5 | werden.

Hausaufgabe 2 [Ziffern und Arrays] — 35 Punkte

Erstellen Sie im Repository ein Package **`h2`** und fügen Sie diesem Package die Klasse **`H2_main`** hinzu, welche die **`main`**-Methode enthält.

- Deklarieren und initialisieren Sie zwei **`int`**-Variablen namens **`n`** und **`digits`**. Deklarieren und initialisieren Sie weiterhin ein **`int`**-Array names **`a`**, welches 9 Zellen enthält, die alle mit 0 vorbelegt sind.
- Schreiben Sie ein Programm, welches die Anzahl der Ziffern, aus denen **`n`** besteht, berechnen und in **`digits`** speichert. Dabei besteht die Zahl 0 aus einer Ziffer, die Zahl 42 aus zwei Ziffern, die Zahl 48889 aus fünf Ziffern, und so weiter.
- Ihr Programm soll neben obigen Anforderungen auch noch jede einzelne Ziffer der Zahl **`n`** in einer Zelle des Arrays **`a`** abspeichern. Dabei soll ganz rechts in **`a`** die letzte Ziffer stehen, links daneben die zweitletzte Ziffer, und so weiter. Besteht die Zahl **`n`** nur aus $k < 9$ Ziffern, so sollen die linken $9 - k$ vielen Zellen den Wert 0 beinhalten.

Beispiel:

- Bei n = 299 soll die Belegung des Arrays so aussehen: | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 9 | 9 |

Hinweis:

- Sie dürfen davon ausgehen, dass die Zahlen, die wir testen, nicht-negativ sind, nicht mehr als 9 Ziffern haben und in eine int-Variable passen.

Hausaufgabe 3 [Stückzahl zu Betrag] **50 Punkte**

Gehen Sie in das Package **h3**. Die **main**-Methode finden Sie in der Klasse **H3_main**.

- Deklarieren Sie das zweidimensionale **int**-Array **einheiten** (mit Dimension **[2][15]**) und initialisieren Sie **einheiten** mit diesen Werten in der ersten Zeile:
  50000, 20000, 10000, 5000, 2000, 1000, 500, 200, 100, 50, 20, 10, 5, 2, 1
  Und überall mit 0 in der zweiten Zeile.

  Weiterhin deklarieren Sie die **int**-Variable **input** und weisen ihr einen beliebigen Wert zu.

- Schreiben Sie ein Programm, welches den Geldbetrag (in Cent), welcher in **input** gespeichert ist, in möglichst wenig Scheine und / oder Münzen zerlegt.

  – Beispiel: Der Geldbetrag **208** lässt sich in 2 1-Euro-Münzen und 4 2-Cent-Münzen zerlegen. Dies sind aber nicht möglichst wenig Scheine und / oder Münzen (6 Münzen). Ihr Programm sollte diesen Betrag in ein 2-Euro-Stück, ein 5-Cent-Strück, ein 2-Cent und ein 1 Cent Stück zerlegen (4 Münzen).

- Um die oben genannte Aufgabe zu erfüllen, tragen Sie in **einheiten** in der zweiten Zeile ein, wie oft das entsprechende Geldstück genommen werden muss, um den Betrag aus **input** mit minimal vielen Elementen darstellen zu können.

  Beispiel: Hat **input** den Wert 6279, ist **einheiten** am Ende wie folgt mit Werten befüllt:
  50000, 20000, 10000, 5000, 2000, 1000, 500, 200, 100, 50, 20, 10, 5, 2, 1
  0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 2, 0.
  Denn 5000 + 1000 + 200 + 50 + 20 + 5 + 2*2 = 6279.

  Hinweis:

  – Bei unseren Tests dürfen Sie davon ausgehen, dass wir das Array **einheiten** genauso initialisieren werden wie wir es Ihnen oben beschrieben haben.

  – Sie dürfen davon ausgehen, dass wir **input** bei unseren Tests nur nicht-negative Werte zuweisen werden.