Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Tag 2

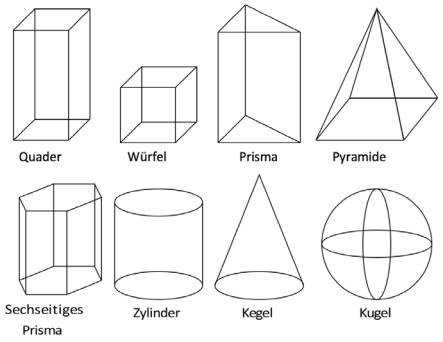
Aufgabe: Volumen- und Oberflächenberechnung

GRUNDLAGE

Schreibt euch ein Programm, das vom Benutzer einen Radius und eine Höhe einliest. Anschließend soll der Benutzer auswählen können, ob das Volumen oder der Oberflächeninhalt des Kegels berechnet und ausgegeben wird.

ZUSATZ

Erweitert das Programm für beliebige dreidimensionale Körper. Fragt den Benutzer dazu am Anfang, für welchen Körper die Berechnungen angestellt werden sollen.



https://www.schuelerhilfe.de/online-lernen/1-mathematik/2-klasse-5/16-k%C3%B6rper/441-geometrische-k%C3%B6rper

Aufgabe: Einlesen verschiedener Datentypen und PI raten

GRUNDLAGE

Schreibt euch ein Programm, das es dem Benutzer ermöglicht einen beliebigen, primitiven Datentyp einzulesen.

ZUSATZ

Schreibt euch ein Programm, das den Benutzer zunächst fragt, auf wie viele signifikante Stellen dieser glaubt PI zu kennen. Anschließend soll der Benutzer bis zu eben dieser Stelle PI eingeben und das Programm überprüfen, ob das Ergebnis richtig ist.

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Tag 4

Aufgabe: ASCII und Strings

GRUNDLAGE

Schreibt ein Programm, das vom Benutzer ein Wort einliest und jedes Zeichen, in ASCII-Codierung (dez) umwandelt und auf der Konsole ausgibt.

ZUSATZ

Schreibt ein Programm, das vom Benutzer ein Wort einliest und die Groß- und Kleinschreibung der einzelnen Zeichen vertauscht.

Aufgabe: Folgen erkennen

GRUNDLAGE

- Versucht folgende Folgen durch Formulierung einer Schleife auf der Konsole auszugeben:
  - -5 -2 1 4 7 10 13 16 19
  - 3 4 6 10 18 34 66 130 258 514 1026 2050 4098
  - 1 2 4 7 11 16 22 29 37 46 56 67 79 92
- Hinweis: Überlegt euch erst welche Systematik hinter den Folgen steckt und fangt dann an zu programmieren.

ZUSATZ

Schreibt ein Programm, das die folgende Zahlenfolge mithilfe einer Schleife auf die Konsole ausgibt:
1, 3, 5, 7, 9, 11, 13, 15, 17, 19

Aufgabe: Harmonische Reihe mal anders

› Schreibt euch ein Programm, mit dessen Hilfe ihr die harmonische Reihe berechnen könnt:

$$H_n = \sum_{i=1}^{n} \frac{1}{i^{\alpha}}$$ (üblicherweise mit α=1)

› Es soll allerdings eine spezielle Implementierung vorgenommen werden,

- die alle $\alpha \geq 1$ erlaubt.
- die alle $\alpha < 1$ auf 1 setzt.
- für $n < 1$ als Ergebnis 0 zurückgibt.

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Tag 5

Aufgabe: Ich packe meinen Koffer

GRUNDLAGE

Schreibt euch ein Programm, welches das Spiel „Ich packe meinen Koffer“ mit zwei Spielern, die gegeneinander spielen, umsetzt. Das Spiel soll zu Ende sein, sobald ein Spieler die Reihenfolge nicht mehr fehlerfrei aufsagen kann.

ZUSATZ

Erweitert das Spiel, sodass nun beliebig viele Spieler spielen können. Vergesst nicht, dass sobald ein Spieler einen Fehler macht dieser in der nächsten Runde bei der Abfrage übergangen wird.

Aufgabe: Kopfrechnen

GRUNDLAGE

Schreibt euch ein Programm, das den Benutzer im Kopfrechnen trainiert mit +- Aufgaben. Dabei sollen sowohl die Zahlen die verrechnet werden, als auch das Ergebnis im Bereich zwischen 0 und 100 liegen.

ZUSATZ

Erweitert euer Programm zum Kopfrechnen, sodass nun auch zusätzlich noch */ Aufgaben trainiert werden können. Achtet wieder auf den Bereich, in dem die Zahlen liegen und dass das Ergebnis aus der Division von Dividend und Divisor eine Ganzzahl ergibt.

Aufgabe: Schiffe versenken

Schreibt euch ein Programm, welches das Spiel „Schiffe versenken“, mit zwei Spielern, umsetzt.
Legt euch dazu zunächst eine Klasse Player an. Diese soll zwei private Attribute haben: den Namen des Spielers und ein Spielfeld mit 8 x 6 Feldern. Schreibt euch zusätzlich Getter- und Setter-Methoden für beide Attribute.
Erstellt euch nun zwei Objekte der Klasse Player, lest von den Nutzern jeweils ihren Namen und 8 kleine Boote ein und spielt nun solange bis ein Spieler keine Boote mehr auf seinem Spielfeld hat (schreibt euch für die Abfrage eine Methode: `public boolean shipLeft()`).

Schreibt euch eine Methode `public static boolean kiPlays(Player p)`. Diese soll eine einfache KI simulieren, die lediglich auf ein zufälliges Feld des übergebenen Spielers p setzt und dann zurückgibt, ob sie getroffen hat oder nicht.

Aufgabe: Reaktionsfähigkeit (eventuell zu schwer)

Schreibt euch ein Programm, das eure Reaktionsfähigkeit testet. Dazu soll Java eine zufällige Zeit warten bis auf der Konsole "Los!" ausgegeben wird. Ab dem Moment soll die Zeit bis zur Betätigung der Enter-Taste gemessen und anschließend ausgegeben werden.

Aufgabe: Matrizenrechnung

Schreibt euch ein Programm, das folgende Matrizenoperationen ausführen kann:

- Addition/Subtraktion zweier Matrizen
- Multiplikation einer Matrix mit einem Skalar
- Multiplikation zweier Matrizen
- Negation einer Matrix

› Erstellt euch für jede Rechenoperation eine Methode und eine die das ausgeben der Matrizen auf der Konsole übernimmt

› Die Methoden dürfen die Übergabematrizen nicht verändern

$$y \downarrow \begin{bmatrix} a_{11} & \cdots & a_{1x} \\ \vdots & \ddots & \vdots \\ a_{y1} & \cdots & a_{yx} \end{bmatrix} \rightarrow$$

```
double value = a[y][x];
```

Aufgabe: Umrechnung von Zahlensystemen

Schreibt euch ein Programm, das es euch ermöglicht vom Dual-, Hexadezimal- und Dezimalsystem eine Konvertierung in die jeweils anderen Zahlensysteme vorzunehmen. Fragt den Benutzer dazu am Anfang welche Umwandlung er vornehmen möchte.

Aufgabe: Maximum und InsertionSort

GRUNDLAGE

Schreibt euch ein Programm, welches das Maximum aus einem Array findet und anschließend dieses mit der vordersten Zahl (Index 0) vertauscht.

ZUSATZ

Implementiert einen einfachen Sortieralgorithmus (InsertionSort). Dieser iteriert durch das Array und setzt dann jeden Wert an die richtige Stelle, indem er den momentanen Wert mit allen vorherigen Elementen vergleicht.

Aufgabe: Passwort auf Qualität überprüfen

GRUNDLAGE

Schreibt euch ein Programm, welches ein vom Benutzer eingegebenes Passwort auf seine Qualität überprüft. Ein gutes Passwort muss mindestens:

- acht Zeichen
- zwei Buchstaben
- zwei Ziffern
- ein Groß- und Kleinbuchstaben
- ein Sonderzeichen

enthalten.

ZUSATZ

Euer Programm soll nun ein Passwort nur noch als gut bewerten, wenn sich mindestens eine Ziffer oder ein Sonderzeichen innerhalb des Wortes befindet.

Aufgabe: Korrekte Klammerung

- Implementiert eine Java-Funktion, die für eine Zeichenkette (oder char-Feld) überprüft, ob die dort enthaltenen runden Klammern den Regeln einer vollständigen Klammerung, wie bei einem Ausdruck, entsprechen. Das heißt: Für jede öffnende Klammer '(' muss eine nachfolgende schließende Klammer ')' existieren und die runden Klammern müssen korrekt verschachtelt sein. Zeichen außer den runden Klammern sollen ignoriert werden.
- Folgendes sind korrekt geklammerte Zeichenketten: "()", "", "(()(a)(()((c))))". Diese nicht: "(()", "a (()()) a)".

Aufgabe: Kleinster Abstand benachbarter Zahlen

Schreibt euch ein Programm, das die Zahlen mit dem kleinsten Abstand zueinander in einem Array findet und den Index der ersten Zahl des Paares zurückgibt.

Aufgabe: Newton-Iterationsverfahren

Das Newton-Iterationsverfahren wird in der Mathematik zum Lösen von nichtlinearen Gleichungen und Gleichungssystemen verwendet.

Zur Berechnung verwendet man folgende Formel:

$$x_{n+1} = x_n + \frac{f(x_n)}{f'(x_n)}$$

Der Startwert x_0 sollte dabei möglichst nahe an dem Endergebnis sein, sodass das Newton-Verfahren in wenigen Iterationsschritten zu einem Ergebnis kommt.

Hinweis: $f'(a) = \frac{f(a+\Delta a) - f(a)}{\Delta a}$

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Tag 6

Aufgabe: Primzahlen

GRUNDLAGE

Schreibt euch ein Programm, welches alle Primzahlen von 1 bis zu einer vorgegebenen Zahl ausgibt. Setzt den Algorithmus rekursiv um.

ZUSATZ

Setzt das ganze nun auch iterativ um.

Aufgabe: Euklidischer Algorithmus

Schreibt euch ein Programm, das den größten gemeinsamen Teiler zweier beliebiger Ganzzahlen rekursiv berechnet.

Aufgabe: Rekursive Funktion implementieren

Gegeben sei folgende rekursiv definierte Funktion f:

$f(n) := 1$, für $n = 1$

$f(n) := f(n-1) + 2n - 1$, für $n > 1$

Implementiert eine rekursive Java-Methode, die f(n) berechnet (ohne Iterationen).

Gebt eine nicht-rekursive Implementierung von f an.