Exercise 4 - Arrays and Structs

Diese Woche werden wir uns beim Übungsblatt auf das implementieren von einigen Funktionen auf `Arrays` und `Structs` fokussieren.

Bei Fragen oder Problemen, bitte in das Moodle-Forum schreiben!

Exercise 1 — Arrays

Schreiben Sie bitte für diese Aufgabe Ihre Funktionen in die `Exercise1_Arrays/Arrays.cpp`*-Datei. Wie immer können Sie Ihren Code dann in* `main.cpp` *testen. Sollten Sie Probleme mit dem Erstellen der Funktionsköpfe haben, können Sie sich gerne die* `Exercise1_Arrays/Arrays.h`*-Datei angucken.*

a) *Gegebenen sei das Array* `grades` *und seine Länge* `amount_of_grades`*. Schreiben Sie eine Funktion* `average`*, die ein* `double[]`*-Array und dessen Länge als* `int` *übergeben bekommt und den Durchschnitt der Einträge des Array als* `double`*-Wert zurück gibt.*

   Den Durchschnitt eines Arrays kann man mit der folgenden Formel berechnen:

$$\frac{1}{n}\sum_{k=1}^{n} a_k \quad \text{Beispiel: } \frac{3+2+1}{3} = 2$$

b) *Nun sollen Sie die Funktion* `contains`*implementieren, die überprüft, ob eine übergebene Nummer ein Element eines übergebenen Arrays ist oder nicht. Da wir die Funktion auf* `double`*-Werten basieren wollen, müssen wir zuvor eine Funktion* `compare` *schreiben, da wir nicht* `==` *zwischen zwei Fließkommazahlen benutzen wollen.*

   1) *Die* `compare` *Funktion bekommt zwei* `double`*-Werte als Parameter übergeben und soll* `true` *zurückgeben, falls die Beiden Werte weniger als* `0.00001` *außeinander liegen, ansonsten soll die Funktion* `false` *zurückgeben.*

      Für die Eingaben `5.0` und `4.9` würde Beispielsweise gelten:

$$|5.0 - 4.9| < 0.00001$$
$$0.1 < 0.00001$$
$$\Rightarrow \text{False}$$

   2) *Implementieren Sie die Funktion* `contains`*, die ein* `double[]`*-Array und dessen Länge als* `int`*-Wert übergeben bekommt sowie ein* `double` *Ziel, dessen Index zurückgegeben werden soll, sofern das Ziel im Array gefunden wird. Ansonsten soll Ihre Funktion den Wert* `-1` *zurück geben. Benutzen Sie die zuvor implementierte Funktion* `compare`*, um Zwei* `double`*-Werte zu vergleichen.*

   3) ***Bonus-Aufgabe:***

Vergleichen Sie Ihre Lösung der Funktion mit der `contains_binary_search`*-Funktion. Machen die Beiden das Selbe? Wie viele Speicherzellen werden überprüft, bevor das Ziel gefunden wird? Welche Annahme muss für das Array gelten, damit die Funktion richtige Ergebnisse erbringt?*

Antworten auf diese Fragen werden Sie auch im Internet finden, wenn Sie `Binäre Suche` in Ihrem Web-Browser eingeben.

Exercise 2 — Structs

Schreiben Sie für diese Aufgabe Ihren Code in die `Exercise2_Structs/Structs.h`*-Datei. Wie gehabt können Sie in der* `main.cpp`*-Datei Ihren Code testen.*

a) *Schreiben Sie eine* `struct`*-Deklaration* `Pizza`*, die die Beiden* `float`*-Felder* `diameter` *und* `price` *besitzt, ähnlich zur der, die Sie aus der Vorlesung kennen. Initialisieren Sie innerhalb Ihrer* `main`*-Funktion zwei Pizzen, wobei Sie die Werte von Ihren liebsten Pizzen nehmen können(Oder sich können sich die Werte selbst ausdenken.)*

b) *Schreiben Sie die Methode* `price_per_cm2`*, die genau den* `float`*-Wert zurück gibt, den Ihr Name vermuten lässt.*

c) *Schreiben Sie die Methode* `five_euro_get_you_x_cm2`*, die intern mit der* `price_per_cm2`*-Methode arbeitet und so die Anzahl an cm2 berechnet, die man für 5€ bekommt und diesen Wert als* `float`*-Wert zurück gibt.*

Testen Sie Ihre Methoden via der `main`*-Funktion um zu sehen, welche Pizza Sie ab sofort öfter bestellen sollten.*