Praktikum 4: Referenzen & Funktionen

1 Lernziele

- Referenzübergabe
- Funktionsüberladung
- Templates

2 Funktionstemplates

In dieser Aufgabe sollen Sie zwei kleine Funktionen als **Templates** verfassen. Die beiden Funktionen führen jeweils sehr generische Operationen auf Vektoren aus, und sollen als Templateversionen für Zahlendatentypen funktionieren.

2.1 `isSorted`

Schreiben Sie eine Funktion zur Überprüfung, ob ein als Parameter übergebener `vector` aufsteigend sortiert ist oder nicht. Ist dies der Fall, soll die Funktion `true` zurück liefern, andernfalls `false`.

An dem `vector` an sich sollen **keine** Änderungen vorgenommen werden, übergeben Sie allerdings aus Effizienzgründen den `vector` als **Referenz**.

2.2 `getAverage`

Schreiben Sie eine weitere Funktion, die den Durchschnitt eines Vektors berechnet und diesen Wert zurück gibt. Bezüglich Referenzen gelten die gleichen Aussagen wie in Aufgabe 2.1.

3 Funktionsüberladung

Definieren Sie folgende **Strukturen** (`struct`) oder Klassen (`class`) für geometrische Objekte:

`Punkt2D` soll eine Koordinate im zweidimensionalen Raum mit je einem X- und Y-Wert darstellen.

`Kreis` hat einen Mittelpunkt sowie einen Radius. Der Radius soll dabei eine reelle Zahl sein.

`Dreieck` ist definiert durch drei Eckpunkte, die beliebig im 2D-Raum liegen können.

`Rechteck` ist entweder über vier Eckpunkte definiert, **oder** über den linken oberen Eckpunkt und den rechten unteren Eckpunkt **oder** aber durch einen Punkt und die zusätzliche Angabe Länge und Breite.
(Entscheiden Sie, was Sie bevorzugen)

Hinweis: Gehen Sie bei den Rechtecken davon aus, das diese **achsparallel** liegen.

3.1 Funktionen

Implementieren Sie die geforderten Funktionen **und** schreiben Sie eine dazugehörige `main`, in der Sie die Funktionen verwenden (testen).

Lesen Sie zunächst die Aufgabenstellung komplett durch und überlegen Sie, ob Sie wieder- kehrende Teilprobleme entdecken, die Sie in weiteren Funktionen auslagern können. Dies vereinfacht für Sie die Aufgabenstellung.

3.1.1 Umfang

Schreiben Sie **überladene** Funktionen `umfang`, die jeweils eins der drei geometrischen Objekte als Parameter besitzen und den entsprechenden Umfang des Objektes berechnen und zurück geben. Überlegen Sie, ob Sie die Objekte sinnvollerweise als

- call-by-value
- call-by-reference
- oder call-by-const-reference

übergeben.

3.1.2 Fläche

Schreiben Sie analog zu Aufgabe 3.1.1 überladene Funktionen `flaeche` zur Berechnung der Fläche.

Hinweis: Die Fläche eines Dreiecks lässt sich auch aus den Seitenlängen berechnen: `http://de.wikipedia.org/wiki/Dreiecksfl%C3%A4che`.

4 Zufallsexperimente

Simulieren Sie den gleichzeitigen Wurf zweier „normaler" sechsseitiger Würfel und zählen Sie die beiden Würfelergebnisse zusammen.

Führen Sie 1.000.000 dieser Experimente durch und speichern die Ergebnisse geeignet ab. Geben Sie anschließend aus, wie häufig jedes Ergebnis erzielt wurde.

Als Beispiel wird hier eine Ausgabe für den Wurf **eines** Würfels gezeigt:

```
1 ->   166511
2 ->   166486
3 ->   166701
4 ->   166703
5 ->   166457
6 ->   167142
```

Hinweis: Der folgende Quellcode gibt 10 zufällig erzeugte Zahlen im Bereich 0 bis 5 aus:

```cpp
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;
int main()
{
  srand(time(0));
  for(int i = 0; i < 10; i++)
    cout << rand() % 6 << " ";
  cout << endl;
  return 0;
}
```

Dabei erzeugt `rand()` die eigentliche Zufallszahl, eine ganze Zahl zwischen 0 und (mindestens) 32.767. Die zusätzliche Modulo-Operation liefert Ihnen den Rest bei einer Division durch 6. Dieser Rest kann nur Werte zwischen 0 und 5 annehmen.
Der Aufruf von `srand(time(0))` sorgt dafür, dass der Pseudozufallszahlengenerator bei jedem Programmstart mit einem anderen Wert initialisiert wird. Der Aufruf von `time(0)` liefert die aktuelle Uhrzeit als Unix Timestamp.

Hinweis: Auf Moodle steht auch eine C++11-Variante des obigen Quellcodes zur Verfügung.

5 Durchgehende Aufgabe: Version 0.3

5.1 Klassen

`Lied` muss als Klasse implementiert sein.

5.2 Funktionen

Wenn noch nicht geschehen, lagern Sie die Menüpunkte in Funktionen aus. Es soll also eine Funktion für Menüpunkt 1, eine Funktion für Menüpunkt 2, usw. existieren.

Der Vektor zur Speicherung der `Lied`-Objekte **muss** eine lokale Variable der `main`-Funktion sein. Globale Variablen sind ab diesem Praktikum nicht mehr zulässig.
Übergeben Sie entsprechend den Menüfunktionen den `vector` als Referenz bzw. `const`-Referenz, je nachdem was sinnvoll ist.

5.3 Weitere Funktionalität

Implementieren Sie die Menüpunkte 3 & 4.

Geben Sie dazu, wenn noch nicht geschehen, bei Menüpunkt 6 („Alle Einträge anzeigen“) zusätzlich zum Songtitel noch eine fortlaufende Nummer (Index) ein. Fragen Sie den Benutzer bei Menüpunkt 3 bzw. 4 zunächst nach dieser Nummer. Der Eintrag mit dieser Nummer soll gelöscht bzw. bearbeitet werden.

5.3.1 Löschen eines Eintrags

Überlegen Sie, wie ein Eintrag aus einem `vector` zu löschen ist. Eine Möglichkeit besteht darin, alle Folgeeinträge um eine Stelle nach links zu verschieben. Sie können auch gerne in einer C++-Referenz[1] recherchieren. Wenn Sie unbekannte, nicht in der Vorlesung diskutierte Funktionalitäten verwenden, müssen Sie diese bei Programmabgabe erklären können!

5.3.2 Bearbeiten eines Eintrags

Hier soll nach Auswahl der Nummer des zu bearbeitenden Eintrags gefragt werden, welche Information des Eintrags geändert werden soll. Anschließend gibt der Benutzer den neuen Wert ein, und wird wieder zurück zum Hauptmenü geführt.

[1] z.B. unter `https://cplusplus.com`