h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Programmieren, Algorithmen und Datenstrukturen I

Praktikum 3: Kontrollstrukturen und komplexe Datentypen

Sommersemester 2021

Prof. Dr. Arnim Malcherek

Allgemeine Hinweise zum Praktikum:

- Bereiten Sie die Aufgaben unbedingt vor dem Praktikumstermin vor. Das beinhaltet:
  - Entwurf der Lösung
  - Codieren der Lösung in einem Qt-Creator-Projekt
- Die Zeit während des Praktikums dient dazu, die Lösung testieren zu lassen sowie eventuelle Korrekturen vorzunehmen.
- Das Praktikum dient auch zur Vorbereitung der praktischen Klausur am Ende des Semesters. Versuchen Sie also in Ihrem eigenen Interesse, die Aufgaben selbständig nur mit Verwendung Ihrer Unterlagen bzw. Ihres bevorzugten C++-Buches und ohne Codefragmente aus dem Netz zu lösen.
- Die Lösung wird nur dann testiert, wenn
  - sie erklärt werden kann bzw. Fragen zur Lösung beantwortet werden können.
  - das Programm ablauffähig und die Lösung nachvollziehbar ist.
  - die Hinweise oder Einschränkungen aus der Aufgabenstellung befolgt wurden.
- Zur Erinnerung hier noch einmal die Regeln des Praktikums, die schon in der Vorlesung besprochen wurden:
  - Sie arbeiten in 2er Gruppen.
  - Ein Testat gibt es nur zum jeweiligen Termin.
  - Abschreiben und Kopieren ist verboten.
  - Es gibt keine Noten. Die Bewertung ist lediglich erfolgreich / nicht erfolgreich.
  - Das Praktikum ist Zulassungsvoraussetzung für die Klausur. Hierfür müssen alle fünf Praktikumsübungen testiert sein.

Lernziele:

- Sie können sicher mit den verschiedenen Kontrollstrukturen von C++ umgehen.
- Sie beherrschen Aufzählungstypen, Arrays und Vektoren, und können aus diesen Typen geeignete benutzerdefinierte Typen für Anwendungsbeispiele erzeugen.
- Sie beherrschen die Datentypen zur Zeichenverarbeitung, und kennen die Unterschiede zwischen einem `char`-Array und einem `string`.

Aufgabe 1

Schreiben Sie eine Funktion, die die Exponentialfunktion e^x über ihre Reihendarstellung berechnet:

$$f(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

Erinnerung: ! steht für die Fakultätsfunktion.

Verwenden Sie für die Reihenberechnung lediglich Schleifen und Grundrechenarten aber keine Bibliotheksfunktionen wie `pow` oder `exp`. Brechen Sie die Berechnung dann ab, wenn der Fehler im Vergleich zur exakten Berechnung kleiner als ein vom Benutzer vorgegebener Wert ist. Der Fehler sei dabei definiert als:

$$\delta = \frac{|f(x) - e^x|}{e^x}$$

Für den exakten Wert von e^x dürfen Sie die Funktion `exp()` und für den Absolutwert bei der Fehlerberechnung die Funktion `abs()` aus der Mathematikbibliothek `cmath.h` nutzen (einbinden über `#include <cmath>`).

Fragen Sie `x` und den erlaubten Fehler vom Benutzer ab und übergeben Sie beides an die Funktion. In der Funktion geben Sie den Wert der Reihenberechnung, den exakten Wert der Exponentialfunktion, den Fehler und die Anzahl der Schleifendurchläufe aus.

Beispielausgabe:

```
Geben Sie das Argument ein: 3
Geben Sie den maximal zugelassenen Fehler ein: 0.001

Reihendarstellung: 20.0797
Exakter Wert: 20.0855
Fehler: 0.000292337
Anzahl Summanden: 11
```

Aufgabe 2

Programmieren Sie ein vereinfachtes Schachspiel.
Teilaufgaben:

- Der Spielplan sei ein 8*8 Array vom Datentyp `string`. Ob Sie für das Array den Datentyp `array` oder `vector` verwenden, bleibt Ihnen überlassen.
- Die zulässigen Werte für die Strings (z.B. wB, sB, wL, sL, wS, sS, wT, sT, wD, sD, wK, sK) stehen für die verschiedenen Schachfiguren (wB für weißer Bauer usw.). Definieren Sie diese Bezeichnungen entweder über `string`-Konstanten oder über einen Aufzählungstypen (`enum`). Wählen Sie selbst aus, welchen Datentyp Sie für geeigneter halten.
- Setzen Sie für das Array die Anfangsaufstellung eines Schachspiels (siehe Abbildung 1).
- Geben Sie diese Anfangsaufstellung aus, indem Sie das Array zeilenweise am Bildschirm ausgeben (möglichst lesbar formatiert, siehe Beispielausgabe in Abbildung 2).
- Implementieren Sie mögliche Spielzüge durch den Benutzer in einer `while`-Schleife mit Abbruch bei Eingabe des Wortes `exit`. Ein Spielzug sei z.B. die Eingabe von `b1-c3`, was dazu führen sollte, dass der weiße Springer seine Position auf `c3` ändert usw..
- Geben Sie die neue Aufstellung nach jedem Spielzug aus, und fragen Sie den nächsten Spielzug ab, so dass zwei Spieler Ihr Programm für eine echte Partie nutzen können.

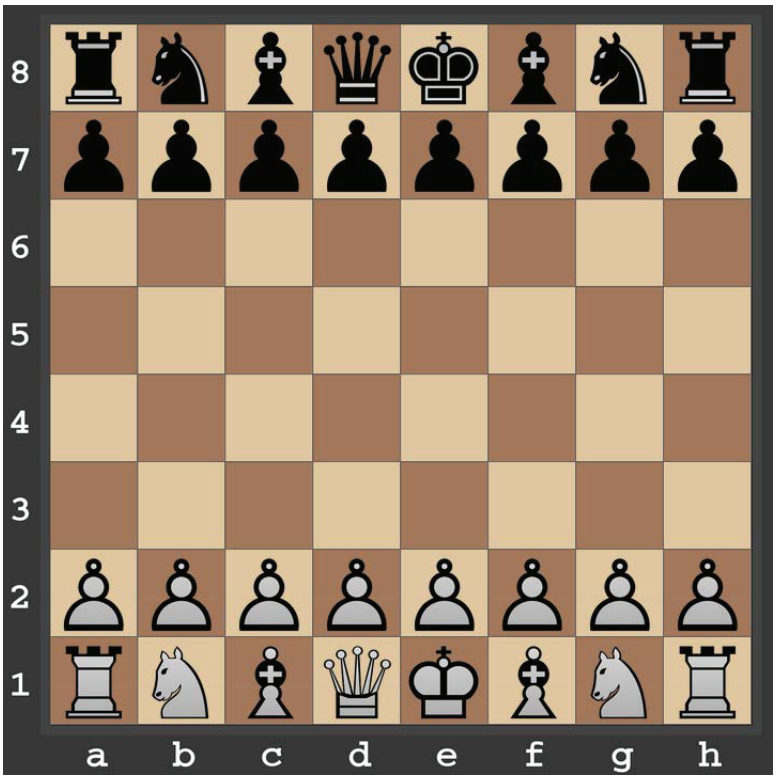

Abbildung 1: Schachaufstellung

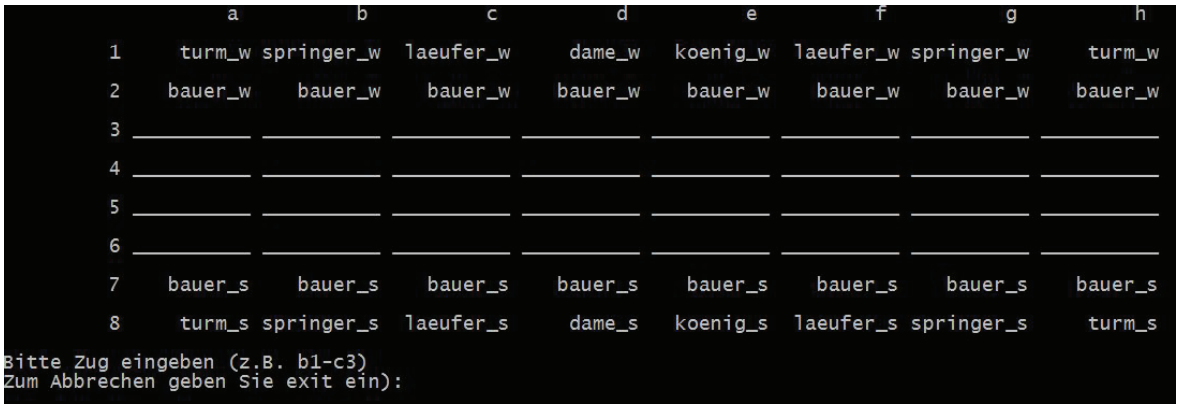

Abbildung 2: Beispielausgabe

Aufgabe 3

Lesen Sie über die Tastatur einen String der Maximal-Länge 5 ein, der nur aus Ziffern bestehen darf (Prüfung nach der Eingabe). Benutzen Sie eine `string`-Variable zur Speicherung. Wandeln Sie diesen String in eine Zahl vom Typ `long` um, berechnen Sie das Quadrat dieser Zahl und geben Sie es aus. Speichern Sie die einzelnen Ziffern des Resultats anschließend in einem `char array`, wobei sie zwischen den Ziffern jeweils eine Leerstelle lassen und geben Sie dieses Array aus. Bei dieser Aufgabe ist die Verwendung von Bibliotheksfunktionen wie z.B. `atol` oder `toString` nicht erlaubt.

Beispielausgabe:

```
Bitte Zahlenstring eingeben: 11222j


String enthaelt nicht nur Ziffern, bitte neu eingeben.
Bitte Zahlenstring eingeben: 1297343


String zu lang, bitte neu eingeben
Bitte Zahlenstring eingeben: 1112


Quadrat der eingegebenen Zahl: 1236544
char-Array: 1 2 3 6 5 4 4
```

Aufgabe 4

Jogi Löw hat die folgenden 22 Spieler für die Fußball EM in seinen Kader berufen:

Tor: Neuer, ter Stegen, Trapp

Abwehr: Klostermann, Ginter, Can, Gosens, Ruediger, Halstenberg, Suele

Mittelfeld: Kimmich, Can, Havertz, Goretzka, Guendogan, Kroos, Mueller, Neuhaus

Angriff: Gnabry, Younes, Sane, Werner

Helfen Sie ihm bei der Aufstellung, indem Sie 11 Spieler auswählen.
Überlegen Sie sich geeignete Datentypen für die Mannschaftsteile, und stellen Sie dann mit Hilfe des Zufallsgenerators `rand()` aus der C++ Standardbibliothek eine sinnvolle

Mannschaft zusammen unter der Annahme, dass auf dem Platz 1 Torwart, 4 Abwehrspieler, 4 Mittelfeldspieler und 2 Angreifer stehen sollen. Natürlich müssen es 11 verschiedene Spieler sein.
Wie viele verschiedene Aufstellungen sind theoretisch möglich, die diese Rahmenbedingung erfüllen? Berechnen Sie diese entweder über eine Formel, oder zählen Sie sie unter der Zuhilfenahme von Schleifen.

Beispielausgabe:

```
Tor: Neuer
Abwehr: Ginter, Boateng, Hummels, Hector
Mittelfeld: Goretzka, Kroos, Mueller, Rudy
Angriff: Werner, Sane
Es sind x verschiedene Aufstellungen möglich.
```

Tipp: `rand()%4` liefert eine Zufallszahl zwischen 0 und 3, `rand()%5` liefert eine Zufallszahl zwischen 0 und 4 usw.. Außerdem müssen Sie den Zufallsgenerator zu Beginn des Programmes noch initialisieren. Das geht über die Anweisung `srand(time(NULL))`, wofür Sie die Bibliothek `ctime.h` über die Anweisung `#include <ctime>` einbinden müssen.