C++ für Promovierende

(Übungsaufgaben)
15. März 2017

Unterlagen zum Seminar finden Sie auf <https://www.uni-trier.de/index.php?id=59534>.

Aufgabe 1 (Strings):

Schreiben Sie ein Programm, welches ein Wort einliest und zählt, wie oft in diesem Wort die Buchstabenkombination „ei“ vorkommt.

Erweitern Sie das Programm so, dass es eine vom Benutzer eingegebene Buchstabenkombination zählt.

Aufgabe 2 (Arrays):

Schreiben Sie ein Programm, welches ein Array `int zahlen[10]` mit Zahlen (Benutzereingabe oder Zufallszahlen) füllt und dann den Mittelwert bestimmt.

Aufgabe 3 (Funktionen):

Implementieren Sie sowohl die *Fakultäts-* als auch die *Fibonacci*funktion. Im Hauptprogramm soll der Benutzer gefragt werden, welche der beiden Funktionen er gerne berechnen möchte und für welche Zahl n . Nach der Berechnung soll die Lösung ausgegeben werden.

Zur Erinnerung:

$$n! = \begin{cases} 1, & n = 0 \\ n \cdot (n - 1)!, & n > 0 \end{cases}$$
$$f_n = \begin{cases} 1, & n \leq 2 \\ f_{n-1} + f_{n-2}, & n > 2 \end{cases}$$

Aufgabe 4 (Funktionsüberladung):

Schreiben Sie eine Funktion `max(a, b)`, die für *Integer* a, b die größere der beiden und für *Strings* a, b den längeren der beiden zurückgibt.

Aufgabe 5 (Parameterübergabe):

Schreiben Sie eine Funktion `void triple()`, die eine aus dem Hauptprogramm übergebene Zahl verdreifacht.

Falls Sie andere Programmiersprachen beherrschen: Wie würden Sie diese Funktion dort implementieren?

Aufgabe 6 (Adressen im Speicher):

Initialisieren Sie mehrere Variablen sowohl gleicher als auch verschiedener Typen und geben Sie die jeweiligen Adressen aus. Was fällt Ihnen auf?

(*Hinweis*: `sizeof(x)` gibt die Speichergröße des zu x gehörigen Datentyps zurück.)

Aufgabe 7 (Klassen):

Schreiben Sie eine Klasse *Kreis* mit Daten

- `int x` – Position des Kreises
- `int y` – Position des Kreises
- `double r` – Radius

und folgenden Funktionen (neben den Konstruktoren)

- `void move(int int)` – verschiebt den Kreis
- `double umfang()` – berechnet den Umfang
- `double flaeche()` – berechnet die Fläche
- `void scale(double)` – skaliert den Radius

(*Hinweis*: Nutzen Sie für π direkt 3.14 anstatt die Konstante über eine passende Bibliothek einzubinden.)

Aufgabe 8 (Vererbung):

Schreiben Sie eine Klasse *Kugel*, die die Variablen und Methoden der Klasse *Kreis* aus der vorigen Aufgabe erbt. Erweitern Sie die Klasse wie folgt:

- `int z` – Position der Kugel
- `double volumen()` – berechnet das Volumen
- `double oberflaeche()` – berechnet die Kugeloberfläche
- `void move(int int int)` – verschiebt die Kugel nun in drei Dimensionen

(*Hinweis*: Nutzen Sie für π direkt 3.14 anstatt die Konstante über eine passende Bibliothek einzubinden.)

Aufgabe 9 (Klassen-Templates):

Schreiben Sie eine Klassen-Template `stack<T>` mit Daten

- `stack_elem<T>* top` – Verweis auf das oberste Element des Stacks (schreiben Sie hierzu auch ein Klassen-Template `stack_elem<T>`, welches einen Wert vom Typ T und einen Verweis auf das nächste Element speichert)
- `int size` – Anzahl der Elemente im Stack

und folgenden Funktionen (neben den Konstruktoren)

- `bool is_empty()` – gibt an ob der Stack leer ist
- `stack_elem<T> pop()` – entfernt das oberste Element vom Stack und gibt es zurück
- `void push(stack_elem<T>)` – legt ein Element oben auf dem Stack ab

Aufgabe 10 (Galgenmännchen):

Implementieren Sie die folgende Variante des Spiels *Galgenmännchen*:

- Zu Beginn gibt ein Spieler ein Wort ein
- Das Programm „merkt“ sich das Wort und löscht die Eingabe vom Bildschirm (bspw. durch Ausgabe von genügend Leerzeilen, so dass die Eingabe verschwindet)
- Das Programm schreibt für jeden Buchstaben des ursprünglichen Worts einen Strich (Bsp. „Hallo“ $\mapsto$ „-----“)
- Der andere Spieler darf nun einzelne Buchstaben eingeben
- Falls der eingegebene Buchstabe im Wort vorkommt, dann soll der Strich (die Striche), die zu dem Buchstaben gehören, durch den Buchstaben selbst ersetzt werden (Bsp. nach Eingabe von „l“: „--ll--“)
- Falls der eingegebene Buchstabe nicht im Wort vorkommt, so soll ein Zähler erhöht werden, der die Fehlversuche zählt
- Der zweite Spieler gewinnt, wenn er alle Buchstaben aufdeckt
- Der erste Spieler gewinnt, wenn der zweite Spieler nach 10 Fehlversuchen noch nicht alle Buchstaben aufgedeckt hat

Aufgabe 11 (Queue):

Schreiben Sie eine Klasse `item`, welche eine Ganzzahl `value` und einen Verweis `next` (auf ein Element vom Typ `item`) speichert.

Schreiben Sie zusätzlich eine Klasse `queue` mit den folgenden Variablen:

- `item* front` – Speichert das vorderste Element der Queue
- `item* back` – Speichert das vorderste Element der Queue
- `int count` – Speichert die aktuelle Größe der Queue

Zusätzlich soll die Klasse die folgenden Funktionen (neben dem Konstruktor) bereitstellen:

- `void enqueue(int x)` – Fügt ein Element mit Wert `x` am Ende der Queue ein
- `item* dequeue()` – Liefert das vorderste Element der Queue zurück
- `bool is_empty()` – Testet, ob die Queue leer ist
- `int size()` – Gibt die Anzahl der Elemente in der Queue zurück
- `void print()` – Gibt die Elemente der Queue nacheinander aus