Klausur zum C++ Programmierkurs SS 2019

Daniel Kondermann, Thorsten Buss

15.04.2019

Die Klausur besteht aus Aufgaben in denen 90 Punkte erreicht werden können. Um die Prüfungsleistung für diesen Kurs zu erbringen, müssen 36 Punkte erreicht werden.
Ende der Klausur: 12:45 Uhr (Danach 15 Minuten Zeit für Abgabe)

Richtlinien (bitte sorgfältig lesen!)

- Als **Hilfsmittel** ist alles erlaubt, außer zu kommunizieren (Beispiele: Chat, Mail, SMS, ...).
- Am Ende der Klausur müssen Sie den Code für alle Aufgaben abgeben.
- Jede Datei, die Sie zu dem Archiv hinzufügen, muss **Ihren Namen als Kommentar in der ersten Zeile** enthalten.
- **Kommentieren Sie Ihren Programm-Code!!!** Fehlerhafter, unkommentierter Code lässt keine Rückschlüsse darauf zu, was Sie gemeint haben könnten. Falls sich jedoch zeigen sollte, dass Sie auf dem richtigen Weg waren, können wir dafür einen Teil der Punkte vergeben.
- Die Abgabe soll in Form **eines** Zip-Archivs erfolgen, welches ihre Quell-Dateien enthält.
- Das Archiv soll **pro Aufgabe ein Unterverzeichnis** mit dem Namen **aufgabe4** enthalten (die 4 ersetzen durch die Aufgabennummer)!
- Dateien sollen entweder auf **.h .cpp oder .txt enden**!
- Das Archiv selbst sollte heißen: **NachnameVorname.zip**
- Sie können ein Archiv auch mehrfach hochladen. Wenn Sie bereits ein Archiv hochgeladen haben, erzeugen Sie einfach ein neues Archiv mit einem anderen Namen, z.B. KondermannDaniel2.zip. Wir werden nur das zuletzt hochgeladene Archiv ansehen. Achten Sie daher darauf, dass in der von Ihnen zuletzt hochgeladenen Archiv-Datei **alle Quell-Dateien** vorhanden sind, die Sie abgeben wollen.

Viel Erfolg!

Aufgabe 1 – Input/Output/Schleifen (3+7 = 10 P)

a) Schreiben Sie ein Programm, das nach dem Namen und dem Alter des Benutzers fragt und anschließend ausgibt „Hallo <*BENUTZER*>, du bist <*ALTER*> alt!“. Dabei sollte <*BENUTZER*> ersetzt werden durch den eingegebenen Namen, sowie <*ALTER*> durch das eingegebene Alter.

b) Das Programm soll prüfen, ob das Alter Zeichen enthält, die keine Ziffern sind. Falls das der Fall ist, soll es „Dein Alter enthält Zeichen, die keine Ziffern sind.“ ausgeben. Anderenfalls soll es die Quersumme des Alters berechnen und „Die Quersumme deines Alters ist <Quersumme>“ ausgeben.

Hinweis:

- Die Quersumme einer Zahl ist die Summe ihrer Ziffern. Beispielsweise ist die Quersumme von 4037 gleich 4 + 0 + 3 + 7 = 14.

Aufgabe 2 – Funktionen (10+5 = 15 P)

a) Schreiben Sie eine Funktion *myExp*, die als Parameter einen double erhält. Die Funktion soll die natürliche Exponentialfunktion mittels ihrer Exponentialreihe annähern und das Ergebnis als double zurückgeben. Die Berechnung soll nach dem 51. Summanden abbrechen.

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} \approx \sum_{n=0}^{50} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^{50}}{50!}$$

b) Schreiben Sie die Main-Funktion. In dieser soll für x = -1, - 0.9, … 0.9, 1 jeweils „x=<x>; y=<exp(x)>“ ausgeben werden. Hierbei ist <x> jeweils durch den Wert von x und <exp(x)> durch den angenäherten Wert der natürliche Exponentialfunktion an der Stelle x zu ersetzen.

Aufgabe 3 – Fehlersuche I (10x2 = 20 P)

Laden Sie aufgabe3.cpp herunter. Finden Sie für jeden Code-Abschnitt (markiert durch geschweifte Klammern) mindestens einen Fehler und kommentieren Sie, was der Fehler ist.

Interpretieren Sie jeden Abschnitt, als befände sich der Code in der main() Funktion.

Nehmen Sie an, dass alle nötigen #includes z.B. <iostream> oder <vector> bereits stattfanden.

Aufgabe 4 – Fehlersuche II (2+3 = 5 P)

Das angehängte Programm aufgabe4.cpp hat einen Fehler. Finden Sie diesen (2 Punkte), und erklären sie in 1-3 Sätzen, warum dies ein Fehler ist und wie man diesen korrigieren würde (3 Punkte).

Aufgabe 5 – Bauernhof (40P)

a) (2P) Richten Sie sich ein Projekt ein, dass eine Terminalanwendung erlaubt und dafür die Terminal-Klasse aus dem OOP-Snake Programm verwendet.

b) (3P) Schreiben Sie eine main.cpp, die die Game-Loop aus dem OOP-Snake-Programm verwendet. Achten Sie darauf, dass keine Snake-spezifischen Reste im Code übrig bleiben.

c) (10P) Definieren sie eine abstrakte Basisklasse Tier mit der rein virtuellen Funktion draw() (kein Rückgabewert). Die Basisklasse erhält im Konstruktur eine Terminal-Referenz, die als public Member Referenz in der Klasse gespeichert wird.

d) (10P) Leiten sie öffentlich zwei Klassen von Tier ab: Schwein und Katze. Implementieren sie für beide eine andere draw-Methode, die mehr als ein Zeichen ausgibt, sie aber nicht zu viel Zeit kostet.

e) (5P) Beim Drücken der Taste s soll ein Schwein an einer beliebigen Stelle auf dem Bild erzeugt werden; beim Drücken der Taste k eine Katze.

f) (5P) Das jeweils zuletzt erzeugte Objekt soll mit den Tasten WASD im Bild bewegt werden können. Markieren sie das aktuell bewegbare Tier mit einem X in der oberen linken Ecke.

g) (5P) Durch drücken von o bzw. l kann das aktive Tier geändert werden, indem das zuvor erzeugte (o) oder danach erzeugte (l) Tier mit WASD bewegbar gemacht werden kann. Wenn beim zuletzt erzeugten Tier l gedrückt wird, soll das zuerst erzeugte Tier selektiert werden.

Hinweise:

- Sie können die Funktionalität aus bereits abgegebenen Aufgaben wiederverwenden.
- Verwenden Sie Polymorphie, indem Sie die eine **rein** virtuelle Funktion draw() in der abstrakten Basisklasse definieren und diese in den Ableitungen implementieren.
- Speichern Sie die Liste der bereits erzeugten Tiere in einem STL vector.