Hochschule Darmstadt
Prof. Dr. Benjamin Meyer

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

7. Januar 2021

Programmieren, Algorithmen und Datenstrukturen I, WS 2020/21
Zwischenprüfung

1 Einleitung

Diese Zwischenprüfung dient für Sie als weitere Vorbereitung für die Klausur und soll Ihnen helfen einzuschätzen, wie gut Sie bereits mit dem bisherigen Vorlesungsinhalt vertraut sind. In diesem Semester ist die Teilnahme an dem Test komplett freiwillig, **Sie können weder durchfallen noch wird der Test benotet**. Versuchen Sie daher, sich strikt an die **Spielregeln** zu halten: Sie haben nur **90 Minuten** Zeit für die Bearbeitung und dürfen **keine Internet-Quellen** nutzen (nur das Vorlesungsskript von Prof. Altenbernd und eigene Mitschriften).
Bearbeiten Sie die Aufgaben alleine, sorgfältig und in der gegebenen Reihenfolge, der Schwierigkeitsgrad nimmt von oben nach unten zu. Ähnlich der finalen Klausur bauen die Aufgaben aufeinander auf und das fehlerfreie Bearbeiten der ersten Teilaufgabe würde zum Bestehen reichen, die restlichen Aufgaben würden die Endzensur um je einen Notenschritt verbessern. Wir werden morgen die Lösungen besprechen.

Viel Erfolg!

1.1 Anlegen einer Datenbank einer Bank

Ein kleines Kreditinstitut hat aktuell fünf Kunden und möchte diese in einer passenden Datenbank speichern. Jeder Kunde wird durch den Namen (Nachname, Vorname), das Geschlecht (m/w/d) sowie eine eindeutige Kontonummer (im Bereich 1111111 - 9999999) gekennzeichnet und verfügt über ein bestimmtes Guthaben bei der Bank. Die Kundendaten sind in der folgenden Tabelle gegeben:

Kundennr.	Name	Geschlecht	Kontonummer	Guthaben
1	Pietersen, Pietra	weiblich	2431109	-17928.90
2	Modaal, Maria	divers	3102938	12000.00
3	Klakson, Klara	weiblich	2439101	4938.02
4	Joskens, Jos	männlich	2555101	-4500.00
5	Janssen, Jan	männlich	2244887	13465.30

Tabelle 1: Kundenliste

Starten Sie mit einem leeren Projekt. Erstellen Sie ein `struct` zur Verwaltung eines einzelnen Kunden sowie ein `enum` für das Geschlecht. Eine Aufteilung des Quellcodes in eine Header- und eine Source-Datei (.h und .cpp) steht Ihnen frei, ist in dieser Prüfung jedoch noch nicht Pflicht.
Legen Sie in der `main()` eine Kundendatenbank an und füllen Sie diese mit den fünf gelisteten Personen. Überlegen Sie sich, ob Sie einen Vektor oder ein Array zum Speichern der Daten nutzen wollen. Begründen Sie Ihre Entscheidung!

Die Bank benötigt zudem ein kleines Interface, um auf die gespeicherten Daten zugreifen zu können. Fragen Sie hierfür eine Integer-Tastatureingabe ab, die einer Kundennummer (also 1 bis n bei n Einträgen in der Datenbank) entsprechen soll. Finden Sie den entsprechenden Eintrag und geben Sie Namen, Geschlecht sowie Kontostand aus. Achten Sie auf ungültige Eingaben! Die Eingabe soll solange fortgesetzt werden können, bis als Abbruchkriterium eine -1 (oder jede andere negative Zahl) eingegeben wird.

1.2 Suchen einer Kontonummer

Als erste Erweiterung der Datenbank sollen einzelne Kunden anhand ihrer Kontonummer gefunden werden können.
Schreiben Sie daher eine neue Methode `getCustomer()`, die eine Kontonummer und eine Referenz auf die Datenbank erhält und die Datenbank durchsucht, bis die entsprechende Kontonummer gefunden wurde. Achten Sie auf falsche Kontonummern, bevor Sie die Datenbank durchsuchen! Geben Sie die den dazugehörigen Kunden als Ergebnis zurück. Wurde die Kontonummer nicht im System gefunden, geben Sie eine Fehlermeldung aus und einen leeren Kunden zurück.
Rufen Sie die Methode in der `main()` auf und suchen Sie die Kontonummer 2555101. Geben Sie das Ergebnis sinnvoll auf der Konsole aus.

1.3 Alphabetisches Sortieren

Zur leichteren Verwaltung sollen im nächsten Schritt die Einträge der Datenbank alphabetisch von A bis Z anhand des Nachnamens sortiert werden.
Implementieren Sie hierfür den Insertion-Sortieralgorithmus (der Quellcode ist im Skript gegeben, Kapitel 3) in einer neuen Methode `sortDatabase()`. Als Vergleichswert soll der jeweilige Name des Kunden dienen. Geben Sie vor und nach dem Sortieren in der `main()` die Namen der Datenbank aus.
Überlegen Sie sich schriftlich/im Kopf, wie viele Vergleichsoperationen die Methode ausführen muss. Wie sieht allgemein die geschätzte Laufzeit des Insertion-Sortieralgorithmus' aus, wie viele Vergleichsoperationen würden Sie daher bei einer Liste mit 5 Elementen normalerweise (im Durchschnitt) erwarten? Wie passt das zu den gezählten Vergleichsoperationen und woran liegt das?

1.4 Zinseszins

In der letzten Programm-Erweiterung soll eine Prognose für die Wertentwicklung der kommenden Jahre berechnet werden (falls keine Ein- oder Auszahlungen mehr vorgenommen werden).
Implementieren Sie hierfür die Methode `compoundInterest()`, die für die nächsten n Jahre die Wertentwicklung mittels Zinseszins berechnet. Die Formel für den Zinseszins sieht wie folgt aus:
Guthaben nach n Jahren = Guthaben $\cdot$ (1+ Zinssatz/100)^n
Die Bank gibt für positive Guthaben dabei 2 Prozent Zinsen pro Jahr, verlangt umgekehrt für Kredite (also negative Guthaben) einen Aufschlag von 5 Prozent pro Jahr. Geben Sie die geschätzten Auslagen der Bank (also das Gesamtguthaben aller Kunden) nach n Jahren zurück.
Rufen Sie die Methode in der `main()` auf, wie hoch ist das Gesamtguthaben auf der Bank nach 5 Jahren?