Dirk Praetorius,
Marcus Page,
Thomas Führer,
Josef Kemetmüller

Sommersemester 2013
28.05.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 10

Aufgabe 10.1. Schreiben Sie eine Klasse `University`. Diese soll neben den Feldern `anzStudents`, `city` und `name` die Methoden `graduate` und `newStudent` haben. Wird `graduate` aufgerufen, so verringert sich die Anzahl der Studenten um 1, wohingegen `newStudent` die Anzahl um 1 erhöht. Alle Datenfelder sollen als `private` deklariert sein. Sie müssen sich also zusätzlich `get`- und `set`-Methoden schreiben. Speichern Sie den Source-Code unter `University.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.2. Für die Personalabteilung der Universität ist es sehr mühsam, Studenten immer nur einzeln hinzuzufügen oder aus dem System zu löschen. Schreiben Sie also Funktionen denen die Anzahl der abschließenden bzw. neu hinzukommenden Studenten mit übergeben werden kann. (Hinweis: Wie bei Konstruktoren können Sie hierfür die gleichen Funktionsnamen `graduate` und `newStudent` verwenden, solange die Funktionen andere Signatur erhalten. Dieses Vorgehen nennt man *Funktionen überladen*.) Speichern Sie den Source-Code unter `University.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.3. Schreiben Sie eine Klasse `Stoppuhr` welche zur Simulation einer Stoppuhr dienen soll. Die Stoppuhr bestehe dabei aus zwei Knöpfen. Wird der erste Knopf gedrückt, so soll die Zeitmessung gestartet werden. Wird dieser Knopf nochmals gedrückt, wird die Zeitmessung gestoppt. Der zweite Knopf dient dazu die Zeit wieder zurückzusetzen. Schreiben Sie dazu die Methoden `pushButtonStartStop` und `pushButtonReset`. Implementieren Sie weiters eine Methode, welche die verstrichene Zeit im Format `hh:mm:ss.xx` ausgibt. (Beträgt die gemessene Zeit also zwei Minuten so soll `00:02:00.00` ausgegeben werden) Sie können diese Stoppuhr nun dazu verwenden Zeitmessungen durchzuführen. Speichern Sie den Source-Code unter `stoppuhr.{hpp,cpp}` in das Verzeichnis `serie10`.
Hinweis: Verwenden Sie den Datentyp `clock_t` und die Funktion `clock()` aus der Bibliothek `time.h`. Vermutlich ist es auch sinnvoll eine Variable `isRunning` vom Typ `bool` einzuführen. Bei Betätigen des ersten Knopfes wird diese Variable entweder von `false` auf `true` gesetzt oder umgekehrt.

Aufgabe 10.4. Erstellen Sie eine Klasse `Name` welche zwei String-Variablen `vorname` und `nachname` enthält. Implementieren Sie auch die Zugriffsfunktion `setName`, welche einen String übernimmt, diesen in Vor- und Nachname aufteilt und in die entsprechenden Variablen abspeichert. Beachten Sie auch, dass mehrere Vornamen vorhanden sein können! Schreiben Sie weiters eine Methode `printName` die Vor- und Nachname am Bildschirm ausgibt. Bei mehreren Vornamen soll, beginnend ab dem zweiten Vornamen, jeder weitere Vorname mit dem Anfangsbuchstaben und einem Punkt abgekürzt werden! Beim Namen `Max Maxi Mustermann` soll also `Max M. Mustermann` am Bildschirm erscheinen. Speichern Sie den Source-Code unter `name.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.5. Der Verschlüsselungsalgorithmus ROT13 ersetzt jeden Buchstaben der Eingabe zyklisch durch den im Alphabet um 13 Buchstaben weiter hinten (oder vorne) stehenden Buchstaben. Die Buchstaben A, B, C, ... werden auf N, O, P, ... abgebildet, die Buchstaben N,O,P, ... auf A, B, C, ... Setzen Sie diesen Algorithmus in der Funktion `rot13` um. Ihre Funktion braucht nur für Großbuchstaben zu funktionieren.
Schreiben Sie nun eine weitere Funktion `rot_n`, der Sie zusätzlich übergeben können, um wieviele Stellen die Eingabe rotiert werden soll. Überlegen Sie sich, wie man mit `rot_n` verschlüsselte Texte entschlüsseln kann. Schreiben Sie dazu eine Funktion `unrot_n`.
Hinweis: Ein Zeichen ist vom Typ `char` und wird intern durch eine Zahl dargestellt, `'A'` beispielsweise durch 65. Ziehen Sie von einem Buchstaben `'A'`, ..., `'Z'` den Buchstaben `'A'` ab, so ist das Ergebnis intern eine Zahl zwischen 0 und 25. Führen Sie dort die notwendigen Rechenoperationen durch und addieren Sie anschließend wieder den Buchstaben `'A'` hinzu! Weitere Informationen zu ASCII finden Sie auf Wikipedia. Speichern Sie den Source-Code unter `rot.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.6. Schreiben Sie eine Klasse `Hangman` mit den Methoden `guessChar`, `solve` und `newString`. Die Klasse soll nun einen string der Länge n speichern, den es zu erraten gilt. Nach und nach dürfen mittels `guessChar` Buchstaben geraten werden, wobei die Methode immer den Index, bzw. die Indizes der eingegebenen Buchstaben auf dem Bildschirm ausgibt. Falls der eingegebene Buchstabe im gesuchten Wort nicht vorkommt, soll eine entsprechende Meldung ausgegeben werden. Der Spieler verliert, wenn er 8 mal falsch geraten hat. Schreiben Sie alle nötigen Zugriffsfunktionen und Konstruktoren und außerdem die Methoden `solve` und `newString` um das Rätsel zu lösen bzw. das Spiel mit einem neuen Wort neu zu beginnen. Testen Sie Ihr Spiel indem Sie mittels einer geeigneten Schleife solange neue Buchstaben raten, bis Sie entweder gewonnen oder verloren haben. Speichern Sie den Source-Code unter `hangman.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.7. Erstellen Sie eine Klasse `SparKonto` mit den Variablen `kontonummer`, `guthaben` und `zinssatz`. Ferner sollen noch die `get` und `set` Funktionen für die Variablen `zinssatz` und `kontonummer` implementiert werden. Um das Guthaben zu ändern schreiben Sie die Methoden `abheben` und `einzahlen`. Beachten Sie, dass Sie bei einem Sparkonto nicht ins Minus gehen können. Der Zinssatz und die Kontonummer dürfen natürlich auch nicht negativ werden. Schließlich implementiere man noch die Methode `berechneGuthaben`. Speichern Sie den Source-Code unter `sparkonto.{hpp,cpp}` in das Verzeichnis `serie10`.

Aufgabe 10.8. Als Kunde einer Bank kann man auch mehrere Sparkonten besitzen. Erstellen Sie eine Klasse `Kunde` welche eine Liste von Sparkonten beinhaltet. Benutzen Sie dazu den Container `vector`. Weiters soll die Klasse auch ein Objekt der Klasse `Name` aus Aufgabe 10.4 enthalten. Implementieren Sie Methoden zum Hinzufügen und Löschen von Konten. Schreiben Sie dann eine Methode die das Guthaben auf allen vorhandenen Sparkonten berechnet. Überlegen Sie welche Funktionen noch sinnvoll wären. Speichern Sie den Source-Code unter `kunde.{hpp,cpp}` in das Verzeichnis `serie10`.