Abteilung für
Angewandte Mathematik

SS 2023
Ludwig Striet
Tatjana Stiefken

Einführung in die Programmierung für Studierende der Naturwissenschaften

Blatt 7 – 12.06.2023

Abgabe: *bis Sonntag, 18.06.2023 24 Uhr per Mail an Ihren Tutor.*

Aufgabe 1 (2 + 3 Punkte + 3 Bonuspunkte)**.** In der Vorlesung und im Skript wurden die Fibonacci-Zahlen eingeführt.

(i) Schreiben Sie eine *rekursive* Funktion

```
unsigned int fibonacci_recu(unsigned int n)
{
}
```

die die $n$te Fibonacci-Zahl berechnet.

(ii) Schreiben Sie eine *iterative* Funktion

```
unsigned int fibonacci_iter(unsigned int n)
{
}
```

die $n$te Fibonacci-Zahl berechnet.

(iii) Messen Sie für $n = 1, \ldots, 35$ die Zeit, die die beiden Methoden benötigen. *Hinweis: Ein Beispiel, wie Sie die Laufzeit einer Funktion messen können finden Sie in der Datei* `stopwatch.cc` *auf der Vorlesungswebseite.*

Aufgabe 2 (5+5 Punkte)**.** Das Ziel dieser Aufgabe ist das Programmieren eines Vokabeltrainers.

a) Schreiben Sie ein Programm, mit dem der Benutzer eine Vokabelliste erstellen kann. Der Benutzer soll die Möglichkeit haben, Vokabeln einzulesen. Eine Vokabel ist hier ein Wortpaar, das aus dem Wort in der Ausgangssprache und dem Wort in der Zielsprache besteht. Nach jeder Eingabe soll das Programm fragen, ob weitere Vokabeln hinzugefügt werden sollen. Ist dies nicht der Fall, speichert das Programm alle Vokabeln in einer Textdatei, wobei jede Zeile genau eine Vokabel enthalten soll.

b) Schreiben Sie nun zweites ein Programm, mit dem Sie die Vokabeln aus der Liste lernen können. Dazu soll das Programm die Liste einlesen, dann ein zufälliges Wortpaar auswählen und das Wort in der Ausgangssprache ausgeben. Der Benutzer wird dann aufgefordert, das Wort in der Zielsprache einzugeben.
   - Ist die Eingabe korrekt, kann der Benutzer wählen, ob das Programm beendet oder ein neues zufälliges Wort abgefragt werden soll.

- Ist die Eingabe falsch, wird eine Fehlermeldung und das richtige Wort ausgegeben. Danach kann der Benutzer wieder entscheiden, ob er weiter lernen oder das Programm beenden möchte.

Hinweis: Sie können die Vokabeln in beiden Sprachen in zwei Vektoren `std::vector<std::string>` speichern, jeweils einer pro Sprache. Schöner ist es, für die Vokabeln eine eigene Klasse `Vokabel` anzulegen und Objekte dieser Klasse in einem `std::vector<Vokabel>` zu speichern. Beachten Sie, dass Sie nur Objekte von Klassen in einem `std::vector` speichern können, wenn diese eine einen Standardkonstruktor (ohne Argumente) hat, also

```
class Vokabel {
  public:
    Vokabel() { // Standardkonstruktor

    }

  // Weitere Implementierung, ggf. weiterer Konstruktor,
  // private Variablen, etc

};
```

Aufgabe 3 (5 Punkte)**.** Schreiben Sie eine Template-Funktion, die zwei Argumente desselben Typs annimmt und deren Summe (wieder vom selben Typ wie die Eingabe) zurückgibt.

Aufgabe 4 (optional)**.** Schildern Sie in einer Datei `erfahrung.txt` kurz Ihre Erfahrung mit dem aktuellen Übungsblatt. Berichten Sie darin wieder in Stichpunkten bzw. ein bis zwei kurzen Sätzen über Ihre Erfahrungen mit Kursinhalt und Übungsaufgaben. Was fiel Ihnen leicht? Was ist noch unklar? Wie viel Zeit haben Sie für die Bearbeitung der Hausaufgaben benötigt und welche Probleme traten dabei auf?