

## Übungen zur Vorlesung Einführung in das Programmieren für TM

### Serie 12

Die folgenden Beispiele erfordern teilweise, dass Sie sich mit Funktionen oder Containern der Standardbibliothek im Internet vertraut machen. Nutzen Sie dazu eine der beiden Internetseiten:

<http://www.cppreference.com> oder <http://www.cplusplus.com>.

**Aufgabe 12.1.** In C können Sie zu Zeigern Ganzzahlen hinzuaddieren. Verdeutlichen Sie sich dieses Konzept anhand von folgendem C-Code:

```
double foo(double* array, int dim) {
    double bar = 0.;
    double* begin = array;
    double* end = array + dim;
    double* ptr;
    for (ptr = begin; ptr != end; ++ptr)
        bar = bar + *ptr;
    return bar / (double) dim;
}
```

Was passiert, wenn man einen Zeiger und eine Ganzzahl addiert? Was berechnet die Funktion `foo`? Beantworten Sie diese Fragen, indem Sie obiges Programm für unterschiedliche Eingaben ausführen!

In C++ verallgemeinern Iteratoren bestimmte Eigenschaften von Zeigern auf C++-Standardcontainer wie `vector` oder `list`. Die Methoden `begin` und `end` liefern analog zu obigem Programmabschnitt Iteratoren auf das erste Element des Containers bzw. auf das erste Element außerhalb des Containers zurück. Schreiben Sie eine C++-Version der Funktion `foo`, die anstelle eines C-Arrays einen `vector<double>`-Container erhält und anstelle von Zeigern Iteratoren verwendet. Speichern Sie den Source-Code unter `iteratoren.cpp` in das Verzeichnis `serie12`.

*Hinweis:* Ist `v` vom Typ `vector<double>`, so könnten Sie mithilfe von Iteratoren mit

```
vector<double>::iterator it = v.begin();
it = it + 2;
cout << *it << endl;
```

das Element `v[2]` ausgeben.

**Aufgabe 12.2.** In Aufgabe 12.1 haben Sie die Funktion `foo` für den Container-Datentyp `vector<double>` implementiert. Schreiben Sie nun eine weitere Variante dieser Funktion, die jedoch als Eingabe anstelle eines Containers vom Typ `vector<double>` einen vom Typ `list<double>` erhält. Schreiben Sie noch eine weitere Variante unter Verwendung der Funktion `accumulate` aus der C++-Standardbibliothek. Binden Sie dazu die Header-Datei `numeric` ein. Speichern Sie den Source-Code unter `iteratorenList.cpp` in das Verzeichnis `serie12`.

*Hinweis:* Die Funktion

```
double accumulate(list<double>::iterator begin,
                  list<double>::iterator end,
                  double startwert);
```

addiert alle Einträge, die zwischen `begin` und `end` liegen zum Startwert `startwert` hinzu.

**Aufgabe 12.3.** Schreiben Sie eine Funktion `sortfile`, die eine Datei zeilenweise in einen Vektor einliest, diesen Vektor lexikografisch sortiert und anschließend ausgibt. Legen Sie eine geeignete Datei an, um Ihr Programm zu testen! Speichern Sie den Source-Code unter `sortfile.cpp` in das Verzeichnis `serie12`.  
*Hinweis:* Folgender Code liest eine Datei zeilenweise ein:

```
fstream datei("Dateiname.txt");
while (datei.good()) {
    string zeile;
    getline(datei, zeile);
}
```

Lösen Sie die Aufgabe, indem Sie diesen Code adaptieren. Binden Sie die Header-Datei `fstream` ein! Hinweis: Der Vergleichsoperator `<` für `strings` entspricht genau der lexikografischen Ordnung. Es gilt also für `strings a, b` genau dann `a < b`, wenn `a` in der Ausgabe vor `b` kommen soll. Sie können die Funktion `sort` aus der Standardbibliothek verwenden.

**Aufgabe 12.4.** Implementieren Sie die Funktionen `unique` und `checkPalindrom` aus der Übungsserie 9 mithilfe der Standardbibliothek. Hinweis: Verwenden Sie die Funktionen `sort` und `unique` um `unique` zu realisieren. Verwenden Sie `equal` und einen reverse iterator (siehe `rbegin`) um `checkPalindrom` zu realisieren.

**Aufgabe 12.5.** Ein Paar ist ein C++-Datentyp, der zwei Werte möglicherweise unterschiedlichen Typs enthält. Ein Paar von `double`-Werten wird beispielsweise mit `pair<double,double>` bezeichnet. Ein Paar von Werten wird beispielsweise mithilfe des Aufrufs `pair<double,double>(5.,3.)` erstellt. Um Paare zu verwenden müssen Sie die Header-Datei `map` einbinden!

Schreiben Sie eine Funktion `minmax`, die eine Liste von Gleitkommazahlen erhält, das Minimum und Maximum dieser Liste ermittelt und in Form eines Paares zurückgibt! Geben Sie das Minimum und Maximum in der `main`-Funktion aus! Speichern Sie den Source-Code unter `minmax.cpp` in das Verzeichnis `serie12`.

*Hinweis:* Auf den ersten Wert eines Paares kann mit `.first` auf den zweiten mit `.second` zugegriffen werden. Beispiel:

```
pair<int, double> x(5, 17.4);
cout << x.first << ", " << x.second << endl; // Ausgabe: 5, 17.4
```

**Aufgabe 12.6.** Tag-Clouds stellen die Wichtigkeit von Begriffen in Texten dar, indem sie diese entsprechend ihrer Häufigkeit unterschiedlich groß darstellen. Häufig eingesetzt werden sie in sozialen Medien und Netzwerken. In den folgenden Aufgaben implementieren Sie eine Tag-Cloud. Machen Sie sich dazu mit dem `map`-Container vertraut. Schreiben Sie eine Funktion `worthaeufigkeit`, der Sie einen Dateinamen übergeben können. Die Funktion soll einen Container vom Typ `map<string,int>` zurückliefern, der alle in der Datei vorkommenden Wörter mit zugehörigen Häufigkeiten enthält. Anleitung: Für jedes eingelesene Wort überprüfen Sie mit der `find`-Methode, ob es sich bereits in Ihrem `map`-Container befindet. Ist das der Fall, so liefert die Funktion einen Iterator – nennen wir ihn `it` – zurück und Sie können die Häufigkeit mittels `it->second += 1` erhöhen. Andernfalls liefert die `find`-Methode den `end`-Iterator zurück und Sie können mittels `insert` ein passendes Werte-Paar einfügen.

**Aufgabe 12.7.** Ihre in Aufgabe 12.6 geschriebene Funktion berechnet nun zwar wie häufig jedes Wort vorkommt, aber Sie kennen die häufigsten Wörter noch nicht. Schreiben Sie eine Funktion `sortiere`, der Sie den in Aufgabe 12.6 berechneten `map<string,int>`-Container übergeben und die einen Container vom Typ `vector<pair<string,int> >` zurückgibt, der die einzelnen Wörter mit zugehörigen Häufigkeiten aufsteigend sortiert nach ihren Häufigkeiten enthält. Anleitung: Verwenden Sie `sort` aus der Standardbibliothek, sowie die Funktion

```
bool less_than(pair<string,int> a, pair<string,int> b) {
    return a.second < b.second;
}
```

als Ordnungsfunktion.

**Aufgabe 12.8.** Unter den häufigsten Wörtern finden sich auch Bindewörter oder Artikel. Schreiben Sie eine Funktion `entferne_bindewoerter`, der Sie zwei Parameter übergeben: den Container vom Typ `vector<pair<string,int> >`, den Sie in Aufgabe 12.7 erzeugt haben und den Namen einer Datei, die eine Liste mit Bindewörtern enthält. Die Funktion soll die Datei wortweise einlesen und alle Einträge aus dem Vektor streichen, deren erster Eintrag gleich einem der eingelesenen Wörter ist. Der veränderte Vektor soll zurückgegeben werden.

*Hinweis:* Alternativ zu einer naiven Implementierung können Sie `remove_if` und eine passende Prädikatsfunktion verwenden.