Programmierkurs
Dr. Steffen Müthing
IWR, Universität Heidelberg

Aufgabenblatt 5
Abgabe 26. November 2018
ipk-exercises:2018-ws-17-ge4d2b98

Aufgabenblatt 5

Allgemeine Hinweise:

- Für die Aufgaben auf diesem Übungsblatt müssen Sie am 26.11. votieren.

Aufgabe 1

Listen von Zahlen

Im folgenden lernen Sie den wichtigsten Container der C++-Standardbibliothek kennen: `std::vector<T>`, eine indizierte Liste mit Einträgen vom Typ `T`. `T` kann hierbei ein (fast) beliebiger Datentyp sein, z.B. `int` oder `double`. Einen `std::vector` können Sie auf verschiedene Weisen anlegen:

```
#include <vector> // vector in Ihrem Programm verfügbar machen

int main(int argc, char** argv)
{
  // Ein leerer vector für ganze Zahlen
  std::vector<int> v1;
  // Ein vector für ganze Zahlen mit 10 Einträgen
  std::vector<int> v2(10);
  // Ein vector mit den Einträgen 3,8,7,5,9,2
  std::vector<int> v3 = {{ 3, 8, 7, 5, 9, 2 }};
}
```

Ein `vector` ist ein *Objekt* und hat sogenannte *member functions*, das sind spezielle Funktionen, die das Objekt verändern. Eine vollständige Referenz finden Sie auf der Website `cppreference.com`[1], die wichtigsten Methoden für diese Aufgabe sind:

```
std::vector<int> v = {{ 3, 8, 7, 5, 9, 2 }};
// Gibt die Anzahl der Einträge zurück
std::cout << v.size() << std::endl; // 6
// Verändert die Länge der Liste
v.resize(42);
```

Um auf einen Eintrag des Vektors zuzugreifen, schreiben Sie den Index des Eintrags in eckigen Klammern hinter den Variablennamen. **Die Nummerierung der Einträge beginnt bei 0, nicht bei 1**. Um einen Eintrag zu verändern, weisen Sie dem Eintrag einfach einen neuen Wert zu:

```
// Zugriff auf einzelne Einträge - Index ist 0-basiert!
std::cout << v[2] << std::endl; // 7
v[0] = v[0] * 2;
std::cout << v[0] << std::endl; // 6
```

Aufgaben:

(a) Legen Sie einen `vector<double>` mit jeder der oben beschriebenen Methoden an und geben Sie jeweils alle Einträge mit einer `for`-Schleife aus. Welchen Wert haben Einträge, für die Sie keinen expliziten Wert angegeben haben?

[1] http://en.cppreference.com/w/cpp/container/vector

(b) Schreiben Sie eine Funktion, die den grössten und den kleinsten Wert in einem Vektor findet und als `std::pair` zurückgibt (erst den kleinsten, dann den größten Wert). Testen Sie die Funktion mit verschiedenen Vektoren.

(c) Schreiben Sie eine Funktion `std::vector<double> reversed(std::vector<double> v)`, die einen Vektor mit Einträgen $x_0, x_1, \ldots, x_{n-1}$ als Parameter nimmt und einen neuen Vektor mit den Einträgen in umgekehrter Reihenfolge $x_{n-1}, x_{n-2}, \ldots, x_0$ zurückgibt. Testen Sie die Funktion mit verschiedenen Vektoren, insbesondere auch mit einem leerem.

(d) Schreiben Sie eine Funktion, die alle Einträge in einem `std::vector<double>` auf ganze Zahlen rundet und diese dann wieder im **selben** Vektor speichert. Zum Runden von Zahlen verwenden Sie folgendes:

```
#include <cmath>

int main()
{
  double x = 2.71;
  double x_rounded = std::round(x);
}
```

Testen Sie die Funktion mit verschiedenen Vektoren.

(e) Schreiben Sie eine Funktion, das die Reihenfolge der Einträge in einem Vektor umkehrt, aber das Ergebnis nun im **selben** Vektor speichert. Verwenden Sie zum Vertauschen einzelner Einträge die Funktion `std::swap(a,b)`. Lesen Sie auf cppreference.com[2] nach, was diese Funktion macht und welche `#include`-Anweisung Sie benötigen. Testen Sie die Funktion mit leeren Vektoren sowie mit welchen, die eine gerade bzw. eine ungerade Grösse haben.

Aufgabe 2

Zahlen einlesen

Anstatt sich auf die eingebauten Funktionen von C++ zu verlassen, schreiben Sie im folgenden eine Reihe von Funktionen, die einen String (z.B. `"1346"`) in eine Zahl umwandeln sowie ein Hauptprogramm, das diese Funktionen testet.

Für den Anfang können Sie davon ausgehen, dass Sie nur gültige Zeichenfolgen als Eingabe bekommen.

(a) Schreiben Sie eine Funktion `int parse_int(std::string number)`, die den String `number` in eine Zahl umwandelt und diese zurückgibt. Sie können davon ausgehen, dass die Eingabe kein Vorzeichen enthält.

Hinweise:

- Ein `std::string`[3] ist auch ein Container! Sie können ihn sich konzeptuell als etwas ähnliches wie ein `std::vector<char>` vorstellen. Ein einzelnes Zeichen können Sie in einer Variablen vom Typ `char` speichern. Das folgende Beispiel zeigt Ihnen, wie Sie an die Information in `std::string` herankommen:

```
std::string s = "47218";    // Zuweisung
std::getline(std::cin,s);   // Zeile einlesen und in s speichern, nicht mit >>
int size = s.size();        // Gibt die Länge des Strings zurück
char c = s[0];              // erstes Zeichen (0-basierter Index)
char d = s[size-1];         // letztes Zeichen
```

- Eine Variable vom Typ `char` enthält eine Zahl, die das zugehörige Zeichen repräsentiert. Hierfür gibt es verschiedene Standards, welche Zahl für welches Zeichen steht, für unsere Zwecke

[2] http://en.cppreference.com/w/cpp/algorithm/swap
[3] http://en.cppreference.com/w/cpp/string/basic_string

reicht ASCII[4]. Dabei ist z.B. der Wert des Zeichens `'3'` nicht 3, sondern 51. Sie müssen den Wert des Zeichens `'0'` (= 48) abziehen, um zum korrekten Ziffernwert zu kommen:

```
char three = '3';
std::cout << three << std::endl; // Ausgabe: 51
three = three - '0';             // alternativ: three - 48
std::cout << three << std::endl; // Ausgabe: 3
```

- Sie können für diesen Aufgabenteil davon ausgehen, dass der gesamte String nichts ausser der einzulesenden Zahl enthält.

(b) Erweitern Sie Ihre Funktion so, dass sie ein optionales `'+'` oder `'-'` am Anfang des Strings korrekt einliest und auf die Zahl anwendet.

(c) Erweitern Sie Ihre Funktion so, dass sie eventuelle Leerzeichen am Anfang des Strings ignoriert und die Zahl so lange weiter einliest, bis ein anderes Zeichen als eine Ziffer kommt. Beispiel: Der String `"  -7628.8text"` soll in die Zahl -7628 umgewandelt werden (`"."` ist kein gültiges Zeichen innerhalb einer ganzen Zahl, also stoppt das Einlesen dort).

Hinweis:

- Um lange Ketten von `if`-Statements zu vermeiden, kann es hier hilfreich sein, stattdessen ein `switch`-Statement[5] zu verwenden, dass in einem Statement mehrere Alternativen abhandeln kann:

```
char c = ...;
switch (c) {
  case ' ':
    // handle blank spaces
    break; // leave switch statement
  case '+':
  case '-':
    // handle plus or minus
    break; // leave switch statement
  case '0':
  case '1':
  ...
  case '9':
    // handle digits
    break; // leave switch statement
  default:
    // handle any other (invalid) symbol
}
```

- Sie müssen wahrscheinlich mit einigen `bool`-Variablen nachverfolgen, ob gerade noch ein Leerzeichen oder ein Vorzeichen kommen darf.

(d) Erweitern Sie Ihre Funktion so, dass sie statt einem `int` ein `std::pair<int,int>` zurückgibt, wobei der erste Eintrag die eingelesene Zahl sein soll und der zweite der Index des ersten Zeichens, das nicht mehr eingelesen wurde.

(e) **Fortgeschrittene / Bonus:** Erweitern Sie Ihre Funktion so, dass sie im Fehlerfall (also, wenn der String z.B. mit einem Buchstaben anfängt) eine Exception vom Typ `std::invalid_argument`[6] wirft.

[4] American Standard Code for Information Interchange, https://en.wikipedia.org/wiki/ASCII
[5] https://en.cppreference.com/w/cpp/language/switch
[6] http://en.cppreference.com/w/cpp/error/invalid_argument