Übung 8

Aufgabe 1: 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`[1] 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:

```
1  std::string s = "47218";    // Zuweisung
2  std::getline(std::cin,s);   // Zeile einlesen und in s speichern, nicht mit >>
3  int size = s.size();        // Gibt die Länge des Strings zurück
4  char c = s[0];              // erstes Zeichen (0-basierter Index)
5  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 reicht ASCII[2]. 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:

```
1  char three = '3';
2  std::cout << three << std::endl; // Ausgabe: 51
3  three = three - '0';             // alternativ: three - 48
4  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:

[1] http://en.cppreference.com/w/cpp/string/basic_string
[2] American Standard Code for Information Interchange, https://en.wikipedia.org/wiki/ASCII

- Um lange Ketten von `if`-Statements zu vermeiden, kann es hier hilfreich sein, stattdessen ein `switch`-Statement[3] 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) 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`[4] wirft.

Aufgabe 2: Primzahlen

Eine natürliche Zahl $p \neq 1$ heißt *prim*, wenn Sie nur durch sich selbst und 1 teilbar ist.

(a) Schreiben Sie eine Funktion `bool isPrime(int number)`, die für eine gegebene Zahl testet, ob diese eine Primzahl ist. In diesem Fall soll sie `true` zurück geben, ansonsten `false`. Beachten Sie dabei insbesondere, dass nicht-positive Zahlen nach Definition nie prim sind.

(b) Schreiben Sie außerdem eine Funktion `void printPrimes(int upto)`, die alle Primzahlen bis einschließlich `upto` auf dem Bildschirm ausgibt. Falls `upto` zu klein ist, um überhaupt eine Zahl auszugeben, soll stattdessen eine entsprechende Fehlermeldung erscheinen. Greifen Sie hierbei auf die von Ihnen implementierte Funktion `isPrime()` von oben zurück.

Testen Sie die beiden Funktionen in einem Programm, das für verschiedene Werte von `upto`, davon mindestens einer negativ, einer gleich Null, und einer größer als 50, die Funktion `printPrimes` aufruft.

[3] https://en.cppreference.com/w/cpp/language/switch
[4] http://en.cppreference.com/w/cpp/error/invalid_argument