C/C++ Programmierung
Übungsblatt 1

Florian Oetke

Entwicklungsumgebung
Für dieses Übungsblatt müssen Sie selbst Code schreiben. Hierzu können Sie die Webseite godbolt.org, die in der Vorlesung vorgestellt wurde, oder eine andere Umgebung ihrer Wahl verwenden.

Für die Verwendung von godbolt.org können Sie auf einem der Beispiele aus der Vorlesung aufbauen oder diese Vorlage (https://godbolt.org/z/3G6a7W1E1) verwenden.

Diejenigen die godbolt.org nicht verwenden wollen, finden auf Panopto entsprechende Videos zur Verwendung von Microsoft Visual Studio oder CLion.

1 Zahlendreieck

In dieser Aufgabe soll eine Funktion `void print_triangle(int size)` implementiert werden, die ein Dreieck aus Zahlen mit der entsprechenden Höhe und Breite ausgibt. Für `size` sollen Werte zwischen 0 und 9 (inklusive) akzeptiert werden (verwenden Sie 0 bzw. 9 bei Eingaben außerhalb dieses Bereichs).

Bei `print_triangle(9)` sollte die Ausgabe wie folgt aussehen:

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
```

Und bei `print_triangle(3)` sollte die Ausgabe so aussehen:

```
1
12
123
```

Als Basis für die Implementierung können Sie folgenden Programmcode verwenden:

```
1 #include <iostream>
2
3 void print_triangle(int size) {
4     std::cout << "TODO " << size << "\n";
5 }
6
7 int main() {
8     print_triangle(9);
9
10     print_triangle(3);
11 }
```

2 Rechtsbündiges Zahlendreieck

In dieser Aufgabe soll die Funktion `print_triangle` aus der vorherigen Aufgabe um einen Parameter `right_align` erweitert werden, der dafür sorgt, dass das Dreieck rechtsbündig ausgegeben wird.

Die bisherigen Aufrufe der Funktion (ohne den neuen Parameter) sollen auch weiterhin unverändert funktionieren.

Bei `print_triangle(9, true)` sollte die Ausgabe so aussehen:

```
        1
       12
      123
     1234
    12345
   123456
  1234567
 12345678
123456789
```

3 FizzBuzz

In dieser Aufgabe soll das einfache Kinder-Zahlenspiel FizzBuzz implementiert werden, bei dem von 1 an gezählt wird, wobei alle Vielfachen von 3 durch Fizz, alle Vielfachen von 5 durch Buzz und alle Zahlen, die sowohl durch 3 als auch durch 5 teilbar sind, durch FizzBuzz ersetzt werden.

Schreiben Sie hierzu eine Funktion `void fizz_buzz(int number)` welche eine Zahl entgegennimmt und den Regeln oben folgend, mit `std::cout` entweder die Zahl, Fizz, Buzz oder FizzBuzz ausgibt.

Rufen Sie ihre Funktion anschließend in der main-Funktion für alle Zahlen von 0 bis 100 auf.

Die Ausgabe sollte mit folgenden Ausgaben anfangen:

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
```

4 Ein einfacher Taschenrechner

In dieser Aufgabe soll ein sehr einfacher Taschenrechner programmiert werden. Hierzu müssen, wie in der Vorlesung gezeigt, Werte mit `std::cin` eingelesen und das Ergebnis anschließend mit `std::cout` ausgegeben werden.

Die Eingabe sollte dabei aus zwei Fließkommazahlen und einer Operation (+ - * / als `char`) bestehen.

Bei einer Eingabe von `2+3.141` sollte die Ausgabe so aussehen:

```
= 5.141
```

5 Fehler finden

Das folgende Programm enthält zwei Fehler: Einen Compiler-Fehler, der bereits die Übersetzung verhindert, und einen Logik-Fehler, durch den das Verhalten nicht dem erwarteten Ergebnis entspricht.

1. Was ist der Compiler-Fehler?
2. Was müsste am Code geändert werden, um ihn zu beheben?
3. Was ist der Logik-Fehler?
4. Wieso kommt es zu einem falschen Ergebnis?
5. Was wären zwei Möglichkeiten, um den Fehler zu beheben?

```cpp
#include <iostream>

namespace fbi_cpp {
    char lower(char c) {
        return c | 0b00100000;
    }

    bool lower(int i) {
        return i<80 ? true : false;
    }
}

int main() {
    // Erwartete Ausgabe:
    // lower('G')   : g
    // lower('G'+2) : i
    // lower(100)   : 0

    std::cout << "lower('G')   : " << lower('G')   << "\n";
    std::cout << "lower('G'+2) : " << lower('G'+2) << "\n";
    std::cout << "lower(100)   : " << lower(100)   << "\n";
}
```