IPI-Zettel 5

Jan Maintok, Raphael Senghaas, Kianusch Vahid Yousefnia
Tutor: Enes Witwit

Aufgabe 2

```cpp
#include <string>
#include <vector>
#include <iostream>
#include <cassert>

using namespace std;

/*Funktion, die ein Kartendeck mit "size" Karten erstellt.
  durch die Einführung der Variablen size kann dieses Programm
  einfacher auf andere Situationen erweitert werden.
*/
vector<int> init_deck(int size){
  vector<int> deck;
  for (int i=0; i<size; i++){
    deck.push_back(i);
  }
  return deck;
}

/*Es wurde hier bewusst eine void Funktion und nicht
  eine vector<int> Funktion erstellt, und stattdessen mit
  einer Referenz gearbeitet um Laufzeit durch den Kopiervorgang
  zu sparen. Es scheint außerdem sinnvoll ein gemischtes
  Kartendeck direkt zu überschreiben anstatt die ungemischte
  Version noch weiter gespeichert zu halten.
*/
void shuffle(vector<int> & cards, bool out){
  vector<int> shuffled_deck;
  for (int i = 0; i < cards.size()/2; ++i) {
    if (out) {
      shuffled_deck.push_back(cards[i]);
      shuffled_deck.push_back(cards[cards.size()/2+i]);
    }
    else {
      shuffled_deck.push_back(cards[cards.size()/2+i]);
      shuffled_deck.push_back(cards[i]);
    }
  }
  cards=shuffled_deck;
}

```

```
/*Funktion, die prüft, ob sich ein Kartendeck in der
  geordneten Anordnung befindet.
*/
bool check_deck(vector<int> cards){
  vector<int> stddeck = init_deck(cards.size());
  for (int i=0; i<cards.size(); i++){
    if (cards[i] == stddeck[i])
      continue;
    else
      return false;
    }
  return true;
}

int main(){
  //Erstellung des Kartendecks
  vector<int> deck = init_deck(52);
  assert(check_deck(deck));

  shuffle(deck, true);
  //Laufvariable, die zählt wie häufig gemischt wurde
  int n=1;
  /*Schleife, die so lange läuft, bis sich das Kartendeck durch
  Out-Shuffles wieder im Ausgangszustand befindet.*/
  while (!check_deck(deck)) {
    shuffle(deck, true);
    n++;
  }
  cout << "Needed shuffles for Out-Shuffle: " << n << endl;
  n=1;
  shuffle(deck, false);
  /*Schleife, die so lange läuft, bis sich das Kartendeck durch
  In-Shuffles wieder im Ausgangszustand befindet.*/
  while (!check_deck(deck)) {
    shuffle(deck, false);
    n++;
  }
  cout << "Needed shuffles for In-Shuffle: " << n << endl;
}
```

Aufgabe 3

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include "text.hpp"

using namespace std;

//Declarations
vector<string> split_words(string s);
```

```
11 string mix(string s);
12 string split_and_mix(string s);
13
14
15 int main(){
16   cout << str1 + "\n\n"
17     << split_and_mix(str1) + "\n\n"
18     << str2 + "\n\n"
19     << split_and_mix(str2) + "\n\n"
20     << str3 + "\n\n"
21     << split_and_mix(str3) + "\n\n"
22     << str4 + "\n\n"
23     << split_and_mix(str4) + "\n\n"
24     << str5 + "\n\n"
25     << split_and_mix(str5) + "\n\n";
26   return 0;
27 }
28
29 //Functions of my own
30 //a)
31 vector<string> split_words(string s){
32   // develop current word
33   string current_word = "";
34   vector<string> output;
35   int i = 0;
36   while (i < s.length()){
37     if (s[i] != ' '){
38       current_word += s[i];
39     }
40     else {
41       output.push_back(current_word);
42       current_word = "";
43     }
44
45     i++;
46   }
47   output.push_back(current_word); // Add last word to output
48
49   return output;
50 }
51
52 //b)
53 string mix(string s){
54   //find indices of first and last letter
55   bool first_letter_found = false;
56   int index_first;
57   int index_last;
58   for (int i = 0; i<s.length(); i++){
59     if (!isalpha(s[i])){      //ignore non-letters
60       continue;
61     }
62     if (!first_letter_found){
```

```
63        index_first = i;
64        first_letter_found = true;
65      }
66      index_last = i;   //is constantly overriden -> desired result is last
        override
67    }
68
69    // shuffle between index_first und index_last
70    random_shuffle(s.begin()+index_first+1,s.begin()+index_last-1);
71    return s;
72  }
73
74  //c)
75  string split_and_mix(string s){
76    vector<string> words = split_words(s);
77    string output = "";
78    for (int i = 0; i<words.size(); i++){
79      output += mix(words[i]) + " ";    //concatinate shuffled words with space
80    }
81    return output;
82  }
```