SCHETT MATTHIAS

SEN-ÜBUNG 10

Inhaltsverzeichnis

Aufgabe 1

Lösungsidee

Es werden drei Funktoren erstellt, CharacterComparator[1], LetterFrequency[2] und CompareByLetterFrequency[3].

Der Quellcode ist im Anhang unter Aufgabe 1 zu finden.

[1] Vergleicht zwei einzelene Character auf gleicheit mittels toLower() um Klein/Großschreibung zu ignorieren

[2] Zählt mittels count_if und CharacterComparator die Anzahl an zu suchenden Charactern

[3] Vergleicht und sortiert die Ergebnisse von LetterFrequency nach der Häufigkeit aufsteigend

Testfälle

Listing 1: Testtreiber

```
1 Please input the character you want to search for: Strings:
      one two three four five
2 Character: e
3 Frequency: 1 0 2 0 1
4 Frequency sorted: two four one five three
```

Aufgabe 2

Lösungsidee

Der Funktor soll in Verbindung mit std::generate funktionieren, daher wird der Funktor ohne die Übergabe von Argumenten aufgerufen. Der Konstruktor bekommt, allerdings einen Startwert und eine Schrittweite. Der Funktor gibt anschließend immer den Startwert += Schrittweite aus.

Der Quellcode ist im Anhang unter Aufgabe 2 zu finden.

Testfälle

Listing 2: Testtreiber

```
1 1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
66 66.2 66.4 66.6 66.8
```

Aufgabe 3

Lösungsidee

In einem Vector mit strings soll die maximale, mittlere und kürzeste String Länge mittels einen std::for_each Aufrufes und eines Funktors realisiert werden. Dazu wird der Funktor StringLength implementiert. Dieser hat folgende Member

- maxLength
- minLength
- averageLenght
- numOfStrings

std::for_each ruft den Funktor für jedes Element innerhalb der Iterator Range einzeln auf, daher wird nur ein einzelener String in den Funktor übergeben, dieser wird analysiert. Abschließend liefert std::for_each eine Kopie des Funktors zurürck, auf dieser Kopie wird dann die Print() Funktion aufgerufen, um die Statistik anzuzeigen. Der Quellcode ist im Anhang unter **??** zu finden.

Testfälle

Listing 3: Testausgabe

```
1 The strings to analyse
2 short
3 LongString
4 AverageL
5 Max Length: 10
6 Min Length: 5
7 Average Length: 7
```

Anhang A
Aufgabe 1

Listing A.1: Implementierung

```
1  ////////////////////////////////////
2  // Workfile    : LetterFrequency.h
3  // Author      : Matthias Schett
4  // Date        : 16-06-2013
5  // Description : analyses strings
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include <functional>
11 #include <algorithm>
12 #include <locale>
13
14 class CharacterComparator: public std::binary_function<char,char,bool>{
15 public:
16     bool operator() (char const& a, char const& b) const{
17         return ( tolower(a) == tolower(b) );
18     }
19 };
20
21
22 class LetterFrequency : public std::unary_function<std::string, size_t> {
23
24 private:
25     char mCharToCompare;
26
27 public:
28     LetterFrequency(char const& c='A'):
29         mCharToCompare(c){
30     }
31
32     size_t operator() (std::string const& str){
33         return count_if(str.begin() , str.end(), bind1st(CharacterComparator(),
               mCharToCompare));
34     }
```

```
35
36 };
37
38
39 class CompareByLetterFrequency : public std::binary_function<std::string, std::string, bool
       >{
40
41 private:
42     std::function<size_t(std::string)> mFunctor;
43
44 public:
45     CompareByLetterFrequency(std::function<size_t(std::string)> const& unaryOP =
           LetterFrequency()):
46         mFunctor(unaryOP){
47     }
48
49     bool operator() (std::string const& strLeft, std::string const& strRight){
50         return (mFunctor(strLeft) < mFunctor(strRight));
51     }
52 };
```

Listing A.2: Testtreiber

```
1  ////////////////////////////////////
2  // Workfile    : Main.cpp
3  // Author      : Matthias Schett
4  // Date        : 16-06-2013
5  // Description : analyses strings
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include <iostream>
11 #include <functional>
12 #include <iterator>
13 #include <vector>
14 #include <string>
15 #include "LetterFrequency.h"
16
17 using namespace std;
18
19 int main(){
20     //
21     size_t const n = 5;
22     string strarr[n] = {"one", "two", "three", "four", "five"};
23
24     char input;
25
26     vector<string> test;
27     copy(strarr, strarr+n, back_inserter(test));
```

```
28
29      cout << "Please input the character you want to search for: ";
30      cin >> input;
31      cin.ignore();
32
33      cout << "Strings: ";
34      copy(test.begin(), test.end(), ostream_iterator<string>(cout, " "));
35      cout << endl;
36
37      cout << "Character: " << input << endl;
38
39      cout << "Frequency: ";
40      transform(test.begin(), test.end(), ostream_iterator<size_t>(cout, " "),
41                LetterFrequency(input));
42      cout << endl;
43
44      cout << "Frequency sorted: ";
45      sort(test.begin(), test.end(), CompareByLetterFrequency(LetterFrequency(input)));
46      copy(test.begin(), test.end(), ostream_iterator<string>(cout, " "));
47      cout << endl;
48
49      cin.get();
50      return 0;
51  }
```

Anhang B
Aufgabe 2

Listing B.1: Testtreiber

```
1  ////////////////////////////////////
2  // Workfile    : Main.cpp
3  // Author      : Matthias Schett
4  // Date        : 16-06-2013
5  // Description : generates a sequence
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10
11 #include "SequenceGenerator.h"
12 #include <algorithm>
13 #include <iostream>
14 #include <iterator>
15
16 using namespace std;
17
18 int main(){
19
20     // Numbers from 1 to 10
21     generate_n(ostream_iterator<int>(cout, " "), 10, SequenceGenerator<int>(1));
22     cout << endl;
23     // Numbers from 10 to 1 (descending order)
24     generate_n(ostream_iterator<int>(cout, " "), 10, SequenceGenerator<int>(10, -1));
25     cout << endl;
26     // Numbers from 0 to 5 (increment 0.5)
27     generate_n(ostream_iterator<double>(cout, " "), 11, SequenceGenerator<double>(0, 0.5));
28     cout << endl;
29     // Letters from A to Z
30     generate_n(ostream_iterator<char>(cout, ""), 26, SequenceGenerator<char>('A'));
31     cout << endl;
32
33     generate_n(ostream_iterator<char>(cout, ""), 26, SequenceGenerator<char>(97));
34     cout << endl;
35
```

```
36      generate_n(ostream_iterator<double>(cout, " "), 5, SequenceGenerator<double>('B', 0.2))
            ;
37      cout << endl;
38
39      cin.get();
40      return 0;
41  }
```

Listing B.2: Testtreiber

```
1   ////////////////////////////////////
2   // Workfile    : SequenceGenerator.h
3   // Author      : Matthias Schett
4   // Date        : 16-06-2013
5   // Description : generates a sequence
6   // Remarks     : -
7   // Revision    : 0
8   ////////////////////////////////////
9
10  template <typename TValue>
11  class SequenceGenerator {
12
13  private:
14      TValue startValue;
15      TValue steps;
16      bool first;
17
18  public:
19
20      SequenceGenerator(TValue startV, TValue step = 1) : startValue(startV), steps(step),
            first(true) {
21      }
22
23      TValue operator() ();
24
25  };
26
27  template <typename TValue>
28  TValue SequenceGenerator<TValue>::operator() (){
29      if(first){
30          first = false;
31          return startValue;
32      } else {
33          return startValue += steps;
34      }
35  }
```

Anhang C
Aufgabe 2

Listing C.1: Testtreiber

```
1  ////////////////////////////////////
2  // Workfile    : Main.cpp
3  // Author      : Matthias Schett
4  // Date        : 16-06-2013
5  // Description : generates a sequence
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include <iostream>
11 #include <algorithm>
12 #include <vector>
13 #include <string>
14 #include "StringLength.h"
15
16 using namespace std;
17
18 int main(){
19
20     vector<string> V1;
21     V1.push_back("short");
22     V1.push_back("LongString");
23     V1.push_back("AverageL");
24
25     cout << "The strings to analyse" << endl;
26     cout << V1[0] << endl;
27     cout << V1[1] << endl;
28     cout << V1[2] << endl;
29
30     StringLength strLen = for_each(V1.begin(), V1.end(), StringLength() );
31
32     strLen.print();
33
34     cin.get();
35     return 0;
```

```
}
```

Listing C.2: Testtreiber

```
//////////////////////////////////////
// Workfile    : Main.cpp
// Author      : Matthias Schett
// Date        : 16-06-2013
// Description : generates a sequence
// Remarks     : -
// Revision    : 0
//////////////////////////////////////

#include <iostream>
#include <string>
#include <functional>

class StringLength : public std::unary_function<std::string, void> {

private:

    size_t maxLength;
    size_t minLength;
    size_t averageLenght;
    size_t numOfStrings;

public:

    StringLength() : maxLength(0), minLength(0), averageLenght(0), numOfStrings(0) {}

    void operator() (std::string const &a) {

        numOfStrings++;

        averageLenght += a.length();

        if(numOfStrings == 1){
            minLength = a.length();
        }

        if(a.length() > maxLength){
            maxLength = a.length();
        } else if(a.length() < minLength){
            minLength = a.length();
        }


    }

    void print(){
```

```
47          std::cout << "Max Length: " << maxLength << std::endl;
48          std::cout << "Min Length: " << minLength << std::endl;
49          std::cout << "Average Length: " << (averageLenght / numOfStrings) << std::endl;
50      }
51
52  };
```