Aufgabe 1

siehe .odg Datei.

Aufgabe 2

(a)

Listing 1: primfaktor.cc

```
1  #include <iostream>
2  #include <cmath>
3
4  bool teilt(int a, int b)
5  //ermittelt, ob b a teilt
6  {
7      if (b%a == 0)
8      {
9          return true;
10     }
11     return false;
12 }
13
14 int main()
15 {
16     //zu untersuchende Zahl
17     const int n = 361;
18     //feld[i] gibt an, durch welche potenz von i n teilbar ist
19     int feld[n];
20     //Initialisierung
21     for (int i = 0; i<n; ++i)
22     {
23         feld[i] = 0;
24     }
25     //immer, wenn ein weiterer Teiler von n gefunden wird, wird restzahl durch diesen
26     //Daher ist restzahl immer gleich n geteilt durch das Produkt aller bisher ermitt
27     int restzahl = n;
28     for (int i = 2; i<=sqrt((double) n); ++i)
29     {
30         //findet heraus, wie oft i restzahl noch teilt
31         while (teilt(i,restzahl))
32         {
33             feld[i] = feld[i] + 1;
34             restzahl = restzahl/i;
35         }
36     }
37     //wenn in der Primfaktorzerlegung noch eine Primzahl groesser als sqrt(n) enthalte
38     //so wird sie in diesem Schritt noch ausgegeben
```

```
39      if (restzahl != 1)
40      {
41          feld[restzahl] = 1;
42      }
43      //Ausgabe
44      for (int i = 0; i<n; ++i)
45      {
46          if (feld[i] > 0)
47          {
48              std::cout << i << "^" << feld[i] << " ";
49          }
50      }
51      std::cout << std::endl;
52      return 0;
53  }
```

(b) Zu Beginn gibt es einen konstanten Initialisierungsaufwand. Ist n prim, so wird die Schleife sqrt(n) mal durchlaufen. Die Schleifenbedingung der while-Schleife ist allerdings nie erfüllt. Pro Schleifendurchlauf wird also genau einmal die teilt-Funktion aufgerufen. Die Komplexität der teilt-Funktion sei $C(n)$. Dann ist die Gesamtkomplexität des Programm $\Omega(C(n) \cdot \sqrt{n})$

Aufgabe 3

Listing 2: taschenrechner.cc

```
1  #include "fcpp.hh"
2  #include <string.h>
3  #include <iostream>
4  using namespace std;
5
6  // fuer strlen, Laenge eines C-Strings
7
8  // Definieren Sie hier Ihren Stack und legen Sie eine Instanz als globale
9  // Variable an
10
11 const int stacklength = 1000;
12
13 struct Stack
14 {
15   int counter;
16   int array[stacklength];
17 };
18
19 Stack stack;
20
```

```
21 // Danach koennen Sie die Funktionen push() und pop() implementieren, die auf dieser
22 // globalen Variable operieren
23
24 void push(int element)
25 {
26   stack.counter += 1;
27   stack.array[stack.counter - 1] = element;
28 }
29
30 int pop()
31 {
32   stack.counter -= 1;
33   return stack.array[stack.counter];
34 }
35
36 bool is_operator(char zeichen)
37 //gibt true aus, wenn zeichen ein operator ist
38 {
39   if ( '+' == zeichen || '-' == zeichen || '*' == zeichen || '/' == zeichen )
40   {
41     return true;
42   }
43   return false;
44 }
45
46
47 bool is_ziffer(char zeichen)
48 //gibt true aus, wenn zeichen eine Ziffer ist
49 {
50   if (zeichen == '0' || zeichen == '1' || zeichen == '2' ||zeichen == '3' ||zeichen == '
51   {
52     return true;
53   }
54   return false;
55 }
56
57 int ziffer(char zeichen)
58 //gibt die Ziffer als int aus, wenn der Buchstabe eine Ziffer ist.
59 //ansonsten fehlermeldung
60 {
61   if (zeichen == '0')
62   {
63     return 0;
64   }
65   if (zeichen == '1')
66   {
```

```
67      return 1;
68    }
69    if (zeichen == '2')
70    {
71      return 2;
72    }
73    if (zeichen == '3')
74    {
75      return 3;
76    }
77    if (zeichen == '4')
78    {
79      return 4;
80    }
81    if (zeichen == '5')
82    {
83      return 5;
84    }
85    if (zeichen == '6')
86    {
87      return 6;
88    }
89    if (zeichen == '7')
90    {
91      return 7;
92    }
93    if (zeichen == '8')
94    {
95      return 8;
96    }
97    if (zeichen == '9')
98    {
99      return 9;
100   }
101   print("Das Argument ist keine Ziffer!");
102 }
103
104 int main(int argc, char* argv[])
105 {
106   // Setzen Sie hier auf einen leeren Stack
107   stack.counter = 0;
108
109   // fange kein Kommandozeilenargument ab
110   if(argc < 2)
111   {
112     print("Eingabe fuer den Taschenrechner erwartet!");
```

```
113     return 1;
114   }
115
116   // arg enthaelt die als Eingabe von der Kommandozeile uebergebene Zeichenfolge
117   char* arg = argv[1];
118
119   // Schleife, die die Zeichen der Eingabe nacheinander ablaeuft
120   // strlen gibt die Anzahl der Zeichen in der Zeichenkette
121   int element = 0;
122   bool lziffer = false;
123   for (int i = 0; i <= strlen(arg); i = i+1)
124   {
125     char zeichen = arg[i]; // aktuelles Zeichen
126     // Beachten Sie: der Inhalt der Variable zeichen ist der ASCII-Code
127     // des entsprechenden Zeichens. Dieser stimmt im Falle der Zeichen '0'..'9'
128     // nicht mit der entsprechenden Ziffer ueberein.
129
130     // Fuegen Sie hier Code ein, der das Zeichen verarbeitet, also Ziffern
131     // zu Zahlen zusammenfuegt, Operatoren anwendet und andere Zeichen
132     // ignoriert
133
134     if (is_ziffer(zeichen) && lziffer == false)
135     //wenn eine neue zahl beginnt, dann setzen wir lziffer auf true
136     //und element auf den Wert der aktuellen Ziffer
137     {
138       lziffer = true;
139       element = ziffer(zeichen);
140     }
141
142     else if (is_ziffer(zeichen) && lziffer == true)
143     //wenn eine Zahl bereits begonnen hat und nun eine weitere ziffer hinzukommt,
144     //so multiplizieren wir die bisherige zahl mit 10 und addieren die neue Ziffer
145     {
146       element = 10*element + ziffer(zeichen);
147     }
148     else //Zeichen ist keine Ziffer
149     {
150       if (lziffer == true)
151       //wenn eine zahl bis hierher ging,
152       //so pushen wir element in den stack
153       {
154         push(element);
155       }
156
157       lziffer = false; //da dieses Zeichen keine Ziffer ist, setzen wir lziffer auf false
158
```

```
159         if (is_operator(zeichen))
160         //handelt es sich bei dem Zeichen um einen Operator, so holen wir die obersten 2 z
161         {
162           int ziffer2 = pop();
163           int ziffer1 = pop();
164           int result;
165           //je nach Operation verknuepfen wir nun diese zahlen
166           if (zeichen == '+')
167           {
168             result = ziffer1 + ziffer2;
169           }
170           else if (zeichen == '-')
171           {
172             result = ziffer1 - ziffer2;
173           }
174           else if (zeichen == '*')
175           {
176             result = ziffer1 * ziffer2;
177           }
178           else if (zeichen == '/')
179           {
180             result = ziffer1 / ziffer2;
181           }
182           //Das Resultat pushen wir in den stack
183           push(result);
184         }
185       }
186     }
187     //bei korrekter Eingabe ist nach Durchlauf der gesamten Schleife der counter 1
188     if (stack.counter == 1)
189     {
190       print("Ergebnis:");
191       print(stack.array[0]);
192     }
193     else
194     {
195       print("Fehlerhafte Eingabe");
196     }
197
198 }
```