Übungsblatt 4

Ralph Gauges Ursula Rost Katja Wegner

14.11.2007

Aufgabe 1:

Schreiben Sie die Funktion für die Berechnung der Fakultät einer Zahl so um, dass das Ergebnis nicht wie bisher iterativ, sondern rekursiv berechnet wird. Überprüfen Sie, ob die Funktion das Richtige macht.
Wie die entsprechende Header Datei aussehen müsste, sollte klar sein, weshalb sie hier nicht aufgeführt ist.

```
1 #include "Aufgabe1.h"
2
3 int fakultaet(unsigned int n)
4 {
5     if(n<=2)
6     {
7         return n;
8     }
9     else
10    {
11        return n*fakultaet(n-1);
12    }
13 }
```

Die Funktion nimmt als Argument einen **unsigned int** Wert ist dieser kleiner oder gleich 2 wird der Wert selbst zurückgegeben, ansonsten wird der Wert, multipliziert mit einem neuen Aufruf der Funktion, zurückgegeben, wobei das Argument für den rekursiven Aufruf der um eins verkleinerte Wert ist.

Aufgabe 2:

Erweitern Sie die Klasse Bruch um zwei Methoden. Eine Methode soll das kleinste gemeinsame Vielfache (KGV) und die andere den größten gemeinsamen Teiler (GGT) zweier ganzer Zahlen ermitteln. Benutzen Sie diese Methoden, um die Ergebnisse der Additions-, Subtraktions-, Divisions- und Multiplikationsmethoden als gekürzte Brüche zurückzuliefern. Testen Sie die Methoden mit einigen Brüchen.
Wir gehen hier nur auf die Unterschiede zur ursprünglichen Bruch Klasse ein.

```
1  unsigned int Bruch::calculateGGT(unsigned int n1,unsigned int n2)
2  {
3      unsigned int ggt=1;
4      unsigned int teiler=2;
5      unsigned int end;
6      if(n1 > n2)
7      {
8          end=n2;
9      }
10     else
11     {
12         end=n1;
13     }
14     bool found=false;
15     for( ;teiler <= end && found==false ; ++teiler)
16     {
17         if(n1%teiler==0 && n2%teiler==0)
18         {
19             ggt*=teiler;
20             ggt*=calculateGGT(n1/teiler,n2/teiler);
21             found=true;
22         }
23     }
24     return ggt;
25 }
26
27 unsigned int Bruch::calculateKGV(unsigned int n1,unsigned int n2)
28 {
29     unsigned int ggt=calculateGGT(n1,n2);
30     unsigned int kgv=ggt*(n1/ggt)*(n2/ggt);
31     return kgv;
32 }
33
```

```
34 void Bruch::kuerzen()
35 {
36     unsigned int ggt=calculateGGT(zaehler,nenner);
37     zaehler/=ggt;
38     nenner/=ggt;
39 }
```

Die Klasse verfügt über drei neue Methoden. Die Methode *calculateGGT* berechnet den größten gemeinsamen Teiler der zwei Werte, die als Argumente übergeben werden. Die Methode beginnt beim Wert zwei und prüft im ungünstigsten Fall alle Zahlen zwischen zwei und dem Wert der kleineren der beiden Argumente, ob es sich um einen Teiler für beide Werte handelt (Rest der Ganzzahldivision ist 0). Wurde ein Teiler gefunden, ruft sich die Methode selbst erneut auf (Rekursion). Als neue Argumente dienen diesmal die ursprünglichen Werte geteilt durch den gefundenen Teiler. Der größte gemeinsame Teiler ergibt sich dabei als das Produkt aller gefundenen Teiler. Wurde gar kein Teiler gefunden gibt die Methode den Wert eins zurück.
Die zweite Methode namens *calculateKGV* benutzt die Methode *calculateGGT* um das kleinste gemeinsame Vielfache (KGV) zu berechnen. Das KGV berechnet sich aus dem größten gemeinsame Teiler der beiden, als Argumente übergebenen, Werte, multipliziert mit den beiden Faktoren, die übrig bleiben, wenn man die zwei Werte durch den GGT teilt.
Die dritte Methode *kuerzen* kürzt den Bruch, auf den die Methode aufgerufen wird, indem sie den größten gemeinsamen Teiler von Zähler und Nenner berechnet und anschließend beide Werte durch diesen teilt.
Mit Hilfe dieser Methoden ist es nun leicht möglich die Additions-, Subtraktions-, Divisions- und Multiplikations-Methoden so umzuschreiben, dass diese gekürzte Brüche zurückliefern. Die einfachste Möglichkeit ist es, die Methode *kuerzen* auf das Ergebnis jeder der Methoden aufzurufen, bevor die Methode es mit return zurückliefert. Dies ist hier exemplarisch am Beispiel der Additions-Methode gezeigt:

```
1  Bruch Bruch::add(Bruch b1,Bruch b2)
2  {
3      Bruch result;
4      if(b1.isNegative()==true)
5      {
6          if(b2.isNegative()==true)
7          {
8              result.setNegative(true);
9              result.setNenner(b1.getNenner()*b2.getNenner());
10             result.setZaehler(b1.getZaehler()*b2.getNenner()+b2.getZaehler()*b1.getN
```

```
11          }
12          else
13          {
14              result=subtract(b2,b1);
15          }
16      }
17      else
18      {
19          if(b2.isNegative()==true)
20          {
21              result=subtract(b1,b2);
22          }
23          else
24          {
25              result.setNenner(b1.getNenner()*b2.getNenner());
26              result.setZaehler(b1.getZaehler()*b2.getNenner()+b2.getZaehler()*b1.getN
27          }
28      }
29      result.kuerzen();
30      return result;
31  }
```

Aufgabe 3:

Schreiben Sie eine Klasse Datum. Die Klasse soll drei Attribute besitzen, welche das Jahr, den Monat und den Tag abspeichern. Benutzen Sie **enum** Attribute, wo diese sinnvoll erscheinen. Die Jahreszahl soll nur zwischen 1700 und 2399 liegen dürfen.

Die Klasse soll über eine Methode verfügen, die den jeweiligen Wochentag zum gespeicherten Datum zurückliefert. Die Methode zur Berechnung des Wochentags aus dem Datum geht zurück auf den Autor von "Alice im Wunderland" Lewis Carroll. Dazu sind vier Werte zu berechnen: einen Jahreswert, einen Monatswert, einen Tageswert und einen Korrekturwert.

Der Jahreswert berechnet sich wie folgt: Man nimmt die letzten beiden Ziffern des Jahres, bei 2007 also 07, und führt eine Ganzzahldivision durch den Wert 4 aus. Das Ergebnis addiert man zu der zweistelligen Jahreszahl. Aus diesem Wert berechnet man dann den Rest einer Ganzzahldivision durch 7. Den Monatswert entnimmt man einer Tabelle, welche in der Klasse gespei-

chert wird (Tabelle 1). Der Tageswert ist einfach der Tag des Monats. Den Korrekturwert entnimmt man wiederum einer zweiten Tabelle, welche auch in der Klasse gespeichert werden soll (Tabelle 2). Liegt das Datum im Januar oder Februar eines Schaltjahres, so muss man vom Korrekturwert noch den Wert 1 abziehen.

Die vier gefundenen Werte addiert man und berechnet den Rest der Ganzzahldivison durch 7. Für den so erhaltenen Wert kann man nun aus der Tabelle 3 den entsprechenden Wochentag entnehmen.

Damit man diese Methode implementieren kann, benötigt man noch eine weitere Methode, um herauszufinden, ob es sich bei einem gegebenen Jahr um ein Schaltjahr handelt. Die Regel dafür, ob ein Jahr ein Schaltjahr ist, ist auch recht einfach. Ist die Jahreszahl ohne Rest durch 4 teilbar, so ist es ein Schaltjahr, es sei denn, die Jahreszahl ist ein Vielfaches von 100. Ist die Jahreszahl jedoch ein Vielfaches von 400, so ist es trotzdem ein Schaltjahr.

Überprüfen Sie die Methode indem Sie den Benutzer ein Datum als drei Zahlenwerte eingeben lassen und für dieses Datum dann den Wochentag als String ausgeben. Die Jahreszahl kann man entweder als Kombination aus zwei zweistelligen Zahlen eingeben lassen, oder man erweitert die entsprechenden Funktionen so, dass die Eingabe von vierstelligen Zahlen möglich wird. Alle Eingaben sollten natürlich wie immer auf Ihre Gültigkeit überprüft werden.

Monat	Wert
Januar	0
Februar	3
März	3
April	6
Mai	1
Juni	4
Juli	6
August	2
September	5
Oktober	0
November	3
Dezember	5

Tabelle 1: Tabelle für den Monatswert.

Beispiel: `14.11.2007`

Jahreswert:	$07 + (07\ div\ 4) = 8$ $8\ modulo\ 7 = 1$
Monatswert:	Der Monat is November, also ergibt sich aus Tabelle 1 ein Wert von 3.
Tageswert:	Der Tag des Monats ist 14.
Korrekturwert:	Ergibt sich aus Tabelle 2 zu 6.
Ergebnis:	$1 + 3 + 14 + 6 = 24$ $25\ modulo\ 7 = 3$

Schaut man in der Tabelle 3 nach, so sieht man, dass der Wert 3 dem Wochentag Mittwoch entspricht.

Auch bei dieser Lösung werden wir auf den Deklarationsteil in der header Datei nicht weiter eingehen, sondern uns auf die Implementierung beschränken.

```
1 #include "Datum.h"
2
3 unsigned int monatsWert[]={0,3,3,6,1,4,6,2,5,0,3,5};
4 unsigned int tageImMonat[]={31,28,31,30,31,30,31,31,30,31,30,31};
5
```

Jahr	Wert
1700	4
1800	2
1900	0
2000	6
2100	4
2200	2
2300	0

Tabelle 2: Tabelle für den Korrekturwert.

Wochentag	Wert
Sonntag	0
Montag	1
Dienstag	2
Mittwoch	3
Donnerstag	4
Freitag	5
Samstag	6

Tabelle 3: Tabelle für den Tageswert.

```
6  Datum::Datum():jahr(1900),monat(Januar),tag(1)
7  {}
8
9  Datum::Datum(unsigned int _tag,Monat _monat, unsigned int _jahr):jahr(2007),monat(Ja
10 {
11     setJahr(_jahr);
12     setMonat(_monat);
13     setTag(_tag);
14 }
15
16 bool Datum::istSchaltjahr()
17 {
18     return ((jahr%4==0) && (jahr%100!=0 || jahr%400==0));
19 }
20
21 Wochentag Datum::berechneWochentag()
22 {
23    unsigned int zweistelligeJahreszahl=jahr%100;
24    unsigned int jahreswert=zweistelligeJahreszahl+(zweistelligeJahreszahl/4);
25    jahreswert%=7;
26    unsigned int monatswert=getMonatswert();
27    unsigned int tageswert=tag;
28    unsigned int korrekturwert=getKorrekturwert();
29    unsigned int wert=jahreswert+monatswert+tageswert+korrekturwert;
30    Wochentag wt=(Wochentag)(wert%7);
31    return wt;
32 }
33
34 unsigned int Datum::getMonatswert()
35 {
36     return monatsWert[monat-1];
37 }
38
39 unsigned int Datum::getKorrekturwert()
40 {
41     unsigned int result=0;
42     if(jahr>=1700 && jahr<1800)
43     {
44         result = 4;
45     }
46     else if(jahr>=1800 && jahr<1900)
47     {
48         result = 2;
49     }
50     else if(jahr>=1900 && jahr<2000)
```

```
51      {
52          result = 0;
53      }
54      else if(jahr>=2000 && jahr<2100)
55      {
56          result = 6;
57      }
58      else if(jahr>=2100 && jahr<2200)
59      {
60          result = 4;
61      }
62      else if(jahr>=2200 && jahr<2300)
63      {
64          result = 2;
65      }
66      else if(jahr>=2300 && jahr<2400)
67      {
68          result = 0;
69      }
70      if(istSchaltjahr() && (monat==Januar || monat==Februar))
71      {
72        result-=1;
73      }
74      return result;
75  }
76
77  unsigned int Datum::getJahr()
78  {
79      return jahr;
80  }
81
82  Monat Datum::getMonat()
83  {
84      return monat;
85  }
86
87  unsigned int Datum::getTag()
88  {
89      return tag;
90  }
91
92  std::string Datum::getWochentagName(Wochentag t)
93  {
94      std::string name;
95      switch(t)
```

```
96      {
97          case Sonntag:
98              name="Sonntag";
99              break;
100         case Montag:
101             name="Montag";
102             break;
103         case Dienstag:
104             name="Dienstag";
105             break;
106         case Mittwoch:
107             name="Mittwoch";
108             break;
109         case Donnerstag:
110             name="Donnerstag";
111             break;
112         case Freitag:
113             name="Freitag";
114             break;
115         case Samstag:
116             name="Samstag";
117             break;
118     }
119     return name;
120 }
121
122 void Datum::setJahr(unsigned int _jahr)
123 {
124     if(_jahr>=1700 && _jahr<2400)
125     {
126         jahr=_jahr;
127     }
128     else
129     {
130       jahr=2007;
131     }
132     setTag(tag);
133 }
134
135 void Datum::setMonat(Monat _monat)
136 {
137     monat=_monat;
138     setTag(tag);
139 }
140
```

```
141 void Datum::setTag(unsigned int _tag)
142 {
143
144   if(_tag!=0 && (_tag <= tageImMonat[monat-1] || (monat==Februar && istSchaltjahr()
145   {
146     tag=_tag;
147   }
148   else
149   {
150     _tag=1;
151   }
152 }
```

Zeile 1: Bindet die Deklarationen aus der Header Datei ein.

Zeile 3: Definiert ein Feld, welches die Werte aus der Tabelle für die Montaswerte enthält

Zeile 4: Definiert ein Feld, welches die Anzahl der Tage für die einzelnen Monate enthält. Für den Februar wird hier der Wert 28 angenommen und erst später in der Methode zum Setzen des Tages wird überprüft, ob es sich um ein Schaltjahr handelt falls der Wert 29 angegeben wurde.

Zeile 6 und 7: Standard Konstruktor der das Jahr, den Monat und den Tag auf Standardwerte setzt (1.1.2007). Die Werte werden über die Initialisierungsliste gesetzt.

Zeile 9-14 Konstruktor welcher das Jahr, den Monat und den Tag auf die angegebenen Werte setzt. Die Werte werden dabei über die Initialisierungsliste zuerst auf Standardwerte gesetzt und anschließend im Körper des Konstruktors über die entsprechenden Methoden der Klasse auf die angegebenen Werte gesetzt. Dies ist notwendig da die Klasse überprüft, ob es sich bei den angegebenen Werten um gültige Werte handelt.

Zeile 16-19 Die Methode *istSchaltjahr* überprüft, ob das Jahr des Datums ein Schaltjahr ist. Ist es ein Schaltjahr wird **true** zurückgeliefert, sonst **false**

Zeile 21-32 Die Methode *berechneWochentag* berechnet nach der Formel, die in der Aufgabe angegeben war, den Wochentag und liefert als Ergebnis einen **enum** Datentyp namens *Wochentag* zurück. Dieser **enum** Datentyp ist in der Header Datei deklariert. Für die Berechnung benötigt

man die hinteren beiden Stellen der Jahreszahl; diese ergibt sich ganz einfach als Rest der Ganzzahldivision (% Operator) der Jahreszahl durch 100.

Zeile 34-37 Die Methode *getMonatswert* liefert den Monatswert aus dem weiter oben deklarierten Feld zurück. Als Index dient hier direkt der ebenfalls in der Header-Datei definierte **enum** *Monat*.

Zeile 39-75 Berechnet den Korrekturwert für das Jahr basierend auf den Werten aus der entsprechenden Tabelle.

Zeile 77-90 Die Methoden liefern respektive die Jahreszahl, den Monat oder den Tag des Datums zurück.

Zeile 92-120 Liefert den Namen des Wochentages der als **enum** Wochentag übergeben wird als String zurück. Das Ganze geschieht in einer **switch** Anweisung.

Zeile 122-133 Setzt die Jahreszahl, wobei überprüft wird, ob das Jahr zwischen 1700 und 2399 liegt. Ist dies nicht der Fall wird die Jahreszahl auf 2007 gesetzt. Am Ende der methode wird die Methode *setTag* aufgerufen, denn es wäre ja möglich, dass das Datum vor dem setzen der Jahreszahl auf den 29. Februar eines Schaltjahres zeigte und dieser Tageswert für die neue Jahreszahl nun nicht mehr gültig ist.

Zeile 135-139 Setzt den Wert für den Monat neu. Auch hier wird am Ende der Methode *setTag* aufgerufen um sicherzustellen, dass der Wert für den Tag ein gültiger Wert für den neuen Wert für den Monat ist.

Zeile 141-152 Die Methode setzt den Wert für den Tag. Dabei wird überprüft, ob der angegebenen Wert für den gesetzten Monat und das gesetzte Jahr gültig ist. Ist dies nicht der Fall wird der Wert für den Tag auf 1 gesetzt.