Solutions to Lars's Übung 2

18. März 2018

1 Aufgabe 1

```c
/* Summe der ersten n Zahlen.
 * (c) 2015 Clelia und Johannes */

#include <stdio.h>

int main () {
    int n = 10;       /* Addiere bis zu dieser Zahl */
    int i;
    int summe;        /* speichert Zwischenergebnis */

    i = 0;

    while (i < n) {
      summe =+ i      /* addiere ite Zahl auf summe */
    }
    printf ("Das Ergebnis ist %f.\n", summe);
    return 0;
}
```

Wir gehen wie bei den gestrigen Aufgaben vor und versuchen zunächst den Quelltext zu kompilieren.

```
temp2.c: In function 'main':
temp2.c:15:5: error: expected ';' before '}' token
     }
     ^
temp2.c:16:13: warning: format '%f' expects argument of type 'double', but argument 2 has
     type 'int' [-Wformat=]
     printf ("Das Ergebnis ist %f.\n"
```

Es besteht also ein Fehler in Zeile 15 und es wird uns eine Warnung zu Zeile 16 ausgegeben. Der Fehler in Zeile 15 ist eigentlich in Zeile 14 zu finden, es fehlt ein Semikolon.

Die Warnung beseitigen wir, indem wir für `printf` den richtigen Platzhalter verwenden. Da es sich um ein `int` handelt, sollte der Platzhalter `%d` lauten.

```c
/* Summe der ersten n Zahlen.
 * (c) 2015 Clelia und Johannes */

#include <stdio.h>

int main () {
    int n = 10;       /* Addiere bis zu dieser Zahl */
    int i;
    int summe;        /* speichert Zwischenergebnis */

    i = 0;

    while (i < n) {
      summe =+ i;     /* addiere ite Zahl auf summe */
    }
    printf ("Das Ergebnis ist %d.\n", summe);
    return 0;
}
```

Der Code kompiliert zwar jetzt, das bedeutet jedoch nicht, dass er richtig funktioniert:

1. In Zeile 9 muss die Variable `summe` richtig initialisiert werden.
2. In der `while`-Schleife wurde vergessen die Zählervariable zu inkrementieren.
3. Die Abbruchbedingung ist so gewählt, dass $i = n$ nicht mehr zur Summe beiträgt. Hier sollte `while(i <= n)` stehen.
4. In Zeile 14 steck ein tückischer Fehler: anstelle des Inkrements `+=`, wird hier der Variable `summe`, der Wert `+i` zugewiesen.

Folgender Code ist also richtig:

```c
/* Summe der ersten n Zahlen.
 * (c) 2015 Clelia und Johannes */

#include <stdio.h>

int main () {
    int n = 10;       /* Addiere bis zu dieser Zahl */
    int i=0;          /* initialisierung */
    int summe=0;      /* speichert Zwischenergebnis */

    while (i <= n) {
      summe += i;     /* addiere ite Zahl auf summe */
      i++;
    }
    printf ("Das Ergebnis ist %d.\n", summe);
    return 0;
}
```

Diesen Code kann man natürlich optimieren, indem man die bekannte Formel für die Summe der ganzen Zahlen bis n nutzt.

```c
/* Summe der ersten n Zahlen.
 * (c) 2015 Clelia und Johannes */

#include <stdio.h>

int main () {
    int n = 10;       /* Addiere bis zu dieser Zahl */
    int i=0;          /* initializierung */
    int summe=0;      /* speichert Zwischenergebnis */

    summe=n*(n+1)/2;
    printf ("Das Ergebnis ist %d.\n", summe);
    return 0;
}
```

2 Aufgabe 2

```c
#include <stdio.h>

// wir muessen math.h einbinden und mit '-lm' kompilieren, damit
// sqrt() bekannt ist und zur Laufzeit zur Verfuegung steht
#include <math.h>

// der Primzahltest aus Algorithmus 1
int primzahltest(const int c){
  int n=2;
  // hier verwenden wir zwei sogenannte explizite 'casts', da sowohl Argument als
  // auch Rueckgabewert von sqrt 'double' sind
  while  ( ((double)n < sqrt((double)c)) && (c%n != 0)){
     n++;
  }

  if (c%n != 0) {
    return 1;
  } else {
    return 0;
  }
}

// Suche nach der ersten Primzahl groesser als 'n'
int naechste_primzahl(const int n){
  int c=n;
  // wir testen und inkrementieren 'c' so lange, bic 'c' eine Primzahl ist
  while( primzahltest(c) == 0 ){
    ++c;
  }
  printf("Erste Primzahl nach %d ist %d\n", n, c);
  return c;
}

int main(){
  int m=31;
```

```
  int testres=primzahltest(m);
  int a,b,c;
  a=naechste_primzahl(20000);
  b=naechste_primzahl(30000);
  c=naechste_primzahl(40000);
  printf("%d\n", testres);
  printf("Loesung b1 %d\n",a);
  printf("Loesung b2 %d\n",b);
  printf("Loesung b3 %d\n",c);
}
```

3 Aufgabe 3

```
// Implementierung der Quadratwurzel
#include<stdio.h>
#include<math.h> // wir benoetigen die Betragsfunktion fabs aus math.h
double quadratwurzel( double a){
  double x=2;
  double fx=1;
  while ( fabs(x-fx)>1e-10){
     x=fx;
     fx=0.5*(x+a/x);
  }
  return x;
}
int main(){
  double x=8.;
  double wurzel=quadratwurzel(x);
  printf("Quadratwurzel von %e ist %e\n", x, wurzel);
}
```

4 Aufgabe 4

```
#include<stdio.h>
int ggt( int a, int b){
  // eine unendliche Schleife
   while (1){
     if (a==0) return b;
     if (b==0) return a;
     if (a>b)
       a= a%b;
     else
       b= b%a;
   }
}
int main(){
   int a=15;
```

```
    int b=20;
    int fn=ggt(a,b);
    printf("n=%d\n",fn);
}
```

5 Aufgabe 5

Dieser Algorithmus steht im Skript.

6 Aufgabe 6

```
#include<stdio.h>
#include<math.h>
int main(){
  int k=1;
  // zwei Zwischenergebnisse
  double xk=1; // (1/1)
  double xkp1=0.5; // (1/2)

  // und unsere Summe
  double sum=0.;

  // als Abbruchbedingung haben wir entschieden, dass die Quadratwurzel sukzessiver
     Summanden
  // sich um mehr als 10^(-8) unterscheiden muessen, damit wir weiter iterieren
  while( fabs( xkp1 - xk ) > 1e-8 ){
    sum += xk/k;
    xk = xkp1;
    ++k;
    xkp1 = 1.0/(k+1);
  }
  printf("Summe: %e\n",sum);
}
```