2014

SWE

ÜBUNG 1

WOLFGANG LUMETSBERGER

S1310307025

Inhalt

1. Dreieck-Tester

1.1. Lösungsweg

Für die Prüfung eines Dreiecks soll ein Programm implementiert werden, welches 3 Übergabeparameter akzeptiert. Dies ist auch die erste Prüfung im Programm, ob die Anzahl der Übergabeparameter = 3 ist. Die eingegebenen Parameter können nun mittels der Hilfsfunktion strtof() auf Double Werte konvertiert werden.

Für diese Konvertierung wird eine eigene Funktion ConvertValue implementiert, an welcher ein String übergeben wird und ein Doublewert zurückkommt.
Wird dem Programm ein Parameter ungleich einer Zahl übergeben, so wird in der Funktion dieser erkannt und der Fehler behandelt.

Anschließend muss geprüft werden ob es sich um ein gültiges Dreieck handelt. Handelt es sich um ein gültiges Dreieck, prüfe ich die weiteren Spezialfälle (Gleichseitig, Gleichschenkelig, Rechtwinkelig). Die jeweiligen Prüfungen werde ich in eigene Funktionen auslagern, welche TRUE oder FALSE zurückliefern.

Da es in C keinen definierten Datentyp für Boolean gibt, wird dieser in einer Datei bool.h erstellt, welchen ich im zu erstellenden Programm importieren werde.

1.1.1 IsRight

Die Prüfung ob es sich um ein Rechtwinkeliges Dreieck handelt, kann mathematisch folgendermaßen berechnet werden:

a^2 + b^2 = c^2

Da ich in unserem Fall nicht weiß, welche Seiten a,b und c sind, muss ich nun folgende Fälle prüfen:

- a^2 + b^2 == c^2
- a^2 + c^2 == b^2
- b^2 + c^2 == a^2

Trifft einer der Fälle zu, so handelt es sich um ein Rechtwinkeliges Dreieck und TRUE kann zurückgegeben werden

1.1.2 IsEquilateral

Es handelt sich genau dann um ein Gleichschenkeliges Dreieck, wenn mindestens 2 Seiten des Dreiecks die gleiche Länge besitzen. Folgende Fälle müssen überprüft werden:

- a == b
- b == c
- c == a

Da man bei einem Vergleichs Operator die Seiten auch vertauschen kann, reichen diese 3 Fälle aus, um festzustellen ob es sich um ein Gleichschenkeliges Dreieck handelt. Trifft also einer der obigen Fälle zu, kann ich TRUE zurückliefern.

1.2.3 IsOsceles

Es handelt sich um ein Gleichseitiges Dreieck, wenn alle 3 Seiten gleich lange sind. Folgender Fall muss überprüft werden

- a == b == c

Trifft dieser Fall zu, so handelt es sich um ein Gleichseitiges Dreieck und ich kann TRUE zurückliefern.

1.2.4 IsTriangle

Diese Funktion liefert FALSE wenn es sich um ein ungültiges Dreieck handelt, und TRUE im Falle eines gültigen Dreiecks. Wie diese Überprüfung funktioniert, kann der Angabe entnommen werden.

1.2. Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <math.h>
#include "bool.h"

/* This makro is needed to define any EPSILON value for checking
 * if triangle is right orthogonal
 */
#define EPSILON 0.00001

/*
 * check if triangle is equilateral
 */
bool IsEquilateral(double a, double b, double c){
	return (a == b && b == c);
}
/*
 *  check if triangle is isosceles
 */
bool IsIsosceles(double a, double b, double c){
	return (a == b || b == c || c == a);
}
/*
 * check if triangle is right
 */
bool IsRight(double a, double b, double c){
	return ((fabs((a * a + b * b) - c * c) < EPSILON  )||
		(fabs((a * a + c * c) - b * b) < EPSILON  )||
		(fabs((b * b + c * c) - a * a) < EPSILON  ));
}
/*
 * check if params are a possible triangle
 */
bool IsTriangle(double a, double b, double c){
  return ((a + b <= c) || ( a + c  <= b) || (b + c <= a));
}
/*
 * convertInputValue to Double
 */
double ConvertValue(char* text){

  double help; /*Hilfsvariable um den Rückgabewert zwischen zu speichern*/
  char *endptr;   /*Pointer needed to use strtol*/

  /*set errno value to zero to be able determine if strtof produced an error*/
  endptr = NULL;
  errno = 0;
  help = strtof(text, &endptr);
```

```
  /*check if any error occured during strtof*/
  if(errno != 0){
    printf("Konvertierungsfehler: , %s \n Verwende den Wert 0",strerror(errno));
  }/*if*/
  else if ( *endptr){
    printf("Wert konnte nur teilweise Konvertiert werden: %f,
    Nicht Konvertierter teil: %s \n ", help, endptr);
    printf("Verwende den Wert %f \n",help);
  }/*if*/
  return help;
}

int main(int argc, char** argv) {

       double a,b,c;       /*Variablen for Triangle*/
       bool isAnyTriangle; /*Variable to determin if any or a Triangle is given*/

       if(argc != 4){
          printf("Falsche anzahl von Übergabeparemeter: 3 Werte erwartet \n");
              return EXIT_FAILURE;
       }/*if*/

       a = ConvertValue(argv[1]);
       b = ConvertValue(argv[2]);
       c = ConvertValue(argv[3]);

       if(IsTriangle(a,b,c)){
              printf("Es handelt sich um ein ungueltiges Dreieck \n");
              return EXIT_SUCCESS;
       }else{
              isAnyTriangle = true;
       }/*if*/

       if(IsEquilateral(a,b,c)){
              printf("Das Dreieck ist Gleichseitig & Gleichschenkelig \n");
              return EXIT_SUCCESS;
       }
       if (IsIsosceles(a,b,c)) {
              printf("Das Dreieck ist Gleichschenkelig \n ");
              isAnyTriangle = false;
       }/*if*/

       if(IsRight(a,b,c)){
              printf("Das Dreieck ist Rechtwinkelig \n");
              isAnyTriangle = false;
       }/*if*/

       if(isAnyTriangle){
              printf("Es handelt sich um kein bestimmtes Dreieck \n ");
       }/*if*/
return EXIT_SUCCESS;
}/*main*/
```

1.3. Test Cases

1.3.1. Eingabe eines Gleichseitigen Dreiecks

Das Programm wird mit folgenden Parameter aufgerufen:

- a = 3
- b = 3
- c = 3

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./dreieckTester 3 3 3
Das Dreieck ist Gleichseitig & Gleichschenkelig
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

1.3.2. Eingabe eines Gleichschenkeligen Dreiecks

Das Programm wird mit folgenden Parameter aufgerufen:

- a = 2
- b = 2
- c = 3

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./dreieckTester 2 2 3
Das Dreieck ist Gleichschenkelig
 wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

1.3.3. Eingabe eines Rechtwinkeligen Dreiecks

Das Programm wird mit folgenden Parameter aufgerufen:

- a = 10
- b = 5
- c = 11.1803398875

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./dreieckTester 10 5 11.1803398875
Das Dreieck ist Rechtwinkelig
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

1.3.4. Eingabe eines Gleichschenkeligen, Rechtwinkeligen Dreiecks

Das Programm wird mit folgenden Parameter aufgerufen:

- a = 4
- b = 4
- c = 5.656854

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./dreieckTester 4 4 5.656854
Das Dreieck ist Gleichschenkelig
 Das Dreieck ist Rechtwinkelig
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

1.3.5. Eingabe eines ungültigen Dreiecks

Das Programm wird mit folgenden Parameter aufgerufen:

- a = 1
- b = 1
- c = 2

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./dreieckTester 1 1 2
Es handelt sich um ein ungueltiges Dreieck
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

1.3.6. Eingabe von nur 2 Übergabeparameter

Das Programm wird mit folgenden Parameter aufgerufen:

- a = 1
- b= 1
- c=

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./dreieckTester 1 1
Falsche anzahl von Übergabeparemeter: 3 Werte erwartet
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

1.3.7. Eingabe von 5 Übergabeparameter

Das Programm wird mit folgenden Parameter aufgerufen:

- a = 1
- b = 1
- c = 2
- 3
- 4

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./dreieckTester 1 1 2 3 4
Falsche anzahl von Übergabeparemeter: 3 Werte erwartet
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

1.3.8. Eingabe von 2 Zahlen und Buchstaben Kombination

Das Programm wird mit folgenden Parameter aufgerufen:

- a= 1
- b= 1
- c= abc

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./dreieckTester 1 1 abc
Wert konnte nur teilweise Konvertiert werden: 0.000000, Nicht Konvertierter teil: abc
 Verwende den Wert 0.000000
Es handelt sich um ein ungueltiges Dreieck
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

1.3.9. Eingabe von 3 Buchstabenkombinationen

Das Programm wird mit folgenden Parameter aufgerufen:

- a = abc
- b= abc
- c= abc

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./dreieckTester abc abc abc
Wert konnte nur teilweise Konvertiert werden: 0.000000, Nicht Konvertierter teil: abc
 Verwende den Wert 0.000000
Wert konnte nur teilweise Konvertiert werden: 0.000000, Nicht Konvertierter teil: abc
 Verwende den Wert 0.000000
Wert konnte nur teilweise Konvertiert werden: 0.000000, Nicht Konvertierter teil: abc
 Verwende den Wert 0.000000
Es handelt sich um ein ungueltiges Dreieck
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

2. Primfaktorenzerlegung

2.1. Lösungsidee

Es wird ein Programm entwickelt, welches einen Übergabeparameter akzeptiert. Als Erstes wird geprüft, ob der Übergabeparameter vorhanden ist, wenn nicht kann das Programm mit einer Meldung beendet werden.

Ist der Parameter vorhanden, wandeln wir den Parameter mittels der atoi() Funktion auf einen Integer Wert um.

Ist der umgewandelte Wert <= 3 kann der Wert ausgegeben werden mit dem Hinweis, dass es sich um eine Primzahl handelt, und das Programm beendet werden.

Mittels einer For-Schleife durchlaufe ich nun die Zahlen von 2 bis zur eingegeben Zahl / schleifenvariable. Die schleifenvariable erhöht sich pro Durchlauf jeweils um 1.

Am Beginn dieser Schleife setzen wir unseren Zähler für die auszugebende Hochzahl pro Primzahl auf 0 (resultCount).

In dieser Schleife, baue ich erneut eine Kopfgesteuerte Schleife, welche solange durchlaufen wird, bis zahl / i keinen Rest ergibt. In dieser Schleife erhöhe ich die oben genannte Variable resultCount jeweils um 1, und setzten die Zahl auf das Ergebnis der Division von Zahl / i.

Unter der ersten Schleife gebe ich nun die geprüfte Primzahl aus. Die Ausgabe findet jedoch nur dann statt, wenn der resultCount > 1 ist.

Hat einmal eine Ausgabe stattgefunden, so merken ich mir dies in einer Boolean Variable. Beim nächsten Durchlauf, kann ich dann auf diese prüfen, und vor der Ausgabe jeweils ein Sternchen hinzufügen.

Nach der ersten Schleife kann ich nun, falls die Zahl >1 ist, dies noch ausgeben. Dies ist jeweils die letzte Primzahl, welche eventuell nur einmal vorkommt.

2.2. Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include "bool.h"

void DoPrintStar(bool print){
  if(print){
    printf(" * ");
  }/*if*/
}/*doPrintStar*/

int main(int argc, char** argv) {

  int number;             /*Variable used to store input parameter*/
  int i;                  /*Variable used as a loop counter */
  int resultCount;        /*Variable used to store the amount of one prime
number */
  bool printStar = false; /*Variable used to recognise if a star has to be
printed befor the prime*/

  /*Check if arguments are ok*/
  if(argc != 2){
    printf("Usage: Wrong Parameter: 1 Number expected\n");
    return EXIT_FAILURE;
  }/*if*/

  /*Convert input parameter to integer*/
  number = atoi(argv[1]);

  printf("Splited Number: ");

  if(number <= 3 ){
    printf(" %d^%d \n",number,1);
    return EXIT_SUCCESS;
  }/*if*/

  for(i = 2; i < number/i; i++){
    resultCount = 0;

    while( number % i == 0){
      resultCount += 1;
      number /= i;
    }/*while*/

    if(resultCount >= 1){
      DoPrintStar(printStar);
      printf(" %d^%d ",i,resultCount);
      printStar = true;
    }/*if*/
  }/*for*/

  if(number >1){
    DoPrintStar(printStar);
    printf("%d^%d ",number,1);
  }/*if*/
  printf("\n");
  return EXIT_SUCCESS;
}/*main*/
```

2.3. Testfälle

2.3.1. Überprüfen der Testfälle anhand der an der Angabe beiliegenden Tabelle

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegung 10
Splited Number:  2^1  * 5^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegung 256
Splited Number:  2^8
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegung 6534
Splited Number:  2^1  *  3^3  * 121^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegung 13332
Splited Number:  2^2  *  3^1  *  11^1  * 101^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

2.3.2. Überprüfen einer Falschen Eingabe

Ich prüfe ob mein Programm auch funktioniert, wenn ich anstatt einer Zahl, einen Buchstaben eingebe. Ich erwarte mir kein sinnvolles Ergebnis, da die Funktion atoi() zur konvertierung keine Fehlerbehandlung durchführt. Vermutetes Ergebnis wäre 0.

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegung abc
Splited Number:  0^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

2.3.3. Verhalten mit keinem Übergabeparameter

Das Programm sollte sich mit einer Meldung an den Benutzer, dass er falsche Übergabeparameter eingegeben hat beenden.

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegung
Usage: Wrong Parameter: 1 Number expected
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

2.3.4. Verhalten mit mehr als einen Übergabeparameter

Das Programm sollte sich mit einer Meldung an den Benutzer, dass er falsche Übergabeparameter eingegeben hat beenden.

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegung 1 2 3
Usage: Wrong Parameter: 1 Number expected
wolfgang@ubuntu:~/Documents/C/Uebung_1$ cl
```

2.3.5. Verhalten im Aufruf mit einer Primzahl

2.3.5.1. Primzahl 5

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegung 5
Splited Number: 5^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

2.3.5.2. Primzahl 2

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegung 2
Splited Number:  2^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

3. ERRNO & Co

3.1. Lösungsidee

Beim Studieren der Manpage von atoi, wird auf eine weitere Funktion strtol verwiesen, welche eine Fehlererkennung aufweist.

Die neue Funktion benötigt jedoch 2 Parameter mehr. Einerseits einen Pointer, welcher auf die Stelle im übergebenen String gesetzt wird, an der nicht mehr konvertiert werden konnte. Andererseits muss übergeben werden, von welchem Zahlensystem ausgegangen werden soll. In meinem Fall übergebe ich hier 10 für das Dezimalsystem.

Der Dokumentation kann man entnehmen, dass wenn ein Fehler in der Funktion strtol() auftritt, als Zahl 0 zurückgeliefert wird, jedoch eine Variable (ERRNO) mit einem ErrorCode befüllt wird. Kann nur ein Teil des übergebenen String konvertiert werden, so zeigt der übergebene Pointer auf den Restlichen String, welcher nicht mehr konvertiert wurde.

Um den Wert von ERRNO korrekt auswerten zu können, ist es wichtig den Wert genau vor dem Aufruf von STRTOL auf 0 zu setzten und ihn gleich nach dem Aufruf auszuwerten. Ebenfalls sollte der Pointer vor dem Aufruf auf NULL gesetzt werden. Zeigt dieser danach auf einen Wert, so wurde dieser nicht umgewandelt.

3.2. Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include "bool.h"

void DoPrintStar(bool print){
  if(print){
    printf(" * ");
  }/*if*/
}/*doPrintStar*/

int main(int argc, char** argv) {

  int number;             /*used to store input parameter*/
  int i;                  /*used as a loop counter */
  int resultCount;        /*used to store the amount of one prime number */
  bool printStar = false; /* used to recognise if a star has to be printed */
  char *endptr;           /*Pointer needed to use strtol*/

  /*Check if arguments are ok*/
  if(argc != 2){
    printf("Usage: Wrong Parameter: 1 Number expected\n");
    return EXIT_FAILURE;
  }/*if*/

  /*Set ERRNO to 0 to determin if strtol had an exception*/
  errno  = 0;
  endptr = NULL;
  /*Convert input parameter to integer*/
  number = strtol(argv[1], &endptr, 10);
  if(errno != 0){
    printf("Conversion error, %s \n",strerror(errno));
    return EXIT_FAILURE;
  }
  else if (endptr == argv[1]){
    printf("No Value to Convert: Please enter a number ! \n");
    return EXIT_FAILURE;
  }
  else if ( *endptr ){
    printf("Converted partially: %d, non-convertible part: %s \n ", number, endptr);
  }/*if*/

  /*Start splitting Number*/
  printf("Splited Number: ");

  if(number <= 3 ){
    printf(" %d^%d \n",number,1);
    return EXIT_SUCCESS;
  }/*if*/

  for(i = 2; i < number/i; i++){
    resultCount = 0;

    while( number % i == 0){
      resultCount += 1;
      number /= i;
    }/*while*/

    if(resultCount >= 1){
      DoPrintStar(printStar);
      printf(" %d^%d ",i,resultCount);
      printStar = true;
    }/*if*/
  }/*for*/

  if(number >1){
    DoPrintStar(printStar);
    printf("%d^%d ",number,1);
  }/*if*/
  printf("\n");
  return EXIT_SUCCESS;
}/*main*/
```

3.3. Testfälle

3.3.1. Testfälle aus 2.müssen noch alle funktionieren

Die Testfälle aus dem Übungsbeispiel 2 müssen nun ebenfalls wieder funktionieren. Ausgenommen jene, die Falsche Übergabe Parameter aufweisen. Diese werden in den folgenden Testfällen speziell behandelt.

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert 10
Splited Number:  2^1  * 5^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert 256
Splited Number:  2^8
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert 6534
Splited Number:  2^1  *  3^3  * 121^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert 13332
Splited Number:  2^2  *  3^1  *  11^1  * 101^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert 5
Splited Number: 5^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert 2
Splited Number:  2^1
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert
Usage: Wrong Parameter: 1 Number expected
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert 1 2 3
Usage: Wrong Parameter: 1 Number expected
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

3.3.2. Eingabe einer Buchstabenfolge

Bei der Eingabe einer reinen Buchstabenfolge, sollte das Programm keine Primzahl berechnen können. Des Weiteren sollte eine Ausgabe stattfinden, dass kein Wert konvertiert werden konnte, und der Benutzer mindestens eine Zahl eingeben sollte.

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert abc
No Value to Convert: Please enter a number !
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

3.3.3. Eingabe einer Zahlenfolge gefolgt von einer Buchstabenfolge

Bei der Eingabe von Zahlen gefolgt von Buchstaben, sollte das Programm eine Primzahl berechnen, mit der Zahl die übergeben wurde. Jedoch wird dem User ausgegeben, dass nicht der gesamte eingegeben Wert umgewandelt werden konnte. Bei dieser Meldung wird der Teil ausgegeben, welcher nicht mehr umgewandelt werden konnte und zusätzlich die umgewandelte Zahl. Das Programm sollte danach versuchen, den umgewandelten Teil in seine Primfaktoren zu zerlegen.

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert 256abc
Converted partially: 256, non-convertible part: abc
 Splited Number:  2^8
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```

3.3.4. Aufruf ohne einen Übergabeparameter

Wir das Programm ohne Übergabeparameter aufgerufen, dann wird dem Benutzer eine Fehlermeldung ausgegeben, und das Programm sollte beendet werden.

```
wolfgang@ubuntu:~/Documents/C/Uebung_1$ ./primFaktorenZerlegungErweitert
Usage: Wrong Parameter: 1 Number expected
wolfgang@ubuntu:~/Documents/C/Uebung_1$
```