Prof. G. Zachmann
Dipl.-Inf. C. Giesemann

TU Clausthal
Institut für Informatik
22. Dezember 2005

Wintersemester 2005/2006

Übungen zu Grundlagen der Programmierung in C - Blatt VIII

Abgabe vom 11.1.2006 bis 17.1.2006 in der angemeldeten Übung

Aufgabe 1 (Wurzelberechnung nach Heron, *6 Punkte*)

Hinweise:
Der Grundgedanke dieses, nach dem Griechen Heron benannten Verfahrens, ist geometrisch. Die Quadratwurzel einer nicht negativen Zahl **a** läßt sich als die Kantenlänge eines Quadrates mit der Fläche **a** beschreiben. Bei diesem Verfahren wird das Quadrat durch ein flächengleiches Rechteck solange angenähert bis seine Kantenlängen hinreichend gleich sind und sie somit der gesuchten Wurzel entsprechen.

- Wählen Sie als Anfangsbedingung die Länge der ersten Rechteckkante **x** beliebig > 0 z.B. als x = a / 2 . Damit die Fläche des Rechtecks gleich **a** ist, muß die andere Kante **y** wie folgt berechnet werden: y = a / x .
- Berechnen Sie ein neues Rechteck, dessen eine Kante dem arithmetischen Mittel der Kanten des letzten Rechtecks entspricht: x = (x + y) / 2 .
  Die Länge der zweiten Kante **y** wird wieder mit: y = a / x bestimmt, damit der Flächeninhalt konstant **a** bleibt.
- Wiederholen Sie den vorherigen Schritt bis der Betrag von x - y hinreichend klein ist: $|x - y| < \epsilon$.

1. Schreiben Sie ein Programm, das zu einer eingegebenen positiven reellen Zahl die Quadratwurzel nach dem Heron-Verfahren berechnet und ausgibt. Überprüfen Sie, ob die Eingabe zulässig ist. Für den Mittelwert und die Betragsbildung ist je eine eigene Funktion zu programmieren (ohne Benutzung der jeweiligen Bibliotheksfunktionen). Als Parameter sollen jeweils **x** und **y** übergeben werden und als Rückgabewert soll das Ergebnis geliefert werden. Überlegen Sie wie ϵ sinnvoll zu wählen ist und warum dieses überhaupt nötig ist. Testbeispiele:
   - $a = 4, \sqrt{a} = 2$
   - $a = 2, \sqrt{a} = 1.414$
   - $a = 0.25, \sqrt{a} = 0.5$

Aufgabe 2 (Rechnen mit komplexen Zahlen, *6 Punkte*)

Hinweise:
Die Rechenregeln für komplexe Zahlen sind z.B. unter http://de.wikipedia.org/wiki/Komplexe_Zahl zu finden.

1. Schreiben Sie ein Programm, das Rechenaufgaben mit 2 komplexen Operanden einliest und deren Ergebnis als eine komplexe Zahl ausgibt. Der Eingabestring soll folgenden Aufbau haben:

   <Operand1><Space><Operationssymbol><Space><Operand2>

Die komplexen Zahlen als Operanden sollen dabei wie folgt dargestellt werden:

(<Realteil><Space><Imaginärteil>i)

um die Klammern und das i einzulesen, müssen diese auch im scanf-Formatstring erscheinen:

`scanf("(%f %fi) ...`

Realteil und Imaginärteil sind `float` Zahlen, i ist die imaginäre Einheit, die Klammern dienen zur Abgrenzung (siehe Testbeispiele).
Als Operationssymbole sind jene der 4 Grundrechenarten (+, −, ∗ *und* /) erlaubt.
Lesen Sie die Eingabe mit einer `scanf` Anweisung und einem geeigneten Formatstring ein. Für die programminterne Darstellung aller komplexen Zahlen ist eine geeignete `struct` zu deklarieren. Das Operationssymbol ist in einem `char` zu speichern. Für jede der 4 Grundrechenarten ist eine eigene Funktion zu entwickeln, die 2 komplexe Zahlen als Parameter und eine komplexe Zahl als Rückgabewert hat. Das Hauptprogramm besteht entsprechend nur aus Eingabe mit Fehlerbehandlung, Fallunterscheidung mit Funktionsaufruf und Ausgabe. Testbeispiele:

Operation	Eingabe	Ausgabe
Addition	(4.0 3.3i) + (1.1 0.3i)	(5.1 3.6i)
Subtraktion	(6.7 -1.3i) - (3.3 3.0i)	(3.4 -4.3i)
Multiplikation	(2.5 1.0i) ∗ (-4.0 3.0i)	(-13.0 3.5i)
Division	(-4.0 2.0i) / (1.0 -1.0i)	(-3.0 -1.0i)

Aufgabe 3 (Caesar Chiffre, *6 Punkte*)

Hinweise:
Die Caesar Chiffre ist ein einfaches Verfahren zur Verschlüsselung von Texten, welches schon der Römer Gaius Iulius Caesar benutzt haben soll. Jeder Buchstabe im Text wird durch den Buchstaben ersetzt, der im Alphabet um k Stellen zirkular versetzt ist. Wenn z.B. k = 2 gewählt ist, wird A durch C, B durch D ... Y durch A und Z durch B ersetzt. Die Entschlüsselung ist eine Verschlüsselung mit negiertem k. Für die Lösung der Aufgabe benötigen Sie die ASCII Tabelle, welche z.B. unter http://de.wikipedia.org/wiki/ASCII zu finden ist. Den Großbuchstaben sind die hexadezimalen Werte 0x41 bis 0x5A zugeordnet. Die Zahlwerte der Kleinbuchstaben sind um 0x20 größer als die der Großbuchstaben. Der Datentyp `char` ist demnach sowohl der Datentyp für Zeichen, als auch ein Ganzzahlendatentyp, der die Werte 0 bis 255 (bzw. hexadezimal 0x00 bis 0x7F) annehmen kann. Sie können mit dem `char` Datentyp wie mit dem `integer` Datentyp Berechnungen ausführen. Beispiel: Wenn eine `char` Variable, in der das Zeichen 'a' gespeichert ist, mit + 2 erhöht wird, steht darin das Zeichen 'c'.

1. Schreiben Sie ein Programm, das einen Verschiebewert **k** und eine Textzeile (Länge max. 80 Zeichen) anfordert und einliest. Danach soll der eingegebene Text zeichenweise verschlüsselt und ausgegeben werden. Für den Algorithmus sollen folgende Vereinbarungen gelten:
   - Kleinbuchstaben werden auf Kleinbuchstaben und Großbuchstaben werden auf Großbuchstaben abgebildet. Verwenden Sie eine Funktion für die Umwandlung der Buchstaben, welche einen Buchstaben und den Verschiebewert **k** als Parameter erhält und den verschlüsselten Buchstaben als Rückgabewert liefert.
   - Sonderzeichen und Ziffern sollen unverändert wieder ausgegeben werden.

   Testen sie, ob Ihr Programm sowohl verschlüsseln, als auch entschlüsseln kann. Testbeispiel:
   In einem Film heißt ein Computer HAL. Nehmen wir an, die Filmemacher meinten etwas anderes, was sie aber nicht sagen durften und haben daher eine Verschlüsselung mit k = -1 angewandt. Nach Entschlüsselung mit k = 1 erhalten wir IBM und sind in unseren Verdacht bestätigt.
   Weit verbreitet ist auch ROT13 (zum Teil auch in Betriebssystemen als rot13 aufrufbar). Bei dieser Verschlüsselung "Rotiere um 13" ist k = 13. Da das Alphabet 26 Zeichen hat, liefert k = 13 das gleiche wie k = -13. Hierbei kann man also auch mit dem nicht-invertierten **k** entschlüsseln. Probieren Sie dieses aus.

Aufgabe 4 (Wurzelberechnung rekursiv, *6 Punkte*)

Hinweise:
Zur Fehlersuche ist es sinnvoll, eine bedingte Ausgabe der Werte von **x** und **y** in den jeweiligen Rekursionsschritten einzubauen, die nur gemacht wird, wenn eine Programmkonstante, z.B. Debug, auf `true` gesetzt ist.

1. Schreiben Sie ein Programm, das die Quadratwurzel wie in Aufgabe 1 berechnet. Ändern Sie aber dieses mal die iterative Berechnung in eine rekursive um. Definieren sie dazu Epsilon und **a** als globale Variablen und benutzen Sie zur Berechnung eine rekursive Funktion mit **x** als Parameter.