Übungen zu "Einführung in die Programmierung für Ingenieure (C)" im WS 98/99

Die im Wintersemester 1998/99 gestellten Übungsaufgaben gleichen weitgehend den Übungen des Sommersemesters 1998 und sind zur Vorbereitung auf die Scheinklausur gedacht. Studenten, die

- bereits im SS98 die Zulassungsvoraussetzung zur Klausur erfüllt haben, müssen die Programmieraufgaben nicht abgeben. Es wird allerdings empfohlen, sie dennoch zu bearbeiten.
- im SS98 nicht an den Übungen teilgenommen oder die Zulassungsvoraussetzungen nicht erfüllt haben, müssen die Programmieraufgaben bearbeiten und bis spätestens Fr. 15.1.1999 abgeben.

Um ein gewisses Feedback über die Qualität der erarbeiteten Lösungen zu erhalten, wäre es sinnvoll, die Programmieraufgaben bis zum jeweilig angegebenen Abgabetermin in den Kasten vor dem Raum 0.044 einzuwerfen und ab und zu die verbesserten Lösungen abzuholen. Um die Zulassungsvoraussetzung zu erhalten, wird die korrekte Bearbeitung von 70% der gestellten Programmieraufgaben erwartet.

Falls Sie Fragen zur Bearbeitung der Aufgaben haben, können Sie sich an die Übungsleiter für Systemprogrammierung wenden (Rechnerraum 01.155, Mo. 16.00-18.00, Di. 16.00-18.00, Mi. 14.00-16.00) oder bei uns vorbeikommen (Jörg Barner, Zi. 0.040 oder Bernd Schnitzer, Zi. 0.044).

Falls Sie ein Login benötigen oder Ihren alten Account weiter benutzen wollen, können Sie sich ab Mi. 11.11.1998 anmelden. Um das Anmeldeprogramm zu starten geben Sie als Login-Namen “can” ein. (Tragen Sie sich in eine beliebige Tafel- und Rechnerübung ein – diese Angabe wird im Wintersemester nicht verwendet.)

Die Klausur findet wahrscheinlich gegen Ende Januar 1999 statt. Die Anmeldung zur Klausur erfolgt Mitte Januar, der genaue Termin wird durch Aushang bekanntgegeben (Schwarzes Brett des Lehrstuhls IV und über dem Abgabekasten).

Übung 1

Aufgabe 1

Wie geht man vor, wenn man eine Aufgabe “Programmieren” möchte?

Aufgabe 2

a) Welche Schritte sind nötig, um ein C-Programm auf den Ausbildungsrechnern einzugeben und auszutesten?
b) Welche Befehle braucht man für die einzelnen Schritte?

Aufgabe 3

c) Was leistet ein Compiler?
d) Was ist der Unterschied zwischen einer “Warning” und einem “Error”

Aufgabe 4(Rechnerübung ohne Abgabe)

Loggen Sie sich ein und geben dann das Programm polynom.c aus der ersten Vorlesung ein. Übersetzen Sie es und lassen Sie es mit verschiedenen Eingabedaten ablaufen!

Übung 2

Aufgabe 5

Schreiben Sie die folgenden Ausdrücke in mathematischer Schreibweise als C-Ausdrücke:

$$d = \frac{b}{2a} + 8\left(\frac{2ab - a^2}{4c}\right)$$

$$e = 2r^2p + 2prh$$

$$f = \sqrt[3]{x^y}$$

$$g = \sin^2 x + \cos y^2$$

(Die beiden oberen Ausdrücke können Sie mit den in der Vorlesung vorgestellten Operatoren lösen. Für die unteren Ausdrücke werden zusätzliche Mechanismen benötigt, die in den Übungen vorgestellt werden.)

Aufgabe 6

Ermitteln Sie die Werte der Variablen a,b,c vom Typ `integer` nach der Ausführung des folgenden Programms.

```
int main() {
    int a = 12;
    int b = 5;
    int c = 20;

    if ( a%3 == 0 )
    {
        b = ++c/7;
        a *= 30/b;
    }
    if( 2*b*c-6 <= a)
    {
        if (0 != 120-a || c/b >= 2)
        {
            a /= 2;
            b += a;
        }
        else
        {
            a *= 2;
            b -= a;
        }
    }
/* welche Werte haben a, b und c an dieser Stelle??? */
}
```


Aufgabe 7

Welche der folgenden Zeichenfolgen sind gültige Bezeichner in C?

a) EinName
b) a3
c) 3_a
d) Einfuehrung_in_die_Programmierung_fuer_Ingenieure
e) float
f) Überschußanteil
g) A-1

Aufgabe 8

Erstellen Sie ein kurzes Programmstück, das überprüft, ob eine eingegebene Jahreszahl zu einem Schaltjahr gehört! Erstellen Sie dazu zuerst ein Struktogramm

Ein Jahr ist ein Schaltjahr wenn die Jahreszahl ein Vielfaches von 4, nicht aber durch 100 teilbar ist. Zusätzlich sind Jahre, die durch 400 teilbar sind, auch Schaltjahre.

Aufgabe 9(Abgabe bis Fr. 13.11.98)

Schreiben Sie ein Programm, das ein vom Anwender eingegebenes Datum (Tag, Monat, Jahr) auf Korrektheit überprüft. Verwenden Sie dazu das in Aufgabe 8 erstellte Programmstück zum Test auf Schaltjahre.

(**Hinweis**: zum Ausdrucken Ihres C-Programms verwenden Sie auf den Übungsrechnern bitte das Kommando

lpr -Pr01-155 *Programmname*.c

Der Ausdruck erscheint dann auf dem Drucker im Raum 01.155.)

Übung 3

Aufgabe 10

Welche Fehler finden Sie in dem folgenden Testprogramm

```
*/ Testprogramm /*
int main()
(
    int limit count
    unsinged long quad;

    while(1)
        printf(Bitte geben Sie die obere Grenze an:);
        scanf("&d", %limit);
        if (limit .gt. 0) then {
            printf(\n");
            for(count=1, count<=Limit, count++) {
                qad = count*count;
                printf("%d * %d = %d\n",count,count,quad);
            }
        }
        else break;
    }
    if(count==0)
        if(quad=0)
            printf("alle Zahlen = 0');
    else
        printf("alle Zahlen ungleich 0")
}
```

Aufgabe 11

Welche der nachfolgenden Ausdrücke stellen korrekte C-Konstanten dar. An welche Datentypen können folgende Konstanten sinnvollerweise zugewiesen werden?

-1.0e5	0.5e0.5	0815
4711	0xAFFE	’\n’
’\033’	3.14159	17E-4711

Aufgabe 12

Wie oft werden die folgenden for-Schleifen durchlaufen (überlegen und am Rechner ausprobieren!)? Wie erklären Sie sich dieses Verhalten?

```
int i;
float f;
double d;

for(i=1;i<=10;i=i+1) printf("%d\n",i);
for(f=0.1;f<=1.0;f=f+0.1) printf("%f\n",f);
for(d=0.1;d<=1.0;d=d+0.1) printf("%lf\n",d);
```

Aufgabe 13(Abgabe bis Fr. 20.11.98)

Schreiben Sie ein Programm, das von einer vom Anwender abzufragenden, ganzen Zahl ermittelt, ob es sich um eine Primzahl handelt oder nicht. Sollte es sich um eine Primzahl handeln, stopt das Programm mit einer Meldung der Art:

```
>n< ist eine Primzahl
```

Sonst wird eine Primzerlegung der eingegebenen Zahl ausgegeben, z.B. in der Art

```
2*2*3*5*17*31 = 31620
```

Übung 4

Aufgabe 14

Wie können Sie in C eine neue Funktion definieren?

Was ist der Unterschied zwischen Funktionen und Prozeduren?

Gibt es in C Prozeduren?

Welchen Vorteil hat der Einsatz von Funktionen in Programmen?

Aufgabe 15

Implementieren Sie eine Funktion pi(), die π mit Hilfe der folgenden Reihe approximiert:

$$\frac{\pi}{4} = \sum_{n=0}^{\infty} (-1)^n \frac{1}{2n+1}$$

Berechnen Sie die Reihe, bis $\frac{1}{2n+1}$ unter eine vorzugebende Grenze fällt.

Aufgabe 16(Abgabe bis Fr. 27.11.98)

Schreiben Sie ein Programm, das die n-te Wurzel ($n \in N$) einer reellen Zahl berechnet. Das Ergebnis soll mit Hilfe einer Intervallschachtelung approximiert werden.

Verwenden Sie hierzu keine der in der Header-Datei `math.h` vorgegeben Funktionen, sondern schreiben Sie eigene Funktionen die die benötigte Funktionalität bereitstellen:

`double potenz(double x, int y)`

berechnet x^y zu einer Zahl x und dem ganzzahligen Exponenten y

`double wurzel(double x, int n, double epsilon)`

berechnet die n-te Wurzel ($n \in N$) einer positiven Zahl x mit Hilfe einer Intervallschachtelung

Hinweise:

- Bei der Intervallschachtelung wird ein Anfangsintervall – repräsentiert durch Ober- und Untergrenze, die vom Programm selbständig geeignet gewählt werden müssen – in zwei Teilintervalle geteilt (die Teilung soll in der Intervallmitte stattfinden).
- Nun muß ermittelt werden, welches der beiden Teilintervalle die gesuchte Lösung enthält (hierbei kann die Potenzfunktion helfen...).
- Das so ermittelte Intervall kann nun seinerseits als Anfangsintervall betrachtet werden.
- Die oben genannten Schritte werden wiederholt, bis eine gewisse, vorzugebende Genauigkeit erreicht wird (d.h. bis die Intervallänge epsilon unterschreitet).

Übung 5

Aufgabe 17

Was berechnet das folgende Programmstück(führen Sie einen Trace durch)?

```
#include <stdio.h>

int a  =0;
int sum=0;

int f(int a)
{
    int b=0;
    if(a>0)
        while(a) b+=a--;
    sum += b;
    return b;
}

int main()
{
    int b;
    printf("Bitte Endwert eingeben: ");
    scanf("%d",&b);
    if(b>0)
        for(a=1;a<=b;a++)
            printf("\tf(%d) = %d\n",a,f(a));
    printf("sum : %d\n", sum);
    return 0;
}
```

Aufgabe 18

In Aufgabe 17 liefert main() einen Return-Wert. Wer verarbeitet diesen und wofür kann er verwendet werden?

Aufgabe 19

Schreiben Sie eine rekursive Funktion, die die n-te Fibonacci-Zahl berechnet. Die Fibbonacci-Zahlen sind durch folgende rekurrente Formel definiert:

$$fib(0) = 0$$

$$fib(1) = 1$$

$$fib(n) = fib(n-1) + fib(n-2), \qquad \text{für } n \geq 2$$

Testen Sie die Funktion, indem Sie in main() wiederholt vom Benutzer Werte eingeben lassen und die entsprechnede Fibonacci-Zahl berechnen und ausgeben. Überlegen Sie sich ein geeignetes Abbruchkriterium.

Aufgabe 20(Abgabe bis Fr. 04.12.98)

(Lassen Sie sich nicht vom Umfang der Aufgabe erschrecken; die Lösung ist kürzer als die Aufgabenstellung)

Binomialkoeffizienten – Pascalsches Dreieck

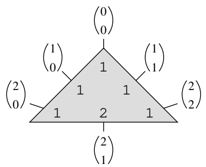
Die Binomialkoeffizienten $\binom{n}{k}$ lassen sich durch folgende rekursive Berechnungsvorschrift ermitteln:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

$$\binom{0}{0} = 1$$

$$\binom{n}{k} = 0 \qquad \text{für} \begin{cases} k > n \\ k < 0 \\ n < 0 \end{cases}$$

Diese Berechnungsvorschrift spiegelt sich im Pascalschen Dreieck wider: In Zeile n an k-ter Stelle steht der Wert des Binomialkoeffizienten $\binom{n}{k}$, der sich aus der Addition der beiden darüberliegenden Werte ergibt:



a) Schreiben Sie eine Funktion, die **rekursiv** – analog der oben angegebenen Formel – den Binomialkoeffizienten $\binom{n}{k}$ berechnet.

b) Die Binomialfunktion aus a) soll in einem eigenen Modul liegen. Erstellen Sie eine zugehörige Header-Datei, die den Funktionsprototypen (= Funktionsdeklaration) der Binomialfunktion enthält.

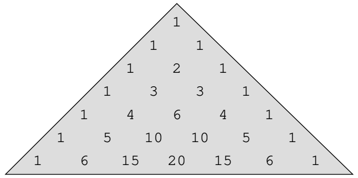
c) Implementieren Sie – in einer eigenen Datei – ein kleines Programm zum Testen des Binomialmoduls. Das Testprogramm soll wiederholt Werte für n und k einlesen, $\binom{n}{k}$ berechnen und das Ergebnis ausgeben. Überlegen Sie sich ein geeignetes Abbruchkriterium.

d) Um die Binomialfunktion effizienter zu gestalten, sollen alle bereits berechneten Werte in einem zweidimensionalen Array abgelegt werden, so daß nachfolgende Aufrufe auf bereits berechnete Werte zurückgreifen können. Erweitern Sie das Binomialmodul entsprechend um ein Array, das nur im Modul global bekannt sein soll. Die Größe des Arrays soll durch eine Präprozessor-Konstante in der zugehörigen Header-Datei festgelegt werden.

Übungen zu "Einführung in die Programmierung für Ingenieure (C)" im WS 98/99

Hinweis: Erinnern Sie sich an den Zusammenhang von Binomialkoeffizienten und dem Pascalschen Dreieck und speichern Sie berechnete Werte an der entsprechenden Stelle.

Pascalsches-Dreieck

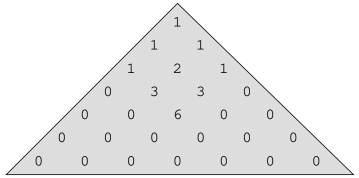


als 2-dim. Array

1						
1	1					
1	2	1				
1	3	3	1			
1	4	6	4	1		
1	5	10	10	5	1	
1	6	15	20	15	6	1

e) Bevor das Array verwendet wird, sollte es mit Nullen initialisiert werden. Hierfür benötigen Sie eine weitere Funktion im Binomialmodul, die alle Elemente des Arrays auf 0 setzt (vergessen Sie nicht den Prototypen in der Header-Datei einzutragen).

f) Ergänzen Sie das Binomialmodul um eine Ausgabefunktion, die das Array als Pascalsches Dreieck ausgibt. Nach der Berechnung von $\binom{4}{2}$ und Array-Größe 7 sollte die Ausgabe etwa so aussehen:



(Diese Funktion kann für eine etwaige Fehlersuche sehr nützlich sein ...)

g) Setzen Sie die Array-Größe auf 50, kompilieren Sie Ihr Testprogramm erneut und berechnen Sie, wieviele Möglichkeiten es beim Lotto gibt, 6 aus den 49 Kugeln zu ziehen.

Einführung in die Programmierung für Ingenieure (C)
© *Jürgen Kleinöder* Universität Erlangen Nürnberg, IMMD IV, 1998

Übung 6

Aufgabe 21

Erstellen Sie ein Programm, das die Determinante einer 3x3-Matrix ganzer Zahlen nach der Regel von Sarrus berechnet. Diese lautet:

$$\det(A) = \begin{vmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{vmatrix} = \begin{array}{l} a_{0,0}a_{1,1}a_{2,2} + a_{1,0}a_{2,1}a_{0,2} + a_{2,0}a_{0,1}a_{1,2} \\ - a_{0,2}a_{1,1}a_{2,0} - a_{1,2}a_{2,1}a_{0,0} - a_{2,2}a_{0,1}a_{1,0} \end{array}$$

Aufgabe 22

Welche der nachfolgenden Variablendefinitionen mit Initialisierung sind korrekt, welche Wirkung besitzen sie bzw. welche Unterschiede bestehen zu den anderen:

Einfache Felder:

a) int i[] = {1,2,3};
b) int i[3] = {1,2,3};
c) int i[3] = {1,2};
d) int i[] = {1,2};

Mehrdimensionale Felder:

e) int m[2][2] = {1,2,3,4};
f) int m[2][2] = {{1,2},{1,2}};
g) int m[][2] = {1,2,3,4};
h) int m[2][] = {1,2,3,4};
i) int m[][] = {1,2,3,4};
j) int m[2][2] = {1,2,3};
k) int m[2][2] = {{1,2},{3}};

Aufgabe 23(Abgabe bis Fr. 11.12.98)

Schreiben Sie ein Programm, das mit Hilfe des "Bubble-Sort"-Algorithmus ein Array (= Feld) von Zahlen sortiert. Der Benutzer soll nach der Anzahl der zu sortierenden Zahlen gefragt werden. Diese werden anschließend von ihm eingegeben.

Sie sollen ein Modul entwerfen, das die benötigten Funktionen zum initialisieren des Arrays (einlesen der Werte vom Benutzer), zur Ausgabe sowie zum sortieren des Arrays enthält (vergessen Sie nicht die Header-Datei mit den Funktionsdeklarationen). Außerdem müssen häufig zwei Werte des Arrays ihren Platz tauschen, was ebenfalls von einer Funktion des Moduls erledigt werden soll. Beachten Sie, daß sich das Array, auf dem die Funktionen arbeiten, nicht im Modul befindet, sondern als Funktionsparameter übergeben werden soll. Zudem muß den Funktionen mitgeteilt werden, wieviele Elemente des Feldes belegt sind.

Zum Testen des Moduls müssen Sie ein kleines Programm implementieren, in dem ein Array definiert wird (bitte nicht global!) und die oben genannten Anforderungen (Einlesen der Werte, Sortieren, etc) mit Hilfe des Moduls erledigt. Das Array soll so angelegt werden, daß es in jedem Fall groß genug für die aufzunehemenden Zahlen ist (z.B. 1000 Elemente).

Beschreibung des "Bubble-Sort"-Algorithmus:

Die Werte im Array werden in aufsteigender Reihenfolge sortiert, indem das Array von vorne durchlaufen wird und je zwei benachbarte Werte ihren Platz tauschen, falls der erste größer als der zweite ist. So steigt in einem Durchlauf das größte Element wie eine "Blase" ans Ende des Arrays. Das wird wiederholt, bis in einem Durchlauf keine Elemente mehr getauscht werden, also die Elemente in der richtigen Reihenfolge sind. Beachten Sie auch, daß nach jedem Durchlauf ein Element mehr an der richtigen Position steht, d.h. es müssen nur die Elemente vom Anfang des Arrays bis zu dieser Position betrachtet werden...

Übungen zu "Einführung in die Programmierung für Ingenieure (C)" im WS 98/99

Einführung in die Programmierung für Ingenieure (C)
© Jürgen Kleinöder Universität Erlangen Nürnberg, IMMD IV, 1998

Übung 7

Aufgabe 24

Erstellen Sie für jede der in Aufgabe 20 implementierten Funktionen (main, binom, init, show) ein Struktogramm, mit den im Vorlesungsskript dargestellten Elementen.

Aufgabe 25

Erstellen Sie ein Programm, das basierend auf der in der Vorlesung vorgestellten Struktur "student" die Daten einer Gruppe von Studenten (z.B. gruppe8) einliest. Anschließend soll für jeden Studenten ermittelt werden, ob dieser die Klausur bestanden hat (es besteht jeder mit einer durch 3 oder 5 teilbaren Matrikelnummer ;-)). Abschließend sollen die Datensätze aller Studenten wieder ausgegeben werden.

Aufgabe 26(Abgabe bis Fr. 18.12.98)

Die Berechnung der Mandelbrotmenge ("Apfelmännchen") geschieht in der komplexen Zahlenebene. Als Vorbereitung hierfür soll in dieser Aufgabe nun ein Modul zum Umgang mit komplexen Zahlen entwickelt werden.

a) Eine komplexe Zahl kann durch ihren Real- und Imaginärteil repräsentiert werden. Erzeugen Sie dementsprechend einen neuen Datentyp, der für die Aufnahme komplexer Zahlen geeignet ist.

b) Natürlich benötigt man auch Rechenoperationen auf den komplexen Zahlen, da die Standard-C-Operatoren +, -, *, /, usw. bekanntlich nur für die C-Datentypen definiert sind. Schreiben Sie für jede der unten aufgeführeten Operationen eine Funktion und erstellen Sie ein Modul, das die komplexen Zahlen als eine Einheit kapselt.
Überlegen Sie sich sinnvolle Signaturen (return-Wert und Funktionsparameter) für die Funktionen.

c) Schreiben Sie zum Testen Ihres Moduls ein kleines Programm, mit dem Sie komplexe Zahlen einlesen und mit den Funktionen verknüpfen können.

Sei $z = (a, bi)$ eine komplexe Zahl. Dann gilt
$Re(z) = a$ und
$Im(z) = b$.

Seien z, z_1 und z_2 komplexe Zahlen. Dann gelten folgende Regeln für die Operationen:

Addition: $z = z_1 + z_2 \rightarrow$
$Re(z) = a_1 + a_2 \qquad Im(z) = b_1 + b_2$

Subtraktion: $z = z_1 - z_2 \rightarrow$
$Re(z) = a_1 - a_2 \qquad Im(z) = b_1 - b_2$

Multiplikation: $z = z_1 z_2 \rightarrow$
$Re(z) = a_1 a_2 - b_1 b_2 \qquad Im(z) = a_1 b_2 + a_2 b_1$

Division: $z = \frac{z_1}{z_2} \rightarrow$
$Re(z) = \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} \qquad Im(z) = \frac{a_2 b_1 - a_1 b_2}{a_2^2 + b_2^2}$

Konjugation: $z = \bar{z}_1 \rightarrow$
$Re(z) = a_1 \qquad Im(z) = -b_1$

Betrag: $|z| = \sqrt{a^2 + b^2}$

Übung 8

Aufgabe 27

Was leistet das folgende Programm? Warum würde der Versuch, den Wert des Pointers **string** in *main()* zu ändern, zu einer Fehlermeldung des Compilers führen, während dies in *conv()* offensichtlich funktioniert? Wo erfolgt die Definition der Macros *isupper(), tolower()* und *toupper()?* Warum erscheint in dem Befehl *scanf()* vor der Variable **string** kein **&**? Wie können Sie das Programm beenden?

```
#include <stdio.h>
#include <ctype.h>
void conv(char *string)
{
    while(*string)
    {
        if(isupper(*string))
            *string = tolower(*string);
        else
            *string = toupper(*string);
        string++;
    }
}

int main()
{
    char string[128];
    while(1)
    {
        printf("Bitte geben Sie ein Wort ein: ");
        if(scanf("%s", string)<EOF)
            break;
        conv(string);
        printf("Konvertiert: %s\n", string);
    }
    return 0;
}
```

Aufgabe 28

Angenommen, die Definitionszeile **int i=10, j=1, *p, z[10];** sei in einem Programm enthalten. Was sind dann die Inhalte der jeweils verwendeten Variablen nach folgenden Anweisungen?

a) p = &z[5];
b) p = &j; i = *p;
c) p = z; i = *p++;
d) p = &j; i = (*p)++;
e) p = z; i = *++p;
f) p = &z[7]; p++; i = p-&z[4];

Was ist an den folgenden Anweisungen falsch:

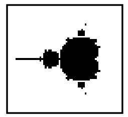
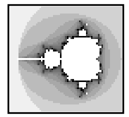
g) z[10] = 10;
h) *z+1 = 1;
i) p = &z[5]; *(p-6) = 4;

<u>Die Mandelbrotmenge</u>

Die Mandelbrotmenge – benannt nach Benoit Mandelbrot – ist eine Menge von komplexen Zahlen. Die Zugehörigkeit einer komplexen Zahl c zu dieser Menge kann mittels eines Tests, basierend auf der Formel $z = z^2 + c$, bestimmt werden, wobei die zu überprüfende Zahl c konstant bleibt und über die komplexe Zahl z iteriert wird (d.h. z wird ständig neu berechnet, beginnend mit $z = (0, 0)$).

Die Zahl c gehört genau dann zur Mandelbrotmenge, wenn der Betrag von z ein gewisses Limit (normalerweise 2) nie überschreitet – also wenn bei jeder Iteration der obigen Formel $|z| < 2$ gilt.

Das linke der beiden folgenden Bilder zeigt die Mandelbrotmenge (schwarz) im Bereich [(-2.2, -1.5), (1.0, 1.5)] der komplexen Zahlenebene. Um ein Bild interessanter zu gestalten färbt man die Bereiche der komplexen Zahlenebene ein, die nicht zur Mandelbrotmenge gehören, wobei für jeden getesteten Punkt der Ebene die Anzahl der benötigten Iterationen bis zum Überschreiten des Limits den Farbwert bestimmt. Im rechten Bild entsprechen dunkle/schwarze Bereiche hohen Iterationstiefen; die Mandelbrotmenge selbst wurde weiß belassen.

Aufgabe 29(Abgabe bis Fr. 08.01.99)

a) Legen Sie in Ihrem Home-Verzeichnis ein Unterverzeichnis (z.B. mandel) an und kopieren Sie die Dateien mandel.c, draw.c und draw.h in dieses Verzeichnis. Sie finden diese Dateien im Verzeichnis /proj/i4cing/mandel/ oder im WWW unter der URL http://www4.informatik.uni-erlangen.de/Lehre/SS98/V_C/Uebung/ . Die Datei mandel.c enthält ein Grundgerüst zum Generieren von "Apfelmännchen" (Bilder der Mandelbrotmenge) und soll von Ihnen um die fehlende Funktionalität erweitert werden. Die beiden anderen Dateien bilden das Modul draw, das zum Speichern der Bilder als Postscript- oder Bitmapdatei gebraucht wird.

b) Erweitern Sie mandel.c um eine Funktion "iterate", die als Argument die zu testende komplexe Zahl c und die maximale Anzahl der durchzuführenden Iterationsschritte erhält. Sie soll so lange die Berechnung $z = z^2 + c$ wiederholen, bis entweder $|z|$ das Limit überschreitet oder die maximale Iterationstiefe erreicht wird. Als Ergebnis liefert die Funktion die Anzahl der durchgeführten Iterationen.

Verwenden Sie für die nötigen Berechnungen die in Aufgabe 26 von Ihnen implementierten Operationen auf komplexen Zahlen.

c) Ein Bild besteht aus vielen Einzelpunkten (Pixel), die Punkten in der komplexen Zahlenebene entsprechen. Schreiben Sie eine Funktion, die in Abhängigkeit vom gewählten Ausschnitt der Komplexen Zahlenebene und den Dimensionen des Bildes (festgelegt durch die Variablen width und height in mandel.c) einem Pixel die ihm entsprechende komplexe Zahl zuordnet.

d) Eine weitere Funktion soll die für das Bild gewünschte Zuordung der Iterationstiefe zu Farbwerten vornehmen. In den Tafelübungen wird eine einfache Funktion vorgestellt, die auch in mandel.c teilweise vorgegeben ist. Um schönere Bilder, evtl. auch in Farbe, zu erzeugen, sind hier der Phantasie keine Grenzen gesetzt...

e) Schreiben Sie die noch fehlende Funktion compute_image (ihr Prototyp ist bereits in mandel.c vorgegeben). Sie muß für jeden Bildpunkt (Pixel) mit der Funktion draw_set_color aus dem draw-Modul einen Farbwert setzen. Um diesen zu erhalten, muß zuerst die dem Pixel zugeordnete komplexe Zahl ermittelt werden (siehe Teilaufgabe c). Dann wird für diese Zahl die Iterationstiefe bestimmt (siehe Teilaufgabe b) und der entsprechende Farbwert gewählt (siehe Teilaufgabe d).
Bem: Die Variable relevant_iter legt fest, welche Iterationstiefen für die Farbgebung interessant sind, das dies i.d.R. wesentlich weniger als die maximale Iterationstiefe sind.

f) Die Bearbeitung dieser Teilaufgabe ist bis auf die Eingabe des Dateinames freiwillig.
In der Main-Funktion sind bisher alle Werte fest kodiert und können bei der Programmausführung nicht vom Benutzer eingegeben werden. Es wäre besser, wenn der Benutzer die jeweiligen Parameter bei der Programmausfüuhrung eingeben könnte. Insbesondere soll der fest vorgegebene Dateiname "test" durch einen vom Benutzer festzulegenden Namen

Bemerkungen zur Programmieraufgabe:

Kompilieren:

- auf HP (CIP-Pool, 1. Stock):
  cc -Ae -o mandel mandel.c complex.c draw.c -lm
- auf Sun (CIP-Pool, !. Stock, Nebenraum):
  cc -Aa -o mandel mandel.c complex.c draw.c -lm
- Linux:
  In draw.c Kommentare der Zeilen 9 und 10 entfernen (INTEL_STUFF/LINUX_STUFF)
  gcc -o mandel mandel.c complex.c draw.c -lm
- Windows/NT
  In draw.c Kommentar der Zeile 9 entfernen (INTEL_STUFF)
  Projekt erstellen und Kompilieren, je nach verwendetem Kompiler...
  Leider kann ich hier nicht viel weiterhelfen...

Das Programm erzeugt eine Datei mit dem angegebenen Dateinamen und der Endung ".ps" falls draw_write_ps verwendet wird, und mit ".bmp", falls draw_write_bmp verwendet wird. Diese Dateien enthalten das berechnete Bild und können mit dem Programm xv angezeigt werden (z.B. xv test.ps). Das xv-Programm wird beendet, indem die Taste 'q' im entsprechenden Fenster gedrückt wird. Auf Windows-Rechnern kann das mitgelieferte Zeichenprogramm zum Betrachten einer Bitmapdatei (.bmp) verwendet werden.

Übungen zu "Einführung in die Programmierung für Ingenieure (C)" im WS 98/99

Übung 9

Aufgabe 30

Entwickeln Sie in den Übungen die nachfolgenden, zum Sortieren von Wörtern notwendigen Funktionen. Die zu bearbeitenden Worte sollen in einem 2-dimensionalen Feld (kein globales Feld!) **char words [MAX_STRINGS][MAX_WORDLENGTH]** abgelegt werden. Desweiteren existiere ein Feld von Zeigern **char *all_words[MAX_STRINGS]**, welches als Liste der eingelesenen Worte dienen soll. Die Anzahl der eingelesenen Worte sei in der Variablen **no_of_words** abgelegt.

a) **int string_cmp(char *s1, char *s2):**
Diese Funktion vergleicht die beiden übergebenen Zeichenketten und liefert als Ergebnis:

-1	falls die Zeichenkette s1 lexikographisch kleiner ist als s2,
0	falls beide Zeichenketten identisch sind
1	falls die Zeichenkette s1 lexikographisch größer ist als s2

b) **char *next_word(char *word):**
Diese Funktion liest das nächste Wort (= Folge von alphanumerischen Zeichen inklusive des Zeichens '-') von der Standardeingabe. Beim Aufruf soll dieser Funktion ein Zeiger auf die nächste freie Zeile des Feldes **words** übergeben werden. Ist man am Ende der Eingabe angelangt (d.h. das aktuelle Zeichen ist EOF), so liefert die Funktion den Wert NULL zurück. Andernfalls wird ein Zeiger auf das eingelesene Wort zurückgeliefert. Worte die länger als **MAX_WORDLENGTH** sind, sollen abgeschnitten werden.

c) **int insert_word(char *string, char *all_words[], int no_of_words):**
Diese Funktion fügt einen Zeiger auf ein eingelesenes Wort in die Wortliste **all_words** ein. Dabei soll zunächst überprüft werden, ob das Wort bereits in der Liste vorhanden ist. Zu diesem Zweck sollte die Funktion **string_cmp** verwendet werden. Ist dies der Fall, so soll **insert_word** den Wert 0 zurückgeben. Andernfalls soll der Wert 1 zurückgeliefert werden.

Entwickeln Sie dabei nicht nur eine Lösung als C-Programm, sondern erstellen Sie vorher jeweils ein Struktogramm.

Aufgabe 31

Führen Sie einen Trace des untenstehenden Programmes durch. Geben Sie den Inhalt der Variablen nach jeder erfolgten Änderung in einer Tabelle an!

```
int k[] = {1,0,1,0};
void f(int *k)
{
    int m=*k-2;
    do {
        m = k[1];
        *(k+1) += *k;
        k++;
    }while(m>0);
```

Einführung in die Programmierung für Ingenieure (C)
© *Jürgen Kleinöder* Universität Erlangen Nürnberg, IMMD IV, 1998

```
        (*k)++;
    }

    int main()
    {
        int *m = k;
        f(m);
        m++;
        f(m);
        return *m;
    }
```

Aufgabe 32(Abgabe bis Fr. 15.01.99)

Entwickeln Sie ein Programm, das einen Text zeichenweise von der Standardeingabe einliest. Dabei soll es, je nach den in der Kommandozeile übergebenen Argumenten, entweder den Text in Wörter zerlegen und diese alphabetisch sortiert ausgeben oder die Anzahl der Zeichen/Wörter/Zeilen des Textes zählen.

a) Jedes mögliche Argument (-s für sortieren, -c für Anzahl der Zeichen, -w für Anzahl der Wörter und -l für Anzahl der Zeilen) soll in der Main-Funktion durch eine Variable repräsentiert werden, die auf "wahr" gesetzt wird, wenn das ihr entsprechende Argument in der Kommandozeile vorhanden ist.
Diese Aufgabe soll von einer Funktion erledigt werden, d.h. sie muß ggf. die o.g. Variablen ändern können. Werden unbekannte Argumente auf der Kommandozeile übergeben, so soll die Funktion eine Fehlermeldung ausgeben und das Programm beenden (siehe "man exit"). Das gleiche soll passieren, wenn -s zusammen mit einer der anderen Optionen verwendet wird. Dagegen ist die gleichzeitige Verwendung von -c, -w und -l erlaubt. Falls das Programm ohne Argumente aufgerufen wird soll es sich so verhalten als wäre -w angegeben.

b) Die Bearbeitung des eingegebenen Textes soll bei den Argumenten -c, -w und -l (oder bei einer Kombination davon) von **einer** Funktion vorgenommen werden. Diese soll den Text zeichenweise einlesen (siehe "man getchar") und dabei mitzählen, wieviele Zeichen/Wörter/Zeilen vorhanden sind. Zur Bestimmung der Wortgrenzen können sie die vorhandenen C-Funktionen verwenden (siehe "man isalnum", "man isspace", usw.).

c) Eine andere Funktion soll die Verarbeitung des Textes erledigen, wenn die Option -s verwendet wird. In dieser Funktion müssen die in Aufgabe 30 erwähnten Felder definiert werden, die dann mit den ebenfalls vorgestellten Funktionen next_word() und insert_word() gefüllt werden können. Achten Sie hierbei darauf, daß ein Wort nur einmal im Feld abgespeichert wird. Nun soll noch in alphanumerisch aufsteigender Reihenfolge (d.h. so wie von string_cmp() in Aufgabe 30 geordnet) sortiert (siehe Teilaufgabe d) und die Wörter in der neuen Reihenfolge augegeben werden.

d) Sie benötigen zwei weitere Funktionen zum sortieren des Feldes all_words: Eine Funktion swap() zum Vertauschen von zwei Elementen des Feldes, sowie eine Funktion string_sort(), die die Feldelemente (bzw. die Strings, worauf die Elemente zeigen) vergleicht und ggf. vertauscht (man erinnere sich an den Bubble-Sort...).

e) Geben Sie zu der im Aufgabenteil a) programmierten Funktion ein Struktogramm ab.

Sie finden unter /proj/i4cing/CountAndSort/ ein Gerüst für Ihr Programm, das u.a. auch die in Aufgabe 30 vorgestellten Funktionen enthält (es ist alles auch im WWW verfügbar unter http://www4.informatik.uni-erlangen.de/Lehre/SS98/V_C/Uebung/). Dort liegt außerdem ein Text, den Sie zum Testen Ihres Programms verwenden können. Wenn das Programm z.B. "cas" heißt könnte ein Testlauf z.B. so aussehen:

```
> cas -l -c < A32.txt
Zeichen: 3115
Zeilen: 65
```

Übung 10

Aufgabe 33

Betrachten Sie die folgenden Struktur-Deklarationen/Definitionen. Was ist der Unterschied? Initialisieren Sie zur Übung einige Strukturen des Vektors member!

```
struct adress{          struct adress {         struct {
    char name[20];          char name[20];          char name[20];
    char vname[20];         char vname[20];         char vname[20];
    int age;                int age;                int age;
};                      }member[10];            }member[10];
```

Wenn ein Pointer p nun durch struct adress *p definiert wurde, wie wird dann erreicht, daß p z.B. auf member[5] zeigt, und wie kann dann über p auf die Komponenten von member[5] zugegriffen werden?

Aufgabe 34

Markieren und verbessern Sie die Fehler (max. 15 Stück!) in dem untenstehenden Programm, welches einen Text von der Standardeingabe einliest und seitenweise mit Seitenzahlen versehen auf die Standardausgabe ausgibt. Die Standardseitenlänge von 60 Zeilen kann mittels der Option -n Zeilenzahl verändert werden. (Achtung: atoi ist eine in stdlib.h korrekt deklarierte Funktion zum Umwandeln eines Strings in einen Integerwert, '\014' ein korrektes Steuerzeichen für den Seitenumbruch!)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_LINELENGTH 80;

int main(int argc, int *argv)
{
    int lines_per_page := 60; \* Default: 60 Zeilen *\
    int line = 0;
    void page = 1;
    char zeile[MAX_LINELENGTH+1];
    if((argc!=1) && (argc!= 3))
        printf("usage: print [-n lines_per_page]\n");
        return;
    }
    if((argc==3) && (strcmp(argv[1], '-n')=0)){
        lines_per_page = atoi(argv[3]);
        if(lines_per_page < 3) {
```



```
            printf("zu kleine Zeilenzahl: $d\n";
                    lines_per_page);
            return 1;
        }
    }
    while(zeile = gets(zeile)) {
        printf("%s\n",zeile);
        if(++line == lines_per_page-2) {
            printf("\n\t\t-- %d --\n\014", page++);
            line = 0;
        }
    }
    if(line>0) {
        for(; line<lines_per_page-2; line++)
            printf("\n");
        printf("\n\t\t-- %d --\n\014");
    }
}
```

Aufgabe 35(Abgabe bis Fr. 22.01.1999)

Schreiben Sie ein Programm zur Verwaltung von Studentendaten. Dabei soll das Programm für beliebig viele Datensätze geeignet sein. Aus diesem Grund sollen Sie eine verkettete Liste für die Studentendaten verwenden.

Für jeden Studenten sollen Vor- und Nachname, sowie Geburtsdatum und Matrikelnummer gespeichert werden. Das Programm soll folgende Wahlmöglichkeiten anbieten:

- Neue Studentendaten einlesen und in die Liste aufnehemen
- Durchsuchen der Liste nach einem spezifizierten Nachnamen
- Anzeigen aller Datensätze der Liste
- Löschen eines Datensatzes (bei gegebenem Nachnamen) aus der Liste
  (Diese Teilaufgabe ist optional)
- Programm beenden (optional können Sie hier den während des Programmlaufs angeforderten Speicher wieder freigeben)

Jeder dieser Vorgänge soll durch eine eigene Funktion erledigt werden. Um Ihr Programm möglichst modular zu gestalten, sollten Sie weitere Funktionen implementieren, die von den oben genannten verwendet werden können:

- Beschaffen von Speicher für einen Datensatz (siehe "man malloc")
- Einfügen eines Datensatzes in die Liste
- Ausgeben eines Datensatzes

Übung 11

Aufgabe 36

Erweitern Sie das Programm aus Aufgabe 35 so, daß die Datensätze auch in eine Datei geschrieben bzw. aus einer Datei gelesen werden können: Falls beim Programmaufruf ein Dateiname als Kommandozeilenparameter übergeben wird, soll die entsprechende Datei geöffnet und die in ihr gespeicherten Datensätze eingelesen werden. Falls während des Programmlaufs der Datenbestand geändert wird (Löschen/Einfügen von Datensätzen), soll der Benutzer beim Beenden des Programms gefragt werden, ob die Änderungen in einer Datei gesichert werden sollen.

Hinweis: Sie können die bestehenden Funktionen zum Einlesen und Ausgeben eines Studentendatensatzes durch eine kleine Modifikation auch zum Arbeiten mit Dateien verwenden:

```
printf("Zahl: %d\n", i);
```

ist äquivalent zu

```
fprintf(stdout, "Zahl: %d\n", i);
```

Aufgabe 37

In welcher Form kann ein C-Programm Parameter von der Kommandozeile abfragen und bearbeiten? Welche Bedeutung hat dabei der erste übergebene Parameter? In welcher Form werden die Kommandozeilenparameter zerlegt und dem Programm zur Verfügung gestellt?

Aufgabe 38

Schreiben Sie ein Programm **printargs**, welches alle Aufrufargumente auf die Standardausgabe ausgibt, sofern es sich dabei nicht um Aufrufoptionen handelt. Ihr Programm soll die beiden folgenden Aufrufoptionen erkennen und entsprechend reagieren:

- bei gesetzter Option -n soll am Ende der Ausgabe kein Zeilenumbruch,
- bei Option -r die Aufrufargumente in umgekehrter Reihenfolge ausgegeben werden.

Bei unbekannten Aufrufoptionen soll das Programm mit einer Fehlermeldunwg terminieren.