7. Übung

1 Auswertung Hausaufgaben
2 Tipps zur Hausaufgabenserie 03
3 Funktionen, Felder, getchar, putchar
4 Aufgaben
5 Lösungen

1. Auswertung Ha - Erwartungsbild

1. Erläutern Sie die Regeln für erlaubte Variablennamen in C. Der Name foobar ist erlaubt, der Name foo&bar dagegen nicht – warum? (3 Punkte)

Lösung:

In C sind für Variablennamen alphanumerische Zeichen (Buchstaben oder Ziffern) und „_“ erlaubt.

Jede Variable muss mit einem Buchstaben oder _ beginnen! (man sollte aber Variablen nicht mit _ beginnen, da diese Form für intern angelegte Namen genutzt wird)

Groß- und Kleinbuchstaben werden unterschieden.

Vordefinierte Namen der Programmiersprache (Schlüsselwörter) dürfen nicht verwendet werden.

foo&bar enthält mit „&“ ein nicht gültiges Zeichen

1. Auswertung Ha - Erwartungsbild

2. Könnte es sein, dass sehr lange Variablennamen in C aus bestimmten Gründen problematisch sind?
Was sagt der ANSI-Standard bzw. das Kernighan-Ritchie Buch hierzu?
(3 Punkte)

Lösung:

(auch Antworten bezüglich neuere Standards als ANSI)
Der Compiler hat eventuell Probleme für das Anlegen und Verwalten der Symboltabellen.
Laut ANSI-Standard 32 Zeichen lang,
K&R, Seite 35: 31 Zeichen signifikant für interne Namen, 6 für externe Namen (z.B. Parameter einer Funktion)

Laut ISO C 2005: 63 Zeichen intern, 31 extern
Laut ISO C 2011: There is no specific limit on the maximum length of an identifier. Implementation decided of an implementation limit.

1. Auswertung Ha - Erwartungsbild

3. Schreiben Sie ein Programm, das Beträge von Euro in US Dollar umrechnet. Der aktuelle Währungskurs ist: 1 Euro entspricht 1.10 USD
Überlegen Sie sich, mit welcher Formel Sie einen Betrag von Euro in USD umrechnen können. Ihr Programm soll solange Beträge einlesen, bis der Benutzer eine 0 eingegeben hat – dann soll das Programm beendet werden. Wenn der Benutzer einen anderen Wert eingibt, dann soll die Umrechnung in USD und die Ausgabe des Wertes erfolgen. (6 Punkte)

Ein exemplarischer Programmablauf ist:

```
Bitte geben Sie einen Betrag in Euro ein (Abbruch mit 0): 10.00
Der Betrag entspricht 11.700 USD
Bitte geben Sie einen Betrag in Euro ein (Abbruch mit 0): 20.00
Der Betrag entspricht 23.40 USD
Bitte geben Sie einen Betrag in Euro ein (Abbruch mit 0): -100000.00
Der Betrag entspricht -117000.00
Bitte geben Sie einen Betrag in Euro ein (Abbruch mit 0): 0
```

Lösung:

1. Auswertung Ha - Erwartungsbild

```
#include <stdio.h>
main() {
        float kurs=1.1;
        float euro, us;
        printf("Bitte geben Sie einen Betrag in Euro ein (Abbruch mit 0):");
        scanf("%f",&euro);
        while(euro!=0){
                us = euro*kurs;
                printf(" Der Betrag entspricht %.2f USD\n", us);
                printf("Bitte geben Sie einen Betrag in Euro ein (Abbruch mit 0):");
                scanf("%f",&euro);
        }
}
```

1. Auswertung Ha - Erwartungsbild

4. Schreiben Sie ein Programm, das eine positive ganze Zahl in das binäre Zahlensystem übersetzt. Dabei dürfen Sie die Binärstellen in umgedrehter Reihenfolge ausgeben (d.h., niederwertigste Stelle zuerst), so dass Sie etwa für die Zahl 1310 die Ausgabe 1011[2]
erhalten (in normaler Reihung wäre die Ausgabe 1101[2]). (10 Punkte)

Lösung :

```
#include <stdio.h>
main() {
          int n, n1;
          printf("Zahl? ");
          scanf("%d",&n);   /* Eingabe der Dezimalzahl */
          printf("%d[10] = ",n); /* Ausgabe der Dezimalzahl */
          while(n > 0) { /* Solange Konvertierung nicht beendet */
            n1 = n/2;
            printf("%d",n - 2*n1); /* Ausgabe der binaeren Ziffer */
            n = n1;
       }
       printf("[2]\n");
}
```

1. Auswertung Ha - Erwartungsbild

Alternative Berechnung mit % - Operator

```
#include <stdio.h>
main() {
        int n;
        printf("Zahl? ");
        scanf("%d",&n);
        while(n>0) {
                printf("%1d",n%2);
                n=n/2;
        }
        printf("[2]\n");
}
```

---

oder kürzer mit for-Schleife:

```
#include <stdio.h>
        main() {
        int n, i;
        printf("Zahl? ");
        scanf("%d",&n);
        for(i=n;i>0;i=i/2)  /* moeglich auch for(;n>0;n=n/2) und dann n%2 */
                printf("%1d",i%2);
        printf("[2]\n");
}
```

1. Auswertung Ha – Erwartungsbild *

Beispiel - Korrekte Reihenfolge der binären Ziffern:

```
#include <stdio.h>
main() {
        int n, j, p;
        printf("Zahl? ");
        scanf("%d",&n);
        printf("\n");
        for (j = 0, p=1; j <= sizeof n; j++) /* hoechste Potenz bestimmen 2 hoch 31*/
                p = p * 64;

        while (p>0) { /* solange nicht alle Binaerziffern konvertiert */
                if (n / p > 0) { /* Division durch aktuelle Zweipotenz ist groesser */
                        printf("1");
                        n = n - p; /* subtrahiere diese Potenz von n */
                }
                else printf("0");
                p = p/2; /* teste naechste kleinere Zweierpotenz */
        }
        printf("[2]\n");

}
```

1. Auswertung Ha - Erwartungsbild

5. Betrachten Sie das folgende Programm:

```
#include <stdio.h>
main() {
        int w, x, y, z;
        printf("W? ");
        scanf("%d",&w);
        printf("X? ");
        scanf("%d",&x);
        y=1;
        while(y<=x) y=y*w;
        y=y/w;
        while(y > 0) {
                z = x/y;
                printf("%d",z);
                x = x - y*z;
        y = y/w;
        }
        printf("\n");
}
```

Erläutern Sie, was das Programm macht. (7 Punkte)
Erläutern Sie, wie es das Programm macht – d. h., welchen Algorithmus das Programm verwendet, um seine Aufgabe zu erfüllen. (8 Punkte)

1. Auswertung Ha - Erwartungsbild

Lösung:

a) **Das Programm wandelt eine im Dezimalsystem angegebene Zahl X in eine Zahl zur Basis W um. Die Stellen werden dabei in korrekter Reihenfolge ausgegeben (höherwertig zu niederwertig).**

b) Kommentare

```
#include <stdio.h>
main() {
        int w, x, y, z;
        /* Einlesen der der Zielbasis w */
        printf("W? ");
        scanf("%d",&w);
         /* Einlesen der Zahl im Dezimalsystem */
         printf("X? ");
        scanf("%d",&x);

         /* Bestimmung der größten Zahl y=w^n mit y <= x  */
        y=1;
        while(y<=x) y=y*w; /* bei Ende der Schleife gilt: y >= x */
        y=y/w; /* Jetzt: y größte Zahl w^n mit y <= x */

        /* ab jetzt jetzt gilt: (z = x / y) in {1..w-1} wobei "/" ganzzahlige Division */
        while(y > 0) {
                z = x/y; /* nächste Ziffer berechnen und ausgeben */
                printf("%d",z);
                x = x - y*z; /* zu konvertierenden Rest berechnen */
                y = y/w; /* nächsten Teiler berechnen */
         }
        printf("\n");
}
```

1. Auswertung Ha - Erwartungsbild

6. In Aufgabe 6 von Aufgabenblatt A01 sollten Sie ein Struktogramm für einen Algorithmus zur Berechnung der Sinusfunktion auf Basis einer Reihenentwicklung angeben. Nun ist Ihre Aufgabe, diesen Algorithmus zu implementieren. Schreiben Sie ein Programm, das einen Winkelwert von der Standardeingabe in eine double-Variable einliest, den zugehörigen Sinuswert als double-Zahl mit Hilfe Ihres Algorithmus berechnet, und anschließend ausgibt.

Anmerkungen:

- Die Reihenentwicklung erwartet, dass der Winkel in Bogenmaß angegeben wird.
- Für das Einlesen eines Wertes in eine double-Variable x können sie den Aufruf scanf("%lf",&x); verwenden.
- Wenn Sie ihren Algorithmus geschickt formulieren, benötigen Sie für die Reihenentwicklung lediglich eine einzige Schleife.

10 Punkte

1. Auswertung Ha - Erwartungsbild

```
#include <stdio.h>
#include <math.h>
int main(void) {
  double x; /* originale Eingabe */
  double x2; /* aequivalenter Winkel da periodische Funktion */
  double rg; /* aktuelles Reihenglied */
  double sinx = 0;
  int n2 = 0;
  printf("x Angabe in Grad? "); /* laut Aufgabe nicht klar, ob Winkel in Bogenmass eingegeben
wird oder in Gradmass, hier Annahme Grad */
  scanf("%lg",&x); /* Vorgabe aus Aufgabenblatt */
  x2 = x;
  /* Diese Optimierung zur besseren Konvergenz */
  while(x2>180) /* Bessere Konvergenz im Intervall -180 bis +180 bei zu grossen Winkel */
            x2=x2-360;
  while(x2>180) /* Bessere Konvergenz im Intervall -180 bis +180 bei negativen Winkel */
            x2=x2+360;
  x2 = x2 *2*M_PI/360;/* Umwandlung Winkel in Bogenmass */
  /* Ab hier Berechnung des Sinus aus einem Winkel in Bogenmass */
  rg = x2; /* Startwert - erste Reihenglied */
  x2 = -x2*x2; /* Bildung von - x hoch 2 */
  while((rg>0.00001)||(rg<-0.00001)){ /* Solange nicht Genauigkeit erreicht */
    sinx = sinx + rg; /* Aktualisierung der Reihensumme */
    n2 = n2 + 2; /* Aktualisierung fuer Fakultaetsberechnung */
    rg = rg * (x2 / (n2 * (n2+1))); /* Aktualisierung Reihenglied -x^2/(2n*2n-1) */
  }
  /* Ausgabe des Wertes, der Vergleich zur Sinus-Funktion ist nur zur opitischen Kontrolle*/
 printf("msin(%lg) = %lg  Vergleich sin-Funktion, sin(%lg) = %lg\n", x, sinx,
x,sin(x*2*M_PI/360));
}
```

1. Auswertung Ha - Erwartungsbild

```
#include <stdio.h>
#include <math.h>
int main(void) {
  double x; /* originale Eingabe */
  double x2; /* aequivalenter Winkel da periodische Funktion */
  double rg; /* aktuelles Reihenglied */
  double sinx = 0;
  int n2 = 0;
  /* Naive fehlerhafte Berechnung durch formale Umsetzung Summenformel, */
  printf("x Bogenmass? ");
  scanf("%lg",&x); /* Eingabe x */
  int i, j, n, minus1;
  double xn, fakn=1;
  sinx = x;
  n = 100; /* grosses n - gewaehlt */
          for(i=1;i < n ; i++){
                    /* Berechnung x hoch n  und 2n +1- Fakultaet */
                    for(j = 1, xn=x, fakn=1, minus1=1; j < i ; j++) {
                              xn = xn*x;
                              fakn = fakn*2*n*(2*n+1);
                              minus1=minus1*(-1);
                    }
                    sinx = sinx +minus1*xn/fakn;
          }
  printf("mysin_naiv(%lg) = %lg (sehr ungenau, da Fakultaet ungenau )\n", x,sin );
  return 0;
}
}
```

2. Aufgabenblatt 3

`Fragen ?`

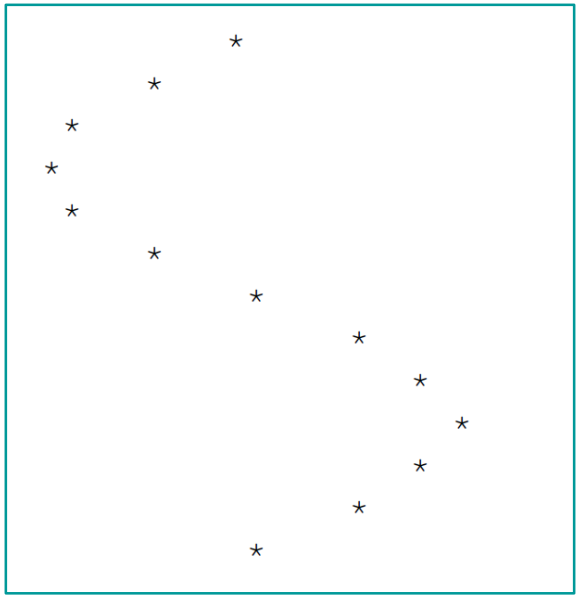
2. Ideen zu Hausaufgabenserie 3

Ausgabe Kurven

Vorlesung S 61

```
# include <stdio .h>
# include <math .h> /* for sin(x) */
main () {
    double x;
    int indent ;
    for (x = -180.0;x <=180.0; x +=30.0) {
    /* compute value */
        indent = 10 + 10* sin(x /180.0* M_PI );
    /* plot star at position */
    for (; indent ;-- indent ) putchar (' ');
        printf ("*\n");
    }
}
```

```
/* Test auf Anforderung Übersetzung */
/* Jetzt 2 Kurven Malen */
```



2. Ideen zu Hausaufgabenserie 3

Ausgabe Kurven (korrekte Uebersetzung)

Vorlesung S 61

```
# include <stdio .h>
# include <math .h> /* for sin(x) */
int main () {
     double x;
     int indent ;
     for (x = -180.0;x <=180.0; x +=30.0) {
     /* compute value */
          indent = 10 + 10* sin(x /180.0* M_PI );
     /* plot star at position */
     for (; indent ;-- indent ) putchar (' ');
          printf ("*\n");
     }
     return(0);
}
/* Test auf Anforderung Übersetzung */
Seite61.c:3:1: warning: return type defaults to 'int' [-Wreturn-type]
 main () {
 ^
seite61.c: In function 'main':
seite61.c:13:1: warning: control reaches end of non-void function [-Wreturn-
     type]
 }
```

```
            *
        *
   *
 *
   *
        *
            *
                *
                    *
                        *
                    *
                *
            *
```

2. Ideen zu Hausaufgabenserie 3

Ausgabe 2 Kurven gleichzeitig malen in Hausaufgabe

```
# include
# include
main () {
    double
    int in
    for (x
    /* com
        in
    /* plo
    for (;
        pr
    }
}
/* Jetzt 2 Kurven Malen */

/* vor der Ausgabe der Kurve Vergleich der Funktionswerte */
```

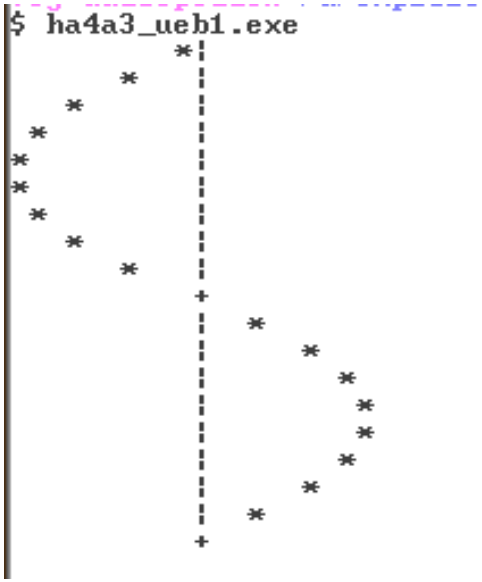
```
$ ha4a3.exe
```

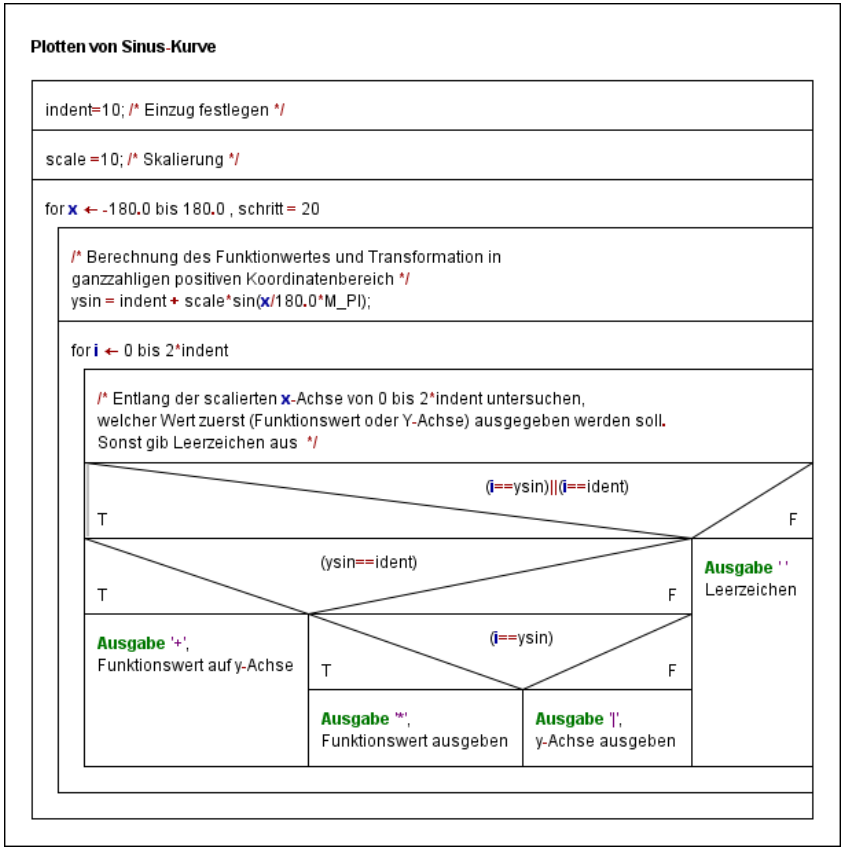
2. Ideen zu Übungsaufgabe heute

Ausgabe der Sinus-Kurve mit y -Achse

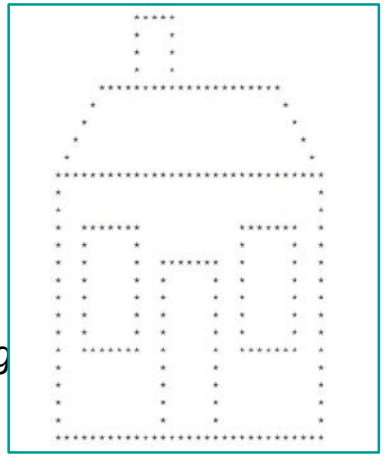
Sinus-Kurve mit Nullachse zeichnen

`Einzug (indent)  und Skalierung(scale aus -1.0..1.0 in ganzzahligen Bereich)`





2. Ideen zu Hausaufgabenserie 3



3. Wir basteln uns ein Grafikprogramm:
(a) Schreiben sie eine Funktion

```
int contained(double x, double a, double b);
```

die eine 1 liefert, falls x in dem geschlossenen Intervall [a; b] liegt und sonst 0.
(Die Intervallgrenzen müssen nicht notwendigerweise in der richtigen Reihenfolge ang
sein.) (3 Punkte)
(b) Schreiben Sie eine Funktion

```
int line(double x, double y, double px, double py, double qx, double qy);
```

die eine 1 liefert, falls der Abstand zwischen dem Punkt X = (x; y) und der Strecke PQ kleiner als 0.5 ist und sonst 0. (Hierbei gilt: P = (px; py);Q = (qx; qy)).
Hinweis: Sei ax + by + c = 0 die Koordinatenform der Geraden, die durch die Punkte P;Q definiert ist. Dann ist d = ax0 + by0 + c der Abstand eines Punktes (x0; y0) von dieser Geraden. (8 Punkte)
(c) Sei ein Rechteck durch seine linke untere Ecke (lx; uy) und seine rechte obere Ecke (rx;oy) definiert. Schreiben Sie eine Funktion

```
int rectangle(double x, double y, double lx, double uy, double rx, double oy);
```

die eine 1 liefert, falls der Abstand zwischen dem Punkt X = (x; y) und einer der vier Seiten eines so definierten Rechtecks kleiner 0.5 ist und sonst 0. (5 Punkte)
(d) Nutzen Sie die Funktionen um ein Programm zu schreiben, das folgende Ausgabe erzeugt:
Hinweis: die Schwierigkeit hier besteht darin, sich zu überlegen, wie das Ergebnis von Funktionen wie line oder rectangle genutzt werden kann, um Ausgabezeichen an der richtigen Stelle zu erzeugen.

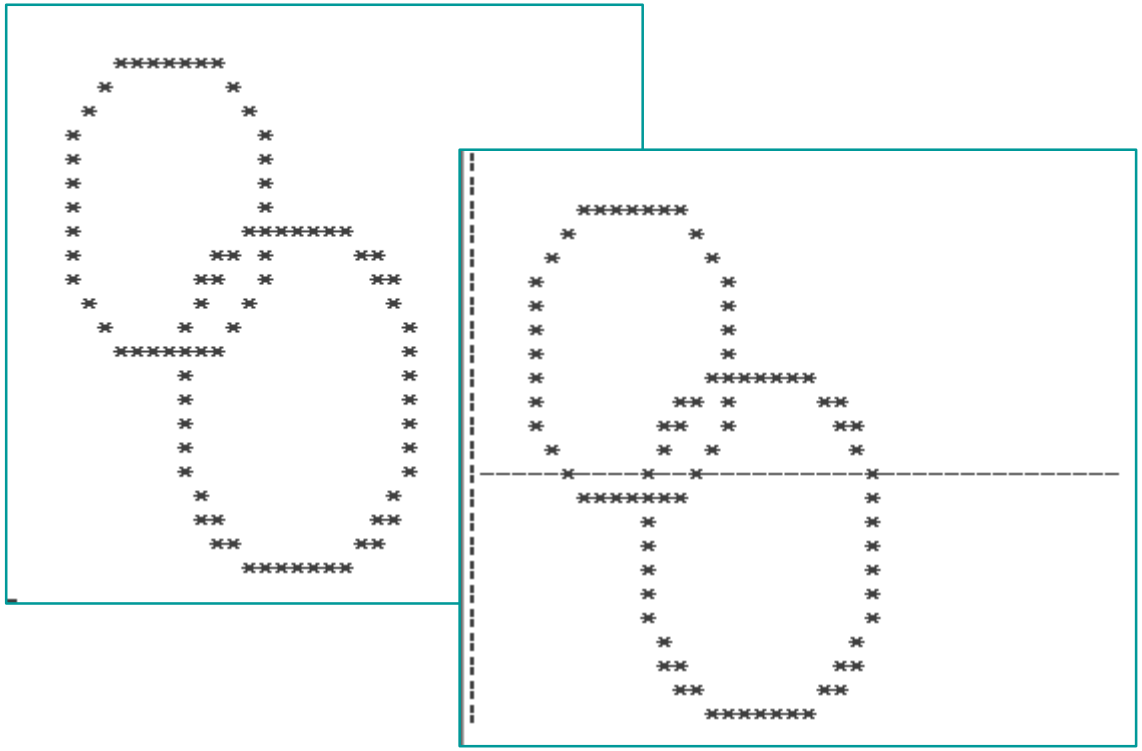
2. Ideen zu Übungsaufgabe heute

3. Wir basteln uns ein Grafikprogramm:
(a) Schreiben sie eine Funktion

```
int kreis(double x, double y, double mx, double my, double radius);
```

die eine 1 liefert, falls der Abstand zwischen dem Punkt X = (x; y) und dem Kreisbogen mit gegebenen Radius und Mittelpunkt kleiner als 0.5 ist und sonst 0.

Für einen Kreis (MP(10,5), Radius 6) und einen Kreis (MP(18,-3), Radius 7) könnte die Darstellung im Bereich von
0<= x <= 40
-10 <=y <= 24
So ähnlich wie abgebildet aussehen.



2. Tipps zu Hausaufgabenserie 3

(a) Schreiben sie eine Funktion

```
int contained(double x, double a, double b);
```

die eine 1 liefert, falls x in dem geschlossenen Intervall [a; b] liegt und sonst 0.
*(**Die Intervallgrenzen müssen nicht notwendigerweise in der richtigen Reihenfolge angegeben sein**.) (3 Punkte)*
(b) Schreiben Sie eine Funktion

```
int line(double x, double y, double px, double py, double qx, double qy);
```

die eine 1 liefert, falls der Abstand zwischen dem Punkt X = (x; y) und der Strecke PQ kleiner als 0.5 ist und sonst 0. (Hierbei gilt: P = (px; py);Q = (qx; qy)).
Hinweis: Sei ax + by + c = 0 die Koordinatenform der Geraden, die durch die Punkte P;Q definiert ist. Dann ist d = ax + by + c der Abstand eines Punktes (x; y) von dieser Geraden.
(5 Punkte)
Koordinatenform der Geradengleichung ay + bx + c = d (Durch Einsetzen von x und y ergeben sich für d Werte !=0, die ein Vielfaches des Vektors (a,b) ergeben, der senkrecht auf der Geraden steht. Das Vorzeichen von d gibt die Seite der Geraden an)

nx = (py – qy);
ny = (qx – px);
*l = sqrt(nx*nx+ny*ny);*
a = nx/l; b= ny/l;
*c = -(a*px +b*py);*

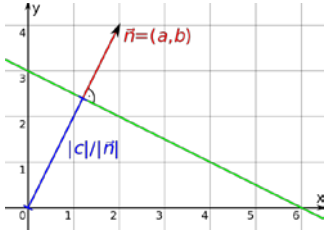


Bild:
Quelle: Wikipedia
„Koordinatenform“

(c)
```
int rectangle(double x, double y, double lx, double uy, double rx, double oy);
```

Ein Rechteck besteht aus 4 Linien!

3. Funktionen

Blöcke mit Namen können in c als Funktionen aufgefasst werden.

Sie können über den Funktionsnamen Werte liefern.

Es können Parameter an den Block übergeben werden.

Beispiel Berechnung Schuhgröße alt:

```
int fusslaenge;
schuhgroesse_de() {
            float schuhgroesse = (int)((fusslaenge + 1.54)/0.667);
            printf("Die Schuhgroesse DE %.0f, bei Fusslaenge von %d cm\n", schuhgroesse, fusslaenge);
}


main() { /* Eingaben und Ausgaben siehe Folie 9 Vorlesung */
            printf("Eingabe der Fusslaenge in cm\n");
            scanf("%d",&fusslaenge);
            schuhgroesse_de();
}
```

Neu :

```
float schuhgroesse_de( int fusslaenge ) { /* Function mit Parameter und Rückgabedatentyp */
            float schuhgroesse = (int)((fusslaenge + 1.54)/0.667);
            return schuhgroesse; /* Rueckgabe der schuhgroesse auf dem Funktionsnamen */
}

int main() {
            int fussl;
            float groesse;
            printf("Eingabe der Fusslaenge in cm\n");
            scanf("%d",&fussl);
            printf("Schuhgroesse %.0f bei Fusslaenge %d\n", schuhgroesse_de(fussl), fussl);
            groesse = schuhgroesse_de(fussl);
            return(0); /* Auch main ist eine Funktion */
}
```

3. Funktionen

Nach Variablen, Schleifen und Ausgaben jetzt ein neues Konzept:

Funktionen

Teile von Algorithmen extra beschreiben (außerhalb von main)!

Vorteile:

- Besser Lesbarkeit
- Wiederverwendbarkeit
- Leichter Wartbarkeit

Generelle Syntax:

```
Rückgabetyp Funktionsname(Parameterliste) { Anweisungen }
```

3. Funktionen

Umrechnung Gradmaß in Bogenmaß:

```
#include <stdio.h>
#include <math.h>

int main() {
    int winkel;
    double xrad;
    for ( winkel = -360; winkel <= 360 ; winkel += 15) {
        xrad = winkel /180.0 * M_PI; /* Umrechnung in Gradmass */
        printf("%10d | %15.5f\n",winkel, xrad); /* Ausgabe */
    }
}
```

Auslagerung der Umrechnung in eine Funktion „bogen“

Rückgabetyp – **double** , da das Bogenmaß als double-Wert benötigt wird,
Funktionsname – **bogen**,
Ein Eingabeparameter vom Typ **int**, um Gradmaß anzugeben, wir bezeichnen mit ihn mit **g**

```
double bogen(int g) {
   double y;    /* lokale Variable */
   y = g/180.0 * M_PI; /* Berechnung des Winkels in Bogenmass */
   return y; /* Rueckgabewert liefern */
}
```

3. Funktionen

Neues Programm unter Beachtung auch main ist Funktion :

```
#include <stdio.h>
#include <math.h>

double bogen(int g) {
    double y;
    y = g/180.0 * M_PI;
    return y;
}

int main() {
    int winkel;
    double xrad;
    for ( winkel = -360; winkel <= 360 ; winkel += 15) {
        xrad = bogen(winkel); /* Umrechnung in Gradmass */
        printf("%10d | %15.5f\n",winkel, xrad);
    }
    return 0;
}
```

Auslagerung der Umrechnung in eine Funktion „bogen“

Bei jedem Aufruf von ***bogen*** wird der Wert von ***winkel*** in die Variable ***g*** kopiert. Das Ergebnis der Funktion wird über **return** zurückgeliefert und in xrad gespeichert.

3. Funktionen

Neues Programm kürzer, da y in Funktion bogen nicht notwendig:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
/*double bogen(int g) {
    double y;
    y = g/180.0 * M_PI;
    return y;
}*/
double bogen(int g) {
    return g/180.0 * M_PI;
}

int main() {
    int winkel;
    double xrad;
    for ( winkel = -360; winkel <= 360 ; winkel += 15) {
        xrad = bogen(winkel); // Umrechnung in Gradmass
        printf("%10d | %15.5f\n",winkel, xrad); // Ausgabe
    }
    return EXIT_SUCCESS;
}
```

3. Funktionen

Beachte Funktionsprototyp muss bei Verwendung bekannt sein!
Probleme, wenn Funktion nach main definiert wird:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/* Definition von Funktionsprototypen */
double bogen(int g);

int main() {
    int winkel;
    double xrad;
    for ( winkel = -360; winkel <= 360 ; winkel += 15) {
        xrad = bogen(winkel); /* Umrechnung in Gradmass */
        printf("%10d | %15.5f\n",winkel, xrad); /* Ausgabe */
    }
    return EXIT_SUCCESS;
}
/* Eigentliche Funktion erst nach Verwendung definiert ! */
double bogen(int g) {
    return g/180.0 * M_PI;
}
```

3. Variable - Felder

Neben einfachen Variablen können wir in C Felder verwenden!
Unter einer Variablenbezeichnung mehrere Speicherplätze verfügbar!

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

 int a;
 double ywerte[100];

double bogen(int g); {
    return g/180.0 * M_PI;
}
int main() {
    int winkel, i, n;
    double xwerte[100];
    n = 0;
    for ( winkel = -360; winkel <= 360 ; winkel += 15) {
         xwerte[n] = winkel;
         ywerte[n++] = bogen( xwerte[i] ); /* Umrechnung in Gradmass */
    }
    for (i =0; i< n, i++) {
         printf("%10d | %15.5f\n",xwerte[i], ywerte[i] ); /* Ausgabe */
    }
    return EXIT_SUCCESS;
}
```

3. getchar, putchar

In der Vorlesung sind neben printf und scanf die Funktionen zum Lesen und schreiben einzelner Zeichen behandelt worden.
Da mit getchar auch Dateien gelesen werden können, wo als Endezeichen der Datei (EOF – End-of-File, welches -1 entspricht) ist der Datentyp häufig int, nicht char!
Mit putchar können einzelne Zeichen geschrieben werden.

```
int main() {
    int c;
    c = getchar(); /* Lesen eines einzelnen Zeichens */
    …

    putchar('a'); /* Schreiben eines einzelnen Zeichens */
    c = 'b';
    putchar(c); /* Schreiben eines einzelnen Zeichens */


    return EXIT_SUCCESS;
}
```

3. getchar, putchar

In Abwandlung von Folie 55, wollen wir ein Programm schreiben, welches Buchstaben und Trennzeichen in einem eingegebenen Text zaehlt.
Folie 55:

```
#include <stdio.h>
#define IN 1 /* inside a word */
#define OUT 0 /* outside a word */

/* count lines, words, and characters in input */
main()
{
  int c, nl, nw, nc, state;

  state = OUT;
  nl = nw = nc = 0;
  while ((c = getchar()) != EOF) {
    ++nc;
    if (c == '\n')
      ++nl;
    if (c == ' ' || c == '\n' || c == '\t')
      state = OUT;
    else if (state == OUT) {
      state = IN;
      ++nw;
    }
  }
  printf("%d lines, %d words, %d characters\n", nl, nw, nc);
}
```

3. getchar, putchar

In Abwandlung von Folie 55, wollen wir ein Programm schreiben, welches Buchstaben und Trennzeichen in einem eingegebenen Text zaehlt.
Folie 55:

```
#include <stdio.h>
#define IN 1 /* inside a word */
#define OUT 0 /* outside a word */

/* count lines, words, and characters in input */
main()
{
  int c, nl, nw, nc, state;

  state = OUT;
  nl = nw = nc = 0;
  while ((c = getchar()) != EOF) {
    ++nc;
    if (c == '\n')
      ++nl;
    if (c == ' ' || c == '\n' || c == '\t')
      state = OUT;
    else if (state == OUT) {
      state = IN;
      ++nw;
    }
  }
  printf("%d lines, %d words, %d characters\n", nl, nw, nc);
}
```

Was können wir nutzen:
While-Schleife

Was müssen wir neu modellieren:
Speicher für 26 Buchstaben und
ein Feld für Sonderzeichen

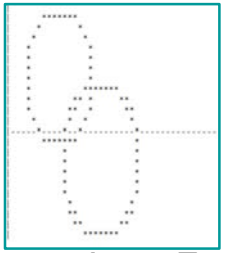
```
int alphabet[27];
```

Kleinbuchstaben liegen zwischen den Dezimalkodierungen 97 – a und 122-z

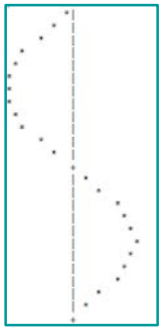
3. getchar, putchar

In Abwandlung von Folie 55, wollen wir ein Programm schreiben, welches Buchstaben und Trennzeichen in einem eingegebenen Text zaehlt.

```
int main() {
     int c,i;
     int alphabet[27]; /*26 Buchstaben und ein Trennzeichen */
     for(i=0; i<=26;i++)
        alphabet[i]=0;
     while((c = getchar())!= EOF) { /* Lesen eines einzelnen Zeichens bis EOF*/
        if((c>=97)&&(c<=122)) /* Kleinbuchstaben */
         alphabet[c-97]++;
         else
           alphabet[26]++;
     }
     for(i=0; i<26;i++)
        printf("Der Buchstabe %c kam %d mal vor\n", 97+1, alphabet[i] );
     printf("Es waren %d sonstige Zeichen\n", alphabet[26] );
}
```



4. Aufgaben für Übung



1. Zeichne die Sinusfunktion mit Nullachse.
   - Berechne die Ganzzahlposition in einer Funktion berechne, die den sin aus math.h nutzt.

   int **berechne**(double x, int offset, int scale) (z.B. offset=scale=10)
   - Gib anschließend * (Funktionswert) |(Koordinatenachse) + (Funktionswert auf Koordinatenachse oder Leerzeichen in den zweifachen offset-Zeichen aus.
2. Erstelle eine Grafikprogramm für Kreise
   - Erstelle eine Funktion ***int kreis(double x, double y, double mx, double my, double radius);***,die entscheidet, ob der Punkt auf den Kreisbogen mit dem Mittelpunkt (mx,my) und dem Radius liegt. Rückgabewert 1, sonst 0.
   - Erstelle eine Funktion ***int isKoordinatenachse(double x, double y);***,die entscheidet, ob der Punkt auf der Koordinatenachse liegt.
   - Geben Sie im Hauptprogramm in 2 Schleifen die Y-Koordinaten rückwärts und die x-Koordinaten vorwärts aus und prüfen, ob Information vorliegt oder nicht (* malen oder Leerzeichen).
3. Erweitern Sie das Programm zum zählen der Buchstaben, um auch Grossbuchstaben neben den Kleinbuchaben in den gleichen Feldern zu zaehlen. Groß- wie Kleinbuchstaben

   (65 – A,... 90 –Z, 97 a ...122 z)