9. Übung

1 Auswertung Hausaufgaben
2 Tipps zur Hausaufgabenserie 4
3 Cäsarkodierung
4 Felder, Referenzen und Adressen
5 Übungen
6 Lösungen

1. Auswertung Hausaufgabeserie 3

1. In der Vorlesung haben Sie gelernt, dass ein C Programm durch Aufrufen der Funktion main gestartet wird und dass der Rückgabewert dieser Funktion vom Betriebssystem als Ergebnisstatus verwendet wird.
Wie können Sie auf den Laborrechnern in der Kommandozeile auf den Ergebnisstatus eines Programms zugreifen? Wie geht das bei Ihrem eigenen Rechner? (5 Punkte)

Lösung:

In der Mingw-Shell kann man über **$?** auf den letzten Rückkehrcode zugreifen.
Also **echo $?** in mingw und auf unix-basierenden Shells

In Windows cmd-Fenstern kann **echo %errorlevel%** geprüft werden, im Windows Powershell ist es die ExitCode Eigenschaft des Process Objektes (**echo $lastexitcode**).

(Bemerkung: In der stdlib.h sind die beiden Makros EXIT_SUCCESS und EXIT_FAILURE mit 0 und 1 vordefiniert!
Versucht man in return(nr) eine int-zahl zu setzen, so ist bei negativen Werten das Ergebnis 0 und bei positiven Werten die Zahl nr modulo 256)

1. Auswertung Hausaufgabeserie 3

Erweitern Sie das Plotprogramm sin0.c (Skript Seite 61) so, dass es zwei Kurven gleichzeitig zeichnen kann. Dabei soll die eine Kurve durch x und die andere Kurve durch + dargestellt werden. Falls beide Zeichen auf der gleichen Position landen, soll statt dessen das Zeichen * ausgegeben werden.
Für das gleichzeitige Plotten von Cosinus (x) und Sinus (+) im Intervall −180° bis 180° bei einer Schrittweite von 15° sollte sich dann folgende Ausgabe ergeben:

(10 Punkte)

```
x         +
x      +
 x   +
   *
 +   x
+      x
+         x
+            x
 +             x
   +             x
     +             x
       +            x
          +         x
             +      x
               +   x
                 *
               x   +
             x      +
          x         +
       x            +
     x              +
   x              +
 x              +
x            +
x         +
```

Lösung: (Idee - Berechnung uns Ausgabe der Anzahl der Leerzeichen trennen)

1. Auswertung Hausaufgabeserie 3

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


int main () {
            double x;
            int ysin, ycos, i;
            for(x = -180.0;x <=180.0; x +=15.0) {
                        /* Berechnung der Anzahl der Leerzeichen fuer jeden Funktionswert */
                        ysin=sin(x/180.0*M_PI) *10 + 10;  /* Sinnvolle Werte fuer 0-Linie und Skalierung */
                        ycos=cos(x/180.0*M_PI) *10 + 10;
                        /* Ausgabe einer Zeile; fuer jede Stelle wird Entscheidung getroffen*/
                        for(i = 0; i <=20; i++){
                                    if(i==ysin && i==ycos) /*  Test auf Sonderbehandlung gleiche Werte */
                                                putchar('*');
                                     else if( i== ysin) /*  Test auf sin */
                                                            putchar('+');
                                                            else if( i== ycos) /*  Test auf cos */
                                                                         putchar('x');
                                                                        else
                                                                         putchar(' '); /* Leerzeichen */

                        }
                        /* Ausgabe einer  Newline am Ende der Zeile */
                        printf("\n"); /* oder putchar('\n'); */
            }
            return 0;
}
```

1. Auswertung Hausaufgabeserie 3

Alternative Lösung:

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

/* Angepasste Funktion aus Vorlesung nach Ausgabe keine automatisches newline */
void star (int indent , char chr) {
            /* Ausgabe einer Anzahl von Leerzeichen ohne Newline */
            for (; indent ;--indent )
                        putchar (' ');
            putchar(chr);
}

int main () {
            double x;
            int ysin, ycos;
            int offset, scale;
            offset=scale=10; /* Sinnvolle Werte fuer Verschiebung 0-Linie und Skalierung */
            for(x = -180.0;x <=180.0; x +=15.0) { /* Steuerung der Schrittweite */
                        /* Berechnung der Anzahl der Leerzeichen fuer jeden Funktionswert */
                        ysin=sin(x/180.0*M_PI) *scale + offset;
                        ycos=cos(x/180.0*M_PI) *scale + offset;
                        /* richtige Stelle fuer Newline finden */
                        if(ysin==ycos) { /* Test auf gleiche Werte sin == cos */
                                    star(ysin,'*'); printf("\n");}
                                    else if( ysin < ycos) { /* Erst sin dann cos  */
                                        /* beruecksichtigen der Differenz zwischen den Funktionswerten*/
                                                star(ysin,'+'); star(ycos-ysin,'x'); printf("\n");}
                                                else { /* Erst cos dann sin */
                                                star(ycos,'x'); star(ysin-ycos,'+'); printf("\n");}
            }
            return 0;
}
```

1. Auswertung Hausaufgabeserie 3

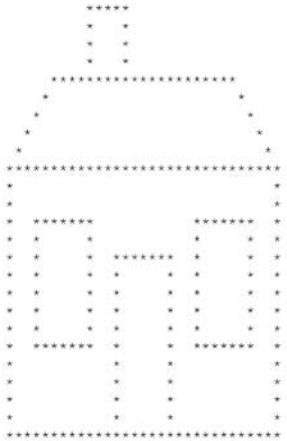
3. Wir basteln uns ein Grafikprogramm:
(a) Schreiben sie eine Funktion **`int contained(double x, double a, double b);`**
die eine 1 liefert, falls x in dem geschlossenen Intervall [a; b] liegt und sonst 0. (Die Intervallgrenzen müssen nicht notwendigerweise in der richtigen Reihenfolge angegeben sein.) (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 $\overline{PQ}$ 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 defi nierten Rechtecks kleiner 0.5 ist und sonst 0. (5 Punkte)
(d) Nutzen Sie die Funktionen um ein Programm zu schreiben, das folgende Ausgabe erzeugt:
(15 Punkte)



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.

1. Auswertung Hausaufgabeserie 3

```
#include <stdio.h>
#include <math.h>
/* Musterloesung Prof Kirste */

/* Beachtung beider Faelle */
int contained(double x, double a, double b) {
  return a < b ? (x >= a && x <= b) :  (x >= b && x <= a);
}

/* Bestimmung von Parameter a, b, c, d der Koordinatenform der Geradengleichung, Normierung des
Abstandes um sinnvolles d zu haben */
int line(double x, double y, double px, double py, double qx, double qy) {
  double nx, ny, l, a, b, c, d;

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

  return sqrt(d*d) < 0.5 && contained(x,px,qx) && contained(y,py,qy);
}
```

1. Auswertung Hausaufgabeserie 3

```
/* jede Seite des Rechtecks ist Strecke */
int rectangle(double x, double y, double lx, double uy, double rx, double oy) {
  return line(x,y,lx,uy,lx,oy)
    ||   line(x,y,rx,uy,rx,oy)
    ||   line(x,y,lx,uy,rx,uy)
    ||   line(x,y,lx,oy,rx,oy);
}
/* Bildaufbau hier aus 5 Rechtecken und 3 Linien */
int figure(double x, double y) {
  return rectangle(x,y,5,0,35,15) /* body of house */
    ||   rectangle(x,y,17,0,23,10) /* door */
    ||   rectangle(x,y,8,5,14,12)  /* left window */
    ||   rectangle(x,y,26,5,32,12)  /* left window */
    ||   line(x,y,5,15,10,20)  /* roof left */
    ||   line(x,y,35,15,30,20) /* roof right */
    ||   line(x,y,10,20,30,20) /* roof top */
    ||   rectangle(x,y,14,20,18,24); /* chimney */
}
int main() {
  double xx;
  double yy;
/* Ausgabe des Bildes (Entscheidung fuer jede Position in jeder Zeile ) */
  for(yy=24;yy>=0;yy-=1) {
    for(xx=0;xx<=40;xx+=1)
      putchar(figure(xx,yy)?'*':' ');
    putchar('\n');
  }
  return 0;
}
```

1. Auswertung Hausaufgabeserie 3

Alternative Lösung :

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

/* Beachtung beider Faelle */
int contained(double x, double a, double b){
          double h;
          if(a > b ) { / * tauschen  von a und b */
                    h = a;
                    a = b;
                    b = h;
          }
          /* Test funktioniert fuer so fuer a <= b */
          if( x >= a && x <= b)
                    return 1;
          else
                    return 0;
}
```

1. Auswertung Hausaufgabeserie 3

```
/* Bestimmung von Parameter a, b, c, d der Koordinatenform der Geradengleichung, Normierung des
Abstandes um sinnvolles d zu haben */
int line(double x, double y, double px, double py, double qx, double qy){
           /* liefert  1, falls der Abstand des Punktes zur Strecke PQ kleiner als 0.5, sonst 0
           Koordinatenform der Geradengleichung ay + bx = c
                       a = py - qy;
                       b = qx - px;
                       c = qx*py - px*qy;
                       d = ay + bx - c
         da Abstand d <.05 ist dann noch Normierung bezueglich des Normalenvektors(a,b) notwendig */
           double a, b, c, fx, l;
           a = py - qy;
           b = qx - px;
           l = sqrt(a*a+b*b); /* Laenge zur Normierung */
           a = a/l;
           b = b/l;
           c = b*py + a*px;
           fx = a*x + b*y - c;
           fx = sqrt(fx*fx);
           /* Abstand kleiner als 0.5 und im Klipprechteck von den Punkten P und Q */
           if((fx<0.5)&&contained(x,px,qx)&&contained(y,py,qy))
                      return 1; else return 0;

}
```

1. Auswertung Hausaufgabeserie 3

```
/* Auch Loesung ohne line ist moeglich, Problem dort Abstand von 0.5 pruefen, hier einfach
Abschneiden auf int-Grenzen  */
int rectangle1(double x, double y, double lx, double ly, double rx, double ry){
          if((( x == lx )||( x == rx))&&(contained(y, ly, ry)) ||((( y == ly) ||
                    (y == ry))&&(contained(x, lx, rx))))
                              return 1;
          else return 0;
}
*/
int rectangle(double x, double y, double lx, double ly, double rx, double ry){
          return (line(x,y, lx, ly, lx, ry)||line(x,y, rx, ly, rx, ry)||line(x,y, lx, ly, rx, ly)
                    ||line(x,y, lx, ry, rx, ry));
}
int imBild(double x, double y){
          /* Koordinatenursprung links unten */
          if (rectangle(x, y, 0, 0, 30, 15)) return 1;/* Haus */
                    else if (rectangle(x, y, 3, 5, 3, 12)) return 1;  /* Fenster */
                       else if (rectangle(x, y, 12, 0, 18, 10)) return 1; /* Tuer */
                          else if (rectangle(x, y, 21, 5, 27, 12)) return 1; /* Fenster */
                              else if (rectangle(x, y, 9, 20, 14, 24)) return 1; /* Schornstein */
                                 else if  (line(x, y, 0 ,15, 5, 20)) return 1; /* Dachschraege */
                                   else if (line(x,y, 25, 20, 30, 15)) return 1; /* Dachschraege */
                                     else if (line(x,y, 6, 20, 25, 20)) return 1; /* Dachfirst */
          else return 0;
}
```

1. Auswertung Hausaufgabeserie 3

```
int main() {
int i, j;
           /* Ausgabe zeilenweise, deshalb Ausgabe von oben nach unten */
           for(i=35; i>=0;i--) {
                for(j=0; j<=35;j++) {
                                   if( imBild(j, i))
                                            printf("*");
                                    else
                                            printf(" ");
                           }
                           printf("\n");
           }
           return(0);
}
```

2. Ideen zu Hausaufgabenserie 4

1. Schreiben Sie ein rekursives Programm, das alle Permutationen von n Elementen ausgibt. (D.h., alle möglichen unterschiedlichen Reihenfolgen der Elemente.) Das Programm soll die Zahl n als Kommandozeilenargument akzeptieren (Sie können die Methode hierfür von hanoi.c kopieren). Sie können davon ausgehen, dass n nicht größer als 26 ist, so dass Sie die Elemente mit Buchstaben bezeichnen können. Ihr Programm soll damit beispielsweise folgende Ausgabe liefern:

```
$ ./perm 3
1: ABC
2: BAC
3: CAB
4: ACB
5: BCA
6: CBA
6 permutations
$
```

Tipp: Heap's Algorithmus. (12 Punkte)

Wikipedia:

Heap's algorithm generates all possible permutations of *n* objects. It was first proposed by B. R. Heap in 1963. The algorithm minimizes movement: it generates each permutation from the previous one by interchanging a single pair of elements; the other $n-2$ elements are not disturbed. In a 1977 review of permutation-generating algorithms, Robert Sedgewick concluded that it was at that time the most effective algorithm for generating permutations by computer.

The sequence of permutations of *n* objects generated by Heap's algorithm is the beginning of the sequence of permutations of $n+1$ objects.

2. Ideen zu Hausaufgabeserie 4

Aufgabe 1 Permutation von n – Elementen

Formale Umsetzung einer Algorithmenbeschreibung

https://en.wikipedia.org/wiki/Heap's_algorithm

```
procedure generate(k : integer, A : array of any):
    if k = 1 then
          output(A)
    else
        for i := 0; i < k - 1; i += 1 do
            generate(k - 1, A)
            if k is even then
                swap(A[i], A[k-1])
            else
                swap(A[0], A[k-1])
            end if
        end for
        generate(k - 1, A)
    end if
```

Legen sie das Feld „A“ z.B. als globales Feld von Zeichen an.

char a[ ] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

*Eine Funktion swap (int *i, int *j) soll die Elemente a[i] und a[j] vertauschen*

Rest ist formale Umsetzung der Algorithmenbeschreibung.

2. Ideen zu Hausaufgabenserie 5

2. Sortieren. Entwickeln Sie eine Funktion zum Sortieren eines Feldes von Zahlen mit Hilfe der Funktion swap aus der Vorlesung.
Die Idee ist einfach:

- *Wenn zwei benachbarte Zahlen im Feld in der falschen Reihenfolge sind, dann werden diese vertauscht.*
- *Dies machen Sie so lange, bis keine Vertauschungen mehr erforderlich sind.*

15 Punkte

2. Ideen Hausaufgabenserie 4

3. Betrachten Sie folgendes Code-Fragment:

```
#define N 29
#define C_SPACE 26
#define C_COMMA 27
#define C_STOP 28

int getcc() {
         intc, haveSpace=0;
         while (isspace(c=getchar())) haveSpace = 1;

         if(haveSpace)return(ungetc(c,stdin),C_SPACE);
         else if(c>=’a’&&c<=’z’) returnc-’a’;
         else if(c>=’A’&&c<=’Z’)returnc-’A’;
         else if(c==’,’) return C_COMMA;
         else if(c==’.’) return C_STOP;
         else if(c==EOF) return EOF;
         else return getcc();}
```

Erläutern Sie, was dieser Code macht. Falls Ihnen die Funktionen isspace und ungetc nicht bekannt sind, recherchieren Sie, was diese Funktionen tun. (5Punkte)

2. Ideen Hausaufgabenserie 4

4. Schreiben Sie ein Programm, das Text in „Pseudo-Englisch“ erzeugen kann. Dazu gehen Sie folgendermaßen vor:

• Bauen Sie ein dreidimensionales Feld auf, das für jedes Zeichen c (Buchstaben sowie Leerzeichen, Komma, und Punkt) die bedingte Wahrscheinlichkeit beinhaltet, dass das Zeichen c auf die Zeichen a und b folgt.

Diese Tabelle können Sie beispielsweise durch die Analyse von Herman Melvilles „Moby Dick“ erzeugen. (10 Punkte)

• Entwickeln Sie ein Verfahren, mit dem Sie aus einem Vektor von Werten (v_1, ... , v_n), für die Sie einen Vektor von Wahrscheinlichkeiten (p_1, ... , p_n) gegeben haben, einen Wert v_i zufällig wählen können, so dass die Wahrscheinlichkeit, dass Sie v_i bekommen eben genau p_i entspricht. Sie können davon ausgehen, dass gilt

$$\sum_{i=1}^{n} p_i$$

10 Punkte

• Erzeugen Sie mit Hilfe des dreidimensionalen Feldes einen Zufallstext dadurch, dass Sie sich jeweils die beiden zuletzt generierten Zeichen a und b merken und dann ein neues Zeichen c zufällig wählen, wobei Sie die Wahrscheinlichkeit eines bestimmten Zeichens c gegeben die Vorgänger a und b aus dem Feld entnehmen. Um das erste Zeichen zu erzeugen können Sie annehmen dass die Vorgängerzeichen der Punkt und das Leerzeichen sind.

10 Punkte

2. Ideen Hausaufgabenserie 4

Idee für Kodierungstabelle:

Neben Buchstaben kommen auch Leerzeichen, Komma und Punkt in Texten vor.
Idee für Kodierungstabelle (26 Buchstaben + Sonderzeichen):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z		,	.	?

Textausschnitt aus Mobi Dick : „once a whale“

Zählen von Häufigkeiten

- *3D – Matrix für 3er-Kombinationen P*
- *Die Buchstabenkombination „**once**“ wird zerlegt in „**onc**“ und „**nce**“ und muss an der Stelle P_abc[14][13][2] und P_abc[13][2][4] gespeichert werden, da o(14), n(13), c(2), e(4), dann folgen „ce “, „e a“, „ a “, „a w“, „ wh“, „wha “, „hal“, „ale“, „le “*

- *2D – Matrix für 2er Kombinationen F*
  F_ab[14][13] und F_ab[13][2]
- *Nach Ende der Zählung eines Gesamttextes, **Bildung der relativen Häufigkeiten für bedingte Wahrscheinlichkeit.***
- *(Division der 3D-Zellen durch zugehörige 2D-Zelle (auf die Folge von 2 Buchstaben folgt mit xx % Wahrscheinlichkeit eines der 26+X Zeichen))*

2. Ideen Hausaufgabenserie 4

Generierung von Text ähnlich wie Analyse.
(Aus 2 vorhergehenden Buchstaben folgt mit gewisser Wahrscheinlichkeit nächster Buchstabe (Oder Sonderzeichen).)

Teilaufgabe 2 Verfahren zur Entsprechung Wahrscheinlichkeit p_i
Bei vorgegebener Wahrscheinlichkeit ist eventuelle keine der 26+X allein groß genug, um die Wahrscheinlichkeit zu erreichen. (Summierung der Werte bis Wahrscheinlichkeit erreicht; dann 3. Buchstaben laufen lassen, Startpunkt eventuell zufällig, aber Modulo Alphabetlänge, eventuell maximalen Wert, ...)
Aufgabe im Wesentlichen eine Formel beschreiben!

Teilaufgabe 3 Textgenerierung mit Nutzung Formel aus 2.
Startwert a = C_STOP; b= C_SPACE
Schleifen können ähnlich zu Teilaufgabe 1 (Analyse) sein!

Aufruf des Programms zur Generierung eines 50 Zeichen langen Pseudoenglisch Textes

Pseudo.exe 50 <"mobidick.txt"

z.B: Ausgabe war: and of to cringed be, abages of offrourfun upook d

3. Cäsar-Kodierung

Verschlüsselung eines Buchstabens durch Verschiebung um n Stellen.

Aus der Zeichenfolge „hallo“ wird bei Verschiebung der Kodierung um 3 Stellen („h“-> „k“, aus „a“ -> „d“, aus „l“ -> „o“, aus „o“->“r“) -> „kdoor“

```
#include <stdio.h>
#define MAX 100
#define N 26

char imsg[MAX];
char omsg[MAX];

int main() {
  int shf, c;
  printf("Shift = ");
  scanf("%d",&shf);
  fflush(stdin);
 /* fpurge(stdin); /* clear input, in stdin_ext.h (ubuntu) */
  printf("Message = ");
  gets(imsg); /* einlesen von Text in das Feld imsg */
  for(i=0;imsg[i];i++) {
    c = imsg[i];
    omsg[i] = (c >= 'a' && c <= 'z') ? ((c-'a')+shift)%N+'a' : c;
  }
  printf("Coded = %s\n",omsg);

return 0;
}
```

3. Cäsar-Kodierung

```
#include <stdio.h>
#define MAX 100
#define N 26

char imsg[MAX];
char omsg[MAX];

int main() {
  int shf, c;
  printf("Shift = ");
  scanf("%d",&shf);
  fflush(stdin);
 /* fpurge(stdin); /* clear input, in stdin_ext.h (ubuntu) */
  printf("Message = ");
  gets(imsg); /* einlesen von Text in das Feld imsg */
  for(i=0;imsg[i];i++) { /* imsg[i] ist aeqivalent mit imsg[i] !=`0` oder imsg[i]!=0 */
    c = imsg[i];
    omsg[i] = (c >= 'a' && c <= 'z') ? ((c-'a')+shf)%N+'a' : c;
    /*  Gleichwertig zu
    if(c >= 'a' && c <= 'z')
           omsg[i] = ((c-'a')+shf)%N+'a';
       else
           omsg[i] = c; */
  }
  printf("Coded = %s\n",omsg);

return 0;
}
```

3. Cäsar-Kodierung mit Funktion

Verschlüsselung ausgelagert in einer Funktion cipher

```
#include <stdio.h>

#define MAX 100
#define N 26
char imsg[MAX];
char omsg[MAX];

void cipher(int shift) {
  int i,c;
  for(i=0;imsg[i];i++) {
    c = imsg[i];
    omsg[i] = (c >= 'a' && c <= 'z') ? ((c-'a')+shift)%N+'a' : c;
  }
  omsg[i]='\0';
}

int main() {
  int shf;
  printf("Shift = ");
  scanf("%d",&shf);
  fflush(stdin);
  printf("Message = ");
  gets(imsg);
  cipher(shf);
  printf("Coded = %s\n",omsg);

  return 0;
}
```

3. Cäsar-Dekodierung

Die Dekodierung (26 –n) kann mit der gleichen Funktion geschehen, es muss nur die Verschiebung n bekannt sein.

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

#define MAX 100
#define N 26
char imsg[MAX];
char omsg[MAX];

void cipher(int shift) {
  int i,c;
  for(i=0;imsg[i];i++) {
    c = imsg[i];
    omsg[i] = (c >= 'a' && c <= 'z') ? ((c-'a')+shift)%N+'a' : c;
  }
  omsg[i]='\0';
}

int main() {
  int shf;
  printf("Shift = ");
  scanf("%d",&shf);
  fflush(stdin);
 // fpurge(stdin); /* clear input, in stdin_ext.h (ubuntu) */
  printf("Message = ");
  gets(imsg);
  cipher(shf);
  printf("Coded = %s\n",omsg);
  strcpy(imsg,omsg); /* Funktion zum kopieren von Strings in c (0-terminiert) */
  cipher(26-shf);
  printf("Decoded = %s\n",omsg);
  return 0;
}
```

3. Cäsar-Dekodierung

Wie lässt sich der Code knacken, wenn n (Shift) unbekannt?

Alle 26 Verschiebungen probieren und visuell kontrollieren (schlecht!)

Idee:

Zählen von Häufigkeiten der Buchstaben
Vergleich von erwarteter Häufigkeit („ist in der Sprache ist der Buchstabe in Textene so häufig“) mit Häufigkeit der Zeichen in verschlüsselter Nachricht.

Vergleich von Häufigkeiten: die X^2 (chi-Quadrat)-Statistik

$$X^2 = \sum_{i=0}^{N-1} \frac{(ofreq_i - efreq_i)^2}{efreq_i}$$

Wählen den kleinsten Wert für X^2 aus, wenn i die Buchstabennummer und N = 26

3. Cäsar-Dekodierung

$$X^2 = \sum_{i=0}^{N-1} \frac{(ofreq_i - efreq_i)^2}{efreq_i}$$

Wählen den kleinsten Wert für X^2 aus, wenn i die Buchstabennummer und N = 26

```
/* Expected frequencies for letters in english text */

/* Verteilung der Buchstaben in englischen Texten In Prozent */
double efreq[N] = {
  8.2, 1.5, 2.8, 4.3, 12.7, 2.2, 2.0, 6.1, 7.0, 0.2, 0.8, 4.0, 2.4,
  6.7, 7.5, 1.9, 0.1, 6.0, 6.3, 9.1, 2.8, 1.0, 2.4, 0.2, 2.0, 0.1
};
```

3. Cäsar-Dekodierung

```
void decypher() {
  double ofreq[N]={0}, nchar;
  double chmin=HUGE_VAL; /* positive infinity,  in math.h */
  double chisq;
  int i,shift,smin;

  /* reset data */
  for(i=0;i<N;i++) ofreq[i] = 0;

  /* compute observed frequencies */
  for(nchar=0.0,i=0;imsg[i];i++) {
    int c = imsg[i];
    if(c >= 'a' && c <= 'z') {
      ofreq[c-'a']++;
      nchar++;
    }
  }
  /* convert to percentages */
  for(nchar/=100.0,i=0;i<N;i++) ofreq[i] /= nchar;

  /* compute chisq stats */
  for(shift=0;shift<N;shift++) {
    chisq = 0.0;
    for(i=0;i<N;i++)
      chisq += pow(ofreq[(i+shift)%N] - efreq[i],2.0) / efreq[i];
    if(chisq < chmin) { chmin = chisq; smin = shift; }
  }
  printf("Best shift: %d\n",smin);
  /* decode */
  encode(N-smin);
}
```

4. Felder

Nach einfachen Variablen jetzt auch Variable für mehrere Werte!
Deklaration

Datentyp Variablenname[Kapazitaet];

Vor der Verwendung eines Arrays muss die Größe feststehen!
Arrays in C können ihren aktuellen Füllstand nicht selbst bestimmen!

```
/* Variable fuer 100 int- Werte */
    int f[100];
/* Variable fuer 100 char- Werte */
    char puffer[100];
/* Variable fuer 100 x 100 char- Werte */
     char feld[100][100];
/* Weise dem 10sten Eintrag den Wert 42 zu ( Zaehlung beginnt bei 0 !) */
     f[9] = 42;

/* Weise dem ersten beiden Einträgen die Zeichen ‚T‘ und ‚r‘ zu; */
     puffer[0] = 'T';
     puffer[0] = 'r';

/* Steht in der 3- Zeile und im 4. Zeichen ein ‚T‘ ? */
    if(feld[2][3] == 'T')
```

4. Felder und Referenzen

In C gibt es einen sehr interessanten Zusammenhang zwischen Feldern und Referenzen: Wenn der Typ eines Ausdrucks „Feld mit Elementen vom Typ T“ ist, dann ist der Wert des Ausdrucks eine Referenz auf das erste Element (das Element Nummer 0) dieses Feldes. (Siehe Folie 126 Vorlesung ff)

```
float f[5] = {13.2, 11.9, 12.5, 12.3, 13.8};
```

Der Ausdruck „f“ ist vom Typ „Feld mit Elementen vom Typ float“,
der Wert von „f“ ist eine Referenz auf die erste Feldvariable- also eine Referenz auf f[0].
Damit ist „f“ ein Zeiger auf mehrere floats.

3. Felder und Referenzen

Betrachten wir folgendes Codefragment, wobei T irgend ein Typ (hier float) ist.

```
float a[5], *r;
r = a;
```

Dies besagt, r ist eine Referenz auf die Variable a[0].
Die Variable a[0] hat den Typ float. Eine Variable vom Typ float benötigt
s = sizeof float ➔ also 4 Byte Speicher.

Wenn also a[0] die Adresse p hat, dann besitzt a[1] die Adresse p + s.
Und allgemein hat a[k] die Adresse p + k * s.

Wenn r den Typ *float hat, und in r irgendeine Adresse p gespeichert ist, liefert der Ausdruck r+k die Adresse p + k * s, wobei gilt s = sizeof(float).

Den Wert s nennen wir manchmal auch Objektgröße einer Referenz.

3. Felder und Referenzen

Beispielprogramm: feldadresse.c

```
#include <stdio.h>

int main(int argc, char *argv[]) {
      int i;
      float *zf;
      double *zg;
      float f[] ={13.2f, 11.9f, 12.5f, 12.3f, 13.8f};
      float f1[5];
      double g[] ={13.2, 11.9, 12.5, 12.3, 13.8};

      printf("Adresse von f ist %d und von &f[0] ist %d \n", f, &f[0]);
      printf("sizeof von f %d\n", sizeof f );

      printf("Adresse von g ist %d und von &g[0] ist %d \n", g, &g[0]);
      printf("sizeof von g %d\n", sizeof g );

      zf=f;
      zg=&g[0];

      printf("Adresse von f ist %d und von zf ist %d \n", f, zf);
      printf("sizeof von zf %d, und sizeof (*zf) %d \n", sizeof zf, sizeof (*zf) );

      printf("Adresse von g ist %d und von zg ist %d \n", g, zg);
      printf("sizeof von zg %d  sizeof (*zg) %d\n", sizeof zg, sizeof(*zg));

      printf("\n");
      ...
```

```
$ feldadresse.exe
Adresse von f ist 6356720 und von &f[0] ist 6356720
sizeof von f 20
Adresse von g ist 6356656 und von &g[0] ist 6356656
sizeof von g 40
Adresse von f ist 6356720 und von zf ist 6356720
sizeof von zf 4, und sizeof (*zf) 4
Adresse von g ist 6356656 und von zg ist 6356656
sizeof von zg 4  sizeof (*zg) 8
```

3. Felder und Referenzen

```
    int i;
    float *zf;
    float f[] ={13.2f, 11.9f, 12.5f, 12.3f, 13.8f}
    float f1[5];

    printf("Ausgabe Schleife mit f[i]\n");
    for (i=0; i< 5; i++)  printf("%f, ", f[i]);
    printf("\n");

    printf("Ausgabe Schleife mit zf\n");
    for (i=0; i< 5; i++)  printf("%f, ", *(zf+i));
    printf("\n");

    printf("Ausgabe Schleife mit f\n");
    for (i=0; i< 5; i++)  printf("%f, ", *(f+i));
    printf("\n");

    f1[0] = f[4];     /* Speichere in Index 0 den Wert von Index 4 aus Feld f */
    *(f1+1) = f[3];   /* Speichere in Index 1 den Wert von Index 3 aus Feld f */
    *(f1+2) = *(zf+2); /* Speichere in Index 2 den Wert von Index 2 aus Feld f */
    f1[3] =  *(zf+1); /* Speichere in Index 3 den Wert von Index 1 aus Feld f */
    *(f1+4) = f[0]; /* Speichere in Index 4 den Wert von Index 1 aus Feld f */
    printf("Ausgabe Schleife mit f1\n");
    for (i=0; i< 5; i++)
          printf("%f, ", *(f1+i));
    printf("\n");
    if(*zf ==f[0])
       printf(" *zf == f[0] ist immer richtig, da *zf= %f und f[0]= %f \n", *zf, f[0]);
    if(*(zf+2) ==f[2])
       printf(" *zf+2 == f[2] ist immer richtig, da *(zf+2)= %f und f[2]= %f \n", *(zf+2), f[2]);
  return 0;
}
```

```
Ausgabe Schleife mit f[i]
13.200000, 11.900000, 12.500000, 12.300000, 13.800000,
Ausgabe Schleife mit zf
13.200000, 11.900000, 12.500000, 12.300000, 13.800000,
Ausgabe Schleife mit f
13.200000, 11.900000, 12.500000, 12.300000, 13.800000,
Ausgabe Schleife mit f1
13.800000, 12.300000, 12.500000, 11.900000, 13.200000,
 *zf == f[0] ist immer richtig, da *zf= 13.200000 und f[0]= 13.200000
 *zf+2 == f[2] ist immer richtig, da *(zf+2)= 12.500000 und f[2]= 12.500000
```

3. Felder und Referenzen

```
void ausgabe(int anz, float *ff){
      int i;
      for (i=0; i< anz; i++)
            printf("%f, ", *(ff+i));
      printf("\n");

}

void ausgabe1(int anz, float ff[]){
      int i;
      for (i=0; i< anz; i++)
            printf("%f, ", ff[i]);
      printf("\n");
}


int main(int argc, char *argv[]) {
      int i;
      float *zf;
      double *zg;
      float f[] = {13.2f, 11.9f, 12.5f, 12.3f, 13.8f};

      ausgabe(5,f);
      ausgabe1(5,f);

  return 0;
}
```

3. Felder und Referenzen

Teilaufgabe aus Prüfung: Gegeben sei die folgende Deklaration eines Feldes: (9 Punkte)

```
float feld [5][4]; /* IEEE 754 single precision */
```

Vervollständigen Sie die folgende Tabelle, in dem Sie den Typ, die Adresse und die Objektgröße der folgenden Ausdrücke angeben.

Ausdruck	Typ	Adresse	Objektgröße
feld	float(*)[4]	p	16
&feld[0][0]			
&feld[1][2]			
feld[2]			
&feld			

(Beispiel aus Klausuraufgabe 2015) (besprechen wir nächste Woche)

4. Aufgaben

1. Aufgabe: Erzeugung eines Feldes von 100 Zahlen zufällig verteilt, keine Zahl soll doppelt vorkommen. (Arbeit mit Funktion swap)
   - Legen sie ein globales int-Array mit 100 Feldern an und speichern sie dort die Werte von 0 bis 99.
   - Schreiben Sie eine Funktion zur Ausgabe des Feldes. Nach jeweils 10 Zahlen soll eine neu Zeile begonnen werden
     (`void ausgabe (int l, int f[])`).
   - Nutzen sie die verbesserte swap-Funktion(`void swap (int *a, int *b)`) aus der Vorlesung und tauschen n mal zwei zufällige Positionen im Array.
   - Nutzen Sie für die Eingabe von n die Methode aus dem Programm hanoi (Das Programm soll die Zahl n als Kommandozeilenargument akzeptieren)

```
int main (int argc , char * argv []) {
      int n;
      if( argc < 2 || (n = atoi ( argv [1])) < 0) {
      printf (" Usage : %s <number -of -permutation >\n",argv [0]);
      return -1;
      }
```

4. Aufgaben

2. Ermittlung der Häufigkeiten von 1er, 2er und 3er Buchstabenkombinationen in Texten

- Globale Variablen und Definitionen

```
/* 26 Buchstaben + Sonstige */
#define N 27

char alphabet[] = "abcdefghijklmnopqrstuvwxyz#";
double F_abc[N][N][N];
double F_ab[N][N];
double F_a[N];
double F;
```

- Schreiben Sie eine Funktion `int possiblechar(int c)`, die für das Zeichen c prüft, ob es sich um ein Zeichen aus dem Alphabet handelt (1 wenn zutreffend, 0 wenn nicht).
- Nutzen Sie die Funktion readchar um zeichenweise Texte einzulesen:

```
int readchar() { /* Lese gueltige Zeichen ein und ueberlese andere, ermittle index in Tabelle */
            int c, anz = 0;
            /* Ueberlese nicht gueltige Zeichen und mehrere Leerzeichen*/
            while (((c = getchar()) != EOF) && (!possiblechar(c))) {
                        anz++;
            }
            if(c == EOF) return EOF;
            if (c >= 'a'&&c <= 'z') return c - 'a';
                        else if (c >= 'A'&&c <= 'Z')return c - 'A';
                                    else return N;
}
```

- Schreiben Sie eine Funktion `void fcount()`, die in den Feldern F_abc, Fab, F_a und der Variablen F die 3er, 2er und 1er Häufigkeiten und die Gesamtanzahl der gültigen Zeichen zählt. Nach der Zählung einer Kombination rücken die Buchstaben von b zu a und von c zu b auf, auf c wird ein neues Zeichen (readchar) gelesen. Stimmt ihre prozentuale Verteilung der Buchstaben in F_a mit der Verteilung aus der Cäsarkodierung überein, wenn Sie das Buch Mobi dick analysieren?