Informatik

Projekt Protokoll

Kryptologie

Stichworte:

Pascal-Datentypen:

- ARRAY
- STRING
- RECORD
- FILE

Pascal-Graphik

Codieren, Decodieren:

- Verschiebealgorithmen
- Caesar
- feste Zuordnung
- Vigenère
- Schlüsselwörter
- Textanalyse
- interaktive Decodierung
- PGP, RSA (nur erwähnt, nicht behandelt)

Schule	Gymnasium Hermeskeil
Kurs	Klasse 12 6 Gruppen zu 2 bis 3 Schülerinnen und Schüler
Lehrer	Mario Spengler
Projektleiter	Mario Spengler

Inhalt

Arbeitsgruppe Informatik LMZ

Vorwort

Zu Beginn des Halbjahres 12/1 wurde größter Wert auf die gemeinsame Wahl des Projektthemas gelegt. Möglichst viele Teilnehmer sollten sich mit dem Thema identifizieren können. Vom Programmieren her sollte das Thema so komplex sein, daß es sich leicht in Teilthemen zerlegen läßt und eine Vertiefung von Pascal im Bereich der Datenstrukturen ermöglicht. Bezüge zur Gesellschaft sollten bestehen. Man beachte, daß 1998 in der Presse das Thema Kryptologie bezüglich der Schlüsselwortlänge international kontrovers diskutiert wurde. Des weiteren sind Verschlüsselungstechniken ein festen Bestandteil von Datenschutz- und Datensicherungsmaßnahmen.

Erst nach dieser Phase der Themenfindung war klar, welche Pascal-Strukturen noch erarbeitet werden mußten, nachdem in 11/2 der Schwerpunkt in der Graphikprogrammierung lag. Daher mußten nun die String-Operationen und das Datei-Konzept vorgestellt werden, bevor die eigentliche Arbeit am Projektthema „Kryptologie“ fortgesetzt werden konnte.

Ein zweiter Schwerpunkt der Projektarbeit lag in der sorgfältigen Ausarbeitung einer Projektspezifikation vor der Programmierphase, welches genau festlegt, welche Aufgabe sich jede Gruppe stellt und zu erfüllen hat. Alle sechs Teile wurden anschließend mit Hilfe der Großbildprojektion von der gesamten Gruppe diskutiert, korrigiert und abgestimmt. Das gedruckte Exemplar war von nun an Pflichtenheft und Nachschlagewerk für die gesamte Gruppenarbeit.

Globale Variablen, Menüstruktur und die Behandlung von Ereignissen wurden vom Projektleiter übernommen, so daß die Programmierarbeit der einzelnen Gruppen darin bestand, das bereits lauffähig vorgegebene Programm

- Start
- Menü einblenden
- Beenden

durch anwendungsspezifische Unterprogramme zu ergänzen.

Der Projektleiter verwendet seit mehreren Jahren stets den gleichen Programmaufbau und die gleiche Ereignisbearbeitung, so daß jedes Jahr nur

- wenige (!) globale Variablen und
- die Menü- Resource

angepaßt werden müssen.

Unterrichtsbeschreibung zur Kryptologie

1. Projektthema 4 h

Erörterung und Festlegung eines geeigneten Themas, welches
- viele Kursteilnehmer interessiert
- Bezüge zu Datenschutz und Datensicherung hat
-eine Vertiefung der Pascal- Datenstrukturen erlaubt, siehe ⇒ Anlage 1

Der Kurs entschied sich für das Thema Codierung-Decodierung,
weil einerseits in den Medien zur Zeit viel Codierung im www die Rede ist.

2. Datenstrukturen in Pascal 10 h

Es wurde in aller Kürze das Filekonzept und die Stringroutinen in Pascal vorgestellt, an einfachen Beispielen programmiert und getestet. Siehe ⇒ Anlage 2
Anschließend eine schriftliche Überprüfung, siehe ⇒ Anlage 3

3. Standardverfahren zur Codierung 2 h

Es wurden sehr kurz Verschiebealgorithmen, freie Zuordnungen und der Vigenèrealgorithmus vorgestellt und Literatur angegeben.

4. Erstellung einer Projektspezifikation 8 h

Einteilung in Arbeitsgruppen mit spezifischen Themen.
Ausarbeiten eines Protokolls, welches die Aufgabenstellung mit Erläuterung und Lösungsansätze der einzelnen Arbeitsgruppen übersichtlich darstellt und als Pflichtenheft für die Programmierarbeit dienen soll.
Überarbeitung durch die gesamte Gruppe, siehe ⇒ Anlage 4
anschließend eine schriftliche Überprüfung wie in ⇒ Anlage 5

5. Programmiephase, Teil 1 8 h

Vorgabe eines lauffähigen Programmtorsos, bestehend aus Ereignisbehandlung (Maus,Tasten) und Menüberarbeitung, jedoch ohne jede Ausformulierung von Codierungsprozeduren
etwa wie auf den ersten Seiten von ⇒ Anlage 4
Gruppenweise Programmierung der einzelnen Unterprogramme
Kursarbeit in ⇒ Anlage 6

6. Programmiephase, Teil 2 10 h

Angleichung der gemeinsam benötigten Unterprogramme, Vereinbarung weiter Schnittstellen, neue Vergabe von Arbeitsaufträgen: Die Themen „Textanalyse“ und „Interaktiv“ mußten weiter verfeinert werden.
Ausgaben, Ergebnisse und Bildschirmkopien in ⇒ Anlage 7
Lauffähiges Programm in THINK-Pascal for Macintosh in ⇒ Anlage 8

8. Datenschutz und Datensicherung 8 h

Vorstellung des PGP -und des RSA - Algoritmus ⇒ Anlage 9
Fragen zur aktuellen politischen Diskussion über Schlüssellängen;
Datenschutz und Datensicherung etwa wie in der Handreichung zum Lehrplan Informatik : Projekte, S.49 - 120

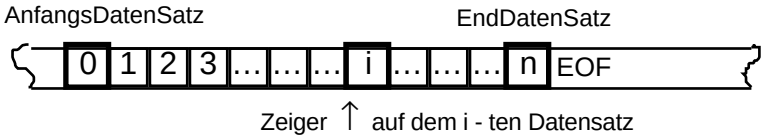
Anlage 1: Projektideen 12/1 mit Bezügen zum Datenschutz

	Titel (+ : Chancen, - : Risiken)	1. Datei	2. Datei
1.	Fragebogenaktion +:Statistik der Antworten -:Repersonalisierung Bewertung der Person	anonyme Antworten	Name, GebDat, Wohnort, ...
2.	Tele-AB-Gebühr +:einf. Abrechnung -:Bewegungsprofil	Name, Einfahrt (Km,Zeit) Ausfahrt (km, Zeit)	Straßennetz, vorgeschr. Geschw., ...
3.	Einkauf mit mit Kreditkarte +: einf. Abrechnung, -: Konsumverhalten, Bewegungsprofil	KäuferName, VerkäuferName, Kosten, Datum, ...	VerkäuferName, Ort, Warenangebot, ...
4.	Einkommen & Steuer +:Steuerberechnung - : Bewertung der Arbeit Krankheitstage, ...	Name, Beruf reales Einkommen, Arbeitstage, (Grund)Besitz,...	Beruf, mittleres Einkommen, VorjahresDaten,
5.	Medikamente +: Abrechnung, Mißbrauch ausschließen, Nebenwirkungen, -: Gesundheitsprofil, Sparen statt heilen,	PatientenName, Medikamentenname, Anzahl, Preis Datum, Diagnose,...	MedikamentenName, geeignet für.., Nebenwirkungen, Alternativen, ...
6.	Müll, Abwasser +:gerechte Abrechnung - :Umweltprofil Datum, ...	Name, Müll in kg, Abwasser in Litern Grundflächen,.....	Name, Personenzahl, Art der Firma,
7.	Börsenspiel +:Entscheidungshilfe - : Käuferprofil	Name des Aktionärs, Name der Aktie, Einsatz pro Tag (+/ -), ...	Name der Aktie, Kurs pro Tag, Tendenz, ...
8.	(De)kodierungen + : sicherer Datentransfer - : Insiderwissen, 2-Klassengesellschaft	TextDatei	Verteilung der - Buchstaben - Bigramme - Trigramme

Anlage 2: Das File-Konzept in THINK-Pascal

dateiName:= erfragt „dateiName“
oldFileName (‘leer‘); zum Lesen

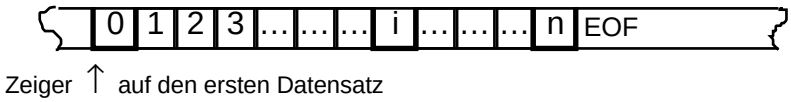
dateiName:= erfragt „dateiName“
newFileName (‘Kommentar‘, ‘Vorschlag‘); zum Schreiben



rewrite (datei, dateiName); setzt den Zeiger auf den Dateianfang,
löscht alle Datensätze und
öffnet zum Schreiben

reset (datei, dateiName); setzt den Zeiger auf den Dateianfang und
öffnet zum Lesen

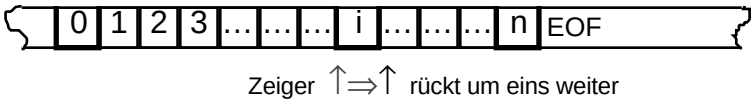
open (datei, dateiName); setzt den Zeiger auf den Dateianfang und
öffnet zum Lesen und zum Schreiben



close(datei); schließt die Datei
Eintrag ins Platteninhaltsverzeichnis

read (datei , datenSatz); liest „datenSatz“ und
rückt Zeiger eins nach rechts

write (datei, datenSatz); schreibt „datenSatz“ und
rückt Zeiger eins nach rechts



eof (datei); ist „True“, wenn das Dateiende erreicht ist

seek (datei , i); setzt Zeiger auf den „i“-ten Datensatz

position:=
filepos (datei); bestimmt die „position“ des aktuellen Zeigers

Anlage 2 : String-Routinen

S := concat (s_1,s_2,…,s_n);	Verkettung der Teilstrings s_1, s_2, …… , s_n zu einem Gesamtstring S.
P := pos (teil, quelle);	Position P des Beginns eines Teilstrings „teil“ im Gesamt-string „quelle“; P ist 0, falls „quelle“ nicht „teil“ enthält.
L := length (quelle);	Länge L des Strings „quelle“.
S := copy (quelle, anf, anz);	Kopiert ab der Position „anf“ aus „quelle“ einen Teilstring der Länge „anz“.
S := omit (quelle, anf, anz);	Aus dem String „quelle“ ab der Position „anf“ genau „anz“ Zeichen entfernen.
insert (teil, quelle, anf);	Fügt den „teil“-String ab der Position „anf“ in den String „quelle“ ein.
delete (quelle, anf, anz);	Löscht im „quelle“-String „anz“ Zeichen ab „anf“.
numToString (num, str);	wandelt eine longInt-Zahl in einen str255-String um.
stringToNum(str, num);	wandelt einen str255-String in eine longInt-Zahl um.

char-Funktionen

C := pred (c);	Vorgänger des Zeichens „c“
C := succ (c);	Nachfolger des Zeichens „c“
C := chr (n);	Zeichen der Ordnungszahl „n“
N := ord (c);	Ordnungszahl von „c“

```
PROGRAM stringTests;                          { --------- Beispielprogramm --------- }
VAR  num      : longInt;
  L, P, N     : integer;
  xStr, yStr  : string;
  zStr        : str255;
  c           : char;
                                               { ↓ ↓ ↓ ↓↓ ↓ ↓ ↓↓ ↓ ↓ ↓ ↓ ↓    Ausgabe }
BEGIN
 xStr := ‚Berta ist eine gute‘;
 writeln(‚xStr   :‘, xStr);                   { xStr: Berta ist eine gute            }
 xStr :=concat(‚Tante ‚, xStr,‘ Frau’);
 writeln(‚xStr   :‘, xStr);                   { xStr: Tante Berta ist eine gute Frau }
 P := pos(‚gute‘, xStr);
 writeln(‚pos(„gute“, xStr) =’, P);           { pos(„gute“, xStr) =      22          }
 xStr := omit(xStr, P, 5);
 writeln(‚xStr   :‘, xStr);                   { xStr   :Tante Berta ist eine Frau    }
 L := length(xStr);
 writeln(‚length(xStr)      =‘, L);           { length(xStr)      =      25          }
 yStr := copy(xStr, 7, L - 6);
 writeln(‚yStr   :‘, yStr);                   { yStr: Berta ist eine Frau            }
 P := pos(‚eine‘, yStr);
 insert(‚k‘, yStr, P);
 writeln(‚yStr   :‘, yStr);                   { yStr: Berta ist keine Frau           }
 delete(yStr, 3, 16);
 writeln(‚yStr   :‘, yStr);                   { yStr: Beau                           }
 c:=yStr[2]);
 writeln(‚yStr[2]       : ‚‚      c);                          {      yStr[2]  : e }
 writeln(‚pred(yStr[2]) : ‚‚ pred(c);                          { pred(yStr[2]) : d }
 writeln(‚succ(yStr[2]) : ‚‚ succ(c);                          { succ(yStr[2]) : f }
 writeln(‚chr(64)   = ‚‚ chr(64));                               { chr(64)   =  @ }
 writeln(‚chr(65)   = ‚‚ chr(65));                               { chr(65)   =  A }
 writeln(‚ord(„B“)  = ‚‚ ord(‚B‘) : 2);                          { ord(„B“)  = 66 }
 writeln(‚ord(„C“)  = ‚‚ ord(‚C‘) : 2);                          { ord(„C“)  = 67 }
 zStr := ‚123‘;
 stringToNum(zStr, num);
 writeln(‚num    :‘, num);                                                { num: 123  }
 num := 3 * 123;
 numToString(num, zStr);
 writeln(‚zStr   :‘, zStr);                                              { zStr: 369 }
END.
```

| ---- Steuerzeichen ---- | ---------- Schriftzeichen ---------- |

HEX		00		10		20		30		40		50		60		70
0	NUL	0	DLE	16		32	0	48	@	64	P	80		96	p	112
1	SOH	1	DC1	17	!	33	1	49	A	65	Q	81	a	97	q	113
2	STX	2	DC2	18	"	34	2	50	B	66	R	82	b	98	r	114
3	ETX	3	DC3	19	#	35	3	51	C	67	S	83	c	99	s	115
4	EOT	4	DC4	20	$	36	4	52	D	68	T	84	d	100	t	116
5	ENQ	5	NAK	21	%	37	5	53	E	69	U	85	e	101	u	117
6	ACK	6	SYN	22	&	38	6	54	F	70	V	86	f	102	v	118
7	BEL	7	ETB	23	'	39	7	55	G	71	W	87	g	103	w	119
8	BS	8	CAN	24	(	40	8	56	H	72	X	88	h	104	x	120
9	HT	9	EM	25	)	41	9	57	I	73	Y	89	i	105	y	121
A	LF	10	SUB	26	*	42	:	58	J	74	Z	90	j	106	z	122
B	VT	11	ESC	27	+	43	;	59	K	75	[	91	k	107	{	123
C	FF	12	FS	28	,	44	<	60	L	76	\	92	l	108	|	124
D	CR	13	OS	29	-	45	=	61	M	77	]	93	m	109	}	125
E	SO	14	RS	30	.	46	>	62	N	78	^	94	n	110	—	126
F	SI	15	US	31	/	47	?	63	O	79	_	95	o	111	DEL	127

Anlage 3 : Zum File- und String-Konzept

1. Erläutern Sie, was folgendes Pascal-Programm bewirkt und ergänzen Sie direkt die noch fehlenden Befehle, indem Sie die Stelle mit <——————————— markieren und den Befehl an den rechten Rand schreiben.

```
PROGRAM xyz;

VAR  datei: text;
     dateiName  : string;

BEGIN
     open(datei,‘Diskette:MatheAufgaben‘);
     WHILE NOT eof(datei) DO
     BEGIN
       read ( datei, zeichen );
       CASE zeichen OF
         ‘(‘ : zeichen := ‘[‘;
         ‘)‘ : zeichen := ‘]‘;
       seek(datei, filepos(datei) - 1 );
       write(datei,zeichen);
     END;
     close(datei);
END.
```

2. Schreiben Sie auf der Rückseite ein Pascalprogramm, welches eine beliebige TextDatei zeilenweise liest, die Zeilen zählt und am Ende die Anzahl ausgibt.

3. Ergänzen Sie hier auf der Vorderseite die Pascal-Wörter `reset` und `concat` jeweils zu einer beispielhaften und syntaktisch korrekten Befehlszeile, vereinbaren Sie die verwendeten Variablen und erläutern Sie Funktion und die Anwendung der beiden Befehle in der Praxis.

3.0 `VAR`

3.1 `reset`

3.2 `concat`

Anlage 4 : Protokoll der Projektspezifikation

Anm.: nur wenig überarbeites Original der Schülerprotokolle

1. Hauptprogramm Platz 02

1.1 Menü

1	2	3	4	5
	Ablage	Codierung	Decodierung	Werkzeuge
1	Original laden O	Caesar C	Caesar R	Textanalyse T
2	Bereinigt laden B	Zuordnung Z	Zuordnung G	----------
3	----------	Vigenère V	Vigenère E	Interaktiv I
4	Sichern als... S			
5	----------			
6	Beenden(Quit) Q			

1.2 Textvorbereitung zur Codierung

Um einen Text zu codieren, müssen zuerst alle Satzzeichen und Wortabstände eliminieret werden.Außerdem verändert man alle Umlaute (ä, ö, ü) zu Einzel-buchstaben. Als nächstes wandelt man den ganzen Text in Groß-buchstaben um. Erst dann kann der Text weiterverarbeitet werden.

Beispiel:

„Es war einmal ein kleines Regenwürmchen namens Bert .“ { Original }
„EswareinmaleinkleinerRegenwuermchennamensBert .“ { ohne Umlaute }
„ESWAREINMALEINKLEINESREGENWUERMCHENNAMENSBERT“ { groß }

1.3 Globale Variablen

Als globale Variablen werden die Variablen festgelegt, die für alle allgemein gültig sind und das Hauptprogramm betreffen, nicht die Unterprogramme:

```
PROGRAM De_Codierung;                                   { Vorgabe Oktober 1998 }

USES   GrafikUnit;

CONST  N     = 4096;

TYPE   str3  = STRING[3];
       typ3  = RECORD   anzahl: integer;
                        tripel: str3;
               END

       hist  = ARRAY[1..26] OF typ3;

VAR  multiFinder, ende                  : boolean;
       ereignis                         : eventRecord;
       dateiName                        : string;
       textLaenge, buAnzahl, gesamt     : integer;
       textVorlage                      : ARRAY[1..N] OF char;
       standard, buchstaben             : ARRAY[1..3] OF hist;
```

1.4 Ereignis-Schleife des Hauptprogramms { Vorgabe Oktober 1998 }

```
BEGIN { Hauptprogramm }

Menue_installieren;
Anfangsbelegungen;
ende := false;

WHILE NOT ende DO  BEGIN
                     Ereignis_behandeln;
                   END;
END.
```

1.5 Zu schreibende Unterprogramme, etwa { Vorgabe Oktober 1998 }

```
PROCEDURE original_laden;
BEGIN
     menu_zeigen(2, 4, true);
     menu_zeigen(3, 0, true);
     menu_zeigen(4, 0, true);
     menu_zeigen(5, 0, true);
END; { original_laden }

PROCEDURE bereinigt_laden;
BEGIN
      menu_zeigen(2, 4, true);
      menu_zeigen(3, 0, true);
      menu_zeigen(4, 0, true);
      menu_zeigen(5, 0, true);
 END; { bereinigt_laden }

PROCEDURE sichern_als;
BEGIN
END; { sichern_als }

PROCEDURE caesar (codieren: boolean);
BEGIN
END; { caesar }

PROCEDURE zuordnung (codieren: boolean);
BEGIN
END; { zuordnung }

PROCEDURE vigenere (codieren: boolean);
BEGIN
END; { vigenere }

PROCEDURE textAnalyse;
BEGIN
END; { textAnalyse }

PROCEDURE interAktiv;
BEGIN
END; { interAktiv }
```

1.6
```
PROCEDURE Menue_bearbeiten (Info: longint);              { Vorgabe Oktober 1998 }
VAR menuNumm, menuItem: integer;
BEGIN { Menue_bearbeiten }
IF info <> 0 THEN
BEGIN   PenNormal;
        IF menu_markiert(4, 4) THEN item_aendern(4, 4);
        MenuNumm := Hiword(info) - RES_ID_NR + 1;
        menuItem := LoWord(info);

CASE MenuNumm OF

1:   { Sehr speziell : Behandelt das -----> };                      { APPLE-Menü }

2:   CASE menuItem OF                                              { Menü : Ablage }
     1:  BEGIN
     original_laden;          dateiname := concat(dateiname, ,.O‘);
     END;
2:   BEGIN
     bereinigt_laden;         dateiname := concat(dateiname, ,.B‘);
     END;
4:   sichern;
6:   ende := true;
     END; { CASE menuItem 2 }

3:   CASE menuItem OF                                            { Menü : Codierung }
     1:  BEGIN
     caesar(true);            dateiname := concat(dateiname, ,C‘);
     END;
2:   BEGIN
     zuordnung(true);         dateiname := concat(dateiname, ,Z‘);
     END;
3:   BEGIN
     vigenere(true);          dateiname := concat(dateiname, ,V‘);
     END;
     END; { CASE menuItem 3 }

4:   CASE menuItem OF                                          { Menü : Dekodierung }
1:   BEGIN
     caesar(false);           dateiname := concat(dateiname, ,c‘);
     END;
2:   BEGIN
     zuordnung(false);        dateiname := concat(dateiname, ,z‘);
     END;
3:       BEGIN
     vigenere(false);         dateiname := concat(dateiname, ,v‘);
     END;
     END;{ CASE menuItem 4 }

5:   CASE menuItem OF                                            { Menü : Werkzeuge }
1:   BEGIN
     textAnalyse(TRUE);       dateiname := concat(dateiname, ,t‘);
     END;
3:   BEGIN
     interAktiv;              dateiname := concat(dateiname, ,i‘);
     END
     END;{ CASE menuItem 4 }

END; { CASE MenuNumm }

hiliteMenu(0);

END; { IF info <> 0 }

END; { Menue_bearbeiten }
```

2. Cäsar- Verfahren

Platz 03

2.1 Codierung:

Ein gegebener Text wird buchstabenweise um eine bestimmte Anzahl „v“ an Plätzen in beliebiger Richtung im Alphabet verschoben (siehe Beispiel 1). Der gegebene Text muss hierzu in Grossbuchstaben, ohne Satzzeichen und ohne Umlaute vorhanden sein. Die zu verschiebende Zahl wird mit der “MOD 26“- Funktion so vereinfacht, dass der jeweilige Buchstabe um maximal v=26 Plätze im Alphabet verschoben wird (siehe Beispiel 2).
Eine Verschiebung um 0 oder 26 Plätze ist keine Verschiebung (Beispiel 3).
Die Buchstaben eines Textes, der in Großbuchstaben und ohne Sonderzeichen vorliegt, werden um eine feste Anzahl v verschoben.

65	66	67	68	69	70	71	72	73	74	Ordnungszahl
A	B	C	D	E	F	G	H	I	J	
01	02	03	04	05	06	07	08	09	10	Position
75	76	77	78	79	80	81	82	83	84	Ordnungszahl
K	L	M	N	O	P	Q	R	S	T	
11	12	13	14	15	16	17	18	19	20	Position
85	86	87	88	89	90					Ordnungszahl
U	V	W	X	Y	Z					
21	22	23	24	25	26					Position
48	49	50	51	52	53	54	55	56	57	Ordnungszahl
0	1	2	3	4	5	6	7	8	9	Ziffer

2.2 Beispiele:

```
SEHRGEEHRTEDAMENUNDHERREN
↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓ v = 1
TFISHFFISUFEBNFOVOEIFSSFO
```

```
SEHRGEEHRTEDAMENUNDHERREN
↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓ v = 40 MOD 26 =  14
GSVFUSSVFHSROASBIBRVSFFSB  ord('E')   -64  =  5
                           chr(5+14   +64) =  'S'
```

```
SEHRGEEHRTEDAMENUNDHERREN
↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓ v = 26 MOD 26 =   0
SEHRGEEHRTEDAMENUNDHERREN
```

2.3 Vorgehensweise bei Buchstaben:

p = ORD(buchstabe)-64 ist die Position der einzelnen Buchstaben im Alphabet;
q = p + v ergibt die neue Position des Buchstabens im Alphabet.
Falls das Ergebnis q > 26, dann berechnet man die neue Position zu
q = q MOD 26. Der alte Buchstabe wird durch den neuen CHR (q+64) ersetzt .

Beispielsatz: „ESWAREINMALEINKLEINERREGENWURMNAMENSBERT“

An diesem Beispielsatz wollen wir die Codierung mit v=8 demonstrieren.

“E“ hat die Ordnungszahl 69,	“S“ hat die Ordnungszahl 83,
p = 69-64 =5,	p = 83-64 = 19,
q = p+v = 5+8 = 13,	q = p+v = 19+8 = 27, q = 27 MOD 26 = 1
CHR(q+64) =CHR(77) =“M“;	CHR(q+64) =CHR(65) =“A“;

Ergebnis:
MAEIZMQVUITMQVSTMQVMZZMOMVEYZUVIUMVAJMZB

2.4 Vorgehensweise bei Ziffern:

Unterschied zwischen der Vorgehensweise bei Ziffern und Buchstaben:
Falls das Ergebnis q > 10, dann berechnet man die neue Ziffer q = q MOD 10 .

Beispiel: 4711, Codierung mit v=4

“4“ hat die Ordnungszahl 52,	“7“ hat die Ordnungszahl 55,
p = 52-48 =4,	p = 55-48 =7,
q = p+v = 4+4 = 8,	q = p+v = 7+4 = 11, q = 11 MOD 10 = 1
CHR(q+48) =CHR(56) =“8“;	CHR(q+48) =CHR(49) =“1“;

Ergebnis: 8155

2.5 Decodierung:

Durch eine Textanalyse werden die Häufigkeiten von verschiedenen Buchstaben festgestellt. In der deutschen Sprache ist, statistisch gesehen, der Buchstabe „E“ der häufigste. Wendet man diese Textanalyse nun an dem verschlüsselten Text an, lässt sich der am häufigsten vorkommende Buchstabe feststellen. Das „E“ nimmt den Alphabetplatz 5 ein; stellt man nun den Alphabetplatz des im verschlüsselten Text am häufigsten auftretenden Buchstaben fest und bildet die Differenz zu Alphabetplatz 5, bekommt man die Zahl, um die der Text verschoben wurde, heraus. Verschiebt man nun den gesamten Text um diese errechnete Zahl in negativer Richtung, ist der Text entschlüsselt. Dabei gilt:

- Eine Verschiebung um v= -1 entspricht einer Verschiebung um v=25;
- Eine Verschiebung um “-v“ entspricht einer Verschiebung um “26-v“.

Im Beispiel 2 ist das „S“ der häufigste Buchstabe. Das „S“ belegt den Alphabetplatz 19. Errechnet man nun die Differenz zwischen Alphabetplatz 5 („E“) und 19 („S“) bekommt man die Zahl (14) heraus, um die der Text verschoben wurde.

Beispiel:
MAEIZMQVUITMQVSTMQVMZZMOMVEYZUVIUMVAJMZB

Ergebnis:
ESWAREINMALEINKLEINERREGENWURMNAMENSBERT

Man stellt fest, welcher Buchstabe im codierten Text am häufigsten vorkommt. Dieser Buchstabe ist im Deutschens tatistisch gesehen ein „E“
O_e ist die Position von „E“ im Alphabet
O_h des codierten Buchstaben muß festgestellt werden
$O_h - O_e = v$ ist die Anzahl, um die jeder Buchstabe verschoben wurde
$O_c - v = O_{neu}$ ist die Position eines beliebigen codierten Buchstabens;
falls O_{neu} negativ ist siehe Ausnahmefall; O_c wird durch O_{neu} ersetzt.

Ausnahmefall:
$O_c - n = x$, $26+x=O_{neu}$, x Variable, die negativ werden kann.

3. Textanalyse

Platz 04

Ein Programm soll erstellt werden, das uns die Häufigkeit bestimmter Buchstaben und Buchstabenkombinationen in einem Text nennt. Dazu müssen Variablen vereinbart werden, in denen die Häufigkeit der jeweiligen Buchstaben und Buchstabenkombinationen gespeichert wird.

```
CONST   H_MAX = 100;

TYPE      str2 =    STRING[2];              str3 =    STRING[3]; { global }
          typ2 =    RECORD                  typ3 =    RECORD
                    anzahl : integer;               anzahl : integer;
                    tripel : str2;                  tripel : str3;
                    END;                            END;

VAR b_Haeufigkeit   : ARRAY[‘A‘..‘Z‘] OF integer;                { Buchstaben }
    bi_Haeufigkeit  : ARRAY[1..H_MAX] OF typ2;                   { Bigramme   }
    tri_Haeufigkeit : ARRAY[1..H_MAX] OF typ3;                   { Trigramme  }
```

3.1 Vorgehen:

Die Häufigkeit von Buchstaben, Paaren (Bigramme) , Tripeln (Trigramme) in einem codierten Text wird untersucht, um als Grundlage zur Entschlüsselung des Textes zu dienen. Je nach Häufigkeit der Buchstaben bzw. Buchstabenfolgen werden diese den in der deutschen Sprache ebenso häufig vorkommenden Buchstaben bzw. Buchstabenfolgen zugeordnet , die wir zuvor anhand von Probetexten ermittelt haben.

Beispiel: “FJTUEFSIBFVGJATVFCVDITVECFJOEFSEFVUTDIFOTQSBDIF“

Als häufigster Buchstabe ergibt sich „ F“. Da in der deutschen Sprache der häufigste Buchstabe das „E“ ist, folgt daraus, dass der Buchstabe „F“ im codierten Text bei der Entschlüsselung durch den Buchstaben „E“ ersetzt wird.

Lösung: “EISTDERHAEUFIGSTEBUCHSTABEINDERDEUTSCHENSPRACHE“

3.2 Algorithmus für das Zählen einzelner Buchstaben:

Jeder der 26 Buchstaben wird durchgezählt, wobei jedem eine Variablen zugeordnet wird, die ihm selber entspricht. Der Text wird buchstabenweise gelesen und die Zähler werden um 1 erhöht, sobald der entsprechende Buchstabe registriert wurde. Die Häu figkeit wird durch „I“ sinnvoll normiert und graphisch dargestellt (etwa ein “|“ für 0,25%), außerdem wird sie in % gerundet ausgegeben.

3.3 Algorithmus für das Zählen der 10 häufigsten Bigramme:

Die ersten zwei Buchstaben im Text werden gelesen und ihre Häufigkeit im Text festgestellt. Danach verfährt man genauso mit dem 2. und 3. Buchstaben, die das nächste Paar bilden. Spätestens beim 100. Buchstaben wird gestoppt und nur die 50 häufigsten aufgehoben. Der Rest des Textes wird genauso bearbeitet. Am Schluss werden aus allen gespeicherten Paare die 26 häufigsten graphisch dargestellt und mit Prozentzahlen angegeben.

Buchstaben:

A: 8	||||||||		
B: 4	||||		
C: 4	||||		
D: 2	||		
E: 28	||||||||||||||||||||||||||		
F: 2	||		
G:6	|||||		
H:5	|||||		
I: 15	||||||||||||||		
J:			
K:1	|		
L: 5	|||||		
M: 3	|||		
N: 20	|||||||||||||||||||		
O:2	||		
P:1	|		
Q:		Bigramme:	Trigramme:
R: 14	|||||||||||||		
S: 9	|||||||||	ei: I	sch: I
T:8	||||||||	ae:	ung: I
U:4	||||	am:	der: I
V:		eu:	die:
W: 3	|||	un:	ein: I I I
X:		in: I	das:
Y:		ng:	
Z: 1	|	en: I I	

3.4 Algorithmus für das Zählen der 10 häufigsten Trigramme:

Die ersten drei Buchstaben im Text werden gelesen und ihre Häufigkeit im Text festgestellt. Danach verfährt man genauso mit dem 2. ,3.und 4. Buchstaben, die das nächste Tripel bilden usw. Spätestens beim 100. Tripel wird gestoppt und nur die 50 häufigsten aufgehoben. Der Rest des Textes wird genauso bearbeitet. Am Schluss werden aus allen gespeicherten Buchstabentripeln die 26 häufigsten herausgefiltert und wie die Buchstaben und Bigramme graphisch dargestellt und mit Prozentzahlen angegeben.

3.5 Datenreduktion

Diese Datenreduktion ist aus Speicherplatzgründen notwendig. Zum Speichern der Buchstabenhäufigkeiten sind nur 26 real-Zahlen notwendig.
Bei Buchstabenpaaren gibt es bereits 26·26 =676 Möglichkeiten, außerdem muß dasbetreffende Paar mit gespeichert werden.
Demzufolge gibt es bei den Tripeln 26·26·26 = 17576 Möglichkeiten, außerdem müssen die betreffenden Tripel mit gespeichert werden.

4. Codierung mit fester Zuordnung Platz 05

4.1 Codieren

Man bestimmt zunächst ein beliebiges Paßwort (mit der Länge des Paßwortes steigt der Schwierigkeitsgrad der Dekodierung). Wenn das Paßwort gleiche Buchstaben mehrmals enthält, werden sie einfach weggestrichen (Beispiel: „SCHIFFFAHRT“ wird zu „SCHIFART“). Nun schreibt man die restlichen Buchstaben des Alphabets zeilenweise unter das Paßwort (siehe Beispiel). Die neue Buchstabenfolge, spaltenweise gelesen, ergibt zugeordnet zu dem Alphabet den Verschlüsselungscode .

Beispiel:
“Es war einmal ein kleiner Regenwurm namens Bert.“
“ESWAREINMALEINKLEINERREGENWURMNAMENSBERT“

Paßwort: **BAUMAUB**

B	**A**	**U**	**M**
C	D	E	F
G	H	I	J
K	L	N	O
P	Q	R	S
T	V	W	X
Y	Z		

Zuordnung:

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
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
B	C	G	K	P	T	Y	A	D	H	L	Q	V	Z	U	E	I	N	R	W	M	F	J	O	S	X

Verschlüsselter Probetext:
RJBNPDZVBQPDZLQPDZPNNPYPZJMNVZBVPZRCPNW

4.2 Decodieren

Passwort bekannnt

Decodieren mit dem Passwort. Hierzu wird das Paßwort und das Alphabet wie beim Codieren hingeschrieben.

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
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
B	C	G	K	P	T	Y	A	D	H	L	Q	V	Z	U	E	I	N	R	W	M	F	J	O	S	X

Nur wird die Zuordnung umgekehrt und zur Vereinfachung umsortiert.

Passwort unbekannt

Decodieren mittels Textanalyse. Zuerst wird die Häufigkeit aller Buchstaben im Text bestimmt . Die Häufigkeiten vergleicht man mit bekannten Häufigkeiten in deutschen Texten. Hier ist meist das „E“ der häufigste Buchstabe. Durch interaktives Ersetzen einzelner Buchstaben wird der Text nun entschlüsselt. Weiter helfen bekannte Buchstabenpaare oder Buchstabentripel.

5. Interaktive Dekodierung

Platz 10

Interaktive Oberfläche zur Decodierung von codierten Texten mit Hilfe der Grafikprogrammierung in Pascal

5.1 Ziel

Das Ziel unserer Arbeitsgruppe ist die Programmierung einer komfortablen, benutzerfreundlichen und grafisch ansprechenden Oberfläche für das Decodieren verschlüsselter Text. Der Benutzer soll interaktiv Schritt für Schritt alle Texte entschlüsseln können, die nach einer festen Zuordung codiert wurden.
Der codierte Text soll in einer sehr schwachen Schrift in einem Grafikfenster zu sehen sein.Nun kann der Benutzer nach der Häufigkeit der Buchstaben suchen. Angenommen in dem codierten Text ist der Buchstabe „**S**“ am häufigsten vertreten, so wird dieser durch ein „**E**“ in kräftiger Farbe ersetzt, weil „**E**“ der häufigste Buchstabe im deutschen Alphabet ist.

5.2 Beispiel

codiert: SA QYT SNBKYP SMB JPSMBST TSFSBQVTK BYKSBA DSTC.
decodiert : ES WAR EINMAL EIN KLEINER REGENWURM NAMENS BERT.

Der Text lautet nun wie folgt:
EA QYT EMBKYP EMB JPEMBET TEFEBQVTK BYKEBA DETC.

Durch logisches Denken oder Betrachtung von Bigrammen kommt man darauf, daß unter Beachtung des ersten Wortes nur ein „**S**“ für ein „**A**“ stehen kann.

Nun lautet der Text schon wie folgt :
ES QYT EMBKYP EMB JPEMBET TEFEBQVTK BYKEBS DETC.

Durch dieses Suchverfahren, verknüpft mit logischem Denken, bekommt man so den Lösungsschlüssel heraus. Graphisch wird dies sichtbar, durch eine Änderung der Farbe, etwa von hellblauem codierten Text zum roten decodierten Text.

5.3 Layout

5.4 Ablauf

Der Benutzer kann einen codierten Text laden und interaktiv entschlüsseln.
Auf einem gelben Hintergrund wird der Text (hellblau) dargestellt; darüber befinden sich zwei Bedienfelder „**alt**“ und „**STOP**“. Das Feld „CHOOSe⇔CHANGE“ erhält abwechselnd die Information „CHOOSE“ zur Wahl eines neuen Buchstabenkandidatens und „CHANGE“ zur Eingabe des zugeordneten Zeichens.

Vorgehensweise:

CHOOSE-Modus:

1. Anklicken des zu ersetzenden Buchstabens im hellblauen „alt“-Alphabet .

CHANGE-Modus:

2. Klicken auf den neuen Buchstaben im hellblauen „alt“-Alphabet .
3. „STOP“ zum Beenden der Ersetzungen

Reaktion:

1. In der „alt“ -Leiste leuchtet der zu ersetztende Buchstabe rot auf.

2.1 In der „neu“ -Leiste wird der „•“ durch den zugeordneten Buchstaben in roter Farbe ersetzt.

2.2 In der „alt“-Leiste wird der zu ersetzende Buchstaben hellblau gefärbt

2.3 Im Text werden alle hellblauen „alt“- Buchstaben durch rote „neu“- Buchstaben ersetzt.

3. Beenden des Menüs „Interaktiv“

Im obigen Layout wurden alle „W“ durch „E“ ersetzt.

6. Vigenère-Codierung

Platz 11

Bei der Vigenère-Codierung wird mit Hilfe eines Keywords um die Platznummer des Schlüsselbuchstabens im Alphabet verschoben (vgl. Cäsar-Verfahren).

Probetext: „Es war einmal ein kleiner Regenwurm namens Bert.“

Paßwort: BERT hat im Alphabet die Positionen 2, 5, 18, 20.

6.1 Codierung:

Der 1. Buchstabe des Textes wird um die Position des ersten Buchstabens des Paßworts verschoben, der 2. Buchstabe um die Platznummer des zweiten Buchstabens des Paßworts, u.s.w.. Nachdem dieser Vorgang mit dem letzten Buchstaben des Paßworts durchgeführt worden ist, wird das Paßwort erneut verwendet.

Tabelle zur Verschiebung der Buchstaben:

	ABCDEFGHIJKLMNOPQRSTUVWXYZ	v = 0
A	BCDEFGHIJKLMNOPQRSTUVWXYZA	v = 1
B	CDEFGHIJKLMNOPQRSTUVWXYZAB	v = 2
C	DEFGHIJKLMNOPQRSTUVWXYZABC	v = 3
D	EFGHIJKLMNOPQRSTUVWXYZABCD	v = 4
E	FGHIJKLMNOPQRSTUVWXYZABCDE	v = 5
F	GHIJKLMNOPQRSTUVWXYZABCDEF	v = 6
G	HIJKLMNOPQRSTUVWXYZABCDEFG	v = 7
H	IJKLMNOPQRSTUVWXYZABCDEFGH	v = 8
I	JKLMNOPQRSTUVWXYZABCDEFGHI	v = 9
J	KLMNOPQRSTUVWXYZABCDEFGHIJ	v = 10
K	LMNOPQRSTUVWXYZABCDEFGHIJK	v = 11
L	MNOPQRSTUVWXYZABCDEFGHIJKL	v = 12
M	NOPQRSTUVWXYZABCDEFGHIJKLM	v = 13
N	OPQRSTUVWXYZABCDEFGHIJKLMN	v = 14
O	PQRSTUVWXYZABCDEFGHIJKLMNO	v = 15
P	QRSTUVWXYZABCDEFGHIJKLMNOP	v = 16
Q	RSTUVWXYZABCDEFGHIJKLMNOPQ	v = 17
R	STUVWXYZABCDEFGHIJKLMNOPQR	v = 18
S	TUVWXYZABCDEFGHIJKLMNOPQRS	v = 19
T	UVWXYZABCDEFGHIJKLMNOPQRST	v = 20
U	VWXYZABCDEFGHIJKLMNOPQRSTU	v = 21
V	WXYZABCDEFGHIJKLMNOPQRSTUV	v = 22
W	XYZABCDEFGHIJKLMNOPQRSTUVW	v = 23
X	YZABCDEFGHIJKLMNOPQRSTUVWX	v = 24
Y	ZABCDEFGHIJKLMNOPQRSTUVWXY	v = 25
Z	ABCDEFGHIJKLMNOPQRSTUVWXYZ	v = 26

Originaler, gesäuberter Probetext:

ESWAREINMALEINKLEINERREGENWURMNAMENSBERT.

Verschlüsselung des Probetextes :

GXOUTJAHOFDYKSCFGNFYTWWAGSOOSRFUOJFMDJJN.

6.2 Entschlüsselung:

Um die Vigenère-Codierung bei unbekanntem Paßwort zu decodieren wird die Anzahl der Buchstaben des Keywords benötigt, welche durch eine Häufigkeitsanalyse herausgefunden werden kann. In der Regel ergibt sich nicht so ein klares Profil wie im folgenden fingiertem Beispiel. Etwas probieren wird man schon müssen.

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	|

Da das „E“ in der deutschen Sprache am häufigsten auftritt, müßten periodisch bestimmte Häufigkeiten auftreten. Im obigen Beispiel ist die Periode 4. Ist diese Länge bekannt, schreiben wir den codierten Text in soviel Spalten, wie die Periodenlänge. Die Zeilenlänge muss jeweils identisch mit der Länge des Paßworts sein.

GXOU
TJAH
OFDY
KSCF
GNFY
TWWA
GSOO
SRFU
OJFM
DJJ

Es ergeben sich Spalten, in denen die Buchstaben jeweils um dieselbe Anzahl v verschoben sind. Die weitere Decodierung erfolgt durch das Häufigkeitsprinzip, das für jede einzelne Spalte angewendet wird.

Die Spaltenschreibweise ist am Beispiel des Probetextes unter der Annahme dargestellt, dass das Paßwort laut obiger Häufigkeitsanalyse 4 Buchstaben hat.

Anlage 5 : Hausaufgabenüberprüfung (De)codierungen

1. Notieren Sie die angestrebte die Menü-Stuktur des Hauptprogramms:

	1	2	3	4	5
1					
2					
3					
4					
5					
6					

2. Der unten stehende Beispielsatz wurde von der Platte gelesen. Welche Inhalte haben dann die nebenstehenden Variablen ?

textLaenge =	textVorlage[02] =
	textVorlage[07] =
	textVorlage[13] =
	textVorlage[14] =

3. Codieren Sie den Beispielsatz : „FASSE DICH KURZ“

3.1 nach Cäsar mit der Verschiebung : v = 5

F	A	S	S	E		D	I	C	H		K	U	R	Z

3.2 nach Vigenère mit dem Paßwort : „BERTA“

F	A	S	S	E		D	I	C	H		K	U	R	Z

3.3 durch Zuordnung mit dem Schlüssel : „TRAGEFI“

F	A	S	S	E		D	I	C	H		K	U	R	Z

4. Wozu dient die interaktive Anzeige ?

5. Notieren Sie in Strichworten, wozu werden beim Decodieren die Buchstaben-häufigkeiten gebraucht werden

5.1 bei Cäsar

5.2 bei Vigenère

5.3 bei Zuordnungen ?

Anlage 6 : Kursarbeit

1. Im nachfolgenden Text wurde die erste Zeile irgendwie nach Cäsar codiert, die zweite nach Vigenère mit dem Paßwort „ARBEIT“. Decodieren Sie beide Zeilen!

   v = ……

N	B	F	J	A	N	R	W	V	J	U	N	R	W	V	J	N	M	L	Q	N	W

   Paßwort = „ARBEIT“

E	S	U	P	J	G	P	X	V	E	D	Z	V	K	U	E	D	L	C	S	P	P

Für alle nachfolgenden Aufgaben gelten die folgenden globalen Vereinbarung aus dem Projekt „(De)codierungen“ :

```
CONST    N=4096;
TYPE     str3 = string[3];
VAR      textLaenge : integer;
         textVorlage     : ARRAY[1...N] OF char;
```

2. Schreiben Sie eine Pascal-Funktion `FUNCTION klecks ( c:char):char;` welche jedem Zeichen „c“ ,das kein Buchstabe und keine Ziffer ist (also Satzzeichen oder Leerzeichen) einen „•“ zuordnet.

3. Schreiben Sie eine Pascal-Prozedur `PROCEDURE ausgabe;` welche den Text der `textVorlage` so auf dem Bildschirm ausgibt, daß eine neue Zeile begonnen wird, wenn in der Textvorlage ein Zeichen mit der Ordnungszahl 13 auftaucht.

4. Schreiben Sie eine Pascal-Funktion `FUNCTION zaehlen:integer;` welche die „ei“ s der `textVorlage` zählt .

5.1 Erläutern Sie die die folgende Pascal-Funktion!

```
FUNCTION grosseFrage:integer;
CONST    xyz  = 'OMA';
VAR      aaa,iii   : integer;
         ttt  : str3;
BEGIN
  aaa:=0; ttt:=xyz;
  FOR iii:=3 TO textLaenge DO
    BEGIN
         ttt[1]:=textVorlage[iii-2];
         ttt[2]:=textVorlage[iii-1];
         ttt[3]:=textVorlage[iii-0];
         IF ttt=xyz THEN aaa:=aaa+1;
    END;
    grosseFrage:=aaa;
END; { grosseFrage }
```

5.2 Was bewirkt die Anweisung `writeln( grosseFrage);` bei folgender Textvorlage : „WESSENOMALIEFERTETOMANSMESSERVOMAUTOSCHIEBER“

Anlage 7 : Programmlauf : Laden eines Textes

Herzlich willkommen,

Um einen Text codieren oder decodieren zu können,
muß er erst geladen werden.

Eine Textanalyse gelingt nur, wenn der Text bereinigt geladen wurde

Intern 80: Mo 01/98 pas: (De)Codierung:TextVorlagen:BND.TXT

Einleitung
Auf einem Treffen westlicher Geheimdienste in Pullach will man herausfinden, welcher von allen der beste ist. Die Agenten-Teams bekommen die Aufgabe gestellt, ein Wildschwein zu fangen. Alle Teams machen sich auf den Weg. Nach einer Stunde kommen die CIA-Mitarbeiter zurück. Sie haben einen von Kugeln durchlöcherten Klumpen Fleisch dabei, der nach einigen Untersuchungen als Wildschweinkadaver identifiziert wird. „Nicht schlecht“, sagt die Jury, „Hundert Punkte“. Nach zwei Stunden kommen die Agenten des israelischen Mossad zurück. Sie bringen eine ganze Wildschweinfamilie mit, jedes Tier mit einem einzigen Kopfschuß getötet. „Nicht schlecht“, sagt die Jury, „Zweihundert Punkte“. Man wartet weiter. Es wird Abend. Kurz bevor die Sonne untergeht, hört man Lärm aus dem Wald. Dann sieht man die Mitarbeiter des Bundesnachrichtendienstes näherkommen: Vier halten einen sich verzweifelt wehrenden Hirsch fest, während der fünfte auf das Tier einprügelt und es anbrüllt: „Gesteh endlich, daß du ein Wildschwein bist.“ Derartige Witze zeigen, daß es um den Ruf der „Dienste“, insbesondere des deutschen Auslandsnachrichtendienstes, des BND, nicht besonders gut bestellt ist. Das müssen seine Mitarbeiter immer wieder erfahren; so klagt etwa die BND-Telefonzentrale darüber, wer alles sich bemüßigt fühlt, dem BND helfend zur Seite zu stehen: Astrologen bieten - gegen horrendes Honorar - dem BND ihre Dienste ebenso an wie Wahrsager und Ufologen; Kriminelle preisen sich als Einbrecher oder Totschläger an, die man sofort engagieren müsse. Was hat man beim BND bloß falsch gemacht? Dem israelischen Mossad sagt man Unbesiegbarkeit nach. Der britische Auslandsgeheimdienst profitiert vom Ruf eines James Bond, und auch der US-amerikanischen CIA gelingt es nicht selten, mit Heldentum und Allwissenheit gleichgesetzt zu werden. Und der BND? Er scheint eine Ansammlung schlapphutbewehrter Beamter und engstirniger Knallchargen zu sein, die eine Panne nach der anderen produzieren. Sind die Pullacher Geheimdienstler etwa unfähig, sind sie gar überflüssig? Lautet das Motto dort nur noch: „Legende am Ende“?

Intern 80: Mo 01/98 pas: (De)Codierung:TextVorlagen:Standard wird geladen

Anlage 7 : Programmlauf : Bereinigtes Laden eines Textes

```
EINLEITUNGAUFEINEMTREFFENWESTLICHERGEHEIMDIENSTEINPULLACH
WILLMANHERAUSFINDENWELCHERVONALLENDERBESTEISTDIEAGENTENTEA
MSBEKOMMENDIEAUFGABEGESTELLTEINWILDSCHWEINZUFANGENALLETEAM
SMACHENSICHAUFDENWEGNACHEINERSTUNDEKOMMENDIECIAMITARBEITER
ZURUECKSIEHABENEINENVONKUGELNDURCHLOECHERTENKLUMPENFLEISCH
DABEIDERNACHEINIGENUNTERSUCHUNGENALSWILDSCHWEINKADAVERIDEN
TIFIZIERTWIRDNICHTSCHLECHTSAGTDIEJURYHUNDERTPUNKTENACHZWEI
STUNDENKOMMENDIEAGENTENDESISRAELISCHENMOSSADZURUECKSIEBRIN
GENEINEGANZEWILDSCHWEINFAMILIEMITJEDESTIERMITEINEMEINZIGEN
KOPFSCHUSSGETOETETNICHTSCHLECHTSAGTDIEJURYZWEIHUNDERTPUNKT
EMANWARTETWEITERESWIRDABENDKURZBEVORDIESONNEUNTERGEHTHOERT
MANLAERMAUSDEMWALDDANNSIEHTMANDIEMITARBEITERDESBUNDESNACHR
ICHTENDIENSTESNAEHERKOMMENVIERHALTENEINENSICHVERZWEIFELTWE
HRENDENHIRSCHFESTWAEHRENDDERFUENFTEAUFDASTIEREINPRUEGELTUN
DESANBRUELLTGESTEHENDLICHDASSDUEINWILDSCHWEINBISTDERARTIGE
WITZEZEIGENDASSESUMDENRUFDERDIENSTEINSBESONDEREDESDEUTSCHE
NAUSLANDSNACHRICHTENDIENSTESDESBNDNICHTBESONDERSGUTBESTELL
TISTDASMUESSENSEINEMITARBEITERIMMERWIEDERERFAHRENSOKLAGTET
WADIEBNDTELEFONZENTRALEDARUEBERWERALLESSICHBEMUESSIGTFUEHL
TDEMBNDHELFENDZURSEITEZUSTEHENASTROLOGENBIETENGEGENHORREND
ESHONORARDEMBNDIHREDIENSTEEBENSOANWIEWAHRSAGERUNDUFOLOGENK
RIMINELLEPREISENSICHALSEINBRECHERODERTOTSCHLAEGERANDIEMANS
OFORTENGAGIERENMUESSEWASHATMANBEIMBNDBLOSSFALSCHGEMACHTDEM
ISRAELISCHENMOSSADSAGTMANUNBESIEGBARKEITNACHDERBRITISCHEAU
SLANDSGEHEIMDIENSTPROFITIERTVOMRUFEINESJAMESBONDUNDAUCHDER
USAMERIKANISCHENCIAGELINGTESNICHTSELTENMITHELDENTUMUNDALLW
ISSENHEITGLEICHGESETZTZUWERDENUNDDERBNDERSCHEINTEINEANSAMM
LUNGSCHLAPPHUTBEWEHRTERBEAMTERUNDENGSTIRNIGERKNALLCHARGENZ
USEINDIEEINEPANNENACHDERANDERENPRODUZIERENSINDDIEPULLACHER
GEHEIMDIENSTLERETWAUNFAEHIGSINDSIEGARUEBERFLUESSIGLAUTETDA
SMOTTODORTNURNOCHLEGENDEAMENDE
```

```
Intern 80: Mo 01/98 pas: (De)Codierung:TextVorlagen:Standard wird geladen
```

Anlage 7 : Programmlauf : Textanalyse

```
------ 1769 Buchstaben erfaßt-------


------ Buchstaben unsortiert-------

A :  6.0 % ••••••••••••
B :  2.1 % •••••
C :  3.2 % •••••••
D :  5.7 % ••••••••••••
E : 17.8 % ••••••••••••••••••••••••••••••••••••••
F :  1.6 % ••••
G :  2.8 % ••••••
H :  4.6 % ••••••••••
I :  8.1 % ••••••••••••••••••
J :  0.2 %
K :  1.0 % ••
L :  4.0 % •••••••••
M :  3.2 % •••••••
N :  9.9 % ••••••••••••••••••••••
O :  2.2 % •••••
P :  0.7 % ••
Q :  0.0 %
R :  6.0 % ••••••••••••
S :  7.1 % ••••••••••••••••
T :  6.3 % ••••••••••••••
U :  4.0 % •••••••••
V :  0.4 % •
W :  1.9 % ••••
X :  0.0 %
Y :  0.1 %
Z :  1.1 % ••
```

----- Bigramme sortiert ------

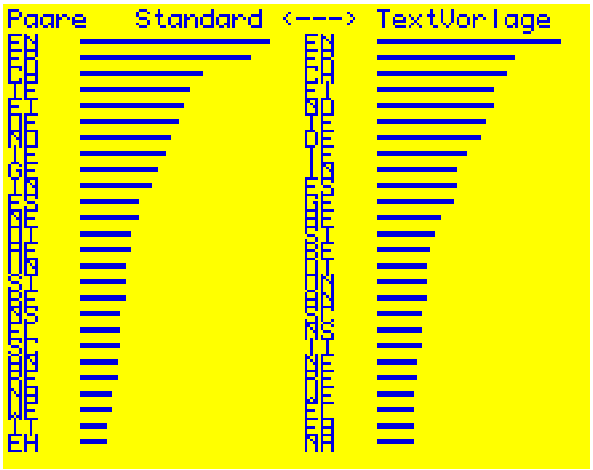
EN : 4.1 % ••
ER : 3.1 % ••••••••••••••••••••••••••••••••
CH : 2.9 % •••••••••••••••••••••••••••••••
EI : 2.6 % •••••••••••••••••••••••••
ND : 2.6 % •••••••••••••••••••••••••
TE : 2.4 % ••••••••••••••••••••••
DE : 2.3 % •••••••••••••••••••••
IE : 2.0 % ••••••••••••••••••
IN : 1.8 % •••••••••••••••••
ES : 1.8 % •••••••••••••••••
GE : 1.7 % ••••••••••••••••
HE : 1.4 % ••••••••••••
ST : 1.3 % ••••••••••••
BE : 1.2 % ••••••••••
DI : 1.1 % ••••••••••
UN : 1.1 % ••••••••••
AN : 1.1 % ••••••••••
SC : 1.0 % •••••••••
NS : 1.0 % •••••••••
IT : 1.0 % •••••••••
NE : 0.9 % ••••••••
RE : 0.9 % ••••••••
WE : 0.8 % •••••••
EL : 0.8 % •••••••
EH : 0.8 % •••••••
NA : 0.8 % •••••••

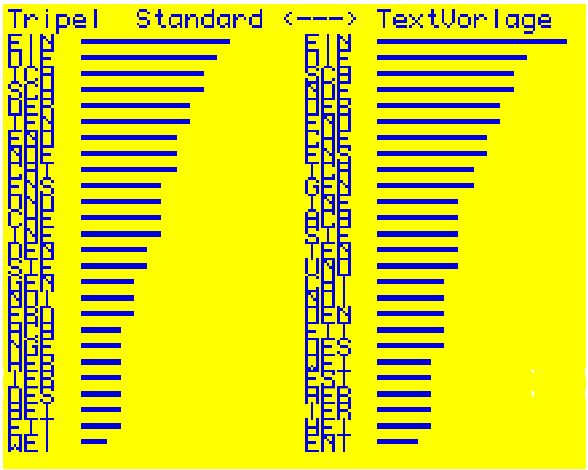
----- Trigramme sortiert ------

EIN: 1.4 % ••
DIE: 1.1 % ••••••••••••••••••••••••••••••
SCH: 1.0 % •••••••••••••••••••••••••••
NDE: 1.0 % •••••••••••••••••••••••••••
DER: 0.9 % ••••••••••••••••••••••
END: 0.9 % ••••••••••••••••••••••
CHE: 0.8 % ••••••••••••••••••••
ENS: 0.8 % ••••••••••••••••••••
ICH: 0.7 % •••••••••••••••••
GEN: 0.7 % •••••••••••••••••
INE: 0.6 % ••••••••••••••••
ACH: 0.6 % ••••••••••••••••
STE: 0.6 % ••••••••••••••••
TEN: 0.6 % ••••••••••••••••
UND: 0.6 % ••••••••••••••••
CHT: 0.5 % ••••••••••••••
NDI: 0.5 % ••••••••••••••
DEN: 0.5 % ••••••••••••••
EIT: 0.5 % ••••••••••••••
DES: 0.5 % ••••••••••••••
WEI: 0.4 % ••••••••••
EST: 0.4 % ••••••••••
HER: 0.4 % ••••••••••
IER: 0.4 % ••••••••••
HEI: 0.4 % ••••••••••
ENT: 0.3 % ••••••••

Anlage 7 : Programmlauf : Interaktiv

Durch Überblendung können jeweils die Statistiken von Buchstabenhäufigkeiten, Bigrammen und Trigrammen eingeblendet werden:





Anlage 8 : Lauffähiges Programm in THINK-Pascal for Macintosh

```
PROGRAM De_Codierung;    { Stand 20. Januar 1999 }

USES   GrafikUnit;

CONST  N                          = 4096;                       { maximale Textlänge }
       STANDARD_ERFASSEN          = FALSE;

TYPE   str2 = STRING[2];                                          { nur zur Abkürzung }
       str3 = STRING[3];
       typ3 = RECORD
                anzahl: integer;
                tripel: str3;
                END;
       hist = ARRAY[1..26] OF typ3;

 VAR   multiFinder, ende           : boolean;
       ereignis                    : eventRecord;
       dateiName                   : string;
       textLaenge, buAnzahl, gesamt : integer;
       textVorlage                 : ARRAY[1..N] OF char;
       standard, buchstaben        : ARRAY[1..3] OF hist;

 FUNCTION gross (c: char): char;
 BEGIN
  CASE c OF
   ,a'..'z': gross := chr(ord(c) - 32);
   OTHERWISE  gross := c;
  END
 END;{ gross }

 FUNCTION verschiebe (c: char; v: integer): char;
 BEGIN
     IF c IN [,A'..'Z'] THEN
            verschiebe := chr(65 + (ord(c) - 65 + v) MOD 26) ELSE
     IF c IN [,a'..'z'] THEN
            verschiebe := chr(97 + (ord(c) - 97 + v) MOD 26) ELSE
     IF c IN [,0'..'9'] THEN
            verschiebe := chr(48 + (ord(c) - 48 + v) MOD 10)
     ELSE verschiebe := c;
 END; { verschiebe }

 PROCEDURE item_aendern (spalte, zeile: integer);                    { kumulativ  }
  VAR  c            : char;
       i, n_item    : integer;
       aktuellesMenu: menuHandle;
 BEGIN
  aktuellesMenu := getMenu(spalte + RES_ID_NR - 1);
  n_item := countMItems(aktuellesMenu);
  IF zeile = 0 THEN
   FOR i := 1 TO n_item DO          checkItem(aktuellesMenu, i, false)
  ELSE
   BEGIN
    getItemMark(aktuellesMenu, zeile, c);
    IF ord(c) = 0 THEN              checkItem(aktuellesMenu, zeile, true)
    ELSE                            checkItem(aktuellesMenu, zeile, false);
   END; { ELSE }
 END; { item_aendern }
```

```
FUNCTION menu_markiert (spalte, zeile: integer): boolean;
 VAR  c: char;
      aktuellesMenu: menuHandle;
BEGIN
 aktuellesMenu := getMenu(spalte + RES_ID_NR - 1);
 getItemMark(aktuellesMenu, zeile, c);
 menu_markiert := (ord(c) = 18);
END; { menu_markiert }

PROCEDURE menu_zeigen (spalte, zeile: integer; zeigen: boolean);
 VAR  einMenu: menuHandle;    { zeile=0 bedeutet das ganze Menu }
BEGIN
 einMenu := getMHandle(RES_ID_NR + spalte - 1);
 IF zeigen THEN    enableItem (einMenu, zeile)
 ELSE                         disableItem(einMenu, zeile);
END; { menu_zeigen }

PROCEDURE standard_laden;
 VAR  datei   : FILE OF typ3;
      satz         : typ3;
      i, k         : integer;
      dName   : string;
BEGIN
 dName := dateiName;
 WHILE dName[length(dName)] <> ,:‘ DO
  delete(dName, length(dName), 1);
 dName := concat(dName, ,Standard‘);
 writeln;
 writeln(,••••• ,, dName, , wird geladen •••••‘);
 reset(datei, dName);
 FOR i := 1 TO 3 DO
  FOR k := 1 TO 26 DO   read(datei, standard[i, k]);
 close(datei);
END; { standard_laden }

PROCEDURE Anfangsbelegungen;
 VAR  i, k, typ: integer;
BEGIN
 textFont(Monaco);       textSize(9);
 gesamt := 0;                 textLaenge := 0;
 TextFenster_oeffnen;    showText;
 WITH Konsole DO
  buAnzahl := (right - left) DIV 7;
 FOR typ := 1 TO 3 DO
  FOR i := 1 TO 26 DO
   BEGIN buchstaben[typ, i].anzahl := i;
         buchstaben[typ, i].tripel := ,   ,;
         standard[typ, i].anzahl := i;
         standard[typ, i].tripel := ,   ,;
         FOR k := 1 TO typ DO
         BEGIN    buchstaben[typ, i].tripel[k] := chr(64 + i);
                  standard[typ, i].tripel[k] := chr(64 + i);
         END;
   END;
 ende := false;
 writeln(,Herzlich willkommen,‘); writeln;
 writeln(,Um einen Text codieren oder decodieren zu können,‘);
 writeln(,muß er erst geladen werden.‘); writeln;
 writeln(,Eine Textanalyse gelingt nur, wenn er bereinigt geladen wurde‘);
 writeln;
END; { Anfangsbelegungen }
```

```
PROCEDURE text_ausgeben;
 VAR  i, nr: integer;
BEGIN
 showText;
 nr := 0;
 FOR i := 1 TO textLaenge DO
  BEGIN
   nr := nr + 1;
   IF (ord(textVorlage[i]) = 13) OR (nr = buAnzahl) OR ((nr > buAnzahl - 20)
       AND (textVorlage[i] IN [, ,, ,-‘])) THEN
    BEGIN
     writeln;
     nr := 0;
    END;
   write(textVorlage[i]);
  END;
 writeln;
END; { text_ausgeben }

PROCEDURE original_laden;
 VAR  datei: text;
      zeichen: char;
BEGIN
 menu_zeigen(2, 4, true);
 menu_zeigen(3, 0, true);
 menu_zeigen(4, 0, true);
 menu_zeigen(5, 0, true);
 menu_zeigen(5, 1, false);
 IF NOT menu_markiert(2, 1) THEN
  item_aendern(2, 1);
 IF menu_markiert(2, 2) THEN
  item_aendern(2, 2);
 dateiname := oldfilename(, ,);
 writeln(dateiName);
 reset(datei, dateiname);
 textLaenge := 0;
 WHILE NOT eof(datei) AND (textLaenge < N) DO
  BEGIN
   WHILE NOT eoln(datei) AND (textLaenge < N - 1) DO
    BEGIN
     read(datei, zeichen);
     textLaenge := textLaenge + 1;
     textVorlage[textLaenge] := zeichen;
    END;
   readln(datei);
   textlaenge := textlaenge + 1;
   textVorlage[textLaenge] := chr(13);
   IF textLaenge MOD 100 = 0 THEN
    writeln(textLaenge);
  END;
 close(datei);
 text_ausgeben;
 IF NOT STANDARD_ERFASSEN THEN
  standard_laden;
END; { original_laden }
```

```
PROCEDURE bereinigt_laden;
 VAR  Datei: text;
      zeichen: char;
      i: integer;
 PROCEDURE ersetzen (nn: integer; c1, c2: char);
 BEGIN
  textlaenge := textlaenge + 1;
  textVorlage[textLaenge] := c1;
  IF nn = 2 THEN
   BEGIN
    textlaenge := textlaenge + 1;
    textVorlage[textLaenge] := c2;
   END;
 END; { zweiterBuchstabe }
BEGIN
 menu_zeigen(2, 4, true);
 menu_zeigen(3, 0, true);
 menu_zeigen(4, 0, true);
 menu_zeigen(5, 0, true);
 menu_zeigen(5, 1, true);
 IF NOT menu_markiert(2, 2) THEN
  item_aendern(2, 2);
 IF menu_markiert(2, 1) THEN
  item_aendern(2, 1);
 dateiname := oldfilename(, ,);
 reset(datei, dateiname);
 textlaenge := 0;
 WHILE NOT eof(Datei) AND (textLaenge < N) DO
  BEGIN
   WHILE NOT eoln(Datei) AND (textLaenge < N - 1) DO
    BEGIN
     read(datei, zeichen);
     CASE zeichen OF
      ‚ä‘, ‚Ä‘:
      ersetzen(2, ‚A‘, ‚E‘);
      ‚ö‘, ‚Ö‘:
      ersetzen(2, ‚O‘, ‚E‘);
      ‚ü‘, ‚Ü‘:
      ersetzen(2, ‚U‘, ‚E‘);
      ‚ß‘:
      ersetzen(2, ‚S‘, ‚S‘);
      ‚a‘..’z’:
      ersetzen(1, gross(zeichen), ‚-‘);
      ‚A‘..’Z’:
      ersetzen(1, zeichen, ‚-‘);
      ‚0‘..’9':
      ersetzen(1, zeichen, ‚-‘);
     END; { CASE }
    END; { WHILE not eoln }
   readln(datei);
  END;{While not eof}
 close(datei);
 text_ausgeben;
 IF NOT STANDARD_ERFASSEN THEN
  standard_laden;
END; { bereinigt_laden }
```

```
PROCEDURE sichern;
 VAR  Datei   : text;
      i            : integer;
BEGIN
 rewrite(Datei, dateiname);
 FOR i := 1 TO textlaenge DO
  write(Datei, textvorlage[i]);
 close(Datei);
END; { sichern_als }

PROCEDURE caesar (codieren: boolean);
 VAR  v, i: integer;
BEGIN
 IF codieren = true THEN
  BEGIN
   writeln;
   write(,Um wieviel Stellen im Alphabet wollen sie den Text verschieben ?‘);
   readln(v);
   FOR i := 1 TO textlaenge DO
    BEGIN
     textvorlage[i] := verschiebe(textvorlage[i], v);
    END;
  END{Codierung}
 ELSE
  BEGIN
   write(,Um wieviel Stellen im Alphabet wurde der Text verschoben ?‘);
   readln(v);
   FOR i := 1 TO textlaenge DO
    BEGIN
     IF textvorlage[i] IN [,a‘..’z’] THEN
      BEGIN
      textvorlage[i] := verschiebe(textvorlage[i], 26 - (v MOD 26));
      END
     ELSE IF textvorlage[i] IN [,A‘..’Z’] THEN
      BEGIN
      textvorlage[i] := verschiebe(textvorlage[i], 26 - (v MOD 26));
      END
     ELSE IF textvorlage[i] IN [,0‘..’9'] THEN
      textvorlage[i] := verschiebe(textvorlage[i], 10 - (v MOD 10));
    END;{Decodierung}
  END;
 text_ausgeben;
END; { caesar }
```

```
PROCEDURE zuordnung (codieren: boolean);
 VAR  codewort                          : STRING;
      Zugriff                           : ARRAY[1..26, 1..26] OF char;
      speicher, saver                   : ARRAY[,A'..'Z'] OF char;
      i, k, z, codewortlaenge, laenge   : integer;
      menge, menge2                     : SET OF char;
      b                                 : char;
 PROCEDURE codewort_kuerzen;
  VAR i, k: integer;
 BEGIN
  menge2 := [];                                      { Alle Sonderzeichen und Zahlen   }
  FOR i := 31 TO 126 DO                              { werden in die Menge aufgenommen }
   BEGIN   IF (i = 65) OR (i = 97) THEN
           i := i + 26;
           menge2 := menge2 + [chr(i)];
   END;
  menge2 := menge2 + [,´'];
  FOR i := 1 TO length(codewort) - 1 DO
   FOR k := i TO length(codewort) DO
    IF (codewort[i] = codewort[k]) AND (i <> k) OR (codewort[k] IN menge2)
    THEN   BEGIN
           delete(codewort, k, 1);               { verkürzen des Codewortes, so dass }
           i := i - 1;                 { jeder Buchstabe nur maximal ein Mal vorkommt }
           END;{ IF }
  codewortlaenge := length(codewort);                                 { Vereinfachung }
  FOR i := 1 TO codewortlaenge DO                  { Kleinbuchstabe -> Großbuchstabe }
   IF codewort[i] IN [,a'..'z'] THEN
    codewort[i] := chr(ord(codewort[i]) - 32);
  menge := [];
 END;{ codewort kuerzen }

 PROCEDURE schluessel_erstellen;
  VAR i, k: integer;
 BEGIN
  IF codewort <> ,' THEN
   BEGIN
    FOR i := 1 TO codewortlaenge DO
     menge := menge + [codewort[i]];              { Codewort -> Menge ( Vereinfachung) }
    laenge := 26 DIV codewortlaenge;             { Festlegen der Länge des Zuordnungs- }
    IF (26 MOD codewortlaenge) <> 0 THEN                                      { gitters }
     laenge := laenge + 1;
    FOR i := 1 TO codewortlaenge DO              { Schreiben des Codewortes ins Gitter }
     zugriff[1, i] := codewort[i];                                    { (1. Reihe) }
    z := 64;
    FOR i := 2 TO laenge DO                            { Übrige Buchstaben werden ins }
     FOR k := 1 TO codewortlaenge DO                                { Gitter geschrieben }
      BEGIN  z := z + 1;
             IF NOT (chr(z) IN menge)  THEN zugriff[i, k] := chr(z)
                                       ELSE k := k - 1;
      END;{ FOR }
    z := 64;
    FOR k := 1 TO (26 MOD codewortlaenge) DO              { Zuordnung der Buchstaben }
     FOR i := 1 TO laenge DO
      BEGIN z := z + 1;
            speicher[chr(z)] := zugriff[i, k];
      END; { FOR }
    FOR k := (26 MOD codewortlaenge) + 1 TO codewortlaenge DO
     FOR i := 1 TO laenge - 1 DO
      BEGIN z := z + 1;
            speicher[chr(z)] := zugriff[i, k];
      END; { FOR }
   END;
 END; { schluessel_erstellen }
```

```
 PROCEDURE codierung;
  VAR i: integer;
 BEGIN
  IF codewort <> ,‘ THEN
   BEGIN
    FOR i := 1 TO 26 DO
     saver[speicher[chr(i + 64)]] := chr(i + 64);
    FOR i := 1 TO textlaenge DO
     BEGIN
      b := textvorlage[i];
      CASE ord(b) OF
      65..91:
      IF codieren THEN
      b := speicher[b]
      ELSE
      b := saver[b];
      97..123:
      IF codieren THEN
      b := chr(ord(speicher[chr(ord(b) - 32)]) + 32)
      ELSE
      b := chr(ord(saver[chr(ord(b) - 32)]) + 32);
      OTHERWISE
      b := b;
      END;{ CASE }
      textvorlage[i] := b;
     END; { FOR }
    writeln;
    IF codieren THEN
     writeln(,Der codierte Text lautet: ,)
    ELSE
     writeln(,Der decodierte Text lautet: ,);
    writeln;
    text_ausgeben;
    writeln;
   END;
 END; { codierung }

BEGIN { zuordnung }
 textfenster_oeffnen;
 writeln;
 write(,Bitte Codewort eingeben: ,);
 readln(codewort);
 codewort_kuerzen;
 schluessel_erstellen;
 codierung;
END; { zuordnung }
```

```
PROCEDURE vigenere (modus: boolean);
 VAR  i, n, v, passwortlaenge, codezaehler: integer;
      passwortstr: STRING;

 FUNCTION decodierung: STRING;
  CONST N_VOR= 5;
  TYPEtyp1    = RECORD
                       buchst: char;
                       anzahl: integer;
                       END;
  VAR j: integer;
      vorschlag: ARRAY[1..N_VOR] OF STRING;
  PROCEDURE analyse (rest, periode: integer; meist: char);
   VAR haeuf: ARRAY[1..26] OF typ1;
       i, k, v, nr, max: integer;
       vermutung: char;
   PROCEDURE sortieren;
    VAR
     i, k: integer;
     sortHilf: typ1;
   BEGIN
    FOR i := 1 TO 25 DO
     FOR k := i + 1 TO 26 DO
      IF haeuf[k].anzahl > haeuf[i].anzahl THEN
      BEGIN { Tauschen }
      sortHilf := haeuf[i];
      haeuf[i] := haeuf[k];
      haeuf[k] := sortHilf;
      END;
   END; { sortieren }
```

```
BEGIN { analyse }
  FOR i := 1 TO 26 DO
   WITH haeuf[i] DO
    BEGIN
     buchst := char(i + 64);
     anzahl := 0;
    END;
  max := 0;
  FOR i := 1 TO textLaenge DO
   IF i MOD periode = rest THEN
    IF textVorlage[i] IN [,A'..'Z', ,a'..'z'] THEN
     BEGIN
     nr := ord(gross(textVorlage[i])) - 64;
     haeuf[nr].anzahl := haeuf[nr].anzahl + 1;
     IF haeuf[nr].anzahl > max THEN
     Max := haeuf[nr].anzahl;
     END;
  IF (rest = 0) AND (periode = 1) THEN
   BEGIN
    writeln;
    FOR i := 13 TO 26 DO
     BEGIN
     write(haeuf[i].buchst, , : ,);
     FOR k := 1 TO round(haeuf[i].anzahl / max * buAnzahl) DO
     write(,•'); writeln;
     END;
    FOR i := 1 TO 26 DO
     BEGIN
     write(haeuf[i].buchst, , : ,);
     FOR k := 1 TO round(haeuf[i].anzahl / max * buAnzahl) DO
     write(,•'); writeln;
     END;
    writeln;
    write(,Bitte Periodenlänge ablesen:');
    readln(passwortlaenge); writeln;
   END { IF ( rest = 0 ) }
  ELSE
   BEGIN
    sortieren;
    FOR i := 1 TO N_VOR DO
     BEGIN
     v := (25 + ord(haeuf[i].buchst) - ord(meist)) MOD 26 + 1;
     vermutung := chr(v + 64);
     vorschlag[i] := concat(vorschlag[i], vermutung);
     END;
   END;
 END; { analyse }
BEGIN { decodierung }
 FOR j := 1 TO N_VOR DO
  vorschlag[j] := ,';
 analyse(0, 1, ,E');
 FOR j := 1 TO passwortlaenge - 1 DO
  analyse(j, passwortlaenge, ,E');
 analyse(0, passwortlaenge, ,E');
 FOR j := 1 TO N_VOR DO
  writeln(j : 1, ,. Vorschlag Passwort: ,, vorschlag[j]);
 writeln;
 write(,> vermutetes Passwort: ,);
 readln(passwortstr);
 decodierung := passwortstr;
END; { decodierung }
```

```
 PROCEDURE passwortabfrage;
  VAR i: integer;
      hilfs: STRING;
 BEGIN
  hilfs := ,‘;
  writeln(,Bitte geben Sie das Verschlüsselungspasswort ein: ,);
  readln(passwortstr);
  IF (passwortstr = ,‘) THEN
   passwortstr := ,ZZZ‘;
  IF NOT modus AND (passwortstr = ,ZZZ‘) THEN
   passwortstr := decodierung;
  passwortlaenge := length(passwortstr);
  FOR i := 1 TO passwortlaenge DO
   CASE passwortstr[i] OF
    ,A‘..’Z’:
     hilfs := concat(hilfs, passwortstr[i]);
    ,a‘..’z’:
     hilfs := concat(hilfs, gross(passwortstr[i]));
   END; { FOR CASE }
  passwortstr := hilfs;
  passwortlaenge := length(passwortstr);
  writeln(passwortstr);
 END; { Passwortabfrage}

BEGIN { vigenere }
 writeln;
 writeln(,Willkommen zum Vigenere Codierungs bzw. Decodierungsprogramm‘);
 writeln(,Aus dem Passwort werden nur Klein- und Großbuchstaben verwendet.
 Andere Zeichen werden gelöscht.‘);
 writeln;
 passwortabfrage;
 codezaehler := 0;
 FOR i := 1 TO textlaenge DO
  BEGIN
   codezaehler := codezaehler + 1;
   IF codezaehler > passwortlaenge THEN
    codezaehler := 1;
   CASE textvorlage[i] OF
    ,A‘..’Z’, ,a‘..’z’:  v := ord(passWortStr[codezaehler]) - ord(,A‘) + 1;
    ,0‘..’9':       v := (ord(passWortStr[codezaehler]) - ord(,A‘)) MOD 10 + 1;
   END; { CASE }
   CASE textvorlage[i] OF
    ,A‘..’Z’:  n := ord(textvorlage[i]) - ord(,A‘);
    ,a‘..’z’:  n := ord(textvorlage[i]) - ord(,a‘);
    ,0‘..’9':  n := ord(textvorlage[i]) - ord(,0‘);
   END; { CASE }
   CASE textvorlage[i] OF
    ,A‘..’Z’, ,a‘..’z’      :IF modus    THEN n := (n + v) MOD 26
                                         ELSE n := (26 + n - v) MOD 26;
    ,0‘..’9'                :IF modus    THEN  := (n + v) MOD 10
                                         ELSE n := (10 + n - v) MOD 10;
   END; { CASE }
   CASE textvorlage[i] OF
    ,A‘..’Z’:  textvorlage[i] := chr(n + ord(,A‘));
    ,a‘..’z’:  textvorlage[i] := chr(n + ord(,a‘));
    ,0‘..’9':  textvorlage[i] := chr(n + ord(,0‘));
   END; { CASE }
  END; { FOR }
 text_ausgeben;
END; { vigenere }
```

```
PROCEDURE textAnalyse (single: boolean);
 CONST MAX    = 225;
 TYPE    haeuf= ARRAY[1..MAX] OF typ3;
 VAR  aktuelle_Anzahl, max_anzahl, vorher, i  : integer;
      B_haeuf                                 : ARRAY[,A'..'Z'] OF integer;
      haeufigkeit                             : haeuf;
 FUNCTION sortiert_und_reduziert_auf (VAR h: haeuf; typ, fest: integer): integer;
  VAR i, k: integer;
      hilf: typ3;
 BEGIN
  FOR i := 1 TO aktuelle_anzahl - 1 DO
   FOR k := i + 1 TO aktuelle_anzahl DO
    IF h[i].anzahl < h[k].anzahl THEN
     BEGIN { tauschen }
      hilf := h[i];
      h[i] := h[k];
      h[k] := hilf;
     END; { tauschen }
  sortiert_und_reduziert_auf := fest;
  IF fest = 26 THEN
   FOR i := 1 TO 26 DO
    BEGIN
     buchstaben[typ, i].tripel := haeufigkeit[i].tripel;
     buchstaben[typ, i].anzahl := round(haeufigkeit[i].anzahl / gesamt * 1000);
    END;
 END; { sortiert_und_reduziert_auf }

 PROCEDURE histogramm (maxWert: integer);
  VAR i, k, proMille    : integer;
      faktor            : real;
 BEGIN
  faktor := ((konsole.right - konsole.left) / 8 - 12) / maxWert * gesamt / 1000;
  FOR i := 1 TO aktuelle_Anzahl DO
   WITH haeufigkeit[i] DO
    BEGIN
     proMille := round(anzahl / gesamt * 1000);
     write(tripel, ,:', proMille / 10 : 6 : 1, , %  ,);
     FOR k := 1 TO round(proMille * faktor) DO
      BEGIN
      write(,•');
      END;
     writeln;
    END; { WITH haeufigkeit }
 END; { histogramm }

 PROCEDURE titel (ttt: STRING);
 BEGIN
  writeln;
  writeln(ttt);
  writeln;
 END; { titel }
```

```
PROCEDURE BuchstabenHaeufigkeit;
 VAR buchstabe: char;
     i, promille  : integer;
     faktor            : real;
     tri                    : str3;
BEGIN
 tri := ,   ,;
 max_anzahl := 0;
 IF single THEN
  gesamt := 0;
 FOR buchstabe := ‚A‘ TO ‚Z‘ DO
  IF single THEN
   B_haeuf[buchstabe] := 0
  ELSE
   BEGIN
    i := round(buchstaben[1, ord(buchstabe) - 64].anzahl / 1000 * gesamt);
    B_haeuf[buchstaben[1, ord(buchstabe) - 64].tripel[1]] := i;
   END;
 FOR i := 1 TO textLaenge DO
  BEGIN
   IF textVorlage[i] IN [‚A‘..’Z’, ‚a‘..’z’] THEN
    BEGIN
     gesamt := gesamt + 1;
     buchstabe := gross(textVorlage[i]);
     B_haeuf[buchstabe] := B_haeuf[buchstabe] + 1;
    END;
  END;
 FOR i := 1 TO 26 DO
  BEGIN
   tri[1] := chr(64 + i);
   haeufigkeit[i].anzahl := B_haeuf[chr(64 + i)];
   haeufigkeit[i].tripel := tri;
   IF haeufigkeit[i].anzahl > max_anzahl THEN
    max_anzahl := haeufigkeit[i].anzahl;
  END;
END; { BuchstabenHaeufigkeit }

FUNCTION erfasse (typ, nn: integer): integer;
 VAR i       : integer;
     tri     : str3;
BEGIN
 tri := ,   ,;
 FOR i := 1 TO typ DO
  tri[i] := gross(textvorlage[nn + i - typ]);
 i := 0;
 IF aktuelle_anzahl = 0 THEN BEGIN    i := i + 1;
                                  aktuelle_anzahl := i;
                                  haeufigkeit[i].tripel := tri;
                                  haeufigkeit[i].anzahl := 1;
                                  END { THEN }
 ELSE  BEGIN    REPEAT i := i + 1;
                UNTIL (haeufigkeit[i].tripel = tri) OR (i = aktuelle_anzahl);
 IF haeufigkeit[i].tripel = tri THEN
     haeufigkeit[i].anzahl := haeufigkeit[i].anzahl + 1 ELSE
    BEGIN    i := i + 1;
             aktuelle_anzahl := i;
             haeufigkeit[i].tripel := tri;
             haeufigkeit[i].anzahl := 1;
    END;
  END; { ELSE }
 erfasse := aktuelle_anzahl;
END;  { erfasse }
```

```
 FUNCTION uebernehmen_von (bbb: hist): integer;
  VAR i, www: integer;
 BEGIN
  FOR i := 1 TO 26 DO
   BEGIN
    www := round(bbb[i].anzahl / 1000 * vorher);
    haeufigkeit[i].anzahl := www;
    haeufigkeit[i].tripel := bbb[i].tripel;
   END;
  uebernehmen_von := 26;
 END; { uebernehmen_von }

BEGIN { textAnalyse }
 vorher := gesamt;
 showText;
 buchstabenhaeufigkeit;
 writeln;
 writeln(,-----‘, gesamt : 5, , Buchstaben ------‘);
 writeln;
 pause(60);
 titel(,----- Buchstaben unsortiert ------‘);
 aktuelle_anzahl := 26;
 histogramm(max_anzahl);
 aktuelle_anzahl := sortiert_und_reduziert_auf(haeufigkeit, 1, 26);
 writeln;
 writeln(,••••• KLICK to CONTINUE •••••‘);
 REPEAT
 UNTIL button;
 writeln;
 titel(,----- Bigramme sortiert ------‘);
 IF single THEN
  aktuelle_anzahl := 0
 ELSE
  aktuelle_anzahl := uebernehmen_von(buchstaben[2]);
 FOR i := 2 TO textlaenge - 1 DO
  BEGIN
   aktuelle_anzahl := erfasse(2, i);
   IF aktuelle_anzahl >= max THEN
    aktuelle_anzahl := sortiert_und_reduziert_auf(haeufigkeit, 2, MAX DIV 3);
  END;
 aktuelle_anzahl := sortiert_und_reduziert_auf(haeufigkeit, 2, 26);
 histogramm(haeufigkeit[1].anzahl);

 titel(,----- Trigramme sortiert ------‘);
 IF single THEN
  aktuelle_anzahl := 0
 ELSE
  aktuelle_anzahl := uebernehmen_von(buchstaben[3]);
 FOR i := 3 TO textlaenge - 1 DO
  BEGIN
   aktuelle_anzahl := erfasse(3, i);
   IF aktuelle_anzahl >= max THEN
    aktuelle_anzahl := sortiert_und_reduziert_auf(haeufigkeit, 3, MAX DIV 3);
  END;
 aktuelle_anzahl := sortiert_und_reduziert_auf(haeufigkeit, 3, 26);
 histogramm(haeufigkeit[1].anzahl);
END; { textAnalyse }
```

```
PROCEDURE interAktiv;
 CONSTRD = 25;
        TS = 18;
 VAR  ueber, alt, neu, altab, neuab, text, ersetzen, feld, stop :rect;
      r1, r2, r3: rect;
      y0, y1, y2, y3, links, breit: integer;
      zuordnung: ARRAY[,A‘..’Z’] OF char;
      halt: boolean;
PROCEDURE hinweis (wo: rect; was: str255; farbe, groesse: integer);
 BEGIN
  IF (farbe = whiteColor) THEN
   eraseRect(wo)
  ELSE
   BEGIN
    foreColor(farbe);
    textsize(groesse);
    textFace([bold]);
    textFormatieren(was, wo, TeCenter)
   END
 END; { hinweis }

 PROCEDURE Bild_zeichnen;
  VAR c: char;
  PROCEDURE zeichne_Rechteck (VAR name: rect;
                        li, ob, re, un, Hfarbe, r, Tfarbe: integer; titel: STRING);
  BEGIN
   forecolor(Hfarbe);
   setrect(name, li, ob, re, un);
   paintRoundrect(name, r, r);
   forecolor(blackcolor);
   frameRoundrect(name, r, r);
   Moveto(li + 5, ob + 20);
   forecolor(Tfarbe);
   writedraw(titel);
   pensize(1, 1);
  END; { zeichne_Rechteck }
  PROCEDURE beschrifte_mit (x, y: integer; titel: str255);
  BEGIN
   forecolor(blackColor);
   moveto(x, y);
   writeDraw(titel);
  END; { beschrifte_mit }
```

```
BEGIN { Bild_zeichnen }
 WITH lokal DO
  BEGIN
   zeichne_Rechteck(alt, left + RD, top + RD, left + RD + 75, top + RD + 25, cyanColor,
   0, blackcolor, ‚alt‘);
   zeichne_Rechteck(neu, left + RD, top + RD + 35, left + RD + 75, top + RD + 60,
   redColor, 0, blackcolor, ‚neu‘);
   zeichne_Rechteck(text, left + RD, top + RD + 70, left + RD + 75, top + RD + 95,
   yellowColor, 0, blackcolor, ‚Text‘);
   zeichne_Rechteck(ersetzen, right - RD - 80, top + RD, right - RD + 5, top + RD + 25,
   magentaColor, 20, blackcolor, ‚change‘);
   zeichne_Rechteck(stop, right - RD - 75, top + RD + 35, right - RD, top + RD + 60,
   redcolor, 20, whitecolor, , STOP‘);
   zeichne_Rechteck(feld, left + RD, top + RD + 105, right - RD, bottom - RD,
   yellowColor, 0, blackcolor, ‚‘);
   zeichne_Rechteck(altab, left + RD + 85, top + RD, right - RD - 85, top + RD + 25,
   blueColor, 20, bluecolor, ‚‘);
   zeichne_Rechteck(neuab, left + RD + 85, top + RD + 35, right - RD - 85, top + RD +
   60, blueColor, 0, bluecolor, ‚‘);
   zeichne_Rechteck(ueber, left + RD + 85, top + RD + 70, right - RD - 85, top + RD +
   95, greenColor, 0, blueColor, ‚‘);
   hinweis(ueber, ‚Interaktive Entschlüsselung‘, greenColor, TS);
   hinweis(ersetzen, , ,, whiteColor, TS);
  END;
 FOR c := ‚A‘ TO ‚Z‘ DO
  zuordnung[c] := ‚•‘;
 y0 := feld.top;
 y3 := feld.bottom;
 y1 := 2 * y0 DIV 3 + 1 * y3 DIV 3;
 y2 := 1 * y0 DIV 3 + 2 * y3 DIV 3;
 beschrifte_mit((xm + feld.left) DIV 2 - 3 * TS, feld.bottom + TS, ‚links‘);
 beschrifte_mit((xm + feld.right) DIV 2 - 3 * TS, feld.bottom + TS, ‚rechts‘);
 beschrifte_mit(feld.left - TS, (y0 + y1) DIV 2, ‚1‘);
 beschrifte_mit(feld.right + 5, (y0 + y1) DIV 2, ‚1‘);
 beschrifte_mit(feld.left - TS, (y1 + y2) DIV 2, ‚2‘);
 beschrifte_mit(feld.right + 5, (y1 + y2) DIV 2, ‚2‘);
 beschrifte_mit(feld.left - TS, (y2 + y3) DIV 2, ‚3‘);
 beschrifte_mit(feld.right + 5, (y2 + y3) DIV 2, ‚3‘);
 breit := (altab.right - altab.left) DIV 26;
 links := altab.left + 5;
 foreColor(cyanColor);
 WITH altab DO
  FOR c := ‚A‘ TO ‚Z‘ DO
   BEGIN
    moveto(links + breit * (ord(c) - 65), 8 + (bottom + top) DIV 2);
    writeDraw(c);
   END;
 foreColor(redColor);
 WITH neuab DO
  FOR c := ‚A‘ TO ‚Z‘ DO
   BEGIN
    moveto(links + breit * (ord(c) - 65), 8 + (bottom + top) DIV 2);
    writeDraw(,•‘);
   END;
END;{ BildZeichnen }
```

```
PROCEDURE interAktion;
 VAR altPosi, neuPosi: point;
     altBuch, neuBuch: char;
 PROCEDURE statistik (wo: rect; was: integer);
  VAR
   i, dh, dv, dx, d2, li, mi, mm, maxi: integer;
   faktor: real;
   balken: rect;
 BEGIN
  IF buchstaben[was, 1].anzahl > standard[was, 1].anzahl THEN
   maxi := buchstaben[was, 1].anzahl
  ELSE
   maxi := standard[was, 1].anzahl;
  dh := (wo.bottom - wo.top) DIV 28;
  dv := (wo.right - wo.left) DIV 2;
  d2 := dh DIV 2 - 2;
  li := wo.left + 30;
  mi := li + dv;
  mm := wo.top - 4;
  faktor := (dv - 40) / maxi;
  textFace([]);
  textSize(9);
  foreColor(yellowColor);
  paintRect(wo);
  foreColor(blueColor);
  moveto(wo.left + 3, wo.top + dh + 3);
  CASE was OF
   1: writeDraw(,Buchst. Standard <-> TextVorlage‘);
   2: writeDraw(,Paare   Standard <-> TextVorlage‘);
   3: writeDraw(,Tripel  Standard <-> TextVorlage‘);
  END; { CASE }
  FOR i := 1 TO 26 DO
   BEGIN
    moveto(wo.left + 3, wo.top + (i + 2) * dh);
    dx := round(faktor * standard[was, i].anzahl);
    writeDraw(standard[was, i].tripel);
    setRect(balken, li, mm + (i+2) * dh - d2, li + dx, mm + (i+2) * dh + d2);
    paintRect(balken);
   END;
  FOR i := 1 TO 26 DO
   BEGIN
    moveto(wo.left + 3 + dv, wo.top + (i + 2) * dh);
    dx := round(faktor * buchstaben[was, i].anzahl);
    writeDraw(buchstaben[was, i].tripel);
    setRect(balken, mi, mm + (i+2) * dh - d2, mi + dx, mm + (i+2) * dh + d2);
    paintRect(balken);
   END;
  textSize(TS);
  textface([bold]);
 END; { statistik }
```

```
PROCEDURE warten_bis_klick_in (ziel: rect; VAR ccc: char; VAR ppp: point);
 VAR punkt: point;
     sBox: rect;
     stat, wahl: boolean;
     lang, xAlt, xNeu, yAlt, yNeu: integer;
BEGIN
 IF NOT halt THEN
  BEGIN
   stat := FALSE;
   wahl := FALSE;
   lang := TS DIV 2 - 2;
   xAlt := 0;
   yAlt := 0;
   setPt(punkt, 0, 0);
   REPEAT
    getmouse(punkt);
    halt := ptInRect(punkt, stop);
    stat := ptInRect(punkt, feld);
    IF stat AND keypressed THEN
    BEGIN
    stat := FALSE;
    foreColor(blackColor);
    xNeu := punkt.h DIV TS * TS + lang DIV 2 + 3;
    yNeu := (punkt.v + lang) DIV TS * TS - 3 * lang DIV 2 + 1;
    IF (xAlt <> 0) AND (yAlt = yNeu) THEN line(xNeu - xAlt - 6, 0)
    ELSE         BEGIN
                 xAlt := 0;
                 yAlt := 0;
            END;
    moveto(xNeu, yNeu);
    line(0, 2 * lang);
    xAlt := xNeu;
    yAlt := yNeu;
    END;
    wahl := ptInRect(punkt, ziel);
   UNTIL button AND (halt OR stat OR wahl);
   REPEAT
   UNTIL NOT button;
   IF stat THEN
    BEGIN
    sBox := feld;
    IF punkt.h < xm    THEN sBox.right := xm
                       ELSE sBox.left := xm;
    insetRect(sBox, 1, 1);
    IF punkt.v < y1    THEN statistik(sBox, 1)
                       ELSE IF punkt.v > y2     THEN statistik(sBox, 3)
                                                ELSE statistik(sBox, 2);
    warten_bis_klick_in(ziel, ccc, ppp);
    END;{ IF stat }
   IF wahl THEN
    BEGIN
     ccc := chr((punkt.h - links) DIV breit + 65);
     WITH ziel DO
     setPt(ppp, links + breit * (ord(ccc)-65), 8 + (bottom + top) DIV 2)
    END; { IF wahl }
  END; { IF NOT halt }
END; { warten_bis_klick_in }
```

```
PROCEDURE buchstaben_ersetzen (buch: char; posi: point; farbe: integer);
 VAR loesch: rect;
BEGIN
 IF NOT halt THEN
  BEGIN
   setRect(loesch, posi.h - 1, posi.v - TS + 3, posi.h + TS - 3, posi.v + 3);
   foreColor(blueColor);
   paintRect(loesch);
   foreColor(farbe);
   moveto(posi.h, posi.v);
   writeDraw(buch);
   foreColor(blackColor);
  END;
END; { buchstaben_ersetzen }
PROCEDURE text_anzeigen;
 VAR
  i, z, s, nz, ns: integer;
  bMenge: SET OF char;
  c: char;
BEGIN
 bMenge := [,A‘..’Z’, ,a‘..’z’];
 foreColor(yellowColor);
 paintRect(feld);
 foreColor(blackColor);
 frameRect(feld);
 textface([]);
 ns := (feld.right - feld.left - TS) DIV TS;
 nz := (feld.bottom - feld.top) DIV TS;
 FOR z := 1 TO nz DO
  FOR s := 1 TO ns DO
   BEGIN
    moveto(feld.left + s * TS, feld.top + z * TS);
    i := (z - 1) * ns + s;
    textface([]);
    foreColor(cyanColor);
    IF i > textLaenge THEN
    writeDraw(,-‘)
    ELSE IF NOT (textVorlage[i] IN bMenge) THEN
    writeDraw(textVorlage[i])
    ELSE
    BEGIN
    c := zuordnung[gross(textVorlage[i])];
    IF c = ,•‘ THEN
    writeDraw(textVorlage[i])
    ELSE
    BEGIN
    foreColor(redColor);
    textface([bold]);
    writeDraw(zuordnung[gross(textVorlage[i])])
    END;
    END;
   END;
END; { text_anzeigen }
```

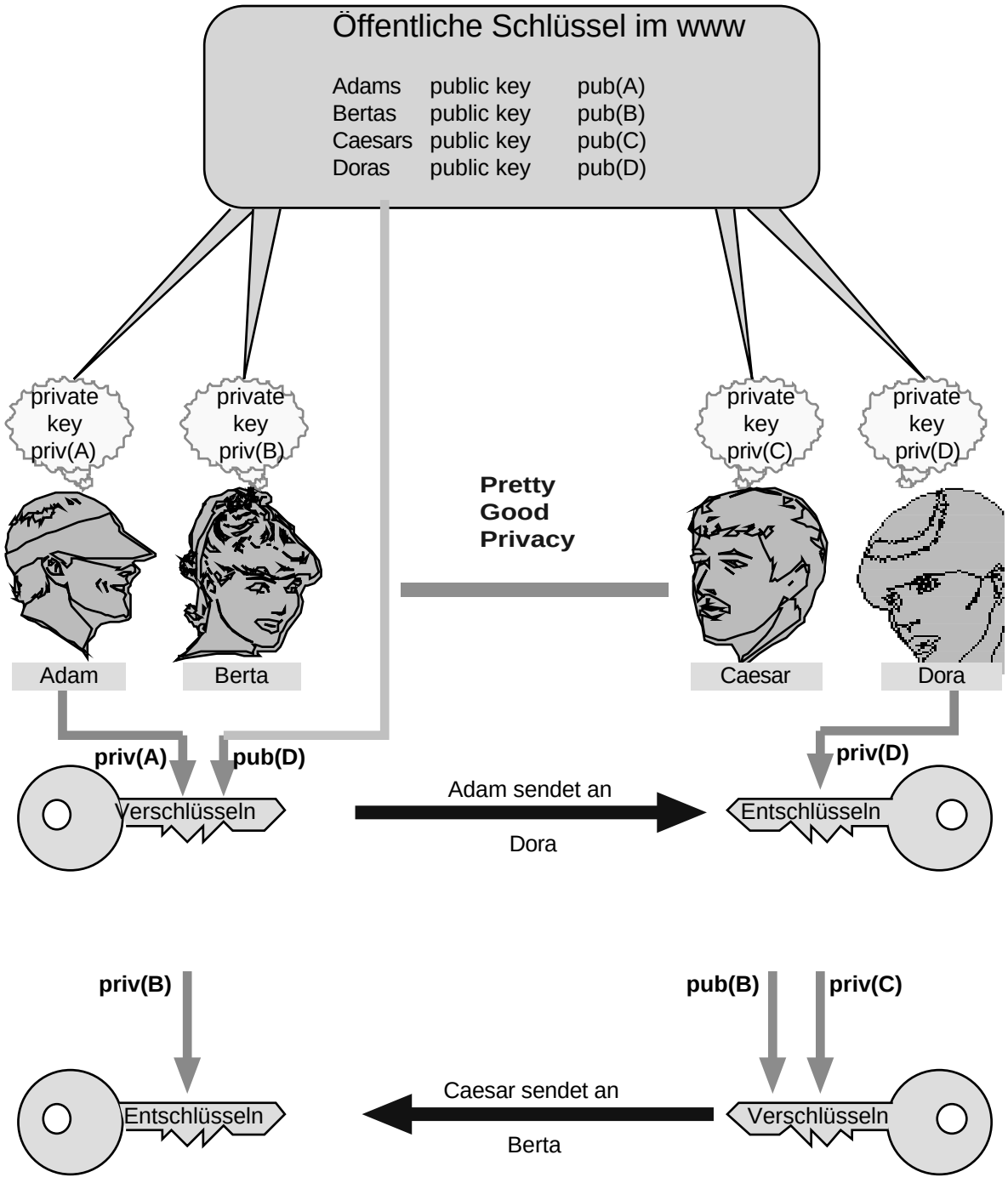
```
 BEGIN { Interaktion }
  hinweis(ersetzen, ‚CHOOSE‘, cyanColor, TS);
  warten_bis_klick_in(altab, altBuch, altPosi);
  buchstaben_ersetzen(altBuch, altPosi, redColor);
  hinweis(ersetzen, ‚CHANGE‘, redColor, TS);
  warten_bis_klick_in(altab, neuBuch, neuPosi);
  neuPosi.h := altPosi.h;
  neuPosi.v := altPosi.v + 35;
  buchstaben_ersetzen(altBuch, altPosi, cyanColor);
  buchstaben_ersetzen(neuBuch, neuPosi, redColor);
  zuordnung[altbuch] := neuBuch;
  text_anzeigen;
 END; { interAktion }

BEGIN { interAktiv }
 textAnalyse(TRUE);
 grafikfenster_oeffnen;
 showdrawing;
 textSize(TS);
 textFont(Monaco);
 textface([bold]);
 Bild_zeichnen;
 halt := FALSE;
 REPEAT
  interAktion
 UNTIL halt;
END; { interAktiv }

PROCEDURE standard_speichern;
 VAR  datei: FILE OF typ3;
       satz: typ3;
       i, k: integer;
BEGIN
 bereinigt_laden;
 textAnalyse(TRUE);
 bereinigt_laden;
 textAnalyse(FALSE);
 bereinigt_laden;
 textAnalyse(FALSE);
 rewrite(datei, newFileName(‚f:‘, ‚Standard‘));
 FOR i := 1 TO 3 DO
  FOR k := 1 TO 26 DO
   write(datei, buchstaben[i, k]);
 close(datei);
END; { standard_speichern }

PROCEDURE Menue_bearbeiten (Info: longint);
 VAR  menuNumm, menuItem: integer;
 PROCEDURE handleApple (dasItem: integer);
  VAR
   accNAme: str255;
   accNumber, itemNumber, dummy: integer;
   appleMenu: menuHandle;
  PROCEDURE infos;
   VAR
    dummy: integer;
  BEGIN
   dummy := Alert(RES_ID_NR, NIL);
  END; { infos }
```

```
 BEGIN { handleApple  }
  CASE dasItem OF
   1:  infos;
   OTHERWISE
    BEGIN
     appleMenu := getMHandle(RES_ID_NR);
     getItem(appleMenu, dasItem, accName);
     accNumber := openDeskacc(accName);
    END;
  END; { CASE }
 END; { handleApple }

BEGIN { Menue_bearbeiten }
 IF info <> 0 THEN
  BEGIN
   IF menu_markiert(4, 4) THEN
    item_aendern(4, 4);
   PenNormal;
   MenuNumm := Hiword(info) - RES_ID_NR + 1;
   menuItem := LoWord(info);
    CASE MenuNumm OF
    1:       handleApple(menuItem);                                    { APPLE }

    2:       CASE menuItem OF                                         { Ablage }
             1:   BEGIN original_laden; dateiname := concat(dateiname, ,.O‘);
                  END;
             2:   BEGIN bereinigt_laden; dateiname := concat(dateiname, ,.B‘);
                  END;
             4:   sichern;
             6: ende := true;
             END; { CASE menuItem 2 }

    3:       CASE menuItem OF                                     { Codierung }
             1:   BEGIN dateiname := concat(dateiname, ,C‘); caesar(true);
                  END;
             2:   BEGIN dateiname := concat(dateiname, ,Z‘); zuordnung(true);
                  END;
             3:   BEGIN dateiname := concat(dateiname, ,V‘); vigenere(true);
                  END;
             END; { CASE menuItem 3 }

       4:         CASE menuItem OF                                 { Dekodierung }
             1:   BEGIN dateiname := concat(dateiname, ,c‘); caesar(false);
                  END;
             2:   BEGIN dateiname := concat(dateiname, ,z‘);zuordnung(false);
                  END;
             3:   BEGIN dateiname := concat(dateiname, ,v‘); vigenere(false);
                  END;
             END;{ CASE menuItem 4 }

       5:    CASE menuItem OF                                        { Werkzeuge }
             1:   BEGIN dateiname := concat(dateiname, ,t‘); textAnalyse(TRUE);
                  END;
             3:   BEGIN dateiname := concat(dateiname, ,i‘); interAktiv;
             END;
       END;{ CASE menuItem 4 }

    END; { CASE MenuNumm }

   hiliteMenu(0);

   END; { IF }

END; { Menue_bearbeiten }
```

```
PROCEDURE behandle_Klick_auf (VAR pkt: point);
BEGIN
 globalToLocal(pkt);
END; { behandle_Klick_auf }

PROCEDURE behandle_Maus (event: EventRecord);
 VAR  mfenster: windowPtr;
      FensterNo: Integer;
      mausPunkt: Point;
BEGIN
 mausPunkt := event.Where;
 FensterNo := FindWindow(Mauspunkt, mfenster);
 CASE FensterNo OF
  InMenuBar:
   Menue_bearbeiten(menuSelect(mauspunkt));
  inContent:
   behandle_Klick_auf(mausPunkt);
  inGoAway:
   hideWindow(mFenster);
  OTHERWISE ;
 END; { CASE }
END; { behandle_Maus }

PROCEDURE behandle_Tasten (event: EventRecord);
 VAR  key: char;
BEGIN
 WITH event DO
  BEGIN
   key := chr(BitAnd(message, charCodeMask));
   IF BitAnd(modifiers, cmdKey) <> 0 THEN
    BEGIN
     IF key IN [,a'..'z'] THEN
      key := chr(ord(key) - 32);
     Menue_bearbeiten(menuKey(key));
    END; { IF }
  END { WITH }
END; { behandle_Tasten }

PROCEDURE Menue_installieren;
 VAR  hauptMenu    : handle;
      einMenu : menuHandle;
      item              : integer;
BEGIN
 InitMenus;
 hauptMenu := getNewMBar(RES_ID_NR);
 setMenuBar(hauptMenu);
 disposHandle(hauptMenu);
 einMenu := getMHandle(RES_ID_NR);
 addResMenu(einMenu, ,DRVR');
 drawMenuBar;
 menu_zeigen(2, 4, false);
 menu_zeigen(3, 0, false);
 menu_zeigen(4, 0, false);
 menu_zeigen(5, 0, false);
END; { Menue_installieren }
```

```
 PROCEDURE Ereignis_behandeln;
  VAR  gefunden: boolean;
 BEGIN
  IF MultiFinder THEN
   BEGIN
    gefunden := waitNextEvent(everyEvent, ereignis, 0, NIL);
   END
  ELSE
   BEGIN
    systemTask;
    gefunden := getNextEvent(everyEvent, ereignis);
   END; { ELSE }
  IF gefunden THEN
   CASE ereignis.what OF
    mouseDown:
     behandle_Maus(ereignis);
    keyDown, autoKey:
     behandle_Tasten(ereignis);
    updateEvt:
     BEGIN
      BeginUpdate(windowptr(ereignis.message));
      EndUpdate(windowptr(ereignis.message));
     END; { udateEvt }
    diskEvt:
     BEGIN
     END; { diskEvt }
    activateEvt:
     BEGIN
      drawGrowIcon(windowPtr(ereignis.message));
     END; { activateEvt }
   END; { CASE }
 END; { Ereignis_behandeln }

 PROCEDURE Ereignis_Schleife;
 BEGIN
  ende := false;
  MultiFinder := (NGetTrapAddress($60, toolTrap) <> nGetTrapAddress($9F, toolTrap));
  WHILE NOT ende DO
   BEGIN
    Ereignis_behandeln;
   END; { WHILE }
 END; { Ereignis_Schleife }



BEGIN { Hauptprogramm }

 Menue_installieren;
 Anfangsbelegungen;

 IF STANDARD_ERFASSEN THEN
  standard_speichern;

 Ereignis_Schleife;

END.
```

Anlage 9 : PGP-Algoritmus



Berta decodiert mit ihrem privaten Schlüssel

Kodierung mit dem **RSA** - Algoritmus,

benannt nach Ronald **R** ivest
Adi **S** hamir
Leonard **A** dleman

Literaturempfehlung:
Albrecht Beutelsbacher, Kryptologie

Caesar verschlüsselt mit dem öffentlichen Schlüssel des Empfängers Berta

und signiert mit seinem privaten Schlüssel.