Dr. Michael J. Winckler
IWR, Raum 302
INF 368
69120 Heidelberg
Michael.Winckler@iwr.uni-heidelberg.de
http://www.iwr.uni-heidelberg.de/teaching/Mathe-Star/

Arbeitsblatt 05 *Vertiefung: Character und Strings* (Version 1.10)

Theorie

Übungsziele: Ziel des Arbeitsblattes ist es, die Verwaltung von einzelnen Zeichen (Variablen vom Typ `char`), Zeichenfolgen (`char`-Arrays) und Zeichenketten (Strings bzw. \0-terminierte `char`-Arrays) im C-Code zu verstehen.

Zeichen werden computerintern auf Basis einer Zeichentabelle verwaltet, die jedem Zeichen einen Zahlwert zuordnet. Für die Darstellung auf dem Bildschirm wird dann aus diesem Zahlwert wieder ein Pixelmuster (Bilddarstellung des jeweiligen Zeichens) auf Basis eines Tabellen-Lookup generiert. Die interne Darstellung der Einzelzeichen hat also zunächst nichts mit dem Verständnis zu tun, das wir von diesem Zeichen haben.

Die Übersetzung zwischen Zahlwert und Zeichen erfolgt auf IT-Systemen anhand von Zeichensatztabellen, deren bekannteste die im amerikanischen Industriestandard festgelegten sind (ASCII). In den ASCII-Tabellen ist in den unteren 7 Bit (Zeichen 0-127) der Kernzeichensatz festgelegt, während die Zeichen in der oberen Hälfte (128-255 → Character werden in der Regel in einem Bit gespeichert) regional unterschiedlich sind (Umlaute etc.)

Übungen

Übung 5.1: Zahldarstellung von Zeichen

In der internen Darstellung werden einzelne Zeichen mit dem Datentyp `char` verwaltet, der ein Byte gross ist. In ANSI-C ist die diesem Datentyp entsprechende Zahldarstellung nicht vorzeichenbehaftet (`unsigned`), aber manche modernen Compiler lassen trotzdem den Spezifizierer `unsigned` für diesen Datentyp zu.

Aufgrund der Ambivalenz zwischen Zahlwert der Speicherstelle einerseits und Zeichendarstellung andererseits kann man Character-Variablen mit unterschiedlicher Ausdruckssyntax einen Wert zuordnen.

1. Schreibe ein Programm, das zwei `char`-Speicherstellen `c_a` und `c_b` deklariert und weise der einen mit

   ```
   c_a = 'f';
   ```

   und der anderen mit

   ```
   c_b = 72;
   ```

   einen Zahl- beziehungsweise Zeichenwert zu. Verwende die Formatspezifizierer `%d` und `%c`, um von beiden Variablen beide Darstellungen auf den Bildschirm auszugeben:

   ```
   Zahlinhalt der Variablen    c_a: 70
   Zeicheninhalt der Variablen c_a:  f
   ...
   ```

2. Erweitere dein Programm, indem du nach erfolgter Ausgabe den Inhalt der beiden Variablen mit dem unären Operator `--` um 1 verkleinerst und gib danach beide Darstellungen nochmals aus.

3. Lagere die Ausgabe einer Character-Variablen in beiden Darstellungen (inklusive Hilfssatz) in eine Unterroutine aus.

Übung 5.2: Caesar-Code

Schon Julius Caesar hat zur Verschlüsselung seiner Feldpost auf Heereszügen eine einfache Codierung verwendet: Auf Basis einer Tabelle hat er in Nachrichten jeden Buchstaben systematisch durch einen anderen ersetzt. Im folgenden wollen wir eine einfache Variante des Caesar-Codes selbst programmieren: rot13. Im rot13-Code wird jeder Buchstabe durch den um 13 Buchstaben im Alphabet weiter hinten liegenden Buchstaben ersetzt, wobei bei Erreichen des Buchstabens Z bei A weitergezählt wird. Es ergibt sich folgende Tabelle:

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

Klartext	T	U	V	W	X	Y	Z
Code	G	H	I	J	K	L	M

Wie man an dieser Tabelle sehr schön sieht, ist dieser Code symmetrisch: wenn man einen codierten Text weider entschlüsseln will, codiert man ihn einfach nochmals!

Wenn wir den Codierungsvorgang analysieren, ergibt sich folgender algorithmische Ablauf für die Durchführung eines Codierungsschritts:

1. Ermittle, ob es sich um einen Gross- oder Kleinbuchstaben handelt. Setze das entsprechende Basiszeichen auf 'A' bzw. 'a'.
2. Berechne den Abstand zum Basiszeichen. Dies ist eine Zahl zwischen 0 und 25 (einschliesslich).
3. Addiere zu dem erhaltenen Wert 13.
4. Wenn die Zahl grösser als 25 ist, ziehe 26 ab.
5. Ermittle das Zeichen, das dem angegebenen Wert entsprechend weit vom jeweiligen Basiszeichen entfernt ist. Dies ist der verschlüsselte Buchstabe.

Um diesen Algorithmus umzusetzen, helfen uns einige Eigenheiten der C-Programmierung und des verwendeten ASCII-Zeichensatzes.

1. Im ASCII-Zeichensatz stehen die Gross- und Kleinbuchstaben zwar an auseinanderliegenden Stellen, folgen aber jeweils in der Anordnung des Alphabets aufeinander. Daher kann man den z.B. testen, ob eine Zeichenvariable `c` einen Kleinbuchstabe repräsentiert, indem man folgende Konstruktion verwendet:

```
if (c >='a' && c <= 'z'){
  ...
}
```

2. Da aufeinanderfolgende Zahlen aufeinaderfolgenden Buchstaben entsorechen, kann man den Abstand einer `char`-Variable `c` vom Zeichen `'a'` durch

```
diff = c-'a';
```

   berechnen.

3. Will man eine Zahl auf den Bereich $0, \ldots, 26$ beschränken (mit Neubeginn bei 0, wenn der Bereich bei 25 überschritten wird), so eignet sich für diese Aufgabe der Modulo-Operator %.

Implementiere auf Basis dieser Hinweise ein Programm, das auf eine Zeichenkette mit den rot13-Operator anwendet. Beachte dabei, daß nur *Buchstaben* verändert werden sollen und daß eine Zeichenkette durch das \0-Zeichen terminiert wird.

Übung 5.3: Erkennen von Wörten mit Buchstabenmix

Aus einer Email:

```
------------ Begin Forwarded Message -------------

Nach eienr Stidue der Cmabridge Uinverstiaet, ist es eagl in wlehcer
Reiehnfogle die Bchustebaen in Woeretrn vokrmomen. Es ist nur withcig,
dsas der ertse und lettze Bchusatbe an der ricthgien Stlele snid. Der Rset
knan total falcsh sein und man knan es onhe Porbelme leesn. Das ist, wiel
das mneschilche Geihrn nciht jeden Bchustbaen liset sodnern das Wrod als
gaznes. Krsas oedr?

------------ End Forwarded Message -------------
```

Schreibe ein Programm, das diesen Sachverhalt testet: Ausgehend von einem Text mit mehreren Worten soll es einen neuen Text generieren, in dem Satzzeichen und die Anfangs- und Endbuchstaben aller Worte unverändert übernommen werden, während die Buchstaben im *Wortinneren* beliebig permutiert werden.

Entwirf für diesen Permutationsoperator eine Subroutine, die diese Aufgabe für *ein Wort* leistet. Beachte dabei, dass auch Teilstrings von \0 terminiert werden müssen.

Im obigen Beispiel wurden übrigens nur paarweise manchmal Buchstaben im Wort vertauscht. Dieser beschränkte Vertauschoperator produziert keine stark verfremdeten Wörter, sondern erhält grosse Teile der Wortstruktur. Implementiere auch diesen Operator und stelle das Ergebnis beider Operatoren gegenüber.

Extraarbeit: Versieh dein Programm mit einem kleinen DOS-Menü, lass den Text (max. 500 Zeichen) einlesen und gib auf Wunsch verschiedenen Varianten aus.