1. Aufgabe: Verschlüsselung

In dieser Praktikumsaufgabe soll ein beliebiger Text verschlüsselt werden. Wir beginnen als erstes mit einem kleinen Programm zum Auswerten der übergebenen Parameter, führen in einem zweiten Schritt eine Verschlüsselung durch, die nach Julius Cäsar benannt wurde. Schließlich soll als drittes die Verschlüsselung für beliebige Schlüsselwörter erweitert werden.

1 Kommandozeilenparameter

Schreiben Sie ein Programm, das alle übergebenen Kommandozeilenparameter auf dem Bildschirm ausgibt. Es gelten folgende Vorgaben:

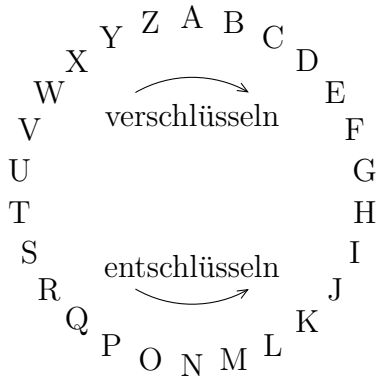
- Das Programm soll *parameter* heißen.
- Jeder Parameter soll in einer eigenen Zeile ausgegeben werden.
- Die Parameter sollen in Anführungszeichen ausgegeben werden.

Wenn Sie Ihr Programm mit „*parameter Guten Tag 123*“ aufrufen, soll sich folgende Ausgabe ergeben:

```
Parameter 0: "parameter"
Parameter 1: "Guten"
Parameter 2: "Tag"
Parameter 3: "123"
```

2 Einfache Verschlüsselung von Textdateien

Es soll ein Programm zum Verschlüsseln von Textdateien nach dem Prinzip von Julius Cäsar erstellt werden.
Für das Prinzip von Cäsar stelle man sich die Buchstaben des Alphabets der Reihe nach im Uhrzeigersinn auf einem Kreis angeordnet vor. Für einen unverschlüsselten Buchstaben suche man sich den entsprechenden Buchstaben im Kreis, gehe um eine feste Schrittweite im Kreis weiter, und verwende den neuen Buchstaben als verschlüsselten Buchstaben. Z.B. wird bei einer Verschiebung von 5 aus einem 'A' ein 'F', aus einem 'B' ein 'G', aus einem 'X' ein 'C' u.s.w.



Für die Dekodierung wird der gleiche Mechanismus mit einer passenden Verschiebung erneut angewandt. Wurde z.B. mit 5 kodiert, so muss mit -5, bzw. 21 $(26 - 5)$ dekodiert werden.

Beispiel: Bei einer Verschiebung im Alphabet um 5 wird aus „HALLO“ das Wort „MFQQT“. Bei einer zweiten Verschlüsselung, diesmal um den Wert 21 wird aus „MFQQT“ wieder „HALLO“.

Das Programm soll folgende Eigenschaften haben:

- Es soll *encrypt1* heißen.
- Es wird wie folgt aufgerufen:
  encrypt1 <Eingabedateiname> <Ausgabedateiname> <Verschiebung>
  Beispiel: `encrypt1 Beispiel1.txt Kodiert1.txt 3`

- Der Wertebereich der Verschlüsselung soll auf den Bereich 1 bis 25 beschränkt werden.
- Kleinbuchstaben werden vor der Verschlüsselung in Großbuchstaben umgewandelt.
- Alle anderen Zeichen, inkl. der Umlaute und dem ß, werden nicht verschlüsselt.
- Mögliche Fehler sollen abgefangen und mit Fehlermeldungen versehen werden.
- Erstellen und verwenden Sie innerhalb des Programms eine Funktion mit Namen *encrypt* zum Ver- bzw. Entschlüsseln des Textes. Die Funktion bekommt die geöffneten Dateien und die Verschiebung im Alphabet als Parameter übergeben und führt die Verschlüsselung des ganzen Textes durch.
- Das Hauptprogramm *main* soll die übergebenen Parameter überprüfen, die Dateien öffnen, die Funktion *encrypt* aufrufen und am Ende die Dateien wieder schließen.

3 Verbesserte Verschlüsselung von beliebigen Dateien

Erstellen Sie eine verbesserte Version des vorherigen Programms. Zum Verschlüsseln soll statt eines konstanten Wertes ein ganzes Schlüsselwort verwendet werden. Es sollen alle Zeichen des Textes, lesbar und nicht-lesbar, verschlüsselt werden (Werte von 0 bis 255).

In der vorigen Aufgabe konnten wir uns die 26 Buchstaben des Alphabets als einen Kreis vorstellen. Jetzt ordnen wir alle 256 möglichen Werte (Zeichen) eines Bytes ringförmig an und bewegen uns im Kreis für die Verschlüsselung im Uhrzeigersinn (steigende Werte) und für die Entschlüsselung gegen den Uhrzeigersinn (zu kleineren Werten).

Die Verschiebung, um die wir uns im Kreis bewegen, wird nach jedem verschlüsselten Zeichen um den Zeichencode des nächsten Buchstabens im Schlüsselwort erhöht und mit dem Modulo-Operator in dem Wertebereich [0,255] gehalten.

Mit *key* als dem Schlüsselwort läuft das Herzstück der Verschlüsselung nach folgendem Schema ab:

1. Für die aktuelle Verschiebung wird eine Integer-Variable *shift* definiert und mit dem Wert null initialisiert.
2. Definieren Sie eine Integer-Variable mit Namen *keyIndex* für den aktuellen Index des Buchstabens im Schlüsselwort und weisen Sie ihr den Wert null zu.
3. Erhöhen Sie die Verschiebung *shift* um den Zeichencode des Buchstabens im Schlüsselwort, auf den der Index *keyIndex* zeigt.
4. Überschreitet die Verschiebung *shift* den Wert 255, so ziehen Sie den Wert 256 von ihr ab. (Es bietet sich die Modulo-Division % an.)
5. Erhöhen Sie den Index *keyIndex* um eins.
   Die Schritte 3 bis 5 können in einer Zeile implementiert werden:
   `shift = (shift + key[keyIndex++]) % 256;`
6. Falls die Variable *keyIndex* die Länge des Schlüsselwortes erreicht hat, setzen Sie sie auf null zurück.
7. Verschlüsseln Sie das nächste Byte Ihrer Eingabedatei um dem Wert der Verschiebung *shift*.

8. Ergibt sich nach der Verschiebung ein Wert größer als 255 ziehen Sie 256 von dem Ergebnis ab. (Es bietet sich wieder die Modulo-Division % an.)

9. Schreiben Sie das verschlüsselte Byte in die Ausgabedatei.

10. Wiederholen Sie die Schritte 3 bis 9 bis zum Ende der Eingabedatei.

Beispiel: Wir nehmen an, das Schlüsselwort lautet „Tag“. Die folgende Tabelle zeigt, wie die Verschiebungen für die Verschlüsselung der ersten Buchstaben bestimmt werden:

Byte	*keyIndex*	Schlüsselbuchstabe	Formel	*shift*
1	0	T (84)	(0+84)%256	84
2	1	a (97)	(84+97)%256	181
3	2	g (103)	(181+103)%256	28
4	0	T (84)	(28+84)%256	112
5	1	a (97)	(112+97)%256	209
6	2	g (103)	(209+103)%256	56
...	...	...	...	...

Das Dekodieren erfolgt in entgegengesetzter Richtung. Damit keine negative Verschiebung entsteht, muss bei jedem Schritt zunächst 256 addiert werden:

Byte	*keyIndex*	Schlüsselbuchstabe	Formel	*shift*
1	0	T (84)	(0+256-84)%256	172
2	1	a (97)	(172+256-97)%256	75
3	2	g (103)	(75+256-103)%256	228
4	0	T (84)	(228+256-84)%256	144
...	...	...	...	...

Das Programm soll folgende Eigenschaften haben:

- Es soll *encrypt2* heißen und folgendermaßen aufgerufen werden:
  encrypt2 <Eingabedatei> <Ausgabedatei> <Schluesselwort> +/−
  Beispiel: encrypt2 Beispiel2.txt Kodiert2.txt Tag +
- Mit einem + soll die Verschlüsselung erfolgen und mit einem − soll der Text wieder entschlüsselt werden.
- Verwenden Sie wieder eine Funktion zum Ver-, bzw. Entschlüsseln der Datei.
- Fangen Sie mögliche Fehler ab.
- Öffnen Sie die Dateien im Binär-Modus (auch wenn es sich um Textdateien handelt). Verwenden Sie die Funktionen fread() und fwrite() zum Lesen und Schreiben.

Viel Spaß beim Programmieren!