Aufgabe 6: Kommandozeilenparameter, Dateien & Zeiger

Diese Praktikumsaufgabe besteht aus drei Teilaufgaben, die voneinander unabhängig sind. **Nutzen Sie Preprozessorkommandos**, um entweder den Quellcode für Teilaufgabe 1, 2 oder 3 zu kompilieren. Beispielsweise soll der Quellcode für Aufgabe 2 kompiliert werden bei:

```
#define AUFGABE 2    // Auswahl der Aufgabenteile über Preprozessor
```

Programmierung & Programmierstil:

Achten Sie auf effiziente Programmierung und einen guten Programmierstil, insbes.

- Verwenden Sie für alle Bezeichner sinnvolle & aussagekräftige Namen
- Benutzen Sie das sog. lowerCamelCase[1] für die Namen ihrer Variablen & Funktionen
- Achten Sie auf korrektes Einrücken
- Verwenden Sie keine globalen Variablen
- Halten Sie die übliche Reihenfolge ein: Makros, Include-Befehle, Funktionsdeklarationen, Hauptprogramm, Funktionsdefinitionen
- Fügen Sie zur besseren Gliederung Leerzeilen und ggf. Linien ein
- Kommentieren Sie ihren Code

Fangen Sie mögliche Fehler im Programm ab und entfernen Sie alle Fehler und Warnungen des Compilers.

Aufgabe 1

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

- Jeder Parameter soll in einer eigenen Zeile ausgegeben werden.
- Die Parameter sollen in Anführungszeichen ausgegeben werden.

Wenn ihr Programm bspw. *parameter* heißt und Sie es mit „*parameter Guten Tag 123*“ aufrufen, soll sich folgende Ausgabe ergeben:

Parameter 0: "parameter"

Parameter 1: "Guten"

Parameter 2: "Tag"

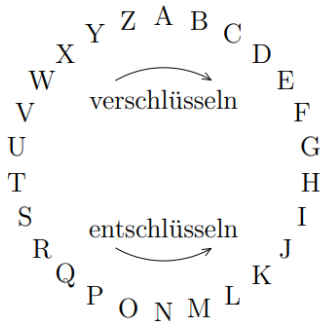
Parameter 3: "123"

[1] Der erste Buchstabe des Bezeichners wird kleingeschrieben und weitere Wörter mit Großbuchstaben hervorgehoben. Beispiel: *getUnsignedShort* oder *returnValue*

Aufgabe 2

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 wird wie folgt aufgerufen:
  <Programmname> <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 und unverändert in die Ausgabedatei geschrieben.
- 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 soll die übergebenen Parameter überprüfen, die Dateien öffnen, die Funktion *encrypt* aufrufen und am Ende die Dateien wieder schließen.

Aufgabe 3

Es soll ein Programm implementiert und getestet werden, das in einem beliebigen Vektor einen bestimmten Wert sucht und alle Positionen im Vektor ausgibt, an denen der Wert vorkommt.

Definieren Sie hierfür die Funktion `find_int()`, die in einem beliebigen Vektor mit `int`-Elementen einen bestimmten Wert sucht. Die Funktion soll einen Zeiger auf das erste gefundene Element zurückgeben (oder `NULL`, falls der Wert nicht vorhanden ist). Als Funktionsparameter erhält die Funktion den gesuchten Wert, den Vektor und seine Länge.

Testen Sie die Funktion im Hauptprogramm `main()`, indem Sie zunächst die Elemente eines `int`-Vektors anzeigen, in dem Werte auch mehrfach vorkommen. Anschließend wird in einer Schleife vom Anwender jeweils eine ganze Zahl eingelesen und durch wiederholten Aufruf von `find_int()` alle Positionen der Zahl im Vektor ausgegeben. Berechnen Sie die Position mithilfe der verwendeten Zeiger. Das Programm soll durch die ungültige Eingabe, z.B. einen Buchstaben, beendet werden.