# Vokabeltrainer/Vocabulary Trainer

Aufgabe ist es, einen **kleinen und schnellen Vokabeltrainer** zu programmieren.
Zwei kleine Beispieldateien mit Vokabeln für Berufe und Tiere jeweils in deutscher und englischer Sprache zum Testen sind dabei vorgegeben.
Realisiert werden soll der Vokabeltrainer über ein sogenanntes assoziatives Feld.
Ein **assoziatives Feld** ist ein Feld, bei dem statt natürlicher Zahlen Zeichenketten als Index, oft **Schlüssel** genannt, verwendet werden.
Oft eingesetzt wird dieses bei elektronischen Wörterbüchern, weshalb auch der Name **Dictionary** dafür verwendet wird./
Task is to program a **small and fast vocabulary trainer**.
Two small example files with vocabularies for professions and animals given in german and english language are given for tests.
The vocabulary trainer shall be realised by a so called associative array.
An **associative array** is an array which uses strings instead of natural numbers as indices, which are called **keys**.
Since they are often used for electronic dictionaries associative arrays are called **dictionaries**, too.

**Beispiele/Examples**

**Schlüssel -> Wert/**
**Key -> Value**

hashtable["Hund"] -> "dog"
hashtable["Katze"] -> "cat"
hashtable["Maus"] -> "mouse"
hashtable["Vogel"] -> "bird"

Zur Realisierung eines solchen assoziativen Felds können sogenannte **Streuwerttabellen**, auch **Hashtabellen** genannt, verwendet werden. Bei diesen wird aus der Zeichenkette, die den Schlüssel darstellt, eine natürliche Zahl als Index errechnet für ein normales Feld.
Die Funktion, die aus einer Zeichenkette $t$ als Schlüssel eine Zahl berechnet, wird als **Streuwertfunktion** oder **Hashfunktion** $h(t)$ bezeichnet, weil die Indizes/belegten Werte in dem normalen Feld typischerweise gestreut/verstreut und nicht geordnet sind, daher auch der Name.
Es gibt verschiedene Möglichkeiten solche Streuwert-/Hashfunktionen zu definieren. Eine einfache Idee stellt die **Divisions-Rest-Methode** dar. Dabei ist $p$ in der Regel eine Primzahl, die auch gleichzeitig die Länge des Feldes definiert. Für eine Zeichenkette $t$ (im 7-Bit-ASCII-Kode, also Zeichen von $0$ bis $2^7-1=127$) mit den Buchstaben $t=$ "$b_1 b_2 b_3 ... b_n$" kann somit ohne Zahlenüberläufe berechnet werden:/
To realise such associative arrays so called **hashing or hash tables** can be used. For these from the string called key a natural number is calculated as index into a normal array.
The function generating a number from a string $t$ as key is called **hashing or hash function** $h(t)$, since the indices/used array elements in the normal array are typically

scattered and not ordered/sorted giving the name to it.
There are different possibilities to define such hash functions. A simple idea is the **division rest method**. A given number $p$ - most often chosen are prime numbers - defines the length of an array. For a string $t$ (in 7-bit-ASCII code, therefore characters from $0$ to $2^7-1=127$) with letters $t=$ "$b_1b_2b_3...b_nb1b2b3…bn$" can then be calculated without number overflows:

$$h(t)=(...((b_1\cdot 128+b_2)\bmod p)\cdot 128+b_3)\bmod p)\cdot 128+...+b_{n-1})\bmod p)\cdot 128+b_n)\bmod p$$

$$h(t)=(…((b1\cdot 128+b2)\bmod p)\cdot 128+b3)\bmod p)\cdot 128+…+bn-1)\bmod p)\cdot 128+bn)\bmod p.$$

(Die Funktion/der Operator $\bmod$ liefert den Rest der Division, in C/C++ der Operator `%`)/
(The function/operator $\bmod$ returns the rest of division, in C/C++ the operator `%`)

**Beispiele Hashwerte/Examples Hashing Values**

Sei $p=661$./Let $p=661$.

$$h(\text{"Hund"})=((((72\cdot 128+117)\bmod 661)\cdot 128+110)\bmod 661)\cdot 128+100)\bmod 661=397$$

$$h("Hund")=(((((72\cdot 128+117)\bmod 661)\cdot 128+110)\bmod 661)\cdot 128+100)\bmod 661=397.$$

$$h(\text{"Katze"})=((((((75\cdot 128+97)\bmod 661)\cdot 128+116)\bmod 661)\cdot 128+122)\bmod 661)\cdot 128+101)\bmod 661=214$$

$$h("Katze")=((((((75\cdot 128+97)\bmod 661)\cdot 128+116)\bmod 661)\cdot 128+122)\bmod 661)\cdot 128+101)\bmod 661=214.$$

Mit einer solchen Hashfunktion können beliebige und beliebig lange Zeichenketten - also eine sehr sehr grosse Anzahl - auf eine natürliche Zahl zwischen $0$ und $p-1$ abgebildet werden. Es kann deshalb vorkommen, dass verschiedene Zeichenketten $t_k$tk, $t_l$tl (und auch noch weitere) auf dieselbe Zahl/denselben Index $h(t_k)=h(t_l)=i$h(tk)=h(tl)=i gehasht werden. Dies wird dann als **Kollision** bezeichnet.
Es gibt nun wiederum verschiedene Ansätze mit solchen Kollisionen umzugehen. Als einfache Lösung kann der berechnete Index $i$ fortlaufend solange um den Wert 1 hochgezählt werden (also $i+1$i+1, $i+2$i+2, ...), bis **der erste freie Feldeintrag** gefunden ist, und stattdessen dann unter diesem hochgezählten Index abgespeichert werden. Wird beim Hochzählen $p$ erreicht, so wird von vorne im Feld ab dem ersten Feldindex 0 weiter gezählt. Dieses Verfahren wird als **lineare Kollisionsstrategie** bezeichnet.
Je kleiner die Primzahl $p$, also die Länge des Felds ist, desto mehr Kollisionen wird es in der Regel geben. Insgesamt können natürlich auch nur maximal $p$ Einträge im Feld abgespeichert werden. Gibt es keine Kollisionen, ist die Suche extrem schnell (nur $O(1)$O(1)-Zeitaufwand plus Berechnung des Hashwertes/Indexes). Hashtabellen mit großem $p$ werden deshalb in der Praxis sehr häufig verwendet und sind oft die Speicherstruktur der ersten Wahl. Gibt es viele Kollisionen, kann sich allerdings die Suche auch linear auf das gesamte Feld erstrecken (also bei $m\leq p$m≤p Einträgen im Feld auch $O(m)$O(m)-Zeitaufwand)./

Such a hash function allows to map arbitrary long strings - a huge infinite number of strings - onto a finite number inbetween $0$0 and $p-$p−. It might happen that different strings $t_k$tk, $t_l$tl (and also further ones) are mapped onto the same number/index $h(t_k)=h(t_l)=i$h(tk)=h(tl)=i. This is called a **collision**.
Different solutions exist to deal with such collisions. A very easy one is to increment the calculated index $i$i in a loop by 1 (therefore $i+1$i+1, $i+2$i+2, ...) **until the first free array element** is found and use instead this. If during incrementing value $p$p is reached counting proceeds at the first index of the array. This method is called **linear collision strategy**.
As smaller prime number $p$p, the length of the array, will be, as more collisions can be expected. In the end maximum $p$p elements can be stored in the array. Are there no collisions search is extremely fast (only $O(1)$O(1) time consumption plus the calculation of the hash value/index). Hashing tables with a big number $p$p are used very often and preferred for many problems. Are there a lot of collisions search can get linear on the whole array (for $m \leq p$m≤p entires in the array $O(m)$O(m) time consumption).

Folgende Teilaufgaben sind zu lösen, **ohne vordefinierte C++-Template-/Containerklassen wie `map` zu benutzen**:/
Following sub tasks have to be solved **without using predefined C++ template/container classes like `map`**:

1. Definieren Sie einen Strukturtyp mit Namen **`Entry`** mit zwei C++ Zeichenketten vom Datentyp **`string`** als Komponenten **`key`** und **`value`** für jeweils einen deutschen und den entsprechenden englischen Begriff./
   Define a type with name **`Entry`** with two C++ **`strings`** as components **`key`** and **`value`** for a german and an english term.
2. Definieren Sie eine Funktion, die die oben beschriebene Hashfunktion $h(t)$h(t) implementiert./
   Define a function implementing above hash function $h(t)$h(t).
3. Definieren Sie eine Funktion, die eine neue Vokabel in die Hashtabelle einträgt.
   Als erster Parameter soll eine deutschsprachige Zeichenkette als Schlüssel, als zweiter die zugehörige englischsprachige Zeichenkette als Wert zu diesem Schlüssel und als dritter Parameter das assoziative Feld/die Hashtabelle vom obigen Typ (1) definiert werden. Beachten Sie, dass Sie im Rumpf neben dem Aufruf der Hashfunktion (2) auch die oben beschriebene lineare Kollisionsstrategie mit definieren./
   Define a function inserting a new vocable into the hash table. As first parameter a german string as key, as second parameter the related english string as value for the key and as third parameter the associative array/hash table of above type (1) shall be defined. Please regard that in its body beside calling the hash function (2) also the linear collision strategy has to be defined.
4. Definieren Sie eine weitere Funktion, die eine Vokabel in der Hashtabelle sucht und die Übersetzung dazu oder eine Fehlermeldung ausgibt, dass diese Vokabel nicht gefunden wurde.
   Als erster Parameter soll eine deutschsprachige Zeichenkette als Schlüssel, als zweiter das assoziative Feld/die Hashtabelle vom obigen Typ (1) definiert werden.
   Beachten Sie, dass auch hier im Rumpf neben dem Aufruf der Hashfunktion (2) die oben

beschriebene lineare Kollisionsstrategie mit berücksichtigt werden muss./
Define a further function searching a vocabulary within the hash table and outputting the translated value or an error message that it can not be found.
As first parameter a german string as key, as second parameter the associative array/hash table of above type (1) shall be defined.
Please regard that also in its body beside calling the hash function (2) the linear collision strategy has to be regarded.

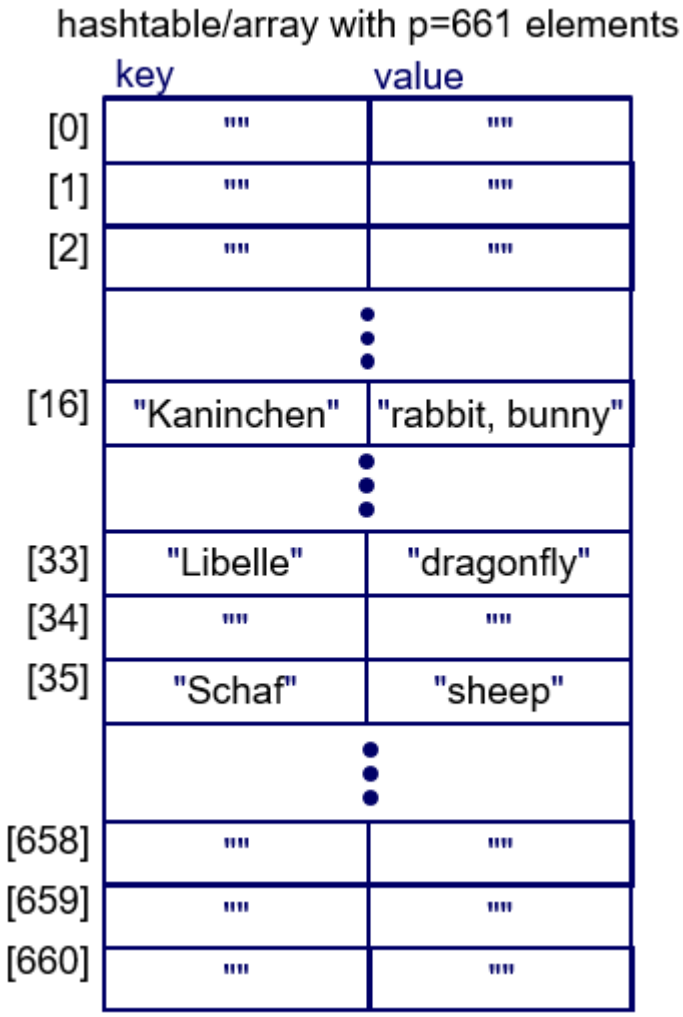
5. Definieren Sie eine Funktion mit einem Dateinamen als erstem Parameter und einem assoziativen Feld/einer Hashtabelle vom obigen Typ (1) als zweiten.
   Die Datei im ersten Parameter enthält in jeder Zeile eine neue deutsche Vokabel als Schlüssel und getrennt mit einem Semikolon (`';'`) als Separator die englischsprachige Übersetzung als Wert.
   Im Rumpf der Funktion soll diese Datei zeilenweise eingelesen und die Vokabeln über einen Aufruf der obigen Funktion (3) in die Hashtabelle eingetragen werden./
   Define a function with a file name as first parameter and an associative array/hash table of type (1) as second.
   The file given by the first parameter has a new german vocable as key and separated by a semicolon (`';'`) as separator the english translation in each line.
   In the body of the function the file shall be read line by line and the vocabulary inserted into the hash table by calling above function (3).
6. Schreiben Sie eine Funktion **`main`**, in der Sie wie in den Beispielen unten
   - zuerst eine Primzahl pp abfragen.
   - Definieren Sie dann ein Feld vom obigen Typ (1) mit pp Elementen als Hashtabelle.
   - Rufen Sie danach Ihre Funktion (5) mit der Datei **`"tiere_animals.txt"`** und der Hashtabelle als Argumente auf.
   - Definieren Sie zuletzt eine Schleife, in der ein deutschsprachiges Wort eingegeben und die englische Übersetzung über einen Funktionsaufruf von (4) ausgegeben wird.

   Write a function **`main`** and like in the examples below

   - first ask for a prime number pp.
   - then define an array of type (1) with pp elements as hash table.

   - afterwards call function (5) with file **`"tiere_animals.txt"`** and the hash table as arguments.
   - last define a loop, where a german word can be inputted from keyboard and the englsih translation is outputted by calling function (4).

**Beispiel Hashtabelle/Example Hash Table**




**Hinweise/Hints**

Für p=661p=661 gibt es für die Datei **`"tiere_animals.txt"`** keine Kollisionen, so dass Sie in einem ersten Schritt Ihr Programm ohne lineare Kollisionsauflösung entwickeln und testen können./
For p=661p=661 there are no collisions for file **`"tiere_animals.txt"`**, therefore in a first step you can develop and test your program without having a collision strategy defined.

Fügen Sie in einem zweiten Schritt dann die lineare Kollisionsauflösung hinzu.
Für p=179p=179 gibt es für die Datei **`"tiere_animals.txt"`** 11 Kollisionen,
für p=53p=53 sogar 157 Kollisionen./
Add in a second step a linear collision strategy. For p=179p=179 and
file **`"tiere_animals.txt"`** 11 collisions exist, for p=53p=53 much more 157 collisions.

Fügen Sie während der Entwicklung an sinnvollen Stellen Ausgaben in Ihr Programm ein, so dass Sie verfolgen können, ob dieses korrekt arbeitet. Beispiele solcher Programmausgaben finden Sie als separate Dateien zum Download hier in Moodle./

Add during programming outputs at senseful points in your program that you can follow whether it works correctly. Examples of such program outputs you can find as separate files to download in Moodle.

Testen Sie danach Ihr Programm auch für die Datei **"berufe_professions.txt"** und verschiedene Primzahlen pp als Feldgröße./
Test your program also for file **"berufe_professions.txt"** and different prime numbers pp as length of array.

**Beispiele Programmläufe/Examples Program Runs**

```
please input value for (prime) number p: 661
translate (0 for end): Hund
Hund -> dog
translate (0 for end): Katze
Katze -> cat
translate (0 for end): Meerschweinchen
Meerschweinchen -> cavy, guinea pig
translate (0 for end): Wildschwein
sorry not known
translate (0 for end): Reh
sorry not known
translate (0 for end): Hase
sorry not known
translate (0 for end): Huhn
sorry not known
translate (0 for end): Kaninchen
Kaninchen -> rabbit, bunny
translate (0 for end): Tiger
Tiger -> tiger
translate (0 for end): 0
please input value for (prime) number p: 53
translate (0 for end): Hund
Hund -> dog
translate (0 for end): Maus
Maus -> mouse
translate (0 for end): Wal
sorry not known
translate (0 for end): Katze
Katze -> cat
translate (0 for end): Delphin
sorry not known
translate (0 for end): Robbe
Robbe -> pinniped, seal
translate (0 for end): Krebs
Krebs -> crab
translate (0 for end): Vogel
Vogel -> bird
translate (0 for end): 0
```