Der Geschäftsführer und sein Handy

Trudchen Taste ist Assistentin von Otto Heitz, Geschäftsführer einer Firma für Sicherheitsschlösser, die erst kürzlich ihre Produktpalette durch elektronische Schlösser ergänzt hat. Heitz hat Schwierigkeiten mit dem neuen Handy, das Trudchen ihm empfohlen hat, damit er in Zukunft auch auf dem Golfplatz wichtige Telefonate führen kann. Das Handy verfügt über einen Rufnummernspeicher, der bei Eingabe eines Namens automatisch die richtige Nummer wählt. Der Name wird dabei über die Zifferntastatur eingegeben, die folgendermaßen belegt ist:

Da die Tasten mehrfach belegt sind, wird der jeweils richtige Buchstabe durch mehrmaliges Drücken der Taste ausgewählt. Die Buchstaben werden durch Eingabepausen voneinander getrennt. So ist z.B. Trudchens Nummer unter "Taste" gespeichert, also durch 8<Pause>2<Pause>7777<Pause>8<Pause>33 erreichbar.
Heitz ist das zu kompliziert; er fragt sich, warum es nicht reicht, jede Taste einmal zu drücken (also 82783 für Trudchen). Trudchen erklärt ihm geduldig, dass dann Mehrdeutigkeiten auftreten können. So hätte Heitz' Geschäftspartner Viktor Vapud dann die gleiche Tastenkombination für seine Nummer wie sie selbst. Um Heitz die Sache zu vereinfachen, gibt Trudchen jedoch eine Änderung der Software in Auftrag. Diese erlaubt es Heitz, tatsächlich jede Taste nur einmal zu drücken und dann aus den passenden Namen den richtigen durch das Drücken weiterer Tasten auszuwählen.

Aufgabe:

▶ Schreibe ein Programm, das eine Liste mit Namen und Telefonnummern (keine Umlaute oder ß, Groß-/Kleinschreibung spielt keine Rolle) einliest und dann Telefonnummern anhand einer Eingabe auf der numerischen Tastatur heraussucht.

▶ Wird dein Programm langsamer, wenn die Liste deutlich länger wird? Beschreibe eine Datenstruktur, die in der Lage ist, die Suchzeiten bei größer werdender Liste gleich zu halten.

Lösungsidee:

Die Namen der Gesprächspartner werden durch Ersetzung der Buchstaben durch entsprechende Ziffern in die (nicht eindeutigen) Ziffernfolgen (auch Suchfolgen genannt) umgewandelt, die auf dem Handy bei der Suche verwendet werden sollen. In einer Hashtabelle werden die Zuordnung von jeder dieser Suchfolgen zu einem oder mehreren Namen (z.B. 82783 zu Vapud und Taste) und den zugehörigen Telefonnummern gespeichert.
Bei der Suche werden alle Einträge, die einer eingegebenen Ziffernfolge zugeordnet sind, zusammen mit weiteren, diesmal eindeutigen Ziffern ausgegeben. Der Benutzer kann den gewünschten Eintrag durch Eintippen der passenden Ziffer(n) auswählen, worauf das Handy die richtige Nummer wählt.

Programm-Dokumentation:

Das Programm ist in Perl geschrieben, so dass für die Hashtabelle, die die Zuordnung zwischen Suchfolgen und Namen/Telefonnummern speichert, die entsprechenden in Perl eingebauten Mechanismen verwendet werden.
Nach dem Start (perl handy.pl) öffnet das Programm die Datei telnum.lst. Sie enthält die Namen und Telefonnummern der Gesprächspartner in der folgenden Form:

Trudchen Taste:766235

Die einzelnen Einträge in der Datei werden in Vorname, Nachname und Telefonnummer zerlegt. Aus den Buchstaben des Nachnamens wird die entsprechende Suchfolge gebildet. Jeder Eintrag wird unter seiner Suchfolge in der Hashtabelle %eintraege abgespeichert.
Danach kann der Benutzer eine Ziffernfolge eingeben. Alle unter dieser Suchfolge abgespeicherten Einträge werden aus der Hashtabelle ausgelesen. Wenn es mehrere sind, wird jedem Eintrag im nächsten Schritt eine eindeutige weitere Ziffernfolge zugewiesen. In einer temporären Hashtabelle wird dann die Telefonnummer des Eintrags unter dieser weiteren Ziffernfolge abgespeichert. Wenn der Benutzer nun seine Auswahl eingibt, kann die Telefonnummer aus der Hashtabelle ausgelesen und vom Handy gewählt werden.

Programm-Ablaufprotokoll

In allen Beispielen wird zuerst die Eingabedatei eingelesen, die u.a. folgende Einträge enthält:

Trudchen Taste:766235
Viktor Vapud:978614
Dunkel Kammer:752452
Steffen Jammer:880123
Beispiel 1:
Name (Kurzform) eingeben: 82783

Wählen Sie einen Gesprächspartner aus!
1. Trudchen Taste: 766235
2. Viktor Vapud: 978614
Ihre Wahl: 1

Wähle 766235
... und zwei weitere Beispiele

2. Teilaufgabe

Wird dein Programm langsamer, wenn die Liste deutlich länger wird? Beschreibe eine Datenstruktur, die in der Lage ist, die Suchzeiten bei größer werdender Liste gleich zu halten.
Das Programm verwendet für die Speicherung der Daten eine Hashtabelle. Die Daten werden unter einem Schlüssel abgelegt; hier ist die Suchziffernfolge der Schlüssel. Eine Hashtabelle basiert auf einem Feld mit M Speicherplätzen, die direkt adressiert werden können (ein Array also). Anhand des Schlüssels können Daten aus einer Hashtabelle ausgelesen werden. Der Schlüssel wird in eine Zahl k umgewandelt. Aus dieser Zahl wird durch weitere Rechenoperationen die Adresse ermittelt, unter welcher der gesuchte Datensatz gespeichert ist. Im einfachsten Fall entspricht die Adresse dem Divisionsrest aus k und M (k mod M). Die Speicheradresse eines gesuchten Datensatzes kann also auf direktem Wege ermittelt werden. Es spielt dabei fast keine Rolle, wie viele Einträge in der Hashtabelle gespeichert sind. Das Programm wird also fast nicht langsamer, wenn die Telefonliste länger ist.
Warum „fast"? Weil es passieren kann, dass mehrere Schlüssel auf die gleiche Adresse abgebildet werden.

Dann muss man sich um Kollisionsbeseitigung kümmern, z.B. indem an einer Stelle in der Tabelle eine Liste mit den dort abgelegten Datensätzen verwaltet wird. Kollisionsbeseitigung passiert hier prinzipiell umso öfter, je mehr Daten abgelegt werden. Idealerweise sollte M also ziemlich groß sein (und eine Primzahl, damit bei der Modulo-Rechnung nur sehr wenige Zahlen auf die gleiche Adresse abgebildet werden).
Noch etwas besser als eine Hashtabelle wäre die Verwaltung einer hierarchischen Struktur, nämlich eines Baumes, dessen Knoten jeweils 8 Nachfolger haben, einen für jede Ziffer, die für einen Buchstaben stehen kann. Bei einer vorgegebenen Suchfolge wie 82783 für „Taste" startet man also am Wurzelknoten, besucht dessen 8. Nachfolger, dann wiederum dessen 2. Nachfolger usw. bis die Folge abgearbeitet ist. An dieser Stelle legt man dann den Datensatz ab bzw. liest ihn aus. Auch hier kann es Kollisionen geben („Taste" und „Vapud" hängen dann am gleichen Knoten), und je länger die Telefonliste, desto wahrscheinlicher ist es, dass zwei Namen die gleiche Suchfolge haben. Prinzipiell kann man aber sagen, dass es für jeden Knoten eine konstante maximale Zahl von Einträgen gibt, unabhängig von der Länge der Liste. Hier ist die Suchzeit also eher von der Länge der Suchfolge (also des Namens) abhängig, weil pro Ziffer eine weitere Ebene in der Hierarchie abgestiegen werden muss. Weil man in einem solchen Baum nur die Knoten anlegen muss, die man wirklich für die Eingabedaten braucht, ist der Speicherverbrauch im Gegensatz zur Hashtabelle übrigens recht günstig.

Programm-Text:

```
#!/usr/bin/perl

open DATEI, "<telnum.lst";
        # Liste mit Telefonnummern öffnen
for (<DATEI>)
        # Daten einlesen
    {
    ($name, $tel) = split /\:/;
            # Telefonnummer und Name trennen
    chop $tel;
            # \n-Zeichen entfernen
    ($vorn, $nachn) = split / /, uc $name;
            # Vorname und Nachname trennen
            # (Großbuchstaben)
    $nachn =~ tr/A-CD-FG-IJ-LM-OP-ST-VW-
    Z/22233344455566677778889999/;
            # Zeichen in entsprechende Ziffern umwandeln
    $eintraege{$nachn} .= "$name:$tel&";
            # Eintrag unter dieser Nummer speichern
    }
            #  (Nummer = zu drückende Handytasten)
close DATEI;
        # Datei wieder schließen

print "\nName (Kurzform) eingeben: ";
chop ($eingabe = <STDIN>);
        # Benutzereingabe einlesen
print "\nWählen sie einen Gesprächspartner aus!";

for (split /&/, $eintraege{$eingabe})
        # Für alle in Frage kommenden Einträge
    {
    ($name, $tel) = split /\:/;
            # Eintrag in Namen und Telefonnummer
            # aufspalten
    $i++;
            # Nummer des Eintrags erhöhen
    print "\n$i. $name: $tel";
            # Nummer, Name und Telefonnummer ausgeben
    $auswahl{$i} = $tel;
            # Telefonnummer in einem Hash speichern
    }
print "\nIhre Wahl: ";
chop ($eingabe = <STDIN>);
        # Benutzereingabe einlesen
print "\nWähle $auswahl{$eingabe}\n";
        # ...und Telefonnummer wählen...
```

Lösung nach Markus Völker

Design: Kuhn, Kammann & Kuhn, Köln Illustration: Chris Hübel, Gunnar Collier, Köln Druck: DFS Druck und Verlag, Köln