

Taskmanager für Wichtel - Teil 2

Wir arbeiten an unserem Datenbank-ähnlichen Programm weiter, in welchem wir Wichtel und ihre Aufgaben verwalten.

Von der letzten Aufgabe sind folgende Funktionen, Subroutinen und Strukturen gegeben:

- ADTs für Zeit (mit Stunden, Minuten und Sekunden), Wichtel (mit Namen und Arbeitszeit), und Aufgaben (mit Namen, Beschreibung, Soll- und Habe- Arbeitszeit)
- Generische Subroutine *GET* zum Einlesen von Zeiten, Wichteln und Aufgaben
- Generische Subroutine *PUT* zum Ausgeben von Zeiten, Wichteln und Aufgaben
- Generische Subroutinen *update_zeit* für Wichtel und Aufgaben, die jeweils eine mitgelieferte Zeit auf die Arbeits- / Habe-Zeit addiert
- Die Operatoren $<$, $+$, $-$ für Zeiten. (Hinweis: $-$ gibt die absolute Differenz zurück, nicht die normale Subtraktion)
- Die Funktion *zeit_string*, welche eine Zeit als Characterstring der Form "hh:mm:ss" zurückgibt

Die Datenbanken für Wichtel und Aufgaben sind Vektoren von den entsprechenden ADTs. Die Zeilen der Datenbank sind dementsprechend die Einträge des Vektors, und die jeweiligen Spalten pro Zeile der Datenbank sind die Komponenten der ADTs.

Es sollen nun weitere Subroutinen in die passenden Module geschrieben werden. Nutzt dafür die schon gegebenen Funktionen und Subroutinen. (Hinweis: generisch = mit Interface)

Diese Subroutinen sind für die grundlegende Struktur notwendig:

- Je eine generische Subroutine *GET* für Wichtel und Aufgaben, in welchen Eingelesen wird, wie viele Einträge die Datenbank haben soll. Es soll ein Vektor von passender Größe und Datentyp allokiert und gefüllt werden.
- Je eine generische Subroutine *PUT* für Wichtel und Aufgaben, in welchen eine Datenbank (ein Vektor vom Typ Wichtel/Aufgabe) kommentiert ausgegeben wird.

Diese Subroutinen sind für das Arbeiten mit den Datenbanken wichtig:

- Je eine generische Subroutine *search* für Wichtel und Aufgaben. In diesen wird ein Name eingelesen und alle Matches in der Datenbank mit ihrem Index ausgibt. (Zusatz: implementiere ein fuzzy search", sodass zB case-insensitive gesucht wird, oder 1 Buchstabe zu viel oder zu wenig in dem Namen sein darf)
- Je eine generische Subroutine *recommend* für Wichtel und Aufgaben, in welchen vorgeschlagen wird welcher Wichtel / an welcher Aufgabe als nächstes gearbeitet werden soll
 - Der vorgeschlagene Wichtel ist der mit der geringsten Arbeitszeit
 - Die vorgeschlagene Aufgabe ist die, mit der geringsten Restarbeitszeit (SOLL-HABE ist minimal)
 - Für die vorgeschlagene Aufgabe lohnt es sich eine Funktion *status* anzulegen, welche zurückgibt ob eine Aufgabe schon begonnen wurde, oder nicht, oder schon beendet (Hinweis: die globale Variable *zeit_null* ist gleich die Zeit Null)
- Eine Funktion *update_work*, in welcher eingelesen wird, welcher Wichtel an welcher Aufgabe gearbeitet hat (nutzte die Indizes) und wie lange, und die Arbeits- und Soll-Zeit updatet