Programmieren 1 Übung: Abschluss / "Denksport": Backtracking

Klaus Kusche

"Rucksack packen"

Schreib ein Programm, das den *optimalen Inhalt eines Rucksacks* berechnet:

- Gegeben ist das *Maximal-Füllgewicht* des Rucksacks.
- Weiters ist eine *fixe Anzahl von Gegenständen* gegeben.
- Jeder Gegenstand hat ein *Gewicht* und einen *Wert*, beides vom Typ **`double`**, und sinnvollerweise einen *Namen*, z.B. "Fotoapparat" oder "Regenmantel".

  Verwende für diese Daten ein Array von **`struct`**'s (eine **`struct`** pro Gegenstand).
- Jeder Gegenstand kann entweder ganz oder gar nicht in den Rucksack gepackt werden, aber nicht teilweise.
- Eine Lösung ist *zulässig*, wenn die *Summe der Gewichte* der eingepackten Gegenstände das *Maximalgewicht* des Rucksacks *nicht überschreitet*.
- Der Wert einer Lösung ist die *Summe der Werte* aller eingepackten Gegenstände.
- Es soll jene Lösung ausgegeben werden, die *von allen zulässigen Lösungen den höchsten Wert* hat.

Für den ersten Versuch darfst du Gewicht und Wert aller Gegenstände fix ins Programm codieren (mittels *Initialisierung* in der Deklaration des Arrays von Strukturen), das Maximalgewicht ist beim Programmaufruf auf der *Befehlszeile* angegeben.

Die **rekursive Lösungsidee (Backtracking)** ist in etwa Folgende:

- Man beginnt mit dem *leeren* Rucksack.
- Jede Ebene des rekursiven Aufrufs prüft *einen* Gegenstand:
  - Wenn der Gegenstand vom Gewicht her noch in den Rucksack *passt*, gibt man den Gegenstand *in den Rucksack* und macht einen *rekursiven Aufruf* für die restlichen Gegenstände.
  - Egal, ob der Gegenstand passt oder nicht, macht man einen *weiteren rekursiven Aufruf* für die restlichen Gegenstände, *ohne* den aktuellen Gegenstand in den Rucksack gepackt zu haben.
- Hat man *alle Gegenstände* entschieden (Rekursionstiefe = Anzahl der Gegenstände), prüft man, ob die aktuelle Lösung einen *höheren Wert* hat als die beste bisher bekannte Lösung. Wenn ja, speichert man sie als neue optimale Lösung.
- Nachdem der oberste Aufruf im Hauptprogramm zurückgekehrt ist, gibt man die gespeicherte optimale Lösung (Packliste, Gesamtgewicht und Gesamtwert) aus.

Hinweise:

- Für das *Speichern der Lösung* bekommt unsere Struktur für einen Gegenstand *zwei* zusätzliche **`bool`**-*Member*: Das erste zeigt an, ob der Gegenstand in der *gerade berechneten Lösung* eingepackt wird (**`true`**) oder zu Hause bleibt (**`false`**).

Das zweite **bool**-Member, zeigt an, ob der Gegenstand in der *bisher besten Lösung* in den Rucksack gepackt wird oder nicht.

- Es ist ganz praktisch, das *Array der Strukturen für die Gegenstände global* zu machen. Auch den *Wert* der bisher *besten Lösung* merkt man sich am besten *global*.
- Die rekursive Funktion hat sinnvollerweise 3 Parameter und keinen Returnwert:
    - Den *Array-Index* **i** des Gegenstandes, den sie als nächstes prüfen soll (= *Rekursionstiefe*, d.h. wächst bei jedem rekursiven Aufruf um 1).
    - Die *Restkapazität* des Rucksackes (= das *noch freie Gewicht*).
    - Den *Gesamtwert* der aktuell im Rucksack befindlichen Gegenstände.

Erweiterungen:

- Baue einen *Aufrufzähler* (welche Art von Variable?) in dein Programm ein und gib am Ende des Programmes aus, wie viele rekursive Aufrufe gemacht wurden.
- Versuche, die Anzahl der rekursiven Aufrufe durch zusätzliche Programmlogik zu reduzieren. Dazu gibt es viele Strategien, eine recht gute *Optimierung* ist folgende:
    - In der *Struktur* für jeden Gegenstand wird *zusätzlich* sein *Wert pro Gewicht* (also sein Wert dividiert durch sein Gewicht) gespeichert.
    - Die Liste aller Gegenstände wird *vor Beginn* der Lösungssuche nach diesem Wert pro Gewicht *sortiert* (z.B. **qsort** verwenden!), und zwar so, dass die *in Relation zum Gewicht wertvolleren Gegenstände zuerst* betrachtet werden.
    - Bevor man prüft, ob man den nächsten Gegenstand noch in den Rucksack gibt, berechnet man jedesmal eine *obere Schranke für den maximal noch möglichen Wert* der aktuellen Lösung: Diese ist "Wert der bisher im Rucksack befindlichen Gegenstände" + "Wert pro Gewicht des aktuellen Gegenstandes" * "Restkapazität des Rucksacks". (Warum?)
    - Ist diese obere Schranke für den Lösungswert *kleinergleich* dem globalen Wert der *bisher besten gefundenen Lösung*, kann man *sofort zurückkehren*, ohne diesen und weitere Gegenstände zu betrachten: Der aktuelle Aufruf kann keine bessere Lösung als die schon bekannte mehr liefern.
- Und wenn dir noch Zeit bleibt: Lies die Werte und Gewichte der *Gegenstände aus einem File* ein, anstatt sie fix im Programm vorzugeben:
    - Der *Filename* wird auf der *Befehlszeile* angegeben.
    - Der File enthält *pro Gegenstand eine Zeile* mit *Name, Gewicht und Wert* (verwende **fscanf**, das Format für **double**-Werte ist **%lf**).
    - Du darfst in deinem Programm eine *fixe obere Schranke* für die *Anzahl* der Gegenstände codieren. Es ist nicht notwendig, statt dem Array fixer Größe für die Gegenstände ein beim Lesen mittels **realloc** wachsendes Array oder eine verkettete Liste zu bauen (das wäre die Super-Extra-Lösung). Auch für die *Namen* der Gegenstände darfst du eine *fixe Maximallänge* vorgeben.
    - Du solltest *Formatierungsfehler* im Eingabefile erkennen und melden, aber du musst nicht versuchen, sie zu überspringen und weiterzulesen.