Das Problem

Bei der uns vertrauten Schreibweise mathematischer Rechenterme, der sog. *Infix-Notation* oder auch *Peano-Russell-Notation*, befindet sich der Operator `(+,-,*)`[1] zwischen den beiden Operanden. Ein solcher Term ist beispielsweise

```
((3 + 7) - (15 * 5)) * 2
```

Da jeder Operator genau zwei Operanden benötigt, müssen ggf. viele Klammern gesetzt werden. Um sie zu vermeiden, wurden sog. Vorrangregeln („Punkt- vor Strichrechnung“, „von-Links-Nach-Rechts“) eingeführt. Damit verkürzt sich der obige Term (geringfügig) zu

```
(3 + 7 – 15 * 5) * 2
```

Der Polnische Mathematiker Jan Łukasiewicz (1878 - 1956) entwickelte eine (für uns zunächst ungewohnte) Schreibweise für arithmetische Terme, bei der man den Operator vor die Operanden setzt. Statt `7 * 3` schreibt man `* (7,3)`. Sind alle zulässigen Operatoren zwei-stellig, kann man auf die Klammern verzichten, muss jedoch zwischen den Operanden einen Trenner (z.B. ein Komma oder ein Leerzeichen) einfügen. Statt `* (7,3)` schreibt man dann `* 7 3`. Diese Art der Schreibweise nennt man *Polnische Notation* oder auch *Präfix-Notation*.

Der obige Term würde dann in der Form

```
*(-(+(3,7),*(15,5)),2)
```

oder klammerlos

```
* - + 3 7 * 15 5 2
```

notiert werden.

Mathematische Funktionen notiert man in der Regel in Präfix-Form. Der Name der Funktion wird vor die Parameter geschrieben, wie z.B. bei:

```
sin (1.5)
```

Die sog. UPN (*Umgekehrt-Polnische-Notaion*) bzw. *Postfix-Notation* setzt den Operator hinter die Operanden, so dass man jetzt statt `7 * 3` bzw. `* 7 3` die Notation `7 3 *` findet. Auch hier muss zwischen den beiden Operanden ein Trenner eingefügt werden.

In UPN ist jetzt der obige Term folgendermaßen zu schreiben:

```
3 7 + 15 5 * - 2 *
```

Wie man sieht, können Klammern sowie Vorrangregeln vollständig entfallen. Das bedeutet, dass man auch komplexe Berechnungen durchführen kann, ohne darauf achten zu müssen, auf welcher verschachtelten Klammerebene man sich momentan befindet. Und das bedeutet wiederum, dass man für eine Berechnung weniger Tastenklicks benötigt.

Die UPN findet heute Anwendung z.B. in der Seitenbeschreibungssprache *Postscript*. So

[1] Hier wird zunächst auf die Division verzichtet, da man sich bei den betrachteten Operanden auf ganzzahlige (ggf. sogar nicht-negativ-ganzzahlige, also natürlichzahlige) Werte beschränkt, um Probleme mit der restbehafteten Division zu vermeiden. Siehe dazu die Vorschläge zur Erweiterung.

schreibt man beispielsweise den Term `sin(5*7)` in Postscript als `5 7 * sin`. Auf manchen älteren Taschenrechnern musste man mathematische Funktionen nach den Operanden eingeben. Wollte man dort z.B. die Wurzel aus 2 berechnen, so musste man **nach** Eingabe der „2“ auf die "Wurzel-Taste" drücken.

Die Firma Hewlett-Packard Co. erkannte die Vorteile der Postfix-Methode gegenüber algebraischen Standardausdrücken beim Einsatz von Taschenrechnern und Computern, und übernahm die UPN 1972 für wissenschaftliche Taschenrechner.

Um den Wert eines in UPN vorliegenden Rechenterms zu bestimmen, muss jeder Operator auf die beiden[2] vor ihm stehenden Operanden angewandt werden. Das "+" im obigen Term bezieht sich auf 3 und 7, das "*" auf 15 und 5. Schwieriger wird es bei dem anschließenden "-". Die beiden Operanden sind die Ergebnisse der beiden vorigen Rechnungen. Hätte man also die Ergebnisse dieser Teilterme statt der (unausgewerteten) Teilterme, wäre die Behandlung des "-" erledigt.

Damit ergibt sich folgendes „Rezept“ zur Auswertung eines UPN-Terms:

- Man liest den Term von links nach rechts.
- Liest man einen Operator, führt man die entspr. Rechenoperation mit den beiden letzten Operanden aus und ersetzt diese beiden Operanden durch das jew. Ergebnis.

[2] Wir benutzen hier ausschließlich zweiwertige Operatoren. Jedoch kann man ebenso nicht-zweiwertige Operatoren auswerten. Siehe dazu die Anregungen für Erweiterungen.

Idee zur Lösung

Um diesen Algorithmus einfach umsetzten zu können, benutzt man einen sog. *Stapel*.

Der Stapel (auch *Kellerspeicher* oder *Stack* genannt) ist eine der einfachsten Datenstrukturen mit einem beschränkten Zugriff auf gespeicherte Elemente. In der Informatik ist ein Stack ein Bespiel eines sog. Abstrakten Datentyps (ADT). Ein ADT besteht aus

- einer Menge von Werten,
- einer Menge von Operationen auf dieser Menge
- sowie einer genauen Beschreibung der Semantik dieser Operationen.

Ein ADT ist programmiersprachenunabhängig. Es wird beschrieben, was die Operationen bewirken, nicht wie sie es tun. Eine Realisierung / Implementierung ist nicht Teil eines ADT.

Ein Stack verwirklicht das so genannte LIFO-Prinzip (LIFO für *Last-In-First-Out-Speicher*). Beim Auslesen eines Elementes wird das jeweils zuletzt gespeicherte Element zuerst ausgelesen. Nachdem man dieses (oberste) gelöscht hat, kann man das vorletzte (jetzt oberste) lesen.

Dabei liegt es nahe, an einen Stapel von Tellern oder Tabletts zu denken, bei dem man auch immer nur auf das oberste Objekt zugreifen kann, dasjenige also, welches als letztes auf dem Stapel abgelegt wurde.

Weitere Beispiele, in denen dieses Prinzip benutzt wird:

- Muss man sich mit einem kleinen Schuppen begnügen, so muss man erst die Wasserkästen und das Fahrrad ausräumen, bevor man an den Karton mit der Weihnachts-Dekoration vom letzten Jahr kommt.
- Sucht man durch Versuch und Irrtum nach der Backtracking-Methode die Lösung eines Rätsels oder den Weg aus einem Labyrinth, so wird bei einer Sackgasse nur die zuletzt getroffene (und zuletzt gespeicherte) Entscheidung rückgängig gemacht und durch einen anderen Versuch ersetzt. Erst, wenn das auch nicht hilft, geht man noch ein Stück weiter auf dem betretenen Weg zurück.

Die (einzigen) Stack-Operationen sind:

- `push(x)` legt das Element x auf den Stack
- `top()` liefert das zuletzt auf den Stack gelegte Element
- `pop()` entfernt das zuletzt auf den Stack gelegte Element.
- `isEmpty()` liefert genau dann `true`, wenn kein Element mehr auf dem Stack liegt.

Hinzu kommt ein Konstruktor, der einen neuen (leeren) Stack erzeugt.

Erweiterungen

Die folgende Auflistung ist ein Vorschlag für eine Abfolge zur Erweiterung der Unterrichtseinheit:

- Die Operanden sind Dezimalzahlen.
- Umwandlung eines Infix-Terms in den äquivalenten UPN-Term.

Weitere Aufgaben zum Thema Stack

- Erforschen Sie den Aufbau der Sprache Postscript. Diese Aufgabe könnte – alternativ zu der UPN-Problematik – am Beginn des Projektes stehen.
- Eine Möglichkeit, Objekte zu sortieren, benutzt zwei Objekte der Klasse `Stack`.
- Simulieren Sie die Verwaltung eines Container-Depots.
  - Das Depot hat eine Anzahl von Stellplätzen.
  - Auf jedem Stellplatz können beliebig viele Container gestapelt werden.
- Ein Rangierbahnhof (siehe Bild unten) soll modelliert werden.
  - Die Zugmaschine kann nur jeweils einen Waggon ziehen oder schieben.
  - Auf den drei Gleisen können beliebig viele Waggons stehen.
  - Die Zugmaschine kann nur den jeweils „linkesten“ Waggon von einem Gleis wegziehen.