Praktische Informatik II SS 2007

Übungsblatt 1a

Abgabe: 30.04.2007

Aufgabe 1: Ringliste reloaded

Wie im Wintersemester gelernt, sind doppelt verkettete Ringlisten folgendermaßen konstruiert:

- Jedes Listenelement `m` ist mit einer Referenz `m.n` auf den **Nachfolger** und einer Referenz `m.p` auf den **Vorgänger** versehen (sog. doppelte Verkettung).
- Jede Liste enthält immer ein sog. **Ankerelement**. Dieses trägt niemals Nutzdaten (Komponente 'data' ist die null-Referenz).
- Die leere Liste wird allein durch das Ankerelement repräsentiert, bei dem Nachfolger und Vorgänger auf sich selbst zeigen.
- Bei einer nicht leeren Liste ist der Nachfolger des Ankerelementes der **Listenkopf**, d.h. das erste Nutzdaten tragende Element der Liste. Der Vorgänger des Ankerelementes ist das **letzte Element** der Liste. Umgekehrt hat der Listenkopf immer das Ankerelement als Vorgänger, und das letzte Element der Liste hat immer das Ankerelement als Nachfolger.

Der Vorteil einer doppelt verketteten Ringliste besteht darin, dass Einfüge- und Löschoperationen, sowie Konkatenation ohne spezielle Fallunterscheidungen "Liste leer/nicht leer", "Löschen des letzten Elementes" etc. auskommen.

Implementiert eine *objekt-orientierte* Version der Ringlisten mit den Methoden, die von einem Ringlistenobjekt werwendet werden:

- `RList()` - Konstruktor erzeugt eine neue leere Liste.
- `rlIsEmpty` - prüft, ob die Liste leer ist.
- `rlLen()` - berechnet die Länge der Liste.
- `rlApp()` - fügt ein Listenelement hinter dem angegebenen an.
- `rlIns()` - fügt ein Listenelement vor dem angegebenen an.
- `rlReset()` - löscht den Listeninhalt.
- `rlRead()` - gibt den Inhalt des referenzierten Listenelements zurück.
- `rlNext()` - gibt die Referenz auf den Nachfolger des referenzierten Listenelements zurück.
- `rlPrev()` - gibt die Referenz auf den Vorgänger des referenzierten Listenelements zurück.

- `rlFind()` - sucht nach einem Listenelement, das den gegebenen String enthält.
- `rlPrint()` - gibt den Nutzdateninhalt aus.
- `rlDel()` - löscht das referenzierte Listenelement.
- `rlInsSorted()` - fügt ein Listenelement sortiert ein.
- `rlCat()` - verkettet mit einer anderen Ringliste. Dabei wird die andere Ringliste verändert - sie ist nach der Operation die leere Liste!
- `rlEquals()` - vergleicht mit einer anderen List auf inhaltliche Gleichheit.