Fachbereich IV - Abteilung Informatik
Algorithmen und Datenstrukturen
Universität Trier

Andrea Jaax
Tel.: (0651)201-3280
jaax@uni-trier.de

3. Übung:

# Softwarepraktikum C++ / LEDA

Wintersemester 2012/13 — 12. November 2012

---

Abgabe per Email bis Montag, 19. November 2012, 10:00

**Aufgabe 3.1:** (10 Punkte)

*Listen:*
Implementieren Sie eine Klasse int_list mit folgenden Funktionen:

- **int_item L.first()** returns the first item of L (nil if L is empty)
- **int_item L.last()** returns the last item of L (nil if L is empty)
- **int_item L.get_item(int i)** returns the item at position i (the first position is 0) Precondition 0 < = i < L.length().
- **int_item L.succ(int_item it)** returns the successor item of item it, nil if it = L.last(). Precondition it is an item in L.
- **int_item L.pred(int_item it)** returns the predecessor item of item it, nil if it = L.first()
- **const int& L.contents(int_item it)** returns the contents L[it] of item it. Precondition it is an item in L.
- **int_item L.push(const int& x)** adds a new item < x > at the front of L and returns it
- **int_item L.append(const int& x)** appends a new item < x > to L and returns it
- **int L.pop()** deletes the first item from L and returns its contents
- **int L.del_item(int_item it)** deletes the item it from L and returns its contents L[it]
- **void L.move_to_front(int_item it)** moves it to the front end of L
- **void L.move_to_rear(int_item it)** moves it to the rear end of L
- **void L.conc(list<int>& L1)** appends list L1 to list L and makes L1 the empty list. Precondition: L! = L1
- **void L.split(int_item it, list<int>& L1, list<int>& L2)** splits L at item it into lists L1 and L2. More precisely, if it! = nil and L = x1,..., xk-1, it, xk+1,..., xn then L1 = x1,..., xk-1 and L2 = it, xk+1,..., xn. If it = nil then L1 is made empty and L2 a copy of L. Finally L is made empty if it is not identical to L1 or L2.

**Aufgabe 3.2:** (5 Punkte)

*Josephus Problem*

Von dem jüdischen Geschichtsschreiber Josephus (1. Jhd.) ist ein interessantes mathematisches Problem überliefert:

N Personen stehen in einem Kreis und werden solange mit einem K-silbigen Abzählvers ausgezählt, bis nur noch eine Person übrigbleibt. Josephus interessierte, wie müssen er und seine Freunde stehen, damit sie alleine übrigbleiben. Lösen Sie das Josephus-Problem mit Hilfe der Klasse int_list aus Ausgabe 3.1.

Es sollen 16 Personen mit einem 5-silbigen Abzählreim ausgezählt werden. An welche Positionen müssen sich Josephus und seine vier Freunde stellen, damit sie übrigbleiben?
Tipp: Um diesen Todeskreis zu simulieren, lesen wir die Anzahl n der Leute ein und speicher die Zahlen 0 bis n-1 in einer Liste. Mit einem Item pos laufen wir in einer Schleife im Kreis herum; wir nennen es das aktuelle Item. Dieses lassen wir zu Beginn auf den ersten Container der Liste verweisen. Wir besorgen uns ein Item new_pos, das auf den Container verweist, welcher 5 Stellen weiter liegt; dieser Container wird in der nächsten Iteration gelöscht werden. Wir löschen den zum aktuellen Item zugehörigen Container aus der Liste. Dann wird das Item new_pos zum aktuellen Item. Die Schleife läuft so lange, bis nur noch 4 Personen in der Liste sind. Diese geben wir dann aus.