Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für theoretische Informatik
Dr. J. Giesen

26. Januar 2005

Informatik I für D-ITET Serie 13 WS 04/05

URL: http://www.ti.inf.ethz.ch/ew/courses/inf1-ITET/

Aufgabe 1 [Schnellübung – 20 Min.] (5 Punkte)

Material aus der Vorlesung

Programm: list.h

```
// Programm: list.h
// Doppelt verkettete Listen

namespace ifet {

  class ListElement {
  public:
    ListElement(int x, ListElement* p, ListElement* n);
    int data;
    friend class List;
  private:
    ListElement* prev_;
    ListElement* next_;
  };

  ListElement::ListElement(int x, ListElement* p, ListElement* n)
    : data(x), prev_(p), next_(n)
  {}

  class List {
  public:
    List();  // POST: Initialisiert zu leerer Liste.
    ~List(); // POST: Alle Listenelemente geloescht.

  private:
    // Kopieren verboten!
    List(const List&);
    List& operator=(const List&);

  public:
    void insert(int x, ListElement* i);
    // PRE: i zeigt auf ein Element der Liste
    //      und *i != sentinel
    // POST: fuege x vor i ein.

    void erase(ListElement* i);
    // PRE: i zeigt auf ein Element der Liste
    // POST: i aus Liste entfernt

  private:
    void destroy(ListElement* b, ListElement* e);
    // POST: alle Elemente aus der Liste, bis auf den
    //       sentinel, wurden geloescht, der belegte
    //       Speicher wurde freigegeben.

    ListElement sentinel;
  };

  List::List() : sentinel(0, 0, 0)
  // POST: Initialisiert zu leerer Liste.
  {
    sentinel.next_ = &sentinel;
    sentinel.prev_ = &sentinel;
  }

  List::~List()
  // POST: Alle Listenelemente geloescht.
  {
    destroy(sentinel.next_, &sentinel);
  }

  void List::destroy(ListElement* b, ListElement* e)
  // POST: alle Elemente aus der Liste, bis auf den
  //       sentinel, wurden geloescht, der belegte
  //       Speicher wurde freigegeben.
  {
    while (b != e) {
      ListElement* d = b;
      b = b->next_;
      delete d;
    }
  }

  void List::insert(int x, ListElement* i)
  // PRE: i zeigt auf ein Element der Liste
  //      und *i != sentinel
  // POST: fuege x vor i ein.
  {
    ListElement* n = new ListElement(x, i->prev_, i);
    n->prev_->next_ = n;
    n->next_->prev_ = n;
  }

  void List::erase(ListElement* i)
  // PRE: i zeigt auf ein Element der Liste
  // POST: i aus Liste entfernt
  {
    i->prev_->next_ = i->next_;
    i->next_->prev_ = i->prev_;
    delete i;
  }

} // namespace ifet
```