```
// Headerdatei Funktionen.h

#include <iostream>
#include <string>
using namespace std;

// Struktur knoten
struct knoten
{
  int    info;
  double info2;
  knoten *next;
};


// neues Element am Ende einfuegen
void einfuegen( int, knoten **, knoten ** );

// Liste ausgeben
void ausgeben( knoten * );

// Element suchen
knoten * suchen ( int, knoten * );

// Element loeschen
void loeschen ( knoten *, knoten * );

================================================================================

// Quellcode-Datei Funktionen.cpp beinhaltet Implementierung der Funktion

#include "Funktionen.h"


/*************************************
  Funktionen
*************************************/

// neues Element am Ende einfuegen
void einfuegen( int neuWert, knoten **anfang, knoten **ende )
{
  struct knoten *hilf = new knoten;  // erzeuge neuen Knoten
  hilf->info = neuWert;         // info-Wert wird neuWert
  hilf->info2 = neuWert * 0.5;
  hilf->next = NULL;           // next-Wert wird mit NULL initialisiert

  if( NULL == *anfang )
  {                   // Leere Liste
    *anfang = hilf;
    *ende = hilf;
  }
  else
  {                   // Liste nicht leer
    (*ende)->next = hilf;         // einfuegen am Ende
    *ende = hilf;           // neues Ende
  }
}
```

```
// Liste ausgeben
void ausgeben( knoten *anfang )
{
  struct knoten *hilf = anfang;     // hilf zeigt auf aktuellen Knoten
  cout << endl << "===> Liste ausgeben <===\n";
  while ( NULL != hilf )
  {
    //cout << hilf->info << " ";     // info-Teil des aktuellen Knotten ausgeben
    cout << "Adresse von hilf->info: " << &hilf->info << "  Wert davon: " << hilf-
>info << endl;
    cout << "Adresse von hilf->info2: " << &hilf->info2 << "  Wert davon: " <<
hilf->info2 << endl;
    cout << "Adresse von hilf->next: " << &hilf->next << "  Wert davon: " << hilf-
>next << endl;
    hilf = hilf->next;         // hilf auf Folgeelement setzen
  }
  cout << endl << endl;
}

// Element suchen
knoten * suchen ( int suchWert, knoten *anfang )
{
  struct knoten *suchElement = anfang;
  bool gefunden = false;

  while ( ( NULL != suchElement ) && ( false == gefunden ) )
  {
    if (suchElement->info == suchWert)
    {
      gefunden = true;
    }
    else
    {
      suchElement = suchElement->next; // suchElement auf Folgeelement setzen
    }
  }
  if ( false == gefunden )
  {
    suchElement = NULL;            // Element nicht gefunden
  }

  return suchElement;
}

// Element loeschen
void loeschen ( knoten *loeschElement, knoten *anfang )
{
  struct knoten *hilf;
  hilf = anfang;
  bool gefunden = false;

  if ( loeschElement == anfang)
  {
    anfang = anfang->next;         // neuer Anfang
    gefunden = true;
  }

  // suche Vorgaenger von p
```

```
  while ( ( hilf->next != loeschElement) && ( false == gefunden ) )
  {
    hilf = hilf->next;
  }

  if ( NULL != hilf->next )
  {
    hilf->next = loeschElement->next;  // verkette neu
    delete loeschElement;         // Speicherfreigabe
  }
}
```

==========

```
// Programm zu rekursiven dynamischen Datenstrukturen
// hier: Verkettete Listen
// Hauptprogramm

#include "Funktionen.h"


int main()
{
  struct knoten *anfang = NULL;     // Leere Liste
  struct knoten *ende   = NULL;     // Leere Liste
  int eingabe = 0;
  struct knoten *zeiger = NULL;

  while ( 0 <= eingabe )
  {
    cout << "Integer (Abbruch mit negativem Wert): ";
    cin >> eingabe;
    if ( 0 <= eingabe )
    {
      // Starte Funktion einfuegen
      einfuegen( eingabe, &anfang, &ende );
    }
  }
  if ( NULL == anfang )
  {
    cout << "Liste ist leer\n";
  }
  else
  {
    // Starte Funktion ausgeben
    ausgeben( anfang );
  }


  // Starte Funktion loeschen
  cout << "Integer, der geloescht werden soll: ";
  cin >> eingabe;
  zeiger = suchen( eingabe, anfang );
  if ( NULL != zeiger )
  {
```

```
    cout << "Adresse von zeiger->info: " << &zeiger->info << "  Wert davon: " <<
zeiger->info << endl;
    cout << "Adresse von zeiger->info2: " << &zeiger->info2 << "  Wert davon: " <<
zeiger->info2 << endl;
    cout << "Adresse von zeiger->next: " << &zeiger->next << "  Wert davon: " <<
zeiger->next << endl;
    ausgeben( anfang );
    loeschen( zeiger, anfang );
  }
  else
  {
    cout << endl << eingabe << " nicht in Liste vorhanden" << endl;
  }
  if ( NULL == anfang )
  {
    cout << "Liste ist leer\n";
  }
  else
  {
    // Starte Funktion ausgeben
    cout << "Liste ist nicht leer\n";
    ausgeben( anfang );
  }

  cout << "\nProgrammende\n";

  return 0;
}
```