

```

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

/* Struktur eines Listenelements */
struct FloatListElem {
    float data;
    FloatListElem *pNext;
};

/* Liste ausgeben mit gegebener Genauigkeit */
void fl_print(FloatListElem * const pAnchor, int precision=2)
{
    // Hilfszeiger auf den Anker (Beginn) der Liste
    FloatListElem *pHelp = pAnchor;
    int i=0;
    // Anzahl Nachkommastellen festlegen
    cout.precision(precision);
    // Liste bis zum Ende durchlaufen
    while(pHelp != NULL)
    {
        // nach 3 Zahlen einen Zeilenumbruch
        if(i%3 == 0 && i>0)
            cout << endl;
        // Darstellung im 15 Zeichen breiten Spalten
        // und in Gleitpunktdarstellung
        cout << setw(15) << fixed << pHelp->data;
        // naechstes Element betrachten
        pHelp = pHelp->pNext;
        i++;
    }
    if(i==0)
        cout << "Noch keine Elemente in der Liste!";
    cout << endl;
}

/* komplette Liste leeren */
void fl_delete(FloatListElem *& pAnchor)
{
    // Hilfszeiger auf den Anker (Beginn) der Liste
    FloatListElem *pHelp = pAnchor;
    // Liste bis zum Ende durchlaufen
    while(pHelp != NULL){
        // Zeiger auf zu loeschendes Element definieren
        FloatListElem *pDelete = pHelp;
        // naechstes Element betrachten
        pHelp = pHelp->pNext;
        // altes Element loeschen
        delete pDelete;
    }
    // Anker zurueck auf NULL setzen (leere Liste)
    pAnchor = NULL;
}

```

```

/* einen Wert an die Liste fuegen */
void fl_append(FloatListElem *& pAnchor, float data)
{
    // Hilfszeiger auf das Ende der Liste (erst auf Anfang)
    FloatListElem *pLast = pAnchor;
    // wenn Liste leer, dann neu initialisieren und Anker setzen
    if(pLast == NULL){
        pAnchor = new FloatListElem;
        pAnchor->data = data;
        pAnchor->pNext = NULL;
    // andernfalls neues Listenelement erzeugen und anhaengen
    } else {
        // an das Ende der Liste springen
        while(pLast->pNext != NULL)
            pLast = pLast->pNext;
        // neues Element erzeugen...
        FloatListElem *pNew = new FloatListElem;
        pNew->data = data;
        pNew->pNext = NULL;
        // und an das Ende haengen
        pLast->pNext = pNew;
    }
}

/* Element mit gegebenem Wert suchen und
   Zeiger auf Ergebnis zurueckliefern */
FloatListElem * fl_search(FloatListElem * const pAnchor, float data)
{
    // Suchzeiger definieren
    FloatListElem *pSearch = pAnchor;
    // Liste durchsuchen bis zum Ende
    if(pSearch != NULL){
        // erstes Element ist gesuchtes Element
        if(pSearch->data == data)
            return pSearch;
        // eines der anderen Elemente ist gesuchtes
        while(pSearch->pNext != NULL) {
            pSearch = pSearch->pNext;
            if(pSearch->data == data)
                return pSearch;
        }
    }
    // wenn nichts gefunden, dann NULL zurueckgeben
    return NULL;
};

```

```
/* Hauptprogrammfunktion */
int main()
{
    // Zeiger auf FloatListElem definieren
    FloatListElem *fl = NULL;
    // Liste mit 0.0 bis 5.0 fuellen
    for(float i=0; i <= 5.0; i=i+0.5)
        fl_append(fl,i);
    // Liste mit Genauigkeit 2 ausgeben
    fl_print(fl,2);
    // Element suchen und Wert zum Test ausgeben
    FloatListElem* search = fl_search(fl,4.5);
    // endl <=> '\n'
    if(search != NULL)
        cout << endl << search->data << endl;
    // Liste freigeben
    fl_delete(fl);
}
```