

```

/**
 * listops_class.cpp:
 * Klassenvariante der prozeduralen Listenimplementation (listops.cpp)
 */
#include <iostream>
#include <iomanip>
using namespace std;

/**
 * Deklaration von Strukturen und Klassen
 */
// Listenstruktur-Datentyp mit zwei Variablen
struct ListElem
{
    int data;                // Datenelement
    ListElem * pNext;       // Zeiger auf Nachfolger
};

class Liste
{
    // Klassen-interne Daten
private:
    ListElem *pA;

    // oeffentliche Daten
public:
    // Konstruktor
    Liste();
    // Element an eine Liste anhaengen
    void list_append(int to_append);
    // Element in eine Liste einsortieren
    void list_insert_sorted(int to_insert);
    // Element aus einer Liste loeschen
    void list_delete(int to_delete);
    // Element in einer Liste suchen
    ListElem * list_search(int to_search);
    // Liste komplett loeschen
    void list_delete_all();
    // Liste ausgeben (max. 5 Elemente pro Zeile)
    void list_print();
};

/**
 * Definition von Klassen-Methoden
 */

Liste :: Liste()
{
    pA = NULL;
}

void Liste :: list_append(int to_append)
{
    // selber Inhalt wie prozedurale Implementation
}

void Liste :: list_print()
{
    // selber Inhalt wie prozedurale Implementation
}

```

```

void Liste :: list_insert_sorted(int to_insert)
{
    // selber Inhalt wie prozedurale Implementation
}

void Liste :: list_delete(int to_delete)
{
    // selber Inhalt wie prozedurale Implementation
}

ListElem * Liste :: list_search(int to_search)
{
    // selber Inhalt wie prozedurale Implementation
}

void Liste :: list_delete_all()
{
    // selber Inhalt wie prozedurale Implementation
}

/**
 * Hauptprogrammfunktion
 */

int main()
{
    // Listenobjekt bereitstellen
    Liste l;
    // Element aus leerer Liste loeschen
    l.list_delete(5);
    // in eine leere Liste einfuegen
    l.list_insert_sorted(-2);
    // Elemente anhaengen
    for(int i=0; i<20; i+=2)
        l.list_append(i);
    // am Anfang der Liste einfuegen
    l.list_insert_sorted(-5);
    // am Ende der Liste einfuegen
    l.list_insert_sorted(1000);
    // in der Mitte der Liste einfuegen
    l.list_insert_sorted(5);
    // erstes Element loeschen
    l.list_delete(-5);
    // letztes Element loeschen
    l.list_delete(1000);
    // mittleres Element loeschen
    l.list_delete(6);
    // nicht vorhandenes Element loeschen
    l.list_delete(50);
    // Element suchen
    if(l.list_search(5))
        cout << "Element 5 gefunden!" << endl;
    else
        cout << "Element 5 nicht gefunden!" << endl;
    // Liste ausgeben
    l.list_print();
    // Liste loeschen
    l.list_delete_all();
    return 0;
}

```