```cpp
#include <string>
#include <vector>
#include <list>
#include <fstream>
#include <algorithm>
#include <iostream>

using std::cin;
using std::cout;
using std::istream;
using std::ostream;
using std::string;
using std::vector;
using std::list;

// ------------------------------------------------------------------------
// Neue Datentypen

struct Datum
{
    int Tag;              // Attribute
    int Monat;
    int Jahr;

    bool gueltig() const // Methode, die das Objekt nicht veraendert
    {
        if (Monat < 1 || Monat > 12)
            return false;
        if (Tag < 1 || Tag > 31)
            return false;
        return true;
    }
};

struct Person
{
    string Nachname;     // Attribute
    string Vorname;
    string Email;
    int    MatNr;
    Datum  Geburtstag;

    Datum Alter( Datum heute) const // Methode, die das Objekt nicht
       veraendert
    {
        Datum a;
        a.Jahr= heute.Jahr - Geburtstag.Jahr;
        a.Monat= heute.Monat - Geburtstag.Monat;
        if (a.Monat<0) { a.Monat+= 12; --a.Jahr; }
        a.Tag= heute.Tag - Geburtstag.Tag;
        if (a.Tag<0) { a.Monat+= 30; --a.Monat; }

        return a;
    }
};

typedef vector<Person> AdressbuchT;  // Typ-Alias
typedef list<Datum> GebListeT;


```

```
60 // ------------------------------------------------------------------------
61 // Funktionen zur Ausgabe
62
63 void Datum_schreiben(ostream& out, const Datum& datum)
64 {
65     out << datum.Tag << "." << datum.Monat << "."
66         << datum.Jahr;
67 }
68
69 void Person_schreiben(ostream& out, const Person& person)
70 {
71     out << person.Nachname << "␣" << person.Vorname << "␣"
72         << person.Email << "␣" << person.MatNr << "␣";
73     Datum_schreiben( out, person.Geburtstag);
74 }
75
76 void AB_schreiben(ostream& out, const AdressbuchT& AB)
77 {
78     for (int i= 0; i < AB.size(); i++) {
79         Person_schreiben( out, AB[i]);
80         out << "\n";
81     }
82 }
83
84 // ------------------------------------------------------------------------
85 // Funktionen zur Eingabe
86
87 void Datum_lesen(istream& in, Datum& datum)
88 {
89     in >> datum.Jahr;
90     in >> datum.Monat;
91     in >> datum.Tag;
92 }
93
94 void Person_lesen(istream& in, Person& person)
95 {
96     in >> person.Nachname;
97     in >> person.Vorname;
98     in >> person.Email;
99     in >> person.MatNr;
100    Datum_lesen( in, person.Geburtstag);
101 }
102
103 void AB_lesen(istream& in, AdressbuchT& AB)
104 {
105     Person p;
106     while (!in.eof()) {
107         Person_lesen( in, p);
108         if (in.good()) // damit letzte Person nicht doppelt auftaucht
109             AB.push_back( p);
110     }
111 }
112
113 // ------------------------------------------------------------------------
114 // Eingabe-/Ausgabeoperatoren
115
116 istream& operator>> (istream& in,  Datum& datum)       { Datum_lesen( in,
    datum); return in; }
117 ostream& operator<< (ostream& out, const Datum& datum) { Datum_schreiben(
    out, datum); return out; }
```

```
118
119 istream& operator>> (istream& in,  Person& person)       { Person_lesen( in,
        person); return in; }
120 ostream& operator<< (ostream& out, const Person& person) { Person_schreiben(
        out, person); return out; }
121
122 istream& operator>> (istream& in,  AdressbuchT& AB)       { AB_lesen( in, AB
       ); return in; }
123 ostream& operator<< (ostream& out, const AdressbuchT& AB) { AB_schreiben(
       out, AB); return out; }
124
125 // ---------------------------------------------------------------------
126
127 void geboren_in(int Monat, const AdressbuchT& AB, AdressbuchT& MonatAB)
128 {
129     for(int i= 0; i < AB.size(); i++)
130         if (AB[i].Geburtstag.Monat == Monat)
131             MonatAB.push_back( AB[i]);
132 }
133
134 void listeGueltigeGeb( const AdressbuchT& AB, GebListeT& geb)
135 {
136     for (int i= 0; i< AB.size(); i++)
137         if (AB[i].Geburtstag.gueltig() )
138             geb.push_back( AB[i].Geburtstag);
139 }
140
141 // Vergleichsoperator fuer Datum
142 bool operator<( const Datum& a, const Datum& b)
143 {
144     if (a.Jahr < b.Jahr)
145         return true;
146     else if (a.Jahr == b.Jahr) {
147         if (a.Monat < b.Monat)
148             return true;
149         else if (a.Monat == b.Monat) {
150             if (a.Tag < b.Tag)
151                 return true;
152         }
153     }
154     return false;
155 }
156
157 // Vergleichspraedikat fuer Person
158 bool geboren_vor(const Person& a, const Person& b)
159 {
160     return a.Geburtstag < b.Geburtstag;
161 }
162
163 // Hauptprogramm
164 int main()
165 {
166     Person Ich;
167     Ich.Nachname= "Gross";
168     Ich.Vorname= "Sven";
169     Ich.Email= "gross@igpm.rwth-aachen.de";
170     Ich.MatNr= 211054;
171     Ich.Geburtstag.Tag= 7;
172     Ich.Geburtstag.Monat= 5;
173     Ich.Geburtstag.Jahr= 1976;
```

```
174
175     AdressbuchT AB;
176     std::ifstream fin("AB.dat");
177     AB_lesen( fin, AB);  // bzw. fin >> AB;
178     fin.close();
179     AB.push_back( Ich);
180     cout << "Das␣Adressbuch␣hat␣" << AB.size() << "␣Eintraege:\n";
181     AB_schreiben( cout, AB);   // bzw. fin << AB;
182
183     cout << "Monat␣eingeben:␣";
184     int monat;
185     cin >> monat;
186     AdressbuchT AB2;
187     geboren_in( monat, AB, AB2);
188     cout << "Im␣" << monat << ".␣Monat␣des␣Jahres␣haben␣folgende␣Personen␣
           Geburtstag:\n";
189     AB_schreiben( cout, AB2);
190
191     GebListeT Geburtstage;
192     listeGueltigeGeb( AB, Geburtstage);
193     Geburtstage.sort();
194     cout << "Gueltige␣Geburtstage␣(sortiert):\n";
195     for (GebListeT::iterator it= Geburtstage.begin(),
196                              end= Geburtstage.end(); it != end; ++it)
197         cout << *it << endl;
198
199     cout << "Nach␣Geburtstag␣sortiertes␣Adressbuch:\n";
200     sort( AB.begin(), AB.end(), geboren_vor);
201     AB_schreiben( cout, AB);
202
203     return 0;
204 }
```