```cpp
/**
 * Progettare un algoritmo che permetta di gestire una rubrica telefonica.
 * L'algoritmo dovrà prevedere la gestione tramite menù dei seguenti aspetti:
 * a) caricamento di un nominativo nella rubrica in modo ordinato
 * (insertion sort);
 * b) ricerca del numero di telefono di un contatto attraverso il nominativo;
 * c) cancellazione di un contatto dalla rubrica;
 * d) visualizzazione dell'intera rubrica.
 * Si usino le funzioni per gestire tutti i distinti aspetti logici
 * dell'algoritmo in un'ottica di progettazione top-down.
 */

/*****************************************************************************/
/*                                main.cpp                                   */
/*****************************************************************************/

#include "phonebook.h"

int main() {
    char s; // menu choice
    tEntry pb[DIM];
    unsigned short n = 0; // first free position in phonebook

    do {
        Menu(s);

        switch (s){
            case '0':
                break;
            case '1':
                // phonebook loading
                Load(pb, n);
                break;
            case '2':
                // phonebook finding
                Find(pb, n);
                break;
            case '3':
                Delete(pb, n);
                break;
            case '4':
                Show(pb, n);
                break;
            default:
                cout << "Wrong choice" << endl;
        }
    } while (s != '0');

    return 0;
}

/*****************************************************************************/
/*                            phonebook.cpp                                  */
/*****************************************************************************/

#include "phonebook.h"

/**
 * Menù per selezionare l'operazione
 * @param s - scelta del menù (1, 2, 3, 0)
 */
void Menu(char& s) {
    cout << "RUBRICA TELEFONICA" << endl;
    cout << "1) Inserimento di un contatto" << endl;
    cout << "2) Ricerca di un numero telefonico tramite nominativo" << endl;
    cout << "3) Cancellazione di un contatto tramite nominativo e numero
```

```
    telefonico" << endl;
67      cout << "4) Visualizzazione dei contatti" << endl;
68      cout << "0) Uscita" << endl;
69      cout << "Scelta: ";
70      cin >> s;
71
72      return;
73  }
74
75  /**
76   * Phonebook loading
77   * @param pb - the phonebook
78   * @param n - first free position in phonebook
79   */
80  void Load(tEntry pb[], unsigned short& n) {
81
82      if (n < DIM) {
83          cout << "Nome: ";
84          cin >> pb[n].name;
85          cout << "Numero di telefono: ";
86          cin >> pb[n].phone;
87          PutInOrder(pb, n);
88          n++;
89      } else {
90          cout << "No free position in phonebook" << endl;
91      }
92  }
93
94  /**
95   * Put recursively in order the entries
96   * @param pb - the phonebook
97   * @param i - position of the entry to put currently in order
98   */
99  void PutInOrder(tEntry pb[], int i) {
100     tEntry temp;
101
102     for (int j = 0; j < i; j++) {
103         if (pb[j].name > pb[i].name) {
104             temp = pb[i];
105             pb[i] = pb[j];
106             pb[j] = temp;
107         }
108     }
109
110     return;
111 }
112
113 /**
114  * Finds a phone number giving the name
115  * @param pb - the phonebook
116  * @param n - first free position in phonebook
117  */
118 void Find(tEntry pb[], unsigned short n) {
119     string name;
120     int whereIs;
121
122     if (n) {
123         cout << "Nominativo: ";
124         cin >> name;
125         whereIs = Binary(pb, 0, n, name);
126         if (whereIs != -1) {
127             cout << "Numero telefonico: " << pb[whereIs].phone << endl;
128         } else {
129             cout << "Contatto non presente in rubrica" << endl;
130         }
131
```

```
132         } else {
133             cout << "Non ci sono contatti in rubrica" << endl;
134         }
135     }
136 
137     /**
138      * Recursively binary searching
139      * @param pb - the phonebook
140      * @param start - starting index in the array
141      * @param end - ending index in the array
142      * @param name - name to find in the array
143      * @return position in the array that matches the given name; -1 if the given
144      *         name in not found
145      */
146     int Binary(tEntry pb[], int start, int end, string name) {
147         int medium = (start + end) / 2;
148 
149         if (start > end) {
150             return -1;
151         } else {
152             if (name == pb[medium].name) {
153                 return medium;
154             } else {
155                 if (name > pb[medium].name) {
156                     return Binary(pb, medium + 1, end, name);
157                 } else {
158                     return Binary(pb, start, medium - 1, name);
159                 }
160             }
161         }
162     }
163 
164     /**
165      * Delete a phonebook entry
166      * @param pb - the phonebook
167      * @param n - first free position in phonebook
168      */
169     void Delete(tEntry pb[], unsigned short& n) {
170         string name;
171         int whereIs;
172 
173         if (n) {
174             cout << "Nominativo: ";
175             cin >> name;
176             whereIs = Binary(pb, 0, n, name);
177             if (whereIs != -1) {
178                 for (int i = whereIs; i < (n - 1); i++) {
179                     pb[i] = pb[i + 1];
180                 }
181                 n--;
182             } else {
183                 cout << "Contatto non presente in rubrica" << endl;
184             }
185 
186         } else {
187             cout << "Non ci sono contatti in rubrica" << endl;
188         }
189     }
190 
191     /**
192      * Showes the phonebook entries
193      * @param pb - the phonebook
194      * @param n - first free position in phonebook
195      */
196     void Show(tEntry pb[], unsigned short n) {
197 
```

```
198         for (int i = 0; i < n; i++) {
199             cout << pb[i].name << " " << pb[i].phone << endl;
200         }
201         return;
202     }
203
204     /*****************************************************************************/
205     /*                              phonebook.h                                  */
206     /*****************************************************************************/
207
208     #ifndef PHONEBOOK_H
209     #define PHONEBOOK_H
210
211     #include <cstdlib>
212     #include <iostream>
213     #include <string>
214
215     #define DIM 100
216
217     using namespace std;
218
219     struct tEntry {
220         string name;
221         string phone;
222     };
223
224     void Menu(char & s);
225     void Load(tEntry pb[], unsigned short& n);
226     void Find(tEntry pb[], unsigned short n);
227     void Delete(tEntry pb[], unsigned short& n);
228     void Show(tEntry pb[], unsigned short n);
229     void PutInOrder(tEntry pb[], int i);
230     int Binary(tEntry pb[], int start, int end, string name);
231
232
233     #endif /* PHONEBOOK_H */
```