```
// clsea.cpp - WORD-BASED CONTACT LOG SEARCH PROGRAM
//
// USAGE
//
// clsea <keyword>
//
// <keyword>    A space separated keyword list
//
// MODULE INDEX
// NAME         CONTENTS
// StrCmp       Case insensitive comparison between string
// KeywordCmp   Compare word with list of keyword
// AddKeyword   Add keyword into keyword set
// OutKeyword   Output keywords used to produce report
// OutLog       Output log record
// main         Main line
//
// MAINTENANCE HISTORY
// DATE     PROGRAMMER AND DETAILS
// 20-10-16 SHT Original
//
//--------------------------------------------------------------------------

#include <string>       // C++ string declarations
#include <cstring>      // C-style string manipulation functions
#include <cstdlib>      // C-style standard library
#include <iostream>     // C++ input/output stream
#include <fstream>      // C++ file input/output stream
#include <sstream>      // C++ string stream
#include <set>          // C++ set declarations
#include <time.h>       // C time declaration
using namespace std;    // Expand the standard namespace
exec sql include sqlca; // Include SQL communications area

//--------------------------------------------------------------------------

// GLOBAL DATA

size_t      cnt;        // Matched record count
set<string> keywordSet; // Keywords set

//--------------------------------------------------------------------------

// CASE INSENSITIVE COMPARISON BETWEEN STRING

bool
StrCmp(
    string first,
    string second)
{
    int i;              // General purpose index

    if (first.length() != second.length())
        return false;

    for (i = 0; i < first.length(); i++)
        if (tolower(first[i]) != tolower(second[i]))
```

```
            return false;

    return true;
}

//---------------------------------------------------------------------------

// COMPARE WORD WITH LIST OF KEYWORDS

bool
KeywordCmp (
    string  word)                   // Word to compare
{
    set<string>::iterator   itr;    // Keyword set iterator

    for (itr = keywordSet.begin(); itr != keywordSet.end(); ++itr)
        if (StrCmp(word, *itr))
            return true;

    return false;
}

//---------------------------------------------------------------------------

// ADD KEYWORD INTO KEYWORD SET

void
AddKeyword(
    string keyword)
{
    string      line;                   // Line string
    string      word;                   // Word string
    ifstream    input ("RELATED.TXT");  // Input file stream

    // Validate whitespace in keyword

    if (keyword.find(' ') != string::npos) {
        cerr << "Error: keyword contains whitespace.\n";
        exit (1);
    }

    // Validate RELATED.TXT can be open

    if (!input) {
        cerr << "Error: fail to open RELATED.TXT\n";
        exit (1);
    }

    // Add keyword into keyword list

    keywordSet.insert (keyword);

    // Add related keywords into keyword list

    while (getline (input, line)) {
        // Looking for related word
```

```
        if (line.find(" " + keyword + " ") == string::npos)
            continue;

        // Add related words

        stringstream ss (line);

        while (ss >> word)
            keywordSet.insert (word);
    }
}

//-----------------------------------------------------------------------------

// GET THE CURRENT TIME STRING

const string
TimeStr()
{
    char        buf[20];        // Output buffer
    struct tm   tstruct;
    time_t      now = time(0);  // Time structure

    tstruct = *localtime(&now);
    strftime(buf, sizeof(buf), "%Y-%m-%d %X", &tstruct);

    return buf;
}

//-----------------------------------------------------------------------------

// OUTPUT KEYWORDS USED TO PRODUCE REPORT

void
OutKeyword ()
{
    set<string>::iterator   itr;    // Keyword set iterator

    cout << "<p align=\"center\">" << '\n';
    cout << "Keywords: ";

    for (itr = keywordSet.begin(); itr != keywordSet.end(); ++itr)
        cout << *itr << " ";

    cout << "</p>" << "\n";
}

//-----------------------------------------------------------------------------

// OUTPUT LOG RECORD

void
OutLog (
    long    xLogKey)                // Log key
{
    int     i;                      // General purpose index
```

```
    exec sql begin declare section;
        long    logKey;             // Log record key
        long    buildingKey;        // Building record key
        long    tenantKey;          // Tenant record key
        long    guardKey;           // Guard record key
        long    extNo;              // Extension number
        char    logTime[21+1];      // Log time
        char    buildingName[80+1]; // Building name
        char    unitNo[20+1];       // Unit number
        char    tenantName[80+1];   // Tenant name
        char    tenantPhone[20+1];  // Tenant phone number
        char    guardName[80+1];    // Guard name
        char    guardPhone[20+1];   // Guard phone number
        char    note[128+1];        // Contact log content
    exec sql end declare section;

    logKey = xLogKey;

    // Fetch data from database

    exec sql select VARCHAR_FORMAT(logTime,'YYYY-MM-DD HH24:MI:SS'),
                    tenantKey, guardKey, note
            into    :logTime,
                    :tenantKey, :guardKey, :note
            from    logs
            where   logKey = :logKey;
    exec sql select buildingKey, unitNo, tenantName, tenantPhone
            into    :buildingKey, :unitNo, :tenantName, :tenantPhone
            from    tenants
            where   tenantKey = :tenantKey;
    exec sql select buildingName
            into    :buildingName
            from    buildings
            where   buildingKey = :buildingKey;
    exec sql select guardName, guardPhone
            into    :guardName, :guardPhone
            from    guards
            where   guardKey = :guardKey;

    // Output the data in html table

    cout << "<tr>" << '\n';
    cout << "<td>" << ++cnt << "</td>" << '\n';
    cout << "<td>" << buildingName << "</td>" << '\n';
    cout << "<td>" << unitNo << "</td>" << '\n';
    cout << "<td>" << tenantName << "</td>" << '\n';
    cout << "<td>" << tenantPhone << "</td>" << '\n';
    cout << "<td>" << guardName << "</td>" << '\n';
    cout << "<td>" << guardPhone << "</td>" << '\n';
    cout << "<td>" << logTime << "</td>" << '\n';

    // Output log content

    i = 0;

    cout << "<td>";
    while (1) {
```

```
        cout << note;

        extNo = i++;

        exec sql select extension
                into    :note
                from    extensions
                where   logKey = :logKey and
                        extNo = :extNo;
        if (SQLCODE != 0) break;
    }

    cout << "</td>" << '\n';
    cout << "</tr>";
}

//------------------------------------------------------------------------------

// MAIN LINE

int
main(
    int     argc,                   // Argument count
    char    *argv[])                // Argument
{
    string      word;               // Word string
    int         i, j;               // General purpose index

    for (i = 1; i < argc; i++)
        AddKeyword (string(argv[i]));

    exec sql connect to cldb;

    exec sql begin declare section;
        long    logKey;
        long    extNo;
        short   extInd;             // Indicator for extension
        char    note[128+1];        // Contact log note
    exec sql end declare section;

    // Output HTML header and body

    cout << "<html>" << '\n';
    cout << "<head>" << '\n';
    cout << "<title>Search Contact Log</title>" << '\n';
    cout << "</head>" << '\n';
    cout << "<body>" << '\n';

    // Output report header

    cout << "<p align=\"center\">";
    cout << "CONTACT LOG SEARCH REPORT";
    cout << "</p>" << '\n';

    cout << "<p align=\"center\">";
    cout << TimeStr ();
    cout << "</p>" << '\n';
```

```
    // Output keyword used to produce report

    OutKeyword ();

    // Output table header

    cout << "<table border=\"1\">" << '\n';
    cout << "<tr>" << '\n';
    cout << "<th>No.</th>" << '\n';
    cout << "<th>Building Name</th>" << '\n';
    cout << "<th>Unit Number</th>" << '\n';
    cout << "<th>Tenant Name</th>" << '\n';
    cout << "<th>Tenant Phone</th>" << '\n';
    cout << "<th>Guard Name</th>" << '\n';
    cout << "<th>Guard Phone</th>" << '\n';
    cout << "<th>Time</th>" << '\n';
    cout << "<th>Contact Log</th>" << '\n';
    cout << "</tr>" << '\n';

    exec sql declare noteCur cursor for
            select  logKey, note
            from    logs;
    exec sql open noteCur;
    while (1) {
    fetch_record:
        exec sql fetch  noteCur
                into    :logKey, :note;
        if (SQLCODE != 0) break;

        i = 0;
        word = "";

        while (1) {
            for (j = 0; j < 128; j++) {
                if (note[j] == ' ') {
                    if (KeywordCmp(word)) {
                        OutLog(logKey);
                        goto fetch_record;
                    }

                    word = "";

                    continue;
                }

                if (note[j] == '\0') {
                    if (KeywordCmp(word))
                        OutLog(logKey);

                    break;
                }

                word += note[j];
            }

            extNo = i++;
```

```
            exec sql select extension
                    into    :note indicator :extInd
                    from    extensions
                    where   logKey = :logKey and
                            extNo = :extNo;
            if (SQLCODE != 0) break;
        }
    }
    exec sql close noteCur;

    cout << "</table>" << '\n';

    // Output matched record count

    cout << "<p>" << '\n';
    cout << "Total matched record: " << cnt << '\n';
    cout << "</p>" << '\n';

    cout << "</body>" << '\n';
    cout << "</html>" << '\n';

    return 0;
}
```