```
// SortUpdates.cpp - SORT UPDATE ENTRIES BY FILE NAME
//
// MAINTENANCE HISTORY
// DATE         PROGRAMMER AND DETAILS
// 24-08-07     JS      Original
//
//-----------------------------------------------------------------------------

#include <cstdlib>              // C-style standard library
#include <string>               // C++ string declarations
#include <iostream>             // C++ I/O stream declarations
#include <fstream>              // C++ file stream declarations
#include <vector>               // C++ vector declarations
#include <algorithm>            // C++ algorithms
using namespace std;            // Expand the standard namespace

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

// UPDATE STRUCTURE

struct Update_t {
        string          updateName;     // Updated file name
        int             updateRelease;  // Release number
        int             updateLevel;    // Level number
        string          updateData;     // Update data

        // Compare Updates

        bool
        operator < (
                const Update_t &update) // Value to compare
        const
        {
                if (updateName < update.updateName) return 1;
                if (updateName > update.updateName) return 0;
                if (updateRelease < update.updateRelease) return 1;
                if (updateRelease > update.updateRelease) return 0;
                return updateLevel < update.updateLevel;
        }
};

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

// UPDATE VECTOR TYPE DEFINITION

typedef vector<Update_t> UpdateVec_t;

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

// MAIN LINE

int
main (
        int             argc,           // Argument count
        char            *argv[])        // Argument value pointers
{
        ifstream        ifs;            // Input file stream
        Update_t        updateRec;      // Update record
        UpdateVec_t     updateVec;      // Update vector
        string          line;           // Input line
        int             j;              // General purpose index
        size_t          i;              // General purpose index

        // Read the updates from the various files
```

```
for (j = 1; j < argc; j++) {

        // Open the file

        ifs.clear();
        ifs.open (argv[j]);
        if ( ! ifs) {
                cerr << "Error: cannot open " << argv[j] << '\n';
                exit (1);
        }

        // Process the lines in the input file

        getline (ifs, line);
        while (line[0] == '=') {

                // Interpret a header line

                i = 0;
                while (line[i] == '=') i ++ ;
                while (line[i] == ' ') i ++ ;
                updateRec.updateName = "";
                do {
                        updateRec.updateName += line[i];
                        i ++ ;
                } while (line[i] != ' ' && line[i] != '\0');
                while (line[i] == ' ') i ++ ;
                updateRec.updateRelease = 0;
                if ( ! isdigit(line[i])) {
                        cerr << "Error: header format error\n";
                        exit (1);
                }
                do {
                        updateRec.updateRelease =
                                updateRec.updateRelease * 10
                                        + line[i] - '0';
                        i ++ ;
                } while (isdigit(line[i]));
                if (line[i] != '.') {
                        cerr << "Error: header format error\n";
                        exit (1);
                }
                i ++ ;
                updateRec.updateLevel = 0;
                if ( ! isdigit(line[i])) {
                        cerr << "Error: header format error\n";
                        exit (1);
                }
                do {
                        updateRec.updateLevel =
                                updateRec.updateLevel * 10
                                        + line[i] - '0';
                        i ++ ;
                } while (isdigit(line[i]));
                while (line[i] == ' ') i ++ ;
                if (line[i] != '=') {
                        cerr << "Error: header format error\n";
                        exit (1);
                }

                // Load the header and change lines into updateData

                updateRec.updateData = "";
```

```
                do {
                        updateRec.updateData += line + '\n';
                        getline (ifs, line);
                } while (line[0] != '=' && line[0] != '\0');
                updateVec.push_back (updateRec);
        }

        // Check for files that do not start with a header line

        if ( ! ifs.eof()) {
                cerr << "Error: data format error\n";
                exit (1);
        }

        // Close the change listing

        ifs.close();
    }

    // Sort the changes in the required order

    sort (updateVec.begin(), updateVec.end());

    // Emit the changes

    for (i = 0; i < updateVec.size(); i++)
        cout << updateVec[i].updateData;

    return 0;
}
```