```
// LogGen.cpp - CONTACT LOG DATABASE GENERATOR
//
// MODULE INDEX
// NAME         CONTENTS
// GenLog       Generate contact log content
// main         Main line
//
// MAINTENANCE HISTORY
// DATE     PROGRAMMER AND DETAILS
// 24-10-16 SHT Original
//
//-----------------------------------------------------------------------------

#include <cstring>      // C-style string manipulation functions
#include <cstdlib>      // C-style standard library
#include <iostream>     // C++ input/output streams
#include <vector>       // C++ vector declarations
using namespace std;    // Expand the standard namespace
exec sql include sqlca; // Include SQL communications area

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

// GENERATE CONTACT LOG CONTENT

void
GenLog()
{
    FILE        *fp;
    char        c;              // Char
    char        *p;             // General purpose pointer
    size_t      len;            // Log length

    exec sql begin declare section;
        long    nextLogKey;     // Next log key
        long    logKey;         // Log key
        long    nextExtNo;      // Next extension no
        long    extNo;          // Extension no
        long    tenantKey;      // Tenant key
        long    guardKey;       // Guard key
        char    note[128+1];    // Log note
    exec sql end declare section;

    nextLogKey = 1;
    nextExtNo = 0;
    logKey = nextLogKey++;
    guardKey = lrand48() % 10 + 1;
    tenantKey = lrand48() % 100 + 1;
    p = note;
    len = 0;

    fp = fopen ("content.txt", "r");
    if (fp == NULL) {
        cerr << "Error: fail to open content file.\n";
        exit (1);
    }

    do {
```

```
        c = fgetc (fp);

        // Process new line

        if (c == '\n' && len > 0) {
            *p++ = '\0';

            if (nextExtNo > 0) {
                exec sql insert into extensions (
                    logKey, extNo, extension
                ) values (
                    :logKey, :extNo, :note
                );
            } else {
                exec sql insert into logs (
                    logKey,
                    logTime,
                    tenantKey, guardKey, note
                ) values (
                    :logKey,
                    CURRENT TIMESTAMP -
                        (RAND() * 1000000) SECONDS,
                    :tenantKey, :guardKey, :note
                );
            }

            exec sql commit work;

            logKey = nextLogKey++;
            nextExtNo = 0;
            guardKey = lrand48() % 10 + 1;
            tenantKey = lrand48() % 100 + 1;
            p = note;
            len = 0;

            continue;
        }

        *p++ = c;
        len++;

        if (len == 128) {
            *p++ = '\0';

            if (nextExtNo > 0) {
                exec sql insert into extensions (
                    logKey, extNo, extension
                ) values (
                    :logKey, :extNo, :note
                );
            } else {
                exec sql insert into logs (
                    logKey,
                    logTime,
                    tenantKey, guardKey, note
                ) values (
                    :logKey,
```

```
                    CURRENT TIMESTAMP -
                        (RAND() * 10000000) SECONDS,
                    :tenantKey, :guardKey, :note
                );
            }

            extNo = nextExtNo++;
            p = note;
            len = 0;
        }
    } while (c != EOF);

    fclose (fp);
}

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

// MAIN LINE

int
main ()
{
    size_t      i;                  // General purpose index

    exec sql begin declare section;
        long    nextBuildingKey;    // Next building key
        long    buildingKey;        // Building key
        long    nextTenantKey;      // Next tenant key
        long    tenantKey;          // Tenant key
        long    nextGuardKey;       // Next guard key
        long    guardKey;           // Guard key
        long    nextLogKey;         // Next log key
        char    buildingName[80];   // Building name
        char    unitNo[20];         // Unit no
        char    tenantName[80];     // Tenant name
        char    tenantPhone[20];    // Tenant phone number
        char    guardName[80];      // Guard name
        char    guardPhone[20];     // Guard phone number
    exec sql end declare section;

    // Connect to the database

    exec sql connect to cldb;

    // Drop tables

    exec sql whenever sqlerror continue;
    exec sql drop table buildings;
    exec sql drop table tenants;
    exec sql drop table guards;
    exec sql drop table logs;
    exec sql drop table extensions;
    exec sql commit work;

    // Jump to DbError whenever an SQL error occurs

    exec sql whenever sqlerror goto DbError;
```

```
    // Create the database tables

    exec sql create table buildings (
        buildingKey integer not null,
        buildingName    char (80) not null
    );
    exec sql create unique index buildingKeyInd on
        buildings (buildingKey);
    exec sql create table tenants (
        tenantKey   integer not null,
        buildingKey integer not null,
        unitNo      char (20) not null,
        tenantName  char (80) not null,
        tenantPhone char (20) not null
    );
    exec sql create unique index tenantsKeyInd on
        tenants (tenantKey);
    exec sql create table guards (
        guardKey    integer not null,
        guardName   char(80) not null,
        guardPhone  char(20) not null
    );
    exec sql create unique index guardKeyInd on
        guards (guardKey);
    exec sql create table logs (
        logKey      integer not null,
        logTime     timestamp not null,
        tenantKey   integer not null,
        guardKey    integer not null,
        note        varchar (128)
    );
    exec sql create unique index logKeyInd on
        logs (logKey);
    exec sql create table extensions (
        logKey      integer not null,
        extNo       integer not null,
        extension   varchar (128) not null
    );

    // Initialise the random number generator

    srand48 (20360L);

    // Initialize the next id for records

    nextBuildingKey = 1;
    nextTenantKey = 1;
    nextGuardKey = 1;
    nextLogKey = 1;

    for (i = 0; i < 5; i++) {
        buildingKey = nextBuildingKey++;
        sprintf (buildingName, "Building%d", buildingKey);
        exec sql insert into buildings (
            buildingKey, buildingName
        ) values (
```

```
            :buildingKey, :buildingName
        );
    }
    exec sql commit work;

    for (i = 0; i < 100; i++) {
        tenantKey = nextTenantKey++;
        buildingKey = lrand48() % 5 + 1;
        sprintf (unitNo, "Unit%ld", lrand48() % 20 + 1);
        sprintf (tenantName, "Tenant%d", tenantKey);
        sprintf (tenantPhone, "%d", 123456);
        exec sql insert into tenants (
            tenantKey, buildingKey,
            unitNo, tenantName, tenantPhone
        ) values (
            :tenantKey, :buildingKey,
            :unitNo, :tenantName, :tenantPhone
        );
    }
    exec sql commit work;

    for (i = 0; i < 10; i++) {
        guardKey = nextGuardKey++;
        sprintf (guardName, "Guard%d", guardKey);
        sprintf (guardPhone, "%d", 123456);
        exec sql insert into guards (
            guardKey, guardName, guardPhone
        ) values (
            :guardKey, :guardName, :guardPhone
        );
    }
    exec sql commit work;

    GenLog();

    return 0;

DbError:
    cerr << "Error(" << __LINE__ << "): SQLCODE=" << SQLCODE << '\n';
    return 1;
}
```