```
// MytelGen.cpp - MYTEL DATA GENERATOR
//
// MODULE INDEX
// NAME                 CONTENTS
// main                 Main line
//
// MAINTENANCE HISTORY
// DATE         PROGRAMMER AND DETAILS
// 29-08-05     JS      Original
//
//-----------------------------------------------------------------------------

#include <iostream>             // C++ input/output streams
#include <iomanip>              // C++ input/output manipulators
#include <sstream>              // C++ string stream declarations
#include <vector>               // C++ vectors
#include <string>               // C++ style strings
#include <functional>           // C++ function objects
#include <cstdlib>              // C-style standard library
#include <cstring>              // C-style string manipulation functions
using namespace std;            // Expand standard namespace to global scope
exec sql include sqlca;         // Include SQL communications area

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

// DEFINITIONS

static const size_t     SUB_CNT = 10000;
                                // Number of subscribers
static const size_t     CALL_CNT = 1000000;
                                // Number of calls
static const long       MIN_SUB_ID = 100000;
                                // Minimum subscriber identifier
static const long       MAX_SUB_ID = 500000000;
                                // Maximum subscriber identifier
static const size_t     SUB_NAME_LEN = 4;
                                // Subscriber's name length
static const long       MIN_SUB_BAL = 0;
                                // Minimum subscriber's balance
static const long       MAX_SUB_BAL = 400000;
                                // Maximum subscriber's balance
static const size_t     CITY_CODE_LEN = 3;
                                // City code length
static const long       START_TIME = 1072915200;
                                // Start time (00:00 01-01-04)
static const long       MIN_INTER_CALL_TIME = 0;
                                // Minimum inter-call time
static const long       MAX_INTER_CALL_TIME = 5;
                                // Maximum inter-call time
static const long       MIN_CALL_DURATION = 10;
                                // Minimum call duration
static const long       MAX_CALL_DURATION = 20*60;
                                // Maximum call duration
static const size_t     CALLED_NO_LEN = 20;
                                // Called number length

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

// CITY CODES

static const char CITY_CODE_ARR [] [CITY_CODE_LEN+1] = {
        "KUL", "SIN", "BKK"
};
static const size_t CITY_CODE_CNT =
        sizeof(CITY_CODE_ARR) / sizeof(CITY_CODE_ARR[0]);

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

// SUBSCRIBER STATUS STRUCTURE

struct SubStat_t {
        long            ssSubId;        // Subscriber identifier
        size_t          ssBaseCityInd;  // Base city index
        size_t          ssLastCityInd;  // Last `from city' index
};

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

// GLOBAL DATA
```

```
SubStat_t       subStatArr[SUB_CNT];    // Subscriber status array

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

// MAIN LINE

int
main ()
{
        size_t          i, j, k, l;     // General purpose indexes
        size_t          nameWordLen;    // Name word length
        size_t          fromCityInd;    // `From city' index
        SubStat_t       *ss;            // Subscriber status row pointer

        exec sql begin declare section;
                long            subId;          // Subscriber identifier
                char            subName[40+1];  // Subscriber name
                long            subBal;         // Subscriber's balance
                long            curTime;        // Current time
                char            fromCity[3+1];  // From city code
                char            calledNo[20+1]; // Called number
                long            disconTime;     // Disconnect time
        exec sql end declare section;

        // Vector to DbError whenever an SQL error occurs

        exec sql whenever sqlerror goto DbError;

        // Connect to the Customer Database

        exec sql connect to mytel;

        // Create the database tables

        // Subscribers Table

        exec sql create table subscribers (
                subId           integer not null,
                subName         char(40) not null,
                subBal          integer not null
        ) not logged initially;
        exec sql create unique index subIdInd on
                subscribers (subId);

        // Call Register

        exec sql create table callRegister (
                crSubId         integer not null,
                crFromCity      char(3) not null,
                crCalledNo      char(20) not null,
                crConnectTime   integer not null,
                crDisconTime    integer not null
        ) not logged initially;
        exec sql create index callRegTimeInd on
                callRegister (crConnectTime);

        // Initialise the random number generator

        srand48 (20360L);

        // Generate the subscriber identifiers and load the initial cities

        for (i = 0; i < SUB_CNT; i++) {
                ss = subStatArr + i;

                // Select a unique, random subscriber identifier

                do {
                        subId = lrand48() % (MAX_SUB_ID - MIN_SUB_ID + 1)
                                + MIN_SUB_ID;
                        j = 0;
                        while (j < i && subStatArr[j].ssSubId != subId)
                                j ++ ;
                } while (j < i);

                // Generate a name for the subscriber

                k = 0;
```

```
        for (j = 0; j < 3; j++) {
                nameWordLen = lrand48() % 3 + 3;
                for (l = 0; l < nameWordLen; l++)
                        subName[k++] = lrand48() % ('Z'-'A'+ 1) + 'A';
                subName[k++] = ' ';
        }
        subName[--k] = '\0';

        // Generate a balance for the subscriber

        subBal = lrand48() % (MAX_SUB_BAL - MIN_SUB_BAL + 1)
                + MIN_SUB_BAL;

        // Load the subscriber identifier into the subscriber
        // status array

        ss->ssSubId = subId;

        // Select a base and last city for the subscriber

        ss->ssBaseCityInd = lrand48() % CITY_CODE_CNT;
        ss->ssLastCityInd = lrand48() % CITY_CODE_CNT;

        // Insert the subscriber into the subscriber's table

        exec sql insert into subscribers (
                subId, subName, subBal
        ) values (
                :subId, :subName, :subBal
        );
}

// Generate calls

curTime = START_TIME;
for (i = 0; i < CALL_CNT; i++) {

        // Advance the current time

        curTime += lrand48() %
                (MAX_INTER_CALL_TIME - MIN_INTER_CALL_TIME + 1)
                        + MIN_INTER_CALL_TIME;

        // Randomly select a subscriber

        j = lrand48() % SUB_CNT;
        ss = subStatArr + j;

        // Randomly select a `from city'.  Make it an 90% chance
        // that the call will be made from the last city and an
        // 80% chance that the next city will be the base city
        // if the last city was not the subscriber's base city.

        if (lrand48() % 100 < 90) {
                fromCityInd = ss->ssLastCityInd;
        } else if (ss->ssLastCityInd != ss->ssBaseCityInd) {
                if (lrand48() % 100 < 80) {
                        fromCityInd = ss->ssBaseCityInd;
                } else {
                        fromCityInd = lrand48() % (CITY_CODE_CNT-2);
                        if (ss->ssLastCityInd < ss->ssBaseCityInd) {
                                if (fromCityInd >= ss->ssLastCityInd)
                                        fromCityInd ++ ;
                                if (fromCityInd >= ss->ssBaseCityInd)
                                        fromCityInd ++ ;
                        } else {
                                if (fromCityInd >= ss->ssBaseCityInd)
                                        fromCityInd ++ ;
                                if (fromCityInd >= ss->ssLastCityInd)
                                        fromCityInd ++ ;
                        }
                }
        } else {
                fromCityInd = lrand48() % (CITY_CODE_CNT-1);
                if (fromCityInd >= ss->ssBaseCityInd)
                        fromCityInd ++ ;
        }

        // Randomly generate a called number
```

```
                for (k = 0; k < CALLED_NO_LEN; k++)
                        calledNo[k] = lrand48() % ('Z'-'A'+ 1) + 'A';
                calledNo[CALLED_NO_LEN] = '\0';

                // Calculate a disconnect time

                disconTime = curTime + lrand48() %
                        (MAX_CALL_DURATION - MIN_CALL_DURATION + 1)
                                + MIN_CALL_DURATION;

                // Update the subscriber status table

                ss->ssLastCityInd = fromCityInd;

                // Insert the Call Register row.

                subId = ss->ssSubId;
                strcpy (fromCity, CITY_CODE_ARR[fromCityInd]);
                exec sql insert into callRegister (
                        crSubId, crFromCity, crCalledNo,
                        crConnectTime, crDisconTime
                ) values (
                        :subId, :fromCity, :calledNo,
                        :curTime, :disconTime
                );
        }

        // And that's all

        exec sql commit work;
        return 0;

        // Process database errors

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