```
// TeamGen.cpp - GENERATE DATA FOR TEAM SELECTION
//
// MODULE INDEX
// NAME                         CONTENTS
// RandName                     Generate a pseudo-random 20-char name
// main                         Main line
//
// MAINTENANCE HISTORY
// DATE         PROGRAMMER AND DETAILS
// 02-10-06     JS      Original
// 27-10-06     ELT     Updated module index
//
//-----------------------------------------------------------------------------

#include <iostream>             // C++ I/O stream declarations
#include <set>                  // C++ set declarations
#include <cstring>              // C-style string manipulation functions
#include <cstdlib>              // C standard library
using namespace std;            // Expand the standard namespace
exec sql include sqlca;         // SQL communications area

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

// DEFINITIONS

static const size_t N_PROGS = 75;
                                // Number of programmers
static const long FIRST_PROG_NUM = 100000;
                                // First programmer number
static const size_t N_PROJS = 3200;
                                // Number of projects
static const long FIRST_PROJ_NUM = 500000;
                                // First project number
static const size_t MIN_PROJS_PER_PROG = 3;
                                // Minimum projects per programmer
static const size_t MAX_PROJS_PER_PROG = 6;
                                // Maximum projects per programmer
static const double MAX_PROJ_VAL = 10.0;
                                // Maximum project value

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

// GENERATE A PSEUDO-RANDOM 20-CHARACTER NAME

void
RandName (
        char            *name)          // Name to load
{
        size_t          i, j;           // General purpose indices
        size_t          wordLen;        // Word length
        char            *p;             // Formatting pointer

        p = name;
        for (i = 0; i < 3; i++) {
                wordLen = rand() % 3 + 3;
                if (p != name) *p++ = ' ';
                for (j = 0; j < wordLen; j++)
                        *p++ = static_cast<char>(rand() % ('Z'-'A'+1) + 'A');
        }
        *p = '\0';
}

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

// MAIN LINE
```

```
int
main ()
{
        size_t          i;              // General purpose index
        size_t          projCnt;        // Number of projects for this prog
        set<long>       projSet;        // Project set
        set<long>       activeProj;     // Active projects
        set<long>::iterator projIter;   // Project iterator
        exec sql begin declare section;
                long            progNum;        // Programmer number
                char            progName[21];   // Programmer name
                long            projNum;        // Project number
                char            projName[21];   // Project name
                double          projValue;      // Project value
        exec sql end declare section;

        // Database connection formalities

        exec sql whenever sqlerror continue;
        exec sql connect to teamdb;
        if (SQLCODE != 0) goto DbError;
        exec sql drop table programmers;
        exec sql drop table projects;
        exec sql drop table assignments;
        exec sql whenever sqlerror goto DbError;
        exec sql commit work;

        // Create the database tables

        exec sql create table programmers (
                progNum         integer not null,
                progName        char(20) not null
        );
        exec sql create table projects (
                projNum         integer not null,
                projName        char(20) not null,
                projValue       double precision not null
        );
        exec sql create table assignments (
                asnProgNum      integer not null,
                asnProjNum      integer not null
        );

        // Generate the programmers

        for (i = 0; i < N_PROGS; i++) {

                // Insert the programmer row

                progNum = FIRST_PROG_NUM + i;
                RandName (progName);
                exec sql insert into programmers (
                        progNum, progName
                ) values (
                        :progNum, :progName
                );

                // Generate the project assignments

                projCnt = rand() % (MAX_PROJS_PER_PROG - MIN_PROJS_PER_PROG + 1)
                        + MIN_PROJS_PER_PROG;
                projSet.clear();
                while (projSet.size() < projCnt) {
                        projNum = rand() % N_PROJS + FIRST_PROJ_NUM;
                        if (projSet.count(projNum) == 0) {
                                projSet.insert (projNum);
```

```
                                if (activeProj.count(projNum) == 0)
                                        activeProj.insert(projNum);
                                exec sql insert into assignments (
                                        asnProgNum, asnProjNum
                                ) values (
                                        :progNum, :projNum
                                );
                        }
                }
        }

        // Emit the project rows

        for (
                projIter = activeProj.begin();
                projIter != activeProj.end();
                projIter ++
        ) {
                projNum = *projIter;
                RandName (projName);
                projValue = MAX_PROJ_VAL * static_cast<double>(rand())
                        / static_cast<double>(RAND_MAX);
                exec sql insert into projects (
                        projNum, projName, projValue
                ) values (
                        :projNum, :projName, :projValue
                );
        }

        // Commit the database changes

        exec sql commit work;
        return 0;

        // Process a database error

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