```
// OrderGen.cpp - GENERATE THE ORDER REPORT
//
// MODULE INDEX
// NAME                 CONTENTS
// main                 Main line
//
// MAINTENANCE HISTORY
// DATE         PROGRAMMER AND DETAILS
// 03-10-17     SHT     Original
// 20-10-17     YGL     Renamed database
// 27-10-17     SHT     Corrected arithmetic exception when a restaurant did
//                      accrpt any order
//
//--------------------------------------------------------------------------

#include <cstring>
#include <iomanip>
#include <iostream>
#include <map>
#include <queue>
#include <vector>
using namespace std;
exec sql include sqlca;

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

// ORDER STRUCTURE

struct Order {
        long    orderNo;
        long    orderValue;
        bool    orderVerified;
};

// RESTAURANT STRUCTURE

struct Restaurant {
        long restId;
        char restName[21];
};

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

int
main()
{
        unsigned long   verCnt;                 // Verified order count
        long            totalVal;               // Total of unverified order
                                                // value
        long            grandTotalVerCnt;       // Grand total of verified
                                                // order count
        long            grandTotalOrder;        // Grand total of order count
        long            grandTotalVal;          // Grand total of unverified
                                                // order value
```

```
double          percent;                // Verified percentage
bool            firstOrder;             // First order flag

Order           ordRec;                 // Order record
Restaurant      restRec;                // Restaurant record

queue<int>      acpResQue;              // Restaurant id queue

vector<Order>   *ordVec;                // Pointer to order vector
vector<Order>::iterator ordIt;          // Order iterator

vector<Restaurant> resVec;              // Restaurant vector
vector<Restaurant>::iterator resIt;     // Restaurant iterator

map<int, vector<Order> > orderMap;      // Order map

exec sql begin declare section;
        long    restId;                 // Restaurant id
        char    restName[21];           // Restaurant name
        long    orderNo;                // Order no
        long    orderValue;             // Order value
        double  orderRspTim;            // Order response time
        long    orderAccept;            // Order accept status
        long    orderVerified;          // Order verified status
exec sql end declare section;

// Connect to database

exec sql connect to egpc;

// Jump to DbError whenever an SQL error occurs

exec sql whenever sqlerror goto DbError;

// Load restaurant records from database (sorted)

exec sql declare resCur cursor for
        select  resRestId, resRestName
        from    res
        order   by resRestName;
exec sql open resCur;
for (;;) {
        exec sql fetch  resCur
                into    :restId, :restName;
        if (SQLCODE != 0) break;

        restRec.restId = restId;
        strcpy(restRec.restName, restName);
        resVec.push_back(restRec);
}
exec sql close resCur;

// Load order records from database (sorted)
```

```
exec sql declare ordCur cursor for
        select  ordOrderNo, ordValue, ordVerified
        from    ord
        order   by ordOrderNo;
exec sql open ordCur;
for (;;) {
        exec sql fetch  ordCur
                into    :orderNo, :orderValue, :orderVerified;
        if (SQLCODE != 0) break;

        // Determine order belong to which restaurant

        acpResQue = queue<int>();

        exec sql declare ordRspCur cursor for
                select  ordRspRestId, ordRspTim, ordRspAccept
                from    ordRsp
                where   ordRspOrderNo = :orderNo
                order   by ordRspTim;
        exec sql open ordRspCur;
        for (;;) {
                exec sql fetch  ordRspCur
                        into    :restId, :orderRspTim, :orderAccept;
                if (SQLCODE != 0) break;

                if (!orderAccept) {
                        if (acpResQue.front() != restId) {
                                cerr << "Bad data" << endl;
                                return 1;
                        }

                        acpResQue.pop();
                }
                else
                        acpResQue.push(restId);
        }
        exec sql close ordRspCur;

        restId = acpResQue.front();

        ordRec.orderNo = orderNo;
        ordRec.orderValue = orderValue;
        ordRec.orderVerified = orderVerified;

        orderMap[restId].push_back(ordRec);
}
exec sql close ordCur;

// Initialize grand total variable

grandTotalOrder = 0;
grandTotalVal = 0;
grandTotalVerCnt = 0;
```

```
// Set up cout format

cout.setf(ios::fixed);

// Output report title

cout << "======== UNVERIFIED ORDERS REPORT ========" << endl
        << endl;

// Output report header

cout << left
        << setw(21) << "RESTAURANT "
        << setw(9) << "ORDER ID "
        << right
        << setw(11) << "VALUE "
        << left
        << setw(9) << "VERIFIED "
        << right
        << setw(6) << "(%)"
        << left
        << endl;

cout << setfill('-')
        << right
        << setw(21) << ' '
        << setw(9) << ' '
        << setw(11) << ' '
        << setw(9) << ' '
        << setw(6) << '-'
        << setfill(' ')
        << left
        << endl;

for (resIt = resVec.begin(); resIt != resVec.end(); resIt++) {
        firstOrder = 1;
        verCnt = 0;
        totalVal = 0;

        cout << setw(21) << resIt->restName;

        ordVec = &orderMap[resIt->restId];

        for (
                ordIt = ordVec->begin();
                ordIt != ordVec->end();
                ordIt++
        ) {
                grandTotalOrder++;

                // Skip verified order and increase verified
                // order count

                if (ordIt->orderVerified) {
```

```
                verCnt++;
                grandTotalVerCnt++;
                continue;
        }

        totalVal += ordIt->orderValue;
        grandTotalVal += ordIt->orderValue;

        if (!firstOrder)
                cout << setw(21) << ' ';
        firstOrder = 0;

        cout << right
                << setw(8) << ordIt->orderNo << ' '
                << setprecision(2)
                << setw(10) << (ordIt->orderValue / 100.0f)
                << setprecision(0)
                << left
                << endl;
}

if (ordVec->size() > 0)
        percent = (verCnt / (double)ordVec->size()) * 100;
else
        percent = 0;

// Output unverified total

if (verCnt < ordVec->size()) {
        cout << right
                << setw(40) << "----------"
                << left
                << endl;
        cout << "TOTAL"
                << right
                << setprecision(2) << setw(35)
                << (totalVal / 100.0f)
                << setprecision(0) << setw(9) << verCnt
                << setprecision(2) << setw(7) << percent
                << setprecision(0)
                << left
                << endl;
        cout << right
                << setw(40) << "----------"
                << left
                << endl;
} else {
        cout << right
                << setw(28) << verCnt
                << setprecision(2) << setw(7) << percent
                << setprecision(0)
                << left
                << endl;
}
```

```
            cout << endl;
        }

        percent = (grandTotalVerCnt / (double)grandTotalOrder) * 100.0f;

        cout << setfill('-')
             << right
             << setw(21) << ' '
             << setw(9) << ' '
             << setw(11) << ' '
             << setw(9) << ' '
             << setw(6) << '-'
             << setfill(' ')
             << left
             << endl;

        cout << "GRAND TOTAL"
             << right
             << setprecision(2) << setw(29) << (grandTotalVal / 100.0f)
             << setprecision(0) << setw(9) << grandTotalVerCnt
             << setprecision(2) << setw(7) << percent
             << setprecision(0)
             << left
             << endl;

        cout << setfill('-')
             << right
             << setw(21) << ' '
             << setw(9) << ' '
             << setw(11) << ' '
             << setw(9) << ' '
             << setw(6) << '-'
             << setfill(' ')
             << left
             << endl;

        return 0;

DbError:
        cerr << "SQLCODE=" << SQLCODE << endl;
        return 1;
}
```