```
// OrderStatus.cpp - GENERATE THE ORDER STATUS REPORT
//
// MODULE INDEX
// NAME                    CONTENTS
// FormatNum         Format a number
// main                    Main line
//
// MAINTENANCE HISTORY
// DATE              PROGRAMMER AND DETAILS
// 24-09-14   MPF    Original
// 10-10-14   MPF    Sort committed stocks and backorders by customer first
// 10-10-14   MPF    Initialise flavour vector
//
//-----------------------------------------------------------------------------

#include <cstdlib>          // C-style standard library
#include <cstring>          // C-style string manipulation functions
#include <cctype>           // C-style character typing functions
#include <iostream>         // C++ input/output streams
#include <iomanip>          // C++ input/output manipulators
#include <sstream>          // C++ string stream declarations
#include <set>                    // C++ set declarations
#include <map>                    // C++ map declarations
#include <vector>           // C++ vector declarations
using namespace std;        // Expand the standard namespace
#include <unistd.h>         // Unix standard functions
exec sql include sqlca;           // Include SQL communications area

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

// ORDER ITEM STRUCTURE

struct OrderItem {
       long          requiredQty;  // Required quantity
       long          remainQty;    // Remaining quantity
};

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

// ORDER STRUCTURE

struct Order {
       long          orderNo;      // Order number
       long          custNo;             // Customer number
       bool          fulfilled;    // Fulfilled flag
       map<int, OrderItem> flavours;      // Flavors map
};

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

// STOCK BATCH STRUCTURE

struct Batch {
       long          batchNo;      // Batch number
       long          batchQty;     // Quantity
};

//-----------------------------------------------------------------------------
```

```
// PRINT PERCENTAGE

void
PrintPerc (
	int		pct)		// Percentage value
{
	cout << right << setw(3) << pct << left << "% ";
}

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

// PRINT QUANTITY

void
PrintQty (
	long		qty)		// Quantity value
{
	int		w;		// Width

	w = 1;
	if (qty >= 1000000) {
		cout << right << setw(w) << qty / 1000000 << ','
			<< setfill('0');
		w = 3;
	} else {
		w += 4;
	}
	if (qty >= 1000) {
		cout << right << setw(w) << qty / 1000 % 1000 << ','
			<< setfill('0');
		w = 3;
	} else {
		w += 4;
	}

	cout << right << setw(w) << qty % 1000 << setfill(' ') << left << ' ';
}

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

// MAIN LINE

int
main (
	int		argc,		// Argument count
	char		*argv[])	// Argument value pointers
{
	map<int, vector<Batch> > stockMap; // Flavour stock map
	map<int, vector<Batch> >::iterator stockIt; // Flavour stock iterator
	vector<Batch> *batchVec;	// Pointer to batch vector
	vector<Batch>::iterator batchIt; // Batch iterator
	Batch		batchRec;	// Stock batch record

	map<int, vector<Order> > custMap; // Customer map
	map<int, vector<Order> >::iterator custIt; // Customer iterator
	vector<Order>::iterator orderIt; // Order iterator
	Order		orderRec;	// Order record
```

```
map<int,OrderItem>::iterator itemIt; // Order item iterator
OrderItem     itemRec;      // Order item record

map<long, long>      shortage;     // Flavour shortage map
map<long, long>::iterator shortageIt; // Flavour shortage iterator

bool          firstOrder;   // First order flag
bool          firstFlavour; // First flavour flag
bool          firstLine;    // First line flag
bool          reqTotal;     // Require total flag
int           remain;              // Remaining quantity
int           required;     // Required quantity
int           pct;          // Percentage value
long          total;        // Total value
long          grandTotal;   // Grand total value

exec sql begin declare section;
       long          batchNo;      // Batch number
       long          flavourNo;    // Flavour number
       long          qty;          // Quantity
       long          custNo;              // Customer number
       long          orderNo;      // Order number
exec sql end declare section;

// Connect to the database

exec sql connect to orderdb;

// Jump to DbError whenever an SQL error occurs

exec sql whenever sqlerror goto DbError;

// Load the stock records

exec sql declare stockCur cursor for
       select stockBatchNo, stockFlavourNo, stockQty
       from   stock
       where  stockQty > 0
       order  by stockBatchNo;
exec sql open stockCur;
for (;;) {
       exec sql fetch       stockCur
              into   :batchNo, :flavourNo, :qty;
       if (SQLCODE != 0) break;

       batchRec.batchNo = batchNo;
       batchRec.batchQty = qty;
       stockMap[flavourNo].push_back(batchRec);
}
exec sql close stockCur;

// Load the customer orders

exec sql declare orderCur cursor for
       select orderNo, orderCustNo
       from   custOrder
       where  delivered = 0
       order  by orderNo;
```

```
exec sql open orderCur;
for (;;) {
	exec sql fetch	orderCur
		into	:orderNo, :custNo;
	if (SQLCODE != 0) break;

	orderRec.orderNo = orderNo;
	orderRec.custNo = custNo;
	orderRec.fulfilled = 1;
	orderRec.flavours.clear();

	// Load the items for the order

	exec sql declare itemCur cursor for
		select itemFlavourNo, itemQty
		from	orderItem
		where	itemOrderNo = :orderNo;
	exec sql open itemCur;
	for (;;) {
		exec sql fetch	itemCur
			into	:flavourNo, :qty;
		if (SQLCODE != 0) break;

		itemRec.requiredQty = qty;
		itemRec.remainQty = qty;

		batchVec = &stockMap[flavourNo];
		batchIt = batchVec->begin();
		while (
			batchIt != batchVec->end() &&
			itemRec.remainQty > 0
		) {
			if (batchIt->batchQty > itemRec.remainQty) {
				batchIt->batchQty -= itemRec.remainQty;
				itemRec.remainQty = 0;
			} else {
				itemRec.remainQty -= batchIt->batchQty;
				batchIt->batchQty = 0;
			}
			batchIt ++;
		}

		if (itemRec.remainQty > 0) {
			orderRec.fulfilled = 0;
			shortage[flavourNo] += itemRec.remainQty;
		}
		orderRec.flavours[flavourNo] = itemRec;
	}
	exec sql close itemCur;

	custMap[custNo].push_back(orderRec);
}
exec sql close orderCur;

cout << "ORDER STATUS REPORT" << endl << endl;

// Display committed stock
```

```
cout << "COMMITTED STOCK" << endl
       << setfill('-') << setw(80) << '-' << endl << setfill(' ')
       << left
       << setw(10) << "CUSTOMER "
       << setw(10) << "ORDER "
       << setw(10) << "FLAVOUR "
       << right << setw(10) << "QUANTITY " << left
       << endl;
for (custIt = custMap.begin(); custIt != custMap.end(); custIt++) {
       firstOrder = 1;

       for (
              orderIt = custIt->second.begin();
              orderIt != custIt->second.end();
              orderIt++
       ) {
              if (orderIt->fulfilled) {
                     if (firstOrder)
                            cout << setw(10) << orderIt->custNo;
                     else
                            cout << setw(10) << ' ';
                     firstOrder = 0;
                     cout << setw(10) << orderIt->orderNo;
                     firstFlavour = 1;
                     for (
                            itemIt = orderIt->flavours.begin();
                            itemIt != orderIt->flavours.end();
                            itemIt ++
                     ) {
                            if (! firstFlavour)
                                   cout << setw(20) << ' ';
                            firstFlavour = 0;
                            cout << left << setw(10)
                                   << itemIt->first
                                   << right << setw(10);
                            PrintQty (itemIt->second.requiredQty);
                            cout << endl;
                     }
                     if (firstFlavour)
                            cout << endl;
              }
       }
}

// Display backorders

cout << endl << endl
       << "BACKORDERS" << endl
       << setfill('-') << setw(80) << '-' << endl << setfill(' ')
       << setw(10) << "CUSTOMER "
       << setw(10) << "ORDER "
       << setw(10) << "FLAVOUR "
       << right << setw(10) << "ORDER " << left
       << setw(15) << "   AVAILABLE"
       << setw(15) << "    SHORTAGE"
       << endl

       << setw(30) << ' ' << right
```

```
        << setw(10) << "QUANTITY "
        << setw(10) << "QUANTITY "
        << "PERC "
        << setw(10) << "QUANTITY "
        << "PERC " << left
        << endl;

for (custIt = custMap.begin(); custIt != custMap.end(); custIt++) {
        firstOrder = 1;

        for (
                orderIt = custIt->second.begin();
                orderIt != custIt->second.end();
                orderIt++
        ) {
                if (!orderIt->fulfilled) {
                        if (firstOrder)
                                cout << setw(10) << orderIt->custNo;
                        else
                                cout << setw(10) << ' ';
                        firstOrder = 0;
                        cout << setw(10) << orderIt->orderNo;

                        firstFlavour = 1;
                        for (
                                itemIt = orderIt->flavours.begin();
                                itemIt != orderIt->flavours.end();
                                itemIt ++
                        ) {
                                remain = itemIt->second.remainQty;
                                required = itemIt->second.requiredQty;
                                if (remain > 0) {
                                        if (! firstFlavour)
                                                cout << setw(20) << ' ';
                                        cout << left << setw(10)
                                                << itemIt->first
                                                << right << setw(10);
                                        PrintQty (required);

                                        pct = (int)floor(100 * remain /
                                                required + 0.5);
                                        PrintQty (required - remain);
                                        PrintPerc (100 - pct);
                                        PrintQty (remain);
                                        PrintPerc (pct);
                                        cout << endl;
                                        firstFlavour = 0;
                                }
                        }
                        if (firstFlavour)
                                cout << endl;
                }
        }
}

// Display shortages

cout << endl
```

```
        << "SHORTAGES" << endl
        << setfill('-') << setw(80) << '-' << endl << setfill(' ')
        << setw(10) << "FLAVOUR "
        << right << setw(10) << "QUANTITY " << left
        << endl;

total = 0;
for (
        shortageIt = shortage.begin();
        shortageIt != shortage.end();
        shortageIt ++
) {
        cout << setw(10) << shortageIt->first;
        PrintQty (shortageIt->second);
        cout << endl;
        total += shortageIt->second;
}
cout << right << setfill('-') << setw(20) << ' '
        << left << setfill(' ') << endl
        << setw(10) << "TOTAL";
PrintQty (total);

// Display surplus stock

cout << endl << endl
        << "SURPLUS STOCK" << endl
        << setfill('-') << setw(80) << '-' << endl << setfill(' ')
        << setw(10) << "FLAVOUR "
        << setw(10) << "BATCH "
        << right << setw(10) << "QUANTITY " << left
        << endl;

grandTotal = 0;
for (
        stockIt = stockMap.begin();
        stockIt != stockMap.end();
        stockIt ++
) {
        total = 0;
        for (
                batchIt = stockIt->second.begin();
                batchIt != stockIt->second.end();
                batchIt ++
        ) {
                total += batchIt->batchQty;
        }

        if (total > 0) {
                cout << setw(10) << stockIt->first;

                firstLine = 1;
                reqTotal = 0;

                for (
                        batchIt = stockIt->second.begin();
                        batchIt != stockIt->second.end();
                        batchIt ++
                ) {
```

```
                        if (batchIt->batchQty > 0) {
                                if (!firstLine) {
                                        cout << setw(10) << ' ';
                                        reqTotal = 1;
                                }
                                cout << setw(10) << batchIt->batchNo;
                                PrintQty (batchIt->batchQty);
                                cout << endl;
                                firstLine = 0;
                        }
                }

                if (reqTotal) {
                        cout << setw(10) << ' ' << right
                                << setfill('-') << setw(20) << ' '
                                << left << endl << setfill(' ')
                                << setw(10) << ' '
                                << setw(10) << "TOTAL";
                        PrintQty (total);
                        cout << endl << endl;
                }
                grandTotal += total;
        }
    }
    cout << right << setfill('-') << setw(30) << ' ' << left << endl
            << setfill(' ') << setw(20) << "GRAND TOTAL";
    PrintQty (grandTotal);
    cout << endl;

    return 0;

    // Process database errors

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