```
// Microbank.cpp - MICROBANK LOAN STATUS ANALYSIS
//
// MODULE INDEX
// NAME                 CONTENTS
// ExecCmp              Executive comparison function for qsort
// ErrFun               Database error handling function
// PrsApp               Process loan approvers
// Title                Print page title
// OutRep               Output report
// Main                 Main line
//
// MAINTENANCE HISTORY
// DATE         PROGRAMMER AND DETAILS
// 08-10-03     PFT     Original
//
//---------------------------------------------------------------------------

#include <stdlib.h>             // Standard library
#include <memory.h>             // Memory manipulation functions
#include <stdio.h>              // Standard input/output declarations
#include <unistd.h>             // Unix standard library
#include <string.h>             // String manipulation functions
#include <time.h>               // Time function declarations

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

// GLOBAL DEFINITIONS

$define NAM_LEN         20;             // Executive name length
#define LNS_P_PG        56              // Lines per page
#define DB_NAME         "Microbank"     // Database name

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

// LOAN DATA STRUCTURE

typedef struct {
        int     Loan;           // Number of loans
        double  LoanVal;        // Value of loans
        int     NpLoan;         // Number of non-performing loans
        double  NpLoanVal;      // Value of non-performing loans
} Loan_t;

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

// EXECUTIVE'S LOAN APPROVAL PROFILE STRUCTURE

typedef struct {
        int     execCode;               // Executive code
        char    execName[NAM_LEN+1];    // Executive name
        Loan_t  execLoan0;              // Executive's loan data
} Exec_t;
#define execLoan        execLoan0.Loan
#define execLoanVal     execLoan0.LoanVal
#define execNpLoan      execLoan0.NpLoan
#define execNpLoanVal   execLoan0.NpLoanVal

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

// GLOBAL DATA

Exec_t          *execArr;               // Executive loan approval profile
Loan_t          totLoan;                // Loan totals
```

```
int             lnsRem;                 // Lines remaining
char            repTdt[30];             // Report time and date
int             pageNo;                 // Page number

exec sql begin declare section;
        int             execCnt;        // Executive count
exec sql end declare section;

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

// EXECUTIVE COMPARISON FUNCTION FOR QSORT

int
ExecCmp (
        Exec_t          *exec1,         // First record
        Exec_t          *exec2)         // Second record
{
        if (exec1->execCode > exec2->execCode) return 1;
        if (exec1->execCode < exec2->execCode) return -1;
        return 0;
}

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

// DATABASE ERROR HANDLING FUNCTION

void
ErrFun ()
{
        printf ("DB error SQLCODE = %d\t", SQLCODE);
        printf ("errd[1] = %d\t", sqlca.sqlerrd[1]);
        printf ("errm = %s\n", sqlca.sqlerrm);
        exit (1);
}

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

// PROCESS LOAN APPROVERS

void
PrsApp (
        int     xLoanId,                // Loan identifier
        double  amt,                    // Amount (in cent)
        short   status)                 // Status (1=good, 0=bad)
{
        Exec_t  *appPtr;                // Approver's record pointer

        exec sql begin declare section;
                int     loanId;         // Loan identifier
                int     execCode;       // Executive code
                short   appType;        // Approval type
        exec sql end declare section;

        // Initialize SQL error handling

        exec sql whenever sqlerror goto :db_error;

        // Load SQL variables

        loanId = xLoanId;

        // Process approvers
```

```
        exec sql declare appCur cursor for
                select  appExecCode, appType
                from    approvals
                where   appLoanId = :loanId;
        exec sql open appCur;
        for (;;) {
                exec sql fetch appCur
                        into    :execCode, :appType;
                if (SQLCODE != 0) break;

                // Validate approval type

                if (appType != 0 && appType != 1) {
                        printf ("Microbank: invalid approval type\n");
                        exit (1);
                }

                // Search approver's record pointer

                appPtr = (Exec_t*) bsearch (
                        &execCode, execArr, execCnt, sizeof(Exec_t),
                        (int(*)(const void*, const void*))ExecCmp
                );
                if (appPtr == NULL) {
                        printf ("Microbank: approver not found\n");
                        exit (1);
                }

                // Post loan data to approver

                appPtr->execLoan ++;
                appPtr->execLoanVal += amt;
                if (status == 0) {
                        appPtr->execNpLoan ++;
                        appPtr->execNpLoanVal += amt;
                }
        }
        exec sql close appCur;
        return;

db_error:
        ErrFun ();
}

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

// PRINT PAGE TITLE

void
Title ()
{
        char    buf[100], *p, *q;       // Formatting variables

        if (pageNo != 1) printf ("\f");

        lnsRem = LNS_P_PG;
        p = buf;
        q = repTdt;
        while (*q != '\n') *p++ = *q++;
        *p = '\0';
        printf (
                "%-28s %-40s %-4s %3d\n\n",
                buf, "MICROBANK LOAN ANALYSIS", "PAGE", pageNo++
```

```
        );
        printf (
                "%-6s  %-20s %7s %16s %7s %16s\n",
                "EXEC-", "", "LOAN", "LOAN", "NLOAN", "NLOAN"
        );
        printf (
                "%-6s  %-20s %7s %16s %7s %16s\n\n",
                "CODE", "EXECUTIVE NAME", "COUNT", "VALUE", "COUNT", "VALUE"
        );

        lnsRem -= 5;
}

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

// OUTPUT REPORT

void
OutRep ()
{
        int     i;              // General purpose index

        // Initialize page number

        pageNo = 1;

        // Print report title

        Title ();

        // Display loan approval profile of each executive

        for (i=0; i<execCnt; i++) {
                if (lnsRem < 1) Title ();
                printf (
                        "%6d  %20s %7d %16.2lf %7d %16.2lf\n",
                        execArr[i].execCode, execArr[i].execName,
                        execArr[i].execLoan, execArr[i].execLoanVal / 100.0,
                        execArr[i].execNpLoan,
                        execArr[i].execNpLoanVal / 100.0
                );
                lnsRem --;
        }

        // Print loan totals

        if (lnsRem < 2)
                Title ();
        else
                printf ("\n");
        printf (
                "%-28s %7d %16.2lf %7d %16.2lf\n",
                "TOTAL",
                totLoan.Loan,
                totLoan.LoanVal/100.0,
                totLoan.NpLoan,
                totLoan.NpLoanVal/100.0
        );
}

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

// MAIN LINE
```

```
int
main ()
{
        time_t          ctime;          // Calendar time
        struct  tm      *ltime;         // Local time
        int             i;              // General purpose index

        exec sql begin declare section;
                char    dbNam[10];              // Database name
                int     execCode;               // Executive code
                char    execName[NAM_LEN+1];    // Executive name
                int     loanId;                 // Loan identifier
                double  amt;                    // Loan amount (in cent)
                short   status;                 // Loan status (1=good, 0=bad)
        exec sql end declare section;

        // Initiate SQL error handling

        exec sql whenever sqlerror goto :db_error;

        // Open the database

        strcpy (dbNam, DB_NAME);
        exec sql database :dbNam;

        // Initialization

        memset (&totLoan, 0, sizeof(totLoan));

        // Set the report time and date

        ctime = time ((time_t*)0);
        ltime = localtime (&ctime);
        strcpy (repTdt, asctime(ltime));

        // Count number of executives

        exec sql select count(*)
                into    :execCnt
                from    executives;

        // Allocate memory for executive array

        execArr = new Exec_t[execCnt];
        memset (execArr, 0, execCnt * sizeof(Exec_t));

        // Load executive array with basic data

        exec sql declare execCur cursor for
                select  execCode, execName
                from    executives;
        exec sql open execCur;
        i = 0;
        for (;;) {
                exec sql fetch execCur
                        into    :execCode, :execName;
                if (SQLCODE != 0) break;
                execArr[i].execCode = execCode;
                strcpy (execArr[i].execName, execName);
                i++;
        }
        exec sql close execCur;
```

```
        // Sort the executive array

        qsort ((char*)execArr, execCnt, sizeof(Exec_t),
                (int(*)(const void*, const void*))ExecCmp);

        // Process each loan

        exec sql declare loansCur cursor for
                select  loanId, loanAmount, loanStatus
                from    loans;
        exec sql open loansCur;
        for (;;) {
                exec sql fetch loansCur
                        into    :loanId, :amt, :status;
                if (SQLCODE != 0) break;

                // Validate loan status

                if (status != 0 && status != 1) {
                        printf ("Microbank: invalid loan status\n");
                        exit (1);
                }

                // Update loan totals

                totLoan.Loan ++;
                totLoan.LoanVal += amt;
                if (status == 0) {
                        totLoan.NpLoan ++;
                        totLoan.NpLoanVal += amt;
                }

                // Process associated approvers

                PrsApp (loanId, amt, status);
        }
        exec sql close loansCur;

        // Output executive loan approval data

        OutRep ();

        // Exit gracefully

        return 0;

db_error:
        ErrFun ();
}
```