```c
#include "gui.h"
#include <libxml/encoding.h>
#include <libxml/xmlwriter.h>

#if defined(LIBXML_WRITER_ENABLED) && defined(LIBXML_OUTPUT_ENABLED)

#define MY_ENCODING "UTF-8"

double ttl = 0.00;
int i = 0;
int cnt = 1;

// Define structure for Journal Entries.
typedef struct JE
{
        char    entry_date[11];
        char    dr_acct_number[6];
        int     dr_acct;                        // numerical debit entry account number
        int     cr_acct;                        // numerical credit entry account number
        char    cr_acct_number[6];
        char    entry_memo[26];
        double  amount;                         // entry amount
        double  dr_ttl;                         // total debits
        double  cr_ttl;                         // total credits
        struct  JE *next;                       // pointer to the next JE
        struct  JE *prev;                       // pointer to the previous JE
} JournalEntry;

void readEntry (xmlDocPtr doc, xmlNodePtr entry, JournalEntry **first)
{
        JournalEntry *head;
        head = *first;
        xmlChar *key2;
        int i =0;

        while (head->next != NULL)
        {
                head = head->next;
                i++;
        }

        head->next = (JournalEntry*)malloc(sizeof(JournalEntry));
        head = head->next;
        head->next = NULL;

        entry = (*entry).xmlChildrenNode;
        while (entry != NULL) {
                // Parse Date
                if((!xmlStrcmp((*entry).name, (const xmlChar *)"DATE"))) {
                        key2 = xmlNodeListGetString(doc, entry->xmlChildrenNode, 1);
                        // Store journal entry date in struct.
                        strcpy(head->entry_date, key2);
                        // Prints name of XML element.
                        ////printf ("%s \t", key2);
                        xmlFree(key2);
                }
                // Parse Debit Account
                if((!xmlStrcmp((*entry).name, (const xmlChar *)"DR_ACCT"))) {
                        xmlChar *dr_acct;
                        dr_acct = xmlNodeListGetString(doc, (*entry).xmlChildrenNode, 1);
                        // Store account number in struct.
                        strcpy(head->dr_acct_number, dr_acct);
                        head->dr_acct = atof(dr_acct);
                        // Prints value of XML element.
                        ////printf ("DR%s | ", dr_acct);
                        xmlFree(dr_acct);
                }
                // Parse Credit Account
                if((!xmlStrcmp((*entry).name, (const xmlChar *)"CR_ACCT"))) {
                        xmlChar *cr_acct;
                        cr_acct = xmlNodeListGetString(doc, (*entry).xmlChildrenNode, 1);
                        strcpy(head->cr_acct_number, cr_acct);
                        head->cr_acct = atof(cr_acct);
                        ////printf ("CR%s\t", cr_acct);
                        xmlFree(cr_acct);
                }
                // Parse memo
                if((!xmlStrcmp((*entry).name, (const xmlChar *)"MEMO"))) {
                        xmlChar *entry_memo;
                        entry_memo = xmlNodeListGetString(doc, (*entry).xmlChildrenNode,
1);
                        strcpy(head->entry_memo, entry_memo);
                        ////printf ("Memo%s\t", entry_memo);
                        xmlFree(entry_memo);
                }
                // Parse Amount
                if((!xmlStrcmp((*entry).name, (const xmlChar *)"AMOUNT"))) {
                        xmlChar *amount;
                        amount = xmlNodeListGetString(doc, (*entry).xmlChildrenNode, 1);
                        head->amount = atof(amount);
                        ttl+=head->amount;
                        xmlFree(amount);
                }
                entry = (*entry).next;
        }

        ////printf("\nParsed into Linked List ...\tLast node is 0x%X\n", je);
        return;
}
void parseEntry(xmlDocPtr doc, xmlNodePtr entry2, JournalEntry *je)
{
        xmlChar *key2;

        while ((je->next != NULL))
                je = je->next;
        je->next = (JournalEntry*)malloc(sizeof(JournalEntry));
        je = je->next;
        je->next = NULL;

        ////printf ("%i) je: 0x%X\tje->prev: 0x%X\tptr->next: 0x%X\n", cnt, je, je->prev,
je->next);

        entry2 = (*entry2).xmlChildrenNode;
        while (entry2 != NULL) {
                // Parse Date
                if((!xmlStrcmp((*entry2).name, (const xmlChar *)"DATE"))) {
                        key2 = xmlNodeListGetString(doc, entry2->xmlChildrenNode, 1);
                        // Store journal entry date in struct.
                        strcpy((*je).entry_date, key2);
                        // Prints name of XML element.
                        ////printf ("%s \t", key2);
                        xmlFree(key2);
                }
                // Parse Debit Account
                if((!xmlStrcmp((*entry2).name, (const xmlChar *)"DR_ACCT"))) {
                        xmlChar *dr_acct;
                        dr_acct = xmlNodeListGetString(doc, (*entry2).xmlChildrenNode, 1);
                        // Store account number in struct.
                        strcpy((*je).dr_acct_number, dr_acct);
                        (*je).dr_acct = atof(dr_acct);
                        // Prints value of XML element.
                        ////printf ("DR%s | ", dr_acct);
                        xmlFree(dr_acct);
                }
                // Parse Credit Account
                if((!xmlStrcmp((*entry2).name, (const xmlChar *)"CR_ACCT"))) {
                        xmlChar *cr_acct;
                        cr_acct = xmlNodeListGetString(doc, (*entry2).xmlChildrenNode, 1);
                        strcpy((*je).cr_acct_number, cr_acct);
                        (*je).cr_acct = atof(cr_acct);
                        ////printf ("CR%s\t", cr_acct);
                        xmlFree(cr_acct);
                }
                // Parse memo
                if((!xmlStrcmp((*entry2).name, (const xmlChar *)"MEMO"))) {
                        xmlChar *entry_memo;
                        entry_memo = xmlNodeListGetString(doc, (*entry2).xmlChildrenNode,
1);
                        strcpy((*je).entry_memo, entry_memo);
                        ////printf ("Memo%s\t", entry_memo);
                        xmlFree(entry_memo);
                }
                // Parse Amount
                if((!xmlStrcmp((*entry2).name, (const xmlChar *)"AMOUNT"))) {
                        xmlChar *amount;
                        amount = xmlNodeListGetString(doc, (*entry2).xmlChildrenNode, 1);
                        (*je).amount = atof(amount);
                        ttl+=(*je).amount;
                        /*
                        if ((*je).amount<0)
                                ////printf ("%s%s$%s%s\t\t%10.2f\n", bold, red, amount,
ttl, none);
                        else
                                ////printf ("$%s\t\t%10.2f\n", amount, ttl);
                        */
                        xmlFree(amount);
                }
                entry2 = (*entry2).next;
        }
        ////printf("\nParsed into Linked List ...\tLast node is 0x%X\n", je);
        return;
}

// Proccess XML Jeneral Gournal datafile.
void readJournal (char *docname, JournalEntry **first)
{
        JournalEntry **head;
        head = first;

        xmlDocPtr doc;
        xmlNodePtr cur;

        doc = xmlParseFile (docname);

        // Check if document parsed successfully.
        if (doc == NULL) { printf ("Document not parsed successfully.\n"); return; }

        cur = xmlDocGetRootElement(doc);

        // Check if document contains XML data.
        if (cur == NULL) {
                printf ("empty document.\n");
                xmlFreeDoc(doc);
                return;
        }
        // Check if document is Jeneral Gournal XML file.
        if (xmlStrcmp(cur->name, (const xmlChar *) "GENERAL_JOURNAL")) {
                printf ("document of the wrong type, root node != entries.\n");
                return;
        }

        cur = cur -> xmlChildrenNode;

        while (cur != NULL) {
                if ((xmlStrcmp(cur->name, (const xmlChar *)"ENTRY"))) {
                        ////printf ("parsing.... %s \n\n", cur->name);
                        xmlChar *key;
                        xmlNodePtr entry;
                        entry = cur->xmlChildrenNode;

                        while (entry != NULL) {
                                if((!xmlStrcmp(entry->name, (const xmlChar *)"ENTRY"))) {
                                        key = xmlNodeListGetString(doc, entry-
>xmlChildrenNode, 1);
                                        ////printf ("Currently at: %s \n", entry->name);
                                        readEntry(doc, entry, first);
                                        ////printf ("saved to 0x%X\n\n", je);
                                        cnt++;
                                }
                                entry = entry->next;
                        }
                        // originally was here & compiled OK. However, started to crash
the program after adding linked lists.
                        //xmlFree(key);
                }
                else {
                        printf ("nothing.\n");
                }
                cur = cur -> next;
        }
        xmlFreeDoc(doc);

        ////printf ("Parsed account info ... \n");
        return;
}

void parseDoc(char *docname, JournalEntry *je)
{

        xmlDocPtr doc;
        xmlNodePtr cur;

        doc = xmlParseFile (docname);

        // Check if document parsed successfully.
        if (doc == NULL) { printf ("Document not parsed successfully.\n"); return; }

        cur = xmlDocGetRootElement(doc);

        // Check if document contains XML data.
        if (cur == NULL) {
                printf ("empty document.\n");
                xmlFreeDoc(doc);
                return;
        }
        // Check if document is Jeneral Gournal XML file.
        if (xmlStrcmp(cur->name, (const xmlChar *) "GENERAL_JOURNAL")) {
                printf ("document of the wrong type, root node != entries.\n");
                return;
        }

        cur = cur -> xmlChildrenNode;

        while (cur != NULL) {
                if ((xmlStrcmp(cur->name, (const xmlChar *)"ENTRY"))) {
                        ////printf ("parsing.... %s \n\n", cur->name);
                        xmlChar *key;
                        xmlNodePtr entry;
                        entry = cur->xmlChildrenNode;

                        while (entry != NULL) {
                                if((!xmlStrcmp(entry->name, (const xmlChar *)"ENTRY"))) {
                                        key = xmlNodeListGetString(doc, entry-
>xmlChildrenNode, 1);
                                        ////printf ("Currently at: %s \n", entry->name);
                                                parseEntry(doc, entry, je);
                                                ////printf ("saved to 0x%X\n\n", je);
                                                cnt++;
                                }
                                entry = entry->next;
                        }
                        // originally was here & compiled OK. However, started to crash
the program after adding linked lists.
                        //xmlFree(key);
                }
                else {
                        printf ("nothing.\n");
                }
                cur = cur -> next;
        }
        xmlFreeDoc(doc);

        ////printf ("Parsed account info ... \n");
        return;
}

void addEntry (JournalEntry *ptr, JournalEntry entry)
{
        int i = 1;
        ////printf ("addEntry... \n");
        while ((ptr->next != NULL)) {
                i++;
                ptr = ptr->next;
        }

        ptr->next = (JournalEntry*)malloc(sizeof(JournalEntry));
        ////printf ("%i) ptr: 0x%X\tptr->next: 0x%X\n", i, ptr, ptr->next);
        ptr = ptr->next;
        ////printf ("node #%i added at address 0x%X\n", i, ptr);
        strcpy(ptr->entry_date, entry.entry_date);
        strcpy(ptr->dr_acct_number, entry.dr_acct_number);
        strcpy(ptr->cr_acct_number, entry.cr_acct_number);
        strcpy(ptr->entry_memo, entry.entry_memo);
        ptr->dr_acct =  entry.dr_acct;
        ptr->cr_acct =  entry.cr_acct;
```

```
	ptr->amount = entry.amount;
	ptr->next = NULL;

	if (entry.amount<0.00)
		printf ("%s\t%s\t%s\t%s\t%s%s%.2f%s\n", entry.entry_date, 
entry.dr_acct_number, entry.cr_acct_number, entry.entry_memo, entry.amount, normal, red, 
none);
	else
		printf ("%s\t%s\t%s\t%s\t%.2f\n", entry.entry_date, entry.dr_acct_number, 
entry.cr_acct_number, entry.entry_memo, entry.amount);
	printf ("\t\tadded.\n");
	return;
}

JournalEntry* findAccount(JournalEntry *loc, int acct_num)
{
	JournalEntry *node=NULL;
	////printf ("0x%X\n", loc);

	do {
		////printf ("starting search for %i from 0x%X\n", acct_num, loc);
		if(loc->dr_acct == acct_num) {
			node = loc;
			break;
		}
		loc = loc->next;
	} while (loc!=NULL);
	return node;
}

void addChartAccount(JournalEntry *chart, int acct_num)
{
	while (chart->next != NULL)
		chart = chart->next;

	chart->next = (JournalEntry*)malloc(sizeof(JournalEntry));
	chart = chart->next;
	////printf ("adding account %i at node 0x%X\n", acct_num, chart);
	strcpy(chart->entry_date,"");
	strcpy(chart->dr_acct_number,"");
	strcpy(chart->cr_acct_number,"");
	strcpy(chart->entry_memo,"");
	chart->amount =0;
	chart->dr_acct = acct_num;
	chart->cr_acct = acct_num;
	chart->next = NULL;
	chart->prev = NULL;

	return;
}

// Creates chart of accouts from XML file.
void populateAccounts (JournalEntry *start, JournalEntry *je, JournalEntry *chart)
{
	char acct_num[6], tmp[6];
	JournalEntry *ptr,*search, *chart_start;
	ptr = je;
	chart_start = chart;

	int cnt = 0;
	////printf ("\n%s%sPupulating list of accounts ...%s\n", normal, red, none);
	////printf ("node: 0x%X\tnext: 0x%X\n", chart, chart->next);
	do {
		cnt++;
		// Scan through DEBIT accounts.
		strcpy(acct_num, je->dr_acct_number);
		//printf ("%3i) %s\n", cnt, acct_num);
		search = NULL;
		search = findAccount(chart, atof(acct_num));
		if(search!=NULL)
			printf ("");
		//	printf ("%s%saccount %3s is located at node 0x%X%s\n", bold, 
green, acct_num, search, none);
		else {
		//	printf ("%s%s%3s not found.%s\n", bold, red, acct_num, none);
			addChartAccount(chart_start, atof(acct_num));
		}
		je = je->next;
	} while (je->next!=NULL);

	je = ptr;

	do {
		// Scan through CREDIT accounts.
		strcpy(acct_num, je->cr_acct_number);
		//printf ("%3i) %s\n", cnt, acct_num);
		search = NULL;
		search = findAccount(chart, atof(acct_num));
		if(search!=NULL)
			printf ("");
			//printf ("%s%saccount %3s is located at node 0x%X%s\n", bold, 
green, acct_num, search, none);
		else {
			//printf ("%s%s%3s\t0x%X not found.%s\n", bold, red, acct_num, 
je, none);
			addChartAccount(chart_start, atof(acct_num));
		}
		je = je->next;
	} while (je->next!=NULL);

	return;
}

void getTrialBalance (JournalEntry *journal, JournalEntry *chart)
{
	JournalEntry *tmp;
	tmp = journal;
	do {
		do {
			if (chart->dr_acct == journal->dr_acct)
				chart->dr_ttl+=journal->amount;
			if (chart->cr_acct == journal->cr_acct)
				chart->cr_ttl+=journal->amount;
			journal = journal->next;
		} while (journal != NULL);

		journal = tmp;
		chart = chart->next;
	} while (chart != NULL);
	return;
}

void printEntries (JournalEntry *ptr)
{
	if (ptr == NULL)
```

```
                return;
        i++;
        printf ("%s%s%3i%s ", bold, white, i, none);
        if (ptr->amount<0.00)
                printf ("%s%s0x%X%s\t%s\t%s\t%s\t%s\t%s%s%10.2f\t%i\t%s\t%10.2f\t%10.2f
\n", normal, green, ptr, none, ptr->entry_date, ptr->dr_acct_number, ptr->cr_acct_number,
ptr->entry_memo, bold, red, ptr->amount, ptr->dr_acct, ptr->dr_ttl, ptr->cr_ttl, none);
        else
                printf ("%s%s0x%X%s\t%s\t%s\t%s\t%s\t%10.2f\t%i\t\t%10.2f\t%10.2f\n",
normal, green, ptr, none, ptr->entry_date, ptr->dr_acct_number, ptr->cr_acct_number, ptr-
>entry_memo, ptr->amount, ptr->dr_acct, ptr->dr_ttl, ptr->cr_ttl);
        printEntries(ptr->next);
        return;
}

void printJournal (JournalEntry **head)
{
        if (head == NULL)
                return;
        i++;
        printf ("%s%s%3i%s ", bold, white, i, none);
        if ((*head)->amount<0.00)
                printf ("%s%s0x%X%s\t%s\t%s\t%s\t%s\t%s%s%10.2f\t%i\t%s\t\t%10.2f\t%10.2f
\n", normal, green, (*head), none, (*head)->entry_date, (*head)->dr_acct_number, (*head)-
>cr_acct_number, (*head)->entry_memo, bold, red, (*head)->amount, (*head)->dr_acct,
(*head)->dr_ttl, (*head)->cr_ttl, none);
        else
                printf ("%s%s0x%X%s\t%s\t%s\t%s\t%s\t%10.2f\t%i\t%10.2f\t%10.2f\n",
normal, green, (*head), none, (*head)->entry_date, (*head)->dr_acct_number, (*head)-
>cr_acct_number, (*head)->entry_memo, (*head)->amount, (*head)->dr_acct, (*head)->dr_ttl,
(*head)->cr_ttl);
        printEntries((*head)->next);
        return;
}

#else
int main(void) {
    fprintf(stderr, "Writer or output support not compiled in\n");
    exit(1);
}
#endif
```