#define MAX\_ITEM 500

#define DATAFILE "144\_fp\_items.txt"

const double TAX = 0.13;

const int SIZE = 20;

struct Item {

double price;

int sku;

int isTaxed;

char name[21]; };

void flushKeyboard(void) {

while (getchar() != '\n') ; }

void pause(void) {

printf("Press ent to cont...");

flushKeyboard(); }

double getDouble(void) {

double value;

double errorCheck;

char checkNewLine;

int done = 1;

while (done == 1)

errorCheck = scanf("%lf%c", &value, &checkNewLine);

if (checkNewLine != '\n' || errorCheck == 0) {

printf("Invalid number, please try again: ");

flushKeyboard(); }

else { done = 0;}

return value;

double getDouble Limited(double lowerLimit, double upperLimit){

double limitCheck;

int done = 1;

while (done == 1) {

limitCheck = getDouble();

if (limitCheck > upperLimit || limitCheck < lowerLimit) {

printf("Invalid value, %lf< value < %lf: ", lowerLimit, upperLimit); }

else { done = 0; }}

return limitCheck; }

int yes(void) {

char yesOrNo;

int value = 0;

scanf(" %c", &yesOrNo);

flushKeyboard();

while (yesOrNo != 'y' && yesOrNo != 'Y' && yesOrNo != 'n' && yesOrNo != 'N') {

printf("Only (Y)es or (N)o ");

scanf(" %c", &yesOrNo);

flushKeyboard();

if (yesOrNo == 'y' || yesOrNo == 'Y') {

value = 1; }

else if (yesOrNo == 'n' || yesOrNo == 'N') {

value = 0; }

return value; }

int locateItem(const struct Item item[], int NoOfRecs, int sku, int\*index) {

int i = 0;

int value = 0;

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

if (item[i].sku == sku) {

\*index = i;

value = 1; }

else if (item[i].sku != sku) {

value = 0; } }

return value; }

void search(const struct Item item[], int NoOfRecs) {

int i = 0; int found = -1;

int sku = 0;

printf("Please enter SKU: ");

sku = getInt();

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

if (item[i].sku == sku) {

found = i; } }

if (found >= 0) {

displayItem(item[found], 0); }

else if (found < 0) {

printf("Item not found!\n");}}

void updateItem(struct Item\* itemptr) {

struct Item temp;

int ans = 0;

printf("Enter new data:\n");

temp = itemEntry(itemptr->sku);

printf("Overwrite old data? (Y)es/(N)o: ");

ans = yes();

if (ans == 1) {

\*itemptr = temp;

printf("--== Updated! ==--\n"); }

else if (ans == 0) {

printf("--= Aborted! =--\n");}}

void addItem(struct Item item[], int \*NoOfRecs, int sku) {

struct Item temp;

int ans = 0;

int count = \*NoOfRecs;

if (count == MAX\_ITEM\_NO) {

printf("Cant+ Storage Full!\n")

else if (count < MAX\_ITEM) {

temp = itemEntry(sku);

printf("Add? (Y)es/(N)o: ");

ans = yes();

if (ans == 1) {

item[count] = temp;

printf("--== Added! ==--\n");

count++;

\*NoOfRecs = count;}

else if (ans == 0) {

printf("-- Aborted!=--\n");}}}

void addOrUpdateItem(struct Item item[], int\* NoOfRecs) {

int sku = 0;

int ans = 0;

int found = -1;

int i = 0;

printf("Please enter SKU: ");

sku = getIntLimited(SKU\_MIN, SKU\_MAX);

for (i = 0; i < \*NoOfRecs; i++) {

if (item[i].sku == sku) {

found = i; } }

if (found >= 0) {

displayItem(item[found], 0);

printf("Item already exists, Update? (Y)es/(N)o: ");

ans = yes();

if (ans == 1) {

updateItem(&item[found]);}

if (ans == 0) {

printf("--Aborted! --\n");}

else if (found == -1) {

addItem(item, NoOfRec, sku); }

saveItems(item, DATAFILE, \*NoOfRecs);}

void adjustQuantity(struct Item item[], int NoOfRecs, int stock)

int sku = 0;

int checkedOut = 0;

int found = -1;

int stocked = 0;

int isLow = 0; int i = 0;

printf("Please enter SKU: ");

sku = getIntLimited(SKU\_MIN, SKU\_MAX);

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

if (item[i].sku == sku) {

found = i; }}

if (found >= 0) {

displayItem(item[found], 0);

if (stock == 0) {

printf("Please enter the quantity to checkout; Maximum of %d or 0 to abort: ", item[found].quantity);

checkedOut = getIntLimited(0, item[found].quantity);

if (checkedOut == 0) {

printf("-- Aborted! =--\n");}

else if (checkedOut > 0) {

printf("-- Checked out! --\n");

item[found].quantity -= checkedOut;

isLow = isLowQuantity(item[found]);

if (isLow == 1) {

printf("Quantity is low"); }}}

else if (stock == 1) {

printf("Please enter the quantity to stock; Maximum of %d or 0 to abort: ", MAX\_QTY - item[found].quantity);

stocked = getIntLimited(0, MAX\_QTY - item[found].quantity);

if (stocked == 0) {

printf("-- Aborted! --\n"); }

else if (stocked > 0) {

printf("--=Stocked! =--\n");

item[found].quantity += stocked; }}}

else if (found == -1) {

printf("SKU not found!\n"); }}

void saveItem(struct Item item, FILE\* dataFile) {

fprintf(dataFile, "%d,%d,%d,%.2lf,%d,%s\n", item.sku, item.quantity, item.minQuantity, item.price, item.isTaxed, item.name); }

int loadItem(struct Item\* temp, FILE\* dataFile) {

int isTrue = 0; int rv = 0;

struct Item item = { 0 };

rv = fscanf(dataFile, "%d,%d,%d,%lf,%d, %20[^\n]", &item.sku, &item.quantity, &item.minQuantity, &item.price, &item.isTaxed, item.name);

if (rv == 6) {

isTrue = 1; }

\*temp = item;

return isTrue; }

int saveItems(const struct Item item[], char fileName[], int NoOfRecs) {

int isTrue = 1;

int i = 0; int rv = 1;

FILE \*fp = fopen(fileName, "w+");

if (fp != NULL) {

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

saveItem(item[i], fp); }

else if (fp == NULL) {

rv = 0; }

fclose(fp);

return rv; }

int loadItems(struct Item item[], char fileName[], int\* NoOfRecsPtr) {

int count = 0;

struct Item temp;

FILE \*fp = fopen(fileName, "r");

if (fp != NULL) {

while (fscanf(fp, "%d,%d,%d,%lf,%d, %20[^\n]", &temp.sku, &temp.quantity, &temp.minQuantity, &temp.price, &temp.isTaxed, temp.name) != EOF) {

count++; }

rewind(fp);

int i = 0;

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

loadItem(&item[i], fp); }

\*NoOfRecsPtr = count; }

fclose(fp); return 1; }

void searchByName(void) {

struct Item temp = { 0 };

char name[21];

int match = 0;

printf("item name to search: ");

scanf("%20[^\n]", name);

flushKeyboard();

FILE \*fp = fopen(DATAFILE, "r");

if (fp != NULL) {

while (fscanf(fp, "%d,%d,%d,%lf,%d, %20[^\n]", &temp.sku, &temp.quantity, &temp.minQuantity, &temp.price, &temp.isTaxed, temp.name) != EOF) {

match = strcmp(temp.name, name); }

if (match == 0) {

printf("\nItem named %s is found:\n", temp.name);

displayItem(temp, FORM);

printf("\n"); }

if (match != 0) {

printf("Item is not found. ensure the name is correct.\n\n", name); }

fclose(fp); } }

void deleteItems(struct Item item[], int \*NoOfRecs) {

int index = -1;

int count = \*NoOfRecs;

int ans = 0; int sku = 0;

int i = 0; int j = 0; int l = 0;

struct Item temp [MAX\_ITEM] = { 0 };

printf("enter SKU to delete: ");

sku = getIntLimited(SKU\_MIN, SKU\_MAX);

FILE \*fp = fopen(DATAFILE, "r+");

if (fp != NULL) {

for (i = 0; i < \*NoOfRecs; i++) {

fscanf(fp, "%d,%d,%d,%lf,%d, %20[^\n]", &temp[i].sku, &temp[i].quantity, &temp[i].minQuantity, &temp[i].price, &temp[i].isTaxed, temp[i].name);

if (temp[i].sku == sku) {

index = i; } }

if (index < 0) {

printf("Item not found! Aborted!!\n"); }

if (index >= 0) {

displayItem(temp[index], FORM);

printf("Confirm to delete item SKU \"%d\" (Y)es/(N)o? ", sku);

ans = yes();

if (ans == 1) {

for (j = index; j < \*NoOfRecs; j++) {

temp[j] = temp[j + 1]; }

count--;

for (l = 0; l < count; l++) {

item[l] = temp[l]; }

\*NoOfRecs = count;

saveItems(item, DATAFILE, \*NoOfRecs);

printf("-- Item deleted!--\n");

fclose(fp); }

else if (ans == 0) {

printf("-- Aborted! =--\n"); }}}}

int isJediMaster(const char name[]) {

int isJedi = 0;

char jediName[31];

FILE \*fp = fopen("jedi\_master.txt", "r");

if (fp != NULL) {

while (fscanf(fp, " %30[^\n]", jediName) != EOF) {

if (strcmp ( jediName, name) == 0) {

isJedi = 1; }}

fclose(fp); }

return isJedi; }

void formatJediPhoneRecord(char formattedRecord[], const char fullName[], const char npa[], const char co[], const char number[]) {

int length = 0;

int numSpace = 0;

int isJedi = 0; int i = 0;

isJedi = isJediMaster(fullName);

length = strlen(fullName);

strcpy(formattedRecord, fullName);

if (length > 16) {

formattedRecord[14] = '\0';

strcat(formattedRecord, ".."); }

if (length < 16) {

numSpace = 16 - length;

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

strcat(formattedRecord, " "); } }

strcat(formattedRecord, " (");

strcat(formattedRecord, npa);

strcat(formattedRecord, ") ")

strcat(formattedRecord, co);

strcat(formattedRecord, "-");

strcat(formattedRecord, number);

if (isJedi == 1) {

strcat(formattedRecord, " Jedi Master"); } }

//// Reading from a file

#include <stdio.h>

int main(void) {

FILE \*fp = NULL;

int sku;

double price;

fp = fopen("produce.txt","r");

if (fp != NULL) {

while(fscanf(fp, "%d%lf", &sku, &price) != EOF){

printf("%5d %6.2lf\n", sku, price);

rewind(fp);

while (fscanf(fp, "%d%lf", &sku, &price) != EOF)

printf("%5d %6.2lf\n", sku, price);

fclose(fp); }

else {

printf("Failed to open file\n"); }

return 0; }

**// Reading from a file**

**#include <stdio.h>**

**int main(void) {**

**FILE \*fp = NULL;**

**int sku;**

**double price;**

**fp = fopen("produce.txt","r");**

**if (fp != NULL) {**

**while(fscanf(fp, "%d%lf",**

**&sku, &price) != EOF)**

**printf("%5d %6.2lf\n" sku, price);**

**rewind(fp);**

**while (fscanf(fp, "%d%lf",**

**&sku, &price) != EOF)**

**printf("%5d %6.2lf\n",**

**sku, price);**

**fclose(fp); } else {**

**printf("Failed to open file\n"); }**

**return 0; }**

**// Tabular Data**

**int main(void) {**

**FILE \*fp = NULL;**

**int sku;**

**char status;**

**double price;**

**fp = fopen("sale.txt","r");**

**if (fp != NULL) {**

**printf(" Produce Items\n", "sku Sale Price\n"--\n");**

**while (fscanf (fp, "%d;%c;%lf",  &sku, &status, &price) == 3)**

**printf("%4d %c %8.2lf\n", sku, status, price);**

**fclose(fp); }**

**return 0; }**