// FS mini project source code

#include<stdio.h>

#include<stdlib.h>

#include <iostream>

#include <iomanip>

#include <conio.h>

#include <string.h>

#include <windows.h>

#define WINWIDTH 113

#define WINHEIGHT 15

#ifndef SINLIST\_H\_INCLUDED

#define SINLIST\_H\_INCLUDED

using namespace std;

class LinkedList

{

private:

struct saver

{

char pname[30]; //Product name

int pid; //Product id

int price; //Product price

int tquant; //Total products purchased from wholesale

int aquant; //How many are available in stock

int rack; //Rack number where product is stored

};

struct saver s;

struct node

{

char pname[30]; //Product name

int pid; //Product id

int price; //Product price

int tquant; //Total products purchased from wholesale

int aquant; //How many are available in stock

int rack; //Rack number where product is stored

struct node \*link;

};

typedef struct node \*nodeptr; //nodeptr is a user-defined

//data-type of type struct node \*

nodeptr H;

public:

LinkedList(); //Constructor

~LinkedList(); //Destructor

struct node \* get\_node() //Create a new node

{

return (new node);

}

int getcount();

void add\_front(char [] , int , int , int , int , int );

void add\_end(char [], int , int , int , int , int );

void add\_middle(int,char [], int , int , int , int , int );

void delete\_node(int);

void delete\_node(char []);

void display();

void Search();

void modify();

void billgen(char [], char [], int, int &);

void SaveToFile();

void ReadFromFile(LinkedList &);

int recsnum()

{

FILE \*fp;

fp = fopen("product\_store.txt","r");

fseek(fp,0,2);

int recs = ftell(fp)/sizeof(s);

fclose(fp);

return recs;

}

};

LinkedList::LinkedList()

{

H = NULL;

}

LinkedList::~LinkedList()

{

nodeptr T;

while(H != NULL)

{

T = H;

H = T->link;

delete T;

}

}

int LinkedList::getcount()

{

nodeptr T;

T = H;

int c = 0;

while(T != NULL)

{

T = T->link;

c++;

}

return c;

}

void LinkedList::add\_front(char pname[], int pid, int price, int tquant, int aquant, int rack)

{

if(H == NULL)

{

H = get\_node();

strcpy(H->pname,pname);

H->pid = pid;

H->price = price;

H->tquant = tquant;

H->aquant = aquant;

H->rack = rack;

H->link = NULL;

}

else

{

nodeptr N;

N = get\_node();

strcpy(N->pname,pname);

N->pid = pid;

N->price = price;

N->tquant = tquant;

N->aquant = aquant;

N->rack = rack;

N->link = H;

H = N;

}

}

void LinkedList::add\_end(char pname[], int pid, int price, int tquant, int aquant, int rack)

{

if(H == NULL)

{

H = get\_node();

strcpy(H->pname,pname);

H->pid = pid;

H->price = price;

H->tquant = tquant;

H->aquant = aquant;

H->rack = rack;

H->link = NULL;

}

else

{

nodeptr T;

T = H;

while(T->link != NULL) //Traverse to last node

T = T->link;

T->link = get\_node();

strcpy(T->link->pname,pname);

T->link->pid = pid;

T->link->price = price;

T->link->tquant = tquant;

T->link->aquant = aquant;

T->link->rack = rack;

T->link->link = NULL;

}

}

void LinkedList::add\_middle(int pos, char pname[], int pid, int price, int tquant, int aquant, int rack)

{

nodeptr T,N;

T = H;

if(pos == 1)

add\_front(pname,pid,price,tquant,aquant,rack);

else if(pos > getcount())

cout << "Enter a valid position" << endl;

else

{

for(int i = 1 ; i < pos - 1 ; i++)

{

T = T->link;

}

N = get\_node();

strcpy(N->pname,pname);

N->pid = pid;

N->price = price;

N->tquant = tquant;

N->aquant = aquant;

N->rack = rack;

N->link = T->link;

T->link = N;

}

}

void LinkedList::delete\_node(int nodenum)

{

nodeptr T,P;

T = H;

if(T == NULL);

else if(T->link == NULL) //Only one node

{

H = NULL;

delete T;

}

else

{

if(nodenum > getcount())

cout << "Enter a valid index" << endl;

else

{

for(int i = 1 ; i < nodenum - 1 ; i++)

{

T = T->link;

}

P = T->link;

T->link = P->link;

delete P;

}

}

}

void LinkedList::delete\_node(char name[])

{

nodeptr T,P;

T = H;

P = T;

if(T == NULL);

else if(stricmp(T->pname,name) == 0)

{

T = T ->link;

H = T;

delete P;

P = T;

}

else

{

int flag = 0;

for(int i = 1 ; i < getcount() ; i++)

{

if(stricmp(T->link->pname,name) == 0)

{

flag = 1;

break;

}

T = T->link;

}

if(flag == 1)

{

P = T->link;

T->link = T->link->link;

delete P;

cout << "Record deleted" << endl;

}

else

cout << "Product not found" << endl;

}

}

void LinkedList::display()

{

nodeptr T;

int cnt = 1;

T = H;

cout << "Number of records = " << getcount() << endl;

cout << "Product name" << "\t"

<< "product id " << "\t"

<< "Price" << "\t"

<< "Total quantity" << "\t"

<< "Availaible quantity" << "\t"

<< "Rack number" << "\t"

<< "Address" <<"\t\t"

<< "Link" << endl << endl;

if(T == NULL)

cout << "List empty" << endl;

while(T != NULL)

{

cout << T->pname << "\t\t"

<< T->pid << "\t\t"

<< T->price << "\t\t"

<< T->tquant << "\t\t"

<< T->aquant << " \t\t "

<< T->rack << "\t\t"

<< T << "\t" << T->link << endl;

cnt++;

T = T->link;

}

}

void LinkedList::modify()

{

nodeptr T;

char pname[30];

//int data;

cout << "Enter name whose record to be modified : ";

cin >> pname;

int cnt = 1;

int flag = 0;

T = H;

if(T == NULL)

cout << "List empty" << endl;

while(T != NULL)

{

if(stricmp(T->pname,pname) == 0)

{

cout << "----------------------------------------------------------------" << endl;

cout << "Node number " << cnt << endl;

cout << "----------------------------------------------------------------" << endl;

cout << "Node address = " << T << endl;

cout << "Product name : " << T->pname << endl

<< "Product id : " << T->pid << endl

<< "Price : " << T->price << endl

<< "Total products initially : " << T->tquant << endl

<< "Available products : " << T->aquant << endl

<< "Rack number : " << T->rack << endl << endl;

cout << "Re-enter data : ";

cout<<"enter the name of the product";

cin >> T->pname ;

cout<<"enter the id of the product";

cin >> T->pid ;

cout<<"enter the price of the product";

cin>> T->price ;

cout<<"enter the quantity of the product";

cin>> T->tquant ;

cout<<"enter the available quantity of the product";

cin>>T->aquant;

cout<<"enter the rack of the product";

cin >> T->rack;

flag = 1;

break;

}

if(flag == 1)

break;

cnt++;

T = T->link;

}

}

void LinkedList::SaveToFile()

{

FILE \*fp;

fp = fopen("BILL","w");

fclose(fp);

nodeptr T;

T = H;

while(T != NULL)

{

strcpy(s.pname,T->pname);

s.pid = T->pid;

s.price = T->price;

s.tquant = T->tquant;

s.aquant = T->aquant;

s.rack = T->rack;

T = T->link;

fp = fopen("product\_store.txt","a");

fwrite(&s,sizeof(s),1,fp);

fclose(fp);

}

}

void LinkedList::ReadFromFile(LinkedList &arg)

{

FILE \*fp;

int recs = recsnum();

fp = fopen("BILL","r");

for(int i = 1 ; i <= recs ; i++)

{

fread(&s,sizeof(s),1,fp);

arg.add\_end(s.pname,s.pid,s.price,s.tquant,s.aquant,s.rack);

}

fclose(fp);

}

void LinkedList::billgen(char filename[],char name[], int quantity, int &total)

{

FILE \*fp;

fp = fopen(filename,"a");

int flag = 0;

nodeptr T;

T = H;

while(T != NULL)

{

if(stricmp(T->pname,name) == 0)

{

if(quantity > T->aquant)

cout << "Requested number of products are not available" << endl;

else

{

fprintf(fp,"%-30s\t%-6d\t%-6d\t\t%-6d\n",T->pname,quantity,T->price,quantity\*T->price);

T->aquant = T->aquant - quantity;

total = total + quantity\*T->price;

flag = 1;

break;

fclose(fp);

}

}

T = T->link;

if(flag == 1)

break;

}

if(flag == 0)

{

fclose(fp);

cout << "Requested Product is not available" << endl;

}

}

void LinkedList::Search()

{

nodeptr T;

char pname[30];

cout << "Enter product name to be searched : ";

cin >> pname;

int flag = 0;

T = H;

if(T == NULL)

cout << "List empty" << endl;

while(T != NULL)

{

if(stricmp(T->pname,pname) == 0)

{

cout << "----------------------------------------------------------------" << endl;

cout << "Enter the product name to be searched for : " << T->pname << endl

<< "Enter the product id to be searched for : " << T->pid << endl

<< "Enter the Price of the product : " << T->price << endl

<< "Enter Total products available quantity initially : " << T->tquant << endl

<< "enter the present available quantity of product : " << T->aquant << endl

<< "Enter the rack number of the product: " << T->rack << endl << endl;

cout << "----------------------------------------------------------------" << endl;

getch();

flag = 1;

break;

}

if(flag == 1)

break;

T = T->link;

}

if(flag == 0)

{

cout << "Product not found" << endl;

getch();

}

}

#endif // SINLIST\_H\_INCLUDED

void filler();

void liner();

void welcome();

void exitscreen();

char pname[30]; //Product name

int pid; //Product id

int price; //Product price

int tquant; //Total products purchased from wholesale

int aquant; //How many are available in stock

int rack;

char filename[50];

class store

{

LinkedList obj;

int t;

public:

store()

{

t = 0;

obj.ReadFromFile(obj);

}

void add()

{

cout<<"enter the product name\n";

cin>>pname;

cout<<"enter the product id\n";

cin>>pid;

cout<<"enter the product price\n";

cin>>price;

cout<<"enter the product total quantity\n";

cin>>tquant;

cout<<"enter the product available quantity\n";

cin>>aquant;

cout<<"Enter the rack number\n";

cin>>rack;

obj.add\_end(pname,pid,price,tquant,aquant,rack);

}

void display()

{

obj.display();

}

int getrecs()

{

return obj.recsnum();

}

void modify()

{

obj.modify();

}

void PrintBill();

void deleteproduct()

{

char name[30];

cout << "Enter name of product : ";

cin >> name;

obj.delete\_node(name);

}

void SearchProduct()

{

obj.Search();

}

void totalprinter();

~store()

{

cout << "Number of records = " << obj.getcount() << endl;

obj.SaveToFile();

exitscreen();

}

};

void store::PrintBill()

{

int choice;

char sname[] = {"Product Name"};

char name[30];

int quantity;

cout << "Enter filename for bill with full path and file extension : ";

cin >> filename;

FILE \*fp;

fp = fopen(filename,"a");

for(int i = 0 ; i < 70 ; i++)

{

fprintf(fp,"\*");

}

fprintf(fp,"\n");

fprintf(fp,"%-30s\tQty.\tPrice/item\tAmount\n",sname);

for(int i = 0 ; i < 70 ; i++)

{

fprintf(fp,"\*");

}

fprintf(fp,"\n");

fclose(fp);

while(1)

{

cout << "Continue Purchasing Product ?\n1 = yes , 2 = no" << endl;

cout << "choice : ";

cin >> choice;

if(choice == 1)

{

cout << "Enter product name and quantity : " << endl;

cin >> name >> quantity;

obj.billgen(filename,name,quantity,t);

}

else

break;

}

fp = fopen(filename,"app");

}

void store::totalprinter()

{

FILE \*fp;

fp = fopen(filename,"a");

fprintf(fp,"\n");

for(int i = 0 ; i < 70 ; i++)

{

fprintf(fp,"\*");

}

fprintf(fp,"\n");

for(int i = 0 ; i < 70 ; i++)

{

fprintf(fp,"\*");

}

fprintf(fp,"\n");

fprintf(fp,"Total = %d\n",t);

for(int i = 0 ; i < 70 ; i++)

{

fprintf(fp,"\*");

}

fprintf(fp,"\n");

fclose(fp);

}

int main()

{

store obj;

int ch;

welcome();

cout << endl << "Press any key to continue" << endl;

getch();

while(1)

{

system("CLS");

cout << "Enter your choice : " << endl

<< "1) Add a new product to list" << endl

<< "2) Display all the product information" << endl

<< "3) Modify an existing product" << endl

<< "4) Delete an existing Product" << endl

<< "5) Search for a Product" << endl

<< "6) sell a product and Generate Bill" << endl

<< "7) Exit " << endl;

cout << "Choice : ";

cin >> ch;

switch(ch)

{

case 1 : obj.add();

break;

case 2 : obj.display();

getch();

break;

case 3 : obj.modify();

break;

case 4 : obj.deleteproduct();

getch();

break;

case 5 : obj.SearchProduct();

break;

case 6 : obj.PrintBill();

obj.totalprinter();

getch();

break;

case 7 : return 0;

}

}

return 0;

}

void filler()

{

for(int i = 0 ; i < WINWIDTH ; i++)

{

cout << "\*";

}

cout << "\a" << endl;

}

void liner()

{

for(int i = 0 ; i < WINHEIGHT ; i++)

cout << endl;

}

void welcome()

{

liner();

cout << "Loading..." << endl;

Sleep(500);

char msg[] = "WELCOME TO ELECTRONIC STORE MANAGEMENT SYSTEM";

int len = strlen(msg);

int x;

x = (WINWIDTH/2) - (len/2);

for(int i = 0 ; i < WINWIDTH ; i++)

{

printf("%c",177);

Sleep(55);

}

cout << endl;

system("CLS");

liner();

filler();

cout << setw(len+x) << msg << endl;

filler();

}

void exitscreen()

{

system("CLS");

char msg[] = "THANKYOU";

int len = strlen(msg);

int x;

x = (WINWIDTH/2) - (len/2);

liner();

filler();

cout << setw(len+x) << msg << endl;

filler();

}