#include "stdafx.h" //// Удалить потом

#include "iostream"

#include "conio.h"

#include <time.h>

using namespace std;

struct Category

{

int ID\_category;

char\* Name;

Category \*next;

};

struct listCat

{

Category \*head;

};

struct Tovar

{

int ID\_tovar;

char\* Name;

bool Type;

int ID\_category;

int Cost;

Tovar \*next;

};

struct listTov

{Tovar \*head;

};

struct Kompl

{

int ID\_kompl;

char\* Name;

int ID\_category;

int Cost;

Kompl \*next;

};

struct listKom

{Kompl \* head;

};

struct Svaz

{

int ID\_svaz;

int ID\_tovar;

int ID\_kompl;

int Kolvo;

Svaz \*next;

};

struct listSvaz

{

Svaz \*head;

};

struct Data

{

unsigned day;

unsigned month;

unsigned year;

Data\* next;

};

struct listData

{Data\* head;

};

struct Prodaza

{

int ID\_prodaza;

int ID\_tovar;

int Kolvo;

Data D\_pradaza;

Prodaza \* next;

};

struct listProd

{Prodaza \* head;

};

class List {

public:

listCat \*cat;

listTov \* tov;

listKom \* kom;

listSvaz \* svaz;

listProd \* prod;

listData \* data;

List() {

cat = new listCat;

tov = new listTov;

kom = new listKom;

svaz = new listSvaz;

prod = new listProd;

data = new listData;

cat->head = NULL;

tov->head = NULL;

kom->head = NULL;

svaz->head = NULL;

prod->head = NULL;

data->head = NULL;

}

~List() {

while (cat->head != NULL) {

Category \*temp = cat->head;

cat->head = cat->head->next;

delete temp;}

while (tov->head != NULL) {

Tovar \*temp = tov->head;

tov->head = tov->head->next;

delete temp;}

while (kom->head != NULL) {

Kompl \*temp = kom->head;

kom->head = kom->head->next;

delete temp;}

while (svaz->head != NULL) {

Svaz \*temp = svaz->head;

svaz->head = svaz->head->next;

delete temp;}

while (prod->head != NULL) {

Prodaza \*temp = prod->head;

prod->head = prod->head->next;

delete temp;}

}

void List::addCat(int ID, char \*Name) {

for (Category \*temp = cat->head; temp != NULL; temp = temp->next) {

if (temp->ID\_category == ID) {

cout << "Categ: ID = " << ID << " already exist" << endl;

return;

}

}

for (Category \*temp = cat->head; temp != NULL; temp = temp->next) {

if (temp->Name == Name) {

cout << "Categ: Name = " << Name << " already exist" << endl;

return;

}

}

Category \*temp = new Category;

temp->ID\_category = ID;

temp->Name = Name;

temp->next = cat->head;

cat->head = temp;

}

void List::addTov(int IDt, char \*Name, int Cost, int Categ, bool Type) {

for (Tovar \*temp = tov->head; temp != NULL; temp = temp->next) {

if (temp->ID\_tovar == IDt) {

cout << "Tovar: ID = " << IDt << " already exist" << endl;

return;

}

}

for (Category \*tmp = cat->head; tmp != NULL; tmp = tmp->next) {

if (tmp->ID\_category == Categ) {

Tovar \*temp = new Tovar;

temp->ID\_tovar = IDt;

temp->Name = Name;

temp->Cost = Cost;

temp->ID\_category = Categ;

temp->Type = Type;

temp->next = tov->head;

tov->head = temp;

return;

}

}

cout << "This ID Tovar not exist" << endl;

}

void List::addKom(int IDk, char \*Name, int Cost, int Categ) {

for (Kompl \*temp = kom->head; temp != NULL; temp = temp->next) {

if (temp->ID\_kompl == IDk) {

cout << "Kompl: ID = " << IDk << " already exist" << endl;

return;

}

}

for (Category \*tmp = cat->head; tmp != NULL; tmp = tmp->next) {

if (tmp->ID\_category == Categ) {

Kompl \*temp = new Kompl;

temp->ID\_kompl = IDk;

temp->Name = Name;

temp->Cost = Cost;

temp->ID\_category = Categ;

temp->next = kom->head;

kom->head = temp;

return;

}

}

cout << "This ID Category not exist" << endl;

}

void List::addSvaz(int ID, int IDt, int IDk, int kol) {

for (Svaz \*temp = svaz->head; temp != NULL; temp = temp->next) {

if (temp->ID\_kompl == ID) {

cout << "Svaz: ID = " << IDk << " already exist" << endl;

return;

}

}

for (Tovar \*tmp = tov->head; tmp != NULL; tmp = tmp->next) {

for (Kompl \*tmp2 = kom->head; tmp2 != NULL; tmp2 = tmp2->next) {

if ((tmp->Type == false) && (tmp->ID\_tovar == IDt) && (tmp2->ID\_kompl == IDk)) {

Svaz \*temp = new Svaz;

temp->ID\_svaz = ID;

temp->ID\_tovar = IDt;

temp->ID\_kompl = IDk;

temp->Kolvo = kol;

temp->next = svaz->head;

svaz->head = temp;

return;

}

}

}

cout << "This \"Kompl\" and \"ID\" not exist" << endl;

}

void List::addProd(int ID, int IDt, int kol, Date date) {

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

if (temp->ID\_prodaza == ID) {

cout << "Prod: ID = " << ID << " already exist" << endl;

return;

}

}

for (Tovar \*tmp = tov->head; tmp != NULL; tmp = tmp->next) {

if (tmp->ID\_tovar == IDt) {

Prodaza \*temp = new Prodaza;

temp->ID\_prodaza = ID;

temp->ID\_tovar = IDt;

temp->Kolvo = kol;

temp->D\_pradaza.day = date.day;

temp->D\_pradaza.month = date.month;

temp->D\_pradaza.year = date.year;

temp->next = prod->head;

prod->head = temp;

return;

}

}

cout << "This ID Tovar not exist" << endl;

}

//int myStrlen (char \* str) {

// int length = 0;

// while (str[length] != '\0') {

// length++;

// }

// return length;

//}

int myAtoi (char \* str) {

int length = 0;

while (str[length] != '\0') {

if (!isdigit(str[length])) {

return 0;

}

length++;

}

return atoi(str);

}

Date List::inputDate() {

Date date;

cout << "Input day: ";

char temp[40];

do {

cin.getline(temp, 40);

if ((myAtoi(temp) <= 0) || (myAtoi(temp) > 31)) {

cout << "Error of input day" << endl << "Try again: ";

} else {

break;

}

} while (true);

date.day = myAtoi(temp);

cout << "Input month: ";

char temp2[40];

do {

cin.getline(temp2, 40);

if ((myAtoi(temp2) <= 0) || (myAtoi(temp2) > 12)) {

cout << "Error of input month" << endl << "Try again: ";

} else {

break;

}

} while (true);

date.month = myAtoi(temp2);

cout << "Input year: ";

char temp3[80];

do {

cin.getline(temp3, 80);

if ((myAtoi(temp3) <= 0) || (myAtoi(temp3) < 1994) || (myAtoi(temp3) > 2013)) {

cout << "Error of input year" << endl << "Try again: ";

} else {

break;

}

} while (true);

date.year = myAtoi(temp3);

return date;

}

void List::findByIDProd(int ID, int IDt, int kol, Date date) {

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

if (temp->ID\_prodaza == ID) {

temp->ID\_tovar = IDt;

temp->Kolvo = kol;

temp->D\_pradaza.day = date.day;

temp->D\_pradaza.month = date.month;

temp->D\_pradaza.year = date.year;

}

}

}

void List::findByIDCat(int ID, char\* Name) {

for (Category \*temp = cat->head; temp != NULL; temp = temp->next) {

if (temp->ID\_category == ID) {

temp->Name = Name;

cout << "Success of change" << endl;

return;

}

}

cout << "Error of change" << endl;

}

void List::findByIDTov(int IDt, char\* Name, int Cost, int ID, bool Type) {

for (Tovar \*temp = tov->head; temp != NULL; temp = temp->next) {

if (temp->ID\_tovar == IDt) {

temp->Name = Name;

temp->Cost = Cost;

temp->ID\_category = ID;

temp->Type = Type;

}

}

}

void List::findByIDKom(int IDk, char\* Name, int Cost, int ID) {

for (Kompl \*temp = kom->head; temp != NULL; temp = temp->next) {

if (temp->ID\_kompl == IDk) {

temp->Name = Name;

temp->Cost = Cost;

temp->ID\_category = ID;

}

}

}

void List::findByIDSvaz(int ID, int Kol) {

for (Svaz \*temp = svaz->head; temp != NULL; temp = temp->next) {

if (temp->ID\_svaz == ID) {

temp->Kolvo = Kol;

}

}

}

bool List::existIDCat(int ID) {

for (Category \*temp = cat->head; temp != NULL; temp = temp->next) {

if (temp->ID\_category == ID) {

return true;

}

}

return false;

}

bool List::existIDTov(int ID) {

for (Tovar \*temp = tov->head; temp != NULL; temp = temp->next) {

if (temp->ID\_tovar == ID) {

return true;

}

}

return false;

}

bool List::existIDKompl(int ID) {

for (Kompl \*temp = kom->head; temp != NULL; temp = temp->next) {

if (temp->ID\_kompl == ID) {

return true;

}

}

return false;

}

bool List::existIDSvaz(int ID) {

for (Svaz \*temp = svaz->head; temp != NULL; temp = temp->next) {

if (temp->ID\_svaz == ID) {

return true;

}

}

return false;

}

bool List::existIDProd(int ID) {

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

if (temp->ID\_prodaza == ID) {

return true;

}

}

return false;

}

void List::delByValueProd(int ID) {

while (prod->head != NULL) {

if (prod->head->ID\_prodaza == ID) {

Prodaza \*temp = prod->head;

prod->head = prod->head->next;

delete temp;

} else {

break;

}

}

if (prod->head == NULL) {

return;

}

Prodaza \*prevNode = prod->head;

while (prevNode->next != NULL) {

if (prevNode->next->ID\_prodaza == ID) {

Prodaza \*temp = prevNode->next;

prevNode->next = prevNode->next->next;

delete temp;

} else {

prevNode = prevNode->next;

}

}

}

void List::delByValueCateg(int ID) {

while (cat->head != NULL) {

if (cat->head->ID\_category == ID) {

for (Kompl \*temp = kom->head; temp != NULL; temp = temp->next) {

if (temp->ID\_category == ID) {

temp->ID\_category = 0;

}

}

for (Tovar \*temp = tov->head; temp != NULL; temp = temp->next) {

if (temp->ID\_category == ID) {

temp->ID\_category = 0;

}

}

Category \*temp = cat->head;

cat->head = cat->head->next;

delete temp;

} else {

break;

}

}

if (cat->head == NULL) {

return;

}

Category \*prevNode = cat->head;

while (prevNode->next != NULL) {

if (prevNode->next->ID\_category == ID) {

for (Kompl \*temp = kom->head; temp != NULL; temp = temp->next) {

if (temp->ID\_category == ID) {

temp->ID\_category = 0;

}

}

for (Tovar \*temp = tov->head; temp != NULL; temp = temp->next) {

if (temp->ID\_category == ID) {

temp->ID\_category = 0;

}

}

Category \*temp = prevNode->next;

prevNode->next = prevNode->next->next;

delete temp;

} else {

prevNode = prevNode->next;

}

}

}

void List::delByValueTov(int ID) {

while (tov->head != NULL) {

if (tov->head->ID\_tovar == ID) {

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

if (temp->ID\_tovar == ID) {

temp->ID\_tovar = 0;

}

}

for (Svaz \*temp = svaz->head; temp != NULL; temp = temp->next) {

if (temp->ID\_tovar == ID) {

temp->ID\_tovar = 0;

}

}

Tovar \*temp = tov->head;

tov->head = tov->head->next;

delete temp;

} else {

break;

}

}

if (tov->head == NULL) {

return;

}

Tovar \*prevNode = tov->head;

while (prevNode->next != NULL) {

if (prevNode->next->ID\_tovar == ID) {

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

if (temp->ID\_tovar == ID) {

temp->ID\_tovar = 0;

}

}

for (Svaz \*temp = svaz->head; temp != NULL; temp = temp->next) {

if (temp->ID\_tovar == ID) {

temp->ID\_tovar = 0;

}

}

Tovar \*temp = prevNode->next;

prevNode->next = prevNode->next->next;

delete temp;

} else {

prevNode = prevNode->next;

}

}

}

void List::delByValueKompl(int ID) {

while (kom->head != NULL) {

if (kom->head->ID\_kompl == ID) {

Kompl \*temp = kom->head;

kom->head = kom->head->next;

delete temp;

} else {

break;

}

}

if (kom->head == NULL) {

return;

}

Kompl \*prevNode = kom->head;

while (prevNode->next != NULL) {

if (prevNode->next->ID\_kompl == ID) {

Kompl \*temp = prevNode->next;

prevNode->next = prevNode->next->next;

delete temp;

} else {

prevNode = prevNode->next;

}

}

}

void List::delByValueSvaz(int ID) {

while (svaz->head != NULL) {

if (svaz->head-> ID\_svaz== ID) {

Svaz \*temp = svaz->head;

svaz->head = svaz->head->next;

delete temp;

} else {

break;

}

}

if (svaz->head == NULL) {

return;

}

Svaz \*prevNode = svaz->head;

while (prevNode->next != NULL) {

if (prevNode->next->ID\_svaz == ID) {

Svaz \*temp = prevNode->next;

prevNode->next = prevNode->next->next;

delete temp;

} else {

prevNode = prevNode->next;

}

}

}

void List::printCat() {

cout << "Print Cat:" << endl << endl;

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

cout.setf(ios::left, ios::adjustfield);

cout << setw(4) << " ID" << setw(30) << "| Name" << endl;

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

for (Category \*temp = cat->head; temp != NULL; temp = temp->next) {

cout << " " << setw(3) << temp->ID\_category << "| " << setw(30) << temp->Name << endl;

}

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

}

void List::printProd() {

cout << "Print Prodaza:" << endl << endl;

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

cout.setf(ios::left, ios::adjustfield);

cout << setw(4) << " ID" << setw(7) << "| Tovar" << setw(7) << "| Kolvo" << setw(10) << "| Data"<< endl;

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

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

cout << " " << setw(3) << temp->ID\_prodaza << "| " << setw(5) << temp->ID\_tovar << "| " << setw(5) << temp->Kolvo

<< "| " << temp->D\_pradaza.day << " / " << temp->D\_pradaza.month << " / " << temp->D\_pradaza.year <<endl;

}

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

}

void List::printTov() {

cout << "Print Tovar:" << endl << endl;

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

cout.setf(ios::left, ios::adjustfield);

cout << setw(4) << " ID" << setw(40) << "| Name" << setw(10) << "| Cost" << setw(10) << "| ID Cat" << setw(7) << "| Type" << endl;

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

for (Tovar \*temp = tov->head; temp != NULL; temp = temp->next) {

//cout << " " << setw(3) << temp->ID\_tovar << "| " << setw(30) << temp->Name << endl;

cout << " " << setw(3) << temp->ID\_tovar << "| " << setw(38) << temp->Name << "| " << setw(8) << temp->Cost

<< "| " << setw(8) << temp->ID\_category << "| " << setw(7) << temp->Type << endl;

}

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

}

void List::printKompl() {

cout << "Print Kompl:" << endl << endl;

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

cout.setf(ios::left, ios::adjustfield);

cout << setw(4) << " ID" << setw(40) << "| Name" << setw(10) << "| Category" << setw(10) << "| Cost" << endl;

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

for (Kompl \*temp = kom->head; temp != NULL; temp = temp->next) {

cout << " " << setw(3) << temp->ID\_kompl << "| " << setw(38) << temp->Name << "| " <<

setw(8) << temp->ID\_category<< "| " << setw(8) << temp->Cost << endl;

}

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

}

void List::printSvaz() {

cout << "Print Svaz:" << endl << endl;

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

cout.setf(ios::left, ios::adjustfield);

cout << setw(4) << " ID" << setw(8) << "| ID Tov" << setw(11) << "| ID Kompl" << setw(10) << "| Kolvo" << endl;

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

for (Svaz \*temp = svaz->head; temp != NULL; temp = temp->next) {

cout << " " << setw(3) << temp->ID\_kompl << "| " << setw(6) << temp->ID\_tovar << "| " <<

setw(9) << temp->ID\_kompl<< "| " << setw(8) << temp->Kolvo << endl;

}

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

}

void List::outFileCat(char \* path) {

FILE \*out;

if ((out = fopen(path,"wb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct CategoryTemp {

int ID\_category;

char \*Name;

};

for (Category \*temp = cat->head; temp != NULL; temp = temp->next) {

CategoryTemp head;

head.ID\_category = temp->ID\_category;

head.Name = temp->Name;

fwrite(&head, sizeof(head), 1, out);

}

fclose(out);

/\*ofstream outFileCat;

outFileCat.open(path);

for (Category \*temp = cat->head; temp != NULL;

outFileCat << temp->ID\_category << endl,

outFileCat << temp->Name << endl,

temp = temp->next);

outFileCat.close();\*/

}

void List::outFileTov(char \* path) {

FILE \*out;

if ((out = fopen(path,"wb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct TovarTemp {

int ID\_Tovar;

char\* Name;

int Cost;

int ID\_Category;

bool Type;

};

for (Tovar \*temp = tov->head; temp != NULL; temp = temp->next) {

TovarTemp head;

head.ID\_Tovar = temp->ID\_tovar;

head.Name = temp->Name;

head.Cost = temp->Cost;

head.ID\_Category = temp->ID\_category;

head.Type = temp->Type;

fwrite(&head, sizeof(head), 1, out);

}

fclose(out);

}

void List::outFileKompl(char \* path) {

FILE \*out;

if ((out = fopen(path,"wb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct KomplTemp {

int ID\_Kompl;

char\* Name;

int ID\_Category;

int Cost;

};

for (Kompl \*temp = kom->head; temp != NULL; temp = temp->next) {

KomplTemp head;

head.ID\_Kompl = temp->ID\_kompl;

head.Name = temp->Name;

head.Cost = temp->Cost;

head.ID\_Category = temp->ID\_category;

fwrite(&head, sizeof(head), 1, out);

}

fclose(out);

}

void List::outFileSvaz(char \* path) {

FILE \*out;

if ((out = fopen(path,"wb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct SvazTemp {

int ID\_Svaz;

int ID\_Tovar;

int ID\_Kompl;

int Kolvo;

};

for (Svaz \*temp = svaz->head; temp != NULL; temp = temp->next) {

SvazTemp head;

head.ID\_Svaz = temp->ID\_svaz;

head.ID\_Tovar = temp->ID\_tovar;

head.ID\_Kompl = temp->ID\_kompl;

head.Kolvo = temp->Kolvo;

fwrite(&head, sizeof(head), 1, out);

}

fclose(out);

}

void List::outFileProd(char \* path) {

FILE \*out;

if ((out = fopen(path,"wb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct ProdTemp {

int ID\_Prod;

int ID\_Tovar;

int Kolvo;

Date Date;

};

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

ProdTemp head;

head.ID\_Prod = temp->ID\_prodaza;

head.ID\_Tovar = temp->ID\_tovar;

head.Kolvo = temp->Kolvo;

head.Date.day = temp->D\_pradaza.day;

head.Date.month = temp->D\_pradaza.month;

head.Date.year = temp->D\_pradaza.year;

fwrite(&head, sizeof(head), 1, out);

}

fclose(out);

}

void List::inFileProd(char \* path) {

FILE \*in;

if ((in = fopen(path,"rb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct ProdazaTemp {

int ID\_Prodaza;

int ID\_Tovar;

int Kolvo;

Date Date;

};

ProdazaTemp head;

while (fread(&head, sizeof(head), 1, in)) {

addProd(head.ID\_Prodaza, head.ID\_Tovar, head.Kolvo, head.Date);

}

fclose(in);

printProd();

}

void List::inFileCat(char \* path) {

//cout << "path->>>" << path;

FILE \*in;

if ((in = fopen(path,"rb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct CategoryTemp {

int ID\_category;

char\* Name;

};

CategoryTemp head;

while (fread(&head, sizeof(head), 1, in)) {

addCat(head.ID\_category, head.Name);

}

fclose(in);

printCat();

//ifstream inFileCat;

//inFileCat.open(path);

//if (!inFileCat) {

// cout << "File " << path << "can not open!" << endl;

// return;

//}

//while (!inFileCat.eof()) {

// int ID;

// char \* name;

// //inFileCat.getline(name, 9);

// inFileCat >> ID >> name;

// cout << "My data: " << ID << "\t" << name << endl;

// addCat(ID, name);

//}

//inFileCat.close();

}

void List::inFileTov(char \* path) {

FILE \*in;

if ((in = fopen(path,"rb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct TovarTemp {

int ID\_Tovar;

char\* Name;

int Cost;

int ID\_Category;

bool Type;

};

TovarTemp head;

while (fread(&head, sizeof(head), 1, in)) {

addTov(head.ID\_Tovar, head.Name, head.Cost, head.ID\_Category, head.Type);

}

fclose(in);

printTov();

}

void List::inFileKompl(char \* path) {

FILE \*in;

if ((in = fopen(path,"rb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct KomplTemp {

int ID\_Komp;

char\* Name;

int ID\_Category;

int Cost;

};

KomplTemp head;

while (fread(&head, sizeof(head), 1, in)) {

addKom(head.ID\_Komp, head.Name, head.Cost, head.ID\_Category);

}

fclose(in);

printKompl();

}

void List::inFileSvaz(char \* path) {

FILE \*in;

if ((in = fopen(path,"rb")) == NULL) {

cout << "File " << path << "can not open!" << endl;

return;

}

struct SvazTemp {

int ID\_Svaz;

int ID\_Tovar;

int ID\_Komp;

int Kolvo;

};

SvazTemp head;

while (fread(&head, sizeof(head), 1, in)) {

addSvaz(head.ID\_Svaz, head.ID\_Tovar, head.ID\_Komp, head.Kolvo);

}

fclose(in);

printSvaz();

}

// больше ли введенная дата чем старт

bool List::isStartDate(Date dateStart, Date dateCheck) {

if (dateStart.year < dateCheck.year) {

return true;

} else {

if (dateCheck.year == dateStart.year) {

if (dateStart.month < dateCheck.month) {

return true;

}else{

if (dateStart.month == dateCheck.month) {

if (dateStart.day < dateCheck.day) {

return true;

} else {

if (dateStart.day == dateCheck.day) {

return true;

}

return false;

}

}

return false;

}

}

return false;

}

}

// меньше ли введенная дата чем end

bool List::isEndDate(Date dateEnd, Date dateCheck) {

if (dateEnd.year > dateCheck.year) {

return true;

} else {

if (dateCheck.year == dateEnd.year) {

if (dateEnd.month > dateCheck.month) {

return true;

}else{

if (dateEnd.month == dateCheck.month) {

if (dateEnd.day > dateCheck.day) {

return true;

} else {

if (dateEnd.day == dateCheck.day) {

return true;

}

return false;

}

}

return false;

}

}

return false;

}

}

int List::summProdPer() {

int cost = 0;

int res = 0;

cout << "Input start of date:" << endl;

Date dateStart = inputDate();

cout << "Input end of date:" << endl;

Date dateEnd = inputDate();

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

if ((isStartDate(dateStart, temp->D\_pradaza)) && (isEndDate(dateEnd, temp->D\_pradaza))) {

for (Tovar \*temp2 = tov->head; temp2 != NULL; temp2 = temp2->next) {

if (temp->ID\_tovar == temp2->ID\_tovar) {

cost = temp2->Cost;

break;

}

}

res += cost \* temp->Kolvo;

}

}

return res;

}

// НЕ ПРОВЕРЕНО!!!

int List::countProdPer() {

int res = 0;

cout << "Input start of date:" << endl;

Date dateStart = inputDate();

cout << "Input end of date:" << endl;

Date dateEnd = inputDate();

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

if ((isStartDate(dateStart, temp->D\_pradaza)) && (isEndDate(dateEnd, temp->D\_pradaza))) {

res += temp->Kolvo;

}

}

return res;

}

void List::summProdCat() {

cout << "Input start of date:" << endl;

Date dateStart = inputDate();

cout << "Input end of date:" << endl;

Date dateEnd = inputDate();

listCat \*catSum = new listCat;

catSum->head = NULL;

for (Prodaza \*temp = prod->head; temp != NULL; temp = temp->next) {

if ((isStartDate(dateStart, temp->D\_pradaza)) && (isEndDate(dateEnd, temp->D\_pradaza))) {

for (Tovar \*temp2 = tov->head; temp2 != NULL; temp2 = temp2->next) {

if (temp->ID\_tovar == temp2->ID\_tovar) {

for (Category \*temp3 = cat->head; temp3 != NULL; temp3 = temp3->next) {

if (temp2->ID\_category == temp3->ID\_category) {

if (catSum->head == NULL) {

Category \*tempCatSum = new Category;

tempCatSum->ID\_category = temp->Kolvo \* temp2->Cost;

tempCatSum->Name = temp3->Name;

tempCatSum->next = catSum->head;

catSum->head = tempCatSum;

break;

}

for (Category \*temp4 = catSum->head; temp4 != NULL; ) {

if (temp4->Name == temp3->Name) {

temp4->ID\_category += temp->Kolvo \* temp2->Cost;

break;

} else {

Category \*tempCatSum = new Category;

tempCatSum->ID\_category = temp->Kolvo \* temp2->Cost;

tempCatSum->Name = temp3->Name;

tempCatSum->next = catSum->head;

catSum->head = tempCatSum;

break;

}

}

}

}

}

}

}

}

cout << "Print Cat sum:" << endl << endl;

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

cout.setf(ios::left, ios::adjustfield);

cout << setw(30) << " Name" << setw(20) << "| Sum" << endl;

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

for (Category \*temp = catSum->head; temp != NULL; temp = temp->next) {

cout << " " << setw(29) << temp->Name << "| " << setw(18) << temp->ID\_category << endl;

}

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

}

// НЕ ПРОВЕРЕНО!!!

int List::findCostTov(int IDt) {

int res = 0;

int kol = 0;

int cost;

for (Svaz \*temp = svaz->head; temp != NULL; temp = temp->next){

if (IDt == temp->ID\_tovar) {

for (Kompl \*temp2 = kom->head; temp2 != NULL; temp2 = temp2->next) {

if (temp->ID\_kompl == temp2->ID\_kompl) {

cost = temp2->Cost;

break;

}

kol = temp->Kolvo;

}

res += kol \* cost;

}

}

return res;

}

using namespace std;

void addCharToEnd(char \*&str, char z) {

int len = strlen(str);

char \*newStr = new char[len + 1 + 1];

strcpy(newStr, str);

newStr[len] = z;

newStr[len + 1] = 0;

delete[] str;

str = newStr;

}

void delCharToEnd(char \*&str) {

int len = strlen(str);

char \*newStr = new char[len];

memmove (newStr, str, len \* sizeof(char));

newStr[len - 1] = 0;

delete[] str;

str = newStr;

}

void inputStr(char \*&str) {

str = new char[1];

str[0] = 0;

do {

char c = \_getch();

if ((c == '\n') || (c == '\r')) {

break;

}

if ((c == '\b') && (strlen(str) > 0)) {

printf("\b \b");

delCharToEnd(str);

} else {

printf("%c", c);

addCharToEnd(str, c);

}

} while (true);

}

int main() {

cout << "Avtor - Firulina Maria, gruppa ISEbd-11"<< endl;

cout << "Uchot prodazh computernoy techniky"<< endl << endl;

List \*st = new List;

char \*value;

bool isExitFromCat = false;

char \*name;

int ID;

int IDCat;

int IDTov;

int cost;

do {

cout << "Select the table:" << endl;

cout << "1. Category" << endl;

cout << "2. Tovar" << endl;

cout << "3. Kompl" << endl;

cout << "4. Svaz" << endl;

cout << "5. Prodaza" << endl;

cout << "6. Output report" << endl;

cout << "7. Exit" << endl;

do {

inputStr(value);

cout << endl;

if ((myAtoi(value) >= 1) && (myAtoi(value) <= 7)) {

break;

}

cout << endl << "Try again: " << endl;

} while (true);

switch(myAtoi(value)) {

case 1:

cout << "You selected Category" << endl;

do {

isExitFromCat = false;

cout << "Select the action:" << endl;

cout << "1. Read from file" << endl;

cout << "2. Add element" << endl;

cout << "3. Change element" << endl;

cout << "4. Delete element" << endl;

cout << "5. Write to file " << endl;

cout << "6. Print screen " << endl;

cout << "7. Go back to main menu" << endl;

do {

inputStr(value);

cout << endl;

if ((myAtoi(value) >= 1) && (myAtoi(value) <= 7)) {

break;

}

cout << endl << "Try again: " << endl;

} while (true);

switch(myAtoi(value)) {

case 1:

cout << "You selected to read from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->inFileCat(name);

cout << endl;

break;

case 2:

cout << "You selected to add element" << endl;

int id;

srand(time(NULL));

id = rand() % 100 + 1;

cout << "Input Name" << endl;

inputStr(name);

cout << endl;

st->addCat(id, name);

break;

case 3:

cout << "You selected to change element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDCat(myAtoi(value))) {

cout << "Error of input id" << endl << "Try again: ";

} else {

break;

}

} while (true);

cout << "Input new name" << endl;

inputStr(name);

st->findByIDCat(myAtoi(value), name);

break;

case 4:

cout << "You selected to delete element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDCat(myAtoi(value))) {

cout << "Error of input id" << endl << "Try again: ";

} else {

break;

}

} while (true);

st->delByValueCateg(myAtoi(value));

break;

case 5:

cout << "You selected to write from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->outFileCat(name);

break;

case 6:

st->printCat();

break;

case 7:

isExitFromCat = true;

break;

}

} while (!isExitFromCat);

isExitFromCat = false;

cout << "You back" << endl;

break;

case 2:

cout << "You selected Tovar" << endl;

do {

isExitFromCat = false;

cout << "Select the action:" << endl;

cout << "1. Read from file" << endl;

cout << "2. Add element" << endl;

cout << "3. Change element" << endl;

cout << "4. Delete element" << endl;

cout << "5. Write to file " << endl;

cout << "6. Print screen " << endl;

cout << "7. Go back to main menu" << endl;

do {

inputStr(value);

cout << endl;

if ((myAtoi(value) >= 1) && (myAtoi(value) <= 7)) {

break;

}

cout << endl << "Try again: " << endl;

} while (true);

switch(myAtoi(value)) {

case 1:

cout << "You selected to read from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->inFileTov(name);

cout << endl;

break;

case 2:

cout << "You selected to add element" << endl;

srand(time(NULL));

ID = rand() % 100 + 1;

cout << "Input Name" << endl;

inputStr(name);

cout << endl;

cout << "Input category" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDCat(myAtoi(value))) {

cout << "Error of input category" << endl << "Try again: ";

} else {

break;

}

} while (true);

IDCat = myAtoi(value);

cout << "Select type" << endl;

cout << "1. Your device" << endl;

cout << "2. Foreign device" << endl;

do {

inputStr(value);

cout << endl;

if ((myAtoi(value) >= 1) && (myAtoi(value) <= 2)) {

break;

}

cout << endl << "Try again: " << endl;

} while (true);

bool type;

switch(myAtoi(value)) {

case 1:

type = false;

cost = st->findCostTov(ID);

break;

case 2:

type = true;

cout << "Input cost" << endl;

do {

inputStr(value);

cout << endl;

if (myAtoi(value) == 0) {

cout << "Error of input cost" << endl << "Try again: ";

} else {

break;

}

} while (true);

cost = myAtoi(value);

break;

}

st->addTov(ID, name, cost, IDCat, type);

break;

case 3:

cout << "You selected to change element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDTov(myAtoi(value))) {

cout << "Error of input id tovar" << endl << "Try again: ";

} else {

break;

}

} while (true);

ID = myAtoi(value);

cout << "Input new name" << endl;

inputStr(name);

cout << "Input new category" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDCat(myAtoi(value))) {

cout << "Error of input category" << endl << "Try again: ";

} else {

break;

}

} while (true);

IDCat = myAtoi(value);

cout << "Select new type" << endl;

cout << "1. Your device" << endl;

cout << "2. Foreign device" << endl;

do {

inputStr(value);

cout << endl;

if ((myAtoi(value) >= 1) && (myAtoi(value) <= 2)) {

break;

}

cout << endl << "Try again: " << endl;

} while (true);

switch(myAtoi(value)) {

case 1:

type = false;

cost = st->findCostTov(ID);

break;

case 2:

type = true;

cout << "Input new cost" << endl;

do {

inputStr(value);

cout << endl;

if (myAtoi(value) == 0) {

cout << "Error of input cost" << endl << "Try again: ";

} else {

break;

}

} while (true);

cost = myAtoi(value);

break;

}

st->findByIDTov(ID, name, cost, IDCat, type);

break;

case 4:

cout << "You selected to delete element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDTov(myAtoi(value))) {

cout << "Error of input id" << endl << "Try again: ";

} else {

break;

}

} while (true);

st->delByValueTov(myAtoi(value));

break;

case 5:

cout << "You selected to write from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->outFileTov(name);

break;

case 6:

st->printTov();

break;

case 7:

isExitFromCat = true;

break;

}

} while (!isExitFromCat);

isExitFromCat = false;

cout << "You back" << endl;

break;

case 3:

cout << "You selected Kompl" << endl;

do {

isExitFromCat = false;

cout << "Select the action:" << endl;

cout << "1. Read from file" << endl;

cout << "2. Add element" << endl;

cout << "3. Change element" << endl;

cout << "4. Delete element" << endl;

cout << "5. Write to file " << endl;

cout << "6. Print screen " << endl;

cout << "7. Go back to main menu" << endl;

do {

inputStr(value);

cout << endl;

if ((myAtoi(value) >= 1) && (myAtoi(value) <= 7)) {

break;

}

cout << endl << "Try again: " << endl;

} while (true);

switch(myAtoi(value)) {

case 1:

cout << "You selected to read from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->inFileKompl(name);

cout << endl;

break;

case 2:

cout << "You selected to add element" << endl;

srand(time(NULL));

ID = rand() % 100 + 1;

cout << "Input Name" << endl;

inputStr(name);

cout << endl;

cout << "Input category" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDCat(myAtoi(value))) {

cout << "Error of input category" << endl << "Try again: ";

} else {

break;

}

} while (true);

IDCat = myAtoi(value);

cout << "Input cost" << endl;

do {

inputStr(value);

cout << endl;

if (myAtoi(value) == 0) {

cout << "Error of input cost" << endl << "Try again: ";

} else {

break;

}

} while (true);

cost = myAtoi(value);

st->addKom(ID, name, cost, IDCat);

break;

case 3:

cout << "You selected to change element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDKompl(myAtoi(value))) {

cout << "Error of input id tovar" << endl << "Try again: ";

} else {

break;

}

} while (true);

ID = myAtoi(value);

cout << "Input new name" << endl;

inputStr(name);

cout << endl << "Input new category" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDCat(myAtoi(value))) {

cout << "Error of input category" << endl << "Try again: ";

} else {

break;

}

} while (true);

IDCat = myAtoi(value);

cout << "Input new cost" << endl;

do {

inputStr(value);

cout << endl;

if (myAtoi(value) == 0) {

cout << "Error of input cost" << endl << "Try again: ";

} else {

break;

}

} while (true);

cost = myAtoi(value);

st->findByIDKom(ID, name, cost, IDCat);

break;

case 4:

cout << "You selected to delete element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDKompl(myAtoi(value))) {

cout << "Error of input id" << endl << "Try again: ";

} else {

break;

}

} while (true);

st->delByValueKompl(myAtoi(value));

break;

case 5:

cout << "You selected to write from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->outFileKompl(name);

break;

case 6:

st->printKompl();

break;

case 7:

isExitFromCat = true;

break;

}

} while (!isExitFromCat);

isExitFromCat = false;

cout << "You back" << endl;

break;

case 4:

cout << "You selected Svaz" << endl;

do {

isExitFromCat = false;

cout << "Select the action:" << endl;

cout << "1. Read from file" << endl;

cout << "2. Add element" << endl;

cout << "3. Change element" << endl;

cout << "4. Delete element" << endl;

cout << "5. Write to file " << endl;

cout << "6. Print screen " << endl;

cout << "7. Go back to main menu" << endl;

do {

inputStr(value);

cout << endl;

if ((myAtoi(value) >= 1) && (myAtoi(value) <= 7)) {

break;

}

cout << endl << "Try again: " << endl;

} while (true);

switch(myAtoi(value)) {

case 1:

cout << "You selected to read from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->inFileSvaz(name);

cout << endl;

break;

case 2:

cout << "You selected to add element" << endl;

srand(time(NULL));

ID = rand() % 100 + 1;

cout << "Input tovar" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDTov(myAtoi(value))) {

cout << "Error of input tovar" << endl << "Try again: ";

} else {

break;

}

} while (true);

IDTov = myAtoi(value);

cout << "Input kompl" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDKompl(myAtoi(value))) {

cout << "Error of input kompl" << endl << "Try again: ";

} else {

break;

}

} while (true);

IDCat = myAtoi(value);

cout << "Input kolvo" << endl;

do {

inputStr(value);

cout << endl;

if (myAtoi(value) == 0) {

cout << "Error of input kolvo" << endl << "Try again: ";

} else {

break;

}

} while (true);

cost = myAtoi(value);

st->addSvaz(ID, IDTov, IDCat, cost);

break;

case 3:

cout << "You selected to change element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDSvaz(myAtoi(value))) {

cout << "Error of input id svaz" << endl << "Try again: ";

} else {

break;

}

} while (true);

ID = myAtoi(value);

cout << "Input new kolvo" << endl;

do {

inputStr(value);

cout << endl;

if (myAtoi(value) == 0) {

cout << "Error of input kolvo" << endl << "Try again: ";

} else {

break;

}

} while (true);

cost = myAtoi(value);

st->findByIDSvaz(ID, cost);

break;

case 4:

cout << "You selected to delete element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDSvaz(myAtoi(value))) {

cout << "Error of input id" << endl << "Try again: ";

} else {

break;

}

} while (true);

st->delByValueSvaz(myAtoi(value));

break;

case 5:

cout << "You selected to write from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->outFileSvaz(name);

break;

case 6:

st->printSvaz();

break;

case 7:

isExitFromCat = true;

break;

}

} while (!isExitFromCat);

isExitFromCat = false;

cout << "You back" << endl;

break;

case 5:

cout << "You selected Prodaza" << endl;

do {

isExitFromCat = false;

cout << "Select the action:" << endl;

cout << "1. Read from file" << endl;

cout << "2. Add element" << endl;

cout << "3. Change element" << endl;

cout << "4. Delete element" << endl;

cout << "5. Write to file " << endl;

cout << "6. Print screen " << endl;

cout << "7. Go back to main menu" << endl;

do {

inputStr(value);

cout << endl;

if ((myAtoi(value) >= 1) && (myAtoi(value) <= 7)) {

break;

}

cout << endl << "Try again: " << endl;

} while (true);

switch(myAtoi(value)) {

case 1:

cout << "You selected to read from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->inFileProd(name);

cout << endl;

break;

case 2:

cout << "You selected to add element" << endl;

srand(time(NULL));

ID = rand() % 100 + 1;

cout << "Input tovar" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDTov(myAtoi(value))) {

cout << "Error of input tovar" << endl << "Try again: ";

} else {

break;

}

} while (true);

IDTov = myAtoi(value);

cout << "Input kolvo" << endl;

do {

inputStr(value);

cout << endl;

if (myAtoi(value) == 0) {

cout << "Error of input kolvo" << endl << "Try again: ";

} else {

break;

}

} while (true);

cost = myAtoi(value);

st->addProd(ID, IDTov, cost, st->inputDate());

break;

case 3:

cout << "You selected to change element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDSvaz(myAtoi(value))) {

cout << "Error of input id svaz" << endl << "Try again: ";

} else {

break;

}

} while (true);

ID = myAtoi(value);

cout << "Input new tovar" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDTov(myAtoi(value))) {

cout << "Error of input tovar" << endl << "Try again: ";

} else {

break;

}

} while (true);

IDTov = myAtoi(value);

cout << "Input new kolvo" << endl;

do {

inputStr(value);

cout << endl;

if (myAtoi(value) == 0) {

cout << "Error of input kolvo" << endl << "Try again: ";

} else {

break;

}

} while (true);

cost = myAtoi(value);

cout << "Input new date" << endl;

//Date date = st->inputDate();

st->findByIDProd(ID, IDTov, cost, st->inputDate());

break;

case 4:

cout << "You selected to delete element" << endl;

cout << "Input id" << endl;

do {

inputStr(value);

cout << endl;

if (!st->existIDProd(myAtoi(value))) {

cout << "Error of input id" << endl << "Try again: ";

} else {

break;

}

} while (true);

st->delByValueProd(myAtoi(value));

break;

case 5:

cout << "You selected to write from file" << endl;

cout << "Input path to file" << endl;

inputStr(name);

st->outFileProd(name);

break;

case 6:

st->printProd();

break;

case 7:

isExitFromCat = true;

break;

}

} while (!isExitFromCat);

isExitFromCat = false;

cout << "You back" << endl;

break;

case 6:

cout << "You selected Output report" << endl;

do {

isExitFromCat = false;

cout << "Select report:" << endl;

cout << "1. Calculate summ for period" << endl;

cout << "2. Calculate summ for categories" << endl;

cout << "3. Calculate kolvo" << endl;

cout << "4. Go back to main menu" << endl;

do {

inputStr(value);

cout << endl;

if ((myAtoi(value) >= 1) && (myAtoi(value) <= 7)) {

break;

}

cout << endl << "Try again: " << endl;

} while (true);

switch(myAtoi(value)) {

case 1:

cout << "You selected calculate summ for period" << endl;

cout << "Your result: " << st->summProdPer() << endl;

break;

case 2:

cout << "You selected calculate summ for categories" << endl;

st->summProdCat();

break;

case 3:

cout << "You selected calculate kolvo" << endl;

cout << "Your result: " << st->countProdPer() << endl;

break;

case 4:

isExitFromCat = true;

break;

}

} while (!isExitFromCat);

isExitFromCat = false;

cout << "You back" << endl;

break;

case 7:

isExitFromCat = true;

break;

}

} while (!isExitFromCat);