#include <iostream>

#include <fstream>

#include <iomanip>

#include <algorithm>

#include <limits>

#include <string>

#include <vector>

#include <Windows.h>

using namespace std;

unsigned max\_barcode\_lenght;

unsigned max\_name\_lenght;

unsigned max\_unit\_lenght;

unsigned max\_file\_lenght;

#if defined(max)

#undef max

#endif

typedef struct

{

string barcode;

string name;

int size;

string unit;

double price;

int amount;

} Product;

bool main\_menu();

void add\_item();

void edit\_item();

void delete\_item();

void search\_item(bool filter);

void sort\_items();

void open\_items();

void save\_items();

int get\_int(int min, int max);

float get\_float(float min, float max);

void get\_str(string &str, unsigned &max);

void get\_item(unsigned i);

void get\_list();

void get\_max\_lenght();

void read\_string(ifstream &fin, string &str);

void write\_string(ofstream &fout, string &str);

vector<Product> prod;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

while(!main\_menu());

return 0;

}

bool main\_menu()

{

cout << "ОСНОВНОЕ МЕНЮ" << endl;

cout << "1. Создать запись" << endl;

cout << "2. Изменить запись" << endl;

cout << "3. Удалить запись" << endl;

cout << "4. Поиск записей" << endl;

cout << "5. Отфильтровать записи" << endl;

cout << "6. Сортировать таблицу" << endl;

cout << "7. Открыть" << endl;

cout << "8. Сохранить" << endl;

cout << "9. Выйти" << endl << endl;

cout << "Выберите пункт меню: ";

switch (get\_int(1, 9))

{

case 1: { add\_item(); break; }

case 2: { edit\_item(); break; }

case 3: { delete\_item(); break; }

case 4: { search\_item(false); break; }

case 5: { search\_item(true); break; }

case 6: { sort\_items(); break; }

case 7: { open\_items(); break; }

case 8: { save\_items(); break; }

case 9: { return true; }

}

return false;

}

void add\_item()

{

system("cls");

cout << "СОЗДАНИЕ ЗАПИСИ" << endl;

Product input;

cout << "Введите штрихкод: ";

get\_str(input.barcode, max\_barcode\_lenght);

cout << "Введите наименование: ";

get\_str(input.name, max\_name\_lenght);

cout << "Введите массу/объём: ";

input.size = get\_int(0, 0);

cout << "Введите единицу измерения: ";

get\_str(input.unit, max\_unit\_lenght);

cout << "Введите цену: ";

input.price = get\_float(0, 0);

cout << "Введите количество: ";

input.amount = get\_int(0, 0);

prod.push\_back(input);

get\_max\_lenght();

system("cls");

cout << "Запись добавлена!" << endl << endl;

}

void edit\_item()

{

system("cls");

if (prod.empty())

{

cout << "Список пуст!" << endl << endl;

return;

}

cout << "РЕДАКТИРОВАНИЕ ЗАПИСИ" << endl;

get\_list();

cout << "Введите номер редактируемой записи (1-" << prod.size() << ", 0 - отмена): ";

int item = get\_int(0, prod.size());

system("cls");

if (item > 0)

{

Product input;

cout << "Введите штрихкод: ";

get\_str(input.barcode, max\_barcode\_lenght);

cout << "Введите наименование: ";

get\_str(input.name, max\_name\_lenght);

cout << "Введите массу/объём: ";

input.size = get\_int(0, 0);

cout << "Введите единицу измерения: ";

get\_str(input.unit, max\_unit\_lenght);

cout << "Введите цену: ";

input.price = get\_float(0, 0);

cout << "Введите количество: ";

input.amount = get\_int(0, 0);

prod[item - 1] = input;

get\_max\_lenght();

system("cls");

cout << "Запись отредактирована!" << endl << endl;

}

}

void delete\_item()

{

system("cls");

if (prod.empty())

{

cout << "Список пуст!" << endl << endl;

return;

}

cout << "УДАЛЕНИЕ ЗАПИСИ" << endl;

get\_list();

cout << "Введите номер удаляемой записи (1-" << prod.size() << ", 0 - отмена): ";

int item = get\_int(0, prod.size());

system("cls");

if (item > 0)

{

prod.erase(prod.begin() + item - 1);

get\_max\_lenght();

cout << "Запись удалена!" << endl << endl;

}

}

void search\_item(bool filter)

{

system("cls");

if (prod.empty())

{

cout << "Список пуст!" << endl << endl;

return;

}

cout << (filter ? "ФИЛЬТРАЦИЯ ЗАПИСЕЙ" : "ПОИСК ЗАПИСЕЙ") << endl;

cout << "1. Штрихкод" << endl;

cout << "2. Наименование" << endl;

cout << "3. Масса/объём" << endl;

cout << "4. Единица измерения" << endl;

cout << "5. Цена" << endl;

cout << "6. Количество" << endl;

cout << "0. Вернуться назад" << endl << endl;

cout << "Выберите столбец: ";

int column = get\_int(0, 6);

cout << "Введите значение: ";

switch (column)

{

case 1:

{

string input;

get\_str(input, max\_barcode\_lenght);

system("cls");

cout << "Результат:" << endl;

for (unsigned i = 0; i < prod.size(); i++)

if (filter ? prod[i].barcode.find(input) == string::npos : prod[i].barcode.find(input) != string::npos)

get\_item(i);

break;

}

case 2:

{

string input;

get\_str(input, max\_name\_lenght);

system("cls");

cout << "Результат:" << endl;

for (unsigned i = 0; i < prod.size(); i++)

if (filter ? prod[i].name.find(input) == string::npos : prod[i].name.find(input) != string::npos)

get\_item(i);

break;

}

case 3:

{

int input = get\_int(0, 0);

system("cls");

cout << "Результат:" << endl;

for (unsigned i = 0; i < prod.size(); i++)

if (filter ? prod[i].size != input : prod[i].size == input)

get\_item(i);

break;

}

case 4:

{

string input;

get\_str(input, max\_unit\_lenght);

system("cls");

cout << "Результат:" << endl;

for (unsigned i = 0; i < prod.size(); i++)

if (filter ? prod[i].unit.find(input) == string::npos : prod[i].unit.find(input) != string::npos)

get\_item(i);

break;

}

case 5:

{

double input = get\_float(0, 0);

system("cls");

cout << "Результат:" << endl;

for (unsigned i = 0; i < prod.size(); i++)

if (filter ? fabs(prod[i].price - input) > 0.001 : fabs(prod[i].price - input) <= 0.001)

get\_item(i);

break;

}

case 6:

{

int input = get\_int(0, 0);

system("cls");

cout << "Результат:" << endl;

for (unsigned i = 0; i < prod.size(); i++)

if (filter ? prod[i].amount != input : prod[i].amount == input)

get\_item(i);

break;

}

case 0:

{

system("cls");

return;

}

}

cout << endl;

}

void sort\_items()

{

system("cls");

if (prod.empty())

{

cout << "Список пуст!" << endl << endl;

return;

}

cout << "СОРТИРОВКА ЗАПИСЕЙ" << endl;

cout << "1. Штрихкод" << endl;

cout << "2. Наименование" << endl;

cout << "3. Масса/объём" << endl;

cout << "4. Единица измерения" << endl;

cout << "5. Цена" << endl;

cout << "6. Количество" << endl;

cout << "0. Вернуться назад" << endl << endl;

cout << "Выберите столбец для сортировки: ";

int input = get\_int(0, 6);

if (!input)

{

system("cls");

return;

}

cout << "Выберите направление для сортировки (1 - прямое, 0 - обратное): ";

int dir = get\_int(0, 1);

switch (input)

{

case 1: { sort(prod.begin(), prod.end(),

[dir](Product a, Product b) { return (dir) ? a.barcode < b.barcode : a.barcode > b.barcode; }); break; }

case 2: { sort(prod.begin(), prod.end(),

[dir](Product a, Product b) { return (dir) ? a.name < b.name : a.name > b.name; }); break; }

case 3: { sort(prod.begin(), prod.end(),

[dir](Product a, Product b) { return (dir) ? a.size < b.size : a.size > b.size; }); break; }

case 4: { sort(prod.begin(), prod.end(),

[dir](Product a, Product b) { return (dir) ? a.unit < b.unit : a.unit > b.unit; }); break; }

case 5: { sort(prod.begin(), prod.end(),

[dir](Product a, Product b) { return (dir) ? a.price < b.price : a.price > b.price; }); break; }

case 6: { sort(prod.begin(), prod.end(),

[dir](Product a, Product b) { return (dir) ? a.amount < b.amount : a.amount > b.amount; }); break; }

}

system("cls");

cout << "Список отсортирован!" << endl << endl;

get\_list();

}

void open\_items()

{

system("cls");

cout << "ОТКРЫТИЕ ФАЙЛА" << endl;

string input;

cout << "Введите имя файла: ";

get\_str(input, max\_file\_lenght);

system("cls");

ifstream fin(input + ".bin", ios::in | ios::binary);

if (!fin.is\_open())

{

cout << "Произошла ошибка при открытии файла!" << endl << endl;

return;

}

char check[10];

fin.read((char \*)&check, 10);

if (strcmp(check, "prod\_list") != 0)

{

cout << "Файл не является поддерживаемой базой данных!" << endl << endl;

fin.close();

return;

}

prod.erase(prod.begin(), prod.end());

while (!fin.eof())

{

Product \*tmp = new Product;

read\_string(fin, tmp->barcode);

read\_string(fin, tmp->name);

fin.read((char \*)&tmp->size, sizeof(int));

read\_string(fin, tmp->unit);

fin.read((char \*)&tmp->price, sizeof(double));

fin.read((char \*)&tmp->amount, sizeof(int));

if(fin.good()) prod.push\_back(\*tmp);

delete tmp;

}

fin.close();

cout << "Файл открыт!" << endl << endl;

get\_max\_lenght();

get\_list();

}

void save\_items()

{

system("cls");

if (prod.empty())

{

cout << "Нет данных для сохранения!" << endl << endl;

return;

}

cout << "СОХРАНЕНИЕ ФАЙЛА" << endl;

string input;

cout << "Введите имя файла: ";

get\_str(input, max\_file\_lenght);

system("cls");

ofstream fout(input + ".bin", ios::out | ios::binary);

if (fout.is\_open())

{

char check[10] = "prod\_list";

fout.write((char \*)&check, 10);//запись проверочной строки

for (unsigned i = 0; i < prod.size(); i++)

{

write\_string(fout, prod[i].barcode);

write\_string(fout, prod[i].name);

fout.write((char \*)&prod[i].size, sizeof(int));

write\_string(fout, prod[i].unit);

fout.write((char \*)&prod[i].price, sizeof(double));

fout.write((char \*)&prod[i].amount, sizeof(int));

}

fout.close();

cout << "Файл сохранён!" << endl << endl;

}

else cout << "Произошла ошибка при сохранении файла!" << endl << endl;

}

int get\_int(int min, int max)

{

int input;

cin >> input;

while (cin.fail() || (input < min && min != max) || (input > max && max > min))

{//если min и max не нужны указать их равными, если не нужен только max указать его меньше чем min.

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Некорректный ввод! Попробуйте снова: ";

cin >> input;

}

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

return input;

}

float get\_float(float min, float max)

{

float input;

cin >> input;

while (cin.fail() || (input < min && min != max) || (input > max && max > min))

{//если min и max не нужны указать их равными, если не нужен только max указать его меньше чем min.

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Некорректный ввод! Попробуйте снова: ";

cin >> input;

}

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

return input;

}

void get\_str(string &str, unsigned &max)

{

cin >> str;

while (cin.fail())

{//если max не нужен указать равным нулю.

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Некорректный ввод! Попробуйте снова: ";

cin >> str;

}

if (str.size() > max && max >= 0) max = str.size();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

void get\_item(unsigned i)

{

cout << setw(4) << i + 1 << " | " <<

setw(max\_barcode\_lenght) << prod[i].barcode << " | " <<

setw(max\_name\_lenght) << prod[i].name << " | " <<

setw(10) << prod[i].size << " | " <<

setw(max\_unit\_lenght) << prod[i].unit << " | " <<

setw(10) << prod[i].price << " | " <<

setw(10) << prod[i].amount << endl;

}

void get\_list()

{

for (unsigned i = 0; i < prod.size(); i++)

{

cout << setw(4) << i + 1 << " | " <<

setw(max\_barcode\_lenght) << prod[i].barcode << " | " <<

setw(max\_name\_lenght) << prod[i].name << " | " <<

setw(10) << prod[i].size << " | " <<

setw(max\_unit\_lenght) << prod[i].unit << " | " <<

setw(10) << prod[i].price << " | " <<

setw(10) << prod[i].amount << endl;

}

cout << endl;

}

void get\_max\_lenght()

{

for (unsigned i = 0; i < prod.size(); i++)

{

if (prod[i].barcode.size() > max\_barcode\_lenght) max\_barcode\_lenght = prod[i].barcode.size();

if (prod[i].name.size() > max\_name\_lenght) max\_name\_lenght = prod[i].name.size();

if (prod[i].unit.size() > max\_unit\_lenght) max\_unit\_lenght = prod[i].unit.size();

}

}

void read\_string(ifstream &fin, string &str)

{

unsigned size;

fin.read((char \*)&size, sizeof(size));//читаем размер строки

if (!fin.good()) return;

char \*buffer = new char[size];//создаем буфер для char > string

fin.read((char \*)buffer, size);//читаем файл

if (fin.good()) str = buffer;//переписываем char > string

delete[] buffer;//удаляем буфер

}

void write\_string(ofstream &fout, string &str)

{

unsigned size = str.size() + 1;//определяем размер строки

fout.write((char \*)&size, sizeof(size));//записываем размер строки

fout.write((char \*)str.data(), size);//переводим в char и записываем

}