Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Уфимский государственный авиационный технический университет»

Кафедра ТК

**ОТЧЕТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

*(рассредоточенной)*

семестр 3

Выполнил: студент гр. ИВТ-221

Каррамов Т.И.

Проверила: доцент каф. ТК

Федорова Н.И.

Уфа 2020**Аннотация**

В ходе данной учебной (рассредоточенной) практики необходимо спроектировать необходимые классы и разработать программу на языке С++ для обработки данных.

**Содержание**

|  |  |
| --- | --- |
| Введение………………………………………………………………... | 4 |
| 1 Описание разработанных функций и глобальных переменных…... | 5 |
| 2 Схемы алгоритмов работы программы…………………….……… | 6 |
| 2.1 Схема алгоритма работы основной программы……..…...….…. | 6 |
| 2.2 Схемы алгоритмов работы подпрограмм.…………………...….. | 7 |
| 3 Тестирование программы…………………………………………… | 9 |
| 4 Листинг программы..……………………………………………...…. | 21 |
| Заключение…………………………………………..………………… | 31 |
| Список литературы…………………………………………………….. | 32 |

## **Введение**

Теоретическая цель учебной практики- проектирование классов С++ для обработки файловых данных.

Практическая цель учебной практики- приобретение практических навыков работы по программированию и разработка программы на языке С++ для обработки данных.

## **Описание разработанных функций и глобальных переменных**

В таблице 1­­ представлены разработанные функции и их краткое описание.

Таблица 1­­ ­– Описание функций

|  |  |
| --- | --- |
| void inputDb(bool fillRand) | Загрузка информации из существующей базы данных (вывод из файла) |
| void outputConsoleDb() | Загрузка информации из существующей базы данных |
| void addRowDb() | Добавление новой записи в базу данных |
| void deleteRowDb() | Удаление записи из базы данных |
| void editRowDb() | Редактирование записи в базе данных |
| void sortNumDb() | Сортировка записей по числовому полю в порядке возрастания |
| void sortStringDb() | Сортировка записей по строковому полю в алфавитном порядке |
| void outputFileDb() | Сохранение результатов обработки (запись в файл) |
| void searchDb() | Поиск в базе данных по наименованию продукта с выводом на экран найденных записей или сообщения о неуспешном поиске |
| void createList() | Формирование перечня |
| void outputConsoleList() | Просмотр (вывод на экран) перечня |
| void sortNumList() | Сортировка перечня по числовому полю в порядке убывания |
| void sortStringList() | Сортировка перечня по строковому полю в алфавитном порядке |
| void outputFileList() | Сохранение перечня в файле |

## **Схемы алгоритмов работы программы**

* 1. **Схема алгоритма работы основной программы**

На рисунке 1 представлена схема работы основной программы.



Рисунок 1 – Блок-схема основной программы.

**2.2 Схемы алгоритмов работы подпрограмм**

На рисунках 2,3 показаны схемы работы подпрограмм.



Рисунок 2 – Блок-схемы подпрограмм (функции добавления, удаления, вывода на экран)



Рисунок 3 – Блок-схемы подпрограмм (функции редактирования, ввода информации из файла)

## **Тестирование программы**

На рисунках 4-25 приведены результаты тестирования программы.

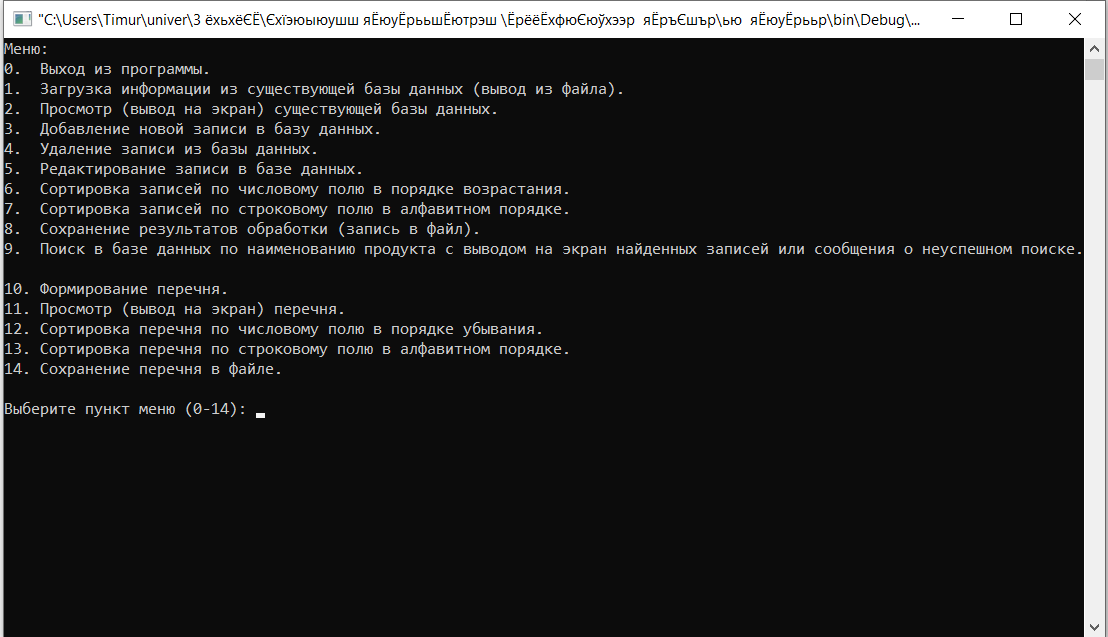


Рисунок 4 – Основное меню

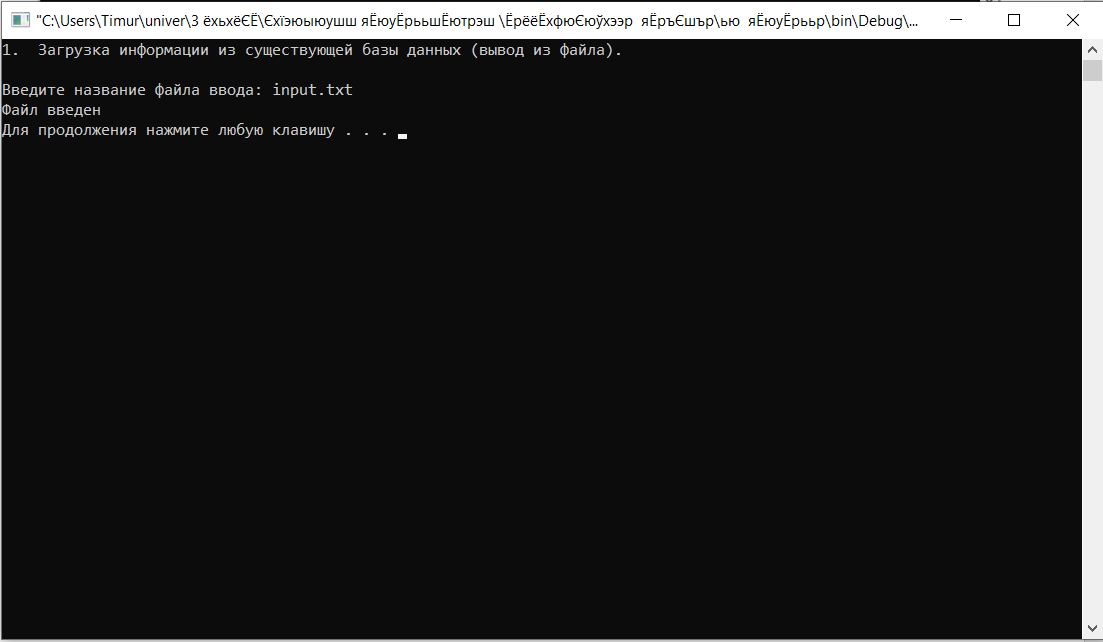


Рисунок 5 – Ввод данных

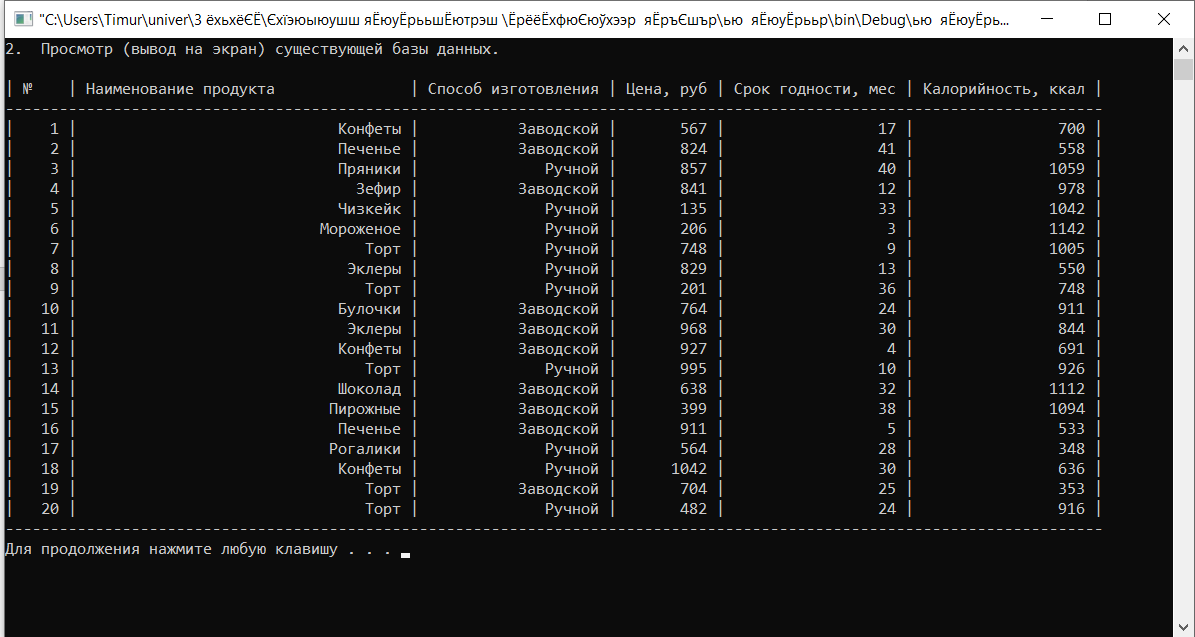


Рисунок 6 – Вывод таблицы на экран

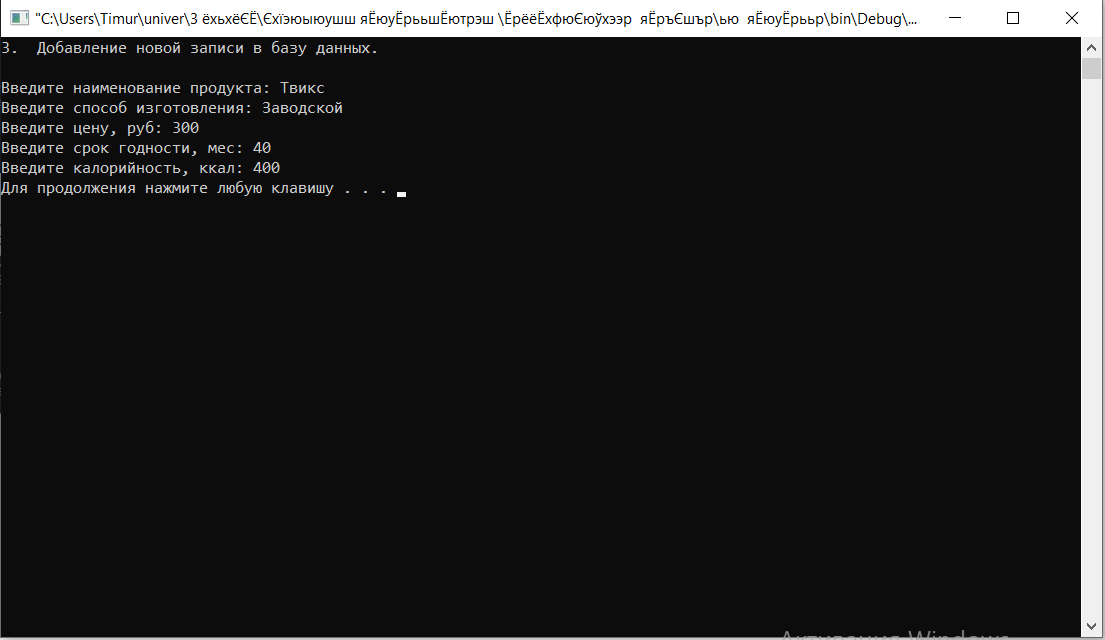


Рисунок 7 – Добавление записи в список

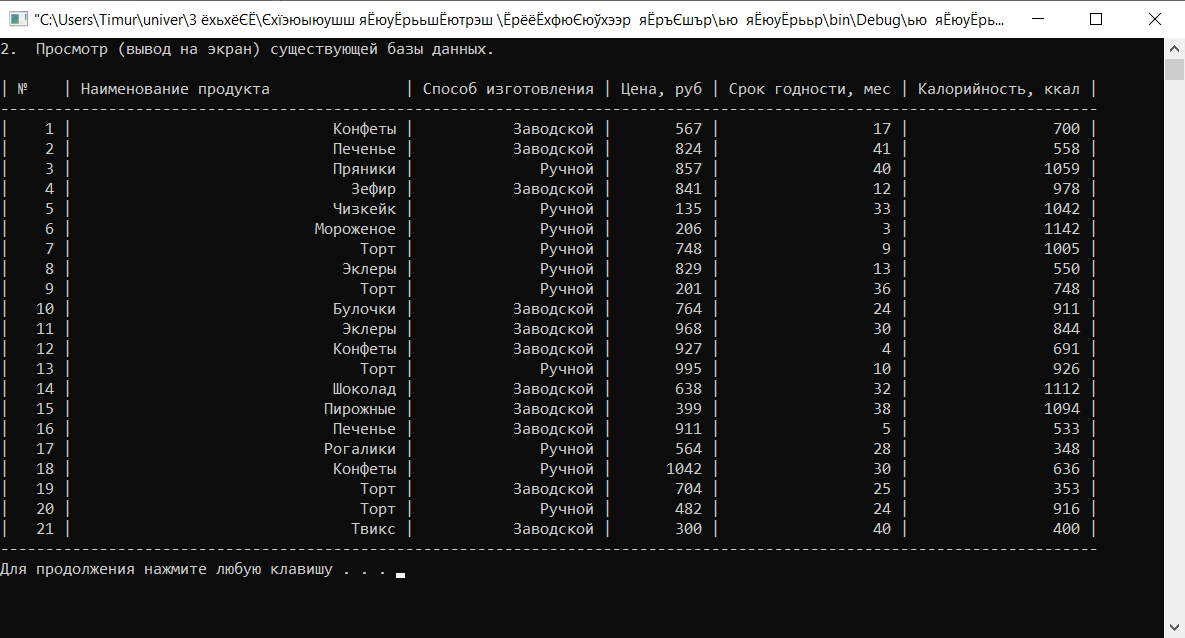
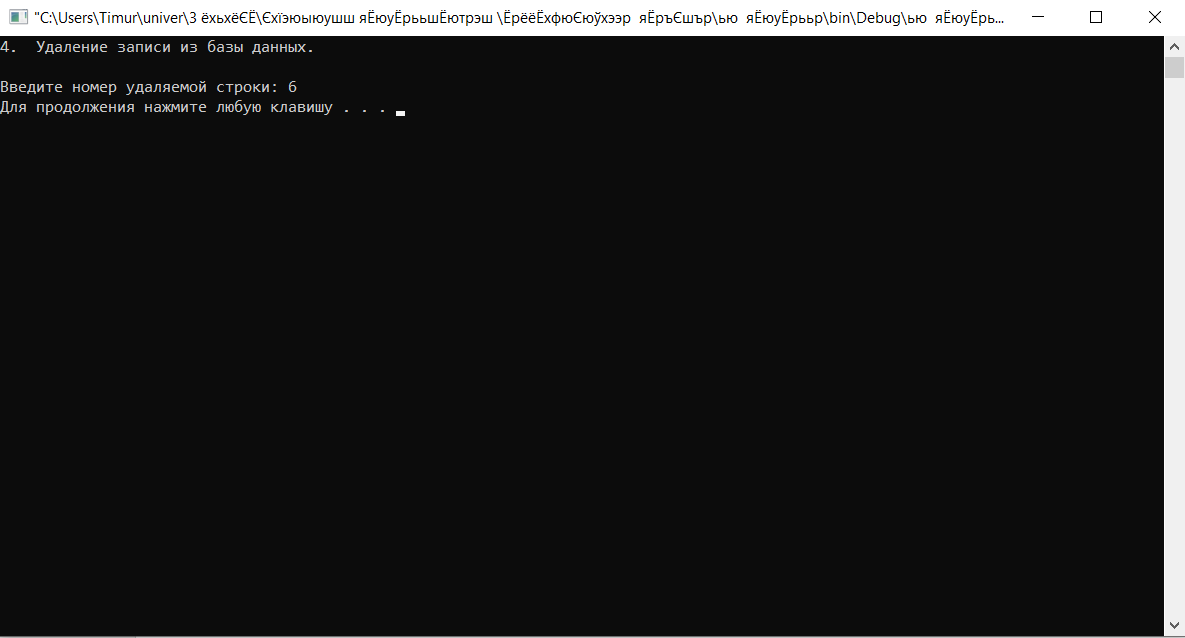


Рисунок 8 – Вывод таблицы на экран с добавленной записью

Рисунок 9 – Удаление записи из списка

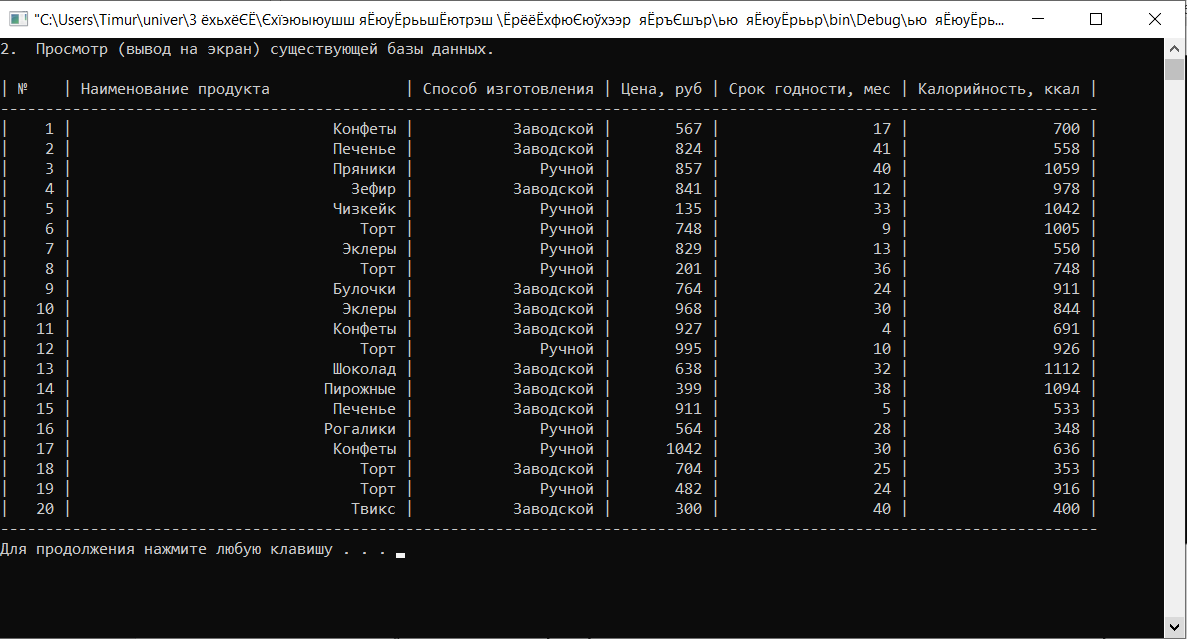


Рисунок 8 – Вывод таблицы на экран с удаленной записью

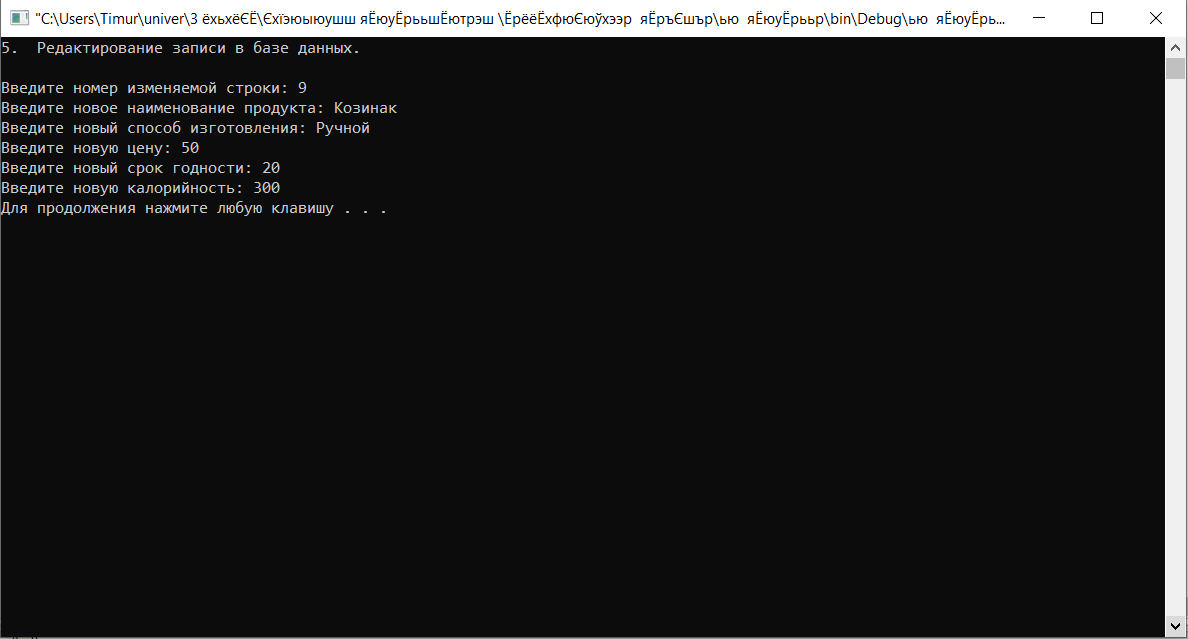


Рисунок 9 – Редактирование записи в базе данных

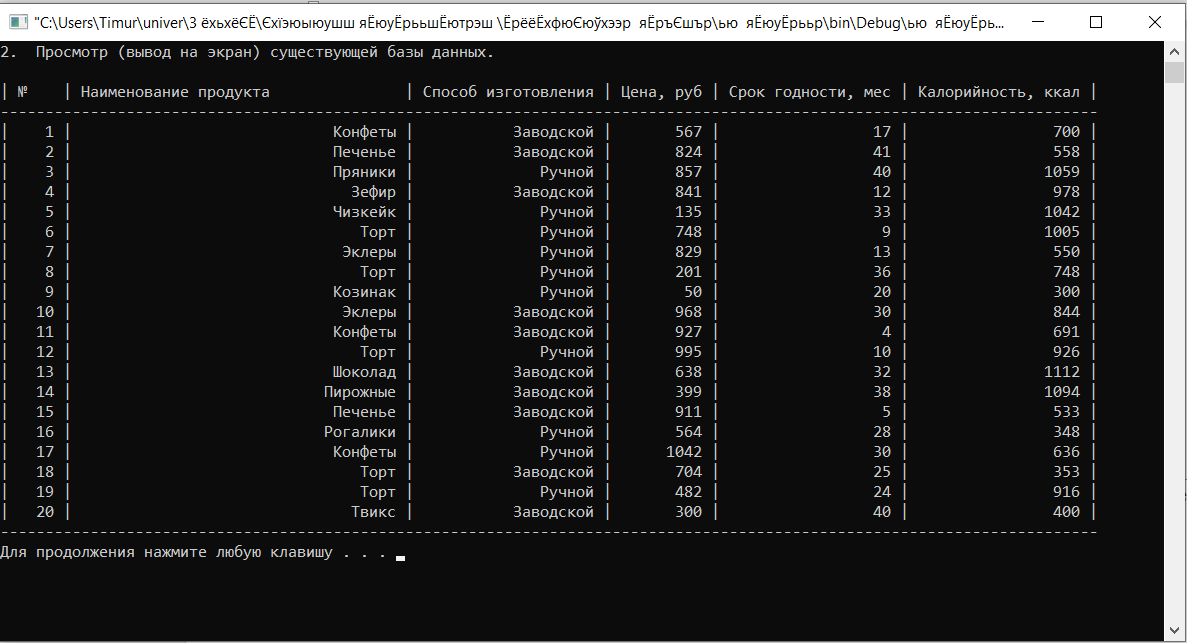


Рисунок 10 – Вывод таблицы на экран с редактируемой записью

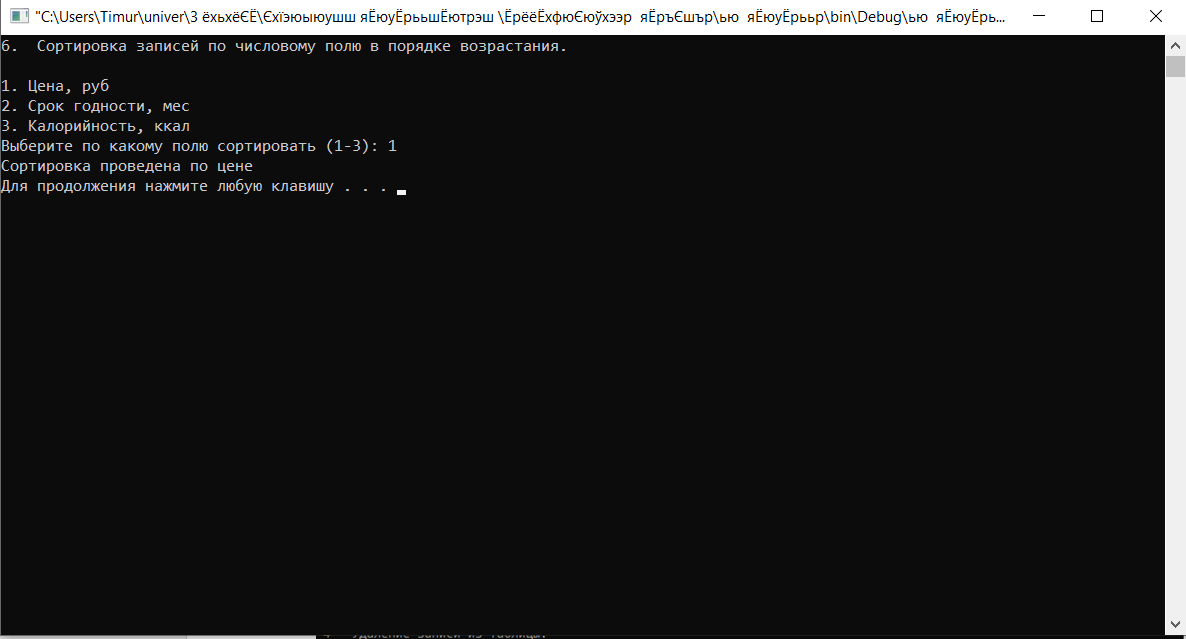


Рисунок 11 – Сортировка записей по цене в порядке возрастания

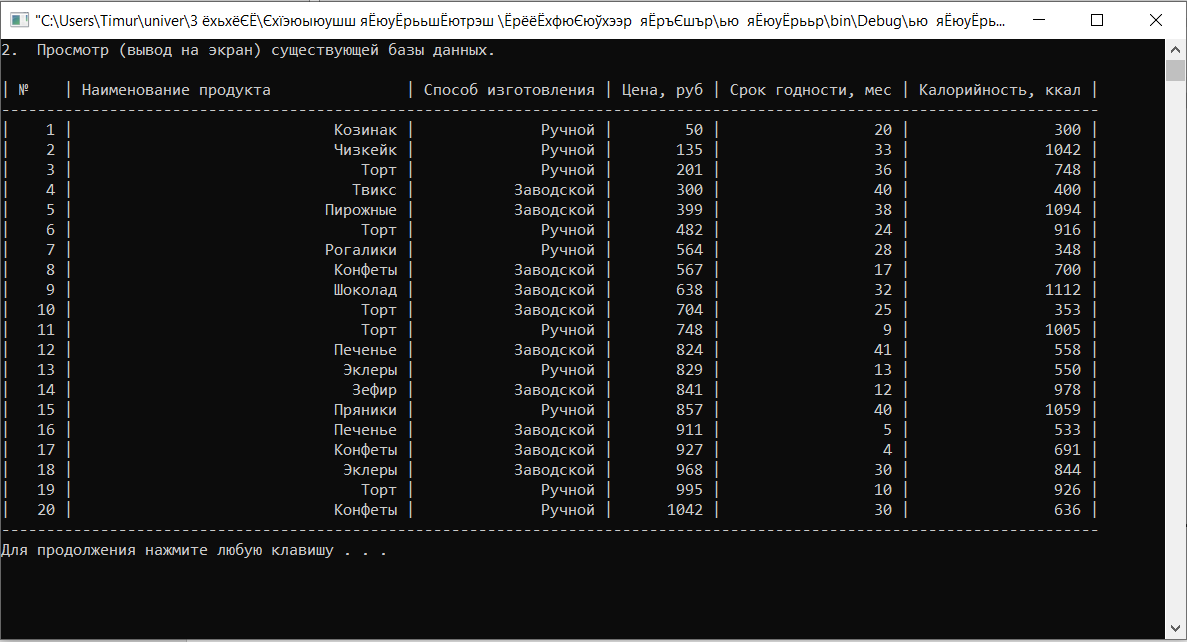


Рисунок 12 – Вывод отсортированной базы данных на экран

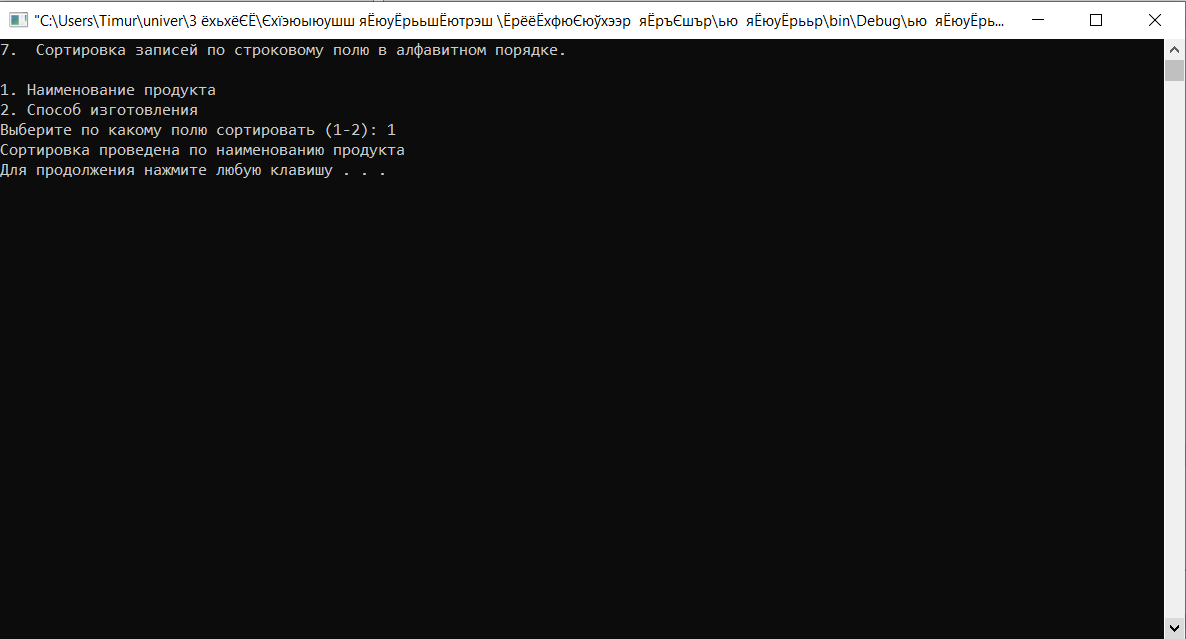


Рисунок 13 – Сортировка записей по наименованию продукта в алфавитном порядке

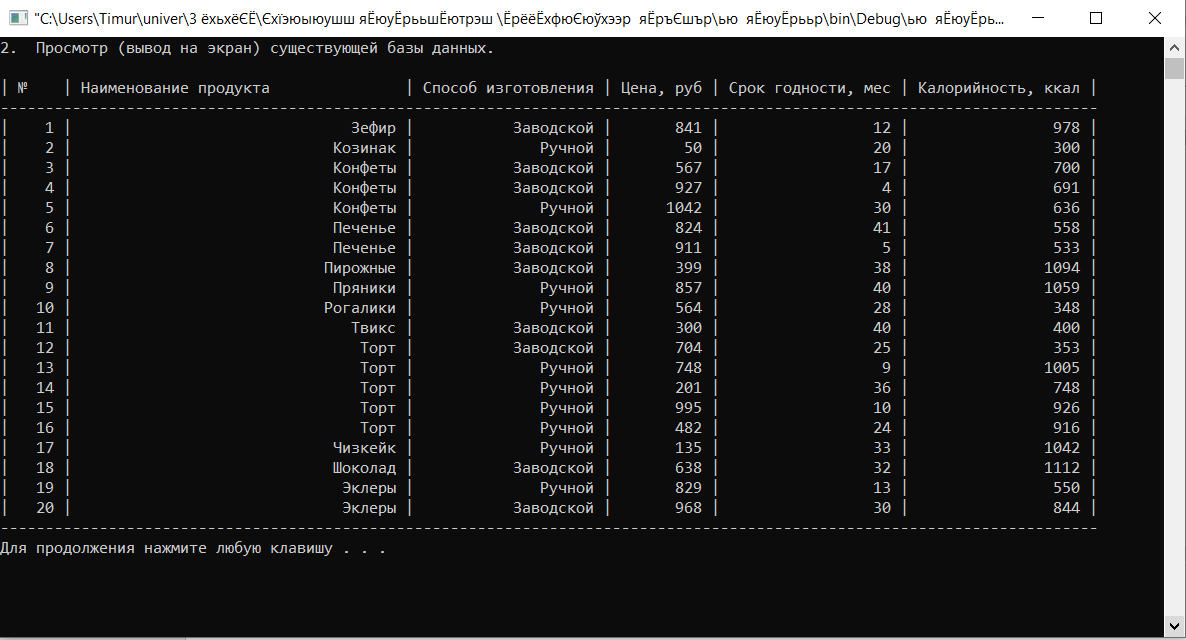


Рисунок 14 – Вывод отсортированной базы данных на экран

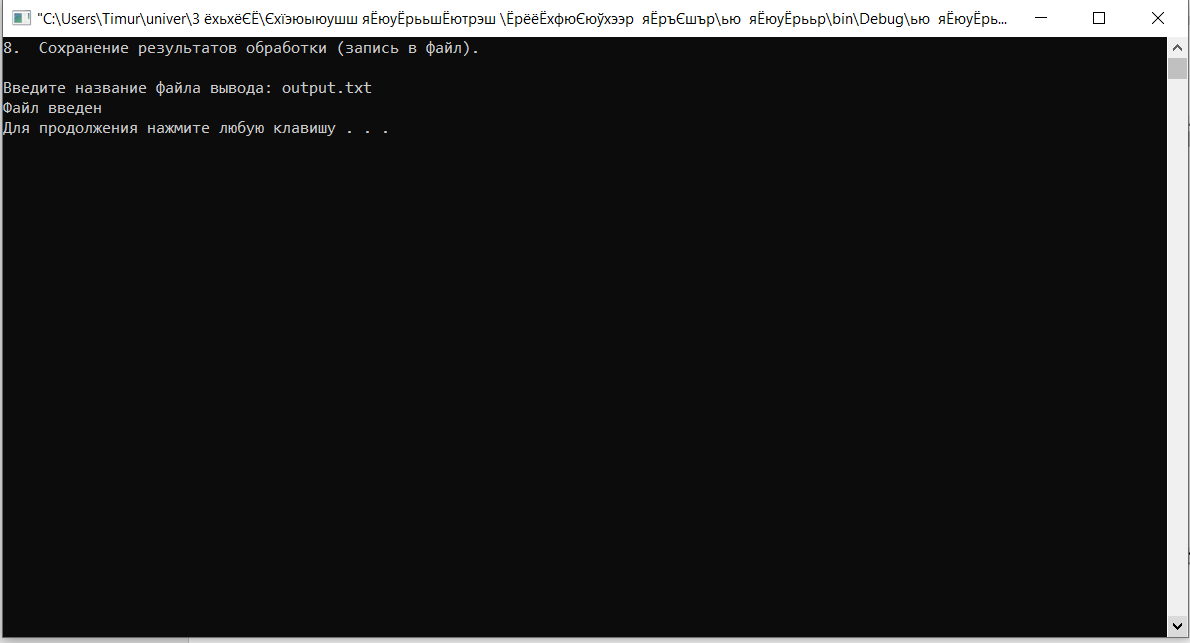


Рисунок 15 – Сохранение результатов обработки (запись в файл)

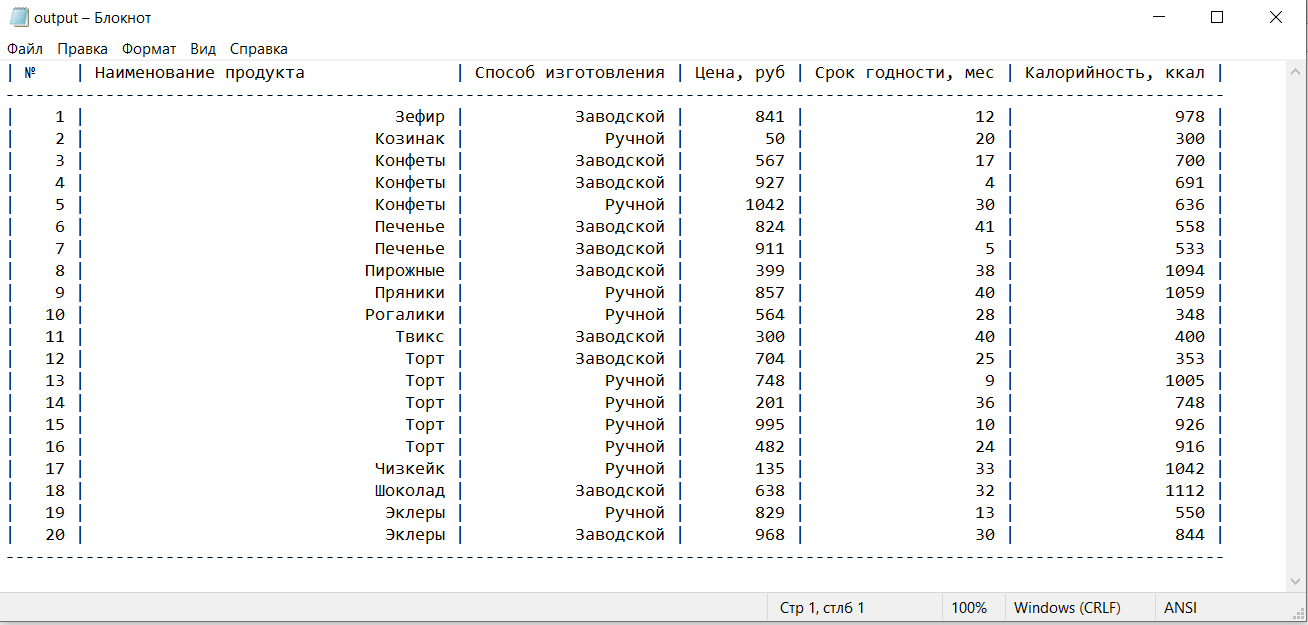


Рисунок 16 – Сохранение результатов обработки (запись в файл)

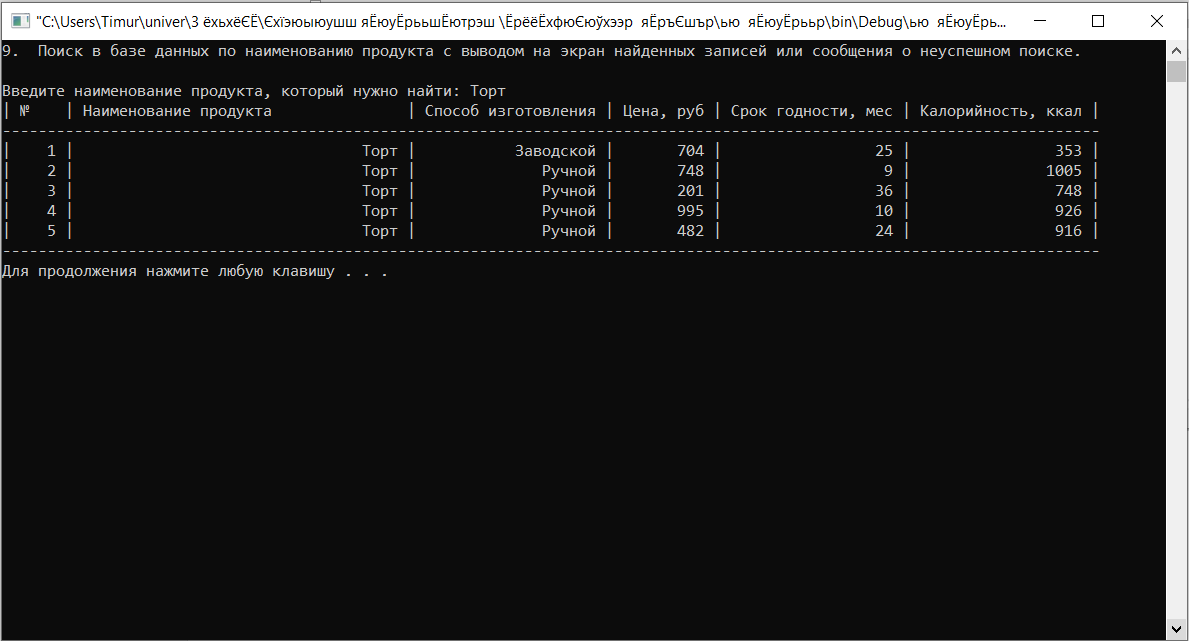


Рисунок 17 – Поиск в базе данных по наименованию продукта с выводом на экран найденных записей или сообщения о неуспешном поиске

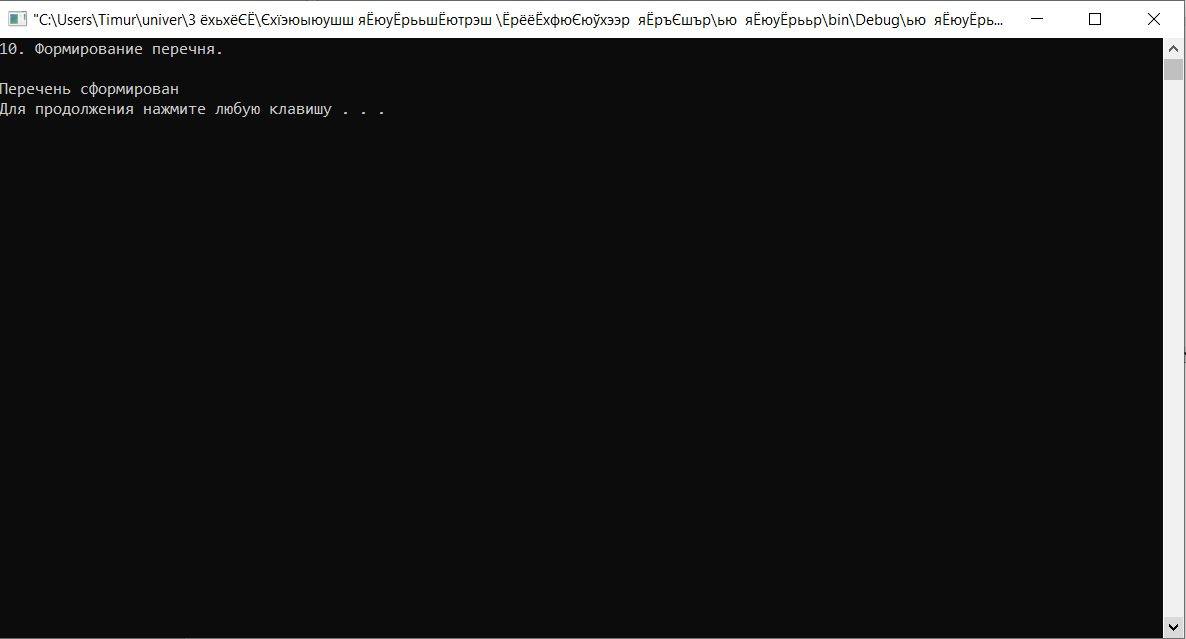


Рисунок 18 – Формирование перечня

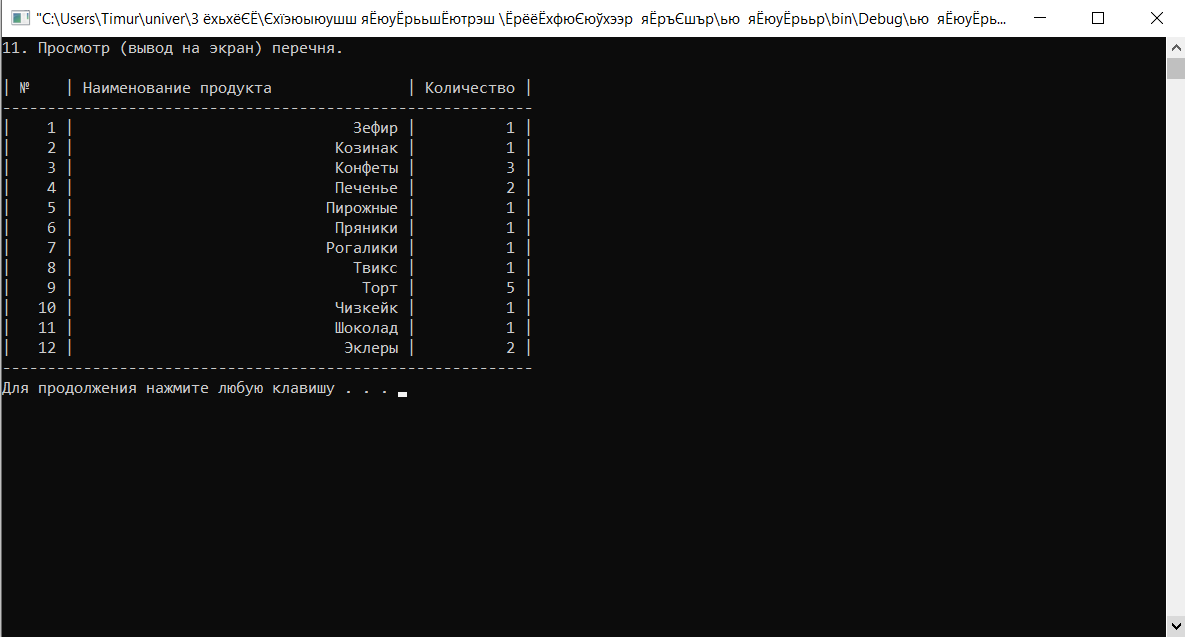


Рисунок 19 – Просмотр (вывод на экран) перечня

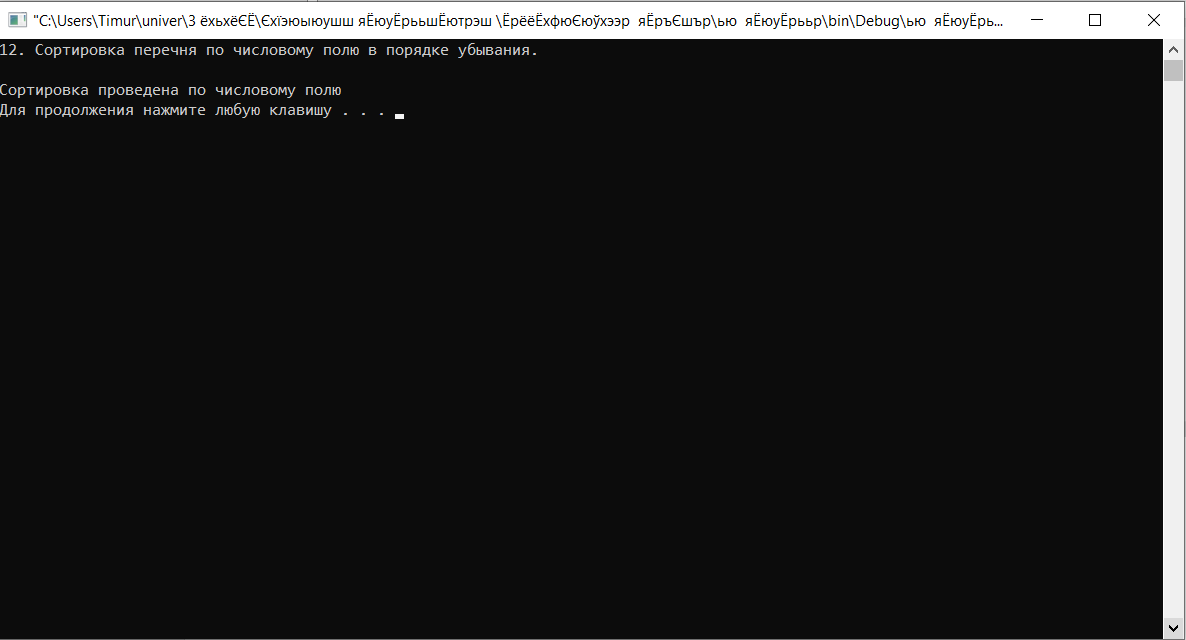


Рисунок 20 – Сортировка перечня по числовому полю в порядке убывания

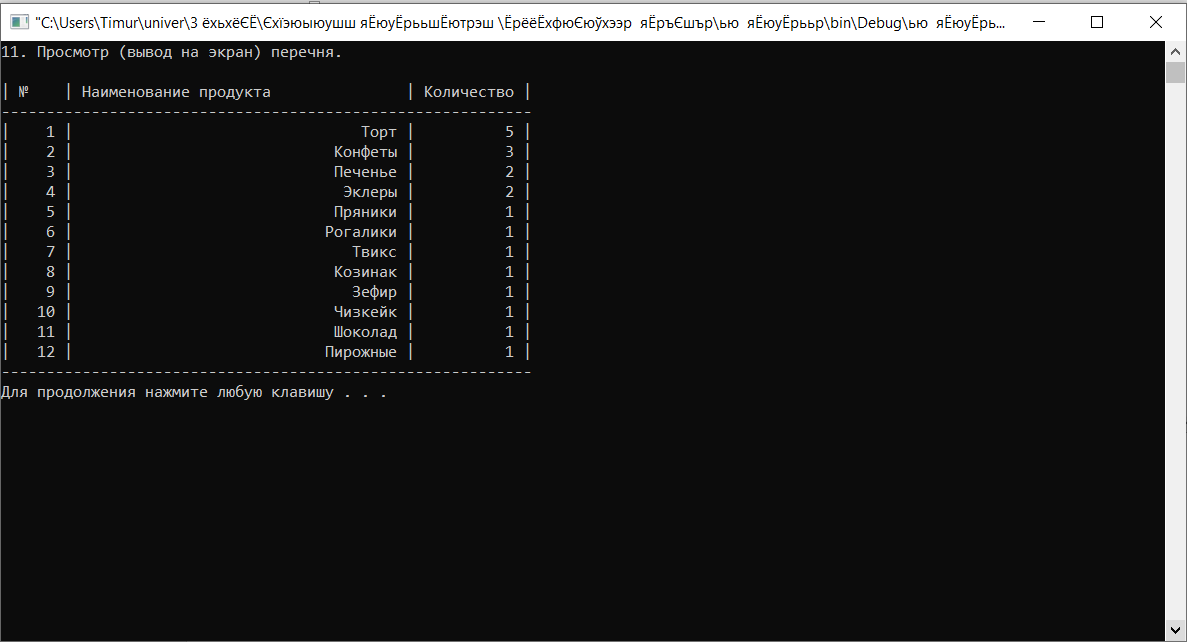


Рисунок 21 – Вывод на экран отсортированного перечня

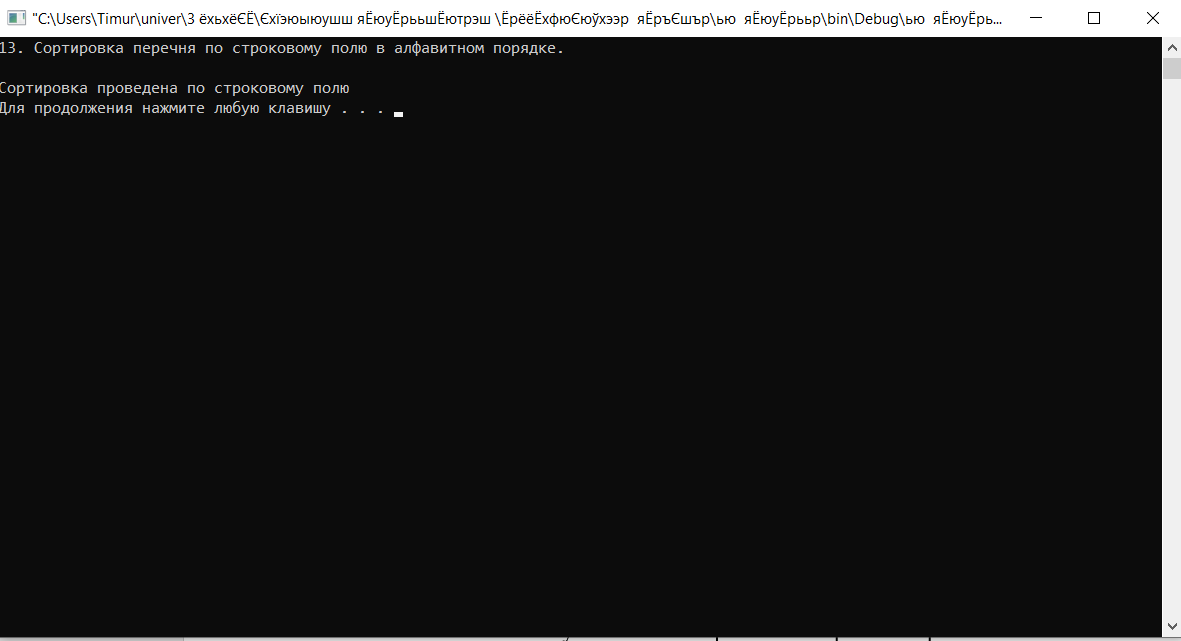


Рисунок 22 – Сортировка перечня по строковому полю в алфавитном порядке

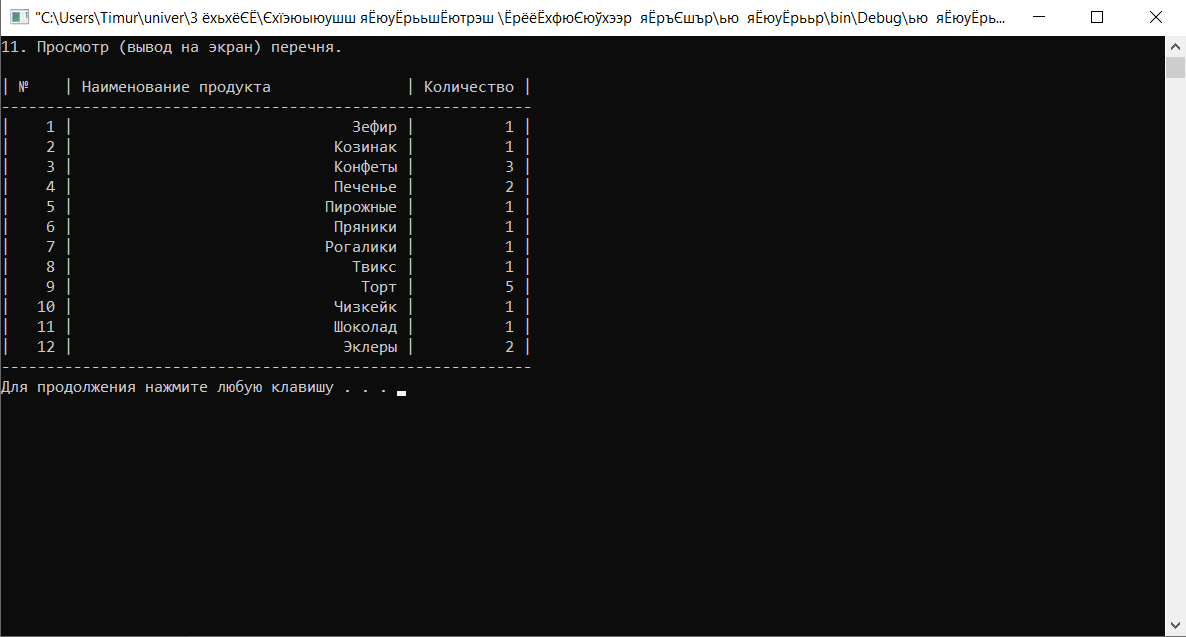


Рисунок 23 – Вывод на экран отсортированного перечня

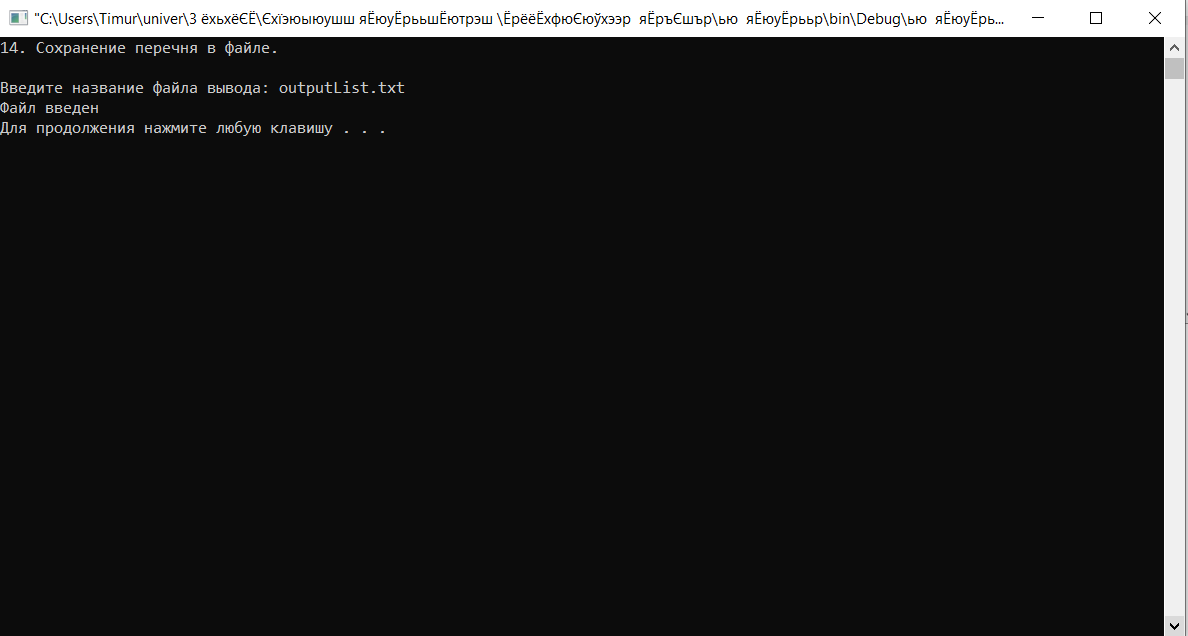


Рисунок 24 – Сохранение перечня в файле

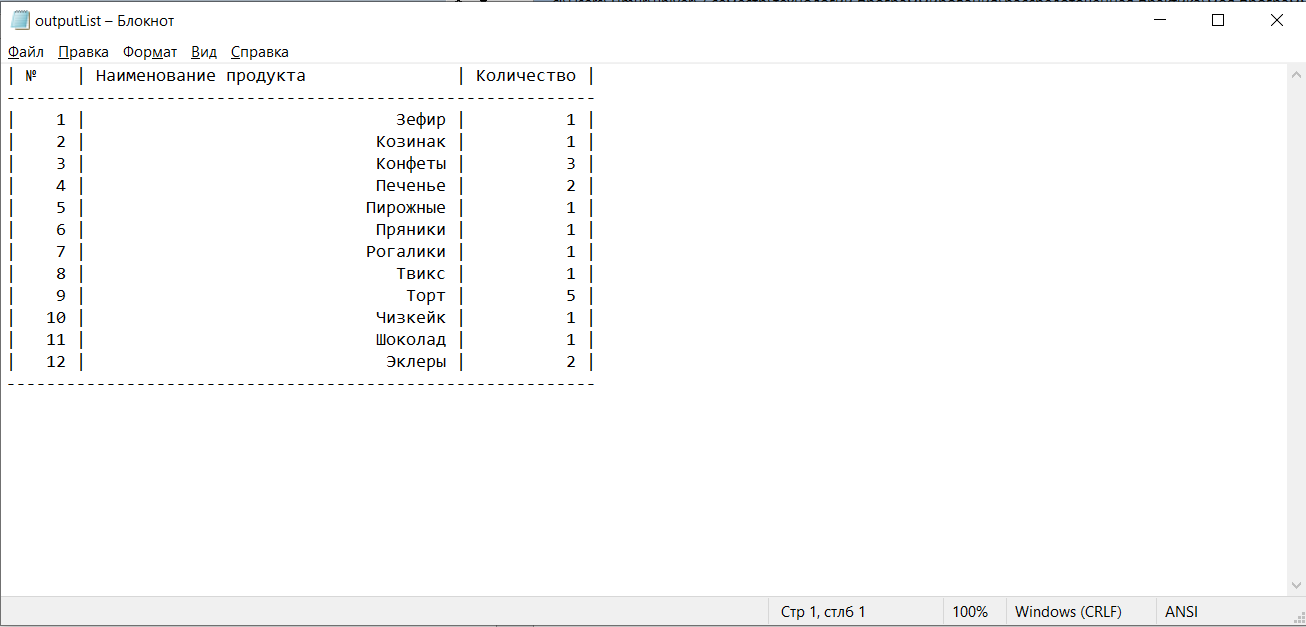


Рисунок 25 – Сохранение перечня в файле

## **Листинг программы**

#include <iostream>

#include <fstream>

#include <string.h>

#include <iomanip> // работа с форматированным выводом

#include <cstdlib> // генерация случайных чисел

#include <Windows.h> // корректное отображение кириллических записей

using namespace std;

// Глобальные переменные

int menuChoice = 0; // Номер выбранного пункта меню

string menuItem[15] = {""}; // Названия всех пунктов меню

struct sweet

{

string name; // Наименование продукта

string productionMethod; // Способ изготовления

int price; // Цена, руб

int storageLife; // Срок годности, мес

int calorificValue; // Калорийность, ккал

};

struct sweetNameNum

{

string name;

int num;

};

const int rowNumMax = 1000; // максимальное кол-во строк в базе данных

int rowNum = 0; // текущее кол-во строк в базе данных

sweet sweetDb[rowNumMax]; // база данных из кондитерских изделий

sweetNameNum sweetList[rowNumMax];

// 1. Загрузка информации из существующей базы данных (вывод из файла).

void inputDb(bool fillRand) // Если true, заполняет базу данных случайными значениями

{

string fileName;

string name, productionMethod; int price, storageLife, calorificValue;

cout << "Введите название файла ввода: "; cin >> fileName; // работа с input.txt

cout << "Файл введен" << endl;

if (fillRand)

{

ofstream fout;

fout.open("input.txt");

for (int i=0; i<20; i++) // Создается база данных из 20 строк

{

name = "Кондитерское\_изделие\_" + std::to\_string(i+1) + " ";

if ((rand()%2) == 0) {

productionMethod = "Ручной ";

}

else {

productionMethod = "Заводской ";

}

price = rand()%1000 + 100;

storageLife = rand()%40 + 3;

calorificValue = rand()%1000 + 200;

fout << name << productionMethod << price << " " << storageLife << " " << calorificValue << endl;

}

// Очистка содержимого файла

fout.close();

}

ifstream fin;

fin.open("input.txt");

do

{

// новая строка в базе данных

fin >> sweetDb[rowNum].name >> sweetDb[rowNum].productionMethod >> sweetDb[rowNum].price >> sweetDb[rowNum].storageLife >> sweetDb[rowNum].calorificValue;

rowNum++;

}

while (sweetDb[rowNum-1].name != ""); // если текущая ячейка (следовательно и строка) непустая, то следует добавлять строки далее

rowNum--; // убирается одна лишняя пустая строка

fin.close();

}

// 2. Просмотр (вывод на экран) существующей базы данных.

void outputConsoleDb()

{

cout << "| № | Наименование продукта | Способ изготовления | Цена, руб | Срок годности, мес | Калорийность, ккал |" << endl <<

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

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

{

cout << "| " << setw(4) << i+1 << " | " << setw(35) << sweetDb[i].name << " | " << setw(19) << sweetDb[i].productionMethod << " | " << setw(9) << sweetDb[i].price

<< " | " << setw(18) << sweetDb[i].storageLife << " |" << setw(19) << sweetDb[i].calorificValue << " | " << endl;

}

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

}

// 3. Добавление новой записи в базу данных.

void addRowDb()

{

rowNum++;

cout << "Введите наименование продукта: "; cin >> sweetDb[rowNum-1].name; // (rowNum-1) - номер последней строки в таблице

cout << "Введите способ изготовления: "; cin >> sweetDb[rowNum-1].productionMethod;

cout << "Введите цену, руб: "; cin >> sweetDb[rowNum-1].price;

cout << "Введите срок годности, мес: "; cin >> sweetDb[rowNum-1].storageLife;

cout << "Введите калорийность, ккал: "; cin >> sweetDb[rowNum-1].calorificValue;

}

// 4. Удаление записи из базы данных.

void deleteRowDb()

{

int row;

if

cout << "Введите номер удаляемой строки: "; cin >> row;

// каждая новая строка встает на место предыдущей

for (int i=row; i<rowNum; i++)

{

sweetDb[i-1].name = sweetDb[i].name;

sweetDb[i-1].productionMethod = sweetDb[i].productionMethod;

sweetDb[i-1].price = sweetDb[i].price;

sweetDb[i-1].storageLife = sweetDb[i].storageLife;

sweetDb[i-1].calorificValue = sweetDb[i].calorificValue;

}

rowNum--;

}

// 5. Редактирование записи в базе данных.

void editRowDb()

{

int row;

cout << "Введите номер изменяемой строки: "; cin >> row;

cout << "Введите новое наименование продукта: "; cin >> sweetDb[row-1].name;

cout << "Введите новый способ изготовления: "; cin >> sweetDb[row-1].productionMethod;

cout << "Введите новую цену: "; cin >> sweetDb[row-1].price;

cout << "Введите новый срок годности: "; cin >> sweetDb[row-1].storageLife;

cout << "Введите новую калорийность: "; cin >> sweetDb[row-1].calorificValue;

}

// 6. Сортировка записей по числовому полю. - по возрастанию

void sortNumDb()

{

sweet temp;

int submenuChoice;

cout << "1. Цена, руб" << endl;

cout << "2. Срок годности, мес" << endl;

cout << "3. Калорийность, ккал" << endl;

cout << "Выберите по какому полю сортировать (1-3): "; cin >> submenuChoice;

// сортировка пузырьком

for (int i=0; i<rowNum-1; i++)

{

for (int j=i+1; j<rowNum; j++)

{

switch (submenuChoice)

{

case 1: if (sweetDb[j].price < sweetDb[i].price) {temp = sweetDb[j]; sweetDb[j] = sweetDb[i]; sweetDb[i] = temp;} break;

case 2: if (sweetDb[j].storageLife < sweetDb[i].storageLife) {temp = sweetDb[j]; sweetDb[j] = sweetDb[i]; sweetDb[i] = temp;} break;

case 3: if (sweetDb[j].calorificValue < sweetDb[i].calorificValue){temp = sweetDb[j]; sweetDb[j] = sweetDb[i]; sweetDb[i] = temp;} break;

}

}

}

switch (submenuChoice)

{

case 1: cout << "Сортировка проведена по цене" << endl; break;

case 2: cout << "Сортировка проведена по сроку годности" << endl; break;

case 3: cout << "Сортировка проведена по калорийности" << endl; break;

default: cout << "Такого пункта меню не существует, сортировка не была проведена" << endl; break;

}

}

// 7. Сортировка записей по строковому полю. - по возрастанию

void sortStringDb()

{

sweet temp;

int submenuChoice;

cout << "1. Наименование продукта" << endl;

cout << "2. Способ изготовления" << endl;

cout << "Выберите по какому полю сортировать (1-2): "; cin >> submenuChoice;

// сортировка пузырьком

for (int i=0; i<rowNum-1; i++)

{

for (int j=i+1; j<rowNum; j++)

{

switch (submenuChoice)

{

case 1: if (sweetDb[j].name < sweetDb[i].name) {temp = sweetDb[j]; sweetDb[j] = sweetDb[i]; sweetDb[i] = temp;} break;

case 2: if (sweetDb[j].productionMethod < sweetDb[i].productionMethod) {temp = sweetDb[j]; sweetDb[j] = sweetDb[i]; sweetDb[i] = temp;} break;

}

}

}

switch (submenuChoice)

{

case 1: cout << "Сортировка проведена по наименованию продукта" << endl; break;

case 2: cout << "Сортировка проведена по способу изготовления" << endl; break;

default: cout << "Такого пункта меню не существует, сортировка не была проведена" << endl; break;

}

}

// 8. Сохранение результатов обработки (запись в файл).

void outputFileDb()

{

string fileName;

ofstream fout;

cout << "Введите название файла вывода: "; cin >> fileName; // работа с input.txt

cout << "Файл введен" << endl;

fout.open("output.txt");

fout << "| № | Наименование продукта | Способ изготовления | Цена, руб | Срок годности, мес | Калорийность, ккал |" << endl <<

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

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

{

fout << "| " << setw(4) << i+1 << " | " << setw(35) << sweetDb[i].name << " | " << setw(19) << sweetDb[i].productionMethod << " | " << setw(9) << sweetDb[i].price

<< " | " << setw(18) << sweetDb[i].storageLife << " |" << setw(19) << sweetDb[i].calorificValue << " | " << endl;

}

fout << "--------------------------------------------------------------------------------------------------------------------------" << endl;

fout.close();

}

// 9. Поиск в базе данных (по одному и двум поисковым признакам) с выводом на экран найденных записей или сообщения о неуспешном поиске. - по наименованию продукта

void searchDb()

{

int rowNumSearch=1;

string name;

cout << "Введите наименование продукта, который нужно найти: "; cin >> name;

cout << "| № | Наименование продукта | Способ изготовления | Цена, руб | Срок годности, мес | Калорийность, ккал |" << endl <<

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

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

{

if (sweetDb[i].name == name) {

cout << "| " << setw(4) << rowNumSearch << " | " << setw(35) << sweetDb[i].name << " | " << setw(19) << sweetDb[i].productionMethod << " | " << setw(9) << sweetDb[i].price

<< " | " << setw(18) << sweetDb[i].storageLife << " |" << setw(19) << sweetDb[i].calorificValue << " | " << endl;

rowNumSearch++;

}

}

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

}

// 10. Формирование перечня. - название

void createList()

{

// создается копия массива sweetDb и далее идет работа с ней, чтобы не "испортить" значения изначального массива

sweet sweetDbCopy[rowNumMax];

for (int i=0; i<rowNum; i++) {

sweetDbCopy[i] = sweetDb[i];

}

// Все следующие повторяющиеся наименования продуктов удаляются, остается по одному слову от каждого наименования

for (int i=0; i<rowNum; i++) {

sweetList[i].name = sweetDbCopy[i].name;

sweetList[i].num = 1;

for (int j=i+1; j<rowNum; j++) {

if (sweetDbCopy[i].name == sweetDbCopy[j].name) {

sweetDbCopy[j].name = "";

sweetList[i].num++;

}

}

}

cout << "Перечень сформирован" << endl;

}

// 11. Просмотр (вывод на экран) перечня.

void outputConsoleList()

{

int row=0;

cout << "| № | Наименование продукта | Количество |" << endl <<

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

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

{

if (sweetList[i].name != "") {

row++;

cout << "| " << setw(4) << row << " | " << setw(35) << sweetList[i].name << " | " << setw(10) << sweetList[i].num << " |" << endl;

}

}

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

}

// 12. Сортировка перечня по числу.

void sortNumList()

{

sweetNameNum temp;

for (int i=0; i<rowNum-1; i++) {

for (int j=i+1; j<rowNum; j++) {

if (sweetList[j].num > sweetList[i].num) {

temp = sweetList[j];

sweetList[j] = sweetList[i];

sweetList[i] = temp;

}

}

}

cout << "Сортировка проведена по числовому полю" << endl;

}

// 13. Сортировка перечня по строке.

void sortStringList()

{

sweetNameNum temp;

for (int i=0; i<rowNum-1; i++) {

for (int j=0; j<rowNum; j++) {

if (sweetList[j].name > sweetList[i].name) {

temp = sweetList[j];

sweetList[j] = sweetList[i];

sweetList[i] = temp;

}

}

}

cout << "Сортировка проведена по строковому полю" << endl;

}

// 14. Сохранение перечня в файле.

void outputFileList()

{

string fileName;

ofstream fout;

cout << "Введите название файла вывода: "; cin >> fileName;

cout << "Файл введен" << endl;

fout.open("outputList.txt");

int row=0;

fout << "| № | Наименование продукта | Количество |" << endl <<

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

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

{

if (sweetList[i].name != "") {

row++;

fout << "| " << setw(4) << row << " | " << setw(35) << sweetList[i].name << " | " << setw(10) << sweetList[i].num << " |" << endl;

}

}

fout << "-----------------------------------------------------------" << endl;

}

void menu()

{

// Вывод меню

cout << "Меню:" << endl;

for (int i=0; i<15;i++)

cout << menuItem[i] << endl;

cout << endl << "Выберите пункт меню (0-14): ";

cin >> menuChoice;

system("cls");

if ((menuChoice >= 0) && (menuChoice <= 14))

cout << menuItem[menuChoice] << endl << endl; // вывести пункт меню, если он существует

switch(menuChoice)

{

case 0: exit(0); break;

case 1: inputDb(false); break;

case 2: outputConsoleDb(); break;

case 3: addRowDb(); break;

case 4: deleteRowDb(); break;

case 5: editRowDb(); break;

case 6: sortNumDb(); break;

case 7: sortStringDb(); break;

case 8: outputFileDb(); break;

case 9: searchDb(); break;

case 10: createList(); break;

case 11: outputConsoleList(); break;

case 12: sortNumList(); break;

case 13: sortStringList(); break;

case 14: outputFileList(); break;

default: cout << "Такого пункта меню не существует, попробуйте другой" << endl; break;

}

system("pause");

system("cls");

menu();

}

int main()

{

SetConsoleCP(1251);// установка кодовой страницы win-cp 1251 в поток ввода

SetConsoleOutputCP(1251); // установка кодовой страницы win-cp 1251 в поток вывода

menuItem[ 0] = "0. Выход из программы.";

menuItem[ 1] = "1. Загрузка информации из существующей базы данных (вывод из файла).";

menuItem[ 2] = "2. Просмотр (вывод на экран) существующей базы данных.";

menuItem[ 3] = "3. Добавление новой записи в базу данных.";

menuItem[ 4] = "4. Удаление записи из базы данных.";

menuItem[ 5] = "5. Редактирование записи в базе данных.";

menuItem[ 6] = "6. Сортировка записей по числовому полю в порядке возрастания.";

menuItem[ 7] = "7. Сортировка записей по строковому полю в алфавитном порядке.";

menuItem[ 8] = "8. Сохранение результатов обработки (запись в файл).";

menuItem[ 9] = "9. Поиск в базе данных по наименованию продукта с выводом на экран найденных записей или сообщения о неуспешном поиске.";

menuItem[10] = "10. Формирование перечня.";

menuItem[11] = "11. Просмотр (вывод на экран) перечня.";

menuItem[12] = "12. Сортировка перечня по числовому полю в порядке убывания.";

menuItem[13] = "13. Сортировка перечня по строковому полю в алфавитном порядке.";

menuItem[14] = "14. Сохранение перечня в файле.";

menu();

return 0;

}

### Заключение

В ходе данной учебной рассредоточенной практики была разработана программа, с помощью которой можно создавать и редактировать файлы, содержащие записи базы данных «Кондитерских изделий». Программа разработана на языке C++. Так же были разработаны исходные файлы данных для верификации программы, программа была протестирована и отлажена.

Разработанная программа «База данных кондитерских изделий» позволяет пользователю вести базу кондитерских изделий. Данная программа разработана для помощи в управлении кондитерской.

### Список литературы

1. ГОСТ 19.701-90 (ИСО 5807-85). Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М., 1990.
2. Лафоре Р. Объектно-ориентированное программирование в С++. – СПб.: Питер, 2017. – 928 с.
3. Павловская Т.А. С++. Программирование на языке высокого уровня: Учебник. СПб.: Питер, 2019. – 464 с.
4. Стив Макконнелл. Совершенный код. Мастер-класс (2016). БХВ-Петербург, 2016. – 896 c.
5. Сталберг Брэд, Магнесс Стив. На пике. Как поддержать максимальную эффективность без выгорания. МИФ. Личное развитие,2018. – 272 с.