|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Внешние структуры данных: текстовый и двоичный файлы»** | |
|  | |
| Выполнил студент группы ИКБО-07-21 | Дамарад Д.В. |
| Принял преподаватель | Скворцова Л.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

# **ЦЕЛЬ РАБОТЫ**

Получить навыки применения файловых потоков языка С++ (или файлов языка Си) по управлению текстовым и двоичным файлами.

# **ПОСТАНОВКА ЗАДАЧИ 1-ОГО ЗАДАНИЯ**

1. Разработать программу, управления текстовым файлом.
2. Реализация ввода-вывода на основе файловых потоков С++: ofstream, ifstream.
3. Имя физического файла вводится пользователем и передается в функции обработки через параметр.

Требования к упражнениям:

1. Создание текстового файла кодировки ASCII, содержащего десятичные числа по несколько чисел на строке.
2. Вывод содержимого текстового файла.
3. Добавление новой строки в конец файла.
4. Прочитать значение числа, указав его порядковый номер в файле, и вернуть его значение.
5. Определить количество чисел в файле.
6. Разработать приложение и выполнить тестирование всех функций. Приложение должно содержать диалоговый интерфейс на основе текстового меню.
7. Приложение должно осуществлять контроль открытия и существования файла в основной программе перед вызовом функций. Перед закрытием файла, проверять отсутствие ошибок ввода и вывода (метод good).
8. Создать модуль и перенести в него все отлаженные функции. Исключить функции из приложения. Отладить приложение, подключив к нему модуль с функциями.
9. Разработать функции для реализации дополнительных операций, определенных вариантом и сохранить их в модуле с остальными функциями.
10. Выполнить тестирование приложения в полном объеме.
11. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Вариант №6. Дополнительное задание:

|  |  |
| --- | --- |
| Дополнительная операция | Создать новый файл из значений исходного, умножив каждое число на сумму первого и последнего чисел исходного файла. |

# **РЕШЕНИЕ 1-ОГО ЗАДАНИЯ**

В С++ аппарат работы с файлами основан на объектах классов, хранящихся в файлах ostream и istream, fstream. Класс файловых потоков – это иерархия, базовым классов этой иерархии является абстрактный класс с именем ios, который находится на вершине иерархии. Он используется только в качестве базового класса для остальных потоков ввода-вывода. Поток – это логический интерфейс к файлу. При работе с потоковой библиотекой ввода-вывода программист обычно достаточно активно использует следующие классы: ifstream, ofstream, fstream. Метод открытия файла потока – open. Так как поток — это объект класса, то открытие можно выполнить методом open.

Формат метода: open([имя файла,] [способ открытия файла]).

Способы открытия файла:

app – открыть существующий файл для записи в конец (указатель в конец);

out - открыть файл для создания и записи в начало (указатель на первую запись);

in - открыть файл для чтения из файла с первой записи (указатель на первую запись);

trunc – очистить файл, если он существует;

ate - переместить указатель на конец файла.

При открытии файла можно задавать несколько режимов открытия в одной строке, в этом случае режимы разделяются символом |.

Закрытие файла осуществляет метод осуществляется методом сlose().

Чтение осуществляется: операцией выбора из потока>>; функцией getline текста с пробелами между словами; методами потока: get().

1. Для создания текстового файла и заполнения его числами используем объект класса ofstream и метод открытия ios::out. Для записи чисел в файл используем рандомайзер для определения количества строк, количества чисел в строке и сами числа.

Функция для создания файла и заполнениями его числами:

void CreateAndFillFile(string fileName) {

ofstream fout;

fout.open(fileName, ios::out);

int lines = rand() % 10 + 1;

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

int countNumbers = rand() % 6 + 1;

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

fout << rand() % 20 + 1 << " ";

}

if (i < lines - 1) {

fout << endl;

}

}

if (fout.good()) {

fout.close();

}

else {

return;

}

}

1. Для открытия текстового файла и вывода его содержимого используем объект класса ifstream и метод открытия ios::in. Для проверки существования файла используем метод is\_open(). Затем воспользуемся методом eof() в цикле для вывода всех значений файла.

Функция для вывода значений файла:

void PrintFile(string fileName) {

ifstream fin;

fin.open(fileName, ios::in);

if (fin.is\_open()) {

cout << "Содержимое файла " << fileName << ":" << "\n";

string line;

while (!fin.eof()) {

getline(fin, line);

cout << line << endl;

}

}

else {

cout << "Файла с таким именем не существует" << endl;

}

if (fin.good()) {

fin.close();

}

else {

return;

}

}

1. Для добавления новой строки к файлу используем объект класса ofstream и метод открытия ios::app. Добавляем новую строку и закрываем файл.

Функция для добавления новой строки в файл:

void AppendLine(string fileName, string line) {

ofstream fout;

fout.open(fileName, ios::app);

fout << endl << line;

if (fout.good()) {

fout.close();

}

else {

return;

}

}

1. Для того чтобы вернуть число по номеру в файле воспользуемся классом ifstream с методом открытия ios::in. Инициализируем переменную resultNumber нулем. Проверяем возможно ли открыть файл методом is\_open и с помощью цикла со счетчиком считываем данные из файла до заданного индекса включительно.

Функция для получения номера по индексу:

int GetNumber(string fileName) {

int resultNumber = 0;

ifstream fin;

fin.open(fileName, ios::in);

if (fin.is\_open()) {

int numberIndex;

cout << "Номер искомого числа, начиная с 0: ";

cin >> numberIndex;

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

fin >> resultNumber;

}

}

else {

cout << "Файла с таким именем не существует" << endl;

}

if (fin.good()) {

fin.close();

}

else {

return -1;

}

return resultNumber;

}

1. Для определения количества чисел в файле используем объект класса ifstream и метод открытия ios::in. Инициализируем локальную переменную count. Используем метод is\_open() для контролирования открытия файла, метод eof() для обхода всех чисел, в цикле увеличиваем значение count на 1.

Функция для подсчета чисел:

int CountNumbers(string fileName){

int n, count = 0;

ifstream fin;

fin.open(fileName, ios::in);

if (fin.is\_open()) {

while (fin >> n) {

count++;

}

}

else {

cout << "Файла с таким именем не существует" << endl;

return -1;

}

if (fin.good()) {

fin.close();

}

return count;

}

1. Для решения дополнительного задания используем объекты классов ifstream и ofstream с методами ios::in и ios::out соответственно. Создаем массив, где будем хранить числа из файла, чтобы определить множитель, который будет равняться сумме первого и последнего элемента массива. Проверяем существование файла с помощью метода is\_open() класса ifsream. Затем заполняем массив числами из файла. Вычисляем множитель. Вторым циклом считываем значения из читаемого файла и записываем эти элементы, умноженные на вычисленный множитель в объект класса ofstream.

Функция для решения дополнительной задачи варианта:

void AdditionalTask(string fileName) {

ifstream fin;

fin.open(fileName, ios::in);

if (fin.is\_open()) {

int temp, n, i = 0, size = CountNumbers(fileName);

int\* numbers = new int[size];

while (fin >> n) {

numbers[i] = n;

i++;

}

fin.close();

int multiplier = numbers[0] + numbers[size-1];

ofstream fout;

fout.open("new\_file.txt", ios::out);

fin.open(fileName, ios::in);

while (fin >> temp) {

fout << temp \* multiplier << " ";

}

if (fin.good() && fout.good()){

fout.close();

fin.close();

}

else {

return;

}

}

else {

cout << "Файла с таким именем не существует" << endl;

}

fin.close();

}

Описание интерфейса программы: интерфейс (Рисунок 1) представляет собой меню из 7 пунктов. После каждого выбранного пункта будет необходим ввод имени файла. Если введено число не соответствующее номеру задания, то программа предложит повоторить ввод. При выполнении очередной задачи меню открывается заново.

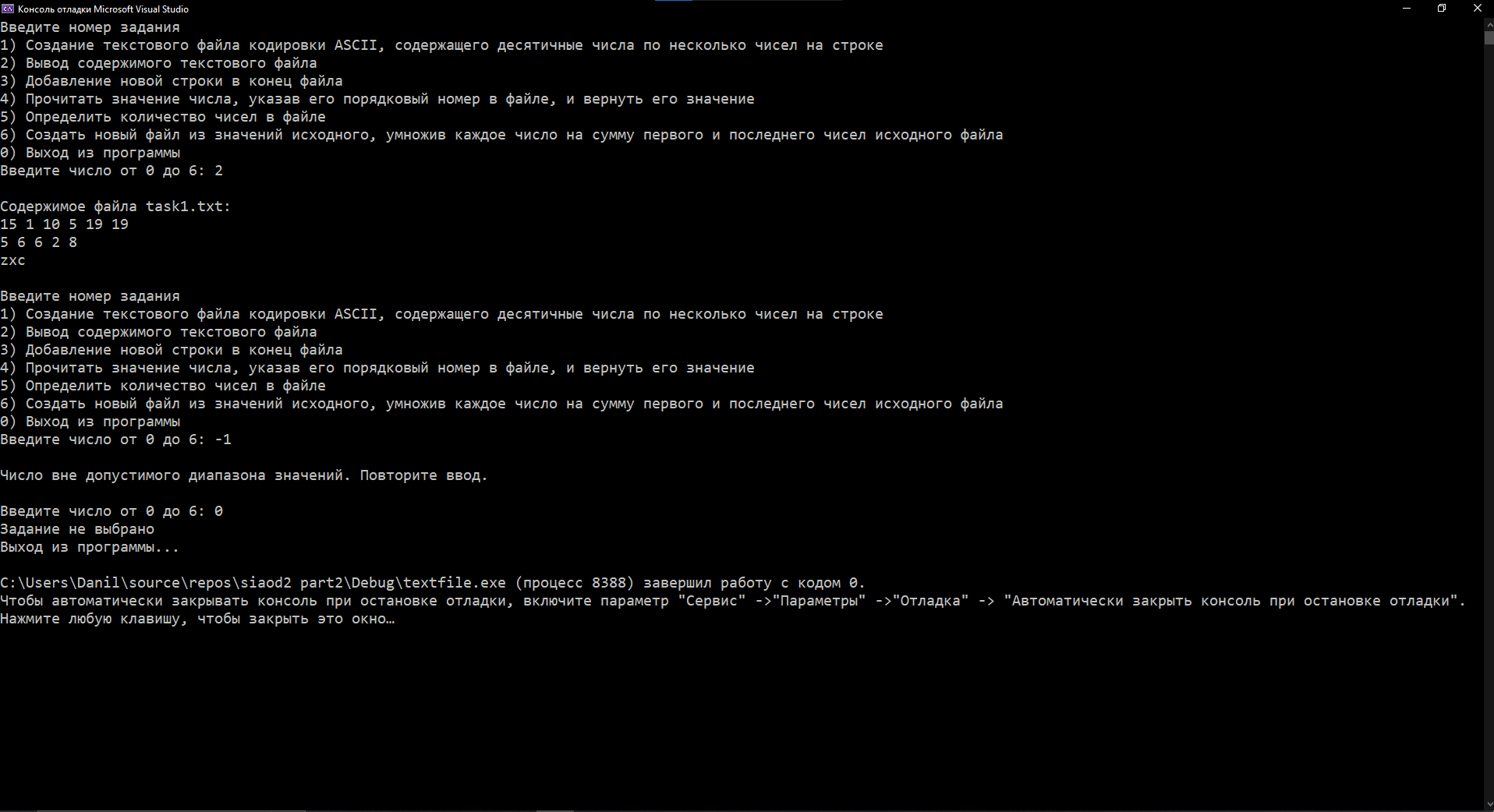


Рисунок 1 - Интерфейс программы для 1-ого задания

# **ТЕСТИРОВАНИЕ 1-ОГО ЗАДАНИЯ**

1. Тестирование для первого задания не представляет необходимости, так как при вводе названия несуществующего файла файл создастся и числа запишутся в новый файл.
2. При тестировании второго задания (Рисунок 2) введем корректный путь и некорректный путь к файлу, в случае с корректным путем будем выведено содержимое файла, ином случае будет выведено уведомление о несуществующем файле и переход в меню.
3. В третьем задании тестирование не необходимо, так как при вводе названия несуществующего файла файл создастся, и введённая строка допишется в конец.
4. В четвертом задании (Рисунок 3) при некорректном пути выводится уведомление об ошибке и переход в меню, при корректном пути файл либо выводит число под требуемым индексом, либо выводит -1, если индекс превосходит количество элементов в файле.
5. При тестировании пятого задания (Рисунок 4) введем корректный путь и некорректный путь к файлу, в случае с корректным путем будет выведено количество чисел в файле, в ином случае будет выведено уведомление о файле и переход в меню.
6. В шестом задании (Рисунок 5) при введении некорректного пути выскочит уведомление и перенесет в меню, в ином случае создатся новый файл с числами из исходного файла, умноженными на сумму первого и последнего чисел исходного файла (Рисунок 6).

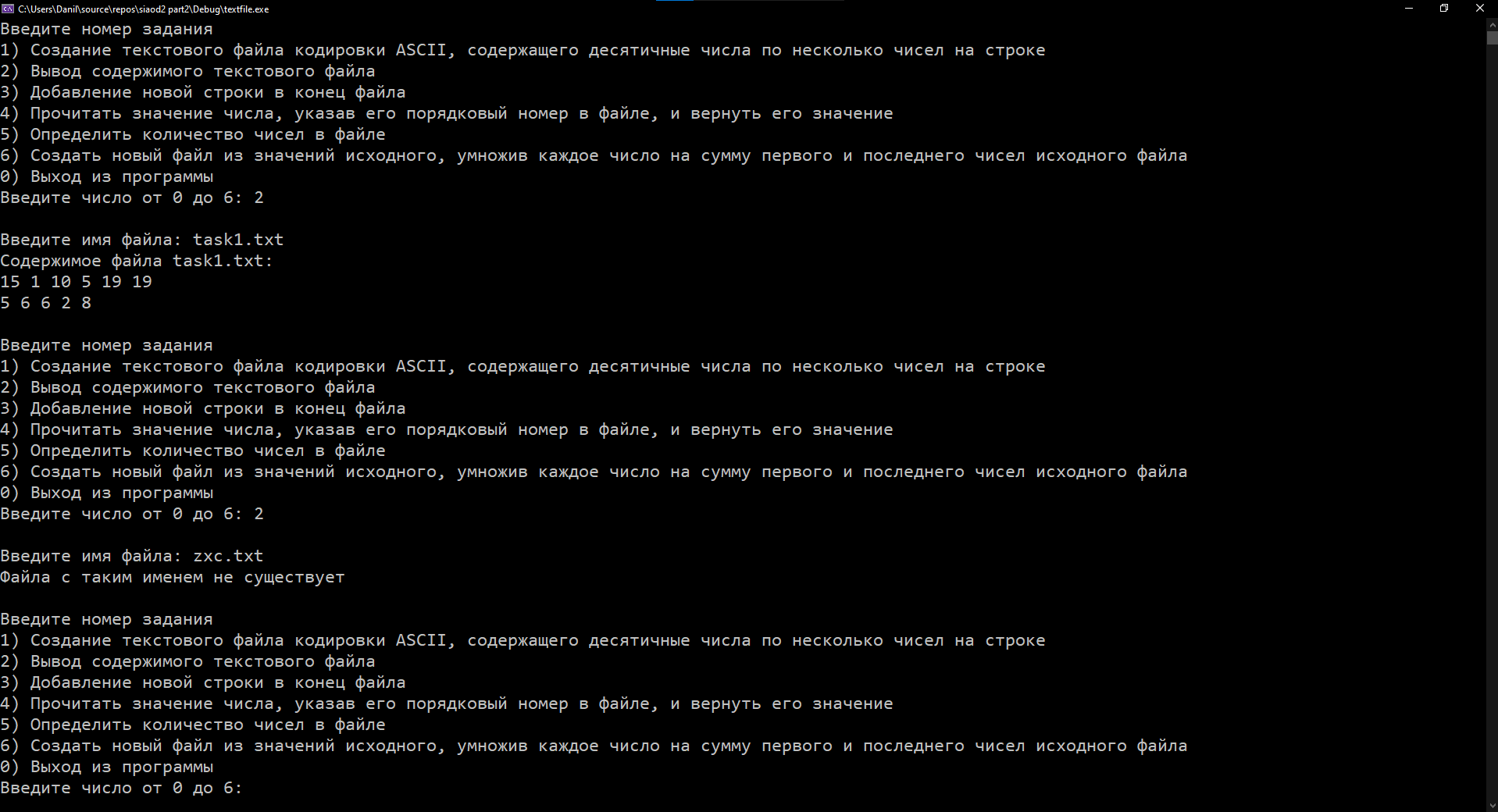


Рисунок 2 - Тестирование 2-ого задания

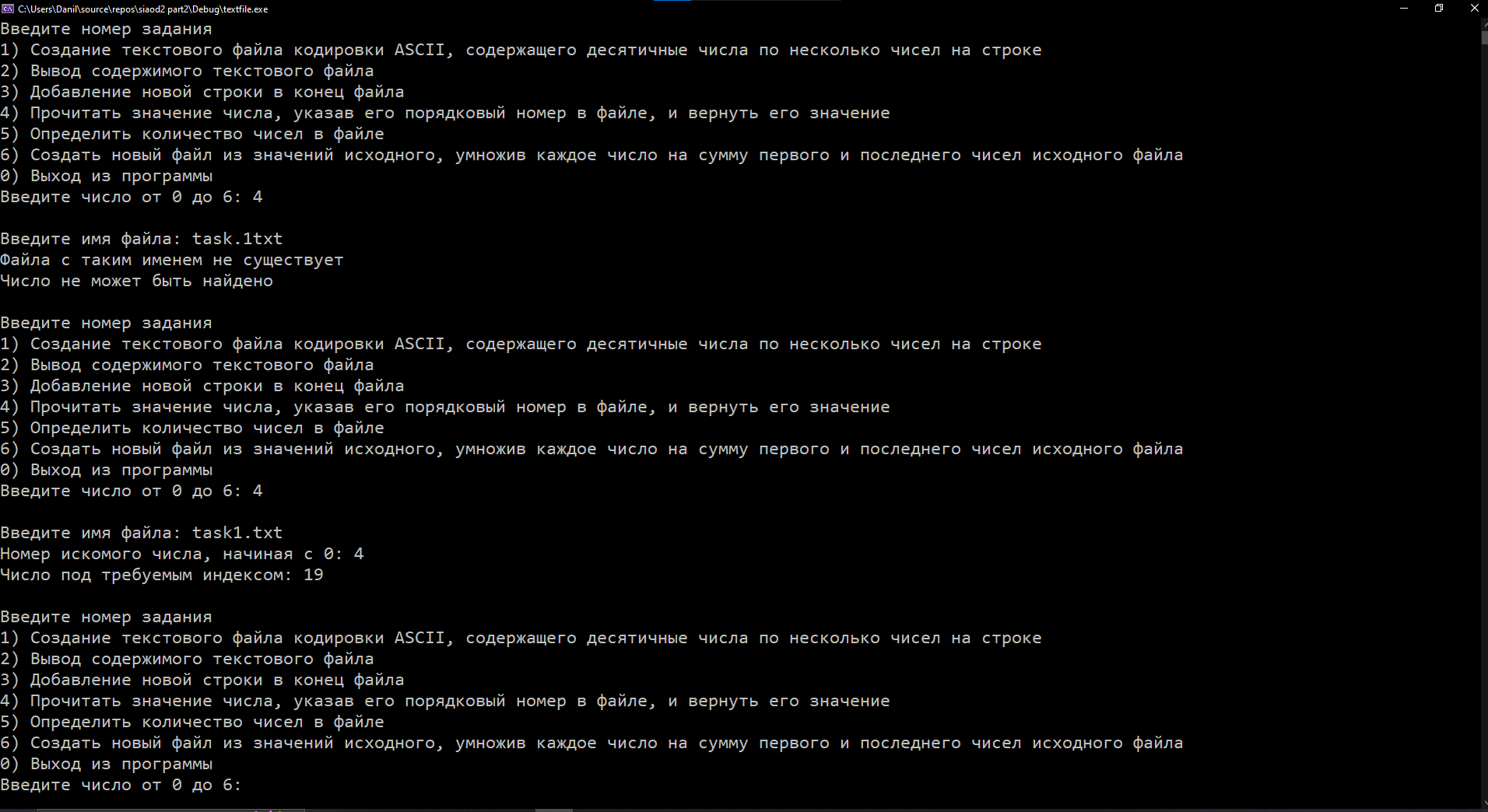


Рисунок 3 - Тестирование 4-ого задания

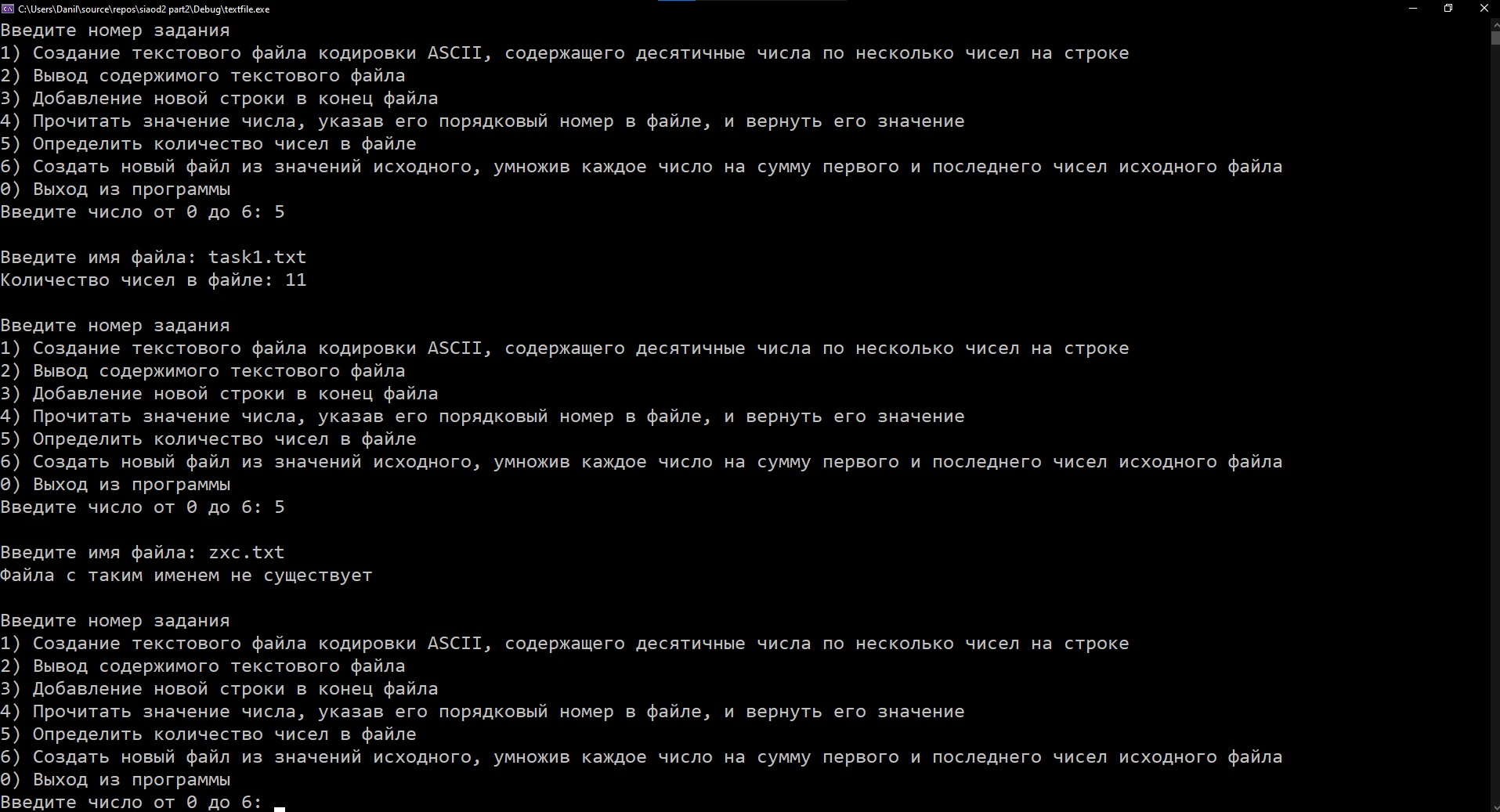


Рисунок 4 - Тестирование 5-ого задания

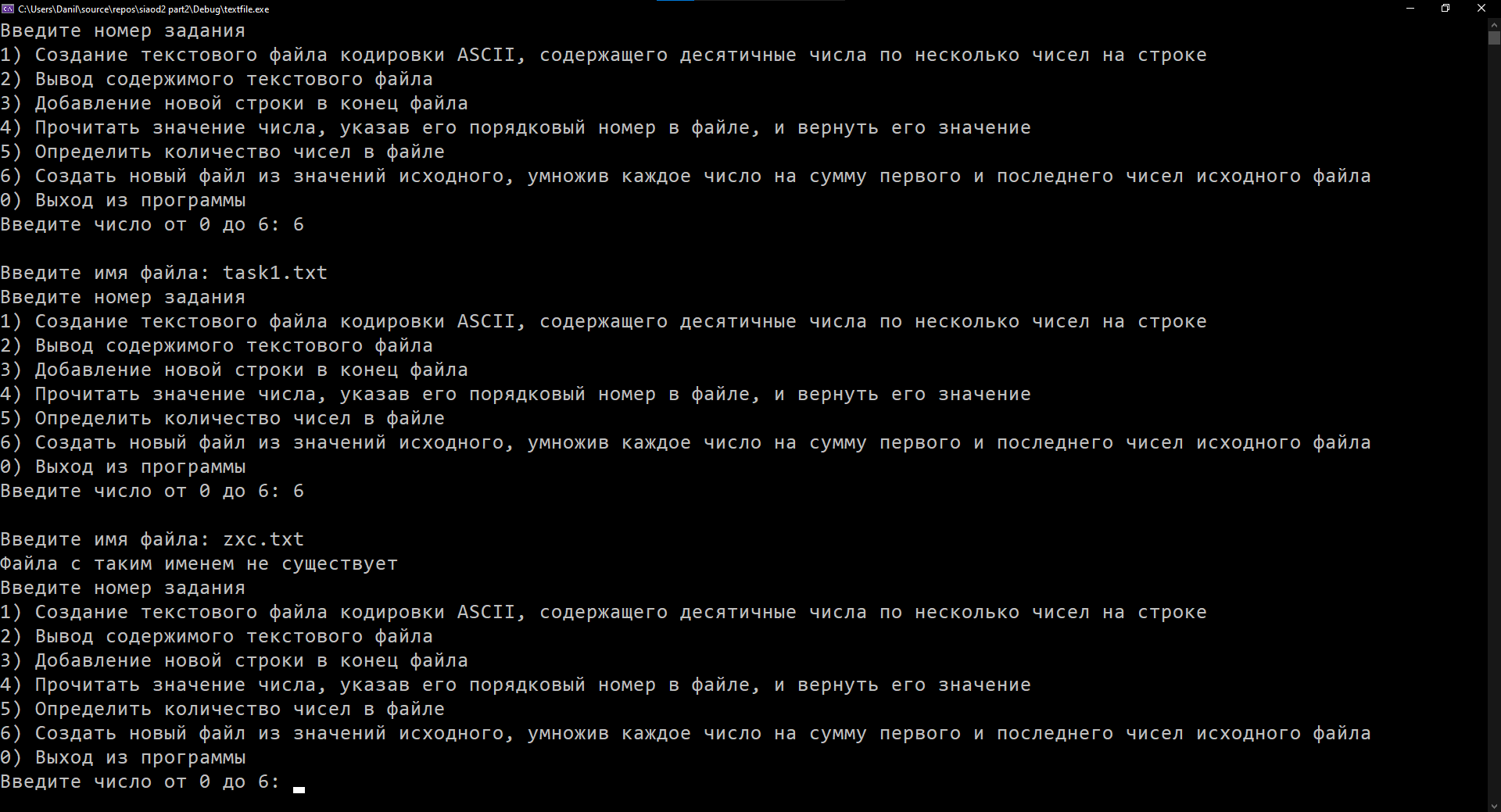


Рисунок 5 - Тестирование 6-ого задания

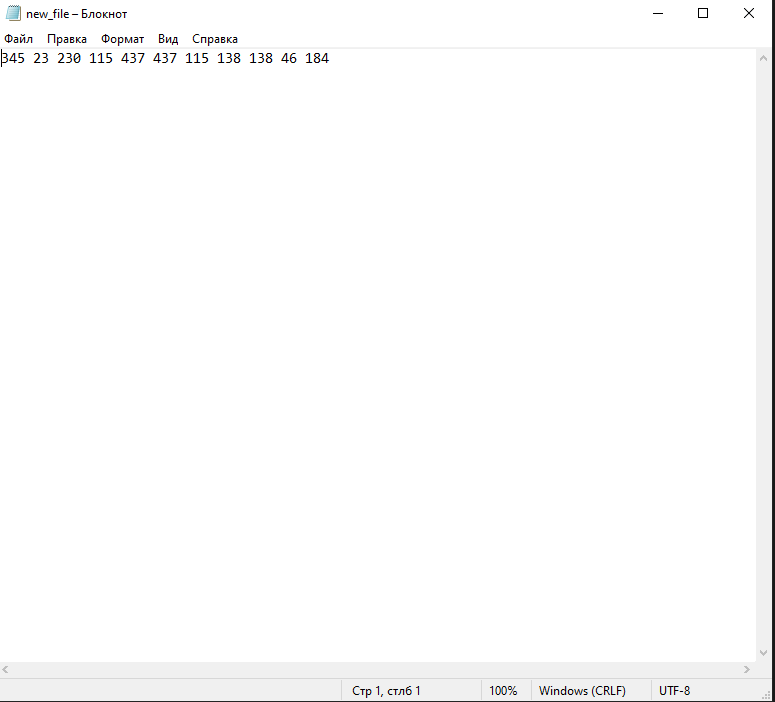


Рисунок 6 - Созданный файл в результате выполнения 6-ого задания

# **ПОЛНЫЙ КОД ПРОГРАММЫ 1-ОГО ЗАДАНИЯ**

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

int inputMenu(int& number) {

cout << "Введите номер задания" << endl;

cout << "1) Cоздание текстового файла кодировки ASCII, содержащего десятичные числа по несколько чисел на строке" << endl;

cout << "2) Вывод содержимого текстового файла" << endl;

cout << "3) Добавление новой строки в конец файла" << endl;

cout << "4) Прочитать значение числа, указав его порядковый номер в файле, и вернуть его значение" << endl;

cout << "5) Определить количество чисел в файле" << endl;

cout << "6) Создать новый файл из значений исходного, умножив каждое число на сумму первого и последнего чисел исходного файла" << endl;

cout << "0) Выход из программы" << endl;

while (1) {

cout << "Введите число от 0 до 6: ";

if ((cin >> number).good() && (number >= 0) && (number <= 6)) {

return number;

}

if (cin.fail()) {

cin.clear();

cout << endl << "Неверный ввод, повторите." << endl << endl;

}

else {

cout << endl << "Число вне допустимого диапазона значений. Повторите ввод." << endl << endl;

}

cin.ignore(100, '\n');

}

}

void CreateAndFillFile(string fileName) {

ofstream fout;

fout.open(fileName, ios::out);

int lines = rand() % 10 + 1;

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

int countNumbers = rand() % 6 + 1;

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

fout << rand() % 20 + 1 << " ";

}

if (i < lines - 1) {

fout << endl;

}

}

if (fout.good()) {

fout.close();

}

else {

return;

}

}

void PrintFile(string fileName) {

ifstream fin;

fin.open(fileName, ios::in);

if (fin.is\_open()) {

cout << "Содержимое файла " << fileName << ":" << "\n";

string line;

while (!fin.eof()) {

getline(fin, line);

cout << line << endl;

}

}

else {

cout << "Файла с таким именем не существует" << endl;

}

if (fin.good()) {

fin.close();

}

else {

return;

}

}

void AppendLine(string fileName, string line) {

ofstream fout;

fout.open(fileName, ios::app);

fout << endl << line;

if (fout.good()) {

fout.close();

}

else {

return;

}

}

int GetNumber(string fileName) {

int resultNumber = 0;

ifstream fin;

fin.open(fileName, ios::in);

if (fin.is\_open()) {

int numberIndex;

cout << "Номер искомого числа, начиная с 0: ";

cin >> numberIndex;

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

fin >> resultNumber;

}

}

else {

cout << "Файла с таким именем не существует" << endl;

}

if (fin.good()) {

fin.close();

}

else {

return -1;

}

return resultNumber;

}

int CountNumbers(string fileName){

int n, count = 0;

ifstream fin;

fin.open(fileName, ios::in);

if (fin.is\_open()) {

while (fin >> n) {

count++;

}

}

else {

cout << "Файла с таким именем не существует" << endl;

return -1;

}

if (fin.good()) {

fin.close();

}

return count;

}

void AdditionalTask(string fileName) {

ifstream fin;

fin.open(fileName, ios::in);

if (fin.is\_open()) {

int temp, n, i = 0, size = CountNumbers(fileName);

int\* numbers = new int[size];

while (fin >> n) {

numbers[i] = n;

i++;

}

fin.close();

int multiplier = numbers[0] + numbers[size-1];

ofstream fout;

fout.open("new\_file.txt", ios::out);

fin.open(fileName, ios::in);

while (fin >> temp) {

fout << temp \* multiplier << " ";

}

if (fin.good() && fout.good()){

fout.close();

fin.close();

}

else {

return;

}

}

else {

cout << "Файла с таким именем не существует" << endl;

}

fin.close();

}

int main() {

system("chcp 1251 > null");

int menu;

string fileName;

while (true) {

inputMenu(menu);

if (menu == 0) {

cout << "Задание не выбрано\nВыход из программы..." << endl;

break;

}

cout << endl;

switch (menu) {

case 1: {

cout << "Введите имя файла: ";

cin >> fileName;

cin.ignore();

cin.clear();

CreateAndFillFile(fileName);

break;

}

case 2: {

cout << "Введите имя файла: ";

cin >> fileName;

cin.ignore();

cin.clear();

PrintFile(fileName);

cout<< endl;

break;

}

case 3: {

cout << "Введите имя файла: ";

cin >> fileName;

cin.ignore();

cin.clear();

string lineToAppend;

cout << "Введите строку, которую хотите добавить: ";

cin.get();

getline(cin, lineToAppend);

AppendLine(fileName, lineToAppend);

break;

}

case 4: {

cout << "Введите имя файла: ";

cin >> fileName;

cin.ignore();

cin.clear();

int result = GetNumber(fileName);

if (result == -1) {

cout << "Число не может быть найдено" << endl << endl;

}

else {

cout << "Число под требуемым индексом: " << result << endl << endl;

}

break;

}

case 5: {

cout << "Введите имя файла: ";

cin >> fileName;

cin.ignore();

cin.clear();

int count = CountNumbers(fileName);

if (count == -1) {

cout << endl;

}

else {

cout << "Количество чисел в файле: " << count << endl << endl;

}

break;

}

case 6: {

cout << "Введите имя файла: ";

cin >> fileName;

cin.ignore();

cin.clear();

AdditionalTask(fileName);

break;

}

default: {

cout << "Задание не выбрано\nВыход из программы..." << endl;

break;

}

}

}

return 0;

}

# **ПОСТАНОВКА ЗАДАЧИ 2-ОГО ЗАДАНИЯ**

1. Разработать программу управления двоичными файлами с записями фиксированной длины. Общие требования: файл состоит из записей определенной структуры, согласно варианту.
2. Разработать структуру записи двоичного файла согласно варианту задания.
3. Имя файла вводит пользователь.

Разработать функции для выполнения операций:

1. Преобразование тестовых данных из текстового файла в двоичный файл.
2. Преобразование данных из двоичного файла в текстовый.
3. Вывод всех записей двоичного файла.
4. Доступ к записи по ее порядковому номеру в файле, используя механизм прямого доступа к записи в двоичном файле.
5. Удаление записи с заданным значением ключа, выполнить путем замены на последнюю запись.
6. При открытии файла обеспечить контроль существования и открытия файла.
7. При применении механизма прямого доступа к записи файла выполнить контроль присутствия записи с заданным номером в файле.
8. Сохраните функции в новом модуле.
9. Разработать приложение, демонстрирующее выполнение всех операций, подключив к нему модуль с функциями.
10. Выполнить тестирование приложения, продемонстрировав выполнение всех операций.

Вариант №10. Дополнительное задание:

|  |  |
| --- | --- |
| Структура записи | Товар: название, код – шестиразрядное число, завод изготовитель, цена, страна (название). |
| Доп. операция | 1. Сформировать текстовый файл о поставках заданной страны, размещая на строке все сведения о товаре, кроме страны. Название страны записать в первой строке файла. 2. Добавить новую запись о товаре в конец файла. |

# **РЕШЕНИЕ**

Двоичные файлы используются для более компактного хранения информации. Хранят данные в машинном формате, т.е. в том виде как они представлены в оперативной памяти. Например, значение вещественной переменной типа float будет записано в файл в формате с плавающей точкой. Под это значение будет отведено столько байт, сколько требуется переменной формата float.

Логически файл состоит из записей фиксированной длины, минимальная длина один байт, максимальная соответствует размеру записанной при создании файла записи. Поэтому из такого файла удобно читать записи: каким форматом записали, таким и будем читать.

Создается файл программным путем. Как и текстовый файл имеет последовательную организацию, т.е. новые записи добавляются в конец файла. Обрабатывается файл так же последовательно от первой записи к последней. Для двоичных файлов с записями фиксированной длины применяется метод произвольного доступа, при котором можно обрабатывать отдельную запись: прочитать, изменить и записать на старое место.

Для применения в программе двоичного файла необходимо определить поток, поддерживающий двоичные операции.

Параметр ios::binary – открытие двоичного потока

Запись данных в двоичный файл обеспечивает метод write

Формат метода:

ostream& write(const char \*buf, streamsize num);

Из двоичного файла можно читать данные блоками по несколько байт. Размер блока должен быть к моменту чтения определен. Определение потока с операциями чтения данных двоичного файла - это объект типа ofstream. Чтение данных из двоичного файла выполняет метод класса istream.

Формат метода read():

istream& read(char \*buf, streamsize num);

1. Структура записи по условию варианта:

struct Product {

char name[20];

char manufacturer[20];

char country[20];

int price;

int code;

};

1. Для перевода данных из текстового файла в бинарный необходимо открыть поток на чтение из текстового файла и поток на запись в бинарный файл. Затем в цикле, пока поток чтения не достигнет конца, считываем значения из текстового файла и записываем с помощью метода write данные в бинарный файл.

Функция для перевода записей из текстового файла в бинарный файл:

void ConvertTextToBin(string bin\_file\_name, string txt\_file\_name) {

ifstream fin(txt\_file\_name, ios::in);

if (fin.is\_open()) {

ofstream fout(bin\_file\_name, ios::out | ios::binary);

Product product;

while (!fin.eof()) {

fin >> product.name >> product.code >> product.manufacturer >> product.price >> product.country;

fout.write((char\*)&product, sizeof(Product));

}

fout.close();

}

else {

cout << "Файла с таким именем не существует" << endl;

}

fin.close();

}

1. Для перевода данных из бинарного файла в текстовый необходимо открыть поток на чтение из текстового файла и поток на запись в бинарный файл. Затем в цикле, пока поток чтения считывает данные, записываем результат в текстовый файл.

Функция для перевода записей из бинарного файла в текстовый:

void ConvertBinToText(string bin\_file\_name, string txt\_file\_name) {

ifstream fin;

fin.open(bin\_file\_name, ios::binary | ios::in);

if (fin.is\_open()) {

ofstream fout;

fout.open(txt\_file\_name, ios::out);

Product product;

while (fin.read((char\*)&product, sizeof(Product))) {

fout << product.name << " " << product.code << " " << product.manufacturer << " " << product.price << " " << product.country << "\n";

}

fout.close();

}

else {

cout << "Файла с таким именем не существует" << endl;

}

fin.close();

}

1. Для вывода данных из бинарного файла откроем поток для чтения и будем считывать, пока это возможно, значения в переменную типа структуры варианта, затем ее выводить.

Функция для вывода всех записей бинарного файла:

void OutputBinFile(string bin\_file\_name) {

ifstream fin(bin\_file\_name, ios::in | ios::binary);

Product product;

if (fin.is\_open()) {

while (fin.read((char\*)&product, sizeof(Product))) {

cout << product.name << " " << product.code << " " << product.manufacturer << " " << product.price << " " << product.country << "\n";

}

}

else {

cout << "Файла с таким именем не существует" << endl;

}

fin.close();

}

1. Для вывода структуры под индексом, необходимо написать функцию для проверки вводимого индекса. Открываем поток для чтения и прогоняем его по всем структурам в бинарном файле, подсчитывая их. Возвращаем результат сравнения общего количества чисел и вводимого индекса.

Функция для сравнения количества элементов и индекса:

bool CheckIndex(string bin\_file\_name, int index) {

ifstream fin;

fin.open(bin\_file\_name, ios::binary | ios::in);

Product temp;

int count = 0;

while (fin.read((char\*)&temp, sizeof(Product))){

count++;

}

fin.close();

if (count > index) {

return true;

}

else {

return false;

}

}

Для получения структуры с помощью прямого доступа откроем поток для чтения, затем, если результат функции CheckIndex возвращает false, возвращаем ошибку, в ином случае с помощью команды seekg получаем адрес нужного элемента и с помощью команды read считываем ее.

Функция для получения записи с помощью прямого доступа:

string GetProduct(string bin\_file\_name, int index) {

ifstream fin;

fin.open(bin\_file\_name, ios::binary | ios::in);

string result = "";

if (fin.is\_open()) {

if (!CheckIndex(bin\_file\_name, index)) {

return "Индекс превосходит количество элементов";

}

Product product;

fin.seekg(index \* sizeof(Product), ios::beg);

fin.read((char\*)&product, sizeof(Product));

result = product.name;

result += " ";

result += to\_string(product.code);

result += " ";

result += product.manufacturer;

result += " ";

result += to\_string(product.price);

result += " ";

result += product.country;

}

else {

result = "Файла с таким именем не существует";

}

return result;

fin.close();

}

1. Для удаления записи открываем поток на чтение и запись, проверяем на возможность считывания структуры, затем считываем структуру последней записи в product с помощью команды seekg и read. Записываем пустую структуру empty в последнюю позицию с помощью команды seekg и write. Затем записываем в структуру по индексу переменную product.

Функция для удаления записи:

void DeleteRecord(string name, int index){

Product product;

Product empty;

fstream file(name, ios\_base::binary | ios::in | ios::out);

if (file.is\_open()) {

if (!CheckIndex(name, index)) {

cout << "Индекс превышает количество записей" << endl;

return;

}

file.seekg((-1) \* (int)sizeof(Product), ios::end);

file.read((char\*)&product, sizeof(Product));

file.seekg((-1) \* (int)sizeof(Product), ios::end);

file.write((char\*)&empty, sizeof(Product));

file.seekg((index) \* sizeof(Product), ios::beg);

file.write((char\*)&product, sizeof(Product));

}

else {

cout << "Файла с таким именем не существует\n";

return;

}

if (file.good()) {

file.close();

}

}

1. Для решения первого дополнительного задания открываем два потока для чтения и записи, пока не дошли до конца файла. Проверяем равны ли введенная страна с страной из файла. Если да, то записываем поля этой записи кроме страны, иначе идем дальше.

Функция для решения первой дополнительной задачи варианта:

void AdditionalTask1(string bin\_file\_name, string new\_file\_name, char country[20]) {

ifstream file(bin\_file\_name, ios\_base::binary);

if (!file) {

cout << "Файла с таким именем не существует\n";

return;

}

ofstream products(new\_file\_name, ios\_base::binary);

Product product;

file.read((char\*)&product, sizeof(Product));

products << country << "\n";

string temp = country;

while (!file.eof()) {

if (checkCountry(product.country, country, temp.size())) {

products << product.name << " " << product.code << " " << product.manufacturer << " " << product.price << "\n";

}

file.read((char\*)&product, sizeof(Product));

}

if (file.good()) {

file.close();

}

if (products.good()) {

products.close();

}

}

1. Для решения второй дополнительной задачи варианта открываем поток для добавления новой записи. Если файл успешно открылся, то вводим значения полей и записываем их в файл.

Функция для решения первой дополнительной задачи варианта:

void AdditionalTask2(string bin\_file\_name) {

ofstream binfout;

binfout.open(bin\_file\_name, ios::app | ios::binary);

Product product;

if (binfout.good()) {

cout << "Название продукта: "; cin >> product.name; cin.ignore(); cin.clear();

cout << "Код продукта: "; cin >> product.code; cin.ignore(); cin.clear();

cout << "Производитель продукта: "; cin >> product.manufacturer; cin.ignore(); cin.clear();

cout << "Цена продукта: "; cin >> product.price; cin.ignore(); cin.clear();

cout << "Страна-производитель продукта: "; cin >> product.country; cin.ignore(); cin.clear();

binfout.write((char\*)&product, sizeof(Product));

binfout.close();

}

else {

return;

}

}

Описание интерфейса программы: интерфейс () представляет собой меню из 8 пунктов. После каждого выбранного пункта будет необходим ввод имени бинарного файла. Если введено число не соответствующее номеру задания, то программа предложит повоторить ввод. При выполнении очередной задачи меню открывается заново.

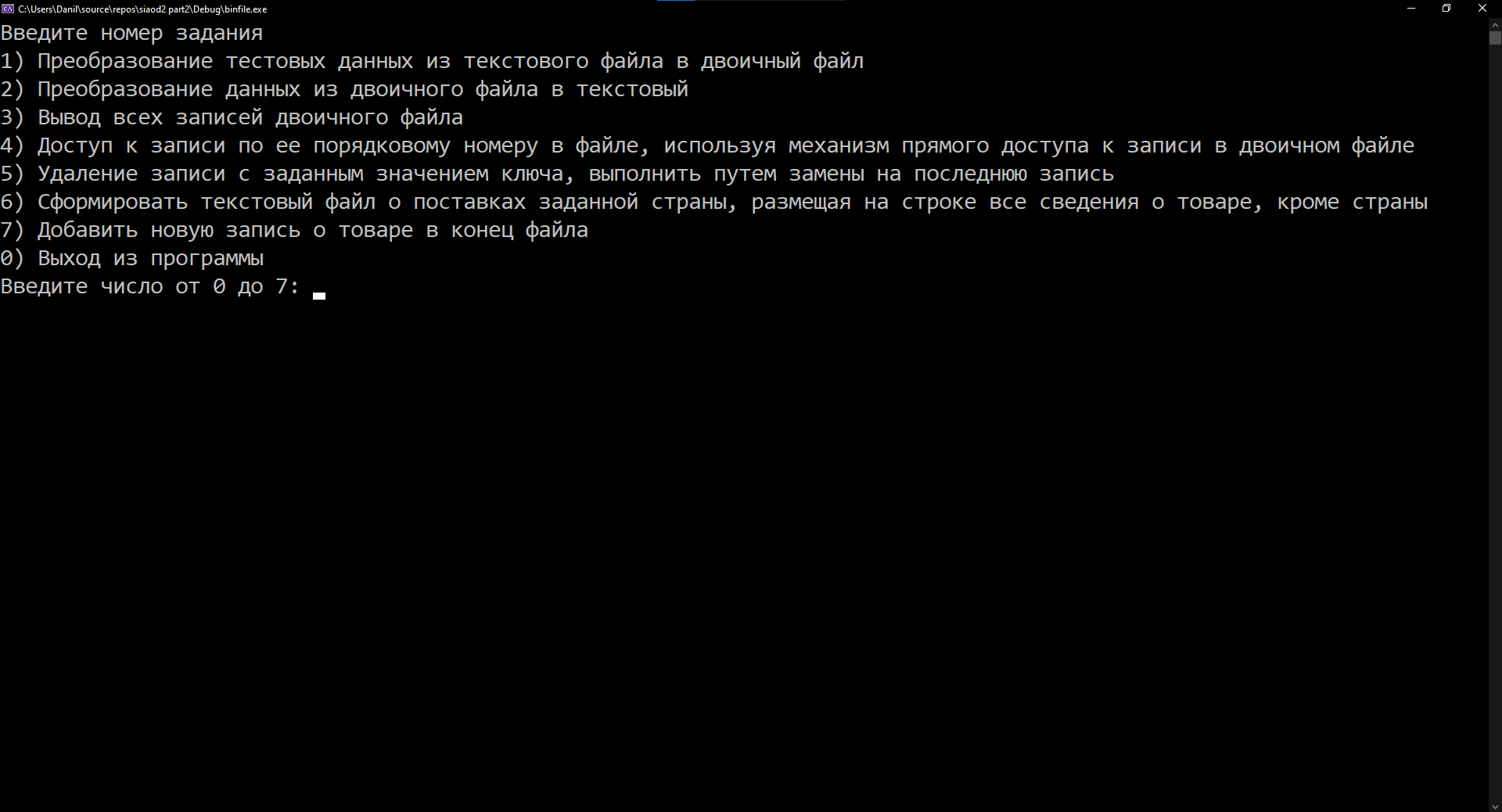


Рисунок 7 - Интерфейс программы для 2-ого задания

# **ТЕСТИРОВАНИЕ**

1. При тестировании первого задания (Рисунок 8) введем некорректный и корректный путь к текстовому файлу, если путь правильный, то программа создаст/откроет бинарный файл и перепишет туда данные файла (Рисунок 9), в ином случае выведется уведомление.
2. При тестировании второго задания (Рисунок 10) введем некорректный и корректный путь к бинарному файлу, если путь правильный, то программа создаст/откроет текстовый файл и перепишет туда данные файла (Рисунок 11), в ином случае выведется уведомление.
3. При тестировании третьего задания (Рисунок 12) введем некорректный и корректный путь к бинарному файлу, если путь правильный, то программа выведет все структуры, содержащиеся в файле, в ином случае выведется уведомление.
4. При тестировании четвертого задания (Рисунок 13) введем некорректный и корректный путь к бинарному файлу, если путь правильный, то продолжит работать, в ином случае выведется уведомление. При продолжении работы программы вводится индекс записи, если индекс превышает или равен количеству элементов, выведется уведомление, в ином случае программа сработает корректно.
5. При тестировании пятого задания (Рисунок 14) введем некорректный и корректный путь к бинарному файлу, если путь правильный, то продолжит работать, в ином случае выведется уведомление. При продолжении работы программы вводится индекс удаляемой записи, если индекс превышает количество записей – выводится уведомление.
6. При тестировании шестого задания (Рисунок 15) вводится страна, поставки которой мы хотим получить в новом файле. Введем некорректный и корректный путь к бинарному файлу, если путь правильный, то продолжит работать, в ином случае выведется уведомление. При продолжении работы программы создастся новый бинарный файл с информацией о поставках введенной страны (Рисунок 16).
7. При тестировании седьмого задания (Рисунок 17) введем некорректный и корректный путь к бинарному файлу, если путь правильный, то продолжит работать, в ином случае выведется уведомление. При продолжении работы программы вводим поля для новой записи, далее эта запись допишется в введенный ранее нами файл (Рисунок 18).

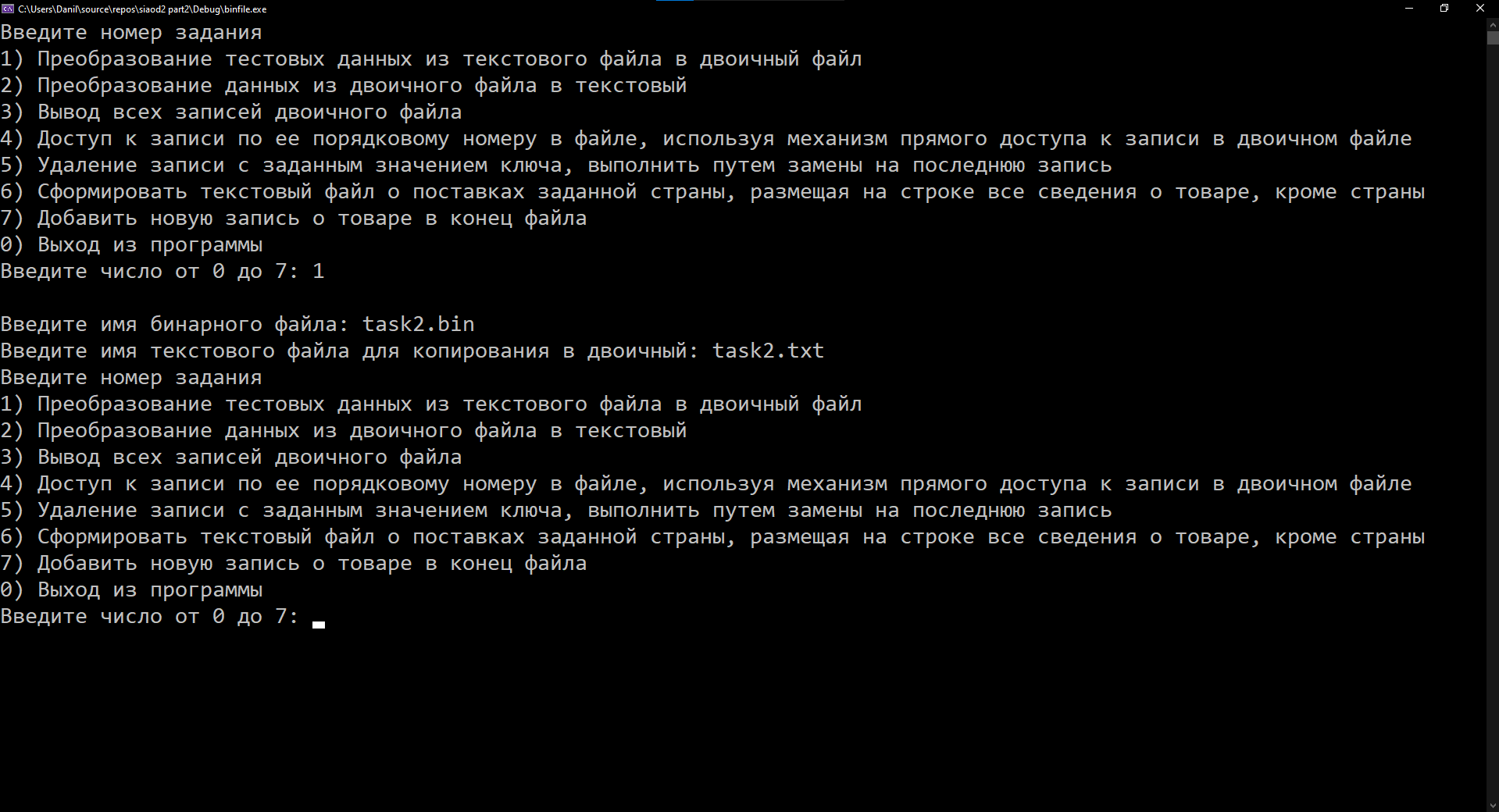


Рисунок 8 - Тестирование 1-ого задания

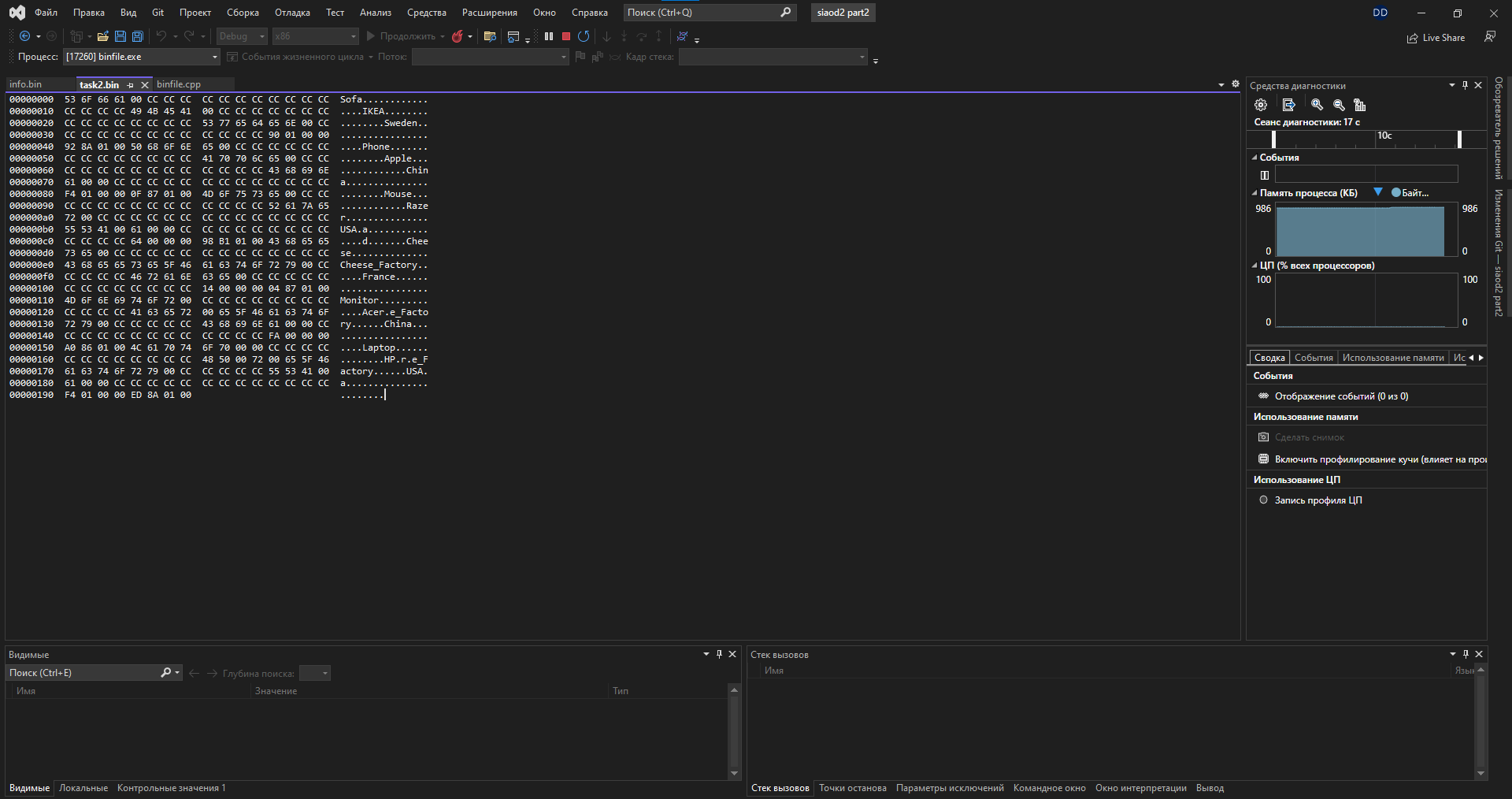


Рисунок 9 - Созданный файл в результате выполнения 1-ого задания

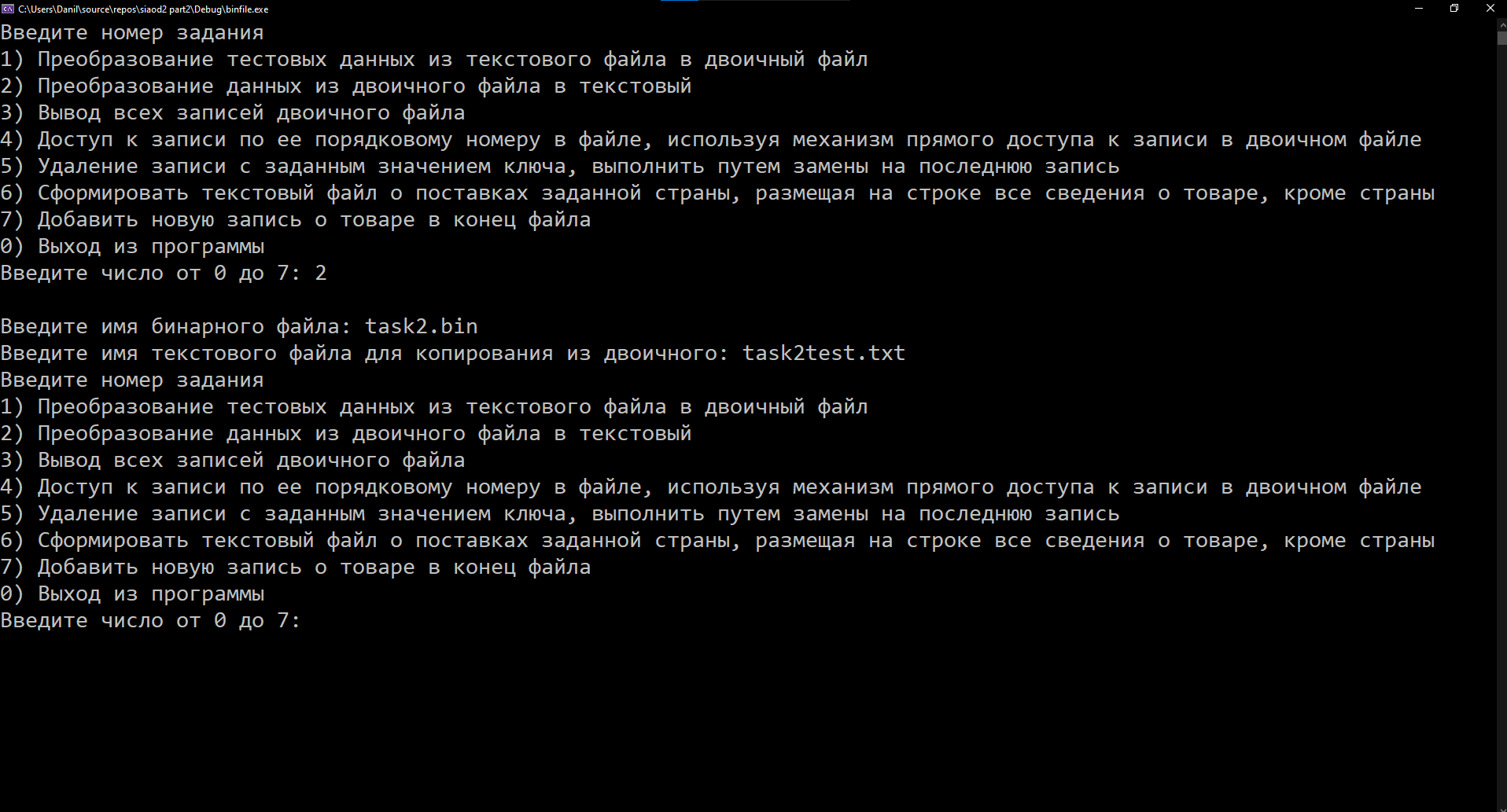


Рисунок 10 - Тестирование 2-ого задания

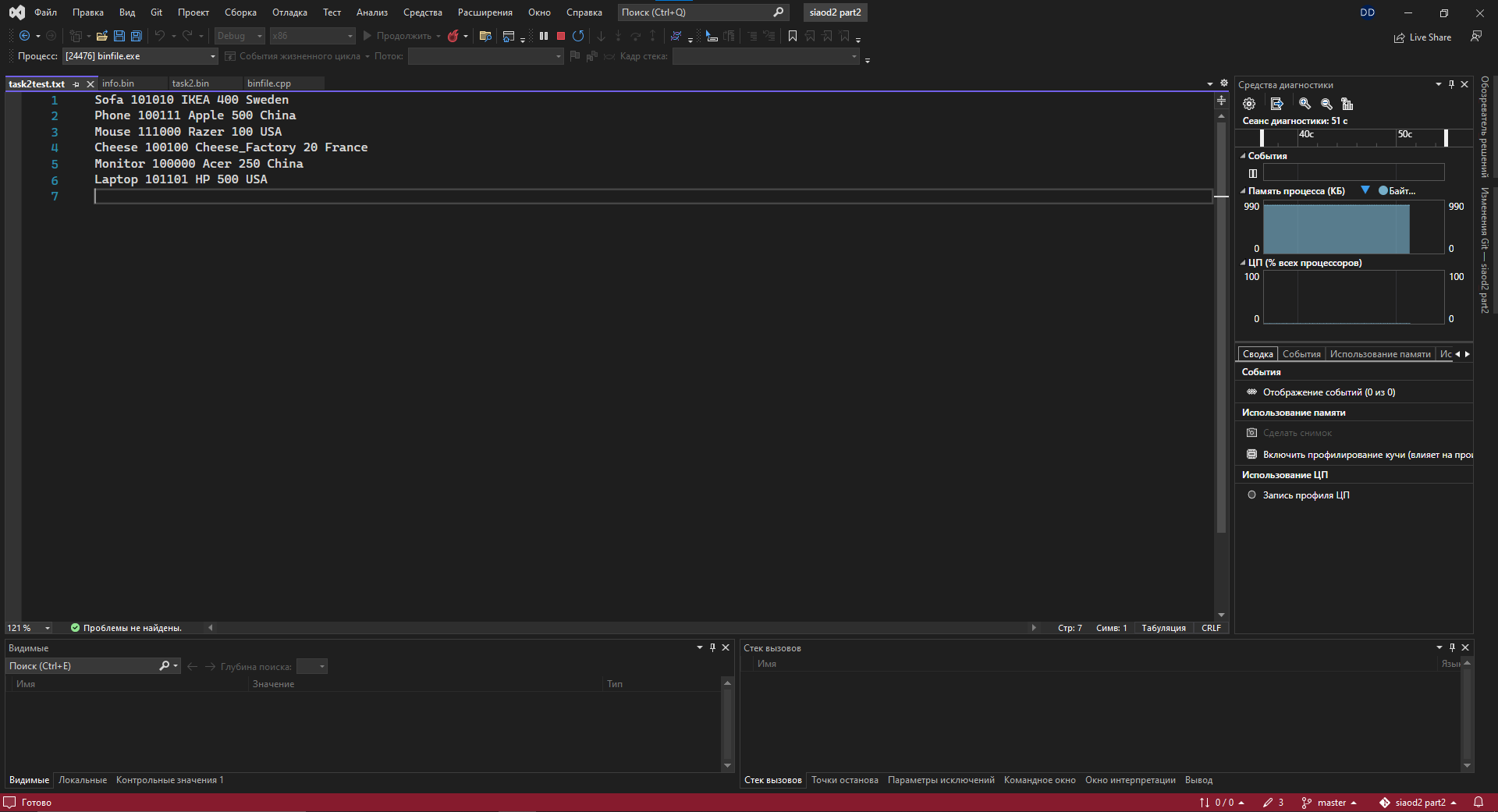


Рисунок 11 - Созданный файл в результате выполнения 2-ого задания

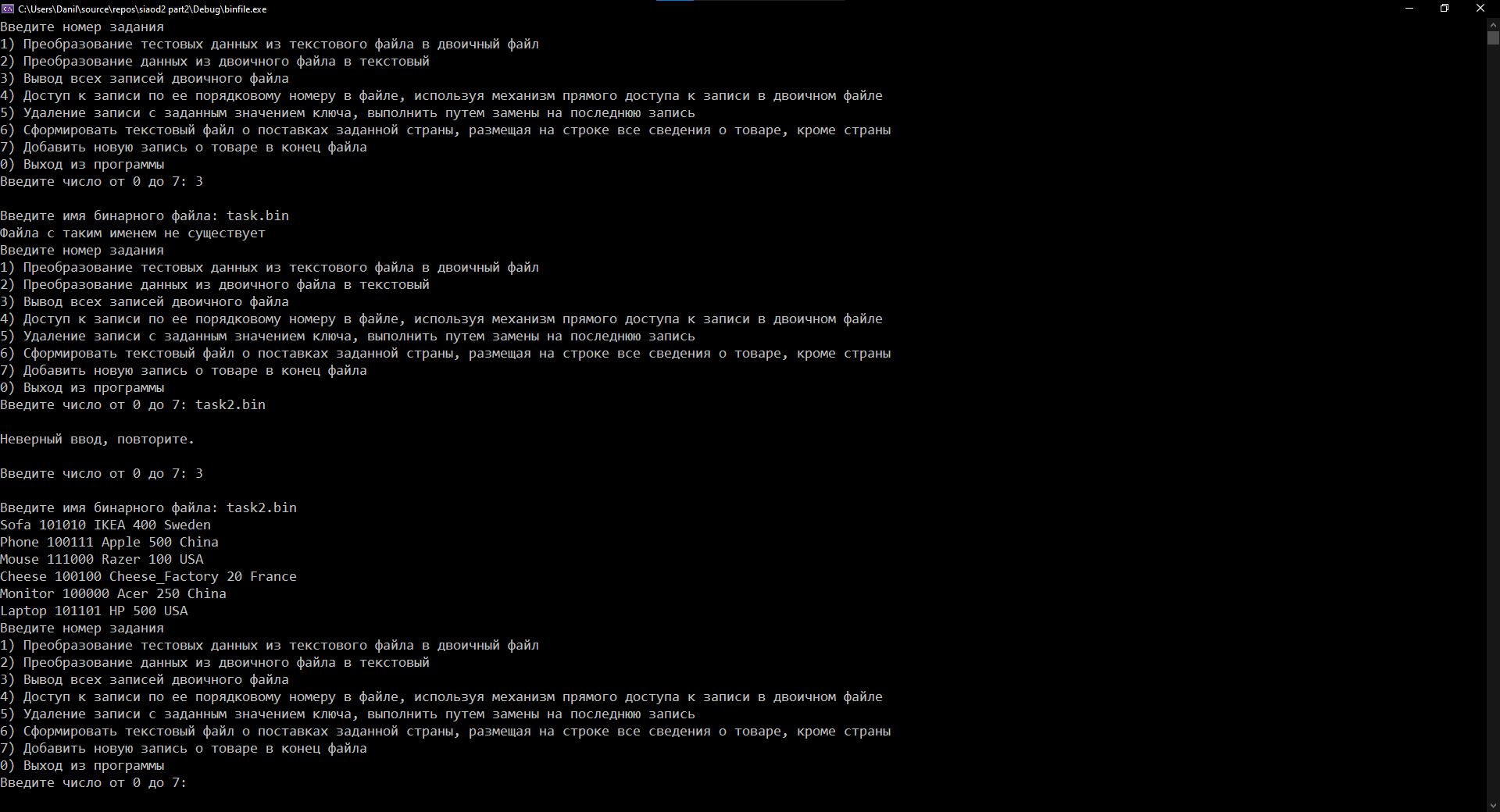


Рисунок 12 - Тестирование 3-ого задания

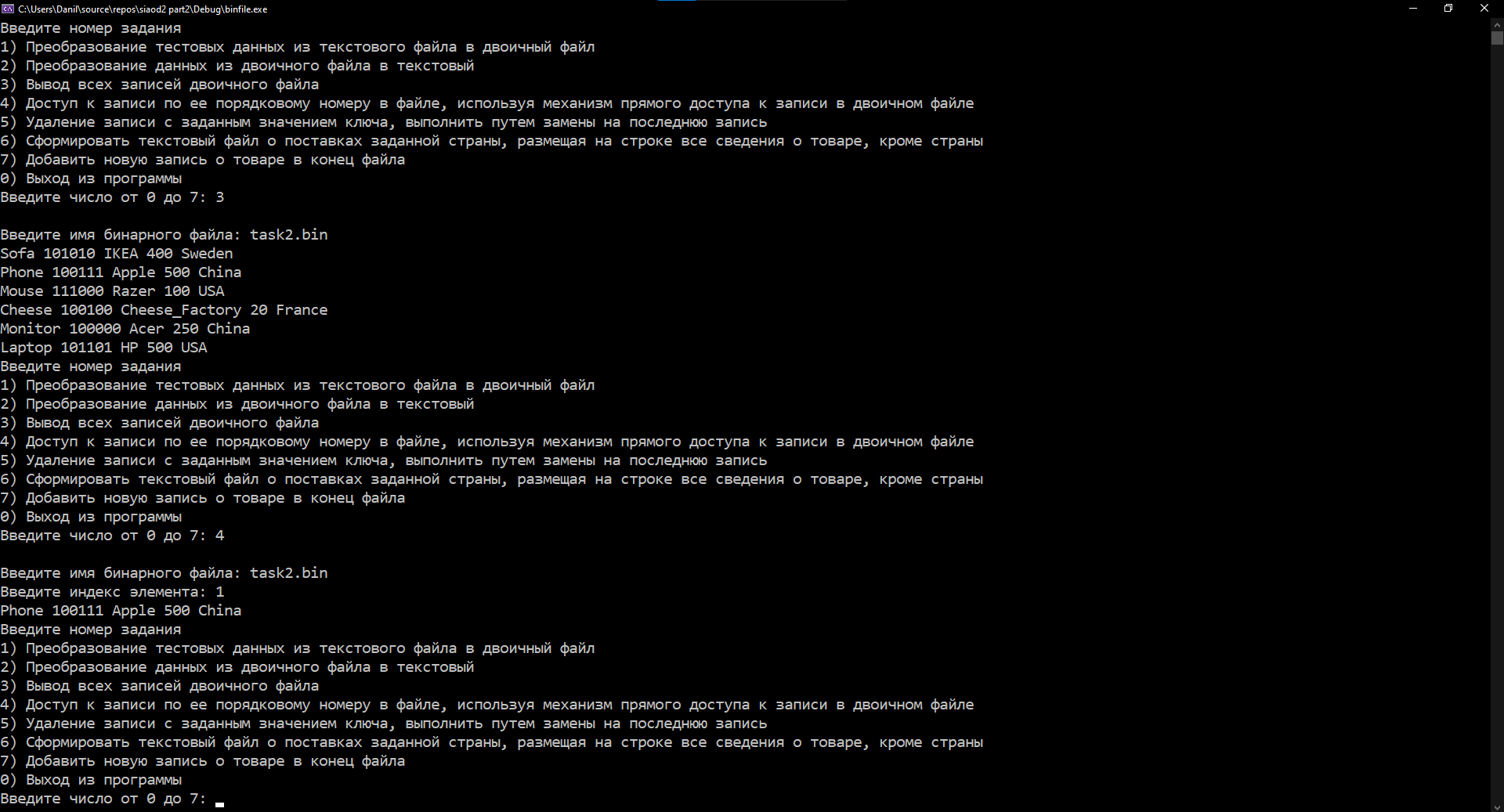


Рисунок 13 - Тестирование 4-ого задания

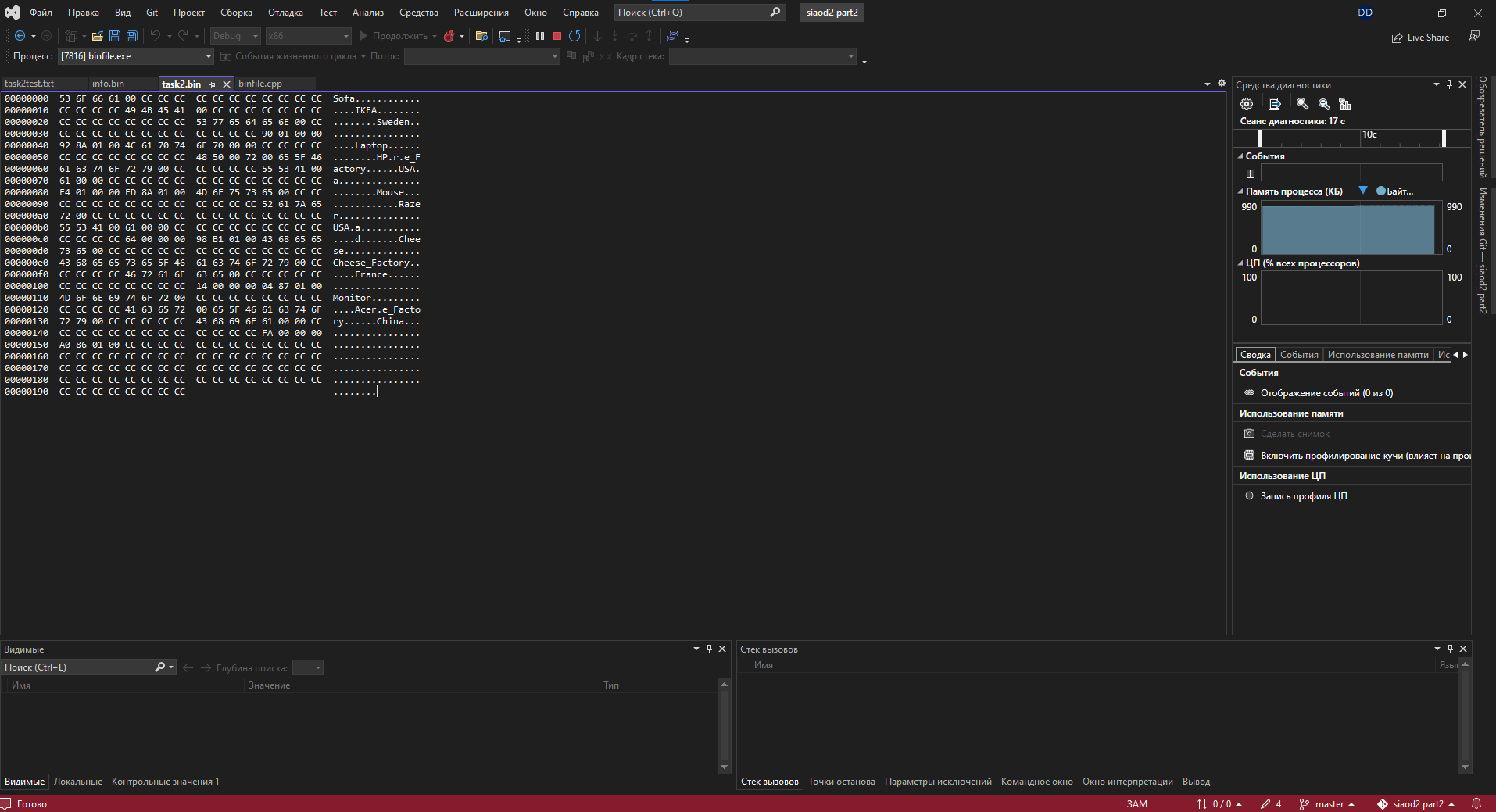


Рисунок 14 - Тестирование 5-ого задания

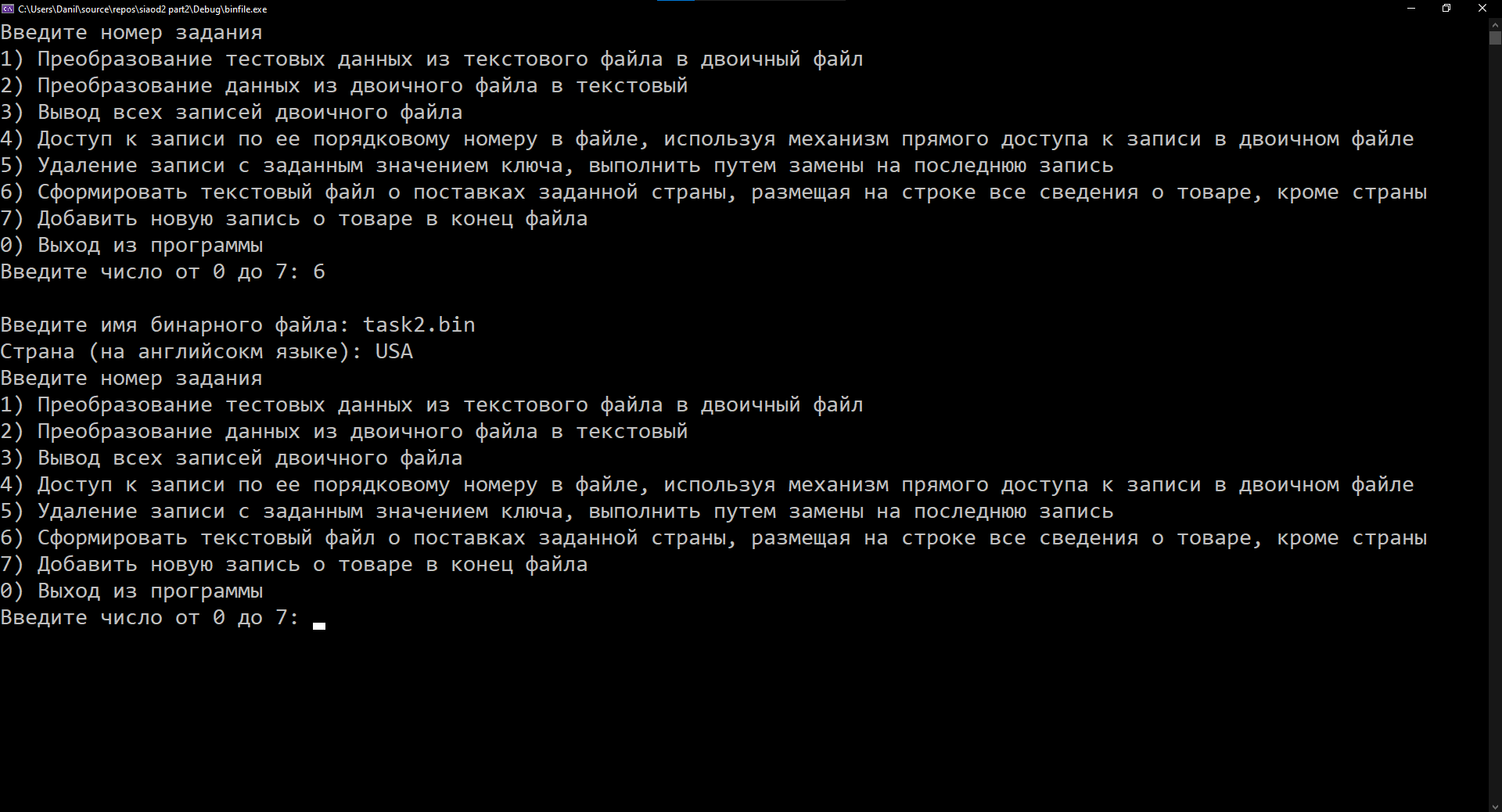


Рисунок 15 - Тестирование 6-ого задания

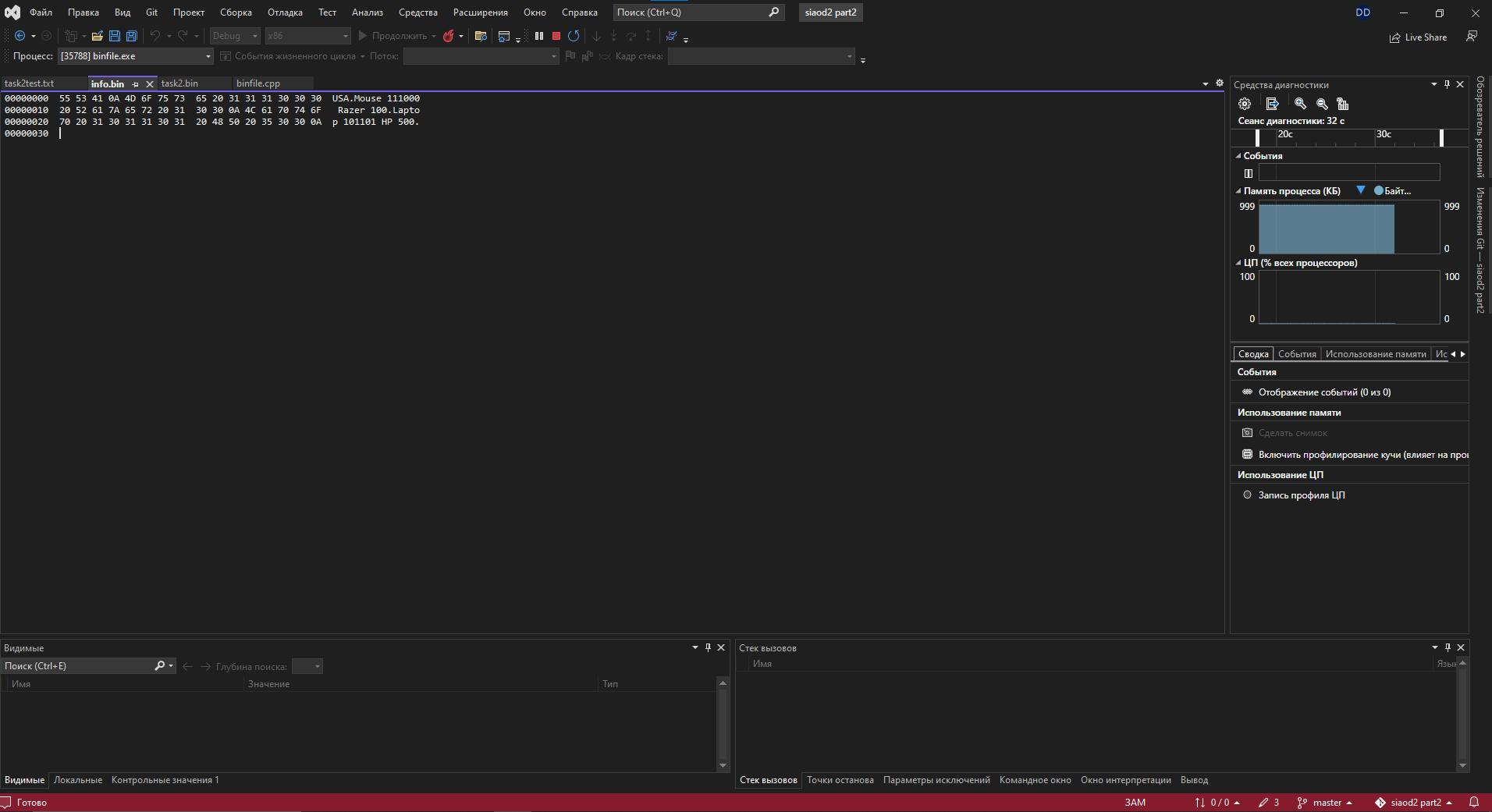


Рисунок 16 - Созданный файл в результате выполнения 6-ого задания

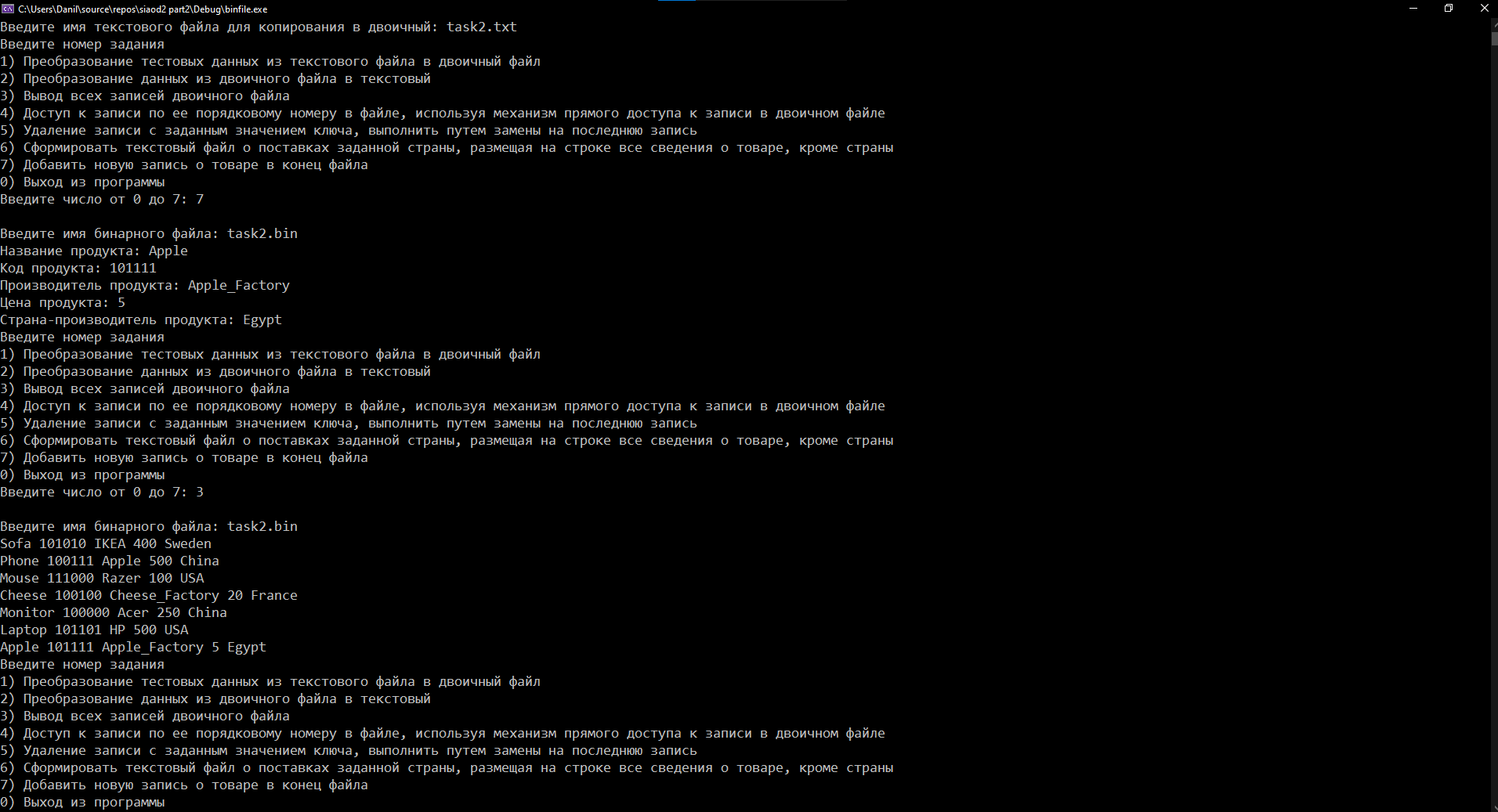


Рисунок 17 - Тестирование 7-ого задания

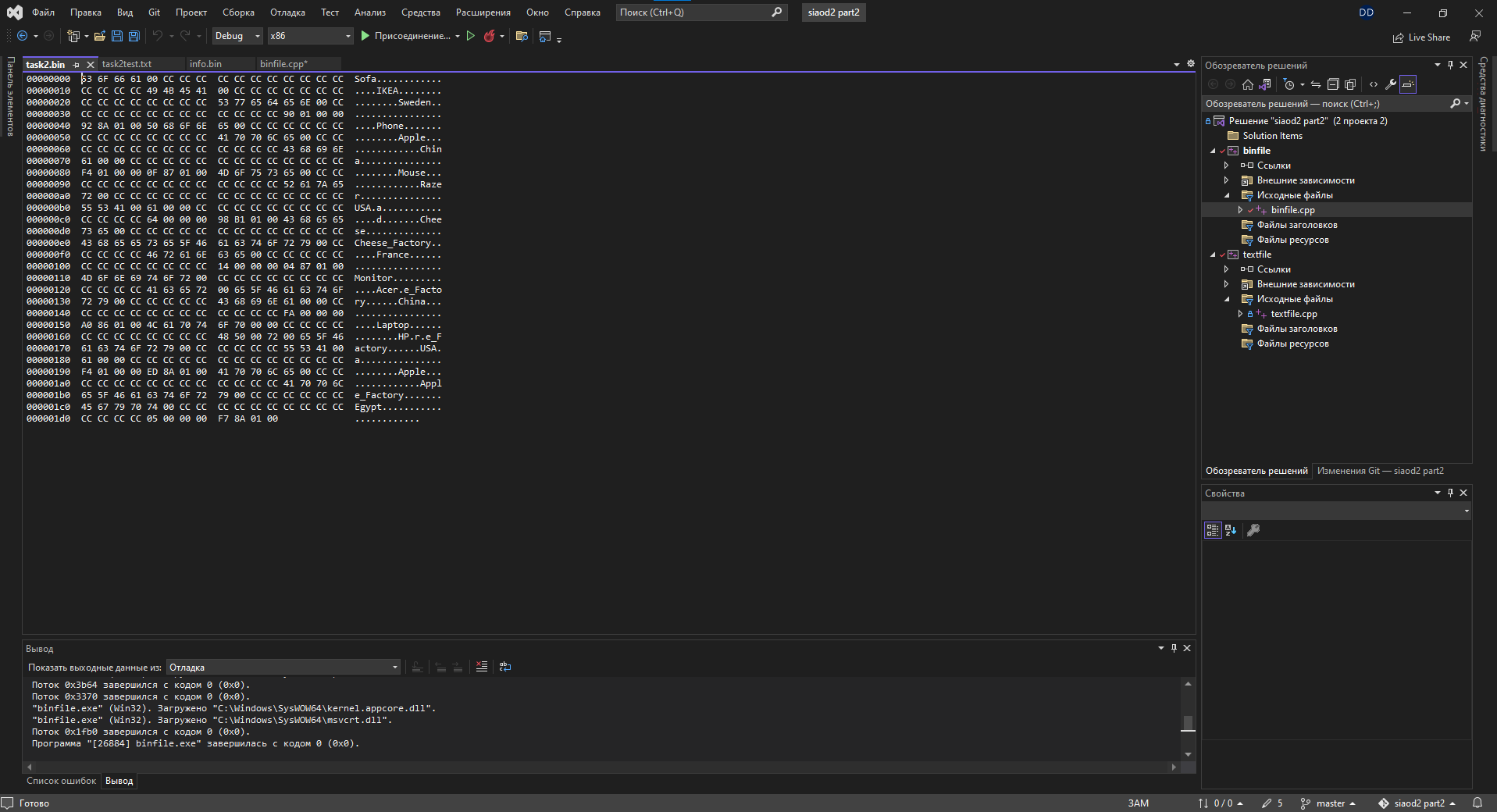


Рисунок 18 - Измененный файл в результате выполнения 7-ого задания

# **ПОЛНЫЙ КОД ПРОГРАММЫ 2-ОГО ЗАДАНИЯ**

#include <string.h>

#include <string>

#include <iostream>

#include <fstream>

#define \_CRT\_SECURE\_NO\_WARNINGS

using namespace std;

struct Product {

char name[20];

char manufacturer[20];

char country[20];

int price;

int code;

};

Product product;

int inputMenu(int& number) {

cout << "Введите номер задания" << endl;

cout << "1) Преобразование тестовых данных из текстового файла в двоичный файл" << endl;

cout << "2) Преобразование данных из двоичного файла в текстовый" << endl;

cout << "3) Вывод всех записей двоичного файла" << endl;

cout << "4) Доступ к записи по ее порядковому номеру в файле, используя механизм прямого доступа к записи в двоичном файле" << endl;

cout << "5) Удаление записи с заданным значением ключа, выполнить путем замены на последнюю запись" << endl;

cout << "6) Сформировать текстовый файл о поставках заданной страны, размещая на строке все сведения о товаре, кроме страны" << endl;

cout << "7) Добавить новую запись о товаре в конец файла" << endl;

cout << "0) Выход из программы" << endl;

while (1) {

cout << "Введите число от 0 до 7: ";

if ((cin >> number).good() && (number >= 0) && (number <= 7)) {

return number;

}

if (cin.fail()) {

cin.clear();

cout << endl << "Неверный ввод, повторите." << endl << endl;

}

else {

cout << endl << "Число вне допустимого диапазона значений. Повторите ввод." << endl << endl;

}

cin.ignore(100, '\n');

}

}

void ConvertTextToBin(string bin\_file\_name, string txt\_file\_name) {

ifstream fin(txt\_file\_name, ios::in);

if (fin.is\_open()) {

ofstream fout(bin\_file\_name, ios::out | ios::binary);

Product product;

while (!fin.eof()) {

fin >> product.name >> product.code >> product.manufacturer >> product.price >> product.country;

fout.write((char\*)&product, sizeof(Product));

}

fout.close();

}

else {

cout << "Файла с таким именем не существует" << endl;

}

fin.close();

}

void ConvertBinToText(string bin\_file\_name, string txt\_file\_name) {

ifstream fin;

fin.open(bin\_file\_name, ios::binary | ios::in);

if (fin.is\_open()) {

ofstream fout;

fout.open(txt\_file\_name, ios::out);

Product product;

while (fin.read((char\*)&product, sizeof(Product))) {

fout << product.name << " " << product.code << " " << product.manufacturer << " " << product.price << " " << product.country << "\n";

}

fout.close();

}

else {

cout << "Файла с таким именем не существует" << endl;

}

fin.close();

}

void OutputBinFile(string bin\_file\_name) {

ifstream fin(bin\_file\_name, ios::in | ios::binary);

Product product;

if (fin.is\_open()) {

while (fin.read((char\*)&product, sizeof(Product))) {

cout << product.name << " " << product.code << " " << product.manufacturer << " " << product.price << " " << product.country << "\n";

}

}

else {

cout << "Файла с таким именем не существует" << endl;

}

fin.close();

}

bool CheckIndex(string bin\_file\_name, int index) {

ifstream fin;

fin.open(bin\_file\_name, ios::binary | ios::in);

Product temp;

int count = 0;

while (fin.read((char\*)&temp, sizeof(Product))){

count++;

}

fin.close();

if (count > index) {

return true;

}

else {

return false;

}

}

string GetProduct(string bin\_file\_name, int index) {

ifstream fin;

fin.open(bin\_file\_name, ios::binary | ios::in);

string result = "";

if (fin.is\_open()) {

if (!CheckIndex(bin\_file\_name, index)) {

return "Индекс превосходит количество элементов";

}

Product product;

fin.seekg(index \* sizeof(Product), ios::beg);

fin.read((char\*)&product, sizeof(Product));

result = product.name;

result += " ";

result += to\_string(product.code);

result += " ";

result += product.manufacturer;

result += " ";

result += to\_string(product.price);

result += " ";

result += product.country;

}

else {

result = "Файла с таким именем не существует";

}

return result;

fin.close();

}

void DeleteRecord(string name, int index){

Product product;

Product empty;

fstream file(name, ios\_base::binary | ios::in | ios::out);

if (file.is\_open()) {

if (!CheckIndex(name, index)) {

cout << "Индекс превышает количество записей" << endl;

return;

}

file.seekg((-1) \* (int)sizeof(Product), ios::end);

file.read((char\*)&product, sizeof(Product));

file.seekg((-1) \* (int)sizeof(Product), ios::end);

file.write((char\*)&empty, sizeof(Product));

file.seekg((index) \* sizeof(Product), ios::beg);

file.write((char\*)&product, sizeof(Product));

}

else {

cout << "Файла с таким именем не существует\n";

return;

}

if (file.good()) {

file.close();

}

}

bool checkCountry(char\* m1, char\* m2,int size) {

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

if (m1[i] != m2[i]) {

return false;

}

}

return true;

}

void AdditionalTask1(string bin\_file\_name, string new\_file\_name, char country[20]) {

ifstream file(bin\_file\_name, ios\_base::binary);

if (!file) {

cout << "Файла с таким именем не существует\n";

return;

}

ofstream products(new\_file\_name, ios\_base::binary);

Product product;

file.read((char\*)&product, sizeof(Product));

products << country << "\n";

string temp = country;

while (!file.eof()) {

if (checkCountry(product.country, country, temp.size())) {

products << product.name << " " << product.code << " " << product.manufacturer << " " << product.price << "\n";

}

file.read((char\*)&product, sizeof(Product));

}

if (file.good()) {

file.close();

}

if (products.good()) {

products.close();

}

}

void AdditionalTask2(string bin\_file\_name) {

ofstream binfout;

binfout.open(bin\_file\_name, ios::app | ios::binary);

Product product;

if (binfout.good()) {

cout << "Название продукта: "; cin >> product.name; cin.ignore(); cin.clear();

cout << "Код продукта: "; cin >> product.code; cin.ignore(); cin.clear();

cout << "Производитель продукта: "; cin >> product.manufacturer; cin.ignore(); cin.clear();

cout << "Цена продукта: "; cin >> product.price; cin.ignore(); cin.clear();

cout << "Страна-производитель продукта: "; cin >> product.country; cin.ignore(); cin.clear();

binfout.write((char\*)&product, sizeof(Product));

binfout.close();

}

else {

cout << "Файла с таким именем не существует\n";

return;

}

}

int main() {

system("chcp 1251 > null");

string bin\_file\_name;

int menu;

while (true) {

inputMenu(menu);

if (menu == 0) {

cout << "Задание не выбрано\nВыход из программы..." << endl;

break;

}

cout << endl;

switch (menu) {

case 1: {

cout << "Введите имя бинарного файла: ";

cin >> bin\_file\_name;

cin.ignore();

cin.clear();

string txt\_file\_name;

cout << "Введите имя текстового файла для копирования в двоичный: ";

cin >> txt\_file\_name;

ConvertTextToBin(bin\_file\_name, txt\_file\_name);

break;

}

case 2: {

cout << "Введите имя бинарного файла: ";

cin >> bin\_file\_name;

cin.ignore();

cin.clear();

string txt\_file\_name;

cout << "Введите имя текстового файла для копирования из двоичного: ";

cin >> txt\_file\_name;

ConvertBinToText(bin\_file\_name, txt\_file\_name);

break;

}

case 3: {

cout << "Введите имя бинарного файла: ";

cin >> bin\_file\_name;

cin.ignore();

cin.clear();

OutputBinFile(bin\_file\_name);

break;

}

case 4: {

cout << "Введите имя бинарного файла: ";

cin >> bin\_file\_name;

cin.ignore();

cin.clear();

int index;

cout << "Введите индекс элемента: ";

cin >> index;

cout << GetProduct(bin\_file\_name, index) << endl;

break;

}

case 5: {

cout << "Введите имя бинарного файла: ";

cin >> bin\_file\_name;

cin.ignore();

cin.clear();

int index;

cout << "Введите индекс элемента, который надо удалить: ";

cin >> index;

DeleteRecord(bin\_file\_name, index);

break;

}

case 6: {

cout << "Введите имя бинарного файла: ";

cin >> bin\_file\_name;

cin.ignore();

cin.clear();

char country[20]; cout << "Страна (на английском языке): ";

cin >> country;

cin.ignore();

cin.clear();

AdditionalTask1(bin\_file\_name, "info.bin", country);

break;

}

case 7: {

cout << "Введите имя бинарного файла: ";

cin >> bin\_file\_name;

cin.ignore();

cin.clear();

AdditionalTask2(bin\_file\_name);

break;

}

default: {

cout << "Задание не выбрано\nВыход из программы..." << endl;

break;

}

}

}

}

//Тестовые данные для 7-ого задания

//Apple

//101111

//Apple\_Factory

//5

//Egypt

# **ВЫВОД**

В результате выполнения практической работы были получены навыки и опыт работы с текстовыми и бинарными файлами.