ФГБОУ ВО

«Уфимский государственный авиационный технический университет»

Кафедра ТК

**ОТЧЕТ**

**по лабораторной работе № 10**

по дисциплине «Методы оптимизации»

**Тема: «Динамический полиморфизм (позднее связывание).**

**Многофайловая программа.»**

**Вариант № 28**

Выполнила: студентка гр. ИВТ-421

Самсонова Е.О.

Проверил: доцент каф. ТК

Хасанов А.Ю.

Уфа 2022

Вариант 28

Спроектировать структуру *PERSON*, содержащую поля: фамилия и инициалы, район, адрес, телефон.

Спроектировать структуру *ABONENT*, содержащую поле типа *PERSON* и дополнительно поля: номер договора, дата заключения, абонентская плата помесячно.

Файл содержит информацию об абонентах кабельного телевидения. Требуется спроектировать класс, позволяющий обрабатывать эти данные. В классе должно быть предусмотрено, как минимум, выполнение следующих видов обработки:

1. Загрузка из файла;
2. Сохранение результатов обработки в файлах;
3. Добавление записи;
4. Удаление записи;
5. Вывод на экран дисплея данных и результатов обработки;
6. Алфавитная сортировка по Ф.И.О.;
7. Сортировка по дате заключения договора;
8. Числовая сортировка по номеру договора;
9. Перечень абонентских плат с указанием числа абонентов с такой платой (результат отсортирован по численности);
10. Перечень абонентских плат с указанием числа абонентов с такой платой (результат отсортирован по абонентской плате);
11. Поиск по району (результат отсортирован по дате заключения);
12. Поиск по району (результат отсортирован по ФИО).

Задание

1. Программу лабораторной работы № 9 реализовать с использованием позднего связывания.
2. В основной программе работы с объектами классов реализовать с помощью указателей.
3. В основной программе в ряде пунктов меню должно быть использовано позднее связывание.
4. Деструкторы следует определить виртуальными.
5. Перегрузить операции присваивания с помощью виртуальных методов.
6. В меню должны быть предусмотрены пункты проверяющие работу перегруженных операций присваивания.
7. Описание каждого из классов, описывающих работу с динамическими массивами структур, поместить в заголовочные файлы с расширением «\*.h» (много файловая программа – вторая версия).
8. Реализацию классов поместить в отдельные файлы с расширением «\*.cpp» (много файловая программа – вторая версия).
9. Описание и реализацию классов иерархии исключений поместить в отдельный файл с расширением «\*.h» (много файловая программа).
10. Основную программу поместить в отдельном файле с расширением «\*.cpp» (много файловая программа – вторая версия).
11. Необходимо продемонстрировать работоспособность программы для всех предусмотренных особых ситуаций.
12. Продемонстрировать работу виртуальных деструкторов.
13. Основная программа должна быть реализована в виде меню и демонстрировать работу всех методов класса.

**Разметка таблицы**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Номер договора | Дата заключения | | | Абонентская плата | Фамилия и инициалы | Район | Адрес | Телефон |
| День | Месяц | Год |
|  |  |  |  |  |  |  |  |  |  |

Код программы

Main.cpp

#define NOMINMAX

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

#include "struct.h"

#include "exception.h"

#include "masa.h"

#include "masb.h"

#include "masc.h"

#include "print.h"

using namespace std;

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "");

masA\* pa, \* pb, \* pc;

masC\* pd;

int j;

char ch;

// Выделение памяти для объектов основного \*pa(\*pd) и рабочих \*pb,\*pc

try {

pa = pd = new masC;

pb = new masC;

pc = new masC;

}

catch (bad\_alloc)

{

cout << "Ошибка: нет памяти для объектов\n";

\_getch(); return;

}

while (1)

{

try {

system("cls"); //clrscr(); очистка экрана

cout << "1. Загрузка исходного массива из файла\n";

cout << "2. Вывод исходного массива на экран\n";

cout << "3. Сохранение исходного массива в текстовом файле\n";

cout << "4. Добавление записи в исходный массив\n";

cout << "5. Удаление записи из исходного массива\n";

cout << "6. Алфавитная сортировка по Ф.И.О. исходного массива\n";

cout << "7. Сортировка по дате заключения договора исходного массива\n";

cout << "8. Числовая сортировка по номеру договора исходного массива\n";

cout << "9. Формирование перечня абонентских плат (1-ый выходной массив)\n";

cout << "10. Вывод перечня абонентских плат на экран\n";

cout << "11. Вывод перечня абонентских плат в текстовый файл\n";

cout << "12. Сортировка перечня абонентских плат по стоимости платы \n";

cout << "13. Сортировка перечня абонентских плат по количеству абонентов в каждом типе платы \n";

cout << "14. Поиск абонентов по району (2-ой выходной массив)\n";

cout << "15. Вывод результатов поиска абонентов на экран\n";

cout << "16. Вывод результатов поиска абонентов в текстовый файл\n";

cout << "17. Сортировка поиска абонентов по районам по Ф.И.О.\n";

cout << "18. Сортировка поиска абонентов по районам по дате заключения\n";

cout << "19. Проверка конструктора копирования\n";

cout << "20.Проверка обычной операции присваивания\n";

cout << "21.Проверка виртуальной операции присв-я \n";

cout << "22.Выход из программы\n\n";

cout << "Ваш выбор (1-21): ";

try

{

if (char(cin.peek()) == '\n')

cin.ignore();

if (cin.fail())

{

cin.clear();

cin.ignore(32767, '\n');

}

cin >> j;

if (cin.fail())

{

throw Err("Это не пункт меню", "main()");

}

}

catch (Err& ex)

{

throw;

}

// Реализация выбранного пункта меню

switch (j)

{

case 1: pa->inputFile(); break;

case 2: pa->masA::outputMas(); break;

case 3: pa->masA::outputFile(); break;

case 4: pa->addAbonent(); break;

case 5: pa->deleteAbonent(); break;

case 6: pa->sort1(); break;

case 7: pa->sort2(); break;

case 8: pa->masA::sort3(); break;

case 9: ((masB\*)pa)->masB::makePerech(); break;

case 10:((masB\*)pa)->masB::outputMas(); break;

case 11: {((masB\*)pa)->masB::outputFile(); break;

case 12: ((masB\*)pa)->masB::sort1(); break;

case 13: ((masB\*)pa)->masB::sort3(); break;

case 14: ((masC\*)pa)->findRegion(); break;

case 15: pa->outputMas(); break;

case 16: pa->outputFile(); break;

case 17: ((masC\*)pa)->sort1(); break;

case 18:((masC\*)pa)->sort2(); break;

case 19: { masC c(\*pd); // конструктор копирования

printA(pd); printA(&c);

printB(pd); printB(&c);

printC(pd); printC(&c);

} break;

case 20: { masC a, b;

a = b = \*pd; // каскадное обычное присваивание

printA(pd); printA(&b); printA(&a);

printB(pd); printB(&b); printB(&a);

printC(pd); printC(&b); printC(&a);

} break;

case 21: { masC\* pb2, \* pc2;

\*pc = \*pb = \*pa; // каскадное виртуальное присваивание

pb2 = (masC\*)pb; // понижающее присваивание указателей

pc2 = (masC\*)pc; // понижающее присваивание указателей

printA(pd); printA(pb2); printA(pc2);

printB(pd); printB(pb2); printB(pc2);

printC(pd); printC(pb2); printC(pc2);

} break;

case 22: cout << "Конец работы\n";

delete pa;

delete pb;

delete pc;

system("pause"); return;

default: throw Err("Нет такого пункта в меню", "main()");

}

}// Конец внешнего блока try

}

catch (Err& ex) { ex.ErrOutput(); }

catch (...)

{

cout << "Непредусмотренное исключение\n"; \_getch();

}

}

}

Struct.h

#ifndef \_struct\_

#define \_struct\_

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

using namespace std;

class masB;

class masC;

struct person {

string name;

string region;

string adress;

string phone;

};

struct dat { int d, m, y; };

struct abonent {

int number;

int plata;

person pers;

dat datzakl;

};

struct PLATAA

{

int plata;

int count;

};

#endif

Exeption.h

#ifndef \_exception\_

#define \_exception\_

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

using namespace std;

class Err

{

protected:

string mes; // Что за ошибка

string mes2; // в каком методе произошла

public:

// Идентифицирующий конструктор

Err(string mes0, string mes20) : mes(mes0), mes2(mes20) {}

virtual void ErrOutput()

{

cout << "Ошибка: " << mes;

cout << " в функции/методе " << mes2 << endl;

\_getch();

}

};

class ErrFile : public Err

{

protected:

string mes1; // при работе с каким файлом

public:

// Идентифицирующий конструктор

ErrFile(string mes0, string mes10, string mes20)

: Err(mes0, mes20), mes1(mes10) {}

virtual void ErrOutput()

{

cout << "Ошибка: " << mes << endl;

cout << "При работе с файлом: " << mes1 << endl;

cout << "Произошла в методе: " << mes2 << endl;

\_getch();

}// Вывод сообщения об ошибке

};

class ErrMem : public ErrFile

{

protected:

public:

// Идентифицирующий конструктор

ErrMem(string mes0, string mes10, string mes20)

: ErrFile(mes0, mes10, mes20) {}

virtual void ErrOutput()

{

cout << "Ошибка: " << mes;

cout << " при выделении памяти для " << mes1;

cout << " в методе " << mes2 << endl;

\_getch();

}

string getMes() { return mes; }

string getMes1() { return mes1; }

};

#endif

print.h

#pragma once

#define NOMINMAX

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

#include "struct.h"

#include "exception.h"

#include "masa.h"

#include "masb.h"

#include "masc.h"

using namespace std;

class masC;

//Прототипы функций, использующих механизм виртуальных функций

void print(masA& z);

void printFile(masA\* p);

void sort3(masA\* p);

//Определение функций, использующих механизм виртуальных функций

void print(masA& z)

{

z.outputMas();

}

void printFile(masA\* p)

{

p->outputFile();

}

void sort3(masA\* z)

{

z->sort3();

}

void printA(masC\* p); // Вывод 1-й таблицы

void printB(masC\* p);

void printC(masC\* p);

void printA(masC\* p) // Вывод 1-й таблицы

{

p->masA::outputMas();

}

void printB(masC\* p) // Вывод 2-й таблицы

{

p->masB::outputMas();

}

void printC(masC\* p) // Вывод 3-й таблицы

{

p->masC::outputMas();

}

masA.cpp

#include "masc.h"

#define NOMINMAX

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

#include "struct.h"

#include "exception.h"

#include "masa.h"

#include "masb.h"

using namespace std;

masC::masC(masC& z)

try : masB(z)

{

int i;

l = z.l;

if (z.pz == NULL) pz = NULL;

else {

try {

pz = new abonent[l];

//throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

pz = NULL; l = 0;

throw ErrMem("Нет памяти", "исх. массива (pz)", "конструктор копирования masC");

}

for (i = 0; i < l; i++)

pz[i] = z.pz[i];

}

}

catch (ErrMem& ex) { throw ErrMem(ex.getMes(), ex.getMes1(), "masC(masC &z)"); }

masC& masC::operator=(masC& z) //перегруженная операция

{

int i;

if (this == &z) return \*this;

// Вызов перегруженной операции присваивания класса masB для

masB :: operator=(z); // присваивания родительской части объекта z

if (pz != NULL) delete[] pz;

l = z.l;

if (z.pz == NULL) pz = NULL;

else {

try {

pz = new abonent[l];

//throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

pz = NULL; l = 0;

throw ErrMem("Нет памяти", "исх. массива (pz)", " перегруженной операции присваивания класса masC");

}

for (i = 0; i < l; i++)

pz[i] = z.pz[i];

}

return \*this;

}

masC& masC :: operator= (masA& z)//перегруженная операция виртуальная

{

int i;

masC r;

if (this == &z) return \*this;

// Вызов перегруженной операции присваивания класса masС для

masB:: operator=(z); // присваивания родительской части объекта z

r = (masC&)z; // понижающее присваивание

if (pz != NULL) delete[] pz;

l = r.l;

if (r.pz == NULL) pz = NULL;

else {

try {

pz = new abonent[l];

//throw bad\_alloc();

}

catch (bad\_alloc)

{

throw ErrMem("Нет памяти", "студентов группы (pz)",

"masC::operator=( masC &z)");

}

for (i = 0; i < l; i++)

pz[i] = r.pz[i];

}

return \*this;

}

void masC::sort1()

{

int i, fl, nn;

abonent t;

nn = l;

do {

fl = 0; nn--;

for (i = 0; i < nn; i++)

if (pz[i].pers.name > pz[i + 1].pers.name)

{

fl = 1; t = pz[i];

pz[i] = pz[i + 1];

pz[i + 1] = t;

}

} while (fl == 1);

cout << "Массив структур упорядочен по ФИО в алфавитном порядке\n";

system("pause");

}

//Сортировка массива структур masC по дате заключения (сортировка по нескольким полям)

void masC::sort2() // сортировка по дате заключения

{

int fl, i, nn;

abonent t;

nn = l;

do {

fl = 0; nn--;

for (i = 0; i < nn; i++)

if (pz[i].datzakl > pz[i + 1].datzakl)

{

fl = 1; t = pz[i];

pz[i] = pz[i + 1];

pz[i + 1] = t;

}

} while (fl == 1);

cout << "Массив структур упорядочен по дате заключения договора\n";

system("pause");

}

//Вывод массива структур на экран

void masC::outputMas()

{

int i;

// Вывод заголовка таблицы (массива структур)

cout << "Результаты поиска: " << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << "| | | Дата заключения | | | | | |\n";

cout << "| № | Номер договора | День| Месяц| Год| Абонентская плата | Фамилия и инициалы | Район | Адрес | Телефон |\n";

cout << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | " << "\n";

cout << left;

// Вывод строк таблицы

for (i = 0; i < l; i++)

cout << "|" << setw(3) << i + 1 << "|" << setw(16) << pz[i].number << "|" << setw(5) << pz[i].datzakl.d << "|" << setw(6) << pz[i].datzakl.m << "|" << setw(4) << pz[i].datzakl.y << "|" << setw(19) << pz[i].plata << "|" << setw(20) << pz[i].pers.name << "|" << setw(10) << pz[i].pers.region << "|" << setw(29) << pz[i].pers.adress << "|" << setw(11) << pz[i].pers.phone << "| \n";

cout << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | \n";

system("pause");

}

//Вывод массива структур masC в текстовый файл

void masC::outputFile()

{

ofstream fout;

string file;

int i;

cout << "Имя выходного файла:"; cin >> file;

fout.open(file.c\_str());

// Защита от неправильного ввода имени файла

if (fout.fail()) throw ErrFile("не создается файл", file,

"masС::outputFile");

//Генерация исключения "Файл не создается"

fout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

fout << "| | | Дата заключения | | | | | |\n";

fout << "| № | Номер договора | День| Месяц| Год| Абонентская плата | Фамилия и инициалы | Район | Адрес | Телефон |\n";

fout << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | " << "\n";

for (i = 0; i < l; i++)

fout << "|" << setw(3) << i + 1 << "|" << setw(16) << pz[i].number << "|" << setw(5) << pz[i].datzakl.d << "|" << setw(6) << pz[i].datzakl.m << "|" << setw(4) << pz[i].datzakl.y << "|" << setw(19) << pz[i].plata << "|" << setw(20) << pz[i].pers.name << "|" << setw(10) << pz[i].pers.region << "|" << setw(29) << pz[i].pers.adress << "|" << setw(12) << pz[i].pers.phone << "| \n";

fout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

fout.close();

cout << "Массив структур сохранен в фаил\n";

system("pause");

}

//Определение метода findRegion(…) класса masС

void masC::findRegion()

{

int i, j;

string region0;

abonent\* y;

try {

y = new abonent[n];

//throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

throw ErrMem("нет памяти", "раб. массива (y)", "findRegion()");

}

cout << "Название региона : ";

cin >> region0;

if (cin.fail())

{

throw Err("при вводе", "findRegion()");

}

l = 0;

for (i = 0; i < n; i++)

if (px[i].pers.region == region0) { y[l] = px[i]; l++; }

if (pz != NULL) delete[] pz;

try {

pz = new abonent[l]; // throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

pz = NULL;

l = 0;

delete[]y;

throw ErrMem("нет памяти", "массива студ. (pz)", "findRegion()");

}

for (j = 0; j < l; j++)

pz[j] = y[j];

delete[] y;

cout << "Массив абонетов искомого района сформирован. \n";

system("pause");

}

ostream& operator<<(ostream& out, masC& z)

{

int i;

out << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

out << "| | | Дата заключения | | | | | |\n";

out << "| № | Номер договора | День| Месяц| Год| Абонентская плата | Фамилия и инициалы | Район | Адрес | Телефон |\n";

out << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | " << "\n";

out << left;

for (i = 0; i < z.l; i++)

cout << "|" << setw(3) << i + 1 << "|" << setw(16) << z.pz[i].number << "|" << setw(5) << z.pz[i].datzakl.d << "|" << setw(6) << z.pz[i].datzakl.m << "|" << setw(4) << z.pz[i].datzakl.y << "|" << setw(19) << z.pz[i].plata << "|" << setw(20) << z.pz[i].pers.name << "|" << setw(10) << z.pz[i].pers.region << "|" << setw(29) << z.pz[i].pers.adress << "|" << setw(11) << z.pz[i].pers.phone << "| \n";

out << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | \n";

system("pause"); return out;

}

masA.h

#pragma once

#ifndef \_masa\_

#define \_masa\_

#define NOMINMAX

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

#include "struct.h"

#include "exception.h"

using namespace std;

int operator>(dat v, dat w) //Определение перегруженной операции сравнения для структур типа data

{

if (v.y > w.y) return 1;

if ((v.y == w.y) && (v.m > w.m)) return 1;

if ((v.y == w.y) && (v.m == w.m) && (v.d > w.d)) return 1;

return 0;

}

int operator>(abonent v, abonent w) //Определение перегруженной операции сравнения для структур

{

if (v.plata > w.plata) return 1;

if ((v.plata > w.plata) && (v.pers.name > w.pers.name)) return 1;

return 0;

}

ostream& operator<<(ostream& out, abonent& px) //Определение перегруженной операции вывода на экран структуры

{

out << "|" << setw(16) << px.number << "|" << setw(5) << px.datzakl.d << "|" << setw(6) << px.datzakl.m << "|" << setw(4) << px.datzakl.y << "|" << setw(19) << px.plata << "|" << setw(20) << px.pers.name << "|" << setw(10) << px.pers.region << "|" << setw(16) << px.pers.adress << "|" << setw(11) << px.pers.phone << "|\n";

return out;

}

class masA

{

protected:

abonent\* px;

int n;

public:

masA() :px(NULL), n(0) {};

masA(masA& z);

virtual ~masA() { if (px != NULL) delete[]px; cout << "~masA()\n"; }

virtual masA& operator=(masA& z);

//Методы только в masA

void inputFile();

void addAbonent();

void deleteAbonent();

//Переопределяемые методы

void sort1();

void sort2();

//Виртуальные методы

virtual void outputMas();

virtual void outputFile();

virtual void sort3();

//Определение перегруженной операции вывода на экран объекта класса masA

friend ostream& operator<<(ostream& out, masA& z);

};//Определение методов класса masA

#include "masa.cpp"

#endif

masB.cpp

#include "masb.h"

#define NOMINMAX

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

#include "struct.h"

#include "exception.h"

#include "masa.h"

using namespace std;

masB::masB(masB& z)

try : masA(z) //конструктор копирования класса masB

{

int i;

k = z.k;

if (z.mas == NULL) mas = NULL;

else {

try {

mas = new PLATAA[k];

throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

mas = NULL; n = 0;

throw ErrMem("Нет памяти", "исх. массива (mas)", "конструктор копирования masB");

}

if (mas == NULL) {

cout << "Нет памяти.\n";

cout << "Конструктор копирования.\n";

system("pause"); exit(1);

}

for (i = 0; i < k; i++)

mas[i] = z.mas[i];

}

}

catch (ErrMem& ex) { throw ErrMem(ex.getMes(), ex.getMes1(), "masB(masB &z)"); }

masB& masB :: operator= (masA& z) //перегруженная операция виртуальная

{

int i;

masB r;

if (this == &z) return \*this;

// Вызов перегруженной операции присваивания класса masА для

masA :: operator=(z); // присваивания родительской части объекта z

r = (masB&)z; // понижающее присваивание

if (mas != NULL) delete[] mas;

k = r.k;

if (r.mas == NULL) mas = NULL;

else {

try {

mas = new PLATAA[k];

//throw bad\_alloc();

}

catch (bad\_alloc)

{

throw ErrMem("Нет памяти", "перечня (py)",

"masB::operator=( masB &z))");

}

for (i = 0; i < k; i++)

mas[i] = r.mas[i];

}

return \*this;

}

masB& masB :: operator=(masB& z) //перегруженная операция присваивания класса masB

{

int i;

if (this == &z) return \*this;

// Вызов перегруженной операции присваивания класса masА для

masA :: operator= (z); // присваивания родительской части объекта z

if (mas != NULL) delete[] mas;

k = z.k;

if (z.mas == NULL) mas = NULL;

else {

try {

mas = new PLATAA[k];

//throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

mas = NULL; n = 0;

throw ErrMem("Нет памяти", "исх. массива (mas)", "перегруженная операция присваивания masB");

}

if (mas == NULL) {

cout << "Нет памяти при присваивании. \n";

system("pause"); return z;

}

for (i = 0; i < k; i++)

mas[i] = z.mas[i];

}

return \*this;

}

//Сортировка перечня по абонентской плате

void masB::sort1()

{

int fl, i, kk;

PLATAA t;

kk = k;

do {

fl = 0; kk--;

for (i = 0; i < kk; i++)

if (mas[i].plata > mas[i + 1].plata)

{

fl = 1; t = mas[i];

mas[i] = mas[i + 1];

mas[i + 1] = t;

}

} while (fl == 1);

cout << "Перечень упорядочен по абонентской плате.\n";

system("pause");

}

// Сортировка перечня по численности

void masB::sort3()

{

int i, fl, kk;

PLATAA t;

kk = k;

do {

fl = 0; kk--;

for (i = 0; i < kk; i++)

if (mas[i].count > mas[i + 1].count)

{

fl = 1; t = mas[i];

mas[i] = mas[i + 1];

mas[i + 1] = t;

}

} while (fl == 1);

cout << "Перечень упорядочен по количеству абонентов по каждому виду платежа. \n";

system("pause");

}

//Вывод перечня в текстовый файл

void masB::outputFile()

{

ofstream fout;

string file;

int i;

cout << "Имя выходного файла\n"; cin >> file;

fout.open(file.c\_str());

// Защита от неправильного ввода имени файла

if (fout.fail()) throw ErrFile("не создается файл", file,

"masВ::outputFile");

//Генерация исключения "Файл не создается"

fout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

fout << "| Плата | Количество | \n";

fout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

for (i = 0; i < k; i++)

fout << setw(6) << mas[i].plata << " | " << setw(11) << mas[i].count << " | \n";

fout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

fout.close();

cout << "Перечень сохранен в файле.\n";

system("pause");

}

//Вывод перечня на экран

void masB::outputMas()

{

int i;

// Вывод заголовка таблицы (массива структур)

cout << "перечень: " << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << "| Плата | Количество | \n";

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << left;

// Вывод строк таблицы

for (i = 0; i < k; i++)

cout << setw(6) << "|" << mas[i].plata << "|" << setw(11) << mas[i].count << " | \n";

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

system("pause");

}

//Определение метода makePerech (…) класса masB

void masB::makePerech()

{

int i, j, fl;

PLATAA\* y;

try {

y = new PLATAA[n];

//throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

throw ErrMem("нет памяти", "рабочего массива (перечня)", "makePerech()");

}

if (y == NULL) {

cout << "Нет памяти.\n";

system("pause");

return;

}

k = 0;

if (mas != NULL) delete[] mas;

for (i = 0; i < n; i++)

{

fl = 0;

for (j = 0; j < k; j++)

if (px[i].plata == y[j].plata)

{

fl = 1; y[j].count++;

}

if (fl == 0) {

y[k].plata = px[i].plata;

y[k].count = 1;

k++;

}

}

try {

mas = new PLATAA[k];

//throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

mas == NULL;

k = 0;

delete[] mas;

throw ErrMem("нет памяти", " перечня(py)", "makePerech()");

}

for (j = 0; j < k; j++)

mas[j] = y[j];

delete[] y;

cout << "Перечень сформирован.\n";

system("pause");

}

ostream& operator<<(ostream& out, masB& z)

{

int i;

out << "перечень: " << endl;

out << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

out << "| Плата | Количество | \n";

out << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

for (i = 0; i < z.k; i++)

out << z.mas[i];

out << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

system("pause");

return out;

}

masB.h

#pragma once

#ifndef \_masb\_

#define \_masb\_

#define NOMINMAX

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

#include "struct.h"

#include "exception.h"

#include "masa.h"

using namespace std;

ostream& operator<<(ostream& out, PLATAA& z) //Определение перегруженной операции вывода на экран структуры типа PLATA

{

out << "|" << setw(8) << std::left << z.plata << "|" << setw(12) << std::left << z.count << "|" << endl;

return out;

}

class masB : public masA

{

protected:

PLATAA\* mas;

int k;

public:

masB() :mas(NULL), k(0) {};

masB(masB& z);

virtual ~masB() { if (mas != NULL) delete[]mas; cout << "~masB()\n"; }

masB& operator=(masB& z);

virtual masB& operator= (masA& z);

//Переопределяемые методы класса

void sort3();

void sort1();

//Методы только в masB

void makePerech();

//Виртуальные методы

virtual void outputMas();

virtual void outputFile();

friend ostream& operator<<(ostream& out, masB& z);

};

#include "masb.cpp"

#endif

masC.cpp

#include "masc.h"

#define NOMINMAX

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

#include "struct.h"

#include "exception.h"

#include "masa.h"

#include "masb.h"

using namespace std;

masC::masC(masC& z)

try : masB(z)

{

int i;

l = z.l;

if (z.pz == NULL) pz = NULL;

else {

try {

pz = new abonent[l];

//throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

pz = NULL; l = 0;

throw ErrMem("Нет памяти", "исх. массива (pz)", "конструктор копирования masC");

}

for (i = 0; i < l; i++)

pz[i] = z.pz[i];

}

}

catch (ErrMem& ex) { throw ErrMem(ex.getMes(), ex.getMes1(), "masC(masC &z)"); }

masC& masC::operator=(masC& z) //перегруженная операция

{

int i;

if (this == &z) return \*this;

// Вызов перегруженной операции присваивания класса masB для

masB :: operator=(z); // присваивания родительской части объекта z

if (pz != NULL) delete[] pz;

l = z.l;

if (z.pz == NULL) pz = NULL;

else {

try {

pz = new abonent[l];

//throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

pz = NULL; l = 0;

throw ErrMem("Нет памяти", "исх. массива (pz)", " перегруженной операции присваивания класса masC");

}

for (i = 0; i < l; i++)

pz[i] = z.pz[i];

}

return \*this;

}

masC& masC :: operator= (masA& z)//перегруженная операция виртуальная

{

int i;

masC r;

if (this == &z) return \*this;

// Вызов перегруженной операции присваивания класса masС для

masB:: operator=(z); // присваивания родительской части объекта z

r = (masC&)z; // понижающее присваивание

if (pz != NULL) delete[] pz;

l = r.l;

if (r.pz == NULL) pz = NULL;

else {

try {

pz = new abonent[l];

//throw bad\_alloc();

}

catch (bad\_alloc)

{

throw ErrMem("Нет памяти", "студентов группы (pz)",

"masC::operator=( masC &z)");

}

for (i = 0; i < l; i++)

pz[i] = r.pz[i];

}

return \*this;

}

void masC::sort1()

{

int i, fl, nn;

abonent t;

nn = l;

do {

fl = 0; nn--;

for (i = 0; i < nn; i++)

if (pz[i].pers.name > pz[i + 1].pers.name)

{

fl = 1; t = pz[i];

pz[i] = pz[i + 1];

pz[i + 1] = t;

}

} while (fl == 1);

cout << "Массив структур упорядочен по ФИО в алфавитном порядке\n";

system("pause");

}

//Сортировка массива структур masC по дате заключения (сортировка по нескольким полям)

void masC::sort2() // сортировка по дате заключения

{

int fl, i, nn;

abonent t;

nn = l;

do {

fl = 0; nn--;

for (i = 0; i < nn; i++)

if (pz[i].datzakl > pz[i + 1].datzakl)

{

fl = 1; t = pz[i];

pz[i] = pz[i + 1];

pz[i + 1] = t;

}

} while (fl == 1);

cout << "Массив структур упорядочен по дате заключения договора\n";

system("pause");

}

//Вывод массива структур на экран

void masC::outputMas()

{

int i;

// Вывод заголовка таблицы (массива структур)

cout << "Результаты поиска: " << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout << "| | | Дата заключения | | | | | |\n";

cout << "| № | Номер договора | День| Месяц| Год| Абонентская плата | Фамилия и инициалы | Район | Адрес | Телефон |\n";

cout << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | " << "\n";

cout << left;

// Вывод строк таблицы

for (i = 0; i < l; i++)

cout << "|" << setw(3) << i + 1 << "|" << setw(16) << pz[i].number << "|" << setw(5) << pz[i].datzakl.d << "|" << setw(6) << pz[i].datzakl.m << "|" << setw(4) << pz[i].datzakl.y << "|" << setw(19) << pz[i].plata << "|" << setw(20) << pz[i].pers.name << "|" << setw(10) << pz[i].pers.region << "|" << setw(29) << pz[i].pers.adress << "|" << setw(11) << pz[i].pers.phone << "| \n";

cout << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | \n";

system("pause");

}

//Вывод массива структур masC в текстовый файл

void masC::outputFile()

{

ofstream fout;

string file;

int i;

cout << "Имя выходного файла:"; cin >> file;

fout.open(file.c\_str());

// Защита от неправильного ввода имени файла

if (fout.fail()) throw ErrFile("не создается файл", file,

"masС::outputFile");

//Генерация исключения "Файл не создается"

fout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

fout << "| | | Дата заключения | | | | | |\n";

fout << "| № | Номер договора | День| Месяц| Год| Абонентская плата | Фамилия и инициалы | Район | Адрес | Телефон |\n";

fout << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | " << "\n";

for (i = 0; i < l; i++)

fout << "|" << setw(3) << i + 1 << "|" << setw(16) << pz[i].number << "|" << setw(5) << pz[i].datzakl.d << "|" << setw(6) << pz[i].datzakl.m << "|" << setw(4) << pz[i].datzakl.y << "|" << setw(19) << pz[i].plata << "|" << setw(20) << pz[i].pers.name << "|" << setw(10) << pz[i].pers.region << "|" << setw(29) << pz[i].pers.adress << "|" << setw(12) << pz[i].pers.phone << "| \n";

fout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

fout.close();

cout << "Массив структур сохранен в фаил\n";

system("pause");

}

//Определение метода findRegion(…) класса masС

void masC::findRegion()

{

int i, j;

string region0;

abonent\* y;

try {

y = new abonent[n];

//throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

throw ErrMem("нет памяти", "раб. массива (y)", "findRegion()");

}

cout << "Название региона : ";

cin >> region0;

if (cin.fail())

{

throw Err("при вводе", "findRegion()");

}

l = 0;

for (i = 0; i < n; i++)

if (px[i].pers.region == region0) { y[l] = px[i]; l++; }

if (pz != NULL) delete[] pz;

try {

pz = new abonent[l]; // throw bad\_alloc();

}

catch (bad\_alloc) // Обработчик без передачи данных

{

pz = NULL;

l = 0;

delete[]y;

throw ErrMem("нет памяти", "массива студ. (pz)", "findRegion()");

}

for (j = 0; j < l; j++)

pz[j] = y[j];

delete[] y;

cout << "Массив абонетов искомого района сформирован. \n";

system("pause");

}

ostream& operator<<(ostream& out, masC& z)

{

int i;

out << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

out << "| | | Дата заключения | | | | | |\n";

out << "| № | Номер договора | День| Месяц| Год| Абонентская плата | Фамилия и инициалы | Район | Адрес | Телефон |\n";

out << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | " << "\n";

out << left;

for (i = 0; i < z.l; i++)

cout << "|" << setw(3) << i + 1 << "|" << setw(16) << z.pz[i].number << "|" << setw(5) << z.pz[i].datzakl.d << "|" << setw(6) << z.pz[i].datzakl.m << "|" << setw(4) << z.pz[i].datzakl.y << "|" << setw(19) << z.pz[i].plata << "|" << setw(20) << z.pz[i].pers.name << "|" << setw(10) << z.pz[i].pers.region << "|" << setw(29) << z.pz[i].pers.adress << "|" << setw(11) << z.pz[i].pers.phone << "| \n";

out << "|" << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << " | \n";

system("pause"); return out;

}

masC.h

#pragma once

#ifndef \_masc\_

#define \_masc\_

#define NOMINMAX

#include <iostream>

#include <limits>

#include <fstream>

#include <cstdlib>

#include <string>

#include <iomanip>

#include <conio.h>

#include <Windows.h>

#include <sstream>

#include "masb.h"

#include "struct.h"

#include "exception.h"

using namespace std;

class masC : public masB

{

protected:

abonent\* pz;

int l;

public:

masC() :pz(NULL), l(0) {};

masC(masC& z);

virtual ~masC() { if (pz != NULL) delete[]pz; cout << "~masC() \n"; }

masC& operator=(masC& z);

virtual masC& operator=(masA& z);

//Методы только masC

void findRegion();

//Переопределяемые методы

void sort2();

void sort1();

//Виртуальные методы

virtual void outputMas();

virtual void outputFile();

friend ostream& operator<<(ostream& out, masC& z);

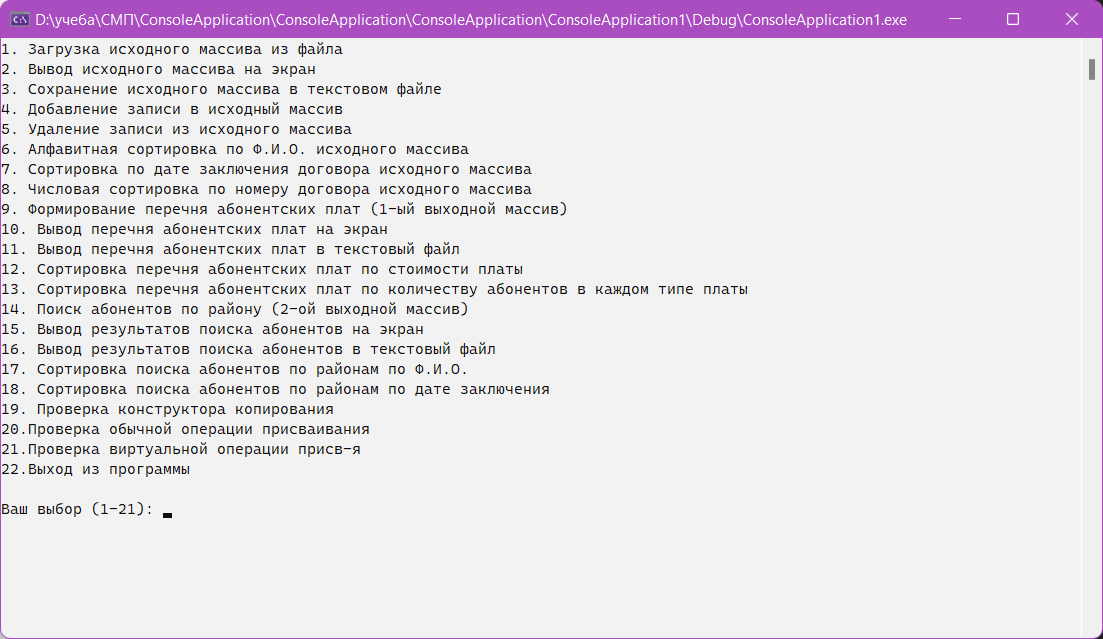
};

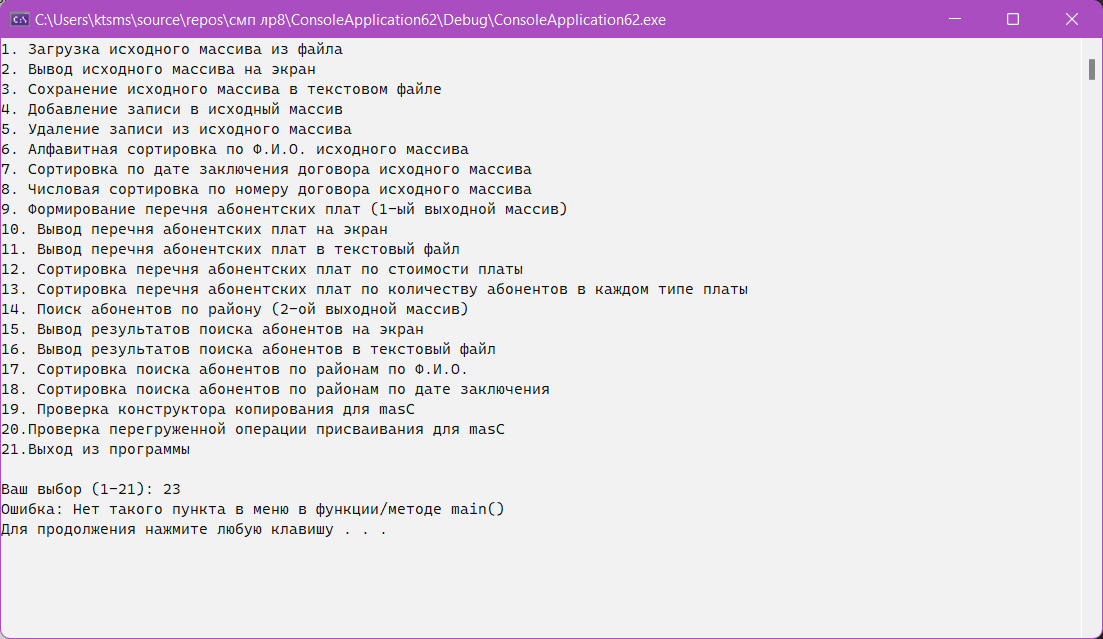
#include "masc.cpp"

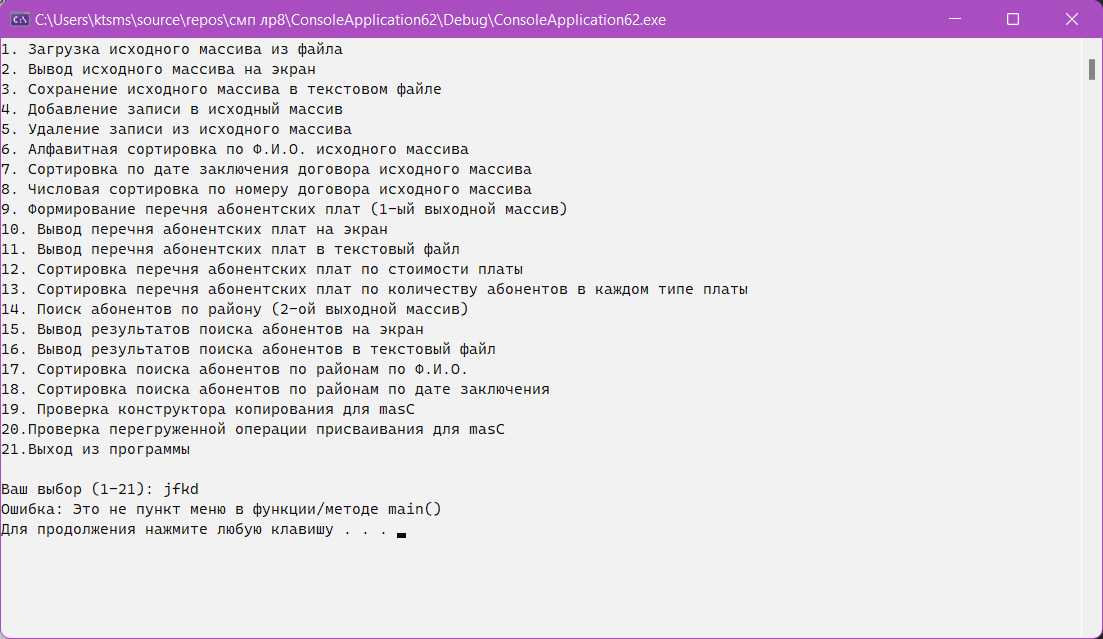
#endif

Скриншоты работы программы

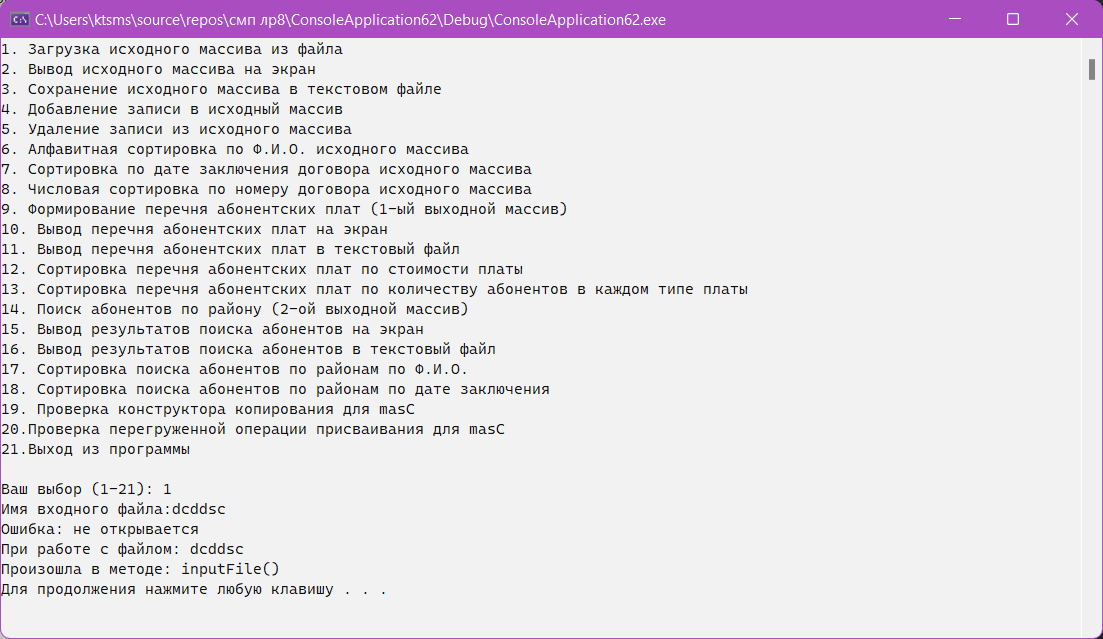
1. Меню

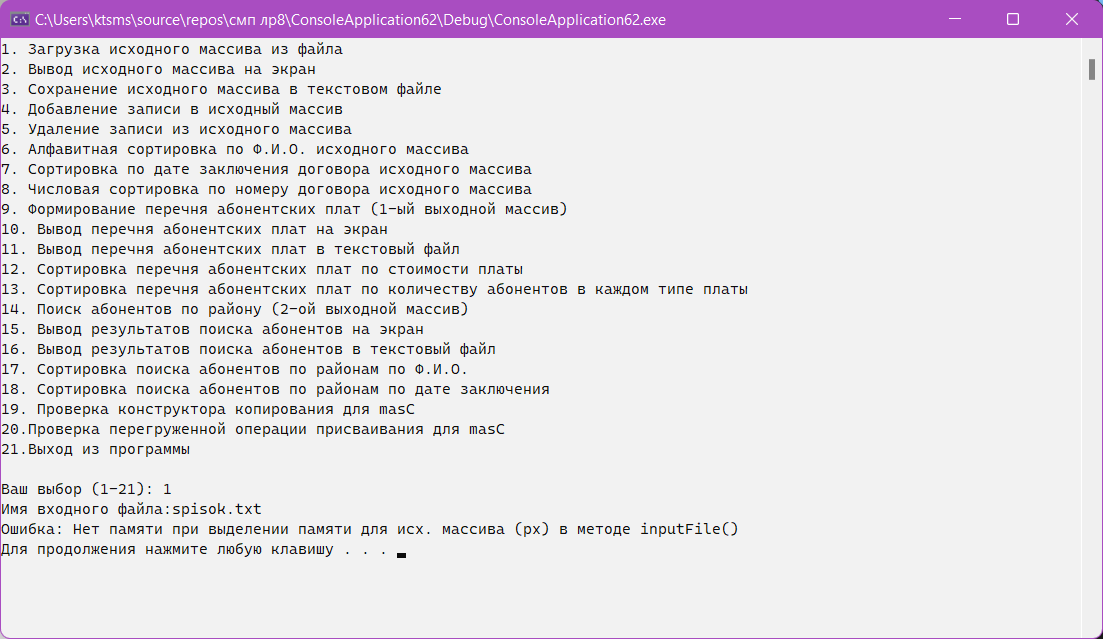


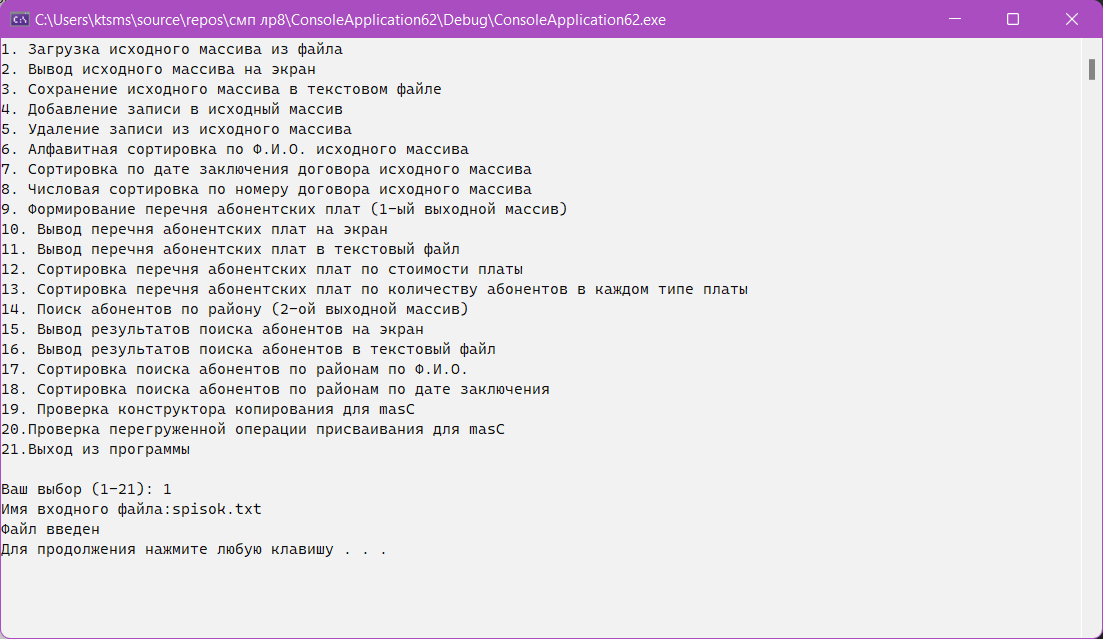




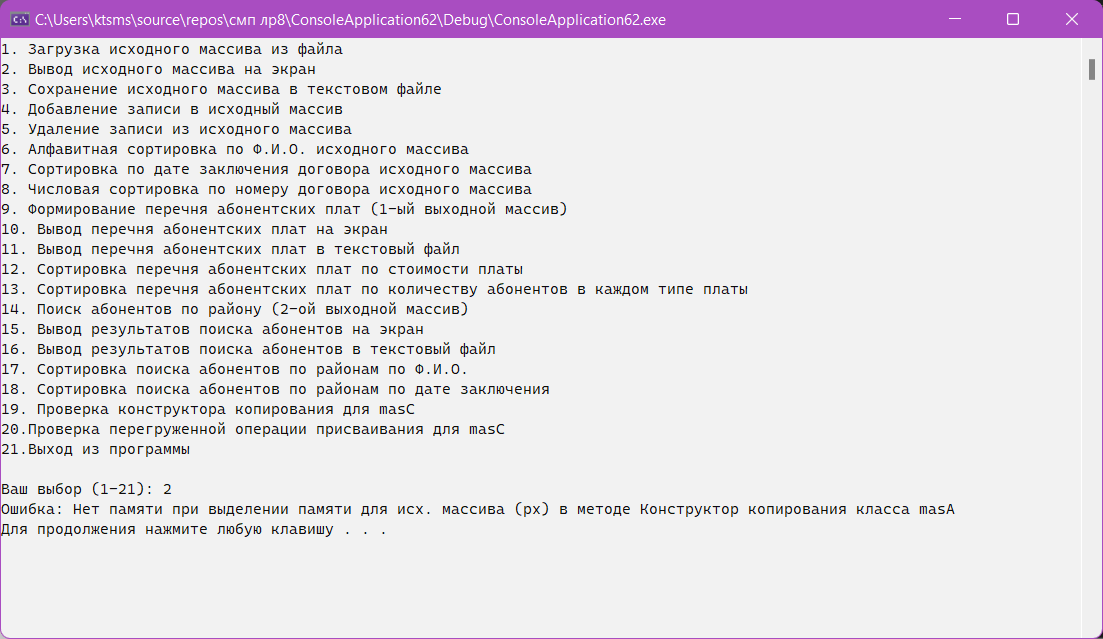
1. Ввод из файла

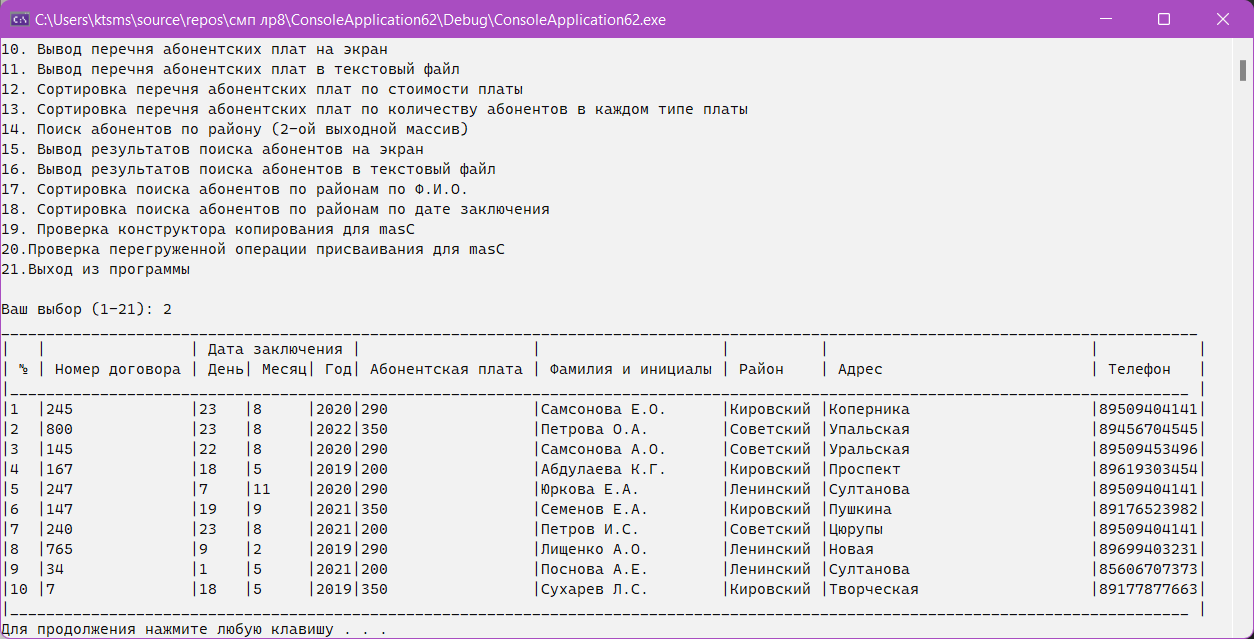




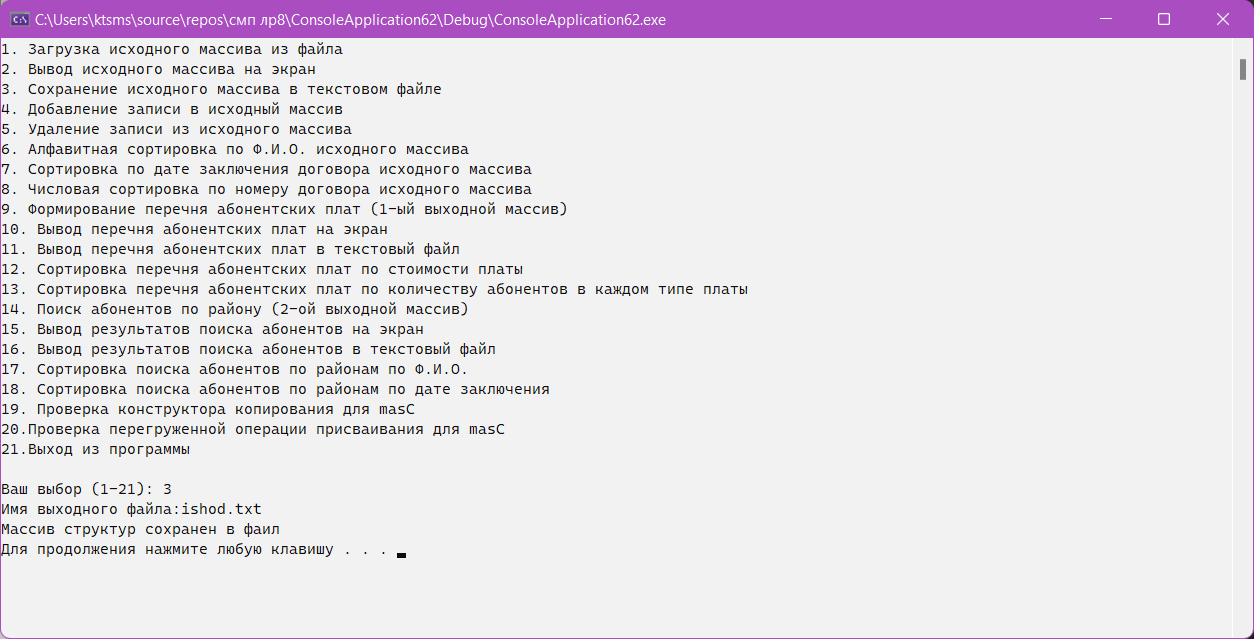


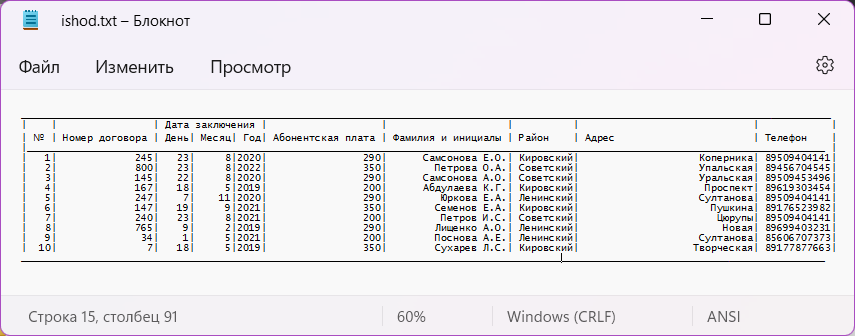
1. Вывод на экран



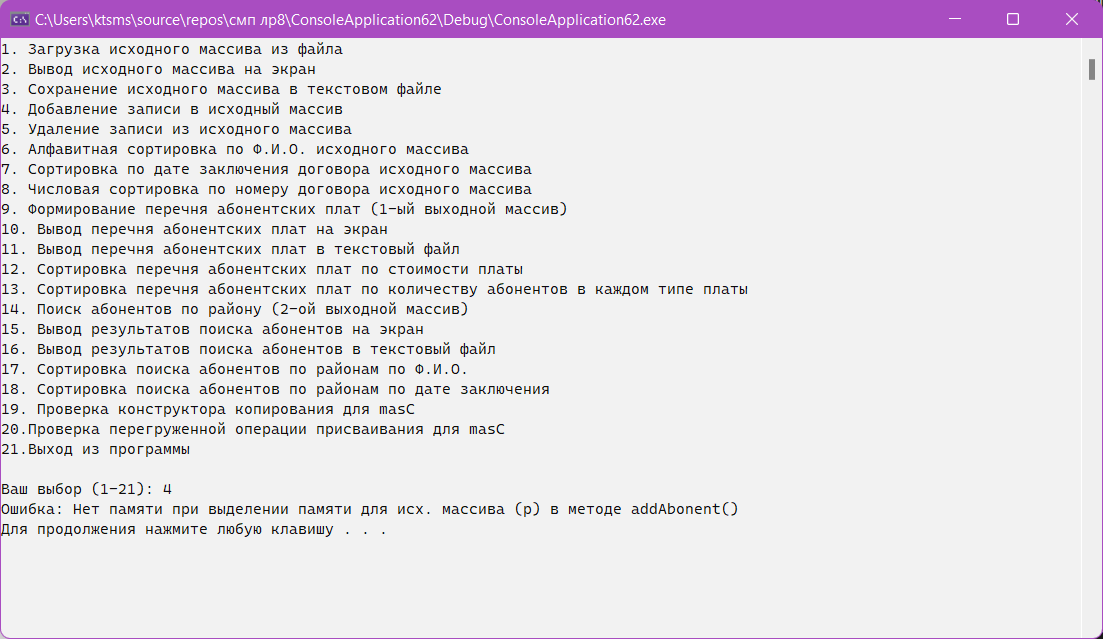


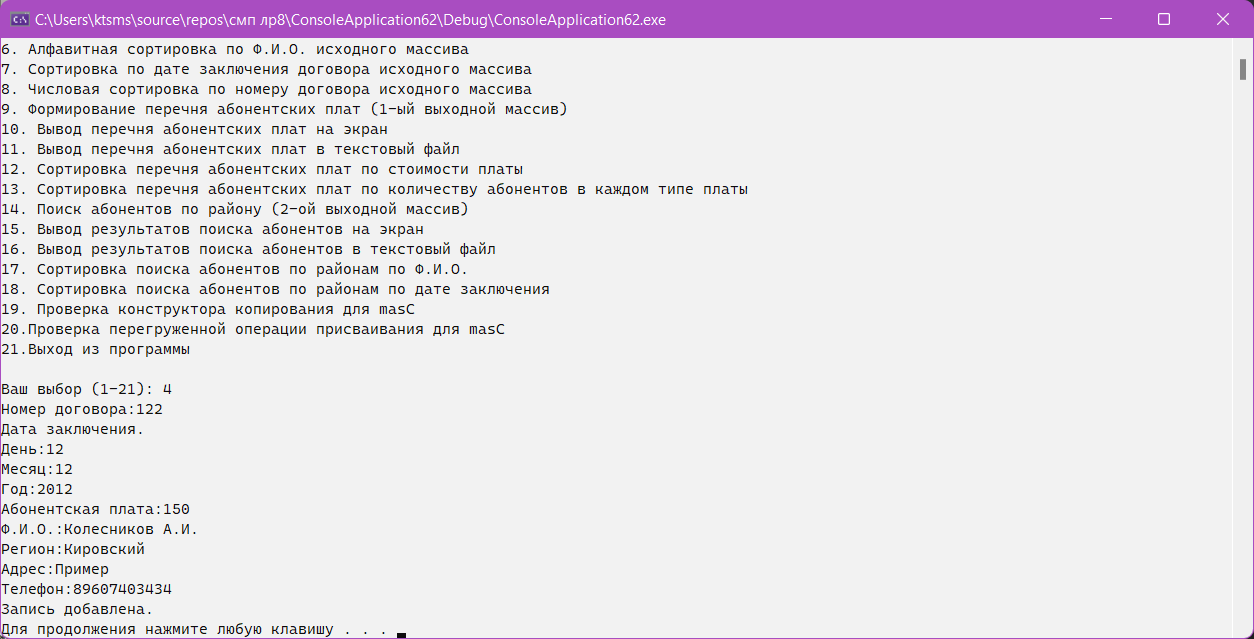
1. Сохранение в файл

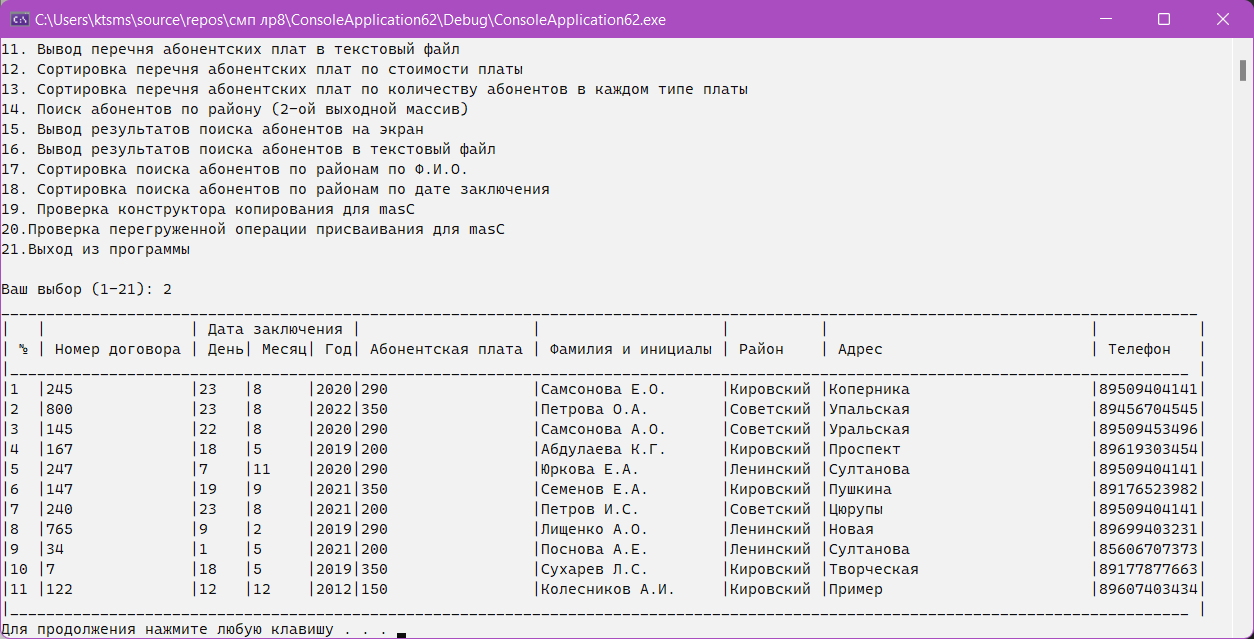




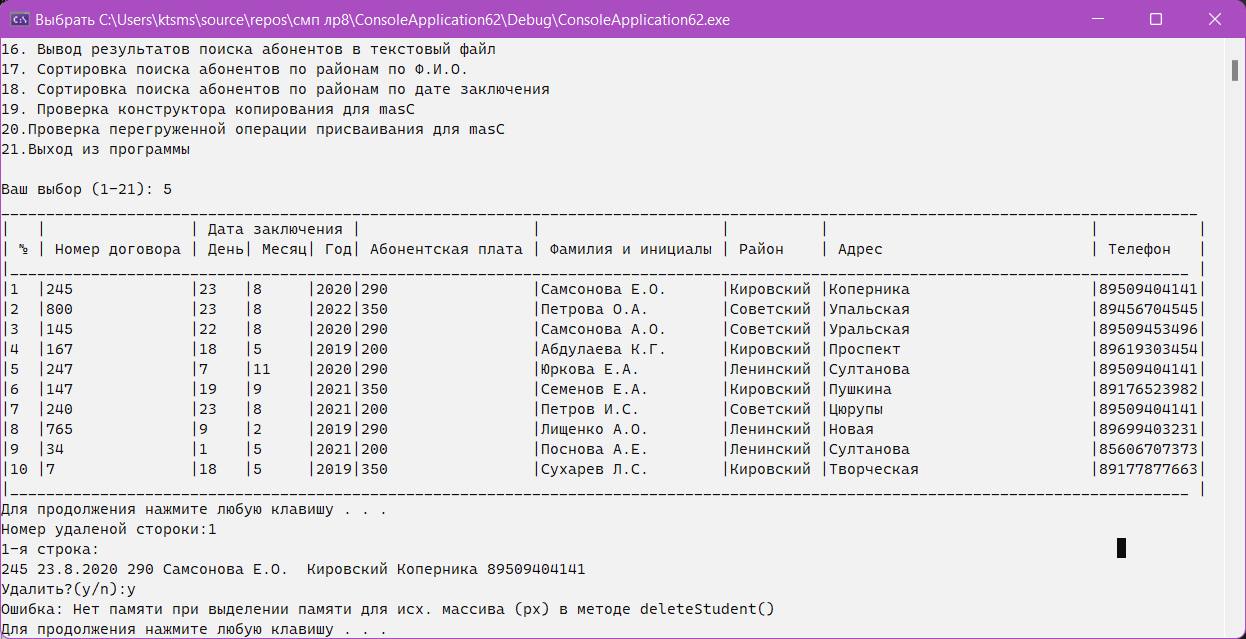
1. Добавление записи

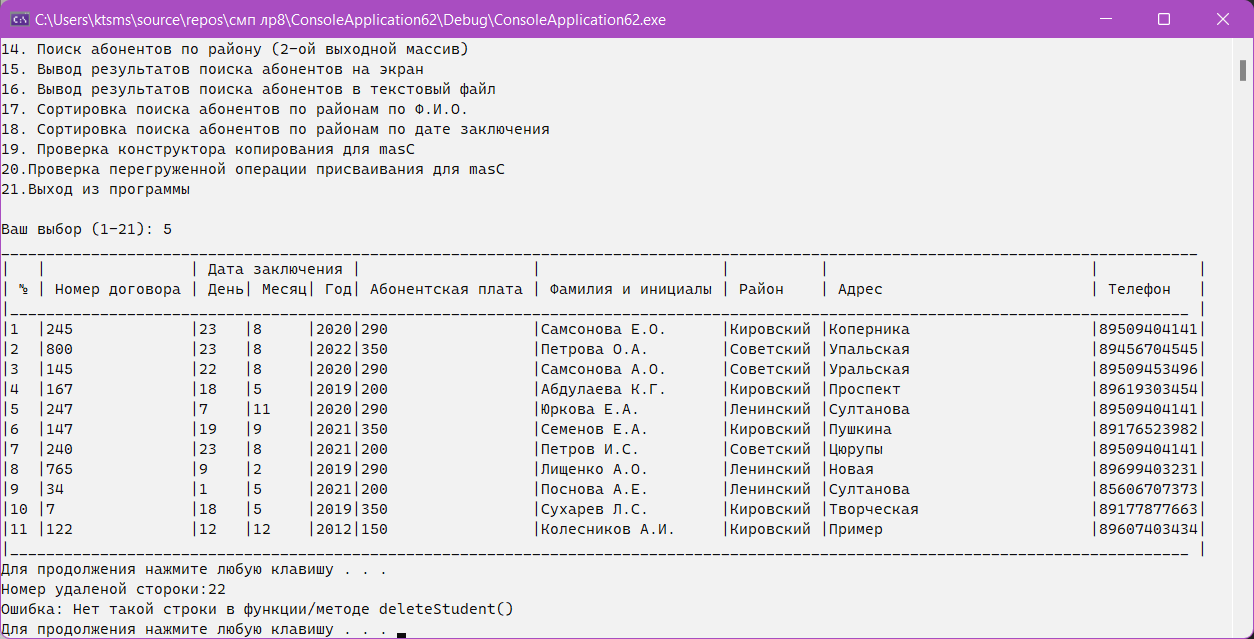


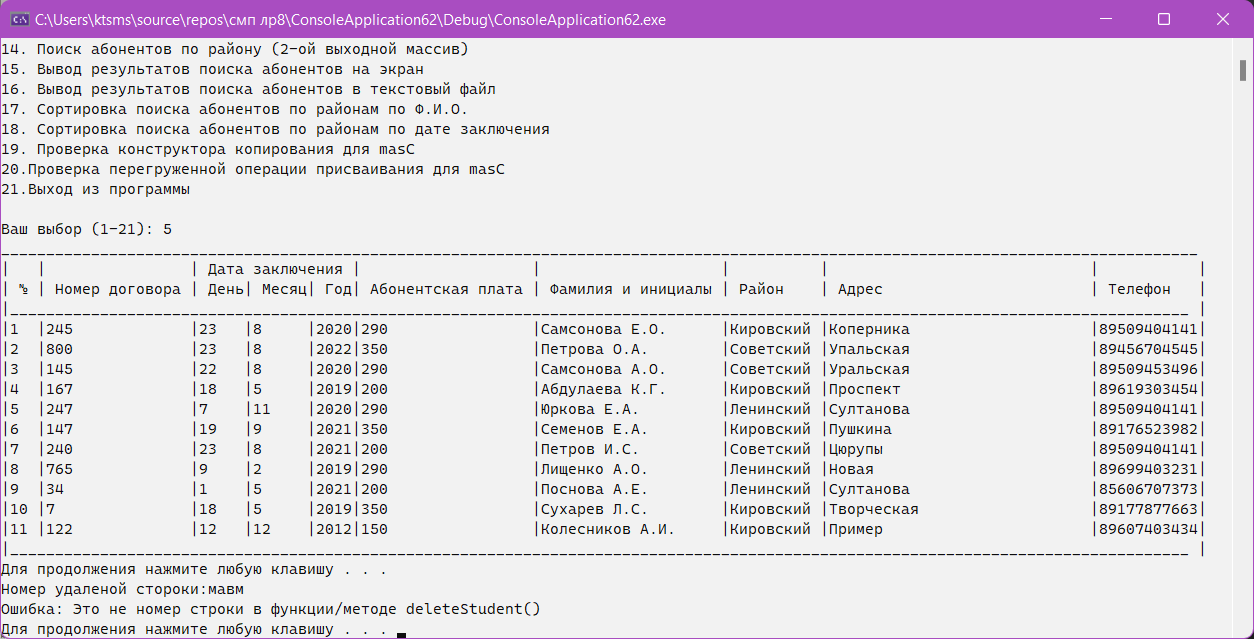


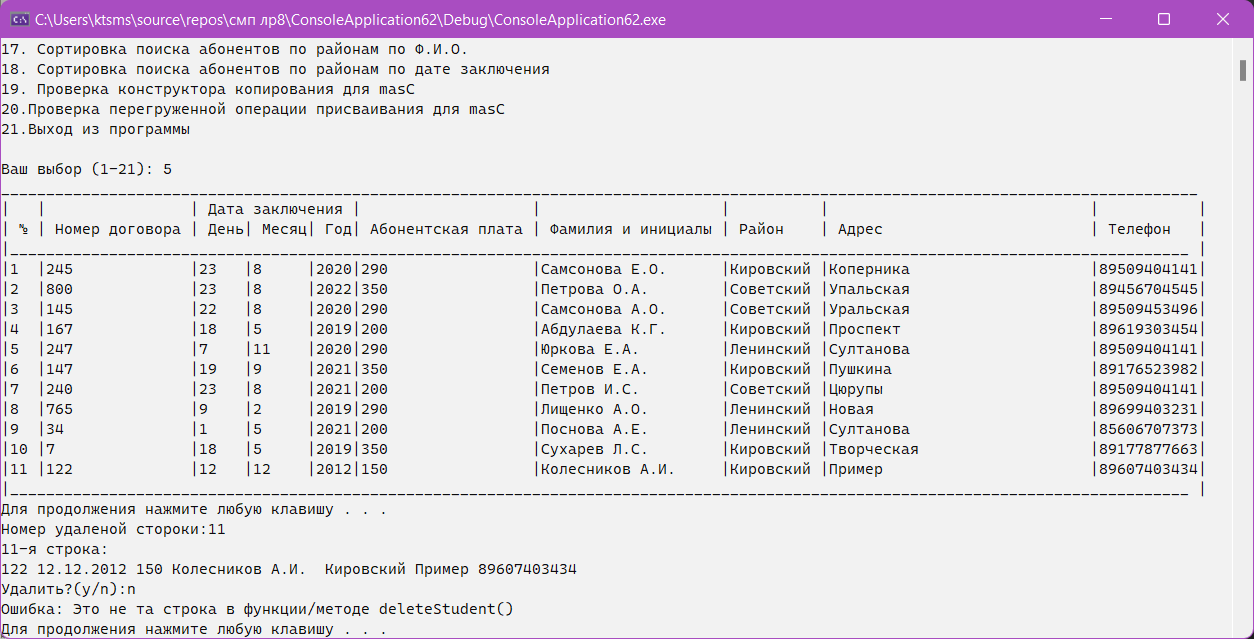


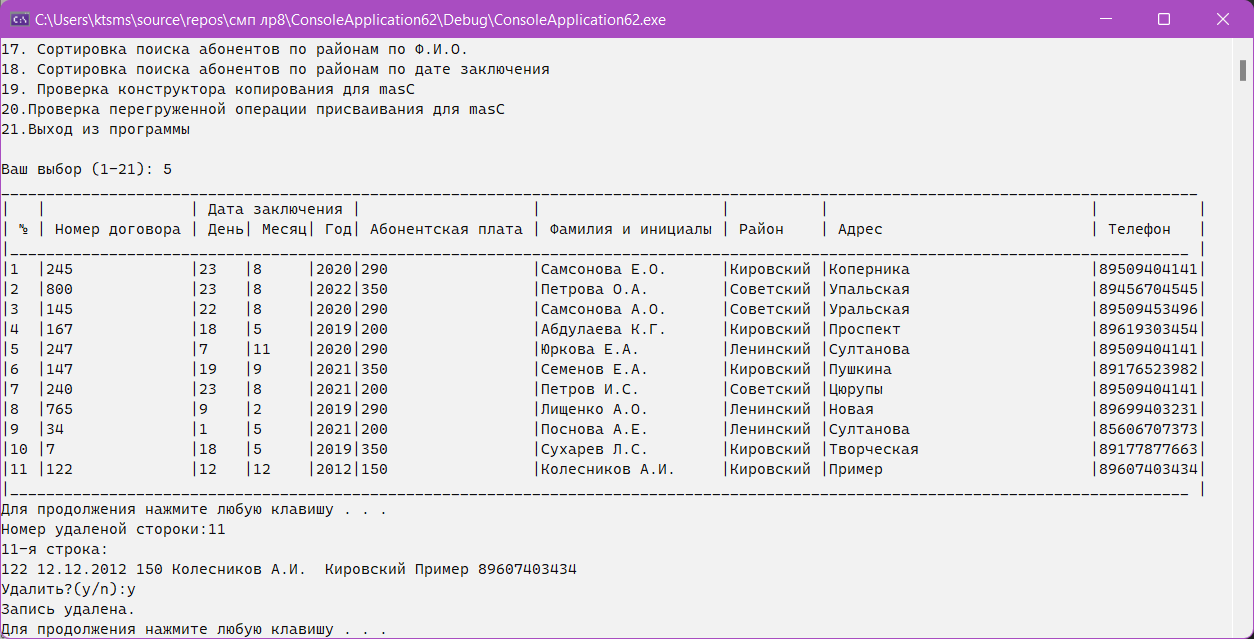
1. Удаление записи

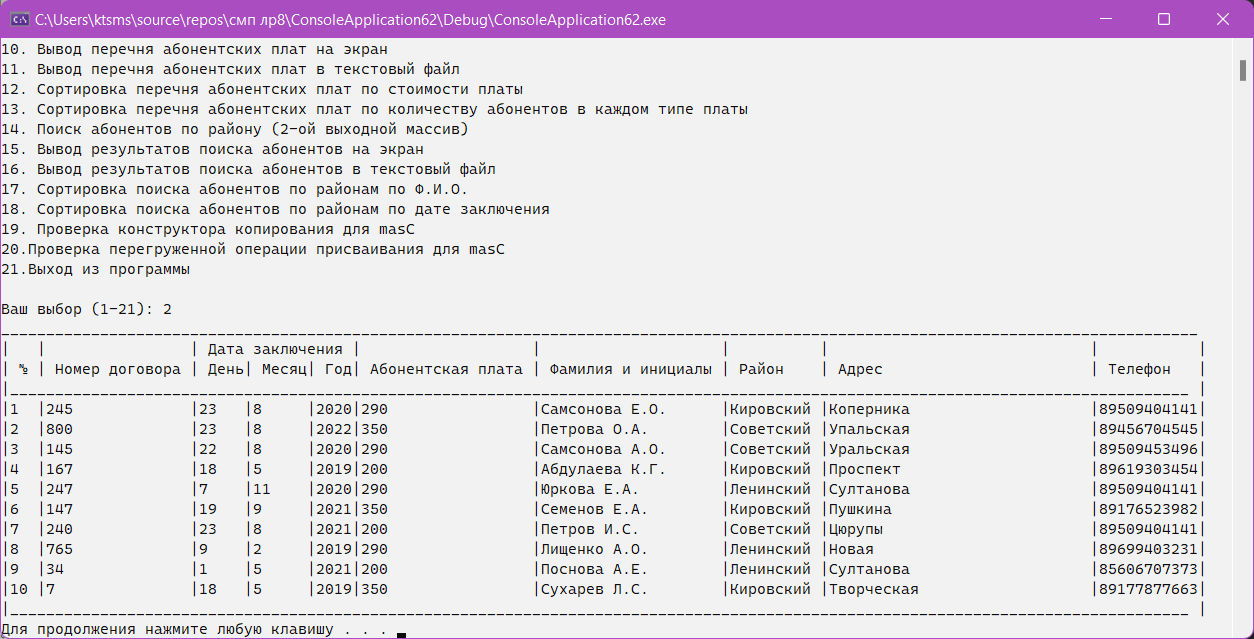




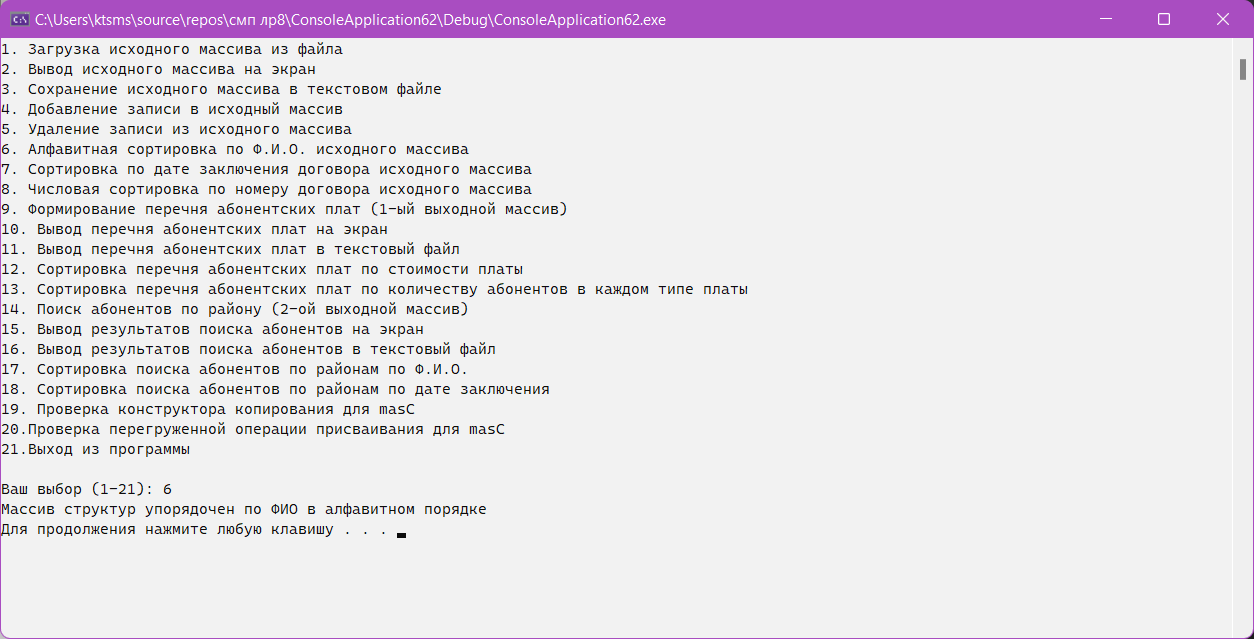


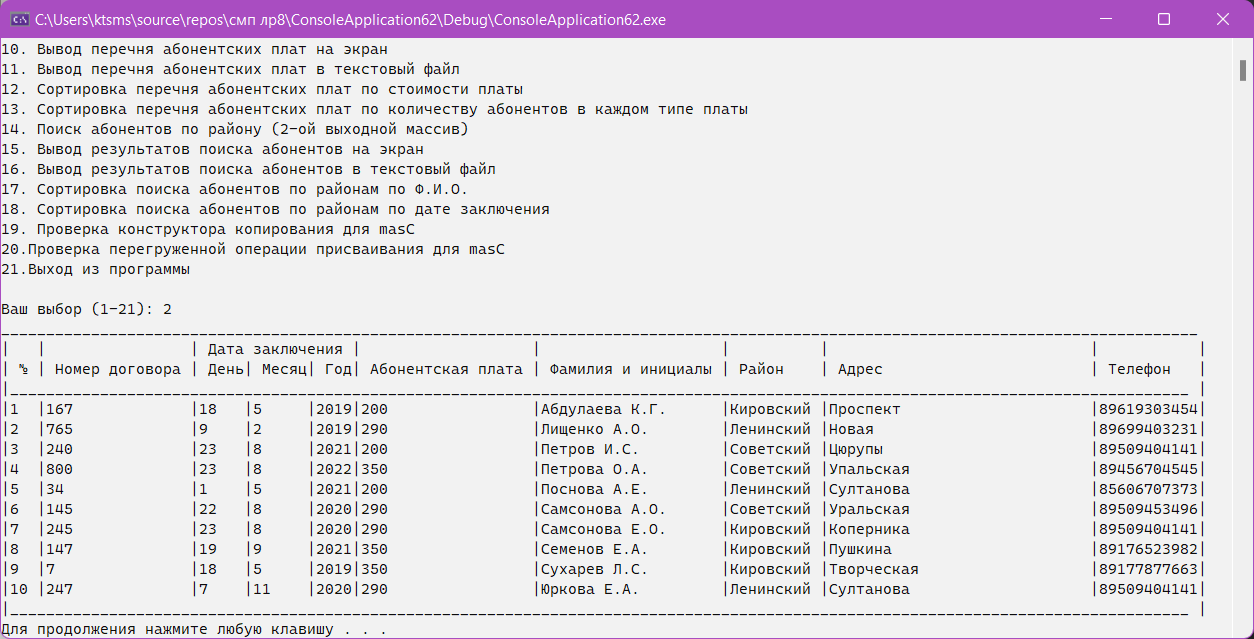




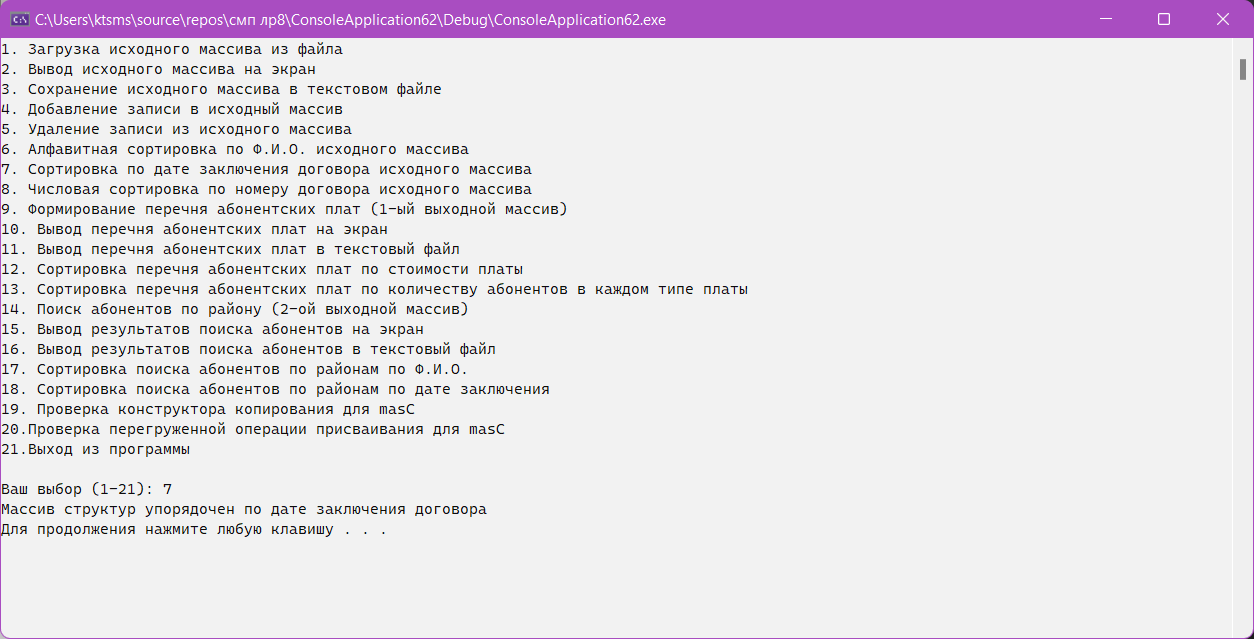


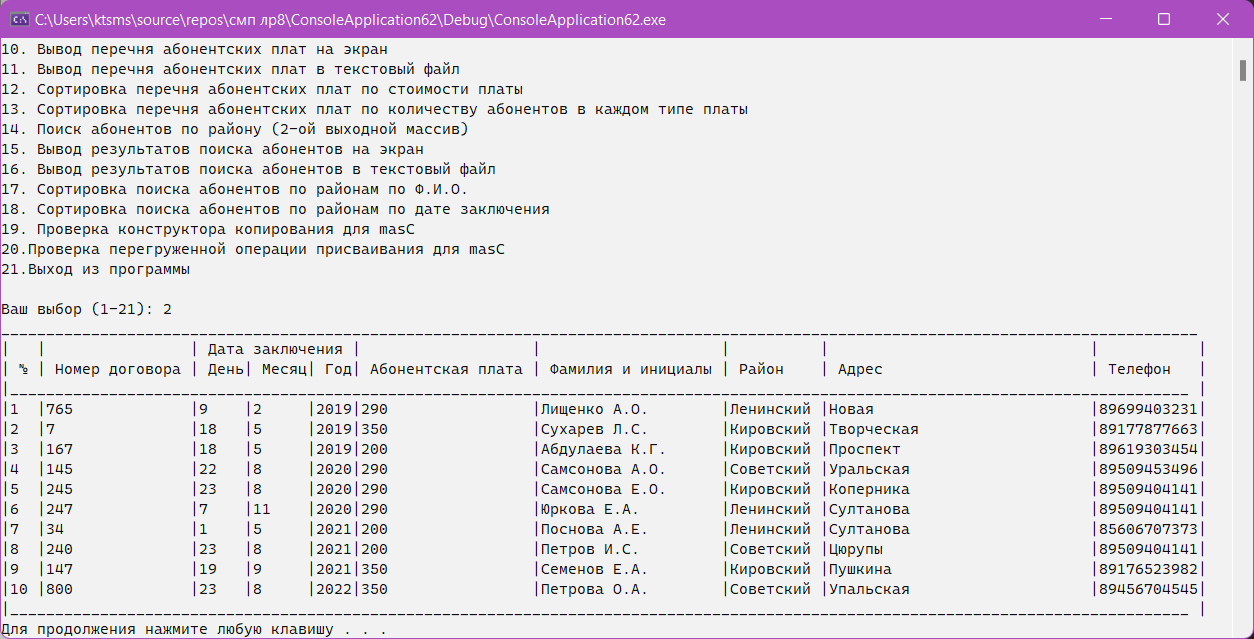
1. Сортировка по ФИО



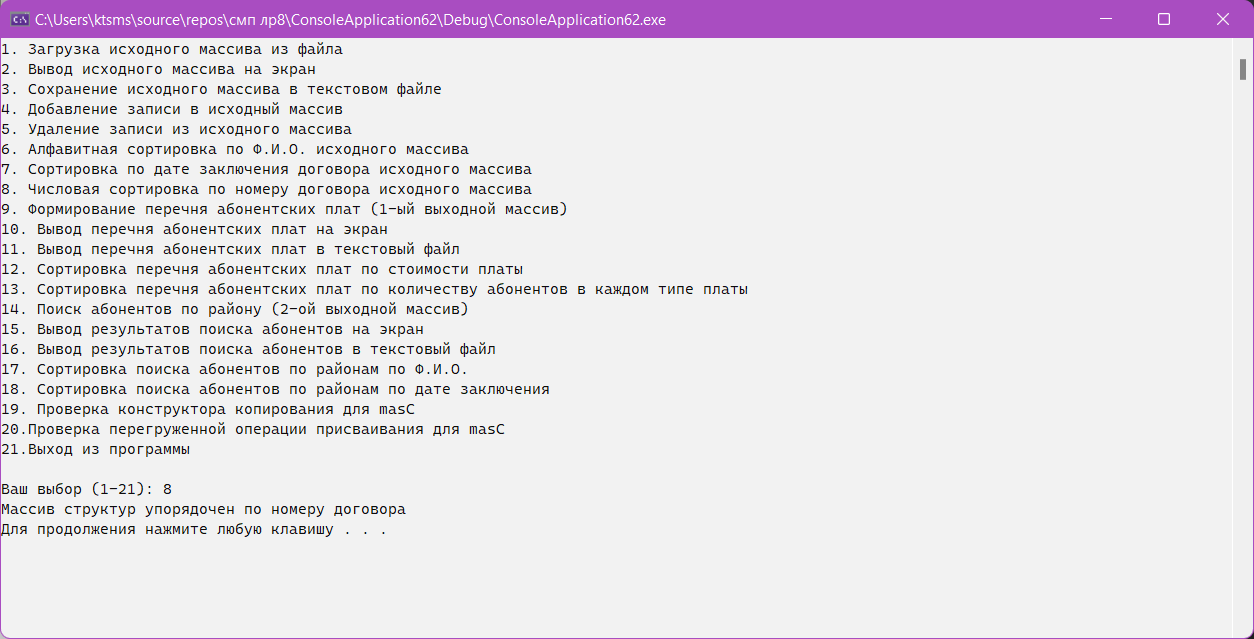


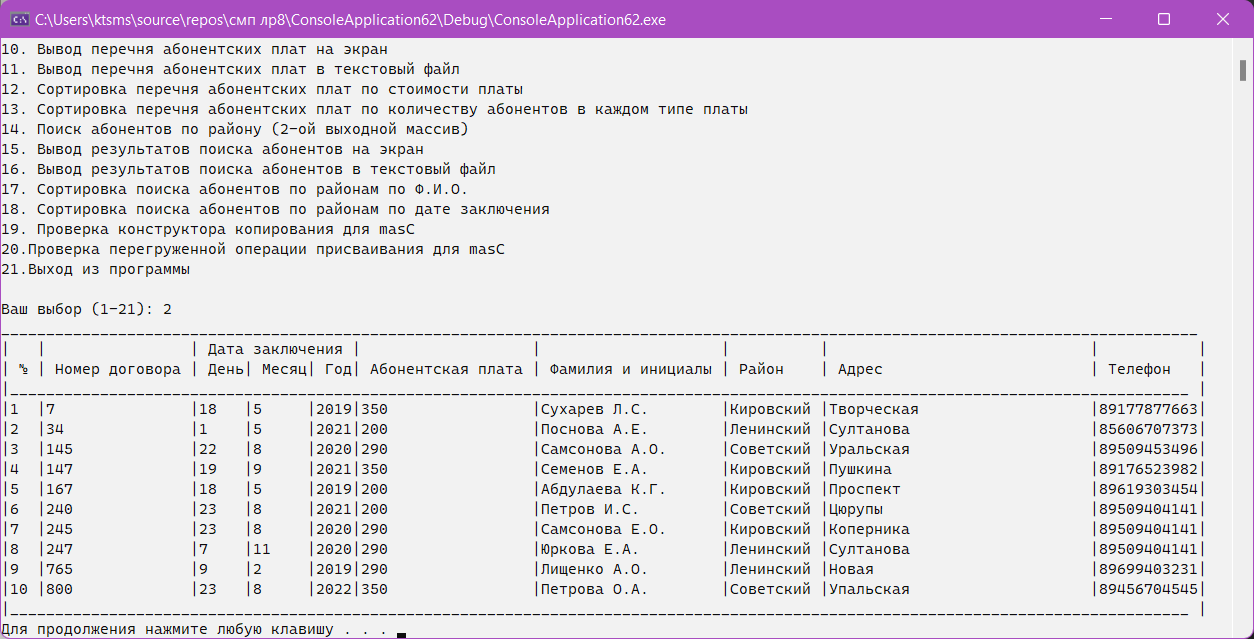
1. Сортировка по дате



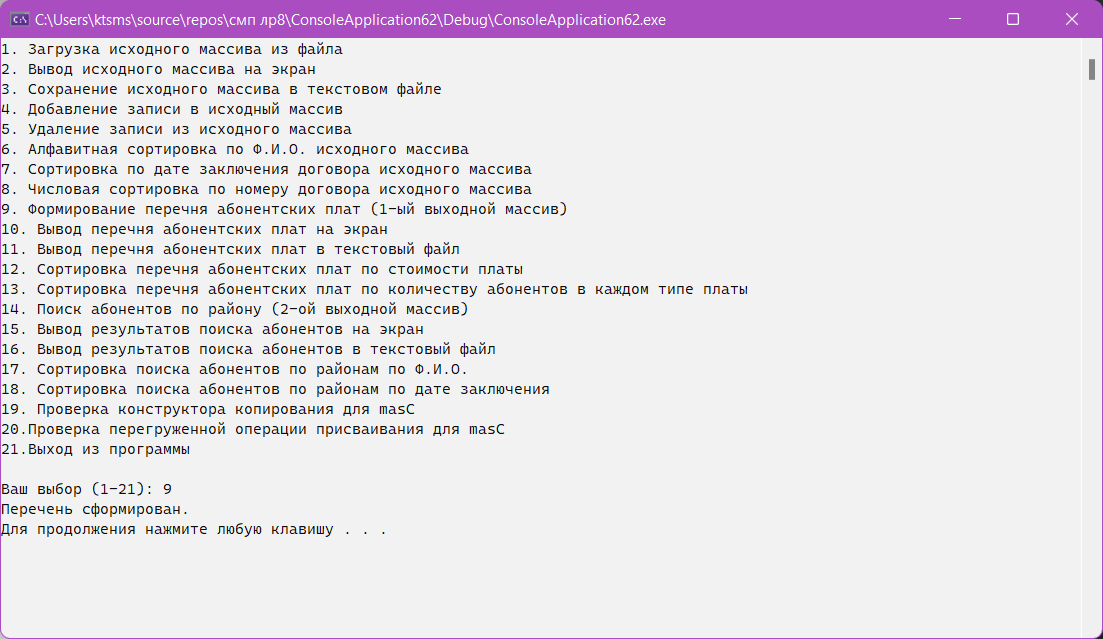


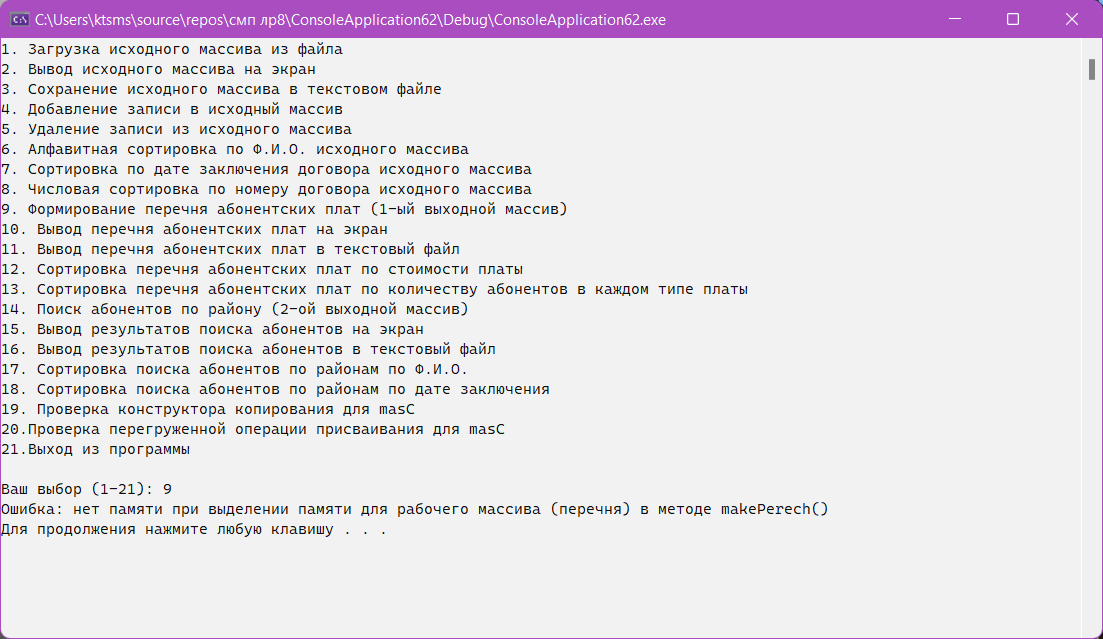
1. Сортировка по номеру договора

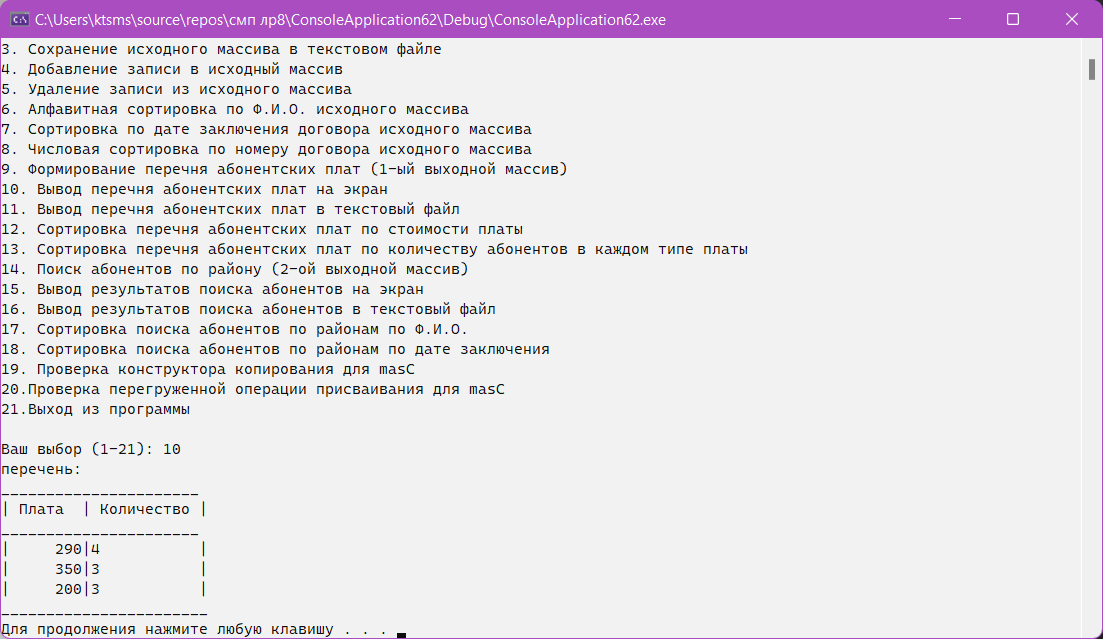


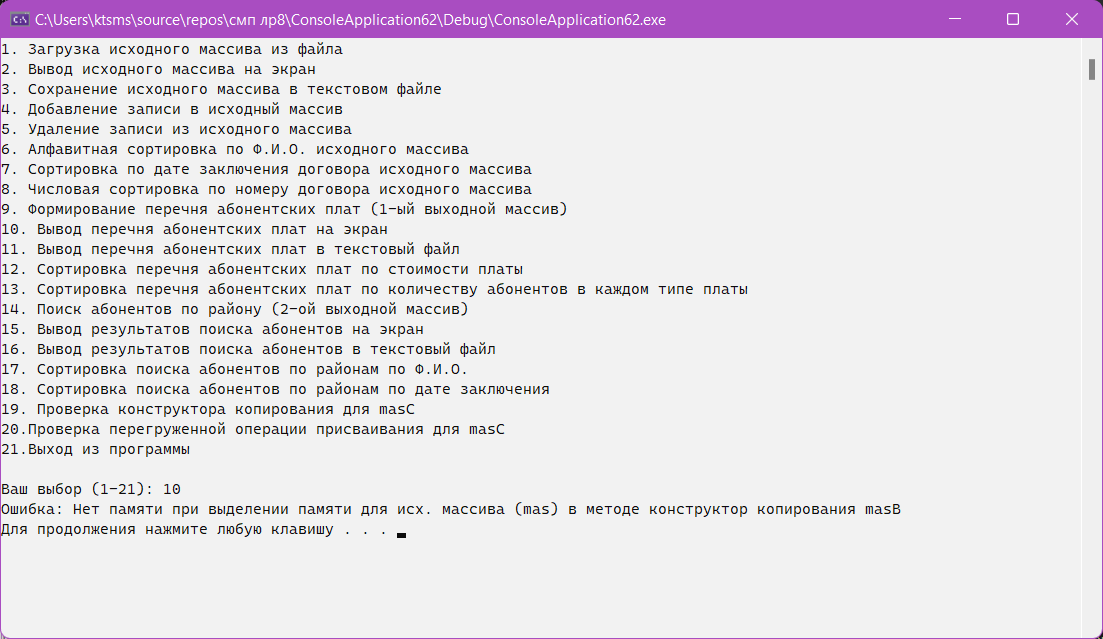


1. Формирование перечня

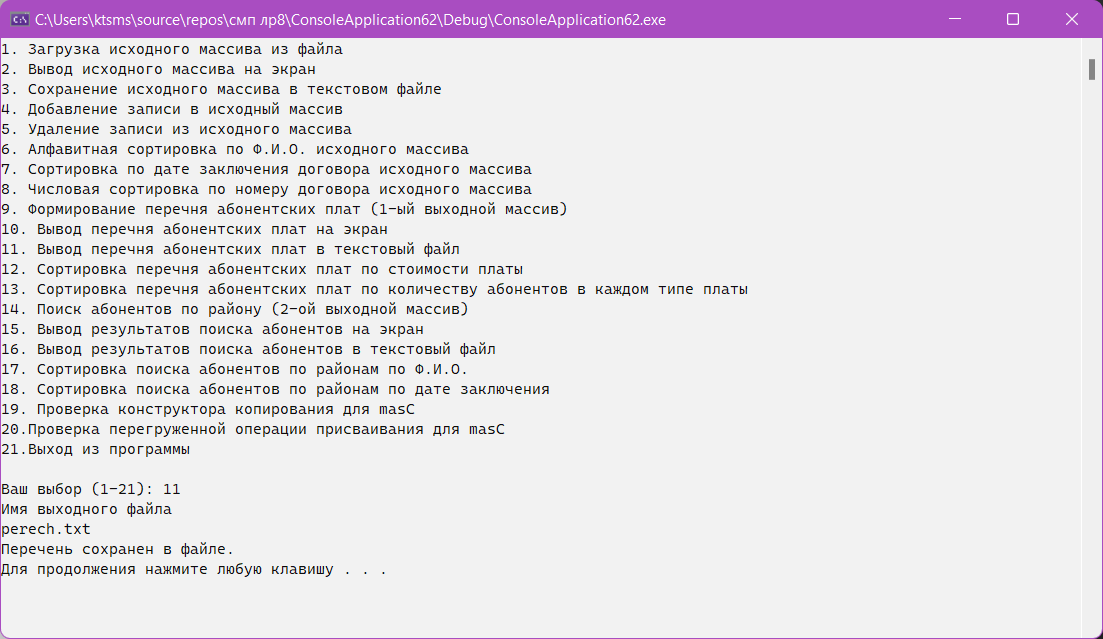


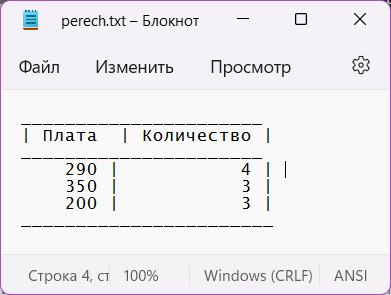




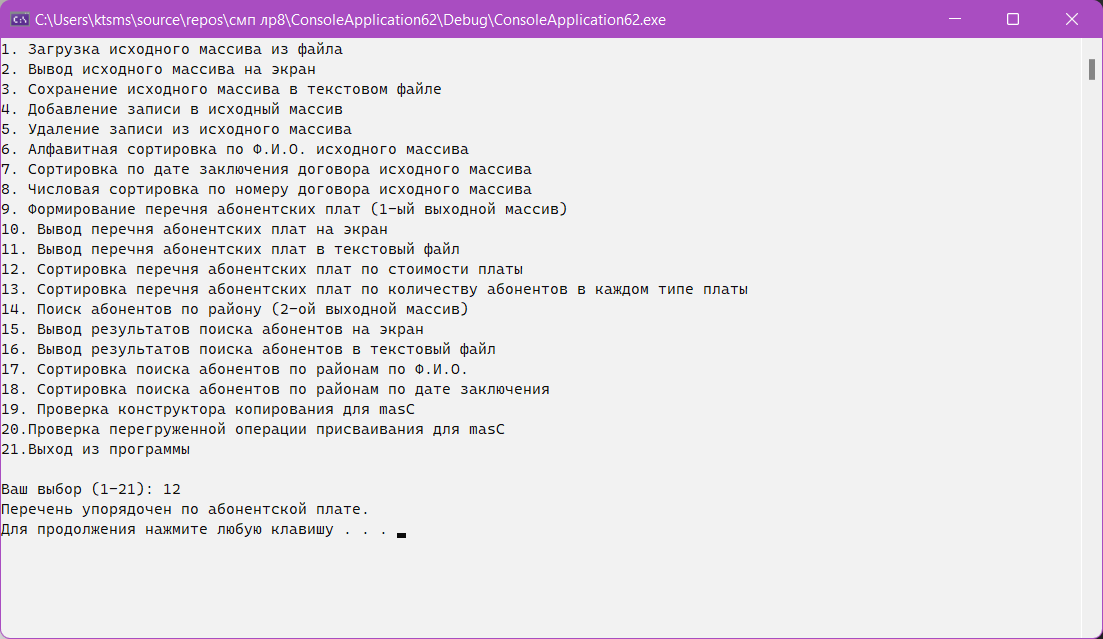


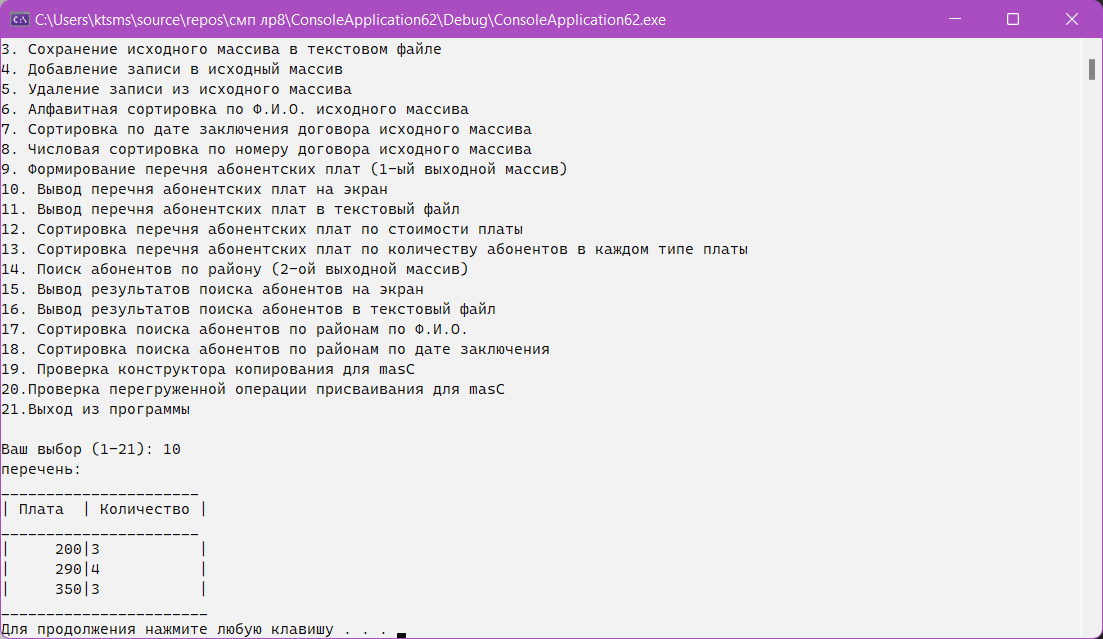
1. Сохранение перечня в файле



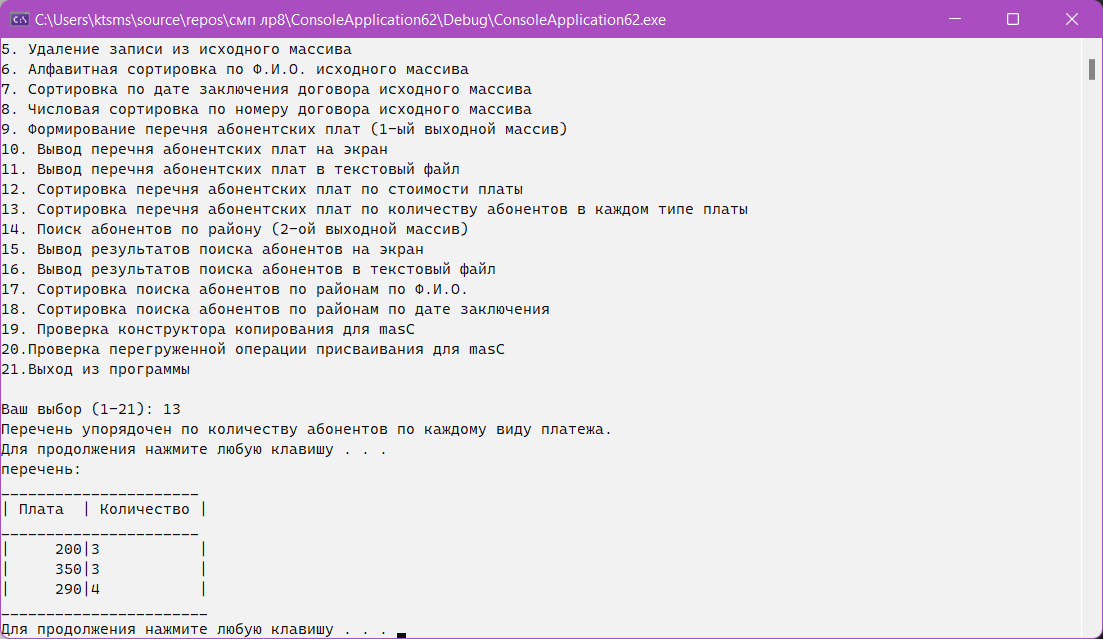


1. Сортировка перечня по плате

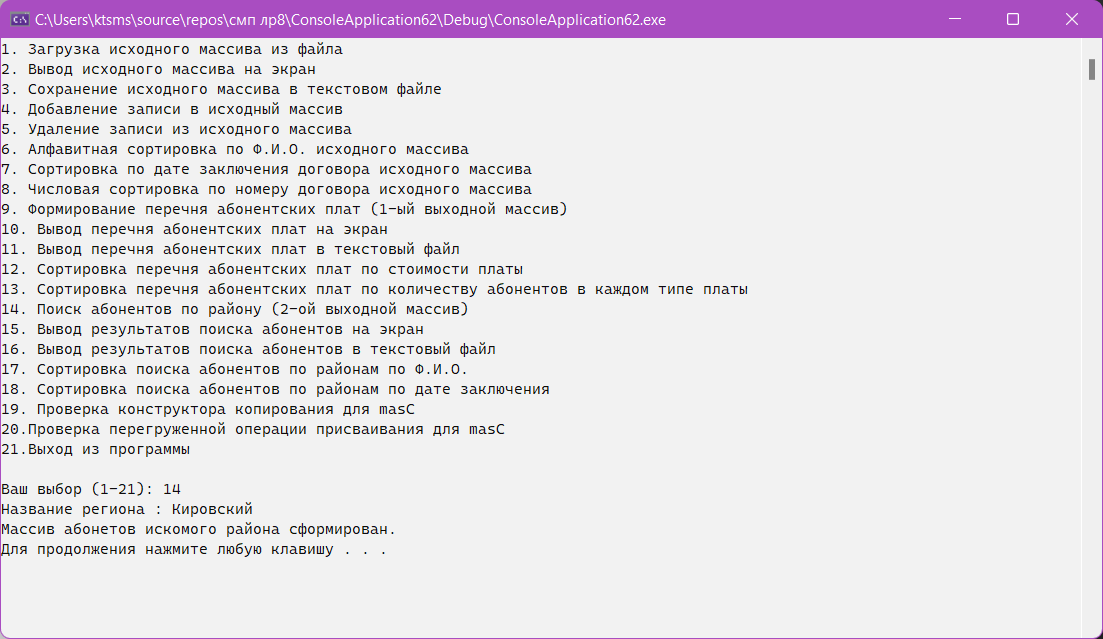


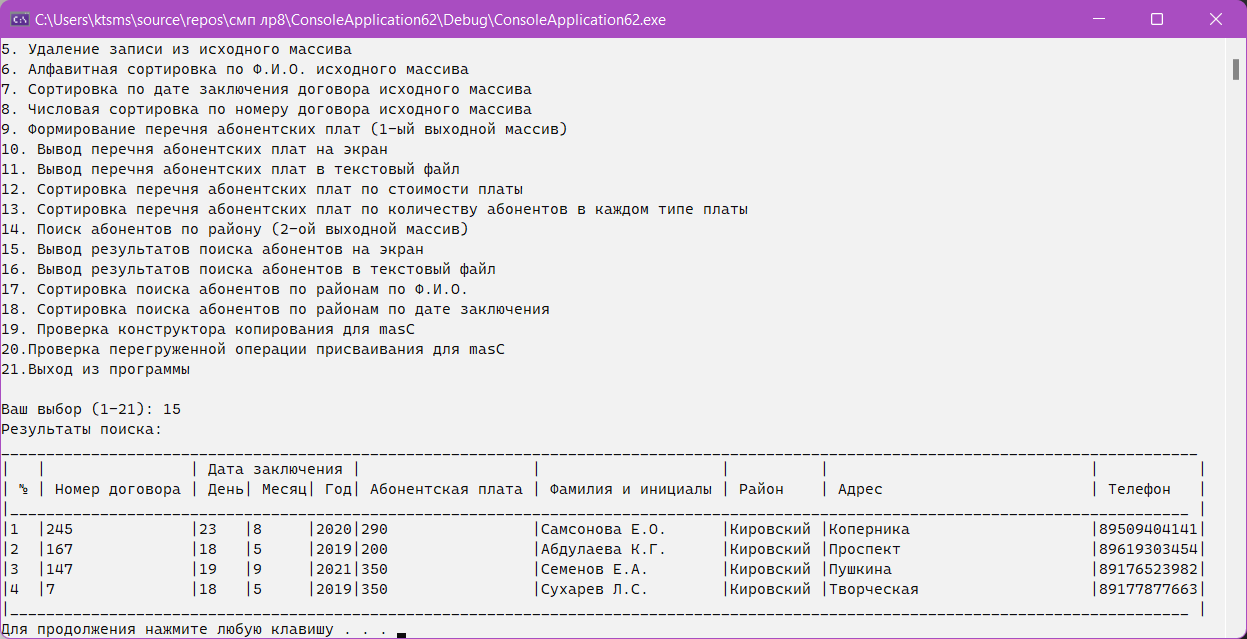


1. Сортировка перечня по количеству

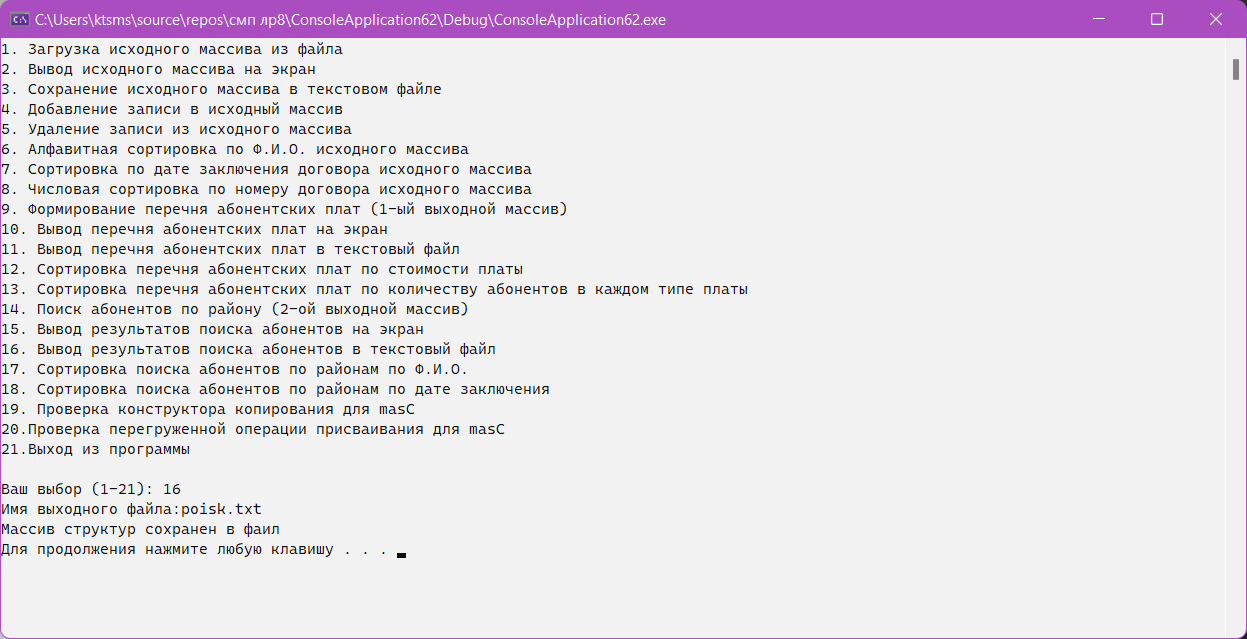


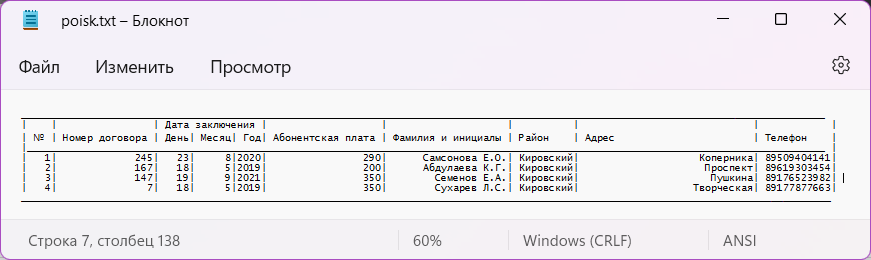
1. Поиск



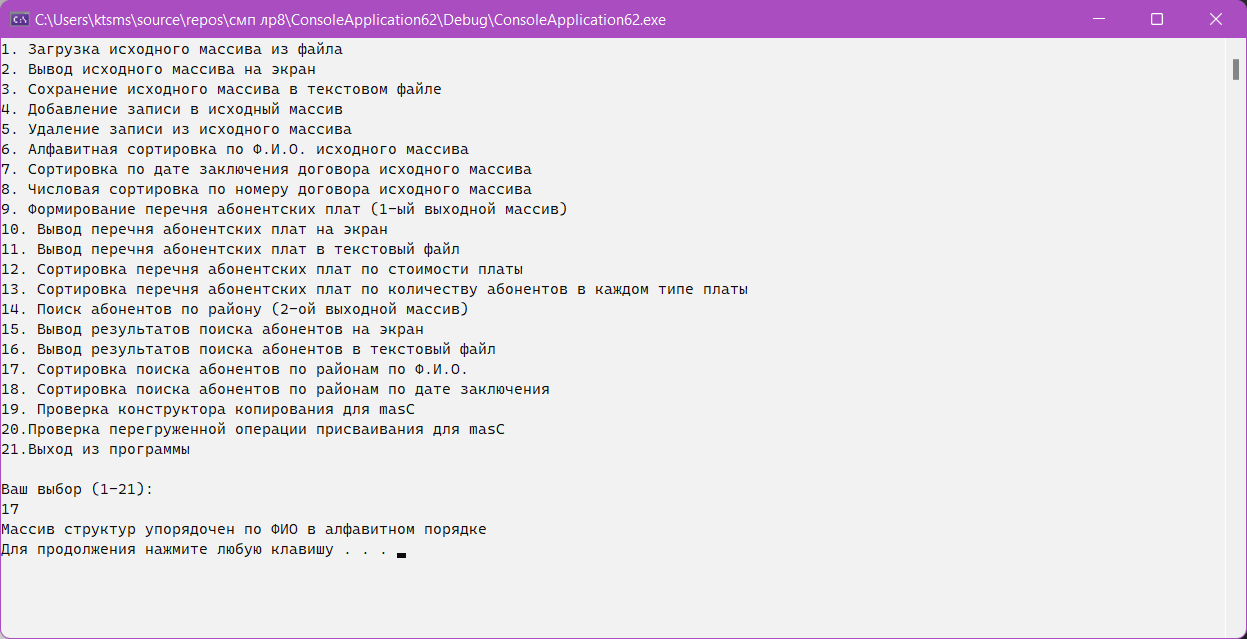


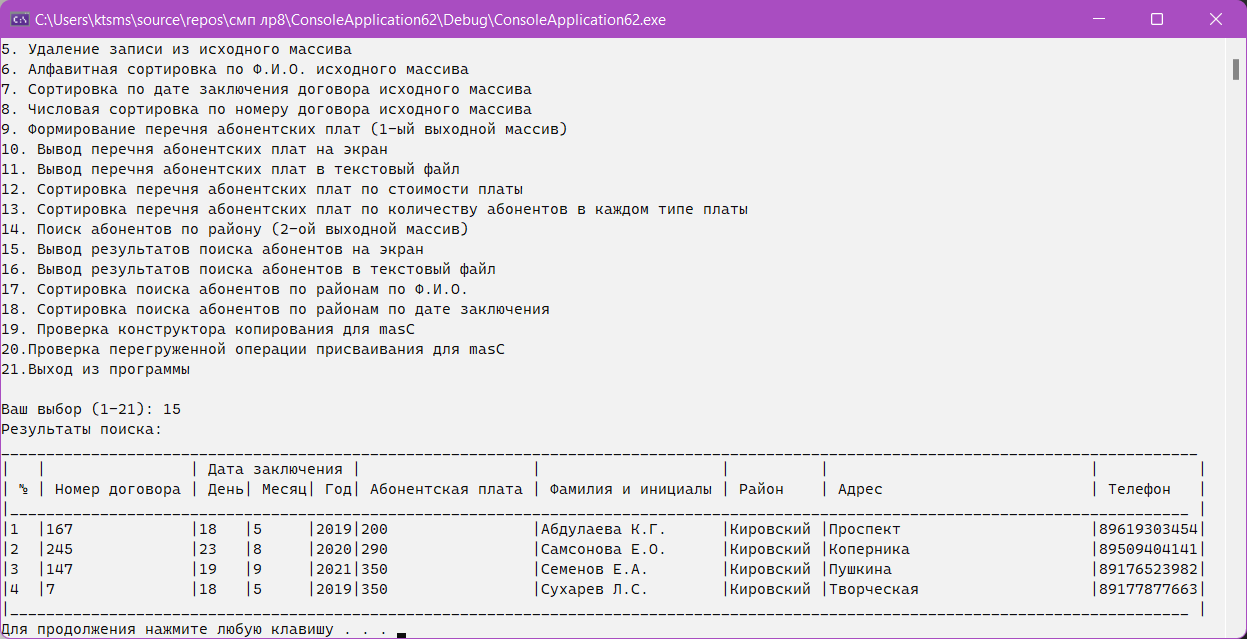
1. Сохранение результатов поиска в файл



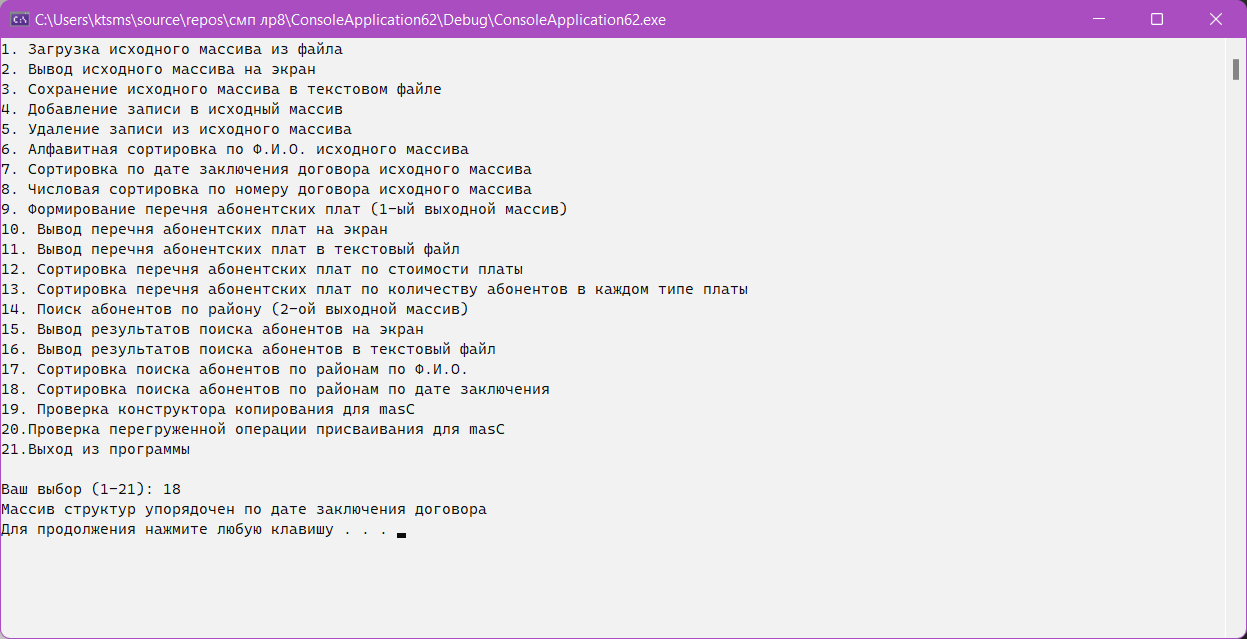


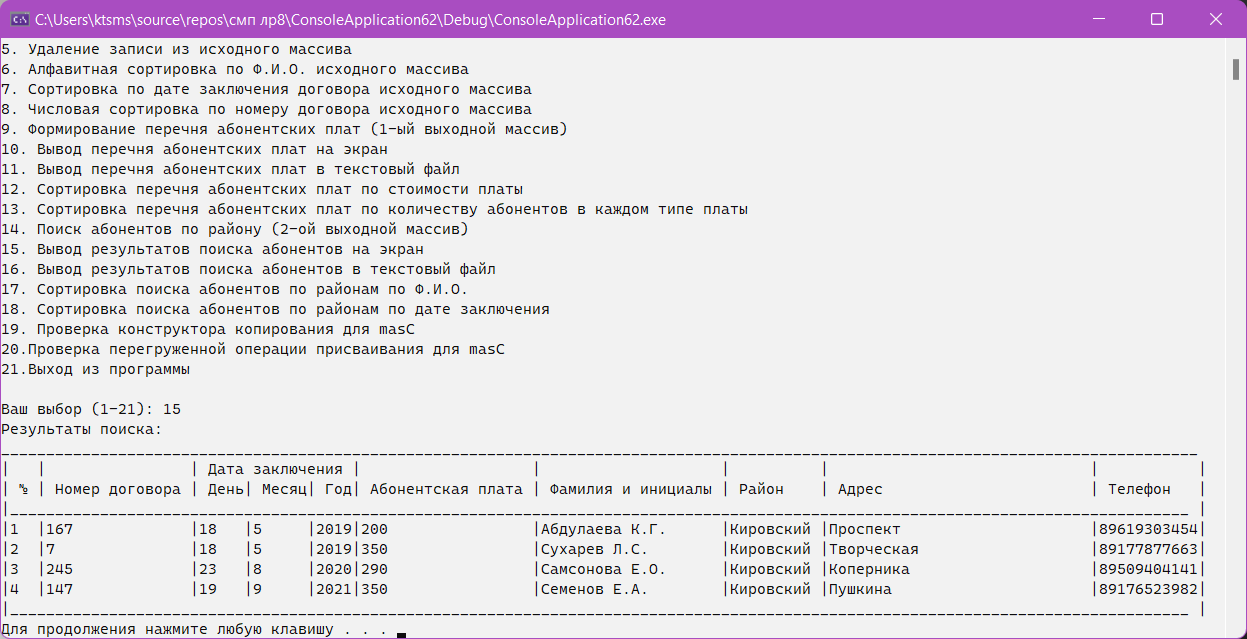
1. Сортировка поиска по ФИО



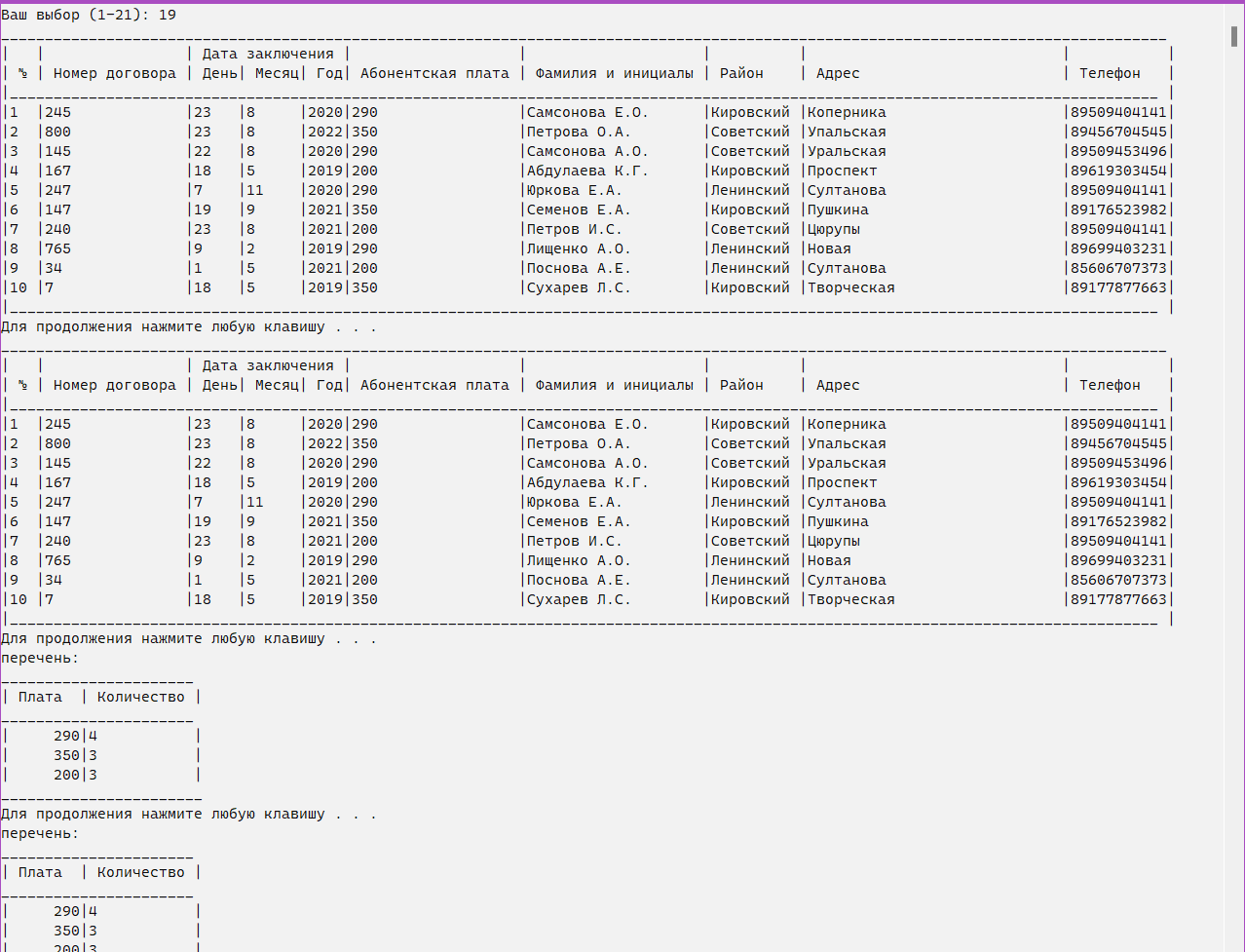


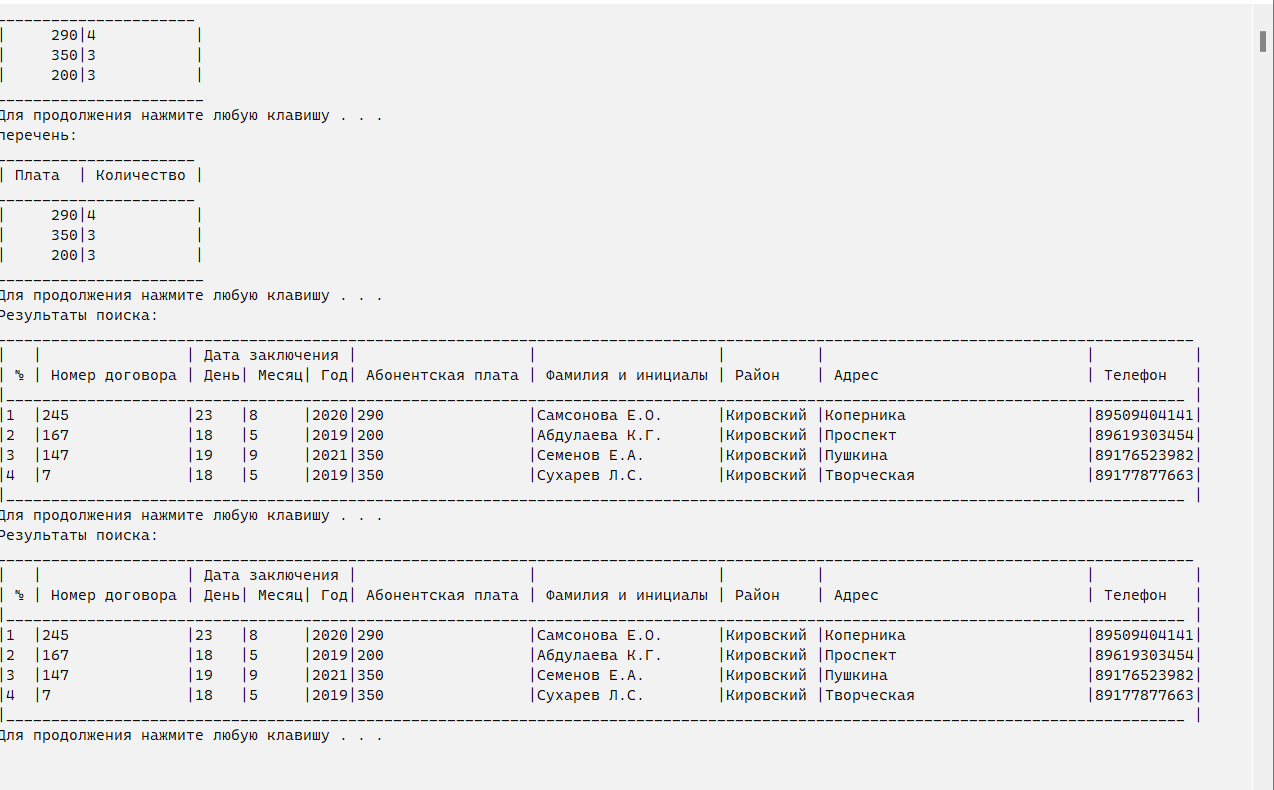
1. Сортировка поиска по дате



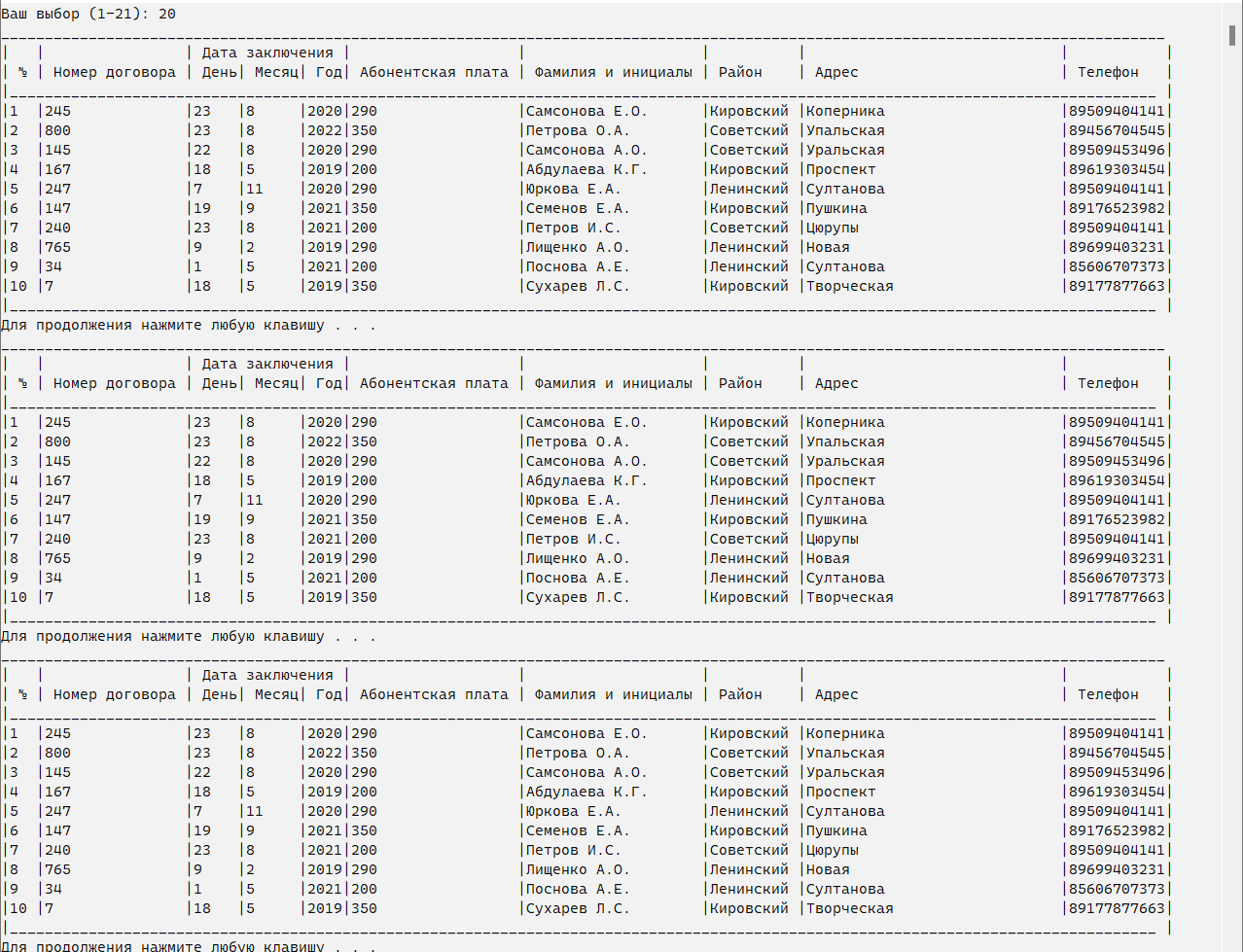


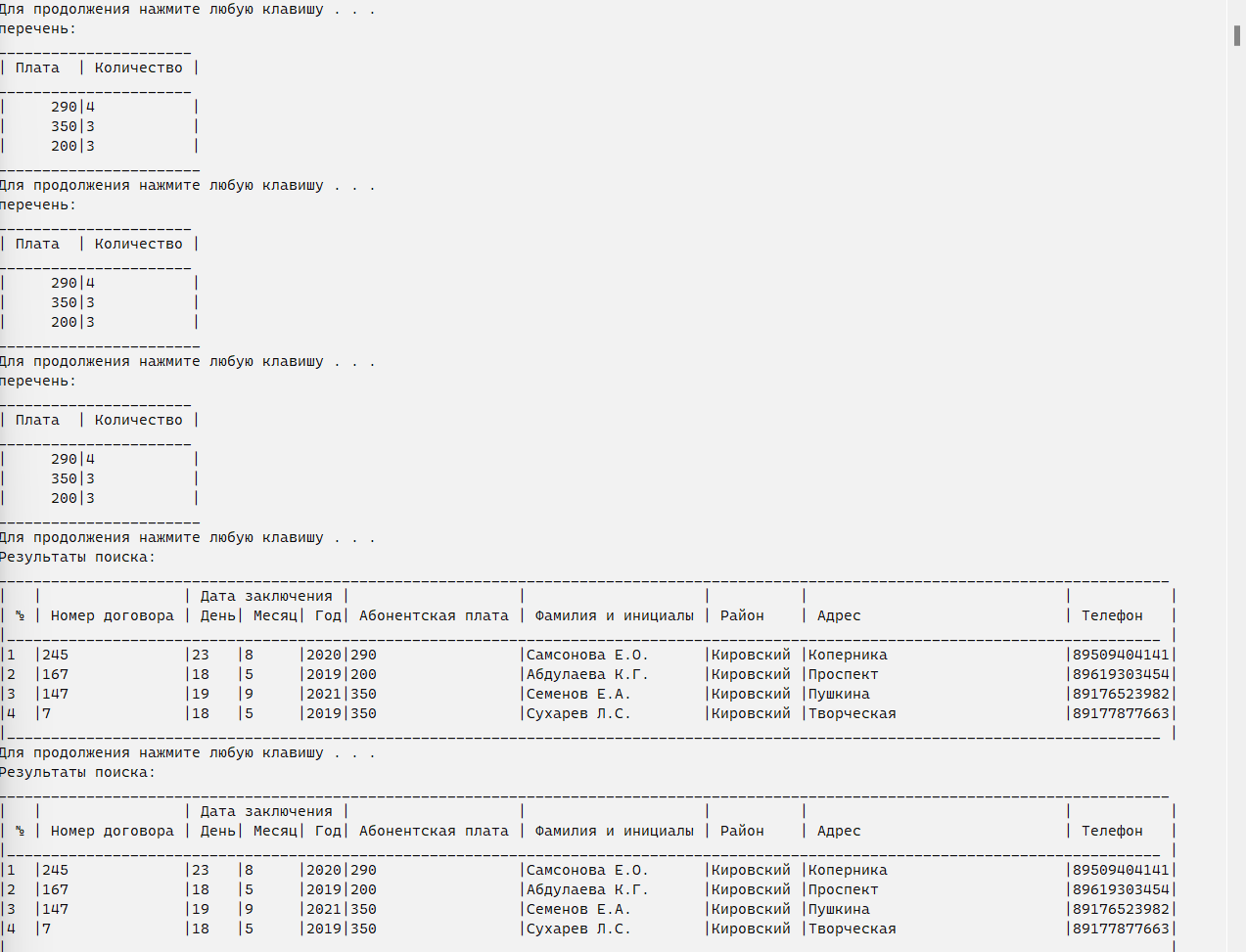
1. Проверка конструктора копирования

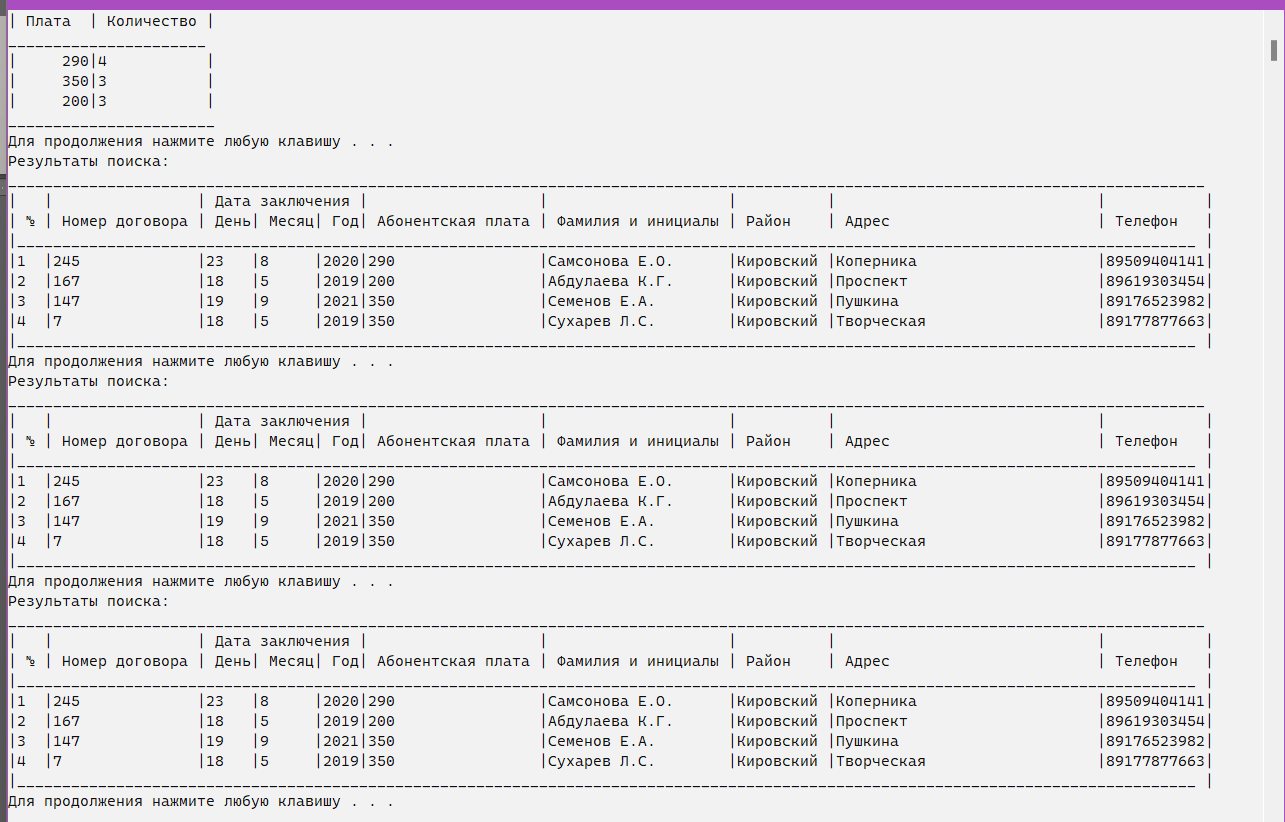




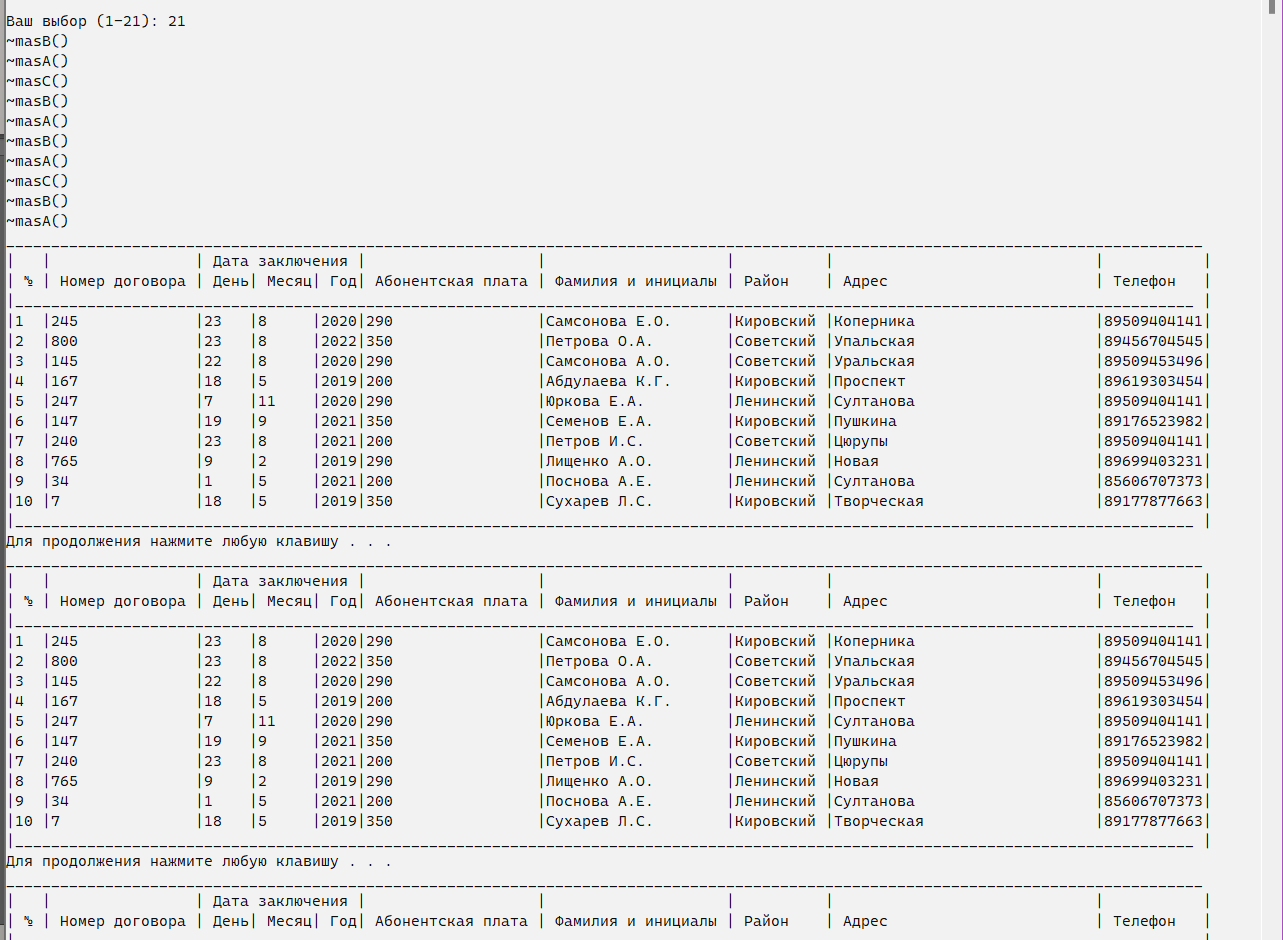
1. Проверка операции присваивания

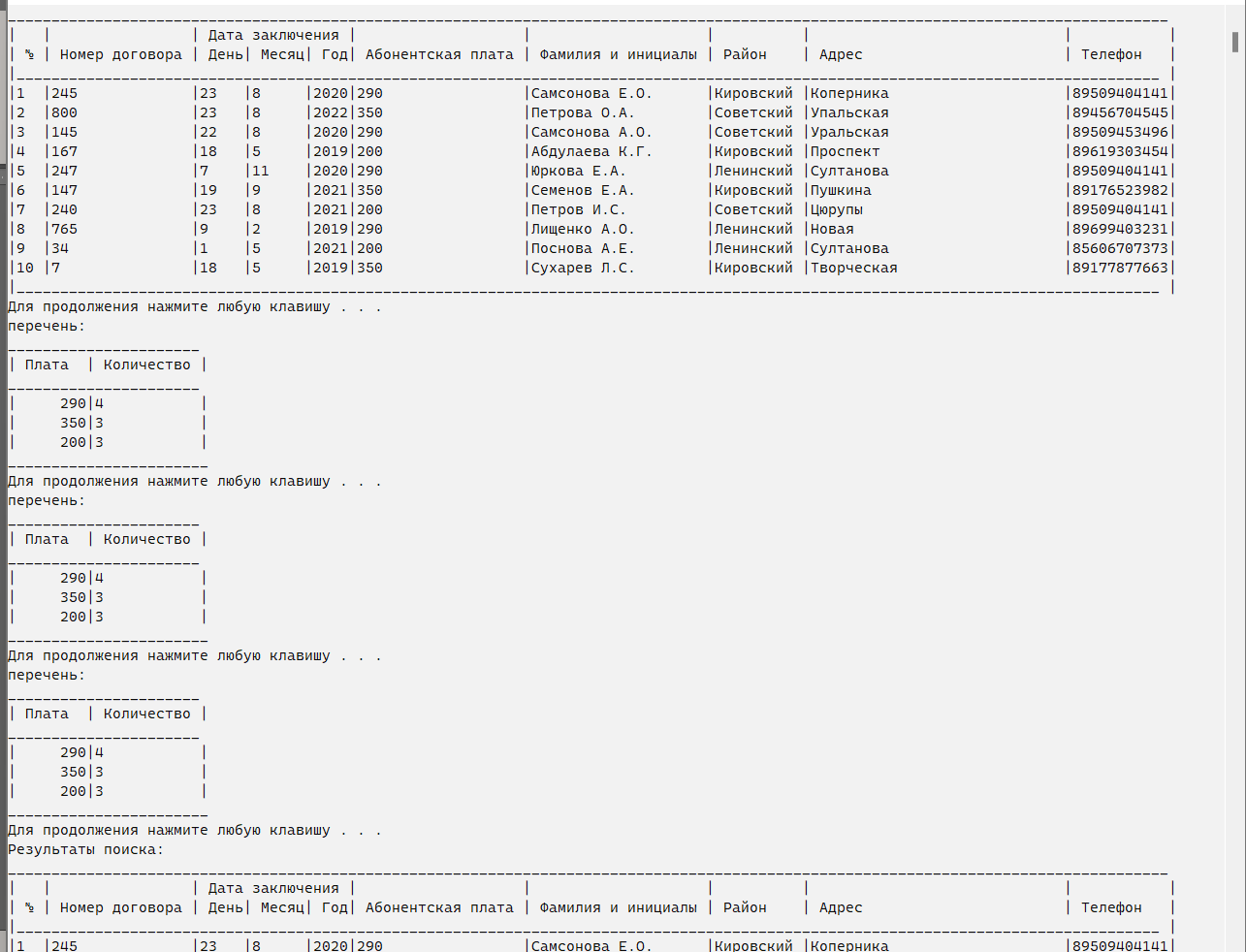


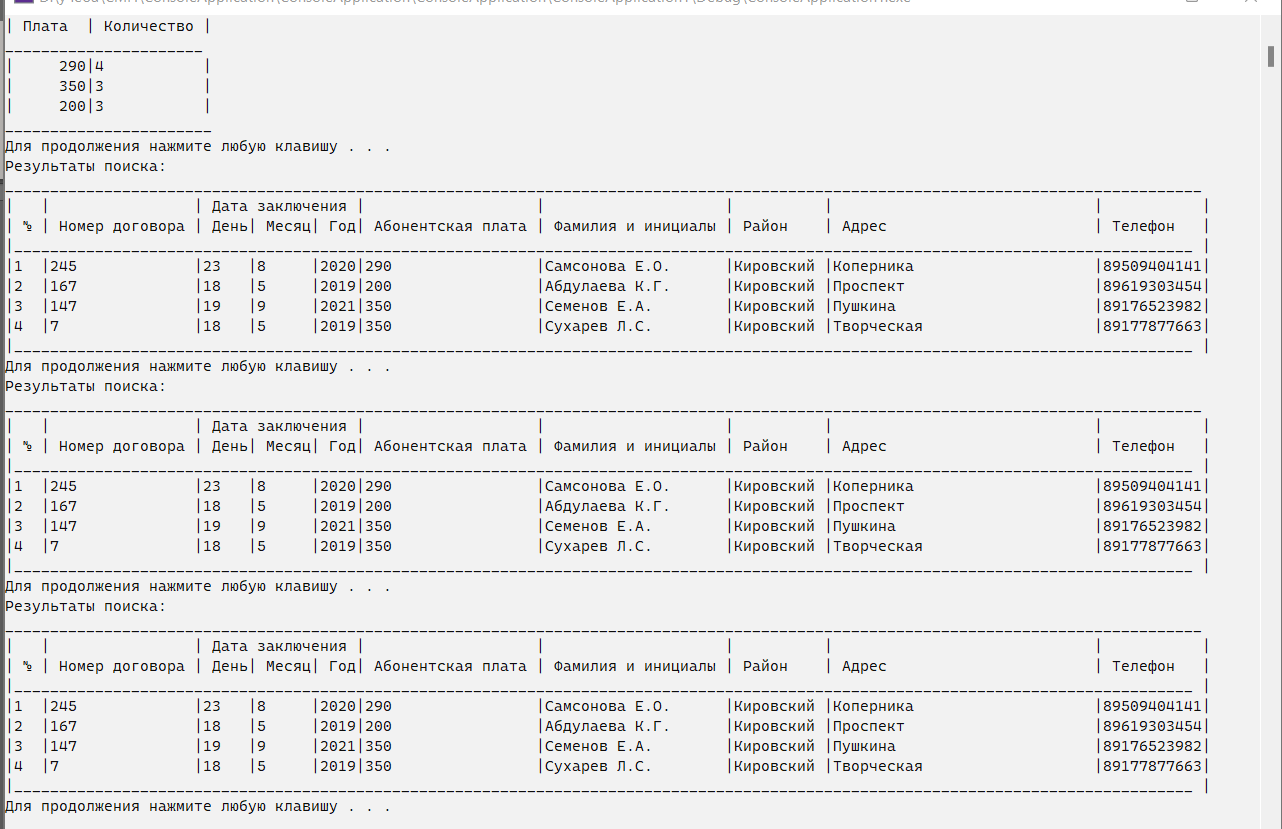




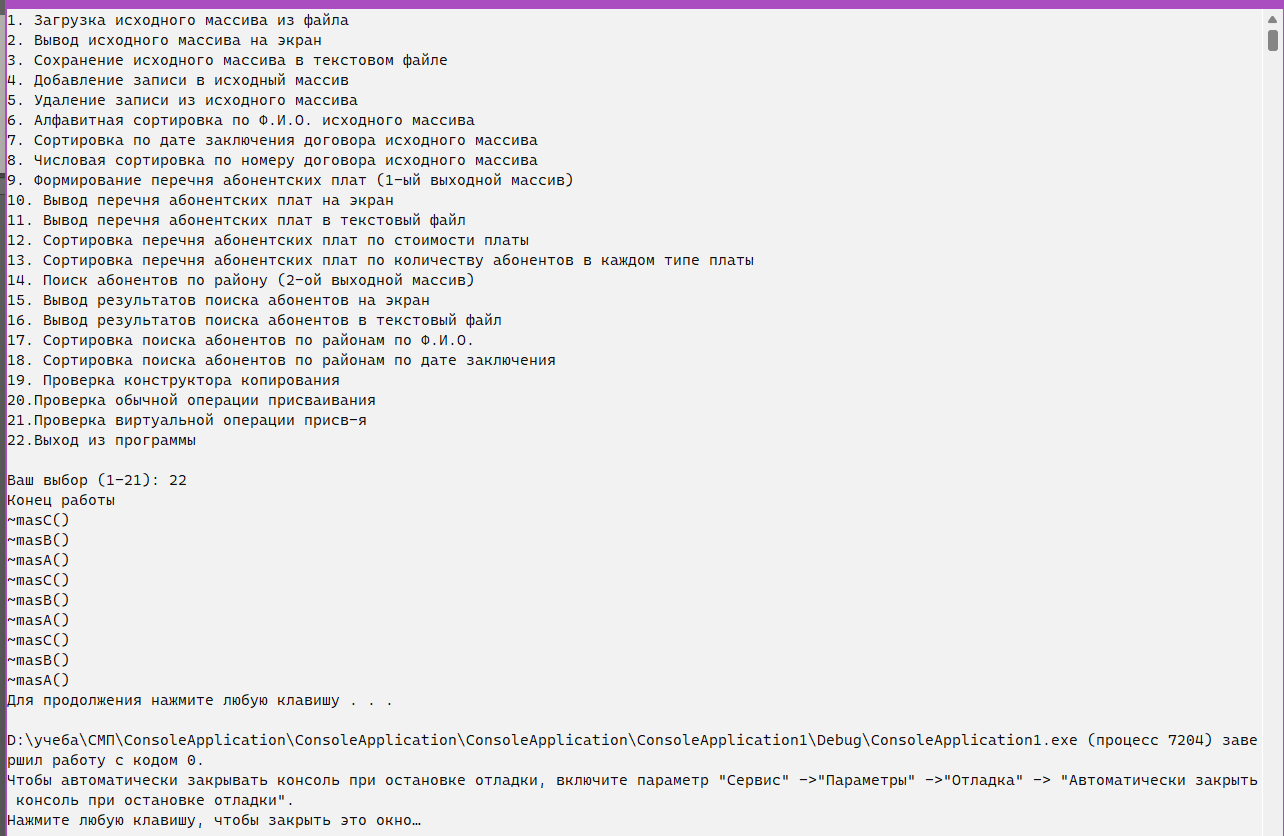
1. Проверка виртуальной операции присваивания







1. Выход из программы



Вывод: в ходе выполнения работы были получены навыки реализации программ проектирования класса, использующего свободную память (содержащего динамический массив структур) на языке *С++.*