ГУАП

КАФЕДРА № 14

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | А. Ю. Петров |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| РАБОТА С ПОТОКАМИ В С++ |
| по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1142 |  |  |  | А.Н. Коновалов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

1. **Постановка задачи**
2. Стандартные потоки

Определить класс с именем NOTE, содержащий следующие поля:

● фамилия, имя;

● номер телефона;

● день рождения.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа NOTE.

Заранее число объектов не известно.

Написать программу, выполняющую следующие действия:

● записи должны быть упорядочены по датам дней рождения;

● вывод на экран информации о человеке, номер телефона которого введен с клавиатуры;

● если такого нет, выдать на дисплей соответствующее сообщение.

1. Файловые и строковые потоки

С использованием файловых и строковых потоков написать программу,

которая считывает текст из файла и выводит на экран только предложения,

начинающиеся с тире, перед которым могут следовать только пробельные

символы

1. **Формализация задачи**

Программа написана в среде разработки Visual Studio 2022. Компилятор - Visual Studio 2022.

* 1. **CPP-файлы, входящие в программу:**

Main.cpp

KeeperN.cpp

KeeperT.cpp

Note.cpp

Senten.cpp

* 1. **Файлы заголовков, входящие в программу:**

KeeperN.h

KeeperT.h

Note.h

Senten.h

* 1. **Описание работы программы**

В файле TP2.cpp реализовано пользовательское меню, из которого конечный пользователь может производить манипуляции над объектами классов, написанных для данной работы. Также в этом файле реализована очистка консоли для упрощения пользования программой.

Через класс KeeperN реализован контейнер-список для хранения элементов класса Note. Работа контейнера сделана при помощи метода push\_back(), который помещает все новые элементы в конец списка, и pop(), который удаляет элемент по выбранному номеру в контейнере. Функция Show() выводит на экран содержимое контейнера. Перед выводом данных на экран используется метод sort(), который сортирует содержимое контейнера в алфавитном порядке на основе сортировки пузырьком. Также есть поле filter, по которому выводимые люди фильтруются по номеру телефона.

В классе Note хранится информация об одномчеловеке, а именно: имя, номер телефона и день рождения.День рождения используется для сравнения с другими людьми во время сортировки KeeperN.

Класс KeeperT реализует контейнер для строк, которые программа берёт из файла 1.txt. Во время считывания текста из файла программа ищет начало предложения (заглавную букву) и записывает предложение, пока не наткнётся на один из разделительных знаков (точка, вопросительный знак и т.д.). После считывания предложений, программа выводит предложения, начинающиеся с тире, перед которым находятся пробелы.

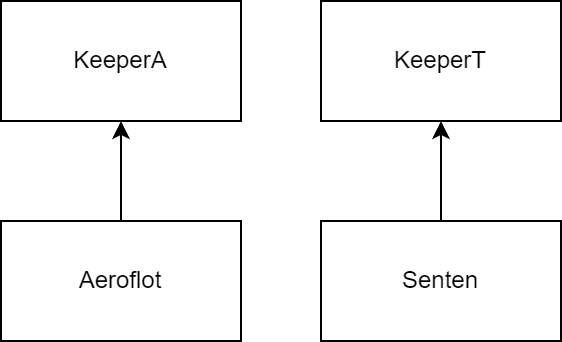
Класс Senten исполняет функцию строкового элемента контейнера KeeperT. Содержит в себе строковое поле, внутрь которого помещается предложение.

* 1. **Таблица классов**

|  |  |  |
| --- | --- | --- |
| Название класса | Назначение класса | Наследование |
| KeeperN | Контейнер-список для хранения данных о людях. | Отсутствует |
| Note | Элемент класса KeeperN. Хранит в себе информацию о пути, номере и самолёте рейса. | Отсутствует |
| KeeperT | Контейнер-список для хранения предложений, полученных из файла. | Отсутствует |
| Senten | Элемент класса KeeperT. Хранит в себе предложение, прочитанное из файла. | Отсутствует |

Таблица 1 – Таблица классов.

* 1. **Иерархия классов**

****

Note

KeeperN

Рисунок 1 – Схема иерархии классов

1. **Исходный код**

#include "KeeperN.h"

#include "KeeperT.h"

#include <list>

#include <iostream>

#define cles 1

using namespace std;

void menu();

void cls();

void men();

void disp();

void inc();

int main(void)

{

menu();

return 0;

}

void menu()

{

int choice = 0;

int k = 0;

int d = 0;

int filt = 0;

cls();

KeeperN bob(0);

KeeperN aero(bob);

KeeperT texo;

k = 0;

men();

//вывод в консоль

while (choice != 3)

{

cin >> choice;

cls();

switch (choice)

{

case 1: //////////////////////////////Open Note

{

aero.push\_back();

k = 0;

cls();

disp();

aero.Show();

while (k != 5)

{

cin >> k;

switch (k)

{

case 1:////Add element

cls();

disp();

aero.push\_back();

cls();

disp();

aero.Show();

break;

case 2://///Remove element

cls();

disp();

cout << "What entry you'd like to remove?" << endl;

cin >> d;

aero.pop(d);

cls();

disp();

aero.Show();

break;

case 3:///Show all entries

cls();

disp();

aero.setfilt();

aero.Show();

break;

case 4:///Filter

cls();

disp();

cout << "Who would you like to see? (Input their phone number)" << endl;

cin >> filt;

aero.setfilt(filt);

cls();

disp();

aero.Show();

break;

case 5:///To main menu

cls();

men();

break;

default:

cls();

disp();

inc();

break;

}

}

break;

}

case 2: //////////////////////////////Output edited text from file

{

cls();

men();

texo.Show();

break;

}

case 3://////////////////////////////Exit

{

cls();

men();

break;

}

default:

{

cls();

men();

inc();

break;

}

}

}

}

void men()

{

cout << "Menu:\n(1)Note\n(2)Output text from file\n(3)Exit"<< endl;

}

void disp()

{

cout << "(1)Add element\n(2)Remove element\n(3)Show all people\n(4)Find by phone number\n(5)To menu" << endl << endl;

}

void cls()

{

if (cles)

system("cls");

}

void inc()

{

cout << "Incorrect input!\n" << endl;

}

#pragma once

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class Senten

{

public:

Senten\* pNext;

Senten\* pPrev;

void setdat(int data = 0);

void settext(string text);

string gettext();

Senten(string data = " ", Senten\* pNext = nullptr, Senten\* pPrev = nullptr);

private:

int data;

string text;

};

#include "Senten.h"

void Senten::setdat(int data)

{

this->data = data;

}

void Senten::settext(string text)

{

this->text = text;

}

string Senten::gettext()

{

return this->text;

}

Senten::Senten(string data, Senten\* pNext, Senten\* pPrev)

{

this->data = 0;

this->pNext = pNext;

this->pPrev = pPrev;

this->text = data;

}

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Note

{

public:

Note\* pNext;

Note\* pPrev;

Note(int data = int(), Note\* pNext = nullptr, Note\* pPrev = nullptr);

Note(string temp);

void setname(string dest="anton");

void setdat(int dat = 0);

void setbd(int d = 0, int m = 0, int y = 0);

string getname();

int getdat();

int& getbday();

friend std::istream& operator>>(std::istream& input, Note& no);

friend std::ostream& operator<<(std::ostream& output, const Note& no);

private:

string name;

int data;

int bday[3];

};

#include "Note.h"

Note::Note(int data, Note\* pNext, Note\* pPrev)

{

this->data = data;

this->pNext = pNext;

this->pPrev = pPrev;

setname();

setbd();

cin >> \*this;

}

Note::Note(string temp)

{

this->data = data;

this->pNext = pNext;

this->pPrev = pPrev;

setname();

setbd();

}

void Note::setname(string name)

{

this->name = name;

}

void Note::setdat(int dat)

{

this->data = dat;

}

void Note::setbd(int d, int m, int y)

{

this->bday[0] = d;

this->bday[1] = m;

this->bday[2] = y;

}

string Note::getname()

{

return this->name;

}

int Note::getdat()

{

return this->data;

}

int& Note::getbday()

{

return \*this->bday;

}

istream& operator>>(std::istream& input, Note& no)

{

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

string name;

string surname;

int day = 99;

int month = 99;

int year = -1;

int num = 0;

cout << "\nName: ";

getline(input, name);

cout << "\nSurname: ";

getline(input, surname);

no.setname(name + " " + surname);

cout << "\nPhone number: ";

cin >> num;

no.setdat(num);

while(day>30 || day<1)

{

cout << "\nDay of birth: ";

cin >> day;

}

while(month>12 || month<1)

{

cout << "\nMonth of birth: ";

cin >> month;

}

while (year < 0)

{

cout << "\nYear of birth: ";

cin >> year;

}

no.setbd(day, month, year);

return input;

}

ostream& operator<<(ostream& output, const Note& no)

{

output << "Name: " << no.name << endl<<"Phone number: "<<no.data << endl <<"Birtday: " << no.bday[0] << "-" << no.bday[1] << "-" << no.bday[2] << endl << endl;

return output;

}

#pragma once

#include <iostream>

#include <string>

#include "Senten.h"

using namespace std;

class KeeperT

{

public:

KeeperT();

~KeeperT();

void pop\_front();

void push\_back(string data);

void clear();

int GetSize();

int request\_strt();

string operator[](const int index);

void Show();

private:

int Size;

Senten\* head;

Senten\* tail;

};

#include "KeeperT.h"

#include<iostream>

#include <conio.h>

#include <windows.h>

class Anyof

{

public:

bool operator()(char\* arr, char symbol)

{

for (int i = 0; arr[i] != '\0'; i++) {

if (arr[i] == symbol) {

return true;

}

}

return false;

}

bool operator()(char symbol, char\* arr)

{

return anyof(arr, symbol);

}

bool operator()(std::string arr, char symbol)

{

for (int i = 0; arr[i] != '\0'; i++) {

if (arr[i] == symbol) {

return true;

}

}

return false;

}

bool operator()(char symbol, std::string arr)

{

return anyof(arr, symbol);

}

}anyof;

KeeperT::KeeperT()

{

Size = 0;

head = nullptr;

tail = nullptr;

int phase = 0;

ifstream fin("1.txt");

string str = "";

char separator[] = { '.', '?', '!', '…', '\0' };

char first\_let[27] = { 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};

char symbol;

char prevsym = ' ';

while (fin.get(symbol))

{

switch (phase)

{

case 0://Поиск Тире

if (symbol == 45 && prevsym == 32)

{

str += symbol;

phase = 1;

}

break;

case 1://Поиск заглавной

if (anyof(first\_let, symbol))

{

str += symbol;

phase = 2;

}

else

{

str = "";

phase = 0;

}

break;

case 2://Запись предложения

str += symbol;

if (anyof(separator, symbol))

{

phase = 3;

}

break;

case 3://Поиск пробела

if (symbol == 32)

{

this->push\_back(str);

str = "";

phase = 0;

}

break;

}

prevsym = symbol;

}

}

KeeperT::~KeeperT()

{

clear();

}

void KeeperT::pop\_front()

{

int si = this->Size;

if (si)//Фикс исключения 1

{

Senten\* temp = head;

head = head->pNext;//Исключение 1: попытка чтения поля пустого элемента

delete temp;

Size--;

}

else

{

cout << "Bad Input: cannot delete an element of an empty container" << endl;

Sleep(1000);

}

}

void KeeperT::push\_back(string data)

{

if (head == nullptr)

{

head = new Senten(data);

tail = this->head;

}

else

{

Senten\* current = this->head;

while (current->pNext != nullptr)

{

current = current->pNext;

}

current->pNext = new Senten(data, current->pNext, current);

tail = current->pNext;

}

Size++;

}

void KeeperT::clear()

{

while (Size)

{

pop\_front();

}

}

int KeeperT::GetSize()

{

return Size;

}

int KeeperT::request\_strt()

{

int strt;

cout << "Input container length: ";

cin >> strt;

return strt;

}

string KeeperT::operator[](const int index)

{

int counter = 0;

string ret;

Senten\* current = this->head;

while (current != nullptr)

{

if (counter == index)

{

ret = current->gettext();

return ret;

}

current = current->pNext;

counter++;

}

}

void KeeperT::Show()

{

Senten\* current = this->head;

int check = 0;

cout << endl;

while (current != nullptr)

{

cout << current->gettext() << endl;

current = current->pNext;

check++;

}

if (!check)

cout << "No sentences alligning with set criterias were found!" << endl;

}

#pragma once

#include "Note.h"

#include <iostream>

#include <string>

using namespace std;

class KeeperN

{

public:

KeeperN();

explicit KeeperN(int siz);

~KeeperN();

KeeperN(const KeeperN& other);

void swap(int fir, int sec);

void sort();

void pop\_front();

void pop\_back();

void pop(int d);

void push\_back(int data = rand() % 1000);

void clear();

void setfilt(int filter = 0);

int GetSize();

int request\_strt();

void Spawn();

void Spawn(int siz);

string operator[](int index);

void test();

void Show();

private:

int Size;

int filter;

Note\* head;

Note\* tail;

};

#include "KeeperN.h"

#include<iostream>

#include <conio.h>

#include <windows.h>

KeeperN::KeeperN()

{

Size = 0;

head = nullptr;

tail = nullptr;

filter = 0;

}

KeeperN::KeeperN(int siz)

{

Size = 0;

head = nullptr;

tail = nullptr;

filter = 0;

Spawn(siz);

}

KeeperN::~KeeperN()

{

clear();

}

KeeperN::KeeperN(const KeeperN& other)

{

int cnt = 0;

this->Size = other.Size;

Note\* current = nullptr;

Note\* current1 = nullptr;

this->head = current;

filter = 0;

for (cnt = 0; cnt < this->Size; cnt++)

{

if (head == nullptr)

{

this->head = new Note(other.head->getdat());

this->tail = this->head;

}

else

{

current = other.head;

current1 = this->head;

while (current1->pNext != nullptr)

{

current = current->pNext;

current1 = current1->pNext;

}

current1->pNext = new Note(current->pNext->getdat(), current1->pNext, current1);

this->tail = current1->pNext;

}

}

}

void KeeperN::swap(int fir, int sec)

{

Note\* first = this->head;

Note\* second = this->head;

int cntf = 0;

int cnts = 0;

int d = 0, m = 0, y = 0;

int\* arr;

while (first != nullptr)

{

if (cntf == fir)

{

break;

}

first = first->pNext;

cntf++;

}

while (second != nullptr)

{

if (cnts == sec)

{

break;

}

second = second->pNext;

cnts++;

}

Note\* temp = new Note("temp");

temp->setname(first->getname());

temp->setdat(first->getdat());

arr = &first->getbday();

d = arr[0];

m = arr[1];

y = arr[2];

temp->setbd(d, m, y);

first->setname(second->getname());

first->setdat(second->getdat());

arr = &second->getbday();

d = arr[0];

m = arr[1];

y = arr[2];

first->setbd(d, m, y);

second->setname(temp->getname());

second->setdat(temp->getdat());

arr = &temp->getbday();

d = arr[0];

m = arr[1];

y = arr[2];

second->setbd(d, m, y);

delete temp;

}

void KeeperN::sort()

{

int i, j;

Note\* fir = this->head;

Note\* sec = fir->pNext;

int\* first;

int\* second;

int dat1;

int dat2;

for (i = 0; i < Size - 1; i++)

{

fir = this->head;

sec = fir->pNext;

for (j = 0; j < Size - i - 1; j++)

{

first = &fir->getbday();

second = &sec->getbday();

dat1 = first[1] \* 30 + first[0];

dat2 = second[1] \* 30 + second[0];

if ((dat1 > dat2))

this->swap(j, j + 1);

fir = fir->pNext;

sec = fir->pNext;

}

}

}

void KeeperN::pop\_front()

{

int si = this->Size;

if (si)//Фикс исключения 1

{

Note\* temp = head;

head = head->pNext;//Исключение 1: попытка чтения поля пустого элемента

delete temp;

Size--;

}

else

{

cout << "Bad Input: cannot delete an element of an empty container" << endl;

Sleep(1000);

}

}

void KeeperN::pop\_back()

{

int si = this->Size;

if (si)//Фикс исключения 2

{

Note\* temp = tail;

tail = tail->pPrev;//Исключение 2: попытка чтения поля пустого элемента

if (si > 1)

{

tail->pNext = nullptr;

delete temp;

Size--;

}

else

{

this->head = nullptr;

this->Size = 0;

}

}

else

{

cout << "Bad Input: cannot delete an element of an empty container" << endl;

Sleep(1000);

}

}

void KeeperN::pop(int d)

{

int si = this->Size;

int i = 0;

if (d<0 || d>si - 1)

{

cout << "Incorrect range!" << endl;

Sleep(1000);

}

else if (d == 0)

this->pop\_front();

else if (d == si - 1)

this->pop\_back();

else

{

Note\* temp = this->head;

Note\* tem1 = nullptr;

while (i != d)

{

tem1 = temp;

temp = temp->pNext;

temp->pPrev = tem1;

i++;

}

temp->pPrev->pNext = temp->pNext;

temp->pNext->pPrev = temp->pPrev;

delete temp;

Size--;

}

}

void KeeperN::push\_back(int data)

{

if (head == nullptr)

{

head = new Note(data);

tail = this->head;

}

else

{

Note\* current = this->head;

while (current->pNext != nullptr)

{

current = current->pNext;

}

current->pNext = new Note(data, current->pNext, current);

tail = current->pNext;

}

Size++;

}

void KeeperN::clear()

{

while (Size)

{

pop\_front();

}

}

void KeeperN::setfilt(int filter)

{

this->filter = filter;

}

int KeeperN::GetSize()

{

return Size;

}

int KeeperN::request\_strt()

{

int strt;

cout << "Input container length: ";

cin >> strt;

return strt;

}

void KeeperN::Spawn()

{

int strt = request\_strt();

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

{

push\_back(rand() % 100);

}

}

void KeeperN::Spawn(int siz)

{

int strt = siz;

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

{

push\_back(rand() % 100);

}

}

string KeeperN::operator[](const int index)

{

int counter = 0;

string ret;

Note\* current = this->head;

while (current != nullptr)

{

if (counter == index)

{

ret = current->getname();

return ret;

}

current = current->pNext;

counter++;

}

}

void KeeperN::test()

{

Note\* current = this->head;

int \*a = &current->getbday();

cout << a[0] << " " << a[1] << " " << a[2] << " " << endl;

}

void KeeperN::Show()

{

if (filter == 0)

cout << "Currently showing all entries" << endl << endl;

else

cout << "Currently showing entry with the following phone number: "<< filter << endl << endl;

Note\* current = this->head;

int cur\_num;

this->sort();

int check = 0;

if (current != nullptr)

{

while (current->pNext != nullptr)

{

cur\_num = current->getdat();

if (filter != 0 && cur\_num == filter)

{

cout << \*current;

check++;

}

else if (filter == 0)

cout << \*current;

current = current->pNext;

}

cur\_num = current->getdat();

if (filter != 0 && cur\_num == filter)

{

cout << \*current;

check++;

}

else if (filter == 0)

cout << \*current;

if (!check && filter != 0)

cout << "No entries with the following number were found!" << endl;

}

else

cout << "The list is empty!!!" << endl;

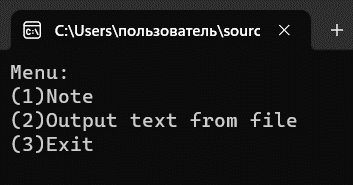
}

* 1. **GitHub**

Ссылка на репозиторий GitHub: [Theomur/Programming\_tech: задания 5 семестра по предмету технологии программирования (github.com)](https://github.com/Theomur/Programming_tech)

1. **Результаты работы программы**

Ниже приведены скриншоты с примерами работы программы. Чтобы проверить все функциональный возможности программы, пройдемся по каждой возможной ветке работы программы.



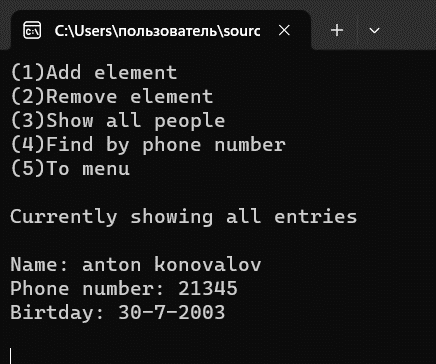
Рисунок 2 – Главное меню

Рисунок 3 – Открытие меню Note просит пользователя ввести данный первого человека

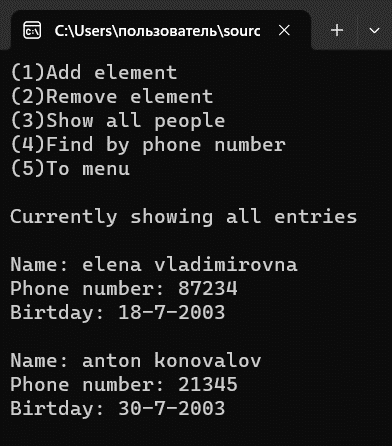


Рисунок 4 – Добавление нового человека. Причём новая запись выводится перед старой, потому что идёт сортировка по полю Birthday.

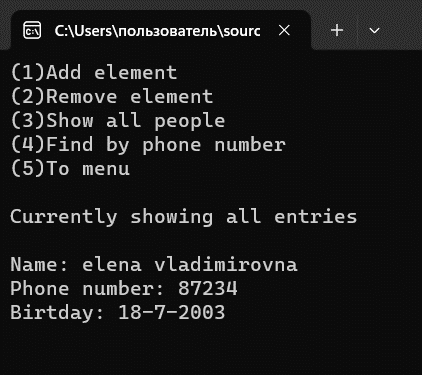


Рисунок 5 – Удаление человека

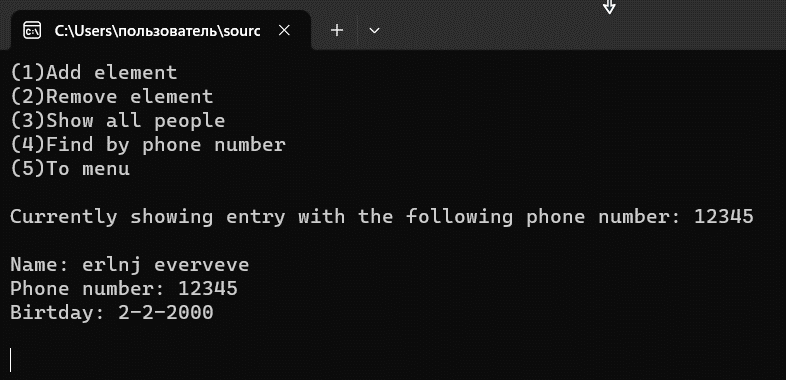


Рисунок 6 – Фильтрация людей по номеру телефона

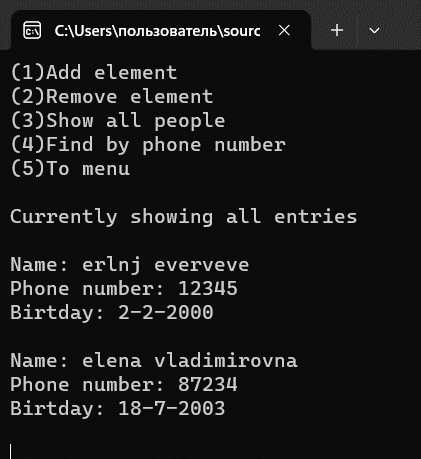


Рисунок 7 – Отключение фильтрации

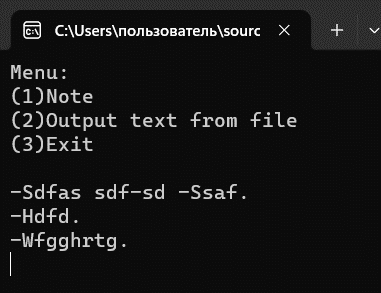
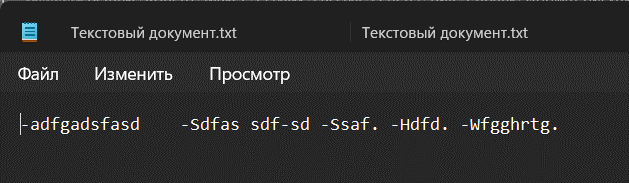


Рисунок 8 – вывод строкового задания

1. **Вывод**

В ходе выполнения лабораторной работы был создан Note, описывающий структуру для хранения информации о человеке;

Были перегружены операторы извлечения и вставки.

Было реализовано пользовательское меню, код программы разделен на заголовочные файлы и файлы, содержащие код.

В итоге тестирования программы выяснилось, что все функции работают исправно, что говорит о корректном написании кода.

По итогу выполнению лабораторной работы были закреплены полученные знания о стандартных, строковых и файловых потоках в языке С++.