

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: Приложение для управления процессами операционной системы

Студент гр. 0336

_____ Глазырин М.А.

Преподаватель

_____ Калмычков В.А.

Санкт-Петербург

2021

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Глазырин М.А.

Группа 0336.

Тема работы: Приложение для управления процессами операционной системы.

Исходные данные: 7.6. Подготовить справочник по продаже недвижимости.

Имеется информация о характеристиках продаваемой недвижимости (например, район, площадь квартиры, количество комнат, этажность, цена, адрес и т.п.) и заявках на покупку недвижимости с аналогичными характеристиками, при этом в заявках могут присутствовать списки желаемых вариантов.

Необходимо обеспечить эффективную выдачу сведений:

- о подходящих по площадям квартирах (в определенном районе или по выбору районов);
- о подходящих по цене квартирах (в определенном районе или по выбору районов) с учетом дополнительных условий (например, этаж, площадь и др.);
- справку по определенному количеству комнат в квартире;
- варианты встречных покупок/продаж.

Содержание пояснительной записки: Введение, теоретическая часть, реализация программы, использованное ПО, результаты работы программы, описание функций, заключение, список использованных источников, приложение 1 – блок-схема, приложение 2 – руководство пользователя, приложение 3 – исходный код.

Предполагаемый объем пояснительной записки:

Не менее 30 страниц.

Студент гр. 0336

Глазырин М.А.

Преподаватель

Калмычков В.А.

АННОТАЦИЯ

Данная курсовая работа содержит реализацию консольного справочника по продаже недвижимости на языке C++. Программа корректно обрабатывает ошибки, возникающие в процессе выполнения, и выводит сообщение о некорректных действиях пользователя на экран.

SUMMARY

This course work contains the implementation of the console directory for the sale of real estate in C++. The program correctly handles errors that occur during execution, and displays a message about incorrect user actions on the screen.

СОДЕРЖАНИЕ

	Введение	6
1.	Теоретическая часть	7
2.	Реализация программы	10
2.1.	Использованное ПО	10
2.2.	Описание структур, классов, функций, методов	10
	Заключение	14
	Список использованных источников	15
	Приложение 1. Блок-схема	16
	Приложение 2. Руководство пользователя	18
	Приложение 3. Исходный код	22

ВВЕДЕНИЕ

В курсовой работе была поставлена задача разработать интерактивный справочник по продаже недвижимости.

В рамках поставленной задачи была создана программа на языке C++, которая производит чтение заранее созданной базы недвижимости и позволяет пользователю получать справочную информацию о комнатах в квартире и видеть пересечения различных заявок на покупку недвижимости. Также реализован интерфейс для создания заявки на покупку, который позволяет найти все квартиры по указанным фильтрам.

Подробнее об этом и не только можно прочесть далее в изложении теоретического материала.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Справочник

Справочная система подразумевает наличие структурированной информации, доступ к которой и предоставляет система, далее – справочник. При всем при этом справочник не подразумевает возможности его пополнения со стороны пользователя.

Частным случаем справочника является токовый словарь. В нем вся информация структурирована по алфавиту, что позволяет ускорить процесс поиска слов. В электронных версиях словарей помимо стандартных возможностей нередко появляется автоматический поиск, фильтр по различным авторам и типам словарей (толковый, этимологический, орфографический).

Таким образом, программа должна иметь ограниченный набор функций, предоставляющий справочную информацию в удобном виде, но не должна позволять пользователю вносить изменения в базу, чтобы не испортить справочную информацию.

1.2. L1-список

L1-список — линейный односвязный список отличается от L2-списка тем, что двигаться по нему можно только в одном направлении (календарь не разрешается листать назад), а вставлять и удалять элементы — только за указателем.

В рамках данной курсовой работы реализуем стандартный L1-список. Для этого нам потребуется структура данных `Node1` для хранения данных одного узла и класс `L1` для реализации самого списка. Их структура представлена на рисунках 1 и 2.

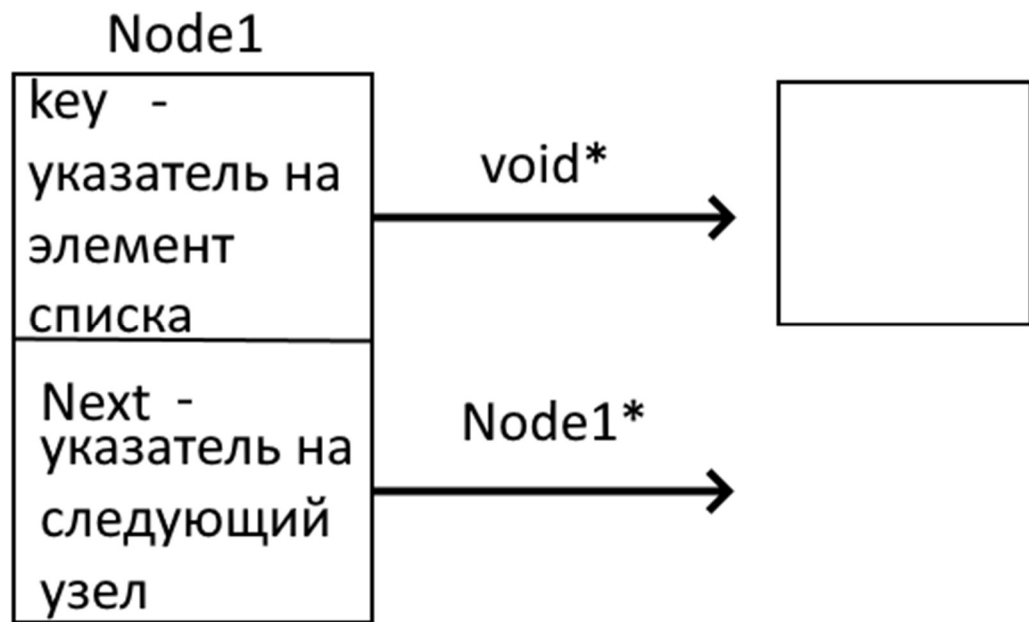


Рисунок 1 – схема устройства узла списка

Т.к. нам потребуется использовать несколько списков будем хранить указатель не на определённый класс или тип, а хранить просто `void*` для того, чтобы можно было использовать список с разными данными.

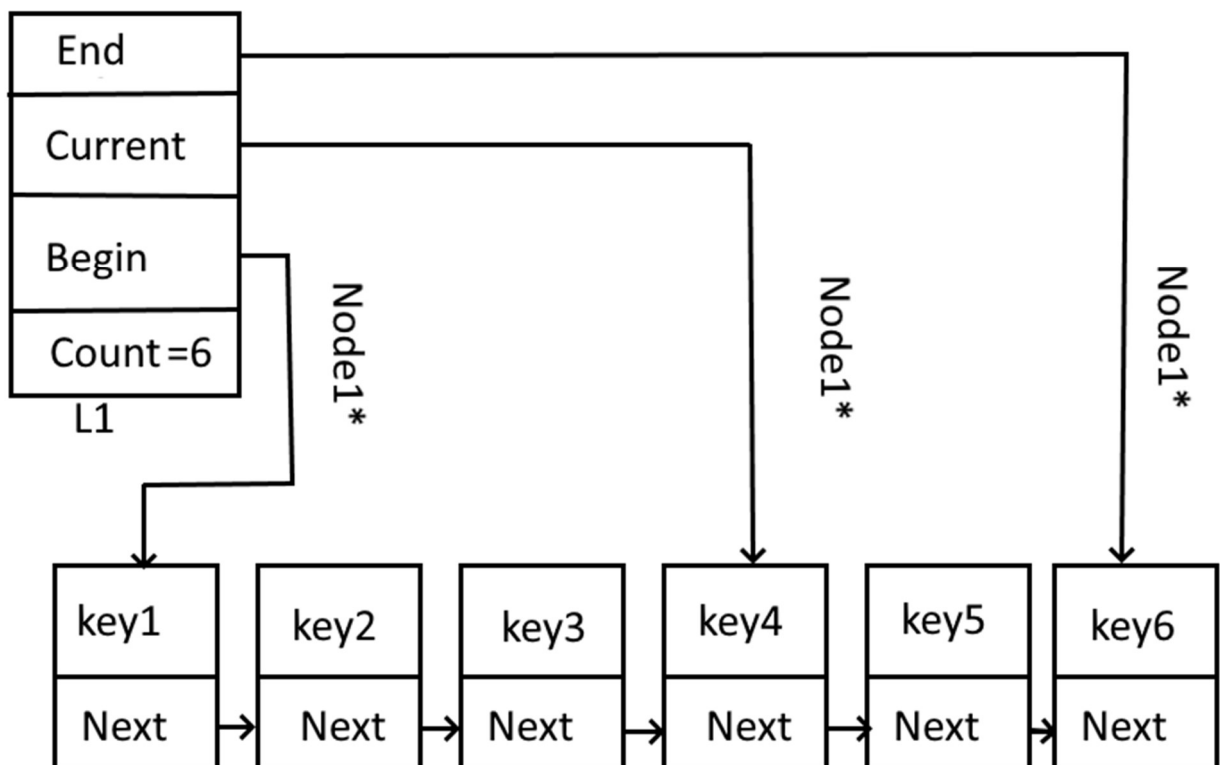


Рисунок 2 – Схема устройства списка, в котором хранится 6 элементов

1.3. Хранение данных в файле

Существует немало способов хранения информации в текстовых файлах. Приведем некоторые из них:

- Json - текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом. Несмотря на происхождение от JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON. Для с++ также существует библиотека для работы с этим форматом, но т.к. по заданию нельзя использовать сторонние библиотеки, откажемся от этого формата.
- Txt с разделителем – табуляция. Т.к. в разных текстовых редакторах табуляция представляется разными способами не будем использовать этот формат, чтобы избежать ошибок.
- CSV (от англ. Comma-Separated Values — значения, разделённые запятыми) — текстовый формат, предназначенный для представления табличных данных. Строка таблицы соответствует строке текста, которая содержит одно или несколько полей, разделенных запятыми. При использовании такого формата можно забыть хранения полей, содержащих запятые. В нашей базе запятая встречается в адресе квартиры, что может привести к некорректному чтению строки. Именно поэтому можно заменить , на ; для решения этой проблемы.

На рисунке 3 пример хранения информации о недвижимости в файле:

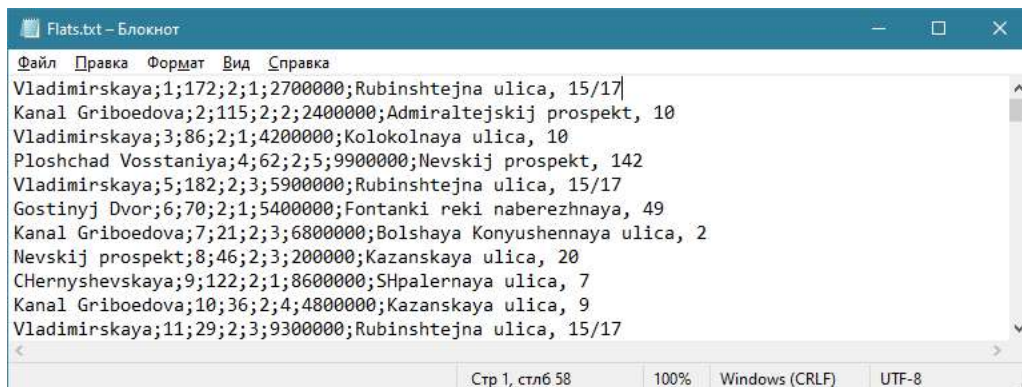


Рисунок 3 – текстовое представление данных в файле

2. РЕАЛИЗАЦИЯ ПРОГРАММЫ

2.1. Используемое ПО

Операционная система: Microsoft Windows 10 Pro.

Среда разработки: Microsoft Visual Studio 2019.

Компилятор: Microsoft Visual C++.

2.2. Описание структур, классов, функций, методов

Имя модуля	Имя структуры/класса/метода	Назначение	Параметры функции	Возвращаемое значение
CurrentBase.h	Класс CurrentBase	Хранит загруженные из файла данные и позволяет корректно их обрабатывать		
	Конструктор CurrentBase	Загружает из файлов данные в базу	Не принимает	
	ShowTrueSquare Flats	Отобразит список недвижимости по заданной площади	Не принимает	Не возвращает
	ShowTrueCostFlats	Отобразит список недвижимости по заданной цене	Не принимает	Не возвращает
	ShowAll	Отобразит весь список недвижимости	Не принимает	Не возвращает
	GetRoomsCount	Вернёт кол-во комнат по идентификатору квартиры	ID квартиры	Число комнат
	GetFlat	Ищет в базе квартиру по ID	ID квартиры	Указатель на квартиру
	WorkWithRequests	Интерфейс работы с заявками	Не принимает	Не возвращает
	GetMinMax (имеет спецификатор доступа private)	Функция, корректно получающая границы фильтра. Создана для расширения возможностей пользователя. Вынесена в отдельную функцию, т.к. этот функционал требуется дважды	Не принимает	Сохраняет результат в поля класса
	LoadPurchaseRequest (имеет спецификатор доступа private)	Загружает из указанного файла список заявок и обрабатывает его	Не принимает	Сохраняет результат в список

	LoadFlats (имеет спецификатор доступа private)	Загружает из указанного файла списка недвижимости	Не принимает	Сохраняет результат в список
flat.h	Структура flat	Структура для хранения информации о единице недвижимости		
	Конструктор flat	Конструктор на тот случай, если появится потребность в сохранении с последующей загрузкой в/из бинарный(ого) файл(а).	Не принимает	
	Конструктор flat	«Парсит» полученную строку в советующие поля Строка должна иметь следующий вид, в противном случае некоторые поля останутся наполненными: Area;ID;Square;RoomCount;Flours;Cost;Address	Строка, содержащая входную строку	
	ToStr	Функция преобразования всего объекта в строку формата: Area;ID;Square;RoomCount;Flours;Cost;Address	Не принимает	Указатель на начало строки, содержащей результат
	print	Выводит в указанный поток строку формата: Area;ID;Square;RoomCount;Flours;Cost;Address	Принимает ссылку на поток вывода	Не возвращает
	printConsole	Выводит в консоль данные информацию о недвижимости	Не принимает	Не возвращает
	Деструктор ~flat	Корректно освобождает захваченные ресурсы		
L1.h	Структура Node1	Структура, необходимая для хранения информации в списке		
	Конструктор Node1	Конструктор узла списка	Значение в узле	
	Деструктор ~Node1	Очищает захваченные ресурсы		

Класс L1	Список типа L1. Работает с не типизированными указателями, чтобы предоставить возможность хранить любые типы данных в этом списке. При извлечении элемента из списка потребуется явное приведение, поэтому нужно помнить, какие типы помещаются в каждый созданный список		
Size()	Размер списка	Не принимает	Текущее кол-во элементов в списке
Конструктор L1	Конструктор списка	Не принимает	
Конструктор L1	Создает список с 1 элементом внутри	Указатель на элемент для помещения в список	
GetCurrent	Функция доступа к элементу списка	Не принимает	Вернёт значение, на которое указывает текущий указатель в списке
TakeStep	Перенести указатель на один элемент вперёд. При попытке выйти за границу списка вернёт указатель в начало и False-значение	Не принимает	True при удачном шаге, иначе False
GoToBegin	Возвращает текущий указатель в начало списка	Не принимает	Не возвращает
Clear	Очистка списка	Не принимает	Не возвращает

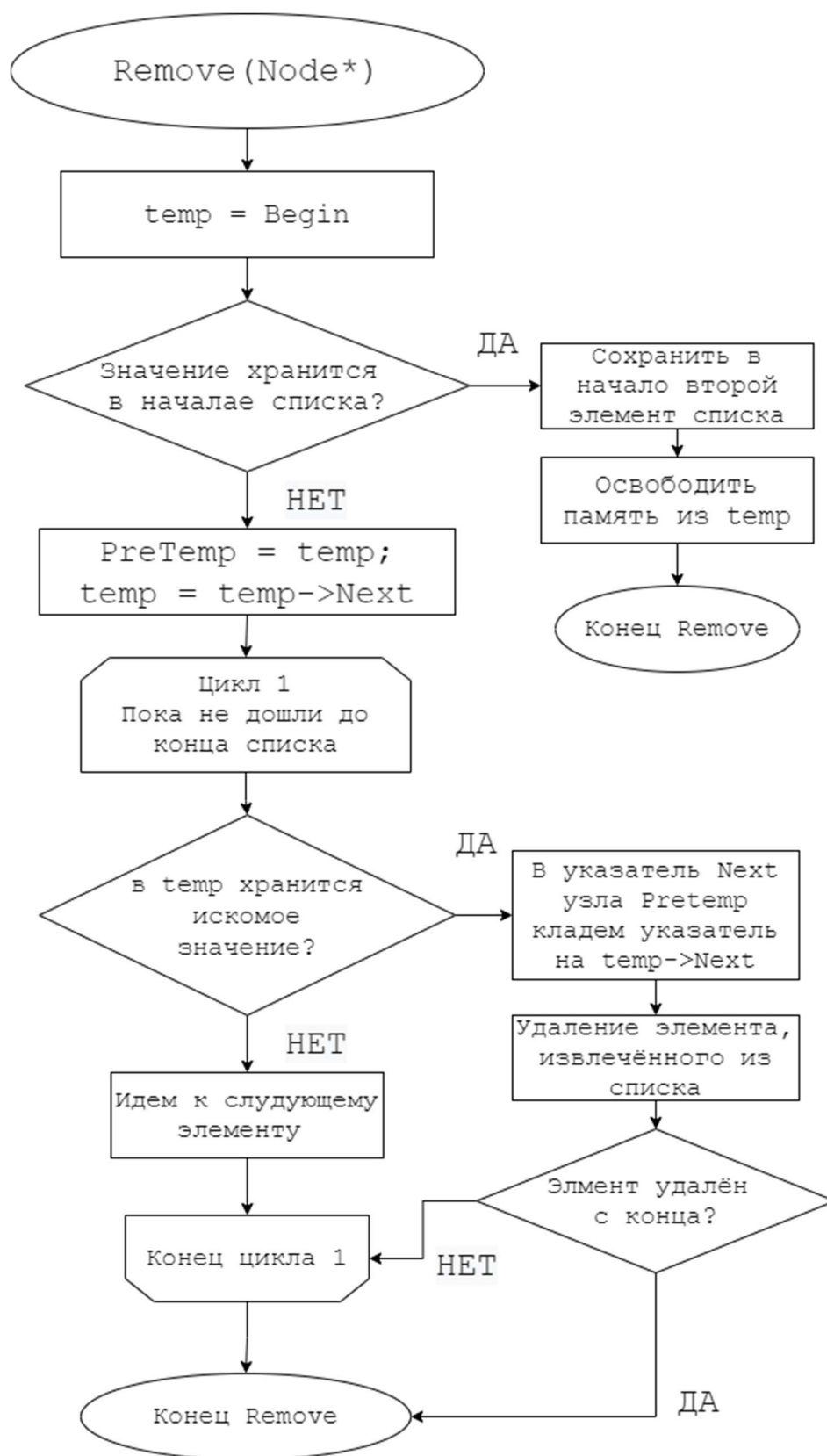
	PushBack	Добавление элемента в конец списка	Указатель на добавляемый элемент	Не возвращает
	PushFront	Добавление элемента в начало	Указатель на добавляемый элемент	Не возвращает
	Remove	Удаление из списка элемента по указателю	Указатель на удаляемый элемент	
	Деструктор ~L1	Очищает список		
Request.h	Структура Request	Хранит информацию о заявках		
	Request	Считывает информацию о заявке из консоли		
	Request	Считывает информацию о заявке из потока ввода	Поток ввода	
	CheckFlat	Проверяет соответствие квартиры критериям запроса	Указатель на квартиру	True, если соответствует

ЗАКЛЮЧЕНИЕ

В процессе написания данной курсовой работы я изучил устройство списков и некоторых других динамических структур данных, изучил различные способы хранения данных в файлах, а также улучшил свои навыки программирования на языке c++.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

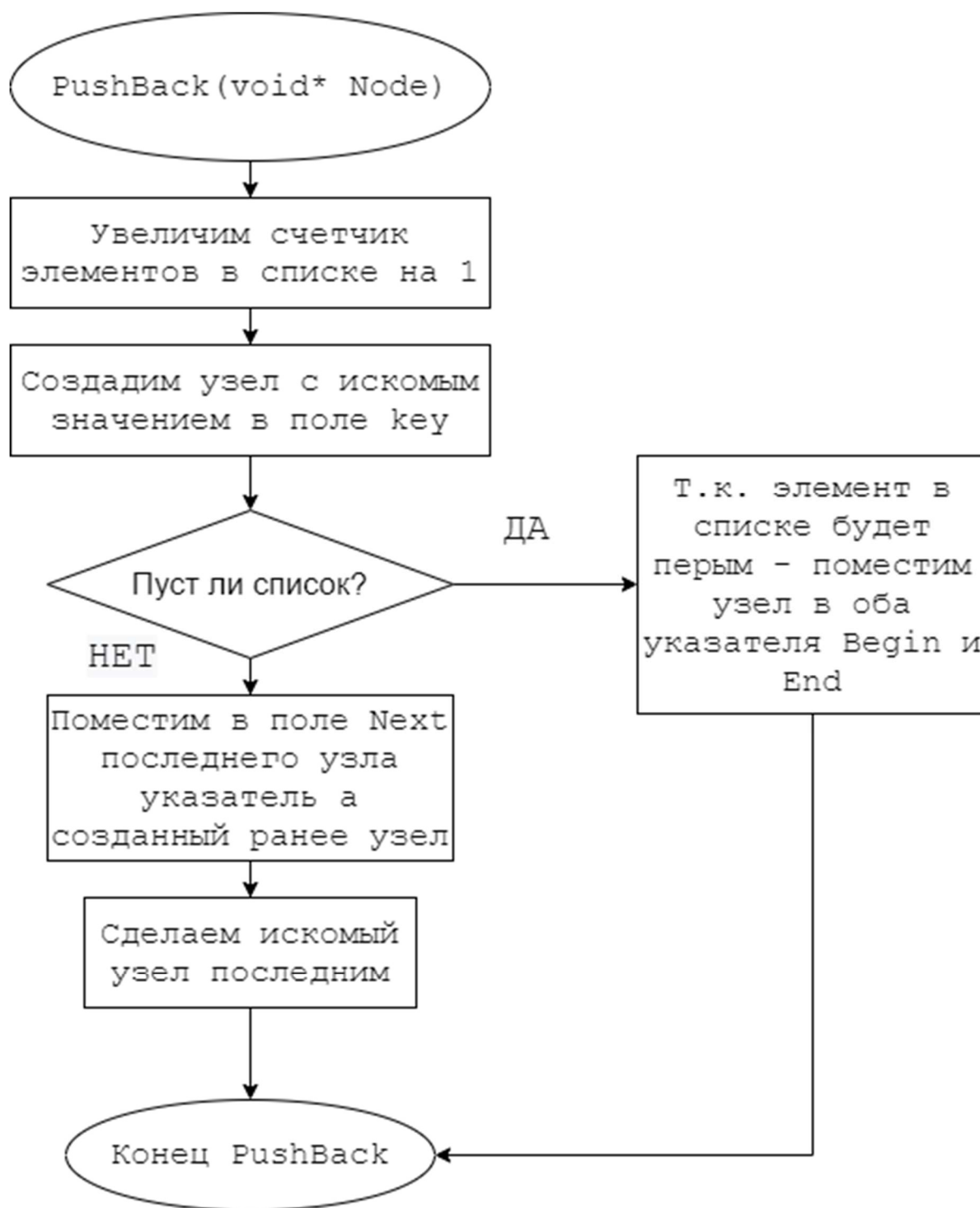
1. Брайан Керниган, Деннис Ритчи. Язык программирования Си. – Москва: Финансы и статистика, 1992. — 272 с.
2. База Данных Квартир <http://www.saint-petersburg-apartments.com/ru/all/index.html> (дата посещения 15.04.2021)
3. Лекция 30: Динамические структуры данных: однонаправленные и двунаправленные списки <https://intuit.ru/studies/courses/648/504/lecture/11456> (дата посещения 19.04.2021)
4. Лекция 10: Основы объектно-ориентированного программирования <https://intuit.ru/studies/courses/105/105/lecture/3077?page=6> (дата посещения 19.04.2021)



Приложение 1. Блок-схема Remove

Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Максимов Ю.Е.		
Провер.		Калмычков В.А.		
Реценз.		Калмычков В.А.		
Н. Контр.		Калмычков В.А.		
Утв.		Калмычков В.А.		

Лит.	Лист	Листов
	1	1



					<i>Приложение 1. Блок-схема PushBack</i>		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Максимов Ю.Е.			17	Лит.	Лист
Провер.		Калмычков В.А.					Листов
Реценз.		Калмычков В.А.					1
Н. Контр.		Калмычков В.А.					1
Утв.		Калмычков В.А.					

ПРИЛОЖЕНИЕ 2. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. Минимальные системные требования

Операционная система: Microsoft Windows 10 Pro. 64 – разрядная операционная система.

Процессор: Как минимум 1 ГГц или SoC.

ОЗУ: 2 ГБ.

Место на жестком диске: 20 ГБ.

Дисплей: 800 x 600.

2. Процесс установки

Скопируйте файл Course_work.exe в выбранную директорию.

3. Процесс запуска программы и работы с ней

Запустите Course_work.exe из выбранной директории. Появится окно, которое показано на рисунке 4:

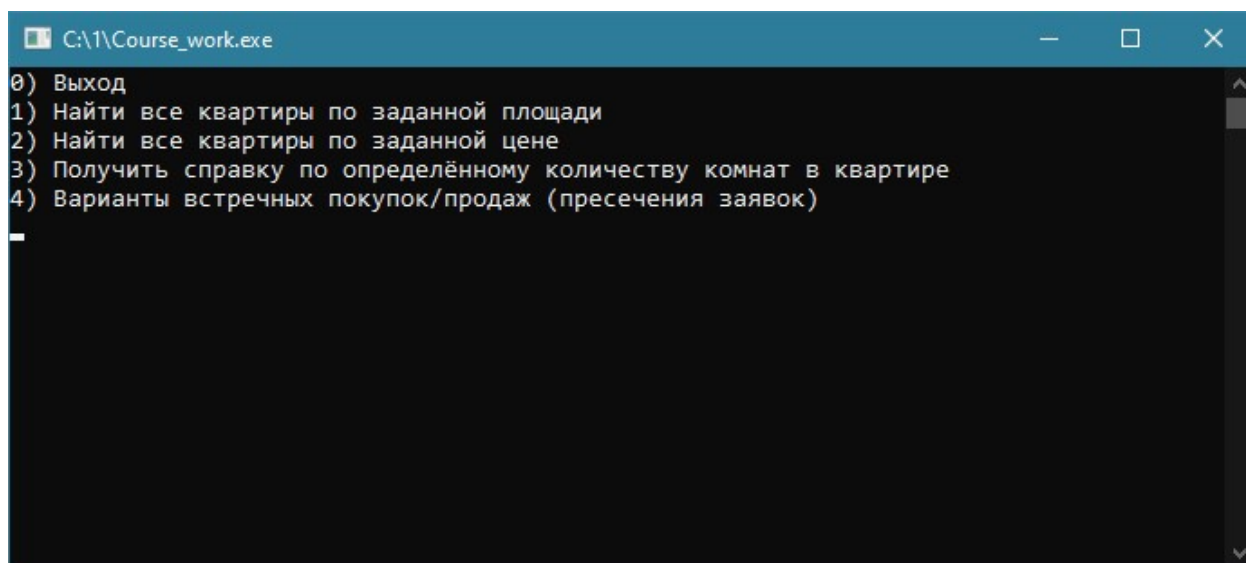


Рисунок 4 – Начальное окно программы

Далее программа попросит Вас ввести номер действия из меню. Выберите и введите его, нажмите клавишу Enter. Если выбрать один из первых двух пунктов, программа попросит ввести промежуток, пример на рисунке 5.

```
C:\1\Course_work.exe
0) Выход
1) Найти все квартиры по заданной площади
2) Найти все квартиры по заданной цене
3) Получить справку по определённому количеству комнат в квартире
4) Варианты встречных покупок/продаж (пресечения заявок)
1
Укажите нужный промежуток по следующим правилам:
>x
<x
x-y
Где x и y - числа
_
```

Рисунок 5 – Консоль после выбора первого или второго действия из меню
Введём промежуток и посмотрим результат, рисунок 6.

```
C:\1\Course_work.exe
0) Выход
1) Найти все квартиры по заданной площади
2) Найти все квартиры по заданной цене
3) Получить справку по определённому количеству комнат в квартире
4) Варианты встречных покупок/продаж (пресечения заявок)
1
Укажите нужный промежуток по следующим правилам:
>x
<x
x-y
Где x и y - числа
10-30
Квартира с ID = 1
Район: Central
Площадь картыры: 25
Количество комнат: 1
Этажность: 5
Стоимость: 3000000
Адрес: St. Petersburg, Marat Street, 8, 56
```

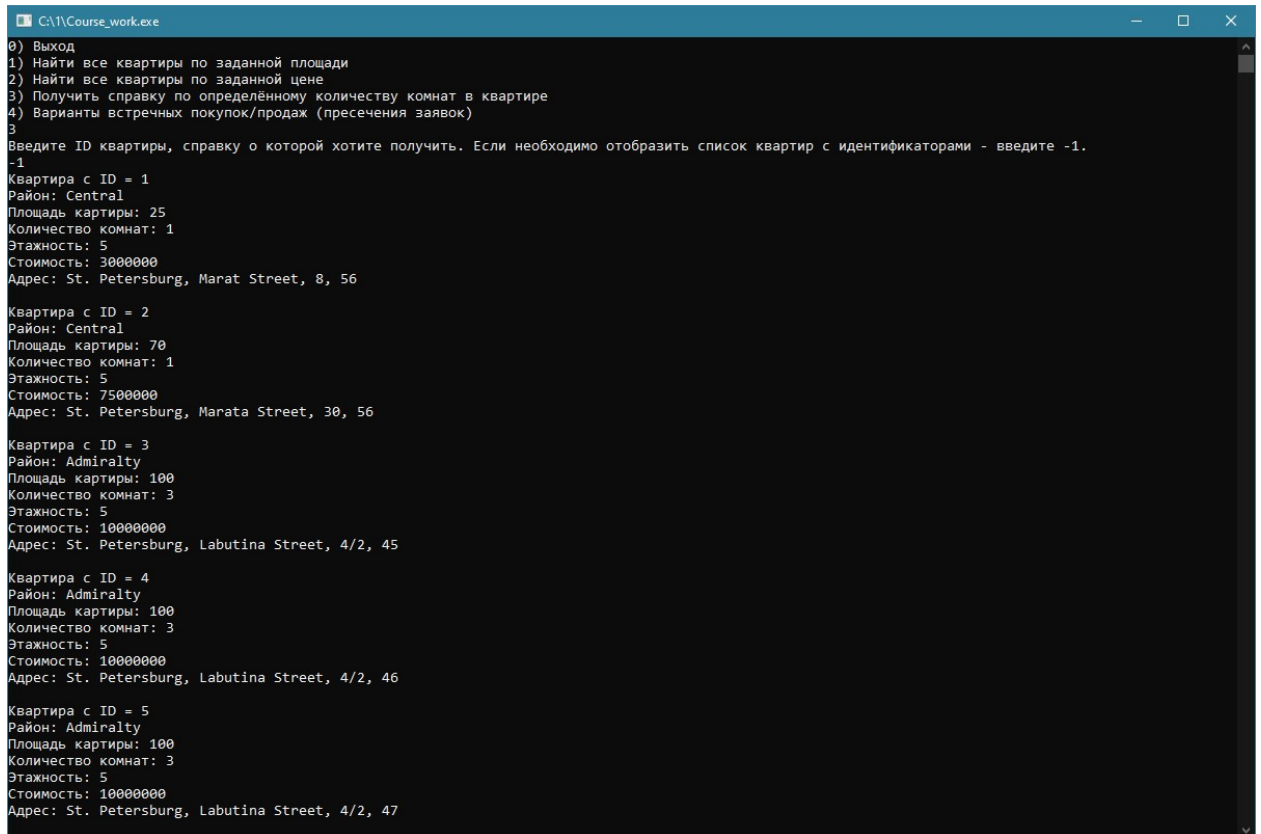
Рисунок 6 – Промежуток 10-30

При выборе 3 пункта появится меню, как на рисунке 7.

```
C:\1\Course_work.exe
0) Выход
1) Найти все квартиры по заданной площади
2) Найти все квартиры по заданной цене
3) Получить справку по определённому количеству комнат в квартире
4) Варианты встречных покупок/продаж (пресечения заявок)
3
Введите ID квартиры, справку о которой хотите получить. Если необходимо отобразить список квартир с идентификаторами - введите -1.
_
```

Рисунок 7 – Пункт 3

При вводе числа -1 программа отобразит список всех квартир в базе, пример на рисунке 8.



```
C:\1\Course_work.exe
0) Выход
1) Найти все квартиры по заданной площади
2) Найти все квартиры по заданной цене
3) Получить справку по определённому количеству комнат в квартире
4) Варианты встречных покупок/продаж (пресечения заявок)
3
Введите ID квартиры, справку о которой хотите получить. Если необходимо отобразить список квартир с идентификаторами - введите -1.
-1
Квартира с ID = 1
Район: Central
Площадь карты: 25
Количество комнат: 1
Этажность: 5
Стоимость: 3000000
Адрес: St. Petersburg, Marat Street, 8, 56

Квартира с ID = 2
Район: Central
Площадь карты: 70
Количество комнат: 1
Этажность: 5
Стоимость: 7500000
Адрес: St. Petersburg, Marata Street, 30, 56

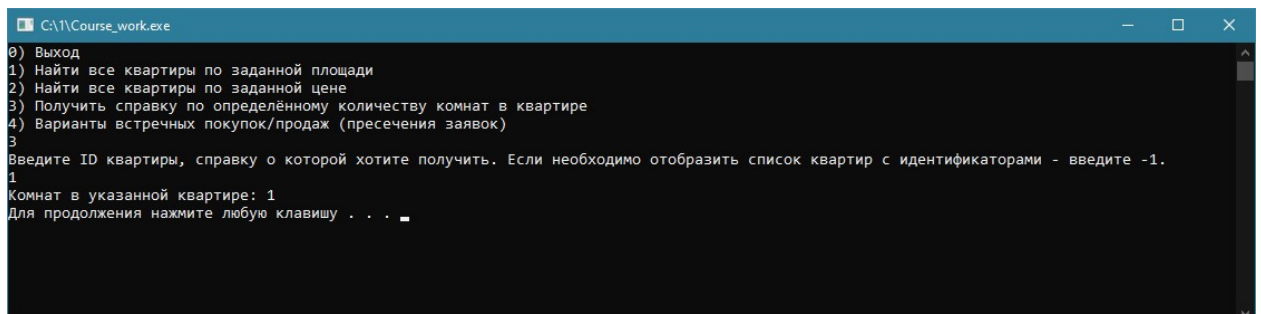
Квартира с ID = 3
Район: Admiralty
Площадь карты: 100
Количество комнат: 3
Этажность: 5
Стоимость: 10000000
Адрес: St. Petersburg, Labutina Street, 4/2, 45

Квартира с ID = 4
Район: Admiralty
Площадь карты: 100
Количество комнат: 3
Этажность: 5
Стоимость: 10000000
Адрес: St. Petersburg, Labutina Street, 4/2, 46

Квартира с ID = 5
Район: Admiralty
Площадь карты: 100
Количество комнат: 3
Этажность: 5
Стоимость: 10000000
Адрес: St. Petersburg, Labutina Street, 4/2, 47
```

Рисунок 8 – Список всех квартир

При вводе ID квартиры программа выведет число комнат в квартире, пример на рисунке 9.



```
C:\1\Course_work.exe
0) Выход
1) Найти все квартиры по заданной площади
2) Найти все квартиры по заданной цене
3) Получить справку по определённому количеству комнат в квартире
4) Варианты встречных покупок/продаж (пресечения заявок)
3
Введите ID квартиры, справку о которой хотите получить. Если необходимо отобразить список квартир с идентификаторами - введите -1.
1
Комнат в указанной квартире: 1
Для продолжения нажмите любую клавишу . . .
```

Рисунок 9 – Число комнат в квартире

При выборе 4 пункта можно создать свою заявку, и программа выдаст результат поиска. Пример на рисунке 10.

```
C:\Users\Хозяин\source\repos\Course_work\Debug\Course_work.exe
0) Выход
1) Найти все квартиры по заданной площади
2) Найти все квартиры по заданной цене
3) Получить справку по определённому количеству комнат в квартире
4) Варианты встречных покупок/продаж (пресечения заявок)
4
Введите команду:
0) Выход
1) Просмотреть список всех заявок на квартиру по ID
2) Составить свою заявку и вывести все совпадающие заявки
1
Введите ID квартиры:250
Список заявок на квартиру: Квартира с ID = 250
Район: Kanal Griboedova
Площадь квартиры: 63
Количество комнат: 5
Этажность: 7
Стоимость: 1400000
Адрес: Millionnaya ulica, 29

ID заявок:
1
```

Рисунок 10 – Результат поиска заявок на квартиру

Второй пункт меню заявок позволяет составить свою заявку и получить результат поиска. Пример на рисунке 11.

```
C:\Users\Хозяин\source\repos\Course_work\Debug\Course_work.exe
0) Выход
1) Найти все квартиры по заданной площади
2) Найти все квартиры по заданной цене
3) Получить справку по определённому количеству комнат в квартире
4) Варианты встречных покупок/продаж (пресечения заявок)
4
Введите команду:
0) Выход
1) Просмотреть список всех заявок на квартиру по ID
2) Составить свою заявку и вывести все совпадающие заявки
2
Введите заявку в виде: Район;Минимальная площадь-Максимальная площадь;Минимальное кол-во комнат-Максимальное;Минимальный этаж-Максимальный;Минимальная цена-Максимальная
Введите - , чтобы не проводить фильтрацию по полю
Mayakovskaya;1-100;1-5;1-5;1000000-4000000
Квартира с ID = 43
Район: Mayakovskaya
Площадь квартиры: 29
Количество комнат: 2
Этажность: 3
Стоимость: 3500000
Адрес: Kolomenskaya ul, 20

Квартира с ID = 115
Район: Mayakovskaya
Площадь квартиры: 69
Количество комнат: 1
Этажность: 4
Стоимость: 3400000
Адрес: Mayakovskogo ulica, 1/96

Квартира с ID = 152
Район: Mayakovskaya
Площадь квартиры: 39
Количество комнат: 3
Этажность: 5
Стоимость: 1000000
Адрес: Shekhova ulica, 4

Введите команду:
0) Выход
1) Просмотреть список всех заявок на квартиру по ID
2) Составить свою заявку и вывести все совпадающие заявки
```

Рисунок 11 – Пример результата обработки заявки

ПРИЛОЖЕНИЕ 3. ИСХОДНЫЙ КОД

Файл main.cpp:

```
#include "L1.h"
#include "flat.h"
#include <iostream>
#include "CurrentBase.h"

int main()
{
    system("chcp 1251&&cls");
    CurrentBase base;
    int command;
    while (true)
    {
        cout << "\n\
0) Выход\n\
1) Найти все квартиры по заданной площади\n\
2) Найти все квартиры по заданной цене\n\
3) Получить справку по определённому количеству комнат в\n\
квартире\n\
4) Варианты встречных покупок/продаж (пресечения заявок)\n\
";

        cin >> command;
        if (cin.fail()) {
            cin.clear();
            cout << "Ошибка при вводе команды\n";
            system("pause&&cls");
            continue;
        }
        cin.ignore(32767, '\n');
        switch (command)
        {
            case 0: return 0; break;
            case 1: base.ShowTrueSquareFlats(); break;
            case 2: base.ShowTrueCostFlats(); break;
            case 3:
                cout << "Введите ID квартиры, справку о\n\
которой хотите получить.\n\
Если необходимо отобразить список квартир с\n\
идентификаторами - введите -1.\n\
\n";
                int ID;
                cin >> ID;
                if (ID < 0)
                {
                    base.ShowAll();
                    cin >> ID;
                }
                cout << "Комнат в указанной квартире: " <<
base.GetRoomsCount(ID) << endl;
                break;
            case 4: base.WorkWithRequests(); break;
```

```

        default: cout << "Ошибка при вводе команды\n";
                break;
        }
        system("pause&&cls");
    }
    return 0;
}

```

Файл flat.h:

```

#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include "L1.h"
using namespace std;
struct flat
{
    flat() {}
    flat(char* input);
    char* ToStr();
    void print(fstream& f);
    void printConsole();
    char* Area = nullptr;
    int ID = 0;
    int Square = 0;
    int RoomCount = 0;
    int Flour = 0;
    int Cost = 0;
    char* Address = nullptr;
    L1 Requests;
    ~flat()
    {
        if (Area != nullptr) delete[] Area;
        if (Address != nullptr) delete[] Address;
    }
};

```

Файл flat.cpp:

```

#include "flat.h"
flat::flat(char* input)
{
    if (input == nullptr) { cout << "Ошибка выделения
памяти"; return; }
    Area = new char[strlen(input) - (strchr(input, ';') - input + 2)];
    //выделим память по размеру района
    Address = new char[strlen(input) - (strrchr(input, ';')
- input) + 2]; //выделим память по размеру адреса
    sscanf(input, "%[^;];%d;%d;%d;%d;%d;%[^\\n]", Area, &ID,
&Square, &RoomCount, &Flour, &Cost, Address);
}

char* flat::ToStr()
{

```

```

        char* output = new char[4 * 10 + strlen(Area) +
strlen(Address)];
        sprintf(output, "%s;%d;%d;%d;%d;%s", Area, Square,
RoomCount, Flour, Cost, Address);
        return output;
    }

    void flat::print(fstream& f) {
        f << Area << ';' << Square << ';' << RoomCount << ';'
<< Flour << ';' << Cost << ';' << Address << endl;
    }

    void flat::printConsole()
    {
        cout << "Квартира с ID = " << ID << endl <<
            "Район: " << Area << endl <<
            "Площадь квартиры: " << Square << endl <<
            "Количество комнат: " << RoomCount << endl <<
            "Этажность: " << Flour << endl <<
            "Стоимость: " << Cost << endl <<
            "Адрес: " << Address << endl << endl;
    }

```

Файл L1.h:

```

#pragma once
struct Node1
{
    Node1(void* key) { this->key = key; }
    void* key = nullptr;
    Node1* Next = nullptr;
    ~Node1() { if (key != nullptr) delete key; }
};

class L1
{
public:
    int Size();
    L1() {}
    L1(void* Node);
    void* GetCurrent();
    bool TakeStep();
    void GoToBegin();
    void Clear();
    void PushBack(void* Node);
    void PushFront(void* Node);
    void Remove(void* Node);
    ~L1() { Clear(); }

private:
    Node1* Begin = nullptr;
    Node1* End = nullptr;
    Node1* Current = nullptr;
    int count = 0;
};

```


Файл L1.cpp:

```
#include "L1.h"
L1::L1(void* Node) { PushBack(Node); }
void L1::GoToBegin() { Current = Begin; }
void* L1::GetCurrent() { return Current->key; }
int L1::Size() { return count; }
void L1::Clear()
{
    Nodel* temp;
    while (Begin != nullptr)
    {
        temp = Begin;
        Begin = Begin->Next;
        delete temp;
    }
    Current = nullptr;
    Begin = nullptr;
    End = nullptr;
    count = 0;
}
bool L1::TakeStep()
{
    if (Begin == nullptr) return false;
    if (Current == nullptr) Current = Begin;
    if (Current->Next == nullptr) {
        Current = Begin;
        return false;
    }
    Current = Current->Next;
    return true;
}
void L1::PushBack(void* Node)
{
    count++;
    Nodel* temp = new Nodel(Node);
    if (Begin == nullptr) End = Begin = temp;
    else
    {
        End->Next = temp;
        End = End->Next;
    }
}
void L1::PushFront(void* Node)
{
    count++;
    Nodel* temp = new Nodel(Node);
    temp->Next = Begin;
    Begin = temp;
}
void L1::Remove(void* Node)
{
    auto temp = Begin;
```

```

        if (Begin->key == Node)
        {
            Begin = Begin->Next;
            delete temp;
            return;
        }
        auto PreTemp = temp;
        temp = temp->Next;
        while (temp!=nullptr)
        {
            if (temp->key == Node)
            {
                PreTemp->Next = temp->Next;
                delete temp;
                temp = PreTemp->Next;
                if (temp == nullptr) return;
            }
            PreTemp = temp;
            temp = temp->Next;
        }
    }
}

```

Файл Request.h:

```

#pragma once
#include "L1.h"
#include "flat.h"
#include <climits>
#include <fstream>
using namespace std;
struct Request
{
    Request();
    Request(ifstream& in);
    L1 FlatsID;
    int ID = 0;
    char* Area = nullptr;
    int MinSquare = 0, MaxSquare = INT_MAX;
    int MinRoomCount = 0, MaxRoomCount = INT_MAX;
    int MinFlour = 0, MaxFlour = INT_MAX;
    int MinCost = 0, MaxCost = INT_MAX;
    bool CheckFlat(void* Flat);
};

```

Файл Request.cpp:

```

#include "Request.h"
bool Request::CheckFlat(void* Flat)
{
    auto check = (flat*)Flat;
    if (check->Flour < MinFlour || check->Flour > MaxFlour)
return false;
    if (check->Cost < MinCost || check->Cost > MaxCost)
return false;
    if (check->Square < MinSquare || check->Square >
MaxSquare) return false;
}

```

```

        if (check->RoomCount < MinRoomCount || check->RoomCount
> MaxRoomCount) return false;
        if (Area != nullptr && strcmp(Area, "-") != 0) if
(strcmp(Area, check->Area) != 0) return false;
        return true;
    }
    Request::Request(ifstream& in)
    {
        int FlatsCount;
        char input[1024];
        in.getline(input, 1023);
        Area = new char[min(abs(strchr(input, ';') - input +
2), 1024)];
        sscanf(input, "%[^;];%d;%d-%d;%d-%d;%d-%d;%d",
Area, &ID, &MinSquare, &MaxSquare, &MinRoomCount, &MaxRoomCount,
&MinFlour, &MaxFlour, &MinCost, &MaxCost, &FlatsCount);
        for (size_t i = 0; i < FlatsCount; i++)
        {
            int* ID = new int;
            in >> *ID;
            FlatsID.PushBack(ID);
        }
        in.clear();
        in.ignore(32000, '\n');
    }
    Request::Request()
    {
        int FlatsCount;
        char input[1024];
        cout << "Введите заявку в виде: Район;Минимальная
площадь-Максимальная площадь;Минимальное кол-во комнат-
Максимальное;Минимальный этаж-Максимальный;Минимальная цена-
Максимальная" << endl
        << "Введите - , чтобы не проводить фильтрацию по
полю" << endl;
        cin.clear(); cin.ignore(32767, '\n');
        cin.getline(input, 1023);
        Area = new char[min(abs(strchr(input, ';') - input +
2), 1024)];
        sscanf(input, "%[^;];%d-%d;%d-%d;%d-%d;%d-%d", Area,
&MinSquare, &MaxSquare, &MinRoomCount, &MaxRoomCount, &MinFlour,
&MaxFlour, &MinCost, &MaxCost);
    }

```

Файл CurrentBase.h:

```

#pragma once
#include <iostream>
#include <fstream>
#include "L1.h"
#include "flat.h"
#include "Request.h"
using namespace std;
class CurrentBase

```

```

{
public:
    CurrentBase();
    void ShowTrueSquareFlats();
    void ShowTrueCostFlats();
    void ShowAll();
    int GetRoomsCount(int RoomId);
    flat* GetFlat(int Id);
    void WorkWithRequests();
private:
    int MinSquare = 0;
    int MaxSquare = INT_MAX;
    L1 Requests;
    L1 Flats;
    void GetMinMax()
    {
        MinSquare = 0, MaxSquare = INT_MAX;
        char str[51];
        cout << "\
Укажите нужный промежуток по следующим правилам:\n\
>x\n\
<x\n\
x-y\n\
Где x и y - числа\n";
        cin.getline(str, 50, '\n');
        if (str[0] == '>') { if (sscanf_s(str, ">%d",
&MinSquare) == 0) { cout << "Ошибка при вводе\n";
system("pause&&cls"); ShowTrueSquareFlats(); return; } }
        else
            if (str[0] == '<') { if (sscanf_s(str, "<%d",
&MaxSquare) == 0) { cout << "Ошибка при вводе\n";
system("pause&&cls"); ShowTrueSquareFlats(); return; } }
            else { if (sscanf_s(str, "%d-%d", &MinSquare,
&MaxSquare) != 2) { cout << "Ошибка при вводе\n";
system("pause&&cls"); ShowTrueSquareFlats(); return; } }
            if (MaxSquare < MinSquare) { cout << "Ошибка при
вводе\n"; system("pause&&cls"); ShowTrueSquareFlats(); return; }
        }
        void LoadFlats();
        void LoadPurchaseRequest();
    };
};

```

Файл CurrentBase.cpp:

```

#include "CurrentBase.h"
CurrentBase::CurrentBase()
{
    LoadFlats();
    LoadPurchaseRequest();
    Requests.GoToBegin();
    do
    {
        auto curReq = (Request*)Requests.GetCurrent();
        curReq->FlatsID.GoToBegin();
    }
}

```

```

        do
        {
            if (curReq->FlatsID.Size() == 0) break;
            GetFlat(*(int*)curReq->FlatsID.GetCurrent())-
>Requests.PushBack(curReq);
        }
        while (curReq->FlatsID.TakeStep());
        Flats.GoToBegin();
        do
            if (curReq->CheckFlat(Flats.GetCurrent()))
((flat*)Flats.GetCurrent())->Requests.PushBack(curReq);
            while (Flats.TakeStep());

        } while (Requests.TakeStep());
    }

    void CurrentBase::LoadFlats()
    {
        ///Ограничим строку для описания одной квартирой 1024
        символами потому, что действительно динамическое чтение файла
        ///создает возможность для вредоносного,
        целенаправленного переполнения стека
        char Path[_MAX_PATH];
        ifstream in("Flats.txt");
        if (in.bad())
        {
            cout << "Введите путь к файлу, содержащему базу
данных квартир\n";
            cin.getline(Path, _MAX_PATH);
            in.open(Path);
        }
        while (!in.eof())
        {
            char input[1024] = "";
            in.getline(input, 1024);
            if (input[0] == '\\0') continue;
            Flats.PushBack(new flat(input));
        }
    }

    void CurrentBase::LoadPurchaseRequest()
    {
        char Path[_MAX_PATH];
        ifstream in("Request.txt");
        if (in.bad())
        {
            cout << "Введите путь к файлу, содержащему базу
данных квартир\n";
            cin.getline(Path, _MAX_PATH);
            in.open(Path);
        }
        while (!in.eof()) Requests.PushBack(new Request(in));
    }

```

```

void CurrentBase::ShowTrueSquareFlats()
{
    GetMinMax();
    Flats.GoToBegin();
    do
    {
        auto cur = (flat*)Flats.GetCurrent();
        if (cur->Square >= MinSquare && cur->Square <=
MaxSquare) cur->printConsole();
    } while (Flats.TakeStep());
}
void CurrentBase::ShowTrueCostFlats()
{
    GetMinMax();
    Flats.GoToBegin();
    do
    {
        auto cur = (flat*)Flats.GetCurrent();
        if (cur->Cost >= MinSquare && cur->Cost <=
MaxSquare) cur->printConsole();
    } while (Flats.TakeStep());
}
flat* CurrentBase::GetFlat(int Id)
{
    Flats.GoToBegin();
    do if (((flat*)Flats.GetCurrent())->ID == Id) return
(flat*)Flats.GetCurrent(); while (Flats.TakeStep());
    return nullptr;
}

int CurrentBase::GetRoomsCount(int RoomId)
{
    Flats.GoToBegin();
    do if (((flat*)Flats.GetCurrent())->ID == RoomId)
return ((flat*)Flats.GetCurrent())->RoomCount; while
(Flats.TakeStep());
    return -1;
}

void CurrentBase::ShowAll()
{
    Flats.GoToBegin();
    do ((flat*)Flats.GetCurrent())->printConsole(); while
(Flats.TakeStep());
}

void CurrentBase::WorkWithRequests()
{
    int command, ID;
    while (true)
    {
        cout << "Введите команду:\n\

```

```

0) Выход\n\
1) Просмотреть список всех заявок на квартиру по ID\n\
2) Составить свою заявку и вывести все совпадающие заявки\n\
";

    cin >> command;
    switch (command)
    {
    case 0: return; break;
    case 1:
    {
        cout << "Введите ID квартиры:";
        cin >> ID;
        auto Finded = GetFlat(ID);
        if (Finded == nullptr) cout << "Нет такой
квартиры\n";

        else
        {
            cout << "Список заявок на квартиру: ";
            Finded->printConsole();
            cout << "ID заявок:" << endl;
            if (Finded->Requests.Size() == 0) { cout
<< "Заявок нет" << endl; continue; }
            auto Req = Finded->Requests;
            Req.GoToBegin();
            do cout << ((Request*)Req.GetCurrent())->
ID << endl; while (Req.TakeStep());
            }
            break;
        case 2:
        {
            Request* CurrRequest = new Request();
            Flats.GoToBegin();
            do
                if
(CurrRequest->CheckFlat(Flats.GetCurrent())) ((flat*)Flats.GetCurrent())->
printConsole();
                while (Flats.TakeStep());
            }
            break;
        default: cout << "Ошибка при вводе команды\n";
            break;
        }
    }
}

```