

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет физики, математики, информатики
Кафедра программного обеспечения и администрирования информационных систем

КУРСОВОЙ ПРОЕКТ

по дисциплине

Структуры и алгоритмы компьютерной обработки данных

на тему: Разработка программных модулей для обработки данных предметной области «Абоненты АТС» на основе очереди

Обучающегося 2 курса
очной формы обучения
направления подготовки
02.03.03 Математическое обеспечение и администрирование информационных систем
Направленность (профиль)
Проектирование информационных систем и баз данных
Салиму Мусонда

Руководитель:
старший преподаватель кафедры
ПОиАИС
Ураева Елена Евгеньевна

Допустить к защите:

_____/_____
« ____ » _____ 20 ____ г.

Курск, 2019

АННОТАЦИЯ

В данной пояснительной записке содержится информация о разработке программы, реализующей операцию на абонентах АТС. приложение включает в себя основные функции, которые позволяют работать с очередью, такие как: создание очереди, добавление и удаление узлов, простой графический вывод.

Ключевые слова: база данных очереди, узел, корень, ключ, приложение.

Число страниц пояснительной записки: 43.

Число рисунков: 23.

Число приложений: 2.

СОДЕРЖАНИЕ

Введение.....	4
1 Краткие теоретические сведения об объекте исследования	6
1.1. Очередь.....	6
1.2. Абоненты АТС.....	6
2 Проектирование и разработка приложения	8
2.1. Функции приложения	8
2.2. Интерфейс пользователя.....	8
2.3. Логическая структура приложения	10
2.4. Распределение исходного кода по файлам	20
2.5. Оценка сложности базового алгоритма	21
3 Тестирование.....	25
Заключение	31
Список использованных источников	32
Приложение А Текст программы	33
Приложение Б Внешний вид графического материала.....	48

ВВЕДЕНИЕ

Основной причиной, по которой я выбрал тему курсового проекта «Абоненты АТС» является то, что система управления базами данных хранит данные таким образом, что становится легче получать, манипулировать и производить информацию.

Базы данных всегда были важнейшей темой при изучении информационных систем. Однако в последние годы всплеск популярности Интернета и бурное развитие новых технологий для Интернета сделали знание технологии баз данных для многих одним из актуальнейших путей карьеры. Технологии баз данных увели Интернет-приложения далеко от простых брошюрных публикаций, которые характеризовали ранние приложения. В то же время Интернет-технология обеспечивает пользователям стандартизированные и доступные средства публикации содержимого баз данных. Правда, ни одна из этих новых разработок не отменяет необходимости в классических приложениях баз данных, которые появились еще до развития Интернета для нужд бизнеса. Это только расширяет важность знания баз данных. В соответствии с целью данного проекта, можно выделить следующие задачи:

- изучить структуру данных очереди;
- разработать способы работы с очередями;
- создать приложение обеспечивающее:
 - создание и просмотр очереди;
 - добавлять, удалять и изменять узлы очереди;
 - отображение дополнительной информации об очереди;
 - сохранение созданной очереди в файл и его последующая загрузка.

Программа будет реализована в объектно-ориентированном стиле и может быть использована пользователем для работы с абонентами АТС.

Возможной перспективой совершенствования разрабатываемого программного продукта является добавление методов работы с очередями и разработка графического вывода.

1 Краткие теоретические сведения об объекте исследования

1.1. Очередь

Очередь - это линейная структура, которая следует определенному порядку, в котором выполняются операции. Порядок «первым пришел - первым вышел» (FIFO). Хорошим примером очереди является любая очередь потребителей для ресурса, где первым обслужен потребитель, который пришел первым. Пример бинарного дерева представлен на рисунке 1.

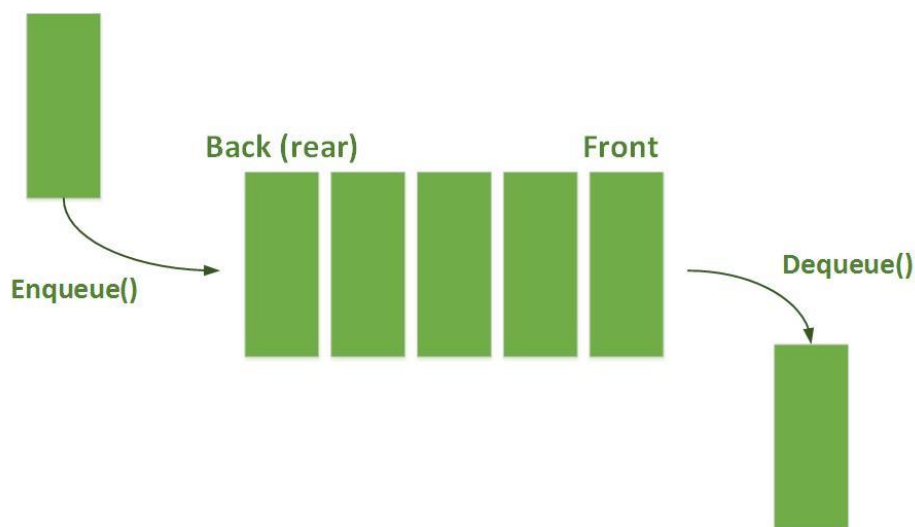


Рисунок 1 – Пример очереди.

1.2. Абоненты АТС.

Имитация абонента АТС или организация очереди вызовов позволяет АТС ставить в очередь входящие вызовы. Очередь - это линия вызовов, на которые нужно ответить. Когда вызов направляется в очередь, по умолчанию на вызовы отвечают в порядке поступления:

- Операции с очередями могут включать в себя инициализацию или определение очереди, ее использование, а затем полное удаление из памяти. Здесь мы попытаемся понять основные операции, связанные с очередями.

- enqueue () - добавить (сохранить) элемент в очередь.
- dequeue () - удалить (получить доступ) элемент из очереди.
- Еще несколько функций требуется, чтобы сделать вышеупомянутую работу очереди эффективной. Это -
- peek () - получает элемент в начале очереди, не удаляя его.
- isfull () - Проверяет, заполнена ли очередь.
- isempty () - Проверяет, пуста ли очередь.
- В очереди мы всегда удаляем (или обращаемся) к данным, указанным передним указателем, и, помещая в очередь (или сохраняя) данные в очереди, мы обращаемся к заднему указателю. Пример очереди показан на рисунке 2.



Рисунок 2 – Пример очереди

2 Проектирование и разработка приложения

2.1. Функции приложения

Разработанная программа позволяет выполнять следующие функции:

- создание очереди абонентов АТС;
- инициализация или определение абонентов очереди АТС, их использование, а затем полное удаление из памяти;
- добавление новых участников в очередь АТС;
- удаление одного или всех участников из очереди АТС;
- примитивный вывод древовидной структуры;
- отображение абонентов АТС, сохраненных в очереди;
- загрузка и сохранение созданного списка подписчиков из и в файл соответственно;

2.2. Интерфейс пользователя

Интерфейс разработанного программного продукта представлен на рисунке 3.

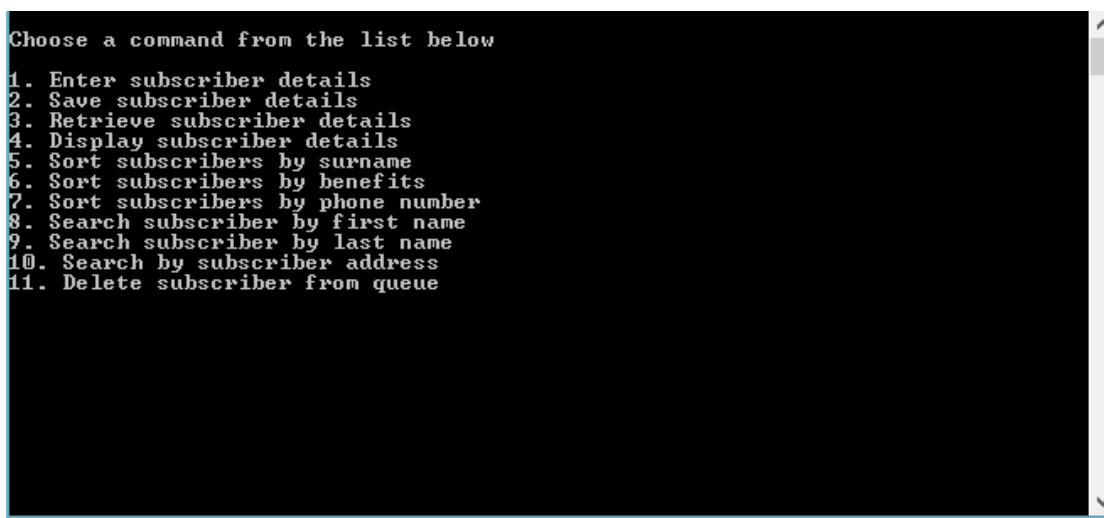


Рисунок 3 – Интерфейс приложения

При выборе пунктов меню от 1 до 11, программа выполняет выбранное действие.

Основной вариант работы приложения:

1. Введите данные подписчика;
2. Сохранить данные подписчика;
3. Получить данные подписчика;
4. Показать данные подписчика;
5. Сортировать подписчиков по фамилии;
6. Сортировка подписчиков по преимуществам;
7. Сортировка подписчиков по номеру телефона;
8. Поиск абонента по имени;
9. Поиск абонента по фамилии;
10. Поиск по адресу подписчика;
11. Удалить подписчика из очереди;

Альтернативный вариант работы приложения:

- 2а - введите имена файлов, в которые будет сохраняться очередь;
- 3а – ввод имени файла, из которого будет загружено дерево;

2.3. Логическая структура приложения

Диаграмма классов разработанного приложения представлена на рисунке 4.

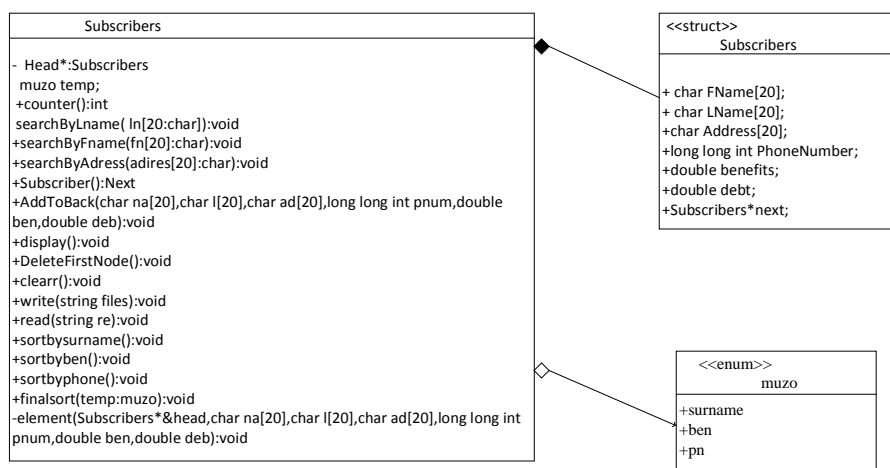


Рисунок 4 – Диаграмма классов

Структура node – содержит данные о каждом узле дерева.

Поля Node:

- FName - ключ узла (имя);
- LName– ключ узла (фамилия);
- Адрес - ключ узла (Адрес);
- PhoneNumber– ключ узла (номер телефона);
- benefits - ключ узла (льготы);
- debt - ключ узла (задолженность);
- next – указатель на следующий узел;

Класс Subscribers – Шаблонный класс, содержащий указатель на первый элемент очереди, как структуру, конструктор и методы.

Конструктор без параметров Subscriber() – инициализирующий новый объект класса, как пустой указатель (nullptr).

Поле Head - указатель на корень очереди.

Методы:

- int counter()- Функциональный метод, чтобы проверить, достигла ли очередь максимальное количество подписчиков.
- void searchByLname(char ln[20])- Функциональный метод для поиска по фамилии.
- void searchByFname(char fn[20])- Функциональный метод для поиска по имени.
- void searchByAdress(char adires[20])- Функциональный метод поиска по адресу.
- void AddToBack(char na[20],char l[20],char ad[20],long long int pnun,double ben,double deb)- Функциональный метод для добавления абонента АТС в очередь.

- `void display()`-Функциональный метод для отображения абонентов АТС в очереди.
- `void DeleteFirstNode`-Функциональный метод удаления абонента АТС из очереди.
- `void clearr()`-Функциональный метод удаления всех абонентов АТС из очереди.
- `void write(string files)`- Функциональный метод для сохранения всех абонентов АТС в файл.
- `void read(string re)`- Функция-метод для чтения всех абонентов АТС из файла.
- `void sortbysurname()`-Функция метода сортировки абонентов АТС по фамилии.
- `void sortbyben()`-Функциональный метод сортировки абонентов АТС по преимуществам.
- `void sortbyphone()`-Функциональный метод сортировки абонентов АТС по номеру телефона.
- `void finalsort(muzo temp)`- Функциональный метод сортировки абонентов АТС по выбранному типу сортировки.

Граф-схемы алгоритмов основных методов класса представлены на рисунках 5-11.

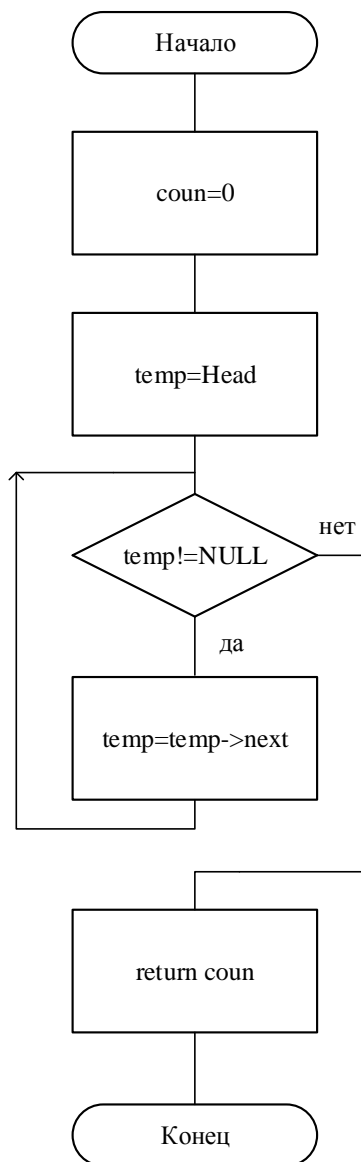


Рисунок 5 – Граф-схема метода int counter()

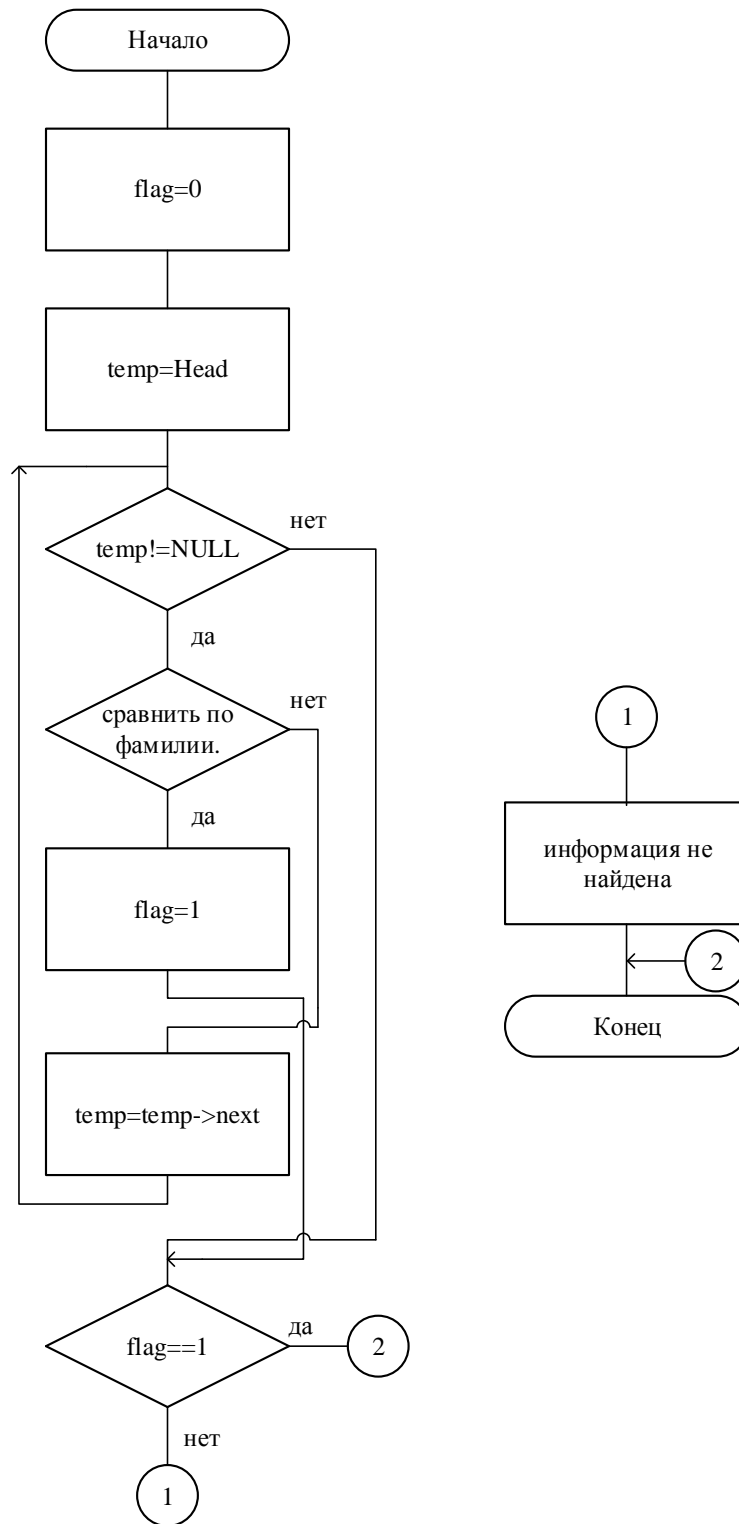


Рисунок 6 – Граф-схема метода void searchByLname(char ln[20])

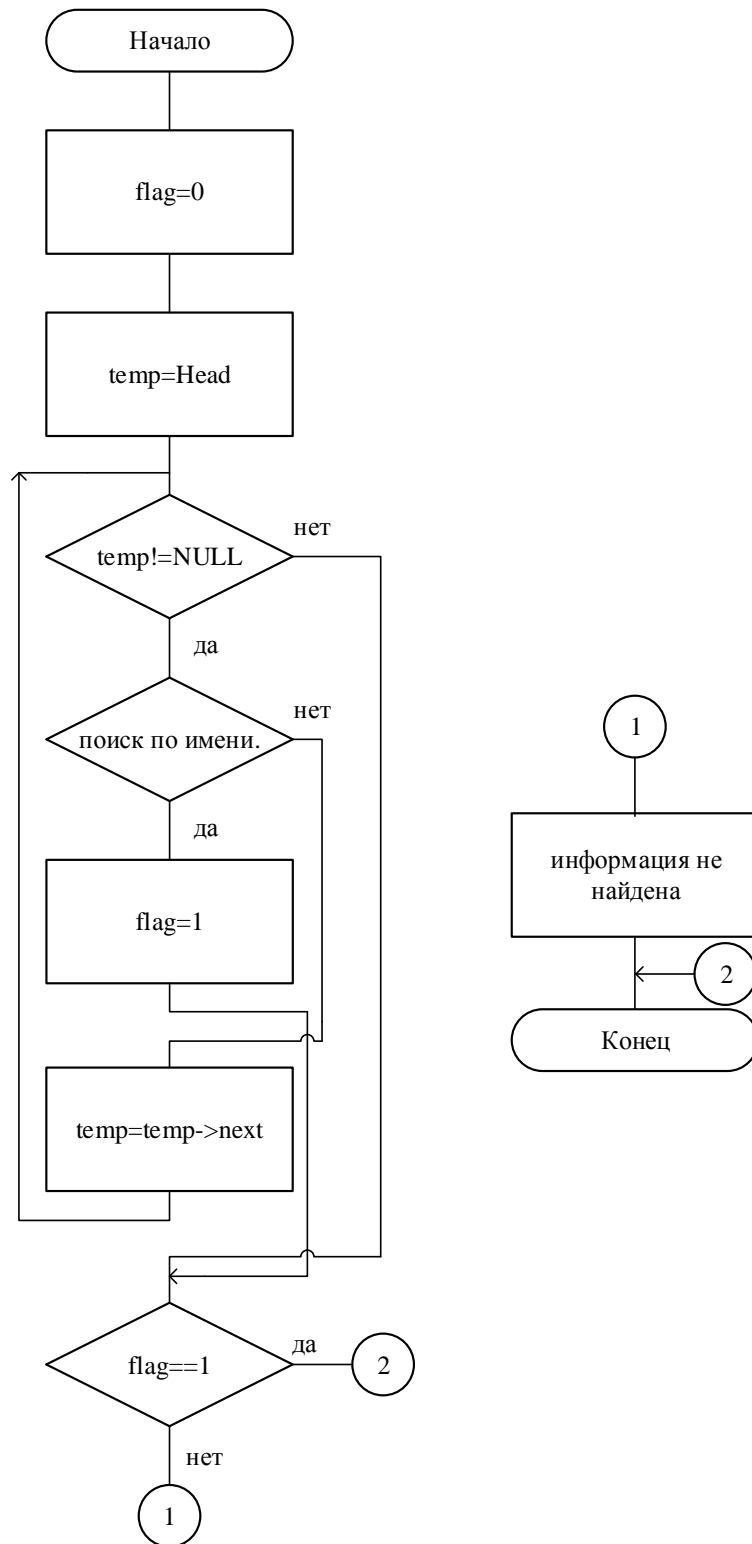


Рисунок 7 – Граф-схема метода void searchByFname(char fn[20])

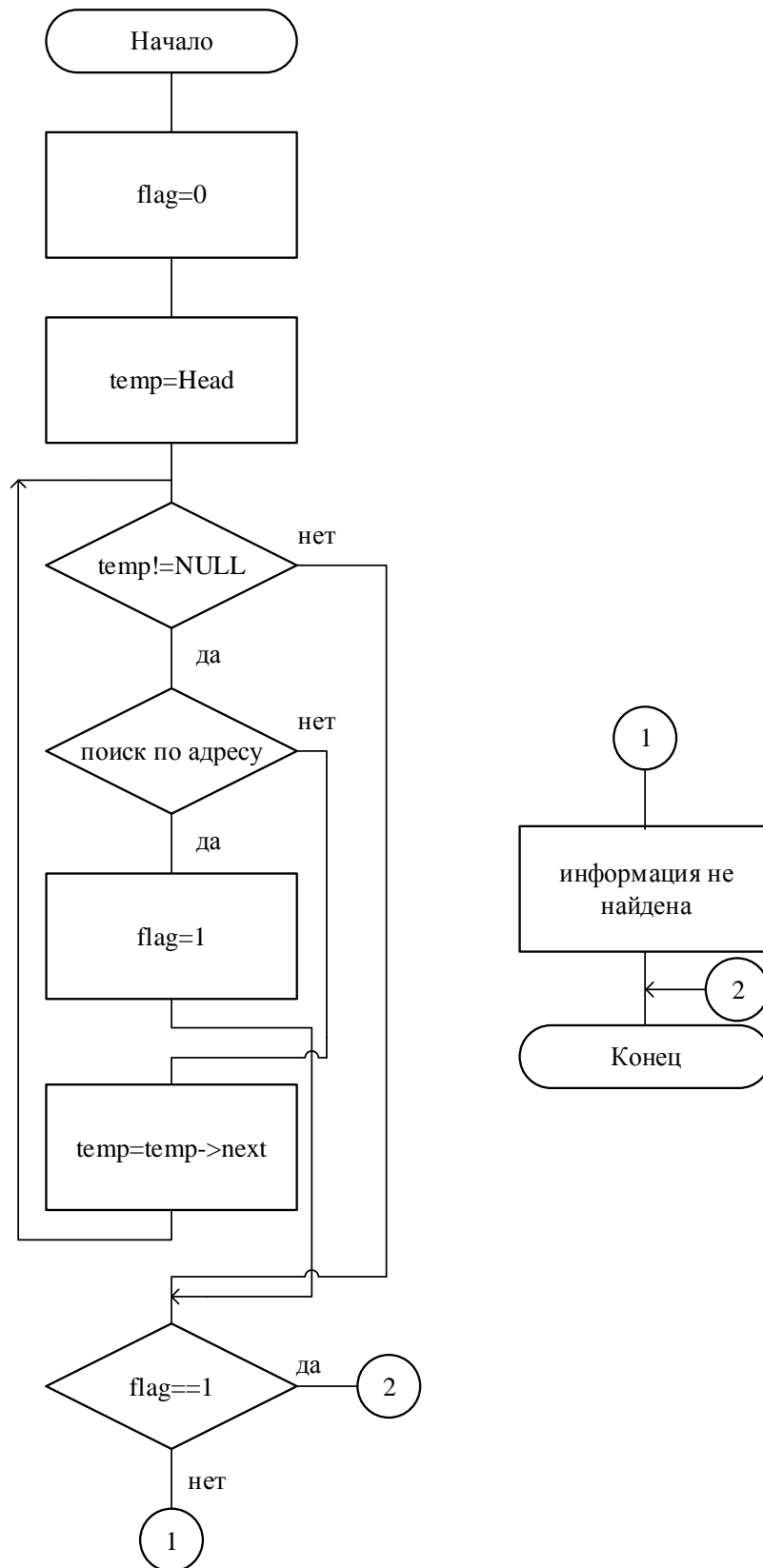


Рисунок 8 – Граф-схема метода void searchByAdress(char adires[20])

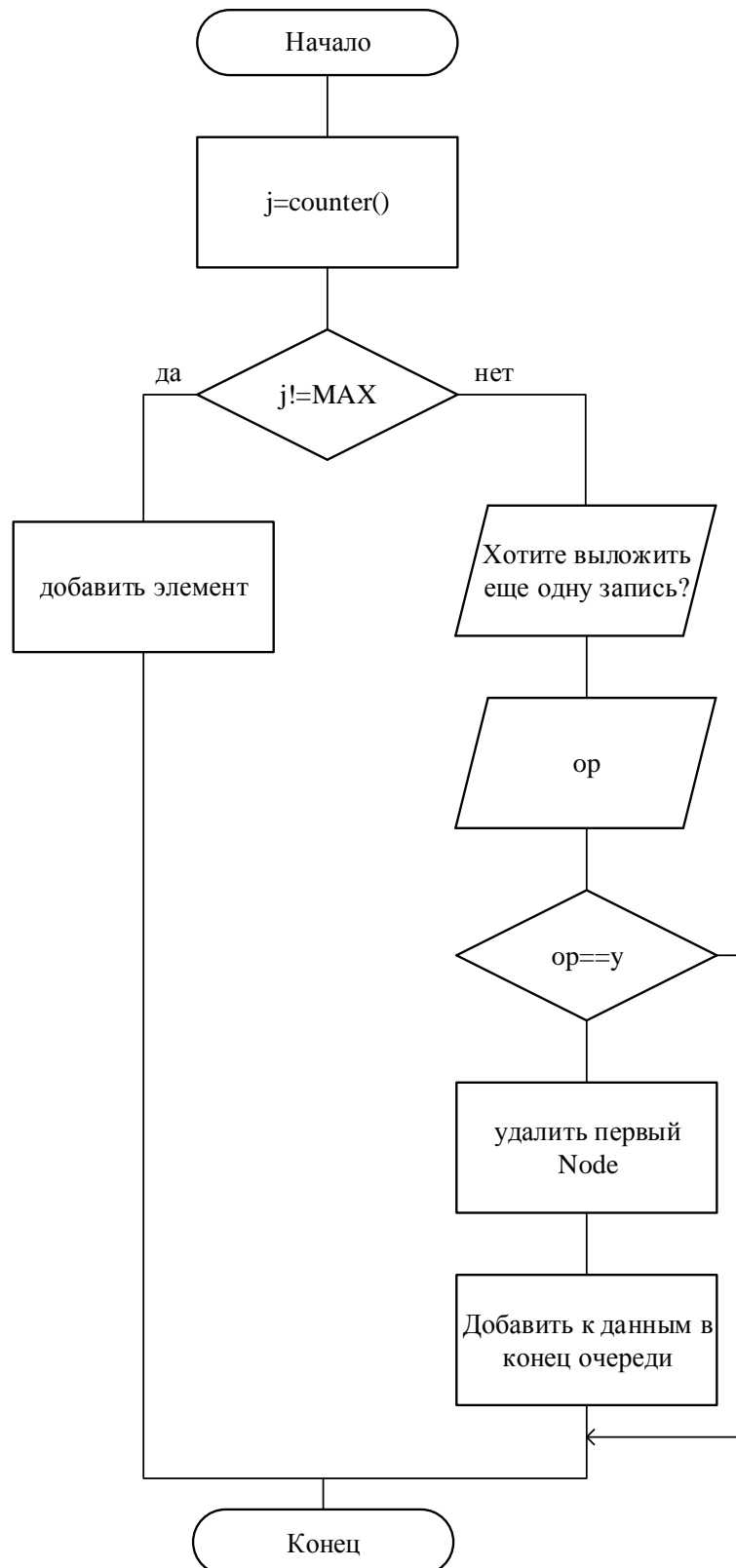


Рисунок 9 – Граф-схема метода void AddToBack(char na[20],char l[20],char ad[20],long long int pnum,double ben,double deb)

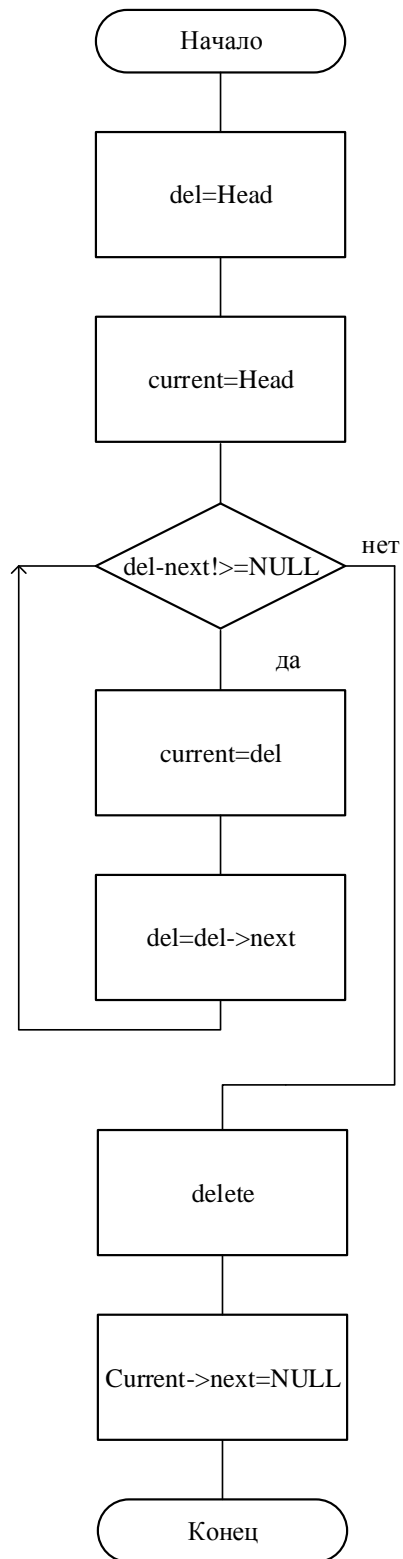


Рисунок 10 – Граф-схемы методов `void DeleteFirstNode()`

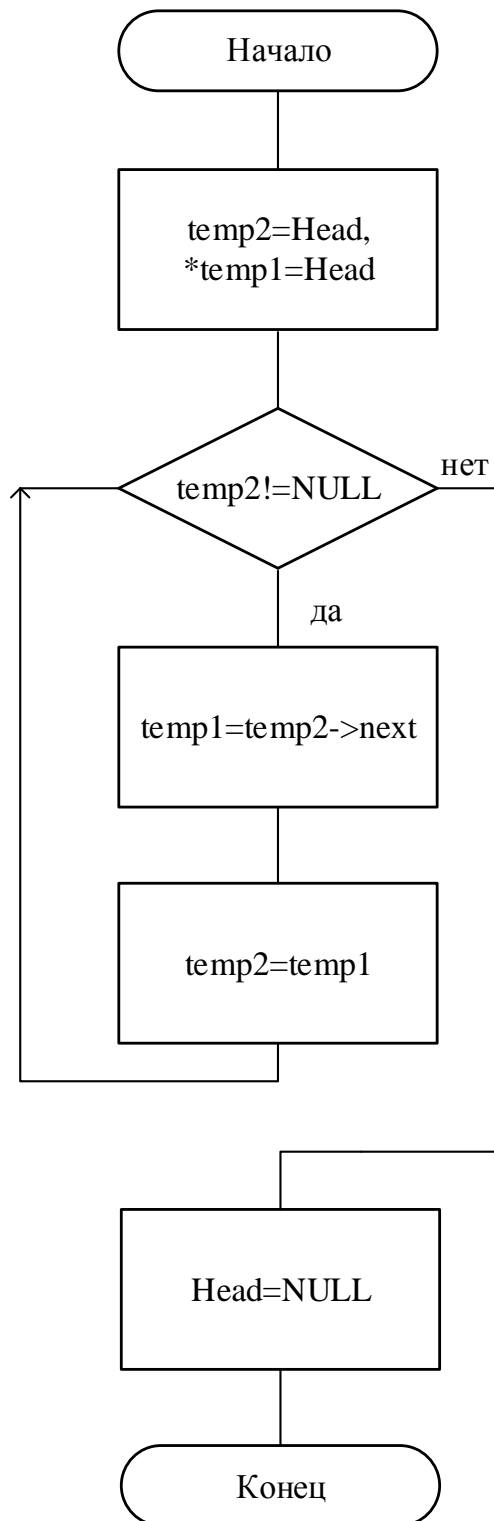


Рисунок 11 – Граф-схема метода void clear().

2.4. Распределение исходного кода по файлам

Исходный код разработанного приложения разделён на модули, некоторые из которых используют подключаемые библиотеки.

- `main.cpp` - это модуль, содержащий метод интерфейса главного меню.
- `Subscriber.h` - это модуль, содержащий описание структуры узла, класса, с прототипами всех методов и методов для вывода, сохранения и загрузки.
- `Subscriber.cpp` - модуль, который содержит все методы класса.
- Связанные библиотеки:
- `fstream` - библиотека для работы с файлами, используемая для сохранения и загрузки данных;

Диаграмма компонентов разработанной программы представлена на рисунке 12.

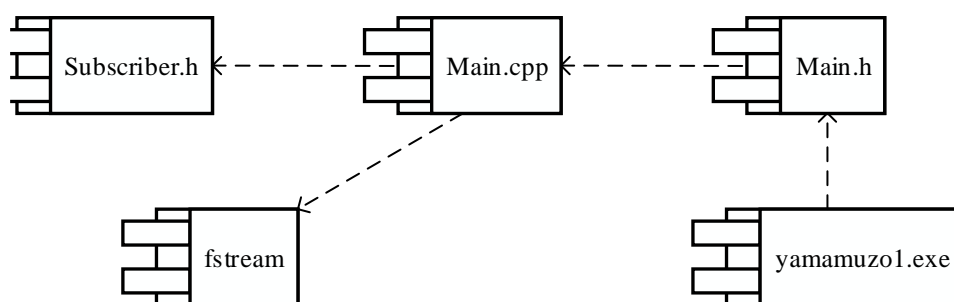


Рисунок 12 – Диаграмма компонентов

2.5. Оценка сложности базового алгоритма

Рассмотрим алгоритм создания очереди и определим его сложность.

```
finalsort(muzo temp)
{
    if(temp==surname) //2
    {
        sortbysurname() //30n;
        return;
    } 2*30n=60n

    if(temp==pn) //2
    {
        sortbyphone() //150n2;
        return;
    } 2*150n2=300n2

    if(temp==ben) //2
    {
        sortbyben() //60n2;
        return;
    }
    } 2*60n2=120n2

sortbysurname()
{
    Subscribers*temp,*temp1;
    for (temp=Head;temp->next!=NULL;temp=temp->next) //n
```

```

        {
            for      (temp1=temp->next;temp1!=NULL;temp1=temp1-
>next) 5n
            {
                if (strcmpi (temp->LName,temp1->LName)>0) //1
                {

                    swap(temp->Address,temp1->Address); //1
                    swap(temp->benefits,temp1->benefits); //1
                    swap(temp->debt,temp1->debt); //1
                    swap(temp->FName,temp1->FName); //1
                    swap(temp->LName,temp1->LName); //1
                    swap(temp->PhoneNumber,temp1-
>PhoneNumber) //1; }
                }
            }
        }
N*(1*(1+1+1+1+1+1))=5n*6=30n
sortbyben()
{
    Subscribers*temp,*temp1,*temp3;
    for (temp=Head;temp->next!=NULL;temp=temp->next) //5n
    {
        for      (temp1=temp->next;temp1!=NULL;temp1=temp1-
>next) //5n
        {
            if (temp->benefits>temp1->benefits) //1 {

```

```

        swap(temp->Address,temp1->Address)//1;
        swap(temp->benefits,temp1->benefits)//1;
        swap(temp->debt,temp1->debt)//1;
        swap(temp->FName,temp1->FName)//1;
        swap(temp->LName,temp1->LName)//1;
        swap(temp->PhoneNumber,temp1-
>PhoneNumber)//1; }

    }

}

5n*(5n*(1*(1+1+1+1+1+1)))=25n2*6=150n2
sortbyphone()
{
    Subscribers*temp,*temp1;
    for (temp=Head;temp->next!=NULL;temp=temp->next)//5n
    {
        for      (temp1=temp->next;temp1!=NULL;temp1=temp1-
>next)//5n
        {
            if (temp->PhoneNumber>temp1->PhoneNumber)//n
            {

                swap(temp->Address,temp1->Address);//1
                swap(temp->benefits,temp1->benefits);//1
                swap(temp->debt,temp1->debt);//1
                swap(temp->FName,temp1->FName);//1

```

```

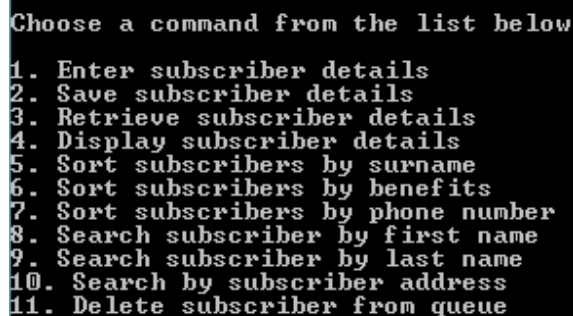
        swap(temp->LName, temp1->LName); //1
        swap(temp->PhoneNumber, temp1->
>PhoneNumber) //1; }
    }
}
}
5n+5n*(n(1+1+1+1+1+1))=10n^2*6=60n^2

```

Сложность алгоритма $120n^2+300n^2+60n$, это асимптотическая сложность в худшем случае при создании очереди равна $O(n^2)$.

3 Тестирование

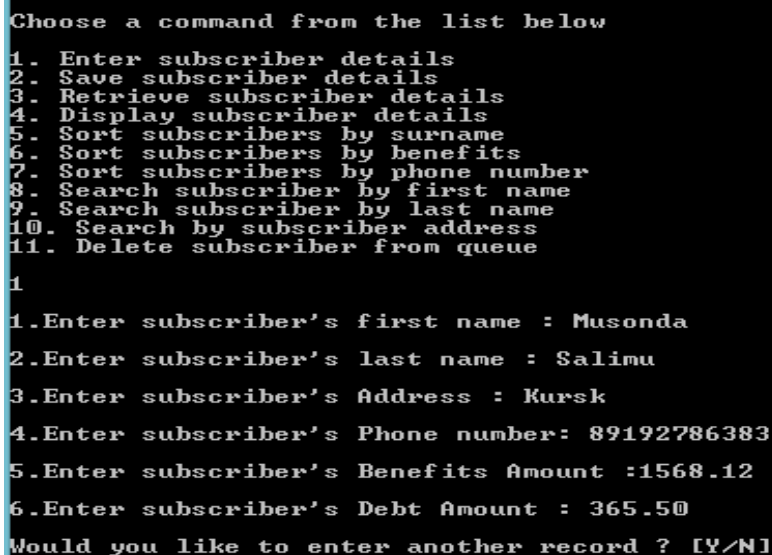
При запуске разработанной программы, на экране появится меню, как показано на рисунке ниже, представлен на рисунке 13.



```
Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
```

Рисунок 13 – Создание нового очереди

При запуске разработанной программы, вариант 1 для добавления нового АТС абонента в очередь. Когда выбран вариант 1, пользователю предлагается ввести данные о АТС абонента, добавляемом в очередь, и выбрать переход в главное меню или добавить дополнительных АТС абонентов. Чтобы сохранить эти данные, мы переходим в главное меню и выбираем опцию 2. Окно загрузки показано на рисунке 14.



```
Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
1
1.Enter subscriber's first name : Musonda
2.Enter subscriber's last name : Salimu
3.Enter subscriber's Address : Kursk
4.Enter subscriber's Phone number: 89192786383
5.Enter subscriber's Benefits Amount :1568.12
6.Enter subscriber's Debt Amount : 365.50
Would you like to enter another record ? [Y/N]
_
```

Рисунок 14 – Создание первого нового АТС абонента в очереди.

```

3.Enter subscriber's Address : Kursk
4.Enter subscriber's Phone number: 89192786383
5.Enter subscriber's Benefits Amount :1568.12
6.Enter subscriber's Debt Amount : 365.50
Would you like to enter another record ? [Y/N]
y
1.Enter subscriber's first name : John
2.Enter subscriber's last name : Sasha
3.Enter subscriber's Address : Moscow
4.Enter subscriber's Phone number: 89606287864
5.Enter subscriber's Benefits Amount :25.2
6.Enter subscriber's Debt Amount : 10.99
Would you like to enter another record ? [Y/N]

```

Рисунок 15 – Создание второго АТС абонента в очереди.

```

3.Enter subscriber's Address : Moscow
4.Enter subscriber's Phone number: 89606287864
5.Enter subscriber's Benefits Amount :25.2
6.Enter subscriber's Debt Amount : 10.99
Would you like to enter another record ? [Y/N]
y
1.Enter subscriber's first name : Peter
2.Enter subscriber's last name : Alex
3.Enter subscriber's Address : Kazan
4.Enter subscriber's Phone number: 89565211234
5.Enter subscriber's Benefits Amount :100.1
6.Enter subscriber's Debt Amount : 15.6
Would you like to enter another record ? [Y/N]

```

Рисунок 16 – Создание третьего АТС абонента в очереди.

```

6.Enter subscriber's Debt Amount : 15.6
Would you like to enter another record ? [Y/N]
n
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]
y
Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
2
To which file would you like to save your data? muzo.bin
YOUR DATA HAS BEEN SAVED! WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]

```

Рисунок 17 – Сохранение данных в файл.

Для загрузки сохраненного очереди, вы должны выбрать опцию 3 в главном меню. Пример загрузки показан на рисунке 18.

```
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
3
From which file would you like to retrieve your data? muzo.bin
YOUR DATA HAS BEEN RETRIEVED! WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]
y
Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
4
Musonda Salimu Kursk 89192786383 1568.12 365.5
John Sasha Moscow 89606287864 25.2 10.99
Peter Alex Kazan 89565211234 100.1 15.6
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]
```

Рисунок 18 – Загрузка элементов из файла.

Для поиска элемента в очереди существует три возможных варианта поиска: фамилия, имя и адрес. Примеры поиска предметов (фамилия, имя и адрес) представлены на рисунках 19-21.

```
Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
8
Enter the first name of the Subscriber Peter
Information found :Peter Alex Kazan 100.1 15.6
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]
```

Рисунок 19 – Поиск элемента по имени в файле.

```

9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
8
Enter the first name of the Subscriber Peter
Information found :Peter Alex Kazan 100.1 15.6
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]
y
Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
9
Enter the last name of the Subscriber Sasha
Information found :John Sasha Moscow 25.2 10.99
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]

```

Рисунок 20 – Поиск элемента по фамилии в файле.

```

From which file would you like to retrieve your data? pbx.bin
YOUR DATA HAS BEEN REVRITVED! WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]
y
Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
10
Enter the address of the Subscriber Kursk
Information found :Musonda Salimu Kursk 1568.12 365.5
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]

```

Рисунок 21 – Поиск товара по адресу в файле.

Для сортировки элементов в очереди есть три возможных варианта: по фамилии, льготам и номеру телефона. Примеры поиска предметов (по фамилии, льготе и номеру телефона) представлены на рисунках 22-24.

```
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue

4
Musonda Salimu Kursk 89606287864 25.2 10.99
John Sasha Moscow 89565211234 100.1 15.6
Peter Alex Kazan 89192786383 1568.12 365.5
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]
y

Choose a command from the list below

1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue

5

YOUR DATA HAS BEEN SUCCESFULLY SORTED BY SURNAME? WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]
y

Choose a command from the list below

1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue

4
Musonda Salimu Kursk 89192786383 1568.12 365.5
John Sasha Moscow 89606287864 25.2 10.99
Peter Alex Kazan 89565211234 100.1 15.6
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]
```

Рисунок 22 – Сортировка по фамилии.

```

Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
6
YOUR DATA HAS BEEN SUCCESFULLY SORTED BY BENEFITS! WOULD YOU LIKE TO GO BACK TO
THE MAIN MENU? [Y/N]
y
Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
4
Musonda Salimu Kursk 89606287864 25.2 10.99
John Sasha Moscow 89565211234 100.1 15.6
Peter Alex Kazan 89192786383 1568.12 365.5
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]

```

Рисунок 23 – Сортировка по льготе

```

Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
7
YOUR DATA HAS BEEN SUCCESFULLY SORTED BY PHONE NUMBER! WOULD YOU LIKE TO GO BACK
TO THE MAIN MENU? [Y/N]
y
Choose a command from the list below
1. Enter subscriber details
2. Save subscriber details
3. Retrieve subscriber details
4. Display subscriber details
5. Sort subscribers by surname
6. Sort subscribers by benefits
7. Sort subscribers by phone number
8. Search subscriber by first name
9. Search subscriber by last name
10. Search by subscriber address
11. Delete subscriber from queue
4
Musonda Salimu Kursk 89192786383 1568.12 365.5
John Sasha Moscow 89565211234 100.1 15.6
Peter Alex Kazan 89606287864 25.2 10.99
WOULD YOU LIKE TO GO BACK TO THE MAIN MENU? [Y/N]

```

Рисунок 24 – Сортировка по номеру телефона.

ЗАКЛЮЧЕНИЕ

В процессе разработки приложения для решения конкретной задачи была изучена специальная литература по теме проекта: структура очереди, документация по среде разработки Code::blocks 17.12, документация по языку C++. Деревообработка алгоритмы и класс и структура были разработаны. Также были изучены язык библиотеки, работа с файловыми потоками, ввод и вывод информации.

Таким образом, в курсовой работе рассмотрена актуальность очередей. Взаимодействие этих двух структур облегчает работу с большим объемом информации, рутинную работу с бумажными носителями, а также гарантирует качество хранения информации, экономит время, что сегодня очень важно.

Мы исследовали важность очередей. Рассматривается классификация очередей, которые делятся по сложности, способу представления, наличию связей, изменчивости, характеру упорядочения элементов, по типу памяти. Особой точкой изучения этой работы является очередь. Очередь - это такой последовательный список с переменной длиной, включение элементов в которое происходит с одной стороны, и исключение элементов с другой стороны списка. Очередь иногда называют циклической памятью или списком типов FIFO («первым пришел - первым вышел» - «первым включен - первым включен, вышел»).

Выбор правильного представления очереди является ключом к успешному программированию и может оказать большее влияние на производительность программы, чем детали используемого алгоритма.

Усовершенствовать данное приложения можно с помощью улучшения интерфейса, добавления новых методов работы со структурой данных очереди, графического вывода, контекстного меню.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. П
2. Рышкин Е.В. Структуры данных и алгоритмы: реализация на C/C++. - СПб.: ФТК СПбГПУ, 2009.- 200 с.
3. Стенли Б. Липпман, Ж. Ложойе. Язык программирования C++ Базовый курс. – М.: Вильямс, 2014. – 1256с.
4. Сайт вопросов и ответов для программистов [Сайт]. URL: <https://ru.stackoverflow.com/> (дата обращения: 08.05.2019).
5. Форум программистов и сисадминов [Сайт]. URL: <http://www.cyberforum.ru/> (дата обращения: 10.05.2019).
6. Cppreference [Сайт]. URL: <https://en.cppreference.com/w/> (дата обращения: 12.04.2019).

.

A

.

C

/

C

+

+

.

П

р

о

г

р

а

м

ПРИЛОЖЕНИЕ А
ТЕКСТ ПРОГРАММЫ

main.cpp

```
#include <iostream>
#include <fstream>
#include <string.h>
#include <iomanip>
#ifndef SUBSCRIBER_H
#define SUBSCRIBER_H
using namespace std;
enum muzo{surname,ben,pn};
struct Subscribers
{
char FName[20];
char LName[20];
char Address[20];
long long int PhoneNumber;
double benefits;
double debt;
Subscribers* next;
};
//Subscribers *Head;
class Subscriber
{
private:
Subscribers* Head;
```

```

        void      element(Subscribers*&head,char      na[20],char
l[20],char  ad[20],long long int  pnum,double  ben,double
deb);

public:
        muzo temp;
        int counter();
void searchByLname(char ln[20]);
void searchByFname(char fn[20]);
void searchByAdress(char adires[20]);
//~Subscriber();
void AddToBack(char na[20],char l[20],char ad[20],long
long int  pnum,double ben,double deb);
        //Добавить элемент в конец очереди
void display();
void DeleteFirstNode(); //Удалить первый элемент из
очереди
void Print(); //Читаем первый элемент очереди
//Subscribers(); //Прототип конструктора копирования
//Прототип оператора присваивания
void clearr();
void write(string files);
void read(string re);
void sortbysurname();
void sortbyben();
void sortbyphone();
void finalsort(muzo temp);
};

```

Subscriber.cpp

```
#include <iostream>
#include <fstream>
#include <string.h>
#include <iomanip>
#include "Subscriber.h"
#define MAX 10
using namespace std;
void Subscriber::element(Subscribers*&head, char na[20],
char l[20],char ad[20],long long int pnum, double ben,
double deb)
{
Subscribers*temp=new Subscribers();
strcpy(temp->FName,na);
strcpy(temp->LName,l);
strcpy(temp->Address,ad);
temp->PhoneNumber=pnum;
temp->benefits=ben;
temp->debt=deb;
temp->next=NULL;
if(head==NULL)
{
head=temp;
return;
}
else {
temp->next=Head;
```

```

head=temp;}
}
//Subscriber::Subscriber{}
/*bool Subscriber::isempty() {
    Subscribers*temp=new Subscribers();
    if(front < 0 || front > Subscribers)
        return true;
    else
        return false;
}*/
int Subscriber::counter()
{
    int coun=0;//1
    Subscribers*temp=Head;//1
    while(temp!=NULL) //N
    { //N*
        coun++; //2
        temp=temp->next; //2
    } //4*N+N
    return coun; //1
} //5*N+3
void Subscriber::searchByLname(char ln[20])
{
    int flag=0;
    Subscribers*temp=Head;
    while (temp!=NULL)
    {
        if (strcmpi(temp->LName,ln)==0)

```

```

        {
            flag=1;
            break;
        }
        temp=temp->next;
    }
    if (flag==1)
        cout<<"Information found : "<< temp->FName<<"
"<<temp->LName<<"      "<<temp->Address<<"      "<<temp-
>benefits<<" "<<temp->debt<<endl;
        else cout<<"Information not found "<<endl;
    }

void Subscriber::searchByFname(char fn[20])
{
    int flag=0;
    while (Head!=NULL)
    {
        if (strcmpi(Head->FName,fn)==0)
        {
            flag=1;
            break;
        }
        Head=Head->next;
    }
    if (flag==1)

```

```

        cout<<"Information found : "<< Head->FName<<"
"<<Head->LName<<"          "<<Head->Address<<"          "<<Head-
>benefits<<" "<<Head->debt<<endl;
        else cout<<"Information not found "<<endl;
    }

```

```

void Subscriber::searchByAdress(char adires[20])

```

```

{
    int flag=0;
    while (Head!=NULL)
    {
        if (strcmpi(Head->FName,adires)==0)
        {
            flag=1;
            break;
        }
        Head=Head->next;
    }
    if (flag==1)
        cout<<"Information found : "<< Head->FName<<"
"<<Head->LName<<"          "<<Head->Address<<"          "<<Head-
>benefits<<" "<<Head->debt<<endl;
        else cout<<"Information not found "<<endl;
    }
}

```

```

Subscriber::Subscriber()

```

```

{

```

```

Head=NULL;
}

void Subscriber::AddToBack(char na[20],char l[20],char
ad[20],long long int pnum,double ben,double deb)
{ char op;
  int j=counter();/////5*N+3
  if (j!=MAX)//1
element(Head,na,l,ad,pnum,ben,deb);
else {
        cout<<"\nLIST IS FULL, WOULD YOU LIKE TO POP OUT
ONE RECORD [Y/N]\n";
        cin>>op;
        if (op=='y' || op=='Y')
        {
                DeleteFirstNode();
                AddToBack(na,l,ad,pnum,ben,deb);

        }

}

}

void Subscriber::display()
{
        Subscribers*temp=Head;
        if (Head!=NULL) {

```

```

        while (temp!=NULL)
        {
            //      cout<<temp->Address    <<"    "<<temp->debt<<"
            "<<temp->FName<<"            "<<temp->LName<<"            "<<temp-
            >PhoneNumber<<" "<<temp->benefits<<endl;
            cout<<temp->FName <<" "<<temp->LName<<" "<<temp-
            >Address<<" "<<temp->PhoneNumber<<" "<<temp->benefits<<"
            "<<temp->debt<<endl;
            temp=temp->next;
        }
        else cout<<"File is empty \n";
    }
void Subscriber::DeleteFirstNode()
{
    Subscribers*del=Head;
    Subscribers*current=Head;
    while (del->next!=NULL)
    {
        current=del;
        del=del->next;
    }
    delete (del);
    current->next=NULL;
};
void Subscriber:: clearr()
{
    Subscribers*temp2=Head, *temp1=Head;
    while (temp2!=NULL)

```



```

    {
        temp1=temp2->next;
        delete temp2;
        temp2=temp1;
    }
    Head=NULL;
}

void Subscriber:: write(string files)
{
    ofstream writetofile(files.c_str(),ios::binary);
    Subscribers*temp=Head;
    while (temp!=NULL)
    {
        writetofile.write(reinterpret_cast<char*>(&temp-
>FName),sizeof(temp->FName));
        writetofile.write(reinterpret_cast<char*>(&temp-
>LName),sizeof(temp->LName));
        writetofile.write(reinterpret_cast<char*>(&temp-
>Address),sizeof(temp->Address));
        writetofile.write(reinterpret_cast<char*>(&temp-
>benefits),sizeof(temp->benefits));
        writetofile.write(reinterpret_cast<char*>(&temp-
>debt),sizeof(temp->debt));
        writetofile.write(reinterpret_cast<char*>(&temp-
>PhoneNumber),sizeof(temp->PhoneNumber));
        temp=temp->next;
    }
}

```

```

}
void Subscriber::read(string re)
{
    Subscribers temp;
        ifstream readfrom(re.c_str()); int j,g;
        // seekg(0,ios::end);
        readfrom.seekg(0,ios::end);
        j=readfrom.tellg();
        g=j/sizeof(temp);
        readfrom.seekg(0,ios::beg);
        // cout<<g;
        char na[20],l[20],ad[20];long long int pnum;
double ben,deb;
        for (int i=0;i<=g;i++)
        {

readfrom.read(reinterpret_cast<char*>(&na),sizeof(na));

readfrom.read(reinterpret_cast<char*>(&l),sizeof(l));

readfrom.read(reinterpret_cast<char*>(&ad),sizeof(ad));

readfrom.read(reinterpret_cast<char*>(&ben),sizeof(ben));

readfrom.read(reinterpret_cast<char*>(&deb),sizeof(deb));

```

```

readfrom.read(reinterpret_cast<char*>(&pnum), sizeof(pnum)
);

        AddToBack (na, l, ad, pnum, ben, deb);
    }

}

void Subscriber::sortbysurname()
{
    Subscribers*temp,*temp1;
    for (temp=Head;temp->next!=NULL;temp=temp->next)
    {
        for      (temp1=temp->next;temp1!=NULL;temp1=temp1-
>next)
        {
            if (strcmpi(temp->LName,temp1->LName)>0)
            {

                swap(temp->Address,temp1->Address);
                swap(temp->benefits,temp1->benefits);
                swap(temp->debt,temp1->debt);
                swap(temp->FName,temp1->FName);
                swap(temp->LName,temp1->LName);
                swap(temp->PhoneNumber,temp1-
>PhoneNumber);}
        }
    }

}

void Subscriber::sortbyben()

```

```

{
    Subscribers*temp,*temp1,*temp3;
    for (temp=Head;temp->next!=NULL;temp=temp->next)
    {
        for      (temp1=temp->next;temp1!=NULL;temp1=temp1-
>next)
        {
            if (temp->benefits>temp1->benefits) {

                swap(temp->Address,temp1->Address);
                swap(temp->benefits,temp1->benefits);
                swap(temp->debt,temp1->debt);
                swap(temp->FName,temp1->FName);
                swap(temp->LName,temp1->LName);
                swap(temp->PhoneNumber,temp1->PhoneNumber);
            }

        }
    }
}

void Subscriber::sortbyphone()
{
    Subscribers*temp,*temp1;
    for (temp=Head;temp->next!=NULL;temp=temp->next)
    {
        for      (temp1=temp->next;temp1!=NULL;temp1=temp1-
>next)

```

```

        {
            if (temp->PhoneNumber>templ->PhoneNumber)
            {

                swap(temp->Address,templ->Address);
                swap(temp->benefits,templ->benefits);
                swap(temp->debt,templ->debt);
                swap(temp->FName,templ->FName);
                swap(temp->LName,templ->LName);
                swap(temp->PhoneNumber,templ-
>PhoneNumber);}
        }
    }
}

void Subscriber::finalsort(muzo temp)
{
    if(temp==surname)
    {
        sortbysurname();
        return;
    }

    if(temp==pn)
    {
        sortbyphone();
        return;
    }

    if(temp==ben)

```

```

    {
        sortbyben();
        return;
    }
}

```

Subscriber.h

```

#include <string.h>
#include <iomanip>
#include <iostream>
#include <fstream>
#ifndef SUBSCRIBER_H
#define SUBSCRIBER_H
using namespace std;
enum muzo{surname,ben,pn};
struct Subscribers
{
    char FName[20];
    char LName[20];
    char Address[20];
    long long int PhoneNumber;
    double benefits;
    double debt;
    Subscribers* next;
};
//Subscribers *Head;
class Subscriber
{
    private:

```

```

Subscribers* Head;

void element(Subscribers*&head,char na[20],char l[20],char
ad[20],long long int pnum,double ben,double deb);

public:
    muzo temp;
    int counter();
void searchByLname(char ln[20]);
void searchByFname(char fn[20]);
void searchByAdress(char adires[20]);
Subscriber();
void AddToBack(char na[20],char l[20],char ad[20],long
long int pnum,double ben,double deb);
void display();
void DeleteFirstNode();
void Print();
void clearr();
void write(string files);
void read(string re);
void sortbysurname();
void sortbyben();
void sortbyphone();
void finalsort(muzo temp);
};

#endif // SUBSCRIBER_H

```

ПРИЛОЖЕНИЕ Б

ВНЕШНИЙ ВИД ГРАФИЧЕСКОГО МАТЕРИАЛА

На рисунке Б.1 представлена диаграмма классов.

На рисунке Б.2 представлена граф-схема метода `void finalsorrt(muzo temp)`.

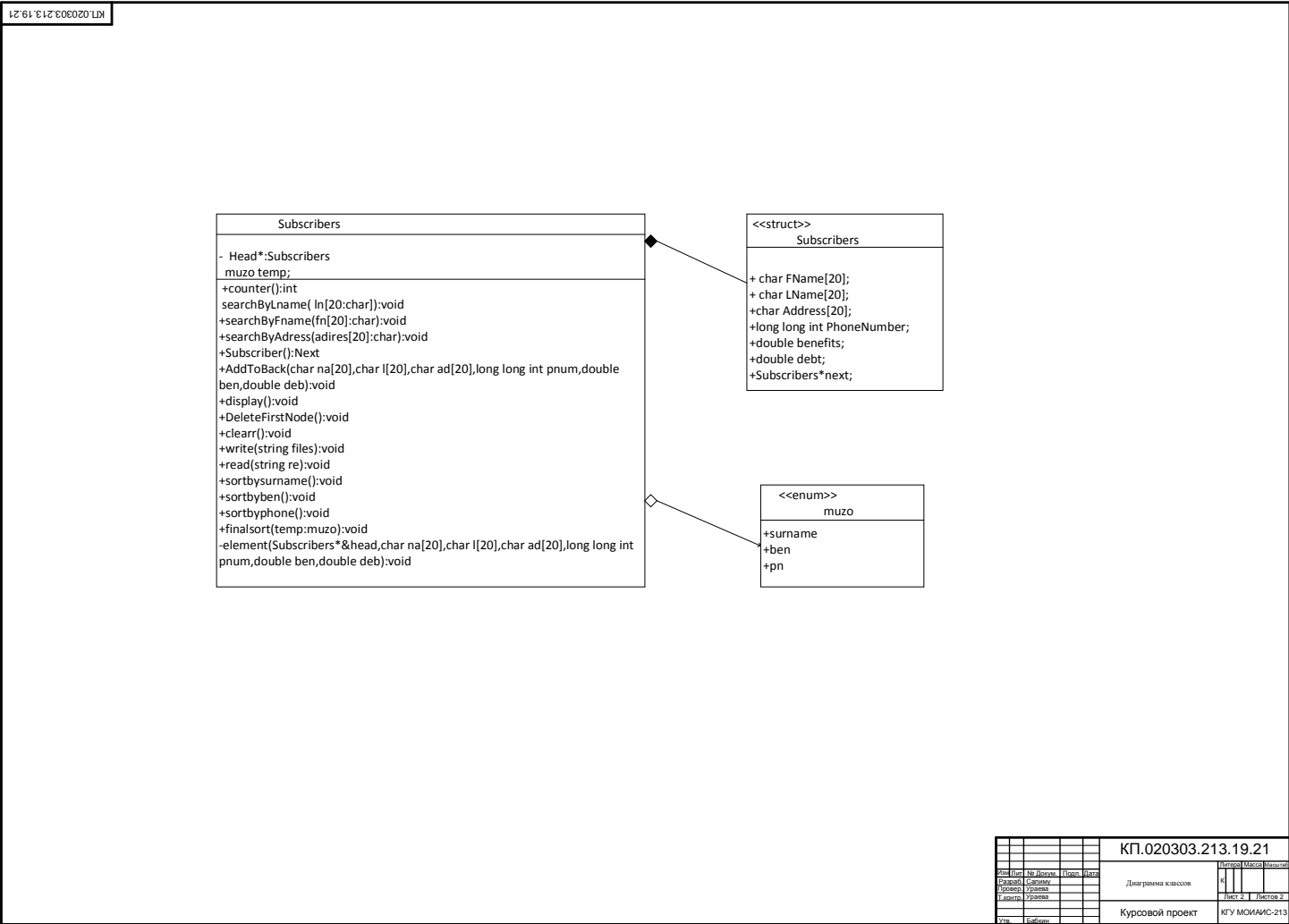


Рисунок Б.1 – Диаграмма классов

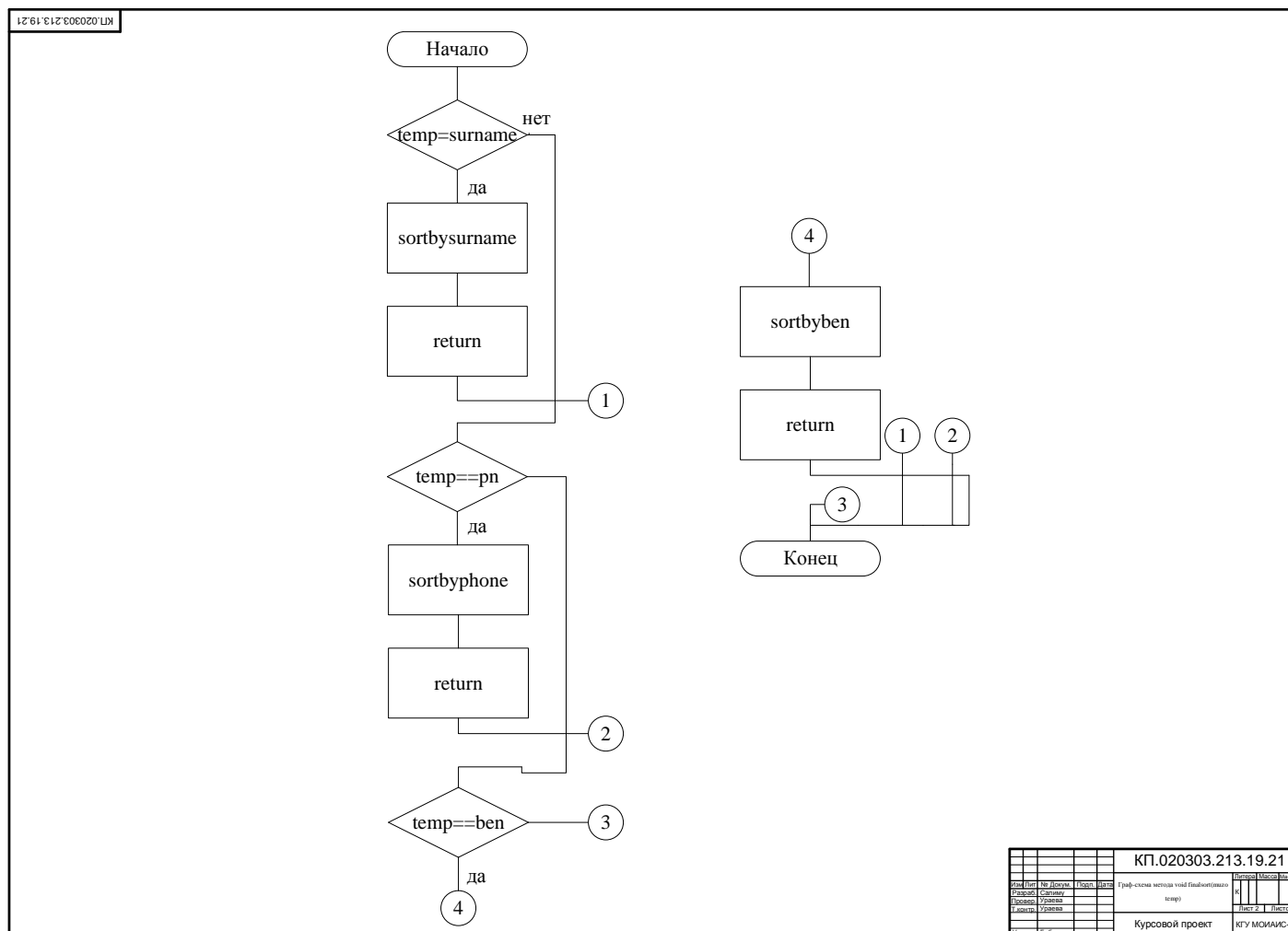


Рисунок Б.2 – Граф-схема метода void finalsor(muzo temp)