Оглавление

[1. Общие положения 2](#_Toc440728662)

[2. Алгоритмы и структуры данных 2](#_Toc440728663)

[3. Описание программы 5](#_Toc440728664)

[4. Тестирование программы 8](#_Toc440728665)

[5. Заключение 13](#_Toc440728666)

# Общие положения

Задачей проекта является разработка информационной системы, которая выполняет функции учёта всех имеющихся клиентов поликлиники, а также медицинского персонала и направлений на приём к специалисту с использованием следующих структур данных и алгоритмов:

* Закрытое хеширование с двойным хешированием.
* Метод сортировки извлечением.
* Циклический однонаправленный список.
* Симметричный обход АВЛ-дерева.
* Прямой алгоритм поиска слова в тексте.

Задача является актуальной на текущий день. Данная система предназначена для упрощения работы сотрудников регистратуры поликлиники. Система позволяет работать с регистрацией и удалением пациентов поликлиники и медицинского персонала, с выдачей направлений к нужному для пациента врачу, с возвратом направлений, а также работать с поиском пациентов и врачей.

Система «База данных поликлиники» позволит уменьшить нагрузку на регистратуру, и кроме того ускорит её работу.

# Алгоритмы и структуры данных

***Хеш-таблица. Закрытое хеширование с двойным хешированием.***Двойное хеширование основано на нелинейной адресации, достигаемой за счёт суммирования значений основной и дополнительной хеш-функций, что позволяет уменьшить число коллизий:

a= hash\_one(key) + i\*hash\_two(key).

Ячейка хеш-таблицы имеет следующую структуру:

// структура данных пациента

struct patient

{

string key;// формат ключа: MM-NNNNNN

string name; // ФИО

unsigned int bday[3]; // дата рождения

string adress; // адрес

string workplace; // место работы

};

// структура ячейки хэш-таблицы

struct hash\_tab

{

patient pat; // структура данных пациента

bool is\_empty; // пуста ли ячейка?

};

Реализации в вычислительных машинах будет следующей: сформированный ключ формата «MM-NNNNNN», где MM – номер участка, NNNNNN – порядковый номер, передаётся в функцию, для дальнейшего хеширования. Далее производится суммирование кодов символов ключа:

Результат = Код i-го символа ключа \* 3i

Затем результат отправляется в функции хеширования:

* Первое хеширование: остаток от деления результата суммирования кодов символов ключа и размера таблицы.
* Второе хеширование: остаток от деления, увеличенный на 1, результата суммирования кодов символов ключа и размера таблицы, уменьшенной на 1.

После ведётся подсчёт адреса согласно формуле:

Адрес=hash\_one(key) + i\*hash\_two(key), где i – номер шага.

Начальное значение i = 0. Если ячейка хеш-таблицы под найденным адресом занята некоторой информацией (т.е. произошла коллизия), то i увеличивается на 1 и адрес подсчитывается заново.

***Циклический однонаправленный список.*** Однонаправленный список считается циклическим тогда и только тогда, когда последний элемент списка ссылается на начальный, тем самым замыкая список в кольцо.

Элемент списка имеет следующую структуру:

struct refferal

{

string key\_patient;// ключ формата: MM-NNNNNN

char namedoctor[SIZE\_NAME]; // ФИО врача

SYSTEMTIME time; // время и дата приема

refferal \*next; // указатель на следующий элемент списка

};

Для корректной и быстрой работы над списком отдельно вводится указатель на последний элемент списка. Каждый новый элемент помещается в конец списка, причём указатель на последний элемент будет ссылаться уже на новый, а сам новый элемент будет ссылаться на начальный элемент. После добавления элемента в список выполняется сортировка списка методом извлечений.

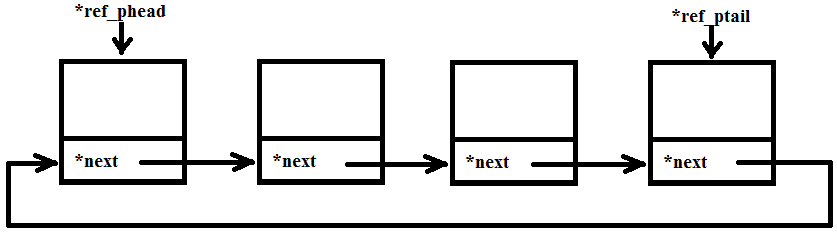


Рис. 2.1. Схема организации списка.

***Метод сортировки извлечением.*** Упорядочивание элементов циклического однонаправленного списка ведётся путём поиска максимального значения в некотором множестве элементов и помещения последнего элемента множества на место найденного максимума, а сам максимум помещается в конец границы множества. Причём граница множества после каждого прохода уменьшается.

Пример сортировки: «5» – текущий максимальный элемент, «\_» - граница множества.

2 2 2 2 1

5 3 2 1 2

1 1 1 2 2

2 2 3 3 3

3 5 5 5 5

***АВЛ-дерево. Симметричный обход АВЛ-дерева.*** Симметричный обход заключается в том, что сначала рассматривают левое поддерево, затем корень, а после уже все деревья слева направо.

Дерево сбалансировано по высоте, т.е. в дереве количество вершин в левом и в правом поддереве отличаются на единицу. У каждого узла имеется высота поддеревьев. Если наблюдается разница высот более чем на 1, то в зависимости от величины будет совершено вращение (правое/левое) вершин и поддеревьев. Если разница по модулю больше 1, но меньше нуля, то совершается правое вращение, иначе левое.

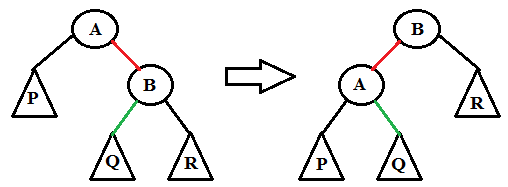


Рис. 2.2. Пример правого вращения.

Узел дерева имеет следующую структуру:

// структура данных врача

struct doctor

{

char name[NAME\_DOC\_SIZE]; // фамилия и инициалы

string noun; // должность

unsigned int cabinet; // номер кабинета

date reception; // график работы

};

// узел дерева

struct node

{

doctor key; // ключ узла дерева

int height; // высота поддерева с корнем в данном узле

node\* left; // указатель на левое поддерево

node\* right;// указатель на правое поддерево

};

Алгоритм рекурсивен, т.к. сначала спускаются по дереву влево до тех пор, пока не дойдём до конца левого поддерева. Как только дошли, то смотрим на корень, а затем на ближайшее правое поддерево. Если правое поддерево имеет ещё ответвления, то алгоритм повторяется.

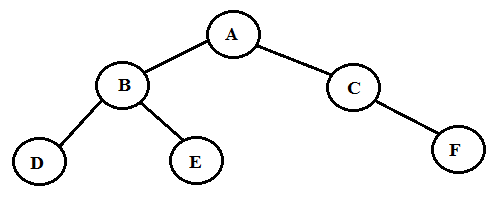


Рис. 2.3. Пример дерева.

Результат симметричного обхода дерева: D B E A C F

***Прямой алгоритм поиска слова в тексте.*** Слово посимвольно сравнивается с текстом. Если в тексте наблюдается вхождение данного слова, то поиск был произведён успешно, иначе поиск был произведён не успешно.

При посимвольном сравнением символы как слова, так и текста становятся прописными (заглавными). Т.е. выполняется процесс преобразования символов: все строчные символы становятся прописными. Это сделано ради того, чтобы исключить регистрозависимость.

Текст: Кукаяка А.П.

Слово: кука

Результат: слово обнаруженно.

# Описание программы

Программа содержит основное меню навигации, которое предоставляет доступ к 3 ключевым подменю:

* Работа с клиентурой (пациентами)
* Работа с направлениями
* Работа с докторами

Каждое подменю содержит в себе поля:

* Добавления нового элемента
* Удаление определённого элемента
* Вывод таблицы со всеми элементами
* Удаление всех данных (освобождения выделенной памяти)
* Поиск элемента по критериям (кроме подменю «Работа с направлениями»)

Хранение данных представлено в трёх файлах:

* Пациенты - patients.dat;
* Доктора - doctors.dat;
* Данные о выдаче направлений на приём - refferals.dat.

При регистрации нового пациента ФИО вводить полностью с большой буквы (к примеру, Иванов Иван Иванович). При заполнении даты рождения, вводить в поле «месяц» номер искомого месяца. В поле «место работы» следует указать «отсутствует», если пациент не учится в учебных заведениях и не работает. Номер участка имеет формат ММ, где М – число натурального ряда.

При удалении одного пациента следует ввести регистрационный номер формата «MM-NNNNNN», где MM – номер участка, NNNNNN – порядковый номер. Если введённый ключ не существует, то программа запросит разрешение на вывод таблицы с полной информацией о пациентах, после запроса программа вновь попросит ввести регистрационный номер.

Поиск пациента разбит на следующие категории:

* Поиск по регистрационному номеру. При поиске пациента по регистрационному номеру требуется ввести сам номер в формате «MM-NNNNNN», где MM – номер участка, NNNNNN – порядковый номер. Результат поиска выводится на экран в виде таблицы, в которой данные о пациенте, а также выводится список ФИО назначенных пациенту врачей.
* Поиск по ФИО. При поиске пациента по ФИО требуется ввести ФИО пациента. В результате на экран выводится таблица со сведениями о пациенте.

При добавлении нового направления следует ввести регистрационный номер формата «MM-NNNNNN», где MM – номер участка, NNNNNN – порядковый номер. Если введённый ключ не существует, то программа запросит разрешение на вывод таблицы с полной информацией о пациентах, после запроса программа вновь попросит ввести регистрационный номер. Далее будет запрос на подтверждение. Если ответ на запрос отрицательный, то программа потребует снова ввести регистрационный номер. После это программа потребует ввести ФИО врача, стоит отметить, что полноценного ввода ФИО не потребуется (Прямой поиск слова в тексте). После ввода будет организован поиск врачей со схожим ФИО. Результат будет содержать ФИО врача и его должность, вывод в форме списка, а также запрос на выбор нужного врача из списка. Затем, после выбора врача и после выбора ответа на запрос на подтверждение, будет выведен график работы специалиста, а также ближайшая дата, когда врач может принять. Стоит отметить, что будет запрос на назначение ближайшей даты приёма. Если ответ отрицательный, то программа попросить ввести дату и время приёма, иначе успех – направление добавлено (Ввод данных контролируется, если дата и время не удовлетворяют графику работы врача, то система запросит ввести другие данные, сопровождая это сообщением об ошибке ввода).

Возврат направления начинается с подменю поиска:

* Поиск врача по ФИО. При поиске врача по ФИО требуется ввести ФИО.

Прямой поиск слова в тексте.

* Поиск пациента по регистрационному номеру формата «MM-NNNNNN», где MM – номер участка, NNNNNN – порядковый номер.

Результат поиска единый: список направлений выданные на искомого пациента или назначенные искомому врачу. Затем система осуществляет запрос на выбор направления, которое будет удалено, после выбора ответа на запрос на подтверждение и успешного выбора программа удалит запись и сообщит об этом. В противном случае сообщение о том, что запись не была удалена.

При добавлении нового врача ФИО вводить не полностью, только фамилию и инициалы (к примеру, Иванов И.И.). При заполнении графика работы врача система вначале запросит ввести количество рабочих дней (0-6), затем на экране выведется список дней недели и система попросит выбрать рабочие дни врача. Далее требуется ввести время приёма врача, стоит отметить, что время работы врача не должно превышать 8 часов.

При удалении одного пациента следует ввести регистрационный ФИО врача, полностью не обязательно (Прямой поиск слова в тексте). После ввода будет организован поиск врачей со схожим ФИО. Результат будет содержать ФИО врача и его должность, вывод в форме списка, а также запрос на выбор нужного врача из списка. Затем, после выбора врача и после выбора ответа на запрос на подтверждение. Если ответ положительный – запись удаляется, иначе сообщение о том, что запись не удалена.

Поиск врача разбит на следующие категории:

* Поиск по ФИО. При поиске врача по ФИО требуется ввести ФИО. Результат поиска – информация о враче и список его пациентов.
* Поиск по должности. При поиске врача по должности вводится нужная должность. В результате на экран выводится таблица с информацией о найденном враче.

При завершении работы все данные сохраняются и выгружаются в соответствующие файлы (Также можно принудительно заставить сохранить информацию через меню). На протяжении исполнения программы выполняется тщательная проверка полей на ввод данных, а именно: на ФИО, контроль ввода строго по заданной форме (ввод регистрационного номера), ввод численных данных, ввод даты и времени. Контроль ведётся с целью избежать аварийного завершения программы, а также для соблюдения достоверности введённой информации.

# Тестирование программы

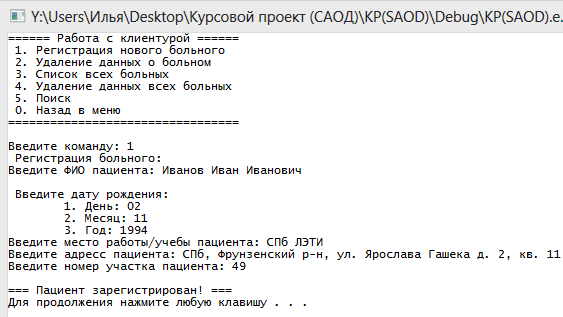
***Работа с пациентами.***

* Регистрация нового пациента

Входные данные:

1. ФИО: Иванов Иван Иванович
2. Дата рождения: 02.11.1994
3. Место учёбы: СПб ЛЭТИ
4. Адрес: СПб, Фрунзенский р-н, ул. Ярослава Гашека д. 2, кв. 11
5. Номер участка: 49

Результат:

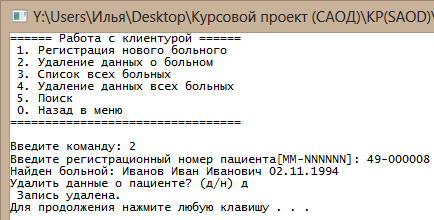


* Удаление данных о больном

Входные данные:

1. Регистрационный номер: 49-000008

Результат:

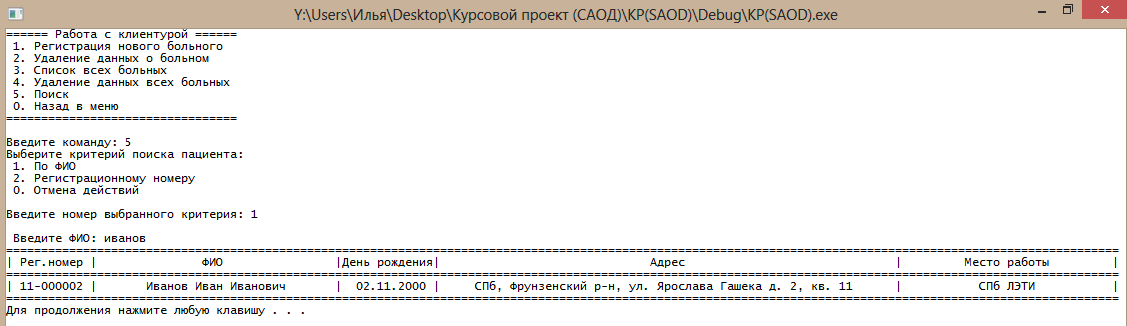


* Поиск по ФИО

Входные данные:

1. ФИО: иванов

Результат:

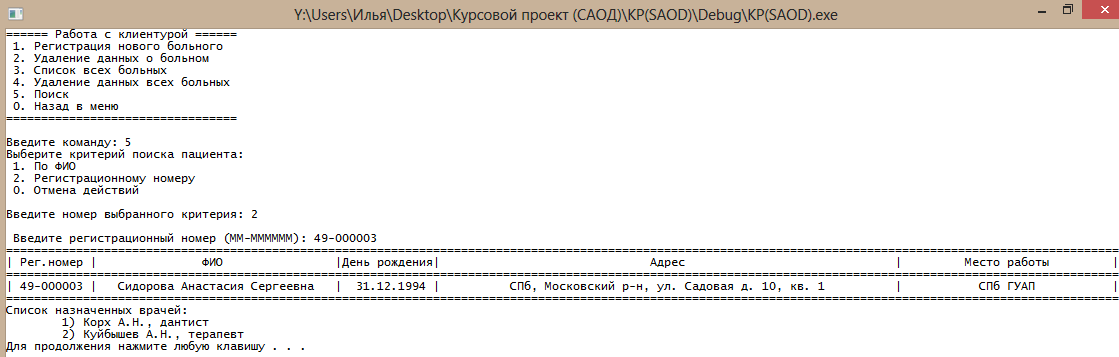


* Поиск по регистрационному номеру

Входные данные:

1. Регистрационный номер: 49-000003

Результат:



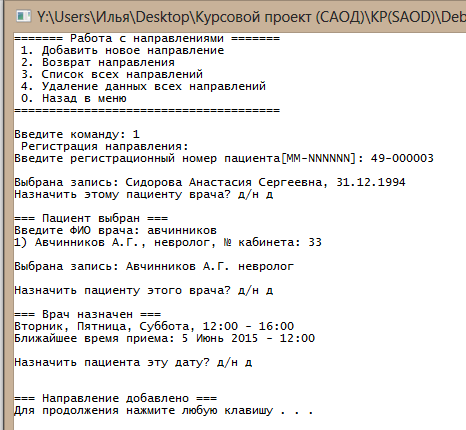
***Работа с направлениями.***

* Добавление нового направления

Входные данные:

1. Регистрационный номер пациента: 49-000003
2. ФИО врача: авчинников
3. Дата: назначить текущую дату

Результат:

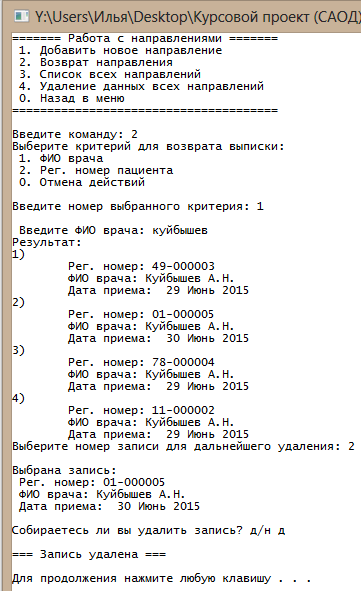


* Возврат направления по ФИО врача

Входные данные:

1. ФИО врача: куйбышев

Результат:

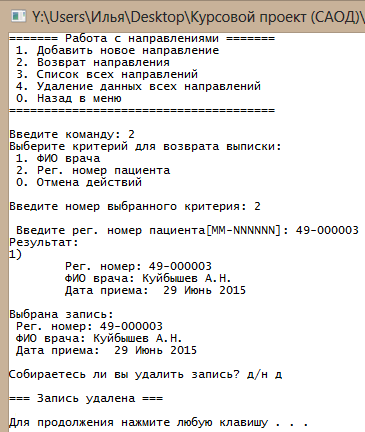


* Возврат направления по регистрационному номеру пациента

Входные данные:

1. Регистрационный номер: 49-000003

Результат:



***Работа с докторами.***

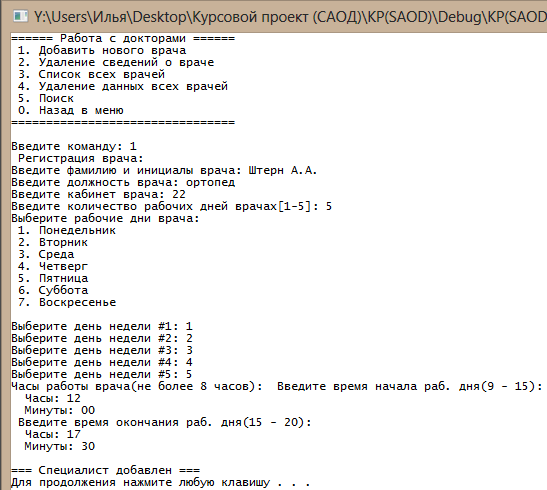
* Добавление нового врача

Входные данные:

1. Фамилия и инициалы: Штерн А.А.
2. Должность: ортопед
3. Кабинет: 22
4. График работы: 5 дней в неделю, с понедельника по пятницу

С 12:00 – 17:30

Результат:

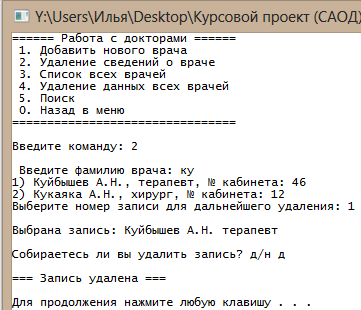


* Удаление сведений о враче

Входные данные:

1. ФИО врача: ку
2. Выбрать для удаления Куйбышева.

Результат:

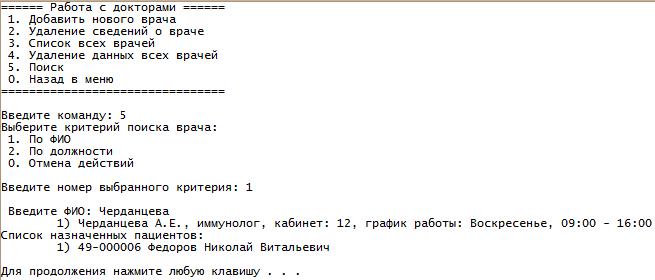


* Поиск врача по ФИО

Входные данные:

1. ФИО: францева

Результат:

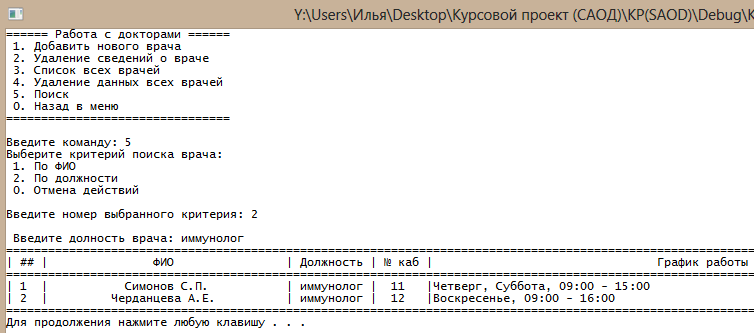


* Поиск врача по должности

Входные данные:

1. Должность: иммунолог

Результат:



# Заключение

Система выполняет функции учёта всех имеющихся клиентов поликлиники, а также медицинского персонала и направлений на приём к специалисту. Для работы использовались хеш-таблица, в которой хранились данные о пациентах, АВЛ-дерево, в котором содержалась информация о врачах, циклический однонаправленный список, в котором располагались сведения о выданных пациенту направления на приём к врачу. Из алгоритмов использовались сортировка методом извлечения, благодаря которому упорядочивались элементы списка по ФИО врачей, алгоритм симметричного обхода дерева, который позволял быстро и легко просматривать всю информацию и находить нужную, а также прямой алгоритм поиска слова в тексте, который используется во всех функциях поиска данных в структурах.

Использование заданных структур и алгоритмов позволило ускорить получение нужной информации, а также ускорило процесс регистрации и учёта информации.

Система «База данных поликлиники» уменьшит нагрузку на регистратуру, и кроме того ускорит её работу, что не мало важно для поликлиники в настоящее время.

Программа выполняет поставленную задачу и работает без ошибок (для корректных тестовых данных).