|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования |
| **«МИРЭА – Российский технологический университет»** |
| **РТУ МИРЭА** |
|  |

|  |  |
| --- | --- |
| **Отчет по выполнению практического задания № 6** | |
| **Тема:** | |
| **«Двунаправленные динамические списки»** | |
| Дисциплина: «Структуры и алгоритмы обработки данных» | |
|  | Выполнил студент: Зернов Н.А. |
|  | Группа: ИКБО-74-23 |

Москва – 2024

СОДЕРЖАНИЕ

[1 ЦЕЛЬ 3](#_gjdgxs)

[2 ЗАДАНИЕ 4](#_30j0zll)

[2.1 Формулировка задачи (В списке №13, Вариант 13) 4](#_1fob9te)

[2.2 Определение списка и описание операций над списком 5](#_3znysh7)

[2.2.1 Определение структуры узла двунаправленного списка 5](#_2et92p0)

[2.2.2 Процесс выполнения операций 7](#_tyjcwt)

[2.2.2.1 Вставка узла. 7](#_rjgck4c3lvht)

[2.2.2.2 Удаление узла. 10](#_iey0ptwr19lr)

[2.2.2.3 Вывод элементов списка на экран. 14](#_8z2ypeeoe7oc)

[2.2.2.4 Поиск узла с заданным значением. 16](#_65gkb5tnjygt)

[2.2.2.5 Вставить в начало списка заболеваний пациента новый узел. 17](#_4wy3p1rbi9su)

[2.2.2.6 Сформировать новый список, включив в него информацию о пациентах, у которых зарегистрировано более 5 заболеваний. 22](#_9ix0efjfz3x4)

[2.2.2.7 Удалить из списка кодов заболеваний указанного пациента, узел с указанным кодом. 22](#_jx91y1l5b9p0)

[2.3 Реализация алгоритма на языке C++ и проведение тестирования 27](#_3dy6vkm)

[2.3.1 Реализация на языке программирования C++ 27](#_1t3h5sf)

[2.3.2 Тестирование 37](#_4d34og8)

[2.4 Вывод по заданию 40](#_2s8eyo1)

[3 ВЫВОДЫ 41](#_lnxbz9)

[4 ЛИТЕРАТУРА 42](#_35nkun2)

# **1 ЦЕЛЬ**

Получение знаний и практических навыков управления двунаправленным списком в программах на языке С++.

# **2 ЗАДАНИЕ**

## **2.1 Формулировка задачи (В списке №13, Вариант 13)**

Разработать многомодульную программу, которая демонстрирует выполнение всех операций, определенных вариантом, над линейным двунаправленным динамическим списком.

Требования к разработке.

1. Разработать структуру узла списка, структура информационной части узла определена вариантом. Для определения структуры узла списка, используйте тип struct или class. Сохраните определение структуры узла и прототипы функций в заголовочном файле.

2. Разработайте функции для выполнения операции над линейным двунаправленным динамическим списком:

• создание списка;

• вставку узла;

• удаление узла;

• вывод списка в двух направлениях (слева направо и справа налево);

• поиск узла с заданным значением (операция должна возвращать указатель на узел с заданным значением).

3. Дополнительные операции над списком, указанные вариантом, оформите в виде функций и включите в отдельный файл с расширением cpp. Подключите к этому файлу заголовочный файл с определением структуры узла.

4. Разработайте программу, управляемую текстовым меню, и включите в меню демонстрацию выполнения всех операций задания и варианта.

5. Проведите тестирование операций.

• Оцените сложность алгоритма первой дополнительной операции.

6. Оформите отчет по разработке программы в соответствии с требованиями задания по однонаправленному списку.

Примечание: в определении информационной части узла варианта, подчеркнутое поле считать полем ключа.

Индивидуальный вариант. Тип информационной части узла: Марка автомобиля, страна изготовитель, год выпуска, цена. Дата продажи (заполняется не сразу).

Дополнительные операции:

Вставить в начало списка заболеваний пациента новый узел.

Сформировать новый список, включив в него информацию о пациентах, у которых зарегистрировано более 5 заболеваний.

Удалить из списка кодов заболеваний указанного пациента, узел с указанным кодом.

## **2.2 Определение списка и описание операций над списком**

### **2.2.1 Определение структуры узла двунаправленного списка**

Двунаправленный список (или двусвязный список) - это структура данных в информатике, которая состоит из узлов, каждый из которых содержит две ссылки: одну на предыдущий узел в списке и одну на следующий узел. Это позволяет эффективно перемещаться как вперед, так и назад по списку. В отличие от однонаправленного списка, где можно перемещаться только в одном направлении, двунаправленный список обеспечивает быстрый доступ как к началу, так и к концу списка, а также возможность вставки и удаления элементов как в начале, так и в конце списка за постоянное время, что делает его полезным для множества задач, где требуется такая гибкость.

Определим структуру узла двунаправленного списка согласно варианту.

Структура Patient\_Node представляет собой элемент двунаправленного списка мед. полюсов. В данной структуре содержатся следующие поля: переменная policy\_number типа string, которая номер медицинского полиса, переменная surname типа string, которая хранит фамилию пациента, указатель diseases на линейный список заболеваний, указатель prev типа Patient\_Node, который указывает на предыдущий элемент в списке, указатель next типа Patient\_Node, который указывает на следующий элемент в списке. Конструктор Patient\_Node принимает значения всех полей структуры и инициализирует соответствующие переменные. Указатели prev и next при создании элемента списка присваиваются значению nullptr.

Данное описание структуры представлено в виде кода на C++ в блоке кода 1, а его изображение на рисунке 1.

**struct** Disease\_Node {

    string disease\_code; // Код заболевания

    string registration\_date; // Дата регистрации заболевания

    Disease\_Node**\*** next;

};

// Структура узла списка пациентов

**struct** Patient\_Node {

    string policy\_number; // Номер мед. полиса

    string surname; // Фамилия больного

    Disease\_Node**\*** diseases; // Указатель на линейный список заболеваний

    Patient\_Node**\*** prev;

    Patient\_Node**\*** next;

};

Блок кода 1 - Структура

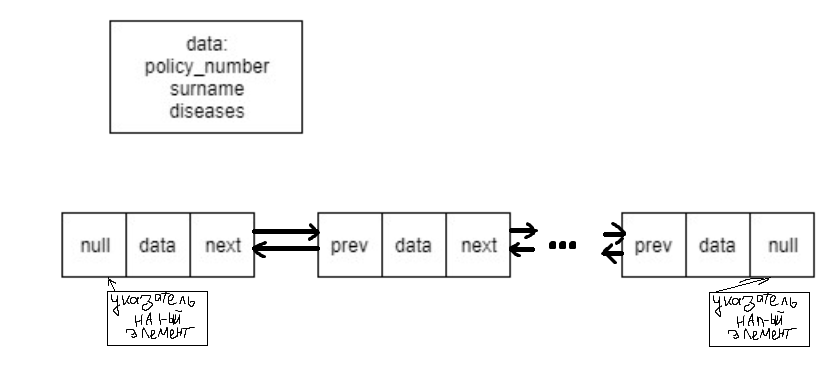


Рисунок 1 - Изображение структуры данных

### **2.2.2 Процесс выполнения операций**

В работе используются операции: вставка узла, удаление узла, вывод списка в двух направлениях (слева направо и справа налево), поиск узла с заданным значением (операция должна возвращать указатель на узел с заданным значением), формирование нового списка болезней, удалить все из списка кодов заболеваний указанного пациента, узел с указанным кодом.

Изобразим, рассмотрим алгоритм, реализуем и предоставим таблицу с данными для тестирования данных операций.

#### 2.2.2.1 Вставка узла.

Отобразим выполнение данной операции на рисунке 2.

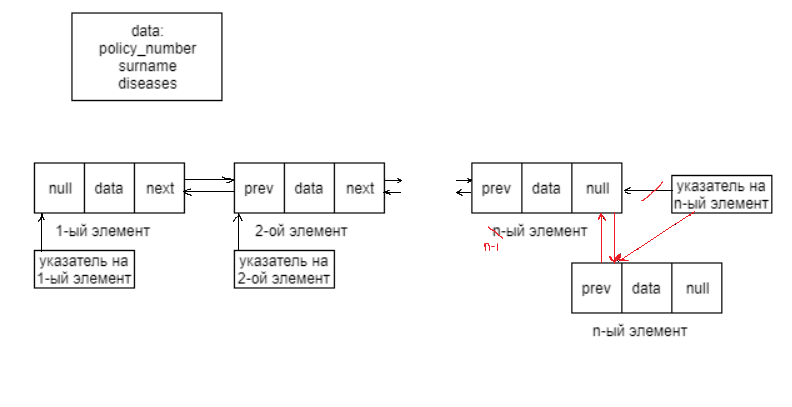


Рисунок 2 - Изображение добавление нового узла

Реализуем вставку метод insert\_node предназначен для вставки нового узла в конец списка. Он создает новый узел и инициализирует его поля значениями номер мед. полиса фамилия, код заболевая и дата ее регистрации, если человек болен. Поле со списком diseases устанавливается nullptr. Затем метод проверяет, пуст ли список. Если да, то новый узел становится как головным, так и хвостовым элементом списка. В противном случае новый узел присоединяется к концу списка, а указатель tail обновляется на новый узел.

Реализация данного алгоритма представлена в блок коде 2. Данные для тестирования будут приведены в таблице 1.

**void** PatientList::insert\_node(string policy\_number, string surname) {

    Patient\_Node**\*** new\_node **=** **new** Patient\_Node;

    new\_node->policy\_number **=** policy\_number;

    new\_node->surname **=** surname;

    new\_node->diseases **=** **nullptr**; // Пациент пока не болен

    new\_node->next **=** **nullptr**;

    new\_node->prev **=** tail;

**if** (tail **!=** **nullptr**)

        tail->next **=** new\_node;

    tail **=** new\_node;

**if** (head **==** **nullptr**)

        head **=** new\_node;

}

Блок кода 2 - Добавление нового узла в конце списка

Таблица 1 - Данные для тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Добавить | Ожидаемый результат |
| 1 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024 | pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024 |
| 2 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024 | pol7: Орлов | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов |

#### 2.2.2.2 Удаление узла.

Отобразим выполнение данной операции на рисунке 3.

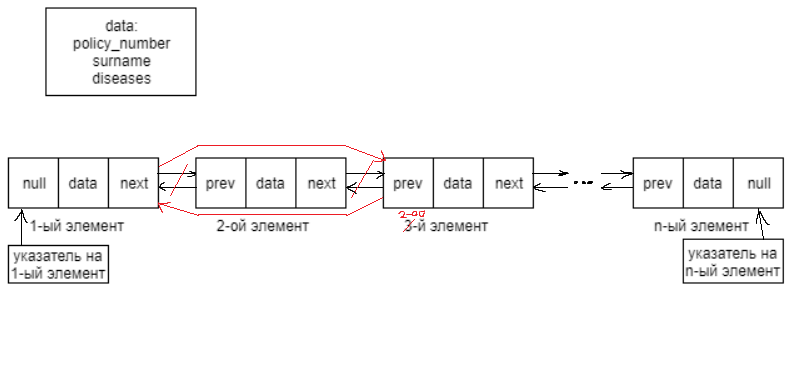


Рисунок 3 - Изображение удаления узла

Сначала метод проверяет, не является ли указатель node равным nullptr. Если это так, то ничего не делается, и метод завершается.

Затем метод проверяет, имеет ли удаляемый узел node предыдущий узел prev. Если у него есть предыдущий узел, то указатель next предыдущего узла обновляется так, чтобы он указывал на узел, следующий за удаляемым узлом. Если удаляемый узел был головным, то обновляется указатель head, чтобы он указывал на следующий узел после удаляемого. Затем метод проверяет, имеет ли удаляемый узел следующий узел. Если у него есть следующий узел, то указатель prev следующего узла обновляется так, чтобы он указывал на узел, предшествующий удаляемому узлу. Если удаляемый узел был хвостовым, то обновляется указатель tail, чтобы он указывал на предыдущий узел перед удаляемым. Удаляемый узел освобождается с помощью оператора delete.

Реализация данного алгоритма представлена в блок коде 3. Данные для тестирования будут приведены в таблице 2.

**void** PatientList::delete\_node(Patient\_Node**\*** node) {

**if** (node->prev **!=** **nullptr**)

        node->prev->next **=** node->next;

**else**

        head **=** node->next;

**if** (node->next **!=** **nullptr**)

        node->next->prev **=** node->prev;

**else**

        tail **=** node->prev;

**delete** node;

}

Блок кода 3 - Удаление узла по ключу

Таблица 2 - Данные для тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Удалить | Ожидаемый результат |
| 1 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов | pol1 | pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов |
| 2 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов | pol2 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов |

#### 2.2.2.3 Вывод элементов списка на экран.

Вывод элементов списка на экран может быть осуществлен в двух направлениях (вперед и назад). Рассмотрим оба направления.

Вывод элементов списка слева направо. Начинаем с головы списка и перебираем каждый элемент, выводя поля: мед. полис, фамилию, информацию о заболеваниях, если они есть. Затем переходим к следующему элементу, используя указатель next, и повторяем процесс до тех пор, пока не достигнем конца списка.

Выводим элементы списка справа налево. Начинаем с хвоста списка и перебираем каждый элемент, выводя поля: мед. полис, фамилию, информацию о заболеваниях, если они есть. Затем переходим к предыдущему элементу, используя указатель prev, и повторяем процесс до тех пор, пока не достигнем начала списка. Реализация данного алгоритма представлена в блок коде 4. Данные для тестирования слева направо будут приведены в таблице 3, а для справа налево в таблице 4.

**void** PatientList::print\_list\_forward() **const** {

**for** (Patient\_Node**\*** node **=** head; node **!=** **nullptr**; node **=** node->next) {

        cout **<<** node->policy\_number **<<** ": " **<<** node->surname **<<** endl;

**for** (Disease\_Node**\*** disease **=** node->diseases; disease **!=** **nullptr**; disease **=** disease->next)

            cout **<<** "Заболевание: " **<<** disease->disease\_code **<<** ", Дата регистрации: " **<<** disease->registration\_date **<<** endl;

    }

}

Блок кода 4 - Вывод элементов на экран

Таблица 3 - Данные для тестирования

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Ожидаемый результат |
| 1 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов |
| 2 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020 |

**void** PatientList::print\_list\_backward() **const** {

**for** (Patient\_Node**\*** node **=** tail; node **!=** **nullptr**; node **=** node->prev) {

        cout **<<** node->policy\_number **<<** ": " **<<** node->surname **<<** endl;

**for** (Disease\_Node**\*** disease **=** node->diseases; disease **!=** **nullptr**; disease **=** disease->next)

            cout **<<** "Заболевание: " **<<** disease->disease\_code **<<** ", Дата регистрации: " **<<** disease->registration\_date **<<** endl;

    }

}

Блок кода 4 - Вывод элементов на экран

Таблица 4 - Данные для тестирования

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Ожидаемый результат |
| 1 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020 | pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023 |

#### 2.2.2.4 Поиск узла с заданным значением.

Отобразим выполнение данной операции на рисунке 4.

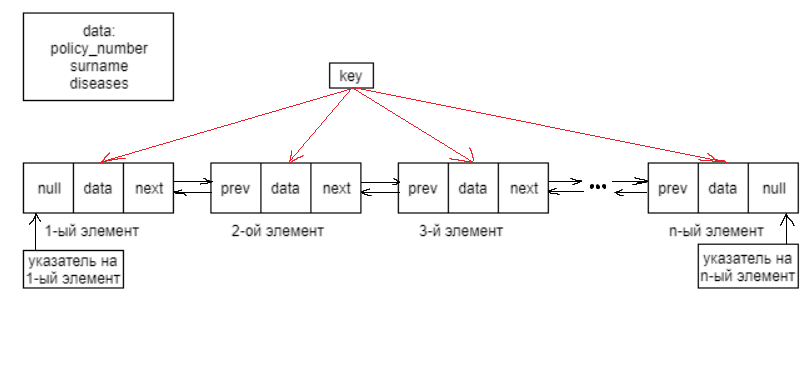


Рисунок 4 - Поиск узла с заданным значением

Метод начинает поиск с головного узла списка, инициализируя указатель current значением указателя head. Затем метод перебирает каждый узел списка, проверяя совпадение мед. полиса текущего узла с заданным мед. полисом policy\_number. Это осуществляется с помощью условного оператора if. Если мед. полис текущего узла соответствует заданной мед. полисом, метод возвращает указатель на текущий узел. Если узел с заданным мед. полисом не найден во всем списке, метод завершается, возвращая nullptr. Этот алгоритм проходит по списку, находит самую длинную последовательность одинаковых символов и выводит информацию о ней.

Реализация данного алгоритма представлена в блок коде 5. Данные для тестирования будут приведены в таблице 6.

Patient\_Node**\*** PatientList::search\_node(string policy\_number) **const** {

**for** (Patient\_Node**\*** node **=** head; node **!=** **nullptr**; node **=** node->next)

**if** (node->policy\_number **==** policy\_number) {

**return** node;

    }

**return** **nullptr**;

}

Блок кода 5 - Поиск узла с заданным значением

#### 2.2.2.5 Вставить в начало списка заболеваний пациента новый узел.

Метод insert\_front\_disease предназначен для вставки нового заболевания в начало списка заболеваний diseases указанного пациента patient. Вот как этот метод работает:

1. Проверка наличия пациента. Сначала метод проверяет, существует ли указанный пациент (patient). Если пациент не существует то есть указатель patient равен nullptr, то операция вставки не выполняется, и выводится сообщение о том, что пациент не найден.
2. Создание нового узла заболевания. Затем метод создает новый узел типа Disease\_Node, который будет содержать информацию о новом заболевании. В этом узле сохраняются переданные код заболевания disease\_code и дата регистрации registration\_date.
3. Вставка узла заболевания в начало списка заболеваний пациента. Новый узел заболевания затем вставляется в начало списка заболеваний пациента. Для этого указатель next нового узла устанавливается на текущий первый узел в списке заболеваний пациента. После этого указатель diseases пациента указывает на новый узел заболевания, делая его новым первым узлом в списке заболеваний.

Таким образом, данный метод обеспечивает эффективное добавление новых заболеваний в начало списка заболеваний для указанного пациента.

Cложность алгоритма метода insert\_front\_disease:

1. Создание нового узла заболевания: Эта операция выполняется за константное время O(1), так как просто выделяется память и инициализируются поля нового узла.
2. Вставка узла заболевания в начало списка заболеваний пациента:
   1. Установка указателя next нового узла на текущий первый узел в списке: Операция также выполняется за константное время O(1).
   2. Установка указателя diseases пациента на новый узел: Это также константная операция O(1), так как просто обновляется указатель.
3. Общая сложность операции вставки узла заболевания в начало списка заболеваний пациента составляет O(1), поскольку все шаги выполняются за константное время независимо от размера списка заболеваний или числа пациентов.
4. Следовательно, общая сложность алгоритма метода insert\_front\_disease равна O(1).

**void** PatientList::insert\_front\_disease(Patient\_Node**\*** patient, string disease\_code, string registration\_date) {

    Disease\_Node**\*** new\_disease **=** **new** Disease\_Node;

    new\_disease->disease\_code **=** disease\_code;

    new\_disease->registration\_date **=** registration\_date;

    new\_disease->next **=** patient->diseases;

    patient->diseases **=** new\_disease;

}

Блок кода 6 - Вставить в начало списка заболеваний пациента новый узел.

Таблица 5 - Данные для тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Вставить | Ожидаемый результат |
| 1 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов | pol7  CVE-14214234  07.05.2024 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов  Заболевание: CVE-14214234, Дата регистрации: 07.05.2024 |
| 2 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов  Заболевание: CVE-14214234, Дата регистрации: 07.05.2024  pol1: Иванов  pol2: Смирнов  pol3: Кузнецов  pol4: Васильев  pol5: Петров  pol6: Соколов  pol7: Орлов | pol1  ABC-1223  01.01.2024 | Введите номер мед. полиса: pol1  Введите код заболевания: ABC-1223  Введите дату регистрации заболевания: 01.01.2024  pol1: Иванов  Заболевание: ABC-1223, Дата регистрации: 01.01.2024  Заболевание: A28, Дата регистрации: 11.11.2023  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов  Заболевание: CVE-14214234, Дата регистрации: 07.05.2024  pol1: Иванов  pol2: Смирнов  pol3: Кузнецов  pol4: Васильев  pol5: Петров  pol6: Соколов  pol7: Орлов |

#### 2.2.2.6 Сформировать новый список, включив в него информацию о пациентах, у которых зарегистрировано более 5 заболеваний.

Метод new\_list\_of\_mediccal\_card выполняет следующий функционалю. Метод создает новый объект PatientList, который будет использоваться для хранения информации о пациентах, у которых зарегистрировано более 5 заболеваний. Метод проходит по исходному списку пациентов, начиная с головы списка (head). Это позволяет ему проверить каждого пациента в списке. Для каждого пациента в списке метод подсчитывает количество заболеваний, пройдя по списку его заболеваний. Это делается с помощью внутреннего цикла, который проходит по списку заболеваний каждого пациента и увеличивает счетчик. Если у пациента более 5 заболеваний (согласно подсчету, выполненному в предыдущем шаге), его информация (номер медицинского полиса и фамилия) добавляется в новый список пациентов. Для этого используется метод insert\_node нового списка. По завершении прохода по исходному списку пациентов метод возвращает указатель на новый список, содержащий информацию только о пациентах с более чем 5 заболеваниями.

Этот метод позволяет выделить пациентов с определенным количеством заболеваний и создать новый список только с такими пациентами, что может быть полезно для дальнейшего анализа или обработки этих данных.

Реализация данного алгоритма представлена в блок коде 7.

PatientList**\*** PatientList::new\_list\_of\_mediccal\_card() {

    PatientList**\*** newList **=** **new** PatientList();

**for** (Patient\_Node**\*** node **=** head; node **!=** **nullptr**; node **=** node->next) {

**int** count **=** 0;

**for** (Disease\_Node**\*** disease **=** node->diseases; disease **!=** **nullptr**; disease **=** disease->next)

**++**count;

**if** (count **>** 5)

            newList->insert\_node(node->policy\_number, node->surname);

    }

**return** newList;

}

Блок кода 7 - Формирование нового списка, включив в него информацию о пациентах, у которых зарегистрировано более 5 заболеваний

#### 2.2.2.7 Удалить из списка кодов заболеваний указанного пациента, узел с указанным кодом.

Метод delete\_disease просматривает список заболеваний пациента patient->diseases и ищет узел заболевания с указанным кодом disease\_code. Он выполняет это, пройдя по списку заболеваний пациента в цикле. Удаление найденного заболевания, если заболевание с заданным кодом найдено, метод удаляет его из списка заболеваний пациента. Для этого выполняются следующие действияю, если заболевание находится в начале списка, то обновляется указатель на начало списка заболеваний patient->diseases. Eсли заболевание находится в середине или конце списка, то перенаправляются указатели, чтобы исключить удаляемый узел из списка. После удаления узла заболевания из списка освобождается память, занимаемая этим узлом, с помощью оператора delete.

Этот метод позволяет удалить заболевание с заданным кодом из списка заболеваний конкретного пациента.

Реализация данного алгоритма представлена в блок коде 8. Данные для тестирования будут приведены в таблице 6.

**void** PatientList::delete\_disease(Patient\_Node**\*** patient, string disease\_code) {

    Disease\_Node**\*** temp **=** patient->diseases;

    Disease\_Node**\*** prev **=** **nullptr**;

**if** (temp **!=** **nullptr** **&&** temp->disease\_code **==** disease\_code) {

        patient->diseases **=** temp->next;

**delete** temp;

**return**;

    }

**while** (temp **!=** **nullptr** **&&** temp->disease\_code **!=** disease\_code) {

        prev **=** temp;

        temp **=** temp->next;

    }

**if** (temp **==** **nullptr**)

**return**;

    prev->next **=** temp->next;

**delete** temp;

}

Блок кода 8 - Удалить из списка кодов заболеваний указанного пациента, узел с указанным кодом

Таблица 6 - Данные для тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Удалить | Ожидаемый результат |
| 1 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов | pol6  B26.0+ | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  pol7: Орлов |
| 2 | pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  Заболевание: A28, Дата регистрации: 11.11.2023  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  Заболевание: B05.3+, Дата регистрации: 01.03.2020  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  Заболевание: B82, Дата регистрации: 31.01.2019  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов  pol1: Иванов  pol2: Смирнов  pol3: Кузнецов  pol4: Васильев  pol5: Петров  pol6: Соколов  pol7: Орлов  pol1: Иванов  pol2: Смирнов  pol3: Кузнецов  pol4: Васильев  pol5: Петров  pol6: Соколов  pol7: Орлов | pol110101010  AC1 | Пациент не найден.  pol1: Иванов  Заболевание: A28, Дата регистрации: 11.11.2023  Заболевание: A28, Дата регистрации: 11.11.2023  Заболевание: A28, Дата регистрации: 11.11.2023  pol2: Смирнов  Заболевание: B05.3+, Дата регистрации: 01.03.2020  Заболевание: B05.3+, Дата регистрации: 01.03.2020  Заболевание: B05.3+, Дата регистрации: 01.03.2020  pol3: Кузнецов  Заболевание: B82, Дата регистрации: 31.01.2019  Заболевание: B82, Дата регистрации: 31.01.2019  Заболевание: B82, Дата регистрации: 31.01.2019  pol4: Васильев  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol5: Петров  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  Заболевание: A4, Дата регистрации: 07.02.2024  Заболевание: A4, Дата регистрации: 06.02.2024  Заболевание: A3, Дата регистрации: 05.02.2024  Заболевание: A2, Дата регистрации: 04.02.2024  Заболевание: A2, Дата регистрации: 03.02.2024  Заболевание: A1, Дата регистрации: 02.02.2024  Заболевание: A0, Дата регистрации: 01.02.2024  pol6: Соколов  Заболевание: B26.0+, Дата регистрации: 01.01.2024  Заболевание: B26.0+, Дата регистрации: 01.01.2024  pol7: Орлов  pol1: Иванов  pol2: Смирнов  pol3: Кузнецов  pol4: Васильев  pol5: Петров  pol6: Соколов  pol7: Орлов  pol1: Иванов  pol2: Смирнов  pol3: Кузнецов  pol4: Васильев  pol5: Петров  pol6: Соколов  pol7: Орлов |

## **2.3 Реализация алгоритма на языке C++ и проведение тестирования**

### **2.3.1 Реализация на языке программирования C++**

Мы реализуем алгоритм на языке C++, используя библиотеки: iostream – это стандартная библиотека C++, предоставляющая классы и функции для ввода и вывода данных через стандартные потоки cout и cin. cout используется для вывода информации на консоль, а cin для ввода данных с консоли, и string - это класс в C++, предоставляемый в заголовочном файле string.h, который обеспечивает удобное управление строками текста. Он позволяет создавать, хранить и манипулировать строками символов.

В нашей программе мы также будем использовать классы и объекты. Классы в C++ представляют собой абстракцию, описывающую методы и свойства объектов. Объекты, созданные на основе классов, называются экземплярами класса.

Структура данных, которую мы будем использовать, - это двусвязный список медицинских полисов, представленный в заголовочном файле medical\_policy.h. Он также содержит класс PatientList, который обеспечивает интерфейс для работы с этим списком.

Файл medical\_policy.cpp содержит определения методов класса PatientList, которые были описаны в заголовочном файле medical\_policy.h.

Файл main.cpp является точкой входа в программу. В функции main содержится код, который будет запускаться при старте программы. Здесь мы будем использовать класс PatientList для демонстрации его функциональности.

**#ifndef** MEDICAL\_POLICY\_H

**#define** MEDICAL\_POLICY\_H

**#include** <string>

**#include** <iostream>

**using** **namespace** std;

// Структура узла списка заболеваний

**struct** Disease\_Node {

    string disease\_code; // Код заболевания

    string registration\_date; // Дата регистрации заболевания

    Disease\_Node**\*** next;

};

// Структура узла списка пациентов

**struct** Patient\_Node {

    string policy\_number; // Номер мед. полиса

    string surname; // Фамилия больного

    Disease\_Node**\*** diseases; // Указатель на линейный список заболеваний

    Patient\_Node**\*** prev;

    Patient\_Node**\*** next;

};

**class** PatientList {

**public:**

    PatientList(); // конструктор

    ~PatientList(); // деструктор

**void** insert\_node(string policy\_number, string surname); // вставка узла

**void** delete\_node(Patient\_Node**\*** node); // удаление узла

**void** print\_list\_forward() **const**; // вывод списка слева направо

**void** print\_list\_backward() **const**; // вывод списка справа налево

    Patient\_Node**\*** search\_node(string policy\_number) **const**; // поиск узла по номеру мед. полиса

    // Дополнительные операции

**void** insert\_front\_disease(Patient\_Node**\*** patient, string disease\_code, string registration\_date); // вставить узел в начало подсписка заболеваний

**void** delete\_disease(Patient\_Node**\*** patient, string disease\_code); // удалить из списка кодов заболеваний указанного пациента, узел с указанным кодом

    PatientList**\*** new\_list\_of\_mediccal\_card(); // сформировать новый список, включив в него информацию о пациентах, у которых зарегистрировано более 5 заболеваний

**private:**

    Patient\_Node**\*** head;

    Patient\_Node**\*** tail;

};

**#endif** // MEDICAL\_POLICY\_H

Блок кода 10 – medical\_policy.h

**#include** "medical\_policy.h"

// Конструктор

PatientList::PatientList() {

    head **=** **nullptr**;

    tail **=** **nullptr**;

}

// Деструктор

PatientList::**~**PatientList() {

**while** (head **!=** **nullptr**) {

        Patient\_Node**\*** temp **=** head;

        head **=** head->next;

**delete** temp;

    }

}

// Вставка узла

**void** PatientList::insert\_node(string policy\_number, string surname) {

    Patient\_Node**\*** new\_node **=** **new** Patient\_Node;

    new\_node->policy\_number **=** policy\_number;

    new\_node->surname **=** surname;

    new\_node->diseases **=** **nullptr**; // Пациент пока не болен

    new\_node->next **=** **nullptr**;

    new\_node->prev **=** tail;

**if** (tail **!=** **nullptr**)

        tail->next **=** new\_node;

    tail **=** new\_node;

**if** (head **==** **nullptr**)

        head **=** new\_node;

}

// Удаление узла

**void** PatientList::delete\_node(Patient\_Node**\*** node) {

**if** (node->prev **!=** **nullptr**)

        node->prev->next **=** node->next;

**else**

        head **=** node->next;

**if** (node->next **!=** **nullptr**)

        node->next->prev **=** node->prev;

**else**

        tail **=** node->prev;

**delete** node;

}

// Вывод списка слева направо

**void** PatientList::print\_list\_forward() **const** {

**for** (Patient\_Node**\*** node **=** head; node **!=** **nullptr**; node **=** node->next) {

        cout **<<** node->policy\_number **<<** ": " **<<** node->surname **<<** endl;

**for** (Disease\_Node**\*** disease **=** node->diseases; disease **!=** **nullptr**; disease **=** disease->next)

            cout **<<** "Заболевание: " **<<** disease->disease\_code **<<** ", Дата регистрации: " **<<** disease->registration\_date **<<** endl;

    }

}

// Вывод списка справа налево

**void** PatientList::print\_list\_backward() **const** {

**for** (Patient\_Node**\*** node **=** tail; node **!=** **nullptr**; node **=** node->prev) {

        cout **<<** node->policy\_number **<<** ": " **<<** node->surname **<<** endl;

**for** (Disease\_Node**\*** disease **=** node->diseases; disease **!=** **nullptr**; disease **=** disease->next)

            cout **<<** "Заболевание: " **<<** disease->disease\_code **<<** ", Дата регистрации: " **<<** disease->registration\_date **<<** endl;

    }

}

// Поиск узла по номеру мед. полиса

Patient\_Node**\*** PatientList::search\_node(string policy\_number) **const** {

**for** (Patient\_Node**\*** node **=** head; node **!=** **nullptr**; node **=** node->next)

**if** (node->policy\_number **==** policy\_number) {

**return** node;

    }

**return** **nullptr**;

}

// Вставка в начало списка заболеваний

**void** PatientList::insert\_front\_disease(Patient\_Node**\*** patient, string disease\_code, string registration\_date) {

    Disease\_Node**\*** new\_disease **=** **new** Disease\_Node;

    new\_disease->disease\_code **=** disease\_code;

    new\_disease->registration\_date **=** registration\_date;

    new\_disease->next **=** patient->diseases;

    patient->diseases **=** new\_disease;

}

// Удалить из списка кодов заболеваний указанного пациента, узел с указанным кодом

**void** PatientList::delete\_disease(Patient\_Node**\*** patient, string disease\_code) {

    Disease\_Node**\*** temp **=** patient->diseases;

    Disease\_Node**\*** prev **=** **nullptr**;

**if** (temp **!=** **nullptr** **&&** temp->disease\_code **==** disease\_code) {

        patient->diseases **=** temp->next;

**delete** temp;

**return**;

    }

**while** (temp **!=** **nullptr** **&&** temp->disease\_code **!=** disease\_code) {

        prev **=** temp;

        temp **=** temp->next;

    }

**if** (temp **==** **nullptr**)

**return**;

    prev->next **=** temp->next;

**delete** temp;

}

// Формирование ногового списка больных, где у каждого по 5 болезний

PatientList**\*** PatientList::new\_list\_of\_mediccal\_card() {

    PatientList**\*** newList **=** **new** PatientList();

**for** (Patient\_Node**\*** node **=** head; node **!=** **nullptr**; node **=** node->next) {

**int** count **=** 0;

**for** (Disease\_Node**\*** disease **=** node->diseases; disease **!=** **nullptr**; disease **=** disease->next)

**++**count;

**if** (count **>** 5)

            newList->insert\_node(node->policy\_number, node->surname);

    }

**return** newList;

}

Блок кода 11 – medical\_policy.cpp

**#include** <iostream>

**#include** <stdlib.h>

**#include** "medical\_policy.h"

**using** **namespace** std;

**void** manual\_input(PatientList**\*\*** patientList) {

    cout **<<** "Введите количество пациентов в списке: ";

**int** size;

    cin **>>** size;

**for** (**int** i **=** 0; i **<** size; **++**i) {

        string policy\_number, surname;

        cout **<<** "Введите номер мед. полиса для пациента " **<<** i **+** 1 **<<** ": ";

        cin **>>** policy\_number;

        cout **<<** "Введите фамилию пациента " **<<** i **+** 1 **<<** ": ";

        cin **>>** surname;

        (**\***patientList)->insert\_node(policy\_number, surname);

        cout **<<** "Введите количество заболеваний у пациента: ";

**int** count;

        cin **>>** count;

**for** (**int** j **=** 0; j **<** count; **++**j) {

            cout **<<** "Введите код заболевания: ";

            string disease\_code, registration\_date;

            cin **>>** disease\_code;

            cout **<<** "Введите дату регистрации заболевания: ";

            cin **>>** registration\_date;

            Patient\_Node**\*** patient **=** (**\***patientList)->search\_node(policy\_number);

**if** (patient **!=** **nullptr**)

                (**\***patientList)->insert\_front\_disease(patient, disease\_code, registration\_date);

        }

    }

}

**void** automatic\_method(PatientList**\*\*** patientList) {

    (**\***patientList)->insert\_node("pol1", "Иванов");

    (**\***patientList)->insert\_front\_disease((**\***patientList)->search\_node("pol1"), "A28", "11.11.2023");

    (**\***patientList)->insert\_node("pol2", "Смирнов");

    (**\***patientList)->insert\_front\_disease((**\***patientList)->search\_node("pol2"), "B05.3+", "01.03.2020");

    (**\***patientList)->insert\_node("pol3", "Кузнецов");

    (**\***patientList)->insert\_front\_disease((**\***patientList)->search\_node("pol3"), "B82", "31.01.2019");

    (**\***patientList)->insert\_node("pol4", "Васильев");

    (**\***patientList)->insert\_front\_disease((**\***patientList)->search\_node("pol4"), "B26.0+", "01.01.2024");

    (**\***patientList)->insert\_node("pol5", "Петров");

**for** (**int** i **=** 1; i **<=** 7; **++**i) {

        string disease\_code **=** "A" **+** to\_string(i **\*** 2 **/** 3);

        string registration\_date **=** "0" **+** to\_string(i) **+** ".02.2024";

        (**\***patientList)->insert\_front\_disease((**\***patientList)->search\_node("pol5"), disease\_code, registration\_date);

    }

    (**\***patientList)->insert\_node("pol6", "Соколов");

    (**\***patientList)->insert\_front\_disease((**\***patientList)->search\_node("pol6"), "B26.0+", "01.01.2024");

    (**\***patientList)->insert\_node("pol7", "Орлов");

}

**void** displayFunctions(PatientList**\*\*** patientList) {

    cout **<<** "Исходный список: " **<<** endl;

    (**\***patientList)->print\_list\_forward();

    cout **<<** "------------------------------------------------------------------------------------------------------------------------" **<<** endl;

    cout **<<** "Выберите функцию:" **<<** endl

**<<** "1. Вставить узел" **<<** endl

**<<** "2. Удалить узел" **<<** endl

**<<** "3. Вывод список слева направо" **<<** endl

**<<** "4. Вывод список справа налево" **<<** endl

**<<** "5. Поиск узла по номеру мед. полиса" **<<** endl

**<<** "6. Вставить узел в начало подсписка заболеваний" **<<** endl

**<<** "7. Удалить из списка кодов заболеваний указанного пациента, узел с указанным кодом" **<<** endl

**<<** "8. Сформировать новый список, включив в него информацию о пациентах, у которых зарегистрировано более 5 заболеваний" **<<** endl

**<<** "9. Выполнить все операции по порядку" **<<** endl

**<<** "10. Вернуться в главное меню" **<<** endl;

    cout **<<** "------------------------------------------------------------------------------------------------------------------------" **<<** endl;

**int** command;

    cin **>>** command;

    cout **<<** "------------------------------------------------------------------------------------------------------------------------" **<<** endl;

**switch** (command) {

**case** 1: { // Вставка узла(мед. полиса)

        cout **<<** "Введите номер мед. полиса для вставки: ";

        string policy\_number, surname;

        cin **>>** policy\_number;

        cout **<<** "Введите фамилию пациента: ";

        cin **>>** surname;

        (**\***patientList)->insert\_node(policy\_number, surname);

        (**\***patientList)->print\_list\_forward();

**break**;

    }

**case** 2: { // Удаление уззла

        cout **<<** "Введите номер мед. полиса для удаления: ";

        string policy\_number;

        cin **>>** policy\_number;

        Patient\_Node**\*** node **=** (**\***patientList)->search\_node(policy\_number);

**if** (node **!=** **nullptr**)

            (**\***patientList)->delete\_node(node);

**else**

            cout **<<** "Узел не найден" **<<** endl;

        (**\***patientList)->print\_list\_forward();

**break**;

    }

**case** 3: { // Вывода списка слева направо

        (**\***patientList)->print\_list\_forward();

**break**;

    }

**case** 4: { // Вывод списка справа налево

        (**\***patientList)->print\_list\_backward();

**break**;

    }

**case** 5: { // Реализация поиска узла по номеру мед. полиса

        string policy\_number;

        cout **<<** "Введите номер мед. полиса: ";

        cin **>>** policy\_number;

        Patient\_Node**\*** node **=** (**\***patientList)->search\_node(policy\_number);

**if** (node **!=** **nullptr**)

            cout **<<** "Узел найден: " **<<** node->policy\_number **<<** ", " **<<** node->surname **<<** endl;

**else**

            cout **<<** "Узел не найден" **<<** endl;

**break**;

    }

**case** 6: {

        // Реализация вставки узла в начало подсписка заболеваний

        string policy\_number, disease\_code, registration\_date;

        cout **<<** "Введите номер мед. полиса: ";

        cin **>>** policy\_number;

        cout **<<** "Введите код заболевания: ";

        cin **>>** disease\_code;

        cout **<<** "Введите дату регистрации заболевания: ";

        cin **>>** registration\_date;

        Patient\_Node**\*** patient **=** (**\***patientList)->search\_node(policy\_number);

**if** (patient **!=** **nullptr**)

            (**\***patientList)->insert\_front\_disease(patient, disease\_code, registration\_date);

**else**

            cout **<<** "Пациент не найден" **<<** endl;

        (**\***patientList)->print\_list\_forward();

**break**;

    }

**case** 7: {

        // Реализация удаления из списка кодов заболеваний указанного пациента, узел с указанным кодом

        string policy\_number, disease\_code;

        cout **<<** "Введите номер мед. полиса: ";

        cin **>>** policy\_number;

        cout **<<** "Введите код заболевания: ";

        cin **>>** disease\_code;

        Patient\_Node**\*** patient **=** (**\***patientList)->search\_node(policy\_number);

**if** (patient **!=** **nullptr**)

            (**\***patientList)->delete\_disease(patient, disease\_code);

**else**

            cout **<<** "Пациент не найден." **<<** endl;

        (**\***patientList)->print\_list\_forward();

**break**;

    }

**case** 8: {

        // Реализация формирования нового списка, включив в него информацию о пациентах, у которых зарегистрировано более 5 заболеваний

        PatientList**\*** newList **=** (**\***patientList)->new\_list\_of\_mediccal\_card();

        newList->print\_list\_forward();

**delete** newList;

**break**;

    }

**case** 9: {

        // Реализация выполнения всех операций по порядку

        // Вставка узла

        string policy\_number, surname;

        cout **<<** "Введите номер мед. полиса для вставки: ";

        cin **>>** policy\_number;

        cout **<<** "Введите фамилию пациента: ";

        cin **>>** surname;

        (**\***patientList)->insert\_node(policy\_number, surname);

        (**\***patientList)->print\_list\_forward();

        // Удаление узла

        cout **<<** "Введите номер мед. полиса для удаления: ";

        cin **>>** policy\_number;

        Patient\_Node**\*** node **=** (**\***patientList)->search\_node(policy\_number);

**if** (node **!=** **nullptr**)

            (**\***patientList)->delete\_node(node);

**else**

            cout **<<** "Узел не найден." **<<** endl;

        (**\***patientList)->print\_list\_forward();

        // Вывод списка слева направо

        (**\***patientList)->print\_list\_forward();

        // Вывод списка справа налево

        (**\***patientList)->print\_list\_backward();

        // Поиск узла по номеру мед. полиса

        cout **<<** "Введите номер мед. полиса: ";

        cin **>>** policy\_number;

        node **=** (**\***patientList)->search\_node(policy\_number);

**if** (node **!=** **nullptr**)

            cout **<<** "Узел найден: " **<<** node->policy\_number **<<** ", " **<<** node->surname **<<** endl;

**else**

            cout **<<** "Узел не найден." **<<** endl;

        // Вставка узла в начало подсписка заболеваний

        string disease\_code, registration\_date;

        cout **<<** "Введите номер мед. полиса: ";

        cin **>>** policy\_number;

        cout **<<** "Введите код заболевания: ";

        cin **>>** disease\_code;

        cout **<<** "Введите дату регистрации заболевания: ";

        cin **>>** registration\_date;

        Patient\_Node**\*** patient **=** (**\***patientList)->search\_node(policy\_number);

**if** (patient **!=** **nullptr**)

            (**\***patientList)->insert\_front\_disease(patient, disease\_code, registration\_date);

**else**

            cout **<<** "Пациент не найден." **<<** endl;

        (**\***patientList)->print\_list\_forward();

        // Удаление из списка кодов заболеваний указанного пациента, узел с указанным кодом

        cout **<<** "Введите номер мед. полиса: ";

        cin **>>** policy\_number;

        cout **<<** "Введите код заболевания: ";

        cin **>>** disease\_code;

        patient **=** (**\***patientList)->search\_node(policy\_number);

**if** (patient **!=** **nullptr**)

            (**\***patientList)->delete\_disease(patient, disease\_code);

**else**

            cout **<<** "Пациент не найден." **<<** endl;

        (**\***patientList)->print\_list\_forward();

        // Сформировать новый список, включив в него информацию о пациентах, у которых зарегистрировано более 5 заболеваний

        PatientList**\*** newList **=** (**\***patientList)->new\_list\_of\_mediccal\_card();

        newList->print\_list\_forward();

**delete** newList;

**break**;

    }

**case** 10:

**break**;

**default**: {

        cout **<<** "Неверный выбор. Пожалуйста, попробуйте еще раз." **<<** endl;

**break**;

    }

    }

}

**int** main() {

    setlocale(LC\_ALL, "Russian");

**int** state **=** 1, num;

    PatientList**\*** patientList **=** **new** PatientList();

**bool** f\_run **=** **true**;

**do** {

**if** (**!**f\_run)

            cout **<<** "------------------------------------------------------------------------------------------------------------------------" **<<** endl;

        cout **<<** "Выберите способ создания списка:" **<<** endl **<<** "1. Ручной ввод списка" **<<** endl **<<** "2. Использование заготовленного списка" **<<** endl;

        cout **<<** "------------------------------------------------------------------------------------------------------------------------" **<<** endl;

        cin **>>** num;

        cout **<<** "------------------------------------------------------------------------------------------------------------------------" **<<** endl;

**switch** (num) {

**case** 1:

            manual\_input(**&**patientList);

**break**;

**case** 2:

            automatic\_method(**&**patientList);

**break**;

**default**:

            cout **<<** "Ошибка! Введите значение 1 или 2!" **<<** endl;

        }

        displayFunctions(**&**patientList);

        cout **<<** "------------------------------------------------------------------------------------------------------------------------" **<<** endl;

        cout **<<** "Подменю" **<<** endl;

        cout **<<** "1. Продолжить" **<<** endl;

        cout **<<** "2. Завершить" **<<** endl;

        cout **<<** "------------------------------------------------------------------------------------------------------------------------" **<<** endl;

        cin **>>** state;

        cout **<<** "------------------------------------------------------------------------------------------------------------------------" **<<** endl;

**if** (state **==** 2)

            cout **<<** "Завершение работы" **<<** endl;

**else** **if** (state **!=** 1) {

            cout **<<** "Вы ввели недопустимые значения!" **<<** endl;

**return** **-**1;

        }

        f\_run **=** **false**;

    } **while** (state **==** 1);

**delete** patientList;

**return** 0;

}

Блок кода 12 – main.cpp

### **2.3.2 Тестирование**

Проведем тестирование основываясь на данные из таблиц 1-6. Результаты тестирования №1 строки представлены на рисунках 5-11.

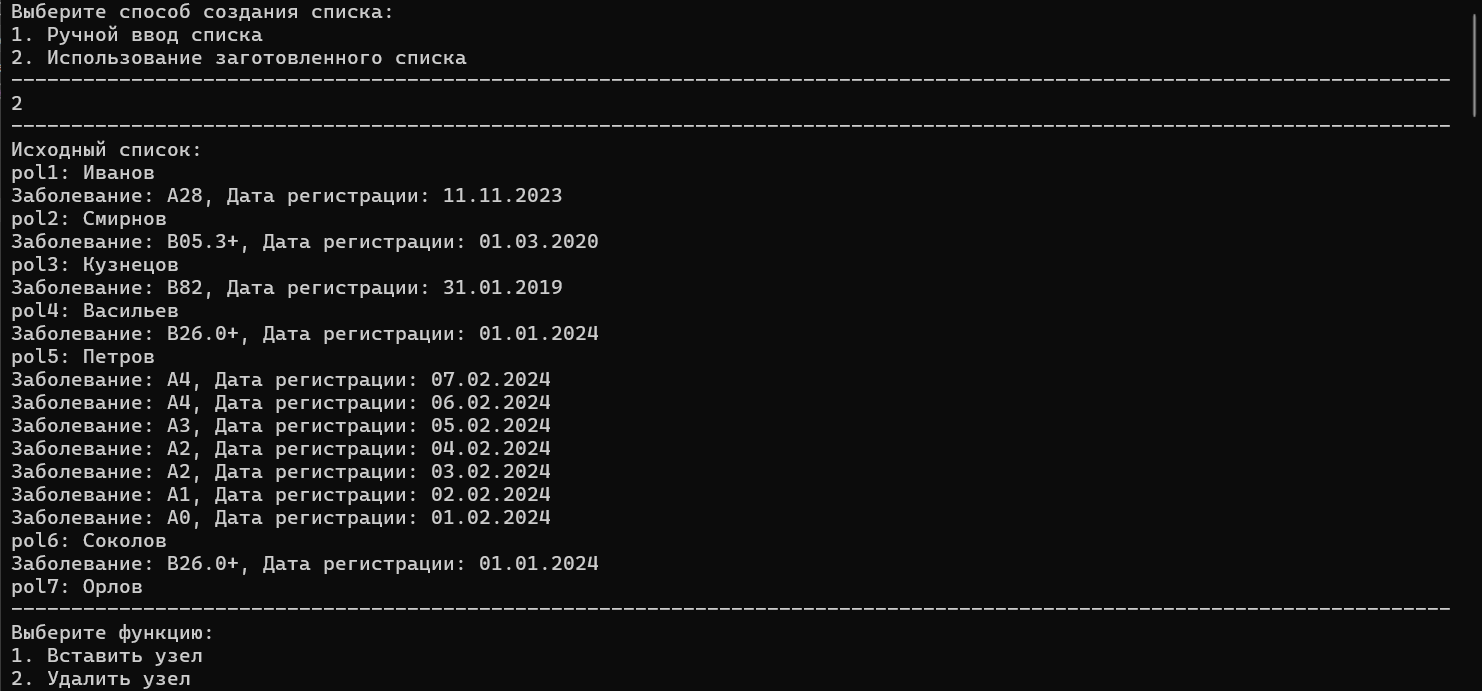


Рисунок 5 - Тестирование программы на данных №1 строки

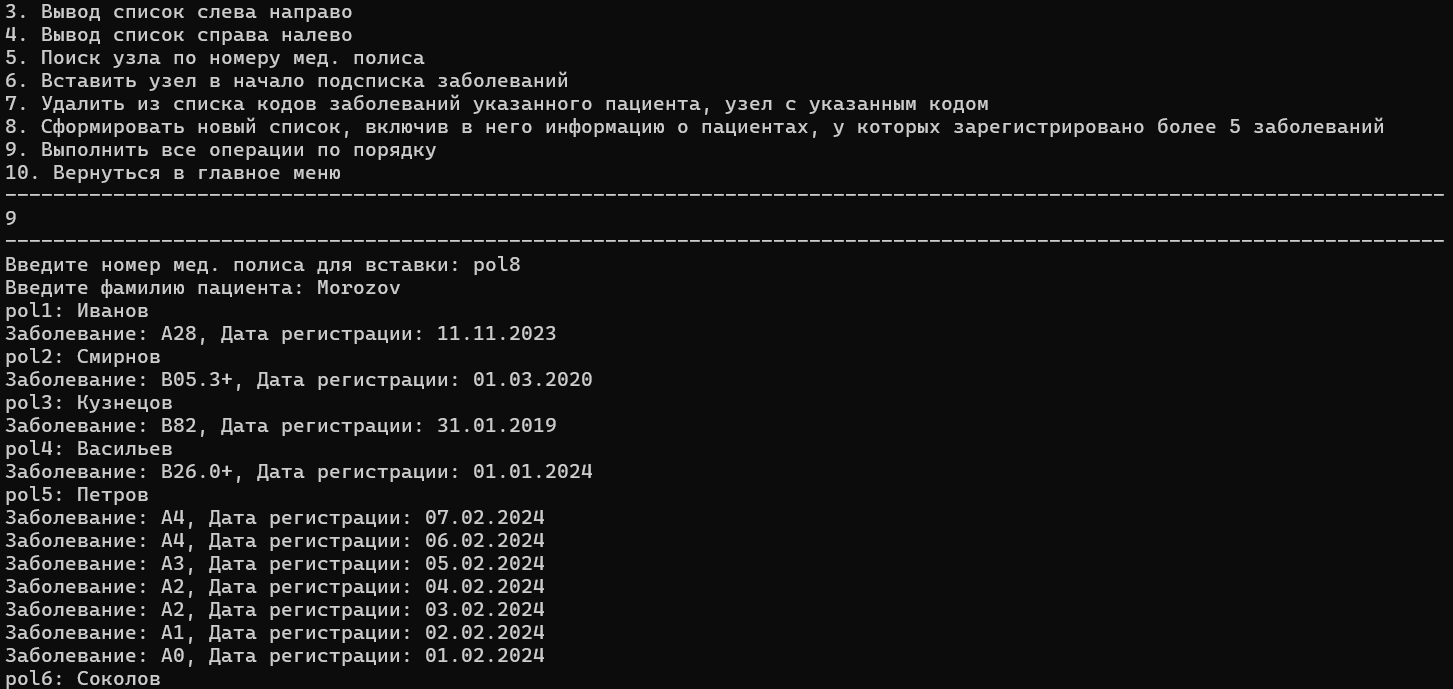


Рисунок 6 - Тестирование программы на данных №1 строки

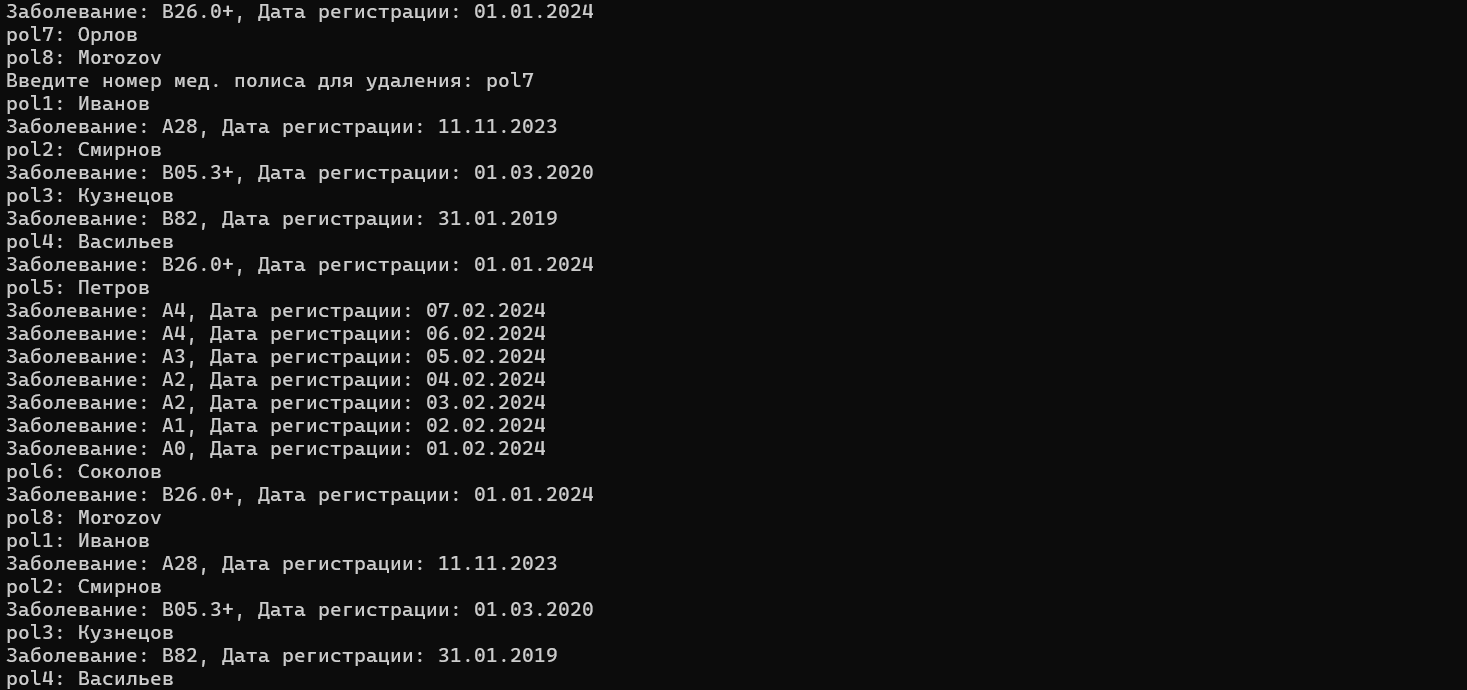


Рисунок 7 - Тестирование программы на данных №1 строки

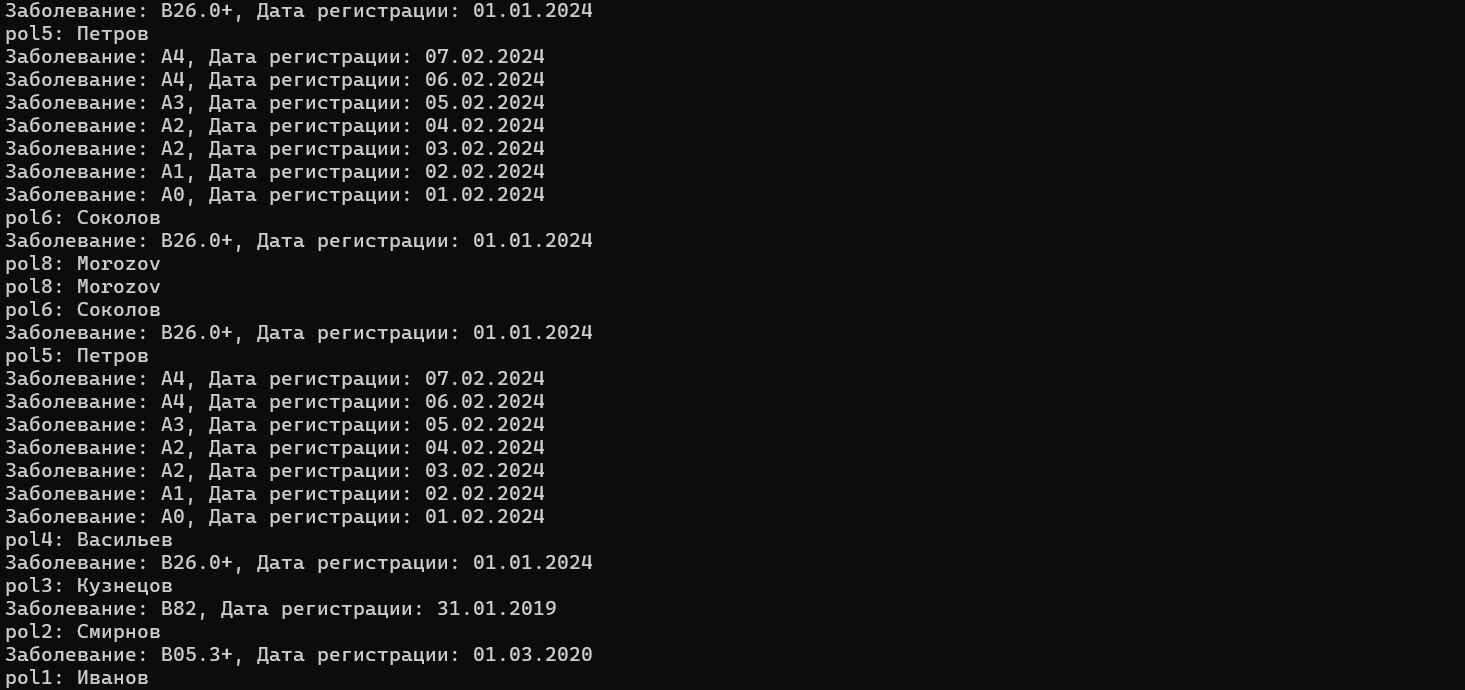


Рисунок 8 - Тестирование программы на данных №1 строки

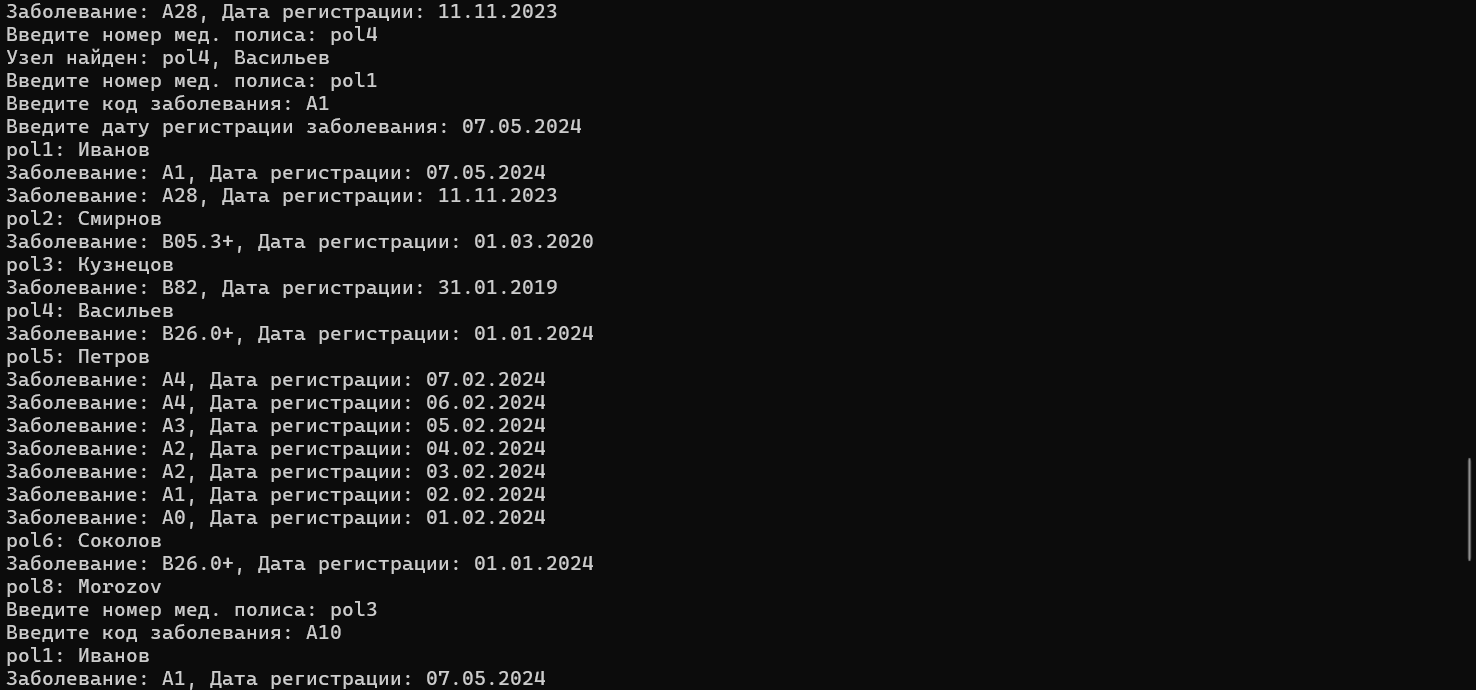


Рисунок 9 - Тестирование программы на данных №1 строки

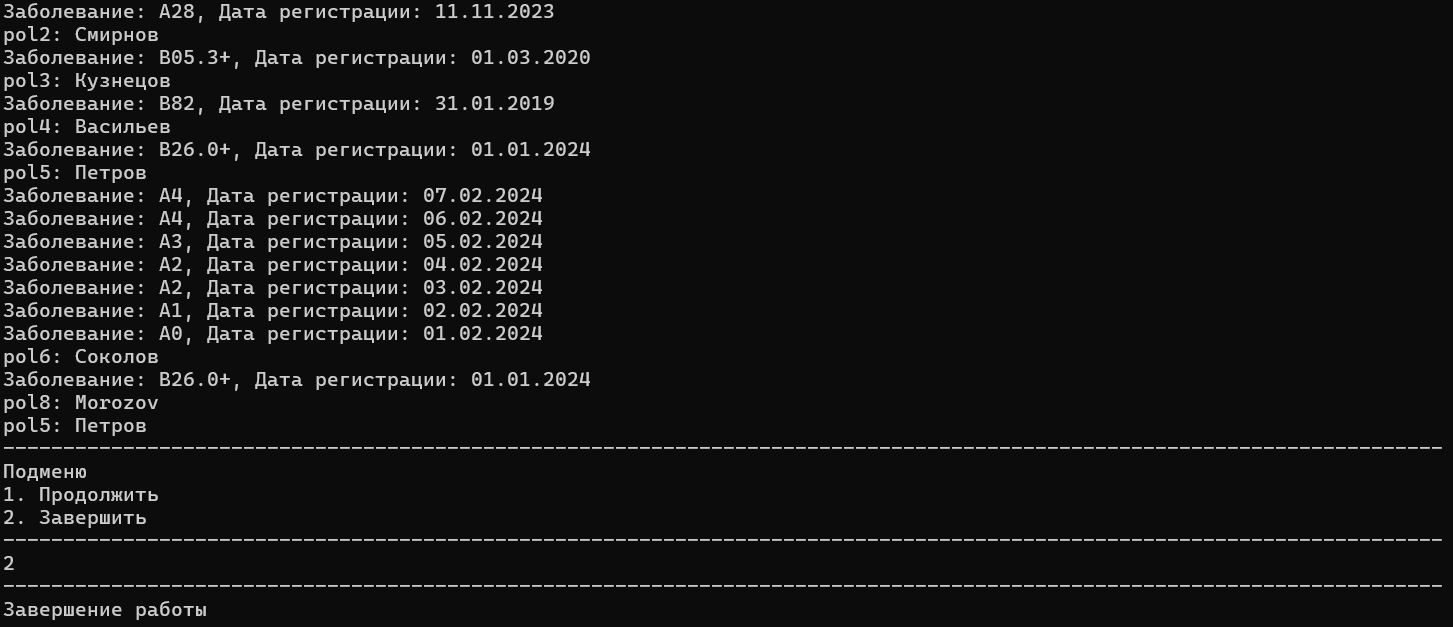


Рисунок 10 - Тестирование программы на данных №1 строки

## **2.4 Вывод по заданию**

Двусвязный список, также известный как двунаправленный список, представляет собой структуру данных, состоящую из узлов, каждый из которых содержит данные и два указателя: на предыдущий и следующий узлы. Эта структура позволяет эффективно вставлять и удалять элементы в любом месте списка, а также обеспечивает возможность перебора элементов как в прямом, так и в обратном направлении.

Для оптимального использования двусвязного списка следует учитывать следующие моменты:

1. При добавлении и удалении элементов необходимо обновлять указатели на предыдущий и следующий узлы.

2. Важно правильно обрабатывать краевые случаи, такие как вставка в начало или конец списка.

3. При использовании итераторов необходимо обращаться к элементам списка, учитывая направление движения.

Сложность алгоритмов для двусвязного списка такова:

1. Вставка и удаление элемента: O(1), если известны узлы, перед которыми или за которыми нужно вставить или удалить элемент.

2. Поиск элемента: O(n), где n - количество элементов в списке.

3. Обращение к элементам: O(n).

Таким образом, двусвязный список предоставляет удобную структуру данных для операций вставки, удаления и обращения к элементам как в прямом, так и в обратном направлении. Это особенно ценно в случаях, когда требуется доступ к элементам в обратном порядке или частое добавление и удаление элементов из середины списка. Однако, если часто необходимо производить поиск элементов в списке, другие структуры данных, такие как хэш-таблицы или деревья, могут быть более эффективными.

# 

# **3 ВЫВОДЫ**

В процессе выполнения практической работы были достигнуты следующие цели:

- Освоены навыки управления двусвязным списком в программах на языке C++.

- Получены практические навыки работы с двусвязным списком в программах на языке C++.

- Проведен анализ структуры согласно индивидуальному варианту.

- Проведен анализ операций в соответствии с индивидуальным вариантом.

- Разработана программа для работы со структурой в соответствии с индивидуальным вариантом.

- Написаны программы для реализации операций в соответствии с индивидуальным вариантом.

- Создано меню выбора способа ввода данных и реализации функций.

- Оценена сложность первой дополнительной операции.

- Протестирована программа с различными данными и методами ввода.

Таким образом, главная цель практической работы - получение знаний и навыков работы с двусвязным списком в программах на языке C++ - была достигнута.

# **4 ЛИТЕРАТУРА**

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – СПб: Питер, 2017. – 288 с.

2. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.

3. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск, 2-е изд. – М.: ООО «И.Д. Вильямс», 2018. – 832 с.

4. Кораблин Ю.П. Структуры и алгоритмы обработки данных: учебно-методическое пособие / Ю.П. Кораблин, В.П. Сыромятников, Л.А. Скворцова. – М.: РТУ МИРЭА, 2020. — 219 с.

5. Кормен Т.Х. и др. Алгоритмы: построение и анализ, 3-е изд. – М.: ООО «И.Д.Вильямс», 2013. – 1328 с.

6. Макконнелл Дж. Основы современных алгоритмов. Активный обучающий метод. 3-е доп. изд., - М.: Техносфера, 2018. – 416 с.

7. Седжвик Р. Фундаментальные алгоритмы на C++. Анализ/Структуры данных/Сортировка/Поиск. – К.: Издательство «Диасофт», 2001. – 688 с.

8. Скиена С. Алгоритмы. Руководство по разработке, - 2-е изд. – СПб: БХВ-Петербург, 2011. – 720 с.

9. Хайнеман Д. и др. Алгоритмы. Справочник с примерами на C, C++, Java и Python, 2-е изд. – СПб: ООО «Альфа-книга», 2017. – 432 с.

10. AlgoList – алгоритмы, методы, исходники [Электронный ресурс]. URL: http://algolist.manual.ru/ (дата обращения 15.03.2022).

11. Алгоритмы – всё об алгоритмах / Хабр [Электронный ресурс]. URL: https://habr.com/ru/hub/algorithms/ (дата обращения 15.03.2022).

12. НОУ ИНТУИТ | Технопарк Mail.ru Group: Алгоритмы и структуры данных [Электронный ресурс]. URL: https://intuit.ru/studies/courses/3496/738/info (дата обращения 15.03.2022).