

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

**ОТЧЕТ**  
**по курсовой работе**  
**по дисциплине «Программирование»**  
**Тема: Типы данных, определяемые**  
**пользователем. Структуры. Линейные**  
**структуры данных. Динамические**  
**массивы и двусвязные списки.**

Студентка гр. 0324

Сотина Е.А.

Преподаватель

Глущенко А.Г.

Санкт-Петербург

2020

## **Цель работы.**

Применить полученные знания о динамических массивах и двусвязных списках, организации структур;

## **Основные теоретические положения.**

Структуры представляют собой группы связанных между собой, как правило, разнотипных переменных, объединенных в единый объект, в отличие от массива, все элементы которого однотипны. В языке C++ структура является видом класса и обладает всеми его свойствами. Чаще всего ограничиваются тем, как структуры представлены в языке C:

```
struct [имя_типа] {  
    тип_1 элемент_1;  
    тип _2 элемент_2;  
    ...  
    тип_k элемент_k;  
} [ список_описателей ];
```

Описание структуры начинается ключевым словом struct. Каждая входящая в структуру переменная называется членом (полем, элементом) структуры и описывается типом данных и именем. Поля структуры могут быть любого типа данных. Их количество не лимитировано.

Вся эта конструкция является инструкцией языка программирования, поэтому после нее всегда должен ставиться символ ‘;’.

При описании структуры память для размещения данных не выделяется. Работать с описанной структурой можно только после того, как будет определена переменная (переменные) этого типа данных, только при этом компилятор выделит необходимую память.

Для инициализации структуры значения ее элементов перечисляют в фигурных скобках в порядке их описания:

```
struct complex{  
    float real, im;  
} data [2][2] = {  
    {{1,1}, {2,2}},  
    {{3,3}, {4,4}}  
};
```

Все поля структурных переменных располагаются в непрерывной области памяти одно за другим. Общий объем памяти, занимаемый структурой, равен

сумме размеров всех полей структуры. Для определения размера структуры следует использовать инструкцию `sizeof()`.

Для того чтобы записать данные в структурную переменную, необходимо каждому полю структуры присвоить определенное значение. Для этого необходимо использовать оператор `'.'` («точка»):

```
struct Stack { // Стек
    float arr[100];
    short topIndex;
};
...
Stack stack; // Объявляем переменную типа Stack
Stack.arr[0] = 1;
...
```

При доступе к определенному полю его следует рассматривать как обычную переменную, тип данных которой соответствует типу этого поля. Поля структур могут участвовать в качестве операндов любых выражений, допускающих использование операндов соответствующего типа данных.

Копирование данных из одной структурной переменной в другую осуществляется простой операцией присваивания, независимо от количества полей и размера структуры (это можно делать только в том случае, когда обе переменные одного и того же типа).

В программировании очень часто используются такие конструкции, как массивы структур. Например, сведения о студентах некоторой учебной группы можно хранить в массиве студентов:

```
t_Student Gruppa_N [30];
```

Был определен 30-элементный массив, каждый элемент которого предназначен для хранения данных одного студента. Получение доступа к данным некоторого студента из группы  $N$  осуществляется обычной индексацией переменной массива. Поскольку поля структуры могут быть любого типа данных, то они в свою очередь могут быть другой структурой или массивом других структур:

```
struct Stud
{
    char FN[100];
    short listNumber;
};

struct Group
{
```

```

    int groupNumber;
    short students;
    Stud stud[30];
};

```

Но в структуре поля нельзя использовать элемент, тип которого совпадает с типом самой структуры, так как рекурсивное использование структур запрещено.

Любая структурная переменная занимает в памяти определенное положение, характеризующееся конкретным адресом. Для работы с адресами структурных переменных (как и для простых переменных) можно использовать указатели. Указатели на структурные переменные определяются точно так же, как и для обычных переменных. Разыменование указателя (обращение к данным по адресу, хранящемуся в указателе) осуществляется также обычным образом.

Через указатели можно работать с отдельными полями структур. Для доступа к полю структуры через указатель используется оператор '→' («стрелка»), а не «точка».

Структуры можно использовать в качестве параметров функций, как и обычные переменные. Для структур поддерживаются все три механизма передачи данных: по значению, через указатели и по ссылке.

Передачу структур в функции по значению необходимо использовать аккуратно:

```

void WriteStudent ( t_Student S )
{
    cout << "Фамилия: " << S.Fam << endl;
    cout << "Имя: " << S.Name << endl;
    cout << "Год рождения: " << S.Year << endl;
    if ( S.Sex )
        cout << "Пол: " << "М\n";
    else
        cout << "Пол: " << "Ж\n";
    cout << "Средний балл: " << S.Grade << endl;
}

```

Вызов такой функции сопровождается дополнительным расходом памяти для создания локальной переменной *S* и дополнительными затратами времени на физическое копирование данных из аргумента в параметр *S*. Учитывая то, что объем структур может быть очень большим, эти дополнительные затраты вычислительных ресурсов могут быть чрезмерными.

Предпочтительно использование передачи структуры по указателю или ссылке:

```

void WriteStudent ( t_Student *S )
{
    cout << "Фамилия: " << S -> Fam << endl;
    cout << "Имя: " << S -> Name << endl;
    cout << "Год рождения: " << S -> Year << endl;
    if ( S -> Sex )
        cout << "Пол: " << "М\n";
    else
        cout << "Пол: " << "Ж\n";
    cout << "Средний балл: " << S -> Grade << endl;
}

```

Фактической передачи данных в функцию не осуществляется. Дополнительные затраты памяти для создания локальной переменной небольшие – это адрес памяти (4 байта, независимо от размера самой структуры). Вызов такой функции будет происходить быстрее, а расход памяти будет существенно меньше, чем при передаче данных по значению.

Передача по ссылке по эффективности эквивалентна передаче данных через указатель. Однако, поскольку при передаче данных по ссылке все адресные преобразования берет на себя компилятор, существенно упрощается программирование действий со структурами. При использовании ссылочных параметров структурных типов доступ к членам структуры осуществляется обычным способом – с помощью оператора «точка».

Недостатком этих способов является то, что случайные изменения значений полей структуры внутри функции отразятся на значении аргумента после окончания работы функции. Если необходимо предотвратить изменения переданных по адресу аргументов, можно при определении соответствующего параметра объявить его константой (использовать спецификатор `const`).

Схема распределения памяти под программу показана на рис. 1.

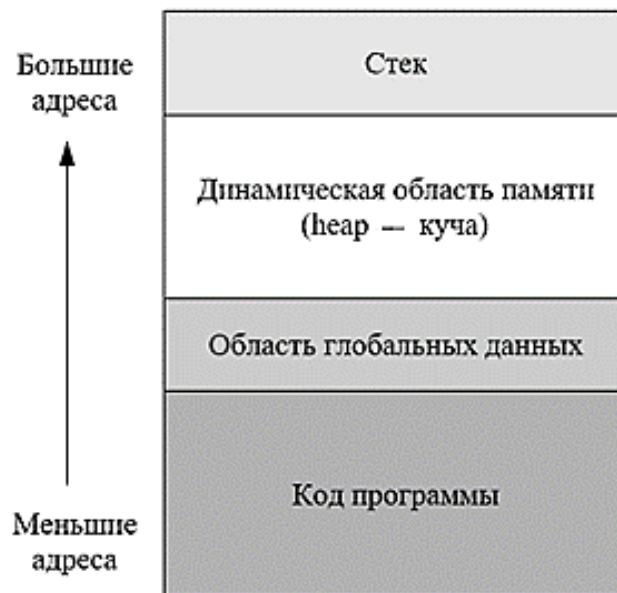


Рис. 1. Схема распределения памяти под программу

Область кода программы предназначена для хранения инструкций функций программы, обеспечивающих обработку данных.

Данные в программе представляются переменными и константами.

Для хранения глобальных данных предназначена область глобальных данных.

Стек программы используется при вызове функций для передачи параметров и хранения локальных данных.

Распределение памяти для хранения всех обычных переменных осуществляется компилятором, адреса и объемы соответствующих участков памяти (в области глобальных данных) жестко закреплены за этими переменными на все время работы программы и изменены быть не могут.

Однако во многих задачах невозможно заранее предсказать, сколько места (количество переменных, объемы массивов и т. д.) потребуется для решения задачи – это так называемые задачи с неопределенной размерностью.

Решить эту проблему можно лишь в том случае, если иметь механизм, позволяющий создавать новые объекты по мере возникновения необходимости в этих объектах или изменять объемы памяти, выделенные под эти объекты (например, объемы массивов).

Между областью глобальных данных и стеком располагается так называемая динамическая область памяти, которую и можно использовать в процессе работы программы для реализации механизма динамического управления памятью.

Для того чтобы создать в динамической области некоторый объект, необходима одна обычная переменная-указатель (не динамическая переменная). Сколько таких объектов понадобится для одновременной обработки, столько

необходимо иметь обычных переменных-указателей. Таким образом, проблема задач неопределенной размерности созданием одиночных динамических объектов решена быть не может.

Решить эту проблему поможет возможность создавать в динамической области памяти массивы объектов с таким количеством элементов, которое необходимо в данный момент работы программы, т. е. создание динамических массивов. Действительно, для представления массива требуется всего одна переменная-указатель, а в самом массиве, на который ссылается этот указатель, может быть столько элементов, сколько требуется в данный момент времени.

Для создания одномерного динамического массива, элементами которого являются, например, действительные числа, используется следующий синтаксис инструкции new (стиль C++):

```
double *Arr = new double [100];
```

Освободить динамическую область от этого массива можно с помощью инструкции delete:

```
delete [] Arr;
```

После этого занятый участок памяти будет возвращен в список свободной памяти и может быть повторно использован для размещения других динамических объектов.

Язык C++ поддерживает и «старый», заимствованный от языка C, стиль работы с динамической областью. Довольно часто бывает полезно использовать именно этот механизм управления динамической памятью.

В языке C отсутствуют инструкции new и delete. Вместо них для управления динамической памятью используются библиотечные функции:

```
// Блок прототипов функций
{
    void *malloc (size);
    void *calloc(num, size);
    void free( void *memblock);
    void *realloc( void *memblock, size);
}
```

Функция malloc выделяет в динамической области size байт памяти и возвращает адрес этого участка в виде указателя (void \*).

Поскольку возвращаемый указатель не привязан ни к какому типу данных, при работе с ним потребуется явное приведение типов данных (см. пример далее).

Функция calloc выделяет в динамической области size \* num байт памяти и возвращает адрес этого участка в виде указателя (void \*).

Функция `free` освобождает участок динамической памяти по адресу `memblock` и возвращает его в список свободной памяти для повторного использования.

Функция `realloc` позволяет изменить размер (уменьшить или увеличить) ранее выделенной по адресу `memblock` памяти, установив новый размер выделенного участка равным `size` байт. При увеличении размера выделенного участка данные, которые хранились в старом участке, копируются в новый участок памяти. При уменьшении объема выделенного участка данные, которые хранились в нем, усекаются до нового размера. Функция возвращает указатель на область памяти нового размера.

Работа с одномерным динамическим массивом осуществляется так же, как и с обычным. При этом стиль использования динамических массивов `C` имеет весомое преимущество над `C++`, которое заключается в изменении размерности массива. Дело в том, что в `C++` нет функций увеличения размерности. Увеличить размер массива можно, создав новый динамический массив нужной размерности, скопировав данные из старого массива в новый и освободив память от старого массива.

Одномерный однонаправленный список представляет собой совокупность отдельных элементов, каждый из которых содержит две части – информационную (`Data`) и адресную (`Tail`).

Информационная часть предназначена для хранения полезных данных и может иметь практически любой тип. Адресная часть каждого элемента содержит адрес следующего элемента списка. Схематическое изображение такого списка представлено на рис. 2.

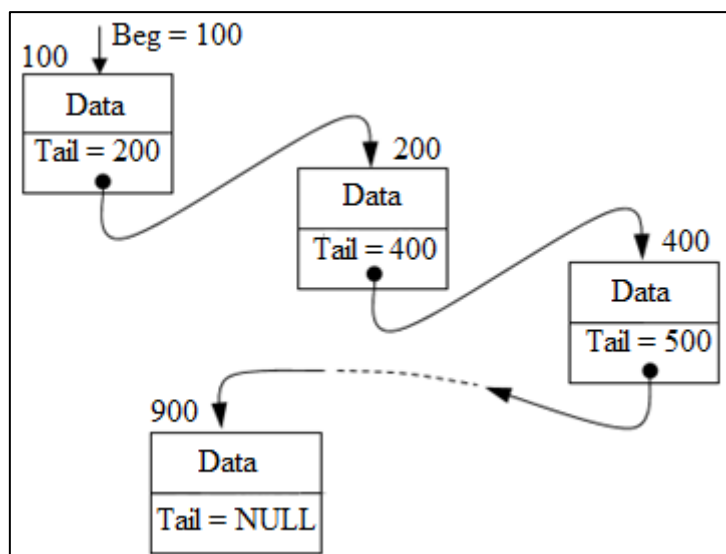


Рис. 2. Схематическое изображение односвязного списка



Для работы со списком достаточно знать только адрес его первого элемента (Beg). Зная адрес первого элемента списка, можно последовательно получить доступ к любому другому его элементу.

Поскольку каждый элемент списка должен иметь две части, логичнее всего представить его в виде следующей структуры:

```
struct list
{
    int data;
    list *tail;
};
```

Типовыми операциями при работе со списками являются:

- 1) создание списка;
- 2) освобождение памяти от списка (удаление списка);
- 3) доступ к заданному элементу списка для манипуляций с его информационной частью;
- 4) добавление нового элемента к списку;
- 5) удаление элемента из списка;
- 6) перестановка элемента списка на новую позицию внутри списка.

Достоинством подобных структур является простота добавления, удаления и перестановки элементов списка, которые осуществляются путем манипуляций с адресными частями без перезаписи всего списка.

Одним из недостатков односвязных списков является то, что узел (элемент списка) имеет указатель только на следующий элемент. Вернуться из текущего элемента к предыдущему явным способом невозможно.

Каждый узел двусвязного (двунаправленного) линейного списка содержит два поля указателей – на следующий и на предыдущий узлы. Указатель на предыдущий узел корня списка содержит нулевое значение. Указатель последнего узла также содержит нулевое значение.

Поскольку каждый элемент списка должен иметь три части, логичнее всего представить его в виде следующей структуры:

```
struct list
{
    int data;
    list *head;
    list *tail;
};
```

На рис. 3 показано схематическое представление двусвязного списка. Поле Head содержит адрес предыдущего элемента, поле Tail содержит адрес

следующего элемента списка. Такая организация списка позволяет перемещаться по его элементам в двух направлениях.

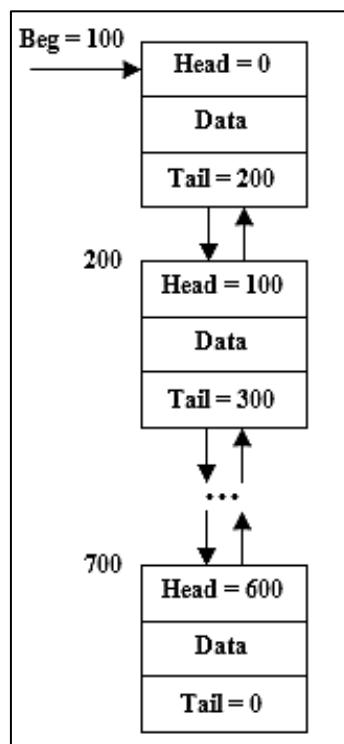


Рис. 3. Схематическое изображение двусвязного списка

Основные действия, производимые над узлами двусвязного линейного списка (ДЛС):

- 1) инициализация списка;
- 2) добавление узла в список;
- 3) удаление узла из списка;
- 4) удаление корня списка;
- 5) вывод элементов списка;
- 6) вывод элементов списка в обратном порядке;
- 7) взаимообмен двух узлов списка.

Порядок действия очень похож на односвязный линейный список, но необходимо учитывать, что в двусвязном списке имеется два указателя: на следующий и предыдущий элементы.

### Постановка задачи.

Необходимо создать массив структур, содержащий информацию о студентах: ФИО, пол, номер группы, номер в списке группы, оценки за прошедшую сессию (всего 3 экзамена и 5 дифференцированных зачетов), форма обучения, отметка

времени о внесении или изменении данных. Ввод и изменение данных обо всех студентах должен осуществляться в файл students.

Написать функции, реализующие операции со структурами (ввод данных с клавиатуры):

1. Создание новой записи о студенте.
2. Внесение изменений в уже имеющуюся запись.
3. Вывод всех данных о студентах.
4. Вывод информации обо всех студентах группы N. N – инициализируется пользователем.
5. Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.
6. Вывод количества студентов мужского и женского пола.
7. Определение количества студентов, которые будут получать стипендию (стипендия начисляется, если у студента нет троек и очная форма обучения).
8. Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;
9. Вывод данных о студентах, имеющих номер в списке – k.
10. Вывод всех записей, сделанных в день, который введет пользователь. Вывод всех записей, сделанных после полудня. Вывод всех записей, сделанных до полудня.

Необходимо реализовать программу, которая выполняет следующие действия.

1. Формирование целочисленного одномерного массива размерности N, где:
  - а) пользователь вводит количество элементов в массиве, который будет автоматически заполняться случайными числами (0 до 99);
  - б) пользователь вводит в консоль элементы массива, N определяется автоматически по количеству введенных элементов;
  - в) массив считывается с файла, N определяется как количество элементов массива в файле.

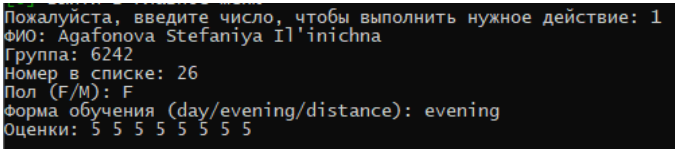
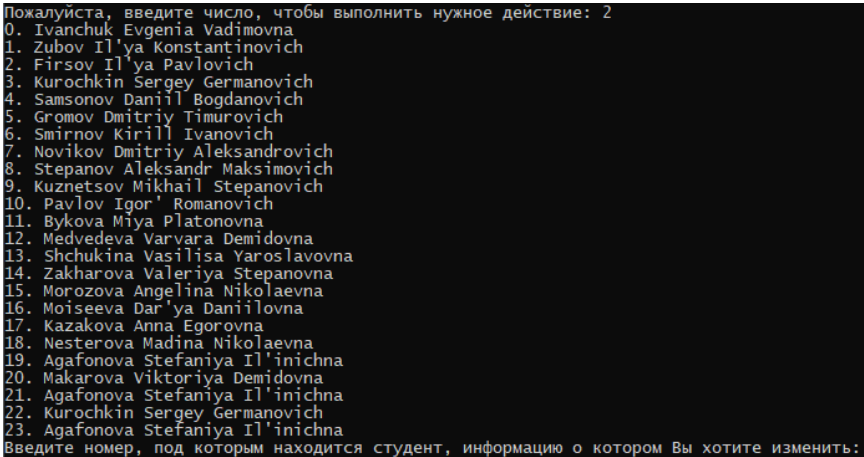
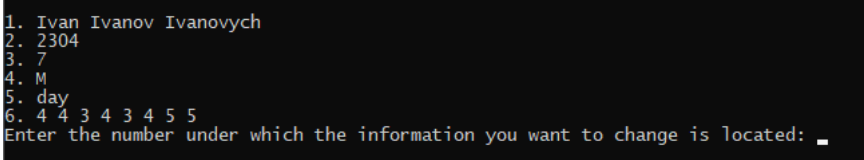
2. Определение скорости создания динамического массива п. 1.
3. Вставка, удаление и получение элемента массива. Удаление и получение элемента необходимо реализовать по индексу и по значению.
4. Определение скорости вставки, удаления и получения элемента массива п. 3.
5. Формирование двусвязного списка размерности  $N$ , где:
  - а) пользователь вводит количество элементов в списке, который будет автоматически заполняться случайными числами (0 до 99);
  - б) пользователь вводит в консоль элементы списка,  $N$  определяется автоматически по количеству введенных элементов;
  - в) список считывается с файла,  $N$  определяется как количество элементов списка в файле.
6. Определение скорости создания двусвязного списка п. 5.
7. Вставка, удаление и получение элемента двусвязного списка. Удаление и получение элемента необходимо реализовать по индексу и по значению.
8. Определение скорости вставки, удаления и получения элемента двусвязного списка п. 7.

Должна быть возможность запуска каждого пункта многократно, если есть возможность (если в списке/массиве нет элементов, то нельзя ничего удалить и об этом нужно сообщить пользователю). Необходимо сравнить результаты. Для этого пункты 1–4 и 5–8 должны принимать одинаковые значения.

## Выполнение работы.

Код программы представлен в приложении А.

Ввод пользователем и обработка данных	Работа алгоритма и вывод на экран
Меню	
При запуске программы перед пользователем появляется окно с главным меню, где он может перейти к интересующей его теме.	<p>Главное меню:</p> <p>Меню для работы со записями о студентах:</p> <pre>Выберите нужный раздел: [1] Создание новой записи о студенте [2] Изменить существующую запись [3] Отобразить все записи о студентах [4] Вывести данные о студентах, выполняющих поставленные условия [5] Вывести топ студентов по среднему баллу за сессию [6] Вывести число студентов, выполняющих поставленные условия [7] Вывести данные о студентах в зависимости от даты сделанной записи [8] Очистить экран консоли [0] Выйти в главное меню Пожалуйста, введите число, чтобы выполнить нужное действие:</pre> <p>Проверка на ввод символов, которые не входят в диапазон выбора:</p> <pre>[0] Выйти в главное меню Пожалуйста, введите число, чтобы выполнить нужное действие: ык Ошибка! Пожалуйста, попробуйте снова  Выберите нужный раздел: [1] Создание новой записи о студенте [2] Изменить существующую запись [3] Отобразить все записи о студентах [4] Вывести данные о студентах, выполняющих поставленные условия [5] Вывести топ студентов по среднему баллу за сессию [6] Вывести число студентов, выполняющих поставленные условия [7] Вывести данные о студентах в зависимости от даты сделанной записи [8] Очистить экран консоли [0] Выйти в главное меню Пожалуйста, введите число, чтобы выполнить нужное действие: 9 Ошибка! Пожалуйста, попробуйте снова</pre>

Создание новой записи	
При вводе пользователем корректного значения пункта меню и выбора создания новой записи, пользователь может добавить данные о студенте, но только на русском языке.	<p>Ввод информации выглядит следующим образом:</p> 
Внесение изменений в имеющуюся запись	
Если пользователь допустил ошибку в создании карточки студента или данные изменились, он может внести изменения по пунктам, которые были созданы ранее.	<p>При необходимости внесения изменений в запись о студенте, перед пользователем появляется меню:</p>  <p>Выбрав нужного ученика, на выбор пользователю дается список информации, которую он может поменять:</p>  <p>Как только пользователь вводит корректное значение, его ответ обрабатывается и предлагается изменить существующие данные на новые. После пользователь возвращается (при необходимости) в главное меню.</p>

Продолжение Таблицы

Вывод всей информации о студентах	
Данное действие позволяет вывести всех студентов, находящихся в списке.	<p>Пожалуйста, введите число, чтобы выполнить нужное действие: 3</p> <p>Ivanchuk Evgenia Vadimovna</p> <p>9894</p> <p>6</p> <p>F</p> <p>evening</p> <p>3 4 5 5 5 5 5</p> <p>Sat Feb 29 20:05:44 2020</p> <p>Zubov Il'ya Konstantinovich</p> <p>5801</p> <p>2</p> <p>M</p> <p>distance</p> <p>4 4 3 3 4 4 5 4</p> <p>Sun Feb 23 15:35:39 2020</p> <p>Firsov Il'ya Pavlovich</p> <p>7878</p> <p>3</p> <p>M</p> <p>evening</p> <p>5 5 4 5 3 3 4 3</p> <p>Sun Feb 23 23:35:39 2020</p> <p>Kurochkin Sergey Germanovich</p> <p>5866</p> <p>3</p> <p>M</p> <p>evening</p> <p>3 3 5 3 4 5 4 3</p> <p>Sun Feb 23 03:38:39 2020</p> <p>Samsonov Daniil Bogdanovich</p> <p>9894</p> <p>15</p> <p>M</p> <p>day</p> <p>3 3 5 5 4 4 5 4</p> <p>Tue Feb 25 00:14:46 2020</p> <p>Gromov Dmitriy Timurovich</p> <p>6242</p> <p>2</p> <p>M</p> <p>day</p> <p>3 3 5 4 5 5 5 4</p> <p>Tue Feb 25 03:26:46 2020</p> <p>Smirnov Kirill Ivanovich</p>

Продолжение Таблицы

Вывод всей информации о студентах в зависимости от условий	
Данные о студентах можно сортировать по номеру группы, по оценкам (хорошисты и отличники), по номеру в списке своей группы.	<p>Если студент с нужным номером не найден:</p> <p>Введите номер группы: 999 Таких студентов не было найдено</p> <p>Вывод информации:</p> <p>Введите номер группы: 6242 Gromov Dmitriy Timurovich 6242 2 M day 3 3 5 4 5 5 5 4 Tue Feb 25 03:26:46 2020 Smirnov Kirill Ivanovich 6242 15 M distance 5 4 4 4 5 5 5 4 Tue Feb 25 18:54:46 2020 Shchukina Vasilisa Yaroslavovna 6242 17 F evening 5 5 5 5 5 5 5 5 Fri Mar 6 10:33:57 2020</p>
Рейтинг студентов	
Пользователь в главном меню может выбрать такую опцию как рейтинг студентов, чтобы наглядно увидеть список студентов с минимальными данными и рейтингу их оценок.	<p>Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию</p> <ol style="list-style-type: none"> <li>1. Medvedeva Varvara Demidovna = 5</li> <li>2. Shchukina Vasilisa Yaroslavovna = 5</li> <li>3. Nesterova Madina Nikolaevna = 5</li> <li>4. Agafonova Stefaniya Il'inichna = 5</li> <li>5. Agafonova Stefaniya Il'inichna = 5</li> <li>6. Zakharova Valeriya Stepanovna = 4.75</li> <li>7. Ivanchuk Evgenia Vadimovna = 4.625</li> <li>8. Pavlov Igor' Romanovich = 4.625</li> <li>9. Bykova Miya Platonovna = 4.625</li> <li>10. Smirnov Kirill Ivanovich = 4.5</li> <li>11. Stepanov Aleksandr Maksimovich = 4.5</li> <li>12. Moiseeva Dar'ya Daniilovna = 4.5</li> <li>13. Kazakova Anna Egorovna = 4.375</li> <li>14. Gromov Dmitriy Timurovich = 4.25</li> <li>15. Kuznetsov Mikhail Stepanovich = 4.25</li> <li>16. Samsonov Daniil Bogdanovich = 4.125</li> </ol>



Информация о студентах в количестве	
Если пользователь хочет узнать количество студентов каждого пола или получивших стипендию.	<p>Подсчёт количества студентов разных полов:</p> <pre>Студентов женского пола: 11 Студентов мужского пола: 12</pre> <p>Получающих стипендию:</p> <pre>Всего студентов со стипендией: 4</pre>
Вывод информации о студентах в зависимости от времени сделанной записи	
Пользователь может отсортировать записи по времени их создания или последнего редактирования: он может ввести дату, чтобы получить записи, сделанные в этот день, получить записи, сделанные до полудня или после полудня	<p>Выбирая возможность получить записи в конкретный день, пользователь может ввести дату в определённом формате. Если записи были найдены, их список будет выведен:</p> <pre>Введите дату в формате mmm dd yyyy (например, Mar 6 2020): May 31 2021 Agafonova Stefaniya Il'ichna 6242 26 M evening 5 5 5 5 5 5 5 Mon May 31 21:29:25 2021 Kurochkin Sergey Germanovich 5866 3 M evening 3 3 5 3 4 5 4 3 Mon May 31 22:08:10 2021</pre> <p>В противном случае, пользователь получит сообщение о том, что таких записей не существует:</p> <pre>Введите дату в формате mmm dd yyyy (например, Mar 6 2020): Mar 6 2022 Таких записей не было найдено.</pre>

Ввод пользователем и обработка данных	Работа алгоритма и вывод на экран
Меню	
При запуске программы перед пользователем появляется окно с меню, где он может выбрать тип будущей вводимой последовательности.	<p>Меню:</p> <pre> Выберите нужный раздел: [1] Преобразование введённого выражения [2] Проверить выражение на корректность [3] Вычислить выражение [4] Очистить экран консоли [0] Выйти в главное меню Пожалуйста, введите число, чтобы выполнить нужное действие: 1  [1] Ввести обычное выражение и преобразовать в обратную польскую запись [2] Ввести обычное выражение и преобразовать в прямую польскую запись [3] Ввести обратную польскую запись и преобразовать в обычное выражение [4] Ввести прямую польскую запись и преобразовать в обычное выражение [3] Ввести обратную польскую запись и преобразовать в прямую польскую запись [4] Ввести прямую польскую запись и преобразовать в обратную польскую запись [0] Вернуться назад Пожалуйста, введите число, чтобы выполнить нужное действие: </pre>
Преобразование выражение из одной записи в другую	
<p>Пользователь может выбрать, из какой записи в какую преобразовать введённое им выражение. При этом, каждый раз его выражение будет проверяться на корректность.</p> <p>Все промежуточные действия преобразования выводятся пользователю на экран</p>	<pre> [1] Ввести обычное выражение и преобразовать в обратную польскую запись [2] Ввести обычное выражение и преобразовать в прямую польскую запись [3] Ввести обратную польскую запись и преобразовать в обычное выражение [4] Ввести прямую польскую запись и преобразовать в обычное выражение [3] Ввести обратную польскую запись и преобразовать в прямую польскую запись [4] Ввести прямую польскую запись и преобразовать в обратную польскую запись [0] Вернуться назад Пожалуйста, введите число, чтобы выполнить нужное действие: 1 Введите выражение: 2222 Ошибка! Некорректное выражение [0] Вернуться назад Пожалуйста, введите число, чтобы выполнить нужное действие: 1 Введите выражение: 15 / (7 - (1 + 1)) * 3 - (2 + (1 + 1)) Рассматриваем символ 1 Символ 1 является числом Считали число 15 Оно сразу попадает в строку вывода Текущая строка вывода: 15 Рассматриваем символ / Символ / является операцией Символ / имеет приоритет 2 Текущий символ попадает в стек Переходим к следующему символу... Переходим к следующему символу... Текущая строка вывода: 15 7 1 1 + - / 3 * 2 1 1 + + Если в стеке что-то осталось, выводим все операции. Преобразованное выражение: 15 7 1 1 + - / 3 * 2 1 1 + + - </pre>

Проверка на корректность	
Чтобы проверить выражение на корректность, не преобразовывая его, пользователь может воспользоваться для этого специальной функцией	<p>Выберите, что хотите сделать:</p> <ul style="list-style-type: none"> <li>[1] Проверить на корректность простого выражения</li> <li>[2] Проверить на корректность выражения в обратной польской записи</li> <li>[3] Проверить на корректность выражения в прямой польской записи</li> <li>[0] Вернуться назад</li> </ul> <p>Пожалуйста, введите число, чтобы выполнить нужное действие: 1 Введите выражение, которое нужно проверить на корректность: 2+2 Следующее выражение является правильным: 2+2</p> <p>В случае, если пользователь ввёл некорректное выражение, он узнает почему:</p> <p>Выберите, что хотите сделать:</p> <ul style="list-style-type: none"> <li>[1] Проверить на корректность простого выражения</li> <li>[2] Проверить на корректность выражения в обратной польской записи</li> <li>[3] Проверить на корректность выражения в прямой польской записи</li> <li>[0] Вернуться назад</li> </ul> <p>Пожалуйста, введите число, чтобы выполнить нужное действие: 2 Введите выражение, которое нужно проверить на корректность: 2 + 2 Выражение должно иметь в начале два числа или переменные Выражение содержит ошибку</p>
Вычисление выражений	
Пользователь также может вычислить выражение в любой форме. Промежуточные действия будут также выведены на экран	<p>Выберите, что хотите сделать:</p> <ul style="list-style-type: none"> <li>[1] Вычислить обычное выражение</li> <li>[2] Вычислить выражение в обратной польской записи</li> <li>[3] Вычислить выражение в прямой польской записи</li> <li>[0] Вернуться назад</li> </ul> <p>Пожалуйста, введите число, чтобы выполнить нужное действие: 1 Введите выражение, которое нужно вычислить: 2 + 2 Рассматриваем символ 2 Символ 2 является числом Считали число 2 Оно сразу попадает в строку вывода Текущая строка вывода: 2 Рассматриваем символ + Символ + является операцией Символ + имеет приоритет 1 Текущий символ попадает в стек Переходим к следующему символу... Текущая строка вывода: 2 Рассматриваем символ 2 Символ 2 является числом Считали число 2 Оно сразу попадает в строку вывода Текущая строка вывода: 2 2 Если в стеке что-то осталось, выводим все операции. Рассматриваем символ 2 Символ 2 является числом Получаем число 2. Оно попадает в стек Рассматриваем символ 2 Символ 2 является числом Получаем число 2. Оно попадает в стек Рассматриваем символ + Символ + является операцией Вытащили второе выражение из стека 2 Вытащили первое выражение из стека 2 Сложили и получили число 4 Добавляем его в стек Рассматриваем символ Ответ: 4</p>

## **Выводы.**

Были изучены и применены структуры на практике.

Разработана программа, способная выводить и записывать данные, введенные пользователем. Также программа способна осуществлять сортировку введенных данных по параметрам, определяемые пользователем.

Были получены практические навыки работы со стеками и очередями; изучены обратная и прямая польская нотации; проведен сравнительный анализ этих структур данных.

## **ПРИЛОЖЕНИЕ А КОД ПРОГРАММЫ**

Название файла: cw.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <ctime>
#include <time.h>
#pragma warning(disable : 4996)
using namespace std;

struct Profile // а н к е т а   с т у д е н т а
{
    string fullName;
    char sex;
    unsigned short int group;
    unsigned short int numberList;
    int term[8];
    char depart[9];
    string date; // Д а т а   в н е с е н и я   и з м е н н е н и я   в   з а п и
с ь (post/update)
};

// у д а л е н и е   л и ш н и х   п р о б е л о в   ( д л я   д а т )
string DelSpaces(string s)
{
    for (int j = 0; j < s.length(); j++)
    {
        if (s[j] == ' ')
        {
            while (s[j + 1] == ' ') s.erase(j + 1, 1);
        }
    }
    if (s[0] == ' ') s.erase(0, 1);
}
```

```

        if (s[s.length() - 1] == ' ') s.erase(s.length() - 1, 1);
        return s;
    }

//считывание с экрана информации о студенте
void newStudent()
{
    Profile Student;
    cin.clear();
    char trash;
    cout << "ФИО: ";
    cin >> trash;
    getline(cin, Student.fullName);
    Student.fullName = trash + Student.fullName;
    cout << "Г р у п п а: ";
    cin >> Student.group;
    cout << "Н о м е р   в   с п и с к е: ";
    cin >> Student.numberList;
    cout << "П о л   (F/M): ";
    cin >> Student.sex;
    cout << "Ф о р м а   о б у ч е н и я   (day/evening/distance): ";
    cin >> Student.depart;
    cout << "О ц е н к и: ";
    bool temp = false; //имеются ли двойки?
    for (int i = 0; i < 8; i++)
    {
        cin >> Student.term[i];
        if (Student.term[i] == 2) { temp = true; } //да, двойка
имеется
    }

    //запоминание даты считывания
    struct tm* localtime;
    time_t curtime;
    time(&curtime);
    localtime = localtime(&curtime);
    Student.date = asctime(localtime);
    Student.date = DelSpaces(Student.date);

    if (temp) { cout << '\n' << "Этот студент будет иск
лючен. Профиль не будет сохранен в базе дан
ных."; }
    else
    {
        /*Попытка создать файл с введенными д
анными*/
        ofstream fout("students.txt", ios_base::app);
        if (!fout.is_open()) { cout << '\n' << "Ошибкa сохрa
нения!"; }
    }
}

```

```

else
{
    /* == Вывод записи == */
    fout << Student.fullName << "\n";
    fout << Student.group << "\n";
    fout << Student.numberList << "\n";
    fout << Student.sex << "\n";
    fout << Student.depart << "\n";
    for (int i = 0; i < 8; i++)
    {
        fout << Student.term[i] << " ";
    }
    fout << "\n";
    fout << Student.date;
    fout.close();
}
}

// Функция подсчёта количества студентов
int countStudents()
{
    ifstream fin("students.txt");
    if (fin.is_open())
    {
        int temp = 0; //количество строк
        string data;
        while (!fin.eof()) //пока указатель потока не
        достигнет конца файла
        {
            getline(fin, data); //считывается строка
            temp++;
        }
        fin.close();
        int n;
        n = temp / 7; //количество строк поделить
на кол-во строк одной анкеты студента = кол-
во анкет студента
        return n;
    }
    else return 0;
}

//вывод файла всей информации о студентах
void outputStudents()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) // если файл не открыт
        cout << "Файл не был открыт\n"; // сообщить
о б э т о м

```

```

else
{
    int temp;
    temp = countStudents();
    if (temp == 0)
        cout << "Ф а й л   п у с т \n";
    else
    {
        string data; // б у ф е р   п р о м е ж у т о ч н о г о   х р
а н е н и я   с ч и т ы в а е м о г о   и з   ф а й л а   т е к с т а
        while (!fin.eof())
        {
            getline(fin, data); // С ч и т ы в а е м   о ч е р е д
н у ю   с т р о ч к у
            cout << data << '\n'; // В ы в о д и м   с т р о к у
н а   э к р а н
        }
        fin.close();
    }
}

// п о д с ч ё т   к о л и ч е с т в а   с т у д е н т о в   р а з н ы х   п о л о
в
void F_and_M()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "О ш и б к а !\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "Б а з а   д а н н ы х   п у с т а .\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            for (int i = 0; i < size; i++) // С ч и т ы в а е м   д а н
н ы е   в с е х   с т у д е н т о в   в   м а с с и в   с т р у к т у р
            {
                getline(fin, student[i].fullName, '\n');
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;
                fin >> student[i].depart;
                for (int j = 0; j < 8; j++) {
                    fin >> student[i].term[j];
                }
                fin >> trash;
                getline(fin, student[i].date, '\n');
            }
        }
    }
}

```

```

    }
    fin.close();

    //подсчёт и вывод
    int f = 0,
        m = 0;
    for (int i = 0; i < size; i++)
    {
        if (student[i].sex == 'F') { f++; }
        if (student[i].sex == 'M') { m++; }
    }
    cout << "\nСтудентов женского пола: "
<< f << "\n";
    cout << "Студентов мужского пола: " <<
m << "\n";

    //конец подсчёта и вывода

    delete[] student;
}
}

//Количество студентов со стипендией
void stipend()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Ошибка!\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "База данных пуста.\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool three; /*наличие троек
            int s = 0; /*количество студентов со
стипендией
            for (int i = 0; i < size; i++) // Считываем дан
ные всех студентов в массив структур
            {
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;

                fin >> student[i].depart;
                three = false; /*по умолчанию троек
нет
                for (int j = 0; j < 8; j++) {

```



```

        fin >> student[i].term[j];
        if (student[i].term[j] == 3) { three = true; }
/*если встречена тройка, запоминаем это
    }
    fin >> trash;
    getline(fin, student[i].date);
    student[i].date = trash + student[i].date;
    if (!(three) && (student[i].depart[1] == 'a')) {
s++; } /*подсчёт студентов со стипендией
    }
    fin.close();

    cout << "\nВсего студентов со стипенд
ией: " << s << '\n'; /*вывод

    delete[] student;
}
}
}

//Вывод данных о студентах без стипендии
void notStipend()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Ошибка!\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "База данных пуста.\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информа
цию. Стипендия не может быть получена
            for (int i = 0; i < size; i++) // Считываем дан
ные всех студентов в массив структур
            {
                check = false; /*по умолчанию false
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;
                fin >> student[i].depart;
                if (!(student[i].depart[1] == 'a')) { check = true;
} /*если не дневное обучение - стипендии нет
                for (int j = 0; j < 8; j++) {
                    fin >> student[i].term[j];

```

```

        if (student[i].term[j] == 3) { check = true; }
/*если встречена тройка, запоминаем это
    }
    fin >> trash;
    getline(fin, student[i].date);
    student[i].date = trash + student[i].date;

//Вывод
    if (check)
    {
        cout << student[i].fullName << "\n";
        cout << student[i].group << "\n";
        cout << student[i].numberList << "\n";
        cout << student[i].sex << "\n";
        cout << student[i].depart << "\n";
        for (int j = 0; j < 8; j++) { cout << stu-
dent[i].term[j] << " "; }
        cout << "\n" << student[i].date << "\n";
    }
//конец вывода
}
fin.close();

delete[] student;
}
}
}

//Вывод данных о студентах-отличниках
void excellentTerm()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Ошибка!\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "База данных пуста.\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информа
цию. Является отличником
            for (int i = 0; i < size; i++) // Считываем дан
ные всех студентов в массив структур
            {
                check = true; /*по умолчанию true
                getline(fin, student[i].fullName);

```

```

        fin >> student[i].group >> student[i].numberList >>
student[i].sex;
        fin >> student[i].depart;
        for (int j = 0; j < 8; j++) {
            fin >> student[i].term[j];
            if ((student[i].term[j] == 3) || (stu-
dent[i].term[j] == 4)) { check = false; } /*е с л и в с т р е ч е н а 3
и л и 4, з а п о м и н а е м э т о
        }
        fin >> trash;
        getline(fin, student[i].date);
        student[i].date = trash + student[i].date;

        // в ы в о д
        if (check)
        {
            cout << student[i].fullName << "\n";
            cout << student[i].group << "\n";
            cout << student[i].numberList << "\n";
            cout << student[i].sex << "\n";
            cout << student[i].depart << "\n";
            for (int j = 0; j < 8; j++) { cout << stu-
dent[i].term[j] << " "; }
            cout << "\n" << student[i].date << "\n";
        }
        //к о н е ц в ы в о д а
    }
    fin.close();

    delete[] student;
}

}

//В ы в о д д а н н ы х о х о р о ш и с т а х
void B_GradeTerm()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "О ш и б к а !\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "Б а з а д а н н ы х п у с т а .\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;

```

```

        bool check; /*true - надо вывести информацию. Является отличником
        for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур
        {
            check = true; /*по умолчанию true
            getline(fin, student[i].fullName);
            fin >> student[i].group >> student[i].numberList >>
student[i].sex;
            fin >> student[i].depart;
            for (int j = 0; j < 8; j++) {
                fin >> student[i].term[j];
                if (student[i].term[j] == 3) { check = false;
} /*если встречена 3, запоминаем это
            }
            fin >> trash; //без него почему-то ломается

            getline(fin, student[i].date);
            student[i].date = trash + student[i].date;

            //вывод
            if (check)
            {
                cout << student[i].fullName << "\n";
                cout << student[i].group << "\n";
                cout << student[i].numberList << "\n";
                cout << student[i].sex << "\n";
                cout << student[i].depart << "\n";
                for (int j = 0; j < 8; j++) { cout << stu-
dent[i].term[j] << " "; }
                cout << "\n" << student[i].date << "\n";
            }
            //конец вывода
        }
        fin.close();

        delete[] student;
    }
}

//Вывод данных о студентах группы N
void groupN(unsigned short int n)
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Ошибка!\n"; }
    else
    {
        int size;
        size = countStudents();

```

```

        if (size == 0) { cout << "База данных пуста.\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информа
цию. Является студентом группы N
            int k = 0;
            for (int i = 0; i < size; i++) // Считываем дан
ные всех студентов в массив структур
            {
                check = false; /*по умолчанию false
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;
                if (student[i].group == n) { check = true; }
                fin >> student[i].depart;
                for (int j = 0; j < 8; j++) { fin >> stu-
dent[i].term[j]; }
                fin >> trash;
                getline(fin, student[i].date);
                student[i].date = trash + student[i].date;

                // вывод
                if (check)
                {
                    cout << student[i].fullName << "\n";
                    cout << student[i].group << "\n";
                    cout << student[i].numberList << "\n";
                    cout << student[i].sex << "\n";
                    cout << student[i].depart << "\n";
                    for (int j = 0; j < 8; j++) { cout << stu-
dent[i].term[j] << " "; }
                    cout << "\n" << student[i].date << "\n";
                    k++;
                }
                //конец вывода
            }
            if (k == 0)
            {
                cout << "Таких студентов не было н
айдено\n";
            }
            fin.close();

            delete[] student;
        }
    }
}

```

```

//Вывод данных о студентах номера k
void numberListK(unsigned short int k)
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "О ш и б к а !\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "Б а з а д а н н ы х п у с т а .\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информа
цию. Является студентом номера k
            int n = 0;
            for (int i = 0; i < size; i++) // Считываем дан
ные всех студентов в массив структур
            {
                check = false; /*по умолчанию false
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;
                if (student[i].numberList == k) { check = true; }
                fin >> student[i].depart;
                for (int j = 0; j < 8; j++) { fin >> stu-
dent[i].term[j]; }
                fin >> trash;
                getline(fin, student[i].date);
                student[i].date = trash + student[i].date;

                //Вывод
                if (check)
                {
                    cout << student[i].fullName << "\n";
                    cout << student[i].group << "\n";
                    cout << student[i].numberList << "\n";
                    cout << student[i].sex << "\n";
                    cout << student[i].depart << "\n";
                    for (int j = 0; j < 8; j++) { cout << stu-
dent[i].term[j] << " "; }
                    cout << "\n" << student[i].date << "\n";
                    n++;
                }
                //конец вывода
            }
            if (n == 0)

```

```

        {
            cout << "Таких студентов не было н
а й д е н о \n";
        }
        fin.close();

        delete[] student;
    }
}

//Вывод данных до полудня
void tillNoon()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "О ш и б к а !\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "Б а з а д а н н ы х п у с т а .\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            bool check; /*true - надо вывести информа
цию. Запись сделана до 12:00
            int k = 0;
            for (int i = 0; i < size; i++) // Считываем дан
ные всех студентов в массив структур
            {
                check = false; /*по умолчанию false
                getline(fin, student[i].fullName);
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;

                fin >> student[i].depart;
                for (int j = 0; j < 8; j++) { fin >> stu-
dent[i].term[j]; }

                fin >> trash;
                getline(fin, student[i].date);
                student[i].date = trash + student[i].date;

                //получаем время, когда была сд
елана запись
                string date1 = student[i].date;
                char date[9];
                for (int i = 0; i < 8; i++)
                {
                    date[i] = date1[i + date1.size() - 13];

```

```

    }
    date[8] = '\0'; //получается время в
формате hh:mm:ss

    int hh = (int)date[0] - (int)'0'; //преобраз
уем из string в int, вырезая ненужное и готовя д
ля сравнения
    hh *= 10 + ((int)date[1] - (int)'0');
    if (hh < 12) { check = true; } //если запис
ь сделана раньше 12 часов, выводим

    //Вывод
    if (check)
    {
        cout << student[i].fullName << "\n";
        cout << student[i].group << "\n";
        cout << student[i].numberList << "\n";
        cout << student[i].sex << "\n";
        cout << student[i].depart << "\n";
        for (int j = 0; j < 8; j++) { cout << stu-
dent[i].term[j] << " "; }
        cout << "\n" << student[i].date << "\n";
        k++;
    }
    //конец вывода
}
if (k == 0)
{
    cout << "Таких записей не было най
дено.\n";
}
fin.close();

delete[] student;
}
}

//Вывод данных после полудня
void afterNoon()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Ошиб ка!\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "База данных пуста.\n" <<
endl; }
        else

```



```

{
    Profile* student = new Profile[size];
    char trash;
    bool check; /*true - надо вывести информа
цию. Запись сделана после 12:00
    int k = 0;
    for (int i = 0; i < size; i++) // Считываем дан
ные всех студентов в массив структур
    {
        check = false; /*по умолчанию false
        getline(fin, student[i].fullName);
        fin >> student[i].group >> student[i].numberList >>
student[i].sex;
        fin >> student[i].depart;
        for (int j = 0; j < 8; j++) { fin >> stu-
dent[i].term[j]; }
        fin >> trash;
        getline(fin, student[i].date);
        student[i].date = trash + student[i].date;

        //получаем время, когда была сд
елана запись
        string date1 = student[i].date;
        char date[9];
        for (int i = 0; i < 8; i++)
        {
            date[i] = date1[i + date1.size() - 13];
        }
        date[8] = '\0'; //получается время в
формате hh:mm:ss

        //часы
        int hh = (int)date[0] - (int)'0'; //преобраз
уем из string в int, вырезая ненужное и готовя д
ля сравнения
        hh *= 10 + ((int)date[1] - (int)'0');
        //минуты
        int mm = (int)date[3] - (int)'0';
        mm *= 10 + ((int)date[4] - (int)'0');
        //секунды
        int ss = (int)date[6] - (int)'0';
        ss *= 10 + ((int)date[7] - (int)'0');

        if (hh >= 12)
        {
            if (hh == 12) { if ((mm != 0) || (ss != 0)) {
check = true; } }
            else { check = true; }
        }
    }
}

```

```

        } //если запись сделана после 12
        часов, выводим

        //вывод
        if (check)
        {
            cout << student[i].fullName << "\n";
            cout << student[i].group << "\n";
            cout << student[i].numberList << "\n";
            cout << student[i].sex << "\n";
            cout << student[i].depart << "\n";
            for (int j = 0; j < 8; j++) { cout << stu-
dent[i].term[j] << " "; }
            cout << "\n" << student[i].date << "\n";
            k++;
        }
        //конец вывода
    }
    if (k == 0)
    {
        cout << "Таких записей не было най-
дено.\n";
    }
    fin.close();

    delete[] student;
}
}

void thatDate()
{
    cout << "\nВведите дату в формате mmm dd yyyy (на
пример, Mar 6 2020): ";
    string Day;
    char trash;
    cin >> trash;
    getline(cin, Day);
    Day = trash + Day;
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "Ошибка!\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "База данных пуста.\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];

```

```

char trash;
bool check; /*true - надо вывести информа
цию. Запись сделана в данный день
int k = 0;
for (int i = 0; i < size; i++) // Считываем дан
ные всех студентов в массив структур
{
    check = false; /*по умолчанию false
    getline(fin, student[i].fullName);
    fin >> student[i].group >> student[i].numberList >>
student[i].sex;
    fin >> student[i].depart;
    for (int j = 0; j < 8; j++) { fin >> stu-
dent[i].term[j]; }
    fin >> trash;
    getline(fin, student[i].date);
    student[i].date = trash + student[i].date;

    //сравниваем
    //24 символа, если дата с %dd, 23 ес
ли %d в полной дате
    //(Day).size(); 11 символов с %dd, 10 сим
олов с %d
    //при совпадении разница разме
ров должна давать 13. Отсеиваем при несовпад
ении
    int daySize = (Day).size();
    if (((student[i].date).size() - daySize) == 13)
    {
        string date1 = student[i].date;
        bool same = true; //проверяем на со
впадение месяцев и дней. По умолчанию true
        for (int i = 0; i < 6; i++)
        {
            if (Day[i] != date1[i + 4]) //у date1 н
адо пропустить первые 4 символа
            {
                same = false;
                break;
            }
        }
        for (int i = daySize - 1; i > 8; i--)
        {
            if (Day[i] != date1[i + 13])
            {
                same = false;
                break;
            }
        }
    }
}

```

```

        if (same) // е с л и м е с я ц а с о в п а л и
        {
            check = true;
        }
    }

    // в ы в о д
    if (check)
    {
        cout << student[i].fullName << "\n";
        cout << student[i].group << "\n";
        cout << student[i].numberList << "\n";
        cout << student[i].sex << "\n";
        cout << student[i].depart << "\n";
        for (int j = 0; j < 8; j++) { cout << stu-
dent[i].term[j] << " "; }
        cout << "\n" << student[i].date << "\n";
        k++;
    }
    // к о н е ц  в ы в о д а
}
if (k == 0)
{
    cout << "Т а к и х  з а п и с е й  н е  б ы л о  н а й
д е н о .\n";
}
fin.close();

delete[] student;
}
}

void topTerm()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "О ш и б к а !\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "Б а з а  д а н н ы х  п у с т а .\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            float* term = new float[size]; /*м а с с и в  с о с р е
д н и м и  о ц е н к а м и
            char trash;

```

```

        for (int i = 0; i < size; i++) // Считываем дан
ные всех студентов в массив структур
        {
            getline(fin, student[i].fullName, '\n');
            fin >> student[i].group >> student[i].numberList >>
student[i].sex;

            fin >> student[i].depart;
            term[i] = 0;
            for (int j = 0; j < 8; j++) {
                fin >> student[i].term[j];
                term[i] += (float)(student[i].term[j]);
            }
            term[i] /= 8;
            fin >> trash;
            getline(fin, student[i].date, '\n');
        }
        fin.close();

//сортировка и вывод
for (int i = 0; i < size; i++)
{
    for (int j = 0; j < size - 1; j++)
    {
        if (term[j] < term[j + 1])
        {
            float m = term[j];
            term[j] = term[j + 1];
            term[j + 1] = m;

            string name = student[j].fullName;
            student[j].fullName = student[j +
1].fullName;

            student[j + 1].fullName = name;
        }
    }
}

//вывод
for (int i = 0; i < size; i++)
{
    cout << i + 1 << ". " << student[i].fullName << " =
" << term[i] << "\n";
}

//конец подсчёта и вывода
delete[] term;
delete[] student;
}
}
}

```

```

void changeFile()
{
    ifstream fin("students.txt");
    if (!fin.is_open()) { cout << "О ш и б к а !\n"; }
    else
    {
        int size;
        size = countStudents();
        if (size == 0) { cout << "Б а з а д а н н ы х п у с т а .\n" <<
endl; }
        else
        {
            Profile* student = new Profile[size];
            char trash;
            for (int i = 0; i < size; i++) // С ч и т ы в а е м д а н
н ы е в с е х с т у д е н т о в в м а с с и в с т р у к т у р
            {
                getline(fin, student[i].fullName, '\n');
                fin >> student[i].group >> student[i].numberList >>
student[i].sex;
                fin >> student[i].depart;
                for (int j = 0; j < 8; j++) {
                    fin >> student[i].term[j];
                }
                fin >> trash;
                getline(fin, student[i].date, '\n');
                student[i].date = trash + student[i].date;
            }
            fin.close();

            //и з м е н е н и е и н ф о р м а ц и и
            for (int i = 0; i < size; i++)
            {
                cout << i << ". " << student[i].fullName << "\n";
            }
            cout << "В в е д и т е н о м е р , п о д к о т о р ы м н
а х о д и т с я с т у д е н т , и н ф о р м а ц и ю о к о т о р о м В ы х
о т и т е и з м е н и т ь : ";
            int numbStud;
            cin >> numbStud;
            if (numbStud >= size) { cout << "Т а к о й с т у д е н т
н е б ы л н а й д е н \n"; return; }
            cout << "\n1. " << student[numbStud].fullName << "\n";
            cout << "2. " << student[numbStud].group << "\n";
            cout << "3. " << student[numbStud].numberList << "\n";
            cout << "4. " << student[numbStud].sex << "\n";
            cout << "5. " << student[numbStud].depart << "\n6. ";
            for (int j = 0; j < 8; j++) { cout << stu-
dent[numbStud].term[j] << " "; }

```

```

        cout << "\nВведите номер, под которым
находится информацию, которую Вы хотите из
менить: ";
        int sw;
        cin >> sw;

        bool check = false; //имеются ли изменения
?
        bool temp = false; //имеются ли двойки?
        switch (sw)
        {
        case 1:
            cout << "\nВведите новое ФИО студе
нта: ";
            cin >> trash;
            getline(cin, student[numbStud].fullName);
            student[numbStud].fullName = trash + stu-
dent[numbStud].fullName;
            check = true;
            break;
        case 2:
            cout << "\nВведите новый номер гру
ппы студента: ";
            cin >> student[numbStud].group;
            check = true;
            break;
        case 3:
            cout << "\nВведите новый номер сту
дента в списке: ";
            cin >> student[numbStud].numberList;
            check = true;
            break;
        case 4:
            cout << "\nВведите пол студента
(F/M): ";
            cin >> student[numbStud].sex;
            check = true;
            break;
        case 5:
            cout << "\nВведите новый формат об
учения студента (day/evening/distance): ";
            cin >> student[numbStud].depart;
            check = true;
            break;
        case 6:
            cout << "\nВведите оценки студента
: ";
            for (int i = 0; i < 8; i++)
            {

```

```

        cin >> student[numbStud].term[i];
        if (student[numbStud].term[i] == 2) { temp =
true; } //да, двойка имеется
    }
    if (temp) { cout << "У студента имеются
я неудовлетворительные оценки, поэтому он
будет удалён.\n"; }
    check = true;
    break;
default:
    cout << "\nВы ничего не выбрали. Пр
оизводится выход из редактора.\n";
}
//изменения внесены

//вводим в файл
if (check)
{
    /*Попытка создать файл с введён
ными данными*/
    ofstream fout("students.txt");
    if (!fout.is_open()) { cout << '\n' << "Ошибк а
сохранения!\n"; }
    else
    {
        for (int i = 0; i < size; i++)
        {

            if (i == numbStud)
            {
                if (!temp) {
                    /* == Вывод записи ==

*/
                    fout << student[i].fullName <<
"\n";
                    fout << student[i].group <<
"\n";
                    fout << student[i].numberList
<< "\n";
                    fout << student[i].sex <<
"\n";
                    fout << student[i].depart <<
"\n";
                    for (int j = 0; j < 8; j++)
                    {
                        fout << stu-
dent[i].term[j] << " ";
                    }
                    fout << "\n";

```



```

        //запоминание даты
        struct tm* localtime;
        time_t curtime;
        string dateChange;
        time(&curtime);
        localtime = localtime(&curtime);
        dateChange = asctime(localtime);
        dateChange = Del-

Spaces(dateChange);

        fout << dateChange;
    }
    cout << "Информация была
успешно изменена.\n";
    }
    else
    {
        /* == Вывод записи == */
        fout << student[i].fullName <<
"\n";

        fout << student[i].group << "\n";
        fout << student[i].numberList <<
"\n";

        fout << student[i].sex << "\n";
        fout << student[i].depart << "\n";
        for (int j = 0; j < 8; j++)
        {
            fout << student[i].term[j] <<
" ";

        }
        fout << "\n";
        fout << student[i].date;
        fout << "\n";
    }
}

    }
    fout.close();
}
//конец вывода в файл
delete[] student;
}

}

int lb1()
{
    bool check = true; //выход из меню

```

```

bool check1 = false; //выход из подменю
bool outp = false;
//false - заканчивает цикл, приводя непосредственно к выходу
do {
    //system("cls");
    char sw = ' '; //переключатель главного меню

    char sw1 = ' '; //переключатель саб-меню
    cout << "\nВыберите нужный раздел: \n";
    cout << "\x1b[32m[1]\x1b[0m Создание новой записи о студенте\n";
    cout << "\x1b[32m[2]\x1b[0m Изменить существующую запись \n";
    cout << "\x1b[32m[3]\x1b[0m Отобразить все записи о студентах\n";
    cout << "\x1b[32m[4]\x1b[0m Вывести данные о студентах, выполняющих поставленные условия\n";
    cout << "\x1b[32m[5]\x1b[0m Вывести топ студент ов по среднему баллу за сессию\n";
    cout << "\x1b[32m[6]\x1b[0m Вывести число студентов, выполняющих поставленные условия\n";
    cout << "\x1b[32m[7]\x1b[0m Вывести данные о студентах в зависимости от даты сделанной записи\n";
    cout << "\x1b[32m[8]\x1b[0m Очистить экран консоли\n";
    cout << "\x1b[32m[0]\x1b[0m Выйти в главное меню\n";

    cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

    cin >> sw;
    while (cin.get() != '\n') { sw = ' '; }; //если строка содержит более одного символа, возвращается ошибка

    switch (sw)
    {

    case '1': //[1] новая запись о студенте
        newStudent();
        break;

    case '2': //[2] изменение записей
        changeFile();
        break;

```

```

        case '3': //[3] вывод всех данных
            outputStudents();
            break;

        case '4': //[4] данные о студентах, выполня
ющих поставленные условия
            do {
                check1 = false;
                sw1 = ' ';
                cout << "\nВыберите, что хотите сде
лать: \n";

                cout << "\x1b[32m[1]\x1b[0m Вывод информа
ции о студентах группы N\n";
                cout << "\x1b[32m[2]\x1b[0m Вывод информа
ции о студентах, которые не получают стипенд
ию\n";
                cout << "\x1b[32m[3]\x1b[0m Вывод информа
ции о студентах, которые имеют только оценок
и \"4\" и \"5\"\n";
                cout << "\x1b[32m[4]\x1b[0m Вывод информа
ции о студентах, которые имеют только оценок
и \"5\"\n";
                cout << "\x1b[32m[5]\x1b[0m Вывод информа
ции о студентах с номером k в списке\n";
                cout << "\x1b[32m[0]\x1b[0m Вернуться на з
ад\n";

                cout << "Пожалуйста, введите числ
о, чтобы выполнить нужное действие: ";

                cin >> sw1;
                while (cin.get() != '\n') { sw1 = ' '; };

                switch (sw1)
                {
                    case '1': //[1] вывод данных студент
ов группы N
                        unsigned short int n;
                        cout << "\nВведите номер группы
: ";

                        cin >> n;
                        groupN(n);
                        break;
                    case '2': //[2] вывод данных студент
ов без стипендии
                        notStipend();

```

```

        break;
    case '3': //[3] вывод данных студент
ов-хорошистов
        B_GradeTerm();
        break;
    case '4': //[4] вывод данных студент
ов-отличников
        excellentTerm();
        break;
    case '5': //[5] вывод данных студент
ов номера k
        unsigned short int k;
        cout << "\nВведите номер студен
та в списке: ";
        cin >> k;
        numberListK(k); //
        break;
    case '0': //[0] Назад
        break;
    default:
        cout << "Ошибка! Пожалуйста, по
пробуйте снова\n";
        check1 = true; //цикл пойдёт заново
        break;
    }
} while (check1);
break;

    case '5': //[5] вывод топа студентов по сре
дней оценке
        topTerm();
        break;

    case '6': //[6] число студентов, выполняющ
их поставленные условия
        do {
            check1 = false;
            sw1 = ' ';
            cout << "\nВыберите, что хотите сде
лать: \n";
            cout << "\x1b[32m[1]\x1b[0m Количество ст
удентов мужского и женского пола\n";
            cout << "\x1b[32m[2]\x1b[0m Количество ст
удентов, получающих стипендию\n";
            cout << "\x1b[32m[0]\x1b[0m Вернуться на з
ад\n";

```

```

        cout << "П о ж а л у й с т а , в в е д и т е ч и с л
о , ч т о б ы в ы п о л н и т ь н у ж н о е д е й с т в и е : ";

        cin >> sw1;
        while (cin.get() != '\n') { sw1 = ' '; };

        switch (sw1)
        {
        case '1': //[1] к о л и ч е с т в о  F и  M
            F_and_M();
            break;
        case '2': //[2] к о л и ч е с т в о  с т у д е н т о в
с о с т и п е н д и е й
            stipend();
            break;
        case '0': //[0] Н а з а д
            break;
        default:
            cout << "О ш и б к а ! П о ж а л у й с т а , п о
п р о б у й т е с н о в а \n";
            check1 = true; //ц и к л  п о й д ё т з а н о
в о
            break;
        }
    } while (check1);
    break;

        case '7': //[7] д а н н ы е  о  с т у д е н т а х  в  з а в и с и
м о с т и  о т  д а т ы
        do {
            check1 = false;
            sw1 = ' ';
            cout << "\nВ ы б е р и т е , ч т о х о т и т е с д е
л а т ь : \n";

            cout << "\x1b[32m[1]\x1b[0m В ы в о д  з а п и с е й
, с д е л а н н ы х  в  у к а з а н н ы й  д е н ь \n";
            cout << "\x1b[32m[2]\x1b[0m В ы в о д  з а п и с е й
, с д е л а н н ы х  п о с л е  п о л у д н я \n";
            cout << "\x1b[32m[3]\x1b[0m В ы в о д  з а п и с е й
, с д е л а н н ы х  д о  п о л у д н я \n";
            cout << "\x1b[32m[0]\x1b[0m В е р н у т ь с я  н а з
а д \n";

            cout << "П о ж а л у й с т а , в в е д и т е ч и с л
о , ч т о б ы в ы п о л н и т ь н у ж н о е д е й с т в и е : ";

            cin >> sw1;
            while (cin.get() != '\n') { sw1 = ' '; };

            switch (sw1)

```

```

        {
            case '1': //[1] вывод всех записей, с
деланных в этот день
                thatDate();
                break;
            case '2': //[2] записи после полудня
                afterNoon();
                break;
            case '3': //[2] записи до полудня
                tillNoon();
                break;
            case '0': //[0] Назад
                break;
            default:
                cout << "Ошибка! Пожалуйста, по
пробуйте снова\n";
                check1 = true; //цикл пойдёт заново
                break;
        }
    } while (check1);
    break;

    case '8': //[8] Очистить экран
        system("cls");
        break;

    case '0': //[0] Закрыть программу
        cout << "Выход из программы...\n";
        check = false; //выход из цикла
        break;
    default: //в случае, если введено что-то и
ное
        cout << "Ошибка! Пожалуйста, попробуйте
снова\n";
        break;
    }
} while (check);

system("Pause");
return 0;
}

// ДИНАМИЧЕСКИЙ ОДНОМЕРНЫЙ ЦЕЛОЧИСЛЕННЫЙ МАССИВ
// == Прочие действия ==
// Вывод массива
void outputArr(int*& arr, int size)
{

```

```

    if (size <= 0) { cout << "О ш и б к а !\n"; return; }
    for (int i = 0; i < size; i++) { cout << arr[i] << ' '; }
    cout << '\n';
}

// Создание массива размера N. Данные - случайные числа от 0 до 99
int* createArr(int N)
{
    if (N <= 0) { cout << "О ш и б к а !\n"; return NULL; }
    int* arr = new int[N]();
    for (int i = 0; i < N; i++)
    {
        arr[i] = rand() % 100; //заполнение случайным
        числом от 0 до 99
    }
    return arr;
}

// == ФУНКЦИИ ПО ПУНКТАМ ==
// пункт 1.а по массивам: N случайных чисел
int taskArr1a(int*& arr, bool time, int repeat)
{
    clock_t start, end; //таймер
    int n;
    cout << "Введите количество элементов нового массива: ";
    while (!(cin >> n) || (n <= 0)) //проверка на корректность ввода
    {
        cout << "О ш и б к а ! Н о в ы й м а с с и в н е б ы л с о з д а н !\n";
        cin.clear();
        cin.sync();
        while (cin.get() != '\n');
        cout << "Введите количество элементов нового массива: ";
    }

    if (time) { start = clock(); } //начало таймера
    for (int r = 0; r < repeat; r++) { if (arr) { delete[] arr; } arr = createArr(n); }
    if (time) { end = clock(); } //конец таймера
    cout << "Создан новый массив: ";
    outputArr(arr, n);
    if (time) { cout << "Время выполнения (только создание): " << ((double)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
    return n;
}

```

```

}

// пункт 1.b по массивам: произвольный размер
int taskArr1b(int*& arr, bool time)
{
    clock_t start, end;
    int x, size = 1;
    cout << "Введите элементы нового массива: ";
    cin >> x;
    if (!cin) { cout << "Ошибка! Новый массив не был
создан\n"; }
    else
    {
        if (time) { start = clock(); } //начало таймера
        if (arr) { delete[] arr; }
        arr = createArr(1);
        arr[0] = x;
        while (cin && cin.get() != '\n') //считываются все
значения, пока не встречен мусор
        {
            int* tmp;
            cin >> x;
            if (cin)
            {
                size++;
                tmp = (int*)realloc(arr, size * sizeof(int));
                if (tmp != NULL) { arr = tmp; arr[size - 1] = x; }
            }
        }
        if (time) { end = clock(); } //конец таймера
        cout << "Создан новый массив: ";
        outputArr(arr, size);
        cout << "Длина массива - " << size << " элемент
тов.\n";
        if (time) { cout << "Время выполнения (только
создание): " << ((double)end - start) / ((double)CLOCKS_PER_SEC)
<< '\n'; }
        return size;
    }
    return 0;
}

```

```

// пункт 1.c по массивам: произвольный размер
с файла
int taskArr1c(int*& arr, bool time, int repeat)
{
    clock_t start, end;
    int x, size = 1;
    ifstream fin("numbers.txt");

```



```

        if (!fin.is_open()) { cout << "Ф а й л   н е   б ы л   н а й д е н .\n";
    } // е с л и   ф а й л   н е   о т к р ы т ,   с о о б щ и т ь   о б   э т о м
        else
        {

                fin >> x;
                if (!fin) { cout << "О ш и б к а !   Н о в ы й   м а с с и в   н е
б ы л   с о з д а н \n"; }
                else
                {

                        if (time) { start = clock(); } //н а ч а л о   т а й м е р
а

                        if (arr) { delete[] arr; }
                        arr = createArr(1);
                        arr[0] = x;
                        while (fin && fin.get() != '\n') //с ч и т ы в а ю т с я
в с е   з н а ч е н и я ,   п о к а   н е   в с т р е ч е н   м у с о р
                        {

                                int* tmp;
                                fin >> x;
                                if (fin)
                                {

                                        size++;
                                        tmp = (int*)realloc(arr, size * sizeof(int));
                                        if (tmp != NULL) { arr = tmp; arr[size - 1] =
x; }

                                }

                        }
                        if (time) { end = clock(); } //к о н е ц   т а й м е р а
                        cout << "С о з д а н   н о в ы й   м а с с и в : ";
                        outputArr(arr, size);
                        cout << "Д л и н а   м а с с и в а   - " << size << "   э л е
м е н т о в .\n";
                        if (time) { cout << "В р е м я   в ы п о л н е н и я   (т о л
ь к о   с о з д а н и е): " << ((double)end - start) / ((dou-
ble)CLOCKS_PER_SEC) << '\n'; }
                        return size;
                }
        }
    }
    return 0;
}

//3.а - в с т а в к а   н о в о г о   э л е м е н т а
void taskArr3a(int*& arr, bool time, int& size, int repeat)
{
    clock_t start, end;
    if (repeat < 1) { cout << "О ш и б к а !   М а с с и в   н е   б ы л   и з
м е н ё н !\n"; return; }

```

```

        if (size < 1) { cout << "О ш и б к а ! М а с с и в н е б ы л н а й
д е н !\n"; return; }
        int x;
        cout << "В в е д и т е з н а ч е н и е н о в о г о э л е м е н т а :
";
        cin >> x;
        if (!cin) { cout << "О ш и б к а ! М а с с и в н е б ы л и з м е н
ё н !\n"; cin.clear(); while (cin.get() != '\n'); }
        else
        {
            if (time) { start = clock(); } //н а ч а л о т а й м е р а

            for (int i = 0; i < repeat; i++)
            {
                int* tmp;
                size++;
                tmp = (int*)realloc(arr, size * sizeof(int));
                if (tmp != NULL) { arr = tmp; arr[size - 1] = x; }
            }

            if (time) { end = clock(); } //к о н е ц т а й м е р а
            cout << "С о з д а н н о в ы й м а с с и в : ";
            outputArr(arr, size);
            cout << "Д л и н а м а с с и в а - " << size << " э л е м е н
т о в .\n";
            if (time) { cout << "В р е м я в ы п о л н е н и я : " << ((dou-
ble)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
            return;
        }
        return;
    }
}

//3ItemInd - п о л у ч и т ь э л е м е н т п о и н д е к с у
void taskArr3ItemInd(int*& arr, bool time, int& size, int repeat)
{
    clock_t start, end;
    if (repeat < 1) { cout << "О ш и б к а ! К о л и ч е с т в о п о в т
о р е н и й н е м о ж е т б ы т ь м е н ь ш е 1!\n"; return; }
    if (size < 1) { cout << "О ш и б к а ! М а с с и в н е б ы л н а й
д е н !\n"; return; }
    int x;
    cout << "В в е д и т е i н d e x э л е м е н т а , к о т о р ы й х
о т и т е п о л у ч и т ь (н у м е р а ц и я э л е м е н т о в с н у л я
): ";
    cin >> x;
    if (!cin || x >= size || x < 0) { cout << "О ш и б к а ! Э л е м е н
т н е б ы л н а й д е н .\n"; cin.clear(); while (cin.get() != '\n'); }
    else
    {

```

```

        if (time) { start = clock(); } //начало таймера

        int* item = nullptr;
        for (int i = 0; i < repeat; i++) { item = &arr[x]; }

        if (time) { end = clock(); } //конец таймера
        if (item != nullptr) { cout << "Получен элемент "
<< *item << " с адресом " << item << '\n'; }
        if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
        return;
    }
    return;
}

//3Item - получить элемент по значению
void taskArr3Item(int*& arr, bool time, int& size, int repeat)
{
    clock_t start, end;
    if (repeat < 1) { cout << "Ошибка! Количество повт
орений не может быть меньше 1!\n"; return; }
    if (size < 1) { cout << "Ошибка! Массив не был най
ден!\n"; return; }
    int x;
    cout << "Введите значение элемента, который
хотите получить: ";
    cin >> x;
    if (!cin) { cout << "Ошибка! Некорректный ввод
.\n"; cin.clear(); while (cin.get() != '\n'); }
    else
    {
        if (time) { start = clock(); } //начало таймера

        int* item = nullptr;
        for (int r = 0; r < repeat; r++)
        {
            for (int i = 0; i < size; i++)
            {
                if (arr[i] == x) { item = &arr[i]; break; }
            }
            if (item == nullptr) { cout << "Не найдено\n";
break; }
        }

        if (time) { end = clock(); } //конец таймера
        if (item != nullptr) { cout << "Получен элемент "
<< *item << " с адресом " << item << '\n'; }
        if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
    }
}

```

```

    }
    return;
}

//3DelInd - удаление элемента по индексу
void taskArr3DelInd(int*& arr, bool time, int& size, int repeat)
{
    clock_t start, end;
    if (repeat < 1) { cout << "Ошибка! Количество повт
орений не может быть меньше 1!\n"; return; }
    if (size < 1) { cout << "Ошибка! Массив не был най
ден!\n"; return; }
    int x;
    cout << "Введите индекс, элемент которого х
отите удалить (нумерация элементов с нуля):
";
    cin >> x;
    if (!cin || x >= size || x < 0) { cout << "Ошибка! Элемен
т не был найден.\n"; cin.clear(); while (cin.get() != '\n'); }
    else
    {
        if (time) { start = clock(); } //начало таймера

        for (int r = 0; r < repeat; r++)
        {
            if (!cin || x >= size || x < 0) { cout << "Ошибка!
Повторение не удалось, так как такой элемент
не был найден.\n"; r = repeat; break; }
            for (int j = x; j < size; j++) { arr[j] = arr[j + 1]; }
//сдвиг влево на место удаляемого элемента
            int* tmp;
            size--;
            tmp = (int*)realloc(arr, size * sizeof(int));
            if (tmp != NULL) { arr = tmp; }
        }
        if (time) { end = clock(); } //конец таймера

        cout << "Новый массив: ";
        if (size) { outputArr(arr, size); }
        else { cout << "Массив пуст.\n"; }
        cout << "Длина массива - " << size << " элемент
тов.\n";

        if (time) { cout << "Время выполнения: " << ((dou-
ble)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
        return;
    }
    return;
}

```

```

//3Del - удаление элемента по значению
//Удаляется только 1 элемент, так как доступ
на функция повтора
void taskArr3Del(int*& arr, bool time, int& size, int repeat)
{
    clock_t start, end;
    if (repeat < 1) { cout << "Ошибка! Количество повт
орений не может быть меньше 1!\n"; return; }
    if (size < 1) { cout << "Ошибка! Массив не был най
ден!\n"; return; }
    int x;
    cout << "Введите значение элемента, который
хотите удалить: ";
    cin >> x;
    if (!cin) { cout << "Ошибка! Некорректный ввод
.\n"; cin.clear(); while (cin.get() != '\n'); }
    else
    {
        bool check; //найден ли элемент?
        if (time) { start = clock(); } //начало таймера

        for (int r = 0; r < repeat; r++)
        {
            check = false;
            for (int i = 0; i < size; i++)
            {
                if (arr[i] == x)
                {
                    for (int j = i; j < size; j++) { arr[j] =
arr[j + 1]; } //сдвиг влево на место удаляемого э
лемента

                    int* tmp;
                    size--;
                    tmp = (int*)realloc(arr, size * sizeof(int));
                    if (tmp != NULL) { arr = tmp; }
                    check = true;
                    break;
                }
            }
            if (!check) { cout << "Элемент не найден\n";
r = repeat; break; }
        }
        if (time) { end = clock(); } //конец таймера
        if (check)
        {
            cout << "Элемент удалён\n";
            cout << "Новый массив: ";
            if (size) { outputArr(arr, size); }
        }
    }
}

```

```

        else { cout << "М а с с и в   п у с т .\n"; }
        cout << "Д л и н а   м а с с и в а   - " << size << "   э л е
м е н т о в .\n\n";
    }

    if (time) { cout << "В р е м я   в ы п о л н е н и я : " << ((dou-
ble)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
    return;
}
return;
}

// Д В У С В Я З Н Ы Й   С П И С О К
struct list
{
    int data;
    list* tail;
    list* head;
};

// == П р о ч и е   д е й с т в и я ==
// В ы в о д   с п и с к а ,   н а ч и н а я   с   в в е д ё н н о г о   у з л а
void outputList(list* beg)
{
    if (!beg) { cout << "С п и с о к   н е   н а й д е н !"; return; }
    list* curr = beg;
    while (curr)
    {
        cout << curr->data << ' ';
        curr = curr->tail;
    }
    cout << '\n';
}

// О ч и щ е н и е   с п и с к а ,   н а ч и н а я   с   в в е д ё н н о г о   у з л а
void deleteList(list*& beg)
{
    list* Next;
    while (beg)
    {
        Next = beg->tail;
        delete beg;
        beg = Next;
    }
}

// Д л и н а   с п и с к а
int lenList(list* roster)
{

```

```

    size_t len = 0;
    while (roster)
    {
        len++;
        roster = roster->tail;
    }
    return len;
}

// == Создание списка размера N. Данные - случайные числа от 0 до 99 ==
list* createList(int N)
{
    if (N <= 0) { cout << "Ошибка!\n"; return NULL; }
    list* Curr = 0, //текущий элемент
        * Next = 0; //следующий
    for (int i = 0; i < N; i++) //заполнение списка с конца
    {
        Curr = new list; //новый элемент

        Curr->data = rand() % 100; //заполнение случайным числом от 0 до 99

        Curr->tail = Next; //в адресной части - следующий элемент
        if (Next) //если существует следующий элемент
        {
            Next->head = Curr;
        } //закрепляем прошлый узел с текущим
        Next = Curr; //переходим к следующему элементу
    }

    Curr->head = 0; //его предыдущий адрес должен отсылаться на NULL
    return Curr; //адрес последнего элемента возвращается как адрес первого
}

// == Заполнение списка данными. N определяет ся автоматически ==
//Добавление нового элемента к списку после указанного узла. Не добавляет элемент в начало списка
list* addItem(list* roster, int a)
{

```

```

list* temp;
temp = new list; //создание добавляемого узла
temp->data = a; //заполняем значение

temp->tail = roster->tail;
temp->head = roster;
if (roster->tail) { (roster->tail)->head = temp; }
roster->tail = temp;

return temp; //адрес добавленного узла
}

//доступ к заданному элементу списка (по индексу) для манипуляций с его информационной частью
list* itemList(list* beg, int index)
{
    //index--; //если индексация не с нуля
    while (beg && (index--))
    {
        beg = beg->tail;
        if (!beg) { return 0; } //элемент не существует
    }
    return beg;
}

//удаление узла из списка;
list* deleteItem(list* delItem, list* beg)
{
    list* prev, * next;
    prev = delItem->head; //элемент, предшествующий удаляемому узлу
    next = delItem->tail; //следующий элемент после удаляемого

    //если НЕ первый
    if (prev) { prev->tail = delItem->tail; }
    else //если в начале
    {
        delete delItem;
        if (next) { next->head = prev; return next; } //если следующий элемент существует - вернуть его как начало
        else { return NULL; } //если нет - список будет удалён
    }
    //если НЕ последний
    if (next) { next->head = delItem->head; }

```



```

        delete delItem;
        return beg;
    }

// == ФУНКЦИИ ПО ПУНКТАМ ==
// пункт 1.a по спискам: N случайных чисел
void taskList1a(list*& beg, bool time, int repeat)
{
    clock_t start, end;
    int n;
    cout << "Введите длину нового списка: ";
    while (!(cin >> n) || (n <= 0))
    {
        cout << "Ошибка! Новый список не был создан!\n";
        cin.clear();
        cin.sync();
        while (cin.get() != '\n');
        cout << "Введите длину нового списка: ";
    }
    if (beg) { deleteList(beg); }
    if (time) { start = clock(); }
    for (int r = 0; r < repeat; r++) { beg = createList(n); }
    if (time) { end = clock(); }
    cout << "Создан новый список: ";
    outputList(beg);
    if (time) { cout << "Время выполнения (только созданные): " << ((double)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
}

// пункт 1.b по спискам: произвольный размер
void taskList1b(list*& beg, bool time)
{
    clock_t start, end;
    int x;
    cout << "Введите элементы нового списка: ";
    cin >> x;
    if (!cin) { cout << "Ошибка! Некорректный ввод.\n"; cin.clear(); while (cin.get() != '\n'); }
    else
    {
        if (time) { start = clock(); }
        if (beg) { deleteList(beg); }
        beg = createList(1);
        beg->data = x;
        while (cin && cin.get() != '\n') //считываются все значения, пока не встречен мусор
        {

```

```

        cin >> x;
        if (cin)
        {
            addItem(beg, x);
            beg = beg->tail;
        }
    }
    if (time) { end = clock(); }
    while (beg->head) { beg = beg->head; }
    cout << "С о з д а н н о в ы й   с п и с о к : ";
    outputList(beg);
    cout << "Д л и н а   с п и с к а   - " << lenList(beg) << "   э л е
м е н т о в .\n";
    if (time) { cout << "В р е м я   в ы п о л н е н и я   (т о л ь к о
с о з д а н и е): " << ((double)end - start) / ((double)CLOCKS_PER_SEC)
<< '\n'; }
}

//п у н к т 1.с   п о   с п и с к а м :   п р о и з в о л ь н ы й   р а з м е р ,   с
ч и т ы в а н и е   с   ф а й л а
void taskList1c(list*& beg, bool time)
{
    clock_t start, end;
    int x;
    ifstream fin("numbers.txt");
    if (!fin.is_open()) { cout << "Ф а й л   н е   б ы л   н а й д е н .\n";
}
// е с л и   ф а й л   н е   о т к р ы т ,   с о о б щ и т ь   о б   э т о м
else
{
    fin >> x;
    if (!fin) { cout << "О ш и б к а !   Н е к о р р е к т н ы й   в в о
д .\n"; cin.clear(); while (cin.get() != '\n'); }
    else
    {
        if (time) { start = clock(); }
        if (beg) { deleteList(beg); }
        beg = createList(1);
        beg->data = x;
        while (fin && fin.get() != '\n') //с ч и т ы в а ю т с я
в с е   з н а ч е н и я ,   п о к а   н е   в с т р е ч е н   м у с о р
        {
            fin >> x;
            if (fin)
            {
                addItem(beg, x);
                beg = beg->tail;
            }
        }
        if (time) { end = clock(); }
    }
}

```

```

        while (beg->head) { beg = beg->head; }
        cout << "С о з д а н  н о в ы й  с п и с о к : ";
        outputList(beg);
        cout << "Д л и н а  с п и с к а  - " << lenList(beg) << "
элементов.\n";
        if (time) { cout << "В р е м я  в ы п о л н е н и я  ( т о л
ь к о  с о з д а н и е ) : " << ((double)end - start) / ((dou-
ble)CLOCKS_PER_SEC) << '\n'; }
        fin.close();
    }
}

//3.a - в с т а в к а  н о в о г о  э л е м е н т а  !!п о с л е  п е р в о г
о  э л е м е н т а
void taskList3a(list*& beg, bool time, int repeat)
{
    clock_t start, end;
    if (repeat < 1) { cout << "О ш и б к а !  С п и с о к  н е  б ы л  и з
м е н ё н !\n"; return; }
    if (!beg) { cout << "О ш и б к а !  С п и с о к  н е  б ы л  н а й д е
н !\n"; return; }
    int x;
    cout << "В в е д и т е  з н а ч е н и е  н о в о г о  э л е м е н т а :
";
    cin >> x;
    if (!cin) { cout << "О ш и б к а !  Н е к о р р е к т н ы й  в в о д
.\n"; cin.clear(); while (cin.get() != '\n'); }
    else
    {
        if (time) { start = clock(); } //н а ч а л о  т а й м е р а

        for (int i = 0; i < repeat; i++)
        {
            addItem(beg, x);
        }

        if (time) { end = clock(); } //к о н е ц  т а й м е р а
        cout << "С о з д а н  н о в ы й  с п и с о к : ";
        outputList(beg);
        cout << "Д л и н а  с п и с к а  - " << lenList(beg) << " э л е
м е н т о в .\n";
        if (time) { cout << "В р е м я  в ы п о л н е н и я : " << ((dou-
ble)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
        }
        return;
    }
}

```

```

//3ItemInd - получить элемент по индексу
void taskList3ItemInd(list*& beg, bool time, int repeat)
{
    clock_t start, end;
    if (repeat < 1) { cout << "Ошибка! Количество повт
орений не может быть меньше 1!\n"; return; }
    if (!beg) { cout << "Ошибка! Список не был найде
н!\n"; return; }
    int x;
    cout << "Введите индекс элемента, который х
отите получить (нумерация элементов с нуля
): ";
    cin >> x;
    if (!cin || x >= lenList(beg) || x < 0) { cout << "Ошибка! Эл
емент не был найден.\n"; cin.clear(); while (cin.get() !=
'\n'); }
    else
    {
        if (time) { start = clock(); } //начало таймера

        list* item = nullptr;
        for (int i = 0; i < repeat; i++) { item = itemList(beg, x); }

        if (time) { end = clock(); } //конец таймера
        if (item != nullptr) { cout << "Получен элемент "
<< item->data << " с адресом " << item << '\n'; }
        if (time) { cout << "Время выполнения: " << ((dou-
ble)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
    }
    return;
}

```

```

//3Item - получить элемент по значению
void taskList3Item(list*& beg, bool time, int repeat)
{
    clock_t start, end;
    if (repeat < 1) { cout << "Ошибка! Количество повт
орений не может быть меньше 1!\n"; return; }
    if (!beg) { cout << "Ошибка! Список не был найде
н!\n"; return; }
    int x;
    cout << "Введите значение элемента, который
хотите получить: ";
    cin >> x;
    if (!cin || x >= lenList(beg) || x < 0) { cout << "Ошибка! Эл
емент не был найден.\n"; cin.clear(); while (cin.get() !=
'\n'); }
    else

```

```

{
    bool check = false; //найден ли элемент?
    if (time) { start = clock(); } //начало таймера

    list* tmp;
    list* item = nullptr;
    for (int r = 0; r < repeat; r++)
    {
        tmp = beg;
        size_t i = 0;
        do
        {
            if (tmp->data == x) { item = tmp; break; }
            tmp = tmp->tail;
            i++;
        } while (tmp);
        if (item == nullptr) { cout << "Не найдено\n";
break; }
    }

    if (time) { end = clock(); } //конец таймера
    if (item != nullptr) { cout << "Получен элемент "
<< item->data << " с адресом " << item << '\n'; }
    if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
    }
    return;
}

//3DelInd - удаление элемента по индексу
void taskList3DelInd(list*& beg, bool time, int repeat)
{
    clock_t start, end;
    if (repeat < 1) { cout << "Ошибка! Количество повт
орений не может быть меньше 1!\n"; return; }
    if (!beg) { cout << "Ошибка! Список не был найде
н!\n"; return; }
    int x;
    cout << "Введите индекс элемента, который х
отите удалить (нумерация элементов с нуля):
";
    cin >> x;
    if (!cin || x >= lenList(beg) || x < 0) { cout << "Ошибка! Эл
емент не был найден.\n"; cin.clear(); while (cin.get() !=
'\n'); }
    else
    {
        if (time) { start = clock(); } //начало таймера

```

```

        for (int i = 0; i < repeat; i++)
        {
            if (itemList(beg, x)) { deleteItem(itemList(beg, x),
beg); }
            else { cout << "О ш и б к а ! Т а к о й э л е м е н т н е
б ы л н а й д е н .\n"; break; }
        }

        if (time) { end = clock(); } //к о н е ц т а й м е р а
        if (beg)
        {
            cout << "С о з д а н н о в ы й с п и с о к : ";
            outputList(beg);
            cout << "Д л и н а с п и с к а - " << lenList(beg) << "
э л е м е н т о в .\n";
        }
        else { cout << "С п и с о к б ы л у д а л ё н \n"; }
        if (time) { cout << "В р е м я в ы п о л н е н и я : " << ((dou-
ble)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
    }
    return;
}

```

```

//3Del - у д а л е н и е э л е м е н т а п о з н а ч е н и ю
void taskList3Del(list*& beg, bool time, int repeat)
{
    clock_t start, end;
    bool check = false; //у д а л ё н л и э л е м е н т
    if (repeat < 1) { cout << "О ш и б к а ! К о л и ч е с т в о п о в т
о р е н и й н е м о ж е т б ы т ь м е н ь ш е 1!\n"; return; }
    if (!beg) { cout << "О ш и б к а ! С п и с о к н е б ы л н а й д е
н !\n"; return; }
    int x;
    cout << "В в е д и т е з н а ч е н и е э л е м е н т а , к о т о р ы й
х о т и т е у д а л и т ь : ";
    cin >> x;
    if (!cin) { cout << "О ш и б к а ! Н е к о р р е к т н ы й в в о д
.\n"; cin.clear(); while (cin.get() != '\n'); }
    else
    {
        list* delItem;
        if (time) { start = clock(); } //н а ч а л о т а й м е р а

        for (int i = 0; i < repeat; i++)
        {
            delItem = beg;
            do {
                if (delItem->data == x)
                {

```

```

        beg = deleteItem(delItem, beg);
        check = true;
        break;
    }
    delItem = delItem->tail;
} while (delItem);
if (!check) { break; }
}

if (time) { end = clock(); } //к о н е ц   т а й м е р а
if (beg)
{
    if (check)
    {
        cout << "С о з д а н   н о в ы й   с п и с о к : ";
        outputList(beg);
        cout << "Д л и н а   с п и с к а   - " << lenList(beg)
<< "   э л е м е н т о в .\n";
    }
    else { cout << "Э л е м е н т   н е   б ы л   н а й д е н \n";
}

    }
    else { cout << "С п и с о к   б ы л   у д а л ё н \n"; }
    if (time) { cout << "В р е м я   в ы п о л н е н и я : " << ((dou-
ble)end - start) / ((double)CLOCKS_PER_SEC) << '\n'; }
}
return;
}

int lb2()
{
    bool time = true;
    int repeat = 1; //к о л и ч е с т в о   п о в т о р е н и й   д е й с т в
и я

    // Д И Н А М И Ч Е С К И Й   М А С С И В
    int size = 0;
    int* arr = new int[size]();

    // Д В У С В Я З Н Ы Й   С П И С О К
    list* beg = NULL;

    bool check = true; //в ы х о д   и з   м е н ю
    bool check1 = false; //в ы х о д   и з   п о д м е н ю
    bool outp = false;
    //false - з а к а н ч и в а е т   ц и к л ,   п р и в о д я   н е п о с р е
д с т в е н н о   к   в ы х о д у
    do {
        //system("cls");

```

```

char sw = ' '; //переключатель главного ме
ню
char sw1 = ' '; //переключатель саб-меню
cout << "\nВыберите нужный раздел: \n";
cout << "\x1b[32m[1]\x1b[0m Создание целочислен
ного одномерного массива\n";
cout << "\x1b[32m[2]\x1b[0m Работа с элементами
массива \n";
cout << "\x1b[32m[3]\x1b[0m Создание двусвязног
о списка\n";
cout << "\x1b[32m[4]\x1b[0m Работа с узлами двус
вязного списка\n";
cout << "\x1b[32m[5]\x1b[0m Включить/выключить
счётчик времени при выполнении какого-либо
действия. Состояние: ";
if (time) { cout << "\x1b[32mВключено\x1b[0m\n"; }
else { cout << "\x1b[33mВыключено\x1b[0m\n"; }
cout << "\x1b[32m[6]\x1b[0m Установить другое ч
исло повторов действия. Количество повторе
ний сейчас: " << repeat << '\n';
cout << "\x1b[32m[7]\x1b[0m Очистить экран конс
оли\n";
cout << "\x1b[32m[8]\x1b[0m Включить/выключить
отображение списка и массива. Состояние: ";
if (outp) { cout << "\x1b[32mВключено\x1b[0m\n"; }
else { cout << "\x1b[33mВыключено\x1b[0m\n"; }
cout << "\x1b[32m[0]\x1b[0m Выйти в главное меню
\n";
if (outp)
{
    cout << "Массив: ";
    if (size) { outputArr(arr, size); }
    else { cout << "Не создан.\n"; }
    cout << "Список: ";
    if (beg) { outputList(beg); }
    else { cout << "Не создан.\n"; }
}
cout << "Пожалуйста, введите число, чтоб
ы выполнить нужное действие: ";

cin >> sw;
while (cin.get() != '\n') { sw = ' '; }; //если строка
содержит более одного символа, возвращаетс
я ошибка

switch (sw)
{

```



```

        case '1': //[1] Создание целочисленного од
номерного массива
            do {
                check1 = false;
                sw1 = ' ';
                cout << "\n\x1b[32m[1]\x1b[0m Ввести колич
ество элементов в массиве, чтобы получить м
ассив со случайными числами (от 0 до 99)\n";
                cout << "\x1b[32m[2]\x1b[0m Ввести значен
ие элементов массива. Его размер определяе
тся автоматически\n";
                cout << "\x1b[32m[3]\x1b[0m Считать масси
в с файла\n";
                cout << "\x1b[32m[0]\x1b[0m Вернуться на з
ад\n";

                cout << "Пожалуйста, введите числ
о, чтобы выполнить нужное действие: ";

                cin >> sw1;
                while (cin.get() != '\n') { sw1 = ' '; };

                switch (sw1)
                {
                    case '1': //[1] Случайные числа
                        size = taskArr1a(arr, time, repeat);
                        break;
                    case '2': //[2] Заполнение массива
                        size = taskArr1b(arr, time);
                        break;
                    case '3': //[3] Заполнение массива с
файла
                        size = taskArr1c(arr, time, repeat);
                        break;
                    case '0': //[0] Назад
                        break;
                    default:
                        cout << "Ошибка! Пожалуйста, по
пробуйте снова\n";
                        check1 = true; //цикл пойдёт зано
во

                        break;
                }
            } while (check1);
            break;

        case '2': //[2] Работа с элементами массив
а

```

```

do {
    check1 = false;
    sw1 = ' ';
    cout << "\nВыберите, что хотите сде
лат ь: \n";

    cout << "\x1b[32m[1]\x1b[0m Вставить новы
й элемент в массив\n";
    cout << "\x1b[32m[2]\x1b[0m Удалить элеме
нт массива (по индексу)\n";
    cout << "\x1b[32m[3]\x1b[0m Удалить элеме
нт массива (по значению)\n";
    cout << "\x1b[32m[4]\x1b[0m Получить элем
ент массива по индексу\n";
    cout << "\x1b[32m[5]\x1b[0m Получить элем
ента массива по значению\n";
    cout << "\x1b[32m[0]\x1b[0m Вернуться на з
ад\n";

    cout << "Пожалуйста, введите числ
о, чтобы выполнить нужное действие: ";

    cin >> sw1;
    while (cin.get() != '\n') { sw1 = ' '; };

    switch (sw1)
    {
    case '1': //[1] Вставить новый элем
нт в массив
        taskArr3a(arr, time, size, repeat);
        break;
    case '2': //[2] Удалить элемент масс
ива (по индексу)
        taskArr3DelInd(arr, time, size, repeat);
        break;
    case '3': //[3] Удалить элемент масс
ива (по значению)
        taskArr3Del(arr, time, size, repeat);
        break;
    case '4': //[4] Получить элемент мас
сива по индексу
        taskArr3ItemInd(arr, time, size, repeat);
        break;
    case '5': //[5] Получить элемента ма
ссива по значению
        taskArr3Item(arr, true, size, repeat);
        break;
    case '0': //[0] Назад
        break;
    default:

```

```

        cout << "Ошибка! Пожалуйста, по
пробуйте снова\n";
        check1 = true; //цикл пойдёт заново
        break;
    }
} while (check1);
break;

case '3': //[3] Создание двусвязного списка
do {
    check1 = false;
    sw1 = ' ';
    cout << "\nВыберите, что хотите сделать: \n";

    cout << "\x1b[32m[1]\x1b[0m Ввести количество элементов в списке, чтобы получить список со случайными числами (от 0 до 99)\n";
    cout << "\x1b[32m[2]\x1b[0m Ввести значение узлов списка. Его размер определяется автоматически\n";
    cout << "\x1b[32m[3]\x1b[0m Считать список с файла\n";
    cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

    cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

    cin >> sw1;
    while (cin.get() != '\n') { sw1 = ' '; };

    switch (sw1)
    {
    case '1': //[1] Случайные числа
        taskList1a(beg, time, repeat);
        break;
    case '2': //[2] Заполнение списка
        taskList1b(beg, time);
        break;
    case '3': //[3] Заполнение списка с
        taskList1c(beg, time);
        break;
    case '0': //[0] Назад
        break;
    default:

```

```

        cout << "Ошибка! Пожалуйста, по
пробуйте снова\n";
        check1 = true; //цикл пойдёт заново
        break;
    }
} while (check1);
break;

case '4': //[4] Работа с узлами двусвязного
списка
do {
    check1 = false;
    sw1 = ' ';
    cout << "\nВыберите, что хотите сде
лать: \n";

    cout << "\x1b[32m[1]\x1b[0m Вставить новы
й элемент в список\n";
    cout << "\x1b[32m[2]\x1b[0m Удалить узел
списка (по индексу)\n";
    cout << "\x1b[32m[3]\x1b[0m Удалить узел
списка (по значению)\n";
    cout << "\x1b[32m[4]\x1b[0m Получить элем
ент списка по индексу\n";
    cout << "\x1b[32m[5]\x1b[0m Получить элем
ент списка по значению\n";
    cout << "\x1b[32m[0]\x1b[0m Вернуться на з
ад\n";

    cout << "Пожалуйста, введите числ
о, чтобы выполнить нужное действие: ";

    cin >> sw1;
    while (cin.get() != '\n') { sw1 = ' '; };

    switch (sw1)
    {
    case '1': //[1] Вставить новый элеме
нт в массив
        taskList3a(beg, time, repeat);
        break;
    case '2': //[2] Удалить элемент масс
ива (по индексу)
        taskList3DelInd(beg, time, repeat);
        break;
    case '3': //[3] Удалить элемент масс
ива (по значению)
        taskList3Del(beg, time, repeat);

```

```

        break;
        case '4': //[4] Получить элемент списка по индексу
            taskList3ItemInd(beg, time, repeat);
            break;
        case '5': //[5] Получить элемент списка по значению
            taskList3Item(beg, time, repeat);
            break;
        case '0': //[0] Назад
            break;
        default:
            cout << "Ошибка! Пожалуйста, попробуйте снова\n";
            check1 = true; //цикл пойдёт заново
            break;
    }
} while (check1);
break;

```

```

        case '5': //[5] Включить/отключить счётчик времени при выполнении какого-либо действия
            time = !time;
            if (time) { cout << "Счётчик времени теперь включён.\n"; }
            else { cout << "Счётчик времени теперь отключён.\n"; }
            break;

```

```

        case '6': //[6] Установить другое число повторений. По умолчанию равен 1
            do
            {
                int x = 0;
                check1 = false;
                cout << "Введите число повторений (выше нуля): ";

                cin >> x;
                while (cin.get() != '\n');

                if (!cin || x <= 0) {
                    cout << "Ошибка! Количество повторений не может быть меньше 1. Значение не было изменено.\n";
                    check1 = true;

```

```

        }
        else { repeat = x; }
    } while (check1);
    break;

    case '7': //[7] Очистить экран
        system("cls");
        break;

    case '8': //[5] Включить/отключить счётчик
времени при выполнении какого-либо действия
        outp = !outp;
        if (outp) { cout << "Отображение списка и
массива включено.\n"; }
        else { cout << "Отображение списка и ма
ссива отключено.\n"; }
        break;

    case '0': //[0] Закрыть программу
        cout << "Выход из программы...\n";
        check = false; //выход из цикла
        break;
    default: //в случае, если введено что-то и
ное
        cout << "Ошибка! Пожалуйста, попробуй
те снова\n";
        break;
    }

} while (check);

if (size) { delete[] arr; }
deleteList(beg);
system("Pause");
return 0;
}

#include <iostream>
#include <string>
using namespace std;

//удаляет лишние пробелы, введённые пользов
ателем
string DelSpaces(string s)
{
    for (size_t j = 0; j < s.length(); j++)
    {

```

```

        if (s[j] == ' ')
        {
            while (s[j + 1] == ' ') s.erase(j + 1, 1);
        }
    }
    if (s[0] == ' ') s.erase(0, 1);
    if (s[s.length() - 1] == ' ') s.erase(s.length() - 1, 1);
    return s;
}

//проверка, является ли символ цифрой
bool isNumber(char s)
{
    if ((s <= '9') && (s >= '0')) { return true; }
    return false;
}

//проверка, является ли текущий символ операцией. если да - возвращает его приоритет
int priorOperation(char symb)
{
    switch (symb) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '(':
            return -1;
        case ')':
            return -2;
        default:
            return 0; //не является операцией
    }
}

//структура стека для символов
struct stack
{
    string data;
    stack* prev = 0;
};

//создание (инициализирование) стека и добавление в него первого элемента
stack* init(string data) {
    stack* tmp = new stack; //создаём начало стека
    tmp->data = data;
    tmp->prev = 0;
}

```

```

        return tmp;
    }

//добавление элемента в существующий стек
stack* pushBack(stack* stk, string data) {
    stack* newHead = new stack;
    newHead->data = data;
    newHead->prev = stk;
    return newHead;
}

//добавление элементов в стек
void push(stack*& stk, string data) {
    if (!stk) { stk = init(data); } //если стек пуст
    else { stk = pushBack(stk, data); }
}

//вывод элементов из стека в строку. oper - раз
//ный вывод в зависимости от операции
string outputStk(stack*& head, int oper)
{
    //oper = 0 вывести весь стек
    //oper = -2 вывести стек до "(" уничтожая "("
    //oper = -1 вывести стек до ")" уничтожая ")"
    //oper = 1 вывести весь стек до "(" не уничтож
а я его
    //oper = 3 вывести верхний элемент
    string output = "\0";

    if (oper == 3)
    {
        output += head->data;
        head = head->prev;
    }
    else
    {
        while (head)
        {
            if (priorOperation(head->data[0]) > 0) //игнориру
ются скобки
            {
                output += head->data;
                output += " ";
            }
            head = head->prev;
            if (oper && head && ((head->data == "(") || (head->data
== ")")))) //если не весь стек, то до "(" или ")"
            {

```



```

        if (oper == -2 || oper == -1) { head = head->prev;
    } //уничтожается "("
        break; //выход из вывода
    }
}

return output; //в конце строки всегда должен б
ыть пробел
}

//обратная польская нотация
string polishNotation(string& expr)
{
    stack* operation = 0; //стек операций
    string push_symb = "\0";
    string result = "\0"; //обработанное выражение

    int k = 0; //индекс строки
    string strval = "\0";
    int prior = 0;

    do {
        if (!expr[k]) { break; }
        cout << "Рассматриваем символ " << expr[k] <<
"\n";
        prior = priorOperation(expr[k]);
        if (prior) //если является операцией/скоб
кой
        {
            cout << "Символ " << expr[k] << " является о
перацией\n";
            //Если подытожить:
            //Игнор если: "++-"
            //НЕ ИГНОР: ")", "*+"
            //Если ")" - текущий символ уничтожа
ется
            //Если не игнор "+" или "-", текущий с
имвол переходит в пустой стек

            switch (prior)
            {
                case -1: //все открывающиеся скобки -
сразу в стек операций
                    cout << "Символ " << expr[k] << " являет
ся открывающейся скобкой, поэтому он идёт в
стек\n";
                    push_symb = expr[k];

```

```

        push(operation, push_symb);
        break;
    case 2: //"*" и "/" - вывод стека, если пр
ошлый символ "*" или "/"
        cout << "Символ " << expr[k] << " имеет п
риоритет 2\n";
        if (operation && (priorOperation(operation-
>data[0]) == 2))
        {
            cout << "Приоритет прошлого си
мвола также равен 2, поэтому происходит вых
од символов из стека\n";
            result += outputStk(operation, 1);
        }
        cout << "Текущий символ попадает
в стек\n";

        push_symb = expr[k];
        push(operation, push_symb);
        break;
    case -2: //закрывается скобка - выход
операций до открывающейся скобки
        cout << "Символ " << expr[k] << " являет
ся закрывающейся скобкой, поэтому происходи
т выход всех операций до открывающейся ск
обки\n";

        result += outputStk(operation, -2);
        break;
    case 1:
        //перед помещением в стек опера
ций + и -, смотрится последняя операция в сте
ке

        //если у текущего символа приор
итет меньше или равно - освобождение стека.
Текущий символ помещается в стек
        cout << "Символ " << expr[k] << " имеет п
риоритет 1\n";
        if (operation && (priorOperation(operation-
>data[0]) >= 1))
        {
            result += outputStk(operation, 1);
            cout << "Приоритет прошлого си
мвола равен 1 или меньше, поэтому происходи
т выход символов из стека\n";
        }
        push_symb = expr[k];
        push(operation, push_symb); //если приорит
ет у текущего символа больше - в стек

```

```

        cout << "Текущий символ попадает
в стек\n";
        break;
    }
    k++;
    cout << "Переходим к следующему символу...\n";
}
else
{
    cout << "Символ " << expr[k] << " является числом\n";
    strval = "\0"; //очищение строки
    do { //пока текущий элемент НЕ является операцией
        strval += expr[k]; //сюда попадают ТОЛЬКО числа
        k++;
    } while (expr[k] && !priorOperation(expr[k]));

    cout << "Считали число " << strval << "\n";
    result += strval; //все числа попадают сразу же в строку выхода
    cout << "Оно сразу попадает в строку вывода\n";
    result += " "; //пробел после числа
}
cout << "Текущая строка вывода: " << result <<
"\n";
} while (expr[k]);

if (operation) { result += outputStk(operation, 0); } //освобождение всего стека, если там что-то есть
cout << "Если в стеке что-то осталось, выводим все операции. \n";

return result;
}

//Обратную польскую в обычное
string fromPolishNotation(string& expr)
{
    string result = "\0"; //обработанное выражение
    stack< numbs > = 0; //стек для чисел/выражений
    size_t k = 0; //индекс строки
    string strval = "\0"; //строка для чисел/выражений
    string tmp1 = "\0"

```

```

, tmp2 = "\\0";

//все числа - в стек
//как встречена операция - забираем числ
а, используем на них операцию
do {
    cout << "Рассматриваем символ " << expr[k] <<
"\n";
    if (priorOperation(expr[k])) //если является опер
ацией
    {
        cout << "Символ " << expr[k] << " является о
перацией\n";
        strval = "\\0"; //очищение строку
        tmp2 = outputStk(numbs, 3); //вытаскиваем вто
рое выражение
        cout << "Вытащили второе выражение и
з стека " << tmp2 << " \n";
        tmp1 = outputStk(numbs, 3); //вытаскиваем пер
вое выражение
        cout << "Вытащили первое выражение и
з стека " << tmp1 << " \n";
        if (expr[k + 1] && (expr[k] == '+' || expr[k] == '-')) {
strval += "("; } //если НЕ последнее действие, ста
вит скобки
        strval += tmp1 + expr[k] + tmp2;
        if (expr[k + 1] && (expr[k] == '+' || expr[k] == '-')) {
strval += ")"; }
        cout << "Получившееся выражение: " <<
strval << ". Добавляем в стек\n";
        push(numbs, strval); //Добавляем в стек пол
учившееся выражение
        k++; //переходим на следующий симво
л
    }
    else //если число или пробел
    {
        if (isNumber(expr[k]))
        {
            cout << "Символ " << expr[k] << " являет
ся числом\n";
            strval = "\\0"; //очищение строку
            while (!(expr[k] == ' ')) //пока текущий
элемент НЕ является пробелом
            {
                strval += expr[k]; //считываем числ
а
                k++;
            }
        }
    }
}

```

```

        }
        cout << "Считали число " << strval << ".
Оно идёт в стек\n";
        push(nums, strval); //все числа попада
ют в стек
    }
    k++;
}
} while (expr[k]);
result = outputStk(nums, 3);
return result;
}

//в прямую польскую нотацию
string directPolishNotation(string& expr)
{
    stack* operation = 0 //стек операций
    , * output = 0; //стек выхода
    string push_symb = "\0";
    string result = "\0"; //обработанное выражение

    int k = expr.length() - 1; //индекс строки
    string strval = "\0";
    int prior = 0;

    do {
        if (!expr[k]) { break; }
        cout << "Рассматриваем символ " << expr[k] <<
"\n";
        prior = priorOperation(expr[k]);
        if (prior) //если является операцией/скоб
кой
        {
            cout << "Символ " << expr[k] << " является о
перацией\n";
            push_symb = "\0";
            //Если подытожить:
            //Игнор если: "++-"
            //НЕ ИГНОР: "(", "*+"
            //Если "(" - текущий символ уничтожа
ется

            //Если не игнор "+" или "-", текущий с
имвол переходит в пустой стек

            switch (prior)
            {
                case -2: //все закрывающиеся скобки -
сразу в стек операций

```

```

        cout << "Символ " << expr[k] << " является
        ся закрывающейся скобкой. Помещаем в стек
        \n";

        push_symb = expr[k];
        push(operation, push_symb);
        break;
    case 2: //"*" и "/" - сразу в стек операци
    й

        cout << "Символ " << expr[k] << " имеет п
        риоритет 2. Помещаем в стек\n";
        push_symb = expr[k];
        push(operation, push_symb);
        break;
    case -1: //открывается скобка - выход
    операций до закрывающейся скобки
        cout << "Символ " << expr[k] << " является
        ся открывающейся скобкой. Вывод операций д
        о закрывающейся\n";
        result += outputStk(operation, -1);
        break;
    case 1:
        //перед помещением в стек опера
        ций + и -, смотрится последняя операция в сте
        ке

        //если у текущего символа приор
        итет меньше или равно - освобождение стека.
        Текущий символ помещается в стек
        cout << "Символ " << expr[k] << " имеет п
        риоритет 1\n";
        if (operation && (priorOperation(operation-
        >data[0]) >= 1))
        {
            cout << "Прошлая операция в ст
            еке имеет приоритет равный 1 или больше. Выв
            од операций\n";
            result += outputStk(operation, 1);
        }
        cout << "Символ " << expr[k] << " помеща
        ем в стек\n";
        push_symb = expr[k];
        push(operation, push_symb); //если приорит
        ет у текущего символа больше - в стек
        break;
    }
    k--;
}
else
{

```

```

        cout << "Символ " << expr[k] << " является ч
ислом\n";
        strval = "\0"; //очищение строки
        do { //пока текущий элемент НЕ являе
тся операцией
            strval += expr[k]; //сюда попадают ТОЛ
ЬКО числа (в обратной записи)
            k--;
        } while ((k > 0) && !priorOperation(expr[k]));

        push_symb = "\0";
        for (size_t i = strval.length(); i > 0; i--) { push_symb
+= strval[i - 1]; } //отражаем полученное число
        cout << "Получаем число " << push_symb << " и
помещаем в строку вывода\n";

        result += strval + " "; //число сразу в стек
выхода
    }
    cout << "Текущая строка вывода: " << result <<
"\n";
} while (k >= 0);

if (operation) { result += outputStk(operation, 0); } //освобо
ждение всего стека, если там что-то есть
    cout << "Освобождаем стек, если там что-то о
сталось. Текущая строка вывода: " << result <<
"\n";
    strval = result;
    result = "\0";
    cout << "Отражаем строку\n";
    for (size_t i = strval.length(); i > 0; i--) { result += strval[i -
1]; } //отражаем полученное выражение

    return result;
}

//Прямую польскую в обычное
string fromDirectPolishNotation(string& expr)
{
    string result = "\0"; //обработанное выражение
    stack* numbs = 0; //стек для чисел/выражений
    int k = expr.length() - 1; //индекс строки
    string strval = "\0" //строка для чисел/выражени
й
        , tmp1 = "\0"
        , tmp2 = "\0"
        , push_symb = "\0";

```

```

//все числа - в стек
//как встречена операция - забираем числ
а, используем на них операцию
do {
    cout << "Рассматриваем символ " << expr[k] <<
"\n";
    if (priorOperation(expr[k])) //если является опер
ацией
    {
        cout << "Символ " << expr[k] << " является о
перацией\n";
        strval = "\0"; //очищение строку
        tmp2 = outputStk(numbs, 3); //вытаскиваем вто
рое выражение
        cout << "Вытащили второе выражение и
з стека " << tmp2 << " \n";
        tmp1 = outputStk(numbs, 3); //вытаскиваем пер
вое выражение
        cout << "Вытащили первое выражение и
з стека " << tmp1 << " \n";
        if ((k != 0) && (expr[k] == '+' || expr[k] == '-')) {
strval += "("; } //если НЕ последнее действие, ста
вит скобки
        strval += tmp2 + expr[k] + tmp1;
        if ((k != 0) && (expr[k] == '+' || expr[k] == '-')) {
strval += ")"; }
        cout << "Получившееся выражение: " <<
strval << ". Добавляем в стек\n";
        push(numbs, strval); //Добавляем в стек пол
учившееся выражение
        k--; //переходим на следующий симво
л
    }
    else //если число или пробел
    {
        if (isNumber(expr[k]))
        {
            cout << "Символ " << expr[k] << " являет
ся числом\n";
            strval = "\0"; //очищение строки
            while (!(expr[k] == ' ')) //пока текущий
элемент НЕ является пробелом
            {
                strval += expr[k]; //считываем числ
а
                k--;
            }
        }
    }
}

```



```

        push_symb = "\0";
        for (size_t i = strval.length(); i > 0; i--) {
push_symb += strval[i - 1]; } //отражаем полученное чис
ло

        cout << "Считали число " << push_symb <<
". Оно идёт в стек\n";

        push(nums, push_symb); //все числа по па
дают в стек
        }
        k--;
    }
} while (k >= 0);
result = outputStk(nums, 3);
return result;
}

//инициализация переменных. возвращает 0, ес
ли всё ок
bool initVar(string& expr)
{
    char sw = '\0';
    for (size_t i = 0; i < expr.length() - 1; i++)
    {
        if (!priorOperation(expr[i]) && !isNumber(expr[i])) //если
не является операцией и не является числом
        {
            cout << "Найден неизвестный символ: "
<< expr[i] << ". Инициализировать его как перемен
ную? (Y/N)\n";
            cin >> sw;
            while (cin.get() != '\n') { sw = ' '; }; //если стр
ока содержит более одного символа, возвращ
ается ошибка

            switch (sw)
            {
            case 'N':
                return 1;
            case 'Y':
                string toReplace = "\0"
                , replaceWith = "\0";
                toReplace = expr[i];
                cout << "Введите значение перемен
ной " << expr[i] << ": ";
                getline(cin, replaceWith);
                for (size_t j = 0; j < replaceWith.length() - 1;
j++)

```

```

        {
            if (!isNumber(replaceWith[j])) //если в с
тречено не число
            {
                cout << "Значение переменн
ой не может содержать иные символы кроме ци
фр.\n";

                return 1;
            }
        }
        size_t pos = 0;
        while ((pos = expr.find(toReplace, pos)) !=
string::npos)
        {
            expr.replace(pos, toReplace.size(), replace-
With);

            pos += replaceWith.size();
        }
        i = 0;
        break;
    }
}
return 0;
}

//проверка на корректность ввода и обработк
а выражения. возвращает 1, если найдена ошиб
ка в вводе, 0 если всё ок
bool invalidInput(string& expr)
{
    if (priorOperation(expr[0]) > 0) { cout << "Выражение не
может начинаться операцией\n"; return 1; } //выра
жение не может иметь операцию в начале
    if (priorOperation(expr[expr.length() - 1]) > 0) { cout << "Выра
жение не может заканчиваться операцией\n";
return 1; } //выражение не может иметь операцию
в конце

    //удаление всех пробелов
    for (size_t j = 1; j < expr.length() - 1; j++) //пробел не м
ожет стоять между числами
    {
        if (expr[j] == ' ')
        {
            if (!(priorOperation(expr[j - 1]) || priorOpera-
tion(expr[j + 1]))) //если пробелом разделены два ч
исла
            {

```

```

        cout << "Выражение содержит лишние
символы\n"; return 1; //ввод некорректный
    }
    expr.erase(j, 1); //если пробел разделяет
число и операцию - стираем пробел
    }
}

int opBrace = 0,
    cBrace = 0;
bool op = false;
//добавляет "*" там, где его не хватает. Пр
оверяет корректность потенциальных перемен
ных
for (size_t i = 0; i < expr.length(); i++)
{
    if (!expr[i]) { break; }
    if (!(isNumber(expr[i]) || priorOperation(expr[i]))) //не я
вляется числом или операцией (т.е. потенциа
льная переменная)
    {
        //работаем со следующим символом
        if (expr[i + 1] != '\0') //если он существуе
т
        {
            if (expr[i] == expr[i + 1]) { cout << "Выраже
ние содержит лишние символы\n"; return 1; } //если
две одинаковой переменной подряд - ошибка
            if (isNumber(expr[i + 1])) { cout << "Выражен
ие содержит лишние символы\n"; return 1; } //если
следующий символ число - ошибка
            if (!(priorOperation(expr[i + 1]) && expr[i + 1] !=
'(')) //если является операцией и не ")" - ничег
о не делаем
            {
                expr.insert(i + 1, "*"); //если "(" или
другая переменная - вставляем "*"
            }
        }

        //работа с предыдущим символом
        if (i > 0) //если он существует
        {
            if (!(priorOperation(expr[i - 1]) && expr[i - 1] !=
')')) //если является операцией и не ")" - ничег
о не делаем
            {

```

```

        expr.insert(i, "*"); //если число ил
и ")" - вставляем "*"
    }
}
//таким образом:
// x5 xx - ошибка
// x* x+ *x +x x) (x - игнор
// xb )x x( 5x - превращается в x*b )*x x*(
5*x
}

if (isNumber(expr[i])) //является числом
{
    //работа со следующим символом || 2(
превращается в 2*(
    if (expr[i + 1] != '\0') //если он существуе
т
    {
        if (expr[i + 1] == '(') { expr.insert(i + 1, "*");
} //если "(" - вставляем "*"
    }

    //работа с предыдущим символом ||
)2 превращается в )*2
    if (i > 0) //если он существует
    {
        if (expr[i - 1] == ')') { expr.insert(i, "*"); } //
если ")" - вставляем "*"
    }
}

if (priorOperation(expr[i]) > 0) //если является о
перацией, не скобки
{
    op = true; //запоминаем, что в выражен
ии есть операция
    //работа со следующим символом
    if (expr[i + 1] != '\0') //если он существуе
т
    {
        if (priorOperation(expr[i + 1]) > 0) { cout << "Д
ве операции не могут находиться рядом\n"; re-
turn 1; } //если две операции рядом - ошибка
        if (priorOperation(expr[i + 1]) == ')') { cout << "
Неправильный порядок скобок\n"; return 1; } //есл
и после операции ")" - ошибка. пример: *)
    }
}

```

```

        // работа со следующим символом
        if (i > 0) // если он существует
        {
            if (priorOperation(expr[i - 1]) == '(') { cout << "
Неправильный порядок скобок\n"; return 1; } // если
и после операции "(" - ошибка. пример: (*
        }
    }

    if (expr[i] == '(')
    {
        opBrace++; // подсчёт "("
        // внутри скобок должны быть как ми
нимум одна операция с двумя переменными/чи
слами

        size_t j = i + 1;
        bool check = false;
        while (expr[j] != '\0') // поиск символа - опе
рации
        {
            switch (priorOperation(expr[j]))
            {
                case -1:
                case 0: // если не операция или "(",
скип
                    j++;
                    break;
                case 1:
                case 2: // если встречена операция
+ - * /, запоминаем
                    check = true;
                    break;
                case -2: // если ")" встречена раньше
других операций - ошибка
                    cout << "Неправильный порядок
скобок\n"; return 1;
            }
            if (check) { break; }
        }
    }
    if (expr[i] == ')')
    {
        cBrace++; // подсчёт ")"
        if (cBrace > opBrace) { cout << "Неправильный
порядок скобок\n"; return 1; } // если закрывающая
ся скобка встречена раньше открывающейся -
ошибка
    }
}

```

```

        if (!op) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; } //если
        в выражении нет операций, это ошибка
        if (opBrace != cBrace) { cout << "Количество операций не соответствует количеству чисел\n"; return
1; } //если количество скобок не совпадает

        if (initVar(expr)) { return 1; }

        return 0;
    }

//Проверка на корректность написания обратной польской
bool invalidInputPolish(string& expr)
{
    //Первые два объекта - числа или переменные
    size_t j = 0;
    while (expr[j] != ' ')
    {
        if (priorOperation(expr[j])) { cout << "Выражение должно иметь в начале два числа или переменные\n"; return 1; }
        j++;
        if (j == expr.length()) { cout << "Количество операций не соответствует количеству чисел\n";
return 1; } //если не найдено пробела или операции
    }
    j++;
    while (expr[j] != ' ')
    {
        if (priorOperation(expr[j])) { cout << "Выражение должно иметь в начале два числа или переменные\n"; return 1; }
        j++;
        if (j == expr.length()) { cout << "Количество операций не соответствует количеству чисел\n";
return 1; } //если не найдено пробела или операции
    }

    //Всегда заканчивается на операцию
    if (!priorOperation(expr[expr.length() - 1])) { cout << "Выражение должно заканчиваться операцией\n"; return
1; }

```

```

//Количество операций = количество чисел
- 1
size_t oper = 0
    , numbs = 0;
for (size_t i = 0; i < expr.length(); i++)
{
    if (priorOperation(expr[i])) { oper++; }
    else //не должно быть переменной+число,
т.е. 5x x5 sg5d 3436f3
    {
        if (expr[i] != ' ') //пробел пропускаем
        {
            if (!isNumber(expr[i])) //если встречена
потенциальная переменная
            {
                if (expr[i + 1] != ' ') { cout << "Выраже
ние имеет лишние символы\n"; return 1; } //последн
ее должен быть пробел
                numbs++; //если всё ок, считаем
ее за число
            }
            else //если встречено число
            {
                j = i + 1;
                while (expr[j] != ' ') //проверяем, н
ет ли внутри числа лишнего до пробела
                {
                    if (!isNumber(expr[i])) { cout << "Выр
ажение имеет лишние символы\n"; return 1; }
                    j++;
                }
                i = j; //пропускаем все символ
ы до пробела
                numbs++;
            }
        }
    }
}
if (oper != (numbs - 1)) { cout << "Количество операц
ий не соответствует количеству чисел\n"; re-
turn 1; }

if (initVar(expr)) { return 1; }

return 0;
}

//Проверка прямой польской
bool invalidInputDirectPolish(string& expr)

```

```

{
    //Последние два объекта - числа или переменные
    size_t j = expr.length() - 1;
    while (expr[j] != ' ')
    {
        if (priorOperation(expr[expr.length() - 1])) { cout << "Выражение должно иметь в конце два числа или переменные\n"; return 1; }
        j--;
        if (j == 0) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; } //если не найдено пробела или операции
    }
    j--;
    while (expr[j] != ' ')
    {
        if (priorOperation(expr[expr.length() - 1])) { cout << "Выражение должно иметь в конце два числа или переменные\n"; return 1; }
        j--;
        if (j == 0) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; } //если не найдено пробела или операции
    }

    //Всегда начинается на операцию
    if (!priorOperation(expr[0])) { cout << "Выражение должно начинаться с операции\n"; return 1; }

    //Количество операций = количество чисел
- 1
    size_t oper = 0
        , numbs = 0;
    for (size_t i = 0; i < expr.length(); i++)
    {
        if (priorOperation(expr[i])) { oper++; }
        else //не должно быть переменной+число,
т.е. 5x x5 sg5d 3436f3
        {
            if (expr[i] != ' ') //пробел пропускаем
            {
                if (!isNumber(expr[i])) //если встречена
потенциальная переменная
                {
                    if (expr[i + 1] != ' ') { cout << "Выражение содержит лишние символы\n"; return 1; } //после неё должен быть пробел

```



```

        numbs++; //если всё ок, считаем
её за число
    }
    else //если встречено число
    {
        j = i + 1;
        while (j < expr.length() && expr[j] != ' ') //
проверяем, нет ли внутри числа лишних симво
лов до пробела
        {
            if (!isNumber(expr[i])) { cout << "Выр
ажение содержит лишние символы\n"; return 1; }
            j++;
        }
        i = j; //пропускаем все символ
ы до пробела
        numbs++;
    }
}
}
if (oper != (numbs - 1)) { cout << "Количество операц
ий не соответствует количеству чисел\n"; re-
turn 1; }

if (initVar(expr)) { return 1; }

return 0;
}

//Вычисление в обратной
float calculate(string expr)
{
    float result = 0; //обработанное выражение
    stack< numbs = 0; //стек для чисел
    size_t k = 0; //индекс строки
    string strval = "\0"; //строка для чисел/выражени
й
    float tmp1 = 0
        , tmp2 = 0;

    //все числа - в стек
    //как встречена операция - забираем числ
а, используем на них операцию
    do {
        cout << "Рассматриваем символ " << expr[k] <<
"\n";
        if (priorOperation(expr[k])) //если является опер
ацией

```

```

        {
            cout << "Символ " << expr[k] << " является о
перацией\n";
            strval = "\0"; //очищение строку
            tmp2 = stof(outputStk(nums, 3)); //вытаскиваем
второе выражение
            cout << "Вытащили второе выражение и
з стека " << tmp2 << " \n";
            tmp1 = stof(outputStk(nums, 3)); //вытаскиваем
первое выражение
            cout << "Вытащили первое выражение и
з стека " << tmp1 << " \n";
            switch (expr[k])
            {
                case '+':
                    result = tmp1 + tmp2;
                    cout << "Сложили и получили число
" << result << " \n";
                    break;
                case '-':
                    result = tmp1 - tmp2;
                    cout << "Вычли и получили число " <<
result << " \n";
                    break;
                case '*':
                    result = tmp1 * tmp2;
                    cout << "Умножили и получили числ
о " << result << " \n";
                    break;
                case '/':
                    result = tmp1 / tmp2;
                    cout << "Разделили и получили чис
ло " << result << " \n";
                    break;
            }
            strval = to_string(result);
            cout << "Добавляем его в стек \n";
            push(nums, strval); //Добавляем в стек пол
учившееся выражение
            k++; //переходим на следующий символ
        }
        else //если число или пробел
        {
            if (isNumber(expr[k]))
            {
                cout << "Символ " << expr[k] << " являет
ся числом\n";

```

```

        strval = "\0"; //очищение строку
        while (!(expr[k] == ' ')) //пока текущий
элемент НЕ является пробелом
        {
            strval += expr[k]; //считываем числ
а
            k++;
        }
        cout << "Получаем число " << strval <<
". Оно попадает в стек\n";
        push(nums, strval); //все числа попада
ют в стек
    }
    k++;
}
} while (expr[k]);

result = stof(outputStk(nums, 3)); //вытаскиваем полу
чившееся выражение
return result;
}

// == ФУНКЦИИ ПО ПУНКТАМ ==
//Обычное выражение в обратную польскую нот
ацию
string toPolishNotation()
{
    string expr; //выражение пользователя
    string result; //результат преобразования
    //Ввод выражения двумя способами: с клави
атуры и * с файла
    cout << "Введите выражение: ";
    getline(cin, expr);
    expr = DelSpaces(expr); //удаляет пробелы в начале
и в конце, пробелы, идущие подряд

    if (!invalidInput(expr))
    {
        result = polishNotation(expr);
        cout << "Преобразованное выражение: " <<
result << '\n';
    }
    else { cout << "Ошибка! Некорректное выражени
е\n"; return "\0"; }

    //После обработанного выражения:

    return result;
}

```

```

}

//Обратную польскую в обычное
string fromPolishNotationToInfix()
{
    string expr; //выражение пользователя
    string result; //результат преобразования
    //Ввод выражения двумя способами: с клави
и а т у р ы и * с ф а й л а
    cout << "Введите выражение: ";
    getline(cin, expr);
    expr = DelSpaces(expr); //удаляет пробелы в начале
и в конце, пробелы, идущие подряд

    if (!invalidInputPolish(expr))
    {
        result = fromPolishNotation(expr);
        cout << "Преобразованное выражение: " <<
result << '\n';
    }
    else { cout << "Ошибка! Некорректное выражени
е\n"; return "\0"; }

    //После обработанного выражения:
    return result;
}

//Обычное выражение в прямую польскую нотац
ию
string toDirectPolishNotation()
{
    string expr; //выражение пользователя
    string result; //результат преобразования
    //Ввод выражения двумя способами: с клави
и а т у р ы и * с ф а й л а
    cout << "Введите выражение: ";
    getline(cin, expr);
    expr = DelSpaces(expr); //удаляет пробелы в начале
и в конце, пробелы, идущие подряд

    if (!invalidInput(expr))
    {
        result = directPolishNotation(expr);
        cout << "Преобразованное выражение: " <<
result << "\n";
    }
    else { cout << "Ошибка! Некорректное выражени
е\n"; return "\0"; }
}

```

```

        //П о с л е о б р а б о т а н н о г о   в ы р а ж е н и я :

        return result;
    }

//П р я м у ю   п о л ь с к у ю   в   о б ы ч н о е
string fromDirectPolishNotationToInfix()
{
    string expr; //в ы р а ж е н и е   п о л ь з о в а т е л я
    string result; //р е з у л ь т а т   п р е о б р а з о в а н и я
    //В в о д   в ы р а ж е н и я   д в у м я   с п о с о б а м и :   с   к л а в
и а т у р ы   и   *   с   ф а й л а
    cout << "В в е д и т е   в ы р а ж е н и е : ";
    getline(cin, expr);
    expr = DelSpaces(expr); //у д а л я е т   п р о б е л ы   в   н а ч а л е
и   в   к о н ц е ,   п р о б е л ы ,   и д у щ и е   п о д р я д

    if (!invalidInputDirectPolish(expr))
    {
        result = fromDirectPolishNotation(expr);
        cout << "П р е о б р а з о в а н н о е   в ы р а ж е н и е : " <<
result << "\n";
    }
    else { cout << "О ш и б к а !   Н е к о р р е к т н о е   в ы р а ж е н и
е\n"; return "\0"; }

    //П о с л е о б р а б о т а н н о г о   в ы р а ж е н и я :

    return result;
}

//О б р а т н у ю   п о л ь с к у ю   в   п р я м у ю
string fromPolishNotationToDirect()
{
    string expr; //в ы р а ж е н и е   п о л ь з о в а т е л я
    string result; //р е з у л ь т а т   п р е о б р а з о в а н и я
    //В в о д   в ы р а ж е н и я   д в у м я   с п о с о б а м и :   с   к л а в
и а т у р ы   и   *   с   ф а й л а
    cout << "В в е д и т е   в ы р а ж е н и е : ";
    getline(cin, expr);
    expr = DelSpaces(expr); //у д а л я е т   п р о б е л ы   в   н а ч а л е
и   в   к о н ц е ,   п р о б е л ы ,   и д у щ и е   п о д р я д

    if (!invalidInputPolish(expr))
    {
        result = fromPolishNotation(expr);
        result = directPolishNotation(result);
    }
}

```

```

        cout << "Преобразованное выражение: " <<
result << '\n';
    }
    else { cout << "Ошибка! Некорректное выражени
e\n"; return "\0"; }

    //После обработанного выражения:
    return result;
}

//Прямую польскую в обратную
string fromDirectToPolishNotation()
{
    string expr; //выражение пользователя
    string result; //результат преобразования
    //Ввод выражения двумя способами: с клави
и а т у р ы и * с ф а й л а
    cout << "Введите выражение: ";
    getline(cin, expr);
    expr = DelSpaces(expr); //удаляет пробелы в начале
и в конце, пробелы, идущие подряд

    if (!invalidInputDirectPolish(expr))
    {
        result = fromDirectPolishNotation(expr);
        result = polishNotation(result);
        cout << "Преобразованное выражение: " <<
result << '\n';
    }
    else { cout << "Ошибка! Некорректное выражени
e\n"; return "\0"; }

    //После обработанного выражения:
    return result;
}

//Проверка на корректность обычного выраже
ния
void taskInvalidInput()
{
    string expr; //выражение пользователя
    cout << "Введите выражение, которое нужно п
роверить на корректность: ";
    getline(cin, expr);
    expr = DelSpaces(expr); //удаляет пробелы в начале
и в конце, пробелы, идущие подряд

```

```

        if (!invalidInput(expr)) { cout << "Следующее выражение является правильным: " << expr << "\n"; }
        else { cout << "Выражение содержит ошибку\n"; }
    }

```

//Проверка на корректность обратной польской записи

```

void taskInvalidInputPolish()
{

```

```

    string expr; //выражение пользователя
    cout << "Введите выражение, которое нужно проверить на корректность: ";
    getline(cin, expr);
    expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

```

```

        if (!invalidInputPolish(expr)) { cout << "Следующее выражение является правильным: " << expr << "\n"; }
        else { cout << "Выражение содержит ошибку\n"; }
    }

```

//Проверка на корректность прямой польской записи

```

void taskInvalidInputDirectPolish()
{

```

```

    string expr; //выражение пользователя
    cout << "Введите выражение, которое нужно проверить на корректность: ";
    getline(cin, expr);
    expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

```

```

        if (!invalidInputDirectPolish(expr)) { cout << "Следующее выражение является правильным: " << expr << "\n"; }
        else { cout << "Выражение содержит ошибку\n"; }
    }

```

//Вычисление в обычной форме

```

void taskCalculateInfix()
{

```

```

    string expr; //выражение пользователя
    cout << "Введите выражение, которое нужно вычислить: ";
    getline(cin, expr);
    expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

```

```

    if (!invalidInput(expr))

```

```

    {
        expr = polishNotation(expr);
        cout << "О т в е т : " << calculate(expr) << "\n";
    }
    else { cout << "В ы р а ж е н и е   н е к о р р е к т н о\n"; }
}

//В ы ч и с л е н и е   о б р а т н о й   п о л ь с к о й
void taskCalculate()
{
    string expr; //в ы р а ж е н и е   п о л ь з о в а т е л я
    cout << "В в е д и т е   в ы р а ж е н и е , к о т о р о е   н у ж н о   в
ы ч и с л и т ь : ";
    getline(cin, expr);
    expr = DelSpaces(expr); //у д а л я е т   п р о б е л ы   в   н а ч а л е
и   в   к о н ц е , п р о б е л ы , и д у щ и е   п о д р я д

    if (!invalidInputPolish(expr))
    {
        cout << "О т в е т : " << calculate(expr) << "\n";
    }
    else { cout << "В ы р а ж е н и е   н е к о р р е к т н о\n"; }
}

//В ы ч и с л е н и е   п р я м о й   п о л ь с к о й
void taskCalculateDirect()
{
    string expr; //в ы р а ж е н и е   п о л ь з о в а т е л я
    cout << "В в е д и т е   в ы р а ж е н и е , к о т о р о е   н у ж н о   в
ы ч и с л и т ь : ";
    getline(cin, expr);
    expr = DelSpaces(expr); //у д а л я е т   п р о б е л ы   в   н а ч а л е
и   в   к о н ц е , п р о б е л ы , и д у щ и е   п о д р я д

    if (!invalidInputDirectPolish(expr))
    {
        expr = fromDirectPolishNotation(expr);
        expr = polishNotation(expr);
        cout << "О т в е т : " << calculate(expr) << "\n";
    }
    else { cout << "В ы р а ж е н и е   н е к о р р е к т н о\n"; }
}

int lb3()
{
    setlocale(0, "");
    string result = "\0"; //с т р о к а   в ы х о д а
    float res = 0;

```



```

bool check = true; //выход из меню
bool check1 = false; //выход из подменю
//false - заканчивает цикл, приводя непосредственно к выходу
do {
    char sw = ' '; //переключатель главного меню

    char sw1 = ' '; //переключатель саб-меню
    cout << "\nВыберите нужный раздел: \n";
    cout << "\x1b[32m[1]\x1b[0m Преобразование введённого выражения\n";
    cout << "\x1b[32m[2]\x1b[0m Проверить выражение на корректность\n";
    cout << "\x1b[32m[3]\x1b[0m Вычислить выражение\n";
    cout << "\x1b[32m[4]\x1b[0m Очистить экран консоли\n";
    cout << "\x1b[32m[0]\x1b[0m Выйти в главное меню\n";

    cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

    cin >> sw;
    while (cin.get() != '\n') { sw = ' '; }; //если строка содержит более одного символа, возвращается ошибка

    switch (sw)
    {

        case '1': //[1] Преобразование введённого выражения
            do {
                check1 = false;
                sw1 = ' ';
                cout << "\n\x1b[32m[1]\x1b[0m Ввести обычное выражение и преобразовать в обратную польскую запись\n";
                cout << "\x1b[32m[2]\x1b[0m Ввести обычное выражение и преобразовать в прямую польскую запись\n";
                cout << "\x1b[32m[3]\x1b[0m Ввести обратную польскую запись и преобразовать в обычное выражение\n";
                cout << "\x1b[32m[4]\x1b[0m Ввести прямую польскую запись и преобразовать в обычное выражение\n";
            }
        }
    }
}

```

```

        cout << "\x1b[32m[3]\x1b[0m Ввести обратн
ую польскую запись и преобразовать в пряму
ую польскую запись\n";
        cout << "\x1b[32m[4]\x1b[0m Ввести прямую
польскую запись и преобразовать в обратную
польскую запись\n";
        cout << "\x1b[32m[0]\x1b[0m Вернуться на з
ад\n";

        cout << "Пожалуйста, введите числ
о, чтобы выполнить нужное действие: ";

        cin >> sw1;
        while (cin.get() != '\n') { sw1 = ' '; };

        switch (sw1)
        {
            case '1': //[1] обычное выражение =>
обратная польская
                toPolishNotation();
                break;
            case '2': //[2] обычное выражение =>
прямая польская
                toDirectPolishNotation();
                break;
            case '3': //[3] обратная польская =>
обычное выражение
                fromPolishNotationToInfix();
                break;
            case '4': //[4] прямая польская => об
ычное выражение
                fromDirectPolishNotationToInfix();
                break;
            case '5': //[5] обратная польская =>
прямая польская
                fromPolishNotationToDirect();
                break;
            case '6': //[6] прямая польская => об
ратная польская
                fromDirectToPolishNotation();
                break;
            case '0': //[0] Назад
                break;
            default:
                cout << "Ошибка! Пожалуйста, по
пробуйте снова\n";
                check1 = true; //цикл пойдёт заново

                break;

```

```

    }
} while (check1);
break;

case '2': //[2] Проверить выражение на кор
ректность
do {
    check1 = false;
    sw1 = ' ';
    cout << "\nВыберите, что хотите сде
лать: \n";

    cout << "\x1b[32m[1]\x1b[0m Проверить на
корректность простого выражения\n";
    cout << "\x1b[32m[2]\x1b[0m Проверить на
корректность выражения в обратной польско
й записи\n";

    cout << "\x1b[32m[3]\x1b[0m Проверить на
корректность выражения в прямой польской з
аписи\n";

    cout << "\x1b[32m[0]\x1b[0m Вернуться на з
ад\n";

    cout << "Пожалуйста, введите числ
о, чтобы выполнить нужное действие: ";

    cin >> sw1;
    while (cin.get() != '\n') { sw1 = ' '; };

    switch (sw1)
    {
    case '1': //[1] корректность простог
о выражения
        taskInvalidInput();
        break;
    case '2': //[2] в обратной польской
записи
        taskInvalidInputPolish();
        break;
    case '3': //[3] в прямой польской за
писи
        taskInvalidInputDirectPolish();
        break;
    case '0': //[0] Назад
        break;
    default:
        cout << "Ошибка! Пожалуйста, по
пробуйте снова\n";
        check1 = true; //цикл пойдёт заново

```

```

        break;
    }
} while (check1);
break;

case '3': //[3] Вычислить выражение
do {
    check1 = false;
    sw1 = ' ';
    cout << "\nВыберите, что хотите сде
лать: \n";

    cout << "\x1b[32m[1]\x1b[0m Вычислить обы
чное выражение\n";
    cout << "\x1b[32m[2]\x1b[0m Вычислить выр
ажение в обратной польской записи\n";
    cout << "\x1b[32m[3]\x1b[0m Вычислить выр
ажение в прямой польской записи\n";
    cout << "\x1b[32m[0]\x1b[0m Вернуться на з
ад\n";

    cout << "Пожалуйста, введите числ
о, чтобы выполнить нужное действие: ";

    cin >> sw1;
    while (cin.get() != '\n') { sw1 = ' '; };

    switch (sw1)
    {
    case '1': //[1] Вычислить обычное вы
ражение
        taskCalculateInfix();
        break;
    case '2': //[2] Вычислить выражение
в обратной польской записи
        taskCalculate();
        break;
    case '3': //[3] Вычислить выражение
в прямой польской записи
        taskCalculateDirect();
        break;
    case '0': //[0] Назад
        break;
    default:
        cout << "Ошибка! Пожалуйста, по
пробуйте снова\n";
        check1 = true; //цикл пойдёт зано
во

        break;
    }
} while (check1);

```

```

        break;

    case '4': //[4] Очистка экрана
        system("cls");
        break;

    case '0': //[0] Закрывать программу
        cout << "Выход из программы...\n";
        check = false; //выход из цикла
        break;

    default: //в случае, если введено что-то и
ное
        cout << "Ошибка! Пожалуйста, попробуйте
те снова\n";
        break;
    }

} while (check);

system("Pause");
return 0;
}

int main()
{
    setlocale(0, "");

    bool check = true; //выход из меню
    //false - заканчивает цикл, приводя непосредственно к выходу
do {
    //system("cls");
    char sw = ' '; //переключатель главного меню
ное
    cout << "\nВыберите нужный раздел: \n";
    cout << "\x1b[32m[1]\x1b[0m Работа со структура
ми; Записи о студентах\n";
    cout << "\x1b[32m[2]\x1b[0m Работа с динамическ
ими массивами и двусвязными списками \n";
    cout << "\x1b[32m[3]\x1b[0m Работа со стеками; П
ольская и обратная нотация\n";
    cout << "\x1b[32m[0]\x1b[0m Закрывать программу\n";
    cout << "Пожалуйста, введите число, чтоб
ы выполнить нужное действие: ";

    cin >> sw;

```

```
while (cin.get() != '\n') { sw = ' '; }; //если строка  
содержит более одного символа, возвращаетс  
я ошибка
```

```
switch (sw)  
{
```

```
case '1': //[1] Создание целочисленного од  
номерного массива
```

```
lb1();  
break;
```

```
case '2': //[2] Работа с элементами массив  
а
```

```
lb2();  
break;
```

```
case '3': //[3] Польская нотация, стеки
```

```
lb3();  
break;
```

```
case '0': //[0] Закрывать программу
```

```
cout << "Выход из программы...\n";
```

```
check = false; //выход из цикла
```

```
break;
```

```
default: //в случае, если введено что-то и  
ное
```

```
cout << "Ошибка! Пожалуйста, попробуйте  
те снова\n";
```

```
break;
```

```
}
```

```
} while (check);
```

```
return 0;
```

```
}
```