

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

ОТЧЕТ
по практической работе №2
по дисциплине «Программирование»
ТЕМА: изучение свойств и организация динамических массивов и
двусвязных списков.

СТУДЕНТ ГР.0324

ПРЕПОДАВАТЕЛЬ

КОШЕЛЯЕВ А.С

ГЛУЩЕНКО А.Г

САНКТ-ПЕТЕРБУРГ

2021

Цель работы.

Изучение свойств и организация динамических массивов и двусвязных списков; получение практических навыков в работе с динамическими массивами и двусвязными списками; проведение сравнительной характеристики скорости вставки, получения и удаления элементов из них.

Основные теоретические положения.

Кроме отдельных динамических объектов в языке C++ мы можем использовать динамические массивы. Для выделения памяти под динамический массив также используется оператор **new**, после которого в квадратных скобках указывается, сколько массив будет содержать объектов:

```
int *numbers = new int[4]; // динамический массив из 4 чисел
```

Причем в этом случае оператор **new** также возвращает указатель на объект типа **int** - первый элемент в созданном массиве. В данном случае определяется массив из четырех элементов типа **int**, но каждый из них имеет неопределенное значение. Однако мы также можем инициализировать массив значениями:

```
int *n1 = new int[4]; // каждый элемент имеет неопределенное значение
int *n2 = new int[4](); // каждый элемент имеет значение по умолчанию - 0
int *n3 = new int[4]{ 1, 2, 3, 4 }; // массив состоит из чисел 1, 2, 3, 4
```

В последнем случае при инициализации массива конкретными значениями следует учитывать, что если значений в фигурных скобках больше, чем длина массива, то оператор **new** потерпит неудачу и не сможет создать массив. Если переданных значений, наоборот, меньше, то элементы, для которых не предоставлены значения, инициализируются значением по умолчанию. После создания динамического массива мы сможем с ним работать по полученному указателю, получать и изменять его элементы:

```
int n = 5; // размер массива
int *p = new int[n]{ 1, 2, 3, 4, 5 };
for (int *q = p; q != p + n; q++)
{
    std::cout << *q << "\t";
}
```

Для удаления динамического массива и освобождения его памяти применяется специальная форма оператора **delete**:

```
delete [] указатель_на_динамический_массив;
```

Некорректная работа с динамической памятью чревата серьезными ошибками. Это ошибка связана с возможным переполнением динамической области памяти, когда после окончания использования динамических данных мы “забываем” освободить память с помощью инструкции **delete**.

Другая категория ошибок называется “утечкой памяти”. Например:

```
int * p;           // Объявляем указатель на целый тип данных
p = new int;       // Выделяем память по некоторому адресу p
.....
p = new int;       // Еще раз выделяем память, и ее адрес записываем опять в p
```

В этом примере повторное присвоение переменной **p** другого адреса нового участка памяти приводит к потере адреса участка памяти, выделенного первой инструкцией **new**. Этот “забытый” участок памяти будет занят до конца работы программы, и его нельзя ни освободить, ни использовать для хранения данных – говорят, что произошла утечка памяти. Такие “утечки” могут привести к тому, что опять произойдет переполнение динамической области памяти. Для недопущения подобных ошибок необходимо внимательно следить за своевременным освобождением памяти, на которую ссылается переменная-указатель.

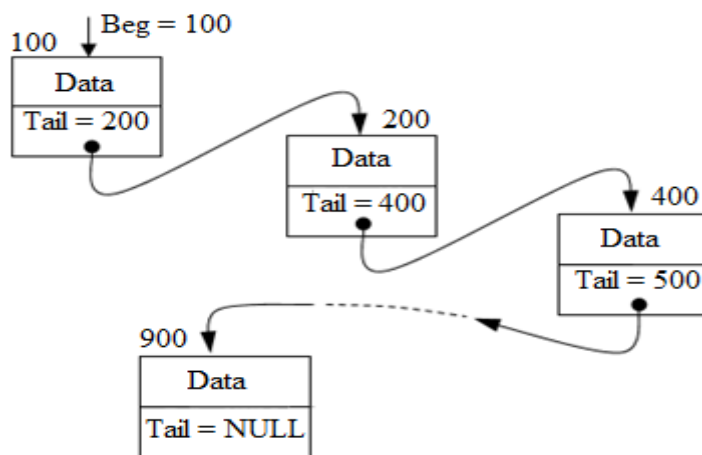
Еще одна категория ошибок связана с попытками обращения к динамической памяти через указатели, не инициализированные с помощью инструкции **new**, а также при попытке обращения к динамической памяти через указатель после освобождения памяти с помощью инструкции **delete**.

Первая структура данных, которую мы рассмотрим — связный список. На то есть две причины: первое — связный список используется практически везде — от ОС до игр, и второе — на его основе строится множество других структур данных. Основное назначение связного списка — предоставление

механизма для хранения и доступа к произвольному количеству данных. Как следует из названия, это достигается связыванием данных вместе в список.

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

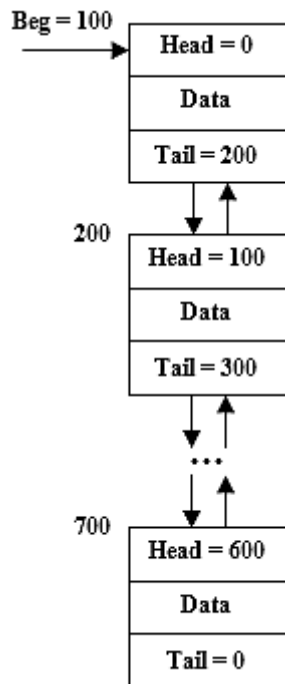
Информационная часть предназначена для хранения полезных данных и может иметь практически любой тип. Адресная часть каждого элемента содержит адрес следующего элемента списка.



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;
};
```

Поле **Head** содержит адрес предыдущего элемента, поле **Tail** содержит адрес следующего элемента списка. Такая организация списка позволяет перемещаться по его элементам в двух направлениях.

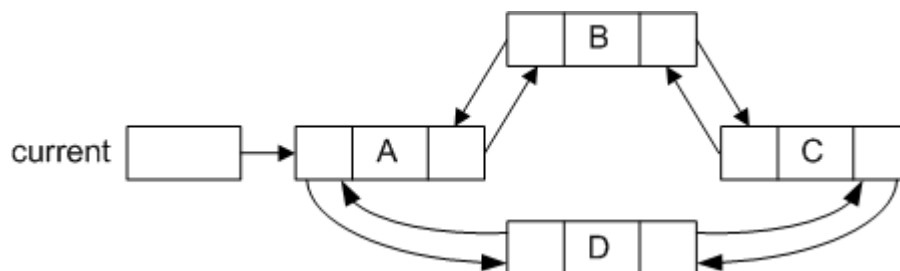


Основные действия, производимые над узлами двусвязного линейного списка (ДЛС):

- 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.

Еще один вид связного списка – кольцевой список. В кольцевом односвязном списке последний элемент ссылается на первый. В случае двусвязного кольцевого списка – плюс к этому первый ссылается на последний. Таким образом, получается зацикленная структура. Кольцо – циклическая структура, которая обычно реализует операции вставки и удаления, а также операции **next** и **prev**, позволяющие двигаться вперед и назад по кольцу.



Структура-кольцо будет, подобно двусвязному списку, хранить указатель на узел кольца и его размер.

Постановка задачи.

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;

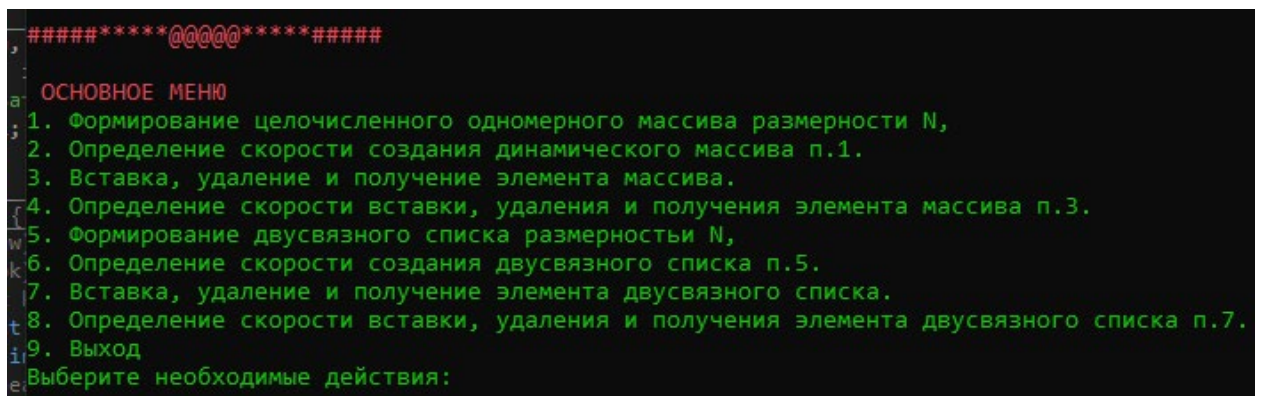
в) список считывается с файла, N определяется как количество элементов списка в файле.

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, удаление 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.

Необходимо реализовать программу, которая выполняет вышеизложенные действия. Написана консольное приложение позволяющее выполнять требуемые операции.



```
#####*****@@@@*****#####
ОСНОВНОЕ МЕНЮ
1. Формирование целочисленного одномерного массива размерности N,
2. Определение скорости создания динамического массива п.1.
3. Вставка, удаление и получение элемента массива.
4. Определение скорости вставки, удаления и получения элемента массива п.3.
5. Формирование двусвязного списка размерности N,
6. Определение скорости создания двусвязного списка п.5.
7. Вставка, удаление и получение элемента двусвязного списка.
8. Определение скорости вставки, удаления и получения элемента двусвязного списка п.7.
9. Выход
Выберите необходимые действия:
```

Рисунок 1 Основное меню программы.

1. Формирование целочисленного одномерного массива размерности N , где:

```

1. Формирование целочисленного одномерного массива размерности N,
#####!!!$$$$Prakt2SemII%%^^^&&&****
=====*****=====
!!!-ПОДМЕНЮ-!!!
1. Пользователь вводит количество элементов в массиве, который будет автоматически заполняться случайными числами (0 до 99);
2. Пользователь вводит в консоль элементы массива, N определяется автоматически по количеству введенных элементов;
3. Массив считывается с файла, (file.txt) N определяется как количество элементов массива в файле;
Выберите необходимые действия:

```

Рисунок 2 Меню формирования массива.

а) пользователь вводит количество элементов в массиве, который будет автоматически заполняться случайными числами (0 до 99);

```

ab Введите количество элементов в массиве
ab 13
ab
ab Скорость создания одномерного динамического массива= 5.23e-05сек.
ab -58 -46 97 67 62 -17 93 -45 -65 11 62 -70 16
ab

```

Рисунок 3 Формирование одномерного динамического массива случайными числами.

б) пользователь вводит в консоль элементы массива, N определяется автоматически по количеству введенных элементов;

```

Введите значение элементов массива окончания ввода буква или знак
Продолжайте ввод элементов
12
Продолжайте ввод элементов
13
Продолжайте ввод элементов
-5
Продолжайте ввод элементов
7
Продолжайте ввод элементов
2012
Возможные значения <В диапазоне от -999 до 999>
Продолжайте ввод элементов
526
Продолжайте ввод элементов
14t
Продолжайте ввод элементов
Ввод 6 элементов закончен
12 13 -5 7 526 14

```

Рисунок 4 Формирование одномерного динамического массива числами, вводимыми пользователем в консоль.

Реализована проверка на ввод только чисел диапазон значений ограничен - 999 до 999.

в) массив считывается с файла, N определяется как количество элементов массива в файле.

```
Загрузка файла test.txt
Всё работает. Файл открыт!

Скорость создания одномерного динамического массива= 0.0009889сек.
7 911 432 -25 -17 -68 8 921 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852
```

Рисунок 5 Формирование одномерного динамического массива числами, из файла.

Реализована проверка на наличие (мусорных данных) не цифр также имеется возможность изменять количество элементов.

2. Определение скорости создания динамического массива п. 1.

```
2. Определение скорости создания динамического массива п.1.
####@@@!!!!$$$$Prakt2SemII%%^&&&&8888***~
Скорость создания одномерного динамического массива = 0.0009889сек.
```

Рисунок 6 Скорость создания массива.

3. Вставка, удаление и получение элемента массива. Удаление и получение элемента необходимо реализовать по индексу и по значению.

```
3. Вставка, удаление и получение элементов массива.
####@@@!!!!$$$$Prakt2SemII%%^&&&&8888***~
=====
!!!=ПОДМЕНЮ=!!!
1. Вставка элемента массива.
2. Удаление элементов массива по индексу.
3. Удаление элемента массива по значению.
4. Получение элемента массива по ндексу.
5. Получение элемента массива по значению.
Выберите необходимые действия:
```

Рисунок 7 Подменю пункта 3.

Вставка элемента массива может осуществляться в начало массива, в конец и в отсортированный массив согласно значению элемента.

```
Введите количество элементов необходимых для вставки в массив
5
Было...
7 911 432 -25 -17 -68 8 921 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852
Введите значение элементов массива
11
Введите значение элементов массива
35
Введите значение элементов массива
7
Введите значение элементов массива
88
Введите значение элементов массива
9999
Значение элемента массива задано <В диапазоне от -999 до 999 >
Введите значение элементов массива
12
Через несколько сотых сек...
7 911 432 -25 -17 -68 8 921 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 7 88 12
```

Рисунок 8 Вставка

Вставка в конец массива нескольких элементов.

Удаление элементов по индексу

```
Введите количество элементов для удаления
3
Введите индекс элементов для удаления
5
Введите индекс элементов для удаления
-47
Введите индекс элементов для удаления
1
Было...
7 911 432 -25 -17 -68 8 921 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 7 88 12
Через несколько сотых сек...
7 432 -25 -17 8 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 7 88 12
```

Рисунок 9 Удаление элементов по индексам.

Удаление элементов по значению

```
Было...
7 432 -25 -17 8 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 7 88 12
Введите значение элемента для удаления
7
Времени нет. Сделать удаления нескольких элементов уви по заданию...
432 -25 -17 8 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 88 12
```

Рисунок 10 Удаление элемента по значению.

Получение элемента по индексу

```
Было...
432 -25 -17 8 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 88 12
Введите индекс элемента для получения значений
-2
Это должно быть число
Введите индекс элемента для получения значений
-2
Индекс должен быть положительным
Введите индекс элемента для получения значений
77
Максимально возможное значение индекса = 29
Введите индекс элемента для получения значений
0
432 -25 -17 8 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 88 12
Значение элемента array[0]=432
Через несколько сотых сек...
432 -25 -17 8 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 88 12
```

Рисунок 11 Получение элемента.

Получение элемента по значению

```
Было...
432 -25 -17 8 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 88 12
Введите значение элемента для поиска в массиве
77
В массиве нет элемента имеющего значение = 77
Через несколько сотых сек...
432 -25 -17 8 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852 11 35 88 12
```

Рисунок 12 Получение элемента.

4. Определение скорости вставки, удаления и получения элемента массива п. 3.

```
4. Определение скорости вставки, удаления и получения элемента массива п.3.
####@!!!$$$$Prakt2SemII%%^8888****
Скорость вставки элементов одномерного динамического массива = 2.4e-05сек.
Скорость удаления элементов одномерного динамического массива = 4.8e-05сек.
Скорость получения элементов одномерного динамического массива = 0.0231823сек.
```

Рисунок 13 Определение скорости.

5. Формирование двусвязного списка размерности N , где:

```
5. Формирование двусвязного списка размерности N,
####@!!!$$$$Prakt2SemII%%^8888****
*****-----*****
!!!=ПОДМЕНЮ=!!!
1. Пользователь вводит количество элементов в списке, который будет автоматически заполняться случайными числами (0 до 99);
2. Пользователь вводит в консоль элементы списка, N определяется автоматически по количеству введенных элементов;
3. Двусвязный список считывается с файла, (file.txt) N определяется как количество элементов списка в файле;
Выберите необходимые действия:
```

Рисунок 14 Подменю формирования списка.

- а) пользователь вводит количество элементов в списке, который будет автоматически заполняться случайными числами (0 до 99);

```
1
ok) Введите количество элементов в двусвязном списке
t k 18
nt -58 -46 97 67 62 -17 93 -45 -65 11 62 -70 16 96 -38 -4 -74 -37
,ir
```

Рисунок 15 Формирование списка случайными числами (0 до 99);

- б) пользователь вводит в консоль элементы списка, N определяется автоматически по количеству введенных элементов;

```
Введите значение элементов двусвязного списка окончания ввода буква или знак
Продолжайте ввод элементов
33
Продолжайте ввод элементов
-45
Продолжайте ввод элементов
-7
Продолжайте ввод элементов
11
Продолжайте ввод элементов
9999
Возможные значения <В диапазоне от -999 до 999>
Продолжайте ввод элементов
78e
Продолжайте ввод элементов

Ввод 5 элементов закончен
33 -45 -7 11 78
```

Рисунок 16 Формирование двусвязного списка числами, вводимыми пользователем в консоль.

в) массив считывается с файла, N определяется как количество элементов массива в файле.

```
Загрузка файла test.txt
Всё работает. Файл открыт!

Скорость создания одномерного динамического массива= 0.0009889сек.
7 911 432 -25 -17 -68 8 921 287 -33 -44 21 962 714 100 -40 999 466 -39 -32 347 72 -76 205 47 -29 903 12 -4 852
```

Рисунок 17 Формирование одномерного динамического массива числами, из файла.

Реализована проверка на наличие (мусорных данных) не цифр также имеется возможность изменять количество элементов.

6. Определение скорости создания двусвязного списка п. 5.

```
6. Определение скорости создания двусвязного списка п.5.
####@!!!$Prakt2SemII%^^^8888***~
Скорость создания двусвязного списка = 0.0005861сек.
```

Рисунок 18 Скорость создания двусвязного списка.

7. Вставка, удаление и получение элемента двусвязного списка. Удаление и получение элемента необходимо реализовать по индексу и по значению.

```
7. Вставка, удаление и получение элемента двусвязного списка.
####@!!!$$$$Prakt2SemII%/%/%/^^^8888***~
=====*****=====*****=====
!!!-ПОДМЕНЮ=!!!
1. Вставка элемента двусвязного списка.
2. Удаление элемента списка по (индексу хз что это значит сделал по счёту).
3. Удаление элемента списка по значению.
4. Получение элемента списка по (индексу хз что это значит сделал по счёту).
5. Получение элемента списка по значению.
Выберите необходимые действия:
```

Рисунок 19 Подменю пункта 7.

Вставка элемента списка может осуществляться в начало списка, в конец и в отсортированный список согласно значению элемента.

```
Было...
-58 -46 97 67 62
Введите значение элементов двусвязного списка для вставки в конец списка, окончание ввода буква или знак
Продолжайте ввод элементов
11
Продолжайте ввод элементов
35
Продолжайте ввод элементов
9999
Возможные значения <В диапазоне от -999 до 999>
Продолжайте ввод элементов
1e
Продолжайте ввод элементов
Ввод элементов закончен
-58 -46 97 67 62 11 35 1
```

Рисунок 20 Вставка

Вставка в конец списка нескольких элементов.

Удаление элемента по индексу.

```
Было...
-58 -46 97 67 62 11 35 1
Введите индекс двусвязного списка для удаления
-5
Индекс элемента должен быть положительным
Введите индекс двусвязного списка для удаления
99
Индекс элемента не может быть больше 8
Введите индекс двусвязного списка для удаления
5
Результат удаления
-58 -46 97 67 11 35 1
```

Рисунок 21 Удаление элемента по индексу.

Списки не имеют индексов по этой причине и для удобства восприятия здесь и далее все индексы списков начинаются с 1.

Удаление элемента по значению.

```
Было...
-58 -46 97 67 62 -17 93 -45 -65 11
Введите значение элемента двусвязного списка для удаления
n
Это должно быть число
Введите значение элемента двусвязного списка для удаления
-58
Элемент -58 удален из списка.
< -46 97 67 62 -17 93 -45 -65 11
```

Рисунок 22 Удаление элемента списка по значению.

Получение элемента списка по индексу

```
Оригинал...
-46 97 67 62 -17 93 -45 -65 11
Введите индекс двусвязного списка для получения элемента <Индексы с 1-ы>
-1
Индекс элемента должен быть положительным <Индексы с 1-ы>
Введите индекс двусвязного списка для получения элемента <Индексы с 1-ы>
0
Это должно быть число
Введите индекс двусвязного списка для получения элемента <Индексы с 1-ы>
5
Значение элемента list[5]=-17 найдено в списке
```

Рисунок 23 Получение элемента списка

Получение элемента списка по значению

```
Оригинал...
-46 97 67 62 -17 93 -45 -65 11
Введите значение элемента для поиска в списке
-17
Элемент -17 найден в списке 1 раз.
```

Рисунок 24 Получение элемента списка по значению.

8. Определение скорости вставки, удаление и получения элемента двусвязного списка п. 7.

```
8. Определение скорости вставки, удаления и получения элемента двусвязного списка п.7.
#####!!!!$$$$Prakt2SemII%%^^^&&&***~~~~~
Скорость вставки элементов двусвязного списка = 2.54e-05сек.
Скорость удаления элементов двусвязного списка = 2.16e-05сек.
Скорость получения элементов двусвязного списка = 0.046795сек.
```

Рисунок 25 Определение скорости.

Выводы.

ПОЛУЧИВ ОПЫТ В ИСПОЛЬЗОВАНИИ МАССИВА И СПИСКА ВИДНО ЧТО КАЖДЫЙ ИМЕЕТ РЯД СВОИХ ДОСТОИНСТВ И НЕДОСТАТКОВ.

ПРИЛОЖЕНИЕ А

ПОЛНЫЙ КОД ПРОГРАММЫ

```
// Практическая Работа №2.

#include <iostream>
#include <iomanip>
#include <string> //отличная библиотека работы со строками
#include <chrono> //для определения классов и функций, которые представляют и обрабатывают
длительность и время ожидания.
#include <windows.h> // разобрался это библиотека подключает функционал ОС Windows
#include <fstream> // читать и писать файлы

using namespace std;
struct Data //создал отдельную структуру чисто под данные
{
    int a;
}qwerty;
struct b //только для пункта б где что-то там автоматически
{
    int ch;
    b *nextb;
};
struct list //структура для описания двусвязного списка
{
    Data z; // информационная часть
    list *next; // адресная часть на следующий элемент
    list *prev; // адресная часть на предыдущий элемент
}listok;
double t, vstavka, udalenie, poluchenie; //это для массива
double tim, vsta, iuda, poliscay; //это для списков
int *arr; // указатель для выделения памяти под массив
list *head = NULL;
list *last;
int strok = 1;
```

```

int menu(int xz) {
    int qwe = 0;
    HANDLE O = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(O, FOREGROUND_RED | FOREGROUND_INTENSITY);
    cout << "\n#####@@@@@#####\n";
    cout << "\n ОЧОБНОЕ МЕНЮ\n";
    SetConsoleTextAttribute(O, FOREGROUND_GREEN | FOREGROUND_INTENSITY);
    cout << setw(4) << "1. Формирование целочисленного одномерного массива размерности N,\n";
    cout << setw(4) << "2. Определение скорости создания динамического массива п.1.\n";
    cout << setw(4) << "3. Вставка, удаление и получение элемента массива.\n";
    cout << setw(4) << "4. Определение скорости вставки, удаления и получения элемента массива
п.3.\n";

    cout << setw(4) << "5. Формирование двусвязного списка размерности N," << endl;
    cout << setw(4) << "6. Определение скорости создания двусвязного списка п.5.\n";
    cout << setw(4) << "7. Вставка, удаление и получение элемента двусвязного списка.\n";
    cout << setw(4) << "8. Определение скорости вставки, удаления и получения элемента
двусвязного списка п.7.\n";

    cout << setw(4) << "9. Выход" << endl;
    while (true)
    {
        cout << setw(4) << "Выберите необходимые действия:" << endl;
        cin >> qwe;
        if (cin.fail())
        {
            cout << "Это должно быть число" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (qwe <= 0)
        {
            cout << " Значение должно быть положительным" << endl;
            continue;
        }
        if (qwe > 9)
        {
            cout << " Возможное значение <В диапазоне от 1-9>" << endl;
            continue;
        }
        else break;
    }
    system("cls");
    return qwe;
}

int podmenu(int qw) {
    int zx = 0;
    cout << "===== " << endl;
    cout << setw(24) << " !!!=ПОДМЕНЮ=!!!" << endl;
    if (qw == 1)
    {
        cout << setw(4) << "1. Пользователь вводит количество элементов в массиве, который
будет автоматически заполняться случайными числами (0 до 99);" << endl;
    }
}

```



```

        cout << setw(4) << "2. Пользователь вводит в консоль элементы массива, N
определяется автоматически по количеству введенных элементов;" << endl;
        cout << setw(4) << "3. Массив считывается с файла, (file.txt) N определяется как
количество элементов массива в файле;" << endl;
        while (true)
        {
            cout << setw(4) << "Выберите необходимые действия:" << endl;
            cin >> zx;
            if (cin.fail())
            {
                cout << "Это должно быть число" << endl;
                cin.clear();
                cin.ignore(32767, '\n');
                continue;
            }
            if (zx <= 0)
            {
                cout << " Значение должно быть положительным" << endl;
                continue;
            }
            if (zx > 3)
            {
                cout << " Возможное значение <В диапазоне от 1-3>" << endl;
                continue;
            }
            else break;
        }
        system("cls");
        return zx;
    }
    if (qw == 2)
    {
        cout << setw(4) << "1. Вставка элемента массива." << endl;
        cout << setw(4) << "2. Удаление элементов массива по индексу." << endl;
        cout << setw(4) << "3. Удаление элемента массива по значению." << endl;
        cout << setw(4) << "4. Получение элемента массива по ндексу." << endl;
        cout << setw(4) << "5. Получение элемента массива по значению." << endl;
        while (true)
        {
            cout << setw(4) << "Выберите необходимые действия:" << endl;
            cin >> zx;
            if (cin.fail())
            {
                cout << "Это должно быть число" << endl;
                cin.clear();
                cin.ignore(32767, '\n');
                continue;
            }
            if (zx <= 0)
            {
                cout << " Значение должно быть положительным" << endl;
                continue;
            }
        }
    }
}

```

```

        if (zx > 5)
        {
            cout << " Возможное значение <В диапазоне от 1-5>" << endl;
            continue;
        }
        else break;
    }
    system("cls");
    return zx;
}

if (qw == 3)
{
    cout << setw(4) << "1. Пользователь вводит количество элементов в списке, который
будет автоматически заполняться случайными числами (0 до 99);" << endl;
    cout << setw(4) << "2. Пользователь вводит в консоль элементы списка, N
определяется автоматически по количеству введенных элементов;" << endl;
    cout << setw(4) << "3. Двусвязный список считывается с файла, (file.txt) N
определяется как количество элементов списка в файле;" << endl;
    while (true)
    {
        cout << setw(4) << "Выберите необходимые действия:" << endl;
        cin >> zx;
        if (cin.fail())
        {
            cout << "Это должно быть число" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (zx <= 0)
        {
            cout << " Значение должно быть положительным" << endl;
            continue;
        }
        if (zx > 3)
        {
            cout << " Возможное значение <В диапазоне от 1-3>" << endl;
            continue;
        }
        else break;
    }
    system("cls");
    return zx;
}

if (qw == 4)
{
    cout << setw(4) << "1. Вставка элемента двусвязного списка." << endl;
    cout << setw(4) << "2. Удаление элемента списка по (индексу хз что это значит
сделал по счёту)." << endl;
    cout << setw(4) << "3. Удаление элемента списка по значению." << endl;
    cout << setw(4) << "4. Получение элемента списка по (индексу хз что это значит
сделал по счёту)." << endl;
    cout << setw(4) << "5. Получение элемента списка по значению." << endl;

```

```

        while (true)
        {
            cout << setw(4) << "Выберите необходимые действия:" << endl;
            cin >> zx;
            if (cin.fail())
            {
                cout << "Это должно быть число" << endl;
                cin.clear();
                cin.ignore(32767, '\n');
                continue;
            }
            if (zx <= 0)
            {
                cout << "Значение должно быть положительным" << endl;
                continue;
            }
            if (zx > 5)
            {
                cout << "Возможное значение <В диапазоне от 1-5>" << endl;
                continue;
            }
            else break;
        }
        system("cls");
        return zx;
    }
    return zx;
}

int shet(int strok) //функция подсчёта строк
{
    ifstream file("test.txt");
    if (!file)
    {
        cout << "Error!!! \n";
    }
    else
    {
        while (true)
        {
            string v;
            getline(file, v);
            if (!file.eof())
                strok++;
            else
                break;
        }
        cin.get();
    }
    file.close();
    /*cout << "\n Техданные  :" << strok;*/
    return strok;
}

int ReadFileA(int kol)

```

```

{
    cout << "\n Загрузка файла test.txt \n";
    ifstream file("test.txt");
    if (!file)
    {
        cout << "Файл не открыт!!!" << endl;
        return 0;
    }
    else
    {
        cout << "Всё работает. Файл открыт!" << endl;
    }
    arr = new int[kol]; // выделение памяти под массив
    auto start = chrono::system_clock::now();
    for (int i = 0; i < kol; i++) {
        arr[i] = 0;
        while (true)
        {
            file >> arr[i];
            if (!file)
            {
                cout << "Ошибка чтения, проверьте данные в файле." << endl;
                return 0;
            }
            else break;
        }
    }
    auto end = chrono::system_clock::now();
    chrono::duration<double> diff = end - start;
    t = diff.count();
    cout << "\n" << "Скорость создания одномерного динамического массива= " << diff.count() <<
"сек." << endl;
    file.close();
    return kol;
}

void ReadFileS(int kol) //открытие файла и заполнение списка
{
    last = head;
    int num;
    cout << "\n Загрузка файла test.txt \n";
    ifstream file("test.txt");
    if (!file)
    {
        cout << "Файл не открыт!!!" << endl;
        return;
    }
    else
    {
        cout << "Всё работает. Файл открыт!" << endl;
    }
    auto start = std::chrono::system_clock::now();
    for (int i = 0; i < kol; i++)
    {

```

```

        while (true)
        {
            file >> num;
            if (!file)
            {
                cout << "Ошибка чтения, проверьте данные в файле." << endl;
                return;
            }
            else break;
        }
        if (head == NULL)
        {
            head = new list;
            head->z.a = num;
            head->next = NULL;
            head->prev = NULL;
            last = head;
        }
        else
        {
            list *tmp = last;
            last->next = new list;
            last->next->z.a = num;
            last->next->next = NULL;
            last->next->prev = tmp;
            last = last->next;
        }
        file.get();
    }
    auto end = std::chrono::system_clock::now();
    chrono::duration<double> diff = end - start;
    tim = diff.count();
    cout << "\n" << "Скорость создания двусвязного списка= " << diff.count() << "сек." <<
endl;
    file.close();
}

void outA(int *q,int kol)
{
    if (kol == 0)
    {
        cout << " В массиве нет элементов " << endl;
        return;
    }
    for (int i = 0; i < kol; i++) {
        cout << q[i] << " ";
    }
    cout << endl;
}

void outS(list *head)
{
    list *out = head;
    if (head == NULL)
    {

```

```

        cout << "Двусвязный список пуст!" << endl;
        return;
    }
    while (out)
    {
        cout << out->z.a << " ";
        out = out->next;
    }
    cout << endl;
}

int a1(int ar)
{
    auto srand(time(0));
    arr = new int[ar]; // выделение памяти под массив
    auto start = chrono::system_clock::now();
    for (int i = 0; i < ar; i++) {
        arr[i] = rand() % (199 - 1) - 99;
    }
    auto end = chrono::system_clock::now();
    chrono::duration<double> diff = end - start;
    t = diff.count();
    cout << "\n" << "Скорость создания одномерного динамического массива= " << diff.count() <<
"сек." << endl;
    return ar;
}

int a3(int ar, int a)
{
    int q = ar + a;
    int *tmp;
    int *news;
    news = new int[a];
    int x = 0;
    for (int i = 0; i < a; i++)
    {
        while (true)
        {
            cout << " Введите значение элементов массива " << endl;
            cin >> x;
            if (cin.fail())
            {
                cout << "Это должно быть число" << endl;
                cin.clear();
                cin.ignore(32767, '\n');
                continue;
            }
            if (x > 999 || x < -999)
            {
                cout << " Значение элемента массива задано <В диапазоне от -999 до
999 > " << endl;
                continue;
            }
            else break;
        }
    }
}

```

```

        news[i] = x;
    }
    tmp = new int[q];
    for (int i = 0; i < ar; i++)
    {
        tmp[i] = arr[i];
    }
    int j = 0;
    auto start = chrono::system_clock::now();
    for (int i = ar ; i < q; i++)
    {
        tmp[i] = news[j];
        j++;
    }
    auto end = chrono::system_clock::now();
    delete [] arr;
    delete[] news;
    arr = tmp;
    chrono::duration<double> diff = end - start;
    vstavka = diff.count();
    return q;
}
bool auxiliary(int q[], int len, int val)
{
    for (int i = 0; i < len; i++)
    {
        if (val == q[i])
            return true;
    }
    return false;
}
int *delA3(int w,int *q,int e)
{
    int tmp;
    for (int i = 0; i < e - 1; i++) {
        for (int j = 0; j < e - i - 1; j++) {
            if (q[j] > q[j + 1]) {
                tmp = q[j];
                q[j] = q[j + 1];
                q[j + 1] = tmp;
            }
        }
    }
    auto start = chrono::system_clock::now();
    int *temp = new int[w - e];
    int j = 0;
    for (int i = 0; i < w; i++)
    {
        if (!auxiliary(q, e, i))
        {
            temp[j++] = arr[i];
        }
    }
}

```

```

        delete[] arr;
        auto end = chrono::system_clock::now();
        chrono::duration<double> diff = end - start;
        udalenie = diff.count();
        return temp;
    }
    int delznak(int *q, int s,int b)
    {
        int j = 0;
        int z = 0;
        for (int i = 0; i < s; i++) {
            if (arr[i] != b) {
                arr[j++] = arr[i];
            }
            else {
                z++;
            }
        }
        s = j;
        if (z == 0)
        {
            cout << " Нет элементов для удаления" << endl;
            return s;
        }
        int *tmp;
        tmp = new int[s];
        for (int i = 0; i < s; i++)
        {
            tmp[i] = arr[i];
        }
        delete[]arr;
        arr = tmp;
        return s;
    }
    int A4(int z,int q)
    {
        for (int i = 0; i < q; i++)
        {
            cout << arr[i] << " ";
        }
        cout << endl;
        cout <<"Значение элемента array["<<z<<"]="<< arr[z] << endl;
        return q;
    }
    int A5(int z, int q)
    {
        int s = 0;
        auto start = chrono::system_clock::now();
        for (int i = 0; i < z; i++) {
            if (arr[i] == q) {
                cout << " Элемент найден " << "array[" << i << "]= " << q << endl;
                s++;
            }
        }
    }

```



```

    }
    if (s == 0)
    {
        cout << " В массиве нет элемента имеющего значение = " << q << endl;
    }
    auto end = chrono::system_clock::now();
    chrono::duration<double> diff = end - start;
    poluchenie = diff.count();
    return z;
}

void creat(int kol)
{
    auto srand(time(NULL));
    auto start = chrono::system_clock::now();
    for (int i = 0; i < kol; i++)
    {
        if (head == NULL)
        {
            head = new list;
            head->z.a = rand() % (199 - 1) - 99;
            head->next = NULL;
            head->prev = NULL;
            last = head;
        }
        else
        {
            list *tmp = last;
            last->next = new list;
            last->next->z.a = rand() % (199 - 1) - 99;
            last->next->next = NULL;
            last->next->prev = tmp;
            last = last->next;
        }
    }
    auto end = chrono::system_clock::now();
    chrono::duration<double> diff = end - start;
    tim = diff.count();
}

void DelSpisok(int x, int raz)
{
    auto start = chrono::system_clock::now();
    if (x == 1 && head->next) //если удаляем первый НО есть и другие зараза забыл с 1 семестра
    эту конструкцию
    {
        list *temp = head;           // указываем что нам нужно начало списка
        head = head->next;           // сдвигаем начало на следующий за началом элемент
        head->prev = NULL;           // делаем так чтоб предыдущий началу элемент был
пустым
        delete temp;                 // удаляем удаляемое начало
        raz--; //счетчик
        return;
    }
    else if (x == 1 && head == last) // если удаляем первый НО в списке только 1 элемент

```

```

    {
        head->next = NULL;
        head = NULL;
        delete head;    // удаляем указатель на начало
        raz = 0;
        return;
    }
    if (x == raz) // вот нафига нужно знать ко-во удаляемый элемент является последним
элементом списка
    {
        list *tmp = last;    // указываем что нужен хвост
        last = last->prev; // отодвигаем хвост немного назад
        last->next = NULL; // обозначаем что впереди за хвостом пусто
        delete tmp;    // очищаем память от бывшего хвоста
        raz--;
        return;
    }
    list *tmp = head, *tmp2;    // если удаляемый элемент где-то в середине списка самое
простое
    //tmp-Удаляемый элемент tmp2 нужен
    for (int i = 0; i < x - 1; i++)
    {
        tmp = tmp->next;    // идем к адресу удаляемого элемента
    }
    tmp2 = tmp;    // временно запоминаем адрес удаляемого элемента
    tmp2->prev->next = tmp->next; // записываем данные что следующий за перед сейчас удаляемым
элементом это следующий от удаляемого
    tmp2->next->prev = tmp->prev; //а предыдущий для следующего - это предыдущий для
удаляемого
    delete tmp;
    auto end = chrono::system_clock::now();
    chrono::duration<double> diff = end - start;
    iuda = diff.count();
}
void quest(int raz, int a)
{
    auto start = chrono::system_clock::now();
    list *q = head;
    int i = 0;
    int net = 0;
    while (q)
    {
        if (q->z.a == a)
        {
            ++i;
            cout << "Элемент " << a << " найден в списке " << i << " раз." << endl;
        }
        q = q->next;
        ++net;
    }
    if (raz == net)
    {
        cout << "Элемента " << a << " в списке отсутствует." << endl;
    }
}

```

```

    }
    auto end = chrono::system_clock::now();
    chrono::duration<double> diff = end - start;
    policay = diff.count();
}
int main()
{
Mem1:
    int ar;
    int xz = 0;
    setlocale(LC_ALL, "Russian");
    int lab = menu(xz);
    if (lab == 1) {
        cout << setw(4) << "1. Формирование целочисленного одномерного массива размерности
N," << endl;
        cout << "####@@@!!!!$$$$Prakt2SemII%%^^^&&&***~::~" << endl;
        int pod1 = 1;
        int znak = podmenu(pod1);
        if (znak == 1)
        {
            int a;
            while (true)
            {
                cout << " Введите количество элементов в массиве " << endl;
                cin >> a;
                if (cin.fail())
                {
                    cout << "Это должно быть число" << endl;
                    cin.clear();
                    cin.ignore(32767, '\n');
                    continue;
                }
                if (a <= 0)
                {
                    cout << " Количество элементов должно быть положительным" <<
endl;
                    continue;
                }
                if (a > 77)
                {
                    cout << " X3 я так решил максимум элементов 77" << endl;
                    continue;
                }
                else break;
            }
            ar = 0;
            ar = a1(a);
            outA(arr, ar);
        }
        if (znak == 2)
        {
            b *xz = NULL;
            b *zx;

```

```

zx = xz;
cout << " Ввидите значение элементов массива окончания ввода буква или
знак" << endl;

while (true)
{
    int a;
    cout << " Продолжайте ввод элементов" << endl;
    cin >> a;
    if (cin.fail())
    {
        cin.clear();
        cin.ignore(32767, '\n');
        break;
    }
    if (a > 999 || a < -999)
    {
        cout << " Возможные значения <В диапазоне от -999 до 999> "
<< endl;

        continue;
    }
    if (xz == NULL)
    {
        xz = new b;
        xz->ch = a;
        xz->nextb = NULL;
        zx = xz;
    }
    else
    {
        zx->nextb = new b;
        zx->nextb->ch = a;
        zx->nextb->nextb = NULL; // Элемент после него пустой
        zx = zx->nextb; // Теперь этот эл-т является последним
    }
}

auto start = chrono::system_clock::now();
if (xz == NULL)
{
    cout << " Вы действительно можете набирать текст со скоростью 600
символов в минуту? " << endl;

    cout << " Да! Но такая ерунда получается..." << endl;
    return 0;
}

int poi = 0;
b *tak = xz;
while (tak)
{
    ++poi;
    tak = tak->nextb;
}

cout << " Ввод " << poi << " элементов закончен" << endl;
int i = 0;
arr = new int[poi];

```

```

        while (xz)
        {
            arr[i++] = xz->ch;
            xz = xz->nextb;
        }
        auto end = chrono::system_clock::now();
        chrono::duration<double> diff = end - start;
        t = diff.count();
        ar = 0;
        outA(arr, poi);
        ar = poi;
    }
    if (znak == 3)
    {
        ar = 0;
        strok = shet(strok);
        ar = ReadFileA(strok);
        outA(arr, ar);
        strok = 1;
    }
    goto Mem1;
}

if (lab == 2) {
    cout << setw(4) << "2. Определение скорости создания динамического массива
п.1.\n";

    cout << "####@!!!!$$$$Prakt2SemII%%^^^&&&*****~";
    cout << "\n" << "Скорость создания одномерного динамического массива = " << t <<
"сек." << endl;
    goto Mem1;
}

if (lab == 3) {
    cout << setw(4) << "3. Вставка, удаление и получение элементов массива." << endl;
    cout << "####@!!!!$$$$Prakt2SemII%%^^^&&&*****~" << endl;
    int pod2 = 2;
    int znak = podmenu(pod2);
    if (znak == 1)
    {
        if (arr == NULL)
        {
            cout << " Создайте динамический массив" << endl;
            cout << " ERROR!" << endl;
            return 0;
        }
        int a;
        while (true)
        {
            cout << " Ввидите количество элементов необходимых для вставки в
массив" << endl;

            cin >> a;
            if (cin.fail())
            {
                cout << "Это должно быть число" << endl;
                cin.clear();
            }
        }
    }
}

```

```

        cin.ignore(32767, '\n');
        continue;
    }
    if (a <= 0)
    {
        cout << " Количество элементов должно быть положительным" <<
endl;

        continue;
    }
    if (a > 14)
    {
        cout << " Проблематично вводить такое количество элементов"
<< endl;

        continue;
    }
    else break;
}
cout << " Было..." << endl;
outA(arr, ar);
int q = a3(ar, a);
cout << " Через несколько сотых сек..." << endl;
outA(arr, q);
ar = q;
}
if (znak == 2)
{
    if (arr == NULL)
    {
        cout << " Создайте динамический массив" << endl;
        cout << " ERROR!" << endl;
        return 0;
    }
    int a;
    while (true)
    {
        cout << " Ввидите количество элементов для удаления" << endl;
        cin >> a;
        if (cin.fail())
        {
            cout << "Это должно быть число" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (a <= 0)
        {
            cout << " Количество элементов должно быть положительным" <<
endl;

            continue;
        }
        if (a > ar)
        {

```

```

        cout << " Максимально возможное количество элементов для
удаления = " << ar << endl;

        continue;
    }
    else break;
}
int *tmp;
tmp = new int[a]; // выделение памяти под массив
for (int i = 0; i < a; i++)
{
    int b;
    while (true)
    {
        cout << " Ввидите индекс элементов для удаления" << endl;
        cin >> b;
        if (cin.fail())
        {
            cout << "Это должно быть число" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (b < 0)
        {
            cout << " Индекс элемента должен быть положительным"
<< endl;

            continue;
        }
        if (b >= ar)
        {
            cout << " Максимально возможный индекс для удаления =
" << ar - 1 << endl;

            continue;
        }
        else break;
    }
    tmp[i] = b;
}
cout << " Было..." << endl;
outA(arr, ar);
arr = delA3(ar, tmp, a);
ar -= a;
cout << " Через несколько сотых сек..." << endl;
outA(arr, ar);
}
if (znak == 3)
{
    if (arr == NULL)
    {
        cout << " Создайте динамический массив" << endl;
        cout << " ERROR!" << endl;
        return 0;
    }
}

```

```

cout << " Было..." << endl;
outA(arr, ar);
int b;
while (true)
{
    cout << " Ввидите значение элемента для удаления" << endl;
    cin >> b;
    if (cin.fail())
    {
        cout << "Это должно быть число" << endl;
        cin.clear();
        cin.ignore(32767, '\n');
        continue;
    }
    if (b > 999 || b < -999)
    {
        cout << " Возможные значения <В диапазоне от -999 до 999> "
<< endl;

        continue;
    }
    else break;
}
int xz = delznak(arr, ar, b);
ar = 0;
ar = xz;
cout << " Времени нет. Сделать удаления нескольких элементов увы по
заданию..." << endl;
outA(arr, ar);
}
if (znak == 4)
{
    if (arr == NULL)
    {
        cout << " Создайте динамический массив" << endl;
        cout << " ERROR!" << endl;
        return 0;
    }
    cout << " Было..." << endl;
    outA(arr, ar);
    int a;
    while (true)
    {
        cout << " Ввидите индекс элемента для получения значений" << endl;
        cin >> a;
        if (cin.fail())
        {
            cout << "Это должно быть число" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (a < 0)
        {

```



```

        cout << " Индекс должен быть положительным" << endl;
        continue;
    }
    if (a > ar)
    {
        cout << " Максимально возможное значение индекса = " << ar-1
<< endl;

        continue;
    }
    else break;
}
int v = A4(a, ar);
ar = 0;
ar = v;
cout << " Через несколько сотых сек..." << endl;
outA(arr, ar);
}
if (znak == 5)
{
    if (arr == NULL)
    {
        cout << " Создайте динамический массив" << endl;
        cout << " ERROR!" << endl;
        return 0;
    }
    cout << " Было..." << endl;
    outA(arr, ar);
    int a;
    while (true)
    {
        cout << " Ввидите значение элемента для поиска в массиве" << endl;
        cin >> a;
        if (cin.fail())
        {
            cout << "Это должно быть число" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (a > 999 || a < -999)
        {
            cout << " Возможные значения <В диапазоне от -999 до 999> "
<< endl;

            continue;
        }
        else break;
    }
    int xo = A5(ar, a);
    ar = 0;
    ar = xo;
    cout << " Через несколько сотых сек..." << endl;
    outA(arr, ar);
}

```

```

        goto Mem1;
    }
    if (lab == 4) {
        cout << setw(4) << "4. Определение скорости вставки, удаления и получения элемента
массива п.3." << endl;
        cout << "####@!!!$$$$Prakt2SemII%%^&&****~" << endl;
        cout << "Скорость вставки элементов одномерного динамического массива = " <<
vstavka << "сек." << endl;
        cout << "Скорость удаления элементов одномерного динамического массива = " <<
udalenie << "сек." << endl;
        cout << "Скорость получения элементов одномерного динамического массива = " <<
poluchenie << "сек." << endl;
        goto Mem1;
    }
    if (lab == 5) {
        cout << setw(4) << "5. Формирование двусвязного списка размерности N," << endl;
        cout << "####@!!!$$$$Prakt2SemII%%^&&****~" << endl;
        int pod3 = 3;
        int znak=podmenu(pod3);
        cout << znak << endl;
        if (znak == 1)
        {
            int a;
            while (true)
            {
                cout << " Введите количество элементов в двусвязном списке" <<
endl;

                cin >> a;
                if (cin.fail())
                {
                    cout << "Это должно быть число" << endl;
                    cin.clear();
                    cin.ignore(32767, '\n');
                    continue;
                }
                if (a <= 0)
                {
                    cout << " Количество элементов должно быть положительным" <<
endl;

                    continue;
                }
                if (a > 77)
                {
                    cout << " X3 я так решил максимум элементов 77" << endl;
                    continue;
                }
                else break;
            }
            creat(a);
            outS(head);
        }
        if (znak == 2)
        {

```

```

        cout << " Введите значение элементов двусвязного списка окончания ввода
буква или знак" << endl;
        while (true)
        {
            int a;
            cout << " Продолжайте ввод элементов" << endl;
            cin >> a;
            if (cin.fail())
            {
                cin.clear();
                cin.ignore(32767, '\n');
                break;
            }
            if (a > 999 || a < -999)
            {
                cout << " Возможные значения <В диапазоне от -999 до 999> "
<< endl;

                continue;
            }
            if (head == NULL)
            {
                head = new list;
                head->z.a = a;
                head->next = NULL;
                head->prev = NULL;
                last = head;
            }
            else
            {
                list *tmp = last;
                last->next = new list;
                last->next->z.a = a;
                last->next->next = NULL;
                last->next->prev = tmp;
                last = last->next;
            }
        }
        auto start = chrono::system_clock::now();
        if (head == NULL)
        {
            cout << "  Вы действительно можете набирать текст со скоростью 600
символов в минуту?  " << endl;

            cout << "  Да! Но такая ерунда получается..." << endl;
            return 0;
        }
        int poi = 0;
        list *tak = head;
        while (tak)
        {
            ++poi;
            tak = tak->next;
        }
        auto end = chrono::system_clock::now();

```

```

        chrono::duration<double> diff = end - start;
        tim = diff.count();
        cout << endl;
        cout << " Ввод " << poi << " элементов закончен" << endl;
        outS(head);
    }
    if (znak == 3)
    {
        strok = shet(strok);
        ReadFileS(strok);
        outS(head);
        strok = 1;
    }
    goto Mem1;
}
if (lab == 6) {
    cout << setw(4) << "6. Определение скорости создания двусвязного списка п.5.\n";
    cout << "####@@@!!!!$$$$Prakt2SemII%%^^^&&&****~::~" << endl;
    cout << "\n" << "Скорость создания двусвязного списка = " << tim << "сек." <<
endl;

    goto Mem1;
}
if (lab == 7) {
    cout << setw(4) << "7. Вставка, удаление и получение элемента двусвязного списка."
<< endl;

    cout << "####@@@!!!!$$$$Prakt2SemII%%^^^&&&****~::~" << endl;
    int pod4 = 4;
    int znak = podmenu(pod4);
    if (znak == 1)
    {
        if (head == NULL)
        {
            cout << " Создайте двусвязный список" << endl;
        }
        cout << " Было..." << endl;
        outS(head);
        cout << " Ввидите значение элементов двусвязного списка для вставки в конец
списка, окончание ввода буква или знак" << endl;
        while (true)
        {
            int a;
            cout << " Продолжайте ввод элементов" << endl;
            cin >> a;
            if (cin.fail())
            {
                cin.clear();
                cin.ignore(32767, '\n');
                break;
            }
            if (a > 999 || a < -999)
            {
                cout << " Возможные значения <В диапазоне от -999 до 999> "
<< endl;

```

```

        continue;
    }
    auto start = chrono::system_clock::now();
    if (head == NULL)
    {
        head = new list;
        head->z.a = a;
        head->next = NULL;
        head->prev = NULL;
        last = head;
    }
    else
    {
        list *tmp = last;
        last->next = new list;
        last->next->z.a = a;
        last->next->next = NULL;
        last->next->prev = tmp;
        last = last->next;
    }
    auto end = chrono::system_clock::now();
    chrono::duration<double> diff = end - start;
    vsta = diff.count();
}
if (head == NULL)
{
    cout << "  Вы действительно можете набирать текст со скоростью 600
символов в минуту? " << endl;
    cout << "  Да! Но такая ерунда получается..." << endl;
    return 0;
}
cout << " Ввод элементов закончен" << endl;
outS(head);
}
if (znak == 2)
{
    if (head == NULL)
    {
        cout << " Создайте двусвязный список" << endl;
        cout << " ERROR!" << endl;
        return 0;
    }
    cout << " Было..." << endl;
    outS(head);
    int raz = 0;
    list *tak = head;
    while (tak)
    {
        ++raz;
        tak = tak->next;
    }
    int a;
    while (true)

```

```

{
    cout << " Ввидите индекс двусвязного списка для удаления" << endl;
    cin >> a;
    if (cin.fail())
    {
        cout << "Это должно быть число" << endl;
        cin.clear();
        cin.ignore(32767, '\n');
        continue;
    }
    if (a <= 0)
    {
        cout << " Индекс элемента должен быть положительным" <<
endl;
        continue;
    }
    if (a > raz)
    {
        cout << " Индекс элемента не может быть больше " << raz <<
endl;
        continue;
    }
    break;
}
DelSpisok(a, raz);
cout << " Результат удаления " << endl;
outS(head);
}
if (znak == 3)
{
    if (head == NULL)
    {
        cout << " Создайте двусвязный список" << endl;
        cout << " ERROR!" << endl;
        return 0;
    }
    cout << " Было..." << endl;
    outS(head);
    int raz = 0;
    list *tak = head;
    while (tak)
    {
        ++raz;
        tak = tak->next;
    }
    int a;
    while (true)
    {
        cout << " Ввидите значение элемента двусвязного списка для
удаления" << endl;

        cin >> a;
        if (cin.fail())
        {

```

```

        cout << "Это должно быть число" << endl;
        cin.clear();
        cin.ignore(32767, '\n');
        continue;
    }
    if (a > 999 || a < -999)
    {
        cout << " Возможные значения <В диапазоне от -999 до 999> "
<< endl;

        continue;
    }
    break;
}
list *tu = head;
int el = 0;
while (tu)
{
    if (tu->z.a != a)
    {
        ++el;
    }
    else
    {
        ++el;
        break;
    }
    tu = tu->next;
}
if (raz == el && last->z.a != a)
{
    cout << " Элемент со значением " << a << " в двусвязном списке
отсутствует" << endl;

    /*cout << " last->z.a=" << last->z.a << "!=a" << a << endl;*/
    goto Mem1;
}
DelSpisok(el, raz);
cout << " Элемент "<< a <<" удален из списка."<< endl;
outS(head);
}
if (znak == 4)
{
    if (head == NULL)
    {
        cout << " Создайте двусвязный список" << endl;
        cout << " ERROR!" << endl;
        return 0;
    }
    cout << " Оригинал..." << endl;
    outS(head);
    int raz = 0;
    list *tak = head;
    while (tak)
    {

```

```

        ++raz;
        tak = tak->next;
    }
    int a;
    while (true)
    {
        cout << " Ввидите индекс двусвязного списка для получения элемента
<Индексы с 1-ы>" << endl;

        cin >> a;
        if (cin.fail())
        {
            cout << "Это должно быть число" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (a <= 0)
        {
            cout << " Индекс элемента должен быть положительным <Индексы
с 1-ы>" << endl;

            continue;
        }
        if (a > raz)
        {
            cout << " Индекс элемента не может быть больше " << raz <<
endl;

            continue;
        }
        break;
    }
    list *gh = head;
    int ele=1;
    while (gh)
    {
        if(ele==a)
        {
            cout << "Значение элемента list[" << a << "]"=" << gh->z.a
<<" найдено в списке"<< endl;

            goto Mem1;
        }
        else
        {
            ++ele;
        }
        gh = gh->next;
    }
}
if (znak == 5)
{
    if (head == NULL)
    {
        cout << " Создайте двусвязный список" << endl;
        cout << " ERROR!" << endl;
    }
}

```



```

        return 0;
    }
    cout << " Оригинал..." << endl;
    outS(head);
    int raz = 0;
    list *tak = head;
    while (tak)
    {
        ++raz;
        tak = tak->next;
    }
    int a;
    while (true)
    {
        cout << " Ввидите значение элемента для поиска в списке" << endl;
        cin >> a;
        if (cin.fail())
        {
            cout << "Это должно быть число" << endl;
            cin.clear();
            cin.ignore(32767, '\n');
            continue;
        }
        if (a > 999 || a < -999)
        {
            cout << " Возможные значения <В диапазоне от -999 до 999> "
<< endl;
            continue;
        }
        else break;
    }
    quest(raz, a);
}
goto Mem1;
}
if (lab == 8) {
    cout << setw(4) << "8. Определение скорости вставки, удаления и получения элемента
двусвязного списка п.7.\n";
    cout << "####@@@!!!!$$$$Prakt2SemII%%^^^&&&***~~~~" << endl;
    cout << "Скорость вставки элементов двусвязного списка = " << vsta << "сек." <<
endl;
    cout << "Скорость удаления элементов двусвязного списка = " << iuda << "сек." <<
endl;
    cout << "Скорость получения элементов двусвязного списка = " << policay << "сек."
<< endl;
    goto Mem1;
}
if (lab == 9) {
    cout << "\n GEME OVER \n";
    return 0;
}
return 0;
}

```

