

Алгоритмы и структуры данных

Семинар 3
Связный список












Вопросы по лекции?





Что будет на уроке сегодня

-  Реализуем односвязный список
-  Реализуем функционал добавления и удаления данных в начало списка
-  Реализуем алгоритм поиска элемента в связном списке
-  Реализуем алгоритм добавления и удаления последнего элемента из связного списка
-  Преобразуем односвязный список в двусвязный список
-  Модифицируем методы добавления и удаления элементов из конца списка
-  Реализуем функцию сортировки для связного списка



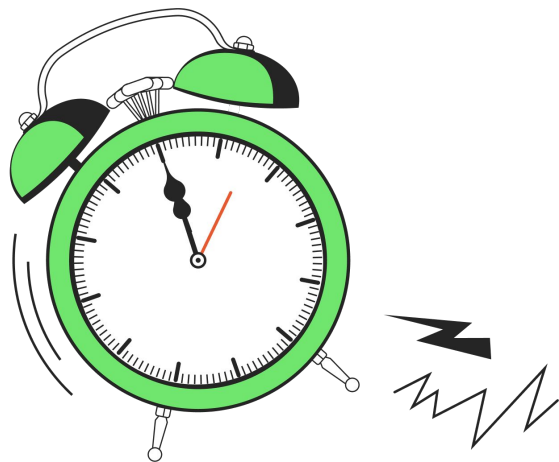
Реализуем структуру
данных: список



Реализуем односвязный список

Связный список — базовая динамическая структура данных в информатике, состоящая из узлов, содержащих данные и ссылки («связки») на следующий и/или предыдущий узел списка.

- Реализуем простой односвязный список.
- Пишем только структуру, никаких методов не требуется.



5 минут

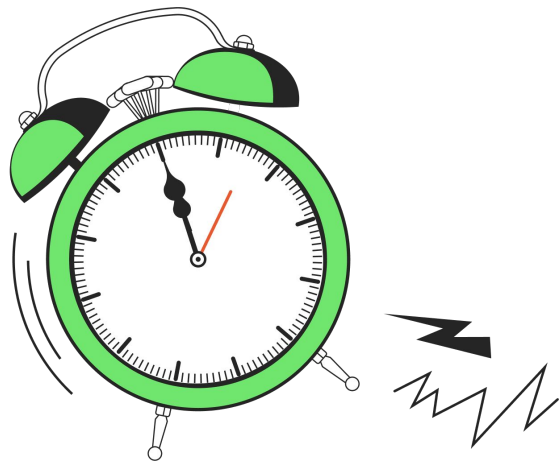


Заготовка класса List

```
1 public class List {  
2     private Node head;  
3     private class Node {  
4         private Node next;  
5         private int value;  
6     }  
7 }
```

Добавление элементов

- Реализуем метод добавления новых элементов в начало списка и удаление первого элемента связного списка.
- Односвязный список всегда имеет ссылку на первый элемент последовательности, потому именно с реализации методов для первого элемента последовательности стоит начать



10 минут

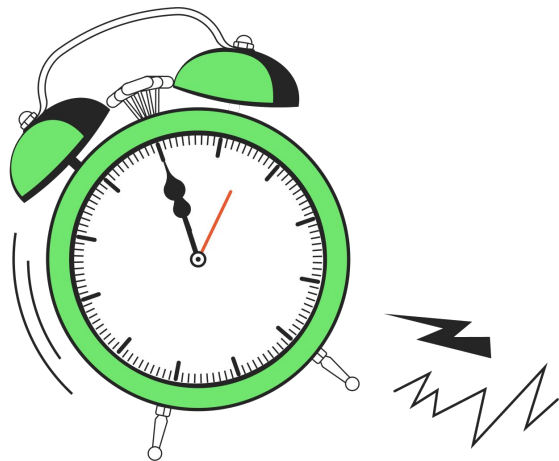


Добавление элементов

```
1 public class List {
2     private Node head;
3     public void addFirst(int value){
4         Node node = new Node();
5         node.value = value;
6         if (head != null) {
7             node.next = head;
8         }
9         head = node;
10    }
11    public void removeFirst(){
12        if (head != null){
13            head = head.next;
14        }
15    }
16    private class Node {
17        private Node next;
18        private int value;
19    }
20 }
```


Проверка вхождения элемента в список

- Реализуем метод поиска элемента в односвязном списке для проверки наличия элемента внутри списка.
- Для корректной работы со связным список необходимо понимать, как именно можно обходить все значения внутри связного списка.
- Для нашего примера проще всего будет написать метод поиска значения в связном списке и возвращения из метода информации о наличии искомого внутри списка.



10 минут

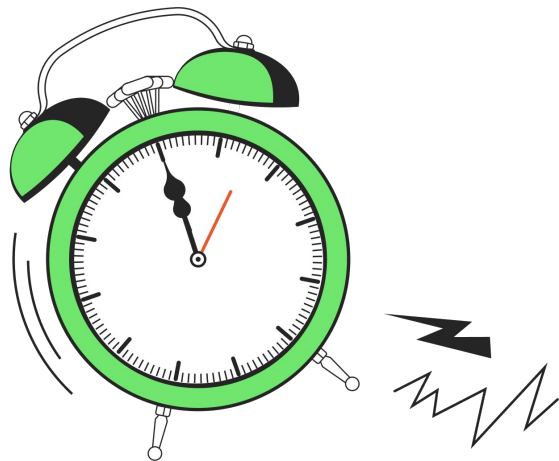


Проверка вхождения элемента в список

```
1 public boolean contains(int value)
2 {   Node node = head;
3     while (node != null){
4         if (node.value == value){
5             return true;
6         }
7         node = node.next;
8     }
9     return false;
10 }
```

Работа с концом списка

- Реализуем метод добавления новых элементов в конец списка и удаление последнего элемента связного списка.
- Теперь, когда мы понимаем, как можно искать значения внутри связного списка, мы можем сделать методы добавления и удаления элементов в конец нашего односвязного списка.



10 минут



Добавление в конец

```
1 public void addLast(int value) {  
2     Node node = new Node();  
3     node.value = value;  
4     if (head == null) {  
5         head = node;  
6     } else {  
7         Node last = head;  
8         while (last.next != null) {  
9             last = last.next;  
10        }  
11        last.next = node;  
12    }  
13 }
```



Удаление с конца

```
1 public void removeLast() {  
2     if (head != null) {  
3         Node node = head;  
4         while (node.next != null) {  
5             if (node.next.next == null) {  
6                 node.next = null;  
7                 return;  
8             }  
9             node = node.next;  
10        }  
11        head = null;  
12    }  
13 }
```

Расширяем структуру связного списка до двусвязного.

- Двусвязный список представляет из себя цепочку элементов, которые умеют ссылаться не только на следующий элемент последовательности, но и на предыдущий.
- Вносить корректировки в уже готовые методы на текущий момент не стоит, их модификацией мы займемся позднее



5 минут



Структура двусвязного списка

```
1 public class List {  
2     private Node head;  
3     private Node tail;  
4     private class Node {  
5         private Node next;  
6         private Node prev;  
7         private int value;  
8     }  
9 }
```

Обновляем методы согласно новой структуре

- Появилась дополнительная переменная, которую необходимо отслеживать во всех операциях.
- Также, благодаря ссылке на последний элемент списка, операции работы с концом стали проще и их стоит заменить на логику аналогичную работе с началом списка



10 минут



Обновляем методы взаимодействия с началом списка

```
1 public void addFirst(int value) {
2     Node node = new Node();
3     node.value = value;
4     if (head != null) {
5         node.next = head;
6         head.prev = node;
7     } else {
8         tail = node;
9     }
10    head = node;
11 }
12 public void removeFirst() {
13     if (head != null && head.next != null) {
14         head.next.prev = null;
15         head = head.next;
16     } else {
17         head = null;
18         tail = null;
19     }
20 }
```

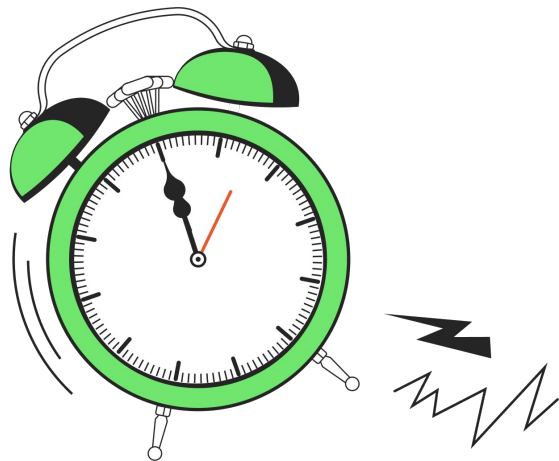


Обновляем методы взаимодействия с концом списка

```
1 public void addLast(int value) {
2     Node node = new Node();
3     node.value = value;
4     if (tail != null) {
5         node.prev = tail;
6         tail.next = node;
7     } else {
8         head = node;
9     }
10    tail = node;
11 }
12 public void removeLast() {
13     if (tail != null && tail.prev != null) {
14         tail.prev.next = null;
15         tail = tail.prev;
16     } else {
17         head = null;
18         tail = null;
19     }
20 }
```

Сортировка списка

- Можно использовать любой алгоритм, что мы использовали на предыдущем семинаре, но с точки зрения работы связного списка лучше ориентироваться на пузырьковую сортировку, т.к. она взаимодействует с соседними элементами, а не только по индексам, как делают все остальные сортировки



15 минут



Сортировка списка

```
1 public void sort() {
2     boolean needSort;
3     do {
4         needSort = false;
5         Node node = head;
6         while (node != null && node.next != null){
7             if (node.value > node.next.value){
8                 Node before = node.prev;
9                 Node after = node.next.next;
10                Node current = node;
11                Node next = node.next;
12                current.next = after;
13                current.prev = next;
14                next.next = current;
15                next.prev = before;
16                if (before != null){
17                    before.next = next;
18                } else {
19                    head = next;
20                }
21                if (after != null) {
22                    after.prev = current;
23                } else {
24                    tail = current;
25                }
26                needSort = true;
27            }
28            node = node.next;
29        }
30    } while (needSort);
31 }
```



Домашнее задание



Домашнее задание

Необходимо реализовать метод разворота связного списка (двусвязного или односвязного на выбор).





Вопросы?





Спасибо за внимание