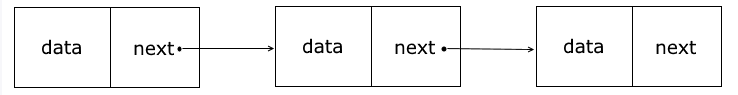
Описать класс, реализующий однонаправленный список. Тип элементов списка выбрать в соответствие с вариантом задания. Добавить следующие методы:

1. Добавление элемента в конец списка.
2. Добавление элемента в начало списка.
3. Добавление элемента в определенную позицию.
4. Удаление элемента по его значению.
5. Удаление элемента по его номеру в односвязном списке.
6. Очистка списка.
7. Поиска номера элемента в списке.
8. Просмотр списка.

Отладить программу, работающую с вышеперечисленными методами согласно варианту задания. После заполнения списка элементами реализовать возможность выбора функции, применяя оператор switch case.

|  |  |
| --- | --- |
| Вариант |  |
| 1 | Дано предложение, оканчивающееся точкой. Из символов предложения построить линейный однонаправленный список. |
| 2 | Дан одномерный массив случайным образом заданных целых чисел. Из элементов массива построить линейный однонаправленный список. |
| 3 | Дано предложение, оканчивающееся точкой. Из букв предложения построить линейный однонаправленный список. |
| 4 | Дан одномерный массив случайным образом заданных вещественных чисел. Из элементов массива построить линейный однонаправленный список. |
| 5 | Дан текст. Из предложений построить линейный однонаправленный список. |
| 6 | Дан одномерный массив слов. Из элементов массива построить линейный однонаправленный список. |
| 7 | Дан одномерный массив символов. Из элементов массива построить линейный однонаправленный список. |

Связный список представляет набор связанных узлов, каждый из которых хранит собственно данные и ссылку на следующий узел.



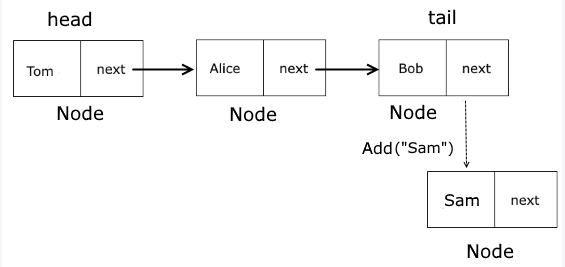
Перед созданием списка надо определить класс узла, который будет представлять одиночный объект в списке и хранить информацию о ссылке на следующий объект и собственно данные.

Один из возможных вариантов реализации однонаправленного списка в виде класса, в котором определены поля соответствующие начальному (head) и конечному (tail) узлу списка, а также количеству элементов в списке.

**Добавление элемента в конец списка**

Если не установлена переменная head (то есть список пуст), то устанавливаем head и tail. После добавления первого элемента они будут указывать на один и тот же объект.

Если же в списке есть как минимум один элемент, то устанавливаем свойство tail.Next - теперь оно хранит ссылку на новый узел. И переустанавливаем tail - теперь она ссылается на новый узел.



**Алгоритм удаления** элемента представляет следующую последовательность шагов:

1. Поиск элемента в списке путем перебора всех элементов
2. Установка свойства Next у предыдущего узла (по отношению к удаляемому) на следующий узел по отношению к удаляемому.

Для отслеживания предыдущего узла применяется переменная previous. Если элемент найден, и переменная previous равна null, то удаление идет сначала, и в этом случае происходит переустановка переменной head, то есть головного элемента.

Если же previous не равна null, то реализуются шаги выше описанного алгоритма.

