Односвязный список

17 июля 2017 г.

Какую структуру выбрать?

1. Нужно получить с консоли ряд чисел и потом вывести эти числа в том же порядке в файл, удвоив каждое число.
2. Нужно получить с консоли ряд чисел и потом вывести эти числа в обратном порядке в файл, увеличив каждое число в 10 раз.
3. Необходимо хранить набор целых чисел, количество которых в ходе выполнения программы меняется редко, а значения — часто.

Список

Список – структура данных, состоящая из узлов, каждый из которых содержит не только данные, но и одну или две ссылки на следующий и/или предыдущий узел

- односвязный
- двусвязный

Особенности

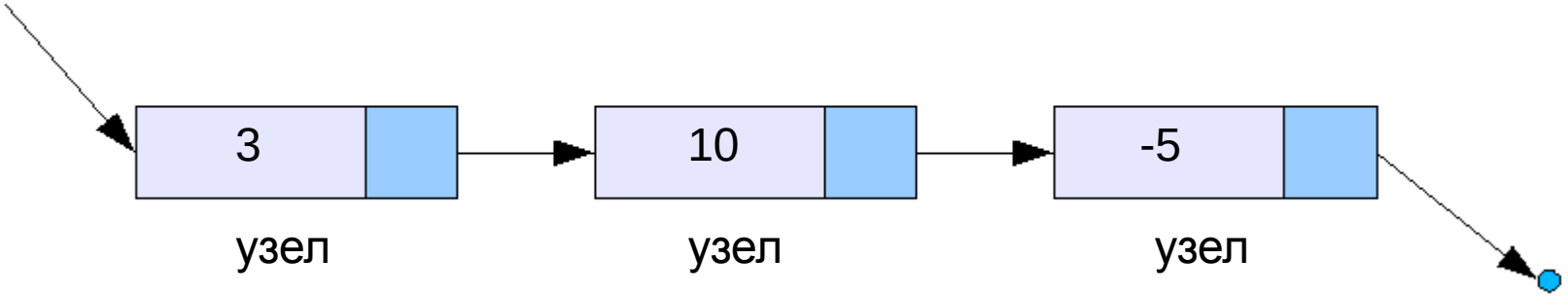
- элементы списка могут храниться не последовательно, а в разных участках памяти
- для связи предыдущего элемента со следующим хранится указатель на следующий элемент
- списки позволяют эффективно вставлять элементы в произвольную позицию списка и удалять элементы из произвольной позиции

Основные операции

- вставить элемент в список (insert)
- удалить элемент из списка (erase)
- добавить элемент в конец списка (push_back)
- удалить элемент из конца списка (pop_back)
- добавить элемент в начало списка (push_front)
- удалить элемент из начала списка (pop_front)
- узнать количество элементов (size)
- проверить на пустоту (empty)

Односвязный список

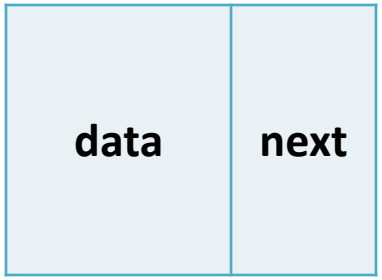
- совокупность узлов, состоящих из двух частей: значения и информации о следующем элементе списка
- только последовательный доступ к элементам
- ни один элемент не указывает на голову
- последний элемент указывает на NULL (nullptr)
- можно передвигаться только в одну сторону



Реализация: узлы

- узел представляет собой структуру (обычно вложенную)
- узел содержит поле с данными и указатель на следующий узел

```
template <typename T>
struct node {
        T data;
        node* next;
};
```



Объявление класса

- создается класс для реализации списка
- в классе дополнительно создаются поля-указатели на голову и хвост списка (head и tail)

```
template <typename T>
class MyList {
  struct node {
      T data;
      node* next;
  }
  node* head;
  node* tail;
};
```

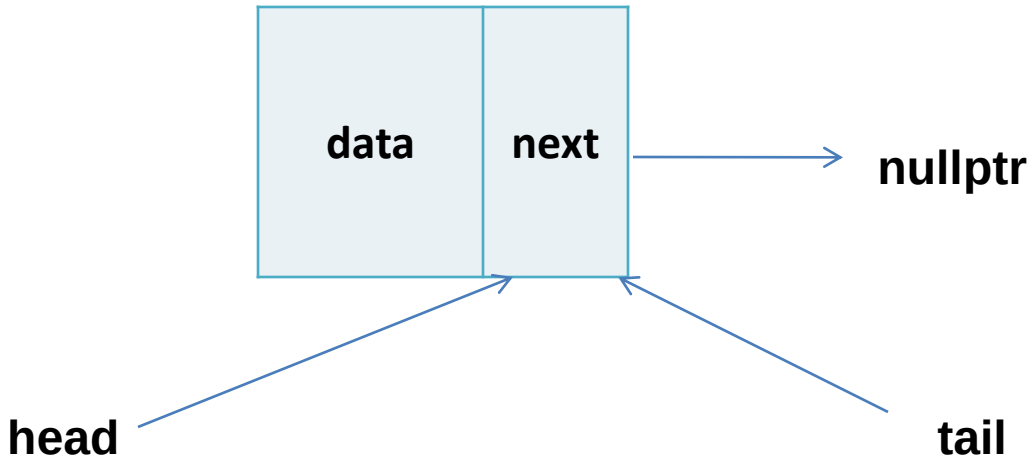
Инициализация

- в начале работы список пуст
- указатели на хвост и на голову ни на что не указывают

```
template <typename T>
MyList <T>::MyList() :
    head(nullptr),
    tail(nullptr)
{
}
```

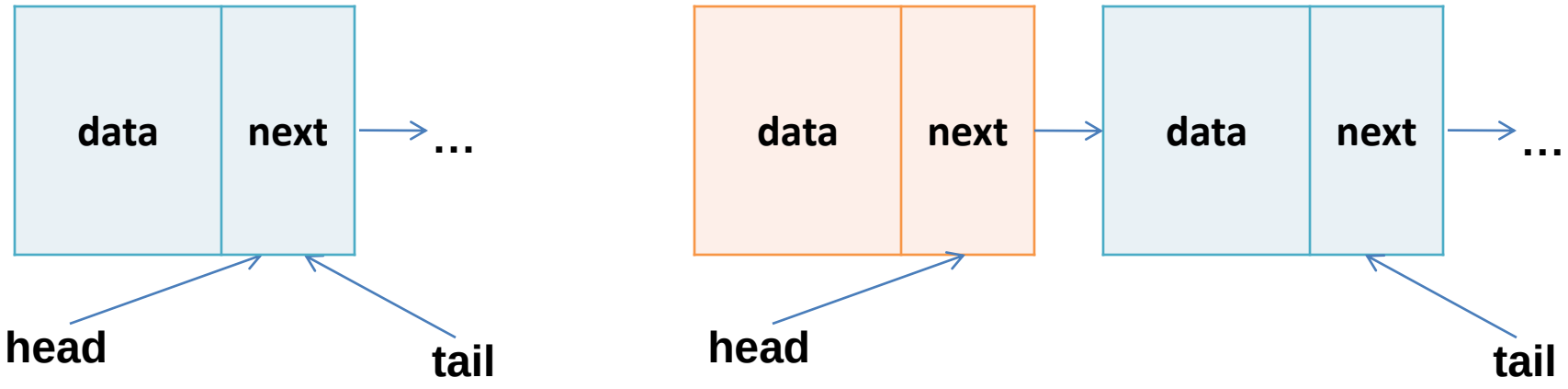
Создание первого узла

- динамически выделяется память под узел
- значение в узле устанавливается равным переданному значению
- указатель в узле устанавливается в nullptr
- head и tail указывают на этот новый элемент



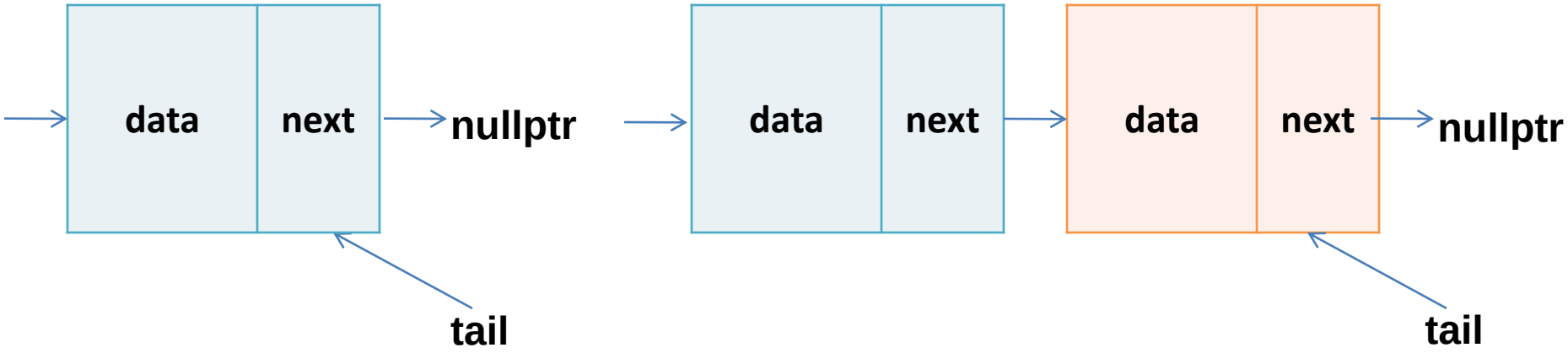
Вставка узла в начало

- значение в новом узле устанавливается равным переданному значению
- next в новом узле указывает на элемент, который прежде был первым
- head указывает на новый элемент



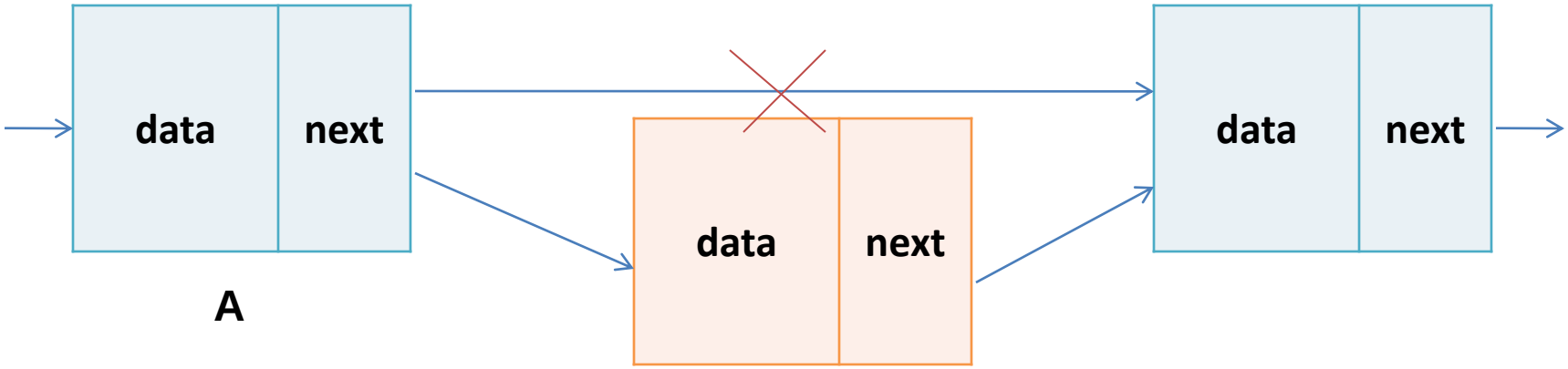
Вставка узла в конец

- значение в новом узле устанавливается равным переданному значению
- next в новом узле указывает на nullptr
- next в элементе, который был последним, указывает на новый элемент
- tail указывает на новый элемент



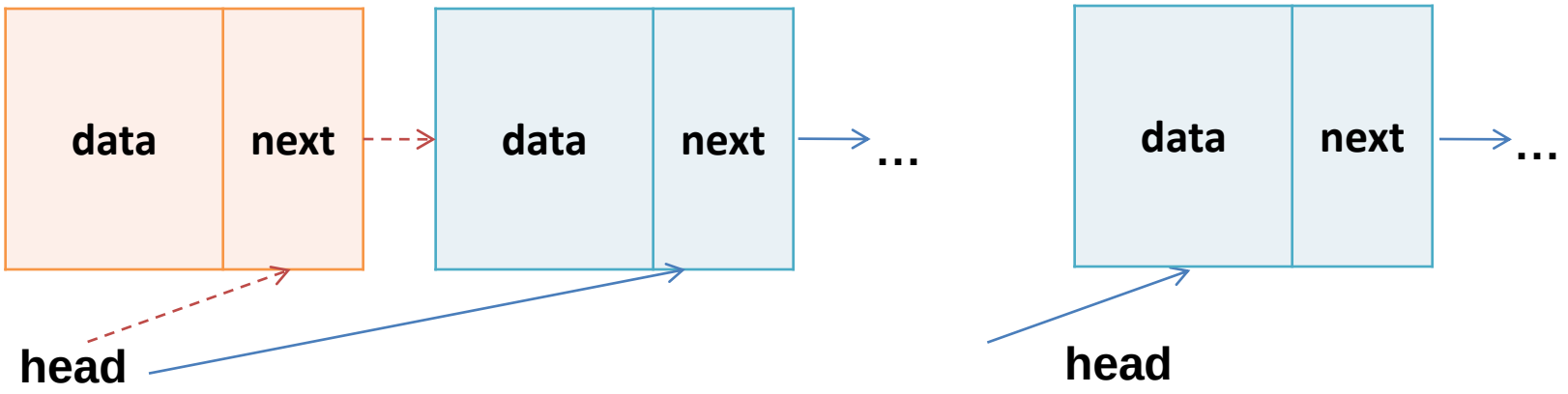
Вставка узла в середину (после существующего узла A)

- значение в новом узле устанавливается равным переданному значению
- next в новом узле указывает на узел, на который указывал next из узла A
- next в узле A указывает на новый элемент



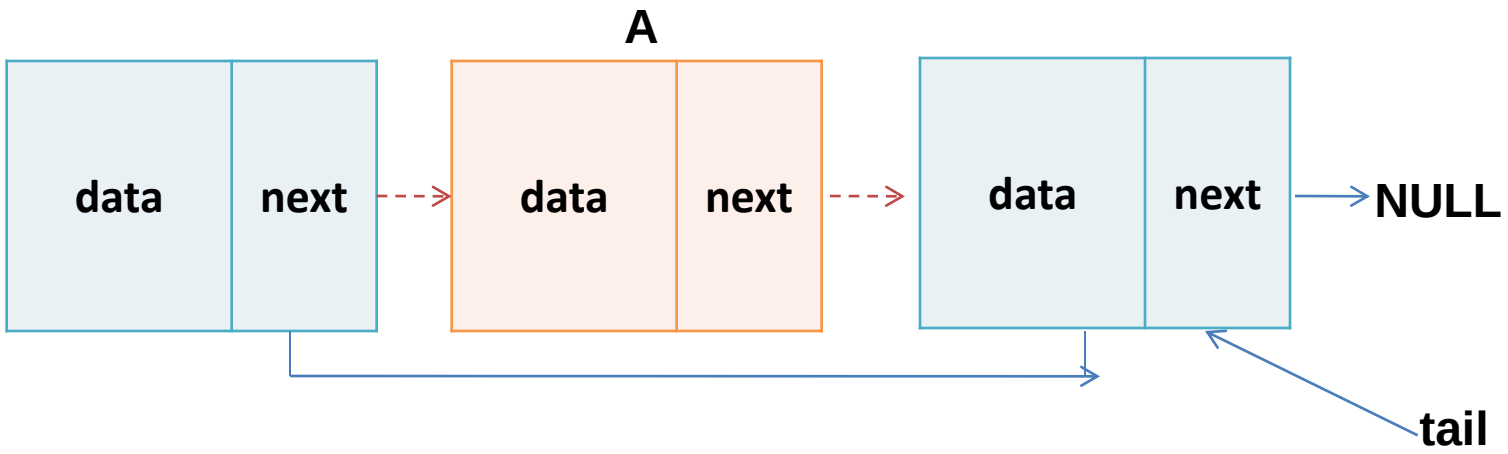
Удаление узла из начала

- head указывает на элемент, который следовал за первым
- освобождается память, которую занимал первый узел



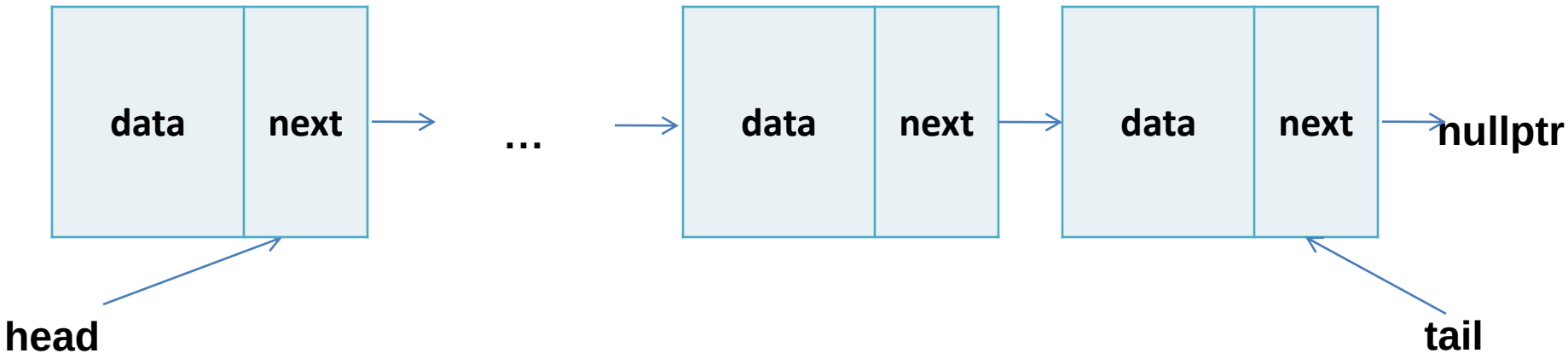
Удаление узла из середины (заданного узла A)

- указатель в предыдущем узле начинает указывать на узел, на который указывает удаляемый узел A
- освобождается память, которую занимал узел A



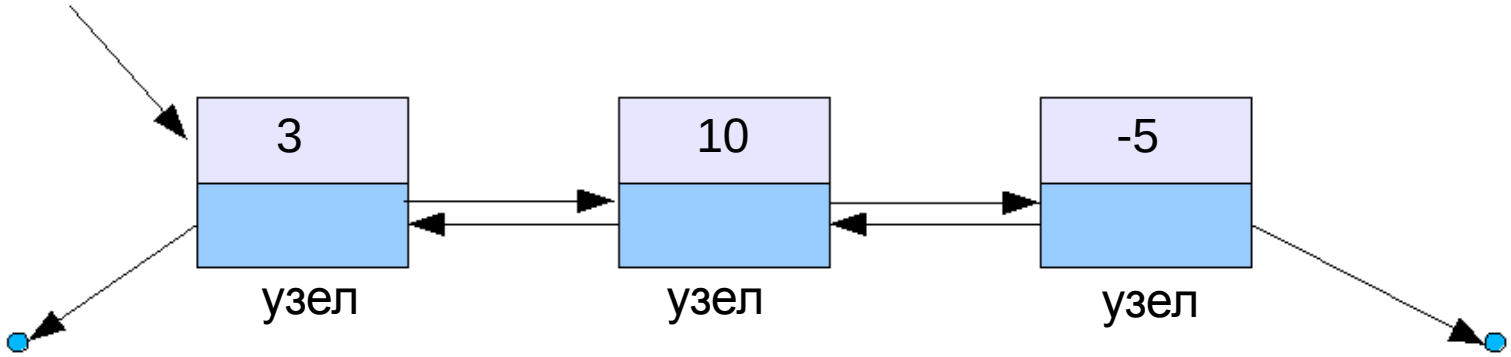
Обход списка

- обход организуется в цикле (или рекурсивно)
- создается указатель
- начиная с головы, перемещаемся по узлам по полям next, пока значение указателя не станет nullptr



Двусвязный список

- совокупность узлов, состоящих из двух частей: значения и информации о предыдущем и о следующем элементах списка
- только последовательный доступ к элементам
- можно передвигаться в обе стороны



Преимущества и недостатки списков

+

- размер списка не ограничен размером свободного последовательного участка памяти
- эффективное динамическое добавление и удаление элементов

–

- на поля-указатели расходуется дополнительная память
- осуществляется только последовательный доступ к элементам
- => сложность прямого доступа к элементу

Вопросы?