ITMO UNIVERSITY

How to Win Coding Competitions: Secrets of Champions

Week 2: Computational complexity. Linear data structures
Lecture 4: List

Pavel Krotkov
Saint Petersburg 2016

Let's define operations we need for this data structure.

Let's define operations we need for this data structure.
O(1) operations

Let's define operations we need for this data structure.
O(1) operations

- inserting an element to any place of data structure

Let's define operations we need for this data structure.
O(1) operations

- inserting an element to any place of data structure
- removing an element from any place of data structure

Let's define operations we need for this data structure.
O(1) operations

- inserting an element to any place of data structure
- removing an element from any place of data structure

Accessing element by its index can be implemented in linear time.

Consider the following structure.

Consider the following structure.

- all elements are stored separately

Consider the following structure.

- all elements are stored separately
- link to the whole structure is link to the first element (or `null` if the structure is empty)

Consider the following structure.

- all elements are stored separately
- link to the whole structure is link to the first element (or `null` if the structure is empty)
- first element stores its value and link to the second element (or `null` if the structure consists of one element)

Consider the following structure.

- all elements are stored separately
- link to the whole structure is link to the first element (or `null` if the structure is empty)
- first element stores its value and link to the second element (or `null` if the structure consists of one element)
- second element stores its value and link to the third element (or `null` if the structure consists of two elements)

Consider the following structure.

- all elements are stored separately
- link to the whole structure is link to the first element (or `null` if the structure is empty)
- first element stores its value and link to the second element (or `null` if the structure consists of one element)
- second element stores its value and link to the third element (or `null` if the structure consists of two elements)
- etc.

Consider the following structure.

- all elements are stored separately
- link to the whole structure is link to the first element (or `null` if the structure is empty)
- first element stores its value and link to the second element (or `null` if the structure consists of one element)
- second element stores its value and link to the third element (or `null` if the structure consists of two elements)
- etc.

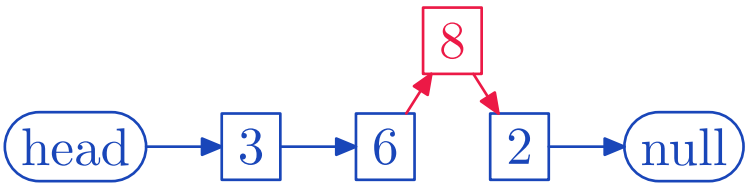
We can easily insert any element to any place in this structure.

We can easily insert any element to any place in this structure.

- insertion takes only creating a new element and changing one link

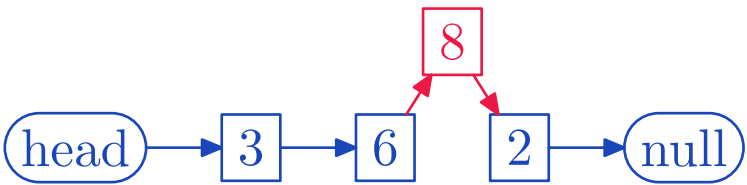
We can easily insert any element to any place in this structure.

- insertion takes only creating a new element and changing one link



We can easily insert any element to any place in this structure.

- insertion takes only creating a new element and changing one link



Note

- we need to have a link to an element after which we want to insert a new one

We also can delete any element from any place in this structure.

We also can delete any element from any place in this structure.

- deletion takes only changing one link

We also can delete any element from any place in this structure.

- deletion takes only changing one link

We also can delete any element from any place in this structure.

- deletion takes only changing one link

Note

- we need to have a link to an element previous to an element we want to delete

C++ code example

```
template <typename T>
struct list_node {
  list_node *next;
  T value;
};
```

C++ code example

```
template <typename T>
struct list_node {
  list_node *next;
  T value;
};
```

Java code example

```
class ListNode <T> {
  ListNode next;
  T value;
}
```

```
insert(p, new)
  new.next ← p.next
  p.next ← new
```

```
insert(p, new)
  new.next ← p.next
  p.next ← new


delete_next(p)
  p.next ← p.next.next
```

You should be very careful with potential `null` references while implementing this data structure.

You should be very careful with potential `null` references while implementing this data structure.

List can also be doubly linked.

- each node store links to next and previous nodes

Thank you
for your attention!