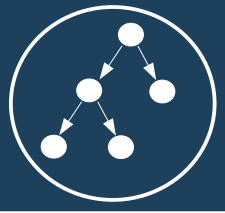


Data Structures and Algorithms

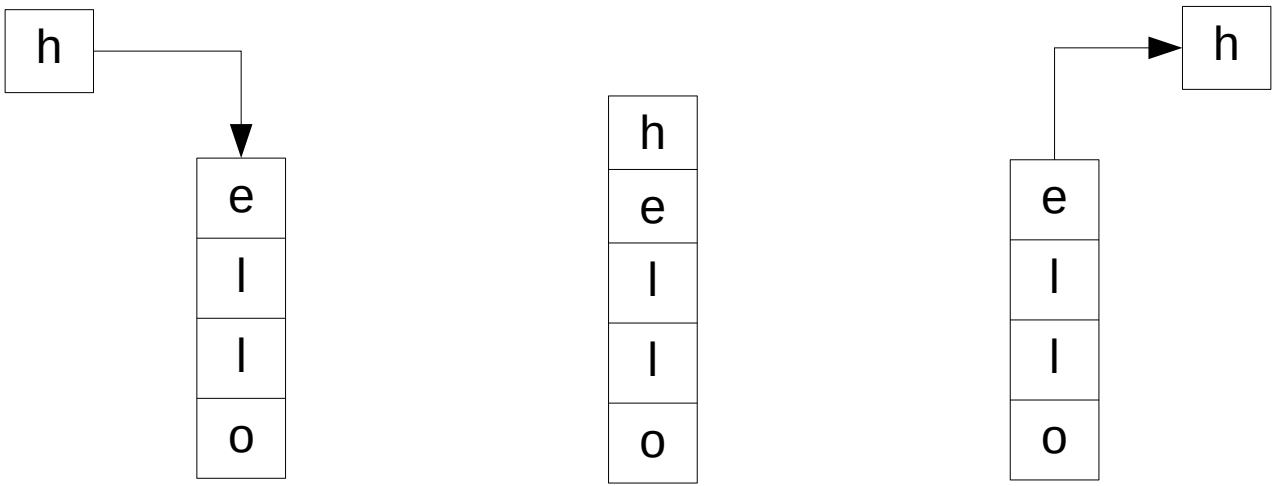
Структуры данных. Стек на основе связанного списка

Стек

Стек — это абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»). Стек является динамической структурой данных.

Поддерживаемые операции:

- Добавление элемента в вершину стека (push)
- Удаление элемента из вершины стека (pop)
- Получение элемента с вершины стека без удаления (peek)
- Получение размера стека (size)

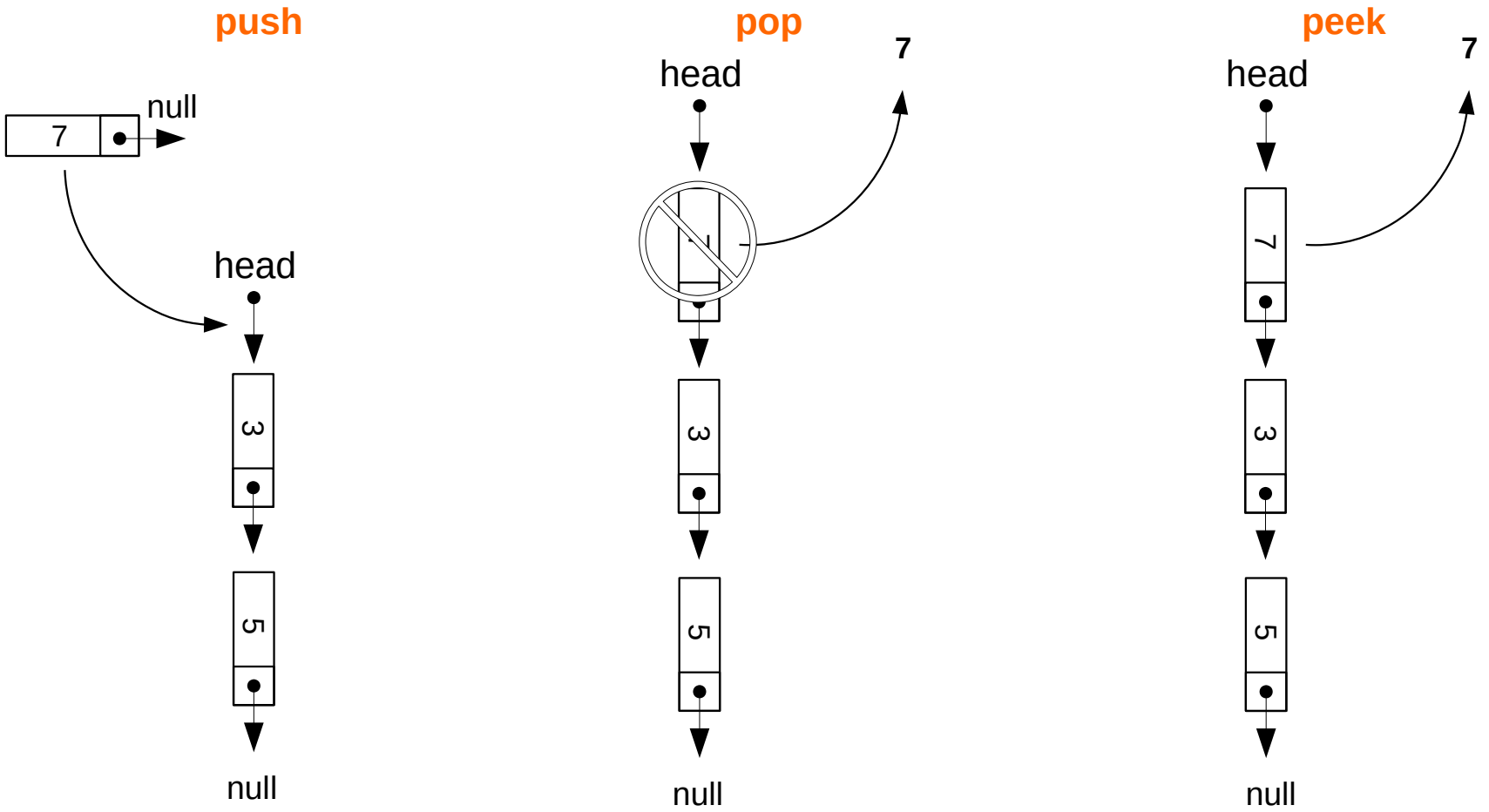


Реализация стека с помощью односвязного списка

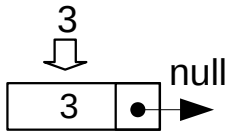
Для односвязного списка наиболее эффективными операциями являются операции добавления и удаления элемента из начала списка. Это дает возможность реализовать стек на основе связанного списка. Такая реализация будет оптимальной и производительной. Для реализации стека стоит использовать следующие операции односвязного списка:

- push (добавление элемента в голову списка)
- pop (удаление элемента из головы списка)
- peek (получение значение из головы списка)
- size (получение размера списка)

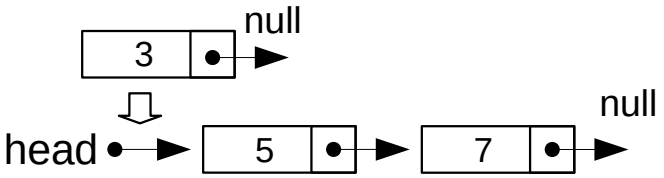
Изображение реализации стека с помощью связанного списка



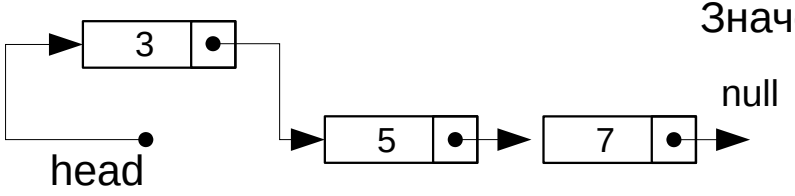
Добавление значения в стек

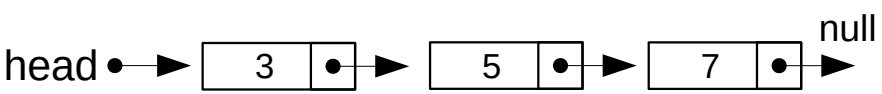


При добавлении нового значения нужно создать новый узел который хранит добавляемое значение и next = null

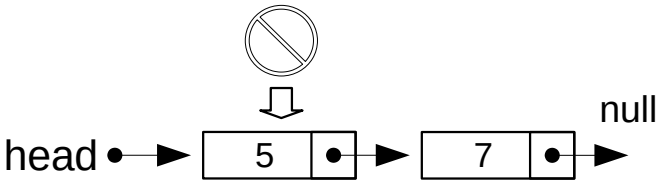


Установить значение next добавляемого узла равное head. Значение head установить равным ссылке на добавляемый узел.

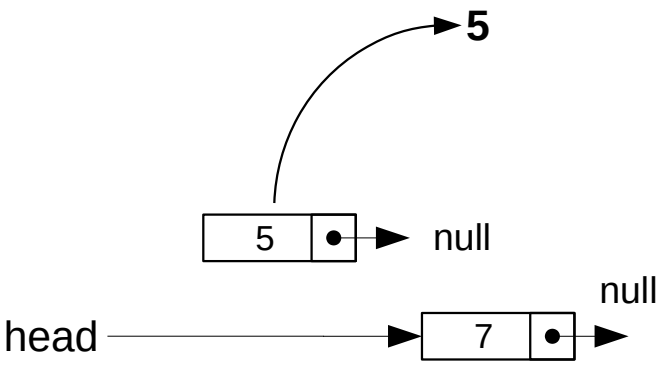




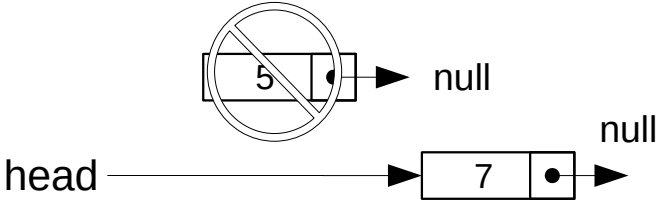
Получение значения с удалением



Проверить значение head на пустоту. Если head не указывает на узел, то закончить (стек пуст).

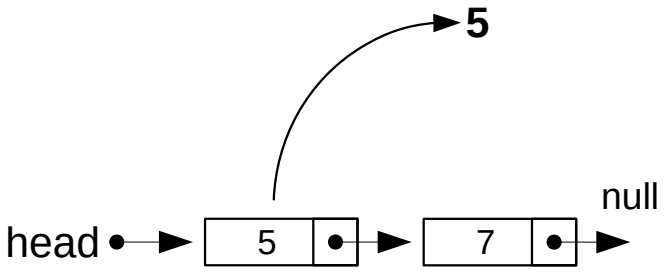


Установить значение head равное значению next удаляемого элемента. Указать, значение next удаляемого узла равным null. Сохранить для возврата значение удаляемого узла.



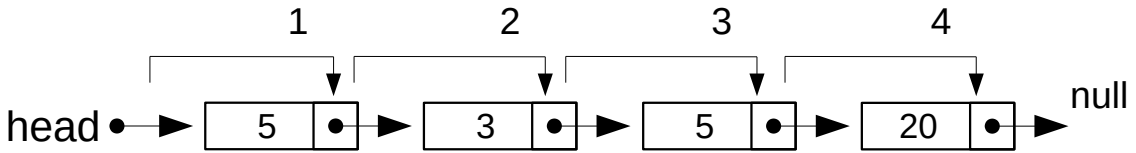
Освободить память занимаемую удаляемым узлом.

Получение значения без удаления



Проверить значение head на пустоту. Если ссылка не пустая, то вернуть значение данных в узле.

Получение размера



Для получения размера стоит объявить переменную с начальным значением 0. Начиная с начала списка выполнять переход по ссылке к следующему узлу. На каждом переходе увеличивать значение этой переменной на 1. Закончить на узле для которого next == null.

Python

Реализация на Python

Описание узла и стека

```
class Node:
    def __init__(self, data=None, next=None):
        self.data = data
        self.next = next

    def __str__(self):
        return str(self.date)


class Stack:

    def __init__(self):
        self.head = None
```

Метод добавления

```
def push(self, value):
        new_node = Node(value)
        new_node.next = self.head
        self.head = new_node
```

Методы для получения с удалением и без

```
def pop(self):
    if self.head is None:
        return
    value = self.head.data
    self.head = self.head.next
    return value


def peek(self):
    if self.head is None:
        return
    value = self.head.data
    return value
```

Метод для получения размера

```
def size(self):
    length = 0
    current_node = self.head
    while current_node is not None:
        length += 1
        current_node = current_node.next
    return length
```

Реализация на Java

Описание узла и стека

```
class Stack {
     private class Node {
          String date;
          Node next;

          public Node(String date, Node next) {
               this.date = date;
               this.next = next;
          }

          public Node() {
          }
     }

     private Node head;

     public Stack() {
          super();
     }
```

Метод добавления

```
public void push(String value) {
     Node newNode = new Node(value, head);
     head = newNode;
}
```

Методы получения с удалением и без

```
public String pop() {
     if (head != null) {
          String result = head.date;
          head = head.next;
          return result;
     }
     return null;
}

public String peek() {
     if (head != null) {
          String result = head.date;
          return result;
     }
     return null;
}
```

Метод для получения размера

```
public long size() {
     long size = 0;
     for (Node currentNode = head; currentNode != null; currentNode = currentNode.next) {
          size += 1L;
     }
     return size;
}
```

Реализация на Fortran

Описание узла и стека

```
type Node
    integer::data_value
    class(Node),pointer::next=>null()
end type Node

type Stack
    class(Node), pointer::head=>null()

    contains
        procedure,pass::push
        procedure,pass::pop
        procedure,pass::peek
        procedure,pass::stack_size
        procedure,pass::clear
        procedure,pass::print_stack
end type Stack
```

Метод добавления

```
subroutine push(this, data_value)
    class(Stack)::this
    integer, intent(in)::data_value
    class (Node), pointer::new_node => null()
    allocate(new_node)
    new_node%next => null()
    new_node%data_value = data_value
    new_node%next => this%head
    this%head => new_node
end subroutine push
```

Методы получения с удалением и без

```
integer function pop(this, res)
    class(Stack), intent(inout)::this
    class(Node), pointer::current_node
    logical,intent(inout)::res
    res = .false.
    if(.not. associated (this%head)) then
        return
    end if
    pop = this%head%data_value
    current_node => this%head%next
    this%head%next => null()
    deallocate (this%head)
    this%head => current_node
    res = .true.
end function pop

integer function peek(this, res)
    class(Stack), intent(inout)::this
    logical,intent(inout)::res
    res = .false.
    if(.not. associated (this%head)) then
        return
    end if
    peek = this%head%data_value
    res = .true.
end function peek
```

Метод для получения длины

```
integer function stack_size(this)
    class(Stack),intent(inout)::this
    class(Node), pointer::current_node
    stack_size = 0
    current_node => this%head
    do
        if(.not. associated(current_node)) then
            exit
        end if
        stack_size = stack_size + 1
        current_node => current_node%next
    end do
end function stack_size
```

Список литературы

1) Роберт Седжвик, Кевин Уэйн «Алгоритмы на java 4-е издание» Пер. с англ. - М. : ООО "И.Д. Вильямс", 2013. ISBN 978-5-8459-1781-2.