Стек и очередь

Булгаков Илья, Гусев Илья

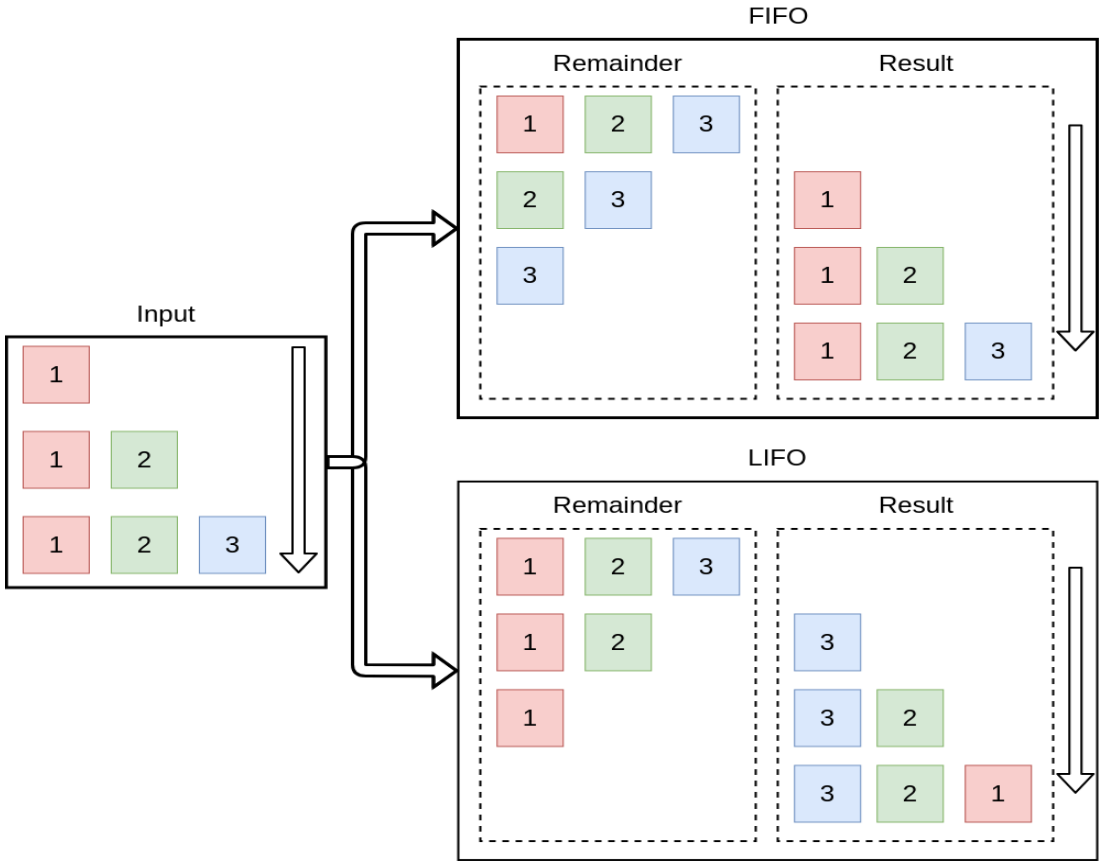
Московский физико-технический институт

Москва, 2020

Содержание

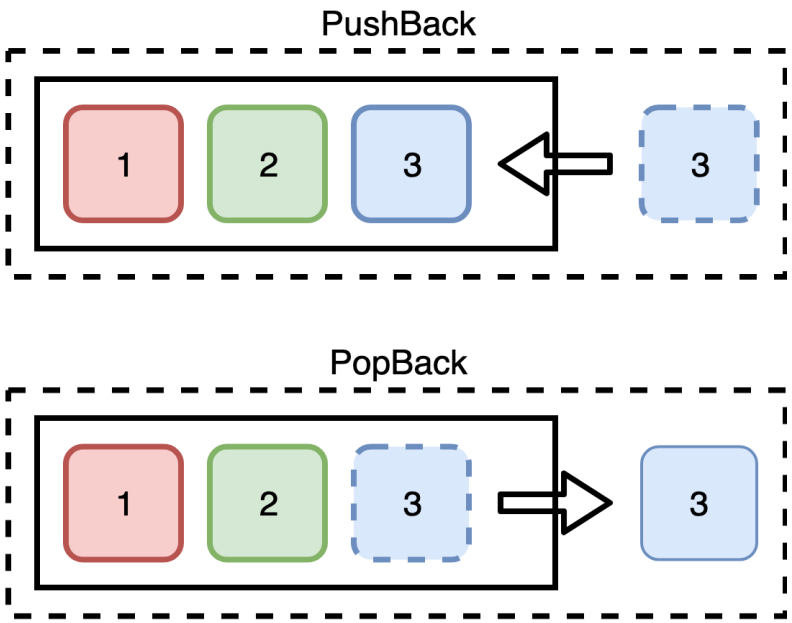
Принципы LIFO vs FIFO

LIFO и FIFO - подходы к порядку обработки элементов. Применяются в разных сферах (даже бухгалтерском учете!)
LIFO = last in first out (последний вошёл, первый вышел)
FIFO = first in first out (первый вошёл, первый вышел)



Стек

АТД по LIFO принципу

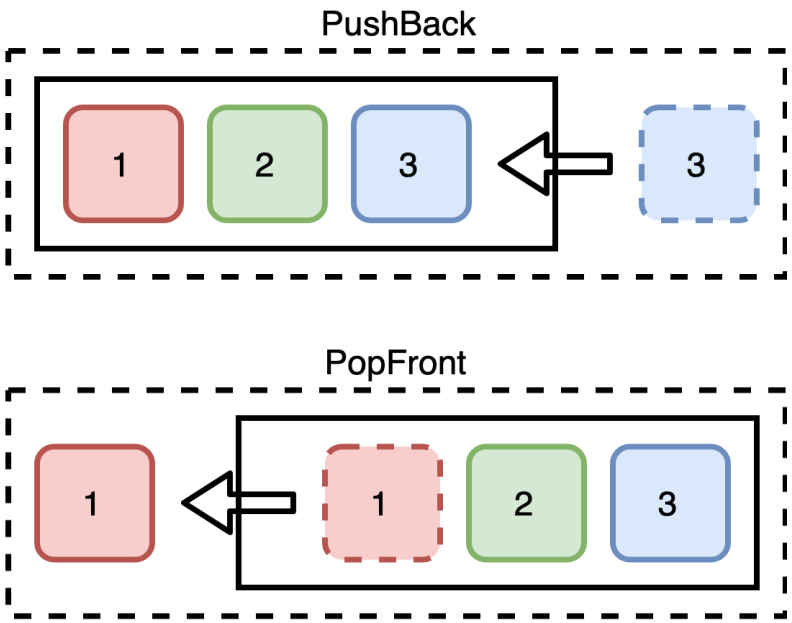


Интерфейс для стека int'ов:

```
// Вставка элемента в конец
void PushBack(Stack* stack, int
    element);
// Снятие элемента с конца
int PopBack(Stack* stack);
// Проверка на пустоту
bool IsEmpty(Stack* stack);
```

Очередь

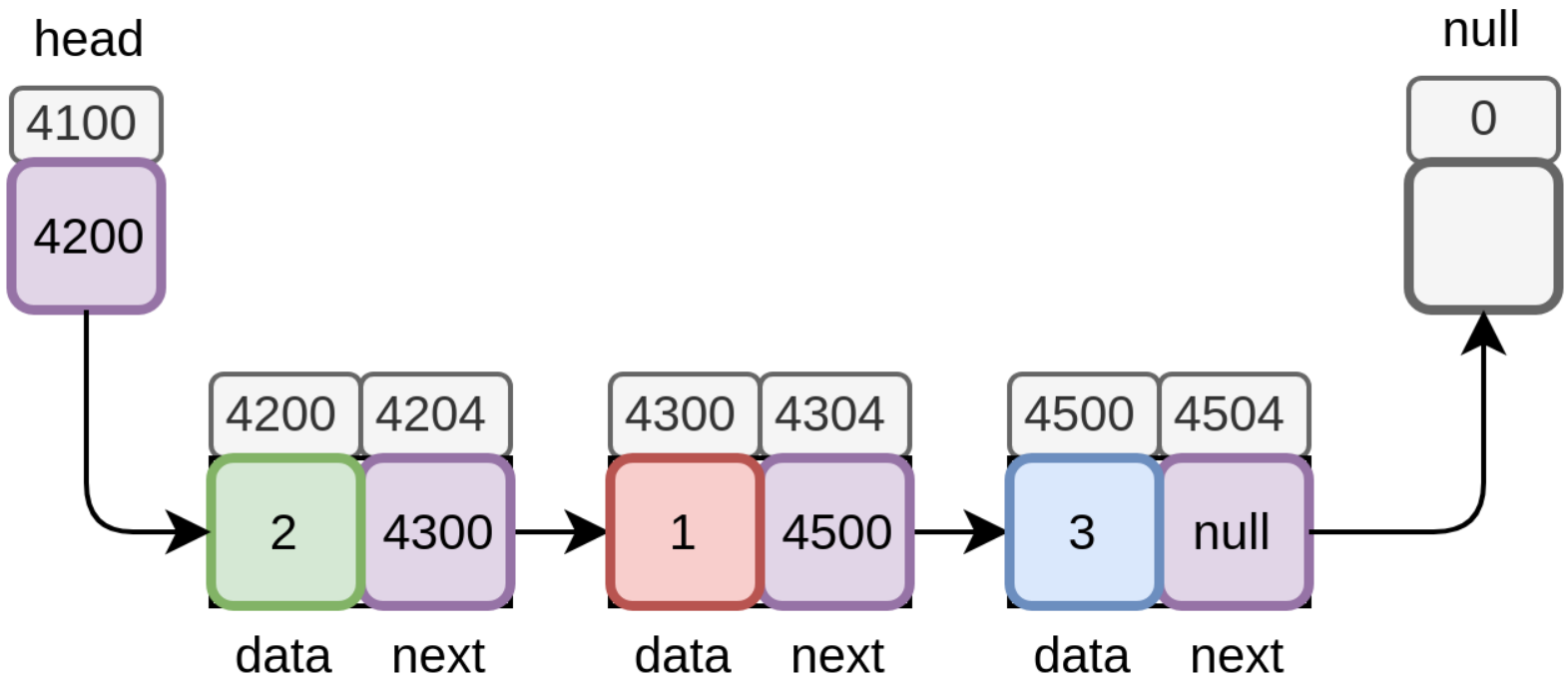
АТД по FIFO принципу



Интерфейс для очереди int'ов:

```
// Вставка элемента в конец
void PushBack(Queue* queue, int
    element);
// Снятие элемента с начала
int PopFront(Queue* queue);
// Проверка на пустоту
bool IsEmpty(Queue* queue);
```

Однонаправленный связный список




```
struct Node {
    int data;
    struct Node* next;
};
typedef struct Node Node;
```

```
struct SingleLinkedList {
    struct Node* head;
};
typedef struct SingleLinkedList
    SingleLinkedList;
```

Однонаправленный связный список

Реализация стека

```
void PushBack(SingleLinkedList* list, int element) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = element;
    newNode->next = list->head;
    list->head = newNode;
}

int PopBack(SingleLinkedList* list) {
    Node* backNode = list->head;
    assert(backNode != NULL);
    list->head = backNode->next;
    int data = backNode->data;
    free(backNode);
    return data;
}

bool IsEmpty(SingleLinkedList* list) {
    return (list->head == NULL);
}
```

Однонаправленный связный список

Сложность

Интерфейс стека:

- PushBack: $O(1)$
- PopBack: $O(1)$
- IsEmpty: $O(1)$

Другие операции:

- PushFront: $O(n)$
- PopFront: $O(n)$
- GetByIndex: $O(n)$
- Find: $O(n)$
- ExtractMax: $O(n)$

Реализация на массиве

Если известно число элементов в стеке, либо число элементов небольшое (например $<= 10^5$ элементов), то его можно реализовать на простом массиве.

Реализация на массиве

```
int stack[100];
int n = 100;
int top = -1;
void push(int val) {
    if(top>=n-1)
    cout<<"Stack Overflow"<<endl;
    else {
        top++;
        stack[top]=val;
    }
}
void pop() {
    if(top<=-1)
    cout<<"Stack Underflow"<<endl;
    else {
        cout<<"The popped element is "<< stack[top] <<endl;
        top--;
    }
}
```

Стек и очередь в библиотеке C++

std::stack

std::stack - шаблонный контейнер в стандартной библиотеке. Позволяет работать с разными типами.
Операции:

- top()
- empty()
- size()
- push()
- pop()

```
stack <int> s;
s.push(10);
s.push(30);
cout << s.size();
cout << s.top();
s.pop();
```

Стек и очередь в библиотеке C++

другие контейнеры

По такому же принципу работают и другие контейнеры
Операции:

- std::stack
- std::queue
- std::deque

```
stack <int> s;
s.push(10);
s.push(30);
cout << s.size();
cout << s.top();
s.pop();
```

Полезные ссылки I

Nearly All Binary Searches and Mergesorts are Broken
https://bit.ly/2MdGqfU

Т.Кормен, Ч.Лейзерсон, Р.Ривест, К.Штайн - Алгоритмы. Построение и анализ. Глава 3
https://bit.ly/2wFzphU