Лекция 7. Стеки и очереди

Казахский национальный университет имени аль-Фараби

Кафедра компьютерных наук

Алгоритмы и структуры данных

Лектор: старший преподаватель, PhD, Кенжебек Ержан Ғалымжанұлы

Цель и план лекции

Цель лекции

Изучить структуры данных «стек» и «очередь», стратегии LIFO и FIFO, их операции, а также реализацию на массивах и в стандартной библиотеке шаблонов C++ (STL).

План лекции

1. Контейнеры и динамические множества
2. Стек: операции Push и Pop
3. Реализация стека на массиве
4. Очередь: операции Push и Pop
5. Контейнеры stack и queue в C++ STL

Стеки и очереди

- Термин контейнер (container) обозначает структуру данных, позволяющую хранить и извлекать данные независимо от содержимого. В противоположность контейнерам, словари представляют собой абстрактные типы данных, которые извлекаются по ключевому значению или содержимому.
- Контейнеры различаются по поддерживаемому ими типу извлечения данных. В двух наиболее важных типах контейнеров порядок извлечения зависит от порядка помещения:
  - Стеки
  - Очереди

- Стеки и очереди представляют собой динамическое множества, элементы из которых удаляются с помощью предварительно определенной операцией Delete.
  - Первым из стека (**stack**) удаляется элемент, который был помещен туда последним: **стратегия Last-In, First-Out.**
  - В очереди (**queue**) всегда удаляется элемент, который содержится в множестве больше других: **стратегия First-In, First-Out.**

Stack

- Операции вставки Insert применительно к стекам называется записью Push, а операция удаления Delete называется Pop. Стек обладает атрибутом top, который предоставляет **индекс** последнего помещенного в стек элемента. Пример:

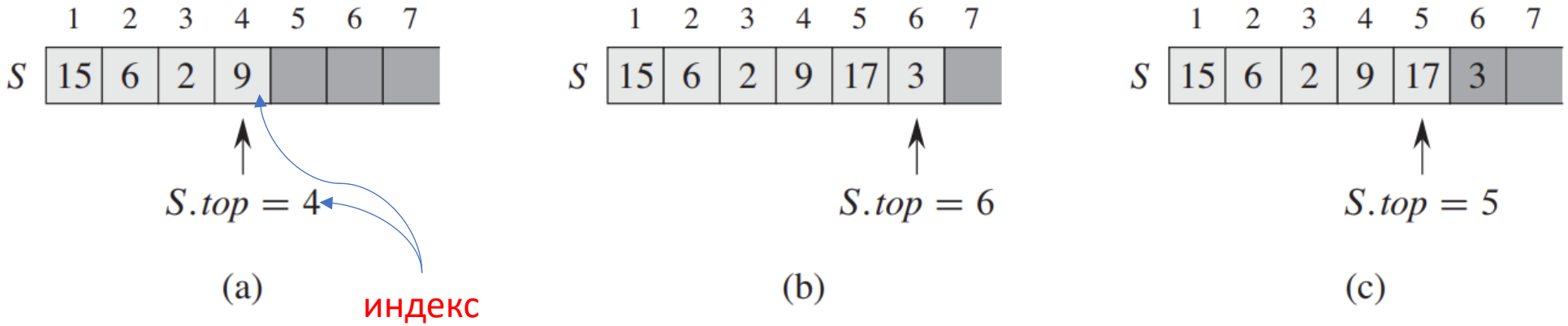


Figure 1. An array implementation of a stack S. Stack elements appear only in the lightly shaded positions.

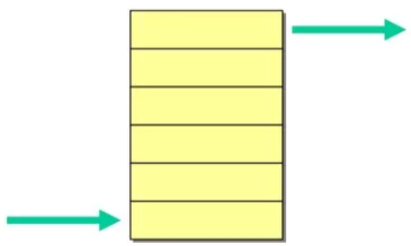
(a) Stack S has 4 elements**.** The top element is 9.

(b) Stack S after the calls PUSH(S, 17) and PUSH(S , 3).

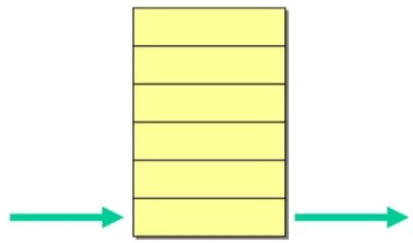
(c) Stack S after the call POP(S) has returned the element 3, which is the one most recently pushed. Although element 3 still appears in the array, it is no longer in the stack; the top is element 17.

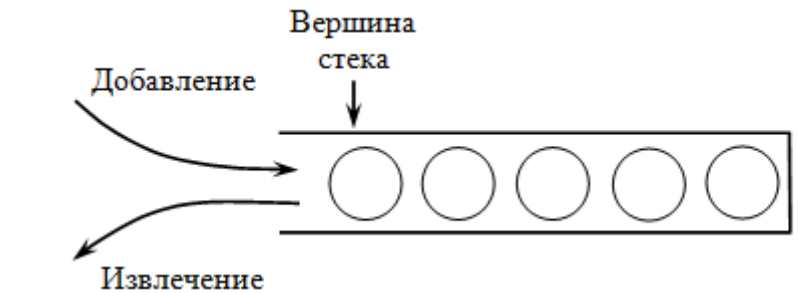
- Стек состоит из элемнетов S[1, …, S.top], где S[1] – элемент на дне стека, а S[top] – элемент на его вершине.

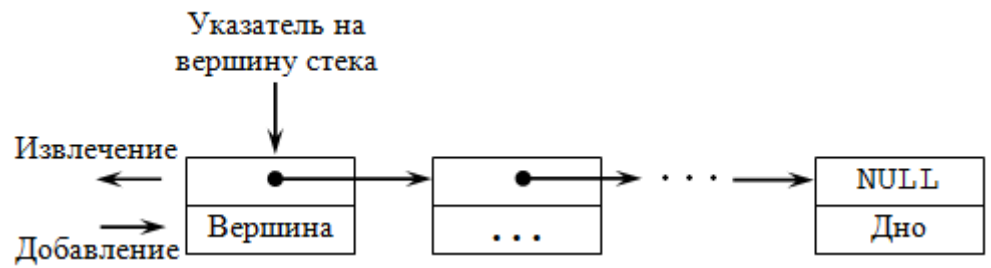
Очередь (FIFO)



Стек (LIFO)







Операции стека

- Stack-Empty(S). Если S.top = 0, то стек не содержит ни одного элемента и является пустым (empty). Операция проверки Stack-Empty(S):

```
Stack-Empty(S)
1  if S.top == 0
2     return true
3  else return false
```

- Запись в стек. Операция Push ()

```
Push(S,x)
1  S.top = S.top + 1
2  S[S.top] = x
```

- Снятие со стека. Операция Pop():

```
Pop(S)
1  if Stack-Empty(S)
2    error “underflow”
3  else S.top = S.top - 1
4    return S[S.top + 1]
```

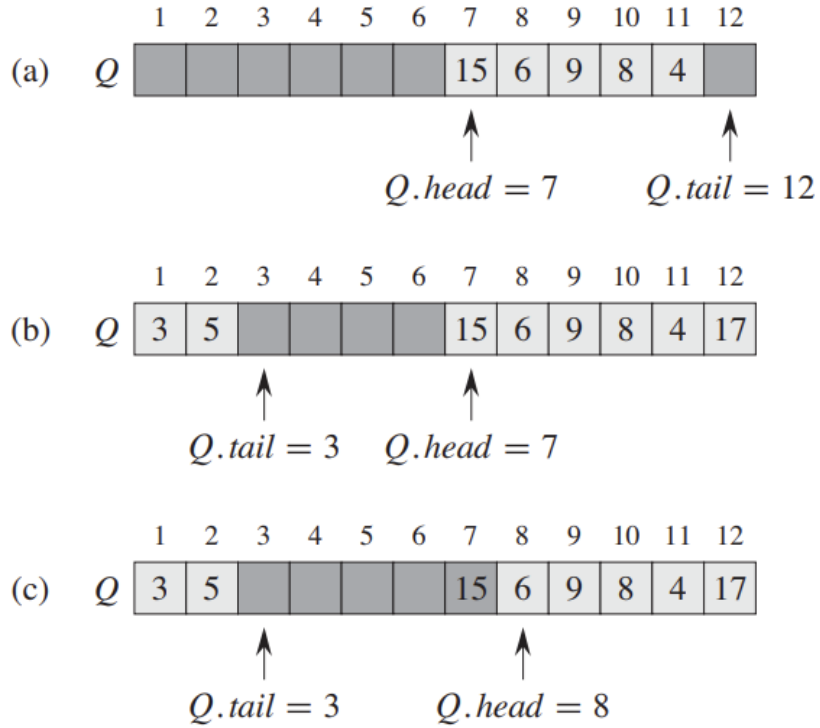
O(1)

Если выполняется попытка снятие элемента с пустого стека, говорят что он опустошается (underflow), что приводит к ошибке.

Если значение S.top больше n, то стек переполняется (overflow)

Очереди (queue)

- Операции вставки применительно к очереди называется записью Push, а операция удаления Delete называется Pop. Благодаря свойству FIFO, очередь например похож на живой очередь в поликлинике. У очереди имеется голова (head) и хвост (tail). Когда элемент помещается в очередь, он занимает место в хвосте очереди. Из очереди всегда выводится элемент, который находится в ее головной части. Пример:



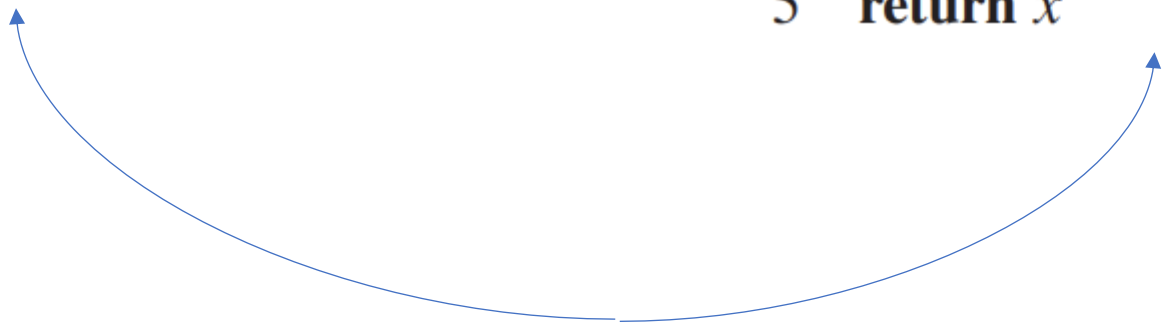
Операции очереди

Push

```
1  Q[Q.tail] = x
2  if Q.tail == Q.length
3        Q.tail = 1
4  else Q.tail = Q.tail + 1
```

Pop

```
1  x = Q[Q.head]
2  if Q.head == Q.length
3        Q.head = 1
4  else Q.head = Q.head + 1
5  return x
```



O(1)

Stack в C++ STL(Standard Template Library)

```
#include <iostream>
#include <stack>
using namespace std;
int main() {
    stack<int> stack;
    stack.push(21);
    stack.push(22);
    stack.push(24);
    stack.push(25);

    stack.pop();
    stack.pop();

    while(!stack.empty()) {
        cout << stack.top() <<" ";
        stack.pop();
    }
}
```

stack операции:

empty()
size()
top()
push(a)
pop()



O(1)

Queue в C++ STL(Standard Template Library)

```
 // Print the queue
void showq(queue<int> gq)
{
    queue<int> g = gq;
    while (!g.empty()) {
        cout << '\t' << g.front();
        g.pop();
    }
    cout << '\n';
}
int main(){
queue<int> q;
    q.push(10);
    q.push(20);
    q.push(30);
    cout << "The queue q is : ";
    showq(q);

    cout << "\n q.size() : " << q.size();
    cout << "\n q.front() : " << q.front();
    cout << "\n q.back() : " << q.back();

    cout << "\n q.pop() : ";
    q.pop();
    showq(q);
    return 0;
}
```

queue операции:

empty()
size()
emplace()
front()
back()
push(a)
pop()



O(1)

Выводы

1. Стек и очередь — динамические множества, в которых порядок удаления элементов определён заранее.
2. Стек работает по принципу LIFO («последним пришёл — первым вышел»), очередь — по принципу FIFO («первым пришёл — первым вышел»).
3. При реализации на массиве операции стека и очереди выполняются за O(1).
4. Необходимо учитывать опустошение (underflow) и переполнение (overflow) структуры.
5. В C++ готовые реализации предоставляют контейнеры std::stack и std::queue.

Контрольные вопросы

1. Чем контейнер отличается от словаря?
2. Сравните стратегии LIFO и FIFO. Приведите примеры.
3. Какие операции определены для стека? Для чего нужен атрибут top?
4. Что такое опустошение и переполнение стека?
5. Как устроена очередь и какие операции для неё определены?
6. Какие методы предоставляют контейнеры stack и queue в C++ STL?

Список литературы

Основная литература

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 3-е изд. — М.: Вильямс, 2013.
2. Вирт Н. Алгоритмы и структуры данных. — М.: ДМК Пресс, 2010.
3. Седжвик Р., Уэйн К. Алгоритмы на Java. 4-е изд. — М.: Вильямс, 2013.

Дополнительная литература

4. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. — Addison-Wesley, 1983.
5. Страуструп Б. Программирование: принципы и практика с использованием C++. 2-е изд. — М.: Вильямс, 2016.
6. Бхаргава А. Грокаем алгоритмы. — СПб.: Питер, 2017.

Электронные ресурсы

7. VisuAlgo — визуализация алгоритмов и структур данных: visualgo.net
8. Справочник по C++ и STL: en.cppreference.com

Спасибо за внимание!