Динамические структуры данных

15 июня 2017 г.

Динамические структуры данных

- применяются тогда, когда заранее неизвестно, сколько памяти нужно выделить
- позволяют во время выполнения программы выделять память при добавлении новых элементов и освобождать память – при удалении ненужных
- используется динамическое распределение памяти (память под отдельные элементы выделяется при выполнении программы)

Особенности динамической структуры данных

- память выделяется при выполнении программы, а не при компиляции
- количество элементов структуры может не фиксироваться
- размер структуры может изменяться во время выполнения программы
- в процессе выполнения программы могут меняться связи между элементами структуры

Указатель на динамическую структуру

- каждой динамической структуре данных сопоставляется указатель (адрес структуры)
- доступ к структуре осуществляется через этот указатель
- при компиляции память выделяется только под указатель

Основные виды динамических структур

- стек
- очередь
- список (односвязный или двусвязный)
- бинарное дерево

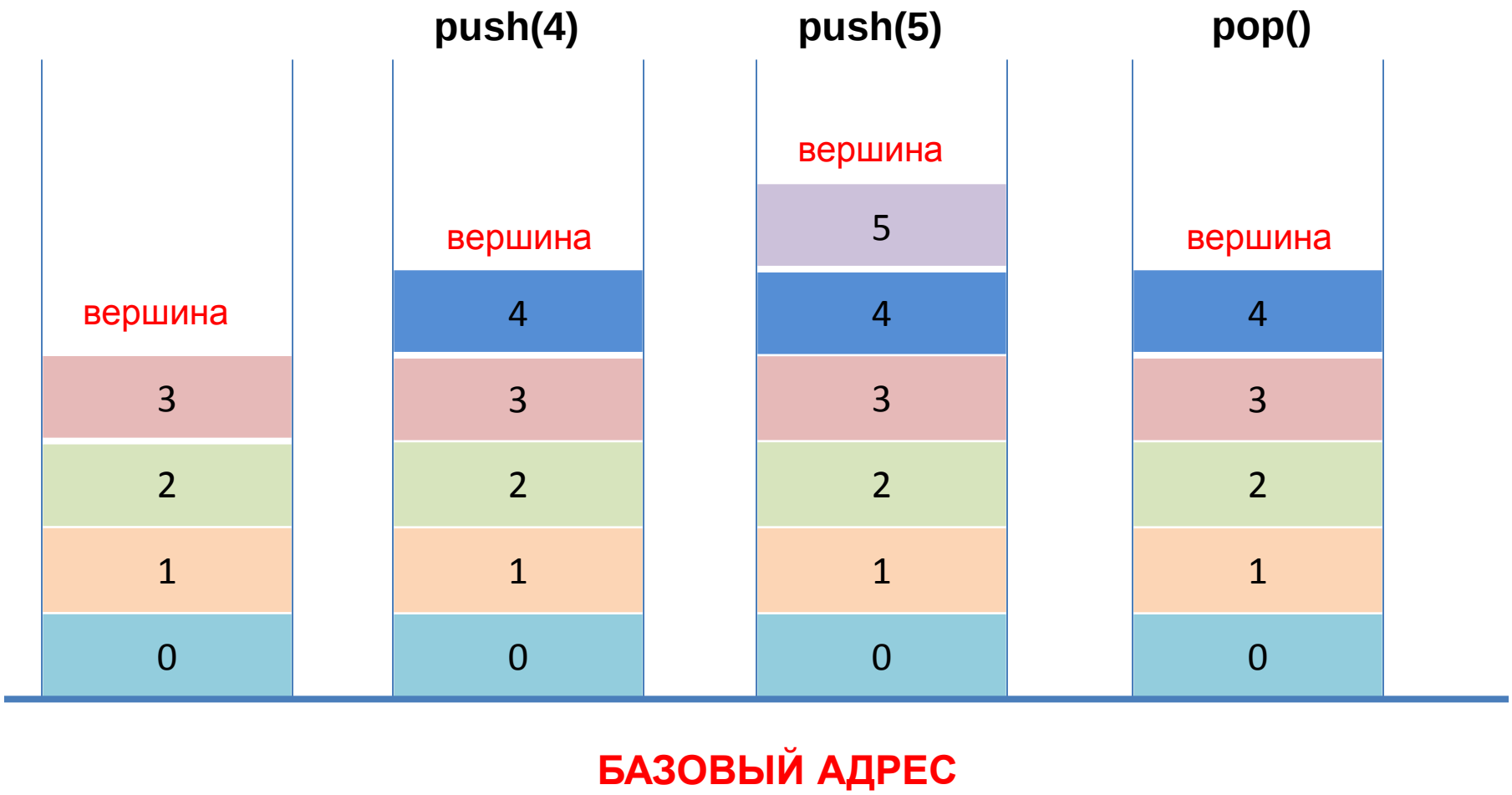
Стек

- стек – это упорядоченный набор элементов, в котором добавление новых и удаление существующих элементов допустимо только с одного конца
- вершиной стека называется тот его конец, на котором добавляются и удаляются элементы
- стек действует по принципу LIFO (последний пришел – первый ушел)
- базовый адрес – начальный адрес памяти, в которой размещается стек

Основные операции

- добавить элемент (push)
- получить значение верхнего элемента и удалить его (pop)
- только получить значение, не удаляя
- узнать количество элементов
- проверить на пустоту

Пример работы



Пример реализации

```
class Stack
{
   int * stackPtr;              // указатель на стек
   const int maxSize;           // максимальный размер стека
   int top;                     // индекс вершины
public:
   ...
   void push(const int&);     // поместить элемент в стек
   void pop();                  // удалить элемент из стека
   int getSize() const;         // получить размер стека
};
```

Пример реализации

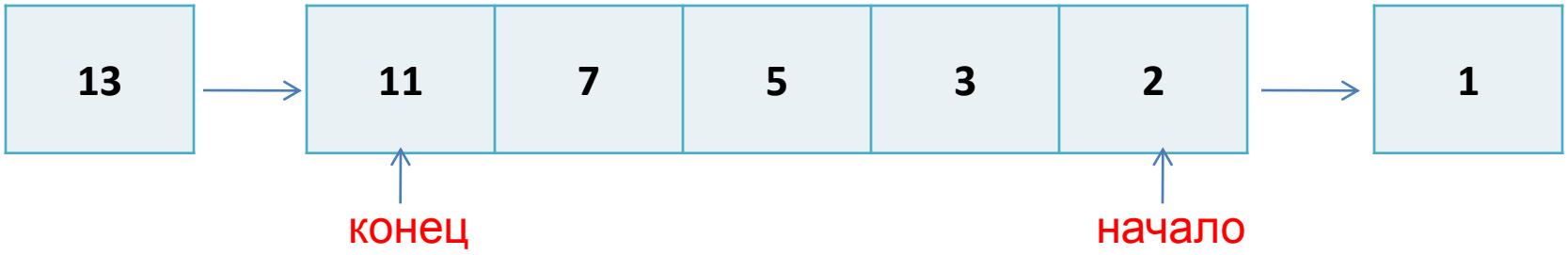
```
void Stack::push(const int& value) {
    if (top < maxSize) {
      stackPtr[top++] = value;
    }
}

void Stack::pop() {
    if (top > 0) {
      stackPtr[--top];
    }
}

int Stack::getSize() const {
    return top+1;
}
```

Очередь (Queue)

- очередь – это последовательный набор элементов, длина которого может меняться
- элементы добавляются в конец очереди, а удаляются – из начала очереди
- очередь действует по принципу FIFO (первый пришел – первый ушел)



Основные операции

- узнать значение первого элемента очереди, не удаляя его (front)
- узнать значение последнего элемента очереди (back)
- добавить элемент в конец очереди (push)
- удалить первый элемент из очереди (pop)
- узнать количество элементов
- проверить на пустоту

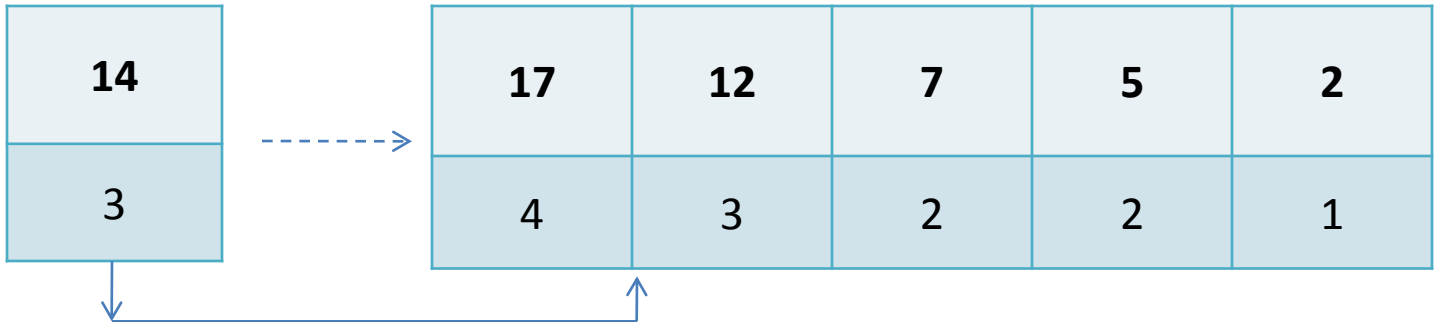
Очередь с приоритетами

- каждому элементу в очереди сопоставляется его приоритет
- порядок выхода элементов из очереди определяется их приоритетом (первым выходит не первый элемент, а элемент с наибольшим приоритетом)

Виды очередей с приоритетами

Очередь с приоритетным включением

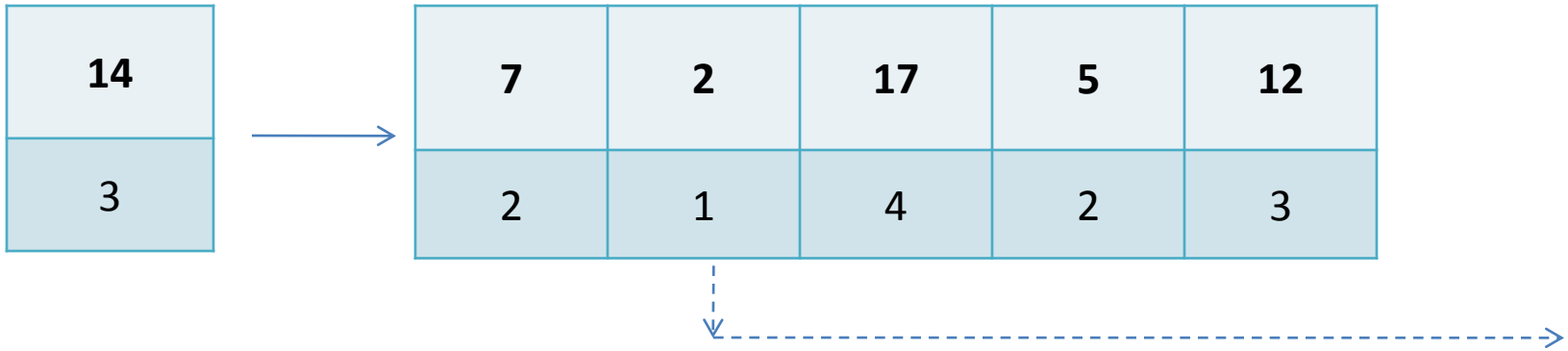
- элемент сразу вставляется на место, соответствующее его приоритету
- при удалении извлекается первый элемент



Виды очередей с приоритетами

Очередь с приоритетным исключением

- элемент вставляется в конец очереди
- при удалении извлекается тот элемент, который имеет наибольший приоритет



Вопросы?