Стек

Стеком называется упорядоченный набор элементов, в котором размещение новых и удаление существующих происходит с одного конца, называемого ***вершиной***.

Дисциплина обслуживания — это совокупность правил (упорядочение и алгоритм) обслуживания элементов динамической структуры данных.

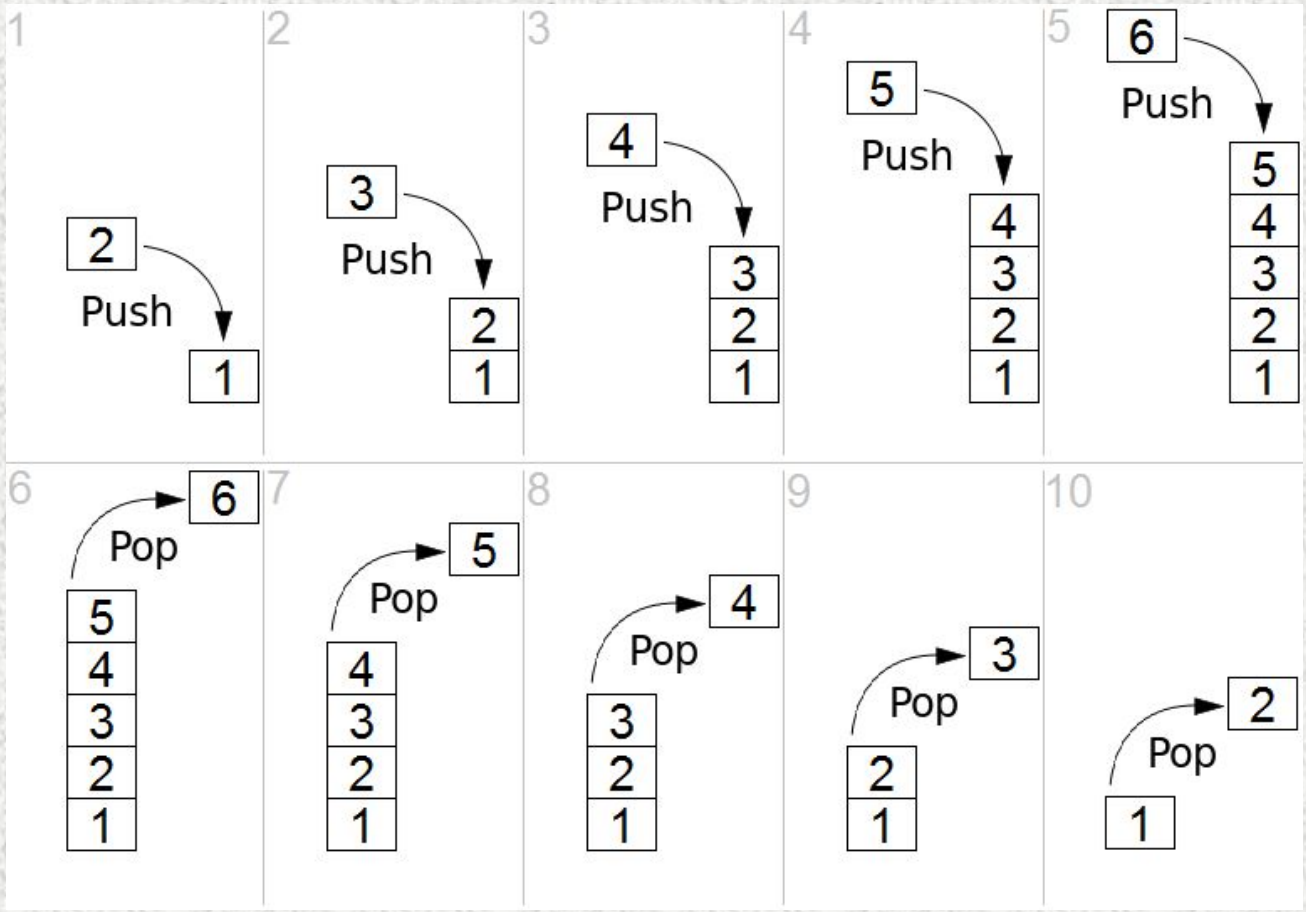
В стеке реализуется дисциплина обслуживания LIFO:

LAST - последний

INPUT - вошел

FIRST - первый

OUTPUT - вышел



Операции для работы со стеком

Над стеком реализованы следующие операции:

- инициализация стека **init**(s), где s — стек
- помещение элемента в стек **push** (s, i), где s — стек, i — помещаемый элемент;
- удаление элемента из стека **i=pop**(s);
- определение верхнего элемента без его удаления **i=stkTop**(s), которая эквивалентна операциям **i=pop** (s); **push** (s, i);
- получение вершины стека (количества элементов) **i=gettop**(s), где s — стек
- печать стека **stkPrint**(s), где s — стек
- определение пустоты стека **isempty**(s)
  возвращает **true** если стек пустой и **false** в противном случае.

Способы реализации стека

Существует несколько способов реализации стека:

- с помощью одномерного массива;
- с помощью связанного списка;
- с помощью класса объектно-ориентированного программирования.

Пример реализации стека

Стек можно реализовать в виде следующей структуры:

```
#define NMAX 100
struct stack {
  float elem[NMAX];
  int top;
};
```

NMAX — максимальное количество элементов в стеке;
elem — массив из NMAX чисел типа float, предназначенный для хранения элементов стека;
top — индекс элемента, находящегося в вершине стека.

Инициализация стека

Индекс элемента, находящегося в вершине стека, равен 0.

```
void init(struct stack *stk) {
  stk->top = 0;
}
```

Помещение элемента в стек (push)

```
void push(struct stack *stk, float f) {
  if(stk->top < NMAX) {
    stk->elem[stk->top] = f;
    stk->top++;
  } else
    printf("Стек полон, количество элементов: %d !\n",
stk->top);
}
```

Удаление элемента из стека (pop)

```
float pop(struct stack *stk) {
  float elem;
  if((stk->top) > 0) {
    stk->top--;
    elem = stk->elem[stk->top];
    return(elem);
  } else {
    printf("Стек пуст!\n");
    return(0);
  }
}
```

Извлечение вершины стека

```
float pop(struct stack *stk) {
  float elem;
  if((stk->top) > 0) {
    stk->top--;
    elem = stk->elem[stk->top];
    return(elem);
  } else {
    printf("Стек пуст!\n");
    return(0);
  }
}
```

Получение верхнего элемента стека без его удаления

```
int gettop(struct stack *stk) {
 return(stk->top);}
```

Определение пустоты стека

```
int isempty(struct stack *stk) {
  if((stk->top) == 0)    return(1);
  else return(0);
}
```

Вывод элементов стека

```
void stkPrint(struct stack *stk) {
  int i;
  i=stk->top;
  if(isempty(stk)==1) return;
  do {
    i--;
    printf("%f\n", stk->elem[i]);
  }while(i>0);
}
```

Вывод элементов стека

```
void stkPrint(struct stack *stk) {
  int i;
  i=stk->top;
  if(isempty(stk)==1) return;
  do {
    i--;
    printf("%f\n", stk->elem[i]);
  }while(i>0);
}
```

Вывод элементов стека

```
void stkPrint(struct stack *stk) {
  int i;
  i=stk->top;
  if(isempty(stk)==1) return;
  do {
    i--;
    printf("%f\n", stk->elem[i]);
  }while(i>0);
}
```