Стек (stack)

1. **Определение.**

Стекът (stack) представлява свързана, линейна поредица от еднотипни данни. Първият добавен елемент се нарича „дъно“ (bottom), а последният – „връх“ (top). Върхът е най-важният елемент от стека, понеже служи като точка за достъпи в него се извършват всички операции свързани със стека. Стек, който няма елементи се нарича „празен“ (empty).

Примери за стек са: пълнителят на автомат, купчина от книги наредена една върху друга, пакет с бисквити и т.н.



Използването на стекове е свързано с редица ограничения по отношение на допустимите операции спрямо тях.

2. **Допустими операции и ограничения:**

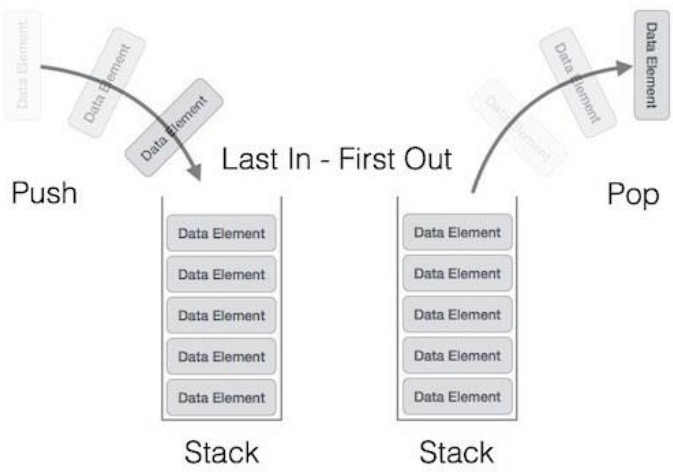
2.1.Добавяне (push): Нов елемент може да се добави само пред върха на стека. След като се добави, новият елемент, се обявява за връх. Елементите се добавят един по един.

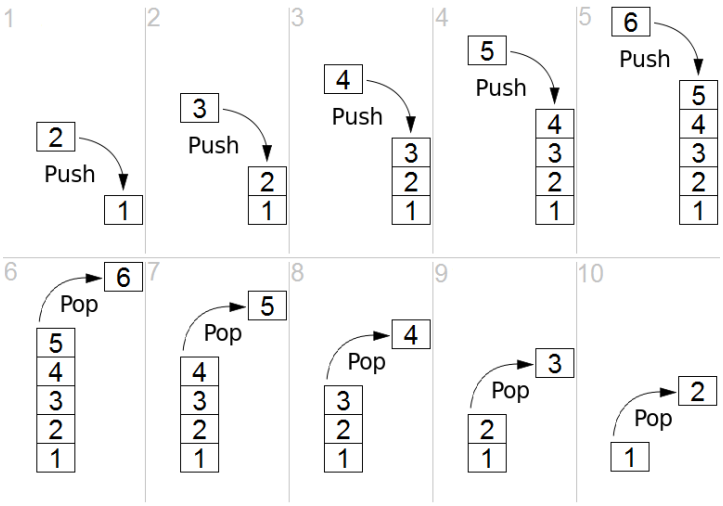
2.2.Премахване (pop): Когато се налага изваждане на елемент от стек се премахва неговият връх. Следващият елемент, който е бил след върха се обявява за връх. Елементите се премахват един по един.

2.3.Проверка дали стекът е празен.

3. **Особености при стековете.**

Поради факта, че добавянето и премахването стават на едно и също място (във върха на стека) последният добавен елемент пръв напуска стека. Когато това правило последователно се приложи спрямо всички елементи се получава, че те напускат стека в обратен ред на този, който са постъпили. Затова стековете се наричат също и LIFO (Last In First Out) списъци.





Употреба.

Стекове се употребяват при обработката на някой команди от процесора. Освен в хардуера стековете се използват и в софтуера – например при реализацията на рекурсията от страна на компилатора. Стекове се използват и при математически изчисления на по-сложни изрази. Някой калкулатори използват т.нар. „обратен полски запис“ за по-опростено представяне на математически изрази.

4. Програмна реализация.

Съществуват различни начини за създаване и използване на стекове в програмите:

4.1. Най-лесният е да се използва наготово библиотеката „stack“. Тя предлага шаблонен клас с няколко полезни метода, както за трите основни операции, така и няколко допълнителни като:

- Четене на стойността във върха, без той да се унищожава: top;
- Определяне размера на стека: size;

4.2. Статична реализация – с едномерен масив.

4.3. Най-сложна е динамичната реализация чрез използване на указатели. Тя, обаче, е най-ефективна, понеже не се заделя излишна памет.

5. Примерна реализация чрез библиотеката „stack“:

Пример 1:

```
#include<iostream>
#include<stack>
using namespace std;
int main()
{
   stack<int> st;
   for (int i=1; i<10; i++) st.push(i);
   while (st.empty() == false)
   {
      cout << st.top() << " ";
      st.pop();
   }
   return 0;
}
```

Пример 2:

```
#include <iostream>
#include <stack>
using namespace std;

int main()
{
	stack<int> s;
	cout<<"Broi elementi:"<<s.size()<<endl;
	s.push(1);
	s.push(2);
	s.push(3);
	cout<<"Broi elementi:"<<s.size()<<endl;
cout<<"Gorniq element:"<<s.top()<<endl;
	s.pop();
cout<<"Broi elementi:"<<s.size()<<endl;
cout<<"Gorniq element:"<<s.top()<<endl;
	s.pop();
cout<<"Broi elementi:"<<s.size()<<endl;
cout<<"Gorniq element:"<<s.top()<<endl;
stack<char> a;
	a.push('a');
	a.push('b');
	a.push('c');
	a.push('d');
	a.push('e');
	a.push('f');
	a.push('g');
cout<<"Broi elementi:"<<a.size()<<endl;
while(!a.empty())
	{
	cout<<"Gorniq element:"<<a.top()<<endl;
	a.pop();
	}
cout<<"Broi elementi:"<<a.size()<<endl;
	return 0;
}
```