Stack

Какво е Stack? Stack (стек) е структура от данни, която ни позволява да добавяме и премахваме елементи. Характерно е, че елементите се премахват в обратен ред, спрямо реда им на добавяне. Във всеки момент можем да проверим, кой е най-новият (най-горният) елемент на стека, както и дали изобщо има елементи.

Когато една програма работи, финкциите, които се извикват в нея, се съхраняват в стек. Това позолява лесно да излизаме от функциите в ред, обратен на реда на извикването им. Това става автоматично. Рекурсивните функции работят благодарение на това.

Накратко:

- **bool** `push(`**int** `element)` - Добавя нов елемент на върха на стека;
- **bool** `pop()` - Премахва най-горният елемент на стека. Ако няма такъв - връща false;
- **int** `peek()` **const** - Връща стойността на елемента, намиращ се на върха на стека;
- **bool** `isEmpty()` **const** - Казва, дали стекът е празен.

Push и pop връщат истина или лъжа, в зависимост от това, дали операцията е успешна.

Нашият стек ще работи с елементи от тип **int**.

Задача 1:

Реализирайте класа, като използвате клас Vector (от предното упражнение).

Указание: Съхранявайте елементите в **private** инстанция на класа Vector и използвайте неговите методи, за да реализирате четирите метода на стека. Преценете какви конструктури и деструктури ще са ви необходими. Помислете и за присвояващ оператор (operator =).

Задача 2:

Реализирайте класа, като свързан списък.

Какво е свързан списък? Свързан списък е структура от данни, при която отделните елементи се съхраняват в клетки, които са разположени **НЕпоследователно** в паметта. Всяка клетка има указател към следващата. За да достигнем дадена клетка, трябва да преминем по цялата верига от клетки, започвайки от началото.

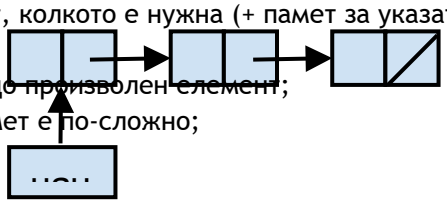
Предимства:

- Не се налага да заделяме последователна памет за съхранение на данните;
- Използва се толкова памет, колкото е нужна (+ памет за указателите - тя също е нужна);

Недостатъци:

- Нямаме директен достъп до произволен елемент;

Управлението на динамичната памет е по-сложно;



-

Указание: Създайте клас стек и като единствена член-данна пазете адреса на най-горната клетка. Самата клетка реализирайте като клас/структура с две полета:

- данните, които съдържа;
- указател към следващата клетка;

Например:

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

Последната клетка от веригата има указател, равен на NULL

Push:

1. Създавате нова клетка (динамично); Давате данните, подадени като параметър на функцията и като следващ елемент, записвате стойността на първия елемент от стека;
   push(10):
   [10] -> [5] -> ...
   node top

2. Записвате адреса на новата клетка в член-данната на класа Stack;
   [10] -> [5] -> ...
   top
   node
3. Връщате true.

Pop:

1. Ако не съдържа елементи (т.е. top == NULL или isEmpty() == true), връщате false;
2. Записвате адреса на най-горния елемент в нова променлива;
   [10] -> [5] -> ...
   top
   node
3. Като най-горен елемент, записвате следващата клетка (top->next);
   [10] -> [5] -> ...
   node top
4. Изтривате старата клетка;
   [5] -> ...
   ~~**node**~~ **top**
5. Връщате true.

Peek:

1. Връщате данните на най-горната клетка на стека.

IsEmpty:

1. Връща true, ако top == NULL, иначе - false.

Помислете какви конструктори, деструктор и присвояващ оператор, ще трябва да напишете.