Задание #2

Структуры данных: стек, очередь.

Рассмотрено на лекции:

АТД стек – Stack – абстрактный класс

СД стек «ограниченный» - StackArray – реализуется через массив

СД стек «неограниченный» - StackList – реализуется через список (односвязный)

1. Реализуйте структуру данных «ограниченный» стек

1. Используйте шаблон для абстрактного типа данных класса стек:

```
template <class T>
class Stack
{
public:
    virtual ~Stack() {}
    virtual void push(const T& e) = 0; // Добавление элемента в стек
    virtual T    pop() = 0;            // Удаление верхнего элемента
    virtual bool isEmpty() = 0;        // Проверка стека на пустоту
};
```

2. На основе шаблона для стека создайте шаблон для реализации структуры данных «ограниченный» стек (через массив).
3. Создайте классы StackOverflow и StackUnderflow для работы с двумя исключительными ситуациями, которые могут возникнуть при работе со стеком.
4. Создайте класс WrongStackSize для работы с исключительной ситуацией, которая может возникнуть, если в конструкторе стека, реализуемого через массив, неправильно задан размер.
5. Выполните тестирование созданного класса.

2. Реализуйте функцию анализа правильности расстановки скобок

Функция должна возвращать True, если количество открывающих и закрывающих скобок одного типа совпадает, и они имеют правильную вложенность. Допускаются три вида скобок: круглые, квадратные и фигурные.
Прототип функции:

```
bool checkBalanceBrackets (const char* text, const int maxDeep);
```

или

```
bool checkBalanceBrackets (const string& text, const int maxDeep);
```

`text` - анализируемый текст, содержащий скобки
`maxDeep` - максимально возможный уровень вложенности скобок

В реализации используйте шаблон «ограниченный стек».

3. Реализуйте структуру данных «ограниченная» очередь

1. Используйте шаблон для абстрактного типа данных класса очередь:

```
template <class T>
class Queue
{
public:
    virtual ~Queue () {}
    virtual void enQueue(const T& e) = 0; // Добавление элемента в очередь
    virtual T    deQueue() = 0;           // Удаление элемента из очереди
    virtual bool isEmpty() = 0;           // Проверка очереди на пустоту
};
```

2. На основе шаблона для очереди создайте шаблон для реализации структуры данных «ограниченная» ***кольцевая*** очередь (через массив).

3. Создайте классы QueueOverflow и QueueUnderflow для работы с двумя исключительными ситуациями, которые могут возникнуть при работе с очередью.

4. Создайте класс WrongQueueSize для работы с исключительной ситуацией, которая может возникнуть, если в конструкторе очереди, реализуемой через массив, неправильно задан размер.

5. Выполните тестирование созданного класса