Практика #5.

АТД: очередь.

Итеративные алгоритмы обхода двоичных деревьев.

## 1. Реализуйте структуру данных «кольцевая» 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
   virtual T  deQueue () = 0;                  // Удаление и возвращение элемента
   virtual bool isEmpty() = 0;                 // Проверка на пустоту
};
```

2. На основе шаблона для очереди создайте шаблон для реализации структуры данных «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 Wrong Queue Size для работы с исключительной ситуацией, которая может возникнуть, если в конструкторе очереди, реализуемой через кольцевой массив, неправильно задан размер.

## 2. В шаблон класса BinarySearchTree добавьте методы:

1. Итеративный метод обхода двоичного дерева по уровням (в ширину). В реализации использовать класс очередь.

2. Метод, определяющий, являются ли два дерева похожими. Похожими будем называть деревья поиска, содержащие одинаковые наборы ключей. Рекомендация: параллельно обходить инфиксным обходом сравниваемые деревья.

3. Метод, определяющий есть одинаковые ключи в двух деревьях поиска. Рекомендация: параллельно обходить инфиксным обходом сравниваемые деревья.