

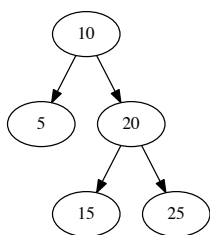
ЗАДАЧИ ЗА ЗАДЪЛЖИТЕЛНА  
САМОПОДГОТОВКА  
ПО  
Структури от данни и програмиране  
*Двоични дървета 2*

*email: kalin@fmi.uni-sofia.bg*

27 октомври 2016 г.

1. Да се реализира метод `vector<T> BTree<T>::listLeaves ()` намиращ списък със стойностите на листата на дървото.
2. Да се дефинира метод `string BTree<T>::findTrace (const T& x)`. Ако `x` е елемент на дървото, функцията да връща следата на `x` (според дефиницията на “следа”, обсъдена на лекции). Ако `x` не е елемент на дървото, функцията да връща низа “\_”.

*Пример: За дървото от фигура 1, следата на елемента със стойност 25 е “RR”.*



Фигура 1. Двоично наредено дърво

3. Да се дефинира метод `void BTree<T>::prettyPrint ()`, отпечатащ дървото на конзолата по следния начин: (1) всеки наследник е

вдясно от родителя си, (2) елементите на еднакво ниво в дървото се отпечатват на еднаква колона от екрана, (3) десните наследници са на предишен ред от родителя си и (4) левите наследници са следващ ред спрямо родителя си.

Например, дървото от Фигура 1 би изглеждало по следния начин (включени са номерата на редовете на конзолата):

```
1:      25
2:    20
3:      15
4: 15
5:    5
```

4. Да се дефинират методи за сериализация и де-сериализация на двоично дърво, като се използва “Scheme формат”. Например, дървото от фигура 1 се представя по следния начин:

```
(15 (20 (25 () ())) (15 () ())) (5 () ()))
```

5. Да се дефинира оператор `T& BTree<T>::operator[] (int i)`, който намира  $i$ -тият пореден елемент на дървото при обхождане корен-ляво-дясно.

Пример: За дървото от фигура 1, елементът с пореден номер 0 е 15, с номер 1 е 5, с номер 2 е 20 и т.н.

6. Да се дефинира метод `vector<T> BTree<T>::level (int k)`, който намира и връща вектор, съдържащ стойностите на всички елементи на дървото, които са на ниво  $k$  (т.е. има път от корена до тях с дължина в брой върхове  $k$ ).