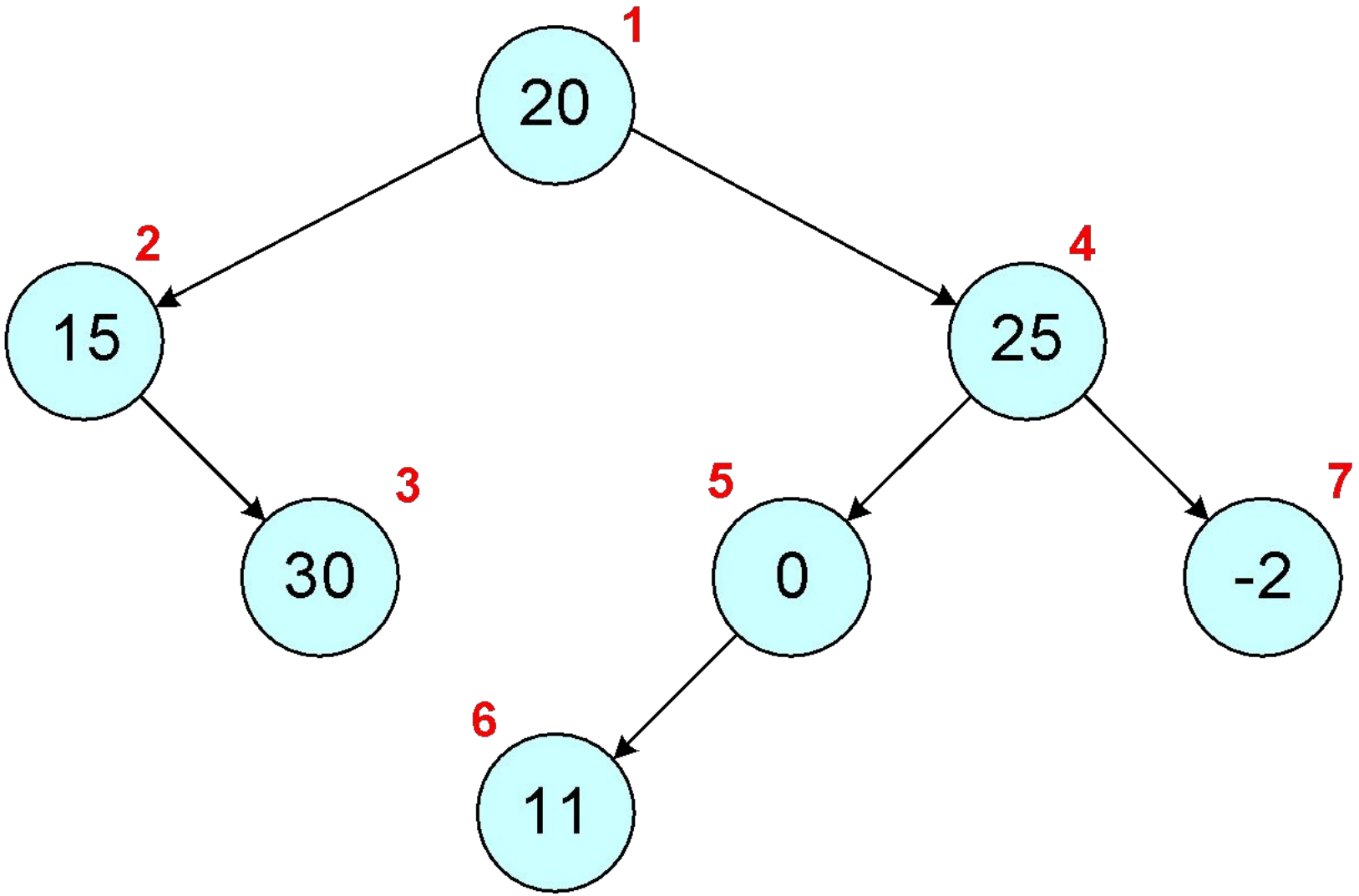
Практическое занятие 3

Бинарные деревья

Структура данных

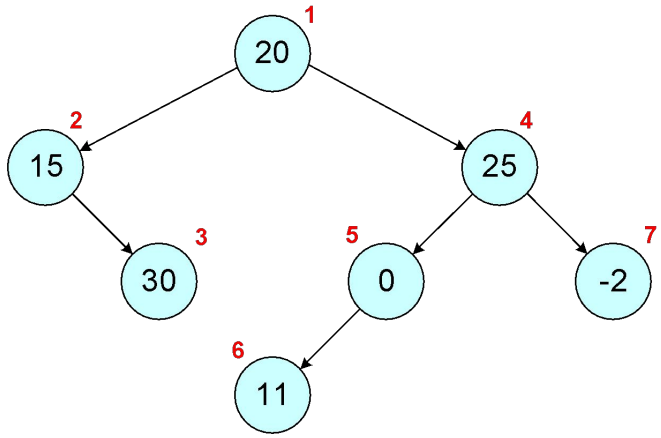
```
#include <iostream>
using namespace std;

typedef struct Node {  // Узел дерева
    int data;             // Или другой тип данных
    Node *left, *prev;    // Сыновья
} Node;

typedef Node* Tree;   // Указатель на дерево
```

Пример бинарного дерева



Бинарное дерево на входе - 1

- Рекурсивное описание узлов:
  - после каждого узла описан его левый сын (рекурсивно), затем правый сын рекурсивно);
  - пустые сыновья не описываются.



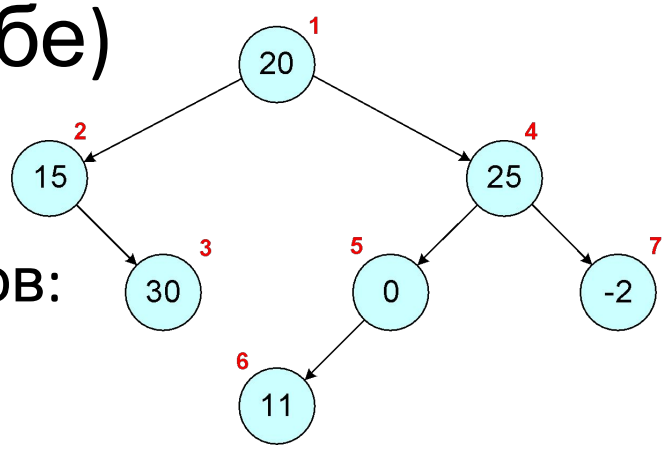

```
7
20  1  1
15  0  1
30  0  0
25  1  1
 0  1  0
11  0  0
-2  0  0
```

- Количество узлов (n).
- n строк описаний узлов:
  - значение;
  - есть левый сын? (1/0) ;
  - есть правый сын? (1/0).

Бинарное дерево на входе - 2

- Иерархия узлов (как в лабе)

```
7
0 20
00 15
001 30
01 25
010 0
0100 11
011 -2
```



- Количество узлов (n).
- n строк описаний узлов:
  - уровень;
  - значение.
- Корню дерева присваиваем уровень “0”.
- Корню дерева присваиваем уровень “0”.
- Левый сын отличается от отца добавлением “0” в уровень.
- Правый сын – добавлением “1”.

Ввод/вывод бинарного дерева (1)

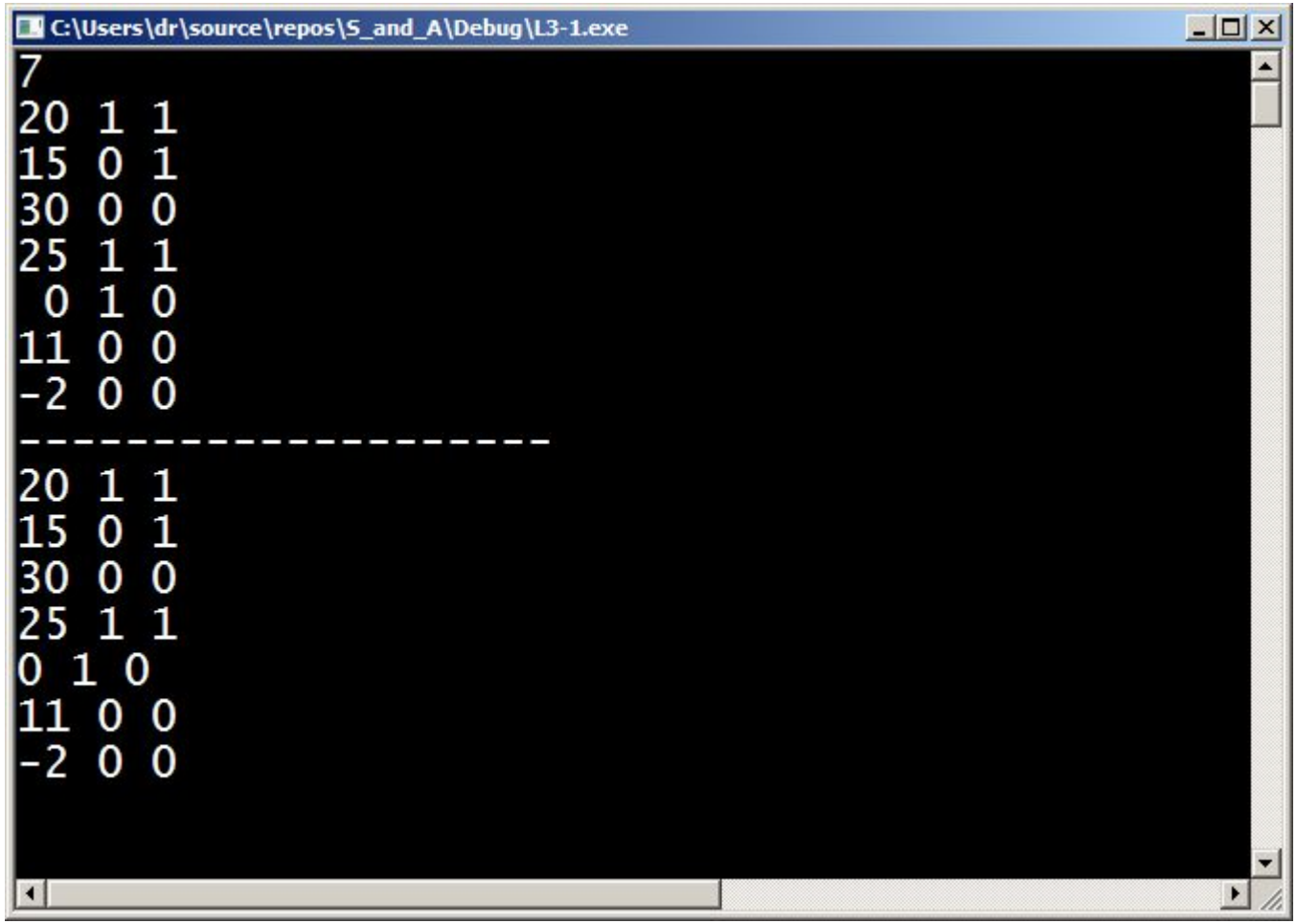
```
Tree InTree() { // Ввод НЕПУСТОГО дерева
    int l, r;
    Tree t = new(Node);
    cin >> t->data >> l >> r;
    t->left = l ? InTree() : NULL;
    t->right = r ? InTree() : NULL;
    return t;
}

void OutTree(Tree t) { // Вывод НЕПУСТОГО дерева
    Tree l = t->left, r = t->right;
    cout << t->data << " " << (l ? 1 : 0) << " "
         << (r ? 1 : 0) << endl;
    if (l)
        OutTree(l);
    if (r)
        OutTree(r);
}
```

Ввод/вывод бинарного дерева (2)

```
int main() {
    int n;
    Tree tree = NULL;
    cin >> n;
    if (n)
        tree = InTree();
    cout << "--------------------" << endl;

    // Здесь можно вставить обработку дерева

    OutTree(tree);
    return 0;
}
```

- Поскольку описание дерева рекурсивно, функции ввода/вывода тоже легче всего написать рекурсивно.
- Заметим, что значение **n** фактически нигде не используется, кроме как для определения «ноль / не ноль».

Результат работы



Задача 1

- Ввести дерево. Выдать значения всех положительных узлов в порядке их расположения слева направо в дереве.
- **Входные данные**: описание непустого дерева.
- **Выходные данные**: список положительных значений через пробел.

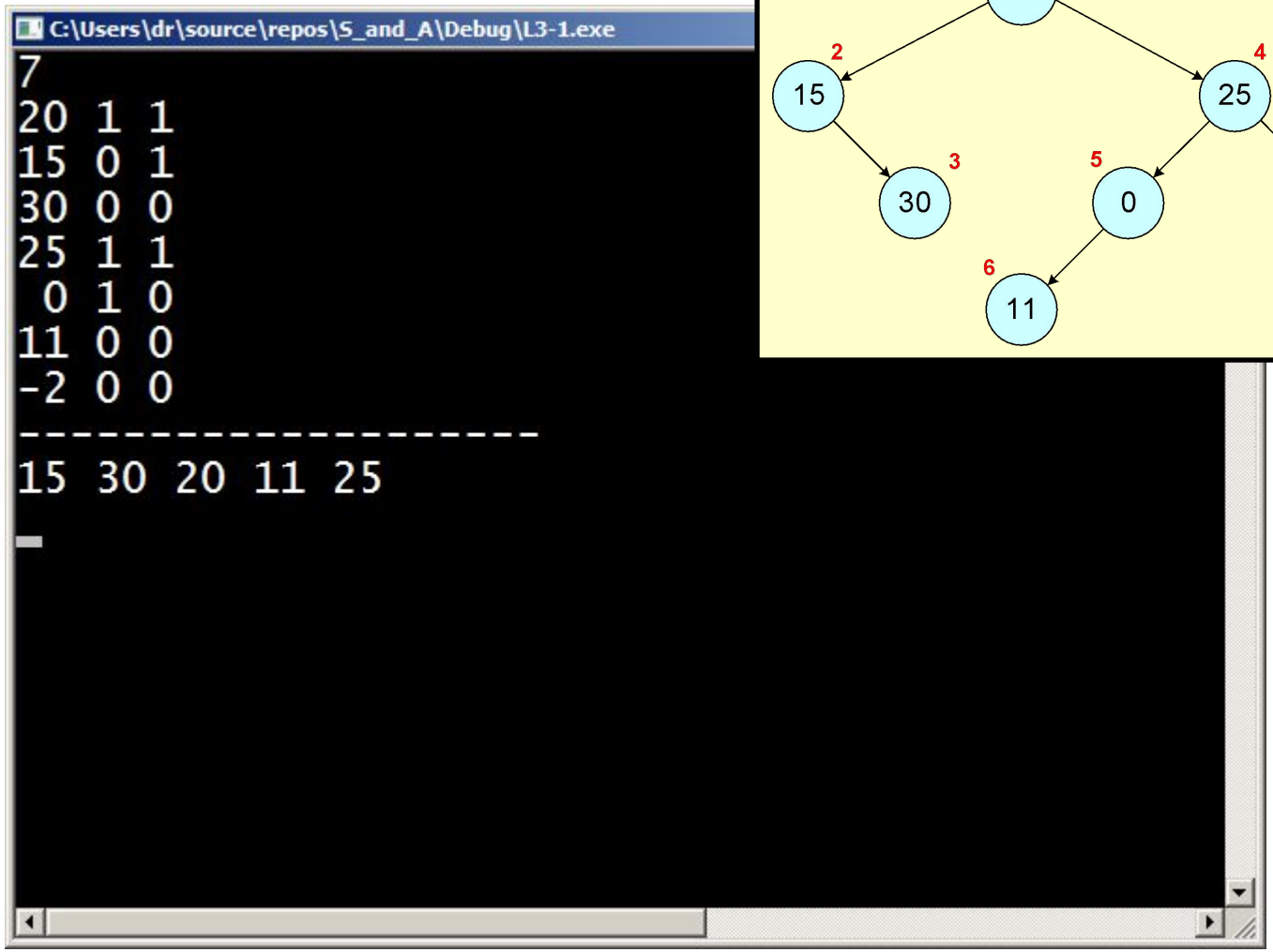
Решение задачи 1 *(лист 1)*

```
void ListPositive(Tree t) {
    if (!t)
        return;
    if (t->left)
        ListPositive(t->left);
    if (t->data > 0)
        cout << t->data << " ";
    if (t->right)
        ListPositive(t->right);
    return;
}
```

Решение задачи 1 *(лист 2)*

```
int main() {
    int n;
    Tree tree = NULL;
    cin >> n;
    tree = InTree();
    cout << "----------------------" << endl;
    ListPositive(tree);
    cout << endl;
    return 0;
}
```

Результат работы



Задача 2

- Ввести дерево. Построить новое дерево, значения узлов которого равны удвоенным значениям узлов исходного дерева.
- **Входные данные**: описание непустого дерева.
- **Выходные данные**: описание дерева-результата.

Решение задачи 2 *(лист 1)*

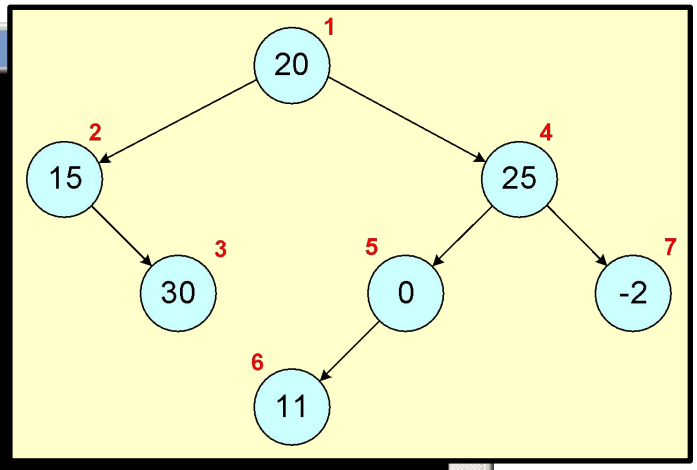
```
Tree DoubleTree(Tree t1) {
    if (!t1)
        return NULL;
    Tree t2 = new(Node);
    t2->data = t1->data * 2;
    t2->left = DoubleTree(t1->left);
    t2->right = DoubleTree(t1->right);
    return t2;
}
```

Решение задачи 2 *(лист 2)*

```
int main() {
    int n;
    Tree tree1 = NULL, tree2;
    cin >> n;
    tree1 = InTree();
    cout << "----------------------" << endl;
    tree2 = DoubleTree(tree1);
    OutTree(tree2);
    return 0;
}
```

Результат работы

```
C:\Users\dr\source\repos\S_and_A\Debug\L3-3.exe
7
20 1 1
15 0 1
30 0 0
25 1 1
 0 1 0
11 0 0
-2 0 0
--------------------
40 1 1
30 0 1
60 0 0
50 1 1
0 1 0
22 0 0
-4 0 0
```



Задача 3

- Ввести дерево. Удалить все узлы дерева, кроме самой правой ветви.
- **Входные данные**: описание непустого дерева.
- **Выходные данные**: описание дерева-результата.

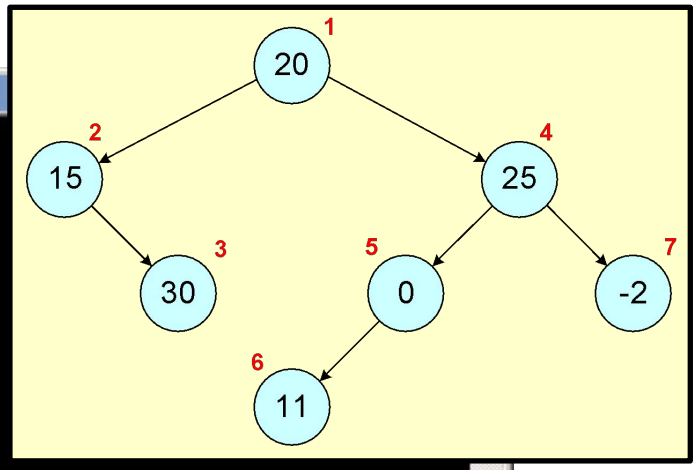
Решение задачи 3 *(лист 1)*

```
void DelTree(Tree t) {
    if (!t)
        return;
    DelTree(t->left);
    DelTree(t->right);
    delete(t);
    return;
}
void OnlyRight(Tree t) {
    while (t) {
        DelTree(t->left);
        t->left = NULL;
        t = t->right;
    }
    return;
}
```

Решение задачи 3 *(лист 2)*

```
int main() {
    int n;
    Tree tree = NULL;
    cin >> n;
    if (n)
        tree = InTree();
    cout << "----------------------" << endl;
    OnlyRight(tree);
    OutTree(tree);
    return 0;
}
```

Результат работы




```
C:\Users\dr\source\repos\S_and_A\Debug\L3-2.exe
7
20 1 1
15 0 1
30 0 0
25 1 1
 0 1 0
11 0 0
-2 0 0
--------------------
20 0 1
25 0 1
-2 0 0
```