Содержание

1 Текст задания

Задача 10

Дано бинарное дерево. Каждую вершину с чётным номером поменять местами с сыном, имеющим чётный номер.

Ввод входных параметров осуществляется с помощью клавиатуры и мыши.

2 Исходные тексты программы

index.html

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <title>Лабораторная работа №3 - Смена четных вершин в бинарном дереве</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" type="text/css" href="lib/treant/treant.css">
        <link rel="stylesheet" type="text/css" href="css/style.css">
        <script type="text/javascript" src="lib/jquery/jquery-2.2.4.min.js"></script>
        <script src="lib/raphael/raphael.js"></script>
        <script src="lib/treant/treant.js"></script>
        <script type="text/javascript" src="js/binaryTree.js"></script>
    </head>
    <body>
        <div class="wrapper">
            <div id="condition">
                <h2>Условие задачи</h2>
                <p>Дано бинарное дерево. Каждую вершину с чётным номером поменять местами с сыном, имеющим чётный номер.</p>
            </div>
            <div id="input">
                <h2>Входные параметры</h2>
                <p><label>Число связей: <input id="countNodes" type="number" placeholder="Число связей в бинарном дереве"></label></p>
                <div id="binaryTree"></div>
            </div>
            <div id="output">
                <h2>Выходные параметры</h2>
                <p id="tree_input">Заполните входные параметры</p>
                <div class="chart" id="chart_input"></div>
```

```
                    <div id="steps"></div>
                </div>
            </div>
        </body>
    </html>
```

style.css

```
    @font-face
    {
      font-family:Roboto;
      src:url(fonts/roboto/Roboto.woff2) format(woff2),
url(fonts/roboto/Roboto.woff) format(woff), url(fonts/roboto/Roboto.ttf)
format(truetype);
      font-weight:400;
      font-style:normal;
    }

    *
    {
      margin:0;
      padding:0;
    }

    html
    {
      font-size:62.5%;
    }

    body
    {
      font-family:Roboto;
    }

    .wrapper
    {
      max-width:800px;
      width:calc(100%-20px);
```

```
  border-left:1px solid #39c;
  border-right:1px solid #39c;
  margin:0 auto;
}

h2
{
  font-size:2em;
  color:#fff;
  text-align:center;
  padding:5px 8px;
  background:#39c;
}

p
{
  font-size:1.6em;
  margin:5px;
}

input
{
  font-size:1em;
}

label
{
  display:block;
  width:100%;
  text-align:center;
}

label.connection
{
  text-align:left;
}

label > input
```

```
{
  width:100%;
  box-sizing:border-box;
  text-align:center;
  outline:0;
  padding:4px 2px;
}

label > input.node
{
  width:20%;
  margin:5px;
}

#condition p
{
  text-align:justify;
}

#det
{
  display:flex;
  flex-wrap:wrap;
  text-align:center;
  margin:16px 6px;
}

.cell
{
  display:inline-block;
  font-size:1.6em;
  text-align:center;
  border:1px solid #39c;
  box-sizing:border-box;
}

.chart
{
```

```
  width:100%;
  overflow-x:auto;
}

.node-name
{
  font-weight:700;
}

.nodeStyling
{
  border-radius:3px;
  border:1px solid #000;
  min-width:50px;
  text-align:center;
  padding:2px;
}

.gray
{
  background:#909090;
}

.light-gray
{
  background:#D3D3C7;
}

.blue
{
  background:#A2BDFD;
}

#output
{
  border-bottom:1px solid #39c;
}
```

```
#output p
{
  margin-top:10px;
  text-align:center;
}
```

binaryTree.js

```
/**
 * Получить количество связей между узлами
 */
function getCoutNodes()
{
    return jQuery('#countNodes').val();
}

/**
 * Проверка значения на int
 * @param {int} val
 *   Значение для проверки
 */
function isInt(val)
{
    var result = false;
    if (Math.floor(val).toString() === val && jQuery.isNumeric(val))
    {
        result = true;
    }
    return result;
}

/**
 * Является ли число положительным
 * @param {int} val
 *   Значение для проверки
 */
function numPositive(val)
{
```

```
    var result = false;
    if (val >= 0)
    {
        result = true;
    }
    return result;
}

/**
 * Реализует хранение бинарного дерева
 *
 * @param {string} name
 *   Имя корневого узла
 * @param {string} data
 *   Значение корневого узла
 * @returns {object}
 *   Бинарное дерево
 */
function BinaryTree(name, data)
{
    var self = this; // Текущий узел
    var root; // Корневой узел
    var names = []; // Уникальные имена узлов
    var chart = []; // Параметры отрисовки бинарного дерева
    var container = '#answer'; // Имя контейнера для отрисовки дерева
    var configChart = {
        // Контейнер для отрисовки дерева
        container: container,
        // Выравнивание связей для узлов
        nodeAlign: 'bottom',
        // Внешний вид линии связи между узлами
        connectors:
        {
            type: 'step'
        },
        // Стилизация узла
        node:
        {
```

```
        HTMLclass: 'nodeStyling'
    }
};
if (name === undefined || data === undefined)
{
    throw new Error('Не задан корневой узел или его значение');
}
else
{
    var root = new BinaryTreeNode(name, data, null);
    names.push(name);
    chart.push(configChart); // Конфигурация
    chart.push(root.chart); // Отрисовка родительского узла
}

/**
 * Реализует хранение в бинарного дерева
 *
 * @param {string} child
 *   Имя дочернего узла
 * @param {type} data
 *   Данные в узле
 * @param {type} chartParent
 *   Параметры отрисовки родительского узла
 * @returns {BinaryTree.BinaryTreeNode}
 *   Бинарный узел
 */
function BinaryTreeNode(child, data, chartParent)
{
    var nameNode, dataNode, leftNode, rightNode;
    nameNode = child;
    dataNode = data;
    this.name = nameNode; // Имя узла
    this.data = dataNode; // Данные в узле
    this.left = leftNode; // Левый дочерний узел
    this.right = rightNode; // Правый дочерний узел
    // Параметры отрисовки узла
    this.chart = {
```

```
                HTMLclass: 'light-gray',
                text:
                {
                    name: this.name,
                    title: this.data
                }
            };
            if (chartParent !== null)
            {
                this.chart.parent = chartParent;
            }
        }

        /**
         * Реализует хранение узла в бинарном дереве
         *
         * @param {string} parent
         *   Имя родительского узла
         * @param {string} child
         *   Имя дочернего узла
         * @param {type} data
         *   Данные в узле
         */
        self.add = function(parent, child, data)
        {
            if (names.indexOf(child) === -1)
            {
                var node = self.search(parent); // Текущий узел
                var chartParent = node.chart; // Парметры отрисовки текущего
узла
                var createdNode = {}; // Создаваемый узел
                if (node !== null)
                {
                    if (node.left === undefined)
                    {
                        node.left   =   new   BinaryTreeNode(child,   data,
chartParent);
                        createdNode = node.left;
```

```
                    }
                    else if (node.right === undefined)
                    {
                        node.right = new BinaryTreeNode(child, data,
chartParent);
                        createdNode = node.right;
                    }
                    else
                    {
                        throw new Error('Невозможно добавить более двух
дочерних узлов');
                    }
                }
                names.push(child);
                // Отрисовка дочерних узлов
                if (chart.indexOf(createdNode.chart) === -1)
                {
                    chart.push(createdNode.chart);
                }
            }
            else
            {
                // В бинарном дереве доступны только два узла
                throw new Error('Добавляемый узел должен быть уникальным');
            }
        };

        /**
         * Реализует поиск узла по уникальному имени
         *
         * @param {string} name
         *   Имя узла, который надо найти
         * @param {string} node
         *   Текущий узел
         * @returns {BinaryTreeNode}
         *   Найденный бинарный узел
         */
        self.search = function(name, node = root)
```

```
        {
            var node, result = null,
                childLeft, childRight;
            if (node && node.name === name)
            {
                result = node;
            }
            else if (node.left !== undefined || node.right !== undefined)
            {
                if (node.left !== undefined && node.left instanceof
BinaryTreeNode)
                {
                    childLeft = self.search(name, node.left);
                    if (childLeft !== null)
                    {
                        result = childLeft;
                    }
                }
                if (node.right !== undefined && node.right instanceof
BinaryTreeNode)
                {
                    childRight = self.search(name, node.right);
                    if (childRight !== null)
                    {
                        result = childRight;
                    }
                }
            }
            return result;
        };

        /**
         * Получение корневого узла
         *
         * @returns {node}
         *   Корневой узел
         */
        self.root = function()
```

```
        {
            return root;
        };

        /**
         * Смена мест вершины с чётным номером с сыном, имеющим чётный номер
         */
        self.swapEvenNodes = function()
        {
            var i = 1; // Счетчик

            // Проход по все узлам
            names.forEach(function(nodeName)
            {
                // Индекс текущего узла в массиве отрисовки
                var indexNode;
                // Индекс дочернего узла в массиве отрисовки
                var indexChildNode;
                // Текущий ход действий
                var messageID = 'message_' + i;
                // Текущий селектор шага
                var stepID = 'step_' + i;
                // Родительский селектор
                var steps = '#steps';
                // Созданный селектор шага
                var step = '<div id="' + stepID + '"></div>';
                // Временное хранилище
                var tempData = '';
                // Текущий узел
                var node = self.search(nodeName);
                // Получение индекса узла отрисовки
                indexNode = chart.indexOf(node.chart);
                // Отмечаем пройденный узел
                chart[indexNode].HTMLclass = 'blue';
                // Вспомогательные сообщения
                var text = 'Шаг №' + i + '. Проверяем узел ' + node.name +
'(' + node.data + '). ';
                // Значение в текущем узле четное
```

```
                if (node.data % 2 === 0)
                {
                    if ((node.left !== undefined || node.right !== undefined))
                    {
                        tempData = node.data;
                        if (node.left !== undefined && node.left.data % 2 === 0)
                        {
                            node.data = node.left.data;
                            node.left.data = tempData;
                            indexChildNode = chart.indexOf(node.left.chart);
                            chart[indexChildNode].text.title = node.left.data;
                            text += 'Меняем вершины текущего узла и дочернего узла ' + node.left.name + '(' + node.data + '). ';
                        }
                        else if (node.right !== undefined && node.right.data % 2 === 0)
                        {
                            node.data = node.right.data;
                            node.right.data = tempData;
                            indexChildNode = chart.indexOf(node.right.chart);
                            chart[indexChildNode].text.title = node.right.data;
                            text += 'Меняем вершины текущего узла и дочернего узла ' + node.right.name + '(' + node.data + '). ';
                        }
                        chart[indexNode].text.title = node.data;
                    }
                }
                // Созданный селектор информационного сообщения
                var message = '<div id="' + messageID + '"><p>' + text + '</p></div>';
                if (jQuery(steps + ' #' + messageID).length === 0 || jQuery(steps + ' #' + stepID).length === 0)
```

```
            {

                jQuery(steps).append(message);
                jQuery(steps).append(step);
                new Treant(self.chart('#' + stepID));
            }
            else
            {
                jQuery(steps + ' #' + messageID).html(message);
                jQuery(steps + ' #' + stepID).html(step);
                new Treant(self.chart('#' + stepID));
            }

            i += 1;
        });
    };

    /**
     * Получить параметры отрисовки бинарного дерева
     * @param {string} selector
     *   Контейнер для отрисовки дерева
     * @returns {array}
     *   Массив объектов для отрисовки бинарного дерева
     */
    self.chart = function(selector = null)
    {
        if (selector !== null)
        {
            chart['0'].container = selector;
        }
        return chart;
    };
}

function readBinaryTree()
{
    var i = 1;
    var selector = '#binaryTree .connection .node';
```

```
var quantity = jQuery(selector).length; // Число связей
if (quantity > 0)
{
    var tree;
    // Перебор связей
    jQuery(selector).each(function()
    {
        var nodeParent, nodeName, nodeValue;
        nodeName = jQuery('#nodeName_' + i).val();
        nodeValue = parseInt(jQuery('#nodeValue_' + i).val());
        if (nodeName !== '' && !isNaN(nodeValue))
        {
            if (i === 1)
            {
                tree = new BinaryTree(nodeName, nodeValue);
            }
            else
            {
                nodeParent = jQuery('#nodeParent_' + i).val();
                if (nodeParent !== '')
                {
                    tree.add(nodeParent, nodeName, nodeValue);
                }
            }
        }

        i += 1;
    });
    if (tree instanceof BinaryTree)
    {
        var message = 'Исходное бинарное дерево:';
        // Исходное бинарное дерево
        var treeInput = Object.assign(
        {}, tree);
        // Отрисовка бинарного дерева
        jQuery('#tree_input').html(message);
        new Treant(treeInput.chart('#chart_input'));
        tree.swapEvenNodes();
```

```
            }

        }
    }

    /**
     * Выполняем действия когда DOM полностью загружен
     */
    jQuery(document).ready(function()
    {
        // Отслеживаем изменение числа узлов в бинарном дереве
        jQuery('#countNodes').on('keyup change', function()
        {
            jQuery('#tree_input').html('Заполните входные параметры');
            jQuery('#chart_input').html('');
            jQuery('#steps').html('');
            var countNodes = getCoutNodes();
            var message = ''; // Сообщение пользователю
            var inputData = ''; // Входные данные
            if (countNodes === '')
            {
                message = 'Введите число связей в бинарном дереве.';
            }
            else if (!isInt(countNodes))
            {
                message = 'Число узлов должно быть целочисленным
параметром.';
            }
            else if (!numPositive(countNodes) || countNodes === '0')
            {
                message = '<b>Ошибка! </b>Число узлов это положительное
значение и должен быть в дереве хотя бы один узел!';
            }
            else if (countNodes > 100)
            {
                message = '<b>Ошибка! </b>Число узлов не должно превышать
100';
            }
```

```
            if (message === '')
            {
                countNodes = parseInt(countNodes);
                for (var i = 1; i <= countNodes; i++)
                {
                    if (i === 1)
                    {
                        inputData += '<p><label class="connection">Связь ' +
i + ': ' +
                            '<input class="node" id="nodeName_' + i +
                            '" placeholder="Имя">' +
                            '<input class="node" id="nodeValue_' + i +
                            '" type="number" placeholder="Значение">' +
                            '(корневой узел)</label></p>';
                    }
                    else
                    {
                        inputData += '<p><label class="connection">Связь ' +
i + ': ' +
                            '<input class="node" id="nodeParent_' + i +
                            '" placeholder="В какой узел">' +
                            '<input class="node" id="nodeName_' + i +
                            '" placeholder="Имя узла">' +
                            '<input class="node" id="nodeValue_' + i +
                            '" type="number" placeholder="Значение узла">' +
                            '</label></p>';
                    }
                }
                jQuery('#binaryTree').html(inputData);
            }
            else
            {
                jQuery('#binaryTree').html('<p>' + message + '</p>');
            }
            // Отслеживаем изменение связей узлов
            jQuery('#binaryTree').on('keyup change', '.node', function()
            {
                readBinaryTree();
```

```
        });
    });

    /* Событие при изменении ширины окна */
    jQuery(window).resize(function()
    {
        readBinaryTree();
    });
});
```

3 Скриншоты выполнения программы

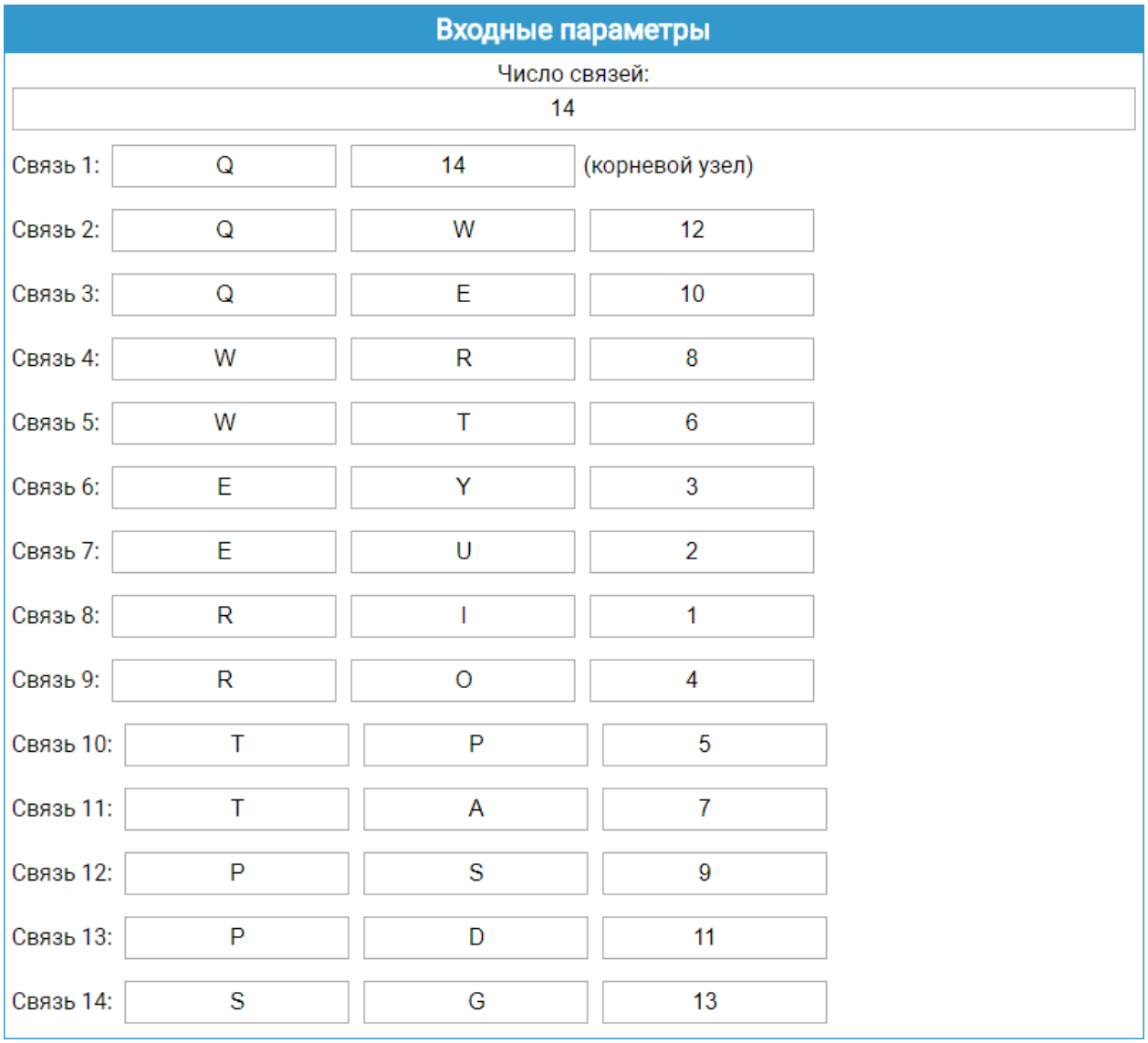

Рисунок 1 – Ввод входных параметров с клавиатуры

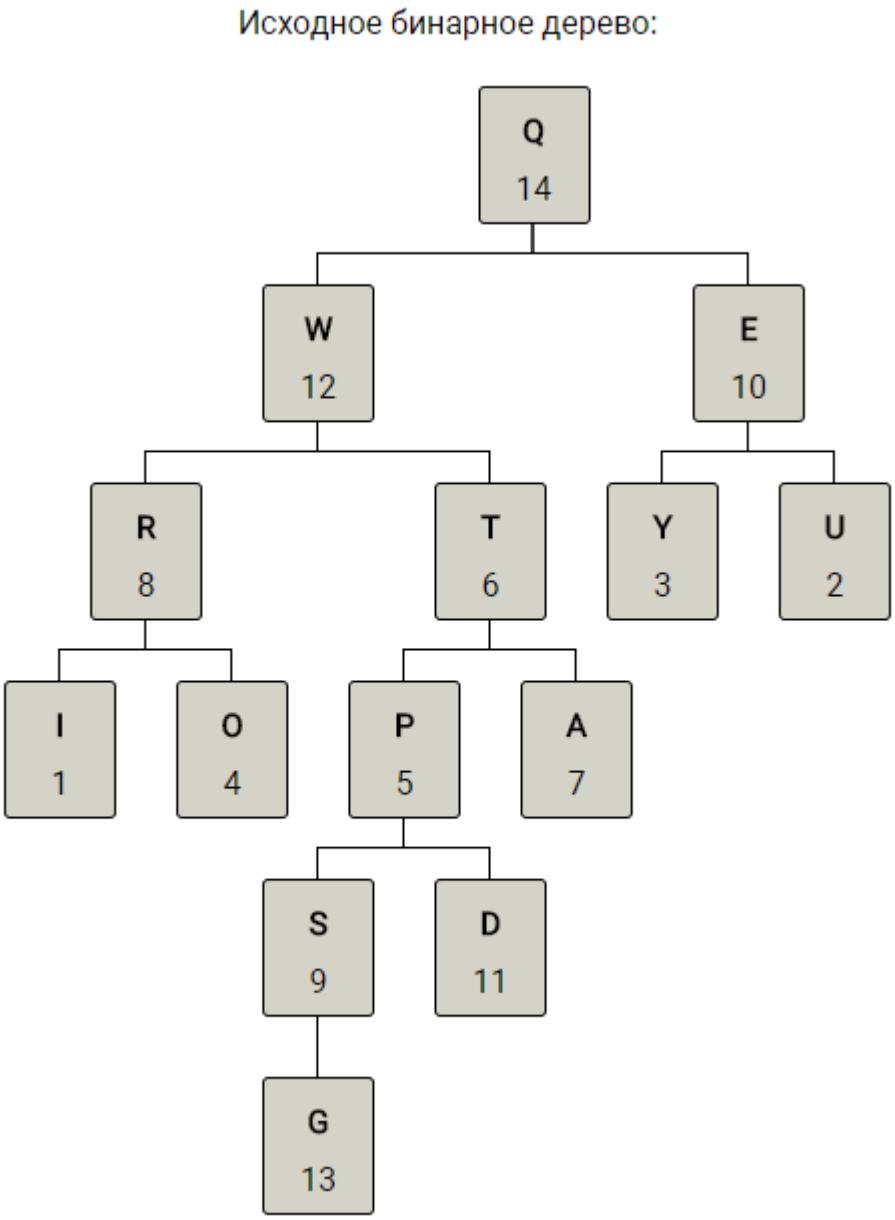

Рисунок 2 – Результат выполнения программы. Введенное бинарное дерево

Шаг №1. Проверяем узел Q(14). Меняем вершины текущего узла и дочернего узла W(12).

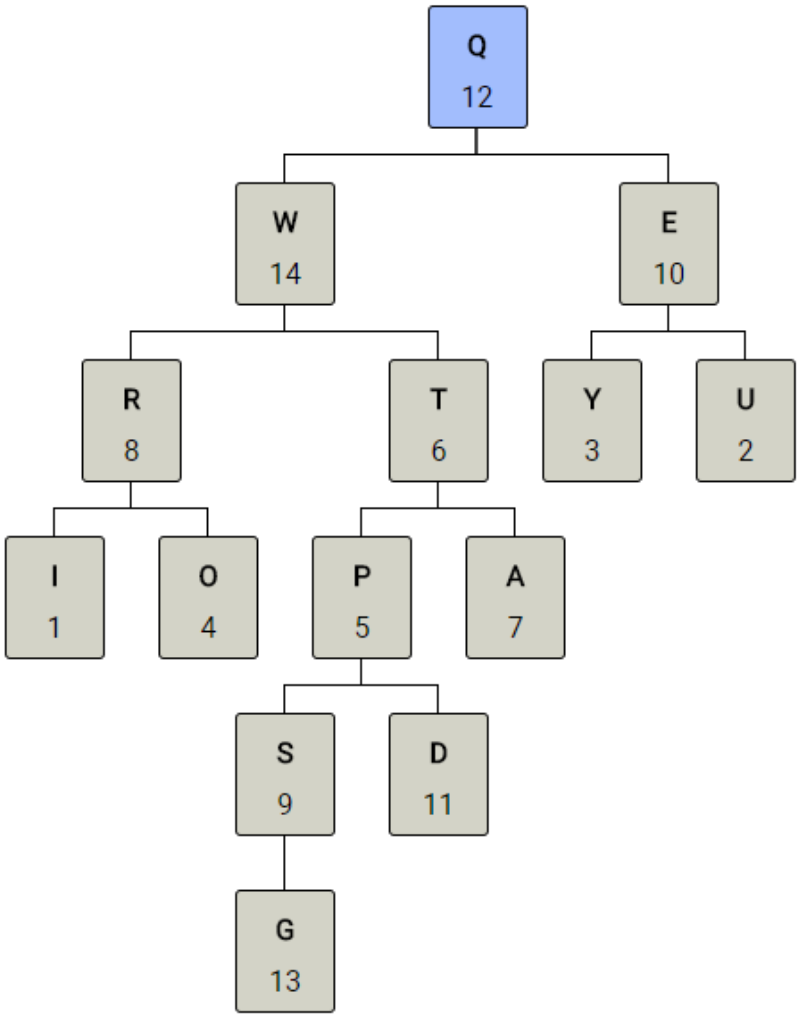

Рисунок 3 – Результат выполнения программы (шаг 1)

Шаг №2. Проверяем узел W(14). Меняем вершины текущего узла и дочернего узла R(8).

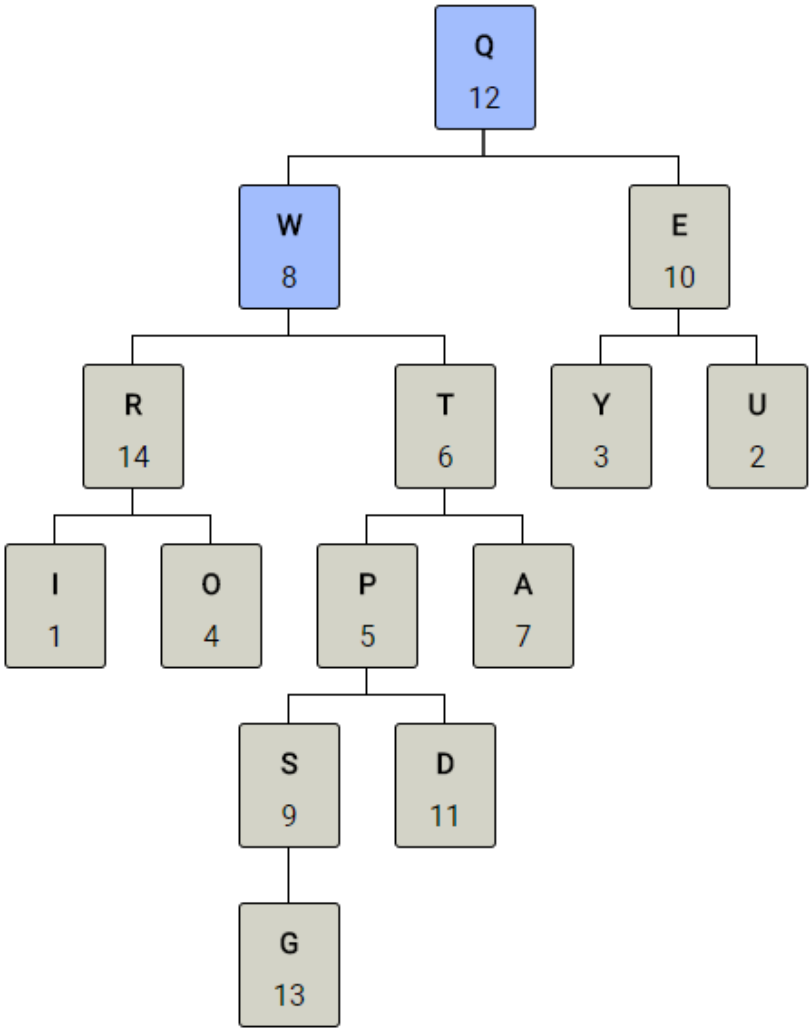

Рисунок 4 – Результат выполнения программы (шаг 2)

Шаг №3. Проверяем узел E(10). Меняем вершины текущего узла и дочернего узла U(2).

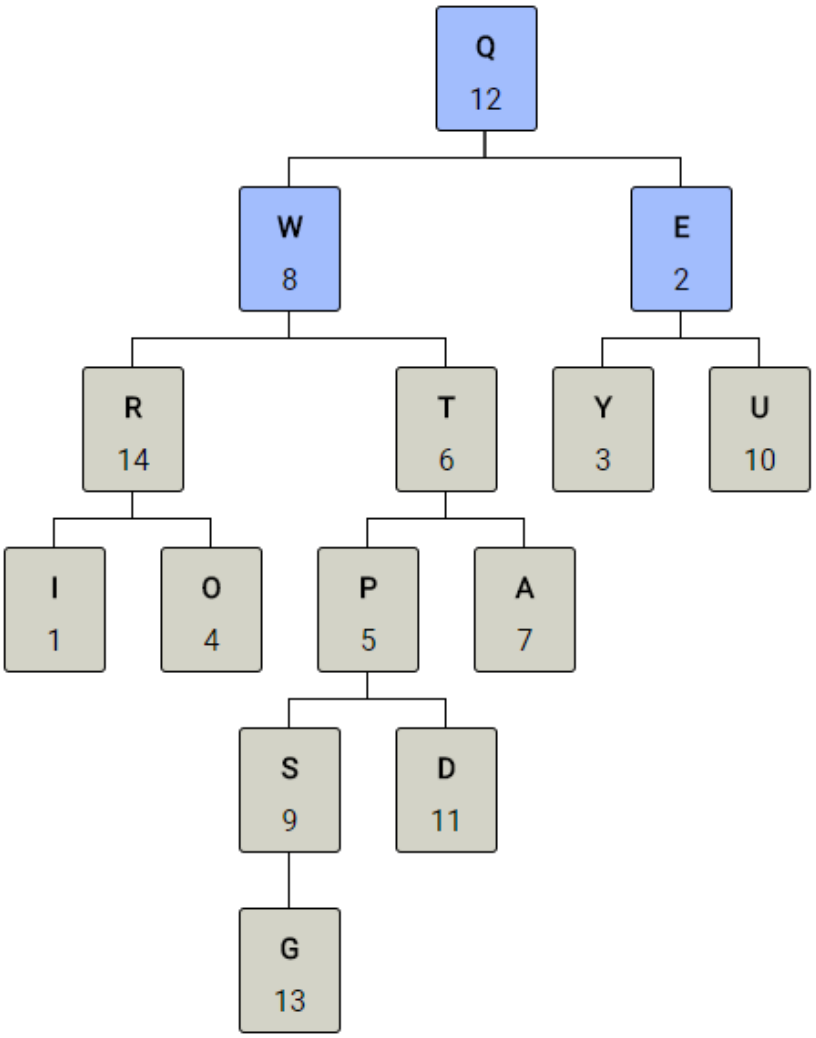

Рисунок 5 – Результат выполнения программы (шаг 3)

Шаг №4. Проверяем узел R(14). Меняем вершины текущего узла и дочернего узла O(4).

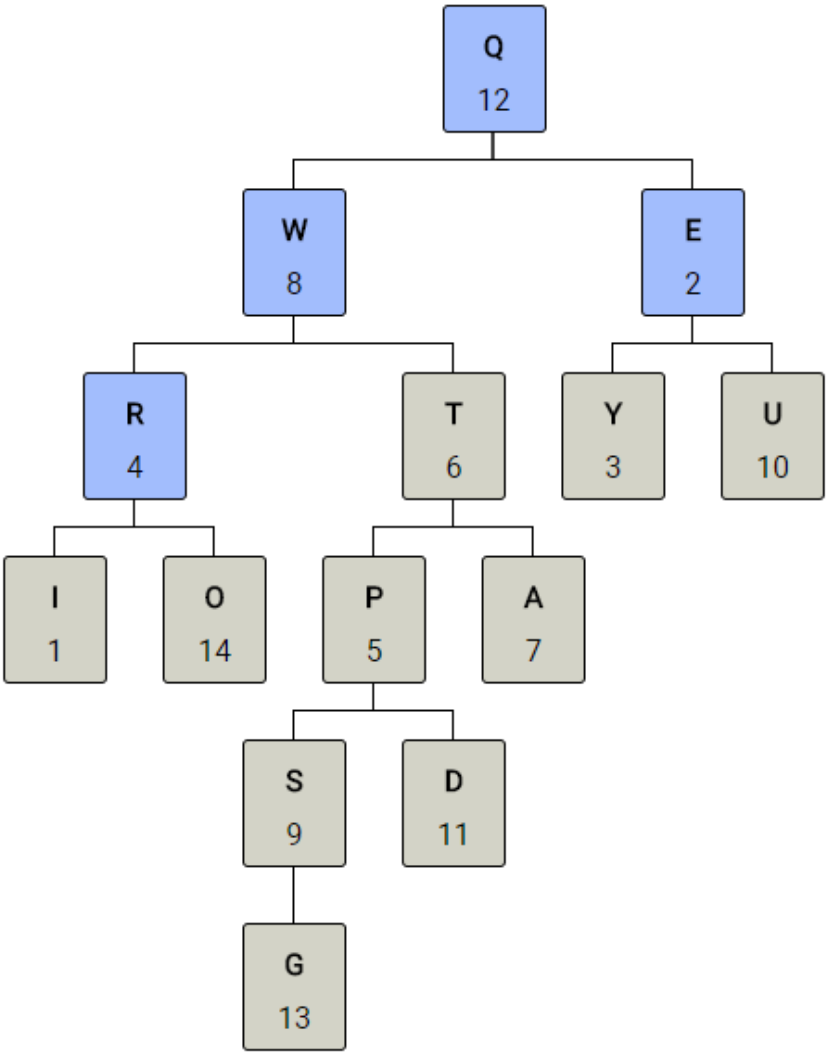

Рисунок 6 – Результат выполнения программы (шаг 4)

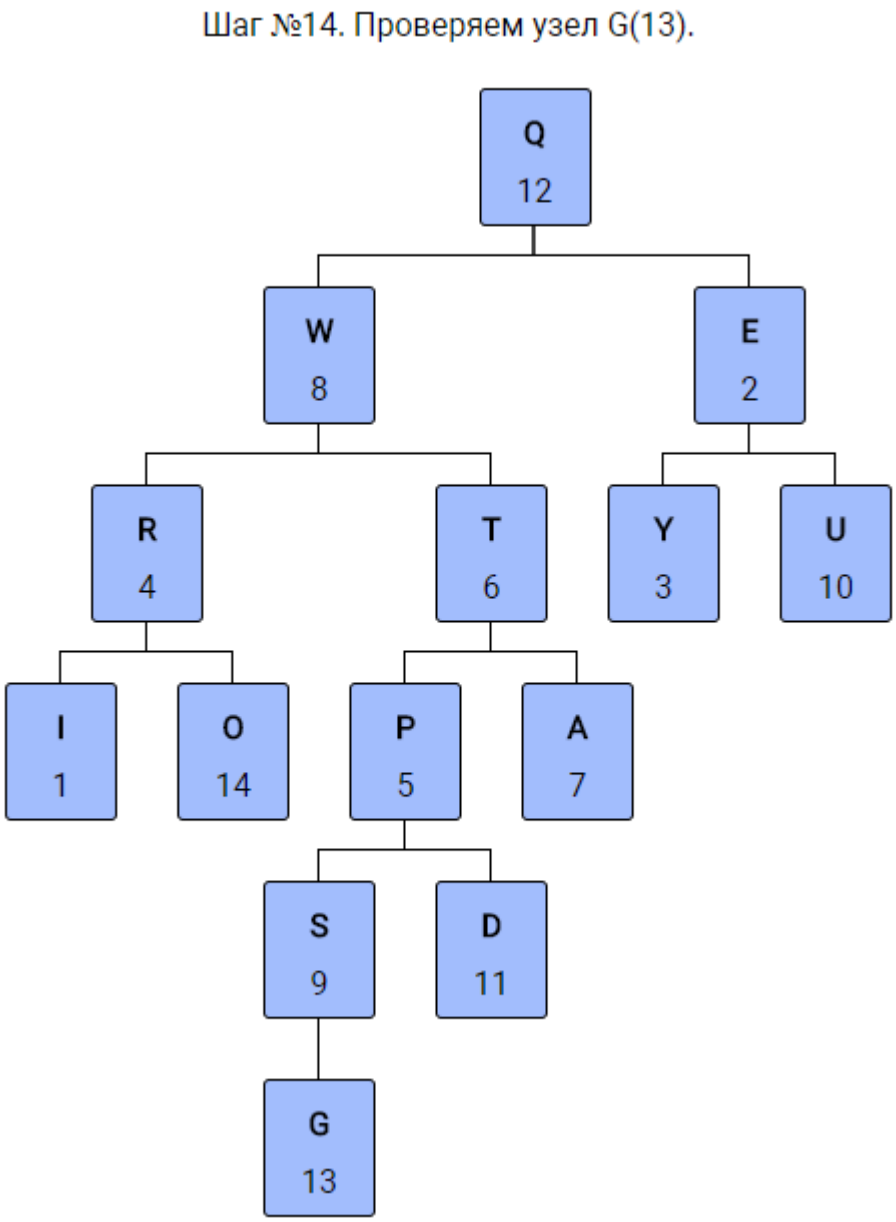

Рисунок 7 – Результат выполнения программы (последний шаг)

4 Заключение

В ходе выполнения лабораторной работы изучены принципы работы с рекурсией, графами и деревьями.

5 Используемая литература

1 Методические указания по выполнению лабораторных работ по дисциплине «Технологии программирования» – Нижний Новгород.: НГТУ, 2015. – 15 с.

2 Стандарт предприятия СТП 1-У-НГТУ-2004. Общие требования к оформлению пояснительных записок дипломных и курсовых проектов. – Нижний Новгород.: НГТУ, 2004. – 22 с.