НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«Московский энергетический институт»
Кафедра математического и компьютерного моделирования

«Технологии программирования»

Лабораторная работа
Вариант №16

Выполнил: Сошников С. А.
Группа: А-16-19

Преподаватель: Князев А.В.

Задание на лабораторную работу

Общее:

Разработать функции для включения описания идентификатора в таблицу, поиска по имени и исключения описания идентификатора из таблицы. Описание идентификатора состоит из имени идентификатора и его атрибута.

Разработать программу, демонстрирующую использование указанных функций. Программа должна быть разработана как консольное приложение на языке C# в среде Visual Studio.

Подготовить тесты для проверки работоспособности программы.

Отчёт по лабораторной работе должен содержать:

- Титульный лист
- Задание на работу (общее и индивидуальное)
- Описание работы программы
- Алгоритмы выполнения основных операций
- Тесты
- Распечатки экранов при работе программы
- Листинг программы

Индивидуальное:

Способ представления таблицы – сбалансированное дерево.
Метод поиска – бинарный поиск.

Описание работы программы

При включении программы создаётся пустое дерево, далее перед пользователем предстаёт меню, в котором нужно выбрать один из 5 вариантов:

- Ввод 1 => Добавление нового элемента в дерево
- Ввод 2=> Вывод дерева в упорядоченном порядке
- Ввод 3 => Поиск идентификатора по имени
- Ввод 4 => Удаление идентификатора по имени
- Ввод 5 => Прекращение работы программы

При введение иного значения пользователю будет выведено соответствующее сообщение с предложением ввести значение заново.
Более подробно алгоритмы выше будут расписаны на следующей странице.

Описание алгоритмов на псевдокоде

Создаётся пустое дерево => запускается бесконечный цикл с вводом пользовательского значения.

Вариант 1: пользователь ввёл «1»

Срабатывает функция добавления элемента в дерево, принимая стартовый узел и идентификатор, который нужно вставить => Если стартовый узел отсутствует, то на его месте создаётся узел с идентификатором => Если значение имени добавляемого идентификатора предшествует имени идентификатора в стартовом узле, то выполняется рекурсивный вызов функции добавления этого же идентификатора в корень левого поддерева стартового узла, иначе выполняется рекурсивный вызов функции добавления этого же идентификатора в корень правого поддерева стартового узла. => После отработки инструкций выполняется балансировка стартового узла.

Вариант 2: пользователь ввёл «2»

Срабатывает функция вывода дерева в порядке возрастания, в который передаётся корень дерева => Если корень левого поддерева существует, то функция рекурсивно вызывается для корня левого поддерева => Если корень левого поддерева не существует, выводится имя идентификатора в узле => После выполняется рекурсивный вызов для корня правого поддерева, если такой существует.

Вариант 3: пользователь ввёл «3»

Срабатывает функция поиска по имени, принимающая стартовый узел и искомое имя => Если стартовый узел не существует, возвращается null => Если имя идентификатора в стартовом узле и искомое имя совпадают, возвращается стартовый узел => Если имя идентификатора в стартовом узле опережает искомое имя, возвращается рекурсивный вызов для корня левого поддерева стартового узла => Если имя идентификатора в стартовом узле предшествует искомому имени, возвращается рекурсивный вызов для корня правого поддерева стартового узла

Вариант 4: пользователь ввёл «4»

Срабатывает функция удаления узла по заданному имени, принимающая стартовый узел и искомое имя => Если стартовый узел не существует, возвращается null => если искомое имя предшествует имени в стартовом узле, выполняется рекурсивный вызов для узла левого поддерева стартового узла => если искомое имя опережает имя в стартовом узле, то выполняется рекурсивный вызов для узла правого поддерева стартового узла => В случае равенства имён => Запоминаются корни левого и правого поддеревьев удаляемого узла => узел принимает значение null => если узел правого поддерева отсутствует, то возвращается узел левого поддерева (встаёт на место удалённого элемента) => Запоминается узел с минимальным именем в правом поддереве => Удаляется узел с минимальным именем в правом поддереве => В левый корень поддерева минимального элемента записывается корень левого поддерева удалённого элемента => Для удалённого минимума из правого поддерева вызывается балансировка => Для

удалённого искомого элемента выполняется балансировка

Вариант 5: пользователь ввёл «5»

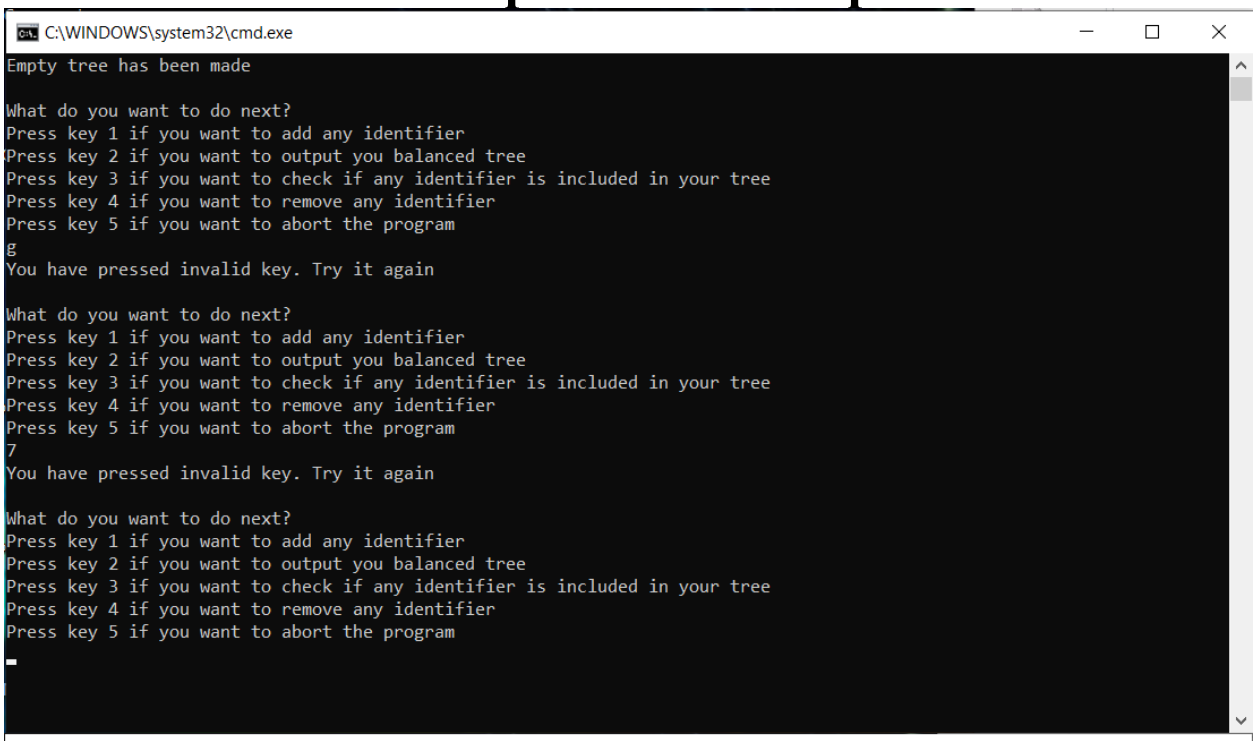
Срабатывает выход из бесконечного цикла => Выводится сообщение о завершении программы => Программа завершает свою работу

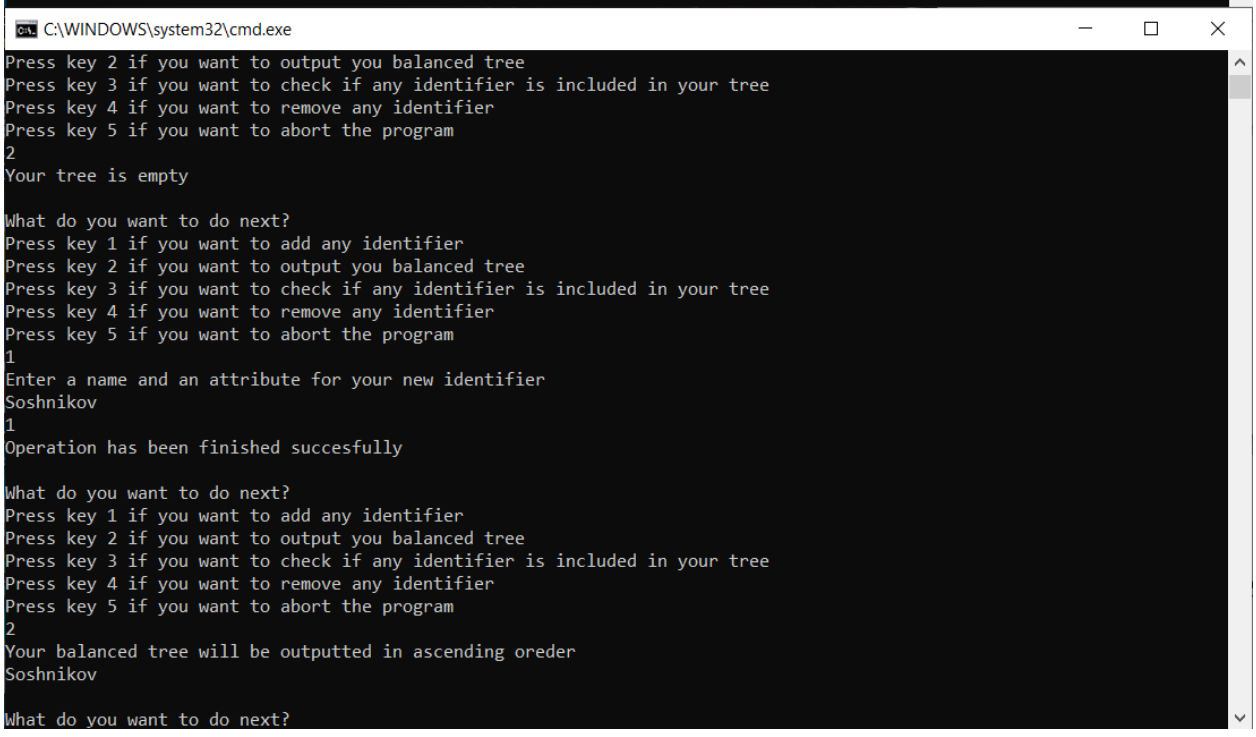
Вариант 6: пользователь ввёл любое отличное значение, от вышеперечисленных = > Через бесконечный цикл программа предлагает пользователю повторить ввод

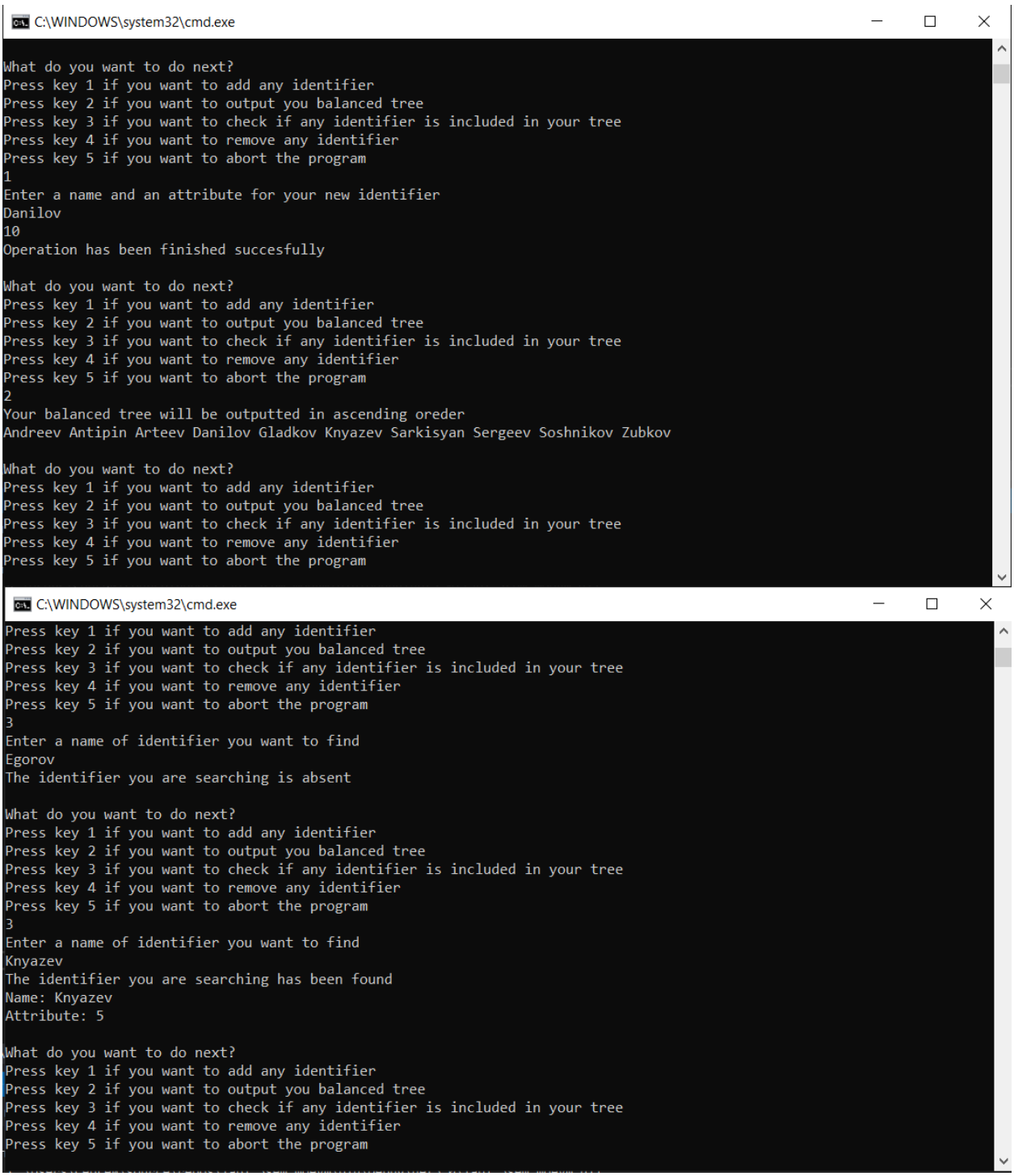
Служебные функции:

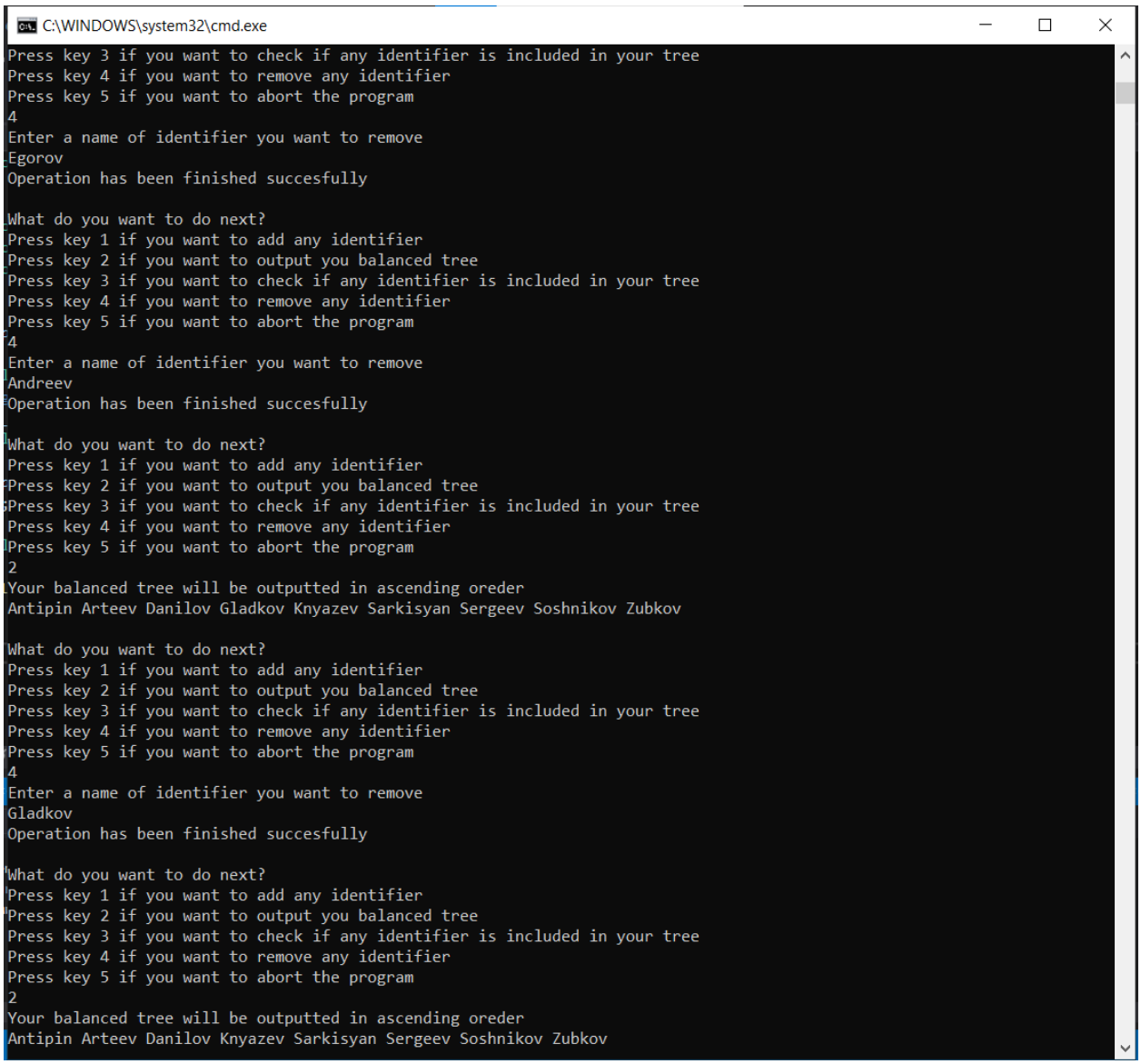
```
public int get_height(Node root) – возвращает параметр height (высота) для root
public int balance_factor(Node root) – возвращает разность между высотами правого и левого поддеревьев для root
public void fix_height(Node root) – устанавливает значение height (высота) для root
public Node small_rotate_right(Node p) – выполняет малое правое вращение вокруг p
public Node small_rotate_left(Node q) – выполняет малое левое вращение вокруг p
public Node find_min(Node start_node) – находит минимальное значение в дереве с корнем start_node
public Node delete_min(Node start_node) – удаляет минимальное значение из дерева с корнем в start_node
public Node make_balance(Node p) – выполняет балансировку для узла p
```

Тесты и распечатки экрана






C:\WINDOWS\system32\cmd.exe
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
1
Enter a name and an attribute for your new identifier
Danilov
10
Operation has been finished succesfully
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
2
Your balanced tree will be outputted in ascending oreder
Andreev Antipin Arteev Danilov Gladkov Knyazev Sarkisyan Sergeev Soshnikov Zubkov
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
C:\WINDOWS\system32\cmd.exe
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
3
Enter a name of identifier you want to find
Egorov
The identifier you are searching is absent
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
3
Enter a name of identifier you want to find
Knyazev
The identifier you are searching has been found
Name: Knyazev
Attribute: 5
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program


C:\WINDOWS\system32\cmd.exe
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
4
Enter a name of identifier you want to remove
Egorov
Operation has been finished succesfully
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
4
Enter a name of identifier you want to remove
Andreev
Operation has been finished succesfully
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
2
Your balanced tree will be outputted in ascending oreder
Antipin Arteev Danilov Gladkov Knyazev Sarkisyan Sergeev Soshnikov Zubkov
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
4
Enter a name of identifier you want to remove
Gladkov
Operation has been finished succesfully
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
2
Your balanced tree will be outputted in ascending oreder
Antipin Arteev Danilov Knyazev Sarkisyan Sergeev Soshnikov Zubkov

```
What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
1
Enter a name and an attribute for your new identifier
Gladkov
999
Operation has been finished succesfully

What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
1
Enter a name and an attribute for your new identifier
Gladkov
555
The identifier with this name is already in the tree. Operation has been failed

What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
2
Your balanced tree will be outputted in ascending oreder
Antipin Arteev Danilov Gladkov Knyazev Sarkisyan Sergeev Soshnikov Zubkov

What do you want to do next?
Press key 1 if you want to add any identifier
Press key 2 if you want to output you balanced tree
Press key 3 if you want to check if any identifier is included in your tree
Press key 4 if you want to remove any identifier
Press key 5 if you want to abort the program
```

Листинг программы

```
using System;

namespace lab1_5sem_moevm
{
    struct Identifier
    {
        public string name { get; set; }
        public int attribute { get; set; }
        public Identifier(string new_name, int new_attribute)
        {
            name = new_name;
            attribute = new_attribute;
        }
    }
    class Node
    {
        public Identifier info { get; set; }
        public Node left_down { get; set; } = null;
        public Node right_down { get; set; } = null;
        public int height { get; set; } = 1;
        public Node(Identifier new_info) => info = new_info;
    }
    class Balanced_tree
    {
        public Node root { get; set; } = null;
        public void output_tree(Node root)
        {
            if (root.left_down != null)
                output_tree(root.left_down);
            Console.Write(root.info.name);
            Console.Write(" ");
            if (root.right_down != null)
                output_tree(root.right_down);
        }
        public int get_height(Node root)
        {
            if (root == null)
                return 0;
            else
                return root.height;
        }
        public int balance_factor(Node root) => get_height(root.right_down) -
get_height(root.left_down);
        public void fix_height(Node root)
        {
            int h_left = get_height(root.left_down);
            int h_right = get_height(root.right_down);
            root.height = (h_left > h_right ? h_left : h_right) + 1;
        }
        public Node small_rotate_right(Node p)
        {
            Node q = p.left_down;
            p.left_down = q.right_down;
            q.right_down = p;
            fix_height(p);
            fix_height(q);
            return q;
        }
        public Node small_rotate_left(Node q)
        {
            Node p = q.right_down;
            q.right_down = p.left_down;
            p.left_down = q;
```

```
    fix_height(q);
    fix_height(p);
    return p;
}
public Node make_balance(Node p)
{
    fix_height(p);
    if (balance_factor(p) == 2)
    {
        if (balance_factor(p.right_down) < 0)
            p.right_down = small_rotate_right(p.right_down);
        return small_rotate_left(p);
    }
    if (balance_factor(p) == -2)
    {
        if (balance_factor(p.left_down) > 0)
            p.left_down = small_rotate_left(p.left_down);
        return small_rotate_right(p);
    }
    return p;
}
public Node find_min(Node start_node)
{
    if (start_node.left_down == null)
        return start_node;
    else
        return find_min(start_node.left_down);
}
public Node delete_min(Node start_node)
{
    if (start_node.left_down == null)
        return start_node.right_down;
    start_node.left_down = delete_min(start_node.left_down);
    return make_balance(start_node);
}
public Node add_new(Node start_node, Identifier new_node)
{
    if (start_node == null)
        return new Node(new_node);
    if (String.Compare(new_node.name, start_node.info.name) < 0)
        start_node.left_down = add_new(start_node.left_down, new_node);
    else
        start_node.right_down = add_new(start_node.right_down, new_node);
    return make_balance(start_node);
}
public Node search(Node start_node, string need_name)
{
    if (start_node == null)
        return null;
    if (String.Equals(need_name, start_node.info.name))
        return start_node;
    if (String.Compare(need_name, start_node.info.name) < 0)
        return search(start_node.left_down, need_name);
    else
        return search(start_node.right_down, need_name);
}
public Node delete_elem(Node root, string need)
{
    if (root == null)
        return null;
    if (String.Compare(need, root.info.name) < 0)
        root.left_down = delete_elem(root.left_down, need);
    else if (String.Compare(need, root.info.name) > 0)
        root.right_down = delete_elem(root.right_down, need);
    else if (String.Equals(need, root.info.name))
```

```
            {
                Node q = root.left_down;
                Node r = root.right_down;
                root = null;
                if (r == null)
                    return q;
                Node min = find_min(r);
                min.right_down = delete_min(r);
                min.left_down = q;
                return make_balance(min);
            }
            return make_balance(root);
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Balanced_tree table_of_identifiers = new Balanced_tree();
            Console.WriteLine("Empty tree has been made");
            while (true)
            {
                Console.WriteLine();
                Console.WriteLine("What do you want to do next?");
                Console.WriteLine("Press key 1 if you want to add any identifier");
                Console.WriteLine("Press key 2 if you want to output you balanced tree");
                Console.WriteLine("Press key 3 if you want to check if any identifier is
included in your tree");
                Console.WriteLine("Press key 4 if you want to remove any identifier");
                Console.WriteLine("Press key 5 if you want to abort the program");
                string pointer = Console.ReadLine();
                if (pointer == "1")
                {
                    Console.WriteLine("Enter a name and an attribute for your new
identifier");
                    string name = Console.ReadLine();
                    int attribute = Convert.ToInt32(Console.ReadLine());
                    Identifier identifier = new Identifier(name, attribute);
                    if (table_of_identifiers.search(table_of_identifiers.root,
identifier.name) == null)
                    {
                        table_of_identifiers.root =
table_of_identifiers.add_new(table_of_identifiers.root, identifier);
                        Console.WriteLine("Operation has been finished succesfully");
                    }
                    else
                        Console.WriteLine("The identifier with this name is already in
the tree. Operation has been failed");
                }
                else if (pointer == "2")
                {
                    if (table_of_identifiers.root != null)
                    {
                        Console.WriteLine("Your balanced tree will be outputted in
ascending oreder");
                        table_of_identifiers.output_tree(table_of_identifiers.root);
                        Console.WriteLine();
                    }
                    else
                        Console.WriteLine("Your tree is empty");
                }
                else if (pointer == "3")
                {
                    Console.WriteLine("Enter a name of identifier you want to find");
                    string need_identifier = Console.ReadLine();
```

```
                    Node result_of_search =
table_of_identifiers.search(table_of_identifiers.root, need_identifier);
                    if (result_of_search == null)
                        Console.WriteLine("The identifier you are searching is absent");
                    else
                    {
                        Console.WriteLine("The identifier you are searching has been
found");
                        Console.WriteLine($"Name: {result_of_search.info.name}");
                        Console.WriteLine($"Attribute:
{result_of_search.info.attribute}");
                    }
                }
                else if (pointer == "4")
                {
                    Console.WriteLine("Enter a name of identifier you want to remove");
                    string need_identifier = Console.ReadLine();
                    table_of_identifiers.root =
table_of_identifiers.delete_elem(table_of_identifiers.root, need_identifier);
                    Console.WriteLine("Operation has been finished succesfully");
                }
                else if (pointer == "5")
                    break;
                else
                    Console.WriteLine("You have pressed invalid key. Try it again");
            }
            Console.WriteLine("The programm has been aborted");
        }
    }
}
```