МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Кафедра комп’ютерних наук

ЗВІТ
З ЛАБОРАТОРНОЇ РОБОТИ №8

Студента 2 курсу групи 244А
Спеціальності: 122 Комп’ютерні науки
Андроніка Веніаміна Ігоровича
Керівник Угрин Дмитро Ілліч
Варіант - 1

м.Чернівці
2022 рік

Фундаментальні алгоритми на деревах.

Завдання.

1. Створити й вивести на екран дерево, елементи якого вводяться із клавіатури й мають цілий тип. Причому для кожного вузла дерева у всіх лівих вузлах повинні перебувати числа менші, а в правих більші, ніж числа, що зберігаються в цьому вузлі. Таке дерево називається деревом пошуку.
2. Здійснити обхід дерева в прямому, зворотньому та симетричному порядках.
3. Написати функцію, яка знаходить найбільший та найменший елементи дерева.
4. Написати процедуру, яка видаляє з дерева всі парні елементи.
5. Написати процедуру, яка визначає число входжень заданого елемента в дерево.
6. Написати функцію, яка підраховує суму всіх елементів дерева.

Хід роботи

Клас, описуючий вузол(корінь та дві гілки)

```
Ссылок: 20 | kazmi, 6 ч назад | Автор: 1, изменение: 1
class Node
{
    Ссылок: 17 | kazmi, 6 ч назад | Автор: 1, изменение: 1
    public Node? LeftNode { get; set; } = null;
    Ссылок: 16 | kazmi, 6 ч назад | Автор: 1, изменение: 1
    public Node? RightNode { get; set; } = null;
    Ссылок: 21 | kazmi, 6 ч назад | Автор: 1, изменение: 1
    public int Data { get; set; }
}
```

Методи, які реалізують додавання елементів в дерево. Ввід через консоль або випадкові числа

```
Ссылок: 1 | kazmi, 6 ч назад | Автор: 1, изменений: 2
public void AutoAdd(int numbers) // додавання в дерево numbers вибадкових значень
{
    for (int i = 0; i < numbers; i++)
        Add(new Random().Next(1,100));
}
```

```
Ссылок: 2 | kazmi, 6 ч назад | Автор: 1, изменений: 7
public bool Add(int value) // додавання певного елемента в дерево
{
    Node? before = null;
    Node? after = this.Root;

    while (after != null)
    {
        before = after;
        if (value <= after.Data) // чи є новий вузол у лівому дереві
            after = after.LeftNode;
        else if (value > after.Data) // чи є новий вузол у правому дереві
            after = after.RightNode;
    }

    Node newNode = new Node();
    newNode.Data = value;

    if (this.Root == null) // дерево порожнє
        this.Root = newNode;
    else
    {
        if (value < before?.Data)
            before.LeftNode = newNode;
        else
            before.RightNode = newNode;
    }

    Count++;
    return true;
}
```

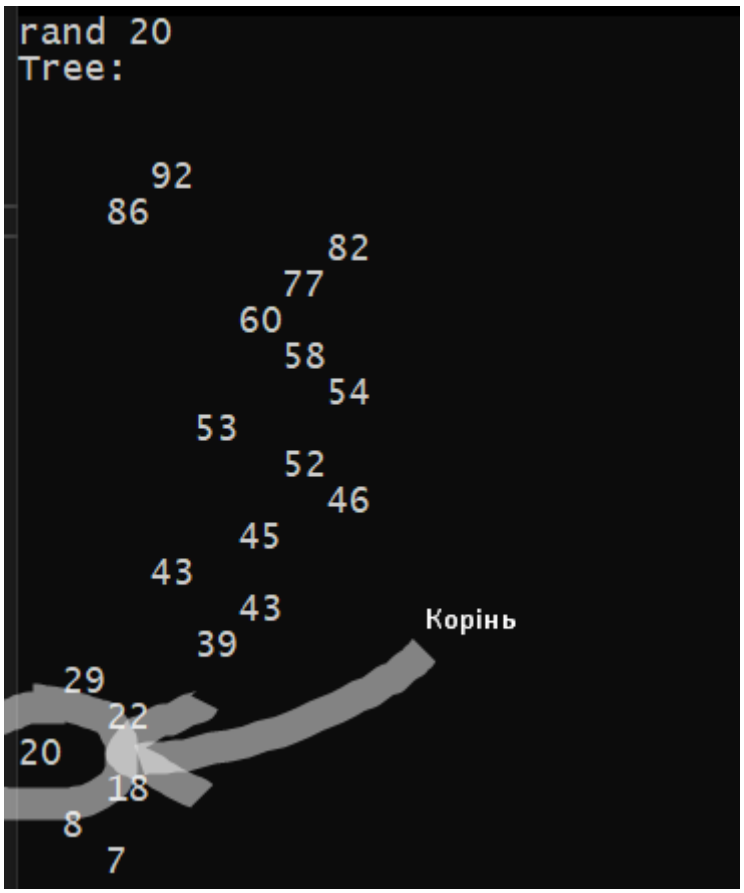
Виконання методів в програмі

```
BinaryTree binaryTree = new();

while (true)
{
    string[]? value = Console.ReadLine()?.Split(' ');

    if (value != null)
    {
        if (value[0]?.ToLower() == "end")
            break;
        else if (value[0]?.ToLower() == "rand")
        {
            binaryTree.AutoAdd(Convert.ToInt32(value[1]));
            break;
        }
        else if (value[0] != "")
            binaryTree.Add(Convert.ToInt32(value[0]));
    }
}
```

Метод відповідаючий за вивід дерева

```
Ссылок: 3 | kazmi, 6 ч назад | Автор: 1, изменений: 4
public void Print_Tree(int l, Node? Tree) // вивід дерева
{
    if (Tree != null)
    {
        Print_Tree(l + 2, Tree.RightNode);

        for (int i = 1; i <= l; i++) Console.Write(" ");
        Console.WriteLine(Tree.Data);

        Print_Tree(l + 2, Tree.LeftNode);
    }
}
```

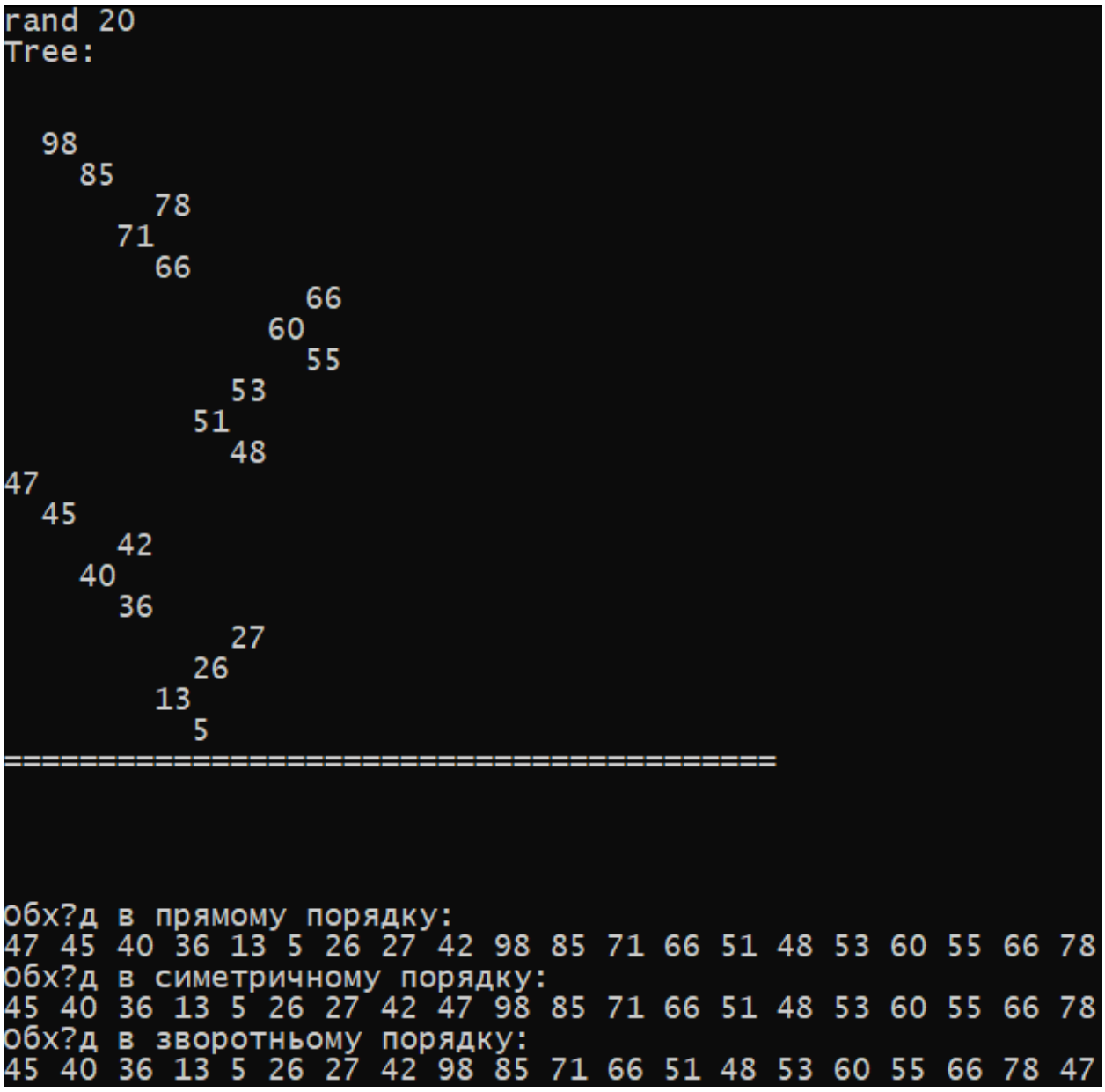
Результат виводу в консолі(вниз іде ліва гілка, вверх права)



Методи прямого, симетричного, зворотнього обходів

```
Ссылок: 7 | kazmi, 6 ч назад | Автор: 1, изменение: 1
public void PreOrder(Node? Tree) // обхід в прямому порядку
{
    if (Tree == null) return;

    Console.Write(Tree.Data + " ");
    PreOrder(Tree.LeftNode);
    PreOrder(Tree.RightNode);
}

Ссылок: 1 | kazmi, 6 ч назад | Автор: 1, изменение: 1
public void InOrder(Node? Tree) // обхід в симетричному порядку
{
    if (Tree == null) return;

    PreOrder(Tree.LeftNode);
    Console.Write(Tree.Data + " ");
    PreOrder(Tree.RightNode);
}

Ссылок: 1 | kazmi, 6 ч назад | Автор: 1, изменение: 1
public void PostOrder(Node? Tree) // обхід в зворотньому порядку
{
    if (Tree == null) return;

    PreOrder(Tree.LeftNode);
    PreOrder(Tree.RightNode);
    Console.Write(Tree.Data + " ");
}
```

Результат в консолі



Пошук максимального та мінімального значень

```
Ссылок: 1 | kazmi, 6 ч назад | Автор: 1, изменений: 2
public int FindMax(Node Tree) // максимальне значення
{
    if (Tree == null || Tree?.Data == null) throw new Exception("Tree is null");


    while (Tree.RightNode != null) Tree = Tree.RightNode;

    return Tree.Data;
}

Ссылок: 2 | kazmi, 6 ч назад | Автор: 1, изменений: 2
public int FindMin(Node Tree) // мінімальне значення
{
    if (Tree == null || Tree?.Data == null) throw new Exception("Tree is null");


    while (Tree.LeftNode != null) Tree = Tree.LeftNode;

    return Tree.Data;
}
```

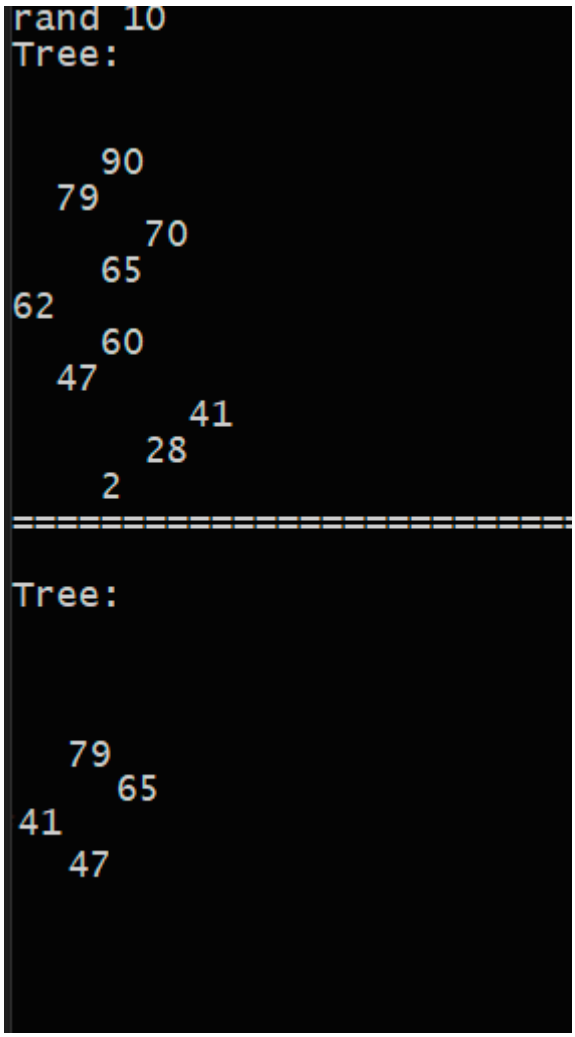
Результат в консолі

```
rand 20
Tree:


      99
    89
      84
        82
  79
    76
      72
70
      64
    62
  58
    47
      41
            40
          39
        35
            35
              11
          5
            4
========================================



Max: 99
Min: 4
```

Видалення парних елементів. Розбив на декілька частин, тому що не получалось реалізувати на пряму. Шукаю парний елемент, запускаю метод який видаляє за значенням і видаляю цей елемент, і так з усіма елементами

```
Ссылок: 4 | kazmi, 6 ч назад | Автор: 1, изменений: 2
public int FindEvenElements(Node? Tree) // пошук парного елемента
{
    if (Tree == null || Tree?.Data == null) throw new Exception("Tree is null");

    if (Tree.Data % 2 == 0) return Tree.Data;
    FindEvenElements(Tree.LeftNode);
    FindEvenElements(Tree.RightNode);

    return 0;
}

Ссылок: 0 | kazmi, 6 ч назад | Автор: 1, изменений: 7
public Node? DeleteEvenValue(Node? Tree) // видалення парних елементів
{
    for (int i = 0; i < Count; i++)
    {
        Console.WriteLine(FindEvenElements(Tree));
        Tree = Remove(Tree, FindEvenElements(Tree));
    }

    return Tree;
}

Ссылок: 4 | kazmi, 6 ч назад | Автор: 1, изменений: 2
private Node? Remove(Node? parent, int key) // видалення елемента за значенням
{
    if (parent == null) return parent;

    if (key < parent.Data) parent.LeftNode = Remove(parent.LeftNode, key);
    else if (key > parent.Data) parent.RightNode = Remove(parent.RightNode, key);
    else
    {
        if (parent.LeftNode == null)
            return parent.RightNode;
        else if (parent.RightNode == null)
            return parent.LeftNode;

        parent.Data = FindMin(parent.RightNode);

        parent.RightNode = Remove(parent.RightNode, parent.Data);
    }

    Count--;
    return parent;
}
```

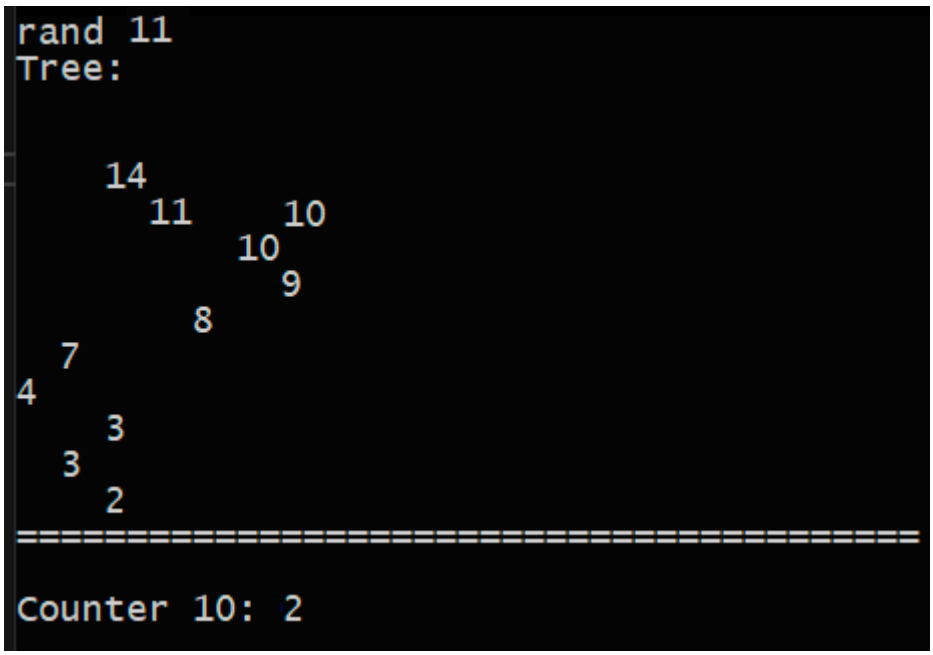
Результат в консолі



Метод, який рахує кількість входжень вказаного елемента

```
Ссылок: 4 | kazmi, 7 ч назад | Автор: 1, изменение: 1
public int Counter(Node? Tree, int key, int num = 0) // кількість входжень елемента
{
    if (Tree == null) return 0;

    if (Tree.Data == key) num++;
    Counter(Tree.LeftNode, key, num);
    Counter(Tree.LeftNode, key, num);

    return num;
}
```

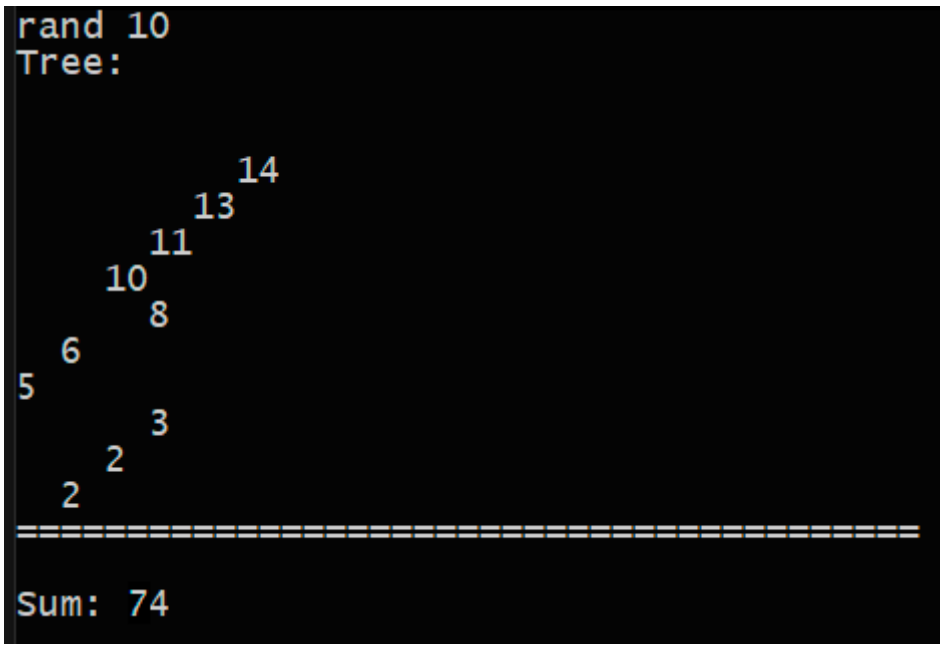
Результат в консолі



Метод, який рахує суму всіх елементів дерева

```
Ссылок: 0 | kazmi, 7 ч назад | Автор: 1, изменение: 1
public int Sum(Node? Tree, int sum = 0) // сума всіх елеметів
{
    if (Tree == null) return 0;

    sum += Tree.Data;
    Counter(Tree.LeftNode, sum);
    Counter(Tree.LeftNode, sum);

    return sum;
}
```

Результат в консолі



Висновки: вивчив способи ефективного зберігання та обробки інформації на прикладі бінарних дерев.