

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації
і управління

Звіт

з лабораторної роботи № 5 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

«Дерезовидні структури даних»

Виконав(ла)

ІІІ-02 Гуцїн Дмитро
(шифр, прізвище, ім'я, по батькові)

Перевірив

Головченко М.М.
(прізвище, ім'я, по батькові)

Київ 2021

ЗМІСТ

1	МЕТА ЛАБОРАТОРНОЇ РОБОТИ	3
2	ЗАВДАННЯ	4
3	ВИКОНАННЯ.....	4
3.1	ПСЕВДОКОД АЛГОРИТМІВ.....	8
3.2	ПРОГРАМНА РЕАЛІЗАЦІЯ.....	8
3.2.1	<i>Вихідний код.....</i>	<i>8</i>
3.2.2	<i>Приклади роботи</i>	<i>10</i>
	ВИСНОВОК	12
	КРИТЕРІЇ ОЦІНЮВАННЯ	13

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні підходи формалізації та імплементції алгоритмів побудови та обробки базових деревовидних структур даних.

2 ЗАВДАННЯ

Розробити алгоритм розв'язання задачі відповідно до варіанту. Виконати програмну реалізацію задачі. Не використовувати вбудовані деревовидні структури даних (контейнери). Зробити висновок по лабораторній роботі.

Варіанти завдань.

1. Задано фрагмент програми мовою C++. Надрукувати в алфавітному порядку всі ідентифікатори цієї програми, вказавши для кожного з них число входжень у текст програми. Для збереження ідентифікаторів використати структуру типу дерева, елементами якого є ідентифікатор і число його входжень у текст.

2. Побудувати дерево, елементами якого є дійсні числа. Обчислити середнє арифметичне усіх його елементів.

3. Побудувати дерево, що відображає формулу $(a*(b+c))/d$, де коренем дерева та його підкореннями є операції $*, +, -, /$, а листками - змінні a, b, c, d . Надрукувати відповідне дерево .

4. Побудувати дерево, елементами якого є числа. Надрукувати дерево. Визначити кількість від'ємних та додатніх елементів дерева.

5. Побудувати двійкове дерево, елементами якого є символи. Визначити, чи знаходиться у цьому дереві елемент, значення якого вводиться з клавіатури. Якщо елемент знайдений, то підрахувати число його входжень.

6. Побудувати двійкове дерево пошуку з літер заданого рядка. Видалити з дерева літери, що зустрічаються більше одного разу. Вивести елементи дерева, що залишилися, при його постфіксному обході.

7. Заданий текст. Підрахувати кількість повторень кожного слова. Побудувати дерево із слів тексту. Слова, що зустрічаються найчастіше розмістити на верхньому рівні, на інших рівнях дерева розмістити слова з меншою кількістю повторень.

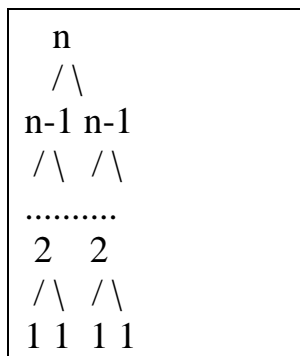
8. Побудувати бінарне дерево, елементами якого є дійсні числа. Знайти значення найбільшого елемента цього дерева та надрукувати його.

9. Заданий рядок символів латинського алфавіту. Побудувати дерево, в якому значеннями вершин є символи, що розміщуються на рівнях відповідно до кількості їх повторень у рядку.

10. Побудувати дерево, елементами якого є цілі числа. Визначити кількість вершин на n -му його рівні та кількість рівнів.

11. Побудувати дерево, що відображає формулу $((a+b)*c-d)$, де коренем дерева та його підкореннями є операції $+$, $-$, $*$, $/$, а листками є змінні a , b , c , d . Вивести значення дерева-формули. Надрукувати відповідні піддерева $y_1=a+b$, $y_2=y_1*c$, $y_3=y_2-d$.

12. Побудувати дерево наступного виду:



, де n - додатне ціле число

13. Побудувати бінарне дерево для зберігання даних виду: деталь, її кількість, постачальник. Забезпечити виконання операцій додавання нового елемента у дерево в діалоговому режимі та визначення постачальника найбільшої кількості деталей.

14. Побудувати дерево, елементами якого є символи. Визначити максимальну глибину дерева (число гілок на найбільшому з маршрутів від кореня дерева до листків).

15. Побудувати бінарне дерево, елементами якого є цілі числа. Підрахувати кількість вершин на n -му рівні цього дерева (нульовий рівень - корінь цього дерева) та надрукувати ці елементи.

16. Побудувати дерево, що відображає формулу $((a+b)/c)*d$, де коренем дерева та його підкореннями є операції, а листками - змінні. Ввести значення змінних та визначити значення дерева-формули. Надрукувати відповідні

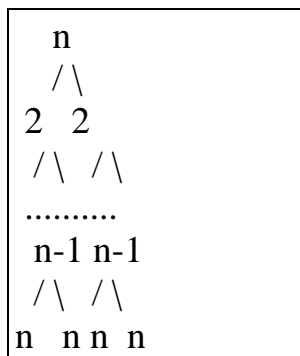
піддерева, наприклад: $y_1 = a + b$, $y_2 := y_1 / c$, $y_3 = y_2 * d$.

17. Відповідно до виразу, що вводиться з клавіатури та може містити операції $+$, $-$, $*$, $/$, побудувати дерево-формулу та обчислити значення цієї формули.

18. Побудувати бінарне дерево, елементами якого є слова. Знайти у ньому значення слова, введенного з клавіатури, визначивши номер відповідного рівня.

19. Побудувати дерево, елементами якого є символи. Знайти довжину шляху (число гілок) від кореня до значення символу, введенного з клавіатури. Різновид дерева вибрати самостійно.

20. Побудувати дерево наступного типу:



, де n - додатне ціле число

21. Побудувати два бінарних дерева, елементами якого є цілі числа. Об'єднати їх, уникаючи дублювання елементів в сумарному дереві.

22. Побудувати дерево, елементами якого є дійсні числа. Поміняти місцями найбільше та найменше значення дерева.

23. Побудувати бінарне дерево для зберігання даних виду: найменування товару, його кількість, вартість одиниці. Забезпечити виконання операцій додавання нового елемента у дерево в діалоговому режимі та підрахунку загальної вартості вказаного товару.

24. Побудувати двійкове дерево пошуку, в вершинах якого знаходяться слова введені з клавіатури. Визначити кількість вершин дерева, що містять слова, які починаються на зазначену букву.

25. Побудувати дерево, елементами якого є цілі числа. Визначити кількість вузлових вершин (не листків) даного дерева та надрукувати їх координати

(номер рівня та номер гілки).

26. Написати програму, що будує дерево-формулу та перетворює в ньому всі піддерева, що відповідають формулам $((f_1 \pm f_2) * f_3)$, на піддерева виду $((f_1 * f_3) \pm (f_2 * f_3))$.

27. Побудувати дерево, елементами якого є символи. Визначити і вивести на друк усі термінальні вершини (листя) цього дерева.

28. Побудувати і вивести на екран бінарне дерево наступного виразу: $9 + 8 * (7 + (6 * (5 + 4) - (3 - 2)) + 1))$. Реалізувати постфіксний, інфіксний та префіксний обходи дерева і вивести відповідні вирази на екран.

29. Побудувати двійкове дерево пошуку, в вершинах якого знаходяться цілі числа. Визначити максимальну висоту дерева, тобто число ребер в найдовшому шляху від кореня до листків.

30. Побудувати двійкове дерево пошуку, в вершинах якого знаходяться цілі числа. Визначити висоту дерева, тобто число ребер в шляху від кореня до заданої вершини.

31. Побудувати дерево, елементами якого є цілі числа. Видалити з дерева усі листки, надрукувати початкове та модифіковане дерево.

32. Побудувати двійкове дерево пошуку, в вершинах якого знаходяться цілі числа. Визначити чи є дерево збалансованим.

33. Побудувати бінарне дерево, елементами якого є цілі числа. Вивести усі вузли, які мають лише правого потомка і усі вузли, які мають лише лівого потомка.

34. Побудувати дерево, елементами якого є цілі числа. Видалити з дерева усі вузли, які є батьками листків, надрукувати початкове та модифіковане дерево.

35. Реалізувати структуру даних бінарна піраміда на основі бінарного дерева, передбачити можливість додавання та видалення елементів.

3 ВИКОНАННЯ

3.1 Псевдокод алгоритмів

```
Function Insert(root, data)
    if root == NULL
        root = GetNewNode(data)
    end if
    else if data <= root->data
        root->left = Insert(root->left, data)
    end else if
    else
        root->right = Insert(root->right, data)
    end else
    return root
End Function
```

```
Procedure printLevel(node, level, vect) {
    if node == nullptr
        return
    end if
    else
        printLevel(node->left, level - 1, vect)
        if level == 0
            vect.push_back(node->data)
        end if
        printLevel(node->right, level - 1, vect)
    end else
end procedure
```

3.2 Програмна реалізація

3.2.1 Вихідний код

```
#include <iostream>
#include <vector>

const int COUNT = 10;

using namespace std;

struct Node {
    int data;
    Node* left;
    Node* right;
};

Node* GetNewNode(int data);
Node* Insert(Node* root, int data);
void printLevel(Node* node, int level, vector<int>& vect);
Node* FillTree(Node* root);
void GetLevel(Node* root);
void ShowTree(Node* root, int space);

int main() {
    Node* root = NULL;
    root = FillTree(root);
    ShowTree(root, 0);
    cout << endl;
```



```

    GetLevel(root);
    system("pause");
    return 0;
}

Node* GetNewNode(int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}

Node* Insert(Node* root, int data) {
    if (root == NULL) {
        root = GetNewNode(data);
    }
    else if (data <= root->data) {
        root->left = Insert(root->left, data);
    }
    else {
        root->right = Insert(root->right, data);
    }
    return root;
}

void printLevel(Node* node, int level, vector<int>& vect) {
    if (node == nullptr) {
        return;
    }
    else {
        printLevel(node->left, level - 1, vect);
        if (level == 0) {
            vect.push_back(node->data);
        }
        printLevel(node->right, level - 1, vect);
    }
}

Node* FillTree(Node* root) {
    int count;
    cout << "Enter the amount of elements you want to enter: "; cin >> count;
    for (int i = 0; i < count; i++) {
        int elem;
        cout << "Enter a number: ";
        cin >> elem;
        root = Insert(root, elem);
    }
    return root;
}

void GetLevel(Node* root) {
    int level;
    cout << "Enter level: "; cin >> level;
    vector<int> vect;
    printLevel(root, level, vect);
    cout << "There are " << vect.size() << " elements on the " << level << " level: ";
    for (int i = 0; i < vect.size(); i++)
        cout << vect[i] << " ";
    cout << endl;
}

void ShowTree(Node* root, int space)
{
    if (root == NULL)

```

```

        return;

// Increase distance between levels
space += COUNT;

ShowTree(root->right, space);

// Print current node
cout << endl;
for (int i = COUNT; i < space; i++)
    cout << " ";
cout << root->data << endl;

ShowTree(root->left, space);
}

```

3.2.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

The screenshot shows a terminal window with the following text:

```

Enter the amount of elements you want to enter: 7
Enter a number: 6
Enter a number: 3
Enter a number: 8
Enter a number: 2
Enter a number: 5
Enter a number: 7
Enter a number: 9

```

The tree structure is displayed as follows:

```

          9
        /  \
       8    7
      /  \  /  \
     6   5 3   5
    /  \ /  \ /  \
   2   3 2   3 2   3

```

Below the tree, the program prompts for a level and displays the elements at that level:

```

Enter level: 2
There are 4 elements on the 2 level: 2 5 7 9
Для продолжения нажмите любую клавишу . . .

```

Рисунок 3.1

```
Enter the amount of elements you want to enter: 15
Enter a number: 7
Enter a number: 3
Enter a number: 11
Enter a number: 1
Enter a number: 5
Enter a number: 9
Enter a number: 13
Enter a number: 0
Enter a number: 2
Enter a number: 4
Enter a number: 6
Enter a number: 8
Enter a number: 10
Enter a number: 12
Enter a number: 14

      14
    13
  12
11
      10
    9
      8
7
      6
    5
      4
    3
      2
    1
      0

Enter level: 3
There are 8 elements on the 3 level: 0 2 4 6 8 10 12 14
Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.2

ВИСНОВОК

При виконанні даної лабораторної роботи ми вивчили основні підходи формалізації та імплементації алгоритмів побудови та обробки базових деревовидних структур даних, розробили алгоритм розв'язання задачі відповідно до варіанту, а також виконали програмну реалізацію задачі на мові C++, не використовуючи контейнери.

КРИТЕРІЇ ОЦІНЮВАННЯ

За умови здачі лабораторної роботи до 27.04.2020 включно максимальний бал дорівнює – 5. Після 27.04.2020 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

- псевдокод алгоритму – 10%;
- програмна реалізація алгоритму – 80%;
- висновок – 10%.