Об’єктно орієнтоване програмування

**Лабораторна робота №6**

# Бінарне дерево

# Теоретичні відомості

Бінарне (двійкове) дерево (binary tree) – це впорядковане дерево, кожна вершина якого має не більше двох піддерев, причому для кожного вузла виконується правило: в лівому піддереві містяться тільки ключі, що мають значення, менші, ніж значення даного вузла, а в правому піддереві містяться тільки ключі, що мають значення, більші, ніж значення даного вузла.

Бінарне дерево є рекурсивної структурою, оскільки кожне його піддерево саме є бінарним деревом і, отже, кожен його вузол в свою чергу є коренем дерева. Вузол дерева, що не має нащадків, називається листом. Графічне зображення бінарного дерева показано на рис. 6.1.

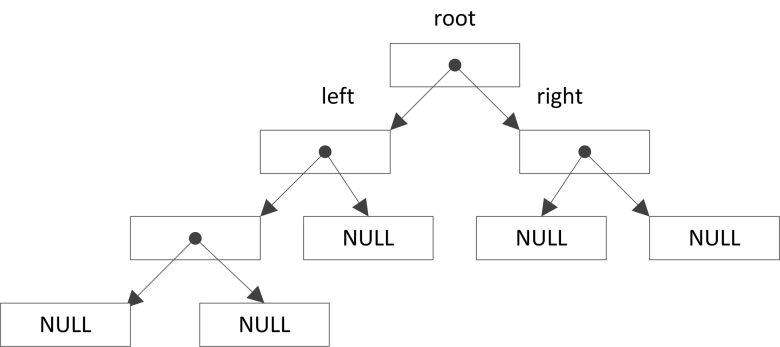
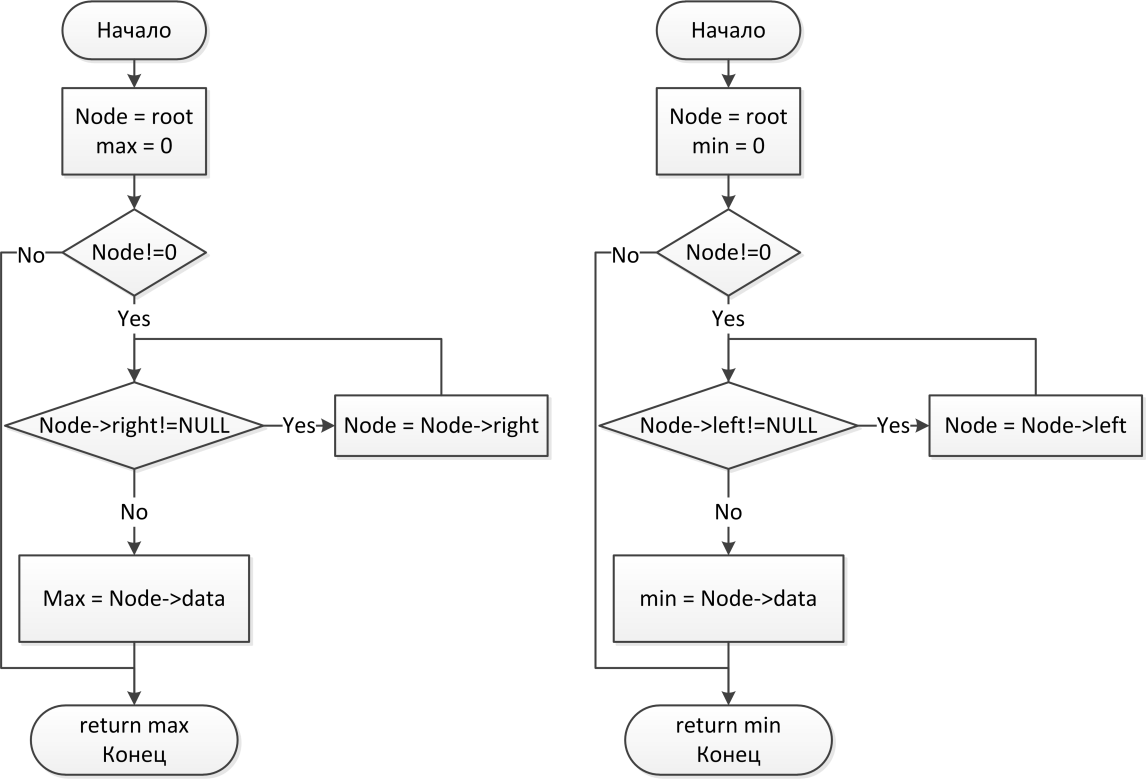


Рисунок 6.1 – Графічне зображення бінарного дерева

Через те, що бінарне дерево є впорядкованим, знаходження максимального і мінімального вузла зводиться до задачі про пошук самого правого і самого лівого вузла відповідно. Алгоритми пошуку мінімуму і максимуму наведені на рис. 6.2.



а б

Рисунок 6.2 – Блок схема алгоритму пошуку:

а – максимального вузла; б – мінімального вузла

Рішення задачі про знаходження суми елементів дерева доцільно вирішувати за допомогою рекурсивних алгоритмів через те, що дерево є рекурсивної структурою. На рис. 6.3 приведена блок схема алгоритму знаходження суми елементів дерева.

|  |  |
| --- | --- |
|  |  |
| Рисунок 6.3 — Блок-схема алгоритму знаходження суми елементів дерева | Рисунок 6.4 — Приклад роботи програми |

**Приклад.** Написати програму, яка реалізує бінарне дерево для зберігання цілих чисел. Користувач може поміщати випадкове число в дерево. Вміст дерева зв’язується з *qtTreeWiget* таким чином, щоб користувач міг бачити актуальний стан дерева (рис. 6.4).

Лістинг 6.1 – Файл binary\_tree.h

#ifndef BINARY\_TREE\_H

#define BINARY\_TREE\_H

#include <QTreeWidget>

struct Elem

**{**

int data**;**

Elem **\***left**,** **\***right**;**

**};**

class binary\_tree

**{**

Elem **\*** root**;** // корінь

QTreeWidget**\*** tv**;**

void PrintTree**(**Elem **\*** Node**,** QTreeWidgetItem**\*** trNode**);**

void AddElem**(**Elem**\*\*** node**,** Elem **\***data**);**

public**:**

binary\_tree**();** // Конструктор

void Print**();** // друк дерева

void Insert**(**int data**);** // вставка елементу

void SetQTree**(**QTreeWidget **\***tv**);** // Установка посилання на QTreeWidget

**};**

#endif // BINARY\_TREE\_H

Лістинг 6.2 – Файл binary\_tree.cpp

#include "binary\_tree.h"

#include <QTreeView>

binary\_tree**::**binary\_tree**()**

**{**

root **=** **nullptr;**

**}**

void binary\_tree**::**Print**()**

**{**

**if** **(**tv **==** **nullptr)**

**return;**

tv**->**clear**();** // Очищумо TreeWidget

QTreeWidgetItem **\***treeItem **=** **new** QTreeWidgetItem**(**tv**);**

treeItem**->**setText**(**0**,**"root"**);**

PrintTree**(**root**,** treeItem**);** // Викликаємо рекурсивний друк

tv**->**expandAll**();**

**}**

void binary\_tree**::**PrintTree**(**Elem **\*** Node**,** QTreeWidgetItem**\*** trNode**)**

**{**

**if** **(**Node **==** **nullptr)**

**return;**

QTreeWidgetItem**\*** trNode1 **=** **new** QTreeWidgetItem**(**trNode**);**

trNode1**->**setText**(**0**,** QString**::**number**(**Node**->**data**));**

trNode**->**addChild**(**trNode1**);**

PrintTree**(**Node**->**left**,** trNode1**);**

PrintTree**(**Node**->**right**,** trNode1**);**

**}**

void binary\_tree**::**AddElem**(**Elem**\*\*** node**,** Elem **\***data**)**

**{**

**if(\***node **==** **nullptr)** // Якщо лист -- вставляємо елемент

**\***node **=** data**;**

**else**

**{**

**if** **((\***node**)->**data **>** data**->**data**)** // В яку гілку?

AddElem**(&((\***node**)->**left**),** data**);**

**else**

AddElem**(&((\***node**)->**right**),** data**);**

**}**

**}**

void binary\_tree**::**Insert**(**int data**)**

**{**

Elem \* z = new Elem;

z->left = nullptr;

z->right = nullptr;

z->data = data;

AddElem(&root, z); // викликаємо метод вствки елементу

}

void binary\_tree::SetQTree(QTreeWidget \*tv)

{

this->tv = tv;

}

Лістинг 6.3 – Файл mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "binary\_tree.h"

binary\_tree tree;

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

// налаштування

ui->treeWidget->setColumnCount(1);

ui->treeWidget->setHeaderHidden(true);

tree.SetQTree(ui->treeWidget);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

tree.Insert(rand()%100);

tree.Print();

}

**Лабораторна робота №6 (3-й семестр)**

Написати програму, в якій

* 1. Реалізовано клас, що описує бінарне дерево для зберігання дійсних чисел;
  2. Користувач має можливість додавати **випадкове дійсне число** в дерево;
  3. Вміст дерева зв'язується з деяким компонентом таким чином, щоб користувач міг бачити його актуальний стан. Користувачеві також виводиться результат виконання п. 4, 5;
  4. У класі реалізований метод для обчислення суми додатних елементів дерева;
  5. У класі реалізовані методи для знаходження максимального і мінімального елемента дерева.

Написати програму, в якій

* 1. Реалізовано клас, що описує бінарне дерево для зберігання дійсних чисел;
  2. Користувач має можливість додавати **випадкове дійсне число** в дерево;
  3. Вміст дерева зв'язується з деяким компонентом таким чином, щоб користувач міг бачити його актуальний стан. Користувачеві також виводиться результат виконання п. 4, 5;
  4. У класі реалізований метод для обчислення суми від’ємних елементів дерева;
  5. У класі реалізовані методи для знаходження максимального і мінімального елемента дерева.

Написати програму, в якій

* 1. Реалізовано клас, що описує бінарне дерево для зберігання дійсних чисел;
  2. Користувач має можливість додавати **випадкове дійсне число** в дерево;
  3. Вміст дерева зв'язується з деяким компонентом таким чином, щоб користувач міг бачити його актуальний стан. Користувачеві також виводиться результат виконання п. 4, 5;
  4. У класі реалізований метод для обчислення середнього значення елементів дерева;
  5. У класі реалізовані методи для знаходження максимального і мінімального елемента дерева.

Написати програму, в якій

* 1. Реалізовано клас, що описує бінарне дерево для зберігання дійсних чисел;
  2. Користувач має можливість додавати **випадкове дійсне число** в дерево;
  3. Вміст дерева зв'язується з деяким компонентом таким чином, щоб користувач міг бачити його актуальний стан. Користувачеві також виводиться результат виконання п. 4, 5;
  4. У класі реалізований метод для обчислення суми квадратів елементів дерева;
  5. У класі реалізовані методи для знаходження максимального і мінімального елемента дерева.

Написати програму, в якій

* 1. Реалізовано клас, що описує бінарне дерево для зберігання дійсних чисел;
  2. Користувач має можливість додавати **випадкове дійсне число** в дерево;
  3. Вміст дерева зв'язується з деяким компонентом таким чином, щоб користувач міг бачити його актуальний стан. Користувачеві також виводиться результат виконання п. 4, 5;
  4. У класі реалізований метод для обчислення добутку елементів дерева;
  5. У класі реалізовані методи для знаходження максимального і мінімального елемента дерева.

Написати програму, в якій

* 1. Реалізовано клас, що описує бінарне дерево для зберігання дійсних чисел;
  2. Користувач має можливість **додавати випадкове дійсне число** в дерево;
  3. Вміст дерева зв'язується з деяким компонентом таким чином, щоб користувач міг бачити його актуальний стан. Користувачеві також виводиться результат виконання п. 4, 5;
  4. У класі реалізований метод для обчислення суми модулів елементів дерева;
  5. У класі реалізовані методи для знаходження максимального і мінімального елемента дерева.

Написати програму, в якій

* 1. Реалізовано клас, що описує бінарне дерево для зберігання дійсних чисел;
  2. Користувач має можливість додавати **випадкове дійсне число** в дерево;
  3. Вміст дерева зв'язується з деяким компонентом таким чином, щоб користувач міг бачити його актуальний стан. Користувачеві також виводиться результат виконання п. 4, 5;
  4. У класі реалізований метод для обчислення суми обернених (1/data) елементів дерева;
  5. У класі реалізовані методи для знаходження максимального і мінімального елемента дерева.

Написати програму, в якій

* 1. Реалізовано клас, що описує бінарне дерево для зберігання дійсних чисел;
  2. Користувач має можливість додавати **випадкове дійсне число** в дерево;
  3. Вміст дерева зв'язується з деяким компонентом таким чином, щоб користувач міг бачити його актуальний стан. Користувачеві також виводиться результат виконання п. 4, 5;
  4. У класі реалізований метод для обчислення суми синусів від елементів дерева;
  5. У класі реалізовані методи для знаходження максимального і мінімального елемента дерева.

Написати програму, в якій

* 1. Реалізовано клас, що описує бінарне дерево для зберігання дійсних чисел;
  2. Користувач має можливість **додавати випадкове дійсне число** в дерево;
  3. Вміст дерева зв'язується з деяким компонентом таким чином, щоб користувач міг бачити його актуальний стан. Користувачеві також виводиться результат виконання п. 4, 5;
  4. У класі реалізований метод для обчислення суми косинусів від елементів дерева;
  5. У класі реалізовані методи для знаходження максимального і мінімального елемента дерева.