Изображение выглядит как символ, эмблема, герб, нашивка

Контент, сгенерированный ИИ, может содержать ошибки.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Отчет по выполнению практического задания №7.1

**Тема: «**Балансировка дерева поиска»

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Жижикин Л.С.

группа ИКБО-50-24

**Москва 2025**

Содержание

[Цель 3](#_Toc210661235)

[Формулировка задачи 3](#_Toc210661236)

[Ход работы 3](#_Toc210661237)

[Код работы 3](#_Toc210661238)

[Файл Tree.hpp 3](#_Toc210661239)

[Структура узла 3](#_Toc210661240)

[Структура дерева 3](#_Toc210661241)

[Вставка узла 3](#_Toc210661242)

[Симметричный обход 4](#_Toc210661243)

[Сумма листьев 4](#_Toc210661244)

[Высота дерева 4](#_Toc210661245)

[Вывод в консоль 5](#_Toc210661246)

[Файл main.cpp 5](#_Toc210661247)

[Вспомогательные функции 5](#_Toc210661248)

[Консольное приложение 5](#_Toc210661249)

[Тестирование 6](#_Toc210661250)

[Минимальное дерево 6](#_Toc210661251)

[10 элементов 7](#_Toc210661252)

[Большое дерево 8](#_Toc210661253)

[Вывод 8](#_Toc210661254)

[Список литературы 9](#_Toc210661255)

# Цель

Освоить приёмы реализации алгоритмов поиска образца в тексте.

# Формулировка задачи

Составить программу создания двоичного дерева поиска и реализовать процедуры для работы с деревом согласно варианту.

# Ход работы

## Код работы

### Файл Tree.hpp

#### Структура узла

1. **struct** Node {

2.     **char** **value**;

3.     Node\* left = **nullptr**;

4.     Node\* right = **nullptr**;

5.     Node() {};

6.     Node(**char** **value**) {  **this**->**value** = **value**;  }

7.     **bool** is\_leaf() {    **return** (left == **nullptr** && right == **nullptr**);   }

8. };

#### Структура дерева

1. **struct** Tree {

2.   **protected**:

3.     Node\* root = **nullptr**;

4.     Node\* createNode(**char** **value**);

5.     Node\* insert\_private(Node\* cur, **char** **value**);

6.     **void** traverse\_private(Node\* cur);

7.     ui64 sum\_of\_leaves\_private(Node\* cur);

8.     ui64 height\_private(Node\* cur);

9.     **void** print\_private(**const** std::**string**& prefix, **const** Node\* node, **bool** isLeft);

10.

11.   **public**:

12.     Tree();

13.     Tree(**char** **value**);

14.     Tree(**string** values);

15.     Node\* insert(Node\* cur, **char** **value**);

16.     **void** traverse(Node\* cur);

17.     ui64 sum\_of\_leaves(Node\* cur);

18.     ui64 height(Node\* cur);

19.     **void** **print**(**const** std::**string**& prefix, **const** Node\* node, **bool** isLeft);

20. };

#### Вставка узла

1. Node\* createNode(**char** **value**) {

2.     Node\* newNode = **new** Node();

3.     newNode->**value** = **value**;

4.     newNode->left = newNode->right = **nullptr**;

5.     **return** newNode;

6. }

7.

8. Node\* insert\_private(Node\* cur, **char** **value**) {

9.     **if** (cur == **nullptr**)

10.         **return** createNode(**value**);

11.     **if** (**value** < cur->**value**)

12.         cur->left = insert\_private(cur->left, **value**);

13.     **else** **if** (**value** > cur->**value**)

14.         cur->right = insert\_private(cur->right, **value**);

15.     **return** cur;

16. }

17. **void** insert(**char** **value**) {

18.     root = insert\_private(root, **value**);

19. }

#### Симметричный обход

1. **void** traverse\_private(Node\* cur) {

2.     **if** (cur != **nullptr**) {

3.         traverse\_private(cur->left);

4.         **if** (cur->**value** == '\t')

5.             std::cout << "\\t";

6.         **else** **if** (cur->**value** == 0)

7.             std::cout << "\\0";

8.         **else**

9.             std::cout << cur->**value**;

10.         std::cout << "(" << (**int**)cur->**value** << ") ";

11.         traverse\_private(cur->right);

12.     }

13. }

14. **void** traverse() {

15.     traverse\_private(root);

16.     std::cout << std::endl;

17. }

#### Сумма листьев

1. ui64 sum\_of\_leaves\_private(Node\* cur) {

2.     **if** (cur == **nullptr**)

3.         **return** 0;

4.

5.     ui64 s = cur->**value**;

6.     //  if cur isnt leaf, we need to add sum from his children

7.     **if** (!cur->is\_leaf())

8.         s += sum\_of\_leaves\_private(cur->left) + sum\_of\_leaves\_private(cur->right);

9.     **return** s;

10. }

11. ui64 sum\_of\_leaves() {

12.     **return** sum\_of\_leaves\_private(root);

13. }

#### Высота дерева

1. ui64 height\_private(Node\* cur) {

2.     **if** (cur == **nullptr**)

3.         **return** 0;

4.     **int** dl = height\_private(cur->left), dr = height\_private(cur->right);

5.     **return** (1 + (dl > dr ? dl : dr));

6. }

7. **int** height() {

8.     **return** height\_private(root);

9. }

#### Вывод в консоль

1. **void** print\_private(**const** std::**string**& prefix, **const** Node\* node, **bool** isLeft) {

2.     **if**( node != **nullptr** ) {

3.         std::cout << prefix << (isLeft ? "├──" : "└──" );

4.         **if** (node->**value** == '\t')

5.             std::cout << "\\t";

6.         **else** **if** (node->**value** == 0)

7.             std::cout << "\\0";

8.         **else**

9.             std::cout << node->**value**;

10.

11.         std::cout << "("<< (**int**)node->**value** << ")" << std::endl;

12.         print\_private( prefix + (isLeft ? "│     " : "      "), node->left, **true**);

13.         print\_private( prefix + (isLeft ? "│     " : "      "), node->right, **false**);

14.     }

15. }

16. **void** **print**() {

17.     print\_private("", root, **false**);

18.     std::cout << std::endl;

19. }

### Файл main.cpp

#### Вспомогательные функции

1. **char** get\_char() {

2.     **string** \_;

3.     std::getline(std::cin, \_);

4.     **return** \_[0];

5. }

6. **char** context\_menu() {

7.     std::cout

8.             << "\nChoose what to do:\n"

9.             << "0. Exit\t\t"    <<  "3. Print\n"

10.             << "1. Insert\t"    <<  "4. Find sum of leaves\n"

11.             << "2. Traverse\t"  <<  "5. Find height\n\n";

12.     **return** get\_char();

13. }

#### Консольное приложение

1. **void** cli() {

2.     std::cout << "Enter all characters to fill the tree:\n";

3.     **string** \_;

4.     std::getline(std::cin, \_);

5.

6.     Tree tree(\_);

7.     std::cout << "Tree:\n";

8.     tree.**print**();

9.

10.     **while** (**true**) {

11.         **char** choice = context\_menu();

12.         **switch**(choice) {

13.             **case** '0':

14.                 **return**;

15.             **case** '1':

16.                 std::cout << "Enter char to insert:\t";

17.                 tree.insert(get\_char()); **break**;

18.             **case** '2':

19.                 tree.traverse(); **break**;

20.             **case** '3':

21.                 tree.**print**(); **break**;

22.             **case** '4':

23.                 std::cout << "Sum of all leaves is " << tree.sum\_of\_leaves(); **break**;

24.             **case** '5':

25.                 std::cout << "Height is " << tree.height(); **break**;

26.             **default**:

27.                 std::cout << "No such option\n";

28.         }

29.     }

30. }

## Тестирование

### Минимальное дерево

A screenshot of a computer program

AI-generated content may be incorrect.

10 элементов  
A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

### Большое дерево

A computer screen shot of a graph

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

# Вывод

Освоить приёмы реализации алгоритмов поиска образца в тексте.

# Список литературы

1. Документация по языку С++ [Электронный ресурс]. URL: https://cppreference. com/ (дата обращения: 11.09.2025).

2. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения: 11.09.2025).

3. Керниган Б., Ритчи Д. Язык программирования Си / Керниган Б., Ритчи Д., 3- е изд., 1988.

4. Страуструп Б. Программирование. Принципы и практика с использованием C+ + / Страуструп Б., 2-е изд., 2016.