Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет (НИУ)»

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ОТЧЕТ

о выполнении дополнительного практического задания №2

по дисциплине

«Структуры и алгоритмы обработки данных»

Вариант 5

Проверил:

ст. преподаватель кафедры СП

Петрова Л.Н.

Выполнил:

Студент группы КЭз-391

Галиулин Р.Р.

Челябинск
2025

Содержание

1. Описание задачи

Задание: Двоичные (бинарные) деревья

Используя рекурсивную функцию, напишите программу, которая вычисляет среднее арифметическое всех элементов непустого бинарного дерева

Входные данные

Данные вводимые пользователем отсутствуют.

- Количество узлов (листьев) бинарного дерева - определяется автоматически с помощью генератора псевдослучайный чисел на основе времени в диапазоне [2..30]
- Значения узлов (листьев) бинарного дерева - определяется автоматически с помощью генератора псевдослучайный чисел на основе времени в диапазоне [0..99]

Выходные данные

- Среднее арифметическое до удаления - вещественное число
- Изображение дерева с помощью псевдографики, до удаления
- Среднее арифметическое после удаления - вещественное число
- Изображение дерева с помощью псевдографики, после удаления

Все данные выводятся с помощью стандартного потока вывода

При разработке программы применялась следующая логика обработки удаления элемента бинарного дерева.

При удалении узла из бинарного дерева возможны три случая:

- Удаление листового узла (у узла нет потомков)
  Просто удаляем узел, никаких дополнительных действий не требуется.
- Удаление узла с одним потомком
  Заменяем удаляемый узел его единственным потомком.
  Родитель удаляемого узла начинает ссылаться на этого потомка.
- Удаление узла с двумя потомками
  Находим наименьший узел в правом поддереве (или наибольший в левом).
  Копируем его значение в удаляемый узел.
  Удаляем этот найденный узел (он гарантированно имеет не более одного потомка, поэтому его удаление попадает под один из первых двух случаев).

Таким образом, структура дерева изменяется, но остается корректной.

2. Листинг программы

Язык программирования: C++ 14. Среда разработки: Ubuntu 24.10, gcc 14.2.0, nvim

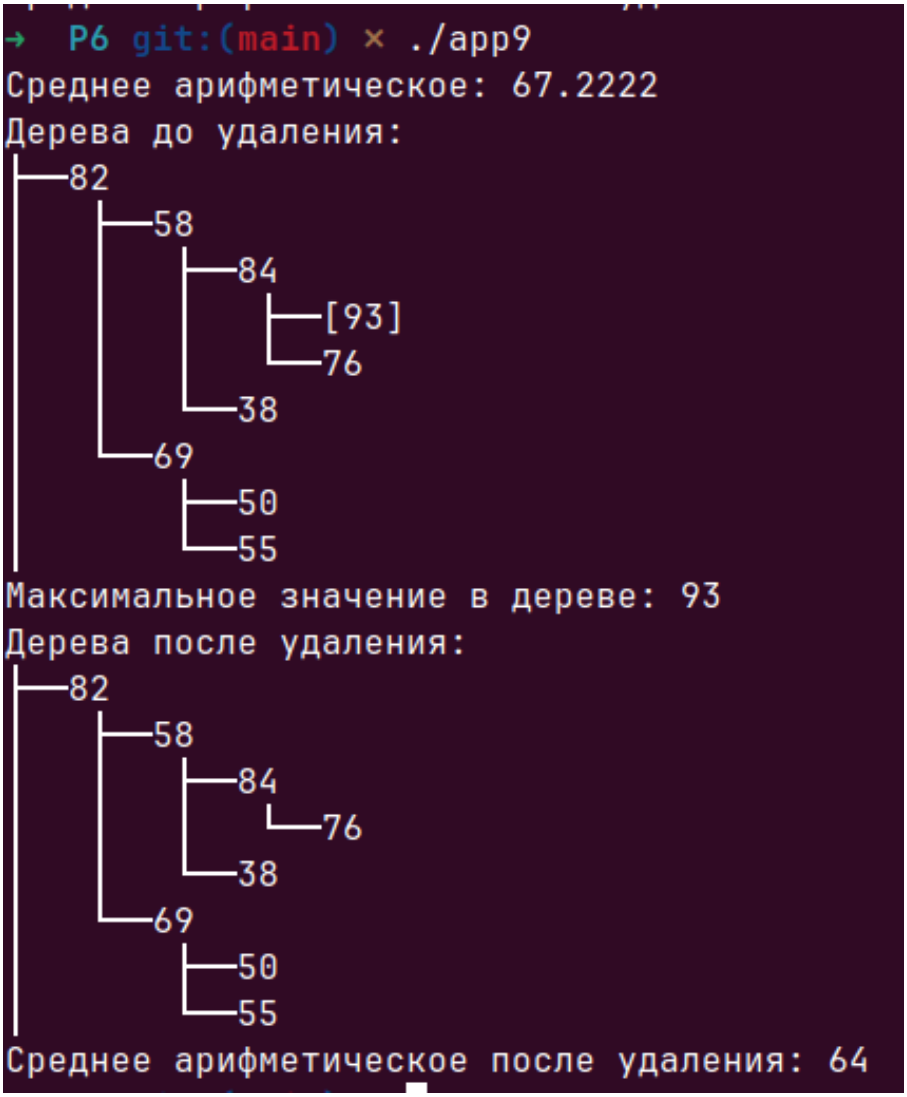
Листинг 1: Задание: Двоичные (бинарные) деревья

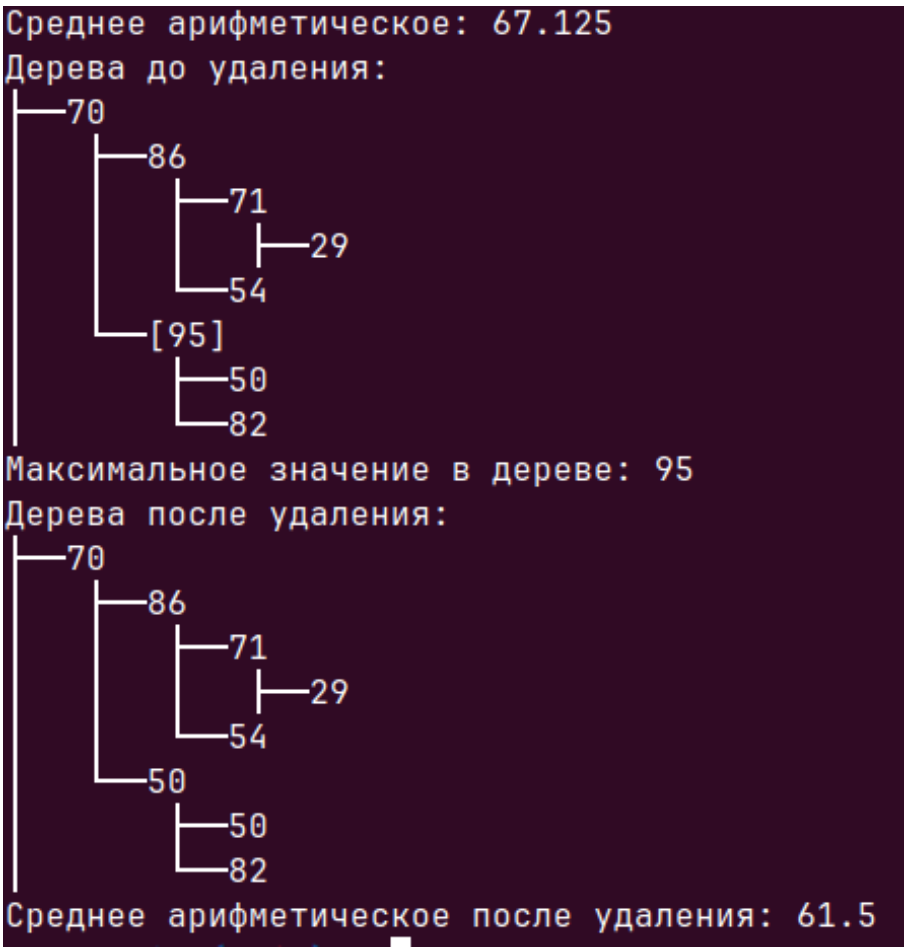
```
// Галиулин РР.. КЭз-391
// Структуры и алгоритмы обработки данных
// Дополнительное практическое занятие №1

// В созданном списке определить максимальное значение и удалить его.

#include <iostream>
#include <cstdlib>
#include <ctime>
#include <queue>
#include <algorithm>

// Определение структуры узла дерева
struct TreeNode {
    int val;
    TreeNode* left;  // Указатель на левого потомка
    TreeNode* right; // Указатель на правого потомка
    TreeNode(int x) : val(x), left(NULL), right(NULL) {}
};
// Рекурсивный подсчет суммы и количества элементов дерева
void SumAndCount(TreeNode* node, int& sum, int& count) {
    if (!node) return; // Если узел пуст
    sum += node->val;
    count++;
    SumAndCount(node->left, sum, count); //Обходим левую часть дерева
    SumAndCount(node->right, sum, count); //Обходим правую часть дерева
}

// Генерации полного бинарного дерева
TreeNode* GenerateCompleteBinaryTree(int num_nodes) {
    if (num_nodes <= 0) return NULL; // Если узлов нету
    TreeNode* root = new TreeNode(rand() % 100); //дело случая от 0 до 100
    std::queue<TreeNode*> q;
    q.push(root);
    int current_nodes = 1;  // Счетчик добавленных узлов
    while (current_nodes < num_nodes) { // До победного
        TreeNode* node = q.front();
        q.pop();
        if (current_nodes < num_nodes) { //Левый потомок
            node->left = new TreeNode(rand() % 100);
            q.push(node->left);
            current_nodes++;
        }
        if (current_nodes < num_nodes) { //Правый потомок
            node->right = new TreeNode(rand() % 100);
            q.push(node->right);
            current_nodes++;
        }
    }
    return root;
}

// Вывода дерева с ветками псевдографики
void PrintTree(TreeNode* root, int maxVal = 0, std::string prefix = "", bool
    isLeft = true) {
```

```
55      if (root != nullptr) {
56          std::cout << prefix;
57          const char* leftSymbols = "\xE2\x94\x9C\xE2\x94\x80\xE2\x94\x80";
58          const char* rightSymbols = "\xE2\x94\x94\xE2\x94\x80\xE2\x94\x80";
59          const char* vSymbols = "\xE2\x94\x82   ";
60          std::cout << (isLeft ? leftSymbols : rightSymbols);
61
62
63              if (root->val == maxVal) {
64                  std::cout << "[" << root->val << "]" << std::endl; //
    Отмечаем максимальное значение
65              } else {
66                 std::cout << root->val << std::endl;
67              }
68
69
70          PrintTree(root->left, maxVal, prefix + (isLeft ? vSymbols : "    "),
     true);
71          PrintTree(root->right, maxVal, prefix + (isLeft ? vSymbols : "    ")
    , false);
72      }
73  }
74
75  // Функция для поиска максимального значения в дереве
76  int FindMax(TreeNode* node, int currentMax) {
77      if (!node) {
78          return currentMax;
79      }
80
81      currentMax = std::max(currentMax, node->val);
82      currentMax = FindMax(node->left, currentMax);
83      currentMax = FindMax(node->right, currentMax);
84
85      return currentMax;
86  }
87
88  // Функция для удаления узла с максимальным значением. Возвращает корень
    модифицированного дерева
89  TreeNode* DeleteMax(TreeNode* root, int maxVal, bool& removed) {
90       if (!root) {
91          return nullptr; //Если пусто
92      }
93
94      if (root->val == maxVal && !removed) {
95          removed = true;
96          if (!root->left && !root->right) { // Случай 1: Удаление листового узла
97              delete root;
98              return nullptr;
99          } else if (!root->left) {  // Случай 2: Удаление узла с правым потомком
100             TreeNode* temp = root->right;
101             delete root;
102             return temp;
103         } else if (!root->right) { // Случай 2: Удаление узла с левым потомком
104             TreeNode* temp = root->left;
105             delete root;
106             return temp;
107         } else {  // Случай 3: Удаление узла с двумя потомками
108              TreeNode* temp = root->left;
109             while(temp->right){
110                 temp = temp->right;
```

```
            }
             root->val = temp->val;
            root->left = DeleteMax(root->left, temp->val, removed);
            return root;
        }
    }
    root->left = DeleteMax(root->left, maxVal, removed);
    root->right = DeleteMax(root->right, maxVal, removed);
    return root;
}

int main() {
    srand(time(0));

    int num_nodes = rand() % 29 + 2;

    // Создание полного бинарного дерева
    TreeNode* root = GenerateCompleteBinaryTree(num_nodes);

    // Подсчет суммы и количества элементов
    int sum = 0;
    int count = 0;
    SumAndCount(root, sum, count);

    double average = (double)sum / count;
    std::cout << "Среднее арифметическое: " << average << std::endl;

    int maxVal = FindMax(root, 0); // Ищем максимальное значение


    // Вывод дерева на экран
    std::cout << "Дерева до удаления:" << std::endl;
    PrintTree(root, maxVal);

    maxVal = FindMax(root, 0); // Ищем максимальное значение

    std::cout << "Максимальное значение в дереве: " << maxVal << std::endl;

    // Удаляем максимальное значение
    bool removed = false;
    root = DeleteMax(root, maxVal, removed);

    std::cout << "Дерева после удаления:" << std::endl;
    PrintTree(root);

    // Подсчет суммы и количества элементов после удаления
    sum = 0;
    count = 0;
    SumAndCount(root, sum, count);

    average = (double)sum / count;
    std::cout << "Среднее арифметическое после удаления: " << average << std::endl
    ;

    return 0;
}
```

3. Контрольные тесты





3.1 Задание №1: Двух связанный список

Исходные данные	**Результат**
10 43 45 86 11 94 29 75 99 41 12	Исходный список: 43 45 86 11 94 29 75 [99] 41 12 Список после удаления максимального элемента: 43 45 86 11 94 29 75 41 12
5 5 5 9 2 10	Исходный список: 5 5 9 2 [10] Список после удаления максимального элемента: 5 5 9 2

Таблица 1: Таблица с результатами контрольных тестов Задания №1