Вариант 21. Шерхан Тимур. Б24-3ВВС2.

Выполнение работы заключается в разработке программы с использованием консольного или графического пользовательского интерфейса на усмотрение студента. Создание бинарного дерева и работа с ним должны осуществляться с использованием указателей; применение объектных типов данных типа TTree, встроенных библиотечных компонентов типа TreeView, JTree, TreeNode и иных средств, применяемых в различных современных языках программирования, не допускается. При выборе языка программирования основным требованием является поддержка работы с указателями. Разработанная программа должна выполнять последовательно две задачи.

Дано число N (>0) и произвольный набор из N чисел. Сформировать начальную структуру числовых данных в виде двоичного дерева заданного вида. Вывести указатель на корень созданного дерева.

Создать дерево из N вершин, в котором каждая левая дочерняя вершина является листом, а правая дочерняя вершина является внутренней. Для каждой внутренней вершины вначале создавать левую дочернюю вершину, а затем – правую (если она существует); каждой создаваемой вершине присваивать очередное значение из исходного набора

main.cpp

```
#include "my_tree.h"
#include <iostream>
#include <vector>
#include <windows.h>

#define GREEN   "\033[1;32m"
#define RED     "\033[1;31m"
#define RESET   "\033[0m"

int main() {

    SetConsoleOutputCP(CP_UTF8);
    SetConsoleCP(CP_UTF8);

    Node* root = nullptr;
    std::vector<int> values;
    int choice;

    do {
```

```
        std::cout << "\nМеню:\n"
                  << "1. Ввести N и значения\n"
                  << "2. Построить дерево\n"
                  << "3. Показать дерево\n"
                  << "4. Показать указатель на корень\n"
                  << "0. Выход\n"
                  << "Выберите пункт: ";
        std::cin >> choice;

        switch (choice) {
            case 1: {
                int n, val;
                std::cout << "Введите количество вершин (N): ";
                std::cin >> n;
                values.clear();
                for (int i = 0; i < n; ++i) {
                    std::cout << "Введите значение для вершины " << i + 1 << ":
";
                    std::cin >> val;
                    values.push_back(val);
                }
                break;
            }

            case 2:
                if (values.empty()) {
                    std::cout << RED << "Ошибка. Вы не ввели данные." << RESET <<
std::endl;
                } else {
                    if (root) {
                        deleteTree(root); // Удаляем старое дерево, если было.
                        root = nullptr;
                    }
                    root = buildTree(values.data(), values.size());
                    std::cout << GREEN << "Дерево построено." << RESET <<
std::endl;
                }
                break;

            case 3:
                if (root) {
                    printTree(root);
                } else {
                    std::cout << RED << "Ошибка. Вы не построили дерево." <<
RESET << std::endl;
```

```
                }
                break;

            case 4:
                std::cout << "Указатель на корень: " << root << std::endl;
                break;

            case 0:
                std::cout << "Выход.\n";
                break;

            default:
                std::cout << "Неверный выбор.\n";
        }
    } while (choice != 0);


    if (root) {
        deleteTree(root);
    }
    return 0;
}
```

tree.cpp

```
#include "my_tree.h"
#include <iostream>

Node::Node(int val) : data(val), left(nullptr), right(nullptr) {}

Node* buildTree(const int* values, int size) {
    if (size == 0) return nullptr;
    Node* root = new Node(values[0]);
    Node* current = root;
    int idx = 1;

    while (idx < size) {

        // Сначала левая дочерняя вершина - лист.
        current->left = new Node(values[idx++]);
        if (idx >= size) break;
```

```
        // Потом правая дочерняя вершина.
        current->right = new Node(values[idx++]);
        current = current->right;  // Движемся по дереву вправо.
    }

    return root;
}

void printTree(Node* root, std::string prefix, bool isLeft) {
    if (root == nullptr) return;
    std::cout << prefix;
    std::cout << (isLeft ? "├──" : "└──");
    std::cout << root->data << std::endl;
    printTree(root->left, prefix + (isLeft ? "│   " : "    "), true);
    printTree(root->right, prefix + (isLeft ? "│   " : "    "), false);
}

void deleteTree(Node* root) {
    if (!root) return;
    deleteTree(root->left);
    deleteTree(root->right);
    delete root;
}
```

my_tree.h

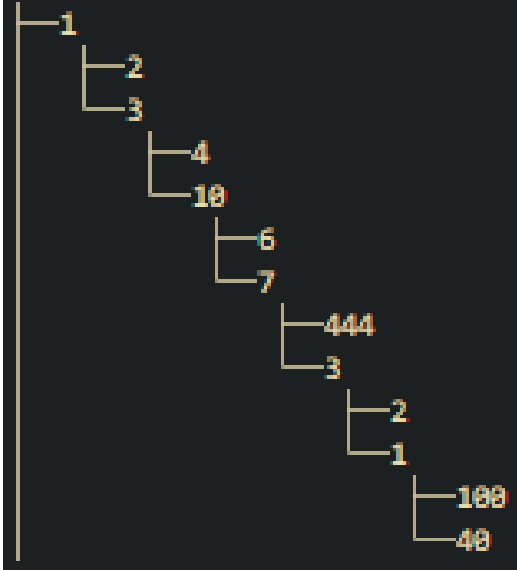
```
#ifndef TREE_H
#define TREE_H

#include <string>

struct Node {
    int data;
    Node* left;
    Node* right;
    Node(int val);
};

Node* buildTree(const int* values, int size);
void printTree(Node* root, std::string prefix = "", bool isLeft = true);
void deleteTree(Node* root);
```

```
#endif
```

Пример работы программы.

```
Меню:
1. Ввести N и значения
2. Построить дерево
3. Показать дерево
4. Показать указатель на корень
0. Выход
Выберите пункт: 3
Ошибка. Вы не построили дерево.

Меню:
1. Ввести N и значения
2. Построить дерево
3. Показать дерево
4. Показать указатель на корень
0. Выход
Выберите пункт: 2
Ошибка. Вы не ввели данные.

Меню:
1. Ввести N и значения
2. Построить дерево
3. Показать дерево
4. Показать указатель на корень
0. Выход
Выберите пункт: 4
Указатель на корень: 0

Меню:
1. Ввести N и значения
2. Построить дерево
3. Показать дерево
4. Показать указатель на корень
0. Выход
Выберите пункт: 1
Введите количество вершин (N): 13
Введите значение для вершины 1: 1
Введите значение для вершины 2: 2
Введите значение для вершины 3: 3
Введите значение для вершины 4: 4
Введите значение для вершины 5: 10
Введите значение для вершины 6: 6
Введите значение для вершины 7: 7
Введите значение для вершины 8: 444
Введите значение для вершины 9: 3
Введите значение для вершины 10: 2
Введите значение для вершины 11: 1
Введите значение для вершины 12: 100
Введите значение для вершины 13: 40

Меню:
1. Ввести N и значения
2. Построить дерево
3. Показать дерево
4. Показать указатель на корень
0. Выход
Выберите пункт: 3
Ошибка. Вы не построили дерево.

Меню:
1. Ввести N и значения
2. Построить дерево
3. Показать дерево
4. Показать указатель на корень
0. Выход
Выберите пункт: 2
Дерево построено.
```

```
Меню:
1. Ввести N и значения
2. Построить дерево
3. Показать дерево
4. Показать указатель на корень
0. Выход
Выберите пункт: 3
```




```
Меню:
1. Ввести N и значения
2. Построить дерево
3. Показать дерево
4. Показать указатель на корень
0. Выход
Выберите пункт: 4
Указатель на корень: 0x1fb63be26a0

Меню:
1. Ввести N и значения
2. Построить дерево
3. Показать дерево
4. Показать указатель на корень
0. Выход
Выберите пункт:
```

Оценка сложности алгоритмов.

Функция	Время (среднее)	Время (худшее)	Память (среднее)	Память (худшее)
buildTree	$O(n)$	$O(n)$	$O(n)$	$O(n)$
printTree	$O(n \log n)$	$O(n^2)$	$O(n)$	$O(n^2)$
deleteTree	$O(n)$	$O(n)$	$O(\log n)$	$O(n)$