**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Кафедра «САПР»

Структуры и алгоритмы обработки данных

**Отчет о лабораторной работе № 1**

по теме

«Деревья в прикладных алгоритмах»

**Выполнили:**

Студент гр. 045

Анохин В.А.

Вашкулатов Н.А.

**Проверили:**

Проф.Скворцов С.В

**Цель работы:** изучение и анализ структур данных, применяемых для описания деревьев в прикладных программах; анализ и программная реализация операторов для работы с бинарными деревьями; разработка и исследование прикладных программ, в которых применяются бинарные деревья.

**Ход работы:**

**Вариант задания:** представление деревьев в динамической памяти.

На рисунках 1 – 2 показаны схемы построения дерева. На рисунках 3 – 4 показаны схемы поиска в массиве при помощи дерева. На рисунке 5 показаны схемы способов обходов бинарных деревьев.

**Код программы:**

**Node.h**

namespace tree {  
 typedef int data\_type;  
 struct Node{  
 data\_type data;  
 int index;  
 Node\* left;  
 Node\* right;  
 Node(data\_type \_data, int \_index): data(\_data),index(\_index), left(nullptr), right(nullptr){};  
 };  
 Node\* LEFT(Node\* node);  
 Node\* RIGHT(Node\* node);  
 data\_type INFO(Node\* node);  
 int FIND(data\_type key, data\_type data[], int n);  
 Node\* MAKE\_TREE(data\_type data[], int n);  
 Node\* INSERT(Node\* tree, data\_type data, int index);  
 void TLR(Node\* tree);  
 void LTR(Node\* tree);  
 void LRT(Node\* tree);  
}

**Node.cpp**

using namespace std;  
namespace tree {  
 Node \*MAKE\_TREE(data\_type data[], int n) {  
 Node \*root = new Node(data[0], 0);  
 for (int i = 1; i < n; i++) {  
 INSERT(root, data[i], i);  
 }  
 return root;  
 }  
 Node \*INSERT(Node \*tree, data\_type data, int index) {  
 if (tree == nullptr) {  
 return new Node(data, index);  
 } else if (data < INFO(tree)) {  
 tree->left = INSERT(LEFT(tree), data, index);  
 } else {  
 tree->right = INSERT(RIGHT(tree), data, index);  
 }  
 return tree;  
 }  
 Node \*LEFT(Node \*node) {  
 return node->left;  
 }  
 Node \*RIGHT(Node \*node) {  
 return node->right;  
 }  
 data\_type INFO(Node \*node) {  
 return node->data;  
 }  
 int FIND\_IN\_TREE(data\_type key, Node \*tree) {  
 if (tree == nullptr) return -1;  
 else if (INFO(tree) == key)  
 return tree->index;  
 else if (key < INFO(tree))  
 return FIND\_IN\_TREE(key, LEFT(tree));  
 else return FIND\_IN\_TREE(key, RIGHT(tree));  
 }  
 int FIND(data\_type key, data\_type data[], int n) {  
 Node \*tree = MAKE\_TREE(data, n);  
 return FIND\_IN\_TREE(key, tree);  
 }  
 void TLR(Node \*tree) {  
 if (tree == nullptr) return;  
 cout << INFO(tree) << " ";  
 TLR(LEFT(tree));  
 TLR(RIGHT(tree));  
 }  
 void LTR(Node \*tree) {  
 if (tree == nullptr) return;  
 LTR(LEFT(tree));  
 cout << INFO(tree) << " ";  
 LTR(RIGHT(tree));  
 }  
 void LRT(Node \*tree) {  
 if (tree == nullptr) return;  
 LRT(LEFT(tree));  
 LRT(RIGHT(tree));  
 cout << INFO(tree) << " ";  
 }  
}

**main.cpp**

#include <iostream>  
#include "Node.h"  
  
using namespace tree;  
using namespace std;  
  
int main() {  
 int n = 7;  
 int data[] = {2, 5, 6, 1, -1, 7, 9};  
 int found = FIND(-1, data, n);  
 int notFound = FIND(99, data, n);  
 cout << found << endl;  
 cout << notFound << endl;  
  
 Node \*tree = MAKE\_TREE(data, n);  
 LTR(tree);  
 return 0;  
}

**Схемы алгоритмов**

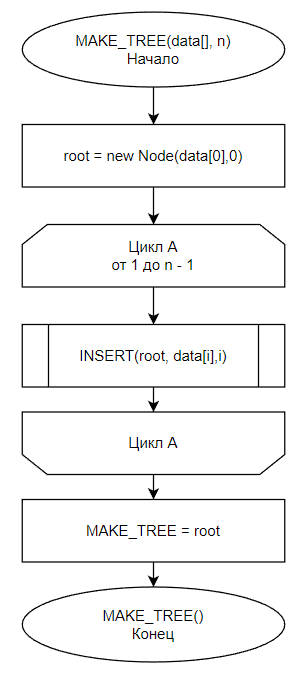
****

Рисунок 1 – Построение бинарного дерева

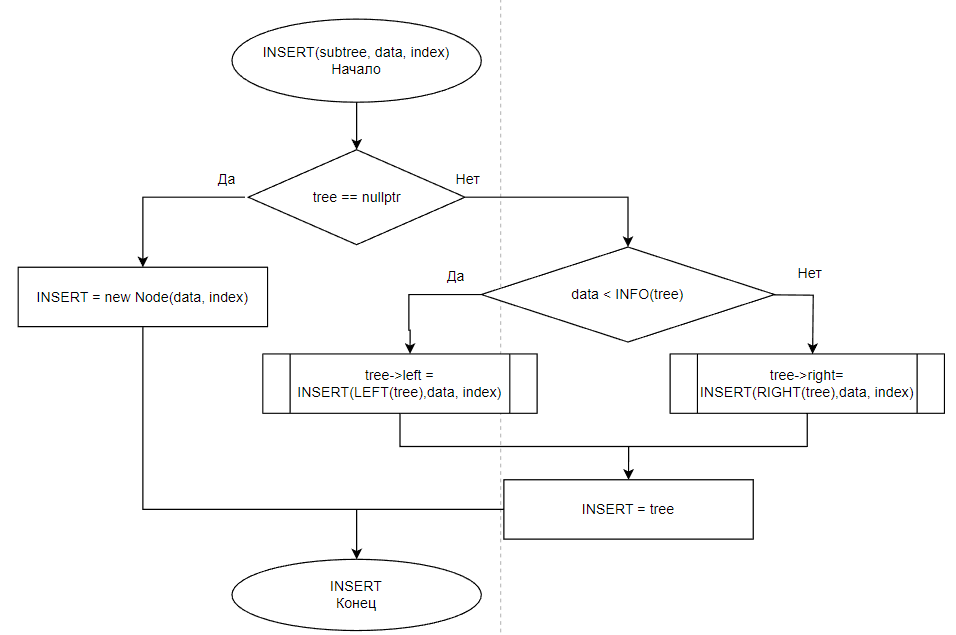
****

Рисунок 2 – Вставка элемента в дерево

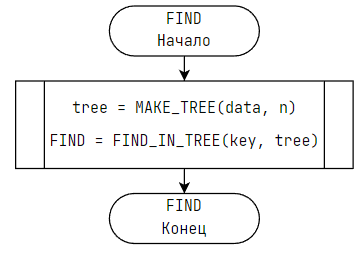


Рисунок 3 – Поиск в массиве

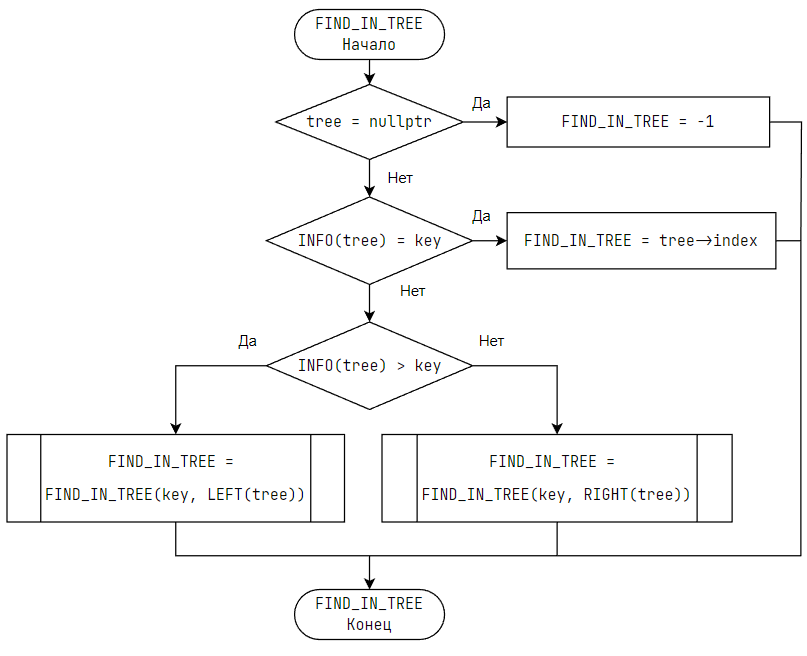


Рисунок 4 – Поиск по бинарному дереву

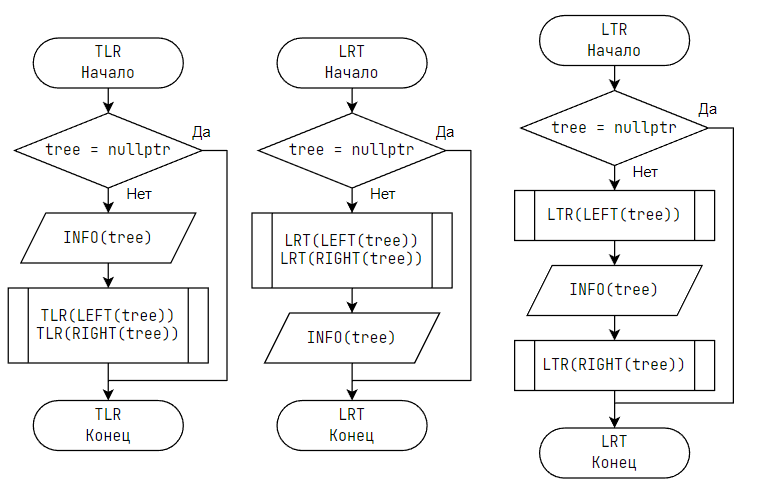


Рисунок 5 – Схемы обхода дерева

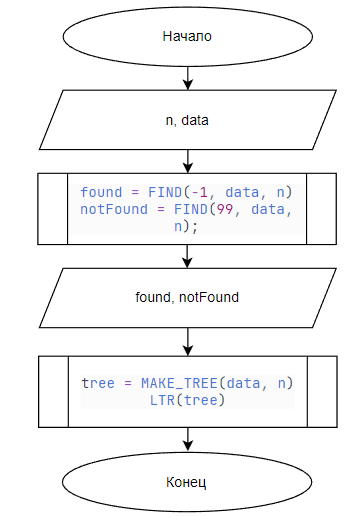


Рисунок 6 – Основная программа

**Исходный массив:** 2, 5, 7, 1, -1, 9, 6

На рисунке 7 представлено дерево, построенное из исходного массива.

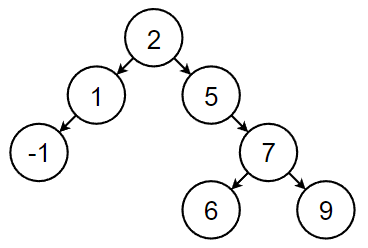


Рисунок 7 – Построенное дерево

Поиск числа -1 будет успешным. Результатом будет индекс числа в массиве: 4.

Поиск числа 99 завершится неудачей. Результат -1.

Чтобы получить отсортированный массив, нужно обойти дерево методом LTR. Результат: -1, 1, 2, 5, 6, 7, 9.

На рисунке 8 показан вывод программы.

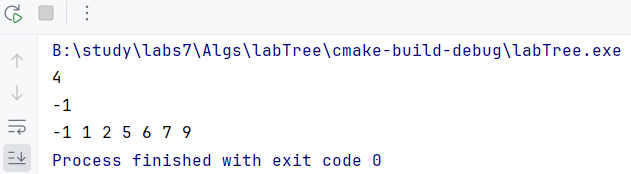


Рисунок 8 – Вывод программы

**Вывод:** мы изучили и провели анализ структур данных, применяемых для описания деревьев в прикладных программах; анализ и программная реализация операторов для работы с бинарными деревьями; разработали и исследовали прикладные программ, в которых применяются бинарные деревья.