Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу “Логика и основа алгоритмизации в ИЗ”

на тему “ Бинарное дерево поиска”



Пенза 2022

**Название**

Бинарное дерево поиска.

**Цель работы:** выполнить лабораторные указания 1-4 реализуя бинарное дерево поиска.

**Лабораторное задание:**

1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
2. Реализовать функцию подсчета числа вхождений заданного элемента в дерево.
3. Изменить функцию добавления элементов для исключения добавления одинаковых символов.
4. Оценить сложность процедуры поиска по значению в бинарном дереве.

**Задание 1:**

**Листинг**

#include <iostream>

#include <string>

using namespace std;

class Tree {

Tree\* left = nullptr;

Tree\* right = nullptr;

public:

int data;

Tree() { data = 0; };

Tree(int data) : data(data) { };

void add(int a) {

Tree\* tmp = new Tree;

tmp->data = a;

Tree\* main = this;

for (;;) {

if (main->data > tmp->data) {

if (main->left == nullptr) {

main->left = tmp;

break;

}

else main = main->left;

}

else {

if (main->right == nullptr) {

main->right = tmp;

break;

}

else main = main->right;

}

}

}

bool add\_unique(int data) {

Tree\* tmp = new Tree;

tmp->data = data;

Tree\* main = this;

for (;;) {

if (main->data == data) {

return 0;

}

if (main->data > tmp->data) {

if (main->left == nullptr) {

main->left = tmp;

break;

}

else main = main->left;

}

else {

if (main->right == nullptr) {

main->right = tmp;

break;

}

else main = main->right;

}

}

return 1;

}

string find() {

int num;

int deep = 1;

string flue = "H";

while (true) {

cout << "Введите число, которое хотите найти: ";

cin >> num;

if (!cin.good()) {

cout << endl << "Данное значение не коректно" << endl << endl;

cin.clear();

cin.ignore(100, '\n');

continue;

}

break;

}

Tree\* main = this;

for (;;) {

if (main->data == num) {

cout << "Данное значение есть в дереве" << endl;

flue += " " + std::to\_string(deep);

return flue;

}

if (main->data > num) {

main = main->left;

deep++;

flue += 'L';

}

else {

main = main->right;

deep++;

flue += 'R';

}

if (main == nullptr) {

cout << "Данного значения нет в дереве" << endl;

return " ";

}

}

}

int repeats(int data) {

int repeats = 0;

Tree\* main = this;

for (;;) {

if (main->data == data) {

repeats++;

}

if (main->data > data) main = main->left;

else main = main->right;

if (main == nullptr) {

return repeats;

}

}

}

friend void print\_tree(Tree\* r, int l);

};

void print\_tree(Tree\* r, int l)

{

if (r == NULL)

{

return;

}

print\_tree(r->right, l + 1);

for (int i = 0; i < l; i++)

{

printf(" ");

}

printf("%3d\n", r->data);

print\_tree(r->left, l + 1);

}

int main() {

setlocale(LC\_ALL, "RUS");

srand(static\_cast<int>(time(NULL)));

Tree\* a = new Tree;

cout << "Введите вершину дерева" << endl;

cin >> a->data;

int size = 0;

int count;

cout << "Введите кол-во:";

cin >> count;

size += count;

for (int i = 0; i != count; i++) {

int data;

cout << "Введите данные: ";

cin >> data;

a->add(data);

}

print\_tree(a, count);

cout << "Введите кол-во неповторяющихся данных: ";

cin >> count;

for (int i = 0; i != count; i++) {

int data;

cout << "Введите данные: ";

cin >> data;

size += a->add\_unique(data);

}

print\_tree(a, size);

cout << endl << endl << endl;

cout << "Проверка функции поиска элемента" << endl;

string search = " ";

while (search != "NO") {

cout << endl << "Продолжить поиск?" << endl << "ДА" << " " << "НЕТ" << endl;

cin >> search;

cout << endl;

if (search == "NO") break;

cout << a->find() << endl;

}

search = " ";

while (search != "NO") {

cout << endl << "Поиск повторов?" << endl << "число" << " или " << "НЕТ" << endl;

cin >> search;

cout << endl;

if (search == "NO") break;

cout << a->repeats(stoi(search)) << endl;

}

return 0;

}

**Результаты работы программы:**

Результаты работы программы показан на рисунке 1.

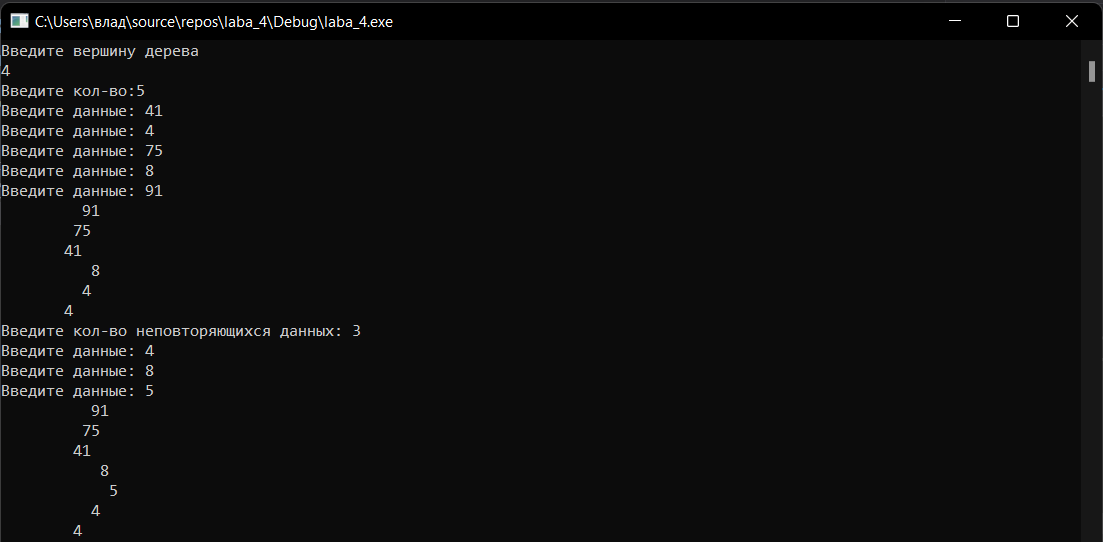
****

Рис. 1

Реализован алгоритм введения элементов с клавиатуры в уже готовое дерево.

**Задание 2:**

**Листинг**

#include <iostream>

#include <string>

using namespace std;

class Tree {

Tree\* left = nullptr;

Tree\* right = nullptr;

public:

int data;

Tree() { data = 0; };

Tree(int data) : data(data) { };

void add(int a) {

Tree\* tmp = new Tree;

tmp->data = a;

Tree\* main = this;

for (;;) {

if (main->data > tmp->data) {

if (main->left == nullptr) {

main->left = tmp;

break;

}

else main = main->left;

}

else {

if (main->right == nullptr) {

main->right = tmp;

break;

}

else main = main->right;

}

}

}

bool add\_unique(int data) {

Tree\* tmp = new Tree;

tmp->data = data;

Tree\* main = this;

for (;;) {

if (main->data == data) {

return 0;

}

if (main->data > tmp->data) {

if (main->left == nullptr) {

main->left = tmp;

break;

}

else main = main->left;

}

else {

if (main->right == nullptr) {

main->right = tmp;

break;

}

else main = main->right;

}

}

return 1;

}

string find() {

int num;

int deep = 1;

string flue = "H";

while (true) {

cout << "Введите число, которое хотите найти: ";

cin >> num;

if (!cin.good()) {

cout << endl << "Данное значение не коректно" << endl << endl;

cin.clear();

cin.ignore(100, '\n');

continue;

}

break;

}

Tree\* main = this;

for (;;) {

if (main->data == num) {

cout << "Данное значение есть в дереве" << endl;

flue += " " + std::to\_string(deep);

return flue;

}

if (main->data > num) {

main = main->left;

deep++;

flue += 'L';

}

else {

main = main->right;

deep++;

flue += 'R';

}

if (main == nullptr) {

cout << "Данного значения нет в дереве" << endl;

return " ";

}

}

}

int repeats(int data) {

int repeats = 0;

Tree\* main = this;

for (;;) {

if (main->data == data) {

repeats++;

}

if (main->data > data) main = main->left;

else main = main->right;

if (main == nullptr) {

return repeats;

}

}

}

friend void print\_tree(Tree\* r, int l);

};

void print\_tree(Tree\* r, int l)

{

if (r == NULL)

{

return;

}

print\_tree(r->right, l + 1);

for (int i = 0; i < l; i++)

{

printf(" ");

}

printf("%3d\n", r->data);

print\_tree(r->left, l + 1);

}

int main() {

setlocale(LC\_ALL, "RUS");

srand(static\_cast<int>(time(NULL)));

Tree\* a = new Tree;

cout << "Введите вершину дерева" << endl;

cin >> a->data;

int size = 0;

int count;

cout << "Введите кол-во:";

cin >> count;

size += count;

for (int i = 0; i != count; i++) {

int data;

cout << "Введите данные: ";

cin >> data;

a->add(data);

}

print\_tree(a, count);

cout << "Введите кол-во неповторяющихся данных: ";

cin >> count;

for (int i = 0; i != count; i++) {

int data;

cout << "Введите данные: ";

cin >> data;

size += a->add\_unique(data);

}

print\_tree(a, size);

cout << endl << endl << endl;

cout << "Проверка функции поиска элемента" << endl;

string search = " ";

while (search != "NO") {

cout << endl << "Продолжить поиск?" << endl << "ДА" << " " << "НЕТ" << endl;

cin >> search;

cout << endl;

if (search == "NO") break;

cout << a->find() << endl;

}

search = " ";

while (search != "NO") {

cout << endl << "Поиск повторов?" << endl << "число" << " или " << "НЕТ" << endl;

cin >> search;

cout << endl;

if (search == "NO") break;

cout << a->repeats(stoi(search)) << endl;

}

return 0;

}

**Результаты работы программы:**

Результаты работы программы показан на рисунке 1.

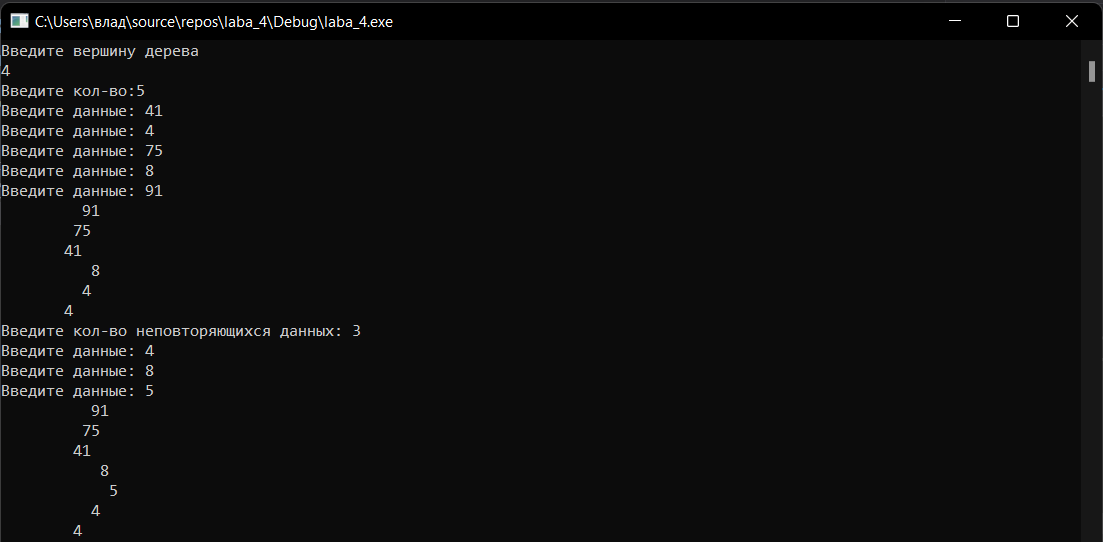
****

Рис. 1

Результат получился правильный, все действия совершены верно.

Реализована функция подсчета числа вхождений заданного элемента в дерево.

**Задание 3:**

**Листинг**

#include <iostream>

#include <string>

using namespace std;

class Tree {

Tree\* left = nullptr;

Tree\* right = nullptr;

public:

int data;

Tree() { data = 0; };

Tree(int data) : data(data) { };

void add(int a) {

Tree\* tmp = new Tree;

tmp->data = a;

Tree\* main = this;

for (;;) {

if (main->data > tmp->data) {

if (main->left == nullptr) {

main->left = tmp;

break;

}

else main = main->left;

}

else {

if (main->right == nullptr) {

main->right = tmp;

break;

}

else main = main->right;

}

}

}

bool add\_unique(int data) {

Tree\* tmp = new Tree;

tmp->data = data;

Tree\* main = this;

for (;;) {

if (main->data == data) {

return 0;

}

if (main->data > tmp->data) {

if (main->left == nullptr) {

main->left = tmp;

break;

}

else main = main->left;

}

else {

if (main->right == nullptr) {

main->right = tmp;

break;

}

else main = main->right;

}

}

return 1;

}

string find() {

int num;

int deep = 1;

string flue = "H";

while (true) {

cout << "Введите число, которое хотите найти: ";

cin >> num;

if (!cin.good()) {

cout << endl << "Данное значение не коректно" << endl << endl;

cin.clear();

cin.ignore(100, '\n');

continue;

}

break;

}

Tree\* main = this;

for (;;) {

if (main->data == num) {

cout << "Данное значение есть в дереве" << endl;

flue += " " + std::to\_string(deep);

return flue;

}

if (main->data > num) {

main = main->left;

deep++;

flue += 'L';

}

else {

main = main->right;

deep++;

flue += 'R';

}

if (main == nullptr) {

cout << "Данного значения нет в дереве" << endl;

return " ";

}

}

}

int repeats(int data) {

int repeats = 0;

Tree\* main = this;

for (;;) {

if (main->data == data) {

repeats++;

}

if (main->data > data) main = main->left;

else main = main->right;

if (main == nullptr) {

return repeats;

}

}

}

friend void print\_tree(Tree\* r, int l);

};

void print\_tree(Tree\* r, int l)

{

if (r == NULL)

{

return;

}

print\_tree(r->right, l + 1);

for (int i = 0; i < l; i++)

{

printf(" ");

}

printf("%3d\n", r->data);

print\_tree(r->left, l + 1);

}

int main() {

setlocale(LC\_ALL, "RUS");

srand(static\_cast<int>(time(NULL)));

Tree\* a = new Tree;

cout << "Введите вершину дерева" << endl;

cin >> a->data;

int size = 0;

int count;

cout << "Введите кол-во:";

cin >> count;

size += count;

for (int i = 0; i != count; i++) {

int data;

cout << "Введите данные: ";

cin >> data;

a->add(data);

}

print\_tree(a, count);

cout << "Введите кол-во неповторяющихся данных: ";

cin >> count;

for (int i = 0; i != count; i++) {

int data;

cout << "Введите данные: ";

cin >> data;

size += a->add\_unique(data);

}

print\_tree(a, size);

cout << endl << endl << endl;

cout << "Проверка функции поиска элемента" << endl;

string search = " ";

while (search != "NO") {

cout << endl << "Продолжить поиск?" << endl << "ДА" << " " << "НЕТ" << endl;

cin >> search;

cout << endl;

if (search == "NO") break;

cout << a->find() << endl;

}

search = " ";

while (search != "NO") {

cout << endl << "Поиск повторов?" << endl << "число" << " или " << "НЕТ" << endl;

cin >> search;

cout << endl;

if (search == "NO") break;

cout << a->repeats(stoi(search)) << endl;

}

return 0;

}

**Результаты работы программы:**

Результаты работы программы показан на рисунке 1.

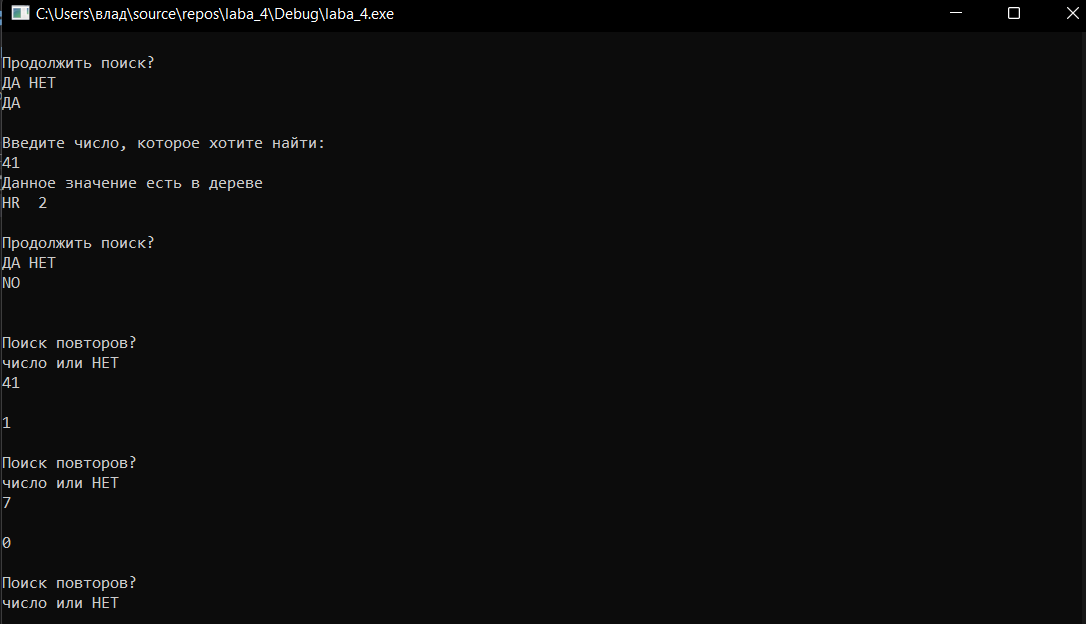
****

Рис. 1

Результат получился правильный, все действия совершены верно.

Изменена функция добавления элементов для исключения добавления одинаковых символов.

**Задание 4:**

Оценив процедуры поиска по значению в бинарном дереве, мы рассчитали сложность, которая ровна n\*Log n.

**Вывод:** Выполнив лабораторные задания 1-4 мы научились реализовывать структуру данных, бинарное дерево и реализовывать функцию работы с деревом.