Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра вычислительной техники

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 22

Двоичное дерево поиска
Вариант № 12

Преподаватель _____ Пушкарев К. В.
подпись, дата

Студент КИ18-09б, 031830645 _____ 18.04.2019 Котов С.А.
подпись

Красноярск 2019

1 Цель работы

Получить практические навыки обработки нелинейных динамических структур данных.

2 Назначение функции (Упражнение № 1 вариант 6)

Данная функция предназначена вывода элементов, находящихся на n-ом уровне.

6.	`void treeprint_level (node *p, int n) {`
	`  if (p!= NULL) {`
	`  treeprint_level (p->ll, n-1);`
	`  if (n ==1) printf("\t%d\n",  p->info);`
	`  treeprint_level (p->rl, n-1);`
	`  }`
	`}`

3 Комментарии к функции

Параметр n содержит номер уровня дерева, который необходимо вывести на экран.

Изначально программа спускается по левой ветке дерева до уровня, указанного пользователем. Спустившись на указанный уровень программа выводит значение, находящееся на этом месте, после чего начинает спускать по правым веткам дерева и повторяет проход по левым веткам, если они существуют.

4 Графическая схема алгоритма

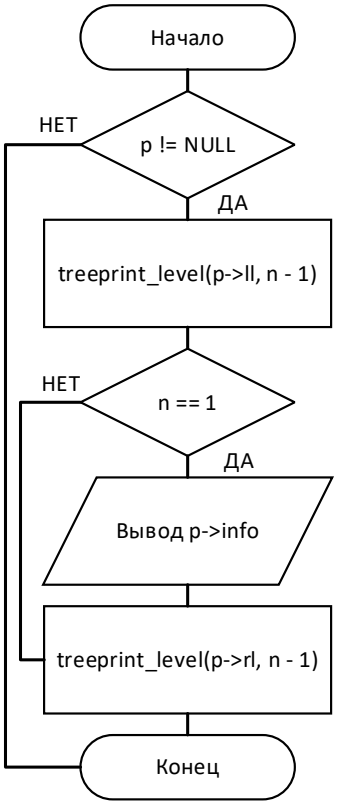


Рисунок 1 – Блок-схема алгоритма функции.

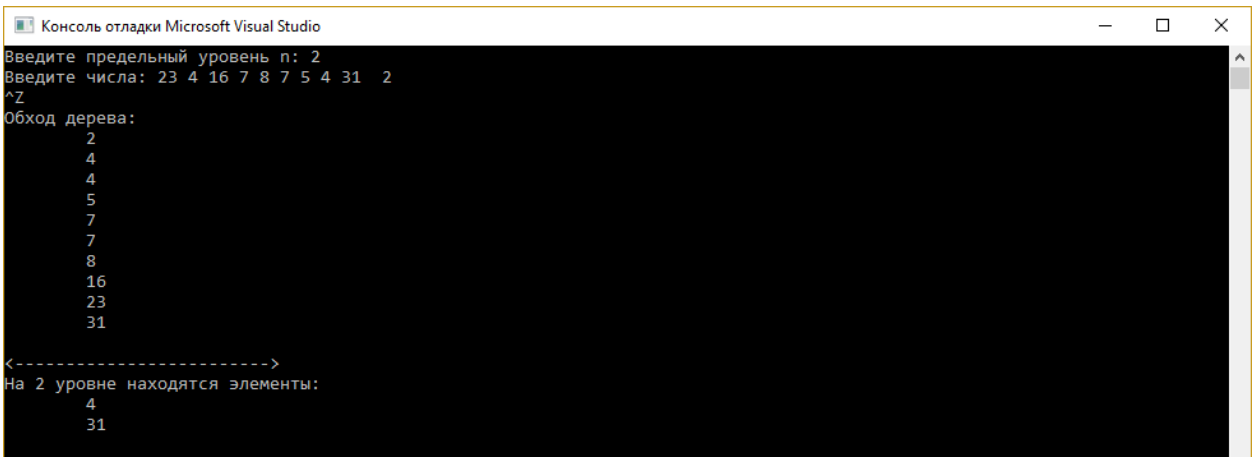
5 Код программы

```
#include "pch.h"
#include <iostream>
#include <clocale>

using namespace std;

struct node {
	int info; // Информационное поле
	node *ll, *rl; // Левый и правый указатели
};

// Построения дерева
node *tree(node *p, int current) {
	if (p == NULL) {
		p = new node;
		p->info = current;
		p->ll = NULL;
		p->rl = NULL;
	}
	else if (current < p->info) {
		p->ll = tree(p->ll, current);
	}
	else {
		p->rl = tree(p->rl, current);
	}
	return p;
}

// Обхода дерева
void treeprint(node *p){
```

```
31              if (p != NULL) {
32                      treeprint(p->ll); // по левому указателюъ
33                      cout << "\t" << p->info << endl;
34                      treeprint(p->rl); // по правому указателю
35              }
36  }
37
38  // Вывод элементов n-ого уровня дерева
39  void treeprint_level(node *p, int n) {
40              if (p != NULL) {
41                      treeprint_level(p->ll, n - 1); // по левому указателюъ
42                      if (n == 1) {
43                              cout << "\t" << p->info << endl;
44                      }
45                      treeprint_level(p->rl, n - 1); // по правому указателю
46              }
47  }
48
49  int main() {
50              setlocale(LC_ALL, "");
51
52              node *root;
53              root = NULL; // Обнуляем корень дерева
54              int current, n;
55
56              cout << "Введите предельный уровень n: ";
57              cin >> n;
58              cout << "Введите числа: ";
59              cin >> current;
60
61              while (!feof(stdin)) {
62                      root = tree(root, current);
63                      cin >> current;
64              }
65
66              cout << "Обход дерева: " << endl;
67              treeprint(root);
68              cout << endl << "<------------------------>" << endl;
69              cout << "На " << n << " уровне находятся элементы:" << endl;
70              treeprint_level(root, n);
71
72              return 0;
73  }
```

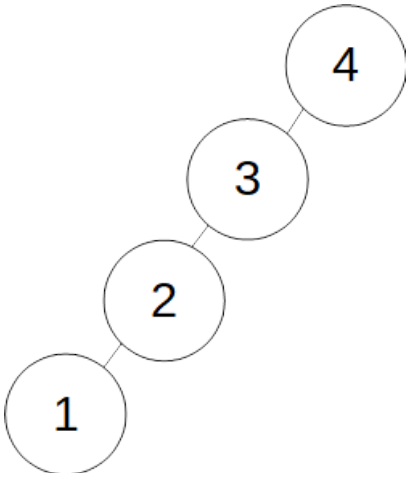
6 Результат выполнения экспериментальной части работы.

Результаты запуска программы с различными входными значениями приведены в таблице 1.



7 Набор тестовых данных

Тестовый набор: 4 3 2 1; n = 2.



Тестовый набор: 5 2 1 8 3; n = 3.