Федеральное государственное автономное

образовательное учреждение

высшего профессионального образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №10**

**“Двоичное дерево поиска”**

Преподаватель Покидышева Л. И. \_

подпись, дата

Студент Максимович Ю. В. Группа КИ15-07Б \_

подпись, дата

Красноярск 2016

**Цели работы:**

Получить практические навыки обработки нелинейных динамических структур данных.

**Порядок выполнения работы:**

Выполнить задания, указанные в экспериментальной части, подготовить отчет и защитить лабораторную работу преподавателю.

Выполняется только одно из упражнений №10\_1, №10\_2 или №10\_3 для получения соответственно удовлетворительной, хорошей или отличной оценки согласно вариантам.

**Упражнение №10\_2**

**(по индивидуальным вариантам, 80% - хор)**

Для представленного фрагмента программы, оформленного как функция и предназначенного для решения конкретной задачи обработки двоичного дерева поиска, выполнить следующее:

1. определить назначение функции;

2. прокомментировать смысл формальных параметров и возвращаемого значения, а также ход решения задачи;

3. привести графическую схему алгоритма для представленной функции;

4. дополнить предложенный фрагмент функциями построения дерева, обхода дерева и главной функции;

5. подготовить наборы тестовых данных, сопровождая их рисунками, для отладки программы, отладить и продемонстрировать преподавателю полученную программу;

6. подготовить отчет согласно выше перечисленным пунктам и защитить работу.

**Вариант 5:**

int tree\_sum\_node\_level (node \*p, int n)

{

if (p == NULL) return 0;

if (n ==1) return p->info;

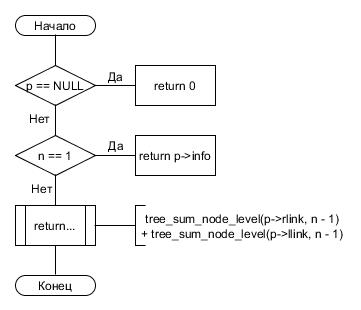
return tree\_sum\_node\_level(p->rl, n-1) +

tree\_sum\_node\_level(p->ll, n-1);

}

**Описание функции:**

1. Данная функция предназначена для нахождения суммы узлов дерева одного уровня.
2. Функция принимает рабочий указатель дерева и вводимую с клавиатуры переменную целого типа – уровень дерева, сумму узлов которого мы хотим найти. Далее идет проверка: если пользователь ввел 0(ноль), функция вернет 0(ноль), т.к. нулевого уровня не существует; если 1(один), функция вернет значение информационного поля корневого узла, т.к. на этом уровне находится только он; если большее значение, функция начнет обходить дерево и суммировать значения информационных полей указанного уровня, после полного обхода и возвращения в корневой узел функция возвратит эту самую сумму.



**Код программы:**

#include <iostream>

#include <fstream>

#include <stdio.h>

#include <conio.h>

#include <clocale>

using namespace std;

struct node

{

int info;

int c;

node \*llink, \*rlink;

};

node \*tree(node \*p, int buf)

{

if (p == NULL)

{

p = new node;

p->info = buf;

p->llink = NULL;

p->rlink = NULL;

p->c = 1;

}

else

if (buf == p->info)

p->c = p->c + 1;

else

if (buf < p->info)

p->llink = tree(p->llink, buf);

else

p->rlink = tree(p->rlink, buf);

return p;

}

void obhod(node \*p)

{

if (p != NULL)

{

cout << p->c << " " << p->info << endl;

obhod(p->llink);

obhod(p->rlink);

}

}

int tree\_sum\_node\_level(node \*p, int n)

{

if (p == NULL) return 0;

if (n == 1) return p->info;

return tree\_sum\_node\_level(p->rlink, n - 1) + tree\_sum\_node\_level(p->llink, n - 1);

}

int main()

{

setlocale(LC\_ALL, "Russian");

node \*p;

int buf, n, k;

p = NULL;

cout << "Введите количество цифр: ";

cin >> k;

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

{

cout << "Введите число: ";

cin >> buf;

p = tree(p, buf);

}

cout << endl << "Построенное дерево:" << endl;

obhod(p);

cout << "Введите уровень дерева: ";

cin >> n;

cout << tree\_sum\_node\_level(p, n);

\_getch();

return 0;

}

Пример выполнения:

