ЛАБОРАТОРНАЯ РАБОТА №4

«ПРОГРАММИРОВАНИЕ НЕЛИНЕЙНЫХ СТРУКТУР ДАННЫХ НА ЯЗЫКЕ С/С++»

* 1. **Цель работы**

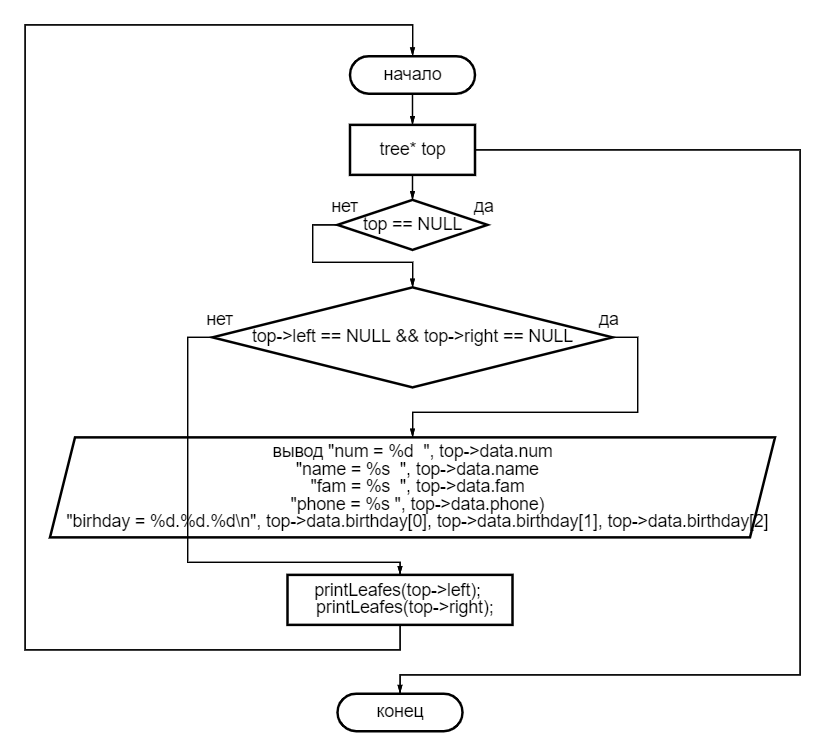
Изучение нелинейных структур данных и приобретение навыков разработки и отладки программ, использующих древовидные структуры данных. Исследование особенностей работы с поисковыми бинарными деревьями на языке С/С++.

* 1. **Вариант задания**

Функция, которая печатает элементы всех листьев дерева.

Функция печати информации о человеке, номер телефона которого введен с клавиатуры, а если такого нет, выдать на дисплей соответствующее сообщение.

**1.3** **Алгоритмы программы**



**1.4 Код программы**#include <stdlib.h>

#include <string.h>

#include <iostream>

const int d\_p = 12; // длина строки номера телефона -> 11 цифр + \0

struct NOTE

{

int num;

char name[30], fam[30];

char phone[d\_p];

int birthday[3];

};

struct tree // упорядочивание по номеру телефона

{

NOTE data;

tree\* left;

tree\* right;

};

tree\* addtree(tree\* top, const NOTE& newtree) {

if (top == NULL) { //если находимся на уровне листа,

//то выделить память под узел

top = (tree\*)malloc(sizeof(tree));

if (!top) { //выход если память не выделена

printf("Не хватает памяти\n");

return NULL;

}

top->data = newtree; //запись данных в узел

top->left = NULL; //обнуление указателей

top->right = NULL;

}

else // иначе сравниваем значение в узле с добавляемым

if (strcmp(top->data.phone, newtree.phone) > 0)

// добавляем в левое поддерево

top->left = addtree(top->left, newtree);

else //или в правое поддерево

top->right = addtree(top->right, newtree);

return top; //возвращаем указатель на корень дерева

}

void printTree(tree\* node, int level)

{

if (node == NULL)

return;

printTree(node->right, level + 1);

if (level != 0) {

for (int i = 0;i < level - 1;i++)

printf("|\t");

printf("|-------%d\n", node->data.num);

}

else

printf("%d\n", node->data.num);

printTree(node->left, level + 1);

}

void printLeafes(tree\* top) { // напечатать все листья дерева

if (top == NULL) return;

if (top->left == NULL && top->right == NULL) { // является листом

printf("num = %d ", top->data.num);

printf("name = %s ", top->data.name);

printf("fam = %s ", top->data.fam);

printf("phone = %s ", top->data.phone);

printf("birhday = %d.%d.%d\n", top->data.birthday[0], top->data.birthday[1], top->data.birthday[2]);

}

else

{

printLeafes(top->left); //обход левого поддерева

printLeafes(top->right); //обход правого поддерева

}

}

void findNote(tree\* top, char value[d\_p])

{

if (top == NULL) return;

if (strcmp(top->data.phone, value) == 0)

{

printf("num = %d ", top->data.num);

printf("name = %s ", top->data.name);

printf("fam = %s ", top->data.fam);

printf("phone = %s ", top->data.phone);

printf("birhday = %d.%d.%d\n", top->data.birthday[0], top->data.birthday[1], top->data.birthday[2]);

}

else if (strcmp(top->data.phone, value) > 0)

{

// ищем в левом поддереве

if (top->left == NULL)

{

printf("Информация не найдена!\n");

return;

}

findNote(top->left, value);

}

else // ищем в правом поддереве

{

if (top->right == NULL)

{

printf("Информация не найдена!\n");

return;

}

findNote(top->right, value);

}

}

int main()

{

setlocale(NULL, "ru");

tree\* my\_tree = NULL;

// вставляю узлы в дерево:

NOTE note;

note = { 1, "Alexey", "Nekrasov", "74948486204", {1, 6, 2005} };

my\_tree = addtree(my\_tree, note);

note = { 2, "Maxim", "Pechenkin", "76766783436", {2, 10, 2007} };

addtree(my\_tree, note);

note = { 3, "Sasha", "Serb", "77192030526", {1, 4, 2002} };

addtree(my\_tree, note);

note = { 4, "Danil", "Lekomtsev", "76613381725", {17, 2, 2005} };

addtree(my\_tree, note);

note = { 5, "Vladimir", "Lomaev", "75727767284", {4, 1, 2005} };

addtree(my\_tree, note);

note = { 6, "Slava", "Annenko", "72959093810", {5, 6, 1990} };

addtree(my\_tree, note);

note = { 7, "Oleg", "Potemkin", "74189817674", {7, 9, 2003} };

addtree(my\_tree, note);

note = { 8, "Zhitomir", "Kalachov", "79034943224", {5, 7, 2010} };

addtree(my\_tree, note);

note = { 9, "Petr", "Karlovich", "72959093850", {2, 6, 1992} };

addtree(my\_tree, note);

note = { 10, "Arkadiy", "Petrovich", "74189897674", {10, 4, 2009} };

addtree(my\_tree, note);

note = { 11, "Mistofel", "Pavlovich", "79734943224", {9, 12, 2000} };

addtree(my\_tree, note);

// отобразить дерево

printf("Отображение всего дерева:\n");

printTree(my\_tree, 0);

printf("\n\n");

// отобразить листья

printf("Отображение листьев дерева:\n");

printLeafes(my\_tree);

// ищем по существующему номеру

printf("\nИщем по существующему номеру:\n");

char phone1[] = "76613381725";

findNote(my\_tree, phone1);

// ищем по несуществующему номеру

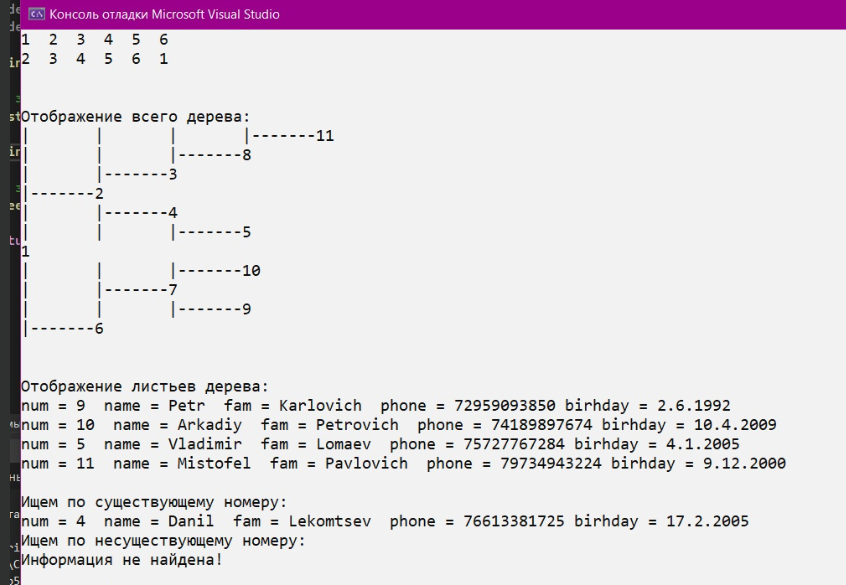
printf("Ищем по несуществующему номеру:\n");

char phone2[] = "76613381726";

findNote(my\_tree, phone2);

}

* 1. **Результат тестирования программы**



**Выводы:**  
В ходе выполнения данной лабораторной работы я ознакомился с программированием нелинейных структур данных на языке С/С++. Было изучено бинарное дерево, методы работы с ним: созданием узлов, поиск по ним. Получены навыки работы с особенностями поисковых бинарных деревьев.  
Результатом выполнения данной работы является программа, которая выводит все листья бинарного дерева, то есть те узлы, которые не имеют дочерних элементов.