МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «АиСД»

ТЕМА: Деревья

Студент гр. 9382 Пя С.

Преподаватель Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Познакомиться со структурой деревьев. Научиться создавать бинарные деревья, реализовывая их через динамическую (связанную) память (на базе указателей), и пользоваться ими. Закрепить их реализацию с помощью рекурсии на примере языка C++, освоить применение структур и классов, получить навыки работы с template c++. Выполнить работу в соответствии с заданием.

Основные теоретические положения.

Дерево – конечное множество T, состоящее из одного или более узлов, таких, что

а) имеется один специально обозначенный узел, называемый корнем данного дерева;

б) остальные узлы (исключая корень) содержатся в m >= 0 попарно не пересекающихся множествах T1, T2, ..., Tm, каждое из которых, в свою очередь, является деревом. Деревья T1, T2, ..., Tm называются поддеревьями данного дерева.

Лес – это множество (обычно упорядоченное), состоящее из некоторого (быть может, равного нулю) числа непересекающихся деревьев. Используя понятие леса, пункт *б* в определении дерева можно было бы сформулировать так: *узлы дерева, за исключением корня, образуют лес*.

КЛП – префиксная запись

ЛКП – инфиксная запись

ЛПК – постфиксная запись

Задание

Вариант №9д.

Рассматриваются бинарные деревья с элементами типа Elem (в качестве Elem использовать char). Заданы перечисления узлов некоторого дерева b в порядке КЛП и ЛКП. Требуется:

- восстановить дерево b и вывести его изображение;

- перечислить узлы дерева b в порядке ЛПК.

Ход работы.

1) Разработан алгоритм:

На вход подаются два выражения, являющиеся разными записями одного и того же бинарного дерева (ЛКП и КЛП). Используя теоретические знания о том, что во втором выражении первыми узлами записываются корни поддеревьев, находим эти корни в первом выражении, они будут в не с краю, потому что в ЛКП записи сначала записываются левые, потом корни, после правые поддеревья. Затем мы делим первое выражение на два, они будут образовывать два поддерева и будут являться левым и правым поддеревом соответственно нашего главного дерева. Затем таким образом рекурсивно составляем из выражений бинарное дерево. Были устранены все утечки памяти, написана реакция на не открытие файла и на некорректные данные.

Предусмотрен механизм простейшего взаимодействия с пользователем, позволяющий понять алгоритм исполнения программы, с помощью вывода сообщений. Также был предусмотрен ввод данных с клавиатуры.

2) Использованы функции:

1. main

Сигнатура: int main().

Назначение: является основной функцией и телом программы.

Описание аргументов: без аргументов.

Возвращаемое значение: Функция возвращает 0.

Использованы структуры:

1. struct Node

Назначение: играет роль узла дерева, нужен для составления дерева.

Описание содержимого: Elem data – переменная типа Elem (параметр шаблона класса, в качестве Elem использован char), которая хранит в себе содержимое узла; struct Node *left и struct Node *right – указатели на структуры, являющиеся левым поддеревом и правым поддеревом соответственно; Node() – конструктор структуры, который инициализирует указатели left и right.

Использованы классы:

1. BinaryTree

Назначение: в нем хранятся методы для создания дерева и его отображения.

Описание содержимого: private: структура struct Node;

public: int i – счетчик для КЛП записи, чтобы по очереди брать узлы в строке; node* tree – указатель на структуру, хранящую дерево, к которому можно обратиться.

Использованы методы класса BinaryTree:

1. createBinaryTree

Сигнатура: node* createBinaryTree (char LKP[], char KLP[]).

Назначение: приватный метод: создает бинарное дерево, состоящее из структур.

Описание аргументов: строки LKP и KLP, являющимися формами записи узлов дерева.

Возвращаемое значение: указатель на основной корень дерева типа node.

2. BinaryTree

Сигнатура: BinaryTree(char LKP[], char KLP[], int n = 0).

Назначение: Публичный конструктор, инициализирующий переменные. В нем создается дерево.

Описание аргументов: строки LKP и KLP, являющимися формами записи узлов дерева, n – переменная типа int, которая указывает номер уровня узла.

3. printInorder

Сигнатура: void printInorder(node* node, int n, char* shift).

Назначение: составляет схематическое изображение дерева.

Описание аргументов: node - указатель на структуру типа node, являющуюся деревом, который надо вывести, n – номер уровня узла, shift – строка, содержащая возможные номера уровней узлов.

4. printLPK

Сигнатура: static void printLPK(node* node).

Назначение: выводит постфиксную запись ЛПК дерева.

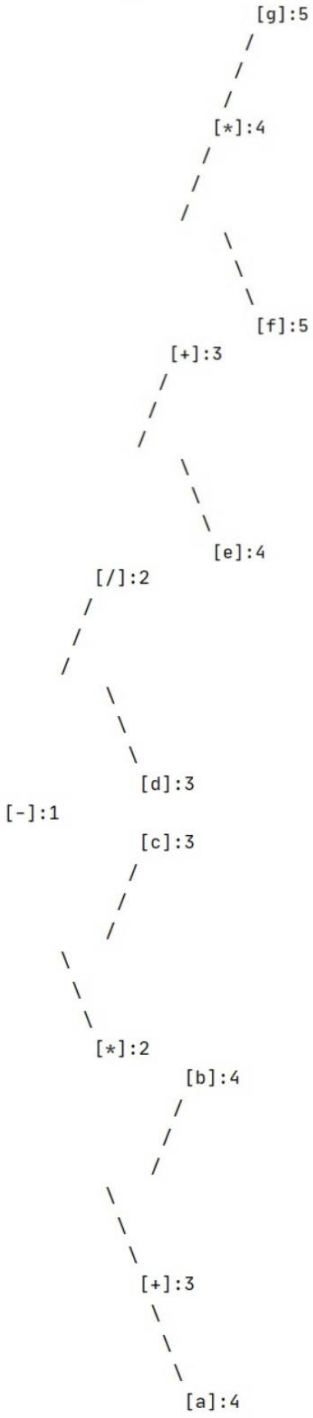
Описание аргументов: указатель на узел основного корня дерева типа node.

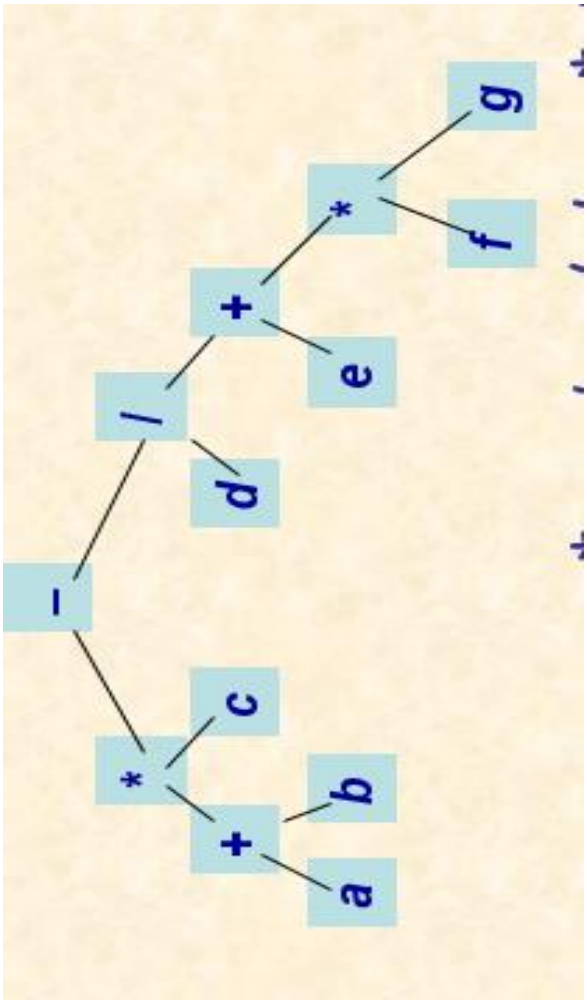
Реализация рекурсивного метода:

На вход подаются два выражения, являющимися ЛКП и КЛП записями. Затем проверяются данные на корректность. Создается узел дерева, куда записывается следующее значение в выражении КЛП. После значение ищется в выражении

ЛКП, и это выражение делится на две части, которые отдаются в следующие методы рекурсии, пока в выражениях есть значения. После возвращается все дерево.

Пример работы программы.





Входные данные	Выходные данные
3 a+b*c-d/e+f*g - *+abc/d+e*fg	Do you want to enter data(0) or read it from file(write number of file)? 3 Data is: a+b*c-d/e+f*g -*+abc/d+e*fg 1.Data of current node is: - 1.Left subtree is: a+b*c 2.Data of current node is: * 2.Left subtree is: a+b 3.Data of current node is: + 3.Left subtree is: a 4.Data of current node is: a 4.Left subtree is: 4.Right subtree is: 3.Right subtree is: b 4.Data of current node is: b 4.Left subtree is: 4.Right subtree is: 2.Right subtree is: c 3.Data of current node is: c 3.Left subtree is: 3.Right subtree is: 1.Right subtree is: d/e+f*g 2.Data of current node is: / 2.Left subtree is: d 3.Data of current node is: d 3.Left subtree is: 3.Right subtree is: 2.Right subtree is: e+f*g 3.Data of current node is: + 3.Left subtree is: e 4.Data of current node is: e 4.Left subtree is:

	4.Right subtree is: 3.Right subtree is: f*g 4.Data of current node is: * 4.Left subtree is: f 5.Data of current node is: f 5.Left subtree is: 5.Right subtree is: 4.Right subtree is: g 5.Data of current node is: g 5.Left subtree is: 5.Right subtree is: [g]:5 / / / [*]:4 / / / \ \ \ [f]:5 [+]:3 / / / \ \ \ [e]:4 [/]:2 / / / \

	\ \ [d]:3 [-]:1 [c]:3 / / / \ \ \ [*]:2 [b]:4 / / / \ \ \ [+]:3 \ \ \ [a]:4 LPK is a b + c * d e f g * + / -

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии

1.	1 425136 124536	Do you want to enter data(0) or read it from file(write number of file)? 1 Data is: 425136 124536 1.Data of current node is: 1 1.Left subtree is: 425 2.Data of current node is: 2 2.Left subtree is: 4 3.Data of current node is: 4 3.Left subtree is: 3.Right subtree is: 2.Right subtree is: 5 3.Data of current node is: 5 3.Left subtree is: 3.Right subtree is: 1.Right subtree is: 36 2.Data of current node is: 3 2.Left subtree is: 2.Right subtree is: 6 3.Data of current node is: 6 3.Left subtree is: 3.Right subtree is: [6]:3 / / / [3]:2 / / / [1]:1 \ \ \ [5]:3 \ \ \ [2]:2 [4]:3 / / / LPK is 4 5 2 6 3 1	Проверка на корректность работы программы

2.	2 1+4*9-5 *+14-95	Do you want to enter data(0) or read it from file(write number of file)? 2 Data is: 1+4*9-5 *+14-95 1.Data of current node is: * 1.Left subtree is: 1+4 2.Data of current node is: + 2.Left subtree is: 1 3.Data of current node is: 1 3.Left subtree is: 3.Right subtree is: 2.Right subtree is: 4 3.Data of current node is: 4 3.Left subtree is: 3.Right subtree is: 1.Right subtree is: 9-5 2.Data of current node is: - 2.Left subtree is: 9 3.Data of current node is: 9 3.Left subtree is: 3.Right subtree is: 2.Right subtree is: 5 3.Data of current node is: 5 3.Left subtree is: 3.Right subtree is: [5]:3 / / / [-]:2 / / / \ \ \ [9]:3 [*]:1 [4]:3 / / / \ \ \ [+]:2 \ \	Проверка на корректность работы

		\ [1]:3 LPK is 1 4 + 9 5 - *	
4.	4	Do you want to enter data(0) or read it from file(write number of file)? 4 Data is: Error Data	Проверка на корректность работы с пустыми строками
5.	5 14235 12345	Do you want to enter data(0) or read it from file(write number of file)? 5 Data is: 14235 12345 1.Data of current node is: 1 1.Left subtree is: 1.Right subtree is: 4235 2.Data of current node is: 2 2.Left subtree is: 4 2.Right subtree is: 35 Error Data	Проверка на корректность работы с неверными данными
6	0 14235 12345	Do you want to enter data(0) or read it from file(write number of file)? 0 14235 12345 Data is: 14235 12345 1.Data of current node is: 1 1.Left subtree is: 1.Right subtree is: 4235 2.Data of current node is: 2 2.Left subtree is: 4 2.Right subtree is: 35 Error Data	Проверка на корректность работы с входными данными с клавиатуры

Выводы.

В ходе работы была освоена реализация бинарного дерева на основе рекурсии и структур, отработано понимание его применения, и отработаны навыки письма в C++.

Код программы можно найти в приложении А.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <cstring>

using namespace std;
template <typename Elem>// Elem - параметр шаблона класса, в качестве него
используем char
class BinaryTree {
private:
    typedef struct Node {//структура, описывающая содержимое узла
        Elem data;//данные, хранящиеся в узле
        struct Node *left;//левое поддерево
        struct Node *right;//правое поддерево
        Node() {
            left = nullptr;
            right = nullptr;
        }
    } node;

    node *createBinaryTree(char LKP[], char KLP[], int n = 0) {//рекурсивный
метод для создания дерева
        n++;
        if (strlen(KLP) <= i) {//в случае некорректных данных выводится
сообщение об ошибке
            cout << "Error Data";
            return nullptr;
        }
        node *binaryTree = new node();
        binaryTree->data = KLP[i++];//содержимое узла
        char *rshift = strchr(LKP, binaryTree->data);
        if (rshift == nullptr) {//в случае некорректных данных выводится
сообщение об ошибке
            i = strlen(KLP) + 1;
            return nullptr;
        }
        char *lshift = new char[100]();
        strncat(lshift, LKP, rshift - LKP);
        strcat(lshift, "\0");
        for (int y = 0; y < n;y++)
            cout << " ";
        cout << n << ".Data of current node is: " << binaryTree->data << "\n";
        for (int y = 0; y < n;y++)
            cout << " ";
        cout << n << ".Left subtree is: " << lshift << "\n";
        if (strlen(lshift) != 0) {
            binaryTree->left = createBinaryTree(lshift, KLP, n);//передаем левую
часть строки, из которой формируется поддерево
        }
        for (int y = 0; y < n;y++)
            cout << " ";
        cout << n << ".Right subtree is: " << rshift + 1 << "\n";
```

```
        if (strlen(rshift) != 1) {//так как в правой части хранится искомый
узел, то здесь рекурсивный цикл прекращается на количестве элементов в правой
части, равном 1
            binaryTree->right = createBinaryTree(rshift + 1, KLP, n);//передаем
правую часть строки, из которой формируется поддерево
        }
        delete[] lshift;
        return binaryTree;
    }

public:
    int i;//счетчик для KLP
    node *tree;//бинарное дерево, созданно для многократного безопасного
обращения к одному и тому же дереву

    BinaryTree(char LKP[], char KLP[]) {
        i = 0;
        tree = createBinaryTree(LKP, KLP, 0);//создание дерева
    }

    void printInorder(node *node, int n, char *shift) {//вывод дерева
        n++;//номер уровня
        char k[10];
        char *_k_ = new char[10]();//строка, в которой будет хранится номер
уровня
        _k_[0] = ' ';
        if (node == NULL) {//завершение рекурсивного цикла
            delete[] _k_;
            return;
        }
        printInorder(node->right, n, shift);//вход в самую левую ветку
        sprintf(k, "%d", n);
        strcat(_k_, k);
        strcat(_k_, " ");
        char *t = strstr(shift, _k_);//поиск номера в строке из номеров
        delete[] _k_;
        if (!(n == 1) &&
            t) {//в случае нахождения текущего номера в строке номеров знаем,
что узел находится левее в бинарном дереве
            for (int i = 0; i < n + 3 * n; i++)//сдвигаем узел до его уровня
                cout << " ";
            cout << "[" << node->data << "]" << ":" << n << "\n";
            char *tshift = new char[50]();
            strncat(tshift, shift, t - shift);//убираем номер из строки номеров
            strcat(tshift, t + 2);
            delete[] shift;
            shift = new char[50]();
            strcat(shift, tshift);
            delete[] tshift;
            int m = 1;
            while (m != 4) {
                for (int i = 0; i < n + 3 * n - m; i++)//рисуем ветку
                    cout << " ";
                cout << "/\n";
                m++;
            }
        } else if (n != 1 &&
                   !t) {//в случае отсутствия текущего номера в строке номеров
знаем, что узел находится правее в бинарном дереве
            int m = 3;
            while (m != 0) {
                for (int i = 0; i < n + 3 * n - m; i++)//рисуем ветку
                    cout << " ";
                cout << "\\\n";//рисуем ветку
```

```
                m--;
            }
            for (int i = 0; i < n + 3 * n; i++)
                cout << " ";//сдвигаем узел до его уровня
            cout << "[" << node->data << "]" << ":" << n << "\n";
            sprintf(k, "%d", n);//добавляем номер обратно в строку номеров
            strcat(shift, k);
            strcat(shift, " ");
        } else//если номер 1, то это вершина дерева, его корень
            cout << "[" << node->data << "]" << ":" << n << "\n";
        printInorder(node->left, n, shift);//после самого левого прохода идем
вправый
    }

    static void printLPK(node *node) {//вывод постфиксной записи (ЛПК)
        if (node != nullptr) {
            printLPK(node->left);//сначала идем в левые поддеревья
            printLPK(node->right);//после в правые поддеревья
            cout << node->data << " ";//пишем узел
        }
    }
};

int main() {
    auto *in = new char[30]();//строка для инфиксной записи ЛКП
    auto *pre = new char[30]();//строка для префиксной записи КЛП
    char x;//переменная для выбора ввода пользователем
    cout << "Do you want to enter data(0) or read it from file(write number of
file)?\n";
    cin >> x;
    if (x ==
        48) {//так как на вход принимается символ, то номер нуля = 48, это
работает, пока количество тестов не привышает 9
        cin >> in >> pre;//считывание строк из консоли
    } else {
        char *file = new char[50]();
        strcat(file, ".//Tests//");
        file[strlen(file)] = x;
        strcat(file, ".txt");
        fstream fin(file);//файл для считывания выражений
        if (!fin.is_open())
            throw ("File cannot be opened");
        fin >> in >> pre;//считывание строк из файла
        delete[] file;
    }
    cout << "Data is:\n" << in << "\n" << pre << "\n";
    BinaryTree<char> tree(in, pre);//создание экземпляра класса BinaryTree
    if (tree.tree == nullptr || strlen(pre) < tree.i)
        return -1;
    delete[] in;
    char *shift = new char[50]();//строка, в которой будут находиться номера
уровней узлов для вывода дерева
    shift[0] = ' ';
    char k[10];
    for (int i = 1; i < strlen(pre) + 1; i++) {//запись номеров в строку, они не
будут превышать количество узлов
        sprintf(k, "%d", i);
        strcat(shift, k);
        strcat(shift, " ");
    }
    shift[strlen(shift)] = '\0';
    tree.printInorder(tree.tree, 0, shift);//вывод дерева
    cout << "\nLPK is\n";
    tree.printLPK(tree.tree);//вывод постфикной записи
```

```
    delete[] pre;
    delete[] shift;
    return 0;
}
```