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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Деревья

Студент гр. 9303 _____ Молодцев Д.А.

Преподаватель _____ Филатов Ар.Ю.

Санкт-Петербург
2020

Цель работы.

Создание базового функционала для работы с бинарным деревом, а ткаже написания функции-преобразования для бинарного дерева-формулы.

Задание.

Вариант 14 (ссылочная реализация)

Преобразовать дерево-формулу t, заменяя в нем все поддеревья, соответствующие формулам $((f_1*f_2)+(f_1*f_3))$, $((f_1*f_3)+(f_2*f_3))$ на поддеревья, соответствующие формулам $(f_1*(f_2+f_3))$, $((f_1+f_2)*f_3)$.

Основные теоретические положения.

Дадим формальное определение дерева.

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 $m \geq 0$ попарно не пересекающихся множествах $T_1, T_2, \ldots, T_m$, каждое из которых, в свою очередь, является деревом. Деревья $T_1, T_2, \ldots, T_m$ называются поддеревьями данного дерева. При программировании и разработке вычислительных алгоритмов удобно использовать именно такое рекурсивное определение, поскольку рекурсивность является естественной характеристикой этой структуры данных.

Выполнение работы.

Для представления бинарного дерева (БД) в памяти были реализованы следующие шаблонные классы:

BinTree – базовый класс, который представляет собой бинарное дерево. Содержит 2 указателя на класс BinTree, а также все основные методы для работы с бинарным деревом.

Для работы с деревом для класса BinTree были реализованы следующие функцц:

void read_tree() – считывает строку-формулу, преобразует ее в дерево.
bool Is_formula() – true, если поддерево имеет вид искомой формулы.
void Change_tree() – преобразует поддерево-формулу к требуемому виду.
void Detour() – вызывает функцию обхода для преобразования.
void ShowTree() – вызывает функцию обхода для печати.

Исходный код программы представлен в приложении А. Результаты тестирования включены в приложение Б

Выводы.

Был реализован класс, позволяющий работать с бинарным деревом. Так же были реализованы функции для работы с бинарным деревом Также было проведено тестирование программы. Следует отметить, в данном случае некорректными считаются те данные, которые не содержат скобочного представления дерева, такие строки не будут считываться и программа перейдет к следующей строке файла.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

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

using namespace std;

class BinTree
{
private:
    string value;
    BinTree* left;
    BinTree* right;

public:
    BinTree(){
        left = nullptr;
        right = nullptr;
    }

    BinTree* Get_left(){
        return this->left;
    }

    BinTree* Get_right(){
        return this->right;
    }

    void Create_right(){
        this->right=new BinTree;
    }

    void Create_left(){
```

```
        this->left=new BinTree;
    }

    string Get_value(){
        return this->value;
    }

    void Delete_left(){
        if(left!= nullptr){
            left->Delete_left();
            left->Delete_right();
            delete left;
        }
    }

    void Delete_right(){
        if(right!= nullptr){
            right->Delete_left();
            right->Delete_right();
            delete right;
        }
    }

    void Set_value(string val){
        this->value=val;
    }
};




string getLeft( string &str, int to)
{
    string out = "";
```

```
    for (int i = 1; i < to; i++)
    {
        out += str[i];
    }
    return out;
}

string getRight( string &str, int from)
{
    string out = "";
    for (int i = from+1; i < str.length()-1; i++)
    {
        out += str[i];
    }
    return out;
}

void read_tree(string str, BinTree *head ){
    string c="";
    int i = 0;
    int depth = 0;
    while (i < str.length()){
        if(str[i] == '(') depth++;
        if(str[i] == ')') depth--;
        if ( (str[i]=='+' || str[i]=='*') && depth == 1) {
            c  += str[i];
            break;
        }
        i++;
    }
    if(c == "+" || c == "*" ) {
        head->Set_value(c);
        head->Create_left() ;
        head->Create_right();
```

```
        string Left = getLeft(str, i);
        string Right = getRight(str, i);
        read_tree(Left, head->Get_left()) ;
        read_tree(Right, head->Get_right());
    }else{
        head->Set_value(str);
        return;
    }

}



bool Is_formula(BinTree &head) {
    if(head.Get_value()=="*" || head.Get_value()=="+") {
        if(head.Get_right()->Get_value()=="*" &&
head.Get_left()->Get_value()=="*") {
            if( head.Get_right()->Get_right()->Get_value() ==
head.Get_left()->Get_right()->Get_value()){
                return true;
            }else if(head.Get_right()->Get_left()->Get_value()
== head.Get_left()->Get_left()->Get_value()){
                return true;
            }
            else if(head.Get_right()->Get_left()->Get_value() ==
head.Get_left()->Get_right()->Get_value()){
                return true;
            }else if(head.Get_right()->Get_right()->Get_value()
== head.Get_left()->Get_left()->Get_value()){
                return true;
            }
        }
        return false;
    }
```

```
    return false;
}


void Change_tree(BinTree *left, BinTree *right, string root){
    string a,b,c;
    if(right->Get_right()->Get_value() == left->Get_right()-
>Get_value()){
        c=right->Get_right()->Get_value();
        a=right->Get_left()->Get_value();
        b=left->Get_left()->Get_value();
    }
    else if(right->Get_left()->Get_value() == left->Get_left()-
>Get_value()){
        c=right->Get_left()->Get_value();
        a=right->Get_right()->Get_value();
        b=left->Get_right()->Get_value();
    }
    else if(right->Get_left()->Get_value() == left->Get_right()-
>Get_value()){
        c=right->Get_left()->Get_value();
        a=right->Get_right()->Get_value();
        b=left->Get_left()->Get_value();
    }
    else if(right->Get_right()->Get_value() == left->Get_left()-
>Get_value()){
        c=right->Get_right()->Get_value();
        a=right->Get_left()->Get_value();
        b=left->Get_right()->Get_value();
    }
    right->Set_value(c);
    left->Set_value(root);
    left->Get_right()->Set_value(b);
    left->Get_left()->Set_value(a);
```

```
    right->Delete_left();
    right->Delete_right();
}

string ShowTree(BinTree &head){
    if(head.Get_value() == "+" || head.Get_value() == "*" )
    {
        return "("
               + ShowTree( *(head.Get_left()))
               + head.Get_value()
               + ShowTree( *(head.Get_right()))
               + ")";
    }
    return head.Get_value();
}

void DetourTree(BinTree *head){
    if((head->Get_value() == "+" || head->Get_value() == "*") &&
Is_formula(*head)){
        string s=head->Get_value();
        head->Set_value(head->Get_right()->Get_value());
        Change_tree(head->Get_left(),head->Get_right(),s);
    }else{
        if(head->Get_value()=="*" || head->Get_value()=="+"){
            if(head->Get_left()->Get_value()=="*" || head-
>Get_left()->Get_value()=="+"){
                DetourTree(head->Get_left());
            }
            if(head->Get_right()->Get_value()=="*" || head-
>Get_right()->Get_value()=="+"){
                DetourTree(head->Get_right());
            }
        }
    }
```

```
}

int main() {
    string path;
    cout<<"Enter file name:"<<'\n';
    getline(cin,path);
    ifstream input(path);
    if(!input){
        cout<<"Wrong file name, try again!\n";
    }
    else{
        string line;
        while(!input.eof()){
            getline(input,line);
            if(input.eof()) break;
            cout<<"Input formula:\n";
            cout<<line<<"\n";
            BinTree *tree=new BinTree;
            read_tree(line, tree);
            DetourTree(tree);
            cout<<"Output formula:\n";
            cout<<ShowTree(*tree)<<"\n";
        }
    }
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Файл со входными данными: test.txt

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Input.txt ((a*1)+(c*1)) ((c+a)*1)	((c+a)*1)	Программа работает корректно.
2.	Input.txt ((a*b)+(a*c))	((b+c)*a)	Программа работает корректно.
3.	test.txt ((b+c)*a)	((b*a)+(c*a))	Программа работает корректно.
4.	Input.txt (((a*b)+(a*c))+((a*c)+(b*c)))	(((c+b)*a)+((b+a)*c))	Программа работает корректно.
5.	Input.txt ((a+b)*(c+d))	((a+b)*(c+d))	Программа работает корректно.
6. (тест на некорректных данных)	test.txt ((a+b)+(c+d))	((a+b)+(c+d)	Программа работает корректно.
7. (тест на некорректном имени файла)	asdasd	Wrong file name, try again!	Программа работает корректно.
8.(тес на некорректных данных)	Input.txt In vino veritas!	Wrong string!	Программа работает корректно.