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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Бинарные деревья

Студент гр. 7383 _____ Левкович Д.В.

Преподаватель _____ Размочева Н.В.

Санкт-Петербург

2018

ОГЛАВЛЕНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель работы: познакомиться с основными понятиями и реализацией бинарного дерева на языке программирования C++.

Формулировка задачи:

а) для заданной формулы *f* построить дерево-формулу *t*;

б) для заданного дерева-формулы *t* напечатать соответствующую формулу *f*;

д) если в дереве-формуле *t* терминалами являются только цифры, то вычислить (как целое число) значение дерева-формулы *t*;

и) построить дерево-формулу *t*1 – производную дерева-формулы *t* по заданной переменной.

2. РЕАЛИЗАЦИЯ ЗАДАЧИ

В функции main выводится приглашение выбрать способ ввода входных данных либо выйти из программы. Далее вызывается рекурсивная функция readForm, которая считывает исходную строку по символам и создает бинарное дерево с помощью функции ConsBT.

Функция writeForm рекурсивно обходит дерево в прямом порядке выводя его на экран.

Функция print дает графическое представление о дереве.

Функция Right возвращает правое поддерево, функция Left – левое, функция Root – возвращает значение в узле.

Функция destroy удаляет наше дерево, функция isNULL проверяет дерево, пустое оно или нет. Это базовые функции для работы с бинарным деревом.

Функция calc подсчитывает значение выражения, если она встречает хотя бы один символ – выбрасывается исключение и результатом возвращается 0.

Функция derivitate находит производную нашей формулы по заданной переменной и записывает результат в новое дерево. Если в узле переменная,

создаем дерево, где корнем уже будет «1», иначе «0», так как производная константы равна нулю. Если в узле лежит знак операции «+» или « - », то создаем дерево, где в узле лежит данный оператор. Если в узле знак « * », то создаем новое дерево используя формулу вычисления производной произведения двух функций. С помощью функции writeForm выводим новое дерево на экран.

3. ТЕСТИРОВАНИЕ

Программа была собрана в компиляторе G++ с ключом -std=c++14 в OS Linux Ubuntu 16.04 LTS. В ходе тестирования было проверено, что программа работает верно.

4. ВЫВОД

В ходе работы были получены навыки реализации и работы с бинарным деревом на базе указателей. Бинарное дерево является рекурсивной структурой данных, для которой легко определяются рекурсивные алгоритмы.

ПРИЛОЖЕНИЕ А.

ТЕСТОВЫЕ СЛУЧАИ

Таблица 2 —Тестирование

Входные данные	Результат
x ((a+x)*(x*x))	Empty (((0+1)*(x*x))+((a+x)*((1*x)+(x*1))))
x ((3+4)*5)	35 (((0+0)*5)+((3+4)*0))
x x ((x-a)*(((x*x)+(y*x))+(b*a)))	Empty 1 (((1-0)*(((x*x)+(y*x))+(b*a)))+((x-a)*((((1*x)+(x*1))+((0*x)+(y*1)))+((0*a)+(b*0)))))
A (a+b (x*a)	expected ')' ((1*a)+(x*0))

ПРИЛОЖЕНИЕ Б.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <sstream>
#include <cstdlib>
#include <fstream>
#include <cstring>
#include <exception>
#include <windows.h>

using namespace std ;

typedef char base;
    class node {
    public:
        base info;
        node *lt;
        node *rt;
            // constructor
            node () {
                lt = nullptr;
                rt = nullptr;
            }
    };
    typedef node *binTree; // "представитель" бинарного дерева
    binTree Create(void);
    bool isNull(binTree);
    base RootBT (binTree); // для непустого бин.дерева
    binTree Left (binTree);// для непустого бин.дерева
    binTree Right (binTree);// для непустого бин.дерева
    binTree ConsBT(const base &x, const binTree &lst,  const binTree &rst);
    void destroy (binTree&);

    typedef binTree Form;
    Form readForm(std::istream &in);
    void writeForm(std::ostream &out, const Form f);
    Form deriviate(const Form f, char var);


    binTree Create()
    {
        return NULL;
    }

    bool isNull(binTree b)
    {
        return (b == NULL);
    }

    base RootBT (binTree b)                    // для непустого бин.дерева
    {   if (b == NULL) { cerr << "Error: RootBT(null) \n"; exit(1); }
        else return b->info;
    }

    binTree Left (binTree b)            // для непустого бин.дерева
    {   if (b == NULL) { cerr << "Error: Left(null) \n"; exit(1); }
        else return b ->lt;
    }
```

```
    binTree Right (binTree b)           // для непустого бин.дерева
    {   if (b == NULL) { cerr << "Error: Right(null) \n"; exit(1); }
        else return b->rt;
    }

    binTree ConsBT(const base &x, const binTree &lst, const binTree &rst)
    {   binTree p;
        p = new node;
        if ( p != NULL) {
            p ->info = x;
            p ->lt = lst;
            p ->rt = rst;
            return p;
        }
        else {cerr << "Memory not enough\n"; exit(1);}
    }
//-------------------------------------
    void destroy (binTree &b)
    {   if (b != NULL)  {
            destroy (b->lt);
            destroy (b->rt);
            delete b;
            b = NULL;
        }
    }

    bool isSign(char c){
        return c=='+'||c=='-'||c=='*';
    }
    bool isTerminal(char c){
        return isdigit(c)||(c>='a'&&c<='z');
    }
    Form readForm(istream &in){
        if(isTerminal(in.peek())){
            return ConsBT(in.get(),NULL,NULL);
        }
        if(in.peek()=='('){
            in.get();
            Form b=readForm(in);
            if(!isSign(in.peek()))throw "expected sigh\n";
            char s=in.get();
            b=ConsBT(s,b,readForm(in));
            if(in.peek()!=')')throw "expected ')'";
            in.get();
            return b;
        }
        throw "expected term or '('\n";
    }
    void writeForm(ostream &out, const Form f){
        if(isNull(f))throw "Error, empty formula\n";
        if(isNull(Right(f))&&isNull(Left(f))){
            out<<RootBT(f);
            return;
        }
        if(isNull(Right(f))||isNull(Left(f)))
            throw "Knot should have Left and Right\n";
        out<<'(';
        writeForm(out,Left(f));
        out<<RootBT(f);
        writeForm(out,Right(f));
```

```
    out<<')';
}

Form deriviate(const Form f, char var){
    if(var==RootBT(f)||isdigit(RootBT(f))||isalpha(RootBT(f))){
        if(!isNull(Right(f))||!isNull(Left(f)))
            throw "variable or const shouldn't be in root\n";
        if(var==RootBT(f))             // x'=1
            return ConsBT('1',NULL,NULL);
        return ConsBT('0',NULL,NULL);  // const'=0
    }
    if(RootBT(f)=='+'||RootBT(f)=='-') // (f(x) +/- g(x))' = f'(x) +/- g'(x)
        return ConsBT(
                    RootBT(f),
                    deriviate(Left(f),var),
                    deriviate(Right(f),var)
                );
    if(RootBT(f)=='*')                 // (f(x)*g(x))'=f'(x)*g(x)+f(x)*g'(x)
        return ConsBT(
                    '+',
                    ConsBT(
                        '*',
                        deriviate(Left(f),var),
                        Right(f)
                    ),
                    ConsBT(
                        '*',
                        Left(f),
                        deriviate(Right(f),var)
                    )
                );
    throw "unknown char\n";
}
int calc(Form f){
    if(isalpha(f->info) || isalpha(f->info))
        throw "Empty\n";
    int x, y, z;
    if((f->info=='+')||(f->info=='-')||(f->info=='*')){
        x=calc(f->lt);
        y=calc(f->rt);

        if (f->info=='+'){
            cout<<"you are here: "<<x<<'+'<<y<<endl;
            z=x+y;
        }
        else if (f->info=='-'){
            z=x-y;
            cout<<"you are here: "<<x<<'-'<<y<<endl;
        }
        else if (f->info=='*'){
            cout<<"you are here: "<<x<<'*'<<y<<endl;
            z=x*y;
        }
        return z;
    }
    else return f->info - '0';
}
void print(Form f, int l){
    if(f==nullptr){
        for(int i = 0;i<l;i++)
            cout<<"\t";
```

```
            cout<<'#'<<endl;
            return;
        }
        print(f->rt, l+1);
                for(int i = 0;i<l;i++)
                    cout<<"\t";
                cout<<f->info<<endl;
                print(f->lt, l+1);
    }

int main (){
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    Form f;
    Form fPrime;
    filebuf file;
    string file_name;
    stringbuf exp;
    string temp_str;
    int run = 1;
    string k;
    int m;
    while(run){
    cout<<"enter the variable for derivative"<<endl;
    char c;
    cin>>c;
    cin.ignore();
    try{
        cout<<"Source formula:";
        writeForm(cout, (f=readForm(cin)));
        cout<<endl;
        try{
        cout<<"expression value: ";
        cout<<calc(f)<<endl;
        }
        catch(const char* e){
            cout<<e;
        }
        cout<<"graphical representation:"<<endl;
        cout<<endl;
        print(f, 0);
        cout<<"derivative for "<<c<<endl;
        writeForm(cout, (fPrime=deriviate(f,c)));
        cout<<endl;
    }
    catch(const char* t){
        cout<<t;
    }
    destroy(f);
    destroy(fPrime);
    cout<<"Continue? Press 1. Else press 0"<<endl;
        cin>>m;
        cin.ignore();
        if(m==0)
            return 0;
    }
    return 0;
}
```