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

Студентка гр. 7383 _____ Чемова К.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

СОДЕРЖАНИЕ

Цель работы

Познакомиться с абстрактным типом данных бинарное дерево. Реализовать программу для преобразования дерева-формулы.

Формулировка задания варианта 9(а, б, в, ж)-в:

Формулу вида

< формула > ::= < терминал > | (< формула > < знак > < формула >)

< знак > ::= + | – | *

< терминал > ::= 0 | 1 | ... | 9 | *a* | *b* | ... | *z*

можно представить в виде бинарного дерева («***дерева-формулы***») с элементами типа *char* согласно следующим правилам:

- формула из одного терминала представляется деревом из одной вершины с этим терминалом;
- формула вида (f_1 *s* f_2) представляется деревом, в котором корень – это знак *s*, а левое и правое поддеревья – соответствующие представления формул f_1 и f_2. Например, формула (5 * (*a* + 3)) представляется деревом-формулой, показанной на рис. 1.

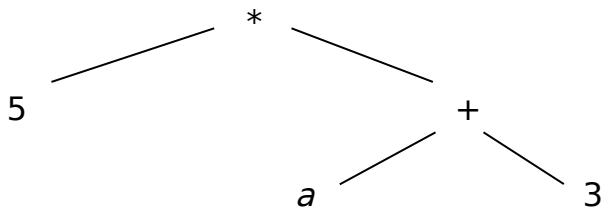

Рисунок 1 – Дерево-формула

Требуется:

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

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

в) с помощью построения дерева-формулы *t* преобразовать заданную формулу *f* из инфиксной формы в префиксную;

ж) преобразовать дерево-формулу *t*, заменяя в нем все поддеревья, соответствующие формулам (f_1 * (f_2 + f_3)) и ((f_1 + f_2) * f_3), на поддеревья, соответствующие формулам ((f_1 * f_2) + (f_1 * f_3)) и ((f_1 * f_3) + (f_2 * f_3));

Реализация задачи

Для решения поставленной задачи были реализованы следующие функции:

void buildTree – строит дерево на основе массива;

void print – печатает дерево;

void prefix – преобразует в префиксную форму;

string change – преобразует дерево-формулу в формулу этого дерева;

void right – раскрывает скобки с множителем пред скобкой;

void left – раскрывает скобки с множителем после скобки;

bool test – проверка на некорректные данные;

Тестирование

Программа была собрана в компиляторе g++ в OS Linux Ubuntu 16.04 при помощи g++. В других системах тестирование не проводилось. Результаты тестирования приведены в приложении А.

Выводы

В ходе лабораторной работы было изучено бинарное дерево как тип данных и способ его реализации на векторе. Были получены практические навыки работы с бинарным деревом. Была написана программа, выводящая дерево и формулу, преобразующая инфиксную форму записи в префиксную, а также раскрывающую скобки при умножении.

ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ

В табл. 1 приведены примеры работы программы.

Таблица 1 – Тестовые случаи

Входные данные	Результат
(a+b)	Дерево-формула: +ab [a] [+] [b] Формула дерева: (a+b) Префиксная форма записи дерева: +ab Упрощенная формула: (a+b)
((a+b)*c)	Дерево-формула: *+cab## [a] [+] [b] [*] [c] Формула дерева: ((a+b)*c) Префиксная форма записи дерева: *+abc Упрощенная формула: ((a+c)*(b+c))
(a/c)*7	Некорректноая строка.

ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ

```
#include <iostream>
#include <vector>
#include <cstring>
#include <math.h>
#include <fstream>
#include <cctype>

class Tree {
public:
  char data;
};

using namespace std;

void buildTree(vector<Tree>&arr, int index, int max, char buffer[], int j) { //
строит дерево на основе массива
  if (index >= max)
    return;
  int stet = 0;
  int k = j;
  if (buffer[j] == '\0')
    return;
  if (buffer[j] == '('){
    j++;
    while (buffer[j] != '*' && buffer[j] != '+' && buffer[j] != '-') {
      while (buffer[j] == '(') {
        stet++;
        j++;
      }
      if (stet == 0)
        j++;
      while (stet>0) {
        if (buffer[j] == ')')
          stet--;
          j++;
      }
    }
  }
  else {
    arr[index].data = buffer[j];
    return;
  }
```

```
  if (buffer[j] == '*' || buffer[j] == '+' || buffer[j] == '-')
    arr[index].data = buffer[j];
  buildTree(arr, 2*index+1, max, buffer, k+1);
  buildTree(arr, 2*(index+1), max, buffer, j+1);
}

void print(vector<Tree>&arr, int index, string &str, int max, int count) { //
печатает дерево
  if (index >= max)
    return;
  print(arr, 2*index+1, str, max, count+1);
  for (int i = 0; i < count; i++)
  str = str + "   ";
  if (arr[index].data != '#')
    str = str + "[" + arr[index].data + "]\n";
  else str = str + "   \n";
  print(arr, 2*(index+1), str, max, count+1);
}

void prefix(vector<Tree>arr, unsigned int index) { //преобразует в
префиксную форму
  if (index>=arr.size())
    return;
  if (arr[index].data != '#')
    cout<<arr[index].data;
  prefix(arr, 2*index+1);
  prefix(arr, 2*(index+1));
}

string change(vector<Tree>arr, int index, string Lskob, string Rskob,string
answer) { // преобразует дерево-формулу в формулу этого дерева
  if (index >= arr.size())
    return answer;
  if (arr[index].data == '#')
    return answer;
  if (arr[index].data == '*' || arr[index].data == '+' || arr[index].data == '-')
    return Lskob + change(arr, 2*index+1, Lskob, Rskob, answer) +
arr[index].data+change(arr, 2*(index+1), Lskob, Rskob, answer) + Rskob;
  else return change(arr, 2*index+1, Lskob, Rskob, answer) + arr[index].data +
change(arr, 2*(index+1), Lskob, Rskob, answer);
}

void right(vector<Tree>&arr, int index) { //раскрывает скобки множителем
пред скобкой
```

```
  char symb = arr[2*index+1].data;
  arr[2*(2*index+1)+1].data = symb;
  arr[2*index+1].data = arr[2*(index+1)].data;
  char copy = arr[4*(index+1)+1].data;
  arr[4*(index+1)+1].data = symb;
  arr[2*(2*index+2)].data = copy;
}

void left(vector<Tree>&arr, int index) { //раскрывает скобки множителем
после скобки
  char symb = arr[2*(index+1)].data;
  arr[2*(2*(index+1)+1)].data = symb;
  arr[2*(index+1)].data = arr[2*index+1].data;
  char copy = arr[2*(2*index+2)].data;
  arr[2*(2*index+2)].data = symb;
  arr[4*(index+1)+1].data = copy;
}

bool test(char buff[]) { // проверка на  некорректные данные
  int count=0, countSign = 0, countSkob = 0;
  if (buff[0]!='(' || buff[strlen(buff)-1]!=')')
    return false;
  for (int i=0;buff[i]!='\0';i++) {
    if (buff[i]=='('){
      count++;
      countSkob++;
    }
    if (buff[i]==')')
      count--;
    if (buff[i]=='/')
     return false;
    if (buff[i]=='(' && buff[i+1]==')')
      return false;
    if (buff[i]==')' && buff[i+1]=='(')
      return false;
    if (buff[i]=='+'||buff[i]=='*'||buff[i]=='-') {
      countSign++;
      if (i>1 && (buff[i-2]=='+'||buff[i-2]=='*'||buff[i-2]=='-'))
        return false;
      if (i<2||i>strlen(buff)-3)
        return false;
      if (!isalnum(buff[i-1]) && !isalnum(buff[i+1]))
         return false;
    }
```

```
    if ((!isalnum(buff[i]) || isupper(buff[i])) && buff[i] != '+' && buff[i] != '-' &&
buff[i] != '*' && buff[i] != '(' && buff[i] != ')')
      return false;
    if (isalnum(buff[i]) && (isalnum(buff[i-1]) || isalnum(buff[i+1])))
      return false;
    if (i<strlen(buff)-4 && buff[i] == '(' && (buff[i+1] == ')' || buff[i+2] == ')' ||
buff[i+3] == ')'))
      return false;
  }
  if (countSign == countSkob && count == 0)
    return true;
  else return false;
}

int main() {

  cout<<"Выберите действие:"<<endl;
  cout<<"1 – Ввести данные вручную."<<endl;
  cout<<"2 – Считать данные из файла."<<endl;
  cout<<"3 – Выйти из программы."<<endl;
  int choose;
  char buffer[100];
  cin>>choose;
  cin.ignore();

  switch (choose) {
    case 1: {
      cout<<" Введите формулу: "<<endl;
      cin.getline(buffer, 100);
      break;
      }
    case 2: {
      ifstream inp("file.txt");
      inp.getline(buffer, 100);
      inp.close();
      cout<<"Введенная формула: "<<buffer<<endl;
      break;
    }
    case 3: {
     return 0;
    }
    default: {
      cout<<"Неправильные входные данные, попробуйте снова."<<endl;
      return 0;
```

```
  }
}

if (!test(buffer)) {
  cout<<"Некорректноая строка"<<endl;
  return 0;
}
if (!strlen(buffer)) {
  cout<<"Пустая строка"<<endl;
  return 0;
}
int ct = 1, max_skob = 0, N, j = 0, minus_flag = 0;
for(int i = 0; buffer[i] != '\0'; i++) {
  if (buffer[i] == '(')
    ct++;
  if (buffer[i] == ')') {
    if (ct > max_skob)
      max_skob = ct;
      ct = 1;
  }
  if (buffer[i]  == '-')
    minus_flag = 1;
}
if (max_skob < 2)
  max_skob = 2;
N = pow(2, max_skob)-1;
vector <Tree> arr(N);
for (int i=0;i<N;i++)
  arr[i].data = '#';
cout<<endl;
buildTree(arr,0,N,buffer,j);
cout<<"Дерево-формула: ";
for(int i= 0; i<N; i++)
  cout<<arr[i].data;
cout<<endl;
string str;
int count = 0;
print(arr,0,str,N,count);
cout<<str;
string Lskob = "(", Rskob = ")", answer = "";
answer += change(arr, 0, Lskob, Rskob, answer);
cout<<"Формула дерева: "<<answer<<endl;
cout<<"Префиксная форма записи дерева: ";
prefix(arr, 0);
```

```
  cout<<endl;

  cout<<"Упрощенная формула: ";
  if (minus_flag == 0) {
    int left_index = 0,right_index = 0;
    for (int i=0; 2*(i+1) < arr.size() && 2*i+1 < arr.size(); i++) {
      if (arr[i].data == '*' && arr[2*i+1].data != '*' && arr[2*i+1].data != '+' &&
arr[2*(i+1)].data == '+') {
        right_index = i;
        right(arr,right_index);
      }
      if(arr[i].data == '*' && arr[2*(i+1)].data != '*' && arr[2*(i+1)].data != '+'
&& arr[2*i+1].data == '+') {
        left_index = i;
        left(arr, left_index);
      }
    }
    string ans = "";
    ans += change(arr, 0, Lskob, Rskob, ans);
    cout<<ans<<endl;
  }
  else cout<<"Входные данные с минусом! "<<buffer<<endl;
  return 0;
}
```