Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Катедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 5 з дисципліни

«Основи програмування 2.

Модульне програмування»

«Дерева»

Варіант 18

Виконав студент ІП-11 Лесів Владислав Ігорович

Перевірив Вітковська Ірина Іванівна

Лабораторна робота 5

Дерева

Мета – вивчити особливості організації і обробки дерев.

Варіант №18.

18. Відповідно до виразу, що читається з текстового файлу, побудувати деревоформулу та обчислити значення цієї формули.

Постановка задачі.

Результатом розв'язку є побудоване дерево-формула класу ExpTree і обчислене значення цієї формули. Вершини дерева представляють собою об'єкти структури TNode. Для визначення результату повинна бути задана формула дерева у текстовому файлі. Інших початкових даних для розв'язку не потрібно.

Математичне формулювання задачі зводиться до знаходження найменших за пріоритетністю операторів і відповідне занесення їх як вузлів дерева згори вниз. Знаходження значення відбуватиметься рекурсивно: спочатку шукаємо значення лівого піддерева, далі правого піддерева, а потім виконуємо між ними операцію, задану у вузлі.

Виконання мовою С++.

ExpTree tree(s);

}

tree.build(tree.root, tree.form);

cout << "\n\nBuilt tree:\n\n";
tree.printTree("", tree.root, false);</pre>

Код програми:

tree.h

```
#pragma once
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>
using namespace std;
struct TNode {
                                                       //Структура вершини
         string inf;
         TNode* left, * right;
};
class ExpTree {
                                                       //Клас дерева
public:
         TNode* root;
         string form;
         ExpTree(string s) { root = new TNode; form = s; }
         ~ExpTree() { delete root; }
         TNode* build(TNode*, string);
         float search(TNode*);
         void printTree(string, TNode*, bool);
};
int* operatorLoop(bool, int,string, char, char);
Lab5_cp.cpp
#include "tree.h"
int main()
{
         ifstream file("input.txt");
         string s;
         getline(file, s);
         file.close();
         cout << "Entered expression: " << s << endl;</pre>
```

cout << "\nCalculated value = "<<tree.search(tree.root) << endl;</pre>

tree.cpp

```
#include "tree.h"
TNode* ExpTree::build(TNode* node,string s) {
                                                                    //Побудова дерева
         int c = 0;
         for (int i = 0; i < s.length(); i++) {</pre>
                   if (isdigit(s[i])==0 && s[i]!='.')
                             c = 1;
         }
         if (c==0){
                                                                                                           //Якщо вираз - число
                   node->inf = s;
                   node->left = NULL;
                   node->right = NULL;
                   return node;
         }
         else {
                   if (s[0] == '(' && s[s.length() - 1] == ')')
                             s = s.substr(1, s.length() - 2);
                   c = 0;
                   TNode* nextLeft = new TNode;
                   TNode* nextRight = new TNode;
                   int* a = operatorLoop(c, 0, s, '+', '-'); //Шукаємо оператори з найменшим пріоритетом
                   int i = a[1];
                                       c = a[0];
                   a = operatorLoop(c, i, s, '*', '/');
                   i = a[1]; c = a[0];
                   a = operatorLoop(c, i, s, '^', '^');
                   i = a[1];
                   node->inf = s[i];
                   node->left = build(nextLeft, s.substr(0, i));
                   node->right = build(nextRight, s.substr(i + 1, s.length() - i - 1));
                   return node;
         }
}
float ExpTree::search(TNode* curr) {
                                                //Обчислюємо вираз
         if (isdigit(curr->inf[0])==0) {
                   float IIn = search(curr->left);
                   float rIn = search(curr->right);
                   switch ((curr->inf)[0]) {
                   case '+':
                             return lln + rln;
                             break;
                   case '-':
                             return IIn - rIn;
                             break;
                   case '*':
                             return lln * rln;
                             break;
                   case '/':
                             return IIn / rIn;
                             break;
                   case '^':
                             return pow(lln, rln);
                   }
         }
         else
```

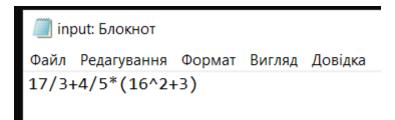
```
return stof(curr->inf);
}
void ExpTree::printTree(string prefix, TNode* node, bool isLeft)
                                                                              //Виводимо дерево на консоль
          if (node != NULL)
         {
                   cout << prefix+"|__";</pre>
                   cout << node->inf << endl;
                    printTree(prefix + (isLeft ? " | " : " "), node->left, true);
                    printTree(prefix + (isLeft ? " | " : " "), node->right, false);
         }
}
int* operatorLoop(bool c,int i,string s, char fir, char sec) {
                                                                             //Шукаємо оператори
         int d = 0;
         if (c == 0) {
                   i = s.length() - 1;
                   while (i != -1) {
                             if (s[i] == ')')
                                       d++;
                             if (s[i] == '(')
                                       d--;
                             if (d==0 \&\& c == 0 \&\& (s[i] == fir || s[i] == sec)) {
                                       c = 1;
                                       break;
                             }
                             i--;
                   }
         int a[2] = { c,i };
          return a;
}
```

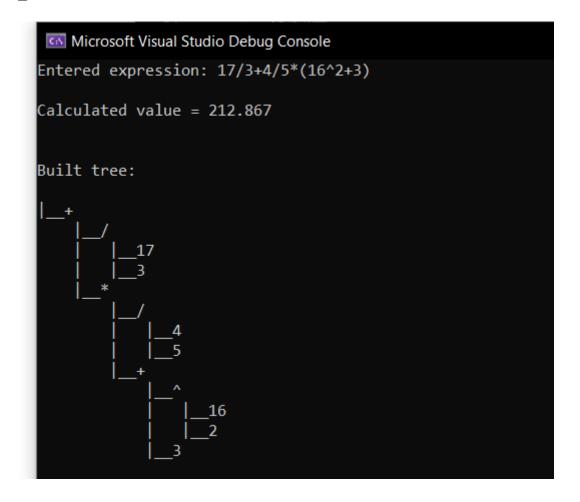
Випробування алгоритму.

1)

```
input: Блокнот
Файл Редагування Формат Вигляд Довідка
2.1-((3+4.3)*4)-(4.5+1.23^2)*2
```

2)





Висновок. Отже, у цій роботі я вивчив особливості організації і обробки дерев. У результаті лабораторної роботи було розроблено програму, яка виконує задачу відповідно до постановки. Використовуючи розроблену структуру вершин дерева, а також клас дерева з атрибутами кореня дерева і його формули, методами побудови, обчислення значення і виведення дерева на консоль, отримуємо коректний результат.