Отчет по лабораторной работе 24.

Студент группы М80-108БВ-24 № по списку 8

Контакты e-mail gubiev.david2007@gmail.com

Работа выполнена: «5» мая 2025 г. Преподаватель: каф. 806 Бучкин Тимур Артемович Входной контроль знаний с оценкой \_ \_ \_

Отчет сдан «5» мая 2025 г., итоговая оценка \_ \_ \_

Подпись преподавателя \_ \_ \_

1. Тема: Составление и обработка деверьев на языке С

2. Цель работы: Редуцировать выражения, заменив операцию возведения переменной в целую степень n и на произведение n слагаемых

3. Задание:

4. Оборудование asus vivobook

5. Программное обеспечение ubuntu

Система программирования: нет. Редактор текстов: VSCode.

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок- схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями):

Идея: найти знак степени, заменить и далее рекурсивно добавлять умножение исходного числа на само себя n-1 раз где n это степень числа.

Оценка сложности алгоритма: O(n) где количество ячеек

7. протокол:

Итоговый алгоритм, уже использующий функции написанные ранее

#include "../include/ExprTree.h"

#include <stdlib.h>

typedef struct Node\_ {

    Token token;

    struct Node\_ \*left;

    struct Node\_ \*right;

    struct Node\_ \*parent;

} Node;

struct ExprTree\_ {

    Node \* root;

};

Node \* Node\_create(Token token, Node \* parent) {

    Node \* node = (Node \*)malloc(sizeof(Node));

    node->token = token;

    node->left = NULL;

    node->right = NULL;

    node->parent = parent;

    return node;

}

Node \* prefixForm2Tree(Tokens \* prefixForm, Node \* parent) {

    if(Tokens\_empty(prefixForm)) {

        return NULL;

    }

    Node \* node = Node\_create(\*Tokens\_top(prefixForm), parent);

    Tokens\_pop(prefixForm);

    if(node->token.type != TOKEN\_OPERATION)

        return node;

    node->right = prefixForm2Tree(prefixForm, node);

    if(node->token.data.op != OP\_UNARY\_MINUS) {

        node->left = prefixForm2Tree(prefixForm, node);

    }

    return node;

}

ExprTree \* TreeCrete(Tokens \* prefixForm) {

    ExprTree \* tree = (ExprTree \*)malloc(sizeof(ExprTree));

    tree->root = prefixForm2Tree(prefixForm, NULL);

    return tree;

}

void TreePrint\_impl(Node \* node, int depth) {

    if(!node)

        return;

    TreePrint\_impl(node->right, depth + 1);

    for(size\_t i = 0; i < depth; ++i)

        printf("->");

    printToken(node->token);

    printf("\n");

    TreePrint\_impl(node->left, depth + 1);

}

void TreePrintExpr\_impl(Node \* node) {

    if(!node) {

        return;

    }

    if(node->left) {

        if(

                node->left->token.type == TOKEN\_OPERATION &&

                getOpPriority(node->token.data.op) > getOpPriority(node->left->token.data.op)

            ) {

            printf("(");

            TreePrintExpr\_impl(node->left);

            printf(")");

        } else {

            TreePrintExpr\_impl(node->left);

        }

    }

    printToken(node->token);

    if(node->right) {

        if(

                node->right->token.type == TOKEN\_OPERATION &&

                getOpPriority(node->token.data.op) > getOpPriority(node->right->token.data.op)

            ) {

            printf("(");

            TreePrintExpr\_impl(node->right);

            printf(")");

        } else {

            TreePrintExpr\_impl(node->right);

        }

    }

}

void GetRoot(ExprTree \* tree) {

    SearchPow(tree->root);

}

void SearchPow(Node \* node){

    if (node->token.data.op == OP\_POW){

        // число под степенью сохроняю это на английском (5^6) Five to the power of six отсюда название переменной

        Token power\_of = node->left->token;

        PowToMult(node, power\_of, node->right->token.data.num, 0);

    }

    else{

        if (node->left)

            SearchPow(node->left);

        if (node->right)

            SearchPow(node->right);

    }

    return;

}

void PowToMult(Node \* node, Token power\_of, int degree, int current){

    if (degree == 1){

        node->token.type = TOKEN\_NUMBER;

        node->token.data.num = power\_of.data.num;

        node->left = NULL;

        node->right = NULL;

    }

    else if (degree == 0){

        node->token.type = TOKEN\_NUMBER;

        node->token.data.num = 1;

        node->left = NULL;

        node->right = NULL;

    }

    if (current < degree - 1){

        node->token.type = TOKEN\_OPERATION;

        node->token.data.op = OP\_MULT;

        node->left = Node\_create(power\_of, node);

        node->right = Node\_create(power\_of, node);

        PowToMult(node->right, power\_of, degree, (current + 1));

    }

    else{

        return;

    }

}

void TreePrint(ExprTree \* tree) {

    TreePrint\_impl(tree->root, 0);

}

void TreePrintExpr(ExprTree \* tree) {

    TreePrintExpr\_impl(tree->root);

}

8. Вывод: В результате работы я улучшил свои навыки работы с деревьями на языке программирования С.

Подпись студента: \_ \_ \_