Отчет по лабораторной работе № 24 по курсу

"Языки и методы программирования"

Студент группы *М8О-102Б-21* *Кажекин Денис Андреевич*, № по списку *9*

Контакты e-mail: deniskazhekin@mail.ru

Работа выполнена: « » 2022 г.

Преподаватель: *доцент каф. 806 Никулин Сергей Петрович*

Входной контроль знаний с оценкой _____

Отчет сдан «» 2022 г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Динамические структуры данных. Обработка деревьев.

2. **Цель работы:** Составить программу выполнения заданных преобразований арифметических выражений с применением деревьев.

3. **Задание:** (*вариант № 23*)**:** Упростить дробь, сократив в числителе и знаменателе общие переменные и константы

4. **Оборудование:**
   Оборудование ПЭВМ студента: Macbook
   Процессор M1 с ОП 16 Гб, SSD 256 Гб.

5. **Программное обеспечение:**
   Программное обеспечение ЭВМ студента:
   Операционная система семейства GNU\Linux, наименование Ubuntu версия 18.04.5
   Система программирования GNU/Linux версия 8.1
   Редактор текстов LibreOfifice Writer, Xcode
   Утилиты gcc cat
   Местонахождение и имена файлов программ и данных /home/denis/Рабочий стол/labs/labs/2sem/24

6. **Идея, метод, алгоритм:**
   Задача выполняется путём поиска одинаковых поддеревьев основного дерева, рассматривая его левую и правую части. Найдя совпадение, удаляем эти деревья, подтягивая соседнее вверх (для каждого из найденных), заменяя предка на брата. Сам предок при этом тоже удаляется. Процедура повторяется пока будут находиться и удаляться одинаковые поддеревья.

7. **Сценарий выполнения работы :**
   gcc main.c -o run && ./run
   тест программы на различных выражениях

Допущен к выполнению работы. **Подпись преподавателя** _____

8. **Распечатка протокола:**

```
deniskazhekin@MacBook-Air-Denis ~/Рабочий стол/labs/labs/2sem/24$ cat Tree.h
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
// (1+a*b)/(c*a-1)
// ((1+a)*b)/(c*(a-1))
// a/(a*b+c)
typedef struct Tree {
    int depth;
    char value;
    struct Tree* left, *right, *root;
} Tree;
Tree* Create_tree(char x) {
    Tree* T = (Tree*)malloc(sizeof(Tree));
    T->value = x; T->left = 0; T->right = 0; T->depth = 0; T->root = T;
    return T;
}
bool is_full(Tree* root) { return root->left != 0 && root->right != 0; }
bool is_empty(Tree* root) { return root->left == 0 && root->right == 0; }
```

```
bool is_liter(char t) { return ('0' <= t && t <= '9') || ('a' <= t && t <= 'z'); }
bool equal(Tree* a, Tree* b) {
    if (a == 0 || b == 0) return a == 0 && b == 0;
    bool t = equal(a->left, b->left) && (a->value == b->value) && equal(a->right, b->right);
    bool e = equal(a->left, b->right) && (a->value == b->value) && equal(a->right, b->left);
    return t || e;
}
void Delete_tree(Tree* root) {
    if (root == 0) return;
    Delete_tree(root->left); Delete_tree(root->right);
    free(root);
}

void set_depths(Tree* root, int d) {
    if (root == 0) return;
    set_depths(root->left, d + 1);
    root->depth = d;
    set_depths(root->right, d + 1);
}

Tree* Buid_tree() {
    char t, was;
    Tree* tree = Create_tree('\0');
    do {
        scanf("%c", &t);
        if (t == '(') {
            if (tree->left == 0) {
                tree->left = Create_tree('\0');
                tree->left->root = tree;
                tree = tree->left;
            } else {
                tree->right = Create_tree('\0');
                tree->right->root = tree;
                tree = tree->right;
            }
        } else if (t == ')' || t == '\n') {
            if (is_liter(was)) {
                tree->right = Create_tree(was);
                tree->right->root = tree;
            }
            while (is_full(tree) && tree != tree->root) tree = tree->root;
        } else if (t == '+' || t == '-') {
            if (tree->left == 0) {
                tree->left = Create_tree(was);
                tree->left->root = tree;
                tree->value = t;
            } else if (tree->value == '\0') tree->value = t;
            else {
                tree->right = Create_tree(was);
                tree->right->root = tree;
                Tree* r = Create_tree(t);
                r->left = tree;
                r->root = tree->root;
                if (tree->root->right == 0) tree->root->left = r;
                else tree->root->right = r;
                tree->root = r;
                tree = r;
            }
        } else if (t == '*' || t == '/') {
            if (tree->left == 0) {
                tree->left = Create_tree(was);
                tree->left->root = tree;
                tree->value = t;
            } else if (tree->value == '\0') tree->value = t;
            else {
                tree->right = Create_tree(t);
                tree->right->root = tree;
                tree = tree->right;
                tree->left = Create_tree(was);
                tree->left->root = tree;
            }
        }
        was = t;
    } while (t != '\n');
    set_depths(tree, 0);
    return tree;
}

void inorder(Tree* tree) {
    if (tree == 0) return;
```

```
    inorder(tree->right);
    for (int i = 0; i < tree->depth; i++) printf("   ");
    printf("%c\n", tree->value);
    inorder(tree->left);
}

void show_equation(Tree* root) {
    if (root == 0) return;
    if (root->left != 0 && (root->value == '/' || (root->value == '*' && (root->left->value == '+'
|| root->left->value == '-'))) && !is_empty(root->left))
        printf("(");
    show_equation(root->left);
    if (root->left != 0 && (root->value == '/' || (root->value == '*' && (root->left->value == '+'
|| root->left->value == '-'))) && !is_empty(root->left))
        printf(")");
    printf("%c", root->value);
    if (root->right != 0 && (root->value == '/' || (root->value == '*' && (root->right->value ==
'+' || root->right->value == '-'))) && !is_empty(root->right))
        printf("(");
    show_equation(root->right);
    if (root->right != 0 && (root->value == '/' || (root->value == '*' && (root->right->value ==
'+' || root->right->value == '-'))) && !is_empty(root->right))
        printf(")");
}
deniskazhekin@MacBook-Air-Denis ~/Рабочий стол/labs/labs/2sem/24$ cat main.c
#include <stdio.h>
#include "Tree.h"
// (a*b)/(c*b*a) -> 1/c
// ((1+a)*b)/(c*(a+1)) -> b/c
// (a*b)/(b*(a+c)) -> a/(a+c)
// (a+b*c)/(b+a*c) -> (a+b*c)/(b+a*c)
// (a*(a+1))/((a-b)*a) -> (a+1)/(a-b)

bool simplify(Tree* left, Tree* right) {
    if (left != 0 && equal(left, right)) {
        Tree* l = left->root;
        if (l->value == '/') left->value = '1';
        else {
            l = (left == l->left) ? l->right : l->left;
            left->root->value = l->value;
            left->root->right = l->right;
            left->root->left = l->left;
            Delete_tree(left);
            free(l);
        }
        Tree* r = right->root;
        if (r->value == '/') right->value = '1';
        else {
            r = (right == r->left) ? r->right : r->left;
            right->root->value = r->value;
            right->root->right = r->right;
            right->root->left = r->left;
            Delete_tree(right);
            free(r);
        }
        return true;
    } else if (left != 0 && right != 0) {
        if (right->value == '*' && left->value == '*')
            return simplify(left->left, right->left) || simplify(left->left, right->right) ||
                   simplify(left->right, right->left) || simplify(left->right, right->right);
        else if (right->value == '*')
            return simplify(left, right->left) || simplify(left, right->right);
        else if (left->value == '*')
            return simplify(left->left, right) || simplify(left->right, right);
    }
    return false;
}

int main() {
    Tree* root = 0;
    printf("Select: 1) Enter equation 2) show equation ");
    printf("3) show tree 4) simplify 5) exit\n");
    int state = 0;
    while (state != 5) {
        printf(">");
        scanf("%d", &state);
        switch (state) {
        case 1:
            root = Buid_tree();
            break;
```

```
        case 2:
            show_equation(root);
            printf("\n");
            break;
        case 3:
            inorder(root);
            break;
        case 4:
            if (root != 0)
                while (simplify(root->left, root->right)) {}
            set_depths(root, 0);
            break;
        }
    }
    return 0;
}
deniskazhekin@MacBook-Air-Denis ~/Рабочий стол/labs/labs/2sem/24$ gcc main.c -o run && ./run
Select: 1) Enter equation 2) show equation 3) show tree 4) simplify 5) exit
>2

>3
>4
>1 (a*b)/(c*b*a)
>2
(a*b)/(c*b*a)
>3
         a
      *
         b
   *
      c
/
      b
   *
      a
>4
>2
1/c
>3
   c
/
   1
>1 ((1+a)*b)/(c*(a+1))
>2
((1+a)*b)/(c*(a+1))
>3
         1
      +
         a
   *
      c
/
      b
   *
         a
      +
         1
>4
>2
b/c
>3
   c
/
   b
>1 (a*b)/(b*(a+c))
>2
(a*b)/(b*(a+c))
>3
         c
      +
         a
   *
      b
/
      b
   *
      a
>4
>2
a/(a+c)
```

```
>3
      c
   +
      a
/
   a
>1 (a+b*c)/(b+a*c)
>3
         c
      *
         a
   +
      b
/
         c
      *
         b
   +
      a
>4
>2
(a+b*c)/(b+a*c)
>1 (a*(a+1))/((a-b)*a)
>3
      a
   *
         b
      -
         a
/
         1
      +
         a
   *
      a
>4
>2
(a+1)/(a-b)
>3
      b
   -
      a
/
      1
   +
      a
>5
deniskazhekin@MacBook-Air-Denis ~/Рабочий стол/labs/labs/2sem/24$
```

9. **Выводы**

В ходе выполнения лабораторной работы научился работать с деревьями выражений в программах, написанных на языке Си.

Подпись студента:_____