Отчет по лабораторной работе №24

по курсу: языки и методы программирования

студент группы : М8О-105Б-21 Козлов Егор Сергеевич, № по списку: 9

Адреса www, e-mail, jabber, skype: iamaghoulzxc@gmail.com

Работа выполнена: "18 мая 2022 г."

Преподаватель: каф. 806 В.К.Титов

Входной контроль знаний с оценкой: _____

Отчет сдан "___"_____20___г., итоговая оценка _____

Подпись преподавателя _____

1. Тема: Алгоритмы и структуры данных.

2. Цель работы: Составить программу на языке Си для выполнения заданных преобразований арифметических выражений с применением деревьев

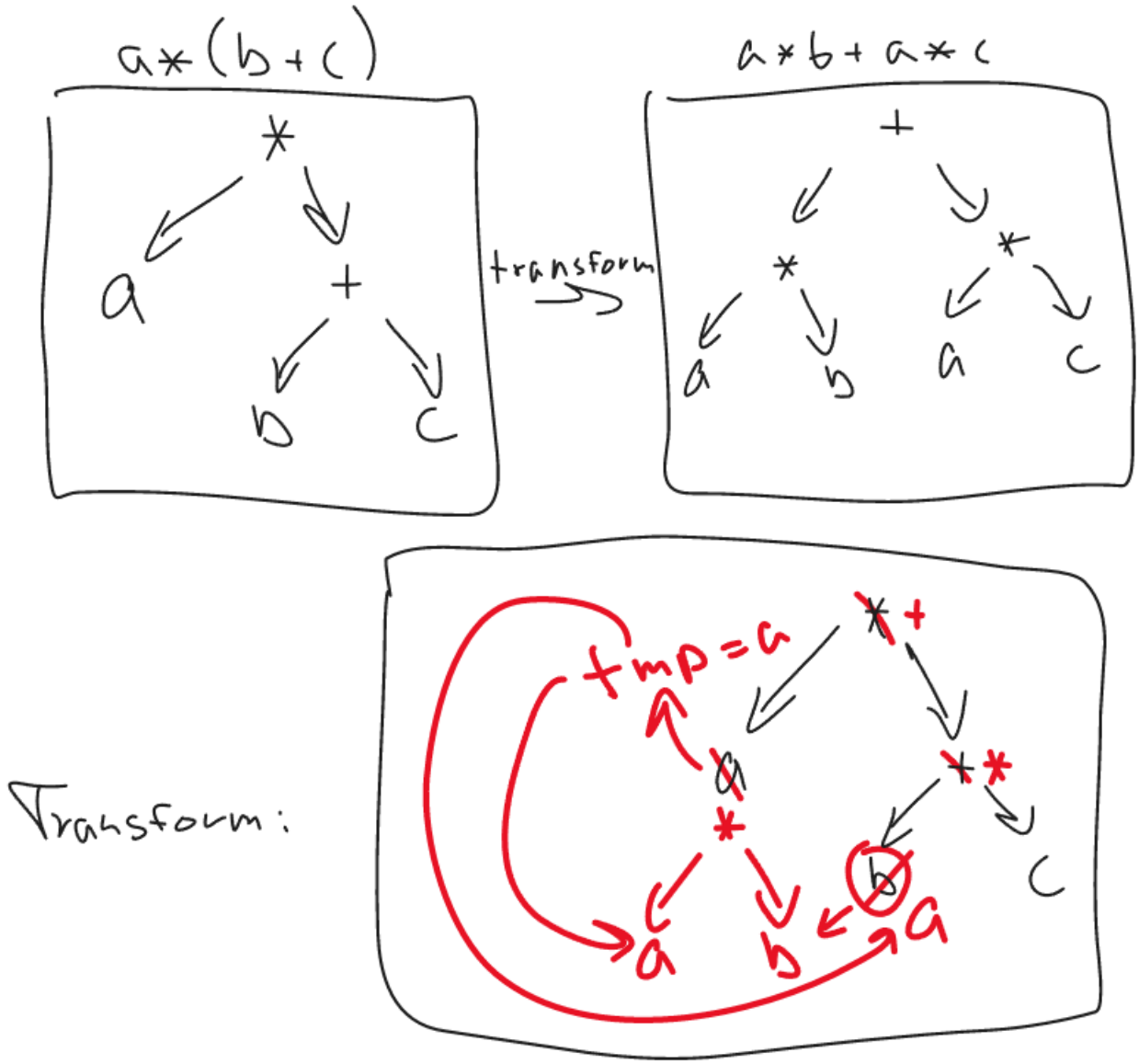
3. Задание (*вариант 9*): Умножение переменной на сумму заменить на сумму произведений

4. Оборудование(*лабораторное*):
ЭВМ _____, процессор _____, имя узла сети ____ с ОП ___ ГБ
НМД _____ ГБ. Терминал _____ адрес _____. Принтер _____
Другие устройства _____
Оборудование ПЭВМ студента, если использовалось:
Процессор Ryzen 7 5800 @ 8x 3.2 GHz , ОП 16384 МБ, НМД _____ГБ. Монитор Встроенный
Другие устройства _____

5. Программное обеспечение(*лабораторное*):
Операционная система семейства UNIX, наименование _____ версия _____
Интерпретатор команд: _____ версия _____
Система программирования: _____ версия _____
Редактор текстов: _____ версия _____
Утилиты операционной системы: _____
Прикладные системы и программы: _____
Местонахождение и имена файлов и программ данных: _____

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства UNIX, наименование Ubuntu версия 22.04
Интерпретатор команд: bash версия _____
Система программирования: C версия _____
Редактор текстов: Emacs версия _____
Утилиты операционной системы: _____
Прикладные системы и программы: _____
Местонахождение и имена файлов и программ данных: /usr/bin , а также /bin

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



Трансформация дерева:

Находим вершину, которая содержит операцию умножения (*). Заменяем операцию умножения (*) в этой вершине операцией сложения (+) Вершина с множителем является правым или левым сыном, сохраняем множитель в временную переменную. Заменяем множитель операцией умножения (*) и добавляем ей два сына. В одного из сыновей кладём множитель. Возвращаемся в корень, идем в другого сына и заменяем операцию сложения (+) операцией умножения (*) и кладем одного из сыновей этого сына в свободного сына предыдущей операции умножения. Вместо этого сына кладем множитель из временной переменной.

7. Сценарий выполнения работы [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

24.*cpp*:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

struct Node;
typedef char treeData;
typedef Node *link;
int i;
char ch;
```

```
10
11 struct Node {
12     treeData data;
13     link left, right;
14 } *tree;
15
16 void printTree(link tree) {
17     static int nodeDeep = 0;
18     ++nodeDeep;
19     if (tree) {
20         printTree(tree->right);
21         for (i = 0; i < nodeDeep; ++i)printf("    ");
22         printf("\\_\'%c\'\n", tree->data);
23         printTree(tree->left);
24     }
25     --nodeDeep;
26 }
27
28 int isAN() { return (ch >= 'a') && (ch <= 'z') || (ch >= '0') && (ch <= '9'); }
29
30 int isCH(char c) { return (c >= 'a') && (c <= 'z') || (c >= '0') && (c <= '9'); }
31
32 link makeNode(char c, link l, link r) {
33     link t = new Node;
34     t->data = c;
35     t->left = l;
36     t->right = r;
37     return t;
38 }
39
40 link expr();
41
42 link fact() {
43     link t;
44     scanf("%c", &ch);
45     if (ch == '(') {
46         t = expr();
47         if (ch != ')') printf("ERROR: not )\n");
48     } else if (isAN()) t = makeNode(ch, 0, 0);
49     else printf("ERROR: not AN\n");
50     return t;
51 }
52
53 link term() {
54     link tm;
55     int done;
56     char ch1;
57     tm = fact();
58     done = 0;
59     while ((ch != '\n') && (!done)) {
60         scanf("%c", &ch);
61         if ((ch == '*') || (ch == '/')) {
62             ch1 = ch;
63             tm = makeNode(ch1, tm, fact());
64         } else done = 1;
65     }
66     return tm;
67 }
68
69 link expr() {
70     link ex;
71     int done;
72     char ch1;
73     ex = term();
74     done = 0;
75     while ((ch != '\n') && (!done)) {
76         if ((ch == '+') || (ch == '-')) {
77             ch1 = ch;
78             ex = makeNode(ch1, ex, term());
79         } else done = 1;
80     }
81     return ex;
82 }
83
84 void treeToExpr(link tree) {
85     if (tree) {
86         if ((tree->data == '+') || (tree->data == '-')) printf("(");
87         treeToExpr(tree->left);
88         printf("%c", tree->data);
```

```
        treeToExpr(tree->right);
        if ((tree->data == '+') || (tree->data == '-')) printf(")");
    }
}

void transformTree(link tree) {
    char multiplier, summation;
    if (tree) {
        if (tree->data == '*') {
            if ((isCH(tree->left->data)) && (tree->left->right == NULL) && (tree->left->left ↙
    ↳ == NULL) &&
                (tree->right->data == '+')) {
                tree->data = '+';
                multiplier = tree->left->data;
                tree->left->data = '*';
                tree->left->left = new Node;
                tree->left->right = new Node;
                summation = tree->right->left->data;
                tree->left->left->data = multiplier;
                tree->left->right->data = summation;
                tree->right->data = '*';
                tree->right->left->data = multiplier;
            } else if ((isCH(tree->right->data)) && (tree->right->right == NULL) && ↙
    ↳ (tree->right->left == NULL) &&
                        (tree->left->data == '+')) {
                tree->data = '+';
                multiplier = tree->right->data;
                tree->right->data = '*';
                tree->right->left = new Node;
                tree->right->right = new Node;
                summation = tree->left->left->data;
                tree->right->left->data = multiplier;
                tree->right->right->data = summation;
                tree->left->data = '*';
                tree->left->left->data = multiplier;
            }
        }
        transformTree(tree->left);
        transformTree(tree->right);
    }
}

int main() {
    printf("Input expression:\n");
    tree = expr();
    printTree(tree);
    printf("\n\n-----------------------\n\n");
    treeToExpr(tree);
    i = 1;
    while (i) {
        i = 0;
        transformTree(tree);
    }
    printf("\n\n-----------------------\n\n");
    printTree(tree);
    printf("\n\n-----------------------\n\n");
    treeToExpr(tree);
    printf("\n\n-----------------------\n\n");
    return 0;
}
```

Пункты 1-7 отчѐта составляются ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя _____

8. Распечатка протокола (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем).

```
isitmuse@isitmuse:~/lab/secondSem/24$ cat head
--------------------------------
|||ЛАБОРАТОРАНЯ   РАБОТА  №24|||
|||||АЛГОРИТМЫ И СТРУКТУРЫ||||||
||||||||||||||ДАННЫХ||||||||||||||
|||ВЫПОЛНИЛ  СТУДЕНТ  ГРУППЫ||||
|||М8О-105Б-21  КОЗЛОВ  ЕГОР||||
\\\\\\\\\\\\\\\\||///////////////
isitmuse@isitmuse:~/lab/secondSem/24$ cat 24.cpp
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

struct Node;
typedef char treeData;
typedef Node *link;
int i;
char ch;

struct Node {
    treeData data;
    link left, right;
} *tree;

void printTree(link tree) {
    static int nodeDeep = 0;
    ++nodeDeep;
    if (tree) {
        printTree(tree->right);
        for (i = 0; i < nodeDeep; ++i)printf("    ");
        printf("\\_\'%c\'\n", tree->data);
        printTree(tree->left);
    }
    --nodeDeep;
}

int isAN() { return (ch >= 'a') && (ch <= 'z') || (ch >= '0') && (ch <= '9'); }

int isCH(char c) { return (c >= 'a') && (c <= 'z') || (c >= '0') && (c <= '9'); }

link makeNode(char c, link l, link r) {
    link t = new Node;
    t->data = c;
    t->left = l;
    t->right = r;
    return t;
}

link expr();

link fact() {
    link t;
    scanf("%c", &ch);
    if (ch == '(') {
        t = expr();
        if (ch != ')') printf("ERROR: not )\n");
    } else if (isAN()) t = makeNode(ch, 0, 0);
    else printf("ERROR: not AN\n");
    return t;
}
```

```
link term() {
    link tm;
    int done;
    char ch1;
    tm = fact();
    done = 0;
    while ((ch != '\n') && (!done)) {
        scanf("%c", &ch);
        if ((ch == '*') || (ch == '/')) {
            ch1 = ch;
            tm = makeNode(ch1, tm, fact());
        } else done = 1;
    }
    return tm;
}

link expr() {
    link ex;
    int done;
    char ch1;
    ex = term();
    done = 0;
    while ((ch != '\n') && (!done)) {
        if ((ch == '+') || (ch == '-')) {
            ch1 = ch;
            ex = makeNode(ch1, ex, term());
        } else done = 1;
    }
    return ex;
}

void treeToExpr(link tree) {
    if (tree) {
        if ((tree->data == '+') || (tree->data == '-')) printf("(");
        treeToExpr(tree->left);
        printf("%c", tree->data);
        treeToExpr(tree->right);
        if ((tree->data == '+') || (tree->data == '-')) printf(")");
    }
}

void transformTree(link tree) {
    char multiplier, summation;
    if (tree) {
        if (tree->data == '*') {
            if ((isCH(tree->left->data)) && (tree->left->right == NULL) && (tree->left->left == NULL) &&
                (tree->right->data == '+')) {
                tree->data = '+';
                multiplier = tree->left->data;
                tree->left->data = '*';
                tree->left->left = new Node;
                tree->left->right = new Node;
                summation = tree->right->left->data;
                tree->left->left->data = multiplier;
                tree->left->right->data = summation;
                tree->right->data = '*';
                tree->right->left->data = multiplier;
            } else if ((isCH(tree->right->data)) && (tree->right->right == NULL) && (tree->right->left
                       (tree->left->data == '+')) {
                tree->data = '+';
                multiplier = tree->right->data;
                tree->right->data = '*';
```

```
                tree->right->left = new Node;
                tree->right->right = new Node;
                summation = tree->left->left->data;
                tree->right->left->data = multiplier;
                tree->right->right->data = summation;
                tree->left->data = '*';
                tree->left->left->data = multiplier;
            }
        }
        transformTree(tree->left);
        transformTree(tree->right);
    }
}

int main() {
    printf("Input expression:\n");
    tree = expr();
    printTree(tree);
    printf("\n\n----------------------\n\n");
    treeToExpr(tree);
    i = 1;
    while (i) {
        i = 0;
        transformTree(tree);
    }
    printf("\n\n----------------------\n\n");
    printTree(tree);
    printf("\n\n----------------------\n\n");
    treeToExpr(tree);
    printf("\n\n----------------------\n\n");
    return 0;
}isitmuse@isitmuse:~/lab/secondSem/24$ g++ -o 24 24.cpp
isitmuse@isitmuse:~/lab/secondSem/24$ ./24
Input expression:
a*(b+c)
            \_'c'
        \_'+'
            \_'b'
    \_'*'
        \_'a'


----------------------

a*(b+c)

----------------------

            \_'c'
        \_'*'
            \_'a'
    \_'+'
            \_'b'
        \_'*'
            \_'a'


----------------------

(a*b+a*c)

----------------------
```

```
isitmuse@isitmuse:~/lab/secondSem/24$ ./24
Input expression:
a*(3+d)-2*(5+7)+d*(a+7)
                \_'7'
            \_'+'
                \_'a'
        \_'*'
            \_'d'
    \_'+'
                    \_'7'
                \_'+'
                    \_'5'
            \_'*'
                \_'2'
        \_'-'
                    \_'d'
                \_'+'
                    \_'3'
            \_'*'
                \_'a'


----------------------

((a*(3+d)-2*(5+7))+d*(a+7))

----------------------

                \_'7'
            \_'*'
                \_'d'
        \_'+'
                \_'a'
            \_'*'
                \_'d'
    \_'+'
                    \_'7'
                \_'*'
                    \_'2'
            \_'+'
                    \_'5'
                \_'*'
                    \_'2'
        \_'-'
                    \_'d'
                \_'*'
                    \_'a'
            \_'+'
                    \_'3'
                \_'*'
                    \_'a'


----------------------

(((a*3+a*d)-(2*5+2*7))+(d*a+d*7))

----------------------

isitmuse@isitmuse:~/lab/secondSem/24$ ./24
Input expression:
5*(3+7)-d*(a+7)-f*(g+h)+e*(5+e)
                \_'e'
```

```
            \_'+'
                \_'5'
        \_'*'
            \_'e'
    \_'+'
                    \_'h'
                \_'+'
                    \_'g'
            \_'*'
                \_'f'
        \_'-'
                        \_'7'
                    \_'+'
                        \_'a'
                \_'*'
                    \_'d'
            \_'-'
                        \_'7'
                    \_'+'
                        \_'3'
                \_'*'
                    \_'5'
```

```
(((5*(3+7)-d*(a+7))-f*(g+h))+e*(5+e))
```

```
                \_'e'
            \_'*'
                \_'e'
        \_'+'
                \_'5'
            \_'*'
                \_'e'
    \_'+'
                    \_'h'
                \_'*'
                    \_'f'
            \_'+'
                    \_'g'
                \_'*'
                    \_'f'
        \_'-'
                        \_'7'
                    \_'*'
                        \_'d'
                \_'+'
                        \_'a'
                    \_'*'
                        \_'d'
            \_'-'
                        \_'7'
                    \_'*'
                        \_'5'
                \_'+'
                        \_'3'
                    \_'*'
                        \_'5'
```

```
-----------------------

((((5*3+5*7)-(d*a+d*7))-(f*g+f*h))+(e*5+e*e))

-----------------------

isitmuse@isitmuse:~/lab/secondSem/24$
```

9. Дневник отладки должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечание автора по существу работы _____

_____

11. Выводы Я научился работать с арифметическими преобразованиями при помощи деревьев.

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом _____

_____

Подпись студента _____