



Отчет по лабораторной работе №24 по курсу Языки и методы программирования _____

Студент группы М8О-101Б-21 Постнов Александр Вячеславович, № по списку 17

Контакты www, e-mail: 61pav03@mail.ru

Работа выполнена: «» 2022г.

Преподаватель: ____ каф. 806 _____ Титов В.К. _____

Входной контроль знаний с оценкой _____

Отчет сдан « » _____ 2022_ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Дерево выражений

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

3. **Задание (вариант №17):** Перемножить степени с одинаковыми степенями(в простейшем случае можно рассматривать основания, состоящие из одной переменной или константы).
 $a^2 * a^k \rightarrow a^{(2+k)}$

4. **Оборудование(лабораторное):**

ЭВМ -, процессор -, имя узла сети - с ОП - ГБ,

НМД - ГБ, терминал- адрес -, принтер -

Другие устройства -

Оборудование ПЭВМ студента, если использовалось:

Процессор AMD Ryzen 5 4500U, с ОП 8 ГБ

Другие устройства -

5. **Программное обеспечение:**

Операционная система семейства -, наименование - версия -
интерпретатор команд - версия -

Система программирования - версия -

Редактор текстов - версия -

Утилиты операционной системы -

Прикладные системы и программы -

Местонахождение и имена файлов программ и данных -

Программное обеспечение ЭВМ студента, если использовалось:

Операционная система семейства GNU/Linux, наименование Manjaro версия 5-13-12-1

интерпретатор команд GNOME Terminal версия 3.38.2.

Система программирования _____ версия _____

Редактор текстов emacs версия 3.27.20

Утилиты операционной системы

Прикладные системы и программы -

Местонахождение и имена файлов программ и данных -

6. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок,

Построение дерева выражений по заданному выражению реализовано методом Рутисхаузера.
алгоритм Рутисхаузера:

Один из наиболее ранних алгоритмов.

Предполагает полную скобочную структуру выражения – такую форму записи, при которой порядок действий задается расстановкой скобок.

Неявное старшинство операций при этом не учитывается. Например:

$$D = ((C-(B*L))+K)$$

Обработывая выражение с полной скобочной структурой, алгоритм присваивает каждому символу – лексеме исходного выражения – номер уровня по следующему правилу:

- 1) если это открывающаяся скобка или переменная, то значение уровня увеличивается на 1;
- 2) если лексема - знак операции или закрывающаяся скобка, то уровень уменьшается на 1.

Дерево по заданному алгебраическому выражению строится вызовом функции `link expr()`, результатом которой является адрес корня построенного дерева.

В обратную сторону, по дереву выражения создаётся выражение функцией:

`void tree2expr(link tree).`

Алгоритм преобразования по моему варианту:

- 1) прохожу по дереву стандартным алгоритмом
- 2) если нахожу узел, значения вершины которой == "*" и он указывает на "^" слева и справа, то проверяем этот случай:
 - 1) нужно проверить, что основания степеней одинаковые и они являются терминальными вершинами(т.е. никуда не указывают). Если какое-то условие не выполняется, то этот случай не подходит
 - 2) заменяем значение рассматриваемой вершины на "^"; будет указывать слева на основание(терминальная вершина), справа будет указывать на сумму показателей.

Работа с деревом будет реализована с помощью меню;

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

```
#include <stdio.h>

#include <stdlib.h>

typedef char tdata;

int i;

char ch;
```

```

struct node;

typedef node *link;

struct node {
    tdata data;
    link left, right;
} * tree;

void printtree(link t) {
    static int l = 0;
    l++;
    if (t) {
        printtree(t->right);
        for (i = 0; i < l; i++)
            printf("    ");
        printf("\\__%c\\n", t->data);
        printtree(t->left);
    }
    l--;
} // printtree-----

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

int isN(char c) { return (c >= '0') && (c <= '9'); }

link mknode(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 = mknode(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 == '*') || (ch == '/')) {  
            ch1 = ch;  
            tm = mknode(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 = mknode(ch1, ex, term());
    } else
        done = 1;
}
return ex;
}

void tree2expr(link tree) {
    if (tree) {
        if ((tree->data == '+') || (tree->data == '-'))
            printf("(");
        tree2expr(tree->left);
        printf("%c", tree->data);
        tree2expr(tree->right);
        if ((tree->data == '+') || (tree->data == '-'))
            printf(")");
    }
}

void trans_tree(link tree) { //
    if (tree) {
        if (tree->data == '*')
            if (tree->left->data == '^' && tree->right->data == '^') { //будем упрощать
только простые случаи
                link l = tree->left; //переходим к узлу со значением ^
                link r = tree->right;

```

```

    link base1 = l->left; //переход к узлу с основанием степени
    link base2 = r->left;

    if (base1->data == base2->data && !base1->left && !base1->right &&
!base2->left && !base2->right) { //проверка на то основание одинаковое и узлы с
ономанием терминальные

        tree->data = '^';
        tree->left = base1;

        link plus = new node; //создаем новый узел для сложения показателей степени
        plus->data = '+';
        plus->left = l->right;
        plus->right = r->right;
        tree->right = plus;
        i = 1;
    }
}

trans_tree(tree->left);
trans_tree(tree->right);
}
}

int main() {
    int k = 0;
    printf("Menu:\n"
"1) Enter the expression\n"
"2) Output the expression tree\n"
"3) Output an expression from the expression tree\n"
"4) Main action\n"
"5) Menu\n"
"6) Exit \n"
);
    tree = 0;
    for(;;) {
        printf("Input the number of menu: ");
        scanf("%d", &k);

        char c = 0; //съесть символ перехода

```

```
c = getchar();

if (k == 1) {
    printf("Input expression:\n");
    tree = 0;
    tree = expr();
}

else if (k == 2) {
    if (tree) {
        printtree(tree);
        printf("\n");
    }
    else {
        printf("The tree is empty\n");
    }
}

else if (k == 3) {
    if (tree) {
        tree2expr(tree);
        printf("\n");
    }
    else {
        printf("The tree is empty\n");
    }
}

else if (k == 4) {
    if (tree) {
        i = 1;
        while (i) {
            i = 0;
            trans_tree(tree);
        }
    }
    else {
        printf("The tree is empty\n");
    }
}
```

```

    }

}

else if (k == 5) {

    printf("Menu:\n"

    "1) Enter the expression\n"

    "2) Output the expression tree\n"

    "3) Output an expression from the expression tree\n"

    "4) Main action\n"

    "5) Menu\n"

    "6) Exit\n"

    );

}

else if (k == 6) {

    break;

}

else {

    printf("There is no such menu item.\n");

}

}

return 0;

}

```

Тестирование:

- 1) $a^k * a^2 \rightarrow a^{(2+k)}$ (пример из методички)
- 2) $6+2+4*7+5+2+(5^k)*(6^k) \rightarrow$ то же самое (так как основания разные)
- 3) $6+(2^3)*(2^{(3*2+4)}) \rightarrow 6+2^{(3+3*2+4)}$
- 4) $(5^{(7+8)})*(5^{(5*3)})+(4^4)*(4^9) \rightarrow (5^{(7+8+5*3)})+(4^{(4+9)})$

Пункты 1-7 отчета составляются строго до начала лабораторной работы.

Допущен к выполнению работы. Подпись преподавателя _____

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

~/P/mai_labs/2/24(не делал) main !2 ?7 cat head.txt ✓

	Лабораторная работа №24	
	Дерево выражений	
	Выполнил: студент группы М8О-101Б-21	
	Постнов Александр Вячеславович	

~/P/mai_labs/2/24(не делал) main !2 ?7 g++ main.cpp ✓

~/P/mai_labs/2/24(не делал) main !2 ?7 ./a.out ✓

Menu:

- 1) Enter the expression
- 2) Output the expression tree
- 3) Output an expression from the expression tree
- 4) Main action
- 5) Menu
- 6) Exit

Input the number of menu: 1

Input expression:

$(a^k) \cdot (a^2)$

Input the number of menu: 2

```
  \_2
  \_^
  \_a
 \_*
  \_k
  \_^
  \_a
```

Input the number of menu: 3

$a^k \cdot a^2$

Input the number of menu: 4

Input the number of menu: 3

$a^{(k+2)}$

Input the number of menu: 2

```
  \_2
  \_+
  \_k
 \_^
  \_a
```

Input the number of menu: 1

Input expression:

$6+2+4*7+5+2+(5^k) \cdot (6^k)$

Input the number of menu: 2

```
  \_k
  \_^
  \_6
 \_*
  \_k
  \_^
  \_5
 \_+
  \_2
  \_+
```

$$\sqrt[5]{\sqrt[7]{\sqrt[4]{\sqrt[2]{\sqrt[6]{x}}}}}$$

Input the number of menu: 3

$$((((6+2)+4*7)+5)+2)+5^k*6^k)$$

Input the number of menu: 4

Input the number of menu: 3

$$((((6+2)+4*7)+5)+2)+5^k*6^k)$$

Input the number of menu: 1

Input expression:

$$6+(2^3)*(2^{(3+3*2+4)})$$

Input the number of menu: 2

$$\begin{array}{l} \backslash_4 \\ \backslash_+ \\ \backslash_2 \\ \backslash_* \\ \backslash_3 \\ \backslash_+ \\ \backslash_3 \\ \backslash^{\wedge} \\ \backslash_2 \\ \backslash_* \\ \backslash_3 \\ \backslash^{\wedge} \\ \backslash_2 \\ \backslash_+ \\ \backslash_6 \end{array}$$

Input the number of menu: 3

$$(6+2^3*2^((3+3*2)+4))$$

Input the number of menu: 4

Input the number of menu: 2

$$\begin{array}{l} \setminus_4 \\ \setminus_+ \\ \setminus_2 \\ \setminus_* \\ \setminus_3 \\ \setminus_+ \\ \setminus_3 \\ \setminus_+ \\ \setminus_3 \\ \setminus^{\wedge} \\ \setminus_2 \\ \setminus_+ \\ \setminus_6 \end{array}$$

Input the number of menu: 3

Input the number of menu: 2

$$\begin{array}{l} \setminus_9 \\ \setminus^{\wedge} \\ \setminus_4 \\ \setminus^* \\ \setminus_4 \\ \setminus^{\wedge} \\ \setminus_4 \\ \setminus^+ \\ \setminus_3 \\ \setminus^* \\ \setminus_5 \\ \setminus^{\wedge} \\ \setminus_5 \\ \setminus^* \\ \setminus_8 \\ \setminus^+ \\ \setminus_7 \\ \setminus^{\wedge} \\ \setminus_5 \end{array}$$

Input the number of menu: 2

$$\begin{array}{c} \setminus 9 \\ \setminus + \\ \setminus 4 \\ \setminus ^ \\ \setminus 4 \\ \setminus + \\ \setminus 3 \\ \setminus * \\ \setminus 5 \\ \setminus + \\ \setminus 8 \\ \setminus + \\ \setminus 7 \\ \setminus ^ \\ \setminus 5 \end{array}$$

Input the number of menu: 6

9. Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) 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
1	дом	5.05	14:00	в некоторых пунктах меню, не проверял, есть ли вообще дерево, и из-за возникала ошибка	проверял, что дерево выражений существует.	

10. Замечания автора

11. Выводы

В ходе лабораторной работы я изучил как строятся деревья выражений, познакомился с алгоритмом Рутисхаузера

Недочёты при выполнении задания могут быть устранены следующим образом:

Подпись студента ____ Постнов _____