	Отчет по лабораторной работе №24 по курсу				
	Языки и методы программирования				
	Студент группы <u>М8О-101Б-21 Постнов Александр Вячеславович,</u> № по списку <u>17</u>				
	Контакты www, e-mail: 61pav03@mail.ru				
	Работа выполнена: «» 202 <u>2</u> г.				
	Преподаватель: каф. 806 <u>Титов В.К.</u>				
	Входной контроль знаний с оценкой				
	Отчет сдан « » 2022_ г., итоговая оценка				
	Подпись преподавателя				
. То	ема: Дерево выражений				
	Цель работы: Составить программу выполнения заданных преобразований арифмитеческих выражений с применением деревьев.				
pa	Задание (вариант №17): Перемножить степени с одинаковыми степенями(в простейшем случае можно рассматривать основания, состоящие из одной переменной или константы). $a^2 * a^k -> a^2(2+k)$				
ЭЕ Н	борудование(лабораторное): ВМ _, процессор _, имя узла сети _ с ОП _ ГБ, МД _ ГБ, терминал- адрес _, принтер _ ругие устройства _				
П	борудование ПЭВМ студента, если использовалось: роцессор AMD Ryzen 5 4500U, с ОП 8 ГБ ругие устройства <u>-</u>				
О: ИI Ст Ре У: П]	рограммное обеспечение: перационная система семейства _, наименование _ версия _ нтерпретатор команд _ версия истема программирования _ версия _ едактор текстов _ версия _ гилиты операционной системы _ рикладные системы и программы _ естонахождение и имена файлов программ и данных _				
Оп ин Си Ре, Ут	пограммное обеспечение ЭВМ студента, если использовалось: перационная система семейства GNU/Linux, наименование Manjaro версия 5-13-12-1 перпретатор команд GNOME Terminal версия 3.38.2. пстема программирования				

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)
```

Обрабатывая 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;
1++;
if (t) {
 printtree(t->right);
 for (i = 0; i < 1; i++)
   printf(" ");
  printf("\\__%c\n", t->data);
 printtree(t->left);
}
1--;
} // printtree------
int isAN() { return (ch >= 'a') && (ch <= 'z') || (ch >= '0') && (ch <= '9'); }</pre>
int isN(char c) { return (c >= '0') && (c <= '9'); }</pre>
link mknode(char c, link l, link r) {
link t = new node;
t->data = c;
 t->left = 1;
 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 ^{\circ}
      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
онованием 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; // cъесть 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->a^(2+k) (пример из методички)
- 2) 6+2+4*7+5+2+(5^k)*(6^k) ->то же самое(так как основания разные)
- 3) 6+(2^3)*(2^(3*2+4))->6+2^(3+3*2+4)
- 4) (5^(7+8))*(5^(5*3))+(4^4)*(4^9)->(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)*(a^2)
Input the number of menu: 2
Input the number of menu: 3
a^k*a^2
Input the number of menu: 4
Input the number of menu: 3
a^(k+2)
Input the number of menu: 2
Input the number of menu: 1
Input expression:
6+2+4*7+5+2+(5^k)*(6^k)
Input the number of menu: 2
```



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

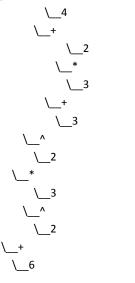
(((((6+2)+4*7)+5)+2)+5^k*6^k)

Input the number of menu: 1

Input expression:

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

Input the number of menu: 2

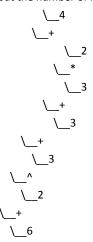


Input the number of menu: 3

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

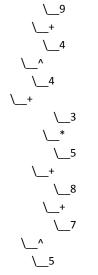
Input the number of menu: 4

Input the number of menu: 2



Input the number of menu: 3

Input the number of menu: 3 (5^(7+8)*5^5*3+4^4*4^9) Input the number of menu: 4 Input the number of menu: 2



Input the number of menu: 3 (5^((7+8)+5*3)+4^(4+9))
Input the number of menu: 0
There is no such menu item.
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.	Дата	Время	Событие	Действие по исправлению	Примечание
	или					
	дом.					
1	дом	5.05	14:00	в некоторых пунктах	проверял, что дерево	
				меню, не проверял, есть	выражений существует.	
				ли вообще дерево, и		
				из-за возникала ошибка		

10. Замечания автора

оды	
лабораторной работы я изучил как строятся деревья выражений, познакомился с алгоритмом	1 Рутисхауз
едочёты при выполнении задания могут быть устранены следующим образом:	

Подпись студента ____Постнов_____