

Отчет по Лабораторной работе № 24

по курсу: 1 фундаментальная информатика

студент группы : М8О-105Б-21 Титеев Рамиль Маратович , № по списку: 23

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

Работа выполнена: "2 мая 2022г"

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 выполнения заданных преобразований арифметических выражений с применением деревьев.

3. Задание(*вариант 5*): Упростить выражения, выполнить возведение числа в степень с положительным показателем.

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 4600 @ 6x 3.0 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 20.04

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)

Описание трансформации дерева:

Находим вершину, которая содержит символ операции(+/-/*...). Затем берем значения слева и справа от этой вершины и считаем его. После записываем его вместо вершины, в которой раньше находился символ операции. Однако если результат оказался больше 10 или меньше 0, то изменения не происходят.

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

lab_24.cpp:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.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--;
}

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

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

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

link fact();
link term();
link power();
link expr();
link power(){
```

```

link pw;
int done;
char ch1;
pw=term();
done=0;
while((ch!='\n')&&(!done)){
    ch = getchar();
    // scanf("%c",&ch);
    if((ch=='^')){
        ch1=ch;
        pw=mknnode(ch1,pw,term());
    }
    else if((ch=='*')||(ch=='/')){
        ch1=ch;
        pw=mknnode(ch1,pw, power());
        done=1;
    }
    else done=1;
}
return pw;
}

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

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

link expr(){
link ex;
int done;

```

```

char ch1;
ex=power();
done=0;
while((ch!='\n')&&(!done)){
    if((ch=='*')||(ch=='/')){
        ch1=ch;
        ex=mknode(ch1,ex,power());
    }
    else if((ch=='+')||(ch=='-')){
        ch1=ch;
        ex=mknode(ch1,ex,power());
    }
    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 transtree(link tree){
    char c, cl, cr;
    if(tree){
        if(tree->data=='^'){
            cl=tree->left->data;
            cr=tree->right->data;
            if(isN(cl)&&isN(cr)){
                c = pow(cl-'0',cr-'0');
                if(c<10 && c>0){
                    tree->data=c+'0';
                    tree->left=0;
                    tree->right=0;
                    i=1;
                }
            }
        }
        if(tree->data=='+'){
            cl=tree->left->data;
            cr=tree->right->data;
            if(isN(cl)&&isN(cr)){
                c = (cl-'0') + (cr-'0');
                if(c<10){
                    tree->data=c+'0';
                    tree->left=0;
                    tree->right=0;
                    i=1;
                }
            }
        }
    }
}

```

```

    }
}
if(tree->data=='-'){
    cl=tree->left->data;
    cr=tree->right->data;
    if(isN(cl)&&isN(cr)){
        c = (cl-'0') - (cr-'0');
        if(c<10){
            tree->data=c+'0';
            tree->left=0;
            tree->right=0;
            i=1;
        }
    }
}
if(tree->data=='/'){
    cl=tree->left->data;
    cr=tree->right->data;
    if(isN(cl)&&isN(cr)){
        c = (cl-'0') / (cr-'0');
        if(c<10){
            tree->data=c+'0';
            tree->left=0;
            tree->right=0;
            i=1;
        }
    }
}
if(tree->data=='*'){
    cl=tree->left->data;
    cr=tree->right->data;
    if(isN(cl)&&isN(cr)){
        c = (cl-'0') * (cr-'0');
        if(c<10){
            tree->data=c+'0';
            tree->left=0;
            tree->right=0;
            i=1;
        }
    }
}
transtree(tree->left);
transtree(tree->right);
}
}

int main(){
    int k=1;
    printf("-----\n");
    printf("Input expression:\n\n");

    tree=expr();
    printf("-----\n");
    while(k){

```

```

printf("\n    MENU\n"
       " 0 - exit\n"
       " 1 - print tree\n"
       " 2 - tree to expression\n"
       " 3 - transform tree\n");
printf("==> ");
scanf("%d",&k);
printf("-----\n");
if(!k) break;
if(k==1){
    if(tree){
        printtree(tree);
    }
    else
        printf("Tree is empty\n");
    printf("-----\n");
}
if(k==2){
    tree2expr(tree);
    printf("\n-----\n");
}
if(k==3){
    i=1;
    while(i){
        i=0;
        transtree(tree);
    }
}
}
return 0;
}

```

Пункты 1-7 отчёта 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).

```

constantfear@constantfear:~/projects/laboratory/2_semester/lab_24$ cat header.txt
*****
*                Лабораторная работа №24                *
*                Алгоритмы и структуры данных.            *
*                Выполнил студент гр. М80-105-Б            *
*                Титеев Рамиль Маратович                  *
*****
constantfear@constantfear:~/projects/laboratory/2_semester/lab_24$ g++ lab_24.cpp
constantfear@constantfear:~/projects/laboratory/2_semester/lab_24$ ./a.out
-----
Input expression:

2+2^2-1*4+3^2/3
-----

    MENU
0 - exit
1 - print tree
2 - tree to expression

```

3 - transform tree

==> 1

```
-----
      \_3
     \_/_
      \_2
     \_~
      \_3
     \_+
      \_4
     \_*
      \_1
     \_-
      \_2
     \_~
      \_2
     \_+
      \_2
-----
```

MENU

- 0 - exit
- 1 - print tree
- 2 - tree to expression
- 3 - transform tree

==> 2

(((2+2^2)-1*4)+3^2/3)

MENU

- 0 - exit
- 1 - print tree
- 2 - tree to expression
- 3 - transform tree

==> 3

MENU

- 0 - exit
- 1 - print tree
- 2 - tree to expression
- 3 - transform tree

==> 2

5

MENU

- 0 - exit
- 1 - print tree
- 2 - tree to expression
- 3 - transform tree

==> 1

_5

MENU

- 0 - exit
- 1 - print tree
- 2 - tree to expression
- 3 - transform tree

```

==> 0
-----
constantfear@constantfear:~/projects/laboratory/2_semester/lab_24$ ./a.out
-----
Input expression:

3+2+4^2-3*4
-----

      MENU
0 - exit
1 - print tree
2 - tree to expression
3 - transform tree
==> 1
-----
      \__4
     \__*
    \__\__3
   \__-
      \__2
     \__^
    \__\__4
   \__+
      \__2
     \__+
    \__\__3
-----

      MENU
0 - exit
1 - print tree
2 - tree to expression
3 - transform tree
==> 2
-----
(((3+2)+4^2)-3*4)
-----

      MENU
0 - exit
1 - print tree
2 - tree to expression
3 - transform tree
==> 3
-----

      MENU
0 - exit
1 - print tree
2 - tree to expression
3 - transform tree
==> 2
-----
((5+4^2)-3*4)
-----

      MENU
0 - exit
1 - print tree
2 - tree to expression
3 - transform tree
==> 1

```



```

-----
      \_4
     \_ *
    \_ \_3
   \_ -
      \_2
     \_ ^
    \_ \_4
   \_ +
      \_5
-----

```

```

-----
MENU
0 - exit
1 - print tree
2 - tree to expression
3 - transform tree
==> 0
-----

```

constantfear@constantfear:~/projects/laboratory/2_semester/lab_24\$./a.out

Input expression:

3*3-4+2*(4-1)^2-8/2^2

```

-----
MENU
0 - exit
1 - print tree
2 - tree to expression
3 - transform tree
==> 1
-----

```

```

      \_2
     \_ ^
    \_ \_2
   \_ /
  \_ \_8
 \_ -
      \_2
     \_ ^
      \_1
     \_ -
      \_4
   \_ *
  \_ \_2
 \_ +
   \_4
  \_ -
      \_3
     \_ *
      \_3
-----

```

```

-----
MENU
0 - exit
1 - print tree
2 - tree to expression
3 - transform tree
==> 2
-----

```

((3*3-4)+2*(4-1)^2)-8/2^2)

