Отчёт по лабораторной работе № 24 по курсу Языки и методы программирования

Студент группы М8О-104Б-19 Черница Артём Александрович, № по списку 22

Контакты www, e-mail, icq, skype aachernitsa@mai.education

Работа выполнена: « 28 » мая 20 20 г.

Преподаватель: Титов В.К. каф.806 Вычислительная математика и программирование

Входной контроль знаний с оценкой _____

Отчёт сдан « » _____201 __ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Деревья выражений.

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

3. **Задание** (*вариант №* 22): Упростить выражение для операций умножения и вычитания для рациональных дробей вида 1/5, выражение вида 1/3+1/6-1/4.

4. **Оборудование(лабораторное)**:
ЭВМ _____, процессор _____, имя узла сети_____ с ОП _____ Мб,
НМД_____Мб. Терминал_____адрес _____. Принтер _____
Другие устройства _____

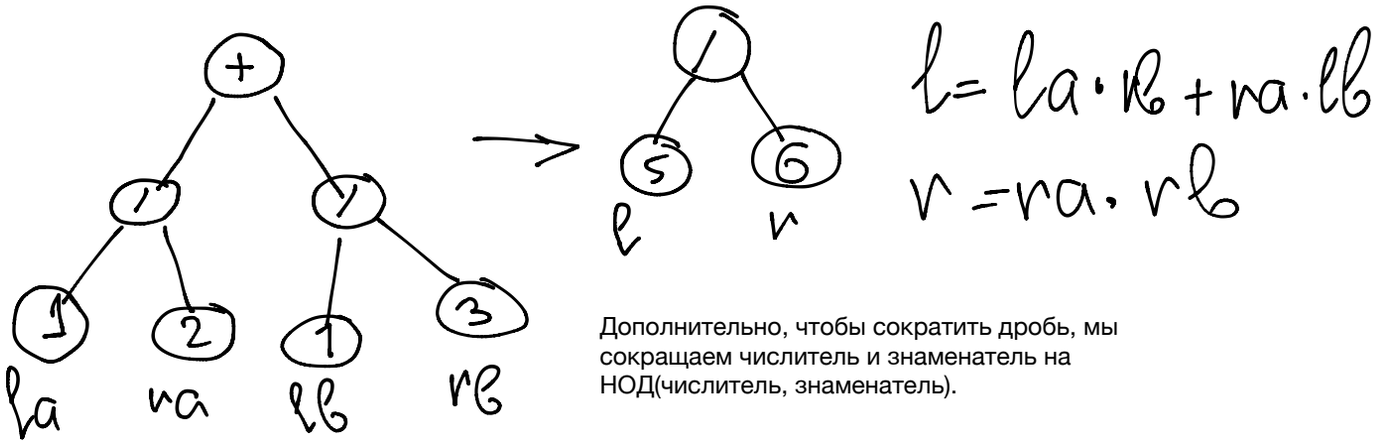
Оборудование ПЭВМ студента, если использовалось:
Процессор Intel Core i5 1.6GHz с ОП 8192 Мб, НМД 128000 Мб. Монитор _____
Другие устройства _____

5. **Программное обеспечение(лабораторное):**
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 MacOS, наименование Catalina версия 10.15.4
интерпретатор команд zsh версия _____.
Система программирования C версия _____
Редактор текстов vim версия _____
Утилиты операционной системы g++, cat
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных на домашнем компьютере_____

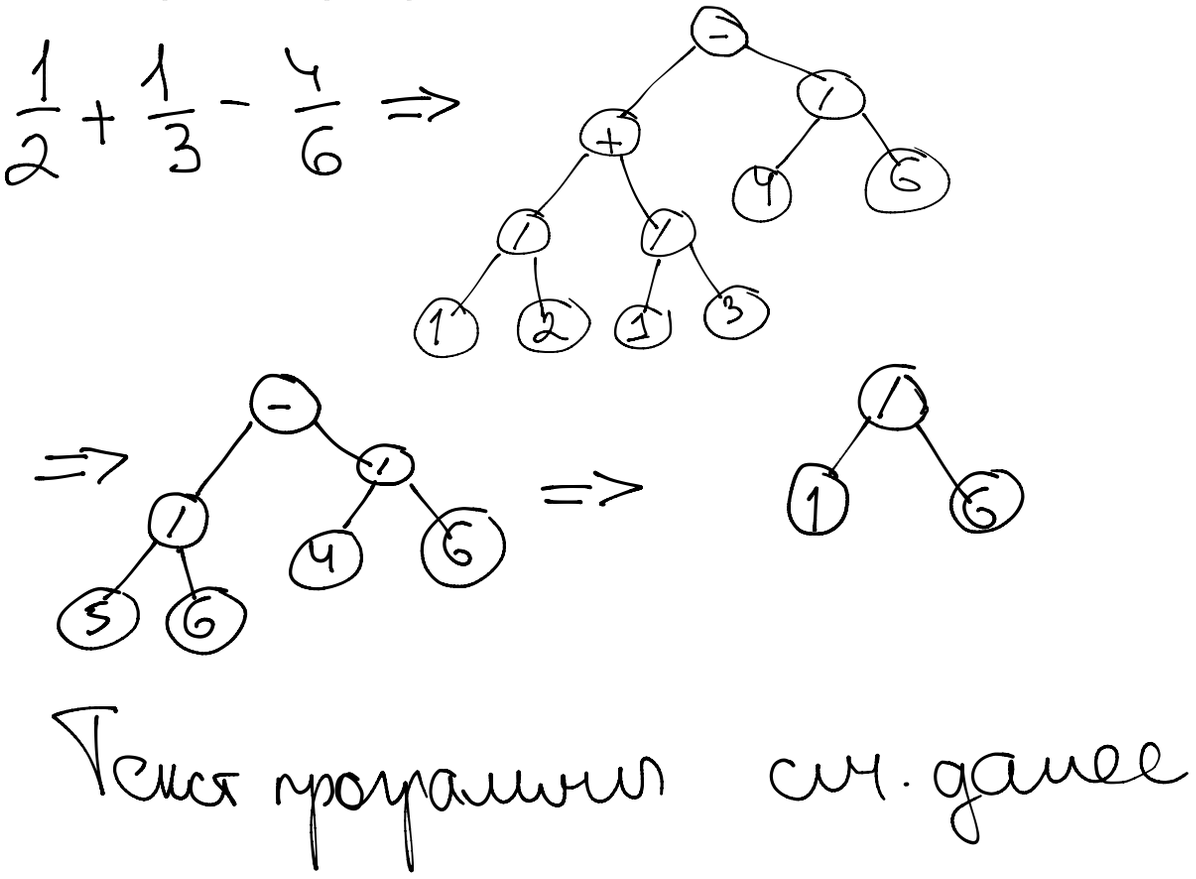
6. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

Построение дерева выражений по заданному выражению в предложенной программе реализовано методом Рутисхаузера. 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).
Сложение/вычитание дробей осуществляется следующим образом: значение левой ветки для узла '/' числитель, а правая – знаменатель. У + или - если в обоих дочерних узлах есть знаки '/', то с дробями можно работать. Схема перемножения представлена на рисунке.



Дополнительно, чтобы сократить дробь, мы сокращаем числитель и знаменатель на НОД(числитель, знаменатель).

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



Текст программы см. далее

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

Допущен к выполнению работы. Подпись преподавателя _____

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

int gcd(int a, int b) {
   while (a != b) {
      if (a > b) {
         int tmp = a;
         a = b;
         b = tmp;
      }
      b -= a;
   }

   return a;
}

typedef char tdata;

int Llen = 0;
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 == '/')) {
         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)) {
      Llen++;
      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 transtree(link tree) {
   char c, cl, cr;
   if (tree) {
      if (tree->data == '+') {
         auto lhs = tree->left;
         auto rhs = tree->right;
         if (lhs->data == '/' && rhs->data == '/') {
            Llen++;
            int la, ra, lb, rb;
            la = lhs->left->data - '0';
            ra = lhs->right->data - '0';
            lb = rhs->left->data - '0';
            rb = rhs->right->data - '0';
            int numerator = rb * la + ra * lb;
            int denominator = ra * rb;
```

```
            int nod = gcd(numerator, denominator);
            numerator /= nod;
            denominator /= nod;

            tree->data = '/';
            tree->left->data = numerator + '0';
            tree->right->data = denominator + '0';
            tree->left->left = 0;
            tree->left->right = 0;
            tree->right->left = 0;
            tree->right->right = 0;
         }
      }
      else if (tree->data == '-') {
         auto lhs = tree->left;
         auto rhs = tree->right;
         if (lhs->data == '/' && rhs->data == '/') {
            Llen++;
            int la, ra, lb, rb;
            la = lhs->left->data - '0';
            ra = lhs->right->data - '0';
            lb = rhs->left->data - '0';
            rb = rhs->right->data - '0';
            int numerator = rb * la - ra * lb;
            int denominator = ra * rb;
            int nod = gcd(numerator, denominator);
            numerator /= nod;
            denominator /= nod;

            tree->data = '/';
            tree->left->data = numerator + '0';
            tree->right->data = denominator + '0';
            tree->left->left = 0;
            tree->left->right = 0;
            tree->right->left = 0;
            tree->right->right = 0;
         }
      }
      transtree(tree->left);
      transtree(tree->right);
   }
}

int main() {
   printf("Input expression:\n");
   tree = expr();
   printtree(tree);
   printf("\n\n----------------------\n\n");
   tree2expr(tree);
   i = 1;
   while (Llen > 0) {
      Llen--;
      transtree(tree);
   }
   printf("\n\n----------------------\n\n");
   printtree(tree);
   printf("\n\n----------------------\n\n");
   tree2expr(tree);
   printf("\n\n----------------------\n\n");
   return 0;
}
```

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

См. далее результат

```
198097@client17:~/Lab24$ cat head.txt
**********************************
*                                *
*    Лабораторная работа №22     *
*      Деревья выражений         *
*  Выполнил студент группы 104   *
*  Черница Артём Александрович   *
*                                *
**********************************
198097@client17:~/Lab24$ g++ -std=c++11 main.cpp -o main.exe
198097@client17:~/Lab24$ ./main.exe
Input expression:
1/2+1/3
            \__3
        \__/
            \__1
    \__+
            \__2
        \__/
            \__1


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

(1/2+1/3)

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

        \__6
    \__/
        \__5


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

5/6

------------------------
198097@client17:~/Lab24$ ./main.exe
Input expression:
1/2-2/6
            \__6
        \__/
            \__2
    \__-
            \__2
        \__/
            \__1


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

(1/2-2/6)

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

        \__6
    \__/
        \__1
```

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

1/6

------------------------
198097@client17:~/Lab24$ ./main.exe
Input expression:
1/2+1/3-4/6
            \__6
        \__/
            \__4
    \__-
                \__3
            \__/
                \__1
        \__+
                \__2
            \__/
                \__1


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

((1/2+1/3)-4/6)

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

        \__6
    \__/
        \__1


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

1/6

------------------------
198097@client17:~/Lab24$
```

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

10. **Замечания автора** по существу работы _____

11. **Выводы**

Научился работать с деревом выражений, реализовал алгоритм решения предложенной задачи.

Недочёты при выполнении задания могут быть устранены следующим образом:_____

Подпись студента