Отчёт по лабораторной работе № 24 по курсу Практикум на ЭВМ

Студент группы М8О-111Б Тимирчев Янис Федорович, № по списку 23

Контакты www, e-mail, icq, skype yanis.timirchev@yandex.ru

Работа выполнена: «26» апреля 2022 г.

Преподаватель: доцент каф.806 Никулин С.П.

Входной контроль знаний с оценкой _____

Отчёт сдан « » _____201 __ 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 №* 17)**:** Перемножить степени с одинаковыми основаниями (в простейшем случае можно рассматривать основания, состоящие из одной переменной или константы):
   a^2*a^k=a^(2+k)

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

   Оборудование ПЭВМ студента, если использовалось:
   ПроцессорIntel Core i5-7300HQ с ОП 16Гб Мб, НМД 1 ТБ Мб. Монитор 1920x1080 Full HD
   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 Linux, наименование Ubuntu версия 20.04.3
   интерпретатор команд bash версия 5.0.17 (1).
   Система программирования _____версия _____
   Редактор текстов _____версия 26.2
   Утилиты операционной системы_____
   Прикладные системы и программы emacs
   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 реализовано методом Рутисхаузера.\
Алгоритм Рутисхаузера:
Один из наиболее ранних алгоритмов предполгаает полную скобочную структуру выражения - такую форму записи, при которой порядок действий задается расстановкой скобок. Неявное старшинство при этом не учитывается.
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 термальными 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 (термальная 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].

Код:
```
#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 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 == '^') {
        link l = tree->left;
        link r = tree->right;
        link base1 = l->left;
        link base2 = r->left;
        if (base1->data == base2->data && !base1->left && !base1->right && !base2->left && !base2->right) {
          tree->data = '^';
          tree->left = base1;
          link plus = new node;
          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;
    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;
}
```

*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).

```
ianis@dev1l $ g++ main.cpp
ianis@dev1l $ ./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
   \__2
  \__^
   \__a
 \__*
   \__k
  \__^
   \__a

Input the number of menu: 3
a^k*a^2
Input the number of menu: 4
Input the number of menu: 2
   \__2
  \__+
   \__k
 \__^
  \__a

Input the number of menu: 3
a^(k+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
     \__k
   \__^
    \__6
  \__*
    \__k
   \__^
    \__5
 \__+
   \__2
  \__+
    \__5
   \__+
      \__7
     \__*
      \__4
    \__+
      \__2
     \__+
      \__6
```

```
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: 2
    \__k
  \__^
    \__6
 \__*
    \__k
  \__^
    \__5
\__+
  \__2
 \__+
    \__5
  \__+
      \__7
    \__*
      \__4
   \__+
      \__2
    \__+
      \__6
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
     \__4
   \__+
      \__2
    \__*
      \__3
    \__+
      \__3
  \__^
    \__2
 \__*
    \__3
  \__^
    \__2
\__+
 \__6
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
     \__4
   \__+
      \__2
    \__*
      \__3
    \__+
      \__3
  \__+
    \__3
 \__^
  \__2
\__+
 \__6
Input the number of menu: 3
(6+2^(3+((3+3*2)+4)))
Input the number of menu: 1
Input expression:
(5^(7+8))*(5^(5*3))+(4^4)*(4^9)
```

```
Input the number of menu: 2
    \__9
  \__^
    \__4
 \__*
    \__4
  \__^
    \__4
\__+
    \__3
  \__*
    \__5
  \__^
    \__5
 \__*
    \__8
  \__+
    \__7
  \__^
    \__5

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
    \__9
  \__+
    \__4
 \__^
  \__4
\__+
    \__3
  \__*
    \__5
  \__+
    \__8
  \__+
    \__7
 \__^
  \__5

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

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

11. **Выводы**

В ходе лабораторной работы я составил программу выполнения преобразований арифметических выражений с применением деревьев.

Недочёты при выполнении задания могут быть устранены следующим образом:_____

Подпись студента_____