Отчет по лабораторной работе №24 по курсу практикум на ЭВМ

Студент группы М8О-106Б-21, Деревянко Е.А. № по списку 6

Контакты www, e-mail, icq, skype derevankok9@gmail.com

Работа выполнена: «28» мая_____2022__г.

Преподаватель: ст.преподаватель каф. 806 Дубинин А.В.

Входной контроль знаний с оценкой _____

Отчет сдан «30» мая_____2022__ г., итоговая оценка _4___

Подпись преподавателя _____

1. Тема: Деревья выражений

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

_____

3 **Задание** (*вариант № 44*)**:** Проверить, упорядочены ли слагаемые заданного многочлена по степеням x в порядке возрастания или убывания.

4 **Оборудование** (лабораторное):
ЭВМ _____, процессор _____, имя узла сети_____с ОП
НМД _____ . Терминал _____ адрес_____ Принтер
Другие устройства _____
_____

Оборудование ПЭВМ студента, если использовалось:
Процессор __ _____ с ОП _____, НМД _____. 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 _____, 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 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)

Опишем принцип преобразования входной строки в дерево выражений и проверки условия задания:
1. структура токена состоит из перечислимого типа типов токенов и объединения, в вариантах которого будут располагаться значения токенов.
2. void token_next(Token *t) — функция, переводящая непрерывный ввод char из потока stdin в токены. В зависимости от того, какой символ будет прочтен, ему присваиваем соответствующий токен; числа с помощью ungetc и scanf переводим в целый тип, тут же учитываем унарный минус (могут быть переменные или целые отрицательные числа с унарным минусом); переменной может быть только x; всему остальному даем тип ERROR; «\n» и EOF присваиваем тип FINAL.
3. Пока не тип FINAL, записываем разобранные токены на вектор, сразу же отсекаем все варианты с неправильным вводом выражения и тип ERROR;
4. int get_priority(char c) — устанавливаем приоритет операций; если ввод не является оператором, то присваиваем максимальный приоритет;
5. Tree tree_create(vector *tokens, int idx_left, int idx_right) — строим дерево по вектору токенов. Сразу же отсекаем скобки с помощью счетчика (скобок в дереве не должно быть). Далее ищем оператор с помощью приоритета и рекурсивно строим все дерево в соответствии с ним (в корне всегда оператор); с оператором возведения в степень сложнее, т.к. он в приоритете;
6. void tree_delete(Tree *t) — удаление всего дерева. Просто рекурсивно удаляем поддеревья, а затем корень;
7. int check_reverse(Tree t) и int check_right(Tree t) — проверка упорядочены ли слагаемые заданного многочлена по степеням x в порядке возрастания или убывания (вариант задания). Одна из функций проверяет степени многочлена в правильном порядке, вторая — в обратном. Отличаются они только знаками равенства. Т.к. это многочлен, то до последнего слагаемого в левом узле будет +. Установим изначальный счетчик степени узла, и циклом пройдем по левым узлам, пока там +, на каждом шаге вычисляя значение степени правого узла, и если оно меньше (больше) первоначального, то возвращаем . После цикла необходимо также проверить степень левого поддерева последнего +. Если порядок сохранен, то возвращаем 1.

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

1. Создадим вектор токенов и заполним его с помощью token_next и push;
2. Проверяем вектор на правильность арифметического выражения;
3. Если ввод правильный, построим дерево;
4. Выведем дерево на экран;
5. Выполним вариант задания и выведем результат на экран;
6. Удалим дерево.

Тесты:

```
katerina@katerina-VivoBook-ASUSLaptop-X521EA-S533EA:~/newlabs/lab24/zayc/dynamic$ ./main
x + x ^2 + 3



Expression tree:
+
    +
        x
        ^
            x
            2
    3

Tree's infix linearization:
((x+(x^2))+3)
Checking:
Polinom is not ordered

katerina@katerina-VivoBook-ASUSLaptop-X521EA-S533EA:~/newlabs/lab24/zayc/dynamic$ ./main
x ^ 2 + 2 * x + 4



Expression tree:
+
    +
        ^
            x
```

```
            2
        *
            2
            x
    4
```

Tree's infix linearization:
*(((x^2)+(2*x))+4)*
Checking:
Polinom in reverse order

katerina@katerina-VivoBook-ASUSLaptop-X521EA-S533EA:~/newlabs/lab24/zayc/dynamic$ ./main
*x ^ 2 + 2 * x + 4*

```
Expression tree:
+
    +
        ^
            x
            2
        *
            2
            x
    4
```

Tree's infix linearization:
*(((x^2)+(2*x))+4)*
Checking:
Polinom in reverse order
katerina@katerina-VivoBook-ASUSLaptop-X521EA-S533EA:~/newlabs/lab24/zayc/dynamic$ ./main
1 + 2 + 3 + x

```
Expression tree:
+
    +
        +
            1
            2
        3
    x
```

Tree's infix linearization:
(((1+2)+3)+x)
Checking:
Polinom in right order

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).

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

Арифметические выражения удобно реализовывать в виде деревьев, так как это интуитивно понятное представление, к которому применимо множество алгоритмов эффективной обработки выражений. Такие деревья имеют широкую распространенность благодаря возможности обрабатывать строку вместо огромного числа типов данных и операторов.

Недочёты при выполнении задания могут быть устранены следующим образом:

Подпись студента _____