имеют вид целая_часть**.**дробная_часть, где либо одно, либо другое может быть опущено. Целочисленные константы записываются в десятичной форме (восьмеричное и шестнадцатиричное представление не используется).

Старшинство операций и правила приведения типов аналогичны правилам Си. Тип переменной определяется по виду константы – инициализатора.

Некоторые рекомендации:

- если синтаксический анализ реализован методом рекурсивного спуска, то для выхода из «глубокой» рекурсии удобны функции setjmp и longjmp.
- в функции метода рекурсивного спуска можно вставить действия по переводу анализируемого выражения в польскую инверсную запись (ПОЛИЗ). Это позволит за один просмотр провести анализ исходного выражения и генерацию его польской записи; при использовании алгоритма Дейкстры может потребоваться еще один просмотр.
- выражения в ПОЛИЗе можно интерпретировать многократно при разных значениях переменных (если пользователь потребует этого в процессе диалога).
- таблицу переменных удобно представлять с использованием объединений, т.к. для переменных разных типов придется хранить значения типа int и типа double.

Приведенный здесь вариант калькулятора можно упростить: реализовать только целочисленную либо только вещественную арифметику.

9.3 Моделирование работы интерпретатора SHELL

(программа My_Shell)

Входной язык: подмножество командного языка SHELL (определяется вариантом).

Поток команд:

1. командный файл, т.е. каждая строка файла - это отдельная команда, которая должна быть выполнена интерпретатором (имя файла - аргумент в командной строке при вызове интерпретатора);
2. стандартный входной поток команд;
3. для исполнения каждой команды требуется запуск программы My_Shell.

ВНИМАНИЕ!!! "побочный" эффект выполнения уже обработанных команд (например, перенаправление ввода-вывода) не должен влиять на выполнение последующих команд.

Входной язык (варианты):

Общая часть (одинаковая для всех вариантов):

- # конвейер pr_1 | pr_2 | ... | pr_N для произвольного N>=2; считать, что аргументов у pr_I (1 <= I<= N) нет (но возможна реализация с произвольным числом аргументов у каждого процесса)

- # перенаправление ввода-вывода <, >, >> (в том числе для pr_1 и pr_N в конвейере)

Например, pr < data > res
pr1 | pr2 > res.txt

- # запуск в фоновом режиме & (в том числе и для конвейеров)
  Например, pr arg1 arg2 &
  pr1 | pr2 | pr3 > res.all &

Вариантная часть:

В каждый вариант входит (как минимум) один из подпунктов каждого пункта, отмеченного римской цифрой. Звездочкой отмечены более сложные подпункты. Части подпунктов, содержащие слово «возможно», могут быть опущены при выборе варианта; их реализация усложняет вариант. Вариант определяет преподаватель.

I. 1. mv old_file new_file
2. cp file copy_file

II. 1. wc filename
результат: filename строк слов символов (возможен список имен файлов; в этом случае подобная информация выдаются о каждом файле)
2. grep substring filename
результат: строки файла filename, содержащие substring как подстроку (возможен флаг -v; в этом случае результат - это строки, которые не содержат substring как подстроку)
3. cmp filename1 filename2
результат: информация о первом различии в содержимом двух файлов
Например, filename1 differs from filename2: line 5 char 36
*4. sort filename
сортировка строк файла в соответствии с кодировкой ASCII
возможны флаги:
-r обратный порядок
-f не различать большие и малые буквы
-n числовой порядок
+n начать сортировку с (n+1)-ой строки

III. 1. cat filenames
возможен флаг:
-n с нумерацией строк (если файлов несколько, то нумерация сквозная)
2. tail filename
вывод 10 последних строк файла
возможны флаги:
-n n последних строк
+n с n-ой строки и до конца файла
3. od filename
вывод содержимого файла по 10 символов в строке с указанием номера первого символа в каждой десятке

Например, 000001 a b c d \n e f g h i
000011 j k \t l m n

возможен флаг:

-b с указанием восьмеричных кодов символов

IV. 1. pr_1 ; pr_2 ; ... ; prN

последовательное выполнение команд pr_I - как если бы они были переданы интерпретатору по одной команде в строке

ВНИМАНИЕ !!! приоритет операции | выше, чем приоритет операции **;** однако, возможно использование скобок: напрмер, (pr1; pr2) | pr3, что приведет к конкатенации результатов работы pr1 и pr2, которые будут переданы процессу pr3 как входные данные.

2. pr1 && pr2

выполнить pr1; в случае успеха выполнить pr2

3. pr1 || pr2

выполнить pr1; в случае неудачи выполнить pr2