Язык для реализации матричных выражений

Алексей Сальников

2018

1. Введение

Требуется написать программу, которая будет вычислять матричные выражения, которые вводятся со стандартного потока ввода. Для этого разрабатывается свой язык матричных вычислений. Предполагается, что у нас матрицы рациональных чисел (см. предыдущее задание). Над матрицами и векторами можно производить вычисления, которые могут быть присвоены другим матрицам и векторам. Матрицы и вектора можно описывать как переменные. При помощи специальных конструкций можно читать из файлов или писать в файлы. Так же их можно распечатывать в стандартный поток вывода.

Предполагается, что пользователь может написать некоторый текстовый файл (программу для интерпретатора) который направит на вход интерпретатору, подобно shell скрипту в bash.

Требуется представить грамматику для лексического разбора в виде текстового файла в формате, описанным в конце текста данного задания, либо на бумаге. А для синтаксического разбора требуется представить файл в синтаксисе Gnu bison. См. документацию на описание файлов грамматики Gnu Bison.

2. Описание языка

Программа может содержать комментарии в виде конструкций с решёточкой и комментарии языка Си.

2.1. Декларации

В начале идёт Секция Деклараций переменных. Она начинается с ключевого слова *declare:*. далее разделённые пробельными символами, в том числе возможно, переносами строк следуют описания переменных. Описание переменных производятся подобно тому, как это происходит в C++. Сперва указывается тип переменной: *integer*, *float*, *rational*, *vector*, *matrix*. Далее после ':' список переменных, с возможной инициализацией значений в виде параметров конструктору (см. предыдущее задание). Имя переменной – идентификатор языка Си. Элементы в списке разделены запятыми, заканчивается он символом ';'. Если у чисел не указано значение, то они должны быть инициализированы нулём.

Далее представлен пример такого описания переменных.

```
declare:
    #
    # Описания матричных переменных
    #
    matrix: a,b, c("c_sparse_mtr.txt"),
            D("D sparse mtr\"pseudo\".txt");

    vector: v1,v2 ("../vector_sparse.txt");
```

```
rational:
    number1(1), number2(-10 / 3), number3 ;

/*
 * представлять как double в программе
 */
float: f(0.99999999);
integer: i,j,k(10);
```

2.2. Выражения

Далее следует блок, выражений. Выражения начинаются с конструкции *process:*. В выражениях допустимо указание скобочек, а так же операций присвоения. Приоритет операций естественный для языка программирования C++. Допускаются конструкции доступа к элементам матрицы и вектора через квадратную скобку. Например так: *v1[3]* и *c[6000,2]*.

Через двоеточие, в выражениях могут быть указаны действия над объектами. Общие для всех это: *read("имя файла")*, *write("имя файла")*, *print*. Действия *read*, *write* предполагают чтение и запись в файл, *print* распечатку в объекта в стандартный поток вывода.

Также должна присутствовать функция *info*, которая распечатывает строку. При этом строка не привязана ни к какому объекту, в отличии от *print*. Может встречаться в любом месте выражения и печать происходит в момент, когда процесс выполнения достигает данное место.

Для векторов и матриц дополнительно определено действие *rotate*, Которое меняет ориентацию вектора с с горизонтальной на вертикальную в выражениях, и для матриц оно транспонирует матрицу.

Для матриц определены действия *row(номер строки)* и *column(номер столбца)*, которые возвращают строку и столбец матрицы соответственно.

Для рациональных чисел, матриц и векторов, должно быть определено действие *make_ canonical*.

Приоритет действия выше любой другой операции, например сложения или умножения.

В случае, если присвоение происходит переменной типа *integer* или *float* необходимо произвести соответствующие преобразования типов.

Далее следует пример выражений.

```
process:
    a=(b+c info("сложили\n") ) * (E = D ^ num2) :rotate ;
    m=v1:rotate * v2;
    r=m[1,2];

    info("всем привет\n");

    r: write("лес.txt");

    number1=r;

    E: print;  E: write("hz.txt");
```

3. Замечания по реализации

При реализации, предполагается, что будут задействованы классы предыдущего задания. В случае ошибочной ситуации, например объекты класса генерируют исключение, все такие собы-

тия, должны быть напечатаны на стандартный поток ошибок, с описанием ошибки, её предполагаемой причины, а также координат в файле (строка : колонка) на которой случилась ошибка.

4. Язык описания грамматики для лексического разбора

Для лексического разбора опишем грамматику в следующих предположениях накладываемых на правила грамматики:

- $< A > \to \neg' a' | \neg' a' < B >$ – любой символ кроме символа a.
- $< A > \to .|. < B >$ – любой символ.
- $< A > \to' a' | ... |' z'$ – все символы от a до z.
- $< A > \to$ "$word$" – это сокрытие набора правил, которые задают определённое слово.
- $< A > \to \neg('a''b')$ - группировка символов для применения операции.
- $< A > \to < B >$ – пустые правила, которые делают грамматику не автоматной, но нужны для повышения читаемости грамматики. Грамматика с такими правилами легко приводится к каноническому виду регулярной грамматики путём внесения в правую часть правила A всех правых частей правила B.