

Учебный тренажер Pars.

Руководство по применению.

В тренажере реализована объектно-ориентированная модель SLR-анализатора.

В начале работы тренажер настраивается на грамматику, выдав запрос

Input grammar name>

В ответ нужно ввести имя файла с описание грамматики

Input grammar name>mlisp21

Тренажер подтверждает имя файла

Grammar:mlisp21.txt

Если файл не открывается, выдается сообщение

GRAMMA: can't open file mlisp21.txt

Good bye! >

Тренажер завершает работу.

Если файл найден и успешно открыт, тренажер создает управляющие таблицы синтаксического анализатора и запрашивает текст для анализа

Source>

Есть два способа задания исходного текста.

При первом способе вводится имя файла, содержащего текст:

Source>disp-id

При втором способе текст задается непосредственно в строке ввода:

Source>(display pi)

В этом случае текст сначала переписывается во временный файл temp.ss, а затем уже этот файл подается на вход анализатора.

Анализатор имеет два режима работы – обычный и режим трассировки действий.

В обычном режиме, если в тексте не обнаружено ошибок, выдается сообщение
Good source!

Если ошибка обнаружена, выдается сообщение об ошибке(см. ниже).

По умолчанию, если текст задан именем файла включается обычный режим, а если в строке ввода – режим трассировки.

Трассировку можно включить, записав перед именем файла ``(апостроф)`

Source>'disp-id

Source:disp-id.ss

1|(display pi)

2|

<- (

<- display

<- \$id

E -> \$id #5

<-)

DISPSET -> (display E) #73

CALC -> DISPSET #85

CALCS -> CALC #80

PROG -> CALCS #2

S -> PROG #1

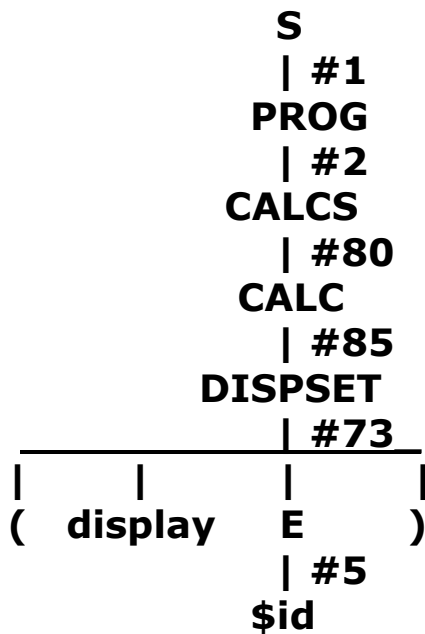
Good source!

Трасса содержит перечень действий анализатора.

Стрелка **<-** обозначает перенос терминала в стек.

Стрелка **->** обозначает свертку с применением указанной продукции.

NB!!! Если прочитать трассу от конца к началу, можно однозначно восстановить дерево разбора – сверху вниз и СПРАВА налево.



Анализатор различает два типа ошибок в исходном тексте – лексические и грамматические.

Пример лексической ошибки.

```

1|; terr1
2|-pi
3|
  
```

Lexis: unknown token!

```

2|-pi
  ^
  
```

Цепочка –pi не принадлежит ни одному токenu языка МИКРОЛИСП, идентификатор не может начинаться знаком -

Пример грамматической ошибки.

```

1|; terr2
2|(define (p? x? y?)x? y?)
3|
  
```

Syntax: unmatched token \$idq
expected:)

```

2|(define (p? x? y?)x? y?)
  ^
  
```

По правилам грамматики МИКРОЛИСПа тело процедуры-предиката может содержать только одно булевское выражение, а записано два. Второй token \$idq «лишний». Анализатор ожидает на его месте) , завершающую тело процедуры.