Лепов Алексей, гр. 1308.  
**МЛиТА, ИДЗ № 2**

(Пункты 1-10)

# **Задание (вариант № 9)**

Правильная скобочная запись арифметических выражений с двумя видами скобок. Подряд могут идти не более двух скобок одного вида. Не должно быть “лишних” скобок, и одна буква не может браться в скобки.

Правильная запись: [(a+b)/([a-b\*c]\*([a-b]\*(a+b)\*(a\*a+b\*b)))]

Неправильная запись: [(a-b)\*(a+b)\*(a\*a+b\*b)]/[a+c]\*[(a-c)/[a+c]]

# **Уточнения к заданию**

Под “лишними” скобками будем понимать скобки, без которых выражение не потеряет смысл, например: (([[a+b]]-c)) = ([a+b]-c).

## **1. Примеры и контрпримеры**

### ***1.1. Граничные примеры:***

* [a+(b\*c)/([b\*d]/[a+b+c])\*a]
* [a-b-c-d]/[(a\*a)\*(b\*b)]\*d
* [(a-b-c)\*(a+b+c+d)]/([[a+b]\*[c-d]\*d+c-a]/b)
* ([(a\*b-c/d)\*e]/([e\*d]\*[a-b-c-d-e])/a)
* [(a+b+c+d+e)/(a-b-c-d-e)]\*a\*b\*c\*d\*e

### ***1.2. Контрпримеры:***

* [(a)+(b\*c)/(([d]/[b+c])\*a)]
* ([a-b-c-d]/[(a\*a)\*(b\*b)]\*[c\*d])
* [(a-b-c)\*(b+c+d)]/(([a+b]\*[c])\*(d+c-a)/b)
* [(a\*b)-(c/d)\*(e)]/([e\*d]\*[a-b-c-d-e]/[a])\*(a\*b\*e)
* (a+b+c+d+e)/(a-b-c-d-e)\*(a\*b\*c\*d\*e)

## **2. КС-грамматика языка**

<псз> ::= (<ксз>)<кон5> | [<ксз>]<кон4> | <буква><кон1>

<ксз> ::= (<ксз>)<знак><сз1> | [<ксз>]<знак><сз2> | <буква><знак><псз>

<сз1> ::= (<ксз>)<кон2> | [<ксз>]<кон4> | <буква><кон1>

<сз3> ::= [<ксз>]<кон4> | <буква><кон1>

<сз2> ::= (<ксз>)<кон5> | [<ксз>]<кон3> | <буква><кон1>

<сз4> ::= (<ксз>)<кон5> | <буква><кон1>

<кон1> ::= <знак><псз> | λ

<кон2> ::= <знак><сз3> | λ

<кон3> ::= <знак><сз4> | λ

<кон4> ::= <знак><сз2> | λ

<кон5> ::= <знак><сз1> | λ

<знак> ::= + | - | \* | / | λ

<буква> ::= a | ... | z

## 

## **3. Грамматический разбор одного примера**

Разбор правильного примера из задания, [(a+b)/([a-b\*c]([a-b]\*(a+b)\*(a\*a+b\*b)))]:

<псз> =

[<ксз>]<кон4> =

[(<ксз>)<знак><сз1>]<кон4> =

[(<буква><знак><псз>)<знак><сз1>]<кон4> =

[(a+<буква><кон1>)<знак><сз1>]<кон4> =

[(a+b)/<сз1>]<кон4> =

[(a+b)/(<ксз>)<кон2>]<кон4> =

[(a+b)/([<ксз>]<знак><сз2>)<кон2>]<кон4> =

[(a+b)/([<ксз>]<сз2>)<кон2>]<кон4> =

[(a+b)/([буква><знак><псз>]<сз2>)<кон2>]<кон4> =

[(a+b)/([a-<псз>]<сз2>)<кон2>]<кон4> =

[(a+b)/([a-<буква><кон1>]<сз2>)<кон2>]<кон4> =

[(a+b)/([a-b<знак><псз>]<сз2>)<кон2>]<кон4> =

[(a+b)/([a-b\*<буква><кон1>]<сз2>)<кон2>]<кон4> =

[(a+b)/([a-b\*<буква><кон1>]<сз2>)<кон2>]<кон4> =

[(a+b)/([a-b\*c](<ксз>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([<ксз>]<знак><сз2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([<буква><знак><псз>]<знак><сз2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-<буква><кон1>]<знак><сз2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(<ксз>)<кон5>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(<буква><знак><псз>)<кон5>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+<псз>)<кон5>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+<буква><кон1>)<кон5>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)<кон5>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)<знак><сз1>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(<ксз>)<кон2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(<буква><знак><псз>)<кон2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(a\*a<кон1>)<кон2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(a\*a<знак><псз>)<кон2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(a\*a+<буква><кон1>)<кон2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(a\*a+b<знак><псз>)<кон2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(a\*a+b\*<буква><кон1>)<кон2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(a\*a+b\*b)<кон2>)<кон5>)<кон2>]<кон4> =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(a\*a+b\*b)λ)λ)λ]λ =

[(a+b)/([a-b\*c]([a-b]\*(a+b)\*(a\*a+b\*b)))]

## **4. Проверка на принадлежность классу LL(1)**

<псз> ::= (<ксз>)<кон5> | [<ксз>]<кон4> | <буква><кон1>

L(w1) = ( L(w1) ⋂ L(w2) = ∅

L(w2) = [ L(w1) ⋂ L(w3) = ∅

L(w3) = a..z L(w2) ⋂ L(w3) = ∅

<ксз> ::= (<ксз>)<знак><сз1> | [<ксз>]<знак><сз2> | <буква><знак><псз>

L(w1) = ( L(w1) ⋂ L(w2) = ∅

L(w2) = [ L(w1) ⋂ L(w3) = ∅

L(w3) = a..z L(w2) ⋂ L(w3) = ∅

<сз1> ::= (<ксз>)<кон2> | [<ксз>]<кон4> | <буква><кон1>

L(w1) = ( L(w1) ⋂ L(w2) = ∅

L(w2) = [ L(w1) ⋂ L(w3) = ∅

L(w3) = a..z L(w2) ⋂ L(w3) = ∅

<сз3> ::= [<ксз>]<кон4> | <буква><кон1>

L(w1) = [

L(w2) = a..z L(w1) ⋂ L(w2) = ∅

<сз2> ::= (<ксз>)<кон5> | [<ксз>]<кон3> | <буква><кон1>

L(w1) = ( L(w1) ⋂ L(w2) = ∅

L(w2) = [ L(w1) ⋂ L(w3) = ∅

L(w3) = a..z L(w2) ⋂ L(w3) = ∅

<сз4> ::= (<ксз>)<кон5> | <буква><кон1>

L(w1) = (

L(w2) = a..z L(w1) ⋂ L(w2) = ∅

<кон1> ::= <знак><псз> | λ

L(w1) = (

L(w2) = ])конец строки L(w1) ⋂ L(w2) = ∅

<кон2> ::= <знак><сз3> | λ

L(w1) = (

L(w2) = ])конец строки L(w1) ⋂ L(w2) = ∅

<кон3> ::= <знак><сз4> | λ

L(w1) = (

L(w2) = ])конец строки L(w1) ⋂ L(w2) = ∅

<кон4> ::= <знак><сз2> | λ

L(w1) = (

L(w2) = ])конец строки L(w1) ⋂ L(w2) = ∅

<кон5> ::= <знак><сз1> | λ

L(w1) = (

L(w2) = ])конец строки L(w1) ⋂ L(w2) = ∅

Грамматика удовлетворяет классу LL(1).

## **5. Модификации не требуются**

## 

## **6. Таблица перевода в диаграммы/алгоритм/код**

| Правила | Диаграмма | Алгоритм (псевдокод) | Программный код |
| --- | --- | --- | --- |
| <псз> ::= (<ксз>)<кон5> | [<ксз>]<кон4> | <буква><кон1> |  | function PSZ()  {  если ch == `(` то  {  если ch == `(` то read()  иначе ошибка  KSZ()  если ch == `)` то read()  иначе ошибка  KON5()  }  иначе если ch == `[` то  {  если ch == `[` то read()  иначе ошибка  KSZ()  если ch == `]` то read()  иначе ошибка  KON4()  }  иначе если ch ∈ {a…z} то  {  letter()  KON1()  }  иначе ошибка  } | def PSZ():  global ch  if ch in {"("}:  if ch in {"("}:  read()  else:  raise Exception()  KSZ()  if ch in {")"}:  read()  else:  raise Exception()  KON5()  elif ch in {"["}:  if ch in {"["}:  read()  else:  raise Exception()  KSZ()  if ch in {"]"}:  read()  else:  raise Exception()  KON4()  elif ch in latin:  letter()  KON1()  else:  raise Exception() |
| <ксз> ::= (<ксз>)<знак> <сз1> | [<ксз>]<знак> <сз2> | <буква><знак><псз> |  | function KSZ()  {  если ch == `(` то  {  если ch == `(` то read()  иначе ошибка  KSZ()  если ch == `)` то read()  иначе ошибка  sign()  SZ1()  }  иначе если ch == `[` то  {  если ch == `[` то read()  иначе ошибка  KSZ()  если ch == `]` то read()  иначе ошибка  sign()  SZ2()  }  иначе если ch ∈ {a…z} то  {  letter()  sign()  PSZ()  }  иначе ошибка  } | def KSZ():  global ch  if ch in {"("}:  if ch in {"("}:  read()  else:  raise Exception()  KSZ()  if ch in {")"}:  read()  else:  raise Exception()  sign()  SZ1()  elif ch in {"["}:  if ch in {"["}:  read()  else:  raise Exception()  KSZ()  if ch in {"]"}:  read()  else:  raise Exception()  sign()  SZ2()  elif ch in latin:  letter()  sign()  PSZ()  else:  raise Exception() |
| <сз1> ::= (<ксз>)<кон2> | [<ксз>]<кон4> | <буква><кон1> |  | function SZ1()  {  если ch == `(` то  {  если ch == `(` то read()  иначе ошибка  KSZ()  если ch == `)` то read()  иначе ошибка  KON2()  }  иначе если ch == `[` то  {  если ch == `[` то read()  иначе ошибка  KSZ()  если ch == `]` то read()  иначе ошибка  KON4()  }  иначе если ch ∈ {a…z} то  {  letter()  KON1()  }  иначе ошибка  } | def SZ1():  global ch  if ch in {"("}:  if ch in {"("}:  read()  else:  raise Exception()  KSZ()  if ch in {")"}:  read()  else:  raise Exception()  KON2()  elif ch in {"["}:  if ch in {"["}:  read()  else:  raise Exception()  KSZ()  if ch in {"]"}:  read()  else:  raise Exception()  KON4()  elif ch in latin:  letter()  KON1()  else:  raise Exception() |
| <сз3> ::= [<ксз>]<кон4> | <буква><кон1> |  | function SZ3()  {  если ch == `[` то  {  если ch == `[` то read()  иначе ошибка  KSZ()  если ch == `]` то read()  иначе ошибка  KON4()  }  иначе если ch ∈ {a…z} то  {  letter()  KON1()  }  иначе ошибка  } | def SZ3():  global ch  if ch in {"["}:  if ch in {"["}:  read()  else:  raise Exception()  KSZ()  if ch in {"]"}:  read()  else:  raise Exception()  KON4()  elif ch in latin:  letter()  KON1()  else:  raise Exception() |
| <сз2> ::= (<ксз>)<кон5> | [<ксз>]<кон3> | <буква><кон1> |  | function SZ2()  {  если ch == `(` то  {  если ch == `(` то read()  иначе ошибка  KSZ()  если ch == `)` то read()  иначе ошибка  KON5()  }  иначе если ch == `[` то  {  если ch == `[` то read()  иначе ошибка  KSZ()  если ch == `]` то read()  иначе ошибка  KON3()  }  иначе если ch ∈ {a…z} то  {  letter()  KON1()  }  иначе ошибка  } | def SZ2():  global ch  if ch in {"("}:  if ch in {"("}:  read()  else:  raise Exception()  KSZ()  if ch in {")"}:  read()  else:  raise Exception()  KON5()  elif ch in {"["}:  if ch in {"["}:  read()  else:  raise Exception()  KSZ()  if ch in {"]"}:  read()  else:  raise Exception()  KON3()  elif ch in latin:  letter()  KON1()  else:  raise Exception() |
| <сз4> ::= (<ксз>)<кон5> | <буква><кон1> |  | function SZ4()  {  если ch == `(` то  {  если ch == `(` то read()  иначе ошибка  KSZ()  если ch == `)` то read()  иначе ошибка  KON5()  }  иначе если ch ∈ {a…z} то  {  letter()  KON1()  }  иначе ошибка  } | def SZ4():  global ch  if ch in {"("}:  if ch in {"("}:  read()  else:  raise Exception()  KSZ()  if ch in {")"}:  read()  else:  raise Exception()  KON5()  elif ch in latin:  letter()  KON1()  else:  raise Exception() |
| <кон1> ::= <знак><псз> | λ |  | function KON1()  {  если ch ∈ {+,\*,-,/} то  {  sign()  PSZ()  }  } | def KON1():  global ch  if ch in {"\*","+","-","/"}:  sign()  PSZ() |
| <кон2> ::= <знак><сз3> | λ |  | function KON2()  {  если ch ∈ {+,\*,-,/} то  {  sign()  SZ3()  }  } | def KON2():  global ch  if ch in {"\*","+","-","/"}:  sign()  SZ3() |
| <кон3> ::= <знак><сз4> | λ |  | function KON3()  {  если ch ∈ {+,\*,-,/} то  {  sign()  SZ4()  }  } | def KON3():  global ch  if ch in {"\*","+","-","/"}:  sign()  SZ4() |
| <кон4> ::= <знак><сз2> | λ |  | function KON4()  {  если ch ∈ {+,\*,-,/} то  {  sign()  SZ2()  }  } | def KON4():  global ch  if ch in {"\*","+","-","/"}:  sign()  SZ2() |
| <кон5> ::= <знак><сз1> | λ |  | function KON5()  {  если ch ∈ {+,\*,-,/} то  {  sign()  SZ1()  }  } | def KON5():  global ch  if ch in {"\*","+","-","/"}:  sign()  SZ1() |
| <знак> ::= + | - | \* | / | λ |  | function sign()  {  если ch ∈ {+,\*,-,/} то read()  } | def sign():  global ch  if ch in {"\*","+","-","/"}:  read() |
| <буква> ::= a | ... | z |  | function letter()  {  если ch ∈ {a..z} то read()  иначе ошибка  } | def letter():  global ch  if ch in latin:  read()  else:  raise Exception() |

## 

## **9. Исходный код программы**

def PSZ():

global ch

if ch in {"("}:

if ch in {"("}:

read()

else:

raise Exception()

KSZ()

if ch in {")"}:

read()

else:

raise Exception()

KON5()

elif ch in {"["}:

if ch in {"["}:

read()

else:

raise Exception()

KSZ()

if ch in {"]"}:

read()

else:

raise Exception()

KON4()

elif ch in latin:

letter()

KON1()

else:

raise Exception()

def KSZ():

global ch

if ch in {"("}:

if ch in {"("}:

read()

else:

raise Exception()

KSZ()

if ch in {")"}:

read()

else:

raise Exception()

sign()

SZ1()

elif ch in {"["}:

if ch in {"["}:

read()

else:

raise Exception()

KSZ()

if ch in {"]"}:

read()

else:

raise Exception()

sign()

SZ2()

elif ch in latin:

letter()

sign()

PSZ()

else:

raise Exception()

def SZ1():

global ch

if ch in {"("}:

if ch in {"("}:

read()

else:

raise Exception()

KSZ()

if ch in {")"}:

read()

else:

raise Exception()

KON2()

elif ch in {"["}:

if ch in {"["}:

read()

else:

raise Exception()

KSZ()

if ch in {"]"}:

read()

else:

raise Exception()

KON4()

elif ch in latin:

letter()

KON1()

else:

raise Exception()

def SZ3():

global ch

if ch in {"["}:

if ch in {"["}:

read()

else:

raise Exception()

KSZ()

if ch in {"]"}:

read()

else:

raise Exception()

KON4()

elif ch in latin:

letter()

KON1()

else:

raise Exception()

def SZ2():

global ch

if ch in {"("}:

if ch in {"("}:

read()

else:

raise Exception()

KSZ()

if ch in {")"}:

read()

else:

raise Exception()

KON5()

elif ch in {"["}:

if ch in {"["}:

read()

else:

raise Exception()

KSZ()

if ch in {"]"}:

read()

else:

raise Exception()

KON3()

elif ch in latin:

letter()

KON1()

else:

raise Exception()

def SZ4():

global ch

if ch in {"("}:

if ch in {"("}:

read()

else:

raise Exception()

KSZ()

if ch in {")"}:

read()

else:

raise Exception()

KON5()

elif ch in latin:

letter()

KON1()

else:

raise Exception()

def KON1():

global ch

if ch in {"\*","+","-","/"}:

sign()

PSZ()

def KON2():

global ch

if ch in {"\*","+","-","/"}:

sign()

SZ3()

def KON3():

global ch

if ch in {"\*","+","-","/"}:

sign()

SZ4()

def KON4():

global ch

if ch in {"\*","+","-","/"}:

sign()

SZ2()

def KON5():

global ch

if ch in {"\*","+","-","/"}:

sign()

SZ1()

def letter():

global ch

if ch in latin:

read()

else:

raise Exception()

def read():

global ch

global string

global current

current += 1

ch = string[current]

def sign():

global ch

if ch in {"\*","+","-","/"}:

read()

latin = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

rule = """

===========================================================================

Правила

===========================================================================

Правильная скобочная запись арифметических выражений с двумя видами скобок.

Подряд могут идти не более двух скобок одного вида.

Не должно быть “лишних” скобок, и одна буква не может браться в скобки.

==========================================================================="""

print(rule)

while True:

print("Введите выражение:")

string = f"{input()}#"

current = 0

ch = string[current]

try:

PSZ()

print("Эта запись ПРАВИЛЬНАЯ!")

except BaseException:

print("Эта запись НЕПРАВИЛЬНАЯ!")

print("Попробовать снова?")

print("Ведите \"Y\" или \"N\": \n ^")

if input() == "N":

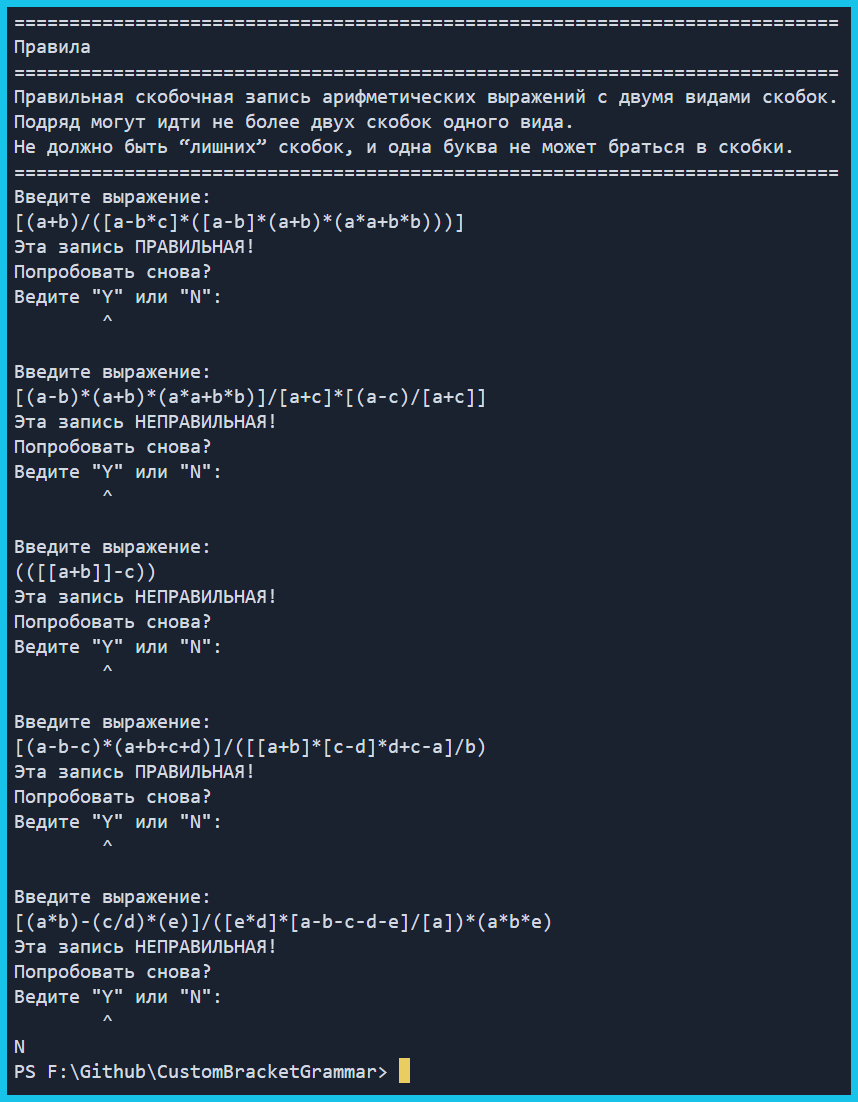
exit()

## **10. Исполняемый файл и примеры работы программы**

### ***10.1. Файл программы:***

<https://github.com/AlexeyLepov/CustomBracketGrammar/releases/download/v2.0.0-final-portable/CustomBracketGrammar.zip>

### ***10.2. Примеры работы программы:***



# **Вывод**

В результате данной работы была получена грамматика языка, грамматический разбор примера, проверка на удовлетворение условия однозначности ветвления по первому символу, таблицу перевода языка в диаграммы, таблицу перевода синтаксических диаграмм в алгоритм синтаксического анализа и таблицу перевода алгоритма в программу.

Таким образом, была проведена полная аналитическая процедура для построения компилятора для данного языка.