**Цель работы:** Перевод исходной программы в обратную польскую запись.

**Понятие обратной польской записи**

Обратная польская запись (ОПЗ) - представляет собой одну из форм записи выражений и операторов, отличительной особенностью которой является расположение аргументов (операндов) перед операцией (оператором).

Например, выражение, записанное в обычной скобочной записи, (a+d)/c+b\*(e+d), в ОПЗ имеет следующее представление: ad+c/bed+\*+.

**Код программы на языке Python:**

def js\_to\_rpn(code):

priority\_table = {

"[": 0, "{": 0, "if": 0,

"]": 1, "}": 1, ",": 1, ";": 1,

"=": 2,

">": 3, ">=": 3, "<": 3, "<=": 3, "==": 3,

"+": 4, "-": 4,

"\*": 5, "/": 5,

"\*\*": 6

}

token = code.split()

postfix = []

stack = []

# Создание списка из букв и цифр

str = string.ascii\_letters

alnum = []

for i in str:

alnum.append(i)

ifcount = 0

if2 = 0

cic = 0

ciccount = 0

otrc = 0

fco = 0

cicend = 0

for i in range(len(token)):

if token[i] in alnum or token[i].isdigit():

postfix.append(token[i])

elif token[i] in priority\_table:

while stack and stack[-1] in priority\_table and priority\_table[token[i]] <= priority\_table[stack[-1]]:

postfix.append(stack.pop())

stack.append(token[i])

elif token[i] == '(':

stack.append(token[i])

elif token[i] == ')':

while stack and stack[-1] != '(':

postfix.append(stack.pop())

stack.pop()

# Массивы

if token[i] == '[':

countmass = 2

while token[i] != ']':

if token[i] == ',':

countmass += 1

i += 1

str = "{}".format(countmass)

elif token[i] == ']':

postfix.append(str + "АЭМ")

# Функции

k = 0

if token[i] == 'f(':

fco += 1

stack.append('(')

postfix.append('f')

countfun = 2

while token[i] != ')':

k += 1

if token[i] == ',':

countfun += 1

i += 1

str = "{}".format(countfun)

elif token[i] == ')' and fco > 0:

if k == 1:

str = "1"

postfix.append(str + "Ф")

# Условный оператор

if token[i] == 'if(':

if2 += 1

for j in code:

if j == '{':

otrc += 1

stack.append('(')

if token[i] == ')' and if2 > 0:

qwer = "{}".format(if2)

postfix.append("М" + qwer + " УПЛ")

if token[i] == '{' and if2 > 0:

ifcount += 1

if token[i] == '}' and if2 > 0:

str2 = "{}".format(ifcount)

if ifcount != otrc:

str3 = "{}".format(ifcount + 1)

postfix.append("M" + str3 + "БП")

postfix.append("М" + str2 + ":")

# Циклы

if token[i] == 'while(':

stack.append('(')

cic += 1

if token[i] == ')' and cic > 0 and cicend == 0:

postfix.append('МЦ1 УПЛ')

cicend = 1

if token[i] == '{' and cic > 0:

ciccount += 1

if token[i] == '}' and cic > 0:

str5 = "{}".format(ciccount)

postfix.append("МЦ" + str5 + ":")

while stack:

postfix.append(stack.pop())

while '[' in postfix:

postfix.remove('[')

while ']' in postfix:

postfix.remove(']')

while 'function' in postfix:

postfix.remove('function')

while ',' in postfix:

postfix.remove(',')

while ';' in postfix:

postfix.remove(';')

while '{' in postfix:

postfix.remove('{')

while '}' in postfix:

postfix.remove('}')

return ' '.join(postfix)

**Примеры работы программы:**

Пример № 1 – Работа с обыкновенными математическими выражениями.

Исходная строка: ( a + b ) \* c / ( d \* ( 4 + b ) )

Результат работы программы: a b + c \* d 4 b + \* /

Пример № 2 – Работа с элементами с индексами.

Исходная строка: s = 0 ; s += a[ a , b \* ( 5 + 2 ) ]

Результат работы программы: s 0 = s a a b 5 2 + \* 3АЭМ +=

Пример № 3 – Работа с условными выражениями.

Исходная строка: if( a > b ) { a = x - y } else { a = y - x }

Результат работы программы: a b > М1 УПЛ a x y - = M2БП М1: a y x - = М2:

Пример № 4 – Работа с элементами с указателями на функцию.

Исходная строка: res = f( x , y , z + 1 )

Результат работы программы: f x y z 1 + 4Ф =

Пример № 5 – Работа с циклами.

Исходная строка: var a = 5 ; var b = 10 ; while( a < b ) { a = ( a + 5 ) \* 2 }

Результат работы программы: a 5 = b 10 = a b < МЦ1 УПЛ a a 5 + 2 \* = МЦ1:

**Вывод:** в ходе данной лабораторной работы было осуществлен перевод кода программы входного языка javascript в обратную польскую запись.