НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №4

з дисципліни **«**Системне програмування 2**»**

Виконав:

студент групи ІО-42, ФІОТ

Коваленко В’ячеслав Сергійович

ЗК : 4209

м. Київ 2016 р.

Варіант



Лістинг програми

expression = 'b=c?d:2\*a[n];'

tokens = {'=', 'ASSIGN',

':', 'COLON',

'?', 'QUESTION\_MARK',

'\*', 'MUL',

'[', 'LEFT\_SQUARE\_BRACKET',

']', 'RIGHT\_SQUARE\_BRACKET',

';', 'SEMI+COLON'}

brackets\_open = ['(', '[']

brackets\_close = [')', ']']

simple\_operations = ['\*']

def compare\_brackets(br\_o, br\_c):

if (br\_o == brackets\_open[0]) and (br\_c == brackets\_close[0]):

return True

if (br\_o == brackets\_open[1]) and (br\_c == brackets\_close[1]):

return True

return False

def common\_check(var):

# check brackets

is\_bracket = False

balance = []

last\_bkt\_ind = 0

for i in range(len(var)):

if var[i] in brackets\_open:

balance.append(var[i])

is\_bracket = True

last\_bkt\_ind = i

if var[i] in brackets\_close:

if len(balance) != 0:

if compare\_brackets(balance[len(balance) - 1], var[i]):

balance.pop(len(balance) - 1)

else:

print('Exception: wrong statement ' + var)

return False

if (i - last\_bkt\_ind) == 1:

print('Exception: wrong statement ' + var + ' . Tip : error in brackets.')

if not len(balance) == 0:

print('Exception: wrong statement ' + var)

return False

if not is\_bracket:

if var not in tokens:

return True

else:

print('Exception: wrong assignment' + var)

return False

return True

def check\_simple\_var(var):

return True if common\_check(var) else False

def check\_complex\_tokens(var):

if common\_check(var):

for ind in range(len(var)):

if var[ind] in simple\_operations:

try:

if (var[ind-1] in tokens) or (var[ind+1] in tokens):

print('Exception: wrong statement ' + var)

return False

except IndexError:

print('Exception: wrong statement ' + var)

return False

else:

return False

return True

def main():

global expression

print('Expression ' + expression)

# delete all spaces

expression = expression.replace(' ', '')

# check semi\_colon

if expression[-1] != ';':

print('Exception: SEMI\_COLON ";" was not found')

return

# check necessary elements of ternary operator

necessary\_tokens = ['=', '?', ':']

token\_index = 0

for i in expression:

if i == necessary\_tokens[token\_index]:

token\_index += 1

if token\_index == len(necessary\_tokens):

break

if token\_index != 3:

print('Exception: not all necessary elements or wrong sequence. Tip : error with "' + necessary\_tokens[token\_index] + '"')

return

# check variables

if check\_simple\_var(expression[:expression.index(necessary\_tokens[0])]):

if check\_complex\_tokens(expression[expression.index(necessary\_tokens[0]) + 1: expression.index(necessary\_tokens[1])]):

if check\_complex\_tokens(expression[expression.index(necessary\_tokens[1]) + 1:expression.index(necessary\_tokens[2])]):

if check\_complex\_tokens(expression[expression.index(necessary\_tokens[2]) + 1:-1]):

print('No exceptions found')

if \_\_name\_\_ == '\_\_main\_\_':

main()

Результати програми





