НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №4**

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

Варіант 3

Виконала:

студентка 3 курсу гр. ІО-64

Бровченко А. В.

Перевірив:

Павлов В. Г.

Київ 2018 р.

**Тема:** Створення та на­стро­ювання висхідних синтаксичних аналізаторів.

**Мета**: Одержання навичок настроювання таблиць висхідних синтаксичних аналізаторів та програм побудови графів та дерев розбору на етапі синтаксичного аналізу з запам’ятовуванням покажчиків на вхідні образи та внутрішні коди. Вивчення програм формування повідомлень про помилки та побудови графа підлеглості операцій та операторів в процесі висхідного синтаксичного аналізу.

**Варіант:**

**Лістинг програми:**

**import** lab4.tokens **as** t  
**import** re  
**from** termcolor **import** colored  
  
expression = **'b-=a[n.32];'  
  
  
def** compare\_brackets(br\_o, br\_c):  
 **if** (br\_o == t.brackets\_open[0]) **and** (br\_c == t.brackets\_close[0]):  
 **return True  
 elif** (br\_o == t.brackets\_open[1]) **and** (br\_c == t.brackets\_close[1]):  
 **return True  
 else**: **return False  
  
  
def** common\_check(var):  
 **if** len(var) == 0:  
 print(colored(**'Exception: empty statement'**, **'red'**))  
 **return False** *# check brackets* is\_bracket = **False** balance = []  
 last\_bkt\_ind = 0  
 **for** i **in** range(len(var)):  
 **if** var[i] **in** t.brackets\_open:  
 balance.append(var[i])  
 is\_bracket = **True** last\_bkt\_ind = i  
  
 **if** var[i] **in** t.brackets\_close:  
 **if** len(balance) != 0:  
 **if** compare\_brackets(balance[len(balance) - 1], var[i]):  
 balance.pop(len(balance) - 1)  
 **else**:  
 print(colored(**'Exception: wrong statement '** + var, **'red'**))  
 **return False  
 if** (i - last\_bkt\_ind) == 1:  
 print(colored(**'Exception: wrong statement '** + var + **'Exception: empty brackets.'**, **'red'**))  
 **return False  
  
 if not** len(balance) == 0:  
 print(colored(**'Exception: wrong statement '** + var, **'red'**))  
 **return False  
  
 if** var[-1] **in** t.simple\_operations:  
 print(colored(**'Exception: wrong statement '** + var, **'red'**))  
 **return False  
  
 if** is\_bracket:  
 **try**:  
 in\_brackets = re.search(**'(?<=\[).\*?(?=\])'**, var).group()  
 **if** bool(re.search(**'[^A-Za-z0-9]'**, in\_brackets)):  
 print(colored(**'Exception: wrong statement in brackets '** + in\_brackets, **'red'**))  
 **return False  
 except**:**pass  
  
 if** bool(re.search(**'[^ 0-9=+-]+[0-9]+?[.]?[0-9]+'**, var)):  
 print(colored(**'Exception: wrong statement '** + var, **'red'**))  
 **return False  
  
 if not** is\_bracket:  
 **if** var **not in** t.tokens:  
 **return True  
 else**:  
 print(colored(**'Exception: wrong statement '** + var, **'red'**))  
 **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** t.simple\_operations:  
 **try**:  
 **if** (var[ind+1] == **'='**) **or** (var[ind:ind+2] **or** var[ind-1:ind+1] **in** t.inc\_dec\_tokens):  
 **pass  
 elif** (var[ind-1] **in** t.tokens) **or** (var[ind+1] **in** t.tokens):  
 print(colored(**'Exception: wrong statement '** + var, **'red'**))  
 **return False  
 except** IndexError:  
 print(colored(**'Exception: wrong statement '** + var, **'red'**))  
 **return False  
 else**:  
 **return False  
 return True  
  
  
def** main():  
 **global** expression  
  
 print(**'Expression '**, colored(expression, **'magenta'**))  
  
 *# delete all spaces* expression = expression.replace(**' '**, **''**)  
  
 *# check semi\_colon* **if** expression[-1] != **';'**:  
 print(colored(**'Exception: SEMI\_COLON ";" was not found'**, **'red'**))  
 **return** *# check necessary elements* index\_assign\_token = 0  
 **for** i **in** t.assignment\_tokens:  
 **if** i **in** expression:  
 index\_assign\_token = expression.index(i)  
 **break  
  
 if** index\_assign\_token !=0:  
 **if** check\_simple\_var(expression[:index\_assign\_token]):  
 **if** check\_complex\_tokens(expression[index\_assign\_token:]):  
 print(colored(**'No exceptions found'**, **'green'**))  
 **else**: print(colored(**'Exception: Assignment error.'**, **'red'**))  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

**tokens.py**

tokens = {**'='**: **'ASSIGN'**,  
 **':'**: **'COLON'**,  
 **'?'**: **'QUESTION\_MARK'**,  
 **'\*'**: **'MUL'**,  
 **'['**: **'LEFT\_SQUARE\_BRACKET'**,  
 **']'**: **'RIGHT\_SQUARE\_BRACKET'**,  
 **';'**: **'SEMI\_COLON'**,  
 **'/'**: **'DIVIDE'**,  
 **'%'**: **'DIVIDE'**,  
 **'-'**: **'SUBTRACT'**,  
 **'+'**: **'ADD'**,  
 **'+='**: **'SUM\_ASSIGN'**,  
 **'-='**: **'SUB\_ASSIGN'**,  
 **'&'**: **'LOGIC\_AND'**,  
 **'|'**: **'LOGIC\_OR'**,  
 **'^'**: **'LOGIC\_EXCLUSIVE\_OR'**,  
 **'++'**: **'INC'**,  
 **'--'**: **'DEC'**}  
  
assignment\_tokens = [**'+='**, **'-='**, **'='**]  
  
brackets\_open = [**'('**, **'['**]  
brackets\_close = [**')'**, **']'**]  
  
inc\_dec\_tokens = [**'++'**, **'--'**]  
  
simple\_operations = [**'\*'**, **'+'**, **'-'**, **'/'**, **'&'**, **'|'**, **'^'**]

**Контрольні приклади:**

**Приклад1:**

D:\Program\_Files\Python36\python.exe D:/Python/sys\_prog/lab4/data.py

Expression b+=a[n];

No exceptions found

Process finished with exit code 0

**Приклад2:**

D:\Program\_Files\Python36\python.exe D:/Python/sys\_prog/lab4/data.py

Expression b-=a3.2[n];

Exception: wrong statement -=a3.2[n];

Process finished with exit code 0

**Приклад3:**

D:\Program\_Files\Python36\python.exe D:/Python/sys\_prog/lab4/data.py

Expression b-=a[n.32];

Exception: wrong statement in brackets n.32

Process finished with exit code 0