Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА _____КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ_____

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Компьютерные системы и сети

О Т Ч Е Т

по домашней работе № 2

Дисциплина: __Машинно-зависимые языки и основы компиляции_____

Название домашнего задания: Лексические и синтаксические анализаторы

Студент гр. **ИУ6-42Б** _____ **И.А. Люляев**
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____
(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Вариант 2.17

Цель работы: закрепление знаний теоретических основ и основных методов приемов разработки лексических и синтаксических анализаторов регулярных и контекстно- свободных формальных языков.

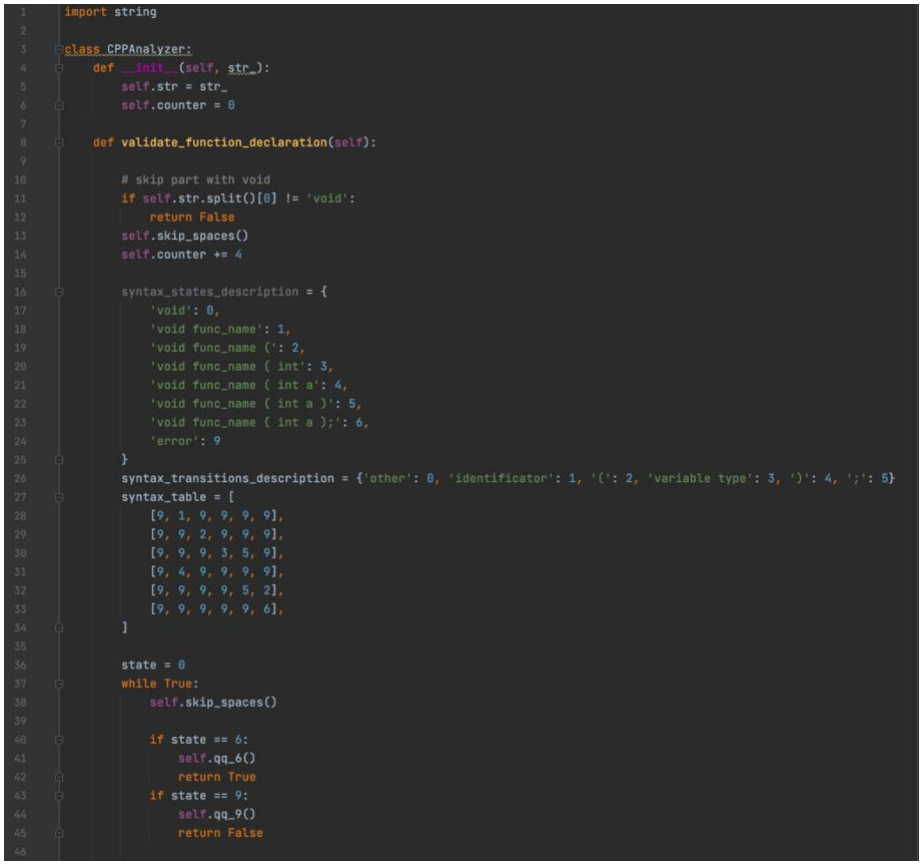
Домашнее задание №2. Лексические и синтаксические анализаторы.
Разработать грамматику и распознаватель имени процедуры для языка программирования C++. Считать, что параметры только стандартных (скалярных) типов. Например:
void f1(int a; float b; double c);

Рисунок 1 – условие

В форме Бэкуса-Наура:
<процедура> ::= void <идентификатор> (<выражение>);
<выражение> ::= <тип> <идентификатор>; <выражение> |
<тип> <идентификатор>; |
<идентификатор> ::= <буква><последовательность символов> | <буква>
<последовательность символов> ::=
<цифра>< последовательность символов > |
<буква>< последовательность символов > |
<цифра> |
<буква>
<буква> ::= 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 | 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 | _
<цифра> ::= 0|1|2|3|4|5|6|7|8|9
<тип> ::=
int | double | float | bool | char | unsigned char | unsigned int | long | long long

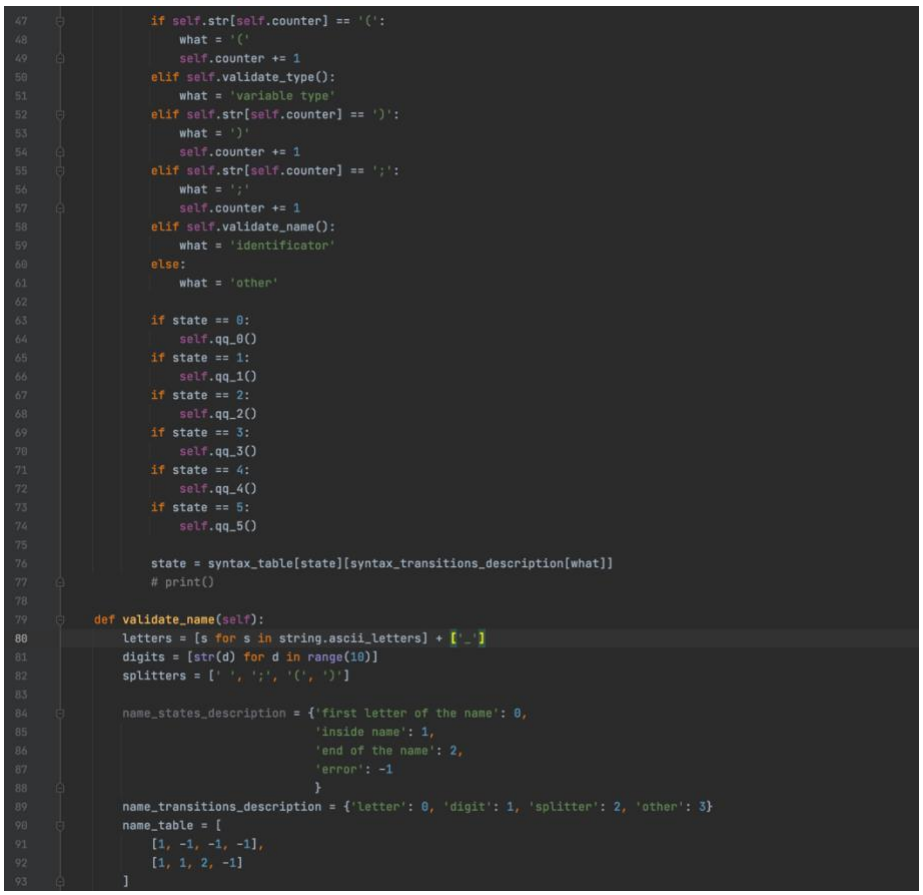
Язык C++ имеет контекстно-свободную грамматику, однако условия данного домашнего задания сводят необходимую для распознавания грамматику до регулярной.

Для лексического анализа мною был использован алгоритм конечного автомата, ввиду его большей универсальности. Для синтаксического анализа также использовался конечный автомат, ввиду того, что грамматика регулярная и автомат с магазинной памятью будет излишен.



```
import string

class CPPAnalyzer:
    def __init__(self, str_):
        self.str = str_
        self.counter = 0

    def validate_function_declaration(self):

        # skip part with void
        if self.str.split()[0] != 'void':
            return False
        self.skip_spaces()
        self.counter += 4

        syntax_states_description = {
            'void': 0,
            'void func_name': 1,
            'void func_name (': 2,
            'void func_name ( int': 3,
            'void func_name ( int a': 4,
            'void func_name ( int a )': 5,
            'void func_name ( int a );': 6,
            'error': 9
        }
        syntax_transitions_description = {'other': 0, 'identificator': 1, '(': 2, 'variable type': 3, ')': 4, ';': 5}
        syntax_table = [
            [9, 1, 9, 9, 9, 9],
            [9, 9, 2, 9, 9, 9],
            [9, 9, 9, 3, 5, 9],
            [9, 4, 9, 9, 9, 9],
            [9, 9, 9, 9, 5, 2],
            [9, 9, 9, 9, 9, 6],
        ]

        state = 0
        while True:
            self.skip_spaces()

            if state == 6:
                self.qq_6()
                return True
            if state == 9:
                self.qq_9()
                return False
```

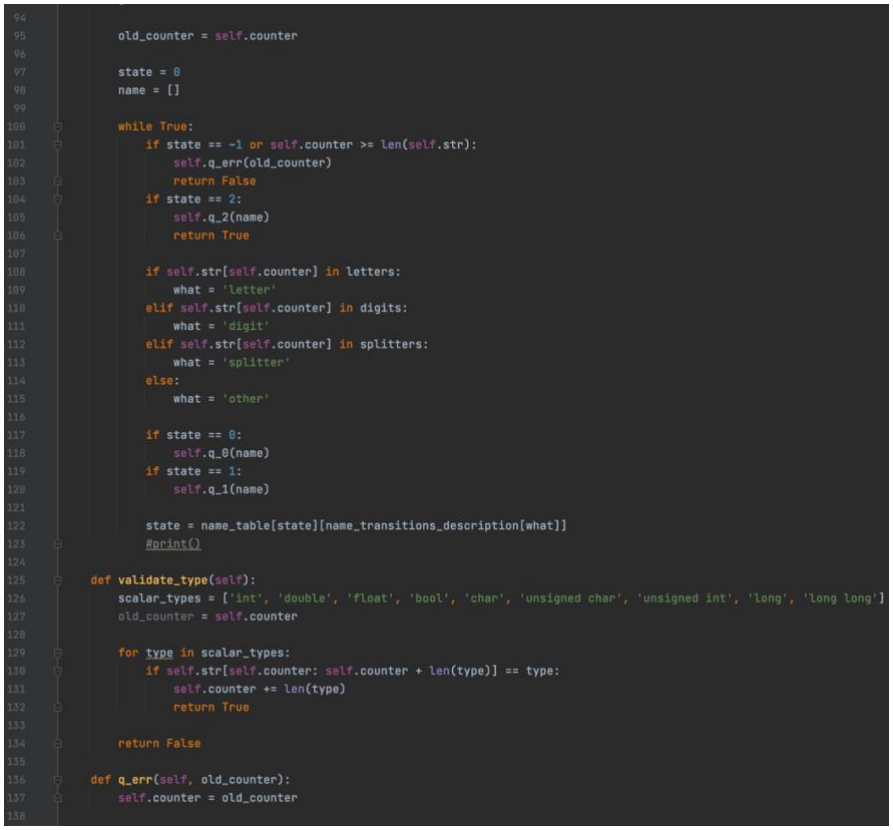
Рисунок 2 - программа (начало)



```
            if self.str[self.counter] == '(':
                what = '('
                self.counter += 1
            elif self.validate_type():
                what = 'variable type'
            elif self.str[self.counter] == ')':
                what = ')'
                self.counter += 1
            elif self.str[self.counter] == ';':
                what = ';'
                self.counter += 1
            elif self.validate_name():
                what = 'identificator'
            else:
                what = 'other'

            if state == 0:
                self.qq_0()
            if state == 1:
                self.qq_1()
            if state == 2:
                self.qq_2()
            if state == 3:
                self.qq_3()
            if state == 4:
                self.qq_4()
            if state == 5:
                self.qq_5()

            state = syntax_table[state][syntax_transitions_description[what]]
            # print()

    def validate_name(self):
        letters = [s for s in string.ascii_letters] + ['_']
        digits = [str(d) for d in range(10)]
        splitters = [' ', ';', '(', ')']

        name_states_description = {'first letter of the name': 0,
                                   'inside name': 1,
                                   'end of the name': 2,
                                   'error': -1
                                   }
        name_transitions_description = {'letter': 0, 'digit': 1, 'splitter': 2, 'other': 3}
        name_table = [
            [1, -1, -1, -1],
            [1, 1, 2, -1]
        ]
```

Рисунок 3 - программа (продолжение)

Москва, 2021



```
        old_counter = self.counter

        state = 0
        name = []

        while True:
            if state == -1 or self.counter >= len(self.str):
                self.q_err(old_counter)
                return False
            if state == 2:
                self.q_2(name)
                return True

            if self.str[self.counter] in letters:
                what = 'letter'
            elif self.str[self.counter] in digits:
                what = 'digit'
            elif self.str[self.counter] in splitters:
                what = 'splitter'
            else:
                what = 'other'

            if state == 0:
                self.q_0(name)
            if state == 1:
                self.q_1(name)

            state = name_table[state][name_transitions_description[what]]
            #print()

    def validate_type(self):
        scalar_types = ['int', 'double', 'float', 'bool', 'char', 'unsigned char', 'unsigned int', 'long', 'long long']
        old_counter = self.counter

        for type in scalar_types:
            if self.str[self.counter: self.counter + len(type)] == type:
                self.counter += len(type)
                return True

        return False

    def q_err(self, old_counter):
        self.counter = old_counter
```

Рисунок 4 - программа (продолжение)

```
    def q_0(self, name):
        name.append(self.str[self.counter])
        self.counter += 1

    def q_1(self, name):
        name.append(self.str[self.counter])
        self.counter += 1

    def q_2(self, name):
        self.counter -= 1

    def skip_spaces(self):
        while self.counter < len(self.str) and self.str[self.counter] == ' ':
            self.counter += 1

    def qq_6(self):
        pass

    def qq_0(self):
        pass

    def qq_9(self):
        pass

    def qq_1(self):
        pass

    def qq_4(self):
        pass

    def qq_3(self):
        pass

    def qq_2(self):
        pass

    def qq_5(self):
        pass


while True:
    str_ = input()
    c = CPPAnalyzer(str_)
    if str_ == 'end':
        break

    if c.validate_function_declaration():
        print('Конструкция распознана')
```

Рисунок 5 - программа (продолжение)

Москва, 2021

Рисунок 6 - программа (конец)

Таблица 1 - Результаты тестирования

Исходные данные	Ожидание	Результат
void func(int a; double feel; long looom);	Конструкция распознана	Конструкция распознана
void func();	Конструкция распознана	Конструкция распознана
void Funkee_123_r(int a; double feel; long looom);	Конструкция распознана	Конструкция распознана
void func(int a; double feel; long looom)	Обнаружена ошибка	Обнаружена ошибка
void func (int a; double feel; long looom);	Конструкция распознана	Конструкция распознана
void func(int 1a; double feel; long looom);	Обнаружена ошибка	Обнаружена ошибка
void func(int 1a; double feel; long looom(;	Обнаружена ошибка	Обнаружена ошибка
void func(int 1a; double feel; long looom);)	Обнаружена ошибка	Обнаружена ошибка

Вывод:

В данном домашнем задании я реализовал простейший лексический и синтаксический анализатор.