Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра вычислительных технологий

**ОТЧЕТ**

о выполнении лабораторной работы №4

по дисциплине “Методы разработки трансляторов”

Выполнил: ст. гр. 36/2

Баева Д. Н.

Проверил: Вишняков Ю.М

Краснодар

2023

**Содержание**

[1 Вариант задания 3](#_Toc97893076)

[2 Построение синтаксического анализатора 4](#_Toc97893077)

[3 Результаты экспериментов 8](#_Toc97893078)

[Приложение А Грамматика языка С++ 15](#_Toc97893080)

[Приложение Б Листинг программы и комментарии к нему 17](#_Toc97893082)

**1 Вариант задания**

Вариант задания представляет собой пару: входной язык и выходной язык (таблица 1).

Таблица 1 – Вариант задания

|  |  |  |
| --- | --- | --- |
| № варианта | Входной язык | Выходной язык |
| 27 | R | C++ |

Разработать программу синтаксического анализатора исходного текста.

Программа получает на входе файл – результат лексического анализа и выполняет синтаксический анализ исходной программы. Результатом работы должно быть сообщение о корректности программы или сообщение об первой обнаруженной ошибке с указанием строки и конструкции языка, при разборе которых обнаружена ошибка.

Отчет по работе должен содержать полную грамматику заданного подмножества входного языка, описание алгоритма синтаксического разбора, листинг программы и комментарии к нему, пример.

**2 Построение синтаксического анализатора**

Основная задача синтаксического анализа – проверка исходной программы на соответствие грамматике языка программирования. Следует еще раз напомнить, что синтаксический анализ производится над кодом программы, который получен на выходе лексического анализа. Результат синтаксического анализа, который часто называется разбором, представляется в виде дерева разбора. Данное дерево должно демонстрировать вывод исходной программы как цепочки символов из начального символа грамматики.

Для построения дерева разбора используются различные методы синтаксического анализа, каждый из которых имеет свои особенности. В нашем случае будет рассмотрен пример разбора «сверху-вниз», который называется рекурсивным спуском. Основная особенность рекурсивного спуска заключается в том, что для анализа каждого нетерминала используется отдельная семантическая процедура, а многократное обращение к ней в процессе анализа и дало название методу – «рекурсивный спуск».

Рассмотрим суть синтаксического анализа методом рекурсивного спуска на примере.

Вначале построим грамматику языка программирования Perl. Его грамматика приведена в приложении А.

При программной реализации данного алгоритма были определены следующие переменные и процедуры, за которыми закреплены определенные функциональные назначения.

1. Переменная NXTSYMB – глобальная переменная, содержащая символ, следующий за разбираемым символом. NXTSYMB содержит символ (лексему) исходной программы, который будет обрабатываться следующим, а при поиске новой цели в переменной NXTSYMB всегда находится первый символ, который должен быть исследован.

При выходе из процедуры с сообщением об успехе символ, следующий за закрытой подцепочкой, помещается в NXTSYMB.

1. Процедура SCAN готовит очередной символ исходной программы и помещает его в NXTSYMB.
2. Процедура ERROR предназначена для обработки ошибочных ситуаций.

Ниже в таблице 2 собраны нетерминалы грамматики и имена соответствующих им рекурсивных семантических процедур.

Таблица 2 – Нетерминалы грамматики и имена соответствующих им рекурсивных семантических процедур

|  |  |
| --- | --- |
| Нетерминальные символы грамматики | Имена рекурсивных процедур |
| <программа> | PROGRAM |
| <операторы> | OPERATORS |
| <оператор> | LABEL |
| <имя> | NAME |
| <функция> | FUNCTION |
| <составной оператор> | COMPOUND\_OPERATOR |
| <выражение> | EXPRESSION |
| <число> | NUMBER |
| <целое число> | INTEGER |
| <вещественное число> | REAL\_NUMBER |
| <строка> | LINE |
| <переменная> | VARIABLE |
| <арифметическая операция> | ARITHMETIC\_OPERATION |
| <опер. присв.> | ASSIGNMENT\_OPERATOR |
| <условный оператор> | CONDITIONAL\_OPERATOR |
| <условие> | CONDITION |
| <унарная лог. операция> | UNARY\_LOG\_OPERATION |
| <лог. выражение> | LOG\_EXPRESSION |
| <операция сравнения> | COMPARISON\_OPERATION |
| <бинарная лог. операция> | BINARY\_LOG\_OPERATION |
| <цикл while> | WHILE\_LOOP |
| <цикл do while> | DO\_WHILE\_LOOP |
| <цикл for> | FOR\_LOOP |
| <оператор goto> | GOTO\_OPERATOR |

Продолжение таблицы 2

|  |  |
| --- | --- |
| Нетерминальные символы грамматики | Имена рекурсивных процедур |
| <оператор break> | BREAK\_OPERATOR |
| <оператор continue> | CONTINUE\_OPERATOR |
| <оператор return> | RETURN\_OPERATOR |
| <оператор print> | PRINT\_OPERATOR |

Процедура NAME возвращает значение ИСТИНА или ЛОЖЬ в зависимости от того, является ли лексема, содержащаяся в переменной NXTSYMB, идентификатором.

Процедуры NUMBER, INTEGER и REAL\_NUMBER возвращают значение ИСТИНА или ЛОЖЬ в зависимости от того, является ли лексема, содержащаяся в переменной NXTSYMB, числовой константой.

Процедура LINE возвращает значение ИСТИНА или ЛОЖЬ в зависимости от того, является ли лексема, содержащаяся в переменной NXTSYMB, символьной константой.

Процедура ARITHMETIC\_OPERATION возвращает значение ИСТИНА или ЛОЖЬ в зависимости от того, является ли лексема, содержащаяся в переменной NXTSYMB, арифметической операцией.

Процедура UNARY\_LOG\_OPERATION возвращает значение ИСТИНА или ЛОЖЬ в зависимости от того, является ли лексема, содержащаяся в переменной NXTSYMB, унарной логической операцией.

Процедура COMPARISON\_OPERATION возвращает значение ИСТИНА или ЛОЖЬ в зависимости от того, является ли лексема, содержащаяся в переменной NXTSYMB, операцией сравнения.

Процедура BINARY\_LOG\_OPERATION возвращает значение ИСТИНА или ЛОЖЬ в зависимости от того, является ли лексема, содержащаяся в переменной NXTSYMB, бинарной логической операцией.

Процедура BREAK\_OPERATOR возвращает значение ИСТИНА или ЛОЖЬ в зависимости от того, является ли лексема, содержащаяся в переменной NXTSYMB, оператором break.

Процедура CONTINUE\_OPERATOR возвращает значение ИСТИНА или ЛОЖЬ в зависимости от того, является ли лексема, содержащаяся в переменной NXTSYMB, оператором continue.

При инициализации синтаксического анализатора идет обращение к процедуре SCAN, которая помещает первый символ исходной программы в глобальную переменную NXTSYMB и вызывает процедуру PROGRAM.

Для рассмотренной грамматики программа синтаксического анализатора по методу рекурсивного спуска приведена в приложении Б.

**3 Результаты экспериментов**

Пример работы программы для программы для тестирования: на рисунке 1 представлен скриншот файла, содержащего верный текст на выходном языке программирования, на рисунках 2 и 3 – скриншот файла, содержащего текст на выходном языке программирования с ошибкой.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1 – Скриншот части файла, содержащего верный текст на выходном языке программирования

Допустим ошибку нарочно: удалим фигурную скобку одного из условных операторов в строке 21 (рисунок 2).

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 2 – Скриншот файла, содержащего текст на выходном языке программирования с ошибкой

Затем еще раз нарочно допустим ошибку: удалим 30 строку кода (рисунок 3).

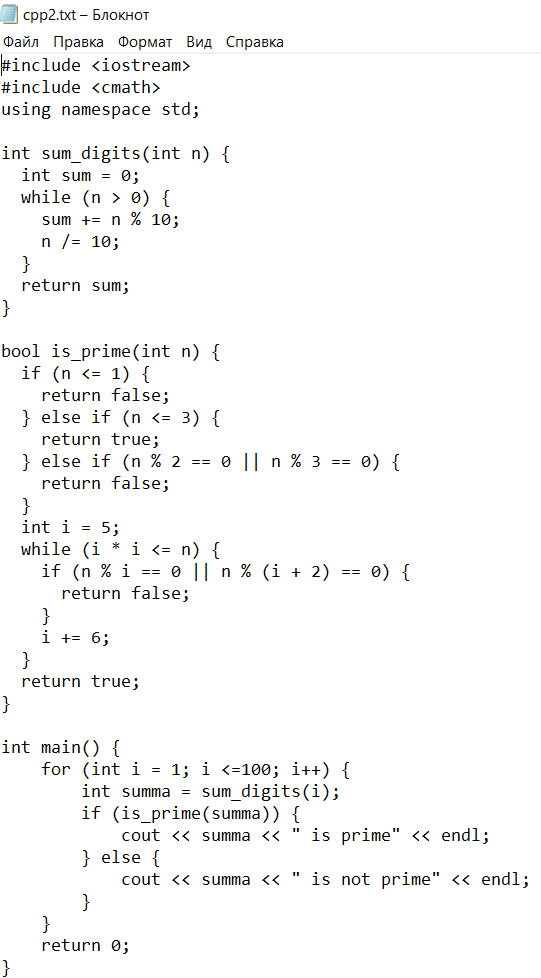


Рисунок 3 – Скриншот файла, содержащего текст на выходном языке программирования с ошибкой

На следующих рисунках 4,5 и 6 будет представлен интерфейс после выполнения каждой из ошибочных программ соответственно.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 4 – Скриншот интерфейса, после проверки правильного текста на выходном языке программирования

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 5 – Скриншот интерфейса, после проверки текста на выходном языке программирования с ошибкой

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 6 – Скриншот интерфейса, после проверки текста на выходном языке программирования с ошибкой

Изображение выглядит как снимок экрана, текст, Прямоугольник, программное обеспечение

Автоматически созданное описаниеНа рисунке 7 представлен интерфейс до выполнения синтаксического анализа.

Рисунок 7 – Скриншот первоначального интерфейса

**ПРИЛОЖЕНИЕ А**

**Грамматика языка С++**

Начальным символом грамматики является нетерминал <программа>. Грамматика имеет следующий вид:

<программа> ::= <операторы>

<операторы> ::= <оператор>; {<оператор>;}

<оператор> ::= [<метка>] (<функция>

| <оператор вызова функции>

| <составной оператор>

| <опер. присв.>

| <условный оператор>

| <цикл while>

| <цикл do while>

| <цикл for>

| <оператор goto>

| <оператор break>

| <оператор continue>

| <оператор return>

| <оператор print>)

<метка> ::= <имя>:

<имя> ::= [($ | & | @ | \_) | <буква> {<буква>} (\_ | <цифра>)}] (\_ | <буква> | <цифра>)

{(\_ | <буква> | <цифра>)}

<буква> ::= *A* | *B* | … | *Z*

<цифра> ::= 0 | 1 | … | 9

<функция> ::= *SUB* <имя>

<составной оператор>

<оператор вызов функции> ::= <имя>«(»<выражение> {, <выражение>}«)»

<выражение> ::= <оператор вызова функции>

| <число>

| <строка>

| <переменная>

| «(»<выражение>«)»

| <выражение> <арифметическая операция> <выражение>

<число> ::= <целое число> | <вещественное число>

<целое число> ::= <цифра> {, <цифра>}

<вещественное число> ::= <целое число>

| <целое число>.<целое число> [(e | E) [(- | +)] <целое число>]

<строка> ::= ''{символ}'' | '{символ}'

<переменная> ::= <имя>[«[»<выражение>«]»]

<арифметическая операция> ::= % | \* | \*\* | + | - | .. | /

<составной оператор> ::= «{»<операторы>«}»

<опер. присв.> ::= <переменная> = <выражение>

<условный оператор> ::= *IF* «(»<условие>«)» <составной оператор> [*ELSE* <составной оператор>]

<условие> ::= <унарная лог. операция> «(»<лог. выражение>«)»

| <лог. выражение> {<бинарная лог. операция> <лог. выражение>}

<унарная лог. операция> ::= !

<лог. выражение> ::= <выражение> <операция сравнения> <выражение>

<операция сравнения> ::= != | < | <= | == | > | >=

<бинарная лог. операция> ::= && | ||

<цикл while> ::= *WHILE* «(»<условие>«)» <составной оператор>

<цикл do while> ::= *DO* <составной оператор> *WHILE* «(»<условие>«)»

<цикл for> ::= *FOR* «(»[<опер. присв.>]; [<условие>]; [<опер. присв.>]«)» <составной оператор>

<оператор goto> ::= *GOTO* <имя>

<оператор break> ::= *BREAK*

<оператор continue> ::= *CONTINUE*

<оператор return> ::= *RETURN* <выражение>

<оператор print> ::= *PRINT* <выражение>

**Приложение Б  
Листинг программы и комментарии к нему**

**import** json  
**import** re  
**import** sys  
**import** tkinter  
**from** tkinter **import** \*  
**import** tkinter.scrolledtext **as** st  
**from** tkinter **import** filedialog  
  
*# лексемы*tokens = {**'W'**: {}, **'I'**: {}, **'O'**: {}, **'R'**: {}, **'N'**: {}, **'C'**: {}}  
  
*# файлы, содержащие все таблицы лексем***for** token\_class **in** tokens.keys():  
 **with** open(**'%s.json'** % token\_class, **'r'**) **as** read\_file:  
 data = json.load(read\_file)  
 tokens[token\_class] = data  
  
*# файл, содержащий последовательность кодов лексем входной программы*f = open(**'tokens.txt'**, **'r'**)  
input\_sequence = f.read()  
f.close()  
  
regexp = **'['** + **'|'**.join(tokens.keys()) + **']'** + **'\d+'**match = re.findall(regexp, input\_sequence)  
  
i = -1 *# индекс разбираемого символа*nxtsymb = **None** *# разбираемый символ*row\_counter = 1 *# счётчик строк  
  
# обработка ошибочной ситуации***def** error():  
 **global** row\_counter  
 *# print('Ошибка в строке', row\_counter)* sys.exit()  
  
*# помещение очередного символа в nxtsymb***def** scan():  
 **global** i, nxtsymb, row\_counter  
 i += 1  
 **if** i >= len(match):  
 error()  
 **for** token\_class **in** tokens.keys():  
 **if** match[i] **in** tokens[token\_class]:  
 nxtsymb = tokens[token\_class][match[i]]  
 **if** nxtsymb == **'\n'**:  
 row\_counter += 1  
 scan()  
 *# print(i, row\_counter, nxtsymb)  
  
# программа***def** program():  
 scan()  
 **if** nxtsymb != **'program'**: error()  
 scan()  
 **if not**(name()): error()  
 scan()  
 **if** nxtsymb != **';'**: error()  
 scan()  
 **if** nxtsymb **in** [**'var'**, **'procedure'**, **'function'**]: descriptions()  
 compound\_operator()  
 scan()  
 **if** nxtsymb != **'.'**: error()  
  
*# описания***def** descriptions():  
 **if** nxtsymb == **'var'**: description\_of\_variables()  
 **elif** nxtsymb == **'procedure'**: description\_of\_procedures()  
 **elif** nxtsymb == **'function'**: function\_description()  
  
*# имя (идентификатор)***def** name():  
 **return** nxtsymb **in** tokens[**'I'**].values() **or** \  
 nxtsymb **in** [**'abs'**, **'cos'**, **'exp'**, **'ln'**, **'read'**, **'readln'**, **'sin'**, **'sqrt'**, \  
 **'write'**, **'writeln'**]  
  
*# описание переменных***def** description\_of\_variables():  
 **if** nxtsymb != **'var'**: error()  
 scan()  
 description\_element()  
 scan()  
 **while** nxtsymb != **'procedure' and** nxtsymb != **'function' and** nxtsymb != **'begin'**:  
 description\_element()  
 scan()  
  
*# элемент описания***def** description\_element():  
 list\_of\_names()  
 **if** nxtsymb != **':'**: error()  
 type()  
 scan()  
 **if** nxtsymb != **';'**: error()  
  
*# список имен***def** list\_of\_names():  
 **if not**(name()): error()  
 scan()  
 **while** nxtsymb == **','**:  
 scan()  
 **if not**(name()): error()  
 scan()  
  
*# тип***def** type():  
 scan()  
 **if** nxtsymb == **'array'**: array\_type()  
 **elif not**(base\_type()): error()  
  
*# тип массива***def** array\_type():  
 scan()  
 **if** nxtsymb != **'['**: error()  
 index\_type()  
 scan()  
 **while** nxtsymb == **','**:  
 index\_type()  
 scan()  
 **if** nxtsymb != **']'**: error()  
 scan()  
 **if** nxtsymb != **'of'**: error()  
 scan()  
 **if not**(base\_type()): error()  
  
*# базовый тип***def** base\_type():  
 **return** nxtsymb **in** [**'integer'**, **'real'**, **'string'**]  
  
*# тип индекса***def** index\_type():  
 scan()  
 **if not**(integer()): error()  
 scan()  
 **if** nxtsymb != **'..'**: error()  
 scan()  
 **if not**(integer()): error()  
  
*# целое число (числовая константа)***def** integer():  
 **return** nxtsymb **in** tokens[**'N'**].values()  
  
*# описание процедур***def** description\_of\_procedures():  
 procedure()  
 scan()  
 **while** nxtsymb == **'procedure'**:  
 procedure()  
 scan()  
  
*# описание функций***def** function\_description():  
 function()  
 scan()  
 **while** nxtsymb == **'function'**:  
 procedure()  
 scan()  
  
*# процедура***def** procedure():  
 **if** nxtsymb != **'procedure'**: error()  
 scan()  
 **if not**(name()): error()  
 scan()  
 **if** nxtsymb != **'('**: error()  
 options()  
 scan()  
 **if** nxtsymb != **')'**: error()  
 scan()  
 **if** nxtsymb **in** [**'var'**, **'procedure'**, **'function'**]: descriptions()  
 compound\_operator()  
 scan()  
 **if** nxtsymb != **';'**: error()  
  
*# функция***def** function():  
 **if** nxtsymb != **'function'**: error()  
 scan()  
 **if not**(name()): error()  
 scan()  
 **if** nxtsymb == **'('**:  
 options()  
 scan()  
 **if** nxtsymb != **')'**: error()  
 scan()  
 **if** nxtsymb != **':'**: error()  
 type()  
 scan()  
 **if** nxtsymb **in** [**'var'**, **'procedure'**, **'function'**]: descriptions()  
 compound\_operator()  
 scan()  
 **if** nxtsymb != **';'**: error()  
  
*# параметры***def** options():  
 section()  
 scan()  
 **while** nxtsymb == **';'**:  
 section()  
 scan()  
  
*# секция***def** section():  
 list\_of\_names()  
 scan()  
 **if** nxtsymb != **':'**: error()  
 scan()  
 **if not**(base\_type()): error()  
  
*# составной оператор***def** compound\_operator():  
 **if** nxtsymb != **'begin'**: error()  
 operators()  
 **if** nxtsymb != **'end'**: error()  
  
*# операторы***def** operators():  
 scan()  
 **while** name() **or** \  
 nxtsymb **in** [**'begin'**, **'if'**, **'for'**, **'while'**, **'repeat'**, **'goto'**, **'break'**, \  
 **'continue'**]:  
 operator()  
 **if** nxtsymb != **';' and** nxtsymb != **'end'**: error()  
 **if** nxtsymb == **';'**:  
 scan()  
 **if** nxtsymb == **'end'**:  
 **break***# оператор***def** operator():  
 **if** name():  
 scan()  
 **if** nxtsymb == **':'**:  
 scan()  
 operator()  
 **elif** nxtsymb == **'('**:  
 scan()  
 **if** nxtsymb != **')'**:  
 expression()  
 **while** nxtsymb == **','**:  
 scan()  
 expression()  
 **if** nxtsymb != **')'**: error()  
 scan()  
 **elif** nxtsymb == **'['**:  
 scan()  
 expression()  
 **while** nxtsymb == **','**:  
 scan()  
 expression()  
 **if** nxtsymb != **']'**: error()  
 scan()  
 **elif** nxtsymb == **':='**:  
 scan()  
 expression()  
 **else**: error()  
 **elif** nxtsymb == **'begin'**:  
 compound\_operator()  
 scan()  
 **elif** nxtsymb == **'if'**:  
 conditional\_operator()  
 **if** nxtsymb != **';'**:  
 scan()  
 **elif** nxtsymb == **'for'**:  
 for\_loop()  
 **if** nxtsymb != **';'**:  
 scan()  
 **elif** nxtsymb == **'while'**:  
 while\_loop()  
 **if** nxtsymb != **';'**:  
 scan()  
 **elif** nxtsymb == **'repeat'**:  
 loop\_repeat()  
 scan()  
 **elif** nxtsymb == **'goto'**:  
 goto\_statement()  
 scan()  
 **elif** nxtsymb == **'break'**:  
 break\_operator()  
 scan()  
 **elif** nxtsymb == **'continue'**:  
 continue\_operator()  
 scan()  
 **else**: error()  
  
*# выражение***def** expression():  
 **if** nxtsymb == **'('**:  
 scan()  
 expression()  
 **if** nxtsymb != **')'**: error()  
 scan()  
 **elif** name():  
 scan()  
 **if** nxtsymb == **'('**:  
 scan()  
 expression()  
 **while** nxtsymb == **','**:  
 scan()  
 expression()  
 **if** nxtsymb != **')'**: error()  
 scan()  
 **elif** nxtsymb == **'['**:  
 scan()  
 expression()  
 **while** nxtsymb == **','**:  
 scan()  
 expression()  
 **if** nxtsymb != **']'**: error()  
 scan()  
 **elif** number() **or** line(): scan()  
 **else**: error()  
 **if** arithmetic\_operation():  
 scan()  
 expression()  
  
*# число (числовая константа)***def** number():  
 **return** nxtsymb **in** tokens[**'N'**].values()  
  
*# вещественное число (числовая константа)***def** real\_number():  
 **return** nxtsymb **in** tokens[**'N'**].values()  
  
*# строка (символьная константа)***def** line():  
 **return** nxtsymb **in** tokens[**'C'**].values()  
  
*# арифметическая операция***def** arithmetic\_operation():  
 **return** nxtsymb **in** [**'\*'**, **'+'**, **'-'**, **'/'**, **'div'**, **'mod'**]  
  
*# переменная***def** variable():  
 **if not**(name()): error()  
 scan()  
 **if** nxtsymb == **'['**:  
 scan()  
 expression()  
 **while** nxtsymb == **','**:  
 scan()  
 expression()  
 **if** nxtsymb != **']'**: error()  
 scan()  
  
*# условный оператор***def** conditional\_operator():  
 **if** nxtsymb != **'if'**: error()  
 scan()  
 condition()  
 **if** nxtsymb != **'then'**: error()  
 scan()  
 operator()  
 **if** nxtsymb == **'else'**:  
 scan()  
 operator()  
  
*# условие***def** condition():  
 **if** unary\_log\_operation():  
 scan()  
 **if** nxtsymb != **'('**: error()  
 log\_expression()  
 **if** nxtsymb != **')'**: error()  
 scan()  
 **elif** nxtsymb == **'('**:  
 log\_expression()  
 **if** nxtsymb != **')'**: error()  
 scan()  
 **while** binary\_log\_operation():  
 scan()  
 **if** nxtsymb != **'('**: error()  
 log\_expression()  
 **if** nxtsymb != **')'**: error()  
 scan()  
 **else**: error()  
  
*# унарная логическая операция***def** unary\_log\_operation():  
 **return** nxtsymb == **'not'***# логическое выражение***def** log\_expression():  
 scan()  
 expression()  
 comparison\_operation()  
 scan()  
 expression()  
  
*# операция сравнения***def** comparison\_operation():  
 **return** nxtsymb **in** [**'<'**, **'<='**, **'<>'**, **'='**, **'>'**, **'>='**]  
  
*# бинарная логическая операция***def** binary\_log\_operation():  
 **return** nxtsymb == **'and' or** nxtsymb == **'or'***# цикл for***def** for\_loop():  
 **if** nxtsymb != **'for'**: error()  
 scan()  
 **if not**(name()): error()  
 scan()  
 **if** nxtsymb != **':='**: error()  
 scan()  
 expression()  
 **if** nxtsymb != **'to'**: error()  
 scan()  
 expression()  
 **if** nxtsymb != **'do' and** nxtsymb != **'downto'**: error()  
 scan()  
 operator()  
  
*# цикл while***def** while\_loop():  
 **if** nxtsymb != **'while'**: error()  
 scan()  
 condition()  
 **if** nxtsymb != **'do'**: error()  
 scan()  
 operator()  
  
*# цикл repeat***def** loop\_repeat():  
 **if** nxtsymb != **'repeat'**: error()  
 operators()  
 scan()  
 **if** nxtsymb != **'until'**: error()  
 scan()  
 condition()  
  
*# оператор goto***def** goto\_statement():  
 **if** nxtsymb != **'goto'**: error()  
 scan()  
 **if not**(name()): error()  
  
*# оператор break***def** break\_operator():  
 **return** nxtsymb == **'break'***# оператор continue***def** continue\_operator():  
 **return** nxtsymb == **'continue'***# program()***def** check():  
 cpp2txt.delete(**"1.0"**,END)  
 **if** file == **"C:/Users/Дианочка/PycharmProjects/Translator\_Development/TranslateSyntaxicAnalyzer/cpp.txt"**:  
 cpp2txt.insert(**"1.0"**, **" В программе нет ошибок. "**)  
 **elif** file == **"C:/Users/Дианочка/PycharmProjects/Translator\_Development/TranslateSyntaxicAnalyzer/cpp1.txt"**:  
 cpp2txt. insert(**"1.0"**, **"Ошибка! 21 строка. "**)  
 **elif** file == **"C:/Users/Дианочка/PycharmProjects/Translator\_Development/TranslateSyntaxicAnalyzer/cpp2.txt"**:  
 cpp2txt. insert(**"1.0"**, **"Ошибка! 30 строка. "**)  
  
  
**def** openfile():  
 **global** file  
 cpptxt.delete(**"1.0"**, END)  
 file = filedialog.askopenfilename()  
 text = open(file, encoding=**'utf-8'**).readlines()  
 text = **''**.join(text)  
 cpptxt.insert(1.0, text)  
  
  
*# создание окна интерфейса*window = tkinter.Tk()  
window.geometry(**'1700x700'**)  
window.title(**"Translate of Syntaxic Analyzer"**)  
window.configure(bg=**'#6e6e6e'**)  
  
*# расположение всех необходимых текстовых окошек, лэйблов, кнопок*label1 = tkinter.Label(window, text=**'Программа на входном языке'**,font=(**"Arial"**, 10),foreground=**"white"**, background=**"#574f4f"**)  
label1.place(x=165, y=50, width=230, height=50)  
cpptxt = st.ScrolledText(window)  
cpptxt.place(x=50,y=150,width=500,height=400)  
  
btn = Button(window, text=**"Выбрать файл"**, command=openfile, bg=**'#6a8bcc'**,font=(**"Arial"**, 10))  
btn.place(x=600, y=380, width=170, height=70)  
  
btnchk = Button(window,text=**"Выполнить проверку"**,command=check,bg=**'#ff0000'**,font=(**"Arial"**, 10))  
btnchk.place(x=800,y=380,width=170,height=70)  
  
label2 = tkinter.Label(window, text=**'Программа на выходном языке'**,font=(**"Arial"**, 10),foreground=**"white"**, background=**"#574f4f"**)  
label2.place(x=1100, y=50, width=230, height=50)  
cpp2txt = st.ScrolledText(window)  
cpp2txt.place(x=1000, y=150,width=500,height=400)  
window.mainloop()