

Documentação Projeto de Compiladores

Adriel Henrique Foppa Lima	24.122.096-1
Alan Mantelatto Mlatisuma	24.122.015-1
Enzo Bozzani Martins	24.122.020-1
Igor Augusto Fiorini Rossi	24.122.023-5

Expressões regulares dos tokens:

if_reserved -> 'if'

else_reserved -> 'else'

number_reserved -> 'number'

string_reserved -> 'string'

while_reserved -> 'while'

for_reserved -> 'for'

output_reserved -> 'output'

input_reserved -> 'input'

in_reserved -> 'in'

bool_reserved -> 'bool'

true -> 'true'

false -> 'false'

op -> '('

cp -> ')'

gt -> '>'

attr -> '='

equal -> '=='

gte -> '>='

lte -> '<='

lt -> '<'

open_curly_braces -> '{'

close_curly_braces -> '}'

string -> '"'.*'"

add -> '+'

sub -> '-'

mult -> '*'

div -> '/'

number -> (0-9)+ | (0-9)*.(0-9)+

id -> (a-z|A-Z)(a-z|A-Z|0-9|_)*

Gramática do analisador sintático:

expr -> factor expr'

expr' -> add factor expr' | sub factor expr' | mult factor expr' | div factor expr' | ϵ

factor -> number | id | op expr cp

value -> true | false | expr | string | input

string -> ".*"

condition -> value condition'

condition' -> comparison_operator value | ϵ

comparison_operator -> gt | equal | gte | lte | lt

if -> if_reserved op condition cp open_curly_braces statement+
close_curly_braces else

else -> else' | ϵ

else' -> else_reserved | open_curly_braces | statement+ | close_curly_braces

statement -> if | for | while | output | attr_expression | init_expression

type -> number_reserved | bool_reserved | string_reserved

attr_expression -> type id attr value | id attr value

init_expression -> type id

while -> while_reserved op condition cp open_curly_braces statement+
close_curly_braces

for -> for_reserved op number_reserved id in_reserved id_or_number cp
open_curly_braces statement+ close_curly_braces

output -> output_reserved value

input -> input_reserved

Características da linguagem criada:

1. Declaração/atribuição/inicialização de variáveis:

number a = 10

string b = "oi"

bool c = true

number d

string e

bool f

d = 16.9

e = "tchau"

f = false

2. Bloco condicional:

if (a > b) {

...

} else {

...

}

if (condition) {

...

- ```

}
3. Loops:
 for (i in 10) {
 ...
 }
 for (count in number_variable) {
 ...
 }
 while (condition) {
 ...
 }
4. Output/input:
 string a = input
 output a

```

#### **OBSERVAÇÕES:**

- Deve haver espaços entre os tokens. Exemplo:
  - ( 1 + 1 == 2 )
  - ( condition ) {
  - number a = 10
- No loop for, a variável declarada (exemplo: i) será um número de 0 até o limite informado (exemplo: 10)
- Para receber um valor de input, basta atribuir à uma variável do tipo string a palavra reservada input
- Para imprimir algum valor, basta usar a palavra output seguida do valor

#### **Execução do compilador:**

1. Criar arquivo com o código:

Exemplo:

```
number a = 10
```

```
number b = 5
```

```
number highest
```

```

if (a > b) {
 highest = a
} else {
 highest = b
}

```

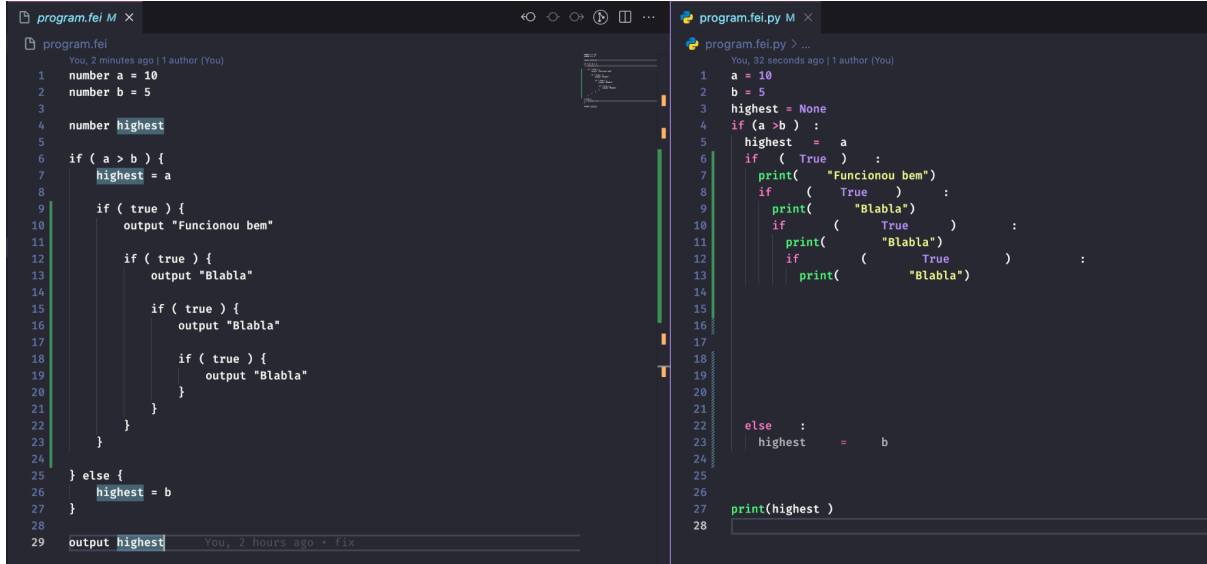
```
output highest
```

2. Executar:

No terminal, na pasta com o código fonte, execute:

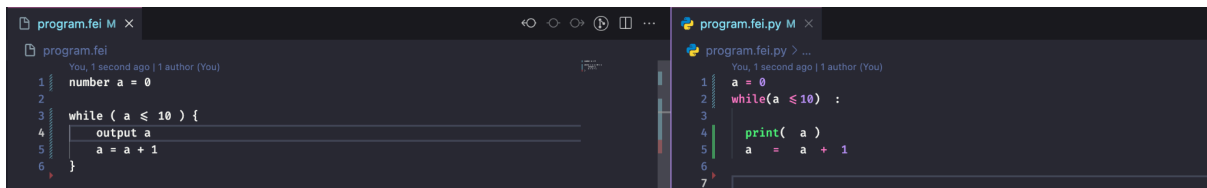
./fei caminho-pro-arquivo

## Exemplos:



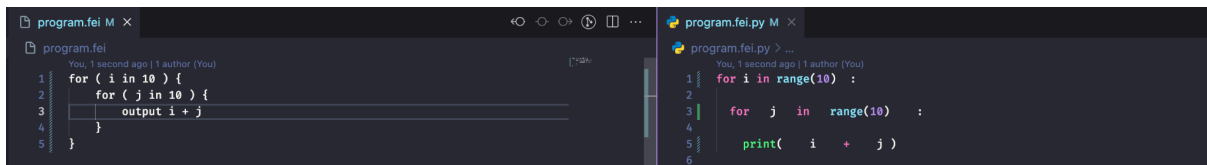
```
program.fei
1 number a = 10
2 number b = 5
3
4 number highest
5
6 if (a > b) {
7 highest = a
8
9 if (true) {
10 output "Funcionou bem"
11
12 if (true) {
13 output "Blabla"
14
15 if (true) {
16 output "Blabla"
17
18 if (true) {
19 output "Blabla"
20 }
21 }
22 }
23 }
24 } else {
25 highest = b
26 }
27
28 output highest

program.fei.py
1 a = 10
2 b = 5
3 highest = None
4 if (a > b) :
5 highest = a
6 if (True) :
7 print("Funcionou bem")
8 if (True) :
9 print("Blabla")
10 if (True) :
11 print("Blabla")
12 if (True) :
13 print("Blabla")
14
15
16
17
18
19
20
21
22 else :
23 highest = b
24
25
26
27 print(highest)
28
```



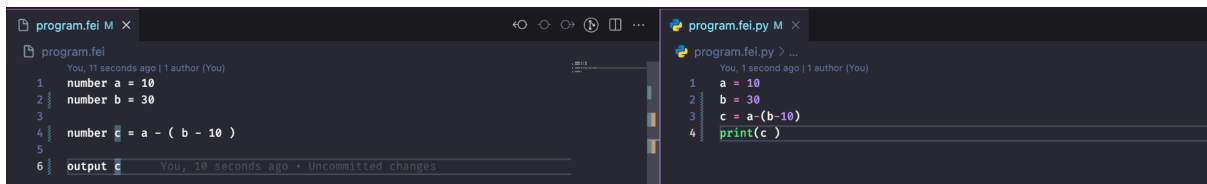
```
program.fei
1 number a = 0
2
3 while (a ≤ 10) {
4 output a
5 a = a + 1
6 }

program.fei.py
1 a = 0
2 while(a ≤ 10) :
3
4 print(a)
5 a = a + 1
6
7
```



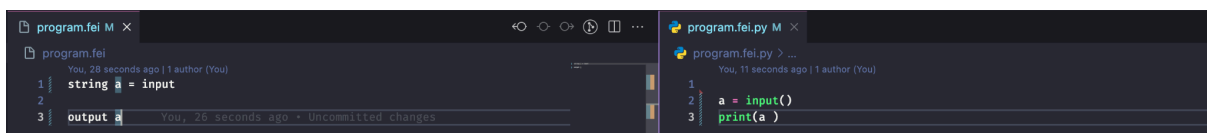
```
program.fei
1 for (i in 10) {
2 for (j in 10) {
3 output i + j
4 }
5 }

program.fei.py
1 for i in range(10) :
2
3 for j in range(10) :
4
5 print(i + j)
6
```



```
program.fei
1 number a = 10
2 number b = 30
3
4 number c = a - (b - 10)
5
6 output c

program.fei.py
1 a = 10
2 b = 30
3 c = a - (b - 10)
4 print(c)
```



```
program.fei
1 string a = input
2
3 output a

program.fei.py
1 a = input()
2
3 print(a)
```