

# Sindarin

De Matheus S.

Para: Compiladores



# Tabela de conteúdo

01

INTRODUCTION

02

PALAVRAS RESERVADAS

03

AUTOMATO

04

OBJETO DE EXEMPLO

05

APLICAÇÃO

# Introduction

Sindarin trata-se da lingaugem elfica, adotada pelos elfos e também povos de Noldor, muito usada entre os espiões elficos durante a grande guerra contra morgoth e posteriormente, Sauron.



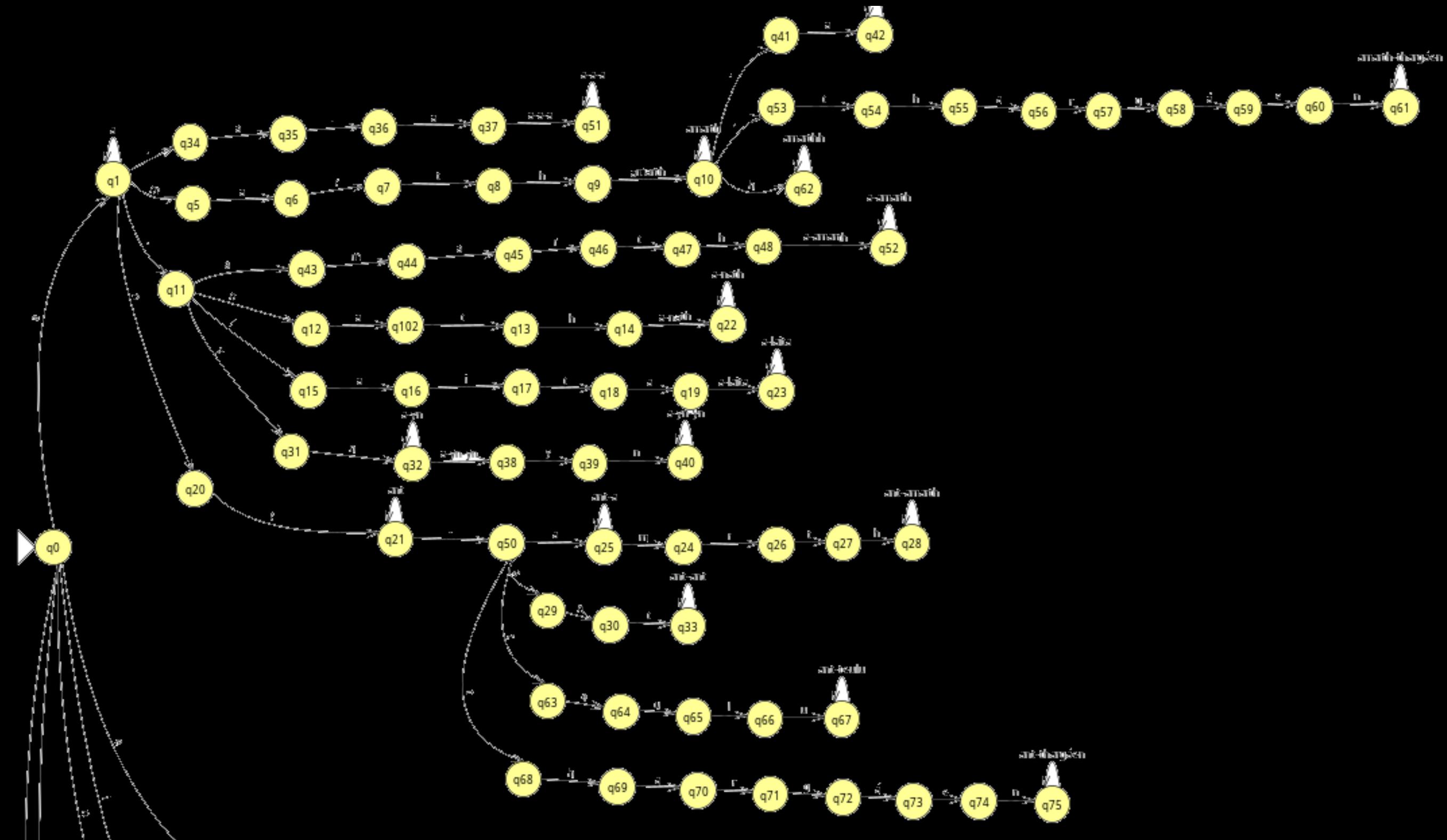
# Python

```
class  
continue  
def  
elif  
else  
except  
from  
if  
import  
in  
is  
raise  
return  
try  
while  
with  
print
```

# Sindarin

class	teulu
continue	thargáen
def	yn-amarth
elif	a-yn
else	ant-a
except	ant-amarth
from	ant-
if	a-amarth
import	yn-
in	ant-thargáen
is	teulu
raise	thargáen-a-amarth
return	amarth-a
try	a-yn-yn
while	i-thargáen
with	amarth-thargáen
print	ensíla

# Automato



# Exemplo

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
    from cryptography.hazmat.primitives import serialization
    from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.asymmetric import padding
    from cryptography.hazmat.primitives import serialization
    from cryptography.hazmat.primitives import serialization
        from cryptography.fernet import Fernet

def gerar_chave_simetrica():
    return Fernet.generate_key()

def codificar_mensagem(mensagem, chave):
    f = Fernet(chave)
    mensagem_bytes = mensagem.encode()
    ciphertext = f.encrypt(mensagem_bytes)
    return ciphertext

def decodificar_mensagem(mensagem_codificada, chave):
    f = Fernet(chave)
    mensagem_bytes = f.decrypt(mensagem_codificada)
    return mensagem_bytes.decode('utf-8')

mensagem = "Os orcs estão se aproximando! Preparem-se para a batalha!"

chave = gerar_chave_simetrica()
```

```
mensagem_codificada =
codificar_mensagem(mensagem, chave)

mensagem_decodificada =
decodificar_mensagem(mensagem_codificada,
                      chave)

print("Mensagem original:", mensagem)
print("Mensagem codificada:",
      mensagem_codificada)
print("Mensagem decodificada:",
      mensagem_decodificada)
```

# Aplicação

```
ant-yn cryptography.hazmat.primitives a-amarth hashes
ant-yn cryptography.hazmat.primitives.kdf.pbkdf2 a-amarth PBKDF2HMAC
    ant-yn cryptography.hazmat.primitives a-amarth serialization
    ant-yn cryptography.hazmat.primitives.asymmetric a-amarth rsa
ant-yn cryptography.hazmat.primitives.asymmetric a-amarth padding
    ant-yn cryptography.hazmat.primitives a-amarth serialization
    ant-yn cryptography.hazmat.primitives a-amarth serialization
        ant-yn cryptography.fernet a-amarth Fernet

    yn-amarth gerar_chave_simetrica():
        amarth-a Fernet.generate_key()

    yn-amarth codificar_mensagem(mensagem, chave):
        f = Fernet(chave)
        mensagem_bytes = mensagem.encode()
        ciphertext = f.encrypt(mensagem_bytes)
        amarth-a ciphertext

    yn-amarth decodificar_mensagem(mensagem_codificada, chave):
        f = Fernet(chave)
        mensagem_bytes = f.decrypt(mensagem_codificada)
        amarth-a mensagem_bytes.decode('utf-8')
```

```
chave = gerar_chave_simetrica()
mensagem_codificada =
codificar_mensagem(mensagem, chave)
mensagem_decodificada =
decodificar_mensagem(mensagem_codificada,
chave)
ensíla("Mensagem original:", mensagem)
ensíla("Mensagem codificada:",
mensagem_codificada)
ensíla("Mensagem decodificada:",
mensagem_decodificada)
```

# OBRIGADO

DE MATHEUS S.

