

Linguagem Computacional

Bruna Mayumi Kimura

Insper - Engenharia de computação - 7º Semestre
Lógica da Computação - Prof. Raul Ikeda

Introdução

Projeto

Esse projeto consiste em fazer uma linguagem computacional com alguma finalidade distinta.

-

O projeto foi separado em três partes:

- EBNF da linguagem
- Bison e Flex
- Compilador

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines, with some nodes highlighted in blue and others in grey.

1.

Motivação

Motivação para criação da linguagem



“

*Essa linguagem tem como objetivo
evitar o problema de **não saber**
onde estão os **caracteres especiais**
em determinados teclados.*



AZERTY

QWERTY (PT)



QWERTY (EUA)

Maior Produtividade

Dessa forma, é possível
programar sem perder a
produtividade tentando
achar os caracteres
especiais



Outros benefícios

- ◎ Evita procurar os caracteres especiais
- ◎ É programável em qualquer tipo de teclado
- ◎ Aumenta a produtividade em teclados diferentes do usual
- ◎ Não há necessidade de utilizar o *ctrl* ou *shift*

Um ponto negativo é o tamanho da palavra (um caracter especial é substituído por uma palavra completa). Além do código ficar mais poluído visualmente.

Diferença entre os caracteres especiais entre teclado português e inglês

pt	en	pt	en	pt	en	pt	en	pt	en	pt	en	pt	en	pt	en	pt	en
‘	`	#	#	&	&	-	-	‘	[ç	;]	\	.	.	/	/
“	~	\$	\$	*	*	—	—	`	{	Ç	:	}		>	>	?	?
!	!	%	%	((=	=	[]	~	‘	,	,	;	/	\	\
@	@	”	^))	+	+	{	}	^	“	<	<	:	?		

Legenda:

Ao selecionar uma determinada tecla o resultado é mostrado primeiro na coluna **azul** (pt) e então no **rosa** (en).

As colunas **vermelhas** correspondem aos valores que destoaram nesses dois teclados.

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles inside, suggesting a hierarchical or multi-layered structure. The lines are thin and gray, connecting the nodes in a non-linear fashion.

2. **Inspiração**

Modelo pela qual essa linguagem foi baseada



VBA

Essa linguagem foi adaptada da linguagem VBA. Assim, a estrutura da língua permanece igual, porém sem os caracteres especiais.

Mudanças na linguagem (tokens alterados)

(→ openp

) → closep

+ → plus

- → minus

* → mult

/ → div

= → assgmt

> → bigger

< → less

, → comma

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are highlighted with a double-circle outline. The lines are thin and gray, creating a mesh-like structure.

3. **EBNF**

EBNF baseada no VBA

EBNF

Esse documento foi baseado na EBNF do VBA. Apenas alguns tokens foram alterados. Para melhor visualização ver arquivo [ebnf_bruna.pdf](#)

```
<additive-expression> ::= <multiplicative-expression>  
    | <additive-expression> plus <multiplicative-expression>  
    | <additive-expression> minus <multiplicative-expression>  
  
<multiplicative-expression> ::= <cast-expression>  
    | <multiplicative-expression> mult <cast-expression>  
    | <multiplicative-expression> div <cast-expression>  
    | <multiplicative-expression> rest <cast-expression>  
  
<cast-expression> ::= <unary-expression>  
    | ( <type-name> ) <cast-expression>  
  
<unary-expression> ::= <postfix-expression>  
    | pp <unary-expression>  
    | mm <unary-expression>  
    | <unary-operator> <cast-expression>  
    | sizeof <unary-expression>  
    | sizeof <type-name>  
  
<primary-expression> ::= <identifier>  
    | <constant>  
    | <string>  
    | ( <expression> )  
  
<constant> ::= <integer-constant>  
    | <character-constant>  
    | <floating-constant>  
    | <enumeration-constant>
```

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by circles of varying sizes, some with concentric rings, and the lines are thin and grey. The diagram is partially cut off by the top and left edges of the slide.

4. **Bison e Flex**

```

identifier string integer constant character const
%%

external-declaration:
    | external-declaration function
    | external-declaration declarator
    ;
function-definition: declaration-specifier declarator
    ;
declaration-specifier:
    | type-specifier
    ;
type-specifier: int
    | bool
    ;
specifier-qualifier: type-specifier
    ;
declarator: direct-declarator
    ;
direct-declarator: identifier
    | open_p declarator close_p
    | direct-declarator
    | direct-declarator constant-expr
    | direct-declarator open_p parameter_list

```

```

%{
#include <stdio.h>
%}

identifier [a-zA-Z_][a-zA-Z0-9_]*
string "[a-zA-Z_][a-zA-Z0-9_]*"
integer-constant [0-9]*
character-constant [a-zA-Z_]*
floating-constant [0-9]*\.[0-9]

%%
int      TOKEN(int)
bool     TOKEN(bool)

or       TOKEN(or)
and      TOKEN(and)
assgmt   TOKEN(assgmt)
bigger   TOKEN(bigger)
less     TOKEN(less)
plus     TOKEN(plus)
minus    TOKEN(minus)
mult     TOKEN(mult)

```

Bison e Flex

O Bison foi feito a partir da EBNF. O Flex possui apenas os tokens utilizado. Para melhor visualização ver arquivos parser.y (Bison) e lexico.l (Flex).

Compilando o Flex e Bison

Bison

Para compilar o bison é necessário instalar seu compilador

```
Apt install bison
```

Já para compilar:

```
bison parser.y
```

Esse comando irá gerar um arquivo *parser.tab.c*

Flex

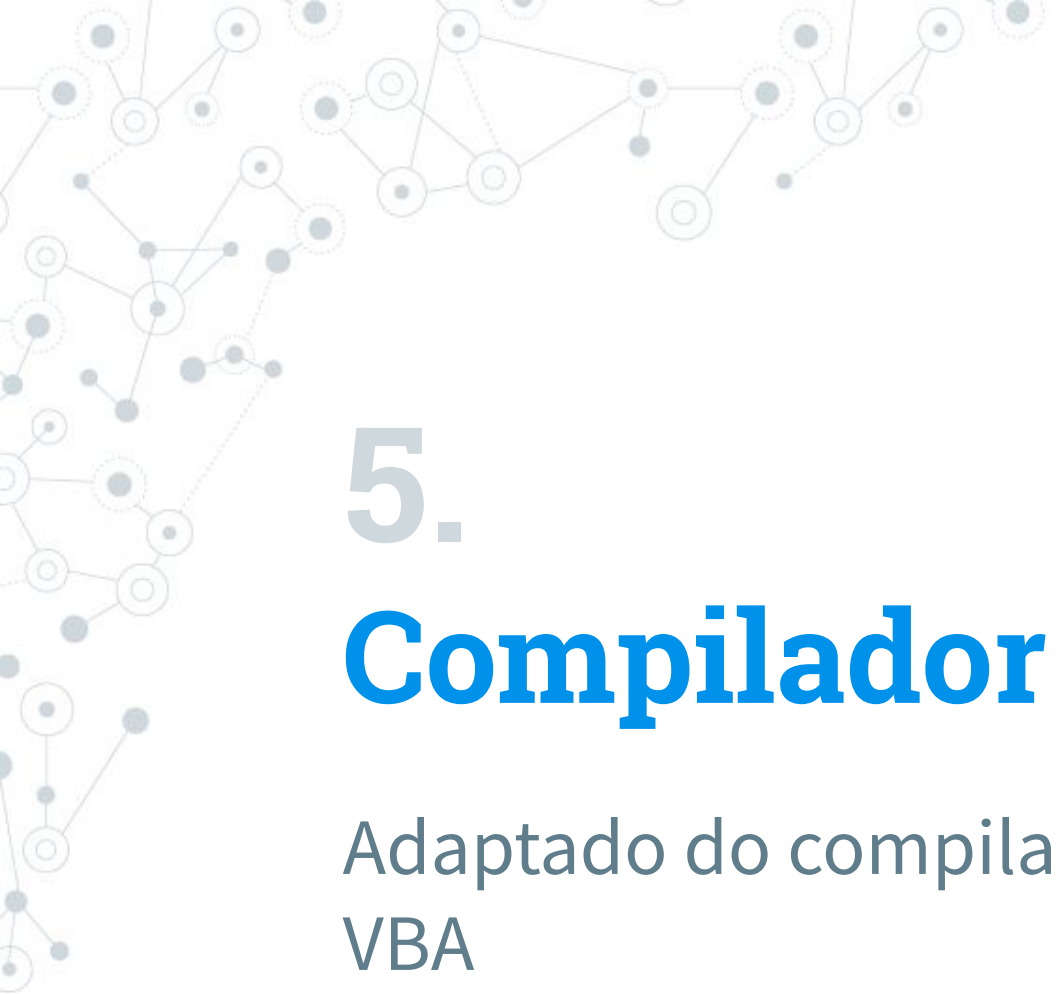
Da mesma forma que o bison, primeiro é necessário instalar o flex

```
Apt install flex
```

Já para compilar:

```
flex lexico.l
```

Esse comando irá gerar um arquivo *lex.yy.c*

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are shaded in light blue, and the lines are thin and grey.

5. **Compilador**

Adaptado do compilador simples de
VBA

Compilador da linguagem

O compilador dessa linguagem foi feito totalmente em python. Além disso ele foi baseado no compilador de VBA, feito nessa mesma matéria.

```
[  
    tokens.actual.type == 'SUB':  
        tokens.selectNext()  
    Parser.tokens.actual.type == 'identifier':  
        nome_func = Parser.tokens.actual.value  
        Parser.tokens.selectNext()  
    if Parser.tokens.actual.type == '(':  
        Parser.tokens.selectNext()  
  
    while Parser.tokens.actual.type != ')':  
        if Parser.tokens.actual.type == 'identi  
            variavel = Parser.tokens.actual.va  
            Parser.tokens.selectNext()  
        if Parser.tokens.actual.type ==  
            Parser.tokens.selectNext()  
        if Parser.tokens.actu  
            lista varia
```

Executando o compilador

Python

Para executar o python basta digitar:

```
Python3 main.py teste.vba
```

E o programa irá printar o resultado no próprio terminal

Entrada

A entrada do programa é um arquivo *.vbs*. Como exemplo de entrada existe um arquivo *teste.vbs*, que pode ser livremente alterado respeitando a EBNF da linguagem. Para executar o programa é necessário passar a entrada na linha de comando como mostrado acima.

Obrigada!

Mais informações:



[brunakimura/LogiComp_Linguagem](https://github.com/brunakimura/LogiComp_Linguagem)