



LINGUAGEM DE PROGRAMAÇÃO GROOTCODE

Alexandre Almeida Edington

MOTIVAÇÃO

- Aprender o passo a passo de como criar uma nova linguagem;
- Criar uma linguagem baseada uma língua com poucas palavras



I AM GROOT

PALAVRAS RESERVADAS

03

Conceitos básicos da linguagem

Definição de variável

Definição de função

Return de função

While

If

Elseif

Else

End

Print

True

False



I_GROOT

I_AM_GROOT

REGROOT

GROOT?!

GROOT?

NOT_GROOT?

NOT_GROOT

GROOT

GROOT:

groot

!groot

BLOCK = { COMMAND | FUNCDEC };

COMMAND = VARDEC | VAREQUAL | FUNCCALL | PRINT | WHILE | IF |
RETURN;

FUNCDEC = "I_AM_GROOT", NAME, "(", {NAME, "::", TYPE}, ")", "::", TYPE,
{COMMAND}, "GROOT";

VARDEC = "I_GROOT", NAME, "::", TYPE;

VAREQUAL = NAME, "=", RELEXP | "readline()";

FUNCCALL = NAME, "(", {RELEXP, ","}, ")";

PRINT = "println(", RELEXP, ")";

WHILE = "GROOT?!", RELEXP, {COMMAND}, "GROOT";

IF = "GROOT?", RELEXP, {COMMAND}, ELSE | ELSEIF | "", "GROOT";

ELSE = "NOT_GROOT", {COMMAND};

ELSEIF = "NOT_GROOT?", RELEXP, {COMMAND}, ELSE | ELSEIF | "",
"GROOT";

RETURN = "REGROOT", RELEXP;

RELEXP = EXP, { "==" | ">" | "<", EXP};

EXP = TERM, { "+" | "-" | "||", TERM};

TERM = FACTOR, { "*" | "/" | "&&", FACTOR};

FACTOR = NUMBER | STRING | BOOL | ("+", FACTOR) | ("-", FACTOR) | ("!",
FACTOR) | ("(", RELEXP, ")") | NAME | FUNCCALL;

NUMBER = DIGIT, { DIGIT };

STRING = "", ((' ' | LETTER | DIGIT | " "), { (' ' | LETTER | DIGIT | " ") }), "";

NAME = ((' ' | LETTER | DIGIT | " "), { (' ' | LETTER | DIGIT | " ") });

LETTER = (a | ... | z | A | ... | Z);

DIGIT = (1|2|3|4|5|6|7|8|9|0);

TYPE = "Bool" | "Int" | "String";

BOOL = ('groot' | '!groot');



Pontos Fortes

- Groot
- Não utiliza ";" para delimitação de linhas
- Suporta funções recursivas
- Semelhante à linguagens como Julia

Pontos Fracos

- !Groot
- Tipos limitados de variáveis
- Linguagem pouco intuitiva para iniciantes
- Linguagem compilada com python

Compilação

GrootCode trabalha com arquivos .gr

É necessário python 3.7+para o compilador

Como Compilar

No terminal:

```
$ python GrootCodeCompilador.py seu_arquivo.gr
```

Código de exemplo

Para demonstrar a linguagem foi criado um código de exemplo recebe um número e determina se ele é primo.

```
PS C:\Users\alexa\OneDrive\Área de Trabalho\GrootCode> python GrootCodeCompilador.py teste.gr
Digite um numero para descobrir se e primo.
471
Esse numero nao e primo.
```

```
PS C:\Users\alexa\OneDrive\Área de Trabalho\GrootCode> python GrootCodeCompilador.py teste.gr
Digite um numero para descobrir se e primo.
523
Esse numero e primo.
```

Utilizaremos esse exemplo para aprender como a linguagem funciona.

Variáveis

Para inicializar uma variável é preciso escrever "I_GROOT", seguido pelo nome da variável, "::" e seu tipo.

Os tipos suportados são: Int, Bool e String.

```
I_GROOT x::Int  
I_GROOT r::Bool
```

Uma vez que a variável foi inicializada, é possível mudar seu valor livremente, com tanto que obedeça o tipo estabelecido.

```
r = groot
```

Funções

Para definir uma função, começamos com "I_AM_GROOT", seguido do nome da função, os elementos que ela recebe entre parênteses e por fim o tipo da função.

```
I_AM_GROOT checaPrimo(x:Int)::Bool
```

Para uma chamada de função, podemos chamá-la como na figura abaixo, assumindo que ela retorne um valor, ou como uma chamada de sistema, tal como "GROOT:".

```
r = checaPrimo(x)
```

While

O while é definido pelo "GROOT?!", seguido pela condição a ser cumprida. Desse ponto em diante, todas as linhas estão dentro do while até o seu respectivo "GROOT" (end).

```
GROOT?! i<x

    j = x/i
    j = j*i
GROOT? j==x
    r = !groot
GROOT

    i = i + 1
GROOT
```

Nesse caso existe um
if dentro do while

For

O for segue a mesma estrutura do while, mas é nomeado por "GROOT?".

```
GROOT? r
|   GROOT:("Esse numero e primo.")
```

O else "NOT_GROOT" segue apenas com as linhas consideradas como else até o "GROOT" (end) no final.

```
GROOT? r
|   GROOT:("Esse numero e primo.")
NOT_GROOT
|   GROOT:("Esse numero nao e primo.")
GROOT|
```

É possível, também, criar um elseif, com "NOT_GROOT?" seguido pela condição.

Interação com o terminal

Para mostrar algo no terminal, utiliza-se a função "GROOT:", com o que deseja mostrar em parênteses.

```
GROOT:("Esse numero e primo.")
```

Para receber um valor do terminal, utiliza-se a função "readline()", como no exemplo abaixo.

```
x = readline()
```

MELHORIAS FUTURAS

- Doubles
- Vetores
- Loops do tipo for
- Geração de executável através do compilador