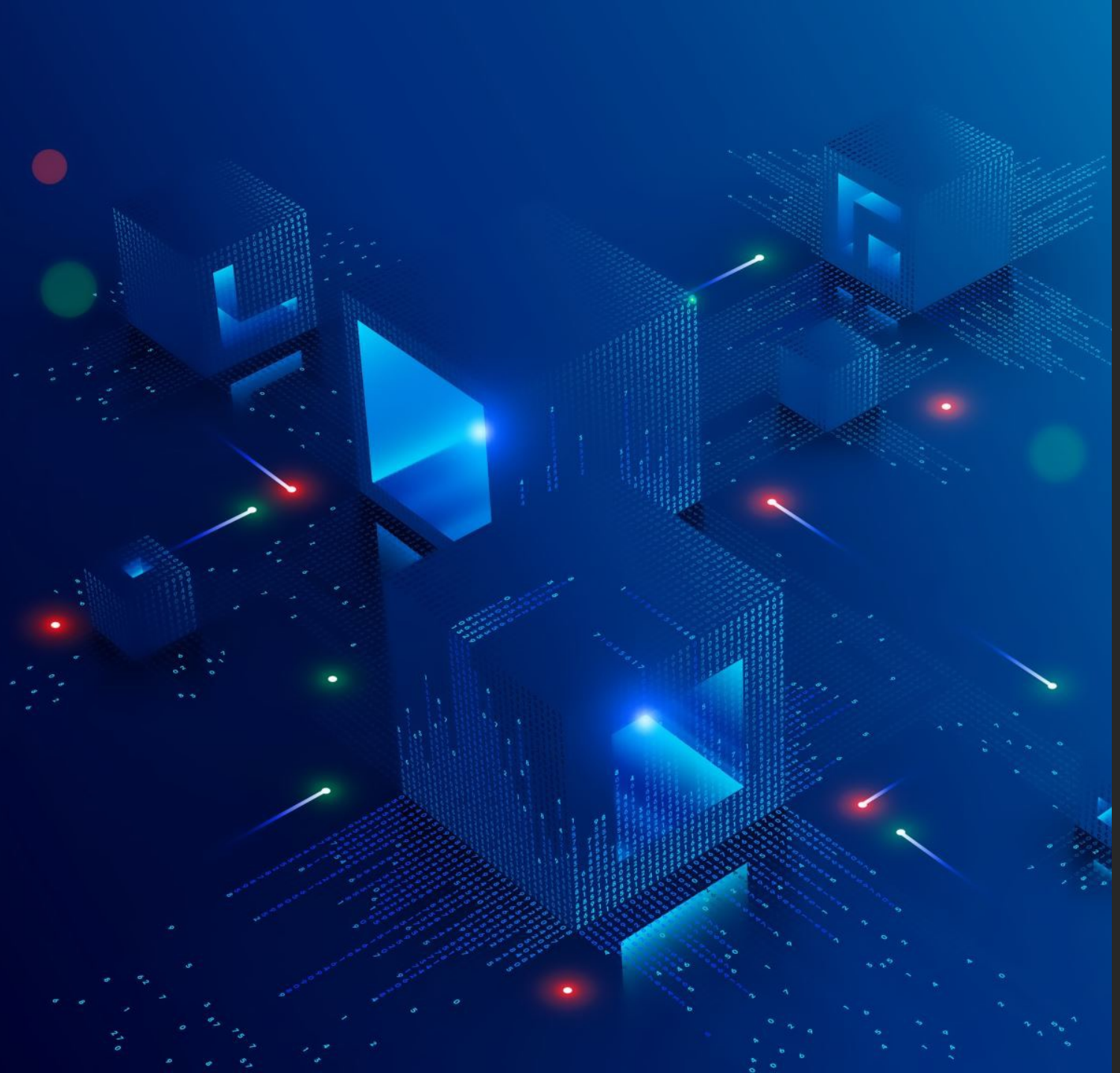

APS
Lógica da Computação

Paulo Wook Kim

Prof. Raul Ikeda

Motivação

- A motivação para este projeto nasceu da vontade de explorar as possibilidades da linguagem de programação Julia e de simplificar sua sintaxe para torná-la mais acessível. Com base em Julia, foi criado uma linguagem de programação com palavras do português substituindo símbolos matemáticos, tornando a linguagem mais fácil de ler e compreender para iniciantes

+	→	MAIS
-	→	MENOS
if	→	SE
while	→	ENQUANTO
&&	→	E
||	→	OU

Características

- Esta linguagem baseada em Julia mantém muitas das poderosas funcionalidades de Julia, como o suporte para programação de alto nível e funções matemáticas complexas. Além disso, os símbolos matemáticos foram substituídos por palavras e os termos técnicos foram traduzidos para o português. O código resultante se parece mais com frases em português do que com código de computador, o que pode facilitar a aprendizagem e a adoção por novos programadores

```
BLOCK = { STATEMENT };
STATEMENT = ( λ | ASSIGNMENT | PRINT | VARDEC | FUNCCALL | RETURN | IF | WHILE | FUNCDEC ), "\n" ;
ASSIGNMENT = IDENTIFIER, "IGUAL", RELEXP ;
VARDEC = IDENTIFIER, "COMO", TYPE, [ "IGUAL", RELEXP ] ;
FUNCCALL = IDENTIFIER, "(", { RELEXP, "," }, ")" ;
RETURN = "RETORNE", RELEXP ;
IF = "SE", RELEXP, BLOCK, [ "SENAO", BLOCK ], "FIM" ;
WHILE = "ENQUNTO", RELEXP, BLOCK, "FIM" ;
FUNCDEC = "FUNCAO", IDENTIFIER, "(", { VARDEC, "," }, ")", "COMO", TYPE, BLOCK, "FIM" ;
PRINT = "IMPRIMA", "(", RELEXP, ")" ;
RELEXP = EXPRESSION, { ("IDENTICO" | "MAIOR" | "MENOR" ), EXPRESSION } ;
EXPRESSION = TERM, { ("MAIS" | "MENOS" | "OU"), TERM } ;
TERM = FACTOR, { ("MULTIPLICADO" | "DIVIDO" | "E"), FACTOR } ;
FACTOR = (("MAIS" | "MENOS" | "CONTRARIO"), FACTOR) | NUMBER | STRING | "(", RELEXP, ")" | FUNCCALL | IDENTIFIER ;
IDENTIFIER = LETTER, { LETTER | DIGIT | "_" } ;
NUMBER = DIGIT, { DIGIT } ;
STRING = """, { LETTER | DIGIT | "_" }, """ ;
TYPE = ("INTEIRO" | "CARACTERES") ;
LETTER = ( a | ... | z | A | ... | Z ) ;
DIGIT = ( 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 ) ;
```

Curiosidades

- Uma das curiosidades desta linguagem é que, apesar de sua simplicidade, ela é capaz de executar funções complexas e poderosas. Além disso, por ser baseada em Julia - uma linguagem de programação dinâmica de alto nível, ela mantém muitas das vantagens dessa linguagem. Isso inclui a habilidade de escrever código de alto desempenho, o que é raro para linguagens com sintaxe tão simples

```
FUNCAO soma{x COMO INTEIRO, y COMO INTEIRO} COMO INTEIRO
  RETORNE x MAIS y
FIM

x_1 COMO INTEIRO
x_1 IGUAL 2
x_1 IGUAL soma{1, x_1}

x_1 IGUAL 3
SE {x_1 MAIOR 1} E CONTRARIO {x_1 MENOR 1}
  x_1 IGUAL 3
SENAO
  x_1 IGUAL {MENOS 20 MAIS 30} MULTIPLICADO 4 MULTIPLICADO 3 DIVIDIDO 40
FIM
IMPRIMA{x_1}
x_1 IGUAL 0

SE {x_1 MAIOR 1} E CONTRARIO {x_1 MENOR 1}
  x_1 IGUAL 3
SENAO
  x_1 IGUAL {MENOS 20 MAIS 30} MULTIPLICADO 12 DIVIDIDO 40
FIM
IMPRIMA{x_1}

ENQUANTO {{x_1 MAIOR 1} OU {x_1 IDENTICO 1}}
  x_1 IGUAL x_1 MENOS 1
  IMPRIMA{x_1}
FIM
```

Exemplo

- Vamos examinar alguns exemplos de código nesta linguagem. Aqui está um exemplo de como definir e chamar uma função, fazer uma declaração de variável, usar uma instrução condicional 'SE' e um loop 'ENQUANTO'. Como você pode ver, a sintaxe é muito semelhante ao português padrão, tornando o código fácil de ler e entender, mesmo para aqueles que não têm experiência em programação.

Obrigado!