

Programação Vetorial

Tipos primitivos de dados e funções

Prof. Edson Alves

Faculdade UnB Gama

Sumário

1. Tipos primitivos de dados
2. *Arrays*

Expressões em APL

- ▶ APL pode ser vista como uma notação matemática que também é executável por máquina

Expressões em APL

- ▶ APL pode ser vista como uma notação matemática que também é executável por máquina
- ▶ A linguagem é composta por funções, operadores, *arrays* e atribuições

Expressões em APL

- ▶ APL pode ser vista como uma notação matemática que também é executável por máquina
- ▶ A linguagem é composta por funções, operadores, *arrays* e atribuições
- ▶ Qualquer código que pode ser aplicado a dados é chamado função

Expressões em APL

- ▶ APL pode ser vista como uma notação matemática que também é executável por máquina
- ▶ A linguagem é composta por funções, operadores, *arrays* e atribuições
- ▶ Qualquer código que pode ser aplicado a dados é chamado função
- ▶ Dois exemplos de funções seriam a adição (+) e subtração (-)

Expressões em APL

- ▶ APL pode ser vista como uma notação matemática que também é executável por máquina
- ▶ A linguagem é composta por funções, operadores, *arrays* e atribuições
- ▶ Qualquer código que pode ser aplicado a dados é chamado função
- ▶ Dois exemplos de funções seriam a adição (+) e subtração (-)
- ▶ As funções de APL podem ser aplicadas monadicamente (prefixada, um operando) ou diadicamente (infixada, dois operando, um à esquerda e outro à direita)

Expressões em APL

- ▶ APL pode ser vista como uma notação matemática que também é executável por máquina
- ▶ A linguagem é composta por funções, operadores, *arrays* e atribuições
- ▶ Qualquer código que pode ser aplicado a dados é chamado função
- ▶ Dois exemplos de funções seriam a adição (+) e subtração (-)
- ▶ As funções de APL podem ser aplicadas monadicamente (prefixada, um operando) ou diadicamente (infixada, dois operando, um à esquerda e outro à direita)
- ▶ O tipo de dados mais elementar é o escalar (*array* de dimensão zero)

Inteiros

- ▶ Números são tratados internamente pela APL quanto ao tamanho e tipo e podem ser misturados sem problemas

Inteiros

- ▶ Números são tratados internamente pela APL quanto ao tamanho e tipo e podem ser misturados sem problemas
- ▶ Em APL os números podem ser inteiros, reais (em ponto flutuante) e números complexos
- ▶ Um escalar inteiro pode ser grafado usando a notação decimal padrão:

2 + 3

5

2 × 3

A a multiplicação é realizada pela função ×

6

Inteiros

- ▶ Números são tratados internamente pela APL quanto ao tamanho e tipo e podem ser misturados sem problemas
- ▶ Em APL os números podem ser inteiros, reais (em ponto flutuante) e números complexos
- ▶ Um escalar inteiro pode ser grafado usando a notação decimal padrão:

2 + 3

5

2 × 3

`⌘` a multiplicação é realizada pela função `×`

6

- ▶ Comentários são precedidos pelo símbolo `⌘`

Inteiros

- ▶ Números são tratados internamente pela APL quanto ao tamanho e tipo e podem ser misturados sem problemas
- ▶ Em APL os números podem ser inteiros, reais (em ponto flutuante) e números complexos
- ▶ Um escalar inteiro pode ser grafado usando a notação decimal padrão:

2 + 3
5

2 × 3 A a multiplicação é realizada pela função ×
6

- ▶ Comentários são precedidos pelo símbolo `A`
- ▶ Números negativos são precedidos pelo símbolo `-` (*macron*)

2 - 3
-1

Novo símbolo

Símbolo	Aridade	Descrição
$+$ (<i>plus</i>)	diádico	Adição escalar

Unicode	TAB	APL
U+002B	-	-

Novo símbolo

Símbolo	Aridade	Descrição
$-$ (<i>minus</i>)	diádico	Subtração escalar


Unicode	TAB	APL
U+002D	-	-

Novo símbolo

Símbolo	Aridade	Descrição
\times (<i>times</i>)	diádico	Multiplicação escalar

Unicode	TAB	APL
U+00D7	x x <tab>	APL + -

Novo símbolo

Símbolo	Aridade	Descrição
 (<i>macron</i>)	monádico	Prefixa um número negativo

Unicode	TAB	APL
U+00AF	- - <tab>	APL + 2

Novo símbolo

Símbolo	Aridade	Descrição
␣ (comment)	monádico	Inicia um comentário. Tudo que o sucede até o fim da linha será considerado comentário
Unicode	TAB	APL
U+235D	o n <tab>	APL + ,

Números reais

- ▶ Em escalares reais, a parte inteira é separada das casas decimais por meio do ponto final

0.2 ÷ 3.5
0.05714285714

Números reais

- ▶ Em escalares reais, a parte inteira é separada das casas decimais por meio do ponto final

```
0.2 ÷ 3.5  
0.05714285714
```

- ▶ APL também trata problemas de precisão de forma transparente ao usuário

```
2÷3 ♦ 6 × (2 ÷ 3)  
0.6666666667  
4
```

Números reais

- ▶ Em escalares reais, a parte inteira é separada das casas decimais por meio do ponto final

```
0.2 ÷ 3.5  
0.05714285714
```

- ▶ APL também trata problemas de precisão de forma transparente ao usuário

```
2÷3 ♦ 6 × (2 ÷ 3)  
0.6666666667  
4
```

- ▶ O símbolo ♦ (*diamond*) separa duas expressões em uma mesma linha

Números reais

- ▶ Em escalares reais, a parte inteira é separada das casas decimais por meio do ponto final

```
0.2 ÷ 3.5
0.05714285714
```

- ▶ APL também trata problemas de precisão de forma transparente ao usuário

```
2÷3 ♦ 6 × (2 ÷ 3)
0.6666666667
4
```

- ▶ O símbolo ♦ (*diamond*) separa duas expressões em uma mesma linha
- ▶ Notação científica pode representar números muito pequenos ou grandes


```
2E-3 ♦ 5e7      A O E pode ser maiúsculo ou minúsculo
0.002
50000000
```

Novo símbolo

Símbolo	Aridade	Descrição
\div (<i>divide</i>)	diádico	Divisão escalar. Divisão por zero resulta em um erro

Unicode	TAB	APL
U+00F7	: - <tab>	APL + =

Novo símbolo

Símbolo	Aridade	Descrição
 (<i>diamond</i>)	diádico	Separador de expressões

Unicode	TAB	APL
U+22C4	< > <tab>	APL + '

Constantes booleanas

- ▶ Em APL: falso é igual a 0 (zero) e verdadeiro é igual a 1 (um)

2 = 3

0

5 = 5.0

1

Constantes booleanas

- ▶ Em APL: falso é igual a 0 (zero) e verdadeiro é igual a 1 (um)

2 = 3

0

5 = 5.0

1

- ▶ Os operadores relacionais retornam valores booleanos

2 ≠ 3

0

5 < 7

1

11 > 13

0

17 ≤ 19 ♦ 23 ≥ 27

1

0

Novo símbolo

Símbolo	Aridade	Descrição
= (<i>equal</i>)	diádico	Igual a


Unicode	TAB	APL
U+003D	-	APL + 5

Novo símbolo

Símbolo	Aridade	Descrição
\neq (<i>not equal</i>)	diádico	Diferente de

Unicode	TAB	APL
U+2260	= / <tab>	APL + 8

Novo símbolo

Símbolo	Aridade	Descrição
 (<i>less than</i>)	diádico	Menor que

Unicode	TAB	APL
U+003C	-	APL + 3

Novo símbolo

Símbolo	Aridade	Descrição
\gt (<i>greater than</i>)	diádico	Maior que

Unicode	TAB	APL
U+003E	-	APL + 7

Novo símbolo

Símbolo	Aridade	Descrição
\leq (<i>less than or equal to</i>)	diádico	Menor ou igual a

Unicode	TAB	APL
U+2264	< = <tab>	APL + 4

Novo símbolo

Símbolo	Aridade	Descrição
\geq (<i>greater than or equal to</i>)	diádico	Maior ou igual a

Unicode	TAB	APL
U+2265	> = <tab>	APL + 6

Números complexos

- ▶ O caractere 'J' separa a parte real da parte imaginária em números complexos

`2J3 × 5j-7`
`31J1`

A O J também pode ser minúsculo

Números complexos

- ▶ O caractere 'J' separa a parte real da parte imaginária em números complexos

```
2J3 * 5j^-7
31J1
```

¶ O J também pode ser minúsculo

- ▶ Lembre-se de que o argumento à direita de uma função diádica é o resultado de toda a expressão à direita do símbolo

```
2 * 3 + 5
16
```

¶ equivale a $2 \times (3 + 5)$

```
2 - 3 - 5 - 7 - 11
8
```

¶ $2 - (3 - (5 - (7 - 11)))$

```
2 ÷ 3 ÷ 5
3.333333333
```

¶ $10 \div 3$

```
2 * 0J1 * 3
0J^-2
```

¶ $2 \times (j \text{ elevado a } 3)$

Novo símbolo

Símbolo	Aridade	Descrição
$*$ (<i>power</i>)	diádico	Eleva o argumento à esquerda a potência indicada no argumento à direita

Unicode	TAB	APL
U+002A	-	APL + p

Caracteres e strings

- ▶ Em APL, strings são vetores de caracteres

Caracteres e strings

- ▶ Em APL, strings são vetores de caracteres
- ▶ Tanto caracteres quanto strings são delimitadas por aspas simples

```
'c'                A um caractere
c
'uma string'
uma string
```

Caracteres e strings


- ▶ Em APL, strings são vetores de caracteres
- ▶ Tanto caracteres quanto strings são delimitadas por aspas simples

```
'c'                A um caractere
c
'uma string'
uma string
```

- ▶ Atribuições podem ser feitas por meio do símbolo ←

```
s ← 'abacate'
'a' = s                A compara 'a' a cada caractere de s
1 0 1 0 1 0 0
s ≠ 'abacaxi'         A compara caracteres em posições correspondentes
0 0 0 0 0 1 1
```

Novo símbolo

Símbolo	Aridade	Descrição
 (<i>assign</i>)	diádico	Atribui o argumento à direita ao argumento à esquerda

Unicode	TAB	APL
U+2190	< - <tab>	APL + ' '

Funções aritméticas monádicas

- ▶ As funções aritméticas apresentadas até o momento tem versões monádicas

$+2j3$	A conjugado complexo
$2j^{-3}$	
$-^2$	A simétrico aditivo
2	
$\times 2j3$	A vetor unitário na direção do complexo
0.5547001962j0.8320502943	
$\div 2$	A inverso multiplicativo
0.5	
$\ast 2$	A e elevado ao quadrado
7.389056099	

Funções aritméticas monádicas

- ▶ As funções aritméticas apresentadas até o momento tem versões monádicas

$+2J3$	A conjugado complexo
$2J^{-3}$	
$-^2$	A simétrico aditivo
2	
$\times 2J3$	A vetor unitário na direção do complexo
$0.5547001962J0.8320502943$	
$\div 2$	A inverso multiplicativo
0.5	
$*2$	A e elevado ao quadrado
7.389056099	

- ▶ Quando aplicada a números reais, a função monádica \times corresponde à função `signum()` de muitas linguagens

Novo símbolo

Símbolo	Aridade	Descrição
$+$ (<i>conjugate</i>)	monádico	Conjugado complexo

Unicode	TAB	APL
U+002B	-	-

Novo símbolo

Símbolo	Aridade	Descrição
$-$ (<i>negate</i>)	monádico	Simétrico aditivo

Unicode	TAB	APL
U+002D	-	-

Novo símbolo

Símbolo	Aridade	Descrição
\times (<i>direction</i>)	monádico	Vetor unitário na direção do número

Unicode	TAB	APL
U+00D7	x x <tab>	APL + -

Novo símbolo

Símbolo	Aridade	Descrição
\div (<i>reciprocal</i>)	monádico	Inverso multiplicativo

Unicode	TAB	APL
U+00F7	: - <tab>	APL + =

Novo símbolo

Símbolo	Aridade	Descrição
$*$ (<i>exponential</i>)	monádico	e elevado ao argumento à direita

Unicode	TAB	APL
U+002A	-	APL + p

Referências

1. APL Wiki. [Quad name](#), acesso em 23/09/2021.
2. **BROCKLEBANK**, Daniel. *APL – The Language*, John Hopkins APL Technical Digest, vol. 5, number 3, 1984.
3. Dyalog. [Dyalog APL Version 12.0](#), acesso em 23/09/2021.
4. Dyalog. [Try APL](#), acesso em 23/09/2021.
5. **IVERSON**, Kenneth E. *A Programming Language*, John Wiley and Sons, 1962.
6. Stack Overflow. [Add APL Keyboard Layout On Linux 20.04](#), acesso em 23/09/2021.
7. **ULMANN**, Bernd. *APL - One of the Greatest Programming Languages Ever*, Vintage Computer Festival Europe 2007.
8. Xah Lee. [Unicode APL Symbols](#), acesso em 23/09/2021.
9. Wikipédia. [APL \(programming language\)](#), acesso em 24/09/2021.
10. Wikipédia. [Array programming](#), acesso em 24/09/2021.