

Programação Vetorial

Tipos primitivos de dados e funções

Prof. Edson Alves

Faculdade UnB Gama

Sumário

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

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

6

A multiplicação é realizada pela função ×

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`

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 `⍝`
- ▶ Números negativos são precedidos pelo símbolo `¯` (*macron*)

2 - 3
¯1

Novo símbolo

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

Unicode	TAB	APL
U+002B	-	-

Novo símbolo

Símbolo	Aridade	Descrição
<div><div>-</div><div>(<i>minus</i>)</div></div>	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
<div><div>—</div><div>(<i>macron</i>)</div></div>	monádico	Antecede 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

$0.2 \div 3.5$
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>÷</div> <div>(<i>divide</i>)</div>	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
<div>=</div> <div>(equal)</div>	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
> (<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

$2J3 \times 5j^{-7}$
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` A O J também pode ser minúsculo
`31J1`

- ▶ 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` A equivale a `2 * (3 + 5)`
`16`
`2 - 3 - 5 - 7 - 11` A `2 - (3 - (5 - (7 - 11)))`
`8`
`2 ÷ 3 ÷ 5` A `10 ÷ 3`
`3.333333333`
`2 * 0J1 * 3` A `2 * (j elevado a 3)`
`0J^-2`

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	
$*2$	A função exponencial
7.389056099	

Funções aritméticas monádicas

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

<code>+2J3</code>	A conjugado complexo
<code>2J^-3</code>	
<code>-^2</code>	A simétrico aditivo
<code>2</code>	
<code>×2J3</code>	A vetor unitário na direção do complexo
<code>0.5547001962J0.8320502943</code>	
<code>÷2</code>	A inverso multiplicativo
<code>0.5</code>	
<code>*2</code>	A função exponencial
<code>7.389056099</code>	

- ▶ Quando aplicada a números reais, a função monádica `×` 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
<div><div>-</div><div>(<i>negate</i>)</div></div>	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>÷</div> <div>(reciprocal)</div>	monádico	Inverso multiplicativo

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

Novo símbolo

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

Unicode	TAB	APL
U+002A	-	APL + p

- ▶ Em APL, a estrutura de dados fundamental é o *array*, e todos os dados estão contidos em *arrays*

Arrays

- ▶ Em APL, a estrutura de dados fundamental é o *array*, e todos os dados estão contidos em *arrays*
- ▶ Um *array* é uma coleção retangular de números, caracteres e *arrays*, arranjados ao longo de um ou mais eixos

Arrays

- ▶ Em APL, a estrutura de dados fundamental é o *array*, e todos os dados estão contidos em *arrays*
- ▶ Um *array* é uma coleção retangular de números, caracteres e *arrays*, arranjados ao longo de um ou mais eixos
- ▶ Os elementos de um *array* podem ter tipos distintos

- ▶ Em APL, a estrutura de dados fundamental é o *array*, e todos os dados estão contidos em *arrays*
- ▶ Um *array* é uma coleção retangular de números, caracteres e *arrays*, arranjados ao longo de um ou mais eixos
- ▶ Os elementos de um *array* podem ter tipos distintos
- ▶ *Arrays* especiais:
 - (a) **escalar**: um único número, dimensão zero
 - (b) **vetor**: um *array* unidimensional
 - (c) **matriz**: um *array* bidimensional

Declaração de *arrays*

- ▶ *Arrays* são declarados separando seus elementos por espaços

```
2 3 5 7 11
2 3 5 7 11
'string' 2.0 3J-5 'c' 7
string 2.0 3J-5 c 7
```

Declaração de *arrays*

- ▶ *Arrays* são declarados separando seus elementos por espaços

```
2 3 5 7 11
2 3 5 7 11
'string' 2.0 3J-5 'c' 7
string 2.0 3J-5 c 7
```

- ▶ Parêntesis podem ser utilizados para agrupar vetores

```
(2 3 5) (7 11) (13) (17 19 23)
2 3 5 7 11 13 17 19 23
((2 3 5) (7 11)) ((13))
2 3 5 7 11 13
10
1 2 3 4 5 6 7 8 9 10
```

A gera os 10 primeiros naturais

Novo símbolo

Símbolo	Aridade	Descrição
ι (<i>iota</i>)	monádico	Gera os primeiros n naturais

Unicode	TAB	APL
U+2373	i i <tab>	APL + i

Profundidade

- ▶ A profundidade (*depth*) de um *array* corresponde a o seu nível de profundidade/recursão

Profundidade

- ▶ A profundidade (*depth*) de um *array* corresponde a o seu nível de profundidade/recursão
- ▶ um vetor de escalares tem profundidade igual a 1

Profundidade

- ▶ A profundidade (*depth*) de um *array* corresponde a o seu nível de profundidade/recursão
- ▶ um vetor de escalares tem profundidade igual a 1
- ▶ um vetor cujos elementos são vetores de profundidade 1 tem profundidade igual a 2

Profundidade

- ▶ A profundidade (*depth*) de um *array* corresponde a o seu nível de profundidade/recursão
- ▶ um vetor de escalares tem profundidade igual a 1
- ▶ um vetor cujos elementos são vetores de profundidade 1 tem profundidade igual a 2
- ▶ um escalar tem profundidade zero

Profundidade

- ▶ A profundidade (*depth*) de um *array* corresponde a o seu nível de profundidade/recursão
- ▶ um vetor de escalares tem profundidade igual a 1
- ▶ um vetor cujos elementos são vetores de profundidade 1 tem profundidade igual a 2
- ▶ um escalar tem profundidade zero
- ▶ APL atribuí a um vetor que mistura escalares e vetores uma profundidade negativa

Profundidade

- ▶ A profundidade de um *array* pode ser obtida por meio da função \equiv

```

       $\equiv$  2
0
       $\equiv$  'string'           A 'string' = 's' 't' 'r' 'i' 'n' 'g'
1
       $\equiv$  ((2 3) (5 7 11)) ('um' 'dois' 'três')
3
       $\equiv$  2 'três'
-2
```

Profundidade


- ▶ A profundidade de um *array* pode ser obtida por meio da função \equiv

```
≡ 2
0
≡ 'string'      A 'string' = 's' 't' 'r' 'i' 'n' 'g'
1
≡ ((2 3) (5 7 11)) ('um' 'dois' 'três')
3
≡ 2 'três'
-2
```

- ▶ Strings vazias são representadas por ''

```
≡ ''
1
```

Novo símbolo

Símbolo	Aridade	Descrição
 (<i>depth</i>)	monádico	Retorna a profundidade do <i>array</i>

Unicode	TAB	APL
U+2261	= = <tab>	APL + Shift + ç

- ▶ O *rank* é definido como o número de dimensões de um *array*

Rank

- ▶ O *rank* é definido como o número de dimensões de um *array*
- ▶ Escalares tem *rank* igual a zero

Rank

- ▶ O *rank* é definido como o número de dimensões de um *array*
- ▶ Escalares tem *rank* igual a zero
- ▶ Vetores tem *rank* igual a 1

Rank

- ▶ O *rank* é definido como o número de dimensões de um *array*
- ▶ Escalares tem *rank* igual a zero
- ▶ Vetores tem *rank* igual a 1
- ▶ Matrizes tem *rank* igual a 2

- ▶ O *rank* é definido como o número de dimensões de um *array*
- ▶ Escalares tem *rank* igual a zero
- ▶ Vetores tem *rank* igual a 1
- ▶ Matrizes tem *rank* igual a 2
- ▶ Em APL os *arrays* são retangulares: cada linha de uma matriz deve ter o mesmo número de colunas

- ▶ O *rank* é definido como o número de dimensões de um *array*
- ▶ Escalares tem *rank* igual a zero
- ▶ Vetores tem *rank* igual a 1
- ▶ Matrizes tem *rank* igual a 2
- ▶ Em APL os *arrays* são retangulares: cada linha de uma matriz deve ter o mesmo número de colunas
- ▶ Para criar *arrays* com rank maior do que 1 é preciso usar a função ρ (*reshape*), que recebe como argumento à esquerda um vetor dos comprimentos das dimensões e os dados como argumento à direita

Novo símbolo

Símbolo	Aridade	Descrição
ρ (<i>reshape</i>)	diádico	Retorna um <i>array</i> com as dimensões e dados indicados
Unicode	TAB	APL
U+2374	r r <tab>	APL + r
