Universidade Nova de Lisboa
OMNIS CIVITAS CONTRA SE DIVISA NON STABIT
Faculdade de Ciências e Tecnologia

Introdução (Informal) à Programação

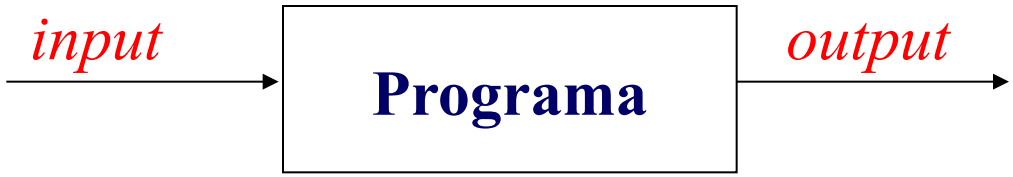
Pedro Barahona
DI/FCT/UNL

Métodos Computacionais

1º Semestre 2016/2017

Programa e Algoritmo

- Um **programa** é um conjunto de instruções que aplicadas aos dados de entrada (input) e a outros intermédios auxiliares produz um resultado (output).



- Um programa é a materialização para uma dada máquina (computador e linguagem de programação) de um **algoritmo**.

Níveis de Abstração

- Um programa pode ser entendido a vários níveis de abstracção (detalhe).
- Quanto mais "inteligente/conhecedor" for o interlocutor, a mais alto nível podem ser dadas as instruções.

Vá para Lisboa ⟺

```
Saia da sala
Saia do Edifício
Dirija-se à portaria
Vá para a paragem de autocarro
Apanhe o autocarro

....
```

Níveis de Abstração

Nível Máquina e Nível "Humano"

- Um computador só executa instruções extremamente simples. Por exemplo:
  - Transferir palavras (binárias) entre a memória e os registos
  - Processar dados nos registos (p.ex. soma binária)
- Um programador humano raciocina a um nível mais alto de abstração.

$C \leftarrow A + B$ ⟺

```
LDA 1005
LDB 22345A91
ADD A,B
STA 1234FE88
```

Linguagens de Programação

- As linguagens de programação permitem a um utilizador especificar um programa de uma forma semelhante ao algoritmo.

- Um compilador/interpretador da linguagem deverá fazer a tradução das intruções de alto nível para as de nível máquina (por exemplo, manter os endereços de memória onde estão guardadas as variáveis).

$C \leftarrow A + B$

$\Longleftrightarrow$

```
LDA 11A810A0
LDB 22345A91
ADD A,B
STA 1234FE88
```

Linguagens de Programação (2)

- Existem vários tipos de linguagens de programação baseadas em diferentes paradigmas (estilos) de programação.
  - **Linguagens imperativas:** Fortran, Pascal, C, Octave/MATLAB
    - Controle explícito da execução
  - Linguagens Orientadas por Objectos: Smalltalk, C++, Java, Python
    - Algum controle implícito na manipulação dos objectos
  - Linguagens Funcionais: LISP, Scheme
    - Baseadas na especificação de funções
  - Linguagens Lógicas: Prolog
    - Implementando a Lógica de Predicados

Programação Imperativa

- No paradigma de programação imperativa, o programador especifica explicitamente o controle de execução, isto é, a sequenciação das instruções base.

- Informalmente podemos considerar as seguintes instruções base, na especificação de algoritmos:

  - Afectação: A ←Expressão
    - A variável A toma o valor da Expressão
  - Entrada: Entra A
    - A variável A toma um valor dado do exterior (teclado, ficheiro)
  - Saída: Sai A
    - A variável A é passada para o exterior (monitor, ficheiro)

Controle de Execução

- A ordem pela qual as várias instruções são executadas é controlada explicitamente por instruções de
  - **Sequência**
  - **Execução Condicional**
  - **Execução Repetida**

- **Sequência** (“;”)
  - Exemplo: Equação do 1º grau ($a_1 x + a_0 = 0$)

```
entra a1;          % O valor de a1 é "entrado". Seja a1 = 2
entra a0;          % O valor de a0 é "entrado". Seja a0 = -6
 c ← - a0/a1;      % A solução é calculada na variável c
sai c ;            % O valor de c é passado para o exterior
```

Controle de Execução

- **Execução condicional** ("se")
  - Exemplo:

```
entra x ;                    % o valor de x é "entrado"
se x > 0 então
   y ← x                     % a variável y toma o
senão                        % valor da variável x
   y ← - x                   % ou o seu simétrico
fim se;
sai y ;                      %  o valor de y é comunicado
```

  - Que função de x é calculada?

Controle de Execução

- **Execução Repetida** (“enquanto”)
  - Exemplo:

```
entra n;                          % o valor de n é “entrado”
f ← 1;                            % o valor de f é inicializado a 1
i = 1;                            % o valor de i é inicializado a 1
enquanto i <= n fazer
    f ← f * i;                    % a variável f vai tomaindo valores
    i ← i + 1;                    % 1, 1*2, 1*2*3, .... , 1*2*3*...*n
fim enquanto;
sai f;                            % o valor de f é comunicado
```

  - Que função de f (dependente de n) é calculada?

Tipos de Dados Simples

- Às variáveis usadas têm sido atribuídos valores inteiros. No entanto podem ser considerados outros valores nos algoritmos (e programas).

- Os tipos de dados **simples** habituais são
  - **Booleanos** (Verdade/Falso ou 1/0)
  - **Numéricos** (Inteiros, Reais e Imaginários)
  - **Não numéricos** (caracteres)

- Normalmente o contexto torna claro os tipos de dados pretendidos para as variáveis, mas as linguagens de programação típicas (não o Octave) requerem a declaração dos tipos de dados das variáveis.

Estruturas de Dados

- Os dados simples podem ser agrupados em estruturas de dados mais complexas.
- As estruturas mais vulgares correspondem a matrizes de dimensão arbitrária.
- Um caso importante são as matrizes unidimensionais (vectores).
- Como o nome indica (MATrix LABoratory), o sistema Octave / MATLAB tem um suporte muito completo dos tipos de dados matriz, permitindo a sua definição e as operações habituais.

Estruturas de Dados

- Exemplo 1:

```
A = [1 2 3 ; 4 5 6 ; 7 8 9]
```

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

- Exemplo 2:

```
B = [3 3 3 ; 2 2 2 ; 1 1 0]
```

$$B = \begin{pmatrix} 3 & 3 & 3 \\ 2 & 2 & 2 \\ 1 & 1 & 0 \end{pmatrix}$$

- Exemplo 3:

```
C = A + B
```

$$C = \begin{pmatrix} 4 & 5 & 6 \\ 6 & 7 & 8 \\ 8 & 9 & 9 \end{pmatrix}$$

- Exemplo 4:

```
D = [1 2 3] * A
```

$$D = \begin{pmatrix} 30 & 36 & 42 \end{pmatrix}$$

Exemplos de Algoritmos

1. Equação do 2º grau
2. Máximo Divisor Comum (Euclides)
3. Raiz Quadrada
4. Logaritmo de 2

Algoritmo 1 – Raízes da equação do 2º grau

```
% ax^2 + bx + c = 0

entra a ; entra b ; entra c ;

disc ← b^2 - 4 * a * c;
se disc < 0 então
   sai 'não há raízes reais'
senão
   d ← sqrt(disc);
   se d = 0 então
      sai ' 2 raízes iguais'
      r ← -b/(2*a); sai R
   senão
      r1 ← (-b + d)/(2*a); sai r1;
      r2 ← (-b - d)/(2*a); sai r2;
   fim se;
fim se;
```

Algoritmo 1 – Em Octave

```
a = input('qual o valor de a? :');
b = input('qual o valor de b? :');
c = input('qual o valor de c? :');
disc = b^2 - 4*a*c;
if disc < 0  disp('a equacao nao tem raizes reais')
else
   d = sqrt(disc);
   if d == 0
      r = -b/(2*a);
      disp('raiz real dupla'), disp(r);
   else
      r1 = (-b+d)/(2*a); r2 = (-b-d)/(2*a);
      disp('2 raizes reais distintas');
      disp(r1); disp(r2);
   endif;
endif
```

Algoritmo 2 – Maior Divisor Comum

A	B
56	182

X		Y		C
182	-	56	=	126
126	-	56	=	70
70	-	56	=	14
56	-	14	=	42
42	-	14	=	28
28	-	14	=	14
14	-	14	=	0

```
C ← X - Y
novo X ← max(Y,C)
novo Y ← min(Y,C)
```

Algoritmo 2 – Maior Divisor Comum"

```
entra a ;
entra b ;
x ← max(a,b); y ← min(a,b);
c ← x - y
enquanto c > 0 fazer
    x ← max(y,c); y ← min(y,c);
    d ← x - y;
fim enquanto
sai x
```

A	B
56	182

X		Y		C
182	-	56	=	126
126	-	56	=	70
70	-	56	=	14
56	-	14	=	42
42	-	14	=	28
28	-	14	=	14
14	-	14	=	0

```
a = input('qual o valor de a? :');
b = input('qual o valor de b? :');
x = max(a,b); y = min(a,b);
d = x - y;
while d != 0
    x = max(y,d);    y = min(y,d);    d = x - y;
endwhile
disp('máximo divisor comum: '); disp(x);
```

Algoritmo 3 – Cálculo da Raiz Quadrada

$z = sqrt(x) \Leftrightarrow z * z = x \Leftrightarrow a_i * b_i = x$

```
entra x ;
b ← x ;
a ← x / b ;
enquanto b-a >= p fazer
   b ← (a + b)/2;
   a ← x / b;
fim enquanto;
sai u ;                  % u ← sqrt(X)
```

i	a	b
1	1,00000	36,00000
2	1,94595	18,50000
3	3,52148	10,22297
4	5,23848	6,87223
5	5,94515	6,05535
6	5,99975	6,00025
7	6,00000	6,00000

```
x = input('qual o numero : ');
b = x ; a = x/b;
while abs(b-a) >= 0.0001
   b = (a + b)/2
   a = x / b;
endwhile;
disp('raiz quadrada do numero: ');
disp(b);
```

Algoritmo 4 – Cálculo de ln(2)

```
ln(2)  = 1 - 1/2 + 1/3 - 1/4 + ...
```

```
ln ← 0 ;
i ← 1;
s ← 1;
enquanto i < 1000 fazer
    ln ← ln + s * 1/i;
    i ← i + 1;
    s ← -s;
fim enquanto;
sai ln ;                % ln(2)
```

```
ln = 0 ;
i  = 1;
s  = 1;
while i < 1000
    ln = ln + s * 1/i;
    i = i + 1;
    s = -s;
endwhile;
disp('ln(2) = ')
disp(ln)
```