ESDI / UERJ

COR e Estruturas Bidimensionais

Prof. Mauro Pinheiro

para a prática de
design de interação
é preciso entender de

programação e prototipação

a programação é a maneira de dizer ao computador o que queremos que ele faça

computador precisa de instruções

algoritmo é a lógica dessas instruções

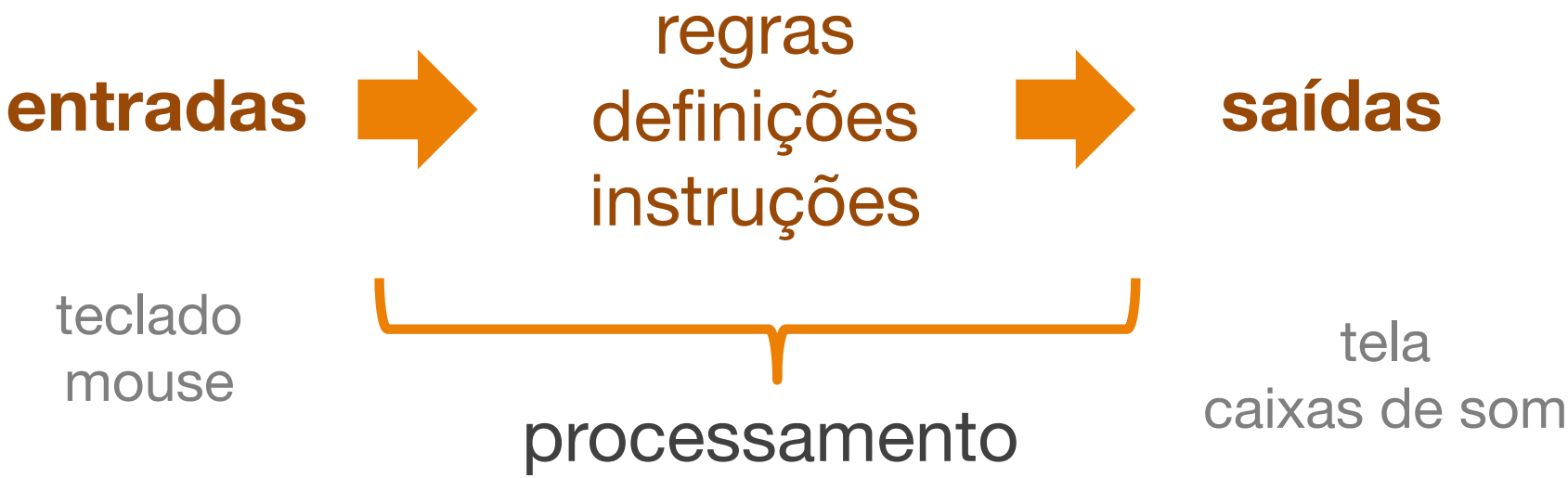
algoritmo

conjunto finito de regras que fornece uma seqüência de operações para resolver um problema específico

algoritmo

conjunto finito de regras que fornece uma seqüência de operações para resolver um problema específico

algoritmo + linguagem de programação  programa

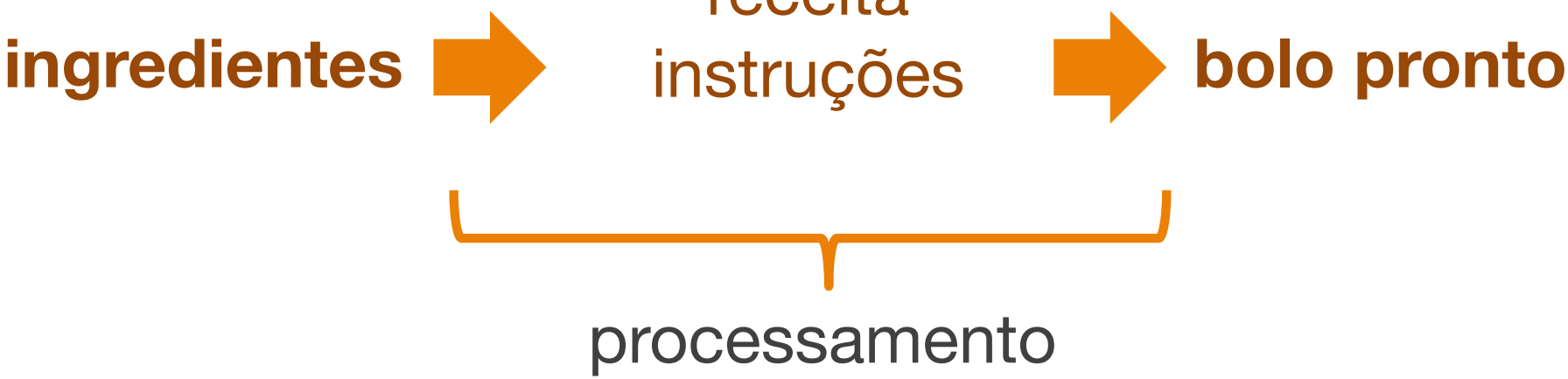

entradas
regras
definições
instruções
saídas
teclado
mouse
processamento
tela
caixas de som

como instruir o computador?

instruções em seqüência lógica

trocar a lâmpada

- pegue uma escada;
- coloque-a embaixo da lâmpada;
- busque uma lâmpada nova;
- suba na escada com a lâmpada nova;
- retire a lâmpada velha;
- coloque a lâmpada nova;
- desça da escada.


ingredientes
receita
instruções
bolo pronto
processamento

instruções para fazer bolo salgado

- misture os ingredientes
- unte o tabuleiro com manteiga
- despeje a mistura no tabuleiro
- **se**
  (há queijo parmesão)
  então
  espalhe sobre a mistura
  - deixe descansar por 10min
  - leve o tabuleiro ao forno
- **enquanto** (não dourar)
  deixe o tabuleiro no forno
- retire do forno
- deixe esfriar

- deixe esfriar
- misture os ingredientes
- unte o tabuleiro com manteiga
- deixe descansar por 10min
- retire do forno
- leve o tabuleiro ao forno
- **se**
  (há queijo parmesão)
  então
  espalhe sobre a mistura
- despeje a mistura no tabuleiro
- **enquanto** (não dourar)
  deixe o tabuleiro no forno

algoritmo é independente da linguagem de programação

cada linguagem tem sua sintaxe própria

mesmo algoritmo
sintaxes distintas

o resultado é o mesmo

- misture os ingredientes
- unte o tabuleiro com manteiga
- despeje a mistura no tabuleiro
- **se**
  (há queijo parmesão)
  então
  espalhe sobre a mistura
- deixe descansar por 10min
- leve o tabuleiro ao forno
- **enquanto** (não dourar)
  deixe o tabuleiro no forno
- retire do forno
- deixe esfriar

- mix the ingredients
- grease the pan with butter
- Pour the mixture into the tray
- **if**
  (there is no parmesan)
  then
  sprinkle over the mixture
- let it rest for 10min
- put the tray in the oven
- while (is not brown)
  leave the tray in the oven
- remove from the oven
- let it cool down

funções

funções

em um programa, funções são instruções específicas que o programa utiliza para processar os dados

cada linguagem de programação tem um conjunto de funções próprias, nativas

no exemplo da receita de bolo, as funções seriam

misturar, untar, espalhar, deixar descansar etc.

parâmetros

parâmetros

valores que complementam uma função

no exemplo da receita de bolo, o parâmetro que complementa a função **deixar descansar** é o tempo **10min**

exemplos na linguagem Processing:

```
print(“texto a ser impresso”);
line(1,0,4,5);
```

condições

estrutura de condição

se (há queijo parmesão)
então (espalhe sobre a mistura)

a condição a ser testada é se *há queijo parmesão*
essa condição precisa ser satisfeita (ser verdadeira) para a execução da ação *espalhe sobre a mistura*

a ação *espalhe sobre a mistura* está subordinada ao fato de existir queijo parmesão; se não existir queijo, nada é feito.

estrutura de repetição

enquanto (não dourar)
deixe o tabuleiro no forno

condição de repetição que determina a execução contínua de uma ação

instruções para cadastro de aluno

- verifique o preenchimento de um formulário
- **se**
  (preenchimento correto)
  então
  arquivar documento;
  fornecer o protocolo;
  senão
  adquirir outro formulário
  fazer novo preenchimento
- despeça-se educadamente do aluno.

condição de teste (se preenchimento correto) que determina duas ações, uma para o caso em que o teste fornecer resultado positivo (então) e outra para o caso em que o teste fornecer resultado negativo (senão).

funções criadas pelo programador

embora em um programa existam funções específicas próprias da linguagem de programação, é possível criar novas funções, isto é, sequências de instruções.

cada vez que “chamarmos” a função, o programa deverá executar a sequência de instruções que aquela função contém

normalmente fazemos isso quando precisamos repetir um conjunto de instruções diversas vezes no programa – é mais prático “chamar” a função diversas vezes do que reescrever o mesmo conjunto de instruções diversas vezes

- **cálculo das áreas de uma mesa e de uma toalha.**
  - **cálculo da área de uma mesa.**
    - medir a largura da mesa e anotar o resultado.
    - medir o comprimento da mesa e anotar o resultado.
    - multiplicar o comprimento pela largura e anotar o resultado.
    - o valor da área da mesa é o resultado anotado no passo anterior.
  - **fim do cálculo da área da mesa.**
  - **cálculo da área da toalha.**
    - medir a largura da toalha e anotar o resultado.
    - medir o comprimento da mesa e anotar o resultado.
    - multiplicar o comprimento pela largura e anotar o resultado.
    - o valor da área da toalha é o resultado anotado passo anterior.
  - **fim do cálculo da área da toalha.**
- **fim do cálculo das áreas da mesa e da toalha.**

- faça o **cálculo da área de um retângulo (toalha)**
- faça o **cálculo da área de um retângulo (mesa)**

função cálculo da área de um retângulo

- início
    - medir a largura do objeto e anotar o resultado.
    - medir o comprimento do objeto e anotar o resultado
    - multiplicar o comprimento pela largura e anotar o resultado.
    - o valor da área e o resultado anotado no passo anterior.
- fim

variável

variável

em um programa, é o nome dado a um espaço reservado na memória para armazenar informações que podem mudar ao longo da execução do programa

exemplo: nota1, nota2, media

variável

as variáveis podem ser de diferentes naturezas, em função dos dados que devem armazenar

- variáveis numéricas: armazenam somente números (inteiros ou reais)
- variáveis alfanuméricas: armazenam seqüências de caracteres (números ou letras)

é fundamental definir bem qual a natureza das variáveis que se vai utilizar. algumas variáveis, por sua própria natureza, ocupam mais espaço na memória;

declarar uma variável

primeiro é preciso informar ao computador para "reservar" um espaço na memória, onde ele vai guardar dados posteriormente.

```
int A
float myHeight
char myKey
String names
```

uma variável só é declarada uma vez ao longo do programa

inicializar uma variável

uma vez reservado o espaço da variável, é preciso "preencher" esse espaço com dados. Esses dados podem mudar ao longo da execução do programa

```
int A;
float myHeight;
char myKey;
String names;
A = 1;
myHeight = 15.35;
myKey = 'm';
names = "mauro";
```

declarar e inicializar uma variável

é possível declarar e inicializar a variável ao mesmo tempo

```
int A = 1;
float myHeight = 15.35;
char myKey = ‘m’;
String names = “mauro”;
```

expressões matemáticas

expressões matemáticas

em geral, os programas executam as operações matemáticas ☺

operadores: + (soma) - (subtração) *(multiplicação) / (divisão) % (módulo, resto de divisão inteira)

exemplos:

```
A+B-C

A/B

3.14*(A+B)

((A*2)+(B*3)+(C*1))/6
```

existe uma ordem de prioridade entre os operadores. Multiplicação e divisão têm prioridade sobre soma e subtração. Pode-se usar parênteses para alterar essa hierarquia. O que estiver dentro do parênteses passa a ter prioridade.

se o resultado de uma operação for armazenado em uma variável, esse resultado pode variar de acordo com o tipo de variável definido: exemplo 1/5=0 se a variável só armazenar números inteiros. Se a variável armazenar números reais, 1/5=0.2

expressões lógicas

expressões lógicas

expressões lógicas são aquelas cujo resultado somente permite os valores verdadeiro ou falso.

exemplos:

se A vale 1;
B vale 2;
C vale 3

A é igual a B?	falso
(A+B) é igual a C?	verdadeiro
A é maior do que B?	falso
A é menor do que B?	verdadeiro
A é diferente de B?	verdadeiro
((A+B) é igual a C) e (A é maior do que B)?	falso

operadores lógicos

usados nas expressões lógicas

operadores de comparação

== (igual a)

!= (diferente de)

< (menor que)

> (maior que)

<= (menor que ou igual a)

>= (maior que ou igual a)

operadores booleanos

&& (e)

ll (ou)

! (não)

expressões lógicas

A é igual a B?	A == B
(A+B) é igual a C?	(A+B) == C
A é maior do que B?	A > B
A é menor do que B?	A < B
A é diferente de B?	A != B
((A+B) é igual a C) e (A é maior do que B)?	((A+B)==C) && (A > B)

atenção!!

A == 2

computador vai verificar qual o valor que a variável ‘A’ tem naquele momento da execução do programa, e vai gerar um resultado VERDADEIRO ou FALSO

é como se disséssemos ao computador para **verificar se A é igual a 2**

A = 2

a partir desse comando, naquele momento de execução do programa, o computador vai atribuir o valor ‘2’ à variável ‘A’.

é como se disséssemos ao computador que a partir de agora **A recebe o valor 2**

loop

loop

conjunto de instruções que o computador repete até que uma determinada condição seja atendida

```
int xPosition = 0;
int maxWidth = 400;

while (xPosition <= maxWidth) {
    point (xPosition, 10);
    xPosition = xPosition+1;
}
```

loop

```
int xPosition = 0;
int maxWidth = 400;

while (xPosition <= maxWidth) {
    point (xPosition, 10);
    point (xPosition, 20);
    xPosition = xPosition+1;
}
```

outro tipo de loop

```
int maxWidth = 400;

for (int xPosition = 0; xPosition <= maxWidth; xPosition++){
    point (xPosition, 10);
    point (xPosition, 20);
}
```

loop

```
int maxWidth = 400;

for (int xPosition = 0; xPosition <= maxWidth; xPosition++) {
    point (xPosition, 10);
    point (xPosition, 20);
}
```

declaro e inicializo a variável **xPosition**
que recebe o valor inicial **0** (zero)

loop

```
int maxWidth = 400;

for (int xPosition = 0; xPosition <= maxWidth; xPosition++) {
    point (xPosition, 10);
    point (xPosition, 20);
}
```

essa é a condição que será verificada a cada execução do loop. enquanto for ‘true’, o loop será repetido.

loop

```
int maxWidth = 400;

for (int xPosition = 0; xPosition <= maxWidth; xPosition++) {
    point (xPosition, 10);
    point (xPosition, 20);
}
```

essa é a instrução do que o loop deverá fazer ao final de cada ciclo

loop

```
int maxWidth = 400;

for (int xPosition = 0; xPosition <= maxWidth; xPosition++) {
    point (xPosition, 10);
    point (xPosition, 20);
}
```

os comandos que serão executados a cada ciclo do loop ficam entre chaves { }

comentários

é uma boa prática de programação colocar comentários ao longo do código. os comentários não são executados pelo programa (não são instruções, são comentários!)

isso é extremamente útil para entender a lógica do programa, e facilita o trabalho quando precisamos revisitar um programa que escrevemos há muito tempo, ou quando lemos um programa feito por outra pessoa

```
/* comentário em mais de uma linha
 * pode conter um texto extenso
 */
int xPosition = 0;  //comentário breve
int maxWidth = 400;

//outro comentário
line(1,0, xPosition, maxWidth);
```

```
/*
 * This sketch uses the text drawn to an offscreen PGraphics to determine
 * if coordinate are inside the drawn text. If so, a growing Circle is added.
 * It grows until either the maxRadius is reached or there will be overlap
 * with another Circle.
 *
 * USAGE:
 * - press 's' to export a single frame as a PDF page
 */

import processing.pdf.*;

ArrayList <Circle> circles = new ArrayList <Circle> ();
color BACKGROUND_COLOR = color(255);
color PGRAPHICS_COLOR = color(0);
PGraphics pg;

boolean saveOneFrame = false; // variable used to save a single frame as a PDF page

void setup() {
  size(1280, 720, P2D);
  smooth(16); // higher smooth setting = higher quality rendering
  // set colorMode for the sketch to Hue-Saturation-Brightness (HSB)
  colorMode(HSB, 360, 100, 100);
  // create the offscreen PGraphics with the text
  pg = createGraphics(width, height, JAVA2D);
  pg.beginDraw();
  pg.textSize(500);
```