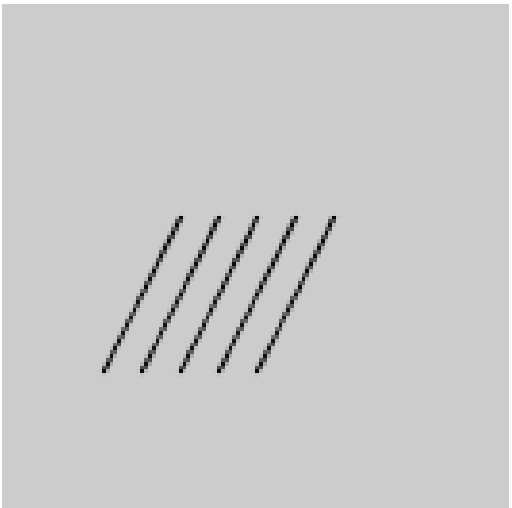
Primeiros passos em Computação 1

Luiz Ernesto Merkle

Universidade Tecnológica Federal do Paraná
Programa de Pós-Graduação em Tecnologia
Departamento Acadêmico de Informática
Grupo de Pesquisa em Ciências Humanas, Tecnologia e Sociedade
Estúdio Xuê

17 de junho de 2013

0.01 - Uma linha depois de outra . . .

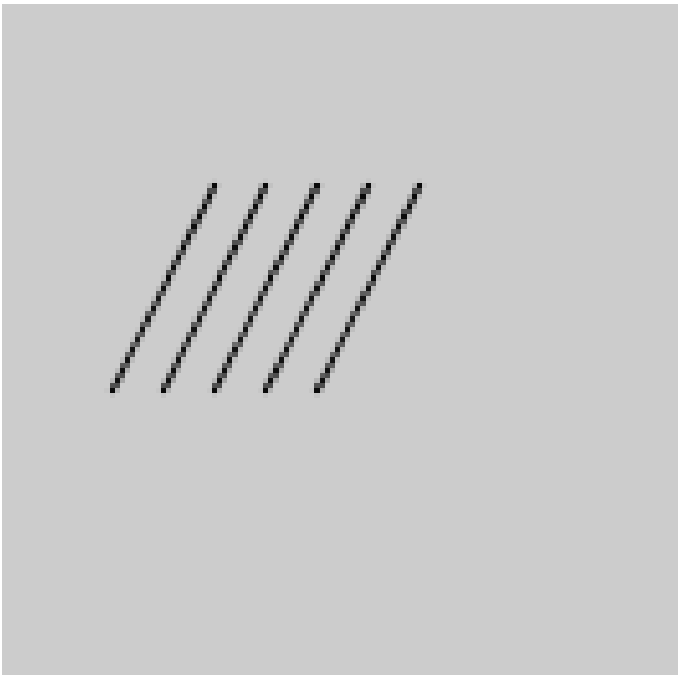


```
/*-Exemplo 01 */
line(10, 80, 30, 40); // linha  esquerda
line(20, 80, 40, 40);
line(30, 80, 50, 40); // linha ao centro
line(40, 80, 60, 40);
line(50, 80, 70, 40); // linha  direita
```

0.02 Para controlar o traço . . .

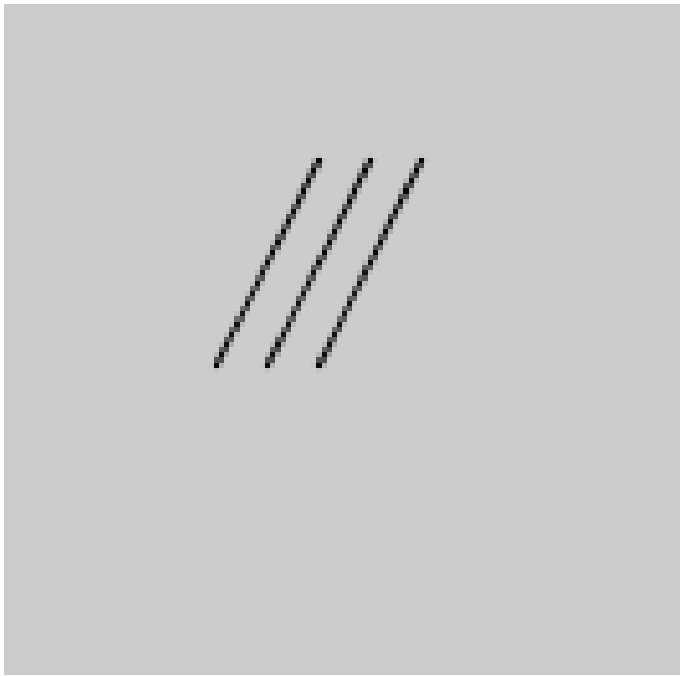
```
background(0); // Sets the black background
stroke(255); // Sets line value to white
strokeWeight(5); // Sets line width to 5 pixels
smooth(); // Makes the lines draw with smooth edges
line(10, 80, 30, 40); // Left line
line(20, 80, 40, 40);
line(30, 80, 50, 40); // Middle line
line(40, 80, 60, 40);
line(50, 80, 70, 40); // Right line
```

Exemplo passo a passo 0-03



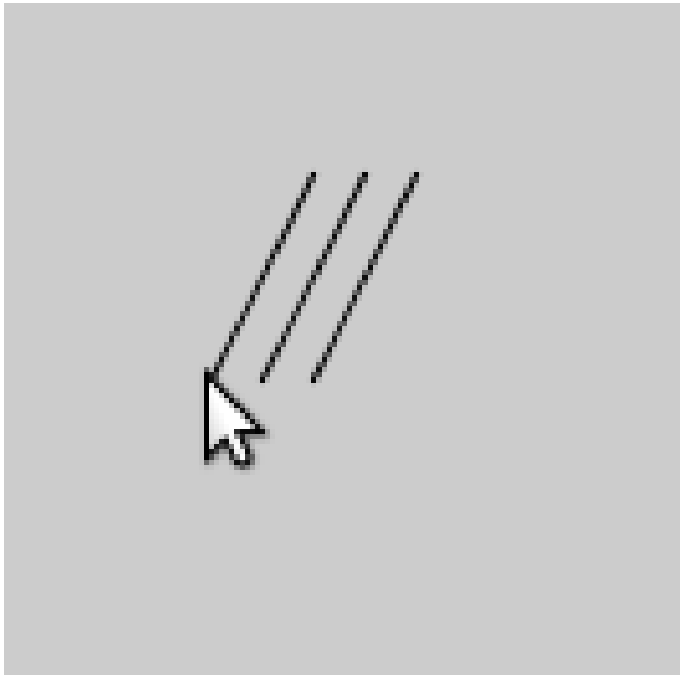
```
int x = 5; // Sets the horizontal position of the lines
int y = 60; // Sets the vertical position of the lines
line(x, y, x+20, y−40); // Draws line from [5,60] to [25,
line(x+10, y, x+30, y−40); // Draws line from [15,60] to
line(x+20, y, x+40, y−40); // Draws line from [25,60] to
line(x+30, y, x+50, y−40); // Draws line from [35,60] to
line(x+40, y, x+60, y−40); // Draws line from [45,60] to
```

0.03 Variáveis agilizam o controle . . .



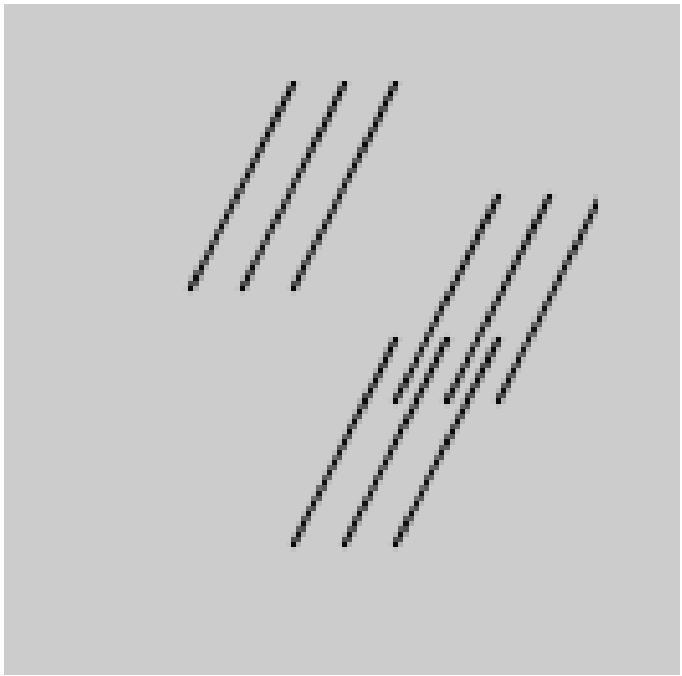
```
int x = 0; // Sets the horizontal position of the lines
int y = 55; // Sets the vertical position of the lines

void setup() {
  size(100, 100); // Sets the window size to 100 x 100 p
}

void draw() {
  background(204);
  line(x, y, x+20, y−40); // Left line
  line(x+10, y, x+30, y−40); // Middle line
  line(x+20, y, x+40, y−40); // Right line
  x = x + 1; // Add 1 to x
  if (x > 100) { // If x is greater than 100
    x = −40; // Assign −40 to x
  }
}
```

0.04 E uma pitada de interação . . .



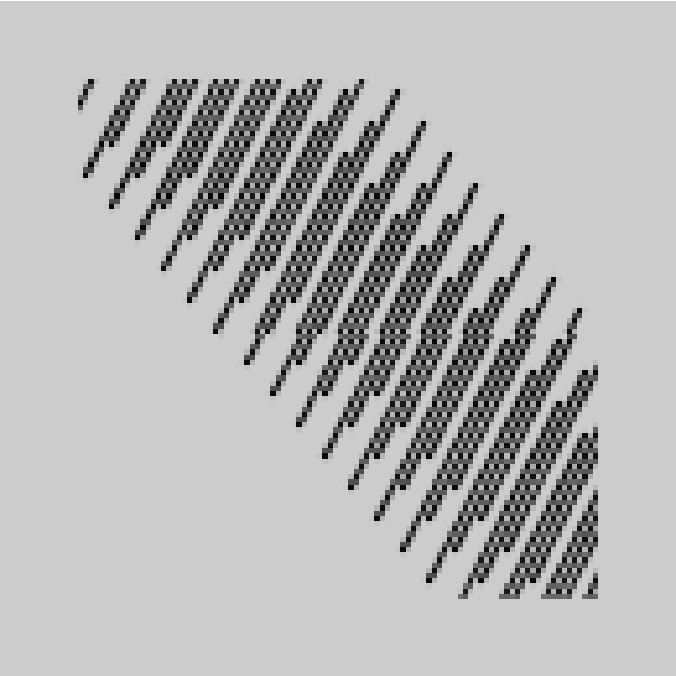
```
void setup() {
  size(100, 100);
}

void draw() {
  background(204);
  // Assigns the horizontal value of the cursor to x
  float x = mouseX;
  // Assigns the vertical value of the cursor to y
  float y = mouseY;
  line(x, y, x+20, y−40);
  line(x+10, y, x+30, y−40);
  line(x+20, y, x+40, y−40);
}
```

Enxuge tudo com funções . . .



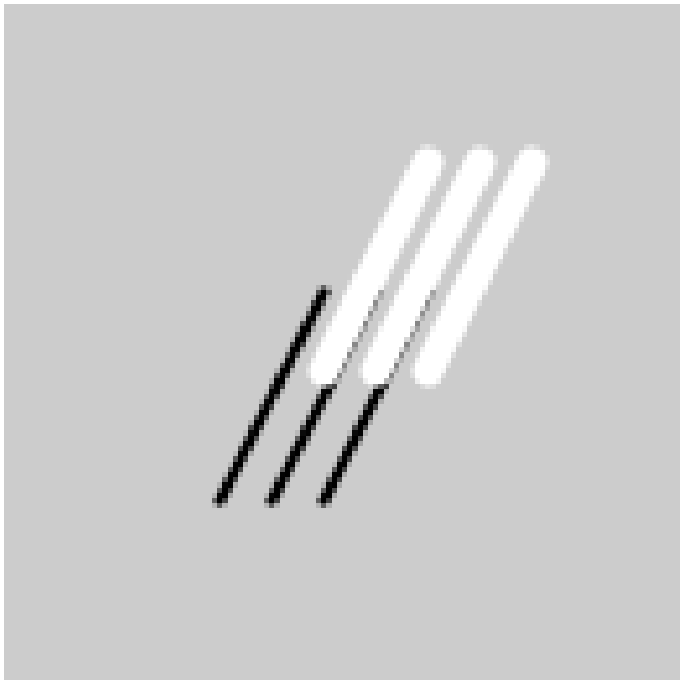
```
void setup() {
  size(100, 100);
  noLoop();
}

void draw() {
  diagonals(40, 90);
  diagonals(60, 62);
  diagonals(20, 40);
}

void diagonals(int x, int y) {
  line(x, y, x+20, y−40);
  line(x+10, y, x+30, y−40);
  line(x+20, y, x+40, y−40);
}
```

0.07 Tempere com conjuntos de dados . . .



```
int num = 20;
int[] dx = new int[num]; // Declare and create an array
int[] dy = new int[num]; // Declare and create an array
void setup() {
  size(100, 100);
  for (int i=0; i<num; i++) {
    dx[i] = i*5;
    dy[i] = 12 + (i*6);
  }
}
void draw() {
  background(204);
  for (int i=0; i<num; i++) {
    dx[i] = dx[i] + 1;
    if (dx[i] > 100) {
      dx[i] = -100;
    }
    diagonals(dx[i], dy[i]);
  }
}
void diagonals(int x, int y) {
  line(x, y, x+20, y-40);
  line(x+10, y, x+30, y-40);
  line(x+20, y, x+40, y-40);
```

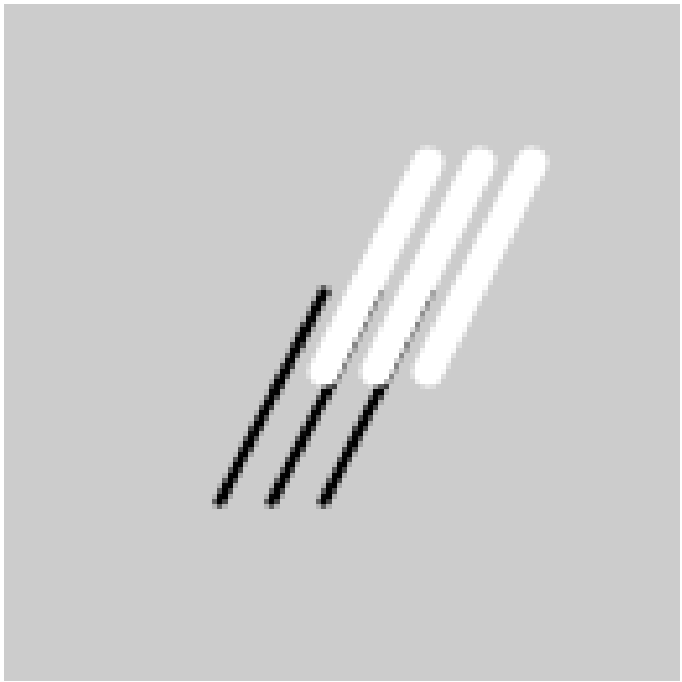
Apimente com classes de objetos . . .



```
Diagonals da, db;

void setup() {
  size(100, 100);
  smooth();
  // Inputs: x, y, speed, thick, gray
  da = new Diagonals(0, 80, 1, 2, 0);
  db = new Diagonals(0, 55, 2, 6, 255);
}

void draw() {
  background(204);
  da.update();
  db.update();
}
```

continua . . .

Apimente com classes de objetos . . .



```
class Diagonals {
  int x, y, speed, thick, gray;

  Diagonals(int xpos, int ypos, int s, int t, int g) {
    x = xpos;
    y = ypos;
    speed = s;
    thick = t;
    gray = g;
  }
  void update() {
    strokeWeight(thick);
    stroke(gray);
    line(x, y, x+20, y−40);
    line(x+10, y, x+30, y−40);
    line(x+20, y, x+40, y−40);
    x = x + speed;
    if (x > 100) {
      x = −100;
    }
  }
}
```

Para abrir uma tela . . .

```
// Two forward slashes are used to denote a comment.
// All text on the same line is a part of the comment.
// There must be no spaces between the slashes. For example,
// the code "/ /" is not a comment and will cause an error
// If you want to have a comment that is many
// lines long, you may prefer to use the syntax for a
// multiline comment

/*
  A forward slash followed by an asterisk allows the
  comment to continue until the opposite
*/

// All letters and symbols that are not comments are translated
// by the compiler. Because the following lines are not comments,
// they are run and draw a display window of 200 x 200 pixels
size(200, 200);
background(102);
```

Variáveis

```
size(200, 200); // Runs the size() function

/** declara uma varivel do tipo inteira */
int x; // Declares a new variable x

/** atribui o valor 102  varivel x */
x = 102; // Assigns the value 102 to the variable x

background(x); // Runs the background() function

/** calcula 2*x e atribui a cor de prenchimento das formas */
fill(2*x);
rect(10,10, 180,180);
```

Procure arrumar seu código de modo legível . . .

```
/** Espaos no fazem diferena para o compilador */
  size
// jose foi a cidade comprar mandioca
( 200,
// comentario fora de lugar


  200)

  ;
background (
102)
            ;
/*- Mas fazem uma enorme diferena para quem o le */
```

Para imprimir mensagens . . .

```
// To print text to the screen, place the desired output in quotes
println("Processing..."); // Prints "Processing..." to the console

// To print the value of a variable, rather than its name, don't put
// the name of the variable in quotes.
int x = 20;
println(x); // Prints "20" to the console

// While println() moves to the next line after the text
// is output, print() does not.
print("10");
println("20"); // Prints "1020" to the console
println("30"); // Prints "30" to the console

// The "+" operator can be used for combining multiple text
// elements into one line.
int x2 = 20;
int y2 = 80;
println(x2 + " : " + y2); // Prints "20 : 80" to the message window
```

Variáveis de tipos básicos

```
int x; // Declare the variable x of type int
float y; // Declare the variable y of type float
double yy; // Decalra uma variavel do tipo double
boolean b; // Declare the variable b of type boolean

x = 50; // Assign the value 50 to x
y = 12.6; // Assign the value 12.6 to f
b = true;

//x = 12.6 // Error - No  possivel atribuir um float a um inteiro
x = int(12.6); // Converte um float em um inteiro
```

Variáveis de tipos básicos

```
int inteiro; // Declare the variable x of type int
long inteiro_maior;

float real; // Declare the variable y of type float
double real_preciso; // Decalra uma variavel do tipo double

boolean b; // Declare the variable b of type boolean
byte B;

char letra;
color vermelho;

inteiro = 50; // Assign the value 50 to x
real = 12.6; // Assign the value 12.6 to f
b = true;
letra = 'c';

String S = "Maria";

//inteiro = 12.6 // Error - No  possivel atribuir um float a um inteiro
inteiro = int(12.6); // Converte um float em um inteiro
```

Variáveis de tipos básicos

boolean Pode assumir o valor true ou o valor false

char Caractere em notação Unicode de 16 bits. Serve para a armazenagem de dados alfanuméricos. Também pode ser usado como um dado inteiro com valores na faixa entre 0 e 65535.

byte Inteiro de 8 bits em notação de complemento de dois. Pode assumir valores entre $-2^7 = -128$ e $2^7 - 1 = 127$.

short Inteiro de 16 bits em notação de complemento de dois. Os valores possíveis cobrem a faixa de $-2^{-15} = -32.768$ a $2^{15} - 1 = 32.767$

int Inteiro de 32 bits em notação de complemento de dois. Pode assumir valores entre $-2^{31} = 2.147.483.648$ e $231 - 1$=2.147.483.647.

long Inteiro de 64 bits em notação de complemento de dois. Pode assumir valores entre -2^{63} e $2^{63} - 1$.

float Representa números em notação de ponto flutuante normalizada em precisão simples de 32 bits em conformidade com a norma IEEE 754-1985. O menor valor positivo represntável por esse tipo é $1.40239846e - 46$ e o maior é $3.40282347e + 38$

double Representa números em notação de ponto flutuante normalizada em precisão dupla de 64 bits em conformidade com a norma IEEE 754-1985. O menor valor positivo representável é $4.9406564584124654e - 324$ e o maior é $1.7976931348623157e + 308$

Sadao Massago e Waldeck Schützer (Sem data) Tipos de dados. In: Programação Java. Disponível em http://www.dm.ufscar.br/ waldeck/curso/java/part22.html

Variáveis de tipos básicos

```
double numeroDecimal = 5.0;
numeroDecimal = 5d;
numeroDecimal = 0.5;
numeroDecimal = 10f;
numeroDecimal = 3.14159e0;
numeroDecimal = 2.718281828459045D;
numeroDecimal = 1.0e-6D;
\\ http://en.wikibooks.org/wiki/Java_Programming/Literals
```

Operadores

+ (adição), − (subtração), ∗ multiplicação, / (divisão), % (modulo ou resto)
() (parenteses)
++ (incremento), −− (decremento), + = (adiciona e atribui), − = (subtrai e atribui)
∗ = (multiplica e atribui), / = (divide e atribuin), − (negação)
ceil(), floor(), round(), min(), max()

```
int a = 8;
int b = 10;
line(a, 0, a, height);
line(b, 0, b, height);
strokeWeight(4);
line(a*b, 0, a*b, height);
```

Operadores Lógicos e Relacionais

&& (E lógico), || (OU lógico), ! (Negação), < (menor), <= (menor igual)
> (maior), >= (maior igual), == (igual)

```
for(int i=5; i<=95; i+=5) {
  if((i > 35) && (i < 60)) {
    stroke(0); //Atribui a varivel color a cor preta
  } else {
    stroke(255); //Atribui a varivel color a cor branca
  }
  line(30, i, 80, i);
}
```

Revisão após recesso

- Instruções;
- Blocos;
- Variáveis;
- Operadores aritméticos;
- Operadores realacionais;
- Condicionais;
- Laços e repetições;
- Laços e repetições aninhadas.

Instruções

Toda instrução termina por um *ponto e vírgula*, inclusive a vazia.

```
// isto  um esboo com seis instrues vazias
//  um esboo que faz nada seis vezes
;
;
;
;
;
;
```

Instruções

Toda instrução termina por um *ponto e vírgula*, inclusive a vazia.

```
// isto  um esboo com seis instrues no vazias
int a; // declara uma varivel a
a=50; // atribui o valor 5  varivel a
print(a); // imprime o valor de a
point(a,a); // chama a funo que desenha um ponto em a,a
```

Instruções

Toda instrução termina por um *ponto e vírgula*, inclusive a vazia. CUIDADO COM O PONTO E VIRGULA DEPOIS DO LAÇO FOR() e do condicional if()

```
// isto  um comando que repete 10 vezes uma instruo vazia:
for (int = 0; i<10; i++);
  xxx ; // instruo qualquer
```

Como espaços não fazem diferença, equivale a:

```
// isto  um comando que repete 10 vezes uma instruo vazia:
for (int = 0; i<10; i++)
  ;
  xxx ; // instruo qualquer
```

Instruções

Toda instrução termina por um *ponto e vírgula*, inclusive a vazia.

CUIDADO COM O PONTO E VIRGULA DEPOIS DOs CONDICIONAIS!

```
// Se verdadeiro, faz−se nada, pois o ponto e virgula  uma instruo vazia.
if (true);
  xxx ; // instruo qualquer
```

Equivale à:

```
// Se verdadeiro, faz−se nada, pois o ponto e virgula  uma instruo vazia.
if (true)
      ;
  xxx ; // instruo qualquer
```

O certo seria, sem o ponto e vírgula:

```
// Se verdadeiro, faz−se nada, pois o ponto e virgula  uma instruo vazia.
if (true)
  xxx ; // instruo qualquer
```

Blocos

Blocos são conjuntos de instruções cercados por chaves

```
// exemplo
{
println("mouseX␣=" + mouseX);
println("mouseY␣=" + mouseY);
}
```

Blocos

Blocos são usados para separar conjuntos de instruções em condicionais, laços, funções, classes.

```
if (true) {
   println("verdadeiro");
   println("At␣daqui␣a␣pouco");
   }
else
   {
   println("falso");
   println("Hasta␣la␣vista,␣Baby");
   }
```

Blocos

Blocos são usados para separar conjuntos de instruções em condicionais, laços, funções, classes, ou ao longo de um esboço, para separar variáveis

```
int a = 10; // esta varivel pode ser acessada de qualquer lugar

a += 10;
{
  int a = 10; // esta variavel a s pode ser acessada de dentro destas chaves
  a *= 10;
}
a *= 10;
// qual o valor de a, 100 ou 1000?
println(a);
```

Variáveis

Variáveis armazenam informação de vários tipos na memória do computador. Podem representar números inteiros(int), de ponto flutuante(float, double), caracteres(char), sequências de caracteres(String), cores (color), valores de ferdade (true, false).

```
// Se declaradas fora de um bloco, podem ser acessadas de dentro dele,
// desde que no haja outra varivel de mesmo nome dentro dele.
// Um contador global pode ser usado para isto.
int xy=0;

void seput(){ // inicializao
 strokeWeight(5)
 };
void draw(){ //repetio
 point(xy*10,xy*xy/100);
 if(xy==10)
    xy=0
    x++; // como a varivel xy foi declarada fora do bloco, seu valor  atualizado.
        // Se decalra dentro do bloco, ela deixa de existir
 }
```

Operadores aritiméticos

Operadores aritiméticos são utilizados para calcular o valor de expressões aritméticas;

```
rect(width/4, height/4, width/2, height/2);
```

Antes de desenhar o retângulo, o programa substitui as variáveis por seus valores

```
rect(100/4, 100/4, 100/2, 100/2);
```

Calcula o valor de cada expressão:

```
rect(25,25,100,100);
```

E chama a função rect(), que desenha um retângulo na tela de saída.

Funções

A linguagem processing tem associada a ela uma biblioteca extensa de funções, para os mais diversos fins.
Se não encontrar, procure nas bibliotecas associadas, que pode have alguma já desenvolvida para o que você precisa.
Se não encontrar, muitas são de código aberto ou livre, e podem ser modificadas, desde que atribuída a autoria de quem a desenvolveu.

Funções

Funções em processing podem receber ferenciados, desde que programados para tal:

```
// color cor;
cor = color(127);
cor = color(125,255);
cor = color(127,127,127);
cor = color(127,127,127,255);
```

Que neste caso, resultam na mesma cor e opacidade.

Condicionais

Um comando if(){}, ou if(){}else{} controla a execução de um programa, podendo-se escolher o que vai se executar quando uma condição verdadeira.

```
void setup(){}
void draw(){
  if(mouseX<height-1)
    background(255,0,0);
  else
    background (0,255,0);
}
```

Condicionais Aninhados

Um condicional pode conter outro condicional, sucessivamente.

```
void setup(){}
void draw(){
  if(mouseX<height/2)
    if(mouseY<height/2 0)
        background(0);
    else
        background(255,0,0);
  else
   if(mouseY<height/2 0)
        background(0,255,0);
    else
        background(0,0,255);
}
```

Laços e Repetições

O computador é uma máquina excelente para repetir instruções.
Os comandos for(){} e while(){} podem ser usados para tal.

```
for(int a=0; a<100; a+=10)
  {
  fill(a,127);
  rect(a,a,10,10);
  }
```

Laços e Repetições

O computador é uma máquina excelente para repetir instruções.
Os comandos for(){} e while(){} podem ser usados para tal.
Como com valores pré-determinados.

```
for(int a=0; a<100; a+=10)
  {
  fill(a,127);
  rect(a,a,10,10);
  }
```

Laços e Repetições

O computador é uma máquina excelente para repetir instruções.
Os comandos for(){} e while(){} podem ser usados para tal.
Como com valores pseudo-randômicos.

```
for(int a=0; a<10000; a+=10)
  {
  fill(random(255),random(255),random(255));
  rect(random(100), random(100),random(10,20),random(5,30));
  }
```

Laços e Repetições Aninhadas

Quando se precisa repetir uma repetição, usam-se laços aninhados
Como com valores pseudo-randômicos.

```
for(int y=0; y<100; y+=10)
  {
  for(int x=0; x<100; x+=15)
    {
    fill(random(255),random(255),random(255));
    rect(x,y, random(5,10),random(5,15));
    }
  }
```

Seleção Múltipla

Quando se têm multiplas escolhas a fazer. Exemplo 1:

```
int num = 1;

switch(num) {
   case 0:
      println("Zero"); // Does not execute
      break;
   case 1:
      println("One"); // Prints "One"
      break;
   default:
}
```

Seleção Múltipla

Quando se têm multiplas escolhas a fazer. Exemplo 2:

```
char letter = 'N';

switch(letter) {
  case 'A':
  case 'a':
    println("Alpha"); // Does not execute
    break;
  case 'B':
  case 'b':
    println("Bravo"); // Does not execute
    break;
  default: // Default executes if the case labels
    println("None"); // don't match the switch parameter
    break;
}
```

Seleção Múltipla

Quando se têm multiplas escolhas a fazer. Exemplo 3a:

```
void setup() {
  size(200, 200);
}

char letter='a';
void draw() {

  switch(letter) {
  case 'R':
  case 'r':
    background(255, 0, 0);
    break;
  case 'G':
  case 'g': //
    background(0, 255, 0);
    break;
  case 'B':
  case 'b':
    background(0, 0, 255, 0);
    break;
  }
}
```

Seleção Múltipla

Quando se têm multiplas escolhas a fazer. Exemplo 3a (continuação):

```
void keyPressed()
{
  letter = key;
  switch(letter) {
  case 'R':
  case 'r':
    println("Encarnado");
    break;
  case 'G':
  case 'g':
    println("Verde");
    break;
  case 'B':
  case 'b':
    println("Azul");
    break;
  }
}
```

Vetores (Parte 1 de 2)

```
int pingos=150;
int [] chuvax;
int [] chuvay;

void setup() {
  size(200, 200);
  chuvax = new int[pingos];
  chuvay = new int[pingos];
  for (int i=0; i<pingos; i++)
  {
    chuvax[i] = floor(random(width));
    chuvay[i] = floor(random(height));
  }
  smooth();
}
```

Vetores (Parte 2 de 2)

```
void draw() {
  stroke(150);
  for (int i=0; i<pingos; i++)
    line(chuvax[i], chuvay[i],chuvax[i]+3, chuvay[i]+10 );
}

void keyPressed()
{
  background(200);
  for (int i=0; i<pingos; i++)
  {
    chuvax[i] = floor(random(width));
    chuvay[i] = floor(random(height));
  }
}
```

Imagens, Formas, Texto, etc.

PShape
PImage
PFont
PGraphics
PVector // s em java mode

Dados compostos

Array
String
ArrayList
*Dict

Bibliotecas

Recursos pré-programados de funcionalidade interessante.

Video
DXF
PDF export
Audio
Rede