PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO

**Estruturado**

**Características Principais:**

Estruturas básicas de controle: sequência, condição e repetição.  
 São formas de raciocínio intuitivamente óbvias. A legibilidade e compreensão de cada bloco de código na solução é enormemente incrementada, proibindo o uso irrestrito de comandos de desvio incondicional (GOTO).  
Subprogramação (ou modularização).

À medida que os programas vão se tornando maiores e mais complexos, é possível simplificar e melhorar a clareza dividindo o programa em partes menores, chamadas subprogramas.

- Vantagens/Desvantagens

As vantagens desse paradigma são: eficiência (porque embute o modelo de Von Neumann); modelagem “natural” de aplicações do mundo real; paradigma dominante e bem estabelecido; e também muito flexível.

As desvantagens são: difícil legibilidade; as instruções são centradas no como e não no o que.

- Linguagens Suportadas

Ada

ALGOL

Assembler

Basic

C

Cobol

Fortran

Pascal

Python

Lua;

- 1 Exemplo de Código

Exemplo:

function fatorial (n: integer):integer;

var fat: integer;

begin

fat := 1;

while (n>1) do

begin fat := fat \* n;

n := n – 1;

end;

fatorial := fat;

end;

Orientado a Objetos:

- Características Principais

A programação Orientada a Objetos é baseada na composição e interação de diversas unidades de softwares denominados objetos. O funcionamento de um software orientado a objetos se dá através do relacionamento e troca de mensagens entre esses objetos. Esses objetos são classes, e nessas classes os comportamentos são chamados de métodos e os estados possíveis da classe são chamados de atributos. Nos métodos e nos atributos também são definidas as formas de relacionamento com outros objetos

- Vantagens/Desvantagens

Esse paradigma possui todas as vantagens do paradigma imperativo entre outras: a alteração de um módulo não incorre na modificação de outros módulos; quanto mais um módulo for independente, maior a chance dele poder ser reutilizado em outra aplicação

Por exigir formas de pensar relativamente complexas, a programação orientada a objetos até hoje ainda não é bem compreendida ou usada pela maioria.

Linguagens Suportadas

• Smalltalk;

• Python;

• Ruby;

• C++;

• Object Pascal;

• Java;

• C#; •

Oberon;

• Ada;

• Eiffel;

• Simula;

• .NET

- 1 Exemplo de Código

Exemplo:

package exemplo

; public class Conta String

{ nrDaCOnta; String descricao;

double saldo; double limite;

Cliente cliente = new Cliente();

public boolean saque (double valor){

if (valor<=(saldo+limite)){ saldo-=valor; return true; } else{ return false;

}

} public boolean deposita (double valor){

if (valor<=(saldo+limite)

){ saldo+=valor; return true;

} else{ return false; } }

public boolean transfere (Conta c, double valor){

if (valor<=(saldo+limite)){

c.saldo+=valor; saldo-=valor; return true;

} else{ return false; } } }

Orientado a Eventos

Características

Diferente de aguardar por um comando completo que processa a informação, o sistema em tal paradigma é programado em sua base em um [laço de repetição](https://pt.wikipedia.org/wiki/La%C3%A7o_de_repeti%C3%A7%C3%A3o) de eventos, que recebem repetidamente informação para processar e disparam uma função de resposta de acordo com o evento.

O método pelo qual a informação é adquirida por camadas mais baixas do sistema é irrelevante. As entradas podem ser [enfileiradas](https://pt.wikipedia.org/wiki/Fila) ou uma [interrupção](https://pt.wikipedia.org/wiki/Interrup%C3%A7%C3%A3o) pode ser registrada para reagir, ou ainda ambos.

Vantagens/Desvantagens

Uma vantagem da programação orientada a eventos é que o conceito de eventos facilita a criação e manipulação de interfaces gráficas para programas. A desvantagem é que o entendimento deste paradigma é complicado para iniciantes, pois a maioria está acostumada a programas sequenciais, com uma única entrada e uma única saída.

- Linguagens Suportadas

Exemplo de Código

#!/usr/bin/env python

"""Implements simple operations on PyGTK.

Hello World is a simple Python program which

uses PyGTK to show a window with a Hello World

button. This window closes when button is clicked.

It's an adaptation of the original code in

PyGTK tutorial

"""

import pygtk

pygtk.require('2.0')

import gtk

class HelloWorld:

def \_\_init\_\_(self):

self.window = gtk.Window(gtk.WINDOW\_TOPLEVEL)

self.window.connect("delete\_event", self.delete\_event)

self.window.connect("destroy", self.destroy)

self.window.set\_border\_width(10)

self.button = gtk.Button("Hello World")

self.button.connect("clicked", self.hello, None)

self.button.connect\_object("clicked", gtk.Widget.destroy, self.window)

self.window.add(self.button)

self.button.show()

self.window.show()

def run(self):

gtk.main()

def destroy(self, widget, data=None):

print "destroy signal occurred"

gtk.main\_quit()

def delete\_event(self, widget, event, data=None):

print "delete event occurred"

return False

def hello(self, widget, data=None):

print "Hello World"

#

# Main

#

if \_\_name\_\_ == "\_\_main\_\_":

hello = HelloWorld()

hello.run()

Funcional