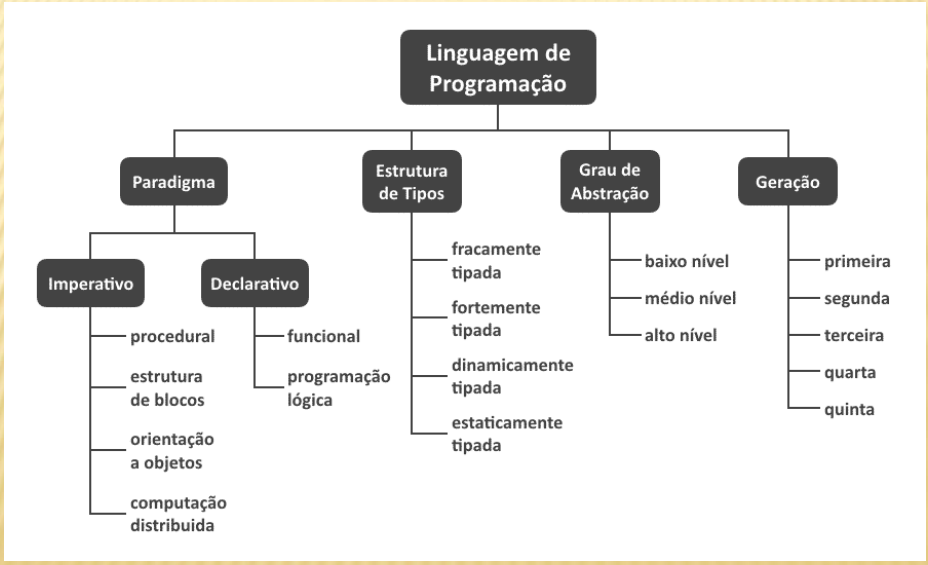
SUMÁRIO - AULA 04

- Classificação das LPs
  - Gerações
- Paradigmas
  - Visão Geral
  - 1) Imperativo / Procedural
  - 2) Estruturado

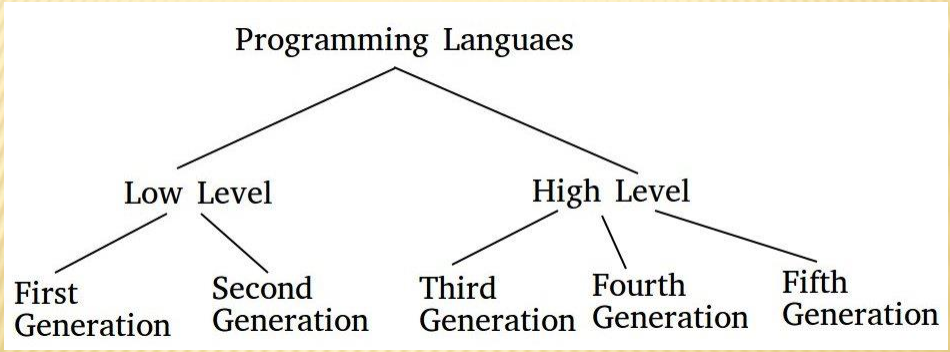
CLASSIFICAR LP

- Um tanto difícil, subjetivo. Optaram por:
  - Gostei
  - Não gostei
  - Conheço
  - Usei
  - Para que serve
  - Ordem
  - Nenhum critério?!
  - Etc.

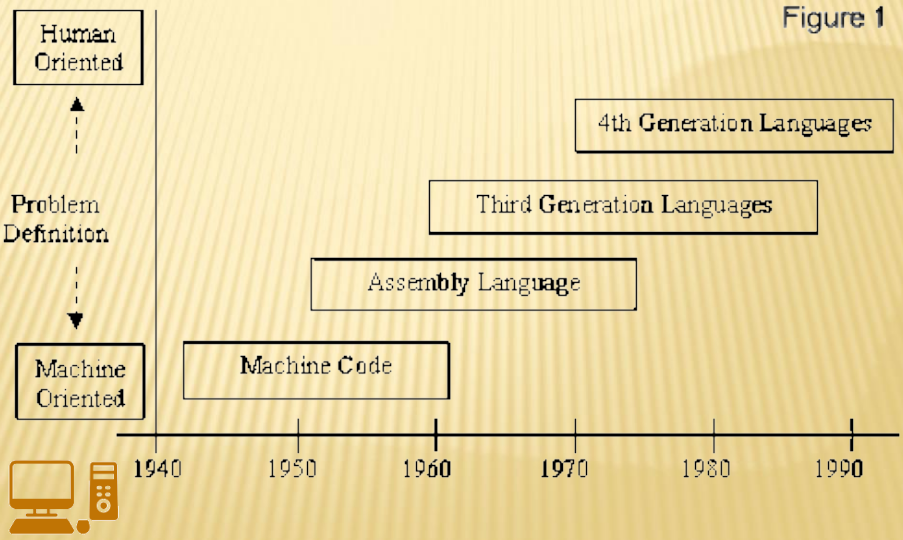
CLASSIFICAR LP



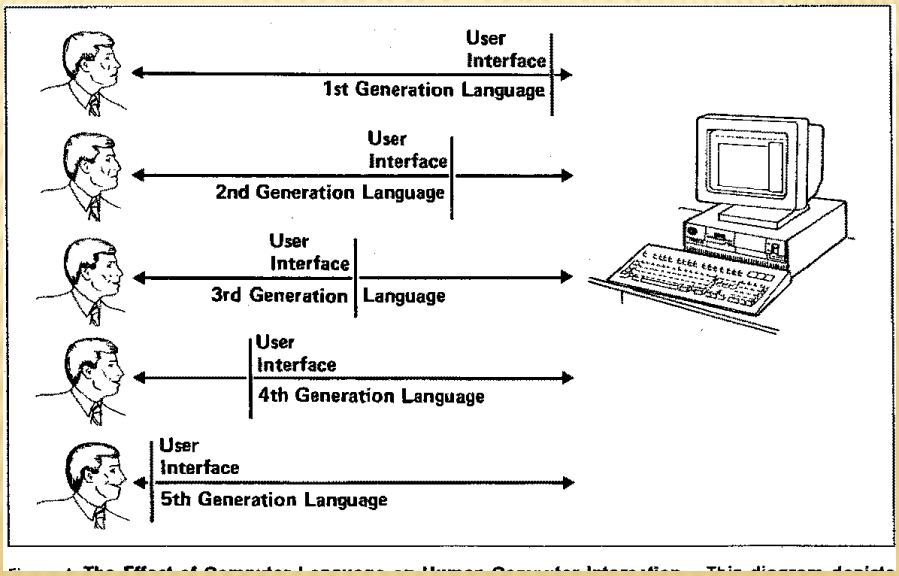
GERAÇÕES DAS LPS



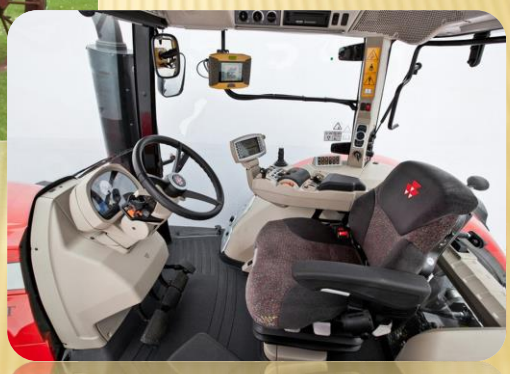
GERAÇÕES DAS LPS



GERAÇÕES DAS LPS



“USUÁRIO, SE AJUSTE AO EQUIPAMENTO!”



GERAÇÕES DAS LPS

- 1ª : linguagem máquina.
  - 10111
- 2ª : linguagens assembler (1950)
  - MOV AH, 02H
- 3ª : linguagens de alto nível (1960)
  - uso científico e comercial
  - orientadas ao usuário.
  - Ex: C, Pascal, Cobol.

GERAÇÕES DAS LPS

- **4ª** : São linguagens capazes de gerar código por si só via metodologia RAD (*Rapid Application Development*).
- Conhecidas por 4GL
  - Linguagens Específica de domínio
    - ***Domain-Specific Language*** - **DSL**
  - Permite um "leigo" desenvolver aplicações.
    - especificar o que deve ser feito.
      - Ex: **SQL**, MatLab(MatrixLaboratory). etc.

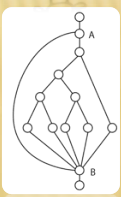
GERAÇÕES DAS LPS

- **5ª :** Aqui se encontram as linguagens para inteligência artificial.
  - "Linguagens do Conhecimento"
  - Ex: LISP, Prolog

CLASSIFICAÇÃO DAS LPS

- LPs podem ser agrupadas segundo o paradigma que seguem para abordar a sua sintaxe e semântica.

- Os paradigmas se dividem em 2 grupos:
  - Declarativo (*what*):
    - sem fluxo de controle
      - *what we want to get, not what to do.*
    - programa especifica uma relação ou função
  - Imperativo (*how*):
    - fluxo de controle é explicito
      - *what to do in what order*
    - computação por meio de mudanças de estado



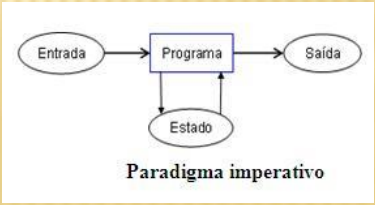
PARADIGMAS DE PROGRAMAÇÃO

PARADIGMAS DE LPS

- 1) Paradigma Imperativo/Procedural
- 2) Paradigma Estruturado
- 3) Paradigma Orientado a Objeto.
- 4) Paradigma Orientado a Aspectos (POA)
- 5) Paradigma Funcional
- 6) Paradigma Lógico
- 7) Paradigma Adaptativo/Generativo

PARADIGMAS DE LPS

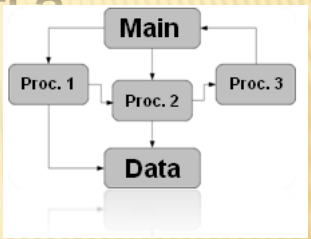
- 1) Paradigma Imperativo/Procedural
  - "imperare" em Latim significa "comandar"
  - Chamadas a procedimentos (ações/comandos) que atualizam variáveis (estado) na memória.
  - Exemplos:
    - C, Pascal, ADA, Fortran,
    - Python, Lua, Cobol, etc.



Paradigma imperativo

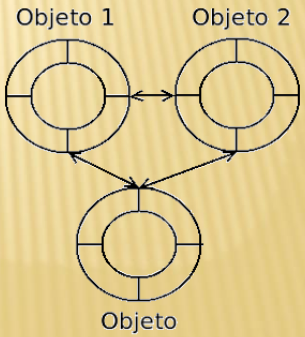
PARADIGMAS DE LPS

- 2) Paradigma Estruturado
  - Desenvolvimento *top-down*
    - criação de subprogramas/módulos
  - **Estruturas** básicas de **controle**:
    - Sequência (faça a ***Tarefa a*** e depois a ***Tarefa b***)
    - condição (**if**)
    - repetição (**while**)
  - **Eliminação** total/parcial de saltos (**GO TO**)



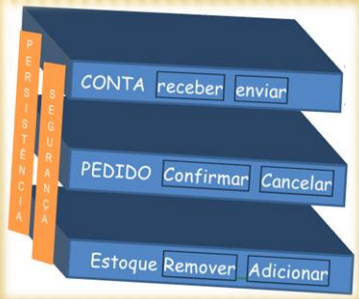
PARADIGMAS DE LPS

- 3) Paradigma Orientado a Objetos
  - Baseia-se no conceito de **classes**
    - Atributos + Métodos = Classes
  - Domínio: todos
  - Exemplos:
    - Java, C++, Simula 67, etc.



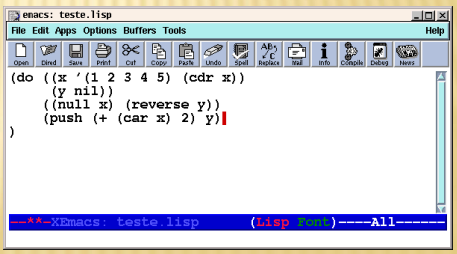
PARADIGMAS DE LPS

- 4) Paradigma Orientado a Aspectos (POA)
  - Baseia-se em capturar interesses, aspectos, características transversais que estão espalhadas pela aplicação.
    - baseado em **meta informações**



PARADIGMAS DE LPS

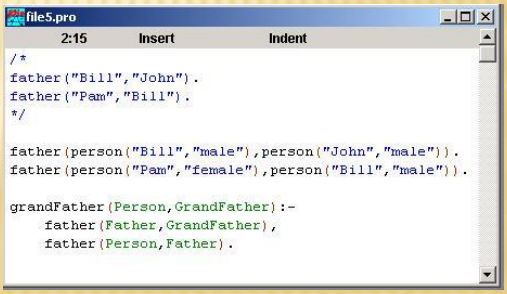
- 5) Paradigma Funcional
  - Baseia-se em princípios da matemática (funções) para realizar computações.
  - Domínio:
    - Matemática e IA
  - Exemplos:
    - Racket, LISP, ML, Scheme, CLOS, etc.



PARADIGMAS DE LPS

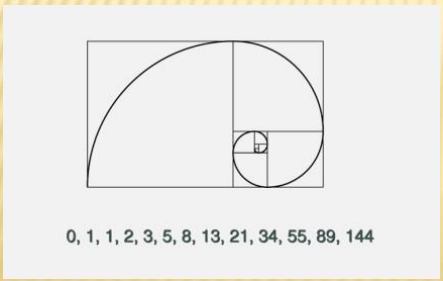
- 6) Paradigma Lógico

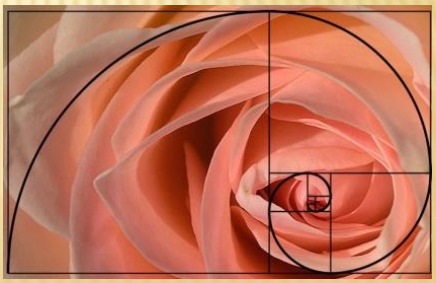
  - Programa implementa fatos e relações.
    - Lógica de Predicados, cláusulas
  - Domínio: IA
  - Exemplos:
    - PROLOG



FIBONACCI

- Proporções na Natureza
  - Regra de Ouro / Proporção áurea
    - https://www.youtube.com/watch?v=AryJTtFHFqQ





FIBONACCI

Pascal

```
program Fibonacci;

function fib(n: Integer): Integer;
var a: Integer = 1;
    b: Integer = 1;
    f: Integer;
    i: Integer;
begin
  if (n = 1) or (n = 2) then
    fib := 1
  else
    begin
      for i := 3 to n do
      begin
        f := a + b;
        b := a;
        a := f;
      end;
      fib := f;
    end;
end;

begin
  WriteLn(fib(6));
end.
```

Python

```
class Fibonacci:
  def __init__(self):
      self.cache = {}
  def fib(self, n):
      if self.cache.has_key(n):
          return self.cache[n]
      if n == 1 or n == 2:
          return 1
      else:
          a = 1
          b = 1
          for i in range(2, n):
              f = a + b;
              b = a;
              a = f;
          self.cache[n] = f;
          return f;

fibonaccyCounter = Fibonacci()
print fibonaccyCounter.fib(6)
```

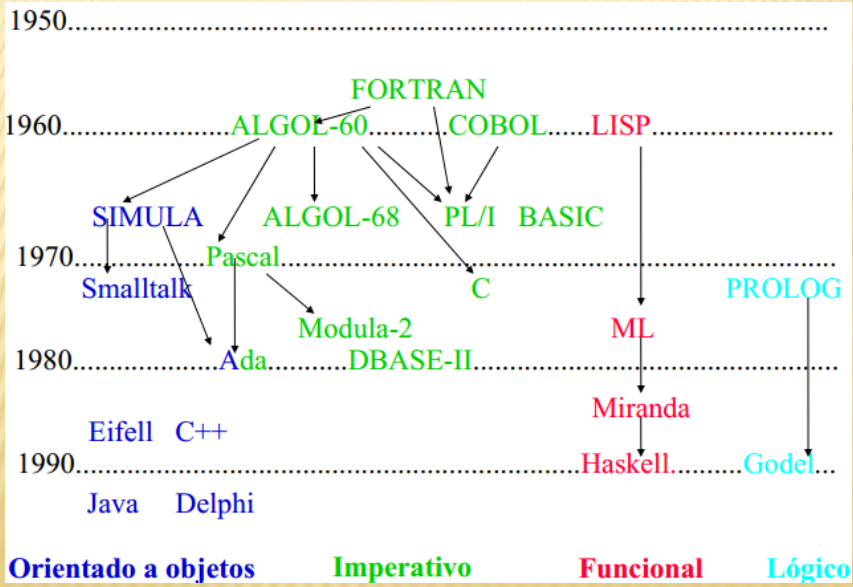
Haskell

```
import Text.Printf

fib :: Int -> Int
fib 0 = 0
fib 1 = 1
fib n = fib (n-1) + fib (n-2)

main = printf "%d\n" (fib 6)
```

Prolog

```
fib(1, 1).
fib(2, 1).


fib(X, Y):-
      X > 1,
      X1 is X - 1,
      X2 is X - 2,
      fib(X1, Z),
      fib(X2, W),
      Y is W + Z.


main :-
  fib(6,X), write(X), nl.
```

BREVE HISTÓRICO DE LPS



CRONOLOGIA DAS LPS

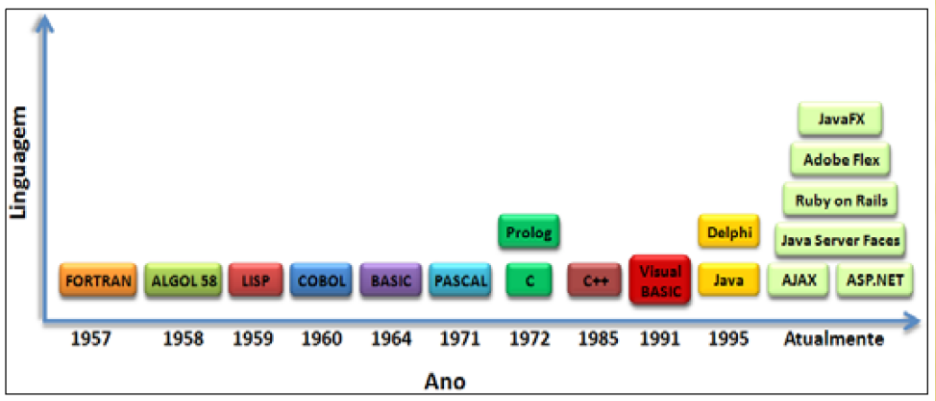


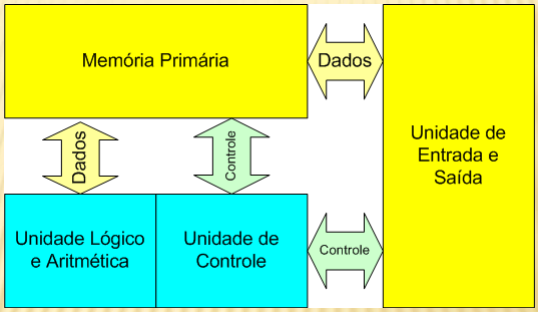
Figura 2 *Ordem cronológica da criação das principais linguagens de programação.*

PARADIGMAS DE LPS

- 1) Paradigma Imperativo/Procedural
- 2) Paradigma Estruturado
- 3) Paradigma Orientado a Objeto.
- 4) Paradigma Orientado a Aspectos (POA)
- 5) Paradigma Funcional
- 6) Paradigma Lógico
- 7) Paradigma Adaptativo/Generativo

1) PARADIGMA IMPERATIVO/PROCEDURAL

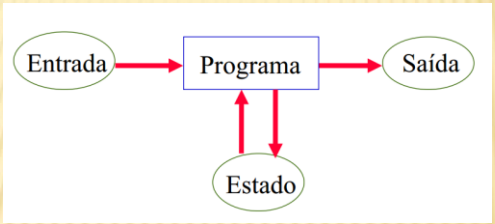
- É baseado na arquitetura de Von Neumann.



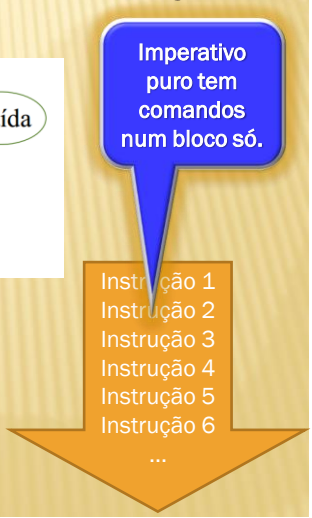
- Foi o primeiro paradigma utilizado.
  - Fortran, BASIC *(Beginner's All-purpose Symbolic Instruction Code)*
    - código de Instruções Simbólicas de Uso Geral para Principiantes

1) PARADIGMA IMPERATIVO/PROCEDURAL

- Segue o conceito de um estado e de ações que manipulam esse estado:



- Conceitos que predominam:
  - variáveis
  - atribuições
  - sequência



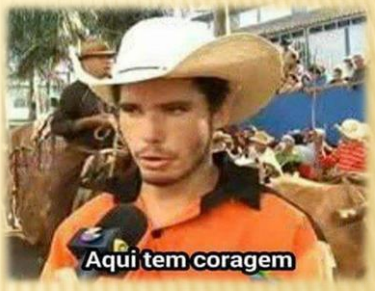
1) PARADIGMA IMPERATIVO/PROCEDURAL

- O procedural já inclui sub-rotinas ou procedimentos para estruturação.
  - Ex: C e Pascal, Assembly
- Admitem saltos (*jumps*) no código, o que dificulta a depuração.

1) PARADIGMA IMPERATIVO/PROCEDURAL

- SO feito em Assembly!
  - http://www.menuetos.net

MenuetOS.net



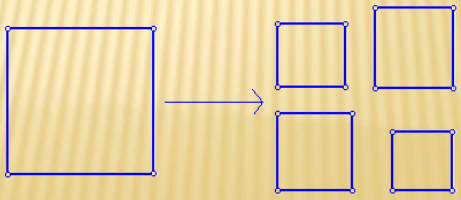
- Simulador Assembly
  - https://schweigi.github.io/assembler-simulator

1) PARADIGMA IMPERATIVO/PROCEDURAL

- Procedural oferece:
  - A possibilidade de dividir um problema complexo em partes menores (*procedures*, procedimentos), mais fáceis de se lidar, e manipular a sequência das instruções.
  - Assim, a qualquer momento do programa, é possível ir a outra parte. Ex:
    - ...
    - ...
    - call procA;

1) PARADIGMA IMPERATIVO/PROCEDURAL

- Uma linguagem pode ser **imperativa** sem ser **procedural** ou **estruturada**. Ex: Assembly
- A divisão de uma **funcionalidade** em **funções** é o que faz uma linguagem ser **imperativo/procedural estruturada**.



1) PARADIGMA IMPERATIVO/PROCEDURAL

- Subdivide-se em:
  - **estruturado** (com blocos + if, else, while) e
    - leitura mais agradável do codigo
  - **não-estruturado** (monobloco + goto). Ex:

```
10 INPUT A,B,C
20 LET SOMA = A+B+C
30 LET MEDIA = SOMA/3
40 PRINT MEDIA
50 PRINT "DESEJA CONTINUAR (S/N)?"
60 INPUT RESPOSTA
70 IF RESPOSTA = "S"THEN GOTO 10
80 END
```

Exemplo em Basic

```
read(x);
2: if x = 0 then goto 8;
writeln(x);
4: read(next);
if next = x then goto 4;
x := next;
goto 2;
8: ...;
```

Exemplo em Pascal

1) PARADIGMA IMPERATIVO/PROCEDURAL

- Eficiência
  - porque incorpora o modelo de Von Neumann;
- Paradigma dominante e bem estabelecido nas LPs, pois tais linguagens são mais fáceis de traduzir para uma forma adequada para execução na máquina.

1) PARADIGMA IMPERATIVO/PROCEDURAL

- Difícil legibilidade:
  - saltos
- As instruções são centradas no como um processamento deve ser feito
  - *what to do in what order*
  - e não no o que está fazendo.

FORTRAN

FORTRAN SPECIALIST GROUP
BCS
THE BRITISH COMPUTER SOCIETY

Example code - FORTRAN 0

```
      DIMENSION A(11)
      READ A
2     DO 3,8,11 J=1,11
3     I=11-J
      Y=SQRT(ABS(A(I+1)))+5*A(I+1)**3
      IF (400>=Y) 8,4
4     PRINT I,999.
      GOTO 2
8     PRINT I,Y
11    STOP
```

http://www.fortran.bcs.org/2005/fortran/img8.html

229

BASIC

```
180 READ N$, S
190                                          REM
200 IF N$ = "END OF FILE" THEN 330
210   IF S >= 500.00 THEN 250
220     LET C = S*P2
230     GOTO 280
240                                          REM
250     LET C = S*P1
260     GOTO 280
270                                          REM
280   LET C1 = C1 + C
290   PRINT N$, S, C
300   READ N$, S
310 GOTO 200
320                                          REM
330 PRINT "Total Commission: "; C1
340 END
```

http://www.logicgenerator.com/Tutorial/FlowchartTutorial01.htm

230

1) PARADIGMA IMPERATIVO/PROCEDURAL

- Não estruturado
  - linguagens caracterizadas pela utilização da expressão **goto**.
    - Usado para repetição e seleção de instruções para execução
    - código "spaguetti"

    - não segue regras de programação estruturada e abusa de desvios.

1) PARADIGMA IMPERATIVO/PROCEDURAL

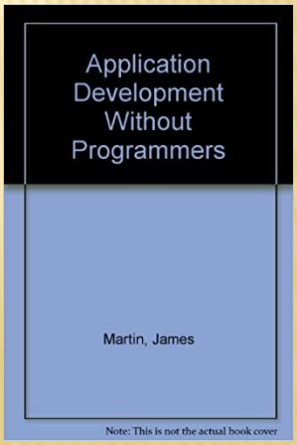
```
template <typename T>
void goto_sort( T array[], size_t n ) {
        size_t i{1}

        first: T current{array[i]};
        size_t j{i};

        second: if ( array[j - 1] <= current ) {
                goto third;
        }

        array[j] = array[j - 1];

        if ( --j ) {
                goto second;
        }

        third: array[j] = current;

        if ( ++i != n ) {
                goto first;
        }
}
```

ATIVIDADE

- Pesquise sobre o uso de algumas linguagens imperativas. Exs:
  - FORTRAN
  - BASIC
  - COBOL
  - Pascal
  - C
  - Python
  - ALGOL
  - Modula

ATIVIDADE

- Pesquise sobre as linguagens de programação de quarta geração, ou **4GL**. Você usa alguma?



- Já em 1982 James Martin preconizavva **"*Applications Development Without Programmers*".** Deu certo?

CURIOSIDADE

- ***Hello World*** em 28 linguagens:
  - https://excelwithbusiness.com/blog/say-hello-world-in-28-different-programming-languages

CURIOSIDADE

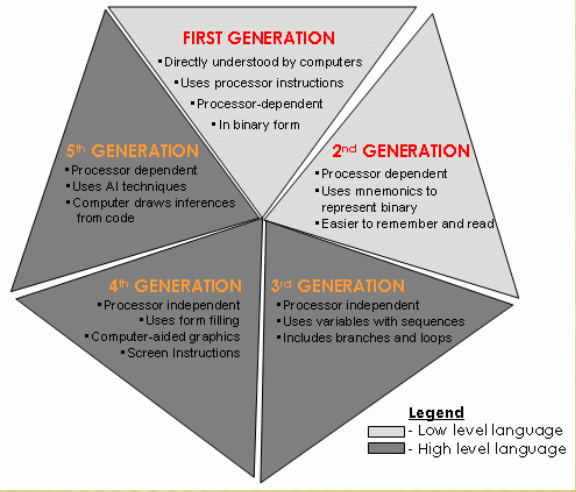
- Código binário nos computadores? Como?
  - **Claude Elwood Shannon** (1916 - 2001)
    - http://www.kerryr.net/pioneers/shannon.htm
    - Em 1937 usou os conceitos de George Boole, inventor da álgebra booleana (1860)

CURIOSIDADE

- Quem foi o primeiro programador(a)?
  - **Augusta Ada Byron** (Condessa de Lovelace) escreveu em 1843 o primeiro algoritmo para ser processado por uma máquina mecânica, a máquina analítica de Charles Babbage.

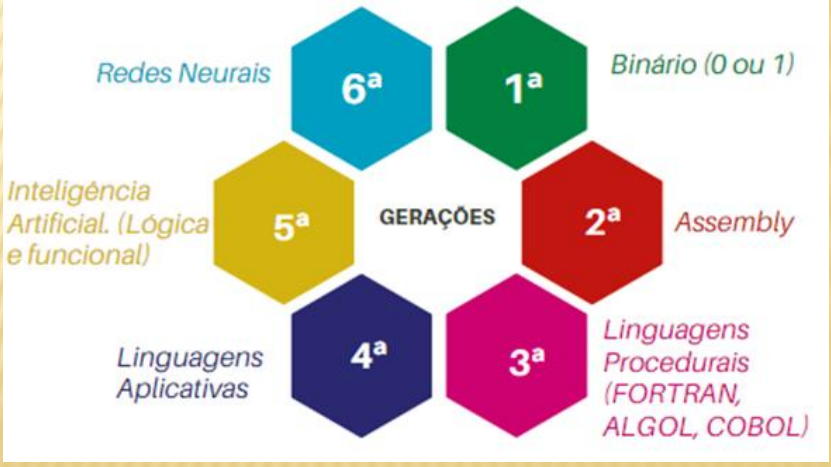
ATIVIDADE

- Pesquise sobre as 5 gerações de linguagens



ATIVIDADE

- Opa! 6 gerações de linguagens?! Quem é a 6ª?



2) PARADIGMA ESTRUTURADO

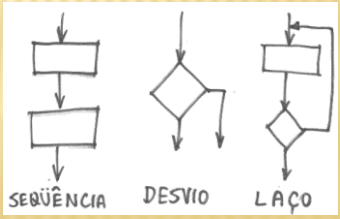
- Linguagens estruturadas surgiram com o objetivo de **facilitar** a **leitura** e acompanhar a execução de algoritmos.
- Normalmente linguagens estruturadas **não** fazem uso de comando **goto**.
  - programação baseada em **estruturas de controle** ao invés de **rótulos**

2) PARADIGMA ESTRUTURADO

- As primeiras linguagens usavam **rótulos** e **goto**'s para criar desvios de fluxo.
- **Algol** e similares vem com nova proposta baseada em **estruturas de controle**
- Instruções são agrupadas em **blocos** acionáveis via **if** e **while**

2) PARADIGMA ESTRUTURADO

- Desenvolvimento via refinamentos sucessivos
  - Quebrando o problema em **unidades menores**
- Preconiza que todos os programas possíveis podem ser reduzidos a apenas 3 estruturas:
  - sequencia
  - decisão e
  - iteração.

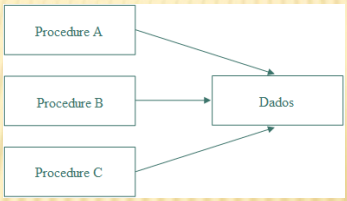


2) PARADIGMA ESTRUTURADO

- Proíbe o uso irrestrito de comandos de desvio incondicional (GOTO)
- Na prática, foi transformada na programação modular (subprogramação).
  - criação de módulos interligados entre si através de uma interface comum.

2) PARADIGMA ESTRUTURADO

- Orienta os programadores para a criação de estruturas simples em seus programas, usando as sub-rotinas e as funções.



- Foi a forma dominante na criação de software entre a programação linear e a programação orientada a objetos.

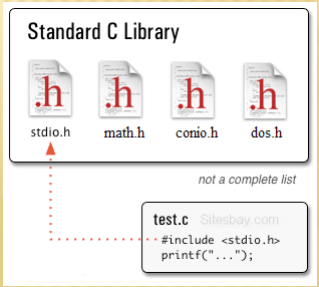
2) PARADIGMA ESTRUTURADO

- Subprograma e seus sinônimos:
  - procedimento
  - função
  - módulo (estrutura modular)
  - métodos (orientação a objetos)
  - subrotina.
- Lema: “dividir para conquistar”

2) PARADIGMA ESTRUTURADO

- Subprogramas que visam solucionar tarefas corriqueiras formam uma biblioteca.

- Reutilização de subprogramas através de bibliotecas de subprogramas.



2) PARADIGMA ESTRUTURADO

- Em C, via **.h**:
  - arquivo-interface (**header-file**)

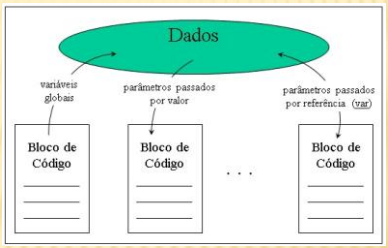
```
#include<stdio.h> /* para usar a função scanf */

void main()
{

 char nome[30];
 int idade;

 printf("Digite seu primeiro nome:");
 scanf("%s",&nome); /*usa-se %s pq se trata de uma string de caracteres*/
 printf("\nDigite a sua idade:");
 scanf("%d", &idade); /*usa-se %d pq se trata de um número inteiro*/

 printf("\n%s tem %d anos!!!",nome,idade);

}
```

2) PARADIGMA ESTRUTURADO

- Os problemas podem ser **quebrados** em vários **subproblemas**;
- A **boa legibilidade** e a **boa compreensão** da estrutura deste paradigma motivam os programadores a **iniciarem** na programação pelo modelo estruturado.
  - POO é mais complicado...

2) PARADIGMA ESTRUTURADO

- **Funções** devem **conhecer** a estrutura dos dados;

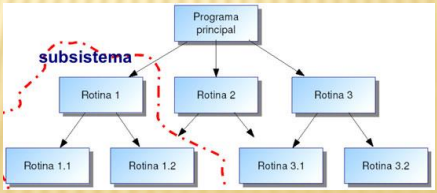


- **Mudanças** na estrutura dos **dados acarreta alteração** em todas as **funções** relacionadas.
- Gera sistemas **difíceis** de serem **mantidos**;

IMPERATIVO X ESTRUTURADO

- No **imperativo** as instruções são centradas em **ações**, enunciados ou **comandos** que mudam o **estado** do programa.

- Já o **estruturado** organiza as instruções em **blocos** que podem ser **reutilizados**.



ESTRUTURADO X OO

Programação Estruturada		Orientação a Objeto
Processo	⟷	Objeto
Revela Dados	⟷	Oculta Dados
Projeto Monolítico	⟷	Projeto Modular
Uso Único	⟷	Reutilização
Algoritmo Ordenado	⟷	Algoritmo Desordenado

PARADIGMA ADAPTATIVO/GENERATIVO

- Comportamento muda dinamicamente, em resposta a estímulos de entrada, sem a interferência de agentes externos.

- Para saber mais:
  - http://lta.poli.usp.br/lta/roteiro-de-estudos/um-roteiro-de-iniciacao-ao-estudo-e-a-pesquisa-da-tecnologia-adaptativa/paradigma-adaptativo
  - http://www.marcusramos.com.br/univasf/Uma%20Breve%20Introducao.pdf

254

ATIVIDADE

- Conhecem linguagens desenvolvidas no Brasil?
  - **Elixir** - https://elixir-lang.org

elixir

  - **Lua** - http://www.lua.org

Lua

255