Programação Imperativa

Esdras Lins Bispo Jr.
bispojr@ufg.br

Concurso para Professor de Carreira do Magistério Superior
Bacharelado em Sistemas de Informação
UPE - Campus Caruaru

20 de fevereiro de 2018

Plano de Aula

1. Motivação
2. Paradigma Imperativo
   - Breve Histórico do Paradigma
   - Características do Paradigma
   - Potencialidades e Fragilidades
3. Próxima aula...

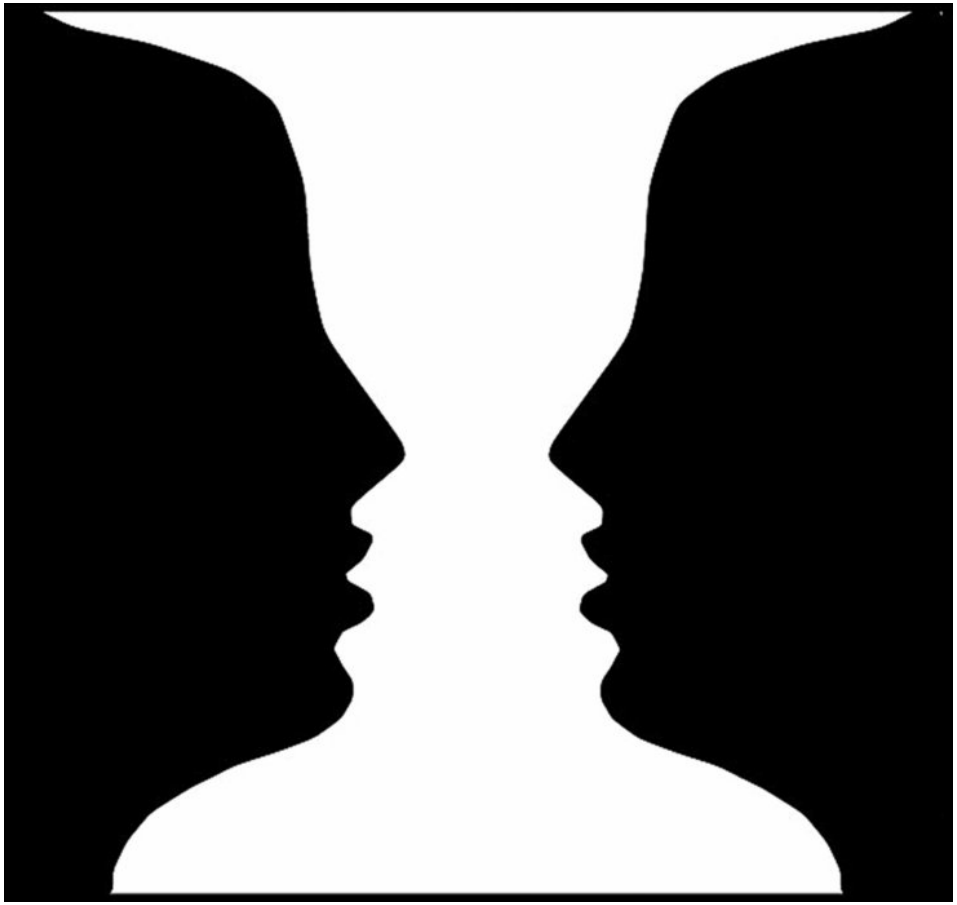
Sumário

1 Motivação

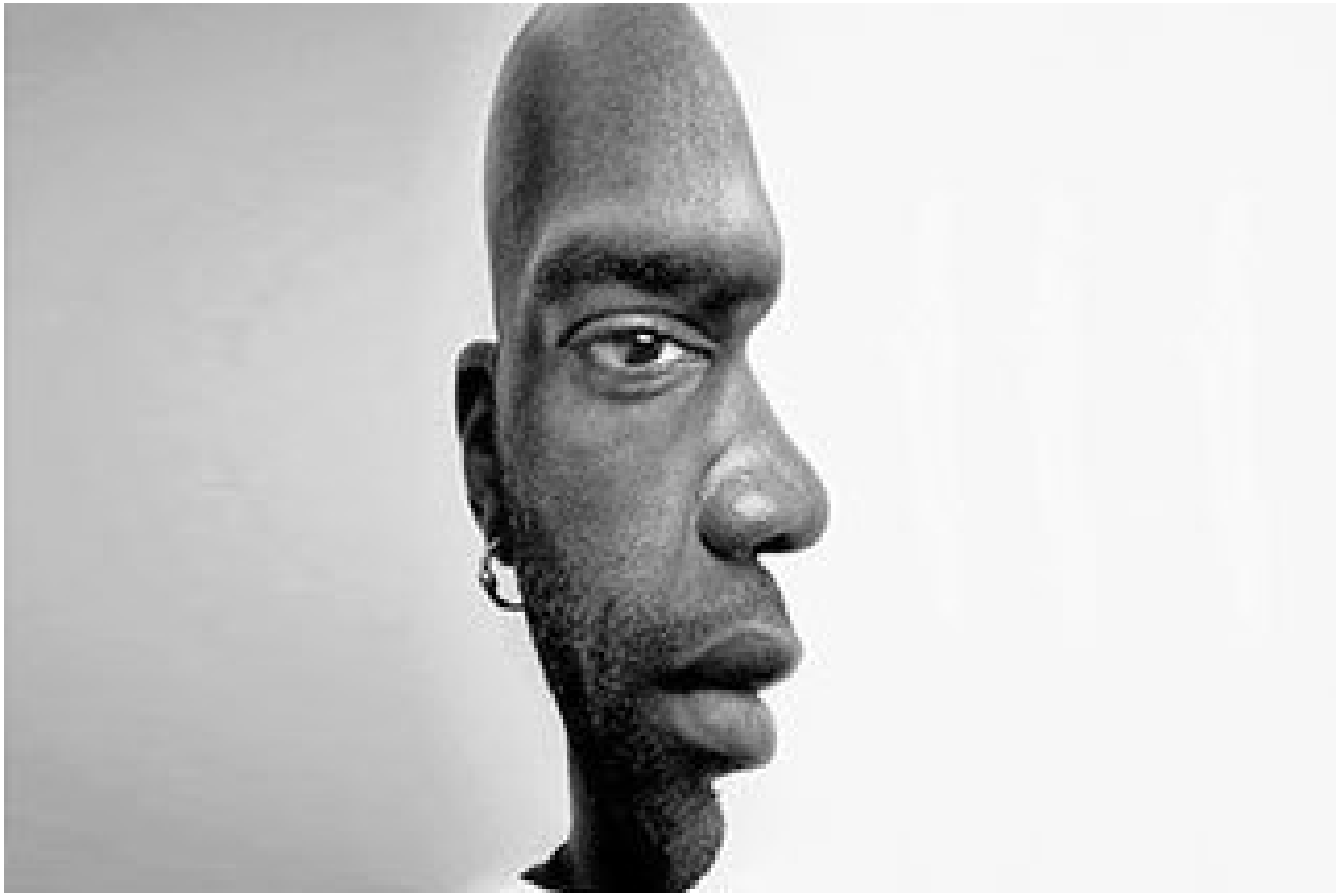
2 Paradigma Imperativo

3 Próxima aula...

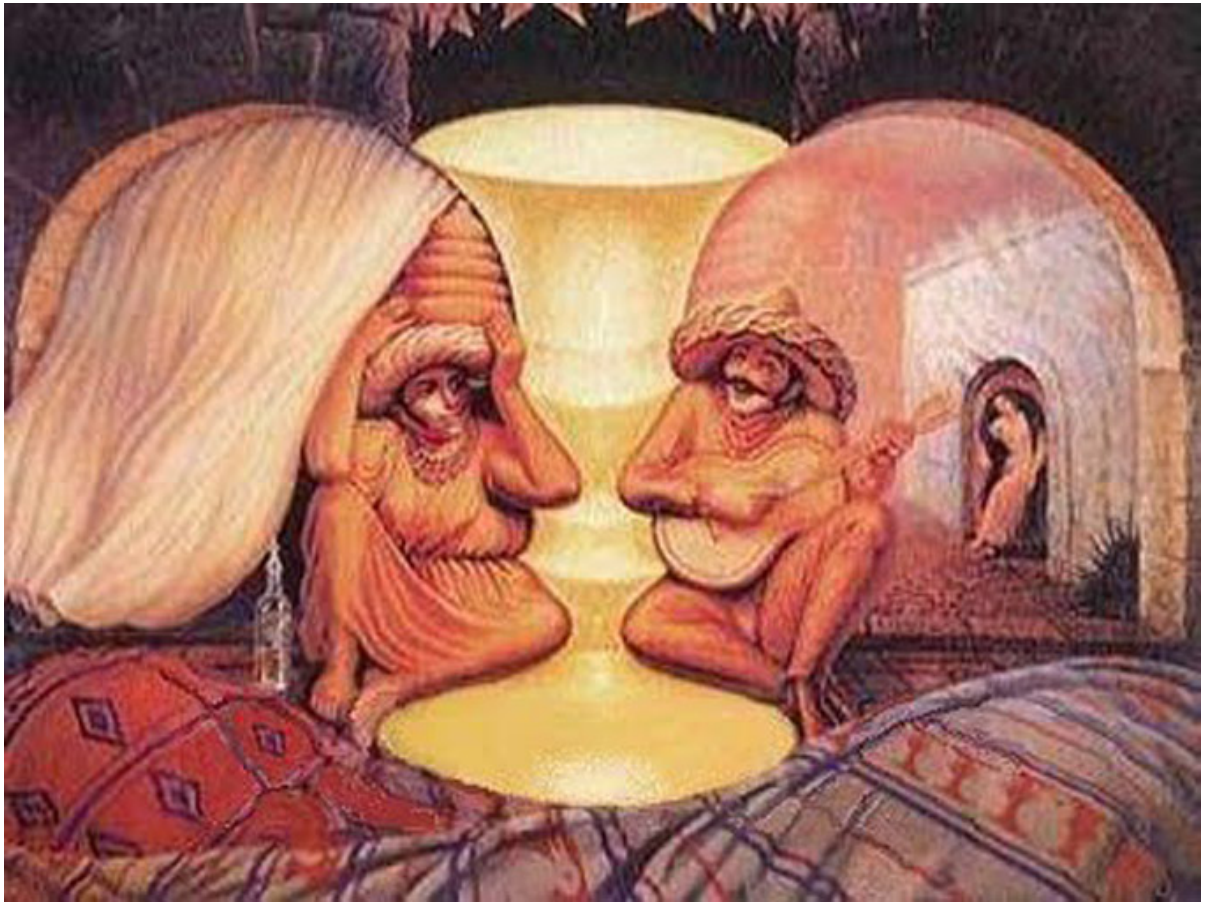
O que você está vendo?



O que você está vendo?



O que você está vendo?



O que é um paradigma?

Paradigma

Um **padrão de pensamento** que guia um conjunto de atividades relacionadas.

O que é um paradigma?

Paradigma

Um **padrão de pensamento** que guia um conjunto de atividades relacionadas.

Paradigma de Programação

Um **padrão de resolução de problemas** que se relaciona a um determinado gênero de programas e linguagens.

Dúvida Comum...

Pergunta

Qual é a melhor linguagem?

Dúvida Comum...

Pergunta

Qual é a melhor linguagem?

Resposta

Para qual tipo de problema?

Dúvida Comum...

Pergunta

Qual é a melhor linguagem?

Resposta

Para qual tipo de problema?

Tome nota...

A natureza do problema apontará
para qual linguagem devemos utilizar.

O que é um paradigma?

Alguns Paradigmas de Programação...

- Programação imperativa
- Programação orientada a objeto
- Programação funcional
- Programação lógica

O que é um paradigma?

Alguns Paradigmas de Programação...

- Programação imperativa
- Programação orientada a objeto
- Programação funcional
- Programação lógica

Aula de hoje...

Paradigma Imperativo

Sumário

1 Motivação

2 Paradigma Imperativo
- Breve Histórico do Paradigma
- Características do Paradigma
- Potencialidades e Fragilidades

3 Próxima aula...

Programa Armazenado

Ideia

Um programa e seus dados podem residir na memória principal de um computador.

Quem?

John von Neumann (1940)

Programa Armazenado

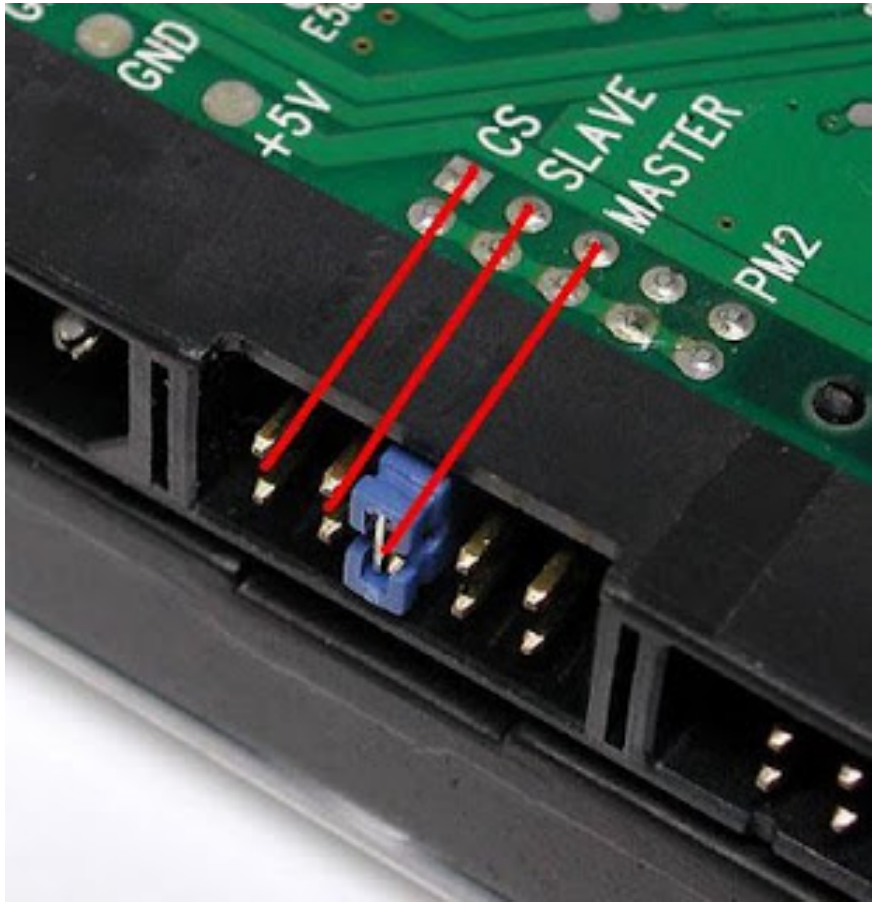
Ideia

É possível simular uma máquina de Turing em uma outra máquina de Turing.

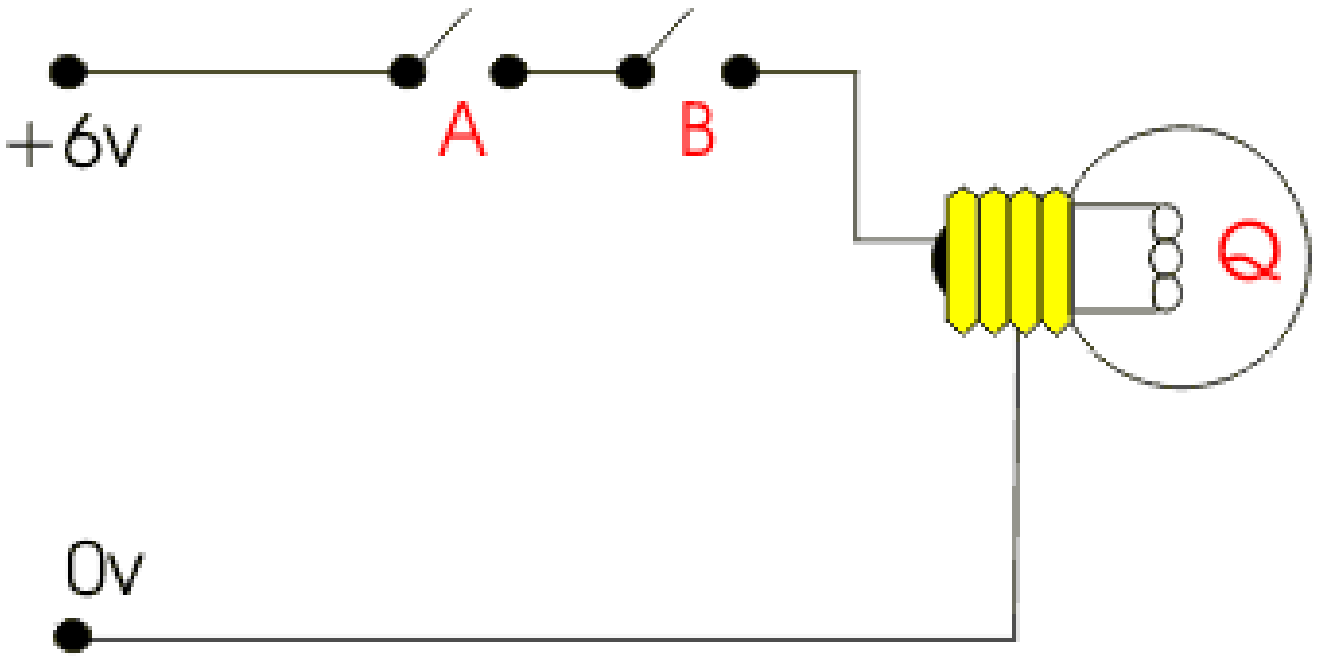
Quem?

Alan Turing (1936)

Um outro tipo de programação

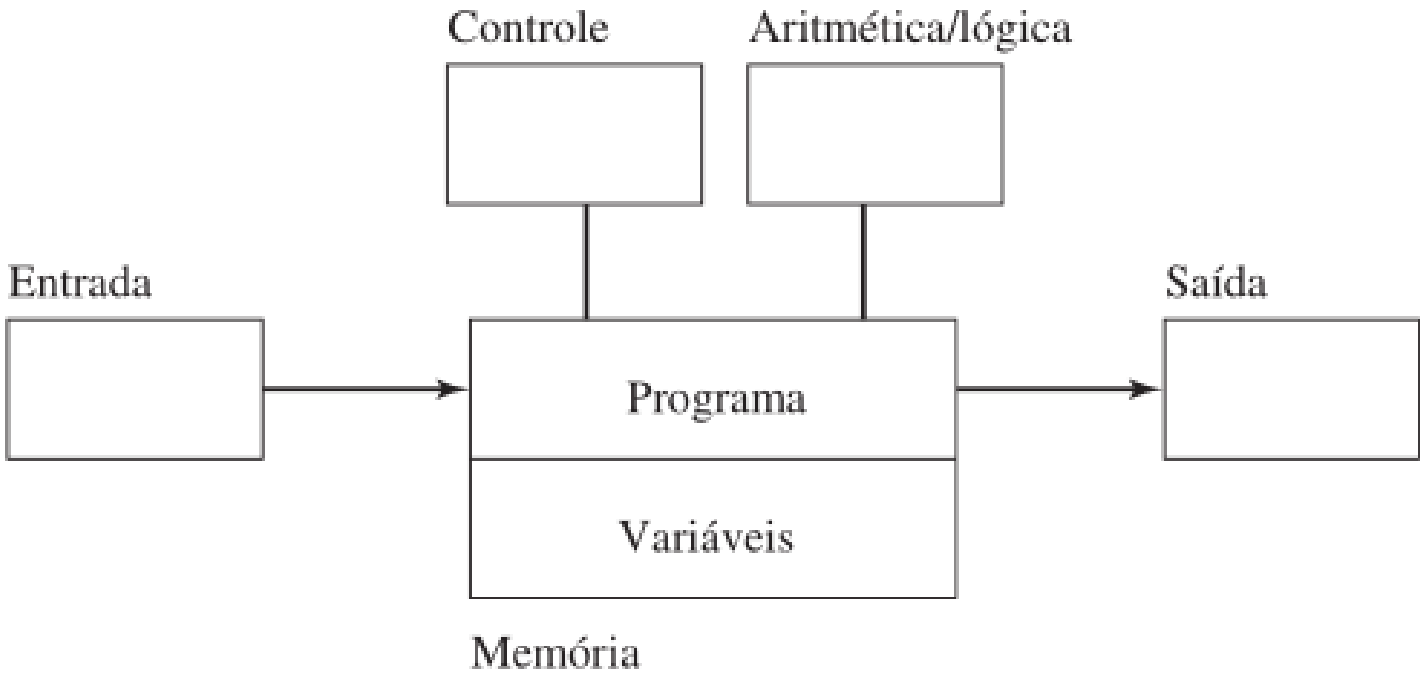


Um outro tipo de programação



Calculadora Simples

Modelo de Neumann-Eckert



Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;

Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;
- Armazenamento de programa: instruções;

Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;
- Armazenamento de programa: instruções;
  - Instruções de atribuição;

Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;
- Armazenamento de programa: instruções;
  - Instruções de atribuição;
  - Instruções condicionais;

Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;
- Armazenamento de programa: instruções;
  - Instruções de atribuição;
  - Instruções condicionais;
  - Instruções de ramificação.

Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;
- Armazenamento de programa: instruções;
  - Instruções de atribuição;
  - Instruções condicionais;
  - Instruções de ramificação.
  - Linguagem "completa quanto a Turing".

Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;
- Armazenamento de programa: instruções;
  - Instruções de atribuição;
  - Instruções condicionais;
  - Instruções de ramificação.
  - Linguagem "completa quanto a Turing".
- Armazenamento de dados: valores de dados;

Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;
- Armazenamento de programa: instruções;
  - Instruções de atribuição;
  - Instruções condicionais;
  - Instruções de ramificação.
  - Linguagem "completa quanto a Turing".
- Armazenamento de dados: valores de dados;
- Atribuição:

Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;
- Armazenamento de programa: instruções;
  - Instruções de atribuição;
  - Instruções condicionais;
  - Instruções de ramificação.
  - Linguagem "completa quanto a Turing".
- Armazenamento de dados: valores de dados;
- Atribuição:
  - alterar o valor de um local de memória;

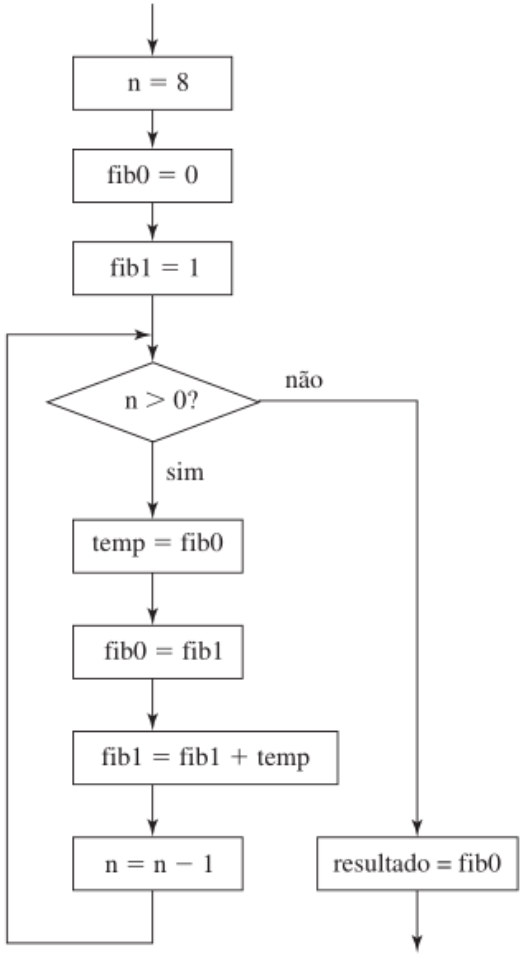
Modelo de Neumann-Eckert

Características

- Contém tanto instruções quanto dados;
- Armazenamento de programa: instruções;
  - Instruções de atribuição;
  - Instruções condicionais;
  - Instruções de ramificação.
  - Linguagem "completa quanto a Turing".
- Armazenamento de dados: valores de dados;
- Atribuição:
  - alterar o valor de um local de memória;
  - destruindo o valor anterior.

UNIVERSIDADE DE PERNAMBUCO

Exemplo: Diagrama de fluxo



Problemas com *goto*

Ideia

O uso excessivo de comandos de ramificação (ou "*go to*") é prejudicial ao processo de desenvolvimento de programas confiáveis.

Quem?

Edsger Dijkstra (1968)

UNIVERSIDADE DE PERNAMBUCO

Consequência da Evolução do Paradigma

Características Atuais

Além de ser completa quanto a Turing, a linguagem necessita ter

- Expressões e atribuição;

Consequência da Evolução do Paradigma

Características Atuais

Além de ser completa quanto a Turing, a linguagem necessita ter

- Expressões e atribuição;
- Estruturas de controle;

Consequência da Evolução do Paradigma

Características Atuais

Além de ser completa quanto a Turing, a linguagem necessita ter

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;

Consequência da Evolução do Paradigma

Características Atuais

Além de ser completa quanto a Turing, a linguagem necessita ter

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;
- Manipulação de exceções e erros;

Consequência da Evolução do Paradigma

Características Atuais

Além de ser completa quanto a Turing, a linguagem necessita ter

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;
- Manipulação de exceções e erros;
- Abstração procedural;

Consequência da Evolução do Paradigma

Características Atuais

Além de ser completa quanto a Turing, a linguagem necessita ter

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;
- Manipulação de exceções e erros;
- Abstração procedural;
- Suporte de biblioteca para estruturas de dados.

Características do Paradigma

Características

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;
- Manipulação de exceções e erros;
- Abstração procedural;
- Suporte de biblioteca para estruturas de dados.

Expressões de Atribuição

Necessidades

- Fundamental para todas as linguagens imperativas;

Expressões de Atribuição

Necessidades

- Fundamental para todas as linguagens imperativas;
- Símbolos populares:
  - Fortran (=)

Expressões de Atribuição

Necessidades

- Fundamental para todas as linguagens imperativas;
- Símbolos populares:
  - Fortran (=)
  - Algol (:=).

Expressões de Atribuição

Necessidades

- Fundamental para todas as linguagens imperativas;
- Símbolos populares:
  - Fortran (=)
  - Algol (:=).
- Semântica:

Expressões de Atribuição

Necessidades

- Fundamental para todas as linguagens imperativas;
- Símbolos populares:
  - Fortran (=)
  - Algol (:=).
- Semântica:
  - expressão avaliada para um valor;
  - valor copiado para o destino;
  - utiliza a semântica de cópia.

Expressões de Atribuição

Necessidades

- Fundamental para todas as linguagens imperativas;
- Símbolos populares:
  - Fortran (=)
  - Algol (:=).
- Semântica:
  - expressão avaliada para um valor;
  - valor copiado para o destino;
  - utiliza a semântica de cópia.

Expressões

Podem usar operadores lógicos e aritméticos, e/ou chamadas a funções-padrão da linguagem.

Características do Paradigma

Características

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;
- Manipulação de exceções e erros;
- Abstração procedural;
- Suporte de biblioteca para estruturas de dados.

Estruturas de controle

Sequência

- Fluxo natural de execução;

Estruturas de controle

Sequência

- Fluxo natural de execução;
- Comandos de desvios:

Estruturas de controle

Sequência

- Fluxo natural de execução;
- Comandos de desvios:
  - **return**,

Estruturas de controle

Sequência

- Fluxo natural de execução;
- Comandos de desvios:
  - **return**,
  - **break**,

Estruturas de controle

Sequência

- Fluxo natural de execução;
- Comandos de desvios:
  - **return**,
  - **break**,
  - **continue**, e

Estruturas de controle

Sequência

- Fluxo natural de execução;
- Comandos de desvios:
  - **return**,
  - **break**,
  - **continue**, e
  - **go to**.

Estruturas de controle

Sequência

- Fluxo natural de execução;
- Comandos de desvios:
  - **return**,
  - **break**,
  - **continue**, e
  - **go to**.

Condicional

- Seleciona caminhos alternativos em execução;

Estruturas de controle

Sequência

- Fluxo natural de execução;
- Comandos de desvios:
  - **return**,
  - **break**,
  - **continue**, e
  - **go to**.

Condicional

- Seleciona caminhos alternativos em execução;
- Comandos comuns: **if** e **case** (ou **switch**).

Estruturas de controle

Laço

- Duas variações: teste no início ou no fim;

Estruturas de controle

Laço

- Duas variações: teste no início ou no fim;
- Em C, é chamado de **while**;

Estruturas de controle

Laço

- Duas variações: teste no início ou no fim;
- Em C, é chamado de **while**;
- O comando **for** é uma variação bastante utilizada;

Estruturas de controle

Laço

- Duas variações: teste no início ou no fim;
- Em C, é chamado de **while**;
- O comando **for** é uma variação bastante utilizada;
- Comandos de desvios podem ser utilizados.

Características do Paradigma

Características

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;
- Manipulação de exceções e erros;
- Abstração procedural;
- Suporte de biblioteca para estruturas de dados.

Entrada e Saída

Necessidades

- Recuperação de dados de uma fonte;

Entrada e Saída

Necessidades

- Recuperação de dados de uma fonte;
- Armazenamento de dados em um destino;

Entrada e Saída

Necessidades

- Recuperação de dados de uma fonte;
- Armazenamento de dados em um destino;
- As fontes e os destinos são denominados arquivos:

Entrada e Saída

Necessidades

- Recuperação de dados de uma fonte;
- Armazenamento de dados em um destino;
- As fontes e os destinos são denominados arquivos:
  - Teclado;

Entrada e Saída

Necessidades

- Recuperação de dados de uma fonte;
- Armazenamento de dados em um destino;
- As fontes e os destinos são denominados arquivos:
  - Teclado;
  - Monitor;

Entrada e Saída

Necessidades

- Recuperação de dados de uma fonte;
- Armazenamento de dados em um destino;
- As fontes e os destinos são denominados arquivos:
  - Teclado;
  - Monitor;
  - Disco rígido;

Entrada e Saída

Necessidades

- Recuperação de dados de uma fonte;
- Armazenamento de dados em um destino;
- As fontes e os destinos são denominados arquivos:
  - Teclado;
  - Monitor;
  - Disco rígido;
  - *Pen-drive*.

Entrada e Saída

Necessidades

- Recuperação de dados de uma fonte;
- Armazenamento de dados em um destino;
- As fontes e os destinos são denominados arquivos:
  - Teclado;
  - Monitor;
  - Disco rígido;
  - *Pen-drive*.

Teclado como arquivo?

Isto é irrelevante para o computador.

Características do Paradigma

Características

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;
- Manipulação de exceções e erros;
- Abstração procedural;
- Suporte de biblioteca para estruturas de dados.

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:
  - Tempo de Execução;

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:
  - Tempo de Execução;
  - Entrada e saída;

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:
  - Tempo de Execução;
  - Entrada e saída;
  - Acesso à memória, etc.

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:
  - Tempo de Execução;
  - Entrada e saída;
  - Acesso à memória, etc.
- Permite manipular erros ao invés de abortar a execução;

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:
  - Tempo de Execução;
  - Entrada e saída;
  - Acesso à memória, etc.
- Permite manipular erros ao invés de abortar a execução;
- Níveis de exceções:

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:
  - Tempo de Execução;
  - Entrada e saída;
  - Acesso à memória, etc.
- Permite manipular erros ao invés de abortar a execução;
- Níveis de exceções:
  - Hardware (e.g. divisão por zero);

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:
  - Tempo de Execução;
  - Entrada e saída;
  - Acesso à memória, etc.
- Permite manipular erros ao invés de abortar a execução;
- Níveis de exceções:
  - Hardware (e.g. divisão por zero);
  - Linguagem de Programação (e.g. índice fora do intervalo);

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:
  - Tempo de Execução;
  - Entrada e saída;
  - Acesso à memória, etc.
- Permite manipular erros ao invés de abortar a execução;
- Níveis de exceções:
  - Hardware (e.g. divisão por zero);
  - Linguagem de Programação (e.g. índice fora do intervalo);
  - Outros níveis (e.g. desempilhar pilha vazia).

Manipulação de Exceções

Necessidades

- Meio de lidar com erros inesperados:
  - Tempo de Execução;
  - Entrada e saída;
  - Acesso à memória, etc.
- Permite manipular erros ao invés de abortar a execução;
- Níveis de exceções:
  - Hardware (e.g. divisão por zero);
  - Linguagem de Programação (e.g. índice fora do intervalo);
  - Outros níveis (e.g. desempilhar pilha vazia).

Garantia de robustez

Uma aplicação é robusta quando ela continua a operar sob todas as situações de erro presumíveis.

Características do Paradigma

Características

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;
- Manipulação de exceções e erros;
- Abstração procedural;
- Suporte de biblioteca para estruturas de dados.

Abstração Procedural

Necessidades

- Programador foca:

Abstração Procedural

Necessidades

- Programador foca:
  - na interface da função;

Abstração Procedural

Necessidades

- Programador foca:
  - na interface da função;
  - no seu valor de retorno.

Abstração Procedural

Necessidades

- Programador foca:
  - na interface da função;
  - no seu valor de retorno.
- Refinamento de passos (Wirth, 1973):

Abstração Procedural

Necessidades

- Programador foca:
  - na interface da função;
  - no seu valor de retorno.
- Refinamento de passos (Wirth, 1973):
  - sequenciamento;
  - iteração;
  - seleção.

Abstração Procedural

Necessidades

- Programador foca:
  - na interface da função;
  - no seu valor de retorno.
- Refinamento de passos (Wirth, 1973):
  - sequenciamento;
  - iteração;
  - seleção.

Exemplo: Algoritmo de Ordenação

Ignora detalhes sobre como essa ordenação é executada.

Características do Paradigma

Características

- Expressões e atribuição;
- Estruturas de controle;
- Entrada/saída;
- Manipulação de exceções e erros;
- Abstração procedural;
- Suporte de biblioteca para estruturas de dados.

Suporte de biblioteca para ED

Necessidades

- Vantagens das bibliotecas:

Suporte de biblioteca para ED

Necessidades

- Vantagens das bibliotecas:
  - facilitam o desenvolvimento de aplicações complexas;

Suporte de biblioteca para ED

Necessidades

- Vantagens das bibliotecas:
  - facilitam o desenvolvimento de aplicações complexas;
  - "inventam a roda".

Suporte de biblioteca para ED

Necessidades

- Vantagens das bibliotecas:
  - facilitam o desenvolvimento de aplicações complexas;
  - "inventam a roda".
- *Standard Template Library*:

Suporte de biblioteca para ED

Necessidades

- Vantagens das bibliotecas:
  - facilitam o desenvolvimento de aplicações complexas;
  - "inventam a roda".
- *Standard Template Library*:
  - manipulam estruturas de dados;
  - foi projetada para a programação imperativa;
  - é uma opção para os desenvolvedores.

Potencialidades

Pontos fortes

- Exigência de alto desempenho;

Potencialidades

Pontos fortes

- Exigência de alto desempenho;
- Programação mais próxima ao nível de máquina;

Potencialidades

Pontos fortes

- Exigência de alto desempenho;
- Programação mais próxima ao nível de máquina;
  - e.g. C é tratada como linguagem de máquina universal.

Potencialidades

Pontos fortes

- Exigência de alto desempenho;
- Programação mais próxima ao nível de máquina;
  - e.g. C é tratada como linguagem de máquina universal.
- Restrições de memória ou potência.

Potencialidades

Pontos fortes

- Exigência de alto desempenho;
- Programação mais próxima ao nível de máquina;
  - e.g. C é tratada como linguagem de máquina universal.
- Restrições de memória ou potência.
  - e.g. programação em dispositivos móveis.

Fragilidades

Limitações da Decomposição Funcional

- Muitas aplicações não são bem servidas neste paradigma;

Fragilidades

Limitações da Decomposição Funcional

- Muitas aplicações não são bem servidas neste paradigma;
- Programação de GUIs e dispositivos embarcados;

Fragilidades

Limitações da Decomposição Funcional

- Muitas aplicações não são bem servidas neste paradigma;
- Programação de GUIs e dispositivos embarcados;
- Outra abordagem: troca de mensagens entre objetos;

Fragilidades

Limitações da Decomposição Funcional

- Muitas aplicações não são bem servidas neste paradigma;
- Programação de GUIs e dispositivos embarcados;
- Outra abordagem: troca de mensagens entre objetos;
- Programação de GUIs:
  - coleção de diferentes tipos de objetos ? botões, áreas de texto, imagens, videoclipes e menus desdobráveis ?, cada um comunicando-se com o programa e com o usuário, mandando e recebendo mensagens.

Fragilidades

Limitações da Decomposição Funcional

- Muitas aplicações não são bem servidas neste paradigma;
- Programação de GUIs e dispositivos embarcados;
- Outra abordagem: troca de mensagens entre objetos;
- Programação de GUIs:
  - coleção de diferentes tipos de objetos ? botões, áreas de texto, imagens, videoclipes e menus desdobráveis ?, cada um comunicando-se com o programa e com o usuário, mandando e recebendo mensagens.
  - "inventam a roda".

Fragilidades

Limitações da Decomposição Funcional

- Muitas aplicações não são bem servidas neste paradigma;
- Programação de GUIs e dispositivos embarcados;
- Outra abordagem: troca de mensagens entre objetos;
- Programação de GUIs:
  - coleção de diferentes tipos de objetos ? botões, áreas de texto, imagens, videoclipes e menus desdobráveis ?, cada um comunicando-se com o programa e com o usuário, mandando e recebendo mensagens.
  - "inventam a roda".
- Diálogo entre os *stakeholders*.

Sumário

1 Motivação

2 Paradigma Imperativo

3 Próxima aula...

Próxima aula...

Paradigma Orientado a Objetos

- Um breve histórico;
- Características gerais;
- Potencialidades e fragilidades.

Programação Imperativa

Esdras Lins Bispo Jr.
bispojr@ufg.br

Concurso para Professor de Carreira do Magistério Superior
Bacharelado em Sistemas de Informação
UPE - Campus Caruaru

20 de fevereiro de 2018