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

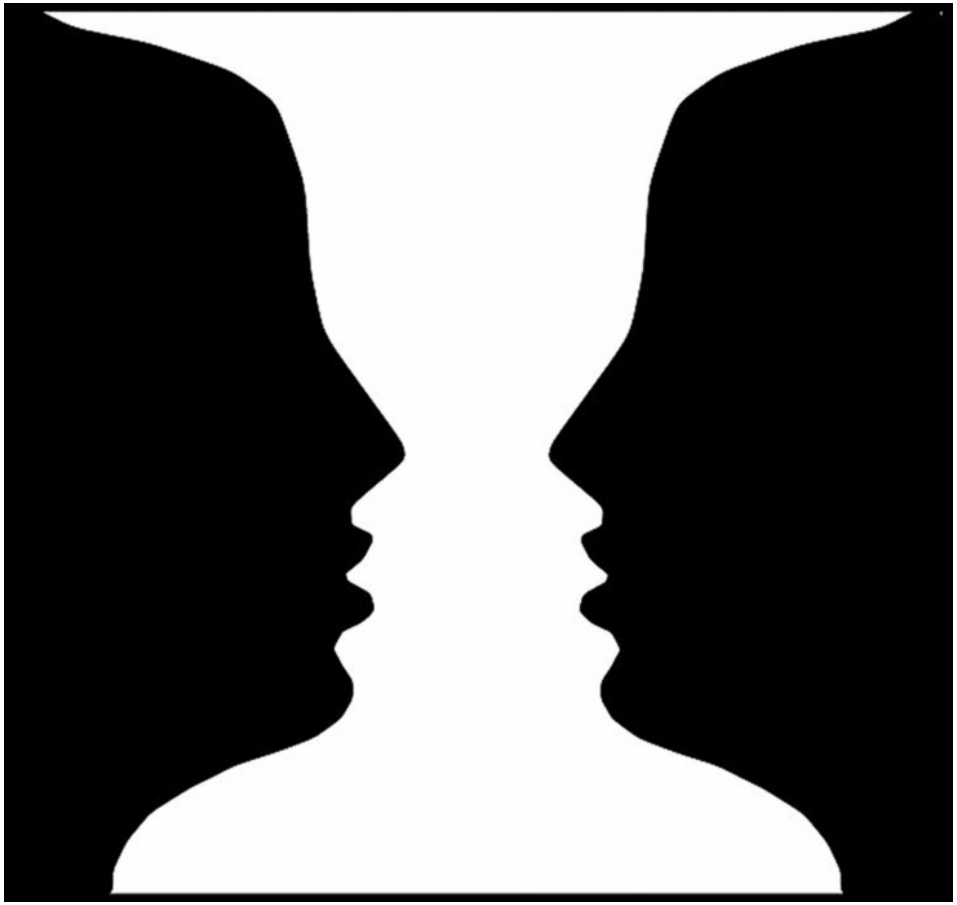
20 de fevereiro de 2018

Plano de Aula

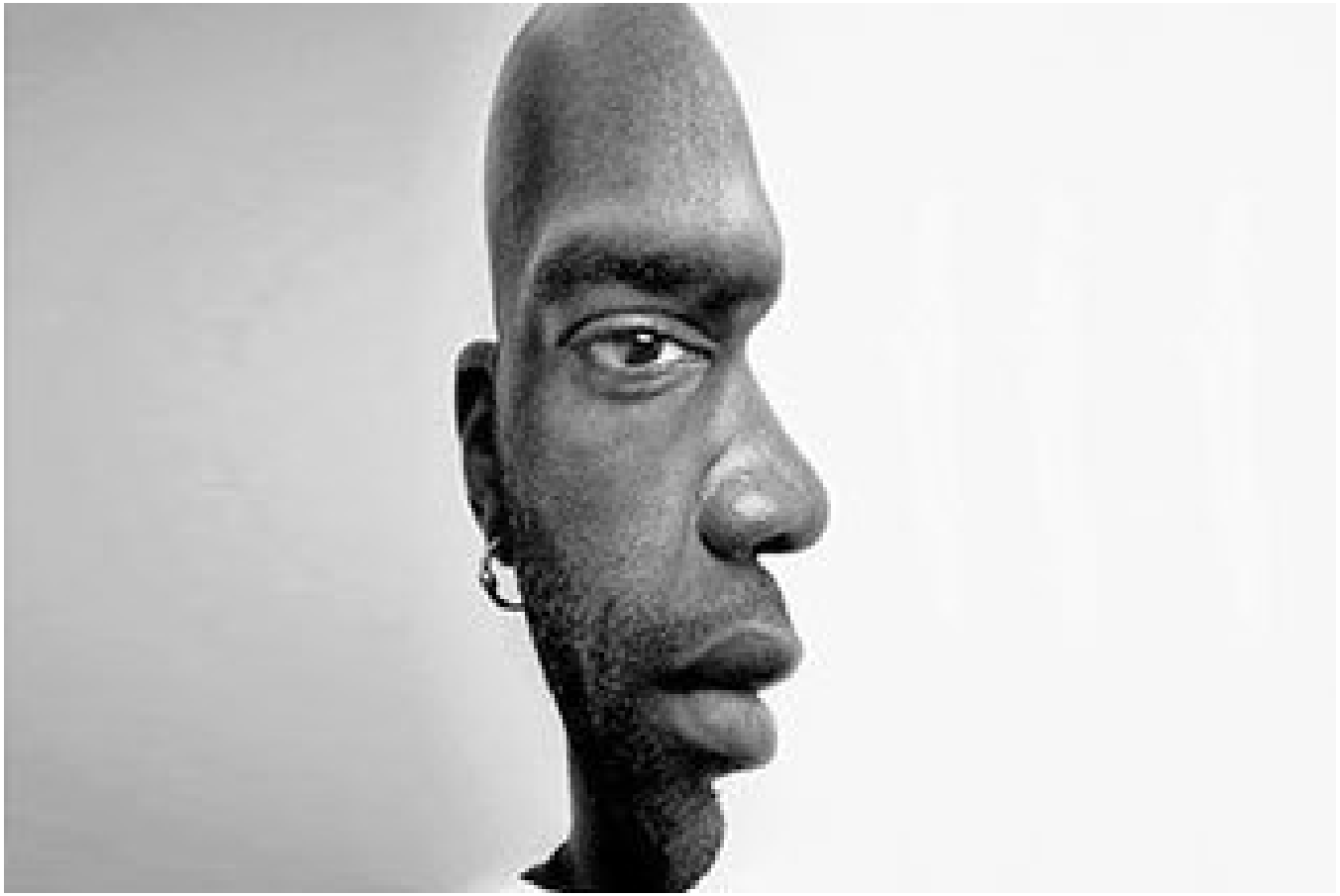
1. Motivação
2. Paradigma Imperativo
   - Breve Histórico do Paradigma
   - Características do Paradigma
   - Potencialidades e Fragilidades (C++)
3. O que veremos na próxima aula?

Sumário

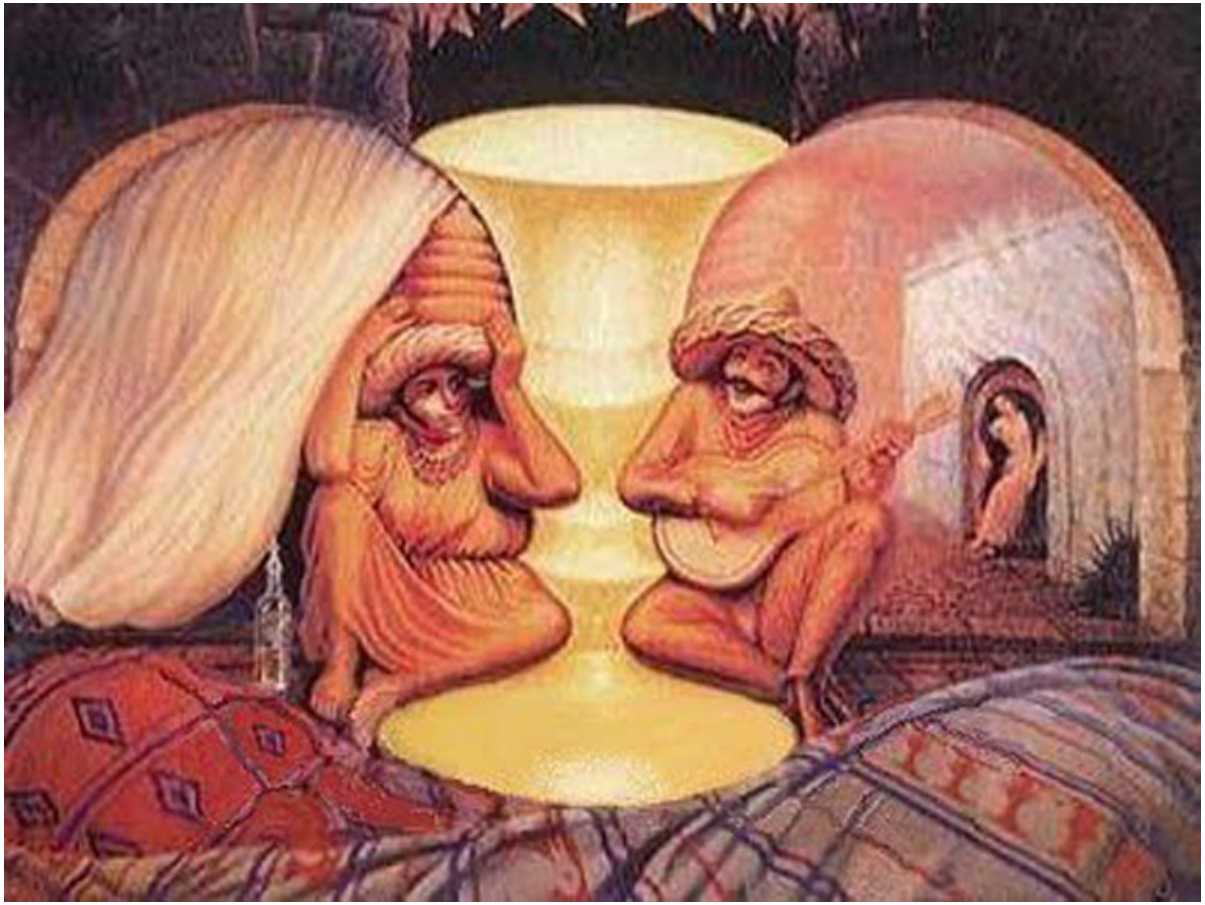
O que você está vendo?



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 (Linguagem de exemplo: C++)

UNIVERSIDADE DE PERNAMBUCO

Sumário

Programa Armazenado

Ideia

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

Quem?

John von Neumann (1940)

Programa Armazenado

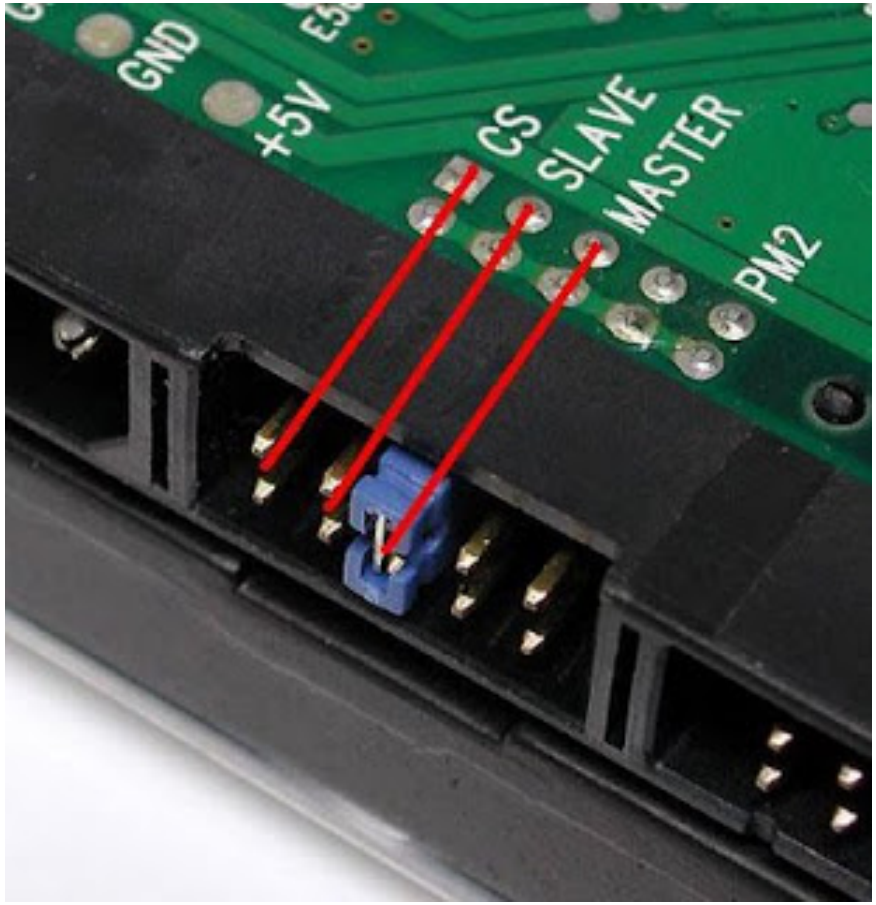
Ideia

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

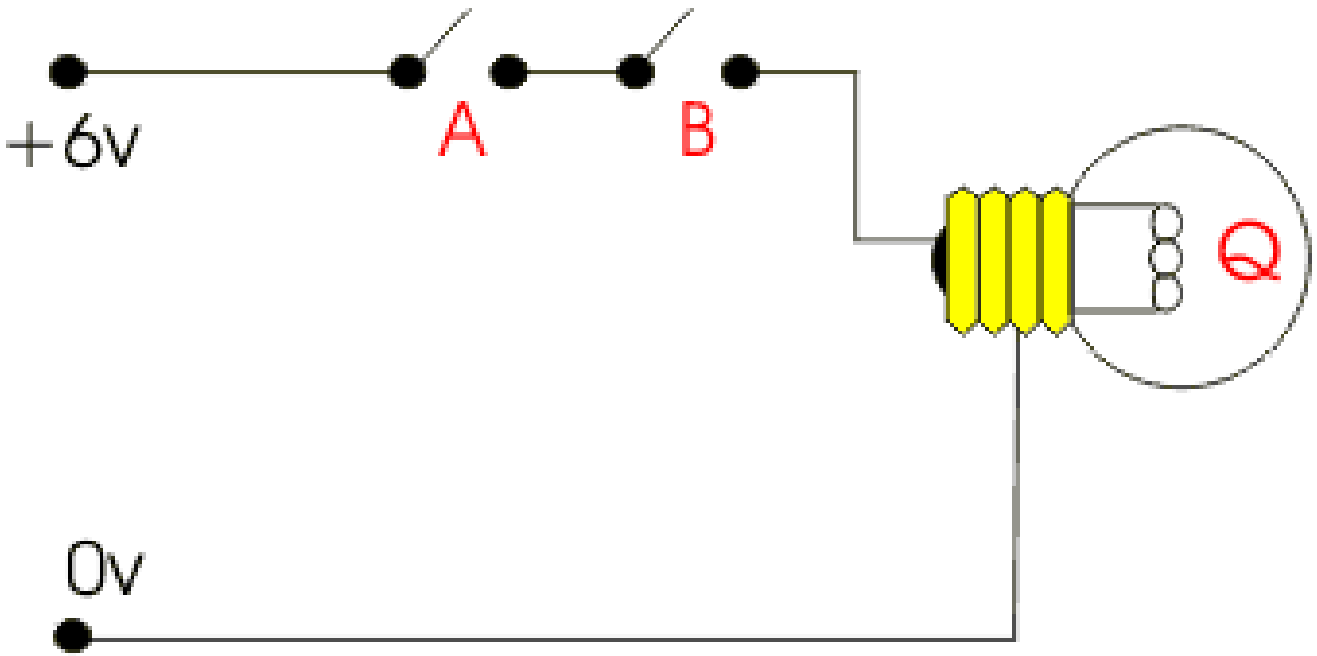
Quem?

Alan Turing (1936)

Um outro tipo de programação

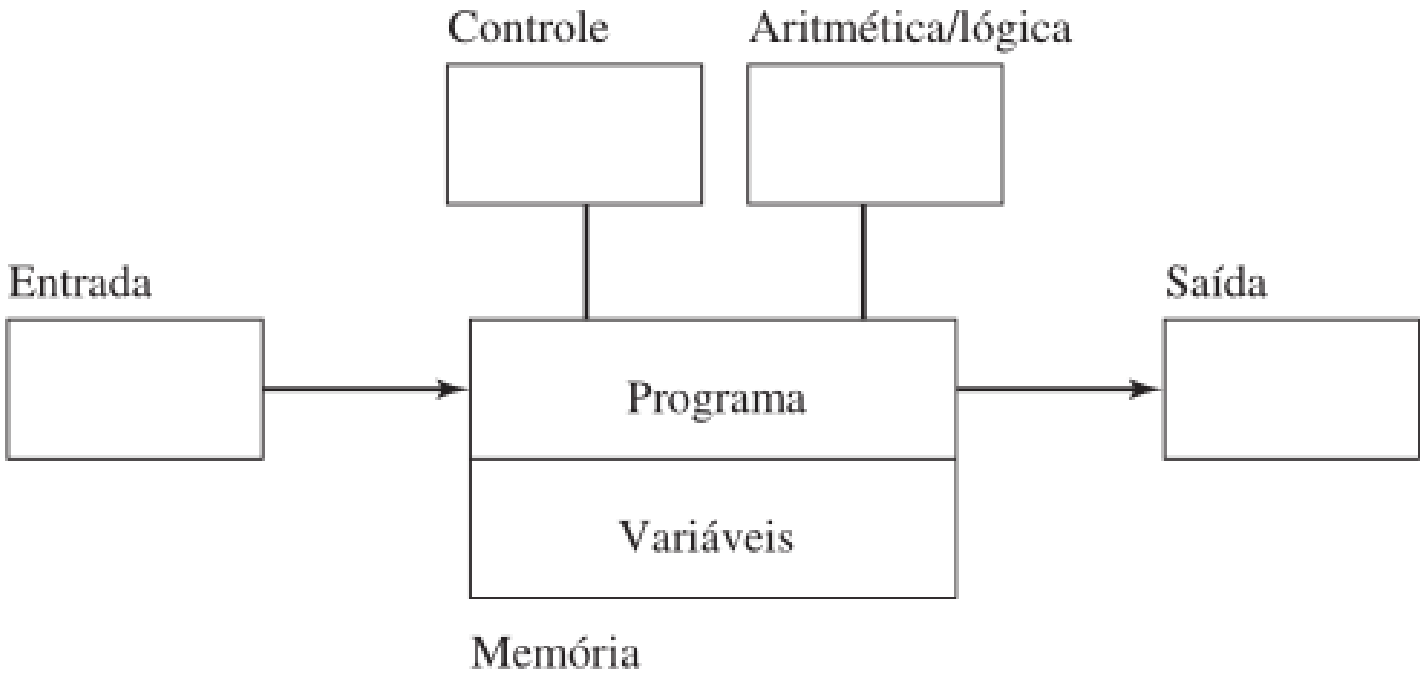


Um outro tipo de programação



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;

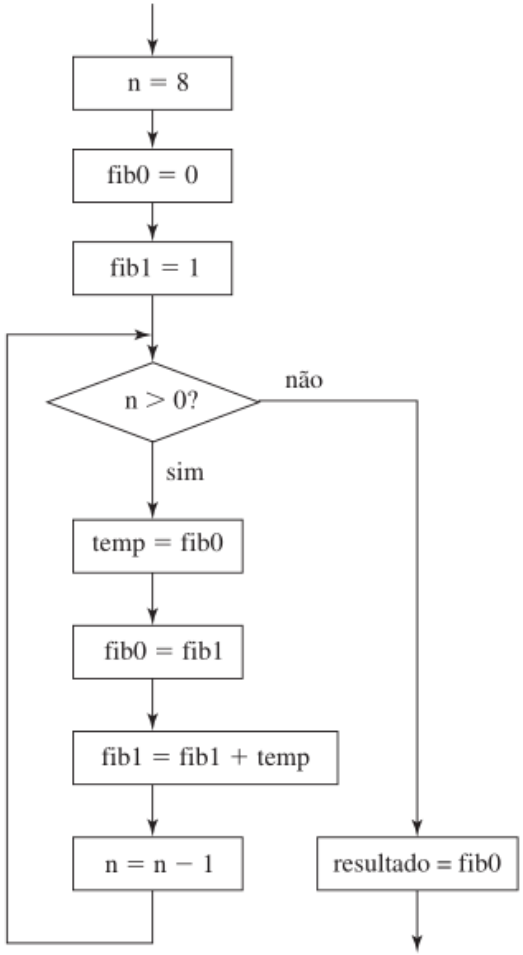
Modelo de Neumann-Eckert

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)

Consequência da Evolução do Paradigma

Características Atuais

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

- Expressões de atribuição;

Consequência da Evolução do Paradigma

Características Atuais

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

- Expressões de atribuição;
- Estruturas de controle;

Consequência da Evolução do Paradigma

Características Atuais

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

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

Expressões de Atribuição

- Fundamental para todas as linguagens imperativas.

```
1  a = b;      //C, Fortran
2  c := d;     //ALGOL
```

Expressões de Atribuição

- Fundamental para todas as linguagens imperativas.

```
1  a = b;      //C, Fortran
2  c := d;     //ALGOL
```

Semântica de Cópia

- expressão avaliada para um valor;
- valor copiado para o destino;

Expressões de Atribuição

```
1  #include <math.h>
2
3  bool a;
4  double b;
5
6  a = (5 > 3) || (2*3 == 4);      // valor de 'a': true
7  b = pow(2.0, 3.0);              // valor de 'b': 8.0
```

Expressões

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

Características do Paradigma

Características

- Expressões de 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.

```
1  a = 5 > b;
2  b = c;
3  c++;
```

Estruturas de controle

Sequência

- Fluxo natural de execução.

```
1  a = 5 > b;
2  b = c;
3  c++;
```

- Comandos de desvios: **return**, **break**, **continue**, e **go to**.

Estruturas de controle

Sequência

- Fluxo natural de execução.

```
1 a = 5 > b;
2 b = c;
3 c++;
```

- Comandos de desvios: **return**, **break**, **continue**, e **go to**.

```
1 #include <iostream>
2
3 rotulo1:
4   int a = 10;
5   a = a - 2;
6   goto rotulo1;
```

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)

Estruturas de controle

Condicional

- Seleciona caminhos alternativos em execução

```
1  if(a > b){
2    a++;
3  }
4
5  switch(a){
6    case 5:
7      a--;
8      break;
9    case 6:
10     a = 3;
11     break;
12   default: a = 1;
13 }
```

Estruturas de controle

Laço

- Duas variações: teste no início ou no fim.

```
1  while(a > b){
2     a++;
3  }
4
5  do{
6     a++;
7  }while(a > b)
```

Estruturas de controle

Laço

- O comando **for** é uma variação bastante utilizada;

```
1  int i, a;
2
3  a=0;
4  for(i=0; i<5; i++){
5     if(i % 4 == 0)
6        continue;
7     a = a + i;
8  }
9  while(a > 0){
10    if(a == 2)
11       break;
12    a--;
13 }
```

Características do Paradigma

Características

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

Entrada e Saída

Deveres

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

Entrada e Saída

Deveres

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

```
1  #include <iostream>
2  using namespace std;
3
4  int a;
5  int b;
6
7  cout << "Digite o valor de 'a': ";
8  cin >> a;
9  cout << "Digite o valor de 'b': ";
10 cin >> b;
11
12 cout << "a + b = " << a + b;
```

Entrada e Saída

Nomenclatura

- As fontes e os destinos são denominados arquivos:

Entrada e Saída

Nomenclatura

- As fontes e os destinos são denominados arquivos:
  - Teclado;

Entrada e Saída

Nomenclatura

- As fontes e os destinos são denominados arquivos:
  - Teclado;
  - Monitor;

Entrada e Saída

Nomenclatura

- As fontes e os destinos são denominados arquivos:
  - Teclado;
  - Monitor;
  - Disco rígido;

Entrada e Saída

Nomenclatura

- As fontes e os destinos são denominados arquivos:
  - Teclado;
  - Monitor;
  - Disco rígido;
  - *Pen-drive*.

Entrada e Saída

Nomenclatura

- 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 de 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

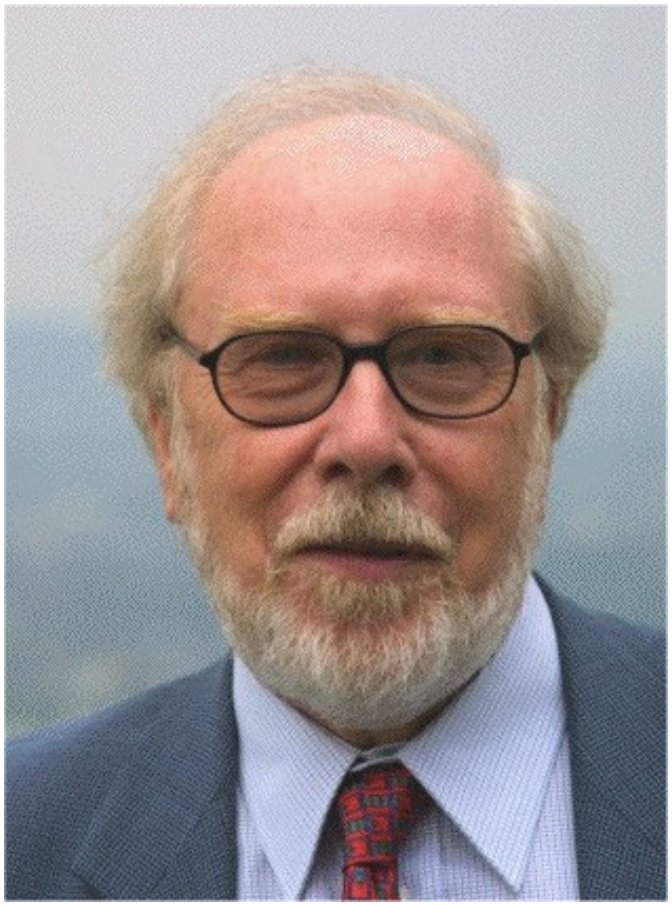
Uma aplicação é robusta quando ela continua a operar sob todas as situações de erro presumíveis.

Características do Paradigma

Características

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

Refinamento de Passos



Ideia

A construção de programas consiste de uma sequência de refinamento de passos.

Quem?

Niklaus Wirth (1973)

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.

```
1  #include <algorithm>     // std::sort
2  #include <vector>        // std::vector
3
4  int colecao[] = {32,71,12,45,26,80,53,33};
5  std::vector<int> meuVetor (colecao, colecao+8);
6
7  std::sort (meuVetor.begin(), meuVetor.begin()+4);
           //(12 32 45 71)26 80 53 33
```

Características do Paradigma

Características

- Expressões de 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;
- Exemplo: Programação de GUIs;

Fragilidades

Limitações da Decomposição Funcional

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

Fragilidades

Limitações da Decomposição Funcional

- Muitas aplicações não são bem servidas neste paradigma;
- Exemplo: Programação de GUIs;
  - Outra abordagem: troca de mensagens entre objetos;
  - Coleção de Objetos: botões, áreas de texto, imagens, videoclipes, menus desdobráveis.

Fragilidades

Limitações da Decomposição Funcional

- Muitas aplicações não são bem servidas neste paradigma;
- Exemplo: Programação de GUIs;
  - Outra abordagem: troca de mensagens entre objetos;
  - Coleção de Objetos: botões, áreas de texto, imagens, videoclipes, menus desdobráveis.
- Diálogo entre os *stakeholders*;

Fragilidades

Limitações da Decomposição Funcional

- Muitas aplicações não são bem servidas neste paradigma;
- Exemplo: Programação de GUIs;
  - Outra abordagem: troca de mensagens entre objetos;
  - Coleção de Objetos: botões, áreas de texto, imagens, videoclipes, menus desdobráveis.
- Diálogo entre os *stakeholders*;
- Separação de interesses pode ser difícil.

Sumário

O que veremos na próxima aula?

Paradigma Orientado a Objetos

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

O que veremos na próxima aula?

Paradigma Orientado a Objetos

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

Avisos

- Exercício de Aquecimento 02:
  Disponível em `tinyurl.com/aquecimento-poo`
- Slides de hoje disponíveis no repositório:
  Disponível em `github.com/bispojr/upe-prova-didatica`

UPE UNIVERSIDADE DE PERNAMBUCO

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