TCC 00308: Programação de Computadores I 2017.1

Introdução

Processo de resolução de problemas

- Definição dos requisitos do **problema** (fazer o programa certo)
  - Entradas
  - Cálculos
  - Casos especiais
  - Saídas
- Desenvolvimento do algoritmo da **solução** (fazer certo o programa)
  - Português estruturado
  - Pseudocódigo
  - Fluxograma
- **Codificação** do programa
  - Python
- **Teste** do programa
  - Instrução com erro de grafia (defeito na codificação)
  - Resultado errado (defeito no algoritmo)

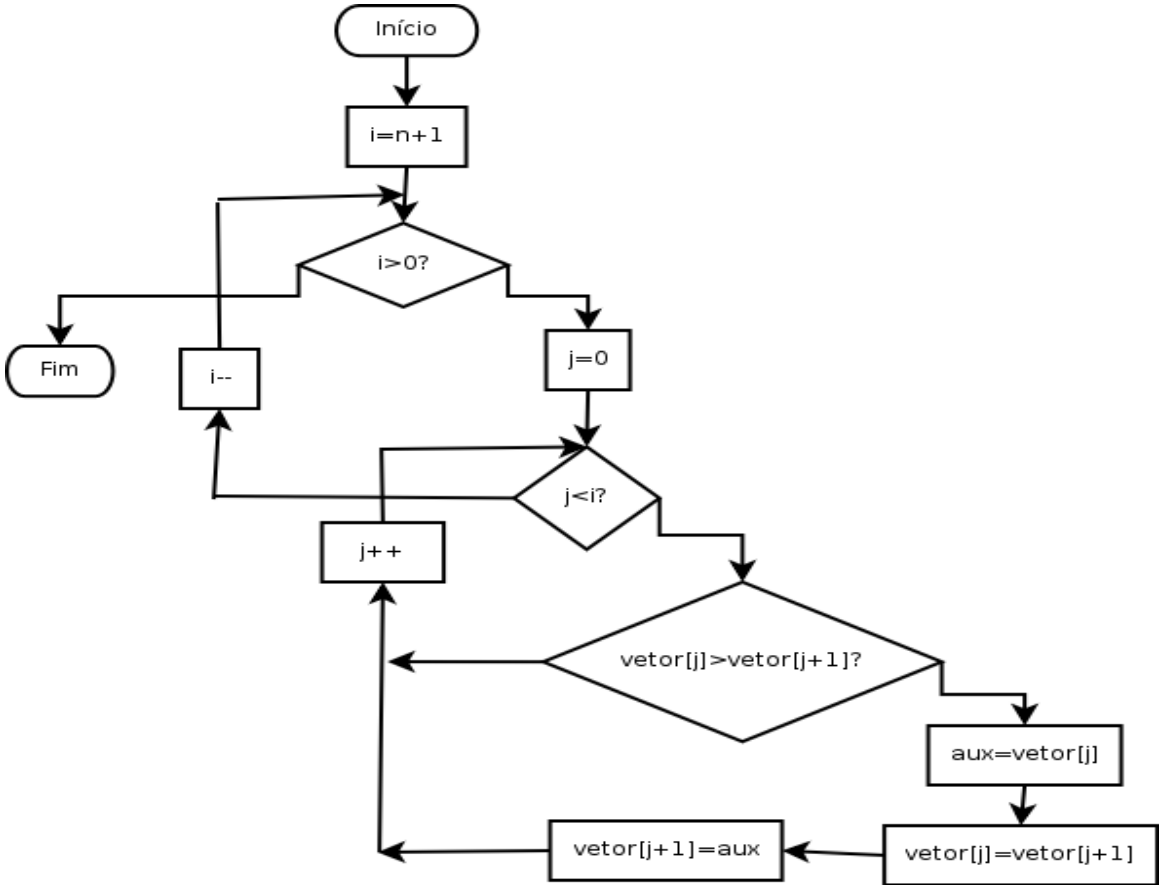
Passo 1: Requisitos

- Qual é o problema a ser resolvido?

Passo 2: Algoritmo

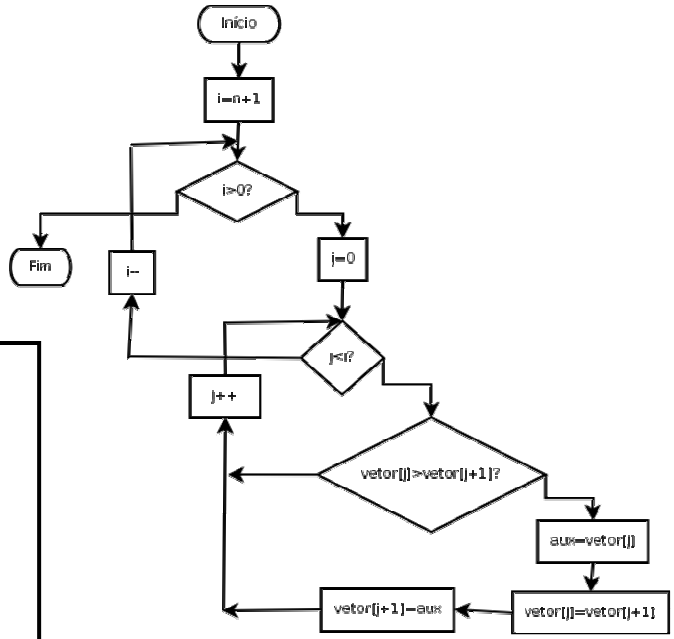
- Conjunto de **ações** para a **resolução** de um problema em um **número finito** de passos
- Parte mais complexa da programação
- Somente iniciar a programação quando
  - Souber qual problema deve ser resolvido
  - Souber como resolver o problema

Passo 2: Algoritmo



Passo 2: Algoritmo

- Independente de linguagem de programação
- Pode ser implementado em diferentes linguagens



C++

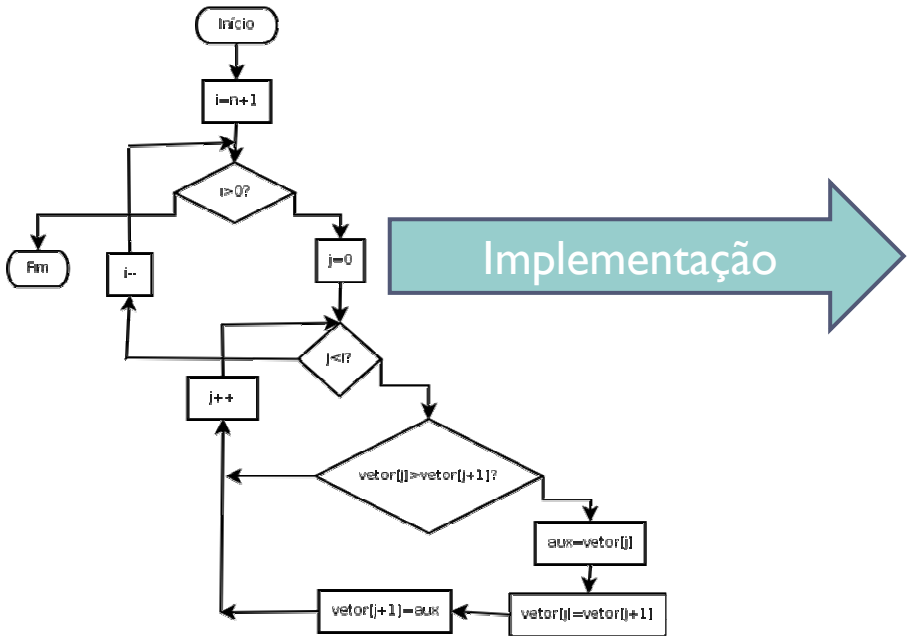
```
#include <algorithm>
using namespace std;

void bubblesort(int a[], int n)
{
 for(int j=0; j<n; j++){
  for(int i=0; i<n-1; i++){
   if(a[i+1] < a[i])
    swap(a[i+1], a[i]);
  }
 }
}
```

Matlab

```
for(i = 1:n-1)
    for(j = 1:n-i)
        if(x(j) > x(j + 1))
            aux = x(j);
            x(j) = x(j + 1);
            x(j + 1) = aux;
        end
    end
end
```

Passo 3: Codificação

- A partir do algoritmo, traduzir (implementar) para a linguagem desejada
  - No nosso caso, Python



Implementação

Python

```
def bubble (vetor):
    houvetroca = True
    while (houvetroca):
        houvetroca = False
        for i in range(len(vetor) - 1):
            if (vetor[i] > vetor[i+1]):
                aux = vetor[i+1]
                vetor[i+1] = vetor[i]
                vetor[i] = aux
                houvetroca = True
    return v
```

Por que não executar diretamente o algoritmo no computador?

- Algoritmo é escrito em linguagem natural
- Linguagem natural é **muito complexa e pouco precisa**
- É necessário usar uma linguagem mais simples e precisa, que o computador compreenda

Teste

- O trabalho não termina com o código
- Todo código pode ter defeito (*bug*)
- Testar o código é fundamental!

Teste (Tipos de Erros)

- Erro de sintaxe
  - Falha na tradução do algoritmo para Python
  - O computador vai detectar e dar dicas
  - Mais fáceis de corrigir
- Erro de lógica
  - Resultados diferentes do esperado
  - Erro de projeto do algoritmo
  - Mais difíceis de corrigir

Exercício

- Escreva um algoritmo que consiga colocar em ordem as cartas de um naipe do baralho

Algoritmos clássicos: *Insertion Sort*

Pegue a pilha de cartas desordenada
Enquanto existir carta na mão faça {
  Pegue a primeira carta da mão
  Se não tem carta sobre a mesa então
    Coloque-a sobre a mesa
  Caso contrário
    Coloque-a na posição correta da pilha da mesa
}

Algoritmos clássicos: *Selection Sort*

Pegue a pilha de cartas desordenada
Enquanto existir carta na mão faça {
  Pegue a maior carta da mão
  Se não tem carta sobre a mesa então
    Coloque-a sobre a mesa
  Caso contrário
    Coloque-a no topo da pilha da mesa
}

Algoritmos clássicos: *Bubble Sort*

Pegue a pilha de cartas desordenada

Enquanto as cartas não estiverem ordenadas faça {

  Para cada carta do baralho faça {

    Se a carta consecutiva for menor que a carta atual

      Inverta a posição destas cartas

  }

}

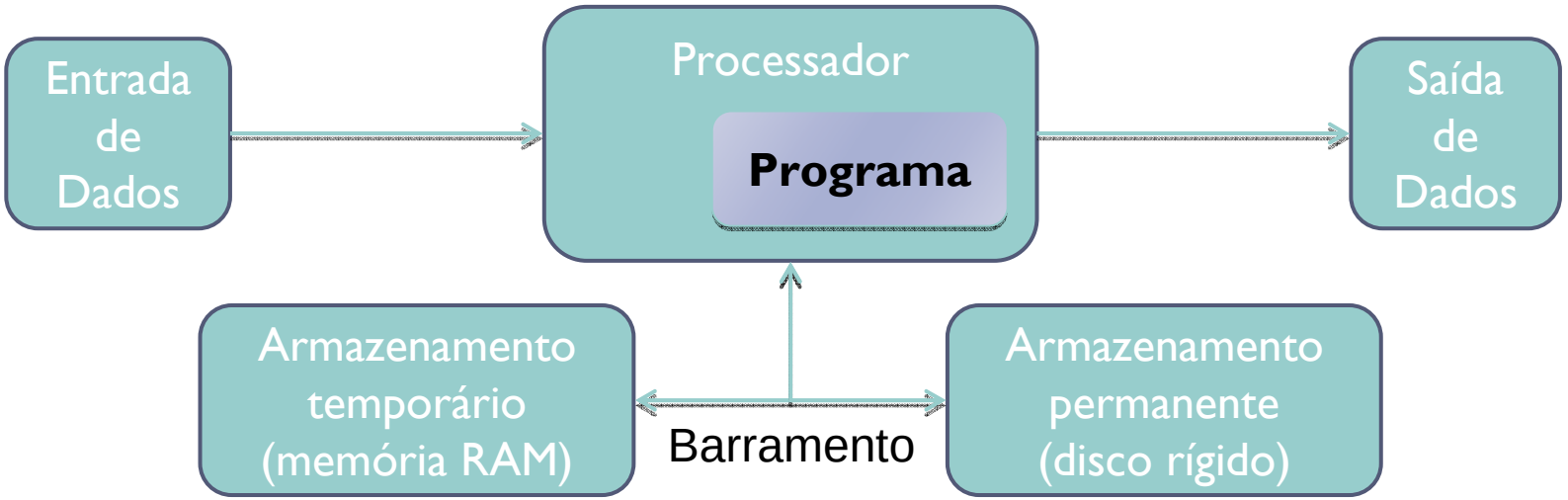
E se tivermos que pedir para o computador resolver?

Pilha de cartas desordenada

Sequência de passos necessários para resolver o problema

Pilha de cartas ordenada

Arquitetura de um computador



Entrada	Saída	Armazenamento
Teclado	Vídeo	Memória
Mouse	Impressoras	Discos rígidos
Scanner	Auto-Falante	CD/DVD
Webcam		Pen drive

Algoritmo para somar dois números

```
Leia um valor do teclado
Escreva esse valor no endereço de memória  A
Leia um valor do teclado
Escreva esse valor no endereço de memória B
Some o valor armazenado no endereço de
  memória A com o valor armazenado no
  endereço de memória B
Escreva o resultado no endereço de memória
  SOMA
Leia o valor do endereço de memória SOMA
Escreva no monitor de vídeo
```

Algoritmo para indicar se um número é maior que outro

```
Leia um valor do teclado
Escreva esse valor no endereço de memória A
Leia um valor do teclado
Escreva esse valor no endereço de memória B
Avalie A > B
Escreva o resultado no endereço de memória R
Leia o valor armazenado no endereço de
  memória R
Escreva o valor do armazenado no endereço de
  memória R no monitor de vídeo
```

Pseudocódigo

- Forma genérica, mas sucinta, para escrever um algoritmo
- Fácil para um humano entender
- Fácil de ser codificada
- Voltando aos exemplos anteriores

```
Leia A
Leia B
SOMA ← A + B
Escreva SOMA
```

```
Leia A
Leia B
R ← A > B
Escreva R
```

Exercício

- Em relação ao pseudocódigo a seguir

```
Leia Valor
Leia Quantidade
Total ← Valor * Quantidade
Escreva Total
```

- Quais são os dados de entrada e saída?
- Quais linhas são somente de processamento?

Exercício

- Qual é a funcionalidade desse algoritmo? Execute para os valores 25 e 7.

```
Leia A
Leia B
C ← 0
Enquanto A >= B faça {
  A ← A – B
  C ← C + 1
}
Escreva C
Escreva A
```

Exercício

- Escreva um algoritmo em pseudocódigo para
  a) Somar três números
  b) Calcular a média de um aluno numa disciplina, sendo
     Média = (Provas + 3 x Trabalho + Participação) / 10
     Provas = 3 x Prova1 + 3 x Prova2
  c) Calcular o peso ideal de uma pessoa, assumindo
     Homem: Peso = (72,7 * Altura) - 58
     Mulher: Peso = (62,1 * Altura) - 44,7

Exercício

- Escreva um algoritmo para separar o líquido de três garrafas com formatos diferentes em duas quantidades iguais, onde
  - Uma garrafa está cheia até a boca, com 8 litros
  - Uma está vazia, com capacidade de 5 litros
  - Uma está vazia, com capacidade de 3 litros

Referências

- Slides baseados no curso de Programação de Computadores I da Prof. Vanessa Braganholo
- Alguns exercícios extraídos do livro Furlan, M., Gomes, M., Soares, M., Concilio, R., 2005, “Algoritmos e Lógica de Programação”, Editora Thomson.