Departamento de Sistemas de Computação
Universidade de São Paulo

# Algoritmo e Pseudo-código

**Responsável**

Prof. Seiji Isotani (sisotani@icmc.usp.br)

## Objetivos do Curso

Desenvolver o

# Pensamento Computacional

# Pensamento Computacional

Técnicas específicas de pensamento computacional incluem:

- **Decomposição do problema**
- Reconhecimento de padrões
- Generalização de padrões para definir abstrações ou modelos
- Projeto de algoritmos
- Análise de dados e visualização

Suponha que você tenha um amigo estrangeiro que quer fazer o bolo de chocolate que você preparou no dia anterior? Como você o ajudaria?

## Decomposição de problemas

Seu amigo estrangeiro se ofereceu a ir ao supermercado para fazer as compras. Como você explicaria o procedimento para passar no caixa?

## Decomposição de problemas

Você tem dois programas de computador que fazem a mesma coisa e precisa indicar ao gestor o melhor dentre eles, o que você faria?

# Pensamento Computacional

Técnicas específicas de pensamento computacional incluem:

- Decomposição do problema
- **Reconhecimento de padrões**
- Generalização de padrões para definir abstrações ou modelos
- Projeto de algoritmos
- Análise de dados e visualização

- Para resolver os problemas apresentados você precisa pelo menos:
  - Indicar quais são as entradas esperadas
  - Indicar quais são as saídas produzidas
  - Um conjunto de passos não ambíguos para transformar as entradas nas saídas.

# Isso é um algoritmo

# Algoritmo

Um algoritmo é uma conjunto de atividades que podem ser executadas passo a passo para resolver problemas

Outras definições:
- http://en.wikipedia.org/wiki/Algorithm
- Robert Sedgewick – Algorithms in C, 3rd Ed, Addison Wesley

# Mas peraí professor ....

- Um algoritmo então é um programa de computador?
  - Não. Um programa de computador é uma das possíveis representações de um algoritmo.
- Um algoritmo então é um processo?
  - Não, processo é a atividade de executar sequência bem definidas e, consequentemente, também é a atividade de executar um algoritmo.

# Algoritmos estão em toda a parte

# Não adianta fugir

- Imagine que você queira dar escalabilidade para realizar atividades em sua escola
  - Um algoritmo pode analisar as atividades dos alunos e recomendar material de reforço
  - Um algoritmo pode avaliar as provas dos alunos
  - Um algoritmo pode identificar a maneira mais eficiente de dar apoio ao professor para criar material didático
- Em Educação a Distância (EAD):
  - Existem algoritmos para personalizar a aprendizagem colocando as atividades no ar de maneira diferenciada
- Em computação tudo vira algoritmos

Se não é possível fugir, então junte-se a ele ☺.

**Seja um criador de algoritmo de primeira linha!**

# Como?

# Como criar algoritmos?

- Para se tornar um bom **cozinheiro**, você não memoriza livros e livros de receitas.
- Você **estuda** receitas existentes.
- Você **pratica** fazê-los.
- Você **experimenta** com suas próprias variações.
- Até que possa **inventar** suas próprias receitas.

# Como criar algoritmos?

- Para se tornar um bom **profissional em exatas**, você não deve memorizar livros e livros de algoritmos.
- Você **estuda** algoritmos famosos.
- Você **pratica** implementá-los e executá-los.
- Depois passa a reconhecer padrões mais gerais, e a **experimentar** implementações com suas próprias variações.
- Até que, finalmente, você irá **inventar** seus próprios algoritmos.

# OK. Agora eu sei o que é um algoritmo!

Mas como “escrever” ou representar um algoritmo?

- **Descrição Narrativa**
- **Fluxograma**
- **Pseudocódigo**

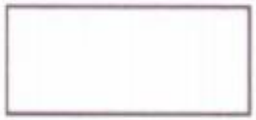
# E como representar um algoritmo?

- **Descrição narrativa:**
  - consiste em analisar o enunciado do problema e escrever, utilizando linguagem natural, os passos a serem seguidos para sua resolução (receita de bolo).
  - **Ponto positivo**: Não é necessário aprender novos conceitos, pois a língua natural já é bem conhecida.
  - **Ponto negativo**: A língua natural abre espaço para várias interpretações, dificultando a transcrição desse algoritmo para programa

# E como representar um algoritmo?

- **Fluxograma:**
  - consiste em analisar o enunciado do problema e escrever, utilizando símbolos gráficos, os passos a serem seguidos para sua resolução
  - **Ponto positivo**: O entendimento de elementos gráficos é mais simples que o entendimento de textos.
  - **Ponto negativo**: Os fluxogramas devem ser entendidos e o algoritmo resultante não é detalhado. Isso dificulta sua transcrição para um programa

**TABELA 1.1:** Conjunto de símbolos utilizados no fluxograma.

| | |
|---|---|
|  | Símbolo utilizado para indicar o início e o fim do algoritmo. |
|  | Permite indicar o sentido do fluxo de dados. Serve exclusivamente para conectar os símbolos ou blocos existentes. |
|  | Símbolo utilizado para indicar cálculos e atribuições de valores. |
|  | Símbolo utilizado para representar a entrada de dados. |
|  | Símbolo utilizado para representar a saída de dados. |
|  | Símbolo que indica que deve ser tomada uma decisão, indicando a possibilidade de desvios. |

Fonte:
http://wiki.icmc.usp.br/index.php/Scc-101(2011101) – Aula 2

# E como representar um algoritmo?

- **Pseudocódigo:**
  - consiste em analisar o enunciado do problema e escrever, por meio de regras predefinidas, os passos a serem seguidos para sua resolução
  - **Ponto positivo**: Representação clara sem as especificações de linguagem de programação. A passagem do algoritmo para qualquer linguagem de programação é mais simples.
  - **Ponto negativo**: As regras do pseudocódigo devem ser aprendidas

## Exemplos

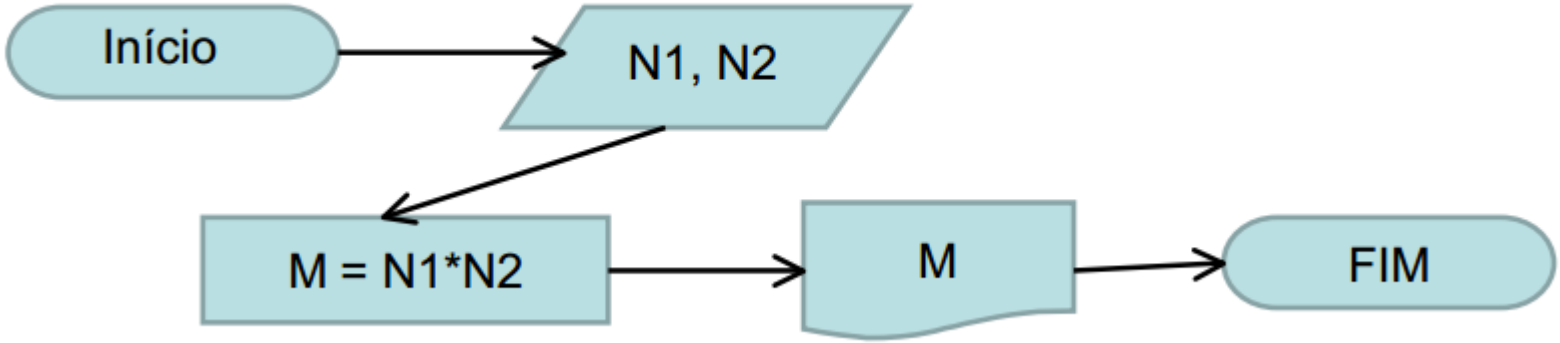
Crie um algoritmo para exibir o resultado da multiplicação de dois números utilizando as três formas de representação

# Exemplo – Descrição Narrativa

- Descrição narrativa
  - PASSO 1 – Receber os dois números que serão multiplicados.
  - PASSO 2 – Multiplicar os números.
  - PASSO 3 – Mostrar o resultado obtido da multiplicação.

# Exemplo - Fluxograma

**Algoritmo em fluxograma**



# Exemplo - Pseudocódigo

Existem várias formas de escrever o pseudocódigo.
Exemplos:

```
Multplicacao (n1, n2)
        m ←n1*n2
        RETORNE m
FIM
```

```
ALGORITMO
        DECLARE n1, n2, m
        LEIA n1, n2
        m  ← n1*n2
        ESCREVA  m
FIM
```

# Exemplo código C

**Código C**

```
#include<stdio.h>
void main() {
    int n1, n2, m;
    scanf("%d %d", &n1, &n2);
    m = n1*n2;
    printf("\n %d", m);
}
```

```
ALGORITMO
    DECLARE n1, n2, m
    LEIA n1, n2
    m ← n1*n2
    ESCREVA  m
FIM
```

# Representação de Algoritmos

- Leitura/Escrita
  - Leitura de dados: LEIA
  - Escrita de dados: IMPRIMA
- Estrutura Condicional
  - Simples: SE-ENTAO
  - Composta: SE-ENTAO-SENAO
- Estrutura de Repetição
  - PARA
  - ENQUANTO
  - REPITA

Durante as aulas utilizaremos pseudocódigo (e scratch) como forma de representar um algoritmo

# Representação de Algoritmos

- Declaração de variáveis:
  - DECLARE
- Leitura/Escrita
  - Leitura de dados: LEIA
  - Escrita de dados: IMPRIMA
- **Estrutura Condicional**
  - **Simples: SE-ENTAO**
  - **Composta: SE-ENTAO-SENAO**
- Estrutura de Repetição
  - PARA
  - ENQUANTO
  - REPITA

# Estrutura Condicional Simples

**SE** <Decisão> **ENTÃO**

Instrução 1

....

Instrução N

# Estrutura Condicional Composta

**SE** <Decisão> **ENTÃO**

Instrução 1

Instrução N

**SENÃO**

Instrução 1

Instrução N

Dado dois inteiros crie um algoritmo para retornar o maior deles

# Estrutura Condicional Composta

**Entrada:** inteiros i e j

**Saída: um** inteiro, maior valor entre i e j

**SE** i < j **ENTÃO**

    IMPRIMA j

**SENÃO**

    IMPRIMA i

Dado três inteiros crie um algoritmo para retornar o menor deles

## Tarefa para Sábado

Dado três inteiros crie um algoritmo para imprimi-los em ordem crescente

Atividade deve ser feita em **duplas**.
Entregue o arquivo do scratch