

Linguagens de Programação

Prof. Miguel Elias Mitre Campista

<http://www.gta.ufrj.br/~miguel>

Parte I

Conceitos Básicos de Programação

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

O que é um Problema?

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

O que é um Problema?

- É uma questão ou um obstáculo para a obtenção de um resultado desejado
 - Um problema existe quando:
 - A realidade difere do que se deseja
 - Não há aquilo que se deseja
- A resolução de um problema é um processo dividido em três etapas:
 - Descobrimiento do problema
 - Delineamento do problema
 - Solução propriamente dita do problema

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Etapas de Resolução de um Problema

- Descobrimiento do problema
 - Requer percepção daquilo que está faltando
- Delineamento do problema
 - Encontrar uma abstração para o problema que possibilite que ele seja solucionado
 - Definir o que é relevante para o problema
- Solução do problema
 - Encontrar um modelo eficiente que consiga transformar uma dada entrada na saída desejada

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Problema Computacional

- Problemas cuja solução envolve o uso de computadores
 - Solução encontrada de forma mais rápida
 - Entretanto,
 - A solução do problema, a entrada e a saída dos dados devem ser modeladas de maneira a serem compreendidas por um computador

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Problema Computacional

- Problemas cuja solução envolve o uso de computadores
 - Solução encontrada de forma mais rápida
 - Entretanto,
 - A solução do problema, a entrada e a saída dos dados devem ser modeladas de maneira a serem compreendidas por um computador



A solução é realizada através de um **ALGORITMO** e a entrada e saída de dados devem ser expressas através de uma **ESTRUTURA DE DADOS**

Problema Computacional

- Tanto o **algoritmo** quanto a **estrutura de dados** escolhidos dependem:
 - Do grau de abstração do problema
 - Das ferramentas computacionais existentes
 - Do modelo empregado para a solução

Problema Computacional

- Tanto o **algoritmo** quanto a **estrutura de dados** escolhidos dependem:
 - Do grau de abstração do problema
 - Das ferramentas computacionais existentes
 - Do modelo empregado para a solução



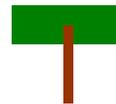
A solução pode ser difícil e pode não ser única...

Abstração X Realidade

- Quanto maior a abstração...
 - Menor a quantidade de características consideradas
 - Maior é a facilidade de obtenção de resultados
 - **Simplificação do problema**
- Entretanto, quanto maior a abstração...
 - Mais distante da realidade se torna o problema
 - A solução ainda é representativa???



X



O que é um Algoritmo?

O que é um Algoritmo?

- Um algoritmo também é uma abstração e pode ser definido como:
 - Um **procedimento bem definido computacionalmente** que recebe uma entrada e produz uma saída
 - **Entrada** pode ser um valor ou um conjunto de valores
 - **Saída** pode ser um valor ou um conjunto de valores
 - **Ambas definidas de maneira estruturada**
- O procedimento executado por um algoritmo pode ser entendido como:
 - Uma **sequência de passos computacionais**

O que é um Algoritmo?

- Uma analogia a um algoritmo poderia ser uma função algébrica



Uma função algébrica:
 Ex.: $f(x): \mathbb{R}^n \rightarrow \mathbb{R}^n$
 $f(x)$ define operações algébricas do tipo soma, subtração, diferenciação etc.

O que é um Algoritmo?

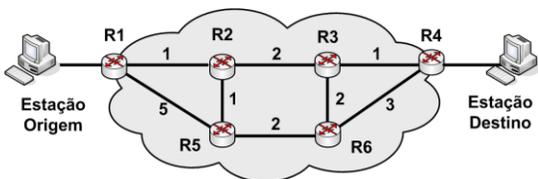
- Uma analogia a um algoritmo poderia ser uma função algébrica



Um algoritmo:
 Ex.: $f(x): A \rightarrow B$
 - A e B são estruturas de dados, ou seja, elementos que podem ser definidos computacionalmente
 - $f(x)$ é uma sequência de passos computacionais

Exemplo de Problema Resolvido por Algoritmo

- Como ligar a estação de origem à estação de destino?
 - Na Internet, busca-se sempre ligar duas estações através do menor caminho

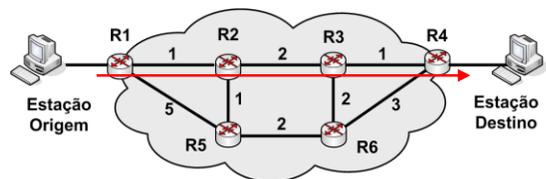


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo de Problema Resolvido por Algoritmo

- Como ligar a estação de origem à estação de destino?
 - Na Internet, busca-se sempre ligar duas estações através do menor caminho

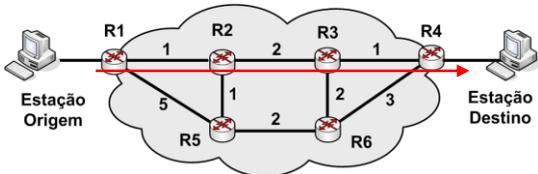


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo de Problema Resolvido por Algoritmo

- Entrada do problema: Rede (conjunto de vértices + conjunto de enlaces)
- Saída do problema: Menor caminho entre as estações de origem e destino.
 - Caminho $\{R1, R2, R3, R4\}$, Custo = 4

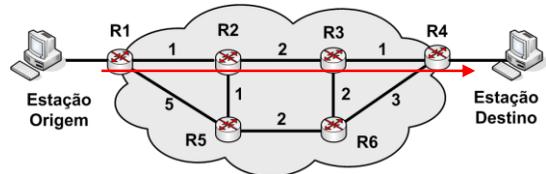


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo de Problema Resolvido por Algoritmo

- Como seria resolvido este problema através de uma função algébrica?

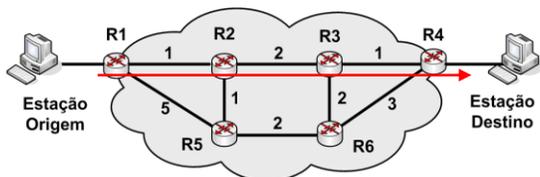


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo de Problema Resolvido por Algoritmo

- Como seria resolvido este problema através de uma função algébrica?
 - Nem sempre pode ser resolvido de maneira simples!
 - **Nem sempre há como definir algebricamente a entrada, nem a saída e nem tampouco a função $f(x)$**

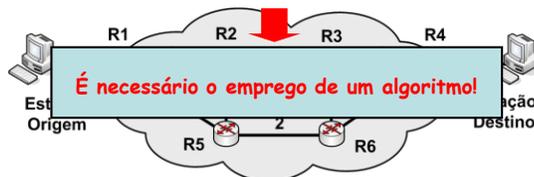


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo de Problema Resolvido por Algoritmo

- Como seria resolvido este problema através de uma função algébrica?
 - Nem sempre pode ser resolvido de maneira simples!
 - **Nem sempre há como definir algebricamente a entrada, nem a saída e nem tampouco a função $f(x)$**



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Instância de um Problema

- Em geral, uma **instância** de um problema consiste de uma entrada para o problema
 - A entrada deve estar de acordo com as restrições impostas pelo problema
 - **No exemplo anterior, a entrada deve ser uma rede que é representada por um grafo que é composto por um conjunto de vértices, enlaces e por uma função que define os pesos dos enlaces**

Instância do problema da busca do menor caminho $G(V, E)$:
 $V = \{R1, R2, R3, R4, R5\}$
 $E = \{(R1, R2), (R1, R5), (R2, R3), (R2, R5), (R5, R6), (R3, R4), (R3, R6), (R4, R6)\}$
 Peso $w(e)$, onde $e \in E$

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Correção de um Algoritmo

- Para ser correto, um algoritmo deve:
 - Terminar a sua execução com a saída correta para uma instância qualquer do problema
 - Somente nesse caso, o algoritmo é dito **correto**
- Um algoritmo **não correto**
 - Pode nunca terminar a sua execução para algumas instâncias do problema
 - **Algoritmos com complexidade não polinomial**
 - Pode terminar a execução com uma saída que não soluciona o problema em questão

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Verificação do Desempenho de um Algoritmo

- Desempenho de um algoritmo é medido em função de dois parâmetros:
 - Velocidade em que ele produz um resultado
 - Quantidade de recursos que ele consome
- Entretanto, os parâmetros dependem da máquina que executa o algoritmo
 - Velocidade depende do processador e da memória
 - Quantidade de recursos é problema dependendo da quantidade de memória da máquina

Avaliação do desempenho de um algoritmo é realizada em função do tamanho de uma instância do problema

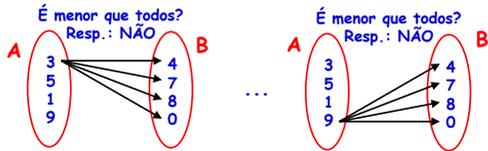
Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 1:** Comparar cada elemento de A com todos os outros de B



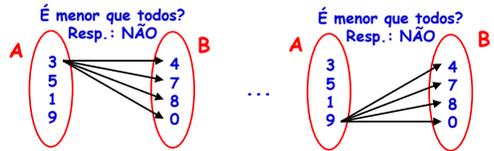
Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 1:** Comparar cada elemento de A com todos os outros de B



Verificação do Desempenho de um Algoritmo

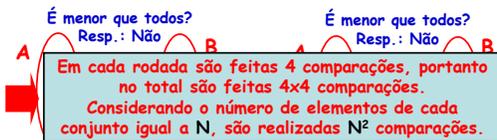
- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 1:** Comparar cada elemento de A com todos os outros de B



Como as respostas são sempre NÃO, o menor elemento está em B

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 1:** Comparar cada elemento de A com todos os outros de B



Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído

Conj. Concatenado = $\{3, 5, 1, 9, 4, 7, 8, 0\}$ Menor elemento = 3

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído

Conj. Concatenado = $\{3, 5, 1, 9, 4, 7, 8, 0\}$ Menor elemento = 3

3 é menor que 5?

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído

Conj. Concatenado = $\{3, 5, 1, 9, 4, 7, 8, 0\}$ Menor elemento = 3

3 é menor que 5?

Sim.

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído

Conj. Concatenado= $\boxed{3}5\boxed{1}9,4,7,8,0$ Menor elemento = 3

3 é menor que 1?

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído

Conj. Concatenado= $\boxed{3}5\boxed{1}9,4,7,8,0$ Menor elemento = 3

3 é menor que 1?

NÃO!

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído

Conj. Concatenado= $\boxed{3}5\boxed{1}9,4,7,8,0$ Menor elemento = 1

3 é menor que 1?

NÃO!

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído

Conj. Concatenado= $\{3,5,1,9,4,7,8,0\}$

Após comparar com todos os elementos, chegaremos na seguinte situação...

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído

Conj. Concatenado= $\{3,5\boxed{1}9,4,7,8\boxed{0}$ Menor elemento = 1

1 é menor que 0?

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído

Conj. Concatenado= $\{3,5\boxed{1}9,4,7,8\boxed{0}$ Menor elemento = 1

1 é menor que 0?

NÃO!

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído
 - Conj. Concatenado= $\{3, 5, 1, 9, 4, 7, 8, 0\}$ Menor elemento = 0

1 é menor que 0?

NÃO!

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído
 - Conj. Concatenado= $\{3, 5, 1, 9, 4, 7, 8, 0\}$

Como o 0 estava na segunda metade do conjunto concatenado, então o menor elemento estava em B.

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 2:** Concatenar os dois conjuntos e descobrir qual a posição do menor elemento. Define-se que o primeiro elemento é o menor e realiza-se a comparações com todos os outros. Toda vez que a premissa falhar, o menor elemento é substituído



São realizadas 7 comparações no total. Considerando o número de elementos de cada conjunto igual a N, são realizadas $2N-1$ comparações.

Verificação do Desempenho de um Algoritmo

- **Problema:**
 - Descobrir qual dos dois conjuntos possui o menor elemento
 - $A = \{3, 5, 1, 9\}$ e $B = \{4, 7, 8, 0\}$
- **Algoritmo 1:** Executado em N^2 comparações
- **Algoritmo 2:** Executado em $2N-1$ comparações
 - Para N muito grande, o algoritmo 2 se torna mais eficiente



Para evitar que a eficiência de um algoritmo seja medida em função de parâmetros dependentes de uma máquina, utiliza-se como parâmetro de análise o número máximo de passos computacionais que o algoritmo precisa para terminar.

Exemplos de Problemas Resolvidos por Algoritmos

- **Projeto Genoma**
 - Análise da sequência de genes humanos requer algoritmos eficientes dada a grande quantidade de dados de entrada
- **Internet**
 - Busca de menor caminho entre qualquer par origem-destino na rede é um exemplo clássico que requer algoritmos eficientes dado o número enorme de nós na Internet
- **Comércio eletrônico**
 - Análise de bancos de dados com informações sigilosas de usuários também requer algoritmos eficientes para tornar a busca e verificação a mais rápida possível

Exemplos de Problemas Resolvidos por Algoritmos

- **Projeto Genoma**
 - Análise da sequência de genes humanos requer algoritmos eficientes dada a grande quantidade de dados de entrada
- **Internet**
 - Busca de menor caminho entre qualquer par origem-destino na rede é um exemplo clássico que requer algoritmos eficientes dado o número enorme de nós na Internet
- **Comércio eletrônico**
 - Análise de bancos de dados com informações sigilosas de usuários também requer algoritmos eficientes para tornar a busca e verificação a mais rápida possível

Representação dos Dados

- Dados de entrada e saída dos algoritmos devem:
 - Ser representados de maneira estruturada
 - A estrutura deve ser conhecida e deve estar de acordo com o algoritmo utilizado
 - Definição de interfaces, protótipos de função
 - Descrever as características de uma instância e da solução encontrada para o problema

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estruturas de Dados

- São formas de armazenar e organizar dados para facilitar o acesso e as possíveis modificações
 - Nenhuma estrutura de dados é adequada a todos os casos
 - É importante conhecer os pontos fortes e as limitações de cada uma de acordo com o problema atacado
 - Ex.: Como representar um número inteiro?
 - » Inteiro com 8 bits → Ocupa pouca memória, mas representa até 256 valores
 - » Inteiro com 32 bits → Ocupa mais memória, mas representa mais de 4 bilhões de valores
 - Dependem do sistema operacional

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Representação dos Dados

- Dados de entrada e saída dos algoritmos devem:
 - Ser representados de maneira estruturada
 - A estrutura deve ser conhecida e deve estar de acordo com o algoritmo utilizado
 - Definição de interfaces, protótipos de função
 - Descrever as características de uma instância e da solução encontrada para o problema



Como representar os dados?

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estruturas de Dados

- As estruturas de dados usadas devem estar de acordo com o algoritmo empregado
 - Algoritmos e estruturas de dados são relacionados
 - A maneira como os dados são representados determinam as operações possíveis
 - As estruturas de dados escolhidas devem contribuir para uma solução eficiente

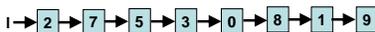
Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estruturas de Dados

- As estruturas de dados usadas devem estar de acordo com o algoritmo empregado
 - Algoritmos e estruturas de dados são relacionados
 - A maneira como os dados são representados determinam as operações possíveis
 - As estruturas de dados escolhidas devem contribuir para uma solução eficiente

Qual o maior elemento da lista? Qual o número máximo de elementos a pesquisar?



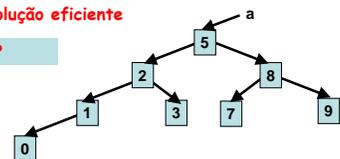
Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estruturas de Dados

- As estruturas de dados usadas devem estar de acordo com o algoritmo empregado
 - Algoritmos e estruturas de dados são relacionados
 - A maneira como os dados são representados determinam as operações possíveis
 - As estruturas de dados escolhidas devem contribuir para uma solução eficiente

E agora?



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Resolução de um problema
 - Algoritmo
 - Representação do comportamento de uma solução
 - Possui característica **ESTÁTICA**
 - Sequência de passos definidos previamente
 - Não pode ser alterada durante a resolução do problema
 - Estrutura de dados
 - Representação da informação
 - Possui característica **DINÂMICA**
 - Evolui conforme o algoritmo vai sendo executado
 - » Execução: Evolui no tempo

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Exemplo:
 - Algoritmo
 - Insira todo novo elemento no final da fila
 - Estrutura de dados
 - A fila

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Exemplo:
 - Algoritmo
 - Insira todo novo elemento no final da fila
 - Estrutura de dados
 - A fila

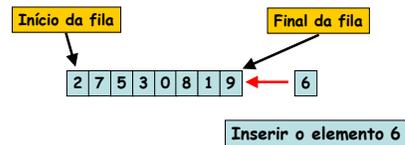


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Exemplo:
 - Algoritmo
 - Insira todo novo elemento no final da fila
 - Estrutura de dados
 - A fila

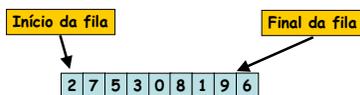


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Exemplo:
 - Algoritmo
 - Insira todo novo elemento no final da fila
 - Estrutura de dados
 - A fila

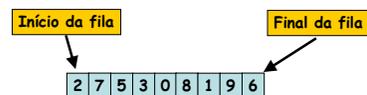


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Exemplo:
 - Algoritmo
 - Insira todo novo elemento no final da fila
 - Estrutura de dados
 - A fila



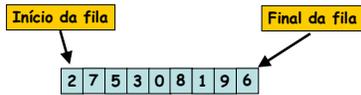
O algoritmo é o mesmo, mas a lista aumentou

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Estado de execução de um algoritmo
 - "Retrato" do sistema em um determinado momento da execução do algoritmo
 - Estado atual das estruturas de dados e variáveis

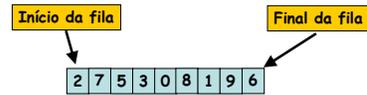


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Estado de execução de um algoritmo
 - "Retrato" do sistema em um determinado momento da execução do algoritmo
 - Estado atual das estruturas de dados e variáveis



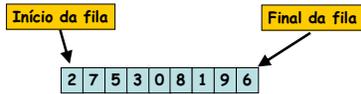
Ex.: Elementos na fila, qual o último elemento, o primeiro etc.

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Ação
 - Modifica o estado atual do sistema
 - Algoritmo descreve uma sequência de ações que podem ser realizadas para alterar o estado atual do sistema
 - Ações obedecem sempre o mesmo padrão
 - » Característica estática

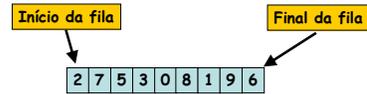


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Ação
 - Modifica o estado atual do sistema
 - Algoritmo descreve uma sequência de ações que podem ser realizadas para alterar o estado atual do sistema
 - Ações obedecem sempre o mesmo padrão
 - » Característica estática



Ex.: A ação é a inserção de um determinado elemento no final da fila

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Dificuldade
 - Compreensão das características dinâmicas e estáticas
 - Como as estruturas de dados evoluem durante a execução do algoritmo?
 - As estruturas de dados utilizadas são mesmo as mais eficientes?

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Algoritmo e Estrutura de Dados

- Dificuldade
 - Compreensão das características dinâmicas e estáticas
 - Como as estruturas de dados evoluem durante a execução do algoritmo?
 - As estruturas de dados utilizadas são mesmo as mais eficientes?

Requisitos necessários para encontrar uma solução adequada para o problema

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Desenvolvimento de um Algoritmo + Estrutura de Dados

- Programador:
 - É quem focaliza o problema e o modela computacionalmente
 - Escolhe as ferramentas que lhe permite resolver o problema
- Algoritmos e estruturas de dados
 - Representados por programas
- Programas
 - Expressos por linguagens computacionais

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

O que é um Programa Computacional?

- É um algoritmo expresso em uma **linguagem de programação** formal onde se conhece...
 - A maneira como representá-lo na linguagem de programação
 - A estrutura de representação dos dados
 - Os tipos e as operações suportados pelo computador

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

O que é um Programa Computacional?

- Assim como um algoritmo, todo programa é executado:
 - Seguindo uma sequência estática de comandos
 - Exemplo:

```
#include <stdio.h>
main() {
    ENQUANTO condição satisfeita FAÇA
        execute ação 1;
    FIM DO ENQUANTO
    imprimir "Acabou";
}
```

Sequência de execução

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

O que é um Programa Computacional?

- Assim como um algoritmo, todo programa é executado:
 - Seguindo uma sequência estática de comandos
 - Exemplo:

```
#include <stdio.h>
main() {
    ENQUANTO condição satisfeita FAÇA
        execute ação 1;
    FIM DO ENQUANTO
    imprimir "Acabou";
}
```

Loop de programação

Sequência de execução

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Linguagens de Programação

- Envolve sempre dois aspectos
 - Semântica
 - Significado
 - Sintaxe
 - Forma
- Utiliza estruturas básicas de controle
 - Formas naturais de pensar e adequadas à construção de algoritmos inteligíveis

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Linguagens de Programação

- Envolve sempre dois aspectos
 - Semântica
 - Significado
 - Sintaxe
 - Forma
- Utiliza estruturas básicas de controle
 - Formas naturais de pensar e adequadas à construção de algoritmos inteligíveis

```
SE condição satisfeita ENTÃO
    execute ação 1
SENÃO
    execute ação 2
```

-Semântica: Condição e desvio
-Sintaxe: Maneira como se representa a condição e o desvio

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Linguagem C

- História do C
 - Evolução de duas outras linguagens de programação
 - BCPL (*Basic Combined Programming Language*)
 - Linguagem originalmente para desenvolver compiladores
 - B
 - Versão mais enxuta do BCPL (memória era escassa)
 - » Só tinha um tipo de dados, a palavra definida pela arquitetura do computador
 - » Operadores aritméticos tratavam as palavras como inteiros
 - » Outros operadores tratavam como endereço de memória

```
main() {  
    extern a, b, c;  
    putchar(a); putchar(b); putchar(c); putchar('\n');  
}  
a 'hell'; b 'o, w'; c 'orld';
```

Linguagem C

- História do C
 - Dennis Ritchie (Bell Laboratories) em 1972
 - Adiciona tipo de dados
 - Linguagem de desenvolvimento do UNIX
 - Independente do Hardware
 - Portabilidade de programas
 - 1989: Padrão ANSI
 - 1990: Publicação do padrão ANSI e ISO
 - ANSI/ISO 9899: 1990

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Por que a Linguagem C?

- Permite o desenvolvimento de programas **menores e mais rápidos**
 - Programador possui controle maior sobre o código
 - Programador deve:
 - Definir onde armazenar as variáveis na memória
 - Alocar e liberar a memória
 - Trabalhar com endereços de memória
 - Em compensação, a programação é mais detalhada
 - Detalhes que não são "preocupações" em linguagens de mais alto nível como: Linguagens de scripts, Java e C++
- Possui sintaxe simples
 - Não possui muitas funções embutidas

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Linguagem C++

- História do C++
 - Extensão do C
 - Bjarne Stroustrup (Bell Laboratories) em 1979
 - Linguagem desenvolvida para análise do kernel do UNIX
 - Extensão do C com características do Simula
 - 1983: Passou de C com classes para C++
 - 1985: Primeira implementação comercial do C++
 - 1998: Primeiro padrão ISO/IEC

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Por que a Linguagem C++?

- Aumento de exigências de mercado
 - Reuso de software
 - Modularidade, facilidade de modificação
 - Rapidez, correção e economia
 - Programação em mais alto nível
- Programação orientada a objetos
 - Objetos: componentes reutilizáveis de software
 - Modelam itens do mundo real
 - Programas orientados a objetos
 - Mais fáceis de compreender, corrigir e modificar

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estrutura de Blocos e Identação

- Formato para apresentação dos programas em linguagens estruturadas
 - Blocos são conjuntos de comandos com uma função bem definida
 - Servem para definir os limites onde as variáveis declaradas em seu interior são definidas
 - São delimitadas (por chaves no C)
 - Início é representado por {
 - Fim é representado por }
 - Um bloco contido em outro nunca deve terminar antes que o bloco anterior
 - Identação (Dentear)
 - Serve para facilitar a identificação dos blocos

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Bloco A {

Bloco B {

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Bloco A {

Bloco B {

Bloco Principal }

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Bloco A {

Bloco B {

Bloco Principal }

Início dos blocos

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Bloco A {

Bloco B {

Bloco Principal }

Final dos blocos

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Diferencia o bloco principal dos blocos internos

Identação: Diferencia os blocos dos seus conteúdos

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Identação:
Diferencia os
blocos dos seus conteúdos

Diferencia os
blocos internos dos
seus conteúdos

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

O que é impresso na tela?

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Um bloco NÃO PODE
começar antes de um outro
terminar e acabar depois!

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Esse código seria
compreendido da seguinte
maneira:

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
    }
    IF (n <= 5) {
        imprimir "n <= 5";
    }
}
```

Esse código seria
compreendido da seguinte
maneira:

Um bloco contido no outro

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Estrutura de Blocos e Identação

```
#include <stdio.h>
main() {
    int n = 3;
    IF (n > 5) {
        imprimir "n > 5";
        IF (n <= 5) {
            imprimir "n <= 5";
        }
    }
}
```

A boa prática da programação exige indentação...

Função

- São programas completos e independentes
 - Podem ser invocados por um outro programa
 - Realizam uma determinada atividade e retornam o resultado obtido quando houver

```
#include <stdio.h>
main() {
    SE condição satisfeita ENTÃO
        função1("a");
    FIM DO SE
}

função1(char c) {
    imprimir(c);
}
```

Como um Programa é Executado?

- Linguagens de programação
 - São projetadas em função da facilidade na construção do código e da confiabilidade dos programas
 - Quanto mais próximo a linguagem de programação estiver da forma de raciocínio humano, mais intuitivo se torna o programa e mais simples é a programação

```
#include <stdio.h>
main() {
    ENQUANTO condição satisfeita FAÇA
        execute ação 1;
    FIM DO ENQUANTO
    imprimir "Acabou";
}
```

Como um Programa é Executado?

- Entretanto, computadores não entendem a linguagem humana...
 - Computadores entendem seqüências de 0's e 1's
 - Chamada de linguagem de máquina

```
#include <stdio.h>
main() {
    ENQUANTO condição satisfeita FAÇA
        execute ação 1;
    FIM DO ENQUANTO
    imprimir "Acabou";
}
```



```
1 0 1 1 0
0 0 1 1 0
...
0 1 0 1 0
0 1 0 0 1
```

Arquitetura de Computadores em Multiníveis

- Linguagem de programação C++
 - Linguagem de nível alto
 - Abstrai ao máximo do programador detalhes da arquitetura do computador
 - Processador, detalhes da memória, dispositivos de entrada e saída etc.
 - Outros exemplos são C, Java, Fortran etc.
- Linguagem de máquina
 - Linguagem de nível baixo
 - Depende da arquitetura do processador
 - Cada processador define uma arquitetura de conjunto de instruções (Instruction Set Architecture - ISA)

Arquitetura de Computadores em Multiníveis

- Existem duas maneiras para decodificar programas
 - Programa em linguagem de nível alto para programa em linguagem de nível baixo
 - Interpretação
 - Tradução

Interpretação

- Na interpretação **cada** comando em linguagem de programação de alto nível é decodificado e executado
 - Processo realizado durante a execução do programa
 - Um comando por vez
- Para isso,
 - Há um programa interpretador sendo executado
 - Um interpretador para cada arquitetura de processador
 - Cada comando do código é visto por esse interpretador como um dado de entrada

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Interpretação

- O computador executa programas auxiliares escritos em linguagem de máquina para interpretar cada comando do programa
 - Os programas auxiliares são invocados em uma ordem apropriada de acordo com a ordem de execução do programa
- Etapas da interpretação
 - Obter o próximo comando
 - Examinar e decodificar o comando
 - Executar as ações

Interpretação

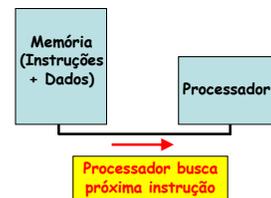
- O computador executa programas auxiliares escritos em linguagem de máquina para interpretar cada comando do programa
 - Os programas auxiliares são invocados em uma ordem apropriada de acordo com a ordem de execução do programa
- Etapas da interpretação
 - Obter o próximo comando
 - Examinar e decodificar o comando
 - Executar as ações



Ciclo de execução de um programa

Interpretação

- O ciclo de execução de um programa é usado durante o processamento
 - Operação realizada para processar linguagem de nível baixo
 - Ciclo denominado "busca-decodificação-execução"

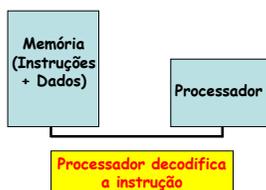


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Interpretação

- O ciclo de execução de um programa é usado durante o processamento
 - Operação realizada para processar linguagem de nível baixo
 - Ciclo denominado "busca-decodificação-execução"

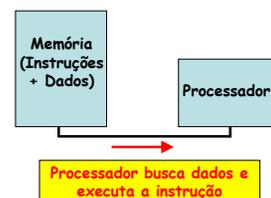


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Interpretação

- O ciclo de execução de um programa é usado durante o processamento
 - Operação realizada para processar linguagem de nível baixo
 - Ciclo denominado "busca-decodificação-execução"



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Interpretação

- Qual é a relação do ciclo de execução com a interpretação?
 - O ciclo de execução é realizado em um nível mais baixo
 - Com instruções em formato que o processador consegue decodificar
 - A interpretação utiliza o mesmo processo de “busca-decodificação-execução” em um nível mais alto
 - Portanto, o processo de interpretação é uma abstração em nível alto de um processo de nível baixo
 - Considera que a linguagem de alto nível é a própria linguagem de baixo nível

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Tradução

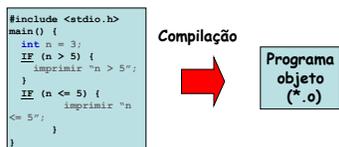
- Programa em linguagem de programação em nível alto é totalmente decodificado em um programa em linguagem de nível baixo
 - Processo realizado antes da execução do programa
 - Processo gera um novo programa
 - Programa gerado em nível baixo é equivalente ao programa original em nível alto
- Para isso,
 - Programa é decodificado em um processo chamado de compilação
 - Programa que realiza a compilação é chamado de **compilador**
 - Um compilador para cada arquitetura de processador

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Tradução

- A tradução pode ser dividida em duas grandes partes:
 - Análise do programa fonte
 - Dados de entrada
 - Síntese do programa objeto executável



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

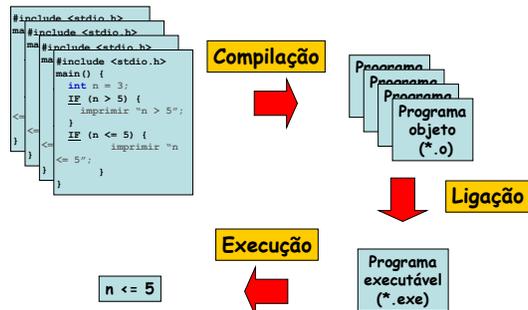
Tradução

- A saída de um processo de compilação consiste em:
 - Programas objetos (*.o)
 - Programas “quase” executáveis
 - Podem fazer referências a dados externos ou outros programas
- Ligação
 - Realizada por um programa **ligador**
 - Une diversos programas objetos em um único programa executável
 - Um programa em alto nível pode ser composto de diversos subprogramas ou pode fazer referência a programas externos ou ainda pode utilizar funções definidas em bibliotecas externas

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Tradução



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Interpretação X Tradução

- Compiladores e interpretadores dependem da arquitetura do processador (ISA)
 - Intel x86, SPARC, AMD64 etc.
- Os programas interpretados são **sempre reinterpretados** durante a execução
 - Independente da arquitetura do processador
 - Em compensação, o desempenho pode ser mais baixo pois todos os comandos são interpretados antes de executar
- Os programas compilados **não precisam ser recompilados**
 - Torna a execução mais rápida
 - Em compensação, é dependente da arquitetura do processador

Estudo de Caso: Linguagem de Programação Java

- Para evitar que todos os programas sejam recompilados para cada arquitetura de processadores
 - O Java utiliza uma máquina virtual (*Java Virtual Machine - JVM*)
 - Assim, os programas são sempre compilados para a mesma arquitetura, a arquitetura da JVM
 - Essa característica torna os programas multiplataformas
 - Funcionam para qualquer arquitetura sem precisar de recompilação
 - Programa compilado chamado de *bytecode*
 - Entretanto, para essa característica ser possível
 - Deve existir uma JVM para cada arquitetura
 - É melhor instalar uma JVM específica e executar qualquer programa do que ter que recompilar cada um dos programas para cada arquitetura

Estudo de Caso: Linguagem de Programação Java



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Perguntas

- O que é um problema computacional?
- Para que servem os algoritmos e as estruturas de dados?
- Qual o problema de se tratar um problema com alto nível de abstração?
- Qual a diferença de tradução e interpretação?

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

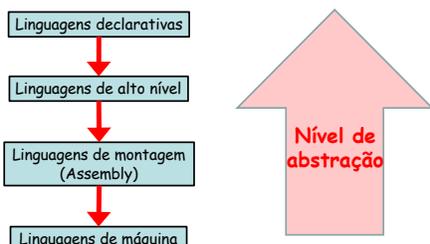
Níveis de Linguagens de Programação

- Afinal quantos níveis de linguagens existem?
- Linguagem de nível alto
 - Representada pelas linguagens que programamos?
- Vs.**
- Linguagem de nível baixo
 - Representada por linguagens que "detestamos"?

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Níveis de Linguagens de Programação



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Níveis de Linguagens de Programação

- Linguagens declarativas
 - Linguagens expressivas como a linguagem oral
 - Expressam o que fazer ao invés de como fazer
- Linguagens de alto nível
 - Linguagens típicas de programação
 - Permitem que algoritmos sejam expressos em um nível e estilo de escrita fácil para leitura e compreensão
 - Possuem características de portabilidade, já que podem ser transferidas de uma máquina para outra
- Linguagens de montagem e linguagens de máquina
 - Linguagens que dependem da arquitetura da máquina
 - Linguagem de montagem é uma representação simbólica da linguagem de máquina associada

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Níveis de Linguagens de Programação

Pascal	Linguagem de Montagem	Linguagem de Máquina
Z:= W+X*Y	LOAD 3,X	41 3 0C1A4
	MULTIPLY 2,Y	3A 2 0C1A8
	ADD 3,W	1A 3 0C1A0
	STORE 3,Z	50 3 0C1A4

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Níveis de Linguagens de Programação

Pascal	Linguagem de Montagem	Linguagem de Máquina
Z:= W+X*Y	LOAD 3,X	41 3 0C1A4
	MULTIPLY 2,Y	3A 2 0C1A8
	ADD 3,W	1A 3 0C1A0
	STORE 3,Z	50 3 0C1A4

Correspondência 1 para 1

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programa Montador

- Responsável por converter o programa na linguagem de máquina correspondente

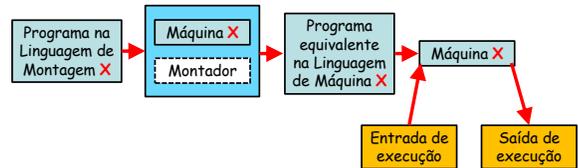


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programa Montador

- Responsável por converter o programa na linguagem de máquina correspondente

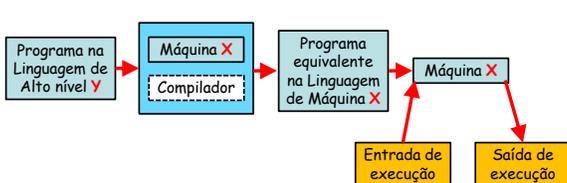


Como ficaria o programa compilador? E o interpretador?

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programa Compilador



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programa Interpretador



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Paradigmas de Programação em Alto Nível

Programação Imperativa

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programação Imperativa

- Chamada também de programação algorítmica
- Descreve a computação em detalhes em termos de **sentenças** que mudam o estado do programa
 - Define sequências de comandos para o computador executar
 - **Semelhante a uma linguagem oral imperativa:**
 - **Chefe:** - Some dois números!
 - **Chefe:** - Exiba o resultado!
 - **Chefe:** - Volte ao seu trabalho anterior!
 - **Chefe:** - etc.
 - **Relembrando:** Estado de um programa é definido pelas suas estruturas de dados e variáveis

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sentenças

- Menor elemento em uma linguagem de programação imperativa capaz de realizar mudança de estado
 - Sentença simples
 - **Atribuição:** `a = a + 1`
 - **Chamada:** `funcao()`
 - **Retorno:** `return 0`
 - **Desvio:** `goto 1`
 - **Assertão:** `assert(a == 0)`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sentenças

- Menor elemento em uma linguagem de programação imperativa capaz de realizar mudança de estado
 - Sentença composta
 - **Bloco:** `begin`
`write('Y');`
`end`
 - **Condição:** `if a > 3 then`
`write('Y');`
`else`
`write('N');`
`end`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sentenças

- Menor elemento em uma linguagem de programação imperativa capaz de realizar mudança de estado
 - Sentença composta
 - **Chaveamento:** `switch (c)`
`case 'a':`
`alert(); break;`
`case 'q':`
`quit(); break;`
`end`
 - **Laço de repetição:** `while a > 3 do`
`write('Y');`
`end`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sentenças

- Diferenças nas sintaxes
 - Separação de sentenças
 - Término de sentenças

Linguagem	Separação/Terminação de Sentença
Cobol	. (ponto)
C e C++	; (ponto e vírgula)
Java, Perl	: (ponto e vírgula)
Python	Nova linha
Lua	Espaço em branco (separando)

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Paradigmas de Programação em Alto Nível



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programação Não-estruturada

- Tipo de programação **imperativa**
 - Código caracterizado pela presença de sentenças do tipo **goto**
 - Cada sentenças ou linha de código é identificada por um rótulo ou um número
 - **Chefe:** 10 - Imprimir resultado
 - **Chefe:** 20 - Se A+B for maior que C
 - **Chefe:** 30 - Vá para 10
 - **Chefe:** 40 - Se A+B for menor que C
 - **Chefe:** 50 - Some mais 1
 - Oferece liberdade de programação
 - Entretanto...
 - Torna o código complexo

Programação Não-estruturada

- Tipo de programação **imperativa**
 - Código caracterizado pela presença de sentenças do tipo **goto**
 - Cada sentenças ou linha de código é identificada por um rótulo ou um número
 - **Chefe:** 10 - Imprimir resultado
 - **Chefe:** 20 - Se A+B for maior que C
 - **Chefe:** 30 - Vá para 10
 - **Chefe:** 40 - Se A+B for menor que C
 - **Chefe:** 50 - Some mais 1
 - Oferece liberdade de programação
 - Entretanto...
 - Torna o código complexo

O "goto" e o "if", por algum tempo, eram as únicas estruturas de controle das linguagens de programação. Não existia, p.ex., o "while" nem o "se-então-senão"!

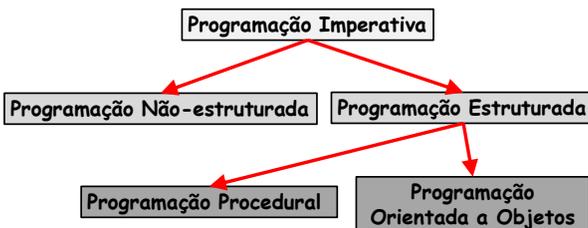
Programação Estruturada

- Tipo de programação **imperativa**
 - Basicamente não utiliza sentenças do tipo **goto**
 - Dispensa os rótulos
 - **Chefe:** - Se A+B for maior que C
 - **Chefe:** - Imprimir resultado
 - **Chefe:** - Caso contrário
 - **Chefe:** - Some mais 1

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Paradigmas de Programação em Alto Nível



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programação Procedural

- Tipo de programação **imperativa e estruturada** baseada em **procedimentos**
 - Procedimentos são sinônimos de funções, métodos ou sub-rotinas
 - Ex.: Linguagem C
 - **Chefe:** - somar(a, b)
 - **Chefe:** - imprimir("terminado!")
 - **Chefe:** - voltar

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programação Procedural

- Tipo de programação **imperativa e estruturada** baseada em **procedimentos**
 - Procedimentos são sinônimos de funções, métodos ou sub-rotinas
 - Ex.: Linguagem C
 - **Chefe:** - somar(a, b)
 - **Chefe:** - imprimir("terminado!")
 - **Chefe:** - voltar



Resolve o problema por partes, subdividindo-o até que a subdivisão seja simples o suficiente para ser resolvida por apenas um procedimento.

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programação Procedural

- Uso de procedimentos permite:
 - Reuso de procedimentos em diferentes partes do código
 - **Chefe:** r = somar(a,b)
 - **Chefe:** imprimir(r)
 - **Chefe:** r = somar(a,r)

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programação Procedural

- Uso de procedimentos permite:
 - Reuso de procedimentos em diferentes partes do código!
 - **Aumenta a eficiência da programação**
 - **Chefe:** r = somar(a,b)
 - **Chefe:** imprimir(r)
 - **Chefe:** r = somar(a,r)

Reuso do mesmo procedimento

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programação Orientada a Objetos

- Tipo de programação **imperativa e estruturada**, porém...
 - Enquanto a programação procedural é estruturada em...
 - **Procedimentos**
 - Estruturas de dados e algoritmos
 - A programação orientada a objetos é estruturada em...
 - **Classes e objetos**
 - Objetos encapsulam estruturas de dados e procedimentos

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Programação Orientada a Objetos

- Tipo de programação **imperativa e estruturada**, porém...
 - Enquanto a programação procedural é estruturada em...
 - **Procedimentos**
 - Estruturas de dados e algoritmos
 - A programação orientada a objetos é estruturada em...
 - **Classes e objetos**
 - Objetos encapsulam estruturas de dados e procedimentos

Procedural

```
main() {
  define_carro();
  entra_carro();
  sair_carro();
}
```

Orientada a objetos

```
main() {
  Carro carro;
  carro.entrar();
  carro.sair();
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Outros Paradigmas de Programação...

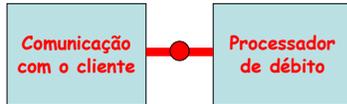
- Programação baseada em eventos
 - Fluxo do programa é determinado pelo surgimento de eventos
 - **Eventos podem ser disparados pela recepção de mensagens ou expiração de temporizadores**
- Programação orientada a agentes
 - Programa é estruturado em agentes
 - **Agente é uma abstração de um software capaz de tomar decisões autônomas**
 - Ao invés de métodos e atributos, um **agente possui comportamento**

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Outros Paradigmas de Programação...

- Programação orientada a componentes
 - Programa cujo o objetivo é unir blocos funcionais
 - Diferente da orientação a objetos, não há a preocupação em modelar objetos como objetos da vida real



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Linguagens de Programação Estruturadas

- Assim como os programadores e as aplicações...
 - As linguagens são especializadas dependendo da aplicação para qual foram desenvolvidas
 - Aplicações científicas
 - Especializadas em manipulação de números e vetores
 - Empregam ferramentas matemáticas e estatísticas
 - Requerem mais processamento que entrada e saída de dados
 - » Ex.: Pascal, Fortran, APL
 - Aplicações de processamento de dados
 - Especializadas na criação, manutenção, mineração e resumo de dados em registros ou em arquivos
 - Requerem entrada e saída e nem tanto de processamento
 - » Exs.: Cobol e PL/I

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Linguagens de Programação Estruturadas

- Assim como os programadores e as aplicações...
 - As linguagens são especializadas dependendo da aplicação para qual foram desenvolvidas
 - Aplicações de processamento de texto
 - Especializadas em manipulação de textos em linguagem natural, ao invés de números e dados
 - » Ex.: SNOBOL
 - Aplicações de inteligência artificial
 - Especializadas na emulação de comportamento inteligente
 - Incluem algoritmos de jogos, reconhecimento de padrão etc.
 - » Exs.: LISP e Prolog

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Linguagens de Programação Estruturadas

- Assim como os programadores e as aplicações...
 - As linguagens são especializadas dependendo da aplicação para qual foram desenvolvidas
 - Aplicações de programação de sistemas
 - Especializadas no desenvolvimento de programas para interface entre o programa e o hardware da máquina
 - Lidam com eventos imprevistos como erros
 - Incluem compiladores, interpretadores, montadores etc.
 - » Exs. Ada e Modula-2

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Linguagens de Programação Estruturadas

- Assim como os programadores e as aplicações...
 - As linguagens são especializadas dependendo da aplicação para qual foram desenvolvidas
 - Aplicações de programação de sistemas
 - Especializadas no desenvolvimento de programas para interface entre o programa e o hardware da máquina
 - Lidam com eventos imprevistos como erros
 - Incluem compiladores, interpretadores, montadores etc.
 - » Exs. Ada e Modula-2

Apesar da motivação inicial de desenvolvimento, com o passar do tempo, as linguagens se tornaram mais versáteis e completas. Ex.: C++, Lua e Python

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

CrITÉRIOS de Avaliação e Comparação de Linguagens

- Expressividade
 - Capacidade de refletir com clareza o seu objetivo
 - Ex.: $C = A + B$
 $C := A + B$
(SETQ C(+ A B))
ADD A, B GIVING C
- Delineamento
 - Capacidade da linguagem não apresentar ambiguidades
- Estruturas e tipos de dados
 - Suporte a diferentes estruturas de dados e tipos
- Modularidade
 - Suporte à subprogramação e à extensão

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Critérios de Avaliação e Comparação de Linguagens

- **Entrada e saída**
 - Suporte a diferentes maneiras de acesso a dados e arquivos
- **Portabilidade**
 - Dependência de máquinas específicas
- **Eficiência**
 - Velocidade de compilação/tradução e execução
- **Generalidade**
 - Capacidade de uso em diferentes aplicações
- **Simplicidade de aprendizado**

Leitura Recomendada

- Deitel e Deitel, "*C++ How to Program*", 5th edition, Editora Prentice Hall, 2005 -- Capítulo 1
- Waldemar Celes, Renato Cerqueira e José Lucas Rangel, "Introdução a Estrutura de Dados com Técnica de Programação em C", Editora Campus-Elsevier, 2004 -- Capítulo 1
- Andrew S. Tanenbaum, "Livro Organização Estruturada de Computadores", Ed. Pearson, 5a edição -- Capítulo 1
- Allen B. Tucker, "Programming Languages", Editora McGrawHill, 2ª Edição, 1985 -- Capítulo 1