

Algoritmos e Programação (Prática)

Profa. Andreza Leite
andrzea.leite@univasf.edu.br

Algoritmos e Programação



□ **Introdução**

- O computador como ferramenta indispensável:
 - Faz parte das nossas vidas;
 - Por si só não faz nada de útil;
 - Grande capacidade de resolução de problemas;
 - Necessita ser instruído;
 - É extremamente rápido;
 - Possui um comportamento previsível;
 - Não se cansa e pode ser usado à exaustão.

Algoritmos e Programação

□ Introdução

- Computador → É uma máquina capaz de possibilitar variados tipos de tratamento automático de informações ou processamento de dados.
- O que deve ser feito para que um determinado tratamento automático de informações ocorra?
 - Deve-se instruir o computador para que o mesmo utilizando-se de sua estrutura, execute determinada tarefa.
- Como?
 - Software (programas) → Sequências de instruções a serem executadas por um computador.

Algoritmos e Programação

□ Introdução

□ Software:

- Parte intangível: conhecimentos e idéias que fazem o hardware exibir um certo comportamento.
- Quanto mais usado, menos propenso à falhas.
- Confere funcionalidade ao hardware.
- Pode ser adquirido ou desenvolvido.

□ Comprar ou desenvolver?

- Depende do problema que se quer resolver;
- Avaliar custo x benefício;
- Diferentes plataformas;
- Manutenção e customização.

Algoritmos e Programação

□ Introdução

□ **Nosso objetivo:** Desenvolver software.

- Organização de idéias;
- Modelo de funcionamento do computador;
- Conceitos básicos de programação;
- Transcrição para linguagens apropriadas;
- Comunicação e interação com o computador;
- Obtenção dos resultados pretendidos;
- Prática em laboratório.

□ **Roteiro:**

1. Problema;
2. Solução;
3. Algoritmo;
4. Programa;
5. Resultados.

Algoritmos e Programação

□ Introdução

1. Problema

- Precisa ser conhecido em todos os seus aspectos;
- É necessário ter resposta para todas as perguntas que dele possam suscitar;
- É fundamental considerar todas as situações adversas;
- Nenhum detalhe deve ser omitido.

2. Solução

- Existe solução para o problema?
- Qual o custo da sua implementação?
- Qual o custo da sua execução?
- Como iremos representá-la?

Algoritmos e Programação

□ Introdução

3. Algoritmo

- Representação de uma solução para um problema, com algumas características:
 - Seqüência finita de etapas;
 - Individualmente, existe realização possível para cada uma das etapas consideradas;
 - Termina após um tempo finito.
- Diversas são as técnicas e métodos existentes para a construção de algoritmos.
 - No entanto, todas elas possuem um mesmo objetivo, onde as suas variações não passam de pequenos detalhes frente a sua organização geral no atendimento a uma ou outra área mais especificamente.

Algoritmos e Programação

□ Introdução

3. Algoritmo

□ Representação:

- Linguagem natural;
- Pseudocódigo (linguagem textual com poucos símbolos e regras, que são simples);
- Fluxograma (linguagem visual composta por poucos símbolos e regras)
- Um algoritmo expressa uma solução para um problema.

□ Acontece que:

- Computadores não entendem (normalmente, ou pelo menos da forma como nós precisamos):
 - Linguagens naturais;
 - Pseudocódigos;
 - Fluxogramas.

Algoritmos e Programação

□ Introdução

4. Programa

□ Precisamos então:

- Estabelecer um mecanismo de comunicação com o computador, de modo a fazê-lo entender tudo que quisermos que ele faça;
- Os computadores entendem somente instruções, postas em uma sequência lógica e seguindo diversas regras;
- Essas instruções e essas regras não são de fácil uso pelas pessoas, daí a necessidade de uma etapa intermediária, que é a construção de um **programa**, a partir do **algoritmo**;
- Para isso, vamos usar uma “linguagem de programação”.
- Um pouco mais complexas do que as linguagens usadas para representar algoritmos;
- Mas mais fáceis de serem entendidas pelo computador.

Algoritmos e Programação

□ Introdução

4. Programa

□ Precisamos então:

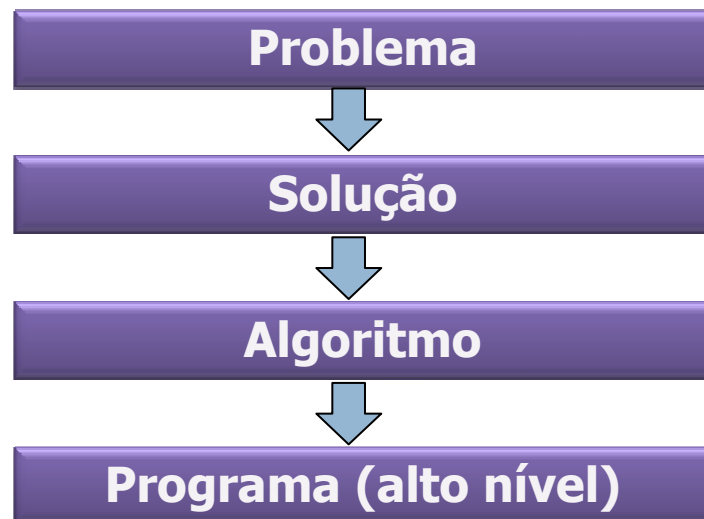
- Traduzir algoritmos para programas;
- Conhecer e utilizar (pelo menos) duas linguagens de programação;
- Entender que erros nas traduções do algoritmo para a linguagem são comuns;

Algoritmos e Programação

□ Introdução

Revendo e separando bem as coisas a partir daqui pra frente:

□ Sua parte:

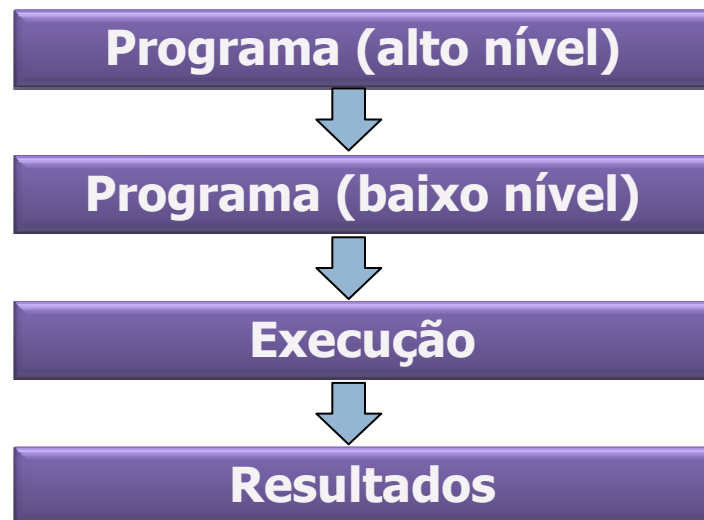


Algoritmos e Programação

□ Introdução

□ E se der errado?

- Voltar a mesa e descobrir onde está o erro.



Algoritmos e Programação

□ Introdução

□ Ciclo de desenvolvimento



Algoritmos e Programação

□ Algoritmos no mundo real

- Qualquer sequência de etapas para se resolver um problema ou chegar a um objetivo é um algoritmo.
- Exemplos:

ALGORITMO: TROCAR UMA LÂMPADA

PASSO 1: Pegar a lâmpada nova

PASSO 2: Pegar a escada

PASSO 3: Posicionar a escada embaixo da lâmpada queimada

PASSO 4: Subir na escada com a lâmpada nova

PASSO 5: Retirar a lâmpada queimada

PASSO 6: Colocar a lâmpada nova

PASSO 7: Descer da escada

PASSO 8: Ligar o interruptor

PASSO 9: Guardar a escada

PASSO 10: Jogar a lâmpada velha no lixo

ALGORITMO: SACAR DINHEIRO

PASSO 1: Ir até o caixa eletrônico

PASSO 2: Colocar o cartão

PASSO 3: Digitar a senha

PASSO 4: Solicitar o saldo

PASSO 5: Se o saldo for maior ou igual à quantia desejada, sacar a quantia desejada; caso contrário sacar o valor do saldo

PASSO 6: Retirar dinheiro e cartão

PASSO 7: Sair do caixa eletrônico

Algoritmos e Programação

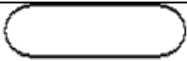



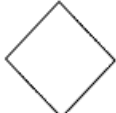


□ Algoritmos

- A elaboração do algoritmo descreve a necessidade dos dados e as suas manipulações durante a execução da lógica proposta por ele, sendo feito por meio de técnicas diferentes que representarão a seqüência desses passos (ou etapas).
- Entre as várias técnicas existentes, serão abordadas:
 - Fluxograma;
 - Pseudocódigo (Português Estruturado).
- **O fluxograma** utiliza figuras geométricas predefinidas para descrever as ações (ou instruções) a serem realizadas na resolução de um problema.
- **Pseudocódigo** consiste na descrição estruturada, por meio de regras pré-definidas, de passos (ou instruções) a serem realizados para a resolução do problema, utilizando a linguagem natural para representar o raciocínio.

Algoritmos e Programação

□ Fluxograma

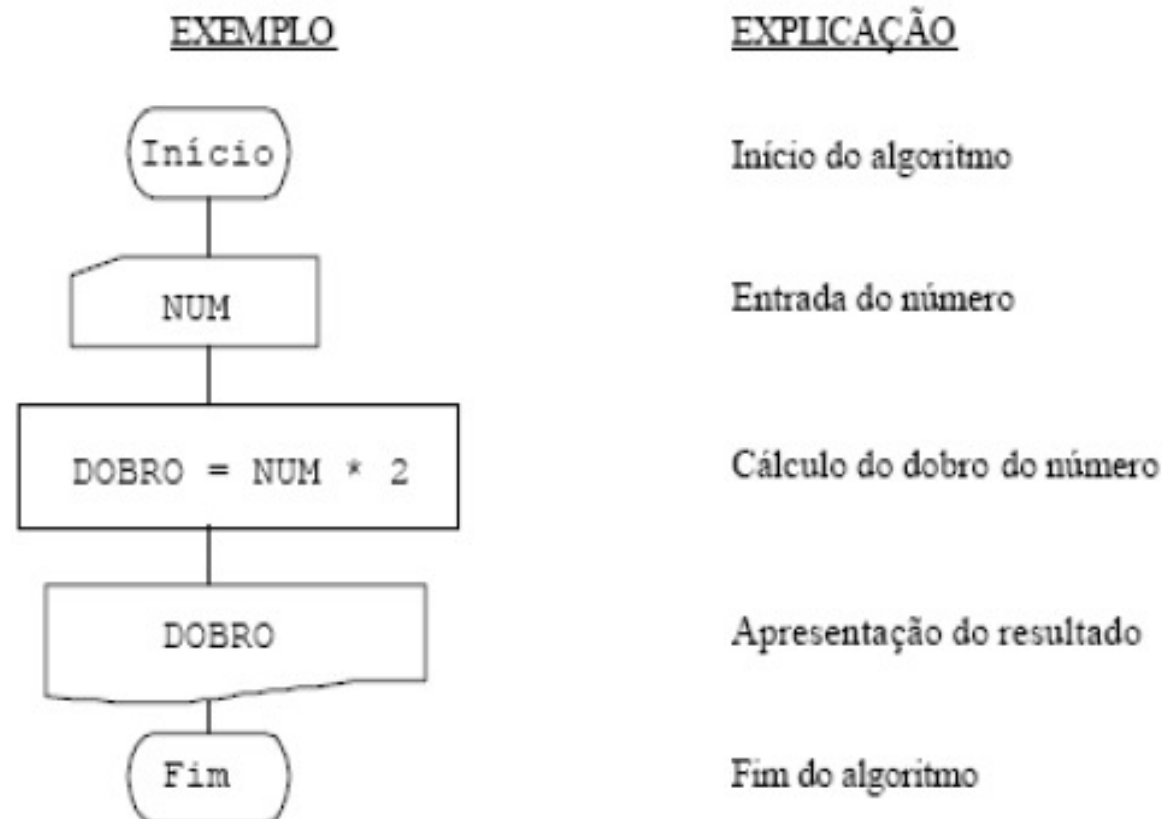
- Vantagem – a representação gráfica é mais concisa que a representação textual.
- Desvantagem – é necessário aprender a simbologia dos fluxogramas

FIGURA	SIGNIFICADO
	Figura para definir início e fim do algoritmo
	Figura usada no processamento de cálculo, atribuições e processamento de dados em geral
	Figura utilizada na representação de entrada de dados
	Figura utilizada para representação da saída de dados
	Figura que indica o processo seletivo ou condicional, possibilitando o desvio no caminho do processamento
	Símbolo geométrico usado como conector
	Símbolo que identifica o sentido do fluxo de dados, permitindo a conexão entre as outras figuras existentes

Algoritmos e Programação

□ Fluxograma

Exemplo de fluxograma: calcular o dobro de um número



Algoritmos e Programação

□ Pseudocódigo (Portugol)

- Descrição narrativa utilizando nosso idioma para descrever o algoritmo.
- Vantagem – sua transcrição para qualquer linguagem de programação é quase que direta.
- Desvantagem – é necessário aprender as regras do pseudocódigo.

□ Exemplo de uma descrição narrativa.

- Receber e efetuar a soma de dois números, exibindo o resultado ao final.
- Receber os dois números.
- Efetuar a soma dos dois números.
- Mostrar o resultado.

Algoritmos e Programação

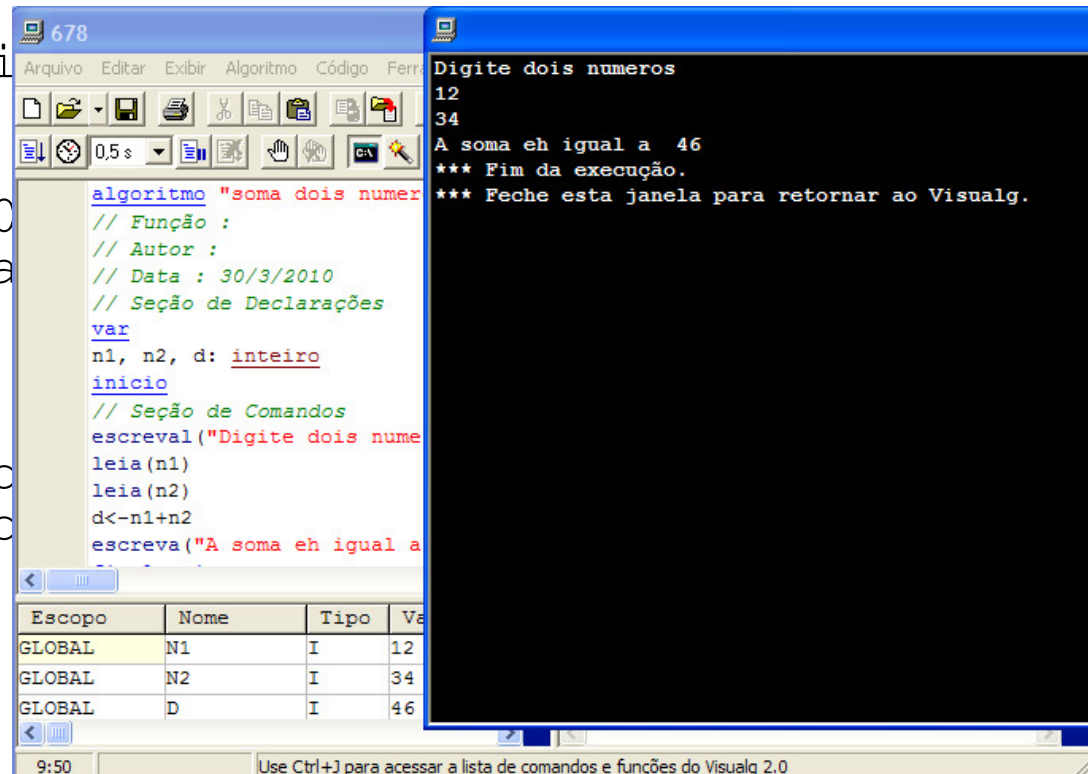
▣ Algoritmo em Pseudocódigo

```
ALGORITMO "SOMA DOIS NÚMEROS"  
DECLARE N1, N2, S NUMÉRICO  
ESCREVA "Digite dois números"  
LEIA N1, N2  
S ← N1 + N2  
ESCREVA "SOMA = " , S  
FIM_ALGORITMO
```

Algoritmos e Programação

Exemplo – Pseudocódigo (Visualg)

```
Algoritmo "soma dois numeros"
// Função :
// Autor :
// Data : 30/3/2010
// Seção de Declarações
var
n1, n2, d: inteiro
inicio
// Seção de Comandos
escreval("Digite dois numeros")
leia(n1)
leia(n2)
d<-n1+n2
escreva("A soma eh igual a ", d)
finalgoritmo
```



The screenshot shows the Visualg 2.0 IDE with two windows. The left window displays the pseudocode for a program that reads two integers and outputs their sum. The right window shows the execution output, including the prompts and the result.

```
algoritmo "soma dois numeros"
// Função :
// Autor :
// Data : 30/3/2010
// Seção de Declarações
var
n1, n2, d: inteiro
inicio
// Seção de Comandos
escreval("Digite dois numeros")
leia(n1)
leia(n2)
d<-n1+n2
escreva("A soma eh igual a ", d)
finalgoritmo
```

Output:

```
Digite dois numeros
12
34
A soma eh igual a 46
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Escopo	Nome	Tipo	Valor
GLOBAL	N1	I	12
GLOBAL	N2	I	34
GLOBAL	D	I	46

Algoritmos e Programação

□ Melhorando um algoritmo

- Tomando como exemplo uma operação simples de trocar uma lâmpada, podemos perceber uma série de detalhes que nos ajudam a compreender melhor o uso de um algoritmo e ainda perceber que podemos torná-lo mais eficiente.

ALGORITMO 1.1 Troca de lâmpada

- pegar uma escada;
 - posicionar a escada embaixo da lâmpada;
 - buscar uma lâmpada nova;
 - subir na escada;
 - retirar a lâmpada velha;
 - colocar a lâmpada nova.
-

- E se a lâmpada não estivesse queimada?

Algoritmos e Programação

□ Refinamento de um algoritmo

- Solução – Efetuar um teste. Uma nova versão do algoritmo seria:

ALGORITMO 1.2 Troca de lâmpada com teste

- pegar uma escada;
 - posicionar a escada embaixo da lâmpada;
 - buscar uma lâmpada nova;
 - acionar o interruptor;
 - se a lâmpada não acender, então
 - subir na escada;
 - retirar a lâmpada queimada;
 - colocar a lâmpada nova.
-

- Embora correto, o algoritmo ainda pode ser melhorado. Como?

Algoritmos e Programação

□ Refinamento de um algoritmo

- Testar antes de pegar a escada e a lâmpada nova.

ALGORITMO 1.3 Troca de lâmpada com teste no início

- acionar o interruptor;
 - se a lâmpada não acender, então
 - pegar uma escada;
 - posicionar a escada embaixo da lâmpada;
 - buscar uma lâmpada nova;
 - acionar o interruptor;
 - subir na escada;
 - retirar a lâmpada queimada;
 - colocar a lâmpada nova.
-

- E se a lâmpada nova não funcionar?

Algoritmos e Programação

□ Refinamento de um algoritmo

□ Usar um teste seletivo com repetição

ALGORITMO 1.4 Troca de lâmpada com teste e repetição indefinida.

- acionar o interruptor;
- se a lâmpada não acender, então
 - pegar uma escada;
 - posicionar a escada embaixo da lâmpada;
 - buscar uma lâmpada nova;
 - acionar o interruptor;
 - subir na escada;
 - retirar a lâmpada queimada;
 - colocar a lâmpada nova;
 - se a lâmpada não acender, então
 - retirar a lâmpada queimada;
 - colocar outra lâmpada nova;
 - se a lâmpada não acender, então
 - retirar a lâmpada queimada;
 - colocar outra lâmpada nova;

□ ... E não para mais?

Algoritmos e Programação

□ Refinamento de um algoritmo

- Usar uma condição de parada.

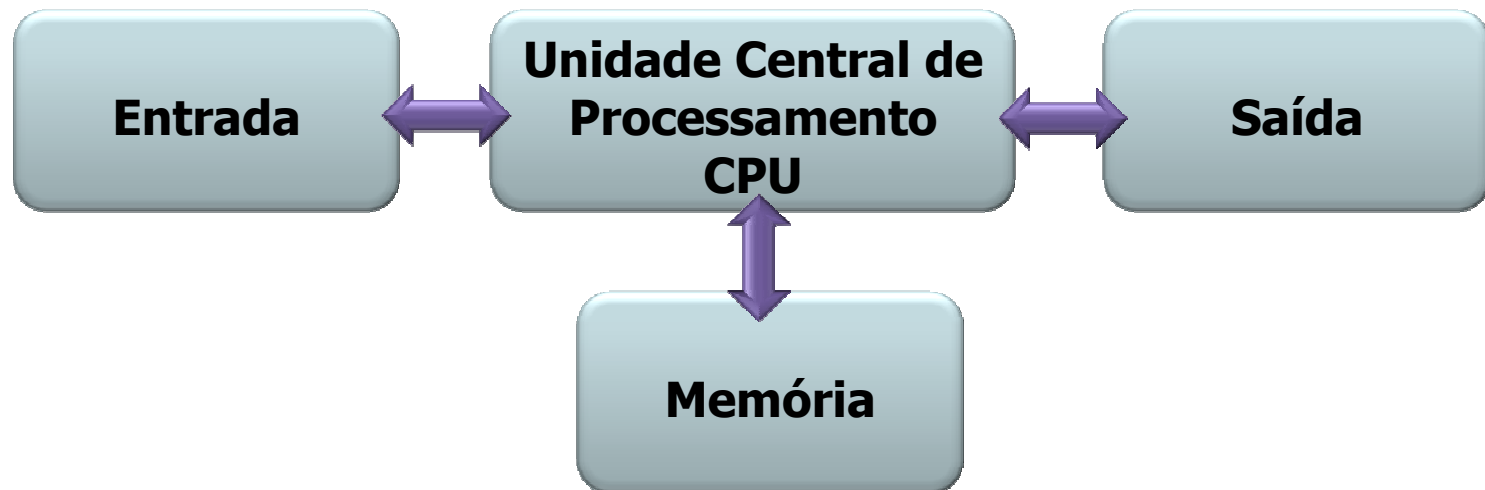
ALGORITMO 1.5 Troca de lâmpada com teste e condição de parada

- acionar o interruptor;
 - se a lâmpada não acender, então
 - pegar uma escada;
 - posicionar a escada embaixo da lâmpada;
 - buscar uma lâmpada nova;
 - acionar o interruptor;
 - subir na escada;
 - retirar a lâmpada queimada;
 - colocar uma lâmpada nova;
 - enquanto a lâmpada não acender, faça
 - retirar a lâmpada queimada;
 - colocar uma lâmpada nova;
-

- O número de repetições é indefinido, porém é finito.

Conceitos Básicos

- Conceitos básicos para construção de algoritmos.
 - Modelo de Von Neumann



Conceitos Básicos



- ❑ Os dados e os programas são armazenados na memória, em regiões distintas.
- ❑ Os programas são formados, essencialmente, por comandos (instruções sobre o que fazer);
- ❑ Os comandos são lidos sequencialmente da memória, um após o outro;
- ❑ A execução de um novo comando inicia apenas depois que a execução do anterior tiver terminado (execução sequencial).
- ❑ Eventualmente, um comando pode modificar o valor de um dado existente na memória, solicitar novos dados ao usuário ou enviar dados para a saída.

Conceitos Básicos

- **Conceitos básicos para construção de algoritmos**
 - Constantes;
 - Variáveis;
 - Identificadores;
 - Palavra-reservada.
- **Constantes:** São valores fixos, que não podem ser alterados pelas instruções do algoritmo, ou seja, é um espaço de memória cujo valor não deve ser alterado durante a execução de um algoritmo.

Exemplos:

Inteiro → 10, -23768, ...

Real → -2.34, 0.149, ...

Caractere → “k”, “computador”

Conceitos Básicos

- ❑ **Variáveis:** Elementos de dado cujo valor pode ser modificado ao longo de sua execução.
- ❑ São espaços alocados na memória que recebeu um nome (identificador) e pode ter tipo (inteiro, caractere e real), armazena um valor que pode ser modificado durante a execução do algoritmo.
- ❑ Um programa pode manipular várias variáveis distintas;
- ❑ Cada variável pode armazenar vários valores, mas apenas um de cada vez;
- ❑ “Variáveis” são criadas no início da execução do programa e destruídas ao término da sua execução;
- ❑ O conjunto de “variáveis” que um programa necessita precisa ser definido antes de se iniciar a execução do programa.

Conceitos Básicos

- Regras para criar nomes de variáveis.
 - ▣ Os nomes das variáveis devem representar o que será guardado dentro dela.
 - ▣ O primeiro caractere de um nome deverá ser sempre alfabético.
 - ▣ Não podem ser colocados espaços em branco no nome de variáveis, usar o UNDERSCORE “_”.
 - ▣ A declaração de uma variável é feita no algoritmo informando o seu nome, seguido por : e terminado com o seu tipo.
- Exemplos:
 - ▣ a: inteiro
 - ▣ nome_do_aluno: caractere
 - ▣ sinalizador: logico
 - ▣ valor1, valor2: real

Conceitos Básicos

- Palavras reservadas (palavras-chave):
 - São identificadores predefinidos que possuem significados especiais para o interpretador do algoritmo.

inicio	senao	para	repita
var	logico	se	ate
faca	inteiro	caractere	real

- Algumas das palavras reservadas definem os tipos de dados.

Conceitos Básicos



□ Tipos de dados

- ▣ Toda “variável” precisa estar associada a algum “tipo” de dados;
- ▣ O “tipo” de uma variável determina a coleção finita de valores que podem ser atribuídos à mesma;
- ▣ O “tipo” de uma variável é fixo durante toda a execução do programa.
- ▣ Os “tipos” de todas as “variáveis” precisam ser definidos antes de se iniciar a execução do programa.

Conceitos Básicos



□ Tipos Primitivos

- **logico** - define variáveis do tipo booleano, ou seja, com valor VERDADEIRO ou FALSO.
- **caractere** – define variáveis do tipo string, ou seja, cadeia de caracteres.
- **inteiro** - define variáveis numéricas do tipo inteiro, ou seja, sem casas decimais.
- **real** - define variáveis numéricas do tipo real, ou seja, com casas decimais.

Conceitos Básicos



□ Declarações

- ▣ Seqüência de instruções que servem para informar quais variáveis estarão sendo usadas pelo programa e quais os seus respectivos tipos;
- ▣ Não é possível mudar o tipo de uma variável durante a execução do programa;
- ▣ Não é possível criar ou destruir variáveis durante a execução do programa;
- ▣ Tudo precisa ser planejado antes – durante a elaboração do algoritmo.

Conceitos Básicos

□ Comandos

- ▣ Determinam quando e quais ações “primitivas” devem ser executadas;
- ▣ São exemplos de ações “primitivas”: leitura de dados, saída de dados, atribuição de valor a uma variável;
- ▣ Além disso, os comandos podem ser “estruturados”;
- ▣ A “estruturação” dos comandos permite que eles sejam executados numa determinada ordem, que a sua execução seja repetida ou que se opte pela escolha de um ou outro comando subordinado.
- ▣ Basicamente, a “estruturação” dos comandos permite o estabelecimento de um “fluxo de controle”, ou seja, uma seqüência de execução de ações primitivas através do qual se pretende alcançar a solução do problema original.