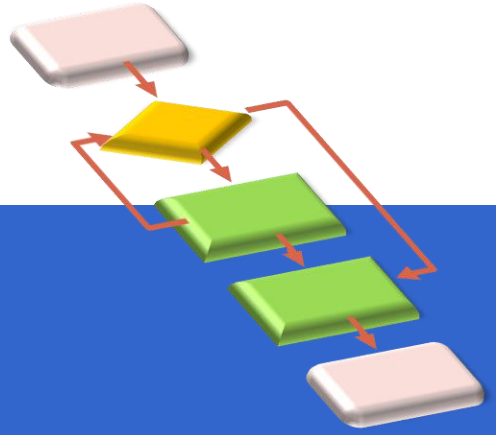


Algoritmos I

Manipulação de Strings

Prof.ª DSc. Vanessa de Oliveira Campos

Manipulação de Strings

- Uma *string* consiste de uma cadeia de caracteres.

- Formas de manipular *strings*:
  - como uma variável simples;
  - como um vetor de caracteres.

Manipulação de Strings

Como uma variável simples

- podem ser realizadas as operações básicas, como preenchimento por leitura, impressão de seu conteúdo, atribuição de valores e testes.

Manipulação de Strings

- **Como uma variável simples** - Exemplo:

```
Algoritmo "Teste"
VAR
        nome: string[20]
        endereço: string
inicio
    . . .
    Leia (nome)
    Leia (endereco)
    Se nome = 'João Dias' entao
        Escreva (endereco)
    fimse
    . . .
fimalgoritmo
```

Aqui foi restringido a qtd de carateres

Observeque foi possível colocar a string num laço

Manipulação de Strings

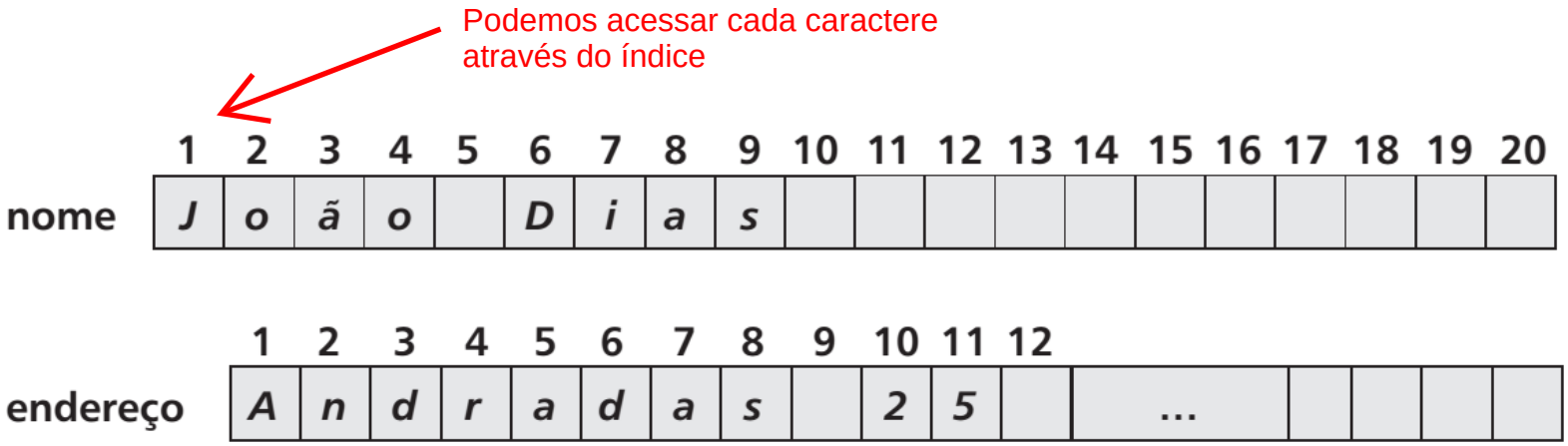
Como vetor de caracteres

- permite que se tenha acesso a cada um dos caracteres da string da mesma forma como é feito o acesso a elementos de um vetor.

* Podemos acessar caractere por caractere de forma isolada, da mesma forma como já fazemos com os vetores

Manipulação de Strings

Como vetor de caracteres



Manipulação de Strings

- **Como vetor de caracteres** - Exemplo:

```
Algoritmo "Teste"
VAR
        nome: string[20]
        endereço: string
        i: inteiro
inicio
    . . .
    i ← 1
    repita
        se nome[i] = 'x' entao
            nome[i] ← 'z'
        fimse
        i ← i + 1
    até i = 21
    escreva(nome)
    . . .
fimalgoritmo
```

Neste exemplo, está trocando cada 'x' por 'z'

nome := 'z' : altera-se toda a cadeia
nome[i] := 'z': altera-se somente uma determinada posição.

Tamanho de uma String

- Na declaração de uma variável do tipo *string* define-se seu tamanho máximo.

```
Algoritmo "Teste"
VAR
        nome: string[20]
        endereço: string
inicio
    . . .
```

- O tamanho padrão de uma *string* geralmente é de 256 caracteres.

Tamanho de uma String

- Entretanto, durante a execução de um programa, não necessariamente todo esse espaço declarado estará ocupado.

- O comprimento efetivo é fornecido por uma função:

*É a quantidade de caracteres de fato utilizada

* Os espaços não utilizados são automaticamente preenchidos e deixados em branco. [Não fica vazio]

compr (<nome da variável string>)

Tamanho de uma String

- O comprimento efetivo de uma variável do tipo string é controlado automaticamente pela linguagem de programação utilizada.
- Entretanto, sempre que a variável for manipulada como um vetor de caracteres e tiver seu tamanho efetivo alterado, o controle automático de seu comprimento será perdido, devendo sua atualização ser feita pelo programador.
- Para isso, o programador precisará saber onde é armazenado o valor do comprimento efetivo.

Strings em Pascal

- Uma *string* em Pascal é armazenada na forma de um vetor de caracteres de 256 posições, indexadas de 0 a 255.
- Esse vetor compreende posições para armazenamento dos caracteres, nas posições de índices de 1 a 255, mais a posição zero, em que fica armazenado o número de posições efetivamente ocupadas a cada momento.

* Essa posição 0 serve para quardar o tamanho da string.

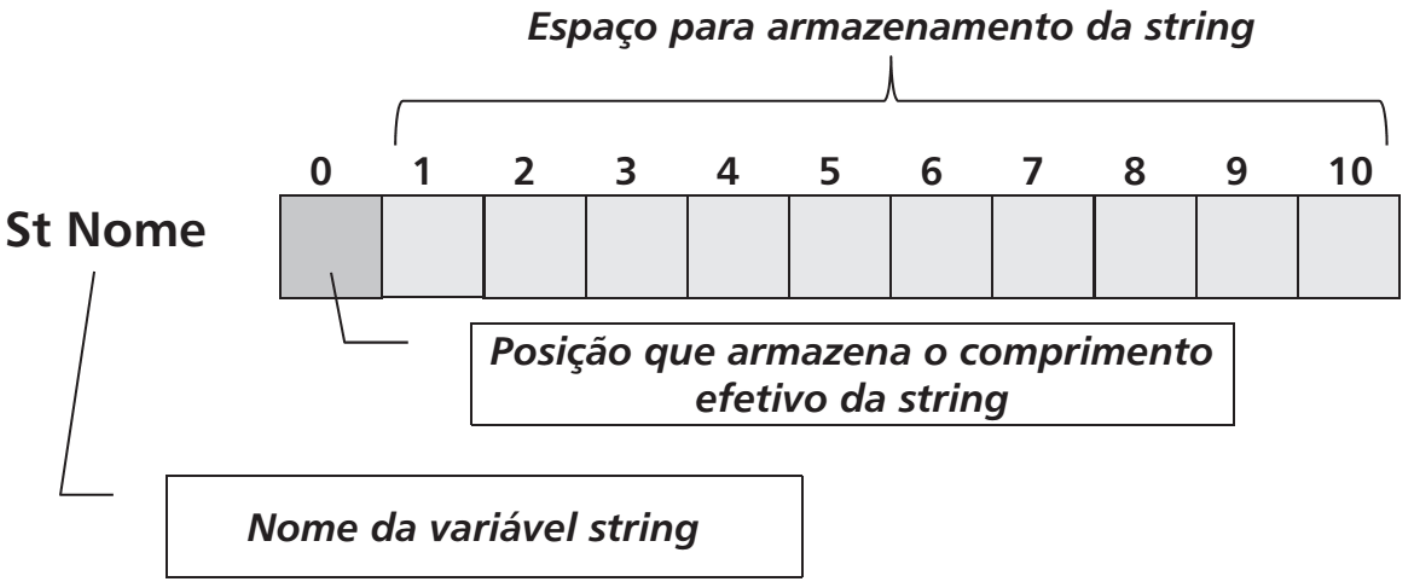
Strings em Pascal

- Na **posição zero** é armazenado um caractere cujo número de posição na tabela ASCII é igual ao número de caracteres efetivamente ocupados pela string no vetor.

* Tabela ascii, é uma tabela padrão que contém todos os caracteres possíeis da linguagem e seus respectivos códigos.
Isto serve para o comptador funcionar, uma vez que a máquina trabaha com numeros

Strings em Pascal

Exemplo: Estrutura de uma string de 10 caracteres



Strings em Pascal

Declaração

```
var
    texto: string;
    nome: string[36];
```

Comprimento Efetivo de Strings

- O comprimento efetivo de uma string é fornecido pela função:

  length (<var_str>)

onde <var_str> é o nome de uma variável ou constante do tipo string.

A função retorna um valor inteiro igual ao número de posições efetivamente ocupadas da string, não incluída a posição que armazena o seu comprimento efetivo.

Comprimento Efetivo de Strings

Atribuição de uma cadeia de caracteres à string.

nome := ‘Maria’;

- Nesse caso, o sistema atualiza automaticamente a posição zero da mesma.

Comprimento Efetivo de Strings

Manipulação da string como um vetor de caracteres.

- Ao manipular uma string como uma cadeia de caracteres, o controle do seu comprimento passa a ser de responsabilidade do usuário.

* O sistema não atualiza automaticamente quando a string é manipulada

* Lembrando que String é sequencial. O sistema sempre vai procurar o primeiro indice

Comprimento Efetivo de Strings

Exemplo:

```
var
      linha : string[80];
      i, num : integer;
. . .
for i:=1 to num do
      linha[i] := '*';
linha[0] := chr(num);   {AJUSTE DA POSICAO ZERO DA STRING}
```

Comprimento Efetivo de Strings

Utilização dos procedimentos *insert* e *delete*

- Os procedimentos insert e delete permitem aumentar ou reduzir o tamanho de uma variável string, enquanto ajustam automaticamente o seu comprimento.

Insert

- Permite inserir caracteres em uma string.

insert (<insertStr>, <targetStr>, <pos>)

O que eu quero incluir

Nome da variável que eu vo incluir

A partir de qual posição

Exemplo:
nome := 'Maria';
insert (' da Silva',nome,6)

* Será a partir da sexta posição

Resultado:
'Maria da Silva'

onde

<*targetStr*> (uma variável string) é a string que sofrerá a inserção;

<*insertStr*> (uma expressão string) é a substring com os caracteres a inserir;

<*pos*> (uma expressão inteira) corresponde à posição a partir da qual deverá ocorrer a inserção na string-alvo.

Delete

- Elimina um determinado número de caracteres de uma string, a partir de uma certa posição.

Exemplo:
nome:= Maria da Silva
delete(nome, 7, 3)

delete (<targetStr>, <pos>, <numChars>)

Nome da variável

A partir de qual posição

Quantidade de caracteres que sofrerão a ação

Resultado:
'Maria Silva'

onde

<*targetStr*> (uma variável string) é a string-alvo, ou seja, a string da qual serão eliminados caracteres;

<*pos*> (uma expressão inteira) é a posição a partir da qual deverá ocorrer a eliminação de caracteres na string-alvo;

<*numChars*> (uma expressão inteira) é o número de caracteres a eliminar.

Comparação entre Strings

- A comparação entre strings é feita caractere a caractere, a partir do início.
- Se um único caractere for diferente na mesma posição das duas strings, o teste resultará falso:

*É necessário que seja exatamente igual
(Incluindo as maiúsculas e minúsculas)

```
if nome_entrada = nome_compara then
    {...}
```

- Neste caso, devem ser totalmente coincidente mesmo na questão de maiúsculas e minúsculas.

Conversão de Tipos – STR

- Este procedimento recebe um valor numérico e devolve a string correspondente.

  * 123 será convertido para '123'

  str(<inNum>, <strVal>);

onde

<inNum> (expressão inteira ou real) é o valor numérico recebido;

<strVal> (variável do tipo string) é a string equivalente, resultante da execução do procedimento.

Conversão de Tipos – VAL

- Este procedimento recebe uma string supostamente numérica e tenta convertê-la para o seu valor numérico correspondente.

* '123' será convertido para 123

val(<inStr>, <numVar>, <codeVar>);

onde

<*inStr*> (expressão string) é a string a ser convertida;

<*numVar*> (variável do tipo inteiro ou real) é onde deve ser armazenado o valor numérico resultante da conversão;

<*codeVar*> (variável inteira) devolve o resultado da tentativa de conversão. 0, se bem-sucedida, ou um valor numérico correspondente à posição do primeiro caractere não numérico encontrado se malsucedida.

* Se tentar converter '141A', ele vai retornar 1414, de acordo com a devida posição

CONCAT

- A função concat concatena strings, ou seja, produz uma nova string formada pela concatenação de todas as strings que lhe são fornecidas como parâmetros, na ordem em que são fornecidas.

  concat (<string1>, <string2>, ..., <stringn>)

onde

  <string1> a <stringn> são expressões string.

- Exemplo:

  * É como se fosse uma adição de strings. Mas strngs não se adiciona, concatena

  nomeCompletoComTitulo := concat(titulo, primeiroNome, sobrenome);

  nomeCompletoComTitulo := titulo + primeiroNome + sobrenome;

COPY

- A função copy copia um trecho de uma string, devolvendo o trecho copiado como uma string.

  copy (<inStr>, <pos>, <numChars>)

onde

  <inStr> (expressão string) é a string de entrada;

  <pos> (expressão inteira) é a posição a partir da qual a substring produzida pelo copy deve começar a ser extraída;

  <numChars> (expressão inteira) é o número de caracteres que devem ser extraídos de inStr.

UPCASE

- Função que converte um caractere que representa uma letra minúscula para a representação da letra maiúscula correspondente.

upcase (<inChar>)

* Retorna o maiúsculo de um caractere

onde

<*inChar*> é uma expressão do tipo caractere.

UPCASE

- Exemplo:

```
if (upcase(continuar) = 'S') then
    {segue processando}
```

ORD

- Recebe como parâmetro um caractere e retorna o inteiro correspondente ao código ASCII referente ao caractere.

  ord (<inChar>);

  * Traz o número corespondente do caractere, de acordo com a tabela ascii

  onde

  <*inChar*> é uma expressão do tipo caractere.

- Exemplo:

```
writeln( 'O codigo ASCII para "c" = ', ord( 'c' ), ' decimal' );
```

CHR

- Recebe como parâmetro um inteiro e retorna o caractere ASCII correspondente ao código identificado com esse inteiro.

  chr(<numChar>)

  * Oposto do ord

  onde

  <*numChar*> é uma expressão do tipo inteiro.

- Exemplo:

```
. . .
var i: integer ;
Begin
    for i:= 32 to 126 do
        write( chr(i) );
End.
```

Exercícios

1. Uma palavra é dita palíndromo se ela puder ser lida da mesma forma nos dois sentidos de escrita. Faça um algoritmo que leia uma palavra do usuário e diga se ela é ou não palíndromo. Exemplos de palíndromos : mirim, mussum, arara, salas, osso.

Exercícios

2. Dado um literal origem, a ser lido do usuário, construa um literal destino que não contenha qualquer ocorrência de um determinado caractere x, também lido do usuário. Ao final, escreva o conteúdo da variável gerada.

Exemplo:

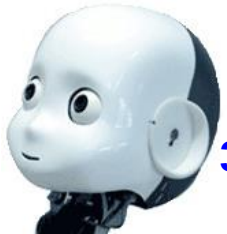
origem = Instituto da Computação

x = t

destino = Insiuo da Compuação

Exercícios

3. Faça um programa que conta as ocorrências de uma determinada string no interior de outra string.



Exercícios

4. Faça um programa que leia uma cadeia de caracteres e transforme os caracteres todos em letras maiúsculas. Use apenas o conceito de cadeia de caracteres e tabela ASCII para resolver esse problema.