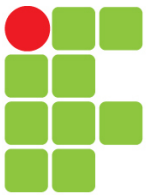
Vetores e Strings

Programação de computadores I

Prof. Renan Augusto Starke

Instituto Federal de Santa Catarina – IFSC
Campus Florianópolis
renan.starke@ifsc.edu.br

1 de junho de 2016



INSTITUTO FEDERAL
SANTA CATARINA

Ministério da Educação
Secretaria de Educação Profissional e Tecnológica
INSTITUTO FEDERAL DE SANTA CATARINA

Tópicos da aula

Introdução

Objetivos

- Introduzir o conceito de vetores em C
- Apreender a trabalhar com string
- Referências:
  - Livro: SCHILDT, H. C Completo e Total.
  - Internet: http://www.cplusplus.com/

Definição

Vetor

É uma coleção de variáveis do mesmo tipo referenciado por um nome comum. Em C, vetores são dispostos em regiões contínuas de memória. Também é conhecido como matriz unidimensional ou *array*.

Declaração

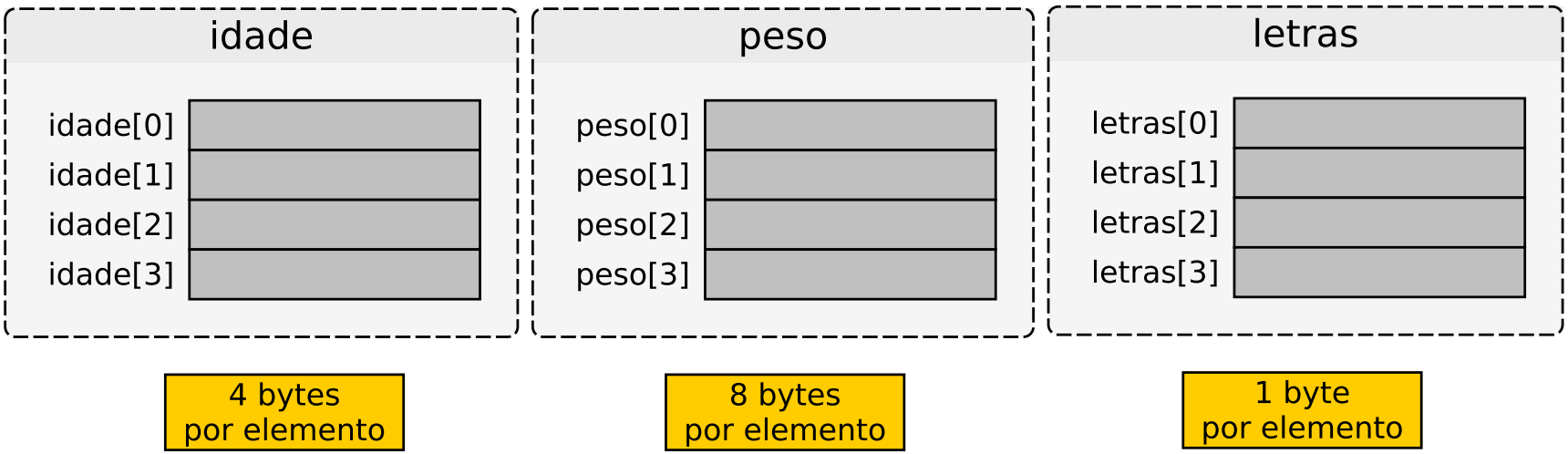
```
tipo nome_var[tamanho];
```

- Vetor é uma variável qualquer: ele deve ser explicitamente declarado.
- *tipo*: tipo do elemento do vetor: int, float, double, char, ...

Declaração

Exemplos

```
int idade[4];
double peso[4];
char letras[4];
```



- Acesso a cada elemento é por colchetes: *idade*[0]
- Índice sempre inicia em **0**

Exemplo

```c
#include <stdio.h>

int main()
{
  int k;
  float a[3];

  // Vetor de float inicializado pelo usuario
  for (k = 0; k < 3; k = k + 1){
    printf("Forneca um numero: ");
    scanf("%f", &a[k]);
  }

  //Imprime todos os valores
  for (k = 0; k < 3; k = k + 1)
    printf("%f\n", a[k]);

  return 0;
}
```

- Índice nunca pode ser maior que o tamanho declarado.
- $a[4]$ ou $a[> 3]$ pode abortar o programa ou causar erro semântico.
- **Não há índices negativos**.

Declaração com inicialização

- Igualmente a variáveis, vetores podem ser inicializados em sua declaração.
- Usa-se { }.

Inicialização

```
int idade[3] = {29, 43, 5};
float peso[3] = {32.1, 89.9, 72.5};
char letras[3] = {'a', 'b', 'c'};
```

- **Separador de elementos é a , (vírgula). O . (ponto) é operador decimal.**

Declaração com inicialização

- Igualmente a variáveis, vetores podem ser inicializados em sua declaração.
- Usa-se { }.

Inicialização

```
int idade[3] = {29, 43, 5};
float peso[3] = {32.1, 89.9, 72.5};
char letras[3] = {'a', 'b', 'c'};
```

- **Separador de elementos é a , (vírgula). O . (ponto) é operador decimal**.

Quanto é *idade*[2] ?
Quanto *letras*[4] ?

Vetores multidimensionais

Vetor multidimensional

Vetor multidimensional pode ser considerado como uma matriz.

```
// Declaracao de vetores bidimensionais
int matriz[3][3];
int notas[2][3];
```

```
// Declaracao com inicializacao de
// vetores multidimensionais
int notas[2][3] = { {5, 6, 7}, {3, 9, 10} };
```

Vetores multidimensionais

Exemplo

```
int main()
{
    int i = 0;
    int j = 0;

    int notas[10][3];

    for (i=0; i < 10; i++)
    {
        printf("Aluno: %d\n", i);
        for (j=0; j < 3; j++){
            printf("Nota[%d]: ", j);
            scanf("%d", &notas[j][i]);
            printf("\n");
        }
    }

    return 0;
}
```

Strings

Definição

String é uma sequencia de caracteres terminados por um marcador especial final. Nas linguagens de programação, uma string é geralmente terminada pelo caractere '\0'. A linguagem C não possui um tipo especial para string. Estas são construídas por vetores do tipo **char**.

```
// Declaracao de strings:
// vetores tipo char
char nome[30];

char titulo[] = "C completo e total";

char nomes[2][20] = { "Joao", "Maria" };
```

Tabela ASCII

Tabela ASCII (códigos de caracteres 0 - 127)

000	016 ▶	032	048 0	064 @	080 P	096 `	112 p
001 ☺	017 ◀	033 !	049 1	065 A	081 Q	097 a	113 q
002 ☻	018 ↕	034 "	050 2	066 B	082 R	098 b	114 r
003 ♥	019 ‼	035 #	051 3	067 C	083 S	099 c	115 s
004 ♦	020 ¶	036 $	052 4	068 D	084 T	100 d	116 t
005 ♣	021 §	037 %	053 5	069 E	085 U	101 e	117 u
006 ♠	022 ▬	038 &	054 6	070 F	086 V	102 f	118 v
007	023 ↨	039 '	055 7	071 G	087 W	103 g	119 w
008	024 ↑	040 (	056 8	072 H	088 X	104 h	120 x
009	025 ↓	041)	057 9	073 I	089 Y	105 i	121 y
010	026 →	042 *	058 :	074 J	090 Z	106 j	122 z
011 ♂	027 ←	043 +	059 ;	075 K	091 [	107 k	123 {
012 ♀	028 ∟	044 ,	060 <	076 L	092 \	108 l	124 |
013	029 ↔	045 -	061 =	077 M	093]	109 m	125 }
014 ♫	030 ▲	046 .	062 >	078 N	094 ^	110 n	126 ~
015 ☼	031 ▼	047 /	063 ?	079 O	095 _	111 o	127 ⌂

- Qual o valor de 'a'?
- Q letra é o número 118?

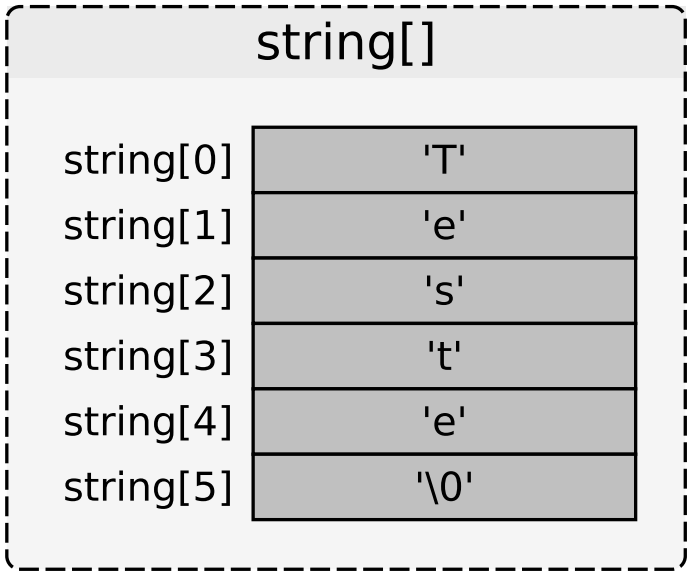
Tabela ASCII

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: www.LookupTables.com

Strings

```
char string[] = "Teste";
```



- Uma string **SEMPRE** deve conter o caractere '\0'.

Funções de manipulação de strings

- O compilador, além de fornecer funções de entrada e saída, fornece também funções de manipulação de strings.
- Incluir string.h

```
#include <string.h>
```

- Funções disponíveis:
  - cópia
  - comparação;
  - concatenação;
  - tamanho;
  - busca de substrings;
  - `http://www.cplusplus.com/reference/cstring/?kw=string.h`

Strings – cópia

strncpy

```
char * strncpy ( char * destination, const char * source, size_t num );
```

Parâmetros de entrada:

- *destination*: vetor de destino
- *source*: vetor que será copiado
- *num*: número máximo de caracteres a ser copiado (geralmente utilizamos o tamanho do destinho)

Retorno da função:

- retorna o destino

OBS: Caso o valor máximo é atingido, o caractere de final de string NÃO é adicionado.

Exemplo

strncpy

```c
#include <stdio.h>
#include <string.h>

int main ()
{
  char str1[]= "To be or not to be";
  char str2[40];
  char str3[40];

  /* copia de str1 pata str2: maximo de 40 caracteres */
  strncpy ( str2, str1, 40 );

  /* copia parcial (apenas 5 caracteres): */
  strncpy ( str3, str2, 5 );
  str3[5] = '\0';   /* Final de string adicionado manualmente */

  puts (str1);
  puts (str2);
  puts (str3);

  return 0;
}
```

```
To be or not to be
To be or not to be
To be
```

Strings – comparação

strcmp

```
int strcmp ( const char * str1, const char * str2 );
```

Parâmetros de entrada:

- *str1*: 1a string a ser comparada
- *str2*: 2a string a ser comparada

Retorno da função:

- < 0 ou > 0: strings são diferentes
- 0: strings são idênticas

Exemplo

strcmp

```c
#include <stdio.h>
#include <string.h>

int main ()
{
  char key[] = "laranja";
  char buffer[80];

  do {
     printf ("Adivinhe minha fruta predileta? ");

     scanf ("%79s",buffer);

  } while (strcmp (key,buffer) != 0);

  puts ("Resposta correta!");
  return 0;
}
```

```
Adivinhe minha fruta predileta? laranja
Resposta correta!
```

Strings – concatenação

strncat

```
char * strncat ( char * destination, const char * source, size_t num );
```

Parâmetros de entrada:

- *destination*: destino da concatenação
- *source*: fonte
- *num*: número máximo de caracteres concatenados

Retorno da função:

- retorna string destino

Exemplo

strncat

```c
#include <stdio.h>
#include <string.h>

int main ()
{
  char str1[20];
  char str2[20];

  strncpy (str1,"Ser ", 20);
  strncpy (str2,"ou nao ser", 20);
  strncat (str1, str2, 6);

  puts (str1);

  return 0;
}
```

```
Ser ou nao
```

Strings – tamanho

strlen

```
size_t strlen ( const char * str );
```

Parâmetros de entrada:

- *str*: string que se deseja conhecer o tamanho

Retorno da função:

- tamanho da string

Exemplo

strlen

```c
#include <stdio.h>
#include <string.h>

int main ()
{
  char szInput[256];

  printf ("Entre com uma sentenca: ");
  fgets(szInput, 256, stdin);

  printf ("A sentenca tem %d catacteres.\n", (int)strlen(szInput));
  return 0;
}
```

```
Entre com uma sentenca: Teste de sentenca
A sentenca tem 18 catacteres.
```

Strings – busca de substrings

strstr

```
char * strstr (char * str1, const char * str2 );
```

Parâmetros de entrada:

- *str1*: string em que se deseja buscar a substring
- *str2*: string buscada

Retorno da função:

- Início da string encontrada
- NULL se não encontrou substring

Exemplo

strstr

```c
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[] = "Este eh um emxemplo de string";
  char * pch;
  pch = strstr (str,"string");

  if (pch != NULL)
    puts (pch);

  return 0;
}
```

```
string
```

Exercícios

- Escreva um programa que receba 10 valores reais (float) fornecidos pelo usuário e depois apresente o maior valor entre todos.
  - Cada valor fornecido pelo usuário deve ser armazenado como um elemento de um vetor.
  - **Dicas:**
    - Criar um vetor com 10 elementos do tipo float.
    - Utilizar uma estrutura de repetição (for, while ou do-while) para receber os valores digitados pelo usuário.
    - Utilizar uma segunda estrutura de repetição (for, while ou do-while) para fazer a varredura no vetor, procurando pelo maior valor.

Exercícios

- Faça um programa em C que some duas matrizes 3x3.
- Faça um programa em C que multiplique duas matrizes 5x5.
- Faça um programa que imprima os caracteres correspondentes da tabela ASCII de 33 a 126.
- Escreva um programa que recebe uma string e um caractere e apague todas as ocorrêncidesses caractere na string.

Exercícios

- Faça um programa em C que receba uma string e mostre na tela quantas vezes aparece cacaractere.
  Ex.: "Essa é a string digitada"
  O caractere 'E' aparece 1 vez;
  O caractere 's' aparece 3 vez;
  O caractere 'a' aparece 4 vez;
  O caractere 'é' aparece 1 vez;
  O caractere 't' aparece 2 vez;
  O caractere 'r' aparece 1 vez;
  O caractere 'i' aparece 3 vez;
  O caractere 'n' aparece 1 vez;
  O caractere 'g' aparece 2 vez;
  O caractere 'd' aparece 2 vez;
  O caractere ' ' aparece 4 vez; // entenda-se o espaço
- Faça um programa que retire todos os caracteres repetidos de uma string.