Introdução à Ciência da Computação

Arranjos: Vetores, Strings & Matrizes em C

Prof. Ricardo J. G. B. Campello

Agradecimentos

- Parte dos slides a seguir são adaptações dos originais gentilmente cedidos por:
  - Prof. Rudinei Goularte
  - Prof. André C. P. L. F. Carvalho

Sumário

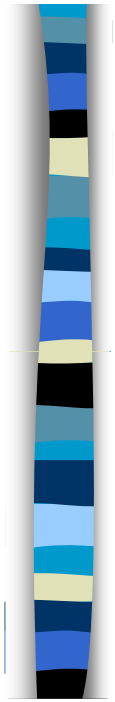
- Vetores
- Strings em C
- Matrizes
- Arranjos Multi-Dimensionais

Estruturas de Dados

- Os tipos primitivos (char, int, float, etc) não são suficientes para representar todos os tipos de informação
- Isso é particularmente verdadeiro quando temos mais de uma informação relacionada
  - Por exemplo: lista dos nomes dos alunos de uma sala, endereço de alguém, etc.
- Utilizaremos os tipos primitivos para construir outras estruturas de dados mais complexas

Estruturas Compostas

- Pode-se organizar os dados dos tipos simples em tipos mais complexos, formando o que se denomina de **Estruturas Compostas**
- Exemplos:
  - Estruturas compostas **homogêneas** unidimensionais (**vetores**) e multidimensionais (**matrizes**)
    - Permitem a manipulação de um conjunto de dados de um mesmo tipo primitivo
  - Estruturas compostas **heterogêneas**: **registros**
    - Permitem a manipulação de um conjunto de dados de diferentes tipos primitivos (veremos na aula seguinte...)



Vetor

- **Definição: variável composta** dada por uma coleção de elementos individuais com as seguintes características:
  - É **ordenado**: os elementos de um vetor são indexados de forma ordenada
  - É **homogêneo**: Todo valor armazenado em um mesmo vetor deve ser do mesmo tipo
    - Por exemplo, um vetor de inteiros só pode ter elementos do tipo inteiro

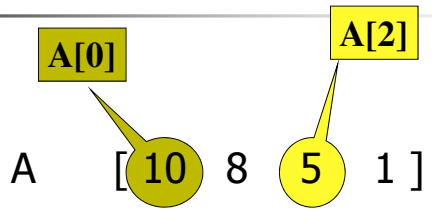
Vetores

- Exemplo de vetor de inteiros com 4 elementos, cujo identificador (nome) é A:
  - A

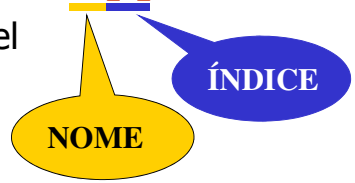
10	8	5	1

- Para fazer referência a um determinado elemento do vetor usa-se um **índice**
- Dependendo da linguagem, o índice j está associado ao j-ésimo ou (j+1)-ésimo elemento do vetor
- No 2º caso, por ex., tem-se para o vetor acima:
  - A[1] faz referência ao segundo elemento do vetor
  - O valor de A[1] é igual a 8

Vetores



- Genericamente, um elemento do vetor é representado por: **A[i]**
  - Trata-se de uma variável como qualquer outra



Vetor

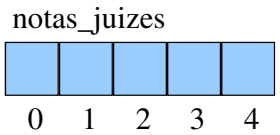
- Pode-se pensar em um vetor como uma seqüência de células, uma para cada elemento:

- Também denominado **arranjo** 1D
- Possui duas propriedades fundamentais:
  - Tipo de elemento
  - Tamanho do vetor

Vetor

- Declaração em C
  - **tipo** identificador[tamanho];
  - Exemplo: **int** vet[10];
    - Vetor de inteiros denominado vet, com 10 elementos: vet[0], vet[1], ..., vet[9]

Vetor



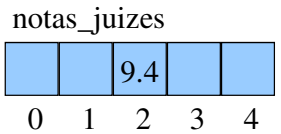
- Para se referir a um elemento específico de um vetor, devem ser fornecidos:
  - nome do vetor e o índice correspondente
- Exemplo:

  double *notas_juizes[5];*

  - A nota do 2o juiz é dada por *notas_juizes[1]*

Vetor

- Células são variáveis
  - Ex.: notas_juizes[2] = 9.4;
  - É importante distinguir entre índice de um elemento e valor de um elemento



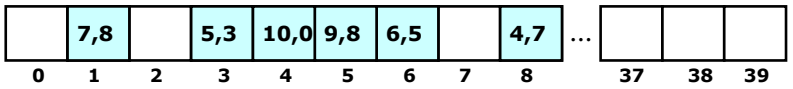
Índice = 2
Valor = 9.4

Vetores

- Exemplos de Manipulação:

```
void main(void){
     float VCLASSE[40];
     int A = 5;
     VCLASSE[6] = 6.5;
     VCLASSE[1] = 7.8;
     scanf("%f", &VCLASSE[8]); /* Suponha que entrada foi 4.7 */
     VCLASSE[3] = 5.3;
     VCLASSE[4] = VCLASSE[3] + VCLASSE[8];
     VCLASSE[A] = 9.8;
}
```

VCLASSE



Vetores

Exemplo: No. de notas acima da média usando vetores

```
#include <stdio.h>
void main(void){
  float NotasClasse[10];
  int NotaAcima, X;
  float Soma, Media;
  Soma = 0;
  NotaAcima = 0;
  for(X=0; X<=9; X++){
          scanf("%f", &NotasClasse[X]);
          Soma = Soma + NotasClasse[X];
  }
  Media = Soma / 10;
  for(X=0; X<=9; X++){
          if ( NotasClasse[X] > Media ) NotaAcima = NotaAcima + 1;
  }
  printf("%d", NotaAcima);
}
```

Exemplo

- A **soma** de dois vetores numéricos A e B de mesma dimensão é um terceiro vetor de mesma dimensão cujos elementos são dados pelas somas individuais dos respectivos elementos de A e B, um a um. Por exemplo, [2 3.5 4] + [–5 0 2] = [–3 3.5 6]
- Pede-se:
  - Faça um programa em C que leia dois vetores de 20 números reais cada (valores inseridos pelo usuário), calcule e escreva o vetor soma resultante

Exemplo

- Solução:

```
#include <stdio.h>
void main(void){
     float A[20], B[20], C[20];
     int cont;
     for(cont=0; cont<=19; cont++){
          scanf("%f %f", &A[cont], &B[cont]);
          C[cont] = A[cont] + B[cont];
          printf("C[%d] = %f\n", cont, C[cont]);
     }
}
```

Exercícios

- Dado o vetor V de inteiros abaixo (com seus índices)

e as variáveis X = 1 e Z = 5, pede-se:

- V[7]; V[X + 1]; V[X + Z];
- V[Z*2–X]; V[Z*X–2]; V[V[1]];
- V[8–V[1]]; V[V[V[9] –V[7]]; V[V[V[V[5]]]];

Exercícios

- O **produto interno** de dois vetores numéricos A e B de mesmo tamanho é definido como a soma total dos produtos individuais de cada elemento de um vetor pelo elemento correspondente do outro vetor, um a um. Por exemplo, o produto interno de [1 3.7 4] e [-5 0 2.5] é 5
- Pede-se:
  - Faça um programa em C que leia dois vetores de 20 números reais cada (valores inseridos pelo usuário) e escreva o produto interno desses vetores

Exercícios

- O **produto de Kronecker** de dois vetores numéricos A e B de mesma dimensão é um terceiro vetor cujos elementos são dados pelos produtos individuais de todas as combinações possíveis de elementos de A e B, um a um. Por ex., o produto de Kronecker de [2 3.5 4] e [–5 0 2] é [–10 0 4 –17.5 0 7 –20 0 8]
- Pede-se:
  - Faça um programa em C que leia dois vetores de 10 números reais (valores inseridos pelo usuário) e escreva o produto de Kronecker desses vetores

Exercícios

- Um dos métodos mais simples (e ineficientes) de ordenar uma seqüência de N números consiste em efetuar N-1 varreduras pela seqüência efetuando trocas de elementos adjacentes que estejam fora da ordem desejada. Pede-se:
  - Implemente um programa utilizando esse método (denominado **método da bolha**) para encontrar o k-ésimo maior e o k-ésimo menor elementos de uma seqüência de 25 números inserida pelo usuário. O usuário deve entrar com o valor de k e os números devem ser armazenados em um vetor

Exercícios

- Faça um programa que leia três vetores de 10 elementos fornecidos pelo usuário, sendo os dois primeiros (A e B) de valores inteiros positivos e o terceiro (C) de caracteres. O programa deve escrever o resultado da operação indicada por cada elemento do vetor C aplicada aos operandos dados pelos respectivos elementos de A e B, nessa ordem, ou 'NULL' caso o elemento de C seja um caractere inválido. Os caracteres válidos são ('+', '–', '*', '/', 'div', 'mod', 'pot').

Exercícios

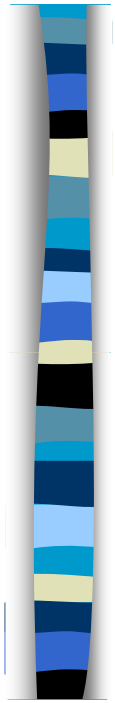
- Um **conjunto** pode ser definido como uma coleção de elementos "distintos" (no sentido que o mesmo elemento não ocorre mais de uma vez na coleção). Um conjunto de elementos, por exemplo, caracteres alfanuméricos, podem ser armazenados em um computador na forma de um vetor, onde cada elemento do conjunto está armazenado em uma célula do vetor

1. Faça um programa que:
   - Leia dois vetores A e B, cada um constituído de 10 **caracteres distintos** inseridos pelo usuário (de forma a constituírem um conjunto)
   - Construa um terceiro vetor C dado pela **união** de A e B
   - Incremente o algoritmo para calcular um vetor D dado pela **intersecção** de A e B

Exercícios

Continuação...

2. Faça agora um outro programa que:
   - Leia dois vetores A e B, cada um constituído de 10 **números reais distintos** (de forma a constituírem um conjunto), supostamente inseridos pelo usuário em **ordem crescente**
   - Construa um terceiro vetor C dado pela **união** de A e B, utilizando a propriedade de ordem dos vetores para fazer um algoritmo mais eficiente do que aquele do problema anterior
   - Incremente o algoritmo para calcular um vetor D dado pela **intersecção** de A e B. Novamente utilize a propriedade de ordem dos vetores para implementar um algoritmo mais eficiente



Vetor

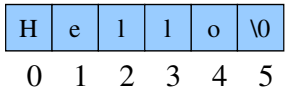
- Inicialização de vetores
  - Valores iniciais podem ser atribuídos a uma variável do tipo vetor quando da sua declaração
    - Ex.: ***int*** *digitos[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };*
    - Neste caso, o tamanho do vetor pode ser omitido
      - Ex.: ***int*** *digitos[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };*
      - Compilador conta o número de inicializadores e reserva a mesma quantidade de elementos
      - Programador não precisa contar (útil em strings)

Vetor

- A Linguagem C não checa se você passou dos limites de um vetor !
  - Se passar do fim de um vetor, você pode:
    - Escrever no espaço reservado para outras variáveis...
    - Escrever no espaço reservado para outros programas...
    - ...

Strings

- *Strings* são vetores de caracteres
  - Cada célula = 1 caractere = 1 byte
  - Final de *string* em C é indicado por '\0'
  - Declarações com inicialização:
    - **char** hello[] = {'H', 'e', 'l', 'l', 'o', '\0'}; ou
    - **char** hello[6] = {'H', 'e', 'l', 'l', 'o', '\0'}; ou
    - **char** hello[] = "Hello";
    - **char** hello[6] = "Hello";



Strings

- Em C, existem várias funções de entrada e saída voltadas especialmente para strings
- Para as funções que estamos utilizando:
  - **scanf** e **printf** utilizam o código de formato **%s** para ler e escrever strings
  - por padrão, **scanf** termina a leitura da string quando recebe quebra de linha (\n, ou seja, ENTER), tabulação ou espaço em branco
    - para ler strings com espaços em branco (p. ex. "Fulano de Tal"), outras funções são mais apropriadas (p. ex. **gets**)
  - **scanf** insere '\0' automaticamente ao final, mas deve haver espaço reservado para tal!

Exemplo

```
#include <stdio.h>
void main(void){
   char nome[11];
   printf("Entre com nome até 10 caracteres: ");
   scanf("%s", nome);
   printf("%s", nome);
}
```

Strings

- Bibliotecas de operações sobre strings
  - Biblioteca ANSI *string.h* para manipular strings
    - Biblioteca padrão da linguagem C
    - Fornece um conjunto de operações avançadas
      - Permite trabalhar com uma ou mais strings inteiras utilizando uma simples chamada de função

Strings

Funções mais comuns de string.h

Nome	Função
strncpy (s1, s2, ...)	Copia s2 em s1
strncat (s1, s2, ...)	Concatena s2 ao final de s1
strlen (s1)	Retorna o tamanho de s1
strncmp (s1, s2, ...)	Retorna 0 se s1 ==s2; menor que 0 se s1<s2; maior que 0 se s1>s2
strstr (s1, s2)	Retorna um ponteiro para a primeira ocorrência de s2 em s1

Exemplo

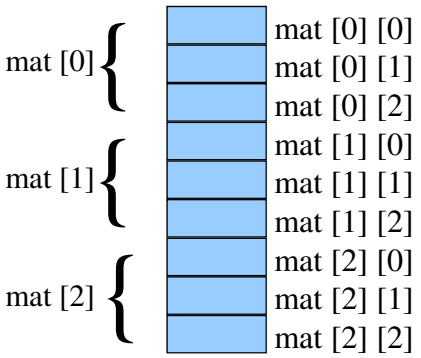
```
#include <stdio.h>
#include <string.h>
void main(void){
   char nome1[12] = "Jose",
        nome2[] = "Maria";
   strncpy(nome1, nome2, 12);
   printf("%s", nome1);
}
```


Arranjos Multi-Dimensionais

- Quando os elementos de um arranjo são arranjos
  - Arranjos bidimensionais (**matrizes**) são a forma mais comum
    - Vetor de vetores
  - Ex.: ***double*** *mat [3][3];*

mat[0] ➡	mat[0][0]	mat[0][1]	mat[0][2]
mat[1] ➡	mat[1][0]	mat[1][1]	mat[1][2]
mat[2] ➡	mat[2][0]	mat[2][1]	mat[2][2]

Matrizes



C trata *mat* como um vetor de três elementos

Cada elemento é, por sua vez, um vetor de três elementos

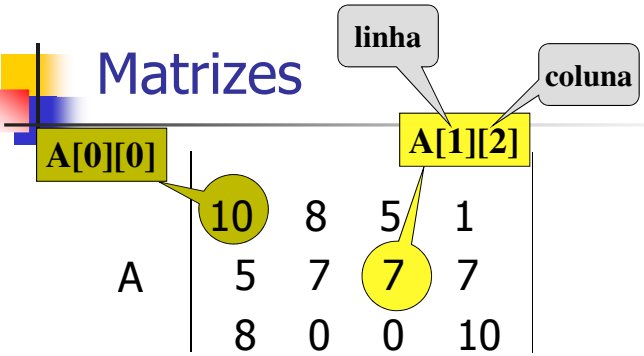
Na *memória*, estes nove valores são armazenados em sequência linear

Matrizes

A	10	8	5	1
	5	7	7	7
	8	0	0	10

- Para fazer referência a um determinado elemento da matriz usa-se dois índices:
  - o primeiro índice representa a linha
  - o segundo índice representa a coluna

Matrizes



- Genericamente, um elemento da matriz é representado por: **A[i][j]**

Matrizes

- Podem ser inicializadas na declaração
  - Para enfatizar a estrutura geral, valores de cada vetor interno são inicializados entre chaves

```
double ident [3][3] = {
        {1.0,   0.0,   0.0 },
        {0.0,   1.0,   0.0 },
        {0.0,   0.0,   1.0 }
        };
```

1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0

Matrizes

– Manipulação (Exemplo 1):

```
#include <stdio.h>
void main(void){
   int mat[4][4];
   int A = 2, B = 1;
   mat[0][1] = 7;
   mat[0][3] = 12;
   mat[1][0] = 10;
   mat[A+B][B] = 8;
   ...
}
```

mat

	0	1	2	3
0		7		12
1	10			
2				
3		8		

Matrizes & Strings

- Cada linha de uma matriz de caracteres é um vetor de caracteres, ou seja, é uma *string*. Logo, uma matriz de caracteres pode ser vista como um vetor de strings

```
char A[3][4];
A[1][2] = 'b';
A[2][0] = 'b';
A[2][1] = 'a';
A[2][2] = 'u';
A[2][3] = '\0';
printf("%s", A[2]);
...
```

A

	0	1	2	3
0				
1			b	
2	b	a	u	\0

Exemplo

```
#include <stdio.h>
void main(void){
   char str[3][11];
   for(i=0; i<=2; i++){
      printf("Entre com 1 string (max. 10 carac.): ");
      scanf("%s", str[i]);
   }
}
```


Exemplo

- Preencher uma matriz de 4 linhas e 3 colunas com os valores da tabela TAB. Os valores serão fornecidos pelo usuário

TAB

10	8	5
5	7	7
8	0	0
1	7	10

Exemplo

Teste de mesa

```
#include <stdio.h>
void main(void){
 int I, J;
 int TAB[4][3];
 for(I=0; I<=3; I++){
  for(J=0; J<=2; J++){
   printf("leia TAB[%d][%d] ", I, J);
   scanf("%d", &TAB[I][J]);
  }
 }
}
```

I	J	
0	0	leia TAB[0][0]
	1	leia TAB[0][1]
	2	leia TAB[0][2]
1	0	leia TAB[1][0]
	1	leia TAB[1][1]
	2	leia TAB[1][2]
2	0	leia TAB[2][0]
	1	leia TAB[2][1]
	2	leia TAB[2][2]
3	0	leia TAB[3][0]
	1	leia TAB[3][1]
	2	leia TAB[3][2]

Entrada: 10, 8, 5, 5, 7, 7, 8, 0, 0, 1, 7, 10

Exemplo

- A **soma** de duas matrizes numéricas A e B de mesma dimensão é uma terceira matriz de mesma dimensão cujos elementos são dados pelas somas individuais dos respectivos elementos de A e B, um a um

- Pede-se:
  - Faça um programa que leia duas matrizes com 5 linhas e 3 colunas (5 x 3) de números reais (valores inseridos pelo usuário), calcule e escreva a soma dessas matrizes

Solução

```
#include <stdio.h>
void main(void){
     float A[5][3], B[5][3], C[5][3];
     int i, j;
     for(i=0; i<=4; i++){
        for(j=0; j<=2; j++){
          scanf("%f %f", &A[i][j], &B[i][j]);
          C[i][j] = A[i][j] + B[i][j];
          printf("C[%d][%d] = %f\n", i, j, C[i][j]);
        }
     }
}
```

Exercício

- Faça um programa para ler os valores inteiros de uma tabela de M linhas e N colunas (M e N fornecidos pelo usuário, ambos limitados a, no máximo, 50), armazenando os elementos em uma matriz A. Use um acumulador para realizar a soma dos elementos da matriz à medida que esses elementos são lidos do teclado

Exercício

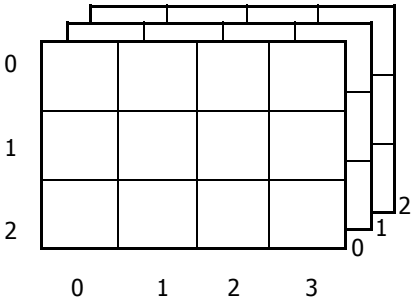
Sabe-se que uma **multiplicação** de uma matriz numérica A por outra matriz numérica B só é definida se o número de colunas de A for igual ao número de linhas de B. Sabe-se ainda que o resultado é uma terceira matriz com o mesmo número de linhas de A e mesmo número de colunas de B

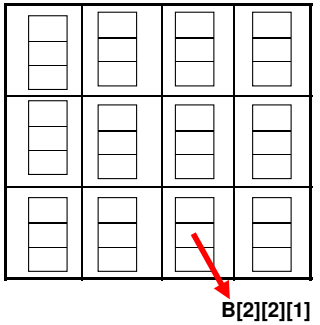
- Faça um programa que:
  - Leia duas matrizes com 5 linhas e 5 colunas (5 x 5) de números reais (valores inseridos pelo usuário)
  - Leia índices (inferiores e superiores) que delimitem submatrizes de A e B
  - Termine avisando o usuário se os índices forem inválidos, ou seja:
    - Não delimitam uma submatriz, ou
    - Não permitem a multiplicação da submatriz de A pela submatriz de B
  - Caso os índices sejam válidos, calcule e escreva a multiplicação da submatriz de A pela submatriz de B

Arranjos Multi-Dimensionais

- Exemplo de declaração

  int B[3][4][3];



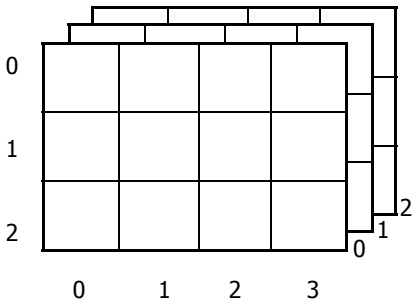


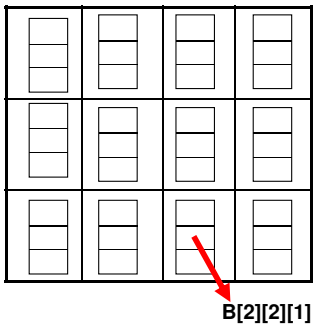
Arranjos Multi-Dimensionais

- Declaração alternativa:

  typedef int vet[3];

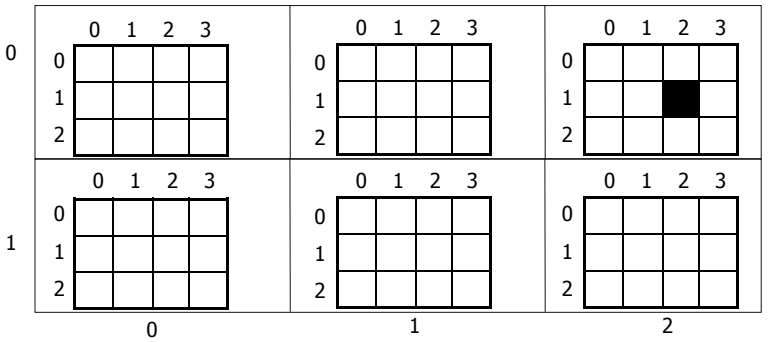
  vet B[3][4];





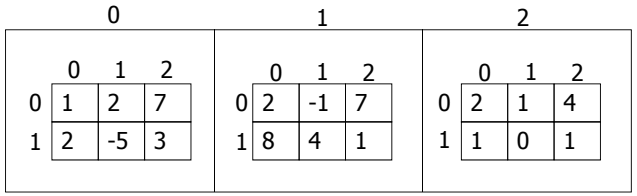
Arranjos Multi-Dimensionais

- Exemplo:
  - Arranjo A com 4 dimensões, ou, equivalentemente...
    - Matriz A 2x3 de matrizes 3x4
  - Elemento em destaque: A[0][2][1][2]



Exercícios

- Seja o seguinte arranjo multi-dimensional A:



0			
	0	1	2
0	1	2	7
1	2	-5	3

1			
	0	1	2
0	2	-1	7
1	8	4	1

2			
	0	1	2
0	2	1	4
1	1	0	1

- 1) Determine os seguintes elementos:
  - a) A[0][0][1] b) A[0][1][2]; c) A[1][1][0]
  - d) A[0][1][A[2][0][0]] e) A[A[2][0][1]][1][1]
- 2) Declare A de forma que represente a estrutura acima

Exercícios

- A **transposta** de uma matriz A com M linhas e N colunas é uma matriz B com N linhas e M colunas onde as linhas de B são as colunas de A e vice-versa. Faça um programa que leia uma matriz A com 4 linhas e 3 colunas de números reais e calcule a sua transposta B
- Faça programas que leiam uma matriz 5x5 de caracteres e escrevam:
  - A diagonal principal
  - O triângulo superior à diagonal principal
  - O triângulo inferior à diagonal principal
  - Tudo exceto a diagonal principal
  - A diagonal secundária
  - O triângulo superior à diagonal secundária
  - O triângulo inferior à diagonal secundária
  - Tudo exceto a diagonal secundária

Agradecimentos

- Prof. Rudinei Goularte
- Prof. André C. P. L. F. Carvalho

Bibliografia

- Schildt, H. "C Completo e Total", 3a. Edição, Pearson, 1997.
- Damas, L. "Linguagem C", 10a. Edição, LTC, 2007