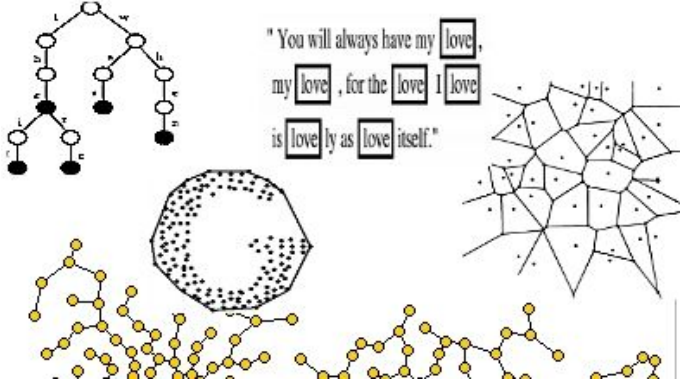
Programação Estruturada
TCC-00.347 - B1



Prof. Aline Paes / alinepaes@ic.uff.br

Vetores e matrizes

Universidade Federal Fluminense

Instituto de Computação

Vetor unidimensional

- Estrutura de dados na forma de um
  - Conjunto
    - finito
    - enumerável
    - de elementos homogêneos

- int vetor[10];
- int vetor[5] = {1,10,20,30,3};
- int vetor[] = {1,10,30,30,3};

Vetor unidimensional

- cujos elementos são acessados pela posição
  - vetor[0]: acessa a primeira posição
  - vetor[4]: acessa a última posição
  - vetor[5]: erro, não existe essa posição no vetor

Vetor unidimensional

- Acesso
  - vetor[i]
- Armazenamento
  - vetor[i] = x;

Vetor unidimensional

- int *v*[10];
- A variável *v*, que representa o vetor, é uma constante que armazena o endereço inicial do vetor
- v, sem indexação, aponta para o primeiro elemento do vetor
- v+0 ⟶ endereço do primeiro elemento do vetor
- v+1 ⟶ endereço do segundo elemento do vetor
- ...

Vetor unidimensional

- Tamanho do vetor não pode ser alterado durante a execução do programa
- Pode definir o tamanho como uma constante, para facilitar alterações no código

Vetor unidimensional

- Os elementos de um vetor são alocados em posições contíguas na memória
  - int vetor[10] = {1,10,20,30,3};

124	
120	3
116	30
112	20
108	10
104	1

Exemplo

- Armazenar 10 valores e calcular a sua média

```
#include <stdio.h>

int main(){
	float valores[10]; // declaração do vetor
	float soma = 0.0, media;
	int cont;
	// lendo e armazenando os valores
	for (cont = 0; cont < 10; cont ++){
		puts("Digite o próximo valor: ");
		scanf("%f", &valores[cont]);
	}

	// cálculo da média
	for (cont = 0; cont < 10; cont ++)
		soma += valores[cont];
	media = soma / 10;
	printf("A média é: %.2f\n", media);
	return 0;
}
```

Vetores em C

- E se não soubermos o tamanho do vetor de antemão?
  - Ou declara um vetor de tamanho grande o suficiente
    - (única opção em versões antigas do compilador C)
    - int vetor[1000];
  - pode desperdiçar espaço, caso a quantidade real esteja muito abaixo da estimada

Vetores em C

- E se não soubermos o tamanho do vetor de antemão?
  - Ou pergunta ao usuário a quantidade de elementos que o vetor terá (com scanf), **antes** de declarar o vetor
    - **VLA**: variable-length array
    - disponível no compilador C99 (versão padrão a partir de 2000)
    - É perigoso se o usuário digitar um valor maior que a quantidade de memória disponível
      - difícil com a quantidade de memória que temos atualmente
    - em todos os casos, uma vez definido o tamanho, ele não muda

Exemplo: Armazenar N valores e calcular a sua média

```
#include <stdio.h>

int main(){
	int qtd, cont;
	float soma = 0.0, media;
	// descobrindo o tamanho do vetor
	puts("Digite a quantidade de elementos a serem digitados: ");
	scanf("%i", &qtd);
	float valores[qtd]; // declaração do vetor, tamanho dele vem da variável
	// lendo e armazenando os valores
	for (cont = 0; cont < qtd; cont ++){
		puts("Digite o próximo valor: ");
		scanf("%f", &valores[cont]);
	}
	// cálculo da média
	for (cont = 0; cont < qtd; cont ++)
		soma += valores[cont];
	media = soma / qtd;
	printf("A média é: %.2f\n", media);
	return 0;
}
```

Passando vetores para funções

- Mesma declaração de tipo
  - int f1(int v[])
  - int f1(int v[10])

- Ou usando o tipo ponteiro (que veremos depois)

Passando vetores para funções

- O que é passado é o endereço de memória do primeiro elemento
- Alterações na função serão refletidas externamente

112	30
108	10
104	5

f1	v1	104
main	v	104

Passando vetores para funções

```
/* Cálculo da média e da variância*/

#include <stdio.h>

/* Função para cálculo da média */
float media (int n, float v[]) {
     float s = 0.0;
     for (int i = 0; i < n; i++)
          s += v[i];
     return s/n;
}

/* Função para cálculo da variância */
float variancia (int n, float v[],
float m) {
     float s = 0.0;
     for (int i = 0; i < n; i++)
          s += (v[i] - m) * (v[i] - m);
     return s/n;
}
```

```
int main ( ) {
     float v[10];
     float med, var;
     /* leitura dos valores */
     for (int  i = 0; i < 10; i++ )
          scanf("%f", &v[i]);

     med = media(10,v);
     var = variancia(10,v,med);
     printf ( "Media = %f Variancia
= %f \n", med, var);
     return 0;
}
```

O que será impresso?

```c
#include <stdio.h>


void altera_vetor(int v[]){
 printf("%d\n", v[0]);
 v[0] = 10;
 printf("%d\n", v[0]);
}

int main(){
 int v[] = {1,2,3};
 printf("%d\n", v[0]);
 altera_vetor(v);
 printf("%d\n", v[0]);
 return 0;
}
```

Tamanho de vetor: o que será impresso?

```c
#include <stdio.h>



int main(){
 int v[] = {1,2,3};
 int len = sizeof(v);
 printf("%d\n", len);
 return 0;
}
```

Tamanho de vetor: o que será impresso?

```c
#include <stdio.h>



int main(){
 int v[] = {1,2,3};
 int len = sizeof(v)/sizeof(v[0]);
 printf("%d\n", len);
 return 0;
}
```

Tamanho de vetor: o que será impresso?

```c
#include <stdio.h>



int tamanho_vetor(int v[]){
 int len = sizeof(v)/sizeof(v[0]);
 return len;
}

int main(){
 int v[] = {1,2,3};
 printf("%d\n", tamanho_vetor(v));
 return 0;
}
```

Passando vetores para funções: o que será impresso?

```c
/* Incrementa elementos de um vetor */
#include <stdio.h>

void incr_vetor ( int n, int v[]) {
 int i;
 for (i = 0; i < n; i++)
   v[i]++;
}

int main ( ) {
 int a[ ] = {1, 3, 5};
 incr_vetor(3, a);
 printf("%d %d %d \n", a[0], a[1], a[2]);
 return 0;
}
```

Exemplo: busca binária

- entrada: vetor ordenado com *n* elementos e elemento a ser encontrado
- saída: *i* se o elemento elem ocorre na posição i do vetor; -1 c.c
- procedimento:
  - compare o elemento com o elemento do meio de vet
  - se elemento for menor, pesquise na primeira metade do vetor
  - se elem for maior, pesquise na segunda parte do vetor
  - se for igual, retorne a posição
  - continue o procedimento, subdividindo a parte de interesse, até encontrar o elemento ou chegar a uma parte do vetor com tamanho 0

Exemplo: busca binária

```
int busca_binaria (int elemento, int termos, int vetor[]) {
  int esquerda = -1, direita = termos;
  while (esquerda  < direita-1) {
     int meio = (esquerda + direita)/2;
     if (vetor[meio] < elemento)
       esquerda = meio;
     else direita = meio;
  }
  if (vetor[direita] == elemento)
   return direita;
 return -1;
}
```

Matrizes

- Declaração:
  - float matriz[10][3];
  - float mat[4][3] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
  - float mat[4][3] = {1,2,3,4,5,6,7,8,9,10,11,12};
  - float mat[][3] = {1,2,3,4,5,6,7,8,9,10,11,12};
    - Número de colunas não pode ser omitido

- Organizada de maneira contígua, linha a linha

Passando uma matriz para uma função

```
int passar_matriz(int m[100][100], int l, int c ){
...
}
int passar_matriz(int l, int c, int m[l][c] ){
...
}
```

Passando uma matriz para uma função

```
int passar_matriz(int  l, int c, int m[][100]){ ...

}

int passar_matriz(int l, int c, int m[][c] ){

...

}
```

Matriz como parâmetro

```c
#include <stdio.h>

void ler_matriz(int m[4][2], int l, int c){
 for (int i = 0; i < l; i++)
   for (int j = 0; j < c; j++)
     scanf("%d", &m[i][j]);

}

void escrever_matriz(int l, int c, int m[][c]){
 for (int i = 0; i < l; i++){
   for (int j = 0; j < c; j++)
     printf("%d ", m[i][j]);
   printf("\n");
 }
}
```

```c
int main(){
 int matriz[4][2];
 ler_matriz(matriz, 4, 2);
 escrever_matriz(4,2,matriz);
}
```

Exercícios

1. Tentando descobrir se um dado era viciado, um dono de cassino honesto o lançou *n* vezes. Leia os *n* resultados dos lançamentos, guarde-os em um vetor e determine o número de ocorrências de cada face.

2. Dados dois vetores *x* e *y*, ambos com *n* elementos, determinar a soma dos produtos dos elementos desses vetores.

3. Dada uma seqüência de *n* números reais, determinar os números que compõem a seqüência e o número de vezes que cada um deles ocorre na mesma.
   a. Exemplo: *n* = 8
   b. Seqüência: -1.7, 3.0, 0.0, 1.5, 0.0, -1.7, 2.3, -1,7
   c. Saída: -1.7 ocorre 3 vezes
      3.0 ocorre 1 vez
      0.0 ocorre 2 vezes
      1.5 ocorre 1 vez
      2.3 ocorre 1 vez

Exercícios

4. Faça uma função que recebe um vetor de reais e retorna o valor mais próximo da média.
Exemplo: supondo vetor = {2.5, 7.5, 10.0, 4.0}
(média = 6.0 (não deve ser impresso))
- *Valor mais próximo da média = 7.5*

5. Faça uma função que recebe dois vetores e intercala os elementos de ambos formando um terceiro vetor. O terceiro vetor deve começar do primeiro elemento do menor vetor. O maior vetor deverá ter seus elementos restantes repetidos no fim do vetor de resposta.
Exemplo:
intercalar_vetor([1,2,3,4], [10,20,30,40,50,15])
[1,10,2,20,3,30,4,40,50,15]

Exercícios

6. Escreva uma função para multiplicar os elementos da diagonal principal de uma matriz por um valor k.

7. Faça uma função para calcular o produto de duas matrizes. A função deve verificar se as matrizes são de tamanhos compatíveis para multiplicação. Lembre que c[i, j] = Somatório de a[i,k] * b[k,j] para todo k

8. Faça um programa para informar a linha de maior soma de uma matriz de inteiros.

Exercícios

9. Faça um programa que leia a ordem de uma matriz quadrada (até 100), posteriormente leia os seus valores e finalmente calcule a sua transposta, onde at[i, j] = a[j, i]

10. Escreva um programa em C que receba um array de inteiros e gere uma representação compacta usando o método RLE (*Run Length Encoding*). Para cada ocorrência de um número inteiro o método RLE armazena o valor correspondente juntamente com o número de ocorrências consecutivas. Exemplo:

1 2 3 5 5 5 7 7 7 7 7 2 2 2 0 0 0 0 0 9 9 9 12 12 12 4 4

1 **1** 2 **1** 3 **1** 5 **3** 7 **5** 2 **3** 0 **5** 9 **3** 12 **3** 4 **2**