Vetores e Matrizes

C3

Universidade Federal de Rio Grande
Centro de Ciências Computacionais

Estruturas Compostas

- Estruturas que armazenam dados do mesmo tipo.
- São chamadas de estruturas compostas homogêneas.
- Facilitam o acesso e a manipulação de uma grande quantidade de dados.

Exemplo

- Faça um programa que leia as notas de uma prova para uma classe de 5 alunos, calcule a média da classe e depois mostre todas as notas maiores que a média.
- Exemplo de resposta:

```c
#include <stdio.h>

int main(){
    int nota0, nota1, nota2, nota3,
        nota4, media;

    printf("Digite a nota do aluno 0: ");
    scanf("%d",&nota0);
    printf("Digite a nota do aluno 1: ");
    scanf("%d",&nota1);
    printf("Digite a nota do aluno 2: ");
    scanf("%d",&nota2);
    printf("Digite a nota do aluno 3: ");
    scanf("%d",&nota3);
    printf("Digite a nota do aluno 4: ");
    scanf("%d",&nota4);
    media = (nota0+nota1+nota2+nota3+nota4)/5;
    printf("media: %d\n",media);

    if (nota0>media)
       printf("Nota 0: %d \n",nota0);
    if (nota1>media)
       printf("Nota 1: %d \n",nota1);
    if (nota2>media)
       printf("Nota 2: %d \n",nota2);
    if (nota3>media)
       printf("Nota 3: %d \n",nota3);
    if (nota4>media)
       printf("Nota 4: %d \n",nota4);
}
```

Exemplo

- Agora resolva o mesmo problema para uma turma de 50 alunos? E para uma escola com 2000 alunos? A solução se torna inviável.
- Um **vetor** ou **array** é um tipo de dado usado para representar uma certa quantidade de variáveis de valores homogêneos, ou seja, do mesmo tipo.

Declaração

- A sintaxe em C para a criação de um vetor é:

  tipo `identificador[`*número de variáveis*`];`

  Onde
  - *tipo* é o tipo das variáveis que devem ser criadas. Exemplo: *int*, *char*, *float*, entre outros;
  - *identificador* é o nome que será utilizado para referenciar o conjunto de variáveis;
  - *número de variáveis* é o tamanho do vetor.
- Um exemplo de declaração de vetor em C:

```
int notas[5];
int alunos[100];
double nums[30];
char nome[80];
```

Acessando o Vetor

- Para acessar um elemento individual do vetor usamos os colchetes e colocamos entre eles o número da posição a ser acessada.

```
int notas[5];
notas[2] = 85; /* Acessa a posicao 2 do vetor */
```

- ATENÇÃO: Todo vetor sempre começa na posição zero.

```
int notas[5];
notas[2] = 85; /* TERCEIRO elemento do vetor */
```

Representação Interna do Vetor

- Por exemplo, um vetor chamado `nums`. Sua declaração e sua estrutura na memória ficaria da seguinte forma:

```
/* declaracao */
float nums[10];

/* Representacao na memoria

          0   1   2   3   4   5   6   7   8   9    <= indice
        ---------------------------------------
  nums =|   |   |   |   |   |   |   |   |   |   |
        |___|___|___|___|___|___|___|___|___|___|
*/
```

- Note que os índices variam de 0 a 9. Em **C**, os vetores *SEMPRE* começam pelo índice 0 (zero).

Vetores, Índices e Variáveis

- Além disso, pode-se referenciar posições do vetor utilizando variáveis inteiras, ou usar expressões que resultem em número inteiro válido.

```
// Declaracao de variaveis e vetores
int notas[5];
float nums[10];
int i = 2;

// Usando vetores
notas[i] = 85;
nums[10+5]=notas[i];
notas[3]=notas[i]+notas[2];
```

Cuidado com Índices Inválidos

- Não acesse o vetor com índices inválidos, ou seu programa será finalizado pelo sistema operacional.

```
double nums[10];

num[0]= 10.3; /* OK */
num[1]= 5.9;  /* OK */

num[10]=5.9;  /* ERRADO: O indice deve
                 variar entre 0 e o tamanho do
                 vetor menos 1, no caso 9 */

num[-1]=10.1;  /* ERRADO - Nao existem
                  indices negativos*/
```

Exemplo - Notas usando Vetores

```c
#include <stdio.h>

int main() {
  int notas[5];
  int soma=0;
  float media;

  for(i=0; i<5; i++){ /* Percorre o vetor de 0 a 4 */
    printf("Digite a nota do aluno %d: ",i);
    scanf("%d",&notas[i]);
    soma=soma+notas[i];
  }

  media=soma/5;
  printf("Media das notas: %d.\n",media);

  for(i=0; i<5; i++) /* Percorre o vetor de 0 a 4 */
    if(notas[i]>media)
       printf(" Nota[%d]: %d.\n",i,notas[i]);
}
```

Tamanho do Vetor

- O tamanho do vetor deve ser determinado em tempo de compilação. Não posso colocar uma variável para delimitar o seu valor.

```
int x=100;
double nums[x]; /* ERRADO */
```

- Mas em compensação posso facilitar a manutenção do meu programa usando constantes.

Constantes em C: Comando `#define`

- `#define` deve ficar no início do programa fonte, logo após a declaração das bibliotecas (`#include`).
- O `#define` define uma macro.
- As operações `#define` são executadas pelo pré-processador do compilador. Ou seja, são processadas antes de compilar o código.
- O pré-processador substitui (literalmente) no código fonte cada constante pelo seu respectivo valor.
- Depois de declarada a constante pode ser usada no programa como se fosse uma variável. Porém ela não pode ter seu valor alterado usando atribuição.

Exemplo

```
#include <stdio.h>

#define MAX 100

int main(){
    int vetor[MAX];
    int i;
    for (i=0;i<MAX;i++)
        vetor[i]=i*MAX;
}
```

Inicializando Vetores

- Assim como as variáveis, pode-se atribuir um valor inicial a um vetor.
- Esta inicialização deve ser feita no momento da criação do vetor.
- O trecho de código a seguir exemplifica a inicialização do vetor `moedas` com os valores 50 na posição 0, 25 na posição 1, 10 na posição 2, 5 na posição 3 e 1 na posição 4.

```
int moedas[5]={50,25,10,5,1};
```

Vetores e Strings

- Em C uma **string** é um vetor do tipo `char` terminada pelo caractere *null* ($'\backslash 0'$).
- No exemplo que segue temos a criação de um vetor de caracteres ou *string*.

```
char str[20];
```

Lendo Strings 01

- Podemos armazenar informações caractere a caractere, como mostra o segmento de código a seguir:

```
int i;
char str[20];

for(i=0; i<20; i++){
  printf("Digite o caracter %d: ",i);
  scanf("%c",&str[i]);
}
```

Lendo Strings 02

- Ou podemos ler todos os caracteres como uma string, usando `scanf()`.

```
char str[20];

printf("Digite a sequencia de caracteres: ");
scanf("%s",str);
```

- Note que não tem & antes da variável no `scanf`. Isso acontece porque `str` é um vetor e não uma variável de tipo primitivo (`int`, `double`, `char`, ...).

Outros Exemplos de Strings

```
char str[20]="teste";

/* 1 - QUAL A LETRA QUE IMPRIME? */
printf("%c",str[2]);

/* 2 - QUAL O NOVO CONTEUDO DA STRING */
str[0]='P';
```

Outros Exemplos de Strings

```c
char str[20]="teste";

/* 1 - QUAL A LETRA QUE IMPRIME? */
printf("%c",str[2]);

/* 2 - QUAL O NOVO CONTEUDO DA STRING */
str[0]='P';
```

- **Respostas:**
  1. Vai imprimir a letra **'s'**, que é a terceira letra da *string*.
  2. A palavra em `str` passa a ser "Peste". Já que a primeira letra de `str` é substituída por `'P'`.
- Na linguagem **C**, um `char` é representado entre aspas simples (`' '`), já uma *string*, que é um vetor de `char`, é representado entre aspas duplas (`" "`).

Exercícios

1. A declaração do vetor está correto? Qual o erro?

```
int vetor(25);
```

Exercícios

1. A declaração do vetor está correto? Qual o erro?

```
int vetor(25);
```

2. Qual o elemento da vetor referenciado pela expressão?

```
vetor[4]
```

Exercícios

3. Faça um programa para ler 10 números e mostre-os na ordem inversa.

Exercícios

3. Faça um programa para ler 10 números e mostre-os na ordem inversa.
   Resposta:

```c
#include <stdio.h>
#define MAX 10
int main() {
  int numeros[MAX];
  int i;

  /* LE OS NUMEROS */
  for (i=0; i < N; i++)
    scanf("%d",&numeros[i]);

  /* MOSTRA OS NUMEROS EM ORDEM INVERSA */
  for (i=(N-1);i >=0; i--)
    printf("%d",numeros[i]);
}
```

Exercícios

4. Faça um programa para ler as notas de 20 alunos e armazená-las em um vetor. Em seguida o programa deve descobrir se existe alguma nota x escolhida pelo usuário. O programa deve permitir que o usuário procure quantas notas quiser. O programa deve encerrar quando uma nota negativa for digitada.

Resposta

```c
#include <stdio.h>
#define N 20
int main() {
  float nota[N],x;
  int i;

  /* LE AS NOTAS E COLOCA NO VETOR */
  printf("Entre com %d notas\n",N);
  for (i=0; i < N; i++) {
    scanf("%f",&nota[i]);
  }

  /* PERGUNTA A NOTA A SER PROCURADA */
  printf("Qual nota deseja procurar? ");
  scanf("%f",&x);

  /* REPETE ATE DIGITAR NOTA NEGATIVA */
  while (x>=0) {
     i = 0;

     /* PROCURA PELA NOTA */
     while ((nota[i] != x) && (i<N))
        i++;

     /* VERIFICA SE A NOTA FOI ACHADA */
     if (i < N)
        printf("nota %f encontrada
                na posicao %d\n",nota[i],i);
     else
        printf("nota %f nao encontrada\n",x);

     /* PERGUNTA PELA PROXIMA NOTA */
     printf("Qual nota deseja procurar? ");
     scanf("%f",&x);
  }
  return 0;
}
```

Exercícios

5. Faça um programa para ler uma palavra e que em seguida mostre a primeira e a última letra da palavra. Dica: Para descobrir o final da palavra, procure pelo caractere especial $'\backslash 0'$ (*null*).

Resposta

```c
#include <stdio.h>
#define TAM 40
int main() {
  char palavra[TAM];
  char prim, ult;
  int i;

  printf("Digite uma palavra: ");
  scanf("%s", palavra);

  /* Primeira letra esta na posicao 0 */
  printf("Primeira letra: %c\n",palavra[0]);


  /* Ultima letra esta na posicao anterior ao '\0' */

  /* Procura o '\0' */
  i=0;
  while(palavra[i]!='\0')
      i++;

  /* Mostra a letra antes do '\0' */
  printf("Ultima letra: %c\n",palavra[i-1]);

  return 0;
}
```

Matrizes e Estruturas de N Dimensões

- Vetores são estruturas de uma dimensão.
- Matrizes são estruturas com duas dimensões.
- A linguagem C permite o uso de estruturas de `N` dimensões.
- Só precisamos tomar cuidado com o tamanho de memória usada.

Declaração de Matrizes

- A sintaxe em C para a criação de uma matriz é:
  tipo `identificador` [*nro var 1*] [*nro var 2*] ;

  Onde
  - `tipo` é o tipo das variáveis que devem ser criados. Exemplo:*int*, *char*, *float*, entre outros;
  - `identificador` é o nome que será utilizado para referenciar o conjunto de variáveis;
  - *nro var 1* é o número de variáveis de cada vetor que será criado.
  - *nro var 2* é o número de vetores que será criado.
- Exemplos de declaração de matriz em C:

```
int notas[3][8];
int matriz[4][4];
float m2[10][7];
```

Declaração de Estruturas com N Dimensões

- Podemos criar também vetores de vetores de vetores de vetores de vetores ...,
- Ou seja, podemos ter estruturas de múltiplas dimensões.
- Um exemplo de declaração de uma estrutura de 3 dimensões em C:

```
int notas[3][8][4];
double var5d[4][3][5][7][10];
```

  - Pergunta: Quantas `notas` foram declaradas?

Declaração de Estruturas com N Dimensões

- Podemos criar também vetores de vetores de vetores de vetores de vetores ...,
- Ou seja, podemos ter estruturas de múltiplas dimensões.
- Um exemplo de declaração de uma estrutura de 3 dimensões em C:

```
int notas[3][8][4];
double var5d[4][3][5][7][10];
```

  - Pergunta: Quantas `notas` foram declaradas?
  - 96 Notas.
  - Pergunta: Quantas `var5d` foram declaradas?

Declaração de Estruturas com N Dimensões

- Podemos criar também vetores de vetores de vetores de vetores de vetores ...,
- Ou seja, podemos ter estruturas de múltiplas dimensões.
- Um exemplo de declaração de uma estrutura de 3 dimensões em C:

```
int notas[3][8][4];
double var5d[4][3][5][7][10];
```

  - Pergunta: Quantas `notas` foram declaradas?
  - 96 Notas.
  - Pergunta: Quantas `var5d` foram declaradas?
  - 4200
  - Pergunta: Qual o tamanho da estrutura `var5d` se o cada double tiver 64 bits?

Declaração de Estruturas com N Dimensões

- Podemos criar também vetores de vetores de vetores de vetores de vetores ...,
- Ou seja, podemos ter estruturas de múltiplas dimensões.
- Um exemplo de declaração de uma estrutura de 3 dimensões em C:

```
int notas[3][8][4];
double var5d[4][3][5][7][10];
```

  - Pergunta: Quantas `notas` foram declaradas?
  - 96 Notas.
  - Pergunta: Quantas `var5d` foram declaradas?
  - 4200
  - Pergunta: Qual o tamanho da estrutura `var5d` se o cada double tiver 64 bits?
  - 4200*64=268800 bits ou 33600Bytes ou 32,8125 KB.

Acessando Matrizes

- Para acessar os elementos de uma matriz, usamos dois índices. Um em cada par de colchetes.
- Exemplo:

```
int matriz[5][3];
```

matriz[0][0]=1
matriz[3][2]=5
matriz[4][1]=3

	0	1	2
0	**1**		
1			
2			
3			**5**
4		**3**	

```
matriz[5][1]; /* ERRADO - Cuidado */
matriz[1][3]; /* ERRADO - Cuidado */
```

Inicializando Matrizes

- Assim como em vetores podemos atribuir um valor inicial a uma matriz, esta inicialização deve ser feita no momento da criação da matriz.

```
int teste[4][4]={ 3,4,7,6,
                  3,2,1,7,
                  1,8,1,8,
                  1,9,0,1 };

/* O C tambem aceita chaves para */
/* delimitar cada linha.  */
int identidade[4][4]={ {1,0,0,0},
                       {0,1,0,0},
                       {0,0,1,0},
                       {0,0,0,1} };
```

Exercícios

6. Faça um programa que leia uma matriz 4×4. O programa deve perguntar o valor de cada posição da matriz, em seguida imprimir esta matriz na tela.

Resposta

```c
#include <stdio.h>
#define COL 4
#define LIN 3
int main(){
  int mat[LIN][COL];
  int i=0,j=0;
  for (i=0; i<LIN; i++) {
    for (j=0; j<COL; j++) {
      printf("Digite o valor da posicao (%d,%d): ",i,j);
      scanf("%d",&mat[i][j]);
    }
  }
  for (i=0; i<LIN i++){
    for (j=0; j<COL; j++)
      printf("%d ",mat[i][j]);
    printf("\n");
  }
  return 0;
}
```

Exercícios

7. Refaça o exercício anterior, mas agora imprima a matriz transposta da matriz lida.

Resposta

```c
#include <stdio.h>
#define COL 4
#define LIN 3
int main(){
  int mat[LIN][COL];
  int i=0,j=0;
  for (i=0; i<LIN; i++) {
    for (j=0; j<COL; j++) {
      printf("Digite o valor da posicao (%d,%d): ",i,j);
      scanf("%d",&mat[i][j]);
    }
  }

   /* Mostra a  Transposta - INVERTE O i E O j */
  for (j=0; j<COL; j++){
    for (i=0; i<LIN; i++)
      printf("%d ",mat[i][j]);
    printf("\n");
  }
  return 0;
}
```

Exercícios

8. Faça um programa que calcule a soma de duas matrizes 10×10. Os elementos da primeira matriz devem ser lidos pelo teclado e os elementos da segunda devem ser definidos pela fórmula $a_{i,j} = i.j$.

Resposta

```c
#include <stdio.h>
#define LIM 10
int main(){
  int mat1[LIM][LIM];
  int mat2[LIM][LIM];
  int matSoma[LIM][LIM];
  int i=0,j=0;

  for (i=0; i<LIM; i++) {
    for (j=0; j<LIM; j++) {
      printf("Digite o valor da posicao (%d,%d) da matriz 1: ",i,j);
      scanf("%d",&mat1[i][j]);
      /* Calcula a matriz 2 */
      mat2[i][j]=i*j;
    }
  }

  /* Soma as matrizes mat1 e mat2 e coloca a resposta em matSoma */
  for (i=0; i<LIM; i++)
    for (j=0; j<LIM; j++)
        matSoma[i][j]=mat1[i][j]+mat2[i][j];

  printf("\n Matriz 1 + Matriz 2\n");
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      printf("%d ",matSoma[i][j]);
    printf("\n");
  }
  return 0;
}
```

Exercícios

9. Leia 2 matrizes 8×8, mostre elas na tela e então calcule e mostre a multiplicação entre elas.

Resposta

```c
#include <stdio.h>
#define LIM 8
int main(){
  int mat1[LIM][LIM];
  int mat2[LIM][LIM];
  int resp[LIM][LIM];
  int i=0,j=0,k=0;

  for (i=0;i<LIM; i++) {
    for (j=0;j<LIM; j++) {
      printf("Digite o valor da posicao (%d,%d)
              da matriz 1: ",i,j);
      scanf("%d",&mat1[i][j]);

      printf("Digite o valor da posicao (%d,%d)
              da matriz 2: ",i,j);
      scanf("%d",&mat2[i][j]);
    }
  }
```

```c
  /* NAO ESQUECER DE ZERAR A MATRIZ RESPOSTA */
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      resp[i][j]=0;


  /*  --------  MULTIPLICACAO ---------*/
  for (i=0; i<LIM; i++)
    for (j=0; j<LIM; j++)
      for (k=0; k<LIM; k++)
        resp[i][j]= resp[i][j]+
                    (mat1[i][k]*mat2[k][j]);

 /* Mostra o resultado */
  printf("\n Matriz 1 * Matriz 2\n");
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      printf("%d ",resp[i][j]);
    printf("\n");
  }
  return 0;
}
```

Referências

- ASCENCIO, Ana F. G.; CAMPOS, Edilene A. V. Fundamentos da programa de computadores. Pearson Prentice Hall, 2007.
- Schildt, Herbert. C completo e total. Pearson Makron Books, 1997.
- Notas de aula do Prof. Flavio Keidi Miyazawa.
- Notas de aula do Prof. Emanuel Estrada.
- Notas de aula do Prof. Alessandro Bicho.