Vetores

Informática – IFSULDEMINAS

Primeiro Semestre de 2013

Roteiro

Problema

Considere o problema a seguir:

Problema

Escreva um programa que leia as notas de uma prova para uma classe de 5 alunos e então calcule a média desta classe.

Uma solução ...

Média das notas

```
float n0, n1, n2, n3, n4, media;
printf("Digite a nota do aluno 0: ");
scanf("%f",&n0);
printf("Digite a nota do aluno 1: ");
scanf("%f",&n1);
printf("Digite a nota do aluno 2: ");
scanf("%f",&n2);
printf("Digite a nota do aluno 3: ");
scanf("%f",&n3);
printf("Digite a nota do aluno 4: ");
scanf("%f",&n4);
media = (n0 + n1 + n2 + n3 + n4)/5;
printf("Media: %f\n",media);
```

E se a turma tivesse 100 alunos? 5000 alunos?

Uma solução ...

Média das notas

```
float n0, n1, n2, n3, n4, media;
printf("Digite a nota do aluno 0: ");
scanf("%f",&n0);
printf("Digite a nota do aluno 1: ");
scanf("%f",&n1);
printf("Digite a nota do aluno 2: ");
scanf("%f",&n2);
printf("Digite a nota do aluno 3: ");
scanf("%f",&n3);
printf("Digite a nota do aluno 4: ");
scanf("%f",&n4);
media = (n0 + n1 + n2 + n3 + n4)/5;
printf("Media: %f\n",media);
```

E se a turma tivesse 100 alunos? 5000 alunos?

Vetores

Vetor

Um vetor é um tipo de dado que agrupa uma quantidade de variáveis do mesmo tipo.

E se agrupássemos as notas dos alunos em um vetor?

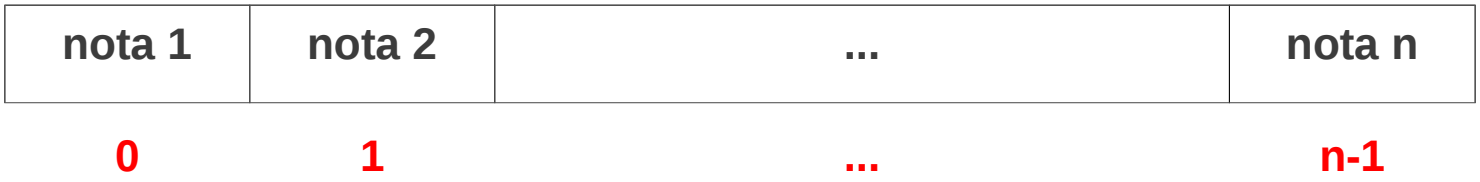
Vetores

Vetor

Um vetor é um tipo de dado que agrupa uma quantidade de variáveis do mesmo tipo.

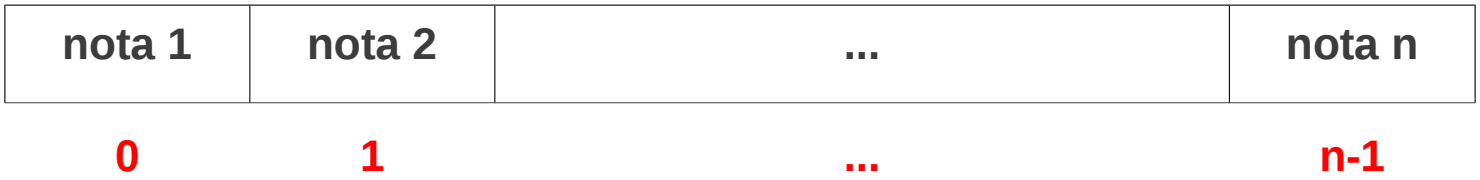
E se agrupássemos as notas dos alunos em um vetor?

Vetores



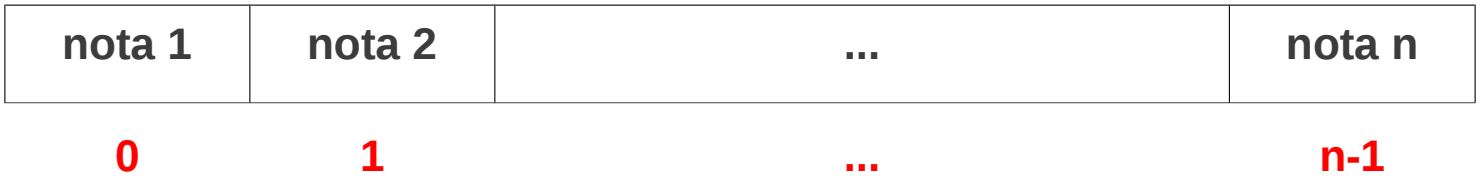
- Um vetor possui um tamanho.
- Cada valor no vetor é acessado por meio de índices ou posições.
- Os índices vão de **0** a **tamanho - 1**.

Vetores



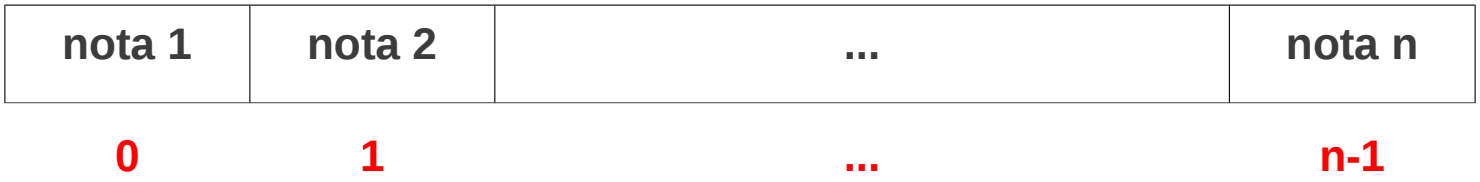
- Um vetor possui um tamanho.
- Cada valor no vetor é acessado por meio de índices ou posições.
- Os índices vão de **0** a **tamanho - 1**.

Vetores



- Um vetor possui um tamanho.
- Cada valor no vetor é acessado por meio de índices ou posições.
- Os índices vão de **0** a **tamanho - 1**.

Vetores



- Um vetor possui um tamanho.
- Cada valor no vetor é acessado por meio de índices ou posições.
- Os índices vão de **0** a **tamanho - 1**.

Declarando vetor

Declaração

```
tipo nome[número_variáveis]
```

- **tipo** - é o tipo das variáveis (int, float, char, etc).
- **nome** - é o nome do vetor.
- **número variáveis** - é o número de variáveis no vetor ou tamanho do vetor.

Declarando um vetor de notas

```
float notas[5];
```

Declarando vetor

Declaração

```
tipo nome[número_variáveis]
```

- **tipo** - é o tipo das variáveis (int, float, char, etc).
- **nome** - é o nome do vetor.
- **número variáveis** - é o número de variáveis no vetor ou tamanho do vetor.

Declarando um vetor de notas

```
float notas[5];
```

Declarando vetor

Declaração

```
tipo nome[número_variáveis]
```

- **tipo** - é o tipo das variáveis (int, float, char, etc).
- **nome** - é o nome do vetor.
- **número variáveis** - é o número de variáveis no vetor ou tamanho do vetor.

Declarando um vetor de notas

```
float notas[5];
```

Declarando vetor

Declaração

```
tipo nome[número_variáveis]
```

- **tipo** - é o tipo das variáveis (int, float, char, etc).
- **nome** - é o nome do vetor.
- **número variáveis** - é o número de variáveis no vetor ou tamanho do vetor.

Declarando um vetor de notas

```
float notas[5];
```

Declarando vetor

Declaração

```
tipo nome[número_variáveis]
```

- **tipo** - é o tipo das variáveis (int, float, char, etc).
- **nome** - é o nome do vetor.
- **número variáveis** - é o número de variáveis no vetor ou tamanho do vetor.

Declarando um vetor de notas

```
float notas[5];
```

Referenciando um elemento do vetor

- Após declarar um vetor, precisamos de um modo de referenciar seus elementos, seja para leitura ou escrita.
- Isto é feito por meio dos índices.

Referenciando elementos

```
int notas[5];
notas[2] = 85;
```

- No código acima:
  - Declaramos o vetor notas de tamanho 5.
  - Atribuímos o número 85 à terceira posição do vetor.
  - As posições do vetor vão de 0 a 4.

Referenciando um elemento do vetor

- Após declarar um vetor, precisamos de um modo de referenciar seus elementos, seja para leitura ou escrita.
- Isto é feito por meio dos índices.

Referenciando elementos

```
int notas[5];
notas[2] = 85;
```

- No código acima:
  - Declaramos o vetor notas de tamanho 5.
  - Atribuímos o número 85 à terceira posição do vetor.
  - As posições do vetor vão de 0 a 4.

Referenciando um elemento do vetor

- Após declarar um vetor, precisamos de um modo de referenciar seus elementos, seja para leitura ou escrita.
- Isto é feito por meio dos índices.

Referenciando elementos

```
int notas[5];
notas[2] = 85;
```

- No código acima:
  - Declaramos o vetor notas de tamanho 5.
  - Atribuímos o número 85 à terceira posição do vetor.
  - As posições do vetor vão de 0 a 4.

Referenciando um elemento do vetor

- Após declarar um vetor, precisamos de um modo de referenciar seus elementos, seja para leitura ou escrita.
- Isto é feito por meio dos índices.

Referenciando elementos

```
int notas[5];
notas[2] = 85;
```

- No código acima:
  - Declaramos o vetor notas de tamanho 5.
  - Atribuímos o número 85 à terceira posição do vetor.
  - As posições do vetor vão de 0 a 4.

Referenciando um elemento do vetor

- Após declarar um vetor, precisamos de um modo de referenciar seus elementos, seja para leitura ou escrita.
- Isto é feito por meio dos índices.

Referenciando elementos

```
int notas[5];
notas[2] = 85;
```

- No código acima:
  - Declaramos o vetor notas de tamanho 5.
  - Atribuímos o número 85 à terceira posição do vetor.
  - As posições do vetor vão de 0 a 4.

Referenciando um elemento do vetor

- Após declarar um vetor, precisamos de um modo de referenciar seus elementos, seja para leitura ou escrita.
- Isto é feito por meio dos índices.

Referenciando elementos

```
int notas[5];
notas[2] = 85;
```

- No código acima:
  - Declaramos o vetor notas de tamanho 5.
  - Atribuímos o número 85 à terceira posição do vetor.
  - As posições do vetor vão de 0 a 4.

Armazenando dados em um vetor

Armazenando dados

```
for(i=0; i<5; i++){
    printf("Digite a nota do aluno %d: ", i);
    scanf("%f",&notas[i]);
}
```

- Podemos utilizar variáveis (**i**) para referenciar elementos do vetor.
- Atribuímos valores à **notas[i]**.

Armazenando dados em um vetor

Armazenando dados

```
for(i=0; i<5; i++){
   printf("Digite a nota do aluno %d: ", i);
   scanf("%f",&notas[i]);
}
```

- Podemos utilizar variáveis (**i**) para referenciar elementos do vetor.
- Atribuímos valores à **notas[i]**.

Armazenando dados em um vetor

Armazenando dados

```
for(i=0; i<5; i++){
   printf("Digite a nota do aluno %d: ", i);
   scanf("%f",&notas[i]);
}
```

- Podemos utilizar variáveis (**i**) para referenciar elementos do vetor.
- Atribuímos valores à **notas[i]**.

Lendo dados em um vetor

Lendo dados

```
soma = 0;
for(i=0; i<5; i++) {
    soma = soma + notas[i];
}
printf("Media das notas: %f.",soma/5);
```

- O código acima percorre o vetor somando as notas.
- Obtemos uma nota por meio do código: **notas[i]**.

Lendo dados em um vetor

Lendo dados

```
soma = 0;
for(i=0; i<5; i++) {
   soma = soma + notas[i];
}
printf("Media das notas: %f.",soma/5);
```

- O código acima percorre o vetor somando as notas.
- Obtemos uma nota por meio do código: **notas[i]**.

Lendo dados em um vetor

Lendo dados

```
soma = 0;
for(i=0; i<5; i++) {
   soma = soma + notas[i];
}
printf("Media das notas: %f.",soma/5);
```

- O código acima percorre o vetor somando as notas.
- Obtemos uma nota por meio do código: **notas[i]**.

Inicializando vetores

Inicializando dados

```
notas[5] = {7.5, 5.6, 10.0, 8.5, 7.4}
```

- Podemos atribuir um valor inicial a um vetor.
- O vetor notas foi inicializado com os valores 7.5, 5.6, 10.0, 8.5 e 7.4 nas posições 0, 1, 2, 3 e 4, respectivamente.

Inicializando vetores

Inicializando dados

```
notas[5] = {7.5, 5.6, 10.0, 8.5, 7.4}
```

- Podemos atribuir um valor inicial a um vetor.
- O vetor notas foi inicializado com os valores 7.5, 5.6, 10.0, 8.5 e 7.4 nas posições 0, 1, 2, 3 e 4, respectivamente.

Inicializando vetores

Inicializando dados

```
notas[5] = {7.5, 5.6, 10.0, 8.5, 7.4}
```

- Podemos atribuir um valor inicial a um vetor.
- O vetor notas foi inicializado com os valores 7.5, 5.6, 10.0, 8.5 e 7.4 nas posições 0, 1, 2, 3 e 4, respectivamente.

Exemplo com vetores

C[] = A[] + B[]

Lendo vetores

```
void ler_vetor(int vetor[], int n) {
   int i;
   for (i=0; i<n; i++) {
      printf("Qual o valor %d?", i);
      scanf("", &vetor[i]);
   }
}
```

Imprimindo vetores

```
void imprimir_vetor(int vetor[], int n) {
   int i;
   for (i=0; i<n; i++) {
      printf("%d ", vetor[i]);
   }
}
```

Exemplo com vetores

Somando vetores

```
void somar_vetor(int A[], int B[], int C[], int n) {
   int i;
   for (i=0; i<n; i++) {
      C[i] = A[i] + B[i];
   }
}
```

```
int main() {
   int A[200], B[200], C[200];

   ler_vetor(A, 200);
   ler_vetor(B, 200);

   somar_vetor(A, B, C, 200);

   imprimir_vetor(C, 200);
}
```

Vetores e strings

- Em C uma string é um vetor de caracteres.
- Toda string possui um caracter terminador, que marca o fim da string: '\0'.
- Para armazenar dados no vetor, podemos percorrer o vetor lendo caracteres, ou podemos usar as funções *scanf* e *gets*.

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

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

Vetores e strings

- Em C uma string é um vetor de caracteres.
- Toda string possui um caracter terminador, que marca o fim da string: '\0'.
- Para armazenar dados no vetor, podemos percorrer o vetor lendo caracteres, ou podemos usar as funções *scanf* e *gets*.

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

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

Vetores e strings

- Em C uma string é um vetor de caracteres.
- Toda string possui um caracter terminador, que marca o fim da string: '\0'.
- Para armazenar dados no vetor, podemos percorrer o vetor lendo caracteres, ou podemos usar as funções *scanf* e *gets*.

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

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

Vetores e strings

- Em C uma string é um vetor de caracteres.
- Toda string possui um caracter terminador, que marca o fim da string: '\0'.
- Para armazenar dados no vetor, podemos percorrer o vetor lendo caracteres, ou podemos usar as funções *scanf* e *gets*.

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

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

Vetores e strings

- Em C uma string é um vetor de caracteres.
- Toda string possui um caracter terminador, que marca o fim da string: '\0'.
- Para armazenar dados no vetor, podemos percorrer o vetor lendo caracteres, ou podemos usar as funções *scanf* e *gets*.

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

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

Exemplo com vetores

Invertendo um nome.

```
int main() {
   int i=0, j=0, tamanho;
   char nome[30], nome_invertido[30];

   printf("Entre com um nome: ");
   scanf("%s", nome);

   while(nome[i] != '\0') {
      i++;
   }

   tamanho = i - 1;
   for (i=tamanho; i>=0; i--){
      nome_invertido[j] = nome[i];
      j++;
   }
   printf("%s", nome_invertido);
}
```

Algumas funções para strings

- **strcpy** - faz uma cópia de strings.

```
strcpy(string_origem, string_destino)
```

- **strlen** - retorna o tamanho de uma string.

```
strlen(string)
```

- **strcmp** - compara duas strings, se as duas forem iguais retorna 0.

```
strcmp(string1, string2)
```

Algumas funções para strings

- **strcpy** - faz uma cópia de strings.

```
strcpy(string_origem, string_destino)
```

- **strlen** - retorna o tamanho de uma string.

```
strlen(string)
```

- **strcmp** - compara duas strings, se as duas forem iguais retorna 0.

```
strcmp(string1, string2)
```

Algumas funções para strings

- **strcpy** - faz uma cópia de strings.

```
strcpy(string_origem, string_destino)
```

- **strlen** - retorna o tamanho de uma string.

```
strlen(string)
```

- **strcmp** - compara duas strings, se as duas forem iguais retorna 0.

```
strcmp(string1, string2)
```

Matrizes

Matriz

Matriz é uma coleção de vetores.

- É um vetor como outro qualquer.
- Cada vetor é acessado por meio de um índice primário.
- Cada variável simples no vetor é acessada por meio de um índice secundário.

Matrizes

Matriz

Matriz é uma coleção de vetores.

- É um vetor como outro qualquer.
- Cada vetor é acessado por meio de um índice primário.
- Cada variável simples no vetor é acessada por meio de um índice secundário.

Matrizes

Matriz

Matriz é uma coleção de vetores.

- É um vetor como outro qualquer.
- Cada vetor é acessado por meio de um índice primário.
- Cada variável simples no vetor é acessada por meio de um índice secundário.

Matrizes

Matriz

Matriz é uma coleção de vetores.

- É um vetor como outro qualquer.
- Cada vetor é acessado por meio de um índice primário.
- Cada variável simples no vetor é acessada por meio de um índice secundário.

Declarando uma matriz

Matriz de tamanho $M \times N$

tipo nome [linhas] [colunas]

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo tipo.
- As linhas são numeradas de 0 a *linhas* $-$ 1.
- As colunas são numeradas de 0 a *colunas* $-$ 1.

Geralmente denotamos

- o número de *linhas* por m.
- o número de *colunas* por n.

Declarando uma matriz

Matriz de tamanho $M \times N$

tipo nome [linhas] [colunas]

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo tipo.
- As linhas são numeradas de 0 a *linhas* $-$ 1.
- As colunas são numeradas de 0 a *colunas* $-$ 1.

Geralmente denotamos

- o número de *linhas* por m.
- o número de *colunas* por n.

Declarando uma matriz

Matriz de tamanho $M \times N$

tipo nome [linhas] [colunas]

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo tipo.
- As linhas são numeradas de 0 a *linhas* $-$ 1.
- As colunas são numeradas de 0 a *colunas* $-$ 1.

Geralmente denotamos

- o número de *linhas* por m.
- o número de *colunas* por n.

Declarando uma matriz

Matriz de tamanho $M \times N$

tipo nome [linhas] [colunas]

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo tipo.
- As linhas são numeradas de 0 a *linhas* $-$ 1.
- As colunas são numeradas de 0 a *colunas* $-$ 1.

Geralmente denotamos

- o número de *linhas* por m.
- o número de *colunas* por n.

Declarando uma matriz

Matriz de tamanho $M \times N$

tipo nome [linhas] [colunas]

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo tipo.
- As linhas são numeradas de 0 a *linhas* $-$ 1.
- As colunas são numeradas de 0 a *colunas* $-$ 1.

Geralmente denotamos

- o número de *linhas* por m.
- o número de *colunas* por n.

Exemplo de declaração de matriz

```
int matriz [5][4];
```

	0	1	2	3
0				
1				
2				
3				
4				

Acessando uma matriz

Podemos utilizar um elemento da matriz, da seguinte forma:

```
nome_da_matriz [<linha>] [<coluna>]
```

Ex: `matriz[2][3]` — Refere-se a variável na 3^a linha e na 4^a coluna da matriz.

Acessando uma matriz

Podemos utilizar um elemento da matriz, da seguinte forma:

```
nome_da_matriz [<linha>] [<coluna>]
```

Ex: `matriz[2][3]` — Refere-se a variável na 3^a linha e na 4^a coluna da matriz.

Acessando uma matriz

Podemos utilizar um elemento da matriz, da seguinte forma:

```
nome_da_matriz [<linha>] [<coluna>]
```

Ex: `matriz[2][3]` — Refere-se a variável na 3^a linha e na 4^a coluna da matriz.

Exemplos com Matrizes

Lendo uma matriz 4×5 do teclado:

Lendo matriz

```
int i, j;
int matriz[4][5]

for (i = 0; i < 4; i++) {
    for (j = 0; j < 5; j++) {
        printf ("Valor da linha %d, coluna %d: ", i, j);
        scanf ("%d", &matriz[i][j]);
    }
}
```

Exemplos com Matrizes

Imprimindo uma matriz 4×5 do teclado:

Escrevendo uma matriz

```
int i, j;
int matriz[4][5]

for (i = 0; i < 4; i++) {
    printf("Linha %d: ", i);

    for (j = 0; j < 5; j++) {
        printf("%d ", matriz[i][j];
    }

    printf("\n");
}
```