

MC-102 — Aula 08

Vetores e Strings

Instituto de Computação – Unicamp

27 de Março de 2012

Roteiro

- 1 Introdução
- 2 Vetores
- 3 Exemplos com Vetores
- 4 Strings
- 5 Exemplos Strings
- 6 Informações Extras sobre Strings

Como armazenar 3 notas?

```
float nota1, nota2, nota3;  
  
printf("Nota do aluno 1: ");  
scanf("%f", &nota1);  
printf("Nota do aluno 2: ");  
scanf("%f", &nota2);  
printf("Nota do aluno 3: ");  
scanf("%f", &nota3);
```

Vetores

Como armazenar 100 notas?

```
float nota1, nota2, nota3, /* .... */ nota100;
```

```
printf("Nota do aluno 1: ");
```

```
scanf("%f", &nota1);
```

```
printf("Nota do aluno 2: ");
```

```
scanf("%f", &nota2);
```

```
/* ... */
```

```
printf("Nota do aluno 100: ");
```

```
scanf("%f", &nota100);
```

Vetores — Definição

Coleção de variáveis do mesmo tipo referenciada por um nome comum.
(Herbert Schildt)

Vetor: também conhecido por “array”.

- acesso por meio de índice inteiro
- posições contíguas na memória
- tamanho pré-definido
- índices fora dos limites podem causar comportamento anômalo do programa.

Declaração de um vetor

`<tipo> identificador [<tamanho do vetor>];`

Exemplo

```
float notas[100];  
int medias[100];  
char nome[200];
```

Em

```
int medias[100];
```

podem ser armazenados **100 valores** (“variáveis”) **todos** do tipo **int**

Usando um vetor

Após declarada uma variável do tipo vetor, pode-se acessar uma determinada posição do vetor utilizando um valor inteiro.

identificador [**<posição>**];

- O acesso de um vetor em uma posição específica tem o mesmo comportamento que uma variável simples.
- A primeira posição de um vetor tem índice 0.
- A última posição de um vetor tem índice *<tamanho do vetor>* - 1.

Exemplo

```
int nota[10];  
int a;  
nota[5] = 95;  
a = nota[5];
```

Usando um vetor

identificador [**<posição>**];

- Você pode usar valores inteiros para acessar uma posição do vetor.
- O valor pode ser inclusive uma variável inteira.

Exemplo

```
int g, vet[10];  
for(g=0; g<10; g++)  
    vet[g]=5*g;
```

Vetores

- Na memória:

```
int d;  
int vetor[5];  
int f;
```

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-

Vetores

- Ao executar `vetor[3]=10;`:

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
					10		

Vetores

- O que ocorre se for executado os comandos:
vetor[5]=5;
vetor[-1]=1;

Vetores

- Ao executar
vetor[3]=10;
vetor[5]=5;
vetor[-1]=1;

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
	1				10		5

- Isto irá causar um erro no seu programa pois você está alterando valores de outras variáveis.
- Em muitos casos o seu programa irá ser encerrado (Segmentation Fault).

Questões importantes sobre vetores

- O tamanho do vetor é pré-definido (durante a execução do programa não pode ser alterado).
- Índices fora dos limites podem causar comportamento anômalo do código.

Como armazenar valores em um vetor?

```
int vet[100];  
  
vet[0] = 1;  
vet[1] = 2;  
.  
.  
.  
vet[98] = 99;  
vet[99] = 100;
```

- Pode ser! É prático?

Como armazenar valores em um vetor?

```
float vet[100];  
int i;  
  
for(i = 0; i < 100; i++){  
    vet[i] = i+1;  
}
```

- Agora sim!

Como armazenar valores em um vetor?

```
float parametros[] = {4.4, 10.2, 15.9, 19.5};
```

- Quantas posições tem o vetor parametros?

Como armazenar n (≤ 100) notas?

```
float nota[100];  
int n, i;  
  
printf("Número de alunos: ");  
scanf("%d", &n);  
  
for (i = 0; i < n; i++) {  
    printf("Nota do aluno %d: ", i+1);  
    scanf("%f", &nota[i]);  
}
```

- O programa acima está correto?

Como armazenar n (≤ 100) notas?

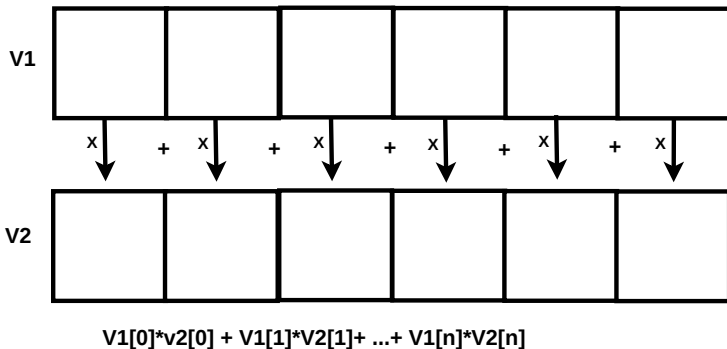
- Você deve testar se $n > 100$ para evitar erros de acesso fora dos limites do vetor!!!

```
float nota[100];
int n, i;

printf("Número de alunos: ");
scanf("%d", &n);
if(n>100){
    n=100;
    printf("\nNumero maximo de alunos alterado para 100");
}
for (i = 0; i < n; i++) {
    printf("Nota do aluno %d: ", i+1);
    scanf("%f", &nota[i]);
}
```

Produto Interno de dois vetores

- Ler dois vetores de dimensão 5 e computar o produto interno destes.



Produto Interno de dois vetores

Ler dois vetores de dimensão 5 e computar o produto interno destes.

```
int main(){
    double vetor1[5], vetor2[5], resultado;
    int i;

    for(i=0; i<5; i++){
        printf("Entre com valor %d para vetor 1:",i+1);
        scanf("%lf",&vetor1[i]);
    }
    printf("\n\n");
    for(i=0; i<5; i++){
        printf("Entre com valor %d para vetor 2:",i+1);
        scanf("%lf",&vetor2[i]);
    }

    //calculando o produto interno
    resultado = 0.0;
    for(i=0; i < 5; i++){
        resultado = resultado + ( vetor1[i]*vetor2[i] );
    }
    printf("\n\nO produto interno e: %lf\n",resultado);
}
```

Elementos Iguais

- Ler dois vetores com 5 inteiros cada.
- Checar quais elementos do segundo vetor são iguais a algum elemento do primeiro vetor.

Elementos Iguais

```
int main(){
    int vetor1[5], vetor2[5];
    int i, j,  umEmComum;

    for(i=0; i<5; i++){
        printf("Entre com valor %d para vetor 1:",i+1);
        scanf("%d",&vetor1[i]);
    }
    printf("\n\n");
    for(i=0; i<5; i++){
        printf("Entre com valor %d para vetor 2:",i+1);
        scanf("%d",&vetor2[i]);
    }

    umEmComum = 0;
    for(i = 0; i < 5 ; i++)
        for(j = 0; j < 5; j++)
            if(vetor1[i] == vetor2[j]){
                umEmComum = 1;
                printf("Posicao %d do vetor1 igual a %d do vetor2.\n",i,j);
            }
    if(!umEmComum)
        printf("Nenhum elemento em comum!\n");
}
```

Strings

- A linguagem C não possui o tipo *string* explicitamente mas podemos considerar um vetor de caracteres como uma *string*.
- Em C uma string é sempre terminada pelo caracter especial:
'\0'
- **Portanto sempre declare uma string com um caracter a mais do que precisa!**
- Se por exemplo estivermos trabalhando com strings de 10 caracteres:
`char st[11];`

Strings

- Para ler ou imprimir uma string do teclado usamos o operador especial `%s`.

```
int main(){
    char st[80];

    int a;

    printf("\nEntre com nome:");
    scanf("%s",st);
    printf("\nEntre com idade:");
    scanf("%d",&a);
    printf("\n Digitado: %s e %d\n",st,a);
}
```

- Note que para strings não é utilizado o `&` no comando `scanf`.

Strings

- Para ler ou imprimir uma string do teclado usamos o operador especial %s.

```
int main(){
    char st[80];
    char msgNome[] = "Entre com nome:";
    char msgIdade[] = "Entre com idade:";

    int a;

    printf("\n %s",msgNome);
    scanf("%s",st);
    printf("\n %s",msgIdade);
    scanf("%d",&a);
    printf("\n Digitado: %s e %d\n",st,a);
}
```

Strings - atenção

- A partir de um espaço em branco o scanf não lê mais os caracteres restantes, imprimindo apenas os caracteres que se encontram antes do espaço em branco
- Solução: `scanf("[A-Z a-z 0-9]",st);`

```
int main(){
    char st[80];

    printf("\nString:");
    scanf("%[A-Z a-z 0-9]",st);}
    printf("\n Digitado: %s \n",st);
}
```

Exemplos Strings

- Ler uma string de até 80 caracteres e salvar a inversa desta em um vetor.
- Imprimir a inversa da string lida.

```

int main(){
    char st1[80], stInversa[80];
    int i, j , tam;

    printf("Digite um texto (max. 80):");
    scanf("%s",st1);

    tam=0;
    while(st1[tam] != '\0' && tam < 80){
        tam++;
    }

    stInversa[tam] = '\0';
    i = 0;
    for(j = tam-1; j >= 0 ; j--){
        stInversa[j] = st1[i];
        i++;
    }

    printf("A inversa:%s\n", stInversa);
}

```

Exemplos Strings

- Ler uma string de até 80 caracteres e salvar a inversa desta em um vetor.
- Imprimir a inversa da string lida.
- **Não usar vetor adicional!**

Exemplos Strings

```
int main(){
    char st1[80], aux;
    int i, j, tam;

    printf("Digite um texto (max. 80):");
    scanf("%s",st1);

    tam=0;
    while(st1[tam] != '\0' && tam < 80){
        tam++;
    }

    i = 0;
    j = tam -1;
    while(i < j){
        aux = st1[i];
        st1[i] = st1[j];
        st1[j] = aux;
        i++; j--;
    }

    printf("A inversa:%s\n",st1);
}
```

fgets

- Ao usar o comando *scanf* para ler uma string, você deve garantir que foi alocado uma string de tamanho suficiente para armazenar todos os caracteres.
- Caso o usuário digite mais caracteres do que o tamanho alocado, isto poderá ocasionar erros!
- Uma alternativa para ler strings é usar o comando *fgets()*.

```
fgets( <identificador>, TAM, stdin);
```

onde *<identificador>* é a variável, TAM é um inteiro indicando até quantos caracteres devem ser lidos (até TAM-1 são lidos e um é reservado para o barra zero) e stdin é o dispositivo de entrada padrão (teclado).

Exemplos fgets

```
int main(){
    char st1[80];

    printf("Digite um texto (max. 80):");
    fgets(st1,80,stdin);

    printf("Digitado: %s",st1);
}
```

fgets - atenção

- Ao ler do teclado com o fgets o “Enter” é capturado e colocado na string como newLine:

`\n`

- A string terá a forma:

`"engenharia agricola\n\0"`

Exercício

- Escreva um programa que:
- 1 - preencha dois vetores (do mesmo tamanho) com valores fornecidos pelo usuário;
- 2 - intercale estes 2 vetores em um terceiro vetor.
- Ex: $v1 = \{1,3,5\}$ e $v2 = \{2,4,6\}$
- Saída: $v3 = \{1,2,3,4,5,6\} \Leftrightarrow v3 = \{v1[0],v2[0],v1[1],v2[1],v1[2],v2[2]\}$

Exercício

- Escreva um programa que lê uma string de até 50 caracteres, e imprime “Palindromo” caso a string seja um palindromo e “Nao Palindromo” caso contrário.
- OBS: Um palindromo é uma palavra ou frase, que é igual quando lida da esquerda para a direita ou da direita para a esquerda (acentos e espaços em brancos são descartados).
- Exemplo de palindromo: O lobo ama o bolo.