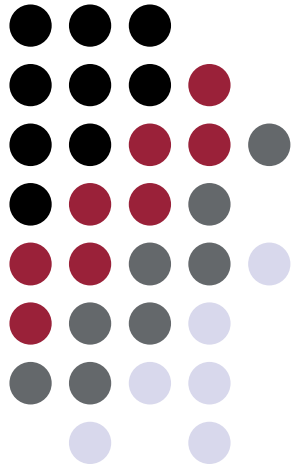
A07 Vetores e Strings

CSI030 – Programação de Computadores I

Prof. Dr. George H. G. Fonseca
Universidade Federal de Ouro Preto



Agenda

- Vetores
- Strings

Motivação

- Frequentemente, utilizamos poucas variáveis nos programas.

- Quando desejamos utilizar diversas variáveis de um mesmo tipo, pode ser inviável declarar cada uma das variáveis.

- Quando queremos armazenar uma quantidade de variáveis definida em função de requisição do usuário, não temos como declarar estas variáveis no momento da codificação.

Definiçãode Vetor

- Vetor é uma coleção de variáveis do mesmo tipo referenciadas por um nome comum.

- Vetores
  - Permitem acesso por meio de um índice inteiro;
  - São criados (alocados) em posições contíguas na memória;
  - Possuem tamanho (dimensão) pré-definido, ou seja, o tamanho de um vetor não pode ser alterado durante a execução do programa;
  - Índices fora dos limites causam comportamento imprevisível no programa.

Declaração de Vetor

```
tipo_vetor nome_vetor[tamanho_vetor];
```

- `tamanho_vetor` é um número inteiro ou uma variável do tipo **`int`**

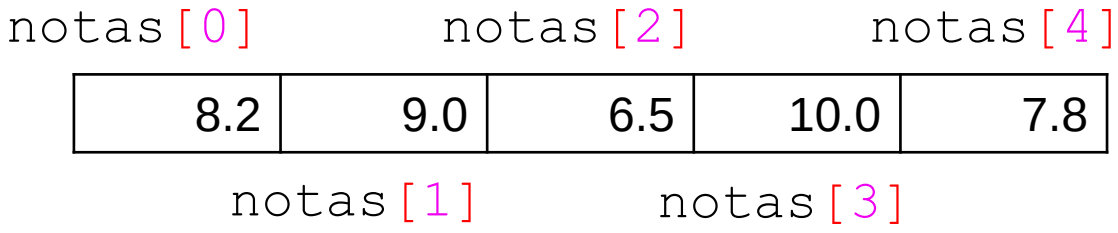
- Esta declaração cria `tamanho_vetor` variáveis do tipo **`tipo_vetor`**.

- As variáveis criadas pelo vetor são acessadas por:

```
nome_vetor[0];
nome_vetor[1];
...
nome_vetor[tamanho_vetor - 1];
```

Exemplo de vetor com índices inteiros

```
float notas[5];
notas[0] = 8.2;
notas[1] = 9.0;
notas[2] = 6.5;
notas[3] = 10.0;
notas[4] = 7.8;
```



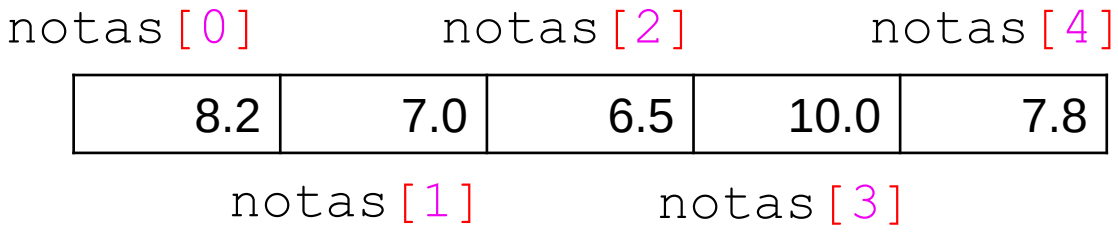

```
float valor = notas[1];
```

valor | 9.0

```
valor = valor - 2;
```

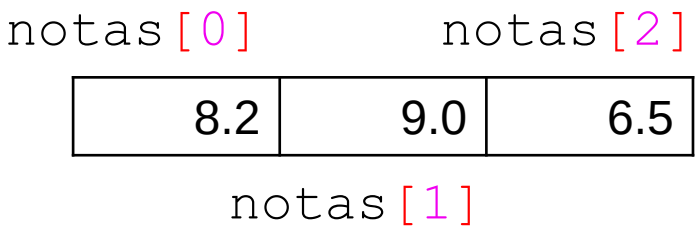
valor | 7.0

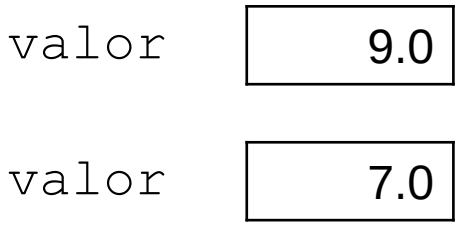
```
notas[1] = valor;
```

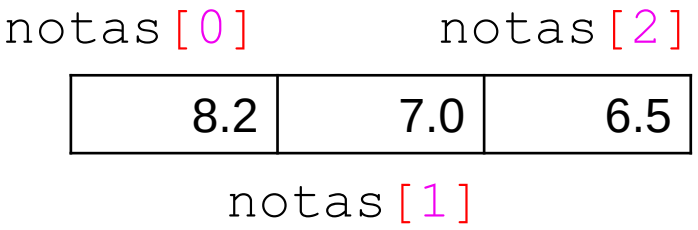


Exemplo de vetor com variáveis nos índices

```
int tamanho = 3;
int indice = 0;

float notas[tamanho];
notas[indice] = 8.2;
++indice;
notas[indice] = 9.0;
++indice;
notas[indice] = 6.5;

float valor = notas[1];

valor = valor - 2;

notas[1] = valor;
```

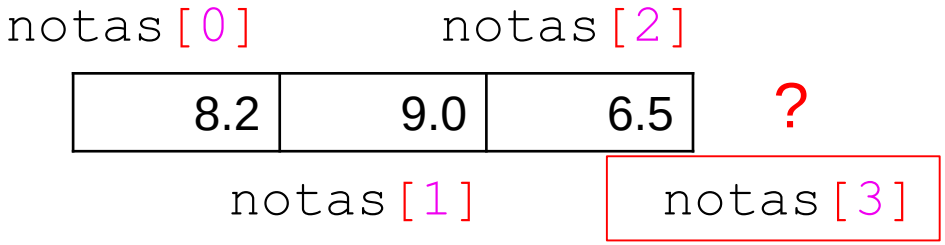






Acesso a índice fora do limite

```
int tamanho = 3;
int indice = 0;
float notas[tamanho];
notas[0] = 8.2;
notas[1] = 9.0;
notas[2] = 6.5;
notas[3] = 10.0;
```



Este acesso altera uma posição de memória inexistente
e causa Segmentation Fault (a.k.a. erro)

Exemplo: Ler, preencher e imprimir um vetor

```c
int i, tamanho =5;
float notas[tamanho];

for (i=0; i< tamanho; i++){
    printf("Digite o valor da posição notas[%d]:\n",i);
    scanf("%f", &notas[i]);
}

printf("Valores lidos:\n");
for (i=0; i< tamanho; i++){
    printf("notas[%d] = %.2f\n", i, notas[i]);
}
```

Exemplo:
Produto interno de dois vetores

- Crie 2 vetores de dimensão 5, leia os valores para estes vetores e calcule o produto interno destes vetores.

- Produto interno de dois vetores é a soma dos produtos entre os elementos em posições equivalentes dos vetores.

- Por exemplo, o produto interno dos vetores:

  {2, 3, 4, 5, 6} e {3, 5, 6, 1, 5}

  é igual a

  2*3 + 3*5 + 4*6 + 5*1+ 6*5 = 80

Exemplo:
Produto interno de dois vetores

```
int main(void){
    int i;
    double vetor1[5], vetor2[5], resultado;
    for(i=0; i<5; i++){
        printf("Digite o valor para vetor1[%d]:\n",i);
        scanf("%lf", &vetor1[i]);
    }
    for(i=0; i<5; i++){
        printf("Entre com valor para vetor2[%d]:\n",i);
        scanf("%lf", &vetor2[i]);
    }

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

Vetores como parâmetro de sub-rotina

- Vetores podem ser passados como parâmetros em subrotinas.
- Ao se passar um vetor como parâmetro, deve-se passar, também, o seu tamanho.
- Ao se passar um vetor como parâmetro de sub-rotina, **não é criado um novo vetor** local na sub-rotina.
- Isto significa que **os valores de um vetor são alterados dentro de uma sub-rotina!**

Vetores como parâmetro de sub-rotina

```
void proc5(int vet[], int tam){
    int i;
    for(i=0; i<tam; i++)
        vet[i]=5;
}

int main(void){
    int i, tamanho = 10, x[tamanho];
    for(i=0; i< tamanho; i++)
        x[i]=8;

    proc5(x, tamanho);
    for(i=0; i< tamanho; i++)
        printf("%d\n", x[i]);
    return 0;
}
```

Vetores e retorno de sub-rotina

- Vetores não podem ser retornados por sub-rotinas
- Pode-se utilizar o fato de que vetores são alterados dentro de sub-rotinas para simular o retorno de um vetor por uma sub-rotina:
  - basta que um procedimento receba o vetor e o altere o seu conteúdo.
- O conteúdo será visto por quem chamou com as alterações.

Vetores e retorno de sub-rotina

```
void leVet(int vet[], int tam){
    int i;
    printf("Digite %d numeros: ", tam);
    for(i = 0; i < tam; i++)
        scanf("%d", &vet[i]);
}

void escreveVet(int vet[], int tam){
    int i;
    for(i=0; i< tam; i++)
        printf("vet[%d] = %d\n", i, vet[i]);
}

int main(int){
    int vet1[10], vet2[20];
    leVet(vet1,10);
    leVet(vet2,20);
    escreveVet(vet1,10);
    escreveVet(vet2,20);
    return 0;
}
```

Inicialização de vetores

- Assim como variáveis dos demais tipos, vetores podem ser declarados e inicializados ao mesmo tempo.

```
tipo nome[] = {elementos separados por vírgula};
```

- Exemplos:

```
double vet1[] = {2.3, 3.4, 4.5, 5.6};
int vet2[] = {5, 4, 3, 10, -1, 0};
```

- O tamanho do vetor é definido pela quantidade de elementos na lista entre chaves.

Exemplo de declaração e inicialização de vetores

```
int main(void){

    double vet1[] = {2.3, 3.4, 4.5, 5.6};
    int vet2[] = {5, 4, 3, 10, -1, 0};
    int i;

    for(i=0; i<4; i++)
        printf("%lf\n", vet1[i]);

    for(i=0; i<6; i++)
        printf("%d\n", vet2[i]);
    return 0;
}
```

Strings
(Cadeia de Caracteres)

- A linguagem C não possui o tipo string (cadeia de caracteres) explicitamente, mas podemos considerar um vetor de caracteres como uma string.

- Em C, uma string é sempre terminada pelo caractere especial: `'\0'`

- Logo, ao declararmos um vetor de caracteres, devemos somar 1 à quantidade desejada de caracteres!

- Exemplo:

```
char st1[7] = "string";
```

Declaração e inicialização de Strings

- Podemos declarar e inicializar um vetor de caracteres de duas formas:

```
char st1[] = "string";
char st2[] = {'s', 't', 'r', 'i', 'n', 'g', '\0'};
```

- O tamanho da variável será a quantidade de caracteres atribuídos MAIS UM, devido ao caractere `'\0'`.

Leitura e impressão de Strings

- Uma cadeia de caracteres é lida ou impressa com o modificador `%s`

- O armazenamento da leitura é interrompido ao se encontrar um espaço, mesmo que existam mais caracteres depois do espaço.

- Para ler uma cadeia de caracteres contendo espaços, indique que a cadeia deve terminar com a quebra de linha, assim:

```
%[^\n]s
```

- A impressão da cadeia de caracteres é feita até o último caractere antes de `'\0'`

Cópia de Strings

- Strings podem ser copiadas através da função strcpy da biblioteca `string.h`

```
char st1[ ] = "string";
char st2[10];
char st3[31];
strcpy(st2, st1);
strcpy(st3, "Programacao de Computadores 1");
```

Função sizeof e strlen

- A função sizeof(Tipo) retorna o tamanho de um elemento de Tipo, em bits

- Pode ser usada para calcular o tamanho do vetor/string:

```
int vetor[] = {3, 3, 5, 2, 4};
int tamanho = sizeof(vetor) / sizeof(int);
```

- Já strlen retorna o comprimento de uma string (requer a inclusão da biblioteco string.h)

```
char string[] = "Hello world";
int tamanho = sizeof(string) -1;
//tamanho = strlen(string); (#include <string.h>)
```

Exercício

- Crie uma função com a assinatura:

```
int maiorValor(int vetor[ ], int tamanho);
```

que receba como parâmetros um vetor e seu tamanho e devolva o maior valor armazenado no vetor.

Exercício

```c
#include <stdio.h>
#include <stdlib.h>

int maiorValor(int vetor[], int tamanho) {
    int maior = vetor[0];
    int i;
    for(i = 0; i < tamanho; ++i) {
        if(vetor[i] > maior) {
            maior = vetor[i];
        }
    }
    return maior;
}

int main() {
    int valores[] = {3, 4, 2, 9, 1};
    int maior, tamanho = 5;
    maior = maiorValor(valores, tamanho);
    printf("Maior = %d\n", maior);
    return 0;
}
```

Exercício

- Crie uma função com a assinatura:

```
float media(int vetor[ ], int tamanho);
```

que receba como parâmetros um vetor e seu tamanho e retorne a média dos valores armazenados no vetor.

Exercício

```c
#include <stdio.h>

float media(int vetor[], int tamanho) {
    int i;
    float media, soma = 0;
    for(i = 0; i < tamanho; ++i) {
        soma += vetor[i];
    }
    media = soma / tamanho;
    return media;
}

int main() {
    int vetor[] = {3, 3, 5, 2, 4};
    //int tamanho = 5;
    int tamanho = sizeof(vetor) / sizeof(int);
    printf("Media: %.2f\n", media(vetor, tamanho));
    return 0;
}
```

Exercício

- Escreva uma função com assinatura

```
int ordenado(int vetor[], int tam)
```

que retorne 1 caso o vetor esteja em ordem crescente e 0 caso contrário

Exercício

```c
#include <stdio.h>

int ordenado(int vetor[], int tam) {
    int i;
    for(i = 0; i < tam - 1; ++i) {
        if(vetor[i] > vetor[i + 1]) {
            return 0;
        }
    }
    return 1;
}

int main() {
    //int vetor[] = {1, 3, 5, 9, 11};
    int i, tam;
    printf("Informe o numero de elementos: ");
    scanf("%d", &tam);
    int vetor[tam];
    printf("Informe %d elementos: \n", tam);
    for(i = 0; i < tam; ++i) {
        scanf("%d", &vetor[i]);
    }
    int ord = ordenado(vetor, tam);
    if(ord == 1)
        printf("Vetor esta em ordem crescente!");
    else
        printf("Vetor NAO esta em ordem crescente!");
    return 0;
}
```

Exercício

- Crie uma função com a assinatura:

```
int palindromo(int vetor[ ], int tamanho);
```

que receba como parâmetros um vetor e seu tamanho e retorne:

- 1, os valores no vetor formam um palíndromo;
- 0, caso contrário.

- Um vetor é palíndromo se a sequência de valores do menor para o maior índice é igual à sequência de valores do maior para o menor índice.
  Ex.:
  {1, 2, 3, 4, 3, 2, 1} é palíndrome, mas {1, 2, 3, 1, 2, 3} não é.

Exercício

```c
#include <stdio.h>

int palindromo(int vetor[], int tamanho) {
    int i;
    for(i = 0; i < tamanho / 2; ++i) {
        if(vetor[i] != vetor[tamanho - i - 1]) {
            return 0;
        }
    }
    return 1;
}

int main() {
    int vetor[] = {1, 2, 3, 2, 1};
    int tamanho = 5;
    if(palindromo(vetor, tamanho))
        printf("Vetor eh palindromo\n");
    else
        printf("Vetor NAO eh palindromo\n");
    return 0;
}
```

Exercício

- Crie um procedimento com a assinatura:

```
void substitui(char str[ ], char antigo, char novo);
```

que receba como parâmetros uma string, um caractere a ser substituído e o novo caractere a entrar no lugar do antigo e faça as devidas substituições na string.

Codifique o programa completo, fazendo a leitura dos dados na função main.

Exercício

```c
#include <stdio.h>
#include <string.h>

void substitui(char str[], char antigo, char novo) {
    int i, tam = strlen(str);
    for(i = 0; i < tam; ++i) {
        if(str[i] == antigo) {
            str[i] = novo;
        }
    }
}

int main() {
    char string[100];
    char antigo, novo;
    printf("Informe a string: \n");
    gets(string);
    printf("Caractere a ser substituido: \n");
    scanf(" %c", &antigo);
    printf("Novo caractere: \n");
    scanf(" %c", &novo);
    substitui(string, antigo, novo);
    printf("Nova string:\n%s\n", string);
    return 0;
}
```

Exercício

- Faça um programa que conte o número de vogais em uma cadeia de caracteres

Exercício

```
 1  #include <stdio.h>
 2
 3  int main() {
 4      char texto[100] = "Programacao de Computadores 1";
 5      int i;
 6      int contador = 0;
 7      for (i=0; texto[i]!= '\0'; ++i) {
 8          if (texto[i] == 'a' || texto[i] == 'A' ||
 9                  texto[i] == 'e' || texto[i] == 'E' ||
10                  texto[i] == 'i' || texto[i] == 'I' ||
11                  texto[i] == 'o' || texto[i] == 'O' ||
12                  texto[i] == 'u' || texto[i] == 'U')
13              contador++;
14      }
15      printf("A frase tem %d vogais.\n", contador);
16      return 0;
17  }
```

Exercício

- Escreva uma função com assinatura:

```
int compara(char str1[ ], char str2[ ]);
```

que retorne 1 se as strings str1 e str2 são iguais e 0 caso contrário

Exercício

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int compara(char str1[], char str2[]) {
    int len1 = strlen(str1);
    int len2 = strlen(str2);
    int i;
    if(len1 != len2) {
        return 0;
    }
    for(i = 0; i < len1; ++i) {
        if(str1[i] != str2[i])
            return 0;
    }
    return 1;
}
int main()
{
    char str1[30], str2[30];
    int res;
    printf("Informe duas strings:\n");
    scanf(" %[^\n]s", &str1);
    scanf(" %[^\n]s", &str2);
    res = compara(str1, str2);
    if(res == 1)
        printf("Strings iguais!");
    else
        printf("Strings diferentes!");
    return 0;
}
```

Bibliografia

- Anido, R., Notas de aula. UNICAMP, disponível em http://www.ic.unicamp.br/~ranido/mc102/, acessado em Maio de 2015.

- Mota, V. F., Notas de aula. UFMG. Disponível em https://sites.google.com/site/virginiaferm/home/disciplinas acessado em Maio de 2015.

- Deitel, P, Deitel, H. C How to Program. 6a Ed. Pearson, 2010.