Introdução à C/C++ para programadores Python - Parte 2

Algoritmos e Programação 2

Prof. Dr. Anderson Bessa da Costa
Universidade Federal de Mato Grosso do Sul

Variáveis Compostas Homogêneas

```
int a;
```

- Ao declararmos uma variável, podemos armazenar apenas um valor por vez
- **Variáveis compostas homogêneas** permitem agrupar várias informações dentro de “uma mesma variável”
    - Mesmo tipo

Vetores

```
int v[100];
```

- Conjunto de variáveis do **mesmo tipo** e **mesmo identificador** (nome)
    - São **alocadas sequencialmente** na memória
- O que as distingue é um **índice** que referencia sua localização dentro da estrutura
    - De 0 (zero) até o tamanho do vetor menos um (*n-1*)

Exemplo 1: Notas

Faça um algoritmo que leia todas as notas de um colégio inteiro e verifique se a média do colégio está acima de 7. Obs.: Suponha que existam 200 estudantes.

Exemplo 1: Solução

```c
#include <stdio.h>

int main () {
    int i;
    float notas[200], media = 0;

    printf("Entre com as notas: ");
    for (i = 0; i < 200; i++) {
        scanf("%f", &notas[i]);
        media += notas[i];
    }
    media = media / 200;

    printf("media = %f\n", media);
    if (media >= 7.0)
        printf("Media da escola maior que 7.\n");
    else
        printf("Media da escola menor que 7.!\n");

    return 0;
}
```

Atribuindo Valores ao Vetor

```
float notas[5];
notas[2] = 8.5;
```

- Outra forma é referenciar posições do vetor utilizando variáveis do tipo inteiro

```
float notas[5];
int i = 2;
notas[i] = 8.5;
```

Exemplo 2: Encontre o maior

Faça um programa que declare um vetor de inteiros de tamanho 10, leia 10 valores e armazene-os no vetor. Após isso, encontre e imprima o maior elemento do vetor.

Exemplo 2: Solução

```c
#include <stdio.h>
#define TAMANHO 10

int main () {
    int v[TAMANHO], i, maior;

    printf("Entre com os valores: ");
    for(i = 0; i < TAMANHO; i++) {
        scanf("%d", &v[i]);
    }

    maior = v[0];
    for(i = 1; i < TAMANHO; i++) {
        if(maior < v[i]) {
            maior = v[i];
        }
    }
    printf("\nmaior = %d\n", maior);

    return 0;
}
```

Matrizes em C/C++

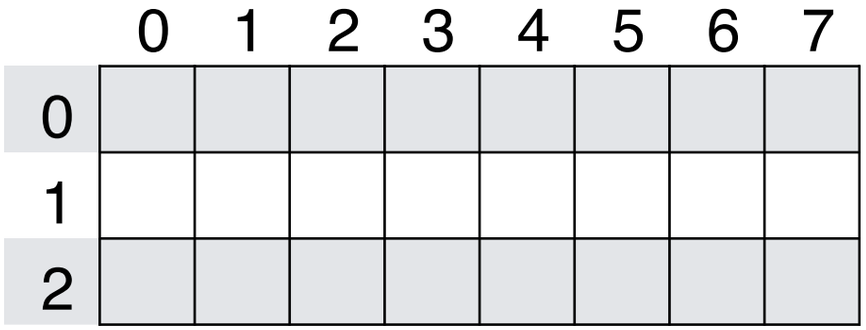
- Para declarar matrizes em C/C++, deve-se definir a quantidade de elementos em cada dimensão

```
tipo nome_vetor[n_linhas][n_colunas];
```

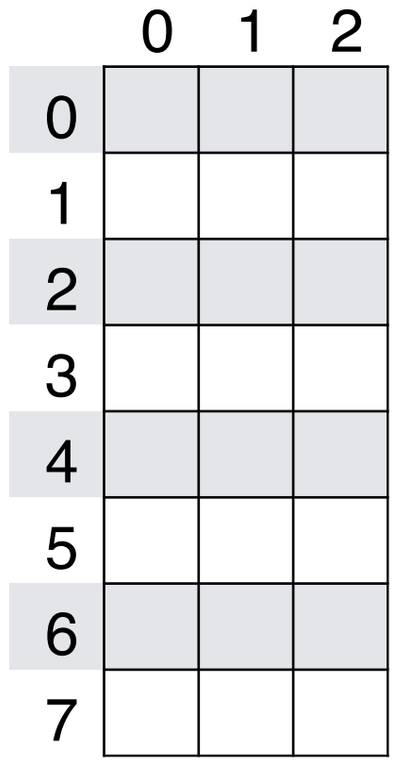
Representação Visual Matrizes

- Considerando-se o problema de ler 3 notas de uma turma de 8 alunos

```
int notas[3][8];
```




```
int notas[8][3];
```



Preenchendo uma Matriz

- Exemplo leitura das notas de cada prova para cada aluno:

```
for (i = 0; i < 8; i++) {
  for (j = 0; j < 3; i++) {
    printf("Digite a nota da prova %d do aluno %d", j, i);
    scanf("%d", &notas[i][j]);
  }
}
```

Percorrendo uma Matriz

- Consiste em passar por todas as posições
  - Uma estrutura de repetição para cada dimensão da matriz

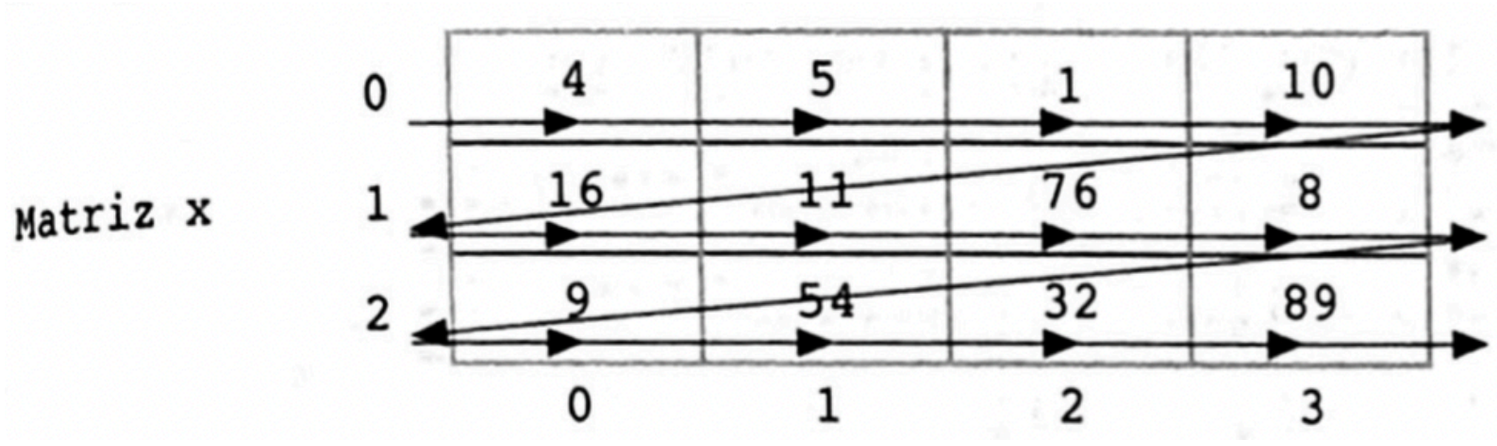
Matriz x

	0	1	2	3
0	4	5	1	10
1	16	11	76	8
2	9	54	32	3

Percorrendo uma Matriz: Forma 1

- **Forma 1:** Mostrar elementos por **linha**

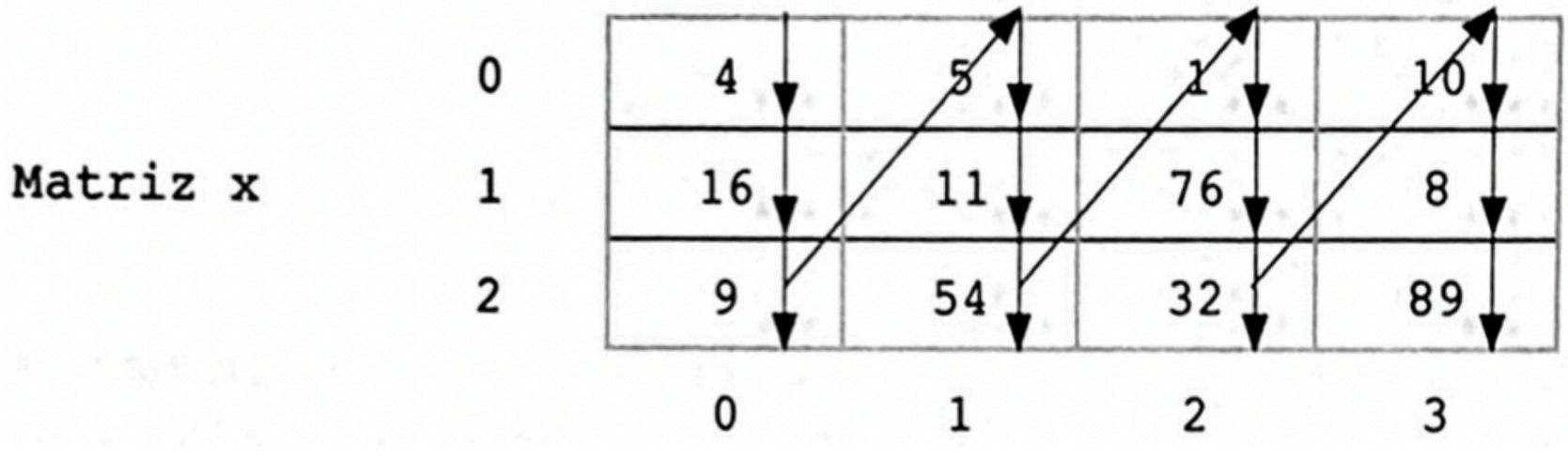
```
for (i = 0; i < 3; i++) {
    printf("Elementos da linha %d", i);
    for (j = 0; j < 4; j++)
        printf("%d", x[i][j])
}
```



Percorrendo uma Matriz: Forma 2

- **Forma 2:** Mostra elementos por **coluna**

```
for (i = 0; i < 4; i++) {
    printf("Elementos da coluna %d", i);
    for (j = 0; j < 3; j++)
        printf("%d", x[j][i])
}
```



Modularização

Funções

- Princípio **programação procedural** é dividir um programa em funções
  - Dividem grandes tarefas em tarefas menores
- Reusabilidade

Chamando uma Função

- Solicitamos que o programa desvie o controle e passe a executar as instruções da função
  - ao término desta, volte o controle para a posição seguinte à da chamada
- Você já escreveu programas que chamam funções

Exemplo 3: Fahrenheit para Celsius

```c
#include <stdio.h>

float converte_fahr_para_celsius(float fahr);

int main () {
    float c, f;

    printf("Digite a temperatura em graus Fahrenheit: ");
    scanf("%f", &f);

    c = converte_fahr_para_celsius(f);

    printf("Celsius = %.2f\n", c);
    return 0;
}

float converte_fahr_para_celsius(float fahr) {
    float c;
    c = (fahr - 32.0) * 5.0/9.0;
    return c;
}
```

CELSIUS TO FAHRENHEIT

$$T_F = \left(\frac{9}{5}T_C\right) + 32$$

FAHRENHEIT TO CELSIUS

$$T_C = \frac{5}{9}\left(T_F - 32\right)$$

Componentes de uma Função

Componentes de uma função:

1. Protótipo da função
2. Definição da função
3. Chamada à função

Protótipo da Função

- Precede definição e chamada
- Fornece ao compilador informações sobre **nome da função**, **tipo de retorno**, **número e o tipo dos argumentos**
- Sem o protótipo, o compilador não tem como verificar e checar se há erros em seu uso
- O nosso exemplo declara o seguinte protótipo:

```
float converte_fahr_para_celsius(float fahr);
```

Passagem de Argumentos por Valor x por Referência

Passagem por Valor

- Em nosso exemplo, a função cria uma nova variável para receber o valor passado
- Sua declaração indica que o valor enviado será armazenado na variável *fahr*, criada quando a função inicia sua execução e destruída quando ela termina

Exemplo 4: a antes de depois

```c
#include <stdio.h>

void f(int a);

int main () {
    int a;
    a = 2;

    printf("a antes = %d\n", a);

    f(a);

    printf("a depois = %d\n", a);

    return 0;
}

void f(int a) {
    a += 1;
}
```

```
% ./a.out
a antes = 2
a depois = 2
```

Exemplo 5: b[] antes e depois

```c
#include <stdio.h>

void f(int b[]);

int main () {
    int b[2] = {2, 10};

    printf("antes\nb[0] = %d, b[1] = %d\n\n", b[0], b[1]);
    f(b);
    printf("depois\nb[0] = %d, b[1] = %d\n", b[0], b[1]);

    return 0;
}

void f(int b[]) {
    b[0] += 1;
    b[1] += 1;
}
```

```
% ./a.out
antes
b[0] = 2, b[1] = 10

depois
b[0] = 3, b[1] = 11
```

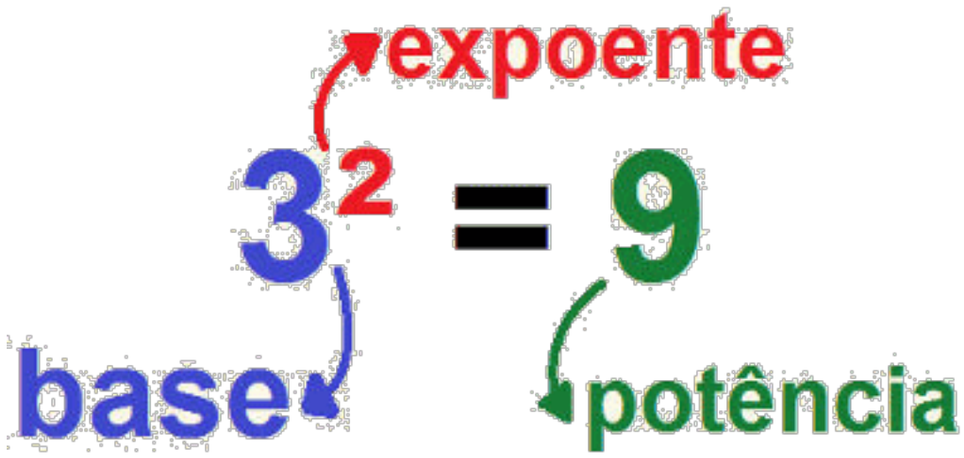
Passagem por Valor x Referência

- Por enquanto precisamos saber somente que:
  - **Passagem por valor:** Para as variáveis, exceto vetores, qualquer alteração do parâmetro dentro da função não irá causar nenhuma modificação fora da função
  - **Passagem por referência:** Para vetores, alterar o conteúdo do vetor dentro de uma função irá alterar também o conteúdo do argumento do vetor passado

Exemplo 6: Potência

Escreva um programa que leia dois valores x e n, e calcule a potência de x^n.

Exemplo 6: Solução




```
double potencia (double base, int expoente) {
  ...
}
```

Exemplo 6: Solução (cont.)

```c
#include <stdio.h>

double potencia(double base, int expoente);

int main() {
    double base, p;
    int expoente;

    printf("Entre com a base e o expoente: ");
    scanf("%lf %d", &base, &expoente);

    p = potencia(base, expoente);
    printf("O resultado eh %lf!\n", p);
    return 0;
}

double potencia(double base, int expoente) {
    int i, pot = 1;

    for(i = 0; i < expoente; i++)
        pot = pot * base;

    return pot;
}
```

Strings

Strings na Linguagem C/C++

- C/C++ não possui um tipo nativo de dado *string*
- Armazena uma cadeia de caracteres em um vetor
  - Cada posição representa um caractere
  - O fim de uma cadeia é identificado por meio do caractere nulo, ou seja, por meio do ‘\0’
- Deve-se declarar sempre o vetor com uma posição a mais para armazenar o caractere nulo

Exemplo 7: Cadeia

```
char palavra[7];
```

índice	...	0	1	2	3	4	5	6	...
valor	...	C	A	D	E	I	A	\0	...
posição memória	...	863	864	865	866	867	868	869	...

Cadeia de Caracteres

- A variável **palavra** pode ocupar qualquer posição disponível na memória
  - Entretanto, posições são adjacentes na memória
- Cada caractere ocupa 1 byte
- Para manipular deve-se utilizar a biblioteca **string.h**

Inicializando Cadeias de Caracteres

- Inicialização no momento da declaração

```
char nome[] = "Programa";
```

- Inicialização por meio da atribuição (depois da declaração)

```
char vet1[10], vet2[5] = "casa";
strcpy(vet1, "Programa");
strcpy(vet1, vet2);
```

Inicializando Cadeias de Caracteres (cont.)

- Inicialização por meio do teclado

```
char vet[10];
scanf("%s", vet);
```

Obs.: Armazena todos os símbolos digitados até a ocorrência do primeiro espaço em branco

ou

```
fgets(vet, 10, stdin);
```

Concatenando Cadeia de Caracteres

```
strcat(str1, str2);
```

- A função **strcat** concatena a cadeia de caracteres **str2** à **str1**

```
strncat(str1, str2);
```

- A função **strncat** concatena os n primeiros caracteres da cadeia **str2** à **str1**

Comparando Cadeia de Caracteres

```
resultado = strcmp(cadeia1, cadeia2);
```

- A função **strcmp** compara duas cadeias e retorna um número inteiro, que poderá ser:
  - 0 se cadeias forem iguais;
  - < 0 se **cadeia1** for alfabeticamente menor que a **cadeia2**;
  - > 0 se **cadeia1** for alfabeticamente maior que a **cadeia2**;
- Essa função considera letras maiúsculas símbolos diferentes de letras minúsculas

Exemplo 8: Strcmp

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

int main () {
    char cor[10];

    printf("Entre com uma cor: ");
    scanf("%s%*c", cor);

    if(strcmp(cor, "verde") == 0) {
        printf("Cor do melhor time do mundo!\n");
    }

    printf("A cor escolhida e %s!\n", cor);

    return 0;
}
```

Descobrindo o Número de Caracteres de uma Cadeia

```
tamanho = strlen(cadeia);
```

- A função **strlen** retorna para a variável **tamanho** o número de caracteres da **cadeia**
- O caractere que indica final da cadeia de caracteres, '\0', não entra no total de caracteres da cadeia

Referências

- ASCENCIO, Ana Fernanda Gomes; DE CAMPOS, Edilene Aparecida Veneruchi. Fundamentos da programação de computadores. 3.ed. Pearson Education do Brasil, 2012.
- DEITEL, Paul; DEITEL, Harvey; C++ How to Program (9 edition). Pearson, 2016.