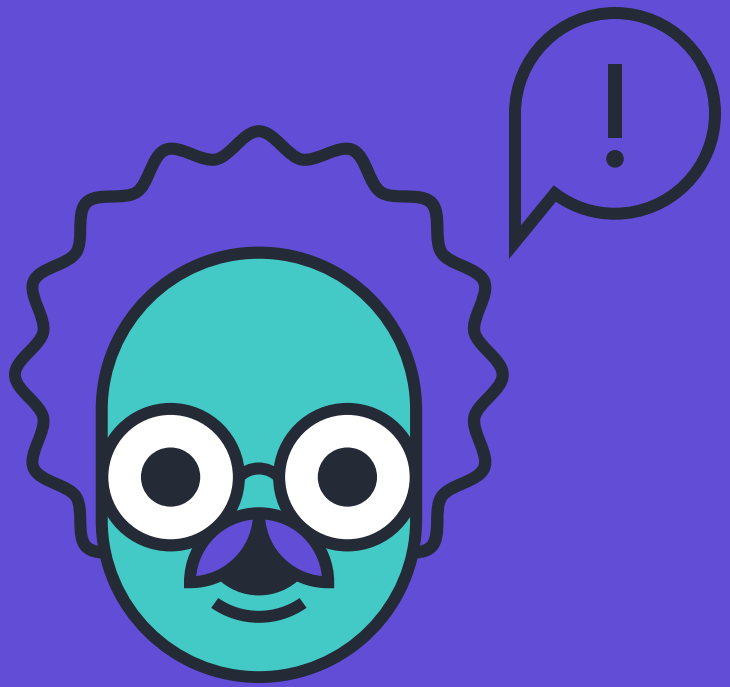
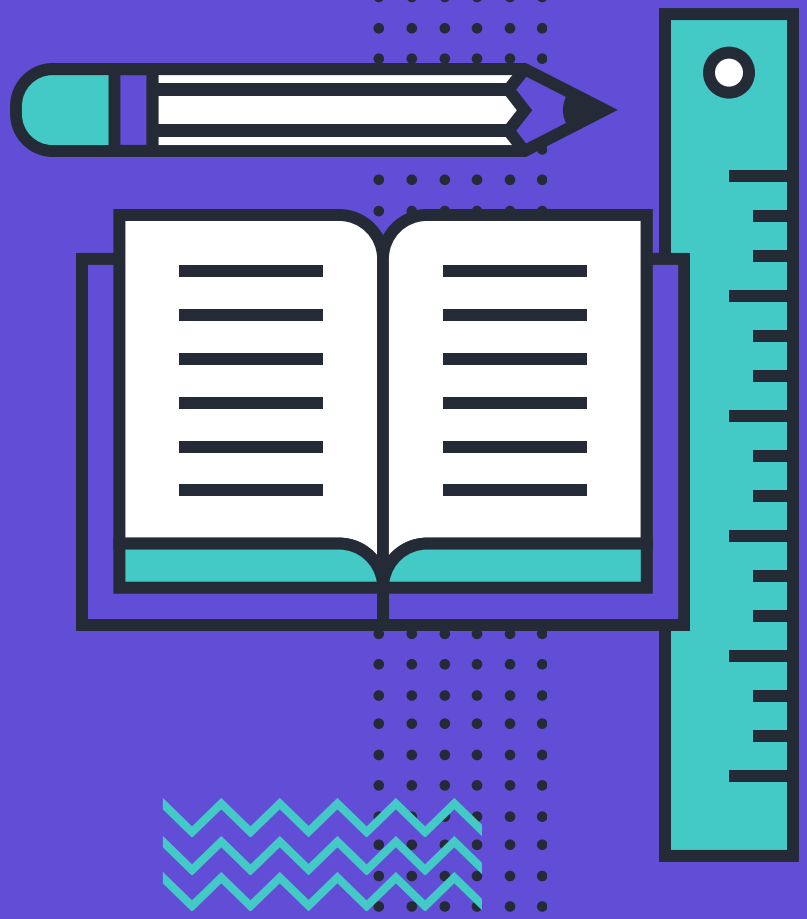


PROGRAMAÇÃO 101

8 - VETORES E MATRIZES

INTRODUÇÃO



POR QUE UTILIZAR

EXISTIRÃO MOMENTOS EM QUE O SEU PROGRAMA EXIGIRÁ UMA QUANTIDADE MUITO GRANDE DE VARIÁVEIS PARA EXERCEREM FUNÇÕES MUITO PARECIDAS

IMAGINE UM PROGRAMA EM QUE VOCÊ TENHA QUE LER A NOTA DE 5 ALUNOS E DEPOIS IMPRIMIR AS QUE SÃO MAIORES QUE A MÉDIA

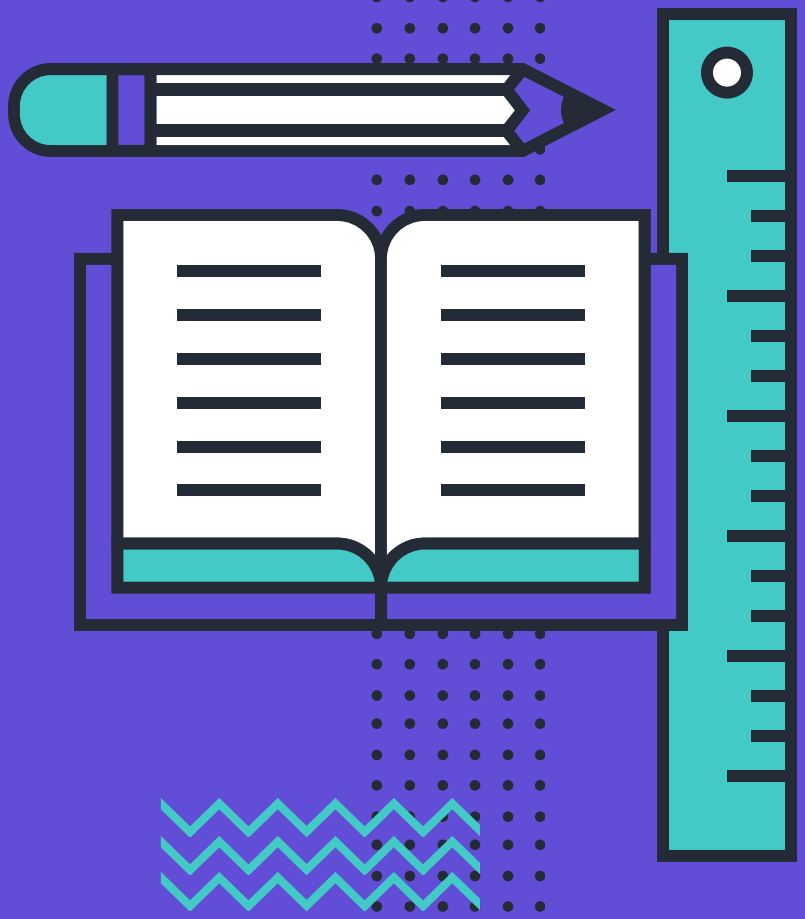
```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    float n1, n2, n3, n4, n5;
    float media;

    printf("Digite a nota de 5 estudantes: ");
    scanf("%f",&n1);
    scanf("%f",&n2);
    scanf("%f",&n3);
    scanf("%f",&n4);
    scanf("%f",&n5);

    media = (n1+n2+n3+n4+n5)/5.0;

    if(n1 > media) printf("nota: %f\n", n1);
    if(n2 > media) printf("nota: %f\n", n2);
    if(n3 > media) printf("nota: %f\n", n3);
    if(n4 > media) printf("nota: %f\n", n4);
    if(n5 > media) printf("nota: %f\n", n5);

    return 0;
}
```

OBSERVAÇÃO

O PROGRAMA ANTERIOR REPRESENTA UMA SOLUÇÃO POSSÍVEL PARA O PROBLEMA. O INCONVENIENTE DESSA SOLUÇÃO É A QUANTIDADE DE VARIÁVEIS PARA ADMINISTRAR E O USO REPETITIVO DE COMANDOS QUASE IDÊNTICOS.

ESSA SOLUÇÃO SERIA INVIÁVEL PARA UMA TURMA DE 100 ALUNOS

VETORES



O QUE É UM VETOR?

É UM CONJUNTO DE VARIÁVEIS DE UM MESMO TIPO, COM A VANTAGEM DE ESTAREM TODOS ASSOCIADOS AO MESMO NOME E IGUALMENTE ACESSÍVEIS POR UM ÍNDICE

VETORES

RELEMBRANDO O EXEMPLO ANTERIOR, EM QUE DEVERÍAMOS CRIAR 100 VARIÁVEIS QUE SÃO DO MESMO TIPO.

ESSA SOLUÇÃO PERMITIRIA USAR APENAS UM NOME (NOTAS, POR EXEMPLO) DE VARIÁVEL PARA REPRESENTAR TODAS AS NOTAS DOS ALUNOS, EM VEZ DE UM NOME PRA CADA VARIÁVEL

DECLARANDO UM VETOR

FORMA GERAL:

tipo_dado nome_array [tamanho];

EXEMPLO:

float notas[100];

IMPORTANTE

O TAMANHO DE UM VETOR É SEMPRE UM VALOR OU EXPRESSÃO POSITIVA E CONSTANTE

INICIALIZANDO UM VETOR NA SUA DECLARAÇÃO

FORMA GERAL:

```
tipo_dado nome_array [tamanho] = {dados};
```

EXEMPLO:

```
int vet[5] = {15, 12, 9, 1, 35};
char str_1 [10] = {'J', 'o', 'a', 'o', '\0'};
char str_2 [10] = {"Joao"};
```

INICIALIZANDO UM VETOR SEM TAMANHO

FORMA GERAL:

```
tipo_dado nome_array [] = {dados};
```

EXEMPLO:

```
int vet[5] = {15, 12, 9, 1, 35};
char str_1 [10] = {'J', 'o', 'a', 'o', '\0'};
char str_2 [10] = {"Joao"};
```

ACESSANDO UM CONTEÚDO DO VETOR

COMO A VARIÁVEL QUE ARMAZENA A NOTA DE UM ALUNO POSSUI AGORA O MESMO NOME QUE AS DEMAIS NOTAS DOS OUTROS ALUNOS, O ACESSO AO VALOR DE CADA NOTA É FEITO UTILIZANDO UM ÍNDICE

EXEMPLO:

```
float notas[100];
notas[0] = 81;
notas[1] = 55;
...
notas[99] = 72;
```

```c
#include<stdio.h>

int main ( ) {
    int i;
    int vetor[100];

    for(i=0; i<100; i++){
        vetor[i] = 0;
     }

    return 0;
}
```

```c
#include<stdio.h>

int main ( ) {
    int i;
    int vetor[100];

    for(i=0; i<100; i++){
        vetor[i] = i;
    }

    return 0;
}
```

```c
#include<stdio.h>

int main ( ) {
    int i;
    int vetor[100];

    for(i=0; i<100; i++){
        scanf("%d", &vetor[i]);
    }

    return 0;
}
```

IMPRIMINDO

```
#include<stdio.h>

int main ( ) {
    int i;
    int vetor[100];

    for(i=0; i<100; i++){
        scanf("%d", &vetor[i]);
    }

    for(i=0; i<100; i++){
        printf("%d ", vetor[i]);
    }
    printf("\n");

    return 0;
}
```

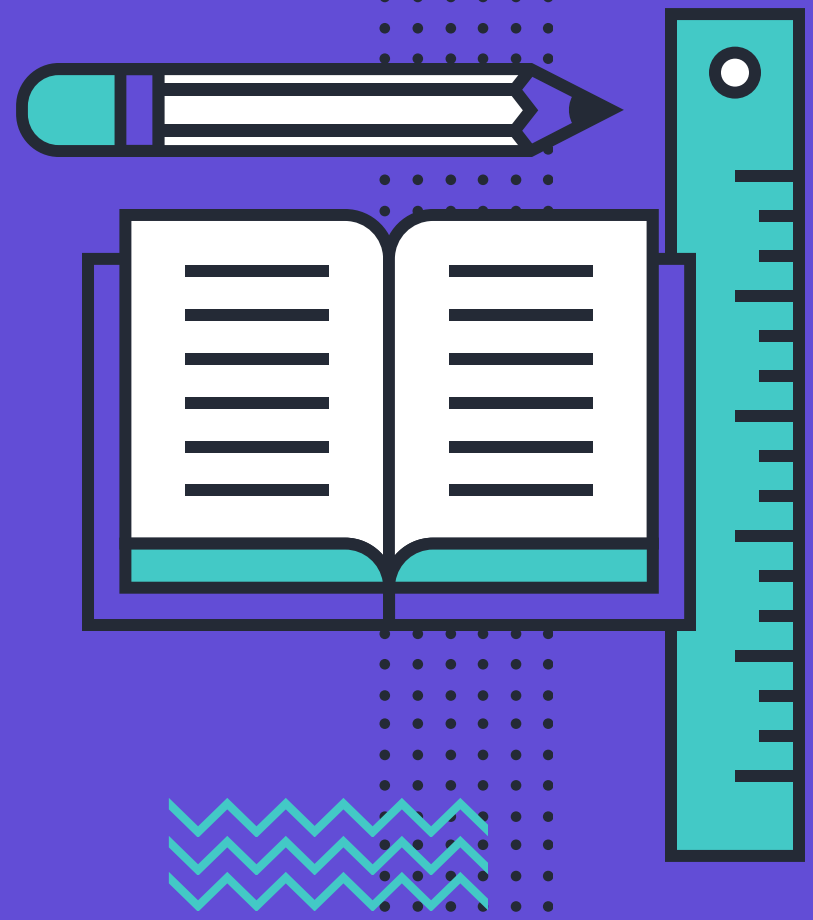
IMPORTANTE

A NUMERAÇÃO DO ÍNDICE DO VETOR COMEÇA SEMPRE NO ZERO E TERMINA SEMPRE EM N-1, EM QUE N É O NÚMERO DE ELEMENTOS DEFINIDOS INICIALMENTE

O NOME DO VETOR INDICA ONDE QUE OS DADOS COMEÇAM E O ÍNDICE QUANTAS POSIÇÕES ELE DEVE PULAR PRA ACESSAR DETERMINADO NÚMERO NA MEMÓRIA

NÃO É POSSÍVEL FAZER ATRIBUIÇÃO DE VETORES INTEIROS, APENAS DE CADA POSIÇÃO INDIVIDUALMENTE

MATRIZES



O QUE É UMA MATRIZ?

É UM ARRAY DE DUAS DIMENSÕES. ELE POSSUI LINHAS E COLUNAS (COMO UMA TABELA).

DECLARANDO UMA MATRIZ

FORMA GERAL:

```
tipo_dado nome_array [n_linhas][n_colunas];
```

PARA CRIAR UMA MATRIZ DE 100 LINHAS E 50 COLUNAS POR EXEMPLO, USA-SE A DECLARAÇÃO A SEGUIR

```
float mat[100][50];
```

INICIALIZANDO UMA MATRIZ NA SUA DECLARAÇÃO

FORMA GERAL:

```
tipo_dado nome_array [n_linhas] [n_colunas] = {dados};
```

EXEMPLO:

```
int matriz_1 [3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
int matriz_2 [3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
```

INICIALIZANDO UMA MATRIZ SEM TAMANHO

FORMA GERAL:

```
tipo_dado nome_array [] [n_colunas] = {dados};
```

NO CASO DA INICIALIZAÇÃO DE MATRIZES, APENAS A PRIMEIRA DIMENSÃO PODE FICAR SEM TAMANHO DEFINIDO

EXEMPLO:

```
int mat[][3] = {15, 12, 9, 1, 35, 14};
```

ACESSANDO UM CONTEÚDO DA MATRIZ

COMO UMA MATRIZ POSSUI DUAS DIMENSÕES, PRECISAMOS UTILIZAR DOIS ÍNDICES: O PRIMEIRO PARA A LINHA E O SEGUNDO PARA A COLUNA

EXEMPLO:

```
int mat[100][50];
mat[0][1] = 99;
```

```c
#include<stdio.h>

int main ( ) {
    int i, j;
    int matriz[10][20];

    for(i=0; i<10; i++){
        for (j=0; j<20; j++){
            matriz[i][j] = 0;
        }
     }

    return 0;
}
```

```c
#include<stdio.h>

int main ( ) {
    int i, j;
    int matriz[10][20];

    for(i=0; i<10; i++){
        for (j=0; j<20; j++){
            matriz[i][j] = 0;
        }
     }

    return 0;
}
```

```c
#include<stdio.h>

int main ( ) {
    int i, j;
    int matriz[10][20];

    for(i=0; i<10; i++){
        for (j=0; j<20; j++){
            matriz[i][j] = 0;
        }
    }

    for(i=0; i<10; i++){
        for (j=0; j<20; j++){
            printf("%d ", matriz[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```