



DEFINIÇÃO

Apresentação de duas formas de estruturação de dados homogêneos (do mesmo tipo), em memória, durante a execução de um programa de computador: vetores e matrizes.

PROPÓSITO

Compreender o armazenamento de dados em estruturas homogêneas (vetores e matrizes) para a tomada decisões sobre melhor a estratégia de armazenamento de dados durante o processamento das instruções do programa.

PREPARAÇÃO

Antes de prosseguir, instale em seu computador ou *smartphone* os seguintes programas obtidos gratuitamente na internet: Portugol Studio e DEV C++.

Assim, ao longo deste estudo, você aplicará os seguintes elementos:

Variáveis, tipos de dados e constantes;

Expressões aritméticas, lógicas e relacionais;

Comandos de atribuição, entrada e saída de dados;

Comandos de decisão (ou seleção) simples, composto e aninhado;

Comandos de repetição (com variável de controle, teste no início e no final da repetição).

OBJETIVOS

MÓDULO 1

Empregar o vetor para armazenamento de dados em um programa

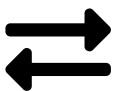
MÓDULO 2

Empregar a matriz para armazenamento de dados em um programa

INTRODUÇÃO

Muitas soluções algorítmicas requerem o armazenamento temporário de dados em estruturas, a fim de manipulá-los (processá-los) antes de serem persistidos em arquivos ou bancos de dados.

Até o momento, os programas definidos usaram **variáveis** de tipos de dados simples, nas quais apenas um dado pode ser armazenado em cada variável por vez.



Entretanto, existem problemas e situações que requerem o armazenamento de mais de um dado por vez, sem necessariamente ter de definir e usar N variáveis, o que tornaria complexa a manipulação desses dados.

VARIÁVEIS

Posições de memória acessadas pelo seu nome.

Vamos ver como as estruturas de dados homogêneas, como vetores e matrizes, permitem que os dados sejam armazenados em única variável (nome) e acessados individualmente.

MÓDULO 1

ARMAZENAMENTO DE DADOS

Durante a execução de um programa, armazenamos os dados necessários ao processamento em variáveis que residem, temporariamente, na memória do computador.

Geralmente, as linguagens de programação definem:

TIPOS DE DADOS SIMPLES

Apenas um dado pode ser armazenado por vez.

TIPOS DE DADOS ESTRUTURADOS OU COMPOSTOS, OU ESTRUTURAS DE DADOS

Capazes de armazenar mais de um valor na mesma variável, usando mais de uma posição de memória para armazenar os dados da estrutura.

Disponível na grande maioria das linguagens de programação, incluindo a C, o vetor é uma das estruturas de dados mais recorrentes, classificadas como homogêneas, uma vez que armazenam dados do mesmo tipo.

EXEMPLO 1

Faça um programa que leia 100 números inteiros e positivos, e mostre o maior deles.

Analise o trecho do código, onde as linhas estão numeradas para efeitos de explicação:

```
int main()
{
int num,maior,ind;
maior = 0;
for (ind= 1; ind <=100; ind=ind+1)
{
scanf ("%d",&num);
if (num > maior)
maior=num;
}
printf ("%d /n O maior dos números lidos e: ",maior);
return 0;
}
```

PELO CÓDIGO ANALISADO, O QUE PODEMOS CONCLUIR?

1

O uso de variáveis simples resolve esse problema. Podemos usar uma variável do tipo inteiro para armazenar cada número a ser lido e processado, bem como o maior dos números lidos a cada momento (linha 3).

2

A cada valor lido na variável simples e inteira **num** (linha 7), o conteúdo anterior é descartado, ficando armazenado o último valor lido. A cada laço da iteração (repetição), comparamos o último valor lido com o conteúdo da variável **maior**, simples e inteira, devidamente inicializada com 0 (zero) na linha 4. Caso o valor lido seja superior ao contido na variável **maior**, atribui-se o número lido na variável **num** à variável **maior** (linhas 8 e 9).

Quando os 100 números forem lidos e processados, e, portanto, houver o término da repetição, teremos o maior dos valores lidos armazenado na variável **maior**. Bastará exibi-lo no dispositivo de saída (linha 11) para concluir a solução do problema apresentado.

EXEMPLO 2

Existem problemas que não podem ser resolvidos usando apenas variáveis do tipo simples. É o que vamos ver neste exemplo.

Faça um programa que leia 100 números inteiros e mostre-os na ordem inversa em que foram lidos.

PROBLEMA

Ao lermos o segundo número na mesma variável, o primeiro número lido será perdido. Assim, não teremos como exibi-los posteriormente.

Não há como usar 100 variáveis simples do tipo inteiro, como no exemplo 1, pois é necessário exibir os dados na ordem inversa daquela em que foram lidos.

Temos de armazenar todos os dados para, depois, mostrá-los na ordem desejada, conforme esta sequência, com apenas 10 números:

Leitura → 10 56 78 90 12 91 23 42 90 58

Exibição → 58 90 42 23 91 12 90 78 56 10

Repare que o primeiro número digitado (lido pelo programa) será o último a ser exibido no dispositivo de saída. Para o caso de 10 números, até poderíamos cogitar a ideia de usar 10 variáveis do tipo simples (inteiro), mas, ainda assim, seria oneroso escrever o código.

Como são 100 números, fica inviável usarmos 100 variáveis do tipo simples, pois a manipulação de 100 variáveis seria exaustiva. Imagine, então, se o enunciado pedisse a leitura de 1.000 números?

TIPOS DE DADOS ESTRUTURADOS OU COMPOSTOS

Dados do tipo simples (int, float, double char etc.) são armazenados em uma variável que ocupa uma posição de memória.

Aqui, nós nos valemos da Lei da Física:

“Dois corpos não podem ocupar o mesmo lugar no espaço.”

Assim concluímos que existem diferenças entre tipos de dados simples e estruturados. Observe:

Tipos de dados simples

Podem armazenar apenas um valor por vez. Ao ser lido ou atribuído um novo valor a uma variável simples, seu conteúdo anterior é substituído por ele.

Tipos de dados estruturados ou compostos

Podem armazenar um conjunto de dados, simultaneamente, em memória, e, claro, usam mais de uma posição de memória.

A forma como os dados são armazenados varia de estrutura para estrutura e depende de como a linguagem de programação os implementa.

Em geral, as linguagens de programação classificam os **dados estruturados** da seguinte forma:

Critérios	Classificação	Descrição	Exemplos
Forma de alocação	Estáticos	Dados declarados e criados no início do programa, e mantidos assim até o final de sua execução.	<ul style="list-style-type: none">• Vetor• Registro• Matriz
	Dinâmicos	Dados criados durante a execução do programa.	Ponteiros
Tipos de dados	Homogêneos	Todos os dados são do mesmo tipo.	<ul style="list-style-type: none">• Vetores• Matrizes
	Heterogêneos	Os dados da estrutura podem ser de tipos diferentes.	Registros

Atenção! Para visualização completa da tabela utilize a rolagem horizontal

CONCEITO DE VETOR

O vetor é um tipo de dado estruturado ou composto, estático e homogêneo, disponível na maioria das linguagens de programação, incluindo a C.

Como estrutura de dados, o vetor permite que mais de um dado do mesmo tipo seja armazenado.

Mas como o vetor organiza os dados que armazena?

O vetor usa posições consecutivas de memória e simula esse funcionamento a partir de índices. Por isso, é uma estrutura **indexada**.

Os números **em verde** representam os índices do vetor, usados para acessar cada um de seus elementos. A posição 0 é a primeira, e a posição 9 é a última.

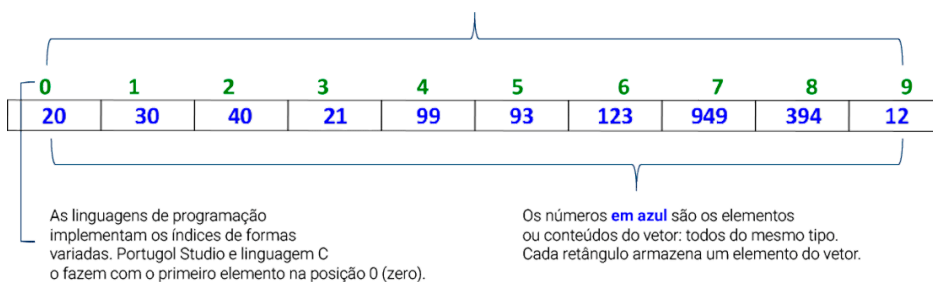


Imagem: Elaborado por Marcelo Vasques de Oliveira

📷 Representação de um vetor.

DECLARAÇÃO DO VETOR

Como qualquer variável, o vetor precisa ser declarado na maioria das linguagens de programação que assim o exigem, como no caso da C. Você deve estar se perguntando:

De forma geral, como os vetores são declarados em Portugol Studio e na linguagem C?

Essa declaração é feita da seguinte maneira:

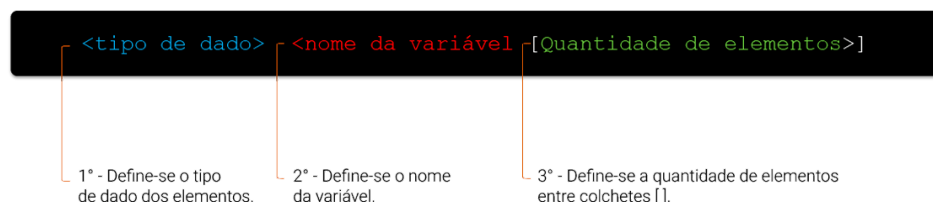


Imagem: Elaborado por Marcelo Vasques de Oliveira

Vamos ilustrar a forma de declarar, em Portugol Studio, um vetor de 5 elementos do tipo **caractere**:

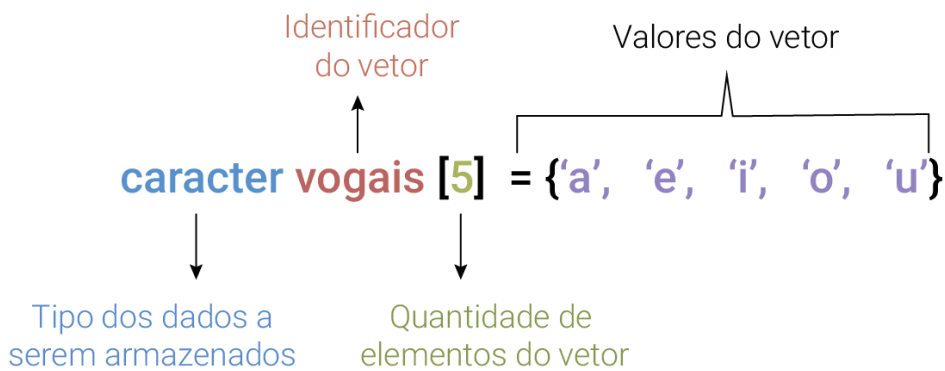


Imagem: Elaborado por Marcelo Vasques de Oliveira

📷 Declaração e inicialização do vetor VOGAIS - Portugol Studio

Como fica declarado o vetor representado na figura 1, em Portugol Studio e na linguagem C?

// Portugol Studio

```
inteiro numeros[10]
```

// Linguagem C

```
int numeros[10];
```

Vamos ver outros exemplos de declaração de vetores:

PORTUGOL STUDIO

inteiro numeros[100] → vetor de 100 elementos do tipo inteiro

real notas [20] → vetor de 20 elementos do tipo real

caractere vogais [5] → vetor de 5 elementos do tipo caractere

LINGUAGEM C

```
int numeros[100];
```

```
float notas [20];
```

```
char vogais [5];
```

Assim como as variáveis de tipo simples, as do tipo vetor também podem ser inicializadas durante sua declaração, conforme observamos a seguir:

// Portugol Studio

```
inteiro numeros[5] = {10,12,14,16,18}
```

// Linguagem C

```
int numeros[5] = {10,12,14,16,18};
```



📢 **ATENÇÃO**

Existem linguagens de programação que possibilitam a definição do índice inicial e final, mas, em Portugol Studio e na linguagem C, definimos apenas a quantidade de elementos do vetor. O primeiro elemento ocupa a posição 0 (zero), e o último, a posição (N-1), na qual N é a quantidade de elementos definida para o vetor. A tentativa de acessar um elemento que esteja em uma posição inferior a 0 (zero) e superior a N-1 resultará em erro.

ACESSO AOS ELEMENTOS DO VETOR

Os elementos de um vetor são acessados, atribuídos, lidos ou exibidos, elemento por elemento.

EXEMPLO

Considere um vetor de 10 elementos do tipo inteiro, com os seguintes comandos, numerados de 1 a 9, tanto em Portugol Studio quanto em linguagem C:

// Portugol Studio

inteiro vet[10]

1 vet [0] = 5

2 vet [1] = 6

3 vet [3] = 8

4 vet [4] = 12

5 vet [5] = 18

6 vet [6] = 22

7 vet [7] = 34

8 vet [8] = 78

9 vet [9] = 45

// Linguagem C

int vet[10];

1 vet [0] = 5;

2 vet [1] = 6;

3 vet [3] = 8;

4 vet [4] = 12;

5 vet [5] = 18;

6 vet [6] = 22;

7 vet [7] = 34;

8 vet [8] = 78;

9 vet [9] = 45;

Ao final da execução de cada comando associado às linhas numeradas de 1 a 9, teremos o vetor preenchido:

0	1	2	3	4	5	6	7	8	9
5	6	??	8	12	18	22	34	78	45

Imagem: Elaborado por Marcelo Vasques de Oliveira

📷 Vetor int vet[10]

Por que isso ocorreu?

1

Não houve inicialização dos elementos do vetor.

2

Não foi atribuído nenhum valor à posição 2 do vetor, que fica com valor indefinido.

Considerando o vetor apresentado, vamos ver o que vai acontecer com os comandos equivalentes em Portugol Studio e na linguagem C.

1. COMANDO PARA EXIBIR UM ELEMENTO DO VETOR

```
// Portugol Studio
```

```
escreva ("Posicao 1 do vetor: ",vet[1])
```

```
// Linguagem C
```

```
printf("%d Posicao 1 do vetor: ",vet[1]);
```

O comando anterior exibe no dispositivo de saída o elemento que ocupa a posição (índice) 1 do vetor. Avaliando a figura 3, o elemento da posição 1 é o 6.

2. COMANDO PARA ATRIBUIR VALOR A UM ELEMENTO DO VETOR

```
// Portugol Studio
```

```
vet[4]=90
```

```
// Linguagem C
```

```
vet[4]=90;
```

O comando anterior armazena o valor 90 no elemento de índice 4 do vetor. Esse elemento, anterior ao comando, é 12 e é substituído pelo 90.

Como ficará o vetor?

0	1	2	3	4	5	6	7	8	9
5	6	??	8	90	18	22	34	78	45

Imagem: Elaborado por Marcelo Vasques de Oliveira

📷 Vetor int vet[10]

// Portugol Studio

vet[2]=56

// Linguagem C

vet[2]=56;

O comando anterior armazena o valor 56 no elemento de índice 2 do vetor. Esse elemento, anterior ao comando, é desconhecido e é substituído pelo 56.

Como ficará o vetor?

0	1	2	3	4	5	6	7	8	9
5	6	56	8	90	18	22	99	78	45

Imagem: Elaborado por Marcelo Vasques de Oliveira

📷 Vetor int vet[10]

3. COMANDO PARA LER UM DADO DO DISPOSITIVO DE ENTRADA E ARMAZENAR EM POSIÇÃO DO VETOR

a) Lendo o dado de entrada direto para uma posição do vetor

// Portugol Studio

leia (vet[7])

// Linguagem C

scanf ("%d",&vet[7]);

O comando anterior armazena o valor lido pelo dispositivo de entrada na posição (índice 7) do vetor. O elemento de índice 7, anterior ao comando, é 34, que é substituído pelo valor lido.

Se o usuário digitar o número 99, como ficará o vetor?

0	1	2	3	4	5	6	7	8	9
5	6	56	8	90	18	22	99	78	45

Imagem: Elaborado por Marcelo Vasques de Oliveira

📷 Vetor int vet[10]

b) Lendo o dado do dispositivo de entrada em uma variável e, depois, atribuindo o conteúdo dessa variável a uma posição do vetor

```
// Portugol Studio
```

```
leia (num)
```

```
vet[9]=num
```

```
// Linguagem C
```

```
scanf ("%d",&num);
```

```
vet[9]=num;
```

O comando anterior armazena, em uma variável, o valor lido pela digitação do usuário no dispositivo de entrada. Na sequência, armazena o conteúdo dessa variável na posição (índice 9) do vetor. O elemento de índice 9, anterior ao comando, é 45, que é substituído pelo valor lido.

Se o usuário digitar o número 122, como ficará o vetor?

0	1	2	3	4	5	6	7	8	9
5	6	56	8	90	18	22	99	78	122

Imagem: Elaborado por Marcelo Vasques de Oliveira

📷 Vetor int vet[10]

Os comandos a seguir resultarão em erro, pois acessam posições (índices do vetor) que não são válidas:

```
// Portugol Studio
```

```
escreva (vet[10])
```

```
vet[10]=901
```

```
leia (vet[10])
```

```
// Linguagem C
```

```
printf("%d Posicao 10 do vetor: ",vet[10]);
```

```
vet[10]=901;
```

```
scanf ("%d",&vet[10]);
```

Embora existam 10 elementos no vetor, nas linguagens Portugol Studio e C, o primeiro elemento ocupa a posição (índice) 0 (zero), e o último elemento ocupa a posição N-1, onde N é a quantidade de elementos do vetor. Assim, ao tentar acessar a posição 10 do vetor, haverá um erro na execução do programa, pois o último elemento é o da posição 9.

CADEIA DE CARACTERES OU *STRING*

Muitas linguagens possuem uma cadeia de caracteres

(*string*), mas a linguagem C não dispõe de um tipo de dado específico para armazenar essa cadeia, como o nome de uma pessoa.

Por isso, se quisermos armazenar nomes de pessoas, de lugares, enfim, qualquer cadeia de caracteres, teremos de

usar um vetor de caracteres.

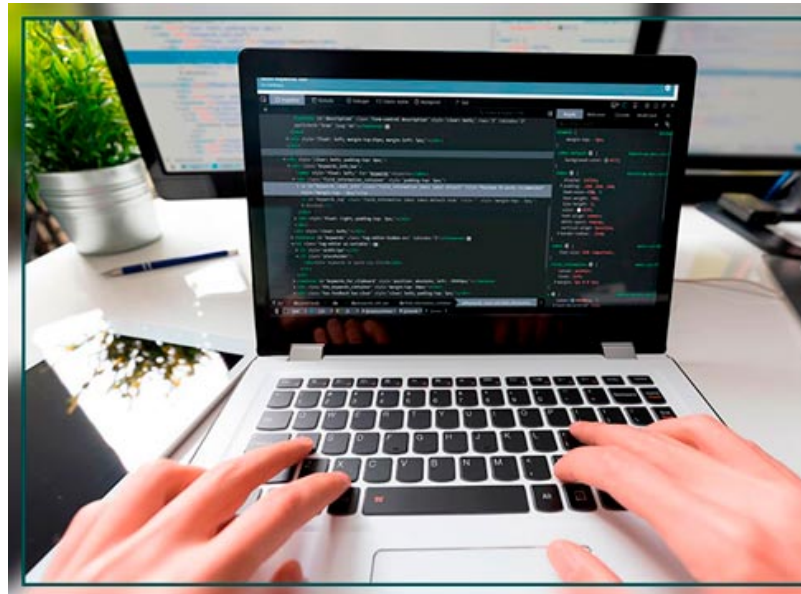


Foto: Shutterstock.com

Uma cadeia de caracteres pode ser definida como um vetor contendo caracteres ou símbolos.

Na linguagem C, uma cadeia de caracteres pode ser tratada como um vetor de elementos do tipo `char` e com o marcador “`\0`”. Se quisermos armazenar uma cadeia com 10 caracteres, deveremos declará-la com 11 elementos, pois o 11º será o caractere de controle, que é “`\0`”.

Vamos ver um exemplo de representação de um vetor de caracteres, armazenando a cadeia “**INTRODUCAO!**”:

0	1	2	3	4	5	6	7	8	9	10
I	N	T	R	O	D	U	C	A	O	\0

Imagem: Elaborado por Marcelo Vasques de Oliveira

Como é feita a declaração da cadeia de caracteres?

`char <Nome> [Tamanho+1]`



★ EXEMPLO

Declaração para o vetor representado (com conteúdo = “INTRODUCAO!”):

```
char nome [11]; //
```

INICIALIZAÇÃO

Para inicializar uma cadeia de caracteres, basta:

1

Atribuir cada dado a uma posição, como vimos no vetor.

2

Usar funções para manipulação de caracteres, contidas na biblioteca padrão *string.h*.

Vamos mostrar aqui, com algumas variações, apenas a primeira opção com o que já sabemos de vetores:

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

```
int main()
```

```
{
```

```
char cadeia[10];
```

```
cadeia[0]= 'a';
```

```
cadeia[1]= 'l';
```

```
cadeia[2]= 'g';
```

```
cadeia[3]= 'o';
```

```
cadeia[4]= 'r';
```

```
cadeia[5]= 'i';
```

```
cadeia[6]= 't';
```

```
cadeia[7]= 'm';
```

```
cadeia[8]= 'o';
```

```
cadeia[9]= 's';
```

```
}
```

OU

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

```
int main()
```

```
{
```

```
char cadeia[10] = {'a', 'l', 'g', 'o', 'r', 'i', 't', 'm', 'o', 's'};
```

```
}
```

LEITURA

Podemos ler uma cadeia, caractere por caractere, como fazemos com qualquer vetor, lendo elemento por elemento.

Contudo, é mais simples lermos a cadeia inteira, usando a formatação “%s”:

```
scanf("%s",cadeia)
```

Note que não usamos **&**.

O comando anterior seria equivalente ao trecho de código:

```
for (ind=0;ind<5;ind++)  
{  
scanf ("%c",&nome[ind]);  
getch();  
}
```

Para leitura de dados pelo dispositivo de entrada com o scanf, devemos nos atentar a um detalhe:

scanf

Ele não lê o caractere de controle (“\0”), mas, se o scanf estiver dentro de uma repetição, haverá tentativa de leitura desse caractere.

Logo, em comandos de leitura de variáveis do tipo char, coloque a função getchar() para resolver esse problema, conforme o exemplo anterior.

EXIBIÇÃO

Podemos usar **%s** para formatação de *strings* e a exibirmos direto, conforme fizemos com scanf. Veja o exemplo:

```
printf ("%s",cadeia)
```

O comando anterior seria equivalente ao trecho de código:

```
for (ind=0;ind<5;ind++)  
{  
printf ("%c",nome[ind]);  
}
```

DEMONSTRAÇÃO DO VETOR

I. O comando declara um vetor para armazenar notas de 10 alunos de uma turma:

```
// Portugol Studio  
real vet[10]
```

```
// Linguagem C  
double vet[10];
```

II. O comando declara um vetor para armazenar o sexo – masculino (M) ou feminino (F) – de 50 alunos de uma turma ou, ainda, para representar o nome de uma pessoa com até 49 caracteres:

```
// Portugol Studio
caracter vet[50]
```

```
// Linguagem C
char vet[50];
```

III. O comando declara um vetor para armazenar notas de 10 alunos de uma turma e inicializa cada nota com 0 (zero):

```
// Portugol Studio
real vet[10] = {0,0,0,0,0,0,0,0,0,0}
```

```
// Linguagem C
double vet[10] = {0,0,0,0,0,0,0,0,0,0};
```

IV. Se fossem 100 alunos em vez de 10, seria inviável inicializar o vetor com 100 zeros entre as chaves. Se fossem 500 alunos, mais inviável ainda.

Nesses casos, como inicializar cada elemento do vetor com 0 (zero)?

RESPOSTA

Podemos usar um comando de repetição para percorrer cada elemento do vetor e atribuir o valor 0 (zero) a cada um.

Nesse caso, o comando de repetição mais adequado é o PARA (FOR), pois a repetição é para número fixo e conhecido de vezes, no caso 100.

Vamos iniciar com a variável de controle, de nome **posicao**, começando do 0 (primeiro elemento do vetor) até 99 e, para cada posição, fazer com que **vet[posição]=0**.

Portugol Studio	Linguagem C
Para (posicao=0;posicao<100;posicao++) { vet[posicao]=0 }	for (posicao=0;posicao<100;posicao++) { vet[posicao]=0; }

Atenção! Para visualização completa da tabela utilize a rolagem horizontal

V. O comando declara um vetor para armazenar a quantidade de vezes que cada vogal aparece em um texto e inicializar com 0 (zero) cada quantidade. São 5 vogais. Então, precisamos de 5 posições no vetor. O algoritmo deve simular o mapeamento da seguinte forma:

1

Posição 0 = quantidade de vezes que a letra A apareceu.

2

Posição 1 = quantidade de vezes que a letra E apareceu.

3

Posição 2 = quantidade de vezes que a letra I apareceu.

4

Posição 3 = quantidade de vezes que a letra O apareceu.

5

Posição 4 = quantidade de vezes que a letra U apareceu.

// Portugol Studio

```
int vet[5]
```

// Linguagem C

```
int vet[5];
```

VI. O trecho de código declara e lê as notas de 80 alunos de uma turma:

// Portugol Studio

```
Para (posicao=0;posicao<80;posicao++)
```

```
{
```

```
leia (vet[posicao])
```

```
}
```

// Linguagem C

```
for (posicao=0;posicao<80;posicao++)
```

```
{
```

```
scanf ("%d",&vet[posicao]);
```

```
}
```

VII. O trecho de código exibe no dispositivo de saída as notas dos 80 alunos da turma:

// Portugol Studio

```
Para (posicao=0;posicao<80;posicao++)
```

```
{
```



```
escreva (vet[posicao])
```

```
}
```

```
// Linguagem C
```

```
for (posicao=0;posicao<80;posicao++)
```

```
{
```

```
printf ("%d",&vet[posicao]);
```

```
}
```

VIII. O trecho de código calcula e mostra a média da turma (soma das notas dos alunos dividida por 80, que é a quantidade de alunos):

```
// Portugol Studio
```

```
soma=0
```

```
para (posicao=0;posicao<80;posicao++)
```

```
{
```

```
soma=soma+vet[posicao]
```

```
}
```

```
escreva ("A media e : ",media/80)
```

```
// Linguagem C
```

```
soma=0;
```

```
for (posicao=0;posicao<80;posicao++)
```

```
{
```

```
soma=soma+vet[posicao];
```

```
}
```

```
printf("media = %.2f ",(soma/80));
```

IX. O trecho de código lê uma cadeia de 8 caracteres e mostra o texto invertido. Por exemplo, ao ler “programa”, o algoritmo deve mostrar “amargorp”:

```
// Portugol Studio
```

```
caracter nome[8]
```

```
inteiro ind
```

```
leia ("%s", nome)
```

```
para (ind=8;ind>=0;ind--)
```

```
escreva (nome[ind])
```

```
// Linguagem C
```

```
char nome[8];
```

```
int ind;
```

```
scanf ("%s",nome);
```

```
for (ind=8;ind>=0;ind--)
```

```
{
```

```
printf ("%c ",nome[ind]);
```

```
}
```

Observe que o programa lê a cadeia de uma vez, usando **%s**, e exibe caractere por caractere para mostrar do final ao início, invertendo a cadeia original.

MÃO NA MASSA

Vamos ver a teoria na prática.

Sugerimos que realize o seguinte procedimento:

A - Desenvolva, você mesmo, o algoritmo em Portugol Studio e, depois, na linguagem C.

B - Garanta que sua solução funcione com dados distintos, executando algumas vezes.

C - Veja a solução que sugerimos e compare com a sua.

Vamos iniciar com uma atividade que motivou o estudo das estruturas homogêneas de dados.

Aqui, alternativamente, vamos executar os programas na linguagem C, na ferramenta Online C++ Compiler – online editor, para que ganhe experiência em novo ambiente, além do DEVC++. Essa ferramenta, que você pode buscar na internet, tem a vantagem de não precisarmos instalar algo no computador. Após compilar o programa, é possível executá-lo normalmente.

ATIVIDADE 1

Faça um programa que leia 100 números inteiros e mostre-os na ordem inversa em que foram lidos.

Estrutura de dados: O vetor vai armazenar 100 números inteiros.

RESOLUÇÃO

I. Inicialmente, devem ser lidos cada um dos 100 números do dispositivo de entrada e armazenados em uma posição do vetor.

II. Depois, percorre-se o vetor de trás para frente, ou seja, iniciando do último elemento (posição 99) até o primeiro (posição 0) e exibindo cada elemento.

Comando de repetição: PARA (FOR), pois a repetição tem número conhecido e fixo de vezes =100.

Observe que:



Ao ler os dados, usamos o PARA (FOR), começando da posição 0 (zero) e incrementando de 1 em 1, enquanto for menor que 100 (99), quando acaba a repetição.



Para exibir os dados em ordem inversa, percorremos o vetor da posição 99 (último elemento) até a posição 0, decrementando de 1 em 1, e mostramos cada elemento do vetor.



Preenchemos o vetor da posição 0 a 99 e, na sequência, mostramos seus elementos da posição 99 até a posição 0 (zero). Assim, mostramos os elementos na ordem inversa daquela que lemos.

Este é o trecho de código da solução tanto em Portugol Studio quanto em linguagem C:

```
// Portugol Studio
inteiro vet[100], posicao
para (posicao=0;posicao<100;posicao++)
{
    leia (vet[posicao])
}
para (posicao=99;posicao>=0;posicao--)
{
```

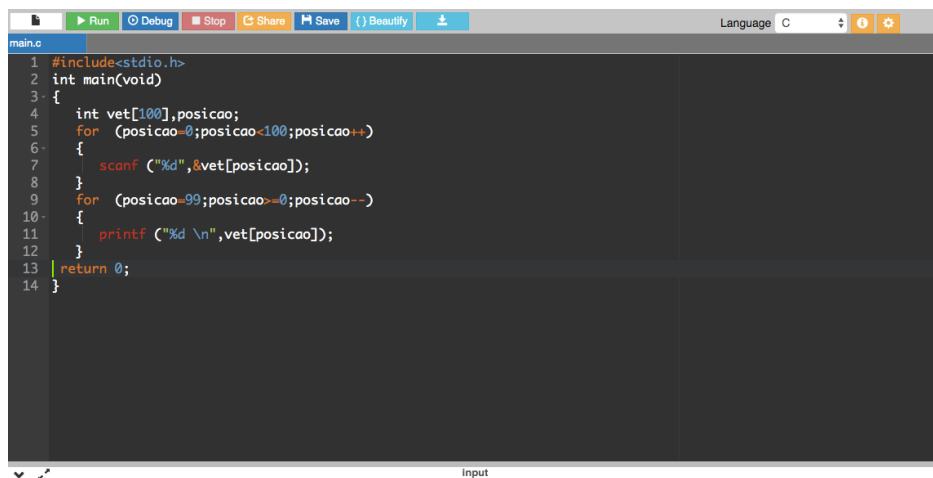
```

escreva (vet[posicao])
}

// Linguagem C
int vet[100],posicao;
for (posicao=0;posicao<100;posicao++)
{
scanf ("%d",&vet[posicao]);
}
for (posicao=99;posicao>=0;posicao--)
{
printf ("%d \n",vet[posicao]);
}

```

Janela da ferramenta online com o código completo em linguagem C:



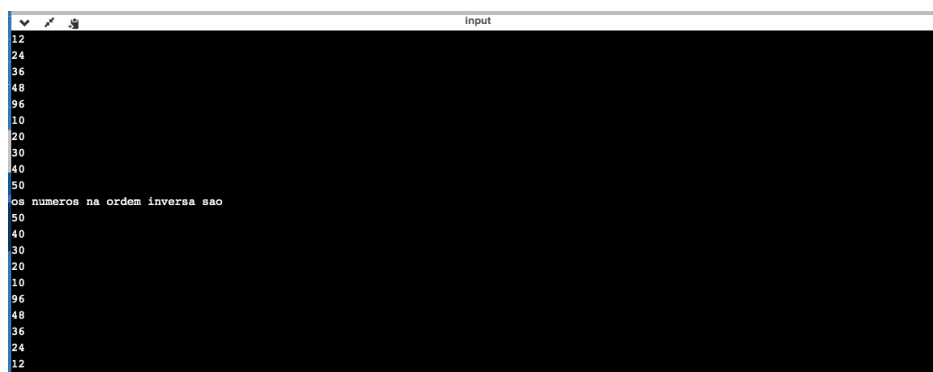
```

1 #include<stdio.h>
2 int main(void)
3 {
4     int vet[100],posicao;
5     for (posicao=0;posicao<100;posicao++)
6     {
7         scanf ("%d",&vet[posicao]);
8     }
9     for (posicao=99;posicao>=0;posicao--)
10    {
11        printf ("%d \n",vet[posicao]);
12    }
13    return 0;
14 }

```

Imagem: Elaborado por Marcelo Vasques de Oliveira

Janela de execução do programa-fonte anterior na ferramenta online:



```

12
24
36
48
96
10
20
30
40
50
os numeros na ordem inversa sao
50
40
30
20
10
96
48
36
24
12

```

Imagem: Elaborado por Marcelo Vasques de Oliveira

Observe que a execução considera apenas 10 números, em vez de 100.

ATIVIDADE 2

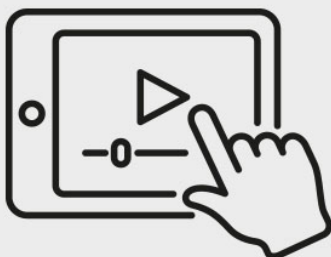
Faça um programa que leia a nota de 20 alunos da turma e mostre as que são iguais ou superiores à média da turma.

Estrutura de dados: O vetor vai armazenar 20 números reais.



Neste vídeo, o professor **Marcelo Vasques** resolverá a atividade 2.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Solução em C

Abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

```
#include <stdio.h>

int main()
{
    float vet[20],soma=0,media; int posicao;
    for (posicao=0;posicao<20;posicao++)
    {
        scanf ("%f",&vet[posicao]);
        soma=soma+vet[posicao];
    }
    media=soma/20;
    printf ("numeros maiores quemedia %.2f \n",media);
    for (posicao=0;posicao<20;posicao++)
    {
        if (vet[posicao] >= media)
            printf ("%f \n",vet[posicao]);
    }
}
```

Atenção! Para visualização completa do código utilize a rolagem horizontal

ATIVIDADE 3

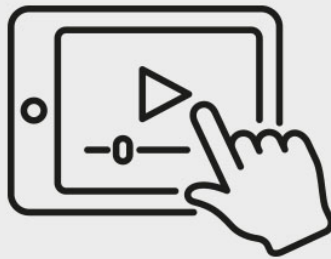
Leia uma sequência de letras, terminada na letra ("z"), e mostre quantas vezes cada vogal (a, e, i, o, u) apareceu.

Estrutura de dados: O vetor vai armazenar 5 números inteiros. Cada posição do vetor vai acumular a quantidade de vezes que cada vogal (A, E, I, O, U) apareceu. O índice 0 (zero) corresponde ao total de vogais "A", o índice 1 corresponde à vogal "E", e assim sucessivamente, até o índice 4, que vai armazenar a vogal "U".



Neste vídeo, o professor **Marcelo Vasques** resolverá a atividade 3.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Solução em C

Abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

```
#include <stdio.h>

int main()
{
    int vet [5] = {0,0,0,0,0},posicao;

    char letra;

    scanf ("%c",&letra);

    while (letra != 'z')
    {
        switch (letra)
        {
            case 'a':
                vet[0]++;
                break;

            case 'e':
                vet[1]++;
```

```
break;

case 'i':

vet[2]++;

break;

case 'o':

vet[3]++;

break;

case 'u':

vet[4]++;

break;

}

scanf("%c",&letra);

}

printf ("Quantidade de cada vogal, em ordem \n");

for (posicao=0;posicao<5;posicao++)

{

printf("%d ",vet[posicao]);

}

}
```

Atenção! Para visualização completa do código utilize a rolagem horizontal

ATIVIDADE 4

Faça um algoritmo que leia 50 números inteiros e armazene-os em um vetor. Copie para um segundo vetor de 50 números inteiros cada elemento do primeiro, observando as seguintes regras:

Se o número for par, preencha a mesma posição do segundo vetor com o número sucessor do contido na mesma posição do primeiro vetor.

Se o número for ímpar, preencha a mesma posição do segundo vetor com o número antecessor do contido na mesma posição do primeiro vetor.

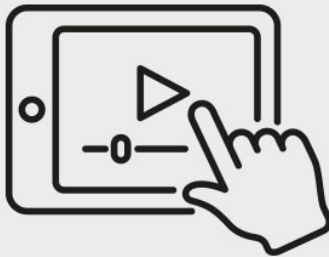
Ao final, mostre o conteúdo dos dois vetores simultaneamente.

Estrutura de dados: Dois vetores de 50 posições de números inteiros.



Neste vídeo, o professor **Marcelo Vasques** resolverá a atividade 4.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Solução em C

Abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

```
#include <stdio.h>

int main()
{
    const int tamvet=50;

    int vet1[tamvet],vet2[tamvet],posicao;

    for (posicao=0;posicao<tamvet;posicao++)
    {
        scanf ("%d",&vet1[posicao]);

        if (vet1[posicao]%2 == 0)

            vet2[posicao]=vet1[posicao]+1;

        else

            vet2[posicao]=vet1[posicao]-1;

    }

    printf ("Elementos de VET 1 e VET2: ");

    for (posicao=0;posicao<tamvet;posicao++)
    {

        printf ("%d ",vet1[posicao]);

        printf ("%d ",vet2[posicao]);

    }

}
```


ATIVIDADE 5

Faça um algoritmo que leia 50 números inteiros e armazene-os em um vetor.

Na sequência, leia uma lista de 10 números inteiros e verifique se cada um deles está contido em alguma posição do vetor. Em caso positivo, mostre a posição do vetor em que ele se encontra.

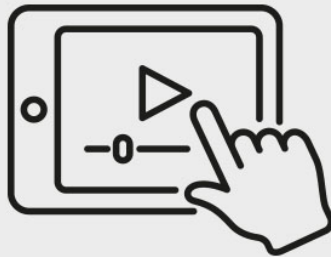
Estrutura de dados: Um vetor de 50 posições de números inteiros.

Comando de repetição: PARA (FOR), pois sabemos que leremos e processaremos 50 elementos. Logo, teremos uma solução com número fixo e conhecido de vezes.



Neste vídeo, o professor **Marcelo Vasques** resolverá a atividade 5.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Solução em C

Abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

```
#include <stdio.h>

int main()
{
    const int tamvet=50, tamlista=10;

    int vet[tamvet],posicao,posvet,achou,numero;

    printf ("\n Digite os dados do vetor \n");

    for (posicao=0;posicao<tamvet;posicao++)

        scanf ("%d",&vet[posicao]);

    printf ("\n Digite numeros a procurar: \n");

    for (posicao=1;posicao<=tamlista;posicao++)

    {
```

```
scanf ("%d",&numero) ;

// verifica se o numero está no vetor

posvet=0;

achou=0;

while (posvet<=tamvet-1 && achou==0)

{

if (numero==vet[posvet])

achou=1;

else posvet++;

}

if (achou==1)

printf ("acho na posicao: %d",posvet) ;

else

printf ("nao achou o numero");

}

}
```

Atenção! Para visualização completa do código utilize a rolagem horizontal

PRATIQUE +

Considere o seguinte problema em uma escola primária:

Em uma turma com 50 alunos, cada um faz 3 provas por semestre. Além de armazenar as 3 provas dos 50 alunos, existe a necessidade de mostrar:

A média de cada prova.

A média de cada aluno.

A média da turma.

Agora, vamos:

a) Identificar as estruturas de dados necessárias à solução do problema.

b) Escrever o trecho de código em linguagem C para calcular e mostrar as médias de cada prova, de cada aluno e da turma.



Foto: Shutterstock.com

RESOLUÇÃO

A grande questão é a organização da estrutura para armazenar os dados. Existem outras soluções: vamos usar 3 vetores de 50 posições de números reais.

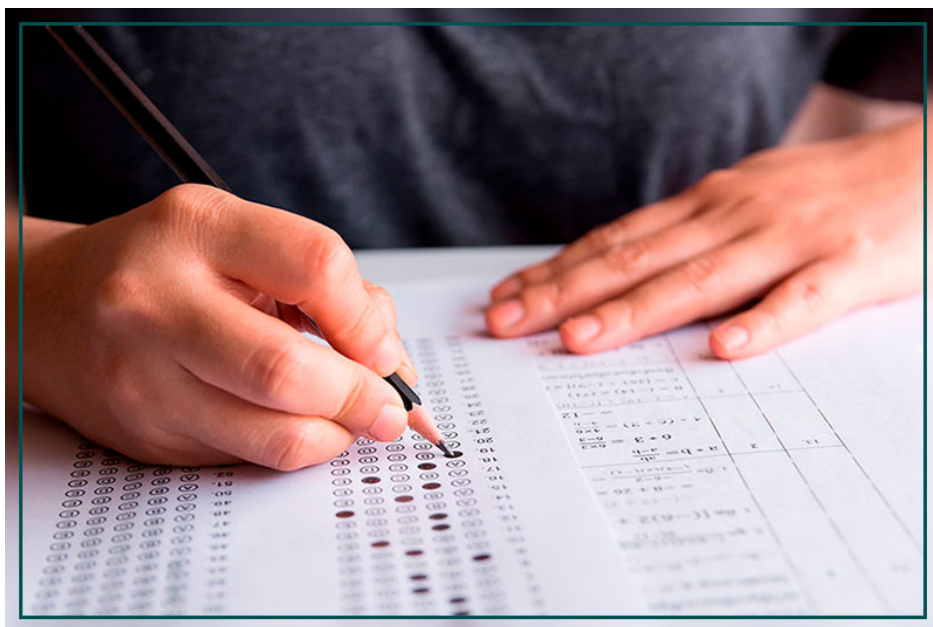


Foto: Shutterstock.com

Cada vetor armazena dados de 1 prova: prova1, prova2, prova3.

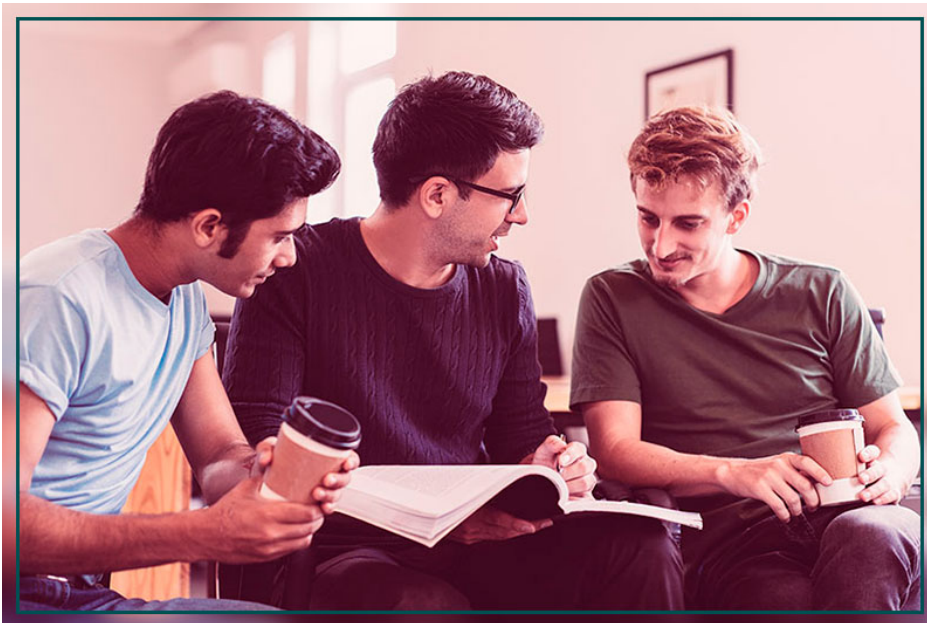


Foto: Shutterstock.com

Cada aluno está representado no mesmo índice dos 3 vetores.

A posição 0 do vetor prova1 é a nota da prova1 do aluno1.

A posição 0 do vetor prova2 é a nota da prova2 do aluno1.

A posição 0 do vetor prova3 é a nota da prova3 do aluno1.

A posição 1 de cada vetor representa as provas do aluno 2, e assim por diante.

A posição 49 de cada vetor armazena as provas do último aluno.

Observe o exemplo em desenho das 3 estruturas:

Vetor prova 1

0	1	2	3	47	48	49
7.00	8.50	9.35	8.45	9.00	7.90	9.60

Imagem: Elaborado por Marcelo Vasques de Oliveira

Vetor prova 2

0	1	2	3	47	48	49
9.70	9.20	8.95	9.00	10.00	6.40	4.30

Imagem: Elaborado por Marcelo Vasques de Oliveira

Vetor prova 3

0	1	2	3	47	48	49
10.00	8.50	7.50	8.80	9.20	8.00	9.00

Imagem: Elaborado por Marcelo Vasques de Oliveira

Veja que o trecho de código, em linguagem C, declara os vetores necessários:

```
float prova1[50], prova2[50], prova3[50];
```

MÉDIA DE CADA PROVA – USAMOS TODAS AS POSIÇÕES DE CADA VETOR

As notas da prova 1 são: 7.00, 8.50, 9.35, 8.45 ... 9.00, 7.90, 9.60.

As notas da prova 2 são: 9.70, 9.20, 8.95, 9.00 ... 10.00, 6.40, 4.30.

As notas da prova 3 são: 10.00, 8.50, 7.50, 8.80 ... 9.20, 8.00, 9.00.

Em outras palavras, para calcular a média de cada prova, basta acumular a nota de cada vetor prova e dividir a soma por 50.

O seguinte trecho de código em linguagem C calcula a média da prova 1:

```
Soma1=0;
for (pos=0;pos<49;pos++)
{
    soma1=soma1+prova1[pos];
}
mediaprova1=soma/50;
printf ("a media da prova1 e: %.02",mediaprova1);
```

Para calcular a média das provas 2 e 3, basta substituir as variáveis **soma1**, **mediaprova1** e o vetor **prova1** neste trecho de código.

MÉDIA DE CADA ALUNO – USA-SE A MESMA POSIÇÃO DOS 3 VETORES

As notas do aluno 1 são: 7.00 9.70 10.00 → média = 8.90.

As notas do aluno 2 são: 8.50 9.20 8.50 → média = 8.73; e assim por diante.

Em outras palavras, para calcular a média de cada aluno, basta somar as notas nas mesmas posições dos 3 vetores e dividir por 3.

O seguinte trecho de código, em linguagem C, calcula a média de cada aluno:

```
for (pos=0;pos<50;pos++)
{
    medialuno=(prova1[pos] + prova2[pos] + prova3[pos])/3;
}
printf ("a media do aluno e: %.02",mediaaluno);
```

MÉDIA DA TURMA – PODE SER CALCULADA DE 2 FORMAS

[1] – PELA MÉDIA DAS PROVAS

Anteriormente, quando calculamos a média de cada prova, chegamos à **mediaprova1**, **mediaprova2** e **mediaprova3**.

A média da turma pode ser obtida somando a média das 3 provas e dividindo por 3, conforme trecho do código:

```
mediaturma = (mediaprova1 + mediaprova2 + mediaprova3)/3;
```

[2] – PELA MÉDIA DOS ALUNOS

Usando o mesmo código para tirar a média de cada aluno, basta acrescentar as linhas de código marcadas em amarelo:

```
somaturma=0;
for (pos=0;pos<50;pos++)
{
    medialuno = (prova1[pos] + prova2[pos] + prova3[pos])/3;
    somaturma = somaturma + medialuno;
}
mediaturma = somaturma/50;
printf ("a media da turma e: %f.02",mediaturma);
```

TEM QUE EU TE EXPLICO!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Tipos estruturados de dados

Vetor: motivação e atribuição de valores

Vetor: percurso e processamento

VERIFICANDO O APRENDIZADO

1. VOCÊ DESEJA ARMAZENAR NA VARIÁVEL MEDIA A MÉDIA ARITMÉTICA ENTRE TODOS OS ELEMENTOS DE UM VETOR COM 20 NÚMERO REAIS CHAMADO VET. O TRECHO DE CÓDIGO CUJA ESTRUTURA REPETITIVA PERMITE QUE ISSO SEJA FEITO É:

```
A) soma=0;
for (ind=0;ind<=20;ind++)
{ soma=soma+VET[ind]; }
media=soma/20;
```

B) soma=0;
for (ind=0;ind<=20;ind++)
{ soma=VET[ind]; }
media=soma/20;

C) soma=0;
for (ind=0;ind=20)
{ soma=soma+VET[ind]; }
media=soma/20;

D) soma=0;
for (ind=0;ind<20;ind++)
{ soma=soma+VET[ind]; }
media=soma/20;

2. CONSIDERE UM VETOR DE 15 ELEMENTOS DO TIPO CHARACTER, CHAMADO VOGAIS. VOCÊ PRECISA QUE O PROGRAMA QUE MANIPULA ESSE VETOR CONTABILIZE A QUANTIDADE DE VOGAIS “A” OU “E” QUE NELE ESTÃO ARMAZENADAS E GUARDAR O TOTAL NA VARIÁVEL CONT. ASSINALE O TRECHO DE CÓDIGO QUE EXECUTA ESSA CONTAGEM CORRETAMENTE:

A) cont=0;
for (ind=0;ind<=14;ind++)
{
if (vogais[ind]=='a' || vogais[ind]=='e')
cont=cont+1;
}

B) cont=0;
for (ind=0;ind<=15;ind++)
{
if (vogais[ind]=='a' && vogais[ind]=='e')
cont=cont+1;
}

C) cont=0;
for (ind=0;ind<=14;ind++)
{
if (vogais[ind]=='a' or vogais[ind]=='e')
cont+1;
}

D) for (ind=0;ind<=14;ind++)
{
if (vogais[ind]=='a' || vogais[ind]=='e')

```
cont+=1;
```

```
}
```

GABARITO

1. Você deseja armazenar na variável MEDIA a média aritmética entre todos os elementos de um vetor com 20 número reais chamado VET. O trecho de código cuja estrutura repetitiva permite que isso seja feito é:

A alternativa **"D "** está correta.

São 20 elementos. Logo, a repetição FOR deve começar com ZERO (primeiro elemento do vetor), ir até 19 (última posição) e ser incrementada de 1 em 1. No interior da repetição, devemos acumular a soma dos números e, ao final, dividi-la por 20, que corresponde à média.

Vamos analisar cada alternativa:

A) Está errada, pois ind=20 não existe e, ao tentar acessar essa posição do vetor, resultará em erro.

B) Está errada por dois motivos. Primeiro, Ind não pode ser igual a 20 (acima). Segundo, o acúmulo da soma está errado, pois encerraria a repetição armazenando o último elemento do vetor.

C) Está errado, pois a condição final = 20 vai sair da repetição no primeiro laço.

2. Considere um vetor de 15 elementos do tipo character, chamado VOGAIS. Você precisa que o programa que manipula esse vetor contabilize a quantidade de vogais "A" ou "E" que nele estão armazenadas e guardar o total na variável CONT. Assinale o trecho de código que executa essa contagem corretamente:

A alternativa **"A "** está correta.

São 15 elementos. Logo, a repetição (FOR, no caso) deve começar na posição ZERO (primeiro elemento do vetor) e ir até a posição 14 (último elemento do vetor). Dentro da repetição, devemos verificar se o elemento é a vogal "a" ou "e" e, em caso positivo, contabilizar mais 1 na variável CONT, que, antes da repetição, deve ser inicializada.

Vamos analisar cada alternativa:

B) A repetição vai até 15 e tenta acessar o elemento vogais[15], que não existe. Além disso, o operador lógico OU está errado. || é o correto.

C) O operador lógico ou está errado. || é o correto. Além disso, o comando "cont+1" não faz nada, quando o correto seria cont=cont+1, cont++ ou cont+=1.

D) A variável cont não está sendo inicializada e, quando for incrementada, dará erro. Além disso, o comando "cont+!=" está errado. O correto seria cont=cont+1, cont++ ou cont+=1.

MÓDULO 2

⦿ Empregar a matriz para armazenamento de dados em um programa

MATRIZ X VETOR

A matriz é uma estrutura de dados homogênea, tal qual o vetor. Por isso, todos os seus elementos são do mesmo tipo. A diferença fundamental da matriz para o vetor é a quantidade de índices que são usados para acessar um elemento:

No vetor, cada elemento é acessado pelo seu índice, que representa a posição relativa desse elemento no vetor.



Na matriz, o acesso a cada elemento ocorre pelo uso de dois índices.

Quando estudamos vetores, vimos que, para representar 3 notas de 50 alunos de uma turma, foi preciso usar 3 vetores de elementos do tipo real, conforme a declaração a seguir:

```
float prova1[50], prova2[50], prova3[50];
```

Vetor prova 1

0	1	2	3	47	48	49
7.00	8.50	9.35	8.45	9.00	7.90	9.60

Imagem: Elaborado por Marcelo Vasques de Oliveira

Vetor prova 2

0	1	2	3	47	48	49
9.70	9.20	8.95	9.00	10.00	6.40	4.30

Imagem: Elaborado por Marcelo Vasques de Oliveira

Vetor prova 3

0	1	2	3	47	48	49
10.00	8.50	7.50	8.80	9.20	8.00	9.00

Imagem: Elaborado por Marcelo Vasques de Oliveira

Com esta estrutura, armazenamos em cada vetor os dados de 1 prova (são 3). Convencionamos que cada aluno tem suas notas no mesmo índice de cada um dos 3 vetores.





Foto: Shutetrstock.com

Com isso, o aluno 1 tem suas notas na posição 0 (zero) de cada vetor, o aluno 2, na posição 1, e assim por diante, até que, na posição de cada vetor, temos as notas do último aluno.

À medida que aumenta o número de provas, precisamos de mais vetores. Se fossem 10 provas, seriam necessários 10 vetores de 50 elementos do tipo float (real). Se fossem 20 provas, seriam necessários 20 vetores, e assim sucessivamente. Manipular 3 vetores pode ser até factível, porém, com mais de 20, começa a complicar.



Foto: Shutetrstock.com



A matriz é a estrutura de dados ideal para esse tipo de situação, pois, em vez de N vetores, declaramos e usamos apenas 1 matriz. É como se juntássemos os N vetores – no caso exemplificado, os 3 vetores.

0	1	2	3			47	48	49
7.00	8.50	9.35	8.45	9.00	7.90	9.60
9.70	9.20	8.95	9.00	10.00	6.40	4.30
10.00	8.50	7.50	8.80	9.20	8.00	9.00

Imagem: Elaborado por Marcelo Vasques de Oliveira

Além das 50 posições (em cada coluna), passamos a ter um segundo índice, que seriam 3 linhas – começando em 0 e indo até 2 no caso apresentado.

	0	1	2	3	47	48	49
0	8.50	9.35	8.45	9.00	7.90	9.60
1	9.20	8.95	9.00	10.00	6.40	4.30
2	8.50	7.50	8.80	9.20	8.00	9.00

Imagem: Elaborado por Marcelo Vasques de Oliveira

CONCEITO DE MATRIZ

A matriz é uma estrutura de dados homogênea, na qual usamos dois índices para acessar cada elemento.

Uma matriz é composta por linhas e colunas, tal qual uma planilha Excel, que também tem cada elemento referenciado por uma coluna (A, B, C, D, E) e uma linha (1 ... 10).

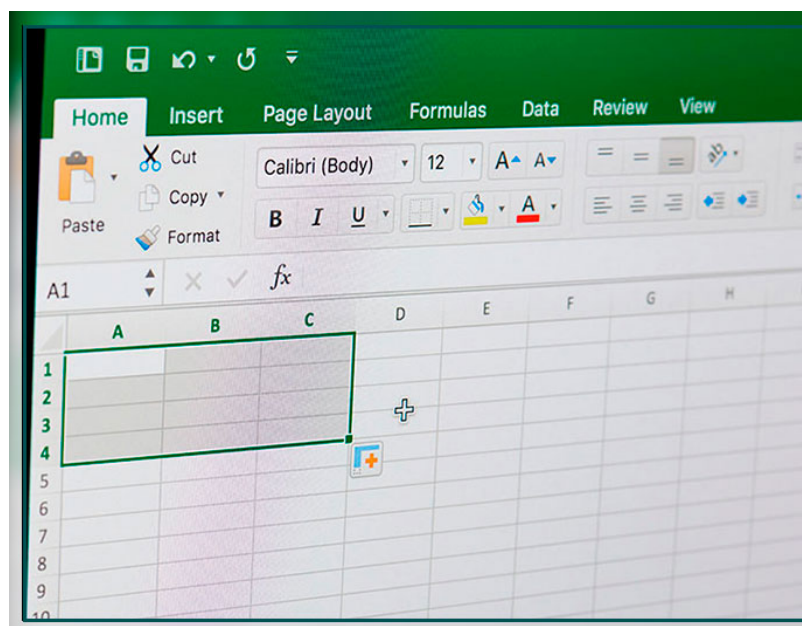


Foto: Shutterstock.com

A diferença fundamental entre uma planilha Excel e uma matriz é que, na primeira, podemos armazenar dados de diferentes tipos.

EXEMPLO DE PLANILHA EXCEL COM DIVERSOS TIPOS DE DADOS

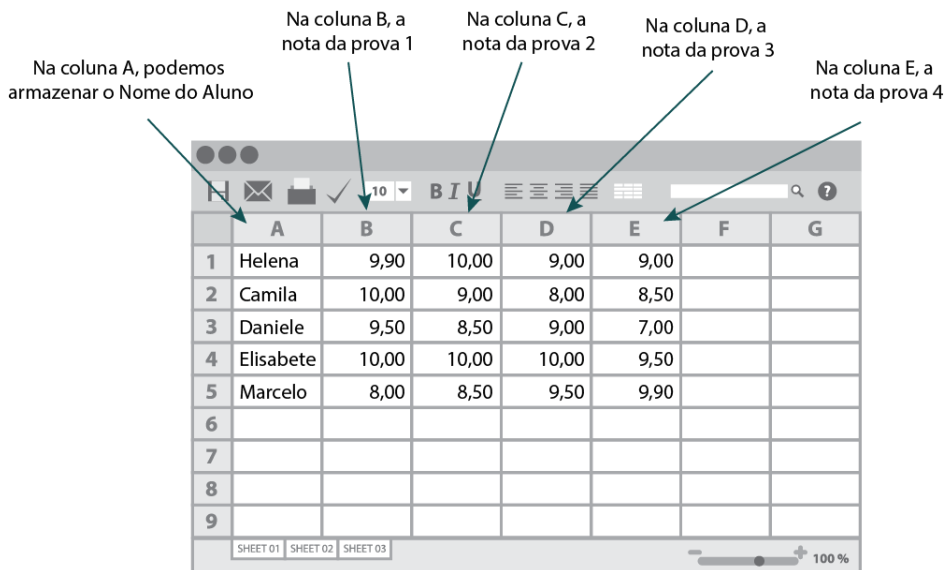


Imagem: Elaborado por Marcelo Vasques de Oliveira

O nome do aluno usa um tipo de dado, e as notas das provas usam outro.

EXEMPLO DE PLANILHA MATRIZ COM DADOS HOMOGÊNEOS

Em uma matriz, podemos armazenar somente dados de um único tipo:

	0	1	2	3
0	9,90	10,00	9,00	9,00
1	10,00	9,00	8,00	8,50
2	9,50	8,50	9,00	7,00
3	10,00	10,00	10,00	9,50
4	8,00	8,50	9,50	9,90

Imagem: Elaborado por Marcelo Vasques de Oliveira

DECLARAÇÃO DA MATRIZ

Como qualquer variável, a matriz precisa ser declarada na maioria das linguagens de programação que assim o exigem, como é o caso da linguagem C. De forma geral, as matrizes são declaradas da seguinte maneira, em Portugol Studio e na linguagem C:

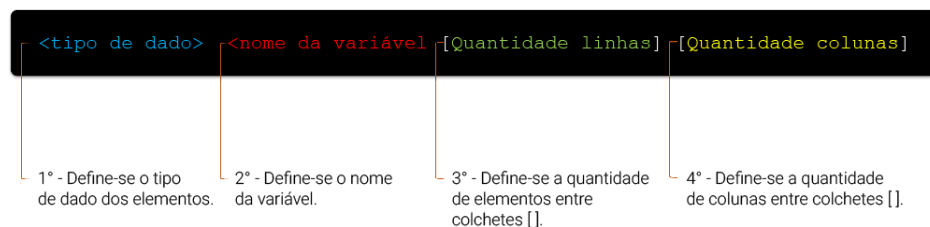


Imagem: Elaborado por Marcelo Vasques de Oliveira

Vamos ver imagens que ilustram uma matriz de 3 linhas e 5 colunas de elementos inteiros, chamada Mat_Num:

	0	1	2	3	4
0					
1					
2					

Mat_Num

Imagem: Elaborado por Marcelo Vasques de Oliveira

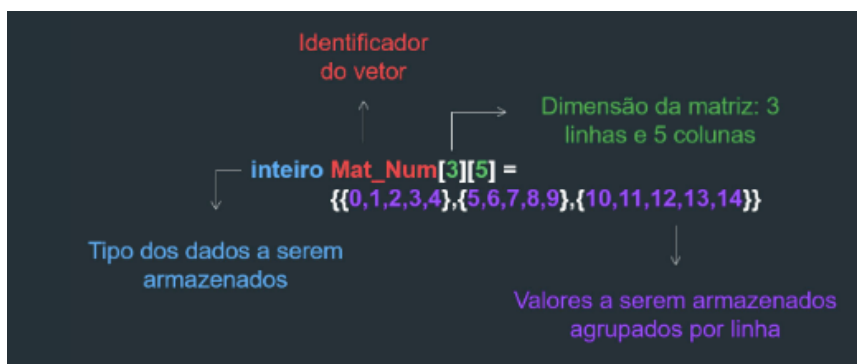


Imagem: Elaborado por Marcelo Vasques de Oliveira

Tanto no Portugol Studio quanto na linguagem C, o conjunto de elementos a serem armazenados na matriz deve estar definido entre chaves { }.

Existem as chaves gerais nos extremos.

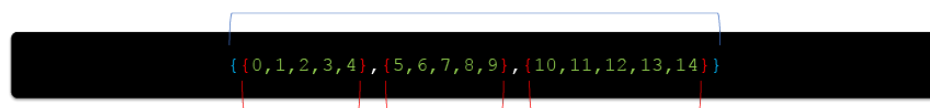


Imagem: Elaborado por Marcelo Vasques de Oliveira

E existem as chaves de cada linha.

A tabela mostra como ficará a matriz após a inicialização anterior de seus elementos:

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14

Imagem: Elaborado por Marcelo Vasques de Oliveira

Imagem da matriz preenchida após a inicialização

A imagem mostra a inicialização da matriz anterior, com 3 linhas e 5 colunas:

```
int Mat_Num[3][5] = {{0,1,2,3,4},{5,6,7,8,9},{10,11,12,13,14}};
```

Elementos da linha 0

Elementos da linha 1

Elementos da linha 2

Imagem: Elaborado por Marcelo Vasques de Oliveira

Vamos ver outros exemplos de declaração de matrizes:

```
// Portugol Studio
inteiro números[100][200] → matriz de 100x200 elementos do tipo inteiro
real notas [20][30] → matriz de 20x30 elementos do tipo real
caractere vogais [5][10]→ matriz de 5x10 elementos do tipo caractere
```

```
// Linguagem C
int numeros[100][200];
float notas [20][30];
char vogais [5][10];
```

Assim como as variáveis de tipo simples e os vetores, as variáveis do tipo matriz também podem ser inicializadas durante sua declaração ou em um comando no trecho de código, conforme vemos a seguir:

```
// Portugol Studio
inteiro numeros[3][5] = {{10,12,14,16,18},{0,9,8,7,6,5},{11,12,13,14,15}}
```

ou

```
inteiro numeros[3][5]
numeros = {{10,12,14,16,18},{0,9,8,7,6,5},{11,12,13,14,15}}
```

```
// Linguagem C
int numeros[3][5] = {{10,12,14,16,18},{0,9,8,6,5},{11,12,13,14,15}};
```

ou

```
int numeros[3][5];
numeros = {{10,12,14,16,18},{0,9,8,6,5},{11,12,13,14,15}};
```

ACESSO AOS ELEMENTOS DA MATRIZ

Os elementos de uma matriz são acessados, atribuídos, lidos ou exibidos, elemento a elemento, referenciando a linha e a coluna que ocupam, respectivamente.

Vamos analisar um exemplo?

Considere a matriz 3x5 (3 linhas x 5 colunas) de elementos do tipo inteiro, de nome mat, bem como os seguintes comandos, numerados de 1 a 15, tanto em Portugol Studio quanto em linguagem C:

// Portugol Studio

```
inteiro mat[3][5]
```

```
1 mat [0][0] = 1
```

```
2 mat [0][1] = 2
```

```
3 mat [0][2] = 3
```

```
4 mat [0][3] = 4
```

```
5 mat [0][4] = 5
```

```
6 mat [1][0] = 6
```

```
7 mat [1][1] = 7
```

```
8 mat [1][2] = 8
```

```
9 mat [1][3] = 9
```

```
10 mat [1][4] = 10
```

```
11 mat [2][0] = 11
```

```
12 mat [2][1] = 12
```

```
13 mat [2][2] = 13
```

```
14 mat [2][3] = 14
```

```
15 mat [3][3] = 15
```

// Linguagem C

```
mat[3][5];
```

```
1 mat [0][0] = 1;
```

```
2 mat [0][1] = 2;
```

```
3 mat [0][2] = 3;
```

```
4 mat [0][3] = 4;
```

```
5 mat [0][4] = 5;
```

```
6 mat [1][0] = 6;
```

```
7 mat [1][1] = 7;
```

```
8 mat [1][2] = 8;
```

```
9 mat [1][3] = 9;
```

```
10 mat [1][4] = 10;
```

```
11 mat [2][0] = 11;
```

```
12 mat [2][1] = 12;
```

```
13 mat [2][2] = 13;
```

```
14 mat [2][3] = 14;
```

```
15 mat [3][3] = 15;
```

Ao final da execução de cada comando associado às linhas numeradas de 1 a 15, teremos a matriz preenchida da seguinte forma:

	0	1	2	3	4
0	1	2	3	4	5
1	5	7	8	9	10
2	11	12	13	14	15

Imagem: Elaborado por Marcelo Vasques de Oliveira

Com base nesta tabela, acompanhe as considerações sobre matrizes:

O 1º elemento ocupa a posição [0][0], linha 0 e coluna 0: é o número 1.

O último elemento ocupa a posição [2][4], linha 2, coluna 4: é o número 15.

Considerando a matriz apresentada, veja o que vai acontecer com os comandos equivalentes em Portugol Studio e na linguagem C.

1. COMANDO PARA EXIBIR UM ELEMENTO DA MATRIZ

// Portugol Studio

escreva ("Linha 2, coluna 1 : ",mat[2][1])

// Linguagem C

printf("%d Linha 2, coluna 1: ",mat[2][1]);

O comando anterior exibe no dispositivo de saída o elemento que ocupa a posição de linha=2 e coluna=1. Avaliando a matriz apresentada, o elemento é 12 (doze).

2. COMANDO PARA ATRIBUIR VALOR A UM ELEMENTO DA MATRIZ

// Portugol Studio

mat[2][4]=90

// Linguagem C

Mat[2][4]=90;

O comando anterior armazena o valor 90 no elemento da linha 2 e coluna 4. O elemento, anterior ao comando, é 15, que é substituído pelo 90. A matriz ficará assim (observe o destaque em amarelo):

	0	1	2	3	4
0	1	2	3	4	5
1	5	7	8	9	10
2	11	12	13	14	90

Imagem: Elaborado por Marcelo Vasques de Oliveira

3. COMANDO PARA LER UM DADO DO DISPOSITIVO DE ENTRADA E ARMAZENAR NA MATRIZ

a) Lendo o dado de entrada direto para uma posição da matriz

```
// Portugol Studio
```

```
leia (mat[2][0])
```

```
// Linguagem C
```

```
scanf ("%d",&mat[2][0]);
```

O comando anterior armazena o valor lido pelo dispositivo de entrada na linha 2, coluna 0, da matriz. O elemento anterior ao comando é 11, substituído pelo valor lido.

Se o usuário digitar o número 99, como ficará a matriz?

Observe o destaque em amarelo:

	0	1	2	3	4
0	1	2	3	4	5
1	5	7	8	9	10
2	99	12	13	14	90

Imagem: Elaborado por Marcelo Vasques de Oliveira

b) Lendo o dado do dispositivo de entrada em uma variável e, depois, atribuindo o conteúdo dessa variável a uma posição da matriz

```
// Portugol Studio
```

```
leia (num)
```

```
mat[1][2]=num
```

```
// Linguagem C
```

```
scanf ("%d",&num);
```

```
mat[1][2]=num;
```

O comando anterior armazena em uma variável o valor lido pela digitação do usuário no dispositivo de entrada. Na sequência, armazena o conteúdo dessa variável na linha 1 e coluna 2 da matriz. O elemento anterior ao comando é 8, que é substituído pelo valor lido.

Se o usuário digitar o número 122, como ficará a matriz?

Observe o destaque em amarelo:

	0	1	2	3	4
0	1	2	3	4	5
1	5	7	122	9	10
2	99	12	13	14	90

Imagem: Elaborado por Marcelo Vasques de Oliveira

Os comandos a seguir resultarão em erro, pois acessam posições (índices da matriz) que não são válidas:

```
// Portugol Studio  
escreva (mat[0][5])  
mat[3][4]=901  
leia (mat[5][5])  
  
// Linguagem C  
>printf("%d linha 0, coluna 5 : ",mat[0][5]);  
mat[3][4]=901;  
scanf ("%d",&mat[5][5]);
```

DEMONSTRAÇÃO DA MATRIZ

I. O comando declara uma matriz para armazenar 2 notas de 30 alunos de uma turma:

```
// Portugol Studio  
real mat[2][30]  
ou  
real mat[30][2]  
  
// Linguagem C  
double mat[2][30];  
ou  
double mat[30][2];
```

II. O comando declara uma matriz para armazenar o sexo – masculino (M) ou feminino (F) – de 50 alunos de 3 turmas:

```
// Portugol Studio  
caracter mat[3][50]  
ou  
caracter mat[50][3]  
  
// Linguagem C  
char mat[3][50];  
ou  
char mat[50][3];
```

III. O comando declara uma matriz para armazenar 5 apostas de 10 alunos de uma turma e inicializa cada aposta com 0 (zero):

```
// Portugol Studio  
real mat[5][10] = {{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},  
{0,0,0,0,0,0,0,0,0,0}{0,0,0,0,0,0,0,0,0,0}}  
ou  
real mat[10][5] =  
{{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},  
{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}}
```

```
// Linguagem C

int mat[5][10] = {{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0}};

ou

float mat[10][5] =
{{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},
{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}};
```

IV. Se fossem 3 notas de 100 alunos, em vez de 10, seria inviável inicializar a matriz com 3x100 zeros entre as chaves. Se fossem 500 alunos, mais inviável ainda.

Nesses casos, como inicializar cada elemento da matriz com 0 (zero)?

RESPOSTA

Podemos usar dois comandos de repetição: um para percorrer cada linha e, dentro desta, outro para percorrer cada coluna de uma linha e atribuir o valor 0 (zero) a cada uma.

O comando de repetição mais adequado, neste caso, é o PARA (FOR), pois a repetição é para número fixo e conhecido de vezes – no caso 3 linhas x 100 colunas. Veja:

Portugol Studio	Linguagem C
<pre>para (linha=0;linha<3;linha++) { para (coluna=0;coluna<100;coluna++) { mat[linha,coluna]=0 } }</pre>	<pre>for (linha=0;linha<3;linha++) { for (coluna=0;coluna<100;coluna++) { mat[linha][coluna]=0; } }</pre>

Atenção! Para visualização completa da tabela utilize a rolagem horizontal

V. O trecho de código exibe no dispositivo de saída as 3 notas dos 100 alunos da turma:

```
// Portugol Studio

para (linha=0;linha<3;linha++)
{
escreva ("dados da linha: ",linha)
para (coluna=0;coluna<100;coluna++)
{
escreva (mat[linha,coluna])
}
}

// Linguagem C

for (linha=0;linha<3;linha++)
{
```

```
printf("%d \n dados da linha= ",linha);<
for (coluna=0;coluna<100;coluna++)
{
printf ("%d",mat[linha][coluna]);
}
}
```

VI. O trecho de código calcula e mostra a média de cada turma (soma das notas dos alunos dividida por 100, que é a quantidade de alunos):

```
// Portugol Studio
soma=0
para (linha=0;posicao<3;linha++)
{
para (coluna=0;coluna<100;coluna++)
{
soma=soma+mat[linha,coluna]
}
escreva ("A media da turma e : ",media/100)
}
```

```
// Linguagem C
soma=0;
for (linha=0;linha<3;linha++)
{
for (coluna=0;coluna<100;coluna++)
{
soma=soma+mat[linha][coluna];
}
printf("%f media da turma = ",soma/100);
}
```

MÃO NA MASSA

Vamos ver a teoria na prática.

Sugerimos que realize o seguinte procedimento:

A - Desenvolva o algoritmo em Portugol Studio e na linguagem C.

B - Garanta que sua solução funcione, executando algumas vezes, com dados distintos.

C - Veja a solução que sugerimos e compare com a sua.

ATIVIDADE 1

Faça um algoritmo que leia dados inteiros e armazene-os em uma matriz 3x4. Em seguida, mostre a quantidade de números pares e ímpares armazenados na matriz.

RESPOSTA

Portugol Studio	Linguagem C
<pre>inteiro mat[3][4], contpar=0,contimpar=0,linha,coluna escreva ("Digite valor para os elementos da matriz"); para (linha=0;posicao<3;linha++) para (coluna=0;coluna<4;coluna++) { escreva ("elemento : ") leia (mat[linha][coluna]) } escreva ("***** Saida de Dados *****") para (linha=0;posicao<3;linha++) para (coluna=0;coluna<4;coluna++) { se (mat[linha][coluna] % 2)==0 contapar++ senao contimpar++ } escreva ("total de pares: ",contpar) escreva ("total de impares: ",contimpar)</pre>	<pre>int mat[3][4],lin, col,contpar=0,contimpar=0; printf ("\nDigite valor para os elementos da matriz\n\n"); for (lin=0; lin<3; lin++) for (col=0; col<4; col++) { printf ("\nElemento[%d][%d] = ", lin, col); scanf ("%d", &mat[lin][col]); } printf("\n\n***** Saida de Dados ***** \n\n"); for (lin=0; lin<3; lin++) for (col=0; col<4; col++) { if (mat[lin][col] % 2==0) contpar++; else contimpar++; } printf ("\nPares: %d ",contpar); printf ("\nImpares: %d ",contimpar);</pre>

Atenção! Para visualização completa da tabela utilize a rolagem horizontal

Veja as telas com o código em linguagem C, na ferramenta online, e a execução do programa:

```
main.c
1 #include<stdio.h>
2 int main()
3 {
4     int mat[3][4],lin, col,contpar=0,contimpar=0;
5     printf("\nDigite valor para os elementos da matriz\n\n");
6     for (lin=0; lin<3; lin++)
7         for (col=0; col<4; col++)
8         {
9             printf("\nElemento[%d][%d] = ", lin, col);
10            scanf("%d", &mat[lin][col]);
11        }
12    printf("\n\n***** Saida de Dados ***** \n\n");
13    for (lin=0; lin<3; lin++)
14        for (col=0; col<4; col++)
15        {
16            if (mat[lin][col] % 2==0)
17                contpar++;
18            else
19                contimpar++;
20        }
21    printf("\nPares: %d ",contpar);
22    printf("\nImpares: %d ",contimpar);
23
24    return(0);
25 }
26
```

Imagem: Elaborado por Marcelo Vasques de Oliveira

```
Input
Digite valor para os elementos da matriz

Elemento[0][0] = 1
Elemento[0][1] = 2
Elemento[0][2] = 3
Elemento[0][3] = 4
Elemento[1][0] = 5
Elemento[1][1] = 6
Elemento[1][2] = 7
Elemento[1][3] = 8
Elemento[2][0] = 9
Elemento[2][1] = 10
Elemento[2][2] = 11
Elemento[2][3] = 12

***** Saida de Dados *****

Pares: 6
Impares: 6
```

Imagem: Elaborado por Marcelo Vasques de Oliveira

ATIVIDADE 2

Faça um algoritmo que leia números inteiros e armazene-os na matriz 4x4. Porém, na diagonal principal, não armazene o número lido, e sim um 0 (zero).

Na diagonal principal, os elementos têm linha = coluna: [0][0], [1][1], [2][2], [3][3].



Neste vídeo, o professor **Marcelo Vasques** resolverá a atividade 2.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Solução em C

Abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

```
#include <stdio.h>

// video 1 do modulo 2

int main()
{
    int mat[4][4],lin,col;

    printf ("\nDigite valor para os elementos da matriz\n");

    for (lin=0;lin<4;lin++)
        for (col=0;col<4;col++)
            if (lin==col)
            {
                printf ("Elemento[%d][%d] = 0 \n",lin,col);
                mat[lin][col]=0;
            }
            else
            {
                printf ("Elemento[%d][%d] = ",lin,col);
                scanf ("%d",&mat[lin][col]) ;
            }

    printf ("\nListagem dos elementos da matriz\n");

    for (lin=0;lin<4;lin++)
        for (col=0;col<4;col++)
            printf("\nElemento[%d][%d] = %d",lin,col,mat[lin][col]);
}
```

Atenção! Para visualização completa do código utilize a rolagem horizontal

ATIVIDADE 3

Faça um algoritmo que leia uma matriz 4x4 de números inteiros. Gere uma segunda matriz, na qual as linhas são as colunas da matriz 1, e as colunas são as linhas da matriz 1.

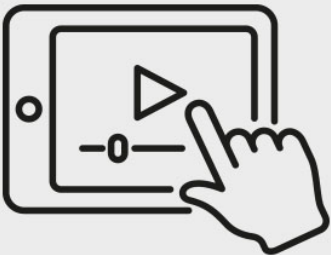
Matriz 1				Matriz 2 gerada			
1	2	3	4	1	5	9	13
5	6	7	8	2	6	10	14
9	10	11	12	3	7	11	15
13	14	15	16	4	8	12	16

Imagem: Elaborado por Marcelo Vasques de Oliveira



Neste vídeo, o professor **Marcelo Vasques** resolverá a atividade 3.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Solução em C

Abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

```
#include <stdio.h>

// video 2 do modulo 2

int main()

{

int mato[4][4],matg[4][4],lin,col;

printf ("\n Digite a matriz original \n");

for (lin=0;lin<4;lin++)

for (col=0;col<4;col++)

{

scanf ("%d",&mato[lin][col]);
```



```

matg[col][lin]=mato[lin][col];

}

printf ("\n Matriz gerada \n");

for (lin=0;lin<4;lin++)

{

for (col=0;col<4;col++)

printf ("%d ",matg[lin][col]);

printf ("\n");

}

}

```

Atenção! Para visualização completa do código utilize a rolagem horizontal

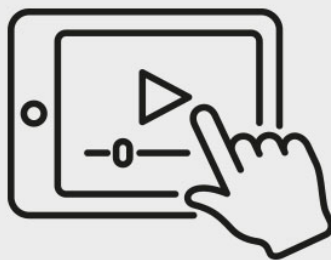
ATIVIDADE 4

Faça um algoritmo que leia dados e armazene em uma matriz 3x3 de números inteiros. Em seguida, mostre os elementos que sejam iguais ao maior número armazenado na matriz.



Neste vídeo, o professor **Marcelo Vasques** resolverá a atividade 4.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



1ª Solução em C

Abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

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

```
// Video 3 do modulo 2 (solucao 2)
```

```
int main()
```

```
{
```

```

int mat1[3][3],lin, col,maior=0,contigual=0;

printf("\n Digite valor para os elementos da matriz \n\n ");

for ( lin=0; lin<3; lin++ )

for ( col=0; col<3; col++ )

{

printf ("Elemento[%d][%d] = ",lin,col);

scanf ("%d", &mat1[lin][col]);

if (mat1[lin][col]>maior)

maior=mat1[lin][col];

}

for (lin=0; lin<3; lin++ )

for (col=0; col<3; col++ )

{

if (mat1[lin][col]==maior)

contigual++;

}

printf("\n Maior: %d ",maior);

printf("\n = maior: %d ",contigual);

}

```

Atenção! Para visualização completa do código utilize a rolagem horizontal

2ª Solução em C

Abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

```

#include <stdio.h>

// video 3 do modulo 2

int main()

{

int mat[3][3],lin, col,maior=0,contigual=0;

printf ("\nDigite valor para os elementos da matriz\n\n"); for ( lin=0; lin<3; lin++ )

for ( col=0; col<3; col++ )

{

printf ("Elemento[%d][%d] = ", lin, col);

```

```
scanf ("%d", &mat[lin][col]);

if (mat[lin][col]>maior)

{

maior=mat[lin][col];

contigual=1;

}

else

if (mat[lin][col]==maior)

contigual++;

}

printf("\n Maior: %d ",maior);

printf("\n Ocorrenias do maior: %d ",contigual);

}
```

Atenção! Para visualização completa do código utilize a rolagem horizontal

ATIVIDADE 5

Faça um programa que gere uma matriz 5x5, conforme esta sequência:

0	1	1	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Imagem: Elaborado por Marcelo Vasques de Oliveira



Neste vídeo, o professor **Marcelo Vasques** resolverá a atividade 5.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Solução em C

Abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

```
#include <stdio.h>

int main()
{
    int mat[5][5],lin, col;

    for (lin=0; lin<5; lin++ )
    {
        for ( col=0; col<5; col++)
        {
            mat[lin][col]=1;

            if (lin==col)
            mat[lin][col]=0;

            printf("%d",mat[lin][col]);

        }

        printf ("\n");

    }
}
```

Atenção! Para visualização completa do código utilize a rolagem horizontal

TEM QUE EU TE EXPLICO!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Matriz: Motivação

Matriz: Atribuição de valores

VERIFICANDO O APRENDIZADO

1. CONSIDERE UMA MATRIZ 10X30, NA QUAL ARMAZENAMOS AS NOTAS DE 10 PROVAS DE 30 ALUNOS DE UMA TURMA. QUAL É O TRECHO DE CÓDIGO CORRETO PARA LER OS DADOS, ARMAZENAR NESSA MATRIZ, BEM COMO ENCONTRAR E MOSTRAR A MAIOR NOTA DE CADA PROVA?

A) maior=0;
for (lin=0;lin<30;lin++)
{ for(col=0;col<10;col++)
{ scanf ("%f ",&mat[lin][col];
if (mat[lin][col]>maior)
maior= mat[lin][col];
}
printf ("%f maior da turma: ",maior)
}

B) maior=0;
for (lin=0;lin<10;lin++)
{ for(col=0;col<30;col++)
{ scanf ("%f ",&mat[lin][col];
if (mat[lin][col]>maior)
maior= mat[lin][col];
}
printf ("%f maior da turma: ",maior)
}

C) for (lin=0;lin<10;lin++)
{ maior=10.00;
for(col=0;col<30;col++)
{ scanf ("%f ",&mat[lin][col];
if (mat[lin][col]>maior)
maior= mat[lin][col];
}
printf ("%f maior da turma: ",maior)
}

D) for (lin=0;lin<10;lin++)
{ maior=0;
for(col=0;col<30;col++)

```
{ scanf ("%f ", &mat[lin][col];  
if (mat[lin][col]>maior)  
maior= mat[lin][col];  
}  
printf ("%f maior da turma: ", maior)  
}
```

2. CONSIDERE UMA MATRIZ 3X3 DE INTEIROS, COM SEUS ELEMENTOS ARMAZENADOS, DE NOME MAT. SUA NECESSIDADE É EXIBIR OS ELEMENTOS DE SUA DIAGONAL PRINCIPAL. PARA TAL, O TRECHO DE CÓDIGO NA LINGUAGEM C É:

A) for (ind=0;ind<3;ind++)

```
{ printf ("%f : ", mat[i][i]); }
```

B) for (ind=0;ind<=3;ind++)

```
{ printf ("%f : ", mat[i][i]); }
```

C) for (ind=1;ind<3;ind++)

```
{ printf ("%f : ", mat[i][i]); }
```

D) for (ind=1;ind<=3;ind++)

```
{ printf ("%f : ", mat[i][i]); }
```

GABARITO

1. Considere uma matriz 10x30, na qual armazenamos as notas de 10 provas de 30 alunos de uma turma. Qual é o trecho de código correto para ler os dados, armazenar nessa matriz, bem como encontrar e mostrar a maior nota de cada prova?

A alternativa **"D "** está correta.

Vamos analisar cada alternativa:

A) A variável maior inicializada nesse ponto vai mostrar a maior nota de toda a matriz, e não apenas a da turma (que seria ao final de cada linha). A dupla de FOR está errada, pois são 10 linhas e 30 colunas: o primeiro FOR deve ir até <10, e o segundo, até <30.

B) A variável maior inicializada nesse ponto vai mostrar a maior nota de toda a matriz, e não apenas da turma (que seria ao final de cada linha).

C) A variável maior inicializada com 10.00 nunca vai receber a menor nota, pois todas as notas serão menores que ela.

2. Considere uma matriz 3x3 de inteiros, com seus elementos armazenados, de nome MAT. Sua necessidade é exibir os elementos de sua diagonal principal. Para tal, o trecho de código na linguagem C é:

A alternativa **"A "** está correta.

Vamos analisar cada alternativa:

B) A matriz é 3x3, porém, com essa repetição, tentará acessar a diagonal [4,4] da matriz (que não existe).

C) A matriz é 3x3, porém, com essa repetição, não acessará o elemento [0,0] da diagonal principal da matriz e tentará acessar a diagonal [4,4] da matriz (que não existe).

D) A matriz é 3x3, porém, com essa repetição, não acessará o elemento [0,0] da diagonal principal da matriz.

CONCLUSÃO

CONSIDERAÇÕES FINAIS

As estruturas de dados são fundamentais para processarmos um conjunto de dados em memória e posterior persistência, se for o caso. Existem diversos modelos e implementações de estruturas de dados nas linguagens de programação.

Aqui, estudamos duas na linguagem C:

Estáticas – porque não podemos variar seu tamanho durante a execução do programa, após sua declaração, embora haja uma forma de burlar o tamanho máximo de um vetor, por exemplo;

Homogêneas – porque a estrutura armazena apenas dados do mesmo tipo em todos os seus elementos.

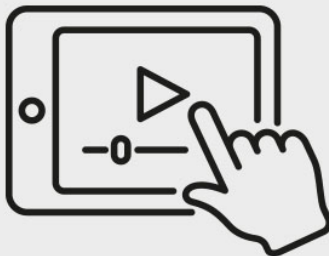
As estruturas estudadas chamam-se, respectivamente, vetor e matriz. A diferença entre elas é a quantidade de índices usados para acessar cada elemento. Enquanto no vetor, utilizamos apenas um índice, na matriz, são necessários dois índices.



VETORES E MATRIZES

Neste vídeo, o professor **Marcelo Vasques** apresenta um resumo do tema Vetores e Matrizes.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Para ouvir um *podcast* sobre o assunto, acesse a versão online deste conteúdo.



REFERÊNCIAS

ANDRADE, M. C. **Algoritmos**. Rio de Janeiro: SESES, 2014.

ASCENCIO, A. F. G.; CAMPOS, E. A. V. **Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ e Java**. 3. ed. São Paulo: Pearson Education, 2009.

FORBELLONE, A. L. V.; EBERSPACHER, H. **Lógica de programação**. 3. ed. São Paulo: Makron Books, 2005.

MANZANO, J. A. N. G.; OLIVEIRA, J. F. **Algoritmos: lógica para desenvolvimento de programação de computadores**. São Paulo: Erica, 2016.

SOFFNER, R. **Algoritmos e programação em linguagem C**. 1. ed. São Paulo: Saraiva, 2013.

EXPLORE+

Pesquise e leia o seguinte artigo:

GATTO, E. C. **Introdução à matriz**. [S. l.]: Portal Embarcados, dez. 2016.

Busque e acesse a seguinte ferramenta:

Online C++ Compiler – online editor – para quando houver necessidade de compilar um programa em linguagem C e não tiver à disposição o DEV C++ instalado em seu computador.

CONTEUDISTA

Marcelo Vasques de Oliveira

 **CURRÍCULO LATTES**