



Vetores e Matrizes

Prof. Marcelo Vasques de Oliveira

Descrição

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

Vetor para armazenamento de dados

Empregar o vetor para armazenamento de dados em um programa.

Módulo 2

Matriz para armazenamento de dados

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.

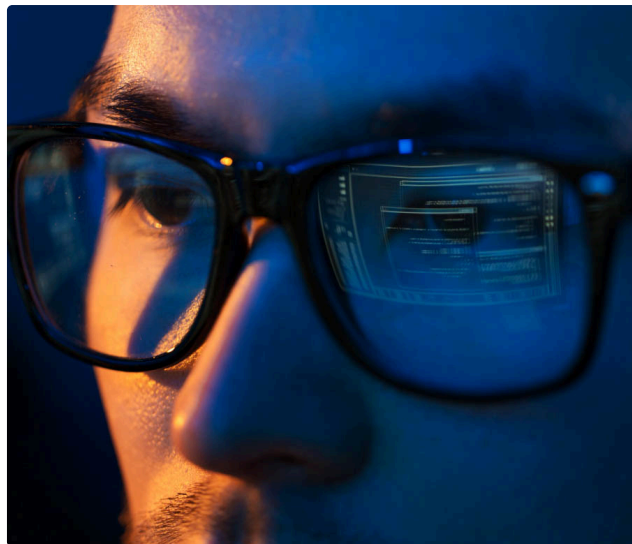
Até o momento, os programas definidos usaram [variáveis](#) de tipos de dados simples, nas quais apenas um dado pode ser armazenado em cada variável por vez.

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.

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.

Variáveis

Posições de memória acessadas pelo seu nome.



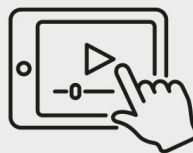
1 - Vetor para armazenamento de dados

Ao final deste módulo, você será capaz de reconhecer e empregar o vetor para armazenamento de dados em um programa.

Armazenamento de dados

Neste vídeo, vamos mergulhar no mundo das variáveis e dos tipos de dados em C. Aprenderemos sobre os tipos de dados simples, como inteiros, caracteres e ponto flutuante, e também sobre os tipos de dados compostos, como structs e arrays.

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



Variáveis, tipos simples e compostos

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.

Considere um programa que leia 100 números inteiros e positivos, e mostre o maior deles. Analise o trecho do código, em que as linhas estão numeradas para efeitos de explicação:

c



Pelo código analisado, podemos chegar às seguintes conclusões:

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 conteúdo 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.

Existem problemas que não podem ser resolvidos usando apenas variáveis do tipo simples. É o que veremos a seguir.

Exemplo

Faça um programa que leia 100 números inteiros e mostre-os na ordem inversa em que foram lidos. Ao lermos o segundo número na mesma variável, o primeiro número lido será perdido. Assim, não teremos como

exibi-los posteriormente, tampouco como usar 100 variáveis simples do tipo inteiro, como anteriormente, 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.” Logo, concluímos que existem diferenças entre tipos de **dados simples e estruturados**. Observe:

Tipos de dados simples		Tipos de dados estruturados ou compostos
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.	X	Podem armazenar um conjunto de dados, simultaneamente, em memória, e, claro, usam mais de uma posição de memória.

Agregação e alocação dos dados estruturados

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:


Crítérios	Classificação	Descrição
Forma de alocação	Estáticos	Dados declarados no início do programa, e mantidos assim

Vetores e Matrizes		
Cr�terios	Classifica��o	Descri��o
Tipos de dados		o final de sua execu��o.
		Dados criados durante a execu��o do programa
	Homog�nicos	Todos os dados s�o do mesmo t
	Heterog�neos	Os dados da estrutura podem ser de tipos diferentes.

Vetores

Neste v deo, vamos explorar os vetores em C, uma das estruturas de dados mais utilizadas. Aprenderemos como declarar, inicializar e manipular vetores, al m de abordar t cnicas avan adas, como ordena  o e busca.

Para assistir a um v deo sobre o assunto, acesse a vers o online deste conte do.



Conceito

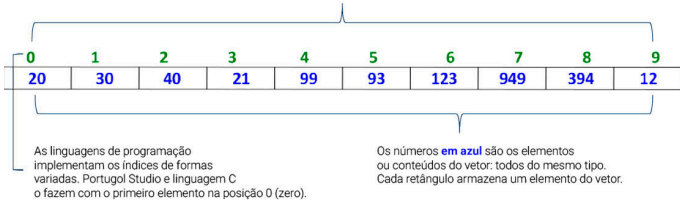
O **vetor**   um tipo de dado estruturado ou composto, est tico e homog neo, dispon vel na maioria das linguagens de programa  o, incluindo a C.

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

Como o vetor organiza os dados que armazena?

O vetor usa posi  es consecutivas de mem ria e simula esse funcionamento a partir de  ndices. Por isso,   uma estrutura **indexada**.

Os n meros em **verde** representam os  ndices do vetor, usados para acessar cada um de seus elementos. A posi  o 0   a primeira, e a posi  o 9   a  ltima, como visto na seguinte imagem:



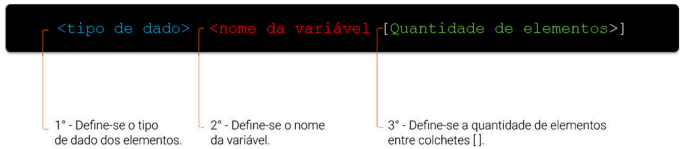
Representação de um vetor.

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:

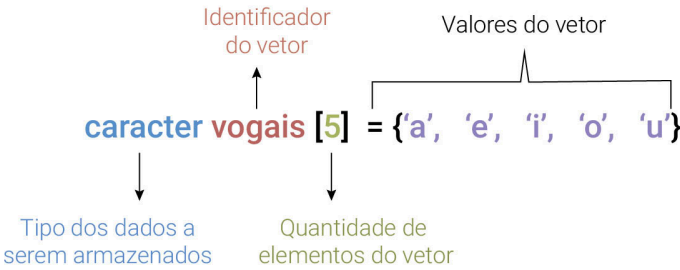
Como os vetores, em geral, são declarados em Portugol Studio e na linguagem C?

Essa declaração é feita da seguinte maneira:



Declaração do vetor.

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



Declaração e inicialização do vetor VOGAIS - Portugol Studio

Como fica declarado o vetor representado na imagem anterior, em Portugol Studio e na linguagem C? É só observar os seguintes códigos:

Portugol



C



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



C



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

Exemplos

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

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



C



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

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.

Comando para exibir um elemento do vetor

O comando a seguir exibe no dispositivo de saída o elemento que ocupa a posição (índice) 1 do vetor. O elemento da posição 1 é o 6:

Portugol



C



Comando para atribuir valor a um elemento do vetor

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

Portugol



C



Como ficará o vetor? Veremos a seguir:

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

Vetor int vet[10]

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

Portugol

C

Como ficará o vetor? Vejamos:

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

Vetor int vet[10]

Comando para ler um dado do dispositivo de entrada e armazenar em posição do vetor

Lendo o dado de entrada direto para uma posição do vetor, fica da seguinte forma:

Portugol

C

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

Vetor int vet[10]

Lendo o dado do dispositivo de entrada em uma variável e, depois, atribuindo o conteúdo dessa variável a uma posição do vetor, fica da seguinte forma:

Portugol



C



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

Vetor int vet[10]

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

Portugol



C



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

Neste vídeo, vamos explorar a manipulação e o processamento de strings em C. Vamos aprender sobre a declaração, inicialização e manipulação de strings, incluindo funções e técnicas avançadas para trabalhar com texto.

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



Conceito e declaração

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.

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

Representação de um vetor de caracteres.

Como é feita a declaração da cadeia de caracteres? Da seguinte forma:

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:

C 

Ou podemos fazer desta maneira:

C 

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"`:

```
C
```



Note que não usamos `&`. O comando anterior seria equivalente ao trecho de código:

```
C
```



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

```
C
```



O comando anterior seria equivalente ao trecho de código:

```
Javascript
```



Exemplos de uso e manipulação de vetores e cadeias de caracteres

Exemplo 1

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

```
Portugol
```



```
C
```



Exemplo 2

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
```



```
C
```



Exemplo 3

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

```
Portugol
```



```
C
```



Exemplo 4

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.

Como inicializar cada elemento do vetor com 0 (zero)?

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

O comando de repetição mais adequado é o PARA (FOR), pois a repetição é para um 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
```



c



Exemplo 5

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.

Em formato de código, fica da seguinte forma:

```
Portugol
```



c



Exemplo 6

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

```
Portugol
```



C



Exemplo 7

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

Portugol



C



Exemplo 8

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



C



Exemplo 9

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



C



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.

Vamos praticar

Vamos por em prática o que aprendemos no decorrer desse conteúdo. Sugerimos que você realize os seguintes procedimentos:

Desenvolvimento



Desenvolva, você mesmo, o algoritmo em Portugol Studio e, depois, na linguagem C;

Confirmação



Garanta que sua solução funcione com dados distintos,

executando algumas vezes;

Comparação



Veja a solução que sugerimos e compare com a sua.

Vamos iniciar com uma prática 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 DEV C++. 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.

Prática 1

Faça um programa que leia 100 números inteiros e mostre-os na ordem inversa em que foram lidos. Em relação à **estrutura de dados**, o vetor vai armazenar 100 números inteiros.

Temos a seguinte resolução:

1. Inicialmente, devem ser lidos cada um dos 100 números do dispositivo de entrada e armazenados em uma posição do vetor;
2. 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 os seguintes encaminhamentos:



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.

Agora, pratique você mesmo usando o emulador a seguir:

Atividade 1

 TUTORIAL  COPIAR

C



null

null



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

Portugol



C



Prática 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, no emulador a seguir.

Estrutura de dados: O vetor armazenará 20 números reais.

Atividade 2

 TUTORIAL  COPIAR

C

null

null



Resolução

Neste vídeo, a prática 2 será resolvida.

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

Além disso, para a solução em C, abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

C



Leia uma sequência de letras, terminada na letra ("z"), e mostre quantas vezes cada vogal (a, e, i, o, u) apareceu, no emulador a seguir.

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".

Atividade 3

 TUTORIAL  COPIAR

C



null

null



Resolução

Neste vídeo, a prática 3 será resolvida.

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



Além disso, para a solução em C, abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

C



Prática 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:

- 1. Se o número for par, preencha a mesma posição do segundo vetor com o número sucessor do contido na mesma posição do primeiro vetor;
- 2. 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 no emulador a seguir:

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

Atividade 5

 TUTORIAL  COPIAR

C



null

null



Resolução

Neste vídeo, a prática 4 será resolvida.

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



Além disso, para a solução em C, abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

C

📄

Prática 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.

Atividade 6

📘 TUTORIAL

📄 COPIAR

C

null

null



Resolução

Neste vídeo, a prática 5 será resolvida.

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



Além disso, para a solução em C, abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

C



Prática 6

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 praticar as duas ações a seguir:

Identificar

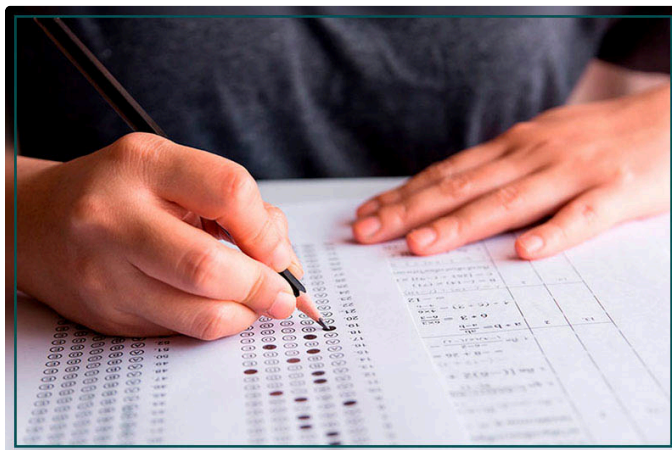
As estruturas de dados necessárias à solução do problema.

Escrever

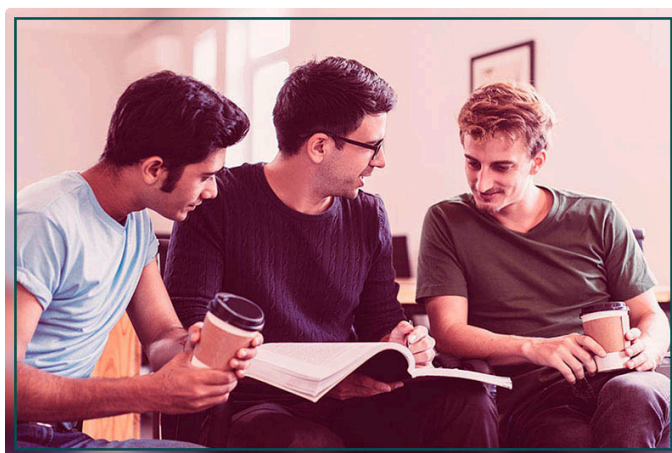
O trecho em linguagem C para calcular, mostrando as médias de cada prova, aluno e turma.

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. Veja o que acontece a seguir:



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



Cada aluno está representado no mesmo índice dos 3 vetores.

As posições de cada vetor ficam da seguinte forma:

- 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:

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

Vetor prova 1

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

Vetor prova 2

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

Vetor prova 3

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

C



Média de cada prova

Usamos todas as posições de cada vetor.

Prova 1

7.00, 8.50, 9.35, 8.45 ... 9.00, 7.90, 9.60.

Prova 2

9.70, 9.20, 8.95, 9.00 ... 10.00, 6.40, 4.30.

Prova 3

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:

C



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

Usamos a mesma posição dos 3 vetores. Dessa maneira, as notas de cada aluno são:

Aluno 1

7.00 9.70 10.00 → média = 8.90.

Aluno 2

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:

C



Média da turma

Pode ser calculada de 2 formas:

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:

C



Pela média dos alunos

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

C



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 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;
```

E

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

Parabéns! 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.

Questão 2

Considere um vetor de 15 elementos do tipo caracter, chamado vogais. Você precisa que o programa que manipula esse vetor contabilize a quantidade de vogais "A" ou "E" que nele estão armazenadas e guarde 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;
}
```
- E
- ```
for (ind=0;ind<=14;ind++)
{
    if (vogais[ind]=='a' and vogais[ind]=='e')
        cont+=1;
}
```

Parabéns! 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.



2 - Matriz para armazenamento de dados

Ao final deste módulo, você será capaz de empregar a matriz para armazenamento de dados em um programa.

Matrizes

Neste vídeo, vamos explorar as matrizes em C e aprender como manipular eficientemente dados multidimensionais. Aprenderemos sobre a declaração, inicialização e manipulação de matrizes, incluindo operações como acesso a elementos, atribuição, cálculos e iteração.

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



Comparação com o 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:

Matriz		Vetor
Na matriz, o acesso a cada elemento ocorre pelo uso de dois índices.	X	No vetor, o elemento é acessado pelo índice, que representa a posição relativa desse elemento no vetor.

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:

C

Vamos observar como que ficam as cadeias dos três vetores a seguir:

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

Cadeia do vetor prova 1.

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

Cadeia do vetor prova 2.

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

Cadeia do vetor prova 3.

Com essa 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.

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.

A matriz é a estrutura de dados ideal para esse tipo de situação.

Em vez de N vetores, declaramos e usamos apenas 1 matriz. É como se juntássemos os N vetores – no caso exemplificado, os 3 vetores –, como na imagem a seguir:

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

Representação dos três vetores.

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. Veja o resultado:

	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

Representação dos três vetores.

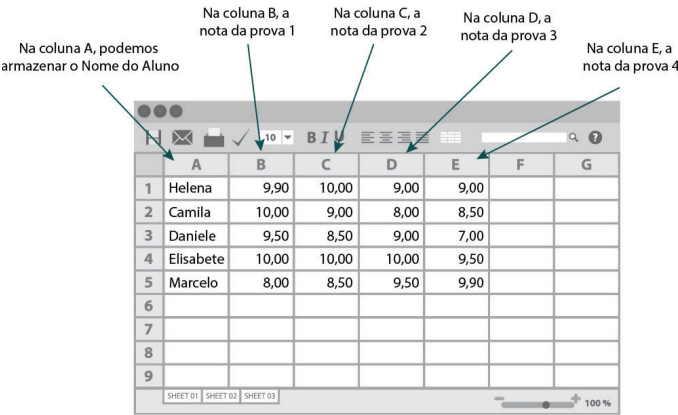
Conceito de matriz

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).

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

Vamos ver um exemplo de planilha Excel com diversos tipos de dados, em que o nome do aluno usa um tipo de dado, e as notas das provas usam outro:



Planilha Excel com diversos tipos de dado.

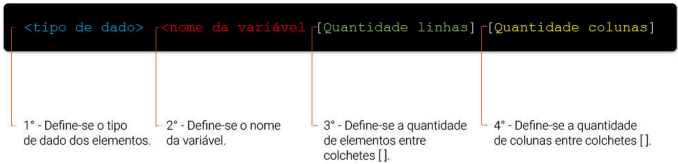
Agora um exemplo de planilha matriz com dados homogêneos, em que podemos armazenar somente dados de um único tipo:

	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

Planilha matriz com dados homogêneos.

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:



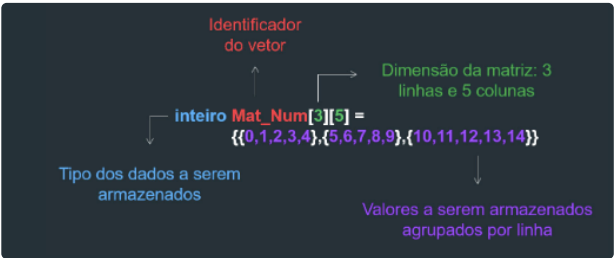
Declarações em Portugol Studio e linguagem C das matrizes.

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					

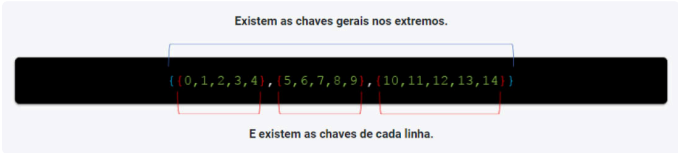
Mat_Num

Representação da Mat_Num.



Representação da Mat_Num.

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



Definição entre chaves do conjunto de elementos.

A tabela a seguir 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

Tabela: Matriz preenchida após a inicialização.

A imagem a seguir 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

Inicialização da matriz.

Vamos ver outros exemplos de declaração de matrizes nos seguintes códigos:

Portugol



C



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



C



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.

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



C



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

Tabela: Resultado da matriz preenchida.

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.

Comando para exibir um elemento da matriz

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

Portugol



C



Comando para atribuir valor a um elemento da matriz

O comando armazena o valor 90 no elemento da linha 2 e coluna 4. O elemento, anterior ao comando, é 15, que é substituído pelo 90, como no código a seguir

Portugol



C



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

Tabela: Representação da matriz.

Comando para ler um dado do dispositivo de entrada e armazenar na matriz

Há duas formas de ler um dado de dispositivo de entrada:

Leitura do dado de entrada direto para uma posição da matriz

O comando 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, como no código a seguir:

Portugol

C

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

Tabela: Representação da matriz.

Leitura do dado do dispositivo de entrada em uma variável

Nesse caso, atribui-se o conteúdo dessa variável a uma posição da matriz. O comando 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, como no código a seguir:

Portugol

C

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

Tabela: Representação da matriz.

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

Portugol



C



Demonstração do uso de matrizes

Exemplo 1

O comando declara uma matriz para armazenar 2 notas de 30 alunos de uma turma, como nos códigos a seguir:

Portugol



C



Exemplo 2

O comando declara uma matriz para armazenar o sexo – masculino (M) ou feminino (F) – de 50 alunos de 3 turmas, como nos códigos a seguir

Portugol

C

Exemplo 3

O comando declara uma matriz para armazenar 5 apostas de 10 alunos de uma turma e inicializa cada aposta com 0 (zero), como nos códigos a seguir:

Portugol

C

Exemplo 4

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.

Como inicializar cada elemento da matriz com 0 (zero)?

Nesses casos, 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, nesse caso, é o PARA (FOR), pois a repetição é para um número fixo e conhecido de vezes – no caso 3 linhas x 100 colunas. Veja:

Portugol



C



Exemplo 5

O trecho de código exibe no dispositivo de saída as 3 notas dos 100 alunos da turma, como nos códigos a seguir:

Portugol



C



Exemplo 6

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), como nos códigos a seguir:

Portugol



C



Vamos praticar

Vamos pôr em prática o que aprendemos no decorrer deste conteúdo. Sugerimos que você realize os seguintes procedimentos:

- Desenvolva o algoritmo em Portugol Studio e na linguagem C;
- Garanta que sua solução funcione, executando algumas vezes, com dados distintos;
- Veja a solução que sugerimos e compare com a sua.

Prática 1

Faça um algoritmo, no emulador a seguir, que leia dados inteiros e armazene-os em uma matriz 3x4. Em seguida, mostre a quantidade de números pares e ímpares armazenados na matriz.

Atividade 6

 TUTORIAL  COPIAR

C

null

null



Temos a seguinte resolução:

Portugol



C



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

```
1 #include<stdio.h>
2 int main()
3 {
4     int mat[4][4],lin,col,contpar=0,contimpar=0;
5     printf("Digite valor para os elementos da matriz\n\n");
6     for (lin=0; lin<4; lin++)
7     for (col=0; col<4; col++)
8     {
9         printf("Elemento[%d][%d] = ", lin, col);
10        scanf("%d",&mat[lin][col]);
11    }
12    printf("\n\n***** Saída de Dados ***** \n\n");
13    for (lin=0; lin<4; 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("\n\nPares: %d ",contpar);
22    printf("\n\nImpares: %d ",contimpar);
23
24    return(0);
25 }
```

Código em linguagem C.

```
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

***** Saída de Dados *****

Pares: 6
Impares: 6
```

Execução do programa.

Prática 2

Faça, no emulador a seguir, 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 = colona: [0][0], [1][1], [2][2], [3][3].

Atividade 7

 TUTORIAL  COPIAR

C

null

null



Resolução

Neste vídeo, a prática 2 será resolvida.

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



Além disso, para a solução em C, abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

C

Prática 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

Matriz 1 e 2 geradas.

A seguir o emulador para execução:

Atividade 8

C

TUTORIAL

COPIAR

null

null



Resolução

Neste vídeo, a prática 3 será resolvida.

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

Além disso, para a solução em C, abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

C



Prática 4

Faça, no emulador a seguir, 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.

C

null

null



Resolução

Neste vídeo, a prática 4 será resolvida.

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

Além disso, para solução em C, abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

C



C



Prática 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

Matrix 5x5.

A seguir, o emulador para execução:

Atividade 10

 TUTORIAL  COPIAR

C

null



null



Resolução

Neste vídeo, a prática 5 será resolvida.

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



Além disso, para a solução em C, abra a sua ferramenta de desenvolvimento em C (DEV++ ou ambiente online), limpe a tela e cole o programa:

C



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 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)
}
```

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

Parabéns! 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.

E) A variável maior igual a 0 é inicializado dentro do segundo FOR, fará com que todo elemento lido, da respectiva coluna, seja sempre o maior.

Questão 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]); }  
  
E for (ind=1;ind==3;ind++)  
{ printf ("%f : ",mat[i][i]); }
```

Parabéns! 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.

E) acessará um único elemento da diagonal principal da matriz.

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.



Ouçã este podcast sobre os principais assuntos abordados no tema Vetores e Matrizes.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.



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.

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.



Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

 Download material

O que você achou do conteúdo?



 Relatar problema