MATRIZES

1- Matrizes - Itinerário rodoviário

Exemplo:

Faça um programa que receba o nome de cinco cidades do itinerário da empresa e suas respectivas distâncias entre si, em km, e armazene esses dados estruturados em uma tabela. Também armazena um valor de consumo médio de combustível de um veículo dessa empresa, em km/L. Em seguida, exibe um relatório contendo o nome das cidades cujo percurso ultrapasse 250 km além da combinação de todos os percursos possíveis juntamente com a quantidade de combustível necessária para percorrê-los.

Entradas:

1. um vetor com o nome de cinco cidades diferentes;
2. uma matriz 5 x 5 com a distância entre as cidades, sendo que na diagonal principal deve ser colocada automaticamente distância zero, ou seja, não deve ser permitida a digitação;
3. o consumo de combustível de um veículo, ou seja, quantos quilômetros este veículo percorre com um litro de combustível.

Saídas:

1. os percursos que não ultrapassam 250 quilômetros (os percursos são compostos pelos nomes das cidades de origem e pelos nomes das cidades de destino);
2. todo os percursos (nome da cidade de origem e nome da cidade de destino), juntamente com a quantidade de combustível necessária para o veículo percorrê-los;

Exemplos de Entradas e Saídas:

Entradas:

Lavras Betim Guarulhos Uberaba Brasilia
208 363 475 826 208 541 449 744 363 541 497 1021 475 449 497 526 826 744 1021 526
13.5

Saídas:

Lavras Betim
Betim Lavras
Lavras Betim 15.4074
Lavras Guarulhos 26.8889
Lavras Uberaba 35.1852
Lavras Brasilia 61.1852
Betim Lavras 15.4074

Betim Guarulhos 40.0741
Betim Uberaba 33.2593
Betim Brasilia 55.1111
Guarulhos Lavras 26.8889
Guarulhos Betim 40.0741
Guarulhos Uberaba 36.8148
Guarulhos Brasilia 75.6296
Uberaba Lavras 35.1852
Uberaba Betim 33.2593
Uberaba Guarulhos 36.8148
Uberaba Brasilia 38.963
Brasilia Lavras 61.1852
Brasilia Betim 55.1111
Brasilia Guarulhos 75.6296
Brasilia Uberaba 38.963

2- SUGESTÃO DE SOLUÇÃO:

Ideia geral:

Para resolver esse problema devemos pensar nos dois tipos de variáveis com a qual iremos trabalhar. Iremos trabalhar com um identificador chave que vai ser a cidade e uma variável que irá relacionar com duas cidades ao mesmo tempo que é a distância. Então para a solução desse problema, deveremos criar uma espécie de tabela, com cidades e distâncias, como no exemplo a seguir:

	cidade 1	cidade 2	cidade 3	cidade 4	cidade 5
cidade 1	distância 1 e 1	distância 1 e 2	distância 1 e 3	distância 1 e 4	distância 1 e 5
cidade 2	distância 2 e 1	distância 2 e 2	distância 2 e 3	distância 2 e 4	distância 2 e 5
cidade 3	distância 3 e 1	distância 3 e 2	distância 3 e 3	distância 3 e 4	distância 3 e 5
cidade 4	distância 4 e 1	distância 4 e 2	distância 4 e 3	distância 4 e 4	distância 4 e 5
cidade 5	distância 5 e 1	distância 5 e 2	distância 5 e 3	distância 5 e 4	distância 5 e 5

Passo 1: Armazenamento dos dados de entrada: Temos 3 dados para podermos ler, o vetor tipo string com o nome de cinco cidades diferentes, uma matriz 5x5 tipo int com a distância entre as cidades e o consumo de combustível que será do tipo float.

como no enunciado já deixa explícito que o programa sempre executará com 5 variáveis, nos declaramos no código um vetor de tamanho 5 e então lemos os valores deste vetor. Para isso usaremos a estrutura de repetição for, terá como parâmetro um inteiro inicializado em 0, uma condição de parada enquanto esse inteiro for menor que 5 e o incremento de 1 em 1.

```
string cidades[5];
for (int i = 0; i < 5; i++) {
    cin >> cidades[i];
}
```

No caso da matriz de distâncias, como foi passado no enunciado, ela será uma matriz de 5x5. Mas afinal, o que são matrizes? Matriz é um vetor de duas dimensões, ou seja, possui altura e largura, no C++ ela é representada por dois pares de colchetes `[][]`, onde o primeiro par de colchetes determina a quantidade de linhas e o segundo par de colchetes determina a quantidade de colunas: `[linhas][colunas]`. Então uma matriz 5x5 teria 5 linhas e cinco colunas, sendo representada como `[5][5]`. Quando uma matriz possui a mesma quantidade de linhas e colunas, nós a chamamos de matriz quadrada, e uma matriz quadrada possui uma propriedade muito interessante, que é a diagonal principal, essa diagonal são os elementos que possuem a posição linha e colunas iguais, como nessa representação gráfica de uma matriz 5x5:

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]
[4][0]	[4][1]	[4][2]	[4][3]	[4][4]

No problema, determina que a diagonal principal deverá receber valor 0 e os demais valores serão digitados pelo usuário. Então para podermos fazer essas operações com a matriz, precisaremos de uma estrutura de repetição para percorrer suas linhas e dentro dessa, uma outra estrutura de repetição para percorrer as colunas.

```
int distancia[5][5];
for ( int linha = 0; linha < 5; linha++ ) {
      for ( int coluna = 0; coluna < 5; coluna++ ) {
```

```
            if ( linha == coluna ) {            // Se a posição da linha for a mesma que a coluna
                distancia[linha][coluna] = 0; // Atribui valor 0 a diagonal principal
            }
            else {                                  // Se for diferente
                cin >> distancia[linha][coluna];  // Lê o valor digitado pelo usuário
            }
        }
}
```

Agora que nós lemos o vetor e a matriz, seguindo o exemplo de entrada, nossa tabela estará dessa forma

	Lavras	Betim	Guarulhos	Uberaba	Brasilia
Lavras	0	208	363	475	826
Betim	208	0	541	449	744
Guarulhos	363	541	0	497	1021
Uberaba	475	449	497	0	526
Brasilia	826	744	1021	526	0

Então para finalizar as entradas nos lemos o valor do consumo:

```
float consumo;
cin >> consumo;
```

Passo 2: Cálculo realizado no programa: os percursos menores que 250km: Para conferirmos qual são os percursos menores que 250 km, teremos que percorrer a matriz de distância, comparando se o valor é menor que 250 km e diferente de 0.

```
for ( int linha = 0; linha < 5; linha++ ) {
  for ( int coluna = 0; coluna < 5; coluna++ ) {
     if ( distancia[linha][coluna] < 250 && distancia[linha][coluna] != 0 ) {
       cout << cidades[linha] << " " << cidades[coluna] << endl;
     }
  }
}
```

Passo 3: Cálculo realizado no programa: combustível. Por fim, deve-se calcular o gasto que de combustível que o veículo vai ter em cada distância. Para isso, iremos criar uma variável do tipo float chamada combustivel. E iremos percorrer a matriz com uma estrutura de repetição, caso a distancia[linha][coluna] for igual a 0, o programa não realizará o cálculo, caso for diferente, ele realizará o cálculo e imprimirá as cidades com o combustivel gasto.

```
float combustivel = 0;
for ( int linha = 0; linha < 5; linha++ ) {
   for ( int coluna = 0; coluna < 5; coluna++ ) {
      if ( distancia[linha][coluna] != 0 ) {
         combustivel = distancia[linha][coluna] / consumo;
         cout << cidades[linha] << " " << cidades[coluna] << " " << combustivel << endl;
      }
   }
}
```

Exemplo de código:

```
#include <iostream>
using namespace std;

int main() {
   string cidades[5];
   int distancia[5][5];
   float consumo;
   for ( int i = 0; i < 5; i++ ) {
      cin >> cidades[i];
   }
   for ( int linha = 0; linha < 5; linha++ ) {
      for ( int coluna = 0; coluna < 5; coluna++ ) {
         if ( linha == coluna ) {
            distancia[linha][coluna] = 0;
         }
         else {
            cin >> distancia[linha][coluna];
```

```
            }
        }
    }
    cin >> consumo;
    for ( int linha = 0; linha < 5; linha++ ) {
        for ( int coluna = 0; coluna < 5; coluna++ ) {
            if ( distancia[linha][coluna] < 250 && distancia[linha][coluna] != 0 ) {
                cout << cidades[linha] << " " << cidades[coluna] << endl;
            }
        }
    }
    float combustivel = 0;
    for ( int linha = 0; linha < 5; linha++ ) {
        for ( int coluna = 0; coluna < 5; coluna++ ) {
            if ( distancia[linha][coluna] != 0 ) {
                combustivel = distancia[linha][coluna] / consumo;
                cout << cidades[linha] << " " << cidades[coluna] << " " << combustivel << endl;
            }
        }
    }
    return 0;
}
```

Passo 5: Teste do programa:

TESTE DE MESA - Clique aqui!

MAIS MATERIAL DE APOIO:

GitHub

Replit