T1 ALG 2

João Victor Balhego e Kauan Dalfovo

```
// Incluindo as bibliotecas que vamos utilizar.
#include <iostream>
#include <cctype>
#include <cstring>
#include <cstdlib>
using namespace std;
// definindo a contante TMAX como 20.
#define TMAX 20

// incluindo os protótipos das funções

// recebendo o valor de cidades que o usuario deseja
int lerNumInteiroIntervalo(int mini, int maxi);
// Validação do tipo do numero
bool validarDigito(string valor);
// Leitura dos nomes das cidades
void lerVetDeString(int n, string VetCidades[]);
// Apresentação dos nomes das cidades
void relatorioDeNomes(int n, string VetCidades[]);
// Leitura das distancias entre cidades
void informarDistancia(int n, int matriz[][TMAX]);
// Apresentação da distancias informadas
void relatorioMat(int n, int matriz[][TMAX]);
// pegando posição 1 para calcular a distancia
void lerPosicao1(int n, int &pos1);
// pegando posição 2 para calcular a distancia
void lerPosicao2(int n, int &pos2);
// Calcula a distancia de de duas cidades de acordo com a matriz
```

```
int distEntreCidades(int cidadeA, int cidadeB, int matriz[][TMAX]);
// Calcula o percurso de 2 ou mais cidades
void calcularPercurso(int n, string VetCidades[], int mat[][TMAX]);
// Apresenta o menu
void menuCidades();
// Retorna o item do Menu
int itemDoMenu();
// função switch do menu
void Switch(int item, int n, string VetCidades[], int &pos1, int &pos2, int mat[][TMAX]);

int main()
{
   // Vamos definir algumas das variáveis que iremos utilizar durante o código
   int n = 0, mat[TMAX][TMAX] = {0};
   int pos1 = 0, pos2 = 0, item = 0;
   string VetCidades[TMAX];
   char resposta;
   n = lerNumInteiroIntervalo(4, TMAX);

   /*linha e coluna recebem a variavél que define
    a quantidade de cidades, para percorrer a matriz*/

   lerVetDeString(n, VetCidades);
   cout << endl;
   relatorioDeNomes(n, VetCidades);
   cout << endl;
   informarDistancia(n, mat);
   cout << endl;
   relatorioMat(n, mat);
   cout << endl;
```

```
   // Um laço de repetição para utilizarmos o menu até o usuario responder NAO à pergunta do final.
   do
   {
      menuCidades();
      cout << endl;
      item = itemDoMenu();
      cout << endl;
      Switch(item, n, VetCidades, pos1, pos2, mat);
      cout << endl;

      if (item == 0)
      {
         break;
      }

      // Laço de repetição para perguntar se deseja ultilizar o menu mais uma vez
      do
      {
         cout << "Você deseja repetir e voltar para o menu? (S/N): ";
         cin >> resposta;
         resposta = toupper(resposta);
      } while (resposta != 'S' && resposta != 'N');

      cin.ignore();

      if (resposta == 'S')
      {
         // Comando para limpar o terminal

         #ifdef _WIN32
```

```
            system("cls"); // Windows
        #else
            system("clear"); // Linux
        #endif
    }

  } while (resposta == 'S');

  if (resposta == 'N')
  {
    cout << "encerrando o programa.";
  }

  return 0;
}

int lerNumInteiroIntervalo(int mini, int maxi)
{
  // Função para ler quantas cidades teremos no mapa.
  int n;
  string num;
  do
  {
    while (num == "")
    {
      cout << "Informe quantas cidades deseja, entre " << mini << " e " << maxi << ": ";
      getline(cin, num);
    }
    if (validarDigito(num))
    {
      n = stoi(num);
```

```
        }
        else
        {
            cout << "informe um valor valido! " << endl
                 << endl;
        }
    } while (n < mini || n > maxi);
    return n;
}

void lerVetDeString(int n, string VetCidades[])
{
    /*Entrada de dados(nomes) das cidades, caso presença de números no nome da cidade
     devera ser representado em números romanos!*/

    string cidade;
    for (int i = 0; i < n; i++)
    {
        do
        {
            cout << "Informe o nome da cidade N° " << i + 1 << ": ";
            getline(cin, cidade);
        } while (cidade[0] == ' ' || !isalpha(cidade[0]) || !isupper(cidade[0]));
        VetCidades[i] = cidade;
    }
}

void relatorioDeNomes(int n, string VetCidades[])
{
    cout << "-------------------------" << endl;
    cout << "  RELATORIO DAS CIDADES" << endl;
```

```
    cout << "-------------------------" << endl;
    for (int i = 0; i < n; i++)
    {
        cout << i + 1 << " - " << VetCidades[i] << endl;
    }
}

void informarDistancia(int n, int mat[][20])
{

    // Leitura das distâncias das cidades
    cout << "Não existe distância da cidade com a própria." << endl
         << endl;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
            {
                // A distância de uma cidade para ela mesma é 0
                mat[i][j] = 0;
            }
            else
            {
                string entrada;
                int distancia;
                do
                {
                    if (i < j)
                    {
```

```
                cout << "Informe a distância em km entre cidade (" << i + 1 << "-" << j + 1 << "): ";
                cin >> entrada;
                cin.ignore();

                if (validarDigito(entrada)) // Verifica se a entrada é um digito válido
                {

                    distancia = stoi(entrada);

                    mat[i][j] = distancia;
                    mat[j][i] = mat[i][j];
                }
                else
                {
                    cout << "Entrada inválida, por favor digite um número válido." << endl;
                }
            }
        } while (!validarDigito(entrada) || distancia < 0);
    }
  }
}
void relatorioMat(int n, int mat[][20])
{
   cout << "-------------------------" << endl;
   cout << "RELATORIO DAS DISTÂNCIAS" << endl;
   cout << "-------------------------" << endl;
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {
```

```
            if (i == j)
            {
               cout << "[ ]";
            }
            else
               cout << "[" << mat[i][j] << "]";
         }
         cout << endl;
      }
   }

int distEntreCidades(int cidadeA, int cidadeB, int matriz[][TMAX])
{
   // Recebe o valor da matriz de distancia e armazena em outra varíavel retornavel

   int distancia;
   distancia = matriz[cidadeA][cidadeB];
   if (distancia == 0)
   {
      cout << "Essas cidades nao tem nenhuma ligacao direta. ";
      return 0;
   }
   else if (cidadeA == cidadeB)
   {
      cout << "A distancia eh zero pois a mesma cidade foi passada duas vezes";
   }
   else
   {
      return distancia;
   }
}
```

```
void menuCidades()
{
   cout << "                MENU                " << endl;
   cout << "----------------------------------------" << endl;
   cout << "1 - Relatorio com o nome das cidades     " << endl;
   cout << "2 - Relatorio com a matriz das distâncias" << endl;
   cout << "3 - Calcular distancia entre duas cidades" << endl;
   cout << "4 - Calcular Percusrso de Cidades        " << endl;
}

void lerPosicao1(int n, int &pos1)
{
   cout << endl;
   do
   {
      cout << "informe a primeira cidade para calcular a distancia: ";
      cin >> pos1;
   } while (pos1 > n || pos1 < 0);
}
void lerPosicao2(int n, int &pos2)
{
   do
   {
      cout << "informe a cidade destino para calcular a distancia: ";
      cin >> pos2;
   } while (pos2 > n || pos2 < 0);
}
int itemDoMenu()
{
   int item = 0;
```

```
   do
   {
      cout << "Selecione um item do menu: ";
      cin >> item;
   } while (item < 0 || item > 4);
   return item;
}

void Switch(int item, int n, string VetCidades[], int &pos1, int &pos2, int mat[][TMAX])
{
   switch (item)
   {
   case 1:
      relatorioDeNomes(n, VetCidades);
      break;

   case 2:
      relatorioMat(n, mat);
      break;

   case 3:
      relatorioDeNomes(n, VetCidades);
      lerPosicao1(n, pos1);
      lerPosicao2(n, pos2);
      cout << "A distancia entre " << VetCidades[pos1 - 1] << " e " << VetCidades[pos2 - 1] << " e
de:  " << distEntreCidades(pos1 - 1, pos2 - 1, mat) << "km." << endl;
      break;

   case 4:
      // calcularPercurso(n, VetCidades, mat);
      relatorioDeNomes(n, VetCidades);
```

```
        relatorioMat(n, mat);
        calcularPercurso(n, VetCidades, mat);
        break;

    default:
        cout << "Opção inválida." << std::endl;
        break;
    }
}

void calcularPercurso(int n, string VetCidades[], int mat[][TMAX])
{
    /* Calculo de percurso das cidades informadas pelo usuario,
    fazendo a validação se as cidades não tem ligação
    e informando se for o caso*/

    int cidadesPercurso = 0;
    cout << "Informe o número de cidades no percurso: ";
    cin >> cidadesPercurso;
    cin.ignore();

    if (cidadesPercurso < 2 || cidadesPercurso > n)
    {
        cout << "Número inválido de cidades para o percurso!" << endl;
        return;
    }

    int percurso[cidadesPercurso];

    for (int i = 0; i < cidadesPercurso; i++)
    {
```

```
        string cidade;
        bool achou = false;

        while (!achou)
        {
            cout << "Informe o nome da cidade " << i + 1 << ": ";
            getline(cin, cidade);

            for (int j = 0; j < n; j++)
            {
                if (VetCidades[j] == cidade)
                {
                    percurso[i] = j;
                    achou = true;
                    break;
                }
            }

            if (!achou)
            {
                cout << "Cidade " << cidade << " não encontrada no mapa, informe novamente." << endl;
            }
        }
    }

    int distTotal = 0;
    for (int i = 0; i < cidadesPercurso - 1; i++)
    {
        int dist = mat[percurso[i]][percurso[i + 1]];
        if (dist == 0)
```

```
      {
         cout << "Percurso inválido, cidades " << VetCidades[percurso[i]] << " e " <<
VetCidades[percurso[i + 1]] << " não se conectam!" << endl;
         return;
      }
      distTotal += dist;
   }

   cout << "A distância total do percurso é: " << distTotal << " km." << endl;
}

bool validarDigito(string valor)
{
   /* Validaçao do número recebendo ele por string e tranformando em char,
      passando indice por indice da string verificando se é um digito
      e retornando uma variavel booleana*/

   char c;
   for (int i = 0; i < valor.size(); i++)
   {
      c = valor[i];

      if (!isdigit(c))
      {
         return false;
      }
   }
   return true;
}
```