Apêndice
Lista de Exercícios

"Só se aprende a programar, programando"

- **Anônimo** -

1 Fundamentos

1.1-Desenvolva um programa em C, para ler três numeros inteiros diferentes e determinar o maior e o menor. Utilize apenas o comando condicional sem condições compostas.

1.2-Desenvolva um programa para ler um número real x e um número inteiro n. Calcule x^n , utilizando apenas a operação de multiplicação.

1.3-Desenvolva um programa para calcular a distancia que um elevador percorreu. O seu programa deve ler uma distancia em metros entre um andar, em seguida, ler um número n > 0 que representa o número de viagens feitas por um elevador. Depois, ler uma sequencia de n + 1 números inteiros positivos que representam o início e fim de cada viagem. Por exemplo, para a sequencia 4, 7, 0, 7, 5, 3, 6, 2, 4, 0, temos: 4 = distância em metros entre um andar; 7= número de viagens feitas; 0, 7, 5, 3, 6, 2, 4, 0 = viagens feitas, que mostra que o elevador saiu do rés-do-chão para o 7º, depois do 7º para o 5º, depois do 5º para o 3º, e assim por diante.

1.4-Desenvolva um programa para ler duas datas no formato dd/mm/aaaa, que fazem referencia a data de nascimento de duas pessoas. Vamos supor que essas datas serão armazenadas em seis variáveis do tipo inteiro: dia1, mes1, ano1, dia2, mes2 e ano2. Mostre na tela as seguintes mensagens: têm a mesma idade; a pessoa da data 1 é a mais velha ou a pessoa da data 2 é a mais velha.

1.5-Dado um número inteiro n e quatro números inteiros a, b, c e d que representam as extremidades dos intervalos [a, b] e [c, d], supondo que a < b, c < d e a < c. Determinar se n pertence sómente ao intervalo [a, b] ou sómente ao intervalo [c, d] ou pertence a ambos, ou não pertence a nenhum dos dois. Em cada caso, imprimir uma mensagem adequada.

1.6-Desenvolva um programa para ler um número inteiro positivo n, e a seguir, as n notas dos estudantes de uma turma. Determinar as duas melhores notas, e mostrar na tela, os seus valores.

1.7- Dada uma sequência com n números inteiros positivos, terminados por zero. Desenvolva um programa para determinar quantos segmentos de números iguais consecutivos compõem essa sequência. Por exemplo: a seguinte sequência é formada por cinco segmentos de números iguais: 7, 5, 5, 3, 8, 8, 8, 8, 2, 2, 0

1.8- Dois números inteiros positivos são amigos entre si quando a soma dos divisores próprios do primeiro é igual ao segundo e quando a soma dos divisores próprios do segundo é igual ao primeiro. Desenvolva um programa para ler dois números inteiros positivos e verificar se eles são amigos. Por exemplo: 220 tem como divisores próprios: 1,2,4,5,10,11,20,44,55,110 cuja soma é 2284. Por outro lado, 2284 tem como divisores próprios: 1,2,4,71,142, cuja soma é 220. Estes números são amigos.

1.9- Um número inteiro positivo é chamado de alternante, se a sequencia de dígitos que o forma alterna entre valores pares e valores ímpares. Por exemplo, são números alternates: 32, 85, 125, 432587,

1.10-Desenvolva um programa para ler um conjunto com n números inteiros positivos terminados por um número negativo. Mostrar na tela, os números que são maiores do que os seus vizinhos. Por exemplo, para o conjunto {8, 2, 4, 1, 6, 12, 5, 9} os números 4 e 12 satisfazem essa propriedade.

1.11-O programa que descrevemos a seguir, calcula o produto de dois números naturas, através de operações de adição e subtracção. Faça uma simulação passo-a-passo para x = 7 e Y = 8.

```
#include <stdio.h>
#include <stdlib.h>
int main()
 {
    int x, y, u, z;
    printf ("\n Entre com um o valor de x: ");
    scanf ("%d", &x);
    printf ("\n Entre com um o valor de y: ");
    scanf ("%d", &y);
    z = 0;
    u = x;
    while ( u != 0)
     {
       z += y;
       u--;
     }
    printf ("\n Produto de %d  com %r = %d", x, y ,z);
    system ("PAUSE");
```

```
    return 0;
}
```

2 Funções

2.1-Desenvolva uma função que recebe como parâmetro um número real, e retorna o valor desse número arredondado com base na seguinte tabela. Se casa decimal estiver entre 0.0 a 0.24 arredondar para 0.25; entre 0.25 a 0.54 arredondar para 0. 55; entre 0.56 a 0.74 arredondar para 0.75; entre 0.75 a 0.99 arredondar 1.0.

2.2- Dada a fórmula François Viète (1540-1603)

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2+\sqrt{2}}}{2} \cdot \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} \cdots$$

Desenvolva um programa com funções para determinar o valor aproximado de π. Considere que o processo de cálculo deve terminar quando

$$| x_n - \pi | \leq Erro.$$

onde

$\pi = 3.14159265358979323846$

e

$Erro = 1.0e\text{-}18$

2.3-Cada termo da sequência de Tribonacci é dado pela soma dos três termos anteriores. Supondo que a sequência inicia com os números $T_1 = 1$, $T_2 = 1$ e $T_3 = 2$, os 8 restantes termos são os números inteiros: 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, …. Desenvolva uma função que recebe como parâmetro um número inteiro positivo n > 0 e os três termos iniciais de Tribonacci. Calcule e retorne o valor de Tribonacci.

2.4- Desenvolva uma função que recebe como parâmetro o ano, e retorna 1 se o ano for bissexto e 0 no caso contrário. No calendário Gregoriano os anos bissextos têm 366 dias em vez dos normais 365. As regras para determinar se um ano é bissexto são: São bissextos todos os anos múltiplos de 400 (por exemplo: 1600, 2000, 2400, 2800, . . .); São bissextos todos os múltiplos de 4, exceto se forem múltiplos de 100, mas não de 400 (por exemplo: 1996, 2000, 2004, 2008, 2012, 2016, . . .); Não são bissextos todos os demais anos.

2.5-Desenvolva uma função que recebe como parâmetro, um número inteiro não negativo n, e calcula o superfactorial desse número. Por definição, o superfactorial de um número inteiro não negativo n é definido por:

$sf(n) = \prod_{i=1}^{n} i! = 1! \times 2! \times 3! \times \ldots \times n!$

2.6- Desenvolva uma função que recebe como parâmetros, dois números intiros x e y. Retorne a divisão de x por y, utilizando apenas operações de divisão.

2.7- Desenvolva uma função que recebe como parâmetros, o valor de um ângulo em graus e um número inteiro positivo k. Calcule o valor do seno hiperbólico desse ângulo, utilizando para o efeito a série de Taylor.

$$\sinh x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots.$$

2.8-Desenvolva uma função que recebe como parâmetro um número inteiro positivo, e retorne 1 se esse número for binário e 0 no caso contrário. Por exemplo: para 11001 retornar 1, enquanto que, para 1021 retornar zero.

3 Indução Matemática

3.1-Prove que $2^n > 2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2^0$, Para n > 1.

3.2- Prove que $1^3 + 2^2 + 2^3 + \ldots.. + 2^n = 2^{n+1} - 1$ para todo n≥ 0.

3.3- Prove que $1^3 + 3^3 + 5^3 + \ldots.. + (2n-1)^3 = 2n^4 - n^2$ para todo n≥ 1.

3.4- Prove que $1^3 + 2^2 + 2^3 + \ldots.. + 2^n = 2^{n+1} - 1$ para todo n≥ 0.

3.5- Prove que $2^n > n^2$, para todo n ≥ 4.

3.6-Determine a relação de recorrência que expressa a seguinte sequencia:

2, 4, 8, 16, …

3.7-Determine a relação de recorrência que expressa a seguinte série:

1 + 1/2 + 1/3 + ... + 1/n

3.8-Determine a relação de recorrência que expressa a seguinte sequência:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

3.9- Escreva os cinco primeiros termos da fórmula de recorrência.

D(1) = 2
D(2) = 5
D(n) = (n-1)xD(n-1) + (n-2)xD(n-2) para todo n > 2

3.10- Escreva os sete primeiros termos da fórmula de recorrência.

S(1) = 1
S(n) = S(n-1) + 1 para todo n ≥ 2

3.11- Escreva os sete primeiros termos da fórmula de recorrência.

T(1) = 1

T(n) = T(n-1) + 3 para todo n ≥ 2

4 Recursão Linear

4.1-Desenvolva uma função recursiva que recebe como parâmetro um número inteiro positivo n, e calcula o valor da série Harmônica:

$H_n = 1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + \ldots + 1/n$

4.2-Dada a seguinte soma:

0 + 1 + 2 + 3 + 4 + ... + n

Desenvolva uma função recursiva para calcular os n primeiros termos.

4.3-Desenvolva uma função recursiva que recebe como parâmetro um número inteiro positivo, e retorna o valor dos n primeiros termos da soma:

$2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^n$

4.4-Desenvolva uma função recursiva que recebe um número inteiro positivo e retorna esse número na ordem inversa. Por exemplo, para n = 123, a sua função deve mostrar o número 321.

4.5-Desenvolva uma função recursiva para calcular o máximo divisor comum de dois números inteiros passados como parâmetros, através da seguinte relação de recorrência.

$$\text{mdc}(x, y) = \begin{cases} 0 & se\ x = y \\ x & se\ x < y \\ mdc\,(x-y, x) & se\ x > y \end{cases}$$

mdc (10,6) = mdc (4,6) = mdc (6,4) = mdc (2,4) = mdc (4,2) = mdc (2,2) = 2

4.6-Desenvolva uma função recursiva que recebe como parâmetro um número inteiro positivo e devolve o número de dígitos desse número.

4.7-Faça a simulação da seguinte função recursiva. Suponha que n = 8

```
int func (int n )
 {
     if ( n == 0 )  return 0;
```

```
    else return n + func (n-1);
}
```

Utilize para o efeito o diagrama de linha e o diagrama de pilha.

4.8-Desenvolva uma função recursiva que recebe como parâmetro um número inteiro positivo e mostra o seguinte triangulo. Por exemplo, para n = 5 a sua função deve mostrar na tela a seguinte triangulo:

1
2 1
3 2 1
4 3 2 1
5 4 3 2 1

4.9- A função que descrevemos em seguida, calcula o mdc (m,n) pelo algoritmo de Euclides, para m e n são número inteiros estritamente positivos. Desenvolva uma função recursiva equivalente

```
int Euclides (int m , int n)
 {
   int r;
   do
     {
        r = m % n;
        m = n;
        n = r;
     }
   while (r != 0);
   return m;
 }
```

4.10- A razão áurea, denotada por Φ é um número real, com uma grande aplicação em várias áreas do conhecimento humano, cujo valor é determinado pela equação:

$$\Phi = \frac{1+\sqrt[2]{5}}{2} = 1,\ 618033988749894848204586834365638117720309 1\ldots$$

Mas podemos obter esse resultado com a fórmula:

$$\Phi = \sqrt[2]{1+\sqrt[2]{1+\sqrt[2]{1+\sqrt[2]{1+\sqrt[2]{1}}}}} + \ldots$$

Desenvolva uma função recursiva que entre outros parâmetros recebe um número inteiro positivo n e retorna o valor da razão áurea.

4.11- John McCarthy é um famoso cientista da computação (matemático). No seu trabalho, ele definiu uma função recursiva, denominada por f91, que recebe como parâmetro um número inteiro n, e retorna um número inteiro positivo definido por:

$$\text{f91(n)} = \begin{cases} n - 10 & se\ n > 100 \\ f91(f91(n + 11) & se\ n \leq 100 \end{cases}$$

5 Vectores Unidimensionais

5.1-Desenvolva uma função recursiva com a estratégia de decrementar para conquistar, que entre outros parâmetros, recebe um vector com elementos do tipo inteiro e o número de elementos inseridos. Retornar o número de elementos positivos.

5.2-Desenvolva uma função recursiva com a estratégia de decrementar para conquistar, que entre outros parâmetros, recebe um vector com elementos do tipo real e o número de elementos inseridos. Devolva esse vector com os seus elementos na ordem inversa, ou seja, troque o primeiro com o último, o segundo com o penúltimo e assim sucessivamente.

5.3-Desenvolva uma função iterativa que recebe como parâmetros um vector com elementos do tipo real e o número de elementos inseridos. Devolva o elemento máximo e o elemento mínimo.

5.4-Desenvolva uma função recursiva com a estratégia de decrementar para conquistar, que entre outros parâmetros recebe um vector com elementos do tipo real, uma determinada posição k, e o número de elementos inseridos. Remova o elemento que ocupa essa posição.

5.5-Desenvolva uma função iterativa que recebe como parâmetros um vector com elementos do tipo inteiro, o número de elementos inseridos e um número inteiro k. Retornar o elemento que mais se aproxima de k. O elemento que mais se aproxima de k é aquela cuja diferença modular é a menor.

5.6-Desenvolva uma função recursiva com a estratégia de decrementar para conquistar, que recebe como parâmetros dois vectores com elementos do tipo real com o mesmo número de elementos inseridos. Verifique se esses vectores são iguais. A sua função deve retornar 1 se for verdadeiro e 0 no caso contrário.

5.7-Desenvolva uma função iterativa que recebe como parâmetros, um vector com elementos do tipo real e o número de elementos inseridos. Verifique se os

elementos desse vector estão ordenados em ordem crescente. A sua função deve retornar 1 se for verdadeiro e 0 no caso contrário.

5.8-Desenvolva uma função iterativa que recebe como parâmetros um vector com números inteiros, e o número de elementos inseridos. Verifique se nesse vector os elementos de índice par contêm um conteúdo impár e vice-versa. A sua função deve retornar 1 se essa condição for verdadeira, e 0 no caso contrário.

5.9-Desenvolva uma função iterativa que recebe como parâmetros dois vectores com números inteiros, e o número de elementos inseridos em cada vector. Suponha que cada vector não possua elementos repetidos. Construa e devolva num terceiro vector, com os elementos comuns aos dois vectores sem repetições. Para além disso, retorne o número de elementos inseridos nesse vector.

5.10-Um número é chamado de capicua se a sequência de dígitos do número lidos da esquerda para a direita for igual a sequência de dígitos lidos da direita para a esquerda. Por exemplo: os seguintes números são capicuas: 123454321, 54445, 789987, 121. Desenvolva uma função recursiva com a estratégia de decrementar para conquistar que entre outros parâmetros recebe um vector com elementos do tipo inteiro e o número de elementos inseridos. Cada elemento desse vector contêm um e apenas um algarismo. Verificar se esse número é do tipo capicua.

5.11-Desenvolva uma função iterativa que recebe como parâmetro dois vectores com elementos do tipo real, os números de elementos inseridos em cada vector e determinado valor x. Separar esse vector em dois de tal modo que o primeiro é formado pelos elementos que vão do início até o valor x e o segundo pelos restantes.

5.12-Desenvolva uma função iterativa que recebe como parâmetros, um vector com elementos do tipo inteiro, e o número de elementos inseridos. Calcule e retorne o valor com maior frequência nesse vector. Se houver empate, considere a primeira ocorrência.

5.13-Desenvolva uma função iterativa que recebe como parâmetros, um vector com números inteiros, e o número de elementos inseridos. Verifique se existe pelo menos um número nesse vector maior do que os seus vizinhos.

5.14-Desenvolva uma função iterativa que recebe como parâmetros, dois vectores com números inteiros, e o número de elementos inseridos em cada vector. Construa e retorne num terceiro vector, com os elementos que pertencem a um vector mas não pertencem a outro. Para além disso, retorne o total de elementos inseridos nesse vector.

5.15-Desenvolva um subprograma que recebe como parâmetros, dois vectores com números reais, e o número de elementos inseridos em cada vector. Construa e devolva num terceiro vector, a junção desses vectores. Por definição, a junção do

vector A com o vector B é um vector C formado pelos elementos do vector A seguidos dos elementos do vector B. Para além disso, retorne o total de elementos inseridos nesse vector

.

5.16-Desenvolva uma função recursiva com a estratégia de decrementar para conquistar, que entre outros parâmeros, recebe dois vectores com números inteiros, e o número de elementos inseridos em cada vector. Verifique se esses vectores são disjuntos. A sua função deve retornar 1 se for verdadeiro e 0 no caso contrário.

5.7.17-Desenvolva uma função iterativa e a correspondente função recursiva com a estratégia de decrementar para conquistar, que entre outros parâmetros, recebe um vector com números inteiros, e o total de elementos inseridos. Devolva num segundo vector, a soma acumulada dos seus elementos. Por definição, seja X um vector com n elementos. A soma acumulada é um vector S, com o mesmo número de elementos, tais que, cada elemento s_i é representado pela soma de todos os elementos x_j para $0 \leq j \leq i$, ou seja:

$$s_i = \sum_{j=0}^{i} x_j$$

Por exemplo, para o vector X = {4, 3, 6, 1, 6, 9, 0, 3, 1, 7, 9, 2} a soma acumulada é o vector S = {4, 7, 13, 14, 20, 29, 29, 32, 33, 40, 49, 51}.

6 Cadeias de Caracteres

6.1-Desenvolva uma função recursiva, que entre outros parâmetros uma cadeia de caracteres e retorna o número de vogais existentes nessa cadeia.

6.2-Desenvolva uma função iterativa que recebe como parâmetro uma cadeia de caracter e converte todas as letras minúsculas em letras minúsculas.

6.3-Desenvolva uma função iterativa que recebe como parâmetro uma cadeia de caracteres e coloca as letras maiúsculas à esquerda e a minúsculas à direita.

6.4-Desenvolva uma função iterativa que recebe como parâmetros duas cadeias de caracteres, e retorna o número de vezes que a cadeia menor está contida na cadeia maior. Desenvolva também um segmento de código que invoca essa função

6.5-Desenvolva uma função iterativa que recebe como parâmetro duas cadeias de caracteres e um número inteiro positivo k. Compara os k primeiros dígitos das duas cadeias e retornar -1 se a primeira cadeia for menor do que a segunda; 0 se as cadeias forem iguais; e 1 se a primeira cadeia é maior do que a segunda.

6.6-Desenvolva uma função iterativa que recebe como parâmetro uma cadeia de caracteres e retorna essa cadeira de caracteres com um traço entre cada dígito.

6.7-Desenvolva uma função iterativa que recebe como parâmetro uma cadeira de caracteres e um determinado caracter. Retorna a posição da última ocorrência desse caracter se existir ou -1 no caso contrário.

6.8-Desenvolva uma função iterativa que recebe como parâmetro uma cadeia de caracteres e a devolve sem caracteres repetidos.

6.9-Desenvolva uma função iterativa que recebe como parâmetro uma cadeia de caracteres e a devolve com os caracteres maiúsculos e minúsculos alternados.

6.10-Desenvolva uma função recursiva que recebe como parâmetro duas cadeias de caracteres e verifica se elas são iguais. Desenvolva também um segmento de código que invoca essa função.

6.11-Desenvolva uma função iterativa que recebe como parâmetro duas cadeias de caracteres com a mesma dimensão e retorna -1 se a cadeia s1 é menor do que a cadeia s2, 0 se a cadeia s1 é igual a cadeia s2, e 1 se a cadeia s1 é maior do que a cadeia s2.

7 Divisão e Conquista

7.1-Desenvolva uma função para calcular o número de Tribonacci. A sequência de Tribonacci pode ser definida pela seguinte relação de recorrência.

$$Tribonacci(n) = \begin{cases} 0 & se\ \ n = 0 \\ 1 & se\ n = 1\ ou\ n = 2 \\ Tribonacci(n-1) + Tribonacci(n-2) + Tribonacci(n-3)\ se\ n > 3 \end{cases}$$

Desenvolva uma função recursiva com a estratégia de divisão e conquista e uma iterativa com a programação dinâmica.

7.2-Desenvolva uma função recursiva que entre outros parâmetros recebe dois vectores do mesmo tipo e com o mesmo número de elementos. Verifique se esses vectores são iguais. A sua função deve retornar 1 se os vectores forem iguais e 0 no caso contrário.

7.3-Desenvolva uma função recursiva que entre outros parâmetros recebe um vector com elementos do tipo real e um valor do mesmo tipo. Retornar o elemento que mais se aproxima do valor passado como parâmetro.

7.4-Desenvolva uma função recursiva que entre outros parâmetros recebe dois vector cujos elementos são do tipo lógico (0 = false 1 = true). Devolva um vector C, constituído por elementos baseados na disjunção exclusiva, descrita pela tabela. Por definição, a disjunção exclusiva de duas proposições é falsa quando eles têm o mesmo valor lógico e verdadeira no caso contrário.

7.5- Dada a seguinte função:

```
int S (int n)
 {
    if (n == 1)
       return 1;
    else if (n == 2)
              return 3;
          else if (n % 2)
                    return 3 * s((n - 1) / 2) +  s((n + 1) / 2));
                else
                    return 3 * s (n / 2) + s(n / 2 - 1);
  }
```

a) Calcule o valor de s(9).
b) Quantas chamadas recursivas a função executa.
c) Utilize a estratégia de programação dinâmica para desenvolver uma função iterativa equivalente.

7.6-Desenvolva uma função recursiva que entre outros parâmetros, recebe um vector e uma determinada posição k. Separar o vector em dois de tal forma que os elementos que se encontram nas posições de 0 à k, vão para o primeiro vector enquanto os restantes para o segundo.

7.7-Desenvolva uma função recursiva que recebe entre outros parâmetros recebe um vector. Contar o número elementos iguais a zeros.

7.8-Desenvolva uma função recursiva que recebe entre outros parâmetros uma cadeia de caracteres. Verifique se essa cadeia de caracteres é do tipo capicua.

8 Vectores Bidimensionais

8.1-Desenvolva uma função iterativa que recebe como parâmetro uma matriz retangular com n linhas e m colunas, cujos elementos são do tipo inteiro. Verifique se essa matriz é simétrica.

8.2-Desenvolva uma função iterativa que recebe como parâmetros duas matrizes retangulares com n linhas e m colunas, cujos elementos são do tipo inteiro. Verifique se essas matrizes são iguais.

8.3- Desenvolva uma função iterativa que recebe como parâmetros uma matriz quadrada, com n linhas e n colunas, cujos elementos são do tipo inteiro. Verifique se essa matriz é um quadrado latino de tamanho n. Por definição, num quadrado

latino de tamanho n, em cada linha e em cada coluna aparecem todos os inteiros 1,2,3, …, n, ou seja, cada linha ou coluna é uma permutação dos n primeiros inteiros. Por exemplo: n = 3.

$$\begin{vmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{vmatrix}$$

8.4-Suponha que a matriz de adjacência M representa a ligação entre n cidades. Por exemplo, na matriz abaixo, temos duas estradas que saem da cidade 1, uma chega a cidade 2, e a outra chega a cidade 3; na cidade 2 temos duas estradas, uma que chega a cidade 1 e outra que chega a cidade 3; na cidade 3 temos três estradas, uma que chega a cidade 1, a outra que chega a cidade 2, e por fim, a última que chega a cidade 4; na cidade temos apenas uma estrada que chega a cidade 3.

$$\begin{vmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

Desenvolva uma função iterativa quer recebe como parâmetro uma matriz de adjacencia e uma determinada cidade. Mostrar todas as cidades vizinhas.

8.5-Desenvolva uma função iterativa que recebe como parâmetro uma matriz quadrada com n linha e n colunas, cujos elementos são do tipo inteiro. Devolva essa matriz com os elementos ordenados em ordem crescente por colunas. Por exemplo:

3	1	9
7	6	2
5	8	4

← Antes

Depois →

1	4	7
2	5	8
3	6	9

8.6-Desenvolva uma função recursiva que entre outros parâmetros recebe duas matrizes rectangulares com n linhas e m colunas. Devolve uma matriz rectangular com n linhas e m colunas cujos elementos são a adição dos correspondentes elementos das matrizes entradas como parâmtros.

8.7-Desenvolva uma função recursiva para calcular o determinante de uma matriz quadrada com m linhas e m colunas.

8.8- Desenvolva uma função iterativa que recebe como parâmetros uma matriz de adjacência, que representam a quantidade de quilómetros as rotas da TAAG. Dado um determinado aeroporto, mostre todas as possíveis viagens que um passageiro passa fazer e a quantidade de quilómetros que irá percorrer.

8.9-Desenvolva uma função recursiva que entre outros parâmetros, recebe uma matriz de adjacência, que representa as cidades de um país que possuem aeroportos internacionais. Retorne o número de aeroportos internacionais que esse país tem e as cidades onde esses estão localizados.

8.10-Desenvolva um procedimento iterativo, que recebe como parâmetro uma matriz rectangular, com n linhas e m columas e devolve o maior e o menor elemento.