

Estudo para a prova 2a

No material que segue, nos exemplos, os dados fornecidos pelo operador estão escritos em *itálico*, enquanto as respostas dadas pelo programa estão em **negrito**.

1. Escreva uma função de nome **takedrop** que receba um inteiro n ($1 < n \leq 10$), uma chave k ($-n < k < n$) e um vetor de n inteiros e aplique a operação *take* se k for positivo e a função *drop* se k for negativo. Acompanhe nos exemplos:

3 *takedrop* 1 2 3 4 5 6 é igual a 1 2 3

-3 *takedrop* 1 2 3 4 5 6 é igual a 4 5 6

1 *takedrop* 10 20 30 40 é igual a 10

-1 *takedrop* 10 20 30 40 é igual a 20 30 40

Depois, escreva um programa que leia n , a chave k e o vetor, aplique a função sobre o vetor e imprima o resultado. Veja os exemplos: 6, 3, 1 2 3 4 5 6 **1 2 3**

6, -3, 1 2 3 4 5 6 **4 5 6**

7, 2, 10 20 30 40 50 60 70 **10 20**

5, 0, 1 2 3 4 5 **1 2 3 4 5**

2. Escreva uma função que receba um inteiro n ($n \leq 10$), uma base b ($1 < b \leq 9$) e um vetor de n inteiros. Os valores v do vetor serão $0 \leq v < b$. Este vetor representa um número expresso na base b . O objetivo da função é converter este número para a base decimal. Acompanhe nos exemplos:

2 2 4 $1_3 = 1 \times 3^0 + 4 \times 3^1 + 2 \times 3^2 + 2 \times 3^3 = 1 \times 1 + 4 \times 3 + 2 \times 9 + 2 \times 27 = 1 + 12 + 18 + 54 = 85$

5 4 $3_7 = 3 \times 7^0 + 4 \times 7^1 + 5 \times 7^2 = 3 \times 1 + 4 \times 7 + 5 \times 49 = 3 + 28 + 245 = 276$

3 $4_5 = 4 \times 5^0 + 3 \times 5^1 = 4 \times 1 + 3 \times 5 = 4 + 15 = 19$. Depois escreva um programa que leia n , b e o vetor de números, chame a função e imprima o resultado. Exemplos de uso:

4, 3, 2 2 0 1 **73**

3, 7, 5 4 3 **276**

2, 5, 3 4 **19**

3. Escreva uma função chamada **arredond** que receba n ($n \leq 10$) e um vetor de n reais e devolve um vetor de n inteiros que contém os valores do vetor de entrada devidamente arredondados. Use a regra tradicional para arredondar x , a saber: se a parte fracionária de x é igual ou maior a 0.5, o arredondamento é igual à parte inteira de $x + 1$ enquanto se a parte fracionária é menor que 0.5, o arredondamento é igual à parte inteira de x . Acompanhe os exemplos
arredond(1.5, 2, 2.99, 30.01, 99) = 2, 2, 3, 30, 99
arredond(1, 1.2, 1.4, 1.6, 1.8, 2) = 1, 1, 1, 2, 2, 2)

Depois escreva um programa que leia o vetor de reais e imprima o vetor de inteiros. Veja:

1.5 2 2.99 30.01 99 **2 2 3 30 99**

1 1.2 1.4 1.6 1.8 2 **1 1 1 2 2 2**

4. Escreva uma função de nome **emord** que receba um inteiro n ($n \leq 10$) e um vetor composto de n inteiros. A função deve analisar o vetor e responder: 1 = se o vetor estiver em ordem crescente, 2 = se estiver em ordem estritamente crescente e 0 = nenhuma das duas alternativas anteriores. Definição: o vetor v está em ordem crescente se e somente se $v[i] \geq v[j] \forall i > j$ e em ordem estritamente crescente se e somente se $v[i] > v[j] \forall i > j$. Depois escreva um programa que leia n e o vetor associado, chame a função e imprima o

resultado devolvido pela função. Acompanhe os exemplos:

6, 1 2 3 4 5 6 **2**

6, 1 2 2 3 4 5 **1**

6, 1 2 3 4 3 6 **0**

5. Escreva uma função de nome **areatri** que receba 6 valores reais, representando 3 pares de pontos, pela ordem x_1, y_1, x_2, y_2, x_3 e y_3 e devolva a área ocupada pelo triângulo determinado por esses 3 pontos. A área de um triângulo é determinada pela metade do módulo do determinante da seguinte matriz

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Depois que a função estiver operacional, escreva um programa que leia 3 pontos (pelas suas coordenadas cartesianas) e devolva 1 se a origem (o ponto 0,0) estiver DENTRO do triângulo definido pelos 3 pontos e 0 senão. Para resolver este problema pense no seguinte algoritmo. Constrói-se 3 triângulos, tomando 2 pontos dos fornecidos e mais a origem. Se a soma dos 3 triângulos for igual à área do triângulo original, (0,0) está dentro do triângulo. Se a soma for maior que a área do triângulo original, então a origem está fora. Se não entendeu a explicação, faça os dois desenhos num papel e medite sobre eles. Veja os exemplos

0 0, 0 1, 1 0 **1**

2 2, 3 3, 5 2 **0**

6. Escreva um programa C++ que leia n ($n \leq 10$) e a seguir uma matriz quadrada $A_{n \times n}$ de inteiros e depois imprima a diferença entre o maior valor na diagonal principal e o menor valor da diagonal secundária. Um elemento $v[i][j]$ pertence à diagonal principal se e somente se $i = j$. Já um elemento $v[i][j]$ pertence à diagonal secundária se e somente se $i + j = n - 1$. Exemplos:

3, 7 4 2 9 6 1 5 3 9 **7** (resultado de 9-2)

3, 0 1 1 3 4 5 6 7 5 **4** (resultado de 5-1)

7. Escreva um programa que leia um inteiro b e um real x ($x > 1$, para o logaritmo ser positivo) e imprima o logaritmo base b de x . Use o algoritmo das médias geométricas para chegar neste valor. Considere que se $a^b = c$ então $\log_a c = b$. Eis a descrição do algoritmo. Primeiro, há que se estabelecer limites para a atuação do algoritmo. Começa-se com $t = 1$ e daí segue-se enquanto $x > b^t, t++$. Ao final deste ciclo, o intervalo de busca é composto por $t - 1$ e t . Calcula-se a média m entre estes dois valores e se $b^m > x$ então a média substitui o segundo valor, senão o primeiro. Retorna-se daqui para o cálculo da média. Quando o erro cometido $|b^m - x| < 0.001$ o algoritmo imprime m e acaba.
8. Imagine uma matriz retangular de m ($3 \leq m \leq 10$) linhas por n colunas ($3 \leq n \leq 12$). Agora imagine os 8 vizinhos dos elementos que não estão na borda da matriz. Seu programa C++ deve ler m, n e os elementos de $A_{m \times n}$ e depois totalizar os valores do vizinhos e apontar qual a célula (*linha, coluna*) cujo total de vizinhos é maior. Se houver empate mostre qualquer um dos empatantes.

```

#include<iostream>
#include<iomanip>
#include<string>
#include<cmath>
using namespace std;
//----- takedrop -----
void takedrop(int &n, int k, int v[]){
    int i;
    if (k<0){
        i=0;
        while(i+abs(k)<=n){
            v[i]=v[i+abs(k)];
            i++;
        }
        n=i-1;
    }
    else {
        n=k;
    }
}
/*
int main(){
    int n,i;
    int v[10]={1,2,3,4,5,6};
    n=6;
    takedrop(n,1,v);
    for (i=0;i<n;i++){cout<<v[i]<<" ";}
} */
//----- conversao de base -----
int converte(int n, int b, int v[]){
    int res=0;
    int i;
    for(i=n-1;i>=0;i--){
        res=res+v[i]*pow(b,(n-(i+1)));
    }
    return res;
} /*
int main(){
    int i,x,n,b;
    int v[10];
    cin>>n>>b;
    for (i=0;i<n;i++){cin>>v[i];}
    x=converte(n,b,v);
    cout<<x;
} */
//----- arredondamento -----
void arredond(int n, float va[], int vd[]){
    int i;
    for (i=0;i<n;i++){
        if (va[i]==ceil(va[i])){vd[i]=va[i];}
        else{
            vd[i]=trunc(va[i]+0.5);
        }
    }
} /*
int main(){
    float a[10]={0.4, 5, 1.9, 3, 5, 7.99, 8.01};
    int i,b[10];
    arredond(7,a,b);
    for (i=0;i<7;i++){cout<<b[i]<<" ";}
} */
//----- vetor em ordem crescente e estritamente crescente -----
int emord(int n, int v[]){

```

```

int rec=2,rc=1,i;
for(i=1;i<n;i++){
    if (v[i]<v[i-1]){rec=rc=0;}
    if (v[i]==v[i-1]){rec=0;}
}
if (rec!=0){return rec;}
else{
    if(rc!=0){return rc;}
}
return 0;
} /*
int main(){
    int n,i,v[10];
    cin>>n;
    for (i=0;i<n;i++){cin>>v[i];}
    cout<<emord(n,v);
} /*
//----- triângulo inclue a origem ? -----
float areatri(float x1, float y1, float x2, \
    float y2, float x3, float y3){
    float a;
    a=(x1*y2)+(y1*x3)+(x2*y3);
    a=a-((x3*y2)+(x1*y3)+(x2*y1));
    a=a/2;
    return a;
} /*
int main(){
    float a,b,c,d,e,f;
    cin>>a>>b>>c>>d>>e>>f;
    float area1,area2;
    area1=abs(areatri(a,b,c,d,e,f));
    area2=abs(areatri(0,0,c,d,e,f));
    area2=area2+abs(areatri(a,b,0,0,e,f));
    area2=area2+abs(areatri(a,b,c,d,0,0));
    if (area1==area2){cout<<1;}
    else{cout<<0;}
} /*
/*
//----- maior na d.p. menos menor na d.s. -----
int main(){
    int n,i,j;
    int m[10][10];
    cin>>n;
    for(i=0;i<n;i++){for(j=0;j<n;j++){cin>>m[i][j];}}
    int maxdp=-999999;
    int minds=+999999;
    for (i=0;i<n;i++){
        if (m[i][i]>maxdp){maxdp=m[i][i];}
        if (m[i][(n-1)-i]<minds){minds=m[i][(n-1)-i];}
    }
    cout<<maxdp-minds;
}
/*  /*
//----- logaritmo na marra -----
int main(){
    int t,b,lixo;
    float m,x,z[2];
    cin>>b>>x;
    if (x<1){cout<<"erro"; return 1;}
    t=1;
    while(x>pow(b,t)){t++;}
    z[0]=t-1; z[1]=t;
    while (1==1){

```

```

        if (z[1]==z[0]){cout<<"nao foi possivel"; return 1;}
        m=(z[0]+z[1])/2.0;
        if(0.0001>abs((pow(b,m)-x))) {cout<<m;return 0;}
        if (pow(b,m)>x){z[1]=m;}
        else {z[0]=m;}
    }
}  */
//----- maiores vizinhos -----
int main(){
    int m,n,i,j,maxv=-99999,maxi,maxj,s;
    int a[10][12];
    cin>>m>>n;
    for(i=0;i<m;i++){for(j=0;j<n;j++){cin>>a[i][j];}}
    for(i=1;i<m-1;i++){
        for(j=1;j<n-1;j++){
            s=a[i-1][j-1]+a[i-1][j]+a[i-1][j+1];
            s=s+a[i][j-1]+a[i][j+1];
            s=s+a[i+1][j-1]+a[i+1][j]+a[i+1][j+1];
            if (s>maxv){
                maxv=s;
                maxi=i;
                maxj=j;
            }
        }
    }
    cout<<maxi<<" "<<maxj;
}

```