

## LISTA DE EXERCÍCIOS VETOR – RESOLUÇÕES (VISUALG/C++)

### ALGORITMO "vetorEx1"

```
VAR
    n, i, j, contAbaixo: INTEIRO
    nomes, nomesAcima: VETOR [1..20] DE CARACTERE
    notas: VETOR [1..20] DE REAL
    media: REAL
INICIO
    // ENTRADAS
    REPITA
        ESCREVA("Qtde de alunos (max 20): ")
        LEIA(n)
    ATE (n > 0) E (n <= 20)
    PARA i DE 1 ATE n FAÇA
        ESCREVA("Nome: ")
        LEIA(nomes[i])
        REPITA
            ESCREVA("Nota: ")
            LEIA(notas[i])
        ATE (notas[i] >= 0) E (notas[i] <= 10)
    FIMPARA
    // PROCESSAMENTOS
    PARA i DE 1 ATE n FAÇA
        media <- media + notas[i]
    FIMPARA
    media <- media/n
    // so depois de calcular a media pode-se comparar os alunos
    PARA i DE 1 ATE n FAÇA
        SE notas[i] >= media ENTAO
            j <- j + 1
            nomesAcima[j] <- nomes[i] // nome[i] fica na pos[j] agora
        SENAO
            contAbaixo <- contAbaixo + 1
        FIMSE
    FIMPARA
    // RESULTADOS
    ESCREVAL("Media da turma:", media)
    ESCREVAL("Alunos acima da media")
    PARA i DE 1 ATE j FAÇA
        ESCREVAL(nomesAcima[i])
    FIMPARA
    ESCREVAL("Qtde abaixo da media: ", contAbaixo)
FIMALGORITMO
```

```

#include <iostream> // tradução do algoritmo anterior
using namespace std;
#define TMAX 20

int main(){
    int n, j=0, contAbaixo=0;
    string nomes[TMAX], nomesAcima[TMAX];
    float notas[TMAX], media=0;
    do{
        cout<<"Qtde de alunos: "; cin>>n;
    }while(n<=0 or n>TMAX);
    for(int i=0; i<n; i++){
        cin.ignore(); // limpar o buffer
        cout<<"Nome: "; getline(cin,nomes[i]);
        do{
            cout<<"Nota: "; cin>> notas[i];
        }while (notas[i]<0 or notas[i]>10);
    }
    for(int i=0; i<n; i++)
        media += notas[i];
    media = media/n;
    for(int i=0; i<n; i++)
        if(notas[i] >= media){
            nomesAcima[j] = nomes[i];
            j++;
        }
        else contAbaixo++;
//RESULTADOS
    cout<<"Media da turma:"<< media<< endl;
    cout<<"Alunos acima da media"<<endl;
    for(int i=0; i<j; i++)
        cout<<nomesAcima[i]<<endl;
    cout << "Qtde abaixo da media: "<< contAbaixo<< endl;
    return 0;
}

```

**ALGORITMO "vetorEx2"**

VAR

n, i: INTEIRO  
 v, vs: VETOR [1..10] DE REAL

INICIO

// LEITURA

REPITA

ESCREVA("Qtde de elementos (3 a 10): ")

LEIA(n)

ATE (n &gt;=3) E (n &lt;= 10)

PARA i DE 1 ATE n FAÇA

ESCREVA(i, "o. elemento: ")

LEIA(v[i])

FIMPARA

//SUAVIZACAO

vs[1] &lt;- v[1]

vs[n] &lt;- v[n]

PARA i DE 2 ATE n-1 FAÇA

vs[i] &lt;- (v[i-1] + v[i] + v[i+1])/3

FIMPARA

//RESULTADO

ESCREVA("Novo vetor")

PARA i DE 1 ATE n FAÇA

ESCREVA("[", vs[i],"]") // vai mostrar [1][10][-5] p.ex.

FIMPARA

**FIMALGORITMO**

#include &lt;iostream&gt; // tradução do algoritmo anterior

using namespace std;

int main(){

int n;

float v[10], vs[10];

// LEITURA

do{

cout &lt;&lt;"Qtde de elementos (3 a 10): "; cin&gt;&gt;n;

}while(n&lt;3 or n&gt; 10);

cout &lt;&lt; "Vetor" &lt;&lt; endl;

for(int i=0; i&lt;n; i++){

cout&lt;&lt;i+1&lt;&lt;"o. elemento: "; cin&gt;&gt;v[i];

}

//SUAVIZACAO

vs[0] = v[0];

vs[n-1] = v[n-1];

for(int i=1; i&lt;n-1;i++)

vs[i] = (v[i-1]+ v[i]+v[i+1])/3; // nao precisa bloco {}

//RESULTADO

cout&lt;&lt;"Vetor suavizado"&lt;&lt;endl;

for(int i=0; i&lt;n; i++) cout&lt;&lt;"["&lt;&lt;vs[i]&lt;&lt;"]";

return 0;

}

**ALGORITMO "exerc3"**

VAR

n, i: INTEIRO

soma: REAL

z, v, w: VETOR[1..15] DE REAL // z sempre serah o vetor resultante

INICIO

REPITA

ESCREVA("Qtde de elementos (1 a 15): ")

LEIA(n)

ATE (n &gt;= 1 E n &lt;= 15)

ESCREVA("Vetor V: ")

PARA i DE 1 ATE n FAÇA

ESCREVA(i, "o. elemento: ")

LEIA(v[i])

FIMPARA

ESCREVA("Vetor W: ")

PARA i DE 1 ATE n FAÇA

ESCREVA(i, "o. elemento: ")

LEIA(w[i])

FIMPARA

PARA i DE 1 ATE n FAÇA

z[i] &lt;- v[i] + w[i]

FIMPARA

ESCREVA("Vetor resultante da soma: ")

PARA i DE 1 ATE n FAÇA

ESCREVA("[", z[i], "]") //vai mostrar assim [1][10][-5] p.ex.

FIMPARA

PARA i DE 1 ATE n FAÇA

z[i] &lt;- v[i] - w[i]

FIMPARA

ESCREVA("Vetor resultante da subtracao: ")

PARA i DE 1 ATE n FAÇA

ESCREVA("[", z[i], "]")

FIMPARA

PARA i DE 1 ATE n FAÇA

z[i] &lt;- v[i] \* w[i]

FIMPARA

ESCREVA("Vetor resultante da multiplicacao: ")

PARA i DE 1 ATE n FAÇA

ESCREVA("[", z[i], "]")

FIMPARA

PARA i DE 1 ATE n FAÇA

z[i] &lt;- v[i] / w[i]

FIMPARA

ESCREVA("Vetor resultante da divisao: ")

PARA i DE 1 ATE n FAÇA

ESCREVA("[", z[i], "]")

FIMPARA

soma &lt;- 0

PARA i DE 1 ATE n FAÇA //poderia aproveitar o vetor de multiplic

soma &lt;- soma + v[i] \* w[i]

FIMPARA

SE soma = 0 ENTAO

ESCREVA ("VETORES V E W SAO ORTOGONAIS")

SENAO

ESCREVA ("VETORES V E W NAO SAO ORTOGONAIS")

FIMSE

**FIMALGORITMO**

```

#include <iostream> // tradução do algoritmo anterior
using namespace std;
int main(){
    int n;
    float v[15], z[15], w[15], soma;
// LEITURA
    do{
        cout <<"Qtde de elementos (max 15): "; cin>>n;
    }while(n<1 or n> 15);
    cout << "Vetor V" << endl;
    for(int i=0; i<n; i++){
        cout<<i+1<<"o. elemento: "; cin>>v[i];
    }
    cout << "Vetor W" << endl;
    for(int i=0; i<n; i++){
        cout<<i+1<<"o. elemento: "; cin>>w[i];
    }
// CALCULOS SOLICITADOS + APRESENTACAO RESULTADO
    for(int i=0; i<n; i++) z[i] = (v[i] + w[i]);
    cout<<"Vetor resultante da soma"<<endl;
    for(int i=0; i<n; i++) cout<<"["<<z[i]<<"]";

    for(int i=0; i<n; i++) z[i] = (v[i] - w[i]);
    cout<<"Vetor resultante da subtracao"<<endl;
    for(int i=0; i<n; i++) cout<<"["<<z[i]<<"]";

    for(int i=0; i<n; i++) z[i] = (v[i] * w[i]);
    cout<<"Vetor resultante da multiplicacao"<<endl;
    for(int i=0; i<n; i++) cout<<"["<<z[i]<<"]";

    for(int i=0; i<n; i++) z[i] = (v[i] / w[i]);
    cout<<"Vetor resultante da divisao"<<endl;
    for(int i=0; i<n; i++) cout<<"["<<z[i]<<"]";

    soma = 0;
    for(int i=0; i<n; i++) z[i]=(v[i]* w[i]);
    if(soma == 0) cout <<"VETORES V E W SAO ORTOGONAIS"<<endl;
    else cout <<"VETORES V E W NAO SAO ORTOGONAIS"<<endl;

    return 0;
}

```

**ALGORITMO "vetorEx4"**

```
VAR
    n, i, maior, menor: INTEIRO
    numeros: VETOR [1..15] DE INTEIRO
INICIO
    REPITA
        ESCREVA("Qtde de elementos: ")
        LEIA(n)
    ATE (n >= 2) E (n <= 15)
    PARA i DE 1 ATE n FAÇA
        ESCREVA(i, "o. elemento: ")
        LEIA(numeros[i])
    FIMPARA

    menor <- numeros[1] // fica melhor fazer separado por legibilidade
    PARA i DE 2 ATE n FAÇA
        SE menor > numeros[i] ENTAO
            menor <- numeros[i]
        FIMSE
    FIMPARA

    maior <- numeros[1] // fica melhor fazer separado por legibilidade
    PARA i DE 2 ATE n FAÇA
        SE maior < numeros[i] ENTAO
            maior <- numeros[i]
        FIMSE
    FIMPARA

    ESCREVAL("Menor elemento:", menor)
    ESCREVAL("Maior elemento:", maior)
FIMALGORITMO
```

```

#include <iostream> // tradução do algoritmo anterior
#define TMAX 15
using namespace std;
int main(){
    int n, maior, menor;
    float numeros[TMAX];
    // LEITURA
    do{
        cout <<"Qtde de elementos: "; cin>>n;
    }while(n<1 or n> TMAX);
    for(int i=0; i<n; i++){
        cout<<i+1<<"o. elemento: "; cin>>numeros[i];
    }
    // CALCULOS SOLICITADOS
    menor = numeros[0]; // melhor fazer separado
    for(int i=1; i<n; i++)
        if(numeros[i] < menor)
            menor = numeros[i];

    maior = numeros[0]; // melhor fazer separado
    for(int i=1; i<n; i++)
        if(numeros[i] > maior)
            maior = numeros[i];
    // RESULTADOS
    cout<<"Menor elemento:"<<menor<<endl;
    cout<<"Maior elemento:"<<maior<<endl;
    return 0;
}

```

**ALGORITMO "vetorEx5 - sem alturas repetidas, com posicionamento"**

VAR

```
    inscr: VETOR [1..100] DE CARACTERE
    i, n, posMaior, posMenor, soma: INTEIRO
    alturas: VETOR [1..100] DE INTEIRO
```

INICIO

REPITA

ESCREVA("informe n atletas")

LEIA(n)

ATE (n&gt;0) E (n&lt;=100)

PARA i DE 1 ATE n FACA

ESCREVAL("Inscricao ",i)

LEIA(inscr[i])

ESCREVAL("Altura(em cm) ",i)

LEIA(alturas[i]) //incluir laco para validacao desta leitura

FIMPARA

// a solucao esta baseada na posicao do atleta

posMaior &lt;- 1 //usa o 1o atleta como referencia para comparacao

posMenor &lt;- 1

soma &lt;- alturas[1]

PARA i DE 2 ATE n FACA

SE alturas[i] &gt; alturas[posMaior] ENTAO

posMaior &lt;- i

SENAo

SE alturas[i] &lt; alturas[posMenor] ENTAO

posMenor &lt;- i

FIMSE

FIMSE

soma &lt;- soma + alturas[i]

FIMPARA

ESCREVAL("Atleta +alto: ",inscr[posMaior],"-", alturas[posMaior], "cm")

ESCREVAL("Atleta +baixo: ",inscr[posMenor],"-", alturas[posMenor], "cm")

ESCREVAL("Media: ", soma/n, "cm")

**FIMALGORITMO**



```

#include <iostream> // tradução do algoritmo anterior
using namespace std;
#define TMAX 100
int main(){
    string inscr[TMAX];
    int n, posMaior, posMenor, soma, alturas[TMAX];
// LEITURA
    do{
        cout <<"Qtde de atletas (max"<< TMAX << "): ";
        cin>>n;
    }while(n<1 or n> TMAX);
    for(int i=0; i<n; i++){
        cin.ignore(); // limpar buffer do cin
        cout<<i+1<<"a. inscricao: ";
        getline(cin,inscr[i]);
        do{
            cout<<"Altura(em cm) ";
            cin>>alturas[i];
        }while(alturas[i]<110 or alturas[i]>220);
    }
// CALCULOS SOLICITADOS
// a solucao esta baseada na posicao do atleta

    posMaior = posMenor = 0; //usa 1o atleta como referencia para comparação
    soma = alturas[0];

    for(int i=1; i<n; i++){
        if(alturas[i] > alturas[posMaior])
            posMaior = i;
        else if(alturas[i] < alturas[posMenor])
            posMenor = i;
        soma += alturas[i]
    }
// RESULTADOS
    cout<<"Atleta+alto:"<<inscr[posMaior]<<"-"<<alturas[posMaior]<<"cm";
    cout<<endl;
    cout<<"Atleta+baixo:"<<inscr[posMenor]<<"-"<<alturas[posMenor]<<"cm";
    cout<<endl;
    cout<<"Media:"<< soma/n<<"cm"<<endl;
    return 0;
}

```

**ALGORITMO "vetorEx6"**

```

VAR
    i,n,aux: INTEIRO
    vet: VETOR [1..10] DE INTEIRO
INICIO
    REPITA
        ESCREVA("informe n (2 a 10):")
        LEIA(n)
    ATE (n>=2) E (n<=10)
    PARA i DE 1 ATE n FACA
        ESCREVA("Elemento ",i)
        LEIA(vet[i])
    FIMPARA
    PARA i DE 1 ATE n DIV 2 FACA
        aux<- vet[i]
        vet[i]<-vet[n-i+1]
        vet[n-i+1]<-aux
    FIMPARA
    ESCREVAL("VETOR INVERTIDO: ")
    PARA i DE 1 ATE n FACA
        ESCREVA("[", vet[i],"]")
    FIMPARA

```

**FIMALGORITMO**

```

#include <iostream>
using namespace std;
int main(){
    int n, aux;
    int vet[10];
    // LEITURA
    do{
        cout <<"Qtde de elementos (2 a 10): "; cin>>n;
    }while(n<2 or n>10);
    for(int i=0; i<n; i++){
        cout<<i+1<<"o. elemento: "; cin>>vet[i];
    }
    // CALCULOS SOLICITADOS
    for(int i=0; i<n/2; i++){ // n/2 vai dar int
        aux = vet[i];
        vet[i] = vet[n-1-i];
        vet[n-1-i] = aux;
    }
    // RESULTADOS
    cout<<"Vetor invertido:"<<endl;
    for(int i=0; i<n; i++) cout<<"["<<vet[i]<<"]";
    return 0;
}

```

**ALGORITMO "vetorEx7"**

VAR

i, n, codpraia: INTEIRO

umarenda: REAL

nturistas: VETOR [1..42] DE INTEIRO //C++ lembrar: inicializar com 0

rendas: VETOR [1..42] DE REAL //C++ lembrar: inicializar com 0

INICIO

REPITA

ESCREVA("informe n pesquisados")

LEIA(n)

ATE (n&gt;0) E (n&lt;=3000)

PARA i DE 1 ATE n FACA

REPITA

ESCREVAL("Codigo da praia pref (1-42)")

LEIA(codpraia)

ATE (codpraia&gt;=1) E (codpraia&lt;=42)

REPITA

ESCREVAL("Sua renda")

LEIA(umarenda)

ATE (umarenda&gt;=0)

nturistas[codpraia] &lt;- nturistas[codpraia] + 1

rendas[codpraia] &lt;- rendas[codpraia] + umarenda

FIMPARA

PARA i DE 1 ATE 42 FACA

SE (nturistas[codpraia] &gt; 1) ENTAO

rendas[codpraia] &lt;- rendas[codpraia] / nturistas[codpraia]

FIMSE

FIMPARA

ESCREVAL("Praia = N.Turistas = RendaMedia")

PARA i DE 1 ATE 42 FACA

ESCREVAL(i, " ", nturistas[i], " ", rendas[i])

FIMPARA

**FIMALGORITMO**

```

#include <iostream> // tradução do anterior
#include <iomanip>
using namespace std;
int main(){
    int n, codpraia;
    int nturistas[42]= {0}; // precisa inicializar o vetor de contadores
    float umarenda, rendas[42]= {0.0}; // idem
    // LEITURA
    do{
        cout <<"Qtde de pessoas a serem pesquisadas: "; cin>>n;
    }while(n<0 or n> 3000);

    for(int i=0; i<n; i++){
        do{
            cout<<"Codigo da praia pref (1-42):"; cin>>codpraia;
        }while(codpraia<1 or codpraia>42);
        do{
            cout<<"Sua renda R$:"; cin>>umarenda;
        }while(umarenda<0);
        codpraia--; // para ajustar aos indices do vetor
        nturistas[codpraia]++;
        rendas[codpraia] += umarenda;
    }
    for(int i=0; i<42; i++)
        if(nturistas[i]>1) rendas[codpraia] /= nturistas[codpraia];

    cout<< fixed << setprecision(2);
    cout<<"Praia = N.Turistas = RendaMedia"<<endl;
    for(int i=0; i<42; i++){
        cout<<setw(3)<<i+1<<setw(10)<<nturistas[i];
        cout<<setw(16)<<rendas[i]<<endl;
    }
    return 0;
}

```