

Material de apoio para disciplina GCC224 - Introdução aos Algoritmos
Período: ERE 2020

Neste material você encontrará dicas para pensar antes de escrever uma solução, códigos discutidos passo-a-passo.

Não pense que as sugestões de códigos a seguir são as melhores soluções. Sempre se pergunte como você faria. Se quiser, compartilhe conosco sua solução.

Link do vídeo do atendimento:

18/06/2020 (vetores e for): <https://youtu.be/65znGswogYY>

22/06/2020 (matrizes e for): <https://youtu.be/cnguYFgB1Z4>

Vetores

1) Entrada de dados: com while e for

usando while	usando for
<pre>int main () { int tamanho,i; cin>>tamanho; int v[tamanho]; i=0; while (i<tamanho) { cin>>v[i]; i++; // próxima posição } return 0; }</pre>	<pre>int main () { int tamanho,i; cin>>tamanho; int v[tamanho]; for (i=0; i<tamanho; i++) { cin>>v[i]; } return 0; }</pre>

O tamanho do vetor deve ser conhecido na sua declaração.

2) Entrada de dados: preencher um vetor com n números pares. Os números serão lidos do teclado.

Aqui, o usuário pode digitar ímpares e pares, mas somente quando achar um valor par que ele é inserido no vetor. A solução a seguir utiliza a estrutura de repetição while. Observe que o valor de i (posição) somente é incrementada após o teste de if.

Pense um pouco: *isso poderia ser feito com a estrutura for?*

```

int tamanho,i, valor;
cin>>tamanho;
int v[tamanho];

i=0;
while (i<tamanho) {
    cin>>valor;
    if (valor%2==0){
        v[i]=valor;
        i++; //incrementa somente aqui
    }
}

```

3) Endereço de memória.

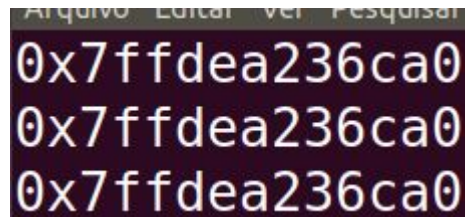
O endereço de memória de uma variável é obtido por meio do operador &.

O valor de um vetor é o seu endereço de memória, do seu primeiro elemento. Veja como as 3 linhas do cout tem o mesmo efeito.

```

int vetor[5] = {1,2,3,4,5};
cout << vetor << endl;
cout << &vetor << endl;
cout << &vetor[0] << endl;

```



4) Passando um vetor como parâmetro para uma função/procedimento.

Pode ser necessário imprimir o conteúdo de um vetor em diversos momentos de um programa. É interessante ter um subprograma que faça isso. Para isso, o subprograma precisa receber a quantidade de elementos do vetor (que é um valor inteiro) e o vetor propriamente dito.

Exemplo com procedimento. Veja que o parâmetro que corresponde ao vetor tem o tipo, o nome do parâmetro e o par de colchetes vazio. Sem o par de colchetes, o compilador não entenderia que se trata de um vetor.

```

void saida (int v[], int t){
    int i;
    for (i=0; i<t; i++)
        cout<<v[i]<<" ";
    cout<<endl;
}

```

Vetores e matrizes sempre são passados por referência para funções e procedimentos. Dessa forma, as alterações sofridas por um vetor em um subprograma, se refletirá fora

daquele subprograma. Exemplo: o subprograma modifica faz a multiplicação de cada valor do vetor por 2.

```
#include <iostream>
using namespace std;
void saida (int v[], int t){
    int i;
    for (i=0; i<t; i++)
        cout<<v[i]<<" ";
    cout<<endl;
}

void modifica (int v[], int t) {
    int i;
    for (i=0; i<t; i++)
        v[i]*= 2;
}

int main () {
    const int tamanho = 5;
    int i, v[tamanho];

    for (i=0;i<tamanho;i++)
        cin>>v[i];

    saida (v,tamanho);
    modifica(v,tamanho);
    cout<<"Apos subprograma Modifica\n";
    saida(v,tamanho);
    return 0;
}
```

For e For each (Para cada)

Nesse momento, acredito que você já entendeu como transformar um programa escrito com a estrutura de repetição `while` para a estrutura de repetição `for` e vice-versa.

Já deve ter percebido que embora o `for` seja “lindo”, nem sempre ele poderá ser usado.

Além do `for`, há a estrutura `for each`. Esse `for` poderá ser usado quando o vetor estiver preenchido. Sua sintaxe `for (tipo elemento : colecao)` quer dizer para cada elemento da coleção, ou seja, se o vetor estiver vazio, não dará certo.

No exemplo, o último `for` tem a variável `f` do tipo `string`. A cada rodada `f` será um elemento de frutas, ou seja, uma `string`.

<pre>int main () { const int tamanho = 3; int i; string frutas [tamanho]; for (i=0; i<tamanho; i++) cin>>frutas[i]; //imprimindo os dados for (string f : frutas) cout<<"Gosto de "<<f<<endl; return 0; }</pre>	 <pre>uva laranja manga Gosto de uva Gosto de laranja Gosto de manga</pre>
<pre>//outro exemplo: string teste = "Ola, mundo!"; for (char c: teste) { cout << c << " "; }</pre>	
<pre>//outro exemplo float v [5] = {10.3, 56.2, 80, 39.1, -3}; for (float valor : v) cout<<valor<<endl;</pre>	

Vetores de caracteres e strings

- O tipo string, apesar de ser considerado um tipo básico em C++, é na verdade um tipo derivado de dado que possui, internamente, um vetor de caracteres. Assim, independentemente de usarmos o tipo char [] ou o tipo string, podemos tratar strings em C++ exatamente como tratamos um vetor de caracteres.
- As posições de uma string podem ser acessadas como se a string fosse um vetor.
- O operador + concatena duas strings.
- As funções length() e size() retornam o tamanho de uma string.
- A função strlen() retorna o tamanho de um vetor de char.
- A função getline() é usada para ler uma string que possui espaços em branco no meio, por exemplo, Minas Gerais.

1) Faça um programa que dada uma string, somente com letras minúsculas, sem espaços, conta quantas vogais aparecem. Lista de estudos.

Exemplo: abacaxi. Saída: 4

```
int main () {
    string entrada;
    cin>>entrada;
```

```
int cont=0, i, tam=entrada.size();
for (i=0; i<tam; i++)
    if (entrada[i]=='a' or entrada[i]=='e' or entrada[i]=='o' or
        entrada[i]=='u' or entrada[i]=='i')
        cont++;

cout<<cont<<endl;
return 0;
}
```

Busca em Vetores

Quase não fazemos busca!

- buscar um nome na sua lista de contatos
- buscar um livro na biblioteca
- etc

1) Busca sequencial

Não é necessário que o vetor esteja ordenado. Contudo, se ordenado, o desempenho do algoritmo será melhor.

Pode ser implementada de diferentes maneiras mantendo a ideia de busca sequencial.

Problema: Dado um vetor com elementos do tipo caractere, elaborar um algoritmo para realizar essa busca. Mostrar todas as posições do vetor onde o elemento foi encontrado. Caso o elemento não seja encontrado, imprimir -1.

Entradas:

1. O número de caracteres que devem ser lidos.
2. Os caracteres separados por brancos.
3. Um caractere para ser buscado.

Saídas:

1. Os índices de cada elemento do vetor que for igual ao caracter buscado.

Exemplo de entrada:

```
5
A R A R A
A
```

Exemplo de saída:

```
0 2 4
```

```

#include <iostream>
using namespace std;

void saida (int t, char c[]){
    for (int i=0; i<t; i++)
        cout<<c[i]<<" ";
    cout<<endl;
}

void busca (char v[],int t, char caractere) {
    int i=0, achou=-1;
    while (i<t) {
        if (v[i] == caractere){
            achou=i;
            cout<<i<<" ";
        }
        i++;
    }
    if (achou==-1)
        cout<<achou<<endl;
}

int main () {
    int tam,i;
    cin>>tam;
    char v [tam], caractere;

    for (i=0; i<tam; i++)
        cin>>v[i];
    saida(tam,v);

    cin>>caractere;

    busca (v,tam,caractere);
    return 0;
}

```

2) Busca binária

Considerando um vetor ordenado de forma crescente, o elemento procurado é comparado ao elemento do meio do arranjo.

Se igual, busca bem sucedida.

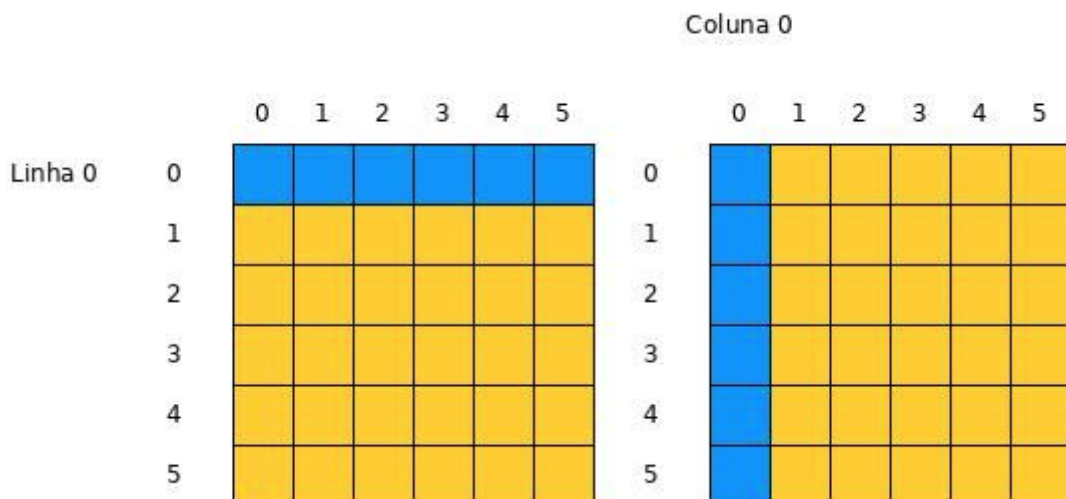
Se menor, busca-se na metade inferior do arranjo.

Se maior, busca-se na metade superior do arranjo.

Toda vez que uma comparação é feita, o número de elementos é cortado pela metade. Por causa da divisão do vetor em duas partes iguais, esse método chama-se busca binária.

Diagram illustrating the first step of binary search. The array is [12, 25, 33, 37, 48, 57, 86, 92]. The search range is from $inf=1$ to $sup=N=8$. The middle element ($meio$) is 37. Since $25 < 37$, the search range is updated to the left half.

Matrices



Diagonal principal							Diagonal secundária						
	0	1	2	3	4	5		0	1	2	3	4	5
0	■						0						■
1		■					1					■	
2			■				2				■		
3				■			3			■			
4					■		4		■				
5						■	5	■					

1) Criação, preenchimento e impressão de matriz.

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

int main(){

    // Criar matriz - declarar e preencher

    int matrizA[3][2]; // 3 linhas e 2 colunas
    cout << "Digite 6 elementos para a preencher a matriz A:" << endl;
    for (int i=0; i<3; i++){
        for (int j=0; j<2; j++){
            cin >> matrizA[i][j];
        }
    }

    // Imprime matriz
    cout << "Matriz A: " << endl;
    for (int i=0; i<3; i++){
        for (int j=0; j<2; j++){
            cout << matrizA[i][j] << " ";
        }
        cout << endl;
    }

    // Criar matriz com dimensões informadas pelo usuário
    int linha, coluna;
    cout << "Digite o número de linhas e colunas da matrizB:" << endl;
    cin >> linha >> coluna;

    //Somente após ler os valores de linha e coluna é que a matriz poderá ser criada
```



```

float matrizB[linha][coluna];
cout << "Digite os elementos para a preencher a matriz B:" << endl;
for (int i=0; i<linha; i++){
    for (int j=0; j<coluna; j++){
        cin >> matrizB[i][j];
    }
}
// Imprime matriz
cout << "Matriz B: " << endl;
for (int i=0; i<linha; i++){
    for (int j=0; j<coluna; j++){
        cout << matrizB[i][j] << " ";
    }
    cout << endl;
}

return 0;
}

```

2) Imprimindo elementos pelas colunas, diagonal principal e diagonal secundária

```

#include <iostream>
using namespace std;
//Número de linhas e colunas da matriz
const int LINHA = 3;
const int COLUNA = 3;

// Preenche matriz
void preenche_matriz( int matriz_nova[LINHA][COLUNA]){
    cout << "Informe " << LINHA*COLUNA << " elementos: " << endl;
    for (int i=0; i < LINHA; i++){
        for (int j=0; j < COLUNA; j++){
            cin >> matriz_nova[i][j];
        }
    }
}

// Imprime coluna
void imprime_coluna(int matriz_nova[LINHA][COLUNA]){
    cout << "Impressão das colunas da matriz: " << endl;
    for (int indice_coluna = 0; indice_coluna < COLUNA; indice_coluna++){
        for (int indice_linha=0; indice_linha < LINHA; indice_linha++){
            cout << matriz_nova[indice_linha][indice_coluna] << endl;
        }
    }
    cout << endl;
}

```

```

    }
}

// Imprime diagonal principal
void diagonal_principal( int matriz_nova[LINHA][COLUNA]){
    cout << "Imprime diagonal principal " << endl;
    for (int indice_linha=0; indice_linha<LINHA; indice_linha++){
        for (int indice_coluna=0; indice_coluna<COLUNA; indice_coluna++){
            if (indice_linha == indice_coluna)
                cout << matriz_nova[indice_linha][indice_coluna] << endl;
        }
    }
}

// Imprime diagonal secundária
void diagonal_secundaria( int matriz_nova[LINHA][COLUNA]){
    cout << "Imprime diagonal secundária " << endl;
    for (int indice_linha=0; indice_linha<LINHA; indice_linha++){
        for (int indice_coluna=0; indice_coluna<COLUNA; indice_coluna++){
            //indice_coluna é calculado em função do número de elementos e do
            indice

            if (indice_coluna == LINHA-1 - indice_linha)
                cout << matriz_nova[indice_linha][indice_coluna] << endl;
        }
    }
}

int main(){
    int matrizM[LINHA][COLUNA];
    preenche_matriz(matrizM);
    // Chama imprime coluna
    imprime_coluna(matrizM);
    // Chama imprime diagonal principal
    diagonal_principal(matrizM);
    // Chama imprime diagonal secundária
    diagonal_secundaria(matrizM);
    cout << endl;
    return 0;
}

```

3) Encontrar o maior elemento da matriz e menor elemento da linha informada pelo usuário

```

#include <iostream>
using namespace std;

int main(){
    int LINHA, COLUNA;
    cout << "Informe o número de linhas e de colunas: " << endl;

```

```

cin >> LINHA >> COLUNA;
int matrizM[LINHA][COLUNA];
//Preenche matriz
cout << "Informe " << LINHA*COLUNA << "valores:" << endl;
for (int i=0; i < LINHA; i++){
    for (int j=0; j < COLUNA; j++){
        cin >> matrizM[i][j];
    }
}

//Encontra maior elemento
int maior=matrizM[0][0];
for (int indice_coluna = 0; indice_coluna < COLUNA; indice_coluna++){
    for (int indice_linha=0; indice_linha < LINHA; indice_linha++){
        if (matrizM[indice_linha][indice_coluna] > maior) {
            maior = matrizM[indice_linha][indice_coluna];
        }
    }
}

cout << "Maior valor da matriz: " << maior << endl;

//Encontra menor elemento da linha informada pelo usuário
int linha_informada;
cout << "Informe a linha para buscar o menor valor:" << endl;
cin >> linha_informada;
int menor=matrizM[linha_informada][0]; // Valor inicial de menor
for (int indice_coluna=1; indice_coluna< COLUNA; indice_coluna++){
    if (matrizM[linha_informada][indice_coluna] < menor) {
        menor = matrizM[linha_informada][indice_coluna];
    }
}
cout << "Menor valor= " << menor << endl;
return 0;
}

```

4) Criar uma matriz de inteiros positivos (pode haver repetição). Buscar o valor informado pelo usuário na matriz substituindo-o por -1. Ao final, imprimir a matriz resultante.

```

#include <iostream>
//Número de linhas e colunas da matriz
const int LINHA = 3; //poderia usar também #define LINHA 4
const int COLUNA = 3; // #define COLUNA 2
using namespace std;
// Preenche matriz

```

```

void preenche_matriz( int matriz_nova[LINHA][COLUNA]){
    cout << "Informe " << LINHA*COLUNA << " elementos inteiros e positivos: " << endl;
    for (int i=0; i<LINHA; i++){
        for (int j=0; j<COLUNA; j++){
            cin >> matriz_nova[i][j];
        }
    }
}

// Imprime matriz
void imprime_matriz( int matriz_nova[LINHA][COLUNA]){
    for (int i=0; i<LINHA; i++){
        for (int j=0; j<COLUNA; j++){
            cout << matriz_nova[i][j] << " ";
        }
        cout << endl;
    }
}

// Outra forma de passar matriz
void imprime_matriz2( int matriz_nova[][COLUNA]){
    for (int i=0; i<LINHA; i++){
        for (int j=0; j<COLUNA; j++){
            cout << matriz_nova[i][j] << " ";
        }
        cout << endl;
    }
}

void altera_elemento(int matriz_nova[][COLUNA], int elemento){
    //Percorre a matriz buscando o elemento a ser substituído por -1
    for (int i=0; i<LINHA; i++){
        for (int j=0; j<COLUNA; j++){
            if (elemento == matriz_nova[i][j]){
                matriz_nova[i][j] = -1;
            }
        }
    }
}

int main(){

    int matrizM[LINHA][COLUNA];
    preenche_matriz(matrizM);
    imprime_matriz(matrizM);
}

```

```
int elemento;  
cout << "Informe o elemento a ser substituído na matriz:" << endl;  
cin >> elemento;  
altera_elemento(matrizM, elemento);  
imprime_matriz2(matrizM);  
  
return 0;  
}
```