BCC702 - Programação de Computadores II

# Arranjos unidimensionais e bidimensionais

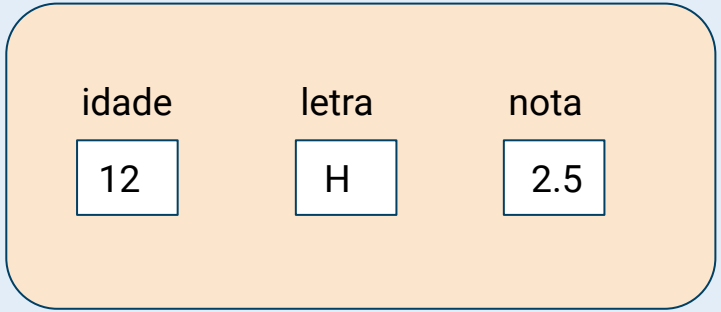
Prof José Romildo Malaquias
Profª Valéria de Carvalho Santos

UFOP

decom
departamento de computação

# Variável Simples

- Cada variável armazena um dado na memória



# Arranjos

- Há situações em que é preciso armazenar um conjunto de dados

- Exemplo: Faça um programa para calcular a média das notas de 50 alunos e imprima as notas que estão acima da média.

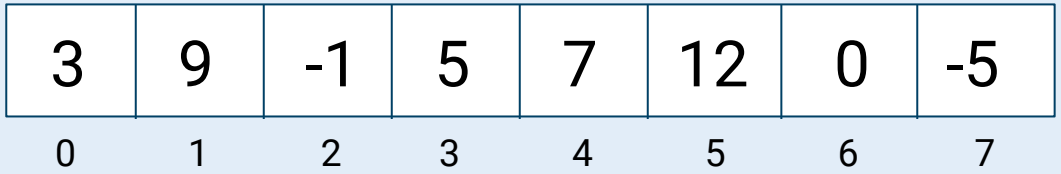
# Arranjos

- É preciso uma estrutura capaz de armazenar um conjunto de dados de uma vez

- **Arranjo**: estrutura que armazena um conjunto de dados do mesmo tipo em posições consecutivas na memória
  - Unidimensionais: **vetores**
  - Bidimensionais: **matrizes**

# Vetores

- Cada elemento é acessado pelo identificador e pela posição do elemento no vetor (**índice**)




- Em C++, índices iniciam sempre em 0 e terminam em *n-1*, sendo *n* a quantidade de elementos do vetor

# Sintaxe

- Declaração:
  - tipo nomeVetor[tamanho];

- Exemplos:
  - int minhasNotas[4];
  - char nome[30];
  - float temperaturas[50];

# Sintaxe

- Para referenciar um elemento específico de um vetor, utiliza-se o nome do vetor, seguido pelo índice (posição) do elemento

- A posição do elemento deve estar delimitada pelos símbolos de colchetes

# Exemplo 1

Declaração

Atribuição

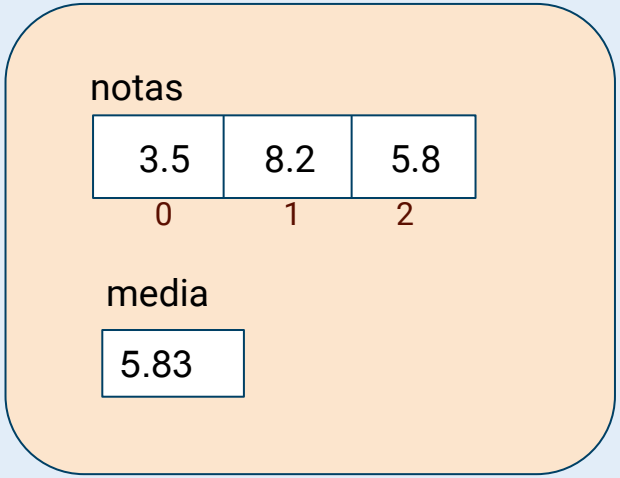
```
#include <iostream>
using namespace std;

int main () {
        double notas[3];

        notas[0] = 3.5;
        notas[1] = 8.2;
        notas[2] = 5.8;

        double media = (notas[0] + notas[1] + notas[2]) / 3;
        cout << “Media = ” << media << endl;
        return 0;
}
```

# Exemplo 2

Declaração com inicialização

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

int main () {
        double notas[3] = {3.5, 8.2, 5.8};

        double media = (notas[0]+notas[1]+notas[2])/3;
        cout << “Media = ” << media <<endl;
        return 0;
}
```

# Exemplo 2



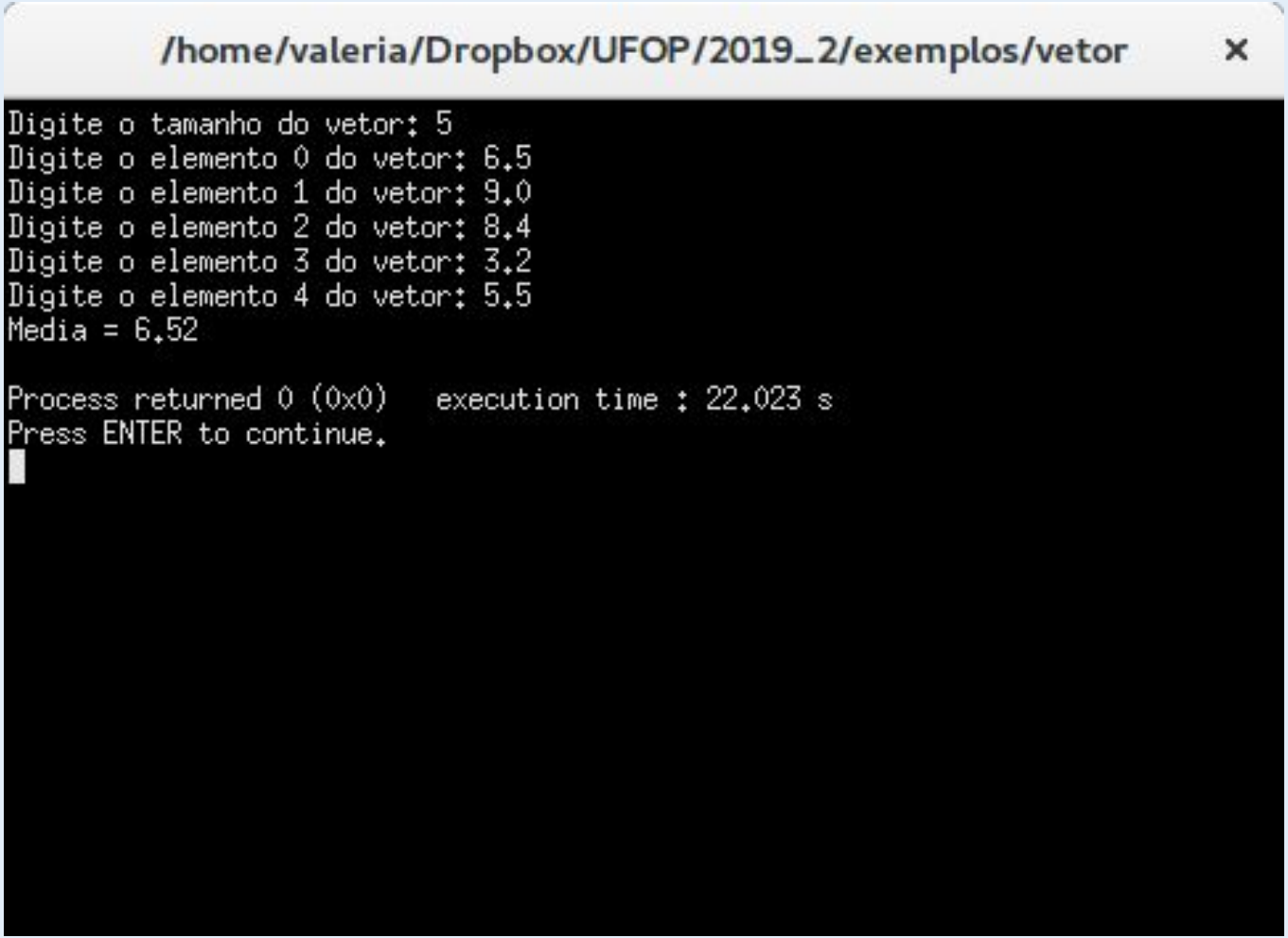

```
#include <iostream>
using namespace std;

int main () {
	double notas[3] = {3.5, 8.2, 5.8};

	double media = (notas[0]+notas[1]+notas[2])/3;
	cout << “Media = ” << media <<endl;
	return 0;
}
```

# Exemplo 2

Atribuindo valores de entrada: só é possível ler um valor por vez!

```
int main () {
	int tamanho;
	cout << “Digite o tamanho do vetor: ”;
	cin >> tamanho;

	double notas[tamanho];

	for(int i = 0; i < tamanho; i++){
		cout << “Digite o elemento ” << i << “ do vetor: ”;
		cin >> notas[i];
	}

	double soma = 0;
	for(int i = 0; i < tamanho; i++)
		soma = soma + notas[i]; // soma += notas[i];

	double media = soma/tamanho;
	cout << “Media = ” << media <<endl;
	return 0;
}
```

# Exemplo 2


/home/valeria/Dropbox/UFOP/2019_2/exemplos/vetor
Digite o tamanho do vetor: 5
Digite o elemento 0 do vetor: 6.5
Digite o elemento 1 do vetor: 9.0
Digite o elemento 2 do vetor: 8.4
Digite o elemento 3 do vetor: 3.2
Digite o elemento 4 do vetor: 5.5
Media = 6.52
Process returned 0 (0x0) execution time : 22.023 s
Press ENTER to continue.

# Exemplo 3

É importante notar que o valor do tamanho do vetor deve ser lido antes da declaração dele.

```
int main () {
	int tamanho;
	cout << “Digite o tamanho do vetor: ”;
	cin >> tamanho;
	double notas[tamanho];

	for(int i = 0; i < tamanho; i++){
		cout << “Digite o elemento ” << i << “ do vetor: ”;
		cin >> notas[i];
	}
	double soma = 0;
	for(int i = 0; i < tamanho; i++){
		soma = soma + notas[i]; // soma += notas[i];
	}
	double media = soma/tamanho;
	cout << “Media = ” << media <<endl;
	return 0;
}
```

# Exercício

- Altere o programa do exemplo anterior para também imprimir as notas que estão acima da média.

# Leitura e escrita dos elementos de um vetor

- Em C++, os elementos de um vetor só podem ser lidos e escrito um de cada vez, acessando cada elemento pela posição.

- Exemplo: Faça um programa para ler um conjunto de N valores inteiros (o valor de N deve ser informado pelo usuário) e armazená-los em um vetor. Depois, percorra o vetor para imprimir na tela os seus elementos, na mesma linha.

```
int main () {
	int n;
	cout << “Digite o tamanho de N: ”;
	cin >> n;
	int vetor[n];

	for(int i = 0; i < n; i++) {
		cout << “Digite o elemento ” << i << “ do vetor: ”;
		cin >> vetor[i];
	}

	cout << “Elementos do vetor:” << endl;
	for(int i = 0; i < n; i++)
		cout << vetor[i] << “ ”;

	return 0;
}
```

```
int main () {
        int n;
        cout << “Digite o tamanho de N: ”;
        cin >> n;
        int vetor[n];

        cin >> vetor;

        cout << “Elementos do vetor:” << endl;
        cout << vetor;

        return 0;
}
```

Não funciona!!!

# Matrizes

- São arranjos homogêneos bidimensionais

- Cada elemento é acessado por um índice que indica a *linha* e outro índice que indica a *coluna*

# Matrizes

$$M_{3x3} = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix}$$

# Matrizes

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 4 | 19 | -9 |
| 1 | -1 | 5 | 8 |
| 2 | 3 | 1 | 11 |

# Sintaxe

- Declaração:
  - tipo nomeMatriz[tamLinha] [tamColuna];

- Exemplos:
  - int matriz[2][3];
  - float temperaturas[10][10];

# Sintaxe

- Cada elemento da matriz é acessado por dois índices entre colchetes da seguinte forma:
  - nomeDaMatriz [posicaoLinha][posicaoColuna]

# Declaração e inicialização

```
int main () {
	int matriz1 [3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

	int matriz2 [3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8} , {9, 10, 11, 12}};

	for (int linha=0; linha<3; linha++) {
		for (int coluna=0; coluna<4; coluna++)
			cout<<matriz1[linha][coluna]<<" ";

		cout << "\n"; //ou cout << endl;
	}
	for (int linha=0; linha<3; linha++) {
		for (int coluna=0; coluna<4; coluna++)
			cout<<matriz2[linha][coluna]<<"\t";

		cout << "\n";
	}
}
```

# Exemplo

- Faça um programa para ler os elementos inteiros de uma matriz 2x3 e calcular e exibir a soma dos elementos da matriz.

# Solução 1

```
int main () {
	int matriz [2][3];

	cout << “Informe os elementos da matriz:” << endl;
	for (int linha=0; linha<2; linha++)
		for (int coluna=0; coluna<3; coluna++)
			cin >> matriz[linha][coluna];

	int soma = 0;
	for (int linha=0; linha<2; linha++)
		for (int coluna=0; coluna<3; coluna++)
			soma += matriz[linha][coluna];

	cout << “Soma=” << soma << endl;

	return 0;
}
```

# Solução 2

```
int main () {
	int matriz [2][3];
	int soma = 0;

	cout << “Informe os elementos da matriz:” << endl;
	for (int linha=0; linha<2; linha++)
		for (int coluna=0; coluna<3; coluna++) {
			cin >> matriz[linha][coluna];
			soma += matriz[linha][coluna];
		}

	cout << “Soma=” << soma << endl;
	return 0;
}
```

# Solução 3

```
#define L 2
#define C 3

int main () {
        int matriz [L][C];
        int soma = 0;

        cout << “Informe os elementos da matriz:” << endl;
        for (int linha=0; linha<L; linha++)
                for (int coluna=0; coluna<C; coluna++) {
                        cin >> matriz[linha][coluna];
                        soma += matriz[linha][coluna];
                }

        cout << “Soma=” << soma << endl;
        return 0;
}
```

Definições de **macros**: representam constantes neste caso

# Solução 4

```
const int L 2;
const int C 3;

int main () {
        int matriz [L][C];
        int soma = 0;

        cout << “Informe os elementos da matriz:” << endl;
        for (int linha=0; linha<L; linha++)
                for (int coluna=0; coluna<C; coluna++) {
                        cin >> matriz[linha][coluna];
                        soma += matriz[linha][coluna];
                }

        cout << “Soma=” << soma << endl;
        return 0;
}
```

Definições de **constantes:** variáveis que não podem ser modificadas

# Exercício

- Faça um programa que lê as dimensões *n* e *m* de uma matriz e seus elementos inteiros. O programa deve exibir qual é a linha em que se encontra o maior elemento da matriz.

# Vetores de caracteres

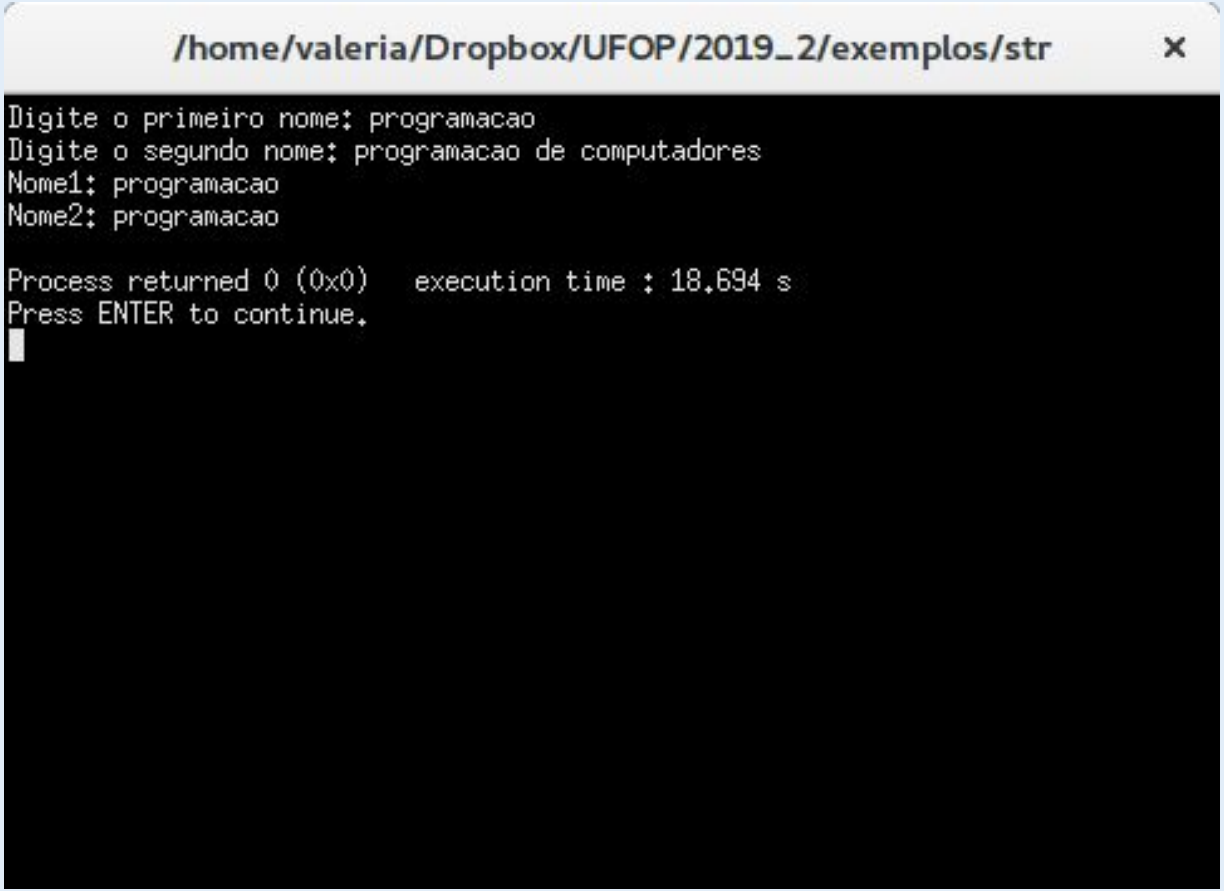
- A biblioteca **string** proporciona o *tipo* **string**, cujo tamanho pode ser alterado durante a execução, além de funções de manipulação

- Leitura
  - O operador **>>** lê até encontrar o espaço branco (espaço, tabulação, final de linha)
  - A função **getline** lê até encontrar um final de linha

# Exemplo *cin*

```
#include <iostream>
#include <string>

using namespace std;

int main () {
        string nome1, nome2;
        cout << “Digite o primeiro nome:”;
        cin >> nome1;
        cout << “Digite o segundo nome:”;
        cin >> nome2;

        cout << “Nome1: ” << nome1 << endl;
        cout << “Nome2: ” << nome2 << endl;

        return 0;
}
```

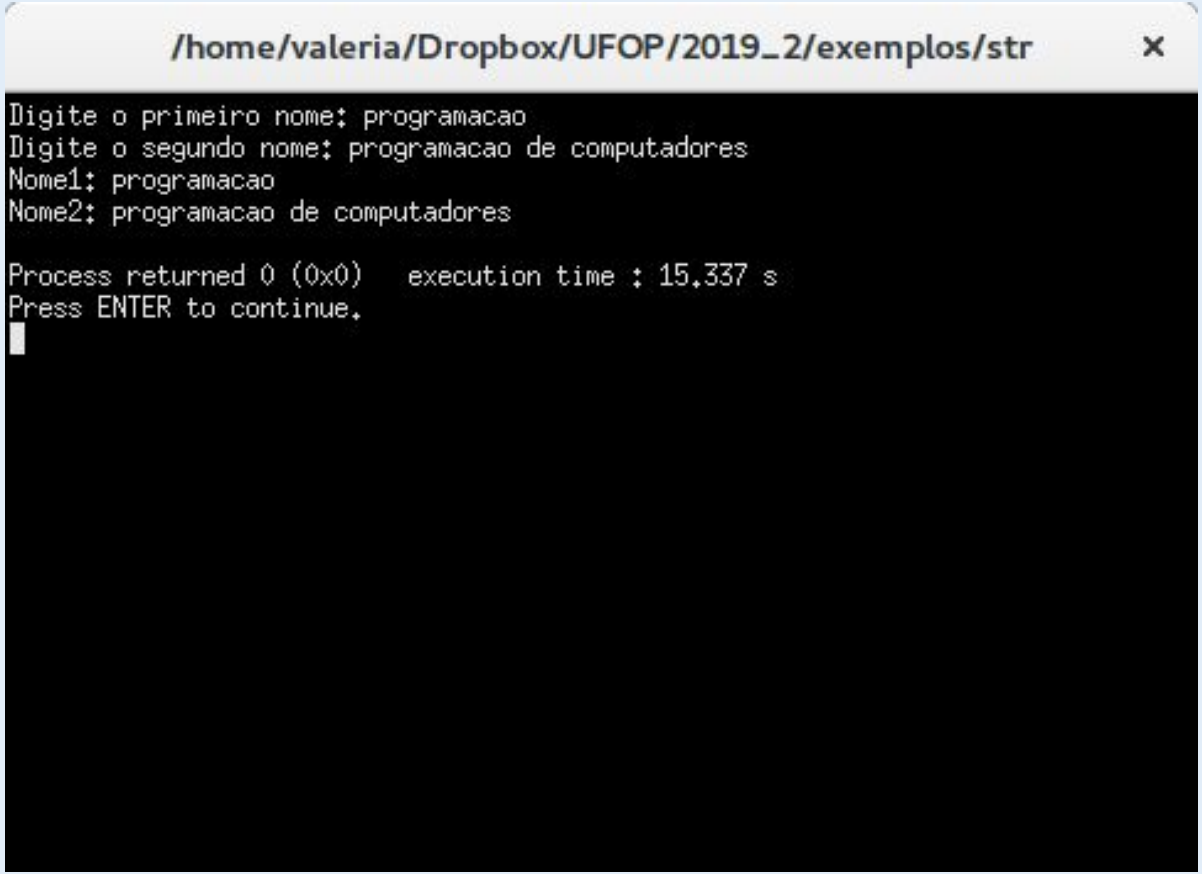
# Exemplo *usando* >>



# Exemplo usando *getline*

```
#include <iostream>
#include <string>

using namespace std;

int main () {
        string nome1, nome2;
        cout << “Digite o primeiro nome:”;
        getline(cin, nome1);
        cout << “Digite o segundo nome:”;
        getline(cin, nome2);

        cout << “Nome1: ” << nome1 << endl;
        cout << “Nome2: ” << nome2 << endl;

        return 0;
}
```

# Exemplo *getline*



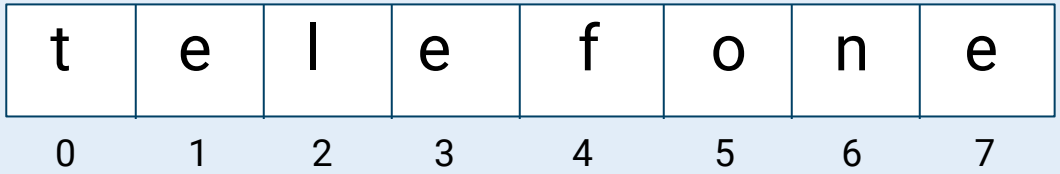
# Operações em strings

- Sejam s, s1 e s2 variáveis declaradas como sendo do tipo string.

- **s.size()** → o número de caracteres em s.
- **s.length()** → o número de caracteres em s.
- **s1 == s2** → verifica a equivalência entre s1 e s2.
- **s1 = s2** → copia a informação de s2 para s1.
- **s = s1 + s2** → o valor de s é a concatenação de s1 e s2.
- **s1 += s2** → s2 é adicionada ao final do valor de s1.

# Acesso por caractere

- Os caracteres de uma string podem ser acessados individualmente pelo **índice**, como nos vetores




- Exemplo: Faça um programa que lê uma palavra e uma letra e conta quantas vezes a letra ocorre na palavra.

# Exemplo

```
#include <iostream>
#include <string>

using namespace std;

int main () {
	char letra;
	string nome;
	cout << "Digite uma letra: ";
	cin >> letra;
	cout << "Digite uma palavra: ";
	cin.ignore(2, '\n');
	getline(cin, nome);

	int cont=0;
	for(int i=0; i<nome.length(); i++)
		if(nome[i] == letra)
			cont++;

	cout << "Quantidade = " << cont;
	return 0;
}
```

# Exemplo

Atenção: quando **>>** é usado antes de ***getline***, o getline captura o ENTER e não funciona corretamente.
Para corrigir isso, é preciso usar a função *ignore* antes do *getline* para descartar o ENTER.

```
#include <iostream>
#include <string>

using namespace std;

int main () {
	char letra;
	string nome;
	cout << "Digite uma letra: ";
	cin >> letra;
	cout << "Digite uma palavra: ";
	cin.ignore(1, '\n');

	int cont=0;
	for (int i=0; i<nome.length(); i++)
		if (nome[i] == letra)
			cont++;

	cout << "Quantidade = " << cont;
	return 0;
}
```

# Exercício

- Faça um programa que lê uma frase e escreve a frase com os caracteres ao contrário.