Declaração de Vetores:

1. A declaração de um vetor começa com a especificação do tipo de dado que o vetor irá armazenar, seguido pelo nome do vetor e o tamanho entre colchetes.
2. Exemplo: int numeros[5]; declara um vetor de inteiros chamado "numeros" com espaço para 5 elementos.

Inicialização de Vetores:

3. Os vetores podem ser inicializados durante a declaração ou em etapas posteriores.
4. Exemplo de inicialização durante a declaração: int numeros[5] = {1, 2, 3, 4, 5};
5. Exemplo de inicialização posterior: numeros[0] = 1; numeros[1] = 2; /* ... */

Entrada e saída de Dados em Vetores:

6. Você pode usar loops para ler dados do usuário e armazená-los em um vetor

```
for (int i = 0; i < 5; i++) {
   cin>>numeros[i];
}
for (int i = 0; i < 5; i++) {
   cout<<numeros[i];
}
```

Lembre-se de que os índices em vetores C começam em 0, ou seja, o primeiro elemento é acessado com [0], o segundo com [1], e assim por diante.

Exercícios de vetor:

1.Crie um vetor com as notas de 10 alunos, as notas serão digitadas pelo usuário do programa. Calcule os seguintes valores:

a. maior valor;
b. menor valor;
c. média das notas;
d. número de alunos de recuperação (notas menores que seis);
e. número de alunos aprovados(nota maior ou igual a seis);

2. Crie e preencha um vetor com 10 valores inteiros, calcule e imprima na tela:

f. a quantidade de números pares;
g. a quantidade de números ímpares;
h. quais os números positivos;
i. quais os números negativos;

3.Crie e preencha dois vetores com 10 valores inteiros em cada um deles.

j. Crie um terceiro vetor onde o valor de uma determinada posição é a soma da respectiva posição do primeiro vetor e do segundo vetor. C=A+B
k. Crie um terceiro vetor onde o valor de uma determinada posição é a diferença da respectiva posição do primeiro vetor e do segundo vetor. C=A-B

4.Crie e preencha dois vetores com 10 valores inteiros e verifique se o mesmo está ordenado.

Declaração de Strings em C++:

- Para declarar uma string em C++, você pode usar a classe string da biblioteca padrão.
- Exemplo: string nome; declara uma string chamada "nome".

Inicialização de Strings em C++:

- Você pode inicializar uma string durante a declaração ou em etapas posteriores.
- Exemplo de inicialização durante a declaração: string saudacao = "Olá, mundo!";
- Exemplo de inicialização posterior: nome = "Alice";

Entrada de Dados em Strings em C++:

- Para ler dados do usuário e armazená-los em uma string em C++, você pode usar a função getline ou operadores como >>.
- Exemplo com std::getline:

```
string frase;
cout << "Digite uma frase: ";
getline(cin, frase);
```

Saída de Dados de Strings em C++:

- Para imprimir uma string em C++, você pode usá-la diretamente em comandos de saída, como cout.
- Exemplo:

```
string saudacao = "Olá, mundo!";
cout << saudacao << endl;
```

Em C++, a classe string da biblioteca padrão fornece vários métodos para manipulação de strings. Aqui estão alguns dos **métodos mais comuns**:

length() ou size(): Retorna o tamanho da string (número de caracteres).

```
string texto = "Hello, World!";
int tamanho = texto.length();
```

empty(): Verifica se a string está vazia (sem caracteres).

```
string texto = "Hello";
bool vazia = texto.empty();
```

clear(): Remove todos os caracteres da string, deixando-a vazia.

```
string texto = "Hello";
texto.clear(); // Agora, texto está vazia.
```

append() ou +=: Adiciona uma string ou caractere ao final da string.

```
string texto = "Hello";
texto.append(", World!"); // Ou: texto += ", World!";
```

insert(): Insere uma string ou caractere em uma posição específica da string.

```
string texto = "Hello!";
texto.insert(5, " World"); // Insere " World" após o índice 5.
string texto = "Hello, World!";
texto.erase(7, 7); // Remove a partir do índice 7 (incluindo) até o índice 14 (excluindo).
```

replace(): Substitui parte da string por outra string.

```
string texto = "Hello, World!";
texto.replace(0, 5, "Hi"); // Substitui "Hello" por "Hi".
```

find() e rfind(): Encontra a primeira ou última ocorrência de uma substring na string.

```
string texto = "Hello, World!";
size_t pos = texto.find("World"); // Encontra "World" na posição 7.
```

substr(): Extrai uma parte da string com base em índices.

```
string texto = "Hello, World!";
string sub = texto.substr(0, 5); // Extrai os primeiros 5 caracteres.
```

c_str(): Converte a string em um array de caracteres C-style (const char*).

```
string texto = "Hello";
const char* cstring = texto.c_str();
```

Exercícios Strings:

1. Leia uma frase digitada pelo usuário e imprima na tela as seguintes informações:
    a. o número de caracteres da frase;
    b. quantas vogais a frase possui;
    c. imprima quantas vezes a letra a aparece na frase;
    d. imprimir a frase ao contrário (de trás p/ frente);
    e. colocar a primeira letra de cada palavra com letra maiúscula;
    f. imprimir quantas palavras a frase tem;
2. Leia no teclado duas strings, e imprima na tela qual palavra aparece primeiro no dicionário, ou se são iguais.
3. Leia uma frase digitada pelo usuário e verifique se a palavra ‘ana’ está inclusa nessa frase.

length() ou size():
    std::string::length() ou std::string::size() retornam o tamanho da string.
empty():
    std::string::empty() verifica se a string está vazia.
clear():
    std::string::clear() remove todos os caracteres da string, deixando-a vazia.
append() ou +=:
    std::string::append() ou o operador += adicionam uma string ou caractere ao final da string.
insert():
    std::string::insert() insere uma string ou caractere em uma posição específica da string.
erase():
    std::string::erase() remove parte da string com base em índices ou intervalo.
replace():
    std::string::replace() substitui parte da string por outra string.
find() e rfind():
    std::string::find() e std::string::rfind() encontram a primeira ou última ocorrência de uma substring na string.
substr():
    std::string::substr() extrai uma parte da string com base em índices.
c_str():
    std::string::c_str() converte a string em um array de caracteres C-style (const char*).
compare():
    std::string::compare() compara duas strings lexicograficamente.
toupper() e tolower():
    std::toupper() converte um caractere em maiúscula.
    std::tolower() converte um caractere em minúscula.
getline():
    std::getline() lê uma linha completa de entrada, incluindo espaços em branco.
stoi(), stod(), stof():
    Funções de conversão para converter strings em números inteiros, números de ponto flutuante ou números de ponto flutuante de precisão simples.
strlen():
    Da biblioteca C (<cstring>), strlen() retorna o comprimento de uma string C-style.
strcpy() e strncpy():
    Da biblioteca C (<cstring>), strcpy() copia uma string para outra.
    strncpy() copia uma parte de uma string para outra.
strcat():
    Da biblioteca C (<cstring>), strcat() concatena duas strings C-style.
strstr():
    Da biblioteca C (<cstring>), strstr() encontra a primeira ocorrência de uma substring em uma string C-style.
strcmp():
    Da biblioteca C (<cstring>), strcmp() compara duas strings C-style.