



Linguagem de Programação

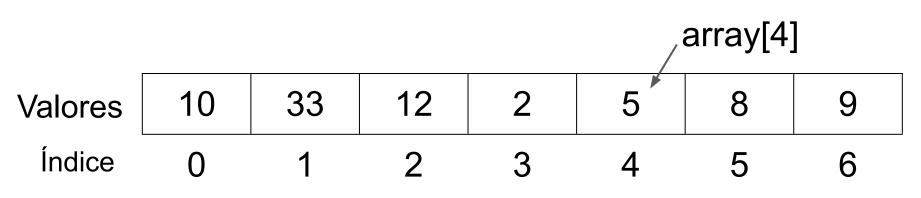
Arrays unidimensionais (vetores)

ECT2303

helton.maia@ufrn.br

Arrays

Um array é uma sequência de valores que compartilham o mesmo nome para armazenamento, e podem ser acessadas utilizando um índice.



Arrays - Motivação

Considere a seguinte situação, você está fazendo uma pesquisa com 100 pessoas e precisa armazenar a idade de cada uma delas.

Como resolver este problema em C++?

Arrays - Motivação

Como resolver este problema em C++?

Em vez de se declarar variáveis individuais, como idade0, idade1, ... e idade99, você pode declarar um array idades[], desta forma, será possível guardar todas as idades na mesma variável.

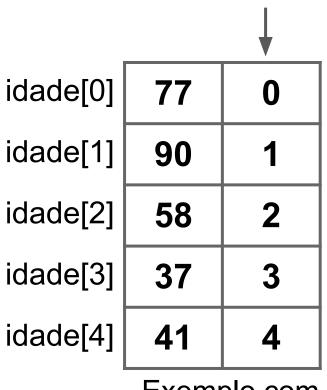
Arrays

- O tamanho de um array é dado por uma expressão ou constante;
- O primeiro elemento de um array é indicado pelo índice 0. Por exemplo, idades[0] armazena o primeiro elemento do array idades, idades[1] o segundo elemento, e assim por diante;
- Na prática, utilizamos uma constante inteira "i" para indicar o índice do array.

Arrays - Armazenamento

 Todas as matrizes consistem em locais de memória contíguos. O endereço menor corresponde ao primeiro elemento, e o endereço maior, ao último elemento;

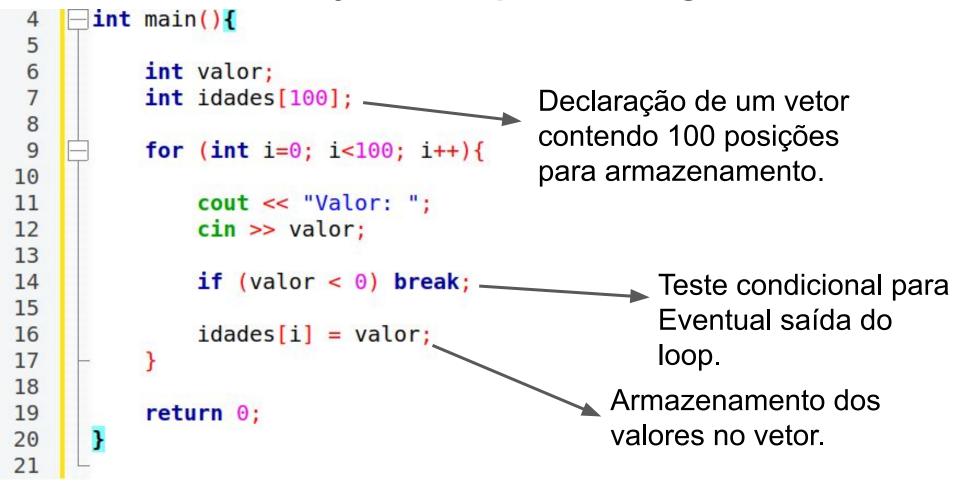
 Um elemento específico em uma matriz é acessado por um índice.



Exemplo com 5 elementos

indices

Arrays - Exemplo de código



Arrays - Definições

- → Conjunto de espaços de memória que se relacionam. Definidos por um nome e um tipo comum entre eles;
- → Para acessar um elemento da array, especificamos seu nome e a posição(índice), onde o elemento se encontra;
- → Os espaços de memória são alocados linearmente. Desta forma, o número da posição serve para calcular o endereço de memória em que o elemento está localizado.

Arrays - Declaração e Inicialização

Sobre os índices:

- Este valor vai dimensionar o vetor e deve ser um inteiro ou mesmo uma expressão inteira equivalente;
- Constantes inteiras.

Declarando a array:

```
tipo identificador [tamanho];
```

Arrays - Declaração e Inicialização

```
#include ...
#define TAM 100
...
float v[TAM];
```

```
const int n = 100;
float v[n];
```

```
Float v[100];//constante inteira
```

Arrays - Declaração e Inicialização

tipo nome[tamanho];

- O dimensionamento (tamanho) do array deve ser uma constante inteira. O total de elementos contidos na array serve para informar ao compilador a quantidade de memória necessária que deve ser reservada;
- Atenção ao se utilizar variáveis.

```
int num;
cin >> num;
float v[num];
```

Obs: Note que uma array é uma estrutura homogênea.

Arrays - Inicializando

```
Inicializa todos os elementos com o valor "zero"
int n[10] = { 0 };

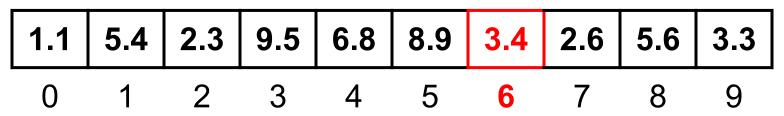
Inicialização individual de cada posição do vetor
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

Caso seja omitido o tamanho do array, será criada uma array de forma a adequar todos os elementos a serem inicializados.

double balance[] = $\{1000.0, 2.0, 3.4, 17.0, 50.0\}$;

Arrays - Acessando elementos

dados



- Para acessar os elementos de vetor ou array, é necessário especificar a sua posição (índice). Ex. dados[6]
- Lembre-se que o primeiro elemento de um vetor no C++ possui índice zero.

Arrays - Exemplos

```
1 #include <iostream>
2 #define TAM 100
4 using namespace std;
6 int main(){
      int array1[TAM]; //declarando array
      cout << array1[0] << endl; // ?
      array1[5] = 11; //atribuição
      cout << array1[5] << endl; // ?
      cin >> array1[6]; //leitura de um valor
      cout << array1[6] << endl; // ?
     //imprimindo a soma de 2 elementos
      cout << "Soma: " << array1[5] + array1[6] << endl;</pre>
19 return 0;
```

Arrays - Consumo de memória

int tamanho=sizeof(<tipo_elementos>)*<tamanho_array>

```
#include <iostream>
2 using namespace std;
4 int main(){
      int x;
      int arrayTest[]={1,2,3,4,5};
      cout << "Tam Bytes x: " << sizeof(x) << endl;</pre>
      cout << "Tam Bytes arrayTest: " <<</pre>
          sizeof(arrayTest) << endl;</pre>
      return 0;
```

Arrays - Em estruturas de repetição

```
int main () {
   int id , dados[10];
//Leitura de dados
for ( id = 0; id < 10; id ++) {
   cin >> dados [id];
//Imprimindo valores do vetor
for ( id = 0; id <= 10; id ++) {
   cout << dados [id];</pre>
return 0;
```

Atenção: o C++ não verifica os limites da array. Tenha cuidado para não sobrepor dados da memória!

O que acontece?

Arrays - Exercício

Escreva um programa que armazena inicialmente 5 elementos. Depois, verifica o maior valor entre estes elementos, e imprime o resultado na tela.

?