Residência em Software

Linguagem C++: Arrays

Professores:

Álvaro Coelho, Edgar Alexander, Esbel Valero e Hélder Almeida

INSTITUIÇÃO EXECUTORA: cepedi, UESC

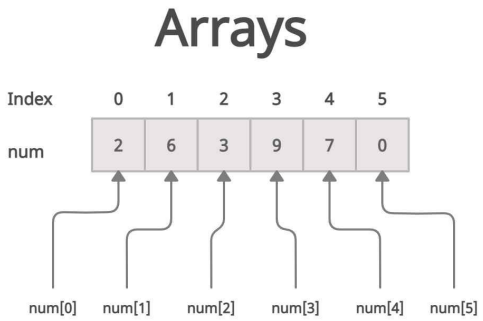
COORDENADORA: MCTI Futuro, Softex

APOIO: Ministério da Ciência, Tecnologia e Inovação, Governo Federal Brasil União e Reconstrução

Residência em Software

Arrays

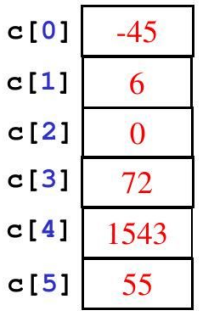
- Um array geralmente é associado a uma lista ou conjunto de elementos similares;
- Exemplos:
  - Os números inteiros de 1..10;
  - As notas de uma turma em uma disciplina;
  - Os resultados de uma enquête sobre a cantina da UESC;
  - Preços de venda dos produtos de uma loja;
- Arrays:
  - Itens de dados do mesmo tipo, relacionados entre se;
  - Coleção de variáveis do mesmo tipo que é referenciado por um nome comum.

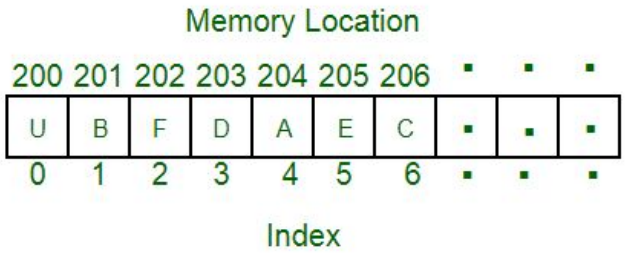


Arrays

- Os arrays são entidades estáticas, i.e. permanecem do mesmo tamanho ao longo da execução do programa;
- Um array é um grupo de locais de memória relacionados pelo fato que tem o mesmo nome e o mesmo tipo;
- Os elementos do array ocupam posições contíguas na memória;
- O endereço mais baixo corresponde ao primeiro elemento e o mais alto ao último;

c - nome do array
- valores do array
- índice do array





Residência em Software

Indexação do Array

- Para fazer referência a um elemento no array precisamos o nome do array e a posição de aquele elemento no array;

```
c[0], a[21]
```

- A posição de um elemento no array é formalmente conhecida como índice;
- Um índice deve ser um inteiro ou uma expressão inteira;
- Para um array de N elementos os índice variam de 0 a (N-1),

Residência em Software

Declaração de Array

- Ao declarar um array reservamos espaço na memória para ele;
- Desta forma deve-se especificar o tipo e a quantidade de elementos;

```
tipo nome_do_array[tamanho]
```

- Exemplos:

```
int c[6];
float b[100];
char s[25];
```

- A linguagem C++ não incorpora verificação de índices, utilizar um valor do índice fora do array não gera nenhuma mensagem de erro;

Residência em Software

Exemplo

```
1   #include <iostream>
2   using namespace std;
3
4   int main(void)
5   {
6       int n[10];
7       for(int i = 0; i < 10; i++)
8           n[i] = i;
9
10      cout << "Elemento" << "\t" << "Índice" << "\t" << "Valor" << endl;
11      for(int i = 0; i < 10; i++)
12          cout << i+1 << "\t\t" << i << "\t" << n[i] << endl;
13
14      // acessando um elemento fora do limite do array
15      cout << n[10] << endl;
16      return 0;
17  }
```

```
Elemento        Índice  Valor
1               0       0
2               1       1
3               2       2
4               3       3
5               4       4
6               5       5
7               6       6
8               7       7
9               8       8
10              9       9
648282120
```

Residência em Software

Inicializando Arrays

- Da mesma forma que as variáveis, os elementos de um arrays podem ser inicializados na declaração do array;
- Para inicializar um arrays utilizamos um sinal de igual e uma lista de inicializadores separados por vírgulas, entre chaves;

```
int n[5] = {1, 2, 3, 4, 5};
char pal[6] = {'b','r','a','s','i','l'};
```

- Se houver menos inicializadores que o número de elementos do array, os elementos restantes são inicializados com zero;

```
float num[10] = {1.0, 2.0, 3.0};
int n[5] = {};
```

Residência em Software

Inicializando Arrays

- Se houver mais inicializadores que termos no array teremos um erro de compilação;

```
int n[2] = {1, 2, 3};
```

- Se o tamanho do array for omitido na declaração, o número de elementos será igual ao número de inicializadores;

```
float num[] = {1.0, 2.0, 3.0};
```

- Que acontece se utilizarmos os elementos de um array sem ter inicializado eles?

Residência em Software

Exemplo

```
i       n[i]    cont[i] num[i]
0       1       0       1
1       2       0       2
2       3       0       3
3       4       0       0
4       5       0       0
brasil
```

```
1   #include <iostream>
2   using namespace std;
3
4   int main(void)
5   {
6       int n[5] = {1, 2, 3, 4, 5};
7       char pal[6] = {'b','r','a','s','i','l'};
8       float num[10] = {1.0, 2.0, 3.0};
9       int cont[5] = {};
10      int i = 0;
11      cout << "i" << "\t" << "n[i]" << "\t"
12           << "cont[i]" << "\t" << "num[i]" << endl;
13      while(i < 5){
14          cout << i << "\t"
15               << n[i] << "\t"
16               << cont[i] << "\t"
17               << num[i] << endl;
18          i++;
19      }
20      for(i = 0; i < 6; i++)
21          cout << pal[i];
22
23      cout << endl;
24      return 0;
25  }
```

Residência em Software

Armazenamento de um Array

- A quantidade de memória ocupada por um array é reservada pelo compilador no ato de compilação do programa;
- A quantidade memória para armazenar um array é alocada durante a compilação;
- A quantidade de memória alocada dependerá do número de elementos e o tipo de dado dos elementos;
- Podemos mostrar a quantidade de memória ocupada pelo array utilizando o operador `sizeof(nome_do_array)`

Residência em Software

Exemplo

```
1   #include <iostream>
2   using namespace std;
3
4   int main(void)
5   {
6       double val[5] = {1.5, 2.4, 3.3, 4.2, 5.1};
7       double soma = 0;
8       for(int i = 0; i < 5; i++)
9           soma += val[i];
10
11      cout << "A soma dos elementos do vetor val eh: " << soma << endl;
12      cout << "O tamanho do vetor val eh: " << sizeof(val) << endl;
13      return 0;
14  }
```

```
A soma dos elementos do vetor val eh: 16.5
O tamanho do vetor val eh: 40
```

Residência em Software

Arrays

- Os arrays do **C++** são um recurso de baixo nível que devem ser usados, principalmente, na implementação de estruturas de dados, como na classe vector, que estudaremos mais adiante;
- IMPORTANTE: Não há atribuição de arrays, e o nome do mesmo é convertido implicitamente em um ponteiro para seu primeiro elemento;

```
double val[5] = {1.5, 2.4, 3.3, 4.2, 5.1};
//double x[5] = val;
double x[5], y[5];
//y = val; // erro: a expressão deve ser um lvalue modificável
for(int i = 0; i < 5; i++)
{
    x[i] = val[i];
    y[i] = val[i];
}
```

Residência em Software

Arrays

- Um dos tipos de arrays mais amplamente usados é um array de caracteres terminado em zero.
- É assim que **C** armazena strings, então um array de `char` terminado em zero é frequentemente chamado de **string estilo C**.
- Literais de cadeia de caracteres **C++** seguem essa convenção.

Residência em Software

Que são Srtings?

- Strings (cadeias de caracteres): são um conjunto de caracteres que podem ser tratados como uma unidade simples;
- Várias entidades do dia a dia são representados computacionalmente utilizando strings:
  - nomes;
  - endereços;
  - números de telefones;
  - CPF;
  - ....

Residência em Software

Que são Srtings?

- Uma string pode incluir caracteres alfanuméricos (letras ou dígitos), e caracteres especiais (+, -, *, _, $, @, #);
- Dados de tipo strings são utilizados em quase todas as aplicações computacionais:
  - editores de texto;
  - bancos de dados;
  - aplicações simples;
  - ...
- A diferença de outras linguagens de programação, a linguagem **C++** não fornece um tipo string como tipo de dado básico;

Residência em Software

Como Tratar Strings em C++

- Para manipular strings podemos utilizar arrays unidimensionais de tipo caracter;
  - STRING = conjunto de caracteres + caractere nulo;
  - caractere nulo = `'\0'`, é utilizado para indicar o final da string;
- Strings literais ou constantes de strings são escritas em C++ utilizando aspas duplas:
  - “Paulo P. Silva”
  - “Rua Alberto Rangel, s/n, Vila Nova, RJ”
  - “456.789.534-02”

Residência em Software

Como Tratar Strings estilo C

- Declaração de uma string:

```
char cor[] = “azul”;
char cor[] = {'a', 'z', 'u', 'l', '\0'};
```

- Em **C++**, 'a' é diferente de “a”:
  - `‘a’`: representa o caracter `a` minúsculo;
  - `“a”`: representa a string contendo apenas um caracter, ou seja. equivale à declaração:

```
char a[2] = {‘a’, ‘\0’}; -> char a[]= ”a”
```

Residência em Software

Como Tratar Strings estilo C

- Declaração de uma string:

```
char cor[] = “azul”;
char cor[] = {'a', 'z', 'u', 'l', '\0'};
```

- Em **C++**, 'a' é diferente de “a”:
  - `‘a’`: representa o caracter `a` minúsculo;
  - `“a”`: representa a string contendo apenas um caracter, ou seja. equivale à declaração:

```
char a[2] = {‘a’, ‘\0’}; -> char a[]= ”a”
```

Residência em Software

Exemplo

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

int main(void)
{
    char word[6] = {'H', 'e', 'l', 'l', 'o'};
    char name[30];
    cout << "Enter your name: ";
    cin >> name;
    cout << word << ", " << name << endl;
    return 0;
}
```

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

int main(void)
{
    char word[] = "Hello";
    char name[30];
    cout << "Enter your name: ";
    cin.getline(name, 30);
    cout << word << ", " << name << endl;
    return 0;
}
```

Residência em Software

Como Tratar Strings estilo C

- O caracteres '\0' sinaliza o final da string e pode ser utilizado para percorrer a mesma.

```
1   #include <iostream>
2   using namespace std;
3
4   int main(void)
5   {
6       char word[100];
7       cout << "Entre com uma frase: ";
8       cin.getline(word, 100);
9       int i = 0;
10      while(word[i] != '\0')
11          i++;
12      cout << "A frase tem " << i << " caracteres." << endl;
13      return 0;
14  }
```

Residência em Software

Como Tratar Strings estilo C

- C++ inclui um conjunto de funções para tratar strings estilo C.

```
1   #include <iostream>
2   #include <cstring>
3   using namespace std;
4
5   int main(void)
6   {
7       char word[100];
8       cout << "Entre com uma frase: ";
9       cin.getline(word, 100);
10      int i = strlen(word);
11      cout << "A frase tem " << i << " caracteres." << endl;
12      return 0;
13  }
```

Residência em Software

Funções da Biblioteca `cstring`

- `char* strcpy(char s1[], char s2[])`: copia a `string s2` para o array `s1` e retorna um ponteiro a `s1`;
- `char * strcat(char s1[], char s2[])`: anexa a `string s2` ao array `s1`. O primeiro caractere de `s2` é sobrescrito ao caractere `'\0'` de `s1`. Um ponteiro para `s1` é retornado;
- `int strcmp(char s1[], char s2[])`: retorna `0` se a `string s2` é igual a `string s1`; menor que `0` se `s1<s2` e maior que `0` se `s1>s2`;
- `int strlen(char s1[])`: determina o comprimento da `string s1`. Retorna o número de caracteres que antecedem ao caractere `'\0'`.

Residência em Software

Arrays Multidimensionais

- Os arrays em C podem ser multidimensionais;
- Um array com mais de um índice é chamado array multidimensional;
- Arrays de dois índices ou bidimensionais são usualmente utilizados para representar tabelas de valores do mesmo tipo;
- Exemplos:
  - Notas de uma turma em uma disciplina considerando várias avaliações;
  - Resultados de um campeonato de futebol (equipes, rodadas, pontos);
  - Estoque de uma loja de roupas (tamanho, cores). Como considerar vários produtos?

Residência em Software

Array Bidimensional

	Coluna 0	Coluna 1	Coluna 2
Linha 0	a[0][0]	a[0][1]	a[0][2]
Linha 1	a[1][0]	a[1][1]	a[1][2]
Linha 2	a[2][0]	a[2][1]	a[2][2]
Linha 3	a[3][0]	a[3][1]	a[3][2]

```
nome do array
índice da linha
índice da coluna
```

Residência em Software

Array Bidimensional

	Avaliação 0	Avaliação 1	Avaliação 2
Aluno 0	5.3	8.3	7.1
Aluno 1	4.2	6.8	7.3
Aluno 2	6.0	5.4	1.0
Aluno 3	9.7	10.0	9.6
Aluno 4	2.1	5.8	7.9

Residência em Software

Declaração do Array

- Sintaxe:

```
tipo nome_matriz[num_linhas][num_colunas];
```

- Exemplos:

```
float n[5][3];

int a[4][4];

char ch[10][5];
```

Residência em Software

Inicialização de Arrays

- Ao igual que arrays unidimensionais, matrizes podem ser inicializadas na declaração utilizando uma lista de inicializadores;
- Exemplos:

```
int n[3][3] = {{0,1,2},{1,2,3},{2,3,4}};

float a[4][4] = {{1,2,3,4},{5,6},{6}};

char c[2][2] = {{'a','b'},{'c','d'}};
```

Residência em Software

Exemplo

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4   #define NLIN 5
5   #define NCOL 3
6
7   int main(void)
8   {
9       float notas[NLIN][NCOL] = { { 5.3, 8.3, 7.1 },
10                                  { 4.2, 6.8, 7.3 },
11                                  { 6.0, 5.4, 1.0 },
12                                  { 9.7, 10.0, 9.6},
13                                  { 2.1, 5.8, 7.9 } };
14      int i, j;
15      cout << fixed;
16      cout << setprecision(1);
17      for(i = 0; i < NLIN; i++)
18      {
19          for(j = 0; j < NCOL; j++)
20              cout << notas[i][j] << "\t";
21          cout << endl;
22      }
23      return 0;
24  }
```

```
5.3     8.3     7.1
4.2     6.8     7.3
6.0     5.4     1.0
9.7     10.0    9.6
2.1     5.8     7.9
```

Residência em Software

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
#define NLIN 5  // 5 alunos
#define NCOL 3  // 3 notas por aluno

int main(void)
{
    float notas[NLIN][NCOL] = { { 5.3, 8.3, 7.1 },
                                { 4.2, 6.8, 7.3 },
                                { 6.0, 5.4, 1.0 },
                                { 9.7, 10.0, 9.6},
                                { 2.1, 5.8, 7.9 } };
    float mediaAluno[NLIN] = {0.0}, mediaAvalia[NCOL] = {0.0};
    cout << fixed;
    cout << setprecision(1);
    for(int i = 0; i < NLIN; i++){
        cout << "Notas do aluno " << i+1 << ": ";
        for(int j = 0; j < NCOL; j++){
            cout << notas[i][j] << "\t";
            mediaAluno[i] += notas[i][j];
            mediaAvalia[j] += notas[i][j];
        }
        mediaAluno[i] /= NCOL;
        cout << "Média do aluno " << i+1 << ": " << mediaAluno[i] << endl;
    }
    for(int j = 0; j < NCOL; j++){
        mediaAvalia[j] /= NLIN;
        cout << "Média da turma na avaliação " << j+1 << ": " << mediaAvalia[j] << endl;
    }
    return 0;
}
```

```
Notas do aluno 1: 5.3   8.3     7.1     Média do aluno 1: 6.9
Notas do aluno 2: 4.2   6.8     7.3     Média do aluno 2: 6.1
Notas do aluno 3: 6.0   5.4     1.0     Média do aluno 3: 4.1
Notas do aluno 4: 9.7   10.0    9.6     Média do aluno 4: 9.8
Notas do aluno 5: 2.1   5.8     7.9     Média do aluno 5: 5.3
Média da turma na avaliação 1: 5.5
Média da turma na avaliação 2: 7.3
Média da turma na avaliação 3: 6.6
```

Residência em Software

Exemplo

```
./"arrayBiExample_03" --with 640 --hight 480 --cmap RGB --fps 30
O nome do programa: ./arrayBiExample_03
Quantidade de argumentos: 9
Os argumentos passados são:
argv[1]: --with
argv[2]: 640
argv[3]: --hight
argv[4]: 480
argv[5]: --cmap
argv[6]: RGB
argv[7]: --fps
argv[8]: 30
```

```
1   #include <iostream>
2   #include <iomanip>
3   using namespace std;
4
5   int main(int argc, char* argv[])
6   {
7       cout << "O nome do programa: " << argv[0] << endl;
8       cout << "Quantidade de argumentos: " << argc << endl;
9
10      if (argc == 1)
11          cout << "Nenhum argumento passado." << endl;
12      else
13      {
14          cout << "Os argumentos passados são: " << endl;
15          for (int i = 1; i < argc; i++)
16              cout << "argv[" << i << "]: " << argv[i] << endl;
17      }
18      return 0;
19  }
```