

Linguagens de Programação

Prof. Miguel Elias Mitre Campista

<http://www.gta.ufrj.br/~miguel>

Parte IV

Introdução à Programação em C++
(Continuação)

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Relembrando da Última Aula...

- Arrays
- Mais exemplos de programação orientada a objetos...

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Ponteiros

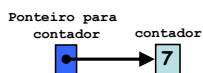
- Poderosos, mas difíceis de utilizar
- Podem ser usados para fazer passagem de parâmetro por referência
 - Podem ser utilizadas para gerenciar estruturas de dados dinâmicas
 - Aumentam e diminuem
- Aproximam o relacionamento entre arrays e strings

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Declaração e Inicialização de Variáveis Ponteiros

- Variáveis ponteiros
 - Contêm endereços de memória como valores
 - Normalmente, variáveis contêm valores específicos
 - Referência direta
- Ponteiros contêm endereços de variáveis que possuem valores específicos
 - Referência indireta



Declaração e Inicialização de Variáveis Ponteiros

- Indireção
 - Referência de valor por ponteiro
- Declaração de ponteiro
 - * indica que a variável é um ponteiro
 - `int *myPtr;` (declara ponteiro para int, ponteiro do tipo int *)
 - Múltiplos ponteiros requerem múltiplos asteriscos
 - `int *myPtr1, *myPtr2;`
 - `int **myPtrtoPtr1;`



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Declaração e Inicialização de Variáveis Ponteiros

- Pode declarar ponteiros para qualquer tipo de dados
- Inicialização de ponteiro
 - Inicializado com 0, NULL, ou endereço
 - 0 ou NULL aponta para nada

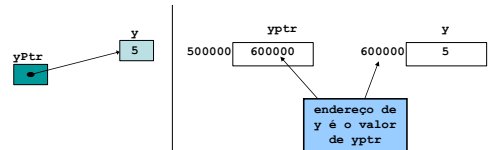
Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Operadores Ponteiros

- & (endereço do operador)
 - Retorna endereço de memória do operando
 - Ex:


```
int y = 5;
int *yPtr;
yPtr = &y; // yPtr recebe endereço de y
```
 - yPtr "aponta para" y



Operadores Ponteiros

- * (operador de indireção)
 - Retorna sinônimo para objeto para o qual o operando ponteiro aponta
 - *yPtr retorna y, porque yPtr aponta para y
 - Ponteiro de indireção é lvalue (valor à esquerda)
 - *yPtr = 9; // atribui 9 para y
- * e & são opostos entre si

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Primeiro Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 1
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int a; // É um inteiro
    int *aPtr; // É um ponteiro para um inteiro

    a = 7;
    aPtr = &a;

    cout << "O endereço de a eh " << &a
        << "\nO valor de aPtr eh " << aPtr;

    cout << "\nO valor de a eh " << a
        << "\nO valor de *aPtr eh " << *aPtr;

    cout << "\nMostrando que * e & são opostos entre si."
        << "\n*aPtr = " << *aPtr
        << "\n&aPtr = " << &aPtr << endl;

    return 0;
}
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Primeiro Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 1
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;
```

```
shell>$ g++ -Wall exemplo.cpp -o ex1
shell>$ ./ex1
O endereço de a eh 0x28ff44
O valor de aPtr eh 0x28ff44

O valor de a eh 7
O valor de *aPtr eh 7
```

```
Mostrando que * e & são opostos entre si
&aPtr = 0x28ff44
&&aPtr = 0x28ff44
shell>$
```

```
return 0;
}
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Chamada de Funções por Referência

- Três maneiras de passar argumentos para funções
 - Passagem por valor
 - Passagem por referência com ponteiros como argumentos
 - Passagem por referência com referências como argumentos
- return pode retornar um valor da função
- Argumentos passados para a função usando referências como argumentos
 - Modifica valores originais de argumentos
 - Mais de um valor "retornado"

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Chamada de Funções por Referência

- Passagem por referência com ponteiros como argumentos
 - Passagem por referência
 - Usa ponteiros e operador de indireção
 - Passagem de endereço do argumento usando o operador &
 - Arrays não são passados com & porque o nome do array já é um ponteiro
 - * operador usado como alias/apelido da variável dentro da função

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Segundo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 2
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int cubeByValue(int);

int main() {
    int number = 5;

    cout << "O valor original de number eh " << number;

    // Passagem de number por valor
    number = cubeByValue(number);

    cout << "\n\nO novo valor de number eh " << number << endl;

    return 0;
}

int cubeByValue(int n) { return n * n * n; }
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Segundo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 2
 * Autor: Miguel Campista
 */
#include <iostream>
```

```
shell>$ g++ -Wall exemplo.cpp -o ex2
shell>$ ./ex2
O valor original de number eh 5
O novo valor de number eh 125
shell>$
```

```
// Passagem de number por valor
number = cubeByValue(number);

cout << "\n\nO novo valor de number eh " << number << endl;

system("PAUSE");
return 0;
}

int cubeByValue(int n) { return n * n * n; }
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Segundo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 2
 * Autor: Miguel Campista
 */
#include <iostream>
```

```
shell>$ g++ -Wall exemplo.cpp -o ex2
shell>$ ./ex2
O valor original de number eh 5
O novo valor de number eh 125
shell>$
```

Como ficaria se a passagem de parâmetro fosse por referência?

```
int cubeByValue(int n) { return n * n * n; }
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Terceiro Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 3
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

void cubeByReference(int *);

int main() {
    int number = 5;

    cout << "O valor original de number eh " << number;

    // Passagem de number por valor
    cubeByReference(&number);

    cout << "\n\nO novo valor de number eh " << number << endl;

    return 0;
}

void cubeByReference(int *nPtz) { *nPtz = (*nPtz) * (*nPtz) * (*nPtz); }
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Terceiro Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 3
 * Autor: Miguel Campista
 */
#include <iostream>
```

```
shell>$ g++ -Wall exemplo.cpp -o ex3
shell>$ ./ex3
O valor original de number eh 5
O novo valor de number eh 125
shell>$
```

```
// Passagem de number por valor
cubeByReference(&number);

cout << "\n\nO novo valor de number eh " << number << endl;

return 0;
}

void cubeByReference(int *nPtz) { *nPtz = (*nPtz) * (*nPtz) * (*nPtz); }
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Usando const com Ponteiros

- Qualificador `const`
 - Valor da variável não deve ser modificado
 - `const` usado quando a função não precisa mudar a variável
- Princípio do menor privilégio
 - Garante a função acesso suficiente para realizar a tarefa, mas nada além disso

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Usando const com Ponteiros

- Quatro maneiras para passar o ponteiro para a função
 - Pontoeiro não constante para dado não constante
 - Quantidade maior de acesso
 - Pontoeiro não constante para dado constante
 - Pontoeiro constante para dado não constante
 - Pontoeiro constante para dado constante
 - Quantidade menor de acesso

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quarto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 4
 * Autor: Miguel Campista
 */
#include <iostream>
#include <ctype>

void convertToUpperCase(char *);

using namespace std;

int main() {
    char phrase [] = "caracteres e $32,99";

    cout << "A frase antes da conversao eh: " << phrase;
    convertToUpperCase(phrase);
    cout << "\n\nA frase depois da conversao eh: " << phrase << endl;

    return 0;
}

void convertToUpperCase(char *sPtr) {
    while (*sPtr != '\0') {
        if (islower(*sPtr))
            *sPtr = toupper(*sPtr);
        sPtr++;
    }
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quarto Exemplo Usando Ponteiros em C++

Pontoeiro não constante para dado não constante

```
/*
 * Aula 8 - Exemplo 4
 * Autor: Miguel Campista
 */
#include <iostream>
#include <ctype>

void convertToUpperCase(char *);

using namespace std;

int main() {
    char phrase [] = "caracteres e $32,99";

    cout << "A frase antes da conversao eh: " << phrase;
    convertToUpperCase(phrase);
    cout << "\n\nA frase depois da conversao eh: " << phrase << endl;

    return 0;
}

void convertToUpperCase(char *sPtr) {
    while (*sPtr != '\0') {
        if (islower(*sPtr))
            *sPtr = toupper(*sPtr);
        sPtr++;
    }
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quarto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 4
 * Autor: Miguel Campista
 */
#include <iostream>
#include <ctype>
```

```
shell>$ g++ -Wall exemplo.cpp -o ex4
```

```
shell>$ ./ex4
```

```
A frase antes da conversao eh: caracteres e $32,99
```

```
A frase depois da conversao eh: CARACTERES E $32,99
```

```
shell>$
```

```
    cout << "\n\nA frase depois da conversao eh: " << phrase << endl;

    return 0;
}

void convertToUpperCase(char *sPtr) {
    while (*sPtr != '\0') {
        if (islower(*sPtr))
            *sPtr = toupper(*sPtr);
        sPtr++;
    }
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quinto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 5
 * Autor: Miguel Campista
 */
#include <iostream>
```

```
void printCharacteres(const char *);
```

```
using namespace std;
```

```
int main() {
    char phrase [] = "caracteres e $32,99";

    cout << "A frase antes da conversao eh: " << phrase << endl;
    printCharacteres(phrase);
    cout << endl;

    return 0;
}
```

```
void printCharacteres(const char *sPtr) {
    for (; *sPtr != '\0'; sPtr++)
        cout << *sPtr;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Quinto Exemplo Usando Ponteiros em C++

Ponteiro não constante para dado constante

```
/*
#include <iostream>

void printCaracteres(const char *);

using namespace std;

int main() {
    char phrase [] = "caracteres e $32,99";

    cout << "A frase antes da conversao eh: " << phrase << endl;
    printCaracteres(phrase);
    cout << endl;

    return 0;
}

void printCaracteres(const char *sPtr) {
    for ( ; *sPtr != '\0'; sPtr++)
        cout << *sPtr;
}
*/
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Quinto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 5
 * Autor: Miguel Campista
 */
#include <iostream>
```

```
shell> g++ -Wall exemplo.cpp -o ex5
```

```
shell> ./ex5
```

```
A frase antes da conversao eh: caracteres e $32,99
caracteres e $32,99
```

```
shell>
```

```
cout << "A frase antes da conversao eh: " << phrase << endl;
printCaracteres(phrase);
cout << endl;

return 0;
}

void printCaracteres(const char *sPtr) {
    for ( ; *sPtr != '\0'; sPtr++)
        cout << *sPtr;
}
}
*/
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Sexto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 6
 * Autor: Miguel Campista
 */
#include <iostream>

void f(const int *);

using namespace std;

int main() {
    int y;

    f(4y); // Tenta modificação ilegal

    return 0;
}

void f(const int *xPtr) {
    *xPtr = 100;
}
*/
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Sexto Exemplo Usando Ponteiros em C++

**Como é feita a passagem de parâmetro?
O programa está correto?**

```
#include <iostream>

void f(const int *);

using namespace std;

int main() {
    int y;

    f(4y); // Tenta modificação ilegal

    return 0;
}

void f(const int *xPtr) {
    *xPtr = 100;
}
*/
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Sexto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 6
 * Autor: Miguel Campista
 */
#include <iostream>

void f(const int *);

using namespace std;

int main() {
    int y;

    f(4y); // Tenta modificação ilegal

    return 0;
}

void f(const int *xPtr) {
    *xPtr = 100;
}
*/
```

```
21 C:\Users\Miguel\Documents\UFRJ\... In function 'void f(const int*)':
C:\Users\Miguel\Documents\UFRJ\... assignment of read-only location
C:\Users\Miguel\Documents\UFRJ\... [Build Error] [aula8-ex6.o] Error 1
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Usando const com Ponteiros

- Ponteiros const
 - Sempre aponta para o mesmo local de memória
 - O próprio nome do array
 - Deve ser inicializado quando declarado

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Sétimo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 7
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x, y;

    /*
     * ptr é um ponteiro constante para um inteiro que pode
     * ser modificado através de ptr, mas ptr aponta sempre para
     * mesma posição de memória
     */
    int * const ptr = &x;

    *ptr = 7; // Permitido: *ptr não é constante
    ptr = &y; //Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sétimo Exemplo Usando Ponteiros em C++

Ponteiro constante para dado não constante

```
#include <iostream>

using namespace std;

int main() {
    int x, y;

    /*
     * ptr é um ponteiro constante para um inteiro que pode
     * ser modificado através de ptr, mas ptr aponta sempre para
     * mesma posição de memória
     */
    int * const ptr = &x;

    *ptr = 7; // Permitido: *ptr não é constante
    ptr = &y; //Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Sétimo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 7
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x = 5, y;

    /*
     * ptr é um ponteiro constante para um inteiro constante;
     * ptr sempre aponta para a mesma posição de memória; o
     * inteiro naquela posição não pode ser modificado
     */
    const int *const ptr = &x;

    cout << *ptr << endl;

    *ptr = 7; // Erro: *ptr é constante, não pode receber um novo valor
    ptr = &y; //Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Oitavo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 8
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x = 5, y;

    /*
     * ptr é um ponteiro constante para um inteiro constante;
     * ptr sempre aponta para a mesma posição de memória; o
     * inteiro naquela posição não pode ser modificado
     */
    const int *const ptr = &x;

    cout << *ptr << endl;

    *ptr = 7; // Erro: *ptr é constante, não pode receber um novo valor
    ptr = &y; //Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Oitavo Exemplo Usando Ponteiros em C++

Ponteiro constante para dado constante

```
/*
 * Aula 8 - Exemplo 8
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x = 5, y;

    /*
     * ptr é um ponteiro constante para um inteiro constante;
     * ptr sempre aponta para a mesma posição de memória; o
     * inteiro naquela posição não pode ser modificado
     */
    const int *const ptr = &x;

    cout << *ptr << endl;

    *ptr = 7; // Erro: *ptr é constante, não pode receber um novo valor
    ptr = &y; //Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Oitavo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 8
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

int main() {
    int x = 5, y;

    /*
     * ptr é um ponteiro constante para um inteiro constante;
     * ptr sempre aponta para a mesma posição de memória; o
     * inteiro naquela posição não pode ser modificado
     */
    const int *const ptr = &x;

    cout << *ptr << endl;

    *ptr = 7; // Erro: *ptr é constante, não pode receber um novo valor
    ptr = &y; //Erro: ptr é constante, não pode receber um novo endereço

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Bubble Sort Usando Passagem por Referência

- Implementando `bubbleSort` usando ponteiros
 - Precisa da função `swap` para acessar elementos do array
 - Elementos do array individual: `escalares`
 - Passagem por valor por padrão
 - Passagem por referência usando operador de endereço `&`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Nono Exemplo Usando Ponteiros em C++

```
/*  
 * Aula 8 - Exemplo 9  
 * Autor: Miguel Campista  
 */  
#include <iostream>  
#include <omanip>  
  
using namespace std;  
  
void bubbleSort (int *, const int);  
void swap (int * const, int * const);  
  
int main() {  
    const int arraySize = 10;  
    int a [] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};  
  
    cout << "Dados na sequencia original\n";  
    for (int i = 0; i < arraySize; i++)  
        cout << setw(4) << a [i];  
  
    bubbleSort (a, arraySize);  
  
    cout << "\n\nDados em ordem crescente\n";  
    for (int i = 0; i < arraySize; i++)  
        cout << setw(4) << a [i];  
    cout << endl;  
  
    return 0;  
}
```

Nono Exemplo Usando Ponteiros em C++

```
void bubbleSort (int *array, const int size) {  
    for (int pass = 0; pass < size - 1; pass++) {  
        for (int j = 0; j < size - 1; j++) {  
            if (array [j] > array [j+1])  
                swap (&array[j], &array[j+1]);  
        }  
    }  
  
    void swap (int * const element1Ptr, int * const element2Ptr) {  
        int hold = *element1Ptr;  
        *element1Ptr = *element2Ptr;  
        *element2Ptr = hold;  
    }  
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Nono Exemplo Usando Ponteiros em C++

```
void bubbleSort (int *array, const int size) {  
    for (int pass = 0; pass < size - 1; pass++) {  
        for (int j = 0; j < size - 1; j++) {  
            if (array [j] > array [j+1])  
                swap (&array[j], &array[j+1]);  
        }  
    }  
  
    void swap (int * const element1Ptr, int * const element2Ptr) {  
        int hold = *element1Ptr;  
        *element1Ptr = *element2Ptr;  
        *element2Ptr = hold;  
    }  
}
```

```
shell>$ g++ -Wall exemplo.cpp -o ex9  
shell>$ ./ex9  
Dados na sequencia original  
2 6 4 8 10 12 89 68 45 37  
  
Dados em ordem crescente  
2 4 6 8 10 12 37 45 68 89  
shell>$
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Bubble Sort Usando Passagem por Referência

- `sizeof`
 - Operador unário retorna o tamanho do operando em bytes
 - Para arrays, `sizeof` retorna **(tamanho de 1 elemento) * (número de elementos)**
 - Se `sizeof (int) = 4`, então

```
int myArray[10];  
cout << sizeof (myArray); // imprime 40
```
- `sizeof` pode ser usado com:
 - Nomes de variáveis, nomes de tipos e valores constantes

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Exemplo Usando Ponteiros em C++

```
/*  
 * Aula 8 - Exemplo 10  
 * Autor: Miguel Campista  
 */  
#include <iostream>  
  
using namespace std;  
  
size_t getSize (double *);  
  
int main() {  
    double array [20];  
  
    cout << "O numero de bytes no array eh: "  
        << sizeof (array) << endl;  
  
    cout << "\nO numero de bytes retornados de getSize eh: "  
        << getSize (array) << endl;  
  
    return 0;  
}  
  
size_t getSize (double *ptr) { return sizeof (ptr);}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 10
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

size_t getSize (double *);

int main() {
    double array [20];

    cout << "\nO numero de bytes no array eh: "
         << sizeof(array) << endl;

    cout << "\nO numero de bytes retornados de getSize eh: "
         << getSize(array) << endl;

    return 0;
}

size_t getSize (double *ptr) { return sizeof(ptr);}

```

Um alias para unsigned int em muitos compiladores

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Décimo Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 10
 * Autor: Miguel Campista
 */
#include <iostream>

shell>$ g++ -Wall exemplo.cpp -o ex10
shell>$ ./ex10
O numero de bytes no array eh: 160

O numero de bytes retornados de getSize eh: 4
shell>$

    cout << "\nO numero de bytes retornados de getSize eh: "
         << getSize(array) << endl;

    return 0;
}

size_t getSize (double *ptr) { return sizeof(ptr);}

```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Décimo Primeiro Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 11
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

size_t getSize (double *);

int main() {
    char c;
    short s;
    int i;
    long l;
    float f;
    double d;
    long double ld;
    int array [20];
    int *ptr = array;
}

```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Décimo Primeiro Exemplo Usando Ponteiros em C++

```

    cout << "sizeof c = " << sizeof c
         << "\tsizeof(char) = " << sizeof(char)
         << "\tsizeof s = " << sizeof s
         << "\tsizeof(short) = " << sizeof(short)
         << "\tsizeof i = " << sizeof i
         << "\tsizeof(int) = " << sizeof(int)
         << "\tsizeof l = " << sizeof l
         << "\tsizeof(long) = " << sizeof(long)
         << "\tsizeof f = " << sizeof f
         << "\tsizeof(float) = " << sizeof(float)
         << "\tsizeof d = " << sizeof(d)
         << "\tsizeof(double) = " << sizeof(double)
         << "\tsizeof ld = " << sizeof ld
         << "\tsizeof(long double) = " << sizeof(long double)
         << "\tsizeof array = " << sizeof array
         << "\tsizeof ptr = " << sizeof ptr << endl;

    return 0;
}

size_t getSize (double *ptr) { return sizeof(ptr);}

```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Décimo Primeiro Exemplo Usando Ponteiros em C++

```

    cout << "sizeof c = " << sizeof c
         << "\tsizeof(char) = " << sizeof(char)
         << "\tsizeof s = " << sizeof s
         << "\tsizeof(short) = " << sizeof(short)
         << "\tsizeof i = " << sizeof i
         << "\tsizeof(int) = " << sizeof(int)

    return 0;
}

size_t getSize (double *ptr) { return sizeof(ptr);}

```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Expressões com Ponteiros e Aritmética com Ponteiros

- Aritmética com ponteiro
 - Incremento/decremento de ponteiro (++ ou --)
 - Adição/subtração de inteiro para/de um ponteiro (+ ou +=, - ou -=)
 - Ponteiros podem ser subtraídos entre si
 - Aritmética de ponteiro sem significado exceto se realizado sobre ponteiro para array

Linguagens de Programação – DEL-Pol/UFRJ

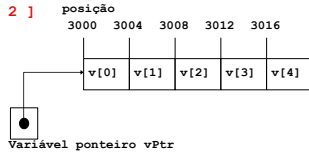
Prof. Miguel Campista

Expressões com Ponteiros e Aritmética com Ponteiros

- Array de 5 elementos `int` em uma máquina usando inteiros de 4 bytes
 - `vPtr` aponta para o primeiro elemento `v[0]`, que está na posição 3000


```
cout << vPtr; // Imprime 3000
```
 - `vPtr += 2`; atribui 3008 a `vPtr`

```
vPtr aponta para v[ 2 ]
```



Expressões com Ponteiros e Aritmética com Ponteiros

- Subtração de ponteiros
 - Retorna número de elementos entre dois endereços


```
vPtr2 = &v[ 2 ]; vPtr = &v[ 0 ];
cout << vPtr2 - vPtr; // Imprime 2
```
- Atribuição de ponteiro
 - Ponteiro pode ser atribuído para outro ponteiro se ambos forem do mesmo tipo
 - Se não forem, operador `cast` deve ser usado
 - Exceção: ponteiro para `void` (tipo `void *`)
 - Ponteiro genérico, representa qualquer tipo
 - Casting não é necessário para converter ponteiro para ponteiro `void`
 - Ponteiros `void` não podem ser acessados indiretamente

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Expressões com Ponteiros e Aritmética com Ponteiros

- Comparação de ponteiros
 - Uso de sinal de igualdade ou operadores relacionais
 - Comparações não fazem sentido exceto quando ponteiros apontam para algum membro do mesmo array
 - Comparações de endereços armazenados em ponteiros
 - Uso comum para determinar se um ponteiro é zero
 - O que significa que ele não aponta para nada

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Relação entre Ponteiros e Arrays

- Arrays e ponteiros são proximamente relacionados
 - Nome do array como ponteiro constante
 - Ponteiros podem fazer operações de arrays

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Relação entre Ponteiros e Arrays

- Acesso a elementos de array com ponteiros
 - Elemento `b[n]` pode ser acessado por `*(bPtr + n)`
 - Chamada notação deslocada de ponteiro
 - Endereços
 - `&b[3]` o mesmo que `bPtr + 3`
 - Nome do array pode ser tratado como ponteiro
 - `b[3]` o mesmo que `*(b + 3)`
 - Ponteiros podem ser indexados
 - `bPtr[3]` o mesmo que `b[3]`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Segundo Exemplo Usando Ponteiros em C++

```

//
// Aula 8 - Exemplo 12
// Autor: Miguel Campista
//
#include <iostream>

using namespace std;

int main() {
    int b [ ] = {10, 20, 30, 40};
    int *bPtr = b; // faz bPtr apontar para b

    cout << "Array b impresso com:\n";
    << "Notacao de endereços de array:\n";
    for (int i = 0; i < 4; i++)
        cout << "b[" << i << "] = " << b [i] << "\n";

    cout << "\nArray b impresso com:\n";
    << "Notacao de array como ponteiro deslocado:\n";
    for (int offset1 = 0; offset1 < 4; offset1++)
        cout << "b[" << offset1 << "] = "
            << (b + offset1) << "\n";

    cout << "\nArray b impresso com:\n";
    << "Notacao de ponteiro indexado:\n";
    for (int i = 0; i < 4; i++)
        cout << "bPtr [" << i << "] = " << bPtr [i] << "\n";

    cout << "\nArray b impresso com:\n";
    << "Notacao de ponteiro deslocado:\n";
    for (int offset2 = 0; offset2 < 4; offset2++)
        cout << "**bPtr + " << offset2 << ") = "
            << *(bPtr + offset2) << "\n";

    return 0;
}

```

Décimo Segundo Exemplo Usando Ponteiros em C++

```

/* Aula 8 - Exemplo 12
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

void copy1 (char *, const char *);
void copy2 (char *, const char *);

int main () {
    char string1 [10];
    char string2 [] = "Hello";
    char string3 [10];
    char string4 [] = "Good Bye!";

    copy1 (string1, string2);
    cout << "string1 = " << string1 << endl;

    copy2 (string3, string4);
    cout << "string3 = " << string3 << endl;

    return 0;
}

void copy1 (char *s1, const char *s2) {
    for (int i = 0; (s1 [i] = s2 [i]) != '\0'; i++)
        ; // Copio não faz nada
}

void copy2 (char *s1, const char *s2) {
    for ( ; (*s1 = *s2) != '\0'; s1++, s2++)
        ; // Copio não faz nada
}

```

Décimo Terceiro Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 13
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

void copy1 (char *, const char *);
void copy2 (char *, const char *);

int main () {
    char string1 [10];
    char string2 [] = "Hello";
    char string3 [10];
    char string4 [] = "Good Bye!";

    copy1 (string1, string2);
    cout << "string1 = " << string1 << endl;

    copy2 (string3, string4);
    cout << "string3 = " << string3 << endl;

    return 0;
}

void copy1 (char *s1, const char *s2) {
    for (int i = 0; (s1 [i] = s2 [i]) != '\0'; i++)
        ; // Copio não faz nada
}

void copy2 (char *s1, const char *s2) {
    for ( ; (*s1 = *s2) != '\0'; s1++, s2++)
        ; // Copio não faz nada
}

```

Décimo Terceiro Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 13
 * Autor: Miguel Campista
 */
#include <iostream>

using namespace std;

shell>$ g++ exemplo.cpp -o ex13
shell>$ ./ex13
string1 = Hello
string3 = Good Bye
shell>$

    copy1 (string1, string2);
    cout << "string1 = " << string1 << endl;

    copy2 (string3, string4);
    cout << "string3 = " << string3 << endl;

    return 0;
}

void copy1 (char *s1, const char *s2) {
    for (int i = 0; (s1 [i] = s2 [i]) != '\0'; i++)
        ; // Copio não faz nada
}

void copy2 (char *s1, const char *s2) {
    for ( ; (*s1 = *s2) != '\0'; s1++, s2++)
        ; // Copio não faz nada
}

```

Décimo Quarto Exemplo Usando Ponteiros em C++

- Escreva um programa que recebe strings e as armazene em um vector



Décimo Quarto Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 14
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>
#include "names.h"

int main () {
    Names n;
    string fname;

    for (int i = 0; i < Names::names; i++) {
        cout << "Entre com o primeiro nome: ";
        getline(cin, fname);
        n.insertNames (fname);
    }

    cout << "\nNúmero de nomes inseridos eh: "
         << n.getNumberNames() << endl;
    n.getNames();

    return 0;
}

```

Décimo Quarto Exemplo Usando Ponteiros em C++

```

#include <iostream>
#include <vector>

using namespace std;

class Names {
public:
    static const int names = 3;

    Names();
    void insertNames (string);
    void getNames();
    int getNumberNames();

private:
    vector <string> v;
};

```

Décimo Quarto Exemplo Usando Ponteiros em C++

```
#include "names.h"
Names::Names() {
    cout << "Antes " << v.size() << endl;
    v.resize(names);
    cout << "Depois " << v.size() << endl;
}
void Names::insertNames(string name) {
    static int id;
    v.at(id) = name;
    cout << "Inserido: " << v.at(id) << endl;
    id++;
}
void Names::getNames() {
    for (int i = 0; i < v.size(); i++)
        cout << "Nome " << i << ": " << v.at(i) << endl;
}
int Names::getNumberNames() { return v.size(); }
```

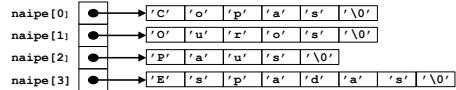
Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Arrays de Ponteiros

- Arrays podem conter ponteiros
 - Comumente usados para armazenar array de strings


```
char *naipe [4] = {"Copas", "Ouros", "Paus", "Espadas"};
```
 - Cada elemento de `naipe` aponta para um `char *` (uma string)
 - Array não armazena strings, somente ponteiros para strings



Linguagens de Programação – DEL-Poli/UFRJ

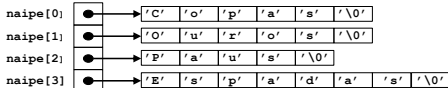
Prof. Miguel Campista

Arrays de Ponteiros

- Arrays podem conter ponteiros
 - Comumente usados para armazenar array de strings


```
char *naipe [4] = {"Copas", "Ouros", "Paus", "Espadas"};
```

→ **Array de ponteiros tem tamanho fixo, o valor apontado pelos arrays, não strings**



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Quinto Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 15
 * Autor: Miguel Campista
 */
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <iomanip>

void embaralha(int b[13][13]);
void aponta(const int b[13][13], const char *f[], const char *n[13]);

using namespace std;

int main() {
    const char *naipe [] = {"Copas", "Ouros", "Paus", "Espadas"};
    const char *face [] = {"As", "Dois", "Tres", "Quatro", "Cinco",
        "Seis", "Sete", "Oito", "Nove", "Dez",
        "Valete", "Dama", "Bes"};
    int baralho [4][13] = {0};
    srand(time(0));
    embaralha(baralho);
    aponta(baralho, face, naipe);
    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Quinto Exemplo Usando Ponteiros em C++

```
void embaralha(int b[13][13]) {
    int linha, coluna;
    for (int carta = 0; carta <= 52; carta++) {
        do {
            linha = rand() % 4;
            coluna = rand() % 13;
        } while (b [linha] [coluna] != 0);
        b [linha] [coluna] = carta;
    }
}

void aponta(const int b [13][13], const char * f [], const char * n [13]) {
    for (int carta = 1; carta <= 52; carta++) {
        for (int l = 0; l <= 3; l++) {
            for (int c = 0; c <= 12; c++) {
                if (b [l] [c] == carta) {
                    cout << setw(3) << right << f [c]
                        << " de " << setw(10) << left
                        << n [l] << (carta % 2 == 0 ? "O" : "A") << '\n';
                }
            }
        }
    }
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Quinto Exemplo Usando Ponteiros em C++

```
void embaralha(int b[13][13]) {
    int linha, coluna;
    for (int carta = 0; carta <= 52; carta++) {
        do {
            linha = rand() % 4;
            coluna = rand() % 13;
        } while (b [linha] [coluna] != 0);
        b [linha] [coluna] = carta;
    }
}

void aponta(const int b [13][13], const char * f [], const char * n [13]) {
    for (int carta = 1; carta <= 52; carta++) {
        for (int l = 0; l <= 3; l++) {
            for (int c = 0; c <= 12; c++) {
                if (b [l] [c] == carta) {
                    cout << setw(3) << right << f [c]
                        << " de " << setw(10) << left
                        << n [l] << (carta % 2 == 0 ? "O" : "A") << '\n';
                }
            }
        }
    }
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Ponteiros para Funções

- Ponteiros para funções
 - Contêm endereço da função
 - Parecido com o motivo pelo qual o nome do array é o endereço do primeiro elemento
 - Nome da função inicia endereço de código que define a função
- Ponteiros para funções podem ser
 - Passados para funções
 - Retornados das funções
 - Armazenados em arrays
 - Atribuídos a outros ponteiros para funções

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Ponteiros para Funções

- Funções que chamam funções através de ponteiros
 - Assumir parâmetro:
 - `bool (*compare) (int, int)`
 - Executar a função com os dois inteiros
 - `(*compare) (int1, int2)`
 - Referência indireta a um ponteiro para função executar
- OU
- `compare(int1, int2)`
 - Poderia ser confuso
 - » Usário pode pensar em comparar nome atual da função no programa e não usar o ponteiro

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

```

//
// Aula 8 - Exemplo 16
// Autor: Miguel Campista
//
#include <iostream>
#include <string>

void bubble(int [], const int, bool (*)(int, int));
void swap(int *, const, int * const);
bool ascending(int, int);
bool descending(int, int);

using namespace std;

int main() {
    const int arraySize = 10;
    int order, counter;
    int a [arraySize] = {2, 6, 4, 8, 0, 10, 12, 09, 68, 45, 37};

    cout << "Entre com 1 para ordenar em ordem ascendente, 1a"
    << "Entre 2 para ordenar em ordem descendente: ";
    cin >> order;
    cout << "\nAdubao na ordem original\n";

    // Array original
    for ( counter = 0; counter < arraySize; counter++ )
        cout << setw( 4 ) << a[ counter ] ;

    if (order == 1) {
        bubble( a, arraySize, ascending );
        cout << "\nAdubao em ordem ascendente\n";
    } else {
        bubble( a, arraySize, descending );
        cout << "\nAdubao em ordem decrescente\n";
    }

    // Array ordenado
    for (counter = 0; counter < arraySize; counter++ )
        cout << setw(4) << a [counter];

    cout << endl;

    return 0;
}
    
```

Décimo Sexto Exemplo Usando Ponteiros em C++

```

void bubble( int work[], const int size, bool (*compare) ( int, int ) ) {
    for ( int pass = 1; pass < size; pass++ ) {
        for ( int count = 0; count < size - 1; count++ ) {
            if ((*compare)(work [count], work [count + 1]))
                swap(&work[count], &work[count + 1] );
        }
    }
}

void swap( int * const element1Ptr, int * const element2Ptr ) {
    int hold = *element1Ptr;
    *element1Ptr = *element2Ptr;
    *element2Ptr = hold;
}

bool ascending( int a, int b ) {
    return b < a; // swap se b for menor que a
}

bool descending( int a, int b ) {
    return b > a; // swap se b for maior que a
}
    
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Décimo Sexto Exemplo Usando Ponteiros em C++

```

// void bubble( int work[], const int size, bool (*compare) ( int, int ) ) {
shell>$ g++ -Wall exemplo.cpp -o ex16
shell>$ ./ex16
Entre com 1 para ordenar em ordem ascendente,
Entre com 2 para ordenar em ordem descendente: 1

Dados na ordem original
2  6  4  8  10 12 09 68 45 37
Dados na ordem ascendente
2  4  6  8  10 12 37 45 68 89
shell>$

bool descending( int a, int b ) {
    return b > a; // swap se b for maior que a
}
    
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Ponteiros para Funções

- Arrays de ponteiros para funções
 - Sistemas orientados a menu
 - Ponteiro para cada função armazenada em array de ponteiros para funções
 - Todas as funções devem ter o mesmo tipo de retorno e os mesmos tipos de parâmetros
 - Escolha no menu → índice do array de ponteiros para funções

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

```

/*
 * Aula 9 - Exemplo 17
 * Autor: Miguel Campista
 */
#include <iostream>

void function1(int);
void function2(int);
void function3(int);

using namespace std;

int main() {
    void (*f[3])(int) = {function1, function2, function3};

    int choice;

    cout << "Entre um numero entre 0 e 2, 3 para terminar: ";
    cin >> choice;

    while (choice >= 0 && choice < 3) {
        // chama função na posição escolhida do array f
        // e passa a escolha como argumento
        (*f[choice])(choice);

        cout << "Entre um numero entre 0 e 2, 3 para terminar: ";
        cin >> choice;
    }

    cout << "Execução do program completa." << endl;
    return 0;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Sétimo Exemplo Usando Ponteiros em C++

```

void function1(int a) {
    cout << "Voce digitou " << a
        << " então function1 foi chamada!\n";
}

void function2(int b) {
    cout << "Voce digitou " << b
        << " então function2 foi chamada!\n";
}

void function3(int c) {
    cout << "Voce digitou " << c
        << " então function3 foi chamada!\n";
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Sétimo Exemplo Usando Ponteiros em C++

```

shell-$ g++ -Wall exemplo.cpp -o ex17
shell-$ ./ex17
Entre um numero entre 0 e 2, 3 para terminar: 0
Voce digitou 0 entao function1 foi chamada

Entre um numero entre 0 e 2, 3 para terminar: 1
Voce digitou 0 entao function2 foi chamada

Entre um numero entre 0 e 2, 3 para terminar: 2
Voce digitou 0 entao function3 foi chamada

Entre um numero entre 0 e 2, 3 para terminar: 0
Execução do programa completa
shell-$

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo 1

- Escreva um programa que calcule o valor mínimo e máximo de um vetor. Para isso, utilize a classe vector e utilize ponteiro para funções.



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo 1

```

/*
 * Aula 8 - Exemplo 23
 * Autor: Miguel Campista
 */
#include <iostream>
#include <vector>

using namespace std;

void processaData(int a, vector<int> &v, int (*f)(vector<int> &)) {
    int maximum(vector<int> &);
    int minimum(vector<int> &);
}

int main() {
    const int arraySize = 10;
    vector<int> v(arraySize);
    int resultado;

    int array [] = {9, 7, 23, 9, 10, 45, 65, 21, 18, 32};

    for (int i = 0; i < arraySize; i++)
        v[i] = array[i];

    processaData(resultado, v, maximum);
    cout << "O valor maximo do vetor eh: " << resultado << endl;

    processaData(resultado, v, minimum);
    cout << "O valor minimo do vetor eh: " << resultado << endl;

    return 0;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo 1

```

void processaData(int a, vector<int> &vec, int (*op)(vector<int> &)) {
    int r = (*op)(vec);
}

int maximum(vector<int> &a) {
    int max = 0;
    int i;
    for (i = 0; i < a.size(); i++) {
        if (a[i] > max)
            max = a[i];
    }
    return max;
}

int minimum(vector<int> &a) {
    int min = 100;
    int i;
    for (i = 0; i < a.size(); i++) {
        if (a[i] < min)
            min = a[i];
    }
    return min;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Exemplo 2

- Modifique o programa do Exemplo 1 para utilizar vetor de ponteiros. É possível?



Exemplo 2

```
/*  
 * Aula 8 - Exemplo 24  
 * Autor: Miguel Campista  
 */  
#include <iostream>  
#include <vector>  
  
using namespace std;  
  
void maximum(int i, vector<int> &v);  
void minimum(int i, vector<int> &v);  
  
int main() {  
    void (*f[2])(int i, vector<int> &) = {maximum, minimum};  
  
    const int arraySize = 10;  
    vector<int> v(arraySize);  
    int resultado;  
  
    int array [] = {3, 7, 23, 9, 10, 45, 65, 21, 15, 32};  
    for (int i = 0; i < arraySize; i++)  
        v[i] = array[i];  
  
    (*f[0])(resultado, v);  
    cout << "O valor maximo do vetor eh: " << resultado << endl;  
    (*f[1])(resultado, v);  
    cout << "O valor minimo do vetor eh: " << resultado << endl;  
  
    return 0;  
}
```

Exemplo 2

```
void maximum(int &r, vector<int> &a) {  
    int max = 0;  
    int i;  
    for (i = 0; i < a.size(); i++) {  
        if (a[i] > max)  
            max = a[i];  
    }  
    r = max;  
}  
  
void minimum(int &r, vector<int> &a) {  
    int min = 100;  
    int i;  
    for (i = 0; i < a.size(); i++) {  
        if (a[i] < min)  
            min = a[i];  
    }  
    r = min;  
}
```

Fundamentos de Caracteres e Arrays

- Constante caractere
 - Valor inteiro representado como caractere e aspas simples
 - 'z' é o valor inteiro de z
 - 122 em ASCII

Fundamentos de Caracteres e Arrays

- String
 - Série de caracteres tratados como uma única unidade
 - Pode incluir letras, dígitos, caracteres especiais (+, -, ...)
 - String literal (string constante)
 - Entre aspas duplas. Ex.: "Eu gosto de C++"
 - Array de caracteres, termina com caractere nulo '\0'
 - String é um ponteiro constante
 - Ponteiro para primeiro caractere da string
 - Como os arrays

Fundamentos de Caracteres e Arrays

- Atribuição de string
 - Array de caractere
 - char cor[] = "azul";
 - Cria 5 elementos char no array cor
 - » Último elemento é '\0'
 - Variável do tipo char *
 - char *corPtr = "azul";
 - Cria ponteiro corPtr para letra a na string "azul"
 - » "azul" em algum lugar na memória
 - Alternativa para array de caractere
 - char cor[] = { 'a', 'z', 'u', '\1', '\0' };
 - » '\1' é o valor inteiro de 1

Fundamentos de Caracteres e Arrays

- Leitura de strings
 - Atribui entrada para array de caracteres word[20]
 - `cin >> word`
 - Lê caracteres até espaço em branco ou EOF
 - String poderia exceder o tamanho do array
 - `cin >> setw(20) >> word;`
 - Lê 19 caracteres (espaço reservado para '\0')

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Fundamentos de Caracteres e Arrays

- `cin.getline`
 - Lê linha de texto
 - `cin.getline(array, size, delimiter);`
 - Copia entrada em array específico até ou
 - Tamanho menos um é alcançado
 - delimiter caractere é inserido
 - Ex.:


```
char sentence[ 80 ];
cin.getline(sentence, 80, '\n');
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções de Manipulação de Strings da Biblioteca cstring

- Biblioteca de manipulação de strings <cstring> prove funções para
 - Manipula dados do tipo string
 - Compara strings
 - Busca strings por caracteres e outras strings
 - Divide pedaços de strings
 - Separa strings em pedaços lógicos

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções de Manipulação de Strings da Biblioteca cstring

| Interface | Objetivo |
|---|--|
| <code>char *strcpy(char *s1, const char *s2);</code> | Copia a string s2 no array de caracteres s1. O valor de s1 é retornado. |
| <code>char *strncpy(char *s1, const char *s2, size_t n);</code> | Copia até n caracteres da string s2 no array de caracteres s1. O valor de s1 é retornado. |
| <code>char *strcat(char *s1, const char *s2);</code> | Adiciona a string s2 na string s1. O primeiro caractere de s2 sobrescreve o caractere de terminação nulo de s1. O valor de s1 é retornado. |
| <code>char *strncat(char *s1, const char *s2, size_t n);</code> | Adiciona até n caracteres da string s2 na string s1. O primeiro caractere de s2 sobrescreve o caractere de terminação nulo de s1. O valor de s1 é retornado. |
| <code>int strcmp(const char *s1, const char *s2);</code> | Compara a string s1 com a string s2. A função retorna um valor zero, menor que zero ou maior que zero se s1 for igual a, menor que ou maior que s2, respectivamente. |

Funções de Manipulação de Strings da Biblioteca cstring

| Interface | Objetivo |
|--|--|
| <code>int strcmp(const char *s1, const char *s2, size_t n);</code> | Compara até n caracteres a string s1 com a string s2. A função retorna zero, menor que zero ou maior que zero se s1 for igual a, maior que ou menor que s2, respectivamente. |
| <code>char *strtok(char *s1, const char *s2);</code> | A sequência de chamadas a strtok quebra a string s1 em pedaços (pedaços lógicos como palavras em uma linha de texto) delimitados por caracteres contidos na string s2. A primeira chamada contém s1 como primeiro argumento. Já as chamadas posteriores para continuar separando a mesma string contêm NULL como o primeiro argumento. Um ponteiro para o primeiro pedaço é retornado a cada chamada. Se não houver mais nenhum pedaço quando a palavra for chamada, NULL é retornado. |
| <code>char *strlen(const char *s);</code> | Determina o comprimento da string s. O número de caracteres precedendo o caractere de terminação nulo é retornado. |

Funções de Manipulação de Strings da Biblioteca cstring

- Cópia de strings
 - `char *strcpy(char *s1, const char *s2)`
 - Copia o segundo argumento no primeiro argumento
 - Primeiro argumento deve ser grande o suficiente para armazenar a string e terminar no caractere nulo
 - `char *strncpy(char *s1, const char *s2, size_t n)`
 - Especifica o número de caracteres a serem copiados da string no array
 - Não necessariamente copia o caractere de terminação nulo

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Oitavo Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 18
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>

using namespace std;

int main() {
    char s1[] = "Feliz aniversario para voce";
    char y[ 25 ];
    char z[ 15 ];

    strcpy( y, x ); // copia conteúdo de x em y

    cout << "A string no array x eh: " << x
         << "\nA string no array y eh: " << y << "\n";

    // copia primeira 14 caracteres de x em z
    strncpy( z, x, 14 ); // não copia caractere nulo
    z[ 14 ] = '\0'; // adiciona '\0' em z

    cout << "A string no array z eh: " << z << endl;

    return 0;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Oitavo Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 18
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>

using namespace std;

```

```
shell>$ g++ -Wall exemplo.cpp -o ex18
```

```
shell>$ ./ex18
```

A string no array x eh: Feliz aniversario para voce

A string no array y eh: Feliz aniversario para voce

A string no array z eh: Feliz aniversa

```
shell>$
```

```

strcpy( y, x, 14 ); // não copia caractere nulo
z[ 14 ] = '\0'; // adiciona '\0' em z

cout << "A string no array z eh: " << z << endl;

return 0;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções de Manipulação de Strings da Biblioteca cstring

• Concatenação de strings

- char *strcat(char *s1, const char *s2)
 - Adiciona segundo argumento no primeiro argumento
 - Primeiro caractere de segundo argumento substitui o caractere de terminação nulo do primeiro argumento
 - Assegura que primeiro argumento é grande o suficiente par armazenar o resultado da concatenação mais caractere nulo
- char *strncat(char *s1, const char *s2, size_t n)
 - Adiciona número específico de caracteres do segundo argumento no primeiro argumento
 - Adiciona caractere de terminação nulo ao resultado

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Décimo Nono Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 19
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>

using namespace std;

int main() {
    char s1[ 20 ] = "Feliz ";
    char s2[] = "Ano Novo ";
    char s3[ 40 ] = "";

    cout << "s1 = " << s1 << "\ns2 = " << s2;

    strcat( s1, s2 ); // concatena s2 e s1

    cout << "\n\nDepois de strcat(s1, s2):\ns1 = " << s1
         << "\ns2 = " << s2;

    // concatena primeira 6 caracteres de s1 com s3
    strncat( s3, s1, 6 ); // coloca '\0' depois do último caractere

    cout << "\n\nDepois de strncat(s3, s1, 6):\ns1 = " << s1
         << "\ns3 = " << s3;

    strcat( s3, s1 ); // concatena s1 e s3
    cout << "\n\nDepois de strcat(s3, s1):\ns1 = " << s1
         << "\ns3 = " << s3 << endl;

    return 0;
}

```

Décimo Nono Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 19
 * Autor: Miguel Campista
 */

```

```

shell>$ g++ -Wall exemplo.cpp -o ex19
shell>$ ./ex19
s1 = Feliz
s2 = Ano Novo

```

```

Depois do strcat(s1, s2):
s1 = Feliz Ano Novo
s2 = Ano Novo

```

```

Depois do strncat(s3, s1, 6):
s1 = Feliz Ano Novo
s3 = Feliz

```

```

Depois do strcat(s3, s1):
s1 = Feliz Ano Novo
s3 = Feliz Feliz Ano Novo
shell>$

```

```

strncat( s3, s1, 6 ); // concatena s1 com s3
cout << "\n\nDepois de strncat(s3, s1, 6):\ns1 = " << s1
     << "\ns3 = " << s3 << endl;

return 0;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções de Manipulação de Strings da Biblioteca cstring

• Comparação de strings

- Caracteres representados como códigos numéricos
 - Strings comparadas usando códigos numéricos
- Códigos de caractere / conjuntos de caractere
 - ASCII
 - "American Standard Code for Information Interchange"
 - EBCDIC
 - "Extended Binary Coded Decimal Interchange Code"

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

Funções de Manipulação de Strings da Biblioteca cstring

- Comparação de strings

- `int strcmp(const char *s1, const char *s2)`

- Compara caractere por caractere
- Retorna
 - Zero se strings forem iguais
 - Valor negativo se primeira string for menor que a segunda
 - Valor positivo se primeira string for maior que a segunda

- `int strncmp(const char *s1, const char *s2, size_t n)`

- Compara até o número especificado de caracteres
- Para de comparar se alcança um caractere nulo em um dos argumentos

Vigésimo Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 20
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>
#include <cstring>

using namespace std;

int main() {
    char *s1 = "Feliz Ano Novo";
    char *s2 = "Feliz Ano Novo";
    char *s3 = "Boas Festas";

    cout << "s1 = " << s1 << "\ns2 = " << s2
        << "\ns3 = " << s3 << "\n\n";
    cout << "strcmp(s1, s2) = " << strcmp(s1, s2) << "\n";
    cout << "strcmp(s1, s3) = " << strcmp(s1, s3) << "\n";
    cout << "strcmp(s2, s3) = " << strcmp(s2, s3) << "\n";

    cout << "strncmp(s1, s3, 6) = " << strncmp(s1, s3, 6) << "\n";
    cout << "strncmp(s1, s3, 7) = " << strncmp(s1, s3, 7) << "\n";
    cout << "strncmp(s2, s1, 7) = " << strncmp(s2, s1, 7) << "\n";
    cout << "strncmp(s3, s1, 7) = " << strncmp(s3, s1, 7) << "\n";

    return 0;
}
    
```

Vigésimo Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 20
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>
#include <cstring>

using namespace std;

int main() {
    char *s1 = "Feliz Ano Novo";
    char *s2 = "Feliz Ano Novo";
    char *s3 = "Boas Festas";

    cout << "s1 = " << s1 << "\ns2 = " << s2
        << "\ns3 = " << s3 << "\n\n";
    cout << "strcmp(s1, s2) = " << strcmp(s1, s2) << "\n";
    cout << "strcmp(s1, s3) = " << strcmp(s1, s3) << "\n";
    cout << "strcmp(s2, s3) = " << strcmp(s2, s3) << "\n";

    cout << "strncmp(s1, s3, 6) = " << strncmp(s1, s3, 6) << "\n";
    cout << "strncmp(s1, s3, 7) = " << strncmp(s1, s3, 7) << "\n";
    cout << "strncmp(s2, s1, 7) = " << strncmp(s2, s1, 7) << "\n";
    cout << "strncmp(s3, s1, 7) = " << strncmp(s3, s1, 7) << "\n";

    return 0;
}
    
```

Funções de Manipulação de Strings da Biblioteca cstring

- Divisão em pedaços

- Quebra de strings em pedaços, separados através da delimitação dos caracteres
- Divisão em unidade lógicas, como palavras (separação por espaços em branco)
- "Essa eh a minha string" tem 5 palavras que podem ser divididas (separadas por espaço)

- `char *strtok(char *s1, const char *s2)`

- Múltiplas chamadas necessárias

- Primeira chamada contém dois argumentos, string para ser partida em pedaços e string contendo caracteres delimitadores
 - » Encontrar próximo ao delimitador next e substituir com caractere NULL
- Subsequentes chamadas continuam separando
 - » Chamada com o primeiro argumento NULL

Vigésimo Primeiro Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 21
 * Autor: Miguel Campista
 */
#include <iostream>
#include <string>
#include <cstring>

using namespace std;

int main() {
    char sentenca[] = "Essa eh uma sentenca com 7 partes";
    char *tokenPtr;

    cout << "A string para ser dividida eh:\n" << sentenca
        << "\n\n";

    // Comparar separação da sentenca
    tokenPtr = strtok( sentenca, " " );

    while ( tokenPtr != NULL ) {
        cout << tokenPtr << "\n";
        tokenPtr = strtok( NULL, " " ); // pegue o próximo pedaço
    }

    cout << "\nDepois strtok, sentenca = " << sentenca << endl;

    return 0;
}
    
```

Vigésimo Primeiro Exemplo Usando Ponteiros em C++

```

/*
 * Aula 8 - Exemplo 21
 */
#include <iostream>
#include <string>
#include <cstring>

using namespace std;

int main() {
    char sentenca[] = "Essa eh uma sentenca com 7 partes";
    char *tokenPtr;

    cout << "A string para ser dividida eh:\n" << sentenca
        << "\n\n";

    // Comparar separação da sentenca
    tokenPtr = strtok( sentenca, " " );

    while ( tokenPtr != NULL ) {
        cout << tokenPtr << "\n";
        tokenPtr = strtok( NULL, " " ); // pegue o próximo pedaço
    }

    cout << "\nDepois strtok, sentenca = " << sentenca << endl;

    return 0;
}
    
```

Funções de Manipulação de Strings da Biblioteca `cstring`

- Determinação do comprimento das strings
 - `size_t strlen(const char *s)`
 - Retorna o número de caracteres na string
 - Caractere de terminação nulo não está incluído no comprimento

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Vigésimo Segundo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 22
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>
#include <cstring>

using namespace std;

int main() {
    char *string1 = "abcdefghijklmnopqrstuvwxyz";
    char *string2 = "quatro";
    char *string3 = "Rio de Janeiro";

    cout << "\n comprimento de \\\n" << string1
         << "\n eh " << strlen( string1 )
         << "\n\n comprimento de \\\n" << string2
         << "\n eh " << strlen( string2 )
         << "\n\n comprimento de \\\n" << string3
         << "\n eh " << strlen( string3 ) << endl;

    return 0;
}
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Vigésimo Segundo Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 22
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>
#include <cstring>
```

```
shell>$ g++ -Wall exemplo.cpp -o ex22
shell>$ ./ex22
O comprimento de "abcdefghijklmnopqrstuvwxyz" eh 26
O comprimento de "quatro" eh 6
O comprimento de "Rio de Janeiro" eh 14
shell>$
```

```
cout << "\n comprimento de \\\n" << string1
     << "\n eh " << strlen( string1 )
     << "\n\n comprimento de \\\n" << string2
     << "\n eh " << strlen( string2 )
     << "\n\n comprimento de \\\n" << string3
     << "\n eh " << strlen( string3 ) << endl;

    return 0;
}
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Exemplo 3

- Escreva um programa que receba um cadastro <nome, idade> e escreva em um arquivo. O programa deve ainda ser capaz de exibir todos os cadastros do arquivo e de excluir o arquivo.



Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Exemplo 3

```
/*
 * Aula 8 - Exemplo 26
 * Arquivo CapEx26.h
 * Autor: Miguel Campista
 */
#define CAPEX26_H
#define CAPEX26_H_
```

```
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <string>
#include <iomanip>
#include <cstdlib>
```

```
using namespace std;
```

```
class Cadastro {
public:
    Cadastro (string);
    ~Cadastro ();

    void writeLinesToFile (string, string);
    void readLinesFromFile();
    void clearFile();

private:
    string fileName;
    fstream fileStr;

    void checkIfFileOpen ();
};
```

```
#endif //CAPEX26_H_
```

Exemplo 3

```
/*
 * Aula 8 - Exemplo 26
 * Arquivo CapEx26.cpp
 * Autor: Miguel Campista
 */
#include "CapEx26.h"
```

```
Cadastro::Cadastro (string file) {
    fileName = file + ".txt";
    cout << "file: " << fileName << "\n\n" << endl;
}
```

```
Cadastro::~Cadastro () {
    cout << "Fechando o arquivo..." << endl;
}
```

```
void Cadastro::checkIfFileOpen () {
    if (!fileStr.is_open ()) {
        cout << "Problemas ao abrir o arquivo...\nFalando" << endl;
        exit (1);
    }
}
```

```
void Cadastro::writeLinesToFile (string name, string age) {
    // Abrindo arquivo para escrita
    fileStr.open (fileName.c_str(), fstream::in | fstream::out | fstream::app);
    checkIfFileOpen ();
}
```

```
// Escrevendo no arquivo uma linha
fileStr << left << setw(25) << name << setw(5) << age << endl;
fileStr.close ();
}
```

Linguagens de Programação – DEL-Pol/UFRJ

Prof. Miguel Campista

Exemplo 3

```
void Cadastro::readLinesFromFile() {
    string line;
    ifstream.open(fileName_c_str(), fstream::in | fstream::out | fstream::app);
    checkIfFileOpen();
    while (!ifstream.eof()) {
        getline (ifstream, line);
        cout << line << endl;
    }
    ifstream.close ();
}

void Cadastro::clearFile() {
    if (!remove(fileName_c_str()))
        cout << "Arquivo removido..." << endl;
    else
        cout << "Não foi possível remover o arquivo..." << endl;
}
```

Exemplo 3

```
/*
 * Aula 8 - Exemplo 26
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include "CapEX25.h"

int main() {
    string file, op;
    cout << "Entre com o nome do arquivo: ";
    getline(cin, file);
    Cadastro cad (file);

    while (1) {
        cout << "Entre com a operação desejada: ";
        getline (cin, op);
        cout << "Operação selecionada: " << op << endl;
        if (!op.compare ("Inserir")) {
            string name, age;
            cout << "Entre com o nome: ";
            getline (cin, name);
            cout << "Entre com a idade: ";
            getline (cin, age);
            cout << "Cadastro a ser inserido:\n"
                << left << setw(25) << name << setw(3) << age << endl;
            cad.writeLinesToFile (name, age);
        }
```

Exemplo 3

```
    } else if (!op.compare ("Mostrar")) {
        cad.readLinesFromFile();
    } else if (!op.compare ("Limpar")) {
        cad.clearFile();
    } else if (!op.compare ("terminar")) {
        cout << "Saindo do programa..." << endl;
        break;
    } else
        cout << "Operação desconhecida..." << endl;
    }
    return 0;
}
```

Leitura Recomendada

- Capítulos 8 do livro
- Deitel, *C++ How to Program*, 5th edition, Editora
Prentice Hall, 2005