Linguagens de Programação

Prof. Miguel Elias Mitre Campista

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

Parte IV

Introdução à Programação em C++
(Continuação)

Relembrando da Última Aula...

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

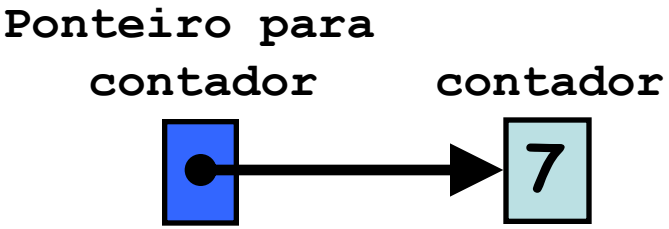
Ponteiros

- 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

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

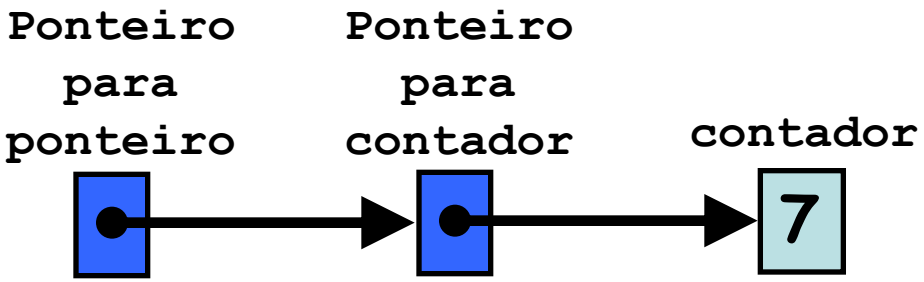


Declaração e Inicialização de Variáveis Ponteiros

- 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;
```



Declaração e Inicialização de Variáveis Ponteiros

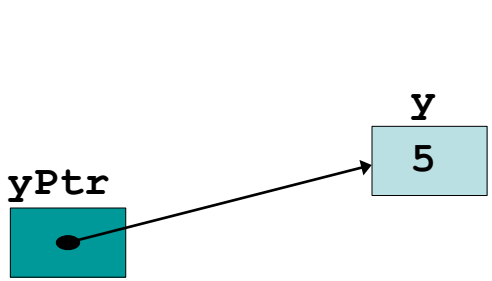
- Pode declarar ponteiros para qualquer tipo de dados
- Inicialização de ponteiro
  - Inicializado com 0, `NULL`, ou endereço
    - 0 ou `NULL` aponta para nada

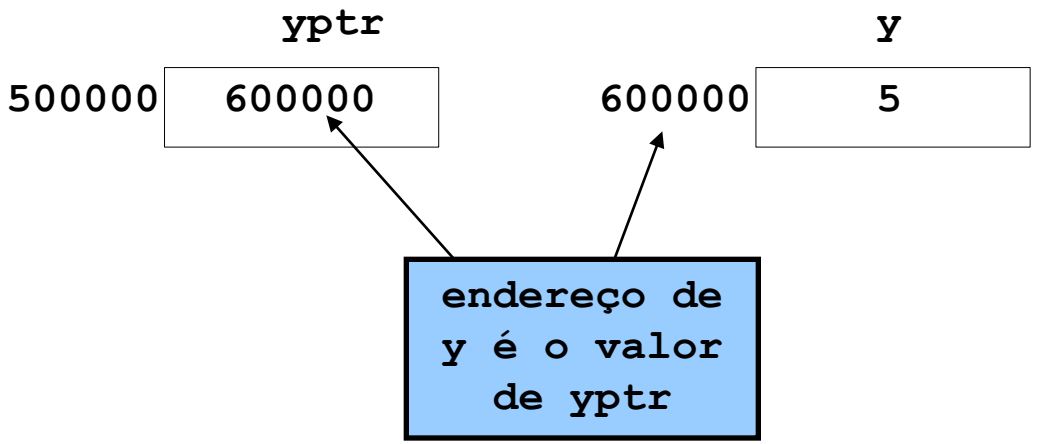
Operadores Ponteiros

- & (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

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 endereco de a eh " << &a
         << "\nO valor de aPtr eh " << aPtr;

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

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

    return 0;
}
```

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;
}
```

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"

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

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; }
```

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; }
```

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;
```

Como ficaria se a passagem de parâmetro fosse por referência?

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

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 *nPtr) { *nPtr = (*nPtr) * (*nPtr) * (*nPtr); }
```

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 *nPtr) { *nPtr = (*nPtr) * (*nPtr) * (*nPtr); }
```

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

Usando `const` com Ponteiros

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

Quarto Exemplo Usando Ponteiros em C++

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

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++;
     }
}
```

Quarto Exemplo Usando Ponteiros em C++

Ponteiro não constante para dado não constante

```
/*
#include <iostream>
#include <cctype>

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++;
     }
}
```

Quarto Exemplo Usando Ponteiros em C++

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

```
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++;
     }
}
```

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;
}
```

Quinto Exemplo Usando Ponteiros em C++

Ponteiro não constante para dado constante

```
/*
 */
#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;
}
```

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;
    printCharacteres(phrase);
    cout << endl;

    return 0;
}

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

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(&y); // Tenta modificação ilegal

    return 0;
}

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

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(&y); // Tenta modificação ilegal

    return 0;
}

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

Sexto Exemplo Usando Ponteiros em C++

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

void f(const int *);

int main() {
    int y;

    f(&y); // Tenta modificação ilegal

    return 0;
}

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

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

Usando `const` com Ponteiros

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

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;
}
```

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;
}
```

Sétimo Exemplo Usando Ponteiros em C++

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

using namespace std;

int main() {
```

	C:\Users\Miguel\Documents\UFRJ\...	In function `int main()':
20	C:\Users\Miguel\Documents\UFRJ\...	assignment of read-only variable `ptr'
	C:\Users\Miguel\Documents\UFRJ\...	[Build Error] [aula8-ex7.o] Error 1

```
     * 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;
}
```

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;
}
```

Oitavo Exemplo Usando Ponteiros em C++

Ponteiro constante para dado constante

```
 */
#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;
}
```

Oitavo Exemplo Usando Ponteiros em C++

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

using namespace std;

int main() {
```

	C:\Users\Miguel\Documents\UFRJ\...	In function `int main()':
21	C:\Users\Miguel\Documents\UFRJ\...	assignment of read-only location
22	C:\Users\Miguel\Documents\UFRJ\...	assignment of read-only variable `ptr'
	C:\Users\Miguel\Documents\UFRJ\...	[Build Error] [aula8-ex8.o] Error 1

```
    * 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;
}
```

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 **`&`**

Nono Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 9
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

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;
}
```

Nono Exemplo Usando Ponteiros em C++

```
void bubbleSort (int *array, const int size) {
    for (int pass = 0; pass < size - 1; pass++) {
        for (int i = 0; i < size - 1; i++) {
```

```
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>$
```

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

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);}
```

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);}
```

Um alias para **unsigned int** em muitos compiladores

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);}
```

Décimo Primeiro Exemplo Usando Ponteiros em C++

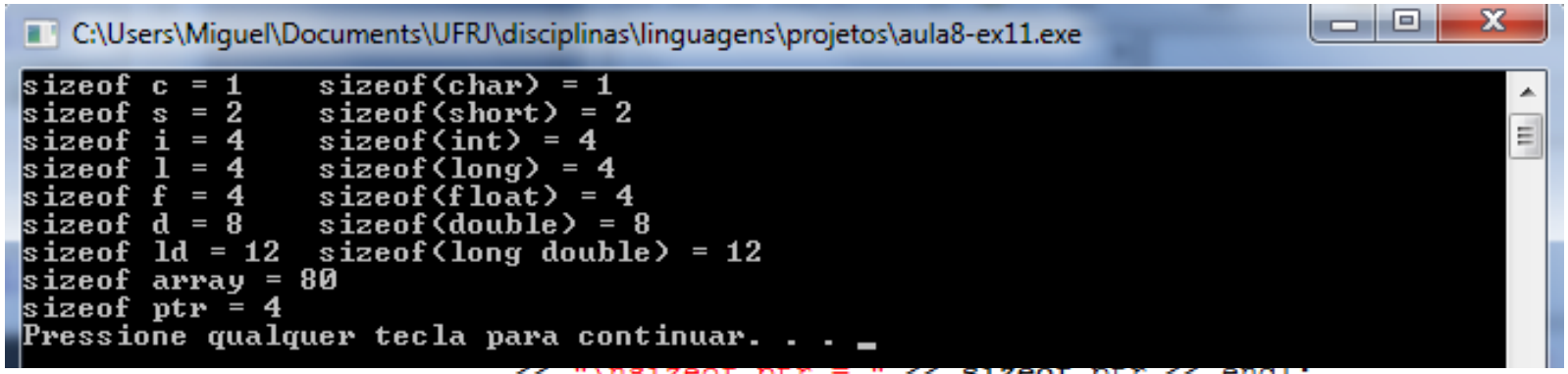
```
/*
 * Aula 8 - Exemplo 11
 * Autor: Miguel Campista
 */
#include <iostream>

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;
```

Décimo Primeiro Exemplo Usando Ponteiros em C++

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

    return 0;
}

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

Décimo Primeiro Exemplo Usando Ponteiros em C++

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




```
         << "\nsizeof ptr = " << sizeof ptr << endl;

    return 0;
}

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

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

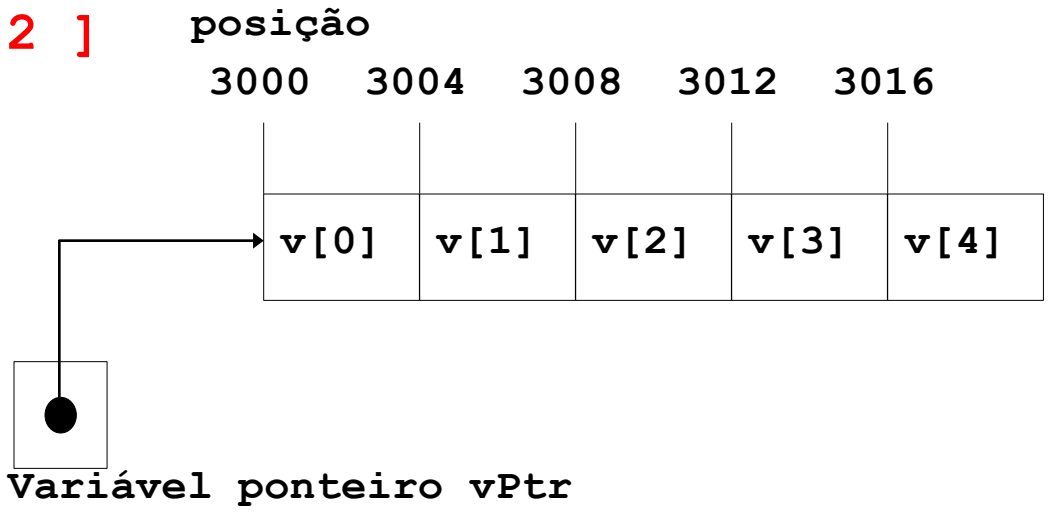
Expressões com Ponteiros e Aritmética com Ponteiros

- 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

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

Relação entre Ponteiros e Arrays

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

Relação entre Ponteiros e Arrays

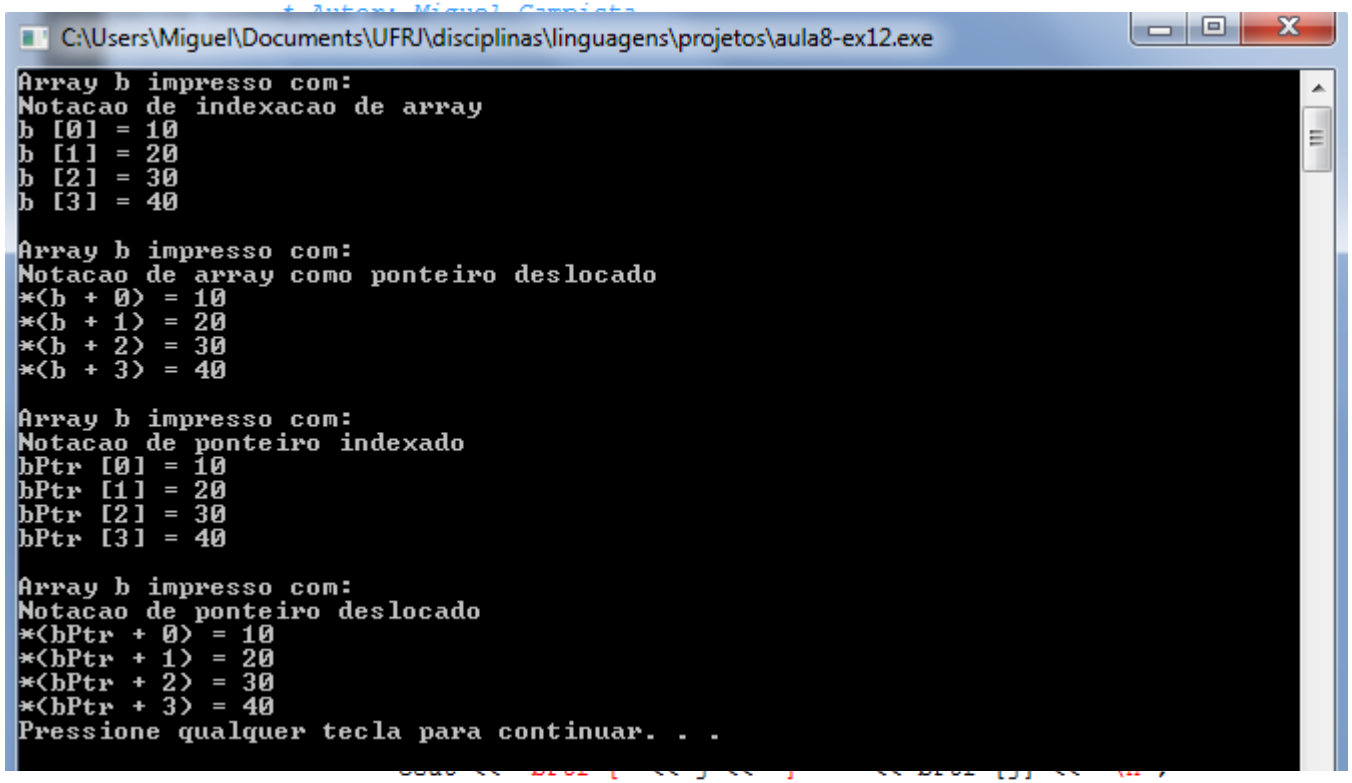
- Acesso a elementos de array com ponteiros
  - Elemento `b[ n ]` pode ser acessado por `*(bPtr + n)`
    - Chamada notação deslocada de ponteiro
  - Endereços
    - `&b[ 3 ]` o mesmo que `bPtr + 3`
  - Nome do array pode ser tratado como ponteiro
    - `b[ 3 ]` o mesmo que `*( b + 3 )`
  - Ponteiros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

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 indexacao 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 j = 0; j < 4; j++)
         cout << "bPtr [" << j << "] = " << bPtr [j] << "\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
```




```
    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 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++)
         ; // Corpo não faz nada
}
void copy2 (char *s1, const char *s2) {
     for ( ; (*s1 = *s2) != '\0'; s1++, s2++)
         ; // Corpo 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++)
         ; // Corpo não faz nada
}
void copy2 (char *s1, const char *s2) {
     for ( ; (*s1 = *s2) != '\0'; s1++, s2++)
         ; // Corpo 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 << "\n\nNumero 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++

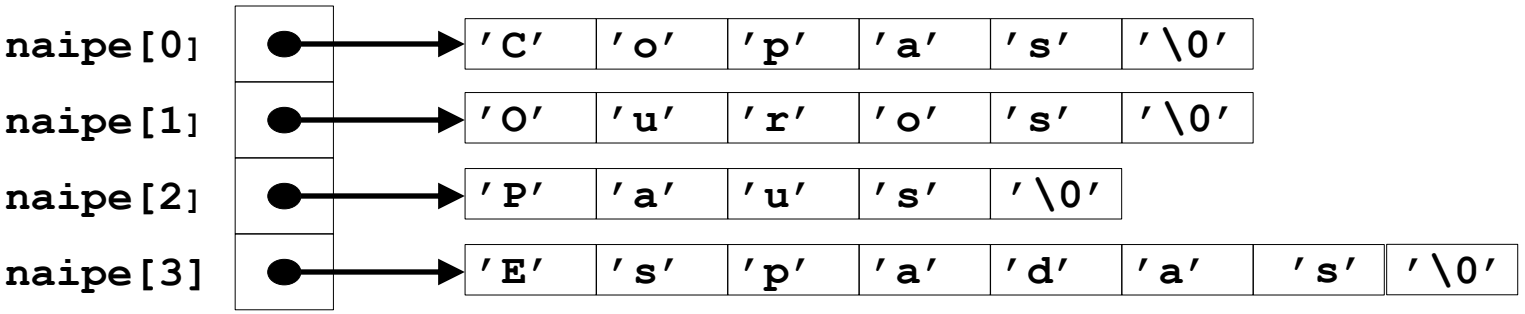
```
#include "names.h"

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(); }
```

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



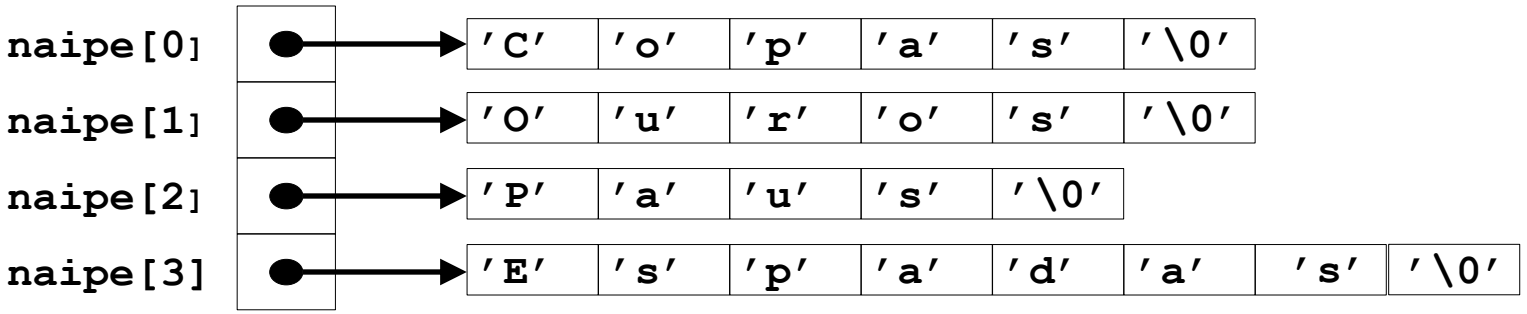
Arrays de Ponteiros

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

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

  - Cada elemento do naipe aponta para um [illegible] (uma strings

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



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 [][13]);
void aposta(const int [][13], const char *[], const char *[]);

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", "Rei"};
    int baralho [4][13] = {};

    srand(time(0));

    embaralha(baralho);
    aposta(baralho, face, naipe);

    return 0;
}
```

Décimo Quinto Exemplo Usando Ponteiros em C++

```
void embaralha(int b[][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 aposta(const int b [][13], const char * f [], const char * n []) {
     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(5) << right << f [c]
                            << " de " << setw(8) << left
                            << n [l] << (carta % 2 == 0 ? '\n' : '\t');
                 }
             }
         }

     }
}
```

Décimo Quinto Exemplo Usando Ponteiros em C++

```
void embaralha(int b[][13]) {
    int
    for
    }
}

void a
    f
    }
}
```

```
Valete de Paus          Dama de Ouros
  Rei de Espadas        Nove de Ouros
 Seis de Copas          Dois de Copas
 Nove de Paus          Cinco de Paus
 Seis de Espadas          As de Espadas
 Sete de Paus             As de Paus
 Tres de Copas          Dois de Espadas
 Nove de Espadas        Oito de Copas
 Seis de Ouros         Quatro de Ouros
 Oito de Espadas       Cinco de Copas
 Sete de Espadas          As de Copas
 Seis de Paus          Quatro de Copas
 Dois de Paus          Valete de Copas
  Dez de Espadas        Dama de Espadas
Cinco de Espadas         Dez de Ouros
 Tres de Espadas         Dez de Paus
 Tres de Paus            Dez de Copas
 Oito de Ouros          Sete de Copas
Valete de Espadas       Dama de Copas
 Nove de Copas         Quatro de Espadas
 Tres de Ouros           Rei de Paus
  Rei de Ouros          Sete de Ouros
  Rei de Copas         Valete de Ouros
Cinco de Ouros          Dois de Ouros
 Dama de Paus          Quatro de Paus
   As de Ouros           Oito de Paus
```

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

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
        - » Usuário pode pensar em comparar nome atual da função no programa e não usar o ponteiro

```
/*
 * Aula 8 - Exemplo 16
 * Autor: Miguel Campista
 */
#include <iostream>
#include <iomanip>

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, 10, 12, 89, 68, 45, 37};

    cout << "Entre com 1 para ordenar em ordem ascendente,\n"
         << "Entre 2 para ordenar em ordem descendente: ";
    cin >> order;
    cout << "\nDados 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 << "\nDados em ordem ascendente\n";
    } else {
         bubble( a, arraySize, descending );
         cout << "\nDados em ordem descendente\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
}
```

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   89   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
}
```

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

```
/*
 * Aula 8 - 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 << "Execucao do program completa." << endl;

    return 0;
}
```

Décimo Sétimo Exemplo Usando Ponteiros em C++

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

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

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

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
Execucao do programa completa
shell>$
```

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.

?

Exemplo 1

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

using namespace std;

void processData(int &, vector <int> &, int (*)(vector <int> &));
int maximum(vector <int> &);
int minimum(vector <int> &);

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

    int array [] = {3, 7, 23, 9, 10, 45, 65, 21, 18, 32};

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

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

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

    return 0;
}
```

Exemplo 1

```
void processData(int &r, vector <int> &vec, int (*op)(vector <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;
}
```

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 &, vector <int> &);
void minimum(int &, vector <int> &);

int main() {
    void (*f[2])(int &, vector <int> &) = {maximum, minimum};

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

    int array [] = {3, 7, 23, 9, 10, 45, 65, 21, 18, 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', 'l', '\0' };`

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'`)

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');
```

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

Funções de Manipulação de Strings da Biblioteca `cstring`

Interface	Objetivo
`char *strcpy(char *s1, const char *s2);`	Copia a string **s2** no array de caractere **s1**. O valor de **s1** é retornado.
`char *strncpy( char *s1, const char *s2, size_t n );`	Copia até **n** caracteres da string **s2** no array de caractere **s1**. O valor de **s1** é retornado.
`char *strcat(char *s1, const char *s2);`	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.
`char *strncat(char *s1, const char *s2, size_t n);`	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.
`int strcmp(const char *s1, const char *s2);`	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
`int strncmp( const char *s1, const char *s2, size_t n );`	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.
`char *strtok( char *s1, const char *s2 );`	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.
`char *strlen(const char *s);`	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

Décimo Oitavo Exemplo Usando Ponteiros em C++

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

using namespace std;

int main() {
    char x[] = "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 primeiros 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;
}
```

Décimo Oitavo Exemplo Usando Ponteiros em C++

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

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>$
```

```
    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;
}
```

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

Décimo Nono Exemplo Usando Ponteiros em C++

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

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 primeiros 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>$
```

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

    return 0;
}
```

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 Interchage"
    - EBCDIC
      - "Extended Binary Coded Decimal Interchange Code"

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 <iomanip>
#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\nstrcmp(s1, s2) = "
         << setw( 2 ) << strcmp( s1, s2 )
         << "\nstrcmp(s1, s3) = " << setw( 2 )
         << strcmp( s1, s3 ) << "\nstrcmp(s3, s1) = "
         << setw( 2 ) << strcmp( s3, s1 );

    cout << "\n\nstrncmp(s1, s3, 6) = " << setw( 2 )
         << strncmp( s1, s3, 6 ) << "\nstrncmp(s1, s3, 7) = "
         << setw( 2 ) << strncmp( s1, s3, 7 )
         << "\nstrncmp(s3, s1, 7) = "
         << setw( 2 ) << strncmp( s3, s1, 7 ) << endl;

    return 0;
}
```

Vigésimo Exemplo Usando Ponteiros em C++

```
shell>$ g++ exemplo.cpp –o ex20
shell>$ ./ex20
s1 = Feliz Ano Novo
s2 = Feliz Ano Novo
s3 = Boas Festas

strcmp(s1, s2) = 0
strcmp(s1, s3) = 1
strcmp(s3, s1) = -1

strncmp(s1, s3, 6) = 4
strncmp(s1, s3, 7) = 4
strncmp(s3, s1, 7) = -4
shell>$
```

```
/*
 * Aula 8 - Exemplo 20
 * Autor: Miguel Campista
```

```
    cout << "\n\nstrncmp(s1, s3, 6) = " << setw( 2 )
         << strncmp( s1, s3, 6 ) << "\nstrncmp(s1, s3, 7) = "
         << setw( 2 ) << strncmp( s1, s3, 7 )
         << "\nstrncmp(s3, s1, 7) = "
         << setw( 2 ) << strncmp( s3, s1, 7 ) << endl;

    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 <iomanip>
#include <cstring>

using namespace std;

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

    cout << "A string para ser dividida eh:\n" << sentence
         << "\n\nAs partes sao:\n\n";

    // Começar separação da sentença
    tokenPtr = strtok( sentence, " " );

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

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

    return 0;
}
```

Vigésimo Primeiro Exemplo Usando Ponteiros em C++

```
/*
 * Aula 8 - Exemplo 21
```

```
shell>$ g++ -Wall exemplo.cpp –o ex21
shell>$ ./ex21
A string para ser dividida eh:
Essa eh uma sentenca com 7 partes

As partes são:

Essa
eh
uma
sentenca
com
7
partes

Depois strtok, sentenca = Essa
shell>$
```

```
    cout << "\nDepois strtok, sentenca = " << sentence << 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

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 << "O comprimento de \"" << string1
         << "\" eh " << strlen( string1 )
         << "\nO comprimento de \"" << string2
         << "\" eh " << strlen( string2 )
         << "\nO comprimento de \"" << string3
         << "\" eh " << strlen( string3 ) << endl;

    return 0;
}
```

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 << "O comprimento de \"" << string1
         << "\" eh " << strlen( string1 )
         << "\nO comprimento de \"" << string2
         << "\" eh " << strlen( string2 )
         << "\nO comprimento de \"" << string3
         << "\" eh " << strlen( string3 ) << endl;

    return 0;
}
```

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.

?

Exemplo 3

```
/*
 * Aula 8 - Exemplo 25
 * Arquivo Cap8Ex25.h
 * Autor: Miguel Campista
 */
#ifndef CAP8EX25_H_
#define CAP8EX25_H_

#include <iostream>
#include <cstdio>
#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 checkIsFileOpen ();
};

#endif /*CAP8EX25_H_*/
```

Exemplo 3

```
/*
 * Aula 8 - Exemplo 25
 * Arquivo Cap8Ex25.cpp
 * Autor: Miguel Campista
 */
#include "Cap8Ex25.h"

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

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

void Cadastro::checkIsFileOpen () {
	if (!filestr.is_open ()) {
		cout << "Problemas ao abrir o arquivo...\nSaindo" << 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);
	checkIsFileOpen ();

	// Escrevendo no arquivo uma linha
	filestr << left << setw(25) << name << setw(3) << age << endl;
	filestr.close ();
}
```

Exemplo 3

```
void Cadastro::readLinesFromFile() {
	string line;

	filestr.open (fileName.c_str(), fstream::in | fstream::out | fstream::app);
	checkIsFileOpen ();

	while (!filestr.eof ()) {
		getline (filestr, line);
		cout << line << endl;
	}
	filestr.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 25
 * Arquivo principal
 * Autor: Miguel Campista
 */
#include "Cap8Ex25.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