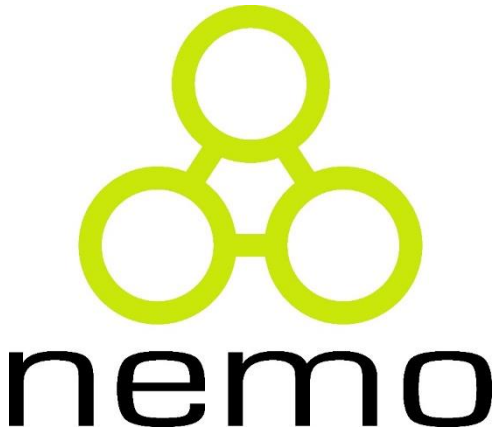


Programação III

Jordana S. Salamon

jssalamon@inf.ufes.br

jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Arquivos

Introdução

- Nós aprendemos anteriormente que para realizar leituras do teclado e escritas na tela, nos utilizamos os comandos **CIN** e **COUT** da biblioteca **<iostream>.**

- Para leitura e escrita em arquivos, uma novo biblioteca padrão da linguagem C++ é requerida para a manipulação de arquivos.
  - **<fstream>**

- Ainda assim, para realizar o processamento de arquivos em C++, as bibliotecas **<iostream>** e **<fstream>** precisam ser incluídas no programa.

nemo

Tipos de Dados – Biblioteca FStream

- A biblioteca **<fstream>** define 3 novos tipos de dados:
  - **Ofstream**: este tipo de dados representa um stream de saída (output) de arquivos e é utilizado para criar arquivos e para escrever informações nesses arquivos.
  - **Ifstream**: este tipo de dados representa um stream de entrada (input) de arquivos é utilizado para ler informações de arquivos.
  - **Fstream**: este tipo de dados representa um stream genérico, e tem a capacidade tanto do ofstream quanto do ifstream, ou seja, ele pode criar arquivos, escrever informações em arquivos e ler informações de arquivos.

Abrindo um arquivo

- Um arquivo precisa ser aberto antes de poder ler ou escrever algo.
- Tanto os tipos de dados ofstream ou fstream pode ser utilizados para abrir um arquivo de escrita.
- E o objeto ifstream é usado para abrir arquivos de leitura.
- A sintaxe para abertura de arquivo é a seguinte:

```
void open(const char *filename, ios::openmode mode);
```

- O primeiro argumento especifica a localização do arquivo a ser aberto.
- Enquanto o segundo argumento define em qual modo o arquivo será aberto.

Modo de abertura de arquivo

Modo	Descrição
ios::app	Modo de apende. Todos as saídas serão acrescentadas no final do arquivo.
ios::ate	Abre um arquivo para escrita e move o controlador de leitura/escrita para o final do arquivo.
ios::in	Abre um arquivo para a leitura.
ios::out	Abre um arquivo para escrita.
ios::trunc	Se o arquivo já existe, seu conteúdo ser truncado antes da abertura do arquivo.

Abrindo um arquivo

- É possível combinar dois ou mais desses modos.
- Por exemplo se você quer abrir um arquivo no modo de escrita e quer truncá-lo caso o arquivo já exista, basta utilizar a seguinte sintaxe:

```
ofstream outfile;
outfile.open("file.dat", ios::out | ios::trunc );
```

- Um outro exemplo, você pode também abrir um arquivo para os propósitos de leitura e escrita:

```
fstream  afile;
afile.open("file.dat", ios::out | ios::in );
```

Verificar se o arquivo foi aberto

- Lembre-se de verificar se o arquivo foi aberto corretamente, caso ele não exista, o programa não gera erro, mas todo o processamento é comprometido.
- Para verificar se o arquivo foi aberto corretamente, basta utilizar o seguinte código:

```
ifstream infile;
infile.open("saida.txt");
if(!infile.is_open()){
    cerr << "Erro ao abrir o arquivo, o mesmo nao existe!" << endl;
    exit(1);
}
```

nemo

Fechando um arquivo

- Quando um programa C++ termina sua execução, ele automaticamente fecha todos os streams abertos, liberando toda a memória alocada e fecha todos os arquivos abertos.
- Contudo, de todo modo, é uma boa pratica que o programador deve fechar todos os arquivos abertos antes do final do programa.
- A sintaxe para fechar o arquivo é a seguinte:

```
void close();
```

Escrevendo em um arquivo

- Para imprimir uma informação em um arquivo, nós utilizamos o operador (<<).

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

using namespace std;

int main(){
    int n;
    double y;
    char c;
    string s;

    ofstream outfile;
    outfile.open("saida.txt");
    cout << "Digite um inteiro: ";
    cin >> n;
    cout << "Digite um real: ";
    cin >> y;
    cout << "Digite um caracter: ";
    cin >> c;
    cout << "Digite uma palavra: ";
    cin.ignore();
    getline(cin, s);

    outfile << n << endl;
    outfile << y << endl;
    outfile << c << endl;
    outfile << s << endl;
}
```

Lendo de um arquivo

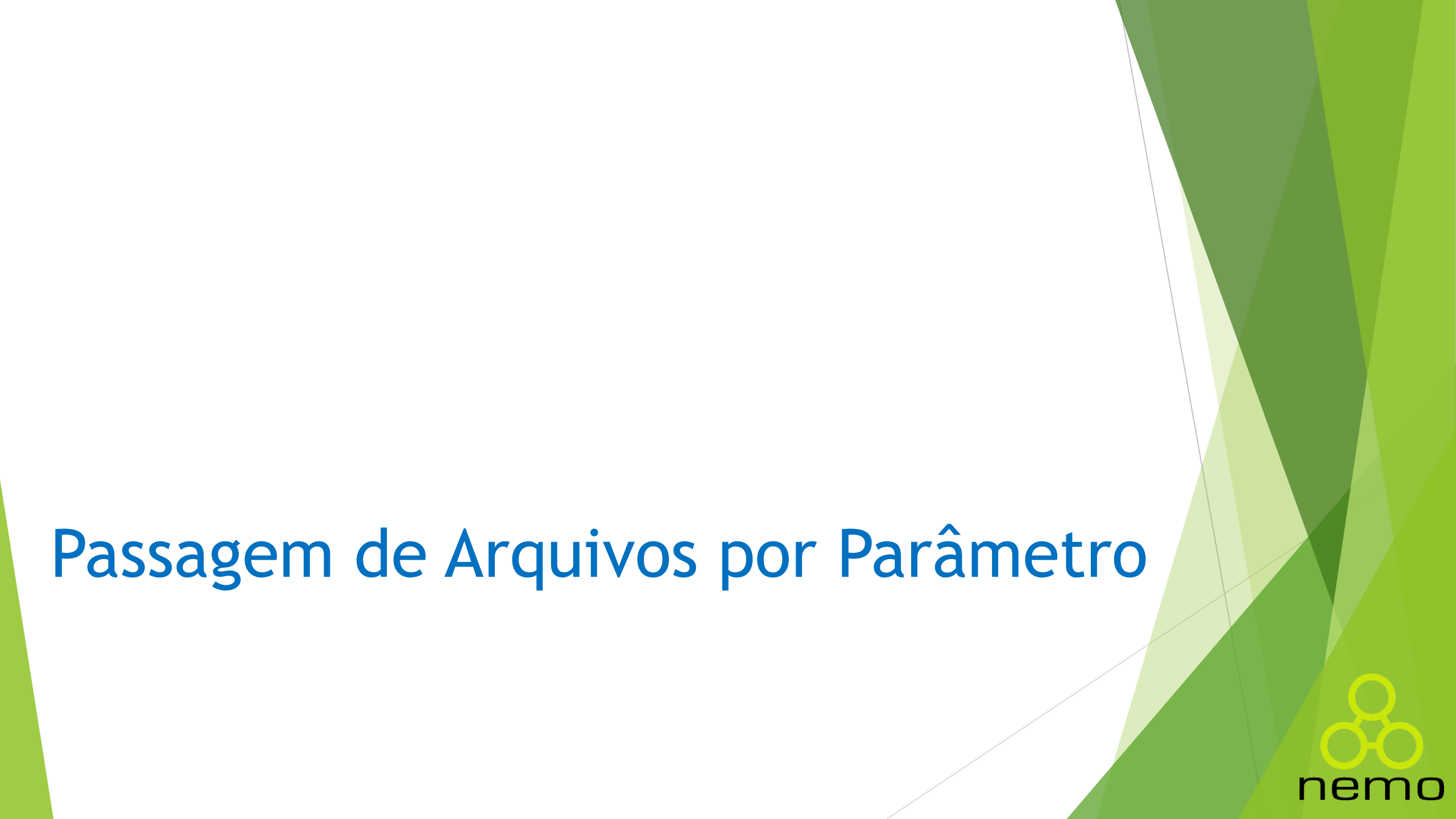
- Para leruma informação em um arquivo, nós utilizamos o operador (>>).

```
#include <fstream>
#include <string>

using namespace std;

int main(){
    int n;
    double y;
    char c;
    string s;

    ifstream infile;
    infile.open("saida.txt");
```

```
    ifstream infile;
    infile.open("saida.txt");

    infile >> n;
    infile >> y;
    infile >> c;
    infile.ignore();
    getline(infile, s);

    cout << "Numero inteiro: ";
    cout << n << endl;
    cout << "Numero real: ";
    cout << y << endl;
    cout << "Caractere: ";
    cout << c << endl;
    cout << "Palavra: ";
    cout << s << endl;

    infile.close();
}
```

nemo

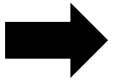
Passagem de Arquivos por Parâmetro



Passagem de Arquivos por Parâmetro

- A linguagem permite que um arquivo, aberto em uma parte do programa, seja passado por parâmetro para um determinado método.

```
int main(){
    Loja *loja;
    loja = new Loja()
    ofstream arq;
    arq.open("Loja.txt");

    loja->salvarDados(arq);
}
```



```
void Loja::salvarDados(ofstream &outfile) {
    outfile << this->nome << endl;
}
```

- Caso o arquivo de parâmetro já tenha sido aberto, ele se mantém aberto na função.
- Desta forma é possível que diferentes métodos escrevam suas informações no arquivo, sem que os dados fiquem fora de ordem.

nemo

Exemplo

```
class Jogo {
public:
    string nome;
    float valor;
    int anoLancamento;
    string urlImagem;

    void imprimeNoArquivo(ofstream &outfile);
    void carregaArquivo(ifstream &infile);
};
```

```
class Loja{
private:
    string nome;
    vector<Jogo*> bibliotecaJogos;
public:
    Loja();
    Loja(string nome);
    void salvarDados(ofstream &outfile);
    void carregarDados(ifstream &infile);
};
```

Arquivos de Saída

```
int main(){
    Loja *loja;
    loja = new Loja()
    ofstream arq;
    arq.open("Loja.txt");

    loja->salvarDados(arq);
}
```

```
void Loja::salvarDados(ofstream &outfile) {
    outfile << this->nome << endl;
    outfile << this->bibliotecaJogos.size() << endl;

    for(int i=0; i< this->bibliotecaJogos.size();i++){
        this->bibliotecaJogos[i]->imprimeNoArquivo(outfile);
    }

    cout << "Loja \"" << this->nome << "\" salvo com sucesso!" << endl;
}
```

```
void Jogo::imprimeNoArquivo(ofstream &outfile) {
    outfile << this->nome << endl;
    outfile << this->valor << endl;
    outfile << this->urlImagem << endl;
    outfile << this->anoLancamento << endl;
}
```

```
PSN
5
Bloodborne
50
https://site.com/images/bloodborne.jpg
2015
God of War
125
https://site.com/image/godofwar.jpg
2018
Dark Souls III
175
https://site.com/darksoul3.jpg
2017
Fifa 19
110
https://site.com/fifa19.jpg
2019
```

nemo

Arquivos de Entrada

```
int main() {
    Loja *loja;
    loja = new Loja();
    ifstream arq;
    arq.open("Loja.txt");
    loja->carregarDados(arq);
}
```

```
void Loja::carregarDados(ifstream &infile) {
    int tam;
    Jogo *jogo;
    getline(infile, this->nome);
    infile >> tam;
    infile.ignore(1, '\n');
    for ( int i = 0; i < tam; i++ ){
        jogo = new Jogo();
        jogo->carregaArquivo(infile);
        this->bibliotecaJogos.push_back(jogo);
    }
    cout << "Loja \"" << this->nome << "\" carregado com sucesso!" << endl;
}
```

```
void Jogo::carregaArquivo(ifstream &infile) {
    getline(infile, this->nome);
    infile >> this->valor;
    infile.ignore();
    getline(infile, this->urlImagem);
    infile >> this->anoLancamento;
    infile.ignore();
}
```

```
PSN
5
Bloodborne
50
https://site.com/images/bloodborne.jpg
2015
God of War
125
https://site.com/image/godofwar.jpg
2018
Dark Souls III
175
https://site.com/darksoul3.jpg
2017
Fifa 19
110
https://site.com/fifa19.jpg
2019
```

nemo

That’s all Folks!