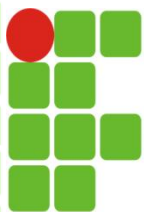


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

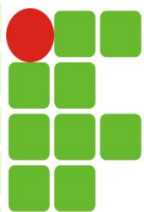
ALGORITMOS II

Prof. Adilso Nunes de Souza



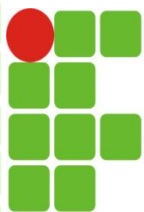
A BIBLIOTECA

- ❑ A linguagem C++ possui um pacote de classes e funções que trabalham com arquivos de forma bastante semelhante às classes cout e cin já estudadas.
- ❑ Biblioteca fstream "file stream" (fluxo de arquivo)
- ❑ `#include <fstream>`



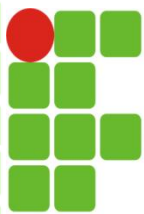
OBJETOS DA BIBLIOTECA

- ❑ A biblioteca `fstream` define objetos que podem tanto ler quanto escrever em arquivos texto:
- ❑ `ofstream`: cria um objeto para escrita
`ofstream <nome do objeto>;`
- ❑ `ifstream`: cria um objeto para leitura
`ifstream <nome do objeto>;`
- ❑ `fstream`: cria um objeto tanto para leitura como para escrita.



ABERTURA DO ARQUIVO

- ❑ Associar o objeto criado a um arquivo, para isso utiliza-se a função "open" que abre ou cria o arquivo.
- ❑ `<objeto>.open("nome do arquivo", tipo de abertura)`
 - `ofstream escreve;`
 - `escreve.open("teste.txt", ios::out);`
- ❑ Também é possível direto ao instanciar o objeto já definir os parâmetros de abertura.
 - `ofstream escreve ("teste.txt", ios::out);`



ABERTURA DO ARQUIVO

❑ Tipos de abertura:

`ios::in` Abre arquivo para leitura.

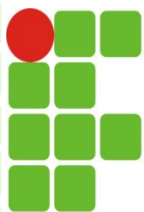
`ios::out` Abre arquivo para escrita.

`ios::ate` Procura o final do arquivo ao abrir ele.

`ios::app` Anexa os dados à serem escritos ao final do arquivo.

`ios::trunc` Trunca os dados existentes no arquivo.

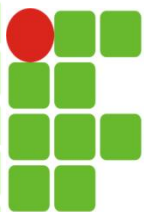
`ios::binary` Abre e trabalha com arquivos em modo binário.



FECHAR

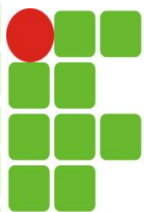
- ❑ Depois de aberto o arquivo deve ser fechado, para isso utilize o método “close”

`escreve.close();`



VERIFICAÇÕES

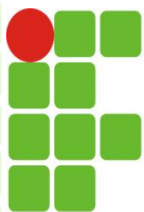
- ☐ A biblioteca fstream possui funções que permitem verificar se um objeto ofstream/ifstream conseguiu abrir um determinado arquivo, e se continua conectado corretamente à este arquivo:



VERIFICAÇÕES

- `<objeto>.is_open()`: Esta função checa o estado do objeto, e retorna `true (1)` para o programa caso tudo esteja certo, ou `false (0)` indicando que o arquivo não pode ser aberto pelo objeto.
- ✓ Exemplo:

```
if(escreve.is_open())  
    cout << "Arquivo aberto com sucesso";  
else  
    cout << "Erro ao abrir o arquivo";
```

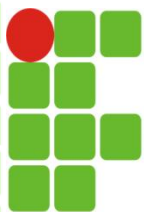



VERIFICAÇÕES

- `<objeto>.good()`: Esta função verifica se o arquivo foi aberto satisfatoriamente retornando `true(1)` caso tenha sido aberto corretamente ou `false(0)` caso não tenha sido aberto.

✓ Exemplo:

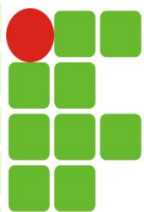
```
if(escreve.good())  
    cout << "Arquivo aberto com sucesso";  
else  
    cout << "Erro ao abrir o arquivo";
```



VERIFICAÇÕES

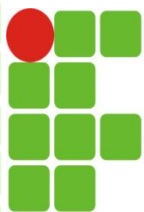
- `<objeto>.fail()`: função que verifica se ocorreu alguma falha com o arquivo ou se ele já está no final, caso não tenha ocorrido falha retorna 0, caso tenha ocorrido falha ou tenha encontrado o final retorna 1.
- ✓ Exemplo:

```
if(escreve.fail() == 0)
    cout << "Arquivo aberto com sucesso";
else
    cout << "Erro ao abrir o arquivo";
```



ESCRITA NO ARQUIVO

- ❑ Após associar o objeto ao arquivo pode-se enviar dados através do objeto "escreve", e estes dados serão escritos no arquivo da mesma forma que utilizamos o comando cout.
- ❑ Utilizando variáveis, strings, formatação, etc.
 - escreve << "Conteúdo que será escrito";
 - escreve << variável;

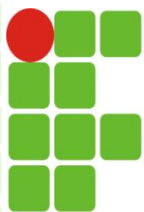


LEITURA DO ARQUIVO

- ❑ Para ler os dados de um arquivo é necessário criar um objeto para este fim e vincular este objeto com o arquivo a ser lido:

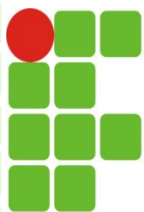
```
ifstream leitura;  
leitura.open("teste.txt", ios::in);
```
- ❑ Após pode-se utilizar o método get com o objeto para ler um caracter do arquivo e armazenar em uma variável

```
leitura.get(variável);
```



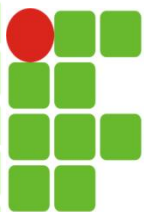
LEITURA DE UM CARACTER

- ❑ É possível utilizar um laço de repetição para ler todos os caracteres do arquivo
char c;
while (leitura.get(c))
{
 cout << c;
}
- ❑ Quando encontrar o final do arquivo o laço é interrompido.



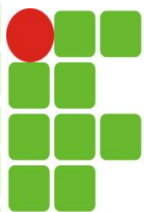
LEITURA DE CONJUNTO DE CARACTERES

- ❑ Também é possível ler a linha inteira, para isso utiliza-se o método `getline`.
- ❑ O método `getline` lê uma linha inteira de entrada, até que o tamanho máximo especificado seja atingido ou até encontrar uma quebra de linha ou o fim de arquivo.
`char conteudo[80];`
`leitura.getline(conteudo, 80);`
- ❑ Da mesma forma que o método `get` é possível colocar a rotina dentro de um laço para ler todo o conteúdo do arquivo.



LEITURA DE CONJUNTO DE CARACTERES

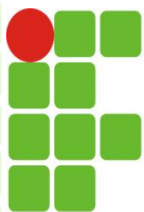
- ❑ O método `getline` pode ser utilizado para ler até encontrar um delimitador, para isso basta especificar um terceiro parâmetro.
`char conteudo[80];`
`leitura.getline(conteudo, 80, `;`);`
- ❑ Desta forma o comando vai ler até encontrar o primeiro "ponto e vírgula";



LEITURA DE UMA PALAVRA

- ❑ Para lermos uma palavra inteira de um arquivo da mesma maneira que o comando `cin`, o objeto lerá todos os caracteres em seu caminho, até que a matriz atinja seu tamanho máximo especificado ou encontre um espaço em branco, uma tabulação, uma quebra de linha ou o fim do arquivo.

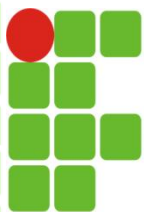
```
char conteudo[80];  
leitura >> conteudo;
```

ENCONTRAR O FIM DO ARQUIVO

- ❑ Para ler todo o arquivo palavra por palavra pode-se utilizar a função `fail()` para indicar se já encontrou o final do arquivo ou não.

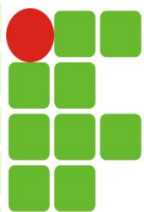
```
leitura >> conteudo;  
while(!leitura.fail())  
{  
    cout << conteudo << endl;  
    leitura >> conteudo;  
}
```



ENCONTRAR O FIM DO ARQUIVO

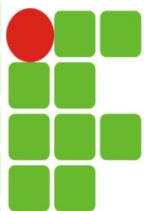
- ❑ Da mesma forma é possível utilizar o operador EOF (End Of File) para identificar se o arquivo já chegou ao final ou não.

```
while (!leitura.eof())  
{  
    leitura >> conteudo;  
    cout << conteudo<< endl;  
}
```



MÉTODO CONSTRUTOR

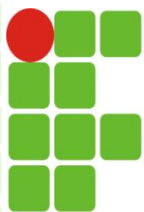
- O método construtor é acionado sempre que ao criar o objeto é informado o parâmetro de abertura, no caso o nome do arquivo a ser aberto;
- Exemplo:
`ofstream escreve("teste3.txt");`
- Mesmo que executar:
`ofstream escreve;`
`escreve.open("teste3.txt", ios::out);`



MÉTODO CONSTRUTOR

- Da mesma forma com os objetos ifstream:
ifstream leitura("teste3.txt");
- Mesmo que executar:
ifstream leitura;
leitura.open("teste3.txt", ios::in);

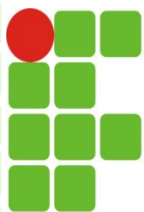
OBS: usando o objeto ofstream o método construtor abre com o formato ios::out, usando o objeto ifstream o formato é ios::in



FINALIZAR ENTRADA EM ARQUIVO

- É possível implementar rotinas para ler informações, gravando diretamente em arquivos texto. O comando CTRL + Z representa a finalização do arquivo:

```
cout << "Entre com o numero e nome\n"  
    << "(Ctrl+Z) termina a entrada de dados\n? ";  
while(cin >> numero >> nome)  
{  
    escreve << numero << ';' << nome << '\n';  
    cout << "Próximo registro? ";  
}
```



LER DADOS COMO MATRIZ

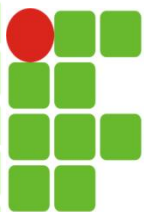
- Imagine um arquivo com números separados por espaço e distribuídos em n linha e n colunas.



*teste5 - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

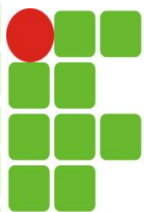
```
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```



LER DADOS COMO MATRIZ

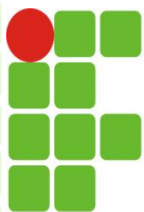
- A rotina a seguir executa a leitura de cada valor e insere na matriz m nas respectivas posições.

```
ifstream ler("exemplo_34.txt");  
if(ler.is_open())  
{  
    for(int l = 0; l < 5; l++)  
    {  
        for(int c = 0; c < 5; c++)  
            ler >> m[l][c];  
    }  
    ler.close();  
}
```



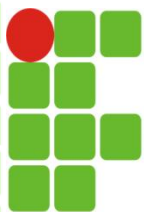
LER DADOS PARA STRUCT

- Para ler dados de um arquivo e armazenar em uma variável do tipo struct, é necessário observar os tipos de dados, lembrando que tudo que é lido do arquivo é texto, para atribuir à campos de outros tipos é necessário realizar as respectivas conversões.
- Observar também a estrutura dos dados no arquivo, para realizar a leitura e escrita corretamente.



LER DADOS PARA STRUCT

- Exemplo: ler dados em um vetor de struct
- Ver exemplo disponibilizado no classroom.



REFERÊNCIAS

- SCHILDT Herbert. C Completo e Total 3ª edição
- KERNIGHAN Brian W. C a linguagem de programação
- ASCENCIO, Ana Fernanda Gomes. CAMPOS, Edilene Aparecida Veneruchi de. *Fundamentos da programação de computadores*. 2 ed. São Paulo: Pearson, 2007.