ALGORITMOS III

Prof. Ms. Ronan Loschi.

ronan.loschi@unifasar.edu.br

31-98759-9555

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

ARQUIVOS EM

C++

DEFINIÇÃO:

A maioria dos programas de computador trabalham com arquivos. Processadores de texto criam e editar arquivos de texto; navegadores de internet interpretam e exibem arquivos HTML; compiladores leêm arquivos-fonte e geram arquivos executáveis.

Quando trabalhamos com arquivos em programação, precisamos de meios para conectar um programa à um arquivo, de modo que o programa possa ler e escrever dados dentro deste arquivo, e também meios para criar novos arquivos e salvá-los em um dispositivo.

DEFINIÇÃO: A biblioteca **fstream**

O primeiro passo para manipular um arquivo em C++ é adicionar a biblioteca específica para a manipulação de dados em arquivos ao cabeçalho de nosso programa.

Esta biblioteca é chamada de “fstream”, de “**file stream**” (fluxo de arquivos).

Para adicioná-la ao cabeçalho, fazemos:

#include <fstream>

DEFINIÇÃO: Os **objetos** de fstream

Uma vez adicionada a biblioteca, podemos criar objetos em nossos programas que servirão de intermediários entre o programa e os arquivos manipulados.

A biblioteca **fstream** define três tipos de objeto para esta função, cada um com objetivo definido:

- **ofstream**: objetos que escrevem dados em um arquivo.
- **ifstream**: objetos que leêm dados em um arquivo.
- **fstream**: objetos que podem tanto ler como escrever em um arquivo

DEFINIÇÃO: **Escrevendo** em um arquivo

Etapas para **escrever** em um arquivo através de um programa em C++:

- Cria-se um objeto do **tipo** ofstream.
- **Associa-se** este objeto com um arquivo (criando ou abrindo um existente).
- Usa-se o objeto para **enviar dados para este arquivo**, de forma bem parecida como usamos o comando cout. A diferença é que os dados vão para o arquivo, ao invés de serem exibidos na tela.
- Para criar um objeto ofstream, declaramos seu nome de maneira parecida com a declaração de uma variável:

SINTAXE: ofstream <nome do objeto>

DEFINIÇÃO: EXEMPLO

ofstream **escreve**;

- A linha acima cria o objeto “escreve”, do tipo ofstream, capaz de escrever em arquivos.
- O próxima passo é associar este objeto a um arquivo. Para isto, utilizamos a função **open( ),** que abre o arquivo desejado ou cria um arquivo no disco rígido

SINTAXE: <objeto>.open(“nome do arquivo”);

EXEMPLO: escreve.open(“strings.txt”);
// cria o arquivo strings.txt

DEFINIÇÃO: EXEMPLO

Dessa forma, estamos associando o objeto escreve ao arquivo strings.txt. Agora podemos enviar dados através do objeto "**escreve**", e estes dados serão escritos no arquivo "**strings.txt**".

Para fazer isso, utilizamos o objeto que criamos da mesma forma que utilizamos o comando **cout**.

PARA **FECHAR** O ARQUIVO:

SINTEXE: <objeto>.close();

EXEMPLO: escreve.**close**( );

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

int main(){
string frase;
 int numero = 100;
 int matriz[3] = {42, 19, 99};

cout<<"Escreva uma frase para ser escrita no arquivo string.txt:";
getline(cin, frase);
cout<<"Obrigado. Escrevendo dados no arquivo strings.txt...\n\n\n\n";

ofstream escreve;
escreve.open("strings.txt");
escreve << "Utilizamos os objetos ofstream para escrever em arquivos\n\n\n";
escreve<< "Note que podemos utilizar os caracteres \\n pra quebrar a linha, como fazíamos em cout\n\n\n";


escreve<<"Podemos escrever o valor de variáveis numéricas: " <<numero <<"\n\n\n";


escreve<<"Podemos também escrever matrizes!";
for (int i=0; i < 3; i++){
escreve.width(6);
escreve<<matriz[i]<<" ";
}

escreve<<"\n";
escreve<<"Finalmente, podemos receber dados via cin e escrever estes dados no arquivo!\n\n\n";

escreve<<"A frase que você digitou durante a execução do programa: "<<frase<<"\n\n\n";
escreve.close();
cout<<"Dados escritos no arquivo. Fim do Programa!";
system("PAUSE");
return 0;
}
```

EXEMPLO 1

NOVO ARQUIVO

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

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

int main(){

 setlocale(LC_ALL, "PORTUGUÊS");
 string frase;
 int numero = 100;
 int matriz[3] = {42, 19, 99};

cout<<"Escreva uma frase para ser escrita no arquivo string2.txt:";
getline(cin, frase);
cout<<"Obrigado. Escrevendo dados no arquivo strings2.txt...\n\n\n\n";

ofstream escreve;
escreve.open("strings2.txt");
escreve << "Utilizamos os objetos ofstream para escrever em arquivos\n\n\n";
escreve<< "Note que podemos utilizar os caracteres \\n pra quebrar a linha, como fazíamos em cout\n\n\n";


escreve<<"Podemos escrever o valor de variáveis numéricas: " <<numero <<"\n\n\n";


escreve<<"Podemos também escrever matrizes!";
for (int i=0; i < 3; i++){
escreve.width(6);
escreve<<matriz[i]<<" ";
}

escreve<<"\n";
escreve<<"Finalmente, podemos receber dados via cin e escrever estes dados no arquivo!\n\n\n";

escreve<<"A frase que você digitou durante a execução do programa: "<<frase<<"\n\n\n";
escreve.close();
cout<<"Dados escritos no arquivo. Fim do Programa!";
system("PAUSE");
return 0;
}
```

EXEMPLO 2

COM ARQUIVO EXISTENTE

OBS. ANTES DE COMPILAR CRIE O ARQUIVO strings2.txt na pasta do projeto.

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

DEFINIÇÃO: Checando se o arquivo abriu sem problemas

Sempre existe a possibilidade de erros quando trabalhamos com arquivos. Talvez o arquivo que desejamos ler tenha sido **apagado, ou renomeado**, ou esteja **sendo usado** por outro programa.

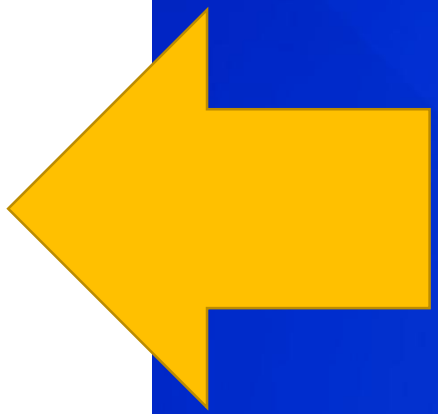
Para checar se um objeto conseguiu abrir um determinado arquivo, e se continua conectado corretamente à este arquivo: é a função **is_open( ).**

SINTEXE:

```
if (!nomedoobjeto.is_open()) {
        cout <<“Não foi possível abrir o arquivo!”
         nomedoobjeto.clear();
         return 0;
        }
```

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

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

int main(){

 setlocale(LC_ALL, "PORTUGUÊS");
 string frase;



 ofstream escreve;
escreve.open("strings2.txt");

if (!escreve.is_open()) {
      cout <<"Não foi possível abrir o arquivo! \n\n\n";
      escreve.clear();
      return 0;
   }  else {
     cout <<"Tudo certo, foi possível abrir o arquivo! \n\n\n";
     }


cout<<"Escreva uma frase para ser escrita no arquivo string2.txt:";
getline(cin, frase);
cout<<"Obrigado. Escrevendo dados no arquivo strings2.txt...\n\n\n\n";
escreve<<"A frase que você digitou durante a execução do programa: "<<frase<<"\n\n\n";
escreve.close();
cout<<"Dados escritos no arquivo. Fim do Programa!";
system("PAUSE");
return 0;
}
```



UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

DEFINIÇÃO: Lendo os dados de um arquivo

Para **ler** os dados de um arquivo em um programa, é preciso antes de mais nada criar um objeto do tipo **ifstream**, capaz de ser o intermediário entre o arquivo à ser lido e o programa.

SINTAXE:

```
ifstream leitura;
leitura.open("string.txt");
```

Após criarmos o objeto e conectarmos ele à um arquivo, podemos **começar a ler** através deste arquivo.

DEFINIÇÃO: Lendo os dados de um arquivo

Para lermos somente **um caractere** de um arquivo, por exemplo, nossa melhor opção é utilizar o método **.get**, que serve justamente para esta função. Criamos uma variável do tipo char para armazenar este caractere, e utilizamos o objeto ifstream da forma mostrada no exemplo abaixo:

```
char armazena;
 leitura.get (armazena);
```

O exemplo de código acima utiliza o objeto “leitura” e o método .get para ler um caractere de um arquivo, **e armazena este caractere na variável “armazena”.**

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

DEFINIÇÃO: Lendo os dados de um arquivo

leitura.clear(); // Limpa possíveis flags de erro

leitura.seekg(0); // Volta para o início do arquivo

```
leitura.get(letra); // lê apenas o 1 caracter
cout << letra;
leitura.get(letra); // lê apenas o 2 caractar
cout << letra;
leitura.get(letra); // lê apenas o 3 caractar
cout << letra;
cout<<"\n";
leitura.clear(); //reseta o objeto leitura, para limpar memória do sistema}
leitura.seekg(0); // Volta o pnteiro para o início do arquivo
while (leitura.get(letra)) { // lê todos os caractares não lidos
        cout << letra;
        }
```

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

```
#include <iostream>
 #include <fstream>
 #include <string>
 using namespace std;
 int main(){
 char letra;
 ifstream leitura;
 leitura.open("strings.txt");
 if(!leitura.is_open( ))
{
 cout<<"Não foi possível abrir arquivo! Programa será terminado!\n";
 leitura.clear( ); //reseta o objeto leitura, para limpar memória do sistema}
 }
 leitura.get(letra); // lê apenas o 1 caracter
 cout << letra;
 leitura.get(letra); // lê apenas o 2 caractar
 cout << letra;
 leitura.get(letra); // lê apenas o 3 caractar
 cout << letra;
 cout<<"\n";
 leitura.clear(); //reseta o objeto leitura, para limpar memória do sistema}
 leitura.seekg(0); // Volta o pnteiro para o início do arquivo
 while (leitura.get(letra)) { // lê todos os caractares não lidos
        cout << letra;
        }
 leitura.close();
 system("PAUSE");
 return 0;
 }
```

DEFINIÇÃO:

Lendo os dados de um arquivo

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

DEFINIÇÃO: Lendo os dados de um arquivo

Para lermos **uma palavra inteira** de um arquivo, ao invés de utilizarmos uma simples variável de tipo char, **utilizamos uma matriz do tipo string**. 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 quebra de linha ou o fim do arquivo.

string matriz_chars;
leitura >> matriz_chars

```cpp
#include <iostream>
 #include <fstream>
 #include <string>
 using namespace std;
 int main(){
 string palavra;
 ifstream leitura;
 leitura.open("strings.txt");
 if(!leitura.is_open( )){
   cout<<"Não foi possível abrir arquivo! Programa será terminado!\n";}
 leitura >> palavra; // lê apenas o 1 palavra
 cout << palavra;
 leitura >> palavra; // lê apenas o 2 palavra
 cout <<" "<< palavra;
 leitura >> palavra; // lê apenas o 3 palavra
 cout <<" "<< palavra;
 cout<<"\n";
 while (leitura >> palavra) { // lê todos os caractares não lidos um a um
        cout << palavra <<" " << endl;}
 leitura.clear(); //reseta o objeto leitura, para limpar memória do sistema}
 leitura.seekg(0); // Volta o pnteiro para o início do arquivo
  while (getline(leitura, palavra)) { // lê todos os caractares não lidos linha a linha
        cout << palavra << endl;}  // Imprime a linha e adiciona a quebra de linha
 leitura.close();
 system("PAUSE");
 return 0;
}
```

DEFINIÇÃO: Lendo os dados de um arquivo

Finalmente, para lermos uma linha inteira de um arquivo, utilizamos o método **.getline**

getline (objeto, string);

<nome do objeto>.getline (<matriz_destino>, <limite de caracteres>);

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

using namespace std;
int main(){
string palavra;
ifstream leitura;
leitura.open("strings.txt");
if(!leitura.is_open( )){
  cout<<"Não foi possível abrir arquivo! Programa será terminado!\n";}
getline(leitura, palavra); // lê apenas a linha 1
cout << palavra <<"\n";
getline(leitura, palavra); // lê apenas a linha 2
cout <<" "<< palavra <<"\n";
getline(leitura, palavra) ; // lê apenas a linha 3
cout <<" "<< palavra <<"\n";
cout<<"\n\n\n";

leitura.clear(); //reseta o objeto leitura, para limpar memória do sistema}
leitura.seekg(0); // Volta o pnteiro para o início do arquivo
while (getline(leitura, palavra)) { // lê todas as linhas
    cout << palavra << endl;}  // Imprime todas a linhas e adiciona a quebra de linha
leitura.close();
system("PAUSE");
return 0;
}
```

DEFINIÇÃO: MODOS DE ARQUIVO

Quando abrimos um arquivo, podemos querer fazer diferentes coisas com o conteúdo deste arquivo. Podemos querer **continuar à escrever no fim do arquivo**, dando continuidade aos dados já escritos nele. Podemos querer **apagar todos os dados já escritos** e começar do zero novamente. Os modos de arquivo servem justamente para isto

SINTAXE:

<objeto ofstream>.open(“nome do arquivo”, ofstream::<**modo de arquivo**>);
ou
<objeto ifstream>.open(“nome do arquivo”, ofstream::<**modo de arquivo**>);

A tabela abaixo resume os modos de arquivo disponíveis em C++.

`ios_base::in`	Abre arquivo para leitura.
`ios_base::out`	Abre arquivo para escrita.
`ios_base::ate`	Procura o final do arquivo ao abrir ele.
`ios_base::app`	Anexa os dados à serem escritos ao final do arquivo.
`ios_base::trunc`	Trunca os dados existentes no arquivo.
`ios_base::binary`	Abre e trabalha com arquivos em modo binário.

Note que um arquivo aberto por um objeto ofstream não necessita que definamos o modo de arquivo **ofstream::out**

O mesmo ocorre com o modo **ifstream::in** e os objetos **ifstream**.

```
#include <iostream>
 #include <fstream>
 #include <string>
 using namespace std;
 int main(){
 int dia, mes;
 char letra[1000];
cout<<"PROGRAMA AGENDA Versão 0.00042\n";
 ofstream agenda;
 agenda.open("agenda.txt", ofstream::app);
 cout<<"Digite o compromisso no espaço abaixo (): \n";
 cin.getline(letra, 1000);
 cout << "Digite o dia do compromisso (1-seg. 2-terc ...):\n";
 cin >> dia;
 cout << "Digite o mês do compromisso: 1--jan, 2-fev\n";
 cin >> mes;
 agenda << "Compromisso marcado para dia "<<dia<<" de"<<mes<<": ";
 agenda << letra;
 agenda<<"\n";
 cout<<"Obrigado! Sua agenda foi atualizada com sucesso!";
 agenda.close();
 system("PAUSE");
 return 0;
}
```

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

Exemplo: somar vetor

```
#include <iostream>
#include <fstream>
using namespace std;
/* run this program using the console pauser or add yo

int main(  ) {
setlocale(LC_ALL, "Portuguese");
int vetnum [5], i, soma;
soma=0;

 for (i=0; i<4; i++){
    cout <<"Digite o "<< i+1 <<"º valor:";
    cin >> vetnum[i];
 }

ofstream escrever;
escrever.open("vetordenum.txt", ios::out);

 for (i=0; i<4; i++){
    escrever<<vetnum[i]<<" ";
}
escrever.close();

ifstream leitura;
leitura.open("vetordenum.txt", ios::in);

for (i=0; i<4; i++){
    leitura >> vetnum[i];
    soma= soma + vetnum[i];

}
cout <<" \n\n O valor da soma é = "<< soma <<"\n\n";

    system("PAUSE");
    return 0;
}
```

- Criar objeto de escrita
- Abrir o arquivo
- Escrever no arquivo
- Fechar o arquivo

- Criar objeto de leitura
- Abrir o arquivo
- Ler no arquivo
- Somar e imprimir
- Fechar o arquivo

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

```cpp
#include <iostream>
#include <fstream>
using namespace std;
/* run this program using the console pauser or add your own getch, system("pause") or input loop */

int main(  ) {
setlocale(LC_ALL, "Portuguese");
int vetnum [5][5], i,j, soma;
float media;
soma=0;

 for (i=0; i<5; i++){
    for (j=0; j<5; j++){
    cout <<"Digite o "<< i+1 <<"º valor:";
    cin >> vetnum[i][j];
 }
 }

ofstream escrever;
escrever.open("matrizdenum.txt", ios::out);

 for (i=0; i<5; i++){
    escrever.width(3); // Define o espaçamento entre os números
    for (j=0; j<5; j++){
    escrever<<vetnum[i][j]<<" "; // Quebra de linha após cada linha da matriz
}
escrever << endl;
}
escrever.close();

ifstream leitura;
leitura.open("matrizdenum.txt", ios::in);

for (i=0; i<5; i++){
    cout<<"\n";
    for (j=0; j<5; j++){
    leitura >> vetnum[i][j];
    soma= soma + vetnum[i][j];
}
}
media=soma/25;
cout <<" \n\n A média é = "<< media <<"\n\n";

    system("PAUSE");
    return 0;
}
```

Exemplo: media matriz

- Criar objeto de escrita
- Abrir o arquivo
- Escrever no arquivo
- Fechar o arquivo

- Criar objeto de leitura
- Abrir o arquivo
- Ler no arquivo
- Somar e imprimir
- Fechar o arquivo

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

```
#include <iostream>
#include <fstream>
using namespace std;
/* run this program using the console pauser or ad

int main(  ) {
setlocale(LC_ALL, "Portuguese");
int vetnum [5][5], i,j, soma, cont, maior, menor;
float media;
soma=0; cont=1; maior=-9999; menor=+99999;

 for (i=0; i<5; i++){
    for (j=0; j<5; j++){
    cout <<"Digite o "<< cont <<"º valor:";
    cin >> vetnum[i][j];
    cont++;
 }
 }

ofstream escrever;
escrever.open("matrizdenum.txt", ios::out);

 for (i=0; i<5; i++){
    escrever.width(3); // Define o espaçamento ent
    for (j=0; j<5; j++){
    escrever<<vetnum[i][j]<<" "; // Quebra de linh
}
escrever << endl;
}
escrever.close();
```

```
ifstream leitura;
leitura.open("matrizdenum.txt", ios::in);

for (i=0; i<5; i++){
    for (j=0; j<5; j++){
    leitura >> vetnum[i][j];
    if (vetnum[i][j] > maior){
        maior = vetnum[i][j];
    }
    if (vetnum[i][j] < menor){
        menor=vetnum[i][j];
    }
}
}
media=soma/25;
cout <<" \n\n O maior é = "<< maior <<"\n\n";
cout <<" \n\n O menor é = "<< menor <<"\n\n";

    system("PAUSE");
    return 0;
}
```

Exemplo: MAIOR. MENOR

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

EXRCÍCIOS

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

1. Registro de Temperaturas Semanais - Aplicação em Meteorologia
Você é um meteorologista e foi solicitado a registrar as temperaturas durante a semana, divididas em três períodos do dia (manhã, tarde e noite). O objetivo é armazenar essas informações em um arquivo e, depois, calcular a maior, menor e média das temperaturas para análise.

2. Notas de Alunos em uma Turma - Aplicação Educacional
Você é um professor e está registrando as notas dos alunos em três provas. Depois de gravar essas notas, o programa calcula a média de cada aluno e armazena esses dados em um arquivo.

3. **Registro de Estoque de Produtos - Aplicação Comercial**
Você trabalha em uma loja e precisa registrar a quantidade e o preço de cada produto em estoque. O programa armazena esses dados e calcula o valor total do estoque.

4. Relatório de Viagens de um Motorista

Você é um motorista de aplicativo e precisa registrar a quilometragem e o valor arrecadado por dia durante uma semana. O objetivo é calcular o total de quilômetros percorridos e o faturamento total.

5. Contador de Palavras em um Arquivo de Texto

Você é um analista de dados e precisa contar o número de palavras em um arquivo de texto. Esse processo pode ser usado, por exemplo, para analisar artigos, documentos ou qualquer outro tipo de texto.

6. Cadastro de Funcionários em RH

O sistema de recursos humanos (RH) deve registrar as informações de funcionários, gravar em um arquivo e, depois, fazer a leitura desses dados. Utilizando structs para representar um funcionário e alocação dinâmica para armazenar os dados, o programa grava as informações no arquivo e lê esses dados posteriormente.

7. Cadastro de Livros em uma Biblioteca - Aplicação em Gestão de Biblioteca:

Um sistema para cadastrar livros em uma biblioteca, onde as informações de cada livro são gravadas em um arquivo de texto e depois lidas para exibição.

OBRIGADO!

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA