BCC 201 - Introdução à Programação I

# Arquivos de Texto

Guillermo Cámara-Chávez
UFOP

# Arquivos I

- Podem armazenar grande quantidade de informação
- Dados são persistentes (gravados em disco)
- Acesso aos dados pode ser não sequencial (acesso direto a registros em um banco de dados)
- Acesso à informação pode ser concorrente (mais de um programa ao mesmo tempo)

# Nomes e extensões I

| | |
|---|---|
| `arq.txt` | arquivo texto simples |
| `arq.c` | código fonte em C |
| `arq.pdf` | *portable document format* |
| `arq.html` | arquivo para páginas WWW |

# Tipos de arquivos I

- Arquivo texto - armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de textos simples. Por exemplo:
  `*** Arquivo texto. ***`
- Arquivo binário - sequência de bits sujeita às convenções dos programas que o gerou (exemplos arquivos executáveis, arquivos compactados, arquivos de registros).

# Diretório I

- também chamada de pasta
- contém arquivos e/ou outros diretórios

# Caminhos absolutos ou relativos I

- Caminho absoluto
  - Descrição de caminho desde o diretório raiz:

```
/bin/emacs
/home/usr/arq.txt
```

- Caminho relativo
  - Descrição de um caminho desde o diretório corrente:

```
arq.txt
mc102/lab.c
```

- C++ fornece três classes para lidar com arquivos:
  - **ofstream**: para escrever em arquivos ("o" = *output*);
  - **ifstream**: para ler arquivos ("i" = *input*);
  - **fstream**: para ler e/ou escrever em arquivos.
- O uso destas classes é bastante similar ao uso de **cin** e **cout**.
- Para trabalhar com essas classes precisamos usar a biblioteca `<fstream>`
- *Stream*: é uma seqüência de bytes

# Classe *fstream* I

- Declaração

```
fstream arquivo;
```

- Abrir arquivo só para leitura:

```
arquivo.open("meuArquivo.txt", ios::in);
```

- Abrir arquivo só para escrita:

```
arquivo.open("meuArquivo.txt", ios::out);
```

- Abrir arquivo para escrita e leitura:

```
arquivo.open("meuArquivo.txt", ios:in|ios:out);
```

# Modos de abertura de arquivos I

- Especifica como o arquivo deve ser **aberto** e o que **pode ser feito** com ele
- `ios:in` e `ios:out` são exemplos de modos de abertura de arquivos
- *Flags* de abertura de arquivo podem ser combinados na abertura

# Modos de abertura de arquivos II

- *Flags* de abertura

| | |
|---|---|
| `ios::app` | Cria novo arquivo, ou adiciona ao final de um arquivo existente |
| `ios::ate` | Vai para o final do arquivo; escreve em qualquer lugar |
| `ios::binary` | Lê/escreve em modo binário |
| `ios::in` | Abre para leitura |
| `ios::out` | Abre para escrita |

# Modos de abertura de arquivos III

- `ifstream` e `ofstream` possuem modos *default*
- O **segundo parâmetro** da função `open` é **opcional** quando utilizado `ifstream` e `ofstream`

# Modos *Default* I

- `ofstream`:
  - Abertura somente para escrita
  - Não são permitidas leituras
  - Se não existe o arquivo é criado
  - O conteúdo é apagado caso o arquivo exista

# Modos *Default* II

- `ifstream`:
  - Abertura somente para leitura
  - Não é permitido escrever no arquivo
  - A abertura falha caso o arquivo não exista

# Escrita e Leitura I

- Leitura e Escrita simultânea em arquivos
  - Leitura de dados do arquivo: memória
  - Atualização de dados
  - Escrita dos dados atualizados no arquivo

# Arquivo de Entrada I

```
#include <fstream>
int main(){
    ...
    ifstream arqIn;
    ...
}
```

- `arqIn.open("nomeArquivo.txt");`
  - Conecta a *stream arqIn* ao arquivo `"nomeArquivo.txt"`.
- `arqIn.close()`
  - Desconecta a *stream* do arquivo associado.
- `arqIn >> c;`
  - Comportamento idêntico ao `cin`.

# Arquivo de Saída I

```
#include <fstream>
int main(){
    ...
    ofstream arqOut;
    ...
}
```

- `arqOut.open("nomeArquivo.txt");`
  - Conecta a *stream arqOut* ao arquivo `"nomeArquivo.txt"`.
- `arqOut.close()`
  - Desconecta a *stream* do arquivo associado.
- `arqOut << c;`
  - Comportamento idêntico ao `cout`.

# Exemplo I

Ler três numeros salvos no arquivo `numeros.dat`, somar o mesmos e escrever a seguinte mensagem em outro arquivo

```
A soma dos 3 primeiros números
no arquivo numeros.dat
eh soma
```

# Exemplo II

```
// Lê tres números do arquivo numeros.dat,
// soma os numeros, e escreve o resultado em soma.dat.
#include <iostream> // para cin, cout
#include <fstream> // para ifstream, ostream
int main()
{
    // declaração das streams de entrada e saída
    ifstream input_stream;
    ofstream output_stream;
    // abertura do arquivo de entrada
    input_stream.open("numeros.dat");
    // abertura do arquivo de saída
    output_stream.open("soma.dat");
```

# Exemplo III

```
    int num1, num2, num3;
    // leitura dos dados (3 numeros)
    input_stream >> num1 >> num2 >> num3;
    // escrita no arquivo de saída
    output_stream << "A soma dos 3 primeiros números\n"
                  << "no arquivo numeros.dat\n"
                  << "é " << (num1 + num2 + num3)
                  << endl;
    // fecha os arquivos de entrada e saída
    input_stream.close();
    output_stream.close();
    return 0;
}
```

# Lendo um vetor de um arquivo I

Lêr um vetor de números inteiros que foi salvo dentro um arquivo de texto. O arquivo possui todos os elemento do arquivo assim como a dimensão do vetor. O formato do arquivo é como segue:

```
dimensao_vetor elem_1 elem_2 elem_3 ... elem_n
```

# Lendo um vetor de um arquivo II

```
#include<iostream>
#include<fstream>
int main(){
    ifstream arqIn;
    int i, n, *v;
    arqIn.open("vetor.txt");
    arqIn >> n;
    v = new int[n];
    for(i = 0; i < n; i++)
        arqIn >> v[i];
    for(i = 0; i < n; i++)
        cout << v[i] << " ";
    arqIn.close();
    if (n != nullptr) delete[] v;
    return 0;
}
```

# Escrevendo um vetor em um arquivo I

Escrever um vetor de inteiro em um arquivo seguindo o formato da questão anterior

# Escrevendo um vetor em um arquivo II

```
int* Inserir(int n);
void EscreverVetor(ofstream& file, int* v, int n);

int main()
{
    ofstream arqOut;
    int n, *v;
    arqOut.open("vetor.txt");
    if (arqOut.isopen())
    {
        cout << "Inserir dimensao do vetor \n";
        cin >> n;

        v = Inserir(n); // Inserir dados no vetor
        EscreverVetor(fw, v, n);// escrita no arquivo

        arqOut.close();
        if (v != nullptr) delete[] v;
    }
    return 0;
}
```

# Escrevendo um vetor em um arquivo III

```
int* Inserir(int n)
{
    int i, *vet = nullptr;
    vet = new int[n];
    cout << "Inserir " << n << " numeros \n";
    for (i = 0; i < n; i++)
    {
        cin >> vet[i];
    }
    return vet;
}
```

# Escrevendo um vetor em um arquivo IV

```
void EscreverVetor(ofstream& pfile, int* v, int n)
{
    int i;
    pfile << n <<" "; // dimensao
    for(i = 0; i < n; i++)
        pfile << v[i] <<" ";
}
```

# Escrevendo uma matriz em um arquivo I

Criar uma matriz de $m \times n$ e salve ela em um arquivo. Use o mesmo formato que no exercício anterior. Primeiro salve o tamanho da matriz e depois os dados

# Escrevendo uma matriz em um arquivo II

```
int** Inserir(int lin, int col);
void EscreverMatriz(ofstream& arqOut, int lin, int col,
                    int** m);
int main()
{
    ofstream arqOut;
    int lin, col, **m = NULL;
    arqOut.open("matriz.txt");
    cout << "Inserir dimensao da matriz \n";
    cin << lin << col;

    m = Inserir(lin, col); // Inserir dados no vetor
    EscreverMatriz(arqOut, lin, col, m);// escrita no arquivo

    arqOut.close();
    if (m[0] != nullptr) delete[] m[0];
    if (m != nullptr) delete[] m;
    return 0;
}
```

# Escrevendo uma matriz em um arquivo III

```
int** Inserir(int lin, int col)
{
    int i, j, **mat = NULL;
    mat = new int*[lin];
    mat[0] = new int[lin*col];
    for (i = 1; i < lin; i++)
        mat[i] = &mat[0][col*i];

    arqOut << "Inserir " << lin*col <<" numeros\n";

    for (i = 0; i < lin; i++)
        for (j = 0; j < col; j++)
            cin >> mat[i][j];
    return mat;
}
```

# Escrevendo uma matriz em um arquivo IV

```
void EscreverMatriz(ofstream& arqOut, int lin, int col,
                    int** mat)
{
    int i, j;
    arqOut << lin << " " << col << endl; // dimensao
    for(i = 0; i < lin; i++)
    {
        for(j = 0; j < col; j++)
            arqOut << mat[i][j] << " ";
        arqOut << endl;
    }
}
```

# Exercícios I

Criar uma matriz de $10 \times 10$. Preencher a matriz com valores aleatórios, depois salvá-la em um arquivo. Cada número deverá ter 4 decimais.

# Exercícios II

```
#define LIN 10
#define COL 10

void Preenche(float M[LIN][COL]);
void Escreve(ofstream &arqOut, float M[LIN][COL]);

int main()
{
    ofstream fw;
    float num[LIN][COL];
    fw.open("random.txt");
    srand(time(NULL));
    Preenche(num);
    Escreve(fw, num);
    fw.close();
    return 0;
}
```

# Exercícios III

```
void Escreve(ofstream &pfile, float M[LIN][COL])
{
    int i, j;
    for (i = 0; i < LIN; i++)
    {
        for (j = 0; j < COL; j++)
            pfile << fixed << precision(4) << M[i][j] <<" ";
        pfile << endl;
    }
}

void Preenche(float M[LIN][COL])
{
    int i, j;
    for (i = 0; i < LIN; i++)
        for (j = 0; j < COL; j++)
            M[i][j] = static_cast<float>(rand()) / RAND_MAX;
}
```

# Exercícios IV

Gere uma estrutura `Aluno` com os seguintes dados: nome, idade, nota1, nota2. Insira $n$ alunos, depois calcular a média de notas para cada aluno. Finalmente, salvar em um arquivo os dados da estrutura e a média de notas. Repita o processo para cada aluno.

# Exercícios V

```
typedef struct Aluno{
    string nome;
    int idade;
    double nota1, nota2;
}ALU;

void Inserir(ALU* alunos, int n);
void Escreve(ofstream &arqOut, ALU* alunos, int n);
```

# Exercícios VI

```
int main()
{
    ofstream fw;
    ALU* alu = nullptr;
    int n;
    cout  << "Inserir numero de alunos";
    cin >> n;
    alu = new ALU[n];

    fw.open("aluno.txt");
    Inserir(alu, n);
    Escreve(fw, alu, n);
    fw.close();
    if (alu != nullptr) delete[] alu;

    return 0;
}
```

# Exercícios VII

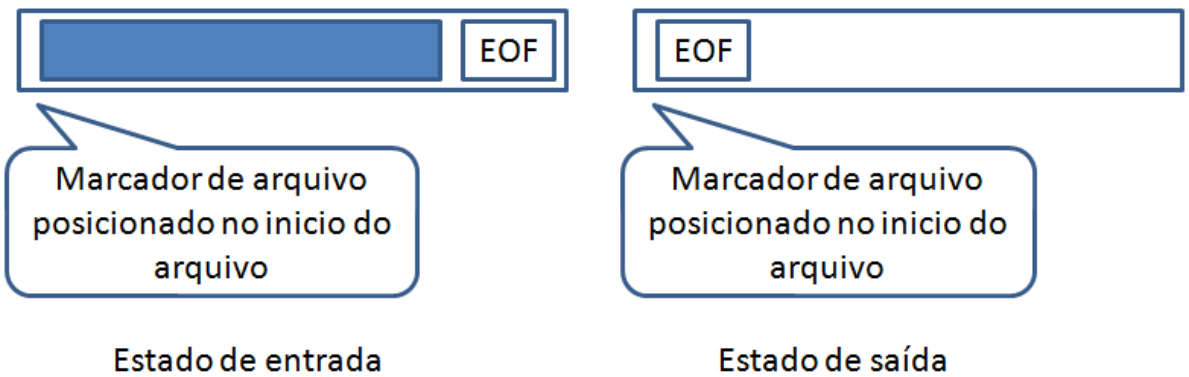
```
void Escreve(ofstream &file, ALU *alu, int n)
{
    int i, pos;
    for (i = 0; i < n; i++)
    {
        file << alu[i].nome << " "
             << alu[i].idade << " "
             << alu[i].nota1 << " "
             << alu[i].nota2 << " "
             << (alu[i].nota1+alu[i].nota2)/2.0
             << endl;
    }
}
```

# Exercícios VIII

```
void Inserir(ALU* alu, int n)
{
    int i;
    for (i = 0; i < n; i++)
    {
        cout << "\n Cadastro Aluno " << i+1;
        cout << "\n Inserir nome: ";
        cin.ignore(100, '\n');
        getline(cin, alu[i].nome);
        cout << "Inserir idade: ";
        cin >> alu[i].idade;
        cout << "Inserir duas notas: ";
        cin >> alu[i].nota1 >> alu[i].nota2;
    }
}
```

# Outras funções I

- `ifstream arqIn;`
  - `arqIn.get(char& character)`: extrai o próximo caracter da *stream* `arqIn` e coloca na variável `character`
- `ofstream arqOut;`
  - `arqOut.put(char character)`: insere o caracter `character` na *stream* de saída `arqOut`
- `arqIn.eof()/arqOut.eof()`
  - Testa condição de fim de arquivo



# Outras funções II

Conta o número de brancos em cada linha do Arquivo “test.txt”

```
a b c
top10 methods to count spaces

1 3
```

Saída:

```
a b c
Brancos: 2
top10 methods to count spaces
Brancos: 4
Brancos: 0
1 3
Brancos: 1
```

# Outras funções III

```
int main(){
    ifstream arqIn;  char c;
    int cont;
    arqIn.open("test.txt");
    while(!arqIn.eof())
    {
        arqIn.get(c);
        cont = 0;
        while(c != '\n' && !arqIn.eof())
        {
            cout << c;
            if (c == ' ') cont++;
            arqIn.get(c);
        }
        cout << endl << "brancos: " << cont <<endl;
    }
    arqIn.close();
    return 0;
}
```

# Exercicios Propostos I

1. Considere um arquivo que possui uma lista de pares de números reais (valores de $x$ e $y$). Cada linha deve possuir um par de valores separados por tabulação. Elabore um programa para calcular o valor da função $z = x^2 + y^2$, para cada par de valores, e colocar a tabela $x$ $y$ $z$ resultado em outro arquivo. Este outro arquivo deve ter, em cada linha, uma tripla de valores $x$ $y$ $z$ separados por tabulação (caracter `'\t'`)
2. Elabora um programa para ler um arquivo de texto de, no máximo, 100 linhas e criar um arquivo com as linhas de texto em ordem inversa. Utilize para tanto um vetor de *strings* para armazenar temporariamente as linhas. DICA, utilize a função `getline` para ler uma linha completa do arquivo.

# FIM