Estruturas de dados II

Revisão de arquivos em C++

André Tavares da Silva

andre.silva@udesc.br

UDESC

Arquivos em C++

- Em C++ temos três classes para trabalhar com arquivos:
  - **ofstream**: usada para criar e gravar informações em arquivos, representando o fluxo de saída;
  - **ifstream**: Usada para ler informações de arquivos de dados, representando o fluxo de entrada;
  - **fstream**: Combina os recursos de ifstream e ofstream, sendo capaz de criar arquivos, gravar arquivos e ler arquivos de dados.
- Fazem parte da biblioteca <fstream>:

```
#include <stream>
```

Abrindo Arquivos em C++

- Antes de realizar qualquer operação em um arquivo, é preciso abrir ele. Caso seja necessário gravar dados no arquivo, deve-se usar as classes **fstream** ou **ofstream**. Caso precise apenas ler a informação do arquivo, abra o mesmo usando a classe **ifstream**.

```
open(<file_name>, <mode>);
```

- O parâmetro <*file_name*> é o nome (e caminho) do arquivo a ser aberto;
- O parâmetro <*mode*> é opcional e pode assumir os diferentes valores (veremos a seguir).

UDESC

Abrindo Arquivos em C++

- O parâmetro <mode> assume um valor ou a combinação dos seguintes valores:
- **ios::in** – abre o arquivo para leitura;
- **ios::out** – abre arquivo para escrita;
- **ios:binary** – abre arquivos que não são de texto (são arquivos binários);
- **ios::ate** – coloca posição de leitura/escrita para para o final do arquivo;
- **ios:app** – todas as operações de escrita devem ser feitas no final do arquivo (*append*)
- **ios::trunc** – se o arquivo aberto para leitura já existia anteriormente, seu conteúdo é apagado.

Exemplo 1

```
#include <iostream> //biblioteca para operações de E/S em C++
#include <string> //biblioteca para a classe string

using namespace std;

int main (){
  // Arquivo apenas de saída em C++
  ofstream arq1;

  // Abre o arquivo e cria caso não exista
  arq1.open("nomes.txt");

  // insere nomes no arquivo (usando operador << )
  arq1 << "Fulano de Tal" << endl;
  arq1 << "João da Silva" << endl;

  // Fecha o arquivo
  arq1.close();

  return (0);
}
```

Exemplo 1

```
#include <iostream> //biblioteca para operações de E/S em C++
#include <string> //biblioteca para a classe string

using namespace std;

int main (){
  // Arquivo apenas de saída em C++
  ofstream arq1;

  // Abre o arquivo e cria caso não exista
  arq1.open("nomes.txt");

  // insere nomes no arquivo (usando operador << )
  arq1 << "Fulano de Tal" << endl;
  arq1 << "João da Silva" << endl;
```

Similar a std::cout, mas agora usa um nome de arquivo em vez da saída padrão

```
  // Fecha o arquivo
  arq1.close();

  return (0);
}
```

Exemplo 2

```
#include <iostream> //biblioteca para operações de E/S em C++
#include <string> //biblioteca para a classe string

using namespace std;

int main (){
  // Arquivo a ser aberto já é indicado após o nome da variável
  ofstream arq1("nomes.txt");

  // função is_open testa se o arquivo está realmente aberto
  if(arq1.is_open()) {
    arq1 << "Fulano de Tal" << endl;
    arq1 << "João da Silva" << endl;
    arq1.close();
  }

  return (0);
}
```

Usando método open

- **arq1.open (**"nomes.txt", **ios::out)**
  - ios::out é o modo padrão para ofstream
- **arq1.open (**"nomes.txt", **ios::out | ios::app)**
  - Abre no final do arquivo “nomes.txt”. Ou seja, na posição depois do último caractere inserido
- **arq1.open (**"nomes.txt", **ios::app);**
  - Mesmo que o anterior, uma vez que ios::out é o padrão para ofstream

Escrita no final do arquivo

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

int main (){
  ofstream arq1;
  // Abre o arquivo no final (depois do último caractere) com ios::app
  arq1.open("nomes.txt", ios::app);

  // função is_open testa se o arquivo está realmente aberto
  if(arq1.is_open()) {
    // insere nomes no final do arquivo
    arq1 << "Fulano de Tal" << endl;
    arq1 << "João da Silva" << endl;
    arq1.close();
  } else {
    cerr << "ERRO: arquivo não pode ser aberto" << endl;
  }

  return (0);
}
```

Arquivos binários em C++

- C++ não impõe estrutura para arquivos.
- Os registros são **gravados** no arquivo através do método **`write`**:

```
write(const char *<ptr>, <size_t>);
```

- O parâmetro <*ptr*> é um ponteiro constante para o tipo char;
- O parâmetro <*size_t*> é um inteiro especificando o número de bytes a serem escritos. Normalmente utiliza-se da função *sizeof* para determinar o tamanho de um registro.

Arquivos binários em C++

- Os registros são **lidos** do arquivo através do método **`read`**:

```
read(char &<ptr>, <size_t>);
```

- O parâmetro <*ptr*> é um ponteiro para o tipo char. Assim, é necessário converter a variável para char * (cast);
- O parâmetro <*size_t*> é um inteiro especificando o número de bytes a serem lidos na memória. Normalmente utiliza-se da função *sizeof* para determinar o tamanho de um registro.

Escrita binária

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

struct Vec3D {
  float x, y, z;
};

int main (){
  Vec3D p1;
  ofstream arq1("nuvem.dat");

  if(arq1.is_open()) {
    for(int i=0; i<100; i++)
      arq1.write((const char*)&p1, sizeof(Vec3D));
    arq1.close();
  } else {
    cerr << "ERRO: arquivo não pode ser aberto" << endl;
  }

  return (0);
}
```

Leitura binária

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

struct Vec3D {
  float x, y, z;
};

int main (){
  Vec3D p1;
  ifstream arq1("nuvem.dat");

  if(arq1.is_open()) {
    while(arq1 && !arq1.eof()) {
      arq1.read((char*)&p1, sizeof(Vec3D));
      cout << "(" << p1.x << "," << p1.y << "," << p1.z << ")" << endl;
    }
    arq1.close();
  }
  return (0);
}
```

Escrita aleatória

```
#include <fstream>
#include <cstdlib>
using namespace std;

struct Vec3D { float x, y, z; };

int main (){
  Vec3D p1;
  ofstream arq1("nuvem.dat", ios::ate);

  if(arq1.is_open()) {
    for(int i=0; i<10; i+=(1+rand()%5)) {
      p1.x = rand()%10;
      p1.y = rand()%10;
      p1.z = rand()%10;
      arq1.seekp(i * sizeof(Vec3D));
      arq1.write((const char*)&p1, sizeof(Vec3D));
    }
    arq1.close();
  }
  return (0);
}
```

Estruturas de dados II

Revisão de arquivos em C++

André Tavares da Silva

andre.silva@udesc.br