Programação Orientada a Objetos

Prof. Delano Beder

Roteiro 18 – Leitura e Escrita de Arquivos (Binario)

1. Crie um projeto no Netbeans (Aplicação C/C++) denominado ArquivoBinario

1.1 Escolha: Criar arquivo principal main (C++)

2. Implementação da abstração Aluno

Crie uma Nova classe C++ denominada Aluno

Verifique se dois arquivos foram gerados: Aluno.h (Cabeçalho) e Aluno.cpp (Código-fonte)

2.1 Atualize o arquivo Aluno.h

```
#ifndef ALUNO_H
#define ALUNO_H

#include <iostream>
using namespace std;

class Aluno {
public:
  Aluno();
  Aluno(int RA, double NP, double NT);
  virtual ~Aluno();
  void imprime();
  int getRA() const;
  double media();
private:
  int RA;
  double NP;
  double NT;
};

#endif /* ALUNO_H */
```

2.2 Atualize o arquivo denominado Aluno.cpp

```
#include "Aluno.h"

Aluno::Aluno() {
}

Aluno::Aluno(int RA, double NP, double NT) :
RA(RA), NP(NP), NT(NT) {
}

Aluno::~Aluno() {
}

int Aluno::getRA() const {
  return RA;
}

double Aluno::media() {
  return NP * 0.8 + NT * 0.2;
}
void Aluno::imprime() {
  cout << "RA: " << RA;
  cout << ", NP: " << NP;
  cout << ", NT: " << NT;
  cout << ", Média: " << media() << endl;
}
```

3. Implementação da abstração Leitura/Escrita arquivos binários

Crie uma Nova classe C++ denominada BinaryFile

Verifique se dois arquivos foram gerados: BinaryFile.h (Cabeçalho) e BinaryFile.cpp (Código-fonte)

3.1 Atualize o arquivo BinaryFile.h

```
#ifndef BINARYFILE_H
#define BINARYFILE_H

#include <vector>
#include <string>
#include <fstream>
#include "Aluno.h"
using namespace std;

class BinaryFile {
public:
  BinaryFile(string nomeArquivo);
  virtual ~BinaryFile();
  void grava(int nro);
  void imprime();
  void imprimeRA(int RA);
  void imprimePos(int pos);
  static char opcao();
private:
  string nomeArquivo;
};

#endif /* BINARYFILE_H */
```

3.2 Atualize o arquivo denominado BinaryFile.cpp

```
#include "BinaryFile.h"

BinaryFile::BinaryFile(string nomeArquivo) :
nomeArquivo(nomeArquivo) {
}

BinaryFile::~BinaryFile() {
}

void BinaryFile::grava(int nro) {
  int RA;
  double NP, NT;
  Aluno* a;

  ofstream ofs(nomeArquivo.c_str(), ios::binary);

  for (int i = 0; i < nro; i++) {
    cout << "Digite RA, NP e NT: ";
    cin >> RA >> NP >> NT;
    a = new Aluno(RA, NP, NT);
    ofs.write(reinterpret_cast<char *> (a), sizeof (Aluno));
    cout << endl;
  }

  ofs.close();
}

void BinaryFile::imprime() {

  Aluno a;

  ifstream ifs(nomeArquivo.c_str(), ios::binary);

  if (ifs.is_open()) {
    ifs.read(reinterpret_cast<char *> (&a), sizeof (Aluno));

    while (ifs.good()) {
      a.imprime();
      ifs.read(reinterpret_cast<char *> (&a), sizeof (Aluno));
    }
  }

  ifs.close();
}




void BinaryFile::imprimeRA(int RA) {
```

```
  Aluno aluno;
  bool achou = false;
  int pos = 0;

  ifstream ifs(nomeArquivo.c_str(), ios::binary);

  if (ifs.is_open()) {
    while (!ifs.eof() && !achou) {
      ifs.read(reinterpret_cast<char *> (&aluno), sizeof (Aluno));
      achou = (aluno.getRA() == RA);
    }
  }

  ifs.close();

  if (achou) {
    aluno.imprime();
  } else {
    cout << "Aluno não encontrado" << endl;
  }
}

void BinaryFile::imprimePos(int pos) {
  Aluno aluno;

  ifstream ifs(nomeArquivo.c_str(), ios::binary);

  if (ifs.is_open()) {
    ifs.seekg((pos - 1) * sizeof (Aluno));
    ifs.read(reinterpret_cast<char *> (&aluno), sizeof (Aluno));
    if (ifs.good()) {
      aluno.imprime();
    } else {
      cout << "Aluno não encontrado" << endl;
    }
  }

  ifs.close();
}

char BinaryFile::opcao() {
  char c;
  cout << "[G] Grava arquivo" << endl;
  cout << "[I] Imprime todos" << endl;
  cout << "[R] Imprime por RA" << endl;
  cout << "[P] Imprime por posiÇão" << endl;
  cout << "[F] Fim" << endl;
  cout << "> ";
  cin >> c;
  return (toupper(c));
}
```

4. Atualize o arquivo main.cpp

```
#include "BinaryFile.h"
#include <iostream>
using namespace std;

int main(int argc, char** argv) {

  char op;
  int nroAlunos, pos;
  int RA;

  BinaryFile bf("info.dat");

  do {
    op = BinaryFile::opcao();
    switch (op) {
      case 'G':
        cout << "Numero de Alunos: ";
        cin >> nroAlunos;
        bf.grava(nroAlunos);
        break;
      case 'I':
        bf.imprime();
        break;
      case 'R':
        cout << "RA: ";
        cin >> RA;
        bf.imprimeRA(RA);
        break;
      case 'P':
        cout << "PosiÇão: ";
        cin >> pos;
        bf.imprimePos(pos);
        break;
    }
    cout << endl;
  } while (op != 'F');

  return 0;
}
```

5. Compile e exercite as diferentes opções (verifique a saída impressa)

6. Fim

Exercicios:

- Implementar na classe BinaryFile o método void imprimeOrdenado() que imprime os alunos em ordem crescente de média
- Implementar na classe BinaryFile o método void imprimeAprovados() que imprime os alunos aprovados (média >= 6.0)
- Implementar na classe BinaryFile o método void imprimeSAC() que imprime os alunos em exame SAC (5.0 <= média < 6.0)
- Implementar na classe BinaryFile o método void imprimeSAC() que imprime os alunos reprovados (média < 5.0)