




## 2022\_1 - PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II - TA\_TN - METATURMA

PAINEL > MINHAS TURMAS > 2022\_1 - PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II - TA\_TN - METATURMA > GERAL  
> L02E04 - CONVERSOR DE ARQUIVOS (4,0 PTS)

[Descrição](#)[Enviar](#)[</> Editar](#)[Visualizar envios](#)

### L02E04 - Conversor de Arquivos (4,0 pts)

 **Data de entrega:** sexta, 1 Jul 2022, 23:59

 **Arquivos requeridos:** csv.hpp, csv.cpp, file.hpp, ireadable.hpp, ireadable.cpp, pessoa.hpp, pessoa.cpp ( [Baixar](#))

 **Número máximo de arquivos:** 8

**Tipo de trabalho:**  Trabalho individual

Polimorfismo Avançado	
VPL: 4	Nome: Conversor de Arquivo

#### Objetivo:

Seu objetivo neste exercício é usar os conceitos de *polimorfismo* para criar uma interface de conversor de arquivos. Você irá implementar apenas o conversor de CSV, mas note que a estrutura sugerida neste exercício permite criar outros tipos de conversores também, isto é, nós poderíamos ter, por exemplo, uma outra classe chamada **Empresa** que herda de **IReadable** e com isso poderíamos criar objetos do tipo **Empresa** a partir de uma entrada no formato de um CSV ou gerar um saída no formato de CSV a partir de objetos do tipo **Empresa**.

Você pode baixar a *main.cpp* [aqui](#) se desejar reproduzir o exercício em sua máquina.

#### TADs do seu programa:

Classe IReadable	
Atributos:	Nenhum novo atributo
Métodos:	<pre>protected: virtual void print(std::ostream&amp; out) → Este método tem que ser <i>pure virtual</i>.  public: virtual void GetCampos(std::vector&lt;std::string&gt;&amp; out) → Este método tem que ser <i>pure virtual</i>. public: virtual void setAtributo(std::string key, std::string valor) → Este método tem que ser <i>pure virtual</i>. public: virtual std::string GetAtributo(std::string key) → Este método tem que ser <i>pure virtual</i>.  friend std::ostream&amp; operator&lt;&lt; (std::ostream&amp; out, IReadable&amp; readable) → Este método implementa a sobrecarga do operador "&lt;&lt;" e nos permite imprimir um objeto <i>IReadable</i>. Aqui você deve chamar o método <i>print</i> do argumento do tipo <i>IReadable</i>, pois as classes derivadas de <i>IReadable</i> já personalizam como serão impressas. Note que este é um método <i>friend</i>, isso significa que não é um método da classe e sim um método que vamos implementar fora dela que tem acesso a conteúdos privados e protegidos da classe <i>IReadable</i>. É diferente do que vimos em aula, onde não foi preciso colocar a assinatura dentro da outra classe que iríamos utilizar (porque tudo era público).</pre>

Classe Pessoa : IReadable	
Atributos:	<pre>private std::string nome → Nome da pessoa private int idade → Idade da pessoa private unsigned long cpf → CPF da pessoa</pre>
Métodos:	

**protected: virtual void print(std::ostream& out) override** → Este método deve enviar todos os dados das pessoas que serão impressas para o stream *out*. O formato da impressão deve ser:

```
(nome = x, idade = y, CPF = z)
```

Onde "x" corresponde ao nome, "y" à idade e "z" ao CPF da pessoa. Atenção, **não** adicione quebra de linha ao final.

**public: virtual void GetCampos(std::vector<std::string>& out) override** → Deve preencher o array *out* com o nome de todos os atributos da classe *Pessoa*.

**public: virtual void setAtributo(std::string key, std::string valor) override** → Deve atualizar um certo atributo da classe conforme o valor informado. Por exemplo, se *key* for "nome" e *valor* for "maria", então devemos alterar o nome da pessoa para maria. Dica: veja os últimos dois links das referências ao final.

**public: virtual std::string GetAtributo(std::string key) override** → Retorna o valor de um atributo da pessoa conforme a *key* informada (que indica o nome do atributo)

**bool operator==(Pessoa& rhs)** → Sobrecarga de operador que nos permite verificar se a pessoa *rhs* é igual a pessoa que representa a instância. Duas pessoas são iguais se possuem o mesmo CPF

#### Classe File

**Atributos:**

**Métodos:**

**public: virtual void readLine(std::string& head, std::string& line, IReadable& object)** → Esse método tem que ser *pure virtual*

**public: virtual std::string write(IReadable& object)** → Esse método tem que ser *pure virtual*

#### Classe CSV : File

**Atributos:**

*Nenhum novo atributo*

**Métodos:**

**private: void split(std::string& str, std::vector<std::string>& out)** → Este método deve preencher o vetor *out* com todas as palavras existentes na string *str* que são separadas por ";", ou seja, quebra a string de acordo com o delimitador.

**public: std::string getHeader(IReadable& object)** → Deve retornar uma string contendo o cabeçalho do arquivo CSV, que correspondem aos campos do *object* passado por parâmetro. Cada atributo do cabeçalho deve ser separado por ponto e vírgula (não deve-se ter ponto e vírgula ao final). Dica: o método *GetCampos* pode ser útil.

**public: virtual void readLine(std::string& head, std::string& line, IReadable& object) override** → O *head* e o *line* são strings no formato CSV (ou seja, separadas por ponto e vírgula). Você deve usar essas strings para setar os atributos do *object* recebido como parâmetro. Dica: o método *split* da própria classe pode ser útil.

**public: virtual std::string write(IReadable& object) override** → Você deve converter o *object* recebido como parâmetro em uma string. A string deve corresponder os valores do objeto correspondente a cada campo no cabeçalho separados por ponto e vírgula. Dica: os métodos *GetCampos* e *GetAtributo* podem ser úteis.

#### Main

As especificações estão no próprio arquivo.

Você tem liberdade para implementar quaisquer outros métodos na TAD que julgar necessário.

#### Exemplos de entrada e saída:

##### Exemplo 1

##### Entrada:

```
nome;idade;cpf
Mario;28;12646352447
Thanos;49;76446351001
Luigi;25;27471240602
Peter;45;16663852135
Drake;92;76446351001
Portioli;25;57824716743
```

##### Saída:

```
Linha 1 -- Pessoa: (nome = Mario, idade = 28, CPF = 12646352447)
Linha 2 -- Pessoa: (nome = Thanos, idade = 49, CPF = 76446351001)
Linha 3 -- Pessoa: (nome = Luigi, idade = 25, CPF = 27471240602)
Linha 4 -- Pessoa: (nome = Peter, idade = 45, CPF = 16663852135)
Linha 5 -- Pessoa: (nome = Drake, idade = 92, CPF = 76446351001)
Linha 6 -- Pessoa: (nome = Portioli, idade = 25, CPF = 57824716743)
```

```
-----
CSV de CPFs repetidos:
nome;idade;cpf
Thanos;49;76446351001
Drake;92;76446351001
-----
```

```
CSV de CPFs únicos:
nome;idade;cpf
Mario;28;12646352447
Luigi;25;27471240602
Peter;45;16663852135
Portioli;25;57824716743
```

Exemplo 2	
<b>Entrada:</b> nome;idade;cpf Norris;28;12646352447 Tribunal;49;76446351001	<b>Saída:</b> Linha 1 -- Pessoa: (nome = Norris, idade = 28, CPF = 12646352447) Linha 2 -- Pessoa: (nome = Tribunal, idade = 49, CPF = 76446351001) ----- CSV de CPFs repetidos: nome;idade;cpf ----- CSV de CPFs únicos: nome;idade;cpf Norris;28;12646352447 Tribunal;49;76446351001

**Links Úteis:**

Um pouco sobre a [função friend](#)  
Mais informações sobre o [set da biblioteca <set>](#)  
Sobrecarga do [operador <<](#)  
<https://www.cplusplus.com/reference/string/stoi/>  
<https://www.cplusplus.com/reference/string/stoul/>

[VPL](#)[◀ L02E03 - Campo Minado \(3,0 pts\)](#)

Seguir para...

[L02E05 - Revisão de código e Refatoração \(4,0 pts\) ▶](#)