Este exercício possui dois objetivos:

1. Familiarizar o estudante com o padrão de projetos "[template method](https://pt.wikipedia.org/wiki/Template_Method)"
2. Treinar o estudante em técnicas para manipular strings.

A fim de alcançar tais objetivos, o aluno deverá implementar diferentes classes concretas para o algoritmo abaixo:

class LineProcessor {

private:

std::istream &\_in;

unsigned \_numLinhasValidas;

public:

LineProcessor(): \_in(std::cin), \_numLinhasValidas(0) {}

virtual ~LineProcessor() {}

unsigned numeroDeLinhasValidas() const;

unsigned numeroDeLinhasLidas() const;

virtual bool linhaValida(const std::string &str) const = 0;

virtual void processaLinha(const std::string &str) = 0;

void processaEntrada() {

std::string linha;

while (std::getline(\_in, linha)) {

if (linhaValida(linha)) {

\_numLinhasValidas++;

processaLinha(linha);

}

}

}

};

Este algoritmo lê a entrada padrão, uma linha de cada vez, e processa as linhas consideradas válidas. Porém, tanto a definição de linha válida e a definição do processamento são deixadas em aberto. Note que os métodos linhaValida e processaLinha são abstratos. Diferentes classes podem implementar esses métodos. Para cada implementação podemos ter comportamentos muito diferentes. Por exemplo, abaixo vê-se uma classe que considera qualquer linha válida, e simplesmente imprime as linhas lidas na saída padrão:

class LinePrinter: public LineProcessor {

public:

bool linhaValida(const std::string &str) const override {

return true;

}

void processaLinha(const std::string &str) override {

std::cout << str << std::endl;

}

};

Para resolver esse exercício, você deverá realizar as seguintes ações:

1. Modifique a classe LineProcessor, implementando o método numeroDeLinhasLidas. Essa modificação vai permitir que objetos desse tipo possam contar o número de linhas lidas. Note que a classe já conta o número de linhas válidas. Essa extensão será bem similar. Você deverá adicionar mais um atributo à classe, e incrementá-lo sempre que uma nova linha for vista (seja ela válida ou não). Lembre-se de inicializar este novo atributo no construtor da classe.
2. Implemente a classe ContadorNumNaturais, que imprime a soma dos números naturais encontrados em linhas que contém somente números.
3. Implemente a classe LeitorDeFutebol, que imprime o resultado de jogos cujo placar aparece em linhas válidas.
4. Implemente a classe ContadorDePalavras, que conta a quantidade de palavras em cada linha. Note que uma palavra é um "*token*", isto é, uma string que não possui espaços brancos.
5. Implemente a classe InversorDeFrases, que inverte a ordem em que palavras aparecem nas linhas válidas. Linhas válidas contém somente tokens com letras (e possivelmente espaços em branco). Tokens que contém qualquer tipo de caractere especial (incluindo números) invalidam a linha. Quando a linha for impressa, os tokens devem ser separados por espaços simples.
6. Implemente a classe EscritorDeDatas, que substitui números que denotam meses pelas três iniciais daquele mês. Linhas válidas contém datas, no formato dia/mês/ano, sendo dia e mês inteiros com um ou dois dígitos, e ano um inteiro com quatro dígitos. A saída deve ser somente as iniciais do mês, e.x.: "Jan", "Fev", etc.

O arquivo LineProcessor.hpp contém vários comentários mostrando exemplos de padrões válidos e inválidos, e o que deve ser impresso. Adicionalmente, [este arquivo](https://virtual.ufmg.br/20212/pluginfile.php/223526/mod_vpl/intro/ExemplosEntradas.pdf) contém exemplos de entradas válidas e inválidas e exemplos de saídas esperadas para diferentes testes. Além disso, duas classes já estão implementadas como exemplos, a saber, ContadorPopRural e LinePrinter. Note que a classe ContadorPopRural usa expressões regulares para determinar as linhas válidas. Você não precisa usar expressões regulares neste exercício. Porém, há duas vantagens em usá-las:

1. Elas tornam algumas questões deste exercício bem mais fáceis de serem resolvidas.
2. Expressões regulares são muito úteis em geral, para vários tipos de processamento de texto. Vale a pena aprender!

Alguns recursos para aprender expressões regulares seguem abaixo:

* [Site](https://docs.microsoft.com/pt-br/cpp/dotnet/regular-expressions-cpp-cli?view=msvc-160) da Microsoft sobre o assunto.
* Tutorial do [imaster](https://imasters.com.br/back-end/trabalhando-com-expressoes-regulares-em-c11).

Caso você queira testar seu VPL localmente, você pode baixar o arquivo main.cpp, e usá-lo com diferentes arquivos de texto, assim:

g++ \*.cpp -o main.exe

./main.exe < t2.txt

Seguem, abaixo, alguns exemplos de teste. Uma [versão mais detalhada](https://virtual.ufmg.br/20212/pluginfile.php/223526/mod_vpl/intro/ExemplosEntradas.pdf) dessas entradas está disponível para consulta. Note que o primeiro número em cada teste serve para o programa de testes saber qual função executar. Aquele número não é parte da entrada que deve ser processada:

