|  |
| --- |
| Project 2  Bibliotecas EASY\_STD em C++  Feito por: João Roque e Licínio Feliciano |



Conteúdo

[Introdução 2](#__RefHeading___Toc505_2479917017)

[Implementação e Instruções de Uso 3](#__RefHeading___Toc286_93532997)

[Biblioteca EASY\_UTILS 4](#__RefHeading___Toc412_1195533232)

[Biblioteca EASY\_TEXT 6](#__RefHeading___Toc282_93532997)

[Biblioteca EASY\_RANDOM 8](#__RefHeading___Toc284_93532997)

[Gestão de viaturas 9](#__RefHeading___Toc524_1642213040)

[Conclusão 11](#__RefHeading___Toc511_2479917017)

# **Introdução**

Este projecto tem como objectivo desenvolver um conjunto de bibliotecas pessoais de utilitários em C/C++, com o foco de consolidar os conhecimentos de programação nesta linguagem. As bibliotecas, visam a simplificar o desenvolvimento em C++, oferecendo mecanismos que, embora não estejam presentes na biblioteca padrão do C++, são necessários para diversas aplicações. As bibliotecas abordarão três temas principais: Utilitários Gerais, Texto e Aleatoriedade. Todas as definições destas bibliotecas serão integradas no namespace easy\_std.

As bibliotecas Easy\_STD são projectadas para serem úteis, simples, fáceis de implementar e legíveis, priorizando a utilidade e a simplicidade sobre o desempenho. Isto é especialmente relevante em cenários de programação de sistemas, onde o bom desempenho e a gestão dos recursos do computador são requisitos importantes. No entanto, nas bibliotecas Easy\_STD, o bom desempenho é considerado um requisito secundário.

# Implementação e Instruções de Uso

**Compilacao:**

Para compilar o programa, use um compilador C++ com suporte para o padrão C++20.

No Linux/Unix: g++ -std=c++20 teste\_easy\_\*.cpp -o prog

No Windows: g++ -std=c++20 teste\_easy\_\*.cpp -o prog.exe

- *Substitua \* pela referência do programa (utils, text, random).*

**Execução:**

Execute o ficheiro compilado: ./prog (Linux/MacOS) ou prog.exe (Windows).

Siga as instruções no ecrã para testar as funções.

**Utilização:**

O programa irá correr vários *checks*, uns interactivos e outros automáticos, de modo a demonstar o funcionamento de cada função.

**Encerrar o programa:**

Para encerrar o programa, basta fazer a combinação de teclas **CTRL+D, CTRL+C**, ou correr todos os checks até ao fim.

# Biblioteca EASY\_UTILS

**Input -** Esta função exibe uma mensagem solicitando a introdução de uma linha de texto através da entrada padrão e, em seguida, retorna a string introduzida.

**to\_string -** Esta função converte um objetco do tipo parametrizado T em uma string, utilizando as facilidades definidas em sstream.

**to\_string (especialização para std::vector) -** Esta função especializada para std::vector converte um vector numa string no formato "{ elem1, elem2, ...}".

**to\_string (especialização para std::list) -** Esta função especializada para std::list converte uma lista numa string no mesmo formato que a função para vectores.

**to\_string (especialização para std::map) -** Esta função especializada para std::map converte um mapa numa string no formato "{chave1 => valor1, chave2 => valor2, ...}".

**to\_string (especialização para std::unordered\_map) -** Esta função especializada para std::unordered\_map converte um mapa não ordenado numa string no mesmo formato que a função para mapas.

**convert -** Esta função converte uma string para um objeto do tipo parametrizado T. Se a conversão falhar, lança uma excepção do tipo std::invalid\_argument.

**print -** Esta função escreve os elementos de uma colecção numa saída especificada, separando os elementos com um delimitador e termina com outro delimitador.

**print (overload para escrever na std::cout) -** Esta função overloaded escreve os elementos de uma coleção na saída padrão, utilizando os delimitadores especificados.

**find\_index -** Dada uma sequência do tipo S (que deve ser um container) contendo itens do tipo T, esta função procura por um item e devolve a sua posição, caso exista, ou -1 caso contrário. A pesquisa começa a partir da posição especificada.

**in (verificar a presença de um item em uma coleção) -** Esta função verifica se um item está presente numa coleção, retorna verdadeiro se estiver e falso caso contrário.

**in (verificar a presença de um item em uma lista de inicialização) -** Esta função verifica se um item está presente numa lista de inicialização, retorna verdadeiro se estiver e falso caso contrário.

**in (verificar a presença de uma chave em um mapa) -** Esta função verifica se uma chave está presente num mapa, retorna verdadeiro se estiver e falso caso contrário.

**in (verificar a presença de uma chave em um mapa não ordenado) -** Esta função verifica se uma chave está presente num mapa não ordenado, retorna verdadeiro se estiver e falso caso contrário.

# Biblioteca EASY\_TEXT

**is\_white\_space(ch) -** Devolve **true** se o caractere for um caractere de espaçamento. (ex: ' ', '\n', '\t' ou '\r').

**is\_white\_space(str) -** Devolve **true** se todos os caractere da string forem de espaçamento. Deve utilizar **is\_white\_space(ch)**.

**is\_digit(ch) -** Devolve **true** se o caractere for um dígito decimal.

**is\_digit(str) -** Devolve **true** se todos os caractere da *string* forem dígitos. Deve utilizar **is\_digit(ch)**.

**to\_lower(ch) -** Converte um caractere (letra) maiúscula para minúscula.

**to\_lower(str) -** Converte um caractere (letra) maiúscula para minúscula. Esta

função modifica a *string* e devolve-a para que possa ser utilizada em expressões

do tipo *string*.

**trim\_left(str) -** Remove os caracteres de espaçamento à esquerda da *string*. A função modifica a *string* e devolve-a para que possa ser utilizada.

**trim\_right(str) -** Remove os caracteres de espaçamento à direita da *string*. A função modifica a *string* e devolve-a para que possa ser utilizada.

**trim(str) -** Remove os caracteres de espaçamento à esquerda e direita da *string*. A função modifica a *string* e devolve-a para que possa ser utilizada.

**reverse(str)-** Inverte a *string*. A função modifica a *string* e devolve-a para que possa ser utilizada. (ex: antonio jose => esoj oinotna)

**Reversed(str) -** Devolve uma cópia invertida da *string* com os caracteres por ordem inversa. (ex: antonio jose => oinotna esoj )

**replace (STRING, CH1, CH2, [START], [END]) -** Substitui todas as ocorrências do caractere **CH1** pelo caractere **CH2**. A função modifica a *string* e devolve-a para que possa ser utilizada.

Opções:

**STRING** - A *string* que vai sofrer alterações.

**CH1 –** Caractere a ser modificado.

**CH2** – Caractere que vai substituir **CH1**.

**START (int)** – A posiçao inicial para substituições [default: 0].

**END (int) –** A posição final para substituições [default: -1, ou seja, último caractere].

**replace (STRING, SUBSTR1, SUBSTR2, [START], [END]) –** Sunstitui todas as ocorrências da **substring1** pela **substring2**. A função modifica a *string* e devolve-a para que possa ser utilizada.

Opções:

**STRING -** A *string* que vai sofrer alterações.

**SUBSTR1 -** *substring* a ser modificada.

**SUBSTR2 -** A *substring* que vai substituir **SUBSTR1.**

**START (int)-** A posiçao inicial para substituições [default: 0].

**END (int)-** A posição final para substituições [default: -1, ou seja, última substring].

**split (STRING, DELIM) -** Separa a *string* em partes, usando como delimitador a variavel DELIM.

Opções:

**STRING –** *string* a separar. (ex: "ABC.DEF.GHI")

**DELIM –** delimitador a usar para a separação. (ex: split("ABC.DEF.GHI", ".") => {"ABC", "DEF", "GHI"})

**join {PARTS, DELIM} –** Junta todas as *strings*, usando como delimitador a variavel **DELIM.**

Opções:

**PARTS -** *vector* de *strings* a juntar. (ex: {"ABC”, “DEF”, “GHI"})

**DELIM –** delimitador a usar para junção. (ex: join{"ABC, “DEF”, “GHI", "."} => "ABC.DEF.GHI")

# **Biblioteca EASY\_RANDOM**

**randint (int a, int b) -** Devolve um *int* entre *int* **a** e *int* **b** ínclusive.

**random(double a, double b) -** Devolve um *double* entre *double* **a** e *double* **b** ínclusive. Por omissão, a = 0.0 e b = 1.0.

**seed(int value) -** Fornece um novo valor semente para o gerador de número aleatório rand\_gen.

**choice(const Seq& seq) -** Escolhe aleatoriamente um elemento da sequência *seq*.

**shuffle\_(Seq& seq) -** Mistura aleatóriamente os elementos da sequência *seq*.

**sample(const Seq& seq, int n) -** Selecciona uma amostra de *n* valores da sequência *seq*. Os resultados são devolvidos num vector do tipo da variável de retorno ou do tipo de chamamento.

# Gestão de viaturas

Este programa em C++ permite gerir um catálogo/stand de viaturas. O programa oferece funcionalidades como listar o catálogo, pesquisar por determinados campos, eliminar um registo do catálogo e guardar o catálogo num ficheiro.

**Utilização do Programa**

* **Classe Viatura**

A classe Viatura representa uma viatura com os seguintes atributos:

**- Matrícula**

**- Marca**

**- Modelo**

**- Data**

* **Constructores**

**Viatura(const string& matricula, const string& marca, const string& modelo, const string& data) -** Cria um objeto Viatura com os atributos fornecidos.

**Viatura(const string& matricula, const string& marca, const string& modelo) -** Construtor alternativo que utiliza uma data predefinida.

* **Métodos Estáticos**

**from\_csv(const string& viat\_csv) -** Converte uma string CSV numa instância de Viatura.

**to\_csv() -** Converte a Viatura numa string CSV.

**Mostra() -** Mostra os detalhes da viatura.

* **Validação de Dados**

A classe inclui métodos para validar a matrícula, data, marca e modelo.

Função Principal (main())

A função principal demonstra a criação de objectos Viatura, invocação de métodos e escrita em ficheiro.

**Como Utilizar**

Para compilar o programa, use um compilador C++ com suporte para o padrão C++20.

**No Linux/Unix -** g++ -std=c++20 Gestao\_viaturas.cpp -o gestao\_viaturas

**No Windows -** g++ -std=c++20 Gestao\_viaturas.cpp -o gestao\_viaturas.exe

Execute o programa.

O programa irá demonstrar a criação de viaturas com dados de exemplo, mostrando os detalhes de cada uma e guardando essas viaturas num ficheiro CSV.

No fim encaminhará o utilizador para a utilizacão do programa.

# Conclusão

Ao concluir este projecto, é possível afirmar que a programação em C/C++ é uma ferramenta poderosa e versátil. A criação das bibliotecas EasySTD, focadas em simplificar e facilitar o desenvolvimento em C++, demonstra a importância de ter ferramentas que integrem e simplifiquem tarefas comuns, permitindo que os desenvolvedores se concentrem na lógica do programa e na solução de problemas específicos.

O desenvolvimento de um programa para a gestão de um catálogo de viaturas de um stand automóvel, além de ser um exemplo prático da aplicação da biblioteca EasySTD, também ilustra a capacidade da linguagem C++ de ser utilizada em contextos diversos, desde aplicações simples até sistemas mais complexos. Este projecto, portanto, não apenas consolida os conhecimentos de programação em C/C++, mas também destaca a importância de criar bibliotecas personalizadas que atendam às necessidades específicas dos projectos, contribuindo para a eficiência e produtividade do desenvolvedor.