E-fólio Global | Folha de resolução do E-fólio

AbERTA

UNIDADE CURRICULAR: Programação por Objetos

CÓDIGO: 21093

DOCENTES: Jorge Morais e Leonel Morgado

TUTOR: Rúdi Oliveira

A preencher pelo estudante

NOME: Ivo Vieira Baptista

N.º DE ESTUDANTE: 2100927

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 06 de fevereiro de 2023

TRABALHO / RESOLUÇÃO:

Estou a utilizar o c++17 e 20, no sistema operativo IOs da apple, e o visual studio code como IDE, o programa tem os ficheiros .h e .cpp com as classes bem definidas, por falta de tempo falta comentários e validações, o programa tem uma pasta principal e uma subpasta **src** onde esta os ficheiros .h e .cpp, no terminal estando na pasta principal, temos um ficheiro **txt** com as configurações para compilar com **cmake** da seguinte forma:

Para gerar utilizamos:

cmake -S . -B out/

depois para compilar utilizámos:

cmake --build out/

este vai criar o ficheiro executável dentro da pasta **out** e para executar basta escrever na pasta principal:

out/pfoliog

```
→  PfolioGlobal out/pfoliog
Produtos
========
Bola de Futebol; Bola de Futebol Oficial da FIFA; 25.99; 0
Pneu; Pneu para carros; 50.5; 0
Smartphone; Smartphone XPTO OPTX; 450; 0
Televisao; Televisao LED 4K; 500; 0
Guitarra; Guitarra Eletrica; 250; 0
Bicicleta; Bicicleta de montanha; 300; 0
Camara; Maquina Fotografica Digital; 600; 0
Computador Portatil; Notebook MinhaMarca; 950; 0
Fogao; Fogao a gas de 4 bocas; 250; 0
Microondas; Microondas de encastrar; 200; 0
Ar Condicionado; Unidade Split interior; 850; 0
Maquina de Lavar; Maquina de Lavar Roupa; 450; 0
Aspirador; Aspirador de Po sem fio; 450; 0
Ventilador; Ventilador de Teto; 75; 0
Purificador; Purificador de Agua; 150; 0
Colunas; Colunas de som ambiente; 200; 0
Churrasqueira; Churrasqueira a Gas; 350; 0
Maquina de barbear; Maquina de barbear CortaBem; 100; 0
Relogio; Relogio PonteirosSoAqui; 450; 0
Secador; Secador de Cabelo; 150; 0
Fritadeira; Fritadeira Eletrica; 100; 0
Cafeteira; Cafeteira vermelha; 200; 0
Ferro; Ferro de Passar; 75; 0
Maquina de Cafe Expresso; Maquina de Cafe Expresso multicapsula; 250; 0
Livro; Livro de ficcao cientifica; 20; 0
Isqueiro; Isqueiro descartavel; 2.5; 0
Mochila; Mochila escolar; 25; 0
Caneta; Caneta azul; 0.5; 0
Bloco de Notas; Bloco de Notas A5; 2; 0
Calculadora; Calculadora cientifica; 15; 0
Lapiseira; Lapiseira com minas 0.7mm; 2; 0
Tesoura; Tesoura escolar para papel; 5; 0
Lapis de Cor; Conjunto de 24 Lapis de cor; 3.5; 0
Gorro; Gorro de lã; 10; 0
Luvas; Luvas de inverno; 15; 0
Meias; Par de meias de algodao; 5; 0
Toalha de Banho; Toalha de Banho extra grande; 20; 0
Almofada; Almofada decorativa; 15; 0
Edredao; Edredao para cama de casal; 50; 0
Cortina; Cortina de duche; 25; 0


Campanhas
========
Campanha de Ferias; 0.1; NÃO ATIVA
Campanha de Aniversario; 0.15; NÃO ATIVA
Campanha de Natal; 0.2; NÃO ATIVA
Campanha de Pascoa; 0.15; NÃO ATIVA
Campanha de Verao; 0.25; NÃO ATIVA
Campanha de Primavera; 0.2; NÃO ATIVA
Campanha de Outono; 0.1; NÃO ATIVA
Campanha de Inverno; 0.15; NÃO ATIVA
Black Friday; 0.3; NÃO ATIVA
Cyber Monday; 0.35; NÃO ATIVA
Campanha do Dia dos Namorados; 0.15; NÃO ATIVA
Campanha do Dia da Mae; 0.2; NÃO ATIVA
Desconto do Dia do Pai; 0.25; NÃO ATIVA
Saldos; 0.4; NÃO ATIVA
Liquidacao; 0.5; NÃO ATIVA
Campanha de Final de Ano; 0.3; NÃO ATIVA
Regresso 'as Aulas; 0.2; NÃO ATIVA
Campanha de Ano Novo; 0.25; NÃO ATIVA
Descontos da Nova Colecão; 0.15; NÃO ATIVA
Campanha de Halloween; 0.35; NÃO ATIVA
Desconto do Dia da Crianca; 0.25; NÃO ATIVA
```

```
Loja
====
Produtos em Stock da Loja
=========================
Bola de Futebol; Bola de Futebol Oficial da FIFA; 25.99; 0
Pneu; Pneu para carros; 50.5; 0
Smartphone; Smartphone XPTO OPTX; 450; 0
Televisao; Televisao LED 4K; 500; 0
Guitarra; Guitarra Eletrica; 250; 0
Bicicleta; Bicicleta de montanha; 300; 0
Camara; Maquina Fotografica Digital; 600; 0
Computador Portatil; Notebook MinhaMarca; 950; 0
Fogao; Fogao a gas de 4 bocas; 250; 0
Microondas; Microondas de encastrar; 200; 0
Ar Condicionado; Unidade Split interior; 850; 0
Maquina de Lavar; Maquina de Lavar Roupa; 450; 0
Aspirador; Aspirador de Po sem fio; 450; 0
Ventilador; Ventilador de Teto; 75; 0
Purificador; Purificador de Agua; 150; 0
Colunas; Colunas de som ambiente; 200; 0
Churrasqueira; Churrasqueira a Gas; 350; 0
Maquina de barbear; Maquina de barbear CortaBem; 100; 0
Relogio; Relogio PonteirosSoAqui; 450; 0
Secador; Secador de Cabelo; 150; 0
Fritadeira; Fritadeira Eletrica; 100; 0
Cafeteira; Cafeteira vermelha; 200; 0
Ferro; Ferro de Passar; 75; 0
Maquina de Cafe Expresso; Maquina de Cafe Expresso multicapsula; 250; 0
Livro; Livro de ficcao cientifica; 20; 0
Isqueiro; Isqueiro descartavel; 2.5; 0
Mochila; Mochila escolar; 25; 0
Caneta; Caneta azul; 0.5; 0
Bloco de Notas; Bloco de Notas A5; 2; 0
Calculadora; Calculadora cientifica; 15; 0
Lapiseira; Lapiseira com minas 0.7mm; 2; 0
Tesoura; Tesoura escolar para papel; 5; 0
Lapis de Cor; Conjunto de 24 Lapis de cor; 3.5; 0
Gorro; Gorro de lã; 10; 0
Luvas; Luvas de inverno; 15; 0
Meias; Par de meias de algodao; 5; 0
Toalha de Banho; Toalha de Banho extra grande; 20; 0
Almofada; Almofada decorativa; 15; 0
Edredao; Edredao para cama de casal; 50; 0
Cortina; Cortina de duche; 25; 0
=========================



Campanhas da Loja
=================
Campanha de Ferias; 0.1; NÃO ATIVA
Campanha de Aniversario; 0.15; NÃO ATIVA
Campanha de Natal; 0.2; NÃO ATIVA
Campanha de Pascoa; 0.15; NÃO ATIVA
Campanha de Verao; 0.25; NÃO ATIVA
Campanha de Primavera; 0.2; NÃO ATIVA
Campanha de Outono; 0.1; NÃO ATIVA
Campanha de Inverno; 0.15; NÃO ATIVA
Black Friday; 0.3; NÃO ATIVA
Cyber Monday; 0.35; NÃO ATIVA
Campanha do Dia dos Namorados; 0.15; NÃO ATIVA
Campanha do Dia da Mae; 0.2; NÃO ATIVA
Desconto do Dia do Pai; 0.25; NÃO ATIVA
Saldos; 0.4; NÃO ATIVA
Liquidacao; 0.5; NÃO ATIVA
Campanha de Final de Ano; 0.3; NÃO ATIVA
Regresso 'as Aulas; 0.2; NÃO ATIVA
Campanha de Ano Novo; 0.25; NÃO ATIVA
Descontos da Nova Colecão; 0.15; NÃO ATIVA
Campanha de Halloween; 0.35; NÃO ATIVA
Desconto do Dia da Crianca; 0.25; NÃO ATIVA
=========================
```

Ficheiro txt comas as configurações para compilar com Cmake:

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.20)

project(Pfolio_Global)

set(CMAKE_CXX_STANDARD 20)

set(CMAKE_CXX_STANDARD_REQUIRED ON)

file(GLOB_RECURSE SOURCE_FILES "src/*")

message("======================")
foreach(SOURCE ${SOURCE_FILES})
    message("Arquivo: ${SOURCE}")
endforeach()
message("======================")

add_executable(pfoliog "${SOURCE_FILES}")
```

Também podemos compilar dentro da pasta **src** com **gcc**:

comando no terminal dentro da pasta dos ficheiros .cpp e .hpp:

g++ *.cpp -o pfoliog -std=c++17

isto cria o ficheiro executável pfoliog que executamos da seguinte forma:

./pfoliog

Os ficheiros do programa:
produto.h
campanha.h
loja.h
e os seus respectivos .cpp
e o main.cpp

main.cpp

```cpp
/*
**============================================================================
** UC: 21093 - Programação por Objectos - 03
** e-fólio A  2022-23 (efolioGlobal.cpp)
**
** Aluno: 2100927 - Ivo Baptista
** Name        : EfolioG.cpp
** Author      : Ivo Baptista
** Version     : 1.1
** Copyright   : Your copyright notice
** Description : Atividades Imersivas in C++, Ansi-style
** ============================================================================
*/

#include <iostream>
#include <memory>
#include <fstream>
#include <string>
#include <list>
#include "produto.h"
#include "campanha.h"
#include "loja.h"

int main(int argc, char **argv) {
/*************************************************************
            abre ficheiro produtos.csv
*************************************************************/
    const std::string kProdutosFilename{"produtos.csv"};
    std::ifstream in_produtos;
    in_produtos.open(kProdutosFilename);
    std::string line;
    std::getline(in_produtos, line);
/*************************************************************
    leitura de objeto produto com ponteiro
    inteligentes aproveitando o polimorfimo
*************************************************************/
    std::list<std::shared_ptr<Produto>> list_de_produtos;
    while(!in_produtos.eof()) {
        auto produto = std::make_shared<ProdutoDeLoja>();
        produto->Ler(in_produtos);
        list_de_produtos.push_back(produto);
    }
/*************************************************************
        abre ficheiro descontos.csv
*************************************************************/
    const std::string kDescontosFilename{"descontos.csv"};
    std::ifstream in_descontos;
    in_descontos.open(kDescontosFilename);
    std::getline(in_descontos, line);
/*************************************************************
```

```
     leitura de objeto campanha com ponteiros
     inteligentes aproveitando o polimorfimo
*************************************************************/
    std::list<std::shared_ptr<Campanha>> list_de_campanhas;
    while(!in_descontos.eof()) {
        auto campanha = std::make_shared<CampanhaAtiva>();
        campanha->Ler(in_descontos);
        list_de_campanhas.push_back(campanha);
    }
/*************************************************************
         mostra na tela a lista de produtos
*************************************************************/
    std::cout << "Produtos" << std::endl;
    std::cout << "========" << std::endl;
    for(auto produto : list_de_produtos) {
        produto->Escrever();
        std::cout << std::endl;
    }
    std::cout << std::endl
              << std::endl
              << std::endl;
/*************************************************************
         mostra na tela a lista de capanhas
*************************************************************/
    std::cout << "Campanhas" << std::endl;
    std::cout << "=========" << std::endl;
    for(auto campanha : list_de_campanhas) {
        campanha->Escrever();
        std::cout << std::endl;
    }
    std::cout << std::endl
              << std::endl
              << std::endl;
 /*************************************************************
  Carrega os fichiero lidos com os operadores no objeto loja
*************************************************************/
    std::cout << "Loja" << std::endl;
    std::cout << "====" << std::endl;
    Loja loja;
    loja = loja + list_de_produtos;
    loja = loja + list_de_campanhas;
    loja.Escrever();


    return EXIT_SUCCESS;
}
```

Produto.h

```h
#pragma once
#include <string>
#include <fstream>

/*************************************************************
            definiçao de classe Produto
*************************************************************/
class Produto {
    public:
/*************************************************************
  declaramos metodo ler recebendo um stream como parametro
*************************************************************/
        void Ler(std::istream &is);
/*************************************************************
           declaramos metodo escrever sem parametro
*************************************************************/
        virtual void Escrever();

        const std::string GetNome();
        void SetNome(const std::string &nome);

        const std::string GetDescricao();
        void SetDescricao(const std::string &descricao);

        const float GetPreco();
        void SetPreco(float preco);

    private:
        std::string nome_;
        std::string descricao_;
        float preco_{0.0};
};
```

Produto.cpp

```cpp
#include <iostream>
#include <string>
#include <sstream>
#include "produto.h"

const std::string Produto::GetNome() {
    return nome_;
}

void Produto::SetNome(const std::string &nome) {
    nome_ = nome;
}
```

```cpp
const std::string Produto::GetDescricao() {
    return descricao_;
}

void Produto::SetDescricao(const std::string &descricao) {
    descricao_ = descricao;
}

const float Produto::GetPreco() {
    return preco_;
}

void Produto::SetPreco(float preco) {
    preco_ = preco;
}
/*************************************************************
       metodo ler recebendo um stream como parametro
*************************************************************/
void Produto::Ler(std::istream &in) {
    constexpr char kCommaDelimiter{','};
    std::string line;

    std::getline(in, line);

    std::stringstream ss{line};
    std::string token;

    if(std::getline(ss, token, kCommaDelimiter)) {
        SetNome(token);
    }

    if(std::getline(ss, token, kCommaDelimiter)) {
        SetDescricao(token);
    }

    if(std::getline(ss, token, kCommaDelimiter)) {
        SetPreco(std::stof(token));
    }
}
/*************************************************************
          metodo escrever sem parametro
*************************************************************/
void Produto::Escrever() {
    std::cout << GetNome() << "; "
              << GetDescricao() << "; "
              << GetPreco();
}
```

Campanha.h

```
#pragma once
#include <string>
#include <vector>
#include <fstream>

/*************************************************************
            definiçao de classe Campanha
*************************************************************/
class Campanha {
    public:

/*************************************************************
  declaramos metodo ler recebendo um stream como parametro
*************************************************************/
        void Ler(std::istream &is);

/*************************************************************
           declaramos metodo escrever sem parametro
*************************************************************/
        virtual void Escrever();

        const std::string GetTitulo();
        void SetTitulo(const std::string &titulo);

        const float GetDesconto();
        void SetDesconto(float desconto);

    private:
        std::string titulo_;
        float desconto_{0.0};
};
```

Campanha.cpp

```
#include <iostream>
#include <string>
#include <sstream>
#include "campanha.h"

const std::string Campanha::GetTitulo() {
    return titulo_;
}

void Campanha::SetTitulo(const std::string &titulo) {
    titulo_ = titulo;
}

const float Campanha::GetDesconto() {
```

```
    return desconto_;
}

void Campanha::SetDesconto(float desconto) {
    desconto_ = desconto;
}

/*************************************************************
       metodo ler recebendo um stream como parametro
*************************************************************/
void Campanha::Ler(std::istream &in) {
    constexpr char kCommaDelimiter{','};
    std::string line;

    std::getline(in, line);

    std::stringstream ss{line};
    std::string token;

    if(std::getline(ss, token, kCommaDelimiter)) {
        SetTitulo(token);
    }

    if(std::getline(ss, token, kCommaDelimiter)) {
        SetDesconto(std::stof(token));
    }
}
/*************************************************************
           metodo escrever sem parametro
*************************************************************/
void Campanha::Escrever() {
    std::cout << GetTitulo() << "; "
              << GetDesconto();
}
```

Loja.h

```
#pragma once
#include <memory>
#include <list>
#include "campanha.h"
#include "produto.h"

/*************************************************************
  lista de objestos produtodeloja que tem stock diponivel
*************************************************************/
class ProdutoDeLoja : public Produto {
    public:
```

```
        void SetStock(int stock);
        int GetStock();

        void Escrever() override;

    private:
        int stock_{0};
};

/*************************************************************
  lista de objestos campanhaativa verifica se ta ativa ou nao
*************************************************************/
class CampanhaAtiva : public Campanha {
    public:
        void SetAtiva(bool ativa);
        bool GetAtiva();

        void Escrever() override;

    private:
        bool ativa_{false};
};

/*************************************************************
            definiçao de classe Loja
*************************************************************/
class Loja {
    public:
        ~Loja();

/*************************************************************
              definição de operadores
*************************************************************/
        Loja operator+(std::list<std::shared_ptr<Produto>> &other);
        Loja operator+(std::list<std::shared_ptr<Campanha>> &other);

        void Escrever();

    private:
/*************************************************************
     definição de listas com ponteiros inteligentes
*************************************************************/
        std::list<std::shared_ptr<Produto>> produtos_;
        std::list<std::shared_ptr<Campanha>> campanhas_;
};
```

Loja.cpp

```cpp
#include <fstream>
#include <string>
#include "loja.h"
#include <iostream>

/*************************************************************
  lista de objestos produtodeloja que tem stock diponivel
*************************************************************/
void ProdutoDeLoja::SetStock(int stock) {
    stock_ = stock;
}

/*************************************************************
  lista de objestos produtodeloja que tem stock diponivel
*************************************************************/
int ProdutoDeLoja::GetStock() {
    return stock_;
}

void ProdutoDeLoja::Escrever() {
    Produto::Escrever();
    std::cout << "; "
              << GetStock();
}

/*************************************************************
  lista de objestos campanhaativa verifica se ta ativa ou nao
*************************************************************/
void CampanhaAtiva::SetAtiva(bool ativa) {
    ativa_ = ativa;
}

/*************************************************************
  lista de objestos campanhaativa verifica se ta ativa ou nao
*************************************************************/
bool CampanhaAtiva::GetAtiva() {
    return ativa_;
}

void CampanhaAtiva::Escrever() {
    Campanha::Escrever();
    std::cout << "; "
              << (GetAtiva() ? "ATIVA" : "NÃO ATIVA");
}

Loja::~Loja() {
    while(!produtos_.empty()) {
        produtos_.pop_back();
    }
```

```
    while(!campanhas_.empty()) {
        campanhas_.pop_back();
    }
}

/*************************************************************
              definição de operadores
*************************************************************/
Loja Loja::operator+(std::list<std::shared_ptr<Produto>> &other) {
    produtos_.insert(std::end(produtos_)
                   , std::begin(other)
                   , std::end(other));
    return *this;
}

/*************************************************************
              definição de operadores
*************************************************************/
Loja Loja::operator+(std::list<std::shared_ptr<Campanha>> &other) {
    campanhas_.insert(std::end(campanhas_)
                    , std::begin(other)
                    , std::end(other));
    return *this;
}

/*************************************************************
            metodo escrever no objeto loja
*************************************************************/
void Loja::Escrever() {
    std::cout << "Produtos em Stock da Loja" << std::endl;
    std::cout << "=========================" << std::endl;
    for(auto produto : produtos_) {
        (static_cast<ProdutoDeLoja *>(produto.get()))->Escrever();
        std::cout << std::endl;
    }
    std::cout << "=========================" << std::endl << std::endl;

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

    std::cout << "Campanhas da Loja" << std::endl;
    std::cout << "=================" << std::endl;
    for(auto campanha : campanhas_) {
        (static_cast<CampanhaAtiva *>(campanha.get()))->Escrever();
        std::cout << std::endl;
    }
    std::cout << "=========================" << std::endl << std::endl;
}
```