TP2 – GERENCIADOR

DOCUMENTAÇÃO

Gabriel Martins Silva
92539

UFV – 29/11/2017

1 – Introdução

O trabalho consiste em implementar um gerenciador de produtos para uma loja e/ou supermercado. Para tal, foram criadas diversas classe a fim de auxiliar a utilização e funcionalidade do programa: Dinheiro, Produto, Gerenciador de Produtos.

2 – Classes

Dinheiro.h

```
class Dinheiro{
private:
  int r, c; //real e centavo

public:
  Dinheiro(); //construtor padrao
  Dinheiro(int r, int c);

  int getReais() const;
  int getCentavos() const;

  void setReais(int valor);
  void setCentavos(int valor);

  Dinheiro operator=(const Dinheiro d);
  Dinheiro operator+(const Dinheiro d);
  Dinheiro operator-(const Dinheiro d);
  Dinheiro operator+=(const Dinheiro d);
  Dinheiro operator*(const double valor);
  double operator*(Dinheiro d);
  Dinheiro operator-=(const Dinheiro d);
  friend ostream& operator<<(ostream& os, const Dinheiro& d);
};
```

A classe dinheiro recebe duas variáveis “private”, r e c, respectivamente referentes a reais e centavos.

- Dinheiro() : O construtor padrão desta classe cria um dinheiro com zero reais e zero centavos.

- Dinheiro(int r, int c) : Esse construtor recebe como parâmetros reais e centavos para definir tal objeto com esses valores. Como forma de formatação tal construtor verifica a quantidade de centavos e caso seja maior que 99 aumenta em uma unidade o reais.

- int getReais() const : É um método constante que quando chamado retorna o valor “r” do objeto que o chamou.

- int getCentavos() cosnt: É um método constante que quando chamado retorna o valor “c” do objeto que o chamou.

- void setReais(int valor) : Pega o valor recebido como parâmetro e o atribui ao “r” do objeto. Não tem retorno.

- void setCentavos(int valor): Pega o valor recebido como parâmetro e o atribui ao “c” do objeto. Não tem retorno.

A seguir foram criadas diversas sobrecargas de operadores para que o compilador saiba como executar operações com “Dinheiro”. Uma sobrecarga em específico é a do “<<” para formatar o cout de modo que quando houver um cout de dinheiro será impresso na tela R$.

Produto.h

```
class Produto{
private:
  int codigo;
  double margemLucro;
  Dinheiro precoCusto, impostoMunicipal, precoVenda;
  char nome[49];

public:
  Produto();
  Produto(int codigo, char *nome, Dinheiro precoCusto, double margemLucro, Dinheiro impostoMunicipal);
  ~Produto();

  Dinheiro getPrecoCusto();
  Dinheiro getPrecoVenda();

  int getCodigo();
  const char *getNome();

  void setCodigo(int codigo);
  void setNome(char *nome);
  void setNomeBlank();
  void setLucro(int lucro);
  void setPrecoCusto(int r, int c);
  void setImposto(int r, int c);
  void setPrecoVenda();

  void cadastrarP();
  void printP();

  Produto & operator=(Produto p);
  friend ostream& operator<<(ostream& os, const Produto& p);
};
```

A classe produto recebe os seguintes parâmetros do tipo int: codigo;

do tipo double: margem de lucro; do tipo Dinheiro: preço de custo, imposto municipal, preço de venda; do tipo char: nome do produto.

- **Produto()** : Construtor padrão que define um Produto nulo quando criado, ou seja, nome em branco, codigo = -1, margem de lucro = 0, preço de custo = 0, imposto municipal = 0, preço de venda = 0.

- **Produto(int codigo, char *nome, Dinheiro precoCusto, double margemLucro, Dinheiro impostoMunicipal)** : Defini todas variaveis de dinheito de acordo com os parâmetros recebidos na criação de tal objeto.

- ~Produto() : Destrutor de Produto, apenas defini o nome do produto como em branco.

- Dinheiro getPrecoCusto() : Retorna um um objeto do tipo dinheiro com as informações de preço de custo.

- Dinheiro getPrecoVenda() : Retorna um um objeto do tipo dinheiro com as informações de preço de venda. Sendo o preço de venda definido por precoVenda = precoCusto + (precoCusto*(margemLucro)*0.01) + impostoMunicipal.

- int getCodigo() : Retorna o código do produto que chamou o metodo.

- const char *getNome() : Retorna o nome do produto.

Os métodos “set” apenas atribuem os parâmetros recebidos em suas respectivas variáveis do objeto.

- void cadastraP() : Método criado para pedir para o usuário digitar na tela as informações de um produto e assim atribuir essas informações para um produto auxiliar que posteriormente será usado para gravar tal informação no arquivo data.dat criado.

- void printP() : Tem a mesma função da sobrecarga de operador “<<”, ou seja, imprimir na tela as informações de um prodtuo em específico.

Gerenciadorprodutos.h

```
class GerenciadorProdutos{
private:
  Produto *produtos;
  static int contador;
  static int tamanho;

public:

  GerenciadorProdutos();
  GerenciadorProdutos(int maxProdutos);
  ~GerenciadorProdutos();

  void armazenaProduto();
  void removeProduto(int codigo);
  void removeTodosProdutos();
  void setProdutoNull(int codigo);

  Produto getProduto(int codigo);
  Produto getIesimoProduto(int i);

  int getNumProdutosCadastrados() const;
  int getTamanho() const;
  void printAllProdutos() const;

  friend ostream& operator<<(ostream& os, const Produto& p);

};
```

Classe criada para gerenciar uma array de “Produto”. Tem como variaveis globais em gerenciadorprodutos.cpp : fstream file, Produto produto. Para que os métodos possam utiliza-los.

Tem como variaveis “private” um array alocado dinâmicamente de Produto. E duas variáveis estáticas utilizadas para manter controle da quantidade atual de produtos e do número máximo de produtos que podem ser inseridos no arquivo data.dat.

- **GerenciadorProdutos()** : Construtor padrão que cria um array de produtos com apenas um produto em branco, ou seja, definido pelo

construtor padrão de Produtos e define o contador de produtos para 1 e o número máximo de produtos para 1.

- GerenciadorProdutos(int maxProdutos) : Construtor que cria um array de produtos com a quantidade máxima definida pelo valor do parametro recebido pelo método.

- ~GerenciadorProdutos() : Destrutor que deleta a array de produtos e diminui o contador.

- void armazenaProduto() : Método criado para armazenar um produto no arquivo data.dat. Primeiro checa se tem espaço no arquivo para inserir mais um produto, se houver ele faz uso da variável Produto global para chamar a função de Produto que define os parâmetros desse produto e por fim atravez do file.write insere tais dados no arquivo data.dat e aumenta o contador de produtos em uma unidade.

- void removeProduto(int codigo) : A função dessa método é de “deletar” um produto que tenha o mesmo código do recebido como parâmetro pelo método. Para isso ele cria um arquivo auxiliar e copia todos os produtos que não serão deletados e por fim renomeia tal arquivo para data.dat, assim causando o mesmo efeito de se deletar um produto da lista.

- void removeTodosProdutos() : Cria um arquivo com o nome data.dat para assim sobrepor o anterior com o mesmo nome e usando a flag ios::trunc esse novo arquivo é gerado em branco, logo, deletando todos os produtos do arquivo data.dat.

- Produto getProduto(int codigo) : Retorna um prodtuo que tenha o mesmo codigo que o recebido pelo método.

- Produto getIesimoProduto(int i) : Retorna o produto que esteja na posição recebida pelo método.

- int getNumProdutosCadastrados() const : Retorna a quantidade de produtos cadastrados no arquivo data.dat.

- int getTamanho() const : Retorna a quantidade máxima de produtos que podem ser cadastrados no arquivo data.dat.

3 – Main

O arquivo main desse projeto está nomeado como supermercado.cpp, ele será executado para utilizar todas as funcionalidades do sistema de gerenciador criado.

Para executar o programa corretamente compile usando o Makefile criado: make all;

Agora digite no terminal: ./supermercado.exe <tamanhoMAX>, (sendo tamanhoMAX um número que define a quantidade máxima de produtos que podem ser inseridos no arquivo data.dat).

O arquivo main tem como variáveis globais: fstream file, Produto produtos; Para que o programa possa fazer uso.

Funções criadas em supermercado.cpp:

- **void ordena()** : Função criada para ordenar os produtos do arquivo data.dat a partir de seu código. Para isso cria uma array de produtos auxiliar que vai receber todos os produtos de data.dat e usando o qsort vai ordena-las e por fim escrever no arquivo data.dat de já ordenado.

- **int compare(const void *a, const void *b)** : Função que o “qsort” chama para definir como ordenar a array.

- **void displayMenu()** : Função para imprimir na tela o menu de uso para o usuário.

- **void listaP()** : Função utilizada para listar todos os produtos armazenados em data.dat.

- **void printProduto(int codigo)** : Utilizada para imprimir um produto que tenha o mesmo codigo que o recebido como parâmetro pela função, caso encontrado. Não utilizar essa função se o arquivo data.dat estiver vazio.

- **void enter()** : Função para pedir que o usuário aperte ENTER para prosseguir e assim limpa o terminal para uma experiência mais limpa.

Funcionamento de supermercado.cpp:

Quando executado, inicialmente o programa limpa o terminal, verifica se foi executado corretamente (colocando como parametro no terminal o tamanho maximo de produtos), cria uma variavel do tipo GerenciadorProdutos p(k), sendo k o tamanho maximo.

A seguir ele abre o arquivo data.dat ou cria um se não tiver e testa se foi possível abrir tal arquivo. Antes de entrar em um loop infinito o programa ordena todos os produtos presentes no arquivo data.dat.

Programa entra em um loop infinito que será a parte de interação com o usuário. Primeiramente imprime o displayMenu e espera pera ação do usuário. O programa, assim, fica em um loop fazendo as operações que o usuário pede até que recebe do usuário a ordem de parada (digitar 6) e assim ele encerra suas atividades.

MAKEFILE:

```
all: supermercado.exe

supermercado.exe: dinheiro.o produto.o gerenciadorprodutos.o supermercado.o
	g++ supermercado.o dinheiro.o produto.o gerenciadorprodutos.o -o supermercado.exe

dinheiro.o: dinheiro.h dinheiro.cpp
	g++ -c dinheiro.cpp

produto.o: produto.h produto.cpp
	g++ -c produto.cpp

gerenciadorprodutos.o: gerenciadorprodutos.h gerenciadorprodutos.cpp
	g++ -c gerenciadorprodutos.cpp

supermercado.o: supermercado.cpp
	g++ -c supermercado.cpp

limpa:
	rm -f *.exe *.o
```

Exemplos de funcionamento:

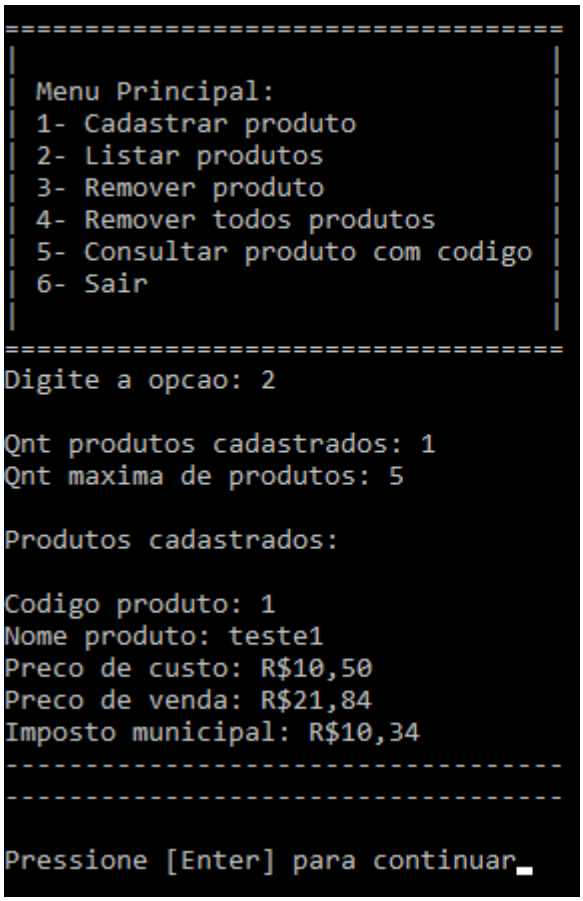
```
pcpika2fly@User-PC: /mnt/c/Users/User/Desktop/PROG/TP2
pcpika2fly@User-PC:/mnt/c/Users/User/Desktop/PROG/TP2$ make all
g++ -c gerenciadorprodutos.cpp
g++ supermercado.o dinheiro.o produto.o gerenciadorprodutos.o -o supermercado.exe
pcpika2fly@User-PC:/mnt/c/Users/User/Desktop/PROG/TP2$
```

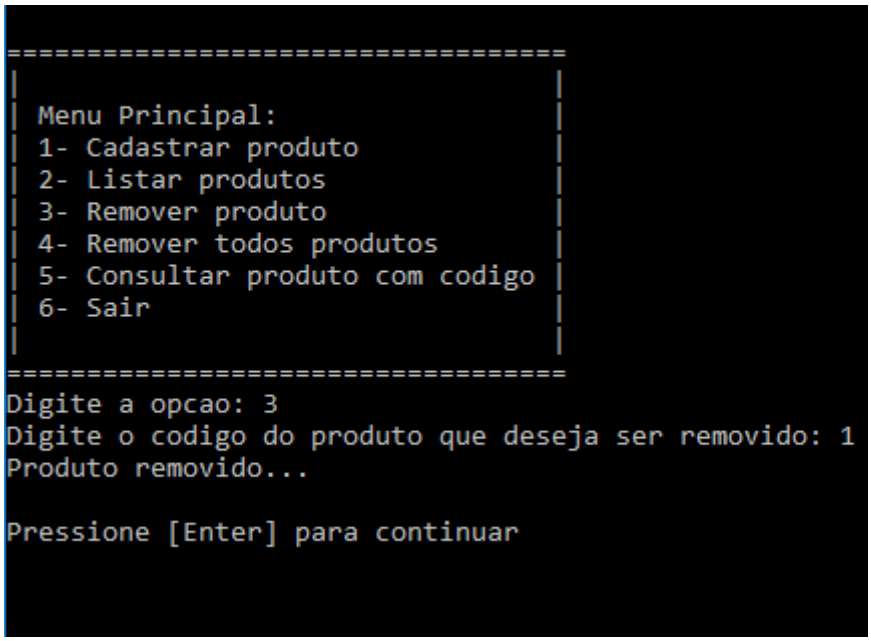
```
pcpika2fly@User-PC: /mnt/c/Users/User/Desktop/PROG/TP2
pcpika2fly@User-PC:/mnt/c/Users/User/Desktop/PROG/TP2$ make all
g++ -c gerenciadorprodutos.cpp
g++ supermercado.o dinheiro.o produto.o gerenciadorprodutos.o -o supermercado.exe
pcpika2fly@User-PC:/mnt/c/Users/User/Desktop/PROG/TP2$ ./supermercado.exe 5
```

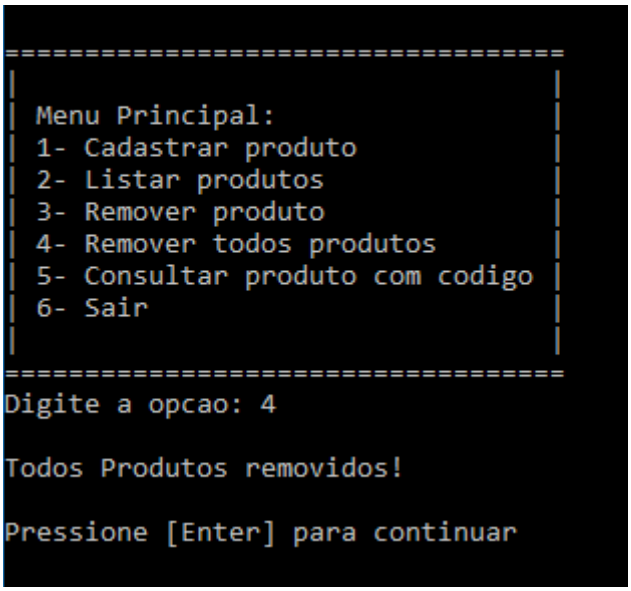
Utilizado no exemplo o número 5 para que seja a quantidade máxima de produtos que podem ser cadastrados no arquivo data.dat.

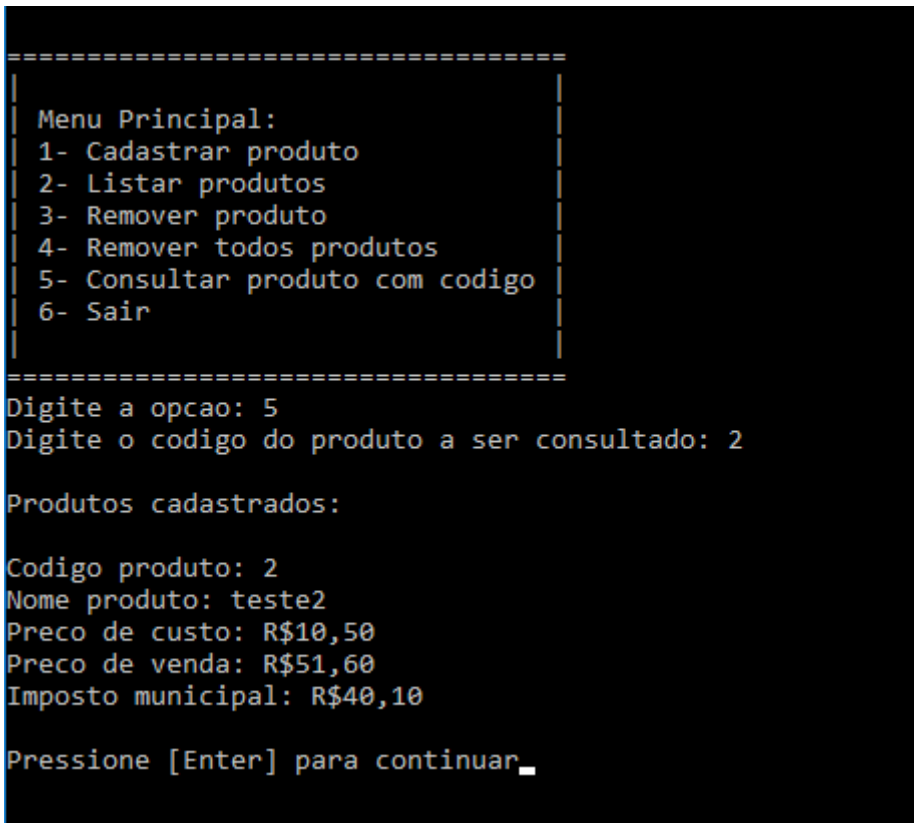
```
====================================
|                                  |
| Menu Principal:                  |
| 1- Cadastrar produto             |
| 2- Listar produtos               |
| 3- Remover produto               |
| 4- Remover todos produtos        |
| 5- Consultar produto com codigo  |
| 6- Sair                          |
|                                  |
====================================
Digite a opcao:
```

```
====================================
|                                  |
| Menu Principal:                  |
| 1- Cadastrar produto             |
| 2- Listar produtos               |
| 3- Remover produto               |
| 4- Remover todos produtos        |
| 5- Consultar produto com codigo  |
| 6- Sair                          |
|                                  |
====================================
Digite a opcao: 1
Digite as informacoes do produto:
Codigo: 1

Nome: teste1

Preco de custo(REAL CENTAVO): 10 50

Margem de Lucro(%): 10

Imposto Municipal (REAL CENTAVO): 10 34

Produto foi armazenado com sucesso!

Pressione [Enter] para continuar
```


Menu Principal:
1- Cadastrar produto
2- Listar produtos
3- Remover produto
4- Remover todos produtos
5- Consultar produto com codigo
6- Sair
Digite a opcao: 2
Qnt produtos cadastrados: 1
Qnt maxima de produtos: 5
Produtos cadastrados:
Codigo produto: 1
Nome produto: teste1
Preco de custo: R$10,50
Preco de venda: R$21,84
Imposto municipal: R$10,34
Pressione [Enter] para continuar


Menu Principal:
1- Cadastrar produto
2- Listar produtos
3- Remover produto
4- Remover todos produtos
5- Consultar produto com codigo
6- Sair
Digite a opcao: 3
Digite o codigo do produto que deseja ser removido: 1
Produto removido...
Pressione [Enter] para continuar


Menu Principal:
1- Cadastrar produto
2- Listar produtos
3- Remover produto
4- Remover todos produtos
5- Consultar produto com codigo
6- Sair
Digite a opcao: 4
Todos Produtos removidos!
Pressione [Enter] para continuar


Menu Principal:
1- Cadastrar produto
2- Listar produtos
3- Remover produto
4- Remover todos produtos
5- Consultar produto com codigo
6- Sair
Digite a opcao: 5
Digite o codigo do produto a ser consultado: 2
Produtos cadastrados:
Codigo produto: 2
Nome produto: teste2
Preco de custo: R$10,50
Preco de venda: R$51,60
Imposto municipal: R$40,10
Pressione [Enter] para continuar