INF112 – Trabalho Prático 2
Documentação

Alunos:

- Armando Serrado de Almeida - 92561
- Rodrigo Eduardo de Oliveira Bauer Chichorro – 92535

Data: 30/11/2017

1. Introdução:

Este trabalho consiste em desenvolver um programa para gerenciar um pequeno cadastro de produtos de uma loja. Ao ser executado, o programa deverá exibir um menu similar ao apresentado na especificação.

O programa consiste em uma pasta de nome “TP2_PARMAHARD”, que contém 9 arquivos:

- dinheiro.h – contém uma classe de nome DINHEIRO
- dinheiro.cpp – implementação dos métodos da classe DINHEIRO
- produto.h – contém uma classe de nome PRODUTO
- produto.cpp – implementação dos métodos da classe PRODUTO
- gerenciadorProdutos.h – contém uma classe de nome GerenciadorProdutos
- gerenciadorProdutos.cpp – implementação dos métodos da classe GerenciadorProdutos
- terminal.h – Declaração de funções usadas na interface do programa.
- terminal.cpp – Implementação de funções usadas na interface do programa.
- main.cpp – contém a função principal do programa.
- makefile – makefile responsável por compilar o programa.

Ao ser chamado no terminal, o makefile gerará um executável Gerenciador.exe, que por sua vez, ao ser executado, iniciará o programa em si.

Durante a execução, o programa criará – caso não exista - um arquivo binário de nome produtos.dat, que armazena os produtos cadastrados pelo programa.

2. Avisos:

- O programa PARMAHARD foi desenvolvido e testado apenas em plataforma LINUX, não garantimos seu funcionamento em plataformas diferentes.

- Esse projeto é sem fins lucrativos, caso você tenha pago por sua utilização, EXIJA seu dinheiro de volta. Não nos responsabilizamos por este tipo de infortúnio.

- Email de contato:
armando.almeida@ufv.br
reobcnegf@gmail.com

3. Classe DINHEIRO

- 3.1. Atributos:

Todos os atributos da classe são privados.

int Reais – Armazena a parte em reais do valor.

int Centavos – Armazena a parte em centavos do valor.

- 3.2. Métodos:

Todos os métodos da classe são públicos.

DINHEIRO(const int, const int) – Construtor da classe. Recebe como parâmetros dois inteiros. O primeiro parâmetro será o valor de int Reais do objeto e o segundo parâmetro será o valor de int Centavos.

DINHEIRO() - Construtor vazio da classe. Atribui 0 para ambos os atributos do objeto.

DINHEIRO(const DINHEIRO &copiado) – Construtor de cópia da classe. Passa os valores dos atributos de const DINHEIRO copiado para o objeto que chamou o construtor de cópia.

int getReais() const – Retorna o valor armazenado pelo objeto em int Reais.

int getCentavos() const – Retorna o valor armazenado pelo objeto em int Centavos.

void setReais(const int) – Atribui o valor do parâmetro para o valor armazenado pelo objeto em int Reais.

void setCentavos(const int) – Atribui o valor do parâmetro para o valor armazenado pelo objeto em int Centavos.

DINHEIRO & operator=(const DINHEIRO &) - Sobrecarga do operador = para a classe. Atribui os valores de Reais e Centavos do parâmetro (que estará do lado direito do operador) para os respectivos atributos do objeto do lado esquerdo do operador.

DINHEIRO operator+(DINHEIRO) const – Sobrecarga do operador + para a classe. Cria um objeto DINHEIRO ObjetoResultado, que armazenará o valor da soma entre os objetos à esquerda e à direita do operador, e, em seguida, retorna ObjetoResultado. Ex.: Ao somar 5,80 e 7,60, Objeto Resultado terá o valor de 13,40.

DINHEIRO operator-(DINHEIRO) const – Sobrecarga do operador - para a classe. Cria um objeto DINHEIRO ObjetoResultado, que armazenará o valor da subtração entre os objetos à esquerda e à direita do operador, e, em seguida, retorna ObjetoResultado. Caso o resultado seja negativo, ObjetoResultado terá o valor 0,00. Ex.: Ao subtrair 7,60 e 5,80, ObjetoResultado terá o valor 1,80. Mas, ao subtrair 5,80 e 7,60, ObjetoResultado terá o valor de 0,00.

DINHEIRO& operator+=(const DINHEIRO) – Sobrecarga do operador += para a classe. Soma os valores dos objetos à esquerda e à direita do operador, e atribui o resultado ao objeto à esquerda. Ex.: Ao somar 5,80 e 7,60, o objeto da esquerda passará a ter o valor de 13,40.

DINHEIRO& operator-=(const DINHEIRO) – Sobrecarga do operador -= para a classe. Subtrai os valores dos objetos à esquerda e à direita do operador, e atribui o resultado ao objeto à esquerda. Caso o resultado seja negativo, o resultado será 0,00. Ex.: Ao subtrair 5,80 e 7,60, o objeto da esquerda passará a ter o valor de 0,00.

DINHEIRO operator*(const double) – Sobrecarga do operador * para classe. Multiplica o valor do objeto por um número (const double) e retorna o resultado.

friend std::ostream& operator<<(std::ostream&, const DINHEIRO&) - Sobrecarga do operador << para a classe. Quando um std::cout é usado em algum DINHEIRO, o objeto será impresso no formato R$ reais,centavos. Ex.: Para um objeto com 15 reais e 5 centavos, temos “R$ 15,05”.

4. Classe PRODUTO:

- 4.1. Atributos:

Todos os atributos da classe são privados.

İnt codigo – Armazena o código do produto.

char nome[50] – Armazena o nome do produto. A intenção original da dupla era utilizar um arranjo dinâmico char* nome, mas nos deparamos com diversos problemas e decidimos utilizar um arranjo automático, já que a especificação garante um tamanho máximo de 49 caracteres para o nome.

DINHEIRO precoCusto – Armazena o preço do custo do produto.

int margemLucro – Armazena a margem de lucro do produto em %.

DINHEIRO impostoMunicipal – Armazena o valor do imposto municipal sobre o produto.

DINHEIRO precoVenda – Armazena o preço de venda do produto, sendo este precoCusto*(1 + (margemLucro/100)) + impostoMunicipal.

- 4.2. Métodos:

Todos os métodos da classe são públicos.

PRODUTO(int codigo, char* nome, DINHEIRO precoCusto, int margemLucro, DINHEIRO impostoMunicipal) – Construtor da classe. O valor de cada um dos parâmetros

é atribuído ao seu corresponde no objeto e, em seguida, o valor de precoVenda é calculado segundo a fórmula descrita acima.

PRODUTO() - Construtor vazio da classe. Um PRODUTO com os valores abaixo representa um PRODUTO vazio, e é um caso especial em várias funções do programa. O construtor inicializa o objeto com os valores:

- -1 para codigo
- '\0' para todas as posições de nome[50]
- 0,00 para precoCusto
- 0 para margemLucro
- 0,00 para impostoMunicipal
- 0,00 para precoVenda

PRODUTO(const PRODUTO &copiado) – Construtor de cópia para a classe. Copia o valor de cada um dos atributos de const PRODUTO copiado para seus correspondentes no objeto.

~PRODUTO() - Destrutor da classe. Como nenhum atributo da classe é ponteiro, o destrutor não possui nenhum comando.

int getCodigo() const – retorna o valor armazenado em codigo.

char* getNome() const – Retorna um arranjo de caracteres contendo o nome do produto.

DINHEIRO getPrecoCusto() const – Retorna o valor de precoCusto.

int getMargemLucro() const – Retorna o valor de margemLucro.

DINHEIRO getImpostoMunicipal() const – Retorna o valor de impostoMunicipal.

DINHEIRO getPrecoVenda() const – Retorna o valor de precoVenda.

PRODUTO & operator=(const PRODUTO &) - Sobrecarga do operador = para a classe. Atribui os valores dos atributos do parâmetro (objeto à direita do operador) para os seus correspondentes no objeto à esquerda do operador.

friend std::ostream& operator<<(std::ostream&, const PRODUTO&) - Sobrecarga do operador << para a classe. Quando um std::cout é usado em algum PRODUTO, os atributos do objeto serão impressos na tela nessa forma:

```
Produtos cadastrados:
=====================================================
|                                                   |
|  Nome: Capivaras sao os                           |
|  Codigo: 3                                        |
|  Preco de Custo: R$ 0,00                          |
|  Margem de Lucro (%): 0                           |
|  Valor do Imposto Municipal: R$ 2,00              |
|  Preco de Venda: R$ 2,00                          |
|                                                   |
=====================================================
=====================================================
|                                                   |
|  Nome: maiores roedores do mundo                  |
|  Codigo: 5                                        |
|  Preco de Custo: R$ 10,00                         |
|  Margem de Lucro (%): 25                          |
|  Valor do Imposto Municipal: R$ 0,10              |
|  Preco de Venda: R$ 12,60                         |
|                                                   |
=====================================================
Pressione 'Enter' para sair...
```

5. Classe GerenciadorProdutos:

- 5.1. Atributos:

Todos os atributos da classe são privados.

PRODUTO* produtos – arranjo dinâmico que armazena os PRODUTOs cadastrados no programa. Os PRODUTOs são armazenados em ordem crescente de código. Caso sobrem posições, elas armazenarão um PRODUTO vazio.

int numProdutos – armazena o número máximo de PRODUTOs que podem ser armazenados pelo objeto. O numProdutos do objeto usado no programa tem o valor de 100.

- 5.2. Métodos:

Todos os métodos da classe são públicos.

GerenciadorProdutos(const int maxProdutos) – Construtor da classe. Primeiro, o método atribui para o atributo numProdutos o valor do parâmetro maxProdutos e aloca numProdutos posições de memória para o atributo *produtos. Em seguida, o método tenta abrir o arquivo produtos.dat. Depois, o método testa se produtos.dat foi aberto com sucesso. Se sim, os produtos armazenados em produtos.dat são lidos pelo programa e armazenados em *produtos. Caso sobrem posições no arranjo, as posições restantes estarão preenchidas com PRODUTOs vazios. Depois disso, o arquivo é fechado. Se não, *produtos armazenará apenas PRODUTOs vazios, até que o usuário adicione algum PRODUTO novo.

GerenciadorProdutos() - Construtor vazio da classe. Atribui 0 para numProdutos e aloca 1 PRODUTO para *produtos. Este método não tem utilidade prática, já que nenhum GerenciadorProdutos é declarado sem passar maxProdutos como parâmetro.

~GerenciadorProdutos() - Destrutor da classe. Primeiro, tenta abrir produtos.dat. Caso o arquivo não exista, ele será criado. Em seguida, todos os PRODUTOs não-vazios armazenados em *produtos serão gravados em produtos.dat. Finalmente, produtos.dat é fechado e *produtos é deleteado.

int getNumProdutos() const – Retorna o valor de numProdutos.

PRODUTO getIesimoProduto(int i) const – Retorna o PRODUTO armazenado na posição i de *produtos.

PRODUTO getProduto(int codigo) const – Retorna o PRODUTO armazenado em *produtos que possui o código igual ao parâmetro. Caso não exista tal PRODUTO, a função retorna um PRODUTO vazio.

int getNumProdutosCadastrados() const – Retorna o número de PRODUTOs não-vazios armazenados em *produtos.

void OrdenarProdutos() - Ordena os PRODUTOs em *produtos em ordem crescente de código, com os PRODUTOs vazios no fim. O método de ordenação usado é um Insertion Sort alterado para tratar os PRODUTOs vazios. Essa escolha se deve ao fato de que, como adicionamos apenas 1 PRODUTO por vez, precisamos apenas “inserir” o novo PRODUTO na posição correta, e o Insertion Sort é perfeito para isso.

void armazenaProduto(const PRODUTO) – Caso o código do PRODUTO passado como parâmetro seja maior ou igual a 0, não exista outro PRODUTO com o mesmo código e o número de PRODUTOs cadastrados seja menor que número máximo de PRODUTOs que podem ser cadastrados, o PRODUTO passado como parâmetro é armazenado na posição getNumProdutosCadastrados(), pois as posições de 0 até getNumProdutosCadastrados estão preenchidas. Em seguida, chamamos OrdenarProdutos() para inserir o PRODUTO novo na posição correta.

void removeProduto(const int codigo) – Busca a posição de *produtos que armazena um PRODUTO com o código igual ao parâmetro e transforma esse PRODUTO em um PRODUTO vazio. Em seguida, chama OrdenarProdutos().

void removeTodosProdutos() – Transforma todos os PRODUTOs armazenados em *produtos em PRODUTOs vazios.

6. Terminal:

Os arquivos terminal.h e terminal.cpp não são classes, mas sim arquivos contendo funções relacionadas à interface do programa.

- 6.1. Funções:

void init() - Inicializador do 'terminal', declarando um objeto GerenciadorProdutos com parâmetro '100'.

void draw_main_menu() - Função que desenha o 'menu inicial' com suas opções de escolha.

void draw_produto(const PRODUTO) - Função que recebe e desenha um objeto do da classe PRODUTO.

void cadastro(GerenciadorProdutos&) - Recebe um objeto GerenciadorProdutos que recebe do usuário as informações pra o 'cadastro' de um produto.

void listar(GerenciadorProdutos&) - Recebe um objeto GerenciadorProdutos e lista os itens de seu 'produtos' (array contendo os objetos de PRODUTO).

- 6.2. Interface:

O usuário, quando inicia o programa, recebe como saída, opções lsitadas de '1 a 6':

Cadastrar Produto – Desenha no terminal informações do 'produto' a ser cadastrado, necessitando a interferência do usuário.

Listar Produtos – Lista os produtos cadastrados ao usuário.

Remover Produto – Requer que o usuário envie o código do produto, que será removido da lista de produtos cadastrados.

Remover Produtos – Limpa o 'cache' de produtos cadastrados.

Consultar Produto com Codigo – Recebe do usuário o código de um produto, em seguida, imprime as informações do produto no terminal.

Sair – Finaliza o programa e salva os dados dos produtos num arquivo externo ("produtos.dat").