RELATÓRIO - TRABALHO PRÁTICO ETAPAS 1 E 2

Matheus Mignoni - 195843 - [mmignoni@inf.ufrgs.br]
Luís Fernando Schauren - 93102 - [lfschauren@inf.ufrgs.br]
Pedro Henrique Frozi de Castro e Souza - 161502 - [phfcsouza@inf.ufrgs.br]
Ciência da Computação
INF01120 - Técnicas de Construção de Programas

1. INTRODUÇÃO

Este relatório visa elucidar as alterações que foram realizadas no código do programa “e-Commerce” disponibilizado para realização das etapas 1 e 2 do trabalho prático da disciplina. Nele veremos uma descrição das modificações que realizamos no código em ambas as etapas, exibindo em quais técnicas daquelas que foram vistas em aula que nos baseamos.

2. ATIVIDADES DESENVOLVIDAS NA ETAPA 1

As alterações realizadas na primeira etapa estiveram centradas na aplicação de técnicas de convenção e boas práticas, apesar de algumas alterações também terem sido realizadas baseadas nas técnicas da área da refatoração de código. Nos dois próximos itens, mostraremos estas alterações de forma mais detalhada.

2.1. CONVENÇÃO E BOAS PRÁTICAS

Alteramos os nomes de variáveis e de funções para um formato mais legível, intuitivo e de acordo com a seguinte convenção:

Nomes de funções e variáveis: *primeiraLetraMinuscula*
Tipos de Dados: MAIUSCULAS

Variáveis do tipo *int* que tinham o propósito de testar se uma determinada condição era verdadeira ou falsa foram alteradas para o tipo *boolean* e suas atribuições alteradas para os valores correspondentes (*false* ou *true*).

As variáveis que não eram utilizadas foram excluídas. Também verificamos a inicialização correta das variáveis antes de seu uso.

Abaixo segue uma lista de todas as alterações que foram feitas nessa etapa:

int codigo para int codigoProduto
int quantidade para int quantidadeProduto
int achou para bool achouProduto
int confirmacao para bool carrinhoDisponivel
int comparaCodigo para int codigoCompara (seguir a mesma convenção da variável anterior)
char string[MAX_LINHA] para char stringTemp[MAX_LINHA]

Linha 635:
Alterada a função imprime_vest para imprimeVestuario assim como suas ocorrências nas linhas 940 e 948
Alterada a variável produtovest para produtoVestuario assim como suas ocorrências nas linhas 638-640, 659 e 929-948

Linha 644:
Alterada a função imprime_eletro para imprimeEletro assim como suas ocorrências nas linhas 777, 785, 793, e 948
Alterada a variável produto para produtoEletro assim como suas ocorrências nas linhas 647-649, 658, 766-794, 941, 948 e 949
OBS: Nas linha 941 e 949 tem uma ocorrência produto.código que foi substituída por produtoEletro.codigo mas que pode se referir a produtoVestuario.codigo - provavelmente um erro do código original que foi detectado nessa fase da refatoração.

OBS: Linha 656 váriavel opção é utilizada em várias funções. Duas alternativas:
a) criarmos variáveis específicas para cada função ou
b) inicializarmos corretamente a váriavel antes da função.

Linha 664 e 665:
Variáveis precomin e precomax por precoMin e precoMax respectivamente assim como suas ocorrências nas linhas 752, 756, 791, 891, 893 e 946

Linha 667:
Alterada a varável produtosencontrados para produtosEncontrados assim como suas ocorrências nas linhas 681, 780, 788, 796, 801, 812, 943, 951, 957, 968,

Linha 668:
A variável inteira maisuma foi substituída por uma booleana continuaPesquisa pois é a isso que se refere. Ocorrências nas linhas 984, 989, 998 também foram substituídas.
OBS: Na linha 1193 essa variável maisuma volta a ocorrer mas refere-se à pergunta se deseja continuar a excluir itens do carrinho.

A variável inteira confirmacao foi substituída por uma variável booleana opcaoValida, pois serve para verificar se o usuário inseriu uma opção correta na escolha do menu. Também substituí suas ocorrências nas linhas 680-697 (5 ocorrências), 713-731 (6 ocorrências), 832-851 (8 ocorrências), 867-887 (8 ocorrências), 904-920 (6 ocorrências), 977-997 (5 ocorrências).

OBS: Variável mudou o tipo de inteiro para booleana. Verificar todas ocorrências usando as atribuições 0 e 1.

Linha 666:
Excluída a variável inteira "inteiro" pois não é utilizada ao longo do texto.
Aproveitei para declarar a variável booleana continuaPesquisa nessa linha para preservar a ordem da numeração.

função pesquisar para pesquisaProduto
função vizualizacompras para visualizaCompras
variável contalinha foi substituída por contaLinhas
exclusão da variável inteira "achou", pois não é utilizada na função cancelaTodasCompras
variável contalinha foi substituída por contaLinhas

variável codigoExclui foi substituída por codigoExcluir
variável inteira achou foi substituída pela variável booleana encontrouProduto
variáveis inteiras "maisuma" e "confirmacao" foram substituídas pelas variáveis booleanas continuaExcluindo e opcaoValida respectivamente na função excluiProdutoCarrinho

* = Alterado neste commit
- = Será alterado nas próximas etapas

1. int contaComprasSuspensas()
 * Alteração do nome da variável tipo char "string" para "linha";
 * Alteração do nome da variável "CarrinhosAbertos" para "listaCarrinhosAbertos";
 - O array tipo int "carrinhosAbertos" no momento não tem utilidade alguma para esta função, apesar de armazenar a lista de carrinhos abertos;
 - Em várias partes do programa há a contagem da quantidade de usuários existentes no arquivo "usuarios.txt". Esta operação deverá ser transformada em uma função;

2. void menuCLIENTE(char nome[], int cadastro)
 * Nome da função alterada para MenuCliente;
 * Alteração do nome da variável local "cadastroEmCar" para "cadastroEmChar";
 * Criação das variáveis do tipo bool "opcaoValida" e "continua" para controle dos laços;
 * Devido ao item anterior, não é mais necessária a variável int "respostaAlg";

3. void relatorioComprasEmAndamento()
 * Alteração do nome da variável tipo char "string" para "linha";
 - Substituição do trecho de código que verifica a quantidade de carrinhos abertos(suspensos) pela função "contaComprasSuspensas" já existente no programa. A lista de carrinhos abertos deverá ser um parâmetro por referência, pois será utilizada na função;
 - Devido à alteração sugerida no intem anterior, as variáves "totalCarrinhosAbertos", "contador", "linha" e "cadastroEmChar" poderão ser eliminadas;

4. void vendas()
 * Alteração do nome da variável tipo char "string" para "linha";
 * Alteração do tipo da variável "encontrou" de int para bool;
 * Alteração do nome da variável tipo float "totalVendas" para "valorTotalVendas";
 - Alteração da cadeia de if's para melhorar o desempenho da operação e torná-la mais intelegível;

5. void verificaEstoque()
 * Alteração do nome da variável tipo char "string" para "linha";

FUNÇÃO void clientesCadastrados() {linha 1852}
char string[MAX_LINHA] para stringTemp[MAX_LINHA];

FUNÇÃO void menuGerente() {linha 1880}
char resposta para char opcaoMenuGerente
int respostaalg para int respostaMenuGerente

FUNÇÃO telaLogin() {linha 1957}
int cadastro para int codCadastro //evitar usar 'num' como cadastroNum
int comparacadastro para int codComparaCadastro
int achou para bool achouCadastro

int GouC para int verificaEntreClienteGerente
char string[MAX_LINHA] para char stringTemp[MAX_LINHA]
char nome[TAM_NOME_MAX] para char nomeCliente[TAM_NOME_MAX]
int i removida, pois não era usada

2.2. REFATORAÇÃO

Não fizemos nenhuma alteração com relacão à refatoração nessa primeira etapa, limitando as mudanças às convenções e boas práticas vistas em aula.

3. ATIVIDADES DESENVOLVIDAS NA ETAPA 2

3.1 REFATORAÇÃO

Realizamos ajustes nas funções de forma a prepará-las para a modularização. Checamos quais eram suas dependêcias, declaração e escopo de variáveis de forma a garantir o acoplamento. Esse passo foi importante para decidir quais partes do código seriam modularizadas.

A lista de funções do código após a refatoração é essa:

void cancelaTodasCompras(int cadastro)
void clientesCadastrados()
int contaComprasSuspensas()
void excluiProdutoCarrinho(int cadastro)
void fechaCompra(int cadastro)
void imprimeEletro(ELETRO produtoEletro)
void imprimeVestuario(VESTUARIO produtoVestuario)
void insereProdutoCarrinho(int cadastro)
void menuCliente(char nome[], int cadastro)
void menuGerente(char nome[], int cadastro)
void mudaCor (Colors cor)
void novoUsuario()
void pesquisaProduto()
void relatorioComprasEmAndamento()
void showInfo()
int telaInicial()
void telaLogin()
void vendas()
void verificaEstoque()
void visualizaCompras(int cadastro)

O próximo passo é a modularização, como veremos a seguir.

3.2. MODULARIZAÇÃO

- Alterações referentes a modularidade;
- Criação do projeto no codeblocks;

Nesta etapa usamos a estratégia de decomposição (*top-down*) para dividirmos o código original em várias

funções. Dessa forma é possível fazer alterações nas funções sem a preocupação de ter que analisar o código completo ou afetar outras funções relacionadas. Além disso, a modularização facilita a implementação de novas funcionalidades sem afetar muitos módulos (princípio da continuidade) bem como a proteção contra situações inesperadas, mantendo os erros dentro do módulo que foi criado.

O código então foi dividido criando arquivos de interface (.h) e arquivos de implemetação (.cpp) para aquelas funções que julgamos que seria interessante separar do código original, baseado nos princípios da modularidade:

- Decomponibilidade
- Componibilidade
- Compreensibilidade
- Continuidade
- Proteção

Criamos os seguintes arquivos de interface e seus respectivos arquivos de implementação:

telaInicial - Primeira tela do sistema
telaLogin - Uma das opções da tela inicial. Tela para acesso ao sistema
novoUsuario - uma das opções da Tela Inicial. Função que cadastra novos usuários.
showInfo - outra opção da tela inicial. Exibe informações do programa
mudaCor - função que é utiliza por praticamente todos os módulos que exibem alguma informação na tela.
menuCliente - para exibir as opções do menu relativas ao Cliente
menuGerente - para exibir as opções do menu relativas à Gerência
pesquisaProduto - função utilizada para lista os Produtos

Também criamos um arquivo de definições globais com as constantes que são compartilhadas por todos os outros arquivos e por duas estruturas amplamente utilizadas. Um dos motivos que nos levaram à declarar estes como globais é que se tratam de objetos pouco mutáveis e muito usados pelo programa.

4. CONCLUSÃO

Nestas etapas do trabalho prático tivemos a oportunidade de aplicar todos os conceitos de refatoração e modularização vistos em aula. Todas as atividades foram realizadas com auxílio de um sistema de controle de versão (SVN), utilizando o ambiente gráfico de usuário TortoiseSVN. O repositório do projeto pode ser encontrado no link https://code.google.com/p/tcp-ecommerce/.