Projeto de Programação II
Loja de Conveniência

Objetivo

O objetivo deste trabalho é implementar um programa de cadastro e venda de produtos para a loja de conveniência *QLeveTudo*.

Um pouco sobre o funcionamento da *QLeveTudo*

QLeveTudo é uma loja de conveniência que comercializa diferentes tipos de produto. São atividades comuns da loja:

- *Cadastro* de um novo produto;
- *Remoção* de um produto;
- *Alteração* dos dados cadastrais de um produto;
- *Consulta* aos dados cadastrais de um determinado produto ou por um tipo de produto;
- *Consulta* de produtos de um determinado fornecedor;
- Todas as atividades relacionadas ao *cadastro* de fornecedores;
- *Venda* de produtos e *geração* de notas fiscais;
- *Controle* do estoque de cada produto (em unidades);

Produtos comercializados na *QLeveTudo*

Buscando atender a sua clientela diversificada, a *QLeveTudo* comercializa uma gama variada de produtos. A fim de manter um bom controle sobre os produtos à venda, cada produto é devidamente identificado por seu código de barras e apresenta uma descrição, um preço, uma quantidade em estoque e outras características que o descrevam, como mostrado a seguir.

- **Bebida:** sobre cada bebida comercializada é necessário identificar o seu teor alcoólico (um percentual do seu volume total) e quantidade de açúcar em miligramas.
- **Fruta:** sobre cada fruta é necessário controlar o número e a data de produção do lote.
- **Salgado:** sobre cada salgado é necessário controlar a quantidade de sódio em miligramas e se contém glúten e/ou lactose.
- **Doce:** sobre cada doce é necessário controlar a quantidade de açúcar em miligramas e se contém glúten e/ou lactose.
- **CD:** sobre cada CD é necessário identificar o seu estilo, artista e nome do álbum.

- **DVD:** sobre cada DVD é necessário armazenar o seu título, gênero e a duração total (em minutos).
- **Livro: s**obre cada livro é necessário identificar o título, autor, editora e ano de publicação.

Para os *produtos perecíveis* (bebida, fruta, salgados, doces) é necessário ainda registrar a data de validade do produto. Assim, aos produtos classificados como perecíveis é possível saber se o produto é *bom para consumo* ou não. Na venda de produtos, qualquer produto que não esteja bom para consumo é substituído por um novo produto, sendo este descartado.

Tarefas

A tarefa central deste Projeto de Programação é desenvolver um programa na linguagem de programação C++ para controlar o cadastro e a venda de produtos da *QLeveTudo*, segundo as características descritas anteriormente. **Todas** as atividades comuns ao dia-a-dia da loja**,** descritas acima, devem ser permitidas pelo seu programa. Para resolver o problema, você deverá implementar um modelo de classes que reflita a situação descrita.

Durante a implementação do seu programa, faça uso os conceitos e boas práticas de Programação Orientada a Objetos aprendidas na disciplina. Explore ao máximo os conceitos desse paradigma em C++ discutidos em sala de aula: classes, métodos, métodos de acesso (*getters*/*setters*), modificadores de acesso, sobrecarga de operadores, herança, polimorfismo, métodos virtuais, classes abstratas, *templates* de classe, entre outros. Você também deverá utilizar o Tipo Abstrato de Dados (TAD) *Lista* (simplesmente ou duplamente encadeada, ordenada ou não, a seu critério) implementado por você para manter a lista de produtos cadastrados, bem como possivelmente as listas de fornecedores e lista de produtos em uma nota fiscal.

Para permitir a venda de produtos e geração da nota fiscal, sobrescreva os operadores de adição + e subtração − de modo a permitir somar e subtrair o preço de dois produtos. Sobrescreva também o operador de igualdade == retornando `true` se dois produtos possuem o mesmo código de barras ou `false` em caso contrário.

Considere ainda que o programa deverá manter em arquivo os dados processados, ou seja, sempre que o programa iniciar, ele deverá carregar os dados (se existirem) presentes nos arquivos correspondentes. O programa deverá utilizar como entrada arquivos no formato CSV (*Comma-Separated Values*). Espera-se a utilização de pelo menos um arquivo com os dados dos produtos cadastrados e outro com os dados dos fornecedores da *QLeveTudo*. Seguindo a mesma lógica, antes de ser encerrado, o programa deverá salvar todo o conteúdo atualmente em memória para os arquivos em disco. Neste caso, por questões de simplicidade, pode-se proceder com a substituição completa do conteúdo dos arquivos já existentes.

Uma vez realizadas as tarefas de implementação, você deverá elaborar um relatório simples contendo minimamente (1) uma introdução, a fim de explicar o propósito do relatório e (2) detalhes de implementação, descrevendo como foi feita a sua implementação em termos de arquivos, classes, métodos, etc. e como o programa funciona de maneira geral.

Orientações gerais

Você deverá atentar para as seguintes observações gerais no desenvolvimento deste trabalho:

1) Apesar da completa compatibilidade entre as linguagens de programação C e C++, seu código fonte **não** deverá conter recursos da linguagem C nem ser resultante de mescla entre as duas linguagens, o que é uma má prática de programação. Dessa forma, deverão ser utilizados **estritamente** recursos da linguagem C++.

2) Durante a compilação do seu código fonte, você deverá habilitar a exibição de mensagens de aviso (*warnings*), pois elas podem dar indícios de que o programa potencialmente possui problemas em sua implementação que podem se manifestar durante a sua execução.

3) Aplique boas práticas de programação. Codifique o programa de maneira legível (com indentação de código fonte, nomes consistentes, etc.) e documente-o adequadamente na forma de comentários. Como anteriormente instruído, o código fonte deverá ser anotado para dar suporte à geração automática de documentação utilizando a ferramenta Doxygen (http://www.doxygen.org/).

4) Busque desenvolver o seu programa com qualidade, garantindo que ele funcione de forma correta e eficiente. Pense também nas possíveis entradas que poderão ser utilizadas para testar apropriadamente o seu programa e trate adequadamente possíveis entradas consideradas inválidas.

5) Lembre-se de aplicar boas práticas de modularização, em termos da implementação de diferentes funções/métodos e separação entre arquivos cabeçalho (.h) e corpo (`.cpp`).

6) A fim de auxiliar a compilação de seu projeto, você deverá **obrigatoriamente** construir um `Makefile` que faça uso da estrutura de diretórios apresentada anteriormente em aula.

7) Garanta o uso consistente de alocação dinâmica de memória. Para auxilia-lo nesta tarefa, você pode utilizar o Valgrind (http://valgrind.org/) como ferramenta de apoio para verificar se existem problemas de gerenciamento de memória.

8) A sua abstração sobre o contexto do trabalho faz parte da avaliação e será refletida no modelo de classes a ser implementado. Fique livre para adicionar funcionalidades e características que julgue necessárias ou interessantes.

Autoria e política de colaboração

Este trabalho deverá ser realizado individualmente ou em equipe de **no máximo** dois integrantes. O trabalho em cooperação entre estudantes da turma é estimulado, sendo admissível a discussão de ideias e estratégias. Contudo, tal interação não deve ser entendida como permissão para utilização de (parte de) código fonte de colegas, o que pode caracterizar situação de plágio. Trabalhos copiados em todo ou em parte de outros colegas ou da Internet serão sumariamente rejeitados e receberão nota zero.

Independentemente de o trabalho ser realizado individualmente ou em equipe, você deverá utilizar o sistema de controle de versões Git no desenvolvimento. Ao final, todos os arquivos de código fonte

do repositório Git local deverão estar unificados em um repositório remoto Git hospedado em algum serviço da Internet, a exemplo do GitHub, Bitbucket, Gitlab ou outro de sua preferência. A fim de garantir a boa manutenção de seu repositório, configure corretamente o arquivo `.gitignore` em seu repositório Git. A atividade de cada aluno deverá ser registrada através de *commits* sobre o repositório Git, que será examinado pelos professores durante a avaliação do trabalho.

Entrega

Você deverá submeter um único arquivo compactado no formato **`.zip`** contendo todos os códigos fonte resultantes da implementação deste exercício, sem erros de compilação e devidamente testados e documentados, **até o horário da aula do dia 6 de junho de 2017** através da opção *Tarefas* na Turma Virtual do SIGAA. Juntamente com os códigos fonte, o arquivo compactado deverá também conter o relatório escrito, preferencialmente em formato PDF. Por fim, você deverá ainda informar, no campo *Comentários* do formulário de submissão da tarefa, o endereço do repositório Git utilizado.

Avaliação

A avaliação deste trabalho será feita principalmente sobre os seguintes critérios: (1) utilização correta dos conteúdos vistos anteriormente e nas aulas presenciais da disciplina; (2) a corretude da execução do programa implementado, que deve apresentar saída em conformidade com a especificação e as entradas de dados fornecidas; (3) a aplicação correta de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte; (4) qualidade do relatório técnico produzido, e; (5) a utilização correta do repositório Git, no qual deverá estar registrado **todo** o histórico da implementação por meio de *commits*. O trabalho possuirá nota máxima de 4,00 (quatro) pontos para a 2ª unidade da disciplina, distribuídos de acordo com a seguinte composição:

Item avaliado	**Nota máxima**
Modularização adequada	0,25
Aplicação correta de conceitos de Programação Orientada a Objetos em C++ - Classes: atributos e métodos; construtores e destrutores; níveis de acesso e métodos de acesso	1,00
- Sobrecarga de operadores	0,50
- Herança, polimorfismo e classes abstratas	0,50
TADs genéricas com o uso de *templates* de classes	0,50
Uso adequado de ferramentas de compilação (com `Makefile`), depuração e verificação de memória	0,25
Uso correto de controle de versão com Git	0,25
Uso consistente de alocação dinâmica de memória	0,25
Qualidade do relatório escrito	0,50
Total	**4,00**

Por sua vez, o não cumprimento de algum dos critérios anteriormente especificados poderá resultar nos seguintes decréscimos, calculados sobre a nota total obtida até então:

Falta	**Decréscimo**
Programa apresenta erros de compilação, não executa ou apresenta saída incorreta	–70%
Falta de comentários no código fonte e/ou de documentação gerada com Doxygen	–10%
Implementação na linguagem C ou resultante de mistura entre as linguagens C e C++	–30%
Programa compila com mensagens de aviso (*warnings*)	–50%
Plagiarismo	–100%