Universidade de Brasília
Departamento de Ciência da Computação
Métodos de Programação 1/2017

Projeto de Disciplina

Datas de entrega:

Final : 29/06/17 ate as 23:55

Rede Social para Economia Compartilhada

Descrição:

Faça um programa em C ou C++

O programa deve implementar uma rede social que pode ser utilizada para economia compartilhada.

A rede social:

A rede social deve ser feita usando as historietas do **trabalho 3** e os casos de uso do **trabalho 4:**

Cada pessoa pode ter interesse em um produto ou serviço. Estes produtos ou serviços quando vendidos, alugados, etc podem virar transações. Ex. Venda de um computador se transforma em uma transação. Vários produtos e serviços são possíveis dentro de uma lista já cadastrada no sistema. O objetivo é oferecer os produtos para os outros usuários de acordo com alguns requisitos.

Um usuário pode solicitar uma transação. Ele pode escolher uma dentro de uma lista cadastrada. Por exemplo, ele escolheu pedir uma carona. O usuário pode escolher alguns requisitos sobre

com quem quer pegar carona. Estes requisitos podem ser qualquer dado do usuário inclusive já ter participado deste tipo de transação anteriormente e avaliação. Além disto, o usuário pode escolher qual o grau de separação ele quer o serviço. Por exemplo, aceita carona de usuários cadastrados como amigos, de amigos de amigos, usuários que já prestaram serviços para um amigo, qualquer um, etc.

Todos os usuários que atendem aos requisitos são notificados a respeito do pedido de carona. Os usuários que aceitaram a notificação de carona são mostrados para o usuário que pediu a carona ordenados por avaliação (se houver). Se o usuário que fez o pedido concordar, depois da transação concluída, todos os usuários envolvidos fazem avaliação uns dos outros inclusive escrevendo comentários a respeito da transação.

a.1) A interface dever permitir ao usuário:

Criar pessoa

Editar pessoa, todos os dados inclusive amizades

Excluir pessoa

Procurar por uma transação de acordo com os critérios desejados

Fazer uma transação e as avaliações

a.2) A interface para o administrador do sistema

Deve ser possível visualizar as informações das pessoas através de um grafo. Indicando cada pessoa e suas relações. Deve mostrar todos os vértices e arestas.

Deve ser possível cadastrar e descadastrar transações.

Deve ser possível saber todas as transações feitas e como elas foram avaliadas.

b) Leitor e escritor da rede social:

Deve ser possível salvar e recuperar todas as informações sobre a rede social em um arquivo.

c) Para fazer a implementação da rede social dever ser utilizada uma biblioteca de grafo como a que foi implementada no **trabalho 1**.

d) Devem ser feitos os diagramas de caso de uso, classes, sequência e atividade.

Requisitos:

1) Devem ser aplicados neste trabalho todos os conceitos vistos nos trabalhos anteriores.

a) Modularização (makefile, .h e .c).

b) utilize o padrão de codificação dado em:

https://google.github.io/styleguide/cppguide.html

O código deverá ser devidamente comentado facilitando o entendimento.

c) Testes utilizando o Gtest ou Catch. Como estes testes mostram que o programa segue a especificação. **O desenvolvimento deve ser feito orientado a testes**.

d) Devem ser feitas revisões no código conforme visto em sala de aula

e) Assertivas de entrada e de saída. Estas assertivas devem ser colocadas no código através do comando **assert** ou **ifs**. Comentários no código também devem especificar quais são estas assertivas

f) Para cada uma das funções adicionar comentários indicando qual são as interfaces explícita, implícita com a devida descrição, quais são os requisitos e as hipóteses de cada função. Além disto, as funções devem ter finalização adequada liberando memória, fechando arquivos, etc.

g) Faça a modelagem física das estruturas de dados. Use cabeças de estrutura de dados.

h) Assertivas de entrada e de saída como parte da especificação (comentários antes das funções).

i) Assertivas como comentários de argumentação.

j) Assertivas estruturais: elas definem a validade de uma coletânea de dados, ou estruturas de dados, e dos estados associados a estes dados.

k) Coloque nos comentários antes das funções quais são as assertivas do **contrato na especificação.** Diga o que deve ser esperado da função cliente em relação à entrada e o que deve ser garantido pela função servidora na saída.

2) Instrumente o código usando o programa gcov. Usando o gcov. (http://gcc.gnu.org/onlinedocs/gcc/Gcov.html). O makefile deve ser modificado de forma incluir as flags -ftest-coverage -fprofile-arcs. Depois de rodar o executável rode gcov nomearquivo e deverá ser gerado um arquivo gcov com anotação.

O gcov é utilizado para saber qual percentual do código é coberto pelos testes. Neste caso os teste devem cobrir pelo menos 80% do código por módulo.

Faça a análise estática do programa utilizando o *cppcheck*, corrigindo os erros apontados pela ferramenta

Utilize cppcheck --enable=warning
para verificar os avisos

3) Sugestão, utilizar o Valgrind (http://valgrind.org/). O Valgrind é muito útil para encontrar problemas relacionados a memória. O Valgrind **não** será utilizado na correção dos trabalhos mas pode tornar o desenvolvimento de depuração bem mais rápida.

4) Interface gráfica poderá ser uma destas:

a) **biblioteca ncurses.**
Ela pode ser instalada no terminal shell do Linux usando o comando

> sudo apt-get install libncurses5-dev

Uma vez instalada, a biblioteca ncurses ela pode ser compilada utilizando o GCC e a diretiva –lncurses.
Dependendo da instalação ncurses pode ser incluída por <ncurses/ncurses.h> ou <ncurses.h>.

Existem vários tutoriais disponíveis como :
http://www.tldp.org/HOWTO/NCURSES-Programming-HOWTO/index.html

b) **SDL**

https://www.libsdl.org/

Ela pode ser instalada com apt-get e deve ser devidamente configurada e referenciada no código.

c) **PlayCB**

Ela pode ser instalada a partir de:

http://pt-br.playcb.wikia.com/wiki/Wikia_PlayCB

d) **Qt**

Ela pode ser instalada a partir de:

https://www.qt.io/

3) Deverá ser entregue o histórico do desenvolvimento orientado a testes feitos através do github (https://github.com/)

Um tutorial inicial pode ser encontrado em:

https://guides.github.com/activities/hello-world/

4) O tempo em cada tarefa pode ser facilmente contabilizado utilizando aplicativos ou sites como:

https://toggl.com

5) O projeto pode ser gerenciado com sites como:

https://taiga.io/

Módulos

O programa deve ser dividido em módulos.

Observações:

Utilize os princípios de modularidade na criação dos módulos, definidos seus módulos de definição, implementação e organizando suas compilações e ligações (**links**) de forma adequada via um único **makefile**.

Na criação dos módulos, lembre-se de usar as diretivas de controle para evitar inclusões múltiplas e identificando também o módulo servidor.

Controle de qualidade das funcionalidades

Depois de pronto, o aluno deve criar um Makefile. Deve-se criar um *módulo controlador de teste* (disciplinado) usando o Gtest ou Catch para testar se as principais funcionalidades e restrições dos módulos, atendendo aos critérios da **Interface para linha de Comando**.

O desenvolvimento deverá ser orientado a testes.

O teste disciplinado deve seguir os seguintes passos:

1) Produzir um roteiro de teste
    a. definir o contexto (cenário) necessário e selecionar a massa de teste contendo a seqüência de ações e valores de teste com os respectivos resultados esperados e que foi criada segundo um critério de teste.
2) Antes de iniciar o teste: estabelecer o cenário do teste
3) Criar (ou modificar) um módulo controlador de teste, usando a ferramenta Gtest ou Catch para testar as principais funcionalidades de cada módulo.
4) Desenvolver o código que deve passar nos testes.
5) Ao testar: produzir um laudo em que todas as discrepâncias encontradas são registradas. Esse laudo pode ser uma saída da execução do Gtest ou Catch. Somente termine o teste antes de completar o roteiro, caso observe que não vale mais a pena continuar executando o roteiro, uma vez que o contexto para o resto está danificado.

6) Após a correção: repetir o teste a partir de **2** até o roteiro passar sem encontrar falhas.

Deve ser gerada uma documentação do código usando o programa DoxyGen (http://www.stack.nl/~dimitri/doxygen/): O programa inteiro terá de ser documento usando DoxyGen. Comentários que vão ficar na documentação devem ser do estilo Javadoc. A documentação deverá ser feita conforme visto em aula

Parte 2. Escrita

· Um arquivo **LEIAME.TXT** contendo a explicação de como utilizar o(s) programa(s).
· Tantos arquivos **RELATORIO-nome.TXT** quantos forem os membros do grupo. O tema **nome** deve identificar o membro do grupo ao qual se refere o relatório. Estes arquivos devem conter uma tabela de registro de trabalho organizada como a seguir:
Data | Horas Trabalhadas | Tipo Tarefa | Descrição da Tarefa Realizada

Na descrição da tarefa redija uma explicação breve sobre o que o componente do grupo fez. Esta descrição deve estar de acordo com o Tipo Tarefa. Cada Tipo Tarefa identifica uma natureza de atividade que deverá ser discriminada explicitamente, mesmo que, durante uma mesma sessão de trabalho tenham sido realizadas diversas tarefas. Os tipos de tarefa são:

- estudar aulas e laboratórios relacionados
- especificar os módulos
- especificar as funções
- revisar especificações
- projetar
- fazer diagramas
- revisar projetos
- codificar módulo
- Rodar os CPPCheck e retirar warnings
- revisar código do módulo
- redigir casos de teste
- revisar casos de teste
- realizar os testes
- instrumentar via gcov
- documentar com Doxygen

Observações:

- Dica: Preencha esta tabela de atividades ao longo do processo. NÃO DEIXE PARA ÚLTIMA HORA, POIS VOCÊ NÃO SE LEMBRARÁ DO QUE FEZ TAL DIA, TAL HORA. Com relatórios similares a esse você aprende a planejar o seu trabalho.

O trabalho pode ser feito em grupo de no máximo 3 alunos.

O programa deve ser feito para Linux, utilizado o GCC e GDB na mesma versão do Linf

Deve ser enviado um único arquivo compactado (.zip) contendo todos os arquivos necessários (.c .h makefile, estrutura de diretório, informação de como utilizar, etc). Deve constar também os documentos especificados. Deverá haver um arquivo leiame.txt com as informações sobre como o programa deverá ser compilado, como poderá ser executado e quais são os arquivos enviados e o que cada um contem.

Apenas um integrante do grupo deve enviar o trabalho. O nome do trabalho deve ser algo como:
MP_Jose_12345_Joao_54321_Maria_12345.zip
onde são os primeiros nomes e matriculas dos integrantes do grupo.

Cópias de trabalho terão nota zero.

Excelente trabalho!

Deve ser enviada pelo ead.unb.br até : 29/06/17 até as 23:55