Universidade Federal do Espírito Santo – Departamento de Informática
Estruturas de Dados I (INF09292)
1o Trabalho Prático

Período: 2015/2
Profa Patrícia Dockhorn Costa
Email: pdcosta@inf.ufes.br

Data de Entrega: 06/10/205 Trabalho em Dupla
Este trabalho tem como objetivo praticar o uso de tipos abstratos de dados e estruturas do tipo Lista.

Regras Importantes

- Não é tolerado plágio. Trabalhos copiados serão penalizados com zero.
- A data de entrega é inadiável. Para cada dia de atraso, é retirado um ponto da nota do trabalho.

Material a entregar

- Impresso: Documentação do trabalho, que deve conter:
  - Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa (em termos de módulos, arquivos, etc.).
  - Implementação: descrição da implementação do programa. Devem ser detalhadas as estruturas de dados utilizadas (de preferência **com diagramas ilustrativos**), o funcionamento das principais funções utilizadas incluindo pré e pós condições, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. **Modularize o seu programa usando a técnica de tipos abstratos de dados**, como discutido em sala de aula.
  - Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
  - Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso.
- Por email (ed20152@inf.ufes.br):
  - O assunto da mensagem deve ser ed201502:trab1:<nome1> <nome2>
  - Por exemplo: ed201502:trab1:<joaosilva> <mariasampaio>
  - Documentação do trabalho (em formato PDF).
  - Todos os arquivos .c e .h criados (exigido código muito bem documentado!).
  - O makefile.

Publish/Subscribe

O uso de sistemas distribuídos tem crescido a cada dia. Grande parte desse crescimento deve-se à popularização dos dispositivos móveis, principalmente celulares e smartphones. Todos os dias, milhões de novos celulares e smartphones são inseridos no mercado, dispositivos que possuem os mais diversos tipos de softwares e hardwares. Apesar disso, esses dispositivos estão cada vez mais acoplados e conectados. Para atender os milhões de possíveis clientes espalhados por todo o mundo, tem-se buscado criar sistemas desacoplados e flexíveis. Por mais diferente que seja a tecnologia usada em cada dispositivo ou sistema, existe uma busca cada vez maior na integração entre os mesmos.

Essa demanda vem despertando o uso do paradigma Publish/Subscribe. Este paradigma permite que dispositivos que não se conhecem se comuniquem sem a necessidade de estarem conectados ou ligados num mesmo instante.

No paradigma Publish/Subscribe, tipicamente, existem duas classes de entidades: uma denominada Publisher (produtor da informação), que publica um evento com informações sobre um determinado conceito ou assunto (chamado *topico*); e uma denominada Subscriber (consumidor da informação), que subscreve um evento informando o seu interesse em receber notificações sobre um determinado tópico.

Existe ainda um serviço responsável por prover o desacoplamento entre os dispositivos. O serviço fica encarregado de receber as informações, tanto dos Publishers quando dos Subscribers, analisá-las e identificar quais Subscribers devem ser notificados. Uma vez identificados os Subscribers que devem ser notificados, o serviço realiza a entrega dessas notificações. Quando uma informação de interesse do Subscriber é encontrada, dizemos que ocorreu o *Matching* entre as informações. Além disso, é possível que o serviço possua vários elementos intermediários, denominados *Brokers*, de forma a criar uma rede interna. A figura abaixo ilustra uma possível configuração de um sistema Publish/Subscribe.

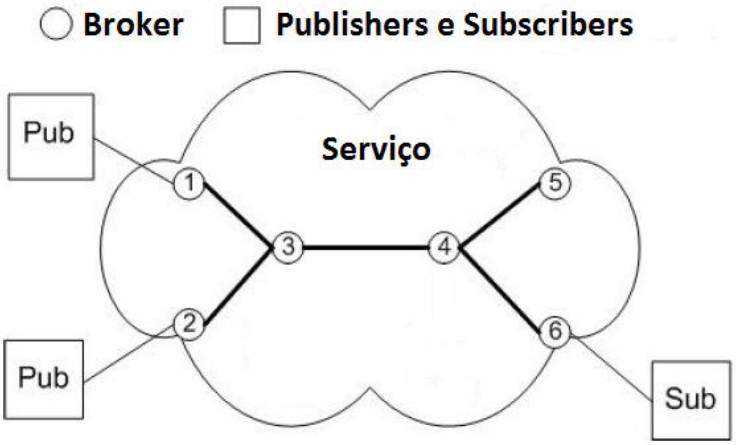
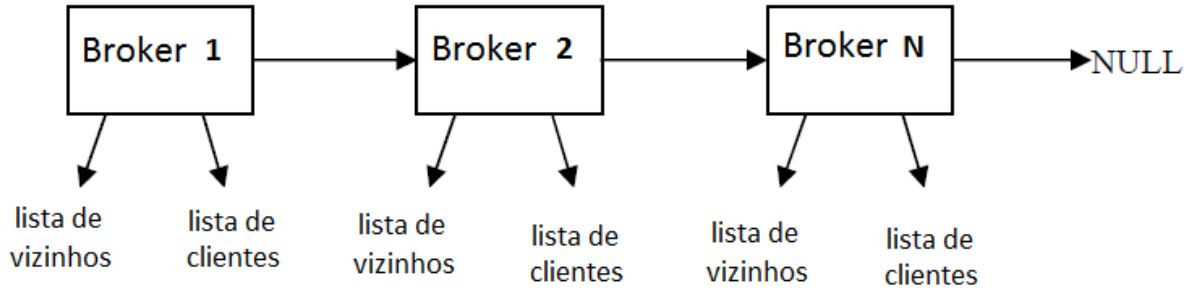
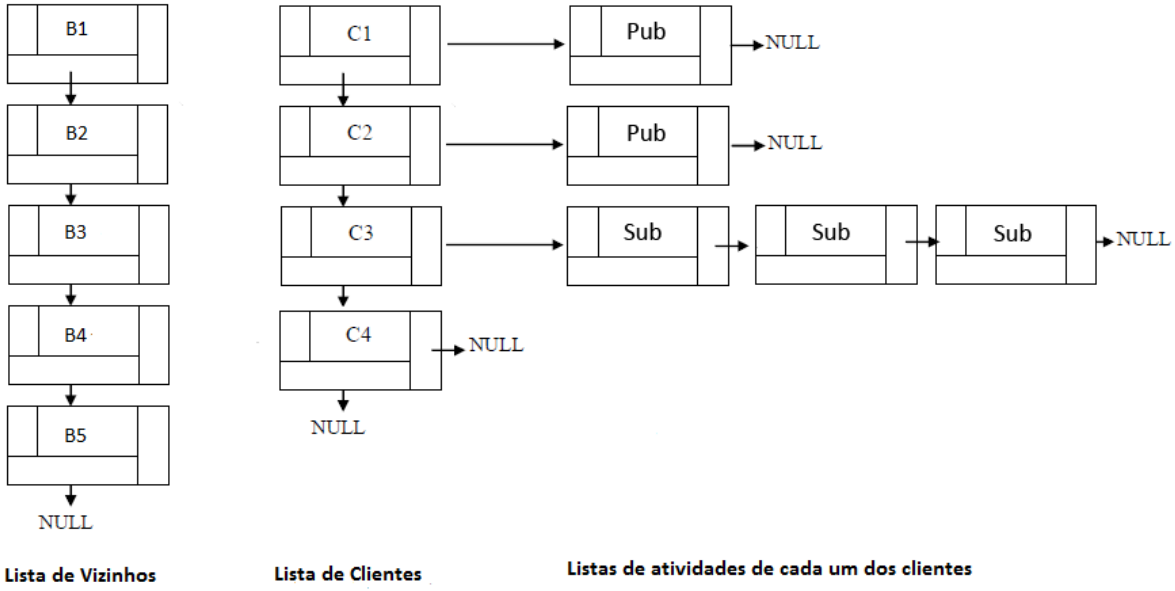


Figura 1. Possível configuração de um sistema Publish/Subscriber. Na imagem é possivel ver uma rede formada por 6 Brokers, com 3 Clientes, sendo dois Publishers e um Subscriber.

Este sistema Publish/Subscribe, de forma simplificada, pode ser implementado com um conjunto de listas encadeadas: pode-se implementar uma lista de Brokers, na qual cada Broker possui uma lista de vizinhos (ou seja, os Brokers conectados diretamente a ele) e uma lista de clientes (ou seja, os Publishers e Subscribers conectados diretamente a ele), como mostrado na figura a seguir.



Cada Cliente pode manter uma lista de suas publicações e subscrições. A figura a seguir sugere, de maneira geral, a organização das listas de vizinhos, clientes, e de suas publicações e subscrições.



Como pode ser visto na Figura, este Broker possui 5 vizinhos e 5 clientes, sendo que os clientes C1 e C2 realizaram uma publicação cada. O cliente C3 realizou 3 subscrições. Por sua vez, o cliente C4 não realizou atividades.

Neste trabalho, você deverá implementar um sistema Publish/Subscribe simplificado. Faz parte do trabalho projetar os Tipos Abstratos de Dados necessários, bem como implementá-los. Como padrão, insira os elementos ao final das listas.

Considerações Importantes

Cada cliente, necessariamente, sempre deve estar conectado a um Broker, e somente um;

Um Broker pode ter vários clientes e vários vizinhos (um broker não pode ser seu vizinho);

Um cliente pode ser tanto consumidor, produtor, mas não os dois simultaneamente. Seu papel é definido em sua inicialização e deve ser mantido ao longo da execução;

Clientes, Subscrições e Brokers podem ser excluídos. Publicações não podem ser excluídas por meio de comandos, assim como ocorre, por exemplo, com as subscrições por meio do comando **EXCLUISUBSCRIÇÃO**. Porém tanto as subscrições quanto as publicações devem ser excluídas assim que o cliente que as criou for removido da rede;

Uma publicação deve conter, obrigatoriamente, um identificador único, um tópico e uma mensagem;

Uma subscrição é constituída por um identificador único e um tópico;

Não podem existir duas ações como mesmo ID, ou seja, não podem existir duas publicações com mesmo ID, duas subscrições com mesmo ID e nem mesmo uma publicação ou subscrição com mesmo ID.

Para simplificar, considere que tanto as mensagens quanto os tópicos são compostos por apenas uma palavra, não sendo necessário se preocupar com nomes compostos.

Detecção do Matching

O matching ocorre quando uma publicação contém um dado de interesse de algum Subscriber, ou seja, publicação e subscrição possuem o mesmo tópico.

Uma mesma publicação pode gerar mais de um matching, um para cada Subscriber que possui interesse no respectivo tópico.

Ao publicar um dado, o Publisher deve especificar um tópico relacionado ao dado. Da mesma forma, Subscribers devem manifestar seus interesses em algum tópico. Se um cliente C1 publica uma mensagem com tópico igual a “futebol” e mensagem igual “Jogo às 4 horas”, deve ocorrer o matching entre todos os subscribers que realizaram alguma subscrição no tópico “futebol”, sendo gerado um matching para cada um destes Subscribers.

É importante observar que um Subscriber NÃO pode gerar uma subscrição para um tópico que já encontra-se inscrito. Um mesmo Publisher pode publicar várias mensagens no mesmo tópico.

A ordem das publicações e das subscrições não interfere na verificação do matching. Caso um cliente C1 realize uma publicação P1, uma publicação P2 e uma subscrição S1, respectivamente, o resultado

esperado para o matching não será alterado caso um mesmo cliente C1 realize uma Publicação P2, uma subscrição S1 e uma Publicação P1, respectivamente.

O programa testador deverá ser capaz de ler as instruções do arquivo texto de entrada e realizar as devidas operações no Sistema Publish/Subscribe. O seu programa testador deverá ler os dados de entrada a partir de um arquivo, cujo nome é passado como parâmetro na linha de comando (faz parte do trabalho descobrir como manipular arquivos e strings em C). Exemplo de execução do programa a partir da linha de comando: simulador entrada.txt

O arquivo de entrada é basicamente uma lista de comandos (um por linha) em formato texto. O último comando é a palavra FIM, que indica o final do arquivo. O formato a ser usado é exemplificado abaixo:

Exemplo de arquivo de entrada

CRIABROKER brokerA
CRIABROKER brokerB
CRIABROKER brokerC
CRIABROKER brokerD
CRIABROKER brokerE
CRIABROKER brokerF
CRIABROKER brokerG
CRIACLIENTE cliente1 P brokerA
CRIACLIENTE cliente2 S brokerA
CRIACLIENTE cliente3 S brokerB
CRIACLIENTE cliente4 P brokerC
CRIACLIENTE cliente5 S brokerC
CRIACLIENTE cliente6 P brokerD
CRIACLIENTE cliente7 S brokerG
INSEREVIZINHO brokerA brokerB
INSEREVIZINHO brokerB brokerC
INSEREVIZINHO brokerD brokerE
INSEREVIZINHO brokerE brokerD
INSEREVIZINHO brokerE brokerF
INSEREVIZINHO brokerF brokerD
INSEREVIZINHO brokerF brokerG
PUBLICA 1 cliente1 esporte Flamengo
SUBSCREVE 2 cliente3 esporte
SUBSCREVE 3 cliente5 esporte
SUBSCREVE 4 cliente7 esporte
SUBSCREVE 5 cliente 7 informatica
PUBLICA 6 cliente6 informatica prova
PUBLICA 7 cliente4 esporte Vasco
IMPRIMEREDE

VERIFICAMATCHING
CANCELAVIZINHO brokerA brokerB
INSEREVIZINHO broker A brokerC
EXCLUISUBSCRICAO 2
EXCLUIBROKER brokerC
EXCLUICLIENTE cliente1
IMPRIMEREDE
VERIFICAMATCHING
FIM

Especificação dos Comandos

CRIABROKER <nome_broker>: cria um broker com o nome especificado;

CRIACLIENTE <nome_cliente> <tipo_cliente> <nome_broker>: cria um cliente com o nome especificado, informando qual o seu tipo (P = Publisher, S = Subscriber) e insere o cliente na lista de clientes do broker especificado;

INSEREVIZINHO <broker1> <broker2>: Insere o Broker 2 na lista de vizinhos do Broker 1;

IMPRIMEREDE: Lista todos os Brokers, os clientes de cada broker e suas atividades (publicações e subscrições);

VERIFICAMATCHING: Lista todos os subscribers (mesmo os que não possuem matching), um por vez, e todas as duplas de subscrição e publicação, uma dupla por linha, onde ocorre o matching. As duplas devem apresentar a letra S concatenada com o identificador da subscrição, seguido de espaço e a letra P concatenada com o identificado da publicação. Logo após deve constar o tópico e mensagem da publicação, necessariamente nessa ordem, separados por espaço simples;

PUBLICA <id_publicação> <nome_cliente> <tópico> <mensagem>: recebe um identificador único para a publicação, o nome cliente que deseja publicar, o tópico para o qual deseja publicar e uma mensagem que o Subscriber deve receber caso manifeste seu interesse no mesmo tópico;

SUBSCREVE <id_subscrição> <nome_cliente> <tópico>: recebe um identificador único para a subscrição, o nome cliente que deseja subscrever e o tópico no qual deseja manifestar seu interesse;

EXCLUISUBSCRICAO <id_subscrição>: exclui a subscrição com ID indicado;

EXCLUIBROKER <nome_broker>: exclui o broker com nome indicado. Se um broker é excluído, todos os seus clientes devem ser excluídos automaticamente;

EXCLUICLIENTE <nome_cliente>: exclui o cliente com nome indicado;

Dica

Ao percorrer as listas de vizinhos, cuidado para não entrar em um loop infinito. Observe que na entrada usada como exemplo, o brokerD é vizinho do brokerE, que por sua vez é vizinho do brokerF. Porém, o broker F é também vizinho do brokerD, criando um ciclo infinito;

Não esqueça de liberar toda memória alocada ao término da execução do programa!!!

Considerando o arquivo de entrada acima, espera-se o seguinte no arquivo de saída:

Arquivo de saída para o arquivo entrada.txt

```
IMPRIMEREDE

BROKER brokerA
  VIZINHO brokerB
  CLIENTE P cliente1
    EVENTO P1 esporte Flamengo
  CLIENTE S cliente2

BROKER brokerB
  VIZINHO brokerA
  VIZINHO brokerC
  CLIENTE S cliente3
    EVENTO S2 esporte

BROKER brokerC
  VIZINHO brokerB
  CLIENTE P cliente4
    EVENTO P7 esporte Vasco
  CLIENTE S cliente5
    EVENTO S3 esporte

BROKER brokerD
  VIZINHO brokerE
  VIZINHO brokerF
  CLIENTE P cliente6
    EVENTO P6 informatica prova

BROKER broker E
  VIZINHO brokerD
  VIZINHO brokerF

BROKER brokerF
  VIZINHO brokerE
```

```
  VIZINHO brokerD
  VIZINHO brokerG

BROKER brokerG
  VIZINHO brokerF
  CLIENTE S cliente7
    EVENTO S4 esporte
    EVENTO S5 informatica

VERIFICACAO MATCHING

CLIENTE cliente2

CLIENTE cliente3
  S2 P1 esporte Flamengo
  S2 P7 esporte Vasco

CLIENTE cliente5
    S3 P7 esporte Vasco
    S3 P1 esporte Flamengo

CLIENTE cliente7
 S5 P6 informatica prova

IMPRIMEREDE

BROKER brokerA
  CLIENTE S cliente2

BROKER brokerB
 CLIENTE S cliente3

BROKER brokerD
  VIZINHO brokerE
  VIZINHO brokerF
  CLIENTE P cliente6
    EVENTO P6 informatica prova

BROKER broker E
  VIZINHO brokerD
  VIZINHO brokerF
```

```
BROKER brokerF
  VIZINHO brokerE
  VIZINHO brokerD
  VIZINHO brokerG

BROKER brokerG
  VIZINHO brokerF
  CLIENTE S cliente7
    EVENTO S4 esporte
    EVENTO S5 informatica

VERIFICACAO MATCHING

CLIENTE cliente2

CLIENTE cliente3

CLIENTE cliente7
 S5 P6 informatica prova
```