T3 - Trabalho 3 de CI1001 e CI067
Introdução aos tipos abstratos de dados
Departamento de Informática/UFPR
Luis Bona e Marcos Castilho

1 Requisitos obrigatórios

- Use boas práticas de programação, como indentação, bons nomes para variáveis, comentários no código, bibliotecas, ... Um trabalho que não tenha sido implementado com boas práticas vale zero.

- Quaisquer dúvidas com relação a este enunciado devem ser solicionadas via sistema moodle sendo que os dois professores devem receber o questionamento. Na dúvida não tomem decisões sobre a especificação, perguntem!

2 O problema

Uma simulação é uma experimentação com um modelo que imita de forma aproximada a realidade e sua evolução ao longo do tempo. A simulação é usada em muitos contextos: na modelagem científica de sistemas naturais; como técnica de ensino, oferecendo para os alunos um ambiente onde é possível experimentar e errar sem as consequências do ambiente real; e também em um grande número de jogos eletrônicos recreativos.

A simulação de eventos discretos (SED) modela a operação de um sistema como uma sequência de eventos discretos no tempo. O modelo possui uma estrutura estática e uma estrutura dinâmica. A estrutura estática específica os possíveis estados do modelo e é geralmente descrita como uma coleção de entidades e seus atributos. O estado do sistema é a coleção de dados que descreve o sistema em um determinado momento.

A estrutura dinâmica específica como o estado muda ao longo do tempo e geralmente é definida em termos de eventos, que ocorrem em um determinado momento e mudam o estado do sistema. Entre eventos consecutivos o estado do sistema não sofre mudança alguma e a simulação pode saltar diretamente do instante de ocorrência de um evento para o próximo. Uma atividade é o estado de um objeto durante um intervalo.

Um simulador deve manter um relógio lógico global e uma lista ordenada por tempo dos eventos que vão ocorrer no futuro (LEF - Lista de Eventos Futuros) e um relógio lógico global. O relógio costuma ser um número inteiro

que representa a quantidade de tempo do sistema real. Dependendo do modelo da simulação, cada unidade do relógio virtual pode representar pequenas frações de tempo (por exemplo, em uma simulação de partículas) ou muito grande (por exemplo, simulando a evolução do clima em uma determinada era.

O ciclo básico de simulação retira o primeiro evento da lista de eventos, atualiza o estado do sistema, agenda os novos eventos na lista e continua o ciclo.

Nós vamos implementar um simulador de vida real simplificado. Os habitantes de nosso mundo virtual são muitos sociáveis e suas vidas são dedicadas à transitar em diferentes locais e disseminar informações, aqui designadas como rumores.

O restante desse texto especifica as entidades e atributos do sistema; os eventos que alteram o estado do nosso mundo virtual; o esqueleto geral da simulação e as mensagens de saída esperadas.

3 Entidades e atributos

Essa seção apresenta as entidades e atributos do nosso mundo virtual.

Pessoa: Representa as pessoas da simulação e os atributos das pessoas determinam seu comportamento nos locais. Os atributos das pessoas são:

- **id:** Inteiro que identifica de maneira única uma pessoa;
- **extroversão:** Inteiro entre 0 e 100 que indica o grau de extroversão da pessoa; em nosso modelo pessoas mais extrovertidas (números mais altos) são mais propensas à disseminar e a ouvir rumores;
- **paciência:** Inteiro entre 0 e 100 que indica quanto paciente uma pessoa é. Em nosso modelo isso afeta as decisões de permanência em locais e filas;
- **idade:** Inteiro entre 18 e 100 indicando a idade em anos de uma pessoa. Em nossa simulação a idade afeta o tempo de deslocamento entre lugares;
- **conjunto de rumores conhecidos (CRC):** Rumores conhecidos por uma determinada pessoa e passíveis de disseminação.

Local: Representam os locais frequentados pelas pessoas; os locais tem uma localização em um plano cartesiano, podem estar cheios e possuem uma fila de espera. Os atributos dos locais são:

id: Inteiro que identifica o local;

lotação máxima: Número de pessoas máximo que estão em um lugar;

pessoas no lugar: Conjunto de identificadores das pessoas que estão atualmente no lugar;

fila: Fila onde as pessoas esperam para poderem entrar caso o mesmo esteja lotado;

localização: Par de inteiros (x, y) indicando uma coordenada em um plano cartesiano. Vamos considerar que o mapa de nossa cidade é representado por um plano cartesiano de tamanho cada unidade representa 1 metro.

Rumor: Não desejamos ser contaminados pelos rumores, só queremos saber como eles se propagam. Um rumor é representado simplesmente por um número inteiro.

Mundo: Nosso mundo é definido por essas entidades e algumas informações gerais.

TempoAtual: Inteiro positivo indicando o tempo atual da simulação. Cada unidade representa 15 minutos de tempo real;

Pessoas: Todas as pessoas;

Locais: Todos os locais;

Rumores: Conjunto de todos os rumores que podem ser conhecidos;

NPessoas: Número total de pessoas no mundo;

NLocais: Número total de locais no mundo;

NTamanhoMundo: Coordenadas máximas do plano cartesiano dos locais.

4 Eventos

Nosso relógio (`TEMPO_ATUAL`) inicia em 0 e cada unidade representa 15 minutos da vida real. Os eventos são registrados em uma Lista de Eventos Futuros (`LEF`) ordenados pelo seu tempo lógico. Cada vez que um evento é lido o `TEMPO_ATUAL` é atualizado para o tempo do evento.

Em nossa simulação temos 4 tipos de eventos diferentes: `CHEGADA`, `SAIDA`, `DISSEMINACAO` e `FIM`. Cada evento possui diferentes atributos que permitem os processamentos na simulação. Portanto, a `LEF` deve suportar diferentes tipos de eventos, o que é um dos desafios dessa implementação.

Nas próximas subseções especificam os atributos de cada um destes eventos e qual seus comportamento esperado.

4.1 CHEGADA

Representa uma pessoa (`ID_PESSOA`) entrando em um local (`ID_LOCAL`). Ao chegar uma pessoa deve saber se o local está ou não lotado, decidir esperar na fila ou sair imediatamente. Caso a pessoa entre no local esta deve decidir seu tempo de permanência e escalonar sua saída. Finalmente, durante sua permanência ela dissemina um conjunto de rumores.

Parte do comportamento de nossa população tem componente aleatório. Nas especificações deste trabalho vamos utilizar a notação ALEAT(x,y) para representar números inteiros gerados aleatoriamente entre x e y [1].

Abaixo uma especificação em alto nível de como a simulação deve tratar o evento de CHEGADA.

```
Se o ID_LOCAL está lotado a ID_PESSOA deve decidir entre ficar  na fila ou ir embora:

        Se (PESSOA[ID_PESSOA].PACIENCIA/4  - tamanho_fila(ID_LOCAL) > 0
            Adiciona ID_PESSOA em ID_LOCAL.fila
        senão
            Cria um evento de SAIDA  de ID_PESSOA  e insere na LEF com TEMPO_ATUAL+0

Se ID_LOCAL não está  lotado,  calcular o Tempo de Permanência no Local (TPL):
    TPL = MAX(1,PESSOA[ID_PESSOA].paciencia/10+ALEAT(-2,6))

    *obs: o MAX é para garantir que o tempo mínimo seja 1
          TPL já é dado em unidades de tempo de simulação (1 == 15 min)

Criar  um evento de SAÍDA para ID_PESSOA e inserir na LEF em TEMPO_ATUAL +  TPL

Criar o evento de DISSEMINACAO:
    - criar um subconjunto aleatório de rumores de  ID_PESSOA.CRC de tamanho NRD:
           NRD = PESSOA[ID_PESSOA].extroversao/10
           NRD  <= cardinalidade de CRC e  NRD <=  TPL
    - inserir o evento na LEF e TEMPO_ATUAL + ALEAT(0,TPL).
```

4.2 SAÍDA

Representa a saída de uma pessoal (`ID_PESSOA`) de um local (`ID_LOCAL`). A saída de uma pessoa pode gerar uma vaga no local tirando uma pessoa que estava na fila. Durante a saída é determinado um novo local de destino para a pessoa (`ID_PESSOA`) como o tempo necessário para sua chegada.

[1] Em C você pode usar simplesmente a função rand(). Em uma única vez do seu código você deve chamar a função srand(semente). Se você usar a mesma semente durante toda sua simulação e em toda execução o comportamento será sempre o mesmo. O que pode ser útil para efeitos de depuração.

Abaixo uma especificação em alto nível de como a simulação deve tratar o evento de SAÍDA.

```
Se ID_LOCAL.fila não estiver vazia (existe um ID_PESSOA_FILA na fila)
   - criar evento de  CHEGADA para ID_PESSOA_FILA em ID_LOCAL
   - inserir esse evento em TEMPO_ATUAL+0 e  INICIO da LEF
   obs: - inserir no início é para evitar que outro evento de CHEGADA na
          LEF possa "furar" a fila.
        - nem toda saída gera lugar na fila, pois pode ser alguma pessoa impaciente
          saindo do local sem nunca ter entrado (não quis ficar na fila).

Criar evento de CHEGADA para ID_PESSOA
    -  ID_LOCAL_DEST deve ser escolhido aleatoriamente entre os locais
        obs: podendo inclusive decidir entrar novamente no mesmo local)
    - inserir o evento  na  LEF considerando  em TEMPO_ATUAL+TDL/15
    - TDL(Tempo de Deslocamento entre Locais):
        TDL = Distância (D) / Velocidade (V)

        onde:
            V = 100-MAX(0,pessoas[ID_PESSOA].idade-40);
            D = Distância cartesiana entre ID_LOCAL e ID_LOCAL_DESTINO
```

4.3 DISSEMINIAÇÃO:

Representa a disseminiação de conjunto de rumores (`CJ_RUMORES`) vindos da `ID_PESSOA_ORIGEM` para as pessoas em um lugar `ID_LOCAL`.

O evento dissemina os rumores de `CJ_RUMOR` para as pessoas que estão em um determinado local (as pessoas da fila não são afetadas). Uma pessoa pode ou não incorporar o rumor dado seu grau de extroversão.

```
Para cada pessoa ID_PESSOA_ESCUTA do conjunto ID_LOCAL.Publico eqto CJ_RUMOR != vazio
    Se ( ALEAT(0,100) < ID_PESSOA_ESCUTA.extroversao )
        ID_RUMOR = retira_evento (CJ_RUMOR)
        insere(ID_PESSOA_ESCUTA.rumores)
    senao
        ignora rumor
```

4.4 FIM

Esse evento determina o final da simulação.

5 Esqueleto da simulação

No início da simulação devemos criar nosso mundo virtual com todas suas entidades e um conjunto inicial de eventos na `LEF`

```
    O tempo inicial é 0
    O tamanho do nosso mundo (N_TAMANHO_MUNDO) é 20000
    O tamanho do conjunto de (N_RUMORES_MUNDO) rumores é 30
    A nossa população (N_PESSOAS) é 100
    O número de locais (N_LOCAIS) é 8
    O final da simulação (FIM_DO_MUNDO) é no instante 34944

Criar o conjunto de rumores com ID sequencia de 0 até N_RUMORES_MUNDO-1

Criar um vetor de N_PESSOAS,  cada PESSOA é inicializada com:
    id = número sequencial 0-N_PESSOAS-1
    extroversao = ALEAT(0,100)
    paciencia    = ALEAT(0,100)
    idade = ALEAT (18,100)
    rumores = subconjunto com tamanho ALEAT(1,5) de CJ_RUMORES_MUNDO

Criar um vetor de N_LOCAIS,  cada LOCAL é inicializada com:
    id = número sequencial 0-N_LOCAIS-1
    lotacao_max = ALEAT (5,30)
    localizacao = ALEAT(0,N_TAMANHO_MUNDO-1), ALEAT(0,N_TAMANHO_MUNDO-1)
    publico = conjunto vazio

Criar para cada  cada PESSOA da população:
    EVENTO de CHEGADA  em ID_LUGAR_DEST=ALEAT(0,N_LOCAIS-1)
    inserir na LEF com tempo: ALEAT(0,96*7)

Criar e  inserir o evento de FIM no tempo FIM_DO_MUNDO
```

Fiat Lux! Nosso mundo foi criado e podemos começar a processar os eventos. Os eventos vão atualizar o estado do nosso mundo e também criar novos eventos. Esse ciclo termina com o evento de final (em determinadas culturas chamado de apocalipse). O laço principal é relativamente simples, e fica elegante usando o comando `switch/case`.

```
ENQUANTO retirar EVENTO da LEF
    Atualizar TEMPO_ATUAL com EVENTO.tempo
    switch(EVENTO->tipo)
    {
        case CHEGADA:
            tratar evento
            break;
        case SAIDA:
            tratar evento
            break;
        case DISSEMINACAO:
            tratar evento
            break
        case FIM
            finalizar simulacao
    }
```

6 Mensagens da simulação

Nenhuma divindade virtual se sente realizada sem conhecimento do que as pessoas de seu mundo estão fazendo. Por isso nossa simulação deve imprimir algumas informações durante o processamento dos eventos. Essa mensagens também podem ser utéis no processo de depuração do código.

6.1 Mensagens para o evento de CHEGADA

Exemplos das mensagens no evento de CHEGADA quando o local não está lotado:

```
  8:CHEGA Pessoa   44 Local  1 ( 0/19), ENTRA
 16:CHEGA Pessoa   11 Local  5 ( 0/26), ENTRA
 23:CHEGA Pessoa   44 Local  2 ( 0/21), ENTRA
```

A primeira mensagem nos diz que no instante 8 ocorreu um evento de chegada da pessoa 44 no local 1; no local 1 estavam 0 pessoas; e sua lotação máxima de 19. Portanto a pessoa 44 "ENTRA"no local.

Exemplos de mensagem no evento de CHEGADA quando o local está lotado

```
232:CHEGA Pessoa   30 Local  3 ( 7/ 7), FILA  1
233:CHEGA Pessoa   74 Local  3 ( 7/ 7), FILA  2
401:CHEGA Pessoa   82 Local  4 ( 5/ 5), DESISTE
```

Nas mensagens acima vemos que no instante 232 que pessoa 30 chegou ao local 3, entretanto esse estava lotado, a pessoa decidiu entrar na fila do local 3 e foi colada na posição 1; Na próxima mensagem no instante 233 temos uma situação semelhante, a pessoa 74 também chega em um local lotado e entra na fila na posição 2. No instante 401 vemos um caso em que a pessoa 82 não teve paciência para esperar e por isso DESISTE de entrar no local.

6.2 Mensagens para o evento de SAÍDA

```
 22:SAIDA Pessoa   44 Local  1 ( 1/19)
 19:SAIDA Pessoa   11 Local  5 ( 1/26)
```

As mensagens acima são análogas às de CHEGADA, além disso é possível ver a correspondência entre o evento de SAIDA no instante 22 com a ENTRADA no instante 8 (da mesma forma entre 19 e 16). Notem que assim é possível saber o TDL dessas pessoas.

```
234:SAIDA Pessoa   35 Local  3 ( 7/ 7), REMOVE FILA Pessoa 30
```

No exemplo acima, vemos que a saída da pessoa 35 do local 3 abriu um espaço para a entrada da pessoa 30. Nas mensagens anteriores é possível observar que no instante 232 de fato a pessoa 30 estava em primeiro lugar na fila.

6.3 Saída para RUMOR

```
12:RUMOR Pessoa   44 Local  1
```

Acima exemplo de saída que mostra a disseminação no instante 12 pela Pessoa 44 no Local 1. A ausência de outras informações significa que a disseminação foi ineficaz pois nenhuma pessoa ficou sabendo de algum rumor novo.

```
84:RUMOR Pessoa    8 Local  5 (P11:R19) (P31:R13) (P93:R5)
```

Acima mostra que a pessoa 8 conseguiu atingir 3 pessoas com seus rumores no Local 5. A pessoa 11 incorporou o rumor 19; a pessoa 31 incorporou o rumor 13; e finalmente a pessoa 93 incorporou o rumor 5. Essa mensagem somente deve ser apresentada quando realmente uma pessoa incorporou um novo rumor, ou seja seu CRC mudou.

6.4 Formatação

Essa saída vai ser útil para você validar o comportamento da sua simulação. Uma das maneiras de fazer isso será usando comando de bash para filtar essa saída. Por exemplo:

```
631:CHEGA Pessoa    1 Local  5 ( 7/26), ENTRA
632:RUMOR Pessoa    1 Local  5 (P42:R28)
639:SAIDA Pessoa    1 Local  5 (10/26)
649:CHEGA Pessoa    1 Local  6 ( 6/25), ENTRA
657:RUMOR Pessoa    1 Local  6 (P37:R29)
660:SAIDA Pessoa    1 Local  6 ( 7/25)
666:CHEGA Pessoa    1 Local  3 ( 7/ 7), FILA  3
670:SAIDA Pessoa   44 Local  3 ( 7/ 7), REMOVE FILA Pessoa    1
670:CHEGA Pessoa    1 Local  3 ( 6/ 7), ENTRA
673:RUMOR Pessoa    1 Local  3 (P27:R19)
684:SAIDA Pessoa    1 Local  3 ( 7/ 7), REMOVE FILA Pessoa   67
```

O comportamento parece consistente, a pessoa entra e sai de lugares espalhando rumores. No instante 666 acaba chegando no local 3 que está lotado; no instante 670 a pessoa 44 sai do local 3 tirando a pessoa 1 da fila e como consequência no instante 670 a pessoa 1 entra no local 3. Finalmente a saída da pessoa 1 do local 3 vai remover a pessoa 67 da fila no instante 684.

A sugestão é que a *string* de formatação (`printf`) coloque espaçamento em branco nos inteiros alinhando a formatação e facilitando a visualização. Especificamente:

```
tempo  %6d
id de pessoas %4d
informações de lotação dos locais %2d
tamanho da fila %2d
os pares de rumores incorporados são apenas P%d/R%d
```

Abaixo exemplo de um trecho completo da saída:

```
232:CHEGA Pessoa   92 Local  5 ( 2/26), ENTRA
232:CHEGA Pessoa   93 Local  6 ( 4/25), ENTRA
232:CHEGA Pessoa   30 Local  3 ( 7/ 7), FILA  1
233:SAIDA Pessoa   13 Local  6 ( 5/25)
233:RUMOR Pessoa   66 Local  4
233:CHEGA Pessoa   13 Local  6 ( 4/25), ENTRA
233:CHEGA Pessoa    4 Local  4 ( 2/ 5), ENTRA
233:RUMOR Pessoa   13 Local  6 (P76:R25)
233:RUMOR Pessoa   43 Local  2
233:SAIDA Pessoa   90 Local  4 ( 3/ 5)
233:CHEGA Pessoa   74 Local  3 ( 7/ 7), FILA  2
234:CHEGA Pessoa   90 Local  1 ( 0/19), ENTRA
234:RUMOR Pessoa   97 Local  6 (P13:R21)
234:RUMOR Pessoa   90 Local  1
234:SAIDA Pessoa   18 Local  6 ( 5/25)
234:SAIDA Pessoa   35 Local  3 ( 7/ 7), REMOVE FILA Pessoa   30
234:CHEGA Pessoa   30 Local  3 ( 6/ 7), ENTRA
235:RUMOR Pessoa    4 Local  4
235:SAIDA Pessoa   13 Local  6 ( 4/25)
235:RUMOR Pessoa   92 Local  5
235:CHEGA Pessoa   13 Local  6 ( 3/25), ENTRA
235:SAIDA Pessoa   66 Local  4 ( 2/ 5)
235:SAIDA Pessoa   34 Local  3 ( 7/ 7), REMOVE FILA Pessoa   74
235:CHEGA Pessoa   74 Local  3 ( 6/ 7), ENTRA
235:RUMOR Pessoa   15 Local  0 (P29:R6)
235:RUMOR Pessoa   74 Local  3 (P44:R15)
```

7 Como entregar o trabalho?

Entregue um único arquivo de nome t3.tar.gz no sistema moodle contendo pelo menos estes arquivos:

- `libconjunto.h`, `libfila.h` e `liblef.h`, mesmo que você não possa modificá-los;
- `libconjunto.c`, `libfila.c` e `liblef.c`;
- `mundo.c`, contendo a sua implementação da simulação;
- `makefile`. Os professores rodarão "make" para compilar e gerar os executáveis;
- Se você usou ou implementou mais arquivos que são importantes para o funcionamento do programa, entregue-os no mesmo pacote.

8 Considerações finais

Esse trabalho vai oferecer a oportunidade de amadurecimento na Linguagem C e em especial no uso de tipos de dados abstratos. É esperado que essa implementação tenha pelo menos 3 módulos independentes implementando os tipos abstratos de dados LEF, conjunto e fila. Esses módulos devem ser previamente testados e validados caso contrário poderá ser muito difícil encontrar erros na sua implementação.

A implementação é relativamente simples, mas exige bastante cuidado. Vocês podem começar implementando os eventos de maneira simplificada, por exemplo fazer que as pessoas cheguem e saiam dos lugares. Durante esse esforço podem notar que decisões ruins foram tomadas. Não hesitem em reescrever código, quase sempre é menos trabalhoso do que conviver com o custo das decisões equivocadas passadas.

Se o código começar a ficar muito complicado melhor repensar! A beleza deste tipo de simulação é justamente conseguir simular diversos cenários com facilidade. A implementação feita para validar a especificação tem menos de 500 linhas e o total com bibliotecas na ordem de mil linhas.

Foram determinados vários parâmetros na simulação. O objetivo foi dar dinamicidade para a simulação, os lugares serem distantes o suficiente para o tempo de deslocamento ser variável, o número de pessoas ser compatível com a lotação dos locais de forma a eventualmente termos fila, entre outros tantos *parafusos*. Durante a fase de testes vocês podem experimentar outros valores para ver o comportamento de seu mundo!

Todos vocês já interagiram com muitas simulações eletrônicas! Essa especificação serve para o requisitos da disciplina, mas os mais audazes poderão ter curiosidade de implementar novos eventos nem nosso mundo. Pessoas podem morrer, rumores podem ser esquecidos ou surgirem, enfim literalmente um universo de possibilidades. Um programador pode criar os mais diversos universos e isso é um privilegio espetacular, nossas possibilidades são infinitas. Terminamos com uma citação para vocês.

> *I etch a pattern of geometric shapes onto a stone. To the uninitiated, the shapes look mysterious and complex, but I know that when arranged correctly they will give the stone a special power, enabling it to respond to incantations in a language no human being has ever spoken. I will ask the stone questions in this language, and it will answer by showing me a vision: a world created by my spell, a world imagined within the pattern on the stone.*
>
> *A few hundred years ago in my native New England, an accurate description of my occupation would have gotten me burned at the stake. Yet my work involves no witchcraft; I design and program computers. The stone is a wafer of silicon, and the incantations are software. The patterns etched on the chip and the programs that instruct the computer may look complicated and mysterious, but they are generated according to a few basic principles that are easily explained* The Pattern on The Stone: The Simple Ideas That Make Computers Work, W. Daniel Hillis