UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br/

Engenharia de Produção
SSC0300 - Linguagem de Programação e Aplicações
Docente: Prof. Dr. Jó Ueyama (jo@icmc.usp.br)
PAE: Heitor Freitas (heitorfv@icmc.usp.br)

Segundo trabalho de LPA

Entrega 10/11/2014 - Apresentação: 17 e 24/11/2014

São Carlos, Setembro de 2014

Lista de Figuras

Sumário

1 Objetivos

Consiste em praticar os conceitos de alocação dinâmica de memória, ponteiros, entrada e saída com arquivos, tipos abstratos de dados e lista ligada.

2 *The Blocks Problem*

Este trabalho é baseado em um dos problemas da maratona de programação da ACM[2].

2.1 Introdução

Várias áreas da computação utilizam domínios simplificados para abstrair diversos tipos de problemas. Por exemplo, algumas das primeiras pesquisas de inteligência artificial nas áreas de planejamento e robótica eram feitas utilizando o ”mundo dos blocos”, no qual um braço robótico realizava tarefas simuladas envolvendo a manipulação de blocos[3].

Neste trabalho você irá implementar um *mundo dos blocos* simples[1], que deverá ser regido por regras e obedecer comandos de movimentação de blocos dados pelo usuário, simulando o que seria a manipulação através de um braço robótico. Esse mundo de blocos começa com cada bloco na sua posição inicial, e depois de uma série de comandos deve terminar em uma configuração final, como mostrado na Figura 1, para 5 blocos (n=5).

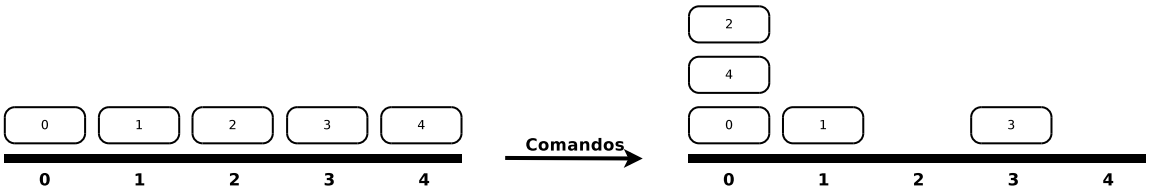

Figura 1: Exemplo com cinco blocos (n=5)

Para criar esse mundo dos blocos, você deve criar um **tipo abstrato de dados - TAD** chamado **TBlocos** que é composto por um vetor de listas encadeadas[1] e pelas funções para manipulação dessa estrutura. Por exemplo, as duas configurações mostradas na Figura 1 deverão estar implementadas no seu TAD como ilustrado pela Figura 2:

[1]Cada lista corresponde a vários blocos e cada célula corresponde a um único bloco.

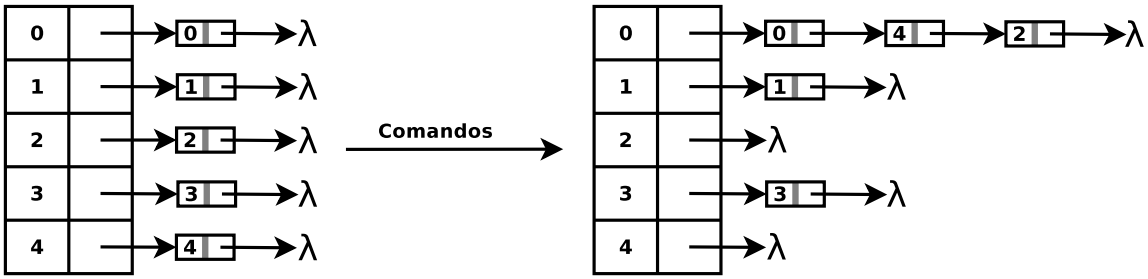

Figura 2: Estrutura de dados do exemplo da Figura 1

Os comandos possíveis para manipulação dos blocos são:

- **move a onto b**: Coloca o bloco **a** em cima do bloco **b** e retorna eventuais blocos que estiverem em cima de **a** ou **b** para seus respectivos lugares de origem, como exemplificado pela Figura 3.

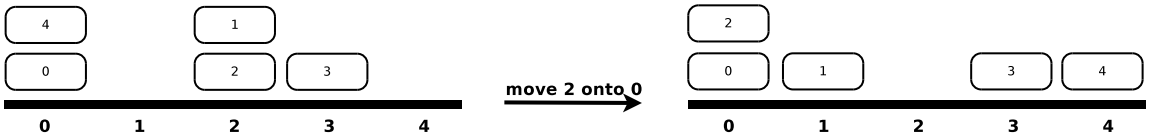

Figura 3: Exemplo do comando *move a onto b*

- **move a over b**: Coloca o bloco **a** no topo do monte onde estiver o bloco **b** e retorna os eventuais blocos que estiverem sobre **a** às suas respectivas posições de origem, como exemplificado pela Figura 4.

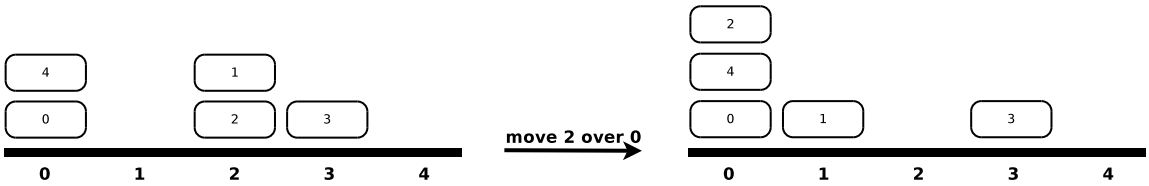

Figura 4: Exemplo do comando *move a over b*

- **pile a onto b**: Coloca o bloco **a** e os demais blocos que estiverem sobre ele em cima do bloco **b** e retorna os eventuais blocos que estiverem em cima **b** para seus respectivos lugares de origem, como exemplificado pela Figura 5.

Figura 5: Exemplo do comando *pile a onto b*

- **pile a over b**: Coloca o bloco **a** e os demais blocos que estiverem sobre ele em cima do bloco que contiver **b**, como exemplificado pela Figura 6.

Figura 6: Exemplo do comando *pile a over b*

- **quit**: Termina a execução do programa.

Comandos onde **a**=**b** ou onde **a** e **b** estejam no mesmo monte devem ser ignorados.

Além de implementar esses comandos, crie também procedimentos auxiliares no seu TAD para simplificar e evitar a repetição de código, como por exemplo, o procedimento para retornar um ou mais blocos para sua posição de origem pode ser utilizada por quase todos os comandos.

O programa principal deverá ser basicamente um *parser*[2] que lê comando a partir de um arquivo de entrada, executa os comandos e imprime o estado final dos blocos. O programa pode imprimir em *stdout* o estado dos blocos logo após cada comando.

Utilize parâmetros de linha de comando para executar o seu programa. Esse tipo de execução é bastante comum em sistemas Unix / Linux. Por exemplo, se o seu programa chama-se LPA e você quiser executar o programa com a entrada *entrada.txt* e gerar a saída no arquivo *saida.txt*.

LPA.exe entrada.txt saida.txt

[2]Programa que subdivide uma entrada para que um outro possa atuar sobre ela.

2.2 Formato da entrada

A primeira linha do arquivo de entrada traz a quantidade *n* de blocos. Em cada linha subsequente há comandos a serem executados e na última linha o comando *quit*. Veja a Tabela 1 na página 6.

2.3 Formato da saída

O arquivo de saída deve representar o estado final dos blocos. Cada linha deve conter a posição de origem seguida por dois pontos (:) e a sequencia de blocos que estão naquela posição. Entre o número dos blocos deverá ter um espaço em branco. Esse formato de saída deve ser rigorosamente obedecido.

Exemplo de entrada e a saída correspondente (ambos em arquivos):

Exemplo de entrada	**Saída para essa entrada**
5 move 3 onto 0 move 2 over 4 pile 4 onto 0 quit	0: 0 4 2 1: 1 2: 3: 3 4:

Tabela 1: Exemplo de entrada e saída

3 Como testar o seu código

Você pode testar arquivos de entrada e saída utilizando um dos vários *problem solvers* disponíveis para problemas da maratona de programação. Por exemplo, acesse http://www.uvatoolkit.com/problemssolve.php e clique no problema 101 *The Blocks Problem* e teste arquivos de entrada e saída.

4 O que deve ser entregue

Deverá ser entregue somente um arquivo zipado, contendo os seguintes arquivos:

- Código fonte C, bem indentado e comentado;
- Código fonte H, bem indentado e comentado;

- Documentação no formato *PDF* (DOCX não é PDF);
- Arquivos de entrada, no formato *TXT*;
- Arquivos de saída, no formato *TXT*.

5 Como deve ser entregue

A entrega deve ser feita via e-mail (heitorfv@icmc.usp.br) até às 7h do dia da entrega na forma de um **único** arquivo zipado contendo o código, os arquivos de teste e a documentação (se necessário). O nome do arquivo deve conter o número do grupo, exemplo: *Grupo01.zip*. Consulte a página da disciplina[3] para saber o número do seu grupo.

O assunto do e-mail deve ser ***TRABALHO 2 LPA***.

6 Como deve ser a apresentação

A apresentação com no máximo 10 minutos e deve constar com os seguintes tópicos:

- Capa com nome e número USP dos integrantes do grupo;
- Explicar como o grupo resolveu o problema;
- Comentar sobre as dificuldades encontradas;
- Referências bibliográficas.

7 Comentários gerais

- Comece a fazer este trabalho o mais cedo possível, enquanto o problema está fresco na memória e há prazo hábil;
- Tenha certeza de que você realmente entendeu o problema antes de codificar, programação é um exercício de lógica e não de tentativa e erro;
- Teste cada comando separadamente antes de executar o programa completo;
- Clareza, indentação e comentários no programa também valem pontos;

[3] http://wiki.icmc.usp.br/index.php/SSC-300-2014(jo)

- O trabalho poderá ser feito em grupos de dois ou três alunos;
- Trabalhos copiados, inclusive fonte, receberão nota ZERO;
- Trabalhos entregue em atraso serão aceitos, todavia a nota atribuída ao trabalho será zero;
- Os trabalhos serão corrigidos em Linux, portanto é **PROIBIDO** o uso de bibliotecas **conio.h** ou **windows.h**. Trabalhos que desrespeitarem essa regra receberão nota zero.

8 Dúvidas

Não tiramos dúvidas por e-mail!
Dúvidas podem ser somente tiradas nos horários de atendimento do professor e PAE. Horários de atendimento:

- **Professor**: Quartas das 17h às 18h na sala 3-115 (Bloco 3 do ICMC)[4];
- **PAE**: Segundas das 15h às 16h na sala 1-006 (Bloco 1 do ICMC)[5];
- **Monitora**: Segundas das 19h às 20h na sala 3-101 ou 3-102 (Bloco 3 do ICMC)[6].

Consulte a página da disciplina, http://wiki.icmc.usp.br/index.php/SSC-300-2014(jo), regularmente.

[4] http://wikimapia.org/#lang=pt&lat=-22.007356&lon=-47.894385&z=19&m=b&show=/8429284/pt/ICMC-Bloco-3
[5] http://wikimapia.org/#lang=pt&lat=-22.008156&lon=-47.895799&z=19&m=b&show=/6251772/pt/ICMC-USP-Bloco-1
[6] http://wikimapia.org/#lang=pt&lat=-22.007356&lon=-47.894385&z=19&m=b&show=/8429284/pt/ICMC-Bloco-3

Referências

[1] The blocks problem - http://online-judge.uva.es/p/v1/101.html.

[2] ACM. Maratona de programação, 2014. http://maratona.ime.usp.br/.

[3] Stuart Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach*. Prentice Hall, 2nd edition, 2003.