Instituto de Computação - Unicamp

MC102 - Algoritmos e Programação de Computadores

Laboratório 15 - Brincando com Cordas

Prazo de entrega: **26/05/2017 23:59:59**

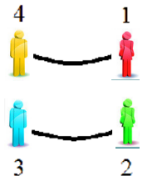
Peso: **1**

Professor: Eduardo C. Xavier
Professor: Guido Araújo

Monitor: Arthur Pratti Dadalto
Monitor: Cristina Cavalcante
Monitor: Klairton de Lima Brito
Monitor: Luís Felipe Mattos
Monitor: Paulo Finardi
Monitor: Paulo Lucas Rodrigues Lacerda
Monitor: Pedro Alves
Monitor: Renan Vilas Novas
Monitor: Vinicius de Novaes Guimarães Pereira

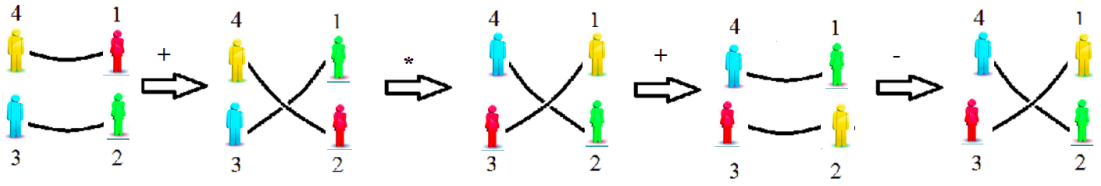
Descrição

Quatro crianças estão brincando segurando 2 cordas. A cada rodada, elas se posicionam formando um quadrado, com as cordas separadas, como mostra a figura abaixo.

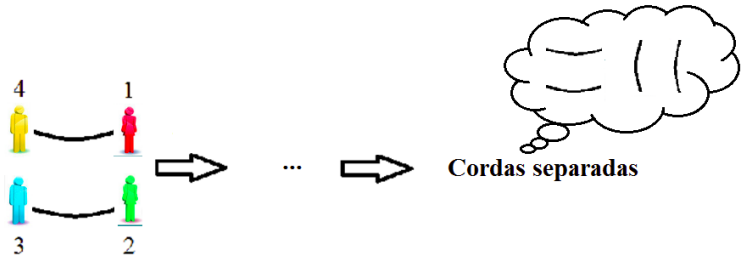


Em seguida, elas realizam uma sequência de movimentos, que podem ser dos seguintes tipos:

- Movimento '+' (TROCA POR CIMA): A criança que está na posição 1 troca de lugar com a criança que está na posição 2, passando sua corda por cima da dela.
- Movimento '-' (TROCA POR BAIXO): A criança que está na posição 1 troca de lugar com a criança que está na posição 2, passando sua corda por baixo da dela.
- Movimento '*' (GIRO): As crianças giram no sentido horário (quem está na posição 1 vai para a 2, quem está na 2 vai para a 3, quem está na 3 vai para 4, e quem está na 4 vai para 1).



Os movimentos podem ser executados em qualquer ordem, e o desafio é saber se, após a execução de uma dada sequência de movimentos '+', '-' e '*', as cordas vão terminar emaranhadas ou vão estar novamente separadas (paralelas na horizontal ou na vertical).



Neste laboratório, você deve fazer um programa que, dada uma sequência de movimentos '+', '-' e '*', consiga prever o estado final das cordas.

O matemático Jonh Conway, pai desta brincadeira (que em inglês é chamada de *rational tangle dance*), dá as seguintes dicas para você:

1 - Você deve processar a sequência de movimentos, removendo subsequências que não afetam o resultado: "+-", "-+", "**".

2 - Se a sequêcia de movimentos começa com um GIRO ('*'), qualquer movimento de TROCA subsequente ('+' ou '-') não vai emaranhar as cordas até que um outro movimento GIRO seja executado. Sendo assim, se a sequência de movimentos começa com '*', você deve remover todos os movimentos da primeira posição da sequência até a segunda aparição de '*', inclusive.

3 - Existem duas subsequências de movimentos que mascaram o resultado. São elas: "+*+" e "-*-". Sempre que uma dessas subsequências aparecer, elas devem ser substituídas por "*-*" e "*+*", respectivamente.

4 - As alterações 1, 2 e 3 devem ser executadas até que a sequência de movimentos não seja mais alterada. Se a sequência final obtida for vazia ou igual a "*", as cordas terminarão separadas. Caso contrário, as cordas terminarão emaranhadas.

Para executar as alterações 1, 2 e 3, você deve implementar três funções:

- **removePadrao:** Remove, de uma dada sequência de movimentos, todas as ocorrências de uma dado padrão (subsequência de movimentos).
- **removeBloco:** Se uma dada sequência de movimentos começa com um determinado caractere, remove da sequência todos os movimentos da primeira posição até a segunda ocorrência (inclusive) do dado caractere.
- **substituiPadrao:** Substitui, em uma dada sequência de movimentos, todas as ocorrências de um dado padrão, por um novo padrão.

As alterações na sequência original devem ser feitas na seguinte ordem:

- removeBloco (alteração tipo 2)
- removePadrao (alteração tipo 1, para os padrões "**", "+-", "-+", nesta ordem).
- substituiPadrao (alteração tipo 3, para os padrões "+*+" e "-*-", nesta ordem).

OBS: Você deve atentar somente aos padrões existentes na string no momento da busca, novos padrões que possam aparecer ao remover/subtituir um padrão já existente devem ser ignorados.

Objetivo

O objetivo deste laboratório é exercitar os conceitos de funções e manipulação de strings, através da criação de um programa que consiga prever o estado final de duas cordas (separadas ou emaranhadas) após a execução de uma sequência de movimentos '+', '-' e '*'.

A descrição geral dos parâmetros de entrada e saída das funções estão nos protótipos de cada uma delas, fornecidos a seguir:

Linguagem C:

```
/* Laboratorio 15 - Brincando com cordas
 * Nome:
 * RA:
 */

#include <stdio.h>
#include <string.h>

/* Funcao: removePadrao
 *
 *    Processa linearmente (e uma unica vez) os carateres de uma dada
 *    removendo as ocorrencias de um dado padrao.
 *
 * Parametros:
 *          str: sequencia de movimentos '+', '-', '*'
 *       padrao: subsequencia de movimentos, cujas ocorrencias devem
 *    targetStr: sequencia obtida apos a remocao de str de todas as o
```

```
 *
 *
 * Retorno:
 *
 *    1, se as occorrencias do dado padrao foram removidas
 *    0, caso contrario
 *
 * Exemplo:
 *
 *     char seq1[15] = "+-*+-*-*+++";
 *     char seq2[15];
 *
 *     char padrao1[3] = "+-"
 *     char padrao2[3] = "**"
 *
 *     removePadrao(seq1, padrao1, seq2); // seq2 sera "**-*+++"
 *     removePadrao(seq1, padrao2, seq2); // seq2 sera "+-*+-*-*+++"
 *
 */
int removePadrao(char str[], char padrao[], char targetStr[]) {
}

/* Funcao: removeBloco
 *
 *    Processa linearmente (e uma unica vez) os carateres de uma dada
 *    removendo o bloco inicial delimitado pelo dado caractere.
 *
 * Parametros:
 *          str: sequencia de movimentos '+', '-', '*'
 *            c: caractere delimitador do bloco a ser removido
 *    targetStr: se str comeca com o caractere do parametro c, target
 *               apos a remocao (de str) do bloco de movimentos da pr
 *               segunda ocorrencia (inclusive) do caractere do param
 *               ocorrencia do caracter do parametro c, targetStr dev
 *
 * Retorno:
 *
 *    1, se o bloco foi removido
 *    0, caso contrario
 *
 * Exemplo:
 *
 *     char seq1[15] = "+-*+-*-*+++";
```

```
 *     char seq2[15] = "*+-*+-*-*+++";
 *     char seq3[15] = "*+-+--+++";
 *     char seq4[15];
 *
 *     removeBloco(seq1, '*', seq4); // seq4 sera "+-*+-*-*+++"
 *     removeBloco(seq2, '*', seq4); // seq4 sera "+-*-*+++"
 *     removeBloco(seq3, '*', seq4); // seq4 sera "" (sequencia vazia
 *
 */
int removeBloco(char str[], char c, char targetStr[]) {
}

/* Funcao: substituiPadrao
 *
 *    Processa linearmente (e uma unica vez) os carateres de uma dada
 *    substituindo as ocorrencias de um dado padrao por um novo padra
 *
 * Parametros:
 *           str: sequencia de movimentos '+', '-', '*'
 *        padrao: subsequencia de movimentos, cujas ocorrencias devem
 *    novoPadrao: subsequencia de movimentos que deve substituir cada
 *     targetStr: sequencia obtida apos a substituicao em str de toda
 *                pelo novoPadrao
 *
 *     Voce pode assumir que as subsequencias padrao e novoPadrao tem
 *
 * Retorno:
 *
 *    1, se as ocorrencias do dado padrao foram substituidas pelo nov
 *    0, caso contrario
 *
 * Exemplo:
 *
 *     char seq1[15] = "+*+-+*+-*++";
 *     char seq2[15];
 *     char seq3[15];
 *
 *     char padrao[5] = "+*+"
 *     char novoPadrao[5] = "*-*"
 *
 *     substituiPadrao(seq1, padrao, novoPadrao, seq2); // seq2 sera
 *     substituiPadrao(seq2, padrao, novoPadrao, seq3);  // seq3 sera
 *
```

```
 */
int substituiPadrao(char str[], char padrao[], char novoPadrao[], cha
}
```

Múltiplos Arquivos e Função Principal

Neste laboratório vamos utilizar o conceito de dividir o código em múltiplos arquivos. Quando se implementa programas grandes é comum separar o código em vários arquivos com a extensão .c, onde cada arquivo implementa um conjunto de funções relacionadas entre si. Isto facilita a manutenção e a leitura do código. Para compilar um código organizado dessa forma, basta passar todos os arquivos na linha de comando para o compilador.

Para esse laboratório você só deverá implementar as funções descritas acima. A função principal (**main**) será fornecida em um arquivo separado, chamado `lab15_main.c` .
Um link para ele também está disponível na página da tarefa.

Vamos ao exemplo de como compilar o seu programa em C. Até agora, a forma que utilizamos (de forma simplificada) era a seguinte:

```
gcc -o labXX labXX.c
```

Nesse laboratório, no entanto, para compilar o seu programa basta adicionar o arquivo extra que provemos (lab15_main.c) ao final da linha de comando, como no exemplo a seguir:

```
gcc -o lab15 lab15.c lab15_main.c
```

OBS: A linha completa de compilação para esse laboratório pode ser vista na sessão de Observações.

Entrada

O programa recebe como entrada uma sequência de até 30 movimentos

'+', '-' e '*'.

Dica para a linguagem C:

Leia a entrada com *fgets* e remova o '\n' do final da string de entrada da seguinte forma:

```
fgets(seq, 30, stdin);
seq[strlen(seq)-1] = '\0';
```

Saída

Para cada alteração (realmente feita) na sequência de movimentos de entrada, o programa deve imprimir a alteração feita e nova sequência obtida. Quando a sequência não for mais alterada, se ela indicar que as cordas estão separadas, o programa deve imprimir "Cordas terminam separadas!"; caso contrário, o programa deve imprimir "Cordas terminam emaranhadas!"

Dicas para a linguagem C:

Para alterações feitas com a função **removeBloco**, imprima o resultado da seguinte forma, onde *novaSeq* é a sequência obtida com a alteração:

```
printf("removeBloco(*): %s\n", novaSeq);
```

Para alterações feitas com a função **removePadrao**, imprima o resultado da seguinte forma, onde *padrao* é o padrão removido e *novaSeq* é a sequência obtida com a alteração:

```
printf("removePadrao(%s): %s\n", padrao, novaSeq);
```

Para alterações feitas com a função **substituiPadrao**, imprima o resultado da seguinte forma, onde *padrao* é o padrão substituído por *novoPadrao*, e *novaSeq* é a sequência obtida com a alteração:

```
printf("substituiPadrao(%s, %s): %s\n", padrao, novoPadrao, novaSeq);
```

Imprima a conclusão sobre os estado final das cordas da seguinte forma:

```
printf("Cordas terminam separadas!\n"); // Se as cordas terminarem se
printf("Cordas terminam emaranhadas!\n"); // Se as cordas terminarem
```

Reforçando

Neste laboratório você não precisará se preocupar em ler a entrada a partir da entrada padrão, nem em escrever a saída. Seu trabalho é apenas implementar as funções descritas. A função `main()` que é fornecida no arquivo `lab15_main.c` se encarrega dessa parte.

Você também **não deve** submeter o arquivo `lab15_main.c` para o *SuSy*, somente o arquivo `lab15.c` .

Também está disponível um protótipo do arquivo que você deve submeter ao *SuSy* (lab15.c). Esse arquivo e o arquivo auxiliar (lab15_main.c) também podem ser encontrados na página da tarefa:

- `lab15.c`
- `lab15_main.c`

As sessões abaixo, de Entrada e Saída, descrevem os formatos de entrada e saída, mas você não precisa se preocupar com eles.

Exemplos

Teste 01

Entrada

```
+
```

Saída

```
Cordas terminam emaranhadas!
```

Teste 02

Entrada

```
*
```

Saída

```
removeBloco(*):
Cordas terminam separadas!
```

Teste 03

Entrada

```
+**-
```

Saída

```
removePadrao(**): +-
removePadrao(+-):
Cordas terminam separadas!
```

Teste 06

Entrada

```
+*+
```

Saída

```
substituiPadrao(+*+, *-*): *-*
removeBloco(*):
Cordas terminam separadas!
```

Teste 07

Entrada

```
+-++*+
```

Saída

```
removePadrao(+-): ++*+
substituiPadrao(+*+, *-*): +*-*
Cordas terminam emaranhadas!
```

Teste 08

Entrada

```
*++++++
```

Saída

```
removeBloco(*):
Cordas terminam separadas!
```

Para mais exemplos, consulte os testes abertos no SuSy.

Observações

- Você **não deve** submeter o arquivo `lab15_main.c` para o *SuSy*, somente o arquivo `lab15.c` .
- O número máximo de submissões é **15**.

- Para a realização dos testes do SuSy, a compilação dos programas desenvolvidos em C irá considerar o comando:
  `gcc -std=c99 -pedantic -Wall -o lab15 lab15.c lab15_main.c .`
- Você deve incluir, no início do seu programa, uma breve descrição dos objetivos do programa, da entrada e da saída, além do seu nome e do seu RA.
- Indente corretamente o seu código e inclua comentários no decorrer do seu programa.

Critérios importantes

Independentemente dos resultados dos testes do *SuSy*, o não cumprimento dos critérios abaixo implicará em nota zero nesta tarefa de laboratório.

- Os únicos headers aceitos para essa tarefa são `stdio.h` e `string.h` .