



MAC122 – Princípios de Desenvolvimento de Algoritmos

DCC - IME - USP

2º Semestre de 2014

Prof.: Dr. Paulo Miranda & Marco Aurélio Gerosa

pmiranda@vision.ime.usp.br, gerosa@ime.usp.br

Lista 03: Pilhas & Filas

Para as questões envolvendo pilhas, assuma a existência das funções:

```
Pilha      CriaPilha();  
void       LiberaPilha(Pilha p);  
bool       PilhaVazia(Pilha p);  
void       Empilha(Pilha p, TipoDado x);  
TipoDado   Desempilha(Pilha p);
```

Assuma também a definição:

```
typedef enum boolean {false,true} bool;
```

Você deve definir **TipoDado** usando **typedef** do modo que considerar melhor.

Questão 1:

Faça uma função que use uma pilha para inverter a ordem das letras de cada palavra de uma string, preservando a ordem das palavras. Por exemplo, dado o texto:

“EXERCICIO MUITO COMPLICADO”

A saída deve ser: “OICICREXE OTIUM ODACILPMOC”.

Questão 2:

Considere a família de cadeias de caracteres formadas apenas pelas letras a, b e c, seguindo o padrão:

c, aca, bcb, abcba, bacab, aacaa, bbcbb, ...

Qualquer cadeia dessa família tem a forma **WcM**, onde **W** é uma sequência de letras que só contém **a** e **b**, e **M** é o inverso de **W**, ou seja, **M** é **W** lido de trás para frente. Faça uma função que use uma pilha para determinar se uma cadeia **X** pertence ou não ao nosso conjunto, ou seja, determina se **X** é da forma **WcM**.

```
bool formaWcM(char X[]);
```

Exemplo de chamada da função:

```
int main() {  
    char X[] = "abaabcbaaba";  
  
    if( formaWcM(X) )  
        printf("Pertence\n");  
    else  
        printf("Falso\n");  
  
    return 0;  
}
```

Questão 3:

Diremos que uma expressão posfixa simples é uma string cujos caracteres pertencem ao conjunto '+', '*', '0', '1', ..., '9' (o caracter ' ' não faz parte do conjunto). Uma expressão posfixa simples representa uma expressão aritmética de modo alternativo, onde os argumentos precedem o símbolo da operação. Por exemplo, "123*+" representa a expressão $1 + (2 * 3)$, cujo valor é 7. Note que cada caracter numérico representa um número entre 0 e 9; assim, a sequência "48" representa o número 4 seguido do número 8, e não o número 48. Escreva uma função que receba uma expressão posfixa simples e devolva o valor da correspondente expressão aritmética.

```
int ValorExpressao(char posfixa[]);
```

Exemplo de chamada da função:

```
int main() {  
  
    char pos[] = "123+*45+*6*";  
    int v;  
  
    v = ValorExpressao(pos);  
    printf("v: %d\n",v); /* Imprime v: 270 */  
  
    return 0;  
}
```

Questão 4:

Item A:

Faça uma função em C que recebe em `cadeia` uma string não balanceada de parênteses e colchetes, e que gera em `corr` uma versão corrigida balanceada, através da inserção de parênteses e colchetes ausentes. O número de caracteres adicionais inseridos deve ser retornado pela função.

```
int CorrigeBalanceamentoFA(char *cadeia, char *corr);
```

Durante o processamento da string `cadeia` (da esquerda para a direita), sempre que for detectado uma falha de balanceamento, a função deve sempre resolver o conflito imediatamente através da inserção de um caracter adicional. Nessa primeira versão, considere que a função sempre dá preferência para inserir um caracter de fechamento para a resolução de conflitos, sendo que caracteres de abertura adicionais só devem ser usados se não houver outra opção imediata.

Dica: Use como base o código visto em aula que verifica o balanceamento de parênteses, e mostre quais modificações são necessárias para a resolução do novo problema.

Exemplos:

Para a `cadeia` = " [(] " a saída esperada na string `corr` é " [()] () ", e o valor de retorno da função é 2, dado que dois novos caracteres foram inseridos durante a correção (ver setas).

Para " [(]) " a saída em `corr` é " [()] () [] ", e o valor de retorno é 3.

Para " [(] " a saída em `corr` é " [()] ", e o valor de retorno da função é 1.

OBS: Para a resolução dessa questão, além das funções de pilhas, considere também as seguintes funções auxiliares:

```
char ParDeAbertura(char c){  
    if(c=='[') return '[';  
    else if(c=='(') return '(';  
}
```

```
char ParDeFechamento(char c){  
    if(c== '[') return ']';  
    else if(c=='(') return ')';  
}
```

Item B:

Considere agora uma segunda versão da função que sempre dá preferência para inserir um caracter de abertura para a resolução de conflitos, sendo que caracteres de fechamento adicionais só devem ser usados se não houver outra opção imediata.

```
int CorrigeBalanceamentoAF(char *cadeia, char *corr);
```

Exemplos:

Para " [(] " a saída em `corr` é " [([])] ", e o valor de retorno é 2.

Para " [(]) " a saída em `corr` é " [([])] ", e o valor de retorno é 1.

Para " [(] " a saída em `corr` é " [([])] ", e o valor de retorno é 3.

Questão 5:

Na notação pré-fixa o operador precede os dois operandos. Exemplos:

infixa	pré-fixa
$a+b$	$+ab$
$a+b*c$	$+a*bc$
$(a+b)*c$	$*+abc$

Considerando os operadores '+', '-', '*' e '/', faça uma função usando pilhas que converte uma expressão em notação infix para pré-fixa. No caso de operadores de mesmo nível de prioridade, considere a associação à esquerda.

OBS: Assim como na notação pós-fixa, a notação pré-fixa também não necessita de parênteses. A própria forma da expressão indica a ordem das operações.

Questão 6:

A notação pós-fixa pode ser estendida para operadores monádicos (de um operando), entretanto eles introduzem ambiguidade. Por exemplo, as expressões infixas distintas " $a - (-b)$ " e " $-(a - b)$ " seriam traduzidas para a mesma expressão pós-fixa " $ab--$ ". A fim de evitar este problema, os operadores devem utilizar símbolos distintos na tradução. Adotando-se o símbolo \sim para o operador unário $-$, teríamos:

$a - (-b)$  $ab\sim-$

$-(a - b)$  $ab--\sim$

Complete a implementação da função de conversão de notação infix para pós-fixa usando pilhas vista em aula para incluir o operador monádico '-'.

Questão 7:

Em algumas aplicações é interessante utilizar o conceito de fila dupla. Nesta estrutura, as inserções e remoções podem ser feitas em qualquer uma das duas extremidades da sequência. Proponha implementações ligada e sequencial para esta estrutura e escreva as rotinas básicas para manipulá-la:

```
FilaDupla CriaFilaDupla();
```

```
void LiberaFilaDupla(FilaDupla p);
```

```
bool FilaDuplaVazia(FilaDupla p);
```

```
void InsereFrenteFilaDupla(FilaDupla *p, TipoDado x);
```

```
void InsereFimFilaDupla(FilaDupla *p, TipoDado x);
```

```
TipoDado RemoveFrenteFilaDupla(FilaDupla *p);
```

```
TipoDado RemoveFimFilaDupla(FilaDupla *p);
```