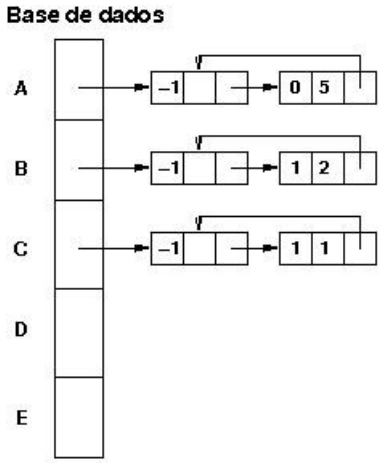
Universidade Federal de São Carlos
Bacharelado em Ciência da Computação
Estruturas de Dados I
Prof. Tiago A. Almeida <talmeida@ufscar.br>

EXERCÍCIO 7 – CALCULADORA DE POLINÔMIOS (PESO 3)

- **Prazo para entrega: 16/05/2013 – 12:00:00**
- **Atenção:**
  1. **Arquivo:** o nome do arquivo referente ao código-fonte deverá respeitar o seguinte padrão: <número do RA>**_ED1_EX**<número do exercício>.c. Exemplo: `123456_ED1_EX07.c`;
  2. **E/S:** tanto a entrada quanto a saída de dados devem ser "secas", ou seja, não devem apresentar frases explicativas. Siga o modelo fornecido e apenas complete as partes informadas.
  3. **Identificadores de variáveis:** escolha nomes apropriados;
  4. **Documentação:** inclua comentários e indentação no programa.
- **Descrição**

  Considere um vetor com 10 posições, sendo que cada uma delas possui um apontador para um polinômio. Assim como no **ex04**, o seu programa deverá ser capaz de ler polinômios e realizar as operações de soma, subtração e agora multiplicação entre eles. Para isso, o usuário poderá informar **expressões aritméticas** utilizando polinômios de A à J. Veja um exemplo de vetor de polinômios na figura abaixo.



  Note que, a representação acima está armazenando três polinômios: $A = 5$, $B = 2x$ e $C = x$. Neste exercício, você precisará implementar um interpretador de expressões aritméticas e

calcular o polinômio resultante. As expressões de entrada estão definidas na forma infixa e, quando necessário são usados parênteses para indicar precedência, conforme exemplo abaixo.

Exemplo de expressão fornecida:

$$((A + B) * C) + D, \text{ onde } A, B, C \text{ e } D \text{ são polinômios.}$$

Ao receber uma expressão em notação infixa, a sua primeira tarefa é convertê-la para as notações prefixa e posfixa, respectivamente. Para isso, utilize a estrutura de dados **pilha**, conforme exposto em aula. Veja abaixo um exemplo das notações:

$$((A + B) * C) + D \Rightarrow \text{notação infixa}$$

$$AB + C * D+ \Rightarrow \text{notação posfixa}$$

$$+ * +ABCD \Rightarrow \text{notação prefixa}$$

Observação: as notações prefixa e posfixa dispensam o uso dos parênteses. Logo, ao converter a expressão da forma infixa para as demais, você precisará removê-los.

O seu programa contará com o seguinte menu:

1. **Ler polinômio:** permite ao usuário informar um polinômio no próximo *slot* vazio e seus termos (expoente e coeficiente, respectivamente) para inserção.
2. **Apagar polinômio:** apaga o polinômio de um dos *slots* (A a J) informados.
3. **Imprimir polinômios:** imprime todos os polinômios armazenados (de A a J) ou a mensagem "`Polinomio vazio!`", caso ele não tenha sido informado. Note que, na impressão, cada polinômio deve ser exibido em ordem decrescente do coeficiente dos seus termos.
4. **Ler expressão:** recebe uma expressão aritmética válida escrita na forma infixa usando parênteses quando necessário. Observe que não há espaços em branco entre os operadores, operandos e parênteses. A saída deverá ser a expressão de entrada escrita na forma prefixa e posfixa, respectivamente.
5. **Resolver expressão:** resolve a última expressão aritmética informada e exibe o polinômio resultante.
6. **Sair:** apaga todos os polinômios e libera memória alocada.

Complete o arquivo `ex07.c`

Você deve apenas completar as operações nos lugares indicados e não deve realizar nenhuma alteração nas funções *main*, *imprimirTermo*, *lerPolinomio* e *precedencia*. Inclusive, os comandos de entrada e saída de dados não podem ser alterados.

Complete as seguintes funções:

- `inicPolinomio`: cria o nó cabeça com expoente igual a -1 (termo identicamente nulo).
- `polinomioVazio`: verifica se o polinômio está vazio.
- `imprimirPolinomio`: imprime o polinômio na tela em ordem decrescente de expoente. Utilize a função `imprimirTermo` para auxílio.
- `inserirTermo`: insere um novo termo no polinômio. O termo deve ser inserido e armazenado em ordem crescente de expoente. **Note que o armazenamento do polinômio é o contrário de sua impressão**.
- `apagarPolinomio`: remove todo o polinômio, liberando a memória de cada nó alocado, inclusive do nó cabeça.
- `somarPolinomios`: soma os polinômios A e B, lembrando que se os expoentes forem diferentes, seu programa deve mantê-los em ordem crescente.
- `subPolinomios`: subtrai o polinômio B de A, ou seja, $A - B$. Lembre-se que se os expoentes forem diferentes, seu programa deve mantê-los em ordem crescente.
- `multPolinomios`: faz a multiplicação entre os polinômios A e B. Lembre-se que se os expoentes forem diferentes, seu programa deve mantê-los em ordem crescente.
- `criarPilhaExpr`: inicializa a pilha de expressões.
- `pilhaExprVazia`: retorna VERDADEIRO caso e pilha de expressões esteja vazia. Retorna FALSO caso contrário.
- `liberarPilhaExpr`: libera memória alocada para a pilha de expressões.
- `empilharExpr`: cria um novo nó para a pilha de expressões e o coloca no topo. Retorna VERDADEIRO se a operação foi realizada com sucesso, FALSO caso contráio.
- `desempilharExpr`: remove o elemento que está no topo da pilha e retorna VERDADEIRO. Caso haja erro na remoção, retornar FALSO.
- `topoExpr`: recupera o elemento que está no topo da pilha e retorna VERDADEIRO. Retorna FALSO em caso de erro.
- `criarPilhaPol`: inicializa a pilha de polinômios.
- `pilhaPolVazia`: retorna VERDADEIRO caso e pilha de polinômios esteja vazia. Retorna FALSO caso contrário.
- `pilhaPolCheia`: retorna VERDADEIRO caso e pilha de polinômios esteja cheia. Retorna FALSO caso contrário.
- `empilharPol`: coloca um polinômio no topo da pilha e retorna VERDADEIRO. Retorna FALSO caso contrário.
- `desempilharPol`: remove o elemento que está no topo da pilha e retorna VERDADEIRO. Caso haja erro na remoção, retorna FALSO.
- `infixa_para_posfixa`: converte uma expressão em notação infixa para uma expressão em notação posfixa. **Observação**: Não esqueça que a notação posfixa dispensa o uso de parênteses.

- `infixa_para_prefixa`: converte uma expressão em notação infixa para uma expressão em notação prefixa. **Observação**: Não esqueça que a notação prefixa dispensa o uso de parênteses.

Importante: Note que a pilha de expressões (conversor de expressões) é dinâmica e a pilha de polinômios (calculadora) é estática.

Exemplo de E/S (os comentários entre parênteses não deverão ser exibidos):

Entrada	**Saída**
`1` (opção de inserir termo)	
`3 2` (insere $2x^3$)	
`4 2` (insere $2x^4$)	
`3 3` (insere $3x^3$)	
`-1` (fim da inserção)	$A = 2.0x^4 + 5.0x^3$
`1` (opção de inserir termo)	
`4 3` (insere $3x^4$)	
`2 4` (insere $4x^2$)	
`4 2` (insere $2x^4$)	
`-1` (fim da inserção)	$B = 5.0x^4 + 4.0x^2$
`3` (imprime os polinômios)	$A = 2.0x^4 + 5.0x^3$
	$B = 5.0x^4 + 4.0x^2$
	C = Polinomio vazio!
	D = Polinomio vazio!
	E = Polinomio vazio!
	F = Polinomio vazio!
	G = Polinomio vazio!
	H = Polinomio vazio!
	I = Polinomio vazio!
	J = Polinomio vazio!
`4` $A * B$ (informa uma expressão na notação infixa)	Notação infixa: $A * B$
	Notação prefixa: $*AB$
	Notação posfixa: $AB*$
`5` (resolve a última expressão informada)	$AB* = 10.0x^8 + 25.0x^7 + 8.0x^6 + 20.0x^5$
`6` (finaliza o programa)	

- **Cuidados:**

1. **Erros de compilação:** nota **zero** no exercício
2. **Tentativa de fraude:** nota **zero** para todos os envolvidos.