UNIVERSIDADE FEDERAL DE UBERLÂNDIA
2º Trabalho de Algoritmos e Estrutura de Dados 1
Prof. Luiz Gustavo Almeida Martins

- Os códigos deverão ser implementados somente em Linguagem C, sendo necessária a utilização das estruturas de dados conforme discutido nas aulas.
- A implementação do TAD deve contemplar todas as operações básicas da estrutura em questão, a saber: criação, verificação de vazia e cheia (quando pertinente), inserção e remoção de elementos, apagar e esvaziar a estrutura.
- Deve-se aproveitar o conhecimento sobre a estrutura para buscar a maior eficiência das operações implementadas no código.
- A entrega do trabalho consiste no envio de um arquivo compactado (**Trab2_NomeAluno.zip**), contendo todos os arquivos necessários para a compilação e execução do programa, organizados em diretórios (uma pasta por questão). O envio é individual e deve ser feito até a data e horário especificados na tarefa. **A integridade desse arquivo é de responsabilidade dos alunos**.

1) (2 ptos) Implemente um TAD **Pilha estática/sequencial**, de no máximo 10 **números inteiros** e todas as operações básicas.

Utilizando esse TAD, desenvolva um programa aplicativo que faça a conversão de um número inteiro positivo (incluindo o zero), digitado pelo usuário, para outras bases. A escolha da base que o número será convertido (B-binário, O-octal ou H-hexadecimal) também será informado pelo usuário. Esse processo deve se repetir até que o usuário digite um número negativo.

2) (5 ptos) Implemente um TAD **Pilha dinâmica/encadeada de caracteres**, e outro TAD **Pilha dinâmica/encadeada** de **números reais**. Ambos com todas as operações básicas.

Utilizando esses TADs de acordo com a necessidade, desenvolva um programa aplicativo para a manipulação de expressões matemáticas. As expressões só aceitam variáveis (de A a F) como operandos; operadores de adição (+), subtração (-), divisão (/), multiplicação (*) e potenciação (^); e os delimitadores de escopo parênteses, colchetes e chaves. O programa deve solicitar ao usuário a entrada da expressão na forma infixa. Em seguida, a expressão digitada deve ser submetida às seguintes operações da aplicação:

- **Validação de escopo:** percorre a expressão verificando se os escopos estão sendo abertos e fechados corretamente. Nessa verificação, também deve ser **considerada a precedência entre os delimitadores**, ou seja, chaves não podem ser usadas dentro de colchetes ou parênteses, e colchetes não podem estar dentro de parênteses. Os demais aninhamentos são permitidos (ex: parênteses dentro de parênteses). Seguem alguns exemplos de expressões válidas e inválidas:

- Expressão **[({A+D)/B}*F]** inválida (ordem dos fechamentos divergem da ordem das aberturas).
- Expressão **[({A+D}/B)*F]** inválida (chaves dentro de parênteses)
- Expressão **{[(A+D)/B]*F}** é válida
- Expressão **(A-B)^C+[D/A]** é válida

Ao final do processo, a função deve retornar se a expressão é válida ou não. Em caso de erro, ela deve mostrar uma mensagem indicando o problema encontrado (ex: fechamento sem abertura, mais aberturas que fechamentos de parênteses, etc.). As próximas operações só devem ser realizadas se a expressão for válida. Caso contrário, o programa deve solicitar uma nova expressão.

- **Conversão da expressão:** realizar a conversão da expressão infixa para a forma pós-fixa. Além de retornar a expressão pós-fixada, a operação também deve indicar se a conversão foi bem-sucedida (1) ou não (0). Em caso de erro, ela deve mostrar uma mensagem indicando o problema encontrado. Quando a conversão for bem-sucedida, o programa deve exibir a expressão no formato pós-fixado e iniciar sua avaliação. Caso contrário, ele deve solicitar uma nova expressão.

- **Avaliação da expressão:** esse módulo deve solicitar os valores das literais usadas na expressão (um único valor por literal) ao usuário. Em seguida, resolvê-la utilizando esses valores e mostrar o resultado obtido ou uma mensagem indicando o erro encontrado (ex: falta de operando ou de operador). Exemplos de falha:
  - **(A+D)*(^C)** não é válida (falta um operando)
  - **(AD)/B^F** não é válida (falta operador)

O programa repetirá todo processo até que o usuário digite "FIM" para a expressão.

3) (5 ptos) Implemente um TAD Fila estática/sequencial (com desperdício de posição) de carros. O TAD deve contemplar todas as operações básicas e a função tamanho que retorna a quantidade de elementos na fila passada como entrada. Para cada carro são guardados os seguintes dados: placa, tipo do serviço (A – avulso ou M – mensal) e hora de entrada (usar a biblioteca time.h para o registro e manipulação desse dado).

Utilizando as operações desse TAD, desenvolva um programa aplicativo para controlar a entrada e saída de veículos em um estacionamento com as seguintes características:

- O estacionamento é composto por 3 boxes, sendo que em cada box pode ter no máximo 5 veículos estacionados.

- Para retirar um veículo do meio de um box, é necessário retirar os veículos que estão na frente e colocá-los novamente no final do box.

Além da opção de sair, o programa deve apresentar um menu com as seguintes funcionalidades:

a) **Entrada de veículos:** onde o atendente (usuário) cadastrará a placa e o tipo do serviço do veículo que está entrando no estacionamento. A hora de entrada deve ser gerada automaticamente, usando o horário do sistema (biblioteca time.h). O veículo cadastrado será colocado no box mais vazio, visando minimizar o remanejamento de veículos na retirada. Em caso de empate, será usada a ordem natural dos boxes, ou seja, o boxe 1 tem maior precedência que os boxes 2 e 3 e o boxe 2 tem maior precedência que o boxe 3. A hora de entrada e o número do box selecionado devem ser mostrados na tela após a inclusão do veículo no respectivo box. Quando todos os boxes estiverem cheios, uma mensagem deve informar que o estacionamento está lotado e o veículo será rejeitado.

b) **Saída de veículos:** onde o atendente indica a placa do veículo que está saindo. A partir desta placa, o programa deve buscar o veículo nos boxes e, caso encontre-o, deve retirá-lo do estacionamento. Se o carro for avulso (tipo de serviço), o sistema deve apresentar a hora de saída do veículo. Caso contrário (serviço mensal), deve informar que o cliente é mensalista. Quando o veículo não for encontrado, o sistema deve emitir uma mensagem indicando que a placa digitada não está no estacionamento.

4) (5 ptos) Após assistir a vídeo-aula referente à outras estruturas de dados lineares (***21-OutrasEstruturas.mp4***), implemente, usando alocação **dinâmica/encadeada circular e inserção ordenada**, um **Fila de Prioridade Ascendente (FPA)** de pacientes, contendo as seguintes informações: nome, idade, peso, altura, tipo da doença (‘C’-cardíaca, ‘N’- neurológica, ‘S’ - sanguínea ou ‘R’ – respiratória) e a gravidade da doença (usado como prioridade), sendo 1 a mais grave e 5 a menos grave. A estrutura que representa o paciente deve ser declarada de modo que possa ser acessada pelo programa aplicativo. Além das operações básicas, esse TAD deve conter as funções ***tamanho***, que retorna a quantidade de elementos na FPA passada como entrada, e ***get_elem_pos***, que devolve o paciente que está na posição solicitada (começando em 1), caso ela exista. Se a posição não existir, essa operação deve retornar falha. Essa última operação deve ser utilizada na aplicação para imprimir a FPA.

Utilizando apenas as operações básicas disponibilizadas no TAD, implemente um programa aplicativo que permita criar, esvaziar, apagar e imprimir uma FPA, bem como inserir e remover elementos (mostrando os dados do paciente removido) em uma FPA já criada. Essas ações devem ser executadas repetidamente até que o usuário solicite a saída do sistema, exceto pela criação da FPA que só pode ser executada se a FPA não existir (no início do programa ou após a FPA anterior ser apagada).