Estruturas de Dados

Trabalho Prático I - Pilhas

1. Introdução:

Os objetivos principais deste trabalho são:

- Implementar duas bibliotecas da estrutura de dados pilha, uma utilizando vetor e outra utilizando ponteiros (alocação dinâmica de memória)
- Praticar o encapsulamento: as duas bibliotecas devem ter as mesmas interfaces (métodos com mesma assinatura) de forma que os programas principais não precisem ser alterados quando a biblioteca importada é alterada.

2. Requisitos Funcionais do Programa de transformação de decimal para binário

A transformação de números da base decimal para a binária envolve seguidas divisões inteiras por 2 armazenando os restos dessas divisões, conforme exemplificado em sala e já estudado em outras disciplinas.

O programa a ser desenvolvido deve solicitar que o usuário digite um número decimal inteiro positivo para transformar em binário ou um número não positivo para finalizar a execução. Caso o usuário digite um número positivo, o programa deve convertê-lo para base binária e solicitar a digitação de um novo número, ficando nesse laço até que seja digitado um número não positivo.

O programa deve utilizar pilhas para fazer a transformação.

3. Requisitos não funcionais:

Devem ser implementadas duas bibliotecas de pilha (uma com vetores e outra com ponteiros), contendo ao menos os seguintes métodos:

- Iniciar pilha: no caso da implementação com vetores esse método deve receber, o tamanho desejado.
- Empilhar: na implementação com vetor deve-se tratar a possibilidade de overflow.
- Desempilhar na implementação com vetor deve-se tratar a possibilidade de underflow.
- Consultar elemento do topo da pilha;
- Imprimir todos elementos da pilha;

Os métodos das duas bibliotecas devem ter assinaturas iguais de forma que as bibliotecas possam ser substituídas sem afetar o código fonte. A única exceção será o método de iniciar pilha, já que na implementação com vetores deve-se informar o tamanho desejado para a pilha.

4. Critérios de avaliação

Além da adequação aos requisitos acima listados, serão considerados na avaliação:

- Modularidade do código(divisão do código em funções e procedimentos);
- Legibilidade do código: endentação, padrões de nomes de variáveis;
- Qualidade dos comentários.

5. Regras

- O trabalho deverá ser desenvolvido individualmente.
- O trabalho será dividido em duas etapas: (i) implementação com vetores e (ii) implementação com alocação dinâmica
- A primeira etapa do trabalho deve ser entregue através do Moodle até as 23:55 horas do dia 28/03. Os códigos dos programas desenvolvidos devem ser postados, em um arquivo compactado (.zip) e o nome do arquivo deve ser trab1_etapa1_nomeAluno.zip. Por exemplo, caso o trabalho tenha sido feito por João, deve ser enviado um arquivo com o nome trab1_etapa1_joao.zip.
- A segunda etapa do trabalho deve ser entregue através do Moodle até as 23:55 horas do dia 11/04. Os códigos dos programas desenvolvidos devem ser postados, em um arquivo compactado (.zip) e o nome do arquivo deve ser trab1_etapa2_nomeAluno.zip. Por exemplo, caso o trabalho tenha sido feito por João, deve ser enviado um arquivo com o nome trab1_etapa2_joao.zip.
- Em cada entrega devem estar compactados no .zip o programa principal (.c) e a biblioteca de pilhas (.c e .h).
- O trabalho (duas etapas) vale 15 pontos.
- Trabalho copiado ou entregue fora do prazo vale 0(zero).

O professor poderá realizar arguição individual sobre o trabalho a fim de averiguar a compreensão de cada aluno sobre o trabalho realizado podendo a nota do trabalho ser substituída pela nota na arguição. A arguição será especialmente importante em casos de suspeita de cópias.