TRABALHO

DESCRIÇÃO

Em computação, **o tipo abstrato de dado (TAD)** é uma especificação de um conjunto de dados e operações que podem ser executadas sobre esses dados.

O trabalho consiste em inicialmente, implementar os tipos abstratos de dados especificados nos itens 1, 2 e 3. Para cada um deles, crie um arquivo .c e um arquivo .h (arquivo cabeçalho). Atenção: leia todo o trabalho antes de começar a implementar.

Em seguida, escolha um dos problemas (4 ou 5) e implemente-o. Após escolher o problema (4 ou 5), você deverá utilizar um dos tipos (TAD) especificados nos itens (1, 2 ou 3) para resolver o problema. Para isso, já especifique-o de acordo com a sua escolha. Você também deverá entregar um arquivo .c para o exercício escolhido, correspondente ao seu programa (*main*).

1) Especificar e implementar o TAD lista **ordenada** usando alocação dinâmica/encadeada (sem cabeça). Operações que o TAD deve contemplar:

- Inicializar lista;
- Verificar se lista é vazia;
- Inserir elemento na lista, de forma que lista fique ordenada;
- Remover elemento da lista;
- Remover na posição: remove o elemento de uma posição ***pos*** específica, caso ela exista na lista. Se a posição não existir, a operação "falha".
- Tamanho: retorna o número de elementos da lista;
- Média: retorna a média aritmética simples dos elementos da lista;
- Iguais: recebe duas listas ordenadas e verifica se elas são iguais;
- Intercalar: recebe duas listas ordenadas (L1 e L2) e retorna uma nova lista (L3) com os elementos das duas listas intercalados conforme a ordenação.

2) Implementar o TAD lista **não ordenada** usando alocação dinâmica com encadeamento CÍCLICO (sem cabeça). Operações que o TAD deve contemplar:

- Inicializar lista;
- Verificar se lista é vazia;
- Inserir no final;
- Remover no início;
- Inserir elemento: inserir o elemento no início;
- Remover elemento: remover o elemento *x* na posição que ele estiver;

- Tamanho: retorna o número de elementos da lista;
- Maior: retorna o maior elemento da lista;
- Concatena: recebe duas listas não ordenadas L1 e L2 e retorna uma nova lista L3 com os elementos de L1 seguidos dos elementos de L2.

3) Implementar o TAD lista **ordenada** usando alocação dinâmica com encadeamento duplo (sem cabeça). Operações que o TAD deve contemplar:

- Inicializar lista;
- Verificar se lista é vazia;
- Inserir elemento na lista, de forma que lista fique ordenada;
- Remover elemento;
- Remover todos: remove todas as ocorrências de um elemento *x* em uma lista;
- Tamanho: retorna o número de elementos da lista;
- Iguais: recebe duas listas ordenadas e verifica se elas são iguais.

ESCOLHA O PROBLEMA 4 OU 5 ABAIXO PARA RESOLVER.

4) Implementar o problema de Josephus utilizando o TAD lista.

Problema: um grupo de soldados está cercado pelo inimigo e existe apenas um cavalo para a fuga. Decidiu-se que o soldado que se salvará será definido na sorte, independente da patente. O processo de escolha seria por eliminação, sendo que o último soldado a ser selecionado se salvaria. O processo de eliminação consiste em: organizar os soldados em volta da fogueira; escolher um soldado para iniciar a contagem e sortear um único número. Ao final da contagem, o soldado escolhido seria eliminado e o processo seria reiniciado a partir do próximo soldado, até só restar o soldado ganhador.

Entradas:

- Nomes dos soldados que estão cercados
- Opção de início de contagem:
  (1) Iniciar contagem a partir do primeiro soldado da lista.
  (2) Iniciar contagem a partir de um soldado sorteado aleatoriamente da lista.
  (3) Informar o nome do soldado para iniciar a contagem.

Saídas:

- No caso da opção de contagem (2), imprimir o nome do soldado sorteado.
- Imprimir o número sorteado.
- Imprimir os nomes dos soldados eliminados, na ordem de eliminação.
- Imprimir o nome do Sobrevivente.

OBS: o aluno deve escolher a MELHOR técnica de implementação para o problema.

5) Implementar um programa para manipulação de polinômios do tipo P(x) =

$$a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0$$

Para tal, o polinômio deve ser armazenado através de uma lista ordenada, sendo que cada elemento i da lista deve armazenar o k-ésimo termo do polinômio (diferente de 0), e deve conter o valor k da potência de x (inteiro) e o coeficiente a_k correspondente (inteiro). Por exemplo, o polinômio $P(x) = 3x^6 - 2x^4 + x^3 + 5x + 2$ deve ser representado pela lista:

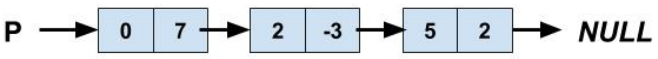
Fica a critério do aluno a escolha da técnica de implementação, se o encadeamento cíclico/simples/duplo. Deve ser criada uma interface que permita ao usuário executar qualquer uma das operações abaixo, a qualquer momento:

- **Inicializar um polinômio**. Fazer $P(x)= 0x^0$.

- **Inserir um novo termo $a_k x^k$ no polinômio existente**. Se já existe um termo $a_{k'} x^k$ no polinômio o valor do coeficiente do novo termo a_k deve ser adicionado ao já existente $a_{k'}$, assim:

$$P(x) = a_n x^n + ...(a_k + a_{k'})x^k ...+ a_1 x^1 + a_0$$

- **Imprimir P(x)**. Se o polinômio for $P(x) = 2x^5 - 3x^2 + 7$, a representação interna será:



A seguinte expressão deverá ser visualizada na tela: +7 -3x^2 +2x^5

- **Eliminar o termo associado à *k*-ésima potência.** Se o polinômio atual for $P(x) = 2x^5 - 3x^2 + 7$ (representação interna no exemplo acima) e o usuário solicitar a remoção do termo associado à potência 2 de *x*, o polinômio resultante será $P(x) = 2x^5 + 7$ e o nó referente à potência 2 de *x* deve ser liberado resultando na estrutura:

- **Reinicializar um polinômio**. Fazer P(x)= $0x^0$ e liberar os nós do P(x) anterior.

- **Calcular o valor de P(x) para um valor de x solicitado**. Por exemplo, se o polinômio atual for P(x) = $3x^6 - 2x^4 + x^3 + 5x + 2$ e o usuário solicitar o cálculo de P(x) para x = 2, o valor de P(2) deve ser calculado: P(2) = $3(2)^6 - 2(2)^4 + (2)^3 + 5(2) + 2$= 3x64 – 2x16 + 1x8 + 5x2 + 2 = 180 e o valor 180 deve ser apresentado na tela.

Entrega

Instruções para entrega do seu trabalho:

1. **Cabeçalho**
   Seu trabalho deve ter um cabeçalho com o seguinte formato:

   /
   *** Nome do(a) estudante**
   *** Trabalho**
   *** Professor(a): Nome do(a) professor(a)**
   ***/**

2. **Compilador**

   Os(as) professores(as) usam o compilador da linguagem C da coleção de compiladores GNU gcc , com as opções (de compilação) **-Wall -ansi -pedantic** para corrigir os programas. Se você usar algum outro compilador para desenvolver seu programa, antes de entregá-lo verifique se o seu programa tem extensão .c, compila sem mensagens de alerta e executa corretamente. Qualquer observação/execução diferente desta, notifique o professor.

3. **Forma de entrega**

   A entrega será realizada diretamente no Sistema de Suporte a Disciplinas (Moodle), na disciplina de Algoritmos e Programação II. Para entrega do trabalho, você deve estar cadastrado na página http://ead.facom.ufms.br na disciplina Algoritmos e Programação II. Após abrir uma sessão digitando seu *login* e sua senha, vá até o tópico – Trabalhos, e escolha “Entrega do trabalho”. Você pode entregar o trabalho quantas vezes quiser até às **23 horas e 55 minutos** do dia **17 de JUNHO de 2018**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais

aceitos trabalhos.

4. **Atrasos**

   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

5. **Erros**

   Trabalhos com erros de compilação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação.

6. **O que entregar?**

   Você deve entregar um único arquivo compactado contendo APENAS os seus programas fontes, como por exemplo, michael_jackson.c, michael_jackson.h . NÃO entregue qualquer outro arquivo, tal como o programa executável, já compilado.

7. **Verificação dos dados de entrada**

   Não se preocupe com a verificação dos dados de entrada do seu programa. Seu programa não precisa fazer consistência dos dados de entrada. Isto significa que se, por exemplo, o seu programa pede um número entre 1 e 10 e o usuário digita um número negativo, uma letra, um cifrão, etc, o seu programa pode fazer qualquer coisa, como travar o computador ou encerrar a sua execução abruptamente com respostas erradas.

8. **Arquivo com o programa fonte**

   Seu arquivo contendo o programa fonte na linguagem C deve estar bem organizado. Um programa na linguagem C tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. Não esqueça da documentação de seu programa.

9. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE . Cada estudante tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não faça o trabalho em grupo e não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, ao vivo ou durante a aula, monitoria e com o professor, mas NÃO copie o programa!

   Trabalhos considerados plagiados terão nota ZERO.