Quiz 10 - Estruturas de Dados Flexíveis

- Entrega 23 abr em 10:30
- Pontos 100
- Perguntas 7
- Disponível 23 abr em 8:50 - 23 abr em 10:30 1 hora e 40 minutos
- Limite de tempo Nenhum

Instruções

Este quiz aborda Classe Autorreferencial e Lista flexível. Ele tem 7 questões de múltiplas escolha sendo que a primeira vale 10 pontos e as demais, 15 pontos cada. Após o preenchimento de uma questão, o aluno não tem a opção de retorno à mesma. Este trabalho deve ser efetuado sem consulta.

Este teste foi travado 23 abr em 10:30.

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	10 minutos	60 de 100

Pontuação deste teste: 60 de 100
Enviado 23 abr em 10:24
Esta tentativa levou 10 minutos.

Pergunta 1
0 / 10 pts

As listas são estruturas de dados que podem ser implementadas de forma linear usando um *array* ou de forma flexível efetuando alocação de memória sob demanda.

Considerando o método remover no início das duas implementações, avalie as asserções que se seguem:

I. Tal método na implementação flexível é computacionalmente mais barato que na linear.

PORQUE

II. Na lista linear, o método em questão desloca os elementos do início para o final do array. Na

flexível, ele apenas faz algumas mudanças de referência.

A respeito dessas asserções, assinale a opção **correta**:

- As asserções I e II são proposições falsas.
- A asserção I é uma proposição falsa, e a asserção II é uma proposição verdadeira.

Resposta correta

- A asserção I é uma proposição verdadeira, e a asserção II é uma proposição falsa.
- As asserções I e II são proposições verdadeiras, mas a segunda não é uma justificativa correta da primeira.

Você respondeu

- As asserções I e II são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.

A afirmação I é verdadeira e a II é falsa.

A primeira é verdadeira porque na lista linear, o método em questão desloca os elementos do final para o início do array. Na flexível, o método para remover no início apenas faz algumas mudanças de referência.

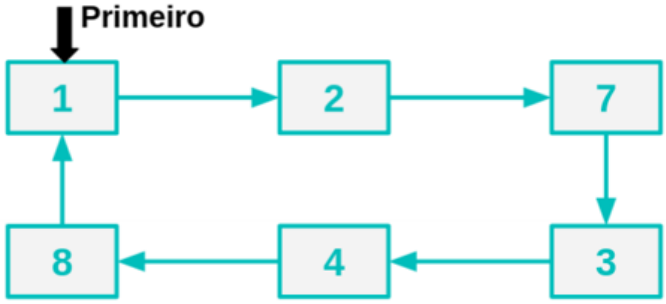
A segunda afirmação é falsa porque ela coloca a ordem inversa do deslocamento dos elementos.

Pergunta 2
15 / 15 pts

Uma classe é dita autorreferencial quando ela faz referência a objetos da própria classe.

Considere a classe Célula e a figura abaixo:

```
class Celula {
  public int elemento;
  public Celula prox;
  public Celula() {
    this.elemento = 0;
    this.prox = null;
  }
}
```



Sobre tal classe e figura, avalie as afirmações a seguir:

I) O comando “primeiro.prox.prox” faz referência à caixinha que contém o 7.

II) O código abaixo remove a caixinha do 2

```
Celula tmp = primeiro.prox;
primeiro.prox = tmp.prox.
tmp.prox = null;
```

III) O código abaixo soma todos os elementos encadeados:

```
int soma = 0;
for(Celula i = primeio; i.prox != primeiro; i = i.prox, soma += i.elemento);
```

É **correto** o que se afirma em:

- I, II e III.
- I e III, apenas.

Correto!

- I e II, apenas.

A afirmação I é verdadeira dado que o primeiro aponta para a caixinha do 1; o primeiro prox para a do 2 e o segundo para o do 7.

A afirmação II é verdadeira dado que o primeiro comando faz com que tmp faça referência para a caixinha do 2. O segundo muda a referência do prox da primeira caixinha para a do 7. O terceiro comando desconecta a caixinha do 2 de nossa sequência.

A afirmação III é falsa porque o 8 (da última caixinha) não é somado.

- II e III, apenas.

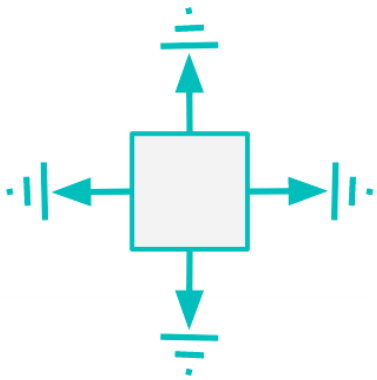
Pergunta 3
0 / 15 pts

A classe autorreferencial faz referência a objetos da própria classe como mostrado no exemplo abaixo:

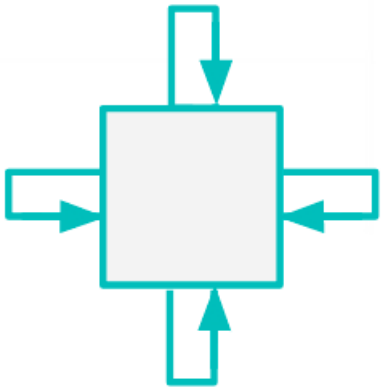
```
class Cidade{
  public string cidade;
  public Cidade norte, sul, leste, oeste;
  public Cidade(){
    norte = sul = leste = oeste;
    cidade = "";
  }
  public Cidade(Cidade norte, Cidade sul, Cidade leste, Cidade oeste){
    this.norte = norte;
    this.sul = sul;
    this.leste = leste;
    this.oeste = oeste;
  }
}
```

Sobre a classe Cidade, avalie as afirmações a seguir:

I) Dado o comando “Cidade tmp = new Cidade()”, podemos representar graficamente tmp como na figura abaixo.



II) Dado o comando Cidade tmp = new Cidade(new Cidade(), new Cidade(), new Cidade(), new Cidade()), podemos representar o objeto referenciado por tmp como na figura abaixo.



É **correto** o que se afirma em:

Resposta correta

- As duas afirmações são falsas.
- I, apenas.

Você respondeu

- I e II.

A afirmação I é falsa porque tmp é uma referência para o objeto criado por new Cidade(). A figura mostra o objeto, não a referência.

A afirmação II é falsa porque cada referência (norte, sul, leste e oeste) do objeto referenciado por tmp faz referência para um objeto distinto criado no construtor.

- II, apenas.

Pergunta 4
15 / 15 pts

A manipulação eficiente de dados é essencial para o desenvolvimento de software robusto e performático. Estruturas de dados, como listas, desempenham um papel fundamental nesse processo, permitindo o armazenamento e a recuperação de informações de maneira organizada. Entre as várias estruturas de dados, as listas flexíveis, ou listas encadeadas, destacam-se por sua flexibilidade na gestão de memória e capacidade de crescimento dinâmico. Uma lista flexível é composta por uma sequência de nós, onde cada nó contém um dado e uma referência ao próximo nó. Essa estrutura permite inserções e remoções eficientes em qualquer ponto da lista, sem a necessidade de realocações extensivas como em arrays. Em particular, o método InserirInicio, que adiciona elementos no começo da lista, exemplifica bem a agilidade das listas flexíveis em adaptar-se a mudanças dinâmicas no conjunto de dados. Com relação ao comentado, avalie as afirmações a seguir.

I. Em aplicações nas quais a ordem de inserção dos elementos precisa ser preservada pode-se utilizar o método InserirIncio e os de remover no início, fim ou em qualquer posição.

II. A utilização de InserirInicio em uma lista flexível é mais custosa em termos de tempo de execução do que adicionar elementos em um array.

III. InserirInicio permite adições rápidas e eficientes, especialmente útil em cenários onde o tamanho final da lista não é conhecido antecipadamente.

É correto o que se afirma em:

- I, apenas.
- I, II e III.
- I e II, apenas.
- III, apenas.

Correto!

- I e III, apenas.

I. Correta. O método InserirInicio garante que o último elemento inserido seja o primeiro na lista, o que é crucial para manter a ordem cronológica das inserções, especialmente em aplicações que dependem dessa ordem para funcionamento correto, como em pilhas de execução ou gerenciamento de transações. Nesse caso, as remoções podem acontecer em qualquer ordem garantindo a ordem de inserção.

II. Incorreta. Diferentemente do afirmado, InserirInicio em uma lista flexível tende a ser mais rápido do que em um array, pois não requer o deslocamento de todos os outros elementos já presentes na estrutura. Em um array, inserir um elemento no início significa deslocar todos os elementos subsequentes, o que aumenta o tempo de execução conforme o tamanho do array.

III. Correta. A afirmação destaca uma das principais vantagens das listas flexíveis: a capacidade de adicionar elementos de forma eficiente sem conhecer o tamanho final da lista. Isso é especialmente vantajoso em situações como a coleta de dados em tempo real, onde o volume de informações pode variar significativamente.

Pergunta 5
0 / 15 pts

As listas são estruturas de dados utilizadas na programação de computadores para armazenar elementos. Duas formas tradicionais para implementar tal estrutura de dados são a sequencial e

a flexível. A linguagem C# tem as classes List<T> e LinkedList<T> que implementam as listas sequencial e flexível, respectivamente. O código abaixo cria um objeto de cada uma dessas classes e, em cada uma, insere os números 0 a 1023 e os mostra na tela.

```
    LinkedList<int> flexivel = new
LinkedList<int>();
    List<int> sequencial = new
List<int>();

    for (int i = 0; i < 1023; i++){
        sequencial.Add(i);
        flexivel.AddLast(i);
    }
    foreach (int i in sequencial)
 Console.WriteLine(i);
    foreach (int i in flexivel)
Console.WriteLine(i);
```

Considerando o código acima, avalie as afirmações abaixo:

I) O custo para acessar a milésima posição na lista sequencial é menor que na flexível.

II) O custo para inserir mais um elemento no final do código acima na lista flexível é menor que na sequencial.

III) O custo de comparação entre elementos para pesquisar na lista flexível é menor que na sequencial.

É correto o que se afirma em:

Você respondeu

- I, apenas.

Resposta correta

- I e II, apenas.
- I, II e III.
- II e III, apenas.
- I e III, apenas.

A afirmação I é verdadeira porque na lista sequencial podemos acessar diretamente qualquer posição da lista. Na flexível, precisamos percorrer a mesma desde o início

A afirmação II é verdadeira porque a inserção na lista flexível implica em apenas criar mais uma célula. Na classe List<T> do C# (lista sequencial), o tamanho inicial da lista é zero e, após a primeira inserção, ele se torna quatro. Em seguida, sempre que a capacidade máxima da lista é alcançada, o C# dobra a quantidade de elementos armazenados, realocando mais espaço de memória. No final do código acima, cada lista tem 1024 elementos e a inserção de um novo elemento na lista sequencial implica em aumentar sua capacidade para 2048 elementos.

A afirmação III é falsa porque ambas as listas tem o mesmo custo de pesquisa.

Pergunta 6
15 / 15 pts

A habilidade de gerenciar e manipular dados eficientemente é um componente crucial na formação de um profissional competente. As listas flexíveis ou listas encadeadas, permitem operações dinâmicas de manipulação de dados devido à sua estrutura baseada em nós, que armazenam dados e referências ao próximo elemento. Essa estrutura facilita a inserção e a remoção de dados em qualquer posição da lista sem a necessidade de realocação de todos os elementos, como ocorre em arrays. O seguinte trecho de código exemplifica o método Inserir(int elemento, int posicao) para adicionar um elemento em uma posição específica de uma lista flexível:

```
public void Inserir(int elemento, int posicao) {
    if (posicao < 0 || posicao > tamanho) {
        throw new IndexOutOfRangeException("Posição
inválida");
    }
    Celula nova = new Celula(elemento);
    if (posicao == 0) {
        nova.prox = primeiro;
        primeiro = nova;
    } else {
        Celula atual = primeiro;
        for (int i = 0; i < posicao - 1; i++) {
            atual = atual.prox;
        }
        nova.prox = atual.prox;
        atual.prox = nova;
    }
    tamanho++;
}
```

Considerando esse contexto, avalie as asserções a seguir e a relação proposta entre elas.

I) O método Inserir(int elemento, int posicao) permite a inserção de um novo elemento em qualquer posição válida, incluindo no início ou no fim da lista, ajustando-se dinamicamente ao tamanho atual da lista.

PORQUE

II) A flexibilidade para inserir elementos em qualquer posição sem a necessidade de deslocar fisicamente todos os elementos existentes na memória torna as listas flexíveis particularmente úteis em aplicações que requerem modificações frequentes e imprevisíveis dos dados.

A respeito dessas asserções, assinale a opção correta.

- A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.
- A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.

Correto!

- As asserções I e II são proposições verdadeiras, e a II é uma justificativa correta da I.
- As asserções I e II são proposições falsas.
- As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa correta da I.

As asserções I e II são ambas verdadeiras.

A I é correta porque o método descrito realmente permite a inserção em qualquer posição controlada pelo índice, inclusive no início ou no fim da lista, adaptando-se ao tamanho atual sem restrições de espaço pré-alocado, como mostrado no código.

A II também é verdadeira e **serve como justificativa correta para a I**, pois uma das principais vantagens das listas flexíveis é a capacidade de inserir elementos em qualquer ponto sem necessitar da realocação de todos os elementos já presentes, o que é fundamental em ambientes dinâmicos onde o volume e a ordem dos dados podem mudar rapidamente.

Pergunta 7
15 / 15 pts

O estudo de estruturas de dados como as listas flexíveis é crucial para a compreensão de como os dados podem ser gerenciados e manipulados de maneira eficiente. As listas flexíveis, em particular, oferecem uma abordagem dinâmica na gestão de dados através de nós que se ligam sequencialmente. Cada nó contém uma informação e uma referência ao próximo nó, permitindo operações de inserção e remoção com grande flexibilidade. Essas operações podem ser implementadas de forma que impactem minimamente o desempenho global do sistema, destacando-se em contextos onde alterações frequentes são necessárias. Considerando esse contexto, avalie as asserções a seguir e a relação proposta entre elas.

I) O método RemoverInicio() em listas flexíveis é crucial para a eficiência de algoritmos que processam dados em tempo real.

PORQUE

II) Remover o primeiro elemento de uma lista flexível dispensa a realocação dos demais elementos como observado em estruturas baseadas em *arrays*.

A respeito dessas asserções, assinale a opção correta.

- As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa correta da I.
- A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.
- A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.
- As asserções I e II são proposições falsas.

Correto!

- As asserções I e II são proposições verdadeiras, e a II é uma justificativa correta da I.

As duas asserções são verdadeiras.

A I é correta porque a capacidade de remover rapidamente o primeiro elemento de uma lista sem afetar os demais elementos é essencial em aplicações que requerem processamento e resposta em tempo real, como sistemas de controle de tráfego ou gerenciamento de eventos em tempo real.

A II também é verdadeira, pois uma das vantagens das listas flexíveis sobre as arrays é que a remoção de um elemento no início da lista não exige a realocação dos demais elementos, apenas

a atualização de uma referência no nó antecedente, o que é uma operação de custo constante.

A afirmação II justifica a primeira porque o método RemoverInicio() nas listas flexíveis não demanda as realocações do mesmo nas listas lineares.

Pontuação do teste: 60 de 100