SEMANAS:
- INICIO/APRESENTAÇÃO
- INTRODUÇÃO AOS ALGORITMOS E ESTRUTURAS DE DADOS
- ESTRUTURAS DE CONTROLO DE FLUXO EM C#
- CONCEITO DE CLASSE E OBJETO
- SEPARAÇÃO ENTRE INTERFACE PÚBLICA E IMPLEMENTAÇÃO PRIVADA
- ESTRUTURAS DE DADOS – INTRODUÇÃO
- ANÁLISE DE ALGORITMOS E COMPLEXIDADE
- ALGORITMOS DE ORDENAÇÃO
- ALGORITMOS DE ORDENAÇÃO (PARTE 2)
- PROVA DE AVALIAÇÃO
- LISTAS E FILAS
- TABELAS DE HASH
- TRATAMENTO DE COLISÕES
- ÁRVORES BINÁRIAS
- TIPOS GENÉRICOS E TRATAMENTO DE EXCEPÇÕES
- PROVA DE AVALIAÇÃO

Algoritmos e Estruturas de Dados

CTeSP - Tecnologias e Programação de Sistemas de Informação 2020/21

Docente: Ricardo Henriques.

Sumário:

- Pilhas (*stack*)
- Listas (*queue* & deque)

Breve introdução à teoria dos conjuntos

A noção de conjunto está desde sempre presente na linguagem comum, desde bem cedo, mesmo ao nível do ensino básico, ou mesmo antes. Noções como um rebalho, um cacho de uvas, ou o conjunto de pontos de uma recta, fazem parte desse processo abstrato que mais tarde se formaliza, particularmente na matemática.

A noção de conjunto, tal como hoje entendemos, advém do matemático alemão Georg Cantor e assenta na relação entre elementos diferentes que são agrupados num todo, formando um conjunto. O seu trabalho no último quarto do séc XIX forneceu uma base conceptual e uma linguagem universal para a matemática da época, ao mesmo tempo que formulava uma teoria matemática do infinito.

Conjunto

> *É uma coleção de objectos de natureza qualquer, encarada como um todo, em que as suas partes são designados de elementos ou membros.*

Operações fundamentais que reconhecemos de utilização corrente são: a pertença de um elemento a um conjunto, a reunião de conjuntos, a interseção de conjuntos, o complemento, a cardinalidade, entre outros.

Coleções

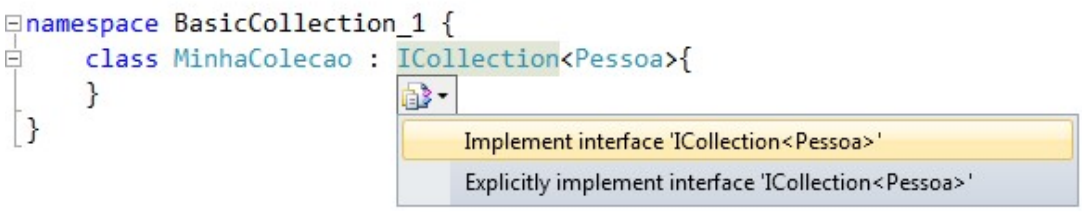
Uma coleção é um tipo de dados estruturado que armazena dados e dispõem de operações para a manipulação dos mesmos, como por exemplo: adição, remoção, alteração, consulta. As coleções podem ser divididas em duas grandes categorias: lineares e não lineares.

As lineares tem associada de alguma forma a noção de ordem e são estruturas de mais simples implementação. As coleções lineares podem ainda ser classificadas como permitindo o acesso direto, como é o caso de um *array*, ou sequencial, tendo já sido alvo do nosso estudo e avaliação.

As não lineares incluem estruturas como árvores e grafos, e podem ser classificadas como hierárquicas ou agrupadas.

Se desejar implementar a sua própria classe compatível com `ICollection` bastará utilizar o Visual Studio de forma semelhante à que se exibe para gerar a assinaturas dos métodos ↑

a implementar:



Listas lineares

Estruturas de acesso direto (em C#)

Em C# existem várias estruturas lineares disponíveis para o programador. Por exemplo, os *arrays* são não só um tipo primitivo, mas também uma classe. Os *arrays* podem ser usados para armazenar coleções lineares de elementos. De modo geral são estruturas estáticas no sentido em que o seu tamanho fica definido aquando da sua declaração. Adicionar um elemento num *array* é fácil, bastando para isso encontrar a próxima posição livre. Contudo inserir ou apagar um elemento num *array* "cheio" não é simples nem eficiente. Para obviar este esforço o .NET faculta por exemplo a classe `ArrayList`.

Outro tipo de coleção de acesso direto disponível no C# é o tipo *string*, a qual é composta por um conjunto de carateres que podem ser individualizados pelo seu índice. Contudo é também uma classe, o que coloca à disponibilidade do programador um conjunto de operações habituais sobre *strings*. As *strings* são imutáveis, significa isso que a modificação de uma *string* impõe a criação de uma nova cópia com as alterações da antiga, o que pode limitar o desempenho em determinadas ações da sua utilização.

Outro tipo de dados de acesso linear indexado é o tipo `struct` (*records*), o qual em C# evoluíu ao ponto de permitir definir dentro de si, modularmente, as operações que manipulam os seus dados, quase à semelhança de uma classe - porém sem poder haver derivações ou heranças.

Coleções de acesso sequencial

Pese embora este tipo de coleções, como o nome indica, não faça o acesso direto, tem como vantagem a não limitação do espaço pré-alocado. Assim, o espaço é proporcional aos elementos que compõem e teoricamente terão a possibilidade de expansão infinita.

As operações que podemos efetuar sobre uma lista linear incluem, entre as mais comuns, as seguintes:

- Aceder ao k-ésima nodo da lista para examinar/alterar algum campo do seu conteúdo;
- Inserir um novo nodo mesmo antes do k-ésimo nodo;
- Apagar o k-ésimo nodo;
- Combinar duas ou mais listas lineares numa única;
- Dividir uma lista linear em duas ou mais listas lineares;
- Obter uma cópia de uma lista linear;
- Determinar o número de nodos numa lista;
- Ordenar os nodos de uma lista numa forma ascendente/descendente com base em certos campos dos nodos;
- Procurar a ocorrência de nodos com um determinado valor num campo de conteúdo.

Uma aplicação informática raramente efetua as nove operações indicadas, sendo por isso recomendável que se represente a lista linear de forma mais conveniente para o tipo de operações que mais frequentemente se espera que ocorram. Nas listas lineares, cujas inserções, remoções e acesso aos valores ocorram maioritariamente no início ou no fim

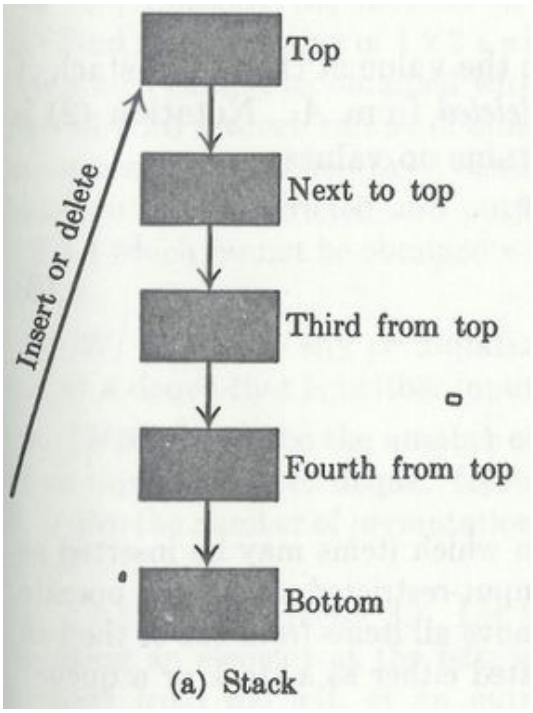
da lista tem nomes especiais:

Stack
É uma lista linear cujas inserções e remoções (e geralmente o acesso) são feitas na mesma extremidade da lista;

Queue
É uma lista linear cujas inserções são feitas numa extremidade e as remoções na extremidade oposta;

Deque
É uma lista linear onde as inserções e remoções (e geralmente o acesso) é feito numa ou noutra extremidade da lista.

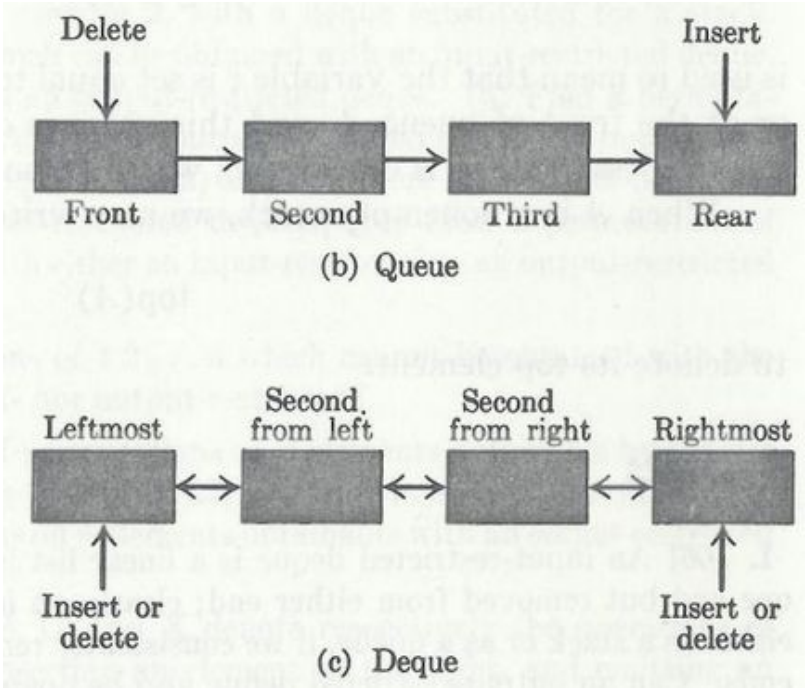
As seguintes ilustrações poderão ajudar a melhor compreender em abstrato estes conceitos.



Stack

É comum, dada a sua natureza, utilizarem-se algoritmos recursivos para o acesso a *stacks*.

É usada alguma terminologia particular associada a estas estruturas. Diz-se que se colocou um item no *top* da *stack* ou que se retirou um item do *top* da *stack*. Como se compreende, o *bottom* corresponde ao primeiro item colocado na *stack* e será o item menos acessível, dado que só poderá ser acedido retirando os outros que estão sobre ele. Outros termos como *push* e *pop* estão associados, respetivamente, à ação de colocar e retirar items da *stack*



Queue & Deque

Nas queues é usual usar termos como *front* e *rear* Os itens são colocados na *rear* e são retirados pela *front,* de forma análoga a uma fila de espera no atendimento ao público.

Em relação aos deques é usual usar termos como extremidades *left/right* ou *bottom/up*

Exercício 10:

Utilizando os conceitos elementares de programação por objectos desenvolva os exercícios.

Questão 1

Utilize a noção de lista ligada, para representar uma *stack* de números inteiros. Implemente as ações de:

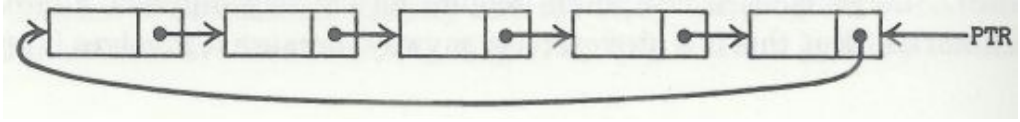
- *push* - introduzir um elemento.
- *peek* - validar se um determinado elemento está no topo, mas sem o retirar.
- *pop* - retirar o elemento de topo.
- *reverse* - obter uma réplica invertida da lista original.

Questão 2

Converta a sua solução de 1 para que os elementos da *stack* possam ser usados para guardar elementos da classe `Pessoa` (nome + data nascimento).

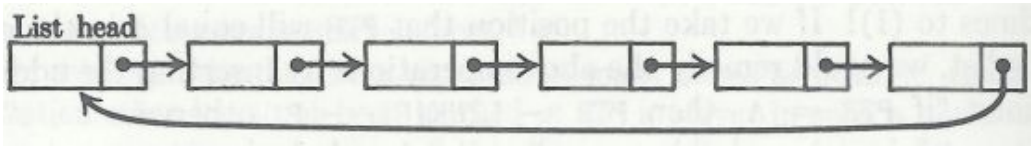
Listas circulares

Uma pequena alteração na forma de efectuar as ligações faculta uma forma diferente de implementação de uma lista ligada. Uma lista ligada circular, ou apenas lista circular, tem a propriedade do último nodo ligar de volta ao primeiro.



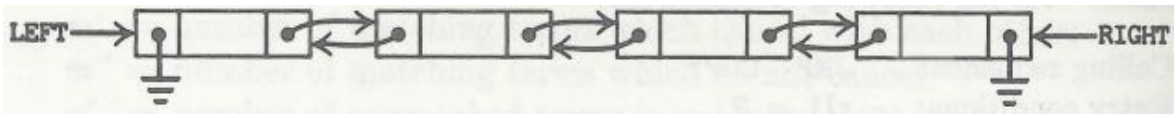
Nestas listas é possivel aceder a todos os elementos iniciando a partir de qualquer nodo. É possível ver uma lista circular utilizada não só para representar estruturas inerentemente circulares, mas também estruturas sequenciais. Uma lista circular é equivalente a uma lista simples com um apontador para a *front* e outro para a *rear*.

A questão natural que se coloca numa lista circular é como saber quando se atingiu o fim da lista. Com uma pequena variação na sua definição, em que é usado o *list head* poderá simplificar a definição de algumas operações. Este nodo especial funciona como ponto de paragem fica sempre presente, mesmo que a lista esteja vazia e, usualmente, associada a uma posição fixa da memória.

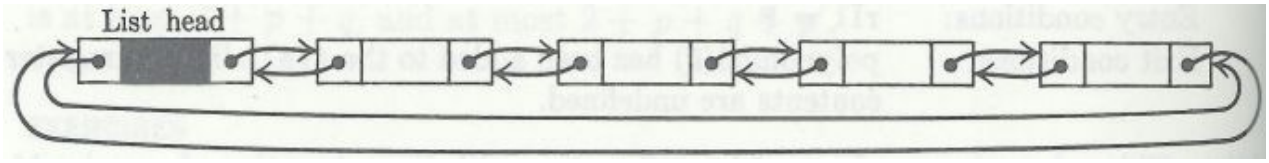


Listas duplas

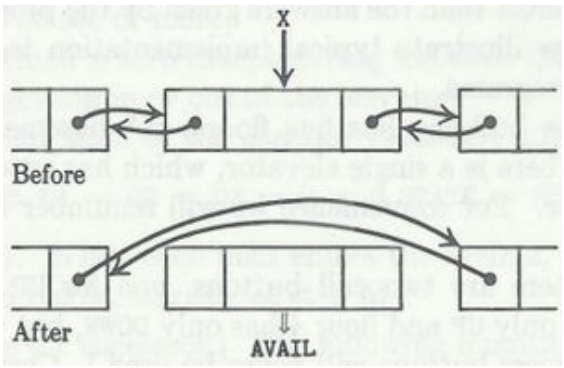
Para uma maior flexibilidade na manipulação de listas lineares é possível incluir duas ligações para cada nodo, apontando um para cada nodo adjacente.



Os apontadores *left* e *right* detém ligações para a parte esquerda e direita da lista, como se observa. Outra abordagem de simplificação é garantir o mesmo efeito através, mais uma vez, da introdução da *list head*



Uma lista ligada dupla objectivamente irá utilizar mais espaço de memória do que uma lista simples. Contudo esta desvantagem é amplamente compensada pelo facto de permitir a navegação bidirecional e, em especial, por simplificar significamente a operação de remoção. Onde antes era necessário possuir referências (apontadores) para os nodos anterior e seguinte ao nodo a eliminar, neste caso basta apenas aceder diretamente ao nodo e restabelecer as ligações, como é indicado:



Exercício 11 (exercício de grupo nº 4):

Considere a noção de antigo aluno de uma instituição de ensino superior, caracterizado por:

- *número* - antigo número mecanográfico na instituição;
- *nome* - nome completo;
- *dataNascimento* - data de nascimento (AAAA-MM-DD);
- *curso* - (primeiro) curso em que se inscreveu; utilize um enumerado para representar o curso.

Construa duas implementações utilizando listas ligadas. 1) a primeira deverá ser implementada por uma lista com inserção no fim. 2) a segunda implementação a lista deverá estar ordenada pelo número do aluno.

Em ambas deverá implementar as seguintes operações:

- Inserir um novo registo;
- Consultar (devolver) um *array* com todos os antigos alunos de um curso;
- Remover - com base no seu número;
- Listar todos os elementos da lista.
- Consultar todos os alunos que frequentaram um determinado curso.

Outros pormenores da entrega deste exercício serão indicados através da plataforma *moodle* associada à disciplina.

MRH, 2020/21