**Nome:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Nº\_\_\_\_\_\_\_\_**

# Parte Teórico-Prática

1. Distingue estrutura estática de estrutura dinâmica:

A estrutura estática tem um tamanho definido ao invés da estrutura dinâmica que não tem.

1. Indica uma situação para que na elaboração de um programa terias de utilizar a classe List em vez de ArrayList:

Basta ter que inserir uma struct

1. Preenche os espaços em branco:

Nas estruturas dinâmicas, utilizando a classe “List” a instrução \_ADD\_ serve para se inserir um novo elemento no vetor. Paralelamente a instrução \_remove\_\_\_\_\_ retira um elemento do vetor utilizando a classe “ArrayList”.

Já na estrutura FIFO a instrução \_\_peek\_\_\_ permite saber o conteúdo do 1ºelemento da estrutura, sendo que a instrução\_push\_\_\_\_\_\_\_\_ adiciona um elemento na pilha, enquanto a instrução \_\_\_dequeue\_\_\_\_\_\_\_\_\_ remove um elemento na fila.

1. Sendo este o vetor, definido num programa:

static List<**alugueres**> *vetor\_carros*=**new** List<**alugueres**>();

Preenche os espaços sublinhados , tendo presente que a função seguinte utilizaria este vetor:

static \_void\_ **inserir** (List<alugueres> x){  
            **alugueres** a;  
            **char** resp;  
            **int** i=0;

**\_\_\_\_do\_**{  
                Console.**WriteLine**("Insira o nome do cliente {0} ",(i+1));  
                \_\_\_a\_.*nome\_cli*=Console.**ReadLine**();  
                Console.**WriteLine**("Insira o ano de nascimento {0} ",(i+1));  
                \_\_a\_\_.*ano*=**int**.**Parse**(Console.**ReadLine**());  
                Console.**WriteLine**("Insira o custo do aluguer {0} ",(i+1));  
                \_a\_\_\_\_.*custo*=**double**.**Parse**(Console.**ReadLine**());                  
                \_\_\_X\_.**Add**(\_a\_\_\_);  
                Console.**WriteLine**("Deseja inserir um novo aluguer? ");  
                \_\_resp\_\_\_\_\_\_=**char**.**Parse**(Console.**ReadLine**());  
                i++;  
                Console.**WriteLine**();      
            } \_while\_\_ (resp=='S'|| resp=='s');  
        }

**Parte Prática (Mod6)**

**Nome:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Nº\_\_\_\_\_\_\_\_**

1. - Elabore um programa (**utilizando funções com os nomes que figuram nas alineas**) que permita ao utilizador (vendedor de seguros) realizar as seguintes tarefas:
2. Registe um cliente novo (**nome, a localidade, o telefone, nº da apólice, custo, data do contrato) (função registar\_novo);**
3. Apresentar o nome dos clientes que fizeram seguros numa certa data (dia, mês, ano) **(função seg\_data);**
4. Apresentar a percentagem de clientes pertencentes a uma certa localidade, introduzida pelo utilizador **(função perc\_loc);**
5. Calcular o total dos seguros vendidos considerando todos os clientes **(função total\_seg);**
6. Listar todos os dados do nome do cliente, sendo introduzido o seu número da apólice (**função list\_apolice)**;
7. Listar todos os dados de todos os registos por ordem alfabética do nome do cliente (**list\_todos**);

2 - Pretende-se que desenvolva um programa, que simule uma **fila de espera** de uma impressora, utilizando uma fila de **strings, em que cada string corresponderá ao nome do documento**.

O programa deverá implementar as seguintes tarefas:

1 – Enviar documento para a fila de espera;

2 – Imprimir próximo documento (informa o utilizador que está a imprimir o doc com o respetivo nome);

3 – Saber o número de documento em espera;

4 – Sair.

O menu com as opções deve ser sempre apresentado até que o utilizador escolha a opção 4 (Sair).

**Nota: Não é permitido o uso da internet, nem**Guarda os 2 programas, no teu ambiente de trabalho e espera que o professor faça a sua recolha.