Trabalho 1 - Gestão de uma Biblioteca (Parte 1)

Desenvolva uma aplicação em C++ para gestão de uma biblioteca. O sistema deve conter informação sobre livros, leitores, funcionários e empréstimos de livros.

Um empréstimo está associado a um leitor e diz respeito a um livro. Um livro pode ser emprestado por um período máximo de 1 semana ou não pode ser emprestado.

A informação sobre livros deve incluir, pelo menos, título, autores, ISBN, cota e nº de páginas.

Cada leitor não pode ter mais de 3 livros emprestados em simultâneo. Deve ser possível notificar o leitor por telefone ou email, caso os livros não sejam devolvidos dentro do prazo. Por cada dia de atraso na devolução de um livro, o leitor incorre numa multa de 0,25Eur por dia na primeira semana e de 0,50Eur por dia nas semanas seguintes.

Cada empréstimo é efetuado por um funcionário. Os funcionários podem ou não ser supervisores. Um supervisor é responsável por um ou mais funcionários. Um supervisor não pode ser responsável por outro supervisor. O número de funcionários sob a responsabilidade de cada um dos supervisores deve ser equilibrado.

Alguns requisitos obrigatórios são (pode e deve incluir outros que considere relevantes):

- Manutenção de livros
- Manutenção de funcionários
- Manutenção de leitores
- Manutenção de empréstimos
- Associar funcionários a supervisores
- Associar empréstimos a leitores
- Associar livros a empréstimos
- Associar funcionários a empréstimos
- … e outras associações que julgue necessárias

A aplicação deve permitir registar e gerir os empréstimos efetuados na biblioteca, assim como toda a informação relativa aos livros, leitores e funcionários.

Sugere-se como entidades a implementar: Leitor, Funcionario, Livro, Emprestimo, entre outras que julgue necessárias (considerar hierarquia de classes).

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de leitores
- Manutenção do conjunto de funcionários
- Manutenção do conjunto de empréstimos
- Listagens várias (totais ou parciais com critérios a definir pelo utilizador, com possível ordenação) de, por exemplo:
  - leitores
  - funcionários
  - livros
  - empréstimos efetuados, por leitor, por funcionário, por livro, por data, …
  - …

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 2 - Gestão de uma AppStore (Parte 1)

Desenvolva uma aplicação em C++ para gestão de uma AppStore que disponibiliza apps para smartphones. O sistema deve conter informação sobre apps, developers, clientes e vendas.

Uma app deve incluir, pelo menos, informação sobre preço, categoria e descrição.

Um cliente pode comprar uma ou mais apps em cada transação e pode obter um desconto de 5% fornecendo um código de desconto (voucher). O cliente pode ainda classificar uma app (de 1 (não gosto) a 5 (espectacular!)) e deixar um comentário.

Os developers podem ser individuais ou empresas. Para cada developer deve ser conhecido o nome e morada e, no caso de empresas, o nome oficial da empresa e o NIF. Cada developer pode publicar uma ou mais apps na AppStore. O developer pode deixar de vender uma app a qualquer momento, mas não deve ser perdido o histórico de vendas. Por cada app vendida, o developer recebe 80% do seu preço.

Cada cliente deve estar registado na AppStore e fornecer os seus dados pessoais.

Alguns requisitos obrigatórios são (pode e deve incluir outros que considere relevantes):

- Manutenção de clientes
- Manutenção de developers
- Manutenção de apps
- Manutenção de vendas
- Associar apps a developers
- Associar vendas a clientes
- Associar apps a vendas
- … outras associações que julgue necessárias

A aplicação deve permitir registar e gerir toda a informação relativa às vendas efetuadas na AppStore e quando foram efetuadas.

Sugere-se como entidades a implementar: Cliente, Developer, App, Venda entre outras que julgue necessárias (considerar hierarquia de classes).

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de clientes
- Manutenção do conjunto de developers
- Manutenção do conjunto de apps
- Manutenção do conjunto de vendas
- Listagens várias (totais ou parciais com critérios a definir pelo utilizador, com possível ordenação) de, por exemplo:
  - clientes
  - developers, individuais e/ou empresas
  - apps (e respectiva classificação)
  - vendas efetuadas, por cliente, por developer, por app, …
  - todas as apps adquiridas por um cliente
  - todas as apps publicadas por um developer…

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 3 - Gestão de um Clube Desportivo (Parte 1)

Desenvolva uma aplicação em C++ para gestão de um clube desportivo. O sistema deve conter informação sobre modalidades, jogadores, sócios, despesas e quotas.

Uma modalidade pode conter várias sub-modalidades, p.ex., a modalidade futebol pode conter as sub-modalidades futebol masculino, feminino e infantil.

Um jogador pode pertencer a uma ou mais (sub-)modalidades. Por exemplo, um jogador pode pertencer simultaneamente às equipas de pólo aquático e natação 100 metros.

As despesas mais importantes a gerir incluem despesas com pessoal externo, salários dos jogadores e deslocações dos jogadores a outros clubes para competições.

Um sócio paga uma quota por cada modalidade de que é adepto e beneficia de um desconto de 5% sobre a quota total, se for adepto de mais de três modalidades. As quotas devem ser pagas até ao dia 8 do mês seguinte a que dizem respeito, sob pena de sofrerem uma penalização de 5Eur por cada mês de atraso. Cada modalidade pode ter uma quota diferente.

Devem ser conhecidas as informações pessoais dos jogadores, sócios e pessoal externo, assim como o NIF.

Alguns requisitos obrigatórios são (pode e deve incluir outros que considere relevantes):

- Manutenção de jogadores
- Manutenção de modalidades
- Manutenção de sócios
- Manutenção de despesas
- Manutenção de quotas
- Associar jogadores a modalidades
- Associar sócios a modalidades
- … outras associações que julgue necessárias

Sugere-se como entidades a implementar: Jogador, Modalidade, Despesa, Socio, Quota entre outras que julgue necessárias (considerar hierarquia de classes).

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de jogadores
- Manutenção do conjunto de sócios
- Manutenção do conjunto de modalidades
- Listagens várias (totais ou parciais com critérios a definir pelo utilizador, com possível ordenação) de, por exemplo:
  - jogadores
  - modalidades e sub-modalidades
  - despesas, por data, com jogadores, pessoal externo e deslocações
  - quotas pagas e em atraso por sócios
  - …

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 4 – Reserva de bilhetes de avião (Parte 1)

Desenvolva uma aplicação em C++ para realização de reservas de bilhetes de avião numa companhia aérea. O sistema deve conter informação sobre passageiros, aviões, voos e reservas.

Os passageiros podem ser se dois tipos: passageiro normal ou passageiro com cartão. Os passageiros com cartão são passageiros já registados na companhia aérea e possuem um cartão associado sistema no qual está registado o nº médio de voos (*nmVoo*) efetuados por ano (pode incluir mais informação, se considerar útil). Quando da aquisição de um bilhete, o passageiro com cartão tem sempre uma redução relativa ao seu número médio de voos anual. Assim, o preço do bilhete para este passageiro é:

PrecoPCartao = Preco*(100-trunc(*nmVoo*))%.

A companhia aérea disponibiliza voos comerciais e voos alugados. Um voo comercial é o mais usual, em que a companhia vende os bilhetes do voo individualmente. Num voo alugado, um único passageiro compra todos os bilhetes desse voo.

Alguns requisitos obrigatórios são (pode e deve incluir outros que considere relevantes):

- Manutenção de passageiros (ou clientes)
- Manutenção dos aviões
- Manutenção de voos
- Associar aviões e voos (um avião é usado em mais que um voo)
- Associar passageiros e voos (um passageiro pode efetuar mais que um voo; um voo comercial inclui mais que um passageiro). Isto é, manutenção de reservas.
- … outras associações que julgue necessárias

A aplicação deve permitir registar e gerir toda a informação relativa a quais reservas de bilhetes foram efectuadas pelos passageiros.

Considere o conjunto de algumas das entidades implementadas: Passageiro (entidade no topo da hierarquia de passageiros), Aviao, Voo (entidade no topo da hierarquia de voos), Reserva, entre outras.

Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de passageiros
- Manutenção do conjunto de aviões
- Manutenção do conjunto de voos
- Manutenção do conjunto de reservas
- Listagens várias (totais ou parciais com critérios a definir pelo utilizador, com possível ordenação) de, por exemplo:
    - passageiros (como opção, reservas efetuadas)
    - aviões
    - voos, voos comerciais, voos alugados (como opção, ocupação/disponibilidade)
    - reservas efetuadas por cliente, por data, ...
    - …

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 5 - Gestão Hoteleira (Parte 1)

Desenvolva uma aplicação em C++ para gestão de reserva de espaços (quartos e salas de reuniões) num hotel. Os espaços são reservados por um período a especificar pelo cliente e os preços da reserva dependem da época do ano, do dia da semana, entre outros fatores específicos do espaço.

Os espaços passíveis de reserva podem ser quartos ou salas de reuniões. Os quartos são simples ou duplos, localizados na frente ou nas traseiras, o que influencia o seu preço de reserva diário. As salas de reunião têm uma capacidade, equipamento de vídeo ou não, equipamento de audio ou não, o que influencia o seu preço de reserva diário.

Um funcionário do hotel é responsável por um conjunto de espaços.

O sistema deve conter e gerir informação sobre os espaços (quartos e salas de reunião), e sua ocupação. Deverá ainda incluir informação sobre clientes e funcionários.

Alguns requisitos obrigatórios (pode e deve incluir outros que considere relevantes):

- Manutenção de espaços (quartos e salas de reunião)
- Manutenção de clientes
- Manutenção de funcionários
- Associar espaços e funcionários (um funcionário é responsável por mais que um espaço)
- Associar espaços e clientes (um cliente pode reservar mais que um espaço; um espaço pode ser reservado por mais que um cliente em dias diferentes). Isto é, manutenção de reservas.
- … outras associações que julgue necessárias

Considere o conjunto de algumas das entidades implementadas: Espaco (entidade no topo da hierarquia de tipos de espaço), Cliente, Funcionario, Reserva, entre outras. Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de espaços
- Manutenção do conjunto de clientes
- Manutenção do conjunto de funcionários
- Manutenção do conjunto de reservas
- Listagens várias (totais ou parciais com critérios a definir pelo utilizador, com possível ordenação) de, por exemplo:
  - funcionários (como opção, espaços de que é responsável)
  - espaços (como opção, ocupação/disponibilidade)
  - clientes (como opção, reservas)
  - reservas (por período, cliente,...)
  - ….

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 6 - Gestão de informação da Volta a Portugal (Parte 1)

Desenvolva uma aplicação em C++ para gestão de informação relativa à Volta a Portugal em bicicleta.

Esta competição que anima as estradas portuguesas desde o Verão de 1927, é constituída por diversas etapas que as equipas participantes tentam ganhar. Uma equipa é caracterizada por uma designação, país e um conjunto de patrocinadores. Uma equipa possui vários ciclistas.

A classificação de um ciclista é obtida como o somatório dos tempos gastos nas etapas já realizadas. Deve considerar também o facto de um ciclista desistir a no decorrer da volta e não efetuar todas as etapas. O ciclista com menor tempo gasto, no somatório de todas as etapas, é considerado o vencedor da Volta a Portugal. A classificação de um diretor desportivo é a média das classificações obtidas pelos ciclistas da sua equipa.

O sistema deve registar e gerir de forma integrada toda a informação desportiva relativa à Volta, nomeadamente classificações e prémios obtidos do decorrer de cada etapa.

Alguns requisitos obrigatórios (pode e deve incluir outros que considere relevantes):

- Manutenção de equipas
- Manutenção de membros da equipa (ciclistas, diretor desportivo, treinadores, mecânicos e massagistas)
- Manutenção de etapas
- Manutenção de dados gerais da Volta (patrocinadores, classificações, prémios…)
- Associar equipas e membros
- Associar etapas e ciclistas (participantes, tempos e vencedores)
- … outras associações que julgue necessárias

Considere o conjunto de algumas das entidades implementadas: MembroEquipa (entidade no topo da hierarquia de membros de uma equipa), Equipa, Etapa, entre outras. Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção do conjunto de equipas
- Manutenção do conjunto de membros da equipa
- Manutenção do conjunto de etapas e prémios (montanha, juventude,…)
- Manutenção do conjunto de patrocinadores
- Listagens várias (totais ou parciais com critérios a definir pelo utilizador, com possível ordenação) de, por exemplo:
  - equipas (como opção, por patrocinador, por país, ...)
  - membros da equipa (ciclistas, diretores desportivos,…) e respetiva classificação (como opção, por equipa, por idade, ...)
  - etapas (como opção, realizadas ou não, de duração x,...)
  - ….

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 7 – Boleias Inteligentes (Parte 1)

O aumento de tráfego automóvel tem levado ao surgimento de modos alternativos de transportes, entre eles o conceito de "car pooling", onde grupos de pessoas compartilham itinerários numa mesma viatura. Neste contexto, redes sociais são excelentes instrumentos para identificação de pessoas, dentro de um mesmo círculo de pessoas conhecidas. Um exemplo encontra-se em http://www.caronetas.com.br/

Neste trabalho, desenvolva uma aplicação em C++ para gestão de um sistema de "car pooling", onde empresas e pessoas podem realizar um registo e especificar ofertas e necessidades de mobilidade, com a expectativa de encontrar companhias para os seus carros, ou boleias no carro de outros. As partilhas poderão ser programadas para um determinado período, como um semestre ou ano escolar, ou para um evento, no caso de um espetáculo ou jogo de futebol. Ao registarem-se, as empresas e as pessoas indicam as suas ofertas e as suas procuras. Empresas podem oferecer transporte coletivo, de diferentes origens para diferentes destinos. As origens e os destinos poderão estar guardados numa matriz de distâncias. Os grupos poderão ser formados por pessoas a um determinado raio de distância entre si (no caso das empresas), ou da origem do dono do veículo (no caso de veículo particular), que se estejam a deslocar para um destino próximo, também a um determinado raio de distância. O preço a pagar por quem utiliza as boleias a quem as oferece (empresa ou particular) poderá ser proporcional à distância percorrida, ao tipo de combustível e seu valor, e outros custos que quiser incluir.

O sistema deve conter e gerir informação sobre seus membros (empresas e particulares), e respetiva oferta e procura de boleias, assim como os acordos que se firmam, ao se formar grupos para um itinerário. Considere que as pessoas podem preferir integrar grupos onde já haja conhecidos. Algumas funcionalidades obrigatórias (pode e deve incluir outras que considere relevantes) são:

- Manutenção dos membros da rede (empresas e particulares)
- Manutenção de origens e destinos (de quem oferece e de quem procura boleia)
- Manutenção dos acordos firmados entre os elementos da rede, e dos custos devidos
- Manutenção de anúncios (de oferta e/ou de procura) de boleias
- Associar membros a origens e destinos
- Associar membros aos seus veículos e lugares disponíveis
- Associar membros a anúncios e acordos
- … outras associações que julgue necessárias

Algumas das entidades a serem implementadas: Membro (entidade no toop da hierarquia dos membros do sistema), Origem e Destino, Veiculo (com determinas características, como combustível, consumo), Anuncio (entidade no topo da hierarquia dos anúncios, que podem ser de oferta ou procura) entre outras. Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção de membros do sistema
- Manutenção do conjunto de anúncios (de oferta ou procura) de boleia
- Manutenção dos acordos firmados e dos custos associados
- Listagens várias (totais ou parciais com critérios a definir pelo utilizador, com possível ordenação) de, por exemplo:
  - Número de boleias oferecidas por destino
  - Custos a serem pagos a quem oferece boleia
  - Distâncias, entre origens e destinos
  - Anúncios por membros do sistema

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 8 – "Bike sharing" (Parte 1)

Transportes alternativos, mais amigos do ambiente, têm sido uma preocupação no âmbito do desenvolvimento das cidades do futuro. Algumas cidades, como Paris, Barcelona, Amsterdão, entre outras, oferecem sistemas de partilha de bicicletas, distribuídas em diferentes pontos da cidade, que podem ser alugadas por um determinado período de tempo. As bicicletas podem ser levantadas num ponto e devolvidas noutro ponto. O utilizador paga pelo tempo que a utilizou. Um exemplo pode ser consultado em http://fortworthbikesharing.org/

Neste trabalho, implemente uma aplicação em C++ para a gestão de uma cooperativa de partilha de bicicletas, onde as bicicletas disponíveis podem pertencer a várias empresas. O sistema deverá manter registo das empresas que disponibilizam bicicletas e dos utilizadores, que podem ser utilizadores registados ou ocasionais. Os utilizadores registados têm a possibilidade de pagar um determinado valor, pela utilização do serviço por mês, enquanto os utilizadores ocasionais pagam pelo tempo em que usam efectivamente a bicicleta, fornecendo o seu número de cartão de crédito no ato do levantamento, sendo-lhes cobrado o valor pelo tempo de uso, no ato da entrega. As empresas podem oferecer diferentes bicicletas, como elétricas, com ou sem cesto, com diferente número de velocidades, etc. O sistema também deve manter um registo de todos os pontos espalhados na cidade, onde as bicicletas podem ser levantadas e devolvidas. Cada ponto tem uma capacidade de lotação; entregas só poderão ser realizadas nos pontos com disponibilidade de vaga.

O sistema deve conter e gerir informação sobre as empresas associadas, e o histórico de utilização de bicicletas. Algumas funcionalidades obrigatórias (pode e deve incluir outras que considere relevantes):

- Manutenção das empresas fornecedoras
- Manutenção dos utilizadores (registados e ocasionais)
- Manutenção dos pontos de levantamento/entrega
- Manutenção de avarias
- Associar empresas a bicicletas
- Associar utilizadores registados a planos de pagamento
- … outras associações que julgue necessárias

Algumas das entidades a serem implementadas: Empresa (que fornece bicicletas), Utilizador (entidade no topo da hierarquia de utilizadores), Estacao (ponto de levantamento e entrega), Bicicleta (bicicletas podem ser de tipos e tamanhos variados, como adulto ou infantil), entre outras. Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção das empresas e das suas bicicletas
- Manutenção dos utilizadores registados, e do seu histórico de utilização
- Manutenção das bicicletas e do seu histórico de utilização
- Listagens várias (totais ou parciais com critérios a definir, com possível ordenação) de, por exemplo:
  - Número de horas de um utilizador em posse de uma bicicleta
  - Custos a serem cobrados aos utilizadores (registados ou ocasionais)
  - Frequência de utilizadores por empresa que fornece a bicicleta
  - … outras que considere pertinentes

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático

Trabalho 9 – Gestão de Projetos (Parte 1)

Uma empresa de investigação e desenvolvimento em informática realiza diversos projetos, cuja execução é realizada pelos seus colaboradores. Um projeto, basicamente, tem um prazo para a sua conclusão, e é constituído por um conjunto de tarefas. Cada tarefa tem um tempo estimado de conclusão e um esforço expectável, contabilizado pelo número de colaboradores a dedicarem horas de serviço àquela tarefa. Um exemplo de ferramenta de gestão de projectos é o Redmine: http://www.redmine.org/

Neste trabalho, implemente uma aplicação em C++ para a gestão dos projetos da empresa informática. A aplicação deverá conter informação de cada projeto, dos clientes que o contrataram, e dos colaboradores afectos ao projecto. Um colaborador poderá estar afecto a mais de um projecto ao mesmo tempo, desde que a soma do esforço dedicado a cada projecto não ultrapasse as suas horas de trabalho semanais, contratualizadas com a empresa. As várias tarefas de um projeto poderão necessitar de diferentes colaboradores, que poderão ser Programadores, Arquitectos, Gestores de Projecto, *Testers*, entre outros. À medida que tarefas vão sendo concluídas, os colaboradores ficam disponíveis para outros projectos, na proporção do esforço que haviam dedicado à tarefa concluída. Poderá haver dependências de tarefas, o que poderá ocasionar atrasos no projecto todo, caso haja atrasos em alguma tarefa da qual dependem outras. Cada colaborador, a depender da sua categoria, terá um custo associado às horas de trabalho no projecto. O custo de pessoal, do projecto, é a soma do custo relacionado ao esforço total dedicado ao projecto. Os projectos também podem ser de diferentes tipos, como manutenção, desenvolvimento, entre outros.

O sistema deve conter e gerir informação dos projectos da empresa, dos seus colaboradores, e dos seus clientes. Algumas funcionalidades obrigatórias (pode e deve incluir outras que considere relevantes) são:

- Manutenção dos projetos
- Manutenção dos clientes da empresa (que contratam os projectos)
- Manutenção dos colaboradores
- Associar clientes a projectos
- Associar tarefas a projectos
- Associar colaboradores a tarefas de projetos
- … outras associações que julgue necessárias

Algumas das entidades a serem implementadas: Projeto, Cliente, Colaborador (entidade no topo da hierarquia de colaboradores), Tarefa, entre outras. Usando ***estruturas de dados lineares***, deve implementar:

- Manutenção de clientes e projetos
- Manutenção de colaboradores e das tarefas que realizam em cada projeto
- Manutenção da execução do projeto e do progresso das tarefas
- Listagens várias (totais ou parciais com critérios a definir pelo utilizador, com possível ordenação) de, por exemplo:
  - Número de horas dedicadas a projetos, pelos colaboradores
  - Tempos médios de execução de determinadas tarefas
  - Tempos estimados de conclusão dos projetos
  - … outras que considere pertinentes

Notas:

1. Por "manutenção de dados" entende-se as operações básicas CRUD (*Create, Read, Update, Delete*)
2. O trabalho deve respeitar o que está descrito em "Notas relativas à implementação" no texto relativo aos "Enunciados e instruções" da parte 1 do trabalho prático