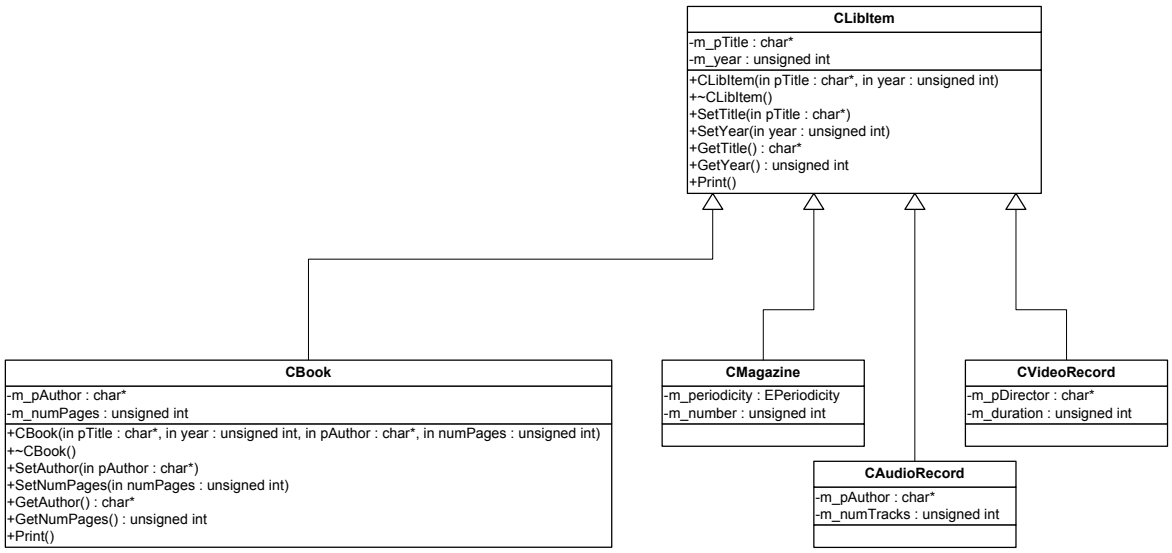
Aula 3

Objectivos: Utilização de herança simples e polimorfismo
Perceber as vantagens da programação orientada por objectos ao nível da extensibilidade e reutilização de código
Compreender a diferença entre uma relação de herança e uma relação de inclusão

Exercício 1

Pretende-se estender o programa da aula anterior de forma a suportar, além de livros, outros tipos de objectos normalmente existentes numa biblioteca, tais como revistas, registos de audio e registos de vídeo. No entanto, em vez de se definir um conjunto de classes independentes para cada um desses objectos, pretende-se tirar partido do mecanismo de herança das linguagens orientadas por objectos, colocando numa classe base (`CLibItem`) os atributos e os métodos comuns a cada um desses objectos (título e ano de edição).
Numa relação de herança entre duas classes, a base e a derivada, a segunda é uma especialização da primeira. Neste caso concreto um livro é também um item de uma biblioteca, mas além de um título e de um ano de edição possui também um autor e um número de páginas.
As classes derivadas `CBook`, `CMagazine`, `CAudioRecord` e `CVideoRecord`, que representam os vários tipos de objectos referidos acima, herdam o título e o ano de edição da classe base. Por outro lado, os atributos e métodos específicos de cada um são colocados nas respectivas classes derivadas da `CLibItem`.
O diagrama seguinte ilustra as relações de herança entre as classes `CLibItem`, `CBook`, `CMagazine`, `CAudioRecord` e `CVideoRecord`. São também mostrados os atributos de cada classe e, a título de exemplo, o construtor, o destrutor e os métodos pretendidos para as classes `CLibItem` e `CBook`. O método `Print` deve escrever no ecrã o valor de todos os atributos da classe (incluindo os da classe base).



1.1 Comece por desenvolver e testar a classe base. Para tal, crie um projecto novo chamado *Aula3Ex1*, implemente a classe `CLibItem` e escreva uma função `main` adequada onde sejam criados vários objectos desta classe e invocados todos os seus métodos.

1.2 Seguidamente, implemente cada uma das classes derivadas, `CBook`, `CMagazine`, `CAudioRecord` e `CVideoRecord`, testando convenientemente cada uma delas antes de passar à classe seguinte. Os métodos das classes `CMagazine`, `CAudioRecord` e `CVideoRecord` devem permitir, à semelhança das classes `CLibItem` e `CBook`, alterar e aceder aos valores armazenados em todos os atributos da respectiva classe.

1.3 A classe `CLibItem` não representa nenhum objecto real, destinando-se apenas a servir de base às restantes classes, pelo que não deve ser possível criar objectos deste tipo. No entanto, esta classe tal como foi implementada também pode ser instanciada. Assim, implemente uma solução que impeça a sua instanciação.

Exercício 2

Modifique o exercício 2 da Aula 2 de forma a suportar os novos tipos de objectos implementados no exercício anterior. Para tal, crie um novo projecto chamado *Aula3Ex2*. A definição da classe que representa a biblioteca pode passar a ser a seguinte:

```
#define NUM_MAX_ITEMS     100

class CLibrary
{
public:
     CLibrary();
     ~CLibrary();

public:
     int CreateBook(const char* pTitle, unsigned int year,
                    const char* pAuthor, unsigned int numPages);
     int CreateMagazine(const char* pTitle, unsigned int year,
                        EPeriodicity periodicity, unsigned int number);
     int CreateAudioRecord(const char* pTitle, unsigned int year,
                           const char* pAuthor, unsigned int numTracks);
     int CreateVideoRecord(const char* pTitle, unsigned int year,
                           const char* pDirector, unsigned int duration);
     int Delete(int code);

     int Print(int code);
     void PrintAll();

private:
     unsigned int m_numItems;
     CLibItem* m_pItems[NUM_MAX_ITEMS];
};
```

Esta classe deve ser instanciada na função `main` do programa onde deverá ser criado um menu com as seguintes opções para aceder interactivamente à base de dados:

```
     B - Criar um livro
     M - Criar uma revista
     A - Criar um registo de audio
     V - Criar um registo de vídeo
     D - Apagar um item
     P - Visualizar um item
     L - Listar todos os items
     x - Sair do programa
```

Nota importante:
Enquanto no exercício anterior só existiam relações de herança entre as classes `CLibItem` e `CBook/Cmagazine/CAudioRecord/CVideoRecord`, neste exercício existe uma relação de inclusão entre a classe `CLibrary` e `CLibItem`, uma vez que nenhuma delas é uma especialização da outra, sendo `CLibrary` um contentor de objectos do tipo `CLibItem` ou de classes derivadas desta.

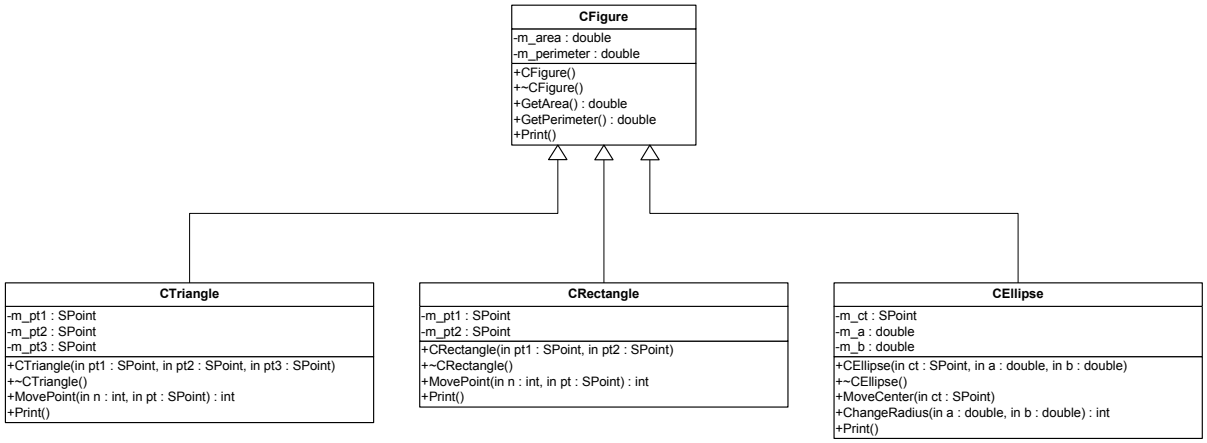
Exercício 3

Pretende-se criar um programa para manipular diferentes figuras geométricas. Nesta primeira versão pretende-se apenas a possibilidade do utilizador criar, apagar e modificar algumas figuras bem como permitir o cálculo do seu perímetro e área.

Como o cálculo da área e do perímetro pode ser complexo para algumas figuras, pretende-se que a sua determinação seja feita apenas quando o objecto for criado ou modificado. Uma vez determinados, devem ser armazenados em variáveis acessíveis através de métodos de leitura. Como todas as figuras possuem uma área e um perímetro, os atributos que armazenam esses valores e os respectivos métodos de acesso devem ser colocados numa classe base (`CFigure`), a partir da qual são derivadas outras classes que representam figuras específicas.

3.1 Comece por criar um novo projecto chamado *Aula3Ex3*. De seguida, implemente e teste as classes adequadas para as seguintes figuras: triângulo (`CTriangle`), rectângulo (`CRectangle`) e elipse (`CEllipse`).

A figura seguinte ilustra uma possível organização hierárquica das classes propostas.



Desenvolva e teste as classes separadamente, certificando-se que cada uma está correctamente implementada antes de passar à seguinte.

Sugestão:
Escreva para cada uma das classes `CTriangle`, `CRectangle` e `CEllipse` métodos `protected/private` para calcular as respectivas áreas e perímetros.

Informação:
A área e o perímetro de cada uma das figuras pedidas são calculados usando as seguintes fórmulas:

Figura	Área	Perímetro
Triângulo (a, b, c – comprimento dos lados)	$A = \sqrt{k \cdot (k-a) \cdot (k-b) \cdot (k-c)}$, $k = \frac{1}{2}(a+b+c)$	$P = a + b + c$
Rectângulo (a, b – comprimento dos lados)	$A = a \cdot b$	$P = 2 \cdot (a+b)$
Elipse (a, b – raios)	$A = \pi \cdot a \cdot b$	$P = 2\pi\sqrt{\frac{1}{2}(a^2+b^2)}$

3.2 Implemente a classe `CDrawing` para representar uma lista de figuras e que disponibilize métodos para criar, apagar e mostrar informação sobre as figuras (use a implementação interna da lista que lhe parecer mais prática).

3.3 Faça um programa que manipule uma lista de figuras, dando a possibilidade do utilizador acrescentar/remover figuras e escrever no ecrã as coordenadas dos pontos e/ou parâmetros que definem a figura bem como o seu perímetro e a sua área.
Sugestão:
Utilize um menu com as seguintes opções:

```
T - Criar um triângulo
R - Criar um rectângulo
E - Criar uma elipse
D - Apagar uma figura
P - Visualizar uma figura
L - Listar todas as figuras
x - Sair do programa
```

3.4 Acrescente ao programa anterior as figuras: quadrado e círculo.
Sugestões:

- Como o quadrado é um caso particular, ou uma especialização de um rectângulo em que os lados são todos iguais, a classe `CSquare` que representa um quadrado deverá ser derivada da classe `CRectangle`.
- Como a circunferência é um caso particular, ou uma especialização de uma elipse em que o raio maior é igual ao raio menor, a classe `CCircle` que representa uma circunferência deverá ser derivada da classe `CEllipse`.
- Os métodos das classes derivadas devem utilizar sempre que possível a funcionalidade da classe base.