作业7：OGR Geometry 类库简化版实现

【背景】

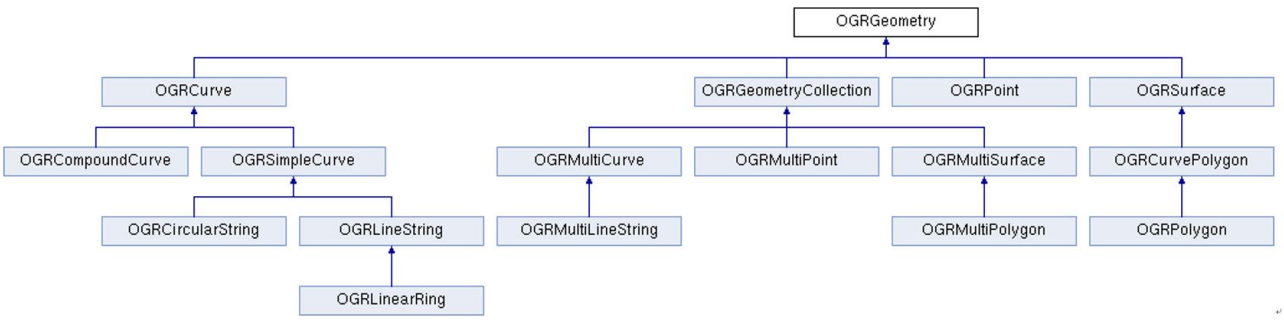
在GIS中，矢量数据的存储、显示、分析等均依赖于矢量几何要素。矢量几何要素是GIS中最重要的数据结构之一。常见的几何要素有简单点(Point)、简单线(Line)和简单面(Polygon)，以及它们的复杂形式如复杂点(MultiPoint)、折线(PolyLine)和复杂线(MultiPolyline)、复杂多边形(MultiPolygon)等。除此之外，还有曲线(Curve)、环(Ring)等几何要素形式。这些几何要素之间存在着复杂的关系，比如：有些复杂要素是简单的要素的集合，当然也可以认为简单要素是复杂要素最特殊最简单化的一种；有些要素则可视为另外一些要素的组合；有些要素则是其他要素的子类。

针对矢量几何要素以及它们之间的关系，业界存在着多种类（class）及体系关系描述形式，其中OGC（OpenGIS协会，Open Geospatial Consortium）给出的OpenGIS（Open Geodata Interoperation Specification, OGIS-开放的地理数据互操作规范）simple features data model得到了业界的广泛认可。

> OGC是一个非赢利性组织，目的是促进采用新的技术和商业方式来提高地理信息处理的互操作性(Interoperability)，它致力于消除地理信息应用（如地理信息系统，遥感，土地信息系统，自动制图/设施管理系统）之间以及地理应用与其它信息技术应用之间的藩篱，建立一个无"边界"的、分布的、基于构件的地理数据互操作环境。

下图即为GDAL(Geospatial Data Abstraction Library)中OGR Architecture对OpenGIS simple features data model中部分矢量要素（geometry）的实现架构。OGR库的应用十分广泛，包括ARCGIS 9.3、Google Earth和跨平台的GRASS GIS系统均采用了该模型。

> GDAL是一个在X/MIT许可协议下的开源空间数据转换库。它利用抽象数据模型来表达所支持的各种文件格式，拥有一系列命令行工具来进行数据转换和处理，其主要目标是实现数据共享和互操作。OGR是GDAL项目的一个分支，提供对矢量数据的支持。



【作业要求】

根据上图以及参考资料，使用C++语言对OGR Architecture进行如下实现。由于OGR Architecture十分复杂，本作业不要求对所有内容进行实现，请仔细阅读下面的具体说明：

1. 所有矢量几何要素均为**二维要素**，不考虑其他维度问题。
2. 参照图中的类结构，定义OGRLinearRing、OGRLineString、OGRMultiLineString、OGRMultiPoint、OGRMultiPolygon、OGRPolygon和OGRPoint共7个矢量几何要素类以及对应的所有基类，保证类之间关系正确，类名称需要和图中名称保持一致。

3. 针对所有要素类：
    - 所有要素均可获取其类型，返回值使用和其类名称一致的字符串，函数名称为 `char * GetGeometryType()` 或等价形式。
    - 所有要素均可获取其最小外包矩形，名称为 `GetMBR()` 。最小外包矩形由其左下角和右上角两个点确定，分别对应最小横纵坐标和最大横纵坐标。对于简单点，其左下角和其右上角坐标均设定为点要素自身的坐标。
    - 所有要素类均具有复制方法，函数为 `Copy(SampleClass & SampleObject)` 。
    - 所有要素均需要可判断是否与另一个要素几何相等(即坐标完全相同)，函数为 `Equals(SampleClass & SampleObject)` 。
    - 为方便检索查找，需要为每个要素分配一个唯一的全局ID，并作为类成员进行实现。需要为所有要素类添加一个 `GetID()` 方法，返回值为 `int` 。简单起见，不需要考虑要素数目过多导致整型变量溢出的问题。
4. 针对具有集合性质的 `OGRGeometryCollection` 各个子类，应包含其对应基本组成元素的添加（即添加到现有元素末尾）、根据索引删除、根据索引查询、根据指定索引更新以及获取元素数目操作，对应的函数名称分别为 `AddGeometry` 、 `RemoveGeometry` 、 `GetGeometry` 、 `UpdateGeometry` 、 `GetNumGeometries` 。 /*提示：可以使用标准模板库中的std::vector来存储元素，也可以基于数组或者链表自定义一个结构作为储存元素的集合。*/
5. `OGRLineString` 类型需要支持获取长度， `OGRPolygon` 类型需要支持获取面积、周长。 `OGRMultiLineString` 类型需要支持获取总长度， `OGRMultiPolygon` 类型需要支持获取总面积、总周长。获取长度、面积、周长的函数名称分别为 `GetLength` 、 `GetArea` 、 `GetPerimeter` 。 简单起见，作业中我们约定：所有多边形均为**凸多边形**；如果多边形存在“岛”或“洞”，那么集合类第一个元素（下标索引为0）为最外侧的“环”，如有后续元素，均为第一个元素内部的“环”（表示“岛”或“洞”），具体图形实例可参考附件PDF的27页、28页的图11和图12。
6. 实现Polygon和MultiPolygon类的包含方法，支持判断Point、LineString和Polygon是否位于Polygon和MultiPolygon类要素内部，函数名为 `Contains` 。
7. 实现Point、Polygon、MultiPoint和MultiPolygon这四个类的wkt的读写多态方法。在实现这几个方法过程中，如果涉及到其他类也需要实现wkt读写的多态方法，可以只给出声明以及简单实现使编译通过即可，不需要具体实现其他类的wkt读写方法。关于wkt的具体含义和形式，可以参见参考资料4的第7部分61页示例以及该部分其他内容。wkt读写方法名称分别为 `ImportFromWkt` 、 `ExportToWkt` ，参数均为文件路径，分别对应从文件导入和导出到文件。
8. 对各个类中有几何（即空间位置）性质的数字，精度规定为 `double` ；涉及到元素个数的表示，使用 `int` 即可；对于返回要素类型、最小外包矩形方法，可以定义枚举类型或者专门定义类来定义返回值类型。

注意：本作业是基于OGR Architecture的一个简化版练习。对于OGR Architecture中的其他方法，如变换、空间参照坐标系、wkb相关类型以及空间关系、空间分析、几何关系判断、不同类型要素间显式转换相关方法不作为要求。如果上述要求中命名和OGR Architecture存在不一致的地方，以本作业要求为准。

【提交要求】

本次作业以类库的形式提交，要求如下：

1. 只提交和上述各个类相关的所有头文件和源文件（即.h和.cpp文件）以及全局性参数文件（如果有的话，比如有同学也许将各个类的名称字串书写在一个全局性的文件中），不必提交其他文件（比如自行测试使用的界面程序等）。包含main函数的主程序界面cpp文件以及编程用测试数据请参见附件。
2. 为保证测试程序正常运行，每个要素类对应一个和它名称完全一致的.h文件和.cpp文件，也就是说，文件命名要和文件内对应的要素类**完全一致**，并且不包含其他要素类，在.h文件中给出声明，.cpp文件中给出实现。例如，OGRGeometry类对应于OGRGeometry.h和OGRGeometry.cpp文件。

【Bonus】

提供对上述类库进行应用的图形界面，实现部分GIS功能，比如交互输入、显示和输出线段或多边形（需**使用C++**实现，例如Qt、GTK、MFC等）

【参考资料和网址】

1. http://www.gdal.org/classOGRGeometry.html
2. http://www.opengeospatial.org/standards/sfa
3. http://www.gdal.org/ogr_arch.html
4. 附件Implementation Specification for Geographic Information Simple feature access Part 1 Common Architecture v1.2.1