郑州大学毕业设计（论文）

题 目 圆柱测量数据的图形化显示系统开发

院 系 机械工程

专 业 机械工程

年 级 2014级

学生姓名 时猛

指导教师 杨炯

2018年 05月 20日

**摘要**

本文对圆柱数据的测量的技术发展进行了研究和阐述，针对圆柱数据的测量系统，本文设计了一种高精度的测量数据处理方法，并开发了一种形象可视化的上位机软件。利用面向对象技术，多线程技术以及OpenGL图形开发技术等，这些关键技术为整个项目的软件架构和软件设计提供了很大的便利，提高了软件的开发效率以及软件的高效性。

测量数据的显示系统包括数据通信，数据拟合，图形展示等。数据通信是基于半双工通信与PLC连接，获取PLC的测量数据；数据拟合采用最小二乘法圆度拟合。图形展示是基于OpenGL将测量的数据形象的展示为图形，处理后的结果精度高，数据重复性好。上位机软件是基于C++编程语言开发而成的，集成了二维图形展示，状态栏，菜单栏，数据导出以及三维图形展示等功能，打造了一个人性化的软件界面。

关键词 圆度拟合， 图形化显示， 圆拟合（关键词3-5个）

Abstract

In this paper, the technical development of cylindrical data measurement is studied and elaborated. For the measurement system of cylindrical data, this paper designs a high-precision measurement data processing method, and develops a visualized upper computer software. Using object-oriented technology, multi-threading technology, and OpenGL graphics development technology, these key technologies provide great convenience for the software architecture and software design of the entire project, improving the efficiency of software development and the efficiency of software.

   The display system of measurement data includes data communication, data fitting, graphic display and so on. Data communication is based on half-duplex communication and PLC connection to obtain the measurement data of PLC; data fitting is based on least-squares roundness fitting. The graphic display is based on OpenGL's display of the measured data image as a graph, with high precision after processing and good data reproducibility. The host computer software is developed based on the C++ programming language. It integrates functions such as 2D graphics display, status bar, menu bar, data export, and 3D graphics display, creating a personal software interface.

key Words Roundness fitting，Graphical display，Circle fitting

目 录

毕业设计任务：……………………………………………………………..I

摘要…………………………………………………………………………..V

Abstrct………………………………………………………………………..V

1 绪论………………………………………………………………………1

1.1 选题的目的意义………………………………………………………1

1.2 国内外研究综述………………………………………………………1

1.3 本题的主要工作…………………………………………………………2

2 算法分析 ……………………………………………………….3

2.1 圆拟合算法分析………………………………………………………3

2.2 圆柱拟合算法分析…………………………………………………7

2.2.1 圆柱面拟合方法原理

2.3 本章小节…………………………………………………………………9

3 网络通信……………………………………………………………

4 系统架构……………………………………………………………

5 结论……………………………………………………………………….25

谢辞…………………………………………………………………………...26

参考文献……………………………………………………………………...26

1 绪论

1.1 选题的目的意义

在计算机视觉中 , 圆、点与直线一样被认为是最基本的图像特征。现实生活中大量物体为圆,圆形特征不但是自然界中的最基本元素, 而且是许多人造产品中普遍存在的形状, 圆的鲁棒精确提取在图像识别与计算机视觉中有着特别重要的意义。

1.2 国内外研究综述

在测绘工程中, 也经常会遇到空间圆形物体的测量问题, 如在一些建筑天顶 、大型仪器和地铁遂道的工程测量中 ,经常需要对空间圆形物体进行测量 ,得出其圆心及半径 , 确定其是否满足施工要求。各种零部件定位圆孔几何中心的定位精度对零部件的成功安装以及物体的整体定位, 有着重要的意义。进行圆中心定位的基本方法为:从图像边缘检测所得数据进行圆的拟合 ,或者将边缘数据转换为三维空间数据进行圆的拟合。因此,圆的拟合是圆中心定位的关键问题之一在机器视觉领域，圆是最为基本的图像特征，广泛应用于定位、测量、识别等方面，其拟合方法具有重要的研究意义。经常会遇到需要利用有限样本点进行圆拟合的问题，如在一些大型仪器和地铁隧道的工程测量中，经常需要对空间圆形物体进行测量，得出其圆心及半径，确定其是否满足施工要求。利用有限样本点进行圆拟合的情况，常用的方法有：

1. 平均值法

针对样本点均匀分布在圆上的情况，从理论上来说，其圆心坐标为各点坐标的平均值，圆心到任一点的距离即为圆的半径。但是在实际问题中，样本点不可能正好均匀的分布在圆上，因此误差较大。

1. 加权平均法

平均值法求得的圆心位置会偏离数据点密集的一边，求出的半径也会偏小。加权平均法是将样本点距离其相邻两点的弧长与圆的整个周长的2倍的比值作为权数。样本点距离其相邻两点的弧长可用相邻两点的距离替代。

（三）最小二乘法

通过最小化误差的平方和找到一组数据的最佳函数匹配。但是最小二乘法拟合的平方项对干扰数据非常敏感。

1. 霍夫变换法

将样本点为圆心，在参数平面画圆，许多点对应的圆会交于一点，该交点即可能为圆心。缺点计算量大，运算时占用内存大。

1. 基于共形代数的圆拟合方法

将欧式空间嵌入到共形空间，将圆表示为向量形式，共形空间中点与圆的内积表示点与圆的位置关系，对具有噪声的样本点进行拟合实验。

1. 基于聚类方法的圆拟合算法

该算法将边缘样本点映射到圆心附近，使用聚类算法进行分类滤噪处理并计算圆心，最后以此为基础计算半径。机器视觉进行圆拟合，首先采用工业相机对圆形特征进行拍照，图像在拍摄、传输和存储过程中都会不可避免地产生噪声，这将带来较大的拟合误差。因此去除圆上其他轮廓干扰或对其局部失真进行纠正是一个十分重要的步骤

1.3 本题的主要工作

基于该项目的目标是完成基于OPEN GL开发圆柱测量数据的显示功能模块；实现利用给定的截面计算圆柱度等数据功能。因此考虑时间因素和学习成本，确定选择利用最小二乘法拟合圆，经过多次迭代生成最终结果。通过C++编程，实现该项目。

2算法分析

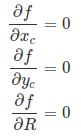
有一系列的数据点深度截图_选择区域_20180511155156，我们知道这些数据点近似的落在一个圆上，根据这些数据点计算出这个圆的参数这是一个非常重要的问题。圆拟合的方法有很多，本文选择的是基于最小二乘法。

2.1圆拟合算法分析

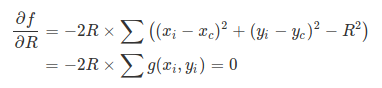
我们知道圆方程为深度截图_选择区域_20180511124008通常的最小二乘法拟合要求距离的平方和最小。也就是深度截图_选择区域_20180511124145最小，但是这个算起来非常的麻烦，也得不到解析解。因此我们退一步。深度截图_选择区域_20180511160736

上面这个式子简单的多。我们定义一个辅助函数：深度截图_选择区域_20180511160836

那么上述公式可以表示为： 深度截图_选择区域_20180511161222按照最小二乘法的通常的步骤，可得f取极值时对应下面的条件。



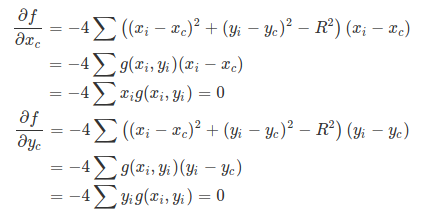
先来化简深度截图_选择区域_20180511161418



我们知道半径R是不可能为0的。所以必然有：

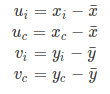
深度截图_选择区域_20180511162353

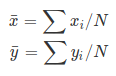
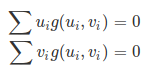
同样

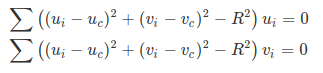


也就是下面两个等式： 深度截图_选择区域_20180511162510

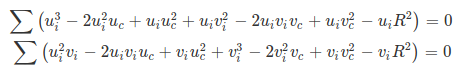
这里设：



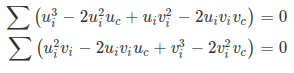
其中： 可以简单推导一下，就会发现：

这个变量替换修改了坐标远点的位置。而我们的等式与坐标原点的具体位置无关。所以必然成立。这两个式子展开可以写成这样：

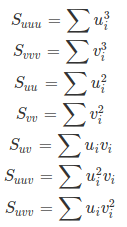
进一步展开：



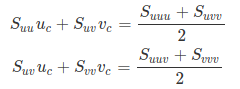
我们知道深度截图_选择区域_20180511163245所以上面两个式子是可以化简的。



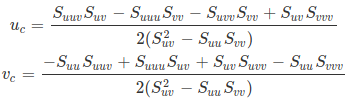
为了简化式子，我们定义几个参数：



那么上面的式子可以写为：

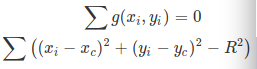


至此，可以解出Uc和Vc了。



那么：

深度截图_选择区域_20180511163627还剩下一个R没有求出来，也很简单。



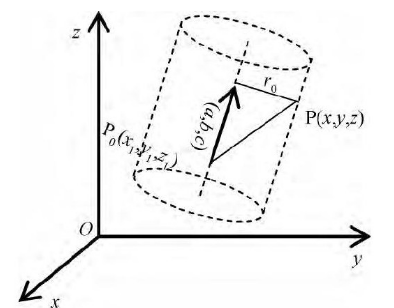
所以：深度截图_选择区域_20180511163750

以上就是最小二乘法的数学原理推导。然而这种方法对误差符合正态分布的数据点很有效。但是当数据中碰到一些干扰点。这些干扰点多数时候是偏向某一个方向的。因此，本文采用的是根据数据点到圆的距离绝对值的和来确定圆的参数，也就是下面的数学公式：

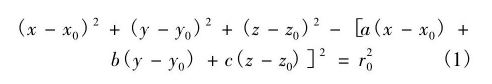
深度截图_选择区域_20180511164157 使得f取得最小值的Xc、Yc和R就是最佳拟合参数。

2.2 圆柱拟合算法分析

已知圆柱面上的点到其轴线上的距离恒等于半径r0，如图所示



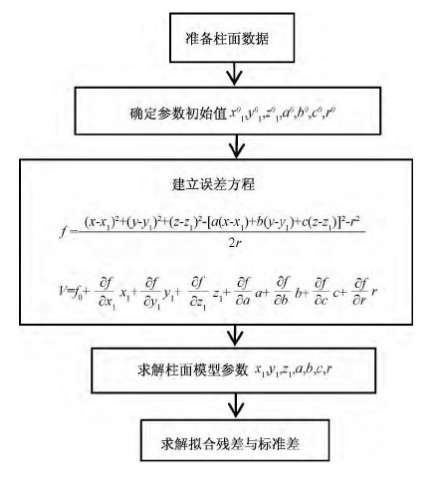
其中点P为圆柱上的任一点，P0为圆柱轴线上任一点，即：



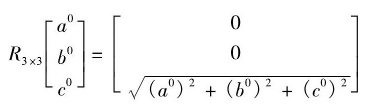
根据以上公式就可以唯一确认一个圆柱。

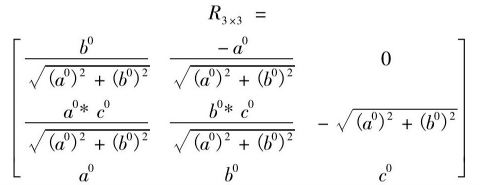
2.2.1 圆柱面拟合方法原理

圆柱面的拟合方法主要包括两个步骤：第一确定初始值；第二建立误差方程求解析值。本文的流程图为：



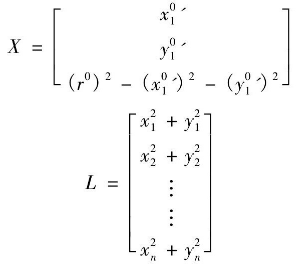
第一：将圆柱面上一点的相邻若干点拟合成一个平面，求得这个平面的法向量即为点的单位法向量。假设旋转矩阵为R3x3,要使：





写成矩阵形式为：

深度截图_选择区域_20180513150817

其中：

深度截图_选择区域_20180513150932

2.3 本章小结

本章主要说明该圆柱测量数据图形化显示系统中圆度拟合的算法分析，圆度拟合算法属于计算机图形学领域的算法分析。无论是在测量，还是在检验的过程中，我们都需要用到圆的拟合，圆拟合算法在工业领域应用的方方面面，我们往往是将一系列的点集生成一个一个的线段，之后依据线段之间的联系组合成为一个圆。针对误差过大的点，我们也应当可以清除数据，这些数据都属于冗余的，因此往往最后都需要做校验以及正确性检测。

3 网络通信

3.1 三菱PLC的分类及FX系列PLC的发展历史

三菱PLC包括一体式的和模块式的，有简单的控制系统，到高端的自动化解决平台。一体式的PLC数据集成式PLC, 包括CPU板、I/O板，显示面板、内存块、电源等，这些组件组合成一个整体。三菱FX系列的PLC属于一体式的，这种PLC多用于中小型系统。其主要特点就是可以实现10点到384点的控制规模，它体积比较小，并且容易安装，性价比高，成本低廉。同时可以实现多种网络、模拟量转换等多种特殊功能。而模块式的PLC包括CPU模块、I/O模块、内存、电源模块、以及各种功能的模块，这些模块可以按照一定规则组合配置。I/o点数可以达到4096点，最快的处理时间可以达到9ns,并且可以使用模拟量、网络等多种特殊功能模块。

70年代日本三菱公司开始研发PLC，之后第一台F系列投入实际应用，至今已经是高性能、高可靠型的最佳代表。FX系列包括FX1S、FX1N、FX2N、FX3U、FX3G等。其中FX1S基本单元有继电器或晶体管输出，不能链接扩展模块和特殊功能模块；FX1N可以连接FX2N系列的特殊功能模块，备有多种功能扩展板供使用；FX2N适用于从普通顺控开始的广泛用途。FX3U属于第三代微型可编程控制器，具有速度、容量、性能、功能的新机器、业内最高水平的高速处理，大大提升了内置功能；FX3G与FX1N系统兼容、高性能、高灵活性。提升用户友好。增强了程序安全性。

在Windows系统下使用的集成编程工具是GX Developer, 在设计、调试方面大大提高了编码的效率，允许以顺序功能图和结构化文本等语言进行编程。网络参数的设定和网络状态监视变得简单容易，并且同时可以在线监控程序、改变程序和数据以及强制I/O的ON/OFF等。可以实现离线编程、模拟测试和仿真等。在整个过程中，PLC通过ON/OFF位软元件或给字软元件赋值来实现真是动作情况。

3.2 PLC通信系统构成

FX系列PLC包括内置的CPU、存储器、输入输出、电源的产品。其中输入输出内置了电源回路和输入输出，用于扩展输入输出的产品，可以在连接其后面的设备供电，输入输出扩展模块内置了输入、输出，用于扩展输入输出。可以连接在基本单元上使用。特殊功能模块是一种可以是实现多种I/O以外其他功能的模块。包括模拟量控制、定位控制、网络系统、高速计数等功能。而像特殊适配器类似与特殊功能模块，也包含了模拟量和定位，通讯等多种功能。其他外部设备也是PLC系统不分割的一部分，它包括通过专用电缆，进行程序编制、对系统作一些设定、监控的手持编程器或编程软件、显示模块的监控设备和存储器以及用于安装特殊适配器用的连接器转换或实现通讯功能时使用。

整个系统构成包含三个方面，一是关于输入输出点数，二是关于特殊扩展设备，第三是关于消耗电流的计算，确认其是否可以扩展的输入输出点数。将剩余的电源容量作为外部负载的电源使用。剩余电源容量是否允许连接，特殊适配器和特殊功能单元。输入输出点数的合计就是计算基本单元、输入输出扩展单元/模块的输入输出点数，以及特殊功能单元/模块的输入输出占用点数，不同型号的输入输出占用点数，可以通过特殊功能单元/模块的台数 X 每台的点数 = 输入输出占用的点数。

如果CPU连接输入输出扩展单元，当上电时，会自动输入输出标号进行标号分配，因此，无需通过参数制定输入输出编号。输入输出编号以8进制进行分配，扩展的输入输出扩展单元/模块，接着前面的输入编号和输出编号，末尾从0开始分配。上电时，CPU会从离其最近的特殊功能模块开始，依次对特俗功能单元分配单元号，而在输入输出扩展单元/模块没有这个单元号。基本单元的输入为内部供电的漏型输入和源型输入的通用产品。通过选择可以将基本单元的所有输入设置为漏型输入，或者源型输入，但是不能混合使用，各基本单元和输入输出扩展单元，可以分别选择漏型输入、或是源型输入。

PLC执行程序是通过循环执行的顺序从而实现顺序控制。分为输入处理阶段，程序处理阶段、输出处理阶段。输入处理是在可编程控制器执行程序前，将可编程控制器中的所有状态读入。程序执行过程中即使输入发生了变化，输入映像区也不会发生变化，在执行下一个循环的输入处理时读取该变化。编程控制器根据内存中的内容，从输入的映像区和其他的映像区读出各软元件的状态，从最初依次开始运算，并将每次得出的结果写入到映像区。因此，各软元件的映像区随着程序的执行逐步改变其内容。所有指令执行结束后，输出映像区中的状态会传送至输出锁存内存。所谓的工程就是至程序、软元件注释、参数、软元件内存的一种集合体。

FX系列可编程控制器中，使用SFC图（顺序功能图)作为实现顺控，以一种便于理解的方式表现基于机械动作的各工序的作用和整个控制流程。在SFC程序中，状态S被作为一个控制工序，在其中对输入条件和输出控制的顺序进行编辑。工序推进后，前工序不再执行，因此可以通过各个工序的简单顺序来控制机械。状态S和驱动指令的动作，SFC程序中，用状态表示机械运行的各个工序。当状态为ON时，在此连接的梯形图动作，当状态为OFF时，与此连接的内部梯形图不发生动作。当各个状态设置的条件(转移条件)被满足时，下一个状态为ON，此前如果状态为ON则为OFF。在状态的转移过程中，仅仅在一个瞬间两个状态会同时ON。转移前的状态在转移后的下一个运算周期OFF。不能重复使用同一个状态编号。

特殊功能模块是为了实现特殊功能如AD、DA转换，输入输出定位等的模块，其带有自有的CPU和处理电路，和基本单元进行通信。PLC是依靠内部的位软元件和字软元件的使用来进行PLC的输出控制的。因此特殊功能模块是为了实现特殊功能如AD、DA转换，输入输出定位等的模块，其带有自有的CPU和处理电路，和基本单元进行通信。PLC是依靠内部的位软元件和字软元件的使用来进行PLC的输出控制的。因此，为了对外界的温度、电压、电流、流量等连续变化的的物理量进行控制，必须采用模拟量控制单元。特殊功能模块分为模拟量输入模块、模拟量输出模块、模拟量输入输出模块、温度传感器输入模块、温度调节模块等。特殊功能模块通过PLC的FROM/TO指令对特殊功能模块的缓冲存储器的数据进行信息交换。FROM指令用于把特殊功能模块的缓冲存储器的信息读书其当前值或者是状态信息。TO指令用于把数据及信息写入到特殊功能模块的缓冲存储器中。特殊功能模块内置了缓冲存储器，其作用就是用来与PLC基本单元进行交互。

，为了对外界的温度、电压、电流、流量等连续变化的的物理量进行控制，必须采用模拟量控制单元。特殊功能模块分为模拟量输入模块、模拟量输出模块、模拟量输入输出模块、温度传感器输入模块、温度调节模块等。特殊功能模块通过PLC的FROM/TO指令对特殊功能模块的缓冲存储器的数据进行信息交换。FROM指令用于把特殊功能模块的缓冲存储器的信息读书其当前值或者是状态信息。TO指令用于把数据及信息写入到特殊功能模块的缓冲存储器中。特殊功能模块内置了缓冲存储器，其作用就是用来与PLC基本单元进行交互。

4.系统架构

圆柱测量数据的图像化显示系统基于C++编程语言，在其上封装的框架Qt，利用面向对象编程在系统之上进行GUL编程。基于面向对象的编程思想，利用类的继承和多态。

4.1 C++简介

C++是一种中级语言， 由Bjarne Stroustrup于1979年在贝尔实验室设计开发的。C++进一步扩从了C语言，是一种面向对象的语言。C++可以在多个平台上使用。C++是一种静态类型的、编译式的、通用的，支持面向过程编程，面向对象编程以及泛型编程。它中和了高级语言和中级语言的特性。C++完全支持面向对象的程序设计，支持封装、抽象、继承和多态。C++包括核心语言，提供了所有的构件，包括变量，数据类型等等、C++标准库，提供了大量的函数，用于操作文件、字符串等等、标准模板库STL，提供了大量的方法，用于操作数据结构。

4.2 环境设置

本地环境设置，C++语言环境必须包括文本编辑器和C++编译器。文本编译器用于输入程序。文本编译器包括NotePad、EMACS和vim/vi等等，并且在不同的操作系统上，文本编译器的名称和版本都会有所不同。通过编辑器创建的 文件通常称为源文件，源文件包含程序源码。C++程序源代码的后缀为.cpp、.cc或者.cp。C++编译器用于将人类可读的源代码编译为机器可读的机器语言，这样CPU可以按给定的指令执行程序。C++编译器用于把源代码编译为最终的可执行程序。大多数的C++编译器对源文件的后缀名并不在意，一般来说大家都是默认为.cpp。最常用的免费可用的编译器是GNU的C/C++编译器, 非常适用C++语言的编译。在Linux或Unix系统上 使用 g++ -v命令来检查系统是否安装了GCC。

4.3 C++ 基础

C++程序可以定义为对象的集合。这些对象通过调用彼此的方法进行数据之间的交互。对象是具有状态和行为的一种数据结构，对象是一个类的实例。类是定义了对象行为和状态的模板。方法表示一种行为，一个类可以包括多种方法。C++语言定义了一些头文件，这些头文件包含了程序必须的信息。在C++中，分号是语句结束符。也就是说，每个语句必须以分号结束，块是一组使用大括号括起来的语句，C++不以行末作为结束符的标志，因此，一行可以使用多个语句。main 函数的返回值是返回给主调进程，使主调进程得知被调用程序的运行结果。标准规范中规定 main 函数的返回值为 int，一般约定返回 0 值时代表程序运行无错误，其它值均为错误号，但该约定并非强制。如果程序的运行结果不需要返回给主调进程，或程序开发人员确认该状态并不重要，比如所有出错信息均在程序中有明确提示的情况下，可以不写 main 函数的返回值。在一些检查不是很严格的编译器中，比如 VC, VS 等，void 类型的 main 是允许的。不过在一些检查严格的编译器下，比如 g++, 则要求 main 函数的返回值必须为 int 型。所以在编程时，区分程序运行结果并以 int 型返回，是一个良好的编程习惯。在 C++ 中 int main() 和 int main(void) 是等效的，但在 C 中让括号空着代表编译器对是否接受参数保持沉默。在 C 语言中 main() 省略返回类型也就相当说明返回类型为 int 型，不过这种用法在 C++ 中逐渐被淘汰。虽然 void main（）在很多系统都适用，但他毕竟不是标准的，所以应该避免这种用法, 应该使用这种 int main(void) 的写法比较妥当。是一种 整型 类型，里面保存的是一个整数，就像 int, long 那样。这种整数用来记录一个大小(size)。size\_t 的全称应该是 size type，就是说 一种用来记录大小的数据类型。通常我们用 sizeof(XXX) 操作，这个操作所得到的结果就是 size\_t 类型。

因为 size\_t 类型的数据其实是保存了一个整数，所以它也可以做加减乘除，也可以转化为 int 并赋值给 int 类型的变量。类似的还有 wchar\_t, ptrdiff\_t。

wchar\_t 就是 wide char type， 一种用来记录一个宽字符的数据, ptrdiff\_t就是 pointer difference type， 一种用来记录两个指针之间的距离的数据类型 。通常，size\_t 和 ptrdiff\_t 都是用 typedef 来实现的。你可能在某个头文件里面找到类似的语句：typedef unsigned int size\_t;而 wchar\_t 则稍有不同。在一些旧的编译器中，wchar\_t 也可能是用 typedef 来实现，但是新的标准中 wchar\_t 已经是 C/C++ 语言的关键字，wchar\_t 类型的地位已经和 char, int 的地位等同了。在标准 C/C++ 的语法中，只有 int float char bool 等基本的数据类型，至于 size\_t, 或 size\_type 都是以后的编程人员为了方便记忆所定义的一些便于理解的由基本数据类型的变体类型。

在程序中，局部变量和全局变量的名称可以相同，但是在函数内的局部变量和全局变量是两个独立的变量，互不影响。当变量间出现重名的情况下，作用域小的屏蔽作用域大的，但是由于在块中申请的变量作用域只限于当前块。C++提供了关键字explicit，可以阻止不应该允许的经过转换构造函数进行的隐式转换的发生。声明为explicit的构造函数不能在隐式转换中使用。C++中， 一个参数的构造函数(或者除了第一个参数外其余参数都有默认值的多参构造函数)， 承担了两个角色。 1 是个构造器 ，2 是个默认且隐含的类型转换操作符。所以， 有时候在我们写下如 AAA = XXX， 这样的代码， 且恰好XXX的类型正好是AAA单参数构造器的参数类型， 这时候编译器就自动调用这个构造器， 创建一个AAA的对象。这样看起来好象很酷， 很方便。 但在某些情况下（见下面权威的例子）， 却违背了我们（程序员）的本意。 这时候就要在这个构造器前面加上explicit修饰， 指定这个构造器只能被明确的调用/使用， 不能作为类型转换操作符被隐含的使用。std 是标准库函数使用的命名空间，是 standard（标准）的缩写。using namespace std ，它声明了命名空间 std，后续如果有未指定命名空间的符号，那么默认使用 std，这样就可以使用 cin、cout、vector 等。假设你不使用预处理 using namespace std;,就要加上 std::cin 或者 std::cout。cin 用于从控制台获取用户输入，cout 用于将数据输出到控制台。cin 是输入流对象，cout 是输出流对象，它们分别可以用 >> 和 <<，是因为分别在其类中对相应运算符进行了重载。

关于C++中this指针的理解：当你进入一个房子后，你可以看到房子内的桌子、椅子、地板等；但是你看不到房子的全貌；对于类来说，你可以看到成员函数、成员变量，但你看不到实例本身，但是应用this可以让我们看到这个实例本身。我的理解：class类就好比这座房子，this就好比一把钥匙，通过钥匙来打开了这座房子的门，那么里面的东西就随意你取用了。因为this作用域是在类的内部，自己声明一个类的时候，还不知道实例化对象的名字，所以用this来使用对象变量的自身。在非静态成员函数中，编译器在编译的时候加上this作为隐含形参，通过this来访问各个成员（即使你没有写上this指针）。C++ 中的 struct 对 C 中的 struct 进行了扩充，它已经不再只是一个包含不同数据类型的数据结构了，它已经获取了太多的功能。struct 能包含成员函数吗？ 能！struct 能继承吗？ 能！！struct 能实现多态吗？ 能！！！既然这些它都能实现，那它和 class 还能有什么区别？最本质的一个区别就是默认的访问控制，体现在两个方面：1）默认的继承访问权限。struct是public的，class是private的。2）struct 作为数据结构的实现体，它默认的数据访问控制是 public 的，而 class 作为对象的实现体，它默认的成员变量访问控制是 private 的。注意我上面的用词，我依旧强调 struct 是一种数据结构的实现体，虽然它是可以像 class 一样的用。我依旧将 struct 里的变量叫数据，class 内的变量叫成员，虽然它们并无区别。其实，到底是用 struct 还是 class，完全看个人的喜好，你可以将你程序里所有的 class 全部替换成 struct，它依旧可以很正常的运行。但我给出的最好建议，还是：当你觉得你要做的更像是一种数据结构的话，那么用 struct，如果你要做的更像是一种对象的话，那么用 class。

当然，我在这里还要强调一点的就是，对于访问控制，应该在程序里明确的指出，而不是依靠默认，这是一个良好的习惯，也让你的代码更具可读性。说到这里，很多了解的人或许都认为这个话题可以结束了，因为他们知道 struct 和 class 的“唯一”区别就是访问控制。很多文献上也确实只提到这一个区别。但我上面却没有用“唯一”，而是说的“最本质”，那是因为，它们确实还有另一个区别，虽然那个区别我们平时可能很少涉及。那就是：“class” 这个关键字还用于定义模板参数，就像 “typename”。但关键字 “struct” 不用于定义模板参数。这一点在 Stanley B.Lippman 写的 Inside the C++ Object Model 有过说明。问题讨论到这里，基本上应该可以结束了。但有人曾说过，他还发现过其他的“区别”，那么，让我们来看看，这到底是不是又一个区别。还是上面所说的，C++ 中的 struct 是对 C 中的 struct 的扩充，既然是扩充，那么它就要兼容过去 C 中 struct 应有的所有特性。也就是说 struct 可以在定义的时候用 {} 赋初值。那么问题来了，class 行不行呢？将上面的 struct 改成 class，试试看。报错！噢~于是那人跳出来说，他又找到了一个区别。我们仔细看看，这真的又是一个区别吗？你试着向上面的 struct 中加入一个构造函数（或虚函数），你会发现什么？对，struct 也不能用 {} 赋初值了。的确，以 {} 的方式来赋初值，只是用一个初始化列表来对数据进行按顺序的初始化，如上面如果写成 A a={'p',7}; 则 c1,n2 被初始化，而 db3 没有。这样简单的 copy 操作，只能发生在简单的数据结构上，而不应该放在对象上。加入一个构造函数或是一个虚函数会使 struct 更体现出一种对象的特性，而使此{}操作不再有效。

事实上，是因为加入这样的函数，使得类的内部结构发生了变化。而加入一个普通的成员函数呢？你会发现{}依旧可用。其实你可以将普通的函数理解成对数据结构的一种算法，这并不打破它数据结构的特性。那么，看到这里，我们发现即使是 struct 想用 {} 来赋初值，它也必须满足很多的约束条件，这些条件实际上就是让 struct 更体现出一种数据机构而不是类的特性。那为什么我们在上面仅仅将 struct 改成 class，{} 就不能用了呢？其实问题恰巧是我们之前所讲的——访问控制！你看看，我们忘记了什么？对，将 struct 改成 class 的时候，访问控制由 public 变为 private 了，那当然就不能用 {} 来赋初值了。加上一个 public，你会发现，class 也是能用 {} 的，和 struct 毫无区别！！！做个总结，从上面的区别，我们可以看出，struct 更适合看成是一个数据结构的实现体，class 更适合看成是一个对象的实现体。

C++多态意味着调用成员函数时，会根据调用函数的对象的类型来执行不同的函数；形成多态必须具备三个条件：1、必须存在继承关系；2、继承关系必须有同名虚函数（其中虚函数是在基类中使用关键字Virtual声明的函数，在派生类中重新定义基类中定义的虚函数时，会告诉编译器不要静态链接到该函数）；

3、存在基类类型的指针或者引用，通过该指针或引用调用虚函数；纯虚函数声明如下： virtual void funtion1()=0; 纯虚函数一定没有定义，纯虚函数用来规范派生类的行为，即接口。包含纯虚函数的类是抽象类，抽象类不能定义实例，但可以声明指向实现该抽象类的具体类的指针或引用。虚函数声明如下：virtual ReturnType FunctionName(Parameter) 虚函数必须实现，如果不实现，编译器将报错，对于虚函数来说，父类和子类都有各自的版本。由多态方式调用的时候动态绑定。实现了纯虚函数的子类，该纯虚函数在子类中就编程了虚函数，子类的子类即孙子类可以覆盖该虚函数，由多态方式调用的时候动态绑定。虚函数是C++中用于实现多态(polymorphism)的机制。核心理念就是通过基类访问派生类定义的函数。在有动态分配堆上内存的时候，析构函数必须是虚函数，但没有必要是纯虚的。友元不是成员函数，只有成员函数才可以是虚拟的，因此友元不能是虚拟函数。但可以通过让友元函数调用虚拟成员函数来解决友元的虚拟问题。析构函数应当是虚函数，将调用相应对象类型的析构函数，因此，如果指针指向的是子类对象，将调用子类的析构函数，然后自动调用基类的析构函数。

4.4 Qt框架

Qt是一个跨平台的C++图形用户界面库，是基于面向对象的C++语言，它提供了信号(signal)和槽(slot)的对象通信机制，具有可查询和可设计的属性，以及强大的事件和事件过滤器。Qt包括大量的联机参考文档，有超文本HTML方式，UNIX帮助页、man手册。其他的工具在应用时都存在一个普遍的问题，就是经常没有真正适合需求的控件，生成的自定义控件对用户来说，也是一个黑匣子。在Qt中，能够创建控件，具有绝对的优越性，生成自定义控件非常简单，并且容易修改控件。由于Qt是一种跨平台的GUI工具包，所以，它对编程者隐藏了在处理不同窗口系统时的潜在问题。为了将基于Qt程序更加方便，Qt包含了一系列的类，该类能够使程序员避免了在文件处理、时间处理等方面存在依赖操作系统方面的问题。Qt也为本地化应用提供了完全的支持，所有用户界面的文本或字符串都可以利用翻译工具将其翻译为各国语言。为了适合用户的需求，Qt提供了丰富的API函数，用于专门的GUI。Qt还提供了基于模板的初始化、文件和通用的I/O设备、目录管理、日期/时间类、常用表达式解析等。目的是利用这些类，建立或生成不同的功能，用它们来实现Qt的通用化。除此之外，也可以利用STL标准模板库。Qt编程的基本模块为控件，一个控件是一个用户界面的组成部分，Qt包含用于创建专业外观的用户界面所需的所有控件。Qt能执行一些基本的任务，比如图形的润色，比一般的基于平台的代码要快。大多数的GUI工具包是基于分层的方法。比如，工具包为本地窗口系统组件提供了很多C++类，这些结构使组件的继承性和通用性变得很差。在层次化的工具包中，GUI功能常成为所有使用的窗口系统所必须的最普遍的基础。Qt仿效本地串口系统的组件，这是一种非常复杂的技术。Qt还提供一些更有用的函数，使用于多平台。

4.4.1 Qt环境配置

Qt Creator作为Qt官方的开发环境，对Qt各方面支持都非常完善，并且使用体验非常好。Qt Creator主界面分为6个模式：欢迎模式、编辑模式、设计模式、Debug模式、项目模式和帮助模式，分别由左边的六个图标进行模式的切换。刚进去显示的是欢迎界面，有一些入门教程、开发的项目列表、Qt提供的示例程序，单击一个示例程序可以直接打开至这个项目。

4.4.2 Qt面向对象

Qt具有模块设计和控件或元素的可重用性的特点。一个控件不需要知道它的内容和用途，通过signal和slot与外界通信、交流。并且所有的Qt都可通过继承。Qt包括一些抽象的类，对象模型是框架的根本。C++标准中虽然有很多的对实时的对象模型的支持，但其静态的特性，导致其仍然缺乏灵活性，Qt提供了自己的对象模型。主要包括对象树，对象属性，以及元对象系统等。1）对象树（Object tree ），由于GUI的设计层次结构比较强，并需要兼顾效率，Qt设计了对象树，并支持了动态类类型转换，其中父对象与子对象相互指向，整体的结构关系相当于一个森林， 父对象与子对象为一对多的关系，并有多个平行的父对象。其中有两点需要注意，首先当父对象析构的时候将析构所有的子对象，如果子对象在栈中或者为全局变量，有可能导致重复析构，程序崩溃（ C++标准中，局部变量析构的顺序是构造过程的逆向 ）；其次如果事件在发送中将QObject( QObject作为其最终的基类，源自官方Qt文档中~QObject() ）析构，也可能导致多线程调用过程中的崩溃。2）对象属性（Object properties ），对象属性是基于元对象系统（ Meta-Class system ，信号槽机制 signals and slots 也基于该模块, 在后边的章节将会具体谈），Qt中的Q\_PROPORTY(...)宏标记了相关属性信息，将其注册到QMetaObject中，QMetaObject记录了所有注册过的属性信息，允许程序在编译时不知情的一些信息，在实时运行阶段能够动态添加进来。相当于一个盒子，假设我是盒子的设计者，我并不知道将来盒子里要放什么东西，设计的要求是，我可以放新的东西进来也可以添加已有的东西，也可以将盒子更改我自定义的样式。3）元对象系统（ meta-class system ），元对象系统负责信号槽机制，实时类型判断，以及对象属性。这个系统依赖于MOC（meta object complier），MOC“阅读”代码，将所有标记信息读入，整理，来满足上面所述3个方面的应用。在QObject所属子类中，所有拥有QMetaObject的类都支持反射模式（并不被C++标准支持），但Qt通过MOC间接支持了该设计模式，QMetaObject提供了类的属性以及方法描述信息。

4.4.3 Qt组件间的通信

Qt提供signal和slot概念，这是一种安全可靠的方法，它允许回调，并支持对象之间在彼此不知道对方信息的情况下，进行合作，这使Qt非常适合于控件编程。信号与槽是多对多的关系，通过connect将信号与槽链接在一起，这样的对应关系维护了指定的函数参数，信号emit后就不再关心是谁来处理，这个是一个单向的过程，不过槽函数可以继续emit信号，从而形成一个环，达到函数级通信的目的。因为信号机制无法返回结果，所以槽函数必须是统一的无返回类型。meta-object system维护了一个信号以及对应函数的关系列表，取代了回调函数，当发射信号signal1后，程序通过元对象系统定位到对应的函数，好比如说我有一个储物柜，分为很多的小格子，每个格子都有自己的编号，我现在将记录有格子编号的物品交给管理员，管理员自然就知道将物品放到哪个格子中了。在底层实现方面，meta-object complie( moc )编译期间，生成了专门的CPP文件，扩展meta class，记录函数所在偏移，并记录对应关系表。 整个信号槽机制依赖于MOC，并没有使用C++的模板的泛型编程，主要原因虽然效率要比模板稍差，但稳定性，扩展性都要比模板好。

4.3.4 Qt的组成部分

QtCore不但包含QString、QList和QFile