魅力·实践·发现

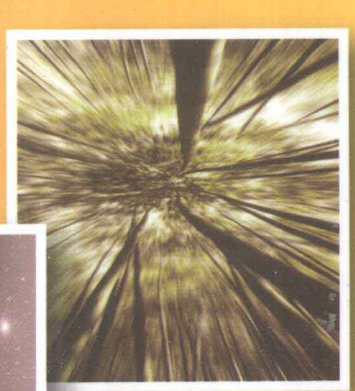
Qt 4 开发实践

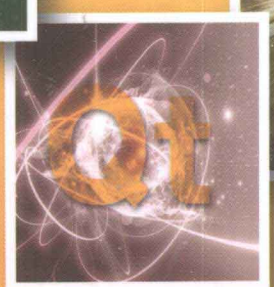
郑阿奇 主编
陈 超 编

展现精英高手发现之旅

站在流行平台开发实践

介绍流行软件神奇魅力





電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

魅力·实践·发现

Qt 4 开发实践

郑阿奇　主　编
陈　超　　编

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

Qt 是诺基亚公司 C++可视化开发平台，目前最新版本为 2010 年发布的 Qt 4.6。本书以 Qt 4.6 作为平台，先介绍 Qt 4.6 平台开发中需要着重注意的 C++主要知识点，在简单介绍开发环境的基础上，用一个小实例，介绍 Qt 4.6 开发应用程序的过程，然后系统介绍 Qt 4.6 应用程序的开发和编程技术，一般均通过实例介绍和讲解内容。本书经过非作者人员审读试做。实例代码可在出版社网站上免费下载。

通过本书学习，结合实例上机练习，一般能够在比较短的时间内掌握 Qt 4 应用技术。

本书可作为 Qt 4 学习和开发人员参考使用，也可作为大学本科、高职高专教材或 Qt 4 培训用书。

图书在版编目（CIP）数据

Qt 4 开发实践 / 郑阿奇主编. —北京：电子工业出版社，2011.1
（魅力 • 实践 • 发现）
ISBN 978-7-121-12669-7

Ⅰ. ①Q… Ⅱ. ①郑… Ⅲ. ①软件工具—程序设计 ②C 语言—程序设计 Ⅳ. ①TP311.56 ②TP312

中国版本图书馆 CIP 数据核字（2010）第 255070 号

策划编辑：郝黎明
责任编辑：张云怡
印　　刷：北京丰源印刷厂
装　　订：三河市鹏成印业有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：31　字数：790 千字
印　　次：2011 年 1 月第 1 次印刷
印　　数：4 000 册　定价：53.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

Qt是诺基亚公司C++可视化开发平台，目前最新版本为2010年发布的Qt 4.6。与Visual C++相比，Qt简单方便、功能完善、跨平台、扩展能力强，可以进行嵌入式开发。

1996年，Qt进入商业领域，已成为全世界范围内数千种成功的应用程序的基础。它也是流行的Linux桌面环境KDE的基础。Qt是一个跨平台的C++图形用户界面应用程序框架(C++ GUI)，能提供给应用程序开发者建立艺术级图形用户界面所需的所用功能。它是完全面向对象的，很容易扩展，并且可进行组件编程。

Qt商业版提供给商业软件开发。它们提供传统商业软件发行版以及在协议有效期内的免费升级和技术支持服务。而Qt开源版提供了和商业版本同样的功能，它是免费的。从Qt 4.5起，诺基亚将为Qt增添新的开源LGPL授权选择，并且Qt源代码库面向公众开放，Qt开发人员可通过为Qt以及与Qt相关的项目贡献代码、翻译、示例以及其他内容，协助引导和塑造Qt未来的发展。

如果你需要可视化学习C++，如果你需要用C++开发应用系统，选择Qt是最佳的。

本书以Qt 4.6作为平台，循序渐进，通过不同实例对内容加以说明，知识和能力融为一体。

本书首先复习Qt开发中需要着重注意的C++主要知识点，使读者此后学习Qt更为轻松。然后在简单介绍开发环境的基础上，用一个小实例，介绍Qt 4.6开发应用程序的过程，从而总体上有所了解。其后，系统介绍Qt 4.6开发应用程序的所有内容和技术，循序渐进，处处通过实例介绍来理解内容。因为本书经过审读和试做，所以从前面读下来，一般不会有什么问题。

为了处理好内容介绍的连续性和内容中出现新的知识详细说明的矛盾，采用由我们首先提出的在纸介质实现超链接的方法。例如，第2章的部分目录如下：

2.2　Qt的安装……31
2.2.1　下载Qt Creator……31
2.2.2　运行Qt Creator……31
2.2.3　Qt的开发环境……32
2.3　Qt的开发步骤及实例……34
2.3.1　设计器Qt Designer实现……34
2.3.2　编写代码实现……42
L2.2　Qt的安装……45
L1　伙伴编辑模式（Edit Buddies）……45
L2.3　Qt的开发步骤及实例……45
L1　信号和槽机制（Signal&Slot）……45
L2　功能模块……47
L3　Qt元对象系统……48
L4　布局管理器……48

表示2.2节中首次出现“伙伴编辑模式（Edit Buddies）”读者可能不理解，但又不便在当

时详细解析，在对这个名词加黑的同时有下划线标识，表示在 L2.2 节中有专门介绍。同样，2.3 节中有 4 个名词在 L2.3 节中详细解析。

本书介绍内容时使用了很多实例，书中提供的开发过程和代码本身就是系统和完备的。但为了方便读者上机练习，在书中实例同时提供源代码编号 CHXx（例如 CH201 是第 2 章的 01 例），这些代码可在出版社网站上免费下载。

通过本书学习，结合实例上机练习，一般能够在比较短的时间内掌握 Qt 应用技术。

本书由南京师范大学郑阿奇主编，东南大学陈超编写。参加本书编写的还有郑进、陶卫冬、邓拼搏、严大牛、韩翠青、王海娇、刘博宇、陈瀚、孙德荣、吴明祥、周何骏、徐斌、孙承龙、袁永福等。

本书配有教学课件和书中所有实例源代码及其相关文件，需要者可从出版社网站上免费下载。网站地址为：http:// www.hxedu.com.cn。

由于编者水平有限，错误之处在所难免，敬请广大读者、师生批评指正。

意见、建议邮箱：easybooks@163.com

编　者

2010.10

目　录

第 1 章　C++相关知识 …… 1
1.1　C++程序结构 …… 1
1.2　C++预处理 …… 2
1.2.1　宏定义命令 …… 2
1.2.2　文件包含命令 …… 3
1.2.3　条件编译命令 …… 3
1.3　C++异常处理 …… 4
1.3.1　使用 C++异常处理 …… 4
1.3.2　嵌套异常和栈展开 …… 7
1.4　C++面向对象程序设计 …… 7
1.4.1　基本概念 …… 8
1.4.2　类的拷贝构造函数和赋值函数 …… 13
1.4.3　模板类 …… 17
1.4.4　继承与接口 …… 22
1.4.5　多重继承及虚继承 …… 25
1.4.6　多态 …… 26
1.4.7　友元 …… 27
第 2 章　Qt 概述 …… 30
2.1　什么是 Qt …… 30
2.2　Qt 的安装 …… 31
2.2.1　下载 Qt Creator …… 31
2.2.2　运行 Qt Creator …… 31
2.2.3　Qt 的开发环境 …… 32
2.3　Qt 的开发步骤及实例 …… 34
2.3.1　设计器 Qt Designer 实现 …… 34
2.3.2　编写代码实现 …… 42
L2.2　Qt 的安装 …… 45
L1　伙伴编辑模式（Edit Buddies） …… 45
L2.3　Qt 的开发步骤及实例 …… 45
L1　信号和槽机制（Signal&Slot） …… 45
L2　功能模块 …… 47
L3　Qt 元对象系统 …… 48

L4 布局管理器 …… 48
第 3 章 Qt 的模板库、工具类及控件 …… 49
3.1 字符串类 QString …… 49
3.1.1 操作字符串 …… 49
3.1.2 查询字符串数据 …… 51
3.1.3 字符串的转换 …… 51
3.2 Qt 容器类 …… 53
3.2.1 QList 类、QLinkedList 类和 QVector 类 …… 53
3.2.2 QMap 类和 QHash 类 …… 59
3.3 QVariant 类 …… 62
3.4 Qt 的算法及正则表达式 …… 64
3.4.1 Qt 常用算法 …… 64
3.4.2 基本的正则表达式 …… 65
3.5 控件 …… 66
3.5.1 按钮组（Buttons） …… 66
3.5.2 输入部件组（Input Widgets） …… 68
3.5.3 显示控件组（Display Widgets） …… 69
3.5.4 空间间隔组（Spacers） …… 71
3.5.5 布局管理组（Layout） …… 71
3.5.6 容器组（Containers） …… 71
3.5.7 项目视图组（Item Views） …… 74
3.5.8 项目控件组（Item Widgets） …… 76
3.5.9 小综合例子 …… 81
L3.1 字符串类 QString …… 84
L1 隐式共享 …… 84
L2 内存分配策略 …… 85
L3.5 控件 …… 86
L1 Qt::WindowFlags 枚举类型 …… 86
第 4 章 布局管理 …… 88
4.1 分割窗口 QSplitter 类 …… 88
4.2 停靠窗口 QDockWidget 类 …… 90
4.3 堆栈窗体 QStackedWidget 类 …… 93
4.4 基本布局（QLayout） …… 96
4.5 综合例子：修改用户资料 …… 101
第 5 章 基本对话框 …… 113
5.1 标准文件对话框：QFileDialog 类 …… 117

5.1.1 函数说明 ……117
5.1.2 创建步骤 ……118
5.2 标准颜色对话框：QColorDialog 类 …… 119
5.2.1 函数说明 ……119
5.2.2 创建步骤 ……119
5.3 标准字体对话框：QFontDialog 类 …… 120
5.3.1 函数说明 ……120
5.3.2 创建步骤 ……120
5.4 标准输入对话框：QInputDialog 类 …… 121
5.4.1 标准字符串输入对话框 ……125
5.4.2 标准条目选择对话框 ……125
5.4.3 标准 int 类型输入对话框……126
5.4.4 标准 double 类型输入对话框 ……127
5.5 消息对话框：QMessageBox 类…… 128
5.5.1 Question 消息框 ……131
5.5.2 Information 消息框 ……132
5.5.3 Warning 消息框 ……132
5.5.4 Critical 消息框 ……133
5.5.5 About 消息框 ……133
5.5.6 About Qt 消息框 ……134
5.6 自定义（Custom）消息框…… 134
5.7 工具盒 QToolBox 类 …… 136
5.8 进度条（QProgressBar/QProgressDialog） …… 141
5.9 QPalette 类与移动图片综合实例 …… 146
5.9.1 QPalette 类 ……149
5.9.2 QTime 类 ……156
5.10 可扩展对话框的基本实现方法 …… 156
5.11 不规则窗体 …… 160
5.12 程序启动画面（QSplashScreen） …… 163
5.13 多文档的创建 …… 164
5.14 使用 Qt Designer 设计对话框 …… 168
第 6 章 QmainWindow 主窗体 …… 172
6.1 创建菜单的菜单、工具栏以及动作 …… 176
6.1.1 动作（Action）的实现 ……177
6.1.2 菜单（Menus）的实现 ……179
6.1.3 工具栏（ToolBars）的实现 ……180

6.2 新建文件 …… 182
6.3 标准文件对话框（QFileDialog） …… 183
6.4 标准打印对话框（QPrintDialog） …… 185
6.4.1 文本打印 …… 185
6.4.2 图像打印 …… 186
6.5 QMatrix 实现图像坐标变换 …… 188
6.5.1 缩放功能 …… 188
6.5.2 旋转功能 …… 189
6.5.3 镜像功能 …… 190
6.6 文本编辑 …… 191
6.6.1 设置字体功能 …… 194
6.6.2 设置字号功能 …… 195
6.6.3 设置文字加粗功能 …… 195
6.6.4 设置文字斜体功能 …… 195
6.6.5 设置文字加下画线功能 …… 196
6.6.6 设置文字颜色功能及标准颜色对话框（QColorDialog） …… 196
6.6.7 设置字符格式功能 …… 197
6.7 实现段落对齐及文本排序功能 …… 197
6.7.1 实现段落对齐功能 …… 198
6.7.2 实现文本排序功能 …… 199
第 7 章 图形与图画 …… 203
7.1 各类位置相关函数的区别 …… 203
7.2 各种与位置相关函数的使用场合 …… 204
7.3 各种基础图形的绘制 …… 208
7.3.1 绘图区的实现 …… 209
7.3.2 主窗口的实现 …… 213
7.4 双缓冲机制 …… 226
7.4.1 绘图区的实现 …… 227
7.4.2 主窗口的实现 …… 231
7.5 SVG 格式图片的显示 …… 234
L7.5 SVG 格式图片的显示 …… 240
L1 XML …… 240
第 8 章 图形视图（GraphicsView）框架 …… 247
8.1 GraphicsView 体系结构 …… 247
8.1.1 GraphicsView 框架结构的特点 …… 247
8.1.2 GraphicsView 框架结构的内容 …… 247

8.2 GraphicsView 坐标系统 …… 249
8.3 GraphicsView 综合例子 …… 251
8.3.1 飞舞的蝴蝶例子 …… 251
8.3.2 地图浏览器例子 …… 254
8.3.3 各种 GraphicsItem 的创建实例 …… 260
8.3.4 GraphicsItem 的旋转、缩放、切变和位移实例 …… 270
第 9 章 模式/视图（Model/View）结构 …… 278
9.1 概念 …… 278
9.1.1 模型 …… 279
9.1.2 视图 …… 279
9.1.3 代理 …… 279
9.1.4 使用已有的模型视图类例子 …… 279
9.2 模型（Models） …… 281
9.3 视图（Views） …… 285
9.4 代理（Delegates） …… 298
第 10 章 文件及磁盘处理 …… 307
10.1 读写文本文件 …… 307
10.2 读写二进制文件 …… 310
10.3 处理目录 …… 312
10.4 获取文件信息 …… 319
10.5 监视文件和目录变化 …… 324
第 11 章 网络与通信 …… 326
11.1 获取本机网络信息 …… 326
11.2 基于 UDP 的网络广播程序 …… 330
11.2.1 UDP 协议工作原理 …… 330
11.2.2 UDP C/S 编程模型 …… 331
11.2.3 UDP 服务器端 …… 331
11.2.4 UDP 客户端 …… 333
11.3 基于 TCP 的网络聊天室程序 …… 337
11.3.1 TCP 协议工作原理 …… 337
11.3.2 TCP C/S 编程模型 …… 338
11.3.3 TCP 服务器端 …… 338
11.3.4 TCP 客户端 …… 340
11.4 实现 HTTP 文件下载 …… 350
11.5 实现 FTP 上传和下载 …… 356

第 12 章 事件处理 …… 363
12.1 鼠标事件 …… 363
12.2 键盘事件 …… 366
12.3 事件过滤 …… 371
第 13 章 多线程 …… 377
13.1 多线程的简单实现 …… 377
13.2 多线程的控制 …… 382
13.2.1 使用 QMutex 类、QMutexLocker 类 …… 383
13.2.2 使用 QSemaphore 类 …… 384
13.2.3 使用 QWaitCondition 类 …… 387
13.3 多线程例子 …… 391
13.3.1 服务器端 …… 391
13.3.2 客户端 …… 396
第 14 章 数据库 …… 401
14.1 数据库基本概念 …… 401
14.2 常用 SQL 命令 …… 404
14.2.1 SELECT 查询 …… 404
14.2.2 数据操作 …… 408
14.3 Qt 操作数据库 …… 409
14.3.1 Qt 操作 SQLite 数据库 …… 409
14.3.2 Qt 综合操作数据库和 XML …… 416
第 15 章 Qt 多国语言国际化 …… 444
15.1 概念 …… 444
15.1.1 QString、QTranslator 等类和 tr()函数的作用 …… 444
15.1.2 *.qm 文件的生成 …… 445
15.2 实例 …… 446
15.2.1 实例 1 …… 446
15.2.2 实例 2 …… 448
第 16 章 Qt 单元测试框架 …… 453
16.1 QTestLib 框架 …… 453
16.2 简单的 Qt 单元测试 …… 454
16.3 数据驱动测试 …… 456
16.4 GUI 测试 …… 459
16.4.1 仿真 GUI 事件 …… 460
16.4.2 重放 GUI 事件 …… 461
16.5 简单性能测试 …… 462

第 17 章　Linux 下 Qt 的安装和使用 …… 464
17.1　安装 Qt 步骤 …… 464
17.2　Hello World 例子 …… 466
17.2.1　通过编写代码实现 …… 467
17.2.2　通过 Qt Designer 实现 …… 470
附录 A　部分名字 …… 475
附录 B　Qt 调试 …… 479

第 1 章

C++相关知识

本章首先复习 C++程序结构、三种 C++预处理命令、异常处理的一种结构化形式的描述机制和有关面向对象程序设计中的一些基本概念。为学习 Qt 进行必要的准备。

1.1 C++程序结构

一个程序是由若干个程序源文件组成的。为了与其他语言相区别，每一个 C++程序源文件通常以.cpp 为扩展名，由编译预处理指令、数据或数据结构定义以及若干个函数组成。代码中，main()表示主函数。无论该函数在整个程序中的哪个位置，每一个程序执行时都必须从 main()函数开始，因此，每一个 C++程序或者由多个源文件组成的 C++项目都必须包含一个且只有一个 main()函数。

下面举一个简单的 C++程序例子 Ex_Simple 来说明一下：

```
1// [例Ex_Simple] 一个简单的C++程序
2/* 第一个简单的C++程序 */
3#include <iostream.h>
4int  main()
5{
6    double  r,  area;                          // 定义变量
7    cout<<"输入圆的半径：";                    // 显示提示信息
8    cin>>r;                                    // 从键盘上输入变量r的值
9    area = 3.14159 * r * r;                    // 计算面积
10   cout<<"圆的面积为："<<area<<"\n";          // 输出面积
11   return 0;                                  // 指定返回值
12}
```

其中：

- 行号为 3 的代码是 C++文件包含#include 的编译指令，称为预处理指令。

#include 后面的 iostream.h 是 C++编译器自带的文件，称为 C++库文件，它定义了标准输入/输出流的相关数据及其操作。由于该程序中用到了输入/输出流对象 cin 和 cout，因而需要用#include 将其合并到该程序中，又由于它们总是被放置在源程序文件的起始处，

所以这些文件被称为头文件（Header File）。C++编译器自带了许多这样的头文件，每个头文件都支持一组特定的“工具”，用于实现基本输入输出、数值计算、字符串处理等方面的操作。

由于 iostream.h 是 C++的头文件，因此这些文件以“.h”为扩展名，以便与其他文件类型相区别，但这是 C 语言的头文件格式。尽管 ANSI/ISO C++仍支持这种头文件格式，但已不建议再采用，即包含头文件中不应再有.h 这个扩展名，而应使用 C++的 iostream。例如：

```
#include <iostream>
```

但为了使 iostream 中的定义对程序有效，还需使用下面名称空间编译指令来指定：

```
using namespace std;                    // 注意不要漏掉后面的分号
```

using 是一个在代码编译之前处理的指令。namespace 称为名称空间，它是 ANSI/ISO C++一个新的特性，用于解决在程序中同名标识存在的潜在危机。

● 上述程序 Ex_Simple 中的“/*...*/”之间的内容或“//”开始一直到行尾的内容是用来注释的，其目的只是为了提高程序的可读性，对编译和运行并不起作用。正是因为这一点，所注释的内容既可以用汉字来表示，也可以用英文来说明，只要便于理解就行。

需要说明的是，C++中的“/*...*/”用于实现多行的注释，它将由“/*”开头到“*/”结尾之间所有内容均视为注释，称为块注释。块注释（“/*...*/”）的注解方式可以出现在程序中的任何位置，包括在语句或表达式之间。而“//”只能实现单行的注释，它是将“//”开始一直到行尾的内容作为注释，称为行注释。

1.2 C++预处理

和 C 语言一样，C++预处理命令也有三种：宏定义命令、文件包含命令、条件编译命令。这些命令在程序中都是以“#”来引导的，每一条预处理命令必须单独占用一行，但在行尾不能有分号“;”。

1.2.1 宏定义命令

用#define 可以定义一个符号常量，如：

```
#define     PI  3.141593
```

这里的#define 就是宏定义命令，它的作用是将 3.141593 用 PI 代替，PI 称为宏名。需要注意的是：

(1) #define、PI 和 3.141593 之间一定要有空格，且一般将宏名定义成大写，以便与普通标识符相区别。

(2) 宏被定义后，一般不能再重新定义，而只有当使用如下命令时才可以重新定义：

```
#undef  宏名
```

(3) 一个定义过的宏名可以用来定义其他新的宏。

(4) 宏还可以带参数，例如：

```
#define MAX(a,b)  ((a)>(b)?(a):(b))
```

其中（a,b）是宏 MAX 的参数表，如果在程序中出现下列语句：

```
x = MAX(3, 9);
```

则预处理后变成：

```
x = (3>9?3:9);                                    // 结果为 9
```

很显然，带参数的宏相当于一个函数的功能，但却比函数简捷。

1.2.2　文件包含命令

所谓“文件包含”是指将另一个源文件的内容合并到源程序中。C/C++语言提供了#include 命令用来实现文件包含的操作，它有如下两种格式：

```
#include  <文件名>
#include  “文件名”
```

文件名一般以.h 为扩展名，因而称它为“头文件”，如前面的程序例子中 iostream.h 是头文件的文件名。“文件包含”的两种格式中，第一种格式是将文件名用尖括号“<>”括起来的，用来包含那些由系统提供的并放在指定子目录中的头文件。第二种格式是将文件名用双引号括起来，用来包含那些由用户定义的放在当前目录或其他目录下的头文件或其他源文件。

1.2.3　条件编译命令

一般情况下，源程序中所有的语句都参加编译。但有时也希望根据一定的条件去编译源文件的不同部分，这就是“条件编译”。条件编译使得同一源程序在不同的编译条件下得到不同的目标代码。C/C++提供的条件编译命令有下列几种常用的形式：

格式 1：

```
#ifdef <标识符>
    <程序段 1>
[
#else
    <程序段 2>
]
#endif
```

其中，#ifdef、#else 和#endif 都是关键字，<程序段>是由若干条预处理命令或语句组成的。这种形式的含义是：如果标识符已被#define 命令定义过，则编译<程序段 1>，否则编译<程序段 2>。

格式 2：

```
#ifndef <标识符>
    <程序段 1>
```

```
[
#else
    <程序段 2>
]
#endif
```

这与前一种形式的区别仅在于，如果标识符没有被#define 命令定义过，则编译<程序段 1>，否则就编译<程序段 2>。

格式 3：

```
#if <表达式 1>
    <程序段 1>
[
#elif <表达式 2>
    <程序段 2>
    ...
]
[
#else
    <程序段 n>
]
#endif
```

其中，#if、#elif、#else 和#endif 是关键字。它们的含义是，如果<表达式 1>为“真”就编译<程序段 1>；否则如果<表达式 2>为“真”就编译<程序段 2>，…；如果各表达式都不为“真”就编译<程序段 n>。

1.3 C++异常处理

程序中的错误通常包括：语法错误、逻辑错误和运行时异常（Exception）。其中，语法错误通常是指函数、类型、语句、表达式、运算符或标识符的使用不符合 C++中的语法，这种错误在程序编译或连接时就会由编译器指出；逻辑错误是指程序能顺利运行，但是没有实现或达到预期的功能或结果，这类错误常需要通过调试或测试才能发现；运行时异常是指在程序运行过程中，由于意外事件的发生而导致程序异常终止，如内存空间不足、打开的文件不存在、零除数、下标越界等。

异常或错误的处理方法有很多，如判断函数返回值、使用全局的标志变量、直接使用 C++中的 exit()或 abort()函数来中断程序的执行。

1.3.1 使用 C++异常处理

程序运行时异常的产生虽然无法避免，但可以预料。为了保证程序的健壮性，必须在程序中对运行时异常进行预见性处理，这种处理称为异常处理。

C++提供了专门用于异常处理的一种结构化形式的描述机制 try/throw/catch。该异常处

理机制能够把程序的正常处理和异常处理逻辑分开表示，使得程序的异常处理结构清晰，通过异常集中处理的方式，解决异常处理的问题。

1. try 语句块

try 语句块的作用是启动异常处理机制，侦测 try 语句块中的程序语句执行时可能产生的异常。如有异常产生，则抛出异常。try 的格式如下：

```
try
{
    // 被侦测的语句...
}
```

语句块

注意： try 总是与 catch 一同出现，在一个 try 语句块之后，至少应该有一个 catch 语句块。

2. catch 语句块

catch 语句块用来捕捉 try 语句块产生的异常或用 throw 抛出的异常，然后进行处理，其格式如下：

```
Catch（形参类形[形参名]）
{
    // 异常处理语句...
}
```

语句块

其中，catch 中的形参类型可以是 C++基本类型（如 int、long、char 等）、构造类型，还可以是一个已定义的类的类型，包括类的指针或者引用类型等。如果在 catch 中指定了形参名，则可像一个函数的参数传递那样将异常值传入，并可在 catch 语句块中使用该形参名。例如：

```
try
{
    throw "除数不能为 0！";
}
catch(const char * s)                    // 指定异常形参名
{
    cout<<s<<endl;                       // 使用异常形参名
}
```

注意：

(1) 当 catch 中的整个形参为“...”时，则表示 catch 能捕捉任何类型的异常。

(2) catch 前面必须是 try 语句块或另一个 catch 块。正因如此，在书写代码时应使用这样的格式：

```
try
{
    …
} catch(…)
{   …
```