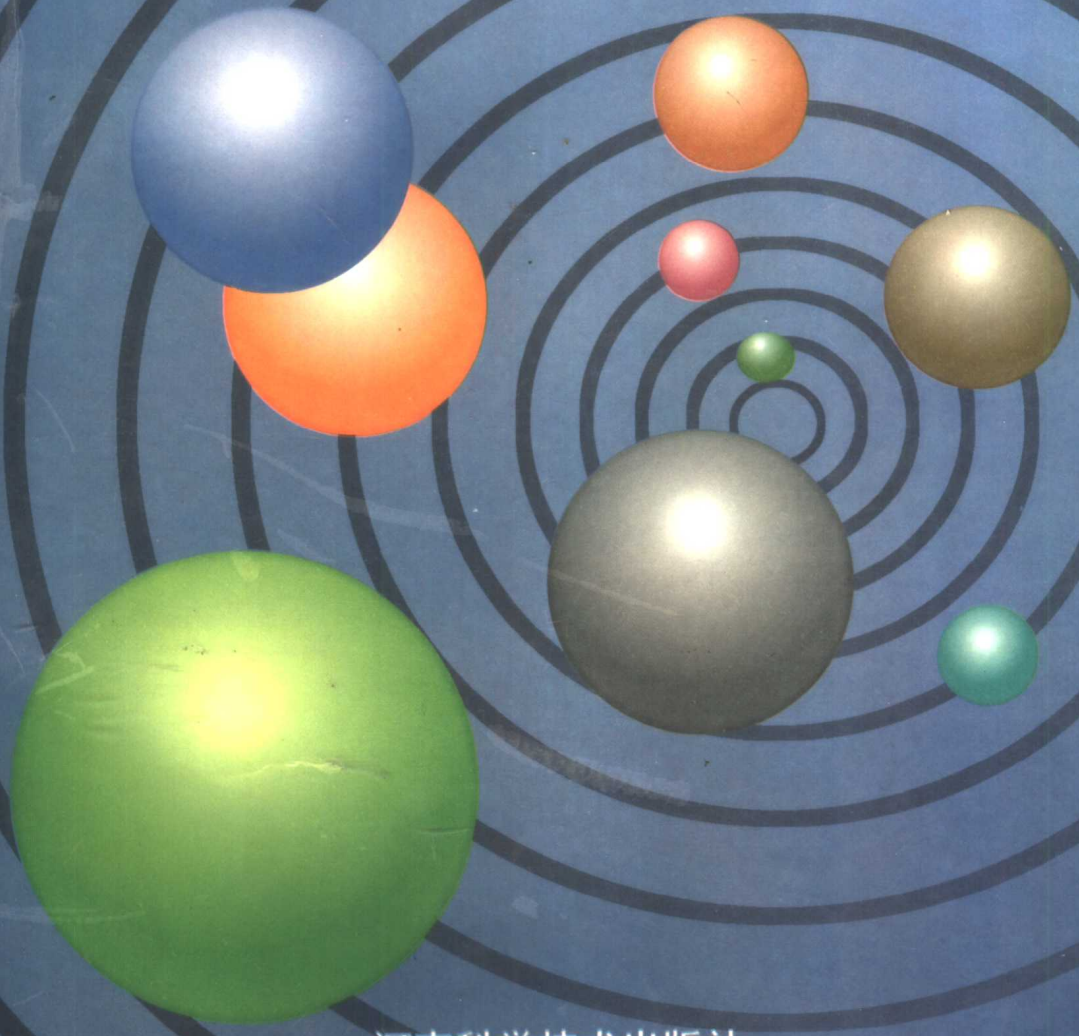
RUANJIAN KAIFA JISHU
YU SHIJIAN

软件开发技术与实践

陈震秋 等著



河南科学技术出版社

软件开发技术与实践

陈震秋　等著

河南科学技术出版社

内 容 提 要

本书根据作者长期从事大型软件开发工作的经验，较系统地阐述了软件开发中所涉及各种技术的现状、特点及适用情况和软件工程在实际工作中的应用。本书分三编，第一编重点讨论软件开发中的技术问题；第二编讨论软件工程的应用，并结合具体实例进行分析说明；第三编讲述了四种典型软件——CAD软件、管理软件、CAE软件及仿真软件的开发特点、技术应用等问题。本书对大专院校计算机系、应用数学系、电子工程系及相近专业的学生和广大软件爱好者有较大的参考作用

软件开发技术与实践　陈震秋 等著　责任编辑 王广照

河南科学技术出版社出版

郑州农业路73号

邮政编码：450002　电话：（0371）5721450

河南省漯河内陆特区报社印刷厂印刷

全国新华书店发行

开本：787×1092 1/16　印张：12.25　字数：262千字

1997年11月第1版　1997年11月第1次印刷

印数：1—3000

ISBN7-5349-2018-3/T·414　定价：13.00元

前　言

在我国，计算机技术的研究和软件开发，起步很早，发展很快，也取得了丰硕成果。中华民族的思维特点，逻辑判断的条理性，特别适合软件开发，对新软件的学习、掌握很快，在小型程序开发上都很出色。从70年代末开始，国家投入大量人力、物力从事大型应用软件系统开发，取得了很大的成功。但是到目前为止，在软件市场上，国产软件的市场占有率不高，在应用软件方面，缺乏著名的、受公众信赖的大型软件，更缺乏大型集成应用软件系统。国产软件形成不了力量，商品化转换困难，其原因很多，作者只是想根据多年来从事大型软件开发工作的经验，探讨一下软件开发的规律，重点放在具体实践中。希望本书能起到抛砖引玉的作用，使大家重视对大型软件开发的研究，为软件的国产化做点贡献。

软件开发属于应用技术研究领域，它紧密依赖于软件应用对象、采用的开发技术、开发人员的素质、软硬件环境等。为了更好地说明问题，这里用一个例子作类比。假设我们打算从甲地去乙地，作为基础理论研究，就要找出从甲地到乙地有几条路可行，其中各条路的特点如何，同样也要发现有几条路是死路，是不可行的。再假设从甲地到乙地有两条路，道路1平坦，上下坡少，但比较绕远；道路2路途最短，但上下坡多，较难走。作为应用科学研究，就要设法找出两条路间的连接通道，在两条道路各选择一些路段，形成最佳组合，使整个路线既不是最短，也不是太远，既不是最平坦，但也不算太难走，基本可以接受。本书第一编软件开发技术和第二编软件工程和质量保证中第一到第七章就是为解决这个问题而编写的。当你想从甲地去乙地，就要根据你的交通工具及身体素质作出选择。如果你的交通工具是一辆普通的自行车，刹车较好，身体素质一般，选路时，上坡尽量选坡度比较平缓的，哪怕多走一些路，下坡可选坡度较大的，路要短一些。若你的车具多级变速，身体素质较好，但刹车较差，那么选路时正好相反。也就是技术在具体应用中还要根据自身条件再作选择。第二编第十六章和第三编应用软件开发就是阐述这个问题的。

作者自从事软件开发工作以来，有幸参加多个大型软件开发工作。其中主要的有“飞机结构系统最优化设计系统”，该系统有50～60人参加，作者为一般开发人员；中国和德国合作的“CADEMAS”项目，该项目高峰时有70多人参加，作者任其中一个子项目的项目经理；对本书的形成起较大作用的是大型软件移植项目，该项目高峰时有近百人同时工作，由傅有光研究员任主任工程师，作者任副主任工程师。该移植项目引入了较多的新思想、新技术、新方法，在项目总体规划、软件工程的应用上，做出了显著成绩。作者通过对这些项目中的实际感受及取得的经验、体会进行分析，归纳，总结，把在实践中被证明是行之有效的东西贡献给读者，以供参考。本书致力于从软件开

发技术和软件工程在实际中应用的角度去解决编制程序和软件中的问题，不是囿于某种具体语言、而是在“方法”上将作者的宝贵经验以简明的语言娓娓道来。无论是从事软件开发，还是编制一些实用程序，相信您都可从本书中获得有益的思想和技巧，避开许多弯路。

参加本书编写工作的还有其他有丰富软件开发经验的人员，其中第三编第十七章(CAI)软件开发是由四川联大殷国富教授编写，第一编第五章5.5节STEP技术简介是由北京斯泰普产品数据中心（C－STEP）林晓星同志编写，第一编第八章由牟光灿副总工程师编写，第三编第二十章仿真软件开发由刘军同志编写，第二编第十三章由张晓光同志编写，第三编第十八章管理程序开发是由孔竟同志根据作者和赵桂香同志合写同名文章改写，第一编第一章1.1节常用编程语言简介由汤橘同志编写。

本书原稿由章远富同志作文字校对，金蓉同志完成全部录入工作，汤橘同志完成插图绘制及原稿排版。本书是集体劳动的结晶。

作者

1997年6月

目　录

第一编　软件开发基本技术

第二编 软件工程与质量保证

第三编　应用软件开发

引　言

Ⅰ．软件开发的特点

什么是软件开发？软件开发的本质是开发者把要让计算机完成的工作，用计算机所能接受的语言（这里称为编程语言）全面、完整、准确地描述出来。由此可见软件开发与文学创作有很大的类似性。要写出一篇好文章有两个必要条件：一个是要有生活，有充实的内容；另一个是有写作技巧。要编出一个好的软件，同样首先要求对所编软件的应用对象要有很深入的了解，要能把应用需求正确地抽象成数学模型；其次在软件开发过程中要能主动、正确地应用各种软件开发技术，提高所开发软件的质量。目前在软件开发中对数学模型的建立比较重视，且作了大量研究，但对软件开发技术本身却重视不够，不太注意“写作技巧”。本书就是针对这一现状，致力于解决“写作技巧”这个问题而编写的。

1．软件开发中的主要问题

文学创作的读者是人，人是有思维及逻辑推理能力的。因此文章写作就比较自由，有时文章可以写得比较朦胧，以留给读者自己想象回味的余地，即使文章中出现句子不通或用词错误也不会影响读者对文章的理解。另外写文章用的是自然语言，表达能力强，容易理解。与文学创作不同，软件的读者是计算机，目前计算机的智能化程度还比较低，因此要求程序必须绝对正确与完整。加上当前普遍使用的编程语言仍属于过程化语言，这种语言的特点是比较死板与繁琐，程序编制中开发者必须把每个细小操作都要完整地进行描述，在开发中甚至一个符号用错都会引起软件的潜在故障。不言而喻，用这种编程语言开发出来的程序其可读性也是很差的，由此给软件开发带来了很多问题。

1）开发效率低。由于目前所使用的编程语言为过程化语言，在程序编制时，对每个操作都必须作繁琐的描述，因此在软件开发时动辄几万行语句，对大型软件或软件系统往往可达几十万或几百万行语句。其所需开发人年数也会高达几百个人年。

2）可靠性差。一个几十万行语句的程序可能会包含几百万个符号，这些符号在编写、录入中难免会出错。另外在几十万行语句的编制中也难免发生思维与记忆的错误。这些就造成了软件的潜在隐患。这里有句行话：“程序能正确运行某些例题不是偶然的，但程序存在故障是必然的”。要消除程序中的隐患，使一个软件成熟，这是个非常费工费力的任务，也是使软件开发人员大为头痛的一项工作。

3）可读性差。由于编程语言层次较低，过程化程度高，为完成一个特定功能往往需要很多条语句。在读程序时好像在一个大森林中摸索道路，动不动就会迷失方向。在实际工作中常会感到读通一个程序有时比重写一个更费劲。在软件开发者中有句笑话：

在软件开发中对程序只有我与上帝知道，经过较长时间后，只有上帝才知道。由于可读性差，在软件开发中，开发者之间的工作协调、配合困难，对程序维护带来的问题就更大了。

4）开发成本高。如上所述，软件开发效率低、可靠性差、成熟周期长，影响着软件开发成本，还有两个因素也在较大程度上影响开发成本。①早期发现问题困难。一般软件开发中需完成语句量的50%～60%，才有可能上机运行，也就是说从顶层设计到正确性验证中间有较大的时间跨度，同时已投入相当大的工作量。一旦验证中发现错误，要对程序进行修改，所花的工作量也大，将有很大一部分劳动变为无效劳动。在软件开发中早期发现问题很重要，问题发现越早，修改所花工作量也就越小，浪费劳动也越少，但是在软件开发中早期发现问题又是一个很困难的工作。②实用化周期长。目前在软件开发中对软件需求往往是根据应用中手工操作模式结合一些典型应用情况而形成的。等软件开发出来，在计算机上实际运行时，就会发现要使软件真正能满足实际应用要求还有大量的修改与功能扩展工作要做。有时这个修改与功能扩展的工作量会大于原始的开发工作量。目前在国内软件开发成本计算时往往忽视了这部分工作的成本，这也是影响国产软件商品化的一个因素。

5）可维护性差。软件可维护性差主要是由于可读性差引起的。程序越大、结构越复杂，维护就越困难。另外在软件开发中，不注意软件开发技术应用，有些程序修改会很困难，牵一发而动全身，一个地方被修改，会引起多处地方的连锁反应，就更增加了维护难度。

在软件开发中，这些问题的根源在于开发工具，主要是编程语言。目前常用的语言主要是过程化语言。这种语言语义不够丰富，语言也不够精练，为完成一个操作要用多条语句，这样就带来了上述问题。在目前短时间内，编程语言还不可能有很大突破，为较好地解决上述问题，在软件开发中必须注意软件开发技术和软件工程的应用，这也是本书讨论的重点。

2．软件质量评估

如何评价一个软件质量好坏，这方面的文章很多，这里不再赘述。就程序而言，一个程序编得好坏的标准也是随着计算机软硬件及技术的发展而变化的。早期机器速度慢、内存小，因此比较重视算法优化及程序优化，哪个程序占用内存少，运行速度快就是好程序。随着软硬件的不断变化，在评价软件质量时，可读性成了重要指标，程序结构化程度成了主要评判标准。目前，根据实践经验，我们重点抓住下述三个指标：

1）可维护性。在一个软件的生命周期中，维护起着至关重要的作用。不可维护的软件是没有生命力的，维护在软件开发成本中占相当大的比例。因此一个软件可维护性的程度，对降低开发成本、延长生命周期是至关重要的。

2）开放性。由于软件开发成本高、开发周期长，所以在大型软件开发中往往采用滚动发展方式，边开发，边应用。开发一个模块，投入应用一个模块，逐步扩充完善。另外软件应用已由单个软件走向集成系统，程序集成也要求程序有较好的开放性。软件开放性涉及数据结构开放及程序结构开放，前者主要考虑容易与外来程序进行数据交换

及数据结构的可扩充性与可修改性，后者为程序模块修改与扩充提供较强的二次开发手段。

3）可移植性。近年来计算机硬件发展非常迅速，系统软件也有所发展。从操作系统看目前有向 UNIX 统一的趋向，但是技术发展总是“分久必合，合久必分”，不断变化的。为了延长软件的生命周期，扩大软件应用面及运行平台是非常必要的，因此软件必须具备可移植性。可移植性涉及程序可移植性与数据可移植性两方面。

Ⅱ．大型软件开发特点

小型软件开发时，由于投入人力较少（一般 1～3 人），因此开发效率关系不大。这类软件通常都由开发者自己维护，开发者可自由发挥个人特点，采用个人所喜欢的方法与技术，开发中任意性较大。这类软件的技术和质量水平主要依赖个人素质。

大型软件开发往往需要几百个人年，同时投入上百人工作，使软件开发变得非常复杂。首先要注意开发效率，开发效率的提高直接影响到开发成本及开发周期，因此在开发工作中，必须充分注意发挥每个开发者的个人特点及主观能动性，尽量避免重复劳动。其次，由于参加人员多，技术水平参差不齐，不能靠个人素质来保证软件的质量，必须要靠一整套措施，如技术规范、技术标准、软件工具、开发平台及质量保证措施来制约每个人的工作，也就是在开发中必须制约个性因素，扩大共性因素，以保证软件的统一性和规范化。另外要充分意识到维护的困难性，在开发时由几百个不同专业、类型的人员参加，但在维护时不可能把这个班子长期维持下去，必须把维护任务落实到少数几个人身上，此时维护工作会变得非常复杂与困难。最后考虑到软件将来的集成与移植，在顶层设计时必须给以充分的考虑和细致的规划。总之，在大型软件开发中，人员的组织与分工、技术的统一与约束、质量保证、项目管理就上升到重要位置，形成一个有机的整体。大型软件的开发，必须要根据实际情况，灵活应用软件工程的原理与方法，按系统工程思想来组织实施。

Ⅲ．软件开发展望

软件开发行业与机械制造工业有一定的类比性。可以设想一下在机械制造初期，只有锉刀、钢锯等工具来加工生产机械产品时，同样存在产品生产效率低、周期长、成本高、产品质量不易保证、产品结构的可理解性差等问题。但是在机械制造中下述三方面的突破大大促进了机械制造业的发展：

1）机械制图。制图学的完善与发展，大大增加了产品的可理解性，使产品结构的表示有了一个统一的语言。有了图纸就可容易地理解产品结构，对产品进行维护。图纸成了人们之间信息传递的有力手段。

2）标准件。标准件的应用，大幅度地提高了产品的共性部分。标准件大批量、低成本的生产大大降低了生产成本，缩短了生产周期。

3）加工机床。加工机床的出现，不但大幅度提高了生产效率，同时也极大地提高了产品质量。

当然机械产品看得见，摸得着，软件开发却是一项脑力劳动，情况要复杂得多，但是从机械制造的发展可以给我们一些启示：

1）表达方法。要重视对表达方法的研究，E－R 图的出现使数据结构的描述变得很清晰，从而推动了数据技术的发展。目前对程序表达方法的研究很多，但是还未找到一个理想的方法。

2）程序重用技术。仿造标准件的思想，使一些常用的子程序能容易地直接用于不同应用程序的开发中，从而避免重复劳动，提高开发效率。

3）软件工具和程序自动生成器。软件开发人员一直为别人应用计算机而奔忙，开发人员也要有意识地使用计算机辅助软件开发。软件工具和程序自动生成器能大幅度提高开发效率与软件质量。

一个好的表达方法，就可有效地解决软件的可读性，使软件更容易理解，并可促进软件开发人员间及软件维护人员的思想交流，从而提高了软件的开放性及可维护性。程序重用技术、软件工具和程序自动生成器，可大幅度提高软件开发效率，降低成本，软件开发的共性问题可集中统一解决，从而提高了软件的可靠性和可移植性。

总之在软件开发中，一定要注意开发技术的研究，技术的提高是开发效率与质量提高的基础。软件开发要尽可能走上工业化大生产的道路。

第一编　软件开发基本技术

第一章　编程语言

1.1　常用编程语言简介

1.1.1　过程化语言

目前较为常用的软件编程语言是 C 和 FORTRAN，它们都属于过程化语言。这类语言的特点是开发人员必须把他所想要完成的一个功能的全过程作出完整的描述。它的优点是通用性强，缺点是开发者必须把数据输入、数据存放、数据查找、数据计算、数据输出的每一个动作用编程语言一一描述出来，这就造成了开发量大，效率低的现象。用过程化语言开发程序，好像用锉刀、锯子等工具加工机械产品，由于工具层次较低，开发者要考虑的问题就比较多。一般用过程化语言开发程序，开发者要考虑程序结构安排、程序结构实现、程序变量安排、变量间协调，子程序调用时还要考虑参数传递一致性，对数组要考虑下标变量控制、防止数组越界，甚至小到作除法时，除数为 0，整型数相加时，整型数上溢等这些微小的细节。这些环节只要有一个地方考虑不周，就会对程序造成隐患，影响程序质量。由于语句太琐碎，可读性也差，要读通别人写的程序就比较困难，用过程化语言开发程序对开发者的编程风格依赖很重，造成开发者之间工作协调的困难。下面举一个例子来说明，这是一个矩阵乘法的 FORTRAN 程序：

```
      PROGRAM ARRMUL
      DIMENSION I1 (4, 5), I2 (5, 6), I3 (4, 6)
      N = 1
      K = 1
      L = 0
      DO 1 I = 1, 4
      DO 2 J = 1, 5
      I1 (I, J) = N
      N = N + 1
2     CONTINUE
1     CONTINUE
```

```
      DO 3 I = 1, 5
      DO 4 J = 1, 6
      I2 (I, J) = K
      K = K + 1
4     CONTINUE
3     CONTINUE
      DO 5 I = 1, 4
      DO 6 J = 1, 6
      DO 7 M = 1, 5
      L = L + I1 (I, M) * I2 (J, M)
7     CONTINUE
      I3 (I, J) = L
      L = 0
6     CONTINUE
5     CONTINUE
      END
```

由上述例子中可以看出，为了实现矩阵的乘法这一功能，就必须给出所需数据定义、数据赋值、数据计算等，用编程语言一一描述，开发效率较低。

1.1.2 非过程化语言

针对过程化语言的缺点，人们从20世纪70年代开始对非过程化语言作了大量研究。非过程化语言的特点在于开发人员只需关心他所要完成的功能而无需关心完成这一功能的每一个具体动作。属于这一类语言的有PROLOG、LISP及其他专用语言。这些语言的优点是可以提高效率，如PROLOG语言可自动安排数据结构，轻易地实现回溯查找与匹配，语言形式较规范，可读性也得到改善。这类语言应用针对性较强，用在人工智能、CAD等领域效果较好，但对某些领域却不是太方便，应用有一定局限性。另外用这类语言编程，在编程思想上有了很大变化，大部分软件开发人员还不容易适应。再加上这类语言开发的程序对机器资源要求较高，运行效率较低，从而影响了这类语言的推广使用。

1.1.3 面向对象语言

20世纪80年代初出现了以C++为代表的面向对象语言。计算机软件开发中时常被两大难题所困扰：一是如何克服程序的复杂性，二是如何在软件中更自然地表达客观世界，即对象模型。而以C++为代表的面向对象程序设计是软件工程学中的结构化程序设计、模块化、数据抽象、数据隐藏、知识表示、并行处理等各种概念的积累与发展，是当代解决上述两难题的最有希望的方法。面向对象程序设计是软件开发的一场变革，它代表了一种新的计算机程序设计的思维方法。该方法与一般结构程序设计的不同在于，它支持一种概念，即旨在使得计算机问题的求解更接近于人的思维活动。面向对象程序设计是软件系统的设计与实现的新方法。这种新方法是通过软件可扩充性和可重

用性，来改善并提高程序员的生产能力，并能控制维护软件的复杂性与软件维护的开销。C++ 表示了从C语言进化的特征，“++”是C的增量操作符，C++是C的扩充，即C++ 语言在提供面向对象程序设计的同时，又保留了C语言的高度简洁和高效率等优点。

面向对象语言的特征之一，是封装和数据隐藏把数据和过程紧紧联在一起，从而操纵数据的过程及函数的作用域与可视性限制于软件系统中的代码局部区域内。这样，数据和与之相关的函数过程变得不可分割。由此定义一个新实体：对象。该对象有自己的数据和处理过程，即用它自己的私有过程及函数，对对象单独存取并负责操纵所有的私有数据。用户可以通过精确定义方法的集合向该对象发送消息来处理这个对象。对象控制整个系统并向其他对象传送消息。一个面向对象软件系统的构造不依赖于对象的内部结构，而仅仅依赖于定义这个对象的方法，此方法能在对象内部数据上处理。

面向对象语言强调将同一作用（及对同一类对象）的数据成员及函数成员捆扎在一起，便于集中管理。这一点与以往的编程方式强调数据与语句体分开的模式截然不同。在C++ 中称这些有共同特性的数据和函数的结合体为“类”。如果大家对“类”这个概念还不太了解，那么我们就举一个较为形象的例子。市场上有很多长虹公司出品的彩电在销售，我们可以把这些彩电看成一个长虹彩电类，为什么它们可以是一个类呢？因为它们都具有以下几个共同特点：①它们都具有长虹的品牌。②这些彩电的元件大都可以通用。每一台长虹彩电都是这个类中的一个享有相同的属性和功能的构成对象。

前面说过C++ 的类是由数据成员和函数成员两部分组成的，那么我们的长虹彩电类也是由这两部分构成，彩电的选频器就好比是长虹彩电类的一个函数成员，而这时显示在屏幕上的电视内容就如同这个长虹彩电类的数据成员。在C++ 中的类可调用函数成员来改变数据成员的内容，正如我们在长虹彩电类中也可用选频器来改变屏幕上的电视内容一样。

“类的继承”这一特征，其实是很简单的一个概念，因为旧的类在发展到一定程度时可能已无法满足人们的需求，这时大家只有创造一个新的类，但这个类并不是凭空生成的，它是从旧的类（即父类）中改造出来的。这个新的类（即子类）除了继承父类的数据成员和函数成员外，它还有属于自己的特有数据成员和函数成员。为了便于大家了解，我们再回到长虹彩电类的例子中去。假设我们要设计一种新款的长虹彩电，这种新型彩电与原有的长虹彩电相比大部分功能设计都是相同的，只是多了一个画中画功能，我们定义这个新款彩电为一个新的长虹+类，这个新类实际上是原有的长虹彩电类的子类，它不仅从父类长虹彩电类中继承了数据成员（显示电视内容）和函数成员（选频器），而且自己定义了一个新的函数成员（画中画），从而实现了自己的设计要求。下面举一段具体的C++ 程序来说明。

```
enum BugColor (Red, Green, Blue, Yellow, Black, White);
Class Bug
    {
    Private:
```