计算机基础教育丛书

Visual Basic 程序设计教程

柴欣 李惠然 主编
范贻明 主审

海洋出版社

计算机基础教育丛书

Visual Basic 程序设计教程

主编：柴欣　李惠然

主审：范贻明

海洋出版社

2002 年 · 北京

《Visual Basic 程序设计教程》

编委会名单

主　编　　柴　欣　李惠然

主　审　　范贻明

副主编　　陈冀川　魏明军　张玉梅

编　委　　于　明　马建红　武优西　苑伯达

韩艳峰　侯立坤　刘洪普　刘靖宇

前　言

计算机科学技术的飞速发展，推动和促进了计算机基础教育的改革和发展。根据教育部对高等学校非计算机专业面向 21 世纪计算机基础教育的要求，按照三个层次教学课程体系的安排，学生在学完第一层次的“计算机文化基础”后，要提高到第二个层次“计算机技术基础”，在这一层次的教育中要求学生必须掌握一门程序设计语言。以往普遍采用 DOS 界面下的 FORTRAN、C、PASCAL 及 TURE BASIC、QUICK BASIC 等语言作为教学内容，由于目前各高校的“计算机文化基础”已全部提升到 Windows 平台，因此作为其后续课程的程序设计语言，自然也应在这一平台上深入下去。Visual Basic 作为 Windows 平台的开发工具，既继承了其先辈所具的有程序设计语言简单易学、易用的特点，又在编程系统中引入了面向对象的机制，用一种巧妙的方法把 Windows 编程的复杂性封装起来，提供了一种可视界面的设计方法，使用窗体和控件设计应用程序界面，从而极大地提高了用户开发应用程序的效率。因而非常适于在高校中作为教学之用。

本书作者有着丰富的教学经验和极强的科研能力，曾利用 VB 及其它开发工具开发了“计算机基础教学、考试、考务集成系统”、“河北省计算机职称考试系统”等软件，因此对 VB 有着较深入的理解。为了使初学程序设计的读者能快速地掌握开发基于图形界面的应用程序的能力和解决实际问题的能力，作者精选了 Visual Basic 的内容，本着加强基础、注重实践、勇于创新、突出应用的原则，力求使本教材达到可读性、适用性和先进性。为了便于读者自学，在全书的体系结构和内容上注意了由浅入深、深入浅出、循序渐进的方针。为了提高读者编程技巧，在大部分章节中都提供了典型例题。

本书共分 12 章，第 1、2 章介绍程序设计语言的发展及中文 VB 6.0 概述；第 3 章介绍 VB 程序设计基础；第 4 章介绍窗体及常用控件；第 5 章为对话框、菜单；第 6 章为过程、模块与类；第 7 章为常用算法；第 8 章文件；第 9 章为工程管理及环境设置；第 10 章为程序调试。为扩大读者的软件技术基础知识，或便于参加全国及省市的二级计算机水平考试，第 11 章介绍了数据结构的基本知识与算法；第 12 章介绍了软件工程的基本知识。

为了实现理论联系实际，达到良好的教学效果，配合本教程我们还编写了《Visual Basic 实验教程》与本教材相呼应，《Visual Basic 实验教程》在各章均相应地安排了若干个上机试验，以方便师生有计划有目的地进行上机操作，从而到达了事半功倍的教学效果。参加本书编写的有：李惠然、柴欣、于明、魏明军、陈冀川、张玉梅、马建红、武优西、苑伯达、韩艳峰、侯立坤、刘洪普、刘靖宇等。本书由范贻明教授主审。

本书写作时，参考了大量文献资料，在此向这些文献资料的作者深表感谢。

在本书的编写过程中得到了河北省高校计算机研究会理事长李凤翔教授、常务副理事长崔来堂教授等的大力支持和帮助，本书已被推荐为河北省高校计算机研究会的统编教材之一，同时得到河北工业大学及天津市有关高校的支持，编审者在此一并表示衷心的感谢。

计算机的发展日新月异，由于编审者水平所限，加之时间十分紧迫，书中定有不妥之处，恳请读者及有关专家批评指正。

编审者 2001 年 8 月于天津

目　录

第 1 章　程序设计概述

1.1　程序设计发展概述

1.1.1　计算机程序设计语言

计算机之所以能自动进行计算，是因为采用了程序存储的原理，计算机的工作体现为执行程序。程序是控制计算机完成特定功能的一组有序指令的集合，编写程序所使用的语言称为程序设计语言，它是人与计算机之间进行信息交流的工具。

从 1946 年世界上诞生第一台计算机起，在短短的 50 余年间，计算机技术迅速发展，程序设计语言经历了机器语言、汇编语言到高级语言的多个阶段。目前世界上已经设计和实现的计算机语言有上千种之多，但实际被人们广泛使用的计算机语言不过数十种。

计算机语言按其与硬件接近的程度，可以划分为低级语言和高级语言两大类。机器语言和汇编语言属于低级语言，它们分别被称为第一代语言和第二代语言；高级语言包括过程式语言和非过程式语言，称为第三代语言。

1.1.2　程序设计的发展历程

回顾程序设计发展的历史，大体上可以划分为如下几个不同的时期。

20 世纪 50 年代的程序都是用指令代码或汇编语言来编写的，这种程序的设计相当麻烦，编制和调试一个稍许大一点的程序常常要花费很长的时间，培养一个熟练的程序员更需经过长期的训练和实习，这种局面严重影响了计算机的普及应用。

20 世纪 60 年代高级语言的出现大大简化了程序设计，缩短了解题周期，因此显示出强大的生命力。此后，编制程序已不再是软件专业人员才能做的事了，一般工程技术人员花上较短的时间学习，也可以使用计算机解题。这个时期，随着计算机的应用日益广泛地渗透到各学科和技术领域，也发展了一系列不同风格的、为不同对象服务的程序设计语言。其中较为著名的有 FORTRAN、COBOL、ALGOL、LISP、PL/1、PASCAL 等十几种语言。高级语言的蓬勃兴起，使得编译和形式语言理论相应日趋完善，这是该时期的主要特征。但就整个程序设计方法而言，并无实质性的改进。

自 20 世纪 60 年代末到 70 年代初，出现了大型软件系统，如操作系统、数据库，这给程序设计带来了新的问题。大型系统的研制需要花费大量的资金和人力，可是研制出来的产品却是可靠性差，错误多，且不易维护和修改。一个大型操作系统有时需要几千人每年的工作量，而所获得的系统又常常会隐藏着几百甚至几千个错误。当时，人们称这种现象为“软件危机”。

“软件危机”震动了软件界，程序设计的传统习惯和工作方式导致了不清晰的程序结构，使得程序的可靠性难以保障；另一方面，程序设计工具的严重缺乏也使得大型系统的开发陷入困境。此时人们开始重新审视程序设计中的一些最基本的问题。例如，程序的基本组成部分是什么？应该用什么样的方法来设计程序？如何保证程序设计正确？程序设计的主要

方法和技术应如何规范等等。

1969 年，E.W.Dijkstra 首先提出了结构化程序设计的概念，他强调了从程序结构和风格上来研究程序设计。经过几年的探索和实践，结构化程序设计的应用确实取得了成效，用结构化程序设计的方法编写出来的程序不仅结构良好，易写易读，而且易于证明其正确性。

到 20 世纪 70 年代末结构化设计方法得到了很大的发展，Niklans Wirth 又提出了“算法+数据结构 = 程序设计”的程序设计方法，他将软件划分成若干个可单独命名和编址的部分，它们被称为模块，模块化使软件能够有效地被管理和维护，能够有效的分解和处理复杂问题。在 80 年代，模块化程序设计方法普遍被人们接受。

虽然几十年来结构化程序设计技术得到了广泛的使用，但有些问题仍未得到很好的解决。由于软件开发是对问题的求解过程，从认识论角度看，软件开发过程包括人们对要解决问题及相关事物的认识和基于认识所进行的描述。而结构化设计方法不能直接反映出人类认识问题的过程。另外，结构化设计方法中，程序模块和数据结构是松散地耦合在一起的，因此，当应用程序比较复杂时，容易出错，难以维护。随着计算机软件的发展，软件系统越来越复杂庞大，结构化程序设计方法已显得力不从心。

20 世纪 80 年代，在软件开发中各种概念和方法积累的基础上，就如何超越程序的复杂性障碍，如何在计算机系统中自然的表示客观世界等问题，人们提出了面向对象的程序设计方法。面向对象的方法不再将问题分解为过程，而是将问题分解为对象。对象将自己的属性和方法封装成一个整体，供程序设计者使用。对象之间的相互作用则通过消息传递来实现。用面向对象的程序设计方法，可以使人们对复杂系统的认识过程与系统的程序设计与实现过程尽可能地一致。有人预测，这种“对象+消息”的面向对象的程序设计模式将逐渐取代“数据结构+算法”的面向过程的程序设计模式。

1.2 结构化程序设计概述

1.2.1 结构化程序设计概念

自提出结构化程序设计的概念后，经过十几年的发展，结构化程序设计已经具有了很广泛的内容，大体上可以归纳为以下几点。

1. 结构化程序的基本结构

结构化程序包含有三种基本结构，人们可以用这三种基本结构来展开程序，表示一个良好的算法，从而使程序的结构清晰、易读易懂且质量好、效益高。这三种基本结构为顺序结构、选择结构和循环结构。

(1) 顺序结构

顺序结构是一种最简单、最基本的结构，在顺序结构内，各块是按照它们出现的先后顺序依次执行。图 1.1 表示了一个顺序结构形式，从图中可以看出它有一个入口 a 点，一个出口 b 点，在结构内 A 框和 B 框都是顺序执行的处理框。

(2) 选择结构

选择结构中包含一个判断框，根据给定的条件 P 是否成立而选择执行 A 框或 B 框，当条件成立时，执行 A，否则执行 B。A 框或 B 框可以是空框，即不执行任何操作，但判断

框中的两个分支，执行完 A 或 B 后都必须汇合在一起，从出口 b 退出，然后接着执行其后的过程。图 1.2 所示的虚线部分就是选择结构，在选择结构中程序产生了分支，但对于整个的虚线框而言，它仍然只具有一个入口 a 和一个出口 b。

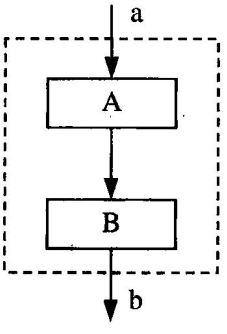

图 1.1　顺序结构示意图

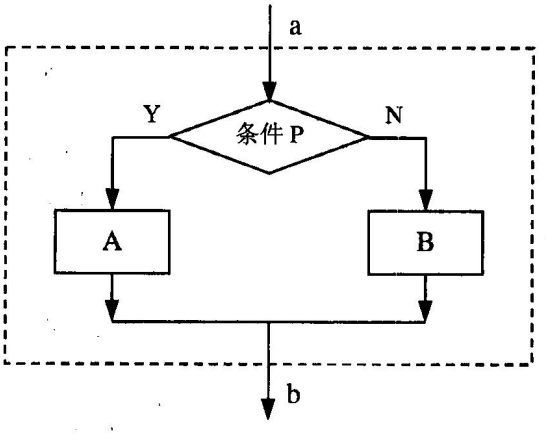

图 1.2　选择结构流程图

(3) 循环结构

循环结构又称重复结构，是指在一定条件下反复执行一个程序块的结构。循环结构也是只有一个入口，一个出口。根据循环条件的不同，循环结构分为当型循环结构和直到型循环结构两种。

① 当型循环的结构如图 1.3，其功能是：当给定的条件 P 成立时，执行 A 框操作，执行完 A 操作后，再判断 P 条件是否成立，如果成立，再次执行 A 操作，如此重复执行 A 操作，直到判断 P 条件不成立才停止循环。此时不执行 A 操作，而从出口 b 脱离循环结构。

② 直到型循环的结构如图 1.4，其功能是，先执行 A 框操作，然后判断给定条件 P 是否成立，如果不成立，再次执行 A 操作；然后再对 P 进行判断，如此反复，直到给定的 P 条件成立为止。此时不再执行 A 框，从出口 b 脱离循环。

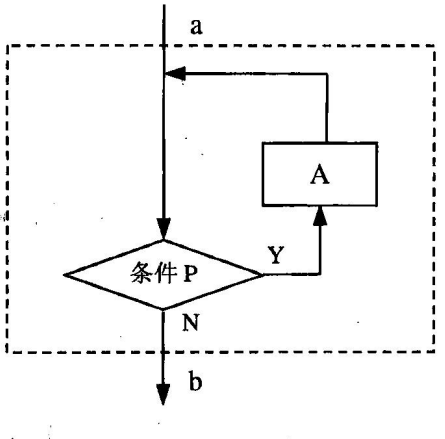

图 1.3　当型循环结构流程图

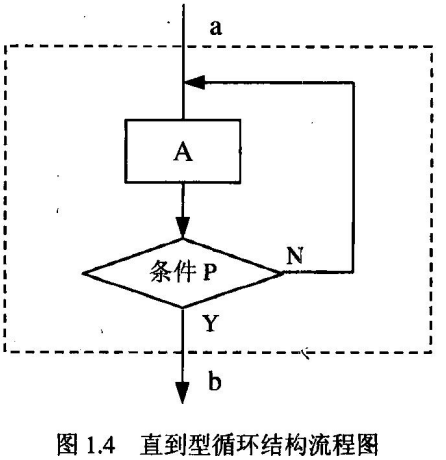

图 1.4　直到型循环结构流程图

由以上三种基本结构构成的程序，称为结构化程序。一个结构化程序，以及三种基本结构中的每一种结构都应具有以下特点：

- 有一个入口。
- 有一个出口。

• 没有死语句，即每一个语句都应该有一条从入口到出口的路径通过它（至少通过一次）。

• 没有死循环（无限制的循环）。

实践证明，任何满足以上四个条件的程序，都可以表示为由以上三种基本结构所构成的结构化程序；反之，任何一个结构化程序都可以分解为一个个基本结构。

2. 结构化程序的设计方法

结构化程序主要采用自上而下、逐步细化的设计方法，即先全局后局部、先整体后细节、先抽象后具体的设计方法。由于实际问题往往比较复杂，为了提高效率，在进行结构化程序设计时，常常伴随着使用关键部分优先考虑的设计方法。

3. 结构化程序的组织结构

在结构化程序中常常用模块化结构来组织程序，图 1.5 给出了用模块化结构组织程序的示意图。

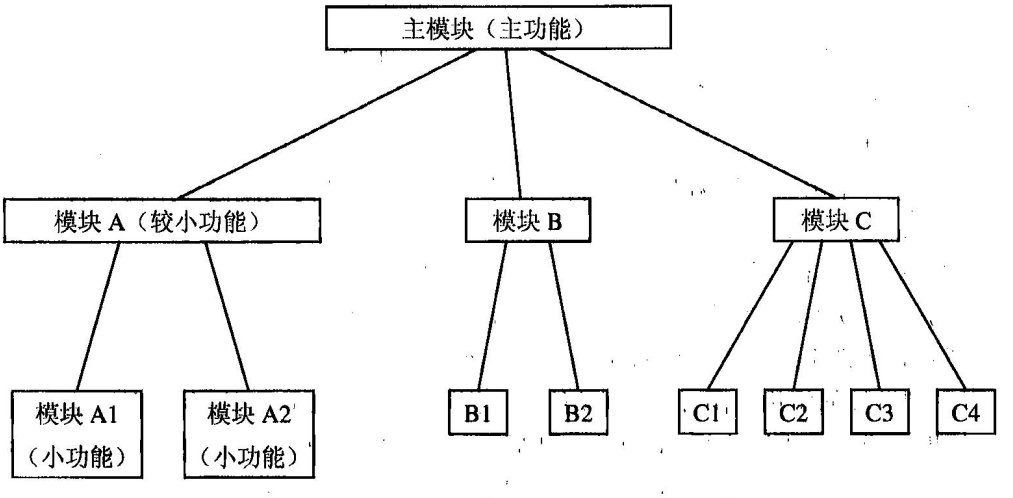

图 1.5　程序模块化分图

1.2.2　结构化程序设计方法

1. 逐步细化法

将一个完整的问题分解成若干相对独立的问题，只要这些问题能分别得到正确的解决，整个问题也就解决了。子问题又可进一步分解为若干子问题，这样可以一直重复下去，直到每个问题都已简单到我们满意的程度。对每步分解，都要作出分解方法的决策，不同的决策会导致不同的解法。把这种程序设计方法称之为逐步细化法，也就是在编写程序时一步一步地不断细化的过程。

细化过程可以自顶开始向下进行，也可以从底端开始向上进行。根据经验，程序自顶向下设计，再不断细化这种处理方法效果较好。细化过程的技术有三种，分别称为“划分和解决”的分割技术、“作出有限进展”的递推技术和“分析情况”的分析技术。

(1) 分割技术

分割技术大致可以分为以下几个步骤：

第一步：把问题划分成不相交的一些部分，直到可用复合语句为止。

第二步：依次解决划分后的每一部分问题。

(2) 递推技术

对问题作出有限进展，直到用循环语句实现。如果我们对问题找到一个解的方向，并作出了有限进展，这时再重复递推，直到最终达到完全解。这是我们在程序设计时常用的方法。

(3) 分析技术

对问题用情况分析来精细化，直到可用条件语句为止。

2. 模块法

模块化设计技术与方法，是程序设计中应用最早的一种重要方法，它早在使用低级语言时期就已出现，但是，只有在结构化程序设计中，这种技术与方法才得以发展、充实、提高与完善。

模块化程序设计方法是指在程序设计中，将一个复杂的算法（或程序）分解成若干个相对独立、功能单一的模块，利用这些模块即可适当地组合成所需的全局算法（或程序）。这里所说的模块，是一个可供调用（即让其他模块调去使用）的相对独立的操作块（或程序段），每个模块都是由三种基本结构组成的结构化模块。在模块化结构中，整个系统犹如搭积木一样，是由各模块适当组合而成。模块之间的相对独立性，使每个模块均可各自独立地分别进行分析、设计、编写、调试、修改和扩充，并且不会影响到其他模块和全局算法（或程序），这表明模块化结构不仅使复杂的软件研制工作得以简化，缩短开发周期，节省开发费用，提高软件质量，而且还可以有效地防止模块间错误的扩张，增加整个系统的稳定性与可靠性。同时还可使软件具备结构灵活、便于组装、层次分明利于维护、条理清晰容易理解的优点。因此，模块化程序设计方法使结构化程序设计的重要方法之一，主要用于算法设计阶段。

1.3 面向对象程序设计概述

结构化程序设计技术虽已使用了几十年，但如下问题仍未得到很好的解决。

(1) 这种面向过程的设计方法与人们习惯的思维方法仍然存在一定的差距，所以很难自然、准确地反映真实世界。因而用此方法开发出来的软件，有时很难保证其质量，甚至需要进行重新开发。

(2) 该方法实现中只突出了实现功能的操作方法（模块），而被操作的数据（变量）处于实现功能的从属地位，即程序模块和数据结构是松散地耦合在一起的。因此，当应用程序比较复杂时，容易出错，难以维护。

由于上述缺陷，结构化程序设计方法已不能满足现代化软件开发的要求，一种全新的软件开发技术应运而生，这就是面向对象的程序设计（Object Oriented Programming，简称OOP）。

1.3.1 面向对象的程序设计概念

面向对象的程序设计在 20 世纪 80 年代初就提出了，它起源于 Smalltalk 语言。用面向

对象的方法来解决问题，不再将问题分解为过程，而是将问题分解为对象。对象是现实世界中可以独立存在、可以被区分的一些实体，也可以是一些概念上的实体，世界是由许多对象组成的。对象有自己的数据（属性），也包括作用于数据的操作（方法）。对象将自己的属性和方法封装成一个整体，供程序设计者使用。对象之间的相互作用通过消息传递来实现。有人预测，这种“对象+消息”的面向对象的程序设计模式将逐渐取代“数据结构+算法”的面向过程的程序设计模式。下面介绍一些面向对象的程序设计中经常用到的一些术语。

① 对象：是属性和服务的封装体。

对象的属性用于描述对象的静态数据特征，对象的属性可用系统的或用户自定义的数据类型来表示，也可用抽象的数据类型表示。对象属性值的集合称为对象的状态（state）。

对象的服务用于描述对象的动态特征，也称之为行为或功能，它是定义在对象属性基础上的一组操作方法（method）的集合。

② 类：是对象的抽象及定义，是具有共同属性和操作的多个对象的相似特征的统一描述。类就成为具有共同特征的对象的集合，而对象只是类的一个实例。

③ 消息：是面向对象系统中实现对象间的通信和请求任务的操作。消息传递是程序运行的基本处理活动。

当然，面向对象的程序设计并不是要抛弃结构化程序设计方法，而是站在比结构化程序更高、更抽象的层次上去解决问题。当它分解为低级代码模块时，仍需要结构化编程技巧，但是，它分解一个大问题为小问题时采取的思路与结构化方法是不同的：

1.3.2 面向对象的程序设计语言

早在 20 世纪 60 年代，就出现了最早的面向对象的程序设计语言 Simula67 语言，具有了类和对象的概念，随后又推出了纯面向对象的程序设计语言。如 80 年代美国 Xerox Palo Alto 研究中心推出了 Smalltalk，它完整地体现并进一步丰富了面向对象的概念，还开发了配套的工具环境，为其最终实用化奠定基础。但由于当时人们已经接受并广泛应用结构化设计理论，而不能一下子完全接受面向对象程序设计的所有思想等诸多原因，这些语言并没有能够广泛流行起来。后来人们开始对流行的语言进行面向对象的扩充，曾经推出过许多种类的版本，而主要成功的代表是在当时流行的程序设计语言 C 语言的基础上开发的 C++语言。C++是一门高效实用的混合型程序设计语言，它是由 AT&T 公司的贝尔实验室的 Bjarne Stioustrup 博士开发的，它最初的设计目标是：支持面向对象编程技术；支持抽象形态的类；更好的 C 语言。C++语言包括两部分：一是 C++基础部分，它是以 C 语言为核心的；另一部分是 C++面向对象特性部分，是 C++对 C 语言的扩充部分。这样它既支持面向对象的程序设计方法，又支持结构化程序设计方法，同时广泛的应用基础和丰富的开发环境的支持，也使面向对象设计得到很快普及。

随着面向对象的程序设计的普及，面向对象的程序设计语言已形成几大类别：一类是纯面向对象的语言，如 Smalltalk 和 Eiffel；另一类是混合型的面向对象语言，如 C++和 Objective C；还有一类是与人工智能语言结合形成的，如 LOOPS、Flavors 和 CLOS；适合网络应用的有 Java 语言等。随着基于图形界面操作系统 Windows 的流行，面向对象的程序设计方法也开始与可视化技术相结合，其典型代表是 Visual C++和 Visual Basic。这些面向对象的程序设计语言虽然风格各异，但都具有共同的概念和编程模式。

1.3.3 面向对象的程序设计方法与结构化程序设计方法的比较

结构化方法强调了功能抽象与模块化，程序（program）是一些过程（procedure）或子过程（Subroutine）的集合，这些过程通过参数传送数据。结构化的分解突出过程，强调的是如何做（How to do？），代码的功能如何完成。因而可以把结构化方法看作是处理一系列问题的过程，也就是以模块（即过程）为中心的开发方法。

在面向对象的系统中，世界被看成是独立对象的集合，对象之间通过“消息”相互通信，对象具有“智能化”的结构，它将数据和消息“封装”（Encapsulation）在一起，对一个对象的存取或修改仅通过其外部接口子程序，其内部的实现细节、数据结构及对它的操作是不可见的。面向对象的分解突出现实世界和抽象的对象，强调的是做什么（What to do？），它将大量的工作由相应的对象来完成，程序员在应用程序中只需说明要求对象完成的任务。因此面向对象是以对象为中心的开发方法。

以对象为中心的面向对象的程序设计方法较之以过程为中心的结构化程序设计方法更为优越。具体表现在如下几个方面：

⑴ 符合人们的习惯的思维方法

由于对象对应用于现实世界的实体，因而可以很自然地按照现实世界中处理实体的方法来处理对象，软件开发者可以很方便地与问题提出者进行沟通和交流。

⑵ 易于软件的维护和功能的增减

对象的封装性及对象之间的松散耦合性，都给软件的修改和维护带来了方便。

⑶ 可重用性好，易于扩充

重复使用一个类（类是对象的定义，对象是类的实例化），可以比较方便地构造软件系统，加上其具有的独特的继承性和更丰富的多态性，极大地提高了软件开发的效率。并使软件更易于扩充，能很好地适应复杂大系统不断发展与变化的要求。

⑷ 与可视化技术相结合，改善了工作界面

随着基于图形界面操作系统的流行，面向对象的程序设计方法也将深入人心。它与可视化技术相结合，已经使人机界面进入 GUI 时代。

⑸ 支持复杂大系统的分析与运行

由于采用了消息传递机制作为对象之间相互通信的惟一方式，这样就使消息传递的机制能很自然地与分布式并行程序、多机系统、网络通信等模型取得一致，从而强有力地支持复杂大系统的分析与运行。

习题 1

1.1 机器语言、汇编语言和高级语言各有什么特点？

1.2 结构化程序设计的三种基本结构是什么？请画出三种基本结构的流程图。

1.3 结构化程序设计的在哪些方面仍存在不足？

1.4 面向对象的程序设计方法的优越性表现在哪些方面？