〔美〕B. W. 利夫菲克 著

软件开发者必读

科学出版社

软件开发者必读

[美] B. W. 利夫菲克 著

潘正伯　毛家骅　陈宏志 译

赵长风 校

科学出版社

1990

内 容 简 介

本书根据作者多年从事软件开发的经验和教学实践，按照软件工程的方法，系统而又详尽地揭示了软件开发的奥秘.

本书的内容包括系统分析，结构化设计思想，结构化编程方法，软件的测试与调试，文档的编制以及系统的维护. 书中除了必要的理论阐述之外，还揉进了作者多年从事软件开发的经验，结合对大量实例深入浅出的讲解，为广大微机用户，特别是为非计算机专业的软件开发人员提供了迅速掌握软件开发的方法.

此外，本书的附录中给出了文档的完整实例，便于读者参考.

本书的读者对象为微机用户、非计算机专业的软件开发人员及与此有关的工程技术人员、大专学生等.

Blaise W. Liffick
THE SOFTWARE DEVELOPER'S SOURCEBOOK
Addison-Wesley, 1985

软件开发者必读

〔美〕B. W. 利夫菲克 著
潘正伯 毛家麟 陈宏志 译
赵长风 校
责任编辑 那莉莉
科学出版社出版
北京东黄城根北街 16 号
邮政编码：100707
中国科学院印刷厂印刷
新华书店北京发行所发行 各地新华书店经售
*
1990 年 11 月第 一 版 开本：787×1092 1/16
1990 年 11 月第一次印刷 印张：14 3/4
印数：0001—5000 字数：335000
ISBN7-03-002013-8/TP・150
定价：8.50 元

译 者 的 话

近十年来，我国的计算机事业，尤其是计算机的应用得到了空前的发展。使用计算机的网点在许多系统、行业、单位和部门里，从无到有、从小到大地建立起来。

回顾近年来我国计算机事业的发展，并将它与美、日、英、法等国作一简单比较，可以很明显地看出，美、日等国搞计算机的历史长、投资多，它们由中型机（当时曾被视为巨型机）搞起，其开发的目标很广，但主要集中于大型机和巨型机。

我国与这些国家的情况不同，我们有自己的特点。我国的第一个特点是计算机的需用量大，但资金有限；第二个特点是我们目前普遍使用的计算机，绝大多数是近十年来购置的微型机和袖珍机；第三个特点是从事计算机开发工作的人，大多是近几年培养出来的新手。从这些特点出发，我们看中了本书。

本书的作者基于自己多年积累的经验，讲解软件开发的基础知识和技巧。他把模块化程序设计方法和软件工程的思想具体化，举了很多例子，深入浅出，通俗易懂，切合实际。特别难得的是作者所选定的开发环境和开发工具，正是我国程序员、科技人员和大中专学生最熟悉的 BASIC 语言。因此不论是训练有素、经验丰富的高级程序员，还是未经系统培训的业余爱好者都能看得懂、用得上。一旦用上，就能收到实效，从而有助于将他们所从事的软件开发工作做得更好、更快。

在本书的翻译过程中，对引言的内容作了适当删减。

由于译者水平所限，译文难免有错误和欠妥之处，望读者不吝赐教。

引　言

软件开发(通称编程)被认为是一门艺术、一门科学，也有人认为它是一种工程训练.这三种看法都有一定的道理，而且它们之间并不是互斥的.

作为一门艺术，编程是一项既有韵味又极赋创造性的工作.我们不难想象，程序员就像艺术家一样，他们要用大量的时间以一种不仅十分灵巧(甚至独具特色)而且便于使用的语序将代码编写出来.在这一点上，他们或许更像设计师或建筑师.

许多计算机专家，特别是我们这些在高等学校工作的人，一直主张把这一领域称为计算机科学.确切地说，编程是应用科学法则的一个具体过程.

理论上的计算机科学属于数学的范畴，实际上它是数学科学的必然产物.

编程可能更类似于工程学.工程学是一门应用科学，涉及建筑、机械、产品、系统以及有益于人类的一切方面.在计算机领域，软件工程这个术语近来得到日益普遍的应用，因为它指出软件开发必须采用结构化的方法.这样的开发需要技巧、方法，并需预先构造出模块.

那么，为什么本书不叫"软件工程入门"呢？第一，软件工程这个术语只是新近才出现的，尚无严格的定义；其次，有关软件工程的现有流行课本都是以初级或高级的计算机科学为主要标准编写的，因而呈现出明显的计算机科学的背景；第三，本书虽涉及若干软件开发的专题，但还只属软件工程的外围.

可以给本书加上一个奇特的副标题：*供教师、学生、小型企业用户、家庭计算机使用者以及爱好计算机的人们用的自顶向下、自底向上、模块化、结构化、运用工程学编程的总体方法的综述*.韦氏大词典对整体性定义为"不只是对单个元素取其总和，而且要按总体上相互影响的关系来正确地看待宇宙的理论."对此可理解为"整体大于它的个体之和".正确使用整体性一词，可以更好地、更全面地描述计算机科学中的复杂问题.

在软件开发的各个阶段，所使用的各种技巧都有各自的拥护者.易被忽视的是，这些技巧并不是互斥的，在整个开发过程中它们是互补的.开发一个具体的软件不一定会用到这里所讨论的每种技巧，这点即使专业程序员也概莫能外.因为现有的书籍往往至多只涉及这里所讨论的一两个题目.除了计算机科学家之外，任何人要找到合适的参考资料以解决软件开发者的困惑，几乎都是不可能的.

本书主要是供业余爱好者，即非专业程序员使用的.虽然这些人都是非专职程序员，但却不能说他们就不需要熟练的编程技巧.这些非专业程序员要为自身的应用而编写程序.然而无论硬件厂家、软件卖主或出版商都没有给他们提供足够的手段，以便实际地开发正确、优质而又有用的软件.这些人越来越意识到，他们需要弄懂一些 BASIC 语句是怎样工作的，以及程序又是如何用它们编写出来的.

本书将在帮助读者学会编程方面起重要作用.第一，作为一本入门读物，它提供了各种技巧以帮助程序员更好地工作；第二，本书提出了有效地评价软件产品的标准，使得读者能够成为更加博学的软件用户(消费者)；第三，用各种方法讨论的一些专题是"长期有

效的”，而且不受限于任何具体的计算机．在今后许多年内本书将作为一本优秀的参考书发挥作用．

如上所述，本书的内容不仅适用于微型计算机，而且也适用于大型计算机系统．事实上，某些技巧就是从大型机的环境中借用过来，而且已普遍为计算机专家所采用．当然，本书的一些内容结合微型机讲解更容易说明白．

此外，许多例子，特别是第三、四、五章中的例子，都采用了BASIC语言，代码格式也力求通用化，以便使这些例子能方便地应用于大多数微机系统．程序编写则采用 Microsoft BASIC，以便于程序直接在 IBM PC，DEC Rainbow 100 以及任何基于 CP/M 的系统上使用．这些例子也适用于与 Apple II 系列兼容的计算机．虽然一种具体的技巧用BASIC实现的细节在不同的机器上不尽相同，但所用的基本概念却是一致的．所以其它 BASIC 语言的用户也能毫无例外地采用本书所阐述的技巧．

此外，其它编程语言如 Pascal 或 COBOL 的用户会发现，本书对他们也很有帮助．事实上，假如你使用的是其它的语言，那么本书的大部分内容对你来说就更简单了，尤其是对于用惯 Pascal 语言的用户来说更是如此．因为 Pascal 语言的设计直接包含了本书所表达的许多概念．为了同样的目的，本书所讨论的专题也大都独立于具体的机器和语言．

目　　录

第一章 系统分析

1.1 引 言

大多数缺乏经验的程序员总是一有点关于程序的想法，就马上着手书写代码。固然，BASIC 语言的设计者想创造出一种能在编程过程中尽快得到答复的编程语言。随着微型计算机和**交互式**程序设计的问世，程序员可以立刻从程序得到反馈，因此那些缺乏经验的程序员急于写代码的愿望就愈加强烈。

但是这样就会引出一种令人伤脑筋的、错误的编程方法，以致大大增加了工作的难度。近来，业余程序员，即那些主要工作不是编制程序的人，觉察到专业程序员已认识多年的事情：为保证编程的成功，要求程序员遵守纪律、有条不紊和相互密切配合。

应用程序不是单独产生的。是用户的要求推动了应用程序的开发。典型的情况是在一家大公司用户与计算机工作人员讨论他的需求。经过这种讨论，计算机工作人员就可以定义和开发一项新的应用。下一步则由程序员去实现这项应用。最后，为用户提供使用这项应用的操作步骤。

什么是系统

在计算机领域里，术语“系统”有几种定义。第一，它是指硬件本身。系统各部分之间的关系同物理连接一样简单。在此种意义下，计算机及其外围设备就代表着一个系统。这些单个的部件组合起来执行一种总体功能。

系统也常用来代表软件，作为应用程序的同义语。在这种意义下，它专指很复杂的程序。

第三种意思指的还是软件，它与系统程序有关。在这种意义下，共同完成某种任务的各个程序组成一个系统，尽管用户不能轻易地区分系统中的各个程序。

最后，系统一词可以指共同执行一种专门功能的硬件与软件的组合，通常说的**通灵系统***就是这样一种系统。其中计算机是为某种特定应用而设置的，只要一通电它就立即执行——运行特定的应用软件包。

对系统一词最一般的定义可以是“一系列相关联的、起某种作用、执行某种功能或操作的单元”[96]。按照这种说法，它与计算机技术并无直接的联系。的确，每个公司都有一些靠人工而不是靠计算机和电力运转的系统。一个完整的系统甚至还包括既由人又由计算机执行的一些操作。

什么是系统分析员

系统分析是研究系统不同部分间的相互关系和相互作用的。考虑到系统一词最一般

---

* 通灵系统又称“交钥匙系统”，是指该系统的完备状态，不用增添或改装就能使用的系统或设施。——译者注

的定义，**系统分析员**不需要在计算机上干任何工作，因为在这种情况下完全是由人对系统进行研究。

为什么系统分析员要研究系统呢？研究的最终结果又是什么呢？

最初，这类工作是由那些观察公司或厂家运行环境的效率检验人员承担的。分析的目的是为了提高效率。然而，从本质上说这还不是最终目的。人们期待的是通过提高运行效率，使生产率得到相应的提高。生产率的提高会极大地减少运行费用，或者以很少的雇员完成同等的工作，或者以同样多的雇员生产更多的产品，其最终目的乃是更高的利润。

今天，系统分析员这一术语主要地同计算机领域联系在一起。也许分析员研究的基本上是人工系统，而这种研究的目的却是决定该系统是否能够或者是否应该计算机化。其主要目的仍然是提高生产率，即提高效益。这种计算机化了的系统是速度更快、价格更便宜并出错少的系统。这有时尽管会产生争议，而任何一个可以用计算机来管理系统最后通常都被自动化了。

分析员的任务：第一部分

分析员肩负着两项主要任务。首先，他的职责是如何更详细、更准确地确定现行系统的功能。这可不是一件一蹴而就的任务。它牵涉到同一工作中由各种工作人员所从事的许多过程，这在任何资料中都是查不到的。大多数公司在怎样完成任务方面，“口授传统”仍起着重要作用。

另一个问题是，这些过程可能不会很好地为采用它们的人所了解，他们可能不了解为什么要按照他们采用的那种方法去做这些事，而仅仅知道“一直是这样在干”。但是，分析员必须知道运行中的每一步是为什么，以便确定其有关的特征，从而知道这些步骤是过时的或无效的。如果你不知道完成的机理而要了解某一步是否无效就有困难。

再者，即使有某些运行文档，但要知道在运行中所有步骤是什么，可能也相当困难。如果特定的工作人员这天不在场，即使分析员拜访了每一个人，重要的细节可能还会被漏掉。因为所研究的整个系统必要的那部分未被当作运行部分考虑，它就不可能为分析员所研讨。

所有这些常常使得确定现有系统是如何工作的这一问题成为分析员工作中最困难的部分。从最靠不住的场所搜集重要信息，分析员应该更像新闻记者。只有对现有系统的各个环节都了如指掌，并加以详细描述之后，才可以着手系统的计算机化。

分析员搜集有关信息的方法主要有三种。第一，必须仔细研究现有系统的各种文档，包括任何描述过程的手册，说明某些机制怎样工作的图表，例如用于每一步中的所有表格（不管多么稀罕），以及为系统自身准备的方框图。

第二，分析员必须同每一个在现有运行系统中负有重要任务的雇员交谈。然而，要精确地指出什么是“重要的”常常是困难的。可靠的方法是与每个有关人员进行谈话，但这样可能要花费很多时间。在庞大而又复杂的系统中，或者涉及许多操作人员的地方，常常要有一班子分析员。

进行采访时，要在最少的时间内获取最多的信息不是一件容易的事情。怎样才能按问题本身的重要程度来提出问题，也应加以考虑。因为与分析员打交道的都是人，这些人并不一定都愿意竭诚合作。因此，分析员必须具备一些人际关系方面的知识。这是最困

难的(关于不同采访方法的进一步的探讨,读者可参阅有关文献).

分析员用来获取信息的最后一种方法是观察. 分析员对正在进行中的各种过程的简单观察,往往比用其它的手段更容易确定正在干的事情,尽管分析员使用本法很难说不会漏掉一些细节,但如果别无它法,则直接观察对于核实用其它方法所获得的信息仍不失为一种有用的工具.

分析员收集有关系统的信息,尽力回答下述五个简短而深刻的问题:

1. 系统目前正在干什么?
2. 为什么要这样做?
3. 谁在执行这个系统?
4. 怎样去运行这个系统?
5. 在现行的方法中涉及哪些问题?

只要这些问题得到了详细解答,则现有系统的说明就出来了.

分析员的任务: 第二部分

分析员工作的第二部分是,为现有系统描述一个用计算机处理的系统,用它去代替现有系统. 由于这部分更直接地涉及到计算机应用,因而也是最常与系统分析员发生联系的部分.

这一阶段分析员有几项任务. 第一,分析员必须准备一份新的计算机应用说明,它是以刚刚分析过的现有系统的说明为基础的. 当说明新的计算机应用将怎样解答与现有系统的有关问题时,这项任务就开始了. 此外,新的应用必须采用新方法,直至极大地改变系统工序. 分析员必须认识到这都是改进系统的机会.

分析员还要按某种顺序将系统描述成一系列应完成的子任务. 原有系统中或许在一定程度上已经这样做了(因为涉及到多方工作人员,每人都有其自身所从事的作业). 现在分析员必须定义计算机能接受的逻辑子任务,或者重新组合那些原有的子任务.

分析员的下一项任务是考虑工作的自动化. 这就产生了两个主要问题. 第一,既然数据被保存和管理的方法能对一个系统的效率产生很大影响,同时,因为任何系统的主要目的又是以不同形式保存和产生数据,因此分析员就必须知道系统最有效和最合适的数据组成方式.

第二,分析员应事先考虑系统的计算机化会引起什么问题. 在系统中某些过程可能与时间有关. 例如,当一个用计算机处理的应用系统在执行一个过程的某些步骤过快时,往往会出问题;由于某个职员为填写账册要花费一定时间而导致的延迟等. 因此保持合适的盘存是必不可少的. 借助计算机化使处理过程加快,就可能使盘存大大减少.

分析员的最后一项任务是,在计算机系统与最终用户之间起通信联络的作用. 作为联络人,分析员要帮助用户掌握新的应用. 此外,分析员还要作为用户的代表与计算机部门打交道. 在阐述新的应用系统时,他(她)要帮助用户阐明他们的需求并保护用户的利益. 因为计算机工作人员要用一定的方法去实现一种应用,但却不能勉强用户去接受用起来不方便的处理过程.

从试凑到应用

有些人常常会提出一些在人工系统中从未有过的计算机应用的想法．例如，设想一个新的计算机游戏． 新应用的最初说明一般地以现在被使用的某一系统的分析为基础，那么怎样着手说明这样的应用呢?

既然没有现成的系统，那么以上所叙述的初始分析显然是不可能的．假若能分析的话，那么就会得到你最终需要的同类系统．要做到这一点，就要设法去回答那些假如系统实际上已经存在，而你将回答的同样问题．这需要丰富的想象力和一定的试凑过程，直到你对得出的说明满意为止．

系统的生命期

不论项目是一个简单程序还是一个复杂、庞大又互相关联的程序系统，**系统的生命期**都是从理论到最终产品均获得某种计算机应用的一系列步骤，遗憾的是，还没有定义程序设计生命期的可接受的标准．这同没有公司管理的标准方法是相同的．生命期是一种程序设计管理的方法．因为各个项目的需求可能极不相同，所以生命期的细节也可能不同．

由于包含可行性研究、正式提出与详细复审，因此多数生命期都包括相当复杂的计划．当涉及大公司或主要工程时，上述步骤是不可少的．业余程序员一般不需要这样做，因而在所定义的生命期中可省去这些额外的步骤．

定义阶段

所有这类应用项目都是从**定义阶段**开始的．像其它要讨论的那些阶段一样，此阶段可以按照你的愿望弄得非常精致或者相当简略．就个人而言，我总是在两者之间取折衷态度．

在此阶段，涉及到定义的是什么应用．上面已经讨论过，它包括系统的一般分析，此阶段还包括**初步设计**．

初步设计的目的是为系统设计提供一个总提纲．系统将怎样实现其功能，最好是留待以后详细地说明．定义阶段的初步设计部分和**设计阶段**有点儿重叠．这是因为分析系统所用的多种方法能提供很多的初步设计．因为术语“初步设计”已非正式地用于计算机领域，所以在大部分文献中都包括这些方法．

设计阶段

系统的**详细设计**是程序的逻辑与各功能部分更加具体的设计描述．此阶段首先是表述程序将怎样实现它的功能，因而仍然要避免涉及具体的机器或编程语言．当然这只是提纲式的一般说明．这样就允许所安排的程序逻辑不受限于具体的机器或者编程语言，使得能用多种不同的计算机或用不同的语言开发程序，而不必每次都从头开始．即使不是某个项目的具体要求，在任何程序的生命期里，设计阶段也是不可缺少的．

设计阶段很像建筑师的蓝图设计．初步设计相当于建筑师所作的草图和模型．这些设计图绘制成某种最终形式，就相当于程序的一般说明．下一步就要根据建筑师的图纸制出蓝图．

按照这些蓝图才能建起楼房．没有这样的蓝图，任何人也无法建成一座大厦．建筑师的工作是遵守正常的建筑实践．与此类似，系统分析员要准备详细的程序设计，遵守正常的编程实践．

实现阶段

在程序全部设计完后，就进入**实现阶段**．至此，就要牵涉到曾经小心翼翼地避开过的有关计算机和语言的细节．在此阶段，设计转变为程序．

如果设计得好，此阶段是较顺利的．在多数大公司里，就在此刻由分析员把项目转给程序员．在详细设计之前，一些公司在初始阶段就这样做了．确切地说，采用哪种做法是由各公司决定的，而且往往取决于公司所采用的生命期的合理性．

检验阶段

在程序编完码以后，评定一下它的运行情况是必要的．**检验阶段**通常是最困难又费时的一步．在此阶段应该发现并改正错误．搞计算机的人一般都知道，有些毛病在这时往往还很难找到．

尽管通常认为此阶段是紧接着实现阶段的，但实际上它应该出现在设计和实现两个阶段之后．一般说来做此设计的目的是为了对它作出评价．这一评价包括为发现错误所作的某些初步测试，也包括该系统的未来用户对它们的检验，看它是否能满足用户的需要．这是件很复杂的工作，但通过这样的检验能极大地提高产品的质量，使用户满意．

文档编制阶段

在系统完全实现后，到说明怎样使用系统的详细文件建立之前，都要抓紧时间编写文档．**文档编制阶段**类似于检验阶段，因为在生命期的不同时刻，都要作编制文档的准备，而不是弄到最后才作这种准备．

由于下述原因，程序员通常都害怕过这一关．第一，在程序员中间一般存在怕写文件的念头，即怎样按照程序开发的真实情况去编写它．第二，编写文档明显不如写程序有趣．第三，最主要的原因恐怕还是公司管理部门通常不重视编写文档．最大的危险是程序员的上司交给他新的工作，使原来的项目仓促通过检验阶段，只留给程序员极少的时间去编制必需的文档，结果就使多数系统很少有充足的文档说明．

产品阶段与维护阶段

当程序最终为用户的实际工作采用时，它就进入**产品阶段**了．对于工作正常的产品系统，此时程序员不需要再做什么；对于较大的系统，有时候程序员还要负责建立专门的执行程序，如在预定周期的结束或在年终．

维护阶段与产品阶段是重叠的．程序员在这一时期的实际任务是监督程序的使用．必要时要将系统做些改动(即维护)．这样做有两个主要原因．第一，用户的需要可能变了，这意味着在程序中要添加、删除或修改其功能以适应用户的新要求．第二，总会发现错误(通常是由用户发现的)，必须查出这些错误并加以改正．然后系统一定要重新测试．当这项工作做完之后，系统的新版本就可代替现在的版本了．

1.2 应用的说明

从几个方面来说，建立应用的一般说明是整个过程中最容易的一步，也是最重要的步骤之一．如果原始说明准备不足或有错误，此缺陷将很快变成隐患．在以后的各步中，如果它没有被发现的话，要改正过来就很困难．因此，为了得到正确的一般说明，大多数专业人员在建立应用说明时可能要花费大量的时间．

一般，说明是通过反复地修正过程来完成的．在这一过程中，分析员将向申请应用的用户提供有关系统的说明．虽然用计算机处理的系统的多数技术说明超出了用户所需了解的范围，但是这样的简要说明常常有助于澄清用户的误解．此外，通过这些磋商，使用户逐渐熟悉用计算机系统的新方式去处理原有工作．

用户可能不喜欢或不了解分析员所表达的每件事情，这就经常要求分析员反复说明系统的各个部分．在这样的重新说明之后，要给分析员与用户提供另一次会晤的机会，以便对系统说明作出评估．这一循环持续到分析员与用户都对该说明满意为止．然而，要达到这一点双方必须有妥善解决问题的愿望，以便使方案进入开发阶段．因为多数专门方案都受预算和时间两方面的约束，因此要兼顾这两个方面．

本节将对应用的四个主要部分的正式说明予以探讨．第一，任何现在使用的系统分析结果必须重新检查．第二，讨论应用的一般说明．当该说明最初可能是以现有系统的分析作为基础时，那它就不是必要的一步，此说明也可能是大大修改了的现系统的结构．如前面提到的，也可能不存在这个现在使用的系统，或现有系统太简单，不能为新的应用系统提供坚实基础．在这种情况下，只好依靠一种设想的系统作出说明．第三，很重要的一点是描述希望输入给系统的是什么．借助于描述这些输入，我们就有了系统的起点．此外，我们还可决定该应用系统将如何同其它程序或系统连接．例如，这一应用系统的输入可能就是来自另外某个系统的输出．最后，我们应该说明系统要产生的一组输出，它确定了应用的最终目的．这样当我们继续开发该应用系统时，在我们面前就会有一个明确的目标．

要注意的是，我们没有以任何方式指明输入将怎样被转变为输出．若现在就加以详细说明，只会把注意力从为获得一个全面说明此应用系统应该做的事情上引开．

现有系统的分析

在本章的引言中，我们知道了系统分析员的任务．当为一个现有系统的计算机化做准备时，分析员必须准确地看出现有系统是怎样运行的．此外，他还必须找出此系统的弱点．最后，在提出一个计算化的系统时，还必须提出克服这些弱点的方法．

这些弱点通常是比较清楚的，它们就是在第一步分析中曾提出过的问题．系统中两个最明显的缺陷是它们太费时，难以履行其功能，或者它们花费太大．系统分析方面的困难在于，表现为系统毛病的常常只是些难以捉摸的迹象．例如上面提到的两个问题就可能是出错比率高的缘故．又如某生产系统，由于生产了大量低劣产品，原来认为是管理费过高的问题，其实是质量控制问题．

一个例子

人们最好从实践中学习．因此先看个例子，以便更好地了解家庭存折结算问题．我们将要看到，存折本身只是整个系统的一小部分，因而我们所考虑的是使银行支付报告结算更加简便的一种应用．这个例子在本章其它部分也将用到．

首先，即使此项应用是为自己使用而开发的，你也要按照是为别人开发的那样做下去．对于分析的对象你也仿佛一无所知，这样会有助于以后的探讨．同时应该记下每个过程的细节，不管它们看起来多么琐碎．尽可能少作假设，并且要避免对自己说，"这都是很明显的事，根本就用不着记下来．"你很快就会改变这些看法，即使在最小的系统中，全部细节都需要关注．因此别错过机会，把它们全都记下来．

对这种应用，从向自己提出关于现有关系是怎样工作的问题开始，会使你获得一系列帮助定义系统的方法．对于存折系统，可以从下述步骤开始：

1. 开一张支票并把它送给你的债权人．
2. 债权人收到这张支票，并把它存到他自己的银行账户上．
3. 贷方银行与你方银行联系安排现金的转账，该款项将增加到你的债权人的账户上．
4. 你方银行从你的账户上扣去支票上的数额并把它转给贷方银行．
5. 每月末，银行给你一张全部支票兑现后的结算表，它也包括你的一份列出了有关账目的各种细节的结算表，其中计入了银行认为是你的本月的余款．

此时，你被认为是接受了银行提供的结算表和存折"结算"．该结算实际上是确定你或银行在你的账户上记账时是否产生了错误的过程．由于银行使用了高度完备的计算机系统，通常认为由存户造成的任何差错都不会造成损失．

在检查用于结算每月结算表的过程之前，先让我们来分析一下用户可能造成了哪些类型的错误．最明显的错误是简单的算术类错误．如果出现了存折中该减的值反被加到了本月的余额上，就出现了基本的简单加减法错误．还有，当错误值进入支票日报时，会产生更加难找的错误．尽管计算也许是正确的，但由于被加减的值不正确其结果仍然是错的．最后，当用户忘记把一份支票记入日报时，又因疏忽而出现错误．我们希望这一新的计算机化的账簿系统有助于克服现有系统的这些不足．至于怎样达到此目的还不明朗，眼下还不需要对此进行详细描述．

其次，让我们看一下在支票账目中需要处理的各种表格．三种主要表格是支票本身、支票登记簿及银行的每月结算表．

支票只要已经兑现，它对我们就没有任何用处．某些银行甚至停止收回每月盖销的支票，而喜欢把每张支票的副本保存在银行的微型胶片上．万一某家银行或公司对正常支付额发生争执，盖销支票仍然是解决问题的最好证据．

一张标准支票上的信息告诉人们它是写给谁的、它的金额、发放日期和填写人的签名．此外，多数支票还有一处空白可用来说明其用途．然而为了使支票合法而要求别的资料都齐全时，该空白也是任意使用的．可惜，正是因为任意使用，即使它以后可能提供有用的资料（例如在纳税时），多数人也不愿用它．

另一条能够从盖销的支票上得到的信息是兑现的日期，通常它是盖在支票的正面或反面的．事实上，此日戳是支票确实已经兑现的唯一证明，尽管它并不是一条不可缺少的

信息，但仍然有它的使用价值。

最后，每张支票有其唯一的支票号．这是必不可少的，它是银行区分支票的唯一方法。

支票登记簿是账目持有者用来逐日记录账目的簿记．每填一张支票，填写人都应该把确定的数据写入支票登记簿．这些数据一般包括支票号、填写日期、写给谁的以及金额．支票持有者应从原先的结算中减去支票金额以得出现在的新结算．这个新结算并不表明填写支票的时刻在银行账目上有多少钱．这是因为在支票登记簿中先前记入的和已减去的支票实际上可能没有兑现．如果不出现"透支"账目，它就是保持能动用的账目储备的标志。

除了支票数据外，任何入账的存款都记入登记簿．该存款再加到现在的结算里．按常规，对于存款需要保存的另一条信息是存款日期。

每月的结算表在银行之间可能有很大差别．结算表一般包括从上月结算表中已兑现的全部支票明细表．这份表通常是按日期而不是按支票号排列．此外，还要给出支票当天的结算．在结算表周期内，这实际上是银行在账目中清付的金额．可是它很难与任何记入支票登记簿的金额一致．这是因为常常有许多"未偿付"的账单，也就是说有一些支票还没有兑现。

为了结算存折，一般需要做以下几件事：

1. 把所有未偿付支票的值都加起来。

2. 把结算表截止日期以后所存入的全部款额加起来，就是把本月或以前的结算表上尚未出现的那些存款加起来。

3. 取在结算表上给出的最终日报余额减去未偿付支票的总和，再加上未付存款的总额，其结果应等于记入你支票登记簿中的最后余额。

此过程是相当简明的，仅需做些简单的算术运算．然而在此范围内可能有极少数已结算的账单．如果一份账单未结算是为什么？又怎样知道？

在上述三步中，我们说明了结果应等于记入支票登记簿中的最后余额．如果是这样，则存折就结算完了，即银行认为你有如账目上所记的那么多钱．但如果数值不相同，那就有了问题．是什么样的问题呢？

如早先所指出的，有一些可能产生错误的原因．更糟的是，可能已经产生了多处错误，这就会使查找问题更加复杂。

因为全部的做法是要说明是否在存折中(即在账目持有者的支票登记簿中)已经发生了错误，所以尽量列举实际上有可能发生的错误是很重要的。

在这种系统中可能有三种错误源．第一，在支票登记簿中的值可能是错的．第二，对于结算存折的过程可能处理不正确．第三就是银行很可能在每月的结算表上出了错，虽然其可能性相当小，但也是存在的。

前面我们已粗略地叙述了在支票记录中产生错误的各种可能性．在结算过程中可能还会产生下述错误：

1. 可能将未付支票或存款的值做了不正确的复制。

2. 把未偿付的支票当成已付的，而实际却未付。

3. 漏掉了一张未付支票，因而没有加到未付支票的总数中去。

4. 把未偿付存款当成已付的，而实际上却未付。

5. 一份实际上是未付的存款被漏掉，因而没有加到未付存款总数中。

6. 对银行每天最后的余额复制错误。

7. 未付支票总计错误。

8. 未付存款总计错误。

9. 最终的计算执行出错。

以上相当全面地概括了存折结算的过程。它几乎包括了该项运算过程的全部基本细节，甚至描述了系统中可能出现错误事件的地方。这样就能产生以计算机化的应用为目的的说明。

在分析中，下一步是确保每一被使用的术语都容易定义且完全明了。多数条目，诸如"支票金额"和"最终日余额"都是不解自明或者很容易根据使用的形式辨别的。

一张未付支票的定义是什么？早先我们使用了一个与是否已兑现有关的不严密的定义。然而，它的先决条件是我们要以某种方法保持对已兑现支票的跟踪。因为无法确定支票兑现的顺序，即支票兑现的先后问题，为此就需要专门的方法。有些支票开出的当天就兑现，而另一些支票半年或更长时间可能也不兑现。任何未兑现的支票，不论是上月开的还是一直没兑现的，都必须作为未偿付的支票处理。

支票登记簿一般有一个备注区用来跟踪未偿付支票。当支票兑现时，把银行转回的支票当作凭证，在备注区加盖标记。任何无这种标记的支票都是目前尚未偿付的，未付存款也以类似方法管理。

把可能产生的各种错误列表如下：

1. 不应盖上兑现标记的支票，实际上盖了。

2. 应该盖上兑现标记的支票却没有盖。

3. 盖了兑现标记的支票错误地加到未偿付支票的明细中。

4. 在未偿付支票中漏掉了无标记的支票。

也可以在存款中列出类似的错误清单。

应用说明

以上面的分析为基础的即将开发的具体应用系统完全取决于用户的要求。在这个实例中，能被描述的合理应用具有很宽的范围，它包括很普通的到相当复杂的应用。

下面先说一下普通的应用。

在结算过程中产生的大部分差错是简单的算术错误，它是在总计未偿付支票与存款和最后的结算计算期间产生的。用户可能需要帮助消除这些错误的程序。此程序描述如下。

存折程序 1. 本程序将把由用户提供的未偿付支票表和未偿付存款表加起来，并将协助用户结算他或她的存折。在给出程序之后，最终结算由用户列入每月银行支付单，程序要输出一个同在用户支票登记簿中的最终结算相符的数。如数值相等，则支票登记簿是正确的，账目也是平衡的。如果两者不等，则支票登记簿可能有错，用户必须查出并改正错误，然后再次运行程序结算存折。

也许以上的说明太具体了，这样紧接着描述关于程序该怎作工作的细节就变得不方