

- 1.说明计算机硬件、软件和使用者的故障的差异。
- 2.如果你的手机出现了问题，手机生产商仅仅给你重装软件，而不是给你换一台手机。这是哪种类型的故障？如果是汽车上的刹车系统，也这样做，你觉得放心吗？
- 3.软件使用的次数和时间越长，越可靠，对吗？为什么？



## 第二章 作业情况

- 总体情况：本科生66名，收到作业63份。成绩为：A+有4个，A为15，B为31，C为11，D为2。
- 主要问题：
  - 1) 没有写明硬件、软件和使用者的故障的差异；
  - 2) 没有写全硬件、软件和使用者的故障；
  - 3) 刹车系统没有考虑是软件、硬件的组合；
  - 4) 第三题错误较多，理由写的不够完整，部分同学认为那句话是对的。

## 第二章 作业示例

1. 硬件故障：硬件、软件、使用者出现故障的总称。

① 硬件故障：a) 由占据空间的物理体导致的故障为硬件故障

b) 通常是因为自然界的物理作用受到风吹、日晒、湿气、空气污染度等外来因素的影响，出现的老化或磨损。

c) 物理体的这种老化和磨损是一个连续渐变的过程。

d) 硬件在其生命周期内发生故障的规律基本符合“磨合期→稳定时期→老化期”。

② 软件故障：a) 由计算机软件引起的计算机称为软件故障。

b) 通常是因为：i) 计算机除零错误；ii) 指向内存地址的指针超出规定的内存范围；iii) 整型数据上、下溢出；iv) 浮点数据上、下溢出。

c) 其他类型的错误包括：i) 数据计算的精度不够（如计算误差太大）。

ii) 输出或显示的数据位数不够。

③ 使用后的错误：a) 由软件开发和维护团队的管理和工作的有关错误。

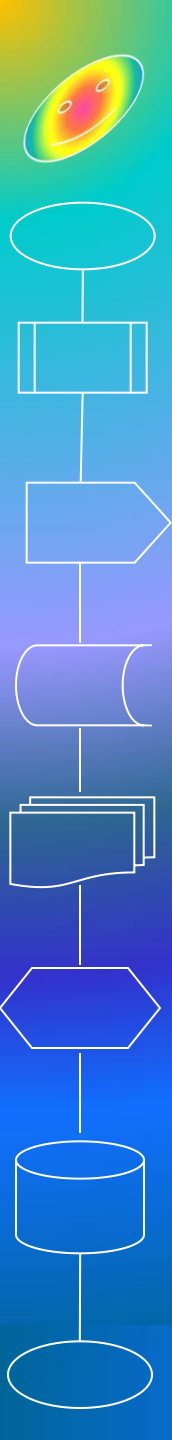
b) 包括配置错误、规程错误和编码错误等。

## 第二章 作业示例

2. 手机生产商重装软件而非换一台手机：属于软件故障，与手机本身的物理硬件无关，属于软件故障。如果汽车上的刹车系统也这样做，会不放心。因为汽车刹车系统属于硬件系统，硬件系统的设计遵循全生命周期设计，即整个系统的使用寿命取决于寿命最短的关键部件（如发动机、车架），而不是取决于可更换的个别零部件的寿命（如轮胎、电瓶）。故当汽车的刹车系统出现故障时，属于汽车硬件系统的关键部件故障，意味着刹车系统，以至于整个硬件系统的关键部件，均逐渐由“平稳工作期”进入“老练期”，若仅更换刹车系统而不进行车辆的硬件系统全面检查，将存在更大的安全隐患。

## 第二章 作业示例

3. 使用次数越多、时间越长, 用户使用越多的软件更比首次使用的软件更可信一些。  
长时间、大批量用户的使用促进了对软件更进一步的测试。根据软件故障修改  
测试后的代码可以降低软件故障率。  
因此, 可以建立软件库, 提高代码的复用率, 可以提高软件系统建设效率, 并最大  
限度地降低编写新代码所带来的缺陷问题。  
但是, 实际的软件故障并不是仅随着使用期间的增长而降低。那是理想化的故障率曲线。  
实际中的软件故障率会在每一次软件修改后有一次较为显著的提升, 而后会在一定的时  
间范围内遵循“使用期间增长, 故障率降低”的规律, 直到下一次修改测试。



王老师：

理论上，使用时间越长，越可靠。但是，如果关键路径（判断组合等）没走过，软件可靠性，并没提高。这就是软件和硬件的差别.....逻辑离散系统，不能用使用时间长短，说明可靠性

另，使用中发现错误，会导致修改，修改可能带来更多错误，见软件故障曲线



# 第三章 作业题目

针对下面各系统，提出合理的软件开发过程模型，并给出理由：

- 1) 汽车刹车的控制系统；
- 2) 学院的一个学生管理系统；
- 3) 一个航班订票系统；
- 4) 一个程序设计课的项目，要求在两个小时内完成；
- 5) 开发一个类似于Excel的制表软件，并打算到市场上销售。





# 第三章 作业情况

- 总体情况：本科生66名，收到作业63份。成绩为：A+有28个，A为7，B为18，C为8，D为2.
- 主要问题：
  - 1) 提出的软件开发过程模型与项目不匹配；
  - 2) 提出软件开发过程模型的原因阐述的过于简单、不准确；
  - 3) 问题较多的是第四个系统。



# 第三章 作业示例

## 11) 高速火车的刹车控制系统

注:每次隔一页,前是chap 03  
后是chap 02

模型:瀑布模型

理由:对于高速火车的刹车控制系统来说,它的需求比较明确,具有严格的技术、行业规范  
这与瀑布模型明确了各阶段活动、工作和责任,输入、输出,以及规定了各个阶段工作的质量、时间、人力资源和其他资源是相符合的

## 12) 本学院的一个学生管理系统

模型:增量模型

理由:对于学生管理系统来说,可以先实现那些需求明确的增量,随着系统开发的进展,人们会对一些需求不明确的需求逐渐清晰起来,那么在第二次,第三次等迭代的过程中,可以更容易地实现这些需求的功能。

# 第三章 作业示例

(3) 一个航班订票系统:

模型: 渐进式开发

理由: 对于航班订票系统来说, 其需求不能完全地区分出所谓的“需求明确部分”和“需求不明确部分”(其中受一部分市场的影响), 这就需要渐进式的多次迭代过程.

(4) 程序设计课程

模型: Build and Fix 模型

理由: 对于程序设计课程来说, 其需要编程人员在非常短的时间内交付给用户

这与 Build and Fix 选编第一个版本(较短时间内), 再多次修改直到用户满意, 最后投入使用的过程相符

# 第三章 作业示例

⑤ 开发类似于 Excel 的制表软件

模型：螺旋式开发

理由：螺旋式开发与渐进式开发类似，但会强调商业上的风险。特别是那么时间周期长（多达数年）、人力资源耗费多的、面向市场的软件项目，期望每次迭代都要对商业风险进行分析，以避免系统最终实现后，已经没有市场需求。

# 第三章 作业情况

项目需求情况	瀑布	增量	渐进	螺旋	形式化	稳定同步	Build-Fix	最佳
1) 汽车刹车的控制系统;	可行	可行	可行	N	可行	N	N	瀑布+形式化
2) 学院的一个学生管理系统;	N	可行	可行		N	N	N	多次迭代
3) 一个航班订票系统;	N	可行	可行	可行	N	N	N	多次迭代 (以及市场风险)
4) 一个程序设计课的项目, 要求在两个小时内完成;	N	N	N	N	N	N	可行	Build-Fix (时间)
5) 开发一个类似于Excel的制表软件, 并打算到市场上销售。	N	可行	可行	可行	N	可行	N	稳定同步 (同时观察市场风险)



## 第四章 作业情况

- 总体情况：本科生66，收到作业62份。成绩为：A+有13个，A为20，B为18，C为11。
- 主要问题：
  - 1) 没有站在不同的角度去分析，过于局限；
  - 2) 分析内容过于简短，没有结合两个系统的特点进行分析；
  - 3) 分析内容不准确；
  - 4) 没有结合第五章分析对二者密安性的单独评价；



# 第四章 作业情况

第三次作业 (ch04-ch05)

雷媛嘉  
2020211005

► 建立一个质量评价模型, 比较Linux和Windows的质量.

# step01: 建立质量模型.

质量模型的建立可以分为 2 个部分: 用户观点和 产品观点.

1. 用户观点: ① 在用户观点中, 使用者最为关注的是系统的使用和运行是否会给其带来效益.

在这种观点中, 质量便为“用户眼里所看到的”, 因为质量是用户对软件系统的最基本需求.

② 用户的质量需求包括在指定的使用环境 (指使用软件产品和系统的用户、任务、设备及物理和社会环境) 对使用质量的需求. 因而评价指标的设置也将围绕用户的使用需求.

2. 产品观点: ① 从产品质量观点出发, 产品质量模型将产品质量分解为许多质量因素 (特征、准则或特性), 然后再分解为可以测量的度量元或指示器.

② 在这种观点下, 产品的质量是一个可精确和可测量的变量, 而这些属性的量化差异也将反映着质量的差异. 因而评价指标的设置也将围绕着产品本身的那些可以被客观评价的属性.

# 第四章 作业情况

#step02: 给出质量指标.

依据上一步的结论可得, 质量指标必然分为"用户主观评价的"和"产品可被定量评价的"属性.

1. 用户主观评价的指标, 即人的主观因素对产品的定性评价.

依据 ISO9126 对产品质量属性的分解, 可得定性的质量指标如下:

- ① 功能性 中的 (互操作性, 兼容性).
- ② 易用性 中的 (易理解性, 易学性, 吸引力).
- ③ 维护性 中的 (易分析性, 易更改性, 易测试性).
- ④ 可移植性 中的 (易安装性, 共存性, 易替换性).

2. 产品可被定量评价的属性,

依据 ISO9126 对产品质量属性的分解, 可得定量的质量指标如下:

- ① 功能性 中的 (正确性, 正确性).
- ② 可靠性 中的 (成熟性, 容错性, 易恢复性).
- ③ 效率 中的 (时间特性, 资源特性).
- ④ 维护性 中的 (稳定性).



# 第四章 作业情况

#Step03: 对 Linux 与 Windows 进行比较

## 1. 功能性:

① 适用性: Linux 和 windows 在各自的内部使用用户群中, 均可以为指定的任务和用户需求目标提供一组合适的功能. 如 windows 在大众化办公 OS 使用群体中可以做利伏什的界面设计, 而 linux 更面向开发人员, 操作更便捷.

② 准确性: 二者均可实现高精度定时器的内核节拍, 精度均小于 1ms, 具有所需精度的正确或相符的结果或效果的能力.

③ 互操作性: 二者均可实现与一个或更多的规定系统进行交互的能力, 如硬件设备、网络、文件系统、数据库系统等. 且二者之间的互操作也通过各类虚拟化技术<sup>实现</sup>.

## 2. 可靠性:

① 成熟性: 二者均投入使用的时间大, 使用用户多的操作系统, 均具备良好的成熟性.

② 容错性: 二者均通过多级 RAID 等技术实现了较高的容错性.

③ 易恢复性: 二者的易恢复性可通过故障发生到恢复的这段时间长度来评价.

# 第四章 作业情况

## 3. 易用性

① 易理解性: 二者均配有使用手册或指南, 然而 windows 的图形化界面与操作成  
会更方便于用户理解并操作。

② 易学性: 学习使用命令行终端形式的 linux 操作使得其易学性低于图形化操作的 windows。

③ 易操作性: 使用命令行终端进行操作的 linux 系统的易操作性低于图形化操作的 windows。

④ 吸引力: 在各自面向(或偏向)的用户群体中, 二者均具备较强的吸引力。

## 4. 效率

① 时间特性: 一般在开放性、多用户、多任务这三个方面 linux 效率优于 windows, 但在网络层面上,  
windows 的 TCP 模型<sup>10CP</sup>优于 linux 的 epoll 模型, 可用平均响应时间等指标来

② 资源利用性: 在显卡方面, 尤其是双显卡方面, linux 目前并无可靠的双显卡交互手段, 占极大劣势。  
导致与 windows 相比, 显卡驱动不足。

# 第四章 作业情况

## 5. 维护性

- ① 易分析性: linux内核的模块化程度高, 因此比更易诊断出软件的失效原因.
- ② 易修改性: linux的开源代码使其透明、更易修改.  
(GLP)
- ③ 稳定性: 二者稳定性相似. 随着二者的不断更新, 当前版本的稳定性均有提升.
- ④ 易测试性: linux的开源代码使其编程、测试更相对windows更容易.

## 6. 可移植性

- ① 易安装性: 总体而言windows更易安装; 但目前安装linux的方式(如ubuntu)已基本实现与windows安装模式难度相似的方法.
- ② 共存性: linux更适合嵌入式开发; 但二者均有较好的共存性.
- ③ 易替换性: windows更易卸载.



# 第四章 作业情况

## #Step 04: linux 和 windows 的密性:

1. 密性指保密性、可用性和完整性的结合.

包括防止非授权用户的存取或访问系统中的信息,以及防止意外事故(如物理损坏)

等引起的不希望的信息泄露:

① 保密性指: 不会发生非法的信息泄露.

② 可用性指: 随时可用的特征.

③ 完整性指: 不会发生不正当的修改信息的情况.

2. Linux 的密性更好:

① Linux 具有开源性. 若发生重大漏洞, 比起 windows 系统中只有内部工程师才可以解决的情况, linux 可以被更多人参与解决.

② Linux 的设计框架采用权限管理的方式, 提高完整性.

③ Linux 中有大量的网络管理等功能, 可以使用户更高效稳定的建立防火墙、路由器、服务器等. 同时, 大量的网络分析与网络安全软件也可提高保密性.

A<sup>+</sup>

# 第四章 作业情况

北京邮电大学数学作业纸

管理

班级: 202021131

姓名: 赵冉

编号: 202021131

第 1 页

## 1) 质量模型:

从产品的角度出发, 产品质量模型将产品质量分解为许多质量因素(特征、准则或特性), 最低层的是可以测量的度量元(或指示器)。

从用户的角度, 使用者最为关注的是系统的使用和运行是否会给其带来效益。质量是用户对软件系统的最基本要求。用户的质量要求包括在指定的使用周境对使用质量的需求:

建立的评价模型更偏向于产品质量模型,

为了清楚地定义、测量和改进质量, 因此要测量影响软件的每个特性。据此, 给出如下

质量指标:

## (2) 指标: 可以定量评价的指标:

0 可靠性 (在指定条件下使用时, 软件产品维持规定性能级别能力, 一般借由 MTBF 表达系统的可靠性)

0 效率 (时间特性、资源利用性)

## 定性评价的指标:

0 功能性 (转软件产品满足明确和隐含要求的功能的能力)

- 适合性
- 准确性
- 互操作性
- 安全性

0 易用性 (从用户使用观点来看, 在指定条件下使用时, 软件产品被理解、学习、使用和

吸引用户的能力)

- 易理解性
- 易学性
- 易操作性
- 吸引性

0 维护性 (软件产品可被修改和维护的能力)

- 成熟性
- 容错性
- 易恢复性

# 第四章 作业情况

◦ 可移植性 (软件产品以一种环境迁移到另一种环境的能力)

↳ 易安装性、共存性、易替换性

3) 评价 Windows 和 Linux 系统:

◦ 可靠性: ① 就易恢复性而言, 当 linux 系统遇到重大故障, 例如系统不能够引导, 恢复起来会相当困难, 而 Windows 拥有文件保护功能, 可以在遇到故障时自动把服务器和应用程序恢复到故障前的状态, 需要时间很短;

② 就容错性而言, Windows 的分布式文件系统提供了额外的容错功能。DFS 允许用户在多个不同的服务器上建立大型的虚拟文件系统, 如果一个服务器出现故障, 客户端会自动地与其他服务器连接。就可靠性而言, 发展时间更久, 成熟度更高的 Windows 系统

◦ 效率: 效率指规定条件下软件产品占用的资源数量和所提供的功能和性能的能力。包括时间特性 (处理响应时间) 和资源利用性 (CPU 时间和存储资源)。已知目前有 Phoronix Test Suite 测试, 对比 Linux 发行版 20.04 与 Win10 的性能, 涉及网页、Java 等 63 项测试内容



# 第四章 作业情况

容,其中60%的情况下Linux领先,在Web开发和PHP编程上速度则更是快得多。据此,在相同的硬件条件下,Linux的时间特性和资源利用性综合来看,优于Windows系统。

①易用性:①易理解性:由于Linux是使用命令行操作,所以不易理解;

②易学性:Linux更多面向需要完成相关项目的技术开发的专业人员,对于非相关从业者,需要时间进行学习;

③易操作性:Windows界面为图形操作,更易操作;

④吸引力:二者都有美观的界面,对用户具有相同的吸引力;

⑤维护性:①易分析性:Linux系统可以看到源代码,内核的模块化程度高,因此更易诊断出软件的失效原因;

②易修改性:Windows和Linux操作系统的代码、设计和文档都很容易被修改;

③稳定性:Linux由于存在大量网络管理、网络服务等



# 第四章 作业情况

方面的功能,故其可以更容易建立高效稳定的防火墙、路由器等,因此稳定性也会更好;

0 可移植性: ① 易安装性: windows 更容易安装;

② 共享性: linux 有更好的网络支持和文件支持,它支持嵌入式开发;

③ 易替换性: windows 相比 linux 而言,更容易被卸载;

0 功能性: 对于功能性分析而言,其中适用性、准确性及互操作性, windows 和 linux 操作系统都很好地表现出以上特性,完成其对应应用的需求。对于功能性中子特性的密安性,应单独分析;

(4) 评价 Linux 和 Windows 的密安性:

密安性: 许多时候人们称之为信息安全性,是指软件产品保护信息和数据的能力,以使未授权的人员或系统不能阅读或修改这些信息和数据,而不拒绝授权人员或系统对它们的访问。

1° linux 系统采用模块化设计,当那感觉 linux 系统的某个部分不安全时,可以随时移除掉这个组件,保证系统安全;

2° linux 中大量的网络管理,网络服务方面的功能,可以使用建立高效稳定的防火墙,路由器,服务器等,为提高安全性,它还提供大量的网络分析软件和安全软件等,因此 linux 密安性更好。A+

- 作业:

- 当用户把个人身份证号放入识别, 系统能自动读取号码, 之后显示目的地名菜单。用户选择其目的地, 并输入信用卡和密码后, 系统就输出火车票, 并扣除相应的费用。要求系统反应快, 同时支持多个用户买票。

- 要求

- 1) 讨论上面需求的歧义语句和遗漏
    - 2) 指出非功能需求,
    - 3) 用结构化语言改写之, 尽可能做到准确和无二义性

- 1) 讨论上面需求的歧义语句和遗漏
  - 歧义
    - 身份证号、放入识别
    - 读取号码
    - 输入信用卡和密码
    - 扣除相应费用包括哪些
  - 遗漏
    - 身份证放哪识别
    - 系统反应速度
    - 用户并发数
    - 只描述主成功场景、没有异常场景
    - 除了选择目的地还要选择乘车时间、车票类型等其他信息
- 2) 指出非功能需求
  - 反应速度、并发量、密安性、易恢复性
- 3) 用结构化语言改写之，尽可能做到准确和无二义性



# 第八章 作业情况

- 总体情况：本科生66，收到作业60份。成绩为：A+有4个，A为26，B为13，C为17。
- 主要问题：
  - 1) 歧义语句和遗漏的地方写的不全；
  - 2) 非功能需求写的不完整或没有结合具体内容进行分析；
  - 3) 结构化语言格式不准确。

# 第八章 作业情况

班主任: 2020211317

学号: 2020211706

姓名: 程康伟

## Chapter 8

### 1. 歧义语句:

- 1) “把个人身份证号放入识别”,  
首先“个人身份证号”与“放入”搭配不当,且并未指明放入地点。  
正确应为“把个人身份证放入售票机的身份证识别区域进行识别”。
- 2) “系统能自动读取号码”,  
并未指明谁的号码,应为“系统能自动读取身份证号码”。
- 3) “输入信用卡”,  
动宾搭配不当,应为“插入信用卡”
- 4) “扣除相应的费用”,  
模糊不清,应为“扣除车票、手续费等费用”

### 2. 遗漏部分:

- 1) 业务流程不完整,选择完目的地后,还需选择车站,出发日期,出发时间,座位等级,座位序号等。
- 2) 要求系统反应快,并未给出完整具体的衡量标准,比如在2秒内对操作做出响应。或是采用最低M,期望D,最好B的衡量标准(M:3, D:2, B:1),即最低3秒,期望2s,最好1s。
- 3) 仅给出了成功描述,对于出现的错误等场景并未定义。如:“信用卡密码错误”、“车票已售完”、“网络异常”、“机器故障”等错误场景也应给出处理应对的流程。
- 4) 并未给出其他支付方式,例如,“支付宝”、“微信”、“银行卡”、“电子人民币”等,选择完相应的支付方式后跳转到对应的支付窗口完成支付。



# 第八章 作业情况

## 3 非功能需求

- 1) 要求系统的响应时间快, 高响应
- 2) 支持多个用户订票, 高并发, 高负载
- 3) 对于涉及信息的操作, 要安全, 可靠。

4) 若系统发生故障时, 要求修复性要好, 能够快速处理错误故障, 进行正常的运行。

5) 作为一个全年龄段使用的系统, 与用户的交互界面应该做到易读, 易理解。

6) 对于残障人士提供支持和帮助, 例如语音播报, 呼叫工作人员帮助等

## 4. 结构化语言修改.

1) <售票机>应当<识别读取><个人身份证>, 当<用户放身份证放入身份证识别区域>

2) <售票系统>应当<显示目的地菜单>, 当<身份证信息识别确认完毕>

3) <售票机>应当<提示选择><支付方式>, 当<用户选择目的地, 日期, 车次, 座位后>

4) <售票系统>应当<扣除><车票, 手续费等相关金额>, 当<用户选择支付方式, 并正确验证后>

5) <售票机>应当<打印><车票>, 当<金额提交完毕后>

6) <售票系统>应当<响应><用户操作>, 其<响应速度>不大于<2><秒>

7) <售票系统>应当能够<同时满足><多个用户的并发请求>, 其<最大同时访问数>不低于<600><人>



# 第十章 作业题目

- 1) 论述一个C语言的IDE(集成开发环境), 例如, 微软的, 其包括编辑器、编译器、链接器(link)、调试器(debug)等的体系结构, 说明其中每个部件(或子系统或工具)的作用等。
- 2) 在此设计中, 你采用了哪些设计方法?
- 3) 该IDE的有哪些质量特征, 如何满足的?





# 第十章 作业情况

- 总体情况：本科生66，收到作业62份。成绩为：A+有11个，A为30，B为16，C为5。
- 主要问题：
  - 1) 未指出具体的IDE；
  - 2) 质量特征的描述过于简短，或者不准确；
  - 3) 未写出如何满足以上质量特征；
  - 4) 设计方法过于简单。

# 第十章 作业情况

2020211317  
2020211720 新学步

(1) 这里将以微软的 Visual Studio 为例, 论述一个 C 语言的 IDE 集成开发环境

① 编辑器: 结构化的编辑器不仅具有正文编辑和修改功能, 而且还能够像编译程序那样对用户所输入的源程序进行分析, 把恰当的层次化结构办到程序上, 让用户在源语言的语法判导下编写程序。那么以 Visual Studio 为例, 分析其编辑器的功能:

a. 语法着色: 用不同颜色对代码和标记文件中某些元素着色, 从而将它们区分开来。例如, 关键字 (如 while, if) 用一种颜色, 类型用另一种颜色

b. 错误和警告标记: 添加代码和脚本方案时, 可能会看到不同颜色的波浪下划线 (波形曲线) 或者灯泡显示在代码当中

c. 括号匹配: 插入点设置在代码文件中的左括号上时, 将突出显示左大括号及与之相匹配的右大括号

d. 结构可视化工具: 在代码文件中, 成对的大括号用虚线相连, 方便轻松地辨识左大括号和右大括号

e. 行号: 可在代码窗口的左边距中显示行号。默认下不显示

f. 更改跟踪: 左边距的颜色方便开发人员能够跟踪在文件中所做的更改

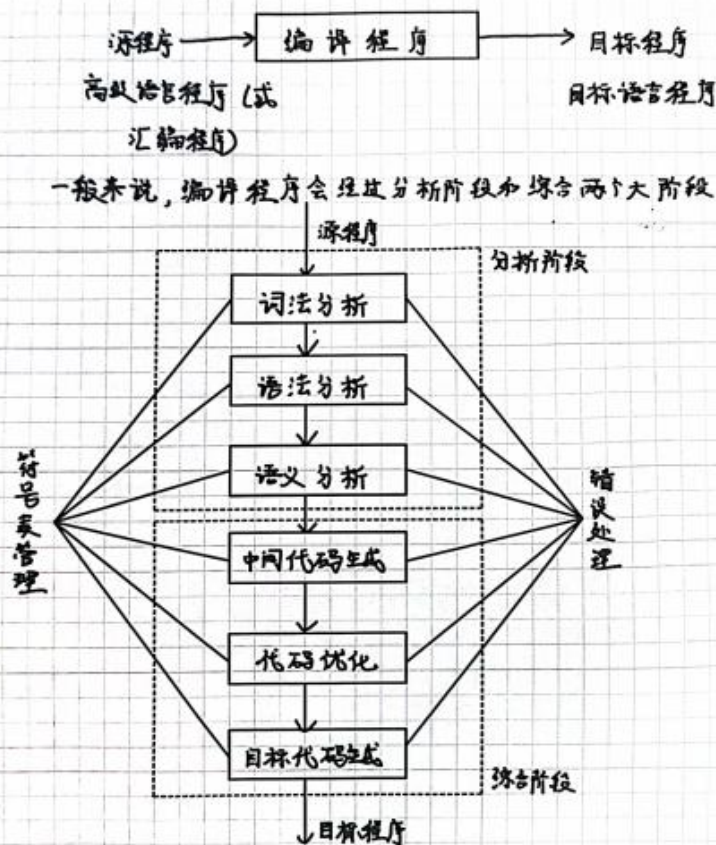
g. 选择代码文本: 可以在标准的连续模式或框模式中选择文本, 在其中选择一个矩形部分文本而非一组文本

h. 全局撤消和重做: “编辑”菜单上的“撤消上次操作”和“重做上一次操作”可撤消或重做影响多文件的全局操作

另外 Visual Studio 的编辑器部分还有许多其它功能, 这里不再赘述

# 第十章 作业情况

② 编译器会将源程序翻译成另外一种表示形式,这种形式应当是能够在计算机上直接运行的。编译程序扫描所输入的源程序,并将其转换为目标程序。如下图所示



具体到 Visual Studio 上,编译阶段做的工作即:将每个 .cpp/.c 和相应的 .h 文件编译成 obj 文件

# 第十章 作业情况

③连接器(link):其作用是把多个经过编译(或汇编)的目标模块连接装配成一个完整的可执行程序

连接装配程序由连接编辑程序和重定位装配程序组成

a. 连接装配程序:扫描外部符号表,寻找所连接的程序段,根据重定位信息解决外部引用和重定位,最终将整个程序涉及的目标模块调入内存并连接在一起,组成一个待装入的组。

b. 重定位装配程序:把目标模块的相对地址换成绝对地址。

具体到 Visual Studio 当中,连接器(link)所做的工作:

将工程中的所有 obj 文件进行链接,生成最终的 .exe 文件

④调试器(debugger):是用于测试和调试其他程序("目标")程序的计算机程序,调试器的主要用途是在受控条件下运行目标程序,允许程序员跟踪其正在进行的操作并监视可能指示代码故障的计算机资源(通常是目标程序或计算机操作系统使用的内存区域)的更改。典型的调试功能包括在特定点运行或停止目标程序的能力、显示内存、CPU 寄存器或存储设备(如磁盘驱动器)的内容,以及修改内存或寄存器内容以输入可能导致程序执行错误的选定测试数据

于是具体针对 Visual Studio 讨论其在调试器方面的功能,下面将由底层到应用介绍

a. 提供的基本功能: Visual Studio 调试器既可用作源代码级调试器,也可用作计算机级调试器。它既适用于托管代码,也适用于本机代码,可用于调试使用 Visual Studio 支持的任何



# 第十章 作业情况

何语言编写的应用程序,此外,还可以附加到正在运行的进程,监视和调试这些进程。如果正在运行的进程的源代码可用,它会在运行时显示代码。如果不可用,它可以显示反汇编。Visual Studio 调试器还可以创建内存转储,以及稍后加载它们以进行调试。

b. 应用层面提供的功能: Visual Studio 调试器允许设置断点(允许在某个位置暂时停止执行)和监视(在运行过程中监视变量的值)。断点可以有条件的,这意味着当满足条件时,它们将被触发。代码可以单步执行,即一次运行一行源代码。它可以单步执行函数以在其内部进行调试,也可以单步执行函数,即函数体的执行不可用于手动检查。

(2) 在设计方法中可以采取“分割”法、“压缩”法、“组合法”和“资源共享”法

① 分割法:

a. 分割法是将不同的功能特性分割到不同的具有明确接口定义的部分中,其目的是把整个系统功能提高成一部分。以编译器为例:将编译器分解成多个阶段:词法分析、语法分析、代码生成等阶段。这是一个明显的,以分割为主的设计过程。

b. 具体到 Visual Studio 当中,一个比较典型的场景是其安装的情况。在 Visual Studio 的安装过程中,可以选择不同的工具。例如在“跨平台移动开发”中,可以对“C#.NET”,“HTML/JavaScript”,“Visual C++”等进行选择性安装。这样可以尽可能降低部分之

# 第十章 作业情况

间的依赖性,以高聚封装其功能,隔离中间件和 COTS 之间的依赖性

## ② 压缩性:

a. 即是消除分割开的系统功能层和接口,即是“分割”方法的逆。

压缩一个软件,意味着把不同功能放到一起

b. 具体到 Visual Studio 中。在 VS 上想要运行代码,只用点击“运行”即可,而不用按个进行编译和链接。而“运行”做的其实是两个步骤,将 .C 文件转化成 obj,再将 obj 转成 exe 文件。这样可以加速系统的开发,消除系统中多个部件上不同的功能需求

## ③ 组合法:

a. 是将两个或多个部件组合到一个更大的部件当中。尤其地,规范化组合是受限的组合操作,即将组合机制限制到一个较小的集合

b. Visual Studio 除了提供普通的代码编译、调试的过程,还提供 web 程序的开发。Visual Studio 的 Web 框架是背多种小组件组合起来,如在 VS 的 Web 框架中,提供了 Angular, jQuery, Bootstrap, Django, Backbone.js 和 Express 的支持。这样可以简化部件的集成、提升作为一个整体的系统的可移植性

## ④ 资源共享法:

a. 是将数据或服务封装,并共享给多个独立的消费者,典型的做法是提供一个资源管理器实施资源访问。开始往往需要花较大的成本,但一旦建立起来,由于减少了部件之间的耦合程度,共享资源可以增强系统的自完整性、可移植性和可修改性

# 第十章 作业情况

b. Visual Studio 是提供共享项目的功能的, 用户可以将所要共享的项目在 Visual Studio 加入协作会话之中 (即 live share), 这样该项目就可以与团队启动协作会话, 共同编写, 以此可以增强该项目的完整性、可修改性和可移植性。

(3) 通过上述的设计方法, 可以对 IDE 的质量特征产生影响, 如下表:

方法	软件质量因素							
	可伸缩性	可修改性	可集成性	可移植性	性能	可靠性	易用性	可维护性
分割 *	+	+	+	+	+/-		+/-	+
抽象	+	+	+	+	-	-	+	+
压缩 *	-	-	-	-	+		+/-	-
松耦合组合 *	+		+				+	
复制	-	-		-	+	+	-	-
资源共享 *		+	+	+	+/-	-	+	+/-

以上的分析方法会对软件造成表中所述影响。例如分割法可以增强 Visual Studio 的可伸缩性、可修改性、可集成性、可移植性。如果开发者想要提高或降低 Visual Studio 的某些特征, 就可以采用上表方法。

同时, 可以根据 GB-T 2566-2006 国家标准中指出的软件质量进行分析。

a. 功能性: 指满足明确或隐含需求的那些功能。通过上述的分析方法已经可以很好地满足该特性。

b. 可靠性: 在规定的时间内条件下, 与软件维持其性能水平的能力有关的一组属性。

c. 易用特性: 指由一组特定或潜在用户为使用软件所需的努力和所



# 第十章 作业情况

作的评价有关的一组属性。Visual Studio 的高集成度与图形界面为用户提供了很好的易用性

d. 效率特征：指与在 规定条件下软件的性能水平与所用资源量之间关系有关的一组属性。Visual Studio 对代码的优化和与 Windows 环境的兼容可以满足这一点。

e. 可维护性：指与进行指定的修改所需的努力有关的一组属性。Visual Studio 中的语法高亮、错误检查方便了代码的维护

f. 可移植性：指与软件从一个环境转移到另一环境的能力有关的一组属性。Visual Studio 充分考虑各平台的性质，以及其高集成度方便了它的移植。

## 第十二章 作业题目

- 1. 用C语言编写一个用学号查询学生名字的程序，分别用顺序查找和二分法查找算法。之后，
  - 1) 画出流程图，
  - 2) 用两种计算方法，分别计算McCabe的复杂度，看是否一样
- 2. 代码质量也可以分为运行时表现出的质量和非运行（工程建造）时的质量指标，请分别列举出3个以上的例子。



# 第十二章 作业题目

- 总体情况：本科生66，收到作业58份。成绩为：A+有8个，A为22，B为25，C为3。
- 主要问题：
  - 1) 算法有错误；
  - 2) 流程图和McCabe计算的式子不匹配；
  - 3) McCabe的复杂度计算有错误；
  - 4) 未计算McCabe；
  - 5) 未给出质量指标或质量指标不恰当。

# 第十二章 作业情况

班级: 2020211316

姓名: 肖鹏辉

编号: 2020211691

第

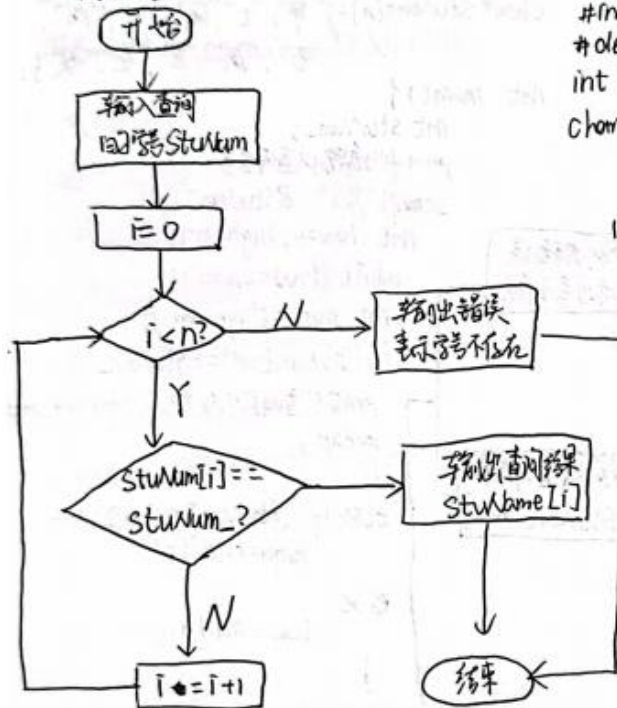
页

1.

## ① 使用顺序查找

假设学号共有  $n$  位 (此处为便于取 10), 存储在 `stuNum` 数组中, 名字同样存储在 `stuName` 数组中, 且与学号数组相匹配 (此处假设学号一位)

流程图



代码

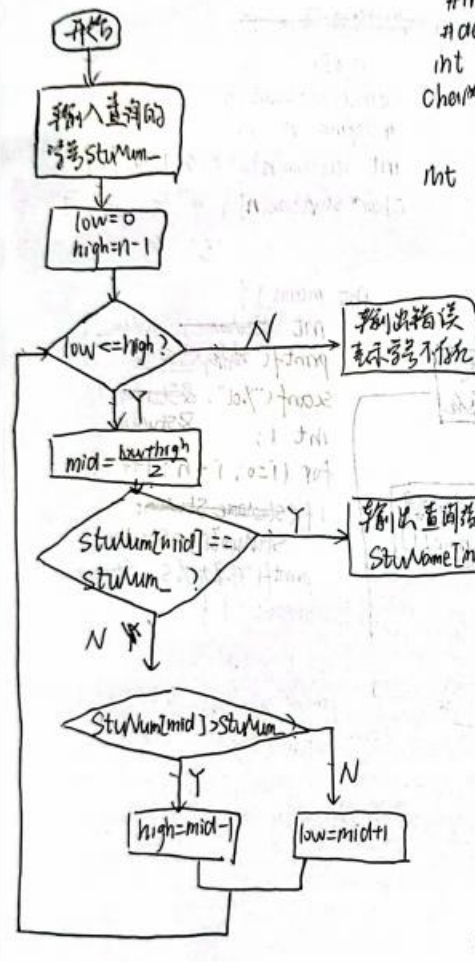
```
#include <stdio.h>
#define n 10
int stuNum[n] = {2, 3, 1, 6, 7, 9, 4, 5, 8, 10};
char* stuName[n] = {"甲", "乙", "丙", "丁", "戊", "己", "庚", "辛", "壬", "癸"};
```

```
int main() {
    int stuName, stuNum;
    printf("请输入查询学号:");
    scanf("%d", &stuNum);
    int i;
    for (i = 0; i < n; i++) {
        if (stuNum == stuNum[i]) {
            printf("结果为 %s", stuName[i]);
            break;
        }
    }
    if (i >= n)
        printf("学号不存在, 查询失败");
    return 0;
}
```

# 第十二章 作业情况

② 使用二分查找。

由于二分查找要求数组按顺序排列，此处找到在 `StuNum` 数组按从小到大的顺序排列流程图。



代码

```
#include <stdio.h>
#define n 10
int StuNum[n] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
char* StuName[n] = {"甲", "乙", "丙", "丁", "戊", "己", "庚", "辛", "壬", "癸"};

int main() {
    int StuNum_;
    printf("请输入查询学号: ");
    scanf("%d", &StuNum_);
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (high + low) / 2;
        if (StuNum[mid] == StuNum_) {
            printf("查询结果为 %s", StuName[mid]);
            break;
        } else if (StuNum[mid] > StuNum_) {
            high = mid - 1;
        } else {
            low = mid + 1;
        }
        if (high < low) {
            printf("学号不存在, 查询失败");
            return 0;
        }
    }
}
```



# 第十二章 作业情况

最后我们依照流程图分别计算两种算法的McCabe复杂度。

易知  $V(G) = e - n + 2p$ ，其中  $n$  为结点数， $e$  为边数， $p$  为连通部件数。

① 顺序查找：  $V(G) = 10 - 9 + 2 = 3$

② 二分查找：  $V(G) = 14 - 12 + 2 = 4$ 。

可见两种算法的McCabe复杂度不相同。

## 2. 代码质量

### 运行时质量

1. 运行时的bug数
2. 运行时处理速度
3. 边界测试下代码的性能高低
4. 运行所与预期结果的出入度
5. 突发情况下代码的鲁棒性与错误处理机制
6. 不同环境下运行时的性能高低

### 非运行时质量

1. 注释行比例
2. 命名与书写的规范性与可读性
3. 使用魔术数字的比例  
(魔术数字指代码中的具体数值，其也  
难以理解数字的意义)
4. 圈复杂度
5. 时间复杂度
6. 空间复杂度

A<sup>+</sup>

# 第十三章 作业题目

给出如下程序的测试用例，要求：

- 1) 语句充分覆盖、分支充分覆盖、多条件覆盖、MC/DC充分覆盖，即覆盖率分别达到100%。(先画出控制流程图，然后分析测试用例)
- 2) 画出 (def/use) 数据流图，给出每个结点的c-use和def

Float computing(alpha, beta , gamma) //接受alpha, beta , gamma

```
{  
  Float x=0.0; y=0.0;  
  while( alpha)  
  { if (beta && gamma )  
    x = 1.0; y = 2.0;  
    else  
    {  
      y= 5.0/(x+1.0);  
      alpha = false;  
    }  
  }  
  y=10.0/ x + y;  
  return y;//输出  
}
```



# 第十三章 作业题目

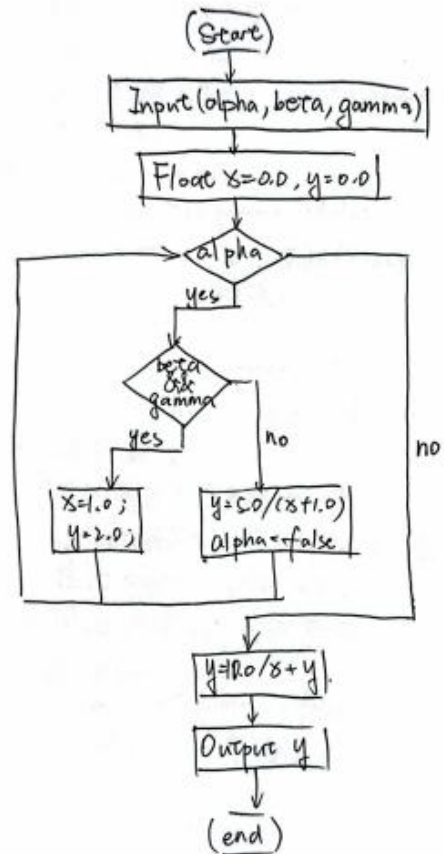
- 总体情况：本科生66，收到作业56份。成绩为：A+有1个，A为37，B为10，C为8。
- 主要问题：
  - 1) 流程图画的不标准或者有错误；
  - 2) 数据流图太乱，缺少结点；
  - 3) 分支充分覆盖基本都理解错误；
  - 4) 结点的c-use和def分析有错误；

# 第十三章 作业情况

北京邮电大学数学作业纸

班级: 姓名: 张盛浩 编号: 第 2 页

控制流程图.



# 第十三章 作业情况

班级:

姓名: 张敬浩

编号:

第 3 页

## 1) 语句充分覆盖:

要求所有语句至少被执行一次

alpha 必过为 true.

同时应分别有 beta & gamma 为 true 和 false 的情况.

故: ① alpha = true.

beta & gamma = false.

此时  $x=0.0$ , 在执行  $y=10.0/x+y$  时  
报除 0 错误.

② alpha = true.

beta & gamma = true.

此  $x \neq 0$ ,  $y \neq 2.0$ , 陷入死循环.

此时存在问题  
永远无法执行  
return y 语句!

## 2) 分支充分覆盖:

要有所有分支至少经过一次.

在 beta & gamma = false 时会置 alpha 为 false.

经历 alpha = false 分支.

故同 1), 只要两个用例即可.

① alpha = true.

beta & gamma = false.

最终报除零错误

② alpha = true.

beta & gamma = true.

陷入死循环



# 第十三章 作业情况

班级: 姓名: 张威浩 编号: 第 4 页

## 13) 多条件覆盖.

要求所有可能的条件取值组合至少出现一次.

$\therefore$   $\alpha = \text{false}$  时, 程序执行与  $\beta$  和  $\gamma$  无关.

$\therefore$  此时无需考虑  $\beta$  和  $\gamma$ .

故用例为:

①  $\alpha = \text{false}$ .

②  $\alpha = \text{true}$ .

$\beta = \text{false}$   
 $\gamma = \text{false}$ .

③  $\alpha = \text{true}$   
 $\beta = \text{false}$   
 $\gamma = \text{true}$ .

④  $\alpha = \text{true}$   
 $\beta = \text{true}$   
 $\gamma = \text{false}$ .

⑤  $\alpha = \text{true}$   
 $\beta = \text{true}$   
 $\gamma = \text{true}$ .

其中 ①, ②, ③, ④ 情况会报错。错误

⑤ 全陷入死循环。

## 14) MC/DC 充分覆盖.

要求“每个入口和退出必须被调用一次, 判断中每个子条件需要包括了所有可能性至少要出现一次, 已经表明每个条件独立地影响最终判断结果次。”

可用如下四个用例:

①  $\alpha = \text{true}, \beta = \text{true}, \gamma = \text{true}$ .

②  $\alpha = \text{false}, \beta = \text{true}, \gamma = \text{true}$ .

③  $\alpha = \text{true}, \beta = \text{true}, \gamma = \text{false}$ .

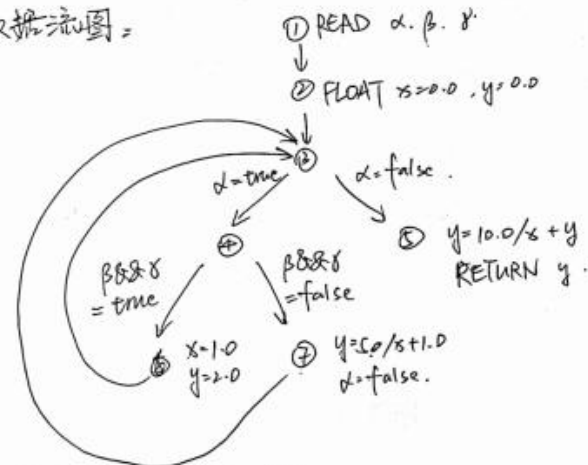
④  $\alpha = \text{true}, \beta = \text{false}, \gamma = \text{false}$ .

# 第十三章 作业情况

其中 ① ~~报错~~ 错误 陷入死循环  
②. ③. ④ 报错 0 错误.

理论上. 对比 ①. ② 可知 alpha 影响  
③. ④ 可知 beta 影响  
①. ② 可知 gamma 影响

数据流图 =



每个结点 c-use 和 def:

	1	2	3	4	5	6	7.
c-use.	∅	∅	∅	∅	{x, y}	∅	{x}
def	{α, β, γ}	{x, y}	∅	∅	∅	{x, y}	{y, α}

A<sup>+</sup>