- 填空 15
- 判断 10
- 应用题 60
  - 理论 30% 实践 70%
  - 作业+课设内容 扣分
- 论述 15
- 测试和需求占分高

- 第一章
  - ppt 1 2 3 4节是重点
  - 软件的定义 必背

    > 软件是计算机系统中与硬件相互依存的另一部分，它包括程序、数据及相关文档的完整集合。其中，程序是按事先设计的功能和性能要求执行的指令序列；数据是使程序能正常操纵信息的数据结构；文档是与程序开发、维护和使用有关的图文材料。

  - 软件的特点 背且懂 可能有判断

    > 1. 软件是一种逻辑实体，而不是具体的物理实体。
    > 2. 软件的生产与硬件不同。（无明显的制造过，存在软件产品的保护问题。）
    > 3. 在软件的运行和使用期间，没有硬件那样的机械磨损、老化等问题。
    > 4. 软件的开发和运行常常受到计算机系统的限制，对计算机系统有着不同程度的依赖性。
    > 5. 软件的开发至今尚未完全摆脱手工艺的开发方式。
    > 6. 软件是复杂的。（软件复杂性来源于它所反映的实际问题的复杂性。）
    > 7. 软件成本相当昂贵。（定制产品、手工开发.成本高）
    > 8. 相当多的软件工作涉及到社会问题。

- 分类 了解

  > 1. 按软件功能划分：
  >
  >   1) 系统软件：使计算机系统各个部件、相关软件和数据协调、高效的工作的软件。（如：操作系统，数据库管理系统，设备驱动程序等）
  >
  >   2) 支撑软件：协助用户开发软件的工具性软件。（如：文本编辑程序，集成开发工具，图形软件包等）
  >
  >   3) 应用软件：在特定领域内开发为特定目的服务的一类软件。
  > 2. 按软件规模划分：
  >
  >   微型 1人 1—4周 0.5K
  >
  >   小型 1人 1—6月 1—2K
  >
  >   中型 2—5人 1—2年 5—50K
  >
  >   大型 5—20人 2—3年 50—100K
  >
  >   甚大型 100—1000人 4—5年 1M
  >
  >   极大型 2000—5000人 5—10年 1M—10M
  >
  >   微型 1人 1—4周 0.5K
  > 3. 按软件的工作方式划分：
  >
  >   1) 实时处理软件：在事件或数据产生时，立即予以处理，并及时反馈信号。
  >
  >   2) 分时软件：允许每个联机用户同时使用计算机。
  >
  >   3) 交互时软件：能实现人通信的软件。
  >
  >   4) 批处理软件：把一组输入作业或一批数据以成批处理的方式一次运行，按顺序逐个处理完的软件。
  > 4. 按软件服务对象的范围划分：
  >
  >   1) 项目软件
  >
  >   2) 产品软件

- 软件危机的两个方面 搞清楚 可能有填空

> 指在计算机软件的开发和维护过程中所遇到的一系列严重问题。

- 危机表现 不背 理解

> 1) 对软件开发的成本和进度的估计常常不准确。
> 导致：成本提高，工程延期，影响信誉。
> 权益之计：损害软件质量，又会引起用户不满。
> 2) 用户对“以完成”的软件系统不满意的现象经常发生。
> 原因：对用户需求不确切，缺少沟通，仓促上阵，闭门造车。
> 导致：不符合用户要求。
> 3) 软件产品质量往往靠不住。
> 原因：软件可靠性和质量保证未认真执行。
> 导致：软件质量问题。
> 4) 软件常常是不可维护的。
> 原因：程序结构固定、死板、变更困难、错误、难以改正，无法增加新的功能和适应新的环境。
> 5) 软件通常没有适当的文档资料。
> 项目负责人：用以控制整体状态，把握工程进度；
> 开发者：用以相互交流；
> 维护人员：维护的依据。
> 6) 软件成本在计算机系统中成本所占比例率上升。
> □ 微电子技术的进步和自动化程度的不断提高，导致硬件成本下降；
> □ 软件需要手工劳动，且大规模和数量不断的扩大，导致软件成本上升。
> 7) 软件开发生产率提高的速度，远远跟不上计算机普及、深入的趁势。
> “供不应求”，无法充分利用硬件。

- 危机原因 背

> 1. 与软件自身的特点有关：
>   逻辑实体、手工开发、复杂度高、成本昂贵。
> 2. 与开发、维护方法不正确有关：
>   忽视用户需求，轻视软件维护。

- 解决途径 ppt p12 底部高亮背

> 总之，为了解决软件危机，既要有技术措施(方法和工具)，又要有必要的组织管理措施。软件工程正是从管理和技术两方面研究如何更好地开发和维护计算机软件的一门新兴学科。

- 什么是软件工程 背

> 是采用工程的概念、原理、技术和方法来指导软件开发和维护的工程学科。

- IEEE 理解 高亮背下来 ppt p15

> ①应用系统化的、学科化的、定量的方法，来开发、运行和维护软件，即，将工程应用到软件。
> ②对①中各种方法的研究。
> 其中，采用的方法包括：
> □计算机科学和数学用于构造模型、分析算法；
> □工程科学用于制定规范、明确风险、评估成本和确定权衡；
> □管理科学用于进度、资源、质量、成本的管理。

- 基本原理
  - 七条全背 理解透彻
  - 以前在论述考过

    > □用分阶段的生命周期计划严格管理。
    > □坚持进行阶段评审。
    > □实行严格的产品控制。

> □采用现代的程序设计技术。
> □结果应能清楚地审查。
> □开发小组的人员应该少而精。
> □承认不断改进软件工程实践的必要性。

- 三个时期 八个阶段
  - 阶段 背 任务说清楚
  - 生命周期方法学定义 了解 word p8

> 1) 三个时期：
> □ 软件定义：确定工程总目标：可行性、采用的策略，需求完成的功能，需要的资源和成本，工程进度表。
> 包括：问题定义，可行性研究，需求分析。
> □ 软件开发：具体设计和实现。
> 包括：概要设计、详细设计（系统设计），编码和单元测试、综合测试（系统实现）
> □ 软件维护：使软件持久地满足用户需要。
> 改正错误，适应新环境，满足新需求。
> 2) 八个阶段：
> □ 问题定义：“要解决的问题是什么？”
> 提出关于问题性质、工程目标和规模的全面报告。
> □ 可行性研究：“对上一个阶段所确定的问题有行的通解决办法吗？”
> 研究问题的范围，进行成本/效率分析，探索问题是否值得解和如何解。
> □ 需求分析：“为了解决问题，目标系统必须做到什么？”
> 确定目标系统所应具备的功能，建立系统逻辑模型（数据流图、数据字典、简要算法）
> □ 概要设计：概括地谈，应该如何解决问题
> 提出几种设计方案：低成本，中等成本，高成本（“十全十美”），确定解决系统的方案和目标系统需要那些程序，设计软件的结构，确定程序模块及模块间关系（层次图或结构图）。
> □ 详细设计：应该怎样具体地实现系统
> 把解决具体化，设计出程序的详细规格说明（HIPO图或PDL语言）
> □ 编码和单元测试：编写程序模块的实现代码，并对其进行测试。
> □ 综合测试：通过各种类型的测试使软件达到预定要求。
> □ 集成测试：根据设计的软件结构，将单元模块按某种策略装配起来进行联合测试。
> □ 验收测试：由用户根据需求规格说明书对目标系统进行整体验收。
> □ 软件维护：通过各种必要的维护活动使系统持久满足用户需要。
> □ 改正性维护（21%）
> □ 适应性维护（25%）
> □ 完善性维护（50%）
> □ 预防性维护（4%）
> 3) 目的和实质：
> 控制开发工作的复杂性，通过有限的确定步骤，把用户需求从抽象的逻辑概念转化为具体的物理实现。

- 画模型按八个阶段画
- 瀑布 特点 & 缺点

> 瀑布模型的特征：
> □从上一项活动中接受该项活动的工作成果（工作产品），作为输入。
> □利用这一输入实施该项活动应完成的内容
> □给出该项活动的工作成果，作为输出传给下一项活动
> □对该项活动实施的工作进行评审。若其工作得到确认，则继续下一项活动。
> 瀑布模型的缺点：
> □从认识论角度看，人的认识是一个多次反复循环的过程，不可能一次完成。但瀑布模型中划分的几个阶段，没有反映出这种认识过程的反复性。特别是瀑布模型过于依赖早期进行的唯一一次需求调查，

> 不能适应需求的变化；
> □软件开发是一个知识密集型的开发活动，需要相互合作完成，但瀑布模型没有体现这一点。特别是由于瀑布模型是单一流程，开发中的经验教训不能反馈应用于本产品的过程。

- 原型 ？？
- 第二章
  - 可行性定义 稍微记一下

    > 可行性研究的含义：
    > □可行性的含义包括可能性、必要性。
    > □可行性分析的对象是系统目标。评价总体方案（系统目标）的可能性、必要性。
    > □甲方和乙方不同的立场，可行性分析是不同的。我们主要以乙方的立场进行分析

  - 可行性内容
    - 技术 经济 操作
  - 可行性目的 & 实质 word p14

    > 1.目的：
    > 用最少的代价，在尽可能短的时间内弄清所定义的项目是不是可能实现和值得进行。（不是解决问题，而是确定问题是否可能解决和值得去解）
    > 2.实质：
    > 是进行一次大大简化了的系统分析和设计的过程，即在较高层次上以较抽象的方式进行的系统分析和设计的过程。

  - 成本/效益分析 了解

    > 通过估计开发成本，运行费用和经济效益，从而达到从经济角度分析开发一个特定的新系统是否划算，帮助使用部门负责人正确的做出是否投资这项工程开发的决定。
    > 1.成本估计：
    > □ 软件开发成本主要表现为人力消耗：
    > 人力消耗×平均工资＝开发费用
    > □ 成本估计技术：
    > □ 代码行技术：源代码行数×每行代码平均成本＝开发成本
    > □ 任务分解技术：按开发阶段划分任务
    > （每个相对独立的开发任务的）成本累加和=开发成本
    > □ 自动估计成本技术：软件工具。
    > 2.运行费用：
    > □ 系统操作费用（操作员人数，工作时间，消耗的物资等）
    > □ 维护费用。
    > 3.经济效益：
    > □ 因使用新系统增加的收入
    > □ 可以节省的运行费用
    > 4.度量效益的方法：
    > 1) 货币的时间价值：
    > 设年利率为i，现已存入P元，则n年后所得：F＝P*(1＋i)n，即为P元钱在n年后的价值。反之，若n年后能收入F元，则其在现在的价值为：P＝F / (1＋i)n。
    > 2) 投资回收期：
    > 是使累计的经济效益等于最初的投资所需要的时间，是衡量一个开发工程价值的经济指标。投资回收期越短，就能越快获得利润，所以工程就越值得投资。
    > 3) 纯收入：
    > 是在整个生存期之内系统的累计经济效益（折合成现在植）与投资之差。
    > 4) 投资回收率：

> 设P为现在的投资的投资额，Fi为第i年底的效益（i＝1，2，…，n），n为系统的使用寿命，j为投资回收率。
> 则 (…(((P(1＋j)－F1)(1＋j)－F2)(1＋j)－…)－Fn＝0
> 即 P＝F1 / (1＋j)＋F2 / (1＋j)2＋…＋Fn / (1＋j)n。

- 可行性结论 能说出几个方面

> □项目可以立即开始进行
> □需要增加资源才能开始，例如增加投资或人力。
> □需要推迟到某些条件具备后才能开始，例如组织机构的调整。
> □需要对系统目标作某些修改才能开始。
> □不能或没有必要进行，例如经济上不合理，投资相差太大。

- 第三章
  - 重点 3.传统需求分析

> 传统软件工程的6个软件生命周期阶段：软件定义、需求分析、软件设计、编码、测试、运行与维护。
> 传统软件过程需求分析阶段的任务是：
> □确定对系统的综合要求；
> □分析系统的数据要求；
> □抽象并确立目标系统的逻辑模型；
> □编制软件需求规格说明。

  - 需求分解 必看
  - 传统需求分析任务 大概知道

> 一．需求分析的任务：
> 1.基本任务：回答“系统必须做什么”？确定目标系统功能和性能。
> 2.具体任务：
> 1)确定对系统的综合要求：功能要求；性能要求；运行要求；将来可能提出的要求。
> 2)分析系统的数据要求：E-R图（概念模型）。
> 3)导出系统的逻辑模型：数据流图，数据字典，加工处理说明书等。
> 4)修正系统开发计划。
> 5)开发原型系统：使用户对目标系统有一个更直接、更具体的概念，从而能更准确提出用户需求。（关键的困难在于成本）

  - 综合要求 功能要求、性能要求、运行要求 等清楚

> □功能要求包括系统应该实现的功能；
> □性能要求包括系统的响应时间、资源限制、数据精确性、系统适应性等；
> □运行要求包括系统硬件环境、网络环境、系统软件、接口等的具体要求；
> □其他要求报刊安全保密、可靠性、可维护性、可移植性、可扩展性等

  - 分层数据流图 五原则 背
    - 连续性一定有
    - 局部文件一定有 外部文件不一定
    - 加工编号一定有

> □ 第一层DFD应当是基本系统模型
> □ 注意父图和子图的平衡，维护信息的连续性
> □ 区分局部文件和局部外部项
> □ 掌握分解的速度，上快下慢
> □ 遵守加工编号原则

  - 。。。。

- 验证软件需求
  - 一致性 完整性 等

> □ 一致性：所有需求必须一致，不能互相矛盾。
> □ 完整性：需求必须完整，包含用户需要的所有功能和性能。
> □ 现实性：指定需求用现有的软、硬件技术基本上可以实现。
> □ 有效性：必须证明需求是正确有效的，确实能解决用户面对的问题。

- 第四章
  - 任务

> 系统概要设计的任务是将分析模型映射为具体的软件体系结构。
> 一．软件设计的任务：
> 把需求阶段所产生的软件需求说明转换为用适当手段表示的软件设计文档。“做什么”——>“怎么做”。

  - 两个阶段

> 二．软件设计划分两个阶段：
> □ 概要设计：确定软件的结构，即软件组成，以及各组成成分（子系统或模块）之间的相互转换。
> □ 详细设计：确定模块内部算法和数据结构，产生描述各模块程序的详细设计文档。

  - 模块、模块化 了解

> 1.模块、模块化：
> □ 模块：是数据说明，可执行语句等程序对象的集合。例：过程，函数，子程序，宏等。
> □ 模块化：是把程序划分成若干个模块，每个模块完成一个子功能，把这些模块集中起来组成一个整体，可以完成指定的功能，满足问题的要求。

  - 信息隐蔽 模块独立性 背

> 3.信息隐蔽：指每个模块的实现细节对于其他模块来说是隐蔽的，即模块中所包含的信息（数据与过程）。应不允许其他不需要这些信息的模块使用（即隐蔽起来）。只有为了完成软件的总体功能而必须在模块间交换的信息。才允许在模块间进行传递。
> 目的：是软件的修改或错误局限在一个或几个模块内部，不会涉及软件其他部分。

    - 耦合

> 1) 度量模块独立性的准则：内聚、耦合。
> □ 内聚：是模块功能强度（即一个模块内部各个元素彼此结合的紧密程度）的度量。模块内部各元素之间联系越紧密，内聚性越强。
> □ 耦合：是模块之间相对独立性（即互相连接的紧密程度）的度量。模块间连接越紧密，联系越多，耦合性越强。
> □ 模块的独立性越高，其块内联系越紧密（内聚性强），块间联系越弱（耦合性越弱）

  - 看图论述？？？
  - 概要设计简单看一下

> 一． 概要设计阶段需要完成的工作：
> 1.制定规范：软件开发组在设计时应共同遵守的标准。
> 1) 规定设计文档的编制标准（文档体系、用纸、样式、记述详细程序、图形画法等）。
> 2) 规定编码的信息形式（代码体系），与硬件、操作系统的接口规约，命名规则等。
> 2.软件结构的总体设计：决定软件的总体结构。
> 1) 采用某种设计方法，将一个复杂的系统按功能划分成模块的层次结构。
> 2) 确定每一个模块的功能，建立与已确定的软件需求的对应关系。
> 3) 确定模块间的调用关系。
> 4) 确定模块间的接口，即模块间传递的信息，设计接口的信息结构。

> 5) 评估模块划分的质量及导出模块结构的规则。
> 3.数据结构的设计：决定文件系统的结构或数据库的模式，子模式以及数据完整性，安全性设计。
> 1) 确定输入、输出文件的详细的数据结构。
> 2) 模式设计：确定物理数据库结构。
> 3) 子模式设计：确定用户使用的数据视图。
> 4) 数据的完整性，安全性设计。
> 5) 数据优化：改进模式与子模式，以优化数据的存取。
> 4.编写文档：《概要设计说明书》
> 1) 引言：编写目的，背景，参数资料等。
> 2) 系统概述：目标，运行环境，需求概述。
> 3) 结构设计：
> □ 软件的总体结构。
> □ 模块的外部设计：（包括关于各模块功能，性能及接口的简要描述）。
> 4) 数据结构设计：文件系统结构，数据库模式，子模式，完整性，安全性，访问方法，存储要求等。
> 5) 初步测试计划：对测试的策略，方法和步骤提出要求。
> 5.技术审查和管理复审。

- 数据流图到SC
- SC形态特征 深度 广度？word p45

- 第五章
  - 结构化系统设计的定义 背下

    > 是一种设计程序的技术，它采用自定向下，逐步求精的设计方法和单入口，单出口的控制结构。

  - 环域复杂度

    > 环形复杂度定量度量程序的逻辑复杂度。有了描绘程序控制流的流图之后，可以用下述3种方法中的任何一种来计算环形复杂度。
    > □(1)流图中的区域数等于环形复杂度。
    > □(2)流图G的环形复杂度V(G)=E-N+2,其中，E是流图中边的条数，N是结点数。
    > □(3)流图G的环形复杂度V(G)=P+1，其中，P是流图中判定结点的数目。

  - 编码风格要求 背

    > 1.实现源程序的文档化：
    > □符号名（即标识符）的命名：名称应该能构反映其所代表的实际东西，具有一定的实际意义，使其能见名知意，有助于对程序功能的理解。
    > □程序进行适当的注解：正确的注解能够帮助读者理解程序，可为后续阶段的测试和维护，提供正确的指导。
    > 注释的位置及情况：
    > 1)每一程序单元开始处。（序号）
    > 2)重要的程序段。（嵌入源代码内部）
    > 3)难懂的程序段。（说明原因或画等效流程图）
    > 4)修改程序。（保持注释，代码一致性）
    > □程序的视觉组织：使用标准的，统一的格式书写源程序清单，有助于改进可读性。
    > 1) 用分层缩进的写法显示嵌套结构的层次。
    > 2) 在注释段周围加上边框。
    > 3) 在注释段与程序段以及不同程序段之间插入空行。
    > 4) 每行只写一条语句。
    > 5) 书写表达式时，适当使用空格或圆括号等做隔离符。
    > 2.数据说明：常量，变量等的声明。
    > □ 数据说明的次序应当规范化，使数据属性容易查找，有利于测试，排错和维护。
    > □ 常量说明—>简单变量类型说明—>数组说明—>公用数据模块说明—>所有文件说明。

> □ 整数量说明－>实型量说明－>字符量说明－>逻辑量说明
> □ 当每个变量名用同一个语句说明时，应将变量按字母顺序排列。
> □ 如果设计了一个复杂数据结构，应使用注释说明在程序实现时这个数据结构的特点。
> 3.语句结构：语句构造应力求简单、直接、不能为了片面的追求效率而使语句复杂化。
> □在一行内只写一条语句，并采取适当的缩进方式，使程序的逻辑和功能变得更加明确。
> □ 程序编写应当首先考虑要清晰性，不要刻意的追求技巧和效率，而丧失清晰。
> □ 首先要保证程序正确，然后才要求提高速度。
> □ 尽量使用公共过程或子程序去代替重复的功能代码段。
> □ 使用括号来清晰地表达算术表达式和逻辑表达式的运算顺序。
> □ 使用标准的控制结构，有规律地使用GOTO语句。
> □ 尽量减少使用“否定”条件的条件语句。（效率低且不易读）
> □ 避免使用过于复杂的条件测试。
> □ 避免过多的循环嵌套和条件嵌套。
> □ 要模块化，确保每个模块的独立性。
> 4.输入和输出：输入和输出的方式和格式应尽可能方便用户的使用，尽量做到对用户友善。
> □ 对所有输入数据都进行检验。
> □ 检查输入项，重复组合的合法性。
> □ 保持输入格式简单。
> □ 使用数据结束标志，不必要求用户指定数据的数目。
> □ 明确提示交互式输入的请求，详细说明可用的选择和边界数目。
> □ 当程序设计语言对格式有严格要求时，应保持输入格式一致。
> □ 设计良好的输出报表。
> □ 给所有输出数据加标志，给出必要的说明。

- 编码目的 word 两句话

> □ 是使用选定的程序设计语言，把模块的过程描述翻译为用该语言书写的源程序（或源代码）。模块的过程描述——>源程序。
> □ 编码是设计的自然结果，程序的质量主要取决于设计的质量。

- 第六章
  - 测试的定义 & 目的

> 定义
>
> 是为了发现错误而执行程序的过程，即根据软件开发各阶段的规格说明和程序的内部结构而精心设计一批测试用例，并利用这些测试用例去运行程序，以发现程序错误的过程。
> 目的
> □软件测试是为了发现错误而执行程序的过程
> □一个好的测试能够在第一时间发现程序中存在的错误
> □一个好的测试是发现了至今尚未发现的错误的测试。软件测试是质量控制的重要手段，保证客户拿到或用户使用高质量的软件产品。

  - 分类 了解
  - 测试步骤 五个空 验收测试

> 单元测试，综合测试，确认测试、系统测试、验收测试

  - 误区和原则 了解

> 软件测试的误区：
> □误区一：如果发布出去的软件有质量问题，都是软件测试人员的错
> □误区二：软件测试技术要求不高，至少比编程容易多了
> □误区三：有时间就多测试一些，来不及就少测试一些
> □误区四：软件测试是测试人员的事，与开发人员无关
> □误区五：根据软件开发瀑布模型，软件测试是开发后期的一个阶段

> □所有测试的标准都是建立在用户需求之上。
> □软件测试必须基于“质量第一”的思想去开展各项工作，当时间和质量冲突时，时间要服从质量。
> □事先定义好产品的质量标准，只有有了质量标准，才能根据测试的结果，对产品的质量进行分析和评估。
> □软件项目一启动，软件测试也就是开始，而不是等程序写完，才开始进行测试。
> □穷举测试是不可能的。甚至一个大小适度的程序，其路径排列的数量也非常大，因此，在测试中不可能运行路径的每一种组合。
> □第三方进行测试会更客观，更有效。
> □软件测试计划是做好软件测试工作的前提。
> □测试用例是设计出来的，不是写出来的，所以要根据测试的目的，采用相应的方法去设计测试用例，从而提高测试的效率，更多地发现错误，提高程序的可靠性。
> □对发现错误较多的程序段，应进行更深入的测试。一般来说，一段程序中已发现的错误数越多，其中存在的错误概率也就越大。
> □重视文档，妥善保存一切测试过程文档（测试计划、测试用例、测试报告等）
> □应当把“尽早和不断地测试”作为测试人员的座右铭
> □回归测试的关联性一定要引起充分的注意，修改一个错误而引起更多错误出现的现象并不少见
> □测试应从“小规模”开始，逐步转向“大规模”。
> □不可将测试用例置之度外，排除随意性。
> □必须彻底检查每一个测试结果。
> □一定要注意测试中的错误集中发生现象，这和程序员的编程水平和习惯有很大的关系
> □对测试错误结果一定要有一个确认的过程。

  - 测试用例 设计
  - 等价分类 逻辑覆盖
  - 几种覆盖方式
- 第七章
  - 软件维护定义目的分类 背

> 1.软件维护：是在软件已经交付使用之后，为了改正错误或满足新的需要而修改软件的过程。
>
> 2.维护的目的：是满足用户对已开发产品的性能与软件环境不断提高的需求，进而达到延长软件的寿命。
> 分类
> 1.完善性维护：在软件使用过程中，为了满足用户对软件的功能与性能提出新的需求而进行的维护。(50%)
> 2.适应性维护：使软件适应运行环境(包括软，硬件环境及数据环境)的变化而进行的维护。(25%)
> 3.纠错性维护：为纠正在开发期间未能发现的错误而进行的维护。(21%)
> 4.其他维护：（如：预防性维护）为改善软件的可维护性，可靠性等，以减少今后对其进行维护所需的工作量的工作而进行的维护。(4%)

  - 可维护性定义 背f

> 是指纠正软件系统出现的错误或缺陷，以及为满足新的要求进行修改，扩充或压缩的容易程度，即衡量维护容易程度的一种属性。