1. 软件生存周期中，准确地确定软件系统必须做什么的阶段是( B )

A.可行性分析和项目开发计划 B.需求分析

C.概要设计 D.详细设计

2. 若一个模块中各个处理元素都密切相关于同一功能且必须顺序执行，前一功能元素的输出就是下一功能元素的输入，则该模块的内聚性类型是( C )

A.时间内聚 B.偶然内聚

C.顺序内聚 D.功能内聚

3. 下列几种耦合类型，耦合性最高的是( A )

A.公共耦合 B.控制耦合

C.标记耦合 D.数据耦合

4. 程序流程图的优点不包括( B )

A.历史最悠久，使用最广泛 B.容易表示逐步求精的过程

C.支持程序的三种基本控制结构 D.直观清晰，易于使用

5. 确认测试也称为( A )

A.有效性测试 B.无效性测试

C.静态测试 D.动态测试

6. 黑盒测试技术不包括( B )

A.等价类划分 B.路径覆盖

C.边界值分析 D.错误推测

7. 为了增加软件功能和性能、提高软件运行效率而进行的维护活动称为( C )

A.校正性维护 B.适应性维护

C.完善性维护 D.预防性维护

8. 瀑布模型与增量模型之间的本质区别是( B )

A.瀑布模型属于非整体开发模型，增量模型属于整体开发模型

B.瀑布模型属于整体开发模型，增量模型属于非整体开发模型

C.瀑布模型容易适应需求变化，增量模型不容易适应需求变化

D.瀑布模型减少技术、应用风险，增量模型增大技术、应用风险

9. 面向对象分析中，对象模型描述了系统的( A )

A.静态结构 B.动态结构

C.控制结构 D.顺序结构

1. 为了支持软件人员开发和维护活动而使用的软件一般是指 软件工具 。

2. 在软件结构的设计中，各个模块之间要力求降低耦合性，提高 内聚性 。

3. 数据流图(DFD)一般可分为 变换型 和 事务性 两类。

4. 软件维护的内容包括校正性维护、适应性维护、 完善性维护 和预防性维护。

5. 状态图反映了 事件与状态 的关系。

6. 组成软件的三个部分是： 指令集、数据和文档 。

7. 变换型数据流图是由输入、 变换（或处理） 和输出三部分组成的。

8. 结构化语言是介于 自然语言 和形式语言之间的一种半形式语言。

9. 软件产品在交付使用之前一般要经过 单元测试 、集成测试、确认测试和系统测试。

10. 子类自动共享父类数据结构和方法的机制是 继承性 ，这是类之间的一种关系。

11. 数据流图描绘数据在系统中 流动和处理 的过程。

12. 测试用例应由输入数据和预期的 输出结果（输出数据） 两部分组成。

13. 软件详细设计阶段常采用的工具有图、表和 语言 。

14. 在软件的详细设计中，根据控制流程从上到下从左到右展开的设计工具是 PAD图 。

15. 一个模块通过传递开关、标志对某一模块的多种功能进行选择，则这两个模块之间的耦合方式是 控制耦合 。

16. 结构化分析是面向 数据流 进行需求分析的方法。

17. 在软件测试的所有覆盖方法中发现错误能力最弱的覆盖是 语句覆盖 。

18. 黑盒测试又被称为数据驱动测试或 基于规格说明的测试 。

19. 数据字典有四类条目，分别是数据流、 数据项 、数据存储和数据加工。

1. 软件工程方法学的三要素。

过程、方法、工具。

2. 数据字典的组成。

数据流条目、数据项条目、数据文件条目、数据加工条目。

3. 验证软件需求的四个方面。

验证需求的一致性：任何一条需求不能和其他需求互相矛盾。

验证需求的完整性：需求规格说明书中应包括用户需求的每一个功能或性能。

验证需求的有效性：必须证明需求是正确有效的，确实能够解决用户面对的问题。

验证需求的现实性：需求在现有硬件和软件技术水平上应该是能够实现的。

4. 系统总体设计过程。

模块化：模块化的依据是使问题复杂度降低，易实现易理解。

抽象：将现实世界中具有共性的一类事物的相似的、本质的方面集中概括起来，而暂时忽略它们之间的细节差异。

逐步求精：为了能集中精力解决主要问题而尽量推迟对问题细节的考虑。逐步求精最初是自顶向下的设计策略，程序的体系结构通过逐步精化处理过程的层次设计出来。

信息隐藏：模块内部的信息对于不需要这些信息的模块来说是不能访问的。提高模块的独立性，模块之间的信息传递只能通过合法的调用接口来实现。

模块独立：模块独立的概念是模块化、抽象、信息隐蔽概念的直接结果。

5. 软件结构设计的启发式规则。

改进软件结构提高模块独立性：通过模块分解或合并，降低耦合提高内聚。

模块规模应该适中：一个模块的规模不应过大，通常不超过 60 行语句。

深度、宽度、扇出和扇入都应适当：顶层高扇出，中层低扇出，底层高扇入。

模块的作用域应该在控制域范围之内。

力争降低模块接口的复杂程度：模块接口设计原则是易理解，传递信息简单且与模块功能一致。

设计单入口单出口的模块：不要使模块间出现内容耦合。当从顶部进入模块并且从底部退出来时，软件是比较容易理解的，因此也是比较容易维护的。

模块功能应该可以预测：相同的输入应产生相同的输出。