2024-软件工程-作业5

题量: 43 满分: 100 作答时间:10-21 16:18至10-31 16:18

一. 单选题（共25题，50分）

1. (单选题, 2分) 所谓“高产”的测试是指（ ）。

A用适量的测试用例，说明被测程序正确无误

B用适量的测试用例，说明被测程序符合相应的要求

C用少量的测试用例，发现被测程序尽可能多的错误

D用少量的测试用例，纠正被测程序尽可能多的错误

2. (单选题, 2分) 在集成测试过程中，如果要在测试的早期对主要的控制或关键的抉择进行检验，并且要在早期实现软件的一个完整的功能并验证这个功能，那么我们可以选用（ ）。

A自底向上集成

B三明治集成

C一次性集成

D自顶向下集成

3. (单选题, 2分) 如果一个软件是给许多客户使用的，大多数软件厂商要使用几种测试过程来发现那些可能只有最终用户才能发现的错误。（ ）测试是由用户在开发者的场所来进行的。

A、Alpha

B、Beta

C、Gamma

D、Delta

4. (单选题, 2分) 如果一个软件是给许多客户使用的，大多数软件厂商要使用几种测试过程来发现那些可能只有最终用户才能发现的错误。（ ）测试是由软件的最终用户在一个或多个用户实际使用环境中来进行的。

A、Alpha

B、Beta

C、Gamma

D、Delta

5. (单选题, 2分) 关于编写程序代码，有下述说法：

①在一行内只写一条语句

②避免使用过于复杂的条件判断，尽量少用“否定条件”的条件语句

③模块功能尽量单一化，确保每个模块的独立性

④尽量采用基本的控制结构来编写程序

⑤尽量使用标准库函数，不要自行编写跟库函数功能一样的自定义函数

⑥避免大量使用循环嵌套和条件嵌套

⑦尽量编写技巧性很高的代码

其中，正确的说法有（ 3 ）。

A①③④⑦

B②③④⑦

C①②③⑤⑦

D①②③④⑤⑥

6. (单选题, 2分) 以下叙述中**不符合**程序设计风格指导原则的叙述是（ ）。

A、嵌套的重数应加以限制

B、不用可以省略的括号

C、把常见的局部优化工作留给编译程序去做

D、使用有意义的变量名

7. (单选题, 2分) **不属于**序言性注释内容的是( )。

A、模块设计者

B、修改日期

C、程序的整体说明

D、语句功能

8. (单选题, 2分) 如果编写系统软件,可选用的语言是( )。

A、FoxPro

B、COBOL

C、C

D、FORTRAN

9. (单选题, 2分) 与编程风格有关的因素**不包括**( )。

A、源程序文档化

B、语句构造

C、输入输出

D、程序的编译效率

【解析】程序的编译效率：这与编程风格无关，它涉及到程序编译后的性能，而不是代码本身的编写风格。编译效率是指程序编译后运行的效率，与代码的可读性和结构没有直接关系。

10. (单选题, 2分) 对建立良好的编程风格,下面描述正确的是( )。

A、程序应简单、清晰、可读性好

B、符号名的命名只要符合语法即可

C、充分考虑程序的执行效率

D、程序的注释可有可无

【解析】

A、程序应简单、清晰、可读性好：这是良好编程风格的关键部分，良好的编程风格强调代码的可读性和可维护性，使得其他开发者能够容易理解代码的意图和逻辑。

B、符号名的命名只要符合语法即可：这个描述是错误的。良好的编程风格要求变量名和函数名等符号名应该具有描述性，能够清晰地表达它们的用途和含义，而不仅仅是符合语法。

C、充分考虑程序的执行效率：虽然程序的执行效率是一个重要的考虑因素，但它不是编程风格的主要关注点。良好的编程风格更侧重于代码的可读性、可维护性和一致性。

D、程序的注释可有可无：这个描述是错误的。良好的编程风格鼓励添加有意义的注释，以帮助解释代码的意图和逻辑，特别是对于那些不是一目了然的复杂部分。注释对于代码的可读性和可维护性至关重要。

11. (单选题, 2分) 源程序中应包含一些内部文档,以帮助阅读和理解程序，源程序的内部文档通常包括合适的标识符、注释和( )。

A、程序的布局组织

B、尽量不使用或少用goto语句

C、检查输入数据的有效性

D、设计良好的输出报表

12. (单选题, 2分) 编制一个好的程序应强调良好的编程风格,例如,选择标识符的名字时应考虑( )。

A、名字长度越短越好,以减少源程序的输入量

B、多个变量共用一个名字,以减少变量名的数目

C、选择含义明确的名字,以正确提示所代表的实体

D、尽量用关键字作名字，以使名字标准化

13. (单选题, 2分) 以下关于编程风格的叙述中，**不应**提倡的是( )。

A、使用括号以改善表达式的清晰性

B、用计数方法而不是用文件结束符判断输入的结束

C、一般情况下,不要直接进行浮点数的相等比较

D、使用有清晰含义的标识符

【解析】用计数方法而不是用文件结束符判断输入的结束：这不是一个应提倡的做法。在处理文件输入时，通常应该使用文件结束符（如EOF）来判断输入的结束，因为计数方法可能会在文件内容变化时导致错误。依赖于计数可能会导致程序在文件内容意外变化时无法正确处理输入结束的情况。

14. (单选题, 2分) 为了提高易读性，源程序内部应加功能性注释，用于说明( )。

A、模块总的功能

B、程序段或语句的功能

C、模块参数的用途

D、数据的用途

【解析】功能性注释嵌在源程序体中，功能性注释用以描述其后的语句或程序段的功能，以及数据的状态。

15. (单选题, 2分) 功能性注释的主要内容不包括( )。

A、程序段的功能

B、模块的功能

C、数据的状态

D、语句的功能

【解析】功能性注释嵌在源程序体中，功能性注释用以描述其后的语句或程序段的功能，以及数据的状态。

16. (单选题, 2分) 在结构化程序设计思想提出之前,在程序设计中曾强调程序的效率,现在人们更重视程序的( )性。

A、技巧性

B、保密性

C、一致性

D、可理解性

17. (单选题, 2分) 下面的（ ）不是良好编码的原则。

A、在开始编码之前建立单元测试

B、建立一种有助于理解的直观布局

C、确保注释与代码完全一致

D、保持变量名简短以便代码紧凑

【解析】A、在开始编码之前建立单元测试：这是良好编码实践的一部分，因为它有助于确保代码的质量和可维护性。

B、建立一种有助于理解的直观布局：这是良好编码原则的一部分，因为它提高了代码的可读性和可维护性。

C、确保注释与代码完全一致：这是良好编码原则的一部分，因为不一致的注释会导致误解和错误。

1. 保持变量名简短以便代码紧凑：这不是良好编码原则。虽然代码的简洁性是一个考虑因素，但变量名应该足够描述性，以清晰地传达变量的用途和含义。过于简短的变量名可能会导致代码难以理解，从而降低代码的可读性和可维护性。

18. (单选题, 2分) 下面的（ ）是错误的。

A、在程序设计中使用括号以改善表达式的清晰性

B、不要修补不好的程序，要重新写

C、在程序设计中应尽可能对程序代码进行优化

D、不要在注释中重复描述代码

【解析】A、在程序设计中使用括号以改善表达式的清晰性：这是正确的，使用括号可以帮助明确表达式的计算顺序，提高代码的可读性。

B、不要修补不好的程序，要重新写：这是一种观点，它强调了重构代码以提高质量和可维护性的重要性。虽然这并不总是适用（有时候修补是必要的），但这个观点鼓励开发者考虑代码的长期健康。

C、在程序设计中应尽可能对程序代码进行优化：这是错误的。过度优化可能会导致代码变得复杂和难以维护。通常推荐的做法是首先编写清晰、可读的代码，然后根据性能测试的结果进行必要的优化。所谓的“过早优化是万恶之源”（premature optimization is the root of all evil）就是指在没有明确需要的情况下进行过度优化。

D、不要在注释中重复描述代码：这是正确的，注释应该提供代码本身没有明确表达的信息，如代码的意图、背景或复杂逻辑的解释，而不是简单地重复代码已经表达的内容。

19. (单选题, 2分) 为了保证软件的质量，使其具有较好的可维护性，关键在于（ ）。

A、选择合适的程序设计语言

B、选择好的程序设计风格

C、具有好的数据结构

D、选择好的运行环境

【解析】B、选择好的程序设计风格：程序设计风格直接影响代码的可读性、可维护性和可扩展性。良好的编程风格可以使得代码更加清晰、易于理解和修改，这对于软件的长期维护至关重要。

20. (单选题, 2分) 下面的（ ）是对提高程序编码效率没有影响的。

A、变量名的使用

B、选择良好的设计方法

C、选择良好的算法

D、选择良好的数据结构

21. (单选题, 2分) 下面的（ ）**不是**一种好的做法。

A、好的注释应解释为什么，而不是怎么样。

B、好的命名应一目了然，不需要读者去猜，甚至不需要注释。

C、如果项目中原有代码不符合新的规范，应允许其存在，同时在新的代码中要延续原有的风格。

D、如果项目中原有代码不符合新的规范，应允许其存在，但不应在新的代码中延续旧的风格。

【解析】C、如果项目中原有代码不符合新的规范，应允许其存在，同时在新的代码中要延续原有的风格：这不是一种好的做法。如果项目中存在不符合新规范的代码，应该考虑重构这些代码以符合新的规范，而不是在新的代码中延续旧的风格。这样可以保持代码风格的一致性，提高代码的可读性和可维护性。

22. (单选题, 2分) 下面的（ ）说法是错误的。

A、代码审查用于检查源代码是否达到模块设计的要求

B、代码在审查之前必须要成功地编译通过

C、代码审查比运行程序进行测试的效率低

D、代码审查可以发现不符合团队代码规范的地方

【解析】C、代码审查比运行程序进行测试的效率低：这是错误的。代码审查和运行程序测试是两种不同的质量保证活动，它们各自有不同的目的和优势。代码审查侧重于代码质量和遵循编码标准，而测试侧重于验证代码的功能和性能。代码审查可以发现一些测试可能无法发现的问题，如编码错误、设计问题和潜在的bug。因此，不能简单地比较两者的效率。

23. (单选题, 2分) 下面的（ ）语句风格是最不利于维护的。

A、return s['name'] if s['age'] >= 18 else s['nickname'] if s['age'] > 14 else 'anonymous'

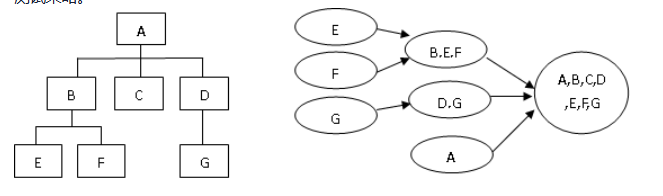
B、main(sys.argv[1:])

C、from my\_module import (Class1, Class2, Class3, Class4)

D、a, b = b, a

【解析】A这种嵌套的三元表达式虽然在某些情况下可以提高代码的简洁性，但它们会降低代码的可读性，特别是在没有注释说明每个条件的具体含义时。对于维护人员来说，这种风格的代码可能难以理解，尤其是当条件逻辑变得复杂时。

24. (单选题, 2分) 下图（左）所示为一个模块层次结构的例子，下图（右）所示为对其进行集成测试的顺序，则此测试采用了 （ ） 测试策略。



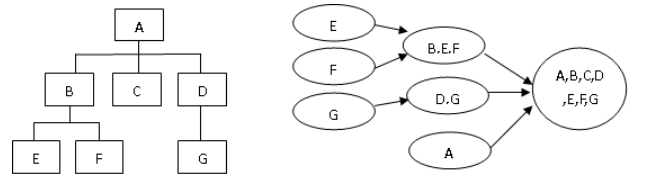
A、自底向上

B、自顶向下

C、三明治

D、一次性

25. (单选题, 2分) 下图（左）所示为一个模块层次结构的例子，下图（右）所示为对其进行集成测试的顺序，此测试策略的优点不包括 （ ） 。



A、不需要太多的驱动程序

B、集成之前彻底地测试了单独的组件

C、只需要较少的存根程序

D、在测试的最开始就可以对控制和公用程序进行测试

【解析】三明治集成得缺点是在集成之前没有彻底地测试单独的组件

二. 判断题（共15题，30分）

26. (判断题, 2分) 测试就是不断寻找程序中的漏洞直到时间耗尽为止。

A、对

B、错

27. (判断题, 2分) 除非对效率有特殊的要求，程序编写要做到清晰第一，效率第二。

A、对

B、错

28. (判断题, 2分) 测试分析报告应把每个模块实际测试的结果，与软件需求规格说明书和概要设计说明书中规定的要求进行对照并作出结论。

A、对

B、错

【解析】测试分析报告应该基于全面的测试结果来评估软件质量，而不仅仅是与需求和设计的简单对照

29. (判断题, 2分) 用计数方法而不是用文件结束符或输入序列结束符来判别输入的结束，这样做是一种良好的程序设计风格。

A、对

B、错

【解析】依赖于文件结束符或输入序列结束符来判断输入的结束是一种更为可靠和通用的方法，也是良好的程序设计风格的一部分。

30. (判断题, 2分) 软件测试是软件质量保证的唯一手段。

A、对

B、错

【解析】软件测试是软件质量保证的重要手段之一，但不是唯一的手段。

31. (判断题, 2分) 白盒测试又叫做功能测试或数据驱动测试，黑盒测试又称为结构测试或逻辑驱动测试。

A、对

B、错

【解析】白盒测试（也称为结构测试或逻辑驱动测试）是指测试人员需要了解程序内部逻辑结构和内部特性，通过测试来检测代码中的分支、路径等是否按照设计被执行。

黑盒测试（也称为功能测试）是指测试人员不需要了解程序内部逻辑，只依据程序的功能规格说明书，检查程序功能是否按照需求正常工作。

32. (判断题, 2分) 根据经验：发现错误多的程序模块，残留在模块中的错误也多。

A、对

B、错

33. (判断题, 2分) 软件测试小组应当为保证软件质量负全部责任。

A、对

B、错

【解析】软件测试小组在软件质量保证中扮演着重要的角色，但不能说他们应当为保证软件质量负全部责任。软件质量保证是一个涉及整个软件开发生命周期的过程，需要多个团队和个人共同协作来实现。

34. (判断题, 2分) 集成测试是测试和组装软件的系统化技术，进行集成测试时可以采用非渐增式测试和渐增式测试，其中渐增式测试实质上是同时完成单元测试和集成测试。

A、对

B、错

35. (判断题, 2分) 采用的编码方法和语言在表达和解决问题方面效率越高，用代码行方法计算的生产率就越高。

A、对

B、错

【解析】生产率的计算不仅仅取决于编码方法和语言的效率。代码行（Lines of Code, LOC）方法是一种过时的度量软件生产率的方式，它简单地以编写的代码行数来衡量生产率。

36. (判断题, 2分) 软件测试小组应当为保证软件质量负全部责任。

A、对

B、错

37. (判断题, 2分) 软件测试是软件质量保证的唯一手段。

A、对

B、错

38. (判断题, 2分) 高级语言表达和解决问题方面效率越高，采用代码行方法计算的生产率也越高。

A、对

B、错

39. (判断题, 2分) 软件测试的目的是尽可能地发现程序中的错误和缺陷，详细严密的测试过程可以证明软件的正确性。

A、对

B、错

【解析】软件测试的目的是尽可能地发现程序中的错误和缺陷，这是正确的。然而，即使进行了详细严密的测试过程，也不能证明软件是完全正确的。

40. (判断题, 2分) 软件测试不仅能表明软件中存在错误，也能说明软件中不存在错误。

A、对

B、错

三. 简答题（共3题，20分）

41. (简答题, 5分)

软件测试的目的是（1）。通常（2）是在代码编写阶段可进行的测试，它是整个测试工作的基础。检验软件是否满足用户需求的测试称为（3）。（4）是维护中常用的方法，其目的是检验修改所引起的副作用。黑盒测试法主要根据（5）来设计测试用例。

1．供选择的答案：

A. 表明软件的正确性 B. 评价软件质量

C. 尽可能发现软件中的错误 D. 判定软件是否合格

2．供选择的答案：

A. 系统测试 B. 安装测试 C. 验收测试 D. 单元测试

3．供选择的答案：

A. 确认测试 B. 有效性测试 C. 系统测试 D. 集成测试

4．供选择的答案：

A. 回归测试 B. 模块测试 C. 路径测试 D. 结构测试

5．供选择的答案：

A. 程序模块结构 B. 程序流程图 C. 程序内部逻辑 D. 程序外部功能

42. (简答题, 10分) 简述软件自顶向下集成测试的过程，说明这种集成的优缺点。

**正确答案：**

**从主控制模块开始，沿着程序的控制层次向下移动，逐渐把各个模块结合起来。在把附属于（及最终附属于）主控制模块的那些模块组装到程序结构中去时，或者使用深度优先的策略，或者使用宽度优先的策略。**

**自顶向下集成**

**优点：**

**（1）自顶向下的结合策略能够在测试的早期对主要的控制或关键的抉择进行检验。**

**（2）自顶向下测试不需要驱动程序。**

**（3）如果选择深度优先的结合方法，可以在早期实现软件的一个完整的功能并且验证这个功能。**

**缺点：**

**（1）需要编写存根程序。**

**（2）为了充分地测试软件系统的较高层次，需要在较低层次上的处理。然而测试初期自下往上的重要数据流都缺失了。**

**（3）存根程序需要的数量可能非常大。**

43. (简答题, 5分)

软件测试的目的是（1）。为了提高测试的效率，应该（2）。使用白盒测试方法时，确定测试数据应根据（3）和指定的覆盖标准。与设计测试数据无关的文档是（4）。软件的集成测试工作最好由（5）承担，以提高集成测试的效果。

1供选择的答案：

A. 评价软件的质量 B. 发现软件的错误

C. 找出软件中的所有错误 D. 证明软件是正确的

2供选择的答案：

A. 随机地选取测试数据 B. 取一切可能的输入数据作为测试数据

C. 在完成编码以后制定软件的测试计划 D. 选择发现错误的可能性大的数据作为测试数据

3供选择的答案：

A. 程序的内部逻辑 B. 程序的复杂程度

C. 使用说明书 D. 程序的功能

4供选择的答案：

A. 软件总体设计文档 B. 需求规格说明 C. 源程序 D. 项目开发计划

5供选择的答案：

A. 该软件的设计人员 B. 该软件开发组的负责人

C. 该软件的编程人员 D. 不属于该软件开发组的软件设计人员