答：

C++技术管理规范：

a.强制：

1. 文件命名：所有的源文件和头文件都必须使用小写字母，每个单词之间使用下划线进行分隔，例如example\_1.cpp和example\_2.h。

2. 文件编码：所有C++文件必须使用UTF-8编码。

3. 头文件保护：头文件必须使用宏保护，防止重复包含。

4. 模块划分：每个模块的功能必须清晰，避免模块之间过度耦合。

5. 缩进：代码必须使用4个空格进行缩进，严禁使用制表符。

6. 行长度：每行代码的最大长度不得超过80个字符。

7. 空格：在每个逗号、分号、冒号后面都必须要有一个空格，同时在加减等运算符两边都必须要有一个空格。

8. 括号：括号内不允许有空格，例如function(arg)。

9. 函数定义：函数名必须使用大小写混合，即驼峰命名法，例如 ExampleFunction()。

10. 类定义：类名必须使用大小写混合，即驼峰命名法，例如ExampleClass。

11. 变量命名：变量名必须使用小写字母，单词之间用下划线分隔，例如example\_variable。

12. 常量命名：常量名必须以k开头，后续单词首字母大写，例如kExampleConstant。

13. 枚举命名：枚举类型名需要使用大小写混合，即驼峰命名法，枚举值使用第12条常量命名的规则。

14. 宏命名：宏定义必须全部大写，单词之间用下划线分隔，例如MY\_MACRO。

15. 异常处理：必须捕获所有可能的异常，并提供有意义的错误信息。

16. 输入验证：所有的函数和方法都必须对输入各个参数进行验证，验证他们的类型和值是否正确。

17. 返回值：函数和方法必须有明确的返回值，严禁返回未定义的值。

18. 循环控制：循环中必须避免无限循环，必须有明确的退出条件。

19. 全局变量：严禁使用全局变量，除非有明确的必要性。

20. 指针和引用：指针和引用必须进行初始化，避免使用野指针。

21. 注释语言：注释必须使用英语或中文，语言清晰、简洁。

22. 函数注释：程序员必须对每个函数写注释，描述函数的功能、参数、返回值和可能的异常。

23. 复杂逻辑注释：对于拥有复杂逻辑的项目，相关程序员必须在对应代码旁边添加注释包括目的和实现方式。

24. 注释更新：当代码修改时，必须同步更新相关注释。

25. 测试覆盖率：单元测试的覆盖率必须达到80%以上。

26. 测试用例：每个函数和方法都必须拥有对应的测试用例，每个测试用例都必须覆盖所有可能的输入和边界情况。

27. 测试独立性：每个测试用例都必须相互独立，如果一个测试用例失败，其他测试用例依然需要能够正确执行。

b.推荐：

28. 模块划分：为了避免每个模块过大或过小，需要尽量将拥有相关联功能的代码放在同一个模块之中。

29. 包命名：每个包的命名都需要尽量使用简洁直观的名字，从而避免它的名字过于复杂。

30. 空行：建议在一个函数和一个类之间使用两个空行以间隔，同理在两个不同方法之间使用一个空行以间隔。

31. 代码块：尽量使用with语句管理文件和资源，避免手动关闭文件或释放资源。

32. 字符串格式化：推荐使用标准库中的字符串格式化函数。

33. 函数长度：尽量将函数长度控制在20行以内，避免函数过于复杂。

34. 类职责：类的职责尽量单一，避免一个类承担过多的功能。

35. 代码复用：尽量复用已有的代码，避免重复编写相似的代码。

36. 注释比例：注释的比例建议在20% - 30%之间，避免注释过多或过少。

37. 注释格式：当程序员编写注释时，需要尽量使用块注释或行注释，避免因注释格式分别不清带来的误解。

38. 测试框架：推荐使用Google Test或其他成熟的测试框架。

39. 测试数据：测试数据尽量使用模拟数据，避免依赖外部数据源。

40. 测试环境：测试环境需要尽量与项目实际的使用环境一致，避免因环境存在差异而导致的测试结果不可靠。

c.允许

41. 临时文件：开发时允许项目成员使用临时文件进行调试，但必须在代码提交前删除。

42. 临时变量：调试时允许项目成员使用临时变量进行调试，但必须在代码提交前删除。

43. 特殊场景测试用例：在某些特殊场景下允许不进行测试用例的编写，但必须在代码提交前补上。