# 实验报告大纲

目录

[实验报告大纲 1](#_Toc191372496)

[实验报告详细内容 2](#_Toc191372497)

[0.创建几个prompt，用于编程、操作系统、数据结构等课程的学习提问。 2](#_Toc191372498)

[1.引言 3](#_Toc191372499)

[ 1.1实验目的： 3](#_Toc191372500)

[ 1.2实验背景： 3](#_Toc191372501)

[2.实验环境 3](#_Toc191372502)

[ 2.1所用工具和平台： 3](#_Toc191372503)

[3.实验内容 3](#_Toc191372504)

[ 3.1创建的prompt示例： 3](#_Toc191372505)

[ 3.2选择的课程和问题 3](#_Toc191372506)

[4.R版大模型与普通大模型比较 3](#_Toc191372507)

[5.分析与结论 10](#_Toc191372508)

# 实验报告详细内容

## 0.创建几个prompt，用于编程、操作系统、数据结构等课程的学习提问。

答：

编程

1. 解释什么是递归函数，并提供一个递归函数的例子。
2. 请描述JavaScript中的闭包是什么，并给出一个实际例子。
3. 对比同步和异步编程的区别，并举例说明。
4. 什么是面向对象编程？请解释其核心概念，并提供一个简单的面向对象编程示例。
5. 请解释什么是设计模式，并给出一个常见的设计模式及其应用场景。

操作系统

1. 解释进程和线程的区别，并举例说明。
2. 什么是死锁？请描述死锁的四个必要条件，并给出一个避免死锁的方法。
3. 解释内存管理中的分页和分段技术，并比较它们的优缺点。
4. 什么是进程调度？请描述几种常见的进程调度算法及其适用场景。
5. 解释文件系统的层次结构，并描述文件系统的常见操作（如创建、删除、读取、写入）。

数据结构

1. 解释什么是二叉树，并描述其基本操作（如插入、删除、查找）。
2. 什么是哈希表？请描述哈希表的工作原理，并给出一个哈希冲突解决方法。
3. 解释图的表示方法（邻接矩阵和邻接表），并描述它们的优缺点。
4. 什么是堆？请描述堆的性质，并给出一个堆排序的实现。

## 1.引言

* 1.1实验目的：通过对比R版大模型和普通大模型在编程课程中的表现，了解两者在生成复杂推理内容方面的差异。
* 1.2实验背景：随着大模型技术的发展，R版大模型在复杂推理任务中的表现越来越受到关注。本实验旨在对比R版大模型和普通大模型在编程课程中的表现。

## 2.实验环境

* 2.1所用工具和平台：DeepSeek-R1、DeepSeek-V3、Python

## 3.实验内容

* 3.1创建的prompt示例：
  + 解释什么是递归函数，并提供一个递归函数的例子。
  + 请描述JavaScript中的闭包是什么，并给出一个实际例子。
  + 对比同步和异步编程的区别，并举例说明。
  + 什么是面向对象编程？请解释其核心概念，并提供一个简单的面向对象编程示例。
  + 请解释什么是设计模式，并给出一个常见的设计模式及其应用场景。
* 3.2选择的课程和问题：编程课程，选择上述5个问题进行比较。

## 4.R版大模型与普通大模型比较

**问题1：解释什么是递归函数，并提供一个递归函数的例子。**

* **普通大模型（DeepSeek-V3）**：
* 递归函数是指一个函数在其定义中直接或间接调用自身的函数。递归函数通常用于解决可以被分解为更小的相同问题的任务。例如，计算阶乘的递归函数如下：
* ```python
* def factorial(n):
* if n == 0:
* return 1
* else:
* return n \* factorial(n - 1)
* **R版大模型（DeepSeek-R1）**：
* 递归函数是指一个函数在其定义中直接或间接调用自身的函数。递归函数通常用于解决可以被分解为更小的相同问题的任务。递归函数的关键在于有一个基准条件（base case）来终止递归调用，否则会导致无限递归。例如，计算阶乘的递归函数如下：
* ```python
* def factorial(n):
* if n == 0: # 基准条件
* return 1
* else:
* return n \* factorial(n - 1) # 递归调用

递归函数的优点是可以使代码更简洁和易于理解，但缺点是可能导致较高的时间和空间复杂度。

* **分析**：R版大模型在回答递归函数时提供了更详细的解释，包括基准条件的重要性，而普通大模型的回答较为简略。

**问题2：请描述JavaScript中的闭包是什么，并给出一个实际例子。**

* **普通大模型（DeepSeek-V3）**：
* 闭包是指一个函数能够记住并访问它的词法作用域，即使这个函数在其词法作用域之外执行。例如：
* ```javascript
* function createCounter() {
* let count = 0;
* return function() {
* count++;
* return count;
* };
* }
* const counter = createCounter();
* console.log(counter()); // 输出 1
* console.log(counter()); // 输出 2
* **R版大模型（DeepSeek-R1）**：
* 闭包是指一个函数能够记住并访问它的词法作用域，即使这个函数在其词法作用域之外执行。闭包的核心在于它能够捕获并保存外部函数的作用域变量。例如：
* ```javascript
* function createCounter() {
* let count = 0; // 外部函数的作用域变量
* return function() {
* count++; // 访问外部函数的作用域变量
* return count;
* };
* }
* const counter = createCounter();
* console.log(counter()); // 输出 1
* console.log(counter()); // 输出 2

闭包在JavaScript中非常有用，可以用于创建私有变量、实现工厂函数等。然而，过度使用闭包可能会导致内存泄漏。

* **分析**：R版大模型在回答闭包时提供了更深入的解释，包括闭包的核心概念和应用场景，而普通大模型的回答较为基础。

**问题3：对比同步和异步编程的区别，并举例说明。**

* **普通大模型（DeepSeek-V3）**：
* 同步编程是指程序按顺序执行，每个语句必须等待前一个语句执行完毕才能执行。异步编程是指程序可以在等待某些操作完成的同时继续执行其他操作。例如，在JavaScript中：
* ```javascript
* // 同步示例
* console.log('Start');
* for (let i = 0; i < 1000000000; i++) {}
* console.log('End');
* // 异步示例
* console.log('Start');
* setTimeout(() => {
* console.log('End');
* }, 1000);
* console.log('Doing something else');
* **R版大模型（DeepSeek-R1）**：
* 同步编程是指程序按顺序执行，每个语句必须等待前一个语句执行完毕才能执行。同步编程的优点是代码逻辑简单，易于理解和调试，但缺点是当遇到耗时操作时会阻塞整个程序。异步编程是指程序可以在等待某些操作完成的同时继续执行其他操作。异步编程的优点是可以提高程序的性能和响应速度，但缺点是代码逻辑相对复杂，容易出现回调地狱等问题。例如，在JavaScript中：
* ```javascript
* // 同步示例
* console.log('Start');
* for (let i = 0; i < 1000000000; i++) {} // 阻塞操作
* console.log('End');
* // 异步示例
* console.log('Start');
* setTimeout(() => {
* console.log('End'); // 1秒后执行
* }, 1000);
* console.log('Doing something else'); // 立即执行

在异步编程中，可以使用Promise、async/await等机制来简化代码逻辑。

* **分析**：R版大模型在回答同步和异步编程时提供了更全面的分析，包括各自的优缺点和应用场景，而普通大模型的回答较为基础。

**问题4：什么是面向对象编程？请解释其核心概念，并提供一个简单的面向对象编程示例。**

* **普通大模型（DeepSeek-V3）**：
* 面向对象编程（OOP）是一种编程范式，通过对象和类来组织代码。OOP的核心概念包括类、对象、继承、封装和多态。例如，在Python中：
* ```python
* class Dog:
* def \_\_init\_\_(self, name):
* self.name = name
* def bark(self):
* print(f"{self.name} says woof!")
* my\_dog = Dog("Buddy")
* my\_dog.bark()
* **R版大模型（DeepSeek-R1）**：
* 面向对象编程（OOP）是一种编程范式，通过对象和类来组织代码。OOP的核心概念包括：
* - 类（Class）：定义对象的蓝图。
* - 对象（Object）：类的实例。
* - 继承（Inheritance）：子类可以继承父类的属性和方法。
* - 封装（Encapsulation）：将数据和操作数据的方法封装在一起，隐藏内部实现细节。
* - 多态（Polymorphism）：不同的对象可以对同一消息作出不同的响应。
* 例如，在Python中：
* ```python
* class Animal:
* def speak(self):
* pass
* class Dog(Animal):
* def \_\_init\_\_(self, name):
* self.name = name
* def speak(self):
* return f"{self.name} says woof!"
* class Cat(Animal):
* def \_\_init\_\_(self, name):
* self.name = name
* def speak(self):
* return f"{self.name} says meow!"
* my\_dog = Dog("Buddy")
* my\_cat = Cat("Whiskers")
* print(my\_dog.speak()) # 输出 Buddy says woof!
* print(my\_cat.speak()) # 输出 Whiskers says meow!

OOP的优点是可以提高代码的可重用性和可维护性，但缺点是增加了代码的复杂性。

* **分析**：R版大模型在回答面向对象编程时提供了更详细的解释，包括各个核心概念的具体含义和应用场景，而普通大模型的回答较为简略。

**问题5：请解释什么是设计模式，并给出一个常见的设计模式及其应用场景。**

* **普通大模型（DeepSeek-V3）**：
* 设计模式是一套被反复使用的、经过分类的、代码设计经验的总结。常见的设计模式包括单例模式、工厂模式、观察者模式等。例如，单例模式确保一个类只有一个实例，并提供一个全局访问点。例如，在Python中：
* ```python
* class Singleton:
* \_instance = None
* def \_\_new\_\_(cls):
* if cls.\_instance is None:
* cls.\_instance = super(Singleton, cls).\_\_new\_\_(cls)
* return cls.\_instance
* s1 = Singleton()
* s2 = Singleton()
* print(s1 is s2) # 输出 True
* **R版大模型（DeepSeek-R1）**：
* 设计模式是一套被反复使用的、经过分类的、代码设计经验的总结。设计模式旨在解决特定的问题，提高代码的可重用性和可维护性。常见的设计模式包括单例模式、工厂模式、观察者模式等。例如，单例模式确保一个类只有一个实例，并提供一个全局访问点。单例模式的应用场景包括数据库连接池、日志记录器等。例如，在Python中：
* ```python
* class Singleton:
* \_instance = None
* def \_\_new\_\_(cls):
* if cls.\_instance is None:
* cls.\_instance = super(Singleton, cls).\_\_new\_\_(cls)
* return cls.\_instance
* s1 = Singleton()
* s2 = Singleton()
* print(s1 is s2) # 输出 True

单例模式的优点是可以减少资源消耗，确保系统中只有一个实例。然而，过度使用单例模式可能会导致代码难以测试和扩展。

* **分析**：R版大模型在回答设计模式时提供了更详细的解释，包括应用场景和潜在的问题，而普通大模型的回答较为基础。

## 5.分析与结论

* **对比结果的分析**：从上述五个问题的比较可以看出，R版大模型在生成复杂推理内容时提供了更详细和全面的解释，而普通大模型的回答较为简略。R版大模型在处理需要推理和解释的问题时表现更好。
* **实验总结**：通过本次实验，我们发现R版大模型在处理复杂推理任务时具有明显的优势。这表明R版大模型在教育和学习场景中具有更大的潜力。