2.1 基础理论与概念——编程时光之旅

开发要求：

- 创建一个时间线，展示C++的历史背景与演变
- 编写简单代码，对比C++与其他编程语言的语法

开发思路：

项目名称：编程时光之旅

引入环节：通过提问方式询问学生对编程的了解，并简要介绍各种编程语言的起源与发展。展示一些C++的趣味事实，引发学生的好奇心。

2.1.1. 制作C++历史时间线

- **目标**：通过制作一个详细的时间线，帮助学生们深入理解C++的发展历程和重要里程碑。
- **互动方式**：学生们需要研究C++的发展历史，包括其起源、主要版本的发布日期和每个版本的新特性等，然后将这些信息整理成一个清晰、易读的时间线。
- **展示**：教师可以提供一个C++历史时间线的示例，解释每个重要事件的背景和影响，以此来启发学生们如何制作自己的时间线。

2.1.2. 对比C++与其他编程语言

- **目标**：通过编写和对比代码，让学生们了解C++与其他编程语言（如Python或Java）在语法和设计理念上的差异。
- **互动方式**：学生们需要用C++和另一种编程语言编写同一个程序（如一个简单的“Hello, World!”程序），然后对比两种语言在语法结构、代码长度、执行效率等方面的差异。
- **展示**：教师可以先用C++和另一种语言编写同一个程序，然后详细解释两种语言的语法差异和设计理念，以此来引导学生们进行深入的对比和思考。

游戏化和互动元素：

- **编程时光机**：让学生们在时间线上添加自己最喜欢的编程语言，并分享他们选择这种语言的原因，以及他们使用这种语言的经验。
- **语法对比**：让学生们选择两种他们熟悉的编程语言，比较他们的语法差异，并分享他们对这些差异的看法和理解。
- **编程语言的未来**：让学生们想象一种未来的编程语言，描述它的特性和用途，并解释为什么他们认为这种语言会有用。

小结

1. 本节课我们通过创建一个时间线，深入了解了C++的历史背景与演变。
2. 我们研究了C++的历史，展示了C++的主要发展阶段和重要事件。
3. 通过编写简单的代码，我们理解了C++与其他编程语言的语法差异。
4. 我们编写了一段简单的C++代码，然后再用其他编程语言（如Python或Java）编写同样的功能，比较了两者的语法差异。
5. 在问答环节，提问关于C++的历史背景与演变，以及C++与其他编程语言的语法差异的问题。
6. 回顾在本节课中编写的代码，强调了C++的语法特性，以及与其他编程语言的差异。

2.2 基础理论与概念——简易计算器

开发要求：

- 设计一个支持基本算术运算的计算器
- 加入修饰符选项，以展示其效果
- 对比C与C++的计算器实现

开发思路：

项目名称：简易计算器

引入环节：展示一个现实生活中的场景，如在商店购物时需要计算总价，以此强调计算器工具的重要性。

2.2.1. 设计一个支持基本算术运算的计算器

- **目标**：通过设计一个支持基本算术运算的计算器，让学生们了解如何使用C++实现基本的算术运算。
- **互动方式**：让学生们设计并编写一个简单的计算器程序，支持加、减、乘、除等基本算术运算。
- **展示**：教师可以先演示一个已经完成的计算器程序，然后解释每个部分的功能，以此来引导学生们设计并编写自己的计算器程序。

2.2.2. 加入修饰符选项，以展示其效果

- **目标**：通过加入修饰符选项，让学生们了解C++中修饰符的使用和效果。
- **互动方式**：让学生们在计算器程序中加入修饰符选项，如const、volatile等，然后观察并分析其效果。
- **展示**：教师可以先演示如何在计算器程序中加入修饰符选项，然后解释其效果，以此来引导学生们理解和使用C++中的修饰符。

2.2.3. 对比C与C++的计算器实现

- **目标**：通过对比C与C++的计算器实现，让学生们了解两种编程语言在实现同一功能时的差异和优劣。
- **互动方式**：让学生们先用C语言实现一个简单的计算器程序，然后再用C++实现同样的功能，最后对比两者的实现方式和效果。
- **展示**：教师可以先演示如何用C语言和C++分别实现一个简单的计算器程序，然后对比两者的实现方式和效果，以此来引导学生们理解和比较C与C++的差异和优劣。

游戏化和互动元素：

- **计算器设计**：让学生们设计一个支持基本算术运算的计算器，以此来展示他们对C++的理解和应用。
- **修饰符选项**：让学生们在计算器程序中加入修饰符选项，如const、volatile等，以此来展示他们对C++修饰符的理解和应用。
- **C与C++的对比**：让学生们用C语言和C++分别实现一个简单的计算器程序，以此来展示他们对C与C++的理解和应用。

小结

- **问答环节**：在本节课结束时，进行一个问答环节，让学生们回答关于如何设计一个支持基本算术运算的计算器，如何使用C++中的修饰符，以及如何对比C与C++的计算器实现的问题。
- **代码回顾**：回顾并讨论在本节课中编写的计算器程序，强调C++的语法特性，以及与C语言的差异。
- **反馈与表扬**：对学生们在理解和应用C++的知识，以及在设计和编写计算器程序方面的努力给予反馈和表扬，鼓励他们在未来的编程学习中继续使用这些知识。
- **生活实际**：强调计算器在我们日常生活中的应用，如在商店购物时计算总价，或者在做家庭预算时计算收支，以此来强调编程学习的实用性。

2.3 基础理论与概念——编码解密机

开发要求：

- 使用各种运算符制作一个简单的编码器和解码器
- 通过赋值运算实现简易的字符替换功能
- 实现一个使用条件运算符的加密逻辑
- 利用位运算为加密增加复杂性

开发思路：

项目名称：编码解密机

引入环节：讲述一个简单的故事，如一位间谍需要发送加密消息以避免被敌方截获，以此突出加密的重要性。

2.3.1 使用各种运算符制作一个简单的编码器和解码器

- **目标**：让学生们掌握如何使用C++中的运算符来实现基本的编码和解码操作。这将涉及到字符与ASCII码之间的转换，以及使用运算符进行基本的数学运算。
- **互动方式**：学生们将被要求使用C++中的运算符（如加法、减法、异或等）来制作一个简单的编码器和解码器。这个编码器和解码器应该能够接受一串字符，然后对这些字符进行编码，再将编码后的字符解码回原始字符。
- **展示**：教师将首先演示一个简单的编码器和解码器的实现，解释如何使用C++中的运算符进行编码和解码操作。然后，教师将解释每个运算符的作用，以及它们在编码和解码过程中的应用。

2.3.2. 通过赋值运算实现简易的字符替换功能

- **目标**：让学生们掌握如何使用C++中的赋值运算符来实现字符的替换，这是实现编码和解码操作的基础。
- **互动方式**：学生们将被要求使用C++中的赋值运算符来实现一个简单的字符替换功能。这个功能应该能够接受一串字符，然后将其中的某些字符替换为其他字符。
- **展示**：教师将首先演示如何使用C++中的赋值运算符来实现字符的替换，解释赋值运算符在字符替换过程中的作用。然后，教师将解释如何将这个字符替换功能应用到编码和解码操作中。

2.3.3. 探索条件运算符在加密逻辑中的应用

- **目标**：让学生们了解并掌握如何使用C++中的条件运算符来构建加密逻辑。这将涉及到条件判断，以及如何根据条件进行不同的操作。
- **互动方式**：学生们将被要求在编码器和解码器程序中实现一个使用条件运算符的加密逻辑。例如，当输入字符的ASCII码值为奇数时，加密方式与偶数时不同。

- **展示**：教师将首先演示一个使用条件运算符的加密逻辑的实现，解释如何根据不同的条件进行不同的加密操作。然后，教师将解释条件运算符在加密逻辑中的重要性。

2.3.4. 利用位运算增加加密的复杂性

- **目标**：让学生们了解并掌握如何使用C++中的位运算来增加加密的复杂性。这将涉及到二进制数的操作，以及如何使用位运算进行加密。
- **互动方式**：学生们将被要求在编码器和解码器程序中利用位运算来增加加密的复杂性。例如，可以通过位移运算来改变字符的ASCII码值，从而实现加密。
- **展示**：教师将首先演示如何使用位运算来增加加密的复杂性，解释如何通过位移运算来改变字符的ASCII码值。然后，教师将解释位运算在加密过程中的重要性。

游戏化和互动元素：

- **编码解密竞赛**：学生们可以分成两组，一组负责编写编码器，另一组负责编写解码器。然后，两组可以互相交换编码后的消息，看哪一组能最快解码出原始消息。
- **角色扮演**：学生们可以扮演间谍，使用他们自己编写的编码器和解码器发送和接收秘密消息。这不仅可以提高学生们对编码和解码的理解，也可以增加课程的趣味性。
- **解密挑战**：教师可以提供一些已经编码的消息，让学生们使用他们的解码器来解密。这可以帮助学生们检验他们的解码器是否正确，也可以提高他们解决问题的能力。
- **优化比赛**：学生们可以尝试优化他们的编码器和解码器，使其更加复杂和难以解密。然后，可以进行一个比赛，看哪个编码器的消息最难解密。

小结

- **问答环节**：在本节课结束时，进行一个问答环节，让学生们回答关于如何使用编码器和解码器，以及如何使用运算符进行编码和解码的问题。
- **代码回顾**：回顾并讨论在本节课中编写的编码器和解码器，强调运算符的使用方法，以及如何通过运算符实现编码和解码。
- **反馈与表扬**：对学生们在理解和应用运算符的知识，以及在编写编码器和解码器方面的努力给予反馈和表扬，鼓励他们在未来的编程学习中继续使用这些知识。
- **生活实际**：编码和解码在我们日常生活中无处不在。例如，当我们在网上购物时，我们的信用卡信息会被编码然后发送到服务器，服务器在接收到信息后会进行解码。这样可以保护我们的信息安全。又如，当我们在手机上查看网页时，网页的内容会被编码成数据包，然后通过网络发送到我们的手机，手机接收到数据包后会进行解码，然后显示网页的内容。这些都是编码和解码在日常生活中的应用，体现了编程学习的实用性。

2.4 核心程序设计——冒险游戏模拟

开发要求：

- 设计一个基于条件结构的文字冒险游戏
- 通过循环添加多个关卡
- 使用跳转结构为玩家提供捷径或隐藏路径

开发思路：

项目名称：冒险游戏模拟

引入环节：通过简短的故事或视频片段，展示一个冒险家遇到的选择题关卡，例如：“你面前有两条路，左边通往密林，右边通往山洞，你会选择哪条路？”以此激发学生对接下来的文字冒险游戏的兴趣。

2.4.1. 设计一个基于条件结构的文字冒险游戏

- **目标**：本节课的目标是让学生们理解和掌握如何使用C++中的 `if` 和 `switch-case` 的使用方法和应用场景。
- **互动方式**：教师将实时编写简单的冒险游戏代码，展示如何根据玩家的选择走不同的剧情线。
- **展示**：教师将展示一个简单的游戏示例，解释如何通过条件结构来实现游戏的逻辑。

2.4.2. 通过循环添加多个关卡

- **目标**：本节课的目标是让学生们理解和掌握C++中的 `for`、`while` 和 `do-while` 循环的基本概念和用法。
- **互动方式**：教师将在现有的游戏中添加多个关卡，每个关卡都有新的选择题，引导学生逐步构建完整的冒险故事。
- **展示**：教师将展示如何使用循环结构来添加多个关卡，解释循环结构的工作原理。

2.4.3. 使用跳转结构为玩家提供捷径或隐藏路径

- **目标**：本节课的目标是让学生们理解和掌握C++中的 `break`、`continue` 和 `goto` 的作用和用法。
- **互动方式**：教师将为游戏添加特殊的路径或捷径，如找到一个秘密通道可以跳过某个关卡。
- **展示**：教师将展示如何使用跳转结构来实现捷径或隐藏路径，解释跳转结构的工作原理。

游戏化和互动元素：

- **角色扮演**：学生们可以在游戏中扮演不同的角色，如冒险家、导游等，通过角色扮演，学生们可以更好地理解和体验游戏的故事情节。
- **故事情节**：学生们可以设计各种各样的故事情节，如寻找宝藏、解救公主等，通过故事情节，学生们可以更好地理解和体验游戏的冒险和挑战。
- **互动选择**：学生们可以在游戏中做出各种选择，如选择走哪条路、选择使用哪种工具等，通过互动选择，学生们可以更好地理解和体验游戏的策略和决策。

小结

- **问答环节**：在本节课结束时，进行一个问答环节，让学生们回答关于如何使用条件结构设计一个文字冒险游戏，如何在游戏中实现角色扮演和故事情节，以及如何通过互动选择和团队合作增加游戏的趣味性的问题。
- **代码回顾**：回顾并讨论在本节课中设计的文字冒险游戏，强调条件结构的使用方法，以及如何通过条件结构实现游戏的策略和决策。
- **反馈与表扬**：对学生们在理解和应用条件结构的知识，以及在设计和实现文字冒险游戏方面的努力给予反馈和表扬，鼓励他们在未来的编程学习中继续使用这些知识。

联系人管理系统

5.1. 设计一个可以添加、删除和搜索联系人的系统

5.2. 通过函数参数传递实现联系人信息的编辑功能

5.3. 利用重载函数增加多种搜索方式

5.4. 通过默认参数实现联系人的分类功能