|  |  |  |  |
| --- | --- | --- | --- |
| 第一课：编程时光之旅 | | 学时： |  |
| 主要内容 | 1. 通过提问方式询问学生对编程的基本了解程度，包括他们是否曾接触编程、对编程的兴趣程度等。逐步引导，介绍计算机语言的基本概念和作用，突出计算机语言在解决问题和实现功能上的作用。随后，简要介绍Python、C语言和Java的起源与发展，使学生对这三种语言有初步认识，为后续深入学习做铺垫。 2. 将C++中的STL（标准模板库）类比为日常生活中常见的事物，旨在激发学生对C++的求知欲。通过类比，解释STL为C++提供的丰富数据结构和算法，使学生能够将抽象的概念与实际应用场景联系起来。 3. 创建一条清晰的时间线，从C语言的起源开始，追溯到C++的引入，最终到现代C++的发展。通过关键事件和版本迭代，呈现C++的演变历程，强调C++在软件开发领域中的重要性和持久性。此外，为了更好地理解，将每个阶段的技术特点和应用场景详细说明，使学生对C++的发展有更全面的认知。 4. 通过简单的代码示例，对比Python和C++在性能、类型声明、语言风格、使用库和功能等方面的区别。强调C++相对于Python的优势，引导学生理解两者的差异，并为从熟悉的Python学习过渡到C++做好心理准备。 5. 通过实例演示，使用Python和C++分别解决鸡兔同笼问题，突出C++的不同书写特点。详细解析代码，强调C++的语法和编码方式，使学生能够初步了解C++的基本结构。   6.设计课后提问，涉及C++的结构和语法的正确书写，以促使学生深化对理论知识的理解。通过互动性的学习方式，鼓励学生积极参与讨论，分享彼此的思考和体会，巩固对C++编程语言的理论知识。提供详细的反馈和建议，引导学生更熟练地应用所学概念。 | | |
| 学习目标 | 1.在课程中，学生将通过详细的历史回顾，全面了解C语言、Python、Java和C在编程发展史上的地位和应用领域。通过研究它们的演变过程，学生将培养对编程语言演变的敏感性，了解每种语言在软件开发历史上的重要性。  2.学生将深入学习C++是如何从C语言演变而来的，理解C++在软件开发领域的重要性。通过探讨C++的先进特性和在现代软件工程中的广泛应用，学生将建立对C++在不同领域的实际应用的深刻认识。   1. 课程将系统性地介绍C++与其他主要编程语言（如Python和Java）的基础语法和特点。学生将深入了解语法结构、编程范式以及语言特性，使其能够在多语言环境下更灵活地应用编程知识。 2. 通过实际编程演示，学生将编写简单的代码示例，比较Python和C++在基础语法方面的异同。这将帮助学生更具体地理解变量声明、条件判断等方面的差异，并分析它们在不同应用场景中的优缺点。 3. 学生将通过编程实践培养逻辑思维和解决问题的能力。课程将强调通过编程理解和解决问题的方法，让学生能够将抽象的问题具体化，并通过编程实现有效的解决方案，提升他们的实际应用能力。   6.通过引入趣味事实和实际应用案例，课程将激发学生的好奇心和兴趣。这种学习方式将使学生更加积极主动地去深入了解和探索编程领域的知识，从而提高他们对编程的热情和主动性。 | | |
| 学生课前阅读材料与其他准备 | 1.推荐书目：  《计算机世界探险系列：C++》（作者：Judy and Brian Marshall）：这本书以图文结合的方式，适合小学生的阅读水平，简要介绍了C++的基本概念和历史。  2.思考问题：  1.你知道计算机是什么时候产生的吗？它最初是用来做什么的？可以去了解一下最早的计算机是如何工作的。  2.什么是第一代编程语言？你能想象一下人们是如何用它们编写程序的吗？为什么人们觉得需要更高级的编程语言？  3.你听说过C语言吗？它是什么时候产生的？有什么特点？为什么人们觉得需要一种新的编程语言？  4.你知道Bjarne Stroustrup吗？他是做什么的？为什么他决定创建一门新的编程语言，即C++？  5.你知道C++为什么叫这个名字吗？有什么特别的含义吗？C++的设计目标是什么？有哪些特性是为了解决什么问题而引入的？  5.C++是如何发展壮大的？有哪些重要的版本或标准？了解一下C++的标准化过程，为什么标准很重要？   1. 你知道C++在哪些领域被广泛应用吗？有没有听说过一些用C++编写的有名的软件或游戏？C++对计算机科学和软件开发有什么影响？ 2. 你认为未来C++会有什么样的发展？有没有你期待的新功能或应用？ 3. 编程对你来说是什么样的体验？你有兴趣学习更多关于计算机科学和编程的知识吗？ | | |
| 教学方式 | 1.交互式提问：利用提问引导学生主动参与，深入了解不同编程语言的背景、特性及应用场景。设计问题以促使学生思考，激发他们的好奇心和主动学习兴趣。  2.实时编写代码演示：在课堂上通过实际编写代码的方式，直观展示C++的编程过程，强调语法规范和良好的编码风格。通过实例演示，帮助学生理解C++的基本结构、变量、控制流等概念，培养实际编程能力。  3巩固性提问：在课后通过提问巩固学生的学习成果，检验他们对C++知识的掌握程度。问题设计旨在考察学生对课堂内容的理解，并激发他们进一步思考和探索。  4.PPT讲解与示范：利用专业设计的PPT，系统性地讲解C++的基本语法特点，包括变量、数据类型、控制结构等重要概念。通过示范具体案例，帮助学生理解抽象的概念，提供直观的实例以加深印象。 | | |