教　案

2025-2026 学年第二学期

课程名称	C 语言程序设计
专业班级	机电一体化技术（三二分段）251
总学时数	48 学时
任课教师	林耿萱

课程基本信息

<table>
<tr><td>课程名称</td><td colspan="4">C 语言程序设计</td></tr>
<tr><td>课程性质</td><td colspan="2">专业基础课</td><td>学分</td><td>3</td></tr>
<tr><td>学　时</td><td colspan="4">总学时：48 学时。其中：实训 48 学时。</td></tr>
<tr><td>开课部门</td><td colspan="2">机电工程系</td><td>任课教师</td><td>林耿萱</td></tr>
<tr><td>授课
专业、班级</td><td colspan="2">机电一体化技术（三二分段）251 班</td><td>开课学期</td><td>2025-2026 第二学期</td></tr>
<tr><td>成绩评定</td><td colspan="2">平时成绩占 60%；期末成绩占 40%</td><td>考核方式</td><td>考查</td></tr>
<tr><td rowspan="2">选用教材</td><td>书　　名</td><td>主 编</td><td>出版社</td><td>出版日期</td></tr>
<tr><td>C 程序设计（第四版）</td><td>谭浩强</td><td>清华大学出版社</td><td>2010 年</td></tr>
<tr><td>本课程在本专业人才培养方案中的地位和作用</td><td colspan="4">《C 语言程序设计》是全日制专科机电一体化技术专业等工科学生必修的专业基础课，也是培养学生处理解决问题思维的重要课程。</td></tr>
<tr><td>本课程
教学目标</td><td colspan="4">以阐明编程思想及方法的系统知识为主，通过对 C 语言的学习和使用，使学生深入理解面向过程的程序设计思想，掌握利用 C 语言编程思想及方法分析问题、解决问题的方法，为进一步学习单片机、嵌入式系统、高级程序设计等其他系列课程打下基础。</td></tr>
<tr><td>素质(思政)
内容与要求</td><td colspan="4">科技强国、创新驱动：结合我国在计算机科学和信息技术领域取得的成就，例如：超级计算机、人工智能、5G 通信等，增强学生的民族自豪感和科技自信。
严谨治学、精益求精：强调编写高质量代码的重要性，引导学生养成严谨治学的态度和精益求精的精神。
团队合作、责任担当：培养学生的团队合作精神和沟通协调能力，引导学生关注社会热点问题，利用所学知识解决实际问题，例如：开发疫情防控系统、设计智能垃圾分类系统等，增强学生的社会责任感和使命感。</td></tr>
<tr><td>学生用主要
参考资料</td><td colspan="4">课本、实验指导资料</td></tr>
</table>

第一章 程序设计与C语言（3课时）

教学目标：

了解编程的本质

了解C语言的发展历史、特点和应用领域。

掌握C语言程序的基本结构。

熟悉C语言的编译和运行过程。

能够编写简单的C语言程序。

教学方法、手段：

讲授、上机操作

教学重点：

C语言的特点和应用领域。

C语言程序的基本结构。

C语言的编译和运行过程。

教学难点：

理解C语言程序的基本结构。

掌握C语言的编译和运行过程。

课程思政：

科技强国、创新驱动

介绍C语言的发展历史和应用领域，强调C语言在计算机科学和信息技术领域的重要性，激发学生学习C语言的兴趣和热情。

结合我国在计算机科学和信息技术领域取得的成就，例如：超级计算机、人工智能、5G通信等，增强学生的民族自豪感和科技自信。

教学过程：

一、编程的本质

1．编写程序的载体，专业编程软件的特点

2．程序由函数和数据构成，数据的本质为保存在内存条或者硬盘上的电子，0和1的排列组合，编程的主要工作就是编写函数去处理数据，它的难度由数据量的大小及处理逻辑的复杂程度决定。函数是一个接收输入参数处理功能输出结果的一个过程。程序运行的过程即调用函数处理数据的过程。

3．举例说明，最简单的加法函数，接收两个输入参数，第一个数字和第二个数字，它的处理功能就是把两个数字相加，最后输出结果。

二、C语言的特点

C语言既具有一般咼级语言特性，又具有低级语言特性。 8个特点。

1．C语言简洁、紧凑。

2．运算符丰富。

3．数据结构类型丰富。

4．具有结构化的控制语句。

5．语法限制不太严格，程序设计自由度大。

6．C语言允许直接访问物理地址，能实现汇编语言的大部分功能，可以直接对硬件进行操

作。

7．生成目标代码质量高，程序执行效率高。

8．与汇编语言相比，用 C 语言写的程序可移植性好。

三、课程章节概览

讲解课程各章节学习的内容和内在的联系，建立学习框架：

1．程序设计与 C 语言

2．数据类型、运算符与表达式

3．C 程序中的输入、输出

4．选择结构程序设计

5．循环结构程序设计

6．函数实现模块化程序设计

7．数组

8．指针

9．用户自己建立数据类型

10．文件的输入输出

四、编程语言的发展

1．机器语言（由 0 和 1 组成的指令）

2．符号语言（用英文字母和数字表示指令）

3．高级语言（接近于人的自然语言和数学语言）

4．面向过程的语言（非结构化的语言、结构化语言）

5．面向对象的语言

五、简单 C 程序

1．主函数，也称 main 函数，程序执行的入口，必不可少且只能有 1 个

2．printf 输出函数，在程序运行框中输出文本“机电一体化”

3．printf 函数被封装在<stdio.h>函数库中，须预编译该库才能调用其中的函数

代码示例：

```
#include <stdio.h>
int main( )
{
    printf( “机电一体化” );
}
```

六、课程使用编程软件的安装与使用（Dev C++）

演示 Dev C++软件的安装和使用

第二章 数据类型、运算符与表达式（3 课时）

教学目标：

理解并掌握 C 语言的基本数据类型及其使用方法。

掌握 C 语言中常用运算符的使用及其优先级。

能够正确书写和理解 C 语言中的表达式。

能够运用数据类型、运算符和表达式解决简单的编程问题。

教学方法、手段：

讲授、上机操作

教学重点：

基本数据类型及其取值范围。

算术运算符、关系运算符、逻辑运算符、赋值运算符的使用。

表达式的求值顺序和类型转换。

教学难点：

理解不同类型数据在内存中的存储方式。

掌握运算符的优先级和结合性。

理解隐式类型转换和显式类型转换的区别。

课程思政：

严谨治学、精益求精

讲解数据类型时，强调选择合适的数据类型可以提高程序的效率和准确性，引导学生养成严谨治学的态度。

讲解运算符时，强调运算符的优先级和结合性会影响程序的运行结果，引导学生注重细节，精益求精。

教学过程：

一、数据类型

引入

通过生活中的例子引入数据类型的概念，例如：年龄用整数表示，身高用小数表示，姓名用字符串表示等。

强调数据类型在程序中的重要性，它决定了数据的存储空间、取值范围和可进行的操作。

讲解基本数据类型

介绍 C 语言中的基本数据类型：int、float、double、char。

讲解每种数据类型的取值范围、存储空间和常用场景。

通过代码演示不同类型变量的声明、赋值和使用。

讲解常量

介绍常量的概念，包括整型常量、实型常量、字符常量和字符串常量。

讲解常量的定义方法和使用注意事项。

课堂练习

布置一些简单的练习题，例如：声明不同类型的变量并赋值、输出不同类型的数据等。

总结

回顾本节课的主要内容，强调数据类型的选择和使用对程序的影响。

二、运算符

引入

通过简单的数学运算引入运算符的概念，例如：加、减、乘、除等。

强调运算符在程序中的重要性，它用于对数据进行各种操作。

讲解算术运算符

介绍 C 语言中的算术运算符：+、-、*、/、%。

讲解每种运算符的功能和使用方法，重点讲解除法运算符和取余运算符的区别。

讲解关系运算符

介绍 C 语言中的关系运算符：>、<、>=、<=、==、!=。

讲解每种运算符的功能和使用方法，重点讲解等于运算符和赋值运算符的区别。

讲解逻辑运算符

介绍 C 语言中的逻辑运算符：&&、||、!。

讲解每种运算符的功能和使用方法，重点讲解逻辑运算符的短路特性。

讲解赋值运算符

介绍 C 语言中的赋值运算符：=、+=、-=、*=、/=、%=。

讲解每种运算符的功能和使用方法，重点讲解复合赋值运算符的使用。

课堂练习

布置一些简单的练习题，例如：使用不同的运算符进行运算、比较两个数的大小、判断一个数的奇偶性等。

总结

回顾本节课的主要内容，强调运算符的优先级和结合性。

三、表达式

引入

通过简单的数学表达式引入表达式的概念，例如：a + b * c。

强调表达式在程序中的重要性，它用于计算和处理数据。

讲解表达式的组成

介绍表达式的组成元素：操作数、运算符和括号。

讲解操作数的类型和运算符的优先级。

讲解表达式的求值顺序

介绍表达式的求值顺序：先计算括号内的表达式，再按照运算符的优先级进行计算。

通过代码演示不同表达式的求值过程，重点讲解运算符的优先级和结合性。

讲解类型转换

介绍类型转换的概念，包括隐式类型转换和显式类型转换。

讲解不同类型数据之间进行运算时的类型转换规则，重点讲解数据精度丢失的问题。

课堂练习

布置一些简单的练习题，例如：计算不同表达式的值、分析表达式的求值顺序、进行类型转换等。

总结

回顾本节课的主要内容，强调表达式的书写规范和类型转换的注意事项。

第三章 C 程序中的输入、输出（3 课时）

教学目标：

掌握 C 语言中常用的输入输出函数及其使用方法。

理解格式化输入输出的概念，能够使用格式控制符进行数据的格式化输入输出。

能够编写简单的 C 程序，实现数据的输入输出功能。

教学方法、手段：

讲授、上机操作

教学重点：

printf() 和 scanf() 函数的使用。

格式控制符的使用。

字符输入输出函数的使用。

教学难点：

理解格式控制符的含义和使用方法。

掌握不同类型数据的输入输出格式。

处理输入输出过程中的错误。

课程思政：

信息素养、网络安全

讲解输入输出函数时，强调用户输入数据的合法性和安全性，引导学生提高信息素养，增强网络安全意识。

讲解文件操作时，强调数据备份和隐私保护的重要性，引导学生养成良好的数据管理习惯。

教学过程：

一、标准输入输出函数

引入

通过简单的例子引入输入输出的概念，例如：从键盘输入数据，在屏幕上显示结果等。

强调输入输出在程序中的重要性，它是程序与用户交互的桥梁。

讲解 printf() 函数

介绍 printf() 函数的功能：将格式化的数据输出到标准输出设备（通常是屏幕）。

讲解 printf() 函数的基本语法和常用格式控制符，例如：%d、%f、%c、%s 等。

通过代码演示不同类型数据的输出，例如：整数、浮点数、字符、字符串等。

讲解 scanf() 函数

介绍 scanf() 函数的功能：从标准输入设备（通常是键盘）读取格式化的数据。

讲解 scanf() 函数的基本语法和常用格式控制符，例如：%d、%f、%c、%s 等。

通过代码演示不同类型数据的输入，例如：整数、浮点数、字符、字符串等。

课堂练习

布置一些简单的练习题，例如：使用 printf() 函数输出个人信息，使用 scanf() 函数输入两个整数并计算它们的和等。

总结

回顾本节课的主要内容，强调 printf() 和 scanf() 函数的使用方法和注意事项。

二、格式化输入输出

引入

通过例子引入格式化输入输出的概念，例如：控制输出的数字位数、对齐方式等。

强调格式化输入输出可以使程序的输出更加美观和易读。

讲解格式控制符

详细介绍常用的格式控制符，例如：

1. %d：输出十进制整数。
2. %f：输出浮点数。
3. %c：输出字符。
4. %s：输出字符串。
5. %x：输出十六进制整数。
6. %o：输出八进制整数。
7. %e：以科学计数法输出浮点数。
8. %g：根据数值大小自动选择 %f 或 %e 格式输出浮点数。

讲解格式控制符的修饰符，例如：

-：左对齐。

+：显示正负号。

0：用前导零填充。

m.n：指定输出宽度和小数位数。

通过代码演示不同格式控制符的使用效果。

讲解 getchar() 和 putchar() 函数

介绍 getchar() 函数的功能：从标准输入设备读取一个字符。

介绍 putchar() 函数的功能：向标准输出设备输出一个字符。

通过代码演示 getchar() 和 putchar() 函数的使用。

课堂练习

布置一些简单的练习题，例如：使用格式控制符控制输出的数字位数和对齐方式，使用 getchar() 和 putchar() 函数实现简单的字符输入输出等。

总结

回顾本节课的主要内容，强调格式控制符的使用技巧和注意事项。

三、综合练习

复习

复习输入输出函数和格式控制符的相关知识，解答学生疑问。

综合练习

布置一些综合性的练习题，例如：

编写程序，输入学生的姓名、年龄、成绩等信息，并按照指定格式输出。

编写程序，实现简单的计算器功能，能够进行加、减、乘、除运算，并格式化输出结果。

编写程序，输入一个字符串，统计其中字母、数字和其他字符的个数，并输出统计结果。

总结

总结本章节的主要内容，强调输入输出函数在程序设计中的重要性。

第四章 选择结构程序设计（9 课时）

教学目标：

理解选择结构的概念和作用。

掌握 C 语言中 if 语句、if-else 语句、嵌套 if 语句和 switch 语句的使用方法。

能够运用选择结构解决实际问题。

教学方法、手段：

讲授、上机操作

教学重点：

if 语句、if-else 语句的语法和执行流程。

嵌套 if 语句的使用。

switch 语句的语法和执行流程。

教学难点：

理解选择结构的执行流程。

掌握嵌套 if 语句的逻辑关系。

理解 switch 语句中 break 语句的作用。

课程思政：

逻辑思维、批判性思维

讲解选择结构时，强调逻辑思维的重要性，引导学生学会分析问题、解决问题。

讲解嵌套选择结构时，强调批判性思维的重要性，引导学生学会多角度思考问题，避免思维定势。

教学过程：

一、if 语句

引入

通过生活中的例子引入选择结构的概念，例如：如果明天不下雨，我们就去公园玩。

强调选择结构在程序中的重要性，它可以根据条件决定程序的执行路径。

讲解 if 语句

介绍 if 语句的功能：根据条件判断是否执行某段代码。

讲解 if 语句的基本语法和执行流程。

通过代码演示 if 语句的使用，例如：判断一个数是否为正数，判断一个年份是否为闰年等。

课堂练习

布置一些简单的练习题，例如：使用 if 语句判断一个数的奇偶性，判断一个字符是否为字母等。

总结

回顾本节课的主要内容，强调 if 语句的使用方法和注意事项。

二、if-else 语句

引入

通过例子引入 if-else 语句的概念，例如：如果明天不下雨，我们就去公园玩，否则就在家看电影。

强调 if-else 语句可以处理两种不同的情况。

讲解 if-else 语句

介绍 if-else 语句的功能：根据条件判断执行不同的代码块。

讲解 if-else 语句的基本语法和执行流程。

通过代码演示 if-else 语句的使用，例如：判断一个数的正负性，判断一个学生的成绩是否及格等。

课堂练习

布置一些简单的练习题，例如：使用 if-else 语句判断一个数的最大值，判断一个字符是否为数字等。

总结

回顾本节课的主要内容，强调 if-else 语句的使用方法和注意事项。

三、嵌套 if 语句

引入

通过例子引入嵌套 if 语句的概念，例如：如果明天不下雨，我们就去公园玩，如果公园人多，我们就去动物园。

强调嵌套 if 语句可以处理更复杂的条件判断。

讲解嵌套 if 语句

介绍嵌套 if 语句的功能：在 if 语句或 if-else 语句中嵌套另一个 if 语句或 if-else 语句。

讲解嵌套 if 语句的基本语法和执行流程。

通过代码演示嵌套 if 语句的使用，例如：判断一个数的范围，判断一个学生的成绩等级等。

课堂练习

布置一些简单的练习题，例如：使用嵌套 if 语句判断一个数的符号和大小，判断一个字符的类型等。

总结

回顾本节课的主要内容，强调嵌套 if 语句的逻辑关系和注意事项。

四、switch 语句

引入

通过例子引入 switch 语句的概念，例如：根据用户的选择执行不同的操作。

强调 switch 语句可以处理多分支选择的情况。

讲解 switch 语句

介绍 switch 语句的功能：根据表达式的值选择执行不同的代码块。

讲解 switch 语句的基本语法和执行流程，重点讲解 break 语句的作用。

通过代码演示 switch 语句的使用，例如：根据用户输入的数字显示对应的星期几，根据用户选择的运算符进行相应的计算等。

课堂练习

布置一些简单的练习题，例如：使用 switch 语句实现简单的计算器功能，根据用户输入的成绩等级显示相应的评语等。

总结

回顾本节课的主要内容，强调 switch 语句的使用方法和注意事项。

五、综合练习

复习

复习选择结构的相关知识，解答学生疑问。

综合练习

布置一些综合性的练习题，例如：

编写程序，输入一个年份，判断该年是否为闰年。

编写程序，输入一个学生的成绩，判断其等级（90 分以上为 A，80-89 分为 B，70-79 分为 C，60-69 分为 D，60 分以下为 E），并输出结果。

编写程序，实现简单的计算器功能，能够进行加、减、乘、除运算，并输出结果。

总结

总结本章节的主要内容，强调选择结构在程序设计中的重要性。

六、项目实践

项目介绍

介绍一个简单的项目，例如：学生成绩管理系统、简易计算器等。

讲解项目的功能需求和实现思路。

项目实践

学生分组完成项目，教师进行指导。

项目展示

学生展示项目成果，教师进行点评。

总结

总结项目实践的经验教训，强调选择结构的实际应用。

第五章 循环结构程序设计（9 课时）

教学目标：

理解循环结构的概念和作用。

掌握 C 语言中 while 循环、do-while 循环和 for 循环的使用方法。

能够运用循环结构解决实际问题。

教学方法、手段：

讲授、上机操作

教学重点：

while 循环、do-while 循环和 for 循环的语法和执行流程。

循环控制语句（break、continue）的使用。

嵌套循环的使用。

教学难点：

理解循环结构的执行流程。

掌握循环控制语句的使用时机。

理解嵌套循环的逻辑关系。

课程思政：

坚持不懈、勇于创新

讲解循环结构时，强调坚持不懈的重要性，引导学生克服困难，勇于挑战。

讲解循环控制语句时，强调勇于创新的重要性，引导学生探索新的解决方案，优化程序性能。

教学过程：

一、while 循环

引入

通过生活中的例子引入循环结构的概念，例如：重复执行某个动作直到满足某个条件。

强调循环结构在程序中的重要性，它可以简化重复性代码的编写。

讲解 while 循环

介绍 while 循环的功能：当条件为真时，重复执行循环体中的代码。

讲解 while 循环的基本语法和执行流程。

通过代码演示 while 循环的使用，例如：计算 1 到 100 的和，输出九九乘法表等。

课堂练习

布置一些简单的练习题，例如：使用 while 循环计算 1 到 n 的和，输出指定范围内的偶数等。

总结

回顾本节课的主要内容，强调 while 循环的使用方法和注意事项。

二、do-while 循环

引入

通过例子引入 do-while 循环的概念，例如：先执行一次循环体，再判断条件是否成立。

强调 do-while 循环与 while 循环的区别。

讲解 do-while 循环

介绍 do-while 循环的功能：先执行循环体中的代码，再判断条件是否为真，如果为真则继续执行循环体。

讲解 do-while 循环的基本语法和执行流程。

通过代码演示 do-while 循环的使用，例如：输入一个正整数，计算其各位数字之和等。

课堂练习

布置一些简单的练习题，例如：使用 do-while 循环实现简单的猜数字游戏，输入一个字符串并统计其长度等。

总结

回顾本节课的主要内容，强调 do-while 循环的使用方法和注意事项。

三、for 循环

引入

通过例子引入 for 循环的概念，例如：已知循环次数的情况下，使用 for 循环更加简洁。

强调 for 循环的简洁性和高效性。

讲解 for 循环

介绍 for 循环的功能：在已知循环次数的情况下，重复执行循环体中的代码。

讲解 for 循环的基本语法和执行流程，重点讲解循环变量的初始值、循环条件和循环变量的更新。

通过代码演示 for 循环的使用，例如：输出 1 到 100 的整数，计算 1 到 n 的阶乘等。

课堂练习

布置一些简单的练习题，例如：使用 for 循环输出指定范围内的素数，计算斐波那契数列的前 n 项等。

总结

回顾本节课的主要内容，强调 for 循环的使用方法和注意事项。

四、循环控制语句

引入

通过例子引入循环控制语句的概念，例如：在循环过程中，根据某些条件提前结束循环或跳过某次循环。

强调循环控制语句可以增强程序的灵活性。

讲解 break 语句

介绍 break 语句的功能：提前结束循环。

讲解 break 语句的使用方法，重点讲解 break 语句在嵌套循环中的作用。

通过代码演示 break 语句的使用，例如：在循环中查找指定元素，找到后立即结束循环等。

讲解 continue 语句

介绍 continue 语句的功能：跳过本次循环，继续执行下一次循环。

讲解 continue 语句的使用方法，重点讲解 continue 语句在嵌套循环中的作用。

通过代码演示 continue 语句的使用，例如：在循环中输出 1 到 100 的奇数，跳过偶数等。

课堂练习

布置一些简单的练习题，例如：使用 break 语句实现简单的猜数字游戏，使用 continue 语句输出指定范围内的素数等。

总结

回顾本节课的主要内容，强调循环控制语句的使用方法和注意事项。

五、嵌套循环

引入

通过例子引入嵌套循环的概念，例如：在循环体中嵌套另一个循环。

强调嵌套循环可以处理更复杂的问题。

讲解嵌套循环

介绍嵌套循环的功能：在循环体中嵌套另一个循环，用于处理二维数组、矩阵等问题。

讲解嵌套循环的基本语法和执行流程，重点讲解内外层循环的执行顺序。

通过代码演示嵌套循环的使用，例如：输出九九乘法表，打印菱形图案等。

课堂练习

布置一些简单的练习题，例如：使用嵌套循环输出指定范围内的素数，打印杨辉三角等。

总结

回顾本节课的主要内容，强调嵌套循环的逻辑关系和注意事项。

六、综合练习

复习

复习循环结构的相关知识，解答学生疑问。

综合练习

布置一些综合性的练习题，例如：

编写程序，输入一个正整数，判断其是否为素数。

编写程序，输入一个字符串，统计其中每个字符出现的次数。

编写程序，实现简单的猜数字游戏，并统计用户猜测的次数。

总结

总结本章节的主要内容，强调循环结构在程序设计中的重要性。

七、项目实践

项目介绍

介绍一个简单的项目，例如：学生成绩管理系统、简易计算器等。

讲解项目的功能需求和实现思路。

项目实践

学生分组完成项目，教师进行指导。

项目展示

学生展示项目成果，教师进行点评。

总结

总结项目实践的经验教训，强调循环结构的实际应用。

八、复习与测试

复习

复习循环结构的相关知识，解答学生疑问。

测试

进行本章节的测试，检验学生的学习效果。

总结

总结本章节的学习情况，强调循环结构在程序设计中的重要性。

第六章 函数实现模块化程序设计（6 课时）

教学目标：

理解函数的概念和作用。

掌握 C 语言中函数的定义、声明和调用方法。

理解函数参数传递的方式（值传递和地址传递）。

掌握函数的递归调用。

能够运用函数实现模块化程序设计。

教学方法、手段：

讲授、上机操作

教学重点：

函数的定义、声明和调用。

函数参数传递的方式。

函数的递归调用。

教学难点：

理解函数参数传递的方式。

掌握递归函数的编写方法。

理解模块化程序设计的思想。

课程思政：

团队合作、代码规范

讲解函数时，强调团队合作的重要性，引导学生学会与他人协作，共同完成项目。

讲解代码规范时，强调代码可读性和可维护性的重要性，引导学生养成良好的编程习惯。

教学过程：

一、函数的基本概念

引入

通过生活中的例子引入函数的概念，例如：计算器中的加减乘除功能可以看作是不同的函数。

强调函数在程序中的重要性，它可以提高代码的复用性和可维护性。

讲解函数的基本概念

介绍函数的功能：将一段完成特定功能的代码封装起来，方便调用。

讲解函数的组成部分：函数名、参数列表、返回值类型、函数体。

通过代码演示一个简单的函数，例如：计算两个数的和。

课堂练习

布置一些简单的练习题，例如：编写一个函数，计算两个数的乘积；编写一个函数，判断一个数是否为偶数等。

总结

回顾本节课的主要内容，强调函数的基本概念和使用方法。

二、函数的定义、声明和调用

引入

通过例子引入函数的定义、声明和调用的概念，例如：在使用函数之前，需要先定义或声明函数。

强调函数的定义、声明和调用的区别。

讲解函数的定义

介绍函数定义的功能：指定函数的名称、参数列表、返回值类型和函数体。

讲解函数定义的语法规则。

通过代码演示函数的定义，例如：定义一个函数，计算两个数的最大值。

讲解函数的声明

介绍函数声明的功能：告诉编译器函数的名称、参数列表和返回值类型。

讲解函数声明的语法规则。

通过代码演示函数的声明，例如：声明一个函数，计算两个数的最小值。

讲解函数的调用

介绍函数调用的功能：执行函数体中的代码。

讲解函数调用的语法规则。

通过代码演示函数的调用，例如：调用之前定义的函数，计算两个数的最大值和最小值。

课堂练习

布置一些简单的练习题，例如：编写一个函数，计算圆的面积；编写一个函数，判断一个字符串是否为回文字符串等。

总结

回顾本节课的主要内容，强调函数的定义、声明和调用的方法和注意事项。

三、函数参数传递

引入

通过例子引入函数参数传递的概念，例如：在调用函数时，需要将实际参数传递给形式参数。

强调函数参数传递的方式对函数执行结果的影响。

讲解值传递

介绍值传递的功能：将实际参数的值复制一份传递给形式参数。

讲解值传递的特点：形式参数的改变不会影响实际参数。

通过代码演示值传递的使用，例如：编写一个函数，交换两个数的值。

讲解地址传递

介绍地址传递的功能：将实际参数的地址传递给形式参数。

讲解地址传递的特点：形式参数的改变会影响实际参数。

通过代码演示地址传递的使用，例如：编写一个函数，交换两个数的值。

课堂练习

布置一些简单的练习题，例如：编写一个函数，计算数组元素的平均值；编写一个函数，将字符串中的大写字母转换为小写字母等。

总结

回顾本节课的主要内容，强调值传递和地址传递的区别和使用方法。

四、函数的递归调用

引入

通过例子引入递归的概念，例如：计算阶乘、斐波那契数列等。

强调递归函数的简洁性和高效性。

讲解递归函数

介绍递归函数的功能：函数调用自身。

讲解递归函数的编写方法，重点讲解递归终止条件。

通过代码演示递归函数的使用，例如：编写一个递归函数，计算阶乘；编写一个递归函数，计算斐波那契数列的第 n 项等。

课堂练习

布置一些简单的练习题，例如：编写一个递归函数，计算汉诺塔问题的移动步骤；编写一个递归函数，判断一个字符串是否为回文字符串等。

总结

回顾本节课的主要内容，强调递归函数的编写方法和注意事项。

五、模块化程序设计

引入

通过例子引入模块化程序设计的概念，例如：将一个大程序分解成多个小模块，每个模块完成特定的功能。

强调模块化程序设计的优点：提高代码的复用性、可维护性和可读性。

讲解模块化程序设计

介绍模块化程序设计的思想：将程序分解成多个模块，每个模块完成特定的功能。

讲解模块化程序设计的实现方法：使用函数将程序分解成多个模块。

通过代码演示模块化程序设计的使用，例如：编写一个学生成绩管理系统，将不同的功能模块化。

课堂练习

布置一些简单的练习题，例如：编写一个简单的计算器程序，将不同的运算功能模块化；编写一个简单的图书管理系统，将不同的功能模块化等。

总结

回顾本节课的主要内容，强调模块化程序设计的思想和实现方法。

六、综合练习

复习

复习函数的相关知识，解答学生疑问。

综合练习

布置一些综合性的练习题，例如：

编写一个程序，计算并输出 1 到 100 之间的所有素数。

编写一个程序，输入一个字符串，统计其中每个字符出现的次数。

编写一个程序，实现简单的猜数字游戏，并统计用户猜测的次数。

总结

总结本章节的主要内容，强调函数在程序设计中的重要性。

第七章　数组（6 课时）

教学目标：

理解数组的概念和作用。

掌握一维数组和二维数组的定义、初始化和使用方法。

掌握字符数组和字符串的使用方法。

能够运用数组解决实际问题。

教学方法、手段：

讲授、上机操作

教学重点：

一维数组和二维数组的定义、初始化和使用。

字符数组和字符串的使用。

数组作为函数参数的使用。

教学难点：

理解数组在内存中的存储方式。

掌握二维数组的遍历方法。

理解字符串的存储方式和常用操作。

课程思政：

数据结构、算法效率

讲解数组时，强调数据结构的重要性，引导学生理解不同数据结构的优缺点。

讲解数组操作时，强调算法效率的重要性，引导学生优化算法，提高程序性能。

教学过程：

一、一维数组

引入

通过生活中的例子引入数组的概念，例如：一个班级的学生名单可以看作是一个数组。

强调数组在程序中的重要性，它可以存储一组相同类型的数据。

讲解一维数组的定义和初始化

介绍一维数组的功能：存储一组相同类型的数据。

讲解一维数组的定义语法和初始化方法。

通过代码演示一维数组的定义和初始化，例如：定义一个数组存储 5 个学生的成绩。

讲解一维数组的访问和遍历

介绍一维数组的访问方法：通过下标访问数组元素。

讲解一维数组的遍历方法：使用 for 循环遍历数组。

通过代码演示一维数组的访问和遍历，例如：输出数组中的所有元素，计算数组元素的平均值等。

课堂练习

布置一些简单的练习题，例如：定义一个数组存储 10 个整数，计算数组元素的和；定义一个数组存储 5 个学生的成绩，输出最高分和最低分等。

总结

回顾本节课的主要内容，强调一维数组的定义、初始化、访问和遍历方法。

二、二维数组

引入

通过例子引入二维数组的概念，例如：一个班级的学生成绩表可以看作是一个二维数组。

强调二维数组可以存储表格形式的数据。

讲解二维数组的定义和初始化

介绍二维数组的功能：存储表格形式的数据。

讲解二维数组的定义语法和初始化方法。

通过代码演示二维数组的定义和初始化，例如：定义一个二维数组存储 3 个学生的 4 门课程成绩。

讲解二维数组的访问和遍历

介绍二维数组的访问方法：通过行下标和列下标访问数组元素。

讲解二维数组的遍历方法：使用嵌套 for 循环遍历数组。

通过代码演示二维数组的访问和遍历，例如：输出二维数组中的所有元素，计算每行元素的平均值等。

课堂练习

布置一些简单的练习题，例如：定义一个二维数组存储 3 个学生的 4 门课程成绩，输出每个学生的平均成绩；定义一个二维数组存储 3x3 矩阵，计算矩阵的转置等。

总结

回顾本节课的主要内容，强调二维数组的定义、初始化、访问和遍历方法。

三、字符数组和字符串

引入

通过例子引入字符数组和字符串的概念，例如：一个字符串可以看作是一个字符数组。

强调字符串在程序中的重要性，它可以存储和处理文本数据。

讲解字符数组的定义和初始化

介绍字符数组的功能：存储一组字符。

讲解字符数组的定义语法和初始化方法。

通过代码演示字符数组的定义和初始化，例如：定义一个字符数组存储一个字符串。

讲解字符串的存储和操作

介绍字符串的存储方式：以'\0'结尾的字符数组。

讲解字符串的常用操作：字符串的输入输出、字符串的比较、字符串的连接等。

通过代码演示字符串的存储和操作，例如：输入一个字符串，输出其长度；比较两个字符串的大小；连接两个字符串等。

课堂练习

布置一些简单的练习题，例如：定义一个字符数组存储一个字符串，输出其反转字符串；定义一个字符数组存储一个字符串，判断其是否为回文字符串等。

总结

回顾本节课的主要内容，强调字符数组和字符串的定义、初始化、存储和操作方法。

四、数组作为函数参数

引入

通过例子引入数组作为函数参数的概念，例如：将一个数组传递给函数进行处理。

强调数组作为函数参数可以提高代码的复用性。

讲解数组作为函数参数的使用

介绍数组作为函数参数的功能：将数组传递给函数进行处理。

讲解数组作为函数参数的语法规则和使用方法。

通过代码演示数组作为函数参数的使用，例如：编写一个函数，计算数组元素的平均值；编写一个函数，对数组元素进行排序等。

课堂练习

布置一些简单的练习题，例如：编写一个函数，计算数组元素的最大值；编写一个函数，查找数组中的指定元素等。

总结

回顾本节课的主要内容，强调数组作为函数参数的使用方法和注意事项。

五、综合练习

复习

复习数组的相关知识，解答学生疑问。

综合练习

布置一些综合性的练习题，例如：

编写一个程序，输入一个字符串，统计其中每个字符出现的次数。

编写一个程序，实现简单的学生成绩管理系统，能够存储和查询学生的成绩。

编写一个程序，实现简单的矩阵运算，能够进行矩阵的加、减、乘运算。

总结

总结本章节的主要内容，强调数组在程序设计中的重要性。

六、项目实践

项目介绍

介绍一个简单的项目，例如：学生成绩管理系统、简易计算器等。

讲解项目的功能需求和实现思路。

项目实践

学生分组完成项目，教师进行指导。

项目展示

学生展示项目成果，教师进行点评。

总结

总结项目实践的经验教训，强调数组的实际应用。

第八章　指针（3 课时）

教学目标：

理解指针的概念和作用。

掌握指针变量的定义、初始化和使用方法。

掌握指针与数组、指针与函数的关系。

能够运用指针解决实际问题。

教学方法、手段：

讲授、上机操作

教学重点：

指针变量的定义、初始化和使用。

指针与数组的关系。

指针与函数的关系。

教学难点：

理解指针的概念和内存地址的关系。

掌握指针运算和指针类型的转换。

理解指针数组和数组指针的区别。

课程思政：

内存管理、安全意识

讲解指针时，强调内存管理的重要性，引导学生理解指针操作的潜在风险。

讲解指针安全时，强调安全意识的重要性，引导学生编写安全可靠的代码。

教学过程：

一、指针的基本概念

引入

通过生活中的例子引入指针的概念，例如：门牌号可以看作是指针，它指向具体的房屋。

强调指针在程序中的重要性，它可以间接访问内存中的数据。

讲解指针的基本概念

介绍指针的功能：存储内存地址。

讲解指针变量的定义、初始化和使用方法。

通过代码演示指针变量的定义、初始化和使用，例如：定义一个指针变量，指向一个整数变量。

课堂练习

布置一些简单的练习题，例如：定义一个指针变量，指向一个字符变量；定义一个指针变量，指向一个数组等。

总结

回顾本节课的主要内容，强调指针的基本概念和使用方法。

二、指针与数组

引入

通过例子引入指针与数组的关系，例如：数组名可以看作是指向数组首元素的指针。

强调指针可以方便地遍历数组。

讲解指针与数组的关系

介绍数组名的含义：指向数组首元素的指针。

讲解指针运算和数组元素的访问方法。

通过代码演示指针与数组的关系，例如：使用指针遍历数组，使用指针访问数组元素等。

课堂练习

布置一些简单的练习题，例如：使用指针计算数组元素的和；使用指针查找数组中的最大值等。

总结

回顾本节课的主要内容，强调指针与数组的关系和使用方法。

三、指针与函数

引入

通过例子引入指针与函数的关系，例如：可以将指针作为函数参数，传递数组或结构体等复杂数据。

强调指针可以提高函数的灵活性。

讲解指针与函数的关系

介绍指针作为函数参数的功能：传递数组或结构体等复杂数据。

讲解指针作为函数返回值的功能：返回动态分配的内存地址。

通过代码演示指针与函数的关系，例如：编写一个函数，交换两个数的值；编写一个函数，返回动态分配的数组等。

课堂练习

布置一些简单的练习题，例如：编写一个函数，计算数组元素的平均值；编写一个函数，查找数组中的指定元素等。

总结

回顾本节课的主要内容，强调指针与函数的关系和使用方法。

第九章 用户自己建立数据类型（3 课时）

教学目标：

理解结构体和共用体的概念和作用。

掌握结构体和共用体的定义、初始化和使用方法。

掌握枚举类型的定义和使用方法。

能够运用用户自定义数据类型解决实际问题。

教学方法、手段：

讲授、上机操作

教学重点：

结构体的定义、初始化和使用。

共用体的定义、初始化和使用。

枚举类型的定义和使用。

教学难点：

理解结构体和共用体的内存分配方式。

掌握结构体数组和结构体指针的使用。

理解枚举类型的应用场景。

课程思政：

抽象思维、问题建模

讲解结构体时，强调抽象思维的重要性，引导学生将现实世界中的对象抽象为程序中的数据结构。

讲解共用体时，强调问题建模的重要性，引导学生根据实际问题选择合适的数据类型。

教学过程：

一、结构体

引入

通过生活中的例子引入结构体的概念，例如：学生信息可以包含姓名、学号、成绩等不同类型的数据。

强调结构体可以将不同类型的数据组合在一起。

讲解结构体的定义和初始化

介绍结构体的功能：将不同类型的数据组合在一起。

讲解结构体的定义语法和初始化方法。

通过代码演示结构体的定义和初始化，例如：定义一个结构体存储学生信息。

课堂练习

布置一些简单的练习题，例如：定义一个结构体存储图书信息；定义一个结构体存储日期信息等。

总结

回顾本节课的主要内容，强调结构体的定义、初始化和使用方法。

二、共用体和枚举类型

引入

通过例子引入共用体和枚举类型的概念，例如：共用体可以节省内存空间，枚举类型可以提

高代码的可读性。

强调共用体和枚举类型在特定场景下的优势。

讲解共用体的定义和使用

介绍共用体的功能：节省内存空间。

讲解共用体的定义语法和使用方法。

通过代码演示共用体的定义和使用，例如：定义一个共用体存储不同类型的数据。

讲解枚举类型的定义和使用

介绍枚举类型的功能：提高代码的可读性。

讲解枚举类型的定义语法和使用方法。

通过代码演示枚举类型的定义和使用，例如：定义一个枚举类型表示星期几。

课堂练习

布置一些简单的练习题，例如：定义一个共用体存储不同类型的数据；定义一个枚举类型表示颜色等。

总结

回顾本节课的主要内容，强调共用体和枚举类型的定义和使用方法。

三、综合练习

复习

复习用户自定义数据类型的相关知识，解答学生疑问。

综合练习

布置一些综合性的练习题，例如：

编写一个程序，定义一个结构体存储学生信息，并实现学生信息的输入输出。

编写一个程序，定义一个共用体存储不同类型的数据，并测试其内存占用情况。

编写一个程序，定义一个枚举类型表示星期几，并输出对应的中文名称。

总结

总结本章节的主要内容，强调用户自定义数据类型在程序设计中的重要性。

第十章 文件的输入输出（3 课时）

教学目标：

理解文件的概念和作用。

掌握文件的打开、关闭、读写等操作。

能够运用文件的输入输出解决实际问题。

教学方法、手段：

讲授、上机操作

教学重点：

文件的打开、关闭、读写等操作。

文件指针的使用。

教学难点：

理解文件指针的概念和作用。

掌握文件读写操作的细节和注意事项。

理解文件定位和文件缓冲区的概念。

课程思政：：

持久化、信息安全

讲解文件操作时，强调数据持久化的重要性，引导学生理解数据存储和读取的原理。

讲解文件安全时，强调信息安全的重要性，引导学生保护数据隐私，防止数据泄露。

教学过程：

一、文件的基本操作

引入

通过生活中的例子引入文件的概念，例如：文本文件、图片文件、视频文件等。

强调文件在程序中的重要性，它可以持久化存储数据。

讲解文件的基本操作

介绍文件的基本操作：打开、关闭、读写等。

讲解文件指针的概念和使用方法。

通过代码演示文件的基本操作，例如：打开一个文件，读取文件内容，写入文件内容等。

课堂练习

布置一些简单的练习题，例如：编写一个程序，读取一个文本文件的内容并输出；编写一个程序，将用户输入的内容写入一个文本文件等。

总结

回顾本节课的主要内容，强调文件的基本操作和使用方法。

二、文件的读写操作

引入

通过例子引入文件的读写操作，例如：读取配置文件、保存用户数据等。

强调文件的读写操作是文件操作的核心。

讲解文件的读写操作

介绍文件的读写操作：fread、fwrite、fscanf、fprintf 等。

讲解文件读写操作的细节和注意事项。

通过代码演示文件的读写操作，例如：读取一个二进制文件的内容，写入一个二进制文件的内容等。

课堂练习

布置一些简单的练习题，例如：编写一个程序，读取一个二进制文件的内容并输出；编写一个程序，将用户输入的内容写入一个二进制文件等。

总结

回顾本节课的主要内容，强调文件的读写操作和使用方法。

三、综合练习

复习

复习文件的输入输出相关知识，解答学生疑问。

综合练习

布置一些综合性的练习题，例如：

编写一个程序，实现简单的学生成绩管理系统，能够将学生信息保存到文件中，并能够从文件中读取学生信息。

编写一个程序，实现简单的文件加密解密功能。

编写一个程序，统计一个文本文件中每个单词出现的次数。

总结

总结本章节的主要内容，强调文件的输入输出在程序设计中的重要性。