

《13794 计算机程序设计基础》

实践考核大纲

一、课程性质与目标

(一) 课程性质和特点

本课程是计算机应用技术及相关专业的核心专业基础实践课，依托 C/C++ 语言，聚焦程序设计的实际操作与应用能力培养。课程将程序设计理论与上机实践深度结合，强调计算思维的塑造和编程实操技能的训练，要求学生熟练运用集成开发环境完成程序的编辑、编译、调试与运行，具备运用结构化程序设计方法解决实际数据处理问题的能力，是衔接理论学习与后续专业课程（数据结构、算法设计等）的关键课程。

(二) 课程目标

本课程设置的目的在于培养学生具备程序设计基础实操能力，能够熟练使用 C 语言集成开发环境，遵循语法规则编写、调试简单程序；具备结构化程序设计应用能力，能灵活运用顺序、选择、循环三种控制结构及数组、函数、指针等核心知识处理批量数据；具备简单问题求解与编程实现能力，能分析实际问题并设计基础算法，完成完整程序的开发；具备文件操作与数据持久化能力，掌握文件的打开、读写、关闭等基本操作，实现数据的存储与读取。通过本课程实践训练，为学生后续专业课程学习和从事软件开发相关实际工作奠定坚实的编程实践基础。

(三) 课程的重点

本课程的重点内容包括：**基础编程操作**，主要包括 C 语言基本数据类型、运算符与表达式的实际运用，scanf/printf 输入输出的规范使用；**结构化程序设计**，学生需掌握选择、循环控制结构的嵌套使用，数组与字符串的批量数据处理技巧；**模块化与指针应用**，熟练进行函数的定义、调用与参数传递，掌握指针与数组、变量的关联操作；**文件基础操作**，能够完成文件的打开、关闭及字符 / 字符串的读写操作；**程序调试与问题修正**，具备识别并修正程序语法错误、简单逻辑错误的能力。

二、考核内容和考核目标

第一章 数据表示、运算符与输入输出

一、学习目的与要求

(1) 掌握 C 语言基本数据类型的分类、定义与初始化规则，能根据实际需求选择合适的数据类型。

(2) 理解常量与变量的区别，掌握符号常量的声明方式和变量的赋值规范。

(3) 熟练运用算术、关系、逻辑运算符构建合法表达式，理解运算符的优先级与结合性。

(4) 掌握 scanf、printf 函数的格式控制方法，能完成不同数据类型的精准输入与输出，避免格式匹配错误。

二、课程内容

(1) 基本数据类型：int（整型）、char（字符型）、float（单精度浮点型）、double（双精度浮点型）的定义、使用场景与取值范围。

(2) 常量与变量：字面常量、符号常量（#define）的声明，变量的声明、初始化与赋值规则。

(3) 运算符与表达式：算术运算符（+、-、*、/、%）、关系运算符（>、<、== 等）、逻辑运算符（&&、||、!）的使用，表达式的合法书写与计算。

(4) 输入输出：scanf 函数不同数据类型的格式控制符使用，printf 函数的格式输出、精度控制与占位符规范。

三、考核知识点及要求

1. 了解 C 语言基本数据类型、常量变量及输入输出的基础原理；

2. 理解理解各类数据类型的使用特点、运算符优先级及输入输出函数的格式规则。

识记：根据程序需求选择合适的数据类型，规范声明常量与变量，正确使用输入输出基础格式控制符。

领会：展现出对表达式合法书写、运算符运算逻辑及输入输出参数匹配的准确理解与应用。运算符的优先级与结合性，变量初始化与赋值的本质区别，输入输出函数的参数匹配规则，表达式运算的类型转换规律。

应用：能根据需求定义并使用变量与常量，编写合法的表达式完成数值计算；能熟练运用 scanf/printf 实现不同数据类型的输入输出，保证格式匹配、数据精准，无语法错误。

第二章 结构化程序设计

一、学习目的与要求

(1) 掌握单分支、双分支 if 语句及多分支 switch-case 语句的语法，能处理嵌套条件逻辑判断。

(2) 熟练运用 for、while、do-while 三种循环语句，能根据实际场景选择合适的循环结构完成批量数据处理。

(3) 掌握 break、continue 跳转语句的功能与使用场景，能通过跳转语句灵活控制程序流程。

(4) 能实现条件语句与循环语句的嵌套使用，运用结构化程序设计思想解决累加、计数、数据筛选等简单实际问题。

二、课程内容

(1) 条件语句: if 单分支语句、if-else 双分支语句、switch-case 多分支语句的语法结构, 条件语句的嵌套实现。

(2) 循环语句: for、while、do-while 循环的语法与执行流程, 循环语句的嵌套使用, 批量数据的遍历与处理。

(3) 跳转语句: break 语句(跳出循环 /switch)、continue 语句(跳过循环本次迭代)的功能与正确使用。

(4) 复合结构: 循环内嵌套条件语句、条件语句内嵌套循环的实现, 复杂逻辑的程序设计与实现。

三、考核知识点及要求

1. 了解结构化程序设计三种基本结构的设计原理, 跳转语句的功能原理;

2. 理解条件、循环语句的语法特点, 嵌套结构的执行逻辑, break 与 continue 的使用区别。

识记: 根据程序需求选择合适的控制结构, 规范书写 if/switch、for/while/do-while 语句格式。

领会: 展现出对条件判断逻辑、循环执行流程及嵌套结构运行顺序的准确理解与把控。

应用: 程序设计中控制结构的灵活组合、嵌套逻辑的合理设计和跳转语句的精准使用能力。

第三章 数组与字符串

一、学习目的与要求

(1) 掌握一维、二维数组的定义、初始化方式, 能通过下标访问数组元素, 利用循环完成数组的遍历。

(2) 理解字符数组与字符串的关联与区别, 掌握字符串结束符 '\0' 的作用与意义。

(3) 熟练运用常用字符串处理函数(strlen、strcpy、strcmp、strcat), 能完成字符串的长度计算、复制、比较与拼接操作。

(4) 能运用数组完成批量数值计算、字符串处理等实际问题, 实现数据的统计与分析。

二、课程内容

(1) 数值数组: 一维数组、二维数组的定义、完全初始化与部分初始化, 数组下标的使用规则, 利用循环遍历数组元素。

(2) 字符数组与字符串: 字符数组的定义与初始化, 字符串的存储特点, 结束符 '\0' 的作用。

(3) 字符串处理函数: strlen(求字符串长度)、strcpy(字符串复制)、

strcmp（字符串比较）、strcat（字符串拼接）的调用格式与使用场景。

（4）数组应用：利用数组完成数据统计、排序、查找，字符串的遍历、修改与简单处理。

三、考核知识点及要求

1. 了解数组的存储原理、字符串的存储特性及字符串结束符的作用原理；
2. 理解一维 / 二维数组的访问特点、字符数组与字符串的关联区别，以及常用字符串处理函数的调用规则。

识记：根据程序需求选择合适的数组类型（一维 / 二维），规范定义与初始化数组，熟记常用字符串处理函数（strlen/strcpy/strcmp/strcat）的功能与基础调用格式。

领会：展现出对数组下标规则、字符串结束符 '\0' 作用，以及字符串函数参数匹配和返回值逻辑的准确理解。

应用：程序设计中数组的遍历与批量数据处理、字符串的增删改查，以及数组与字符串结合解决实际问题的能力。

第四章 函数与指针基础

一、学习目的与要求

（1）掌握函数的定义、声明与调用规则，理解实参与形参的匹配原则和参数传递方式。

（2）区分局部变量与全局变量的作用域，能避免变量重名冲突，理解变量的生命周期。

（3）掌握指针变量的定义、初始化与解引用操作，理解指针与变量、数组的关联关系。

（4）能进行模块化程序设计，将特定功能封装为函数，能运用指针完成简单的内存访问、数据修改与数组操作。

二、课程内容

（1）函数基础：函数的返回值类型、参数列表、函数体的书写规范，函数声明（原型）的位置与作用，无参函数与有参函数的调用。

（2）变量作用域：局部变量（函数内定义）与全局变量（函数外定义）的作用范围，变量的重名处理规则。

（3）指针定义与使用：指针变量的声明格式，指针指向变量 / 数组的语法，NULL 指针的概念，指针解引用操作（*）的使用。

（4）指针与数组：指针访问数组元素的方式（*(p+i) 等价于 arr [i]），数组名与指针的关系，通过指针遍历数组。

（5）模块化应用：自定义函数实现特定功能（求和、求最大值、数据统计等），函数的嵌套调用，指针作为函数参数的传递。

三、考核知识点及要求

1. 了解函数的封装原理、参数传递的底层逻辑，以及指针的内存寻址原理；
2. 理解函数定义 / 声明 / 调用的语法特点、局部 / 全局变量的作用域差异，以及指针与变量 / 数组的关联特性。

识记：根据程序需求设计合理的函数结构，规范声明指针变量，熟记函数参数传递（值传递）、指针访问数组的基础语法格式。

领会：展现出对函数执行流程、变量作用域与生命周期，以及指针解引用、数组名与指针等价性的准确理解。

应用：程序设计中函数的模块化封装与调用、指针对变量 / 数组的精准访问，以及利用指针作为函数参数实现数据传递的能力。

第五章 文件操作

一、学习目的与要求

(1) 掌握文件指针的定义方式，理解文件打开与关闭的基本流程，掌握 `fopen`、`fclose` 函数的使用规则。

(2) 熟练运用字符读写函数 (`fgetc`、`fputc`)、字符串读写函数 (`fgets`、`fputs`) 完成文件的基本读写操作。

(3) 能判断文件打开失败的情况，并进行简单的错误提示处理，避免未关闭文件导致的资源泄漏。

(4) 能实现将数据写入文件、从文件读取数据并进行简单处理，完成数据的持久化存储与读取。

二、课程内容

(1) 文件基础：文件指针的定义，文件的打开模式（只读 `r`、只写 `w`、追加 `a` 等）及适用场景。

(2) 文件的打开与关闭：`fopen` 函数的调用格式与返回值判断，`fclose` 函数的使用，文件打开失败的处理。

(3) 文件读写操作：字符读写函数 `fgetc/fputc`、字符串读写函数 `fgets/fputs` 的语法与使用方法。

(4) 文件操作流程：规范的“打开文件→执行读写操作→关闭文件”流程，数据的文件存储与读取实现。

三、考核知识点及要求

1. 了解文件的存储原理、文件指针的寻址原理，以及文件读写的底层逻辑；
2. 理解文件打开模式的特性、不同文件读写函数的适用场景，以及文件操作“打开 - 读写 - 关闭”流程的规范要求。

识记：根据程序需求选择合适的文件打开模式 (`r/w/a` 等)，规范定义文件指针，熟记 `fopen/fclose`、`fgetc/fputc`、`fgets/fputs` 等核心文件函数的功能与基础调用格式。

领会：展现出对文件打开失败的判断逻辑、不同读写函数的参数匹配规则，以及文件操作中资源保护（必关文件）、数据持久化的准确理解。

应用：程序设计中文件的规范打开与关闭、字符 / 字符串的文件读写实现，以及结合数组 / 函数完成文件数据的读取、处理与写入的能力。

三、参考教材与考核实施要求

（一）本课程使用的参考书

《计算机程序设计基础》，孙践知、肖媛媛、张迎新编著，机械工业出版社，2024 年版。

（二）本课程的考试要求

1. 考察学生的**基础编程实操能力**，能熟练使用指定 C 语言集成开发环境（Dev C++/VS2012/VS Code），遵循语法规则编写无基础语法错误的程序代码。
2. 考察学生的**程序补全与函数实现能力**，能根据程序上下文和功能要求补全缺失代码，能按指定需求实现自定义函数，保证程序完整、正常运行。
3. 考察学生的**综合程序设计能力**，能分析实际问题需求，设计基础算法，综合运用数据类型、控制结构、数组、函数、指针、文件等知识编写完整程序，实现指定功能，程序逻辑清晰、运行正确。
4. 考察学生的**程序调试与问题修正能力**，能根据程序运行结果识别并修正基础语法错误和简单逻辑错误，具备基本的程序调试思维和技巧。

（三）关于本课程考试命题的若干规定

1. 本门课程采用上机闭卷考试，考试场所为计算机机房，考试时间为 150 分钟，考场提供指定 C 语言集成开发环境。
2. 本大纲各章所规定的基本要求、知识点及知识细目，均属于考核内容。考试命题覆盖所有考核模块，突出课程重点，加大核心知识点（控制结构、数组、函数、指针、文件操作）的覆盖度。
3. 命题不超出大纲考核知识点范围，考核目标不高于大纲规定的最高能力层次要求。命题着重考核学生对基本编程操作的掌握、对结构化程序设计方法的运用，以及解决实际问题的编程实现能力，不出偏题、怪题。
4. 本课程在试卷中对不同能力层次要求的分数比例大致为：识记占 20%，领会占 30%，简单应用占 30%，综合应用占 20%。
5. 本门课程考试可选用的命题题型范围为单项选择题、填空题、程序填空题、程序阅读题、程序设计题。