教　案

2025-2026 学年第一学期

课程名称　C 语言程序设计

专业班级　机电一体化 3+证书 251

总学时数　48　学时

任课教师　刘小铭

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：课堂讲授 0 学时；实训/实验 48 学时；线上教学 0 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>刘小铭</td></tr>
<tr><td>授课
专业、班级</td><td colspan="2">机电一体化 3+证书 251 班</td><td>开课学期</td><td>2025-2026 第一学期</td></tr>
<tr><td>成绩评定</td><td colspan="2">平时成绩占 50%；期末成绩占 50%</td><td>考核方式</td><td>考试</td></tr>
<tr><td rowspan="2">选用教材</td><td>书　　名</td><td>主 编</td><td colspan="2">出版社</td><td>出版日期</td></tr>
<tr><td>C 程序设计</td><td>谭浩强</td><td colspan="2">清华大学出版社</td><td>2017.7</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="5">《C 语言程序设计》在机电一体化技术专业的人才培养方案中具有不可替代的地位和作用。它不仅是专业基础课程的核心内容，更是培养学生编程能力、逻辑思维、模块化设计思想和职业素养的重要环节。通过 C 语言编程课程的学习，学生能够为后续的专业核心课程和实践环节打下坚实的基础，同时为未来从事机电一体化技术相关工作提供重要的知识和技能储备。</td></tr>
<tr><td>本课程
教学目标</td><td colspan="5">通过教学，使学生较好地掌握 C 语言各方面的知识，掌握基本的程序设计过程和技巧，具备初步的高级语言程序设计能力，并能熟练应用 VC6.0 集成环境进行 C 语言的编与编译与调试，达到应用 C 语言解决一般编程问题的水平。</td></tr>
<tr><td>素质(思政)
内容与要求</td><td colspan="5">• 培养学生严谨的科学态度和良好的编程习惯，树立精益求精的工匠精神。
• 引导学生理解程序设计在社会发展中的重要作用，增强社会责任感和使命感。
• 通过案例分析和实践，培养学生的团队协作精神和创新意识。</td></tr>
<tr><td>学生用主要
参考资料</td><td colspan="5">C 语言程序设计　作者：谭浩强 清华大学出版社</td></tr>
</table>

认识 C 语言

教学目标与要求

- ✓ 什么是计算机程序及计算机语言
- ✓ C 语言的发展及其特点
- ✓ 最简单的 C 语言程序
- ✓ 运行 C 程序的步骤与方法

教学重点与难点

- ✓ 认识 DEV-C++：一个可视化集成开发环境，用此软件可以实现 C++程序的编辑、编译、运行和调试等工作。
- ✓ DEV-C++的安装与应用

教学课时： 3 课时

教学方式： 讲授、上机实践、实例讲解

思政目标： 通过介绍 C 语言的发展历程，引导学生了解计算机科学的发展对社会的推动作用，增强社会责任感。

实例内容：

1. 应用 DEV-C++新建源代码，运行课本第一章的 3 个例题，熟悉上机运行 C 程序的方法。

例 1.1：

```c
#include <stdio.h>                          // 这是编译预处理命令
int main( )                                 // 定义主函数
{                                           // 函数开始的标志
 printf ("This is a C program.\n");          // 输出所指定的一行信息
 return 0;                                  // 函数执行完毕时返回函数值0
}
```

例 1.2

```c
#include <stdio.h>                          // 这是编译预处理命令
int main( )                                 // 定义主函数
{                                           // 函数开始
  int a,b,sum;                              // 本行是程序的声明部分，定义a、b、sum为整型变量
  a = 123;                                  // 对变量a赋值
  b = 456;                                  // 对变量b赋值
  sum = a + b;                              // 进行a+b 的运算，并把结果存放在变量sum中
  printf("sum is %d\n",sum);                // 输出结果
  return 0;                                 // 使函数返回值为0
}                                           // 函数结束
```

例 1.3

```c
#include <stdio.h>
int main( )                          // 定义主函数
{                                    // 主函数体开始
 int max(int x,int y);               // 对被调用函数max的声明
 int a,b,c;                          // 定义变量a，b，c
 scanf("%d,%d",&a,&b);               // 输入变量a和b的值
 c = max(a,b);                       // 调用max函数，将得到的值赋给c
 printf("max=%d\n",c);               // 输出c的值
 return 0;                           // 返回函数值为0
}

int max(int x,int y)                 //定义max函数，函数值为整型，形式参数x和y为整型
{
 int z;                              // max函数中的声明部分，定义本函数中用到的变量z为整型
 if (x > y) z = x;
 else z = y;
 return(z);                          //将z的值返回，通过max带回到调用函数的位置
}
```

2. 参照课本例 1.1，编写一个 C 程序，输出以下信息（请将姓名和学号修改为自己的信息），运行结果如下图：

```c
#include <stdio.h>

int main() {
    printf("班级：机电一体化技术\n");
    printf("姓名：张三\n");
    printf("学号：19411111\n");
    return 0;
}
```

```
班级：机电一体化技术
姓名：张三
学号：18411111
```

3. 请参照本章例 1，编写一个 C 程序，输出以下信息:

```c
#include <stdio.h>
int main ( )
{ printf ("**************************\n\n");
  printf("       Very  Good!\n\n");
  printf ("**************************\n");
  return 0;
}
```

```
******************************
Very good!
******************************
```

4. 列出课本中关于 C 语言程序结构的 8 个特点。

(1) 一个程序由一个或多个源程序文件组成

(2) 函数是 C 程序的主要组成部分

(3) 程序总是从 main 函数开始执行

(4) 一个函数包括两个部分：声明部分与执行部分

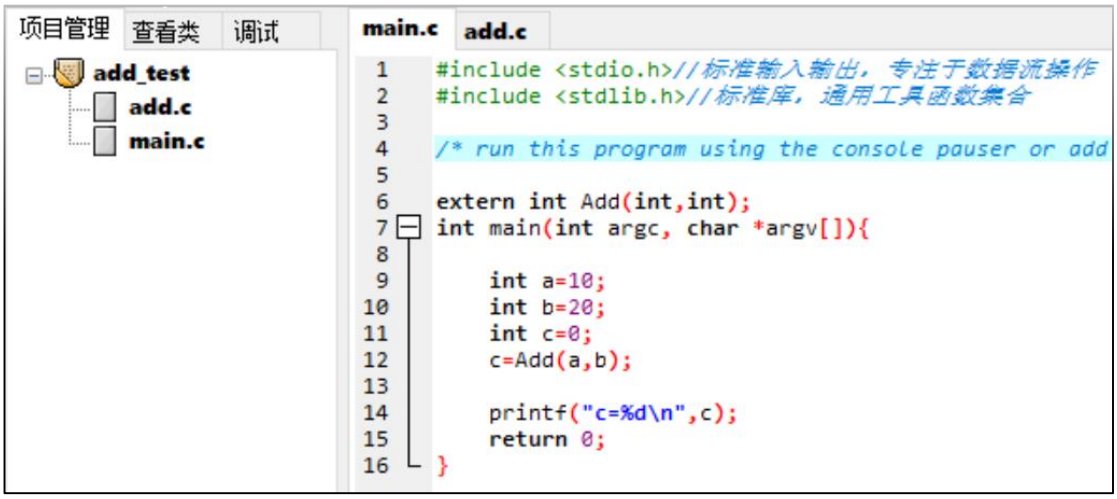
(5) C 程序对计算机的操作由 C 语句完成

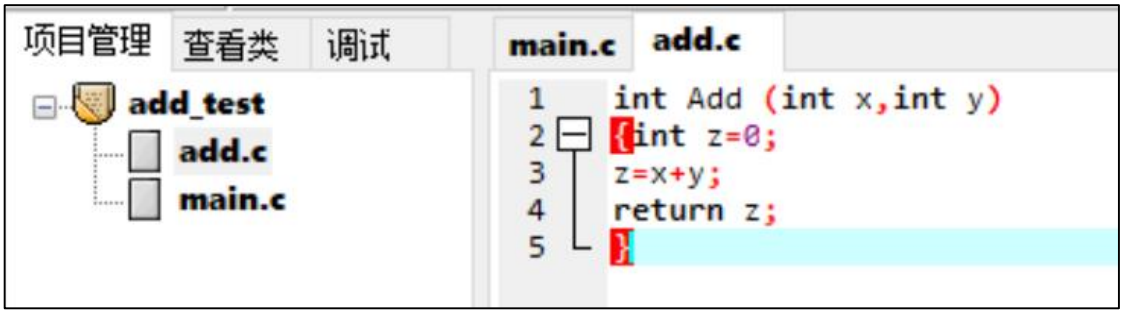
(6) 数据声明和语句最后必须有分号

(7) C 语言本身不提供输入输出语句

(8) 程序应当包含注释，增加可读性

5. （选做）应用 DEV-C++新建项目，用调动加法函数的方式实现例 1.2；注意项目文件的保存。对比新建源文件和新建项目的区别。





顺序程序设计

教学目标与要求

- ✓ 掌握 C 语言的基本输入输出函数（`printf`、`scanf`）及格式控制符的使用
- ✓ 掌握数据的表现形式：常量、变量、标识符
- ✓ 认识数据类型：整型、浮点型
- ✓ 熟练运用算术运算符（加、减、乘、除）和表达式解决数值计算问题

教学重点与难点

- ✓ 输入输出规范：`scanf`的格式控制（如`%d`、`%f`）与`printf`的精度控制（如`%.2f`）
- ✓ 运算与类型转换：运算与类型转换：不同数据类型的取值范围

教学课时：3 课时

教学方式：讲授、上机实践、实例讲解

思政目标：强调代码规范的重要性，强调数据类型选择的重要性，引导学生在编程中注重细节，培养严谨态度。

实例内容：

1. 运行课本实例 3.1，利用公式把华式温度转化为摄氏温度

```c
# include <stdio.h>
int main()
{float c,f;
printf("请输入华氏温度值:");
scanf("%f",&f);
c=(5.0/9)*(f-32);
printf("摄氏温度为%.2f\n",c);
return 0;
}
```

2. 运行**课本**实例 3.2，计算存款利息。

```c
#include <stdio.h>
int main( )
 {
   float p0=1000,r1=0.0036,r2=0.0225,r3=0.0198,p1,p2,p3;
   p1 = p0 * (1 +  r1);
   p2 = p0 * (1 +  r2);
   p3 = p0 * (1 +  r3/2) * (1 + r3/2);
   printf("p1=%f\np2=%f\np3=%f\n",p1,p2,p3);
   return 0;
 }
```

3. **计算两个数的和**

要求：输入两个数，输出它们的和（以下是两种方式，非连续输入与连续输入两个数）。

```c
#include <stdio.h>

int main() {
    // 定义两个变量用于存储输入的数字
    double num1, num2,sum;

    // 提示用户输入第一个数
    printf("请输入第一个数: ");
    scanf("%lf", &num1);

    // 提示用户输入第二个数
    printf("请输入第二个数: ");
    scanf("%lf", &num2);

    // 计算两个数的和
    sum = num1 + num2;

    // 输出结果
    //printf("您输入的数为: %.2lf 和 %.2lf\n", num1, num2);
    printf("它们的和为: %.2lf\n", sum);

    return 0;
}
```

```c
#include <stdio.h>

int main() {
    // 定义两个变量用于存储输入的数字
    double num1, num2,sum;

    // 提示用户输入两个数
    printf("请输入两个数（请用空格格开）: ");
    scanf("%lf %lf", &num1,&num2);
    // 计算两个数的和
    sum = num1 + num2;
    printf("它们的和为: %.2lf\n", sum);
    return 0;
}
```

4. **计算两个整数的差**

要求：输入两个整数，输出它们的差。

```c
#include <stdio.h>
int main() {
int a, b;
printf("输入两个整数: ");
scanf("%d %d", &a, &b);
printf("差: %d\n", a - b);
return 0;
}
```

5. **计算两个数的乘积**

要求：输入两个数，输出它们的乘积（保留两位小数）。

参考以上题目 1 或者题目 2 来写

6. **计算圆的周长与面积**

要求：输入圆的半径值，输出圆的周长与面积（保留两位小数），定义 PI 值为符号常量。

```c
#include <stdio.h>
#define  PI  3.14
int main()
{
    float r;
    printf("请输入圆有半径: r=");
    scanf("%f",&r);
    printf("周长=%.2f\n",2*PI*r);
    printf("面积=%.2f\n",PI*10*r);
    return 0;
}
```

运算符与表达式

教学目标与要求

- ✓ 认识字符型数据
- ✓ 掌握基本算术运算符
- ✓ 掌握算术表达式和运算符的优先级与结合性
- ✓ 掌握不同类型数据间的混合运算
- ✓ 理解自增、自减运算符的作用及其在表达式中的行为差异

教学重点与难点

- ✓ 算术表达式和运算符的优先级与结合性
- ✓ 不同类型数据间的混合运算
- ✓ 理解表达式中类型转换的隐式与显式规则
- ✓ 分析和调试常见表达式错误

教学课时：3 课时

教学方式：讲授、上机实践、实例讲解

思政目标：通过理解运算符优先级和数据类型转换规则，引导学生树立规则意识，养成一丝不苟的编程习惯；在解决混合运算问题时，培养系统思维和精益求精的精神，为成长为德才兼备的计算机专业人才奠定基础。

实例内容：

7. 运行课本实例 3.3，给定一个大写字母，要求用小写字母输出并输出其对应的 ASCII 代码值。

```c
#include <stdio.h>
int main ( )
{
  char c1,c2;
  c1='A';                          // 将字符'A'的ASCII代码放到c1变量中
  c2=c1+32;                        // 得到字符'a'的ASCII代码，放在c2变量中
  printf("%c\n",c2);               // 输出c2的值，是一个字符
  printf("%d\n",c2);               // 输出c2的值，是字符'a'的ASCII代码
  return 0;
}
```

8. 运行图中代码，了解字符常量的转义字符表示方法。通过字符型数据的转换实践，深入理解 ASCII 编码规律与字符间的关系。

```c
#include <stdio.h>

int main() {
    // 八进制转义字符示例
    printf("=== 八进制转义字符 (\\ddd) ===\n");
    char oct1 = '\101';     // 八进制 101 = 十进制 65 = 'A'
    char oct2 = '\141';     // 八进制 141 = 十进制 97 = 'a'
    char oct3 = '\60';      // 八进制 60 = 十进制 48 = '0'
    printf("八进制 101 (\\101) = %c (ASCII %d)\n", oct1, oct1);
    printf("八进制 141 (\\141) = %c (ASCII %d)\n", oct2, oct2);
    printf("八进制 60 (\\60) = %c (ASCII %d)\n\n", oct3, oct3);

    printf("=== 十六进制转义字符 (\\xhh) ===\n");
    char hex1 = '\x41';     // 十六进制 41 = 十进制 65 = 'A'
    char hex2 = '\x61';     // 十六进制 61 = 十进制 97 = 'a'
    char hex3 = '\x30';     // 十六进制 30 = 十进制 48 = '0'

    printf("十六进制 41 (\\x41) = %c (ASCII %d)\n", hex1, hex1);
    printf("十六进制 61 (\\x61) = %c (ASCII %d)\n", hex2, hex2);
    printf("十六进制 30 (\\x30) = %c (ASCII %d)\n", hex3, hex3);

    return 0;
}
```

9. **运行图中代码，**认识 C 语言中字符串操作函数 strcpy 与 strcat，掌握字符串复制与连接的基本用法。

```c
#include <stdio.h>
#include <string.h>  // 包含字符串处理函数

int main() {
    char name[50];      // 字符数组用于存储字符串
    char greeting[100]; // 问候语字符串

    // 输入字符串
    printf("请输入您的姓名: ");
    scanf("%s", name);

    // 创建问候语
   strcpy(greeting, "你好, ");//将源字符串复制到目标字符串
   strcat(greeting, name);//将源字符串连接到目标字符串的末尾
   strcat(greeting, "! 欢迎使用C语言程序。");

    // 输出结果
    printf("\n%s\n", greeting);
    printf("姓名长度: %zu 个字符\n", strlen(name));
    printf("问候语长度: %zu 个字符\n", strlen(greeting));
    return 0;
    /* %zu 是 printf 和 scanf 系列函数中的格式说明符，用于输出或输入 size_t 类型的数据。
    %: 格式说明符的开始;
    z: 表示长度修饰符，指定对应的参数是 size_t 类型;
    u: 表示无符号十进制整数
    size_t 类型:无符号整数类型，主要用于表示对象的大小和数量*/
}
```

10. 运行图中代码，掌握基本算术运算符及自增、自减运算符的运用。

```c
#include <stdio.h>
//算术运算符 +、 一、*(乘)、/(除)、%(求余)、++(自增)、--(自减)/和%
//除分为整除和除，整除结果取商舍弃余数。
//余只能是整数进行操作，结果取余舍弃商++和--(单目)前置：先运算再赋值后置：先赋值再运算
//强制类型转换符,如下语句中的（int）y
int main(){
    int a=10;
    int b=a/3;
    printf("%d\n",b);
    int z=a%b;
    printf("%d\n",z);

    float x=10;
    float y=x/3;
    printf("%f\n",y);
    printf("%d\n",y);
    printf("%d\n",(int)y);
    printf("%d\n",(int)(x/3));

    int agea=20;
    printf("%d\n",agea++) ;
    printf("%d\n",++agea);
    int ageb=20;
    printf("%d\n",ageb--) ;
    printf("%d\n",--ageb);
    return 0;}
```

11. 运行并理解以下代码，应用数学系统函数的方式求一个数的绝对值、算术平方根。

```c
#include <stdio.h>
#include <math.h>
int main()
{   double x;
    int y;
    double z;

    printf("请任意输入一个数求其绝对值: ");
    scanf("%lf", &x);
    printf("%lf\n",fabs(x));//返回双精度型参数的绝对值
    printf("请输入一个整数求其绝对值: ");
    scanf("%d", &y);
    printf("%d\n",abs(y));//返回整型参数的绝对值
    printf("请输入一个整数求其正平方根: ");
    scanf("%lf", &z);
    printf("%.2lf\n",sqrt(z));//返回整型参数的绝对值
}
```

了解常用数学系统函数

函　　数	说　　明
int abs(int i)	返回整型参数 i 的绝对值
double fabs(double x)	返回双精度参数 x 的绝对值
long labs(long n)	返回长整型参数 n 的绝对值
double exp(double x)	返回指数函数 e^x 的值
double log(double x)	返回 $\log_e x$ 的值
double log10(double x)	返回 $\log_{10} x$ 的值
double pow(double x, double y)	返回 x^y 的值
double sqrt(double x)	返回 x 正平方根的值
double acos(double x)	返回 x 的反余弦 $\cos^{-1}(x)$ 值
double asin(double x)	返回 x 的反正弦 $\sin^{-1}(x)$ 值
double atan(double x)	返回 x 的反正切 $\tan^{-1}(x)$ 值
double cos(double x)	返回 x 的余弦 cos(x) 值，x 为弧度数
double sin(double x)	返回 x 的正弦 sin(x) 值，x 为弧度数
double tan(double x)	返回 x 的正切 tan(x) 值，x 为弧度数

12. 运行图中代码，计算出每个表达式的值，了解运算符的优先级和结合性规则。

```c
#include <stdio.h>

int main() {
    int a = 10, b = 5, c = 3, d = 2;
    int result1, result2, result3, result4, result5;

    // 表达式1：混合运算
    result1 = a + b * c - d;

    // 表达式2：括号改变优先级
    result2 = (a + b) * (c - d);

    // 表达式3：结合性示例
    result3 = a - b - c;

    // 表达式4：除法和取模
    result4 = a * b / c % d;

    // 表达式5：复杂表达式
    result5 = a + b * c / d - a % c;

    printf("result1 = %d\n", result1);
    printf("result2 = %d\n", result2);
    printf("result3 = %d\n", result3);
    printf("result4 = %d\n", result4);
    printf("result5 = %d\n", result5);

    return 0;
}
```

13. 请参考上题中的内容编写代码，运行并输出 ex1,ex2,ex3 的结果。

```
int p = 6, q = 4, r = 2, s = 3;
int ex1, ex2, ex3;
ex1 = p * q + r / s - p % q;
ex2 = (p + q) * (r - s) / p;
ex3 = p - q + r - s;
```

数值输入输出及赋值语句

教学目标与要求

- ✓ 掌握输出函数：printf ,putchar,puts
- ✓ 掌握输入函数:scanf,getchar,fgets
- ✓ 掌握赋值运算符与复合赋值运算符

教学重点与难点

- ✓ 不同输入输出函数的区别和适用场景选择
- ✓ 复合赋值运算符的运算优先级理解
- ✓ 字符与整型数据的相互转换关系
- ✓ 程序调试和错误排查能力培养

教学课时：3 课时

教学方式：讲授、上机实践、实例讲解

思政目标：通过 C 语言输入输出及语句的学习，培养学生严谨细致的编程习惯和逻辑思维能力，在代码编写过程中树立精益求精的精神，增强解决实际问题的系统思维能力和创新意识。

实例内容：

1. 运行课本实例 3.5，求 $ax^2+bx+c=0$ 的方程的根。a,b,c 由键盘输入。

```c
#include <stdio.h>
#include  <math.h>                                  // 程序中要调用求平方根函数sqrt
int main ( )
 {double a,b,c,disc,x1,x2,p,q;                      // disc是判别式sqrt(b*b-4ac)
  printf("请从键盘输入abc三个数的值，用空格键隔开：\n");
  scanf("%lf%lf%lf",&a,&b,&c);                        // 输入实型变量的值要用格式声明"%f"
  disc=b*b-4*a*c;
  if (disc<0) printf("方程无实数根。\n");
  else
  { p=-b/(2.0*a);
    q=sqrt(disc)/(2.0*a);
    x1=p+q;x2=p-q;                                   // 求出方程的两个根
    printf("x1=%7.2f\nx2=%7.2f\n",x1,x2);            // 输出方程的两个根
  }
  return 0;
}
```

输入验算，如：a=1;b=-3;c=2;

x1= 2.00

x2= 1.00

2. 运行图中代码，掌握 printf、putchar 和 puts 三个输出函数的应用：

```c
#include <stdio.h>
#include <string.h>
int main() {
    // 定义一些变量用于演示
    int age = 20;
    float score = 85.5;
    char grade = 'A';
    char name[] = "张三";
    char message[] = "欢迎学习C语言！";
    char a[]="read book";

    printf("=== printf 函数演示 ===\n");
    // 1. printf - 格式输出函数（最常用）
    printf("1. 基本输出:\n");
    printf("姓名: %s, 年龄: %d\n", name, age);
    printf("成绩: %.1f, 等级: %c\n", score, grade);

    printf("\n2. 格式控制:\n");
    printf("十进制: %d, 八进制: %#o, 十六进制: %#x\n", age, age, age);
    printf("浮点数: %f, 科学计数法: %e\n", score, score);
    printf("字符串: %s, 字符: %c\n", message, grade);

    printf("\n3. 宽度和对齐:\n");
    printf ("%8.4s\n",a);
    printf ("%.4s\n",a);
    // ========================================
    printf("\n=== 综合应用示例 ===\n");
    // 综合使用三种输出函数
    printf("学生信息:\n");
    printf("姓名: ");
    puts(name);
    printf("年龄: %d\n", age);
    printf("成绩信息: ");
    putchar(grade);
    printf(" (%.1f分)\n", score);
    printf("评语: ");
    puts(message);
    // 使用puts输出分割线
    puts("================================");
    printf("程序演示结束！\n");
    return 0;
}
```

总结：

printf - 最强大灵活；支持多种格式控制；可以输出各种数据类型；不会自动换行

putchar - 最简单；一次只能输出一个字符；效率较高；不会自动换行

puts - 最方便；专门用于输出字符串；自动在结尾添加换行符使用简单，不需要格式说明符

3. 运行图中代码，掌握输入函数 scanf、getchar 及输出函数的综合应用

```c
#include <stdio.h>
int main() {
    // 变量定义
    char studentName[30];
    int studentAge;
    float mathScore;
    char mathGrade;
    char confirm;

    printf("=== 学生数学成绩录入系统 ===\n");  // 1. 使用 printf 输出程序标题
    puts("--------------------------------");   // 2. 使用 puts 输出分隔线
    printf("请输入学生姓名: ");                 // 3. 使用 printf 输出提示信息
    scanf("%s", studentName);                  // 4. 使用 scanf 输入学生姓名
    printf("请输入学生年龄: ");                   // 5. 使用 printf 输出提示信息
    scanf("%d", &studentAge);                   // 6. 使用 scanf 输入学生年龄
    printf("请输入数学成绩: ");                 // 7. 使用 printf 输出提示信息
    scanf("%f", &mathScore);                   // 8. 使用 scanf 输入数学成绩
    getchar();                                  // 9. 清理输入缓冲区
    printf("请输入数学等级 (A/B/C/D/F): ");     // 10. 使用 printf 输出提示信息
    mathGrade = getchar();                     // 11. 使用 getchar 输入数学等级
    getchar();                                 // 12. 清理输入缓冲区
    puts("");                                  // 13. 使用 puts 输出空行

    printf("学生信息汇总:\n");                  // 14. 使用 printf 输出汇总标题
    puts("====================");              // 15. 使用 puts 输出分隔线
    printf("姓名: %s\n", studentName);         // 16. 使用 printf 输出学生姓名
    printf("年龄: %d\n", studentAge);          // 17. 使用 printf 输出学生年龄
    printf("数学成绩: %.2f\n", mathScore);     // 18. 使用 printf 输出数学成绩
    printf("数学等级: %c\n", mathGrade);       // 19. 使用 printf 输出数学等级
    puts("----------------------");            // 20. 使用 puts 输出分隔线

    printf("请确认信息是否正确 (Y/N): ");       // 21. 使用 printf 输出确认提示
    confirm = getchar();                       //22. 使用 getchar 输入确认信息
    getchar();                                // 23. 清理输入缓冲区
    puts("");                                 // 24. 使用 puts 输出空行
    printf("信息已保存到系统! \n");             //25. 使用 printf 输出结果信息

    puts("");
    puts("*** 谢谢使用 ***");

    return 0;
}
```

getchar（） 函数功能：

从终端（键盘）输入一个字符，以回车确认。函数的返回值就是输入的字符。

清理输入缓冲区:读取并丢弃缓冲区中的残留字符,防止后续输入操作被意外影响,确保程序能够正常接收用户输入,提高程序的健壮性（当使用 scanf 等函数输入时，用户输入的内容会先存储在输入缓冲区中。如果输入的内容与预期不符，或者有残留字符，可能会影响后续的输入操作）

4. 运行图中代码，掌握赋值运算符、混合赋值运算符及逗号表达式的应用

```c
#include<stdio.h>
int main() {
    int a, b, c, d, e;
    printf("=== 赋值运算符演示 ===\n");// 1. 基本赋值运算符
    a = 10;
    b = 20;
    printf("初始值: a = %d, b = %d\n", a, b);

    // 2. 链式赋值表达式
    c = d = e = 5;  // 从右到左执行: e=5, d=e, c=d
    printf("链式赋值: c = %d, d = %d, e = %d\n", c, d, e);

    // 3. 混合赋值运算符
    printf("\n=== 混合赋值运算符演示 ===\n");
    a += b;  // 等价于 a = a + b
    printf("a += b 后: a = %d\n", a);
    b -= 5;  // 等价于 b = b - 5
    printf("b -= 5 后: b = %d\n", b);
    c *= 2;  // 等价于 c = c * 2
    printf("c *= 2 后: c = %d\n", c);
    d /= 2;  // 等价于 d = d / 2
    printf("d /= 2 后: d = %d\n", d);
    e %= 3;  // 等价于 e = e % 3
    printf("e %%= 3 后: e = %d\n", e);

    // 4. 逗号表达式
    printf("\n=== 逗号表达式演示 ===\n");
    int x, y, z;  // 逗号表达式: 从左到右计算，返回最后一个表达式的值
    z = (x = 15, y = 25, x + y);
    printf("逗号表达式 z = (x=15, y=25, x+y): z = %d\n", z);
    printf("x = %d, y = %d\n", x, y);
    printf("\n=== 综合应用 ===\n");
    int m = 8, n = 12;
    int result;
```

```c
    // 5.使用赋值表达式和逗号表达式
    result = (m += 2, n -= 3, m * n);
    printf("(m += 2, n -= 3, m * n) = %d\n", result);
    printf("现在 m = %d, n = %d\n", m, n);

    // 6. 在printf中使用赋值表达式
    printf("\n=== 在printf中使用赋值表达式 ===\n");
    int p = 5, q = 8;
    printf("赋值前: p = %d, q = %d\n", p, q);
    printf("p = q 的结果: %d\n", p = q);  // 执行赋值并输出结果
    printf("赋值后: p = %d, q = %d\n", p, q);

    return 0;
}
```

5. 第三章节课后习题第 6 题，请编程序将“China”译成密码，密码规律是：用原来的字母后面第 4 个字母代替原来的字母。例如，字母“A”后面第 4 个字母是“E”,用“E”代替“A”。因此，“China”应译为“Glmre”。请编一程序，用赋初值的方法使 cl,c2,c3,c4,c5 这 5 个变量的值分别为'C’,’h’,'i','n’,'a',经过运算，使 c1,c2,c3,c4,c5 分别变为’G’,'I','m','r','e’。分别用 putchar 函数和 printf 函数输出这 5 个字符。

```c
#include <stdio.h>
int main()
{char c1='C',c2='h',c3='i',c4='n',c5='a';
 c1=c1+4;
 c2=c2+4;
 c3=c3+4;
 c4=c4+4;
 c5=c5+4;
 printf("passwor is %c%c%c%c%c\n",c1,c2,c3,c4,c5);
 return 0;
}
```

6. 完第第三章第 7 题：设圆半径 r=1.5,圆柱高 h=3,求圆周长、圆面积、圆球表面积、圆球体积、圆柱体积。用 scanf 输入数据，输出计算结果，输出时要求有文字说明，取小数点后 2 位数字。请编程序。

```c
#include <stdio.h>
int main ()
{float h,r,l,s,sq,vq,vz;
 float pi=3.141526;
 printf("请输入圆半径r. 圆柱高h: ");
 scanf("%f,%f",&r,&h);                    //要求输入圆半径r和圆柱高h
 l=2*pi*r;                                //计算圆周长l
 s=r*r*pi;                                //计算圆面积s
 sq=4*pi*r*r;                             //计算圆球表面积sq
 vq=3.0/4.0*pi*r*r*r;                     //计算圆球体积vq
 vz=pi*r*r*h;                             //计算圆柱体积vz
 printf("圆周长为:      l=%6.2f\n",l);
 printf("圆面积为:      s=%6.2f\n",s);
 printf("圆球表面积为:  sq=%6.2f\n",sq);
 printf("圆球体积为:    v=%6.2f\n",vq);
 printf("圆柱体积为:    vz=%6.2f\n",vz);
 return 0;
 }
```

7. 完成第三章节第 8 题：编程序，用 getchar 函数读入两个字符给 c1 和 c2,然后分别用 putchar 函数和 printf 函数输出这两个字符。思考以下问题：

(1)变量 c1 和 c2 应定义为字符型还是整型?或二者皆可?

(2)要求输出 c1 和 c2 值的 ASCII 码，应如何处理?用 putchar 函数还是 printf 函数?

```c
#include <stdio.h>
int main()
{
char c1,c2;
printf("请输入两个字符c1,c2:");
c1=getchar();
c2=getchar();
printf("用putchar语句输出结果为:");
putchar(c1);
putchar(c2);
printf("\n");
printf("用printf语句输出结果为:");
printf("%c %c\n",c1,c2);
return 0;
}
```

(3)整型变量与字符变量是否在任何情况下都可以互相代替?如：

char c1,c2;

与

int c1,c2;

是否无条件地等价?

(1) c1 和 c2 可以定义为字符型或整型，二者皆可。

(2) 可以用 printf 函数输出，在 printf 函数中用%d 格式符，即：

```
printf("%d,%d\n",c1,c2);
```

(3) 字符变量在计算机内占 1 个字节，而整型变量占 2 个或 4 个字节。因此整型变量在可输出字符的范围内(ASCII 码为 0～127 之间的字符)是可以与字符数据互相转换的。如果整数在此范围外，不能代替。

为了进一步说明 char 型与 int 型数据的关系，请注意分析以下 3 个程序：

程序 1：

```
#include <stdio.h>
int main( )
{
   int c1,c2;                          //定义整型变量 c1,c2
   printf("请输入两个整数 c1,c2:");
   scanf("%d,%d",&c1,&c2);
   printf("按字符输入结果:\n");
   printf("%c,%c\n",c1,c2);
   printf("按 ASCII 码输入出结果为:\n");
   printf("%d,%d\n",c1,c2);
   return 0;
}
```

程序 2：

```
#include <stdio.h>
int main( )
{
  char c1,c2;                  //c1,c2 定义为字符型变量
  int i1,i2;                   //定义整型变量
  printf("请输入两个字符 c1,c2:");
  scanf("%c,%c",&c1,&c2);
  i1=c1;                       //赋值给整型变量
  i2=c2;
  printf("按字符输入结果:\n");
  printf("%c,%c\n",i1,i2);
  printf("按整数输入出结果:\n");
  printf("%d,%d\n",c1,c2);
  return 0;
}
```

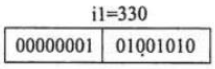
```
int main( )
{
  char c1,c2;                    //c1,c2 定义为字符型
  int i1,i2;                     //i1,i2 定义为整型
  printf("请输入两个整数 i1,i2:");
  scanf("%d,%d",&i1,&i2);
  c1=i1;                         //将整数赋值给字符变量
  c2=i2;
  printf("按字符输入结果:\n");
  printf("%c,%c\n",c1,c2);
  printf("按整数输入出结果:\n");
  printf("%d,%d\n",c1,c2);
  return 0;
}
```

请注意 i,i1 和 i2 占 2 个或 4 个字节(Visual C++ 对它分配 4 个字节),而 c1 和 c2 是字符变量,只占 1 个字节,如果是 unsigned char 类型,可以存放 0～255 范围内的整数,如果是 signed char 类型,可以存放－128～127 范围内的整数。而现在输入给 i1 和 i2 的值已超过 0～255 的范围,i1 的值为 289,在内存中 i1 的存储情况如图 3.1(a) 所示(为简单起见,用 2 个字节表示),在赋给字符变量 c1 时,只将其存储单元中最后一个字节(低 8 位)赋给 c1,见图 3.1(b)。而图 3.1(b)中的数据是整数 33,是字符'!'的 ASCII 代码,所以用字符形式输出 c1 时,会输出字符'!'。图 3.2 表示 i2 和 c2 的情况,c2 的值为 74,是字符'J'的 ASCII 码,因此。按字符形式输出 c2 时就输出字符'j'。

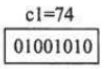
(a)

(b)

图 3.1



(a)



(b)

图 3.2

关系运算符、逻辑运算符与 IF 语句

教学目标与要求

- ✓ 掌握关系运算符和逻辑运算符的使用方法
- ✓ 理解关系表达式和逻辑表达式的概念与求值规则
- ✓ 掌握 if 语句的三种基本结构：单分支、双分支和多分支
- ✓ 能够正确使用关系运算符和逻辑运算符构建表达式
- ✓ 能够运用 if 语句解决实际的选择结构问题

教学重点与难点

- ✓ 关系运算符和逻辑运算符的优先级区分
- ✓ 复杂逻辑表达式的分析与求值
- ✓ if 语句嵌套使用时的 else 配对问题
- ✓ 逻辑运算符短路特性的理解与应用
- ✓ 多分支 if-else if 结构的正确使用

教学课时：3 课时

教学方式：讲授、上机实践、实例讲解

思政目标：通过 C 语言关系运算符、逻辑运算符和 if 语句的学习，培养学生严谨求实的科学态度，在条件判断中培养精确思维，避免逻辑错误；系统化的问题分析能力，通过多分支结构学习全面思考问题的方法；逻辑思维与判断能力，通过条件判断培养在实际生活中的决策能力

实例内容：

1. 运行图中实例，掌握关系运算符与逻辑运算符的应用

```c
#include <stdio.h>

int main() {
    int a = 10, b = 20, c = 30;
    int result1, result2, result3, result4, result5;
    printf("=== 关系表达式和逻辑表达式计算演示 ===\n\n");
    // 初始值显示
    printf("初始值: a = %d, b = %d, c = %d\n\n", a, b, c);

    // 1. 关系表达式计算
    printf("1. 关系表达式计算:\n");
    result1 = a > b;
    printf("a > b = %d\n", result1);

    result2 = b <= c;
    printf("b <= c = %d\n", result2);

    result3 = a == 10;
    printf("a == 10 = %d\n", result3);

    result4 = c != 25;
    printf("c != 25 = %d\n", result4);
    printf("\n");
```

```c
    // 2. 逻辑表达式计算
    printf("2. 逻辑表达式计算:\n");
    result1 = (a > 5) && (b < 25);
    printf("(a > 5) && (b < 25) = %d\n", result1);

    result2 = (a >15) || (c == 30);
    printf("(a > 15) || (c == 30) = %d\n", result2);

    result3 = !(a == 10);
    printf("!(a == 10) = %d\n", result3);

    result4 = (a < b) && (b < c);
    printf("(a < b) && (b < c) = %d\n", result4);

    result5 = (a > b) || (c < 40);
    printf("(a > b) || (c < 40) = %d\n", result5);
    printf("\n");

    // 3. 复合逻辑表达式
    printf("3. 复合逻辑表达式:\n");
    int x = 5, y = 15, z = 25;
    printf("新变量: x = %d, y = %d, z = %d\n", x, y, z);

    result1 = (x < y) && (y < z) && (z > 20);
    printf("(x < y) && (y < z) && (z > 20) = %d\n", result1);

    result2 = (x > 10) || (y == 15) || (z < 20);
    printf("(x > 10) || (y == 15) || (z < 20) = %d\n", result2);

    result3 = !(x == 5) && (y != 15);
    printf("!(x == 5) && (y != 15) = %d\n", result3);

    result4 = (x + y > z) || (z - y < x);
    printf("(x + y > z) || (z - y < x) = %d\n", result4);
    printf("\n");



    return 0;
}
```

2. 运行课本实例 4.2，输入两个实数，按由小到大的顺序输出这两个数。

```c
#include <stdio.h>
int main()
{
    float a,b,t;
    scanf("%f,%f",&a,&b);
    if(a>b)
     {
       t=a;
       a=b;
       b=t;
     }
    printf("%5.2f,%5.2f\n",a,b);
    return 0;
}
```

3. 运行课本实例 4.3，输入三个数 a,b,c ,要求按由小到大的顺序输出这两个数

```c
#include <stdio.h>
int main()
{
    float a,b,c,t;
    scanf("%f,%f,%f",&a,&b,&c);
    if(a>b)
    {
        t=a;
        a=b;
        b=t;
    }                                   // 实现a和b的互换
    if(a>c)
    {
        t=a;
        a=c;
        c=t;
    }                                   // 实现a和c的互换
    if(b>c)
    {
        t=b;
        b=c;
        c=t;
    }                                   // 实现b和c的互换
    printf("%5.2f,%5.2f,%5.2f\n",a,b,c);
    return 0;
}
```

4. 运用下图中多分支语句，把学生的成绩分成优秀、良好、中等、合格和不合格 5 个等级。

```c
#include<stdio.h>
int main()
{
        int score;
        scanf("%d",&score);
        if(score<0||score>100)
                printf("成绩输入有误！\n");
        else if(score<60)
                printf("不合格\n");
                else if(score<70)
                printf("合格\n");
        else if(score<80)
                printf("中等\n");
        else  if(score<90)
                 printf("良好\n");
        else
                 printf("优秀\n");
        return 0;
}
```

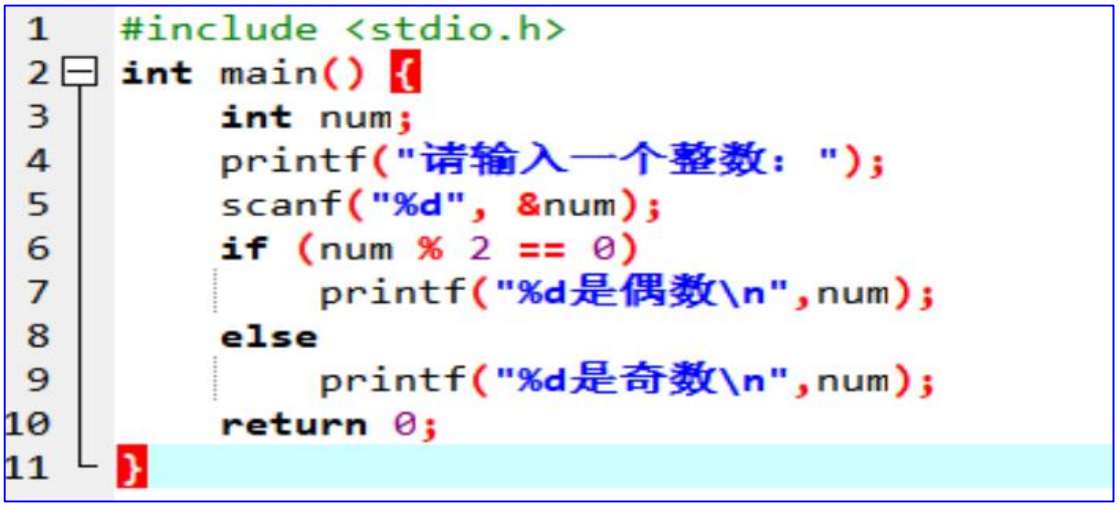
5. 运用下图中多分支语句，对比 IF 语句的用法

```c
#include <stdio.h>

int main() {
    int score;
    scanf("%d",&score);

    // 多个独立的if语句
    if (score >= 90) {
        printf("优秀\n");
    }

    if (score >= 80 && score < 90) {
        printf("良好\n");
    }

    if (score >= 70 && score < 80) {
        printf("中等\n");
    }

    if (score >= 60 && score < 70) {
        printf("及格\n");
    }

    if (score < 60) {
        printf("不及格\n");
    }

    return 0;
}
```

6. 根据本次课所学内容编写程序判断闰年：输入一个年份，判断它是否为闰年。

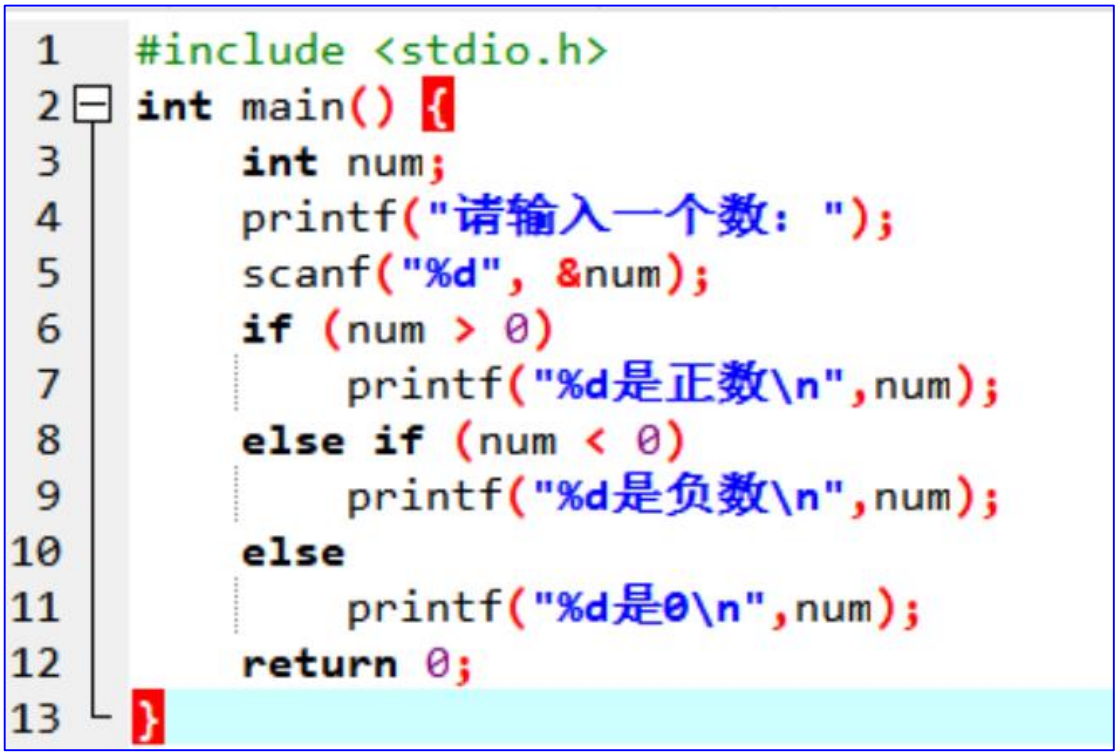
```c
#include <stdio.h>
int main() {
    int year;
    printf("Enter a year: ");
    scanf("%d", &year);
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
        printf("%d年是闰年\n",year);
    else
        printf("%d年不是闰年\n",year);
    return 0;
}
```

7. 根据本次课所学内容编写程序判断奇偶数：输入一个整数，判断它是奇数还是偶数。



```c
#include <stdio.h>
int main() {
    int num;
    printf("请输入一个整数: ");
    scanf("%d", &num);
    if (num % 2 == 0)
        printf("%d是偶数\n",num);
    else
        printf("%d是奇数\n",num);
    return 0;
}
```

8. 根据本次课所学内容编写程序判断正负数:输入一个整数,判断它是正数、负数还是零。



```c
#include <stdio.h>
int main() {
    int num;
    printf("请输入一个数: ");
    scanf("%d", &num);
    if (num > 0)
        printf("%d是正数\n",num);
    else if (num < 0)
        printf("%d是负数\n",num);
    else
        printf("%d是0\n",num);
    return 0;
}
```

switch 分支结构应用

教学目标与要求

- ✓ 理解条件运算符的语法格式和执行过程
- ✓ 掌握 switch 语句的基本结构和执行流程
- ✓ 根据实际问题选择合适的分支结构

教学重点与难点

- ✓ 条件运算符的优先级和结合性
- ✓ switch 语句中 break 语句的正确使用
- ✓ if-else 语句与 switch 语句的选择与应用场景
- ✓ 能够编写包含 switch 语句的完整程序

教学课时：3 课时

教学方式：讲授、上机实践、实例讲解

思政目标：通过 C 语言条件判断的教学，引导学生树立正确的价值观，明辨是非通过程序逻辑的严谨性，培养学生一丝不苟的工作态度；通过解决实际问题的编程练习，增强学生的社会责任感和创新精神。

实例内容：

1. 运行图中实例，掌握嵌套 IF 语句及条件运算符的应用，判断成绩等级

```c
#include <stdio.h>
int main() {
    int score;
    char grade;

    printf("=== 应用IF语句与条件运算符判断成绩等级===\n");
    printf("1. 成绩等级判断:\n");
    printf("请输入学生成绩(0-100): ");
    scanf("%d", &score);

    // 在IF条件中使用条件运算符
    if ((score >= 0 && score <= 100) ? 1 : 0) {
        // 使用条件运算符确定等级
        grade = (score >= 90) ? 'A' :
                (score >= 80) ? 'B' :
                (score >= 70) ? 'C' :
                (score >= 60) ? 'D' : 'F';

        printf("成绩等级: %c\n", grade);

        // 在IF条件中嵌套条件运算符
        if ((grade == 'A' || grade == 'B') ? printf("优秀学生! ") : 0) {
            printf("继续保持!\n");
        }
    } else {
        printf("成绩输入无效!\n");
    }
    printf("\n");
    return 0;
}
```

2. 运行图中实例，掌握在 IF 条件中应用条件运算符，比较数字大小与奇偶判断

```c
#include <stdio.h>

int main() {
    int num1, num2;
    // 数字比较与奇偶判断
    printf("数字比较与奇偶判断:\n");
    printf("请输入两个整数: ");
    scanf("%d %d", &num1, &num2);

    // 使用条件运算符确定较大值和较小值
    int max = (num1 > num2) ? num1 : num2;
    int min = (num1 < num2) ? num1 : num2;

    printf("较大值: %d, 较小值: %d\n", max, min);

    // 在IF条件中使用条件运算符进行复合判断
    if (((num1 % 2 == 0) ? "偶数" : "奇数") &&
        ((num2 % 2 == 0) ? "偶数" : "奇数")) {
        printf("两个数字的奇偶性已检查\n");
    }

    // 更复杂的条件判断
    if ((num1 > num2 ? num1 - num2 : num2 - num1) > 10) {
        printf("两个数字的差值大于10\n");
    } else {
        printf("两个数字的差值不大于10\n");
    }
    printf("\n");
```

3. 运行图中实例，掌握条件运算符在输出函数中的应用，判断天气情况

```c
#include <stdio.h>

int main() {
    //条件运算符在输出中的应用
    printf("条件运算符在输出中的应用:\n");
    int temperature;
    printf("请输入当前温度: ");
    scanf("%d", &temperature);

    printf("天气状况: %s\n",
           (temperature > 30) ? "炎热" :
           (temperature > 20) ? "温暖" :
           (temperature > 10) ? "凉爽" : "寒冷");

    // 在IF条件中使用复杂的三元表达式
    if ((temperature > 0 ? "高于零度" : "低于或等于零度") &&
        (temperature < 100 ? "低于沸点" : "达到或超过沸点")) {
        printf("温度在常规范围内\n");
    }

    return 0;
}
```

4. 运行课本实例 4-10，根据要求计算运输费用。

例 4.10 运输公司对用户计算运输费用。路程(*s*km)越远,每吨·千米运费越低。标

准如下：

$s<250$	没有折扣
$250\leqslant s<500$	2%折扣
$500\leqslant s<1000$	5%折扣
$1000\leqslant s<2000$	8%折扣
$2000\leqslant s<3000$	10%折扣
$3000\leqslant s$	15%折扣

解题思路：设每吨每千米货物的基本运费为 p(price 的缩写),货物重为 w(weight 的缩写),距离为 s,折扣为 d(discount 的缩写),则总运费 f(freight 的缩写)的计算公式为 $f=p\times w\times s\times(1-d)$。

经过仔细分析发现折扣的变化是有规律的：从图 4.15 可以看到,折扣的“变化点”都是 250 的倍数(250,500,1000,2000,3000)。利用这一特点,可以在横轴上加一种坐标 c,c 的值为 $s/250$。c 代表 250 的倍数。当 $c<1$ 时,表示 $s<250$,无折扣;$1\leqslant c<2$ 时,表示 $250\leqslant s<500$,折扣 $d=2\%$;$2\leqslant c<4$ 时,$d=5\%$;$4\leqslant c<8$ 时,$d=8\%$;$8\leqslant c<12$ 时,$d=10\%$;$c\geqslant 12$ 时,$d=15\%$。

```c
#include <stdio.h>
int main()
{
   int c,s;
   float p,w,d,f;
   printf("please enter price,weight,discount:");      // 提示输入的数据
   scanf("%f,%f,%d",&p,&w,&s);                         // 输入单价、重量、距离
   if(s>=3000)  c=12;                                  // 3000km以上为同一折扣
   else         c=s/250;                               // 3000km以下各段折扣不同，c的值不相同
   switch(c)
   {
     case 0:   d=0; break;                             // c=0,代表250km以下,折扣d=0
     case 1:   d=2; break;                             // c=1,代表250到500km以下,折扣d=2%
     case 2:
     case 3:   d=5; break;                             // c=2和3,代表500到1000km以下,折扣d=5%
     case 4:
     case 5:
     case 6:
     case 7:   d=8; break;                             // c=4-7,代表1000到2000km以下,折扣d=8%
     case 8:
     case 9:
     case 10:
     case 11:  d=10; break;                            // c=8-11,代表2000到3000km以下,折扣d=10%
     case 12:  d=15; break;                            // c12,代表3000km以上,折扣d=15%
   }
   f = p * w * s * (1 - d / 100);                      // 计算总运费
   printf("freight=%10.2f\n",f);                       // 输出总运费，取两位小数
   return 0;
}
```

5. 思考并编程（课后第 8 题）：给出一百分制成绩，要求输出成绩等级'A’、'B’、'C’、'D’、'E’。90 分以上为'A',80～89 分为'B’,70～70 分为'C’,60～69 分为’D’,60 分以下为'E'。

```c
#include <stdio.h>
int main()
  { float score;
    char grade;
    printf("请输入学生成绩:");
    scanf("%f",&score);
    while (score>100||score<0)
    {printf("\n 输入有误,请重输");
    scanf("%f",&score);
    }
    switch((int)(score/10))
       {case 10:
    case 9: grade='A';break;
    case 8: grade='B';break;
    case 7: grade='C';break;
    case 6: grade='D';break;
    case 5:
    case 4:
    case 3:
    case 2:
    case 1:
    case 0: grade='E';
    }
    printf("成绩是 %5.1f,相应的等级是%c\n ",score,grade);
    return 0;
}
```

6. 思考并编程（课后第 10 题）：企业发放的奖金根据利润提成。利润 I 低于或等于 100000 元的，奖金可提 10%;利润高于 100000 元，低于 200000 元(100000<I≤200000)时，低于 100000 元的部分按 10%提成，高于 100000 元的部分，可提成 7.5%;200000<I≤400000 时，低于 200000 元的部分仍按上述办法提成(下同)。高于 200000 元的部分按 5%提成；400000<I≤600000 元时，高于 400000 元的部分按 3%提成；600000<I≤1000000 时，高于 600000 元的部分按 1.5%提成；I>1000000 时，超过 1000000 元的部分按 1%提成。从键盘输入当月利润 I,求应发奖金总数。要求：

把如下 IF 语句的程序改为 **switch 语句编程**。

```
#include <stdio.h>
int main()
{
  int i;
  double bonus,bon1,bon2,bon4,bon6,bon10;
  bon1=100000*0.1;
  bon2=bon1+100000*0.075;
  bon4=bon2+100000*0.05;
  bon6=bon4+100000*0.03;
  bon10=bon6+400000*0.015;
  printf("请输入利润i:");
  scanf("%d",&i);
  if (i<=100000)
     bonus=i*0.1;
  else if (i<=200000)
     bonus=bon1+(i-100000)*0.075;
  else if (i<=400000)
     bonus=bon2+(i-200000)*0.05;
  else if (i<=600000)
     bonus=bon4+(i-400000)*0.03;
  else if (i<=1000000)
     bonus=bon6+(i-600000)*0.015;
  else
     bonus=bon10+(i-1000000)*0.01;
  printf("奖金是: %10.2f\n",bonus);
  return 0;
 }
```

循环结构程序设计（一）

教学目标与要求

- ✓ 掌握 while 循环语句的基本语法和执行流程
- ✓ 理解 do-while 循环与 while 循环的区别
- ✓ 掌握 for 循环语句的基本语法和执行流程

教学重点与难点

- ✓ 三种循环结构(while、do-while、for)的语法和执行流程
- ✓ 循环条件的正确设置，避免死循环
- ✓ 循环变量初始化和更新的时机
- ✓ 不同循环结构的选择和应用场景

教学课时：3 课时

教学方式：讲授、上机实践、实例讲解

- **思政目标：**通过 C 语言循环结构的学习，培养学生"生命不息，奋斗不止"的积极人生态度;通过解决实际问题，增强学生服务社会、解决问题的责任感,培养学生严谨求实的科学态度和耐心细致的工作作风。

实例内容：

1. 运行课本实例 5.1，掌握 while 循环语句的基本语法与执行流程。编程计算 1+2+3...+100 的和。

```
#include<stdio.h>
int main()
{
int i=1,sum=0;
while(i<=100)
{
sum=sum+i;
i++;
}
printf("1+2+...+100的和:sum=%d\n",sum);return 0;
}
```

2. 运行课本实例 5.2，掌握 do...while 循环语句的基本语法与执行流程。编程计算 1+2+3...+100 的和。

```c
#include <stdio.h>
int main()
{   int i=1,sum=0;
    do
     {
        sum=sum+i;
        i++;
     }while(i<=100);
     printf("1+2+...+100的和:sum=%d\n",sum);
     return 0;
}
```

3. 改写下图中的代码，应用 while 循环编写一个程序实现：从键盘输入一个正整数并计算它的阶乘。

```c
#include<stdio.h>
int main()
{
int i=1,p=1;
while(i<=10)
{
p=p*i;
i++;
}
printf("1*2*3...*10的积:p=%d\n",p);return 0;
}
```

```c
#include<stdio.h>
int main()
{
    int i = 1,  n;
   long long p = 1;  // 使用long long类型
    printf("请输入要求阶乘的数(正整数): ");
    scanf("%d", &n);

    // 计算阶乘
    while(i <= n)
    {
        p = p * i;  // 累积相乘
        i++;
    }

    printf("%d! = %lld\n", n, p); // 使用%lld格式输出
    return 0;
}
```

4. 运行以下两图的代码，修改循环变量的初始值，或修改循环的条件（如 count=5;count>=5 等），对比 while 与 do...while 的输出结果，理解两种结构的执行流程。

```c
int main()
{// 初始化计数器
int count = 5;

// while循环
while (count <5) {
    printf("当前计数器的值为%d\n", count);
    count++;
}
printf("循环结束，count = %d", count);
return 0;
}
```

```c
#include<stdio.h>
int main()
{int count = 5;

// do...while循环
do {
    printf("当前计数器的值为%d\n", count);
    count++;
} while (count <5);
printf("循环结束，count = %d", count);
return 0;
}
```

while 循环：先判断条件，如果条件为假，则循环体一次也不执行。

do...while 循环：先执行循环体，再判断条件，即使条件为假，循环体也至少执行一次。

5. 运行图中代码，掌握 for 循环语句的基本语法和执行流程。计算 1+2+3+...100 的和。

```c
#include <stdio.h>
int main()
{   int i=1,sum=0;
    for (i=1;i<=100;i++)
{   sum=sum+i;
     printf("%d ", i );
     printf("sum=%d\n",sum);
}

     return 0;
}
```

6. 运行图中代码，应用 for 循环语句编写一个程序实现：从键盘输入一个正整数并计算它的阶乘。

```c
#include <stdio.h>

int main() {
    int i, n;
    long long s = 1;

    printf("请输入一个整数: ");
    scanf("%d", &n);

    if (n < 0) {
        printf("错误: 阶乘只能计算非负整数! \n");
        return 1;
    }

    for (i = 1; i <= n; i++) {
        s = s * i;
    }

    printf("%d的阶乘为: %lld\n", n, s);
    return 0;
}
```

7. 运行图中代码，应用 for 循环语句编写一个程序实现：从键盘输入一个整数作为起始数字，开始合计时，每显示一个数字暂停一秒钟。

```c
#include <stdio.h>
#include <unistd.h>  // 用于sleep函数

int main() {
    int count;

    printf("请输入倒计时起始数字：");
    scanf("%d", &count);

    printf("倒计时开始:\n");

    // for循环实现倒计时，每秒显示一个数字
    for(int i = count; i >= 1; i--) {
        printf("%d\n", i);
        sleep(1);  // 暂停1秒
    }

    printf("时间到！\n");

    return 0;
}
```

循环结构程序设计（二）

教学目标与要求

- ✓ 掌握 for 循环语句的语法结构和执行过程
- ✓ 理解 break 和 continue 语句在循环中的作用
- ✓ 学会使用循环嵌套解决复杂问题
- ✓ 掌握循环结构在实际问题中的应用

教学重点与难点

- ✓ 循环嵌套的执行流程和变量控制
- ✓ break 和 continue 语句的恰当使用
- ✓ 复杂循环问题的分析与算法设计
- ✓ 循环程序的调试与优化

教学课时：3 课时

教学方式：讲授、上机实践、实例讲解

思政目标：通过 C 语言条循环结构程序设计的学习，培养学生坚持不懈、持之以恒的精神;通过程序调试过程，培养学生面对困难不放弃的坚韧品质;通过算法设计，培养学生的逻辑思维和系统思考能力

实例内容：

8. 运行课本实例 5.4，掌握 break 语句的应用，统计捐款情况。

```c
#include <stdio.h>
#define SUM 100000
int main()
{
  float amount,aver,total;
    int i;
// amount: 单笔捐款金额
// aver: 平均每笔捐款
// total: 累计总金额
// i: 捐款次数计数器

for (i = 1, total = 0; i <= 1000; i++)  // 初始化计数器i=1, total=0
{
    printf("please enter amount:");  // 提示输入
    scanf("%f", &amount);            // 读取单笔捐款
    total = total + amount;          // 累加总金额

    if (total >= SUM) break;         // 达到目标金额，立即退出循环
}

 aver = total / i;  // 计算平均每笔捐款
printf("num=%d\naver=%10.2f\n", i, aver);  // 格式化输出
return 0;
}
```

9. 运行课本实例 5.5，掌握 continue 语句的应用，输出 100-00 间不能被 3 整除的数。

```c
#include <stdio.h>
int main()
{
    int n;  // 循环变量n，用于遍历100到200

    for (n = 100; n <= 200; n++)  // 循环从100到200
    {
        if (n % 3 == 0)  // 如果n能被3整除
            continue;     // 跳过本次循环的剩余部分

        printf("%d  ", n);  // 输出不能被3整除的数
    }

    printf("\n");  // 最后输出换行符
    return 0;
}
```

10. 运行下图实例，掌握条件与循环结构的综合运用。判断输入的数是否为素数。

```c
#include <stdio.h>
#include <math.h>
int main() {
int n, i, isPrime = 1;
printf("请输入一个正整数: ");
scanf("%d", &n);

// 处理特殊情况
if (n <= 1) {
    isPrime = 0;  // 1及以下的数不是素数
} else {
    // 只需检查到sqrt(n)
    for (i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) {  // 如果n能被i整除
            isPrime = 0;   // 说明不是素数
            break;         // 立即退出循环
        }
    }
}
if (isPrime) {
printf("%d 是素数\n", n);
} else {
printf("%d 不是素数\n", n);
}
return 0;
}
```

① 变量 i 是一个循环计数器，用于遍历可能的因子,如果 n = 100：sqrt(100) = 10; i 的值依次为：2, 3, 4, 5, 6, 7, 8, 9, 10 当 i = 2 时，100%2=0，发现因子，立即退出循环

② 为什么检查到 sqrt(n)就够了？如果 n 不是素数，那么它至少有一对因子 a 和 b，使得：a × b = n 其中一个因子必然 ≤ $\sqrt{n}$，另一个 ≥$\sqrt{n}$

如：n=36，因子对：(2,18), (3,12), (4,9), (6,6); $\sqrt{36}$ = 6，只需要检查 2-6 即可发现所有可能的因子。

11. 运行下图实例，掌握 break 与 continue 语句的综合运用。遍历数字按要求输出结果。

```c
#include <stdio.h>
int main() {
    int i;
    printf("开始遍历1到20的数字：\n");

    // 遍历1到20的数字
    for (i = 1; i <= 20; i++) {
        // 当遇到大于15的数字时终止循环
        if (i > 15) {
            printf("遇到数字 %d，大于15，终止循环\n", i);
            break;  // 终止整个循环
        }

        // 当遇到5的倍数时跳过
        if (i % 5 == 0) {
            //printf("跳过5的倍数：%d\n", i);
            continue;  // 跳过本次循环的剩余部分
        }

        // 其他数字正常输出
        printf("正常数字：%d\n", i);
    }

    printf("\n循环结束\n");
    return 0;
}
```

① 使用 for 循环从 1 遍历到 20

a. i=1 初始化循环变量

b. i<=20 循环条件

c. i++每次循环后递增

② break 的用法：

a. 当 i>15 时，执行 break

b. break 会立即终止整个循环，不再执行后续的任何迭代

c. 所以 16 之后的数字(17、18、19、20)都不会被处理

③ continue 的用法：

a. 当 i%5==0 时(即 5 的倍数),执行 continue

b. continue 会跳过当前循环的剩余代码，直接进入下一次循环

c. 所以 5、10、15 这些数字只会打印"跳过"信息，不会执行后面的 printf 语句

d. 正常流程：既不是 5 的倍数，也没有大于 15 的数字会正常输出

④ 这个 C 程序清晰地展示了 break 和 continue 在循环控制中的作用：

a. continue:跳过当前迭代，继续下一次循环

b. break:完全终止整个循环

12. 运行以下两图实例，均为输出九九乘法表，掌握 for 循环结构的嵌套应用。

```c
#include <stdio.h>
int main() {
    int i,j;  // 定义两个循环变量
    for (i=1;i<=9;i++)      // 外层循环：控制行数（1-9）
    {
        for (j=1;j<=9;j++) // 内层循环：控制列数（1-9）
            printf("%dx %d=%2d ",i,j,i*j);  // 打印乘法算式
        printf("\n");        // 每行结束后换行
    }
    return 0;
}
```

```c
#include <stdio.h>

int main() {
    int i, j;
    // 外层循环控制行数（1-9）
    for(i = 1; i <= 9; i++) {
        // 内层循环控制每行的列数（1-i）
        for(j = 1; j <= i; j++) {
            // 打印每个乘法式子，使用%-2d保证对齐，内层循环次数随i变化
            printf("%dx %d=%-2d  ", j, i, i * j);
        }
        printf("\n");  // 每行结束后换行
    }

    return 0;
}
```

13. 运行课本实例 5.6，输出如图所示的 4*5 矩阵。进一步掌握循环嵌套处理问题的方法。用外循环输出一行数据，内循环输出一列数据。

```
1    2    3    4    5
2    4    6    8    10
3    6    9    12   15
4    8    12   16   20
```

```c
#include <stdio.h>
int main()
{
    int i, j, n = 0;  // 变量声明
    // i: 外层循环变量（行）
    // j: 内层循环变量（列）
    // n: 计数器，记录当前输出的是第几个数据

    for (i = 1; i <= 4; i++)                    // 外层循环，控制行数（1-4）
        for (j = 1; j <= 5; j++, n++)           // 内层循环，控制列数（1-5）
        {
            if (n % 5 == 0) printf("\n");       // 每输出5个数据后换行
            printf("%2d\t", i * j);             // 输出i*j的结果
        }

    printf("\n");                               // 最后再换一行
    return 0;
}
```

14. 运行下图实例，进一步应用循环嵌套解决问题：有 100 匹马，驮 100 担货，大马驮 3 担，中马驮 2 担，两匹小马驮 1 担，编程计算共有多少种驮法。

```c
#include <stdio.h>
int main() {
    int b, m, s;  // 定义变量：b-大马，m-中马，s-小马
    // 外层循环：大马数量从0到33（因为100/3≈ 33.3）
    for(b=0; b<=33; b++)
        // 内层循环：中马数量从0到50（因为100/2=50）
        for(m=0; m<=50; m++)
        {
            s = 100 - b - m;  // 计算小马数量（总数100）
            // 判断条件：总担数等于100
            // 大马*b*3担 + 中马*m*2担 + 小马*s*0.5担 == 100
            if(b*3 + m*2 + 0.5*s == 100)
                printf("%2d, %2d, %2d\n", b, m, s);  // 输出解
        }

    return 0;
}
```

15. 课后第 8 题：编程输出所有的“水仙花数”，特指每个数位上的数字的 3 次幂之和等于其本身的三位数，如 $153=1^3+5^3+3^3$

```c
#include <stdio.h>
int main()
{ // 每个数位上的数字的3次幂之和等于其本身的三位数，如153=13+53+33
  int i, j, k, n;  // 变量声明
  // i: 百位数字
  // j: 十位数字
  // k: 个位数字
  // n: 当前检查的三位数

  printf("水仙花数是：");

  for (n = 100; n < 1000; n++)  // 遍历所有三位数（100-999）
  {
    i = n / 100;         // 提取百位数字（整除100）
    j = n / 10 - i * 10; // 提取十位数字
    k = n % 10;          // 提取个位数字（取模10）

    // 判断是否为水仙花数
    if (n == i*i*i + j*j*j + k*k*k)
      printf("%d ", n);  // 输出水仙花数
  }

  printf("\n");
  return 0;
}
```

扩展其他位数的水仙花数：

四位：1634, 8208, 9474

五位：54748, 92727, 93084

六位：548834

循环结构程序设计（三）

教学目标与要求

- ✓ 掌握嵌套循环的设计与实现
- ✓ 学会使用循环结构打印各种图形
- ✓ 掌握字符统计等实际问题的循环解决方案

教学重点与难点

- ✓ 嵌套循环的变量控制和执行流程
- ✓ 图形打印中的规律分析和算法设计
- ✓ 复杂循环问题的分析与解决
- ✓ 循环结构的优化与调试

教学课时：3 课时

教学方式：讲授、上机实践、实例讲解

思政目标：通过 C 语言条循环结构程序设计的深入学习，培养学生系统思考和整体规划的能力；通过图形打印练习，培养学生的空间想象力和审美能力；通过问题解决过程及算法优化，培养学生的效率意识和创新精神。

实例内容：

16. 应用*形打印一个 n 行*2n 列的矩形。

```c
#include <stdio.h>
int main() {
    int n, i, j;
    printf("请输入矩形的行数:");
    scanf("%d", &n);
    for(i = 1; i <= n; i++) {
        for(j = 1; j <= 2*n; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

```
请输入矩形的行数:10
********************
********************
********************
********************
********************
********************
********************
********************
********************
********************
```

执行流程分析（假设输入 n=3）：

初始化：声明变量 n, i, j

输入：用户输入 3，n 被赋值为 3

外层循环开始：i=1，条件检查：1≤3 ✓

内层循环：j=1 到 j≤6(2*3=6)，打印 6 个星号：******

打印换行符，i++ → i=2

外层循环第 2 轮：i=2，条件检查：2≤3 ✓

内层循环：j=1 到 j≤6，打印 6 个星号：******
打印换行符，i++ → i=3
外层循环第 3 轮：i=3，条件检查：3≤3 ✓
内层循环：j=1 到 j≤6，打印 6 个星号：******
打印换行符，i++ → i=4
外层循环结束：i=4≤3 ✗

17. 修改上题中的代码打印直角三角形

```c
#include <stdio.h>

int main() {
    int n, i,j;
    printf("请输入图形的行数:");
 scanf("%d", &n);
for(int i=1;i<=n;i++){

    for(j = 1; j <= (2 * i - 1); j++) {
            printf("*");
        }
        printf("\n");
}
}
```

```
请输入图形的行数:10
*
***
*****
*******
*********
***********
*************
***************
*****************
*******************
```

18. 分析上题中的代码，修改代码，打印等腰三角形（思考：如何打印倒三角形）。

```c
#include <stdio.h>
int main() {
    int n, i, j;

    printf("请输入三角形的大小：");
    scanf("%d", &n);

    // 上半部分
    for(i = 1; i <= n; i++) {
        // 打印空格
        for(j = i; j < n; j++) {
            printf(" ");
        }
        // 打印星号
        for(j = 1; j <= (2 * i - 1); j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

```
请输入三角形的大小：10
          *
         ***
        *****
       *******
      *********
     ***********
    *************
   ***************
  *****************
*******************
```

19. 分析上题中的代码，修改代码，打印菱形

```c
#include <stdio.h>
int main() {
    int n = 5; // 菱形的大小

    // 上半部分
    for(int i = 1; i <= n; i++) {
        for(int j = i; j < n; j++) {
            printf(" ");// 打印空格
        }
        for(int j = 1; j <= (2 * i - 1); j++) {
            printf("*");// 打印星号
        }
        printf("\n");
    }

    // 下半部分
    for(int i = n - 1; i >= 1; i--) {
        for(int j = n; j > i; j--) {
            printf(" ");// 打印空格
        }
        for(int j = 1; j <= (2 * i - 1); j++) {
            printf("*");// 打印星号
        }
        printf("\n");
    }

    return 0;
}
```

20. 完成课本实例 5.7，用公式π/4≈1−1/3+1/5−1/7+...求 PI 的近似值 P131

```c
#include <stdio.h>
#include<math.h>
int main()
{
  int sign=1;                          // sign用来表示数值的符号
  double pi=0.0,n=1.0,term=1.0;        // pi代表π ,n代表分母，term代表当前项的值
  while(fabs(term)>=1e-8)              // 检查当前项term的绝对值是否大于或等于10的(-6)次方
  {
   pi=pi+term;                         // 把当前项term累加到pi中
   n=n+2;                              // n+2是下一项的分母
   sign=-sign;                         // sign代表符号，下一项的符号与上一项符号相反
   term=sign/n;                        // 求出下一项的值term
  }
  pi=pi*4;                             // 多项式的和pi乘以4，才是π 的近似值
  printf("pi=%10.8f\n",pi);            // 输出π 的近似值
}
```

21. 习题 5.5：求 Sn=a+aa+aaa+...+aa...a 的值

```
#include <stdio.h>
int main()
 {
  int  a,n,i=1,sn=0,tn=0;
  printf("a,n=:");
  scanf("%d,%d",&a,&n);
  while (i<=n)
  {
  tn=tn+a;  /*赋值后的tn为i个 a组成数的值*/
  sn=sn+tn; /*赋值后的sn为多项式前i项之和*/
  a=a*10;
  ++i;
  }
  printf("a+aa+aaa+...=%d\n",sn);
  return 0;
  }
```

22. 习题 5.7，编辑计算 求 $\sum_{k=1}^{100} k + \sum_{k=1}^{50} k^2 + \sum_{k=1}^{10} \frac{1}{k}$。

```
#include <stdio.h>
int main()
 {
  int n1=100,n2=50,n3=10;
  double k,s1=0,s2=0,s3=0;
  for (k=1;k<=n1;k++)   /*计算1到100的和*/
    {s1=s1+k;}
  for (k=1;k<=n2;k++)   /*计算1到50各数的平方和*/
    {s2=s2+k*k;}
  for (k=1;k<=n3;k++)   /*计算1到10的各倒数和*/
    {s3=s3+1/k;}
  printf("sum=%15.6f\n",s1+s2+s3);
  return 0;
 }
```

23. 习题 5.12：猴子吃桃问题

猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半零一个。到第10天早上想再吃时，见只剩下一个桃子了。求第一天共摘了多少桃子。

```c
#include <stdio.h>
int main()
 {
  int day,x1,x2;
  day=9;
  x2=1;
  while(day>0)
   {x1=(x2+1)*2;     /*第1天的桃子数是第2天桃子数加1后的2倍.*/
    x2=x1;
    day--;
   }
  printf("total=%d\n",x1);
  return 0;
 }
```

一维数组

教学目标与要求

- ✓ 理解数组的概念和作用
- ✓ 掌握一维数组的定义、初始化和引用方法
- ✓ 学会使用数组处理批量数据
- ✓ 掌握数组在排序、统计等实际问题中的应用

教学重点与难点

- ✓ 一维数组的定义和初始化
- ✓ 数组元素的引用方法
- ✓ 数组在循环中的使用
- ✓ 数组在排序算法中的应用
- ✓ 数组下标的理解和使用
- ✓ 数组内存布局的理解

教学课时：3 课时

教学方式：讲授、上机实践、实例讲解

思政目标：通过 C 语言条数组的学习，培养学生系统化、条理化处理问题的能力；通过排序算法的实现，培养学生追求效率和优化的精神；通过批量数据处理，培养学生的数据意识和信息素养。

实例内容：

1、参考课本实例 6.1，对 10 个数组元素依次赋值并逆序输出，输出效果如下右图所示。。

```c
#include <stdio.h>
int main()
 {
  int i,a[10];
  for (i=0; i<=9;i++)

     a[i]=i; //正序初始化
  for(i=9;i>=0; i--)
    printf("%d ",a[i]);
  printf("\n");
  return 0;
 }
```

```
a[0]=9
a[1]=8
a[2]=7
a[3]=6
a[4]=5
a[5]=4
a[6]=3
a[7]=2
a[8]=1
a[9]=0
```

修改后：

```c
#include <stdio.h>

int main()
{
    int i, a[10];  // 定义循环变量i和长度为10的整型数组a

    // 第一个循环：正序初始化数组
    for (i = 0; i <= 9; i++)
        a[i] = i;  // 正序初始化：a[0]=0, a[1]=1, ..., a[9]=9

    // 第二个循环：逆序输出数组元素，但使用转换后的索引显示
    for (i = 9; i >= 0; i--)
        // 输出格式：a[转换后的索引] = 数组元素值
        // 9-i 将逆序索引转换为正序索引显示
        // 例如：当i=9时，显示a[0]=9；当i=8时，显示a[1]=8；...当i=0时，显示a[9]=0
        printf("a[%d]=%d \n", 9 - i, a[i]);

    printf("\n");
    return 0;
}
```

2、完成课本实例 6.2，用数组处理求 Fibonacci 数列问题，输出如图所示的前 20 项。

```
    1        1        2        3        5
    8       13       21       34       55
   89      144      233      377      610
  987     1597     2584     4181     6765
```

分析：

fibonacci数列的各项遵循以下特点：

$$f(n)=\begin{cases}1 & n=1\\ 1 & n=2\\ f(n-1)+f(n-2) & n>=3\end{cases}$$

```c
#include <stdio.h>

int main()
{
    int i;  // 定义循环计数器变量
    int f[20]={1,1};  // 定义长度为20的整型数组，并初始化前两个元素为1（斐波那契数列的前两项）
 // 生成斐波那契数列的循环：从第3项开始计算到第20项
    for(i=2;i<20;i++)
        f[i]=f[i-2]+f[i-1];  // 斐波那契数列规则：当前项 = 前两项之和

    // 输出斐波那契数列的循环：格式化输出所有20个元素
    for(i=0;i<20;i++)
    {
        // 每输出5个元素换一行（当i=0,5,10,15时执行换行）
        if(i%5==0) printf("\n");
        // 输出当前元素，每个数字占12个字符宽度，右对齐
        printf("%12d",f[i]);
    }

    printf("\n");  // 最后输出一个换行，使光标移动到下一行
    return 0;
}
```

3、参考课本实例 6.3，应用“冒泡法”完成编程：输入十个整数，要求把这十个数由小到大的顺序排列输出（输入十个整数后，程序通过双重循环实现冒泡排序：外层控制排序轮数，内层比较相邻元素并交换位置。每轮结束后，最大值逐步“上浮”至数组末尾。经过九轮比较与交换，数组按升序排列。最终输出有序序列，验证算法正确性）。

```c
#include <stdio.h>
int main()
{
    int a[10];      // 定义一个长度为10的整型数组，用于存储用户输入的10个数字
    int i,j,t;      // 定义循环变量i,j和临时变量t（用于交换数组元素）

    printf("输入十个整数:\n");

    // 循环读取用户输入的10个数字并存入数组
    for (i=0;i<10;i++)
      scanf("%d",&a[i]);  // 读取一个整数并存入数组a的第i个位置

    printf("\n");  // 输出一个空行，使界面更清晰

    // 冒泡排序算法开始
    // 外层循环：控制排序的趟数（共进行9趟比较）
    for(j=0;j<9;j++)
      // 内层循环：在每一趟中进行相邻元素的比较
      // 每趟比较的次数递减（9-j次），因为每趟都会将当前最大的元素"冒泡"到末尾
      for(i=0;i<9-j;i++)
        // 比较相邻的两个元素，如果前一个大于后一个，则交换它们的位置
        if (a[i]>a[i+1])
          {
            // 交换a[i]和a[i+1]的值
            t=a[i];       // 将a[i]的值暂存到临时变量t中
            a[i]=a[i+1];  // 将a[i+1]的值赋给a[i]
            a[i+1]=t;     // 将暂存在t中的原a[i]值赋给a[i+1]
          }
    printf("由小到大的排序结果是 :\n");
    for(i=0;i<10;i++)
      printf("%d ",a[i]);  // 输出数组的第i个元素，后面跟一个空格
    printf("\n");
    return 0;
}
```

4、编程输入 10 个整数，求所有元素之和及平均值（可以考虑边输入数字时边求和）。

参考 1：

```c
#include <stdio.h>

int main()
{
    int a[10];
    int i, sum = 0;
    float average;

    // 输入并计算和值
    printf("请输入10个整数: \n");
    for(i = 0; i < 10; i++)
    {
        printf("请输入第%d个整数: ", i + 1);
        scanf("%d", &a[i]);
        sum += a[i];  // 在输入的同时计算和值
    }

    // 计算平均值
    average = (float)sum / 10;

    // 输出结果
    printf("\n输入的10个整数为: ");
    for(i = 0; i < 10; i++)
    {
        printf("%d ", a[i]);
    }

    printf("\n所有元素之和为: %d\n", sum);
    printf("所有元素的平均值为: %.2f\n", average);

    return 0;
}
```

参考二

```c
#include <stdio.h>
int main()
{
    int a[10];  // 定义存储10个整数的数组
    int i, sum = 0;    // 循环变量和求和变量
    float average;     // 平均值变量

    // 输入10个整数
    printf("请输入10个整数: \n");
    for(i = 0; i < 10; i++)
    {
        printf("请输入第%d个整数: ", i + 1);
        scanf("%d", &a[i]);
    }

    // 计算所有元素之和
    for(i = 0; i < 10; i++)
    {
        sum += a[i];  // 累加每个元素
    }

    // 计算平均值
    average = (float)sum / 10;  // 强制类型转换确保浮点数除法

    // 输出结果
    printf("\n输入的10个整数为: ");
    for(i = 0; i < 10; i++)
    {
        printf("%d ", a[i]);
    }

    printf("\n所有元素之和为: %d\n", sum);
    printf("所有元素的平均值为: %.2f\n", average);

    return 0;
}
```

5、编程实现：输入指定一个一维数组的长度并为数组元素赋值，然后求出该一维整型数组中所有元素的最大值并输出。

```c
#include <stdio.h>
int main()
{
    int n;  // 数组大小

    printf("请输入数组的大小: ");
    scanf("%d", &n);

    int arr[n];  // 定义数组

    // 输入数组元素
    printf("请输入%d个整数: \n", n);
    for(int i = 0; i < n; i++)
    {
        printf("元素 %d: ", i + 1);
        scanf("%d", &arr[i]);
    }
    // 寻找最大值
    int max = arr[0];
    for(int i = 1; i < n; i++)
    {
        if(arr[i] > max)
        {
            max = arr[i];
        }
    }

    printf("数组中的最大值为: %d\n", max);
    return 0;
}
```

6、修改上题代码，实现输入指定一个一维数组的长度并为数组元素赋值，然后求出该一维整型数组中所有元素的最小值并输出。

```c
#include <stdio.h>
int main()
{
    int n;  // 数组大小

    printf("请输入数组的大小: ");
    scanf("%d", &n);

    int arr[n];  // 定义数组

    // 输入数组元素
    printf("请输入%d个整数: \n", n);
    for(int i = 0; i < n; i++)
    {
        printf("元素 %d: ", i + 1);
        scanf("%d", &arr[i]);
    }

    // 寻找最小值
    int min = arr[0];
    for(int i = 1; i < n; i++)
    {
        if(arr[i] < min)
        {
            min = arr[i];
        }
    }

    printf("数组中的最小值为: %d\n", min);
    return 0;
}
```

二维数组+字符数组与字符串

教学目标与要求

- ✓ 理解二维数组的概念，掌握二维数组的定义、初始化及元素引用方法，能结合二重循环处理二维数组数据。
- ✓ 掌握字符数组的定义、初始化方式，理解字符串与字符数组的区别，熟悉字符串结束标志'\0'的作用。
- ✓ 熟练使用常用字符串处理函数（puts/gets/strcat/strcpy/strlen/strcmp），能解决字符串输入输出、拼接、拷贝、比较等实际问题。

教学重点与难点

- ✓ 二维数组的定义、初始化（按行分段/连续赋值）及元素引用，二重循环遍历二维数组。
- ✓ 字符数组的初始化（逐个字符/字符串方式），字符串结束标志'\0'的理解与应用。
- ✓ 核心字符串处理函数的使用场景与语法：strcat（拼接）、strcpy（拷贝）、strcmp（比较）、strlen（长度）。
- ✓ 二维数组的内存存储逻辑（按行存储），第一维长度省略的条件理解。
- ✓ 字符串初始化时'\0'的自动补充机制，scanf 输入字符串时空格作为结束符的限制。
- ✓ strcpy 函数使用时字符数组 1 长度不足的问题，strcmp 函数返回值的逻辑判断。

教学课时：**6 课时**

教学方式：**讲授、上机实践、实例讲解**

思政目标

通过二维数组处理批量结构化数据（如成绩表），培养学生系统化、条理化处理问题的思维方式。通过数组排序、字符串比较等算法实现，培养学生追求效率优化、精益求精的工匠精神。在编程解决实际问题（如成绩统计、国家名称排序）的过程中，增强学生的数据意识和工程实践能力。

教学内容

1. 二维数组

① 二维数组的定义

语法格式：类型说明符 数组名[常量表达式 1][常量表达式 2]

常量表达式 1：第一维下标长度（行数）；常量表达式 2：第二维下标长度（列数）。

数组名代表数组首地址，元素按行存储（先存第一行所有元素，再存第二行，以此类推）。

示例：int a[3][3]; 定义 3 行 3 列的整型二维数组，元素依次为 a[0][0]、a[0][1]、a[0][2]、a[1][0]...a[2][2]。

② 二维数组的初始化

③ 二维数组的引用与实例

引用格式：数组名[下标][下标]（下标为整型常量/表达式，需避免越界）。

实例 1：输出二维数组矩阵

实例 2：5 人 3 门课成绩统计（分科平均+总平均）

2. 字符数组与字符串

① 字符数组的定义与初始化

定义：char 数组名[常量表达式]（一维）/ char 数组名[常量 1][常量 2]（二维）。

② 初始化方式：

逐个字符赋值：char c[10]={'c',' ','p','r','o','g','r','a','m'};（未赋值元素为'\0'）。

字符串赋值：char c[]="C program";（自动在末尾添加'\0'，占 10+1=11 字节）。

③ 实例：输出二维字符数组

```
#include int main()
{ int i,j; char a[][5]={{'C','H','I','N','A'},{'B','a','s','i','c'}};
for(i=0;i<=1;i++)
{ for(j=0;j<=4;j++)
printf("%c",a[i][j]); printf("\n"); }
return 0; }
```

3. 字符串处理函数

4. 字符串函数应用实例

实例 1：用户登录验证（strcmp 应用）

实例 2：5 个国家名称按字母排序（二维字符数组+strcmp）

五、总结：

二维数组下标越界：如 int a[3][3]，访问 a[3][0]会导致未定义行为。

strcpy 函数风险：char str1[5]; strcpy(str1, "hello"); 因 str1 长度不足（需 6 字节）导致内存溢出。

scanf 输入字符串：输入"hello world"时，仅"hello"会被存入，空格后内容被丢弃，需用 gets/scanf("%[^\n]")解决。

字符串长度：strlen("abc")返回 3，sizeof("abc")返回 4（含'\0'）。

函数应用

一、教学目标与要求

- ✓ 理解函数的概念，区分库函数与用户自定义函数，掌握函数定义的语法结构（返回值类型、函数名、形参列表、函数体）。
- ✓ 掌握函数调用的形式（无参/有参）、参数传递方式（值传递/地址传递），理解形参与实参的区别与联系。
- ✓ 掌握函数返回值的使用方法，理解 return 语句的作用，能正确接收和处理函数返回值。
- ✓ 理解函数的嵌套调用与递归调用逻辑，能编写简单的递归函数（如求和、年龄计算）。

二、教学重点与难点

- ✓ 函数的定义与调用：无参/有参函数的定义语法，函数调用的三种形式（语句/表达式/参数）。
- ✓ 参数传递：值传递的特点（形参实参独立存储），地址传递（数组参数）的内存逻辑。
- ✓ 函数返回值：return 语句的使用，有返回值/无返回值函数的设计。
- ✓ 变量的作用域：局部变量（函数内/复合语句内）与全局变量的作用范围，同名变量的优先级。
- ✓ 函数声明的必要性：未声明函数导致的编译错误，声明位置（内部/外部）的差异。

教学课时：6 课时

教学方式：讲授、上机实践、实例讲解

思政目标

通过函数的模块化编程思想，培养学生拆分复杂问题、系统化解决问题的思维能力；在函数参数传递、返回值设计中，培养学生严谨的逻辑思维和代码规范意识；通过递归算法的实现，培养学生逆向思维和追求逻辑闭环的工匠精神；通过简易计算器等实际项目开发，增强学生的工程实践能力和解决实际问题的成就感。

教学内容:

1. 函数定义

(1) **函数分类：**

① 库函数：系统提供（printf/scanf/strcat/strlen），直接调用即可。

② 用户自定义函数：按需设计，包含返回值类型、函数名、形参列表、函数体。

(2) **定义语法：**

返回值类型 函数名([形式参数列表]) { // 函数体：实现功能的 C 语句 [return 表达式;] // 有返回值函数需包含 }

(3) **函数类型（按参数/返回值）：** <

分类维度	类型	示例
按参数	有参函数	void add(int x, int y) { ... }
	无参函数	void fun() { ... }
按返回值	有返回值函数	int add(int x, int y) { return x+y; }
	无返回值函数	void add(int x, int y) { printf("%d", x+y); }

(4) **实例：两数求和函数定义**

// 无返回值有参函数 #include void add(int x, int y) { int result = x + y; printf("x+y=%d\n", result); } int main() { add(5, 10); // 调用函数 return 0; }

2. 函数调用

(1) 调用形式

① **无参函数调用**：函数名();

#include void fun() { printf("这是一个无参函数\n"); }

int main() { printf("函数调用前\n"); fun(); // 无参函数调用

printf("函数调用后\n"); return 0; }

② **有参函数调用**：函数名(实参列表);

#include void add(int x, int y)

{ int result = x + y; printf("%d+%d=%d\n", x, y, result); }

int main() { add(5, 10); // 实参 5、10 传递给形参 x、y return 0; }

③ **函数调用的三种方式**：

a. 函数语句：fun();（无返回值函数常用）

b. 函数表达式：sum = add(5, 10);（接收返回值）

c. 函数参数：printf("%d", add(5, 10));（返回值作为其他函数参数）

(2) 参数传递方式

① **值传递：**

a. 特点：形参、实参占用不同内存，形参修改不影响实参。

b. 实例：交换函数（值传递无法修改实参）

printf("函数外交换后：num1=%d,num2=%d\n", num1, num2); // num1=10,num2=20 return 0; }

(3) **地址传递（数组参数）：**

特点：实参传递数组首地址，形参和实参共用同一块内存，形参修改会影响实参。

实例：找数组最小值

(4) 函数返回值

return 语句作用：返回值给主调函数，终止函数执行。

实例：有返回值求和函数

#include int add(int x, int y)

{ int result = x + y; return result; // 返回计算结果 }

int main() { int sum = add(5, 10); // 接收返回值

printf("sum=%d\n", sum); // 输出 15

return 0; }

3. 函数的嵌套与递归调用

(1) 嵌套调用

规则：函数不能嵌套定义，但可嵌套调用（一个函数内调用另一个函数）。

递归调用

规则：函数调用自身，必须有终止条件（if 语句控制），否则无限递归。

实例 1：1~n 求和

实例 2：递归计算年龄

(2) 函数声明

语法：函数定义首行 + 分号（可省略形参名），如：int add(int, int);。

声明位置：

主调函数内部：仅在该函数内有效。

主调函数外部：整个源文件有效。

实例：

#include int add(int, int); // 外部声明

```
int main() { int sum = add(5, 10); printf("%d\n", sum); return 0; }

int add(int x, int y) // 函数定义 { return x + y; }
```

4. 变量的作用域与存储类别

(1) 变量作用域

变量类型 <	定义位置 <	作用范围 <
局部变量	函数内/复合语句内/形参	仅定义区域内有效
全局变量	所有函数外	整个源文件有效

(2) 变量存储类别

存储类别 <	适用变量 <	特点
auto（自动）	局部变量	函数调用时分配，结束后释放（默认）
static（静态）	局部/全局变量	局部：函数调用后不释放；全局：作用域限制在本文件
register（寄存器）	局部变量	存储在寄存器，访问更快
extern（外部）	全局变量	扩展全局变量作用域到其他文件/本文件后续位置

5. 综合实例：简易计算器

6. 总结

(1) 递归调用无终止条件：导致栈溢出，程序崩溃，必须通过 if 设置明确的终止条件。

(2) 值传递修改实参：值传递形参是实参的副本，修改形参无法改变实参，需用地址传递（数组/指针）。

(3) 函数声明与定义不匹配：参数类型/个数不一致，导致编译错误。

(4) 全局变量滥用：增加代码耦合性，尽量使用局部变量，必要时用 extern 扩展作用域。

(5) static 局部变量初始化：仅第一次调用时初始化，后续调用沿用之前的值。

指针

教学目标与要求

- ✓ 理解地址、指针、指针变量的核心概念，掌握取地址运算符“&”和取值运算符“*”的使用方法。
- ✓ 掌握指针变量的定义、赋值及运算规则，能通过指针变量访问普通变量、数组元素。
- ✓ 理解指向一维数组、字符串、二维数组的指针逻辑，掌握数组指针的运算（加减、关系运算）。
- ✓ 掌握指针变量作为函数参数的用法，理解地址传递与值传递的区别，能通过指针实现多值返回。

教学重点与难点

- ✓ 核心概念：地址与指针的关系，“&”和“*”运算符的逆运算特性。
- ✓ 指针变量的定义与使用：指向普通变量、一维数组的指针赋值及元素访问。
- ✓ 指针作为函数参数：通过地址传递修改实参，实现变量交换、数组遍历。
- ✓ 字符串的指针处理：字符串常量赋值给字符指针，指针实现字符串复制。
- ✓ 一维数组指针的运算：p++/p+i 的内存逻辑，*(p+i)与 a[i]的等价性。
- ✓ 二维数组的指针：行指针（a）与列指针（*a）的区别，*(*(a+i)+j)访问元素的逻辑。

教学课时：6 课时

教学方式：讲授、上机实践、实例讲解

思政目标

通过指针对内存地址的精准操作，培养学生严谨的逻辑思维和细节把控能力。在指针数组处理字符串排序等问题中，培养学生模块化、高效率解决问题的工程思维。通过指针与数组的等价性理解，引导学生发现事物的本质联系，培养抽象思维能力。

教学内容

1. 地址和指针基础

核心概念：

地址：变量在内存中的唯一编号（如 0x62fe44），通过“&变量名”获取。

指针：变量的地址即为指向该变量的指针。

指针变量：存放地址（指针）的变量，定义语法：数据类型 *指针变量名;。

运算符：

取地址&：&i 获取变量 i 的地址。

取值*：*p 获取指针 p 指向的变量的值（与&互为逆运算）。

实例 1：指针访问整型变量

```
#include int main() { int a=5, b, *p; p = &a; // 指针 p 指向 a 的地址
 b = *p + 5; // *p 等价于 a
b=5+5=10 printf("a=%d,*p=%d,b=%d\n", a, *p, b); // 输出 a=5
*p=5,b=10
return 0; }
```

实例 2：&与*的逆运算验证

```
#include int main() { int a=5, *p; p = &a; // &a、p、&(*p)均为 a 的地址；
a、*p、*(&a)均为 a 的值
printf("%d,%d,%d\n", &a, p, &(*p));
printf("%d,%d,%d\n", a, *p, *(&a));
 return 0; }
```

2. 指向数组的指针

(1) 指向一维数组的指针

核心规则：

数组名 a 是地址常量，等价于&a[0]，指向数组首元素。

指针变量赋值：int *p = a;或 int *p = &a[0];。

指针运算：p+i 指向 a[i]，*(p+i) 等价于 a[i]；p++ 指向数组下一个元素。

实例 1：数组指针的基本使用

```
#include int main() { int a[5] = {1,2,3,4,5};
int *p = a; // p 指向 a[0]
 printf("%d,%d\n", a, &a[0]); // 地址相同
 printf("%d,%d\n", a[0], *a); // 值相同
 printf("%d,%d\n", *(p+2), a[2]); // 均输出 3
return 0; }
```

实例 2：指针实现两数排序

```
#include int main() { int *p1, *p2, *p, a, b;
```

```
 scanf("%d%d", &a, &b);
 p1 = &a; p2 = &b;
if(a < b) { p = p1; p1 = p2; p2 = p; } // 交换指针指向
printf("max=%d,min=%d\n", *p1, *p2);
 return 0; }
```

(2) 指向字符串的指针

核心规则:

字符串常量赋值给字符指针：char *p = "China";（p 指向字符串首地址）。

字符数组名是常量，不可赋值（如 char ch[10]; ch="hello";错误）。

指针遍历字符串：通过 p++遍历，直到*p == '\0'。

实例: 指针实现字符串复制

```
#include int main()
 { char str1[10], str2[10], *p = str1, *q = str2; gets(str1); // 输入字符串
while(*p) // 直到遇到'\0'
{ *q = *p; // 复制字符 p++; q++; }
*q = '\0'; // 补充字符串结束符 puts(str1);
 puts(str2);
return 0; }
```

(3) 指向二维数组的指针

核心规则：

二维数组名 a 是行指针，a+i 指向第 i 行首地址；*a 是列指针，*a+i 指向第 0 行第 i 列。

元素访问：a[i][j] = *(a[i]+j) = *(*(a+i)+j)。

行指针定义：int (*p)[4];（p 指向包含 4 个 int 的一维数组）。

实例: 行指针输出二维数组

```
#include int main() { int a[3][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}}; int (*p)[4], i; for(p = a; p < a+3; p++) // 遍历每一行 { for(i=0; i<4; i++) printf("%3d", *(*p + i)); // 访问该行第 i 列元素 printf("\n"); } return 0; }
```

(4) 4.3 指针数组

定义:数组元素为指针变量,语法:类型名 *数组名[常量表达式];(如 char *book[5];)。

区别: int *p[5]（指针数组，5 个 int 指针）≠ int (*p)[5]（指向 5 个 int 的行指针）。

实例: 指针数组排序字符串

```
#include #include int main() { char *book[5] = {"Computer network", "Visual C++", "Data structure", "Operating system", "Java"}; char *temp; int i, j; // 冒泡排序 for(i=0; i<4; i++) { for(j=0; j<4-i; j++) { if(strcmp(book[j], book[j+1]) > 0) { temp = book[j]; book[j] = book[j+1]; book[j+1] = temp; } } } // 输出排序结果 for(i=0; i<5; i++) printf("%s\n", book[i]); return 0; }
```

(5) 指针变量作为函数参数

核心规则:

指针参数实现地址传递，形参修改会影响实参（区别于值传递）。

数组名做实参时，传递的是首地址，等价于指针。

实例 1: 指针参数交换两个变量

```
#include void swap(int *p, int *q) { int t = *p; *p = *q; *q = t; } int main() { int a=3, b=5; swap(&a, &b); // 传递地址 printf("a=%d,b=%d\n", a, b); // 输出 a=5,b=3 return 0; }
```

实例 2: 指针参数遍历数组

```
#include void print(int *p, int n) { for(int i=0; i<="" div="" data-doubao-line="359" data-doubao-column="19" data-doubao-key="137">
```

4.5 返回指针值的函数

定义：类型名 *函数名(参数列表)（如 int *fun(int *q);）。

注意：禁止返回局部变量的地址（函数结束后局部变量内存释放）。

实例: 合法返回指针

```
#include int *fun(int *q) { *q = *q + 5; // 修改实参的值 return q; // 返回传入的实参地址（合法） } int main() { int i=5, *p; p = fun(&i); printf("%d", *p); // 输出 10 return 0; }
```

3. 综合实例

实例 1: 统计字符串中子串出现次数

```
#include #include int count_sub(char *str, char *sub)
{ int count = 0; int len_sub = strlen(sub); while(*str)
{ // 匹配子串 if(strncmp(str, sub, len_sub) == 0)
{ count++; str += len_sub; // 跳过已匹配部分
 }else str++; // 指针后移 } return count; }
int main() { char str[] = "sayyousaymesaywetogether";
 char sub[] = "say";
 printf("子串出现次数：%d\n", count_sub(str, sub)); // 输出 3
 return 0; }
```

实例 2：指针实现 3x3 矩阵转置

```
#include void transpose(int (*a)[3], int (*b)[3])
 { int i, j; for(i=0; i<3; i++) for(j=0; j<3; j++)
 b[j][i] = *(*(a+i)+j); // 行列互换 }
int main() { int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
int b[3][3], i, j; transpose(a, b); // 输出转置矩阵
for(i=0; i<3; i++) { for(j=0; j<3; j++)
printf("%3d", b[i][j]); printf("\n"); }
return 0; }
```

4. 总结

指针未初始化：直接使用 int *p; *p = 5; 会导致野指针，程序崩溃。

数组越界访问：int a[5]; int *p=a; p+6; 访问超出数组范围的内存，引发未定义行为。

字符指针修改字符串常量：char *p="hello"; *p='H'; 错误，字符串常量不可修改。

返回局部变量地址：函数内定义 int i=5; return &i;，返回后 i 内存释放，指针指向无效地址。

指针运算混淆：int *p 中 p+1 是地址+4（int 占 4 字节），而非地址+1。