泉州七中科技创新与机器人校本课程模块

C 语言程序设计基础

泉州市第七中学科技创新发展中心

目　录

第一章　　程序设计和 C 语言

1.1 C语言概述

教学目的

■ 了解 C 语言出现的历史背景

■ 掌握 C 语言程序的结构、书写格式和上机步骤

教学重点：C 语言程序的结构

教学难点：上机步骤

一、新课引入

从计算机应用基础中学过的计算机语言及语言处理系统引出 C 语言。

1.C 语言出现的背景

2.C 语言的特点

■ 语言简洁、紧凑，使用方便、灵活；

■ 运算符丰富

■ 据类型多（整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类等）

■ 具有结构化的控制语句

■ 语法不太严格，自由度大

■ 既是高级语言，又具有低级语言的功能

■ 成目标代码质量高，程序执行效率

■ 可移植性好

二、C 语言程序构成（采用程序实例加以说明，并提倡良好的程序设计书写风格）

■ C 语言是由函数构成的，至少有一个 main()函数；

■ 每个函数由函数首部和函数体组成；函数体由说明语句、执行语句组成；

■ 每个 C 程序从 main()函数开始执行，并在 main()中结束；

■ 每个语句和数据定义的最后必须加分号；

■ C 程序无输入、输出语句：输入功能由 scanf()函数完成；输出功能由 printf()函数完成；可加注释/*……*/

三、上机步骤（上机环境：Turbo C 2.0）

■ 进入环境

■ 编辑源程序

■ 保存源程序

■ 编译源程序

■ 执行程序，查看结果

■ 退出 C 环境

四、课堂小结

■ C 语言的构成要素，main 函数在程序中的作用

■ 上机操作的过程

第二章　　算法——程序的灵魂

2.1算法

教学目标

■ 了解算法的概念

■ 掌握结构化程序的三种基本结构，及算法的表示方法

教学重点:算法的表示方法

教学难点:结构化程序的三种基本结构

一、复习引导

从 C 程序的构成到 C 程序的设计过程

二、讲授新课

一个程序包括以下两方面内容：

■ 对数据的描述。在程序中要指定数据的类型和数据的组织形式，即数据结构；

■ 对操作的描述。即算法，为解决一个问题而采取的方法和步骤。著名计算机科学家 Wirth 提出一个公式：数据结构+算法=程序

1.简单的算法举例

■　例 1：设有两个杯子 A 和 B，分别盛放酒和醋，要求将它们互换。

S1：C ← A　　S2：A ← B　　S3：B ← C

■　例 2：求 1~100 的和

S1：sum←0，t←1；　　S2：sum←sum+t

S3：t←t+1　　S4：若 t <= 100，则转到 S2，否则转到 S5；

S5：输出 sum，结束。

2.算法的特征

■ 有穷性

■ 确定性

■ 有零个或多个输入

■ 有一个或多个输出

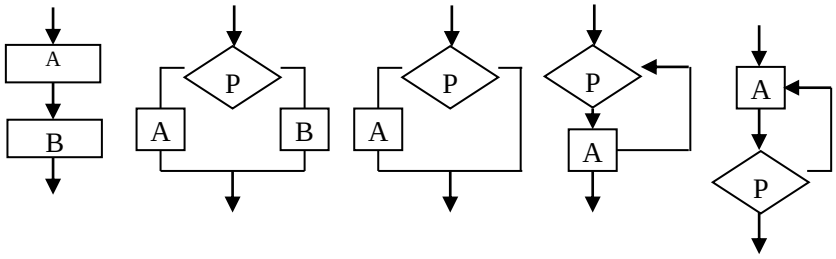
■ 有效性

三、算法的表示

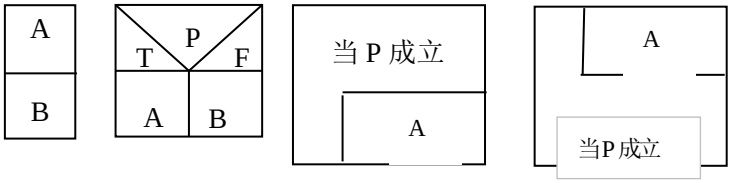
1.用自然语言表示算法–通俗易懂，但有“歧义”。

2.用传统流程图表示算法–直观、易懂。

程序的三种基本结构：顺序结构、选择结构、循环结构



3.N-S 流程图表示算法



■ 用伪代码表示算法

■ 用计算机语言表示算法（即实现算法）

四、设计方法

结构化程序设计方法结构化程序设计方法强调：

■ 自顶向下

■ 逐步细化

程序设计风格和程序结构的规范化，提倡清晰的结构：

■ 模块化设计

■ 结构化编码

五、课堂小结

1.程序的三种基本结构：顺序、选择、循环

2. 5 种描述算法的方法，关键是 N-S 图

3.灵活运用三种基本结构，学会结构化的程序设计方法

六、布置作业

用 N-S 图表示求解以下问题的算法：

1.求 10！

2.将 100~200 之间的素数打印出来

3.求两个数 m ， n 的最大公约数

第三章　顺序程序设计

3.1数据类型

教学目的

1.掌握 C 的数据类型

2.掌握整型、实型、字符型数据的常量及变量

教学重点：各种基本数据类型的常量和变量

教学难点：不同类型的数据在内存中的物理存储形式

一、复习引导

上次课我们已经学习了程序的一个方面算法，现在来学习另一方面数据结构。C 有四种基本数据类型，分别是整型、字符型、实型、枚举型

二、常量与变量

1.常量：在程序运行过程中，其值不能被改变的量。两种形式：一般常量和符号常量

■　直接常量(字面常量)：

整型常量：如 12、0、-3 等

实型常量：如 4.5、-1.234 等

字符常量：如‘a’、‘1’等，用单引号表示；

字符串常量：如“a”、“abc”、“1”，用双引号表示。

■　符号常量：符号常量即是用一个标识符来代替一个常量；符号常借助于预处理命令#define 来实现；

定义形式：#define　标识符　字符串

如：#define　PI　3.1415926535

说明：习惯上，符号常量用大写字母表示；定义符号常量时，不能以“；”结束；一个#define 占一行，且要从第一列开始书写；一个源程序文件中可含有若干个 define 命令，不同的 define 命令中指定的“标识符”不能相同；

2.变量：在程序运行过程中，其值会发生变化。每个变量必须有一个名字，变量名是标识符。标识符是用来标识数据对象，是一个数据对象的名字。命名规则：以字母或下划线开始，后跟字符、数字或下划线。

例：x1，_average，lotus_1_2_3，#abc，1fs，M.D.Jhon

- 变量名不能是关键字(即保留字，是 C 编译程序中保留使用的标识符。 如：auto、break、char、do、else、if、int 等)
- 变量必须先定义再使用

三、整型数据

1.整型常量的表示方法

- 十进制　　如：123，-456，0
- 八进制数　　如：0123，-011　　(以 0 开头的数)
- 十六进制数　如：0x123，-0x12，0xABC　(以 0x 开头的)

2、整型变量

- 整型数据在内存中以二进制形式存放，每一个整型变量在内存中占 2 个字节。

  例：定义整型变量 i=10 和 j= -10 的存放形式。
- 整型变量的分类：基本型 int、短整型 short、长整型 long、无符号型 unsigned
- 整型变量的定义。对变量的定义，一般放在函数体开头部分的声明部分(也可放在函数中某一分程序内)

```
例：#include <stdio.h>
main( )
{   int a, b, c, d;     unsigned u;
    a=12;   b=-24;   u=10;
    c=a+u; d=b+u;
    printf("a+u=%d, b+u=%d\n",c,d); }
```

- 整型数据的溢出。一个 int 型变量的最大允许值为 32767，如果再加 1，其结果不是 32768，而是-32768。即“溢出”。

四、实型数据

1.实型常量的表示方法

- 十进制浮点数

  如：0.123，.456，0.0，123.，123.0

  整数部分和小数部分都可省，但不能同时省
- 指数形式

  如：123e3，123E3 代表 123×10^3

指数部分为整常数；尾数部分可以是整常数，也可以是实常数；尾数部分和指数部分均不可省。E10, 100.e15.2,　.e5 均为不合法的浮点数。

- 实型变量：实型数据在内存中的存放形式。一个实型数据一般在内存中占 4 个字节(32 位)。实型数据是按照指数形式存储的。实型变量的分类：单精度 float、双精度 double、长双精度 long double

五、字符型数据

1.字符常量

- 括在一对单引号中的一个字符(单引号仅作界限符)，如：‘a’ 、‘X’
- 一个字符常量占 1 个字节，存放的是字符的 ASCII 码值。
- 转义字符：以‘ \ ’开头，后跟一个约定的字符或所要表示字符的十六进制（或者八进制）的编码；

2. **字符变量**

字符变量用来存放字符常量，只能放一个字符。例：char c1=‘a’, c2=‘A’; 一个字符变量在内存中占一个字节。

六、字符串常量

- 括在一对双引号中的 0 个或多个字符组成的序列；双引号仅作界限符；如：“C language programming”、“a\\n”、“#123”、“ ”等为字符串常量；
- 字符串常量的实际存储：在存储完字符串中的有效字符后还应存储字符串结束标志‘\0’。

七、变量赋初值

在定义变量时对变量进行赋值称为变量的初始化；

格式：类型说明符　变量 1=值 1，变量 2=值 2，……；

如：　　int　a=3, b=4, c=5;

　　float　x=3.4, y=0.75;

　　char　ch1=‘K’, ch2=‘P’;

八、课堂小结

- C 的基本数据类型 int、float、double、char
- 基本数据类型的常量表示、变量定义，及不同类型的数据在内存中的存储形式

九、布置作业

课后习题

3.2运算符与表达式

教学目的

1.掌握 C 语言中的各种运算符

2.掌握运算符的优先级与结合性

教学重点：C 语言中各种运算符的使用

教学难点：混合表达式中运算符的运算顺序

一、讲授新课

用运算符和括号将运算对象（数据）连接起来的、符合 C 语法规则的句子称为表达式。优先级是指表达式中包含多个运算符时，先进行优先级高的运算符操作，然后在进行优先级低的运算符操作； 当表达式中包含的几个运算符的优先级全相同时，由运算符的结合性来决定他们的运算顺序。

- 从左至右
- 从右至左

二、算术运算符与算术表达式

1.基本的算术运算符: + – * / %

- 优先级： * / % 高于 + –
- 结合性：左结合性

2.算术表达式：用算术运算符和括号将运算对象（操作数）连接起来的、符合 C 语法规则的式子称为算术表达式。

3.强制类型转换运算符：（类型名）（表达式）

4.自增、自减运算符：++ ––。作用是使变量的值增一或减一。

三、赋值运算符与赋值表达式

- 简单的赋值运算符：= 除逗号表达式外，优先级最低
- 复合赋值运算符： += *= %=等
- 赋值表达式： <变量><赋值运算符>< 表达式/值>

- 嵌套的赋值表达式

四、逗号运算符与逗号表达式

- 逗号运算符： ， 所有运算符中优先级最低
- 逗号表达式： 表达式 1，表达式 2，……，表达式 n
- 求解过程：先求表达式 1，再求表达式 2，依次求下去，直到求出表达式 n，整个逗号表达式的值就是表达式 n 的值

五、 课堂小结

1. ++、--运算

2. 各种运算符的优先级 3、表达式值的求解

六、 布置作业

练习：P83 习题

3.3顺序程序设计

教学目的

1.了解 C 语句的概念及种类、掌握 C 语言常用的输入/出方式

2.学会简单的顺序程序设计

教学重点：C 语言常用的输入/出方式

教学难点：格式输入输出

一、复习引导

上一章介绍的常量、变量、运算符、表达式等都是构成程序的基本成分。本章将介绍为编写简单程序所必需的一些内容。

二、C 语句概述

1.C 语句分类：

- 控制语句：二个分支语句（if-else、switch），三个循环语句（for、while、do-while），四个转移语句（continue、break、goto、return）

- 函数调用语句　如：printf(“Hello, world!”);
- 表达式语句 如：　x+y;　　i++; a=2;　a=3*5,　40 ;
- 空语句　　;
- 复合语句　　{　语句序列　}

2.赋值语句：赋值语句是由赋值表达式加上一个分号构成，如：b=3;

三、 数据输入输出的概念及在 C 语言中的实现

所谓输入输出是以计算机主机为主体而言的。C 语言本身不提供输入输出语句，输入输出操作是通过函数调用实现的。要使用 C 语言库函数，应用“#include”将有关头文件包括到用户源程序中。

四、字符数据的输入输出

1.字符输出函数——putchar

- 语法：putchar(c)
- 语义：（向 stdout 终端）输出一个字符；

2.字符输入函数——getchar

- 语法：getchar ( )，是一个无参函数；
- 语义：(从 stdin 终端上) 输入一个字符，函数的值就是从输入设备得到的字符。

五、格式输入输出

1.格式输出函数——printf

- 语法：printf (“格式控制”, 输出表列)；
- 格式控制：是用双引号括起来的字符串，包含两种信息：普通字符和转义字符（这类字符总是原样输出）；格式说明：由%和格式控制符组成。如：%d, %f 等；（P77）如：printf(“a=%d, b=%d”,a,b); 若 a、b 的值分别为 2 和 3，则输出结果为： a=2, b=3

2.格式输入函数　scanf

- 语法：scanf(“格式控制”，地址表列)；
- 格式控制：包含三类符号

空白字符：（空格、Tab 或\ t 、\ n ），输入时不必一一对应；

普通字符：（非格式转换说明符、非空白符、非%），输入时必须一一对应；

格式转换说明符：

注意： scanf 函数规定，组成输入项表的输入对象须是地址量；如：

scanf(“%d,%d,%d”,&a,&b,&c);

scanf(“a=%d,b=%d,c=%d”,&a,&b,&c);

第一个输入语句，正确的输入数据流为：123，456，789<enter>，处理的结果为：

123→a，456→b，789→c

同理对第二个输入语句，正确的输入数据流应是：a=123,b=456,c=789<enter> 该输入数据流中除 123，456，789 被赋给相应变量外，其余都被丢弃

六、顺序结构程序设计举例

■ 例 1：输入三角形的三边长，求三角形面积。

分析：三边为 a,b,c，面积 area2=s(s-a)(s-b)(s-c)，其中 s=(a+b+c)/2

程序：

```
#include <math.h>
main()
{ float  a,b,c,s,area;
        sacnf(%f,%f,%f”,&a,&b,&c);
        s=1.0/2*(a+b+c);
   area=sqrt(s*(s-a)*(s-b)*(s-c));
printf(“a=%7.2f,b=%7.2f,c=%7.2f,s=%7.2f\n”,a,b,c,s);
printf(“area=%7.2f\n”,area);
}
```

第四章　　选择结构程序设计

4.1选择结构程序设计

教学目的

1.掌握实现选择结构的两种语句、两个运算符

2.学会编写选择结构的程序

教学重点：关系运算符、if-else 语句

教学难点：嵌套 if-else 语句中的 if 与 else 匹配问题

一、复习引导

上一次课已经介绍了程序基本结构之一，但在大多数情况下都要用到选择结构。

二、关系运算符与关系表达式

1、关系运算符：<　　<=　　>　　>=　　= =　　! =

优先级：<　　<=　　>　　>= 高于 = =　　！ =

关系运算符低于算术运算符，高于赋值运算符

2、关系表达式：用关系运算符将两个表达式连接起来的式子。

关系表达式求值：关系成立，值为 1；关系不成立，值为 0

三、双分支选择语句

1、if 语句的三种形式

- 语法 1：if (表达式)语句;
- 语法 2：if (表达式) 语句 1；else　　语句 2；
- 语法 3：if (表达式 1） 语句 1；

else　if (表达式 2)　语句 2；

else　　if (表达式 3)　语句 3；

……

else　语句 n+1；

说明：

- if 关键字后均为表达式（逻辑表达式、关系表达式、赋值表达式、变量等）；

如：if (a=5) 语句；　　if (b) 语句；

- 条件表达式必须用括号括起来，在语句后必须加分号；

- 满足条件需执行一组语句时，该组语句必须用{　}括起来；
- if 语句嵌套时，else 总是与它最靠近的未配对的 if 匹配；
- 因为 if 语句执行时总是简单地测试其中作为条件的“表达式”的值是 0 还是非 0，便可利用这种特性来简化程序设计。如对于：

  if (expression!=0)　完全可用　　if (expression)　来代替；

  同理：　if(!exp)语句；等价于：if(exp= =0) 语句；

四、课堂小结

- 关系运算符与关系表达式
- if-else 语句的应用。

4.2选择结构程序设计

教学目的

1、掌握实现选择结构的两种语句、两个运算符

2、学会编写选择结构的程序

教学重点：switch 语句、条件运算符

教学难点：嵌套 if-else 语句中的 if 与 else 匹配问题

一、三目条件运算符及其表达式

1.条件运算符：？：

2.格式：e1？e2：e3

3.语义：判 e1 的值，为 1 时计算 e2，否则计算 e3；

如：max=(a>b)?a:b　等价于 if (a>b) max=a;

else　max=b;

4.说明：

- 条件运算符的结合方向自右至左

  如：a>b?a:c>d?c:d　等价于 a>b?a:(c>d?c:d)

  若 int a=1,b=2,c=3,d=4;则表达式的值为_____

- 条件运算符的优先级仅高于逗号运算符与赋值运算符；
- 只有当 if 语句的真假均只执行一个赋值语句且给同一变量赋值时，才能用条件表达式取代；如：if (a>b) max=a; else max=b;

二、switch 语句

1.语法 1： switch (表达式)

{ case C1：语句序列 1；

case C2：语句序列 2；

……

case Cn：语句序列 n；

default：语句序列 n+1；

}

2.语法 2： switch (表达式）

{ case C1：语句序列 1；break;

case C2：语句序列 2；break;

……

case Cn：语句序列 n；break;

default： 语句序列 n+1；break;

}

3. 说明：

- switch 后面的（）内的表达式，ANSI 标准允许他为任何类型
- case 后的常量表达式一般不可以为实型数据。
- 当表达式的值与某个 case 后面的常量表达式的值相等时，就执行此 case 后面的语句，若所有 case 中的常量表达式的值都与表达式的值不相等，执行 default 后面语句。
- 每个 case 后面的常量表达式的值必须互不相同。
- 各个 case 与 default 出现次序不影响结果。
- break 的使用（P99）
- 多个 case 可以共用一组语句

三、选择结构程序举例

例：求 ax2+bx+c=0 方程的根。

分析：

- a=0，不是二次方程。
- b2-4ac=0，有两个相等的实根。
- b2-4ac>0，有两个不等的实根。
- b2-4ac<0，有两个共轭的复根。

N-S 图：

四、课堂小结

1.switch 语句

2.在编写程序过程中，注意分支的作用范围，及复合语句的运用。

五、布置作业

1、上机作业（P112）：6、8

2、书面作业（P111）3、7

第五章　　循环结构程序设计

5.1循环结构程序设计

教学目的

1.掌握三种循环语句的语法结构

2.灵活运用循环语句

教学重点：三种循环语句 while、do-while、for

教学难点：三种循环语句的区别

一、引入新课

- 问题 1：假如全班 41 人，欲从键盘上输入每人的数学成绩，然后计算出平均成绩；
- 问题 2： 编程计算 n!。

重复执行一组语句是程序设计要求的基本功能之一。在 C 语言中可以用以下语句来实现循环：

- if 和 goto
- while
- do-while
- for

二、goto 语句及用 goto 构成循环

- **语法**：goto label;

其中：label 是语句标号，它命名的规则同变量名；

- **语义**：使程序无条件地转向标号为 label 的语句去执行；

三、while 语句

- **语法**：while (exp)

  循环体语句；
- **语义**：当 exp 为真时，执行循环体；为假时，执行循环语句的后续语句；

如：用 while 语句构成循环，求 sum=1+2+…+100

程序如下：main()

```
{   int   i=1, sum=0;
     while (i<=100)
       { sum+=i;
```

```
        i++; }
    printf("%d",sum);
}
```

说明：

- 循环体可以用复合语句；
- 在 while 语句前应有为测试表达式（exp）中的循环控制变量赋初值的语句，以确保循环的正常开始；
- 循环体内应有改变循环控制变量的语句，以确保循环进行有限次后正常结束； 如：i=1；

```
while (i<=100)
        sum=sum+1;  （死循环）
```

- while 循环的特点是先判断后执行，故循环有可能一次都不被执行；

```
如：    i=3;
        while (i<3)
        printf("i=%d\n",i);
```

四、do-while 语句

- **语法**：do

```
        循环体语句;
        while (exp);
```

- **语义**：当 exp 为真时，执行循环体；为假时，执行循环语句的后续语句；

如：用 do-while 语句构成循环，求 sum=1+2+…+100

程序如下：main()

```
{   int i=1,sum=0;
    do
      { sum+=i;
        i++;   }
     while (i<=100);
     printf("%d",sum);
}
```

说明：

- 循环体可以用复合语句；
- 循环控制变量在执行 do 前必须赋初值；循环体内应有改变循环控制变量的语句；
- do-while 循环的特点是先执行后判断，故循环至少被执行一次；

如：i=3；

```
do
{   sum+= i;
    i++;
 } while (i>10);
```

五、课堂小结

1.while、do-whilr 语句的语法结构

2. whie 与 do-while 区别

3. 注意循环控制的范围

5.2循环结构程序设计

教学目的

1.掌握三种循环语句的语法结构

2.灵活运用循环语句

教学重点：三种循环语句 while、do-while、for

教学难点：三种循环语句的区别

一、复习引导

从 while 和 do-while 语句中引入新的循环语句：for 语句

二、for 语句

1.语法：for(表达式 1；表达式 2；表达式 3)

循环体语句；

2.语义：

- 先求表达式 1；
- 求解表达式 2，若其值为真，则执行第三步；若为假，则结束循环；
- 执行循环体中的语句；
- 求解表达式 3；
- 转回第二步继续执行

如：　　for(i=1; i<=100; i++) sum=sum+i;可看成：

for(循环变量赋初值；循环条件；循环变量增值）语句；

3.说明：

- 显然 for 循环更简洁，更灵活；
- 循环体可以是复合语句；
- for 语句中的三个表达式均可以是逗号表达式，故可同时对多个变量赋初值及修改。如：for(i=0, j=1; j<n && i<n; i++, j++) …
- for 语句中三个表达式可省：

三、几种循环的比较

- 可以相互代替使用
- while, do-while 循环，在 while 后面指定循环条件，在循环体中应包含使循环趋向于结束的语句
- 凡是在 while 中能完成的，在 for 语句中也能完成。

四、break 和 continue 语句

1.break 语句：可以用于 switch 语句中，也可以用于循环语句中，当用于循环语句中时，用于在满足条件情况下，跳出本层循环。

2.continue 语句：用于循环语句中，在满足条件情况下，跳出本次循环。即跳过本次循环体中下面尚未执行的语句，接着进行下一次的循环判断。

五、循环结构程序设计

例 1：用公式求 π 的近似值，直到最后一项的绝对值小于 10-6 为止。

π/4≈　　1-1/3+1/5-1/7+……

```
 # include <stdio.h>
main( )
{    int s=1; float    n=1.0,    t=1,    pi=0;
     while((fabs(t))>1e-6)
     { pi=pi+t;    n+=2;
          s=-s;    t=s/n;
     }
     pi=pi*4;
     printf("pi=%10.6f\n",pi);
}
```

六、课堂小结

1.for 语句的语法结构，特别是 for 语句中三个表达式的作用

2. 注意循环控制的范围

七、布置作业

上机练习：（P141）12、16

书面练习：（P140）3、5

第六章　　利用数组处理批量数据

6.1数组

教学目的

1、掌握一维数组的定义和引用

2、掌握二维数组的定义和引用

教学重点:一维、二维数组的定义、引用、初始化

教学难点:数组的存储形式，数组的首地址

一、 引入新课

数组是有序数据的集合，数组中每一个元素都属于同一个数据类型。

一、一维数组的定义和引用

定义数组，就是要：

- 规定数组的名称，其取名规则与变量名相同；
- 规定数组的类型，包括其数据类型和存储类型；
- 规定数组的大小，即数组的维数及包含的数组元素的个数。数组元素就是包含在数组中的变量。

1.一维数组的定义：

类型说明符　数组名[常量表达式] 例如：int data[10], number[5];

2.一维数组元素的引用

数组名[下标]　例如：a[0]=a[5]+a[7]－a[2*3]

一维数组在内存中占一段连续的存储空间，其首地址：a 或&a[0]

一维数组的初始化

- 在定义数组时对数组元素赋以初值；int a[10]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
- 可以只给一部分元素赋值； int a[10]={0, 1, 2, 3, 4};
- 如果想使一个数组中全部元素值为 0，可简便实现；

  int a[10]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

  其实，对 static 数组不赋初值，系统会对所有数组元素自动赋以 0 值，即上句等价于：static int a[10];
- C 允许通过所赋初值的个数来隐含定义一维数组的大小；int a[]={0,1,2,3,4,5,0};　相当于

int a[7]={0,1,2,3,4,5,0};

二、二维数组的定义和引用

1.二维数组的定义

类型说明符　数组名[常量表达式 1][常量表达式 2];

如：int number[5][4];

数组的存储结构：以行为主序的连续空间

2.二维数组的引用： 二维数组元素的表示形式为：数组名[下标][下标]

3.二维数组的初始化

- 分行给二维数组赋初值：如

  static int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

- 可以将所有数据写在一个花括号内，按数组排列的顺序对元素赋初值；如： static int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};

- 如果花括号内的初值个数少于每行中的数组元素个数，则每行中后面的元素自动赋初值 0；也允许代表给每行元素赋初值的花括号的数目少于数组的行数，这时，后面各行的元素也自动赋 0 值。

- C 语言规定，可以用初始化的方法来隐含定义二维数组第一维的大小，即可以省略数组定义中第一个方括号中的表达式，但不能省略第二个方括号中的表达式。如：static int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12}; 等价于

  int a[ ][4]={1,2,3,4,5,6,7,8,9,10,11,12};

在定义时也可以只对部分元素赋初值而省略第一维长度，但应分行赋初值。如： static int a[ ][4]={{0,0,3},{0},{0,10}};

4.二维数组的输入与输出

用二重循环，以 a[m][n]为例 for(i=0;i<m;i++)

for(j=0;j<n;j++)

{……}

三、课堂小结

1.一维数组、二维数组的定义、引用及初始化

2.一维数组、二维数组的存储形式

四、布置作业

上机练习：（P168）2

书面练习：（P168）5

6.2字符数组

教学目的

1.掌握字符数组的定义、初始化、引用，及输入与输出

2.掌握字符串处理函数

教学重点：字符数组的输入与输出

教学难点：字符串处理函数

一、字符数组的输入

1.用格式符“%s”控制的 scanf（）；

■ 如： static char str1[5]，str2[5]，str3[5];

scanf(“%s%s%s”, str1，str2，str3); /*不能写成&str1*/

若输入数据流为：How are you? str1、str2、str3 分别接收到“How”、“are”、“you？”，且在各个字符串的最后自动加‘\0’。如果利用一个 scanf 函数输入多个字符串时，则以空格分隔；

2.用 gets( )；如：char ch[16]; gets(ch);

注意：

■ gets 一次只能输入一个字符串；

■ 自变量应是数组名，而不能是数组元素名；

■ 要求从键盘上输入一个字符串直到遇到换行符为止，系统会自动把换行符换成“\0”加在字符串末尾。

■ 与 scanf 不同，输入字符串中的空格也会被接收。

二、字符串处理函数

1.puts(字符数组）

■ 功能：将一个以‘\0’结束的字符序列输出到终端；

如： static char str[ ]=“China”； puts(str);

■ 说明：输出的字符串中可含转义字符。

2.gets(字符数组)

■ 功能：从终端输入一个字符串到字符数组中，并得到一个函数值，该函数值是字符数组的起始地址； 如： gets(str);

■ 说明：gets 与 puts 只能输入或输出一个字符串。

3、strcat(字符数组 1，字符数组 2）

■ 功能：将字符串 2 接到字符串 1 的后而且去掉字符串 1 的尾空；

如： static char str1[30]=“YangZhou ”, str2[ ]=“China”；

printf(“%s\n”,strcat(str1,str2));

■ 说明： ①字符数组 1 的长度需足够大； ②去掉字符串 1 的尾空。

4、strcpy(字符数组 1，字符串 2)

■ 功能：将字符串 2 拷贝到字符数组 1 中去；

如： static char str1[10],str2[ ]=“China”;

strcpy (str1,str2);

5、strcmp(字符串 1，字符串 2）

■ 功能：比较字符串 1 和字符串 2，返回：①串 1= 串 2，返回 0；

②串 1＞串 2，返回正整数；③串 1＜串 2，返回负整数。

6、strlen(字符数组）

■ 功能：测试字符串的长度；

如： static char str1[10]＝“China”; printf(“%d\n”,strlen(str));

■ 说明：不包含‘\0’在内。特殊字符%%、\’、\\、\n 代表一个字符。

7、strlwr(字符串）

■ 功能：将字符串中的大写字母转换成小写字母；

8、strupr(字符数组)

■ 功能：将字符串中的小写字母转换成大写字母。

三、课堂小结

1.字符串的结束标记

2.字符串的输入与输出，gets、puts

3.字符串处理函数

四、 布置作业

作业：（P169）13

6.3数组

教学目的

学习用数组解决一些问题

教学重点：数组应用

教学难点：数组应用

一、复习举例

我们已经学习了数组这种数据结构，但还要学会用数组来解决具体问题。

- 例题：利用选择法对 10 个整数进行由小到大排序。
- 分析：选择排序的基本思想如下：第 i 趟排序选出第 i 小的元素，将其与第 i 位上的元素进行交换，n 个元素共需进行 n-1 趟。

```
main()
{ int i, j, min, temp, a[11];
    printf("Enter data:\n");
    for(i=1; i<=10; i++)
     {   printf("a[%d]=", i);
          scanf("%d", &a[i]); }
    printf("\n");
    for(i=1; i<=10; i++)
        printf("%d", a[i]);
    printf("\n");
```

二、课堂小结

1.三种排序方法：冒泡排序、选择排序、插入排序

2.字符数组或字符串的处理

三、布置作业

上机作业：（P168）4、6

书面作业：（P169）8

第七章　　用函数实现模块化程序设计

7.1函数的定义、调用

教学目的

1.掌握函数定义的一般形式

2.掌握函数调用的一般形式

教学重点：函数定义、调用的一般形式

教学难点：形式参数和实际参数

一、复习引导

一个 C 语言源程序可由一个主函数和若干个其他函数组成。由主函数调用其他函数，其他函数也可以互相调用。

一、概述

- 一个源程序文件由一个或多个函数组成。
- 一个 C 程序由一个或多个源程序文件组成。这样可以分别编写、分别编译，提高调度效率。
- 程序的执行从 main 函数开始，在 main 函数中结束整个程序的运行。
- 所有函数都是平行的，即函数不能嵌套定义，函数可以互相调用，但不能调用 main 函数。
- 用户使用的角度看，函数有两种：标准函数(库函数)和用户自己定义的函数
- 从函数的形式看，函数分为两类：无参函数和有参函数

二、函数定义的一般形式：

1.无参函数的定义

[类型说明符]　函数名（）

{　　变量声明部分；

　　执行部分；}

2.有参函数的定义形式

[类型说明符]　函数名（形式参数列表）

{ 变量声明部分；

　执行部分；}

3.空函数

[类型说明符]　函数名（）｛　　｝

三、函数参数和函数的值

1.形式参数和实际参数：

- 形式参数：函数定义时函数名后括号中的变量
- 实际参数：函数调用时函数名后括号内的变量名

关于形式参数和实际参数的说明：

- 形式参数只有在函数被调用时才分配存储单元，调用结束就释放。
- 实际参数可以是变量、常量或表达式，但要求有确定值。
- 在被定义的函数中，必须指明形参类型。
- 实际参数与形式参数的类型应相同或赋值兼容
- 实参变量对形参变量的数据传递是“单向值传递”，即只由实参传递给形参，而不能由形参传回给实参。

2.函数返回值

- 函数的返回值是通过函数中的 return 语句获得的。
- 函数返回值类型：定义函数时应予以指定，若不加指定，则当作 int 处理，并且， 定义函数时，对函数返回值类型的说明一般应和 return 语句中表达式的类型保持一致。
- 若函数值类型与 return 语句中的表达式类型不一致，以函数值类型为准进行类型转换。
- 函数中若没有 return 语句，带回的是一个不确定的、无用的值。
- 可以用“void”定义“无类型”

四、函数的调用

1.函数调用的一般形式：

- 无参函数的调用形式

函数名（）;

- 有参函数的调用：函数名（实际参数列表）;

2.函数调用方式

- 函数语句：不要求函数有返回值
- 函数表达式：函数出现在一个表达式中，函数会带回某一确定值。
- 函数参数：函数调用作为另一个函数的参数。

3.对被调用函数的声明和函数原型

在一个函数中调用另一个函数的条件：

- 被调用函数必须存在且允许调用；
- 必须给出满足函数运行时要求的参数；
- 在调用一个函数之前一般应该对被调用函数进行声明。

函数说明

函数类型　　函数名(参数类型 1,参数类型 2, …, 参数类型 n)；

或：函数类型　　函数名(参数类型 1 参数名 1,参数类型 2 …)；

五、课堂小结

1.函数的定义形式

2.形参和实参的区别

3.函数的声明和调用

六、布置作业

书面作业：（P218）2

7.2函数的嵌套调用和递归调用

教学目的：

1.掌握函数的嵌套调用和递归调用

教学重点：嵌套和递归调用

教学难点：递归调用

一、函数的嵌套调用

1.函数间可以相互调用

主函数可以调用其他函数，其他函数也可以调用除主函数以外的任何函数（含自身）

2.函数的嵌套调用：在调用一个函数的过程中有调用了另一函数

3.函数嵌套调用执行过程

4.举例：

例 1：采用弦截发求方程根

二、函数的递归调用

直接或间接调用自身的函数为递归函数。一个问题采用递归方法来解决时必须符合以下条件：

- 可将一个问题转化为具有同样解法的规模较小的问题；
- 必须有明确的结束条件。

例题：有 5 个人坐在一起，问第 5 个人多少岁，他说比第 4 个人大 2 岁，问第 4 个人的岁数，他说比第 3 个人大 2 岁，问第 3 个人的岁数，他说比第 2 个人大 2 岁，问第 2 个人，他说比第 1 个人大 2 岁，问第一个人，他说是 10 岁。请问第 5 个人的岁数？（P158）

程序：

```
age(int n)
{   int c;
     if(n= =1) c=10;
     else c=age(n-1)+2;
     return (c);
}
main( )
{   printf(“%d”,age(5));}
```

三、课堂小结

1.函数的嵌套调用

2.函数的递归调用

7.3数组作为函数参数

教学目的

1.掌握数组作为函数参数的应用（虚实结合）

教学重点：数组名作为函数参数

教学难点：虚实结合

一、数组元素作为函数实际参数

单向值传递：用赋值的方法，把实在参数的值赋给被调函数对应的形式参数。

不希望破坏调用函数中作为实际参数对象的值时，使用“值传递”方式；

注意：数组元素作为函数实参时如同简单变量。

二、数组名作为函数参数

- 此时，应分别在主调函数和被调用函数中定义数组。
- 实参数组应与形参数组类型保持一致
- 实参数组与形参数组大小可以不一值
- 型参数组可以不指定大小，再定义数组时，在数组名后面跟一个空的方括号，有时为了在被调用函数中处理数组元素的需要，可以另设一个参数，传递需要处理的数组元素的个数。
- 用数组名作为函数实际参数时，不是把数组元素的值传递给形式参数数组，而是把实参数组的起始地址传递给形参数组，这样两个数组就共用同一段存储单元。这种参数传递有时也可以称为“地址传递”

例题：用选择法对数组中 10 个整数按由小到大排序。

```
main()
{ int   a[10],i;
    printf(“enter the array\n”);
    for(i=0; i<10; i++)
            scanf(“%d”,&a[i]);
    sort(a,10);     printf(“the sorted array:\n”);
    for(i=0; i<10; i++)
            printf(“%d”,a[i]);
```

printf(“\n”); }

三、课堂小结

数组作为函数参数有两种情况：传递数组元素的值、传递数组名

四、 布置作业

1.书面作业：4、5

2.上机作业：3、 6

7.4全局与局部变量、动态与静态变量

教学目的

1.掌握局部变量和全局变量的作用范围

2.掌握变量的存储类别，了解内部函数和外部函数

教学重点：局部变量和全局变量、变量的存储类别

教学难点：变量的作用域和生存期

一、复习引导

在函数调用过程中，不仅要注意实参和形参的数据结合，而且要注意各个变量的作用域和生存期。

二、局部变量和全局变量

1.局部变量

在一个函数内部定义的变量是内部变量，它只在本函数范围内有效。

2.全局变量

■ 在函数之外定义的变量称为外部变量，即全局变量（全程变量）。

- 全局变量可以为本文件中其他函数所共用。它的有效范围为从定义变量的位置开始到本源文件结束。
- 使用全局变量可以增加函数间的数据联系；
- 在同一源文件中，如果外部变量与局部变量同名，则在局部变量的作用范围内外部变量不起作用；

[例 8.16]：外部变量与局部变量同名

```
int    a=3,b=5;              /*a,b 外部变量*/
max(int    a, int b);        /*a,b 局部变量*/
{ int c;
    c=a>b?a:b;
    return(c);
}
main()
{ int    a=8;                    /*a 局部变量*/
   printf("%d",max(a,b));}
```

二、变量的存储类别

从变量值存在的(生存期)时间来分，可以分为：动态存储方式与静态存储方式

- 静态存储方式：指在程序运行期间分配固定的存储空间的方式。
- 动态存储方式：在程序运行期间根据需要进行动态的分配存储空间的方式。

1、auto 变量

调用函数时系统自动分配存储空间，在函数调用结束时自动释放这些存储空间，称这类局部变量为自动变量。

自动变量用关键字 auto 作存储类别的声明。它也可省。

2、用 static 声明局部变量

函数中变量的值在函数调用结束后不消失而保留原值，在下次该函数调用时，该变量已有值，即为上一次函数调用结束时的值。该局部变量为静态局部变量。

说明：

- 静态局部变量属于静态存储类别，在静态存储区内分配存储单元。在程序整个运行期都不释放。自动变量属于动态存储类别，占动态存储区空间，函数调用结束后即释放。

- 对静态局部变量在编译时赋初值，程序运行时，它已有初值，以后每次调用函数时不再重新赋初值而只是保留上次函数调用结束时的值。对自动变量赋初值，不是在编译时进行的，而是在函数调用时进行，每调用一次函数重新给一次初值。
- 对静态局部变量来说，如不赋初值，编译时自动赋初值 0 或空字符。
  对自动变量来说，如不赋初值，它的值是一个不确定的值。
- 虽然静态局部变量在函数调用结束后仍然存在，但其他函数不能引用。

3.register 变量

将局部变量的值放在 CPU 中的寄存器中，需要用时直接从寄存器取出参加运算，不必再到内存中去存取。这种变量称为寄存器变量。

4.用 extern 声明外部变量

- 在一个文件内声明外部变量
- 在多文件的程序中声明外部变量

5.用 static 声明外部变量

在定义外部变量时，加一个 static 声明，可以使此变量只能用于本文件中。

注：对外部变量加或不加 static 声明，都是静态存储，只是作用范围不同，都是在编译时分配内存的。

三、内部函数和外部函数

1.内部函数

一个函数只能被本文件中其他函数所调用。

即：static　类型标识符　函数名(形参表)

如：static　　int　　fun(int a,int b)

2.外部函数

一个函数可供其他文件调用，称为外部函数

定义时声明或调用时声明，即：

extern　int　　fun(int　a, int b)　　定义时

或　extern　fun(int　a, int　b)　　调用时

在定义时，也可省写 extern ，即为外部函数

三、课堂小结

1.局部变量和全局变量的作用域

2.变量的存储类别有动态存储方式和静态存储方式

3.关键字有 auto、static、register、extern

四、布置作业

书面作业：（P219）17

第八章　　善于利用指针

8.1变量的指针与指针变量

教学目的

1.了解指针与地址的概念

2.掌握指针变量的定义、引用及指针变量作为参数

教学重点：指针变量的定义、引用及指针变量作为参数

教学难点：指针变量作为参数

一、地址与指针的概念

二、变量的指针与指针变量

变量的指针就是变量的地址。 指针变量是一种特殊类型的变量，它是用于专门存放地址的。

1.指针变量的定义

定义形式：基类型　　*指针变量名；

注意：

- 指针变量前的“*”，表示改变量的类型为指针型变量，“*”后的才是指针变量名。
- 在定义指针变量时必须指定基类型

2.指针变量的引用

指针变量只能存放地址，不要将一个整型量（或其他任何非地址类型的数据）赋值给一个指针变量。

两个相关运算符：

- &：取地址运算符。可以获取某个变量的地址
- *： 指针运算符，获取某个指针变量所值向的变量的值

3.关于&和*运算符的说明：

假设已执行　　pointer_1=&a;

- &*pointer_1 含义是什么？

  &*pointer_1 与&a 相同，即变量 a 的地址。

- *&a 的含义是什么？

  先进行&a 运算，得 a 的地址，再进行*运算。

  *&a、*pointer_1 及变量 a 等价。

■ (*pointer_1) + + 相当于 a + +。

它与*pointer_1 + + 不同。

■ *pointer_1 + + 等价于*(pointer_1 + +)，即先进行*运算，得到 a 的值，然后使 pointer_1 的值改变，这样 pointer_1 不再指向 a 了。

4.指针变量作为函数参数

函数的参数不仅可以是整型、实型、字符型等数据，还可以是指针类型，它的作用是将一个变量的地址传送到另一个函数中。

例 10.3 对输入的两个整数按大小顺序输出。

先考察如下程序，看是否能得到预期的结果

```
swap(int    p1, int    p2)
{ int    temp;
  temp = p1; p1 = p2;    p2 =temp; }
main()
{ int    a, b;
    scanf("%d, %d", &a, &b);
    if(a<b) swap(a, b);
    printf("\n%d,%d\n",a,b); }
```

不能得到预期的结果。

改为：

```
swap(int *p1,int *p2)
{ int    temp;
  temp = *p1; *p1 = *p2; *p2 =temp；}
main()
{ int    a,b;    int    *pointer_1,*pointer_2;
    scanf("%d,%d",&a,&b);    pointer_1=&a;    pointer_2=&b;
    if(a<b)        swap(pointer_1,pointer_2);
    printf("\n%d,%d\n",a,b); }
```

注：如果想通过函数调用得到 n 个改变的值，可以：

■ 在主调函数中设 n 个变量，用 n 个指针变量指向它们；

- 然后将指针变量作实参，将这 n 个变量的地址传给所调用的函数的形参；
- 通过形参指针变量，改变该 n 个变量的值；
- 主调函数中就可以使用这些改变了值的变量。

三、 课堂小结

本课介绍了指针与地址的概念，指针变量的定义、引用及作为参数的使用。

- 指针：就是地址，即内存单元的编号。
- 指针变量：用来存放另一变量的地址(即指针)的变量。

例如：int a=5, *p=&a;

printf (“%d”, *p);

注意：运算符*和&的用法，指针变量的自加自减运算。

8.2数组与指针

教学目的

掌握指针与数组的知识

教学重点：指向数组的指针变量

教学难点：指向二维数组的指针

一、复习引导

上节课介绍了指针变量的定义及其赋值。一个变量有地址，一个数组包含若干元素，每个数组元素都在内存中占用存储单元，它们都有相应的地址。指针变量既然可以指向变量，当然也可以指向数组和数组元素（把数组起始地址或某一元素的地址放到一个指针变量中）。所谓数组的指针是指数组的起始地址，数组元素的指针是数组元素的地址。引用数组元素可以用下标法（如 a[3]），也可以用指针法，即通过指向数组元素的指针找到所需的元素。使用指针法能使目标程序质量高（占内存少，运行速度快）。

二、指向一维数组的指针

定义形式：

int a[10];

int *p;

p=&a[0]; 或 p=a;

含义：把数组的首地址赋给指针变量 p。

也即： int *p=&a[0]; 或 int *p=a;

三、通过指针引用数组元素

按 C 的规定：如果指针变量 p 已指向数组中的一个元素，则 p+1 指向同一个数组中的下一个元素(而不是简单地加 1)。

如果 p 的初值为&a[0]，则：

p+i ↔ a+i ↔ &a[i]，即指向 a 数组的第 i 个元素。

*(p+i) ↔ *(a+i) ↔ a[i]。

指向数组的指针变量也可以带下标，如 p[i]与*(p+i)等价

引用数组元素时，可以用：

1、下标法，如：a[i]

2、指针法，如：*(a+i) 或 *(p+i)

其中，a 是数组名，p 是指向数组的指针

注意指针变量的运算。如果 p 指向数组 a 的首个元素，则：

⑴ p++ (或 p+=1)，使 p 指向下一元素 a[1]。

⑵ *p++ 等价 *(p++)。作用是先得到 p 指向的变量的值(即*p)，然后再使 p+1→p。

⑶ *(p++)与*(++p) 不同。前者为 a[0]，后者为 a[1]

⑷ (*p)++表示 p 指向的元素值加 1，即(a[0])++

⑸ 如果 p 当前指向 a 数组中第 i 个元素，则：

*(p- -)相当于 a[i- -]，先对 p 进行*运算，再使 p 自减；

*(+ + p)相当于 a[+ +i]，先使 p 自加，再作*运算。

*(- - p)相当于 a[- -i]，先使 p 自减，再作*运算。

四、数组名作函数参数

用数组名作实参，在调用函数时是把数组的首地址传送给形参。即实参数组与形参数组共占同一段内存。

如果有一个实参数组，想在函数中改变此数组的元素的值，实参与形参的对应关系有以下4 种情况：

1.形参和实参都用数组名；

```
main()
  { int   a[10];
    f (a,10);……
  }
f ( int x[],int   n ){      ……          }
```

a 和 x 指的是同一个数组

2.实参用数组名，形参用指针变量；

```
main()
  { int   a[10];
    f( a, 10 );
    …… }
f( int   *x, int   n ) {   …… }
```

开始时，x 指向 a[0]

3.实参形参都用指针变量；

```
main()
{ int   a[10], *p;
   p = a;
   f ( p, 10 );
   …… }
 f ( int   *x,   int   n ) { ……   }
```

实参 p 和形参 x 都指向数组 a

4.实参为指针变量，形参为数组名。

```
main()
{ int   a[10], *p;
   p = a;
   f( p, 10 );
```

……}

f (int x[], int n) { …… }

p 指向数组 a，x 和 a 共用同一段内存单元

五、指向二维数组的指针和指针变量

1.二维数组的地址

int a[3][4];

- 一级指针：（二维数组某一个元素的指针）

  a[i], *(a+i) , &a[i][0] 都是元素 a[i][0]的地址。

  a[i]+j , *(a+i)+j , &a[i][j] 都是元素 a[i][j]的地址。

- 二级指针：（二维数组某一行的地址）

  a+i , &a[i] 是二维数组中第 I 行的地址（行号从 0 计算）

- 二维数组元素的引用：

  下标法：a[i][j]

  指针法： *（a[i]+j） ， *（*(a+i)+j）

2.二维数组与指针

- 指向二维数组元素的指针变量（一级指针变量）

  定义： 二维数组元素类型 *指针变量名；

  使用过程与指向一维数组元素的指针变量基本相似，只是要注意数组越界情况。

- 指向二维数组某一行的指针变量（二级指针变量）

  定义形式：数组元素类型 （*指针变量名）[常量表达式]；

  含义：定义了一个指针变量，该指针变量指向一个长度为常量表达式值的一维数组

  例如：int (*q)[4];

- 定义了一个指针变量 q，他指向一个长度为 4 的整型数组，此时，q 的值是该一维数组的起始地址，而不是该一维数组的第一个元素的地址。

  有二维数组：

  int a[3][4];

  若有：int (*q)[4];

- 使 q 指向二维数组第 0 行的赋值语句是：q = a;或 q=&a[0];
- 使 q 指向二维数组第 i 行的赋值语句是：q = a+i;或 q=&a[i];

在此前提下：二维数组元素 a[i][j]可以采用 *(*q+j)引用。

若有：int *q ；

■ 使 q 指向二维数组第一行第一个元素的赋值语句是 q = a[0];或 q = *a;

三、课堂小结

本课介绍了指向数组的指针，主要是指向一维数组的指针。用指针变量 p 指向数组 a，指针变量 p 可以++、--，表示指向数组的上一元素或下一元素。但 C 编译程序不作下标越界检查。使用指针既方便有灵活，但初学者容易搞错。还介绍了指针与二维数组，指向二维数组的指针有指向元素的指针和行指针，使用时应注意它们的区别。

四、布置作业

《C 语言习题集》同步练习

8.3指针与字符串

教学目的

在掌握指针与数组的知识基础上，掌握字符串的指针与指向字符串的指针变量

教学重点：指向字符串的指针变量

教学难点：用指针处理字符串

一、多维数组的指针作函数参数

一维数组的地址可以作为函数参数传递，多维数组的地址也可作函数参数传递。在用指针变量作形参以接受实参数组名传递来的地址时，有两种方法：

1.用指向变量的指针变量；

2.用指向一维数组的指针变量。

二、字符串的指针和指向字符串的指针变量

1.字符串的表示形式

■ 用字符数组存放一个字符串。

如：main()

```
{ char   string[ ]="I love China!";
  printf("%s\n", string);   }
```

■ 用字符指针指向一个字符串。

如：main()

```
{ char   *string="I love China!";
  printf("%s\n", string);   }
```

2.字符串指针作函数参数

3.字符指针变量和字符数组的讨论

■ 字符数组由若干个元素组成，每个元素中放一个字符，而字符指针变量中存放的是地址，决不是将字符串放到字符指针变量中。

■ 赋值方式。对字符数组只能对各个元素赋值，不能用以下办法对字符数组赋值；char str[14]; str="I love China.";

对字符指针变量，可以采用下面方法赋值:

```
char   *a;   a= "I love China.";          /*赋给 a 的是串的首地址*/
```

■ 对字符指针变量赋初值：

```
char   *a="I love China.";   等价于   char   *a;   a="I love   China.";
```

■ 而对数组的初始化：

```
char   str[14]={"I love China."}; 不等价于   char   str[14];
                                              str[]="I love   China.";
```

即数组可以在变量定义时整体赋初值，但不能在赋值语句中整体赋值。

8.4指针与函数、指针数组

教学目的

1.了解指针与函数的概念
2.掌握指针数组，二级指针等知识

教学重点：掌握指针数组，二级指针等知识

教学难点：指针数组，二级指针

一、复习引导

前面介绍了指针与维数组、指针与字符串，我们可以用指针变量指向整型变量、字符串、数组，也可以指向一个函数。

二、函数的指针和指向函数的指针变量

函数的地址：函数存储区域的首地址就是该函数的入口点，其函数名表示了入口地址。

1.函数指针变量的定义：

存储类型　数据类型　(*函数指针名)( );

例：static　int　(*p)( );

说明：

- 函数的调用可以通过函数名调用，也可以通过函数指针调用。
- (*p)() 表示定义一个指向函数的指针变量，它不是固定指向哪一个函数的，而只是表示定义了这样一个类型的变量，它是专门用来存放函数的入口地址的。
- 在给函数指针变量赋值时，只需给出函数名而不必给出参数，如：p=max;　。
- 用函数指针变量调用函数时，只需将(*p)代替函数名即可(p 为指针变量名)，在(*p)之后的括号中根据需要写上实参。如：c=(*p)(a,b);对指向函数的指针变量，像 p+n、p++、p--等运算是无意义的。

2.返回指针值的函数

一个函数可以带回一个整型值、字符值、实型值等，也可以带回指针型的数据，即地址。其概念与以前类似，只是带回的值的类型是指针类型而已。

格式：　类型名　*函数名(参数表);

例：　int　*a(int　x,　int　y);

a 是函数名，调用它以后能得到一个指向整型数据的指针(地址)。

关于函数的返回值是指针的情况，程序设计时应注意：

- 因数组名是地址常量，用于接受这种返值的对象不能是数组名，这与把数组名作为实在参数传递给形式参数的情况不同（作为形式参数的数组名总被视为指针）。
- 应将局部于被调用函数的指针作为返值返回给调用者，理由是局部于被调用函数的数据对象执行返回语句离开被调用函数后，原来分配的被调用函数的所有局部对象的存储空间立即被收回（释放），虽然调用者已经获得了正确的地址值，但此时它指向的存储区域的内容可能已经发生了变化，或许已经分配给其他函数了。如果调用函数中仍然使用这个指针去存取那个区域中的数据，得到的可能并非原先的数据。对于这种情况的正确做法是应该把所处理的对象定义成全局对象或 static 型对象。

二、指针数组和指向指针的指针

1.指针数组的概念

一个数组中的元素均为指针类型，称为指针数组。

形式：　存储类型　类型名　*数组名[数组长度]

例如：　static　int　*p[4]

定义指针数组时也可以进行初始化，如：

static char ch[][20]={“Beijing”,“Nanjing”,“Shanghai”,“Guangzhou”};

char *p[ ]={ch[0],ch[1],ch[2],ch[3]};

该例也可以等价定义为：

char *p[ ]={“Beijing”,“Nanjing”,“Shanghai”,“Guangzhou”};

2.指向指针的指针

在本章开头已提到“间接访问”变量的方式。利用指针变量访问另一个变量就是“间接访问”。　如果在一个指针变量中存放一个目标变量的地址，这就是“单级间址”。指向指针的指针用的是“二级间址”方法。从理论上讲，间址方法可以延伸到更多的级。但实际上在程序中很少有超过二级间址的。级数愈多，愈难理解，容易产生混乱，出错机会也多。

- 二级指针的定义：　char　**p;

含义：表示指针变量 p 是指向一个字符指针变量(即指向字符型数据的指针变量)的。

三、 课堂小结

本课介绍了指针数组、二级指针、指针与函数。要搞清它们的定义及应用；

注意区分：　char　a[5];　　与　char　(*a)[5];

int *p(int x); 与 int (*p)( );

8.5指针数组作main函数的形参、掌握指针的应用

教学目的

1.了解指针数组作 main 函数的形参

2.掌握指针的应用

教学重点：掌握指针的应用

教学难点：指针的应用

一、复习引导

上节课介绍了二级指针、指针数组，而指针数组的一个重要应用是作为 main 函数的形参。main()函数是我们 C 语言程序必不可少的，以往使用时 main()是不带参数的。实际上是可带参数的，如：main(argc, argv) 。

二、指针数组作 main 函数的形参

带参数的 main 原型：

```
main( int  argc, char *argv[ ] )
{  ……      }
```

说明：

- 第 1 个参数是指命令行中参数的个数，含文件名本身。
- 第 2 个参数是一个指向字符串的指针数组。

main 函数是由系统调用的。当处于操作命令状态下，输入 main 所在的文件名（经过编译、连接后得到的可执行文件名），系统就调用 main 函数。参数应和命令一起给出。

- 命令形式： 命令名 参数 1 参数 2 ……参数 n

例如：有一个目标文件名 file1，今想将两个字符串“China”, “Beijing”作为传送给 main 函数的参数。可写成： file1 China Beijing

例题：编写一程序 echo.c，实现将命令行上除程序名之外的所有给出的其他参数都回显到显示器上。

```
main(int argc, int    *argv[ ])
{ while(argc>1)
  { ++argv;
     printf(“%s”, *argv);
     -- argc; }}
```

若将该程序编译、连接、装配成 echo.exe，则在命令行上输入：

echo hello, world!<enter>

则通过虚实结合后得：argc=3，argv[0]指向 echo，argv[1]指向 hello，argv[2]指向 world!

结果为：hello, world!

二、有关指针的数据类型和指针运算的小结

1、有关指针的数据类型的小结

见书中的表

2、指针运算小结

- 指针变量加(减)一个整数 例：p++、p--、p+i、p-=i 等
- 指针变量赋值，将一个变量地址赋给一个指针变量。p=&a; p1=p2;
- 指针变量可以有空值，即该指针变量不指向任何变量。如 ： p=NULL;
- 两个指向同一数组元素的指针变量可以相减
- 两个指向同一数组的指针变量可作关系运算

三、习题举例：

例题：有 n 个人围成一圈，顺序排号。从第一个人开始报数(从 1 到 3 报数)，凡报到 3 的人退出圈子，问最后留下的是原来第几号的那位。

程序：

```
main()
  { int   i, k, m, n, num[50], *p;
      printf(“Input number of person:n=”);
      scanf(“%d”,&n);       p=num;
```

```
for( i=0; i<n; i++)      *(p+i) = i+1;
  i=0;    k=0;    m=0;
  while ( m<n-1)
        { if( *(p+i)!=0) k++;
            if(k==3)
                { *(p+i)=0; k=0;    m++; }
            i++;
            if( i= =n) i=0; }
  while(*p==0) p++;
  printf("The last one is NO.%d\n",*p); }
```

三、布置作业

《C 语言习题集》同步练习

第九章　用户自己建立数据类型

9.1结构体类型与变量的定义及使用

教学目的

1.结构体类型的定义

2.结构体变量的定义、初始化及引用

教学重点:结构体变量的定义、初始化及引用

教学难点:结构体变量的使用

一、引入新课

到目前为止，已介绍了基本类型的变量，也介绍了一种构造类型的数据---数组。但是只有这些数据类型是不够的，有时需要将不同类型的数据组合成一个有机的整体，以便与应用。这些组合在一个整体里的数据要求相互关联，这就是我们所要介绍的———结构体。

一、结构体概述

■　C 语言没有提供现成的结构体数据类型，需要用户在程序中根据需要定义。

■　结构体类型定义的一般形式：

struct 结构体名

{　成员列表 }；

其中，结构体名用作结构体类型的标志，成员表列的定义形式如下：

类型名　成员名；

二、定义结构体类型变量的方法

1.结构体类型定义只是指定了一个结构体数据的模型，其中并无具体数据，系统也不为它分配内存单元。为了能在程序中使用结构体类型的数据，应当定义结构体类型的变量，并在其中存放具体数据。

2.结构体变量定义方法：

■　先声明结构体类型再定义变量名

形式：struct 结构体名

{　成员列表 }；

struct 结构体名　　变量名 1，变量名 2；

■　声明结构体类型的同时定义变量名

形式：struct 结构体名

{ 成员列表 }变量名 1，变量名 2；

■ 直接定义结构体类型变量

形式：struct

{ 成员列表 }变量名 1，变量名 2；

其中以第一种方式最常用。

说明：

■ 类型与变量是不同的含义，不要混淆。

■ 对结构体变量中的成员，可以单独使用。

■ 成员也可以是一个类型已定义的结构体变量。

3.结构体变量的引用

结构体变量的引用应遵守以下规则：

■ 不能将一个结构体变量作为一个整体进行输入输出。只能对该结构体变量的各个成员分别进行输入输出。引用结构体变量的成员的方法是：结构体变量名．成员名，其中的“．”是成员运算符

■ 如果成员本身又属于一个结构体类型，则要用若干个成员运算符，一级一级的找到最低的一级的成员，只能对最低级的成员进行赋值或存取以及运算。

■ 对结构体变量的成员可以像普通变量一样进行各种运算。

■ 可以引用结构体变量成员的地址，也可以引用结构体变量的地址。

结构体变量的初始化

例如：

```
main()
{
 struct student
   {
long num;
char name[20];
char sex;
int   age;
}a={89032,”liling”,’M’,21};
……
```

}

三、课堂小结

- 结构体类型定义
- 结构体变量定义
- 结构体变量引用
- 结构体变量初始化

注意区别结构体类型与结构体变量，并且要知道结构体类型不是惟一的，是可以根据不同的需求来建立不同的结构体类型的

四、布置作业

书面作业：（P330）1

9.2结构体数组、结构体类型数据与指针

教学目的

1.结构体数组的应用

2.指向结构体类型数据的指针的应用

教学重点:指向结构体类型数据的指针

教学难点:指向结构体类型数据的指针

一、复习引导

一个结构体变量的指针就是该结构体变量所占据的内存段的起始地址。可以设一个指针变量，指向一个结构体变量，此时指针变量的值就是结构体变量的起始地址。指针变量也可以用来指向结构体数组中的元素。

二、 指向结构体变量的指针定义及赋初始值

- 定义结构体类型：struct 结构体名{ 成员列表 }；
- 定义结构体变量：struct 结构体名 结构体变量名；
- 定义指向结构体变量的指针变量: struct 结构体名 *指针变量名；

■ 指针变量名= & 结构体变量名；

三、引用结构体变量中的成员

1.结构体变量名．成员名

（*指针变量名）．成员名　　注意，这里的（）不能省略，因为“．”运算符的优先级高于“*”2.运算符

3.指针变量名->成员名，其中“->”称为指向运算符。

注意分析以下几种运算：（p 是指向结构体变量的指针变量，n 式结构体变量的成员名）

■ p->n　　得到 p 指向的结构体变量的成员 n 的值

■ p->n++　　得到 p 指向的结构体变量的成员 n 的值，用完该值后再将它加 1

■ ++ p->n　得到 p 指向的结构体变量的成员 n 的值加 1，然后再使用它。

四、指向结构体数组的指针

例如：指向结构体数组的指针的应用。

```
struct student{    int num;    char name[20];    char sex;
                   int age;};
struct student stu[3]={{10101,”Li Lin”,’M’,18},\
                       {10102,”Zhang Fun”,’M’,19},\
                      {10104,”Wang Min”,’F’,20}};
main( )
  { struct student *p;
    for(p=stu; p<stu+3; p++)
     printf(“%5d%-20s%2c%4d\n”,p->num,p->name, p->sex,p->age);
  }
```

对“－＞”、“．”、“[ ]”、“（）”的进一步说明

上述运算符具有相同的运算优先级和结合性；

■ ＋＋p－＞num 的执行效果等价于：　　＋＋（p－＞num）；

■ (+ +p)－＞num 的执行效果是：先执行 p=p+1， 然后再执行 p－＞num(注意 p 的值已经改变)；

■ （p+ +）－＞num 的执行效果是：先存取 p－＞num，然后再执行 p=p+1；

■ p+ +－＞num 的执行效果同（ p + +）－＞num;

与定义指向普通数组的指针一样，C 语言允许定义指向结构数组的指针。如果 p 是指向结构数组的指针，那么 p 将指向该结构数组的起始地址（第 0 个元素的地址）、p+1 将指向这个结构数组的第一个元素的地址、p+2 将指向该结构数组的第三个元素的地址、…，依此类推。下例展示了本节所讨论的两种指针在程序中的实际应用。

例如：

```
int x[3]={4, 5, 6};
    struct ss{int *y; char c;
                }stu[3]={{&x[0],'1'},{&x[1],'2'},{&x[2],'3'}};
    sturct ss *p=stu;
```

■ *p－＞y 即*(p－＞y)，意味着 p－＞y 获得的一个地址量，故＊（p－＞y）实际上是获取 p－＞y 内容的内容；

■ *p－＞y+ +应理解为：先存取 p－＞y 的内容，再使 p－＞y 的内容增 1；

■ (*p－＞y)+ +是先取出 p－＞y，再将取出的内容加 1；

■ *p+ +－＞y 是取出 p－＞y 内容的内容，再使 p 加 1。

四、用结构体变量和指向结构指针作函数参数

1.用结构体变量的成员作参数

2.用结构体变量作实参

说明：用结构体变量作实参时，采取的是“值传递”的方式，将结构体变量所占的内存单元的内容全部顺序传递给形参。形参也必须是同类型的结构体变量。在函数调用期间形参也要占用内存单元。这种传递方式在空间和时间上开销较大，如果结构体的规模很大时，开销也是可观的。此外，由于采用值传递方式，如果在执行被调用函数期间改变了形参（也是结构体变量）的值，该值不能返回主调函数，这往往造成使用上的不便，因此一般较少使用这种方法。

3.用指向结构体变量（或数组）的指针作实参

五、课堂小结

本课主要学习了结构变量与指针的应用，利用指针引用结构成员。

六、布置作业

（P330）3、4

9.3用指针处理链表

教学目的

领会存储动态分配和释放，领会链表的基本概念

教学重点：存储动态分配和释放，链表的概念

教学难点：存储动态分配和释放，链表的概念

一、链表概述

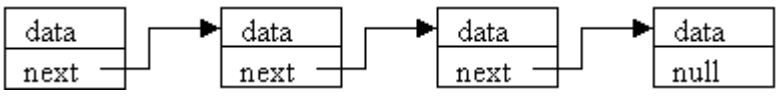
在讨论结构变量定义时曾指出，组成结构的成员项可以是任何数据类型。一个结构中的成员项可以是另一个结构类型的变量，或指向另一个结构类型的结构指针，甚至还可以是指向本结构类型的一个结构指针。如果一个结构中的一个成员是另一个结构变量，这样的结构称之为嵌套结构；如果一个结构的成员项是指向本结构类型的结构指针，这样的结构称之为“自引用结构”。

1.若一结构的成员项是指向本结构类型的结构指针，这样的结构称之为“自引用结构”。例如：

```
struct node {
        int data;
        struct node *next;
        };
```

便是一个典型的自引用结构。该结构的结构名为 node，它由两个成员项组成：一个是 int 变量 data，另一个是结构指针且是一个指向 node 类型对象的指针。

2.下图是 node 的直观表示:



这是一种单向链表数据结构，链中的元素（也称“结点”）个数可以有任意多个（动态地进行存储分配的一种结构）。

二、链表结构可用于实现动态存储分配

1．动态数据结构与结构数组的区别

动态数据结构相当于结构数组，但比结构数组优越

■ 结构数组中的元素必须连续存放，而链表不必；

■ 数组中的元素个数确定，而链表中的元素个数没有限制，在实际使用中如果元素个数不确定，特别是需要动态增加元素的情况，使用链表更合适；

■ C 编译程序必须给数组分配存放其全部元素的存储空间，而对链表不必也不可能预先分配全部存储空间，因为 C 编译程序元法确定链表中的元素的个数。

2．用自引用结构实现链表结构需要解决三个问题

■ 必须指出链表第一个结点的位置，否则无法存取该链表中的结点

实现方法：定义一个指向该结构对象的指针，或定义一个该结构类型的变量，使其指向链表的第一个结点即可。如：

```
struct node *head;
```

■ 在建立一链表时，如何获得下一个新的结点的存放空间。

实现方法：用 C 编译系统提供的库函数 malloc(size)动态分配存储空间得到。除此之外，还可用 calloc 函数。如：

```
struct node *p1, *p2;
    …
p2=(struct node *) malloc(size(struct node));
```

或： p2=(struct node *) calloc(1, size(struct node));

```
if (p2= =NULL)    exit(0);
p1－＞next=p2;   /*此处设 p1 指向新结点的上一结点*/
```

另外：free 函数可用于释放内存区。

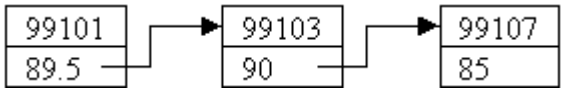
如：free(p); 表示释放由 p 指向的内存区，使这部分内存区能被其他变量使用。

■ 要明确指出链表的链尾。

实现方法：通常把最后结点中的成员项 next 置为空指针 NULL 即可。

三、简单链表

例：建立一个如下图所示的简单链表，并输出各结点中的数据。




```
#define NULL 0
    struct strudent
        { long num;
          float score;
          struct student *next;
        };
main( )
{ struct student a,b,c, *head, *p;
    head=&a;
    a.num=99101; a.score=89.5; a.next=&b;
    b.num=99103; b.score=90; b.next=&c;
    c.num=99107; c.score=85; c.next=NULL;
    p=head;
    do{ printf("%ld%5.1f\n", p->num, p->score);
            p=p->next;
        } while(p!=NULL);   }
```

四、链表的生成

五、链表的输出

例题 编写一个输出链表的函数 print。

```
void    print( struct   student   *head )
{ struct student *p;
    printf("\nNow, These %d   records are :\n",n);
    p = head;
    if ( head!=NULL)
```

```
        do
        { printf("%ld %5.1f\n",p->num,p->score);
          p=p->next;
        } while (p!=NULL);
```

六、链表的删除操作

七、链表的插入操作

例题：写一个函数 insert 插入一结点。

分析：插入点可能有以下三种情况：在链表中间、表头、表尾。

```
struct student *insert(struct student *head, struct student *stud )
{ struct student *p0, *p1, *p2;
   p1=head;   p0=stud;
   if(head= =NULL) {head=p0; p0->next=NULL;}
   else { while((p0->num>p1->num)&&(p1->next !=NULL))
                  { p2=p1; p1=p1->next;}
          if(p0->num<=p1->num)
                  if(head= =p1)   {head=p0;   p0->next=p1; }
                  else { p2->next=p0; p0->next=p1; }
          else {p1->next=p0; p0->next=NULL;}
        }
  return(head);}
```

八、课堂小结

本课主要学习了结构变量与指针的应用，利用指针引用结构成员；并介绍了链表的概念，作为链表结点的结构类型，及存储动态分配与释放的使用，静态链表的建立及其他操作，同学们在对链表操作时，应分析可能的情况，并画出链表的示意图。本章我们只学习了单向链表，在后续课程中还将学习其他类型的链表。。

四 布置作业

（P330） 9

9.4共用体、枚举类型和typedef

教学目的

1.掌握共用体类型的说明、共用体变量的定义、成员的引用
2.领会枚举类型变量的定义，了解 typedef 的作用

教学重点：掌握共用体类型的说明、共用体变量的定义、成员的引用

教学难点：共用体变量的赋值及所占存储空间、成员的引用

一、复习引导

结构体变量所获得的存储空间是各成员项所占空间之和。

二、共用体(联合)

联合（共同体）也是一种构造类型的数据结构。在一个“联合”内可以定义多种不同的数据类型，因为有时我们需要使几种不同类型的变量放到同一内存单元中。允许利用同一存储区域来存储、处理不同类型的数据。使几个不同的变量共占同一段内存的结构，称为“联合”（共同体）类型的结构。

1.共用体的概念

使几个不同类型的变量共占同一段内存的结构。

共用体类型的定义形式:

```
union   共用体名
            {    数据类型 1        变量名 1;
                 数据类型 2        变量名 2;
                  …                 …
                 数据类型 n        变量名 n;
             };
```

例如：

```
union data
  {  int     i;
     char    ch;
     float   f;
    }a,b;
```

其中，变量 a,b 的成员 i,ch,f 共用一段空间。变量 a,b 的空间分别为 4 字节。

2.共用体变量的引用

共用体变量名．成员名

如： a．i 或 a．f

3.共用体变量的特点

- 同一内存段瞬时只能存放成员表中的一种，此时其他成员不起作用；
- 共用体变量的地址及各成员的地址相同；

  即 &a、&a.i、&a.ch、&a.f 均是同一地址。

- 不能用共用体变量名进行赋值、初始化等操作；如：

  union { int i; char ch; float f; }a={1,’a’,1.5}； ×
- 共用体与结构体可以嵌套使用。
- 不能用共用体变量作为函数参数，也不能使函数带回共用体变量，但可用指向共用体变量的指针作函数的参数。

三、枚举类型

1.枚举类型的概念

列举出所有可能的取值的一种数据结构。

2.枚举类型的定义

enum 枚举名

{ 枚举值表 };

例：enum weekday{sun, mon, tue, wed, thu, fri, sat};

enum weekday week_end, workday;

或 enum weekday{sun,mon,tue,wed,thu,fri,sat}workday;

或 enum {sun,mon,tue,wed,thu,fri,sat}week_end;

3.枚举类型变量的赋值和使用

- 枚举值是常量，不是变量；sun=5; mon=2; sun=mon; ×
- 枚举类型是有序数据类型，枚举元素是有值的；

例：main( )

{ enum weekday {sun, mon, tue, wed, thu, fri, sat }a,b,c;

a=sun; b=mon; c=tue;

```
    printf("%d,%d,%d",a,b,c);
}
```

运行结果为：0，1，2

可改变枚举元素的值。

```
enum weekday{sun=7, mon=1,tue,wed,thu,fri,sat} day;
```

- 枚举元素可比较；例：默认时，mon>sun
- 只能将枚举值赋予枚举变量；

例：a=sun; b=mon;　是正确的　而 a=0;　b=1;　是错误的

但　b=(enum weekday)1; 是正确的

- 枚举元素不是字符串常量，使用时不要加引号。
- 由于 C 编译程序将枚举量作为整型数来处理，所以可使用常数的地方，都可以使用枚举常量。

四、 用 typedef 定义类型

C 语言不仅提供了丰富的数据类型，而且还允许用户自己定义类型说明符，即允许用户为数据类型取别名。类型定义符 typedef 可用来完成此项功能。

typedef　可完成为类型取别名.

typedef　的一般形式：

typedef　旧类型名　新类型名；

如：typedef　float　real;

real　a, f ;

再如：typedef　char　*pointer ;

pointer　p, string ="example";

但我们必须明白：首先，定义的新名只是原名的一个别名，并不是建立一个新的数据类型；其次，新名和原名同时存在并有效，即原名并不失去效用，在程序中仍可使用；最后，用新名和原名定义的对象具有相同的性质和效果。

五、课堂小结

本课主要学习了共用体、枚举类型及用 typedef 定义类型。在使用共用体变量时，要和结构体变量做好区分。

六、布置作业

做《C 语言习题集》同步练习第 9 章习题

第十章　对文件的输入输出

10.1文件

教学目的

1.掌握标准设备输入/输出函数的使用
2.掌握缓冲文件系统的使用

教学重点:标准设备输入/输出函数(部分)的使用，文件的使用

教学难点:文件的使用

一、新课导入

C 语言把文件看成是一个字符（字节）的序列。按数据的组织形式，分为 ASCII 文件和二进制文件。前者每一个字节存放一个 ASCII 字符，后者把内存中的数据按其在内存中的存储形式输出到磁盘上存放。前者占空间多，需要转换，后者节省空间和转换时间，但一个字节不对应一个字符，不能直接输出字符形式。

C 语言中可利用高级 I / O 库函数来存取文件，存取文件的过程与其他语言中的处理过程类似。通常按如下顺序进行：

…
打开文件
…
读写文件（若干次）
…
关闭文件

这个处理顺序表明：一个文件被存取之前首先要打开它，只有文件被打开后才能进行读 / 写操作，文件读 / 写完毕后必须关闭。

一、文件的打开

在操作系统中，每一个文件都有一个名字以供识别，如存储在磁盘上的 C 源程序文件 file1．c，file2．c 等。文件名是文件的外部名，通过它可以找到文件的实际存储设备、位置、大小、特性等诸如此类的相关信息。这些信息只能由操作系统的文件管理系统掌握与管理，因此要存取文件必须通过操作系统的文件系统。这意味着一个 C 语言程序没有直接通过文件的外部名存取一个外部文件的能力，程序中要存取文件必须与文件系统取得联系，把要存取

文件的有关信息和要求，诸如文件的名字、读文件还是写文件、以何种方式读 / 写等告诉文件系统，由文件系统在设备中建立、寻找、定位文件，分配存取文件的缓冲区，做好存取文件要求的一切准备工作。

上述存取文件的有关信息和要求都由程序通过 I/O 库函数 fopen 告诉操作系统。

fopen 函数的一般调用形式是：

FILE *fp；

fp=fopen（文件名，存取方式）；

二、文件的关闭

在使用完一个文件后应该关闭它，以防止它再被误用。“关闭”就是使文件指针变量不指向该文件，也就是文件指针变量与文件“脱钩”，此后不能再通过该指针对其相连的文件进行读写操作，除非再次打开，使该指针变量重新指向该文件。用 fclose 函数关闭文件。

fclose 函数调用的一般形式为：

fclose (文件指针)；

例如： fclose(fp)；

三、文件的读写

读/写字符函数（fputc 函数和 fgetc 函数）或（putc 函数和 getc 函数）

1.fputc 函数——把一个字符写到磁盘文件上去。

一般形式为： fputc（ch，fp）；

2.fgetc 函数——从指定文件读入一个字符。该文件必须是以读或读写方式打开的。调用形式为：ch=fgetc（fp）；

四、课堂小结

本课讲解文件指针的说明，并用指针指向文件。利用 fopen()和 fclose()函数打开、关闭文件，注意：需要用文件时，要打开；不用后，立刻关闭。另外，还学习了两个读写文件的函数 fgetc()、fputc()，用于读写字符。

五、布置作业

（P354）第 3 题