第一章C语言及程序设计概述

本章要求:

了解 C 语言的特点、C 语言与其它高级语言相比有什么异同;

了解 C 程序在 PC 机上的建立、编译和运行过程;

了解 C 语言程序设计思想的基本篇;

重点:①C 语言的主要特点;

②C 语言在 PC 机上的运行过程及上机操作过程;

③常用算法的应用

难点: 无

一、C 语言概述

C 语言是目前国际上在各种计算机运行较为广泛流行的一种高级语言.

主要表现为:

C 语言: 适合于作为系统描述语言——可用来写系统软件。

具有高级语言的特点, 又有低级语言(汇编语言) 的特点。

C 语言: 是 AT&T 贝尔实验室的 Dennis Ritchie 在 1972 年发明的。

C 语言: 是在 B 语言的基础上发展起来的.(ALGOL 60)

C 语言:最早开始是用于 UNIX 操作系统。(C 语言和 UNIX 是相辅相成的)

DBASE 、Microsoft Exel 、 Microsoft Word、PC—DOS 等,则是用 C 语言加上若干汇编子程序编写的。

1983 年:——制定的新标准,称为(美国国家标准化协会)ANSI C

1987 年:——Turbo C1.0 (Borland) 1988 年: TC 1.5(图形和文本窗口)

1989 年:——Turbo C2。0 (DOS 操作系统、查错、Tiny 模式生成 com 文件)

1991 年: ——Turbo C++ (3.0) 支持 windows 3.X

说明:Turbo C 语言可以在程序的任何地方嵌入汇编代码,提高速度, 可以直接使用存储器和寄存器。

二. C 语言的特点

1. 语言简洁、紧凑、使用方便、灵活

C 语言: ——共有 32 个关键字, 9 种控制语句;

程序书写自由, 主要用小写字母表示;

2. 运算符丰富

C 语言的运算符包含的范围很广泛, 共有 34 种运算符;

即: 把括号、赋值、强制类型转换都作为运算符处理

3. 有丰富的数据类型

整型、实型、字符型、数据类型、指针类型、结构体类型、共用体(联合) 类型等。实现复杂的数据结构(链表、树、栈、图) 的运算。

4. 具有结构化的功能, 用函数作为程序模块, 实现程序的模块化

5. 语法限制不太严格, 程序设计自由度大. (放宽了语法检查)

例: 1) 对数组下标越界不作检查, 由程序编写者自己保证程序的正确;

2) 整型数据、字符型数据、逻辑型数据可以通用。

6. 能直接访问物理地址, 能进行位(bit) 操作,能实现汇编语言的大部分功能,可以直接对硬

件进行操作。

7．生成目标代码质量高，程序执行效率高。

8．用 C 语言编写的程序，移植性较好。

说明:

C 语言比其它高级语言难掌握，对编程人员要求较高

程序员使用 C 语言编写程序会感到限制少、灵活性大,功能强，可以编写出任何类型的程序.

三．C 程序的构成

例:

```
main( )                          /*主函数*/
{
 int a, b, sum;                  /*定义变量*/
 a=123; b=456;
 sum=a+b;
 printf (“sum is %d\n", sum);    /*输出变量*/
}
```

说明:

1)程序一般用小写字母书写;

2）每个程序必须要有一个 main（ ） （只能一个），称主函数;

注：C 程序是由函数构成的，函数是 C 程序的基本单位。

函数:系统提供的库函数；用户设计的函数。

3）程序体必须在{ }之间;

4）每个语句的结尾，必须要有“;”作为终止符。

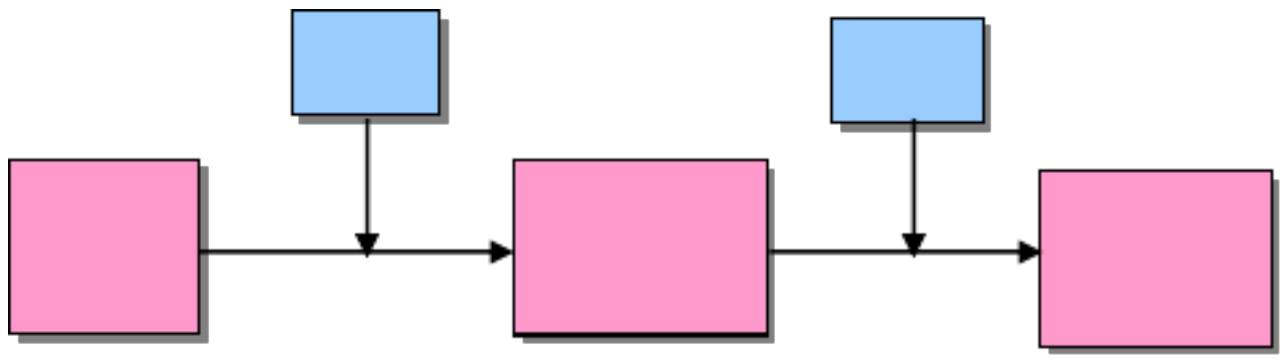
5）用/*……。. */ 作注释

四、C 程序的上机过程

1. 演示　书：P5 程序

2. 上机步骤　　书:P7

注:C 语言是一种编译语言，编译语言的特点:



C 编译程序　　Link 程序

3. DOS 下 C 语言的运行

目前 DOS 下运行 C 语言软件为：Turbo C　2.0

(1) Turbo C　2.0　的特点:

是一个把 ①编辑 ②编译 ③连接 ④运行 等全部操作集中在一个界面上。

(2) Turbo C　2.0 的操作及运行　　书：P8

五、算法的概念

1. 算法的概念　　书：P13

算法：程序的操作步骤。

在程序设计中：→ { 程序中的数据 / 操作步骤 }

有：  　　沃思 Nikiklaus Wirth

程序=算法+数据结构+程序设计方法+语言工具和环境

2. 计算机算法分为两大类：①数值运算算法　②非数值运算算法

(1) 数值运算算法:求数值解；通过运算得出一个具体值，如求方程的根等

注：数值运算一般有现成的模型，算法较成熟。

(2)非数值运算算法：用于事务管理，如图书检索、人事管理等。

3、常用简单算法

(1) 累乘　　即：1×2×3×4×5……×100

1×2 → S(结果)

S×3 → S(结果)

S×4 → S(结果)

⋮

S×100 → S（结果）

(2) 累加　　即：1+2+3+4+5……+100

1+2 → S（结果）

S+3 → S（结果）

S+4 → S（结果）

⋮

S+100 → S（结果）

(3) 找最大值　5，2，4,12，9

5 → max （max 放最大值,后面的数都和它比较）
2 〉 max 不成立
4 〉max 不成立
12 〉 max 成立：12 → max
9 〉 max 不成立

注：该算法可扩展为在一批数据中，找某一个数

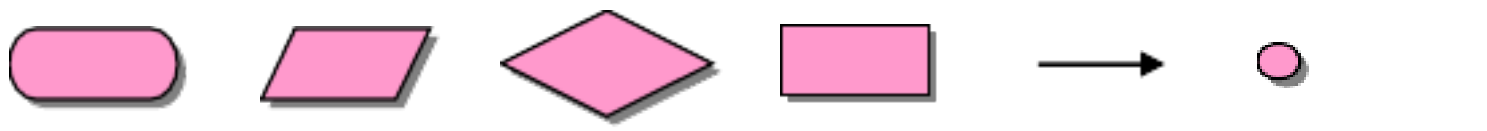
(4)其它算法

P15 例 2。2 例 2。3 例 2。4 例 2。5 课余自学(必须)

4、算法的表示

常用的有：①自然语言②传统流程图③结构化流程图④伪代码⑤PAD 图

(1) 传统流程图

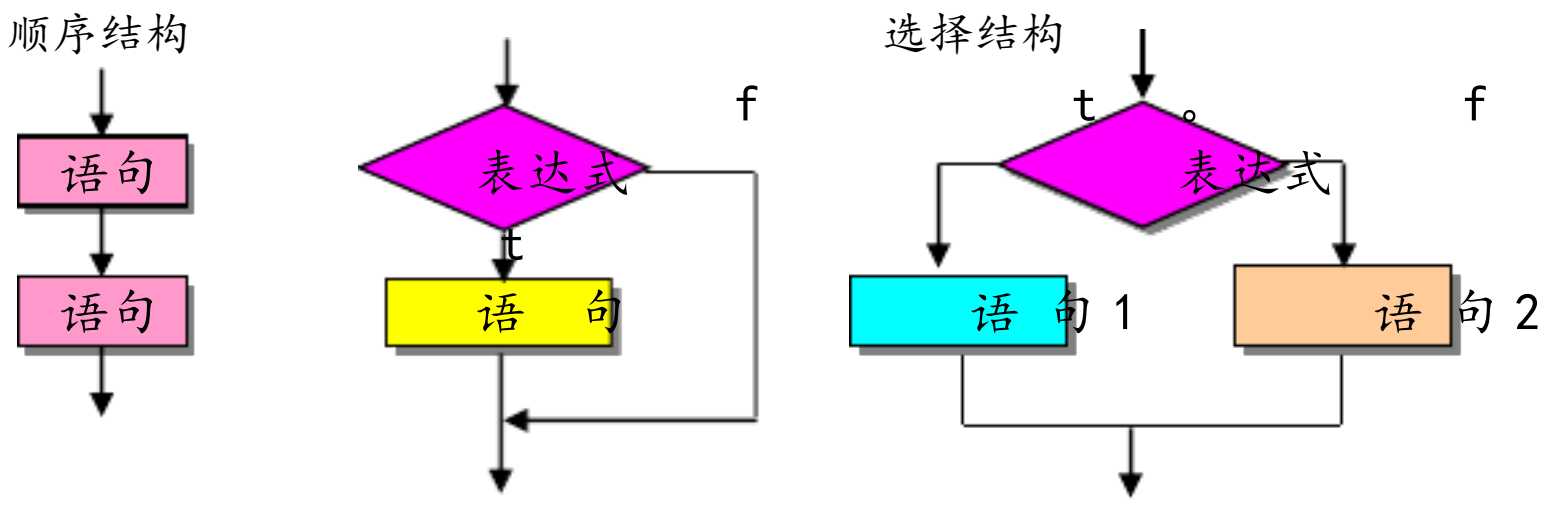


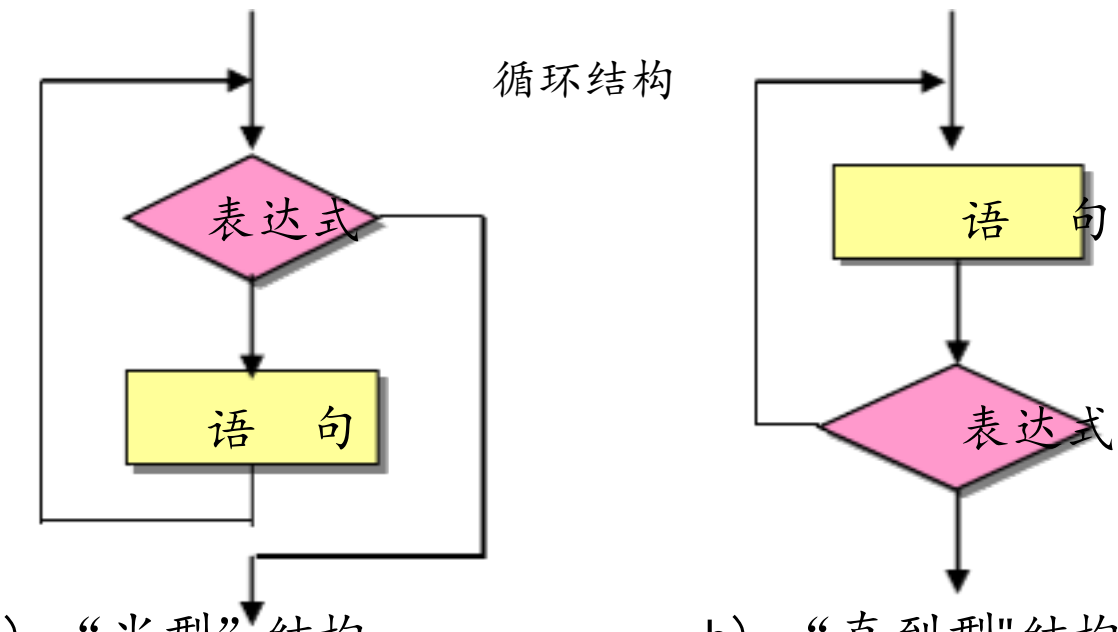
表示方法：P19 例

注：传统流程图对流程线的使用没有严格限制，难以实现结构化程序设计

为了限制流程线的滥用，提出三种基本结构：

①顺序结构 ②选择结构 ③循环结构



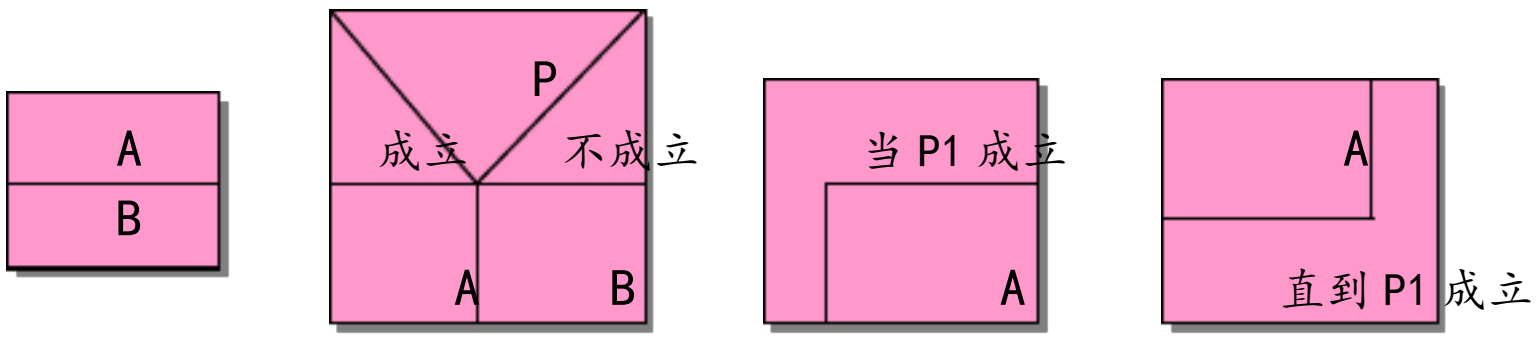


a)　“当型”结构　　b)　“直到型"结构

(2) N—S 流程图

特点：去掉了带箭头的流程线，全部算法写在一个矩形框内.

称 N-S 结构化流程图



顺序结构　　选择结构　　循环结构

作业：p37　2。4

上机:熟悉 Turbo C　2。0 的操作及运行

第三章 数据类型、运算符与表达式

本章要求:

本章是 C 语言的基本概念开篇，所介绍的概念是学习 C 语言的基础;

要求熟练掌握数据类型、运算符与表达式。

重点：①C 语言数据类型定义的方法和分类。

②C 语言运算符、表达式的使用以及它们的主要特点。

难点：自增、自减运算符、赋值运算符、逗号运算符及它们的混合使用

一. 数据类型

概念：算法处理的对象——是数据，而数据是以某种特定的形式出现。

在 C 语言中,数据有：常量、变量；一般它们都有一定的数据类型。

(C 语言的数据结构是以数据类型形式出现的)

1. 数据类型分类:

- 数据类型
  - 基本类型
    - 整型 int
    - 字符型
    - 实型（浮点型）
      - 单精度型 float
      - 双精度型 double
    - 枚举类型
  - 构造类型
    - 数组类型
    - 结构体类型 struct
    - 共用体（联合）类型 union
  - 指针类型 *
  - 空类型 void

说明：在程序中对用到的所有数据都必须指定其数据类型.

2. C 语言中的常量和变量

常量：——在程序运行过程中，其值不能被改变的量。

变量：——在程序运行过程中，其值可以改变的量。

说明:1）常量和变量，都要有一个名字表示它;

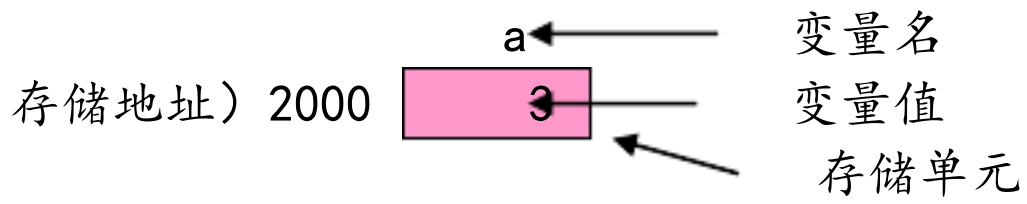
习惯上，符号常量(别名)名用大写，变量用小写。

符号常量的定义为： #define P 30

其它常量：直接用数据表示 例： 23 45。5 ‘e’ 3e+2

2）在 C 语言中，对变量要先定义，后使用。

3）如果一个变量被指定为一确定类型后，在 C 语言编译时，就能为其分配相应的存储单元.



地址的概念

3. 常用的基本数据类型

(1）整型

1)整型常量：十进制整数　123，—11，0　　非 0 开头

八进制整型　0123，—011　　0 开头

十六进制整型 0x11　0x20　0x0D　0XFF　0x4e　0x/0X 开头

说明：在一个整常量后面加一个字母：l 或 L，是长整型常量。22L　0733L　0xae4l

2)整型变量：

	定义	存储字节	数的范围
基本型	int　x	（2 字节）	-32768～32767
短整型	short　int　x	（2 字节）	—32768~32767
长整型	long　int　x	（4 字节）	-2，147，483,648～2，147,483，647
无符号型	unsigned int　x	（2 字节）	0～65535
	unsigned short　x	（2 字节）	0~65535
	unsigned long　x	（4 字节）	0~4，294，967，295

注：　数据在内存中是以：二进制形式存放

如：9　　为　　00001001

例：　x=13；　　x=015 ；　　x=0xD　　存储结构 见 P44 图 3。5

注：数据超过数据范围，会发生数据溢出　　例 3。2、3.3

(2) 实型（浮点数）

1)实型常量：　十进制整数　1.23　　。0123

指数形式　　1。23e3,1。00e—3

2)实型变量：

			数的表示范围 P46 表 3.2
	单精度	float　x	（4 字节）$3.4x10^{-38}$~3。$4x10^{38}$
TC 默认	双精度	double　x	(8 字节)$1.7x10^{-308}$～$1.7x10^{308}$

说明：实型常量不分 float 和 double,只有十进制表达方式

有的 C 编译系统 64 位存储方式，如尾加 f 或 F 可表示 32 位:324。567f

例:x=1。23；　　x=123e3　　P47 例 3。4（存储精度）

(3) 字符型

1）字符常量：

a) 用单引号括起来的一个字符　　‘a’‘A’

b)　转义字符（由“\专用字母”　）　　书 P48 表 3。3 例 3.5

\0 字符串结束符　　\n 换行　　\t 水平制表

\ddd 八进制数　　\xdd 十六进制数

2）字符变量：

char　c　　（1 字节）

说明：一个字符变量，只能放一个字符常量。实际是把该常量的 ASCII 值，送入字符变量中。

即：‘B'（66）　　c（字符变量）

书 P374ASCII 表 p50 例 3.6-7

c=‘B’　c 的内容为 66

故：字符数据可以进行算术运算；即：c=‘B’+2　为 68

字符数据在内存中的存储方式　书 P50

unsigned char　0—255　　char　-128-127 有符号数在 TC 中是用补码表示的

127(0x7f)　0111 1111　　128（0x80）　1000 0000

复习：补码的规定：

正数：其原码、反码、补码相同

负数:最高位为 1，其余各位为原码的反码（原码的相应位取反）,然后对整个数加 1。

140（0x8c） 1000 1100→1111 0011 + 1（取反加 1）→1111 0100→—0x74

-0x74 1000 1011 + 1(取反加 1) → 1000 1100 → 0x8c

(0x80 为负数—74 高位为 1)

3）字符串常量

用双引号括起来的字符 “a”，“ABFD” “a" =“a\0” 〈〉 ‘A’

说明：C 规定，在每一个字符串的结尾应加一个“字符串结束标志”(\0)，

以便系统判断字符串是否结束.

4. 各类数值间的混合运算

1）整型、字符型、单精度型、双精度型数据——可以混合运算

2）字符型、整型——可以通用

规则： 高 double ← float

↑ long

unsigned

低 int ← char，short

例如 10+’a'+1。5—8765。1234*’b’ -858873.593200 （TC 默认 6 位小数）

10+97+1。5-8765.1234*98

二. 运算符和表达式

C 表达式:由变量、常量、函数等运算对象和运算符组成的式子

C 运算符作用：

1）能把一些基本运算模块用单一的运算符处理

2）使用丰富的运算符，构成多种表达式

C 运算符有优先级，优先级高的先执行,同级的由结合规则决定其处理规则

C 运算符有两种结合规则：

1）从左到右的结合运算顺序；即:a=5+x－10;运算对象与左边运算符结合

2）从右到左的结合运算顺序；即：x=y=7；运算对象与右边运算符结合

C 运算符分类 书 P55

1. 基本运算符(算术运算符)

C 语言有 7 个基本运算符

1）单目运算符:（表示符号） 正号“+” 负号“—”

2）双目运算符:+、— 、*、/、%(模运算，求余运算,两侧为整数)

说明：

a)运算顺序的结合方向从左到右

b)求余（取模）运算符是对两个整数相除的余数，计算结果的符号与第一个数的符号相同。即：-5%3 值为-2、 -5%—3 值为—2、 5%—3 值为 2

2. 自增、自减运算符（优先级高于算术，右结合）

作用：——使变量的值增 1 或减 1

++i（——i）——在使用 i 之前，先使 i 的值加 1(减 1）,然后再使用 i.

i++（i——）—-先使用 i 的值,然后再使 i 加 1（减 1)。

例：i =4;

j=++i; i 值为 5,值为 5; j=i++; i 值为 5，j 值为 4

说明：1）自增、自减运算符++（—-)，只能用于变量，不能用于常量或表达式

即：5++ 、 （a+b）++ —— 不合法

2）运算顺序的结合方向从右到左

例

```
main (){
 int  i=3  z, x,y;
 x=y=3;
 //结果: z 值为 9   i 值为 6
 z=x+++y;
 //变量 x 右结合运算符++
 printf ("z=%d,x=%d, y=%d", z,x, y);
}
```

输出结果: z=6 x=4 y=3

如果: z= -x+++y; 输出结果: z=0 x=4 y=3 x 先为-4, 然后+y, 最后 x++

例:

```
z= (i++) +(i++) + (i++)
```

(不是 3+4+5=12)

3. 赋值运算符和赋值表达式(优先级低于算术, 右结合)

(1)“=” ——赋值运算符

作用: 将一个数据赋给一个变量 即: x=21

注意: 如果赋值运算符两侧的类型不一致, 在赋值时, 要进行类型转换 P61 例 3。9

即: float f ; f=23 f 值为 23.0000 符号位扩展

(2)复合的赋值运算符

“+=、-=、*=、/=、%=” —— 在赋值符“=”之前加上其它运算符

称复合运算符(也称为自反运算符)

例: a+=3 等价于 a=a+3

x*=y+8 等价于 x=x*(y+8)

y/=4 等价于 y=y/4

m%=3 等价于 m=m%3

说明:

C 语言采用这种复合运算符:

为了简化程序,使程序精练;提高编译效率。

(3)赋值表达式

格式:变量=表达式

表达式:

可以是 1)常量、变量、算术表达式;

2)赋值表达式

即: a=(b=5) 或 a=b=5 (运算顺序的结合方向从右到左)

例:

a=b=c=6 ; a=5+(c=6) (结果 a=11 c=6)

如果: a=3; 求: a+=a-=a*a; [有: 1] a*a; 9 2)a= a-(a*a); -6 3) a+=-6] -6+-6

a 为: -12 结果值为: -12

4. 逗号运算符和逗号表达式(优先级最低, 左结合)

用“,”将两个表达式连接起来。

格式: 表达式 1, 表达式 2 即: 3+5, 6*9;

例: 3+5, 6*9; 值为 54

(先求解表达式 1, 再求解表达式 2,最后逗号表达式的值为表达式 2 的值)

a=2*5, a*3; a 值为 10 结果值为:30 (先求: a=2*5,后求: a*3)

x=(a=2*5,a*3), a+4 a 值为 10 x 值为 30 结果值为: 14

说明:

使用逗号表达式的目的，只是为了得到各个表达式的值(常用于循环语句)

解表达式 1 的结果可能影响解表达式 2 的值

5. 强制类型转换运算符（优先级高于算术，右结合）

在要转换的数据前，用小括号括上要转换的数据类型

即： （int）x; （int）(a+b) （int）x+y

格式:（类型名）表达式

C 一般会自动进行数据类型转换 P56 例 3。8,用于不能自动转换的地方 x%3(float f)

6. 关系运算符和关系表达式（书 P87）(优先级低于算术，左结合，高于赋值)

(1)6 种关系运算符

< <= > >= （优先级高） == != （优先级低）

(2) 关系表达式

表达式 关系运算符 表达式

即：a>b a+b>=b+c (a=4)〈(b=8) ‘x’==‘y’

说明:

关系表达式值：是一个逻辑值,即:“真”或“假”

C 语言：1 —— 代表“真” 0 —— 代表“假”

7. 逻辑运算符和逻辑表达式（书 P88）(优先级低于关系，左结合)

(1) 3 种逻辑运算符 p89 表 5.1

&& （与） || （或） !（非）

(2) 逻辑表达式

表达式 逻辑运算符 表达式 p89

即： (a〉b) || (a+b>=b+c) (a=4) && (b=8) 1 || ∞ (1)

‘x’ && ‘y' 值为 1 1

说明:

逻辑表达式值：是一个逻辑值,即:“真"或“假" p90

优先次序（由高到低):

!（非） 算术运算符 关系运算符 &&和 || 赋值运算符

求值短路:5>3&&2||8〈4-!0 1||∞ （结果 1）

0&&∞ （结果 0） 如：a&&b&&c a<〉0 才判断 a&&b

8. 条件运算符（书 P97）

格式：表达式 1 ? 表达式 2:表达式 3

执行：先求解表达式 1，为真（非零），求表达式 2

为假（为 0）,求表达式 3

例： y=（a>b）? a : b;

说明:a）条件运算符要求有三个操作对象（称三目元运算符）

b)条件运算符优先于赋值运算符

y=（a〉b）? a : b; 先求 （a>b）? a : b 后赋值

c）条件运算符的结合方向为“自右至左”

（a〉b）? a : b 可写为： a〉b ? a : b

a〉b ? a : b+5 相当于： a>b ? a : (b+5)

a〉b ? a : c> d ? c : d 相当于： a>b ? a : （c〉 d ? c : d）

* 9. 位运算符 （书 P298）(一般掌握)

位运算：—— 是对字节或字中的实际位进行检测、设置、屏蔽、移位.

位运算符:

& —— 位逻辑与　　| ——位逻辑或　　^ —— 位逻辑异或

～ —— 位逻辑反　　〉〉 —— 右移　　〈〈 —— 左移

三、运算符小结

1. 左结合方向的运算符

主要有：算术运算符、关系运算符、逻辑运算符、逗号运算符

2. 右结合方向的运算符

主要有:自增、自减运算符、赋值运算符（复合的赋值运算符)、条件运算符

例:1) i=3; printf("%d, %d",i, i++); 输出：4, 3

2) a=1; b=1; c=2; 求：a=b + = c* = 5; 输出：a 为 11 b 为 11 c 为 10

3) a>b ? a〉 c? a:c: b>c ? b: c 相当于：a>b ? (a〉 c? a: c):(b〉c ? b:c)

（求最大值,如 a=2, b=8, c=5 结果为:8)

3. 运算符的优先次序（书 P375 附录Ⅲ）

()、! ++/——/—转移、算术、关系(<<=〉>=)、关系 (= = ! =)、&&、||、条件 (?)、赋值、逗号

例：1) —a++ 相当于 —(a++)

2) x=4, y=7 求：y+=++x-3 x 的值 5 y 的值 9

3) x=4, y=7 求：y=y+++x-3 x 的值 4 y 的值 9

4) a=7,x=2。5,y=4。7 求：s=x+a%3＊(int)(x+y)%*2/4 s 的值 2。500000

a%3*(int)(x+y)%＊2 =》 1（整型） / 4（整型） =》 0.25 =》0(整型)

5) a%3 && a%5 || a++〉++b

如：a=7, b=5 a 的值 8 b 的值 6 结果值 1 1||1

如：a=15, b=15 a 的值 16 b 的值 16 结果值 0 1||0

如：a=15, b=14 a 的值 16 b 的值 15 结果值 0 0||0

如：a=15, b=13 a 的值 16 b 的值 14 结果值 1 0||1

四、该章的主要例题

数： P39 例 3。1 P44 例 3。2 P44 例 3。3

字符： P49 例 3。5 P50 例 3.6 P51 例 3.7

五、该章的主要习题

P65 3.5 3.6 3。8 3.9 3.10 3。12

六、该章的主要上机内容

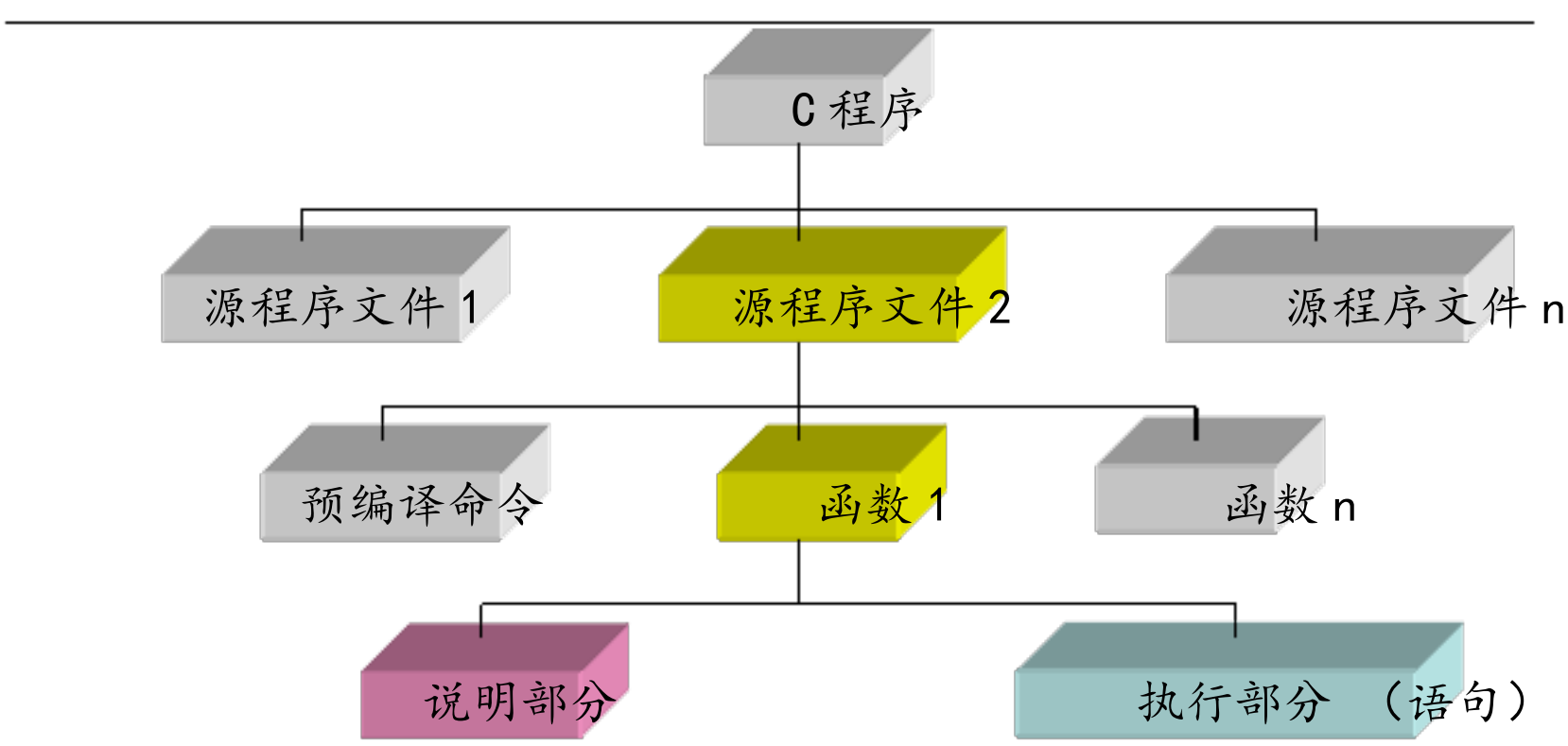
P39 例 3。1 P44 例 3。2 P44 例 3。3 P51 例 3。7

P65 3.6 3。8 3。10

第四章 简单的 C 程序设计（顺序程序设计）

知识点:
1、赋值语句；　2、字符数据的输入/输出；
3、格式输入/输出语句 printf 和 scanf 及其格式控制符；%c、%f、%d（输出宽度、对齐方式、其它格式控制符只作了解)
4、程序的结构框架。

重点:
1. 赋值语句;
2. 基本输入/输出语句 printf 和 scanf 及其格式控制符%c、%f、%d 等的意义.
难点:
　无
一、C 程序结构



语句的分类 P67　P68
1、控制语句
①条件：if　else
②循环：for()
③循环：while()
④循环：do　while()
⑤循环结束:continue
⑥终止循环或 switch：break
⑦多分支：switch
⑧转向：goto
⑨函数返回：return

2、函数调用语句
　printf（“a=%d", a）;
3、表达式语句
　i=i+1　表达式　　i=I+1;　表达式语句　　函数调用也可以认为是表达式语句
4、空语句：一个“;”
5、复合语句（用{}：分程序）

```
{ x=3;   x=x+1;
  print (“x=%x”, x)
}
```

二. 赋值语句

变量名=表达式

注: 是组成顺序结构的主要语句。

```
a=b;          if  (  max〈= x )  max =x ;
```

三. 基本输入/输出语句

C 语言本身不提供输入输出语句, 输入输出操作是由函数来实现。

即: printf putchar scanf getchar

注:(1) 这些函数构成一个标准的 I/O 函数库 (放在“stdio。h”中)

(2)如果要使用 C 语言库函数, 要用预编译命令“＃include”将有关“头文件”包括到用户源文件中。

```
#include 〈stdio。h〉          #include  “stdio。h”
```

当前目录 当前定义的目录中寻找

1. printf 函数 —— 格式输出函数

格式:printf (格式控制, 输出表列)

格式控制:—— 用双引号括起来的字符串, 也称为“转换控制字符串”

printf 的格式字符 书 P77 表 4. 1

输出表列: —— 需要输出的数据、表达式

即: printf(“x=%5.1f, y=%d, z=%c” , x, y, z)

d, i	带符号十进制形式
u	无符号十进制形式 例 3—3
c	字符形式
s	字符串形式
f	实数 (小数) 形式 默认 6 位小数

说明: 1)printf(“ y=%-4d” , y) 输出数据左对齐 (默认为右对齐)

2) 数据输出长度的修正;

l —— 输出长整型或双精度数据 即: printf (“%ld, %lf ” , i, y)

h —— 输出短型数据 即: printf(“%hu ” , a)

—m。n m 最小宽度 n 小数(实数) —左对齐

2. putchar 函数 —— 字符输出函数

向终端输出一个字符

格式: putchar (变量) putchar (x)

putchar (“\n") 输出一个换行符

3. scanf 函数 —— 格式输入函数

格式:scanf (格式控制, 地址表列)

格式控制: —— 用双引号括起来的字符串, 也称为“转换控制字符串”

scanf 的格式字符 书 P80 表 4. 3

地址表列: —— 由若干个地址组成的表列, 有: 变量地址 &x

字符串的首地址

& —— 地址运算符

功能:求变量的地址(只能用于变量)

&x —— 表示 x 变量值的内存单元的地址

即：scanf(“%d, %4d”, &a, &b)

说明：1）不带分隔字符的方式，系统将自动按格式说明截取数据

例： scanf (“ %4d%*3d%f ", &I , &p);

输入：12345678.93 时 得 I=1234 , p=8.93

“*"——作用是“虚读”，跳过相应的数据

2）带分隔字符的方式（分隔字符：空格、tab、回车、逗号）

即：scanf(“%d, %d”, &a, &b)

4. getchar 函数 —— 字符输入函数

从终端输入一个字符

格式：getchar （ ） 即：c=getchar （ ）

函数值是从输入设备得到的字符

四、该章的主要例题

1 已知矩形的两边长，求矩形的面积、周长

2 大小写字母互相转化

3 教材 P83 的例子

五、该章的主要习题 P84 4。5～4.9

第五章　选择结构程序设计

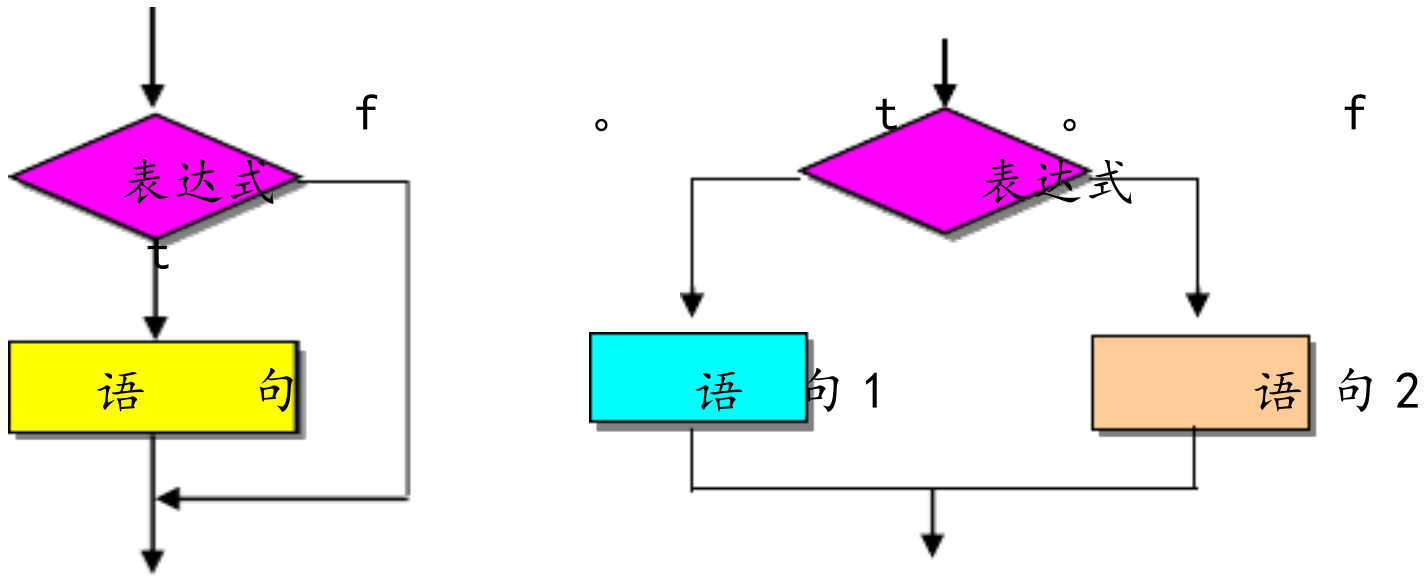
知识点:
1、关系、逻辑运算符、逻辑真/假;
2、关系表达式、逻辑表达式;
3、分支的概念及其算法描述;
4、三种 IF 分支语句;
5、条件运算符;
6、SWITCH 语句。(了解)

重点:
1、关系、逻辑运算符、逻辑真/假;
2、关系表达式、逻辑表达式;
3、三种 IF 分支语句.

难点:
1、逻辑真/假;
2、关系、逻辑运算优先级.

一、关系表达式及逻辑表达式在分支选择结构的作用
　　在程序设计中,表示一些复杂的条件
1、关系运算符及关系表达式（第三章）
2、逻辑运算符及逻辑表达式（第三章）
3、条件运算符(第三章)

二. 分支语句（书 P91）
　　从一个条件表达式中，选择语句的执行　（用 if 语句或 switch 语句来实现）



1. If 语句
(1) C 语言提供三种形式的 if 语句
1) if　(表达式)　语句
例:　if　(x>y)　printf　(“%d” , x);　　注意“;”位置
2)　if　(表达式)　语句 1　else　语句 2
例:　　if　(x>y)
　　printf　(“%d” , x);
　　else
　　printf (“%d” , y);　　注意分号“;”
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 if  (表达式n)  语句n
        else     语句n+1
```

例:p92 图 5。6

（2）if ()和 else 后的执行语句可以使用复合语句 { }

例 5.1、5.2 p93 if 结构 将输入的三个数从大到小排列输出

例 5.3 p95 if 嵌套结构 计算函数值

(3) 用条件运算符代替 if 语句

例: if (a>b)

```
        y=a;
    else
        y=b;
```

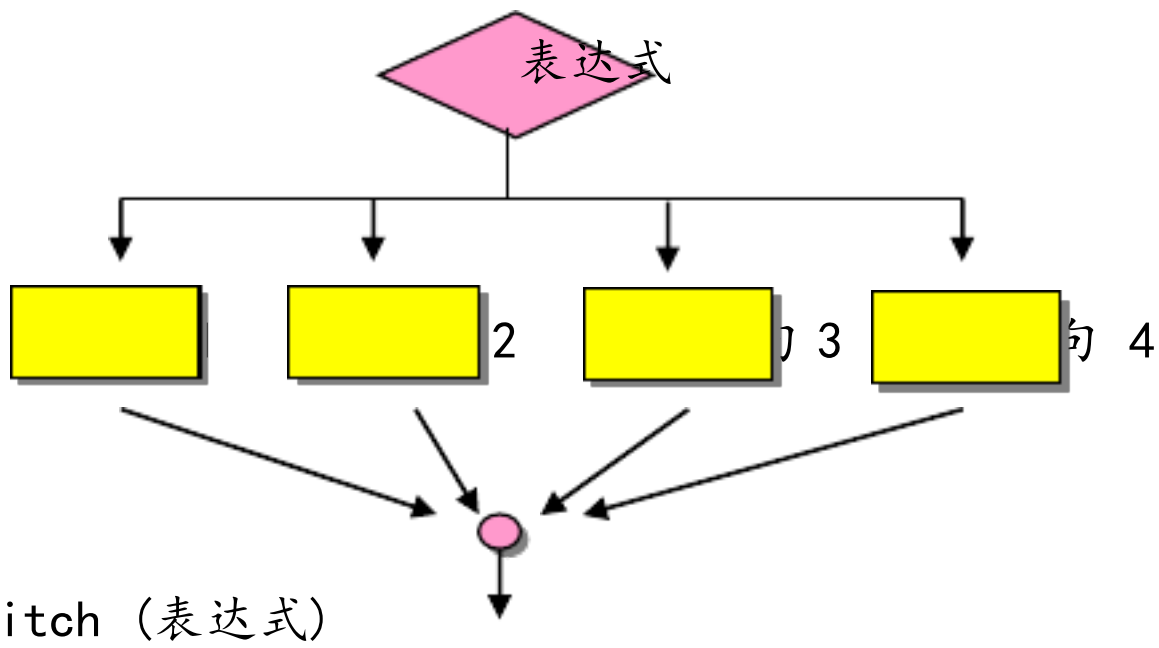
条件运算符

可以用: y=（a〉b）？a : b;

例 5.4 p98 条件运算符代替 if 结构 将输入字符中的大写转换成小写输出

2. switch 语句（了解）

switch 语句——是多分支选择语句



格式: switch（表达式）

表达式=常量表达式 n 执行语句 n

```
{ case  常量表达式1 : 语句1;
                      break;
  case  常量表达式2 : 语句2;
                      break;
  case  常量表达式3 : 语句3;
                      break;
        ┆
  case  常量表达式n : 语句n;
                      break;
  default           :语句n+1;
```

例:四级（ABCD）转化成百分制

```
#include <stdio。h〉
main()
{char grade;
 scanf(”%c”, &grade）;
 switch (grade)
   { case ’A’:printf(”85~100\n"); break;
```

```
        case 'B': printf (” 75~84\n"） ; break;
        case ’ C': printf (” 60～74\n” ） ; break;
        case 'D’ : printf(” <60\n” ） ;break;
        default:printf (” error\n” ） ; }
    }
```

例：百分制成绩转化成五级（ABCDE）五级记分 5—4-1

三、程序举例

例 5.7　p103　　运费计算(自学)

p101　例 5。6　求一元二次方程根(自学)

p100　例 5。5　判断某一年是否是闰年（自学）

五、该章的主要习题　　P104　5。3~5.10

该章的主要上机内容

P91　例 5。2、P100 例 5。5，P101 例 5.6

P104　习题 5。5、5.6、5。9　　其他：上机实验指导书中内容

第六章　循环结构程序设计

知识点:
1、循环的概念及其算法描述；　2、goto 循环;
3、while 循环；重点　　4、do—while 循环；重点
6、for 循环;重点 难点　　7、break、continue 语句;
8、各种循环结构中的执行循环体、判断循环条件的顺序；重点
9、各种循环结构在一定条件下的转化。重点　难点

一、循环结构的概念

循环语句主要用于：重复执行的操作

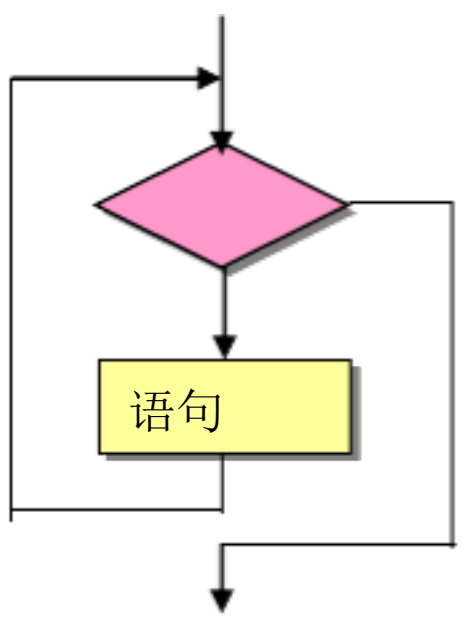
1．C 语言在实现循环过程——可使用以下语句:

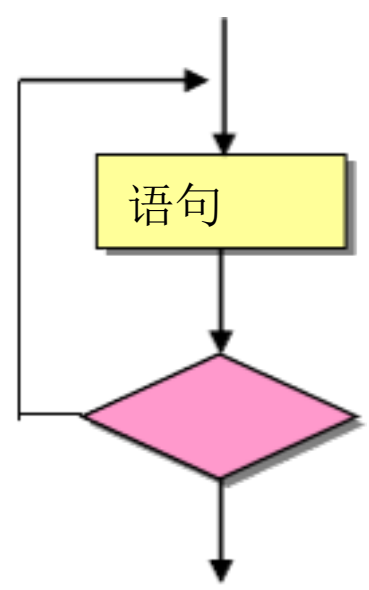
a）　用 goto 语句和 if 语句构成循环　　b）　用 while 语句

c）用 do—while 语句　　d）　用 for 语句

2．循环可分为二种类型结构:

a）　“当型"结构　　b）“直到型”结构





二．循环语句

1．用 goto 语句和 if 语句构成的循环

（1）goto 语句——无条件转向语句

格式:goto　语句标号

语句标号：用标识符表示

例：goto　next;　　如:　　goto 123 ；(不合法)

（2）goto 和 if 语句构成循环

一般较少使用（不符合结构化原则）

例：求 $\sum_{i}^{50} 2n$

```
main ( )
{ int  i ,  sum=0 ;
 i=1 ;
 loop:  if ( i <=50 )
 { sum=sum+2*i ;
   i++ ;  goto loop ; }
```

```
        printf ( "%d\n" , sum ) ;
    }
```

2. while 语句　　——实现“当型"循环结构

格式: while (表达式) 语句

例: 求 $\sum_{i}^{50} 2n$

```
main ( )
{ int  i ,  sum=0 ;
  i=1 ;
  while (i<=50 )
  { sum=sum+2*i ;   i++ ;  }
   printf ( "%d\n" , sum ) ;
   }
```

说明:

语句可为复合语句（用 { } 构成）

3. do—while 语句　　——实现“直到型”循环结构

格式: do 语句
　　while (表达式);

例: 求 $\sum_{i}^{50} 2n$

```
main ( )
{ int  i ,  sum=0 ;
  i=1 ;
  do
   { sum=sum+2*i ;
    i++ ;   }
  while  (i<=50 ) ;
  printf ( "%d\n" , sum ) ;
}
```

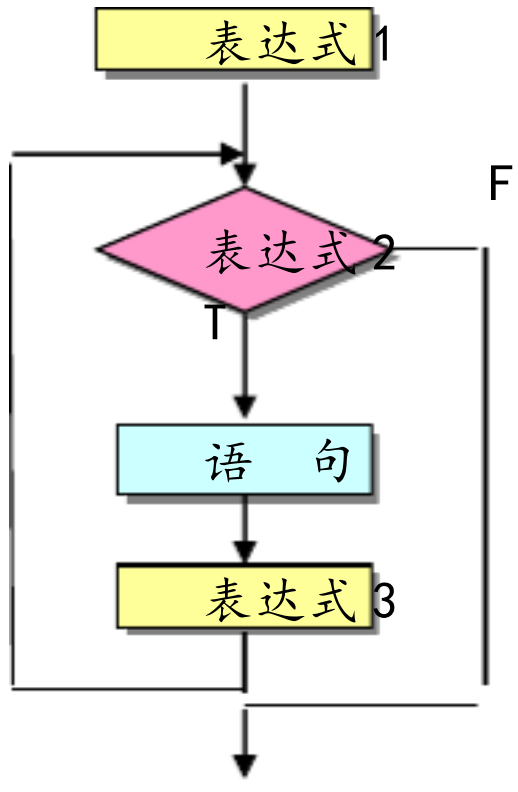
do while 和 while 比较 p109 例 6.4

4. for 语句

可以用于循环次数已知和循环次数不确定 （可以代替 while 语句）
（应用广泛，而且使用较活）

格式: for (表达式1 ; 表达式2 ; 表达式3) 语句

过程:



说明:

表达式 1————循环变量赋初值
表达式 2————循环结束条件
表达式 3————循环变量增值

即：for （循环变量赋初值，循环结束条件， 循环变量增值） 语句

例：求 $\sum_{i}^{50} 2n$　　for （ i=1 ； i〈=50 ； i++ ） sum=sum+2＊i;

for 语句的其它表示

(1) 用 while 来表示

```
表达式1
while （ 表达式2 ）
{ 语句
  表达式3;
 }
```

(2) 表达式可以省略,但分号不能省略

a） 省略表达式 1

```
i=1
for （ ; i〈=50 ; i++ ）   sum=sum+2*i
```

b) 省略表达式 2——产生死循环

```
for （ i=1;  ;i++ ）   sum=sum+2*i
```

c) 省略表达式 3

```
for （ i=1; i〈=50 ;）
  { sum=sum+2*i;
    i++;
  }
```

d) 省略表达式 1 ， 省略表达式 3

```
i=1                                  i=1
for （ ; i<=50 ; ）                    while （ i〈=50 ）
  { sum=sum+2＊i;        相当于          {  sum=sum+2＊i;
    i++;  }                              i++; }
```

(3) 表达式 1, 表达式 3 可以是:　　a) 简单的表达式 b)逗号表达式

表达式 2:　　一般是关系表达式或逻辑表达式

即:　　for （ sum=0 , i=1 ； i〈=50 ; i++,i++ ） sum=sum+2＊i;

或:　　for （ sum=0 , i=1 ； i<=50 ; i=i+2 ） sum=sum+2*i

(4) 循环体语句可使用 break 和 continue 语句

break: 可以用来从循环体内跳出循环体（提前结束循环）

continue: 用于跳过循环体中下面尚未执行的语句,接着重新执行循环的判断

.p115　例 6。5 将 100-200 间不能被 3 整除的数输出

三、程序举例

用多种循环方法实现

1 由泰勒展式计算 sin(x) 的近似值

2 。p115　例 6。6　求 π 的近似值　1-1/3+1/5-1/7+

3。p115　例 6。7 求 Fibonacci（斐波纳契）数列 40 个数

$F_1=1$　　(n=1)

$F_2=1$　　(n=2)

$F_n=F_{n-1}+F_{n-2}$　(n〉=3)

4。p115　例 6。8 判断 m 是否是素数

5.p115　例 6.9 求 100—200 间的所有素数

6.例 求任意输入 10 个数中的最大数 6-max

四、该章的主要习题

P120 6.1~6.15

五、该章的主要上机内容

P115 例 6.5、例 6。6、例 6.7、例 6.9

P120 习题 6。4、6.7、6。9、6。11、6。14

其他：上机实验指导书中内容

作业解答：

5-8 利润提成:类似所得税

第七章 数组

1、数组的概念及数组的定义;
2、在程序设计中的如何使用数组;
3、字符数组是 C 语言存放字符串的主要方法，并注意字符串结束标志的规定。

重点：①数组的组成和特点.
②一维数组和二维的定义、赋值。
③字符串与字符数组的关系以及数组在程序中的使用。
难点:①数组元素存储格式对程序设计的使用.
②字符串在数组中的组成以及字符串操作语句的使用。

C 语言：数据类型 { 基本类型 / 构造类型 { 数组类型 / 结构体类型 / 共用体类型 } / 指针类型 }

构造类型:是由基本类型按一定规则组成的类型

一. 数组的概念

数组：—— 1）是一组有序数据的集合

2)这个集合用一个数组名来表示

3）数组的元素序号用一个下标值来表示

如：a [0]， a[1] ， a [2]， a [3] b[1][1] ， b[1][2] ，b [2] [1] ， b [2] [2]

注意：a）数组名后，用方括号；下标从 0 开始。
b）下标值——可以是常量表达式
常量表达式：包括常量和符号常量(用#define 定义的)，但不能包含变量。
C 语言：不允许数组进行动态定义。

```
即： int  n ;
     scarf (“%d”, &n);
     int  x [n];      /*定义数组 */
```

c）数组分为：一维、二维等，主要由下标值的多少来确定.

二. 一维数组

1. 定义

格式： 类型说明符 数组名[常量表达式]

类型说明符:——表示数组的类型

int , short , long , unsigned , float , double , char 等

数组名： ——表示各数组元素的统一名字，定义方法和变量名相同

常量表达式：——表示数组的长度

即：int a[10]

注意:C 语言不提供数组下标越界的保护，设计时要注意。

2．数组元素的初始化

数组在定义的时候，同时，可对数组元素进行初始化

格式： static int a [10] ={0,0,0,0,0,0, 0, 0,0, 0} 静态定义

a[10]= {0, 1, 2,3,4,5, 6,7, 8, 9}

说明:

1）static 数组不赋初值，系统会对所有数组元素自动赋以 0 值。

即：int a[10] ={0, 0, 0,0,0,0, 0, 0,0, 0}； 等价于： static int a [10] ;

2）对全部数组元素赋初值，可以不指定数组长度。

即： static int a [ ] = {0, 1,2,3, 4,5};

等价于： static int a[6] ={0, 1, 2, 3, 4, 5} ;

3）可以部分赋初值

即：int a [5]={1,3};

3．数组元素的引用

C 语言规定:只能逐个引用数组元素，而不能一次引用整个数组。

4．程序举例

书 P124 例 7。2 例 7。3

三．二维数组和多维数组

1．定义

格式： 类型说明符 数组名 [常量表达式] [常量表达式]

类型说明符:——表示数组的类型

int , short , long , unsigned , float , double , char 等

数组名： ——表示各数组元素的统一名字

常量表达式：——表示数组的长度

即：int a[5] [6]

说明:

a)每一维的下标从 0 开始，整个数组元素为：5×6=30 个

b）C 语言中,二维数组中元素排列的顺序是按行存放。

即：先存放第一行的元素，再存放第二行的元素

$$a=\begin{pmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \\ 1 & 2 & 9 \end{pmatrix}$$

存储单元对应表：

→

3 a [0]
[0] a
[0] [1]
[0] a [1]
[0] a [1]
[0] a
[1] [1]
[1] a [1]
[1] a[2]
[0] a[2]
[1] a[2]
[2]

c)可以计算数组元素在数组中的序号

假设 m×n 的数组 a,

计算 a [i] [j]序号的公式： i×n+j

如: a[0] [1] 0×3+1=1（第二个） 如 3×3

a[1] [2] 的序号是： 1×3+2=5

a[2] [2] 2×3+2=8

d）多维数组的定义

格式： 类型说明符 数组名[常量表达式] [常量表达式] [常量表达式]

即: 类型说明符 数组名[长度 1] [长度 2] [长度 3] …

int a[2] [3][4]

2. 二维数组初始化

1)两种基本方法:

(a)分行给二维数组赋初值

static int a [2][3] ={{1,2,3} , {5, 6, 7}}

（b）所有数据写在一个化括弧内

static int a[2] [3] ={1,2, 3,5,6, 7}

2）可以对部分元素赋初值

static int a [2][3] ={ {1}, {3} }

即:

$$\begin{bmatrix} 1 & 0 & 0 \\ 3 & 0 & 0 \end{bmatrix}$$

static int a[3] [4]={{1},{3,1},{0, 0,9}}

即:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 0 & 0 & 9 & 0 \end{bmatrix}$$

3）定义数组时,第一维的长度可以不指定，但第二维的长度要指定（编译系统自动计算）.

等价于:
- static int a[ ] [3] ={1, 2, 3,4, 5,6,7, 8, 9}
- static int a [3][3] ={1,2,3,4,5, 6,7, 8, 9}

等价于:
- static int a [ ] [4]= { {1} , {3,1}, {0,0,9} }
- static int a[3] [4]= { {1}, {3, 1}, {0,0,9} }

4. 程序举例

书 P128 例 7。4 二维数组行列转换

例 7.5 3×4 矩阵求最大元素及行列号

四. 字符数组

存放字符数据的数组；每一个元素存放一个字符.

1. 定义

char 数组名 [常量表达式]

即: char c [10];

说明: 由于字符与整型是互相通用的,可使用 int c[10] ; 定义。

2. 字符数组的初始化

1）static char c [5] ={ ‘a', ‘b’, ‘c’, ‘d’, ‘e'};

注意: static char c[5]={ ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’ }; （错误，多一个元素）

static char c[5]={ ‘a’, ‘b’, ‘c’ }; （正确） 后面元素自动定为空格

2）static char c[ ]={ ‘a’, ‘b', ‘c’, ‘d’, ‘e’ };

系统自动根据初值个数确定数组长度

3. 字符数组的引用

一个字符数组元素，代表一个字符.

即:x=c [2] x 的值为“c"

4. 字符串的使用

C 语言中，将字符串作为字符串来处理。

规定：一个“字符串结束标志”，以字符‘\0'，在遇到字符‘\0'时，表示字符串结束。

1）使用字符串常量进行数组初始化

static char c [ ] = { “I am happy"};

或 static char c[ ]=“I am happy”;

等价于 static char c[ ]={ ‘I', ‘ ‘, ‘a', ‘m’, ‘h’, ‘a’, ‘p’, ‘p’, ‘y’,’ \0’ };

注：(a)系统对字符串常量自动加一个‘\0’

（b）字符数组并不要求它的最后一个字符为‘\0’

2）数组长度始终保证大于字符串实际长度

static char c [10]={“ABCDE”}; （第 5 个元素\0，第 6 个开始为空字符 ASCII 为 0）

static char c [10] ={ ‘A’, ‘B’, ‘C’, ‘D’, ‘E'}; （后面 5 个元素为空字符 ASCII 为 0）

例 7—6 P131: 字符数组引用

例子 打印图形 (J5_1.C)

```
#define NL printf("\n")
main()
 { char a[10]= {'*',' *', '*' ,' *','*',' *','*',' *' ,'*',' *'};
  int i, j;
   for (i=0;i <=4; i++)
      { NL;
        for (j=1; j<=i;j++)
           printf (" ”);
        for (j=1; j<=10-(2*i-1) ;j++)
          printf (” %c”, a [j]);
       }
  }
```

图案层数

图案前空格

* 个数

```
***** ***
**** ****
 ***** *
  *** *
   * *
    *
```

5. 字符数组的输入输出

1)两种方法:

a）逐个字符输入输出。 用格式符“%c”进行 scanf (“%c”, &c[i]);

b）整个字符串一次输入输出。　用格式符“%s”进行

例：　static　char　c[ ]={“abcde”};
　　printf(“ %s” , c) ;　　(为数组名,不能为“ %s” , c [0])

2）可以用 scanf 函数输入一个字符串

```
static  char  c[6];
      scanf  ( “%s”, c );
```

注意：键盘输入不能超过 5 个字符

c 不用加地址符&,即&c;因数组名代表数组的起始地址

6. 字符串处理函数

在 C 的函数库中提供了一些用来处理字符串的函数

(1) scanf 和 printf 函数在字符串的使用

使用 scanf 和 printf 函数对字符串的处理时,调用语句中的输入或输出项使用的是字符数组名。(即使用字符数组名的首地址)

```
例: main  ( )                              (j5_9。c)
    {  char  s1[3] , s2[5];
       scanf ( “%s%s”, s1 , s2 ) ;
       printf (%s\n%s\n”, s1 ,  s2 ) ;
    }
```

输入:

ABC DEF HK　　(即: 希望 s1=“ABC ” s2=“DEF HK"

输出:

ABC
DEF

说明:C 语言规定, scanf 函数遇空格或回车就结束本次输入。

(2) puts 函数

格式: puts (字符数组)

即:　puts(str)

将一个字符串 str(以‘\0’结束的字符序列)输出到终端

(3) gets 函数

格式:gets (字符数组)

即:　gets (str)

输入: abcde

读取字符串并把它们依次放到 str 指向的字符数组中去.

得到一个函数值, 该函数值是字符数组的起始地址。

str 是字符数组的首地址, 它读取字符串直到遇到换行符或 EOF。

换行符或 EOF 不进入字符串,被转换为“\0”, 作为字符串结束符

(4) strcat 函数

strcat(str1 , str2);　　str1, str2 是字符数组

连接两个字符数组中的字符串, 把 str2 接到 str1 的后面, 结果放在字符数组 str1 中.

例: static char str1[20] ={“ABCD EF” }

static char str2[ ]={“gggg” }

执行 strcat(str1, str2)的结果为:ABCD EFgggg　　书 P136

(5) strcpy 函数

strcpy (str1 , str2) ;　　功能: 把 str2 的内容复制到 str1 中

例: strcpy (str1 , “abcd”);

strcpy(&a [n] , &a [n+1]); 作用:将 n 个字符以后的字符顺序地向前移动一个序号。

说明: a) str1>=str2

b) str1 必须是数组名,str2 可以是数组名或字符串常量

c) 可以解决字符串常量不能赋值给字符数组

即:str1={“abcde"} ; str1=str2;(不合法) 可用 strcpy (str1,str2)

注: strcpy(str1,str2, 2); 是将 str2 中前面 2 个字符拷贝到 str1 中去, 然后再加一个‘\0’。

(6) strcmp 函数

strcmp(str1 , str2);

功能: 把 str1 与 str2 的内容进行比较(按 ASCII 码的大小进行比较)

如果: str1=str2, 则函数返回 0

如果: str1>str2, 则函数返回正整数

如果:str1<str2,则函数返回负整数

(7) strlen 函数

strlen (str) 测试字符串长度, 不包括‘\0’

(8) strlwr 函数

strlwr (str) 将字符串中大写字母换成小写字母

(9) strupr 函数

strupr (str) 将字符串中小写字母换成大写字母

7. 程序举例: 书 P138 例 7.8 例 7.9

五、该章的主要例题: 例 7。1、例 7.2、例 7。3、例 7。4、例 7。5、例 7。6、例 7.8、例 7.9

六、主要习题: 7.1、*7.2、7.3、7.5、7.6、*7。9、7。11、*7。13

七、主要上机内容: 例 7.2、例 7.3、例 7。4、例 7。5、例 7.7、例 7。8、7。1、7.3、*7。9、7。11

其他: 上机实验指导书中内容

第八章　函数与程序结构

1、函数的一般概念:函数形参、实参、返回值，函数的类型;
2、动态存储变量与静态存储变量　　3、变量的生存期与作用域

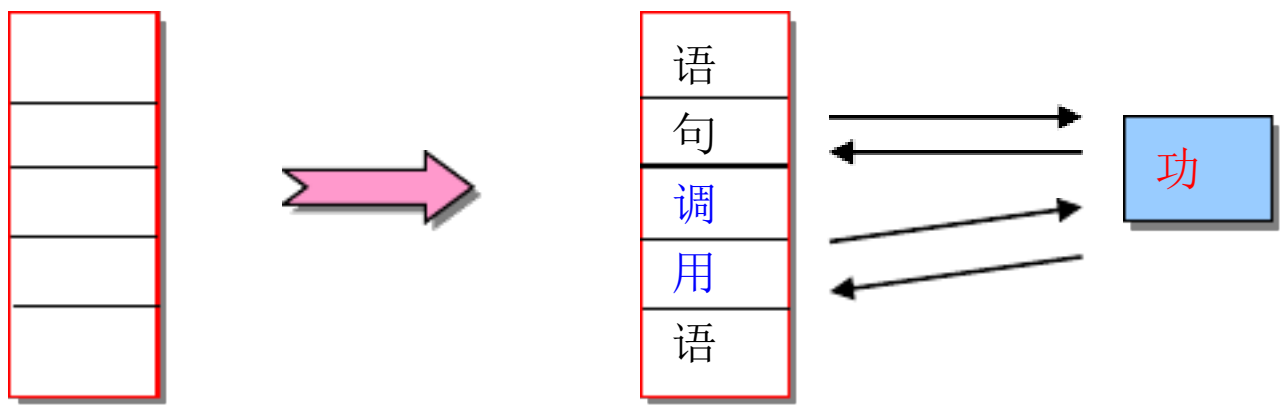
重点：①函数的定义和调用及使用函数的特点。
　　②函数间的参数传递(值传递和地址传递）.
　　③变量存储作用域、编译预处理的使用。
难点：①函数参数传递（数值、地址）及形参、实参的使用;
　　②变量存储作用域及自动变量与静态变量的区别.

一、函数概述

一个较大的程序：可由若干程序模块组成，每一模块用来实现一个特定的功能。
——用子程序可实现该模块功能。
——在 C 语言中：子程序称为：子函数

在 C 程序中：

函数的作用：　　减少重复编写程序段的工作量.



标准库函数:

用户自定义函数:

一般自定义函数和自定义函数库(用户先已编好的)

二、函数定义(子函数的设计)

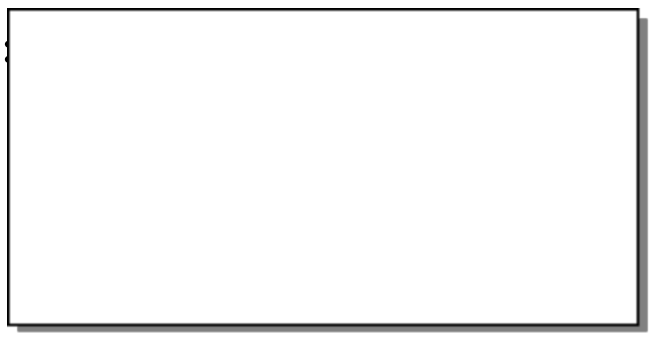
1. 函数定义

格式:　类型定义符　函数名(形式参数表列)
　　{　函数体　}

说明:1）类型定义符：函数返回值的数据类型
　　a）　int ，　char ，　float ，　double 默认 int
　　b）函数无数据返回，使用 void 作类型定义符
　　2）函数名：函数的名称（用户取的：合法的标识符）
　　3）形式参数表列：函数调用时传递信息的通道
　　有两种表示方法：　a)(参数说明 形式参数）
　　b）（形式参数）

传统风格

参数说明

例：  ，f  at add (x , y)
x , y ;
y ;
(x);

a). b)

2. 函数的返回值

- 一般通过函数中的 return 语句获得。
- 如果不需要返回值，可以不要 return 语句.

为了明确表示“不带回值”，可以用“void”定义“无类型"

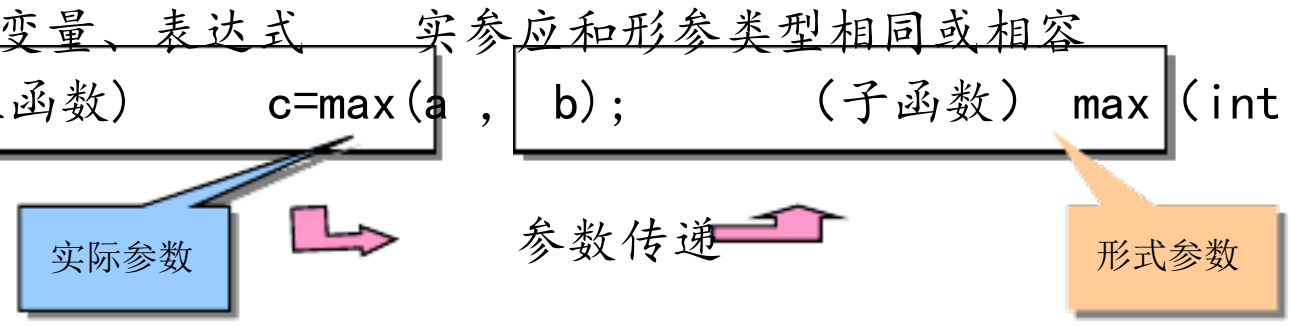
例:
```
main ( )      /*(打印平方表)*/
{ void  prin  ( int x ) ;
  int   i ;
  for  ( i=1 ; i <=10 ; i++)  prin  ( i )  ;
}
void prin ( x )
int   x ;
{
 printf(“%d\t%d\n", x,x*x);
}
```

三、函数间的参数传递

在一个主调函数调用一个子函数时，须把主调函数的参数传递给子函数，称该参数为“实际参数”.

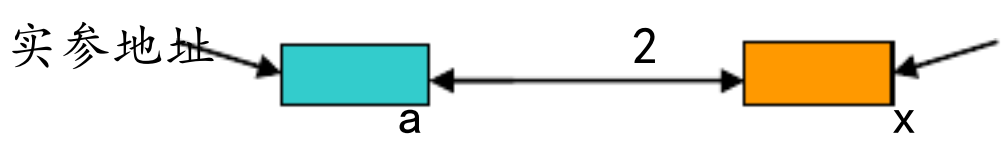
实参可以是常量、变量、表达式　　实参应和形参类型相同或相容

P146 例 8。2 （主函数）　　c=max(a , b);　　（子函数） max(int x , int y)



（1）数值传递方式

特点：实参通过复制的方式传递给形参(占用不同的内存空间)



调用函数的实参：给出具体数据

（2）地址传递方式

特点：参数传递不是数据本身，而是数据的地址（使用同一地址空间）

调用函数实参：一般是数组名或指针变量

四、函数调用及值的返回

1. 函数调用

格式：函数名 （实参表列);

调用的过程:

a）通过函数名，找到定义的函数

b)将实际参数传递给被调函数的形式参数

(实际参数与形式参数一一对应,既一一传递数据)

许多 C 版本是按自右而左的顺序求值

例: P150 例 8。4

```
main  ( )
 {  int  i=2,  p;
    p=f(i, ++i) ;                    /* f 为 f(3, 3) */
    printf ( "%d" ,  p ) ;
 }
    int  f(int a ,  b)
     { int  c;
       if (a〉b)   c=1;
       else if (a= =b)  c=0;
       else c= —1;
      return ( c );
    }          运行结果:0
```

说明:(1) 应避免这种情况的产生, 可写为:

①自左而右顺序

```
j=i;
k=++i;
p=f(j, k) ;
```

②自右而左顺序

```
j=++i;
p=f(j,  j) ;
```

(2) 在 printf(" %d, %d " , i, i++); 中同样存在

当: i=3; 结果为: 4, 3

2. 函数调用的几种方式

（1）函数语句

（2）函数表达式

（3）函数参数

3．函数说明

作用：在函数使用前对函数的特征进行说明的语句．（用户自定义函数）

- 对已定义的函数的返回值进行类型说明。
- 告诉系统在本函数中将要用到的函数是什么类型。
- C 语言规定: 以下几种情况可以不使用函数说明

a）函数的值（函数的返回值）是整型或字符型（系统自动按整说明）
b）如果函数定义在调用函数之前,可以不必加以说明

在函数说明和函数定义中:

a）类型定义、函数名要相同
b)形式参数标识符可以不相同

如: main （ ）

```
{
  double  abc （ int x , float y ） ;  /*函数说明*/
  ————
}
  double  abc（ int a ,  float b ）   /*函数定义  a, b 与 x, y 不同*/
  函数体
}
```

非 int 形参的函数必须在调用前进行函数说明 P148 例 8。3、8。5

不使用函数说明例子

```
例: main  ( )                          float  abc  ( float x ,  float y )
    { int sum , a=2 , b=5;                 { x=x+y ;                    定
      sum=add ( a , b ) ;                    return (x);                义
      ---                              }
      }                                main ( )
      int  add ( int x , int y )                { float  sum ,  a=2.0 ,  b=5。
0;
      {  x=x+y;                  定义     sum=abc ( a ,  b );
        return (x) ;                          语句  }
      }
```

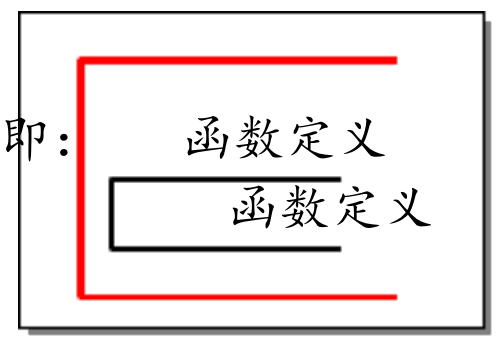
说明: 使用库函数, 一般在文件开头用 # include 命令

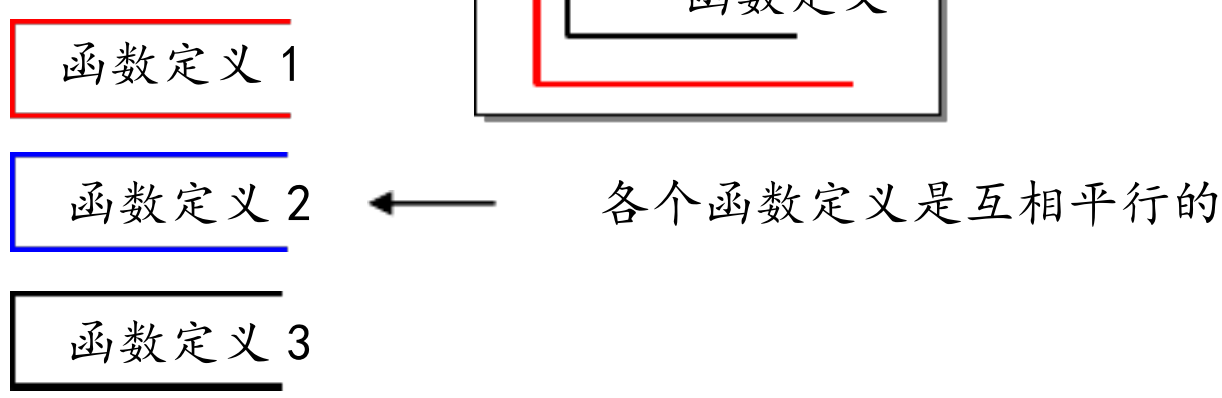
即: #include “math。h"

```
main  ()
{  …
  a=sqrt(x)
  …    }
```

4. 函数的嵌套调用

1）C 语言函数不能嵌套定义　即:

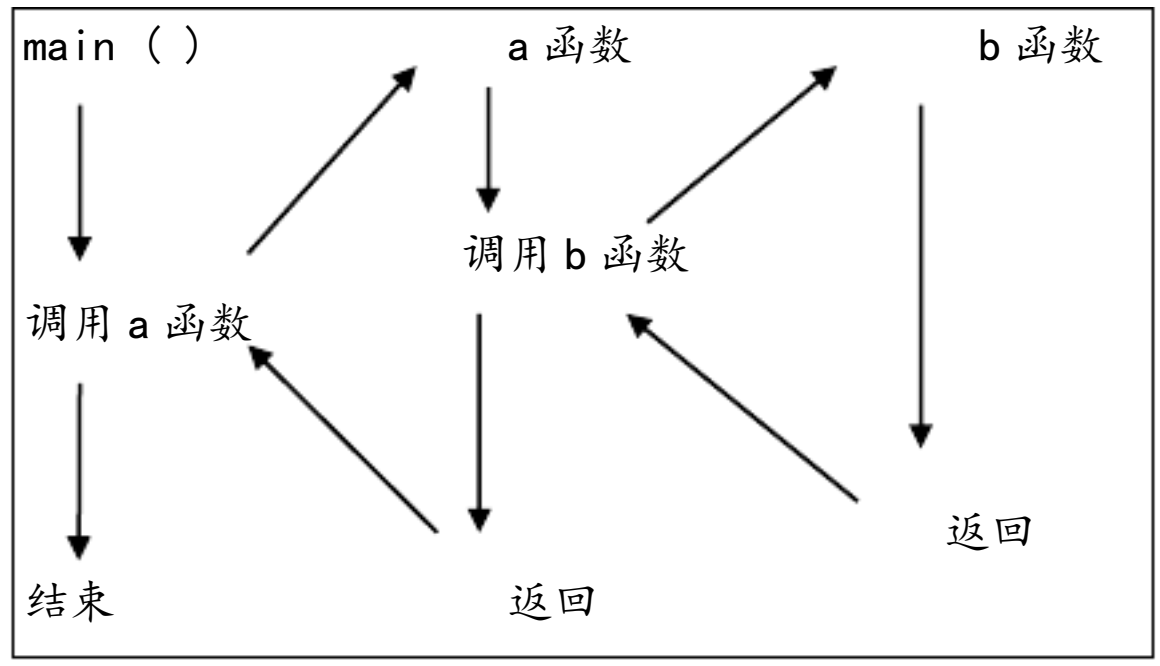




各个函数定义是互相平行的

2）C 函数允许嵌套调用, 即在调用一个函数的过程中, 又调用另一个函数。

示例:



*五、递归调用　一个函数自己调用自己的过程

递归: 由递归方式与递归终止条件两部分组成

即: 一个递归的问题可以分为: 首先“回推”,然后“递推”

在递归过程中, 必须具有一个结束递归过程的条件.

例: N!

$$\text{递归公式} = \begin{cases} 1 & (N=0,\ 1) \\ N * (N-1)! & (N>1) \end{cases}$$

子函数定义

```
long  fac( int n )
```

求: 4!

```
sum=fac(4);
```

```
{ long  fa;
  if
  else
   fa=n*
return(fa);   }
```

第一步　fac(4)　fa=4＊fac（4—1)=4*fac（3)　fa=4*3*2＊1＊1
第二步　fac（3）　fa=3＊fac（3-1)=3＊fac（2）　fa=3*2*1＊1
第三步　fac(2)　fa=2*fac（2—1）=2*fac（1）　fa=2＊1*1
第四步　fac（1）　fa=1*fac（1—1)=1*fac(0)　fa=1＊1
第五步　fac（0）　fa=1

回推　递推

六、函数参数的使用(实参或形参)

- 实参：在调用函数中；　形参:在函数定义中
- 函数参数：1）数值　2）变量　3）数组元素　4)数组名　5）指针

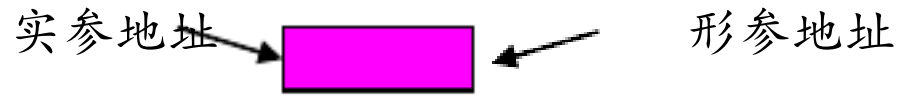
1．函数间的参数传递

（1）数值传递方式

调用函数的实参：给出具体数据

主要是：1）数值　2)变量　3)数组元素

(2)地址传递方式



主要是：调用函数实参：一般是数组名或指针变量

2．数组有两种参数：

a）数组元素：一个数组元素相当于一个变量，所以，数组元素可以用于函数参数，用法与变量相同。　是传递数据方式

b）数组名:数组名作为一个变量用于函数实参和形参，传递的是整个数组元素或部分数据元素。　是传递地址方式

(1)数组元素作实参的传值调用

- 把数组元素当作变量来作为实参使用
- 传值调用最多返回一个值

例：
```
main （ ）
{  int  a[5];
   …
   a[0] =mul( a [1] , a [4] ) ;
   …
}
  int mul(int x , int y )
  { int  s ;
    …
   return ( s ) ;
```

调用函数　函数定义

```
        }
```

(2)数组名作函数参数(实参与形参都用数组名)

具有：存贮地址作实参的传址调用

例：　(J5_3.C)

```
main ()
 { void swap (float x[2]);
   float a[2] ={10.5,2。7} ;
   printf("%4。1f\t, %4。1f\n", a [0], a[1] );
   swap (a);
   printf (" %4.1f\t,%4。1f\n" ,a[0], a [1]);
 }
void swap(float x[2])  ⟶  相当于:  void swap( x )
   {  float t;                      {  float x [2];
      t=x [0];x [0] =x [1]; x [1] =t;
   }
```

说明:a）数组名在主调函数和被调函数中，分别进行定义，并且类型一致。

b）调用的实质：

数组 a 和数组 x

数组 a 独享存贮区	共享存贮区	数组 a 独享存贮区
10．5 / 2．7	2．7 / 10．5	2．7 / 10．5

c)实参数组和形参数组大小可以不一致

(C 语言编译时,对形参数组大小不作检查，只是将实参数组的首地址传给形参数组,这样两个数组共同占用同一段内存单元)

例　(J5_4.C)

```
main ()
{ void fun (float x[4] );
  float a[4]={3.5, 6.1, 0。2, 7.9};
  int i;
  fun (&a[1] ); 第 2 个的开始连续 3 个地址
  for  (i=0;i<=3; i++)
  printf (" %5。1f\t",a[i] );
}
void fun(float x [4])
   {  x[0] =x [0]+1; x[1] =x [1] +1;x[2]=x [2]+1;
   }
```

a [0]　a [1]　a [2]　a [3]

3.5	6.1	0。2	7.9

x [0]　x [1]　x [2]

共用内存单元

结果：3.5　7。1　1.2　8.9

数组 a 和数组 x

数组 a 独享存贮区　　共享存贮区　　数组 a 独享存贮区

a[0]	3．5	a[1]	7．1	a[0] 3．5
a[1]	6．1	a [2]	1．2	a[1] 7．1
a [2]	0．2	a[3]	8．9	a [2] 1．2
a [3]	7．9			a [3] 8．9

注意： 不要使用 x [3] ， 即：x [2] 是子程序最后一个可用元素。

（3）例子

书 P164 例 8。10 例 8。11 例 8.12 书 P166 例 8.13

（4）二维数组作函数参数

二维数组的传值调用与一维数组相似。

例：
```
main ( )
 {  int  a[3][5] ;
    …
    mul ( n , a[1] [2] , a [0] [4] ) ;
    …
 }
```
函数定义
```
    int mul (int x , int y ,  int z )
    { int  s ;
     …
```

例 (j5_8。c)

```
main ()
{  int tran (int n,int x[]);
   int total,a [4] [4]={{3, 8,9,10}, {2,5,-3,5}, {7,0, —1,4} ,{2, 4, 6, 0} };
   tran(2, a);          第一行的第 0 列 地址  9 个
   tran(2,&a [0] );      第一行的第 0 列 地址  9 个
   tran (0, &a [2]) ;     第二行的第 0 列 地址  4 个
}
tran (int n, int arr[])
 { int i;
   for  (i=0;i<4; i++)
       printf("%d,” ,arr[n*4+i]);
       printf("\n");
 }
```
计算数组元素在数组中的序号

程序输出：7, 0, −1, 4,
7, 0， −1， 4,
7, 0, −1, 4,

七、局部变量和全局变量

1．变量存储作用域

（1）作用域：决定一个变量的有效范围.

a）作用域是整个程序的变量——为全局变量

b）作用域只限于程序的一部分的变量——为局部变量

局部变量：——在一个函数内部定义的变量(内部变量)，它只在本函数范围内有效。

全局变量：——在函数之外定义的变量称为外部变量，外部变量是全局变量

(2) 有效范围为：从定义变量的位置开始到本源文件结束。

书 P169

外部变量在文件开头定义，则在整个文件范围内都可以使用该外部变量。书 P170

(3) 作用：增加函数间数据联系的渠道.

同一个文件中的所有函数都能引用全局变量的值

书 P170 例 8。15

八、变量的存储类型

——用来规定变量的作用域以及它们存在的长短

2. 变量的存储类别

- 变量存在的长短：决定一个变量存在的范围（整个程序，部分函数）
- 从变量存在的时间（即生存期）角度来分，有静态存储变量和动态存储变量

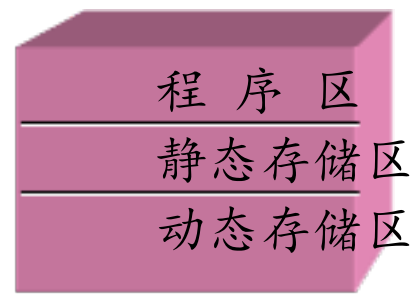
静态存储方式：

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：

指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

供用户使用的存储空间：

三个部分 → 

静态存储区：主要放全局变量

即：在程序开始执行时给全局变量分配存储区,程序执行完毕就释放。

动态存储区：主要放

①函数形参变量。在调用函数时给形参变量分配存储空间.

②局部变量（未加 static 说明的局部变量）

③函数调用时的现场保护和返回地址

在函数套眼调用开始时分配动态存储空间，函数结束时释放这些空间。

（分配和释放是动态的）

C 语言中，变量和函数有两个属性：数据类型 （如整型、字符型等）

存储类型 (数据在内存中存储的方法)

C 语言提供:(四种存贮类型符

① auto（自动） ② register(寄存器)

③ staic（静态） ④ extern （外部）

C 语言规定：存贮类型符放在变量定义的最前面

即：staic int a , b ;

(1)自动变量 (auto)

定义在函数内部的变量,变量的建立与撤消是系统自动进行的.

在调用此函数时才给此变量分配存储单元，当函数执行完毕时,存储单元就被撤消。

定义方式有：在函数体开始处定义 、在复合语句中定义