华图网校
V.HUATU.COM
SINCE 2001
乐享品质　高效学习

C 语言

刘佩贤

华图网校

目录

C 语言程序设计

主要内容

1.概述

2.数据类型及其运算

3.选择结构及其应用

4.循环结构及其应用

5.数组

6.函数

7.预处理命令

8.指针

9.结构体与共用体

1.概述

（1）C 语言的特点

① C 语言简洁、紧凑，使用方便、灵活。

Ø 一共只有 32 个关键字

Ø 9 种控制语句，程序书写自由，主要用小写字母表示，压缩了一切不必要的成分。

② 运算符丰富。

Ø 共有 34 种。C 把括号、赋值、逗号等都作为运算符处理。从而使 C 的运算类型极为丰富，可以实现其他高级语言难以实现的运算。

③ 数据结构类型丰富。

④ 具有结构化的控制语句

⑤ 语法限制不太严格，程序设计自由度大。

⑥ C 语言允许直接访问物理地址，能进行位（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。因此有人把它称为中级语言。

⑦ 生成目标代码质量高，程序执行效率高。

⑧ 与汇编语言相比，用 C 语言写的程序可移植性好。

（2）C 程序的结构

C 程序结构由头文件、主函数、系统的库函数和自定义函数组成，因程序功能要求不同，C 程序的组成也有所不同。其中 main 主函数是每个 C 语言程序都必须包含的部分。

- C 程序的结构特点

① C 程序是由函数构成的，函数是 C 程序的基本单位。至少包含 main 主函数。

② 一个函数由两部分组成：函数头和函数体。

③ 一个 C 程序总是从 main 函数开始执行的。

④ C 程序的每个语句和数据定义的最后必须有一个分号。

⑤ 一个好的、有使用价值的源程序都应当加上必要的注释，以增加程序的可读性。

- 以//开始的单行注释。
- 以/*开始，以*/结束的块式注释。

```
#include <stdio.h> //编译预处理指令

int main()          /*主函数的函数头*/

{

    printf(“这是我编写的第一个 C 语言程序，yeah!! \n”);        /*利用库函数的输出函数输出指定信息*/

    return 0;    //返回值是 0

}
```

（3）C 语言的算法

① 确定性。算法的每一种运算必须有确定的意义，该种运算应执行何种动作应无二义性，目的明确；

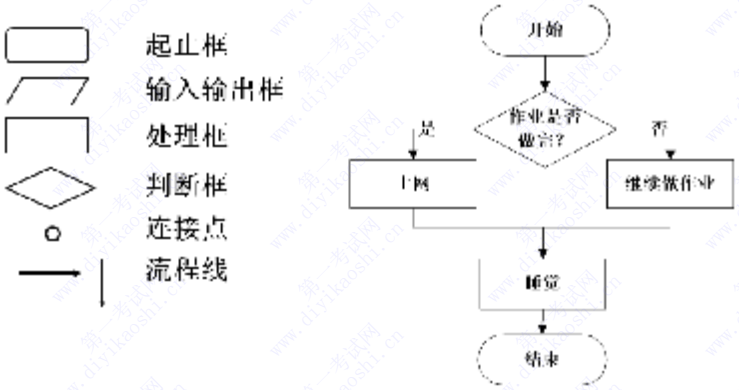
② 有穷性。一个算法总是在执行了有穷步的运算后终止，即该算法是可达的；

③ 输入。一个算法有 0 个或多个输入，在算法运算开始之前给出算法所需数据的初值，这些输入取自特定的对象集合；

④ 输出。作为算法运算的结果，一个算法产生一个或多个输出，输出是同输入有某种特定关系的量；

⑤ 有效性。要求算法中有待实现的运算都是有效的，每种运算至少在原理上能由人用纸和笔在有限的时间内完成；如若 x=0，则 y/x 是不能有效执行的。

（4）算法的表示方法



① 用自然语言表示

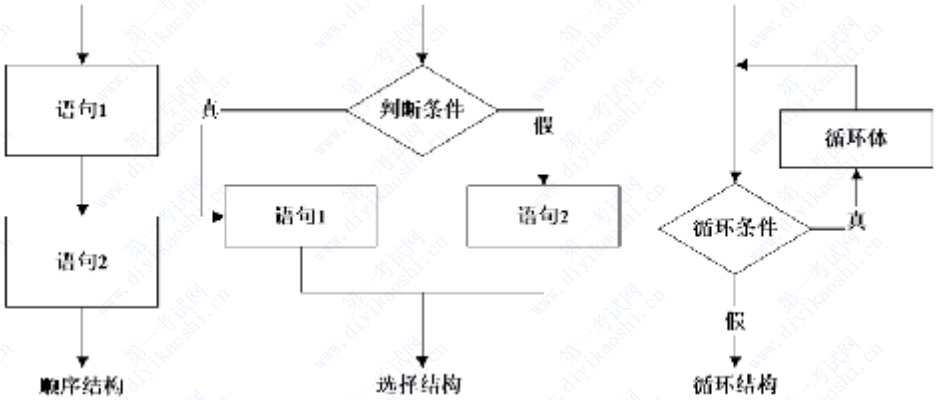
② 用流程图表示

③ 用 N-S 流程图表示

④ 用伪代码表示

（5）C 程序的三种基本结构

顺序结构、选择结构（也称分支结构）和循环结构



（6）结构化程序设计方法

①自顶向下

②逐步细化

③模块化设计

④结构化编码

习题解析

1.程序的三种基本控制结构是（ ）

A.过程，子程序和程序 B.顺序，选择和循环

C.递归，迭代和回溯 D.调用，返回和转移

2.算法具有五个特性，以下选项中不属于算法特性的是（ ）。

A.简洁性 B.有穷性

C.确定性 D.可行性

3.用 C 语言编写的代码程序（ ）。

A. 可立即执行 B. 是一个源程序

C. 经过编译即可执行 D. 经过编译、连接后才能执行

4. 一个 C 程序的执行是从（ ）。

A. 本程序的 main 函数开始，到 main 函数结束

B. 本程序的第一个函数开始，到本程序文件的最后一个函数结束

C. 本程序的 main 函数开始，到本程序文件的最后一个函数结束

D. 本程序的第一个函数开始，到本程序 main 函数结束

5. 一个 C 语言程序是由（ ）

A. 一个主程序和若干个子程序组成 B. 函数组成

C. 若干过程组成 D. 若干子程序组成

2.数据类型及其运算

（1）基本字符和标识符

- 基本字符：

① 英文字母： a~z，A~Z;

② 阿拉伯数字：0~9;

③ 空白符：空格符、制表符、换行符等。

④ 标点和特殊字符。

⑤ 字符常量：字符串常量和注释中还可以使用汉字或其它可表示的图形符号。

- 标识符：用来标明变量名、函数名、数组名、结构体名、共用体名、文件名、类型名等。必须由有效字符组成。

- 标识符的构成规则：

① 标识符只能由字母、下划线、数字三种字符组成，且第一个字符必须是字母或下划线。例：sun，_month1

② 字母区分大小写，如 abc 和 ABC 是不同的标识符。

int	long	short	float	double	char
const	signed	unsigned	if	else	for
while	do	switch	case	continue	break
default	auto	register	static	extern	void

③

return	struct	union	enum	typedef	volatile
goto	sizeof	include			

中
的关键字

- 关键字：C 语言中系统规定具体特别意义的字符串称为关键字。

① 类型说明符。如表明整型的 int、表明字符型的 char 等等。

② 语句定义符。用于表示语句的功能的，如条件选择结构中的 if、else，循环结构中的 for、while、do 等等。

③ 预处理命令字。表示一个预处理命令，如文件包含预处理命令 include 等。

（2）常量与变量

- 常量：程序运行中其值不发生变化的量。

① 字面常量：通常的数字与字符，如 3，‘a’。

② 符号常量：即自定义常量，用一个标识符代表某一个字面常量，#define PI 3.1415926

- 变量：程序运行中其值可以改变的量。属性：

① 变量名，即标识符；

② 存储空间，在内存中占用一定的存储单元，就如同宿舍号码对应一个房间，变量名对应着一段存储空间；存储空间大小由变量类型决定。

③ 变量值，即该存储空间中存放的变量的值。

对于变量，C 语言规定一定要“先定义，后使用”，也就是说必须要先规定了变量的名字、数据类型之后，才能进行各种运算。

变量定义的格式为：

[存储类型]　数据类型　变量 1，… 变量 n ；

（3）数据类型

① 整型数据

整型常量，即整常数。按进制分，有三种表示形式：

- Ø 十进制，例如：10，0，-1289。
- Ø 八进制，以数字 0 开头，例如：010 对应十进制的 8
- Ø 十六进制，以 0x 或者 0X 开头，例如：0x10 对应十进制 16

整型变量

按照所占存储空间的不同，它可以分为三种：

- Ø 基本型：关键字为 int，占 2 个字节（16 位），最高位为正负符号位，取值范围是-2^{15}～$2^{15}-1$，即-32768——32767。
- Ø 短整型：关键字为 short int(int 也可以省略不写)，占 2 个字节（16 位）
- Ø 长整型：关键字为 long int(int 也可以省略不写)，占 4 个字节（32 位），最高位为正负符号位，取值范围是-2^{31}～$2^{31}-1$ ，即-2147483648——2147483647。

• 无符号的整型变量：

- Ø 无符号基本型：以 unsigned int 表示，占 2 个字节（16 位）
- Ø 无符号短整型：以 unsigned short int 表示(int 可省略)，占 2 个字节（16 位）
- Ø 无符号长整型：以 unsigned long int 表示(int 可省略)，占 4 个字节（32 位）

② 实型数据

• 实型常量，即实数或浮点数。有两种表示形式：

- Ø 十进制小数形式。例如：10.5，0.0，-1.86。
- Ø 指数形式。例如：1.3e4 或者 1.3E4，表示 1.3×10^4 。

• 实型变量

- Ø 单精度型：以 float 表示，占 4 个字节（32 位）

Ø 双精度型：以 double 表示，占 8 个字节（64 位）。

Ø 长双精度型：以 long double 表示，占 16 个字节（128 位）

③ 字符型数据

Ø 字符常量：必须用单引号括起来，单引号中只能为单个字符。例：‘A’ (65)

Ø 字符串常量：即一串普通字符用双引号括起来。例：“ Turbo C ”。长度为 n 的字符串，在计算机的存储中占用 n+1 个字节，最后一个字节是’\0’，作为字符串结束标志。

Ø 转义字符：以“ \ ”开头的字符根据右斜杠后面的不同字符表达相应的特定含义。转义字符通常表示一些控制代码和功能定义。

字符	功能
\n	回车换行
\r	回车
\t	水平制表
\b	退格
\v	垂直制表
\f	换页
\a	响铃警报
\"	双引号字符

- 字符型变量：字符型变量在内存中占一个字节。其定义形式和整型、实型变量一样，只是数据类型标识符为 char。
- 将一个字符赋给一个字符变量中，实际上不是把这个字符本身放到内存中，而是将其对应的 ASCII 码值放到内存中。

（4）数据类型转换

隐式类型转换由编译系统自动进行，不需人为干预。

Ø 如参与运算的变量类型不同，则先转换成同一类型，然后进行运算。

Ø “低级向高级转换”原则。例：若 int a=2;double b=1.7;那么计算 a+b 为 double 型

Ø 赋值运算两边的数据类型不同时，赋值号右边量的类型将转换为左边量的类型。

例：a+b 计算结果为 double 型，d 为整型。

```
int d;
d=a+b;
```

- 显式类型转换 即强制类型转换，是直接将某数据强制转换成指定的数据类型。

其格式为：(数据类型标识符)(表达式)；

如：(double)a、(int)(x+y)

注意：强制类型转换只作用于表达式的结果，并不改变各个变量本身的数据类型。

（5）运算符和表达式

(1) 算术运算符	+、-、*、/、%
(2) 关系运算符	<、>、<=、>=、==、！=
(3) 逻辑运算符	&&、||、！
(4) 赋值运算符	=
复合算术赋值	+=、-=、*=、/=、%=
复合位运算赋值	&=、|=、^=、>>=、<<=
(5) 自增自减运算符	++、--
(6) 逗号运算符	,
(7) 条件运算符	？：
(8) 指针运算符	*
(9) 地址运算符	&
(10) 构造类型特殊运算符	.(引用成员运算符) →(指向成员运算符) [](下标运算符)
(12) 圆括号运算符	()
(13) 大括号运算符	{}
(14) 长度运算符	sizeof(类型标识符)
(15) 数据类型转换运算符	(类型标识符)〈表达式〉
(16)位运算运算符	&、^、|、~、<<、>>

- C 语言赋值运算符与数学中的等号含义有着本质的不同。
- 格式 ： 变量 = 表达式 ；
- 先计算运算符右边的表达式，然后将其值赋给等号左边的变量。
- 复合算术赋值运算符

名 称	运算符	例:	
加赋值	+=	a+=b	a=a+b
减赋值	-=	a-=b	a=a-b
乘赋值	*=	a*=b	a=a*b
除赋值	/=	a/=b	a=a/b
模赋值	%=	a%=b	a=a%b

- 复合位运算赋值运算符

按位与赋值运算符&=，如a&=b，等价于a=a&b
按位或赋值运算符|=，如a|=b，等价于a=a|b
按位异或赋值运算符^=，如a^=b，等价于a=a^b，
位右移赋值运算符>>=，如a>>=b，等价于a=a>>b
位左移赋值运算符<<=b，如a<<=b，等价于a=a<<b

- 简单赋值运算与自反赋值运算的结合性是“自右至左”。

例：已知 x=10，则 x += x- =60 的值为？-100

- 自增自减运算符：自增自减运算仅限用于变量。自增自减运算不能用于常量或表达式。右结合性

名称	运算符	说 明	例
增1（前缀）	++	先加1，后使用。	++i
增1（后缀）	++	先使用，后加1。	i++
减1（前缀）	--	先减1，后使用。	--i
减1（后缀）	--	先使用，后减1。	i--

例：int j，i=3;
j=++i;
结果：i=4;
j=4。

例：int j，i=3;
j=i++;
结果：i=4;
j=3。

- 逗号运算符和逗号表达式

逗号表达式由逗号运算符连接常量、变量或表达式构成。表达式 1 ，表达式 2 ，… 表达式 n;

整个表达式的值是最后一个表达式的值。

注意：在逗号表达式中，前后表达式使用同一个变量时，前一变量的计算结果会影响后面表达式的计算结果，不能简单将表达式的值等于最后一个表达式的值。

例如：若 a=4；则 a = 3*5，a*4；的结果是多少？60

（6）数据的输入输出

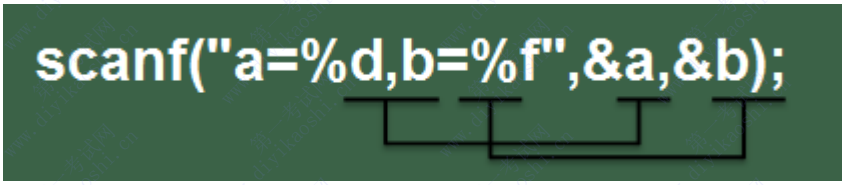
C 语言提供了一批标准输入输出函数，这些函数包含在一些头文件中，称为库函数，要使用这些函数必须在程序开始先用文件包含命令#include 包含这些头文件。

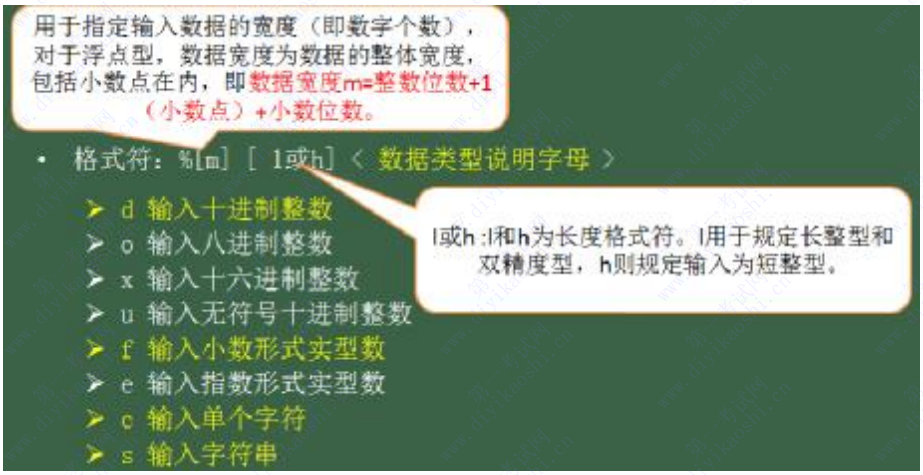
#include<stdio.h>或#include “stdio.h”

① 格式化输入函数

scanf (“格式控制字符串”，输入项地址列表)；

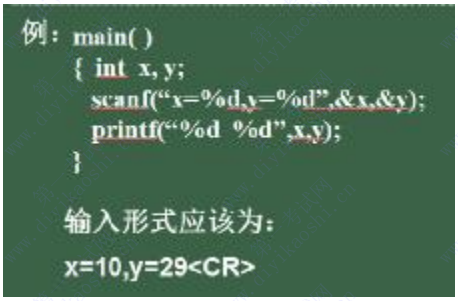
- 格式控制字符串包含两部分：格式符与普通字符。
  - Ø 格式符用于规定输入的格式，如%d、%f 等，规定了输入数据的类型、长度等；
  - Ø 普通字符是需按原样输入的字符。
- 输入项地址列表，由需要输入数据的变量的地址组成。变量的地址需用取地址运算符&得到。多个输入项之间用逗号分开。

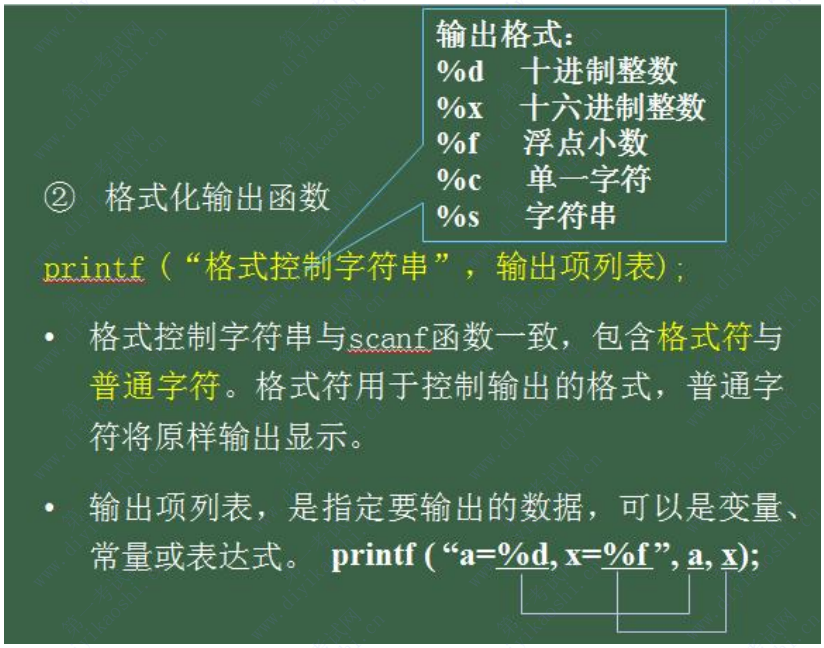


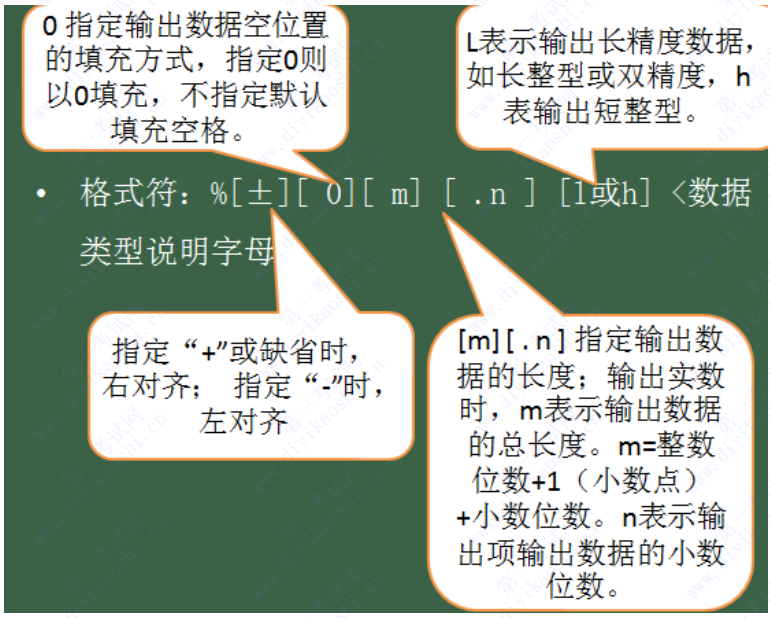


- Ø 用 scanf 函数输入多个数据时，可以用空格键、回车键或 TAB 键作分隔符进行分隔：

- 如果 scanf 的双引号里有“,”、“:”、“;”、“ ”（空格）、“a=”等等的普通字符，输入的时候一定要原样输入，否则可能会发生严重的错误。







数据类型说明字母

- %d 带符号十进制整数输出
- %f 以小数形式输出，含 6 位小数。单精度输出数据有效位 7 位；双精度输出数据有效位 16 位。
- %e 以指数形式输出， 其中尾数部分 6 位小数,指数 3 位,正负号 1 位,e1 位
- %c 以单个字符输出
- %s 以字符串输出
- 字符输入函数 getchar

函数原型：int getchar ( )

功能：从标准输入文件即键盘上读取一个字符，函数返回的是字符的ASCII码（值）。

函数引用格式：getchar（）

```
#include<stdio.h>
main()
{ char c;
  c=getchar( );
}
```

④字符输出函数　putchar

函数原型:int　putchar(char　ch)

```
#include “stdio.h”
main( )
{ int c=97;
  putchar ( c );
}
```

参数 ch 是一个字符常量、字符变量或整型常量、整型变量，其值是字符的 ASCII 码。

功能：向标准输出文件即显示终端输出一个字符。

（7）赋值语句

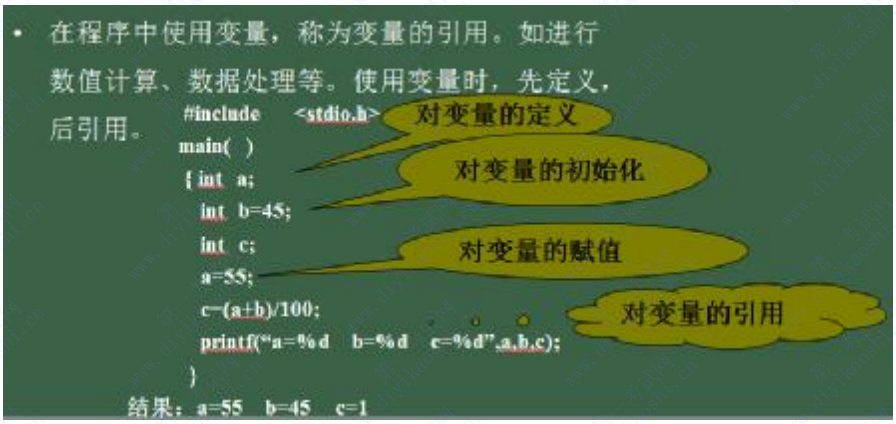
赋值语句即实现赋值功能的语句，其形式：

变量=表达式;

表达式可以为常量、变量或运算式。对变量的赋值可在两个部分实现，一个是变量初始化，一个是赋值语句。

```
int a=5;      int a;
              a=5;
```

- 在程序中使用变量，称为变量的引用。如进行数值计算、数据处理等。使用变量时，先定义，后引用。



（8）顺序结构程序设计

C 语言为结构化程序设计，分为三种基本结构：顺序结构、选择结构、循环结构。

- 顺序结构是最基本的结构，程序从上到下依次执行。
- 实际上选择与循环结构都为局部结构，是在整体顺序结构框架中的。
- 顺序结构程序按照需实现的功能逻辑顺序进行设计。

（9）数学函数

数学函数包含在头文件 math.h 中，因此要使用数学函数，必须在文件头先使用文件包含命令#include “math.h” 或#include<math.h>。

abs()，sqrt() ，log10()，log()（自然对数）exp() ，pow10() （次方函数 10 为底），pow() ，sin() ，cos() ，tan() ，asin() ，acos() ，atan()

习题解析

1. 在 C 语言中，用户能使用的正确标识符是(　　)

A. 5f　B. _for　C. int　D. _f.5

2. 设 char x='a';则 printf("x=%d,y=%c\n",x,97);的输出是（　　）

A. x=a,y=97　B. x=97,y=a　C. x=97,y=97　D. x=a,y=a

3.以下程序运行后，a 和 b 的值分别为(　　)

```
#include <stdio.h>
 main()
 {    int a,b;
       a=10%3,b=5;
       printf( “%%%d,%%%d” , a , b);
  }
```

A. %%1,%%%5　B. %1,%5　C. %%3,%%5　D. 1,5

4.若已定义 x 和 y 为 double 类型，则表达式 x=1，y=x+3/2 的值是(　　)

A. 1　　B. 2　　C. 2.0　　D. 2.5

5.若变量 a、i 已正确定义，且 i 已正确赋值，合法的语句是(　　)

A. a==1　　B. ++(a+i);　　C. a=a++=5;　　D. a=int(i);

6.有如下程序

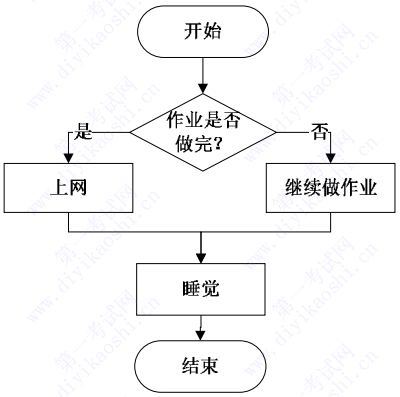
```
main( )
{　　int y=3,x=3,z=1;
　　printf( "%d %d\n" ,(++x,y++),z+2);
}
```

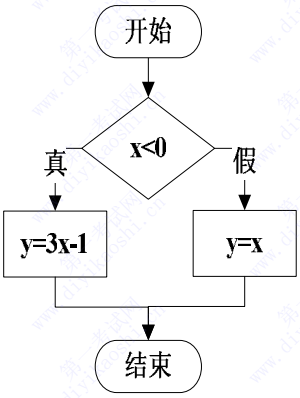
运行该程序的输出结果是(　　)

A. 3 4　　B. 4 2　　C. 4 3　　D. 3 3

3.选择结构及其应用

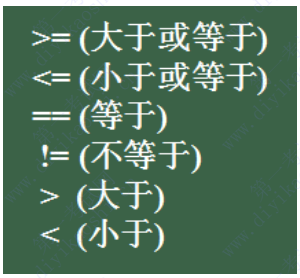
- 选择结构是程序的基本结构。所谓选择结构，就是根据不同的条件，选择不同的程序块（分程序）进行处理。





（1）关系运算符和关系表达式

① 关系运算符是对两个操作量进行大小比较的运算符，其操作结果是“真”或“假”。用“1”表示真；“0”表示假。



- （<、<=、>、>=）的优先级相同，（==、!=）的优先级相同，前 4 种的优先级高于后两种。如 a>=b!= b<=3 等价于（a>=b)!=（ b<=3)。
- 关系运算符的结合性为从左到右。

② 关系表达式：是用关系运算符把操作对象连接起来而构成的式子，操作对象可以是各种表达式。关系表达式运算结果值为 1 或 0。

如： (6>5)<2

结果为：1

（2）逻辑运算符和逻辑表达式

逻辑运算是用来判断一件事情是“成立”还是“不成立”，判断的结果只有两种，分别用数“1”和“0”来表示。判假不判真，非假即真。

Ø 如果参与逻辑判断的数值为“0”，则把它作为“逻辑假”处理。

Ø 如果参与逻辑判断的数值不为“0”，则把它作为“逻辑真”处理。

① 逻辑运算符

a	b	a&&b	a||b	!a	!b
真	真	真	真	假	假
真	假	假	真	假	真
假	真	假	真	真	假
假	假	假	假	真	真

Ø 三种运算符的优先级由高到低依次为：!、&&、||。

Ø 逻辑运算符中的“&&”和“||”的结合性为从左到右，“!”的结合性为从右到左。

Ø 关系运算符的优先级低于算术运算符，逻辑运算符中的“&&”和“||”的优先级低于关系运算符，“!”的优先级高于算术运算符。

② 逻辑表达式：是用逻辑运算符把操作对象连接起来而构成的式子。逻辑表达式运算结果值为 1 或 0。

Ø 逻辑量——凡是参加逻辑运算的变量、常量都是逻辑量，以 0 代表“假”，以非 0 代表“真”。

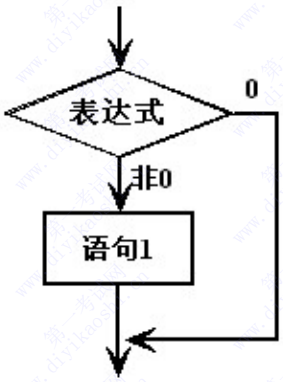
Ø 逻辑值——逻辑量、逻辑表达式的运算结果的值就是逻辑值。逻辑值只能是“0”和“1”这二个数。

- 说明：
- a&&b，只有 a 为真（非 0）时，才需要判断 b 的值，如果 a 为假，就不必判断 b 的值。即：

&&运算符，只有 a≠0，才继续进行其右面的运算。

- a||b，只要 a 为真（非 0），就不必判断 b 的值，只有 a 为假时，才判断 b 的值。即：||运算符，只有 a=0，才继续进行其右面的运算。
- 2＜a＜3 在 C 语言中的表示为（a>2）&&(a＜3)。

（3）if 语句



① if 分支：是最简单的条件语句。

if（表达式）

语句 1;

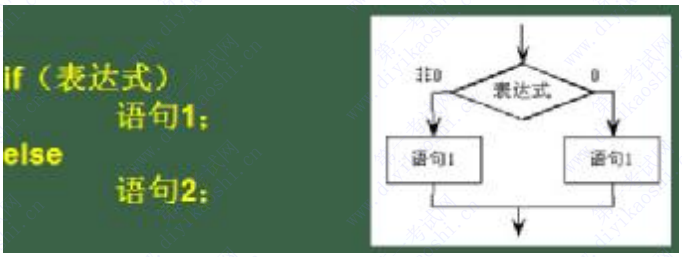
表达式一般为逻辑表达式或关系表达式。语句 1 可以是一条简单的语句或多条语句，当为多条语句时，需要用“{}”将这些语句括起来，构成复合语句。

```
y=8;
if(x＝＝5) y=2*x;
```

```
y=8;
if(x＝5) y=2*x;
```

- 使用 if 分支语句需注意以下几点：
  - Ø if 后面的表达式 e 必须用圆括号括起来；
  - Ø if 后面的表达式可以为关系表达式、逻辑表达式、算术表达式等
  - Ø 尤其需要注意的是：表达式 e 中一定要区分赋值运算符“＝”和关系运算符“＝＝”。
  - Ø if 分支中的 S 语句可以是单语句，也可以是复合语句。若为复合语句一定要将所有的 S 语句用花括号{}括起来。

②if - else 分支语句

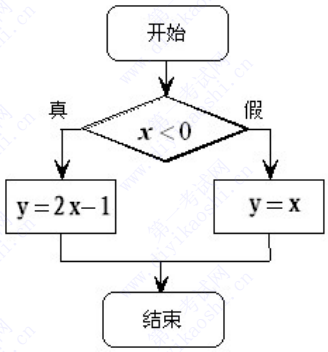


注意整个 if-else 分支是一条语句，else 分支必须是 if 语句的一部分，一定要与 if 配对使用。

例：编程表示下列函数，并输出 y 的值。

$$y=\begin{cases}2x-1 & x<0\\ x & x\geq 0\end{cases}$$

程序流程：

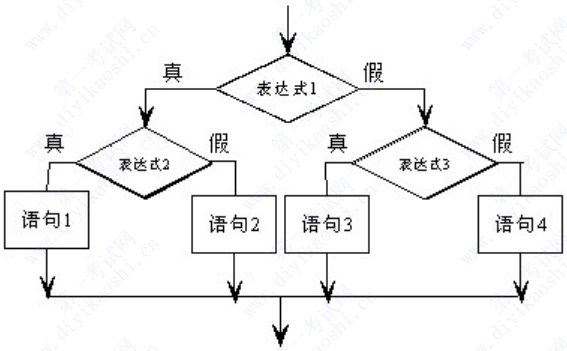



```
#include <stdio.h>
main()
{       int x,y;//定义整型变量x、y
        printf("Please input x:"); //输出屏幕提示
        scanf("%d",&x);//从键盘输入x的值
                if(x<0)
                        y=2*x-1;  //根据x的取值计算y的值
                else
                    y=x;     //根据x的取值计算y的值
        printf("y=%d\n",y);//输出y的值}
```

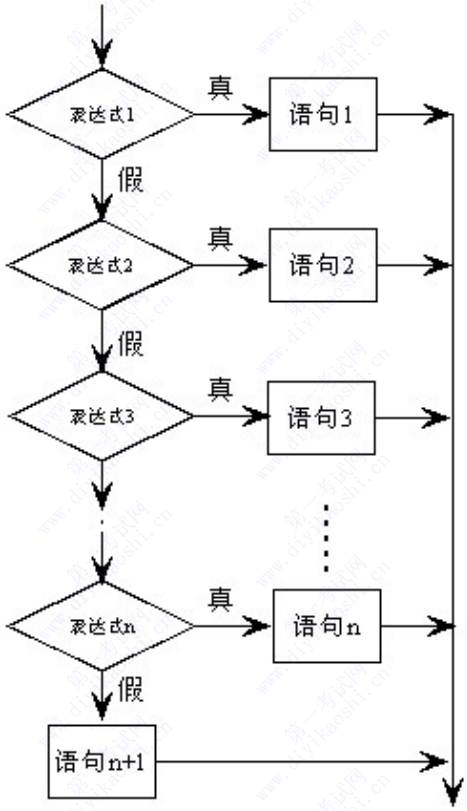
③ 嵌套的 if 语句

if 语句中又包含一个或多个 if 语句称为 if 语句的嵌套。

```
if（表达式1）
        if（表达式2）语句1;
         else 语句2;
else
         if（表达式3）
        语句3;
        else 语句4;
```




```
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：学生成绩可分为百分制和五分制，根据输入的百分制成绩 score，转换成相应的五分制输出。

百分制	五分制
90≤score≤100	A
80≤score＜90	B
70≤score＜80	C
60≤score＜70	D
0≤score＜60	E

```
#include <stdio.h>
main()
{       int score;
        printf("Please enter  score:");//屏幕提示语
        scanf("%d",&score);  //输入百分制的分数
        if(score>100||score<0)//分值不合理时显示出错信息
                printf("Input error!\n");
        else if(score>=90)   //0=<score&&score<=100
                printf("A\n");
        else if(score>=80)   //0=<score&&score<90
                printf("B\n");
        else if(score>=70)   //0=<score&&score<80
                printf("C\n");
        else if(score>=60)  //0=<score&&score<70
                printf("D\n");
        else                //0=<score&&score<60
                printf("E\n");   }
```

- 说明：

Ø if 和 else 的配对关系，else 总是与其前方最靠近的，并且没有其它 else 与其配对的 if 相配对。

Ø 每一个 else 本身都隐含了一个条件，如上题中的第 1 个 else 实质上表示条件 0=<score&&score<=100 成立，此隐含条件与对应的 if 所给出的条件完全相反。

（4）switch 语句——多分支选择语句

```
switch(表达式)
{
case 常量表达式1：[语句系列1]
case 常量表达式2：[语句系列2]
…
case 常量表达式n：[语句系列n]
[default：语句系列n+1]}
```

例：从键盘上输入 1-7 之间的数字时，显示对应的星期几的英文单词，当输入数字不在 1-7 的范围内时，输出“Error!”。

```
#include <stdio.h>
main()
{
int a;
printf("Please enter an integer :");
scanf("%d",&a);
switch(a)
{case 1:printf("Monday\n");break;
case 2:printf("Tuesday\n");break;
case 3:printf("Wednesday\n");break;
case 4:printf("Thursday\n");break;
case 5:printf("Friday\n");break;
case 6:printf("Saturday\n");break;
case 7:printf("Sunday\n");break;
default:printf("Error!\n");}
}
```

- 说明：

Ø switch 后的表达式和 case 后的常量表达式可以是整型、字符型、枚举型，但不能是实型。

Ø 同一个 switch 语句中，各个 case 后的常量表达式的值必须互不相等。

Ø case 后的语句系列可以是一条语句，也可以是多条语句，此时多条语句不必用花括号括起来

Ø default 可以省略，此时如果没有与 switch 表达式相匹配的 case 常量，则不执行任何语句，程序转到 switch 语句的下一条语句执行。

Ø break 语句和 switch 最外层的右花括号是退出 switch 选择结构的出口，如果程序没有 break 语句，则在执行完某个 case 后的语句系列后，将继续执行下一个 case 中的语句系列，直到遇到 switch 语句的右花括号为止。

Ø 各个 case 及 default 的次序是任意的，default 可以位于 case 之前。

```
int a=4;
switch(a)
{    case 1:a++;
     default:a++;
     case 2:a++;
}
printf("a=%d",a);
```

此程序段的运行结果为：a=6。由此可以看出，在上述情况下，执行完 default 后的语句系列后，程序将自动转移到下一个 case 继续执行。

Ø 如果多种情况都执行相同的程序块，则对应的多个 case 可以执行同一语句系列。

```
#include <stdio.h>
main()
{           int score;
            printf("Please enter  score:");//屏幕提示语
            scanf("%d" &score);  //输入百分制的分数
            switch(score/10)     {
            case 10:
            case 9:printf("A\n");break;//score/10等于10和9均执行此分支
            case 8:printf("B\n");break;//score/10等于8执行此分支
            case 7:printf("C\n");break;//score/10等于7执行此分支
            case 6:printf("D\n");break;//score/10等于6执行此分支
            case 5:
            case 4:
            case 3:
            case 2:
            case 1:
            case 0:printf("E\n");break;//score/10等于0~5均执行此分支
          default :printf("Input error!\n");//分数值不合理时显示出错信息
                  }
}如果多个分支执行同样的处理时，只需要在最后一个分支后写上处理语句。
```

（5）条件运算符和条件表达式

条件运算符很特殊，它是 C 语言中唯一的一个三目运算符，它要求有三个运算对象。

表达式 1？表达式 2：表达式 3

条件表达式的执行过程是：若表达式 1 为真，则条件表达式的值等于表达式 2 的值，否则等于表达式 3 的值。

- 说明：

Ø 条件运算符的优先级低于算术运算符、关系运算符及逻辑运算符，仅高于赋值运算符和逗号运算符。

Ø 条件运算符的结合性为从右到左，当有条件运算符嵌套时，按照从右到左的顺序依次运算。

例如：int a=1,b=2,c;

则条件表达式 a<b?(c=3):a>b?(c=4):(c=5)的值为 3，变量 c 的值也为 3。

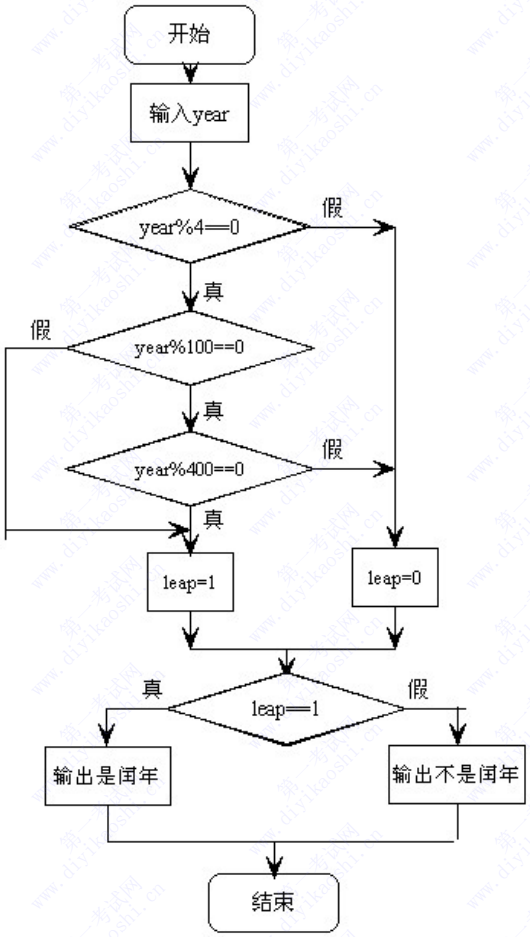
Ø 条件表达式中 3 个表达式的类型可以不同，当表达式 2 与表达式 3 类型不同时，条件表达式值的类型为二者中较高的类型。例如：int a=1,b=2;则条件表达式 a<b?3:4.0 的值为 3.0，而非整型数 3。

例：键盘输入任一年的公元年号，编写程序，判断该年是否是闰年。

闰年的条件是：

（1）能被 4 整除，但不能被 100 整除。

（2）能被 400 整除。




```
#include <stdio.h>
main()
{  int year,leap;
    printf("enter year:");
   scanf("%d",&year);
  if(year%4==0)
   {        if(year%100==0)
            {           if(year%400==0) leap=1;
                        else                leap=0;
            }
            else        leap=1;
   }
else        leap=0;
if(leap)
            printf("%d is a leap year\n",year);
else
            printf("%d is not a leap year\n",year);}
```

习题解析

1.设 int a=7,b=9,t; 执行表达式 t=(a>b)?a:b 后，T 的值是多少？（　　）

A.7　B.9　C.1　D.0

2.当 c 的值不为 0 时，在下列选项中能正确将 c 的值赋给变量 a、b 的是（　　）。

A.c=b=a;　　B.(a=c)||(b=c);

C.(a=c)&&(b=c);　　D. a=c=b;

3.有以下程序

```
main()
{    int a,b,d=25;
     a=d/10%9;b=a&&(-1)
     printf("%d, %d\n", a, b);
}
```

程序运行后的输出结果是（　　）。

A. 6,1　　B. 2,1　　C. 6,0　　D. 2,0

4.如下程序

```
main()
{    int x=1,a=O,b=O;
     switch(x)
     {    case  0: b++;
          case  1: a++;
          case  2: a++; b++;
     }
    printf("a=%d, b=%d\n", a,b);
}
```

该程序的输出结果是（　　）。

A.a=2, b=1　　B. a=1, b=1

C. a=1, b=0　　D. a=2, b=2

4.循环结构及其应用

- 循环结构又称重复结构，可以完成重复性、规律性的操作。如求若干数的和、迭代求根等等。
- C 语言共有三种类型的循环语句：
  - Ø while
  - Ø do-while

Ø for

- 循环语句三个要素

Ø 循环的初始状态；

Ø 循环执行的条件；

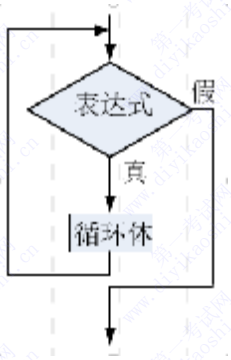
Ø 循环体。

（1）while 语句

功能：先判断表达式的值的真假，若为真（非零）时，就执行循环体的语句系列，否则退出循环结构。

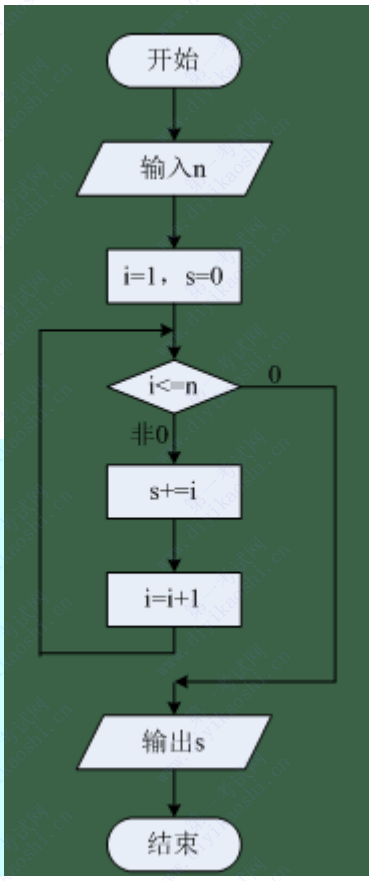
while 语句的一般格式：

```
while（表达式）
{
    循环体;
 }
```



例 4-1 计算从 1 到 n 的 n 个自然数的累加

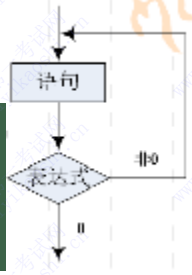
```
#include <stdio.h>
main()
{ int i,n,s;
  printf("Please input n:");
  scanf("%d",&n);
  i=1;
  s=0;
  while(i<=n){
   s+=i;
   i++; } printf("sum=%d\n",s);}
```



（2）do-while 语句

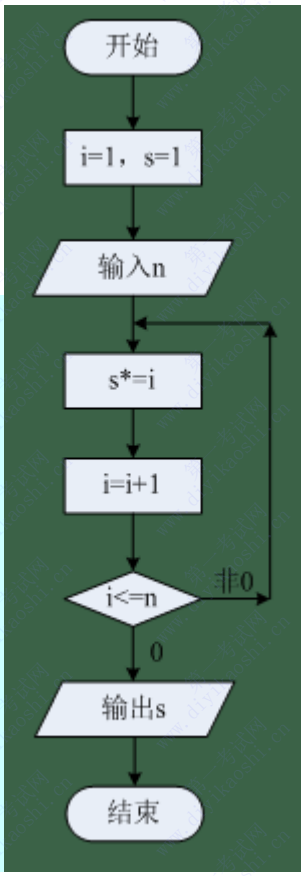
功能：首先执行循环体中的语句一次，然后计算表达式的值，若为真（非 0）时则继续执行循环体，并再计算表达式的值，当表达式的值为假（0），则终止循环，执行 do-while 语句后的下一语句。

```
do
语句; /*循环体*/
while (<表达式>);
```



例 4-2 求 n!。

```
#include "stdio.h"
main ( )
{ int i, n;
  long s;
  s=1;i=1;
  printf ("Please input n:\n");
  scanf ("%d", &n );
  do {
    s*=i;
    i++;
  } while (i<=n);
  printf ("%d!=%ld\n", n, s);      }
```

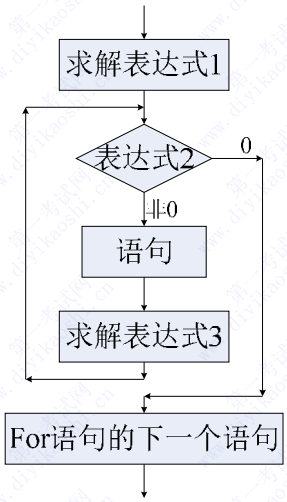


（3）for 循环语句

for 语句的一般形式：

```
for(<表达式1>;<表达式2>;<表达式3>)
   <语句>
如:   for(i=1;i<=100;i++)
            s+=i;
注意：三个表达式之间必须用分号；隔开。
```

- 表达式 1：通常为赋值表达式，用来确定循环结构中的控制循环次数的变量的初始值。
- 表达式 2：通常为关系表达式或逻辑表达式，用来判断循环是否继续进行的条件。
- 表达式 3：通常为表达式语句，用来描述循环控制变量的变化，多数情况下为自增/自减表达式。



例 4-3 用 for 语句完成从 1 到 n 的 n 个自然数的累加。

```
#include "stdio.h"
main()
{ int i,n,s;
  printf("Please input n:");
scanf("%d",&n);
  s=0;
  for(i=1;i<=n; i++)
   s+=i;
  printf("sum=%d\n",s);
}
```

- 注意：

① for 循环中语句可以为复合语句，但要用“{”和“}”将参加循环的语句括起来。

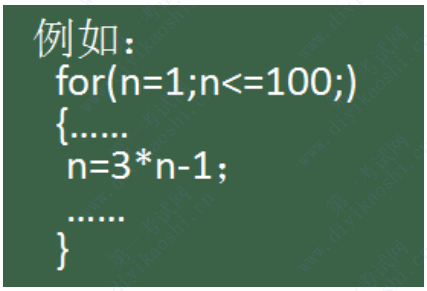
② for 循环中的“表达式 1”、“表达式 2”和“表达式 3”都是选择项，即可以缺省，但分号“;” 绝对不能缺省。

③ 省略表达式 1，表示不对循环控制变量赋初值。语句格式为：for(;表达式 2;表达式 3)

例如： n=20;for(;n<k;n++) 语句;

④ 省略表达式 2，表示不用判断循环条件是否成立，循环条件总是满足的，则不做其它处理时便成为死循环。 for(表达式 1;;表达式 3);

⑤ 省略表达式 3，则不对循环控制变量进行操作，这时可在语句体中加入修改循环控制变量的语句。for(表达式 1;表达式 2;)



⑥ 省略 3 个表达式，语句格式为：for(;;)这是一个无限循环语句，与 while(1)的功能相同，一般处理方法是：在循环体内的适当位置，利用条件表达式与 break 语句的配合中断循环，即当满足条件时，用 break 语句跳出 for 循环。

⑦ for 语句的循环体可以是空语句，表达当循环条件满足时空操作。一般用于延时处理。for(表达式 1;表达式 2;表达式 3) ；

⑧ 在 for 语句中，表达式 1 和表达式 3 都可以是一项或多项。当多于一项时，各项之间用逗号“,”分隔，形成一个逗号表达式。

for(n=1,m=100;n<m;n++,m--){……}

⑨ 循环的条件一开始就是为‘假’，即表达式 2 一开始就为 0，就不执行循环体，而是执行 for 结构之后的语句。for 循环结构可以用等价的 while 循环表示。

⑩ 表达式 3 不仅可以自增，也可以自减，还可以是加/减一个整数。

for(i=0;i<=10;i+=2)/*循环控制变量从 1 变化到 10，每次增加 2*/

（4）三种循环语句的比较

① 在一般情况下，三种循环语句均可处理同一个问题，它们可以相互替代。

② for 语句和 while 语句先判断循环控制条件，后执行循环体；而 do_while 语句是先执行循环体，后进行循环控制条件的判断。do_while 语句更适合于第一次循环肯定执行的场合。

③ 用 while 和 do-while 循环时，循环变量初始化的操作应在 while 和 do-while 语句之前完成。而 for 语句可以在表达式 1 中实现循环变量的初始化。

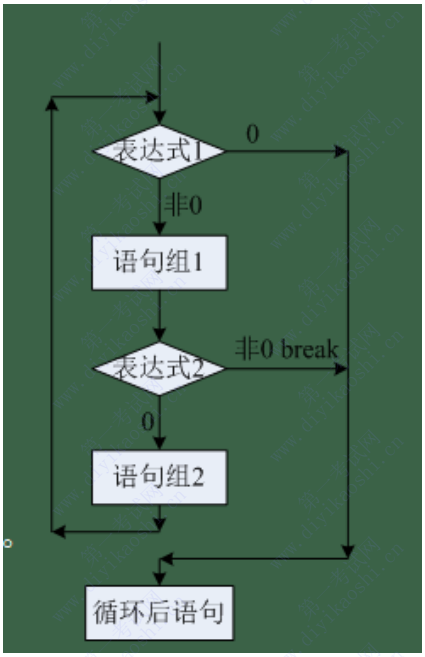
④ while 和 do-while 循环，只在 while 后面指定循环条件，在循环体中应包含使循环趋于结

束的语句（如 i++，或 i=i+1 等）。for 循环可以在表达式 3 中包含使循环趋于结束的操作

（5）break 语句

形式：break;

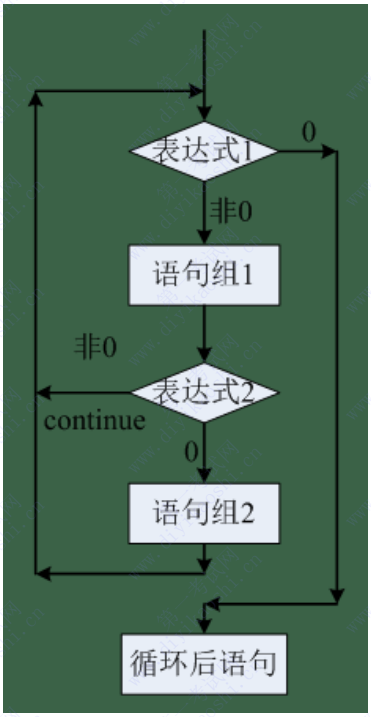
- Ø 在 switch 语句中，用 break 语句终止正在执行的 switch 流程，跳出 switch 结构，继续执行 switch 语句下面的一个语句。
- Ø while、do-while 和 for 语句的循环体中使用 break 语句，强制终止当前循环，执行循环体后面语句。



（6）continue 语句

形式： continue;

功能： 结束本次循环（不是终止整个循环），即跳过循环体中 continue 语句后面的语句，开始下一次循环。

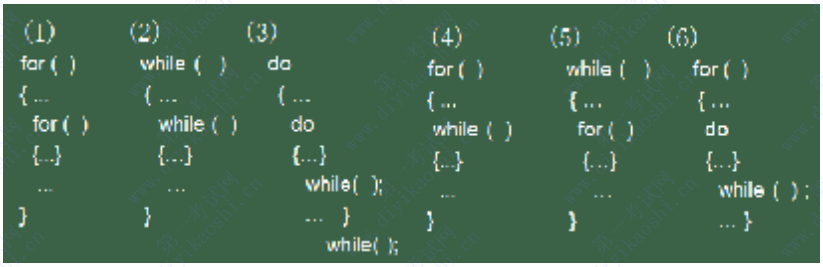


（7）循环嵌套

一个循环语句的循环体内包含另一个完整的循环结构，称为循环的嵌套。

嵌在循环体内的循环称为内循环，嵌有内循环的循环称为外循环。

- 循环嵌套的形式:前面介绍了三种类型的循环，它们自己本身可以嵌套，也可以互相嵌套。



例 4-4 求 3 到 100 之间的所有素数。

分析：为了提高效率，改进为：

- 在 3 到 100 间的素数，应均为奇数，因此，外层循环可以改为：for (k=3; k<＝100; k+=2)
- 若自然数 k 是素数，则 k 不能被 2，3，…… 整除。内层循环：for (i=2; i<=sqrt(k); i++)

```
#include "math.h"
#include "stdio.h"
main ( )
{ int tag, i, k;
   for( k=3; k<=100; k+=2 )
   { tag=1;
     for ( i=2; i<＝sqrt(k); i++ )
       if ( k%i==0 ) { tag=0; break; }
     if ( tag==1 ) printf ("%5d", k);
     }
   }
```

- 注意：
  - Ø 在使用循环嵌套时，要注意外循环和内循环在结构上不能出现交叉。
  - Ø 在同一个循环体中，允许出现多个并列的内循环结构，各个循环的嵌套重数没有限制。
  - Ø C 语言的三种循环语句可以互相嵌套，任何一种循环语句都可以用在其他循环语句的循环体中。

（8）goto 语句

功能：将程序转移到指定的位置继续执行。

格式：goto　语句标号；

语句标号：是一个标识符，它标识程序的一个特定位置。

```
goto error:
……
error: if(x==0)
          printf(“error information”);
```

习题解析

1. 语句 while(!E);中的表达式!E 等价于（　　）。

A.E==0　　B.E!=1　　C.E!=0　　D.E==1

2. 执行语句 for(i=1;i++<4;); 后,i 的值是（　　）。

A. 3　B. 4　C. 5　D.不定

3.设有程序段

```
int k=10;
```

while(k=0)k=k-1;这下面描述中正确的是（　　）。

A.while 循环执行 10 次　　B.循环是无限循环

C.循环体语句一次也不执行 D.循环体语句执行一次

4.下面程序段的运行结果是（　　）。

```
int n=0;
while(n++<=2);
printf("%d",n);
```

A. 2　B.3　C.4　D.有语法错

5.下列说法中正确的是（　　）。

A. break 用在 switch 语句中，而 continue 用在循环语句中。

B. break 用在循环语句中，而 continue 用在 switch 语句中。

C. break 能结束循环，而 continue 只能结束本次循环。

D. continue 能结束循环，而 break 只能结束本次循环。

6.以下程序段的循环次数是（　　）。

```
for (i=2; i==0; )
    printf(“%d” , i--) ;
```

A.无限次　　B.0 次　　C.1 次　　D.2 次

8.C 语言三个循环语句分别是_____语句，_____语句和_____语句。

9.至少执行一次循环体的循环语句是_____。

10.下面程序段是从键盘输入的字符中统计数字字符的个数，用换行符结束循环。请填空。

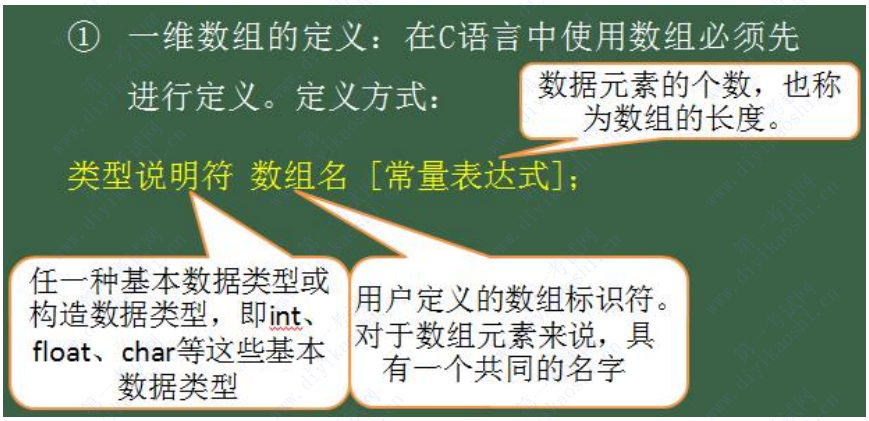
```
int n=0,c;
c=getchar();
while(              )
 {  if(              )
       n++;
    c=getchar();
 }
```

5.数组

- 数组是具有相同数据类型的变量集合，这些变量具有相同的名字，但用不同的下标表明数

据的位置，我们称这些变量为数组元素。

（1） 一维数组



- 例如：int a[5];
- 注意：

Ø 数组的类型实际上是指数组元素的取值类型。对于同一个数组，其所有元素的数据类型都是相同的。

Ø 数组名不能与其它变量名相同。

Ø 方括号中常量表达式表示数组元素的个数，如 a[5]表示数组 a 有 5 个元素。但是其下标从 0 开始计算。因此 5 个元素分别为 a[0],a[1],a[2],a[3],a[4]。

Ø 不能在方括号中用变量来表示元素的个数， 但是可以是符号常数或常量表达式。

② 一维数组元素的存储

每个数组元素都占用内存中的一个存储单元，每个元素都是一个变量，可以像以前讲过的普通变量一样使用，只不过数组元素是通过数组名和方括号“[]”里的下标来确定的。系统为数组元素在内存中分配连续的存储单元。

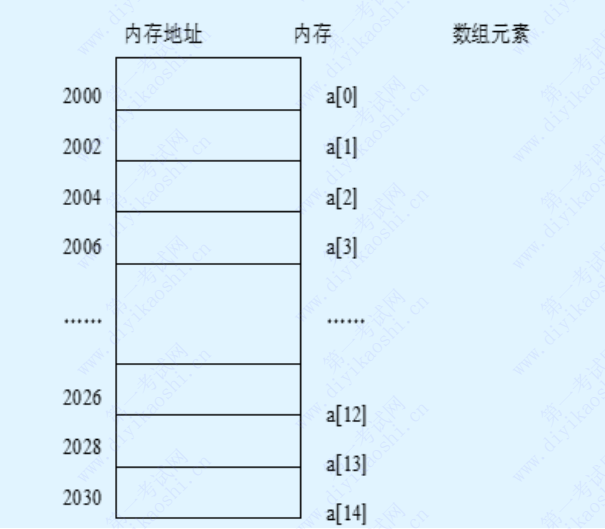
例如：定义语句 int a[15];说明了以下几个问题：

Ø 数组名为 a。

Ø 数组元素的数据类型为 int 整型数据。

Ø 数组元素的下标值从0开始。数组元素的个数为15个，它们是:a[0]、a[1]、a[2] . . . .a[13]、a[14]

Ø 数组名 a 是数组存储区的首地址，即存放数组第一个元素的地址。 a→&a[0]；因此数组名是一个地址常量。不能对数组名进行赋值和进行运算。



③ 一维数组元素的引用

数组的引用就是对数组元素（数据）的读取操作。一维数组的引用格式：

数组名[下标]

- 下标可以是常量或常量表达式，如 a[3]、a[3+2]。
- 下标也可以是变量或变量表达式，如 a[i],a[i+j]，a[i++]。
- 下标如果是表达式，首先计算表达式，计算的最终结果为下标值。
- 引用时，下标值若不是整型，C 系统会自动取整,a[5.3]相当于 a[5]。
- 下标值从 0 开始。而不是从 1 开始。
- 数组的引用下标不能越限，即引用时的下标不能超过或等于定义时的下标值。如 int a[10];a[10]=4;是错误的。

例：有等差数列 an=3n，要求输出 an 的所有值，并求出数列的和。n 为 0 到 9 的整数。

```
#include <stdio.h>
main()
{       int a[10];                      /*定义数组a*/
        int n,sum=0;                    /*定义变量n和sum*/
        for(n=0;n<=9;n++)               /*循环10次*/
        {
                a[n]=3*n;               /*给数组元素赋值*/
                printf("a[%d]=%d\t",n,a[n]);   /*输出数组元素*/
                sum+=a[n];              /*将数组元素作累加*/
        }
        printf("sum=%d\n",sum);         /*输出累加和*/
}
```

④ 一维数组的初始化

初始化赋值的一般形式为：

类型说明符 数组名[常量表达式]={数值 1，数值 2……数值 n}；

- 初始化规定：

Ø 花括号{ }中数值的个数多于数组元素的个数是语法错误。

Ø 可以只给部分元素赋初值。当{ }中值的个数少于元素个数时，只给前面部分元素赋值。例如：int a[10]={0,1,2,3,4};相当于只给 a[0],a[1]，a[2],a[3],a[4]赋初值，而后 5 个元素自动赋 0 值。

Ø 只能给元素逐个赋值，不能给数组整体赋值。例如给十个元素全部赋 1 值，只能写为：int a[10]={1,1,1,1,1,1,1,1,1,1};

Ø 如给全部元素赋值，则在数组说明中， 可以不给出数组元素的个数。例如：int a[5]={1,2,3,4,5};可写为：int a[]={1,2,3,4,5};

例：任意给 n 个数,按由小到大对其排序,并输出结果。采用“冒泡”排序法。

“冒泡”排序法的思路:将相邻两个数比较,将小的数调到前头(或将大的数调到后面)。

```
#include <stdio.h>
#define N 10                                  /*定义符号常量，对几个数排序，N的值就是几*/
main( )
{ int a[N];
  int i,j,t;
  printf("input 10 numbers:\n");
  for(j=0;j<N;j++)                                    /*从键盘接收N个数据放入数组a中*/
     scanf("%d",&a[j]);
                    printf("\n");
  for(j=1;j<N;j++)                                  /*是趟次循环变量（外循环变量）*/
     for(j=0;i<N-j;i++)          /*是每趟中两两比较的次数变量（内循环变量）*/
        if(a[i]>a[i+1])          /*比较相邻两数大小，将较小的数放在前面 */
          {  t =a[i];
             a[i]=a[i+1];
             a[i+1]=t;
          }
   printf("the sorted numbers:\n");
   for(i=0;i<N;i++)    /*将排序好的数组输出*/
      printf("%d\t",a[i]);
}
```

（2）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]；

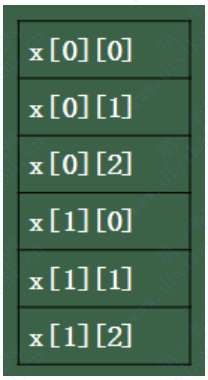
例如： int x[2][3];x 是二维数组名，这个二维数组共有6个元素 (2×3=6)，
它们是：x[0][0]，x[0][1]，x[0][2]，x[1][0]，x[1][1]，x[1][2]。 其全部元素数值均为整型的。

② 二维数组的存储

二维数组在概念上是二维的，比如说矩阵，但是存储器单元是按一维线性排列的。

在 C 语言中,二维数组是按行排列的。如: int x[2][3];先放第一行,即 x[0][0], x[0][1], x[0][2],

再放第二行,即 x[1][0]，x[1][1]，x[1][2]。



③ 二维数组的引用

数组名[下标 1][下标 2]

说明：

- Ø 下标可以是常量(大于等于 0)、常量表达式、变量或变量表达式。
- Ø 数组中要特别注意下标越限。

④ 二维数组的初始化即定义数组的同时对其元素赋值。

- Ø 把初始化值括在一对大括号内，例如二维数组 int x[2][3]={1,2,3,4,5,6};
- Ø 把多维数组分解成多个一维数组 int x[2][3]={{1,2,3},{4,5,6}};

• 说明：

- Ø 可以只对部分元素赋初值，未赋初值的元素自动取 0 值。例如：int x[2][2]={{1},{2}};是对每一行的第一列元素赋值，未赋值的元素取 0 值。赋值后各元素的值为：x[0][0]=1，x[0][1]=0，x[1][0]=2，x[1][1]=0。
- Ø 如对全部元素赋初值，则第一维的长度可以不给出。例如二维数组 x 的初始化过程：

int x[2][3]={1,2,3,4,5,6};也可写成：int x[ ][3]={1,2,3,4,5,6}；即第一下标值省略,但第二下标值不能省略。

（3）字符数组和字符串

① 字符数组的定义和初始化：

用来存放字符量的数组称为字符数组，字符型数组的定义格式：

char 数组名[字符个数]；

字符数组也允许在定义时作初始化赋值。

• 字符型数组与数值型数组在初始化中的区别：

char b[9]={ ‘G’，‘o’，‘o’，‘d’}; /*初始化时，提供的数据个数如果少于数组元素个数，则多余的数组元素初始化为空字符‘\0’ 。而前面讲过数值型数组初始化为 0*/

例：编写程序将字符 Good luck 存放在一维数组中，并输出。

```
#include <stdio.h>
main()
{
        char c[9]={'G','o','o','d',' ','l','u','c','k'};/*初始化字符数组c*/
        int i;
        for(i=0;i<9;i++) /*输出字符数组c中每个元素的值*/
                printf("%c",c[i]); /*格式化输出语句中，输出字符用%c*/
}
```

② 字符串

在C语言中没有专门的字符串变量，通常用一个字符数组来存放一个字符串。

Ø 前面介绍字符串常量时，已说明字符串总是以'\0'作为串的结束符。

Ø 当把一个字符串存入一个数组时，也把结束符'\0'存入数组，并以此作为该字符串是否结束的标志。

- 存放字符串的字符数组的初始化有两种方法。

Ø 用字符常量初始化数组，用字符常量给字符数组赋初值要用花括号将赋值的字符常量括起来。

```
char str[6]={'C','h','i','n','a','\0'};
```

Ø 用字符串常量初始化数组。

```
char string[6]="China";或
char string[6]={"China"};
```

- 字符数组初始化应注意以下几个问题：

Ø 如果提供赋值的字符个数多于数组元素的个数,则为语法错误。

例如： char str[4]={'C','h','i','n','a'}; ×
char string[4]="China"; ×

Ø 如果提供赋值的字符个数少于数组元素的个数,则多余数组元素自动赋‘\0’。

例如：char string[20]="China";字符数组string从第六个元素开始之后全部赋'\0'。

Ø 用字符串常量初始化时，字符数组的下标可以省略，其数组存放字符的个数由赋值的字符串的长度决定。

例如： char str[ ]={“1a2b3c”};等同于char str[7]={“1a2b3c”};

- Ø 初始化时，若字符个数与数组长度相同，则字符末尾不加‘\0’，此时字符数组不能作为字符串处理，只能作字符逐个处理。初始化时是否加‘\0’要看是否作字符串处理。

③ 字符数组的输入和输出

- Ø getchar()/putchar(字符名)是字符输入/输出函数，每次只能输入/输出一个字符；
- Ø gets(字符串数组名)；puts(字符串数组名)；是字符串输入/输出函数，以回车作为字符串结束标志；
- Ø 从键盘读取一个字符：scanf(“%c”,数组元素地址)；
- Ø 从键盘读取一串字符：scanf(“%s”,数组名)；
- Ø 向显示器输出一个字符：printf(“%c”,数组元素)；
- Ø 向显示器输出一串字符：printf(“%s”,数组名)；

④ 字符串处理函数

- Ø 测字符串长度函数 strlen(字符数组名)测字符串的实际长度(不含字符串结束标志‘\0’)并作为函数返回值。
- Ø 字符串比较函数 strcmp(字符数组名 1，字符数组名 2)
- 功能：将两个数组中的字符串从左至右逐个相比较，比较字符的 ASCII 码大小，并由函数返回值返回比较结果。

字符串 1＝字符串 2，返回值＝0；

字符串 1＞字符串 2，返回值＞0；

字符串 1＜字符串 2，返回值＜0。

- Ø 字符串字符大写转换成小写函数 strlwr(字符数组名)
- Ø 字符串字符小写转换成大写函数 strupr (字符数组名)
- Ø 字符串连接函数 strcat (字符数组名 1，字符数组名 2)
- Ø 字符串拷贝函数 strcpy (字符数组名 1，字符数组名 2)

功能：把字符数组 2 中的字符串拷贝到字符数组 1 中。串结束标志“\0”也一同拷贝。字符数组 2，也可以是一个字符串常量。这时相当于把一个字符串赋予一个字符数组。

例：编写程序实现输入一个字符串，求其长度。（不用 strlen 函数实现）

```
#include <stdio.h>
main( )
{ int i=0;
    char str[30];
    printf("Enter a string:");
    gets(str);                    /*从键盘输入字符串*/
    while(str[i]!='\0')/*判断是否到字符串的最后，如果是跳出循环，不是就i++*/
            i++;
    printf("string length:%d\n",i);              /*输出字符串长度i*/
}
```

习题解析

1.下列合法的数组定义是（　　）

A. char a[5]=“string”;

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

C. char s=“string”;

D. int c[]={0,1,2,3,4,5};

2. int a[4]={1,5,8,9};其中 a[3]的值为(　　)。

A.　5　　B.　3　　C. 8　　D. 9

3.以下 4 个数组定义中，(　　)是错误的。

A.　int a[7]; B. #define N 5　long b[N];

C. char c[5]; D. int n,d[n];

4. 以下关于数组的描述正确的是（　　）。

A. 数组的大小是固定的，但可以有不同的类型的数组元素；

B. 数组的大小是可变的，但所有数组元素的类型必须相同；

C. 数组的大小是固定的，但所有数组元素的类型必须相同；

D. 数组的大小是可变的，但可以有不同的类型的数组元素；

5. 在数组中，数组名表示(　　)。

A. 数组第 1 个元素的首地址

B. 数组第 2 个元素的首地址

C. 数组所有元素的首地址

D. 数组最后 1 个元素的首地址

6.执行下面的程序段后，变量 k 中的值为（　　）。

int k=3, s[2];

s[0]=k;

k=s[1]*10;

A. 不定值　　B. 33　　C. 30　　D. 10

7.下列定义的字符数组中，输出 printf("%s\n", str[2]) ;的输出是(　　)。

static str[3][20] ={ "basic", "foxpro", "windows"};

A. basic　　B. foxpro　　C. windows　　D. 输出语句出错

8.下面程序的功能

是读入 20 个整数，

统计负数个数，

并计算负数之和，

请填空。

```
#include “stdio.h”
Main(){
        int i,a[20],s,count;
        s=count=0    ;
        for(i=0;i<20;i++ )
                scanf( “%d”,&a[i]);
        for(i=0;i<20;i++    )
        {
                if(a[i]>=0)
                    continue    ;
                s+=a[i];
                count++    ;
        }
        printf(”s=%d\tcount=%d\n”,s,count);
}
```

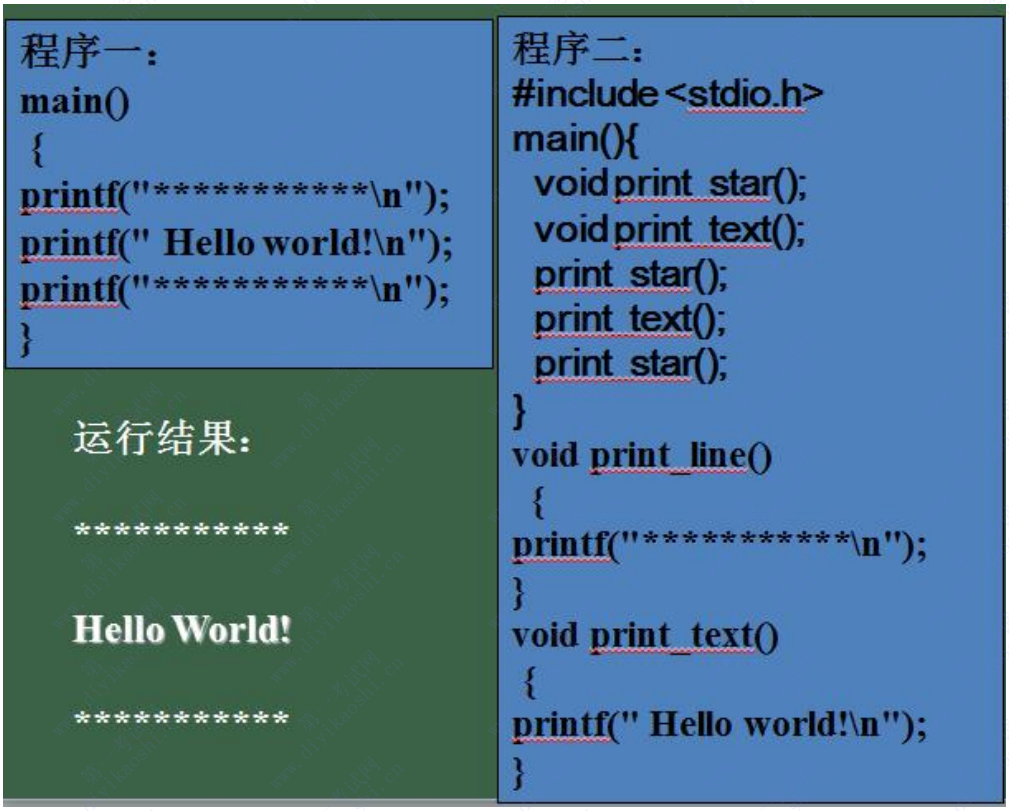
6.函数

（1）函数概述

程序员在设计一个复杂的应用时，往往也是把整个程序划分为若干个功能较为单一的程序模块，然后分别予以实现，最后再把所有的程序模块象搭积木一样装配起来。这种在程序设计中分而治之的策略，被称为模块化程序设计方法。

C 语言程序整体由一个或多个函数组成。每个函数都具有各自独立的功能和明显的界

面。



说明：

- 函数是按规定格式书写的能完成特定功能的一段程序。
- C 语言是以源文件为单位进行编译的，一个源程序由一个或多个函数组成。
- 不管 main 函数放在程序的任何位置，C 语言中程序总是从 main 函数开始执行，调用其它函数后，最终在 main 函数中结束。
- 所有函数都是平行的，在定义时相互独立，一个函数不属于另一个函数。函数不可以嵌套定义，但可以相互调用，main 函数可以调用任何函数，一个函数可以多次被调用，而其它函数不能调用 main 函数。

• 库函数

库函数是由编译系统提供的已设计好的函数，用户只需调用而无须去实现它。

- printf(),scanf()等，包含在<stdio.h>
- strlen(str)等，包含在<string.h>
- pow(),sqrt()等，包含在<math.h>

使用库函数应注意以下几个问题：

- 函数的功能；
- 函数参数的数目和顺序，以及每个参数的意义及类型；

- 函数返回值的意义及类型；
- 需要使用的包含文件。

```
#include <math.h>
#include <stdio.h>
int main(void)
{
double x = 2.0, y = 3.0;
printf("%lf raised to %lf is %lf\n", x, y, pow(x, y));
}
```

运行结果：2.000000 raised to 3.000000 is 8.000000

（2）用户自定义函数

用户自定义函数就是由程序员自己定义和设计的函数,需要程序员来编写函数功能的实现代码。

定义函数时是互相独立的，一个函数并不从属于另一个函数，即函数不能嵌套定义，但可以互相调用，但不能调用 main 函数。

① 函数定义的格式

- 无参函数的定义格式

[类型说明符] 函数名 （）

{

函数体

}

例如：

```
void print_line()
{
 printf(" ########### \n");
 }
```

Ø 有参函数的定义格式

[类型说明符] 函数名（形式参数声明）

{

函数体

}

```
int  max ( int x, int y )
{          int z ;
           z=  x>y? x: y;
           return(z);
}
```

Ø 空函数

C 语言中可以有“空函数”，它的形式为：

[类型说明符] 函数名()

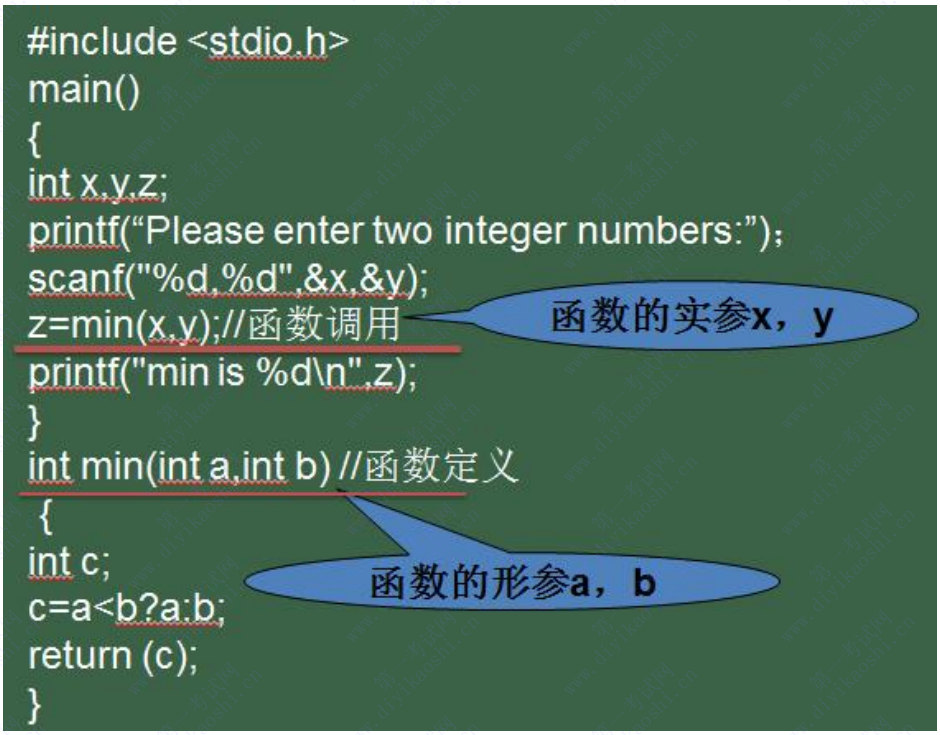
{ }

例如：echoline() {}

② 形式参数和实际参数

Ø 在定义函数时,圆括号内的变量名称为“形式参数”（简称“形参”），把它作被调函数使用时，用于接收主调函数传来的数据。

Ø 在调用函数时，主调函数的函数调用语句中的参数称为“实际参数”（简称“实参”）。实际参数可以是常量、变量或表达式。



关于形参和实参说明如下：

- Ø 函数中指定的形参变量，在未出现函数调用时，并不占用内存中的单元。在发生函数调用时，被调函数的形参被临时分配内存单元，调用结束后，形参所占的内存单元被自动释放。
- Ø 函数一旦被定义，就可多次调用，但必须保证形参和实参数据类型一致。如不同，按照强制数据类型转换进行转换。
- Ø 实参可以是常量、变量或表达式，但要求它们有确定的值。在调用时将实参的值赋给形参。
- Ø 在被定义的函数中，必须指定形参的类型。
- Ø C 语言规定，实参对形参变量的数据传递是“值传递”，即单向传递，只由实参传给形参，而不能由形参传回给实参。

③ 函数的返回值

函数执行的最后一个操作是返回。返回的意义是：

- Ø 使流程返回主调函数，宣告函数的一次执行结束，在调用期间所分配的变量单元被撤消。
- Ø 送函数值到调用表达式中。有些函数有返回值，有些函数没有返回值。

return 语句的使用格式是：

return;

或者 return (表达式);

或者　　return　表达式；

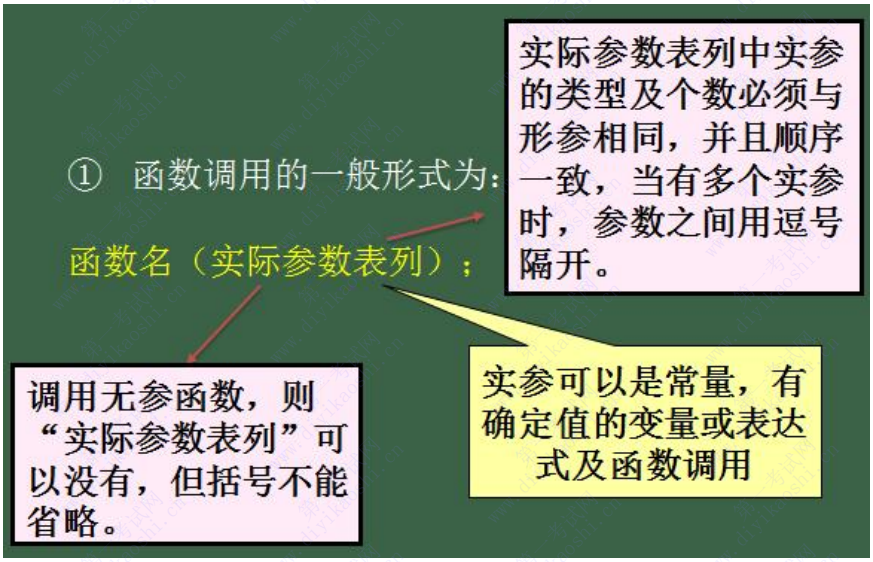
说明：

- return 语句有双重作用：它使函数从被调函数中退出，返回到调用它的代码处，并向调用函数返回一个确定的值。
- 一个函数中可以有多个 return 语句，执行到哪一个 return 语句，哪一个语句就起作用。
- 在定义函数时应当指定函数值的类型，并且函数的类型一般应与 return 语句中表达式的类型相一致，当二者不一致时，应以函数的类型为准，即函数的类型决定返回值的类型。对于数值型数据，可以自动进行类型转换。
- return 只能返回一个值，而不能返回多个值。
- 如果函数中没有 return 语句，并不代表函数没有返回值，只能说明函数的返回值是一个不确定的数。建议函数无返回值时用 void 标记函数的类型。

（3）函数的调用

所谓函数的调用，是指一个函数（调用函数）暂时中断本函数的运行，转去执行另一个函数（被调函数）的过程。被调函数执行完后，返回到调用函数中断处继续调用函数的运行，这是一个返回过程。

① 函数调用的一般形式为：



调用的函数有两种：

- 调用标准函数(库函数)，需在程序前使用包含命令:例：#include <stdio.h>
- 调用用户自定义函数，一般应在调用前“声明”被调用函数的“类型”。

② 函数的调用方式

按被调用函数在主调函数中出现的位置和完成的功能划分，函数调用有下列三种方式：

- 把函数调用作为一个语句。
- 在表达式中调用函数，这种表达式称为函数表达式。这时要求函数带回一个确定的值以参加表达式的运算。c=average(a,b);
- 将函数调用作为另一个函数调用的实参。

例如：

```
printf("The avergae of %5.2f and %5.2f is %5.2f\n",a,b,average(a,b));
```

③ 函数的原型声明

函数声明的一般形式为：

类型说明符　函数名（形参表）;

例如：float average(float x,float y);

```
float average(float,float) ;
```

- 在调用函数前，可以在主调函数内部对被调函数进行说明。如果使用这种说明方式，则本次函数说明只在主调函数内部有效。
- 如果已经在文件的开头（或在函数前）对某函数进行了说明，则从说明处开始，在本文件中对说明函数的调用不需要再次进行说明。

省略函数声明的三种情况：

- 被调函数是整型或字符型函数;
- 被调函数的定义出现在这个函数调用之前;
- 在文件开头已对被调函数的类型作了声明。

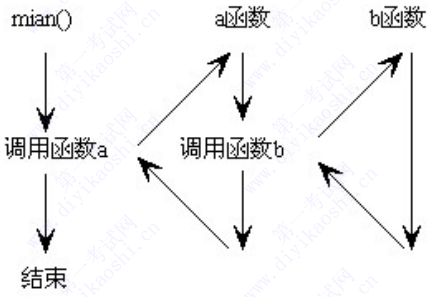
④ 函数的参数传递

在 C 语言中进行函数调用时，有两种不同的参数传递方式。

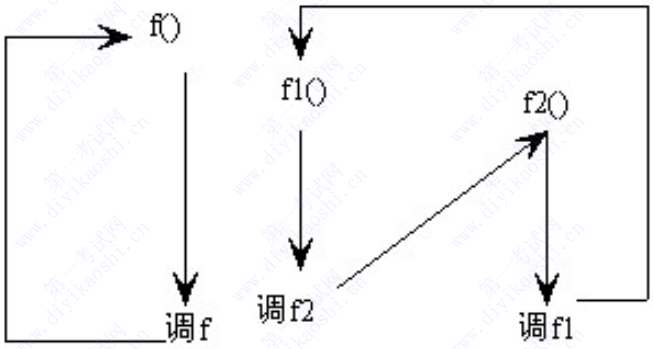
- 值传递：在函数调用时，实参将其值传递给形参，这种传递方式即为值传递。特点：一次（函数调用时），单向（实参传递给形参）。
- 地址传递指的是调用函数时，实参将某些量（如变量、字符串、数组等）的地址传递给形参。这样实参和形参指向同一个内存空间，在执行被调用函数的过程中，对形参所指向空间中内容的变化，能够直接影响到调用函数中对应的量。

（4）函数的嵌套调用和递归调用

C 语言中函数的定义是相互平行的，在定义函数时，一个函数不能包含另一个函数，但是，一个函数在被调用的过程中可以调用其它函数，这就是函数的嵌套调用。



在调用一个函数的过程中又直接或间接地调用该函数本身，称为函数的递归调用。

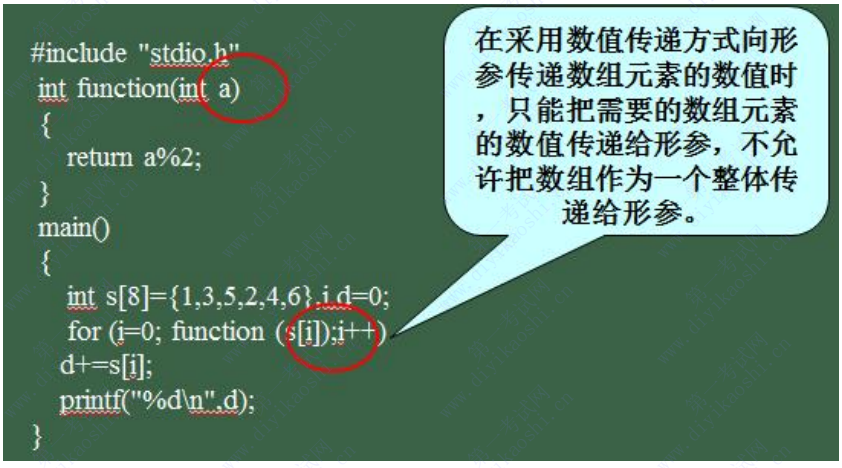


直接递归　　间接递归

（5）数组作为函数的参数

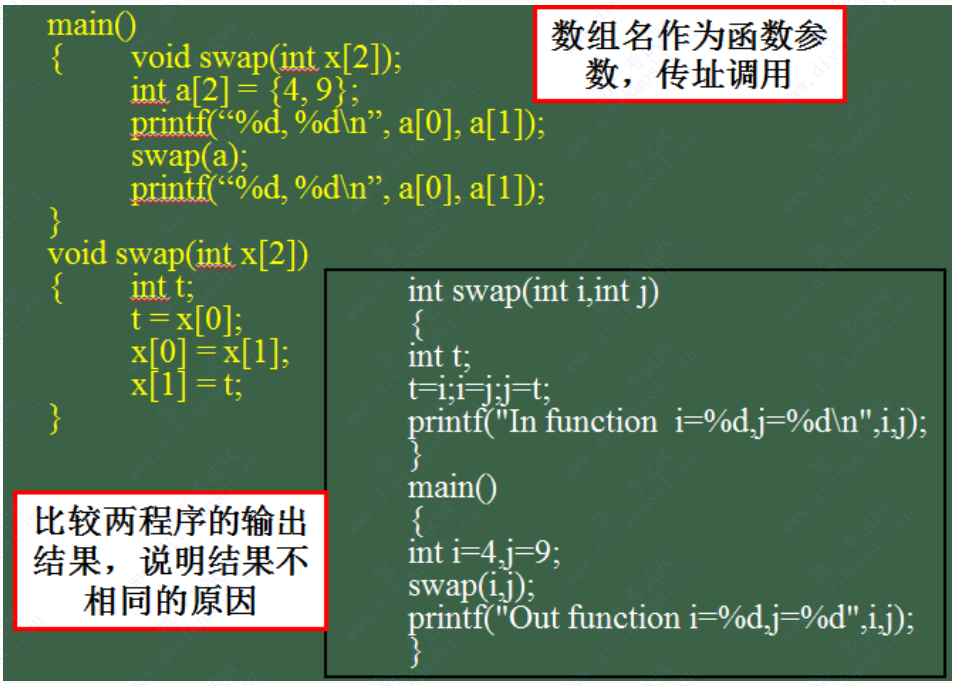
数组作为函数的参数有两种形式：数组元素作为函数的参数和数组名作为函数的参数 。

- 数组元素可作为函数的实参，其用法与变量相同，是单向传递，即“值传递”方式；但不能作形参！



- 数组名既可以作为实参也可做形参，传递的是整个数组。数组名作函数参数时，不是“值传送”，而是把实参数组的起始地址传递给形参数组，这样两个数组就共占有同一段内存单元。这种传递方式叫“地址传送”。

- 用数组名作函数参数，应该在主调函数和被调用函数分别定义数组，不能只在一方定义。



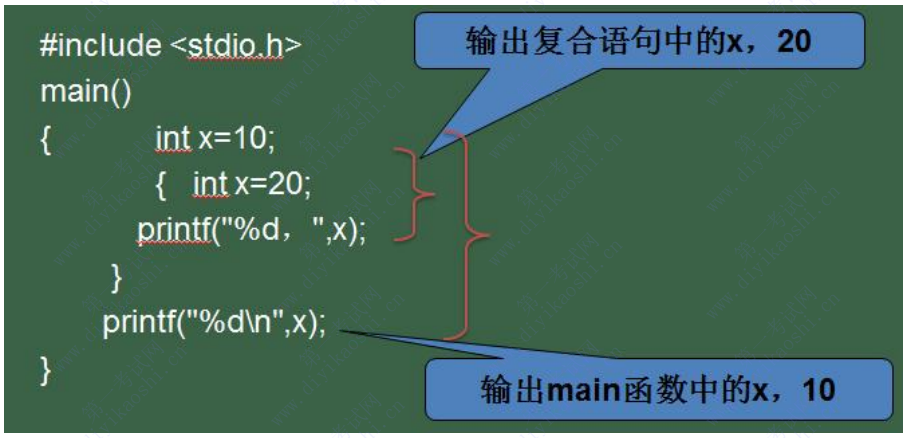
（6）局部变量和全局变量

局部变量：在一个函数内部定义的变量是内部变量(局部变量)，它的作用范围是从定义处开始到本函数的结束为止。

注意：

- 在主函数 main 中定义的变量只在主函数中有效。主函数也不能使用其它函数中定义的变量。
- 不同函数中可以使用名字相同的变量。
- 形参是局部变量。
- 在复合语句中定义的变量是局部变量，它们只在本复合语句中有效，离开该复合语句就无效。

例：分析以下程序的运行结果。



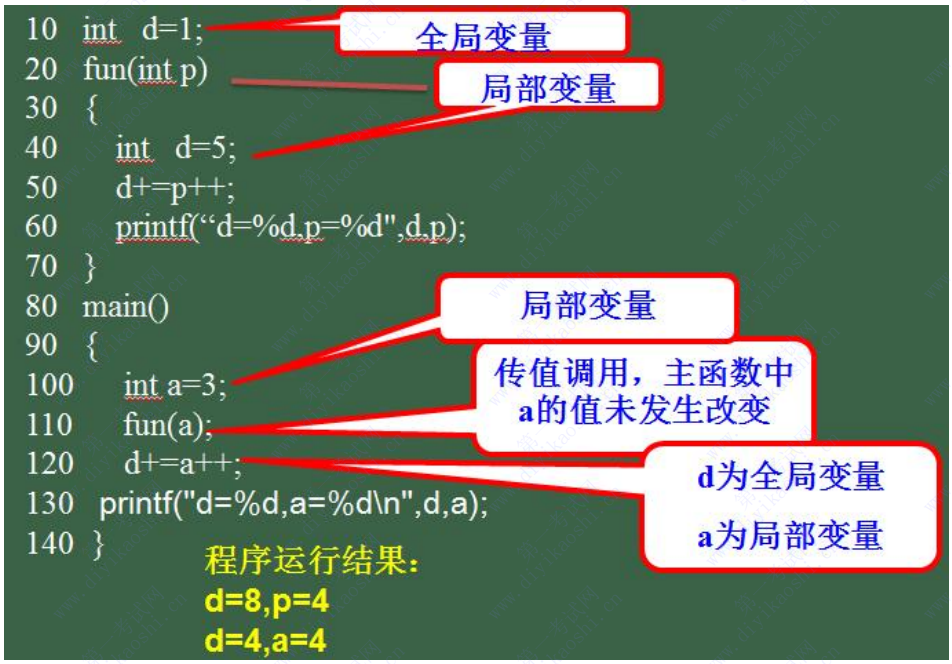
全局变量：在函数之外定义的变量，也称外部变量。

全局变量可以为本文件中其它函数所共有。它的有效范围：从定义变量的位置开始到本源程序文件结束。

在一个函数中既可以使用本函数中的局部变量，又可以使用有效的全局变量。

注意：

- Ø 全局变量的作用：增加了函数间数据联系的渠道。
- Ø 在同一个源文件中，全局变量与局部变量同名，则在局部变量的作用范围内，全局变量不起作用。



（7）变量的存储类别

根据变量存在期的不同，可以将变量的存储方式分为静态存储方式和动态存储方式。

- 局部变量的存储类别

① 自动变量是 C 程序中使用最多的一种变量。

[auto] 数据类型 变量名[= 初始表达式]，.....;

auto 可省略。自动变量是局部变量，未进行初始化时，自动变量的值是不定的。函数的形参也是一种自动变量；对同一函数的两次调用之间，自动变量的值是不保留的。

② 局部静态变量

static 类型说明符 变量名;

Ø 一个变量被声明为静态，在编译时即分配存储空间，在整个程序运行期间都不释放。

Ø 局部静态变量是在编译过程中赋初值的，且只赋一次初值，如未初始化，自动为 0。

```
#include <stdio.h>
main()
{       int i,j=2;
        for(i=0;i<3;i++)
        printf("%4d",function(j));
        printf("\n");
}
int function(int a)
{       int b=0;
    static int c;
    b++;
    c++;
        return (a+b+c);
}
```

	函数调用开始时		函数调用结束时	
	b	c	b	c
第一次调用	0	0	1	1
第二次调用	0	1	1	2
第三次调用	0	2	1	3

运行结果为： 4 5 6

③ 寄存器变量具有与自动变量完全相同的性质。通常把使用频率较高的变量（如循环次数较多的循环变量）定义为 register 类别。

Ø 只有局部自动变量和形参可以作为寄存器变量

Ø 只有 int、char 和指针类型变量可定义为寄存器型

- 全局变量的存储类别

① 外部全局变量

在多个源程序文件的情况下，如果在一个文件中要引用其它文件中定义的全局变量，则应该在需要引用此变量的文件中，用 extern 进行说明。

② 静态全局变量

在程序设计时，如果希望在一个文件中定义的全局变量仅限于被本文件引用，而不能被其它文件访问，则可以在定义此全局变量时前面加上关键字 static。

（8）内部函数和外部函数

- 内部函数：静态函数。内部函数只能被本文件中其它函数所调用，而不能被其它外部文件调用。

用存储类别 static 定义的函数称为内部函数，其一般形式为：static 类型说明符 函数名（形参表）

例如：static float sum(float x,float y){…}

- 外部函数：按存储类别 extern（或没有指定存储类别）定义的函数，作用域是整个程序的各个文件，可以被其它文件的任何函数调用。

extern 类型说明符 函数名（形参表）

例如：extern char compare(char s1,char s2)

{ …}

习题解析

1.下列叙述中错误的是（ ）

A.主函数中定义的变量在整个程序中都是有效的

B.在其它函数中定义的变量在主函数中也不能使用

C.形式参数也是局部变量

D.复合语句中定义的变量只在该复合语句中有效

2.下列正确的函数定义形式是（ ）

A.double fun(int x,int y)

B.double fun(int x;int y)

C.double fun(int x,int y);

D.double fun(int x,y)

3.若调用一个函数，且此函数中没有 return 语句，则正确的说法是该函数（ ）

A.没有返回值

B.返回若干个系统默认值

C.能返回一个用户所希望的函数值

D.返回一个不确定的值

4.下面函数调用语句含实参的个数是（　　）

```
func((exp1,exp2),(exp3,exp4,exp5));
```

A.1　　B.2

C.4　　D.5

5.在 C 语言中，函数的数据类型是指（　　）

A.函数返回值的数据类型

B.函数形参的数据类型

C.调用该函数时的实参的数据类型

D.任意指定的数据类型

6.下面程序的运行结果是（　　）。

```
#include <stdio.h>
int d=1;
fun(int p)
{    static int d=5;
     d+=p;
     printf("%d",d);
     return(d);
}
main()
{    int a=3;
     printf("%d\n",fun(a+fun(d)));
}
```

A. 6 9 9

B. 6 6 9

C. 6 15 15

D. 6 6 15

7.预处理命令

C 语言的预处理命令均以“#”打头，末尾不加分号，以区别于 C 语句、C 声明和定义。

C 提供的预处理功能主要有以下三种：

- Ø 宏定义；

Ø 文件包含；

Ø 条件编译。

（1）宏定义

在C语言源程序中允许用一个标识符来表示一个字符串，称为“宏”。 在C语言中，“宏”分为有参数和无参数两种。

① 不带参数的宏定义的一般形式为：

#define 标识符 字符串

其中的“#”表示这是一条预处理命令。 “define”为宏定义命令。“标识符” 就是所谓的符号常量，也称为“宏名”。“字符串”可以是常数、表达式、格式串等。

例如#define PI 3.1415926

② 带参数的宏定义

带参数的宏定义一般形式为： #define 宏名（参数表） 字符串

带参宏调用的一般形式为：宏名(实参表)；

#define S(a,b) a*b /*定义带参数的宏*/

调用时 area1=S(m,n); /*等价于 area1=m*n; */

③ 撤销宏定义命令

宏定义命令#define 应该写在函数外面，通常写在一个文件之首，这样这个宏定义在整个文件范围内都有效。但是，也可以用命令#undef 撤消已定义的宏，终止该宏定义的作用域。

（2）文件包含命令

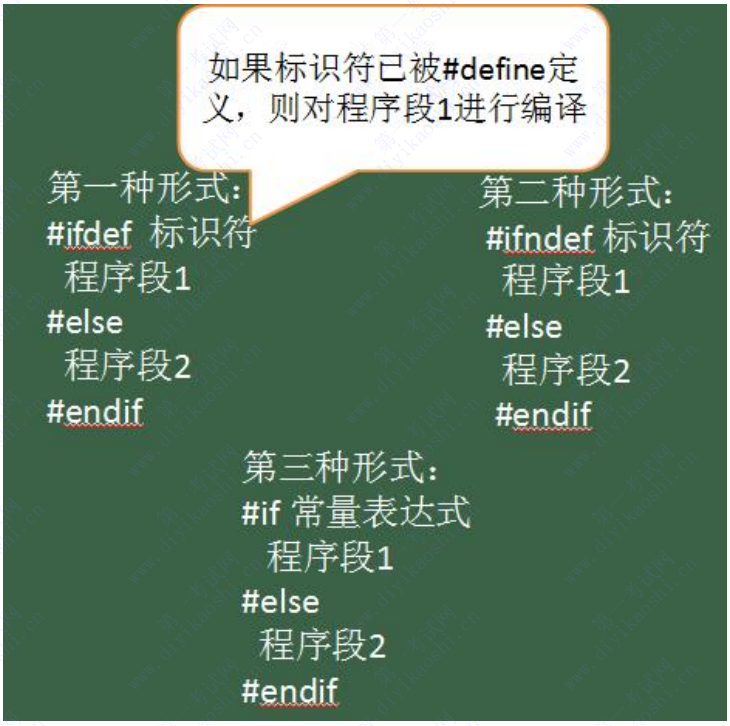
文件包含是指一个源文件可以将另一个源文件的全部内容包含进来，即将另一个文件包含到本文件之中,是以“#include”开头的预处理命令.

[格式 1]#include "文件名"

[格式 2]#include <文件名>

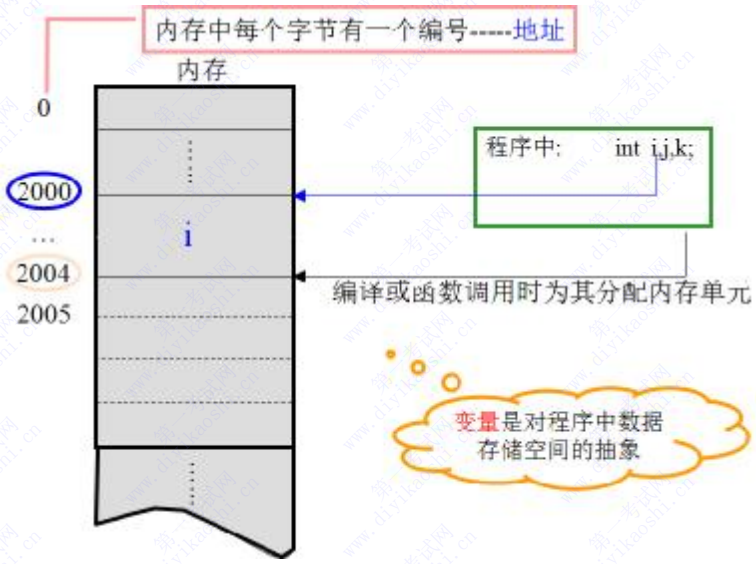
（3）条件编译命令

一般情况下，源程序中所有的行都参加编译。但有时希望其中一部分内容只在满足一定条件下才进行编译，即对一部分内容指定编译条件，这就是“条件编译”。



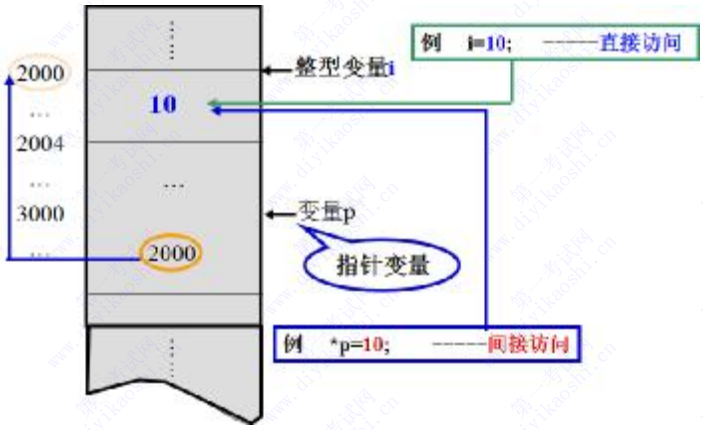
8.指针

（1）变量的地址和指针



直接访问：通过变量名来访问一个变量的值

间接访问：通过存放变量地址的变量去访问变量



（2）指针的概念

用来存放存储单元地址的存储单元被称为指针变量。

使用指针的目的是为了通过指针去访问内存储单元，采用这种方式访问内存储单元的速度和效率都优于普通变量。

- 指针变量的定义

类型说明符 *变量名 1，*变量名 2…;

例如:

```
int *p;        char *pch,*pname;
float *pf;
```

Ø 变量名前的“*”在定义时不能省略，它是说明其后变量是指针类型变量的标志。

Ø 其它类型的变量允许和指针变量在同一个语句中定义，例如：int m,n,*p,*q;

Ø 指针定义中的“数据类型”是指针指向的目标变量的数据类型，而不是指针变量的数据类型。

（3）指针运算

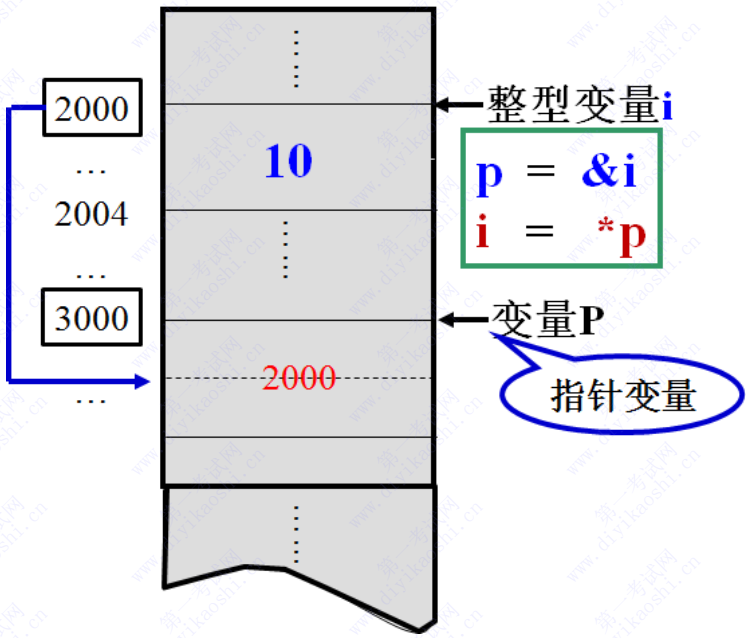
① &与*运算符：两者互为逆运算

&取变量的地址，单目运算符，自右向左。

*取指针所指向变量的内容，单目运算符，自右向左。

例如:

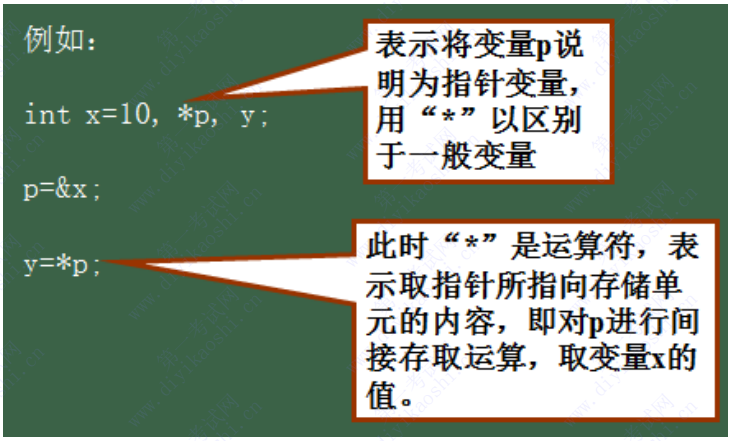
```
int i=10,*p;
p=&i;
```



p-----指针变量，它的内容是地址量

*p----指针的目标变量，它的内容是数据

&p---指针变量占用内存的地址



② 赋值运算

指针变量的初始化，就是在定义指针变量的同时为其赋初值。由于指针变量是指针类型，所赋初值应是一个地址值。

数据类型 *指针变量名 1=地址 1，*指针变量名 2=地址 2…;

其中的地址形式有多种，如：&变量名、数组名、其它的指针变量等。例如：

```
int i;
int *p=&i;
char s[20];
char *str=s;
```

说明：

- Ø 不能用尚未定义的变量给指针变量赋初值。
- Ø 当用一个变量地址为指针变量赋初值时，该变量的数据类型必须与指针变量指向的数据类型一致。
- Ø 除 0 之外，一般不把其他整数作为初值赋给指针变量。0 为空指针。

使用赋值语句赋值，一般格式如下：

指针变量=地址;

例如：

```
int m=100,*p,*q;
p=&m;  /*将变量 m 的地址赋给指针变量 p*/
```

③ 空指针与 void 指针

空指针就是不指向任何对象的指针，表示该指针没有指向任何内存单元。

空指针常常用来初始化指针，避免野指针的出现。

```
例如：
 #define NULL 0
 int *p;
 p=0;
或
 p=NULL;
```

如果一个指针被声明为 void 类型，可以称之为“无类型指针”，可以指向任意类型的数据。

```
int * a, *b;
float *c;
void * d;
d = a;          /*正确*/
d = c;          /*正确*/
```

Ø 在使用 void 指针时必须对其进行强制类型转换，将 void 指针转换成它所指向单元的实际类型。

```
int x = 100;
void * p = &x;          /*定义void指针p指向x*/
int *q = NULL;          /*定义整型指针q*/
 /*printf("*p=%d\n",*p);    错误，非法使用指针p*/
 printf("*p =%d\n",*(int *)p); /*正确，输出p指向单元内容*/
 /*q = p;          错误，非法，void指针赋给整型指针*/
 q = (int *)p;       /*正确，合法，void指针赋给整型指针*/
 printf("*q=%d\n",*q);    /*输出指针q指向单元内容*/
```

```
main( )
{int x=1,y=2,d;
 int *px=&x;
 y=*px+1;

printf("%d",*px);

d=sqrt((double)*
px);
 *px=0;
 *px+=1;
 (*px)++;
 y=*px;

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

```
将变量x地址赋给指针px，使指针px指向变量x。
将目标变量加1(即x+1)后的值赋给变量y。
输出x的当前值。
将x转换成double型后求平方根，将结果赋给d。
即x=0。
即x=x+1或(*px)++。
即 x++。

程序运行结果:
1
2
```

指针的加 1 运算后就指向下一个数据的位置。

指针的减 1 运算后就指向前一个数据的位置。

*、 ++优先级相同，结合性均是右结合。

* px++等价于*（px++）

```
main( )
{ int *px,x;
  int z1,z2,z3,z4;
  x=10;
  px=&x;
  z1=*px++;
  printf("z1=%d\n",z1);      z1=10
  x=10;
  px=&x;
  z2=(*px)++;
  printf("z2=%d\n",z2);      z2=10
  x=10;
  px=&x;
  z3=++(*px);
  printf("z3=%d\n",z3);      z3=11
  x=10;
  px=&x;
  z4=*++px;
  printf("z4=%d\n ",z4);     z4=×× 是一个不确定的随机数。
}
```

（4）指针与数组

① 一维数组与指针

Ø 数组的指针是指数组的起始地址。

Ø 数组元素的指针是数组元素的地址。

Ø 一个数组是由连续的一块内存单元组成的。

Ø 数组名就是这块连续内存单元的首地址。

Ø 定义一个指向数组元素的指针变量的方法，与以前介绍的指针变量相同。

如果指针变量 p 的初值为&a[0],则：

Ø p+i、a+i、&a[i]等价，指向 a 数组的第 i 个元素。

Ø *(p+i)、*(a+i)、 a[i]等价，第 i 个元素的值。

Ø 规定：p[i]与*(p+i)和 a[i]等价。

例如：

int a[10]；定义 a 为包含 10 个整型数据的数组

int *p； 定义 p 为指向整型变量的指针

p=&a[0]； 对指针变量赋值

把 a[0]元素的地址赋给指针变量 p。也就是说，p 指向 a 数组的第 0 号元素。

② 二维数组的指针

若有：int a[4][3], *p;

p = &a[0][0];

数组名称	一维下标的指针含义	二维数组下标表示	元素在内存中的存储顺序	通过指针访问元素	通过指针按下标访问元素
a	a[0]	a[0][0]		p	p[0]
		a[0][1]		p+1	p[1]
		a[0][2]		p+2	p[2]
	a[1]	a[1][0]		p+3	p[3]
		a[1][1]		p+4	p[4]
		a[1][2]		p+5	p[5]
	a[2]	a[2][0]		p+6	p[6]
		a[2][1]		p+7	p[7]
		a[2][2]		p+8	p[8]
	a[3]	a[3][0]		p+9	p[9]
		a[3][1]		p+10	p[10]
		a[3][2]		p+11	p[11]

③ 指针与字符串

用字符数组存放一个字符串，例如：

char string [30]= "This is a string.";

用字符串指针指向一个字符串。例如：

```
char *str = "This is a string.";
```

（5）指针与函数

① 指针作为函数参数

```
main ( )
{ int  a, b;
  printf("please  input  a and b:");
  scanf("%d%d",&a,&b);
  printf("Before Swap: a=%d,  b=%d\n",a,b);
  swap(a,b);
printf("After Swap:  a=%d,  b=%d\n",a,b);
}
```

```
void swap(int p1,int p2)
{ int t;
  t=p1;
  p1=p2;
  p2=t;
}
```

```
运行结果:
please  input  a and b:3 5
Before Swap: a=3,  b=5
After Swap:  a=3,  b=5
```

```
main ( )
{ int  a, b;
  printf("please  input  a and b:");
  scanf("%d%d",&a,&b);
  printf("Before Swap: a=%d,  b=%d\n",a,b);
  swap(&a,&b);
printf("After Swap:  a=%d,  b=%d\n",a,b);
}
```

```
void swap(int *p1,int *p2)
{ int t;
  t=*p1;
  *p1=*p2;
  *p2=t;
}
```

```
运行结果:
please  input  a and b:3 5
Before Swap: a=3,  b=5
After Swap:  a=5,  b=3
```

② 指针作为函数的返回值

可以将地址作为函数返回值，当将地址作为函数返回值时，该函数被称为指针函数。其定义形式为：

数据类型　*　函数名（形参表）

{　函数体；

}

③ 函数的指针

在定义一个函数之后，编译系统为每个函数确定一个入口地址，当调用该函数的时候，系统会从这个“入口地址”开始执行该函数。存放函数的入口地址的指针就是一个指向函数的指针，简称函数的指针。函数的指针的定义方式是：

类型标识符 （* 指针变量名）()

（6）指针数组和指向指针的指针

① 指针数组

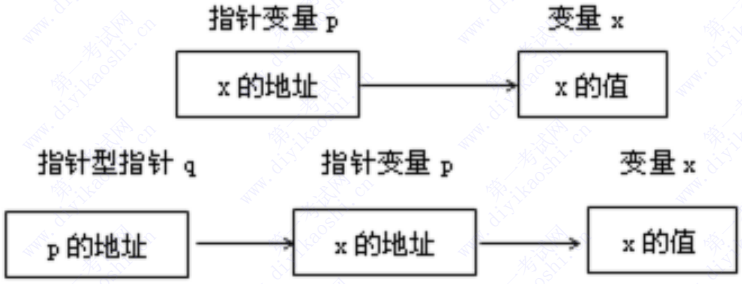
一个数组中的每个元素均为指针类型，即由指针变量构成的数组，这种数组称之为指针数组，它是指针的集合。指针数组说明的形式为：

类型 * 数组名[常量表达式]；

② 指向指针的指针

一个指针可以指向任何一种数据类型，包括指向一个指针。当指针变量 p 中存放另一个指针 q 的地址时，则称 p 为指针型指针，也称多级指针。指针型指针的定义形式为：

类型标识符 ** 指针变量名;



例如

```
int x ; /* 定义整型变量 x */
int *p; /* 定义指向整型变量的指针 p */
int **q; /* 定义多级指针 q */
p=&x; /* 指针 p 指向变量 x */
q=&p      //则有：**q = *(*q) = *p = x
```

习题解析

1. 已知：int *p, a；则语句“p=&a;”中的运算符“&”的含义是（ ）。

A. 位与运算 B. 逻辑与运算 C. 取指针内容 D. 取变量地址

2. 说明语句“int (*p)( );”的含义是（ ）。

A. p 是一个指向一维数组的指针变量

B. p 是指针变量，指向一个整型数据

C. p 是一个指向函数的指针，该函数的返回值是一个整型。

D. 以上都不对

3.已知：char b[5]，*p=b；则正确的赋值语句是（　　）。

A. b="abcd"；　B. *b="abcd"；　C. p="abcd"；　D. *p="abcd"；

4.已知：int a[10]={1,2,3,4,5,6,7,8,9,10}，*p=a；则不能表示数组 a 中元素的表达式是（　　）。

A. *p　B. a[10]　C. *a　D. a[p-a]

5.下面程序的运行结果是（　　）。

```
#include<stdio.h>
void delch(char *s)
{ int i,j;
  char *a;
  a=s;
  for(i=0,j=0;a[j]!='\0';i++)
  if(a[j]>='0'&&a[j]<='9'){s[j]=a[j];j++;}
  s[j]='\0'; }
main()
{ char*item="a34bc";
delch(item);
printf("\n%s",item); }
```

A.abc　B.34

C.a34　D.a34bc

9.结构体与共用体

（1）结构体（structure）是不同数据类型的数据所组成的集合体，是构造类型数据。

每一个结构体有一个名字，称为结构体名。一个结构体由若干成员组成，每个成员有自己的名字，称为结构体成员名。每个成员的数据类型可以不同，也可以相同。

① 声明一个结构体类型的一般形式为：

struct 结构体名

{　数据类型 成员 1 的名字；

　数据类型 成员 2 的名字；

　数据类型 成员 3 的名字；

……

};

例如：

```
struct student
{int sID;              //学号
char sSex;                  //性别
int sMath;                  //高数成绩
int sEng;            //英语成绩
int sC;                //C语言程序设计成绩
};
```

② 结构体变量的定义

Ø 先声明结构体类型再定义变量名，如 struct student S1;

Ø 在声明结构体类型的同时定义结构体变量

```
struct 结构体名
{
        数据类型 成员1的名字;
        数据类型 成员2的名字;
        数据类型 成员3的名字;
                ……
}结构体变量名表;
```

Ø 直接定义结构体变量，不出现结构体名

```
struct
{
    数据类型 成员1的名字;
    数据类型 成员2的名字;
    数据类型 成员3的名字;
            ……
}结构体变量名表;
```

③ 结构体变量的初始值。

结构体变量初始化的格式：

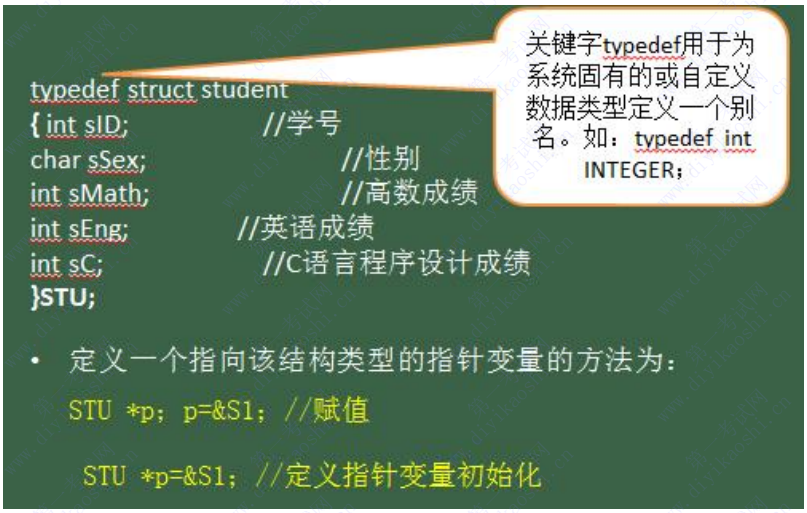
struct 结构体名 结构体变量名={ 初始数据 };

④ 结构体变量的引用

结构体变量名.成员名

如：S1.sSex

⑤ 结构体指针变量是指向结构体变量的指针，该指针变量的值就是结构体变量的起始地址，其目标变量是一个结构体变量。



C 语言规定了两种用于访问结构体成员的运算符。

- Ø 成员运算符，也称圆点运算符（前面介绍过）；
- Ø 另一种是指向运算符，也成箭头运算符，其访问形式为：指向结构体的指针变量名->成员名

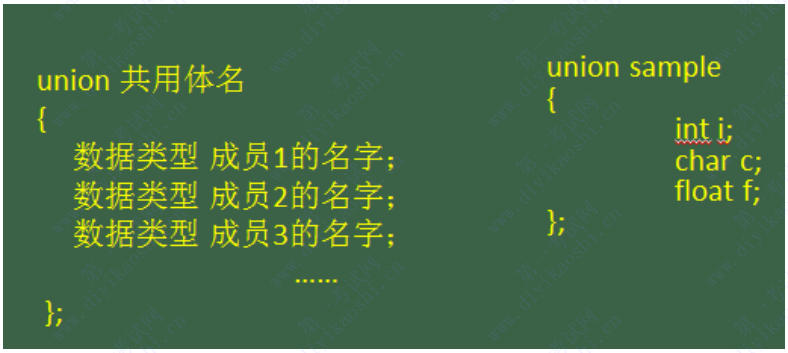
如：p->sEng=90;等价于(*p).sEng=90;

⑥ 指向结构体数组的指针

定义一个结构体数组 STU s[30]；则定义结构体指针变量 p，并将其指向结构体数组 s 有以下三种方法：

- Ø STU *p=s;
- Ø STU *p=&s[0];
- Ø STU *p; p=s;

（2） 共用体，有的也称为联合（Union），是将不同类型的数据组织在一起共同占用同一段内存的一种构造数据类型。声明一个共用体类型的一般形式为：



（3）枚举类型，即“一一列举”之意，当某些量仅由有限个数据值组成时，通常用枚举类型表示。枚举数据类型描述的是一组整型值的集合。声明枚举类型需用关键字 enum，如：enum weekday{sun,mon,tue,wed,thu,fri,sat};

定义：enum weekday a;则枚举变量 a 的取值只有 7 种，即 sun,mon,tue,wed,thu,fri,sat。

习题解析

1.设有以下说明语句，则下面的叙述不正确的是(　　)

```
struct stu
{ int a;
float b;
} stutype;
```

A.struct 是结构体类型的关键字。

B.struct stu 是用户定义的结构体类型。

C.stutype 是用户定义的结构体类型名。

D.a 和 b 都是结构体成员名。

2.若有下列定义：

```
struct b
{ float a[5];
  double c;
  int d;
}x;
```

则变量 x 在内存中所占的字节为（　　）

A.6　　B.10　　C.30　　D.14

3.已知学生记录描述为：设变量 s 所代表的学生生日是“1990 年 8 月 16 日”，下列对“生日”的正确赋值是（　　）。

```
struct date
{
        int year;
        int month;
        int day;
};
struct student
{
        int sID;
        struct  date  birth;

};
struct student s;
```

A. year=1990; month=8; day=16;

B. birth.year=1990; birth.month=8; birth.day=16;

C. s.birth.year=1990; s.birth.month=8; s.birth.day=16;

D. s.year=1990; s.month=8; s.day=16;

华图网校
V.HUATU.COM
SINCE 2001
乐享品质　高效学习

■ 华图网校介绍

华图网校（V.HUATU.COM）于2007年3月由华图教育投资创立，是华图教育旗下的远程教育高端品牌。她专注于公职培训，　目前拥有遍及全国各地500万注册用户，已成为公职类考生学习提高的专业门户网站。

华图网校是教育部中国远程教育理事单位。她拥有全球最尖端高清录播互动技术和国际领先的网络课程设计思想，融汇华图教育十余年公职辅导模块教学法，凭借强大师资力量与教学资源、利用教育与互联网的完美结合，真正为考生带来“乐享品质”的学习体验，通过“高效学习”成就品质人生。

华图网校课程丰富多元，涵盖公务员、事业单位、招警、法院、检察院、军转干、选调生、村官、政法干警、三支一扶、乡镇公务员、党政公选等热门考试、晋升及选拔。同时，华图网校坚持以人为本的原则，不断吸引清华、北大等高端人才加入经营管理，优化课程学习平台，提升用户体验，探索网络教育新技术和教学思想，力争为考生提供高效、个性、互动、智能的高品质课程和服务。

华图网校将秉承“以教育推动社会进步”的使命，加快网站国际化进程，打造全球一流的网络学习平台。

我们的使命：以教育推动社会进步

我们的愿景：德聚最优秀人才，仁就基业长青的教育机构

我们的价值观：诚信为根、质量为本、知难而进、开拓创新。

■ 咨询电话：400-678-1009
■ 听课网址：v.huatu.com（华图网校）