第一章　程序设计和C语言

1、机器指令：计算机能直接识别和接受的二进制代码称为机器指令。

2、机器语言：机器指令的集合就是该计算机的机器语言。

3、符号语言：用一些英文字母和数字表示一个指令，例如：“ADD”代表“加”，“SUB”代表“减”，“LD”代表“传送”。

4、计算机不能直接识别和执行符号语言的指令，需要一种称为汇编程序的软件把符号语言指令转换为机器指令。一般一条符号语言的指令对应转换成一条机器指令。转换过程称为“代真”或“汇编”，因此，符号语言又称为符号汇编语言或汇编语言。

5、高级语言：这种语言功能很强，且不依赖于具体机器，它与具体机器距离较远，故称为高级语言。当然，计算机也不能识别高级语言程序。

6、源程序：高级语言写的程序。

7、目标程序：机器指令的程序。

8、用一种称为编译程序的软件把高级语言写的程序（源程序）转换为机器指令的程序（目标程序），然后让计算机执行指令程序，最终得到结果。

9、高级语言的一个语句往往对应多条机器指令。

10、高级语言经历的发展阶段：

非结构化语言——结构化语言——面向对象语言（C++,C#,Visual Basic和Java）

11、C语言的特点：

（1）语言简洁、紧凑，使用方便、灵活

（2）运算符丰富

（3）数据类型丰富（整形、浮点型、字符型、数组类型、指针类型、结构体类型和共用体类型.....）

（4）具有结构化的控制语句（if.....else、while、do.....while、switch和for语句）

（5）语法限制不太严格，程序设计自由度大

（6）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

（7）可移植性好

（8）生成目标代码质量高，程序执行效率高

12、最简单的C语言程序举例

```
# include <stdio.h>
```

```
Int main( )                //  定义主函数
{
   printf(“Hello world! \n”);     //   输出
   return 0;                //     函数执行完毕时返回函数值 0
}
```

13、stdio.h 是系统提供的一个文件名，stdio 是“standard input&output”的缩写，文件后缀.h 是头文件（hearder file）。在程序中如果要使用到标准函数库中的输入输出函数，应该在本文件模块的开头写上：# include <stdio.h>。

14、“//”表示“注释”，只能占一行或在一行内容的右侧，

“/* ..... */”表示“注释”，可以占多行。注释部分程序不执行

注意，在字符串中的//和/*都不作为注释使用，而是作为字符串的一部分。 如：printf（“//哈哈\n”）；或 printf（“/*哈哈*/\n”）输出结果分别是 “//哈哈”或“/*哈哈*/”

15、C 语言程序结构的特点：

（1）一个程序由一个或多个源程序文件组成，源程序文件可包括三个部分：预处理指令、全局声明和函数定义

（2）函数是 C 程序的主要组成部分，一个 C 语言程序由一个或多个函数组成，其中必须有且只有一个 main 函数

（3）一个函数包括两部分，函数首部和函数体

（4）程序总是从 main 函数开始执行的

（5）程序中对计算机的操作是由函数中的 C 语句完成的

（6）在每个数据声明或语句的最后必须有一个分号

（7）C 语言本身不提供输入输出语句，输入输出语句的操作是由库函数 scanf 和 printf 等函数来完成的

（8）程序应当包含注释

第二章　算法—程序的灵魂

1、一个程序主要包括以下两个方面的信息：

（1）对**数据**的描述。在程序中要指定用到哪些数据以及这些数据的类型和数据的组织形式。这就是**数据结构**。

（2）对**操作**的描述。即要求计算机进行操作的步骤，也就是**算法**。

算法 + 数据结构 = 程序

2、算法的特性：

（1）有穷性，一个算法应该包含有限的步骤，而不能是无限的。

（2）确定性，算法中的每一个步骤都应当是确定的，而不应当是含糊的、模棱两可的。

（3）有零个或多个输入，所为输入指在执行算法时需要从外界取得必要的信息。

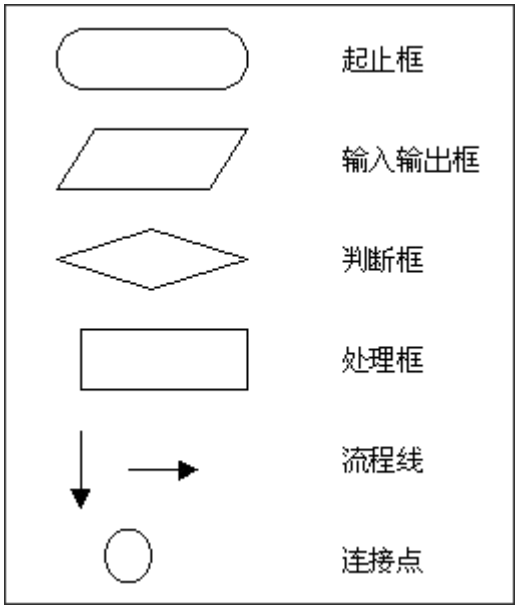
（4）有一个或多个输出，算法的目的就是为了求解，而“解”就是输出。

（5）有效性，算法中的每一个步骤都应当能有效地执行，并得到确定的结果。

3、怎样表示一个算法：

（1）自然语言

（2）传统流程图语言



（3）结构化流程图

（4）伪代码

（5）N-S 流程图

（6）计算机语言

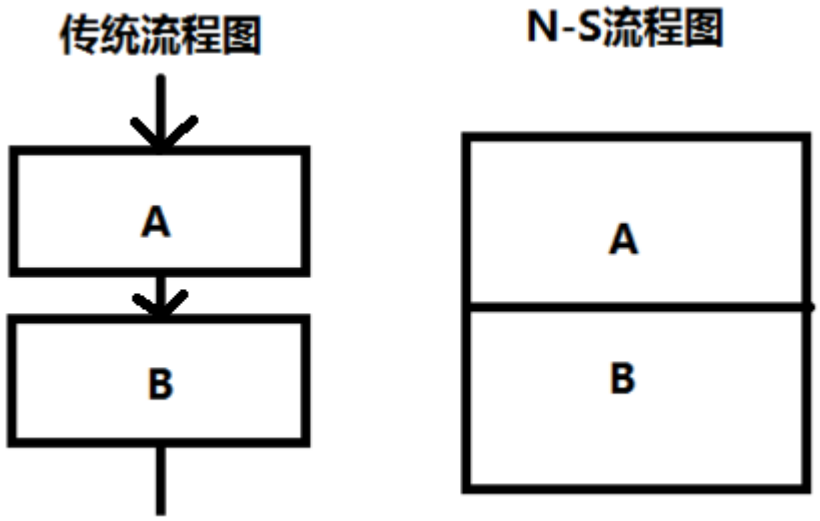
4、三种基本结构：

顺序结构

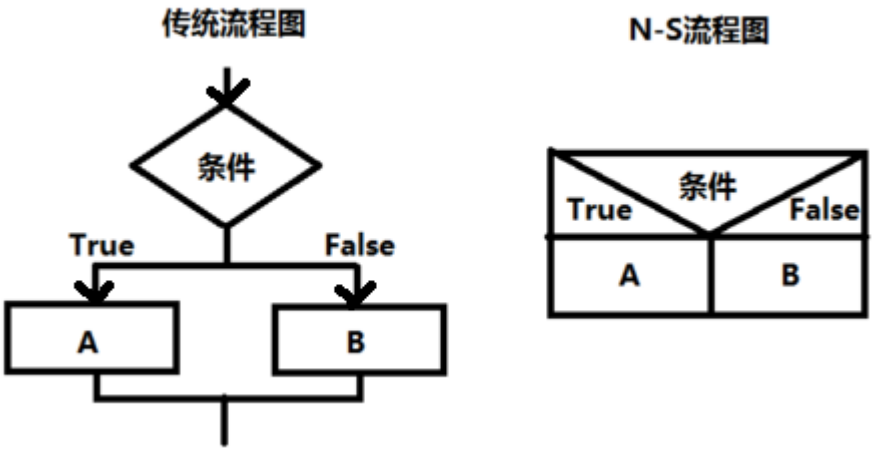
选择结构

循环结构 【循环结构又分为当型（while 型）循环结构和直到（until）型循环结构】

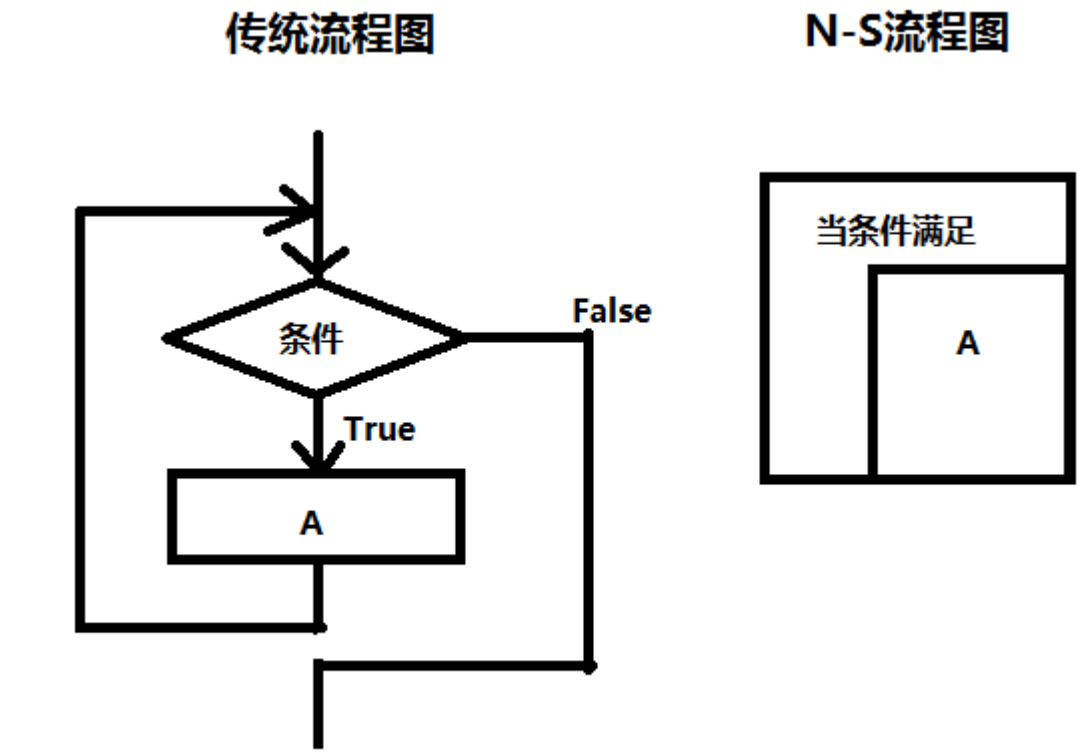
5、传统流程图和 N-S 流程图的区别



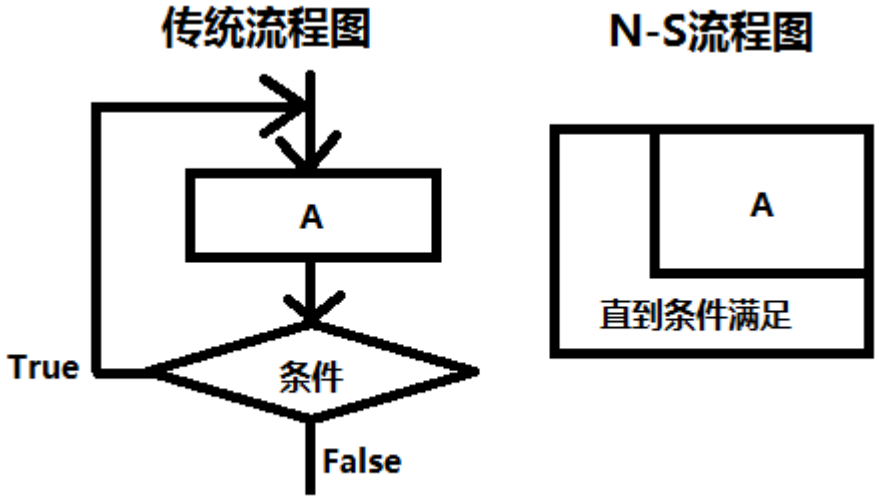
顺序结构



选择结构



当循环结构



直到循环结构

第三章　最简单的C程序设计

1、常量：在程序运行过程中，其值不能被改变的量称为常量。

（1）整形常量，如10000,12345,0，-345

（2）实型常量，有两种表现形式

①十进制小数形式，由数字和小数点组成。如123.45，0.25，0.0，-32.44等

②指数形式，如12.34e3（代表12.34×10^{3}），0.145E-25（代表0.145×10^{-25}）。e或E之前必须要有数字，之后必须为整数。

（3）字符常量

①普通字符，用单撇号括起来的一个字符，如‘a’，‘Z’，‘3’‘？’，‘#’。不能写成‘ab’或‘12’。请注意，单撇号只是界限符，字符常量存储在计算机存储单元中，并不是存储字符（如啊a,Z,3...）本身，而是以其代码（一般采用ASCII码）存储的，例如字符‘a’的ASCII化代码是97,因此,在存储单元中存放的是97(以二进制形式存放)。

②转义字符，以字符\开头的字符序列，意思是将“\”后面的字符转换成另外的意义，如“\n”中的“n”不代表字母n而是作为“换行”符。

常用的转义字符及其作用

转义字符	意义	ASCII码值（十进制）
\a	响铃(BEL)	007
\b	退格(BS)	008
\f	换页(FF)	012
\n	换行(LF)	010
\r	回车(CR)	013
\t	水平制表(HT)	009
\v	垂直制表(VT)	011
\\	反斜杠	092
\?	问号字符	063
\'	单引号字符	039
\"	双引号字符	034
\0	空字符(NULL)	000
\ddd	任意字符	三位八进制
\xhh	任意字符	二位十六进制

（4）字符串常量，如“body”，“123”等，用双撇号把若干个字符括起来，字符串常量是双撇号中的全部字符（但不包括双撇号），不能错写成‘body’，‘123’，单撇号内只能包含1个字符，双撇号内可以包含一个字符串。

（5）符号常量，用#define指令，指定用一个符号名称代表一个常量。

如#define PI 3.1416　　//注意行末没有分号。

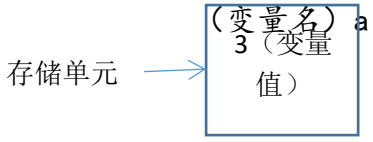
使用符号常量的好处：

①含义清楚，如PI代表圆周率，sum代表和。

②做到“一改全改”只需要改变“3.1416”就可以改变所有 PI 的值。

注意：要区分符号常量和变量，符号常量不占内存，只是一个临时符号，在预编译后这个符号就不存在了，故不能对符号常量赋以新值。为了与变量名相区别，习惯上符号常量用大写表示。

2、变量：变量代表一个有名字的、具有特定属性的一个存储单元。它用来存放数据，也就是存放变量的值。在程序运行过程中，变量的值是可以改变的。**变量必须先定义后使用**。在定义时指定变量的名字和类型。如图注意变量名和变量值的区别。



3、常变量：

如 const int a = 3 表示 a 被定义为一个整型变量，指定其值为 3，而且在变量存在期间其值不能改变。

①常变量与常量的异同：常变量具有变量的基本属性：有类型，占存储单元，只是不允许改变其值。可以说常变量是有名字的不变量，而常量是没有名字的不变量。有名字就便于在程序中被引用。

②常变量和符号常量的区别：

如

```
# define Pi 3.1415926              //定义符号常量，末尾无分号
const float pi = 3.1415926 ;       //定义常变量
```

符号常量 Pi 和常变量 pi 都代表 3.1415926，在程序中都能使用。

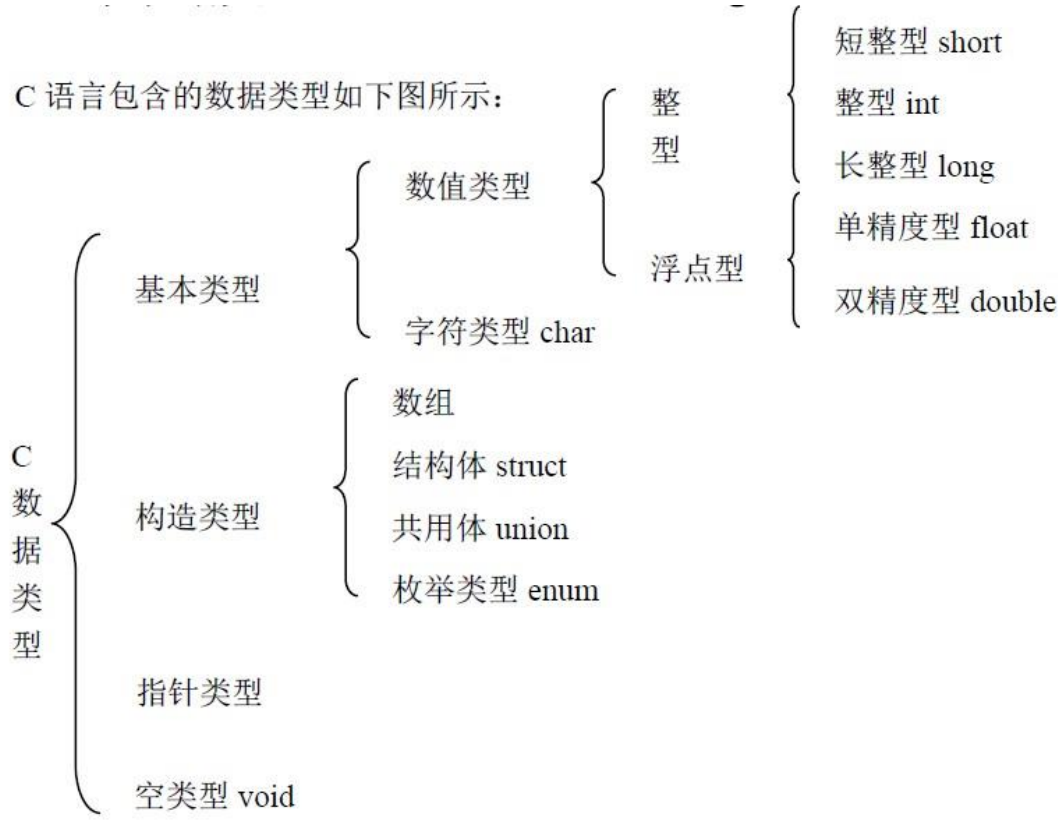
定义符号常量# define 指令，它是预处理指令，它只是用符号常量代表一个字符串，在预编译时仅是进行字符替换，在预编译后，符号常量就不存在了（全部置换成了 3.1415926），对符号常量的名字是不分配存储单元的。而常变量要占用存储单元，有变量值，只是该值不改变而已。从使用至角度来说，常变量具有符号常量的优点，而且使用更方便。有了常变量以后，可以不必多用符号常量。

4、标识符：在计算机高级语言中，用来对变量、符号常变量、函数、数组、类型等命名的有效字符序列统称为标识符。C 语言的标识符只能由字母、数字和下划线三种字符组成，且第一个字符必须为字母或下划线。

下面是不合法的字符：

M.D.John，￥123，#33,3D64，a>b

注意：编译系统将大写字母和小写字母认为是两个不同的字符。



5、数据类型：

（1）基本类型（int 型）【2 或 4 个字节】

在存储单元中的存储方式是：用整型变量的值的补码形式存放。

①一个正数的补码是此数的二进制形式，如 5 的二进制形式是 101，如果用两个字节存放一个整数，则 5 在存储单元中数据形式为 0000000000000101。

②一个负数的补码：先将此数的绝对值写成二进制形式，然后对其各位取反，再加 1。比如求-5 的补码，

5 的源码为 0000000000000101（a）

按位取反　1111111111111010（b）

再加一　　1111111111111011（c）

（c）为-5 的补码

【有关补码知识可以参考计算机原理的书籍】

③如果给整型变量分配 2 个字节，则在存储单元中能存放的最大值为 0111111111111111，第 1 位的 0 代表正数，此数值为（2^{15}-1），即 32767，最小值为 1000000000000000，此数为-2^{15}，即-32768。因此一个整型变量的值的范围是-32768~32767。超出此范围，就出现值的值的“益处”。如果给整型变量分配 4 个字节（Visual C++），其能容纳的数值范围是-2^{31}~（2^{31}-1），即-2147483648~2147483647。

（2）短整型（short int）

类型名为 short int 或 short。如用 Visal C++ 6.0，编译系统分配给 int 数据 4 个字节，短整型 2 个字节。存储方式与 int 型相同，一个短整型变量的值的范围是-32768~32767。

（3）长整型（long int）

类型名为 long int 或 long，分配 4 个字节，范围是-32768~32767.

（4）双长整型（long long int）

一般分配 8 个字节，C99 新增的类型，许多编译系统尚未支持。

6、说明：C 标准没有具体规定各种类型数据所占用存储单元的长度，这是由编译系统自行决定的。C 标准只要求 long 型数据长度不短于 int 型，short 型不长于 int 型，即 short ≤ int ≤ long int【注意溢出问题】

7、整型变量的符号属性

整型数据常见的存储空间和值的范围		
类型	字节数	取值范围
int(基本整型)	2	从-32768至32767
	4	从-2147483648至2147483647
unsigned int(无符号基本整型)	2	从0至65535
	4	从0至4294967295
short(短整型)	2	从-32768至32767
unsigned short(无符号短整型)	2	从0至65535
long(长整形)	4	从-2147483648至2147483647
unsigned long(无符号长整型)	4	从0至4294967295
long long(双长型)	8	从-9223372036854775808至9223372036854775807
unsigned long long(无符号双长整形)	8	从0至18446744073709551615

采用 n 位存储时

带符号[singned]（不写[signed]默认为带符号整型）整型数据的表示范围：-2^{n-1}~$2^{n-1}-1$

无符号[unsigned]整型数据的表示范围：0~2^n-1

有符号整型数据存储单元中最高位代表符号（0 为正，1 为负）

无符号整型数据存储单元全部存放数据

说明：

①只有整型（包括字符型）可以加 signed 或 unsigned 修饰，实型数据（浮点型）不能加。

②对无符号整型数据用“%u”格式输出，%u 表示用无符号十进制数的格式输出。如

```
unsigned short price = 50;      //定义无符号短整型变量
printf(“%u\n”, price);          //指定用无符号十进制格式输出
```

注意 unsigned 不能指定负数

思考：将“50”改成“-1”，输出结果会如何，为什么？

得到结果为 65535

原因：系统对-1 先转换成补码形式，然后再把它存储到 price 变量中，由于 price 是无符号短整型变量，其补码第一位不代表符号，按“u%”的格式输出就是 65535。

8、字符与字符代码并不是任意写一个字符，计算机都能识别的，目前大多数系统采用 ASCII 字符集。

9、字符是以整数形式（字符的 ASCII 代码）存放在内存单元中的。

10、注意：字符‘1’和整数 1 是不同的概念，字符‘1’只是代表一个形状为‘1’的符号，在需要时按原样输出，在内存中以 ASCII 码形式存储。占一个字节，而整数 1 是以整数存储方式（二进制补码方式）存储的，占 2 个或 4 个字节。

11、字符变量：字符变量是用类型符 char（character 的缩写）定义字符变量。

如：char c = ‘？’；

①定义 c 为字符型变量并使初始值为字符‘？’。字符‘？’的 ASCII 代码是 63，，系统把整数 63 赋给变量 c。

②c 是字符变量，实质上是一个字节的整型变量，由于用来存放字符，所以称为字符变量。

③以十进制整数形式和字符形式输出：printf（“%d\n, %c\n”, c, c）;输出结果为：63, ?

12、前面介绍了整型变量可以用 signed 和 unsigned 修饰符表示符号属性。那么字符类型也属于整型，也可以用 signed 和 unsigned 修饰符。

类型	字节数	取值范围
signed char（有符号字符型）	1	-128~127（即-2^7~2^7-1）
unsigned char（无符号字符型）	1	0~255（即 0~2^8-1）

例：signed char c = 255（编译时出现警告）

unsigned char c = 255（不会出现告警）

说明：在使用有符号字符型变量时，允许存储的值为-128~127，但字符的代码不可能为负值，所以在存储字符时实际上只用到 0~127 这一部分，其第一位都是 0。

13、如果将一个负整数赋值给有符号字符型变量时合法的，但它不代表一个字符，而是作为**一个字节**的整型变量存储这个负整数。如 signed char c = -6; printf(“%d\n”, c);

14、如果在定义字符变量时既不加 signed 也不加 unsigned，C 标准并未规定是按 signed char 处理还是按 unsigned char 处理，由各编译系统自己决定。这与其他整型的处理方法不同，如 int 就默认为 signed int。

15、浮点型数据：用来表示带小数点的实数。在 C 语言中，实数是以指数形式存放在存储单元中的。

（1）实数的指数形式称为浮点数。

（2）小数点前面为 0，后面第 1 位不为 0 的表示形式称为规范化的指数形式。

如 3.14 的规范化的指数形式是 0.314×10^1。

类型	字节数	有效数字	数值范围（绝对值）
float （单精度）	4	6	0 以及 1.2×10^{-38}~1.2×10^{38}
double 双精度	8	15	0 以及 2.3×10^{-308}~1.7×10^{308}
long double 长双精度	8	15	0 以及 2.3×10^{-308}~1.7×10^{308}
long double	16	19	0 以及 3.4×10^{-4932}~3.4×10^{4932}

16、实型数据的有关情况

17、C 编译系统把浮点型(实型)常量都按双精度处理,分配 8 个字节。如有:float a = 3.14159; //按双精度处理，分配 8 个字节，编译系统会发出“告警”，一般不会结果的正确性，但会影响结果的精度。可以在末尾加专用字符,强制指定常量类型。如: float a = 3.14159f。

18、运算符

①基本的运算符: +(加)， -（减）， *（乘）， /（除）， %（取余）。

②自增、自减运算符

++i / --i （在使用 i 之前，先使 i 的值加/减 1）

i++ / i-- （在使 i 用之后，使 i 的值加/减 1）

例： ①int i = 3;

```
i++或++1;
printf("%d\n", i);      //输出的结果都为 4
```

②int i = 3;

```
printf("%d\n", ++i);   //输出结果为 4，若改为 i++，则输出结果为 3。
```

19、注意: 自增自减运算符都只能用于变量，而不能用于常量或表达式，如 5++或（a+b）++是不合法的。

20、不同类型数据之间的混合运算:

①+、-、*、/运算中有一个数为浮点型，则结果为 double 型。

②int 型和浮点型进行运算，先把 int 型或 float 型转换成 double 型，然后再进行运算，结果是 double 型。

③字符（char）型数据与整形数据进行运算,就把字符的 ASCII 代码与整形数据进行运算。如 12+‘A’，相当于 12+65=77，如果是字符型与实型数据，则字符的 ASCII 代码转换为 double 型数据再进行运算。以上转换时编译系统自动完成的，用户不必过问。

21、强制类型转换: 如

```
(double) a  //将 a 转换成 double 类型
(int)(x+y)   //将 x+y 的值转换成 int 型
```

其一般形式为（类型名）（表达式）

22、注意: 在强制转换时，原变量的类型和值并未发生变化。例如: a = (int)x,如果已定义 x 为 float 型变量，a 为整型变量。强制转换时，（int）x 得到一个 int 型的临时值赋给 a，赋值后该临时值就不存在了。X 的值与类型并未发生变化。

注: x%3 取余时如果 x 为浮点型（不是整型），则必须使用强制转换（int）x。

运算符	解释	结合方式
() [] -> .	括号（函数 ），数组，两种结构成员访问	由左向右
! ~ ++ -- + - * & (类型) sizeof	否定，按位否定，增量，减量，正负号，间接，取地址，类型转换，求大小	由右向左
* / %	乘，除， 取余	由左向右
+ -	加，减	由左向右
<< >>	左移，右移	由左向右
< <= >= >	小于，小于等于，大于等于，大于	由左向右
== !=	等于，不等于	由左向右
&	按位与	由左向右
^	按位异或	由左向右
|	按位或	由左向右
&&	逻辑与	由左向右
||	逻辑或	由左向右
? :	条件	由右向左
= += -= *= /= &= ^= |= <<= >>=	各种赋值	由右向左
,	逗号（顺序）	由左向右

23、C 语言 9 种控制语句

①if( )~else　　条件语句

②for( )~　　循环语句

③while( )~　　循环语句

④do~while( )　　循环语句

⑤continue　　结束本次循环语句

⑥break　　中止执行 switch 或循环语句

⑦switch　　多分支选择语句

⑧goto　　转向语句

⑨return　　从函数返回语句

24、函数调用语句：由一个函数调用加一个分号构成。如：printf（“haha”）；

25、表达式语句：有一个表达式加一个分号构成。如：a = 3;（带分号的是语句，不带分号的是表达式）

26、空语句： 如　；（只有一个分号，什么也不做，流程从程序其他地方转到此处，也可用于循环语句中，但什么也不做）

27、复合语句：用{}把一些**语句和声明**括起来形成复合语句（又称语句块）。复合语句放在循环中，程序需要连续执行一组语句。

28、复制运算

例：a = 3

a **+=** 3 //等价于 a = a + 3

x ***=** y + 8 //等价于 x = x * （y + 8）

x **%=** 3 //等价于 x = x % 3

注意：上面红色（加粗）部分中间不能用空格隔开。

29、赋值过程中的类型转换

（1）赋值运算符两侧的类型一致，则直接进行赋值。如 int a = 3。

（2）赋值运算符两侧的类型不一致：

①将浮点型数据赋值给整型变量时，对浮点数取整（舍弃小数位），再赋给整型变量。

②将整型数据赋值给浮点型变量时，数值不变，但以浮点数形式存储在变量中。 如 23 转换成 23.0，后赋值给浮点型变量。

③将一个 double 型数据赋值给 float 型变量时，先将双精度转换为单精度，即只取 6-7 位有效数字。

④字符型数据赋给整型变量时，将字符的 ASCII 代码赋给整型变量。

⑤将占多字节的整型数据赋给占少字节的整型数据或字符变量时，只将其低字节原封不动地送到被赋值的变量（即发生“截断”）。

30、注意：要区分赋值表达式和赋值语句

赋值表达式末尾没有分号，而赋值语句末尾有分号。在一个表达式中可以包含一个或多个赋值表达式，但绝不能包含语句。

31、变量赋初值，如

①int 1 = 3; //指定 i 为整型变量，初值为 3

②int a, b, c = 5//指定 abc 为整型变量，但只有 c 初始化，初值为 5。

③int a = 3, b = 3, c = 3;不能写成 int a = b = c = 3;

32、从键盘上输入：scanf(“%d %d”,&a, &b);

33、有关输入输出的概念：

①键盘  计算机  显示器

输入 输出

②C 语言本身不提供输入输出语句，而是由 C 标准函数库中的函数来实现。

putchar(输出字符),getchar（输入字符），printf（格式输出），scanf（格式输入），puts（输出字符串），gets（输入字符串）。

34、在使用系统函数库时，要在程序文件的开头用预处理指令#include 把头文件放在本程序中。如：#include <stdio.h>

35、stdio.h 是 standard input & output（标准输入输出）的缩写，文件后缀“h”是 header 的缩写。

36、预处理指令：

①#include <stdio.h> //从 C 编译系统的子目录寻找

②#include “stdio.h” //先从用户的当前目录寻找

37、格式字符：

（1）d 格式字符。用来输出十进制整数。有以下几种用法：

%d，按整型数据的实际长度输出。

%md，m 为指定的输出字段的宽度（输出数据占 m 列）。

%ld，输出长整形数据。

（2）o 格式符，以八进制整形式输出整数。

（3）x 格式符，以十六进制数形式输出整数。

（4）u 格式符，用来输出 unsigned 型数据，即无符号数，以十进制形式输出。

（5）c 格式符，用来输出一个字符。

（6）s 格式符，用来输出一个字符串。

（7）f 格式符，用来输出实数（包括单双精度），以小数形式输出（%f 是 6 位小数，%lf 是 15 位小数）。

（8）e 格式符，以指数形式输出实数（默认情况下小数位 6 列，指数为 5 列）。

（9）g 格式符，用来输出实数，它根据数值的大小，自动选 f 格式或 e 格式（选择输出是占宽度较小的一种），且不输出无意义的零。

(10)%[scanfset]：

scanfset 有两种形式：一种是以非 “^” 字符开头的 scanset ，表示在读入字符串时将匹配所有在 scanfset 中出现的字符，遇到非 scanfset 中的字符时输入就结束；另外一种形式是以 “^” 字符开头的 scanfset ，表示在读入字符串时将匹配所有不在 scanfset 中出现的字符，遇到 scanfset 中的字符输入就结束。

38、f 格式符中

①指定宽度和小数位数，用%m.nf（输出数据占 m 列，n 位小数）

②输出数据向左对齐，用%-m.nf(作用与%m.nf 基本相同，但当数据长度不超过 m 时，数据向左靠，右端补空格。）

39、如果想输出字符“%”则应当 printf（“%f%%\n”，1.0／3）;输出结果是 0.333333%。

40、scanf 函数应注意的问题

①“&”不能丢，控制符后是变量地址，而非变量

②scanf（“a = %f, b = %f”，&a, &b）；输入时普通字符不能丢（%f 前有什么就得原封不动的输入什么），必须有 a = b =

③②中的逗号也可换成空格，但必须两个同时换，输入时必须要有空格（可以多个空格），总之前后对应。

④在输入数值数据时，如输入空格，回车，tab 键或遇非法字符（不属于数值的字符），认为该数据结束。如：

scanf（“%d, %c, %f”,&a, &b,%c）;若输入 1234a123o26↙，则会编译系统会自动识

别成 a=1234,b=123,c=26。

41、字符数据的输入输出

①putchar(输出) 格式：putchar ( ) //只能输出一个字符，想输出多个字符 就要用多个 putchar。

```
char a = 'A';
char b = 'B';
putchar(a);
putchar(b);
putchar('\n');
```

// （‘\￥’）输出￥，（‘\@’）输出@，（‘\101’）输出 101 对应的 ASCII 代码 A。

若 int a = 66；

putchar（a）； //会输出 33 所对应的 ASCII 代码 B。

putchar (c)中的 c 可以是字符常量、整型常量、字符变量或整型变量（其值在字符的 ASCII 代码范围内）。

②gettchar(输出) 格式： getchar ( ) //只能输入一个字符，想输入多个字符就要用多个 getchar。

例：

```
#include <stdio.h>
int main(void)
{
    char a, b, c;
    a = getchar();
    b = getchar();
    c = getchar();
    putchar(a);
    putchar(b);
    putchar(c);
    putchar('\n');
    return 0;
}
```

42、输出接收到的字符

①

```
#include <stdio.h>
int main(void)
{
    putchar(getchar());//将接收到的字符输出
    putchar(getchar());
    putchar(getchar());
    putchar('\n');
```

```
        return 0;
    }
```
②printf(“%c,getchar()”);

第四章　选择结构程序设计

1、选择语句：

①if 语句：用来实现两个分支的选择

②switch 语句：用来实现多分支的选择

2、整个 if 语句可以写在多行上，也可以写在一行上

如：if（x > 0）y = 1; else y = -1;注意 if 语句无论写几行，都是一个整体，属于同一个语句，不要误认为 if 部分是一个语句，else 部分是一个语句。else 语句不能作为独立语句使用，必须作为 if 语句的一部分，与 if 配对使用。

3、6 种关系运算符：

① < 小于

② <= 小于等于

③ > 大于

④ >= 大于等于

⑤ == 等于

⑥ != 不等于

4、关系表达式：用关系运算符将数值或数值表达式连接起来的式子

5、逻辑运算符：

① && 逻辑与　// 左右都为真，结果为真

② || 逻辑或　　// 左右有一个为真，结果为真

③ ！逻辑非　　// a 为真，则！a 为假

6、逻辑表达式：用逻辑运算符将数值或数值表达式连接起来的式子

7、优先级由高到低：非 > 算数 > 关系 > 与或 > 赋值

8、C 语言编译系统在表示逻辑运算结果是，1 代表真，0 代表假。

若：a =4,b=5, 则!a=0, a&&b=1

但在判断一个量是否为真时，以 0 代表假，非 0 代表真。

if(x = 0) 或 if(x != 0)

9、逻辑型变量：C99 所增加的一种类型数据。bool a，b；

10、max = (a>b) ? a : b;　// a>b? (max=a) :(max=b);

①等号右侧是条件表达式

②“？”和“：”是条件运算符，必须一起使用

11、条件表达式的一般形式：

条件 1？表达式 2：表达式 3

先求解条件 1，若非 0（真），则求解表达式 2,并将 2 的值便作为整个条件表达式的值。若 1 为 0（假），则求解 3 的值，并将 3 的值作为整个条件表达式的值。

12、switch（表达式）

```
{
  case 1：语句 1；break；
  case 2: 语句 2；break；
  case 3：语句 3；break；
  default: ......；break；
}
```

说明：

①switch 后面括号内的“表达式”，其值的类型应为整型（包括字符型）

②紧跟 case 后的是常量（常量表达式），只起标号作用

③执行 switch 语句时，先计算 switch 表达式的值再跟 case 后的常量标号比较，如果相同，则执行此 case 后面的语句，如果不同，则执行 default 后的语句。

④在各 case 语句后面都要有 break 语句。否则程序会继续执行下面的 case 语句。

第五章　循环结构程序设计

1、while 循环语句

①一般形式：while(表达式)

.....;(语句);

②先判断表达式，后执行语句。

③只要循环条件为真，就执行循环体语句。如 while（3）。

2、do.....while 循环语句

①一般形式：do

.....;(语句)

while（表达式）；（不要丢掉分号）

②先无条件执行循环体，然后再判断循环条件是否成立，若成立，则继续执行循环体。

3、for 循环语句

①一般形式：for（循环变量赋初值 1；循环条件 2；循环变量增量 4）

```
.........;(语句)3
for（i = 1;  i <= 100; i++)
    sum = sum + i;
```

②执行顺序：1 2 3 4 2 3 4 2 3 4.....

③表达式 1 可以省略，但表达式 1 后的分号不能省略，表达式 1 可以是与循环控制无关的任意表达式,此时在 for 前必须给循环变量赋初值。

④表达式 2 也可以省略，但表达式 2 后的分号不能省略，即不设置循环条件，此时程序将无休止的运行下去。

⑤表达式 3 也可以省略，但需要另外设法保证循环能够正常终止。

⑥表达式 1 或表达式 3 都可以是逗号表达式

如 for（i = 0,j = 100; i <= 100; i++, j--）

⑦for(i = 1;i <= 100; i += c)；此 for 语句的循环体为空语句，把本来要在循环体中执行的内容放在了表达式 3 中。

⑧ C99 允许在表达式 1 中定义变量并赋初值。for(int i = 1; i < 10; i++)

4、break:用于终止循环

①break 只能用于循环语句和 switch 之中，不能单独使用。

②break 不能直接用于 if，除非 if 属于循环内部的一个子句。

例子：for(i = 0; i < 3; i++)

```
{
 if(3 > 2)
    break; //当 3>2 为真时，break 终止的不是 if 语句，而是最
           接近的 for 语句。
 printf（“哈哈\n”）;//永远不会输出
 }
```

③在多层嵌套时，break 只能终止最近的那个循环或 switch 语句。

5、continue:用于跳过本次循环余下的语句

①例子：

```
for(1; 2; 3)
{
   A;
   B;
   continue; //C/D 语句都不会执行，但跳过 CD 后继续执行 3。
   C;
   D;
 }
```

第六章　利用数组处理批量数据

1、一维数组定义：int a[10]; //类型符 数组名[常量表达式]

①说明：“[]”内常量表达式用来表示元素的个数，即数组长度。例如 a[10] 表示 a 数组有 10 个元素，注意下表从 0 开始，即 a[0]开始，分别是 a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9]。

②常量表达式中可以包括常量和常量符号，如 int a[3+5]是合法的，但不能包含变量，如 int a[n]是不合法的。

③C 语言不允许对数组的大小作动态定义。例如：

```
int n;
scanf(“%d”, &n);
int a[n];          //这是错误的
```

2、数组的初始化：

①全部数组元素赋初值：int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9};//即 a[0] = 1, a[1] = 2, a[2] = 3, a[3] = 4, a[4] = 5, a[5] = 6, a[6] = 7, a[7] = 8, a[8] = 9, a[9] = 10.

②部分数组元素赋初值：int a[10] = {1,2,3,4,5};//1-5 赋值给 a[0]-a[4], a[5]-a[9]自动赋初值 0. int a[10] = 0;//数组元素全部赋初值 0. 字符型数组初始化为‘\0’.

③全部数组元素赋初值时，可以不用指定数组长度。如：

int [] = {1,2,3,4,5};//系统默认为 int a[5] = {1,2,3,4,5};

3、10 个数从小到大排序【冒泡排序法】：

```
#include <stdio.h>
int main()
{
    int i, j, t;
    int a[10];
    printf("请输入 10 个数: \n");
    for(i = 0; i < 10; i++)
        scanf("%d", &a[i]);  //一维数组的输入
    printf("\n");
    for(j = 0; j < 9; j++)
        for(i = 0; i < 9 - j; i++)
            if(a[i] > a[i + 1])
            {t = a[i]; a[i] = a[i + 1]; a[i +1] = t;}
    for(i = 0; i < 10; i++)
    {
        printf("%d", a[i]);  //一维数组的输出
        printf("\t");
    }
    printf("\n");
```

```
    return 0;
}
```

4、二维数组定义：float a[3][4]；//类型 数组名[][]3 行 4 列，不能写成 a[3,4]

5、二维数组的元素表示：

```
a[3][4] :
a[0][0]  a[0][1]  a[0][2]  a[0][3]
a[1][0]  a[1][1]  a[1][2]  a[1][3]
a[2][0]  a[2][1]  a[2][2]  a[2][3]
```

a[i][j]表示第 i+1 行，j+1 列的元素

6、二维数组的初始化

①全部元素赋初值：

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

②部分元素赋初值：

```
int a[3][4] = {{1},{5},{9}};
```

初始化后各数组元素分别为：

```
1  0  0  0
5  0  0  0
9  0  0  0
```

```
int a[3][4] = {{1},{0,6},{0,0,11}};
```

初始化后各数组元素分别为：

```
1  0  0  0
0  6  0  0
0  0  11 0
```

```
int a[3][4] = {{1},{},{1,3}};
```

初始化后各数组元素分别为：

```
1  0  0  0
0  0  0  0
1  3  0  0
```

③

```
int a[][4] = {1,2,3,4,5,6,7,8,9,10,11,12};//编译系统默认成 3 行。
```

```
int a[][4] = {{1},{},{1,3}};
```

初始化后各数组元素分别为：

```
1  0  0  0
0  0  0  0
1  3  0  0
```

7、一维数组的输入输出：

```
int a[10];
for(i = 0; i < 10; i++)
    scanf("%d", &a[i]);  //一维数组的输入
for(i = 0; i < 10; i++)
    printf("%d", a[i]);  //一维数组的输出
```

8、二维数组的输入输出：

```
int a[3][4];
for(i = 0; i < 3; i++)
    for(j = 0; j < 4; j++)
        scanf("%d", &a[i][j]);  //二维数组的输入
  for(i = 0; i < 3; i++)
     for(j = 0; j < 4; j++)
        printf("%d", a[i][j]);  //二维数组的输出
```

9、字符数组定义：char c[3] = {‘a’, ‘b’, ‘c’};

```
char c[] = {‘a’, ‘b’, ‘c’}; //长度自动定为 3
```

定义二维字符数组：

```
char z[2][3] = {{‘a’, ‘b’, ‘c’}, {‘d’, ‘e’, ‘f’}};
```

10、C 语言规定了一个“字符串结束标志”，以字符‘\0’作为结束标志。

11、char c[] = {“I am happy”};

也可以写成：char c[] = “I am happy”;//省去括号

也可以写成：char c[]={‘I’, ‘ ’, ‘a’, ‘m’, ‘ ’, ‘h’, ‘a’, ‘p’, ‘p’, ‘y’, ‘\0’};

此时数组 c 的长度不是 10，而是 11 个字节，因为字符串常量的最后由系统加上一个‘\0’.

12、字符数组的输入输出：

①逐个字符的输入输出用格式符“%c”

②整个字符串输入输出用格式符“%s”

③用“%s”格式符输出字符串时，printf 函数中的输出项是字符数组名，而不是数组元素名。如 11 中 printf（“%s”，c）；//不能写成（“%s”，c[10]）.

④如果一个字符数组中包含一个以上‘\0’时，则遇到第一个‘\0’输出结束。

⑤用 scanf 函数输入一个字符串：scanf（“%s”,c）；

⑥用 scanf 函数输入多个字符串：scanf（“%s %s %s”，c1，c2，c3）；

注：***字符数组输入不用地址符&***。数组之间用空格隔开。

⑦字符数组中未被赋值的元素的值置‘\0’.

⑧printf（“%d”，c）；//可以得到数组 c 的起始地址。

⑨printf（“%s”，c）；//按字符数组名 c 找到其数组起始地址，然后逐个输出其中的字符，知道遇到‘\0’为止。

13、puts——输出字符串的函数

①puts（str）；//输出字符数组 str.

②用 puts 函数输出的字符串中可以包括转义字符。例如

```
char str[] = {china\nbeijing};
puts(str);//输出结果是:
          china
          beijing
```

14、gets 函数——输入字符串的函数

①gets(str);//输入字符数组 str，str 已定义

15、用 puts 和 gets 函数只能输出或输入一个字符串，不能写成

puts(str1,str2);或 gets(str1,str2);

16、strcat 函数——字符串连接函数

①一般形式：stract（str1，str2）；//将字符串 2 接到 1 后面，结果放到字符数组 str1 中，函数调用后得到一个函数值——字符数组 1 的地址。

②字符数组 1 的长度必须足够大，以便容纳连接后的新字符串。

③两个字符串后面都有‘\0’,连接时将字符串 1 后面的‘\0’取消。

17、strcpy 和 strncpy 函数——字符复制函数

①字符串 2 复制到字符数组 1 中，例如：

```
char str1[10], str2[]=“China”;
strcpy(str1,str2);
```

②字符数组 1 的长度不应次小于字符数组 2 的长度。

③“字符数组 1”必须写成数组名形式（如 str1），“字符数组 2”可以是字符数组名形式，也可以是一个字符常量。例如：strcpy(str1,“China”);

④不能用赋值语句将一个字符串复制到另一个字符数组中去。用赋值语句只能将一个字符赋给一个字符型变量或字符数组元素。如下语句是不合法的：

```
str1 = “China”;   str1 = str2; //均不合法
```

⑤可以用 ctrncpy 函数将字符串 2 中前 n 个字符复制到字符数组 1 中去，例如：

```
strncpy（str1，str2，3）；//将 str2 前面 3 个字符复制到 str1 中，取代 str1 前 3 个字符（不包括‘\0’）。
```

18、strcmp 函数——字符串比较函数

①如果字符串 1 = 字符串 2，则函数值为 0.

②如果字符串 1 > 字符串 2，则函数值为一个正整数.

③如果字符串 1 < 字符串 2，则函数值为一个负整数。

19、注意：对两个字符串比较，不能用以下形式：

```
if（str1 > str2）
   printf(“yes”);
```

而只能用：

```
 if(strcmp(str1, str2) > 0)
    printf(“yes”);
```

20、strlen 函数——测字符串长度的函数

所得函数的值是字符串的实际长度（不包括‘\0’）.比如：

```
char str[10] = “China”;
printf(“%d”, strlen(str));//此时，输出结果不是 10，也不是 6，而是 5。
```

21、strlwr 函数——转换为小写的函数：将字符串中的大写字母转换为小写字母。

22、strupr 函数——转换为大写的函数：将字符串中的大写字母转换为小写字母。

23、注意：在使用字符串处理函数时，应当在程序文件的开头用

```
#include <string.h>  //把“string.h”文件包含到本文件中。
```

第七章　函数

1、为什么要用函数：
①避免重复操作
②便于实现模块化的程序设计
2、定义无参函数：
类型名　函数名（）
{
函数体
}
或　类型名　函数名（void）　//void 表示“空”，即函数没有参数
{
函数体
}
3、定义有参函数：
例　int max(int x, int y)
{
int z;
z = x > y ? x : y;　　//执行语句部分
return(z);
}
4、函数调用的一般形式：函数名（实参表列）
5、3 种函数调用方式：
①函数调用语句，把函数调用单独作为一个语句。
②函数表达式，函数调用出现在另一个表达式中。例如：m=max(a,b);
③函数参数，函数调用作为另一个函数调用时的实参。例如：
m=max(a,max(b,c));又如：printf（“%d”，max(a,b)）;
6、形式参数（形参）：定义函数时函数名后面括号内的变量名称为“形式参数”。
7、实际参数（实参）：主调函数中调用一个函数时，函数名后面括号内的参数称为“实际参数”。
8、实参和形参间的数据传递：在调用函数过程中，系统会把实参的值传递给被调函数的形参。或者说形参从实参中得到一个值。
9、例子：

```
//先编写 max 函数
int max(int x, int y)        //定义 max 函数，有两个形参
{
    int z;                   //定义临时变量 z
    z = x > y ? x : y;       //把 x，y 中较大值的赋给变量 z
    return(z);               //z 作为 max 函数的值带回 main 函数
}
```

```
//再编写主函数
#include <stdio.h>
int main()
{
    int max(int x, int y); //对max函数声明
    int a, b, c;
    printf("请输入两个整数:");
    scanf("%d %d", &a, &b);
    c = max(a, b);          //调用max函数，有两个实参，较大值赋给
                              变量c
    printf("较大值为: %d\n", c);
   return 0;
}
```

10、实参可以是常量、常量表达式，但它们必须有确定的值。在调用时将实参的值赋给形参。

11、实参与形参的类型应相同或赋值兼容。

12、定义函数中的形参在未出现函数调用时，它们并不占内存中的存储单元。 在发生函数调用时，函数max的形参被临时分配内存单元。实参和形参在内存中占有不同的存储单元,函数调用结束后，形参单元被释放。

13、函数的返回值是通过return语句获得的，return语句将被调用函数中的一个确定值带回到主调函数中去（供主调函数使用）。

14、return（z）=return z，return后可以是一个表达式，如return（x>y?x:y）

15、函数类型决定返回值的类型，如果函数值的类型和return语句中表达式的类型不一致，则以函数类型为准。

16、对于不带回值的函数，应当用定义函数为“void类型”（空类型）。此时在函数中不得出现return。

17、在一个函数中调用另一个函数（被调函数）需要具备如下条件:

①被调用的函数必须是已经定义的函数（库函数或自己定义的函数）。

②如果使用库函数，应该在本文件开头用#include指令将调用有关库函数时所需用到的信息“包含”到本文件中。如#include <stdio.h>.其中“stdio.h”是一个“头文件”。在stdio.h文件中包含了输入输出库函数的声明。如果不包含“stdio.h”文件中的信息，就无法使用输入输出库中的函数。同样，使用到数学库中的函数，应该用到#include<math.h>。h是文件所用的后缀，表示头文件。

③如果使用用户自己定义的函数，而该函数的位置在调用它的函数（主调函数）后面，应该在主调函数中对被调用的函数做声明。

18、函数声明比函数定义多一个分号，函数声明种形参可以省略，而只写形参类型（形参名可写可不写，写什么都无所谓）。如: float add（float， float）。

19、如果在文件的开头（所有函数之前），对本文件所调用的函数进行了声明（外部声明），则在个函数中不需要重复进行声明。

20、一个程序有且只能由一个主函数，主函数可以调用普通函数，普通函数不能调用主函数，普通函数之间可以相互调用。主函数是程序的入口也是程序的出口。

21、函数声明: 告诉编译器这些字母代表一个函数。

22、函数是 C 语言的基本单位，类是 JAVA、C#、C++的基本单位。

23、C 语言的函数是相互平行，相互独立的，在定义函数时，一个函数内不能再定义另一个函数，也就是不能嵌套定义，但可以嵌套调用函数。

24、函数嵌套举例：求 4 个整数的最大值。

```
#include <stdio.h>
int main()
{
   int max4(int, int, int, int);//对 max4 函数声明
   int a, b, c, d, e;
   printf("请输入四个整数: \n");
   scanf("%d %d %d %d", &a, &b, &c, &d);
   e = max4(a, b, c, d);
   printf("最大值为: %d\n", e);
   return 0;
}
int max4(int a, int b, int c, int d)//定义函数 max4
{
   int max2(int a, int b);//对 max2 声明
   int m;
   m = max2(a, b);
   m = max2(m, c);
   m = max2(m, d);
   return(m);
}
int max2(int a, int b)//定义 max2 函数
{
   int g;
   if(a > b)
      g = a;
   else
      g = b;
   return(g);
}
```

程序改进

```
#include <stdio.h>
int main()
{
   int max2(int a, int b);
   int a, b, c, d, e;
   printf("请输入 4 个整数: \n");
   scanf("%d %d %d %d", &a, &b, &c, &d);
   e = max2(max2(max2(a, b), c), d);
   printf("最大值为:%d\n", e);
}
```

```
int max2(int a, int b)
{
   return(a > b ? a : b);
}
```

24、在调用一个函数的过程中又出现直接或间接地调用到该函数本身，称为函数的递归调用。

25、数组元素可以用作函数实参，不能用作形参。

26、数组名也可以用作函数参数，包括实参和形参。

27、用数组元素作实参时，向形参变量传递的是数组元素的值，而用数组名作函数实参时，向形参（数组名或指针变量）传递的是数组首元素的地址。比如***数组 a[10]，在做实参时，应写 a，而不能写 a[10].***

28、主函数中的表达式语句用实参，函数定义中的表达式语句用形参（要写成两个不同的变量）

29、变量的作用域：变量的有效范围称为作用域。

30、按变量的作用域分：

①全局变量：在所有函数外部定义的变量。使用范围：从定义位置开始到整个程序结束。

②局部变量：在一个函数内部定义的变量或函数的形参。使用范围：只在本函数内部使用。

31、注意：

①在一个函数的内部，如果定义的局部变量的名字和全局变量相同时，局部变量会屏蔽掉全局变量。

②主函数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定义的变量。

③不同函数中可以使用同名变量，它们代表不同的对象，互不干扰。

32、建议不在必要时不要使用全局变量：

①全局变量在程序的全部执行过程中都占用存储单元，而不是仅仅在需要时才开辟单元。

②降低了函数的通用性。

③全局变量过多，会降低程序的清晰性。

33、变量的存储方式：

①静态存储：在程序运行期间由程序分配固定的存储空间。

②动态存储：在程序运行期间根据需要进行动态的分配存储空间。

34、在动态存储区存放一下数据：

①函数的形参。在函数调用时给形参分配存储空间

②函数定义中没有用关键字 static 声明的变量，即自动变量

③函数调用时的现场保护和返回地址等

35、在 C 语言中，每一个变量和函数都有两个属性：数据类型和数据的存储类型。

36、局部变量的存储类别

①自动变量（auto 变量）

函数中的局部变量，如果不专门声明为 static（静态）存储类别，都是动态地分配存储空间。在调用该函数时，系统会给这些变量分配存储空间，在函数调用结束时就自动释放这些存储空间。因此这类局部变量称为自动变量。自动变量用关键字 auto 作存储类别的声明。例如：

```
int f(int a)  //定义 f 函数，a 为形参
{
  auto int b, c = 3;  //定义 b，c 为自动变量
  .
  .
  .
 }
```

关键字 auto 可以省略，默认为“自动存储类别”。

②静态局部变量（static 局部变量）

在函数调用结束后局部变量的值不消失，而是保留上次函数调用结束后的值。

③寄存器变量（register 变量）

有一些变量使用频繁，为提高执行效率，允许将局部变量的值放在 CPU 中的寄存器中。这种变量叫做寄存器变量。

37、全局变量的存储类别：全都存放在静态存储区中。

38、在一个文件内扩展外部变量的作用域：关键字：extern（外部变量声明）

将外部变量的作用域扩展到其他文件：也用 extern 声明

将外部变量的作用域限制在本文件中：在外部变量前用 static 声明

39、内部函数和外部函数

①内部函数：如果一个函数只能被本文件中其他函数调用，则该函数称为内部函数。一般形式：static 类型名 函数名（形参）；如 static int f（）；

②外部函数：一般函数都是外部函数（除了内部函数），用 extern（或省略）。

第八章 指针

1、通过地址能找到所需要的变量单元，可以说，地址指向该变量单元。
2、一个变量的地址称为该变量的“指针”。
3、如果有一个变量专门用来存放另一变量的地址（即指针），则它称为“指针变量”。指针变量的值是地址（即指针）。
4、一个变量的指针的含义包含两个方面，一是以存储单元编号表示的地址（如编号为 2000 的字节），一是它指向的存储单元的数据类型（如 int，char，float）。
5、指向整型数据的指针类型表示为“int *”，读作“指向 int 的指针”或简称“int 指针”。
6、经典指针程序——互换两个数字

```
# include <stdio.h>

void huhuan_1(int , int);
void huhuan_2(int *, int *);
void huhuan_3(int *, int *);

int main(void)
{
    int a = 3;
    int b = 5;

    huhuan_3(&a, &b); //huhuan_2(*p, *q); 是错误的，huhuan_2(a, b);
                        也是错误的
    printf("a = %d, b = %d\n", a, b);

    return 0;
}

//不能完成互换功能
void huhuan_1(int a, int b)
{
    int t;

    t = a;
    a = b;
    b = t;
```

```
    return;
}

//不能完成互换功能
void huhuan_2(int * p, int * q)
{
    int * t;//如果要互换 p 和 q 的值, 则 t 必须是 int *,不能是 int,否则
            会出错

    t = p;
    p = q;
    q = t;
}

//可以完成互换功能
void huhuan_3(int * p, int * q)
{
    int t; //如果要互换*p 和*q 的值, 则 t 必须定义成 int,不能定义成 int
           *, 否则语法出错

    t = *p;  //p 是 int *,  *p 是 int
    *p = *q;
    *q = t;
}
```

7、函数的调用只可以得到一个返回值，而使用指针变量作参数，可以得到多个返回值。

8、数组名不代表整个数组，只代表数组首个元素的地址。

9、在指针指向数组元素时，可以对指针进行一下预算：

加一个整数 p+1:指向同一个数组的下一个元素(而非下一个字节，如果指针变量 p 定义时前面为 int 型，则每个元素占 4 个字节，假设 p 的值为 2000，则 p+1 的值为 2004，而不是 2001)

减一个整数 p-1:指向同一个数组的上一个元素

自加运算 P++

自减运算 P--

两个指针相减，（p1- p2）/每个元素所占的字节数 = 相差的元素个数

注意：指针运算没有加、乘、除运算。

10、通过指针引用一个数组元素的方法：

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 是指向数组元素的指针变量，其初值 p = a。

11、举例：

```
#include <stdio.h>
int main(void)
{
```

```
    int a[5];
    int i;
    int * p;
    for(i = 0; i < 5; ++i)     //根据（3）可以写成 for(p = a; p < (a + 5);
                                  p++)
                          //                    scanf("%d", p);
     scanf("%d", &a[i]);

    /*
  (1) 下标法
  for(i = 0; i < 5; ++i)
     printf("%d", a[i]);

  (2) 通过数组名计算数组元素地址，找出元素值
    for(i = 0; i < 5; ++i)
       printf("%d", *(a + i));
  */

   (3) 通过指针变量指向数组元素
    for(p = a; p < (a + 5); p++)
       printf("%d", *p);
     return 0;
  }
```

以上三种方法的比较：

①（1）（2）执行效率相同，（3）最快；C 语言编译系统是将 a[i]转换成*（a + i）处理的；a[i]与*(a + i)无条件等价

②（3）是指针变量直接指向元素，不必每次重新计算地址，有规律的改变地址值（p++）能大大提高执行效率。

12、假设 p = a（即 p = &a[0]），分析：

① p++;
*p; //使 p 指向下一元素 a[1],然后得到 a[1]的值

② *p++ ：++和*优先级相同，结合方向自右向左，则等价于*（p++），先实现*p 运算，再自增加 1

③ *（p++）与*（++p）不相同，前者是先进行*p 运算，得到 a[0]的值，再自增加 1，后者是先自增加 1，再取*p，得到的是 a[1]的值

④ ++（*p）：表示 p 所指向的元素值加 1，假设*p = a[0] = 3,则执行完结果为 4。注意是元素值加 1，而不是指针 p 加 1

13、实参数组名代表代表一个固定的地址，而形参数组名并不是一个固定地址，而是按指针变量处理。

14、二维数组 a[3][4]={{1,3,5,7},{9,11,13,15}{17,19,21,23}}的有关指针

表示形式	含义	地址
a	二维数组名，指向一维数组 a[0]，即 0 行首地址	2000
a[0],*(a+0),*a	0 行 0 列元素地址	2000

a+1,&a[1]	1 行首地址	2016
a[1],*(a+1)	1 行 0 列元素 a[1][0]的地址	2016
a[1]+2,*(a+1)+2,&a[1][2]	1 行 2 列元素 a[1][2]的地址	2024
(a[1]+2),(*(a+1)+2),a[1][2]	1 行 2 列元素 a[1][2]的值	13

15、在一维数组中*（a+i）为元素 a[i]的值，而二维数组中*（a+i）并不是元素的值，而是 i 行的首地址。（a+i）不指向指向具体单元，而是指向行。

16、指向 m 个整形元素的一维数组的指针变量：int (*p)[m] ={1,2,3...}//表示定义 p 为一个指针变量，它指向包含 m 个整型元素的一维数组。p 的类型不是 int *型，而是 int (*) [4] 型。p 被定义为指向一维数组的指针变量，一维数组有 4 个元素，因此 p 的基类型是一维数组，其长度是 16 字节。

17、指向数组元素的指针变量：

①一维数组的元素值输出

```
int a[5];
int * p;
for(p = a; p < a+5; p++)
printf(“%d”, *p);
```

②二维数组的元素值输出

```
int a[3][4];
int * p;
for(p = a[0]; p < a[0]+12; p++ )
printf(“%d”, *p);
```

18、分析一个小程序：

```
#include <stdio.h>
int main(void)
{
   int a[4] = {1,3,5,7};
    int (*p)[4];
    p = &a;    //不能写成 p=a; p=a 表示 p 的值是&a[0],指向 a[0].
                          p=&a 表示 p 指向一维数组（行）
    printf("%d\n", (*p)[3]); //输出结果：7
    return 0;
}
```

19、“int (*p)[4]”//p 的类型不是 int *型，而是 int （*）[4]型，p 被定义为指向一维整型数组的指针变量，一维数组有 4 个元素，因此 p 的基本类型是一维数组，其长度是 16 字节。

20、① char string[] = “I love China”;//定义字符数组

```
printf(“%s\n”, string);
```

② char * string = “I love China”;//定义字符指针变量

```
printf(“%s\n”, string);
```

21、通过字符数组或字符指针变量可以输出一个字符串，而对于一个数值型数组，是不能用数组名输出它全部的元素。例：int a[3];printf(“%d”,a)此时只输出 a[0]的值。

22、使用字符指针变量和字符数组的比较

①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（字符串第 1 个字符的地址），绝不是将字符串放到字符指针变量中。

②可以对字符指针变量赋值，但不能对数组名赋值（只可以对数组元素赋值）。

③初始化的含义不同：

（a）对字符指针变量初始化

```
char * a = “I love China”;等价于 char * a; a = “I love China”;
//把字符串的第一个元素的地址赋值给 a。
```

（b）对数组的初始化

```
char str[14] = “I love China”;
不等价于
char str[14];
str[] = “I love China”;//错误，数组可以在定义时对各元素赋初值，但不能用赋值语句对字符数组中全部元素整体赋值。
```

④编译时，字符数组分配若干个存储单元，以存放各元素的值，而对字符指针变量，只分配一个存储单元（VC++为指针变量分配 4 个字节）

如果定义了字符数组，但未对它赋值，可以引用（如输出）这些值，结果显然是无意义的，但不会造成严重的后果，容易发现和改正。

如果定义了字符指针变量，应当及时把一个字符变量的地址的地址赋给它，如果为对它赋予一个地址值，它并未具体指向一个确定的对象。此时如果向该指针变量所指的对象输入数据，可能出现严重的后果。因为指针可能指向已存放数据的有用内存，此时输入数据的话可能会破坏程序或有用数据，造成严重后果。

⑤指针变量的值是可以改变，而数组名代表一个固定值，不能改变。

⑥字符数组中各元素的值可以改变，但字符指针变量指向的字符串中的内容不可以改变。

⑦都可以用下标法或地址法引用数组元素（如 a[5],*(a+5);p[5],*(p+5)）

⑧用指针变量指向一个格式字符串，可以用它代替 printf 函数中的格式字符串。例如：

```
char * format;
format = “a = %d, b = %d\n”;
print(format, a, b); //等价于 printf（“a = %d, b = %d”, a, b）;
```

因此，只要改变 format 所指向的字符串，就可以改变输出格式。这种 printf 函数称为可变格式输出函数。也可以用字符数组实现，但在定义数组时就必须初始化（参考③）。

23、什么是函数指针：如果在一个程序中定义的了一个函数，在编译时，编译系统会为函数代码分配一段存储空间，这段存储空间的起始地址（又称入口地址）称为这个函数的指针。

24、如果向调用一个函数，除了可以通过函数名调用以外，还可以通过指向函数的指针变量来调用该函数。

25、用函数指针变量调用函数：

```
#include <stdio.h>

int main(void)
```

```
{
    int max(int, int);              //函数声明
    int (*p)(int, int);             //定义指向函数的指针变量 p
    int a, b, c;
    p = max;                        //使 p 指向函数 max
    printf("请输入 a 和 b 的值: ");
    scanf("%d %d", &a, &b);
    c = (*p)(a, b);                 //通过指针变量调用 max 函数
    printf("max = %d\n", c);
    return 0;
}

int max(int x, int y)               //定义 max 函数
{
   int z;
   if(x > y)
      z = x;
   else
      z = y;
   return(z);
}
```

分析：max 函数部分与通过函数名调用函数完全相同。

int (*p)(int, int); //用来定义 p 是一个指向函数的指针变量，最前面的 int 表示这个函数值（即函数返回值）是整形的。括号内的两个 int 表示函数有两个 int 型参数。*p 两侧的括号不可省略，表示 p 先与*结合，是指针变量，然后再与后面的（）结合，（）表示是函数，即该指针变量不是指向一般的变量，而是指向函数。

p = max; //作用是将函数 max 的入口地址赋给指针变量 p。

c = (*p)(a, b) 与 c = max(a, b) 是等价的。

①在一个程序中，一个指针变量可以先后指向同类型的不同函数。

②如果要用指针调用函数，必须先使指针指向该函数。如：p = max;

③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;

不能写成 p = max(a, b); //如果这样写，就成了先调用 max 函数，再将函数值赋给 p，而不是将函数的入口赋给 p

④调用函数时，只需将（*p）代替函数名即可。

c = (*p)(a, b) 中的（*p）代替 c = max(a, b)中的 max。

⑤对函数的指针变量不能进行算数运算，如 p+n,p++等运算毫无意义。

⑥用函数名调用函数只能调用所指定的一个函数，而通过指针变量调用函数可以根据需求先后调用不同的函数。

26、用指向函数的指针作函数参数：指向函数的指针可以作为函数参数，把函数的入口地址传递给形参，这样就能够在被调用的函数中使用实参函数。

27、举例：有两个整数 a 和 b，由用户输入 1,2 或 3，若输入 1，程序就输出 a 和 b 中的较大者，输入 2，就输出 a 和 b 中的较小者，输入 3，则输出 a、b 之和。

```
#include <stdio.h>

int main(void)
{
                        int fun(int x, int y, int (*p)(int, int));
                        int max(int, int);
                        int min(int, int);
                        int add(int, int);
                        printf("a = 30 b = 27");
                        int a = 30, b = 27;
    printf("\n");
                        printf("请选择 1,2 或 3: \n");
                        int n;
                        scanf("%d", &n);
                        if(n == 1)
                          fun(a, b, max);
                        else if(n == 2)
                          fun(a, b, min);
                        else if(n == 3)
                          fun(a, b, add);
                        return 0;
}

int fun(int x, int y, int (*p)(int, int))
{
                        int result;
                        result = (*p)(x, y);
                        printf("%d\n", result);
                        return(result);
}

int max(int x, int y)
{
                        int z;
                        if(x > y)
                          z = x;
                        printf("max = ");
                        return(z);
}

int min(int x, int y)
{
                        int z;
                        if(x >y)
```

```
                    z = y;
                printf("min = ");
                return(z);
}

int add(int x, int y)
{
                int z;
                z = x + y;
                printf("add = ");
                return(z);
}
```

28、返回指针值的函数：一个函数可以返回一个整型值、字符值还可以返回指针型的数据，即地址。

29、有 a=3 个学生，每个学生有 b=4 门课程的成绩。要求输入学生的序列号后，能输出该学生的全部成绩。用指针函数来实现。

```
#include <stdio.h>

int main(void)
{
    float * search(float (*pointer)[4], int n);  //函数声明
    float score[][4] = {{60, 70, 80, 90}, {56, 89, 67, 88}, {34, 78,
    90, 66}};
    float * p;
    int k;  //学生序列号
    printf("请输入学生序列号：\n");
    scanf("%d", &k);
    printf("第%d 号学生的成绩分别是：\n", k);
    p = search(score, k);  //调用函数
    int i;
    for(i = 0; i < 4; i++)
         printf("%5.2f\t", *(p + i));
    printf("\n");
    return 0;
 }

float *search(float (*pointer)[4], int n)
{
    float * pt;
    pt = *(pointer + n - 1);
    return(pt);
}
```

30、指针数组：一个数组的元素均为指针型数据的数组称为指针数组。

31、定义一个指针数组：int *p[4];//由于[]的优先级高于*，因此 p 先与[4]

结合，然后再与 p 前面的“*”结合。不能写成 int (*p)[4];//这是指向一维数组的指针变量。

32、指针数组比较适合用来指向若干个字符串，当然指针数组的元素也可以不是字符串。举个例子：将若干个字符串按字母顺序（从小到大）输出。

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    void sort(char * name[], int n);  // 函数声明，排序
    void print(char * name[], int n);  //函数声明，输出
    char * name[] = {"Follow me", "BASIC", "Great", "FORTRAN",
    "Computer"};
    //定义指针数组，它的元素分别指向 5 个字符串
    int n = 5;
    sort(name, n);  //调用 sort 函数
    print(name, n);  //调用 print 函数
    return 0;
}

void sort(char * name[], int n)
{
    char * temp;
    int i, j, k;
    for(i = 0; i < n - 1; i++)
        {
            k = i;
            for(j = i + 1; j < n; j++)
            if(strcmp(name[k], name[j]) > 0)
                k = j;
                if(k != i)
                 {temp = name[i]; name[i] = name[k]; name[k] = temp;}
         }
}

void print(char * name[], int n)
{
    int i;
    for(i = 0; i < n; i++)
    printf("%s\n", name[i]);  //按指针元素的顺序输出
}
```

33、指向指针数据的指针，简称指向指针的指针。例 32 中指针数组 name[] = {name[0], name[1], name[3], name[4]};可以设置一个指针变量 p，它指向指针数组的元素。定义：char ** p; p = name + i;//p 指向一个字符指

针变量（这个字符指针变量指向一个字符型数据）。*p 代表 p 所指向的字符指针变量的值（地址），**p 代表地址中字符型数据的值。

34、举个例子：

```
#include <stdio.h>

int main(void)
{
    char * name[] = {"ni hao", "wo bu hao", "bu hao la dao"};
    char ** p;
    int i;
    for(i = 0; i < 3; i++)
    {
        p = name + i; //p 指向 char*型数据的指针变量
        printf("%s\n", *p);
    }
}
```

注意：指针数组的元素也可以不指向字符串，而指向整形数据或实型数据等，例如：

```
#include <stdio.h>

int main(void)
{
    int a[5] = {1, 2, 3, 4, 5};
    int * num[5];//不能写成 int * num = {1, 2, 3, 4, 5}
    int ** p;    //指针数组元素只能存放地址
    int i;
    for(i = 0; i < 5; i++)
    {
        num[i] = &a[i];
        p = &num[i];
        printf("%d\n", **p);
    }
}
```

36、多种指针（二级）：

指针变量 2　　指针变量 1　　变量 a

指针变量 1 的地址 ——→ 变量 a 的地址 ——→ a 的值

37、指针数组作 main 函数的形参：？

38、怎样建立内存的动态分配：对内存的动态分配是通过系统提供的库函数来实现的，主要有 malloc，calloc，free，realloc 这 4 个函数。

39、malloc 函数

其函数原型为：void * malloc(unsigned int size);

其作用是在内存的动态存储区中分配一个长度为 size 的连续空间。形参 size 的类型为无符号整型（不允许为负数）。此函数的值（返回值）是所分配区域的第一个字节的地址，或者说，此函数是一个指针型函数，返回的

指针指向该分配区域的开头位置。
malloc（100）；//开辟100字节的临时分空间，其值为第一个字节的地址。
注意指针的基本类型为void，即不指向任何类型的数据，只提供一个地址。
如果此函数未能成功执行（例如内存空间不足），则返回空指针（NULL）。

40、calloc函数
其函数原型为：void * calloc(unsigned n, unsigned size);
其作用是在内存的动态存储区中分配n个长度为size的连续空间，这个空间一般比较大，足以保存一个数组。
用calloc函数可以为以为数组开辟动态存储空间，n为数组元素个数，每个元素长度为size。这就是动态数组。函数返回指向所分配的起始位置的指针；如果分配不成功，返回NULL。如：
p = calloc(50, 4)//开辟50×4个字节的临时分配域，起始地址赋给变量p

41、free函数
其函数原型为：void free（void * p）；
其作用是释放指针变量p所指向的动态空间，p为最近次调用calloc或malloc函数时得到的函数返回值。
free（p）；//释放指针变量p所指向的已分配的动态空间
free函数无返回值。

42、realloc函数
其函数原型为：void * realloc(void * p, unsigned int size);
如果已通过malloc函数或calloc函数获得了动态空间，可以通过realloc函数重新分配此动态空间的大小。
用realloc函数将p所指向的动态空空间的大小改变为size。p的值不变。如果分配不成功，则返回NULL。如：
realloc（p, 50）;//将p所指向的已分配的动态空间改为50字节。

43、以上4个函数的声明在stdlib.h头文件中，使用时应当“#include <stdlib.h>”指令把stdlib.h头文件包含到程序文件中。

44、以前malloc和calloc函数得到的返回值可以是指向字符型数据的指针，原型为char * malloc(unsigned int size);//返回字符型数据的指针。
但是实际上这些空间不一定是用来存放字符的，如果把开辟的区域用来存放整数，则要进行类型转换，如：
int * pt;
pt = (int *)malloc(100)；//将指向字符数据的指针转换为指向整型数据的指针。说明：类型转换只是产生了一个临时的中间值赋给了pt，但没有改变malloc函数本身的类型。

45、C99标准把以上malloc,calloc,realloc函数的基本类型定义为void型，这种指针称为无类型指针，即不指向哪一种具体的类型数据，只表示用来指向一个抽象的类型的数据，即仅提供一个纯地址，而不指向任何具体的对象。

46、C99中允许使用基本类型为void的指针类型。可以定义一个基本类型为void的指针变量（即void *型变量），它不指向任何类型的数据。注意：不要把“指向void类型”理解为指向“任何类型”的数据，而应该理解为“指向空型”或“不指向确定的类型”的数据。在将它赋值给另一个指针变量时由系统对它进行类型转换，使之适合于被赋值的变量类型。例如：
#include <stdio.h>

```
int main(void)
{
    int a = 3;  //定义 a 为整型变量
    int * p1 = &a;  //p1 指向 int 型变量
    char * p2;  //p2 指向 char 变量
    void * p3;  //p3 为无类型变量（void 型）
    p3 = (void *)p1;  //将 p1 的值转化为 void *类型，然后再赋值给 p3.
    p2 = (char *)p3;  //将 p3 的值转化为 char *类型，然后再赋值给 p2.
    printf("%d\n", *p1);//printf("%d\n", *p3); //错误，p3 是无指向
                                的，不能指向 a.
    return 0;
}
```

47、void * p3; p3 = &a; 编译系统会自动编译，不需用户进行强制转换，相当于 p3 = (void *)&a;，赋值后 p3 得到 a 的纯地址，但并不指向 a，不能通过*p 输出 a 的值。

48、sizeof（int）；sizeof 运算符测定本系统中 int 所分配的字节数。

49、指针变量的类型及含义：

变量定义	类型表示	含义
int i;	int	定义整形变量 i
int * p;	int *	定义 p 为整型数据的指针变量
int a[5];	int [5]	定义整型数组 a，它有 5 个元素
int * p[4];	int * [4]	定义指针数组 p，它由 4 个指向整型数据的指针元素组成
int (*p)[4];	int (*)[4]	p 为指向包含 4 个整型元素的一维数组的指针变量
int f();	int	f 为返回整型函数值的函数
int * p();	int * ()	p 为返回一个指针的函数，该指针指向整型数据
int (*p)();	int (*)()	p 为指向函数的指针，该函数返回一个整型值
int ** p;	int **	p 是一个指针变量，它指向一个指向整型数据的指针变量
void * p;	void *	p 是一个指针变量,基本类型为 void(空类型)，不指向具体对象

50、指针变量可以有空值，即该指针不指向任何变量，可以这样表示：p = NULL;其中 NULL 是一个符号常量，代表整数 0.在 stdio.h 头文件中对 NULL 进行了定义：#define NULL 0 //它指向一个地址为 0 的单元。系统保证该单元不做它用（不存放有效数据）。

51、注意：p 的值为 NULL 和未对 p 赋值是两个概念，前者是有值的（值为 0），不指向任何变量，后者虽未对 p 赋值但并不等于 p 无值，p 是一个无法预料的值，也就是 p 指向一个未知的单元，这种情况很危险。因此，在引用指针变量之前要对它进行赋值。

第九章　用户建立自己的数据类型

1、结构体：C 语言允许用户自己建立由不同类型数据组成的组合型的数据结构，它成为结构体。

2、定义结构体的 3 种方式，推荐使用第一种。

```
//第一种
struct student1  //struct student1 是变量类型，student1 是结构体名
{                    下面出现的 st2，st3 均为结构体变量。
     char name[20];
     int age;
     float score;
};//分号不能丢
/*第二种
struct student2
{
     char name[20];
     int age;
     float score;
} st2;
*/
/*第三种
strcut
{
     char name[20];
     int age;
     float score;
} st3;
*/
```

3、结构体变量的初始化和引用（把一个学生的信息放在结构体变量中然后输出）

```
#include <stdio.h>
int main(void)
{
     struct student
     {
          long int num;   //学号
          char name[20];  //姓名
          char sex;       //性别
          char addr[20];  //住址
```

```
    } a = {10023, "小明", 'M', "家园中路 28 号"};//定义结构体变量 a
                                                  并初始化
    printf("学号: %ld 姓名: %s 性别: %c 住址: %s\n",a.num,a.name,a.sex,
           a.addr);
 return 0;
}
```

4、注意:

①不能企图输出结构体变量名来达到输出结构体变量所有成员的值。

printf(“%s\n”, a);//这是错误的

②如果成员本身又属于一个结构体类型, 则引用方式: a.name.x

③结构体变量的成员可以像普通变量一样进行各种运算。

④同类的结构体变量可以相互赋值: a1 = a2;但不能够进行加减乘除运算

(假设 a1 a2 已定义为同类型的结构体变量)

⑤结构体变量的地址主要作函数参数, 传递结构体变量的地址。

5、定义结构体数组 (3 名学生得分并输出)

```
struct student
    {
        char name[20];  //姓名
        int score;    //得分
    } a[3] = {"小明", 98, “小李”, 96, “老王”, 67};
```

引用结构体数组 (输出)

```
for(i = 0; i < 3; ++i)
    printf("%s %d\n", a[i].name, a[i].score);
```

6、指向结构体变量的指针 (结构体指针)

举个例子:

```
#include <stdio.h>
struct student
{
   char name[20];
   float score;
};

int main()
{
   struct student st1 = {"小明", 93.1};//定义结构体变量的同时赋初值
   struct student * p;
   p = &st1;  //p 指向结构体变量
   printf("%s\t%5.2f\n", st1.name, st1.score);
   printf("%s\t%5.2f\n", (*p).name,    (*p).score);
   printf("%s\t%5.2f\n", p->name, p->score);

   return 0;
}
//st.name = (*p).name = p->name
```

7、指向结构体数组的指针

举个例子：

```
#include <stdio.h>

struct student
{
    char name[20];
    float score;
};

int main()
{
    struct student st1[2] = {"小明", 96.5, "小李", 98.5};
    struct student * p;
    for(p = st1; p < st1+2; ++p)
        printf("%s\t%5.2f\n", p->name, p->score);
    return 0;
}
```

8、链表：链表是一种常见的数据结构，它是动态地进行存储分配的一种结构。

9、有关链表的知识点：

①每个节点都包括两个部分：用户需要的实际数据和下一个节点的地址

②首节点：存放第一个有效数据的节点

③头结点：头结点的数据类型和首节点的数据类型一模一样

④头结点是首节点前面的那个节点，头结点并不存放有效数据

10、建立一个由 abc 三个学生数据组成节点的简单的静态链表，输出各节点数据。

```
#include <stdio.h>
#include <string.h>

struct student //声明结构体类型 struct student
{
   char name[20];
   float score;
   struct student * next;
};//分号不能丢

int main()
{
   struct student a, b, c, * head, * p;//定义 3 个结构体变量，a,b,c
                                       作为链表的节点
   head = &a;  //将 a 的地址给 head
   strcpy(a.name, "小明");  //以下给对各节点中的成员赋值
   a.score = 75.5;
   a.next = &b;
   strcpy(b.name, "小李");  //字符串的赋值
```

```
    b.score = 80;
    b.next = &c;
    strcpy(c.name, "张三");
    c.score = 90;
    c.next = NULL;
    head = &a;
    p = head;  //使 p 也指向 a 节点
    do
    {
       printf("%s\t%5.2f\n", p->name, p->score); //输出 p 所指向的节
                                                    点的数据
       p = p->next;  //使 p 指向下一个节点
    }
    while(p != NULL);

  return 0;
  }
```

11、建立一个由 abc 三个学生数据组成节点的简单的动态链表

将 abc 定义成 struct student *型

使用 malloc 函数，返回 void *型并赋值给 abc。

12、使几个不同的变量共享同一段内存的结构称为“共用体”类型的数据结构

13、“共用体”与“结构体”的定义类型相似，只将 struct 改为 union 即可，但他们的含义是不同的。结构体变量所占内存长度是各成员占内存的长度之和，每个成员分别占有自己的内存单元，而共用体变量所占的内存长度等于最长的成员的长度。

14、共用体类型数据的特点：

①共用体变量中虽然有多个成员，但每一瞬间只能存放一个值。

②可以对共用体变量初始化，但初始化表中只能有一个常量

```
  union date
  {
     int i;
     char ch;
     float f;
   } a = {1, ‘a’, 1.5};//错误
   union date a = {16};//正确，对第一个成员初始化
   union date a = {a.ch = ‘j’};//C99 允许对指定的一个成员初始化
```

③共用体变量中起作用的是最后一个赋值的成员（前面成员赋值被覆盖）

④共用体变量和它每个成员的地址都是同一地址。

⑤共用体的赋值规则与结构体相同

15、枚举：把一个变量的可能取值一一列举出来。

```
enum Weekday{sun, mon, tue, thu, fri, sat};//声明枚举类型
                                                enum Weekday
enum Weekday workday, weekday;//定义枚举变量 workday 和 weekday
```

sun, mon...,sat 称为枚举元素或枚举常量。

声明枚举类型后要定义枚举变量，枚举元素只能是标识符，不能是数字。

16、只有声明枚举类型时才可对枚举元素赋整数初值，如不赋值则默认为0,1,2....其它情况下枚举元素是不可以赋值的。枚举变量可以赋值和引用。

17、除了可以使用C提供的标准类型名（如int,char,float,double和long等）和程序编写者自己声明的结构体、共用体、枚举类型外，还可以用typedef指定新的类型名来代替已有的类型名。

18、typedef的用法:

typedef float real;//指定用real为类型名，作用与float相同。

real i = 98.5;与int i = 98.5完全定价