**C语言谭浩强版笔记**

**第一章 程序设计和C语言**

1、计算机能直接识别和接受的二进制代码称为**机器指令**，机器指令的集合就是该计算机的**机器语言。**

2、语言的发展历史：①机器语言②符号语言③高级语言

3、高级语言的发展：①非结构化语言②结构化语言③面向对象的语言

4、C语言的祖先是BCPL语言

5、在字符串中的//和/\*都不作为注释的开始。而是作为字符串的一部分。

【但是在vc++6.0中//显示编译错误】

6、不要以为在max函数中求出最大值z后就会自动地作为函数值返回调用处，必须用return语句指定将哪个值作为函数值。也不要不加分析地在所有函数后面都写上return 0

7、一个程序由一个或多个源程序文件组成

8、全局声明：即在函数之外进行的数据声明，在函数外面声明的变量称为全局变量。例如把int a,b,sum;放到main函数的前面

9、函数是C程序的主要组成部分，编写C程序的工作主要就是编写一个个函数

10、一个C语言程序是由一个或多个函数组成的，其中必须有且只有一个main函数

11、一个小程序只包含一个源程序文件，在一个源程序文件中包含若干个函数（其中一个为main函数），若程序规模太大，可以使一个程序包含若干个源程序文件，每个源程序文件又包含若干个函数【一个源程序文件就是一个程序模块，一个程序分成若干个程序模块】

12、在进行编译时是以源程序文件为对象进行的【分别对各源程序文件进行编译得到相应的目标程序，然后再将这些目标程序连接成为一个统一的二进制可执行程序】

13、C语言的这种特点使得容易实现程序的模块化

14、一个函数名后面必须跟一对圆括号，括号内写函数的参数名及其类型。如果函数没有参数，可以写void或空括号【如int main(void)或int main()】

15、void dump(){}它是一个空函数，但是是合法的

16、程序总是从mian函数开始执行的，不管位置在哪儿

17、程序中对计算机的操作是由函数中的C语句完成的

18、二进制目标程序在visual C++中的后缀为.obj

19、可执行程序在visual C++中的后缀为.exe

20、一个源程序经过编译后得到的目标程序要经过连接阶段与函数库进行连接才能生成可执行文件

21、程序设计的任务：①问题分析②设计算法③编写程序④对源程序进行编辑、编译和连接⑤运行程序，分析结果⑥编写程序文档

**第二章 算法**

1、对数据的描述：在程序中要指定用到哪些数据类型以及这些数据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**。

3、算法+数据结构=程序

4、算法是灵魂，数据结构是加工对象，语言是工具，编程需要采用合适的方法。

5、算法的概念：为解决一个问题而采取的方法和步骤。

6、计算机算法可分为两大类：数值运算算法和非数值运算算法【数值运算用于求数值解，非数值运算用于事务管理领域】

7、算法的特性：①有穷性②确定性③有零个或多个输入④有一个或多个输出⑤有效性

8、怎样表示一个算法：自然语言、传统流程图、结构化流程图、伪代码等

9、连接点就是将画在不同地方的流程图连起来，流程图别忘记画箭头。

基本结构的特点：①只有一个入口②只有一个出口③结构内的每一部分都有机会被执行到④结构内不存在死循环

10、区分当型和直到型：前者先判断条件成不成立，再执行下一步，后者先执行，再判断条件

11、用N-S图表示的算法都是结构化的算法，如果一个算法不能分解为若干个基本结构，则它必然不是一个结构化的算法【N-S图又称盒图】

12、结构化设计方法的基本思路：把一个复杂问题的求解过程分阶段进行，每个阶段处理的问题都控制在人们容易理解和处理的范围内。

13、方法：①自顶向下②逐步细化③模块化设计④结构化编码

14、程序中的子模块一般不超过50行

15、模块的独立性：使用一个模块完成一项功能，耦合性越少越好

16、结构化程序设计方法用来解决人脑思维能力的局限性和被处理问题的复杂性之间的矛盾

**第三章 最简单的C程序设计**

1、vc++把所有实数都作为双精度数处理。因此提醒用户：把双精度常量赋给float型变量会造成精度损失，知道怎么回事就行了

2、数据有两种表现形式：常量和变量

3、常量分为：①整型常量（1000）

②实型常量（十进制小数形式和指数形式【12.34E3，E后面必须是整数】）

③字符常量（普通字符、转义字符）

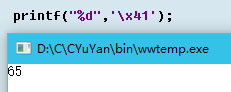
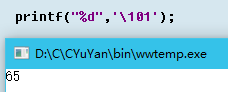
④字符串常量

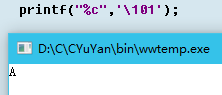
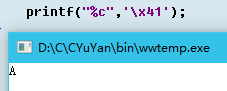
⑤符号常量(#define PI 3.1416)

4、基本字符集中的每个字符必须用一个字节表示，空字符也占一个字节，它的所有二进制位都是0

5、一个tab位置为8列

6、’\101’代表八进制数101的ASCII字符,即’A’(十进制65),’\x41’代表十六进制数41的ASCII字符,也是’A’,’\0’或者’\000’是代表ASCII码为0的控制字符,即空操作字符





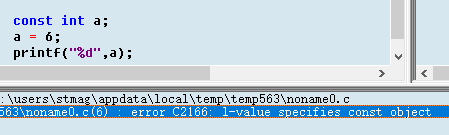
7、单撇号内只能包含一个字符,双撇号内可以包含一个字符串

8、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

9、变量必须先定义后使用

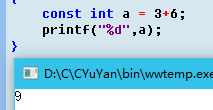
10、常变量const int a = 3，表示a被定义为一个整型变量，指定其值为3，而且在变量存在期间其值不能改变，常变量不能出现在赋值号的左边。例如上面一行不能写成：

const int a;

a=3; 

可以用表达式对常变量初始化，如

const int b=3+6, c=3\*cos(1.5);



11、标识符只能由字母、数字和下划线组成，且第一个字符必须是字母或下划线

12、①基本类型和枚举类型变量的值都是数值，统称为算数类型

②算数类型和指针类型统称为纯量类型（因为其变量的值是以数字来表示的）

③枚举类型是程序中用户定义的整数类型

④数组类型和结构体类型统称为组合类型（共用体类型不属于组合类型，因为在同一时间内只有一个成员具有值

13、TC2.0为每个整型数据分配2个字节（16个二进位）；VC++为每个整型数据分配4个字节（32位）

14、求负数的补码的方法：先将此数的绝对值写成二进制形式，然后对其后面所有各二进位按位取反，再加1

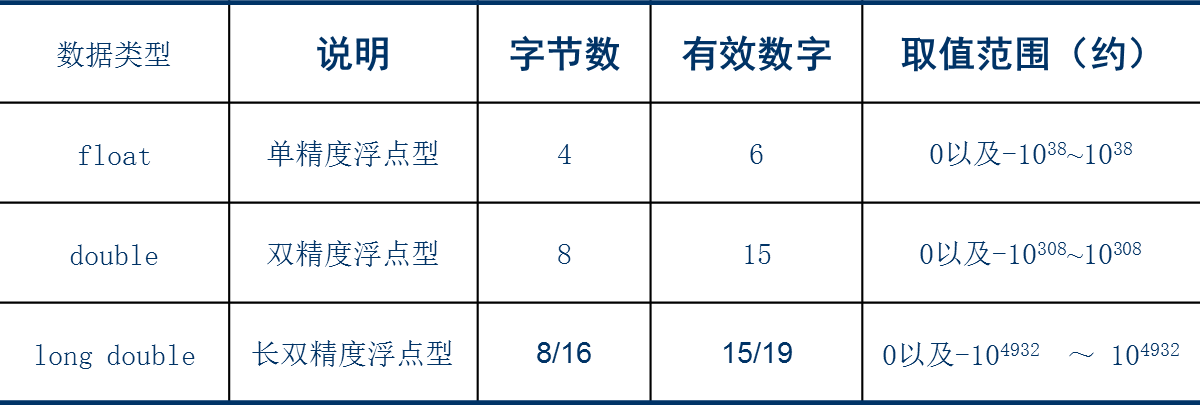
15、取值范围



16、如果既未指定为signed也未指定为unsigned类型，默认为有符号类型

17、只有整型（包括字符型）数据可以加signed或unsigned修饰符，实型数据不能加

18、实型数据取值范围：



19、在一个整数末尾加大写字母L或小写字母l，表示它是长整型

20、浮点型常量（小数或指数）在内存中都以指数形式储存

21、C程序中的实型常量都是双精度浮点型常量

float a = 3.14159；

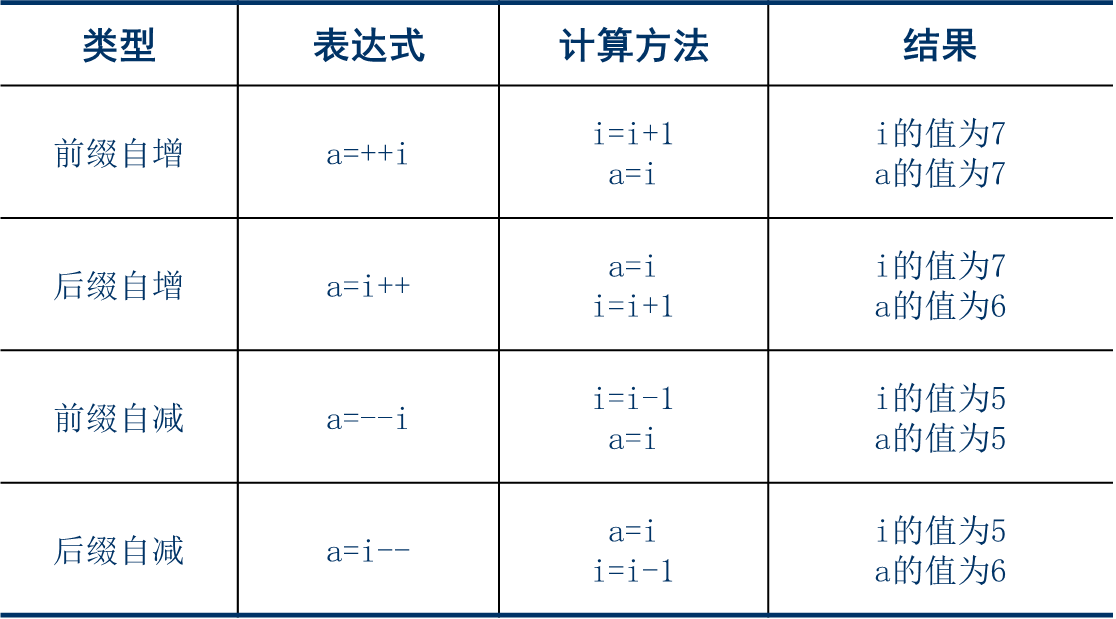
在进行编译时，对float变量分配4个字节，但对于浮点型常量3.14159，则按照双精度处理，分配8个字符

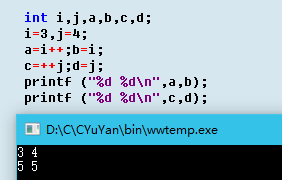
22、如果除数或被除数中有一个是负值，则舍入的方向不固定，一般采取“向零取整”的方法，即-5/3=-1

23、%运算符要求参加运算的对象是整数，除%以外的运算符的操作数都可以是任何算数类型

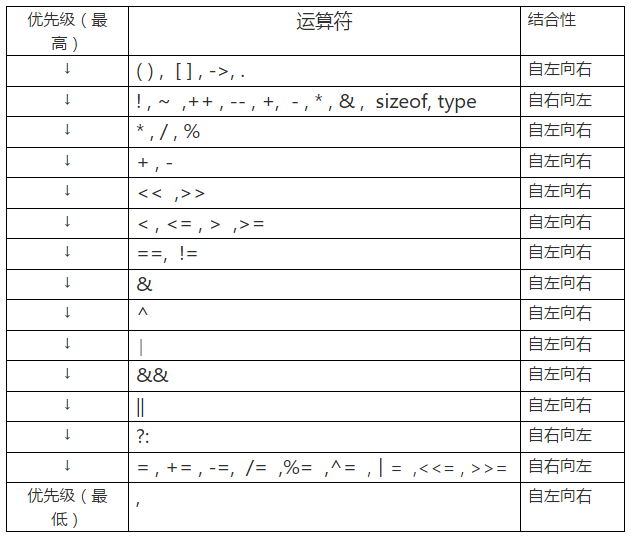
24、++i是先执行i=i+1后，再使用i的值；而i++是先使用i的值后，再执行i=i+1。

设已作声明：int i=6,a；





25、算数表达式和运算符的优先级与结合性



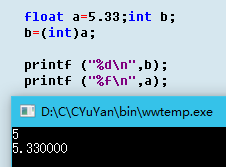
26、不同类型数据间的混合运算规律为：（取精度较高的）

①+，—，\*，/运算的两个数中有一个为float或double型，结果是double型，因为系统将所有float型数据都先转换为double型，然后进行运算。

②如果int型与float或double型数据进行运算，先把int型和float型数据转换为double型，然后进行运算，结果是double型。

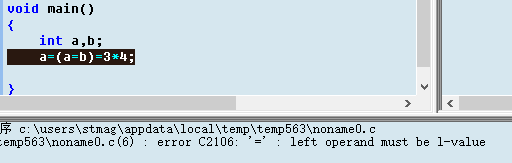
③字符（char）型数据与整型数据进行运算，就是把字符的ASCII代码与整型数据进行运算。字符型可以直接与整型数据进行运算。如果字符型数据与实型数据进行运算，则将字符的ASCII代码转换为double型数据，然后进行运算。

27、强制类型转换：一般形式为（类型名）（表达式），在强制类型转换时，得到一个所需类型的中间数据，而原来的变量的类型未发生变化



28、左值都可以作为右值，但算数表达式（a+b）或常量不能为左值

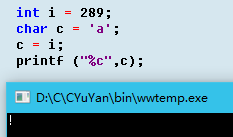
29、a=（a=b）=3\*4；//报错，左侧的操作数必须是左值



30、C语言的截断

i=289 0000 0001 0010 0001

c=33 0010 0001



31、在if条件中可以包含赋值表达式，但不能包含赋值语句

32、一般变量初始化不是在编译阶段完成的【***只有在静态储存变量和外部变量的初始化是在编译阶段完成的***】，而是在程序运行时执行本函数时赋予初值的，相当于执行一个赋值语句。

33、在scanf函数中，输入的数字可以用空格分开

34、在printf函数中，例如%7.2f代表在输出时，指定数据占7列，其中小数占2列

35、C语言本身不提供输入输出语句（printf和scanf不是C语言的关键字。而只是库函数的名字）

36、如果在源程序中有printf函数，在编译时并不把它翻译成目标指令，而是在连接阶段与系统函数库相连接后，在执行阶段中调用库函数中的printf函数

37、头文件<>称为标准模式，””编译系统先在用户的当前目录查找

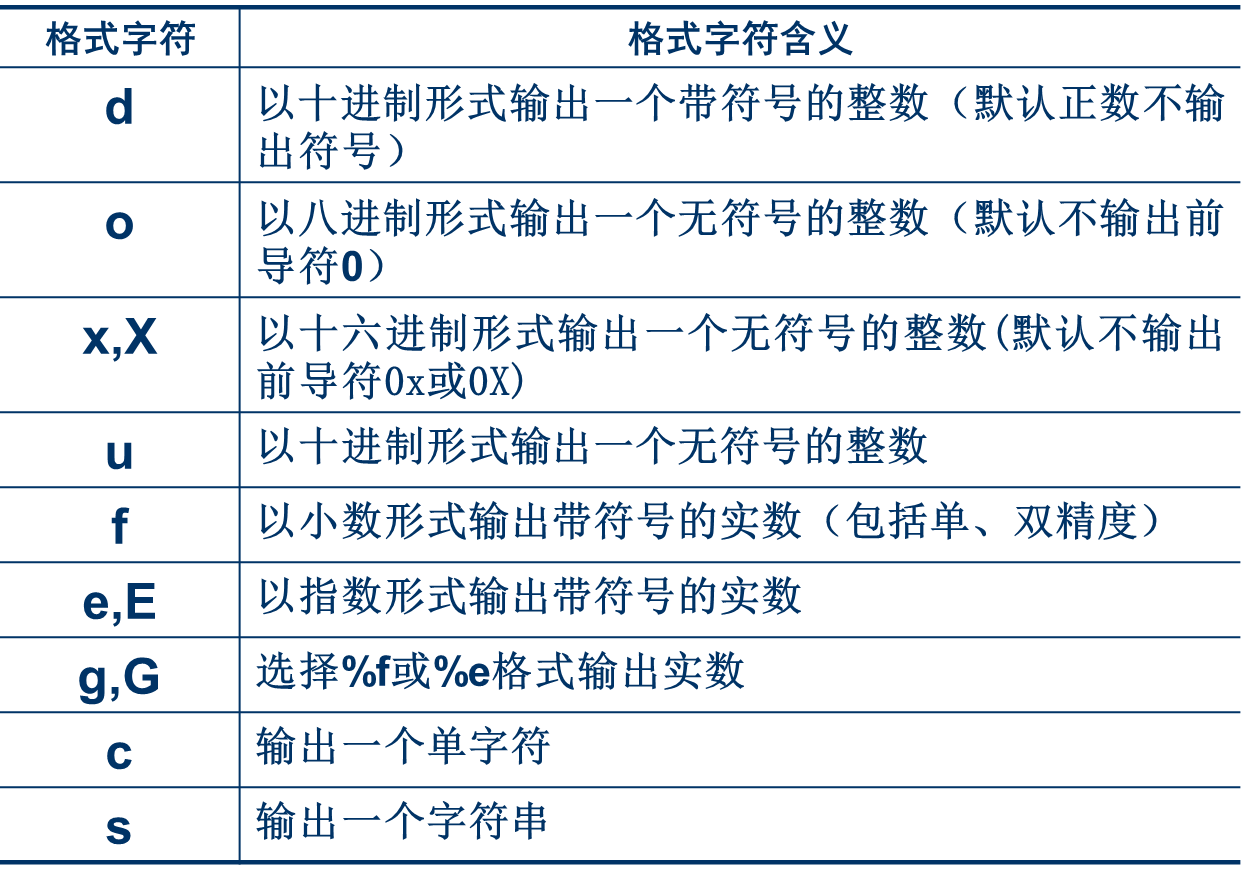
38、由于printf是函数，因此，“格式控制字符串”和“输出表列”实际上都是函数的参数

39、float型数据的存储单元只能保证6位有效数字，double是15位

40、%-m.nf，输出的数据向左对齐

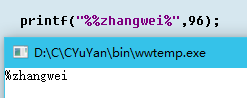
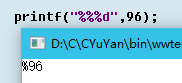
41、e格式符是以指数形式输出实数，默认小数部分6位，指数部分5列

42、格式字符含义（注意大小写）：



43、m.n中的n对实数表示输出n位小数，对字符串表示截取的字符个数

44、如果想输出字符%，在“”中用连续2个%%表示：



（经检测只输一个不显示）

45、在scanf函数中，如果两个%f间有2个空格，在输入时，两个数据之间应有2个或更多的空格

46、用%c格式声明输入字符时，空格字符和转义字符中的字符都作为有效字符输入

47、putchar函数是输出字符的函数，而不是整数，例如putchar（66）输出字符B，putchar可以是字符常量，字符变量，整型常量，整型变量（ASCII码范围内）

48、putchar（getchar（））；将接收到的字符输出

49、printf（“%c”，getchar（））；先从键盘输入一个字符，然后用%c的格式输出

**第四章 选择结构程序设计**

1、if复合语句应当用花括号括起来

2、所谓关系运算就是比较运算

3、关系运算符中等于与不等于优先级最低

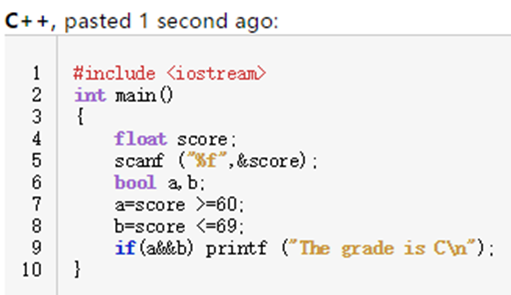
4、算术高于关系高于赋值，关系表达式的值是一个逻辑值，真或假【d=a>b；如果a大于b为真，则a>b的值为1，所以d为1】

5、逻辑运算符中，非运算符（！）和单目的算术运算符同级，高于双目的算术运算符，优先于关系运算符，优先于逻辑运算符与（&&）和或（||）。逻辑运算符中，非运算符（！）的结合性为右结合；与运算（&&）和或运算（||）的结合性为左结合。

6、0为假，非0为真

7、逻辑表达式注意“短路”现象

8、逻辑型变量（C99，参考）：在头文件stdbool.h中，将bool定义为\_Bool的同义词，同时定义了两个符号常量true代表1，false代表2（作用是将关系运算和逻辑运算的结果存到一个逻辑型变量中，以便于分析和运算，定义逻辑变量用类型符\_Bool）



9、条件运算符由？和：组成，是C语言中唯一的一个三目运算符

10、条件与是哪U老虎的优先级别比关系运算符和算术运算符低，所以括号可以不要。

例如：max=（a>b）？a：b+1；可以写成max=a>b？a：b+1；

11、if语句的嵌套：else总是与它上面的最近的未配对的if配对

12、switch语句中break语句的作用是使流程转到switch语句的末尾（即由花括号处）

13、switch语句：

①switch后面括号内的表达式，其值的类型应当是整数类型（包括字符型）

②case后面跟一个常量（或常量表达式）

③语句体内包含多个以关键字case开头的语句行和最多一个以default开头的行

④case出现次序不影响执行结果，可以先出现default标号

⑤在执行switch语句中，根据switch表达式的值找到匹配的入口标号，并不在此进行条件检查，*在执行完一个case标号后面的语句后，就从此标号开始执行下去，不再进行判断！*

⑥最后一个case子句可不加break

⑦多个case标号可以共用一组执行语句，例如：

case ‘a’:

case ‘b’:

case ‘c’:printf(“>60\n”)；break；

如果条件满足，不管是abc，都输出>60

**第五章 循环结构程序设计**

1、while语句中循环体只能是一个语句，特点是先判断条件表达式，然后执行循环体语句【while（表达式）语句】

2、do…while语句的特点是：先执行，再判断【do 语句 while（表达式）；】

3、循环语句中不要忘记给变量赋初始值

4、当while后面的表达式的第1次的值为“真”时，两种循环得到的结果相同；否则，两者结果不相同（指两者具有相同的循环体的情况）

5、for语句与while语句两者无条件等价

6、for语句若没有表达式2或表达式3，循环体会无止境地执行下去

7、表达式1可以是设置循环变量初值的赋值表达式，也可以是与循环变量无关的其他表达式for（sum=0；i<=100;i++）

8、表达式2可以是数值表达式或字符表达式，只有其值为非零，就执行循环体for (i=0;(c=getchar()!=’\n’;i+=c)

9、C99允许在for语句的“表达式1”中定义变量并赋初始值

10、凡用while循环能完成的，用for循环都能实现

11、while、do…while、for都可以用break语句跳出循环，用continue语句结束本次循环

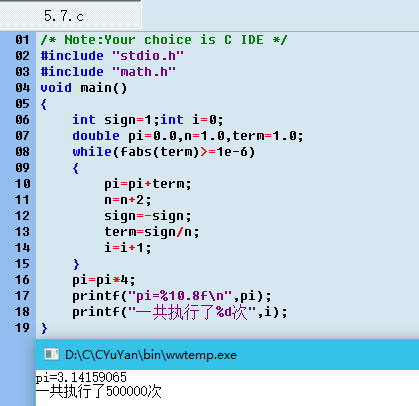
12、break语句的作用是使流程跳到循环体之外。但其只能用于循环语句和switch语句之中，而不能单独使用

13、continue语句的作用是结束本次循环，跳过循环体尚未执行的语句，转到循环体结束点之前，接着执行for语句中的表达式3，然后进行下一轮循环判定

14、如果是双重循环，里面有break或continue，结束的是内循环，而不是整个循环

15、在C库函数中，有两个求绝对值的函数：abs（x），求整数x的绝对值，结果是整型；fabs（x），x是双精度数，得到的结果是双精度型

16、例5.7扩展，如果求出循环体执行多少次：



17、字符串的结尾只能'是'\0','\n'是回车不能用来判断结束，一个字符串中可以有若干个回车

**第六章 利用数组处理批量数据**

1、数组是一组有序数据的集合，数组中的每一个元素都属于同一个数据类型

2、数组的下标是从0开始的，数组的常量表达式可以包括常量和符号常量

3、数组的大小不依赖于程序运行过程中变量的值

4、如果在被调用函数（不包括主函数）中定义数组，其长度可以是变量或非常量表达式void func（int c）{int a[2\*n];…}合法，n的值从实参传来，叫可变长数组。但是在执行函数时，n的值是不变的，数组的长度是固定的。如果指定数组为静态存储方式，不可用，如：static int a[2\*n]；

5、int a[10]；表示定义数组时指定数组包含10个元素

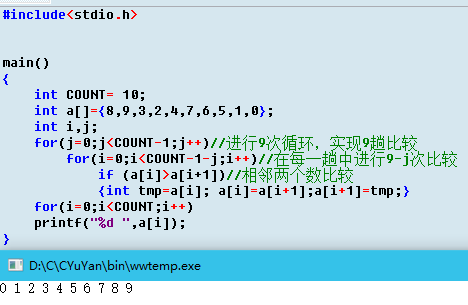
t=a[6];表示引用a数组中序号为6的元素

6、可以只给数组中的一部分元素赋值，系统自动给后5个元素赋初值为0

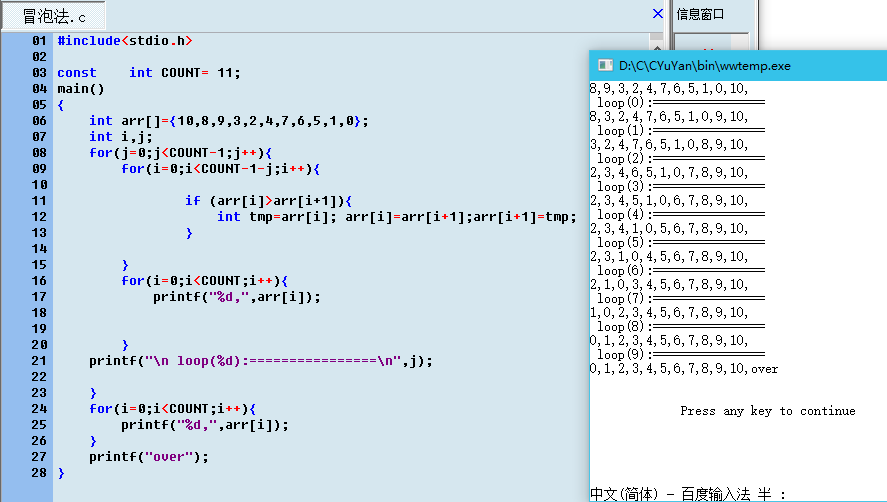
7、在对全部数组元素赋初值时，由于数据的个数已经确定，因此可以不指定数组长度。但如果数组长度与提供初值的个数不相同，则方括号中的数组长度不能省略

8、未被“初始化列表”指定初始化的数组元素，系统会自动把他们初始化为0（如果是字符型数组，则初始化为‘\0’，如果是指针型数组，则初始化为NULL，即空指针

9、冒泡法排序：如果有n个数，则要进行n-1趟比较。在第一趟比较中要进行n-1次两两比较，在第j趟比较中要进行n-j次两两比较



For for if for print【重点】



10、用矩阵形式表示二维数组，是逻辑上的概念，能形象地表示出行列关系。而在内存中，各元素是连续存放的，不是二维的，是线性的

11、注意区分定义数组和数组元素，后者行列序号从0起算

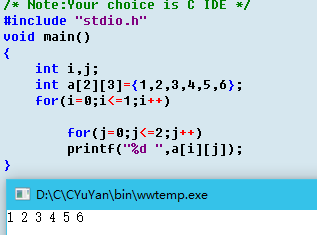
12、数组元素可以出现在表达式中，也可以被赋值：

b[1][2]=a[2][3]/2;

13、二维数据初始化，对中间行不赋初值的方法：

int a[3][4]={{1}，{}，{9}}；

14、如果对全部元素赋初值，则定义二维数组时对第1维的长度可以不指定，但第2维的长度必须要有，系统会自动判断多少行

15、在定义时也可以只对部分元素赋初值而省略第一维的长度，但应当分行赋初值

16、数组遍历：

17、C99标准中，把字符类型归纳为整型类型中的一种，C语言没有字符串类型，字符串是存放在字符型数组中的

18、字符数组中，初值个数大于数组长度，出现语法错误；初值个数小于数组长度，其余元素自动定义为空字符’\0’

19、’\0’作为字符串结束标志，在遇到’\0’时，表示字符串结束，把它前面的字符组成一个字符串

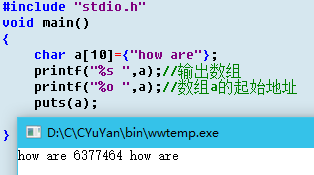
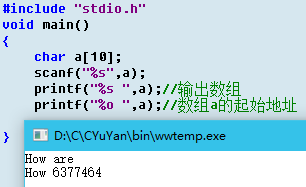
20、字符数组并不要求它的最后一个字符为’\0’，甚至可以不包括’\0’，系统会自动添加，但是加上便于测定字符串的实际长度

21、将整个字符串一次输入或输出，用%s格式符，当用%s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

22、如果一个字符数组包含多个’\0’，则遇到第一个结束

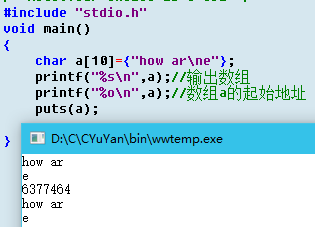
23、可以用scanf函数输入一个字符串scanf（“%s”，c），如果利用一个scanf函数输入多个字符串，以空格分隔【在C语言中，数组名代表该数组的起始地址，不要再加地址符&】

24、注意空格以及如何输出该数组的起始地址：（注意区分）



25、用puts函数输出可以包含转义字符：

（在输出时将字符串结束标志\0转换成\n，即输完字符串后换行）



26、gets函数返回的函数值是字符数组的起始地址

27、strcat函数：把字符串2接到字符串1的后面，结果放在字符数组1中，函数调用后得到的函数值是字符数组1的地址，要求字符数组1必须足够大，连接时字符串1后面的\0取消，只在新串最后保留\0 【格式：strcat（字符数组1，字符数组2）】

28、strcpy函数：作用是将字符串2复制到字符数组1中去，字符数组1必须写成数组名形式，字符串2 可以是字符数组名，也可以是一个字符串常量，如果复制前未对str1初始化，其内容无法预知。不能用赋值语句将一个字符串常量或字符数组直接给一个字符数组，但是可以将字符赋给一个字符型变量或字符数组元素

【格式：strcpy（字符数组1，字符串2）】

29、strncpy（str1，str2,2）；作用是将str2最前面2个字符复制到str1中，取代str1原有的最前面2个字符

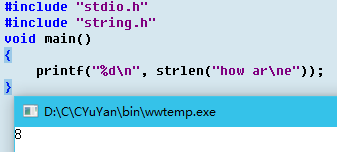
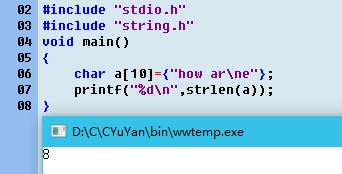
30、strcmp函数：作用是比较2个字符串，规则是自左向右逐个字符相比（按ASCII码大小），直到出现不同的字符或遇到\0为止；若字符都相同，则认为两个字符串相等，若不同，以第一对不相同的字符串比较结果为准。比较的结果由函数值带回，相等为0，字符串1大于字符串2为正整数，否则为负整数。

格式：格式：strcmp（字符串1，字符串2）

例：if（strcmp（str1，str2）>0）;

printf（“yes”）；

31、strlen函数：测字符串实际长度，不包括\0，也可以直接测试字符串常量的长度



32、strlwr函数：将字符串中大写字母转换为小写字母

strupr函数：将字符串中小写字母转换为大写字母

33、在使用字符串处理函数时，头文件应加入#include <string.h>

**第7章 用函数实现模块化程序设计**

1、一个C程序可由一个主函数和若干个其他函数构成，由主函数调用其他函数，其他函数也可互相调用，同一个函数可以被一个或多个函数调用任意多次

2、函数声明的作用是让系统中编译时知道它们是函数而不是变量或其他对象

3、一个C程序由一个或多个程序模块组成，每一个程序模块作为一个源程序文件。由若干个源程序文件组成一个C程序，一个源程序文件可以为多个C程序共用

4、一个源程序文件由一个或多个函数以及其他有关内容组成。一个源程序文件是一个编译单位，*在编译时以源程序文件为单位进行编译，而不是以函数为单位*

5、从main开始，从main结束

6、一个函数并不从属于另一个函数，函数不能嵌套定义（可以嵌套调用）。函数间可以互相调用，但不能调用主函数，mian函数是被操作系统调用的

7、无参函数可以带回或不带回函数值，但一般不带回，有参函数应定义为与返回值相同的类型

8、定义函数，函数体可以包括声明部分和语句部分

9、调用函数并不一定要求包括分号，只有作为函数调用语句才需要有分号。如果作为函数表达式或函数参数，函数调用本身是不必有分号的

10、*定义函数时函数名后面括号中的变量名称为形式参数；在主调函数中调用一个函数时，函数名后面括号中的参数称为实际参数，实参可以是常量、变量或表达式*

11、系统会把实参的值传递给被调用函数的形参

12、如果实参与形参类型不同，则按不同类型数值的赋值规则进行转换

13、在定义函数中指定的形参，在未出现函数调用时，他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

14、应当注意返回值的类型与函数类型一致

15、形参与实参是两个不同的存储单元

16、实参向形参的数据传递是“值传递”，单向传递，只能由实参传给形参，实参无法得到形参的值

17、函数的返回值是通过函数中return语句获得的

18、一个函数可以有一个已上的return语句，执行到哪一个return语句，哪一个return语句就起作用。return语句后面的括号可以不要，return语句后面的值可以是一个表达式

19、如果函数值的类型和return语句中的表达式不一致，则已函数类型为准（不指定函数类型，编译系统默认为int型）

20、对于不带回值的函数，应当用定义函数为“void类型”，或空类型

21、用户定义的函数在调用它的函数的后面（同一文件内），应该在主调函数中对被调用函数作声明

22、函数声明比函数定义中的首行多一个分号

23、函数声明中的形参名可以省写，而只写形参的类型。（函数定义应该不能省略）

24、如果在文件的开头，对本文件所调用函数进行了声明，则在各函数中不必对其所调用的函数再做声明

25、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

26、凡是变量可以出现的地方，都可以用数组元素代替

27、数组名可以作实参和形参，传递的是数组第一个元素的地址。数组元素可以用作函数的实参，不能用作形参。

28、数组元素采用的是值传递的方式，方向是从实参传到形参，单向传递

29、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

30、实参数组与形参数组类型应一致，如不一致，结果将出错

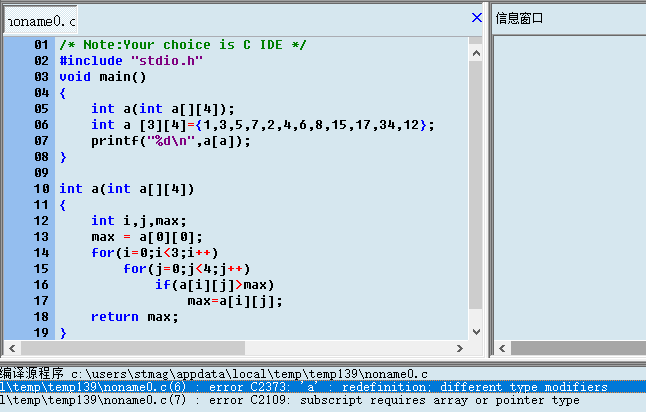
31、C语言编译系统并不检查形参数组的大小，只是将实参数组的首元素的地址传给形参数组名，这样两个数组就共占用一段内存单元

32、因此形参数组中各元素的值如果发生变化会使实参数组元素的值同时发生变化

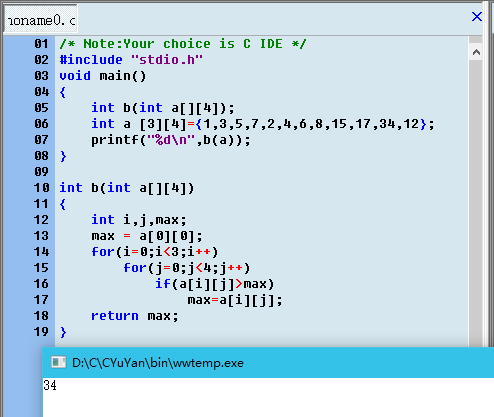
33、多维数组中，在被调用函数中对形参数组定义时可以指定每一维的大小，也可以省略第一维的大小说明

34、中第2维大小相同的前提下，形参数组的第1维可以与实参数组不同，C语言编译系统不检查第一维的大小

35、由于c语言不支持函数重载（同名），所以在定义和声明时，参数个数，类型，返回值类型不一样就会报这个错误C2373；全局变量和局部变量重名会报错



修改如下：



36、局部变量：在一个函数内部定义的变量只在本函数范围内有效，中复合语句内定义的变量只在本复合语句范围内有效

37、在x函数中定义了变量a，b；在y函数中定义了变量a，c。x函数的a与y函数的a不是同一个对象

38、主函数定义的变量也只在主函数中有效，主函数也不能使用其他函数中定义的变量

39、形参也是局部变量，例如x函数的形参a只在x函数中有效，其他函数可以调用x函数，但不能直接引用x函数的形参a，例如在其他函数中输出a的值是不行的

40、全局变量可以为本文件中其他函数所共用。*它的有效范围为从定义变量的位置开始到本源文件结束*

41、由于同一文件中的所有函数都能引用全部变量的值，因此如果在一个函数中改变了全局变量的值，就能影响到其他函数中全局变量的值（习惯：全局变量名的第一个字母用大写表示）

42、若外部变量与局部变量同名：



43、全局变量中程序的全部执行过程中都占用存储单元，而不是仅在需要时才开辟单元

44、全局变量全部存放在静态储存区中

45、如果在一个程序中两次调用同一函数，而在此函数中定义了局部变量，在两次调用时分配给这些局部变量的存储空间的地址可能是不相同的

46、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别。存储类别指的是数据在内存中存储的方式（如静态存储和动态存储）

47、函数中的形参和在函数中定义的局部变量（包括在复合语句中定义的局部变量），都属于自动变量（auto）

48、不写auto则隐含指定为自动存储类型

49、静态局部变量实在编译时赋初值的，只赋值一次，在程序运行时它已有初值，以后每次调用函数时不再重新赋初值而只保留上次函数调用结束时的值。自动变量赋初值，不是在编译时进行的，而是在函数调用时进行的，每调用一次函数重新给一次初值，相当于执行一次赋值语句

50、如果定义局部变量不赋值，对于静态局部变量来说，编译时自动赋值为0或\0，而对自动变量来说，值不确定

51、静态局部变量在函数调用结束后仍然存在，但只能被本函数引用

52、自动变量auto存储动态存储区；静态局部变量static存储在静态存储区；寄存器register存储在CPU中的寄存器

53、用extern声明外部变量时，类型名可以写，也可以不写

54、谨慎使用extern扩展全局变量的作用域，因为可能会改变该全局变量的值

55、在编译时extern时，先在本文件找，再在连接时去其他文件找

56、静态外部变量：用static声明，只能用于本文件的外部变量

57、static的区分：对局部变量来说，把它分配在静态储存区，该变量在整个程序执行期间不释放，其所分配的空间始终存在；对全局变量来说，则该变量的作用域只限于本文件模块

58、报错：重新定义【int a；static a；】

正确写法是【static int a】

59、从作用域角度分:局部变量，全局变量

从生存期区分：动态存储，静态存储

从变量值存放位置区分：内存中的静态存储区，内存中的动态存储区，CPU中的寄存器

60、变量着范围内有效→符合作用域范围→可见→可见性

变量在某一时刻存在→属于生存期→存在

61、凡有static声明的，其作用域都是局限的，或者是局限于本函数内（静态局部变量），或者局限于本文件内（静态外部变量）

62、把建立存储空间的声明称定义，把不需要建立存储空间的声明称为声明，对外部变量的初始化只能在定义时进行，而不能在声明中进行。结论：在函数中，出现对变量的声明（除extern函数）都是定义，在函数中对其他函数的声明不是函数的定义

63、如果不声明，一个文件中的函数既可以被本文件其他函数调用，也可以被其他文件中的函数调用

64、内部函数：一个函数只能被文件中的其他函数所调用

static int fun (int a,int b)

64、外部函数：可供其他文件调用

extern int fun (int a,int b)

C语言规定，如果定义函数时省略extern，默认为外部函数，之前都是外部函数

65、宏定义可以用宏名来表示一个字符串，在宏展开时用字符串取代宏名，行末不能加分号，如果加上也是字符串一部分

66、宏定义的作用域是所在位置开始到源程序结束，如果要终止用#undef命令

67、宏定义允许嵌套，如#define a 3 #define b a\*3

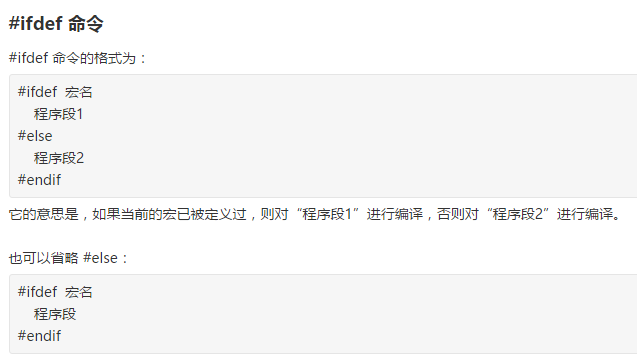
68、有参宏定义中，宏名和形参表之间不能有空格；带参宏调用中，只是符号替换，不存在值传递问题；宏定义中的形参只能是标识符，而宏调用中的实参可以是表达式，形参要用括号括起来

69、编译预处理指令包含：文件包含，宏定义，条件编译

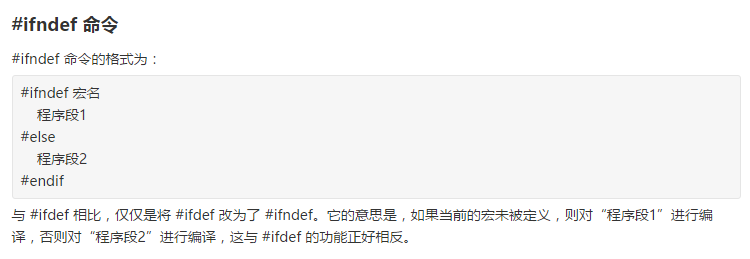
70、条件编译：

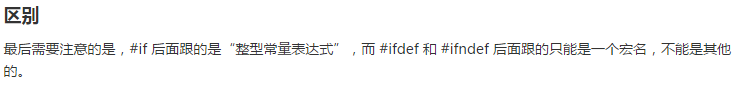
（也不知道考不考，详细介绍链接：

http://c.biancheng.net/cpp/html/68.html}









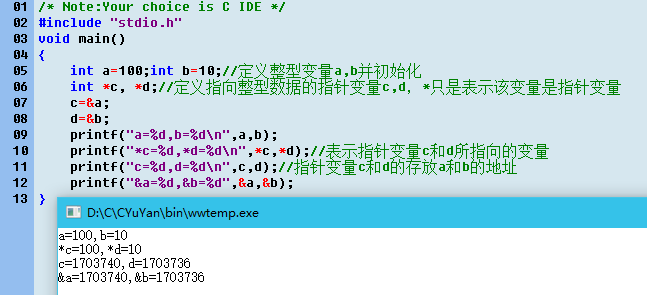
**第8章 善于利用指针**

1、地址指向该变量单元，地址就是指针【指针是一种用来表示地址的类型】

2、如果有一个变量专门用来存放另一变量的地址，则它称为“指针变量”，指针变量的值是地址【例如：变量i的指针（地址）是2000，不能说i的指针变量是2000】

3、可以理解“\*”表示指向，\*p表示指针变量p所指向的变量

4、相关程序定义演示：



5、可以在定义指针变量时，同时对其初始化

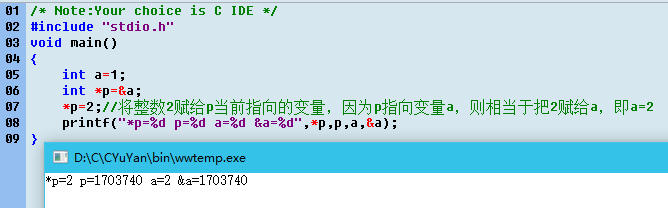
例如：int \*c=&a； //定义指针变量c，并指向a

6、根据演示，7、8两行不能写成\*c=&a；\*d=&b；因为a的地址是赋给指针变量c，而不是赋给\*c（即变量a）

7、一个指针变量只能指向同一个类型的变量

8、指针变量只能存放地址，不要将一个整数赋给一个指针变量

9、引用指针变量指向变量：



10、函数的参数可以是指针类型，它的作用是将一个变量的地址传送到另一个函数中

11、指针变量在函数调用时，将实参变量的值传递给形参变量，采用的还是值传递。指针变量通过调用函数使变量的值发生变化，在主调函数中可以使用这些改变的值

12、不能通过改变指针形参的值而使指针实参的值改变，因为是值传递，不能通过执行调用函数来改变实参指针变量的值，但是可以改变实参指针变量所指变量的值

13、函数的调用只可以得到一个返回值（即函数值），而使用指针变量做参数，可以得到多个变化了的值

14、注意区分8.2和8.3的区别，一个是a，b值未交换，但是指针指向的地址交换了，所以才改变，后者是值交换了

15、如例8.5。可以理解为在指针函数中，使用函数后面跟得是指针变量，而不是指针的值，例如定义是void swap (int \*1,int\*2)，但是使用时是swap(q1,q2)；暂时先这么记

16、数组元素的指针就是数组元素的地址

17、在C语言中，数组名（不包括形参数组名，形参数组并不占据实际的内存单元）代表数组中首元素（即序号为0的元素）的地址

【=&a[0];与p=a;等价】

p=a;的作用是“把a数组的首元素的地址赋给指针变量p”

而不是“把数组a各元素的值赋给p”

18、int \*p=&a[0];等价于int \*p;p=&a[0];（也可以定义成int \*p=a;//作用是将a数组的首元素的地址赋给指针变量p而不是赋给\*p）

19、在指针指向数组元素时，可以对指针进行如下运算：

（加一个整数、减一个整数、自加运算、自减运算、两个指针相减）

20、如果指针变量p已指向数组中的一个元素，则p+1指向同一数组中的下一个元素，p-1指向同一数组中的上一个元素。注意：执行p+1时并不是将p的值（地址）简单地加1，而是加上一个元素所占用的字节数（p+1实际代表p+1\*d，d是字节数）

21、如果p的初值是&a[0]，则p+i和a+i就是数组元素a[i]的地址，例如p+9和a+9的值是&a[9]，它指向a[9]

22、[]实际上是变址运算符，将a[i]按a+i计算地址，然后找出地址单元中的值

23、如果指针变量p1和p2都指向同一数组，p2-p1，结果是p2-p1的值（两个地址的差）除以数组元素的长度，表示p2所指的元素与p1所指的元素之间差2个元素

24、\*(p+i)或\*(a+i)是p+i或 a+i所指向的数组元素，即a[i]。例如：\*(p+5)或\*(a+5)就是a[5]。即\*(p+5)，\*(a+5)，a[5]三者等价

25、可以通过改变指针变量的值指向不同的元素

26、不能用a++，因为数组名a代表数组首元素的地址，它是一个指针型常量，它的值在程序运行期间是固定不变的，a是常量，a++不合法

27、当指针变量指向数组元素时，指针变量可以带下标。因为在程序编译时，对下标的处理方法是转换为地址的，对p[i]处理成\*（p+1）

28、相关运算区分：

①p++；\*p；p++使p指向下一个元素a[1]。然后若再执行\*p，则得到下一个元素的a[1]的值

②\*p++，因为++与\*同优先级，结合方向是自右向左，所以等同于\*（p++）,先引用p的值，实现\*p的运算，然后再使p自加1，这样下一次循环，\*p就是下一个元素的值

③\*（++p）先使p加1，再取\*p，若p初值为a（即&a[0]），若输出\*（p++），得到a[0]的值；若输出\*（++p），得到a[1]的值

④++（\*p）表示p所指向的元素的值加1，注意是元素的值加1，而不是指针p的值加1

29、当用数组名作参数时，如果形参数组中各元素的值发生变化，实参数组元素的值随之变化。实参数组名代表该数组首元素的地址，而形参是用来接收从实参传递过来的数组首元素地址的，因此，形参应该是一个指针变量（只有指针变量才能存放地址）

fun(int arr[],int n)与fun(int \*arr,int n)等价

30、可以理解为形参数组与实参数组共用一个存储单元

31、实参数组名代表一个固定的地址，或者说是指针常量，但形参数组名并不是一个固定的地址，而是按指针变量处理

32、如果一个实参数组，要想在函数中改变此数组中的元素的值，实参与形参的对应关系有4种：

①形参与实参都用数组名

②实参用数组名，形参用指针变量

③实参形参都用指针变量

④实参为指针变量，形参为数组名

（如果用指针变量作实参，必须先使指针变量有确定值，指向一个已定义的对象；已上四种方法都是地址的传递）

33、对于二维数组来说，a代表二维数组首元素的地址，现在的首元素不是一个简单的整型元素，而是由4个整型元素所组成的一维数组，例如a+1的值是a[1]的首地址，a[1]的值是&a[1][0]

34、a[0]+1和\*（a+0）+1是a[0][1]的地址,那么\*（a[0]+1）就是a[0][1]的值,同理\*(\*（a+0）+1)或\*(\*a+1)也是a[0][1]的值

35、a+1与a[0]+1是不同的,a+1是序号为1的行的首地址,a+1指向序号为1的行;而\*(a+1)或a[1]或a[1]+0都指向1行0列的元素,两者虽然地址相同,但是含义不同

36、二维数组名是指向行的,因此a+1中的1代表一行中全部元素所占的字节数。在指向行的指针前面加一个\*,就转为指向列的指针例如a+1变成\*(a+1),它就指向列的指针,指向1行0列元素的指针反之亦然,a[0]变成&a[0],指向二维数组的0行

37、二维数组中,不要简单地认为\*(a+i)是a+i所指单元中的内容,因为其不是指向具体存储单元而指向行

38、计算a[i][j]在数组中的相对位置的公式为:i\*m+j;其中m为二维数组的列数

39、p=&a[0],则p+1不是指向a[0][1],而是指向a[1];如果是p=a[0],那p+1所指向的元素是p所指向的列元素的下一个元素【注意区分】

40、int （\*p）[4]表示定义p为一个指针变量，它指向包含4个整型元素的一维数组。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的基类型是一维数组，其长度是16字节

41、实参与形参如果是指针类型，应当注意它们的类型必须一致。不应把int\*型的指针（即元素的地址）传给int（\*）[4]型（指向一维数组）的指针变量，反之亦然

42、在定义字符数组string时未指定长度，由于对它初始化，因此它的长度是确定的，数组名string代表字符数组首元素的地址，例如：string[7]就是\*（string+7），string+7是一个地址

43、在C语言中只有字符变量，没有字符串变量

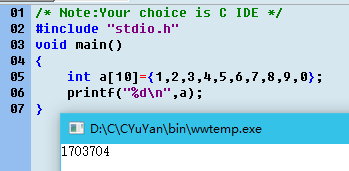
char \*string=“I love china”；

等价于

char \* string；string=“I love china”；

44、可以通过字符指针变量输出它所指向的字符串

45、通过字符数组名或字符指针变量可以输出一个字符串；但是数值型数组是不能用数组名输出它的全部元素，它输出的是数组首元素的地址



46、字符复制注意长度，可能有多余的，所以要用%s格式输出，如果用%c，逐个输出是可以输出后面这些字符的

47、由于字符可以用ASCII码代替，所以while(\* from!=0)与while（\* from）是等价的

48、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第一个字符的地址，绝不是将字符串放到字符指针变量中。可以对字符指针变量赋值，但不能对数组名赋值，因为数组名是地址，是常量

49、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

50、编译时为字符数组分配若干存储单元，而对字符指针变量，只分配一个存储单元。如果定义了字符指针变量却没有赋予一个地址值，后果很危险，应当及时指定指向

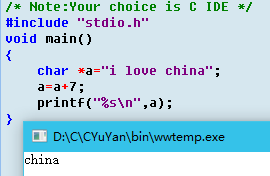
char \*a；str[10]；

a=str；

scanf(“&s”,a)；

先使a有确定值，使a指向一个数组元素，然后输入一个字符串，把它存放在以该地址开始的若干单元中

51、指针变量的值可以改变，数组名不行



输出从a指向的字符开始的字符串

52、字符数组中的各元素的值是可以被改变的，但字符指针变量指向的字符串常量中的内容不能被改变，例如;

char a[]=”house”;

char \*b=”house”;

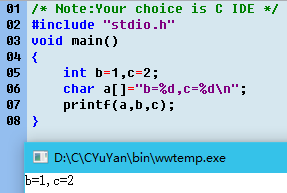
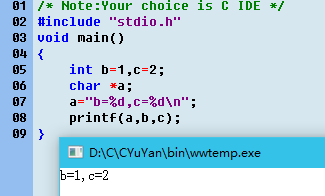
a[2]=’r’；//合法

b[2]=’r’；//不合法

53、若字符指针变量p指向字符串常量，就可以用指针变量带下标的形式引用所指的字符串中的字符

54、可变格式输出函数：

例如：



char \*a="b=%d,c=%d\n";也可以

55、函数指针为函数代码存储空间的起始地址，格式：int(\*p)(int,int)指向函数类型为整型且有两个整型参数的函数，注意\*p不能省略，表示p先与\*结合，是指针变量，()表示是函数

57、函数指针中，p=max的作用是将函数max的入口地址赋给指针变量p，函数名代表该函数的入口地址，因为只能指向入口处，所以\*(p+1)不合法

58、定义指向函数的指针变量【类型名指的是函数返回的类型】

格式：类型名（\*指针变量名）（函数参数表列）

59、指向函数的指针变量只能指向在定义时指定的类型的函数，在程序中，一个指针变量可以先后指向同类型的不同函数

60、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，例如p=max；如果写成p=max（a，b）；那就是将调用max函数所得到的函数值赋给p，而不是将函数入口地址赋给p

61、用函数指针变量调用函数时，只须将（\*p）代替函数名即可（p为指针变量名），在（\*p）之后的括号中根据需要写上实参

62、对指向函数的指针变量不能进行算术运算

63、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。原理：fun函数有两个形参x1和x2，定义其为指向函数的指针变量。调用时，实参为两个函数名f1和f2，给形参传递的是函数f1和f2的入口地址

64、定义返回指针值的函数的一般形式：

类型名 \* 函数名（参数表列）

**第9章 用户自己建立数据类型**

1、声明一个结构体类型的一般形式：

struct 结构体名

{成员列表};//成员列表包括：类型名 成员名；

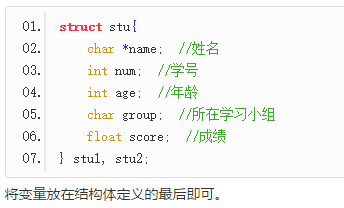
2、成员可以属于另一个结构体类型，例如定义一个struct Date类型，别的类型在引用时可以是struct Date 成员名；

3、定义结构体类型变量的3种方法：

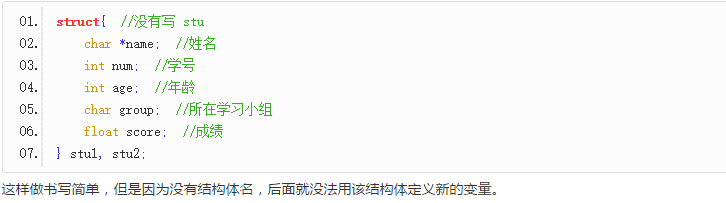
①先声明结构体类型，再定义该类型的变量



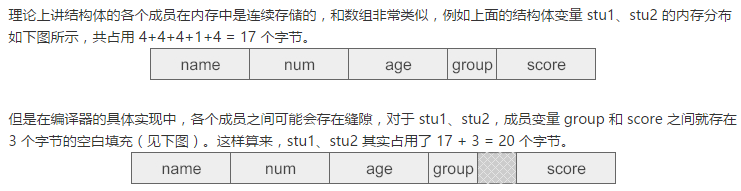
②在声明类型的同时定义变量



③不指定类型名而直接定义结构体类型变量



4、在定义了结构体变量后，系统会为之分配内存单元



5、结构体类型与结构体变量不同。只能对变量赋值、存取或运算，而不能对一个类型这么做，编译时，对类型是不分配空间的，只对变量分配空间

6、结构体类型中成员名可以与程序中的变量名相同，但二者不代表同一对象

7、结构体变量的成员可以单独使用，它的作用与地位相当于普通变量

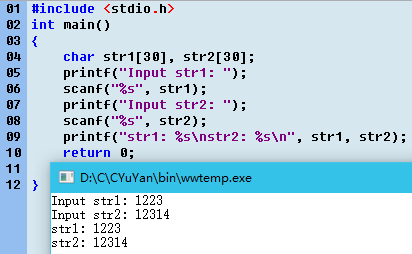
8、在定义结构体变量时可以对他的成员初始化，初始化列表是用花括号括起来的一些常量，注意是对结构体变量初始化，而不是结构体类型【C99允许struct Stu b={.name =”zhang”};】

9、可以引用结构体变量中成语啊你的值，例如student1.mum=10010；

.是成员运算符，优先级最高。不要输出结构体变量名来达到输出结构体变量所有成员的值，只能对各个成员分别输入输出

10、若成员本身属于结构体类型，要用若干个.，只能对最低级的成员进行赋值或存取以及运算

11、结构体变量的成员可以像普通变量一样进行运算，同类型结构体变量可以相互赋值，可以引用结构体变量的地址

12、在scanf中，注意数组名前面不要加&，不要画蛇添足

13、定义结构体数组的一般形式：

①struct 结构体名{成员表列}数组名[数组长度]；

②结构体类型 数组名[数组长度]；

14、对结构体数组初始化是在定义数组的后面加上：={初值列表}；

15、一个结构体变量的起始地址就是这个结构体变量的指针

16、C语言允许把(\*p).num用p->num来代替，“->”代表一个箭头，p->num表示p所指向的结构体变量中的num成员，->称为指向运算符【stu.num与(\*p).num与p->num等价】

17、(++p)->num先使p自加1，然后得p指向的元素中的num成员值

(p++)->num先求得p->num的值，再使p自加1指向stu[1]

18、如果要将某一成员的地址赋给p，可以使用强制类型转换

p=(struct Student \*)stu[0].name

19、定义共用体类型的一般形式：

union 共用体名{成员表列}变量表列；

20、共用体变量所占的内存长度等于最长的成员的长度

21、先定义，后使用，不能引用共用体变量，只能引用共用体变量中的成员，例如printf(“%d”,a.i);

22、可以对共用体变量初始化，但初始化表中只能有一个变量，c99允许对指定的一个成员初始化union data a={.ch=‘j’}；

23、共用体变量中起作用的成员是最后一次被赋值的成员

24、C99允许同类型的共用体变量互相赋值

25、声明枚举类型的一般形式：enum 枚举名 {枚举元素列表}；

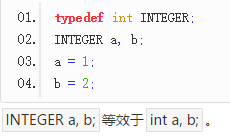
26、C编译对枚举类型的枚举元素按常量处理，故称枚举常量，不能对它们赋值，每一个枚举元素都代表一个整数，按定义时的顺序默认他们的值从0开始

27、枚举常量是可以引用和输出的printf(“%d”,workday);

28、用typedef声明新类型名的2种情况：

①简单地用一个新的类型名代替原有的类型名，格式为：

typedef oldName newName;



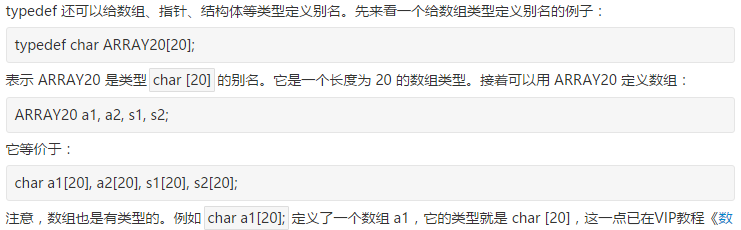
②命令一个简单的类型名代替复杂的类型表示方法

按定义变量的方式，把变量名换上新类型名，并且在最前面加typedef，就声明了新类型名代表原来的类型

int a[100];→int Num[100];→typedef int Num[100]；【Num=a；】

等价于int a[100]；

29、用typedef声明的新类型称为原有类型的typedef名称，用typedef可以将数组类型和数组变量分离开来，利用数组类型可以定义多个数组变量



30、#define是在预编译时处理的，它只能作简单的字符串替换，而typedef是在编译阶段处理的。实际上并不是作简单的字符串替换

**第10章 对文件的输入输出**

1、所谓文件一般指存储在外部介质上数据的集合，输入输出被形象地成为流，即数据流，流表示了信息从源到目的端的流动

2、C语言把文件看作是一个字符（或字节）的序列，即由一个一个字符或字节的数据顺序组成，一个输入输出流就是一个字符流或字节（二进制）流

3、C的数据文件由一连串的字符或字节组成，不考虑行的界限，对文件的存取是以字符或字节为单位的，输入和输出数据流的开始和结束仅受程序控制而不受物理符号（如回车换行）控制

4、文件标识包括：文件路径、文件名主干、文件后缀

5、数据文件分为ASCII文件（文本文件）和二进制文件（映像文件）

6、字符一律以ASCII码形式存储，数值型数据两者皆可

7、缓冲文件系统是指系统自动地在内存区为程序中每一个正在使用的文件开辟一个文件缓冲区，其中关键的概念是文件类型指针，简称文件指针，结构体类型FILE

8、FILE \*fp；指向FILE类型变量的指针变量，可以使fp指向某一个文件的文件信息区，通过该区的信息就可以访问该文件，所以通过文件指针变量能够找到与他关联的文件【又叫指向文件的指针变量】

9、用fopen函数打开数据文件，方式为fopen(文件名。使用文件方式);

例如:

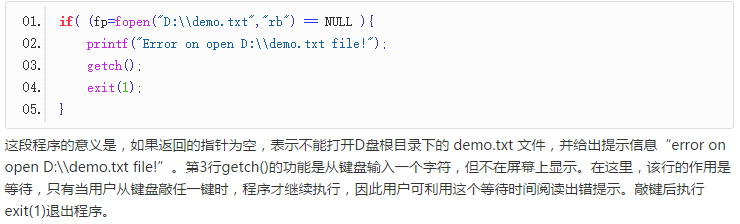
FILE \*p;

fp=fopen(“a1”,”r”);表示要打开名字为a1的文件，使用方式为读入，函数的返回值是指向a1文件的指针



带有w的如果指定文件不存在会建立新文件，其余报错。报错的原因除了文件不存在，还有可能是磁盘故障，磁盘已满等。次数fopen函数将带回一个空指针值NULL（在stdio.h中，NULL已被定义为0）

10、exit函数的作用是关闭所有文件，终止正在执行的程序，待用户检查出错误，修改后重新运行



11、计算机输从ASCII文件读入字符时，遇到回车换行符，系统把它转换为一个换行符，在输出时把换行符转换成为回车和换行两个字符（二进制不需要）

12、程序中可以使用3个标准的流文件：

标准输入流（从终端输入stdin），标准输出流（向终端输出stdout），标准出错输出流（当程序出错时将出错信息发送到终端上sdterr）

13、fclose函数的形式：fclose(文件指针)；

文件正常关闭时，fclose() 的返回值为0，如果返回EOF(-1)则表示有错。

14、顺序读写需要用库函数实现

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 调用形式 | 功能 | 返回值 |
| fgetc | fgetc(fp) | 从fp指向的文件读入一个字符 | 读成功，带回所读的字符，失败则返回文件结束标志EOF（即-1） |
| fputc | fputc(ch,fp) | 把字符ch写到文件指针变量fp所指向的文件中 | 输出成功，返回值就是输出的字符；输出失败，则返回EOF（即-1） |

15、在程序中用putc和fputc作用是一样的，用getc和fgetc作用是一样的

16、C语言允许通过函数fgets和fputs一次读写一个字符串，例如：

fgets（str,n,fp）;

作用是从fp所指向的文件中读入一个长度为n-1的字符串，并在最后加一个\0字符，然后把这n个字符存放到字符数组str中。

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 调用形式 | 功能 | 返回值 |
| fgets | fgets(str,n,fp) | 从fp指向的文件读入一个长度为n-1的字符串, 存放到字符数组str中 | 读成功，返回地址str，失败则返回NULL |
| fputs | fputs(str,fp) | 把str所指向的字符串写到文件指针变量fp所指向的文件中 | 输出成功，返回值0；否则返回非0值 |

17、fgets函数的函数原型为：

char \* fgets （char \*str，int n，FILE \*fp）；

18、fputs函数的函数原型为：

int fputs (char \*str,FILE \*fp);、

19、由于C语言中把\作为转义字符的标志，因此在字符串或字符中要表示\时，应当在\之前再加一个\

20、fputs(“\n”,fp);换行，作为字符之间的间隔

21、按fgets函数的规定，如果遇到\n就结束字符串输入，\n作为最后一个字符也读入到字符数组

22、用格式化方式读写文件，需要fprintf函数和fscanf函数，他们的读写对象不是终端，而是文件，调用方式为：

fprintf(文件指针，格式字符串，输出表列);

fscanf(文件指针，格式字符串，输出表列);

例如：fprintf(fp,”%d,%6.2f”,i,f);

它的作用是将int型变量i和float型变量f的值按%d和%6.2f的格式输出到fp指向的文件中

23、C语言允许用fread函数从文件中读一个数据块，用fwrite函数向文件写一个数据块，在读写时是以二进制形式进行的，调用形式为：

fread（buffer,size,count,fp）;fwrite(buffer,size,count,fp);

buffer：是一个地址，对于fread用来存放从文件读入数据的储存区的地址，；对于fwrite，是要把此地址开始的存储区中的数据向文件输出（指的是起始地址）

size:要读写的字节数

count：要读写多少个数据项（每个数据项长度为size）

fp:FILE类型指针

例如：fread（f，4,10，fp）；

其中f是一个float型数组名（代表数组首元素地址），这个函数从fp所指向的文件读入10个4个字节的数据，存储到数组f中

例如：fread(&stud[i],sizeof(struct Student\_type),1,fp);

fread或fwrite函数的类型为int型，如果执行成功，返回值为形参count的值（一个整数），即输入个数

24、fread和fwrite函数一般用于二进制文件的输入输出。因为他们是按数据块的长度来处理输入输出的，不出现字符转换

25、对流式文件既可以进行顺序读写，也可以进行随机读写，关键在于控制文件的位置标记

26、用rewind函数使文件位置标记指向文件开头，没有返回值

27、用fseek函数改变文件位置标记,调用形式为：

fseek(文件类型指针，位移量，起始点)

起始点用0,1,2代替，0代表文件开始位置，1代表当前位置，2代表文件末尾，位移量指以起始点为基点，向前移动的字节数，位移量应当是long型，在数字末尾加上L，fseek函数一般适用于二进制文件

28、ftell函数的作用是得到流式文件中文件位置标记的当前位置

i=ftell(fp);if(i==-1L)printf(“error\n”);

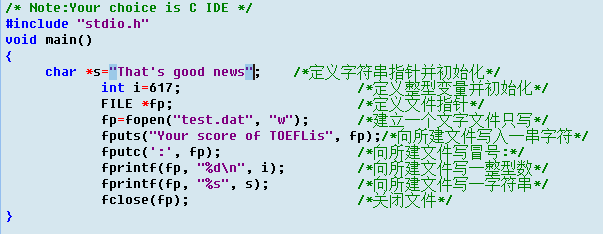
调用函数出错，（如不存在fp指向的文件），ftell函数返回值为-1L

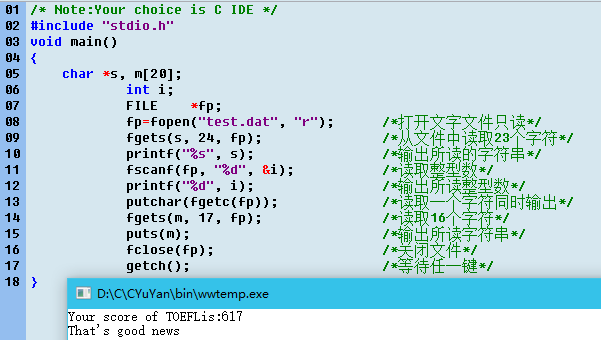
29、文件读写出错检测：

①ferror函数，格式为ferror（fp）；如果返回值为0（假），表示未出错，返回值非零，表示出错。应该在调用一个输入输出函数后立即检查ferror函数的值。在执行fopen函数时，ferror函数的初始值自动为0

②clearerr函数的作用是使文件错误标志和文件结束标志置为0，假设出现错误，ferror为一个非零值，应该立即调用clearerr(fp)，使ferror(fp)的值变为0，便于下次检查，只要出现文件读写错误标志，它就一直保留，直到对同一文件调用clearerr函数或rewind函数，或任何一个输入输出函数

30、相关案例：





feof()函数检测文件位置指示器是否到达了文件结尾,若是则返回一个非0 值, 否则返回0。这个函数对二进制文件操作特别有用, 因为二进制文件中,文件结尾标志EOF也是一个合法的二进制数,只简单的检查读入字符的值来判断文件是否结束是不行的。如果那样的话, 可能会造成文件未结尾而被认为结尾, 所

以就必须有feof()函数。

下面的这条语句是常用的判断文件是否结束的方法。

while(!feof(fp))

fgetc(fp);