C 语言程序设计基础知识期末复习

一、C 语言与算法

1. 程序：一组计算机能识别和执行的指令。
2. C 语言的特点：运算符丰富(共有 34 种运算符)、数据类型丰富、具有结构化的控制语句。
3. C 语言程序的结构特点：

（1）.一个程序由一个或多个源程序文件组成：一个源程序文件中可以包括三个部分：预处理指令、全局声明、函数定义

（2）.函数是 C 程序的主要组成部分：一个 C 程序是由一个或多个函数组成的必须包含一个 main 函数（只能有一个）；每个函数都用来实现一个或几个特定功能；被调用的函数可以是库函数，也可以是自己编制设计的函数。

（3）.一个函数包括两个部分 函数首部和函数体（声明部分：定义在本函数中所用到的变量；对本函数所调用函数进行声明；执行部分：由若干个语句组成，指定在函数中所进行的操作）

（4）. 程序总是从 main 函数开始执行

（5）. C 程序对计算机的操作由 C 语句完成

（6.） 数据声明和语句最后必须有分号

（7.） C 语言本身不提供输入输出语句

（8.） 程序应当包含注释，增加可读性

4、算法 （1）算法 + 数据结构 = 程序

（2）顺序结构、选择结构、循环结构是表示一个良好算法的基本结构

（3）算法的特性：有穷性、确定性、有零个或多个输入、有一个或多个输出、有效性

（4）算法流程图：一个流程图包括以下几部分：表示相应操作的框；带箭头的流程线；框内外必要的文字说明。

流程线不要忘记画箭头，否则难以判定各框的执行次序。

算法流程图的结构特点：只有一个入口；只有一个出口（一个判断框有两个出口；一个选择结构只有一个出口）结构内的每一部分都有机会被执行到。也就是说，对每一个框来说，都应当有一条从入口到出口的路径通过它；结构内不存在“死循环”。

二、顺序结构设计

5、标识符：用来标识变量名、符号常量名、数组名、类型名、函数名等的有效字符序列。

C 语言对标识符的规定：

（1）只能由字母、数字、下划线组成，且第一个字母必须是字母或下划线

（2）长度：在 TC 中最多允许 32 个字符，建议不要超过 8 个

如果系统规定标识符长度为 8，那么 x1234567A 和 x1234567B 会被视为同一标识符。

（3）在 C 语言中，标识符大小写敏感。如 ABC、aBC、abc 分别代表 3 种不同的标识符

（4）标识符不能与 C 语言的保留字、系统标准库函数同名。

program to_ _to file_2 ab1_c3 为合法标识符

非法字符举例：yes? (含有不合法字符“？”)123(第一个字符不能为数字)go to(标识符中不允许有空格）

a_80%(出现非法字符“%”)if 与关键字相同）

6、常量与变量

注意：以下变量定义形式是错误的 int a,int b; int a;b;

（1）整型常量

- 十进制整型：能出现数字 0~9，可带正负号 如：0，11，95，-2

• 八进制整型：以数字 0 开头的数字串，能出现数字 0~7 如：011（十进制 9），0111（十进制 73）

• 十六进制整型：以 0x 开头的，能出现数字 0~9，字母 a~f 或 A~F 如：0x11(十进制 17)，0xa5(十进制 165)

（2）整型变量 基本类型 int

（3）浮点型

• 浮点型常量 十进制小数形式：由数字与小数点组成（必须有小数点）。

如 1.23，-123.，0.0123，.0，0.

指数形式，如 123e3 或 123E3 都代表 123×10^3 注意字母 e 或 E 之前必须有数字，后面必须是整数 E-5，.1234e1.2，6.5E 为非法的浮点型常量

• 浮点型变量 float double

（4）字符型

1）字符常量：分为两类

• 用单撇号括起来的一个字符 如 ：’a’，’9’，’ ’（空格）是合法字符常量

”a”，’99’ 是非法字符常量

• 转义字符：以\（反斜杠）开头的特殊形式的字符。如：’\n’ ,’\r’ ,’\123’ ,’\x3b’

2）字符变量：char，占 1 个字节空间，只能存放一个字符。

存储特点：在字符变量中实际上存储的是字符的 ASCII 码，其存储形式与整数的存储形式相同。

注意：

I.字符数据与整型数据可相互赋值，直接运算。

II.大小写字母转换

小写字母减 32 得到相应的大写字母， 如：‘a’-32 得到‘A’

大写字母加 32 得到相应的小写字母 如：‘B’+32 得到‘b’

（5）字符串常量

定义：用一对双撇号(” ”)括起来的字符序列。如：”hello” ”Mary” ”\\aaa\’\n ”

存储：每个字符串尾自动加一个 ‘\0’ 作为字符串结束标志

（6）变量赋初值

格式：类型说明符 变量 1=常数 1[，变量 2=常数 2[，•••]]；int x=1,y=1,z=1;

int x=y=z=1;（语法错误）

7、算数运算符和算术表达式

（1）运算符的注意事项

• 除法运算符“/”进行求商运算。对于不同类型的运算对象，除法表达式计算结果的类型也会不同。如果 x,y 是整型，则结果为整型，小数部分被略去。如果 x,y 其中一个为浮点型量，则结果为浮点型。

• “%”是求余运算。a%b 计算 a 除以 b 后的余数，也是 a 模 b 的值。它要求两个运算对象必须是整型，其结果也是整型量。

（2）书写算术表达式的注意事项

• 将方括号改成圆括号，即算术表达式内所有的括号均为圆括号，*（乘法）不能省略

• 其中 π 为非字母字符，要用浮点型常量代替

• 其中不能出现分数，改用除法运算符，圆括号不能缺少

（3）算数运算符的优先级

优先级：高 - ---->* / % -----> + - 低 注意:可以用( )来改变运算的顺序.

(2) (3) (4)

8、自增、自减运算符

后缀　i＋＋（ i－－）先使用 i 的值，再使 i 的值加(减)1

前缀　＋＋i （－－i）先使 i 的值加(减)1 ，再使用 i 的值

自增、自减运算只能用于变量，不能用于常量和表达式。

自增、自减运算符高于基本算术运算符。

9、 赋值运算符

（1）简单赋值运算符 运算符：= 表达式格式：变量＝表达式

• 赋值运算符左边必须是变量而不能是表达式；

• 赋值表达式的值是赋值号左边变量被赋值后的值；

（2）算术运算符“+ — * / %”和赋值运算符“=”结合起来，形成复合赋值运算符。

+= ：加赋值运算符; 如 a+=3，等价于 a=a+3 -=、*=、/=、%= 与此类似

• 复合运算符在书写时，两个运算符之间不能有空格。

• 复合运算符右边的表达式计算完成后才参与复合赋值运算；

10、当表达式中的数据类型不同时，要进行类型转换。

转换方式 自动（隐式）转换：系统自动把数据由低级类型向高级转换。

强制转换：将表达式的运算结果强制转换成指定的数据类型。

自动转换规则 强制转换 格式：（目标类型名）（表达式）

double←long←unsigned←int

↑ ↑

float 高 ← 低 char short

逗号运算符与逗号表达式：运算优先级最低为 15 级

求解过程先求表达式 1 的值，再求表达式 2 的值，•••，直至求出表达式 n 的值。整个表达式的值为表达式 n 的值。

x=5*8, 6+9 先把 x 赋值为 40，表达式的值为 15

11、字符数据的简单输入和输出

（1） putchar 函数（字符输出函数） 形式：putchar(c)

• c 可以是字符常量、字符变量或整型变量。putchar(100);

• 用 putchar 函数也可输出转义字符。putchar(‘\n’); /*输出一个换行符*/

putchar(‘\\’); /*输出一个反斜杠\ */

• 该函数包含在 stdio.h 库中，因此应在使用该函数的程序开头加入：#include <stdio.h>

（2）getchar 作用：从终端（如键盘）输入一个字符。

形式：getchar( ) 无参数

说明：该函数只能接收一个字符，其函数值可以赋给一个字符变量或整型变量，也可作为表达式的一部分，该函数包含在 stdio.h 中。

（3）格式输出函数 格式：printf（格式控制，输出表列）

格式控制是用双撇号括起来的字符串，也称“转换控制字符串”，包括 2 种信息。

1）格式说明： % [<附加格式字符>]格式字符

将输出列表中的数据转换为指定格式输出。

2）普通字符：原样输出。

输出表列：需要输出的数据列表，彼此间用逗号分隔。它可是任意合法的表达式。

printf（" a+b = %5.2 f " , c ）

（4）格式字符

1）d 格式符，用来输出十进制整数

%d 按整型数据的实际长度输出。

%md　m 代表某个数字，指定输出数据的最小宽度。若数据的位数小于 m，则左侧补空格（右对齐），若大于 m 则按实际位数输出。%-md 与%md 类似，只是左对齐

2） c　格式符：用来输出一个字符 %c　%mc　%-mc

3） f 格式符：用来以十进制小数形式输出实数（float,double）

%m.nf　输出数据最小占 m 列，其中包括 n 位小数和 1 位小数点，右对齐　(小数点也算占一列)　%-m.nf　与上面类似，只是左对齐

4）S 格式符：用来输出一个字符串

%s　按字符串原长输出　%ms　输出字符串最小占 m 列，右对齐，左补空格

%-ms　输出字符串最小占 m 列，左对齐，右补空格

%m.ns　字符串占 m 列，但只取左端 n 个字符，右对齐

%-m.ns　字符串占 m 列，但只取左端 n 个字符，左对齐

使用说明：

1）格式控制中的格式说明符，必须按从左到右的顺序，与输出表中的每个数据一一对应，否则出错。

printf(“f=%d, i=%f\n”, 5.6,3); 显示：f= 1717986918,i=0.000000

2）格式字符紧跟在“%”后面就作为格式字符，否则将作为普通字符使用（原样输出）。

printf(”c=%c, f=%ff\n“, ‘a’,1.5); 其中的第一个 c 和 f,第三个 f，都是普通字符。

（5）scanf 函数　scanf（格式控制字符串，地址列表）

格式控制与 printf 函数类似。

地址列表是由若干个地址组成的表列(以逗号隔开)，可以是变量的地址，或字符串的首地址。

scanf(“%d%d%d”,&a,&b,&c);

1） 格式符中无普通字符时，可用空格、Tab 键、回车键作分隔符。最后的回车键代表输入结束

2）用 c 格式符输入字符时，空格、回车、转义字符等均为有效字符。

例：　char a,b,c;

scanf（"%c%c%c", &a, &b, &c）;

正确的输入方法： 键入　ABC　则 a=‘A’，b=‘B’，c=‘C’

若键入： A　B　C　则 a=‘A’，b=‘ ’(空格)，c=‘B’

若键入　A　B　C 未送，系统已经认为输入结束了 则： a=‘A’，b=‘\n’（换行符），c=‘B’

若不同类型输入 则综合上述规则

3）在格式控制中除格式说明符外若还有其它字符,则应按顺序原样输入。

4）可以指定输入数据所占列数，系统自动按它截取所需数据。

5）%后的“*”附加说明符，用来表示跳过相应的数据。

如:scanf(“%2d%*3d%2d”,&a,&b); 输入 1234567　则将 12↔a，67 ↔b，345 被跳过

6）输入数据时不能规定精度。如：scanf(“%7.2f”,&a);错误

7） double 类型的变量输入时,要用%lf%le(必须记住!!!）

如:

double x;

scanf(“%lf”,&x);

三、选择结构程序设计

1、if 语句实现选择

if (表达式) 语句 1　表达式可以是关系表达式、逻辑表达式、数值表达式

else 语句 2

最常用的 3 种 if 语句形式：

（1） if (表达式) 语句 1 (没有 else 子句)

（2）.if (表达式) 语句 1

（3） else 语句 2 (有 else 子句)

（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ｍ） 语句ｍ

else 语句 m+1

(在 else 部分又嵌套了多层的 if 语句)

说明：

（1）整个 if 语句可写在多行上，也可写在一行上,但都是一个整体，属于同一个语句

（2）“语句 1”…“语句 m”是 if 中的内嵌语句 内嵌语句也可以是一个 if 语句

（3）“语句 1”…“语句 m”可以是简单的语句，也可以是复合语句

2、关系运算符

（1）关系运算符：

用来对两个数值进行比较的比较运算符

C 语言提供 6 种关系运算符：

① ＜ (小于) ② ＜= (小于或等于) ③ ＞ (大于) ④ ＞= (大于或等于) 优先级高

⑤ == (等于) ⑥ != (不等于) 优先级低

（2）关系表达式

用关系运算符将两个数值或数值表达式连接起来的式子,关系表达式的值是一个逻辑值，即“真”或“假”,在 C 的逻辑运算中，以“ 1 ”代表“真”，以“ 0 ”代表“假”

3、逻辑运算符与逻辑表达式

3 种逻辑运算符：&&（逻辑与） ||（逻辑或） !（逻辑非）

判断年龄在 13 至 17 岁之内？ age>=13 && age<=17

逻辑运算符的优先次序 ! → && → || (!为三者中最高）

逻辑表达式的值应该是逻辑量“真”或“假”

编译系统在表示逻辑运算结果时以数值 1 代表“真”，以 0 代表“假”

但在判断一个量是否为“真”时以 0 代表“假”，以非 0 代表“真”注意：将一个非零的数值认作为“真”

在进行逻辑表达式的求解中，并不是所有的逻辑运算都被执行，只是在必须执行下一个逻辑运算符才能求出表达式的值时，才执行该运算符。

4、条件运算符与条件表达式

条件表达式的一般形式为：表达式 1 ？ 表达式 2 : 表达式 3

条件运算符的执行顺序：

求解表达式 1

若为非 0（真）则求解表达式 2，此时表达式 2 的值就作为整个条件表达式的值

若表达式 1 的值为 0（假），则求解表达式 3，表达式 3 的值就是整个条件表达式的值

条件运算符的结合方向为“自右至左”

5、switch 语句

switch 语句的作用是根据表达式的值，使流程跳转到不同的语句

switch 语句的一般形式：
switch（表达式）整数类型(包括字符型)
{ case 常量 1 ：语句 1；break
case 常量 2 ：语句 2
┆ ┆ ┆
case 常量 n ：语句 n
default : 语句 n+1
}
优先级顺序：赋值运算符→&& 和 ||→关系运算符→算术运算符→!

四、循环结构程序设计

1、用 while 语句实现循环
while 语句的一般形式如下：
while (表达式) 语句→循环体
↓
循环条件表达式 “真”时执行循环体语句“假”时不执行
while 循环的特点是：先判断条件表达式，后执行循环体语句
while 循环的作用：实现“当型”循环
使用说明：
在 while 的循环体中一定要有使循环趋于结束的语句；否则将形成死循环；
注意循环操作的范围、花括号、分号的使用；
注意给循环变量赋初值的位置及初值的正确性
2、用 do...while 语句实现循环
do---while 语句的特点：先无条件地执行循环体，然后判断循环条件是否成立
do---while 语句的一般形式为：
do
语句
while (表达式);
while 和 do...while 语句的比较
当 while 后面的表达式的第一次的值为“真”时，两种循环得到的结果相同；否则不相同
（1）. 循环体内必须有使循环趋于终止的条件

```
while   (i<=100)
{       sum=sum+i;
        i++;
}
```

```
do
{     sum=sum+i;
      i++;
} while   (i<=100);
```

（2）注意循环初值与循环条件

```
i=1;
while   (i<=100)
{   sum=sum+i;
    i++;
}
```

```
i=0;
while   (i<100)
  {     i++;
        sum=sum+i;
   }
```

（3） **do_while 循环的循环体至少执行一 次， while 循环的循环体可能一次也不执行。**
（4） 在循环体至少执行一次的前提下， do_while 与 while 循环等价。
3、for 语句实现循环

for 语句的一般形式为

for(表达式 1；表达式 2；表达式 3)

语句

表达式 1：设置初始条件，只执行一次。可以为零个、一个或多个变量设置初值执行

表达式 2：循环条件表达式，用来判定是否继续循环。在每次执行循环体前先执行此表达式，决定是否继续执行循环

表达式 3：作为循环的调整器，例如使循环变量增值，它是在执行完循环体后才进行的

for 语句说明 ：

1）表达式 1 可省略，但分号不能省 ；

如 ： int i=1,sum=0;

```
for (;i<=100;i++)
    sum=sum+i;
```

2）若表达式 2 省略，循环条件永远为真 ；

如: for(i=1; ;i++)

printf(“%d,”,i); 死循环 自己编程序时不建议采用，

3）表达式 3 也可省略，但应设法保证循环正常结束 ； 但要能看懂别人的程序

如 ： for (i=1;i<=100;)

```
{   sum=sum+i;
i++;
}
```

4）可只给循环条件;

```
i=1;
  for (;i<=100;)
{   sum=sum+i;
i++;
}
```

5）三个表达式都可省 ；

for (;;) 相当于 while (1)

6）表达式 1 和表达式 3 可以是逗号表达式 ；

如 ： for (i=1,sum=0;i<=100;i++)

sum=sum+i;

7）表达式 2 一般为关系表达式或逻辑表达式，但也可以是数值表达式或字符表达式，只要其值为非零就执行循环体 。

如： for (;(c=getchar( ))!='\n';)

printf("%c,",c);

注: 尽量避免用实型变量控制循环次数。

4、改变循环的执行状态

（1） 用 break 语句提前终止循环

break 语句作用 ：

1) 从循环体内跳出，即提前结束循环 ，接着执行循环下面的语句 ；

2) break 语句只能 用于循环语句和 switch 语句

注意：

在循环中使用了 break 语句后， 循环语句的结束可能有两种：

1. 正常结束（正常出口）:由于循环条件表达式为假

2. 非正常结束（异常出口）:由 break 语句引起

（2）continue 语句提前结束本次循环

结束本次循环，即跳过循环体语句中下面尚未执行的语句，接着执行下一次是否执行循环的判定

```
for (n=100;n<=200;n++)
{    if   (n%3= =0) continue;
     printf(”%d”,n);                    这段程序等价于 if   (n%3!=0)    printf(”%d”,n);
}
```

for 语句中执行的是表达式 3

（3）break 语句和 continue 语句的区别

continue 语句只结束本次循环，而不是终止整个循环的执行

break 语句结束整个循环过程，不再判断执行循环的条件是否成立

四、利用数组处理批量数据

1、一维数组

（1）定义一维数组

定义一维数组的一般形式为：类型符　数组名[常量表达式];

说明：

类型符：数组元素的类型。

数组名：即数组的名称，其命名方法同变量名。

在定义数组时，需要指定数组中元素的个数，方括弧中的常量表达式用来表示元素的个数，即数组长度。

注意:

1)在定义数组时,只能使用整常量表达式表明数组的大小,即数组元素的个数,不能是变量。也就是说，C 语言不允许对数组的大小作动态定义.

注意:

```
int n=10;
int arr[n];
```

错误，n 不是常量而是变量

数组说明中其他常见的错误：

① float a[0];　　/* 数组大小为 0 没有意义 */

② int b(2)(3);　/* 不能使用圆括号 */

③ int k, a[k];　/* 不能用变量说明数组大小*/

一维数组的存储方式：每个元素都有一个编号（从 0 开始）,称为下标。

（2）引用一维数组

一次只能引用一个数组元素 不能引用整个数组所有元素！也不能引用多个元素

引用数组元素的表示形式为：数组名［下标］

一个数组元素就是一个普通变量，跟普通变量一样使用。a[3]=a[2]%4

注意：

引用数组元素时，下标可以是整型常量、变量或整型表达式

对数组中所有元素逐个引用时，通常可使用循环结构。

（3）一维数组的初始化

数组的初始化：在定义数组的同时，给各数组元素赋值

格式：

类型符　数组名[表达式]＝{初值表};

给部分元素赋初值。例　　int a[8]={ 0,1,2,3,4 }; 后面的用 0 补齐

给全部元素赋初值时可不指定数组的长度。　int a[　]={ 0,1,2,3,4};注意：只有在初始化时[ ]中可以空着。其他情况不行，例如：int a[ ],b[ ];是错误的

2、二维数组

(1)定义二维数组 二维数组定义的一般形式为 类型符 数组名[常量表达式][常量表达式];

（2）引用二维数组　　数组名[行下标][列下标]　　行、列下标都是从 0 开始

二维数组的存储方式　二维数组在内存中按行存放

（3）二维数组的初始化　二维数组初始化通常是按行进行的

类型符 数组名[表达式 1][表达式 2]={初值表};

给全部元素赋初值。

例　int a[3][4]={{0,1,2,3},{4,5,6,7},{8,9,10,11}};

或写成: int a[3][4]={0,1,2,3,4,5,6,7,8,9,10,11}

给二维数组的全部元素赋初值，可以不指定第一维的长度，但第二维的长度不能省略。

```
#define N 10
#define M 6
```

若有定义：

int a[N][M];

二维数组按行输入标准模板：

```
for(i=0; i<N; i++)
    for(j=0; j<M; j++)
        scanf("%d", &a[i][j]);
```

二维数组按行输出标准模板：

```
for(i=0; i<N; i++)
{   for(j=0; j<M; j++)
        printf("%d", a[i][j]);
    printf("\n");
}
```

3、字符数组

（1）字符数组的定义：定义字符数组的方法与定义数值型数组的方法相同，使用关键字 char

格式：

char　数组名[常量表达式]，•••　；

（2）字符数组的初始化　一维字符数组初始化　1）逐个字符赋给数组中的各元素

2）用字符串常量初始化数组中的各元素

字符串在实际存储时，是用字符数组存储的，系统会自动的在其尾部添加一个结束标志’\0’

二维字符组初始化　二维字符数组的初始化，可以采用逐个字符式或者字符串常量的方式

（3）引用字符数组　　引用字符数组中的元素，与引用其他类型数组元素相同

只能是一个一个字符地引用

（4）字符数组的输入输出

1、利用 getchar、putchar 逐个处理（用循环结构）

利用格式符 %c 逐个输入、输出字符　for(i=0; i<9; i++)

scanf("%c"，&str[i]);

当字符数组存字符串时,可利用格式符%s，可以一次输入、输出整个字符串。

（不用循环结构）
scanf(“%s”,字符数组名); 用于输入一个字符串
printf(“%s”,字符数组名); 用于输出一个字符串
如果一个字符数组中包含多个’\0’，则遇第一个’\0’时输出就结束
scanf 函数中的输入项是已定义的字符数组名，输入的字符串应短于已定义的字符数组的长度
char str[10];
scanf("%s"，str)；
printf("%s\n", str)； 用字符数组名,不要加&，遇空格、TAB、回车结束输入
自动加 1 个‘\0’，输入串长度小于数组长度
怎样输入带空格的字符串？使用%c 格式符
for(i=0;(c=getchar())!=‘\n’;i++)
str[i]=c;
使用字符串处理函数 gets(str);
利用字符串输入输出函数
字符串输入函数 gets
格式：gets(字符数组名)
功能：从键盘接收一个字符串放入字符数组中，
自动加‘\0’,只以回车符作为输入的结束
说明：输入串长度应小于字符数组元素的个数
字符串输出函数 puts
格式：puts(字符数组名或字符串常量)
功能：向显示器输出一个字符串（输出完，自动换行）
说明：字符数组必须以‘\0’结束
gets()函数可以接收包含空格、tab 的字符串。scanf(“%s”,…)不能接收空格
puts( )函数一次输出一个字符串，并自动换行。
printf(“%s”,…)函数可以同时输出多个字符串，并且能灵活控制是否换行。
strcat 函数----字符串连接函数
其一般形式为：
strcat(字符数组 1，字符串 2)
其作用是把两个字符串连接起来，把字符串 2 接到字符串 1 的后面，结果放在字符数组 1 中
字符串拷贝函数 strcpy
格式：strcpy(字符数组 1，字符串 2)
作用：将字符串 2 复制到字符数组 1 中。
注意：
（1） 字符数组 1 必须足够大，以便容纳字符串 2 的内容。
（2） 字符串 2 可以是字符数组名或者字符串常量。当字符串 2 为字符数组名时，只复制第一个’\0’前面的内容（含’\0’ ），其后内容不复制。
（3）不能使用赋值语句为一个字符数组赋值
strcmp 函数----字符串比较函数
一般形式: strcmp(字符串 1，字符串 2)
作用:比较字符串 1 和字符串 2
strcmp(str1,str2);

strcmp(”China”,”Korea”);
strcmp(str1,”Beijing”);
比较的结果由函数值带回
如果字符串 1=字符串 2，则函数值为 0
如果字符串 1>字符串 2，则函数值为一个正整数
如果字符串 1<字符串 2，则函数值为一个负整数
注意：
字符串比较不能用“==”、“<”、 “>等关系运算符直接比较，要用 strcmp 函数进行比较
if(“ad”>”ac”) printf(“ad”);
strlen 函数----测字符串长度的函数
一般形式：strlen (字符串)
作用: 它是测试字符串长度的函数
函数的值为字符串中的实际长度,它的返回值是字符串中字符的个数（不包含’\0’）
strlwr 函数----转换为小写的函数
一般形式: strlwr (字符串) 函数的作用: 将字符串中大写字母换成小写字母
strupr 函数----转换为大写的函数
一般形式: strupr (字符串) 函数的作用: 将字符串中小写字母换成大写字母

六、用函数实现模块化程序设计

1、可以使用哪些函数？
一个 C 程序由两种函数构成：
标准函数（库函数）：
系统提供的，实现各种不同的功能。如 printf, scanf, sqrt, fabs ， strlen,等等
使用它们要包含相应的头文件
自定义函数：
用户自己定义的函数。
用户可以把一些具有独立功能的代码定义成函数，有利于程序的模块化和代码的“重用”
2、说明
（1）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
（2）一个源程序文件由一个或多个函数以及其他有关内容（如预处理指令、数据声明与定义等）组成
（3） C 程序的执行是从 main 函数开始的，如果在 main 函数中调用其他函数，在调用后流程返回到 main 函数，在 main 函数中结束整个程序的运行。
（4）所有函数都是平行的，即在定义函数时是分别进行的，是互相独立的。即函数不能嵌套定义
3、定义函数
（1）C 语言要求，在程序中用到的所有函数，必须"先定义，后使用"
指定函数的名字，以便以后按名调用
指定函数类型，即函数返回值的类型
指定函数参数的名字和类型，以便在调用函数时向它们传递数据
指定函数的功能。这是最重要的，这是在函数体中解决的
（2）定义方法
[函数的类型] 函数名称（ 形式参数列表 ） 函数首部
{
数据说明部分

可执行语句　　函数体

}

定义无参函数

定义无参函数的一般形式为:

类型名　函数名( )　　类型名　函数名 (void)

{　　{

函数体　　函数体

}　　}

定义有参函数

定义有参函数的一般形式为:

类型名 函数名(形式参数表列)

{

函数体

}

4、函数的调用

调用一般形式　函数名(实参表)

（1）　函数调用语句

把函数调用单独作为一个语句，如 printf_star()；这时不要求函数带回值，只要求函数完成一定的操作

（2） 函数表达式

函数调用出现在另一个表达式中，如

c=max(a,b)+2;

m = jiecheng(4)*2;

这时要求函数带回一个确定的值以参加表达式的运算

（3） 函数参数

函数调用作为另一函数调用时的实参，如

m＝max(a,max(b,c));

printf(“%f”,jiecheng(9));

其中 max(b,c)是一次函数调用，它的值作为 max 另一次调用的实参

当用不到函数的返回值时用函数调用语句　当要使用函数的返回值时用函数表达式或函数参数

（4）函数调用时的数据传递

①.形式参数和实际参数

形参（形式参数）是函数定义时，参数表中的参数。形式参数只能是变量。

例如：float　max(float　x，　float y)

{　•••　　}

float ftoc (float　temp)

{　•••　　}

在函数定义时，参数表中的形参并没有具体的值，系统也不为其分配存储单元。

实参（实际参数）是函数调用时主调函数传送给被调用函数形参的实际值。 实参可以是常量、变量和表达式，实参必须有确定的值。

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用函数的形参，或者说，形参从实参得到一个值，该值在函数调用期间有效，可以参加被调函数中的运算，从实参到形参的值传递是

单向的！没有返回的过程！**单向值传递！**
实、形参个数相同，类型应一致（相同或赋值兼容）。
例：int n;
n=max(4,20);
n=max(3.5,7.3); 转化为：n=max(3,7);
（5）函数的调用过程
在定义函数中指定的形参，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，函数的形参被临时分配内存单元。
调用结束，形参单元被释放实参单元仍保留并维持原值，没有改变
如果在执行一个被调用函数时，形参的值发生改变，不会改变主调函数的实参的值!
（6）函数的返回值
希望通过函数调用使主调函数能得到一个确定的值，这就是函数值(函数的返回值)
函数的返回值是通过函数中的 return 语句获得的 return 语句后面的括号可以不要
一个函数中可以有一个以上的 return 语句，执行到哪一个 return 语句，哪一个语句起作用
return 后面的值可以是一个表达式。
例如: max(int x，int y)
{ return(x > y ？ x : y); }
在定义函数时指定的函数类型一般应该和 return 语句中的表达式类型一致。
如果函数值的类型和 return 语句中表达式的值不一致，则以函数类型为准。
对数值型数据，可以自动进行类型转换。即函数类型决定返回值的类型。
5、对被调用函数的声明和函数原型
（1）在一个函数中调用另一个函数需要具备如下条件：
被调用函数必须是已经定义的函数（是库函数或用户自己定义的函数）
如果使用库函数，应该在本文件开头加相应的#include 指令
如果使用自己定义的函数，而该函数的位置在调用它的函数后面，应该进行函数声明
（2） 函数声明
作用：
告诉编译系统函数类型、参数个数及类型，以便检查
形式：
函数类型 函数名（类型 1 形参 1，类型 2 形参 2， …..）；
函数类型 函数名（类型 1，类型 2，…..）；
函数类型 函数名（）； 传统方法
函数声明位置：
主调函数内部开头
或整个文件的开头（所有函数之前）
6、函数的嵌套调用
7、函数的递归调用
（1）直接调用本函数 （2）间接调用本函数 应使用 if 语句控制结束调用
8、数组作为函数的参数
除了可以用数组元素作为函数参数外，还可以用数组名作函数参数(包括实参和形参)
用数组元素作实参时，向形参变量传递的是数组元素的值,单向值传递
用数组名作函数实参时，向形参传递的是数组首元素的地址
说明：
在主调函数与被调函数分别定义数组,且类型应一致

形参数组大小可不指定：C 编译系统不检查形参数组的大小
最好设计一个参数，来指定形参数组大小，也方便确定数组元素个数
数组名做参数，参数传递的是地址：形参数组名只是获得了实参数组的首地址
注意:
数组名做函数参数时,可以用来存储函数的返回值。
当一个函数有多个同类型的结果时，可用数组做参数来保存结果。
9、局部变量和全局变量
（1）局部变量
- 在一个函数内部定义的变量只在本函数范围内有效
- 在复合语句内定义的变量只在本复合语句范围内有效
- 在函数内部或复合语句内部定义的变量称为“局部变量”

说明：
- main 中定义的变量，只在 main 中有效。
- 函数形参是函数的局部变量。
- 不同函数中可以定义同名的变量，它们互不干扰。

（2）全局变量
在函数内定义的变量是局部变量，而在函数之外定义的变量称为外部变量，外部变量是全局变量(也称全程变量)全局变量可以为本文件中其他函数所共用；有效范围为从定义变量的位置开始到本源文件结束
全局变量没有赋初值时，系统自动赋为 0
同一个.c 文件中，全局与局部变量同名时，外部变量被屏蔽，即局部优先。
10、变量的存储方式和生存期
从变量值存在的时间(即生存期)观察，变量的存储有两种不同的方式：静态存储方式和动态存储方式
静态存储方式是指在程序运行期间由系统分配 固定的存储空间的方式
动态存储方式是在程序运行期间根据需要进行动态的分配存储空间的方式
程序开始执行时给全局变量分配存储区，程序执行完毕就释放。在程序执行过程中占据固定的存储单元
函数调用开始时分配，函数结束时释放。在程序执行过程中，这种分配和释放是动态的
不要误认为对外部变量加 static 声明后才采取静态存储方式，而不加 static 的是采取动态存储
声明局部变量的存储类型和声明全局变量的存储类型的含义是不同的
对于局部变量来说，声明存储类型的作用是指定变量存储的区域以及由此产生的生存期的问题，而对于全局变量来说，声明存储类型的作用是变量作用域的扩展问题
用 static 声明一个变量的作用是：
对局部变量用 static 声明，把它分配在静态存储区，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。
对全局变量用 static 声明，则该变量的作用域只限于本文件模块(即被声明的文件中)。
注意：用 auto、register、static 声明变量时，是在定义变量的基础上加上这些关键字，而不能单独使用。