第六次实验报告

PB16000702 韦璐

实验目的：

1. 函数的定义与调用方法
2. 函数形参和实参的数据传递
3. 函数的类型与返回值
4. 函数的说明和声明
5. 函数的递归调用
6. 变量的存储类型
7. 全局变量和局部变量
8. 宏定义
9. 文件包含

一．函数的定义

函数定义的一般形式：

类型标识符 函数名（形参类型说明表列）{

数据定义说明部分；

执行语句部分；

}

（1） 类型标识符确定函数返回值的类型，所以应与函数体内的 return 语句返回值的类型相同。它可以是任何一种有效的类型，当函数类型缺省时为整型，当函数无返回值时可以用 void 类型标识符说明。

（2） 函数名必须符合 C 语言规定的标识符命名规则，且函数名必须唯一，不能与函数体内变量或形式参数同名。

（3） 形参类型说明表列中的形参用于接受主调函数传递过来的相应实参的值。若函数无需接受传递过来的数据，则形参类型说明表列可以为空，但函数名后的圆括号不能省略。

（4） 如果形参传递类型说明表列中的形参个数多于一个，则每一个形参都需要单独说明，并且用逗号分隔，构成“(类型标识符 1 形参 1，类型标识符 2 形参 2，……)”形式。

如果函数定义形式采用如下的传统方式：

类型标识符 函数名（形参表列）

形参类型说明

{

数据定义说明部分；

执行语句部分；

}

则形式参数无需一个一个的单独说明，即同类型元素可在相应类型标识符后面用逗号分离一起说明，例如：

```
Double fun1(x,y,n)
Double x,y;int n;
{
Return(x+y+n);
}
```

（5） 函数的定义在程序中是平行的，不允许函数的嵌套定义，即不能在函数的内部

定义另一个函数。

（6） 函数内定义的变量只在函数内部有效，因此不同函数中变量的名字可以相同，互不干扰。

2. 函数的调用

函数调用时，要求实参与形参在个数、顺序、类型和单位上必须对应一致。

函数调用的一般形式为：函数名{实参表列};

（1） 函数的调用应遵循先定义后使用的调用原则，若函数定义在后，调用在先，则应在主调函数的定义说明部分给出被调函数原型的说明。说明形式如下：

类型标识符 函数名（形参类型说明表列）;

注意正确区分函数定义和函数说明：

1. 定义是对函数功能的确立
2. 说明是对已定义函数的函数名，函数类型，形参的个数、顺序和类型等信息进行申明。

(2) 如果实参表列中含有多个参数，则各参数应用逗号分隔。若实参表列为空，则可以省略，但函数名后的括号不能省略。

（3）函数调用可以出现在任何允许表达式出现的地方，也可以作为独立的函数调用语句。

（4）除主函数外，其他函数可以嵌套调用，也可以递归调用。

3. 函数参数的两种传递方式

（1）传值方式（单向传递），形参可以是简单变量，实参可以是常量，变量，表达式，数组元素等。特点是：形，实参数一旦传递结合，即数据传递完毕，实参与形参不再有任何实际联系，也即形参在函数中的值的变化不会改变主调函数中实参的原有值。

（2）传地址方式（双向传递）：形参可以实数组名和指针变量，实参可以是数组名，指针变量，字符串常量，指定单元的起始地址等。特点是：通过传递数组或某一存储单元的起始值，使被调函数可利用此地址来访问（存放）相应存储单元的数据，实质上为通过存储单元共享达到数据双向传递的目的。

4. 函数的返回值与函数类型说明

函数的类型应与 return 中的表达式类型一致，函数类型决定返回值的类型。

（1） 若 return 后面括号中的表达式为非整型，则必须在函数名前冠以函数的类型说明。因为函数类型决定返回值的类型。

（2） 若无返回值，可用 VOID 类型定义函数。

（3） 一个函数可以有多个 return 的语句。形式可为：

1. Return（表达式）：//带值返回
2. Ruturn；//不带值返回

（4） 在 UNIX 系统，一个函数若无 return 语句，则遇最外层'}'即返回。

5. 函数间的数据联系

函数间的数据联系可通过以下三个途径：

（1） 通过 return 语句返回一个值（数值或地址）/

（2） 形参传递结合（传值方式或传地址方式）。

（3） 通过外部（全局）变量进行数据传递。

6. 函数的递归调用

（1）C 语言的函数除了 main（）函数，都可以递归调用，即可以直接或间接地自己调用自己。

递归调用必须在满足一定条件时可以结束递归调用，否则无限地递归调用将导致程序无法结束。构成递归的条件：

1. 确定的递归结束条件及结束时的值。
2. 能用递归表达式形式表示，并且递归向结束条件发展。

7. 变量的存储类别

C 程序中，每一个变量和函数都有数据类型和存储类别两个属性：数据类型确定数据的存放形式，值域及允许的运算等，存储类别确定数据的存储区域等。

完整的变量定义形式：

存储类别 数据类型 变量名表列;

完整的函数定义形式

存储类别 数据类型 函数名（形参表）{‘‘‘‘}

（1） C 语言说明存储类别的关键字有：

auto （自动）：限定说明每次进入函数时自动分配存储单元，退出函数时系统自动收回存储单元的局部变量，分配动态存储区，数据随着函数的调用而存在，随函数的返回而消失（存储单元回收）

register （寄存器）：限定说明寄存器变量

static（静态）：限定说明局部静态变量，局部静态数组，静态外部变量，数据的生存周期同程序运行期，即变量可以保持到程序运行结束。

extern（外部）：限定说明全局变量，用户程序区。

（2） 函数内定义的变量，若不指定存储类别，则作 auto 处理。函数外定义的变量为全局变量。

（3） 在定义和说明变量的时候，应按照变量的作用范围以及它们在存储单元中保持值的时间长短，对它们的存储类别进行说明限定。

（4） 函数的存储类别只能使用 extern 和 static 两个限定词，以限定函数的作用范围，省略存储类型隐指外部外部函数，可供其他文件中的函数调用，静态函数只能被本程序文件的函数调用；静态函数只能被本程序文件的函数调用。

8. 局部变量及其作用域

函数内部定义的变量通称局部变量，包括形参变量，由于存放在动态区，故也称为动态变量或自动变量。局部变量只有在所属函数别调用时系统才自动为其分配存储单元，当退出作用域后其值不做保留。

（1） 局部变量的作用域：从定义的位置起到函数体（或复合语句）结束止，即函数内定义的变量只限函数内有效，符合语句内定义的变量只限本符合语句内有效，其他函数或语句不能使用。Main（）函数中定义的变量也只有主函数内有效。

（2） 不同函数中定义的变量可以同名，代表不同的对象，互不干扰。

（3） 形参也是局部变量

（4） 没有指定类别做 auto 处理

（5） 内层定义的局部变量你优先

（6） 局部动态变量的初值不定。

10. 寄存器变量及其作用域

为了提高执行效率，允许将局部变量的值放在 CPU 的寄存器中，需要时直接从寄存器取出参加运算。只有局部自动变量和函数的形参可以作为寄存器变量，并且类型只限于 int，char 和指针类型。

11. 外部变量及其作用域

在函数外定的变量称为外部变量，或全局变量，可以为本文件中其他函数公用。也可以使用 static 和 extern 两种关键字进行说明。

外部变量的有效作用域：从定义的变量的位置开始到本源文件结束，即全局变量可以被

在其定义位置之后的函数所共享。因此，外部变量为不同函数间进行数据通信（传递）提供了另一途径。

外部说明：一个 C 程序可以由几个不同的源程序文件组成，如果组成这个程序的几个文件需要用到同一个全局变量，或者在顶一点之前的函数中想引用某外部变量，则应该在其他引用该全局变量的源程序文件中或想换函数中用关键字，extern 做“外部变量“说明，形式：

Extern 类型标识符 变量表列；

静态说明：如果一个源程序的全局变量只能该文件使用，则只要在该全局变量定义是的类型说明加个 static 即可：

Static 类型标识符 变量表列；

使用全局变量会使函数的通用性及程序清晰度降低，要限制使用。

12. 静态变量及其作用域
    （1） 函数体内用 static 存储类别定义的变量为静态局部变量。静态局部变量的生存期是整个程序的运行期，即在程序的整个执行过程主流观念始终存在，但是在作用域之外不能使用。
    （2） 注意动态局部变量与静态局部变量的区别。

    动态：

    1. 作用域在空间上是函数或复合语句的内部，在时间上是函数的一次调用周期或复合语句的执行周期；即进入函数时系统自动为其分配存储单元，函数调用结束时系统自动收回该存储单元。
    2. 初值不定

    静态局部变量：

    1. 其值不随函数调用或复合语句的执行结束而消失，也就是当再次进入函数或复合语句时，这些局部变量仍能保持上次函数调用时的值；也可以说静态局部变量在编译时只分配一次存储单元，并一直保持到程序执行结束。
    2. 具有确定的初值，数值为 0，字符为‘\0’
    3. 静态局部变量的作用域不变，仍为本函数内有效。
13. 宏定义
    （1） 宏定义分为不带参数的宏定义和带参数的宏定义。
        1. 不带参数的宏定义格式：

           #define 宏名 字符串

           它的作用是在编译预处理时，将源程序中所有宏名替换成字符串。
        2. 带参数的宏定义格式：

           #define 宏名（参数表） 字符串

           它的作用是在编译预处理时，将源程序中所有宏名替换成字符串，并且将字符串的参数用实际实际使用的参数替换。注意：宏名和参数表之间不能有空格，否则空格后面的所有字符序列都作为替换的字符串。
    （2） 宏名习惯用大写字母，以与变量区别。
    （3） 宏定义就是用一个共鸣代替一个字符串，简化编程。
    （4） 对宏定义的预处理就是做逆向的置换。
    （5） 可以层层置换，先定义的可被后者引用。
    （6） 带参数的宏定义在预处理时，也值作字符和参数的替换，不进行任何计算操作，只有在编译后运行时，才按照替换后的表达式运算规则进行运算。所以，一般在定义宏时，字符串中的形式参数外面应加一个小括号。

14. 文件包含

文件包含预处理命令的一般形式:

#include<filename> 或 #include"filename"

(1) 系统处理是将到包含 C 的库函数的头文件所在的目录（通常时 include 的子目录中寻找文件。后一种文件名用双引号括起来的形式，系统在处理时先在当前的目录下寻找，若找不到，则再到操作系统的 path 命令设置的自动搜索路径中查找，最后才到 C 语言头文件所在的目录中寻找文件。

(2) 一个#include 命令只能指定一个被包含文件。

(3) 若文件 1 包含文件 2，文件 2 又要用到文件 3 的内容，则可在文件 1 中引用两个 include 命令分别包含文件 2 和文件 3，次序 2 3

以上情况也可分别包含，即一个被包含文件又可以分别包含另一个被包含文件。

15. 集中常见条件编译

条件编译命令允许对程序中的内容做选择性地编译，基可以根据一定的条件选择是否编译。

（1） 条件编译形式 1:

```
#ifdef 标识符
    程序段 1
#else
    程序段 2
#endif
```

条件编译形式 1 的作用是：当“标识符“已经有#define 定义过了，则编译”程序段 1”，否则编译“程序段 2”，其中”#else 程序段 2“可以缺省。

（2） 条件编译形式 2:

```
#ifdef 标识符
   程序段 1
#else
程序段 2
#endif
条件编。。。。
```

范例

1. 输入并运行下列程序，分析运行结果

实验代码:

```
#include<stdio.h>
main(){
 int fun1(int x[],char y[]);
 static int a[6]={10,20,30,40,50},i;
 static char b[6]="USTC";
 fun1(a,b);
 printf("\na:");
 for(i=0;i<6;i++)printf("(%x)%-4d",&a[i],a[i]);
 printf("\nb:");
 for(i=0;i<6;i++)printf("(%x)%-4c",&b[i],b[i]);
 printf("\nb:");
```

```
    for(i=0;i<6;i++)printf("(%x)%-4d",&b[i],b[i]);
    printf("\nString:%s\n",b);
   }
   int fun1(int x[],char y[]){
    int i;
    printf("\nx:");
    for(i=0;i<6;i++)printf("(%x)%-4d",&x[i],x[i]);
    printf("\ny: ");
    for(i=0;i<6;i++)printf("(%x)%-4c",&y[i],y[i]);
    printf("\ny: ");
    for(i=0;i<6;i++)printf("(%x)%-4d",&y[i],y[i]);
    printf("\nString:%s\n",y);
    x[5]=x[0]+x[2]+x[3]+x[4]+x[5];
    return 0;
}
```

调试分析：这个函数是为了让我们体会到函数通过调用动态局部变量也可以实现主函数的效果。在运行之后，发现括号里的内容和书上的不一样，分析应该是计算机不同，分配的存储地址不同，但是差值是一样的，都是四个字节。

实验结果：

```
x:(403010)10  (403014)20  (403018)30  (40301c)40  (403020)50  (403024)0
y: (403028)U   (403029)S   (40302a)T   (40302b)C   (40302c)    (40302d)
y: (403028)85  (403029)83  (40302a)84  (40302b)67  (40302c)0   (40302d)0
String:USTC

a:(403010)10  (403014)20  (403018)30  (40301c)40  (403020)50  (403024)130
b:(403028)U   (403029)S   (40302a)T   (40302b)C   (40302c)    (40302d)
b:(403028)85  (403029)83  (40302a)84  (40302b)67  (40302c)0   (40302d)0
String:USTC
```

实验总结：

无

第二题

输入年，月，日，输出该日期是该年的第几天。

实验代码：

```
#include<stdio.h>
int is_leap_year(int year){
    int leap;
    if(year%4==0&&year%100!=0||year%400==0)
    leap=1;
    else
    leap=0;
    return(leap);
}
int len_of_month(int year,int month){
    int month_days;
    if(month==2)
    if(is_leap_year(year))
```

```
        month_days=29;
        else
        month_days=28;
        else if(month==4||month==6||month==9||month==11)
        month_days=30;
        else
        month_days=31;
        return(month_days);
}
int len_of_days(int year,int month,int date){
        int total_days,n;
        for(n=1,total_days=0;n<month;n++)
        total_days+=len_of_month(year,n);
        total_days+=date;
        return(total_days);
}
main(){
        int year,month,date,days;
        printf("please input your,month,date:");
        scanf("%d%d%d",&year,&month,&date);
        days=len_of_days(year,month,date);
        printf("%d.%d.%d is the %d in %d.\n",year,month,date,days,year);
}
```

实验结果：

```
please input your,month,date:2000 3 1
2000.3.1 is the 61 in 2000.
```

调试分析：

没有

实验总结：

这个是在主函数中调用一个函数来计算，然而这个函数可以引用更多的东西，所以一层层嵌套就可以计算出结果，而且关键是这个不需要在主函数的开头定义什么东西。

第三题

题目：

编制一个程序，把输入中包含指定字或字段，或者说指定“字符串“的那些行打印出来。假定指定的字符串为”the“，输入的以下四行：

Now is the time

For all good

Men to come to the end

Of their party

通过程序的查找，将产生下面的输入行：

Now is the time

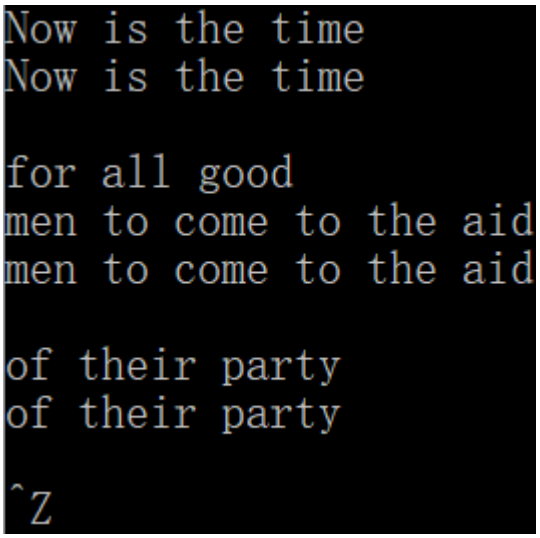
Men to come to the aid

Of their party

程序的设计思想如下：
While(是否还有输入行)
If（该行中有指定字符串）
打印该行
其中，“是否还有输入行“的判断功能可通过函数中的 getline 函数完成，所以这里主要考虑编制”该行中是否有指定字符串“的函数（求字串函数）。假定被寻找指定字符串的行在形参字符数组 s 中，指定字符串在字符数组 t 中。该函数的功能是判断在字符数组 s 中，是否有指定的字符串 t，若有则返回指定字符串 t 在 s 字符串中的开始位置；若 s 中不包含 t 时，返回值为-1。
程序代码：

```
#define MAXLINE 1000
#include<stdio.h>
main(){
	int getline(char s[],int lim);
	int index(char s[],char t[]);
	char line[MAXLINE];
	while(getline(line,MAXLINE)>0)
	if(index(line,"the")>=0)
	printf("%s\n",line);
}
int getline(char s[],int lim){
	int i,c;
	i=0;
	while(--lim>0&&(c=getchar())!=EOF&&c!='\n')
	s[i++]=c;
	if(c=='\n')
	s[i++]=c;
	s[i]='\0';
	return(i);
}
index(char s[],char t[]){
	int i,j,k;
	for(i=0;s[i]!='\0';i++){//在 s 结束之前我们以 s 的结尾为判断条件，然后循环变量在 s 结束前进行判断
		for(j=i,k=0;t[k]!='\0'&&s[j]==t[k];j++,k++);//用新的两个循环变量来检查 st 中是否有相同的单词，在 t 的元素结束之前
		if(t[k]=='\0') return(i);//这里有一个非零的判断条件，然后就在满足这个条件之后直接跳出循环
	}
	return(-1);//这里满足程序要求
}
```

运行结果：



调试分析:

有一个警告

实验总结:

是经典的代码

第四题:

题目: 计算多项式。

实验代码:

```
#include"C:\Apps\bb.cpp"
main(){
	float y,x;
	float f(float x);
	input_a();
	printf("Enter x:");
	scanf("%f",&x);
	y=f(x);
	printf("y=%g\n",y);
}
float f(float x){
	int i;
	float y;
	y=a[N]*x;
	for(i=N-1;i>0;i--)
	y=x*(a[i]+y);
	y=a[0]+y;
	return(y);
}
#include<stdio.h>
#define N 20
static float a[N+1];
float f(float x);
void input_a(){
```

```
    int i;
    printf("Please Enter a0--an:");
    for(i=0;i<N+1;i++)
    scanf("%f",&a[i]);
}
```

实验结果：

```
Please Enter a0--an:1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20 21
Enter x:2.0
y=4.1943e+007
```

调试分析：

只有英文的途径才能额外引用

实验总结：

%f　表示按浮点数的格式输出

%e　表示按指数形式的浮点数的格式输出

%g　表示自动选择合适的表示法输出

实验题目：

1. 写一个判断素数的函数，如果参数为素数，则返回 1，否则返回 0.在主函数输入一个整数，输出是否为素数的判断结果。

实验代码：

```
#include<stdio.h>
int isss(int num)//判断是否是素数  是返回 1  否返回 0
{
    int i;
    if(num==0||num==1)//零和 1 都不是素数
        return 0;
    for(i=2;i<num;i++)
    {
        if(num%i==0)//如果在比自己小的大于一的数中有可以把自己整除的数，那么这个数就不是素数了
            return 0;
    }
    return 1;

}
main(){
 int i;
 printf("shuru:");
 scanf("%d",&i);
 if(isss(i))
```

```
            printf("%d 是素数\n",i);
        else
            printf("%d 不是素数\n",i);
    }
```

实验结果:

```
shuru:7
7 是素数
```

调试分析：无

实验总结：无

第二题：

求两个整数的最大公约数和最小公倍数。要求编写两个函数，一个函数求最大公约数，另一个函数根据求出的最大公约数求最小公倍数。建议使用以下两种数据传递方法分别编程实现：

（1） 不用全局变量，两个整数在主函数输入，并以参数形式传递给函数 1，求出的最大公约数返回主函数，然后再与两个整数一起作为实参传递给函数 2，以求出最小公倍数，返回到主函数输出最大公约数和最小公倍数。

（2） 用全局变量的方法。将两个整数的最大公约数，最小公倍数设为全局变量。

实验代码：

```
#include <stdio.h>
int HCF(int x, int y);                              //定义最大公约数函数
int LCM(int p, int q);                              //定义最小公倍数函数
int main()
{
    int a, b, hcf, lcm;
    scanf("%d %d", &a, &b);                         //输入两个整数
    hcf = HCF(a, b);                                //调用最大公约数函数
    lcm = LCM(a, b);                                //调用最小公倍数函数
    printf("HCF is %d  LCM is %d\n", hcf, lcm);     //输出最大公约数和最小公倍数
    return 0;
}
int sum;                                            //定义外部变量 sum
//最大公约数函数
int HCF(int x, int y)
{
    int i, k, m, n;
    sum = 1;
    k = x > y ? y : x;
```

```
    i = 2;
    while (i <= k){//从 2 开始的数用来除这两个数，然后如果成功了，那就约掉这个数然后将除数重新置 2，如果失败了，那就加 1 再继续除法
        m = x % i;
        n = y % i;
        if (m == 0 && n == 0){
            sum *= i;
            x /= i;
            y /= i;
            i = 2;
        }
        else
            i++;
    }
    return sum;
}
extern int sum;                                          //引用外部变量 sum
//最小公倍数函数
int LCM(int p, int q)
{
    int lc;
    lc = p * q / sum; //有了最小公倍数以后，就可以算最大公约数了
    return lc;
}
```

实验结果：

```
32 34
HCF is 2  LCM is 544
```

调试分析：

无

实验总结：

使用全局变量进行函数计算的时候需要在函数的前面进行声明，并且声明的结构要牢记。

第三题

题目：求组合数和排列数，计算公式已经给出。要求该程序包含两个自定义函数用于完成相应功能。

实验代码：

```
#include<stdio.h>
int factorial (int x){
    int i;
    int sum=1;
    for(i=1;i<=x;i++){
        sum*=i;
    }
```

```
    return(sum);
}
int c_p_mm(int m,int n){
    int pailieshu,zuheshu,i;
    printf("想计算排列数，输入 1，想计算组合数，输入 2\n");
    scanf("%d",&i);
    if(i=2){
    zuheshu=factorial(m)/(factorial(n)*factorial(m-n));
    return(zuheshu);
    }
    if(i=1){
    pailieshu=factorial(m)/(factorial(m-n));
    return(pailieshu);
    }
}
main(){
    int m,n,i;
    scanf("%d%d",&m,&n);
    i=c_p_mm(m,n);
    printf("计算结果为：%d\n",i);
}
```

实验结果：

```
9 3
想计算排列数，输入 1，想计算组合数，输入 2
1
计算结果为：84
```

调试分析：

无，可以更加人性化

实验总结：

无

第四题

题目：

编写函数，其功能是在 float 类型的一维数组中查找最大值，最小值并将它们返回到调用程序。

实验代码：

```
#include <stdio.h>
void fun(float a[],float *mx,float *mi)
{
    int i=0;
    for (i=0;i<10;i++)
    {
        if (i==0)
        {
            *mx = *mi = a[i];
```

```
        }else
        {
            if (*mx<a[i])
            {
                *mx = a[i];
            }
            if (*mi >a[i])
            {
                *mi = a[i];
            }
        }
    }
}


main()
{
    int i=0,n=0;
    float a[10]={0},maxf=0,minf=0;
    scanf("%d",&n);
    for (i=0;i<10;i++)
    {
        scanf("%f",&a[i]);

    }
    fun(a,&maxf,&minf);
    printf("MAXF[%f] MINF[%f]\n",maxf,minf);

}
```

实验结果：

```
10
9
8
7
6
5
4
3
2
1
0
MAXF[9.000000] MINF[0.000000]
```

第五题

题目：写一个函数，使输入的一个字符串按反序存放，在主函数中输入和输出字符串。
实验代码：

```
#include<stdio.h>

#include<string.h>

void input(char s[])

{

    gets(s);

}



void show(char s[])

{

    puts(s);

}



void FanXuCunFang(char s[])//接收首地址，不过要加[]

{

    int i,n;

    n=strlen(s);//得到数组的长度

    char s_temp[100];

    for(i=0;i<n;i++)

    {

        s_temp[i]=s[n-i-1];//倒放
```

```
    }

    for(i=0;i<n;i++)
    {
        s[i]=s_temp[i];//放回去
    }
}

int main()
{
    char s[100];
    printf("请输入非中文字符串：\n");
    input(s);
    printf("您输入的是：\n");
    show(s);
    printf("反序存放……\n");
    FanXuCunFang(s);//输入一个地址
    printf("反序存放后输出：\n");
    show(s);
    return 0;
}
```

实验结果：

请输入非中文字符串：

njgr

您输入的是：
njgr
反序存放……
反序存放后输出：
Rgjn
调试分析:无
实验总结：无
第六题
题目：编写一个函数，由实参传来一个字符串，统计此字符串中字母，数字，空格和其他字符的个数，在主函数中输入字符串并输出统计后的结果。
实验代码：

```
#include<stdio.h>

void TongJi(char s[])//用来统计的函数，得到的参数是输入的数组的首地址

{

    int ZiMu=0,KongGe=0,ShuZi=0,QiTa=0,i;

    for(i=0;s[i]!='\0';i++)//遍历 s 中的元素

    {

        if(s[i]==32)KongGe++;//这是空格的 ascii 码

        else if((s[i]>=48)&&(s[i]<=57))ShuZi++;//这是十个数字的 ascii
码

        else
if(((s[i]>=97)&&(s[i]<=122))||((s[i]>=65)&&(s[i]<=90)))ZiMu++;//不管
是大写的字母还是小写的字母，都在 zimu 下面加上 1

        else QiTa++;//如果不是上面任何一个，就在其他那里加 1

    }

    printf("空格：%d；数字：%d；字母：%d；其他：%d。
\n",KongGe,ShuZi,ZiMu,QiTa);

}

int main()

{
```

```
    char s[100];

    printf("请输入：");

    gets(s);

    TongJi(s);

    return 0;

}
```

实验结果：

```
请输入：hmha;,.;'/  ;;2%^&%&&*$
空格：2；数字：1；字母：4；其他：16。
```

调试分析：

无

实验总结：

记住各个 ascii 码

第七题

题目：输入 10 个学生 5 门课的成绩，分别用函数实现下列功能：

（1） 计算每个学生的平均分。

（2） 计算每门课程的平均分。

（3） 找出所有 50 个分数中最高分所对应的学生和课程。

（4） 计算平均分方差。

实验代码：

```
#include <stdio.h>
#include <math.h>
void aver_stu(int t[][5]);                  //定义学生平均分函数
void aver_course(int t[][5]);               //定义课程平均分函数
void high(int t[][5]);                      //定义最高分函数
void vari(int t[][5]);                      //定义方差函数
int main()
{
    int stu[10][5];
    int i, j;
    for (i=0; i<10; i++)
        for (j=0; j<5; j++)
            scanf("%d", &stu[i][j]);        //输入 10 个学生各 5 门课的成
绩
    aver_stu(stu);                          //调用学生平均分函数
    aver_course(stu);                       //调用课程平均分函数
    high(stu);                              //调用最高分函数
    vari(stu);                              //调用方差函数
```

```
    return 0;
}
//学生平均分函数
void aver_stu(int t[][5])
{
    int i, j;
    float k, ave;
    for (i=0; i<10; i++){
        for (j=0, k=0.0; j<5; j++)
            k+=t[i][j];
        ave=k/5;
        printf("No.%d student average is %f\n", i+1, ave);
    }
}
//课程平均分函数
void aver_course(int t[][5])
{
    int i, j;
    float k, ave;
    for (j=0; j<5; j++){
        for (i=0, k=0.0; i<10; i++)
            k+=t[i][j];
        ave=k/10;
        printf("No.%d course average is %f\n", i+1, ave);
    }
}
//最高分函数
void high(int t[][5])
{
    int i, j, h, stu, cour;
    for (i=0, h=0, stu=0, cour=0; i<10; i++){//赋初值
        for (j=0; j<5; j++)
            if (t[i][j]>h){
                h=t[i][j];//全部遍历，如果有更高的就赋值过去
                stu=i+1;//赋值后还要给这两个量赋值方便提取出这个最高
成绩的出处
                cour=j+1;
            }
    }
    printf("The highest score is %d, from No.%d student & No.%d
course\n", h, stu, cour);
}
//方差函数
void vari(int t[][5])
```

```
{
    int i, j, k, m;
    float temp[10], var, x1, x2;
    for (i=0, m=0; i<10; i++, m++){
        for (j=0, k=0; j<5; j++)
            k+=t[i][j];
        temp[m]=k/5;//每个学生求平均值
    }
    for (i=m=x1=x2=0; i<10; i++){
        x1+=pow(temp[i], 2);//平均值的平方
        x2+=temp[i];//所有平均成绩的总和
    }
    var=x1/10-pow(x2/10, 2);//方差的计算
    printf("The variance is %f\n", var);
}
```

实验结果：

```
1 2 3 4 5
6 5 4 3 7
8 9 0 7 6
5 6 7 8 5
9 8 3 4 6
1 2 6 5 4
7 8 6 5 9
7 7 7 8 7
6 6 6 7 5
7 8 6 9 5
No.1 student average is 3.000000
No.2 student average is 5.000000
No.3 student average is 6.000000
No.4 student average is 6.200000
No.5 student average is 6.000000
No.6 student average is 3.600000
No.7 student average is 7.000000
No.8 student average is 7.200000
No.9 student average is 6.000000
No.10 student average is 7.000000
No.11 course average is 5.700000
No.11 course average is 6.100000
No.11 course average is 4.800000
No.11 course average is 6.000000
No.11 course average is 5.900000
The highest score is 9, from No.3 student & No.2 course
The variance is 2.040003
```

调试分析：

无
实验总结：无
第八题：写一个函数验证哥德巴赫猜想，哥德巴赫猜想是说：一个不小于 6 的偶数可以表示为两个素数之和。在主函数中输入一个不小于 6 的偶数 n，函数中输出类似的结果。
程序代码：

```
#include <iostream>
using namespace std;
void gobaha(int n);                    //定义函数 gobaha
int prime(int n);                      //定义 prime 函数
int main()
{
    int num;
    cout<<"Please enter number: ";
    cin>>num;
    for (; num<6||(num%2)!=0; ){                          //如果输入数字不符合条件重复执行输入
        cout<<"Error! Retry!\nPlease enter number: ";
        cin>>num;
    }
    gobaha(num);
    system("pause");
    return 0;
}
int prime(int n)
{
    int i;
    for (i=2; i<n&&(n%i!=0); i++);
    return n==i ? 1 : 0;                          //判断是否为素数，是返回 1，否返回 0
}
void gobaha(int n)
{
    int m, i;
    for (i=3; i<n; i++)
        if (prime(i)==1){
            m=n-i;
            if (prime(m)==1&&i<=m)//避免重复
                cout<<n<<'='<<i<<'+'<<m<<endl;
        }
}
```

实验结果：

```
Please enter number: 43
Error! Retry!
```

```
Please enter number: 34
34=3+31
34=5+29
34=11+23
34=17+17
请按任意键继续. . .
```
调试分析：
无
实验总结：
无
第十一题
题目：定义一个带参数的宏，功能是两个参数的值互换。在主程序中输入两个数作为使用宏的实参，输出已经交换后的两个值。
实验代码：
```
#include<stdio.h>

#define swap(a,b) t=a;a=b;b=t;

int main()
{
    int x,y,t;
    scanf("%d%d",&x,&y);
    swap(x,y);
    printf("%d %d",x,y);
    return 0;
}
```
运行结果：
```
3 4
4 3
```
调试分析：无
结果分析：就是在使用这个宏以后将参数替换，那么结果也会将参数替换以后将表达式原样带入其中。
第 12 题：
题目：阅读以下程序并给出结果。然后上机运行该程序验证阅读结果，如与运行结果不符合，请分析原因。
实验代码：
```
#include<stdio.h>
#define PRODUCT(a,b) a*b
void fun(int i){
   static int x=1;
   x+=PRODUCT(x+i,x-i);
   printf("x=%d\n",x);
}
main(){
```

```
    int i,x=1;
    for(i=1;i<=3;i++,x++)
    fun(x+i);
}
```

估计结果：

-1

-7

-14

实验结果：

x=2

x=8

x=58

实验总结：

助教我现在知道静态变量下一次的数值不会改变，但是我不知道为什么这里的宏在变量替换之后算出来的数值和我不用宏直接算的结果不同呢。