1、 概念基础：

局部变量：函数内部定义的变量（包括定义在函数内部复合语句中的变量）。

全局变量：定义在函数外部的变量。

作用域：任何标识符（包括变量、函数名、符号常量及新的数据名）都有它的作用范围，此范围称为该标识符的作用域。比如符号常量的作用域是从定义符号常量的地方开始到包含这个#define 命令（作用于该符号常量）的文件末尾或者遇到#undef 命令（作用于该符号常量）为止。

2、 对于变量：

static 关键字：static 可以用于修饰局部变量以扩展局部变量的生存期，被 static 关键字修饰的局部变量的生存期为：在调用该变量所在的函数前已生成，直到程序退出才消亡，因此在调用该局部变量所在的函数后该变量仍然存在并保持最后使用的值。虽然 static 关键字可以改变局部变量的生存期，但是不能改变局部变量的作用域，该局部变量仍然只能在定义它的函数中使用。

static 关键字也可可用于修饰全局变量，此时它的作用在于限制该全局变量的作用域，只能在定义该全局变量的文件中才能使用。

extern 关键字：只能用于扩展没有被 static 关键字修饰的全局变量。默认情况下全局变量只能在定义它的文件中使用（从定义该全局变量开始到所在文件的文件尾），但如果在另一个文件中将这个变量声明为外部变量，那么这个变量的作用域将被扩展到另外一个文件中。也可以在定义全局变量之前声明该变量，从而在文件中可以在定义该全局变量前使用该全局变量。

3、 对于函数：

static 关键字：在定义函数时在函数首部的最左边加上 static 可以把该函数声明为内部函数

（又叫静态函数），这样该函数就只能在其定义所在的文件中使用，如果在不同的文件中有同名的内部函数，则互不干扰。

extern 关键字：在定义函数时如果在函数首部的最左端冠以关键字 extern，则表示此函数是外部函数，可供其他文件调用。C 语言规定，如果在定义函数时省略 extern，则隐含为外部函数。

在文件中要调用其他文件中的外部函数，则需要在文件中用 extern 声明该外部函数，然后就可以使用。

资料 2

1、用 static、extern 可以声明变量

1.1 可用 static 声明全局变量和局部变量（包括数组）。

(1)用 static 声明的局部变量只能被定义该变量的函数识别，在退出函数时该局部变量仍然保持其值，

e.g. void temp()

{

static int array[]； //静态局部变量

.....

}

用 static 声明局部数组可以避免在每次调用函数时都建立和初始化数组，以及在每次退出函数时撤销数组，缩短程序执行时间。

(2)用 static 声明静态外部变量,可以限定外部变量只被本文件引用,不会被其他文件引用.为文件的模块化、通用性提供方便.

e.g. file1.c(file1 文件)

static int A； //静态外部变量

main（）{ }

注意:外部变量和静态外部变量都是静态存储的,只是作用范围不同.

1.2 存储类别说明符 extern

extern 外部变量的作用域是从变量定义处开始,到本程序文件的末尾.

(1)在一个文件内声明外部变量

用 extern 声明外部变量以扩展它在程序文件中的作用域

e.g. main()

{

extern int A,B; //外部变量声明

.....

}

int A=14,B=-3; //定义外部变量

(2)在多文件的程序中声明外部变量

如一个程序包含多个文件,在一个文件中定义外部变量,在其他文件中 extern 做外部变量声明.

e.g. 文件 file1.c 中定义

int A;

文件 file2.c 中

extern int A; //作用域扩大,不再分配内存

2、用 static、extern 可以声明函数

(1)内部函数(静态函数)-用 static 声明

用 static 声明的内部函数,只能被本文件中的其他函数调用,调用范围只局限与所在文件,不同文件中有同名的内部函数,互不干扰.

e.g. static int fun(int a,int b)

(2)外部函数-用 extern 声明

★用 extern 声明的外部函数可供其他文件调用.一般没有任何声明的函数都是外部函数.

e.g. extern int fun(int a,int b)

★在需要调用此函数的文件中.用 extern 声明所用的函数是外部函数.

e.g. file1.c(file1 文件)

```
main()
{
extern printf_string(char str[]); //声明在本函数中要调用其他文件中的函数
.....
}
```

file2.c(file2 文件)

```
void printf_string(char str[])
{.....}
```

注意:extern 可以省略,如使用#include<math.h>,可在文件中直接调用其中的函数.

3、标识符有四种作用域：函数作用域、文件作用域、块作用域、函数原型作用域。

(1)标号（跟有冒号的标识符，如 start：）是唯一具有函数作用域的标识符。编号可用于函数中任何地方，但不能在函数之外引用。用在 swtich 结构(case:)和 goto 语句中。

(2)函数之外的全局变量、函数声明和函数原型具有文件作用域，

e.g. extern x=10;

void temp(void);

(3)static 声明的局部变量具有块作用域，

e.g. void temp(void)

{

static int x=10;

}

(4)用在函数原型参数列表中的标识符具有函数原型作用域，

e.g. void temp(const [],int)；

1、数组传递给函数，模拟传引用调用的方式自动把数组传递给函数，意味着被调用的函数能够修改原是数组的元素值，

e.g. int hourly[24]; //数组定义

void modify(int [],int); //函数原型

modify(hourly,24); //函数调用，数组名即数组第一个元素的地址

注意：为了防止函数中修改数组的值，采用类型限定符 const 做前缀，数组元素成为函数体中的常量，函数体中任何修改数组元素的企图都会导致编译错误，

e.g. void modify(const int [],int);

2、字符数组

字符串读取：scanf("%s",string1); //不用地址符号&,因为数组名是数组的起始地址

字符串显示：printf("%s",string1);

e.g. char string1[]="good girl"; //字符串

char string1[]={'g','o','o','d','\0','g','i','r','l','\0'}; //上下声明等价

注意：'空字符'='\0',终止符,用 printf("%s",string1)输出，得到 good girl。

3、多维数组

e.g. int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

假设数组起始地址 2000，则

表示形式 含义 地址

a 指向 a[0]，即 0 行首地址。 2000

a[0],*(a+0),*a 第 0 行第 0 列元素地址 2000

a+1,&a[1] 第 1 行首地址 2008

a[1],*(a+1) 第 1 行第 0 列元素地址 2008

a[1]+2,*(a+1)+2,&a[1][2] 第 1 行第 2 列元素地址 2012

(a[1]+2),(*(a+1)+2),a[1][2] 第 1 行第 2 列元素的值 7

在二维数组中，a+i=a[i]=*(a+i)=&a[i]=&a[i][0],即他们的的地址值是想等的。

资料 3

```
DE>文件 a.c
static int i; //只在 a 文件中用
int j;  //在工程里用
static void init()       //只在 a 文件中用
{
}
void callme()        //在工程中用
{
  static int sum;
}DE>
```

上 面的全局 i 变量和 init()函数只能用在 a.c 文件中,全局变量 sum 的作用域只在 callme

里。变量 j 和函数 callme()的全局限扩充到整个工 程文件。所以可以在下面的 b.c 中用 extern 关键字调用。extern 告诉编译器这个变量或者函数在其他文件里已经被定义了。

DE>文件 b.c

```
extern int j;    //调用 a 文件里的
extern void callme(); //调用 a 文件里的
int main()
{
  ...
}DE>
```

extern 的另外用法是当 C 和 C++混合编程时如果 c++调用的是 c 源文件定义的函数或者变量，那么要加 extern 来告诉编译器用 c 方式命名函数：

DE>文件 A.cpp 调用 a.c 里面的变量 i 和函数 callme()

```
extern "C" //在 c++文件里调用 c 文件中的变量
{
   int j;
    void callme();
}
int main()
{
    callme();
DE>
```