全局变量与局部变量的区别

一、变量的分类

变量可以分为：全局变量、静态全局变量、静态局部变量和局部变量。

按存储区域分，全局变量、静态全局变量和静态局部变量都存放在内存的静态存储区域，局部变量存放在内存的栈区。

按作用域分，全局变量在整个工程文件内都有效；静态全局变量只在定义它的文件内有效；静态局部变量只在定义它的函数内有效，只是程序仅分配一次内存，函数返回后，该变量不会消失；局部变量在定义它的函数内有效，但是函数返回后失效。

全局变量和静态变量如果没有手工初始化，则由编译器初始化为0。局部变量的值不可知。

静态全局变量：只在本文件内有效。

全局变量是没有定义存储类型的外部变量,其作用域是从定义点到程序结束.省略了存储类型符,系统将默认为是自动型.

静态全局变量是定义存储类型为静态型的外部变量,其作用域是从定义点到程序结束,所不同的是存储类型决定了存储地点,静态型变量是存放在内存的数据区中的,它们在程序开始运行前就分配了固定的字节,在程序运行过程中被分配的字节大小是不改变的.只有程序运行结束后,才释放所占用的内存.

自动型变量存放在堆栈区中.堆栈区也是内存中一部分,该部分内存在程序运行中是重复使用的.

二、变量作用域的介绍

在讨论函数的形参变量时曾经提到， 形参变量只在被调用期间才分配内存单元，调用结束立即释放。 这一点表明形参变量只有在函数内才是有效的， 离开该函数就不能再使用了。这种变量有效性的范围称变量的作用域。不仅对于形参变量， C语言中所有的量都有自己的作用域。变量说明的方式不同，其作用域也不同。 C语言中的变量，按作用域范围可分为两种， 即局部变量和全局变量。

1、局部变量

局部变量也称为内部变量。局部变量是在函数内作定义说明的。其作用域仅限于函数内， 离开该函数后再使用这种变量是非法的。

例如：

int f1(int a) /\*函数f1\*/

{

int b,c;

... ...

}a,b,c作用域

int f2(int x) /\*函数f2\*/

{

int y,z;

... ...

}x,y,z作用域

main()

{

int m,n;

}m,n作用域

在函数f1内定义了三个变量，a为形参，b,c为一般变量。在 f1的范围内a,b,c有效，或者说a,b,c变量的作用域限于f1内。同理，x,y,z的作用域限于f2内。 m,n的作用域限于main函数内。

关于局部变量的作用域还要说明以下几点：

1、主函数中定义的变量也只能在主函数中使用，不能在其它函数中使用。同时，主函数中也不能使用其它函数中定义的变量。因为主函数也是一个函数，它与其它函数是平行关系。这一点是与其它语言不同的，应予以注意。

2、形参变量是属于被调函数的局部变量，实参变量是属于主调函数的局部变量。

3、允许在不同的函数中使用相同的变量名，它们代表不同的对象，分配不同的单元，互不干扰，也不会发生混淆。如在例5.3 中，形参和实参的变量名都为n，是完全允许的。

4、在复合语句中也可定义变量，其作用域只在复合语句范围内。例如：

main()

{

int s,a;

……

{

int b;

s=a+b;

……b作用域

}

……s,a作用域

}

main()

{

int i=2,j=3,k;

k=i+j;

{

int k=8;

if(i==3) printf("%d\n",k);

}

printf("%d\n%d\n",i,k);

}

main()

{

int i=2,j=3,k;

k=i+j;

{

int k=8;

if(i=3) printf("%d\n",k);

}

printf("%d\n%d\n",i,k);

}

　　本程序在main中定义了i,j,k三个变量，其中k未赋初值。 而在复合语句内又定义了一个变量k，并赋初值为8。应该注意这两个k不是同一个变量。在复合语句外由main定义的k起作用，而在复合语句内则由在复合语句内定义的k起作用。因此程序第4行的k为main所定义，其值应为5。第7行输出k值，该行在复合语句内，由复合语句内定义的k起作用，其初值为8，故输出值为8，第9行输出i，k值。i是在整个程序中有效的，第7行对i赋值为3，故以输出也为3。而第9行已在复合语句之外，输出的k应为main所定义的k，此k值由第4 行已获得为5，故输出也为5。

二、全局变量

全局变量也称为外部变量，它是在函数外部定义的变量。 它不属于哪一个函数，它属于一个源程序文件。其作用域是整个源程序。在函数中使用全局变量，一般应作全局变量说明。 只有在函数内经过说明的全局变量才能使用。全局变量的说明符为extern。 但在一个函数之前定义的全局变量，在该函数内使用可不再加以说明。 例如：

int a,b; /\*外部变量\*/

void f1() /\*函数f1\*/

{

……

}

float x,y; /\*外部变量\*/

int fz() /\*函数fz\*/

{

……

}

main() /\*主函数\*/

{

……

}/\*全局变量x,y作用域 全局变量a,b作用域\*/

从上例可以看出a、b、x、y 都是在函数外部定义的外部变量，都是全局变量。但x,y 定义在函数f1之后，而在f1内又无对x,y的说明，所以它们在f1内无效。 a,b定义在源程序最前面，因此在f1,f2及main内不加说明也可使用。

输入正方体的长宽高l,w,h。求体积及三个面x\*y,x\*z,y\*z的面积。

int s1,s2,s3;

int vs( int a,int b,int c)

{

int v;

v=a\*b\*c;

s1=a\*b;

s2=b\*c;

s3=a\*c;

return v;

}

main()

{

int v,l,w,h;

printf("\ninput length,width and height\n");

scanf("%d%d%d",&l,&w,&h);

v=vs(l,w,h);

printf("v=%d s1=%d s2=%d s3=%d\n",v,s1,s2,s3);

}

本程序中定义了三个外部变量s1,s2,s3， 用来存放三个面积，其作用域为整个程序。函数vs用来求正方体体积和三个面积， 函数的返回值为体积v。由主函数完成长宽高的输入及结果输出。由于C语言规定函数返回值只有一个， 当需要增加函数的返回数据时，用外部变量是一种很好的方式。本例中，如不使用外部变量， 在主函数中就不可能取得v,s1,s2,s3四个值。而采用了外部变量， 在函数vs中求得的s1,s2,s3值在main 中仍然有效。因此外部变量是实现函数之间数据通讯的有效手段。

对于全局变量还有以下几点说明：

1、对于局部变量的定义和说明，可以不加区分。而对于外部变量则不然，外部变量的定义和外部变量的说明并不是一回事。外部变量定义必须在所有的函数之外，且只能定义一次。其一般形式为：

[extern] 类型说明符 变量名，变量名… 其中方括号内的extern可以省去不写。

　　例如： int a,b;

　　等效于：

　　extern int a,b;

而外部变量说明出现在要使用该外部变量的各个函数内， 在整个程序内，可能出现多次，外部变量说明的一般形式为： extern 类型说明符 变量名，变量名，…； 外部变量在定义时就已分配了内存单元， 外部变量定义可作初始赋值，外部变量说明不能再赋初始值， 只是表明在函数内要使用某外部变量。

2、外部变量可加强函数模块之间的数据联系， 但是又使函数要依赖这些变量，因而使得函数的独立性降低。从模块化程序设计的观点来看这是不利的， 因此在不必要时尽量不要使用全局变量。

3、在同一源文件中，允许全局变量和局部变量同名。在局部变量的作用域内，全局变量不起作用。

int vs(int l,int w)

{

extern int h;

int v;

v=l\*w\*h;

return v;

}

main()

{

extern int w,h;

int l=5;

printf("v=%d",vs(l,w));

}

int l=3,w=4,h=5;

本例程序中，外部变量在最后定义， 因此在前面函数中对要用的外部变量必须进行说明。外部变量l，w和vs函数的形参l，w同名。外部变量都作了初始赋值，mian函数中也对l作了初始化赋值。执行程序时，在printf语句中调用vs函数，实参l的值应为main中定义的l值，等于5，外部变量l在main内不起作用；实参w的值为外部变量w的值为4，进入vs后这两个值传送给形参l，wvs函数中使用的h 为外部变量，其值为5，因此v的计算结果为100，返回主函数后输出。

三、变量的存储类型

各种变量的作用域不同， 就其本质来说是因变量的存储类型相同。所谓存储类型是指变量占用内存空间的方式， 也称为存储方式。

变量的存储方式可分为“静态存储”和“动态存储”两种。

静态存储变量通常是在变量定义时就分定存储单元并一直保持不变， 直至整个程序结束。全局变量即属于此类存储方式。

动态存储变量是在程序执行过程中，使用它时才分配存储单元， 使用完毕立即释放。 典型的例子是函数的形式参数，在函数定义时并不给形参分配存储单元，只是在函数被调用时，才予以分配， 调用函数完毕立即释放。如果一个函数被多次调用，则反复地分配、 释放形参变量的存储单元。

从以上分析可知， 静态存储变量是一直存在的， 而动态存储变量则时而存在时而消失。我们又把这种由于变量存储方式不同而产生的特性称变量的生存期。 生存期表示了变量存在的时间。 生存期和作用域是从时间和空间这两个不同的角度来描述变量的特性，这两者既有联系，又有区别。 一个变量究竟属于哪一种存储方式， 并不能仅从其作用域来判断，还应有明确的存储类型说明。

在C语言中，对变量的存储类型说明有以下四种：

　　auto　　　　 自动变量

　　register 　　 寄存器变量

　　extern 　　　外部变量

　　static 　　　 静态变量

自动变量和寄存器变量属于动态存储方式， 外部变量和静态变量属于静态存储方式。

在介绍了变量的存储类型之后， 可以知道对一个变量的说明不仅应说明其数据类型，还应说明其存储类型。 因此变量说明的完整形式应为： 存储类型说明符 数据类型说明符 变量名，变量名…； 例如：

static int a,b; 　　　　　　　　　 说明a,b为静态类型变量

auto char c1,c2; 　　　　　　　　　说明c1,c2为自动字符变量

static int a[5]={1,2,3,4,5}; 　　　说明a为静整型数组

extern int x,y; 　　　　　　　　　 说明x,y为外部整型变量。

四、总结与区别：

变量根据定义的位置的不同的生命周期，具有不同的作用域，作用域可分为6种：全局作用域，局部作用域，语句作用域，类作用域，命名空间作用域和文件作用域。

从作用域看：

全局变量具有全局作用域。全局变量只需在一个源文件中定义，就可以作用于所有的源文件。当然，其他不包含全局变量定义的源文件需要用extern 关键字再次声明这个全局变量。

静态局部变量具有局部作用域，它只被初始化一次，自从第一次被初始化直到程序运行结束都一直存在，它和全局变量的区别在于全局变量对所有的函数都是可见的，而静态局部变量只对定义自己的函数体始终可见。

局部变量也只有局部作用域，它是自动对象（auto），它在程序运行期间不是一直存在，而是只在函数执行期间存在，函数的一次调用执行结束后，变量被撤销，其所占用的内存也被收回。

静态全局变量也具有全局作用域，它与全局变量的区别在于如果程序包含多个文件的话，它作用于定义它的文件里，不能作用到其它文件里，即被static关键字修饰过的变量具有文件作用域。这样即使两个不同的源文件都定义了相同名字的静态全局变量，它们也是不同的变量。

从分配内存空间看：

全局变量，静态局部变量，静态全局变量都在静态存储区分配空间，而局部变量在栈里分配空间。

全局变量本身就是静态存储方式，静态全局变量当然也是静态存储方式。这两者在存储方式上并无不同。这两者的区别虽在于非静态全局变量的作用域是整个源程序，当一个源程序由多个源文件组成时，非静态的全局变量在各个源文件中都是有效的。而静态全局变量则限制了其作用域，即只在定义该变量的源文件内有效，在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用，因此可以避免在其它源文件中引起错误。

1)、静态变量会被放在程序的静态数据存储区(全局可见)中，这样可以在下一次调用的时候还可以保持原来的赋值。这一点是它与堆栈变量和堆变量的区别。

2)、变量用static告知编译器，自己仅仅在变量的作用范围内可见。这一点是它与全局变量的区别。

从以上分析可以看出，把局部变量改变为静态变量后是改变了它的存储方式即改变了它的生存期。把全局变量改变为静态变量后是改变了它的作用域，限制了它的使用范围。因此static 这个说明符在不同的地方所起的作用是不同的。应予以注意。

Tips:

　　A.若全局变量仅在单个C文件中访问，则可以将这个变量修改为静态全局变量，以降低模块间的耦合度；

　　B.若全局变量仅由单个函数访问，则可以将这个变量改为该函数的静态局部变量，以降低模块间的耦合度；

　　C.设计和使用访问动态全局变量、静态全局变量、静态局部变量的函数时，需要考虑重入问题，因为他们都放在静态数据存储区，全局可见；

D.如果我们需要一个可重入的函数，那么，我们一定要避免函数中使用static变量(这样的函数被称为：带“内部存储器”功能的的函数)

E.函数中必须要使用static变量情况:比如当某函数的返回值为指针类型时，则必须是static的局部变量的地址作为返回值，若为auto类型，则返回为错指针。