1.预处理&关键字（22道）

1.1宏定义是在编译的哪个阶段被处理的?

答案：宏定义是在编译预处理阶段被处理的。

解读：编译预处理：头文件包含、宏替换、条件编译、去除注释、添加行号。

1.2写一个“标准”宏MIN，这个宏输入两个参数并返回较小的一个。

答案：

```
#define  MIN(A, B)  ((A) <= (B)? (A) : (B))
```

解读：

（1）注意这个题目要用三重条件操作符，在宏中要小心地把参数用括号括起来，并且整个宏也要用括号括起来，防止替换时出现错误。

（2）注意若写“least = MIN(*p++, b);”这句代码会产生副作用，将*p++代入宏体，指针p会做两次自增操作。

1.3已知数组table，用宏求数组元素个数。

答案：

```
#define  COUNT(table)  (sizeof(table) / sizeof(table[0]))
```

解读：sizeof(table)得到数组长度，sizeof(table[0])得到数组元素长度，两者相除即可得到数组元素个数。

1.4带参宏和函数的区别?

（1）带参宏只是在编译预处理阶段进行简单的字符替换；而函数则是在运行时进行调用和返回。

（2）宏替换不占运行时间，只占编译时间；而函数调用则占运行时间（分配单元、保留现　　场、值传递、返回）。

（3）带参宏在处理时不分配内存；而函数调用会分配临时内存。

（4）宏不存在类型问题，宏名无类型，它的参数也是无类型的；而函数中的实参和形参都要定义类型，二者的类型要求一致。

（5）而使用宏定义次数多时，宏替换后源程序会变长；而函数调用不使源程序变长。

1.5内联函数的优缺点和适用场景是什么?

（1）优点：内联函数与宏定义一样会在原地展开，省去了函数调用开销，同时又能做类型检查。

（2）缺点：它会使程序的代码量增大，消耗更多内存空间。

（3）适用场景：函数体内没有循环（执行时间短）且代码简短（占用内存空间小）。

1.6关键字volatile的作用是什么？给出三个不同的例子。

（1）作用：告诉编译器不要去假设（优化）这个变量的值，因为这个变量可能会被意想不到地改变。精确地说就是，优化器在用到这个变量时必须每次都小心地重新读取这个变量的值，而不是使用保存在寄存器里的备份。

（2）例子：

①并行设备的硬件寄存器（如：状态寄存器）。

②一个中断服务子程序中会访问到的非自动变量。

③多线程应用中被几个线程共享的变量（防止死锁）。

1.7如何用C语言实现读写寄存器变量?

答案：

```
#define  rBANKCON0  (*(volatile unsigned long *)0x48000004)
rBankCON0 = 0x12;
```

解读：

（1）由于是寄存器地址，所以需要先将其强制类型转换为 ”volatile unsigned long *”。

（2）由于后续需要对寄存器直接赋值，所以需要解引用。

1.8下面代码能不能编译通过？

```
#define c 3
c++;
```

答案：不能。

解读：自增运算符++用于变量，3是常量。

1.9“在C语言中，凡是以#开头的都是预处理命令，同时预处理命令都是以#开头的”，这句话是正确的吗？

正确。

1.10预处理器标识#error的作用是什么？

答案：编译程序时，只要遇到 #error 就会跳出一个编译错误。

解读：当程序比较大时，往往有些宏定义是在外部指定的（如makefile），或是在系统头文件中指定的，当你不太确定当前是否定义了 XXX 时，可写如下预处理代码：

```
#ifdef XXX
#error "XXX has been defined"
#else
…
#endif
```

这样，如果编译时出现错误，输出了XXX has been defined，表明宏XXX已经被定义了。

1.11用预处理指令#define声明一个常数，用以表明1年中有多少秒（忽略闰年问题）。

答案：

```
#define  SECONDS_PER_YEAR  (60 * 60 * 24 * 365)UL
```

解读：

（1）注意预处理器将为你计算常数表达式的值，并且整个宏体要用括号括起来。

（2）注意这个表达式将使一个16位机的整型数溢出，因此要用到无符号长整型符号UL，告诉编译器这个常数是的无符号长整型数。

1.12关键字static的作用是什么？

（1）static修饰局部变量时：①改变了其存储位置，存储在静态区；②同时改变了其生命周期，为整个源程序，因此它只被初始化一次，若没显式初始化则自动初始化为0。

（2）static修饰全局变量时：改变了其作用域，只可以被文件内所用函数访问。

（3）static修饰函数时：改变了其作用域，只可被这一文件内的其它函数调用。

1.13下面是关键字const的使用示例，请说明它们的作用：

```
（1）const int a;              // a是一个整形常量
   int const a;               // a是一个整形常量
（2）const int *a;             // a是一个指向整型常量的指针变量
   int * const a;             // a是一个指向整型变量的指针常量
   int const * const a = &b;  // a是一个指向整型常量的指针常量
（3）char *strcpy(char *strDest, const char *strSrc);  // 参数在函数内部不会被修改
   const int strcmp(char *source, char *dest);     // 函数的返回值不能被修改
```

答案：以注释形式展示。

解读：const放在*前是修饰指向的对象，放在*后则是修饰指针本身。

1.14一个参数既可以是const还可以是volatile吗？一个指针可以是volatile吗？下面的函数有什么问题？

```
int square(volatile int *ptr)
{
    return *ptr * *ptr;
}
```

（1）是的。一个例子是只读的状态寄存器，它是volatile因为它可能被意想不到地改变，它是const因为程序不应该试图去修改它。

（2）是的。一个例子是当一个中服务子程序修改一个指向一个缓冲区的指针时。
（3）这个函数的目的是用来返指针*ptr指向值的平方，但是，由于*ptr指向一个volatile型参数，编译器将产生类似下面的代码：

```
int square(volatile int *ptr)
{
    int a, b;
    a = *ptr;
    b = *ptr;
    return a * b;
}
```


由于*ptr的值可能被意想不到地该变，因此a和b可能是不同的。结果，这段代码可能返不是你所期望的平方值！正确的代码如下：

```
long square(volatile int *ptr)
{
    int a;
    a = *ptr;
    return a * a;
}
```

1.15关键字typedef 在C语言中频繁用以声明一个已经存在的数据类型的同义字。也可以用预处理器做类似的事。例如，思考一下下面的例子：

```
#define  dPS  struct s *
typedef  struct s *  tPS;  //(顺序、分号、#号)
```

以上两种情况的意图都是要定义dPS 和 tPS 作为一个指向结构体s的指针。哪种方法更好呢？为什么？
（1）typedef更好。

（2）举个例子：

```
dPS  p1, p2;
tPS  p3, p4;
```

第一行代码扩展为 struct s * p1, p2; 即定义p1为一个指向结构体的指针，p2为一个实际的结构体，这也许不是你想要的。第二行代码正确地定义了p3 和p4 两个指针。

1.16关键字sizeof的作用是什么？函数strlen()呢？

（1）sizeof关键字用来计算变量、数据类型所占内存的字节数。sizeof(数组名)得到数组所占字节数，sizeof(字符串指针名)得到指针所占字节数。

（2）而strlen()函数则用来测试字符串所占字节数，不包括结束字符 ’\0’。strlen(字符数组名)得到字符串所占字节数，strlen(字符串指针名)得到字符串所占字节数。

1.17关键字extern的作用是什么?

答案：用于跨文件引用全局变量，即在本文件中引用一个已经在其他文件中定义的全局变量。

解读：

（1）注意引用时不能初始化，如extern var，而不能是extern var = 0。

（2）另外，函数默认是extern类型的，表明是整个程序（工程）可见的，加不加都一样。

1.18 extern”C”的作用?

答案：

（1）在C++代码中调用C函数，用法：extern “C”{C函数库头文件/函数声明}。

（2）在C代码中调用C++函数，用法：在C++的头文件中加extern“C”{头文件/函数声明}。
注意：extern”C”只能用于C++文件中。

1.19关键字auto的作用是什么?

答案：用来定义自动局部变量，自动局部变量在进入声明该变量的语句块时被建立，退出语句块时被注销，仅在语句块内部使用。

解读：其实普通局部变量就是自动局部变量，只是省略了auto这一关键字。

1.20关键字register的作用是什么？使用时需要注意什么?

（1）作用：编译器会将register修饰的变量尽可能地放在CPU的寄存器中，以加快其存取速度，一般用于频繁使用的变量。

（2）注意：register变量可能不存放在内存中，所以不能用&来获取该变量的地址；只有局部变量和形参可以作为register变量；寄存器数量有限，不能定义过多register变量。

1.21 C语言编译过程中，关键字volatile和extern分别在哪个阶段起作用?

答案：volatile在编译阶段，extern在链接阶段。

解读：C语言编译过程分为预处理、编译、汇编、链接。

1.22 const与#define的异同?

（1）异：const有数据类型，编译器可以做静态类型检查；而宏定义没有类型，可能会导致类型出错。

（2）同：两者都可用来定义常数。