Typedef

**用途一：**定义一种类型的别名，而不只是简单的宏替换。（尤其是指针类型别名）

**用途二:** 用在旧的C代码中，帮助struct。以前的代码中，声明struct新对象时，必须要带上struct，

在C或c++中用typedef:定义一个结构体类型

**typedef struct Student**  
**{**

**int a;**

**}Stu;**

　　　　这里的Stu实际上就是struct Student的别名。是种数据类型。

但在c++还可以直接

**struct   Student**  
**{**  
**int   a;**  
**}stu1;**

**Student是数据类型**

**stu1是一个变量**

**malloc**

**一头文件**：#include <malloc.h>

**二功能：**分配长度为num\_bytes字节的内存块

说明：如果分配成功则返回指向被分配内存的指针，否则返回空指针NULL。当内存不再使用时，应使用free()函数将内存块释放。

**三malloc与new的不同点**

　　malloc 和 new 至少有两个不同: **new 返回指定类型的指针，并且可以自动计算所需要大小。**比如：

  int \*p;

　　p = new int; //返回类型为int\* 类型(整数型指针)，分配大小为 sizeof(int);

　　p = new int[100]; //分配大小为 sizeof(int) \* 100;

**而 malloc 则必须由我们计算要字节数，并且在返回后强行转换为实际类型的指针。**

  int\* p;

　　p = (int \*) malloc (sizeof(int));

　　第一、malloc 函数返回的是 void \* 类型，如果你写成：p = malloc (sizeof(int)); 则程序无法通过编译，报错：“不能将 void\* 赋值给 int \* 类型变量”。所以必须通过 (int \*) 来将强制转换。

　　第二、函数的实参为 sizeof(int) ，用于指明一个整型数据需要的大小。如果你写成：

　　int\* p = (int \*) malloc (1);

　　代码也能通过编译，但事实上只分配了1个字节大小的内存空间，当你往里头存入一个整数，就会有3个字节无家可归，而直接“住进邻居家”！造成的结果是后面的内存中原有数据内容全部被清空。

　　第三、如果想分配100个int类型的空间：

　　int\* p = (int \*) malloc ( sizeof(int) \* 100 ); //分配可以放得下100个整数的内存空间。

　　另外有一点不能直接看出的区别是，malloc 只管分配内存，并不能对所得的内存进行初始化，所以得到的一片新内存中，其值将是随机的。

**sizeof有两种用法：**

对对象使用: sizeof(object), 也可以写成sizeof object 的形式

对类型使用: sizeof(typename), 但是写成sizeof typename是非法的。

int i = 2;  
sizeof(i); // sizeof(object)的用法，合理  
sizeof i; // sizeof object的用法，合理  
sizeof 2; // 2被解析成int类型的object, sizeof object的用法，合理  
sizeof(2); // 2被解析成int类型的object, sizeof(object)的用法，合理  
sizeof(int);// sizeof(typename)的用法，合理  
sizeof int; // 错误！对于操作符，一定要加()

可以看出，加()是永远正确的选择。