**malloc**:

原型：extern void \*malloc(unsigned int num\_bytes);  
　　头文件：在TC2.0中可以用malloc.h或 alloc.h (注意：alloc.h 与 malloc.h 的内容是完全一致的)，而在Visual C++6.0中可以用malloc.h或者stdlib.h。  
　　功能：分配长度为num\_bytes字节的内存块  
　　返回值：如果分配成功则返回指向被分配内存的指针(此存储区中的初始值不确定)，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。函数返回的指针一定要适当对齐，使其可以用于任何数据对象。

void \*malloc(size\_t size);  
　　说明：malloc 向系统申请分配指定size个字节的内存空间。返回类型是 void\* 类型。void\* 表示未确定类型的指针。C,C++规定，void\* 类型可以强制转换为任何其它类型的指针。  
　　备注：void\* 表示未确定类型的指针，更明确的说是指申请内存空间时还不知道用户是用这段空间来存储什么类型的数据（比如是char还是int或者...）  
　　从函数声明上可以看出。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);  
　　或：  
　　int\* parr;  
　　parr = new int [100]; //返回类型为 int\* 类型(整数型指针)，分配大小为 sizeof(int) \* 100;  
　　而 malloc 则必须要由我们计算字节数，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)\*128);//分配128个（可根据实际需要替换该数值）整型存储单元，并将这128个连续的整型存储单元的首地址存储到指针变量p中  
　　double \*pd=(double \*) malloc (sizeof(double)\*12);//分配12个double型存储单元，并将首地址存储到指针变量pd中  
　　第一、malloc 函数返回的是 void \* 类型。对于C++，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。而对于C，没有这个要求，但为了使C程序更方便的移植到C++中来，建议养成强制转换的习惯。  
　　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被改写。

工作机制：

malloc函数的实质体现在，它有一个将可用的内存块连接为一个长长的列表的所谓空闲[链表](http://baike.baidu.com/view/549479.htm)。调用malloc函数时，它沿[连接表](http://baike.baidu.com/view/694082.htm)寻找一个大到足以满足用户请求所需要的内存块。然后，将该内存块一分为二（一块的大小与用户请求的大小相等，另一块的大小就是剩下的字节）。接下来，将分配给用户的那块内存传给用户，并将剩下的那块（如果有的话）返回到连接表上。调用[free](http://baike.baidu.com/view/512783.htm)函数时，它将用户释放的内存块连接到空闲链上。到最后，空闲链会被切成很多的小内存片段，如果这时用户申请一个大的内存片段，那么空闲链上可能没有可以满足用户要求的片段了。于是，malloc函数请求延时，并开始在空闲链上翻箱倒柜地检查各内存片段，对它们进行整理，将相邻的小空闲块合并成较大的内存块。如果无法获得符合要求的内存块，malloc函数会返回NULL[指针](http://baike.baidu.com/view/159417.htm)，因此在调用malloc动态申请内存块时，一定要进行返回值的判断。

**calloc**：

calloc是一个C语言函数

　　函数名: calloc

　　void \*calloc(unsigned n,unsigned size)；

　　功 能: 在内存的[动态存储](http://baike.baidu.com/view/589359.htm)区中分配n个长度为size的连续空间，函数返回一个指向分配起始地址的[指针](http://baike.baidu.com/view/159417.htm)；如果分配不成功，返回NULL。

　　跟[malloc](http://baike.baidu.com/view/736228.htm)的区别：calloc在动态分配完内存后，自动初始化该内存空间为零，而malloc不初始化，里边数据是随机的垃圾数据。

**realloc**：

函数简介

　　原型：extern void \*realloc(void \*mem\_address, unsigned int newsize);

　　语法：指针名=（数据类型\*）realloc（要改变内存大小的指针名，新的大小）。//新的大小一定要大于原来的大小不然的话会导致数据丢失！

　　头文件：#include <stdlib.h> 有些编译器需要#include <malloc.h>，在TC2.0中可以使用alloc.h头文件

　　功能：先判断当前的指针是否有足够的连续空间，如果有，扩大mem\_address指向的地址，并且将mem\_address返回，如果空间不够，先按照newsize指定的大小分配空间，将原有数据从头到尾拷贝到新分配的内存区域，而后释放原来mem\_address所指内存区域，同时返回新分配的内存区域的首地址。即重新分配存储器块的地址。

　　返回值：如果重新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。

　　注意：这里原始内存中的数据还是保持不变的。当内存不再使用时，应使用free()函数将内存块释放。

**\_alloca：**

函数原型为：void \* \_\_cdecl alloca(size\_t);

内存分配函数,与malloc,calloc,realloc类似.  
　　但是注意一个重要的区别,\_alloca是在栈(stack)上申请空间,用完马上就释放.  
　　包含在头文件malloc.h中.  
　　在某些系统中会宏定义成\_alloca使用.