§5.动态数据区与静态数据区的研究与思考

测试项	VS2017 x86	VS2017 x64	CodeBlocks (32bit)	Dev C++ (32bit)	Dev C++ (64bit)	Linux C++ (64bit)	思考与心得体会
动态数据区大小（精确到64KB）	1MB	1MB	2MB	2MB	2MB	8MB	不同操作系统和编译器对栈的大小设置有所不同，VS 默认为 1MB，Windows 下 gcc 系列默认为 2MB，Linux 下为8MB
极限定义下动态数据区的上限（起始分配地址）	00EB5138	00000017E4D06060	0x741554f4	0x4e4f50	0x506050	0x7ffe250bcef0	栈的地址有时在堆之上，有时在堆之下
极限定义下动态数据区的下限（结束分配地址）	00FAFC70	00000017E4DFF5C0	0x7434ff00	0x6dfea0	0x6ffe50	0x7ffe258bb1b0	栈中不同变量的地址随定义的次序由高到低排列(VS x64 除外)
静态数据区大小（精确到64KB）	1850MB	1892MB	1852MB	1903MB	1903MB	243MB	不同操作系统对堆的大小设置差异很大，Windows 下为小于 2GB，Linux 下为小于256MB（用户配置）
极限定义下静态数据区的上限（起始分配地址）	0011C138	00007FF6A032D160	0x4c6020	0x491040	0x4a7040	0x601180	堆的地址有时在栈之上，有时在栈之下
极限定义下静态数据区的下限（结束分配地址）	73B3CE38	00007FF7167BCA60	0x74130c20	0x773928d4	0x773a88d4	0xf9b8ffc	堆中不同变量的地址随定义的次序由低到高排列
如果动态数据区和静态数据区都极限定义，两者地址最近相差多少？	4.8MB	127TB	2MB	50MB	50MB	128TB	不同编译器差异很大，但一般来说64 位的编译器会

							将两者的地址分隔非常大的距离（Dev 除外）
动态数据区，char x,y,z; 观察 xyz 间的地址间隔	12B	32B	1B	1B	1B	1B	不同编译器有不同的表现。VS 中变量相隔较远，且为固定常数（x86 与 x64 不同）。gcc 系列为变量的大小，即变量在内存中从高到低连续排列
动态数据区，int x,y,z; 观察 xyz 间的地址间隔	12B	32B	4B	4B	4B	4B	不同编译器有不同的表现。VS 中变量相隔较远，且为固定常数（x86 与 x64 不同）。gcc 系列为变量的大小，即变量在内存中从高到低连续排列
动态数据库，char x; int y; char z; 观察 xyz 间的地址间隔	15B 9B x->y->z	20B	7B 1B x->y->z	7B 1B x->y->z	7B 1B x->y->z	7B 1B x->y->z	不同编译器有不同的表现。VS 默认为相隔较远，且为固定常数（x86 与 x64 不同）。gcc 系列为变量的大小，即变量在内存中从高到低连续排列
动态数据库，int x; double y; int z; 观察 xyz 间的地址间隔	16B 12B x->y->z	36B 28B x->y->z	12B 4B x->y->z	12B 4B x->y->z	12B 4B x->y->z	12B 4B x->y->z	不同编译器有不同的表现。VS 默认为相隔较远，且为

							固定常数（x86 与 x64 不同）。gcc 系列为变量的大小，即变量在内存中从高到低连续排列
动态数据区，int k, a[10];若要使 a[x]就是 k 的地址，x 是几	12	-9	10	11	11	11	不同编译器有不同的表现。其中 Dev 双版本和 Linux 相同。VS x64 最特殊，变量从低到高定义，因此 x 为负数
静态数据区，char x,y,z; 观察 xyz 间的地址间隔	1B	1B	1B	1B	1B	1B	所有编译器表现相同，变量地址从低到高排列
静态数据区，int x,y,z; 观察 xyz 间的地址间隔	4B	4B	4B	4B	4B	4B	所有编译器表现相同，变量地址从低到高排列
静态数据库，char x; int y; char z; 观察 xyz 间的地址间隔	4B -3B x->y->z	4B -3B x->y->z	4B	4B	4B	4B	不同编译器有不同的表现。VS 会将变量位置重新排列以减小内存占用。gcc 系列统一为所有变量分配相同大小的内存
静态数据库，int x; double y; int z; 观察 xyz 间的地址间隔	8B -4B x->y->z	8B -4B x->y->z	8B	8B	8B	8B	不同编译器有不同的表现。VS 会将变量位置重新排列以减小内存占用。gcc 系列统一为所有变量分配相同大小的内存

静态数据区，int a[10], k;若要使 a[x]就是 k 的地址，x 是几	10	10	10	10	10	10	所有编译器表现相同，变量地址从低到高排列

★ 地址用 16 进制，注意 32 位编译器与 64 位编译器的差别
★ 地址之间的差值用 10 进制，转换为 K/M/G 等容易识别的单位即可
★ 某些编译器每次执行时具体地址不同，给出某一次的具体地址即可
★ 思考与心得体会中写出你的一些认识即可