C++11新的内置类型为long long，在vs2017里，长度为8个字节

例：程序C++11\_test1

*cout*<<"int size "<<sizeof(int) <<*endl*;

*cout*<<"long size "<<sizeof(long) <<*endl*;

*cout*<<"long long size "<<sizeof(longlong) <<*endl*;

输出为：

int size 4

long size 4

long long size 8

列表初始化：

例：程序C++11\_test1

// 列表初始化

int unit\_sold = 0;

int unit\_sold1{0}; // 列表初始化

longdouble ld = 3.1415926536;

//int a{ld}; // error，列表初始化不允许收缩转换

int b = 9;

// long double ld1{ b }; // error，列表初始化不允许收缩转换

constexpr：编译时判断是否为常量表达式

例：程序C++11\_test1

// constexpr编译时验证是否为常量表达式

constexpr int mf = 20;

//mf = 4; // error，mf为整型常量

int k = 1;

//constexpr int limit = k + 1; // error，k+1不为常量表达式

constexpr double df = 5.5;

// p为常量指针

constexpr int\* p = &k;

k = 2;

*cout*<< \*p<<*endl*; // 2

// p = nullptr; // error，不能改变p的指向

int k1 = 2;

// 下面是指向常量的指针

const int\* p2 = &k1;

// \*p2 = 5; // error，不能通过\*修改指针指向的对象

p2 = nullptr; // 可以修改指针本身

constexpr和const的区别（不一定全面）：

1. constexpr比const更严格，在编译时就确定是常量表达式。
2. 参考上面的程序，constexpr修饰指针时与const不同
3. constexpr可以修饰构造函数，const不能修饰构造函数。