C++11的一些新特性：

例：程序stl\_test2

// C++11中两个左括号之间没了空格

*vector*<*list*<int>> vec;

// 空指针

int\* ptr = nullptr;

auto关键字，类似C#中的var关键字。auto关键字必须被初始化。

auto i = 43;

*cout* << typeid(i).*name*() << *endl*; // int

*vector*<*string*> strVec;

strVec.*push\_back*("start");

auto begin = strVec.*begin*();

*cout* << typeid(begin).*name*() << *endl*;

一致性初始化：

// 一致性初始化，跟C#的赋值方式类似

int k; // 未初始化

int j{}; // 初始化为0

int\* q{}; // 初始化为nullptr

*cout* << j << *endl*; // 0

int values[] {1, 2, 3}; // 1,2,3

*vector*<int> v{ 2, 3, 5, 7 };

*cout* << v.*size*() << *endl*; // 4

int x1{ 5 }; // x1初始化为5

int x2 = 5.3; // x2为5，精度降低了

// 一致性初始化{}不能用于窄化

// 即精度降低或数值变动

//int x3{ 5.3 }; // error

double d1{ 5 }; // ok

构造函数的一致性初始化：

例：程序stl\_test3

class P1

{

public:

P1(int a, int b)

{

*cout* << "p1 constructor" << *endl*;

}

};

class P2

{

public:

P2(int a, int b)

{

*cout* << "p2 constructor" << *endl*;

}

// 初值列

P2(*initializer\_list*<int> list)

{

*cout* << "p2 initializer list" << *endl*;

}

};

int *main*(int argc, char\* argv[])

{

P1 p(3, 4); // ok

P1 p1{ 4, 5 }; // ok

// 如果构造函数加上一个explicit关键字

// 此处会报错，不能进行隐式转换

P1 p2 = { 5, 6 }; // ok，进行了隐式转换

P2 p3(1, 2); // p2 constructor

P2 p4{ 3, 4 }; // p2 initializer list

P2 p5{ 5, 6, 7, 8 }; // p2 initializer list

return 0;

}

从上面的结果P2的构造函数可以看出，如果“指明实参个数”和“指明初值列”同时存在，“指明初值列”的版本胜出。

Range-Based for循环，类似C#中的foreach

例：程序stl\_test4

class P

{

public:

P(int i)

{

num\_ = i;

*cout* << "P constructor" << *endl*;

}

void Print() const

{

*cout* << "P Print " << num\_ << *endl*;

}

private:

int num\_;

};

int *main*(int argc, char\* argv[])

{

int array[] {2, 3, 5, 7};

for (auto i : array)

{

*cout* << i << " ";

}

*cout* << *endl*;

*vector*<P> vec;

vec.*push\_back*({1});

vec.*push\_back*({ 2 });

vec.*push\_back*({ 3 });

for (const auto& elem : vec)

{

elem.Print();

}

// 等同于

for (*vector*<P> ::*iterator* iter = vec.*begin*(); iter != vec.*end*(); ++iter)

{

(\*iter).Print();

}

return 0;

}