C++11中auto通过变量的初始值来推算变量的类型，所以auto定义的变量必须有初始值。

例：C++11\_test2

// auto

auto x = 5; // x是int型

auto p = new auto(1); // p是int\*类型

const auto v = &x; // v是int \* const，常量指针

const auto \*v1 = &x; // v1是const int\*，指向常量的指针

v1 = nullptr;

int m = 9;

int& c = m;

auto d = c; // d为int，抛弃了引用

auto& h = c; // h为int&

const int n = 8;

auto e = n; // e为int，抛弃了const

const auto f = n; // f为const int

const int& k = 6;

auto& j = k; // j为const int&

auto\* l = &k; // l为const int\*

从上例中，可以看出auto推导表达式类型的规则：

1. 如果auto没有声明为引用，则会抛弃表达式const和引用，如d, e所示
2. 如果auto声明为引用，则会保留表达式的const和引用，如j所示
3. 如果auto声明为const，则会保留表达式的const，如f所示
4. 如果auto声明为指针，则会保留表达式的const，如l所示。

auto的限制：

例：程序C++11\_test2

// error，auto不能为函数形参

//void func(auto a = 1) {}

// 不能修饰非静态常量成员

struct MyStruct

{

//auto var = 1; // error

//const auto var3 = 3; // error

//static auto var4 = 4; // error

const static auto var2 = 2; // ok

};

//auto r[10] = { 0 }; // error， 不能定义数组

//vector<auto> vec = { 0 }; // error， 不能推导模板参数

auto的使用：

auto广泛应用于遍历容器时，如

for (auto c = map.begin(); c != map.end();++c)

{

}

当确定不了变量声明类型时，也可以使用auto：

例：程序C++11\_test2

class A

{

public:

static int Get()

{

return 0;

}

};

class B

{

public:

static *string* Get()

{

return "abc";

}

};

template<class T>

void Func()

{

// 此处无法获取到Get的返回类型，使用auto

auto val = T::Get();

}

Func<A>();

Func<B>();

decltype：在编译时推断表达式的类型，不会计算表达式

decltype不需要初始化。

例：程序C++11\_test1

// decltype类型指示符

// 不需要初始化

int x1;

decltype(x1) y1; // y1为int类型

const int ci = 0;

decltype(ci) x = 0; // x为const int

//x = 5; // error，x不能修改

const int& cj = ci;

decltype(cj) y = x; // y为const int&

int i = 42;

int\* p1 = &i;

int& r = i;

decltype(r + 0) b1; // ok，decltype(r)为引用，但r+0不是引用

//decltype(\*p) c1; // error，\*p为解引用操作，decltype将得到引用

//decltype((i))d; // error，decltype的变量加上1层或多层括号，得到的是引用

相比于auto，decltype不能丢掉引用和cv限定符（const, volatile）。

decltype和auto的结合使用：

例：程序C++11\_test2

// 返回类型后置

// auto作为返回值

// 通过decltype(t + u)推导返回值的类型

template<typename T, typename U>

auto Add(T t, U u) -> decltype(t + u)

{

return t + u;

}

int t = 1;

float u = 2.0f;

auto sum = Add(t, u);

参考：<http://blog.csdn.net/hailong0715/article/details/53928128>