C++中auto用法

1. 自动数据类型

auto num = 10； //int类型

auto num = 10.9； //double类型

1. 作为函数的返回值

template<class T1,class T2>

auto add(T1 a,T2 b)

{

return a+b;

}

1. for循环中用auto

for (auto i : b) Fuck(i);//注意这里是i，不是b(i) 等价于

for (auto bitch = std::begin(b); bitch != std::end(b); bitch++)

{

Fuck(\*bitch);

}

4.使用注意事项

①我们可以使用valatile，pointer（\*），reference（&），rvalue reference（&&） 来修饰auto

**[cpp]** [view plain](http://blog.csdn.net/huang_xw/article/details/8760403) [copy](http://blog.csdn.net/huang_xw/article/details/8760403)

[print?](http://blog.csdn.net/huang_xw/article/details/8760403)

1. auto k = 5;
2. auto\* pK = **new** auto(k);
3. auto\*\* ppK = **new** auto(&k);
4. **const** auto n = 6;

②用auto声明的变量必须初始化

**[cpp]** [view plain](http://blog.csdn.net/huang_xw/article/details/8760403) [copy](http://blog.csdn.net/huang_xw/article/details/8760403)

[print?](http://blog.csdn.net/huang_xw/article/details/8760403)

1. auto m; // m should be intialized

③auto不能与其他类型组合连用

**[cpp]** [view plain](http://blog.csdn.net/huang_xw/article/details/8760403) [copy](http://blog.csdn.net/huang_xw/article/details/8760403)

[print?](http://blog.csdn.net/huang_xw/article/details/8760403)

1. auto **int** p; // 这是旧auto的做法。

④函数和模板参数不能被声明为auto

**[cpp]** [view plain](http://blog.csdn.net/huang_xw/article/details/8760403) [copy](http://blog.csdn.net/huang_xw/article/details/8760403)

[print?](http://blog.csdn.net/huang_xw/article/details/8760403)

1. **void** MyFunction(auto parameter){} // no auto as method argument
3. **template**<auto T> // utter nonsense - not allowed
4. **void** Fun(T t){}

⑤定义在堆上的变量，使用了auto的表达式必须被初始化

**[cpp]** [view plain](http://blog.csdn.net/huang_xw/article/details/8760403) [copy](http://blog.csdn.net/huang_xw/article/details/8760403)

[print?](http://blog.csdn.net/huang_xw/article/details/8760403)

1. **int**\* p = **new** auto(0); //fine
2. **int**\* pp = **new** auto(); // should be initialized
4. auto x = **new** auto(); // Hmmm ... no intializer
6. auto\* y = **new** auto(9); // Fine. Here y is a int\*
7. auto z = **new** auto(9); //Fine. Here z is a int\* (It is not just an int)

⑥以为auto是一个占位符，并不是一个他自己的类型，因此不能用于类型转换或其他一些操作，如sizeof和typeid

**[cpp]** [view plain](http://blog.csdn.net/huang_xw/article/details/8760403) [copy](http://blog.csdn.net/huang_xw/article/details/8760403)

[print?](http://blog.csdn.net/huang_xw/article/details/8760403)

1. **int** value = 123;
2. auto x2 = (auto)value; // no casting using auto
4. auto x3 = **static\_cast**<auto>(value); // same as above

⑦定义在一个auto序列的变量必须始终推导成同一类型

**[cpp]** [view plain](http://blog.csdn.net/huang_xw/article/details/8760403) [copy](http://blog.csdn.net/huang_xw/article/details/8760403)

[print?](http://blog.csdn.net/huang_xw/article/details/8760403)

1. auto x1 = 5, x2 = 5.0, x3='r';  // This is too much....we cannot combine like this

⑧auto不能自动推导成CV-qualifiers（constant & volatile qualifiers），除非被声明为引用类型

**[cpp]** [view plain](http://blog.csdn.net/huang_xw/article/details/8760403) [copy](http://blog.csdn.net/huang_xw/article/details/8760403)

[print?](http://blog.csdn.net/huang_xw/article/details/8760403)

1. **const** **int** i = 99;
2. auto j = i;       // j is int, rather than const int
3. j = 100           // Fine. As j is not constant
5. // Now let us try to have reference
6. auto& k = i;      // Now k is const int&
7. k = 100;          // Error. k is constant
9. // Similarly with volatile qualifer

⑨auto会退化成指向数组的指针，除非被声明为引用

**[cpp]** [view plain](http://blog.csdn.net/huang_xw/article/details/8760403) [copy](http://blog.csdn.net/huang_xw/article/details/8760403)

[print?](http://blog.csdn.net/huang_xw/article/details/8760403)

1. **int** a[9];
2. auto j = a;
3. cout<<**typeid**(j).name()<<endl; // This will print int\*
5. auto& k = a;
6. cout<<**typeid**(k).name()<<endl; // This will print int [9]