C++11特性：auto关键字

阅读目录

正文

前言

本文的内容已经不新鲜了。关于auto，翻来覆去被人知道的都是这些东西，本文并没有提出新颖的auto用法。
本人原是痛恨博客一篇篇都是copy而来缺乏新意的探索，当然，本文不是copy而来，但发布这样一篇大家皆知的文章心里甚是惶恐。
本文对auto的内容加以整理，权当是自己的复习笔记了。

回到顶部

C++98 auto

早在C++98标准中就存在了auto关键字，那时的auto用于声明变量为自动变量，自动变量意为拥有自动的生命期，这是多余的，因为就算不使用auto声明，变量依旧拥有自动的生命期：

```
int a =10 ;  //拥有自动生命期
auto int b = 20 ;//拥有自动生命期
static int c = 30 ;//延长了生命期
```

C++98中的auto多余且极少使用，C++11已经删除了这一用法，取而代之的是全新的auto：变量的自动类型推断。

回到顶部

C++11 auto

auto可以在声明变量的时候根据变量初始值的类型自动为此变量选择匹配的类型，类似的关键字还有decltype。举个例子：

```
    int a = 10;
    auto au_a = a;//自动类型推断，au_a为int类型
    cout << typeid(au_a).name() << endl;
```

typeid运算符可以输出变量的类型。程序的运行结果输出了

```
    int
```

这种用法就类似于C#或java中的var关键字。auto的自动类型推断发生在编译期，所以使用auto并不会造成程序运行时效率的降低。而是否会造成编译期的时间消耗，我认为是不会的，在未使用auto时，编译器也需要得知右操作数的类型，再与左操作数的类型进行比较，检查是否可以发生相应的转化，是否需要进行隐式类型转换。

回到顶部

auto的用法

上面举的这个例子很简单，在真正编程的时候也不建议这样来使用auto，直接写出变量的类型更加清晰易懂。下面列举auto关键字的正确用法。

用于代替冗长复杂、变量使用范围专一的变量声明。

想象一下在没有auto的时候，我们操作标准库时经常需要这样：

```
#include<string>
```

```
#include<vector>
int main()
{
    std::vector<std::string> vs;
    for (std::vector<std::string>::iterator i = vs.begin(); i != vs.end(); i++)
    {
        //...
    }
}
```

这样看代码写代码实在烦得很。有人可能会说为何不直接使用using namespace std，这样代码可以短一点。实际上这不是该建议的方法（C++Primer对此有相关叙述）。使用auto能简化代码：

```
#include<string>
#include<vector>
int main()
{
    std::vector<std::string> vs;
    for (auto i = vs.begin(); i != vs.end(); i++)
    {
        //..
    }
}
```

for循环中的i将在编译时自动推导其类型，而不用我们显式去定义那长长的一串。

在定义模板函数时，用于声明依赖模板参数的变量类型。

```
template <typename _Tx,typename _Ty>
void Multiply(_Tx x, _Ty y)
{
    auto v = x*y;
    std::cout << v;
}
```

若不使用auto变量来声明v，那这个函数就难定义啦，不到编译的时候，谁知道x*y的真正类型是什么呢？

模板函数依赖于模板参数的返回值

```
template <typename _Tx, typename _Ty>
auto multiply(_Tx x, _Ty y)->decltype(_Tx*_Ty)
{
    return x*y;
}
```

当模板函数的返回值依赖于模板的参数时，我们依旧无法在编译代码前确定模板参数的类型，故也无从知道返回值的类型，这时我们可以使用auto。格式如上所示。

decltype操作符用于查询表达式的数据类型，也是C++11标准引入的新的运算符，其目的也是解决泛型编程中有些类型由模板参数决定，而难以表示它的问题。

auto在这里的作用也称为返回值占位，它只是为函数返回值占了一个位置，真正的返回值是后面的decltype(_Tx*_Ty)。为何要将返回值后置呢？如果没有后置，则函数声明时为：

```
decltype(_Tx*_Ty)multiply(_Tx x, _Ty y)
```

而此时_Tx,_Ty还没声明呢，编译无法通过。

回到顶部

注意事项

- auto 变量必须在定义时初始化，这类似于const关键字。
- 定义在一个auto序列的变量必须始终推导成同一类型。例如：

```
auto a4 = 10, a5 = 20, a6 = 30;//正确
auto b4 = 10, b5 = 20.0, b6 = 'a';//错误,没有推导为同一类型
```

使用auto关键字做类型自动推导时，依次施加一下规则：

- 如果初始化表达式是引用，则去除引用语义。

```
int a = 10;
int &b = a;

auto c = b;//c的类型为int而非int&（去除引用）
auto &d = b;//此时c的类型才为int&
```

片了。好诡异...

--善良超哥哥

2. Re:图说C++对象模型：对象内存布局详解

@amyliushu和编译器有关，在GCC下没有0x00000000,在vc++下有...

--melonstreet

3. Re:图说C++对象模型：对象内存布局详解

@善良超哥哥可能是你网络的问题，我这里看图片都是好的:-)...

--melonstreet

4. Re:图说C++对象模型：对象内存布局详解

简直太棒~~~但是我有一个问题 就是关于虚继承内存布局中那用来隔开的四个字节的0 ，为啥我写小测试，没有检测到这四个字节，恳请大大赐教~~

--amyliushu

5. Re:图说C++对象模型：对象内存布局详解

博主，图都挂了，重新编辑一下吧

--善良超哥哥

6. Re:C++11特性：decltype关键字

decltype(arr)是数组才对吧

--默默敲键盘&C

7. Re:图说C++对象模型：对象内存布局详解

6.2中，B1类对象内存布局第二行vbptr单词写错了，还有对应运行结果，[4]B::vptr的值应该是007CFDE00吧，因为对应代码+4写成+3了，有点强迫症的我。。。

--yiyiyi4321

8. Re:图说C++对象模型：对象内存布局详解

好文，感谢，5.2.2 菱形继承下的C++对象模型，图中B2:vptr表的D::f1()，应该是D::f2()吧

--yiyiyi4321

9. Re:数据结构图文解析之：哈夫曼树与哈夫曼编码详解及C++模板实现

@Listenmer是的。多谢指出，已经更正。...

--melonstreet

10. Re:数据结构图文解析之：哈夫曼树与哈夫曼编码详解及C++模板实现

“出现频率较高的源符号采用较短的编码，出现频率较高的符号采用较长的编码”，　这是一样的啊，后面那个是频率较低的吧

--Listenmer

阅读排行榜

1. 图说C++对象模型：对象内存布局详解(7889)
2. C++11特性：decltype关键字(7557)
3. Unity 移动端触摸屏操作(6571)
4. C++内联函数(5232)
5. C++11特性：auto关键字(4847)
6. 细说new与malloc的10点区别(4343)
7. C++ 引用计数技术及智能指针的简单实现(3885)
8. Socket编程（4）TCP粘包问题及解决方案(3180)
9. C++ 隐式类类型转换(2828)
10. 数据结构图文解析之：队列详解与C++模板实现(2784)

评论排行榜

1. 图说C++对象模型：对象内存布局详解(28)
2. 漫谈C++：良好的编程习惯与编程要点(13)
3. 数据结构图文解析之：哈夫曼树与

```
c = 100;//a =10;
d = 100;//a =100;
```

- 如果初始化表达式为const或volatile（或者两者兼有），则除去const/volatile语义。

```
const int a1 = 10;
auto  b1= a1; //b1的类型为int而非const int（去除const）
const auto c1 = a1;//此时c1的类型为const int
b1 = 100;//合法
c1 = 100;//非法
```

- 如果auto关键字带上&号，则不去除const语意。

```
const int a2 = 10;
auto &b2 = a2;//因为auto带上&，故不去除const，b2类型为const int
b2 = 10; //非法
```

这是因为如何去掉了const，则b2为a2的非const引用，通过b2可以改变a2的值，则显然是不合理的。

- 初始化表达式为数组时，auto关键字推导类型为指针。

```
int a3[3] = { 1, 2, 3 };
auto b3 = a3;
cout << typeid(b3).name() << endl;
```

程序将输出

```
int *
```

- 若表达式为数组且auto带上&，则推导类型为数组类型。

```
int a7[3] = { 1, 2, 3 };
auto & b7 = a7;
cout << typeid(b7).name() << endl;
```

程序输出

```
int [3]
```

- 函数或者模板参数不能被声明为auto

```
void func(auto a)  //错误
{
//...
}
```

- 时刻要注意auto并不是一个真正的类型。

auto仅仅是一个占位符，它并不是一个真正的类型，不能使用一些以类型为操作数的操作符，如sizeof或者typeid。

```
cout << sizeof(auto) << endl;//错误
cout << typeid(auto).name() << endl;//错误
```

分类: C++

melonstreet
关注 - 74
粉丝 - 144

« 上一篇：图说C++对象模型：对象内存布局详解
» 下一篇：C++11特性：decltype关键字

posted @ 2015-11-09 21:29 melonstreet 阅读(4847) 评论(4) 编辑 收藏

哈夫曼编码详解及C++模板实现(11)
4. 细说new与malloc的10点区别(11)
5. C++编译期多态与运行期多态(8)
6. 关于传值与传引用的讨论(5)
7. C++ 引用计数技术及智能指针的简单实现(5)
8. C++11特性：decltype关键字(4)
9. C++11特性：auto关键字(4)
10. GC基本算法及C++GC机制(3)

推荐排行榜

1. 细说new与malloc的10点区别(18)
2. 图说C++对象模型：对象内存布局详解(16)
3. 数据结构图文解析之：哈夫曼树与哈夫曼编码详解及C++模板实现(10)
4. 漫谈C++：良好的编程习惯与编程要点(9)
5. C++ 自由存储区是否等价于堆？(8)
6. C++ 异常机制分析(6)
7. C++编译期多态与运行期多态(5)
8. 浅谈 GPU图形固定渲染管线(5)
9. GC基本算法及C++GC机制(5)
10. C++ 引用计数技术及智能指针的简单实现(5)

评论列表

#1楼 2015-11-09 22:02 马三小伙儿

学习啦。

支持(0) 反对(0)

#2楼[楼主] 2015-11-09 22:11 melonstreet

@ 马三小伙儿
o(^▽^)o旧知识啦

支持(0) 反对(0)

#3楼 2015-11-10 15:29 cposture

不错，但是注意事项中第5点是不是有点问题？
"这是因为不是去掉了const，则b2为a2的非const引用，通过b2可以改变a2的值，则显然是不合理的。"中"不是"应该为"如果"吧

支持(0) 反对(0)

#4楼[楼主] 2015-11-10 15:40 melonstreet

@ cposture
多谢指出，已改正。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。