前 几 天 看 代 码 看 到 const\_cast，当 时 只 把 它 当 做 去 除 常 量 性 的 一 个 转 化，没 有 细 细 研 究。今 天 无 意 中 想 到 这 个，想 到 之 前 看 到 网 上 有 说 定 义 了 常 量 后，将 去 除 常 量 性 的 值 传 给  另 一 个 变 量 ， 即 使 新 变 量 发 生 了 变 化 旧 变 量 也 不 发 生 改 变，我 就 觉 得 很 奇 怪，这 个 是 如 何 设 计 的 ？于 是，就想 着 尝 试 一 下，可惜 我 的 测 试 结 果 是 两 者 一 起 变 化 的（ 可 能 还 是 不 够 详细）。  
  
格式const\_cast<type\_id>(expression)  
其 中 对 我 而 言 最 主 要 的 是 type\_id了，它 可 以 是 指 针 ，可 以 是 引 用，也 可 以 是point-to-data-member。  
  
  
先贴一下代码：  
#include<iostream>//  
using namespace std;  
  
class A  
{  
public:  
    int m;  
      
    A(int l)  
    {  
        m = l;  
    }  
};  
  
int main(int , char \*\*)                       //C++には、関数の引数に対して、使わない引数の名前はいらない  
{  
      
    #if 0    
    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Test 1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
    const char \*a = "1234";  
  
    char \*b = const\_cast<char \*>(a);  
  
    cout << "a=" << a << "; b= " << b << endl;  
  
  
    b[0] = 'a';  
  
    cout << "a=" << a << "; b= " << b << endl;  
    #endif  
  
    A l = 10;                                        //もし　コンストラクタA(int l){}がなければ、 『エラー: ‘int’ から非スカラ型 ‘A’ への変換が要求されました』が出る  
  
    const A \*a = new A(l);  
    //a.m = 100;  
  
    A \*b = const\_cast<A\*>(a);                        //const\_cast<type\_id>(expression)   type\_idはポインタ、参照型、もしくはpoint-to-data-memberでなければなりません  
  
    cout << "a.m:" <<a->m << "; b.m= " << b->m << endl;  
  
    //a->m = 300;                                            //  エラー: 読み取り専用オブジェクト内のメンバ ‘A::m’ への代入です a->m = 300;  
  
  
    cout << "a.m:" <<a->m << "; b.m= " << b->m << endl;        // 10 10  
  
    b->m = 200;  
  
    cout << "a.m:" <<a->m << "; b.m= " << b->m << endl;    // 200 200  
  
    //int A::\*p\_m = &A::m;                      
    //cout << "a.m:" <<a->m << "; b.m= " << b->m << "a.p\_m:" <<\*a->p\_m << "; b.p\_m= " << \*b->p\_m<< endl;    // エラー: ‘const class A’ has no member named ‘p\_m’  
  
  
    int A::\*p\_m = &A::m;                                    //point-to-data-member  
    //a->\*p\_m                      
    cout << "a.m:" <<a->m << "; b.m= " << b->m << "a.p\_m:" <<a->\*p\_m << "; b.p\_m= " << b->\*p\_m<< endl;     //OK  
  
  
    return 0;  
}  
  
在这个测试中，  
1.首先int main(int , char \*\*)                                          
这一行，定义的时候也未 对 形 参 命 名，之 前 的 博 客 中 好 像 有 写 到 ，C + + 的 一 个 特 别 之 处就 是 在 不 使 用形 参 的 情况 下 ， 定义 函 数 的 时 候也 可 以 省 略 形参 的 函 数名 。  
2. 其次是A l = 10;这一行，这一行是隐式转换，之 前 看 到 说 这 种形  式可 以 隐 式 的 将 等 号 右 边 的 普 通 变 量  转 换 成 类 的对 象  （对象 成 员 的 赋 值） 。 不 过 ，  在没 有 带 参 数 的构  造函 数 A (int l){m=l;}的情 况下 是 出 错  的， 必须 提 供 带参 的 构 造 函数。  
3.对象变量b是除去常量 性 质 后的 变 量  ，a 是 原 始 的 常 量 对 象， 对 a 进 行 修 改 编译 器  会报 错，对b的成员修改没有问题，最终结果是两者一样。  
当 然， 一 个 失 败 的案  例是 变 量 a和 b均是int \*型， 这 时即使对 b进行 修 改，也会 有问 题，编 译器 (g++4.8.2)不会 报 错， 但 是 运 行会 出 现 段 错误。  
4.point-to-data-member,这个我是第一次见到， 具体 中 身 是 什么  样还 没 有 调 查，只 是 参 照Stack Overflow上的代 码 尝试 了 一下。