Lesson10---C++的类型转换

【本节目标】

- 1. C语言中的类型转换
- 2. C++强制类型转换
- 3. 为什么需要强制类型转换
- 4. RTTI

1. C语言中的类型转换

> 在C语言中，如果**赋值运算符左右两侧类型不同，或者形参与实参类型不匹配，或者返回值类型与接收返回值类型不一致时，就需要发生类型转化**，C语言中总共有两种形式的类型转换：**隐式类型转换和显式类型转换。**
>
> 1. 隐式类型转化：编译器在编译阶段自动进行，能转就转，不能转就编译失败
> 2. 显式类型转化：需要用户自己处理

```
 void Test ()
{
      int i = 1;
      // 隐式类型转换
      double d = i;
      printf("%d, %.2f\n" , i, d);

      int* p = &i;
      // 显示的强制类型转换
      int address = (int) p;

      printf("%x, %d\n" , p, address);
}
```

> **缺陷：**
> **转换的可视性比较差，所有的转换形式都是以一种相同形式书写，难以跟踪错误的转换**

2. 为什么C++需要四种类型转换

> C风格的转换格式很简单，但是有不少缺点的：
>
> 1. 隐式类型转化有些情况下可能会出问题：比如数据精度丢失
> 2. 显式类型转换将所有情况混合在一起，代码不够清晰
>
> 因此C++提出了自己的类型转化风格，注意**因为C++要兼容C语言，所以C++中还可以使用C语言的转化风格。**

3. C++强制类型转换

> **标准C++为了加强类型转换的可视性，引入了四种命名的强制类型转换操作符：**
>
> **static_cast、reinterpret_cast、const_cast、dynamic_cast**

3.1 static_cast

> static_cast用于非多态类型的转换（静态转换），编译器隐式执行的任何类型转换都可用static_cast，但它不能用于两个不相关的类型进行转换

```
int main()
{
  double d = 12.34;
  int a = static_cast<int>(d);
  cout<<a<<endl;
  return 0;
}
```

3.2 reinterpret_cast

> reinterpret_cast操作符通常为操作数的位模式提供较低层次的重新解释，用于将一种类型转换为另一种不同的类型

```
int main()
{
    double d = 12.34;
    int a = static_cast<int>(d);
    cout << a << endl;

    // 这里使用static_cast会报错，应该使用reinterpret_cast
    //int *p = static_cast<int*>(a);
    int *p = reinterpret_cast<int*>(a);

    return 0;
}
```

3.3 const_cast

> const_cast最常用的用途就是删除变量的const属性，方便赋值

```
void Test ()
{
  const int a = 2;
  int* p = const_cast< int*>(&a );
  *p = 3;

  cout<<a <<endl;
}
```

3.4 dynamic_cast

> dynamic_cast用于将一个父类对象的指针/引用转换为子类对象的指针或引用(动态转换)
>
> 向上转型：子类对象指针/引用->父类指针/引用(不需要转换，赋值兼容规则)
> 向下转型：父类对象指针/引用->子类指针/引用(用dynamic_cast转型是安全的)
> 注意：
> 1. dynamic_cast只能用于父类含有虚函数的类

2. dynamic_cast会先检查是否能转换成功，能成功则转换，不能则返回0

```
class A
{
public :
virtual void f(){}
};

class B : public A
{};

void fun (A* pa)
{
// dynamic_cast会先检查是否能转换成功，能成功则转换，不能则返回
B* pb1 = static_cast<B*>(pa);
B* pb2 = dynamic_cast<B*>(pa);

cout<<"pb1:" <<pb1<< endl;
cout<<"pb2:" <<pb2<< endl;
}

int main ()
{
  A a;
  B b;
  fun(&a);
  fun(&b);
  return 0;
}
```

注意
强制类型转换关闭或挂起了正常的类型检查，每次使用强制类型转换前，程序员应该仔细考虑是否还有其他不同的方法达到同一目的，如果非强制类型转换不可，则应限制强制转换值的作用域，以减少发生错误的机会。**强烈建议：避免使用强制类型转换**

4. RTTI（了解）

RTTI：Run-time Type identification的简称，即：运行时类型识别。

C++通过以下方式来支持RTTI：

1. **typeid运算符**
2. **dynamic_cast运算符**
3. **decltype**

5. 常见面试题

1. C++中的4中类型转化分别是：_____、_____、_____、_____
2. 说说4中类型转化的应用场景。