

CS481 - Object-Oriented Programming

RTTI

- ❖ Run-time type identification
 - Run-time type inquiry??
- ❖ Identifies “real” object type
 - Even if all we have is a base pointer
- ❖ A “secondary” feature?
 - Not meant to replace virtual functions

The Problem?

```

graph BT
    Derived --> Base
    
```

- ❖ Upcasting works fine
 - Treating derived type as a base
- ❖ Downcasting?
 - Have base pointer to object
 - Want to treat as derived? Is it safe?
 - ♦ E.g., `cp = (Circle*) shape_ptr;`

Simple RTTI?

```

class Shape
{
    ...
    virtual const char* ID() const;
}

const char* Line::ID()
{ return "Line"; }

const char* Circle::ID()
{ return "Circle"; }
    
```

RTTI: typeid

```
#include <typeinfo.h>
```

- ❖ C++ operator
 - Like sizeof
 - Takes expression (or type name)
- ❖ Returns `const type_info&`
 - Member functions `name()`, `before()`
 - Operators `==`, `!=` (compare to known type?)

RTTI: typeid

```

const Shape* ptr;
ShapeListIter iter1(list);
while ((ptr = *iter1++) != NULL)
{
    outfile << typeid(*ptr).name();
}
    
```

```

Circle
Line
Rectangle
FancyLine
        
```

Borland C++ output; not guaranteed by standard!!
Check your compiler's spec

type_info.before()

- ❖ To sort `type_info` objects
 - E.g., build tree for lookup
 - Values may change each program run!
- ❖ Inheritance relationship??
 - No!! (according to ARM/Stroustrup)

CS481 - Object-Oriented Programming

Dynamic Cast

- ❖ Permits type-safe downcast
 - Shape* sp = ???;
 - Circle* cp =
dynamic_cast<Circle*>sp;
- ❖ Is object of specified type?
 - If so, returns pointer (ref?) to derived
 - Else, returns NULL (or exception if ref)

7

Dynamic Cast

- ❖ What about intermediates?
 - Shape* sp = new FancyLine;
 - Line* lp =
dynamic_cast<Line*>sp;
- ❖ Returns desired pointer
 - Don't need to know class hierarchy
 - Can treat as intermediate-level class

8

RTTI Usage

- ❖ Not virtual function substitute
 - Don't use typeid for if/else if/.../else
- ❖ However ... (possible RTTI use)
 - Want to add a derived class
 - Need to add virtual function
 - But don't "own" base class to modify

9

Other New Casts

- ❖ Cast expression to type T

10

Other New Casts

- ❖ Cast expression to type T
 - static_cast<T> exp
 - ◆ If implicit conversion exists (⇔)


11

Other New Casts

- ❖ Cast expression to type T
 - static_cast<T> exp
 - ◆ If implicit conversion exists (⇔)
 - const_cast<T> exp
 - ◆ Change const and/or volatile

12

CS481 - Object-Oriented Programming

 *Other New Casts*

- ❖ Cast expression to type T
 - static_cast<T> exp
 - ◆ If implicit conversion exists (⇔)
 - const_cast<T> exp
 - ◆ Change const and/or volatile
 - reinterpret_cast<T> exp
 - ◆ Similar to (T)exp

13

