CS 246 Fall 2015 - Tutorial 11

November 27, 2015

1 Summary

- Casting

2 Casting

- In addition to C-style casts, there are 4 more types of casting in C++.
- Casts should be avoided except absolutely necessary.
  1. `const_cast`: used to add or remove const from its parameter
     - Only C++ cast that can remove `const`
     - Warning: this may not behave the way you expect in all cases. Should only be used to remove const from an object to pass to a function which accepts a non-const object but doesn't actually modify it. Modifying the object that you cast const away from is undefined behaviour. (See const-cast-undefined.cc)
  2. `dynamic_cast`: used for pointers and references, ensures result of cast is actually of appropriate type
     - Uses runtime type information (RTTI) to determine cast validity
     - Returns NULL if the object is not of the type you are trying to cast to.
  3. `static_cast`: convert between related classes (base to derived). Can also perform implicit conversions (int to double).
     - Primarily used for reasonable casts, where we can be sure that the cast is meaningful and valid
     - Note: `static_cast` is not checked (and is equivalent in many cases to C-style casts).
  4. `reinterpret_cast`: cast any pointer to any other pointer (EXTREMELY unsafe - should be used sparingly)
     - Typically, only of use when doing systems programming
     - fun example: duck.cc
- Generally, it is bad style to have code that looks like:

```
int foo(Base* bp){
  if(! dynamic_cast<Derived1*>(bp)){
    ...
  } else if (! dynamic_cast<Derived2*>(bp)){
    ...
  } else if (! dynamic_cast<Derived3*>(bp)){
    ...
  } ...
}
```

- This function is tightly coupled to the particular class hierarchy and requires modification whenever the hierarchy is modified
- We've seen better mechanisms to perform similar actions (e.g. Visitor pattern, virtual methods)

- Also, recall from our discussions of exceptions that `bad_cast` can be thrown when `dynamic_cast` fails

```
try{
  Base& br = ...; // Something
  Derived1& dr = dynamic_cast<Derived1&>(br);
}catch (bad_cast& e){
  cerr << "DANGER, WILL ROBINSON. DANGER" << endl;
}
```

- Why **must** `dynamic_cast` throw an exception here?
- Finally, we have to be careful not to shoot ourselves in the foot with `static_cast`:

```
Base* bp = new Derived2;
Derived1* d1p = static_cast<Derived1*>(bp);
```

- This will compile but it may not work as we intend or expect.