Practicum 10 - Casting

Assignment Details

Assigned: April 8th, 2014. Due: April 10th, 2014 at midnight.

A complete submission will include the modified code from the practicum.

Practicum

Ok, in this practicum, we're going to look, experimentally, at the results of the explicit C++ casts on object pointers.

Update your source control, and you'll find a la10 directory with a la10.cpp file which has some base code in it. The code defines two classes – A and B, with B a derived class of A, and then allocates two objects on the heap, and then saves pointers to those objects in two A pointers, and then outputs the value of these pointers to the console.

Step 1

Declare two B pointers; assign a C-style cast of pA1 to a B pointer to the first, and a C-style cast of pA2 to a B pointer to the second. Output both of these pointers to the console.

Step 2

Declare two more B pointers; assign to one a dynamic_cast to a B pointer of pA1, and the same of pA2 to the second. Output both of these to the console.

Step 3

Declare two more B pointers; assign to one a static_cast to a B pointer of pA1, and the same of pA2 to the second. Output both of these to the console.

Step 4

Repeat the "declare two pointers, cast them, and output them to the console" deal, this time with reinterpret_cast.

Step 5

Look at the output of the program. You should see three different results from the casts[1]. Note (by commenting within the .cpp file) which of the eight variables you declared are actually valid, and *why*.

[1] It is worth mentioning that the exact values you get from the cast are, technically, implementation specific; some compilers can give more than three different pointers in this exercise, but GCC on the multilab machines should give three.

Step 6

Pick a variable of each of the three unique pointer values you got from steps 1-4. On each pointer, call:

- doSomething
- doSomethingElse
- doAnotherThing

Note that you will not be able to do all of these in the same execution of the program!

Note (again, via comments in the .cpp file) what the result of each of these calls was.

Step 7

Of the nine calls tested in Step 6, three of these should be "correct" in as much as they do exactly what's expected, as the pointer is valid and points to an object of the correct type.

For the other six, *briefly* explain (or provide a reasonable guess![2]) what is happening when the method gets called – add these explanations as comments in the .cpp file.

[2] The important thing is to think about what happens in these unusual cases. Hopefully the weirdness demonstrates, to some extent, why dynamic_cast is the cast to use in this situation…