Type Casting in C++

A cast in programming is a special operator that manipulates one data type into another for example from a double to an integer. Here are the 5 different ways of type casting in C++ and when they should be used.

Implicit Conversion

Implicit conversion occurs when one type is compatible with another and a cast can occur automatically. An example of this can be seen below where these two variables are compatible and no loss of data occurs during the conversion.

```
// implicit conversion
int i = 300;
long g = 500000;
g = i;
```

Regular C style cast

This is the standard way of casting in C and not so much in C++. By placing brackets with the targeted conversion type in front of the variable you are changing a cast can be done. This has downfalls as it can be abused and checks on these casts are only preformed at runtime meaning your entire application can crash when executed. To overcome this and many other issues 4 additional ways of casting was created by the C++ developers.

```
// regular C style cast
int c = 50;
double m = (double)c;
```

Dynamic cast

Dynamic cast was introduced to fit in with the fact that C++ was an object orientated language. Standard C style casts were not going to cut it in an inheritance/ hierarchy environment. This particular cast is aimed at converting pointers/references within polymorphism. If you have multiple levels of inheritance and wish to convert these types in any direction across your classes it can be done with dynamic casting. The following snippet below is taken from the C++ reference site. It shows how this cast works across different structs that inherit from each other, here is the casting in action.

```
A& a = d; // upcast, dynamic_cast may be used, but unnecessary
D& new_d = dynamic_cast<D&>(a); // downcast
B& new_b = dynamic_cast<B&>(a); // sidecast
```

Static Cast

Static cast is the C++ style of preforming standard casting from one type to another. This is used for ordinary type conversions. The compiler already knows how to convert from an integer to a long and vice versa. The big difference (which was the main flaw of the C style cast) is that the compiler checks this cast at compile time not runtime. It is also much more readable then a standard cast and gives more definition to the type of cast you wish to perform. As you can see in the example below it is very easy to identify.

```
// static cast
int n = static_cast<int>(8.2);
```

Reinterpret Cast

This form of cast is used to read low level bits of memory. An example of this would be comparing the bytes in memory for an integer variable and the bytes for a short. By looking at the underlying bit pattern a conversion may be able to occur. It differs from static cast in that it this cast expression does not compile to any CPU instructions that informs the compiler to treat one sequence of bits as if it was another type.

References

http://stackoverflow.com/questions/332030/when-should-static-cast-dynamic-cast-const-cast-and-reinterpret-cast-be-used

http://en.cppreference.com/w/cpp/language/dynamic_cast

https://www.tutorialspoint.com/cplusplus/cpp_casting_operators.htm

http://www.cplusplus.com/doc/oldtutorial/typecasting/

http://stackoverflow.com/questions/573294/when-to-use-reinterpret-cast

http://en.cppreference.com/w/cpp/language/reinterpret_cast