Typecasting

- Typecasting is the concept of converting the value of one type into another type. For example, you might have a float that you need to use in a function that requires an integer.
- *Implicit conversion:* Almost every compiler makes use of what is called automatic typecasting. It automatically converts one type into another type. If the compiler converts a type it will normally give a warning. For example this warning: *"conversion from 'float' to 'int', possible loss of data."*
- The problem with this is, that sometimes you get a warning (normally you want to compile without warnings and errors) and you are not in control.

```
int intVal;
float fltVal = 2.55;
intVal = fltVal;      // implicit conversion
cout << intVal << endl;
```

- After the implicit conversion `intVal` will have a value of `2`.
- Note C++ floors the value rather than rounds the value, all decimal places are lost.

- *Explicit conversion:* The C and C++ languages have ways to give you back control. This can be done with what is called an explicit conversion. Sure you may still lose data, but you decide when to convert to another type and you don't get any compiler warnings.
- There are two main syntaxes for typecasting: functional and c-like:

```
int intVal;
float fltVal = 10.3;
intVal = int(fltVal);     // functional notation
intVal = (int) fltVal;    // c-like cast notation
```