- Reinterpret cast

```
reinterpret_cast<...>(var);
```

⊕ will allow anything (with consequences), retyping if initially don't know real type

⊖ will reinterpret anything into anything

- Const cast

```
const_cast<int *>(b);
```

⊕ can make a const changeable

⊖ bad sign, need good reason, redesign code instead. (ex: external library compatability).

* Type cast operators:

```
operator float() { return this->_v; }
operator int() { return static_cast<int>(this->_v); }
```

not everyday but,

↳ elegant way of pimping your class to make behavior more natural

→ new syntax that allows you to define w/in class specific operators, to do implicit conversions toward a type that interests you.

* Explicit ~~cast~~ keyword:

cast are conceptually similar to constructors (turn data types into new data type

explicit keyword in front of function in class

↳ won't allow implicit conversion in constructor for ex.

↳ will need an explicit promotion / demotion.