CPP06 — Casts

* types cast operators
* 5 casts w/ special properties

C → proposes two casts implicit & explicit → unreliable behavior

*** Type conversion:**

ex:
```
int a = 42;
double b = a;        → Implicit conversion cast (lost precision in some cases)
double c = (double) a → Explicit conversion cast
double d = a;        → Implicit promot° (OK)
int e = d;           → Implicit demotion (Hazardeous → error)
int f = (int) d;     → Implicit demotion (ok, dev's choice)
```
flag to force and forbid

*** Type reinterpretation:**

↳ keeps bits same like void *... so float* → void* → int* → completely diff no, same bits through addresses there are hierarchy ex. void * most general, promot°/demot° impl/exp

⚠ be careful & use Explicit for demotions to signal intentionality

*** Type qualifiers:**

↳ special case of a reinterpretation → qualifier → keyword const

int → const int all fine const* to int → only if explicit demotion not implicit

mostly used to accomodate library code w/ or w/out const.

C++

*** Upcast & Downcast** (promotion & demotion with classes)

New hierarchy of types & classes., ex: parents, grand parents more & more general

child → parent all fine parent → child → only explicit, & would allow cast into wrong child exec° ptrs.

- Static Cast: (most popular)

ex int a → double b int d = static_cast<int>(b); → also can be used child → parent → child)

⊖ can't make sure the right child is casted

⊕ can make sure it's related, will stop compilation.

- Dynamic Cast: (happens at run time exec not compilat°)

<dynamic_cast<child&>(b);

⊖ can fail at execution so has to be handled in code

can only work in a Polymorphic instance, 1 of mbr fct has to be virtual → by subtyping.

⊕ can make sure the right child is casted. can throw/catch std::bad_cast &bc)