C++ type-casting
Real-Time Type Information

- Version 1: Dr. Ofir Pele
- Version 2: Dr. Miri Ben-Nissan
- Version 3: Dr. Erel Segal-Halevi

C++ style casting

- `reinterpret_cast<type>(expression)`
- `static_cast<type>(expression)`
- `dynamic_cast<type>(expression)`

‘reinterpret_cast‘ operator

```
reinterpret_cast<type>(expression)
```

- Reinterpret byte patterns.
- Circumvents type checking.
- Implementation-dependent.
- (Should be) used rarely.
- Legitimate use example: writing image files (folder 4).
- Very dangerous! (folder 5).

The ‘static_cast’ operator (folder 4)

```
static_cast<type>(expression)
```

When conversion method is known during compilation:

- double → int, int → double, etc.
- Conversion operator / conversion constructor.
- up-cast – Circle → Shape, Circle* → Shape*.

Safer that “old-style” casts

- e.g. won’t cast `int*` to `float*`

Failure causes a compiler error

- No dynamic checking is done

static_cast vs reinterpret_cast

reinterpret_cast does not do anything at runtime.

static_cast does at runtime a conversion determined at compile time.

Copy&paste into godbolt.org to see.

```
int main() {
    int i = 5;
    double d;
    d = (double)i;
    d =
static_cast<double>(i);
    int& ir = i;
    double& dr0 =
(double&)ir;
    double& dr =
reinterpret_cast<double&>(
ir);
}
```

The ‘dynamic_cast‘ operator

```
dynamic_cast<T>(expression)
```

Enables run-time type checking:

When `expression` is a **pointer**:

- Returns a valid pointer if `expression` really points to type `T`
- null pointer value otherwise

The ‘dynamic_cast‘ operator

```
dynamic_cast<T>(expression)
```

Enables run-time type checking:

When `expression` is a **reference**:

- Returns a valid reference if `expression` is really of type `T`
- Throws an exception when it fails (“`bad_cast`”)

dynamic_cast : example

```
Shape* s = container.pop();
Circle* c = dynamic_cast<Circle*>(s);
if (c != nullptr) {// c is a circle
    c->setRadius(42);
} else {
    ...
}
```

dynamic_cast: only for polymorphics

```
class Circle : public Shape
{
   virtual void draw();
}
class Date : public Time
{
    // Time has no virtual functions
}
void foo(Shape * s, Time * t)
{
   Circle * c =
    dynamic_cast<Circle*>( s ); //ok
   Date * date =
    dynamic_cast<Date*>( t ); //compilation error
}
```

Cast comparison

	Compile-time	Run-time
reinterpret_cast	Check that source, target are pointers / refs	nothing
static_cast	Check that there is a conversion source → target	Fixed conversion
dynamic_cast	Check that the class is polymorphic	Expensive check