

Section 2.1 C++11

Braced Init

- As written, the example uses **copy initialization** — `T obj = val;` — and will thus fail to compile if `T` is not implicitly convertible from `U`.
- Switching to **direct initialization** — `T obj(val);` — would allow explicit constructors to also be considered.
- Using **direct list initialization** — `T obj{val};` — would allow aggregates to be supported as well as explicit constructors but not **narrowing conversions**; `initializer_list` constructors are also considered and preferred.
- Switching to **copy list initialization** — `T obj = {val};` — would allow aggregates to be supported but would result in an error if an explicit constructor is the best match, rather than considering the non**explicit** constructors for the best viable match. An error would arise if a **narrowing conversion** is required; `initializer_list` constructors are also considered and preferred.

Table 2 summarizes the different initialization types and highlights the options and trade-offs. In general, there is no one true, universal syntax for initialization in **generic** template code. The library author should make a deliberate choice among the trade-offs described in this section and document that as part of their **contract**.

Table 2: Summary of the different initialization types

Initialization Type	Syntax	Aggregate Support	Explicit Constructor Used	Narrowing	initializer_list Constructor Used
Copy	<code>T obj = val;</code>	only if <code>T==U</code>	no	allow	no
Direct	<code>T obj(val);</code>	only if <code>T==U</code>	yes	allow	no
Direct List	<code>T obj{val};</code>	yes	yes	error	yes
Copy List	<code>T obj = {val}</code>	yes	error if best match	error	yes

Uniform initialization in factory functions

As stated above, one of the design concerns facing an author of **generic** code is which form of syntax to choose to initialize objects of a type dependent on **template parameters**. Here we present the different trade-offs to consider when writing a **factory function** that takes an arbitrary set of type **arguments** to create an object of a user-specified type: