Wording for class template argument deduction for aggregates

Timur Doumler (papers@timur.audio)

Document #: P1816R0
Date: 2019-07-17
Project: Programming Language C++
Audience: Core Working Group

Abstract

This paper provides wording for class template argument deduction for aggregates [P1021R4].

Proposed wording

The proposed changes are relative to the current C++20 working draft [N4820].

In [over.match.class.deduct], append to paragraph 1 as follows:

— For each *deduction-guide*, a function or function template with the following properties:

  — The template parameters, if any, and function parameters are those of the *deduction-guide*.

  — The return type is the *simple-template-id* of the *deduction-guide*.

In addition, if `C` satisfies the conditions for an aggregate class with the assumption that any dependent base class has no virtual functions and no virtual base classes, and the initializer is a non-empty *braced-init-list* or parenthesized *expression-list*, the set contains an additional function template, called the *aggregate deduction candidate*, defined as follows. Let $x_1, ..., x_n$ be the elements of the *initializer-list* or *designated-initializer-list* of the *braced-init-list*, or of the *expression-list*. For each x_i, let e_i be the corresponding element of `C` or of one of its (possibly recursive) subaggregates that would be initialized by x_i ([dcl.init.aggr]) if brace elision is not considered for any subaggregate that has a dependent type. If there is no such element e_i, the program is ill-formed. The aggregate deduction candidate is derived as above from a hypothetical constructor $\texttt{C}(\texttt{T}_1, ..., \texttt{T}_n)$, where $\texttt{T}_i$ is the declared type of the element e_i.

In [over.match.class.deduct], paragraph 3, add to the example as follows:

```
B b{(int*)0, (char*)0};         // OK, deduces B<char*>

template <typename T>
struct S {
    T x;
    T y;
};
```

```
template <typename T>
struct C {
    S<T> s;
    T t;
};

template <typename T>
struct D {
    S<int> s;
    T t;
};

C c1 = {1, 2};          // error: deduction failed
C c2 = {1, 2, 3};       // error: deduction failed
C c3 = {{1u, 2u}, 3};   // OK, C<int> deduced

D d1 = {1, 2};          // error: deduction failed
D d2 = {1, 2, 3};       // OK, braces elided, D<int> deduced

template <typename T>
struct I {
    using type = T;
};

template <typename T>
struct E {
    typename I<T>::type i;
    T t;
};

E e1 = {1, 2};          // OK, E<int> deduced
```

— *end example*]

References

[N4820] Richard Smith. Working Draft, Standard for Programming Language C++. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/n4820.pdf, 2019-06-17.

[P1021R4] Mike Spertus, Timur Doumler, and Richard Smith. Filling holes in Class Template Argument Deduction. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1021r4.html, 2019-06-17.