# 1

When declaring call parameters by reference, even trivial conversions do not apply to type deduction.

When declaring call parameters by value, only trivial conversions that decay are supported: Qualifications with const or volatile are ignored, references convert to the referenced type, and rawarrays or functions convert to the corresponding pointer type.

# 2

A class template usually applies multiple operations on the template arguments it is instantiated for (including construction and destruction).

when trying to declare the friend function and define it afterwards,

1. We can implicitly declare a new function template, which must use a different template parameter, such as U.
2. We can forward declare the output operator for a Stack<T> to be a template, which, however,means that we first have to forward declare Stack<T>.
3. You can define specific deduction guides to provide additional or fix existing class template argument deductions.

# 3

Note that nontype template parameters carry some restrictions. In general, they can be only constant integral values (including enumerations), pointers to objects/functions/members, lvalue references to objects or functions, or std::nullptr\_t (the type of nullptr)

# 4

Operator **sizeof...** It expands to the number of elements a parameter pack contains

Fold Expressions(C++ 17)

Variadic Expressions(C++ 17)

# 5

In general, has to be used whenever a name that depends on a template parameter is a type.

# 19.

C++ allows us to initialize a static constant data member inside its class only if it has an integral or enumeration type。

C++ allows us to initialize a static constant data member inside its class only if it has an integral or enumeration type

An alternative that works prior to C++17 is to use inline member functions for value traits that won’t

always yield integral values