ref class and ref struct (C++/CLI and C++/CX)

Article • 08/03/2021

The **ref class** or **ref struct** extensions declare a class or struct whose *object lifetime* is administered automatically. When the object is no longer accessible or goes out of scope, the memory is released.

All Runtimes

Syntax

```C++
class_access ref class name modifier : inherit_access base_type {};
class_access ref struct name modifier : inherit_access base_type {};
class_access value class name modifier : inherit_access base_type {};
class_access value struct name modifier : inherit_access base_type
{};
```

Parameters

class_access
(Optional) The accessibility of the class or struct outside the assembly. Possible values are `public` and `private` (`private` is the default). Nested classes or structs cannot have a *class_access* specifier.

name
The name of the class or struct.

modifier
(Optional) abstract and sealed are valid modifiers.

inherit_access
(Optional) The accessibility of *base_type*. The only permitted accessibility is `public` (`public` is the default).

base_type
(Optional) A base type. However, a value type cannot act as a base type.

For more information, see the language-specific descriptions of this parameter in the Windows Runtime and Common Language Runtime sections.

Remarks

The default member accessibility of an object declared with **ref class** or **value class** is `private`. And the default member accessibility of an object declared with **ref struct** or **value struct** is `public`.

When a reference type inherits from another reference type, virtual functions in the base class must explicitly be overridden (with override) or hidden (with new (new slot in vtable)). The derived class functions must also be explicitly marked as `virtual`.

To detect at compile time whether a type is a **ref class** or **ref struct**, or a **value class** or **value struct**, use `__is_ref_class (type)`, `__is_value_class (type)`, or `__is_simple_value_class (type)`. For more information, see Compiler Support for Type Traits.

For more information on classes and structs, see

- Instantiating Classes and Structs
- C++ Stack Semantics for Reference Types
- Classes, Structures, and Unions
- Destructors and finalizers in How to: Define and consume classes and structs (C++/CLI)
- User-Defined Operators (C++/CLI)
- User-Defined Conversions (C++/CLI)
- How to: Wrap Native Class for Use by C#
- Generic Classes (C++/CLI)

Windows Runtime

Remarks

See Ref classes and structs and Value classes and structs.

Parameters

base_type
(Optional) A base type. A **ref class** or **ref struct** can inherit from zero or more interfaces and zero or one **ref** types. A **value class** or **value struct** can only inherit from zero or more interfaces.

When you declare an object by using the **ref class** or **ref struct** keywords, the object is accessed by a handle to an object; that is, a reference-counter pointer to the object. When the declared variable goes out of scope, the compiler automatically deletes the underlying object. When the object is used as a parameter in a call or is stored in a variable, a handle to the object is actually passed or stored.

When you declare an object by using the **value class** or **value struct** keywords, the object lifetime of the declared object is not supervised. The object is like any other standard C++ class or struct.

Requirements

Compiler option: `/ZW`

Common Language Runtime

Remarks

The following table lists differences from the syntax shown in the **All Runtimes** section that are specific to C++/CLI.

Parameters

base_type
(Optional) A base type. A **ref class** or **ref struct** can inherit from zero or more managed interfaces and zero or one ref types. A **value class** or **value struct** can only inherit from zero or more managed interfaces.

The **ref class** and **ref struct** keywords tell the compiler that the class or structure is to be allocated on the heap. When the object is used as a parameter in a call or is stored in a variable, a reference to the object is actually passed or stored.

The **value class** and **value struct** keywords tells the compiler that the value of the allocated class or structure is passed to functions or stored in members.

Requirements

Compiler option: `/clr`

See also

Component Extensions for .NET and UWP
