Documentation

Storage

`consteval void make_static(info& data_mem)`

Requires: `is_data_member(data_mem)`

Updates the value of `data_mem` adding a static storage modifier. When injected via declaration cloning, this storage modifier will result in the injected declaration having static storage.

`consteval void clear_storage_modifier(info& data_mem)`

Requires: `is_data_member(data_mem)`

Updates the value of `data_mem` removing any storage modifier. When injected via declaration cloning, the original storage shall be used.

Access

`consteval void make_default(info& base_or_mem)`

Requires: `is_base_class(base_or_mem) or is_class_member(base_or_mem)`

Updates the value of `base_or_mem` adding a default access modifier. When injected via declaration cloning, this access modifier will result in the injected declaration having access equivalent to the default access for members in the given class kind, ie. for injection into a struct this shall be public, for injection into a class this shall be private.

`consteval void make_public(info& base_or_mem)`

Requires: `is_base_class(base_or_mem) or is_class_member(base_or_mem)`

Updates the value of `base_or_mem` adding a public access modifier. When injected via declaration cloning, this access modifier will result in the injected declaration having public access.

`consteval void make_protected(info& base_or_mem)`

Requires: `is_base_class(base_or_mem) or is_class_member(base_or_mem)`

Updates the value of `base_or_mem` adding a protected access modifier. When injected via declaration cloning, this access modifier will result in the injected declaration having protected access.

`consteval void make_private(info& base_or_mem)`

Requires: `is_base_class(base_or_mem) or is_class_member(base_or_mem)`

Updates the value of `base_or_mem` adding a private access modifier. When injected via declaration cloning, this access modifier will result in the injected declaration having private access.

`consteval void clear_access_modifier(info& base_or_mem)`

Requires: `is_base_class(base_or_mem) or is_class_member(base_or_mem)`

Updates the value of `base_or_mem` removing any access modifier. When injected via declaration cloning, the original access shall be used.

Constexpr

`consteval void make_constexpr(info& reflection,`

`bool mark_constexpr = true)`

Updates the value of `reflection` adding a constexpr modifier if `true`, removing any constexpr modifier if `false`. When injected via declaration cloning, a constexpr modifier will result in the injected declaration being constexpr.

Explicit

`consteval void make_explicit(info& ctor_or_conv,`

`bool mark_explicit = true)`

Requires: `is_constructor(ctor_or_conv) or is_conversion(ctor_or_conv)`

Updates the value of `ctor_or_conv` adding an explicit modifier if `true`, removing any explicit modifier if `false`. When injected via declaration cloning, an explicit modifier will result in the injected declaration being explicit.

Virtual

`consteval void make_virtual(info& mem_function,`

`bool mark_virtual = true)`

Requires: `is_member_function(mem_function)`

Updates the value of `mem_function` adding a virtual modifier if `true`, removing any virtual modifier if `false`. When injected via declaration cloning, a virtual modifier will result in the injected declaration being virtual.

```
consteval void make_pure_virtual(info& mem_function,
                                 bool mark_pure_virtual = true)
```

Requires: `is_member_function(mem_function)`

Updates the value of `mem_function` adding a pure virtual modifier if `true`, removing any pure virtual modifier if `false`. A pure virtual modifier effectively acts implicitly as a virtual modifier if present. When injected via declaration cloning, a pure virtual modifier will result in the injected declaration being pure virtual.

Names

```
consteval void set_new_name(info& named,
                            const char* new_name)
```

Requires: `is_named(mem_function)`

Updates the value of `named` adding a name modifier if non-null with the specifier `new_name`, removing any name modifier if null. When injected via declaration cloning, a name modifier will result in the injected declaration having the specifier name.