简易COM实现改进

注意事项

- 不同编译器虚函数
    1. 调用约定 —— `__thiscall`、`__stdcall`
    2. 虚表位置 —— 在对象首地址处
    3. 虚表数量 —— 避免使用多重继承、虚继承（菱形继承）
    4. 虚表项顺序 —— 避免使用重载
    5. 虚表项数量 —— 避免使用虚析构

用户接口设计

- 接口共同基类

```
// 接口共同基类
class IMyUnknow {
public:
    virtual HRESULT __stdcall query_interface(const GUID &type, void **object) =
0;
    virtual HRESULT __stdcall release() = 0;
};
```

- 工厂类接口

```
// 工厂类接口
class IMyClassFactory : public IMyUnknow {
public:
    virtual HRESULT __stdcall create_object(const GUID &interface_id, void
**object) = 0;
};


struct Factory {
    const GUID *class_id;
    void *(__stdcall *create)();       // 创建对象的函数指针
};
```

- 功能接口

```
// SuperMath接口类
class ISuperMath : public IMyUnknow {
public:
    virtual HRESULT __stdcall add(int a, int b, int *ret) = 0;
    virtual HRESULT __stdcall sub(int a, int b, int *ret) = 0;
};

// SuperString接口类
class ISuperString : public IMyUnknow {
public:
    virtual HRESULT __stdcall fast_find(char *buf, char *key, int *offset) = 0;
};
```

- 使用宏定义生成每个功能类的工厂类

```
// 每个类的工厂类
#define IMP_FACTORY(class_name) &CLSID_##class_name,
class_name##Factory::create,

#define IMP_DEFINE_FACTORY(class_name) \
class class_name##Factory : public IMyClassFactory { \
public: \
    static void * __stdcall create() \
    { \
        return new class_name##Factory(); \
    } \
    \
    virtual HRESULT __stdcall query_interface(const GUID &interface_id, void
**object) \
    { \
        if(memcmp(&interface_id, &IID_IMyUnKnow, sizeof(GUID)) == 0) { \
            *object = (IMyUnknow *)this; \
            return S_OK; \
        } else if(memcmp(&interface_id, &IID_IMyClassFactory, sizeof(GUID)) ==
0) { \
            *object = (IMyClassFactory*)this; \
            return S_OK; \
        } \
        return E_NOINTERFACE; \
    } \
    \
    virtual HRESULT __stdcall release() \
    { \
        delete this; \
        return S_OK; \
    } \
    \
    virtual HRESULT  __stdcall create_object(const GUID &interface_id, void
**object) \
    { \
        class_name *obj = new class_name(); \
        if(obj == NULL) { \
            return E_OUTOFMEMORY; \
        } \
        return obj->query_interface(interface_id, object); \
    } \
```

```
};
```

- 定义导出接口

```
// 导出接口
typedef HRESULT(__stdcall *my_get_class_object_ptr)(const GUID &class_id, const
GUID &interface_id, void **object);

extern "C" {
    DLL_API HRESULT __stdcall my_get_class_object(const GUID &class_id, const
GUID &interface_id, void **object);
}
```

- 定义各个对象的GUID

```
// 定义各个对象的GUID
// {97F913A8-3F80-4DD8-9DE1-D79FB9BD586D}
static const GUID IID_IMyUnKnow =
{ 0x97f913a8, 0x3f80, 0x4dd8, { 0x9d, 0xe1, 0xd7, 0x9f, 0xb9, 0xbd, 0x58, 0x6d }
};

// {5E7019B4-25D2-41B7-9626-37294DFBA872}
static const GUID IID_IMyClassFactory =
{ 0x5e7019b4, 0x25d2, 0x41b7, { 0x96, 0x26, 0x37, 0x29, 0x4d, 0xfb, 0xa8, 0x72 }
};

// {F0EBC905-65C6-461F-9FB6-7F39E00649D2}
static const GUID IID_ISuperMath =
{ 0xf0ebc905, 0x65c6, 0x461f, { 0x9f, 0xb6, 0x7f, 0x39, 0xe0, 0x6, 0x49, 0xd2 }
};

// {CFF31088-6D6B-446B-ADAE-48D76397EF70}
static const GUID IID_ISuperString =
{ 0xcff31088, 0x6d6b, 0x446b, { 0xad, 0xae, 0x48, 0xd7, 0x63, 0x97, 0xef, 0x70 }
};

// {FE17E7BB-4B8B-4951-9574-C34F09505DFD}
static const GUID CLSID_SuperMath =
{ 0xfe17e7bb, 0x4b8b, 0x4951, { 0x95, 0x74, 0xc3, 0x4f, 0x9, 0x50, 0x5d, 0xfd }
};

// {0011C118-9A22-4351-99B0-ECC058CA8229}
static const GUID CLSID_SuperString =
{ 0x11c118, 0x9a22, 0x4351, { 0x99, 0xb0, 0xec, 0xc0, 0x58, 0xca, 0x82, 0x29 }
};
```

功能设计

- 实现功能：继承各自的接口类，并实现接口

```
// 实现 SuperMath
class SuperMath : public ISuperMath {
public:
```

```
    virtual HRESULT __stdcall query_interface(const GUID &interface_id, void
**object)
    {
        if(memcmp(&interface_id, &IID_IMyUnKnow, sizeof(GUID)) == 0) {
            *object = (IMyUnknow *)this;
            return S_OK;
        } else if(memcmp(&interface_id, &IID_ISuperMath, sizeof(GUID)) == 0) {
            *object = (ISuperMath *)this;
            return S_OK;
        }
        return E_NOINTERFACE;
    }

    virtual HRESULT __stdcall release()
    {
        delete this;
        return S_OK;
    }

    virtual HRESULT __stdcall add(int a, int b, int *ret)
    {
        *ret = a + b;
        return S_OK;
    }

    virtual HRESULT __stdcall sub(int a, int b, int *ret)
    {
        *ret = a - b;
        return S_OK;
    }
};

// SuperMath的工厂类
IMP_DEFINE_FACTORY(SuperMath)


// 实现 SuperString
class SuperString : public ISuperString {
public:
    virtual HRESULT __stdcall query_interface(const GUID &interface_id, void
**object)
    {
        if (memcmp(&interface_id, &IID_IMyUnKnow, sizeof(GUID)) == 0) {
            *object = (IMyUnknow *)this;
            return S_OK;
        } else if (memcmp(&interface_id, &IID_ISuperString, sizeof(GUID)) == 0)
{
            *object = (ISuperString *)this;
            return S_OK;
        }
        return E_NOINTERFACE;
    }

    virtual HRESULT __stdcall release()
    {
        delete this;
        return S_OK;
    }
```

```
    virtual HRESULT __stdcall fast_find(char *buf, char *key, int *offset)
    {
        return S_OK;
    }
};

// SuperString的工厂类
IMP_DEFINE_FACTORY(SuperString)
```

- 将工厂类添加到数组中统一维护

```
// 工厂设计
Factory factory[] = {
    IMP_FACTORY(SuperMath)
    IMP_FACTORY(SuperString)
};
```

- 实现导出接口

```
// 实现导出接口
extern "C" {
    DLL_API HRESULT __stdcall my_get_class_object(const GUID &class_id, const
GUID &interface_id, void **object)
    {
        IMyClassFactory *obj = NULL;
        for(int i = 0; i < sizeof(factory) / sizeof(Factory); i++) {
            if(memcmp(&class_id, factory[i].class_id, sizeof(GUID)) == 0) {
                obj = (IMyClassFactory *)factory[i].create();
                return obj->query_interface(interface_id, object);
            }
        }
    return E_NOINTERFACE;
    }
```

- def文件导出，禁止名称粉碎

```
LIBRARY
EXPORTS
    my_get_class_object
```

用户使用

1. 加载DLL

```
// 加载dll
HMODULE dll = ::LoadLibraryA("./Tools.dll");
```

2. 获取导出函数

```
// 获取导出函数
my_get_class_object_ptr my_get_class_object =
(my_get_class_object_ptr)GetProcAddress(dll, "my_get_class_object");
```

3. 通过导出函数得到工厂类

```
// 通过导出函数拿到工厂类
HRESULT err = S_OK;
IMyClassFactory *factory = NULL;
err = my_get_class_object(CLSID_SuperMath, IID_IMyClassFactory, (void
**)&factory);
```

4. 通过工厂类获取功能类

```
// 通过工厂类拿到 ISuperMath 抽象接口
ISuperMath *super_math = NULL;
err = factory->create_object(IID_ISuperMath, (void **)&super_math);
```

5. 使用功能

```
int ret;
err = super_math->add(1, 2, &ret);
std::cout << "1 + 2 = " << ret << std::endl;
err = super_math->sub(1, 2, &ret);
std::cout << "1 - 2 = " << ret << std::endl;
```

6. 释放资源

```
// 释放接口
super_math->release();
super_math = NULL;
// 释放工厂类
factory->release();
```