На стыке управляемого и неуправляемого миров

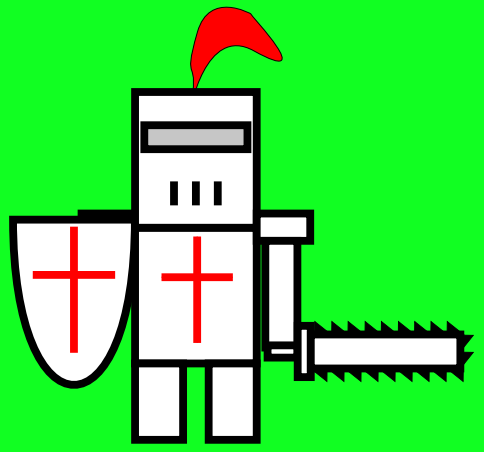
---

by: Иван Мигалёв 
https://github.com/ForNeVeR
https://fornever.me/

Технологии

- C++/CLI
- COM
- P/Invoke

C++/CLI



Что такое C++/CLI

```
System::String ^foo = gcnew System::String("foobaz");
```

```
int x = 10;
int %ref = x;
if (System::Int32::TryParse("42", ref))
    System::Console::WriteLine(x);
```

Нативные #include

```
#include <Windows.h>

void function_called_from_managed_code() {
  HANDLE mutex = CreateMutex(nullptr, false, nullptr);
}
```

#pragma unmanaged

```
#pragma managed
ref class ManagedClass {
    public:
        property int Foo {
            int get() { return 0; }
            void set(int value) {  }
        }
};

#pragma unmanaged
int __fastcall perform_wrapped_call() {
    int argument;
    __asm { mov argument, eax }
    return argument;
}
```

.NET Core

https://github.com/dotnet/coreclr/issues/659

> ***2015-04-10**: There is no plan to support C++/CLI with .NET Core.*
>
> ***2018-09-15**: You can track progress on Windows-only Managed C++ support in #18013.*

Байки из склепа

Delphi

```
type
 TMyCallback = function(number: Integer): Integer;

function DoCall(callback: TMyCallback): Integer; cdecl;
begin
  Result := callback(10);
end;
```

Callbacks

```
Func<int, int> ^callback;

[UnmanagedFunctionPointer(CallingConvention::FastCall)]
delegate int ExportFunction(int);
```

Delegate construction

```
auto exportFunction = gcnew ExportFunction(
    callback,
    &Func<int, int>::Invoke);
```

GetFunctionPointerForDelegate

```
auto ptr = Marshal::GetFunctionPointerForDelegate(
    safe_cast<Delegate^>(exportFunction));
```

Unmanaged

```
Callback *CallbackInstance;

int __fastcall perform_wrapped_call() {
  int real_argument;
  __asm { mov real_argument, eax }
  return CallbackInstance(real_argument);
}
```

Component Object Model

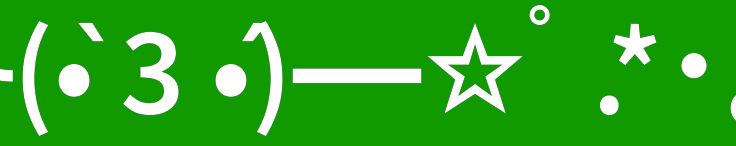
COM 101

```
IComService instance = new IComService();
instance.HelloWorld();
```

dynamic & #if

```
#if COM_LIBRARY_INSTALLED
  IComService instance = new IComService();
#else
  const string TypeGuid =
    "03653ea3-b63b-447b-9d26-fa86e679087b";
  Type type = Type.GetTypeFromCLSID(Guid.Parse(TypeGuid));
  dynamic instance = Activator.CreateInstance(type);
#endif

  instance.HelloWorld();
```

P/Invoke



DllImportAttribute

```
[DllImport("StringConsumer.dll", CharSet = CharSet.Unicode)]
private static extern void PassUnicodeString(string str);
```

DllImportAttribute: подробности

```
public sealed class DllImportAttribute : Attribute
{
    public DllImportAttribute(string dllName) { /* ... */ }
    public string EntryPoint;
    public CharSet CharSet;
    public bool SetLastError;
    public bool ExactSpelling;
    public CallingConvention CallingConvention;
    public bool BestFitMapping;
    public bool PreserveSig;
    public bool ThrowOnUnmappableChar;
}
```

DllImportAttribute: dllName

```
[DllImport("tdjson.dll")] // → tdjson.dll       // Windows
[DllImport("tdjson")]     // → libtdjson.so     // Linux, .NET
[DllImport("libtdjson.so")]  // → libtdjson.so // Linux, Mono
[DllImport("libtdjson.dylib")] // → libtdjson.dylib // macOS

[DllImport("__Internal")] // Mono only
```

DllImportAttribute.EntryPoint, ExactSpelling

```
[DllImport("somelib.dll", EntryPoint = "MyFunctionName")]
[DllImport("somelib.dll", EntryPoint = "#123")] // by ordinal
[DllImport("somelib.dll",
  EntryPoint = "MessageBoxA",
  ExactSpelling = true)]
```

DllImportAttribute: кодировки

```
public CharSet CharSet; // Auto, Ansi, Unicode
public bool BestFitMapping;
public bool ThrowOnUnmappableChar;
```

DllImportAttribute.SetLastError

```
int errorCode = Marshal.GetLastWin32Error();
string errorMessage = new Win32Exception(errorCode).Message;
```

DllImportAttribute.CallingConvention

```
public enum CallingConvention
{
  Winapi = 1,
  Cdecl = 2,
  StdCall = 3,
  ThisCall = 4,
  FastCall = 5,
}
```

DllImportAttribute.PreserveSig

```
[DllImport("my.dll", PreserveSig = true)]
HRESULT GetSomething(/*[out, retval]*/ BSTR *pRetVal);

[DllImport("my.dll", PreserveSig = false)]
extern static string GetSomething();
```

Передача аргументов

Values:

- примитивные типы (`int`, `long`, `double` etc.)
- другие value types (структуры, enums)

Pointers:

- ссылочные типы (классы, делегаты)
- `ref` / `out`
- `IntPtr`
- unsafe-указатели

Blittable / nonblittable

Указатели в C#

```
int[] x = new int[10];
fixed (int* ptr = x) {
    Native.Call(ptr);
}
```

StructLayout: unions

```
[StructLayout(LayoutKind.Sequential)]
public class DBVariant {
  public byte type;
  public Variant Value;

  [StructLayout(LayoutKind.Explicit)]
  public struct Variant {
    [FieldOffset(0)] public byte bVal;
    [FieldOffset(0)] public byte cVal;
    [FieldOffset(0)] public ushort wVal;
    [FieldOffset(0)] public IntPtr pszVal;
    [FieldOffset(0)] public char cchVal;
  }
}
```

StructLayout: Pack

```
[StructLayout(LayoutKind.Sequential, Pack = 8)] // 16 bytes
public class DBVariant1 {
  public byte type;
  // padding: 7 bytes
  public IntPtr Pointer; }

[StructLayout(LayoutKind.Sequential, Pack = 1)] // 9 bytes
public class DBVariant2 {
  public byte type;
  // no padding
  public IntPtr Pointer; }
```

fixed arrays

```
// C
struct X {
  int Array[30];
};
```

```
unsafe struct X {
  fixed int Array[30];
}
```

Тесты на memory layout

```
struct Foo { public int x, y; }
Foo f = new Foo();
int offset1 = (byte*) &f.x - (byte*) &f;
Assert.Equal(0, offset1);
```

"Строки"

MarshalAsAttribute

```
extern void Foo([MarshalAs(UnmanagedType.BStr)] string arg);
extern void Foo([MarshalAs(UnmanagedType.LPStr)] string arg);
extern void Foo([MarshalAs(UnmanagedType.LPWStr)] string arg);
```

MutateString (native)

```
#include <cwchar>
#include <xutility>

extern "C" __declspec(dllexport) void MutateString(
    wchar_t *string) {
    std::reverse(string, std::wcschr(string, L'\0'));
}
```

MutateString (managed)

```
[DllImport("Project1.dll", CharSet = CharSet.Unicode)]
private static extern void MutateString(string foo);

static void Main() {
  var myString = "Hello World 1";
  MutateString(myString);

  Console.WriteLine(myString);
  Console.WriteLine("Hello World 1");
}
```

MutateString (managed) 

```
[DllImport("Project1.dll", CharSet = CharSet.Unicode)]
private static extern void MutateString(string foo);

static void Main() {
  var myString = "Hello World 1";
  MutateString(myString);

  Console.WriteLine(myString);           // => 1 dlroW olleH
  Console.WriteLine("Hello World 1");    // => 1 dlroW olleH
}
```

MutateString (StringBuilder)

```
[DllImport("Project1.dll", CharSet = CharSet.Unicode)]
private static extern void MutateString(StringBuilder foo);

static void Main() {
  var myString = new StringBuilder("Hello World 1");
  MutateString(myString);

  Console.WriteLine(myString.ToString()); // => 1 dlroW olleH
  Console.WriteLine("Hello World 1");     // => Hello World 1
}
```

MutateStruct (native)

```
struct S {
  wchar_t *field;
};
extern "C" __declspec(dllexport) void MutateStruct(S *s) {
  MutateString(s->field);
}
```

MutateStruct (managed)

```
[StructLayout(LayoutKind.Sequential,
    CharSet = CharSet.Unicode)]
struct S {
    public string field;
}

[DllImport("Project1.dll", CharSet = CharSet.Unicode)]
private static extern void MutateStruct(ref S foo);

S s = new S();
s.field = "Hello World 2";
MutateStruct(ref s);

Console.WriteLine(s.field);          // => 2 dlroW olleH
Console.WriteLine("Hello World 2"); // => Hello World 2
```

Сравнение быстродействия

```
void PassAnsiString(char *) {}
void PassUnicodeString(wchar_t *) {}
```

Сравнение быстродействия: бенчмарк

```
[Params(10, 100, 1000)]
public int N;

private string stringToPass;

[GlobalSetup]
public void Setup() => stringToPass = new string('x', N);

[Benchmark]
public void PassAnsiString() => PassAnsiString(stringToPass);

[Benchmark]
public void PassUnicodeString() =>
    PassUnicodeString(stringToPass);
```

Сравнение быстродействия: результаты

Method	N	Mean	Error
PassAnsiString	10	89.89 ns	1.5052 ns
PassUnicodeString	10	34.68 ns	0.4818 ns
PassAnsiString	100	167.77 ns	3.4897 ns
PassUnicodeString	100	36.37 ns	0.7480 ns
PassAnsiString	1000	1,032.29 ns	7.2073 ns
PassUnicodeString	1000	36.05 ns	0.7446 ns

Разное

SafeHandle

```
// extern IntPtr CreateFile(…);
// extern void CloseHandle(IntPtr _);
IntPtr someHandle = CreateFile(…);
if (someHandle == IntPtr.Zero)
    throw new Exception("Invalid handle value");
try { // …
} finally {
  CloseHandle(someHandle);
}
```

```
class MyHandle : SafeHandleZeroOrMinusOneIsInvalid
{
  public MyHandle() : base(true) { }
  protected override bool ReleaseHandle() =>
    CloseHandle(this.handle);
}
```

ICustomMarshaler

```
public interface ICustomMarshaler {
  object MarshalNativeToManaged(IntPtr pNativeData);
  IntPtr MarshalManagedToNative(object ManagedObj);
  void CleanUpNativeData(IntPtr pNativeData);
  void CleanUpManagedData(object ManagedObj);
  int GetNativeDataSize();
}
```

```
public class MyMarshaler : ICustomMarshaler {
  public static ICustomMarshaler GetInstance(string cookie) =>
    new MyMarshaler();
  // …
}

[MarshalAs(UnmanagedType.CustomMarshaler,
           MarshalType = "Foo.Bar.MyMarshaler",
           MarshalCookie = "Test")]
```

UnmanagedFunctionPointer

```
[UnmanagedFunctionPointer(CallingConvention.FastCall)]
delegate void MarshalableDelegate(int param);

[DllImport(…)]
static extern void NativeFunc(MarshalableDelegate x);

var myDelegate = new MarshalableDelegate(myObject.MyMethod);
NativeFunc(myDelegate);
```

GC.KeepAlive

```
var myDelegate = new MarshalableDelegate(myObject.MyMethod);
var context = NativeFuncBegin(myDelegate);
// ...
var result = NativeFuncEnd(context);
GC.KeepAlive(myDelegate);
```

Трюки с CIL: .export

```
.assembly extern mscorlib { auto }
.assembly extern System { auto }
.assembly SpySharp.Hooks {}
.module SpySharp.Hooks.dll

.method assembly static native int
  modopt (
    [mscorlib]System.Runtime.CompilerServices.CallConvStdcall)
  HookProc(
  int32 nCode,
  native int lParam,
  native int wParam) {
  .vtentry 1 : 1
  .export [1] as HookProc

  ldc.i4.0
  ret
}
```

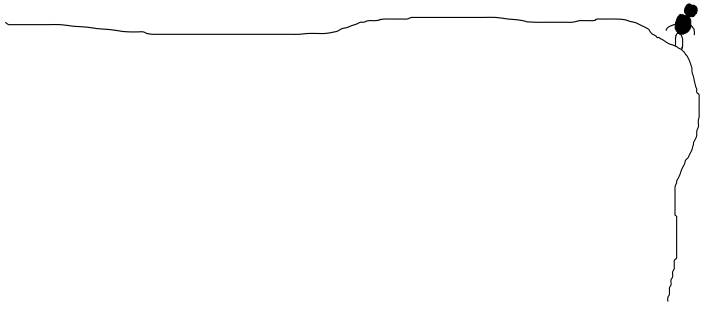
Трюки с CIL: vararg

```
.method public static pinvokeimpl("msvcrt.dll" ansi cdecl)
vararg int32 printf(string) cil managed preservesig {}

// in method:
.locals init(int32 i, void* pb)

// printf("%2.2d : %d\n", i, *(int*)pb);
ldstr "%2.2d : %d\n"
ldloc.0
ldloc.1
ldind.i4
call vararg int32 printf(string, ..., int32, int32)
```

Заключение

1. Не нужно бояться нативного кода.
2. По возможности стоит описывать код в безопасном стиле.
3. `StructLayout` — наш друг.
4. Со строками следует обращаться крайне осторожно.
5. Сохраняйте ссылки на делегаты.
6. Пишите тесты.
7. Можно даже писать тесты на memory layout.

Это конец.



Материалы доклада доступны в репозитории:
https://github.com/ForNeVeR/talk-marshal-unsafe

Контакты автора:
friedrich@fornever.me
https://t.me/fvnever