www.italiancpp.org

Mobile C++ & C#

F. Milicchio, Roma Tre, ++it

Sviluppo Nativo

- Sviluppare più applicazioni native è tedioso
- La UI, in special modo, deve essere aderente agli standard per piattaforma
- Usare SDK multipiattaforma aiuta lo sviluppo
- Le prestazioni potrebbero soffrirne
- Esistono SDK che permettono lo sviluppo nativo e multipiattaforma
- Xamarin è una di queste

Xamarin

- Xamarin è una azienda fondata da Miguel de Icaza
- È un progetto basato su *mono*, dunque C#
- Permette lo sviluppo nativo multipiattaforma (iOS, Android, WP)
- Lo sviluppo consente una suddivisione razionale
- Il codice platform-specific viene ridotto al 40%
- Per la UI è possibile usare Xamarin.Forms
- In questo modo il codice condiviso sale fino al 99%

- Xamarin supporta *librerie native* C e C++

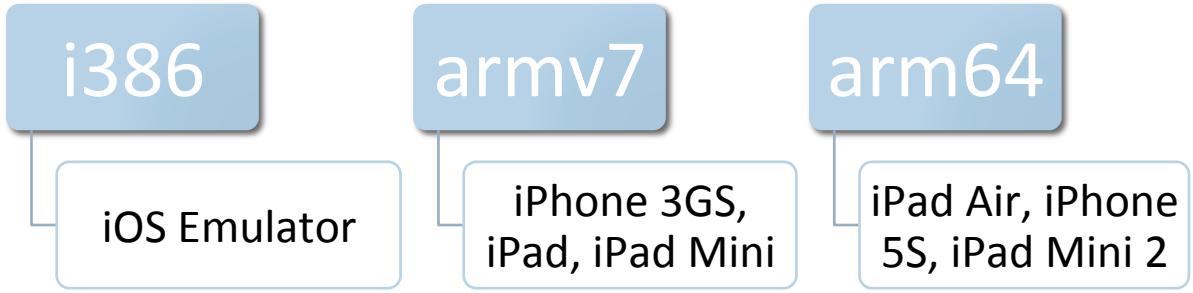
Agenda

Libreria C++

```
int doStuff()
{
    std::string s("abcd");

    return s.length();
}
```

```
extern "C" int doStuff();
```

Binario Multipiattaforma




```
% lipo -create Debug-iphoneos/libcpptest.a Debug-iphonesimulator/
libcpptest.a -o fatbinary.
```

```
% nm fatbinary.a
fatbinary.a(cpptest.o) (for architecture armv7):
000000d8 S __ZNSt3__111char_traitsIcE6lengthEPKc
  ...
00000000 T _doStuff

fatbinary.a(cpptest.o) (for architecture i386):
0000a0b0 S __ZNSt3__111char_traitsIcE6lengthEPKc
   ...
fatbinary.a(cpptest.o) (for architecture arm64):
000000000000a1f4 s L_.str
000000000000a1d0 S __ZNSt3__111char_traitsIcE6lengthEPKc
```

Xamarin C#

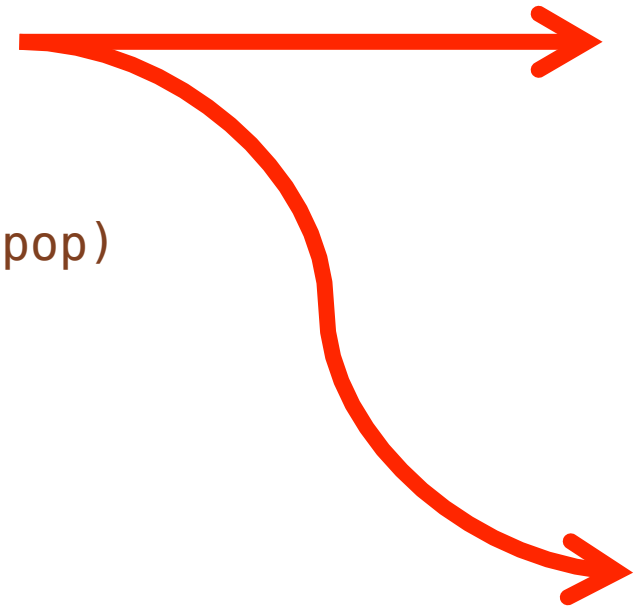
```
namespace xamarincpp.iOS
{
	public class Application
	{
		// This is the main entry point of the application.
		static void Main (string[] args)
		{

			System.Console.WriteLine (LIBRARY RESULT: " + doStuff ());
			UIApplication.Main (args, null, "AppDelegate");
		}

		[DllImport("__Internal")]
		public extern static int doStuff();
	}


}
```

Data Sharing

- Il consiglio è di tenere separate le parti C++ da quelle C#
- Il codice C# è managed, quello C++ unmanaged
- La via più sicura è tramite marshalling
- È anche possibile specificare per tipi in standard layout un corrispettivo C#
- Per STL, è sconsigliabile modificare le strutture dati manualmente
- Con strutture dati con standard layout, è necessario stabilirne l'alignment

Standard Layout

```
#pragma pack(push, 1)
class stuff
{
public:
    char c;
    int  i;
};
#pragma pack(pop)
```



```
[StructLayout(LayoutKind.Sequential,
 CharSet=CharSet.Ansi,
 Pack=1)]
public struct cppclass
{
        public byte nativec;
        public int  nativei;
}
```

```
[StructLayout(LayoutKind.Explicit)]
public class
{
        [FieldOffset(0)] public byte nativec;
        [FieldOffset(8)] public int  nativei;
}
```

Standard Layout

```
void usePtr(stuff *p)
{
    p->c = 'X';
    p->i = 23;
}

stuff returnStuff()
{
    stuff t;

    t.c = 'A';
    t.i = 42;

    return t;
}
```

```
extern "C" void usePtr(stuff *p);

extern "C" stuff returnStuff();
```

Pointers in C#

```
void usePtr(stuff *p)
{
    p->c = 'X';
    p->i = 23;
}

stuff returnStuff()
{
    stuff t;

    t.c = 'A';
    t.i = 42;

    return t;
}
```

```
cppclass p;
p.nativec = 0;
p.nativei = 0;
native.nativeUsePtr (ref p);

var r = native.nativeReturnStuff ();
```