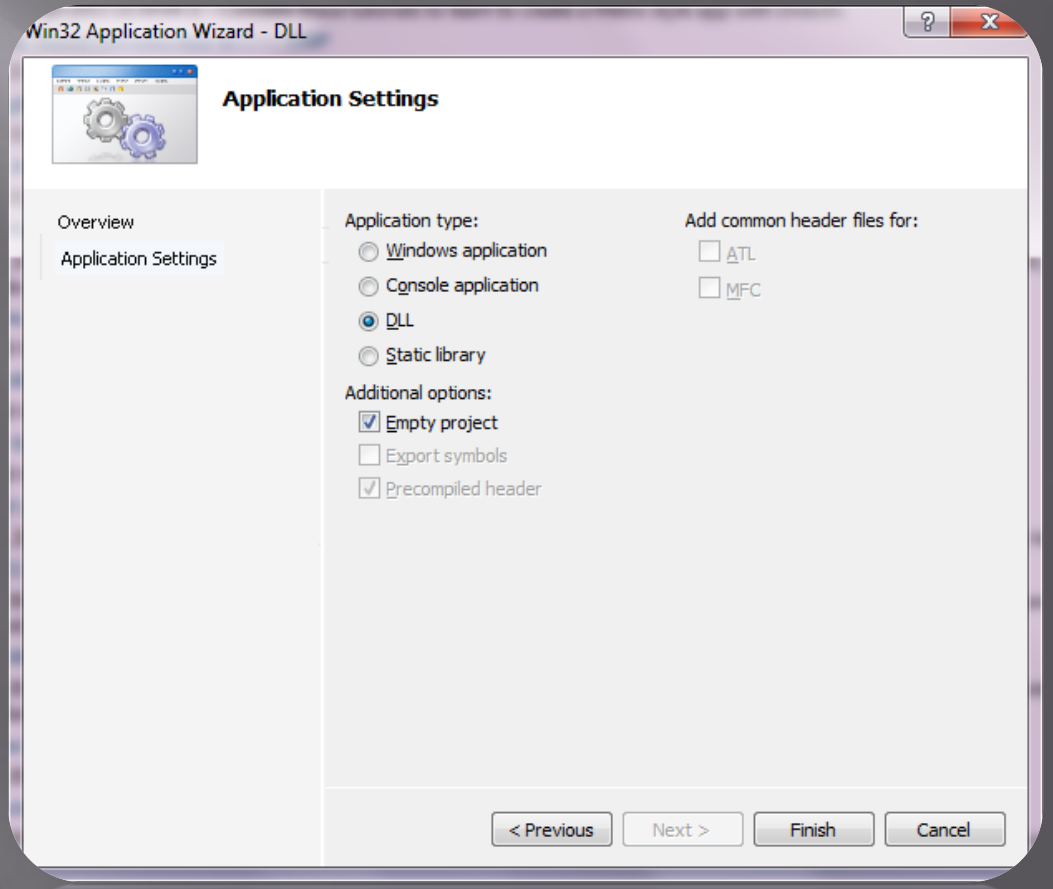
KNJIŽNICE DLL

Pregled in pomoč pri uporabi

Denis Obrul, UM FERI 2012

DLL (C/C++)

- Ustvarite projekt tipa Win32->Win32 Project:
  - DLL
  - Empty project



DLL koda (.h)

- Dostopali boste lahko samo do funkcij, ki jih neposredno izvozite (export) iz DLL.
- Sintaksa je vedno enaka:

```
extern "C" __declspec (dllexport) const char* darwin(const char* vhod);
```

- Ponovite za vsako funkcijo, ki jo želite izvoziti.

DLL koda (.cpp)

- Največ težav povzroča prenos nizov.
- Vedno uporabite **const char*;** tako za vhod v funkcijo, kot izhod (return).
- Primer:

```
const char* darwin(const char* vhod)
{
	// ...
}
```

Delo z nizi (C++)

- Za lažje delo pretvorite vhodni niz char* v spremenljivko tipa std::string

```
const char* darwin(const char* vhod)
{
	string delovniNiz(vhod);
	// ...
}
```

- Razred string ima nekaj uporabnih funkcij (find, copy, compare, replace itd.). Preverite dokumentacijo!

Delo z nizi (C++)

- Ko ste končali z delom, je potrebno vrniti rezultat. Če ste uporabljali string je potrebna pretvorba nazaj v char*.
- Spremenljivke tipa string kot take ni možno vrniti, saj se ob izhodu iz funkcije uniči. Vsebino je potrebno skopirati neposredno v pomnilnik (uporabimo ukaz new).

Delo z nizi (C++)

- Ko ste končali z delom, je potrebno vrniti rezultat. Če ste uporabljali string:

```
const char* darwin(const char* vhod)
{
	string delovniNiz(vhod);
	// ...
	char* izhod = new char[delovniNiz.length() +1];
	izhod[delovniNiz.length()] = '\0';

	for (int i = 0; i < delovniNiz.length(); i++)
	{
		izhod[i] = delovniNiz[i];
	}

	return izhod;
}
```

Delo z nizi (C++)

- Ko ste končali z delom, je potrebno vrniti rezultat. Če ste uporabljali string*, lahko preprosto uporabite funkcijo c_str():

```
const char* darwin(const char* vhod)
{
	string* delovniNiz = new string(vhod);
	// ...
	return delovniNiz->c_str();
}
```

DLL in C#

- Knjižnico je potrebno uvoziti (import). To lahko storite takoj na začetku razreda:

```
using System.Runtime.InteropServices;

public partial class GUI : Form
{
    [DllImport("DLLTest.dll", CallingConvention = CallingConvention.Cdecl)]
    // ...
}
```

DLL in C#

- Za uvozom naštejte vse funkcije, ki jih želite uporabljati:

```
public partial class GUI : Form
{
      [DllImport("DLLTest.dll", CallingConvention = CallingConvention.Cdecl)]

       public static extern IntPtr darwin([MarshalAs(UnmanagedType.LPStr)] string vhod);
      // ...
}
```

- MarshalAs(...) skrbi za pravilno pretvorbo podatkovnih tipov.

DLL in C#

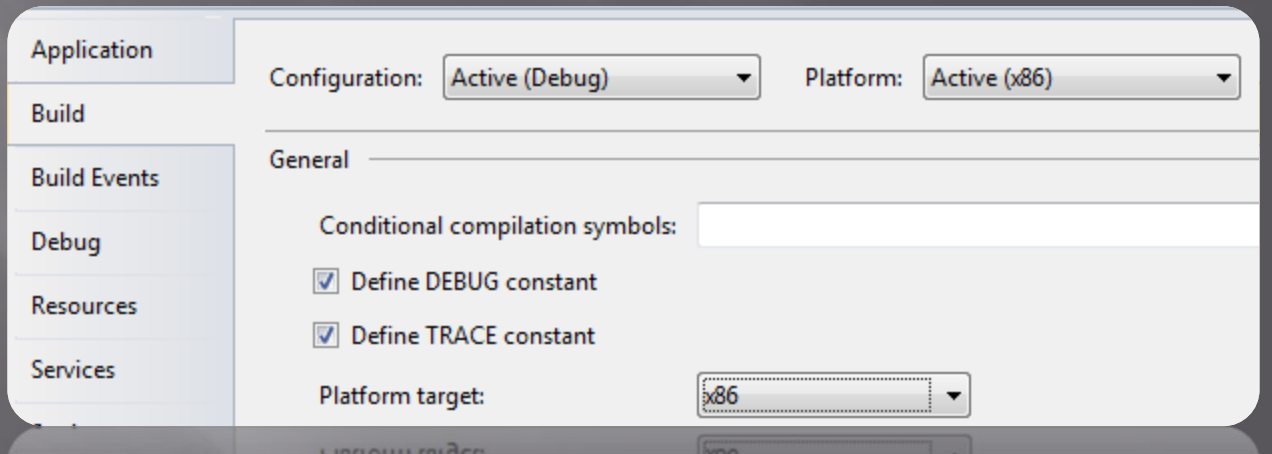
- Klic funkcij je preprost, paziti moramo le, da pretvorimo tudi rezultat (za vhodne podatke poskrbi MarshalAs):

```
public partial class GUI : Form
{
	[DllImport("DLLTest.dll", CallingConvention = CallingConvention.Cdecl)]

	 public static extern IntPtr darwin([MarshalAs(UnmanagedType.LPStr)] string vhod);
	// ...
	public void doSomething()
	{
		string rezultat = Marshal.PtrToStringAnsi(darwin("charles"));
	}
}
```

DLL in C#

- Omenjen postopek je bil uspešno testiran v Visual Studio 2008 in 2010.
- Do napake pride, če je DLL namenjen 32 bitni arhitekturi, C# pa 64 bitni.
- V tem primeru nastavite lastnosti C# projekta (Project -> Properties -> Build -> Platform target = x86).



Za konec

- Z knjižnico DLL načeloma niste omejeni na jezik, kjer jo lahko uporabite.
- Praviloma se grafični vmesnik razvija posebej (Java, C#, Delphi), kritični deli aplikacije pa kot knjižnica DLL.
- Poleg mučenja študentov imajo knjižnice DLL v praksi pomembno vlogo in praktično ni aplikacije, ki jih tako ali drugače ne bi uporabljala. Začetne težave se vam bodo na dolgi rok dobro obrestovale!