Mixing C++ and D

Dragos Carp

28.01.2016, Munich

C++ Interoperability

- Name mangling
- Templates
- SFINAE
- Namespaces
- Overloading
- Argument Dependent Lookup
- `RTTI`
- `Virtual functions`
- `Exceptions`
- `Special member functions`
- `Operator overloading`
- `Const`

Don't *compile* C++

Link to C++

Data Type Compatibility

D Type	C Type
void	void
byte	signed char
ubyte	unsigned char
char	char (chars are unsigned in D)
wchar	wchar_t (when sizeof(wchar_t) is
dchar	wchar_t (when sizeof(wchar_t) is
short	short
ushort	unsigned short
int	int
uint	unsigned
long	long long
ulong	unsigned long long
float	float
double	double
real	long double

D Type	C Type
struct	struct, class
union	union
enum	enum
class	no equivalent
*type**	*type **
no equivalent	*type &*
type[dim]	*type[dim]*
*type[dim]**	*type(*)[dim]*
type[]	no equivalent
type[type]	no equivalent
type function(parameters)	*type(*)(parameters)*
type delegate(parameters)	no equivalent

Source: https://dlang.org/spec/cpp_interface.html

C++ namespace vs. D Name Spaces

```
// foo.cpp;
namespace N {
  namespace M {
    void foo();
  }
}

// bar.cpp;
namespace N {
  void bar();
}
```

- module
- struct
- class
- mixin template

Extend C++ Declaration

```
extern (C++, N.M) {
  void foo();
}
extern (C++, N) {
  void bar();
}
```

Thanks!

- Need to be flexible on both ends
- STL support is on the way, but not portable
- No 100% solution
- Better than C wrappers
- Makes escape from C++ codebase possible

Example: https://gitlab.com/dcarp/MUCplusplus