NPRG051
Pokročilé programování v C++

David Bednárek
Jakub Yaghob
Filip Zavoral

https://www.ksi.mff.cuni.cz/teaching/nprg051-web/NPRG051-F1-interop.pptx

Interoperabilita

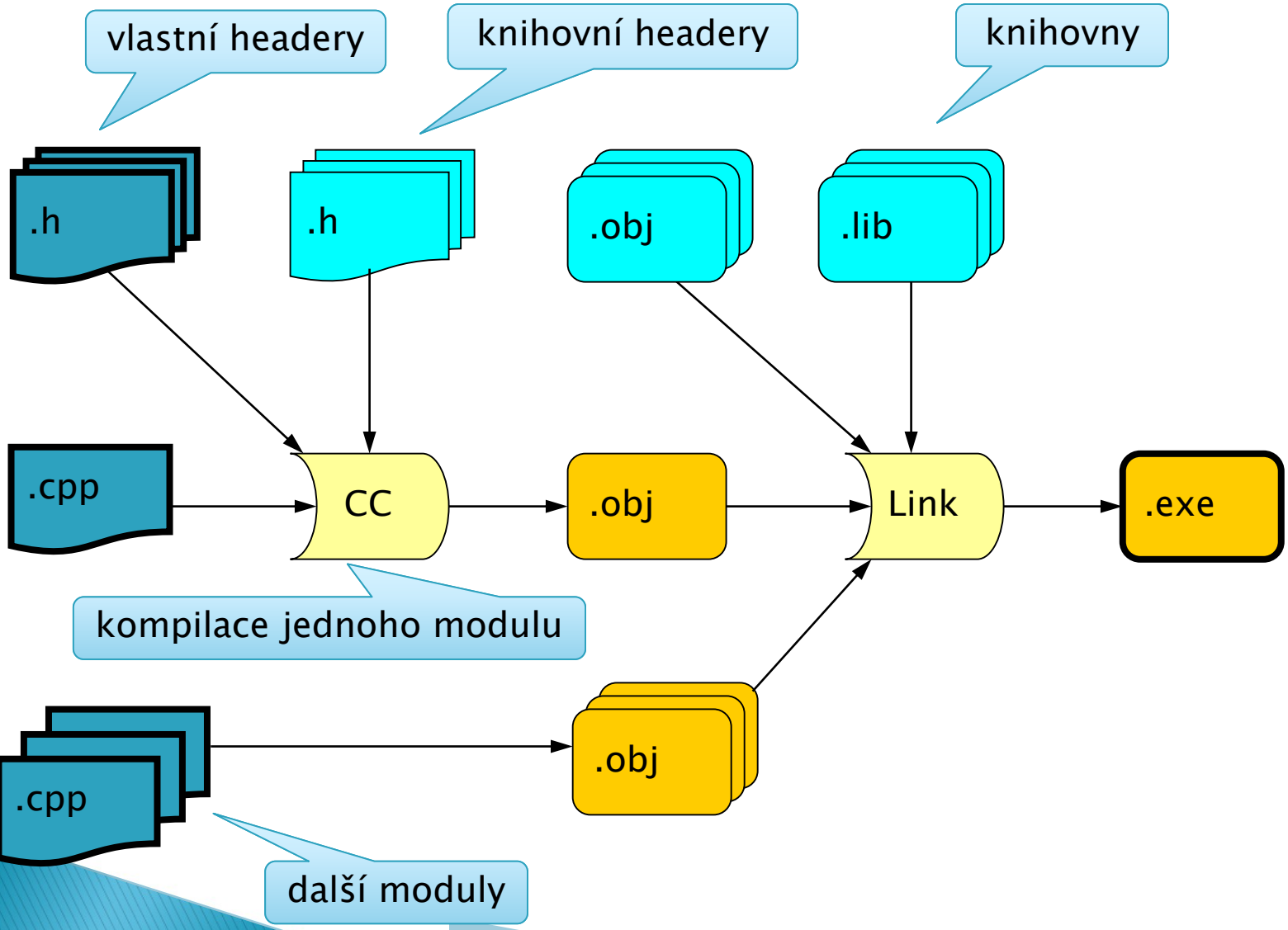
Interoperabilita

- C++ a vlastní C moduly
  - obj, lib, dll/so
  - jak linkovat C a C++ moduly
  - jak dělat společné C/C++ headery

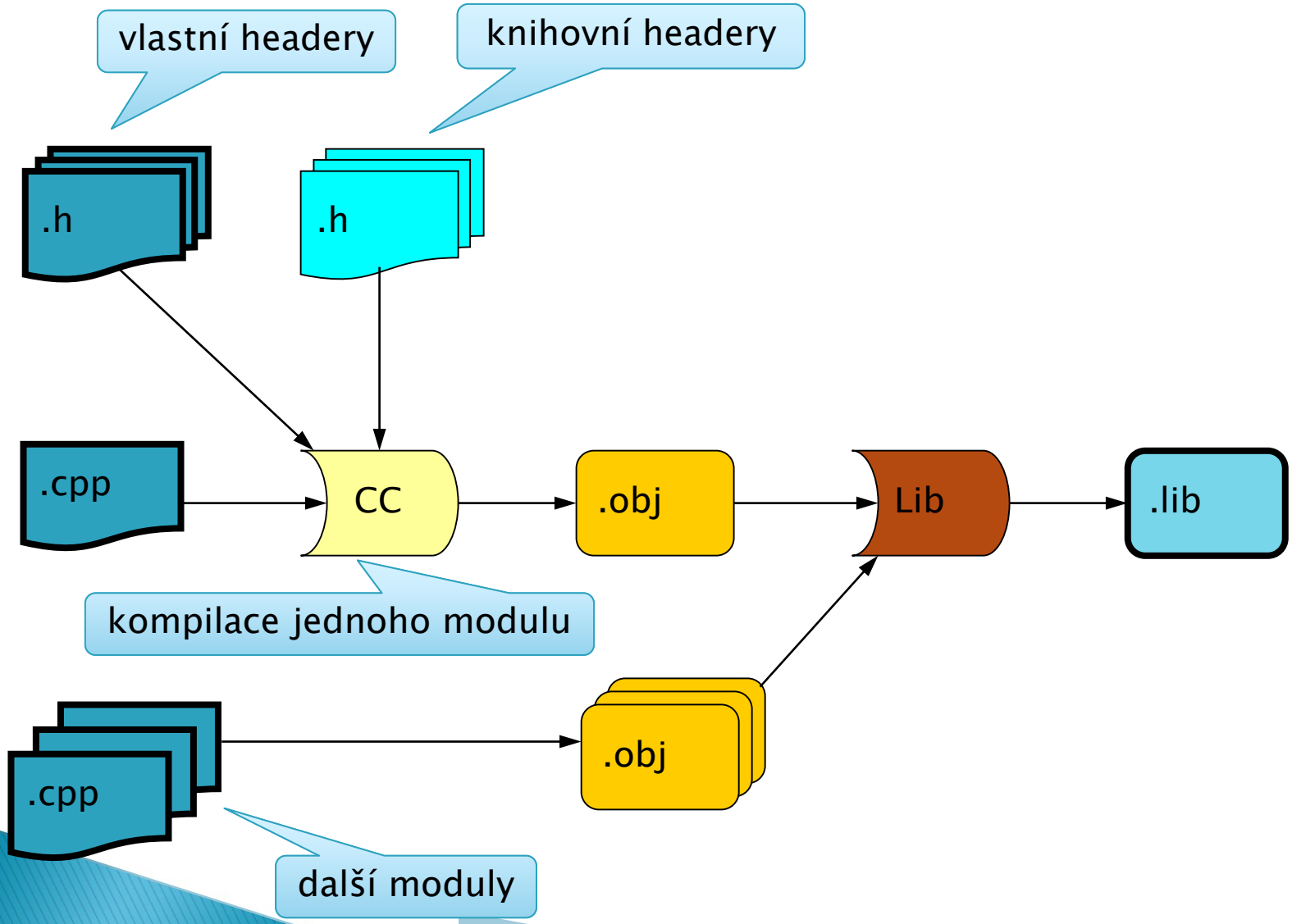
- C++ a cizí C knihovny
  - jak z C++ volat C knihovny
  - callback z C knihoven do C++
  - mandlování, volací konvence
  - dynamicky linkované knihovny

- C++ a .Net/C#/cokoliv#
  - jak spojovat moduly
  - jak volat metody
  - jak sdílet data

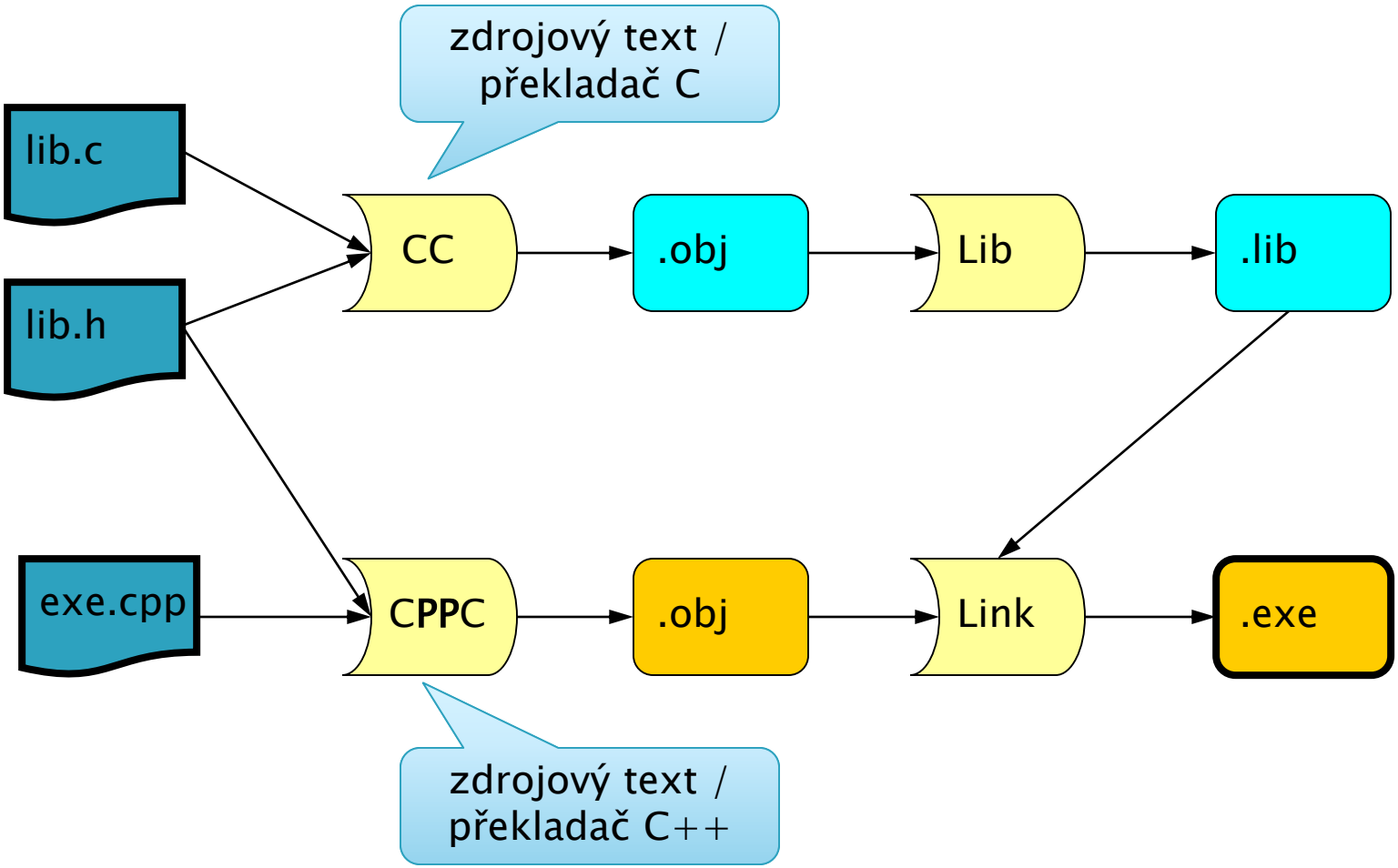
Překlad více modulů



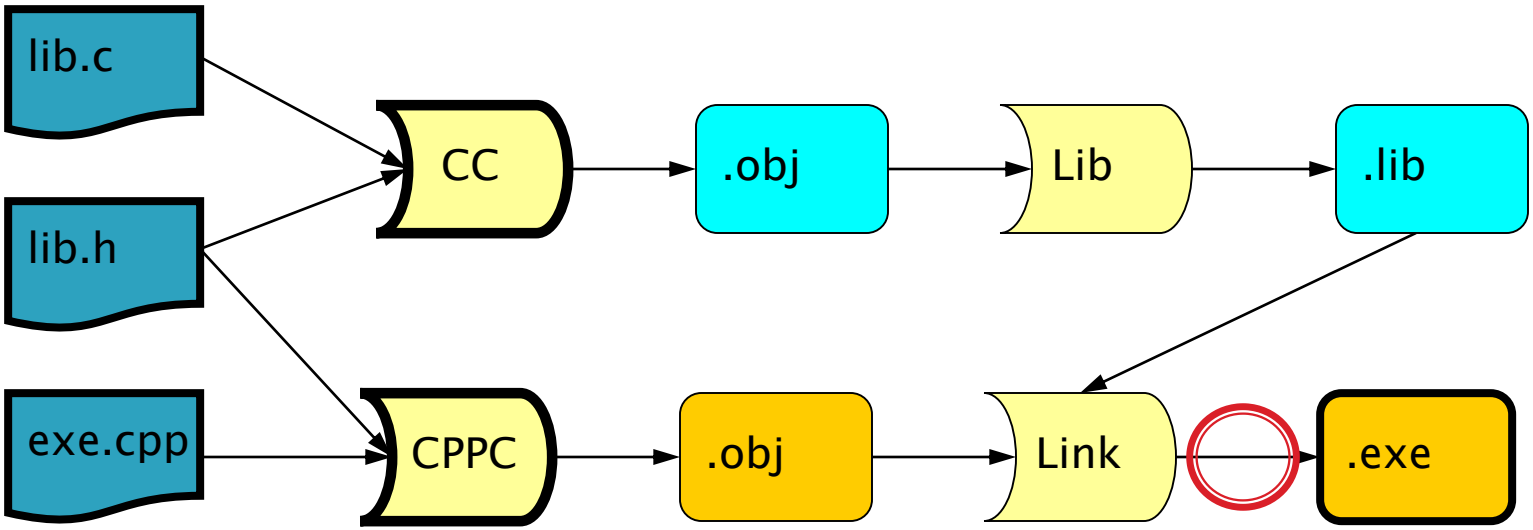
Vytvoření vlastní knihovny



C++ exe / C lib


zdrojový text / překladač C
lib.c
lib.h
CC
.obj
Lib
.lib
exe.cpp
CPPC
.obj
Link
.exe
zdrojový text / překladač C++

C++ exe / C lib



- error LNK2019: unresolved external symbol
  - "int __cdecl lib_fnc(int)" (?lib_fnc@@YAHH@Z)
  - referenced in function _main

what the ... ... hell???

Mandlování

- mangling
  - mandlování, znetvoření
  - *name-decoration*

```
int a;
int a( void);
int a( int, int);
class a {};
class a { int a; };
class a { int a( int); };
```

- syntaktická a sémantická informace o symbolu
  - overloading / přetěžování ➠ nejednoznačnost
- zjednoznačnění identifikátoru
  - proměnná / funkce / operator / metoda
  - typy a typové konstrukce parametrů a návratové hodnoty
  - třída, další atributy (const, volatile, ...)
  - volací konvence
- formát jednotně nedefinovaný
  - závislý na platformě, překladači, ...
  - obecně nepřenositelné

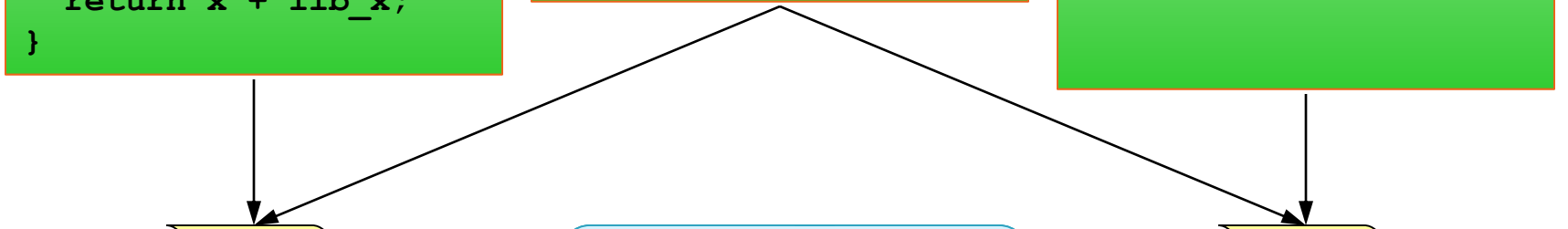
Compiler	void h(int)	void h(int, char)
Intel C++ 8.0 for Linux	_Z1hi	_Z1hic
HP aC++ A.05.55 IA-64	_Z1hi	_Z1hic
IAR EWARM C++ 5.4 ARM	_Z1hi	_Z1hic
GCC 3.*x* and 4.*x*	_Z1hi	_Z1hic
GCC 2.9*x*	h__Fi	h__Fic
HP aC++ A.03.45 PA-RISC	h__Fi	h__Fic
Microsoft Visual C++ v6-v10 (mangling details)	?h@@YAXH@Z	?h@@YAXHD@Z
Digital Mars C++	?h@@YAXH@Z	?h@@YAXHD@Z
Borland C++ v3.1	@h$qi	@h$qizc
OpenVMS C++ V6.5 (ARM mode)	H__XI	H__XIC
OpenVMS C++ V6.5 (ANSI mode)	CXX$__7H__FI0ARG51T	CXX$__7H__FIC26CDH77
OpenVMS C++ X7.1 IA-64	CXX$_Z1HI2DSQ26A	CXX$_Z1HIC2NP3LI4
SunPro CC	__1cBh6Fi_v_	__1cBh6Fic_v_
Tru64 C++ V6.5 (ARM mode)	h__Xi	h__Xic
Tru64 C++ V6.5 (ANSI mode)	__7h__Fi	__7h__Fic
Watcom C++ 10.6	W?h$n(i)v	W?h$n(ia)v

C++ exe / C lib

```
/* pureclib.c */
#include "pureclib.h"

int lib_x;

int lib_fnc( int x)
{
  return x + lib_x;
}
```

```
/* pureclib.h */

#ifndef PURECLIB__H_
#define PURECLIB__H_

extern int lib_x;
int lib_fnc( int x);

#endif
```

```
// cppexe.cpp
#include "pureclib.h"

int main(....)
{
  int i = lib_fnc( 1);
}
```

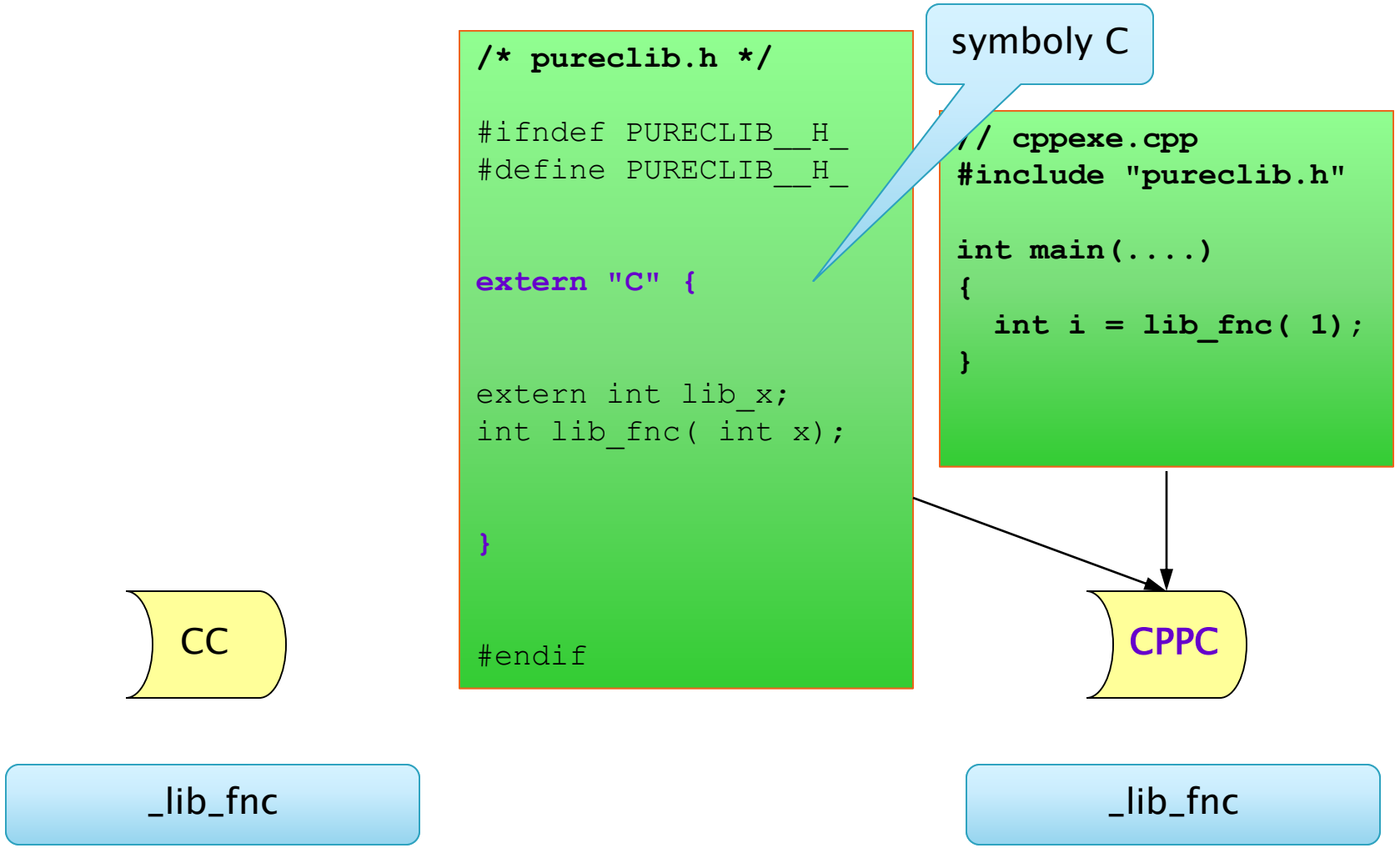


různé překladače
různé jazyky
různé konvence

_lib_fnc

?lib_fnc@@YAHH@Z

extern "C"


/* pureclib.h */
#ifndef PURECLIB__H_
#define PURECLIB__H_
extern "C" {
extern int lib_x;
int lib_fnc(int x);
}
#endif
symboly C
// cppexe.cpp
#include "pureclib.h"
int main(....)
{
int i = lib_fnc(1);
}
CC
CPPC
_lib_fnc
_lib_fnc

extern "C"

```
/* pureclib.c */
#include "pureclib.h"

int lib_x;

int lib_fnc( int x)
{
  return x + lib_x;
}
```

```
/* pureclib.h */

#ifndef PURECLIB__H_
#define PURECLIB__H_


extern "C" {


extern int lib_x;
int lib_fnc( int x);


}


#endif
```

symboly C

```
// cppexe.cpp
#include "pureclib.h"

int main(....)
{
  int i = lib_fnc( 1);
}
```

_lib_fnc

_lib_fnc

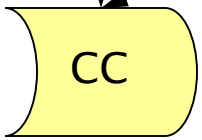
Společné hlavičkové soubory

```
/* pureclib.c */
#include "pureclib.h"

int lib_x;

int lib_fnc( int x)
{
  return x + lib_x;
}
```

```
/* pureclib.h */

#ifndef PURECLIB__H_
#define PURECLIB__H_

#ifdef  __cplusplus
extern "C" {
#endif

extern int lib_x;
int lib_fnc( int x);

#ifdef  __cplusplus
}
#endif

#endif
```

symboly C

```
// cppexe.cpp
#include "pureclib.h"

int main(....)
{
  int i = lib_fnc( 1);
}
```



_lib_fnc

CPPC – definované
CC – nedefinované

_lib_fnc

Volací konvence

- způsob implementace volání funkcí
  - registry vs. zásobník
  - zachovávání registrů
  - pořadí předávání parametrů
  - návratová hodnota
  - příprava a úklid zásobníku

```
f( 1, 2);
```

```
mov eax, 1
mov ebx, 2
call ?f@@X
```

```
mov eax, [ebp+08]
mov ebx, [ebp+04]
...
```

- nutná shoda volající a volané funkce
  - deklarace funkce

- konkrétní konvence
  - není součástí normy – rozšíření
    - __cdecl – default for C and C++, varargs
    - __stdcall – Win32 API functions
    - __fastcall – arguments in registers, faster
    - __thiscall – this
    - __clrcall – C++/CLI, .Net, managed code

C++ callback

callback
knihovní kód volá
klientskou funkci

```
/* pureclib.h */

#ifdef  __cplusplus
extern "C" {
#endif

int lib_cb( int x, int (*cb_fnc)( int));

#ifdef  __cplusplus
}
#endif
```

```
/* pureclib.c */
#include "pureclib.h"

int lib_cb( int x, int (*cb_fnc)( int))
{
  return cb_fnc( x);
}
```

```
// cppexe.cpp
#include "pureclib.h"

cpp_fnc( int x) {
  return x+1;
}

int main() {
  lib_cb( i, cpp_fnc);
}
```

x = ♀◆☺ﷺ

```
mov eax, ...
call [edx]
```

```
mov eax, [ebp+08]
add eax, 1
```

C++ callback

```
/* pureclib.h */

#ifdef __cplusplus
extern "C" {
#endif

int lib_cb( int x, int (*cb_fnc)( int));

#ifdef __cplusplus
}
#endif
```

```
/* pureclib.c */
#include "pureclib.h"

int lib_cb( int x, int (*cb_fnc)( int))
{
  return cb_fnc( x);
}
```

```
mov eax, ...
call [edx]
```

CC očekává funkci
s volací konvencí C

extern "C" určuje i
volací konvenci

```
// cpp_fnc.cpp
#include "pureclib.h"

extern "C" int cpp_fnc( int x) {
  return x+1;
}

int main() {
  lib_cb( i, cpp_fnc);
}
```

```
mov eax, [ebp+08]
add eax, 1
```

Dynamicky linkované knihovny

- použití funkcí dodaných až za běhu
- není součástí normy
  - použití na různých platformách ideově podobné
    - ale nepřenositelné
  - pomocí preprocesoru lze multiplatformní rozhraní

- Windows
  - .dll
  - chová se jako .exe
  - vlastní zásobník, heap, standardní knihovny
- Linux / Unix / POSIX
  - .so
  - chová se jako .lib
  - balíček .o

more details:

http://www.symantec.com/connect/articles/dynamic-linking-linux-and-windows-part-one

...-part-two

dll – Windows

```
// my.cpp [dll]
extern "C" __declspec(dllexport)
int add( int a, int b) {
  return a + b;
}
BOOL APIENTRY DllMain(....) {
  return TRUE;
}
```

```
// exe_explicit.cpp
HINSTANCE dll =
  LoadLibrary( TEXT("my.dll"));
if( dll == NULL)
  return 1;
typedef int dll_fnc(int, int);
dll_fnc* add = (dll_fnc*)
  GetProcAddress( dll, "add");
if( add == NULL) {
  FreeLibrary( dll);
  return 1;
}
int result = add( 1, 2);
FreeLibrary( dll);
```

load dll

explicit runtime linking

volání

```
// exe_import.cpp
extern "C" __declspec(dllimport)
int add(int a, int b);
int result = add(1, 2);
```

"statické" slinkování s moje.**lib**
jen proxy, kód v .dll

dll – POSIX

```
void *dll = dlopen("moje.so", RTLD_NOW);
typedef int dll_fnc(int,int);
dll_fnc* add = (dll_fnc*)dlsym(dll, "add");
add(…);
dlclose(dll);
```

load

volání

runtime linking

```
HINSTANCE dll =
  LoadLibrary( TEXT("moje.dll"));
if( dll == NULL)
  return 1;

typedef int dll_fnc(int, int);
dll_fnc* add = (dll_fnc*)
  GetProcAddress( dll, "add");
if( add == NULL) {
  FreeLibrary( dll);
  return 1;
}

int result = add(1, 2);

FreeLibrary( dll);
```

C++/CLI

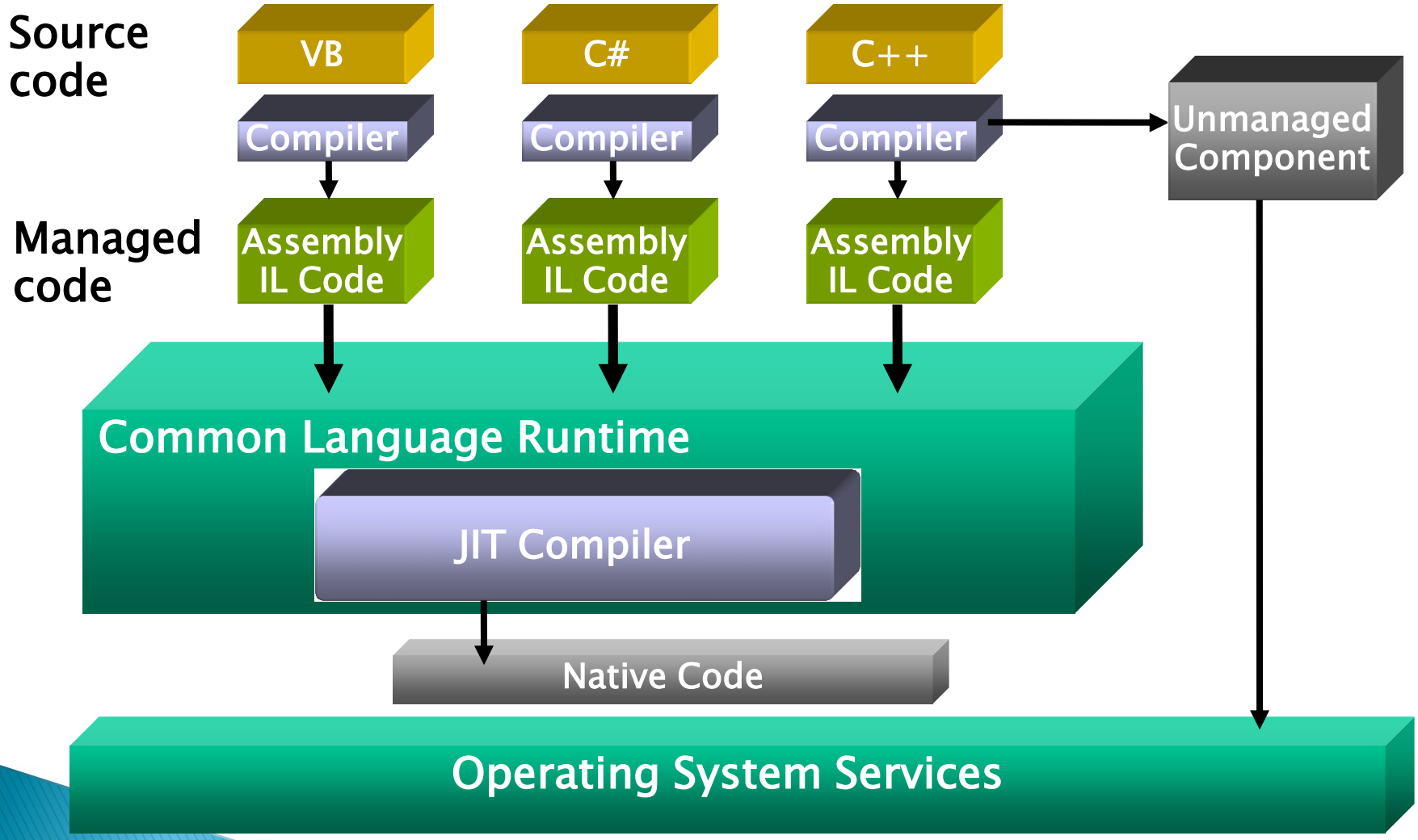
C++/CLI

- Samostatný jazyk standardizovaný ECMA
  - snaha o maximální kompatibilitu s C++ (03)

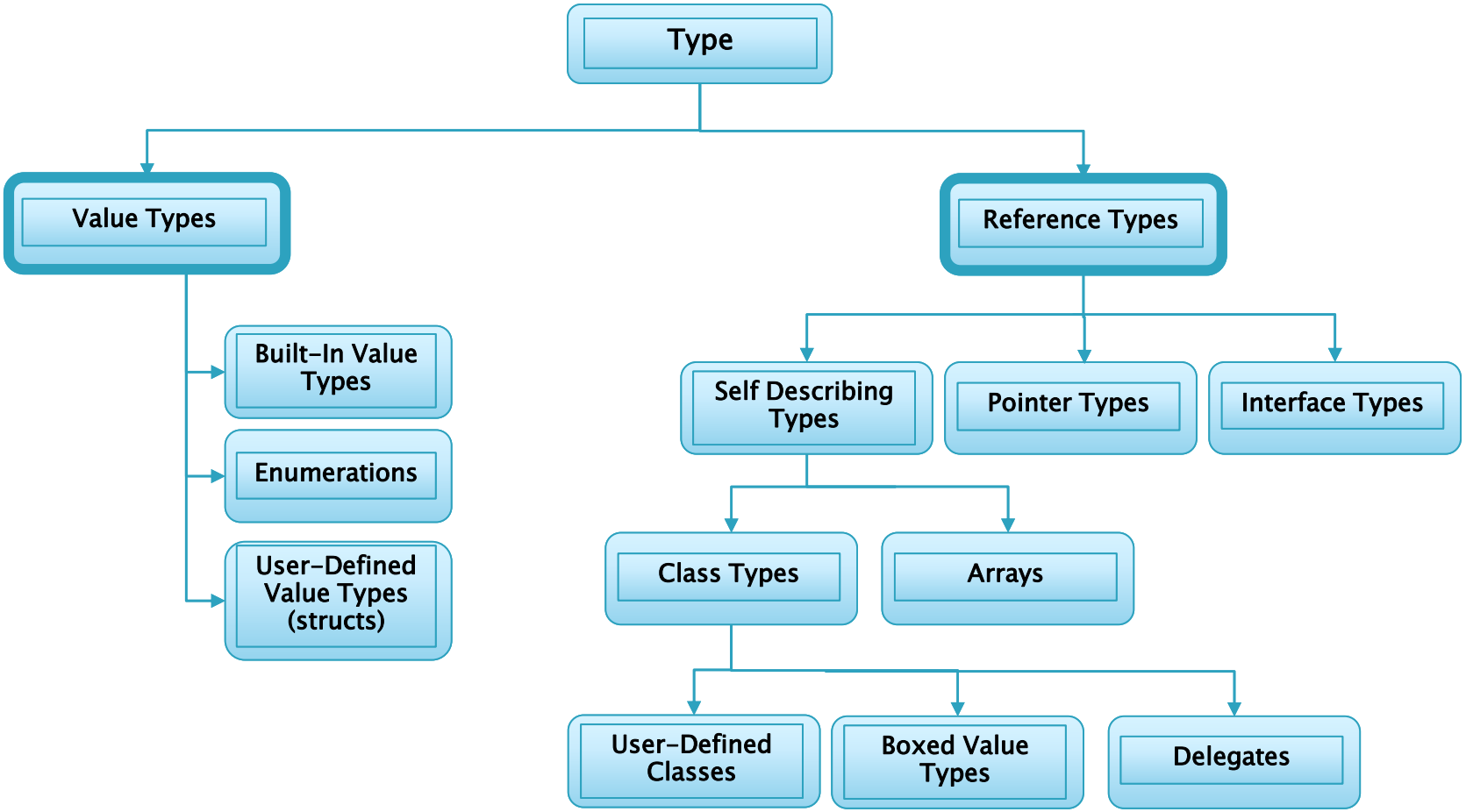
CLI	Common Language Infrastructure	standard ECMA – virtual machine – framework (libraries)
CLR	Common Language Runtime	implementace VM
CIL	Common Intermediate Language	jazyk interpretovaný VM
MSIL	MSIL :–)	konkrétní MS implementace CIL
CTS	Common Type System	jednotný typový systém CLI
.Net Framework		MS implementace nadmnožiny CLI
Mono		multiplatformní implementace CLI http://www.mono-project.com

- managed code
  - spravovaný .Net frameworkem
- přístup k .Net (CLI) knihovnám
- snadná interoperabilita
  - C#, F#, VisualBasic, … COBOL, Eiffel, Mercury …

Architektura CLI



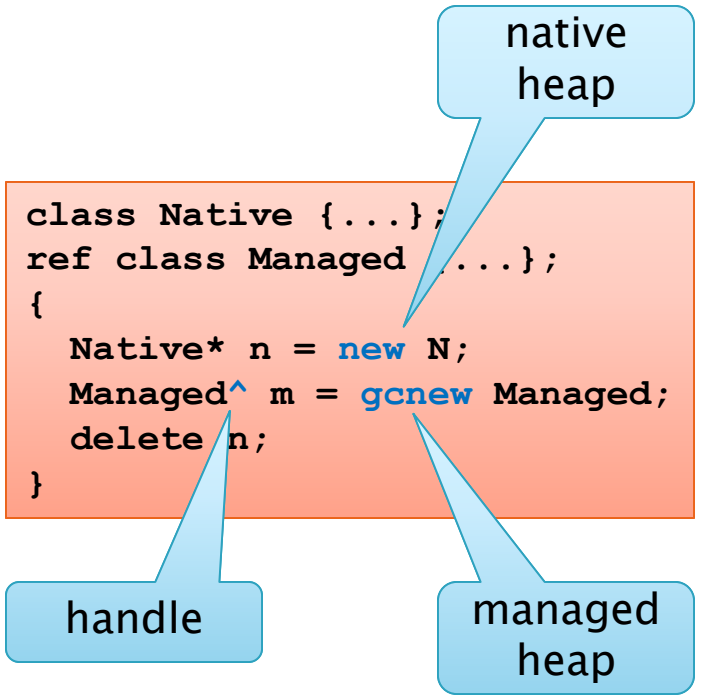
Common Type System



Typový systém C++/CLI

- Ve skutečnosti dva nezávislé typové systémy
  - native
    - original ISO C++
  - managed
    - *fuj – manažovaný – řízený*
    - CTS
  - string vs. String, vector vs. array

- Garbage collection
  - pouze CTS
  - managed heap
  - handle
    - není to ukazatel
    - může se měnit
  - nové operátory: **gcnew ^ %**
  - reference vs. value type



Referenční a hodnotové typy

Typy	hodnotové	referenční
data	primitivní typy, malé kolekce	složitější struktury
umístění	přímo v paměti (zásobník)	vždy na [managed] heapu
přístup	přímo	přes [tracking] handle
přiřazení, parametry	hodnotou	odkazem
dědičnost	`ne`	`jednoduchá` (více interfaces)
copy constr	ano	ne
default sémantika	stack heap s.: boxing	heap stack s.: autoalokace

```
value struct B { int x; };
int main() {
  B b;
  b.x = 0;
}
```

```
ref class A { int f(); };
int main() {
  A^ a = gcnew A;
  a->f();
}
```

Agregované typy

- **`ref class, ref struct`**
  - nesmí obsahovat nemanaged struktury
  - jednoduchá dědičnost, vícenásobná dědičnost interface
- **`value class, value struct`**
  - nepodporují dědičnost !
- **`enum class`**
  - rozšíření enumu o několik metod – ToString
  - value type
- **`interface class`**
  - abstract class bez dat
  - možnost vícenásobné dědičnosti
- **`array`**
  - typované vícerozměrné pole – jagged array
  - kovariance – pokud existuje konverze z A na B
    - array<A> aa; array<B> bb; bb = aa;
- **`generic`**

Další vlastnosti

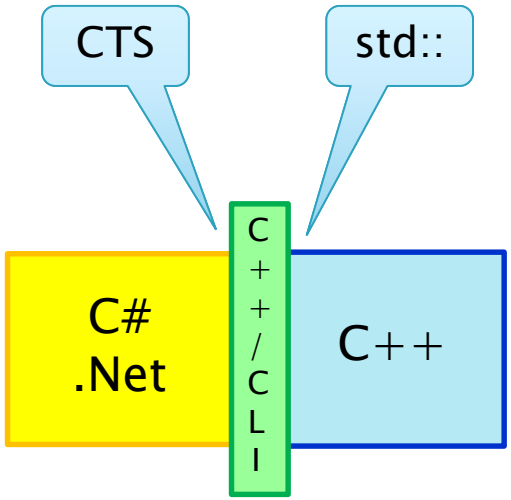
- String ≠ std::string (!!!)


```
String^ ms;
string s = (char*) Runtime::InteropServices::Marshal::
StringToHGlobalAnsi(ms).ToPointer();
```

- templates vs. generics
  - compile- vs. run- time
- CLI kolekce
  - ArrayList BitArray DictionaryBase Hashtable SortedList Stack Dictionary HashSet LinkedList List Queue SortedDictionary SortedList SortedSet Stack SynchronizedCollection
- atributy, reflexe, I/O, ..., ...

Interoperabilita

- C++/CLI ♋ C++
  - jak spojovat moduly a volat metody
    - ☺ jen std:: C++ data
  - jak sdílet managed a native data
    - ☹ nedělejte to – striktní rozhraní
    - speciální šablony
      - **ne**dělejte to!
- C++/CLI ♋ C#, *# ...
  - jak spojovat moduly
    - ☺ class library / add reference *(.dll)*
  - jak volat metody
    - ☺ class library
  - jak sdílet data
    - ☺ CTS
    - 😐 ne std:: C++



Zhodnocení

- ☺ Klady
  - **interoperabilita** – .Net, C#, CTS
  - téměř plná kompatibilita s C++03
  - managed code
- ☹ Zápory
  - dva jazyky v jednom
  - odlišné typové systémy
  - podivná syntaxe a hlavně sémantika
    - mnoho *divných* variant definovaných výčtem
  - nekompatibilní s C++11/14/17

- 👉 Use cases
  - C++ project, tenké C++/CLI rozhraní
    - .Net služby
    - interoperabilita s C#, VB, F#, *#
  - ? first-choice language
    - no!

C++/CLI a C#

Project: C++ CLR Class Library

```
namespace clilib {
  public ref class A {
  public:
    A() : x_(0) {}
    int f() {...};
  };
}
```

clilib.dll

Project: C# Console App

```
namespace App
{
  class Program {
    static void Main(string[] args) {
      clilib.A a = new clilib.A();
      int x = c.f();
    }
  }
}
```

Add Reference:
clilib.dll