Comparison of C++ and C#
Both are Important

• C++ has a huge installed base.
  • Your next employer is very likely to be a C++ house.

• C# is popular for Website development and desktop Graphical User Interface applications.
  • Lots of job openings for C# developers.

• CSE681 – Software Modeling and Analysis
  • Focuses almost exclusively on C# and .Net.

• CSE687 – Object Oriented Design:
  • Focuses almost exclusively on C++ and the Standard Library.
Comparison of Object Models

- **C++ Object Model**
  - All objects share a rich memory model:
    - Static, stack, and heap
  - Rich object life-time model:
    - Static objects live for the duration of the program.
    - Objects on stack live within a scope defined by { and }.
    - Objects on heap live at the designer’s discretion.
  - Semantics based on deep copy model.
    - That’s the good news.
    - That’s the bad news.
  - For compilation, a source file must include information about all the types it uses.
    - That’s definitely bad news.
    - But it has a work-around, e.g., design to interface not implementation. Use object factories.

- **.Net Object Model**
  - More Spartan memory model:
    - Value types are stack-based only.
    - Reference types (all user defined types and library types) live on the heap.
  - Non-deterministic life-time model:
    - All reference types are garbage collected.
    - That’s the good news.
    - That’s the bad news.
  - Semantics based on a shallow reference model.
  - For compilation, a source file is type checked with metadata provided by the types it uses.
    - That is great news.
    - It is this property that makes .Net components so simple.
Language Comparison

- **Standard C++**
  - Is an ANSI and ISO standard.
  - Has a standard library.
  - Universally available:
    - Windows, UNIX, MAC
  - Well known:
    - Large developer base.
    - Lots of books and articles.
  - Programming models supported:
    - Objects
    - Procedural
    - Generic
  - Separation of Interface from Implementation:
    - Syntactically excellent
      - Implementation is separate from class declaration.
    - Semantically poor
      - See object model comparison.

- **.Net C#**
  - Is an ECMA standard, becoming an ISO standard.
  - Has defined an ECMA library.
  - Mono project porting to UNIX
  - New, but gaining a lot of popularity
    - Developer base growing quickly.
    - Lots of books and articles.
  - Programming models supported:
    - objects.
  - Separation of Interface from Implementation:
    - Syntactically poor
      - Implementation forced in class declaration.
    - Semantically excellent
      - See object model comparison.
Library Comparison

- **Standard C++**
  - Portable across most platforms with good standards conformance
  - I/O support is stream-based
    - console, files, and strings
  - Flexible container facility using Standard Template Library (STL)
    - Now has hash-table containers
  - No support for paths and directories
  - Strings, regular expressions
  - Support for threads since C++11
  - No support for inter-process and distributed processing
  - No support for XML
  - Platform agnostic

- **.Net Framework Class Library**
  - Windows only but porting efforts underway
  - I/O support is function-based
    - console and files
  - Fixed set of containers that are not very type safe.
    - Has hash-table containers
  - Strong support for paths and directories
  - Strings and regular expressions
  - Thread support
  - Rich set of inter-process and distributed processing constructs
  - Support for XML processing
  - Deep support for Windows but very dependent on windows services like COM
## Comparison of Library Functionality

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Extendable I/O</td>
<td>Weak</td>
<td>Strong</td>
</tr>
<tr>
<td>strings</td>
<td>Strong</td>
<td>Strong</td>
</tr>
<tr>
<td>Composable Containers</td>
<td>Moderately good</td>
<td>Strong</td>
</tr>
<tr>
<td>Paths and Directories</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Threads</td>
<td>Strong</td>
<td>Good</td>
</tr>
<tr>
<td>Sockets</td>
<td>Moderately good</td>
<td>No</td>
</tr>
<tr>
<td>XML</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Forms</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Reflection</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
## Additions to C++ Library Functionality

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Support Provided in Code from Website</th>
<th>Support Provided by you in Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extendable I/O</td>
<td>NA – part of std library</td>
<td>-</td>
</tr>
<tr>
<td>strings</td>
<td>NA – part of std library</td>
<td>-</td>
</tr>
<tr>
<td>Composable Containers</td>
<td>NA – part of std library</td>
<td>-</td>
</tr>
<tr>
<td>Paths and Directories</td>
<td>FileInfo class, Path, and Directory classes</td>
<td>File Managers</td>
</tr>
<tr>
<td>Threads</td>
<td>Thread Pool</td>
<td>-</td>
</tr>
<tr>
<td>Sockets</td>
<td>Basic Demos, Socket class</td>
<td>Socket channels</td>
</tr>
<tr>
<td>XML</td>
<td>Reader, Writer, XML DOM</td>
<td>-</td>
</tr>
<tr>
<td>GUIs</td>
<td>Excellent WPF Framework</td>
<td>-</td>
</tr>
<tr>
<td>Reflection</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
End of Comparison