Course goals

Ben Langmead
ben.langmead@gmail.com
www.langmead-lab.org

Source markdown available at github.com/BenLangmead/c-cpp-notes
Course goals

Write complex, correct programs in C and C++

Leverage programming ecosystem

- Linux
- Compiler (gcc, g++)
- Debugger (gdb)
- Build system (make)

Learn basic principles of software design and engineering
Experience with Java, Python or similar

C/C++ experience *not* a prerequisite
Goals

Learn C

- Language features
- Pointers & dynamic memory allocation
- “Low-level” programming

Learn C++

- How is it different from C?
- Object-oriented programming
- Generic programming
Goals

Gain proficiency in Linux & related programming tools

- Basic command-line tools
- Compilers, debuggers, profilers

Grow as a programmer & software engineer
Why C/C++?

- Ubiquitous
- Efficient
- Mature
Much of the world’s crucial software is in C

- Used Java? JVM is written in C++, as are many libraries
- Used Python? CPython interpreter written in C
- Used the Internet? Network stacks, routers, web servers, ...
- Like science?
  - https://github.com/collections/software-in-science
  - My lab members & I program in C/C++ a lot
Ubiquitous

TIOBE Programming Community Index

Source: www.tiobe.com

www.tiobe.com/tiobe-index/

Based on search engine hits for "<language> programming"
Higher-level languages like Java & Python present a trade-off:

- High-level languages “take care of things” for you
  - Source code is more concise, abstract
  - Harder to make mistakes

- ...but also hide things from you
  - How variables are laid out in memory
  - When memory is allocated and de-allocated
  - Hardware features, especially non-portable features
Around since the 1970s (C) and 80s (C++)
Undergraduates have learned it for decades
Software jobs often require it; “we need someone who...

- ...can make something really fast if needed"
- ...knows how to program all kinds of weird hardware"
- ...knows how to interact with the operating system"
- ...can handle our large codebase, written in C"
Why C/C++?

Newer “systems languages” aim for a similar level of efficiency as C/C++

- But with simpler language
- Less “burdened” by long history & by need to stay backward-compatible

Examples

- Swift – developer.apple.com/swift
- Go – golang.org
- Rust – rust-lang.org
Why C/C++?

On efficiency, they approach C/C++

But they do not approach C/C++ in maturity/ubiquity

- Some, like Swift, are associated with (& tied to) particular companies
## Why C/C++?

<table>
<thead>
<tr>
<th>Apr 2018</th>
<th>Apr 2017</th>
<th>Change</th>
<th>Programming Language</th>
<th>Ratings</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Java</td>
<td>15.777%</td>
<td>+0.21%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td>C</td>
<td>13.589%</td>
<td>+6.62%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td>C++</td>
<td>7.218%</td>
<td>+2.66%</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>↑</td>
<td>Python</td>
<td>5.803%</td>
<td>+2.35%</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>↓</td>
<td>C#</td>
<td>5.265%</td>
<td>+1.69%</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>↑</td>
<td>Visual Basic .NET</td>
<td>4.947%</td>
<td>+1.70%</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>↓</td>
<td>PHP</td>
<td>4.218%</td>
<td>+0.84%</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td>JavaScript</td>
<td>3.492%</td>
<td>+0.64%</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>↑</td>
<td>SQL</td>
<td>2.650%</td>
<td>+2.65%</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>↑</td>
<td>Ruby</td>
<td>2.018%</td>
<td>-0.29%</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>↓</td>
<td>Delphi/Object Pascal</td>
<td>1.961%</td>
<td>-0.86%</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>↑</td>
<td>R</td>
<td>1.806%</td>
<td>-0.33%</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>↑</td>
<td>Visual Basic</td>
<td>1.798%</td>
<td>-0.26%</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>↓</td>
<td>Assembly language</td>
<td>1.655%</td>
<td>-0.51%</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>↓</td>
<td>Swift</td>
<td>1.534%</td>
<td>-0.75%</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>↓</td>
<td>Perl</td>
<td>1.527%</td>
<td>-0.89%</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td></td>
<td>MATLAB</td>
<td>1.457%</td>
<td>-0.59%</td>
</tr>
<tr>
<td>18</td>
<td>14</td>
<td>↓</td>
<td>Objective-C</td>
<td>1.250%</td>
<td>-0.91%</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>↓</td>
<td>Go</td>
<td>1.180%</td>
<td>-0.79%</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td></td>
<td>PL/SQL</td>
<td>1.173%</td>
<td>-0.45%</td>
</tr>
</tbody>
</table>

[www.tiobe.com/tiobe-index/]