Greetings Fellow D-velopers!

This presentation is available online at

http://flowvel.la/wonk
confessions of a C++ wonk

why D is very, very good for me!

chuck allison
Definition (from dictionary.com)

**wonk**

[wongk]

Examples

1. a student who spends much time studying and has little or no social life; grind.

2. a stupid, boring, or unattractive person.

3. a person who studies a subject or issue in an excessively assiduous and thorough manner:
   *a policy wonk.*
The Language Landscape

Let’s visit tiobe.com
how did we get here?

I was there…
“Baseball … been berry, berry good to me.”

(aka Garrett Morris)
“C++ has been very, very good to me.”
Books

1998
English
Chinese

2004
English
Chinese
Czech
Polish

2010
English
Chinese
Czech
Russian
Japanese
Korean
credit where credit is due

Adopted Organically (Like C)

C++ took OOP to “the masses”

http://padlet.com/chuck_allison/cpp_good
(double-click)
My Two Cents

IT'S CALLED YOUR TWO CENTS
BECAUSE IT'S BASICALLY WORTHLESS
Pre-History

1975 – FORTRAN

1984 – Mark Williams C

1987 – Datalight C
1989–1990

Early adopter of Zortech C++

Wrote 5 chapters of the Turbo-Borland C++ documentation
1991
ANSI Committee J16
“Programming Language C++”

Present at first technical meeting
(March 1991, Nashua, NH)
– represented 3 organizations
– served actively 10 years

Designed/implemented std::bitset
And boost::dynamic_bitset
– with Jeremy Siek
Met this fellow named Walter Bright

What is defensive programming? Like Defensive Driving
Programming under the assumption that:
• anything can happen
• variables contain invalid values
• coprogrammers are out to get you!
• comments lie!
• unspecified behavior is pathological
• correct behavior must be demonstrated
1992–2003

“Code Capsules” Columnist
1992–1994

Consulting Editor, 1996–2001

Senior Editor, 2001–2003
2003–2008

Founder and Editor

The C++ Source
The Premier Online Journal for the C++ Community

Enforcing Code Feature Requirements in C++
by Scott Meyers, September 23, 2008, 16 comments

Advisory Board

David Abrahams
Andrei Alexandrescu
Matthew Austern
Walter Bright
Steve Clamage
Greg Colvin
Jim Coplien
Stephen Dewhurst
Neil Harrison
Kevlin Henney
Howard Hinnant

Andrew Koenig
Angelika Langer
Nathan Myers
Eric Niebler
Thorsten Ottosen
Thomas Plum
Dan Saks
Jeremy Siek
Bjarne Stroustrup
Herb Sutter
Matthew Wilson

http://www.artima.com/cppsource
Why Java Succeeded

- Novelty of Internet
- Portability of VM
- Garbage Collection
- No Raw Pointers!
The D Programming Language
by Walter Bright

D is an advanced systems programming language. It is designed to appeal to C and C++ programmers who need
a more powerful language that has a much lower syntax and hence is easier to master. D is multiplatform,
and looks and feels very much like C and C++. It offers opportunities for advanced compilers to generate more
efficient code than is possible for C/C++, while supporting facilities that reduce the probability of program bugs.

Why D?
Refactoring
C++ has been around for 20 years, but D++ has largely succeeded in adding enormous capability to C++ while
retaining backwards compatibility with it. But with 20 years experience comes the opportunity to reflect on how one
might engineer a language that retains C++ strengths, add modern features, and remove its weaknesses and
more troublesome aspects.

Difficulty in adding modern new features
The longer a language has been evolving, the harder it gets to add new features. Each new feature adds an
untampered layer on top of old ones, in a way that no legacy code breaks. Eventually, it becomes forever to add an
insignificant improvement. The D++ "tapir" is an extreme example of this effect. Designing and then testing a few
years to implement and delivering the apparent benefit. A more mundane observation is that C++ was
standardized 5 years ago and just now conforming compilers are emerging.

While C++ is pioneering generic programming practice, it lags behind in other modern techniques such as design
by contract, modules, automated testing, and static memory management. It's very difficult to add these while
still supporting legacy code.

Brief Tour
D looks a great deal like C and C++, so much so that the canonical hello world program is nearly identical:

```d
import std.stdio;

main()
{
    writeln("Hello world!");
    return 0;
}
```

Look and feel is very much like C and C++

Many years ago in grammar school, we were shown a film about a researcher who wore special goggles that
turned the world upside down. He wore them continuously such that his brain never saw the world right side up.
After 2 weeks, his brain suddenly righted that upside down view. Then, the researcher took the goggles off. The
film dually warned the viewer to not try that ourselves!

I am so, so used to C/C++ syntax that I feel like that poor guy when faced with a new and improved language that
turns the syntax inside out (or so it looks to me). Frankly, I rarely give such languages a chance even when
the feature set looks intriguing. D doesn’t take that route. Its syntax is as comfortable to C/C++ programmers as
an old shoe. Functions, statements, expressions, operator precedence, integral promotions, it’s all there pretty
much unchanged. The world is right side up. It’s just got brighter colors and sharper focus.
why is D Very Good for you?

http://padlet.com/chuck_allison/d_good
why D is very, very good for me!
All D-velopers Love...

D compiles to *native code*

(Optional) **Garbage Collection**

Module System

**Slices**

**Associative Arrays**

**static if**

**Universal Function Call Syntax**

Compile-Time Function Evaluation

Mixins (string and template)

**unittest**

Contract Programming

**debug**
## Other Cool Language Features

<table>
<thead>
<tr>
<th>Slices</th>
<th>Delegates &amp; Decorators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array-wise operations</td>
<td>Delegates &amp; Bound Methods</td>
</tr>
<tr>
<td>lazy evaluation</td>
<td>shared</td>
</tr>
</tbody>
</table>
The Python Library

“Batteries Included”
The D Library

“Nuclear Reactor Included”
Cool Library Features

- scoped
- partial fun evaluation
- Discriminated unions
- memoize
- Fibers
- Ranges
D-veloper Tools & Resources

rdmd
dlang.org
dub
vibe.d
Xamarin Studio (Mono-D, Mac/Linux)

Books: Alexandrescu, Çehreli, Ruppe

http://padlet.com/chuck_allison/d_tools
more to come!

Erich Gubler
Friday, 3:30pm