D PROGRAMMING LANGUAGE

What is it and how we can embedded it

Damian Ziemba
https://github.com/nazriel/cehug

CZYM JEST D?

CZYM JEST D?

- Kompilowany do kodu maszynowego

CZYM JEST D?

- Kompilowany do kodu maszynowego
- Statycznie typowany

CZYM JEST D?

- Kompilowany do kodu maszynowego
- Statycznie typowany
- Składnia C-like

CZYM JEST D?

- Kompilowany do kodu maszynowego
- Statycznie typowany
- Składnia C-like
- W pełni OpenSource

CZYM JEST D?

- Kompilowany do kodu maszynowego
- Statycznie typowany
- Składnia C-like
- W pełni OpenSource
- Język programowania systemowego

CZYM JEST D?

- Kompilowany do kodu maszynowego
- Statycznie typowany
- Składnia C-like
- W pełni OpenSource
- Język programowania systemowego
- Wieloparadygmatowy

HELLO WORLD

```
#!/usr/bin/env rdmd
import std.stdio;

void main()
{
    writeln("Witaj świecie");
}
```

```
nazriel@nyx-wifi ~/Workspace/D/cehug/hello
> $ chmod +x ./hello.d

nazriel@nyx-wifi ~/Workspace/D/cehug/hello
> $ ./hello.d
Witaj świecie
```

HISTORIA

- ?/1999 - Walter Bright decyduję się na stworzenie nowego języka

HISTORIA

- ?/1999 - Walter Bright decyduję się na stworzenie nowego języka
- 12/2001 - pierwsze publiczne wydanie

HISTORIA

- ?/1999 - Walter Bright decyduję się na stworzenie nowego języka
- 12/2001 - pierwsze publiczne wydanie
- 01/2007 - D osiąga krok milowy w postaci wersji 1.0

HISTORIA

- ?/1999 - Walter Bright decyduję się na stworzenie nowego języka
- 12/2001 - pierwsze publiczne wydanie
- 01/2007 - D osiąga krok milowy w postaci wersji 1.0
- 06/2007 - rozpoczęcie prac nad wersją 2.0 języka

HISTORIA

- ?/1999 - Walter Bright decyduję się na stworzenie nowego języka
- 12/2001 - pierwsze publiczne wydanie
- 01/2007 - D osiąga krok milowy w postaci wersji 1.0
- 06/2007 - rozpoczęcie prac nad wersją 2.0 języka
- 12/2010 - Andrei Alexandrescu wydaje książkę "The D Programming Language"

HISTORIA

- ?/1999 - Walter Bright decyduję się na stworzenie nowego języka
- 12/2001 - pierwsze publiczne wydanie
- 01/2007 - D osiąga krok milowy w postaci wersji 1.0
- 06/2007 - rozpoczęcie prac nad wersją 2.0 języka
- 12/2010 - Andrei Alexandrescu wydaje książkę "The D Programming Language"
- 01/2011 - GitHub staję się oficjalnym narzędzie do współtworzenia

HISTORIA

- ?/1999 - Walter Bright decyduję się na stworzenie nowego języka
- 12/2001 - pierwsze publiczne wydanie
- 01/2007 - D osiąga krok milowy w postaci wersji 1.0
- 06/2007 - rozpoczęcie prac nad wersją 2.0 języka
- 12/2010 - Andrei Alexandrescu wydaje książkę "The D Programming Language"
- 01/2011 - GitHub staję się oficjalnym narzędzie do współtworzenia
- 12/2011 - oficjalne ogłoszenie końca wsparcia dla wersji 1.0 języka

ŚRODOWISKO

Kompilator

DMD

GDC

LDC

+

toolchain danej platformy:

- binutils: ld, ar, as, gcc (FreeBSD, Linux, OSX)
- VisualStudio/Digital Mars/Watcom (Windows)

ŚRODOWISKO

IDE/Edytory/Debuggery/Inne

IDE

- DlangIDE - natywne środowisko napisane w D + DlangUI
- DLang Intellij - plugin do JetBrains IDEA
- DDT - plugin do Eclipse
- Visual-D - plugin do Visual Studio
- Mono-D - plugin do Xamarin Studio aka Visual Studio
- Code::blocks - oficjalne wsparcie

Inne

- DCD - completion daemon
- dfmt - formatowanie kodu
- difx - automatyczne aktualizowanie kodu
- dscanner - lint/ctag generator/all-in-one
- libdparse - lexer + parser jako biblioteka

Edytory

- Atom - kilka pluginów dostępnych
- Emacs - oficjalne wsparcie
- Geany - oficjalne wsparcie
- KWrite/Kate - oficjalne wsparcie
- Notepad++ - oficjalne wsparcie
- Sublime Text - kilka pluginów dostępnych
- Vim - plugin
- Visual Studio Code - kilka pluginów dostępnych

Debug

- DWARF - GDB, LLDB
- COFF - Mago/VisualD
- OMF - WinDBG, OllyDBG2, Mago/VisualD
- Frontendy/GUI do w/w

ŚRODOWISKO

Wspierane platformy

	DMD	GDC	LDC
System	• Windows • Linux • OS X • FreeBSD • Android (source)	• Windows (alpha) • Linux • OS X (untested) • FreeBSD (untested)	• Windows • Linux • OS X • FreeBSD • OpenSolaris • iOS (experimental) • Android (alpha)
Architektura	• i386 • amd64	• i386 • amd64 • x32 • armel • armhf • alpha • arm64 (aarch64) • hppa • m68k • mips • mipsel • ppc • pcc64	• i386 • amd64 • armel • arm64 (aarch64) • ppc • ppc64 • ppc64el • mips64 • armhf • mips • s390x

Źródło: https://wiki.dlang.org/Compilers

Dlaczego warto używać?

UNIT TEST

```
import std.exception;

int dodaj(int x, int y) { return x + y; }
void wyrzućCoś() { throw new Exception("Wyjątek!"); }

unittest
{
    assert(dodaj(2, 2) == 4); // #1
    assert(dodaj(2, 2) == 5, "Jak się to mogło stać?"); // #2
    assertNotThrown(2.dodaj(2) == 4); // #3
    assertThrown(wyrzućCoś()); // #4
}
```

```
nazriel@nyx-wifi ~/Workspace/D/cehug/unittest
> $ dmd -main -unittest -run unittest.d
core.exception.AssertError@unittest.d(9): Jak się to mogło stać?
----------------
4   dmd_runMI58XR                    0x0000000101ac55c2 _d_unitt
5   dmd_runMI58XR                    0x0000000101abc7e6 void uni
```

UNIFORM FUNCTION CALL SYNTAX

```
module ucfs;
import std.stdio, std.algorithm.iteration, std.range;
void main()
{
    writeln("Typowy ", "flow");
    writeln(filter!(a => a % 2 == 0)(iota(10)));

    "W stylu ".writeln("UFCS");
    10.iota
        .filter!(a => a % 2 == 0)
        .writeln();
}
```

```
nazriel@nyx-wifi ~/Workspace/D/cehug/ufcs
> $ ldc2 -run ufcs.d
Typowy flow
[0, 2, 4, 6, 8]
W stylu UFCS
[0, 2, 4, 6, 8]
```

SCOPE GUARD

```
import core.stdc.stdio, core.stdc.string: strlen;
import std.string: toStringz;

void otworz_i_zapisz() {
    FILE* f = fopen("/tmp/test.c".toStringz(), "w+".toStringz());
    // cokolwiek sie nie stanie... nowa linia zostanie dopisana, a handle zostanie zamkniety
    scope(exit) { fwrite("\n".ptr, 1, 1, f); fclose(f); }

    const char* udalo = "udalo sie";
    const char* nie_udalo = "nie udalo sie";

    scope(success) { fwrite(udalo, typeof(*udalo).sizeof, udalo.strlen(), f); }
    scope(failure) { fwrite(nie_udalo, typeof(*nie_udalo).sizeof, nie_udalo.strlen(), f); }
    throw new Exception("nie uda sie!");
}

void main() {
    otworz_i_zapisz();
}
```

SCOPE GUARD

```
nazriel@nyx-wifi ~/Workspace/D/cehug/guard
> $ dmd -run scope_guard.d ; echo ; echo ; cat /tmp/test.c
object.Exception@scope_guard.d(14): nie uda sie!
----------------
4   dmd_runzbxzsV                    0x00000001061567af void scope_guard.otworz_
5   dmd_runzbxzsV                    0x0000000106156660 _Dmain + 8
6   dmd_runzbxzsV                    0x0000000106170477 D2rt6dmain211_d_run_main
7   dmd_runzbxzsV                    0x000000010617039f void rt.dmain2._d_run_ma
8   dmd_runzbxzsV                    0x000000010617041c void rt.dmain2._d_run_ma
9   dmd_runzbxzsV                    0x000000010617039f void rt.dmain2._d_run_ma
10  dmd_runzbxzsV                    0x0000000106170312 _d_run_main + 458
11  dmd_runzbxzsV                    0x00000001061566f7 main + 15
12  libdyld.dylib                    0x00007fffd0c55234 start + 0
13  ???                              0x0000000000000000 0x0 + 0


nie udalo sie
```

CONTRACTS

```
uint kwadrat(int a, int b)
in {
    assert(a >= 0, "A musi byc wieksze od 1!");
    assert(b > 0, "B musi byc wieksze od 0!");
}
out(wynik) {
    assert(wynik != 0, "Cos poszlo nie tak..?");
}
body {
    return a * b;
}

unittest {
    assert(kwadrat(0, 1) == 1);
}
```

```
nazriel@nyx-wifi ~/Workspace/D/cehug/contract
> $ ldc2 -main -unittest -run contract.d
core.exception.AssertError@contract.d(7): Cos poszło nie tak..?
----------------
Error: /var/folders/9p/84qwl21x3nx7xkw2z_6j0blh0000gn/T/contract-f
```

- in - walidacja danych na wejściu
- out - walidacja wartości zwracanej
- body - ciała funkcji/metody

COMPILE TIME FUNCTION EVALUATION

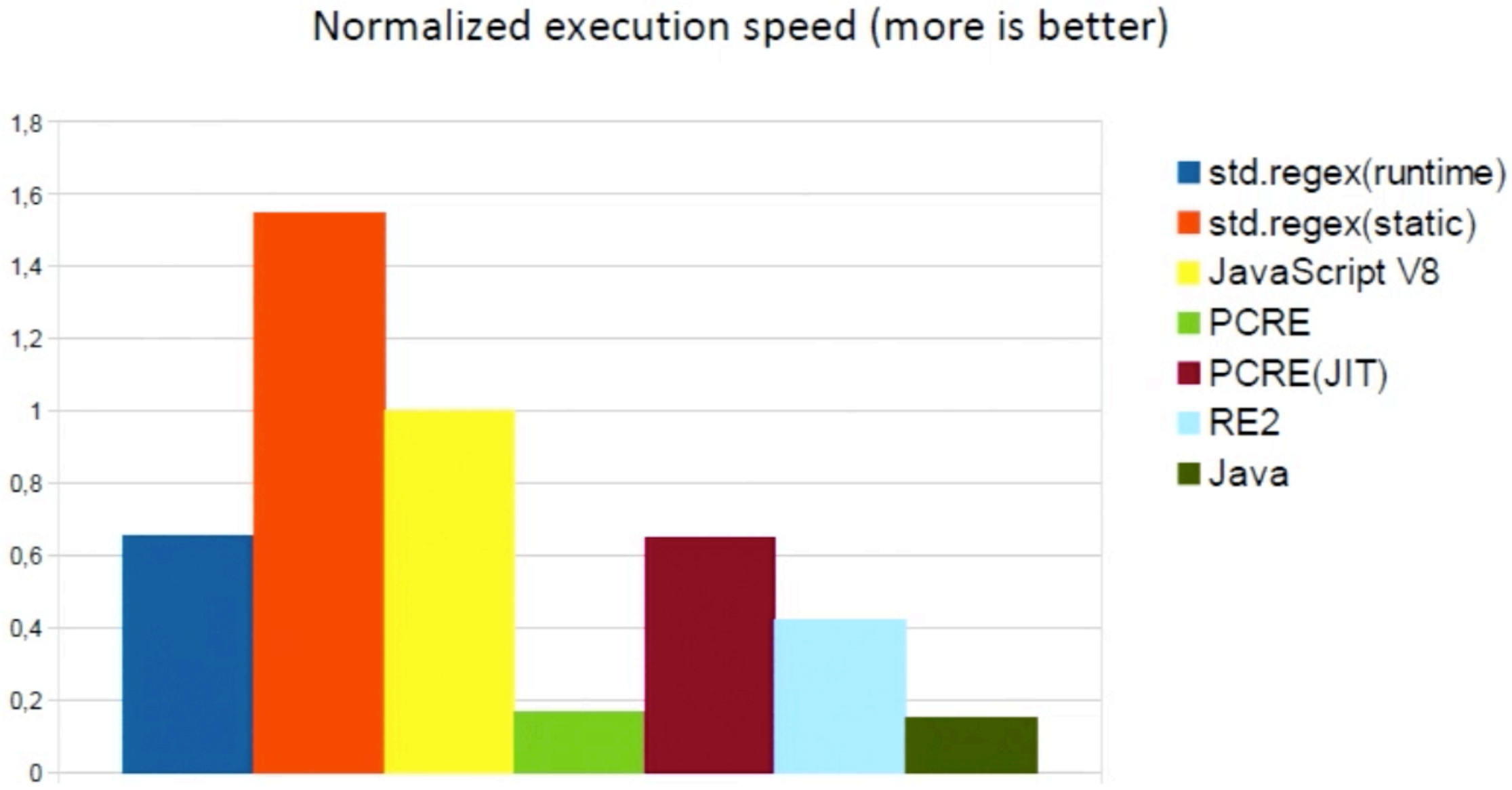
```
uint kwadrat(uint a, uint b) { return a * b; }

int main() {
    // compile time
    enum kw = kwadrat(9, 9);

    // runtime
    return kwadrat(2, kw);
}
```

```
nazriel@nyx-wifi ~/Workspace/D/cehug/ctfe
> $ dmd -c ctfe.d && objdump -S -x86-asm-syntax=intel ctfe.o | grep -A 4 "_Dmain"
__Dmain:
       0:   55  push    rbp
       1:   48 8b ec    mov     rbp, rsp
       4:   be 02 00 00 00      mov     esi, 2
       9:   bf 51 00 00 00      mov     edi, 81
--
       e:   e8 00 00 00 00      call    0 <__Dmain+0x13>
      13:   5d  pop     rbp
      14:   c3  ret
      15:   0f 1f 00    nop     dword ptr [rax]
```

Regex-DNA, popular benchmark



META-PROGRAMMING

SZABLONY

```
import std.stdio;

void funkcjaSzablonowaWariadyczna(Typy...)(Typy obiekty) {
    foreach (num, obiekt; obiekty) {
        write(obiekt);
        if (obiekty.length - 1 != num) { write(", "); }
       else { writeln(); }
    }
}

void funkcjaSzablonowaWariadyczna(Typ : float)(Typ f) {
    writeln("specjalizja dla float'u");
}

void funkcjaZWarunkiem(int n)() if (n > 10) {
    writeln("Jest wiecej niż 10, biorę!");
}

void funckcjaZDomyslnymTypem(T = string)() {
    writeln("typem o ktorym mysle jest: ", T.stringof);
}

struct StrukturaZWariadikiem(T...) {
    T args;
    void przekaz(T argumenty) {}
}
```

```
void main() {
    funkcjaSzablonowaWariadyczna("Witaj",
213, "świecie");
    funkcjaSzablonowaWariadyczna(1.23);
    funkcjaSzablonowaWariadyczna("D jak
dobro");
    funkcjaSzablonowaWariadyczna();
    funkcjaZWarunkiem!15;
    funckcjaZDomyslnymTypem();
    funckcjaZDomyslnymTypem!int();

    StrukturaZWariadikiem!(int, float,
string) s;
    s.przekaz(s.args);
   // wyswietlone podczas kompilacji
    pragma(msg, typeof(s.args));
}
```

```
nazriel@nyx-wifi ~/Workspace/D/cehug/templates-staticif-reflection
> $ ldc2 templates.d -of=templates && ./templates
(int, float, string)
Witaj, 213, świecie
specjalizja dla float'u
D jak dobro
Jest wiecej niż 10, biorę!
typem o ktorym mysle jest: string
typem o ktorym mysle jest: int
```

STATIC IF

```
class Wtyczka { void wsadz(){ } }
class Kontakt { void przyjmij(){ } }

void przyjmij(Typ)(Typ obiekt) {
    // duzo kodu ktorego szkoda by bylo
    static if (is(Typ : Wtyczka))
        obiekt.wsadz();
    else static if (is(typeof(obiekt) : Kontakt))
        obiekt.przyjmij();
    else
        pragma(msg, "Dziwny typ... moze sprawiac problemy", Typ.stringof);
    // duplikowac tylko zeby zrobic specjalizacje
}

void main() {
    przyjmij(new Wtyczka()); przyjmij(new Kontakt()); przyjmij(23);
}
```

REFLEKSJA PODCZAS KOMPILACJI

```
module refleksja;
import std.traits, std.stdio;

struct prad {}
@prad class wtyczka { string pole; int zasieg; }
@prad final class wtyczunia { void bar(){} }

void przyjmij(Typy...)() {
    foreach (Typ; Typy) {
        static if (hasMember!(Typ, "bar")) {
            new Typ().bar();
        } else static if (isFinal!Typ) {
            static assert(__traits(compiles, { class Dziedzic : wtyczunia {} }));
        }
    }
}

void main() {
    foreach (symbol; getSymbolsByUDA!(refleksja, prad)) {
        pragma(msg, symbol.stringof);
    }

    przyjmij!(wtyczka, wtyczunia);
}
```

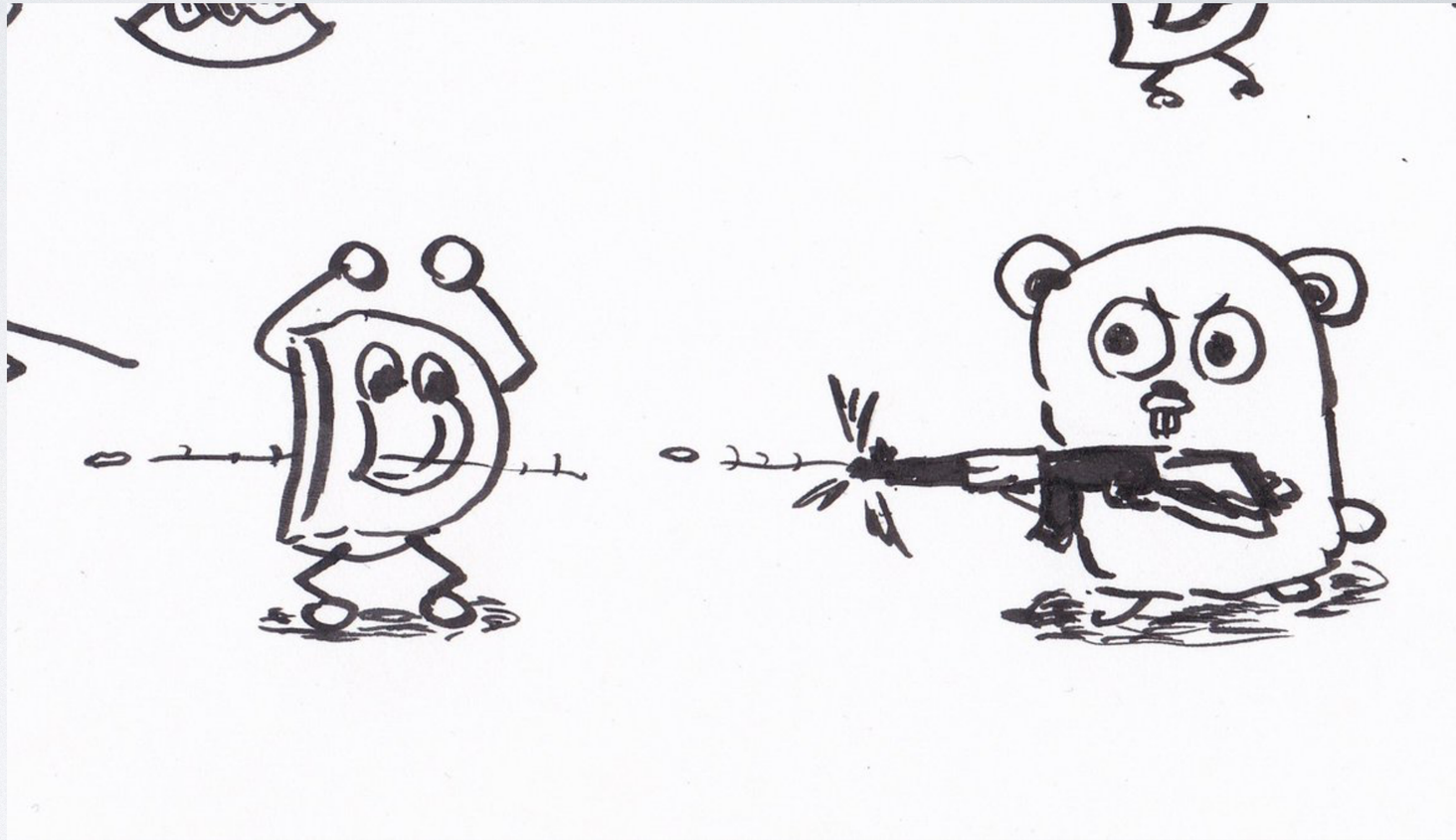
```
nazriel@nyx-wifi ~/Workspace/D/cehug/templates-staticif-reflection
> $ ldc2 -d reflection.d -of=reflection
wtyczka
wtyczunia
reflection.d(13): Error: static assert  (__traits(compiles, ()
{
class Dziedzic : wtyczka
{
void* this;
}
}
)) is false
reflection.d(23):        instantiated from here: przyjmij!(wtyczka, wtyczunia)
```

DLANG W "DZICZY"

Komercyjne zastosowanie D

- Facebook - lint i preprocesor dla C++
- Remedy Games - logika gier
- Sociomantic - real-time aukcje
- Weka.IO - high-io storage
- Funkwerk Aktiengesellschaft - embedded systemy; bilbordy informacyjny na przystankach
- Ebay - narzędzia/skrypty
- infognition.com - video processing
- tripaneer.com - infrastruktura webowa

Więcej:
https://wiki.dlang.org/Current_D_Use
http://dlang.org/orgs-using-d.html



GAMEDEV

- Bindingi i wrappery do popularnych bibliotek
  - https://github.com/DerelictOrg
  - https://github.com/d-gamedev-team
- Wiele dostępnych tutoriali
  - https://github.com/d-gamedev-team/opengl-tutorials
  - https://github.com/kiith-sa/d-gamedev-intro
- Blogi poświęcone tematyce
  - http://h3.gd/code/
  - https://www.gamedev.net/blog/1140-d-bits/
- Gotowe gry
  - mnóstwo; włączając w to duże studia typu Remedy Games
  - przykład https://github.com/WebFreak001/SpaceD

OS DEV

- Wymagania względem języka:
  - inline-assembler ✔
  - wskaźniki oraz arytmetyka na wskaźnikach ✔
  - kod maszynowy jako wynik kompilacji ✔
  - samo-zawierający się plik binarny ✔
  - operacje bitowe & literał binarny ✔
  - łatwa interakcja z innymi językami (C/C++/Obj-C) ✔
- Projekty
  - x86_64 Kernel PowerNex: https://github.com/Vild/PowerNex
  - x86_64 exokernel XOMB: http://wiki.xomb.org

WEB DEVELOPMENT

- Front-End
  - LDC + asm.js
  - vibe.d + generatory dla JS (rest interfejsy, websockety itd).
- Back-End
  - vibe.d - https://vibed.org
  - ae.net - https://github.com/CyberShadow/ae
  - dub - manager pakietów - https://code.dlang.org
- Przykłady:
  - https://forum.dlang.org
  - https://vibed.org/blog

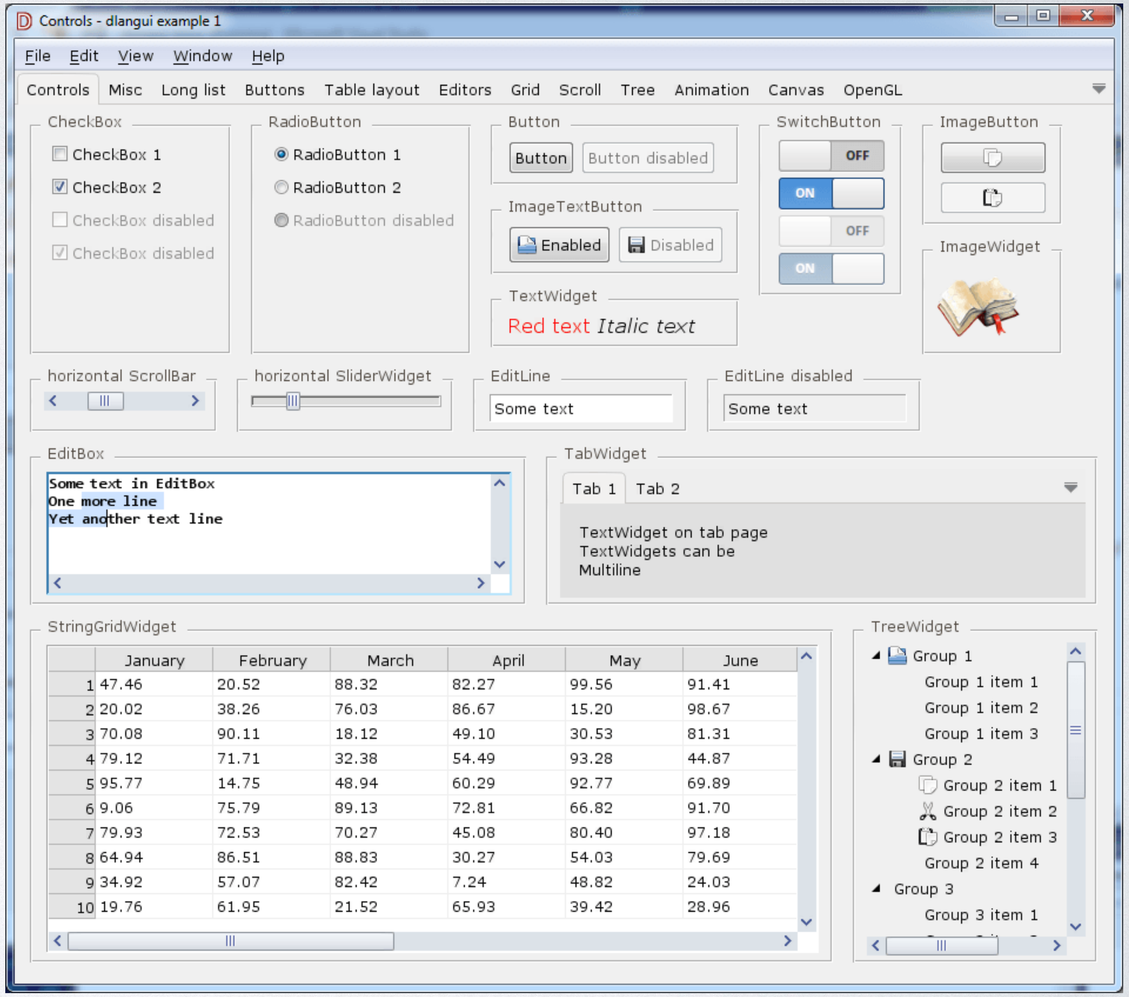
WEB DEVELOPMENT

- Front-End
  - LDC + asm.js
- B
- F
  - https://forum.dlang.org
  - https://vibed.org/blog

```
import vibe.d;
shared static this()
{
  auto settings = new HTTPServerSettings;
  settings.port = 8080;
  settings.bindAddresses = ["::1", "127.0.0.1"];
  listenHTTP(settings, (req, res) => res.writeBody("Hello, World!"));
  logInfo("Please open http://127.0.0.1:8080/ in your browser.");
}
```

SCRIPTING + GUI + MOBILE

- Skryptowanie
  - Obsługa sekwencji shebang
  - Szybka kompilacja
  - Krótki turn-around-time
    Przykład: https://github.com/eBay/tsv-utils-dlang
- GUI
  - łatwo interakcja z istniejącym kodem w innych językach (C/C++/Obj-C)
    Przykłady:
    - http://gtkd.org - Binding oraz obiektowy wrapper dla GTK
    - https://github.com/buggins/dlangui - natywny framework wspierający platformy: Win32, OSX, Linux (X11), Android, Konsola
    - https://github.com/nomad-software/tkd - Tcl/Tk wrapper
- Mobile
  - iOS ✔ https://github.com/smolt/ldc-iphone-dev
  - Android ✔ https://wiki.dlang.org/Build_LDC_for_Android
  - WindowsPhone ❓
  - BlackBerry ❓



EMBEDDED DEVELOPMENT

ŚRODOWISKO

Community/Dokumentacja/Przydatne linki

- Strona główna: https://dlang.org
- Forum/News-group: https://forum.dlang.org
- Oficjalny blog: https://dlang.org/blog
- Wiki: https://wiki.dlang.org/
- Language reference: https://dlang.org/spec/spec.html
- Biblioteka standardowa: https://dlang.org/phobos/index.html
- Repozytorium paczek: https://code.dlang.org
- Przewodnik po języku: https://tour.dlang.org/
- Kompilator online: https://dpaste.dzfl.pl
- Książki: https://wiki.dlang.org/Books
- SO: http://stackoverflow.com/questions/tagged/d
- Zbiór artykułów: http://dlang.org/articles.html
- Github: https://github.com/dlang

PYTANIA?