The D Programming Language

by Walter Bright
Digital Mars

http://www.digitalmars.com/d/

What is D?

- Systems and applications programming language
  - Native code generation
  - Static typing
  - Fast turnaround
- Born of decades of experience with industry projects
- Multi-paradigm

Multiparadigm

- Imperative
- Metal
- Object-oriented
- RAII
- Functional
- Generic
- Generative
- Concurrent

Imperative Programming

```
import std.stdio;

void main(string[] args) {
  writeln("hello world");
  writefln("args.length = %s",
        args.length);
  foreach (arg; args)
      writefln("arg = '%s'", arg);
}
```

Metal Programming

```
int *_memset32(int *p,
               int value,
               size_t count) {
    asm   {
        mov     EDI,p               ;
        mov     EAX,value           ;
        mov     ECX,count           ;
        mov     EDX,EDI             ;
        rep                         ;
        stosd                       ;
        mov     EAX,EDX             ;
    }
}
```

Object Oriented

```
class Shape {
  abstract void Draw ();
}

class Square : Shape {
  this(int x, int y, int w) {
    xpos = x; ypos = y;
    width = w;
  }

  void Draw() {
    writefln("Drawing Square at (%s,%s), width %s\n",
      x, y, width);
  }

  private int x, y, width;
}
```

RAII

```
struct Buffer {
  this(size_t s) {
    buf = malloc(s)[0 .. s];
  }
  this(this) {
    buf = buf.dup;
  }
  ~this() {
    free(buf.ptr);
  }
  void[] buf;
}
```

Functional

```
pure sum_of_squares
  (immutable double[] a)
{
    auto sum = 0;
    foreach (i; a)
        sum += i * i;
    return sum;
}
```

Generic

```
size_t levenshteinDistance
   (alias equals = "a == b", Range1, Range2)
   (Range1 s, Range2 t)
   if (isForwardRange!(Range1) &&
       isForwardRange!(Range2))
{
   ...
}
```

* Works with arbitrary predicates
* Lightweight concepts in the form of template constraints

Generative

```
struct A {
    int a;
    mixin(bitfields!(
        uint, "x",    2,
        int,  "y",    3,
        uint, "z",    2,
        bool, "flag", 1));
}
A obj;
obj.x = 2;
obj.z = obj.x;
```

Concurrent

```
import std.algorithm, std.concurrency, std.stdio;

void main() {
 enum bufferSize = 1024 * 100;
 auto tid = spawn(&fileWriter);
 // Read loop
 foreach (immutable(ubyte)[] buffer;
          stdin.byChunk(bufferSize)) {
   send(tid, buffer);
 }
}

void fileWriter() {
 // Write loop
 for (;;) {
   auto buffer = receiveOnly!(immutable(ubyte)[])();
   tgt.write(buffer);
 }
}
```

from The D Programming Language chapter 13

Compilers

- Digital Mars D compiler
  - Based on the Digital Mars compiler suite
- Gnu D compiler
  - Based on the gnu compiler collection
- LDC compiler
  - Based on the LLVM compiler

Full source code available for all of them

Platform Support

- Windows
- Linux
- OS X