# D Programming

# D Programming

In nutshell

Jonathan MERCIER

October 17, 2012







# Plan

- Introduction
  - Object Functional

  - Meta-programming Parallelism

  - Ressource Management
  - Contract.
  - System and Safe Code • Reference and pointer
  - Generics
  - Inference
  - Loops
  - Functions
  - Debugs
  - Versions
  - Requirement
  - Editors
- Basics
  - My first D program
  - Types
  - Arrays
  - String and characters
  - Const and Immutable
  - Input/Output
  - Algorithm
  - a Structure and Class
  - Template
  - Miscellanous
  - a Let start it!
- GTK D Thanks To
- D Programming









### Introduction

Meta-programming
Parallelism

Ressource Management Contract

System and Safe Code

Generics

Inferenc

Loops

Function Debugs

Versions Bequireme

Editors

#### Basic

My first D program

Array

String and characters

Input/Output

Algorithm

Structure and Cla

Miscellanc

Let start

# Before starting









## Introduction

# Why a new language?

# Few significant dates

• C++ 1983

4/122



## Introduction

Meta-programming Parallelism

Ressource Management

System and Safe Code

Reference and pointe

Generic

Loops

Functions

Functions

Debugs

versions

Requirem

#### Editors

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Cla

Tempiate Miscellano

Lot stant

#### TK D

# Why a new language?

# Few significant dates

- C++ 1983
- Java 1990

4/122





## Introduction

Meta-programming
Parallelism

Contract

System and Safe Code

Reference and point Generics

Inferenc

Interend

Loops

Function

Versions

Requireme

Editors

#### Basics

My first D program

Types Arrays

String and characters

Const and Immutabl

Algorithm

Structure and Clas

Miscelland

Let start

OWN D

# Why a new language?

# Few significant dates

- C++ 1983
- Java 1990
- Python 1995





## Introduction

Meta-programming
Parallelism

Ressource Management Contract

System and Safe Code

Generics

Inference

interenc

Loops

Functio

Debugs

Versions

Requireme

Editors

#### Basic

My first D program

Types

String and characters

Const and Immutabl

Algorithm

Structure and Cla

Miscellano

Let start

### TK D

# Why a new language?

# Few significant dates

- C++ 1983
- Java 1990
- Python 1995
- Ruby 1995





## Introduction

Meta-programming
Parallelism

Ressource Management Contract

System and Safe Code Reference and pointer

Generics

Inferenc

Loops

Functio

Debugs

Vorcion

Requirem

Editors

#### Basic

My first D program

Arravs

String and characters

Const and Immutable Input/Output

Algorithm

Structure and Clas

Miscelland

Let start

# Why a new language?

# Few significant dates

- C++ 1983
- Java 1990
- Python 1995
- Ruby 1995
- And now?





### Introduction

Meta-programming
Parallelism
Ressource Managemen

System and Safe Code

Reference an Generics

Loops

Function

Debugs

Versions

Requireme

#### Basic

My first D program

Arrays

String and characters Const and Immutable

Algorithm

Structure and Class

Miscelland

Let start

amu p

# What is D programming?

D is a modern language programming inspired by:

- C++
- Java
- Haskell
- Python
- Ruby





## Introduction

Meta-programming

Ressource Management

System and Safe Code

Reference and pointer

Generics

Inferenc

Loops

Functio

Debugs

¥7----

Requirem

Requirem

#### Basic

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Cla

Template Miscellan

Lot otant

TK D

# Why a new language?

## D Combines

• Modeling Power

6/122





## Introduction

Meta-programming Parallelism

Ressource Management Contract

System and Safe Code

Generics

Inferenc

·

Daniel Control

- .

Debugs

Version

Requirer

Editors

#### Basics

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Cla

Miscellano

Let start

TKD

# Why a new language?

## D Combines

- Modeling Power
- Modern Convenience





## Introduction

Meta-programming
Parallelism

Contract System and Safe Code

Reference and pointer

Generics

Interenc

Loops

Function

Debugs

Versions

Requirem

Editors

#### Basic

My first D program

Types

Arrays

Const and Immutable

Input/Output

Algorithm

Structure and Cla

Miscellano

Let start

TK D

# Why a new language?

## D Combines

- Modeling Power
- Modern Convenience
- Native Efficiency





# Plan

### Introduction

- Object









#### Introduction

#### Object

Meta-programming

Parallelism
Ressource Manageme

Contract
System and Safe Cod

Reference and pointer

Generics

Inference

Loons

Functio

Debugs

Versions

Requirement

#### Basics

My first D program

Types

String and characters

Const and Immutab

Algorithm

Structure and Cla

Miscelland

Lot start

# TK D

# Object

## • Interface

1 interface foo { ...}

8/122





#### Introduction

#### Object

Meta-programming

Parallelism

Parallelism

Ressource Manageme

Contract
System and Safe Cod

Reference and pointer

Generics

Inference

Loops

Functio

Debugs

Versions

Requirement Editors

#### Basics

My first D program

Types

String and characters

Const and Immutab

Algorithm

Structure and Cla

Miscelland

Lot start

## TK D

# Object

- Interface
  - 1 interface foo { ...}
- class
  - 1 class bar { ...}





### Introduction

#### Object

Meta-programming Parallelism

Ressource Managemer

System and Safe Code

Reference and pointer

Inference

Loops

Functio

Debugs

Versions

Requireme

#### Basic

My first D program

Types Arrays

String and characters

Const and Immutab

Algorithm

Structure and Cla

Template Miscelland

Lot start

### TK D

# Object

- Interface
  - 1 interface foo { ...}
- class
  - 1 class bar { ...}
- inheritance
  - 1 class bar: foo  $\{ \dots \}$





# Object

- Interface
  - 1 interface foo { ...}
- class
  - 1 class bar { ...}
- inheritance
  - 1 class bar: foo { ...}
- multi class inheritance not allowed, instead used interface.





## Introduction

#### Function

Meta-programming

Ressource Manageme

System and Safe Code

Reference and pointer

Generics

Inferen

Loons

Functi

Debugs

Requiren

r. equireii

#### Basic

My first D program

Types

String and characters

Const and Immutable

Input/Out

Algorithm

Structure and Cla

Template

wiisceman

Let start

# Plan

#### Introduction

- Functional
- Functional
- Meta-programming
- Ressource
- C--t---t
- Contract
- System and Safe Code
   Reference and pointer
- Generic
- Innerer
- Loops
- Functions
- Debugs
- Versions
- Requirement

#### Basics

- My first D program
  - Types
  - Arrays
  - String and characters
  - Const and Immutable
  - Input/Output
  - Algorithm
  - Structure and Class
  - Templa
  - Miscellanous
- CTK D
- GTK E

D Programming





### Introduction

#### Function

Meta-programming Parallelism

Ressource Managemen

System and Safe Cod

Reference and p

Y C

Interenc

пооры

Functio

Debugs

Versions

Requirem

Editors

#### Basics

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Cla

Miscellano

Let start

## TK D

# Functional

# • Data immutability

1 immutable int[] a = [4, 6, 1, 2];







### Introduction

#### Function

Meta-programming
Parallelism

Ressource Managemen

System and Safe Cod

Reference and point

Generics

Inferenc

Loops

Functio

Debugs

Versions

Requirem

Editors

#### Basic

My first D program

Arrave

String and characters

Const and Immutab

Algorithm

Structure and Cla

1 emplate Miscellano

Lot start

## TK D

# Functional

- Data immutability
  - 1 immutable int[] a = [4, 6, 1, 2];
- Pure functions
  - 1 pure int square(int x) { return x \* x; }





# Functional

- Data immutability
  - 1 immutable int[] a = [4, 6, 1, 2];
- Pure functions
  - 1 pure int square(int x) { return x \* x; }
- Lambda functions
  - 1 a.sort!( (x,y) => x < y); // [1, 2, 4, 6]







### Introduction

#### Meta-programming

Ressource Managemer

System and Safe Code

Reference and pointer

Generics

Inferen

Loops

Functio

Debugs

Version

Require

Editors

#### Basic

My first D program

Types

Carrays

String and characters

Const and Immutable

4.1

Structure and Cla

Template

T ---

Let start

# Plan

### Introduction

- Object
- Functional

## $\bullet$ Meta-programming

- Paranensin
- Ressource Management
- Contract
- System and Safe Code
   Reference and pointer
- Generic
- Interen
- Loops
- Functions
- Debugs
- Versions
- Requirement

#### Basics

- My first D program
  - Types
  - Arrays
  - String and characters
  - Const and Immutable
  - Input/Output
- Algorithm
- Structure and Class
- Templ
- Miscellanous
- CTK D







### Introductio

#### Meta-programming

Ressource Management

System and Safe Cod

Reference and pointe

Generic

Inference

Loops

Function

Debugs

Version

Requiren

Editors

#### Basics

My first D program

Types

String and characters

Const and Immutabl

Algorithm

Structure and C

Template

Lot start

amrr n

# Meta-programming

## Combination of

• templates







### Introductio

#### Meta-programming

Ressource Manageme

System and Safe Coo

Generic

T ... 6 . . . . . .

Y ----

Function

Debugs

Version

.

Editors

#### Basic

My first D program

Types

String an

Const and Immutable

Input/Output

Algorithm

Structure and Cla

Miscellano

Let start

# Meta-programming

## Combination of

- templates
- compile time function execution





## Introductio

#### Meta-programming

Ressource Manageme

System and Safe Coo

Generic

Inforonc

Loops

Functio

Debugs

versions

Requirem

Editors

#### Basic

My first D program

Types

String and characters

Const and Immutabl

Algorithm

Structure and Cla

Miscellano

I ot otout

TK D

# Meta-programming

## Combination of

- templates
- compile time function execution
- tuples





### Introductio

### Parallelism

Ressource Manageme

System and Safe Coo

Reference and poir

Generic

Interent

Loops

Function

Debugs

versions -

Editors

#### Basic

My first D program

Types

String an

Const and Immutable

input/Out

Algorithm

Structure and Cla

Template

Y --- ----

OTHE D

# Meta-programming

## Combination of

- templates
- compile time function execution
- tuples
- string mixins





# Meta-programming

# Code 1: Example

```
template Factorial(ulong n){
    static if(n < 2)
3
     const Factorial = 1;
    else
     const Factorial = n * Factorial!(n - 1);
6
  const ulong var = Factorial!( 8 ); // compute at compile-time
```







# Plan

### Introduction

- Parallelism









### Introductio

## Meta-progra

Parallelism

Contract

System and Safe Code

Reference and point

Generic

Interenc

Loops

Function

Debugs

Versions

Requirement

#### Basics

My first D program

Types

String and characters

Const and Immutab

Algorithm

Structure and Cla

Miscelland

I of otont

י אידי

# Parallelism

• module to use

1 import std.parallelism;





### Introductio

### Meta-prog

Ressource Manageme

System and Safe Cod

Reference and pointe

Generics

Inference

\_\_\_\_\_

r unctio

¥7----

- Versions

Requirem

#### Editors

My first D program

Types

String and characters

Const and Immutab

Algorithm

Structure and Cla

Miscelland

Lot otost

#### TK D

# Parallelism

- module to use
  - 1 import std.parallelism;
- parallel loop
  - 1 foreach( i; parallel( list ) ){ ...}





# Parallelism

- module to use
  - 1 import std.parallelism;
- parallel loop
  - 1 foreach( i; parallel( list ) ){ ...}
- Pool thread
  - 1 void myfunction(int param1, int param 2) { ...}
  - auto myTask = task!myfunction( param1, param2 );
  - taskPool.put( myTask );
  - doSomething(): // another work in parallel
  - taskPool.finish( true ); // wait alls jobs ending





### Introduction

### THEFOGUCETO

Parallelism

### Ressource Managemen

System and Safe Code

#### Reference and pointe

Generics

#### Inference

Loops

#### . .

- .

#### Debugs

Daniman

#### Requirem

Editors

#### Basic

My first D program

Types

Arrays

#### String and characters

Const and Immutable

#### Input/Out

Algorithm

### Structure and Cla

Template

wiscenan

Let start

# Plan

#### Introduction

- Dogect
- Functional
- Meta-programming

#### • Ressource Management

- Contrac
- System and Safe Code
- Generic
- Imerei
- Loop:
- Function
- Debugs
- Versions
- Requirement

#### Rocice

- My first D program
  - Types
  - Arrays
  - String and characters
  - Const and Immutable
  - Input/Outpu
- · Algorithm
- Structure and Class
- Template
- Miscellanous
- OTEN D
- Thonks T







### Introduction

Parallelism Ressource Management

Contract

System and Safe Code Reference and pointer

Generics

Inference

Loops

Eunetic

Functio

V

Requirem

Editors

#### Basic

My first D program

Arrave

String and characters

Const and Immutabl

Algorithm

Structure and Clas

Miscellano

Let start

TK D

# Ressource Management

# Code 2: Example

- File f = File( "myfile.txt", "r");
  - scope(exit) f.close();
    lockFile( f );
- 3 lockFile( f );
  4 doFoo( f );
- 5 scope(success) doBar(f);
- 6 scope(failure) unlock(f);





# Plan

#### Introduction

#### Contract.

D Programming





### Introductio

Meta-programming Parallelism

Ressource Management

Contrac

System and Safe Code

Reference and pointe

Generic

Interen

Function

Functions

Debugs

Requiremen

Requirement Editors

#### Basic

My first D program

Types

String and characters

Const and Immutab

Algorithm

Structure and Cla

Miscellan

Let start

## TK D

# Contract

• check a statement

```
1 assert( var != null );
```

19/122





Contract

System and Safe Cod

Generics

Loops

Functions Debugs

Versions Requiremen

Editors

#### Basic

My first D program

Types Arrays

String and characters

Const and Immutable
Input/Output

Algorithm

Structure and Clas Template

Let start

amiz D

# Contract

• check a statement

```
1 assert( var != null );
```

• check before entering into a function

- 1 double foo ( int a )
- 2 in{ assert( a > 0 ); }
- 3 body { return a 2; }





### Introductio

Parallelism

## Contract

System and Safe Code

Generics

Loops

Function

Debugs

Versions

Requireme

### Basic

My first D program

Arrays

String and characters Const and Immutable

Const and Immutab Input/Output

Algorithm

Structure and Cla Template

Miscelland

Let start

# Contract

• check a statement

```
1 assert( var != null );
```

• check before entering into a function

```
1 double foo ( int a )
2 in{ assert( a > 0 ); }
```

2 in{ assert( 
$$a > 0$$
 ); } 3 body { return  $a - 2$ ; }

• check at function exit

```
1 int foo (int a)
```

3 body { return 
$$a - 2$$
; }





# Contract

check a statement

```
1 assert( var != null );
```

• check before entering into a function

```
double foo (int a)
2 in{ assert( a > 0 ); }
```

```
3 body { return a - 2: }
```

• check at function exit

```
1 int foo (int a)
2 out{ assert( a > 0 ); }
3 body { return a - 2: }
```

• -release flag will not compute contract

```
$ ldc2 -release foo.d
```





# Plan

### Introduction

# • System and Safe Code



D Programming







# System and Safe Code

# System and Safe Code

D Programming

Safe functions are functions that are statically checked to have no possibility of undefined behavior.

Undefined behavior is often used as a vector for malicious attacks.

Functions are marked with at-tributes: @safe, @system, @trusted

21/122

& Back

· Forward

# Plan

### Introduction

### • Reference and pointer



D Programming







## Jonathan MERCIEF

### Introductio

Meta-programming
Parallelism

Contract System and Safe Code

Reference and pointer

Generics

Inference

Interenc

Loops

Function

Debugs

Versions

Requirem

### Editors

My first D program

Types

String and characters

Const and Immutabl

Algorithm

Structure and Cla

Miscelland

I ot otant

# Reference and pointer

• Pointers exist only to create C interface code

```
1 int* a = cFunction( param );
```



## Jonathan MERCIER

### Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Reference and pointer

Inference

Inferenc

Loops

Functio

Debugs

Versions

Editors

### Basic

My first D program

Types Arrays

String and characters

Const and Immutab

Algorithm

Structure and Clas

Miscellano

Let start

## TK D

# Reference and pointer

- Pointers exist only to create C interface code
  - $1 \ \ {\bf int*} \ a = cFunction(\ param\ );$
- ref into function
  - 1 void foo( ref int[] param ) { ...}





## Jonathan MERCIEF

### Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Reference and pointer

Inference

Inferenc

Loops

Function

V----

versions

Requiremen

### Basic

My first D program

Types Arrays

String and characters

Const and Immutab Input/Output

Algorithm

Structure and Class

l'emplate Miscellano

Let start

# Reference and pointer

- Pointers exist only to create C interface code
  - $1 \ \ \mathbf{int*} \ \mathbf{a} = \mathbf{cFunction(\ param\ )};$
- ref into function
  - 1 void foo( ref int[] param ) { ...}
- ref into a loop
  - 1 foreach( ref item ; list ){ ...}





### Generics

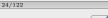
# Plan

### Introduction

### • Generics



### D Programming





### Jonathan MERCIER

### Introduction

Meta-programming Parallelism

Ressource Managemen

System and Safe Code

### Generics

Inference

IIIICICI

Pour seise

- ..

Debugs

- Versions

Requirem

### Basics

My first D program

Types

Arrays

String and characters

Const and Immutab

Algorithm

Structure and Cla

Miscellano

1/2

TK D

# Generic

- class
  - 1 class Foo( T ){ ...}
  - $2 \ \ Foo! \underline{int} \ instance = new \ Foo! \underline{(int)( \ param \ )};$





### Jonathan MERCIEI

### Introductio

Meta-programming Parallelism

Ressource Management

System and Safe Code

### Generics

T ... 6 .....

IIIICICII

Loops

Function

Debug

Versions

Requiren

Requirem

### Basic

My first D program

Types

String and characters

Const and Immutab

Al---ith--

Structure and Cla

Miscellano

1/4

TTK D

# Generic

- class
  - 1 class Foo( T ){ ...}
  - 2 Foo!int instance = new Foo!(int)( param );
- structure
  - 1 struct Foo( T ){ ...}
  - 2 Foo!int instance = Foo!int( param );





### Generics

## Generic

- class
  - 1 class Foo( T ){ ...}
  - 2 Foo!int instance = new Foo!(int)( param );
- structure
  - 1 struct Foo(T){ ...}
  - 2 Foo!int instance = Foo!int( param );
- function
  - 1 T foo( T )(T param){ ...}
  - 2 int var = foo!int( param );







Ressource Management Contract

Reference and pointer

## Generics

Inference Loops Functions Debugs

Requireme

### Basic

My first D program Types

String and characters
Const and Immutable

Algorithm

Structure and Class Template

Let start it!

# Generic

- class
  - 1 class Foo( T ){ ...}
  - 2 Foo!int instance = new Foo!(int)( param );
- structure
  - 1 struct Foo( T ){ ...}
  - 2 Foo!int instance = Foo!int( param );
- function
  - 1 T foo(T)(T param){ ...} 2 int var = foo!int(param);
- macro
  - 1 template TFoo( T )( T param ){ immutable T f = param + 3;
  - 2 int a = TFoo!int(4); // return 7 at compile time





### Inference

# Plan

### Introduction

### Inference



D Programming





### Inference

# Inference

- auto for variable

  - 1 size\_t[] list = [0, 1, 2, 3, 4]; 2 auto item = list[1]; // item type is size\_t

Contract
System and Safe Code

Reference and pointer Generics

Inference

Functio

Debugs

Versions

Requirem

Editors

### Basics

My first D program

Arravs

String and characters

Const and Immutabl

Algorithm

Structure and Class Template

Miscelland

Let start

# Inference

- auto for variable
  - 1 size\_t[] list = [ 0, 1, 2, 3, 4]; 2 auto item = list[1]; // item type is size\_t
- auto for function
  - 1 auto foo( int param ){ ...}



# Plan

### Introduction

### Loops

D Programming









# Loops

• for loop

```
1 for (int i = 0; i < 10; i++) { ...}
```

• while loop

```
1 while (isComputing) { ... }
```

• do while

```
1 do{ ...} while( isComputing );
```

• foreach loop

```
1 foreach( size_t i; list ){ ...}
2 foreach( size_t counter, size_t i = 0; list ){ ...}
3 foreach( counter, i; list ){ ...}
```





# Plan

### Introduction

### • Functions









### Jonathan MERCIEF

### Introductio

Meta-programming
Parallelism

Ressource Management Contract

System and Safe Code

Reference and point

Generics

Interend

Function

T uncero.

Debugs

Versions

Requirement Editors

### Basic

My first D program

Types

String and characters

Const and Immutah

Algorithm

Structure and Cla

Miscellanc

I of stant

COTT D

# Functions

## Functions

- classical
  - 1 void foo( int param ){ ...}

31/122

D Programming





### Jonathan MERCIEF

### Introductio

Parallelism

Ressource Management

Contract
System and Safe Code

Reference and pointer

Generics

Inferenc

Loops

Function

Debugs

Versions

Requiremen

### 2000

My first D program

Types

String and characters

Const and Immutab

Algorithm

Structure and Cla

Miscellano

Lot start

### TK D

# **Functions**

## Functions

- classical
  - 1 void foo( int param ){ ...}
- with default values
  - 1 void foo( int param1, int param2 = 3 ){ ...}





### Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Reference and pointe

Generics

Loops

Function

Debugs

Versions

Requireme

### Basic

My first D program

Arrays

String and characters

Input/Out

Structure and Cla

Tempiate Miscellano

Lot start

TK D

```
Functions
```

- classical
  - 1 void foo( int param ){ ...}
- with default values
  - 1 void foo( int param1, int param2 = 3 ){ ...}
- with variadic parameters

```
1 void foo( int[] params ... ) {
2  foreach( param; params )
3    ...// do something
4 }
```





# Plan

### Introduction

- Debugs

32/122

60 Back





# Debugs

# Debugs

debug block

```
1 debug(int param){ ...} // ldc2 -d-debug ...
```

debug line

```
debug writeln( "foo" );
```

unitest

```
1 unitest ·
    assert( doFoo( x ), true );
    assert(doBar(x), 3);
5
```





```
Debugs
```

debug block

```
1 debug(int param){ ...}
```

debug line

```
1 debug writeln( "foo" ); // ldc2 -d-debug ... x
```

unitest

```
1 unitest {
    assert( doFoo( x ), true );
    assert(doBar(x), 3);
5
```





# Debugs

debug block

```
1 debug(int param){ ...}
```

debug line

```
debug writeln( "foo" );
```

unitest

```
1 unitest { // ldc2 -unittest ...
    assert( doFoo( x ), true );
    assert(doBar(x), 3);
5
```





# Plan

### Introduction

### Versions

### D Programming







### Jonathan MERCIEF

### Introduction

Parallelism

Contract System and Safe Code

Reference and pointer

Generics

Inference

Loops

Functio

Debugs

### · .

Editors

### Basic

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Cla

Miscellano

I of start

TK D

# Versions

## Versions

- version conditional code block to use
  - 1 version(Windows){ ...}
  - 2 else version( linux ) { ...}
  - 3 else { pragma(  ${\rm msg,~"Unknown~operating~syystem"}$  ); }





### Jonathan MERCIEF

### Introductio

Meta-programming Parallelism

Ressource Managemer

System and Safe Cod

Reference and point

Generic

T ----

Function

Daham

Debugs

Version

Editors

### Basic

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Cla

Miscellano

Lot start

### TK D

# Versions

# Versions

- own version identifier
  - 1 version(FullApp){...}
  - 2 else version( DemoApp ){ ...}
  - 3 else { version = DemoApp ); }





### Introduction

Meta-programming
Parallelism
Ressource Managemen

Contract
System and Safe Code

Reference and pointer Generics

Loops

Function

Debugs

Require

### Basic

My first D program

Arrays

String and characters Const and Immutable

Algorithm

Structure and Class Template

. Miscellano

Let star

## Versions

- own version identifier
  - 1 version(FullApp){...}
  - 2 else version( DemoApp ) { ...}
  - 3 else { version = DemoApp ); }
- from command line give wich version to compile

Code 4: Terminal

\$ ldc2 -d-version="FullApp" myApp.d





# Plan

### Introduction

### Requirement

D Programming

39/122

60 Back





## Jonathan MERCIEI

### Introductio

Meta-programming
Parallelism

Contract

System and Safe Code

Generics

Inferenc

Loops

Functi

Debugs

Versions

Requireme

Editors

### Basics

My first D program

Types

String and c

Const and Immutable

Input/Output

Algorithm

Structure and Cla

rempiate Miscellano

Let start

## TTIZ T

# Before beginning...

## Tools

- Compiler: ldc
- Standard library: phobos
- GUI library: gtkd







Editors

# Plan

### Introduction

- Editors

D Programming





### Jonathan MERCIER

### Introductio

Meta-programming Parallelism

Contract

System and Safe Code

Generics

Inference

.//

Functio

Debugs

Versions

Requirem

Editors

### Basics

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Cla

Miscelland

Lot otant

# Editors

# Tools

Geany



D Programming



### Jonathan MERCIEF

### Introductio

Meta-programming Parallelism

Ressource Managemen Contract

System and Safe Code

Generics

Inference

T ----

Functio

Debugs

Debugo

Requiren

Editors

### D . . . .

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Clas

Miscelland

I ot otost

### TK D

# Editors

## Tools

- Geany
- MonoDevelop with Mono-D





### Jonathan MERCIER

### Introductio

Meta-programming
Parallelism

Contract

Reference and pointer

Generics

I . . . . .

Functio

. . .

Debugo

. .

Editors

My first D program

Types

String and characters

Const and Immutabl

Algorithm

Structure and Clas Template

Lot start

Let start

# Editors

### Tools

- Geany
- MonoDevelop with Mono-D
- Eclipse with DDT D Development Tools





## Introductio

Meta-programming
Parallelism

Ressource Management Contract

System and Safe Code Reference and pointer

Generics

Inference

Loops

Functio

Debugs

Versions

Require

Editors

### Basic

My first D program

Types

String and characters

Const and Immutabl

Algorithm

Structure and Cla

Miscellano

Let start

Let start i

## Tools

- Geany
- MonoDevelop with Mono-D
- Eclipse with DDT D Development Tools
- Vim + syntastic





Editors

## Tools

- Geany
- MonoDevelop with Mono-D
- Eclipse with DDT D Development Tools
- Vim + syntastic
- Emacs + d-mode (GitHub)



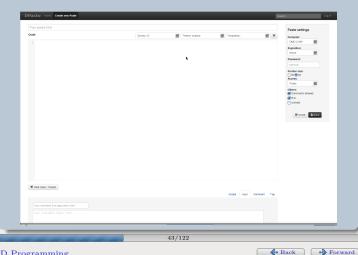


Editors

## Editors

## Online tools: D paste

http://dpaste.dzfl.pl/new



### Introductio

Meta-programming Parallelism

Ressource Management

System and Safe Cod

Reference and pointe Generics

T ... C .......

Interence

Loops

Function

Debugs

versions

Editors

### ъ . . .

My first D program

Types

String and character

Const and Immutable

Algorithm

Structure and Cla

Miscelland

Let start

### TK D

## Installation

## Code 5: Terminal

 $\#\ yum\ install\ ldc-phobos-devel\ gtkd-devel$ 





## Plan

## Basics

- My first D program

D Programming





# My first D program

```
Code 6: hello.d
  module hello;
  import std.stdio;
3
  void main () {
   writeln( "Hello world" );
6
```



### Introductio

Meta-programming Parallelism

Contract

System and Safe Code

Generics

Inference

-

Functio

Vorciona

Requiren

Editors

### Basics

### My first D program

Types Arrays

String and characters

Const and Immutable Input/Output

Algorithm

Structure and Class

Tempiate Miscellano

Let start

# My first D program

```
Code 7: hello.d

1 module hello;
2 import std.stdio;
3
4 void main () {
5 writeln( "Hello world" );
6 }
```



### Introductio

Meta-programming Parallelism

Contract

System and Safe Cod

Generics

Inference

Interenc

\_

Functio

Debugs

versions

Requiren

### Basics

### My first D program

Types Arrays

String and characters

Const and Immutable Input/Output

Algorithm

Structure and Clas

Miscellano

Let start

# My first D program

```
Code 8: hello.d

1 module hello;
2 import std.stdio: writeln;
3
4 void main () {
5 writeln( "Hello world" );
6 }
```



### Introduction

Meta-programming
Parallelism

Contract System and Safe Code

Reference and pointer

Generics

Inferenc

Loops

Functio

Debugs

Versions

Requirem

### Basics

### My first D program

Types

String and characters

Const and Immutable Input/Output

Algorithm

Structure and Clas Template

Miscelland

Let start

# My first D program

```
Code 9: hello.d

1 module hello;
2 import std.stdio: writeln;
3
4 void main () {
5 writeln("Hello world");
6 }
```



# My first D program

```
Code 10: hello.d
  module hello;
  import std.stdio: writeln;
3
  void main () {
   writeln("Hello world");
6
```





### Introduction

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Generics

Inference

Loops

Function

V----

Requirem

### Basics

My first D program

Arrays

String and characters

Const and Immutab

Algorithm

Structure and Clas

Miscellano

Let start

### TK D

## Code 11: Terminal

\$ ldc2 hello.d

\$ ./hello

Hello world





## Plan

### Basics

- Types





# Types

My first D progra

Arrays

Const and Immutable
Input/Output

Туре	bits	Minimum	Maximum
void	Not available	Not available	Not available
byte	8	-128	127
short	16	-32768	32767
int	32	-2147483648	2147483647
long	64	-9223372036854775808	9223372036854775807
ubyte	8	0	255
ushort	16	0	65535
uint	32	0	4294967296
ulong	64	0	18446744073709551615
float	32	1.18e <sup>-38</sup>	$3.40e^{+38}$
double	64	$2.23e^{-308}$	1.80e <sup>+308</sup>
ifloat	32	1.18e <sup>-38</sup>	$3.40e^{+38}$
idouble	64	$2.23e^{-308}$	1.80e <sup>+308</sup>
cfloat	32	1.18e <sup>-38</sup>	$3.40e^{+38}$
cdouble	64	$2.23e^{-308}$	1.80e <sup>+308</sup>
real	128	3.36e <sup>-4932</sup>	$1.19e^{+4932}$
ireal	128	$3.36e^{-4932}$	$1.19e^{+4932}$
creal	28	$3.36e^{-4932}$	$1.19e^{+4932}$
char	utf-8: 8	0	255
wchar	utf-16: 16	0	65535
dchar	utf-32: 32	0	4294967293
bool	8	false	true





## Plan

### Basics

- Arrays

D Programming









### Introductio

Meta-programming

Parallelism

Ressource Managemen

System and Safe Cod

Reference and point

Generics

Interenc

Functio

- .

Debugs

Poquirom

Editors

### Basic

My first D program

Appes

String and characters

Const and Immutable

Algorithm

Structure and Class Template

Miscellan

Let start

# Static arrays

```
1 int[3] a1 = [0, 1, 2];
2 int[3][3] a2 = [[0, 1, 2, 3], [4, 5, 6], [7, 8, 9]];
3 a1.length;
```

```
55/122
```



# Static arrays

```
\begin{array}{ll} \mathbf{int}[3] & \mathbf{a1} = [\ 0,\ 1,\ 2\ ]; \\ \mathbf{int}[3][3]\ \mathbf{a2} = [\ [\ 0,\ 1,\ 2,\ 3\ ],\ [\ 4,\ 5,\ 6\ ],\ [\ 7,\ 8,\ 9\ ]\ ]; \end{array}
```

```
al.length;
```





# Static arrays

```
int[3] a1 = [0, 1, 2];
```

```
int[3][3] a2 = [[0, 1, 2, 3], [4, 5, 6], [7, 8, 9]]; a1.length; // return array size
```



### Introductio

Meta-programming Parallelism

Ressource Management

System and Safe Code

Reference and poi

Inforona

Interent

- .

Debugs

Versions

Requirem

### Basic

My first D program

Arrays

String and characters

Const and Immutabl

Algorithm

Structure and Clas

Miscelland

Lot start

### TK D

# Dynamic arrays

```
1 int[] a1 = [ 0, 1, 2 ];
2 al.length;
3 al.length = al.length + 2;
4 al.length += 2;
```





### Introductio

Parallelism

Ressource Management

System and Safe Cod

Reference and pointe

Generics

Interenc

Loops

Functio

Debugs

Versions

Requirem

### Basic

My first D program

Arrave

String and characters

Const and Immutabl

Algorithm

Structure and Clas

Miscelland

Let start

# Dynamic arrays

```
1 \quad \text{int}[] \ a1 = [0, 1, 2];
```

- 2 al.length; // return array size
- 3 a1.length = a1.length + 2;
- 4 al.length += 2;



### Introductio

Meta-programming Parallelism

Ressource Management

System and Safe Code

Reference and points

Generics

interenc

\_\_\_\_\_

Functio

Debugs

Versions

Requirem

### Basic

My first D program

Arrays

String and characters

Const and Immutabl

Algorithm

Structure and Class Template

Miscelland

Let start

## Dynamic arrays

```
1 int[] a1 = [ 0, 1, 2 ];
2 a1.length;
3 a1.length = a1.length + 2; // resize array
4 a1.length += 2;
```

### Introductio

Meta-programming Parallelism

Ressource Management Contract

System and Safe Cod

Reference and point

Generics

Inference

Loops

Function

Debugs

Version

Requiren

Editors

### Basics

My first D program

Arrays

String and characters

Const and Immutabl

Algorithm

Structure and Cla

Template

1//

CITIZ D

# Dynamic arrays

```
1 int[] a1 = [ 0, 1, 2 ];
2 a1.length;
3 a1 length = a1 length =
```

3 al.length = al.length + 2;

4 al.length += 2; // not allowed





# Matrix arrays





### Introduction

Meta-programming

Ressource Managemen

System and Safe Code

Reference and pointe

Generics

Interenc

Loops

Functio

Debugs

versions

Requiren

### Basics

My first D program

Arrona

Arrays

String and characters

Const and Immutabl

Algorithm

Structure and Clas

Miscenano

Let start

```
1 int[] a1 = [0, 1, 2];

2 a1[0];

3 a1[0..2];

4 a1[0..$];
```



### Introductio

Meta-programming

Ressource Management

System and Safe Cod

Reference and pointer

Generics

Inference

Loops

Function

Debugs

Version

Requirer

Editors

### Basics

My first D program

Arrona

String on

Const and Immutable

Input/Output

Algorithm

Structure and Clas

Miscelland

Let start

```
1 int[] a1 = [ 0, 1, 2 ];
2 a1[0]; // return 0
3 a1[0..2];
4 a1[0..3];
```

### Introduction

Meta-programming

Ressource Managemen

System and Safe Code

Reference and points

Generic

Inferenc

Loops

Functio

Debugs

Versions

Requireme

Editors

### Basics

My first D program

Arrays

String and charact

Const and Immutable

Algorithm

Structure and Clas

Miscellano

I ot stant

## TK D

# Arrays

```
1 int[] a1 = [ 0, 1, 2 ];
2 a1[0];
3 a1[0.2]; // return [0, 1]
4 a1[0..$];
```





### Introduction

Meta-programming

Ressource Managemen

System and Safe Code

Reference and points

Generics

Inferenc

Loops

Functio

Debugs

Version

Requireme

### Basics

My first D program

A ....

Arrays

String and characters

Const and Immutabl

Algorithm

Structure and Cla

Miscellano

Lot otant

```
1 int[] a1 = [ 0, 1, 2 ];
2 a1[0];
3 a1[0..2];
4 a1[0..$]; // return [0, 1, 2]
```



## Introductio

Meta-programming

Ressource Managemen

System and Safe Code

Reference and pointe

Generics

Inferenc

Loops

Functio

Debugs

Versions

Requirement

### Basics

My first D program

Arrona

String and characte

Const and Immutable

Algorithm

Structure and Clas

Miscellano

Let start

CMIZ D

```
1 int[] a = [0, 1, 2];

2 int[] b = a; // 'b' point to 'a' (reference)

3 int* b_ptr = b.ptr;

4 int[] c = a[0..2];

5 int[] d = a[0..2].dup;
```





```
int[] a
           = [0, 1, 2];
int[] b
           = a;
int* b_ptr = b.ptr; // return pointer to given array
int[] c
           = a[0...2];
int[] d
           = a[0 ... 2].dup;
```



```
= [0, 1, 2];
int[] a
int[] b
           = a;
int^* b_ptr = b.ptr;
int[] c
           = a[0 ... 2]; // is a reference
           = a[0 ... 2].dup;
int[] d
```

```
= [0, 1, 2];
int[] a
int[] b
           = a;
int^* b_ptr = b.ptr;
int[] c
           = a[0...2];
           = a [0 ... 2].dup; // is a copy
int[] d
```

## Vectors

```
int[] a1 = new int[](2); // [0,0]
2 \text{ a1}[] = 1;
```

```
3 a1[]+=2;
```





# Vectors

```
\begin{array}{lll} 1 & \inf[] & a1 = new & \inf[](2); \\ 2 & a1[] = 1; \; // \; [1,1] \\ 3 & a1[] += 2; \end{array}
```

$$3 \text{ a1}[]+= 2;$$





### Introduction

Meta-programming

Ressource Management

System and Safe Cod

Reference and

Y C

Interent

\_

Functio

Debugs

Versions

Requirem

My first D program

Types

Arrays

String and characters

Const and Immutabl

Algorithm

Structure and Cla

Template

. .

## GTK D

## Vectors

```
1 int[] al = new int[](2);
2 al[] = 1;
3 al[]+= 2; // [3,3]
```





### Introductio

Meta-programming

Ressource Managemen

System and Safe Cod

Reference and points

Generic

Interenc

Loops

Functio

Debugs

Versions

Requiremen

### Basics

My first D program

Appes

String and ch

Const and Immutable

Input/Out

Algorithm

Structure and Clas

Miscellano

I of stant

### TK D

# Associative arrays

- 1 string[int] dict; ◀
- 2 dict["D"] = 1;
- 3 "D" in dict;





### Introductio

Meta-programming

Ressource Managemen

System and Safe Cod

Reference and pointe

Generic

Interenc

Functions

Debugs

Versions

Requirement

### Basics

My first D program

Arrave

String and

Const and Immutable

Input/Output

Algorithm

Structure and Cla

Miscelland

I ot otost

ם אדי

# Associative arrays

- 1 string[int] dict;
- $2 \operatorname{dict}["D"] = 1;$
- 3 "D" in dict;

75 /199

D Programming





### Introductio

Meta-programming

Ressource Management

System and Safe Cod

Reference and pointe

Generic

Inferenc

Loops

Functio

Debugs

Requireme

Editors

### Basic

My first D program

Appes

 $_{
m Arrays}$ 

String and characters

Const and Immutabl

Algorithm

Structure and Cla

Miscellano

Let start

### TK D

# Associative arrays

- 1 string[int] dict;
- 2 dict["D"] = 1; 3 "D" in dict; // true
- 3 "D" in dict; // true





# Plan

# Basics

- String and characters





# Introduction

Meta-programming
Parallelism

Ressource Management

System and Safe Cod

Reference and pointe Generics

Informa

-

Para anti-

Function

Debugs

Version

Requireme

### Basic

My first D program

Arravs

### String and characters

Const and Immutable

Algorithm

Structure and Clas

Template

. . .

CONT. D

# String and characters

```
1 string words = "With double quote, \alpha\beta\gamma\delta\epsilon"; // UTF-8
2 string words2= words ~ ", and concatenation"; // concat
3 char letter= 'a'; // simple quote
```



# Plan

## Basics

- Const and Immutable







# Const and Immutable

# Const and Immutable

- im-mutable data: that can-not change.
- const data: can-not be changed by the current const ref-er-ence to that data.







# Plan

### Basics

### • Input/Output

### D Programming





## Introduction

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Reference and pointer

Informac

Interenc

Loops

Functio

Debugs

versions

Requirem

Editors

### Basic

My first D program

Arrays

String and characters

Const and Immutabl

Algorithm

Structure and Class Template

Miscellano

I of stant

# TK D

# Input/Output

# Code 12: Read a file

```
1 import std.stdio: open, writeln;
2 ...
3 File f = open( "/path/to/myFile", "r" );
```

4 scope(exit) f.close;

5 foreach ( number, line; f )
6 writeln ( number, line );

7 ...





### Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Generics

Inference

Interenc

Loops

Function

Debugs

Requireme

Editors

### Basics

My first D program

Arrays

String and characters

Const and Immutabl

Algorithm

Structure and Clas Template

Miscelland

Let start

## TK D

# Input/Output

# Code 13: Write a file

```
1 import std.stdio: open, writeln;
2 ....
3 File f = open( "/path/to/myFile", "w" );
4 scope(exit) f.close;
5 writeln( "something" );
6 ...
```



### Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Reference and pointer

Inference

Interenc

Loops

Functio

Debugs

versions -

Requireme

### Editors

My first D program

Types

String and characters

Const and Immutabl
Input/Output

Algorithm

Structure and Clas Template

Miscellano

1/4

# TK D

# Input/Output

# Code 14: Capture keyboard

```
1 import std.stdio: open, writeln;
2 ...
3 char[] name;
4 size_t age;
5 write( "Enter your name : " );
6 readf( "%s" ~ newline, &name );
7 write( "How old are you : " );
8 readf( "%u" ~ newline, &age );
9 ...
```





# Plan

# Basics

- Algorithm





# Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Reference and pointer

Inference

Interenc

Loops

Functio

Debugs

Mondon

.

Editors

### Basics

My first D program

Arravs

String and characters

Const and Immutabl Input/Output

### Algorithm

Structure and Class Template

Let start it

# Code 15: Searching

- 2 ...
- 3 int[] list = [ 0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9 ];
- 4 list.count(1);
- 5 list.countUntil(2);
- 6 list.startsWith(0);
- 7 list.endsWith(9);
- 8 list.canFind(2);





### Introductio

Meta-programming Parallelism Ressource Management

System and Safe Code Reference and pointer

Generics

Inference

Function

Debugs

Requireme

Editors

### Basics

My first D program

Types Arrays

String and characters Const and Immutable

Input/Output

### Algorithm

Structure and Class Template Miscellanous

Let start i

# Code 16: Searching

1 import std.algorihm: count, countUntil, startsWith, endsWith, canFind;

```
2 \dots 3 \text{ int}[] \text{ list} = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9];
```

3 int[] list = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9]; 4 list.count(1); // result => 2

5 list.countUntil(2);

6 list.startsWith(0);

7 list.endsWith(9);

8 list.canFind(2);





# Introductio

Meta-programming Parallelism Ressource Management

System and Safe Code

Reference and point Generics

Inference

Loops

Functio

Dobuga

Debugs

Requirem

Editors

### Basics

My first D program

Types

String and characters

Const and Immutabl Input/Output

### Algorithm

Structure and Class Template Miscellanous

Let start it:

# Code 17: Searching

- 1 import std.algorihm: count, countUntil, startsWith, endsWith, canFind:
- 2 ...
- $3 \quad int[] \quad list = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9];$
- 4 list.count(1);
- 5 list.countUntil(2); // result => 3
  - 6 list.startsWith(0);
  - 7 list.endsWith(9);
- 8 list.canFind(2);



# Code 18: Searching

- import std.algorihm: count, countUntil, startsWith, endsWith, canFind:
- int[] list = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9];
- list.count(1);
- list.countUntil(2);
- list.startsWith(0); // result => true
- list.endsWith(9):
- list.canFind(2):



# Code 19: Searching

import std.algorihm: count, countUntil, startsWith, endsWith, canFind:

- int[] list = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9];
- list.count(1); list.countUntil(2);
- list.startsWith(0);
- list.endsWith(9); // result => true
- list.canFind(2);





# Algorithm

# Code 20: Searching

- import std.algorihm: count, countUntil, startsWith, endsWith, canFind:
- int[] list = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9];
- list.count(1);
- list.countUntil(2);
- list.startsWith(0);
- list.endsWith(9):
- list.canFind(2); // result => true







# Introductio

Meta-programming Parallelism

Ressource Management Contract

System and Safe Code Reference and pointer

Informa

Interence

Loops

Functio

Debugs

Versions

Requireme

Editors

### Basic

My first D program

Arrave

String and characters

Const and Immutabl

### Algorithm

Structure and Class Template

Let start

Let start

# Algorithm

# Code 21: Comparison

```
l import std.algorihm: min, max;
```

 $3 \min(9, 12); // \text{ result} => 9$ 

4 max( 9, 12); // result => 12



### Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Cod

Reference and pointe

T. f.....

Interenc

Loops

Functio

Debugs

versions

Requirem

Editors

### Basic

My first D program

Arrave

String and characters

Const and Immutabl Input/Output

### Algorithm

Structure and Class Template

Let start

# Code 22: Iteration

- 1 import std.algorihm: filter, uniq, map, reduce; 2 int[] list = [ 0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9 ];
- 3 list.filter!( a => a > 6):
  - 1 list.uniq();
  - 5 list.map!( a => a + 2);
  - 6 0.reduce!((a,b) => a + b)(list);
- 7 list.reduce!( min, max )( );





# Algorithm

# Code 23: Iteration

- import std.algorihm: filter, uniq, map, reduce; int[] list = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9];
- list.filter!( a => a > 6 ); // result => [ 7, 8, 9 ]
  - list.uniq();
  - list.map!(  $a \Rightarrow a + 2$ );
  - 0.reduce!( (a,b) => a + b )( list );
- list.reduce!( min, max )( );





# Code 24: Iteration

- import std.algorihm: filter, uniq, map, reduce; int[] list = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9];
- - list.filter!( a => a > 6);
  - list.uniq(); // [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
- list.map!( a => a + 2);
- 0.reduce!( (a,b) => a + b )( list );
- list.reduce!( min, max )( );





import std.algorihm: filter, uniq, map, reduce; int[] list = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9];

list filter! (a => a > 6):

list.uniq();

list.map!( a => a + 2 ); // [ 2, 3, 3, 4, 5, 6, 7, 8, 9, 10, 11 ]

Code 25: Iteration

0.reduce!( (a,b) => a + b )( list );

list.reduce!( min, max )( );





# Introductio

Parallelism

Contract
System and Safe Code

Reference and pointe

Generics

Inferenc

. .

пооры

Functio

Debugs

Versions

Requirem

Editors

### Basics

My first D program

Arrave

String and characters

Const and Immutab Input/Output

### Algorithm

Structure and Class Template

Let start

CITIE D

# Code 26: Iteration

- 1 import std.algorihm: filter, uniq, map, reduce;
- 2 int[] list = [0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9];
- 3 list.filter!( a => a > 6);
  - 1 list.uniq();
- 5 list.map!( a => a + 2);
- 6 0.reduce!( (a,b) => a + b )( list ); // sum all elements => 49
- 7 list.reduce!( min, max )( );





# Code 27: Iteration

- import std.algorihm: filter, uniq, map, reduce;
- int[] list = [ 0, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9 ];
- list.filter!( a => a > 6);
- list.uniq();
- list.map!( a => a + 2);
- 0.reduce!( (a,b) => a + b )( list );
- list.reduce!( min, max )( ); // compute in one pass min an max
- // min: 0 max: 9





# Introduction

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Reference and pointe

Inference

Loops

Function

Versions

Requireme

Editors

### Basics

My first D program

Arrays

String and characters Const and Immutable

Input/Outp

Algorithm

Structure and Class Template

T -4 -4--4

# Algorithm

# Code 28: Sorting



Structure and Class

# Plan

## Basics

- a Structure and Class





Structure and Class

# Structure

# Code 29: Classic implementation

```
struct MyStruct{
          field1:
    int
3
```

float fiel2;

string field3;

5





# Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Reference and pointer

Generics Inference

Loops

Function

Debugs

Requirem

Editors

### Basic

My first D program

Types Arrays

String and characters

Input/Output

Algorithm

Structure and Class

Femplate Miscellanou

Lot start

# TK D

# Structure

# Code 30: With constructor and modifier

```
struct MyStruct{
     private:
 3
           field1:
     int
     float fiel2:
     string field3;
     public:
     this(int f1, float f2, string f3){
 9
       field1 = f1;
10
       field2 = f2:
11
        field3 = f3:
12
13
```





Structure and Class

# Code 31: With method and property

```
struct MvStruct{
     private:
           field1:
 3
     int
     float field2:
     string field3;
     public:
     this(int f1, float f2, string f3){
 8
       field1 = f1;
 9
        field2 = f2;
        field3 = f3:
10
11
12
     this( MyStruct s ){
13
       field1 = s.field1:
14
        field2 = s.field2:
15
        field3 = s.field3;
16
17
     @property int field1( ) const { return field1: }
     @property void field1( int f1 ){ _field1 = f1; }
18
19
     @property float field2() const { return field2; }
     @property void field2( float f2 ) { field2 = f2; }
20
21
     @property string field3( ) const { return _field3; }
22
     @property void field3( string f3 ) { field3 = f3; }
23
     MvStruct dup() const{ return MvStruct( this ); }
24
```



Structure and Class

# Code 32: Polymorphism

```
import std.string: format;
   Interface IPersonnage{
     string name():
     int health();
 6
     int mana();
 7
 8
   class BowMan: IPersonnage{
10
     private:
11
     string _name;
12
     int health;
13
     int mana:
14
     public:
15
     this( string n, int h, int m ){
16
       name = n:
17
       health = h:
18
        mana = m;
19
20
     string name() const { return _name; }
21
     int health() const { return health; }
22
     int mana() const { return mana; }
23
     override string toString() const{
24
      return "name: %s point: %d mana: %d".format(_name, _health,
              mana);
25
26
```





# Plan

### Basics

### • Template

D Programming





# Introductio

Parallelism

Ressource Management Contract

System and Safe Cod

Generics

Inference

T -----

Functio

Debugs

Version

Requirem

### Editors

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Class

 $\Gamma$ emplate

Let start

# Template

# Code 33: Function template

```
1 auto addition(T,U)( T a, U b){
2 return a + b;
```

3 }



Meta-programming
Parallelism

Contract System and Safe Code

Reference and pointer

Generics

Inference

Loops

Functio

Debugs

D -----

Editors

Editors

### Basic

My first D program

Arrays

String and characters
Const and Immutable

Input/Output

Algorithm

Structure and Clas

l'emplate

Let start

# amir D

# Template

# Code 34: Struct template

```
1 struct TStruct(T) {
2    private:
3    T _f1;
4    public:
5    this ( T f1 ) {
6    _f1 = f1;
7    }
8    }
9    void main() {
10    TStruct!string t1 = TStruct!(string)( "test");
11    auto    t2 = TStruct!int( 5 );
12    }
```





# Introduction

Meta-programming
Parallelism

Contract
System and Safe Code

Reference and pointe

Generics

Inference

Loops

Function

Debugs

Requirem

Requiren

### Basics

My first D program

Arrays

String and characters

Const and Immutable

Algorithm

Structure and Clas

 $\Gamma$ emplate

Let start

# Template

# Code 35: Class template

```
1 class TClass(T) {
2    private:
3    T_fl;
4    public:
5    this ( T f1 ) {
6    _f1 = f1;
7    }
8  }
9    auto c1 = new TClass!int( 2 );
```





# Plan

# Basics

### Miscellanous

# D Programming



Miscellanous

# Miscellanous

- 1 int  $\beta$  = 5; // variable name can use UTF-8 char 2 int i = 1\_000\_000; // easy to read number



# Plan

## Basics

- Let start it!

# D Programming







# Introduction

Meta-programming Parallelism

Ressource Management Contract

System and Safe Code

Generics

Inference

-

Functions

r unctions

Debugs

Versions

Requirement Editors

### Basics

My first D program

Types

String and characters

Const and Immutable Input/Output

Algorithm

Structure and Clas

Miscellano

Let start

### TK D

# What Are You Waiting For?

• Web site: dlang.org



## Introductio

Meta-programming
Parallelism

Contract
System and Safe Code

Reference and pointe

Generics

Inferenc

Loons

Functio

Debugs

Versions

Requirem

Editors

### Basics

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Class

Miscellano

Let start

## TK D

# What Are You Waiting For?

- Web site: dlang.org
- Community: forum.dlang.org





## Introductio

Meta-programming Parallelism

Contract

System and Safe Code

Generics

Inference

Loops

Function

Versions

Requirem

Editors

## Basics

My first D program

Types

String and characters

Const and Immutabl

Algorithm

Structure and Class

Miscelland

Let start

## TK D

# What Are You Waiting For?

- Web site: dlang.org
- Community: forum.dlang.org
- Contribute:

www.github.com/D-Programming-Language





## Introductio

Meta-programming Parallelism

Contract

System and Safe Cod

Generics

Inference

Loope

Functio

Debugs

Versions

Editors

## Basics

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Cla

Miscellanc

Let start

## TK D

# What Are You Waiting For?

- Web site: dlang.org
- Community: forum.dlang.org
- Contribute:

www.github.com/D-Programming-Language

• irc on freenode #d







## Introduction

Meta-programming
Parallelism

Contract System and Safe Code

Reference and pointer

Inference

Loops

Function Debugs

Versions

Requireme Editors

## Basic

My first D program

Arrays

String and characters

Const and Immutable

Input/Output

Algorithm

Structure and Cla

Miscellan

Let start

## TK D

# What Are You Waiting For?

- Web site: dlang.org
- Community: forum.dlang.org
- Contribute:

www.github.com/D-Programming-Language

- irc on freenode #d
- french speaker on jabber d-fr@chat.jabberfr.org





## Code 36: First graphical application

```
module myFirstGUI;
   import gtk.MainWindow;
   import gtk.Label;
   import gtk.Main;
 6
   class mvFirstGUI: MainWindow{
     this(){
 9
      super("GtkD");
      setBorderWidth(10):
10
11
      add(new Label("Hello World"));
12
      showAll();
13
14
15
16
   void main(string[] args){
17
     Main.init(args);
18
     new HelloWorld();
19
     Main.run();
20
21
```



## Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Code

Reference and pointe

Generics

Interenc

Function

Debugs

17----

Requirem

Editors

### Basic

My first D program

Types

String and characters

Const and Immutabl

Algorithm

Structure and Cla

Miscellano

GTK D

Functional

Code 37: Terminal

\$ ldc2 -L-lgtkd -L-ldl myFirstGUI.d

\$ ./myFistGUI

GtkD ×

Hello World





## Introductio

Meta-programming
Parallelism
Ressource Management

System and Safe Code
Reference and pointer

Reference and pointer Generics

Inference

Functions

Debugs

Requireme

Editors

## Basic

My first D program

Arrays

String and characters Const and Immutable

Algorithm

Structure and Class Template

Femplate Miscellanous

## GTK D

# Code 38: First graphical application

```
module myFirstGUI;
 2
   import gtk.MainWindow;
   import gtk.Label;
   import gtk. Main;
 6
   pragma(lib, "gtkd"); "
   class myFirstGUI: MainWindow
10
     this(){
11
      super("GtkD");
12
      setBorderWidth(10);
13
      add(new Label("Hello World"));
14
      showAll();
15
16
17
18
   void main(string[] args){
     Main.init(args);
19
     new myFirstGUI();
20
21
     Main.run();
22
23
```



## Code 39: First graphical application

```
module myFirstGUI;
 2
   import gtk.MainWindow;
   import gtk.Label;
   import gtk. Main;
 6
   pragma(lib, "gtkd");
   class mvFirstGUI: MainWindow{
10
     this(){
11
      super("GtkD");
12
      setBorderWidth(10);
13
      add(new Label("Hello World"));
14
      showAll();
15
16
17
18
   void main(string[] args){
     Main.init(args);
19
     new myFirstGUI();
20
21
     Main.run();
22
23
```





## Code 40: First graphical application

```
module myFirstGUI;
 2
   import gtk.MainWindow;
   import gtk.Label;
   import gtk. Main;
 6
   pragma(lib, "gtkd");
   class mvFirstGUI: MainWindow{
10
     this(){
11
      super("GtkD");
12
      setBorderWidth(10);
13
      add(new Label("Hello World"));
14
      showAll();
15
16
17
18
   void main(string[] args){
     Main.init(args);
19
20
     new myFirstGUI();
21
     Main.run();
22
23
```







## Introductio

Meta-programming
Parallelism

Contract System and Safe Code

Reference and pointer

Inference

Interend

E----

Debuge

Versions

Requirem

Editors

### Basic

My first D program

Arravs

String and characters

Const and Immutable

Algorithm

Structure and Clas

emplate Iiscellanou

GTK D

# First graphical Application

Code 41: Terminal

\$ ldc2 -L-ldl myFirstGUI.d

\$ ./myFistGUI





GTK D

## Code 42: First graphical application

```
module mvFirstGUI:
   import gtk.MainWindow;
   import gtk.Label;
   import gtk.Main;
   pragma(lib, "gtkd");
   version(Linux) pragma(lib, "dl");
 9
   class myFirstGUI: MainWindow{
10
11
     this(){
12
      super("GtkD");
13
      setBorderWidth(10);
14
      add(new Label("Hello World")):
15
      showAll():
16
17
18
19
   void main(string[] args){
20
     Main.init(args);
21
     new myFirstGUI();
22
     Main.run();
23
24
```



## Introduction

Meta-programming
Parallelism

System and Safe Code

Reference and pointer Generics

Inferenc

-

Functio

Dobugo

Debugs

Requirem

Editors

## Basic

My first D program

Types

String and characters

Const and Immutable

Algorithm

Structure and Clas

Template Miscellanou

GTK D

# First graphical Application

Code 43: Terminal

\$ ldc2 myFirstGUI.d

\$ ./myFistGUI



## Introductio

Parallelism

Contract

Reference and pointer

Generics

Inferenc

Loops

Function

Debugs

versions

Requirem

Editors

### Basic

My first D program

Types

String and character

Const and Immutab

Algorithm

Structure and Cla

Miscelland

Let start

CITIZ D

## Contact

Comments, suggestions or bug reports?

Please send a mail at:

bioinfornatics@fedoraproject.org





## Introductio

Meta-programming Parallelism

Ressource Management Contract

System and Safe Code Reference and pointer

Generics

Inferenc

. .

Loops

Functio

Debugs

Versions

Requirem

## Basic

My first D program

Types

String and characters

Const and Immutable Input/Output

Algorithm

Structure and Cla

Miscellano

Let start

## Thanks to

- To your attention
- French fedora community
- D community
- Mohamed El Morabity







122/122

D Programming

