無料電子ブック

学習

D Language

Free unaffiliated eBook created from
Stack Overflow contributors.

#d

1: Dのい

DはCのようなとけをつシステムプログラミングです。、、モデリングととプログラマのをみわせています。

バージョン

バージョン	ログ	
D	http://www.digitalmars.com/d/1.0/changelog.html	2007123
D2	https://dlang.org/changelog/2.000.html	20070617

Examples

インストールまたはセットアップ

Dプログラミングのコンパイラ DMDは、すべてのなプラットフォームでできます。 DMDをインストールするにはこちらをください 。コマンドラインでインストールするには、コマンドをしてくださいDウェブサイトにあります

```
curl -fsS https://dlang.org/install.sh | bash -s dmd
```

パッケージマネージャー

アーチLinux

```
pacman -S dlang
```

チョコレート

```
choco install dmd
```

Gentoo

```
layman -f -a dlang
```

OSXホームブリュー

```
brew install dmd
```

Debian / Ubuntu

Debian / Ubuntuディストリビューションにインストールするには、 APTリポジトリをソースリストにするがあります 。

```
wget http://master.dl.sourceforge.net/project/d-apt/files/d-apt.list -O
/etc/apt/sources.list.d/d-apt.list
wget -qO - https://dlang.org/d-keyring.gpg | sudo apt-key add -
apt-get update
apt-get install dmd-bin
```

そのコンパイラ

LDCは、DMDコンパイラフロントエンドとLLVMをバックエンドとしてするDコンパイラです。

GDCは、GCCバックエンドをしてコードをするDコンパイラです。

IDE

をにするために、IDEをインストールすることもできます。 D-Language Wikiには、 ここにあるすべてのプラットフォームのなIDEとプラグインのリストがあります 。

こんにちは

```
import std.stdio;

// Let's get going!
void main()
{
    writeln("Hello World!");
}
```

コンパイルしてするには、このテキストを`main.d`という`main.d`ファイルとしてします。コマンドラインから`dmd main.d`をして、プログラムをコンパイルします。に、`./main`をしてbashシェルでプログラムをするか、Windowsのファイルをクリックします。

こんにちは

な "Hello、world"プログラムをするには、のコードをむテキストエディタで`hello.d`というファイルをします。

```
import std.stdio;
```

```
void main() {
    writeln("Hello, World!");    //writeln() automatically adds a newline (\n) to the output
}
```

```
import std.stdio
```

このは、ライブラリモジュール`std.stdio`されたがされることをコンパイラにえます。コンパイラがどこをすべきかをっているり、どのモジュールもインポートすることができます。 Dのなライブラリのとして、くのがされています。

```
void main() {
```

このは`main`し、 `void`し`void`。 CおよびC ++とはなり、Dではmainが`void`であることがです。 `main`は、プログラムのエントリーポイントであるため、なものです。つまり、プログラムのがされます。なにするいくつかの

- のは、でまり、、、およびでされたものであればどれでもかまいません。
- されるパラメータは、コンマでられたとそのデータのリストです。
- がすとされるは、のデータであるがあり、のreturnステートメントでされるのとしなければなりません。

`{ … }`はペアでされ、コードブロックのとのをします。それらはくのですることができますが、このはのとをします。

```
writeln("Hello, World!");
```

`writeln`は、 `std.stdio`でされているを`stdout`に`std.stdio`です。この、は`"Hello, World"`で、コンソールにきまれます。 `\n` 、 `\r`など、C の`printf`されているものにた、さまざまなのをできます。

すべてのステートメントは、セミコロンでするがあります。

コメントは、コードをむにかをすためにされ、コンパイラによってとしてわれます。のコードでは、これはコメントです

```
//writeln() automatically adds a newline (\n) to the output
```

これらは、コンパイラによってされるコードのです。 Dでコメントするは3つあります。

1. `//` - すべてのテキストをじの`//`ろにコメントする
2. `/* comment text */` - これはのコメントにです
3. `/+ comment text +` - のコメント

それらは、ある/コードがのにをしているのかをえるのににです。

プログラムのコンパイルと

このプログラムをするには、コードをファイルにコンパイルするがあります。これは、コンパイラのけをりてうことができます。

DコンパイラをしてDMDをしてコンパイルするには、をき、したファイル`hello.d`にしてからします。

```
dmd hello.d
```

エラーがつからない、コンパイラはソースファイルのをけたファイルをします。これで、のようにすることができます。

```
./hello
```

すると、プログラムは`Hello, World!`をし`Hello, World!`そのにがきます。

からをみむ

```
import std.format;

void main() {
    string s = "Name Surname 18";
    string name, surname;
    int age;
    formattedRead(s, "%s %s %s", &name, &surname, &age);
    // %s selects a format based on the corresponding argument's type
}
```

フォーマットのドキュメントは、 https : //dlang.org/phobos/std_format.html#std.formatにあります。

オンラインでDのいをむ https://riptutorial.com/ja/d/topic/1036/dのい

2: UFCS - びしの

- aThirdFunanotherFunmyFun、42; //
- myFun。anotherFun42.aThirdFun; // UFCS
- myFun.anotherFun42.aThirdFun; //のはできます

コールでは`ab(args...)`タイプの、`a`というのメソッドはありません`b`、コンパイラはとしてコールをきえしようとする`b(a, args...)`

Examples

がプライムであるかどうかをチェックする

```
import std.stdio;

bool isPrime(int number) {
    foreach(i; 2..number) {
        if (number % i == 0) {
            return false;
        }
    }

    return true;
}

void main() {
    writeln(2.isPrime);
    writeln(3.isPrime);
    writeln(4.isPrime);
    5.isPrime.writeln;
}
```

をつUFCS

```
void main() {
    import std.algorithm : group;
    import std.range;
    [1, 2].chain([3, 4]).retro; // [4, 3, 2, 1]
    [1, 1, 2, 2, 2].group.dropOne.front; //  tuple(2, 3u)
}
```

std.datetimeからのをつUFCS

```
import core.thread, std.stdio, std.datetime;

void some_operation() {
    // Sleep for two sixtieths (2/60) of a second.
    Thread.sleep(2.seconds / 60);
    // Sleep for 100 microseconds.
    Thread.sleep(100.usecs);
```

```
}

void main() {
    MonoTime t0 = MonoTime.currTime();
    some_operation();
    MonoTime t1 = MonoTime.currTime();
    Duration time_taken = t1 - t0;

    writeln("You can do some_operation() this many times per second: ",
            1.seconds / time_taken);
}
```

オンラインでUFCS - びしのをむ https://riptutorial.com/ja/d/topic/4155/ufcs----びしの

3: クラス

- クラスFoo {} //はObjectからします
- class BarFoo {} // BarはFooです。
- Foo f =しいFoo; //ヒープにしいオブジェクトをインスタンスする

をし、クラス 、 にするのをし、 でします。

Examples

```
class Animal
{
    abstract int maxSize(); // must be implemented by sub-class
    final float maxSizeInMeters() // can't be overridden by base class
    {
        return maxSize() / 100.0;
    }
}

class Lion: Animal
{
    override int maxSize() { return 350; }
}

void main()
{
    import std.stdio : writeln;
    auto l = new Lion();
    assert(l.maxSizeInMeters() == 3.5);

    writeln(l.maxSizeInMeters()); // 3.5
}
```

インスタンス

```
class Lion
{
    private double weight; // only accessible with-in class

    this(double weight)
    {
        this.weight = weight;
    }

    double weightInPounds() const @property // const guarantees no modifications
    // @property functions are treated as fields
    {
        return weight * 2.204;
    }
}

void main()
{
```

```
    import std.stdio : writeln;
    auto l = new Lion(100);
    assert(l.weightInPounds == 220.4);

    writeln(l.weightInPounds); // 220.4
}
```

オンラインでクラスをむ https://riptutorial.com/ja/d/topic/4695/クラス

4: コンパイルCTFE

CTFEは、コンパイラがコンパイルにをできるようにするメカニズムです。このをするためになDのなセットはありません。コンパイルにDコンパイラがすることをしたのをがコンパイルにしているはにそうです。

また、CTFEとインタラクティブにプレイすることもできます。

Examples

コンパイルにをする

```
long fib(long n)
{
    return n < 2 ? n : fib(n - 1) + fib(n - 2);
}

struct FibStruct(int n) { // Remarks: n is a template
    ubyte[fib(n)] data;
}

void main()
{
    import std.stdio : writeln;
    enum f10 = fib(10); // execute the function at compile-time
    pragma(msg, f10); // will print 55 during compile-time
    writeln(f10); // print 55 during runtime
    pragma(msg, FibStruct!11.sizeof); // The size of the struct is 89
}
```

オンラインでコンパイルCTFEをむ https://riptutorial.com/ja/d/topic/4694/コンパイル-ctfe-

5: スコープガード

- scopeexit - のブロックがどのようにしてもステートメントはびされます
- スコープ - のブロックがにしたときにステートメントがびされます。
- scopefailure - のブロックがスローされてしたときにステートメントがびされます。

スコープガードをすると、コードがはるかにクリーンになり、リソースのりてやコードのびえがになります。これらのさなヘルパーは、のクリーンアップコードがににどのパスからられるかをにしてするため、もさせます。

Dスコープは、C ++でされているRAIIイディオムをにきえます。これは、リソースのなスコープガードオブジェクトにつながることがよくあります。

スコープガードは、されたのでびされます。

スコープガードをってプレイするか、なチュートリアルをしてください 。

Examples

りてとクリーンアップコードをいににする

スコープガードは、のブロックがっているにのでステートメントをできるようにします。

```
import core.stdc.stdlib;

void main() {
    int* p = cast(int*)malloc(int.sizeof);
    scope(exit) free(p);
}
```

のネストされたスコープ

```
import std.stdio;

void main() {
    writeln("<html>");
    scope(exit) writeln("</html>");
    {
        writeln("\t<head>");
        scope(exit) writeln("\t</head>");
        "\t\t<title>%s</title>".writefln("Hello");
    } // the scope(exit) on the previous line is executed here

    writeln("\t<body>");
    scope(exit) writeln("\t</body>");

    writeln("\t\t<h1>Hello World!</h1>");
}
```

プリント

```
<html>
    <head>
        <title>Hello</title>
    </head>
    <body>
        <h1>Hello World!</h1>
    </body>
</html>
```

オンラインでスコープガードをむ https://riptutorial.com/ja/d/topic/4343/スコープガード

6: ダイナミックアレイスライス

- <タイプ> [] <>;

スライスはのメモリにしいビューをします。らはしいコピーをしません。スライスにそのメモリへのまたはスライスされたがされていないは、ガベージコレクタによってされます。

スライスをすると、1つのメモリブロックでし、にするのあるのみをスライスするパーサーなど、になコードをくことができます。しいメモリブロックをりてるはありません。

Examples

と

```
import std.stdio;

void main() {
    int[] arr = [1, 2, 3, 4];

    writeln(arr.length); // 4
    writeln(arr[2]); // 3

    // type inference still works
    auto arr2 = [1, 2, 3, 4];
    writeln(typeof(arr2).stringof); // int[]
}
```

```
import std.stdio;

void main() {
    int[] arr = [1, 2, 3];

    // concatenate
    arr ~= 4;
    writeln(arr); // [1, 2, 3, 4]

    // per element operations
    arr[] += 10
    writeln(arr); // [11, 12, 13, 14]
}
```

スライス

```
import std.stdio;

void main() {
    int[] arr = [1, 2, 3, 4, 5];

    auto arr2 = arr[1..$ - 1]; // .. is the slice syntax, $ represents the length of the array
    writeln(arr2); // [2, 3, 4]
```

```
    arr2[0] = 42;
    writeln(arr[1]); // 42
}
```

オンラインでダイナミックアレイスライスをむ https://riptutorial.com/ja/d/topic/2445/ダイナミックアレイ-スライス

7: テンプレート

- テンプレートTemplateParameterList{...}
- TemplateParameterList{...}
- クラスTemplateParameterList{...}
- りのTemplateParameterListParameterList{...}
- TemplateInvocationList

Examples

1つのテンプレートによる

```
import std.stdio;

T min(T)(in T arg1, in T arg2) {
    return arg1 < arg2 ? arg1 : arg2;
}

void main() {
    //Automatic type inference
    writeln(min(1, 2));

    //Explicit type
    writeln(min!(ubyte)(1, 2));

    //With single type, the parenthesis might be ommited
    writeln(min!ubyte(1, 2));
}
```

テンプレート

テンプレートがでることができる`template`。やクラス、そののをめることができます。

```
template StaticArray(Type, size_t Length) {
     class StaticArray {
          Type content[Length];

          size_t myLength() {
            return getLength(this);
          }
     }

     private size_t getLength(StaticArray arr) {
          return Length;
     }
}

void main() {
    StaticArray!(int, 5) arr5 = new StaticArray!(int, 5);
    import std.stdio;
    writeln(arr5.myLength());
}
```

オンラインでテンプレートをむ https://riptutorial.com/ja/d/topic/3892/テンプレート

8: メモリとポインタ

- <variable> - によるアクセス=のデータへのポインタを
- * <variable> - デフレクション=ポインタからデータオブジェクトをする
- <type> * - <type>をすデータえば `int *

Examples

ポインタ

Dはシステムプログラミングなので、でしてメモリをしにすることができます。それにもかかわらず、Dはデフォルトでガベージコレクタをしてメモリをします。

DはCのようなポインタT *をします

```
void main()
{
    int a;
    int* b = &a; // b contains address of a
    auto c = &a; // c is int* and contains address of a

    import std.stdio : writeln;
    writeln("a ", a);
    writeln("b ", b);
    writeln("c ", c);
}
```

ヒープにりてる

ヒープのしいメモリブロックは、メモリへのポインタをす`new`をしてりてられます。

```
void main()
{
    int* a = new int;
    *a = 42; // dereferencing
    import std.stdio : writeln;
    writeln("a: ", *a);
}
```

@セーフD

aによってされるメモリがプログラムのによってされなくなると、ガベージコレクタはメモリをします。

Dは`@safe`とマークされたコードをいて、ポインタも`@safe`。

```
void safeFun() @safe
{
```

```
    writeln("Hello World");
    // allocating memory with the GC is safe too
    int* p = new int;
}

void unsafeFun()
{
    int* p = new int;
    int* fiddling = p + 5;
}

void main()
{
    safeFun();
    unsafeFun();
}
```

SafeDのについては、Dデザインチームのをしてください。

オンラインでメモリとポインタをむ https://riptutorial.com/ja/d/topic/6374/メモリとポインタ

9: ユニットテスト

- unittest {...} - "unittesting"モードでのみされるブロック
- assert<ブールにされる>、<オプションのエラーメッセージ>

Examples

ユニットテストブロック

テストは、バグのないしたアプリケーションをするれたです。インタラクティブなドキュメンテーションとしてし、をすれなしにコードをすることができます。 Dは、Dのとして`unittest`ブロックのでネイティブなをします。 Dモジュールのどこでも`unittest`ブロックをして、ソースコードのをテストすることができます。

```
/**
Yields the sign of a number.
Params:
    n = number which should be used to check the sign
Returns:
    1 for positive n, -1 for negative and 0 for 0.
*/
T sgn(T)(T n)
{
    if (n == 0)
        return 0;
    return (n > 0) ? 1 : -1;
}

// this block will only be executed with -unittest
// it will be removed from the executable otherwise
unittest
{
    // go ahead and make assumptions about your function
    assert(sgn(10)  ==  1);
    assert(sgn(1)   ==  1);
    assert(sgn(-1)  == -1);
    assert(sgn(-10) == -1);
}
```

unittestの

`-unittest`フラグをDコンパイラにすと、すべてのunittestブロックがされます。くの、スタブされた`main`をコンパイラにさせるのがです。コンパイルラッパー`rdmd`をして、Dプログラムをテストするのはです

```
rdmd -main -unittest yourcode.d
```

もちろん、このプロセスを2つのステップにすることもできます。

```
dmd -main -unittest yourcode.d
./yourcode
```

dubプロジェクトでは、すべてのファイルをコンパイルしてunittestブロックをするとです。

```
dub test
```

プロのヒントチッピングをするためにシェルエイリアスとして `tdmd`をしてください。

```
alias tdmd="rdmd -main -unittest"
```

あなたのファイルをテストするには

```
tdmd yourcode.d
```

きユニットテスト

テンプレートされたコードでは、の `@nogc`がしくされているかどうかをすることがです。のテストでこれをにするために、 `@nogc`をアノテートすることができます

```
@safe @nogc pure nothrow unittest
{
    import std.math;
    assert(exp(0)  ==  1);
    assert(log(1)  ==  0);
}
```

もちろん、Dのブロックにはをすることができ、コンパイラはそれらがしいことをします。たとえば、のはのにています

```
unittest
{
    import std.math;
    @safe {
        assert(exp(0)  ==  1);
        assert(log(1)  ==  0);
    }
}
```

オンラインでユニットテストをむ https://riptutorial.com/ja/d/topic/6201/ユニットテスト

10: ループ

- for<initializer>; <ループ>; <ループ>{<statements>}
- while<condition>{<statements>}
- do {<ステートメント>} while<>;
- foreach<el>、<collection>
- foreach_reverse<el>、<collection>

- `for`ループin `for` *D* 、
- `while`ループプログラミングの*D* 、
- プログラミング `do while` while `do while` 、
- `foreach` in *Programming in D* 、 opApply 、

あなたはループとforeachオンラインでふことができます。

Examples

ループの

```
void main()
{
    import std.stdio : writeln;
    int[] arr = [1, 3, 4];
    for (int i = 0; i < arr.length; i++)
    {
        arr[i] *= 2;
    }
    writeln(arr); // [2, 6, 8]
}
```

whileループ

```
void main()
{
    import std.stdio : writeln;
    int[] arr = [1, 3, 4];
    int i = 0;
    while (i < arr.length)
    {
        arr[i++] *= 2;
    }
    writeln(arr); // [2, 6, 8]
}
```

をいます

```
void main()
```

```
{
    import std.stdio : writeln;
    int[] arr = [1, 3, 4];
    int i = 0;
    assert(arr.length > 0, "Array must contain at least one element");
    do
    {
        arr[i++] *= 2;
    } while (i < arr.length);
    writeln(arr); // [2, 6, 8]
}
```

フォアハ

Foreachでは、エラーをこしにくく、みやすく、コレクションをすることができます。をしたい、`ref`をうことができます。

```
void main()
{
    import std.stdio : writeln;
    int[] arr = [1, 3, 4];
    foreach (ref el; arr)
    {
        el *= 2;
    }
    writeln(arr); // [2, 6, 8]
}
```

のインデックスにもアクセスできます。

```
void main()
{
    import std.stdio : writeln;
    int[] arr = [1, 3, 4];
    foreach (i, el; arr)
    {
        arr[i] = el * 2;
    }
    writeln(arr); // [2, 6, 8]
}
```

ののでのもです

```
void main()
{
    import std.stdio : writeln;
    int[] arr = [1, 3, 4];
    int i = 0;
    foreach_reverse (ref el; arr)
    {
        el += i++; // 4 is incremented by 0, 3 by 1, and 1 by 2
    }
    writeln(arr); // [3, 4, 4]
}
```

ブレーク、、ラベル

```
void main()
{
    import std.stdio : writeln;
    int[] arr = [1, 3, 4, 5];
    foreach (i, el; arr)
    {
        if (i == 0)
            continue; // continue with the next iteration
        arr[i] *= 2;
        if (i == 2)
            break; // stop the loop iteration
    }
    writeln(arr); // [1, 6, 8, 5]
}
```

ラベルをして、ネストされたループで`break`または`continue`することもできます。

```
void main()
{
    import std.stdio : writeln;
    int[] arr = [1, 3, 4];
    outer: foreach (j; 0..10) // iterates with j=0 and j=1
        foreach (i, el; arr)
        {
            arr[i] *= 2;
            if (j == 1)
                break outer; // stop the loop iteration
    }
    writeln(arr); // [4, 6, 8] (only 1 reaches the second iteration)
}
```

オンラインでループをむ https://riptutorial.com/ja/d/topic/4696/ループ

11: レンジ

コンパイラがforeachにした

```
foreach (element; range) {
```

それはにのようなきえられます

```
for (auto it = range; !it.empty; it.popFront()) {
    auto element = it.front;
    ...
}
```

のインタフェースをたすオブジェクトはすべてとばれ、りしできるです

```
struct InputRange {
    @property bool empty();
    @property T front();
    void popFront();
}
```

Examples

とはです

```
import std.stdio;

void main() {
    auto s = "hello world";
    auto a = [1, 2, 3, 4];

    foreach (c; s) {
        write(c, "!"); // h!e!l!l!o! !w!o!r!l!d!
    }
    writeln();

    foreach (x; a) {
        write(x * x, ", "); // 1, 4, 9, 16,
    }
}
```

しいタイプをする

`InputRange`コンセプトには、の3つのがあります。

```
struct InputRange(T) {
    @property bool empty();
    @property T front();
    void popFront();
```

```
}
```

するに、

1. がであるかどうかをする
2. のをする
3. のにする

のを`InputRange`にするには、これら3つのをするがあります。ののシーケンスをてみましょう。

```
struct SquaresRange {
    int cur = 1;

    @property bool empty() {
        return false;
    }

    @property int front() {
        return cur^^2;
    }

    void popFront() {
        cur++;
    }
}
```

フィボナッチのについては、 Dツアーをごください。

オンラインでレンジをむ https://riptutorial.com/ja/d/topic/3106/レンジ

12:

アサーションはリリースビルドでされます。

Examples

により、プログラマーはをチェックすることができます。には、なパラメータ、しいりのチェック、またはオブジェクトのながまれます。

チェックは、またはメソッドのがされるでされます。

```
void printNotGreaterThan42(uint number)
in {
    assert(number < 42);
}
body {
    import std.stdio : writeln;
    writeln(number);
}
```

アサーションはリリースビルドでされます。

たとえば、メソッドがびされた、オブジェクトのによって、メソッドがのパラメータでびされたかどうかがまったくできないがあります。

```
class OlderThanEighteen {
    uint age;

    final void driveCar()
    in {
         assert(age >= 18); // variable must be in range
    }
    body {
         // step on the gas
    }
}
```

オンラインでをむ https://riptutorial.com/ja/d/topic/6830/

13:

- __traitsTraitsKeyword、TraitsArguments ...

Examples

のメンバーをする

```
import std.stdio;

struct A {
    int b;
    void c();
    string d;
};

void main() {
    // The following foreach is unrolled in compile time
    foreach(name; __traits(allMembers, A)) {
        pragma(msg, name);
    }
}
```

allMembersは、されたのメンバのをむのタプルをします。これらのはコンパイルにされます。

されたメンバーなしでIクラスのメンバーをする

```
module main;

auto getMemberNames(T)() @safe pure {
    string[] members;

    foreach (derived; __traits(derivedMembers, T)) {
        members ~= derived;
    }

    return members;
}

class Foo {
    int a;
    int b;
}

class Bar : Foo {
    int c;
    int d;
    int e;
}

void main() {
    import std.stdio;
```

```
    foreach (member; getMemberNames!Bar) {
        writeln(member);
    }
}
```

*derivedMembers*はリテラルのタプルをします。はメンバです。

このはのとおりです。

```
c
d
e
```

オンラインでをむ https://riptutorial.com/ja/d/topic/3416/

14:

- Dのはです。インプレースするは、 `.dup`をして`char`をします。

Examples

をする

`string`は`alias string = immutable(char)[];`としてされ`alias string = immutable(char)[];` それはするに、 `dup`をしてcharをするがあります

```
import std.stdio;
import std.string;

int main() {

    string x = "Hello world!";
    char[] x_rev = x.dup.reverse;

    writeln(x_rev); // !dlrow olleH

    return 0;

}
```

のまたはのテスト

の

はヌルではありませんがさはゼロです

```
string emptyString = "";
// an empty string is not null...
assert(emptyString !is null);

// ... but it has zero lenght
assert(emptyString.length == 0);
```

ヌル

```
string nullString = null;
```

nullがnullのDe Lapalisse

```
assert(nullString is null);
```

しかし、Cとはなり、nullのさをみってもエラーはしません。

```
assert(nullString.length == 0);
assert(nullString.empty);
```

またはnullのテスト

```
if (emptyOrNullString.length == 0) {
}

// or
if (emptyOrNullString.length) {
}

// or
import std.array;
if (emptyOrNullString.empty) {
}
```

nullのテスト

```
if (nullString is null) {
}
```

- のをテストするしいはですか
- DはCのstring.Emptyをっていますか

をubyte []にし、そのもです

をの ubyte[]

```
string s = "unogatto";
immutable(ubyte[]) ustr = cast(immutable(ubyte)[])s;

assert(typeof(ustr).stringof == "immutable(ubyte[])");
assert(ustr.length == 8);
assert(ustr[0] == 0x75); //u
assert(ustr[1] == 0x6e); //n
assert(ustr[2] == 0x6f); //o
assert(ustr[3] == 0x67); //g
assert(ustr[7] == 0x6f); //o
```

を ubyte[] に ubyte[]

```
string s = "unogatto";
ubyte[] mustr = cast(ubyte[])s;

assert(typeof(mustr).stringof == "ubyte[]");

assert(mustr.length == 8);
assert(mustr[0] == 0x75);
assert(mustr[1] == 0x6e);
assert(mustr[2] == 0x6f);
assert(mustr[3] == 0x67);
assert(mustr[7] == 0x6f);
```

ubyte[] をに ubyte[] する

```
ubyte[] stream = [ 0x75, 0x6e, 0x6f, 0x67];
string us  = cast(string)stream;
assert(us == "unog");
```

- DLangフォーラム

オンラインでをむ https://riptutorial.com/ja/d/topic/5760/

15:

Examples

しいStructの

、、ageXHeightをつPersonというをするには

```
struct Person {
    int age;
    int height;
    float ageXHeight;
}
```

に

```
struct structName {
    /+ values go here +/
}
```

コンストラクタ

Dでは、クラスのようにをするコンストラクタをできます。のでしたのをするには、のようにします。

```
struct Person {
    this(int age, int height) {
        this.age = age;
        this.height = height;
        this.ageXHeight = cast(float)age * height;
    }
}

auto person = Person(18, 180);
```

オンラインでをむ https://riptutorial.com/ja/d/topic/4075/

16: とモジュール

- モジュールmy.package;
- import my.package;
- import my.packagefunction;
- import fancyName = mypackage;
- import my.packagefancyFunctionName = function;

モジュールはにののをします。モジュールははクラスにていますが、そのでなります

- モジュールのインスタンスは1つしかなく、にりてられます。
- テーブルはありません。
- モジュールはされず、スーパーモジュールもありません。
- ファイルごとに1つのモジュールのみ。
- モジュールシンボルをインポートできます。
- モジュールはにグローバルスコープでコンパイルされ、のやそののののをけません。
- モジュールはパッケージとばれるにグループできます。

モジュールはいくつかのをします

- モジュールがインポートされるは、セマンティクスにしません。
- モジュールのセマンティクスは、それをインポートすることによってをけません。
- モジュールCがモジュールAとBをインポートした、Bにをえても、AにするCのコードはかにされません。

Examples

グローバル

```
import std.stdio;
void main()
{
    writeln("Hello World!");
}
```

のインポートは、じにするか、 `comma` か、しいでします。

```
import std.stdio, std.math;
import std.datetime;
void main()
{
    writeln("2^4: ", pow(2, 4));
    writeln("Current time: ", Clock.currTime());
}
```

インポートは、コンパイラがのされたをするだけでむため、をクリーンアップしてコンパイルを

さらにするのにちます。

```
import std.stdio: writeln;
void main()
{
    writeln("Hello world");
}
```

また、のスコープでシンボルをインポートすることもできます。インポートは、スコープがなときつまりコンパイルされたときにのみされ、インポートされたはインポートされたスコープにのみされます。もには、ローカルインポートのスコープは、、クラスです。

```
void main()
{
    import std.stdio: writeln;
    writeln("Hello world");
}
// writeln isn't defined here
```

モジュールは、されているのモジュールに`public imports`することができます。

```
public import std.math;
// only exports the symbol 'pow'
public import std.math : pow;
```

の

インポートのローカルをえることができます。これをして、モジュールのシンボルへのすべてのは、のものでされなければなりません。

```
import io = std.stdio;
void main()
{
    io.writeln("Hello world");
    std.stdio.writeln("hello!"); // error, std is undefined
    writeln("hello!");           // error, writeln is undefined
}
```

されたインポートは、にいインポートをうときにです。

と

なインポートのをすることもできます。

```
void main()
{
    import std.stdio : fooln = writeln;
    fooln("Hello world");
}
```

モジュール

モジュールは、ソースファイルと11でしています。モジュールは、デフォルトではパスとをりいたファイルであり、モジュールでにすることができます。 `ModuleDeclaration`は、モジュールのとそれがするパッケージをします。しない、モジュールは、ソースファイルとじパスとのとみなされます。

```
module my.fancy.module;
```

オンラインでとモジュールをむ https://riptutorial.com/ja/d/topic/4344/とモジュール

17:

Examples

```
int[string] wordCount(string[] wordList) {
    int[string] words;
    foreach (word; wordList) {
        words[word]++;
    }
    return words;
}

void main() {
    int[string] count = wordCount(["hello", "world", "I", "say", "hello"]);
    foreach (key; count.keys) {
        writefln("%s: %s", key, count[key]);
    }
    // hello: 2
    // world: 1
    // I: 1
    // say: 1

    // note: the order in the foreach is unspecified
}
```

リテラル

```
int[string] aa0 = ["x": 5, "y": 6];  //int values, string keys
auto aa1 = ["x": 5.0, "y": 6.0];  // double values, string keys
string[int] aa2 = [10: "A", 11: "B"];  //string values, int keys
```

キーとのペアをする

```
int[string] aa = ["x": 5, "y": 6];
// The value can be set by its key:
aa["x"] = 7;
assert(aa["x"] == 7);
// if the key does not exist will be added
aa["z"] = 8;
assert(aa["z"] == 8);
```

キーとのペアをする

を`aa`としましょう

```
int[string] aa = ["x": 5, "y": 6];
```

キーのがされ、 `remove`が`true` `remove`した、 `.remove()`をしてをでき`true`。

```
assert(aa.remove("x"));
```

されたキーがしない、 `remove`はもせず`false`をし`false`。

```
assert(!aa.remove("z"));
```

キーがするかどうかをする

```
int[string] numbers = ["a" : 10, "b" : 20];

assert("a" in numbers);
assert("b" in numbers);
assert("c" in numbers);
```

オンラインでをむ https://riptutorial.com/ja/d/topic/3159/

クレジット

S. No		Contributors
1	Dのい	Alessio Sacco, André Puel, Bauss, Bennet Leff, Cauterite, Community, EsmaeelE, Gassa, Sirsireesh Kodali, Some coder, T.Furholzer, TuxCopter
2	UFCS - びしの	André Puel, greenify, tre0n
3	クラス	greenify, o3o
4	コンパイルCTFE	André Puel, greenify
5	スコープガード	André Puel, greenify
6	ダイナミックアレイスライス	André Puel, Bauss, greenify, Harry, Shriken
7	テンプレート	André Puel, Bauss, Quonux
8	メモリとポインタ	greenify
9	ユニットテスト	greenify
10	ループ	greenify, o3o
11	レンジ	André Puel, Cauterite, Shriken
12		Quonux
13		André Puel, Bauss
14		Harry, o3o, RamenChef
15		Bennet Leff
16	とモジュール	greenify
17		Bauss, greenify, o3o, Shriken