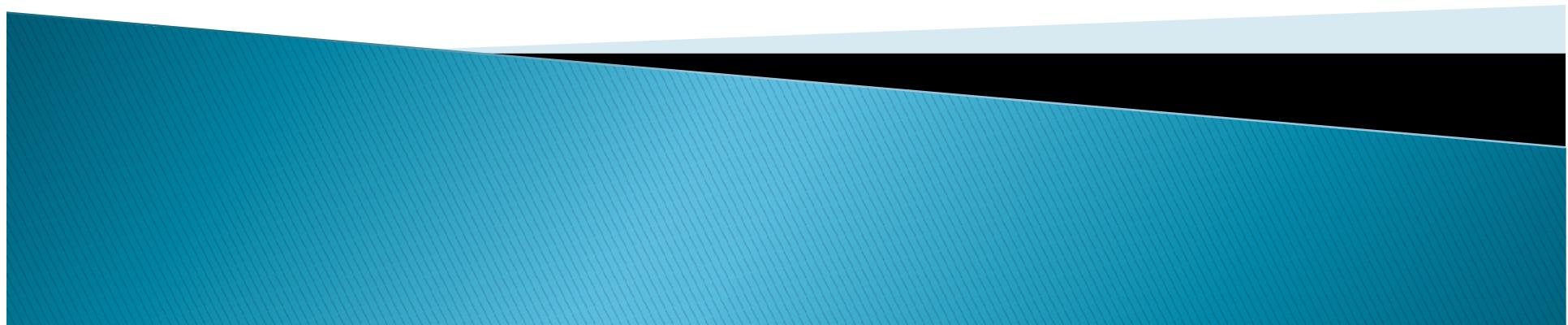


C#でゲームを作ろう2017

第1回 担当:ten

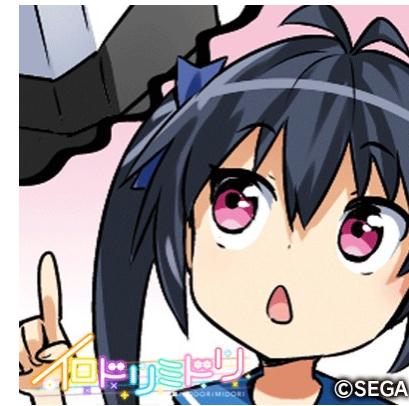


自己紹介

- ▶ ID:ten
- ▶ 京都大学工学部情報学科2回生
 - 計算機科学コースに配属された！！！
- ▶ 第40代会長
- ▶ ゲーム制作とか競プロとか
- ▶ 音ゲーとかパズルゲーとか



Twitter



Slackとか

自己紹介をしましょう

- ▶ 名前 or ID (あれば)
 - ▶ 所属 (学部学科とか)
 - ▶ 好きなゲームとか
 - ▶ なにか一言
-
- ▶ ※すべて任意です



今日の流れ

1. このプロジェクトの概要
2. 環境構築
3. プログラミング入門



概要

- ▶ ゲームを作りましょう
- ▶ C#というプログラミング言語と
Monogameというゲームエンジンを使います



概要

- ▶ 夏休み後あたりに発表会をする予定
- ▶ 夏休み前に内容を終えるので、そこから自分のゲームを作ってもらう予定
- ▶ というわけでゲームの内容を考えてきてください



概要

- ▶ RPGのような分量の多いものは厳しいかも
 - 実際に使える期間は3か月くらい？
 - わりと短い
- ▶ 作れそうなジャンル
 - シューティング
 - パズル
 - ブロック崩し
 - アクション
- ▶ いわゆる3分ゲーのようなものでいいです



概要

- ▶ 分からないことがあったらいつでも質問しましょう
 - 最初は分からなくて当たり前です
 - どんどん質問していきましょう
- ▶ 実際に手を動かしながら、プログラムを書いていきながら学ぶとよいです



概要

- ▶ Slackのチャンネルは #csgame です
- ▶ 質問などはここでどうぞ
- ▶ まだ部員でない方は、口頭でも質問を受け付けます



環境構築

- ▶ PCにいくつかソフトウェアを入れます
- ▶ ちょっと時間がかかるかも



環境構築の目次

1. Visual Studio Community 2017のインストール
2. Monogameのインストール



Visual Studio インストール

- ▶ <https://www.visualstudio.com/ja/downloads/?rr=https%3A%2F%2Fwww.microsoft.com%2Fja-jp%2Fdev%2Fproducts%2Fcommunity.aspx>
- ▶ ダウンロードページ



Visual Studio インストール

Visual Studio のダウンロード

Visual Studio 2017 Community

学生、オープンソース、個人の開発者向けの無料でフル機能の IDE

無償ダウンロード

Visual Studio 2017 Professional

小規模なチームのためのプロフェッショナル開発者用ツール、サービス、サブスクリプション特典

無料試用版

Visual Studio 2017 Enterprise

あらゆる規模のチームにおいて、品質やスケールについての厳しいニーズへの対応を可能にするエンドツーエンドソリューション

無料試用版

リリースノートとドキュメント

Visual Studio 各エディションの比較

オンラインでインストールする方法

Feedback

Visual Studio インストール



Visual Studio インストール

変更しています - Visual Studio Community 2017 (15.0.26228.12)

ワークフロー 個別のコンポーネント 言語パック

Windows (3)

- ユニバーサル Windows プラットフォーム開発
C#、VB、JavaScript、または C++ (オプション) を使ってユニバーサル Windows プラットフォームのアプリケーションを作成し...
- .NET デスクトップ開発
.NET Framework を使用して、WPF、Windows フォーム、コンソール アプリケーションをビルドします。
- C++ によるデスクトップ開発
Visual C++ ツールセットの機能、ATL、および MFC や C++/CLI などのオプション機能を使用して従来の Windows ベースのア...

Web & クラウド (5)

- ASP.NET と Web 開発
ASP.NET、ASP.NET Core、HTML、JavaScript、CSS を使った Web アプリケーションをビルドします。
- Azure の開発
クラウド アプリとリソースを開発するための Azure SDK、ツール、およびプロジェクト。
- Node.js 開発
Node.js (非同期イベント ドリブン JavaScript ランタイム) を使用してスケーラブルなネットワーク アプリケーションをビルドし...
- データの保存と処理
SQL Server、Azure Data Lake、Hadoop、または Azure ML を使用する接続、開発、およびテストデータソリューション。
- Office/SharePoint 開発

場所
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community

概要

- > C++ によるゲーム開発
- > Visual Studio 拡張機能の開発
- > C++ による Linux 開発
- > .NET Core クロスプラット... *

含まれるアクセス許可

- ✓ .NET Core 1.0 - 1.1 開発ツール
- ✓ .NET Framework 4.6.1 開発ツール
- ✓ ASP.NET と Web の開発ツール
- ✓ Developer Analytics Tools

省略可能

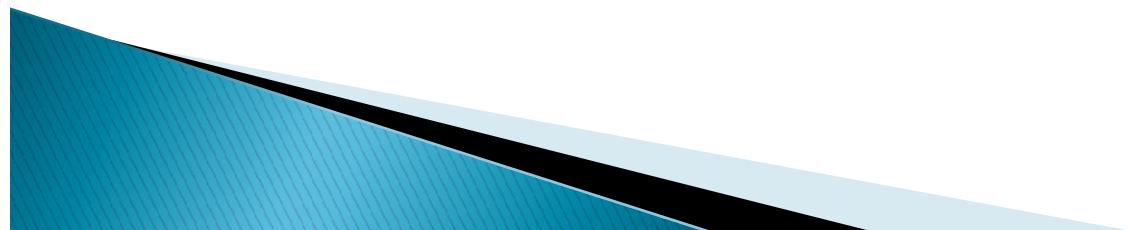
- ✓ コンテナー開発ツール

続行すると、選択した Visual Studio のエディションのライセンスに同意することになります。また、Visual Studio を使用して他のソフトウェアをダウンロードする機能も提供されます。このソフトウェアは、[カードパーティに関する通知](#)または付属するライセンスに記載のとおり、個別にライセンスされています。続行することで、これらのライセンスにも同意することになります。

インストールサイズ: 0 KB

変更

Visual Studio インストール

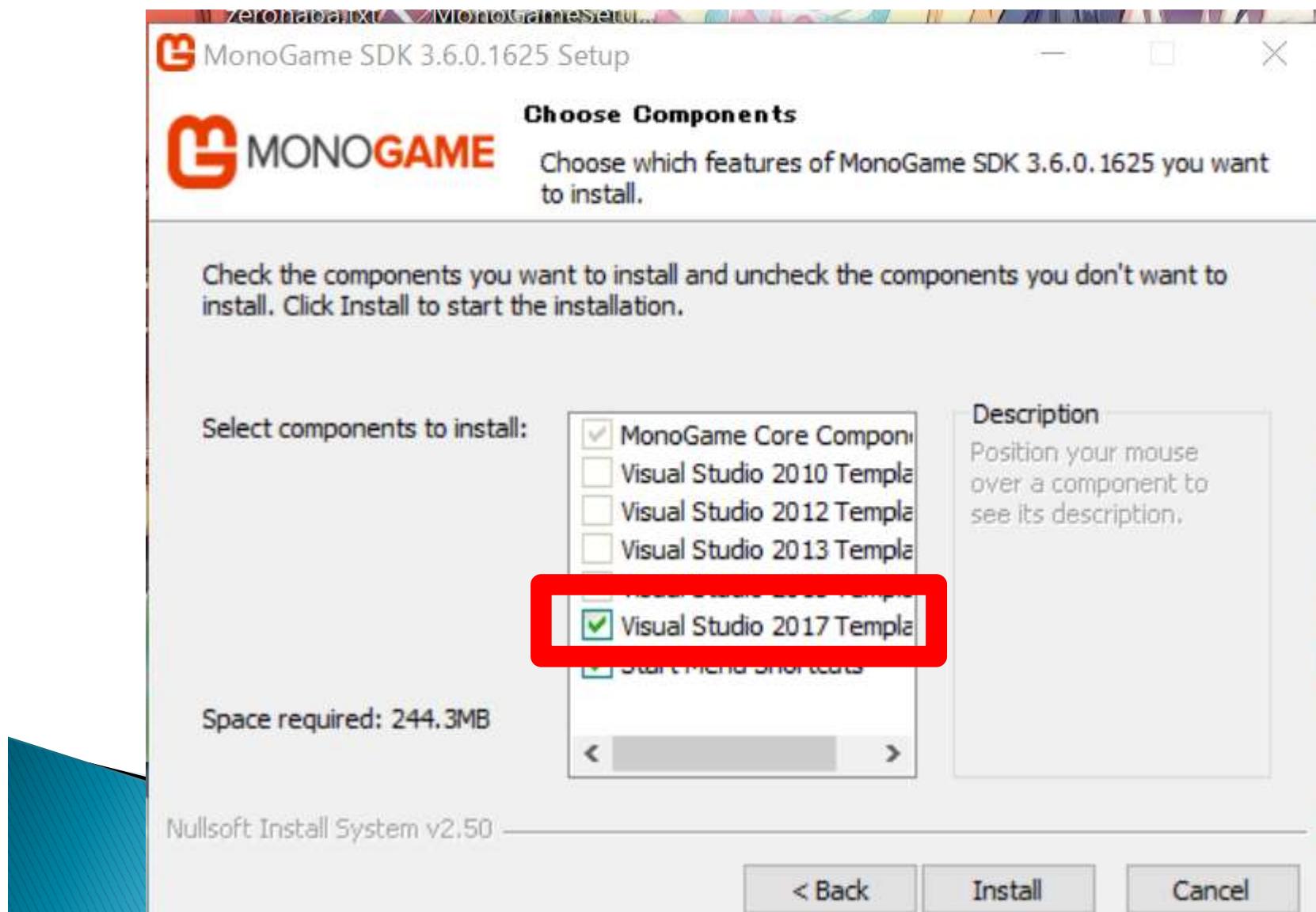


Monogameインストール

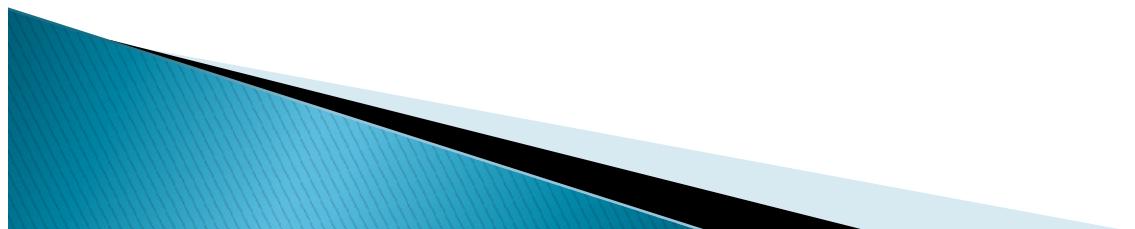
- ▶ <http://www.monogame.net/downloads/>
- ▶ ダウンロードページ
- ▶ Ver3.6をつかいます



Monogameインストール



Monogameインストール



環境構築おわったー

▶ ここまで1時間くらいであってほしい

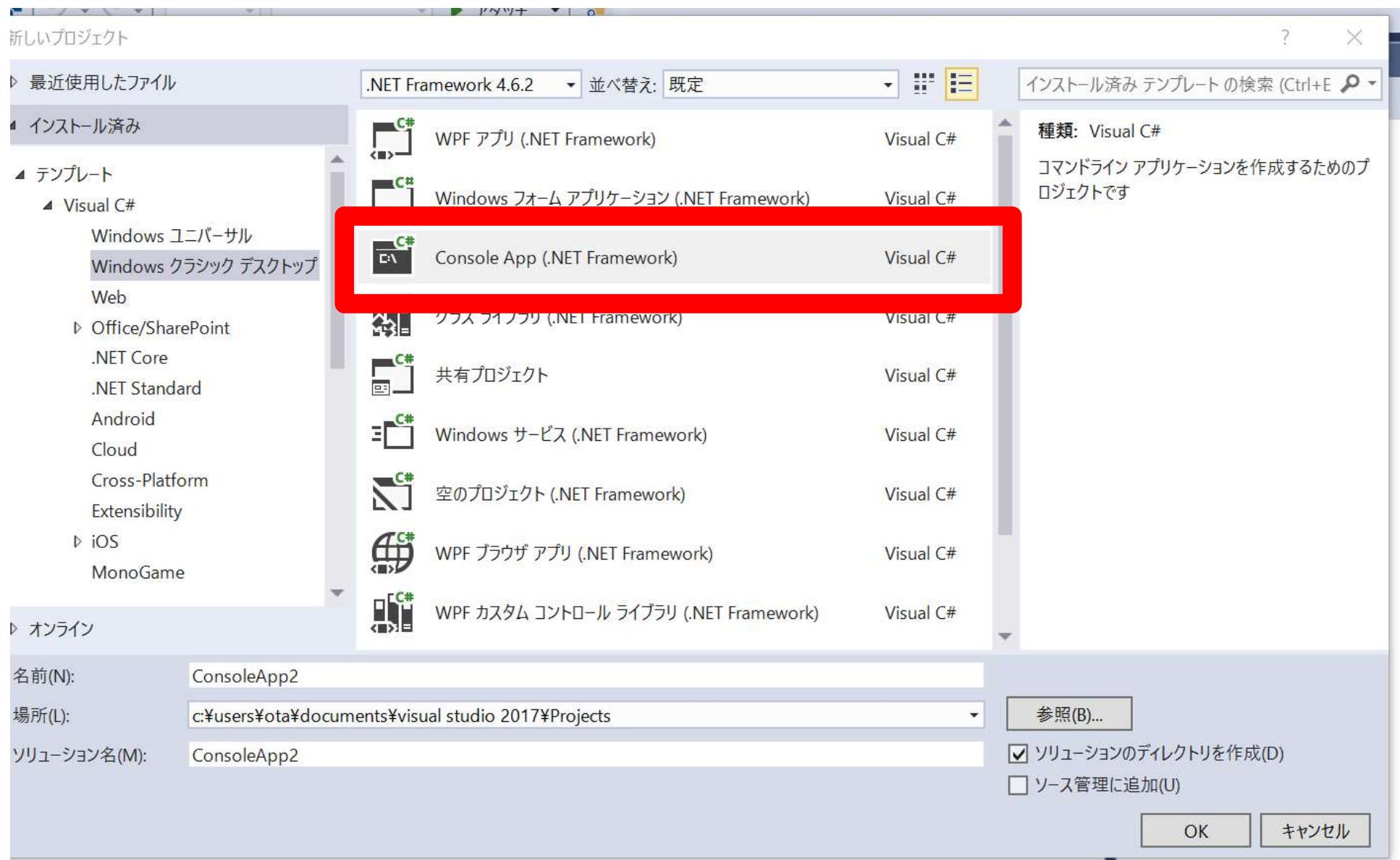


プログラム

▶ プログラム入門しましょう



プログラム



プログラム

- ▶ <https://ideone.com/>
- ▶ <https://wandbox.org/>

- ▶ ブラウザ上でC#のコードが動かせる
 - 最高
- ▶ 環境構築が終わってない方は使いましょう



プログラムの目次

1. Hello,World!
2. 変数
3. 条件分岐
4. ループ
5. 配列
6. その他いろいろ
7. じゃんけんを作る



Hello,World!

```
public class Test {  
    public static void Main(){  
        System.Console.WriteLine("Hello, World!");  
    }  
}
```

- ▶ 枠の中にこのように書いてください



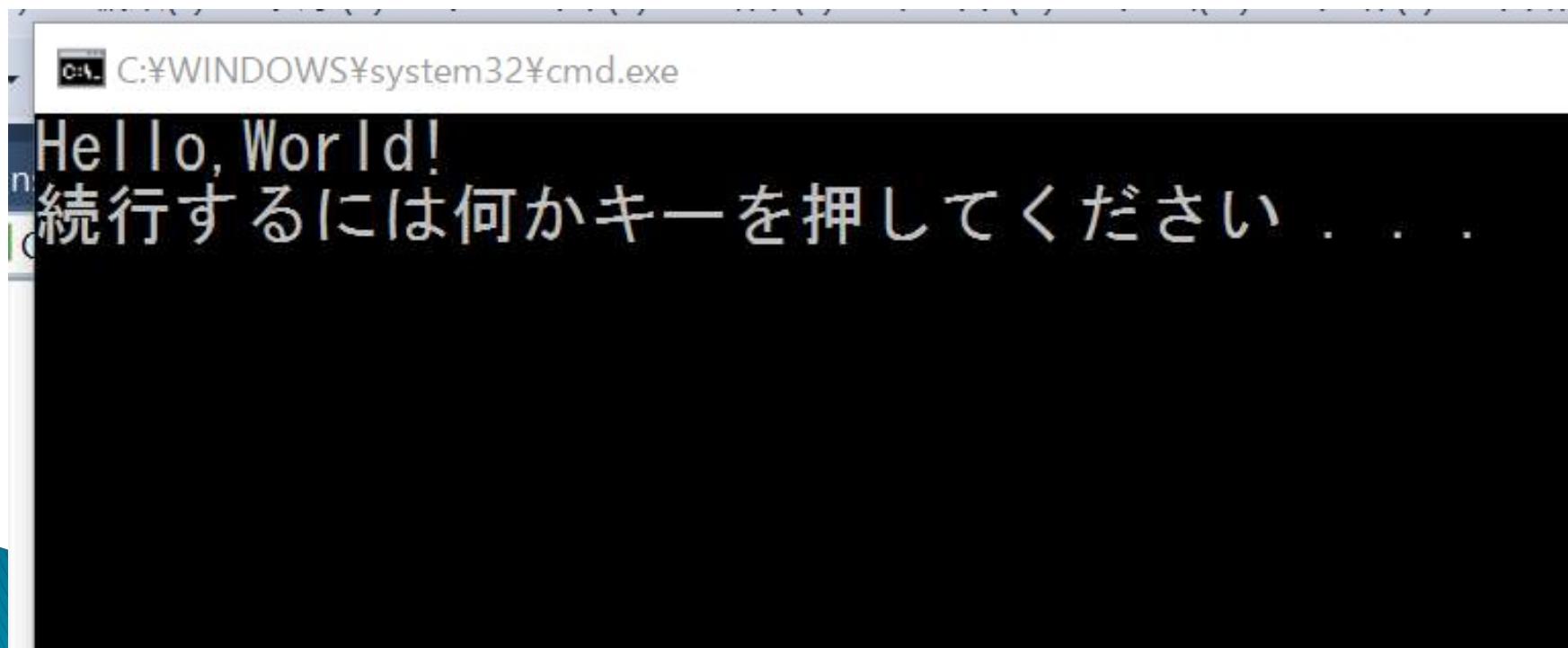
Hello,World!

- ▶ 実行する
- ▶ Hello,World!と出力される
- ▶ けど、一瞬で消える
- ▶ デバッグなしで開始



Hello,World!

- ▶ 実行する
- ▶ Hello,World!と出力される
- ▶ 今度こそ成功



Hello,World!

```
public class Test {  
    public static void Main(){  
        System.Console.WriteLine("Hello, World!");  
    }  
}
```

- ▶ 今はこの部分だけ見てくれば大丈夫です
- ▶ 以降はMain()の{}の中だけ書きます



Hello,World!

```
System.Console.WriteLine("Hello, World!");
```

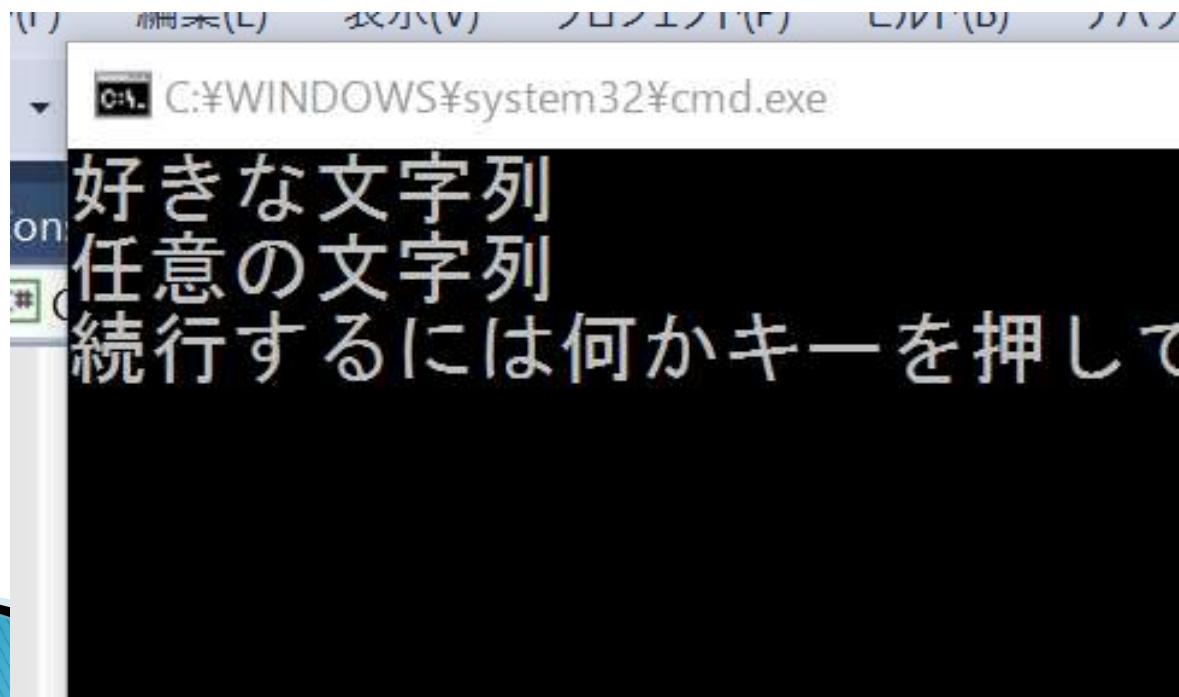
- ▶ System.Console.WriteLine();
 - ()の中に書いた文字列などを出力する
 - これを 2 つ書くと 2 つ出力される



Hello,World!

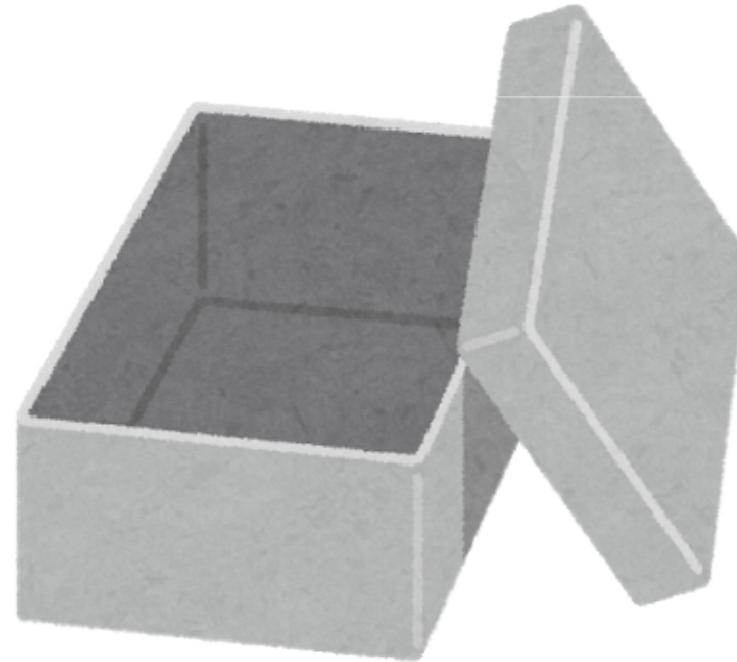
```
System.Console.WriteLine("好きな文字列");  
System.Console.WriteLine("任意の文字列");
```

- ▶ こう書いて実行する



変数

- ▶ いろいろな「値」を入れる箱のようなもの
- ▶ 箱に入れられる値の種類や範囲は「型」によって異なる



変数

- ▶ 型の例
 - int 整数(-2,147,483,648 ~ 2,147,483,647)
 - double 実数
 - string 文字列("Hello,World!")
 - bool 真偽値(true or false)
 - etc
- ▶ 変数を使うには初期化が必要
 - 型名 变数名;のように宣言
 - int a;と書くと、int型のaという名前の変数



変数

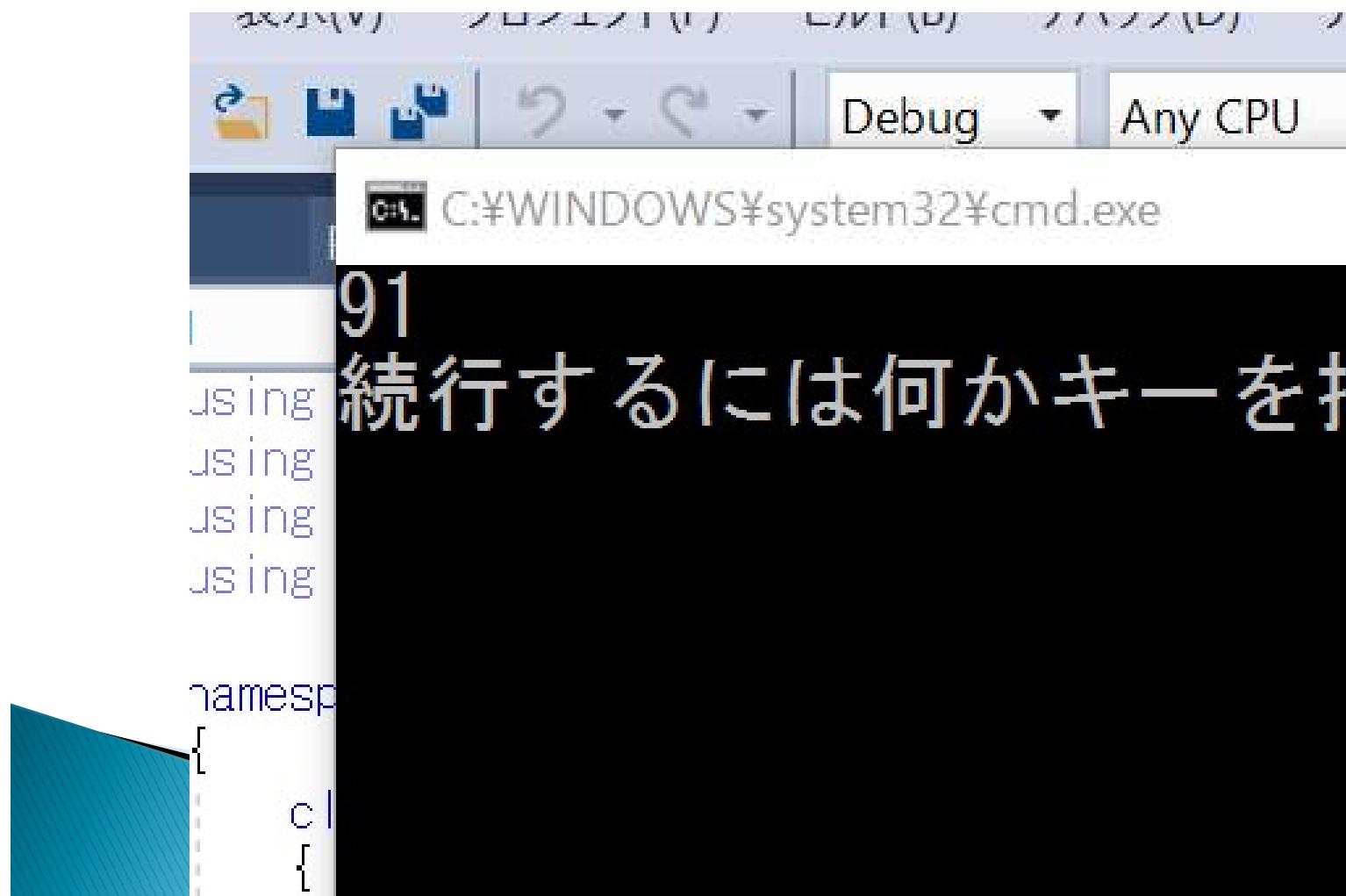
- ▶ じゃあどういう感じに使うの
- ▶ Main()の{}内に次のように書いてみる

```
int a;  
a=91;  
System.Console.WriteLine(a);
```



変数

▶ 91。



The screenshot shows a Microsoft Visual Studio interface. At the top is the menu bar with Japanese text. Below it is the toolbar with icons for file operations. To the right of the toolbar are dropdown menus for "Debug" and "Any CPU". A status bar at the bottom displays the path "C:\WINDOWS\system32\cmd.exe". The main area contains a C# code editor with the following text:

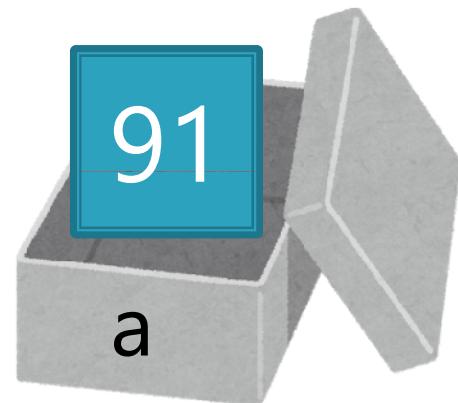
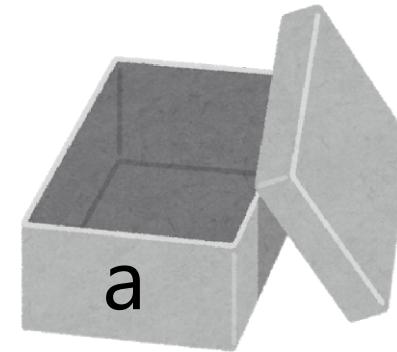
```
91
|
using
using
using
using

namespace
{
    class
    {
}
```

A large red arrow points from the text "91" in the terminal window towards the first "91" in the code editor.

解説

- ▶ `int a;`
 - `int`型の`a`という名前の変数を宣言
- ▶ `a=91;`
 - `a`という変数に`91`という値を入れている
 - 「代入」といいます
 - `a`に初期値を与えている
 - `a`と`91`が等しいという意味ではない
- ▶ `System.Console.WriteLine(a);`
 - ()の中に変数名を入れると、その値を出力できる



解説

- ▶ `a=91;`
 - 91の部分を変えると出力される数字が変わる
 - 整数のみ
 - 大きすぎる値もダメ
- ▶ `int a;`を`double a;`にすると実数が入れられる
- ▶ `string`にすると文字列になる
 - "Hello,World!"も文字列



四則演算

- ▶ intやdoubleなどは四則演算ができる
- ▶ int x;
- ▶ $x=4+3;$ (加算)
- ▶ $x=9-3;$ (減算)
- ▶ $x=7*3;$ (乗算)
- ▶ $x=20/3;$ (除算) (商)
- ▶ $x=20\%3$ (除算) (あまり)

- ▶ stringは+で文字列結合ができる
 - "けもの"+"フレンズ"



四則演算

▶ 省略形

- $x += 2;$
- $x = x + 2;$

- $X +=;$
- $++X;$
- $x += 1;$



四則演算

▶ いろいろ

```
int a=334;  
a*=2;  
a++;  
System.Console.WriteLine(a);  
double b=2.5;  
b=b*b;  
System.Console.WriteLine(b);
```



コメント

- ▶ プログラムを書いてると、何のプログラムか分からなくなることがある
- ▶ 説明文とか書きたい



コメント

- ▶ //のあとに書いた文字は全て実行時は無視される
- ▶ 何でも書き放題
- ▶ /* ~~ */ の中の~~も無視される

```
int a = 10; //ここはコメント  
a=a*a; // 2乗するぞ~~~  
/*  
この間ならなにを書いても  
許される  
*/
```



条件分岐

- ▶ if(){
 - ()の中の条件を満たしているときのみ
{}内の命令を行う

```
int a = 10;  
if (a > 5){  
    System.Console.WriteLine("ハイライト");  
}
```

条件分岐

- ▶ else
 - ()内が満たされない場合
elseの次の{}内が実行

```
int a = 10;  
if (a > 5){  
    System.Console.WriteLine("ハイライト");  
}else{  
    System.Console.WriteLine("キラメキ☆JAPAN");  
}
```



条件分岐

- ▶ これもできる

```
int a = 10;
if (a > 5){
    System.Console.WriteLine("ハイライト");
}else if(a<0){
    System.Console.WriteLine("キラメキ☆JAPAN");
}else{
    System.Console.WriteLine("ジェイムズキッチン");
}
```



条件分岐

▶ $a > 5$ は a が5より大きい

- > より大きい
- < より小さい
- \geq 以上
- \leq 以下
- \equiv 等しい
- \neq 等しくない



条件分岐

- ▶ if(){}の()に入れるのはbool型
 - bool 真偽値(true or false)
 - true 真
 - false 偽

```
bool b=true;  
if (b){  
    System.Console.WriteLine("必ず出力される！！");  
}  
b=false;  
if (b){  
    System.Console.WriteLine("出力されねえ！！！");  
}
```

条件分岐

- ▶ これも可能

```
if (true){  
    System.Console.WriteLine("必ず出力される！！");  
}  
if (false){  
    System.Console.WriteLine("出力されねえ！！！");  
}
```



条件分岐

- ▶ これも可能

```
int a=4;  
bool b=a>3; //true  
if (b){  
    System.Console.WriteLine("ほげ");  
}
```



条件分岐

- ▶ switch
- ▶ ある変数の値がnだったら、case n:～break;を実行
- ▶ defaultはいずれにも当てはまらなかった場合

```
int a=2;
switch (a) {
case 0:
System.Console.WriteLine("キラメキ☆JAPAN"); break;
case 1:
System.Console.WriteLine("キラメキノトリ"); break;
default:
System.Console.WriteLine("夕日のキラメキー乗寺"); break;
}
```

whileループ

- ▶ while(){}
 - ()内の条件を満たしてゐる間
 - { }内を実行し続ける

```
int i=1;
while (i<=12){
    System.Console.WriteLine("第" + i + "話");
    i++;
}
```



whileループ

- ▶ while(true){
 - 無限ループ
 - やめてくれ

```
while (true){  
    System.Console.WriteLine("おいバカやめろ");  
}
```



forループ

▶ for(){}

- ()の中でセミコロンで3つに区切られている
 1. 初期化文（最初のみ呼ばれる）
 2. 繼続条件
 3. 変数の更新（最初以外、ループされる度に呼ばれる）

```
for (int i=1; i<=10; ++i){  
    System.Console.WriteLine("KMC"+i+"回生");  
}
```

ループを抜けたい

- ▶ **break;**
 - 直前のwhileやforを強制的に抜ける
 - 無限ループも抜けれる

```
int i=1;
while (true){
    System.Console.WriteLine("わーい");
    i++;
    if(i>=5){
        break;
    }
}
```

配列とか

- ▶ ゲームでは同じような値を使いたいときがある
- ▶ 下のような書き方はきつい
- ▶ 10ならまだしも1000とかになると...

```
int enemy_hp1=20;  
int enemy_hp2=20;  
int enemy_hp3=20;  
int enemy_hp4=20;  
int enemy_hp5=20;  
int enemy_hp6=20;  
int enemy_hp7=20;  
int enemy_hp8=20;  
int enemy_hp9=20;  
int enemy_hp10=20;
```



配列

- ▶ そこで配列
 - intをint[]にする
 - new int[n] のnは要素数
 - n=3なら enemy_hp[0]～enemy_hp[2]が作られる

```
int[] enemy_hp=new int[3];  
enemy_hp[0]=20;  
enemy_hp[1]=25;  
enemy_hp[2]=30;
```



配列

- ▶ さっきと同じものを表している

```
int[] enemy_hp=new int[]{20,25,30};
```



配列

- ▶ シュッと出力したい
- ▶ そのためのfor

```
int[] enemy_hp=new int[]{20,25,30};  
for(int i=0; i<3; ++i){  
    System.Console.WriteLine("敵"+i+"のHP:"+enemy_hp[i]);  
}
```

配列

- ▶ 1000個とか作りたい
- ▶ そのためのfor

```
int[] enemy_hp=new int[1000];
for(int i=0; i<1000; ++i){
    enemy_hp[i]=20;
}
```



入力

- ▶ System.Console.ReadLine();
 - 黒い画面に入力できる
 - 入力した文字を受け取る

```
string str;  
str=System.Console.ReadLine();  
System.Console.WriteLine(str+str);
```



入力

- ▶ System.Console.ReadLine();
 - string型で受け取る
 - int型にしたい...
- ▶ int.Parse(文字列);
 - “334”みたいな文字列をint型にしてくれる

```
string str;  
str=System.Console.ReadLine();  
int num;  
num=int.Parse(str);  
System.Console.WriteLine(num*num);
```

乱数

- ▶ ゲームにはランダムで選ばれるものがある
- ▶ さいころ、くじ
- ▶ 敵の行動

```
Random r = new Random();
int a = r.Next(10); // 0以上10未満の整数
int b = r.Next(-5, 5); // -5以上5未満の整数
double d = r.NextDouble(); // 0以上1未満の実数
```



変数の見える範囲

- ▶ 変数には見える範囲がある
- ▶ if内やwhile内などで宣言した変数は
その外では使えない

```
if(true){  
    System.Console.WriteLine(a);//宣言前には使えない  
    int a=3;  
    System.Console.WriteLine(a);//大丈夫  
}  
System.Console.WriteLine(a);//if()の外からは見れない
```



じゃんけん

- ▶ 今までの内容でじゃんけんが作れます
- ▶ グー=0、チョキ=1、パー=2として入力を受け取る
- ▶ 敵の手はランダムで決める
- ▶ ちなみに文字列の中に「¥n」を入れると改行

```
System.Console.WriteLine("じゃんけんゲーム！¥n自分の手  
を入力してください¥nグー=0、チョキ=1、パー=2 ");
```



じゃんけん

▶ 次のページから解説するよ



じゃんけん

- ▶ プレイヤーの手を受け取る

```
int player_hand;

while (true) {
    System.Console.WriteLine("じゃんけんゲーム！\n自分の手
    を入力してください\nグー = 0、チョキ = 1、パー = 2 ");

    player_hand = int.Parse(System.Console.ReadLine());

    if (0 <= player_hand && player_hand <= 2){break;}
}
```

じゃんけん

- ▶ プレイヤーの手を表示

```
switch (player_hand) {  
case 0:  
System.Console.WriteLine("あなたの手:グー"); break;  
case 1:  
System.Console.WriteLine("あなたの手:チョキ"); break;  
case 2:  
System.Console.WriteLine("あなたの手:パー"); break;  
}
```



じゃんけん

- ▶ プレイヤーの手を表示

```
switch (player_hand) {  
case 0:  
System.Console.WriteLine("あなたの手:グー"); break;  
case 1:  
System.Console.WriteLine("あなたの手:チョキ"); break;  
case 2:  
System.Console.WriteLine("あなたの手:パー"); break;  
}
```



じゃんけん

- ▶ 敵の手をランダムで決める

```
Random r = new Random();
int enemy_hand = r.Next(3);
```



じゃんけん

- ▶ 敵の手を表示

```
switch (enemy_hand) {  
case 0:  
System.Console.WriteLine("敵の手:グー"); break;  
case 1:  
System.Console.WriteLine("敵の手:チョキ"); break;  
case 2:  
System.Console.WriteLine("敵の手:パー"); break;  
}
```



じゃんけん

▶ 判定

```
if (player_hand == enemy_hand) {  
    System.Console.WriteLine("あいこ！");  
} else  
if ((player_hand + 1) % 3 == enemy_hand) {  
    System.Console.WriteLine("あなたの勝ち！");  
} else {  
    System.Console.WriteLine("敵の勝ち！");  
}
```



じゃんけん

- ▶ じゃんけんができた
- ▶ 難しかったかも

- ▶ 実は戦闘シミュレーションも作れる
 - コマンドを「攻撃/防御/必殺技」とかにしたり



おわり

- ▶ ちょっと多かったかも？
- ▶ 分からなくとも心配しないでください
 - 最初からできるはずはありません
- ▶ 困ったらいつでも質問してください
 - #csgame
- ▶ 実際に書いてみるなどして覚えるといいかも
 - いずれ慣れます



おつかれさまでした

- ▶ 第2回は4/22にやります
- ▶ 来週はTOEFLがあるらしい
- ▶ がんばれ！！

