

# #2

# C#と時の部屋

## ～文字列型と配列編～

旅人「数値と違って、文字列の操作って大変…。」

C#「文字列型ができることは単純じゃよ。

配列はただの入れ物、ループ処理が強いのを。」

# ご注意

- Unityの基本操作の説明は、省略します。  
(実践タイムの時に個別で説明します！)
- 独断と偏見でピックアップした内容です。
- 初めての人向けの内容は、初心者マーク。  
シンプルな方法に絞っています。
- 慣れた人向けの内容は、ビックリマーク。  
初めての人はスルーしても良いです。





# 今回の目的

- 文字列の連結と文字数を見る！
- 文字列は `null` と `empty` がある！
- 配列という入れ物とその中身を見る！
- 配列を `foreach` でループする！
- `char`型 は `string` の1文字分である！

# 目次

1. Sharp1.cs で前回のおさらい！
2. Sharp2.cs を説明します！
3. Sharp2シーンを各自で実践タイム！  
(ブレイクポイントのデバッグ方法など個別に説明)
4. Sharp2Advanced.cs を説明します！
5. まとめを説明します！



# 開発環境（おさらい）

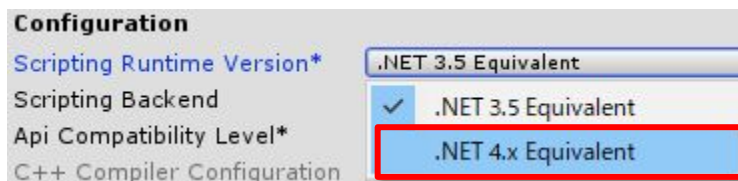


- 開発環境が強力であることも、C#の良さです。
- MonoDevelopやXamarin Studioは捨てましょう。  
<https://blogs.unity3d.com/jp/2018/01/05/discontinuing-support-for-monodevelop-unity-starting-in-unity-2018-1/>
- Windows
  - Visual Studio Community 2017 (Visual Studio Tools for Unity)
  - <http://tsubakit1.hateblo.jp/entry/2016/11/20/233000>
- Mac
  - Visual Studio for Mac
  - <http://kan-kikuchi.hatenablog.com/entry/VisualStudioforMac>



# 開発環境（おさらい）

1. [Edit] → [Project Settings] → [Player]を開く。
2. [Player Settings] の [Other Settings] の [Congiguration] の Scripting Runtime Version を **.NET 4.x** に設定する。



最初のUnityは C# 4.0 / .NET 3.5

現在のUnityは C# 6.0 / .NET 4.7.1

最新(Unity非対応)は C# 7.3 / .NET 4.7.2

<https://blogs.unity3d.com/jp/2018/03/28/updated-scripting-runtime-in-unity-2018-1-what-does-the-future-hold/>

# デバッグ (おさらい)

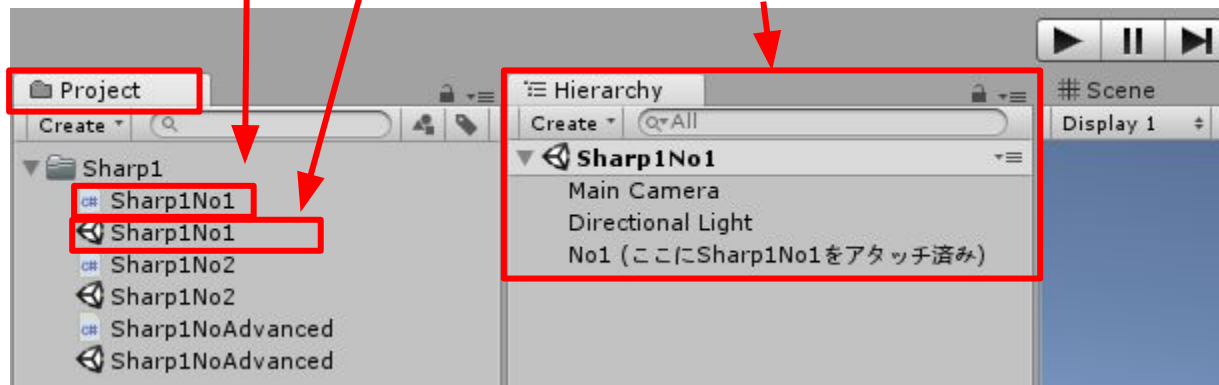


1. シーンとスクリプト(Visual Studio)を開く。

ダブルクリックでVisual Studioが開く。

ダブルクリックでシーンが開く。

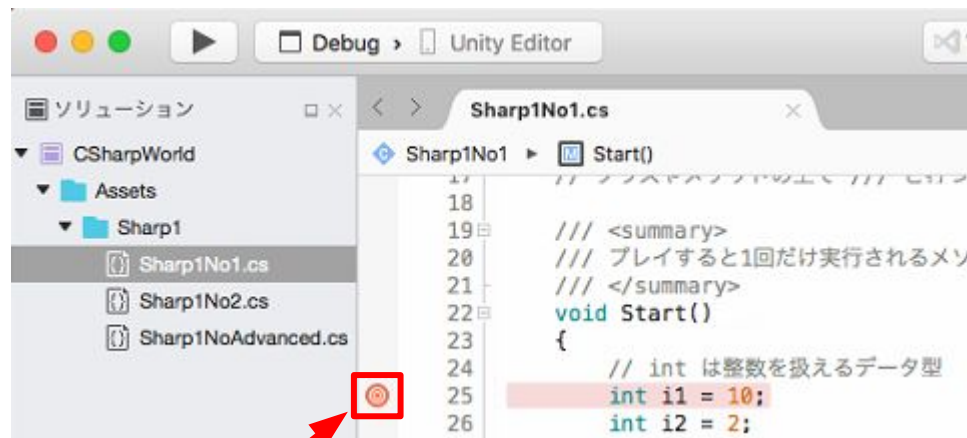
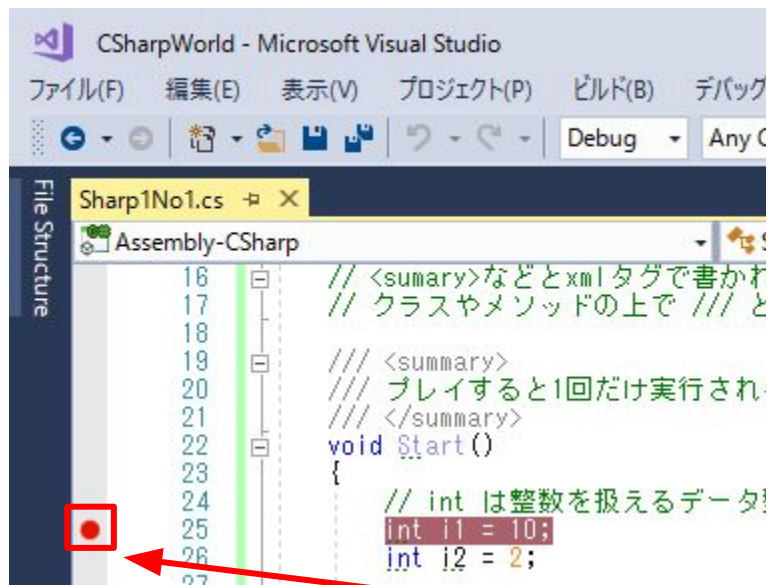
開かれたシーンがこちら。



# デバッグ (おさらい)



## 2. ブレークポイントを設定する。



ここをクリックすると、赤くなる。  
ここでプログラムを一時停止できる。



# デバッグ (おさらい)



## 3. Unity にアタッチする。

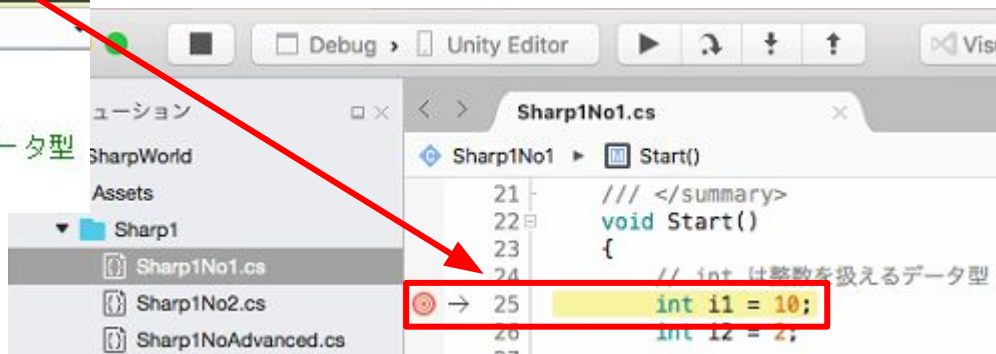
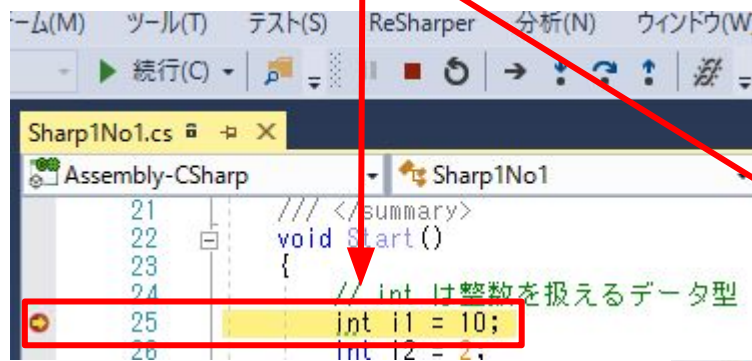
The image shows two overlapping windows. The top window is Microsoft Visual Studio, displaying the code for `Sharp1No1.cs`. The code includes XML documentation comments and a `Start()` method. The `Start()` method contains two lines of code: `int i1 = 10;` and `int i2 = 2;`. A red box highlights the `Unity にアタッチ` button in the Visual Studio toolbar. The bottom window is the Unity Editor, showing the `Sharp1No1` component selected in the Hierarchy. The `Start()` method is visible in the Inspector, and a red box highlights the play button in the Unity Editor toolbar.

```
16 // <summary>などとxmlタグで書かれているものは  
17 // クラスやメソッドの上で /// と打つと挿入さ  
18  
19 /// <summary>  
20 /// プレイすると1回だけ実行されるメソッド  
21 /// </summary>  
22 void Start()  
23 {  
24     // int は整数を扱えるデータ型  
25     int i1 = 10;  
26     int i2 = 2;  
27 }
```

# デバッグ (おさらい)



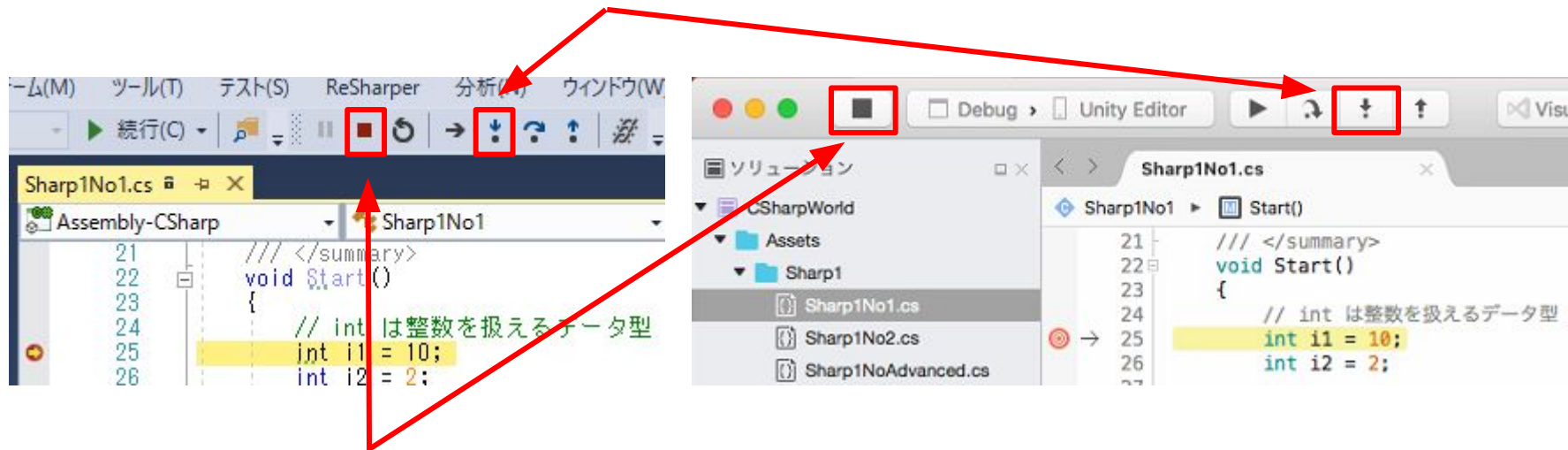
- Unity をプレイする。
- ブレークポイントで一時停止される。  
(行が黄色になる)



# デバッグ (おさらい)

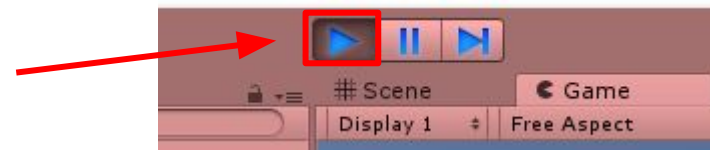


6. ステップインで1命令ずつ進めて確認できる。



7. デバッグを終了する。

8. **その後**に Unity のプレイを終了する。



# ショートカットキー (おさらい)



自分の環境で確認！

次のブレークポイントまで  
一気に進める。

ステップ実行

<オーバー>

メソッドの中に入らずに進める。

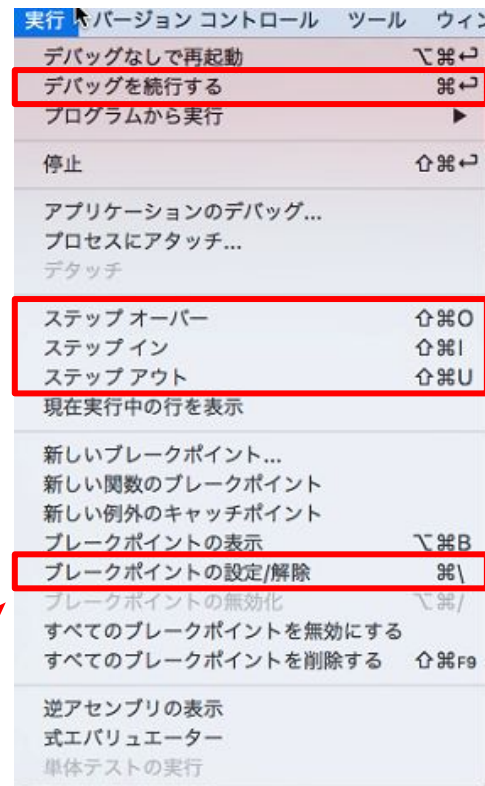
<イン>

メソッドの中に入る。1命令ずつ進む。

<アウト>

メソッドの中を全て実行して、  
メソッドの外に出る。

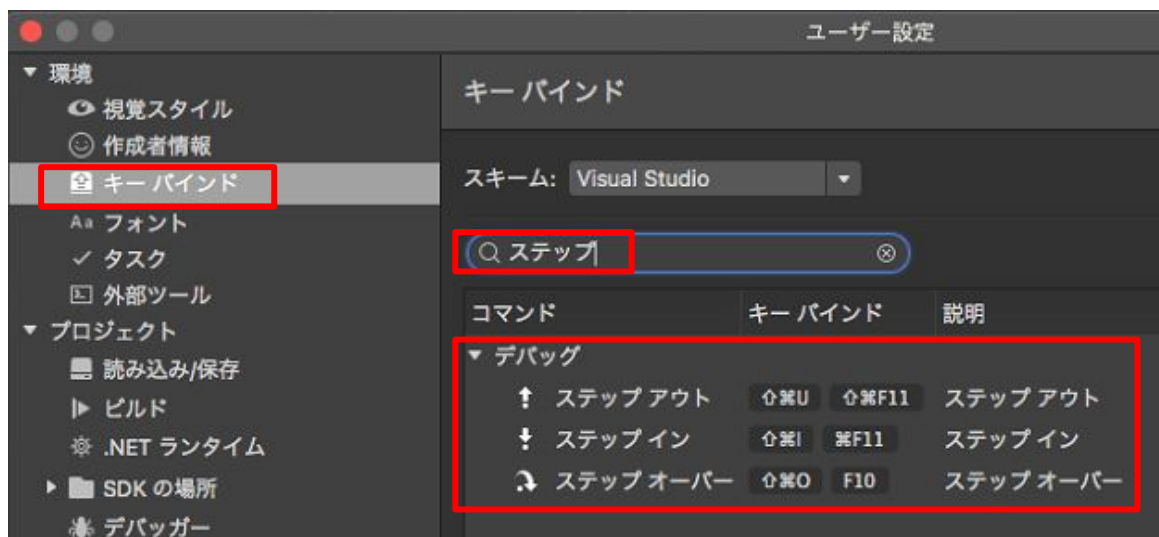
ブレークポイントの設定





# ショートカットキー（おさらい）

Macの場合は他にもショートカットキーがあり、こちらの方が便利だと思う。  
むしろ、押しやすい設定に変えた方が使いやすいのでは…。





# 前回の目的（おさらい）

- 開発環境を準備してデバッグする！
- メソッドを書いて実行する！
- 変数で int(整数型) と bool(論理型) を使う！
- if で条件判定する！
- for で繰り返し(ループ)処理する！



# ソースコード1

Sharp2シーンを開き、Sharp2.csをデバッグ実行する。

1. **string** ... 文字列を扱うという宣言。
2. **文字数** ... 変数.Length というプロパティがある。
3. **操作** ... 変数.Xxx というメソッドがある。
4. **null** ... nullだとプロパティもメソッドも使えない。
  
5. **配列** ... new int[3] のように [] で扱う。
6. **入れ物と中身** ... 中身が同じでも入れ物が異なる点に注意。
7. **foreach** ... 配列を全部ループできる。
8. **char** ... ' (シングルクォーテーション)で囲んで1文字を定義する。
9. **char配列** ... string は char の配列とみなせる。

# ソースコード2



Sharp2シーンを開き、[Sharp2Advanced.cs](#)をデバッグ実行する。

1. **System.Text.StringBuilder** ... 複数回の文字列連結で使用。
2. **型混在の連結** ... string型に合わせてから連結。
3. **比較規則** ... StringComparison.Ordinalを使用。
4. **T[]** ... T[] で済むなら List<T> は使わない。
5. **System.Linq.SequenceEqual** ... コレクションの中身を一致確認。
6. **==とEquals** ... == の方が型付けが強い。
7. **ReferenceEqual** ... GameObjectは == で比較が安全。



# #2完了

# C#と時の部屋

## ～文字列型と配列編～

string は色々な型と変換したり、頻繁に使います。  
配列 はループ処理が出来るので、頻繁に使います。  
基本が分かれば、変換方法やクラスを覚えていくだけ！