#1
C#と時の部屋
～メソッドとデバッグ編～

旅人「C#入門だ！う…、データ型とかクラスとか
いきなりワケワカメや＼(^o^)／！！！」
C#「intが分かればええよ。ifとforで夢が広がるんや。」

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

- 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！

目次

1. 開発環境とデバッグを説明します！(設定済み)

2. Sharp1シーンを説明します。
3. Sharp1シーンを各自で実践タイム (個別に説明します)
4. Sharp2シーンを説明します。
5. Sharp2シーンを各自で実践タイム (個別に説明します)
6. Sharp3シーンを説明します。
7. Sharp3シーンを各自で実践タイム (個別に説明します)

8. まとめを説明します！

開発環境

- 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

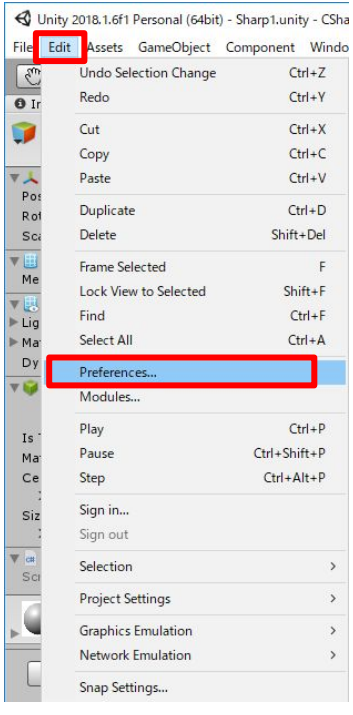
開発環境

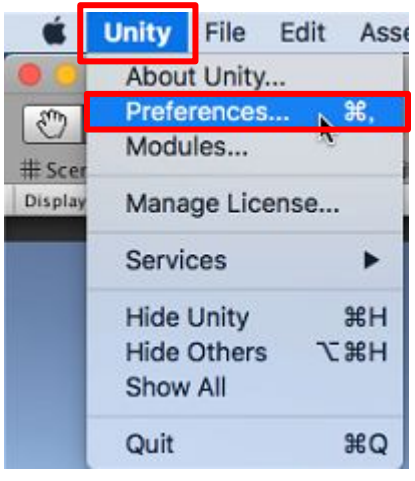
1. [Edit] -> [Project Settings] -> [Player]を開く。
2. [Player Settings] の [Other Settings] の [Congiguration] の Scripting Runtime Version を .NET 4.x に設定する。

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/

エディタ

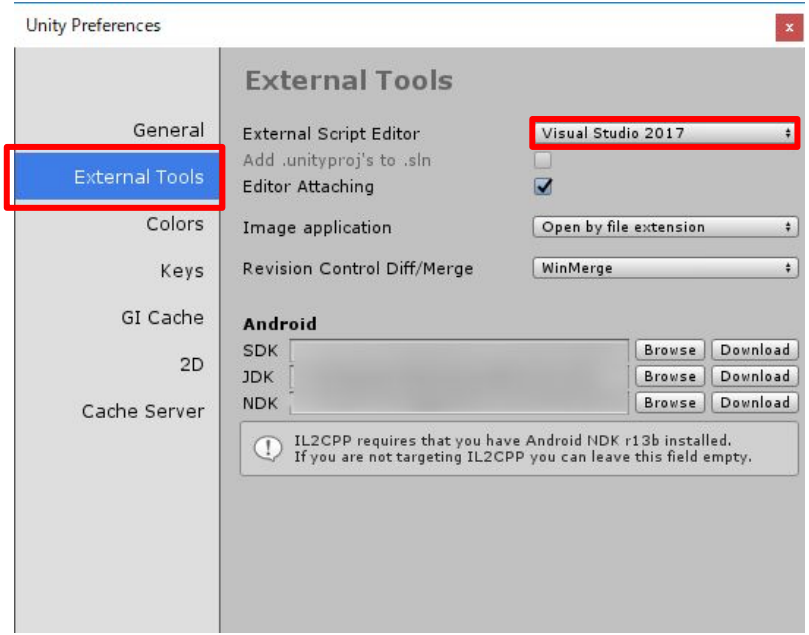
1. [Preference...]を開く。

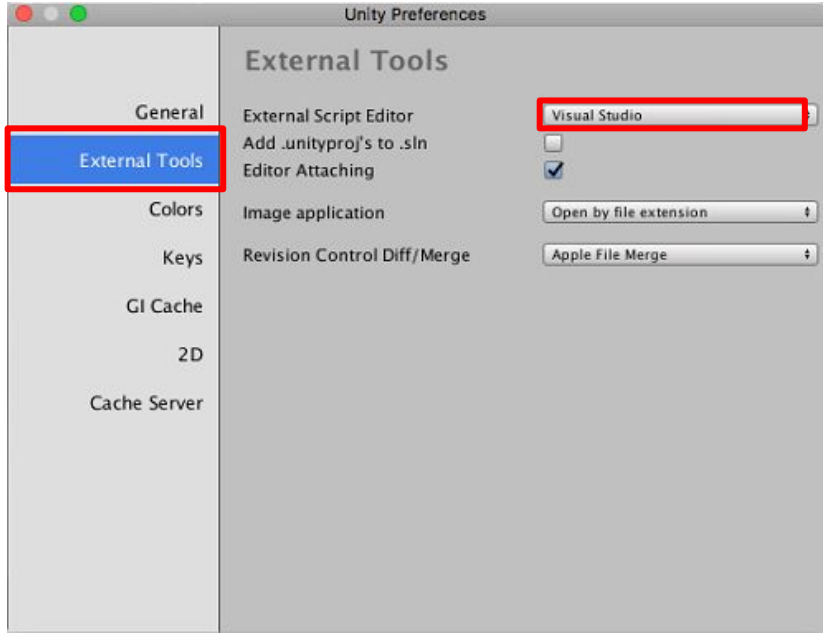




エディタ

2. Visual Studio に設定する。



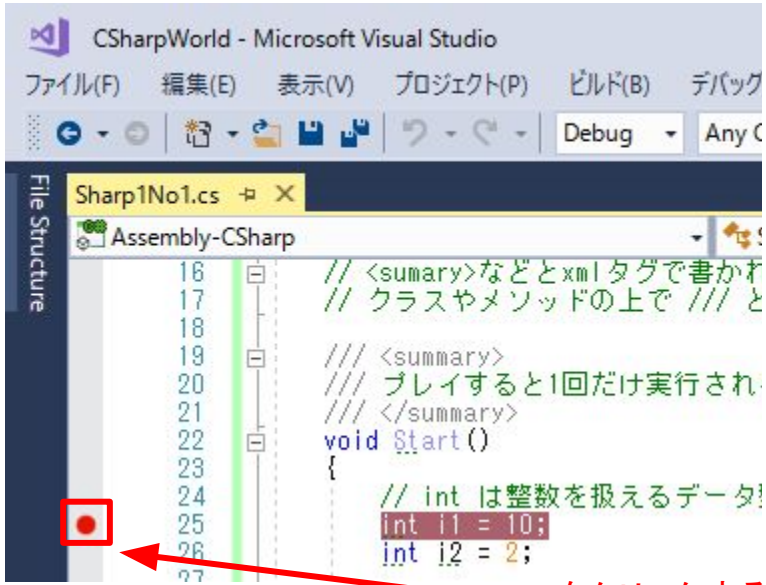


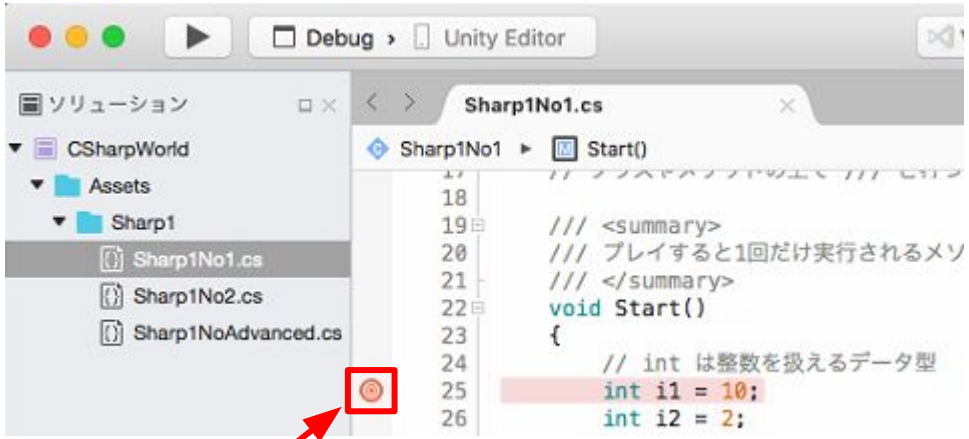
デバッグ

1. シーン と スクリプト(Visual Studio) を開く。

デバッグ

2. ブレークポイントを設定する。

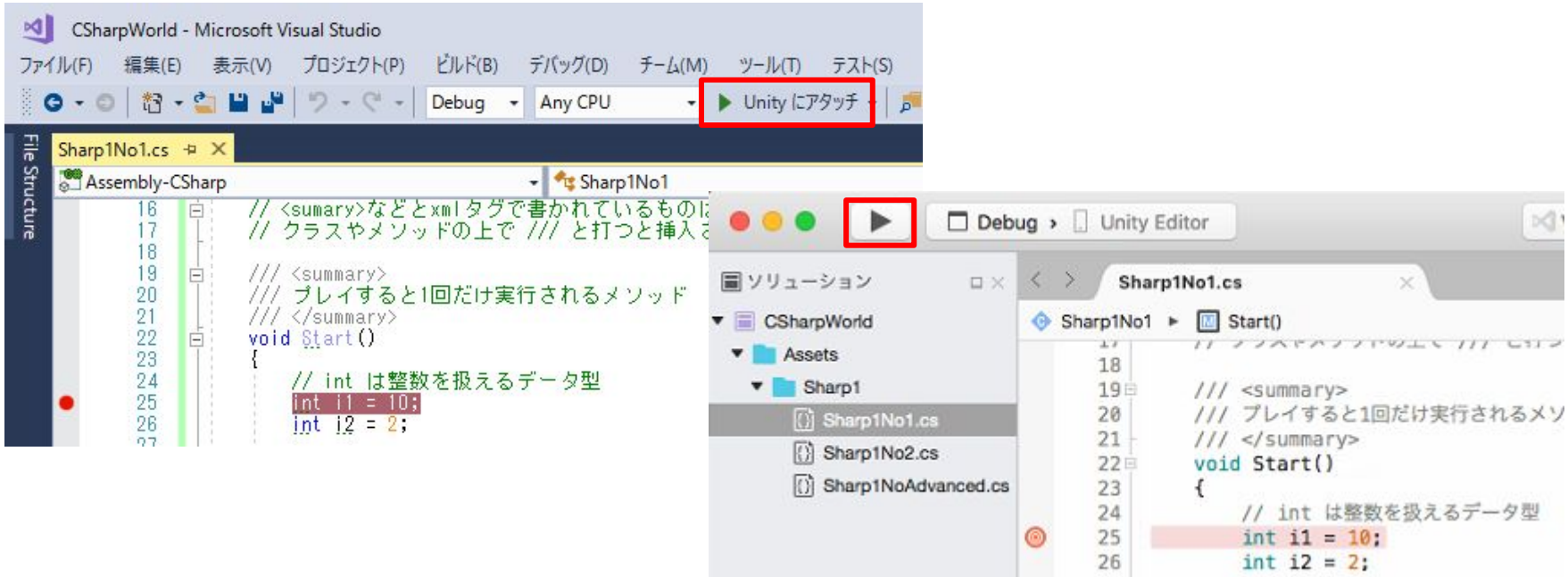




ここをクリックすると、赤くなる。
ここでプログラムを一時停止できる。

デバッグ

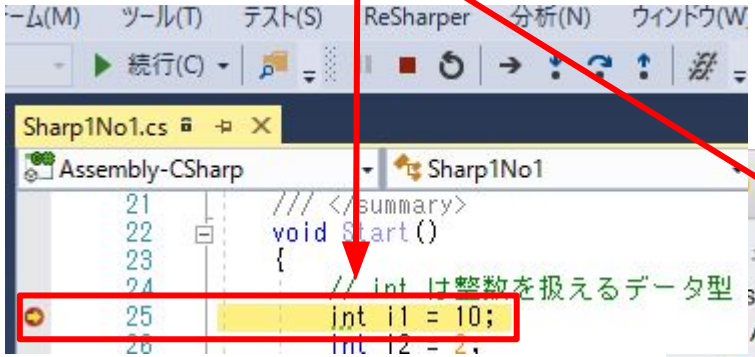
3. Unity にアタッチする。

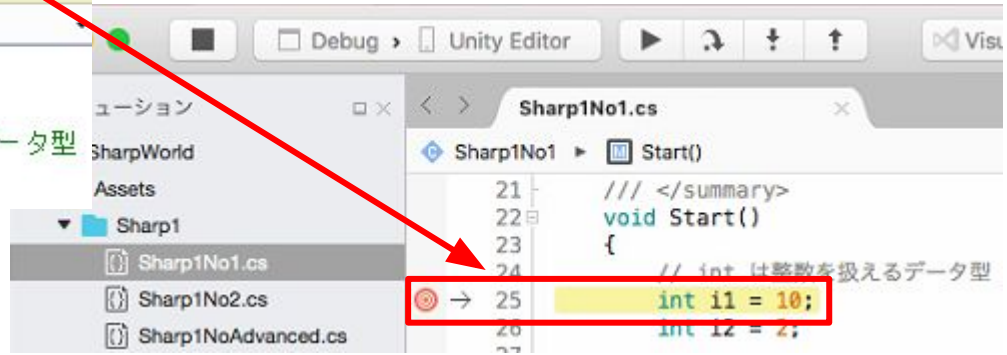


デバッグ

4. Unity をプレイする。

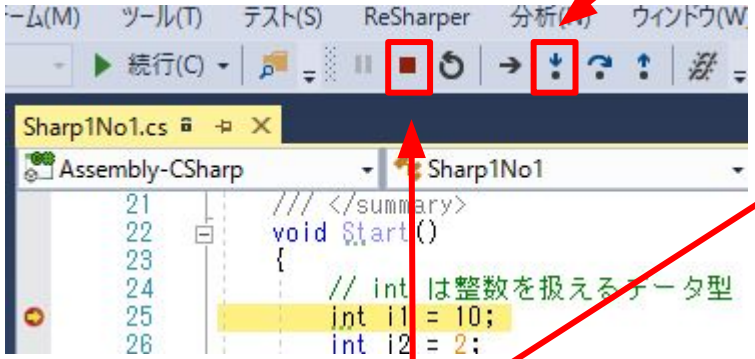
5. ブレークポイントで一時停止される。
（行が黄色になる）

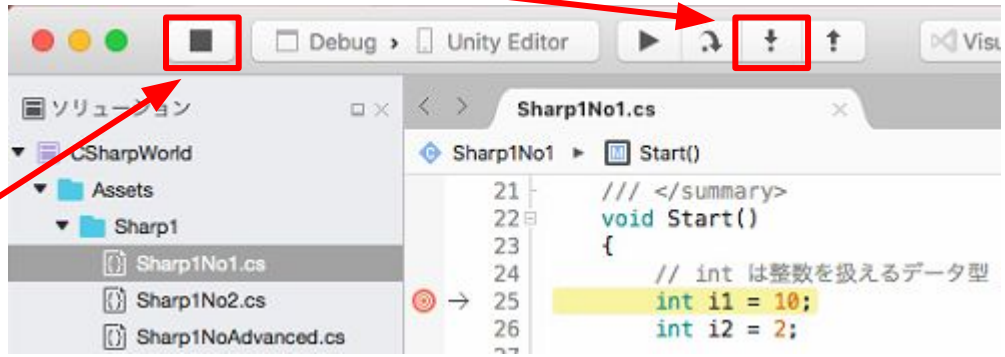




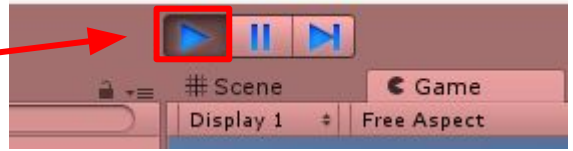
デバッグ

6. ステップインで1命令ずつ進めて確認できる。

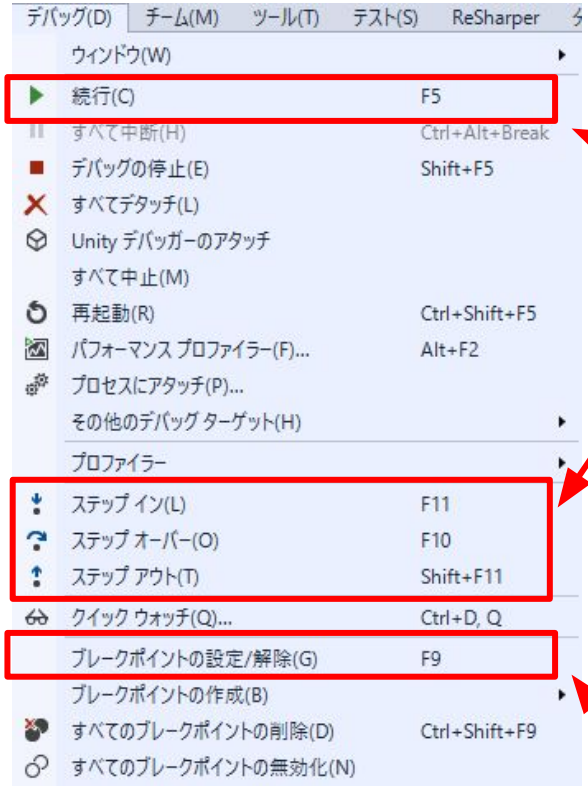




7. デバッグを終了する。
8. その後に Unity のプレイを終了する。



ショートカットキー

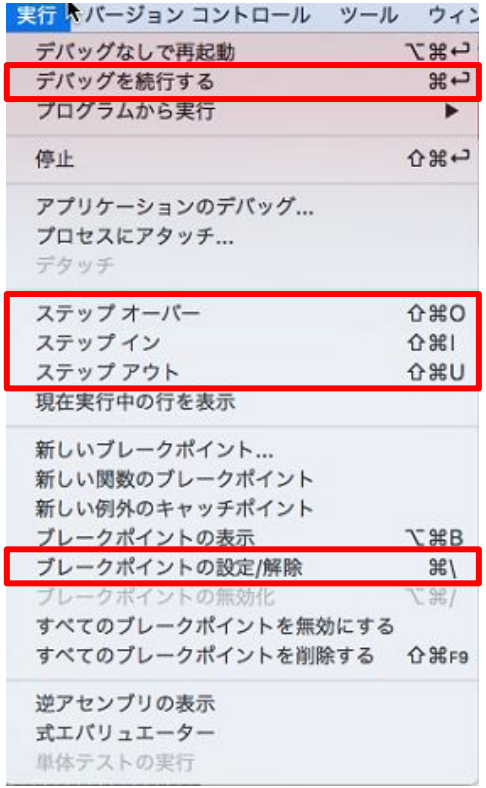


自分の環境で確認！

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。

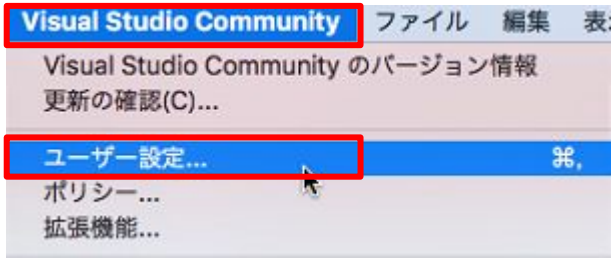
ブレークポイントの設定

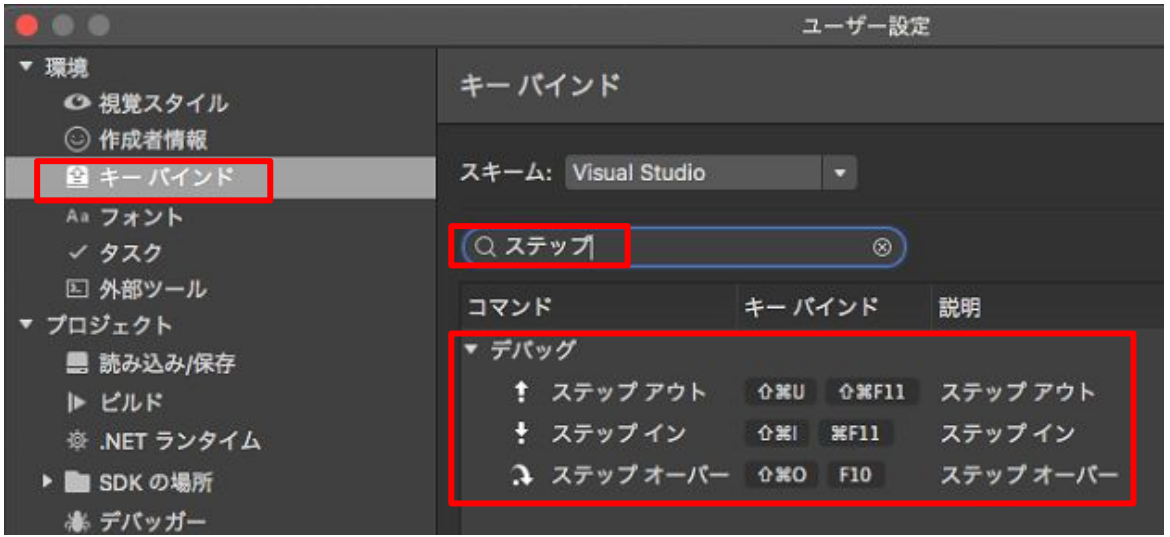


ショートカットキー

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…。





ソースコード1

Sharp1No1シーンを開き、Sharp1No1.csをデバッグ実行する。

1. **コメント** … 実行されないコード。
2. **int** … 整数を扱うという宣言。
3. **変数** … 名前を付けて整数を入れておく。
4. **演算** … 四則演算やインクリメントなど。
5. **メソッド** … 処理のひとかたまり。
6. **戻り値** … メソッドが返す値。
7. **引数** … メソッドに渡す値。
8. **return** … メソッドを抜ける。戻り値を返す。
9. **警告** … 未使用の変数などは解消する。
10. **見やすく** … 途中計算を変数に入れたり、改行する。

ソースコード2

Sharp1No2シーンを開き、Sharp1No2.csをデバッグ実行する。

1. **bool** … 論理値 true か false の2択。
2. **if** … 論理値による条件判定。
3. **&&** … 両方を満たすAND条件。
4. **||** … 片方を満たすOR条件。
5. **演算順序** … 左から右へ、&& が先、|| が後。
6. **? :** … ifを使わない三項演算子。
7. **for** … ループ処理。
8. **contirue** … 次のループへ。
9. **break** … ループを抜ける。
10. **多重ループ** … メソッドを分割して対応する。

ソースコード3

Sharp1NoAdvancedシーンを開き、Sharp1NoAdvanced.csをデバッグ実行する。

1. **System.Int32** … intの正体、intはエイリアス(別名)。
2. **var** … 右辺から型推論。
3. **最適化** … リリースビルドは最適化される。
4. **インクリメント** … 前置と後置。
5. **if else** … { } の省略は注意して。
6. **短絡評価** … & や | は通常使わない。
7. **ifの判定** … boolを渡せば、== trueは不要。
8. **if内の代入** … 条件式内で変数に代入しない。
9. **unchecked** … intは最大値を超えると最小値になる。
10. **checked** … オーバーフロー時に例外になる。

余談

初心者プログラマが犯しがちな過ち25選
(初心者が読むには難しいけど、面白い内容を紹介)

https://qiita.com/rana_kualu/items/379eefb3a40c6b44cb92#24-%E3%82%A8%E3%83%A9%E3%83%BC%E3%82%92%E5%AB%8C%E3%81%86

24) エラーを嫌う … エラーは幸せラッキーヒント。
“プロの開発者はエラーを愛しますが、初心者は嫌います。”
ビルドエラーに苦しむ人へ(過去の自分)。エラーのないバグが辛い。

22) バージョン管理しない … バージョン管理はあなたの力になる。
“初心者はバージョン管理システムの力を知りません。”

#1完了
C#と時の部屋
～メソッドとデバッグ編～

int, bool, if, for は今後もずっとお友達です。
シンプルなコードを心がけて、
怪我(バグ涙)を回避して仲良く遊びましょう！