講義メモ
・p.62「エラーメッセージを読み解こう」から

ミニ演習 mini062

・p.62「エラーが発生しているスクリプト」を試そう

```
using UnityEngine;
public class mini062 : MonoBehaviour{
    void Start(){
        int kazu = 10;
        kazu = kazo * kazu;
        Debug.Log(kazu);
    }
    void Update()  {}
}
```

解説

・名称未発見のような単純な文法エラーであれば、C#(VS)側では日本語で、Unity側でもほぼ同じ意味の英語でエラー内容が示される。
・なお、エラーの位置情報はUnity側で表示されるが、これはエラーの発生場所であり、修正すべき場所がそこであるとは限らない

ミニ演習 mini062・続き

・p.62「エラーが発生しているスクリプト」を修正しよう
・それから、p.63のエラーメッセージが表示されるスクリプトにしよう

```
using UnityEngine;
public class mini062 : MonoBehaviour{
    void Start(){
        int kazu = 10;
        kazu = kazu * kazu;
        Debug.Lag(kazu); //LogをLagと打ち間違えている
    }
    void Update()  {}
}
```

ミニ演習 mini062・続き2

・p.63のエラーメッセージが表示されるスクリプトを修正しよう
・p.33の説明のとおり「using UnityEngine;」を省略(コメントアウト)してみよう

```
//using UnityEngine;
public class mini062 : UnityEngine.MonoBehaviour{
    void Start(){
        int kazu = 10;
        kazu = kazu * kazu;
```

```
        UnityEngine.Debug.Log(kazu); //名前空間指定でクラスのメソッドを呼ぶ
    }
    void Update()  {}
}
```

ミニ演習 mini062・続き3

・p.64のエラーメッセージが表示されるスクリプトにしよう

```
using UnityEngine;
public class mini062 : MonoBehaviour{
    void Start(){
        int kazu = 10.1; //暗黙の型変換ができないのでエラー
        kazu = kazu * kazu;
        Debug.Log(kazu);
    }
    void Update()  {}
}
```

型があっていないときに表示されるエラー

・double型の値や変数をint型の変数に代入すると、小数点以下の情報が失われてしまうので、エラーとなる
※ 言語によってはエラーにならず、小数点以下の情報をカットするものもある
・なお、小数点以下がゼロである実数リテラルであっても、整数と実数の扱いの違いから、同じ意味のエラーになる

ミニ演習 mini062・続き4

・p.64のエラーメッセージが表示されるスクリプトを修正しよう
・int型へのキャストを追記すること

diffツールでスクリプトの間違いをチェックする

・2つのテキストの差異を表示してくれるオンラインツール
 https://www.diffchecker.com/
・なお、オフラインツールがWindowsから提供されている

p.069 trueとfalse

・C#などでは、数値等とは別に真偽値を表現・利用できる仕組みを提供している
 ※ C言語にはないので、代わりに0と非0で表現する
・それがbool型（p.056）で、分岐構文のポイントになっている
・真偽値を示す真偽値リテラルのtrueは「条件に一致」として、falseは「不一致」として扱う
・数値を返す演算子（例：+,-,*,/）や数値を返すメソッド（例：Mathf.Max,Min,Sqrt）と同様に、真偽値を返す演算子やメソッドがある
・これらを分岐の「条件」として利用できる
・真偽値を返す演算子の例が2項<演算子で、左辺が右辺より小さければtrueを、でなければfalseを返す

・真偽値を用いて条件分岐を表すのがif文
・最もシンプルな構文： if(真偽値を返す値や式) { カッコ内がtrueの場合に行う処理 }

p.072 比較演算子

・前述の2項<演算子などを比較演算子といい、値の大小関係によってbool値を返す。
・2項>演算子：左辺が右辺より大きければtrueを、でなければfalseを返す
・2項<=演算子：左辺が右辺以下ならばtrueを、でなければfalseを返す ※「< =」「=<」はエラー
・2項>=演算子：左辺が右辺以上ならばtrueを、でなければfalseを返す ※「> =」「=>」はエラー
・2項==演算子：左辺と右辺が等しければtrueを、でなければfalseを返す ※「= =」はエラー
・2項!=演算子：左辺と右辺が等しくなければtrueを、でなければfalseを返す ※「! =」はエラー
・これらはif文の条件式として用いられることが多いが、単独の演算子としても有効
・bool型の値をDebug.Logで表示すると「True」「False」となる

p.073 Chap2_3_1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Chap2_3_1 : MonoBehaviour {
    void Start() {
        Debug.Log( 4 < 5 ); //trueが返されるので「True」が表示される
    }
    void Update() {

    }
}
```

p.073 Chap2_3_2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Chap2_3_2 : MonoBehaviour {
    void Start() {
        Debug.Log( 6 < 5 ); //falseが返されるので「False」が表示される
    }
    void Update() {

    }
}
```

p.074 public変数を組み合わせてみよう

・public変数の値により、TrueまたはFalseが表示されるようにできる

p.074 Chap2_3_3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Chap2_3_3 : MonoBehaviour {
    public int age; //パブリック変数
    void Start() {
        Debug.Log( age < 20 ); //ageの値が20未満なら「True」でなければFalse」を表
示
    }
    void Update() {

    }
}
```

p.075 文字列を比較する

・C#では、2項==演算子および2項!=演算子で文字列を扱える。
※ 2項==演算子および2項!=演算子を持つ言語が多いが、文字列を扱えないことも多い
(C,Java)

ミニ演習 mini075

・「p.075 文字列を比較する」の4行を実行確認してみよう

```
using UnityEngine;
public class mini075 : MonoBehaviour {
    void Start() {
        Debug.Log("apple" == "apple");  //True
        Debug.Log("apple" == "orange"); //False
        Debug.Log("apple" != "apple");  //False
        Debug.Log("apple" != "orange"); //True
    }
    void Update() { }
}
```

p.076 if文の書き方

・if文は2分岐文で、真偽値を返す値や式を条件文として用いることができる
・基本書式: if(条件文) { 条件文がtrueを返した場合に行う処理 }
・条件文がtrueを返した場合に行う処理は複数記述できるので、その範囲を「{」「}」で示す
・この範囲のことをifのブロックという
  ※Startメソッドの終わりの「}」と見前違いやすいので注意
・条件文がtrueを返した場合に行う処理はif文の内部になるので、一目でわかるように字下げ(インデント)をすると良い
  ※業務上必須であり、Visual Studioがある程度まで自動的に行ってくれる
  ※字下げの方法はチームルールによる(Tabキー、Space4個 など)

p.077 Chap2_4_1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Chap2_4_1 : MonoBehaviour {
    public int age; //パブリック変数
    void Start() {
        if ( age < 20 ) {
            Debug.Log( "未成年" ); //ageの値が20未満なら"未成年"を表示
        }
    }
    void Update() {

    }
}
```

参考:Visual Studioとインデント

・ソース全体のインデントをまとめて揃えるには、Ctrl+A、Ctrl+K、Ctrl+D
・インデントをどう行うかの設定は「ツール」「オプション」で「テキストエディタ」「C#」「タブ」で

p.78 ブロック内で複数の処理を行う

・ifブロックにおいても、ブロック内で複数の処理を行うことができる
・なお、ifブロックの中にさらにifブロックを記述することもできる

p.78 Chap2_4_2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Chap2_4_2 : MonoBehaviour {
    public int age; //パブリック変数
    void Start() {
        if (age < 20) { //ageの値が20未満なら以下を行う
            Debug.Log(age + "歳は"); //ageの値に文字列を連結して表示
            Debug.Log("未成年"); //"未成年"を表示
        }
    }
    void Update() {

    }
}
```

ブロックとフローチャート

・分岐構造を表すフローチャートでは、かならず合流を記述する（例外として合流せずに終了することもあるが）
・この合流点はプログラムではブロックの終わりの直後になる
・よって、これ以降に記述された文は、分岐してもしなくても実行対象になる

アレンジ演習：p.78 Chap2_4_2

・テキストp.079中央のソースのとおりに書き換えよう
・ageの値とは無関係に「ブロック外だよ」が表示されることを確認しよう

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Chap2_4_2 : MonoBehaviour {
    public int age; //パブリック変数
    void Start() {
        if (age < 20) { //ageの値が20未満なら以下を行う
            Debug.Log(age + "歳は"); //ageの値に文字列を連結して表示
            Debug.Log("未成年"); //"未成年"を表示
        }
        Debug.Log("ブロック外だよ"); //分岐とは無関係にを表示
    }
    void Update() {

    }
}
```

p.080 if-else構文

・ifブロックの直後に1つだけelseブロックを追加し、条件式がfalseだった場合に実行したい処理を記述できる
・よって「もしも●なら■を、でなければ▲を」という処理を記述できる
・else文において、条件文がfalseを返した場合に行う処理は複数記述できるので、その範囲を「{」「}」で示す
・if文に所属しないelse文はエラーになる。
・ifブロックの内容とelseブロックの内容のどちらかだけが実行される

p.081 Chap2_5_1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Chap2_5_1 : MonoBehaviour {
    public int age; //パブリック変数
    void Start() {
        if (age < 20) { //ageの値が20未満なら以下を行う
```

```
            Debug.Log("未成年"); //"未成年"を表示
        } else { //ageの値が20未満でない(20以上)なら以下を行う
            Debug.Log("成人"); //"成人"を表示
        }
    }
    void Update() {

    }
}
```

提出 アレンジ演習:ex082

・p.081 Chap2_5_1をアレンジして「"未成年"」または「"成人"」の前に「●歳は」を表示しよう
・また、「"未成年"」または「"成人"」の後に「"です"」を表示しよう
・それぞれの表示は1つずつの記述にすること

次回予告:p.084「else if文」から