講義メモ
・p.104「for文」から

提出フォロー：アレンジ演習：直列ダンジョンへのモンスターの配置作り mini099b

・9匹満室の部屋には「満室」と表示しよう
例：
1号室：●●●●●●●●●満室
2号室：●●
3号室：●●●
4号室：
5号室：●●●●●●●●●満室
6号室：●●●●
ヒント：
・Debug.Logする直前に「nが9なら、sに"満室"を連結する」処理を挿入すれば良い

作成例

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

public class mini099b : MonoBehaviour {
    void Start() {
        System.Random r = new System.Random(); //乱数クラスのオブジェクトを生成
        int a = 1; //1号室から
        while (a <= 6) { //6号室まで繰返す
            string s = a + "号室："; //「■号室：」を用意
            int n = r.Next(10); //モンスター数を0～9にする
            int b = 0; //カウンタを0にする
            while (b < n) { //カウンタがモンスター数未満の間
                s += "●"; //文字列に"●"を連結
                b = b + 1; //次のモンスターへ
            }
            if (n == 9) { //【以下追加】9匹いるなら
                s += "満室"; //文字列の後ろに連結
            }
            Debug.Log(s); //できた文字列を表示
            a = a + 1; //次の部屋へ
        }
    }
    void Update() {

    }
}
```

p.104 for文

・繰返し構文の一つでwhile文の回数指定特化バージョン。

・whileとおなじ前判定繰返しで、for文で書けるものはすべてwhileでも書ける
・回数指定の繰返しをwhile文で行うと、下記のパターン例になる
① カウンタ←0
② while(カウンタ < 回数) {
繰返し内容
③ カウンタに1加算
}
・上記の①②③を1文にするのがfor文
・書式: for(①; ②; ③ { 繰返し内容 }
・パターン例: for (カウンタ←0; カウンタ < 回数; カウンタに1加算) {繰返し内容}
・①では通常、for文の中でしか用いないカウンタ用の変数を用いるので、ここで初期化することが多く、主に「int i = 0;」とする。
・②ではこのカウンタ用の変数iを用いて「i < 回数」とすることが多い
・③でもこのカウンタ用の変数iを用いて「i = i + 1」とする代わりにインクリメント演算子を用いて「i++」とすることが多い
・よって、for文の基本的な形式は「for(int i = 0; i < 回数; i++)」となることが多い。
・ただし、①②③はどれも省略可能であり、上記とは異なる使い方も可能
・whileとおなじ前判定繰返しなので、②が最初から　Falseだと繰返し内容は一度も実行されない

p.105 chap3_3_1

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

public class chap3_3_1 : MonoBehaviour {
    void Start() {
        for (int cnt = 0; cnt < 5; cnt++) { //5回繰り返す
            Debug.Log("ハロー");
        } //forブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

p.105 インクリメント演算子

・複合代入演算子(+=,-=,など)と同様に、省略構文として「● = ● + 1」を「●++」と表記できる
・なお、インクリメント演算子には「●++」と「++●」があり、後置、前置と呼ぶ
・単独で用いる場合、後置、前置は同じ意味だが、式の中などに置くと意味が変わるので注意
・前置「++●」:前もって1加算し、その結果が評価になる。
例: a = 1; Debug.Log(++a); //2と表示し、aの値は2
・前置「●++」:●の値を評価として、その後で1加算。
例: a = 1; Debug.Log(a++); //1と表示し、aの値は2

ミニ演習:mini105

・chap3_3_1をwhile文にしよう

作成例

```
using UnityEngine;
public class mini105 : MonoBehaviour {
    void Start() {
        int cnt = 0; //forの①
        while(cnt < 5) { //forの②
            Debug.Log("ハロー"); //forの繰返し内容
            cnt++; //forの③
        }
    }
    void Update() {
    }
}
```

p.106 繰返し内容においてカウンタの変数を利用する

・for文においてカウンタの変数を初期化した場合、繰返し内容においてカウンタの変数を利用することができる
・これを用いることで「●をカウントアップしながら繰返す」ような処理が簡潔に表せる
・ただし、for文においてカウンタの変数を初期化した場合、その有効範囲はforの中だけなので、繰返しが終わってからカウンタの値を使いたい場合は工夫が必要

p.106 chap3_3_2

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

public class chap3_3_2 : MonoBehaviour {
    void Start() {
        for (int cnt = 0; cnt < 5; cnt++) { //5回繰り返す
            Debug.Log(cnt + "回目のハロー");  //カウンタの値に連結して表示
        } //forブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

アレンジ演習：chap3_3_2 ①

・「Debug.Log(cnt + "回目のハロー");」を改良して「0回目」ではなく「1回目」からにしよう

作成例

```
using System.Collections;
```

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

public class chap3_3_2 : MonoBehaviour {
    void Start() {
        for (int cnt = 0; cnt < 5; cnt++) { //5回繰り返す
            Debug.Log(cnt + 1 + "回目のハロー");  //カウンタの値に加算後連結して
表示
        } //forブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

アレンジ演習:chap3_3_2 ②

・「for (int cnt = 0; cnt < 5; cnt++)」を改良して「0回目」ではなく「1回目」からにしよう

作成例

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

public class chap3_3_2 : MonoBehaviour {
    void Start() {
        for (int cnt = 1; cnt <= 5; cnt++) { //1から5までの5回繰り返す
            Debug.Log(cnt + "回目のハロー");  //カウンタの値に加算後連結して表示
        } //forブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

【補足】インクリメント演算子の演習

・式の中にインクリメント演算子を置いた場合、前置と後置の違いに加えて、途中で変数の値が変わることに注意
・例: `int a = 1, b = a++, c = ++a;`

ミニ演習:mini107a

・上記の例「`int a = 1, b = a++, c = ++a;`」実行後のa、b、cの値を想定してから表示しよう

作成例

```
using UnityEngine;
```

```
public class mini107a : MonoBehaviour {
    void Start() {
        int a = 1, b = a++, c = ++a;
        Debug.Log("a = " + a + " b = " + b + " c = " + c); //a=3 b=1 c=3
となる
    }
    void Update() {
    }
}
```

解説

① a = 1 により、aは1、bは未定、cは未定
② b = a++ により、bがaの1で1、その後aは2になる、cは未定
③ c = ++a により、aは3になる、cがaの3で3に、bは変わらず1

ミニ演習：mini107b

・例「int a = 2, b = (a++) * (a++), c = (++a) * (++a);」実行後のa、b、cの値を想定してから表示しよう

作成例

```
using UnityEngine;
public class mini107a : MonoBehaviour {
    void Start() {
        int a = 2, b = (a++) * (a++), c = (++a) * (++a); ;
        Debug.Log("a = " + a + " b = " + b + " c = " + c); //a=6 b=6 c=30
となる
    }
    void Update() {
    }
}
```

解説

① a = 2 により、aは2、bは未定、cは未定
② b = (a++) * (a++)は、まず、aの値が式に使われるので、式は2 * (a++)になり、aが3になる。
  それから、aの値が式に使われるので、式は2 * 3になり、aが4になる。
  式は2 * 3なので、bは6になる
③ c = (++a) * (++a)は、aが5になり、式は5 * (++a)になる。
  それから、aが6になり、式は5 * 6になる
  式は5 * 6なので、cは30になる

【補足】forのカウンタ用変数の再利用

・通常、forのカウンタ用変数はforの中でのみ有効とすることが望ましいことが多いが、例外的にforの外側でも用いる場合がある

・この場合は、forの前でカウンタ変数を宣言しておき、forの①で初期値を代入すると良い

アレンジ演習：chap3_3_2 ③

・forの繰返しが終わった後の変数cntの値を表示しよう

作成例

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

public class chap3_3_2 : MonoBehaviour {
    void Start() {
        int cnt; //【追加】forの外側でも有効になるように事前に宣言しておく
        for (cnt = 1; cnt <= 5; cnt++) { //【変更】1から5までの5回繰り返す
            Debug.Log(cnt + "回目のハロー");  //カウンタの値に加算後連結して表示
        } //forブロック(繰返し内容)の終わり
        Debug.Log("終了後のカウンタは" + cnt); //【追加】forの外側でも有効(6になる)
    }
    void Update() {

    }
}
```

p.108 逆順で繰返す(カウントダウン)

・forの①で開始値、②で終了値を用いた比較式、③でデクリメントを行うことで、逆順で繰返す(カウントダウン)が可能
・例(5から1まで)： for(int i = 5; i >= 1; i--) {繰返し内容}
・デクリメントは省略構文として「● = ● - 1」を「●--」と表記できる
・デクリメントの仕組みはインクリメントと同じで、前置と後置があることも同じ

p.108 chap3_4_1

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

public class chap3_4_1 : MonoBehaviour {
    void Start() {
        for (int cnt = 5; cnt > 0; cnt--) { //1から5までの5回繰り返す
            Debug.Log(cnt + "回目のハロー");  //カウンタの値に加算後連結して表示
        } //forブロック(繰返し内容)の終わり
    }
    void Update() {

    }
```

}

p.108 10ずつ増やす

・インクリメント、デクリメントの代わりに複合代入演算子(+=、-=)を用いることで、nずつ増やす(減らす)ことができる
・例: for(int i = 0; i < 50; i += 10) {…} //0, 10, 20, 30, 40で5回繰返す

ミニ演習:mini108a

・上の例「for(int i = 0; i < 50; i += 10) {…} //0, 10, 20, 30, 40で5回繰返す」を試してみよう

作成例

```
using UnityEngine;
public class mini108a : MonoBehaviour {
    void Start() {
        for (int i = 0; i < 50; i += 10) { //0, 10, 20, 30, 40で5回繰返す
            Debug.Log("i = " + i);
        }
            }
    void Update() {
    }
}
```

ミニ演習:mini108b

・パブリック変数で整数xを受け取り、2からその値以下の全偶数を表示しよう
・例: xが9なら2,4,6,8。xが10なら2,4,6,8,10
・なお、xが2なら2のみになり、xが1なら何も表示しなくて良い

作成例

```
using UnityEngine;
public class mini108b : MonoBehaviour {
    public int x; //パブリック変数
    void Start() {
        for (int i = 2; i <= x; i += 2) { //2,4,6,8,…でx以下について繰返す
            Debug.Log("i = " + i);
        }
    }
    void Update() {
    }
}
```

ミニ演習:mini108c

・パブリック変数で整数xを受け取り、その値以下の全奇数を大きい順に表示しよう

・例： xが9なら9,7,5,3,1。xが10でも9,7,5,3,1
・なお、xが2または1なら1のみになり、xが0なら何も表示しなくて良い
・ヒント：forの①で用いる初期値を、if文などで先に決める必要がある。
　xが奇数(2で割った剰余が1)ならば、そのまま、でなれば-1した値をforの①の初期値にすると良い。
　forの②の継続条件は「i >= 1」、forの③は「i -= 2」となる

作成例

```
using UnityEngine;
public class mini108c : MonoBehaviour {
    public int x; //パブリック変数
    void Start() {
        int s; //スタートの値
        if (x % 2 == 1) { //入力値が奇数？
            s = x; //その値からスタート
        } else { //偶数？
            s = x - 1; //その値-1の奇数からスタート
        }
        for (int i = s; i >= 1; i -= 2) { //x以下の全奇数について降順で繰返す
            Debug.Log("i = " + i);
        }
    }
    void Update() {
    }
}
```

提出：ミニ演習：mini108cの改良

・条件演算子を用いてシンプルにしよう
・スタートの値は「(x % 2 == 1) ? x : x - 1」となる

次回(1/14)予告：p.109「繰返しからの脱出とスキップ」から