講義メモ

・p.109「繰返しからの脱出とスキップ」から

提出フォロー：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

```
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() {
    }
}
```

作成例

```
using UnityEngine;
public class mini108c : MonoBehaviour {
    public int x; //パブリック変数
    void Start() {
        int s = (x % 2 == 1) ? x : x - 1; //開始値を決める(偶数なら-1した値にす
る)
        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() {
    }
}
```

p.109 繰返しからの脱出とスキップ

・while/do-while/forのブロックの中で継続条件とは無関係に繰返しを脱出したい場合は「break;」を実行すれば良い
・実際には、while/do-while/forのブロックの終わりの「}」の次の行にジャンプする
・よって、次の行がなければ終了する
・なお、break;の実行により「絶対に実行されない行」ができてしまうとエラーまたは警告が出る

ミニ演習：mini109

・int型のパブリック変数aで開始値、bで終了値を得て、aからbまでカウントダウンする
・ただし、カウントが1桁になったら表示せずに終了しよう
・よって、aが12でbが 8なら、12,11,10を表示
・よって、aが14でbが11なら、14,13,12,11を表示

・よって、aが9でbが1なら、なにも表示しない

作成例

```
using UnityEngine;
public class mini109 : MonoBehaviour {
    public int a, b; //パブリック変数
    void Start() {
        for (int i = a; i >= b; i--) { //aからbまでカウントダウン
            if (i < 10) { //１桁になったら
                break; //繰返しを抜ける
            }
            Debug.Log("i = " + i);
        }
    }
    void Update() {
    }
}
```

補足：

・この例はforの継続条件を工夫することで、ifとbreakを無しにすることも可能だが、プログラムが読みづらくなるので、ケースバイケースで。

```
using UnityEngine;
public class mini109 : MonoBehaviour {
    public int a, b; //パブリック変数
    void Start() {
        for (int i = a; i >= 10 && i >= b; i--) { //aからbまで(かつ10まで)カウントダウン
            Debug.Log("i = " + i);
        }
    }
    void Update() {
    }
}
```

p.109 繰返しからの脱出とスキップ（つづき）

・while/do-while/forのブロックの中で現在の繰返し内容の残りをスキップしたい場合は「continue;」を実行すれば良い
・「次いこう、次」のイメージで、繰返しは終了しない
・例：1から100までのうち、3の倍数以外を表示 ⇒ 3の倍数の時はcontinueする
※ 条件が単純な場合は、ifで代用できるので、複雑な場合に用いると良い

ミニ演習：mini109a

・int型のパブリック変数aで開始値、bで終了値を得て、aからbまでカウントダウンする
・ただし、3の倍数なら表示せずに次に進もう
・よって、aが12でbが 8なら、11,10,8を表示
・よって、aが14でbが11なら、14,13,11を表示
・よって、aが12でbが11なら、11のみを表示

作成例

```
using UnityEngine;
public class mini109a : MonoBehaviour {
    public int a, b; //パブリック変数
```

```
    void Start() {
        for (int i = a; i >= b; i--) { //aからbまでカウントダウン
            if (i % 3 == 0) { //３の倍数ならば
                continue; //スキップして次へ
            }
            Debug.Log("i = " + i);
        }
    }
    void Update() {
    }
}
```

補足：breakとcontinue

・どちらも、アルゴリズムを工夫することで、用いずに表現できることが多い
・しかし、用いることで、ネストを減らす効果もあるので、可読性で判断すると良い

p.110 ２重ループ

・繰返し文の中に繰返し文を書くことで、２重ループが可能になる
・内側の繰返しを、外側の繰返し回数だけ繰返すので、動きに注意
・for文の２重ループはマップ処理などに便利。例：縦３回分横４回分繰返す
・for文の２重ループの場合、カウンタに用いる変数の使い分けが必要で、外側からi、jとすることが多い
・例：

```
for (int i = 1; i <= 3; i++) { //縦３回分繰返す
  for (int j = 1; j <= 4; j++) { //横４回分繰返す
    繰返し内容
  }
}
```

これは①①、①②、①③、①④、②①、②②、②③、②④、③①、③②、③③、③④の順になる

p.110 chap3_5_1

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

public class chap3_5_1 : MonoBehaviour {
    void Start() {
        for (int x = 1; x < 10; x++) { //xを1から9までの9回繰り返す
            for (int y = 1; y < 10; y++) { //yを1から9までの9回繰り返す
                Debug.Log(x * y);  //カウンタの積を表示
            } //内側のforブロック(繰返し内容)の終わり
        } //外側のforブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

p.113 chap3_5_2に関して

・forによる２重ループの場合、内側の繰返しでは、２つのカウンタの両方を利用できる

p.113 chap3_5_2

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

public class chap3_5_2 : MonoBehaviour {
    void Start() {
        for (int x = 1; x < 10; x++) { //xを1から9までの9回繰り返す
            for (int y = 1; y < 10; y++) { //yを1から9までの9回繰り返す
                Debug.Log(x + "×" + y + "＝" + x * y); //式と積を表示
            } //内側のforブロック(繰返し内容)の終わり
        } //外側のforブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

アレンジ演習：chap3_5_2a

・式と積を表示する代わりに、その段の積を並べて１度に表示しよう
【出力結果】
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
：（略）
9 18 27 36 45 54 63 72 81
ヒント：
・内側の繰返しの前に、string型変数sを""(空)で初期化する
・内側の繰返しの中で、xとyの積と" "(空白)をsに連結する（※ここでは出力しない）
・内側の繰返しの後で、sを出力する

作成例

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

public class chap3_5_2a : MonoBehaviour {
    void Start() {
        for (int x = 1; x < 10; x++) { //1の段から9の段まで繰り返す
            string s = ""; //連結用の文字列を用意
            for (int y = 1; y < 10; y++) { //×1から×9まで繰り返す
                s = s + x * y + " "; //文字列に積と空白を連結
            } //内側のforブロック(繰返し内容)の終わり
            Debug.Log(s); //出来上がった文字列を表示
        } //外側のforブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

アレンジ演習：chap3_5_2b

・2×3と3×2は同じなので、九九表の右上半分は省略しよう
【出力結果】
1
2 4
3 6 9

```
4 8 12 16
5 10 15 20 25
6 12 18 24 30 36
7 14 21 28 35 42 49
8 16 24 32 40 48 56 64
9 18 27 36 45 54 63 72 81
```

ヒント：

・内側の繰返しの中で、xがy以上の時のみ連結すれば良い

作成例①：if文を用いる

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

public class chap3_5_2b : MonoBehaviour {
    void Start() {
        for (int x = 1; x < 10; x++) { //1の段から9の段まで繰り返す
            string s = ""; //連結用の文字列を用意
            for (int y = 1; y < 10; y++) { //×1から×9まで繰り返す
                if (x >= y) {
                    s = s + x * y + " "; //文字列に積と空白を連結
                }
            } //内側のforブロック(繰返し内容)の終わり
            Debug.Log(s); //出来上がった文字列を表示
        } //外側のforブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

作成例②：内側のforの継続条件を書き換える

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

public class chap3_5_2b : MonoBehaviour {
    void Start() {
        for (int x = 1; x < 10; x++) { //1の段から9の段まで繰り返す
            string s = ""; //連結用の文字列を用意
            for (int y = 1; y <= x; y++) { //1倍からx倍まで繰り返す
                s = s + x * y + " "; //文字列に積と空白を連結
            } //内側のforブロック(繰返し内容)の終わり
            Debug.Log(s); //出来上がった文字列を表示
        } //外側のforブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

補足：２重ループとbreakについて

・２重ループの内側の繰返しの中でbreakすると、内側の繰返しからの脱出になり、外側の繰返しは続行されるので注意

アレンジ演習：chap3_5_2c

・内側の繰返しで積が50以上になったら脱出し、次の段に進むようにしよう
【出力結果】
```
1
2 4
3 6 9
4 8 12 16
5 10 15 20 25
6 12 18 24 30 36
7 14 21 28 35 42 49
8 16 24 32 40 48
9 18 27 36 45
```

作成例

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

public class chap3_5_2c : MonoBehaviour {
    void Start() {
        for (int x = 1; x < 10; x++) { //1の段から9の段まで繰り返す
            string s = ""; //連結用の文字列を用意
            for (int y = 1; y <= x; y++) { //1倍からx倍まで繰り返す
                if (x * y >= 50) { //積が50以上なら
                    break; //内側の繰返しを抜ける
                }
                s = s + x * y + " "; //文字列に積と空白を連結
            } //内側のforブロック(繰返し内容)の終わり
            Debug.Log(s); //出来上がった文字列を表示
        } //外側のforブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

補足：２重ループとbreakについて（続き）

・２重ループの内側の繰返しの中でbreakすると、内側の繰返しからの脱出になり、外側の繰返しは続行されるので注意
・外側の繰返しも脱出したい場合は、外側用のbreakを記述する

アレンジ演習：chap3_5_2d

・内側の繰返しで積が50以上になったら脱出し、その段を出力して終わる（次の段に進まない）ようにしよう
【出力結果】
```
1
2 4
3 6 9
4 8 12 16
5 10 15 20 25
6 12 18 24 30 36
7 14 21 28 35 42 49
8 16 24 32 40 48
```

作成例

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

public class chap3_5_2d : MonoBehaviour {
    void Start() {
        for (int x = 1; x < 10; x++) { //1の段から9の段まで繰り返す
            bool endf = false; //【追加】終了フラグをオフにしておく
            string s = ""; //連結用の文字列を用意
            for (int y = 1; y <= x; y++) { //1倍からx倍まで繰り返す
                if (x * y >= 50) { //積が50以上なら
                    endf = true; //【追加】終了フラグをオンにする
                    break; //内側の繰返しを抜ける
                }
                s = s + x * y + " "; //文字列に積と空白を連結
            } //内側のforブロック(繰返し内容)の終わり
            Debug.Log(s); //出来上がった文字列を表示
            if (endf) { //【以下追加】終了フラグがオン？
                break; //外側の繰返しも抜ける
            }
        } //外側のforブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

補足：２重ループによる図形処理

・棒グラフ風の表現を２重ループで行うことができる
例：
1:■
2:■■
3:■■■
4:■■■■
5:■■■■■

アレンジ演習：chap3_5_2e

・int型のパブリック変数aで開始値、bで終了値を得て、aからbまでカウントダウンする棒グラフを表示しよう
例：aが7でbが4なら
7:■■■■■■■
6:■■■■■■
5:■■■■■
4:■■■■
ヒント
① 外側でカウンタiをaからbまでデクリメントしながら②から⑤を繰返す
② 連結用の文字列sをiと":"を連結した文字列で初期化
③ 内側でカウンタjを0からi未満までインクリメントしながら④を繰返す
④ 文字列sに"■"を連結
⑤ 内側の繰返しの後で、文字列sを出力

作成例

```
using System.Collections;
```

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

public class chap3_5_2e : MonoBehaviour {
    public int a, b;
    void Start() {
        for (int i = a; i >= b; i--) { //カウンタiをaからbまでデクリメントしながら繰返す
            string s = i + ":"; //連結用の文字列を用意
            for (int j = 0; j < i; j++) { //i回繰り返す
                s += "■"; //文字列に■を連結
            } //内側のforブロック(繰返し内容)の終わり
            Debug.Log(s); //出来上がった文字列を表示
        } //外側のforブロック(繰返し内容)の終わり
    }
    void Update() {

    }
}
```

アレンジ演習：chap3_5_2f

・乱数で1から9を5回得て、５本の横棒グラフを表示しよう
例：
7:■■■■■■■
2:■■
6:■■■■■■
5:■■■■■
4:■■■■
ヒント：1から9の乱数を得るには
```
  System.Random r = new System.Random(); //乱数クラスのオブジェクトを生成(１度でOK)
   :
  int n = r.Next(9) + 1; //1から9までのどれかになる
```

提出：アレンジ演習：chap3_5_2f

次回予告：p.114「配列の書き方を覚えよう」