講義メモ
・p.084「else if文」から

提出フォロー　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

作成例

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

public class ex082 : MonoBehaviour {
    public int age; //パブリック変数
    void Start() {
        Debug.Log(age + "歳は"); //【追加】ageの値に連結して表示
        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
        }
        Debug.Log("です"); //【追加】
    }
    void Update() {

    }
}
```

アレンジ演習：ex082・改

・表示を連結して1回の記述にしよう
・文字列型の変数を用いると良い
・Debug.Logをif-elseを抜けてから行うこと

作成例

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

public class ex082 : MonoBehaviour {
    public int age; //パブリック変数
    void Start() {
        string str = age + "歳は"; //【変更】ageの値に連結して格納
        if (age < 20) { //ageの値が20未満なら以下を行う
            str = str + "未成年"; //【変更】"未成年"を連結
```

```
        } else { //ageの値が20未満でない(20以上)なら以下を行う
            str = str + "成人"; //【変更】"成人"を連結
        }
        Debug.Log(str + "です"); //【変更】"です"を連結して表示
    }
    void Update() {

    }
}
```

p.084 else if文

・多分岐構造を実現する構文がif-else if構文で、else ifは「ではなくて、もしも」を意味する
・else ifはifの後であればいくつでも記述できる(ifの前は不可)
・また、elseの上に記述できる(elseの後は不可)
・書式:

```
 if(条件式①) {
  ①がtrueの場合の処理
 } else if(条件式②) {
  ①がfalseで②がtrueの場合の処理
 } else {
  ①も②もfalseの場合の処理
 }
```

・なお、ifの条件式①に含まれるような条件②にすると、else ifの内容は実行されない

p.085 Chap2_6_1

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

public class Chap2_6_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 if (age < 65) { //ageの値が20未満ではなく65未満なら以下を行う
            Debug.Log("成人"); //"成人"を表示
        } else { //どちらでもないなら(65以上なら)以下を行う
            Debug.Log("高齢者"); //"高齢者"を表示
        }
    }
    void Update() {

    }
}
```

p.087 else if文で起こりやすいミス

・if-else if-else構文で書区べきプログラムで、誤って「else if」を「if」にすると、文法エラーにならず、実行も分岐によっては変わらないのでミスに気づきづらいので注意
・下の例では、20以上や65以上での処理は変わらないが、20未満の時のみ誤動作する（未成年と成人の両方が表示されてしまう）

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

public class Chap2_6_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
        } if (age < 65) { //ageの値が20未満ではなく65未満なら以下を行う【誤り】
            Debug.Log("成人"); //"成人"を表示
        } else { //どちらでもないなら(65以上なら)以下を行う
            Debug.Log("高齢者"); //"高齢者"を表示
        }
    }
    void Update() {

    }
}
```

・これは内部的には2つのif文になっているので、下記が動作していると考えると良い

```
        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
        } //【ここで一度切れている】
        if (age < 65) { //ageの値が20未満ではなく65未満なら以下を行う【誤り】
            Debug.Log("成人"); //"成人"を表示
        } else { //どちらでもないなら(65以上なら)以下を行う
            Debug.Log("高齢者"); //"高齢者"を表示
        }
```

アレンジ演習：p.085 Chap2_6_1

・4分岐にして、6歳未満の場合「幼児」と表示しよう。

作成例

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

public class Chap2_6_1 : MonoBehaviour {
    public int age; //パブリック変数
```

```
    void Start() {
        if (age < 6) { //【追加】ageの値が6未満なら以下を行う
            Debug.Log("幼児"); //【追加】"幼児"を表示
        } else if (age < 20) { //【変更】ageの値が6未満ではなく20未満なら以下を行
う
            Debug.Log("未成年"); //"未成年"を表示
        } else if (age < 65) { //ageの値が20未満ではなく65未満なら以下を行う
            Debug.Log("成人"); //"成人"を表示
        } else { //どちらでもないなら(65以上なら)以下を行う
            Debug.Log("高齢者"); //"高齢者"を表示
        }
    }
    void Update() {

    }
}
```

p.088 論理演算子(2項&&演算子)

・2項&&演算子：左辺と右辺がbool型(p.56)であれば、bool型を返す。
　どちらもtrueならtrueを、でなければfalseを返す。よってアンド(かつ)と呼ばれることがある
・if文の条件式に用いると「if(a > 0 && b > 0)」なら「aは正の数、かつ、bも正の数」と意味する
・論理演算の用語では「論理積」ともいう
・この演算子は、2つの不等式を両辺に置くことで、値の範囲をチェックするときに便利
　例：if(a >= 1 && a <= 10) { //aが1以上10以下であれば
・ただし、同時に成立しない式にせぬよう注意
　例：if(a >= 11 && a <= 10) { //aが11以上10以下であれば ←ありえない

p.089 Chap2_7_1

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

public class Chap2_7_1 : MonoBehaviour {
    public int age; //パブリック変数
    void Start() {
        if (age >= 6 && age <= 15) { //ageの値が6以上かつ15以下なら
            Debug.Log("義務教育の対象");
        }
    }
    void Update() {

    }
}
```

【補足】複数の&&演算子

・2つ以上の&&演算子を並べて記述できる
例: a > 0 && b > 0 && c > 0
・この場合、最初の&&でtrueにならなければ、それ以降は無視される
・よって、この構文の後半以降にミスがあると発覚しづらいので注意

【補足】&&演算子と短絡評価

・1つの&&演算子においても「結果が見えているときは無駄なことはしない」という機能が働く
・つまり、&&演算子は左辺がfalseだと、右辺がtrueでもfalseでもfalse確定なので、右辺は見なくなる
・このことを短絡評価(ショートサーキット)という
・これを活用して、プログラムの異常終了を防止できるテクニックがある
例: if(a != 0 && 5 / a > 2) //aが0だと右辺が無視されるので、ゼロ除算による異常終了を防ぐ

p.090 論理演算子(2項||演算子)

・2項||演算子:左辺と右辺がbool型(p.56)であれば、bool型を返す。
どちらもfalseならfalseを、でなければtrueを返す。よってオア(または)と呼ばれることがある
・if文の条件式に用いると「if(a > 0 || b > 0)」なら「aは正の数、または、bは正の数」と意味する
・論理演算の用語では「論理和」ともいう
・この演算子は、2つの不等式を両辺に置くことで、値の範囲外をチェックするときに便利
例: if(a < 1 || a >= 10) { //aが1未満または10以上であれば
・ただし、同時に成立する式にせぬよう注意
例: if(a >= 1 || a <= 10) { //aが1以上または10以下であれば ←全部になり意味がない

p.090 Chap2_7_2

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

public class Chap2_7_2 : MonoBehaviour {
    public int age; //パブリック変数
    void Start() {
        if (age <= 5 || age >= 65) { //ageの値が5以下または65以上なら
            Debug.Log("幼児と高齢者");
        }
    }
    void Update() {

    }
}
```

【補足】複数の||演算子

・2つ以上の||演算子を並べて記述できる

例: a > 0 || b > 0 || c > 0
・この場合、最初の||でtrueにならなければ、それ以降は無視される
・よって、この構文の後半以降にミスがあると発覚しづらいので注意

【補足】||演算子と短絡評価

・1つの||演算子においても「結果が見えているときは無駄なことはしない」という機能が働く
・つまり、||演算子は左辺がtrueだと、右辺がtrueでもfalseでもtrue確定なので、右辺は見なくなる
・このことも短絡評価(ショートサーキット)という
・これを活用して、プログラムの異常終了を防止できるテクニックがある
例: if(a == 0 || 5 / a > 2) //aが0だと右辺が無視されるので、ゼロ除算による異常終了を防ぐ

【補足】&&演算子と||演算子の併用と優先順位

・&&演算子と||演算子の併用が可能だが、優先順位に注意
・&&演算子の方が||演算子より優先される
例: if(a > 0 || b > 0 && c > 0) //aが正の数または、bもcも正の数

p.091 単項!演算子

・右辺がbool型(p.56)であれば、反転したbool型を返す。
つまりtrue⇔falseの反転が可能。よってナット(否定)と呼ばれることがある
・主に、複雑な条件を反転したほうがわかりやすい場合や、bool型を返すメソッドを呼んで結果を反転したい場合に用いる
・例: if(!(a == 0 && b == 0 && c == 0)) //「abcがすべてゼロ」じゃなければ
ちなみに、これは if(a != 0 || b != 0 || c != 0)) と同じだが、わかりやすい方を用いると良い
・なお、単純にbool型の変数の値をtrue⇔falseの反転をしたい場合にも用いる
例: bool a = true, b = !a; //bにはaのtrueを反転したfalseが入る

p.091 Chap2_7_3

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

public class Chap2_7_3 : MonoBehaviour {
    void Start() {
        if (!Mathf.IsPowerOfTwo(9)) { //「9が2のべき乗なら」の否定
            Debug.Log("2のべき乗ではない");
        }
    }
    void Update() {

    }
}
```

アレンジ演習：p.091 Chap2_7_3

・9を固定ではパブリック変数にして、外部から与えられるようにしよう

作成例

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

public class Chap2_7_3 : MonoBehaviour {
    public int n; //【追加】パブリック変数「
    void Start() {
        if (!Mathf.IsPowerOfTwo(n)) { //【変更】「nが2のべき乗なら」の否定
            Debug.Log("2のべき乗ではない");
        }
    }
    void Update() {

    }
}
```

【補足】!演算子と&&演算子と||演算子の併用と優先順位

・!演算子と&&演算子と||演算子の併用が可能だが、優先順位に注意
・!演算子と、&&演算子、||演算子の順に優先される

【補足】!演算子と&&演算子と||演算子の併用とドモルガンの法則

・「論理積の式全体を否定すると、それぞれの否定の論理和になる」という法則
・例：!(a > 0 && b > 0) ⇒ !(a > 0) || !(b > 0) //どちらも「aもbも正の数」の否定
・また「論理和の式全体を否定すると、それぞれの否定の論理積になる」
・例：!(a > 0 || b > 0) ⇒ !(a > 0) && !(b > 0) //どちらも「aかbが正の数」の否定
・よって、否定の否定を含むような複雑な式は、この法則を用いて整理すると見やすくなることがある

【補足】条件演算子（p.039）

・if-else構文による判断結果の代入を、3項演算子で表現できる仕掛け
・例： int a = 2, b = 3, max = (a > b) ? a : b; //a > b ならaをでなければbを代入
・書式：（条件式などのbool型の式） ? 式がtrueの時の値 : 式がfalseの時の値
・これを用いると、p,081 Chap2_5_1のif-elseの5行が1行になる
 → Debug.Log((age < 20) ? "未成年" : "成人");
・なお「:」以降の省略は不可（if文でいうifのみ記述は不可）

提出：ミニ演習 mini091

・上記の通り、p,081 Chap2_5_1を書き換えてみよう

次回予告：条件演算子の補足をしてから、p.092「エラーメッセージを読み解こう②」に進みます