講義メモ
・p.137「２次元配列」から

提出フォロー：アレンジ演習：p.136 average02.cs

・配列の3要素の値をコンソールから入力するようにしよう
・ヒント① 配列の初期化ではなく宣言と生成にする
 例：int[] point = new int[3];
・ヒント② コンソールからの入力は繰返しの中（sumに足し込む前）で行うと良い
・ヒント③ 配列の要素は変数と同様に扱えるので、コンソールからの入力を要素に代入する
 例：Console.Write("{0}番：", i); point[i] = int.Parse(Console.ReadLine());

作成例

```
//アレンジ演習：p.136 average02.cs
using System;
class average02 {
    public static void Main() {
        int[] point = new int[3]; //【変更】int型の配列pointを3要素で生成
        int sum = 0, no; //合計、要素数
        no = point.Length; //プロパティで配列の要素数を得る
        for (int i = 0; i < no; i++) { //iを0から要素数未満まで＝全要素について繰返す
            Console.Write("{0}番：", i); point[i] =
int.Parse(Console.ReadLine());
            sum += point[i]; //i番の要素の値をsumに足し込む
        }
        double average = (double)sum / no; //合計を件数で割って平均値を得る
        Console.WriteLine("合計 = {0}, 平均 = {1:##.#}", sum, average); //合計値
と平均値（小数点以下1桁）を表示
    }
}
```

アレンジ演習：p.136 average02.cs・続き

・入力された値を逆順にしよう
例：10, 20, 30ならば30, 20, 10にする
・２つ方法があり、どちらでもOK
 ①入力時に末尾の要素から格納
 ②全て格納してから反転する

作成例①

```
//アレンジ演習：p.136 average02.cs・続き ①入力時に末尾の要素から格納
using System;
class average02 {
    public static void Main() {
        int[] point = new int[3]; //int型の配列pointを3要素で生成
        int sum = 0, no; //合計、要素数
        no = point.Length; //プロパティで配列の要素数を得る
        for (int i = no - 1; i >= 0; i--) { //【変更】iを末尾から0まで＝全要素について繰
返す
            Console.Write("{0}番：", i); point[i] =
int.Parse(Console.ReadLine());
            sum += point[i]; //i番の要素の値をsumに足し込む
```

```
        }
        double average = (double)sum / no; //合計を件数で割って平均値を得る
        Console.WriteLine("合計 = {0}, 平均 = {1:##.#}", sum, average); //合計値
と平均値(小数点以下1桁)を表示
        for (int i = 0; i < no; i++) { //【以下追加】iを0から要素数未満まで＝全要素につ
いて繰返す
            Console.WriteLine("{0}番:{1}", i, point[i]);
        }
    }
}
```

作成例②

```
//アレンジ演習:p.136 average02.cs・続き ②全て格納してから反転
using System;
class average02 {
    public static void Main() {
        int[] point = new int[3]; //int型の配列pointを3要素で生成
        int sum = 0, no; //合計、要素数
        no = point.Length; //プロパティで配列の要素数を得る
        for (int i = 0; i < no; i++) { //iを0から要素数未満まで＝全要素について繰返す
            Console.Write("{0}番:", i); point[i] =
int.Parse(Console.ReadLine());
            sum += point[i]; //i番の要素の値をsumに足し込む
        }
        double average = (double)sum / no; //合計を件数で割って平均値を得る
        Console.WriteLine("合計 = {0}, 平均 = {1:##.#}", sum, average); //合計値
と平均値(小数点以下1桁)を表示
        int work = point[0]; point[0] = point[2]; point[2] = work; //【以下追加】要
素[0]と[2]の値を交換する
        for (int i = 0; i < no; i++) { //iを0から要素数未満まで＝全要素について繰返す
            Console.WriteLine("{0}番:{1}", i, point[i]);
        }
    }
}
```

p.137 2次元配列

・添字を2つ以上持つ配列を多次元配列といい、2つの場合は2次元配列
・ゲームのマップや行列型の情報の扱いに向いている
・C#には多次元配列の実装方法が2種類あり、他の言語とは異なるので注意
・配列に格納されるデータを要素という
・(通常型の)2次元配列は2つの添え字の数の積の要素が格納できる
・例:添字①が2、添字②が5であれば2×5＝10要素分になる(p.137 表6.1)
・配列は変数とは異なり、配列名の宣言の後で、要素の生成を行うと利用可能になる
・宣言の書式: データ型[,] 配列名;
・要素の生成の書式: 配列名 = new データ型[要素数①, 要素数②];
・宣言と要素の生成は同時に行うことができる
 書式: データ型[,] 配列名 = new データ型[要素数①, 要素数②];
・例: int[,] MonsterMap = new int[100, 100]; //100×100のマップの部屋ごとのモンスター数
・2次元配列なので、要素の利用には2つの添字が必要
・よって、for文による繰り返しを2重化すると良い
・例:

```
 string[,] names = new string[2, 3]; //2種族3匹のモンスター名
```

```
 for (int i = 0; i < 2; i++) { //2種族の分、繰返す
  for (int j = 0; j < 3; j++) { //各3匹の分、繰返す
   name[i,j] = Console.ReadLine();
  }
 }
```

p.138 array01.cs

```
//p.138 array01.cs
using System;
class array01 {
    public static void Main() {
        int[,] MyArray = new int[2,3]; //2×3の2次元配列を生成
        int i, j; //繰返し＆添字用
        MyArray[0, 0] = 1; //要素に値を代入
        MyArray[0, 1] = 2; //   〃
        MyArray[0, 2] = 3; //   〃
        MyArray[1, 0] = 4; //   〃
        MyArray[1, 1] = 5; //   〃
        MyArray[1, 2] = 6; //   〃
        for (i = 0; i < 2; i++) { //添字①の数だけ繰返す
            for (j = 0; j < 3; j++) { //添字②の数だけ繰返す
                Console.WriteLine("MyArray[{0}, {1}] = {2}",i, j, MyArray[i,
j]);
            }
        }
    }
}
```

アレンジ演習:p.138 array01.cs

・コンソールから添字①、添字②、値を入力するようにしよう
・「続ける(y/n)」と表示してyが入力されている間、上記を繰返し、y以外が入力されたら全データを表示しよう
【発展課題】範囲を超える添字①②が入力されたら再入力させる
ヒント:「yが入力されている間」も「再入力させる」もdo-whileでループすると良い

作成例

```
//アレンジ演習:p.138 array01.cs
using System;
class array01 {
    public static void Main() {
        int[,] MyArray = new int[2,3]; //2×3の2次元配列を生成
        int i, j; //繰返し＆添字用
        string ans = "";
        do {
            Console.Write("添字①:"); i = int.Parse(Console.ReadLine());
            Console.Write("添字②:"); j = int.Parse(Console.ReadLine());
            Console.Write("値:"); MyArray[i, j] = int.Parse(Console.ReadLine());
//要素に格納
            Console.Write("続ける(y/n):"); ans = Console.ReadLine();
        } while (ans == "y");
        for (i = 0; i < 2; i++) { //添字①の数だけ繰返す
            for (j = 0; j < 3; j++) { //添字②の数だけ繰返す
```

```
                Console.WriteLine("MyArray[{0}, {1}] = {2}",i, j, MyArray[i,
j]);
            }
        }
    }
}
```

作成例【発展課題】

```
//アレンジ演習:p.138 array01.cs
using System;
class array01 {
    public static void Main() {
        int[,] MyArray = new int[2,3]; //2×3の2次元配列を生成
        int i, j; //繰返し＆添字用
        string ans = "";
        do { //入力の繰返し
            do { //添字①入力の繰返し
                Console.Write("添字①:");
                i = int.Parse(Console.ReadLine());
            } while (i >= 2); //2以上ならやりなおし
            do { //添字②入力の繰返し
                Console.Write("添字②:");
                j = int.Parse(Console.ReadLine());
            } while (j >= 3); //3以上ならやりなおし
            Console.Write("値:"); MyArray[i, j] = int.Parse(Console.ReadLine());
//要素に格納
            Console.Write("続ける(y/n):"); ans = Console.ReadLine();
        } while (ans == "y");
        for (i = 0; i < 2; i++) { //添字①の数だけ繰返す
            for (j = 0; j < 3; j++) { //添字②の数だけ繰返す
                Console.WriteLine("MyArray[{0}, {1}] = {2}",i, j, MyArray[i,
j]);
            }
        }
    }
}
```

p.138（2次元配列の初期化）

・1次元配列と同様に、初期値を列挙することによる初期化が可能
・書式: 型[,] 配列名 = { {値,…}, … }
・例: string[,] names = { {"ヴェルドラ", "ヴェルグリンド"}, {"リムル", "シュナ"} }; //2×2の配列
・要素数は省略できるが「●×■」の形式になる必要がある
・例:
 int[,] MyArray = {{1, 2, 3}, {4, 5}}; //エラーになる
 int[,] MyArray = new int[2, 3]{{1, 2, 3}, {4, 5}}; //エラーになる
 ※テキストp.139の下から4行目の「初期値が設定されていない要素は0で初期化」は誤り

p.139 array02.cs

```
//p.139 array02.cs
using System;
```

```
class array02
{
    public static void Main()
    {
        int[,] MyArray = {{1, 2, 3}, {4, 5, 6}}; //初期化で2×3の配列になる
        for (int i = 0; i < 2; i++) { //添字①の数だけ繰返す
            for (int j = 0; j < 3; j++) { //添字②の数だけ繰返す
                Console.WriteLine("MyArray[{0}, {1}] = {2}", i, j, MyArray[i,
j]);
            }
        }
    }
}
```

p.140 array03.csについて

・このプログラムでは入力チェックを3段階行っている
 ① 文字数が2以上ではないか
 ② 先頭文字が数字以外ではないか
 ③ 範囲は正しいか(クラスは1～2、出席番号は1～5)
・この②で用いているのが、Char.IsNumber(文字列, n番目)で、先頭を0番目として数えたn番目の文字が数字かどうかを返す。
・Charはcharの.NET型で、文字を扱う構造体(第11章にて後述)として提供されている。IsNumberはこれに含まれるメソッド。
・また、③で用いているのが、Int32.Parse(文字列)で、これは、int.Parse(文字列)と同じ意味
・continueについてはp.130参照

p.140 array03.cs

```
//p.140 array03.cs
using System;
class array03 {
    public static void Main() {
        string[,] Name = new string[2, 5] { //2×5要素の文字列の配列
                {"田中六郎", "吉田一郎", "太田太郎", "粂井康孝", "岡田三郎"},
                {"横田芳子", "池田和子", "目黒貴和子", "武田信子", "園田淳子"}
        };
        int MyClass, No; //組と出席番号
        string strClass, strNo; //組と出席番号(入力用)
        while (true) { //無限ループ
            Console.Write("クラスは---");
            strClass = Console.ReadLine();
            if (strClass.Length >= 2) { //2文字以上が入力された？
                Console.WriteLine("入力は1桁のみです");
                continue; //以降をスキップして繰返し続行
            }
            if (Char.IsNumber(strClass, 0) != true) { //1文字目は数字か？
                Console.WriteLine("数字を入力してください");
                continue; //以降をスキップして繰返し続行
            }
            MyClass = Int32.Parse(strClass); //整数に変換(範囲チェックのため)
            if (MyClass <= 0 || MyClass >= 3) {
                Console.WriteLine("クラスは1組か2組です");
                continue; //以降をスキップして繰返し続行
```

```
            }
            break; //チェックを全て済ませたのでチェック完了としてループを抜ける
        }
        while (true) { //無限ループ
            Console.Write("出席番号は---");
            strNo = Console.ReadLine();
            if (strNo.Length >= 2) { //2文字以上が入力された？
                Console.WriteLine("入力は1桁のみです");
                continue; //以降をスキップして繰返し続行
            }
            if (Char.IsNumber(strNo, 0) != true) { //1文字目は数字か？
                Console.WriteLine("数字を入力してください");
                continue; //以降をスキップして繰返し続行
            }
            No = Int32.Parse(strNo); //整数に変換(範囲チェックのため)
            if (No <= 0 || No >= 6) {
                Console.WriteLine("出席番号は1番から5番までです");
                continue; //以降をスキップして繰返し続行
            }
            break; //チェックを全て済ませたのでチェック完了としてループを抜ける
        }
        Console.WriteLine("{0}クラスの出席番号{1}番は{2}さんです", strClass, strNo,
Name[MyClass - 1, No - 1]);
    }
}
```

p.142 3次元以上の配列

・C#では次元数の制限はない
・3次元配列の宣言の書式：データ型[,,] 配列名;
・3次元配列の要素の生成の書式：配列名 = new データ型[要素数①, 要素数②, 要素数③];
・宣言と要素の生成は同時に行うことができる
 書式：データ型[,,] 配列名 = new データ型[要素数①, 要素数②, 要素数③];
・3次元配列では、要素の利用には3つの添字が必要
・よって、for文による繰り返しを3重化すると良い
・例：

```
 string[,,] names = new string[2, 3, 4]; //2種族各3グループ各4匹のモンスター名
 for (int i = 0; i < 2; i++) { //2種族の分、繰返す
  for (int j = 0; j < 3; j++) { //各3グループの分、繰返す
   for (int k = 0; j < 4; j++) { //各4匹の分、繰返す
    name[i, j, k] = Console.ReadLine();
   }
  }
 }
```

・初期値を列挙することによる初期化が可能
・3次元配列初期化の書式：型[,,] 配列名 = { { {値,…}, … }, … }
・例：

```
int[,,] n =
{{{1,2,3,4},{5,6,7,8},{9,10,11,12}},{{13,14,15,16},{17,18,19,20},{21,22,23,24}}}
;
```

・要素数は省略できるが「●×■×▲」の形式になる必要がある

p.142（次元数と要素数の取得）

・配列の次元数は「配列名.Rank」で得られる

・配列の全要素数は多次元配列においても「配列名.Length」で得られる
・C/C++等にように、多次元配列で「配列名[添字].Length」で上位要素数を得ることはできない

p.143 array04.cs

```
//p.143 array04.cs
using System;
class array04 {
    public static void Main() {
        int[,,] ar = new int[2, 2, 3] { //「new int[2, 2, 3]」は省略可
            {
                {0, 1, 2}, //順に[0,0,0][0,0,1][0,0,2]
                {3, 4, 5}  //順に[0,1,0][0,1,1][0,1,2]
            },
            {
                {6, 7, 8}, //順に[1,0,0][1,0,1][1,0,2]
                {9, 10, 11}//順に[1,1,0][1,1,1][1,1,2]
            }
        };
        Console.WriteLine("配列の次元 = {0}", ar.Rank); //次元数を表示
        Console.WriteLine("arの個数 = {0}", ar.Length); //要素数を表示
        for (int i = 0; i < 2; i++) { //1番目の添字
            for (int j = 0; j < 2; j++) { //2番目の添字
                for (int k = 0; k < 3; k++) { //3番目の添字
                    Console.Write("{0}, ", ar[i, j, k]); //順に表示
                }
            }
        }
        Console.WriteLine(); //改行
    }
}
```

アレンジ演習:p.143 array04.cs

・4次元配列[2,2,2,3]にしよう

作成例

```
//アレンジ演習:p.143 array04.cs
using System;
class array04 {
    public static void Main() {
        int[,,,] ar = { //「new int[2, 2, 2, 3]」を省略
            {
                { {0, 1, 2},{3,  4,  5} },
                { {6, 7, 8},{9, 10, 11} },
            },
            {
                { {0, 1, 2},{3,  4,  5} },
                { {6, 7, 8},{9, 10, 11} },
            }
        };
        Console.WriteLine("配列の次元 = {0}", ar.Rank); //次元数を表示
        Console.WriteLine("arの個数 = {0}", ar.Length); //要素数を表示
```

```
        for (int i = 0; i < 2; i++) { //1番目の添字
            for (int j = 0; j < 2; j++) { //2番目の添字
                for (int k = 0; k < 2; k++) { //3番目の添字
                    for (int l = 0; l < 3; l++) { //4番目の添字
                        Console.Write("{0}, ", ar[i, j, k, l]); //順に表示
                    }
                }
            }
        }
        Console.WriteLine(); //改行
    }
}
```

p.144 ジャグ配列

・C#における多次元配列のもう一つの様式で、配列の配列によって多次元配列を表す
・通常の多次元配列より記述が煩雑になるが、内側の要素数の異なる多次元配列を構築できるので、メモリの無駄を省くことができる
・以下は2次元のジャグ配列の場合:
・「[,]」ではなく、C/C++/Java等と同様に「[][]」で表す
・ジャグ配列の例:a[0][0] a[0][1] a[1][0] a[2][0] a[2][1] a[2][2]
・ジャグ配列の宣言の書式: データ型[][] 配列名;
・ジャグ配列の要素の生成の書式:
 配列名 = new データ型[要素数①][];
 配列名[0] = new データ型[要素数②]; 配列名[1] = new データ型[要素数③]; …
・初期値を列挙することによる初期化が可能
・ジャグ配列初期化の書式:
 配列名 = new データ型[要素数①][];
 配列名[0] = new データ型[要素数②]{値,…}; 配列名[1] = new データ型[要素数③]{値,…}; …
・「●×■」の形式になる必要はない
・配列の全要素数はジャグ配列においては「配列名.Length」では得られず、構成する配列数になる
・「配列名[添字].Length」で各配列の要素数が得られる

p.144 jagged01.cs

```
//p.144 jagged01.cs
using System;
class jagged01 {
    public static void Main() {
        int[][] ar; //(2次元)ジャグ配列の宣言
        ar = new int[2][]; //上位次元の要素数＝含む配列の数は2
        ar[0] = new int[3]; //下位次元の要素数＝含まれる配列①の要素数は3
        ar[1] = new int[3]; //下位次元の要素数＝含まれる配列②の要素数は3
        //下記の※は含まれる2配列の要素数がどちらも3だから可能だが、そうとは限らない
        for (int i = 0; i < 2; i++) { //上位次元の要素数＝含む配列の数だけ繰返す
            for (int j = 0; j < 3; j++) { //下位次元の要素数＝含まれる配列の要素数だけ繰返す(※)
                ar[i][j] = (i + 1) * (j + 1); //要素に値を代入
            }
        }
        for (int i = 0; i < 2; i++) { //上位次元の要素数＝含む配列の数だけ繰返す
            for (int j = 0; j < 3; j++) { //下位次元の要素数＝含まれる配列の要素数だけ繰返す(※)
                Console.WriteLine("ar[{0}][{1}] = {2}", i, j, ar[i][j]);
```

```
            }
        }
    }
}
```

p.146 jagged02.cs

```
//p.146 jagged02.cs
using System;
class jagged02 {
    public static void Main() {
        string[][] name = new string[2][]; //(2次元)ジャグ配列の宣言と上位次元の要素数＝含む配列の数は2の指定
        name[0] = new string[2]{"田中", "工藤"}; //含まれる配列①の初期化(2要素)
        name[1] = new string[3]{"吉田", "佐藤", "池田"}; //含まれる配列②の初期化(3要素)
        for (int i = 0; i < name[0].Length; i++) { //含まれる配列①の要素数について繰返す
            Console.WriteLine(name[0][i]); //含まれる配列①の要素を表示
        }
        for (int i = 0; i < name[1].Length; i++) { //含まれる配列②の要素数について繰返す
            Console.WriteLine(name[1][i]); //含まれる配列①の要素を表示
        }
    }
}
```

提出：アレンジ演習：p.146 jagged02.cs

・2つあるforを2重ループにして見やすくしよう
・ヒント：下記の●を0から1まで繰返せばよい

```
        for (int j = 0; j < name[●].Length; j++) { //含まれる配列の要素数について繰返す
            Console.WriteLine(name[●][j]); //含まれる配列の要素を表示
        }
```

次回予告：p.147「暗黙の型指定がなされた配列」から