配列

C#で言う「配列」

配列：同じ型を同時にたくさん使う方法の一つ

「配列」の書き方

型 [] 配列名 ＝ new 型 [要素数];

C#で言う「配列」

例 :

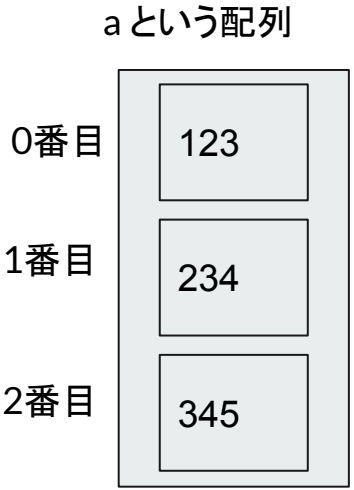
int [] a = new int [3];

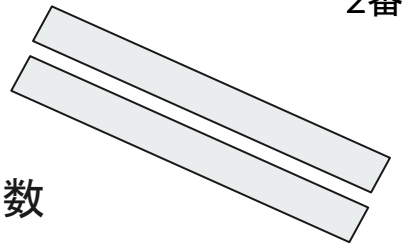
a[0] = 123;

a[1] = 234;

a[2] = 345;

とした時





a.Length ← 配列の要素数

int [] a = { 123 , 234 , 345 }

C#で言う「配列」

例 :

int [] a = new int [3];

とした時

a[0] = 123;

a[1] = 234;

a[2] = 345;

a[3] = 456;　//エラー

a = 567;　//エラー

配列型

配列型は、同じ型の複数の値を1つの固まりとして扱うための型です。配列に入れたデータ（要素）には「0」から順番に番号（インデックスと呼びます）が振られ、インデックスを使って値にアクセスし、データを読み取ったり変更したりできます。

要素の数を増やしたい場合は、新しい配列を丸ごと作り直して上書きする必要があります。

```
int[] values = new int[] {1, 2, 3};
Debug.Log(values.Length);
Debug.Log(values[0]);
values[0] = 100;
Debug.Log(values[0]);
```

「1, 2, 3」という3つのint型データが入った配列を作成する

配列の要素が何個あるかをカウントする。3個入っているので、「3」が出力される

0番目の要素を取得する。「1」が出力される

0番目の要素を「100」に書き換える

「100」が出力される

List型

List型は、配列型と似ていますが、要素の数を後から増やすことができます。

```
List<int> values = new List<int> {1, 2, 3};
1, 2, 3という3つのint型データが入ったListを作成する
Debug.Log(values.Count);  Listの要素が何個あるかをカウントする。3個入っているので、「3」が出力される
Debug.Log(values[0]);  0番目の要素を取得する。「1」が出力される
values[0] = 100;  0番目の要素を「100」に書き換える
Debug.Log(values[0]);  「100」が出力される
values.Add(999);  要素「999」を末尾に追加する
Debug.Log(values[values.Count - 1]);  「999」が出力される
```

Dictionary型

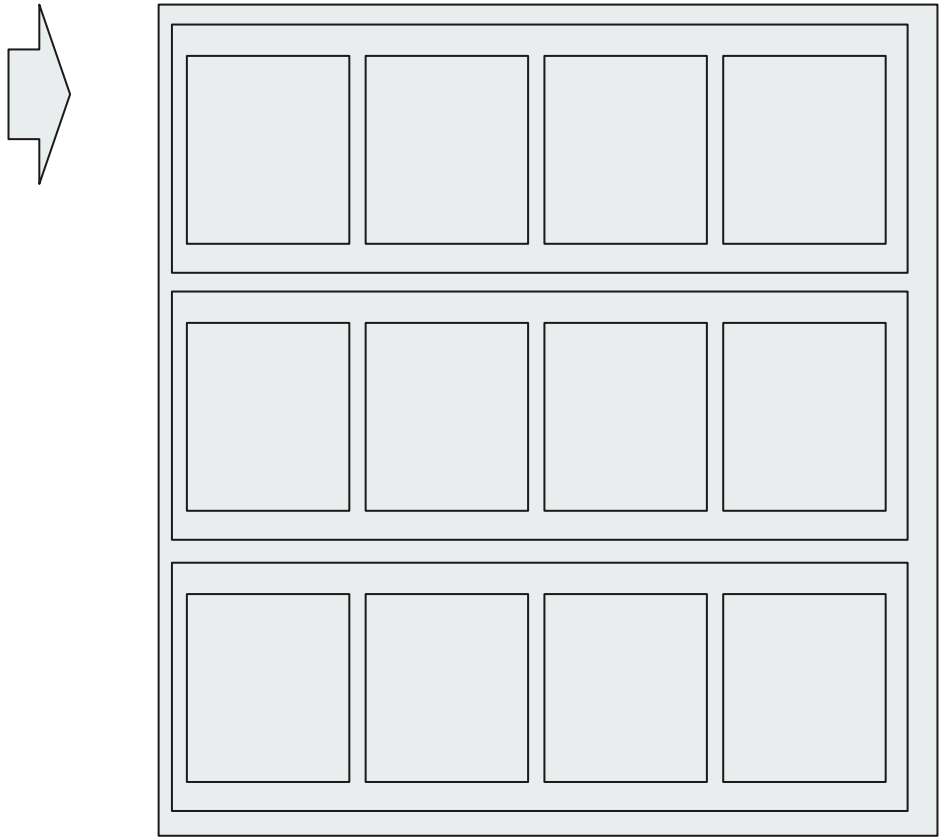
Dictionary型は辞書型とも呼ばれます。List型と似ていますが、自動で振られるインデックスの代わりに「キー」と「値」をセットにしてデータを保持します。

```
Dictionary<string, int> values = new Dictionary<string, int> {
    {"いち", 1},
    {"に", 2},
    {"さん", 3},
};  いち = 1, に = 2, さん = 3という3つのデータが入ったDictionaryを作成する
Debug.Log(values.Count);  Dictionaryの要素が何個あるかをカウントする。「3」が出力される
values["いち"] = 100;  "いち"の要素を「100」に書き換える
Debug.Log(values["いち"]);  「100」が出力される
values.Add("よん", 999);  "よん" = 999 の要素を追加する
Debug.Log(values["よん"]);  「999」が出力される
```

多次元配列
と
配列の配列

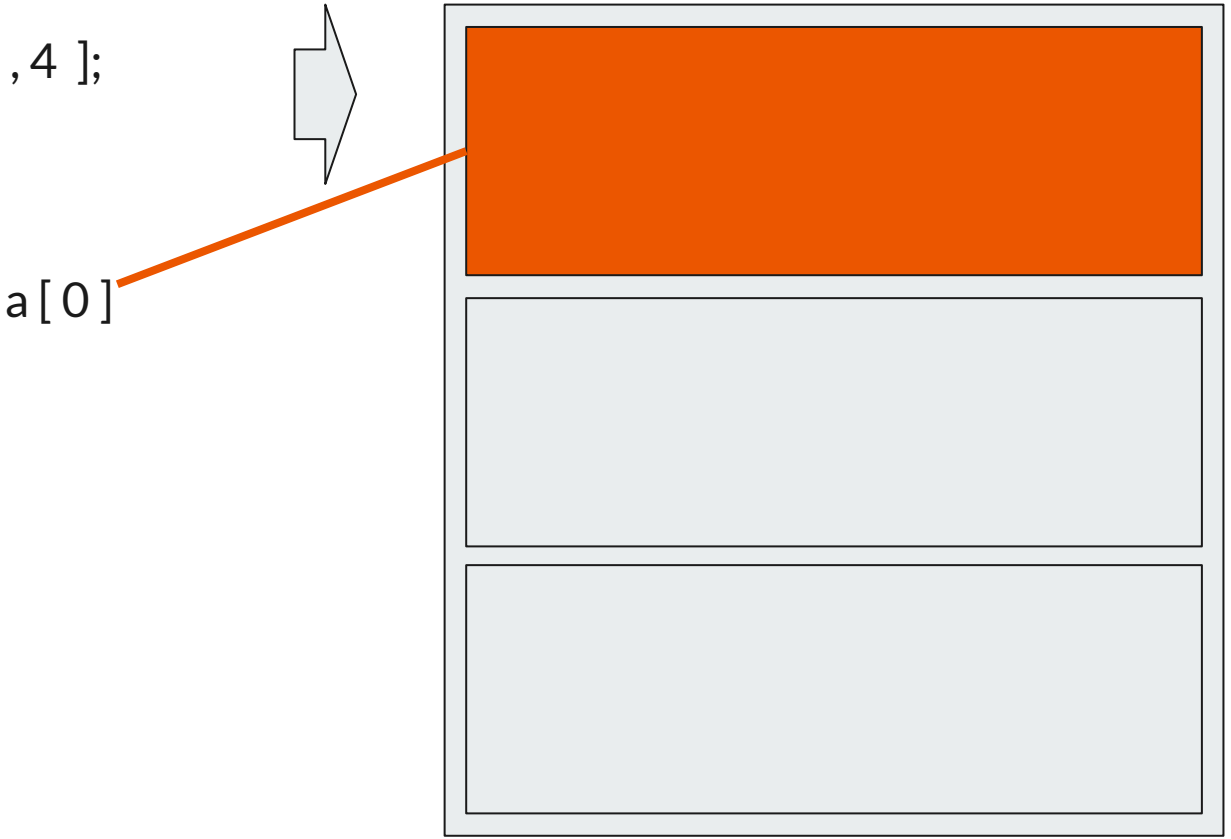
多次元配列

```
int [ , ] a = new int [ 3 , 4 ];
```



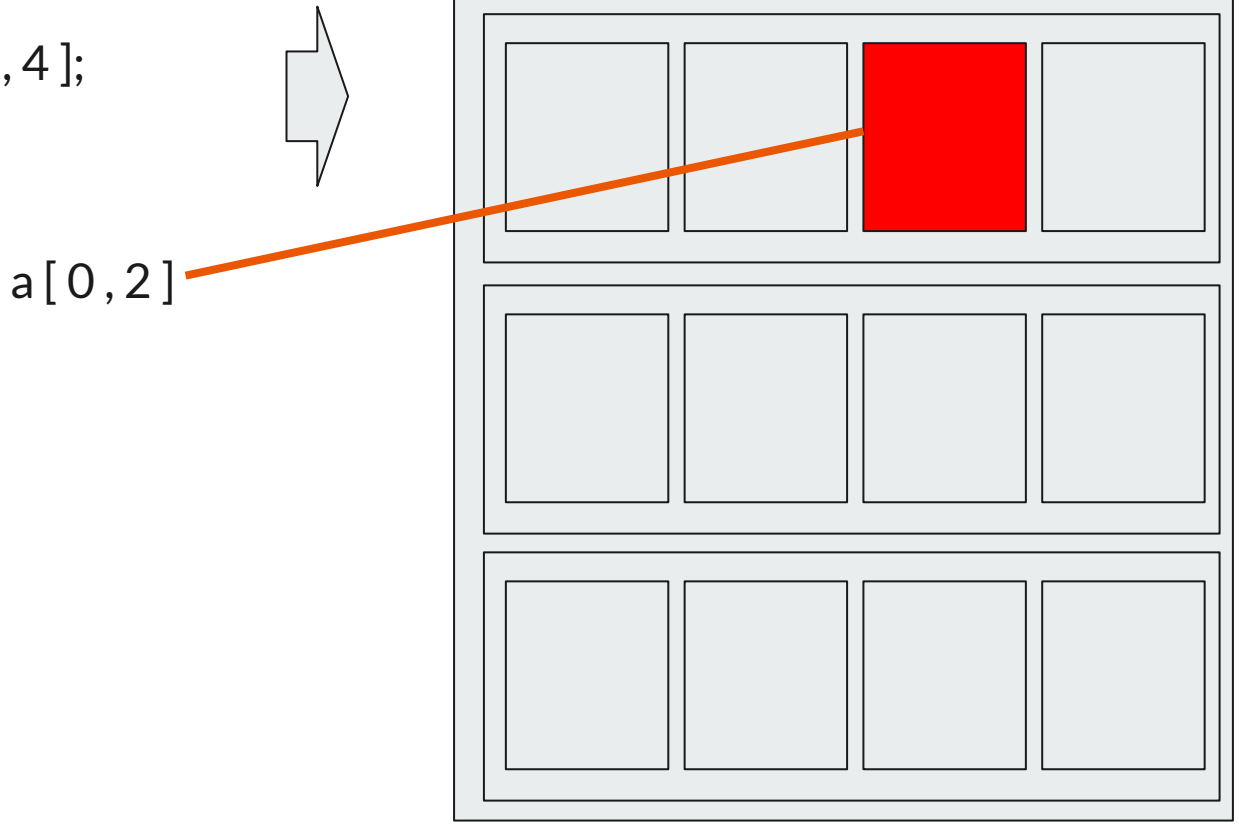
多次元配列

```
int [ , ] a = new int [ 3 , 4  ];
```



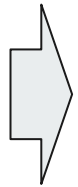
多次元配列

```
int [ , ] a = new int [ 3 , 4 ];
```



配列の配列

```
int [][] a = new int [3][];
```



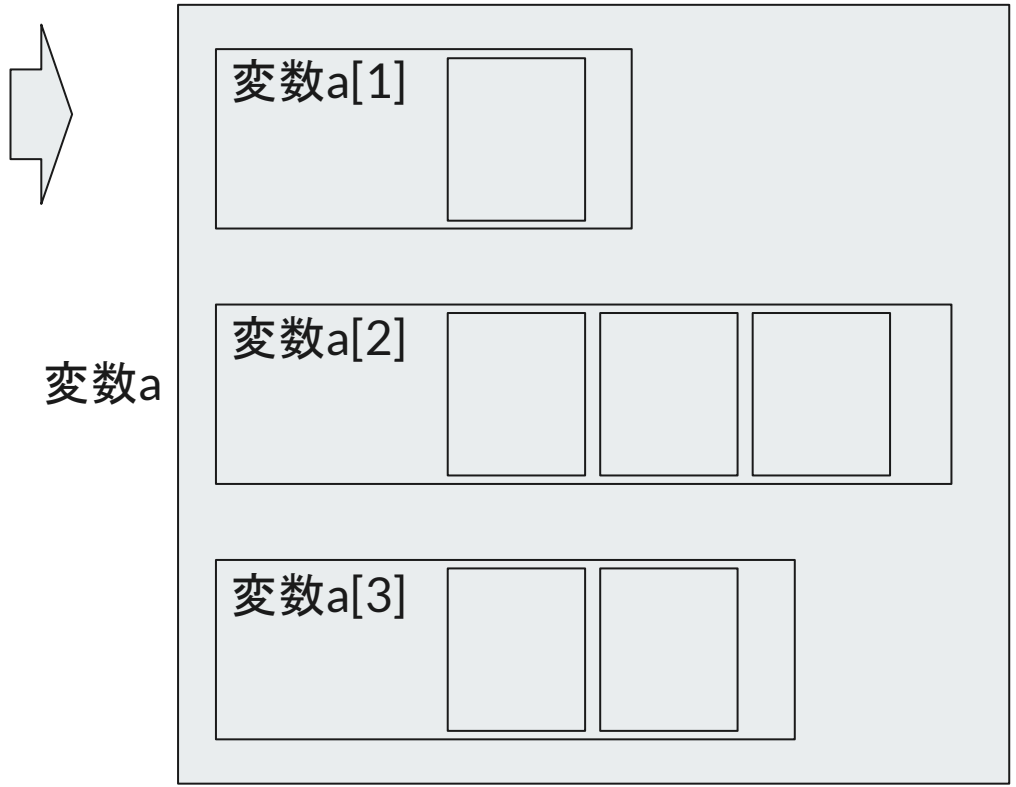
配列の配列

int [][] a = new int [3][];

a[1] = new int[1];

a[2] = new int[3];

a[3] = new int[2];



リバーシなどの例

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

public class GameManager : SingletonMonoBehaviour<GameManager>
{
    public static readonly int x_Row = 8, z_Col = 8;
    //x座標  列数  縦
    //z座標  行数     日本語だと行の数え方が縦も横もあるのでややこしい

    float disk_Interval = 10f;
    public Disk[,] Disks { get; private set; }

    [SerializeField] Disk _diskPrefab;
    [SerializeField] Transform _diskBase;

    void Start(){
        Disks = new Disk[z_Col, x_Row];//多次元配列
        for (var z = 0; z < z_Col; z++){
            for (var x = 0; x < x_Row; x++){
                var disk = Instantiate(_diskPrefab, _diskBase);//diskBase基準に生成 //初めに生成しておく
                var temp = disk.transform;//Transformのアドレス情報をtempに渡す
                temp.localPosition = new Vector3(disk_Interval * x, 0, disk_Interval * z);
                Disks[z, x] = disk;//disk情報を配列に入れる
            }
        }
    }
```