??

????

記入できるファイルをダウンロード

C#???????

以下の表は、　　C#で書いた命令とその日本語の意味が書いてある。空枠を完成してください。

C#	日本語
`int [] numbers = new int[10]`	10個の整数が入る「　　numbers」という配列を作成
`int [] numbers;`	答え 整数型の配列を宣言のみ（初期化なし）
`string [] names = new string[5];`	答え 5個の文字列が入る「　　names」という配列を作成
答え `names[0] = "たろう";`	「names」配列の一番最初（先頭）は「たろう」にする
答え `names[4] = "おおはら";`	「names」配列の一番最後（後尾）は「おおはら」にする

C#	日本語
`int a = numbers[3];`	答え 配列「numbers」の先頭から4つ目の要素を変数「a」の中に代入

答え
`float [] distances = new float[3];`
`distances[0] = 1.3f;`
`distances[1] = 4.5f;`
`distances[2] = 7.9f;`
 	
`// または`
`float [] distances =`
`new float[3] { 1.3f, 4.5f, 7.9f };`
	少数型の配列「distances」を宣言し、1.3f, 4.5f, 7.9fの3つの数値で初期化する

C#????????????

1. 配列とfor文を使い、以下のスクリプトを完成してください（Unityでも OK）
※: 文字列の長さは「変数名.Length」で求めることができる
例:

```
string s = "Hello!";
int size = s.Length; // 文字列「s」の長さを求める
Debug.Log(size); // 6 を表示する
```

※: 期待の結果は「最も長いフルーツ名は: grapefruit」

```
// フルーツの名前の配列
string [] fruits = {"lemon", "mikan", "orange", "grapefruit", "kiwi"};

// 最も長い文字列（結果）
string maxFruit = "";

// フルーツ1個ずつ確認
```

```
for (________; _________; ________)
{
    // i番目のフルーツを求める
    string f = _________________;

    // もし、今回のフルーツ名の方が長ければ
    if (f.Length > maxFruit.Length)
    {
        // 最も長いフルーツを更新
        ____________________;
    }
}

// 最大の長さのフルーツ名を表示
Debug.Log($"最も長いフルーツ名は: {_______________}");
```

答え

```
// フルーツの名前の配列
string [] fruits = {"lemon", "mikan", "orange", "grapefruit", "kiwi"};

// 最も長い文字列（結果）
string maxFruit = "";

// フルーツ1個ずつ確認
for (int i = 0; i < 5; i++)
{
    // i番目のフルーツを求める
    string f = fruits[i];

    // もし、今回のフルーツ名の方が長ければ
    if (f.Length > maxFruit.Length)
    {
        // 最も長いフルーツを更新
        maxFruit = f;
    }
}

// 最大の長さのフルーツ名を表示
```

```
Debug.Log($"最も長いフルーツ名は: {maxFruit}");
```

2. 配列と　　for文を使い、以下のスクリプトを完成してください（　　Unityでも　OK）

```
// 4人分のスコア（整数型）の配列を準備してください
_____ scores = ___________;

//1人目のスコアは100点
__________= 100;

//2人目のスコアは80点
__________= 80;

//3人目のスコアは180点
__________= 180;

//4人目のスコアは20点
__________= 20;

// 同様に、4人分の名前（文字列型）の配列を準備してください
// それぞれの名前は「太郎」「さくら」「オオハラ」「Unity」

______ names = __________;
_________________;
_________________;
_________________;
_________________;

// for文を使って一人ずつのプレーヤーとスコアを表示する
for (________; __________; _________)
{
    Debug.Log($"名前: {_________} → {__________}");
}
```

答え

```
// 4人分のスコア（整数型）の配列を準備してください
int[] scores = int[4];

//1人目のスコアは100点
scores[0] = 100;

//2人目のスコアは80点
scores[1] = 80;

//3人目のスコアは180点
scores[2] = 180;

//4人目のスコアは20点
scores[3] = 20;

// 同様に、4人分の名前（文字列型）の配列を準備してください
// それぞれの名前は「太郎」「さくら」「オオハラ」「Unity」

string[] names = string[4];
names[0] = "太郎";
names[1] = "さくら";
names[2] = "オオハラ";
names[3] = "Unity";

// for文を使って一人ずつのプレーヤーとスコアを表示する
for (int i = 0; i < 4; i++)
{
    Debug.Log($"名前: {names[i]} → {score[i]}");
}
```

???????? : List

配列を作るときに、事前にサイズを決めないといけないが、場合により、サイズが分からない可能性がある。例えば、同じゲームでも、プレーヤーによって拾うアイテムの数が変わるので、事前にアイテムの配列のサイズが分からない。

その場合は、サイズが自動的に変わる「リスト」を使うと便利。「List」は、要素を増やすほど、段々サイズが伸びる"配列"（正しい名称は「コレクション」）である。

便利であるが、弱点もある：リストは配列と比べて処理が遅い。数十〜数百の要素以内だったら、あまり違いがないが、要素を増やすほど（数万程度）、Listの方が明らかに遅い。

List???????

宣言	`List<型> 変数名;`
代入（作成）	`変数名 = new List<型>();`

例:

```
// 名前のリストを作成
List<string> names = new List<string>();

// リストに名前を追加する
names.Add("たろう");
names.Add("さくら");
names.Add("おおはら");

// 0番目の名前を表示（配列と同じ）
Debug.Log(names[0]); // 「たろう」を表示

// 要らなくなった場合、リストから削除できる
names.Remove("たろう");

// 0番目の名前を表示（配列と同じ）
Debug.Log(names[0]); // 「さくら」を表示
```

??

プレーヤーのスコアを表示するスクリプトを配列からリストに書き換える

答え

```
// スコア（整数型）のリストを準備してください
List<int> scores = new List<int>();
```

```
//1人目のスコアは100点
scores.Add(100);

//2人目のスコアは80点
scores.Add(80);

//3人目のスコアは180点
scores.Add(180);

//4人目のスコアは20点
scores.Add(20);

// 同様に、4人分の名前（文字列型）の配列を準備してください
// それぞれの名前は「太郎」「さくら」「オオハラ」「Unity」
List<string> names = new List<string>();
names.Add("太郎");
names.Add("さくら");
names.Add("オオハラ");
names.Add("Unity");

// for文を使って一人ずつのプレーヤーとスコアを表示する
for (int i = 0; i < 4; i++)
{
    Debug.Log($"名前: {names[i]} → {score[i]}");
}
```

?????????????? : foreach

「 for」を使って配列をループする方法を見てきた。変数を使って、インデックスを 0 から配列の最後まで一つずつ繰り返す。例えば:

```
string [] names = string[4] { "アリス", "ボブ", "チャーリー", "ジョン" };
for (int i= 0; i < 4; i++)
{
  string n = names[i]
  Debug.Log(n);
}
```

C#では、配列やリストをもっと簡単にループする方法がある。「`foreach`」（フォー・イーチ）を使うと、インデックス用の変数を用意しなくても、リストの各要素を順番に取り出すことができる。先ほどの例はこうなる:

```
string [] names = string[4] { "アリス", "ボブ", "チャーリー", "ジョン" };
foreach(string n in names)
{
  Debug.Log(n);
}
```

ただし、注意点が二つある。

1. `foreach`は常に要素を一つずつ順番に処理する。一部の要素をスキップしたい場合は、通常の`for`ループを使う必要がある。
2. `foreach`は読み取り専用である。リストや配列の要素を変更したい場合も、通常の`for`ループを使う必要がある。

??

最も長いフルーツの名前を求めるスクリプトを更新し、`for`から`foreach`に書き換える

答え

```
// フルーツの名前の配列
string [] fruits = ["lemon", "mikan", "orange", "grapefruit", "kiwi"];

// 最も長い文字列（結果）
string maxFruit = "";

// フルーツ1個ずつ確認
foreach(string f in fruits)
{
    // もし、今回のフルーツ名の方が長ければ
    if (f.Length > maxFruit.Length)
    {
        // 最も長いフルーツを更新
        maxFruit = f;
    }
}
```

```
// □□□□□□□□□□□□□□□□
Debug.Log($"□□□□□□□□□□□: {maxFruit}");
```

Revision #6
Created 2026-04-07 00:40:23 UTC by Menendez Francisco
Updated 2026-04-28 01:42:44 UTC by Menendez Francisco