??

????

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
`a > b`	変数「　a」は変数「　　b」よりも大きい?
`i <= 10`	答え 変数「　i」は「　10」以下?
答え `playerHP > 0`	プレーヤーの体力は残っている?
`stageSelect == "Tutorial"`	答え 選択されたステージはチュートリアルであるか?
`coinCount >= itemCost`	答え 持っているコイン数はアイテム価格を超える?

C#	日本語
答え `keyCount == 3`	鍵がちょうど 3個ある?
答え `time < 60`	残りの時間は 60秒未満?
`player != "CPU Player"`	答え プレーヤーは CPUプレーヤーではないね?

?????

以下の比較を確認し、「 true」（真・はい）または「 false」（偽・いいえ）をチェックしてください。ただし、事前に以下の変数をすでに作成されている:

```
int a = 100;
int b = 20;
float c = 5.0f;
```

比較	true	false
`a == 100;`	✔	
`b < c`	答え falseでした! >> ・ b は 20 ・ c は 5 ・ b は c 未満 (<) ではない	答え ✔

比較	true	false
`c >= a / b;`	答え ✔	答え << trueでした! ・ c は 5 ・ a / b ⇒ 100 /20 ⇒ 5 ・ 5は 5以上 (>=)である
`c * 6 < b * 2;`	答え ✔	答え << trueでした! ・ c×6 ⇒ 5×6 ⇒ 30 ・ b×2 ⇒ 20×2 ⇒ 40 ・ 30は 40未満 (<) である
`a == b * c;`	答え ✔	答え << trueでした! ・ aは 100 ・ b×c⇒ 20×5 ⇒ 100 ・同じ (==) である
`b != a - 80;`	答え falseでした! >> ・ b は 20 ・ a - 80 ⇒ 100 - 80 ⇒ 20 ・異なる (!=) のではない	答え ✔

??????????????????????

ゲームを開発しているときに、 if文で同時に2つのことを確認したい場合もある。

例えば、レースゲームの勝利条件は:

- ゴールに着いた
- 時間がまだ残っている

両方とも同時に「 true」だったら、ゲームの勝ちになる。2つの条件同時に満たさないといけない場合は「AND演算子」を使いましょう

????????AND??????&&

この演算子で両方の条件を満たさないと「 true」にならない。2つの条件「A」と「B」があるとしたら:

A	B	A && B
false	false	false
true	false	false
false	true	false
true	true	true

つまり、いずれかの条件が「 false」でしたら、すべて「 false」になる。上記の例で、 C#にすると:

```
// 走行距離
float runDistance = 1235.0f;

// ゴールまでの距離
float goalDistance = 1200.0f

// 残りの時間
float timeLeft = 35.7f;

// 走行距離はゴールよりも大きく「かつ」残りの時間が0以上だったら
if (runDistance >= goalDistance && timeLeft > 0)
{
  Debug.Log("勝ち！");
}
```

これで、必ず2つの条件を満たしたら、ゲームクリアになることを保証できる。

?????????OR?????||

もうーつのパターンは「いずれかの条件が true だったら、すべて true になる」かある。例えば、あるゲームの中で2つのドアの開き方がある:

- ドアの鍵を持っている
- プレーヤーの力が十分だったら、自力で壊す

この場合は、1つだけの条件を満たしたら、ドアを開くことができるので、「または」の演算子を使えば良いでしょう。

A	B	A || B
false	false	false
true	false	true
false	true	true
true	true	true

上記の例で、 C#にすると:

```
// プレーヤーの力
float playerPower = 30;

// 今持っている鍵の数
int keyCount = 0;

// ドアの固さ
float doorHardness = 25;

// 鍵があれば「または」力は固さよりも大きければ…
if (keyCount > 0 || playerPower >= doorHardness)
{
  Debug.Log("ドアを開ける！");
}
```

この例だと、鍵がなくても力で壊せるので、ドアを開くことができる!

??

以下、使った「かつ」と「または」の比較を確認し、「 true」（真・はい）または「 false」（偽・いいえ）を書いてください。ただし、事前に以下の変数をすでに作成されている:

```
int coin = 230;        // コインの数
int maxItem = 10;      // 最大のアイテムの数
int nowItem = 9;       // 現在持っているアイテムの数
int potionCost = 100; // ポーションのコスト
```

1. 比較:

```
coin > potionCost && nowItem + 1 <= maxItem
```

true	false

答え
✔
	答え
	<< trueでした!
	コインは 230でポーションは 100なので
	`coin > potionCost`
	は `true` (真) である
	で、持っているアイテムは 9個で、最大 10個なので
	`nowItem + 1 <= maxItem`
	も `true` (真) である
	両方とも「 true」であり、「 true かつ true」は「 true」

2. 比較:

```
potionCost > coin || nowItem >= maxItem
```

true	false

答え

falseでした! >>

ポーション（ 100）はコイン（ 230）を超えないので

```
potionCost > coin
```

は「 false」である。

で、持っているアイテムの数（ 9
）最大アイテム以上ではないため

```
nowItem >= maxItem
```

も「 false」である

「 false」または「 false
」（両方も「偽」）であるため、結果は false である

答え

✔

3. 比較:

```
potionCost * 2 <= coin && maxItem >= nowItem + 2
```

true	false

答え

false でした! >>

ポーション2個は 200で、もっといるコインは 230なので

```
potionCost * 2 <= coin
```

は `true` (真) である

だが、持っているアイテム (9 個) にさらに2つを増やすと、最大数 (10 個) を超えてしまうので

```
maxItem >= nowItem + 2
```

も `false` (偽) である

「かつ」 (&&) は両方とも「 true 」じゃないとだめなので、最終結果は「 false 」になってしまう。

答え

✔

4. 比較:

```
nowItem == 0 && coin != 0
```

true	false

答え

false でした! >>

持っているアイテムは0ではなく、9個なので

```
nowItem == 0
```

は `false` (偽) である

「かつ」 (&&) は両方とも「 true 」じゃないとだめなので、もう一つの条件は関係なく、最終結果は「 false」になってしまう。

答え

✔

5. 比較:

```
maxItem - nowItem > 9 || coin > 0
```

true	false

答え
✔
	答え
	<< trueでした!
	maxItemは 10で、 nowItemは 9なので
	maxItem - nowItem ⇒ 10 - 9 ⇒ 1
	で、 1 は 9 以上ではないので、
	`maxItem - nowItem > 9`
	は `false` (偽) である
	だが、コインの数 (230) は0以上なので
	`coin > 0`
	も `true` (真) である
	「または」 (||) の場合は、 いずれか____ 真だったら、すべて真になるため、この比較は「 true 」である。

Revision #10
Created 2026-03-23 07:33:18 UTC by Menendez Francisco
Updated 2026-04-21 05:09:57 UTC by Menendez Francisco