????

????

記入できるファイルをダウンロード_____

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

1. 以下の　C#のプログラムを完成してください

```
// ゲームでダメージを受け、残機を確認し、ゲームオーバーかどかう判定する

int hp = 50; // 体力
int lives = 1; // 残機
int damage = 100; // ダメージ

// ダメージを受け、体力を減らす

_____

// もし、体力が0以下だったら...

_____
{
    // 残機を一つ減らす

    _____
}

// もし、残機がゼロだったら

_____
{
    Debug.Log("ゲームオーバー");
}
```

```
// そうじゃない場合は、残機を表示
else
{
    Debug.Log($"残機: {lives}");
}
```

答え

```
// ゲームでダメージを受け、残機を確認し、ゲームオーバーかどかう判定する

int hp = 50; // 体力
int lives = 1; // 残機
int damage = 100; // ダメージ

// ダメージを受け、体力を減らす
hp -= damage; // または hp = hp - damage;

// もし、体力が0以下だったら...
if (hp <= 0)
{
    // 残機を一つ減らす
    lives--;
}

// もし、残機がゼロだったら
if (lives == 0)
{
    Debug.Log("ゲームオーバー");
}

// そうじゃない場合は、残機を表示
else
{
    Debug.Log($"残機: {lives}");
}
```

このプログラムはコンソールで何を表示するか? その出力を書いてください:

答え

ゲームオーバー

---

2. 以下の　C#のプログラムを完成してください

```
// レースゲームで、チェックポイントを通過することにより、時間切れを延期する

float timeLeft = 8.51; // 残りの時間（秒数）
float raceDistance = 3643.43; //走行距離

float checkPoint01 = 2300; // スタート → チェックポイント1の距離
float checkPoint02 = 1500; //  チェックポイント1 → チェックポイント2距離
float goalPoint = 2500; // チェックポイント2 → ゴールまでの距離

// もし、ゴールを超えていたら...

_____________________________________________________
{
    Debug.Log("レース終了！");
}

// 一方、もし走行距離がチェックポイント２を超えたら...

_____________________________________________________
{
    Debug.Log("チェックポイント２通過！");
    timeLeft += 80;
}

// 一方、もし走行距離がチェックポイント１を超えたら...

_____________________________________________________
{
    Debug.Log("チェックポイント1通過！");
    timeLeft += 60;
}

// 残りの時間を表示
```

```
Debug.Log($"残りの時間: {timeLeft}");
```

答え

```
// レースゲームで、チェックポイントを通過することにより、時間切れを延期する

float timeLeft = 8.51; // 残りの時間（秒数）
float raceDistance = 3643.43; //走行距離

float checkPoint01 = 2300; // スタート → チェックポイント1の距離
float checkPoint02 = 1500; //  チェックポイント1 → チェックポイント2距離
float goalPoint = 2500; // チェックポイント2 → ゴールまでの距離

// もし、ゴールを超えていたら...
if (raceDistance >= checkPoint01 + checkPoint02 + goalPoint)
{
    Debug.Log("レース終了！");
}

// 一方、もし走行距離がチェックポイント2を超えたら...
else if (raceDistance >= checkPoint01 + checkPoint02)
{
    Debug.Log("チェックポイント2通過！");
    timeLeft += 80;
}

// 一方、もし走行距離がチェックポイント1を超えたら...
else if (raceDistance >= checkPoint01)
{
    Debug.Log("チェックポイント1通過！");
    timeLeft += 60;
}

// 残りの時間を表示
Debug.Log($"残りの時間: {timeLeft}");
```

このプログラムはコンソールで何を表示するか? その出力を書いてください:

答え

チェックポイント1通過!
残りの時間: 68.51

Unity???

体重と身長で BMIを計算し、画面で適切な出力を表示しよう（参考: BMIと適正体重_____）

式: BMI = 体重 kg ÷ (身長 m)^2

日本肥満学会の判定基準

BMI値	判定
18.5未満	低体重(痩せ型)
18.5～25未満	普通体重
25～30未満	肥満(1度)
30～35未満	肥満(2度)
35～40未満	肥満(3度)
40以上	肥満(4度)

ヒント:

- 体重と慎重の変数を作成し、適当な数値を代入
- BMIを計算
- if分で BMIの判定し、コンソールで表示

答え

```
// 体重
float weight = 70;

// 身長
float height = 1.71;

// BMIを計算
float bmi = weight / (height * height);
```

```
if (bmi < 18.5)
{
    Debug.Log("低体重");
}
else if (bmi < 25)
{
    Debug.Log("普通体重");
}
else if (bmi < 30)
{
    Debug.Log("肥満(1度)");
}
else if (bmi < 35)
{
    Debug.Log("肥満(2度)");
}
else if (bmi < 40)
{
    Debug.Log("肥満(3度)");
}
else
{
    Debug.Log("肥満(4度)");
}
```

?????

??????????==???????switch-case?

次の問題について考えてみてください

サイコロの数値は　1から　6までの間にして、その値を漢字で表示したい。つまり「　1」の場合は「一」、「　2」の場合は「二」など。　if-elseで作成するとしたら、以下のようなプログラムになる:

```
// サイコロの値
int dice = 5;
```

```
if (dice == 1)
{
    Debug.Log("一");
}
else if (dice == 2)
{
    Debug.Log("二");
}
else if (dice == 3)
{
    Debug.Log("三");
}
else if (dice == 4)
{
    Debug.Log("四");
}
else if (dice == 5)
{
    Debug.Log("五");
}
else if (dice == 6)
{
    Debug.Log("六");
}
else
{
    Debug.Log("無効な数値");
}
```

このプログラムは、カスケードのように「1じゃなければ、2を確認する、2じゃなければ、3を確認する」、次々に「==」で比較するパターンは時々に現れる。この場合は「switch-case」が便利。上のプログラムは以下のようになる:

```
// サイコロの値
int dice = 5;

switch(dice)
{
case 1: // dice == 1 と同じ
    Debug.Log("一");
```

```
    break;
case 2: // dice == 2 と同じ
    Debug.Log("二");
    break;
case 3: // dice == 3 と同じ
    Debug.Log("三");
    break;
case 4: // dice == 4 と同じ
    Debug.Log("四");
    break;
case 5: // dice == 5 と同じ
    Debug.Log("五");
    break;
case 6: // dice == 6 と同じ
    Debug.Log("六");
    break;
default: // elseと同じ
    Debug.Log("無効な数値");
    break;
}
```

??????????????enum?

次の問題について考えてみてください。武器の種類により、攻撃力を変える処理を作りたい。処理しやすくするため、それぞれの武器に整数を与えた:

0: 非武装（攻撃力　　: 1）
1: 剣（攻撃力　　: 5）
2: ナイフ（攻撃力　　: 2）
3: 鉄砲（攻撃力　　: 10）

「 switch-case」を使えば、以下のような処理になる:

```
int damage = 0;
int weapon = 2;
switch(weapon)
{
case 0:
  damage = 1;
  break;
case 1:
```

```
    damage = 5;
    break;
case 2:
    damage = 2;
    break;
case 3:
    damage = 10;
    break;
}
Debug.Log($"攻撃力: {damage}");
```

これでも実行ができるが、時間がたてば「0はなんだったっけ？」「1は？」のように、それぞれの整数の意味を忘れてしまう可能性がある。各数値に忘れないように、名前を付ける方法がある。「列挙」を作成することにより、各整数に読みやすい名前を付けることができる。

まず、列挙を定義しなければならない:

書式___

```
enum 列挙名
{
    整数「0」の名前,
    整数「1」の名前,
    整数「2」の名前,
    ...
    整数「n」の名前,
}
```

武器の例で、以下の列挙は適切でしょう:

```
enum Weapon
{
    Unarmed, // 0:非武装
    Sword,   // 1:剣
    Knife,   // 2:ナイフ
    Pistol,  // 3:鉄砲
}
```

そして、「　　switch-case」はとても分かりやすくなる!

```
int damage;
Weapon wpn = Weapon.Knife;  // 新し「Weapon」というデータ型を使える!
switch(wpn)
{
case Weapon.Unarmed: //「0」と同じ
  damage = 1;
  break;
case Weapon.Sword:   //「1」と同じ
  damage = 5;
  break;
case Weapon.Knife:   //「2」と同じ
  damage = 2;
  break;
case Weapon.Pistol:  //「3」と同じ
  damage = 10;
  break;
}
Debug.Log($"攻撃力: {damage}");
```

????

ゲームでのアイテムショップを作ろうとして、以下のアイテムとその価格が決まった:

ポーション（小）	10ゼニ
ポーション（大）	25ゼニ
木の盾	50ゼニ
鉄の盾	100ゼニ
皮のブーツ	40ゼニ
魔法のマント	250ゼニ

ここで、次のプログラムを C#で作成し、 Unityで確認してください

1. 各種のアイテムを表すことができる「 `ItemType` 」の列挙を作成
2. プレーヤーを持っているお金の変数を準備（ `playerMoney` - 整数型）
3. プレーヤーが買いたいアイテムの変数を準備（ `buyItem` - `ItemType` の列挙型）
4. switch-caseで定めたアイテムの価格を求める（ `itemCost` - 整数）

5. 変えるかどうかを　　if分で判定し、コンソールで結果を表示

答え

```
// 各アイテムの列挙（名前を多少に異なる可能性あり！）
enum ItemType
{
  PotionSmall,   // ポーション（小）
  PotionLarge,   // ポーション（大）
  WoodShield,    // 木の盾
  IronShield,    // 鉄の盾
  LeatherBoots,  // 皮のブーツ
  MagicCape,     // 魔法のマント
}

// 持っているお金
int playerMoney = 150;

// 買いたいアイテム
ItemType buyItem = ItemType.WoodShield; // なんでもOK

// 買いたいアイテムの価格
int itemCost;
switch(buyItem)
{
    case ItemType.PotionSmall:
        itemCost = 10;
        break;
    case ItemType.PotionLarge:
        itemCost = 25;
        break;
    case ItemType.WoodShield:
        itemCost = 50;
        break;
    case ItemType.IronShield:
        itemCost = 100;
        break;
    case ItemType.LeatherBoots:
        itemCost = 40;
        break;
```

```
    case ItemType.MagicCape:
        itemCost = 250;
        break;
}

// □□□□□□□□□□□□
if (itemCost <= playerMoney)
{
    Debug.Log("□□□□□□");
}
else
{
    Debug.Log("□□□□□□");
}
```

Revision #10
Created 2026-03-23 08:10:48 UTC by Menendez Francisco
Updated 2026-04-22 05:46:10 UTC by Menendez Francisco