

チーム名 : kuwax

作品タイトル「プログラミングで楽しむRPG」



ターゲット

プログラミングを軽くやったことがあり、プログラミング言語C#の学習をしたい人をターゲットにしている。

プレイ人数

windowsパソコンを用いて一人で行うゲーム

ゲームの目的

C#のコードで命令を与えて、主人公をゴールまでたどり着かせることが目的。
その過程で、ユーザーはプログラミングを学べる。



ゲームクリア・勝利条件

主人公がゴールにたどり着くことがゲームクリアの条件。

ゲームオーバー条件

「ギブアップ」ボタンを押せば、そのステージをあきらめることになるため、ゲームオーバーになる。

スイッチを押すメソッドで、スイッチのない場所でそのメソッドを使用したらゲームオーバーとなるものもある。



開発進捗

完成品である。

備考・注意点

画像などのグラフィックが改善の余地がある。



操作説明

※足りない場合は、ご自身で操作説明のページを追加してください。

初めの画面で矢印キーでステージを選んで、エンターキーでそのステージの中に入れる。
以降は、指示があるようにc#のコードを書くことで主人公を操作して、ゴールにたどり着くようにする。

人によっては、もうすでに習得済みなところのコードを書きたくないかもしれないので、「上」「右」「下」「左」「押す」「話しかける」などのボタンを用意している。このボタンを押せばコードを書かなくても主人公を操作できる。

「ギブアップ」を選択したら初めの画面に戻る。



スタッフリスト欄

※足りない場合は、ご自身でページを追加してください。

・ 桑原頌明

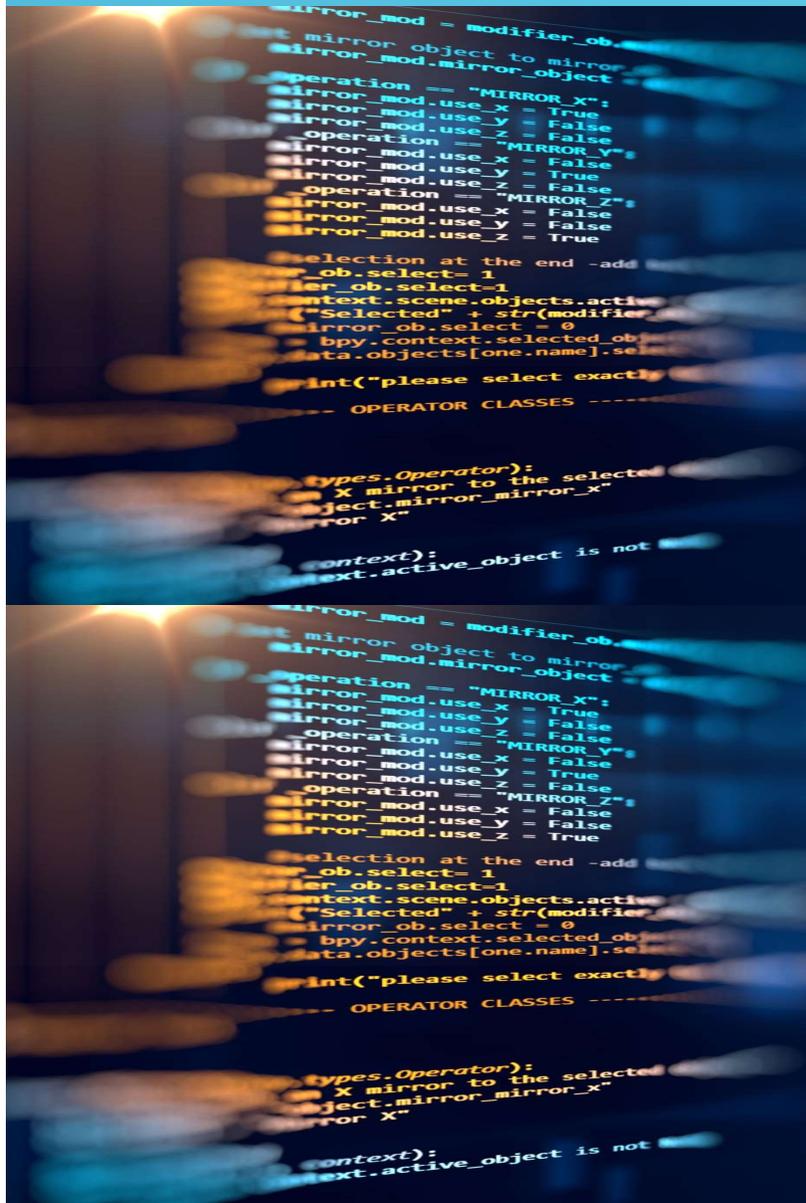
① 全般

② 全般

**以降のページに
企画書を追加してください**



プログラミングで楽しむRPG



プログラミングを学
べるゲームになっ
ている！

コードを書いて、主人公に命令を与えて、
迷路をクリアするゲームになっている！

- ▶C#でコードを書くと、その出力結果をもとに主人公が動いてくれる。

- ▶ Windowsに内蔵されているc#のコンパイラを呼び出して、その実行結果を取得する仕組みになっている。

▶ゲームをプレイしている様子を紹介する。

Form1

上

左 右

下

話しかける

オス

ここにコードを書く！

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade();
        //ここに命令を入力してください。
        //主人公が右へ行く 右を上や下や左などの他の方向に変えて
        System.Console.WriteLine("右へ行け");
    }
}
```

ギブアップ

実行ボタンを押してください。

実行 提出

Form1

上

左 右

下

話しかける

オス

ギブアップ

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade();
        //ここに命令を入力してください。
        //主人公が右へ行く、右を上や下や左などの他の方向に変えて
        System.Console.WriteLine("右へ行く");
    }
}
```

実行ボタンを押すと...

実行ボタンを押してください。

実行 提出

Form1

主人公が命令に従って移動してくれた!

上
左 右
下

話しかける
オス

ギブアップ

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade();
        //ここに命令を入力してください。
        //主人公が右へ行く 右を上や下や左などの他の方向に変えて
        System.Console.WriteLine("右へ行け");
    }
}
```

出力結果がここに表示される

右へ行け

実行 提出

Form1

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade();
        //ここに命令を入力してください。
        //主人公が右へ行く、右を上や下や左などの他の方向に変えて
        System.Console.WriteLine("右へ行け");
    }
}
```

上

左 右

下

話しかける

オス

G

モンスタ-や看板などのセリフがここに表示される!

ギブアップ

右へ行け

実行 提出

こんにちは と出力したらどいてあげよう。
(提出ボタンを押すとモンスターに対して出力できる。)

The image shows a software interface for a game or application. On the left is a grid-based map with a character and various objects. On the right is a code editor with C# code. Below the code editor are input fields and buttons. A large white arrow points from a top callout to a bottom callout. A red arrow points from the bottom callout to a button in the interface.

課された内容のコードを書いて、

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade();
        //ここに命令を入力してください。
        //主人公が右へ行く 右を上や下や左などの他の方向に変えて
        System.Console.WriteLine("こんにちは");
    }
}
```

提出する！

実行 提出

こんにちは と出力したらどいてあげよう。
(提出ボタンを押すとモンスターに対して出力できる。)

Form1

上
左 右
下
話しかける
オス
G

モンスターが
消えて通れる
ようになる！

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade();
        //ここに命令を入力してください。
        //主人公が右へ行く 右を上や下や左などの他の方向に変えて
        System.Console.WriteLine("こんにちは");
    }
}
```

ギブアップ

こんにちは

実行 提出

こんにちは と出力したらどいてあげよう。
(提出ボタンを押すとモンスターに対して出力できる。)

Form1

ゴールまでたどり着いてこの
ステージはクリアできた！

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade();
        //ここに命令を入力してください。
        //主人公が右へ行く、右を上や下や左などの他の方向に変えて
        System.Console.WriteLine("下へ行け");
    }
}
```

ギブアップ

下へ行け

実行

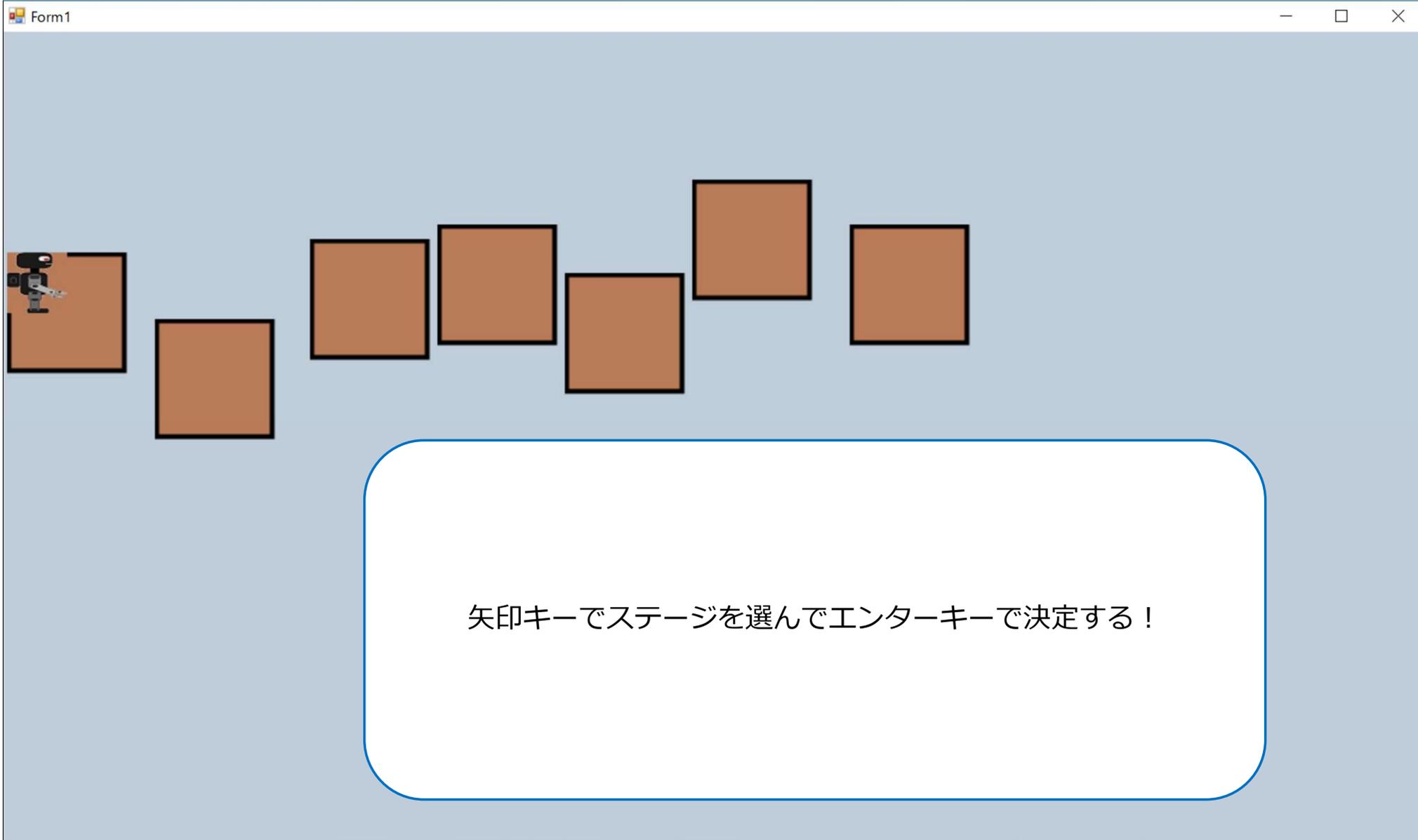
提出

こんにちは と出力したらどいてあげよう。
(提出ボタンを押すとモンスターに対して出力できる。)

- ▶ このステージでは、出力の方法を学べたが、ステージごとによって学べる内容はさまざまである。
- ▶ 入力、条件分岐、繰り返しなどが学べるようになっている。

▶今から、各ステージごとの説明をする





矢印キーでステージを選んでエンターキーで決定する！

Form1

上

左 右

下

話しかける

オス

アップ

G

実行

提出

class Test
{
 static void Main(string[] args)
 {
 Homemade homemade = new Homemade
 System.Console.WriteLine("話しかける");
 }
}

ここではhomemadeという
このゲーム独自のメソッドで、
メソッドについて学ぶ。

次はメソッドの使い方を学ぼう。
では、試しに5行目6行目の間にと
homemade.say()
と入力してみてね。

Form1

上

左 右

下

話しかける

オス

ギブアップ

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade();
        System.Console.WriteLine("話しかける");
        homemade.say();
    }
}
```

話しかける
このようにしてメソッドを呼び出して命令を与えられます

実行 提出

次はメソッドの使い方を学ぼう。
では、試しに5行目6行目の間にと
homemade.say()
と入力してみよう。

homemade.say()メソッドを
呼び出せた！

Form1

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        System.Console.WriteLine("話しかける");
    }
}
```

Homemade.onoffPush()
という扉を閉会するメソッド
を使って、扉を開けるよ
う指示されている！

実行 提出

1歩右にスイッチがあるので、そのスイッチの上に乗って
homemade.onoffPush();
というメソッドを実行して、扉を開閉しよう。
ただこのメソッドはスイッチがない上で実行すると
ゲームオーバーになるので気を付けて使おう。

Form1

上
左 右
下
話しかける
オス
ギブアップ
G

指示通りメソッドを呼び出すとドアが開いて、ゴールまでに行けるようになる！

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        homemade.onoffPush();
    }
}
```

ボタンを押す120820161005

実行 提出

ドアを開閉しました。

▶このようにして、このステージ
ではメソッドを学べた！

Form1

上
左 右
下 話しかける
オス

ギブアップ

実行 提出

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        //ここに命令を入力してください。
    }
}
```

次は条件分岐について学ぼう。if() というものを使うよ。
試しに命令を入力するところに
if(3+6 == 7+2) {
 System.Console.WriteLine("同じ");
}
else {
 System.Console.WriteLine("違う");
}
というコードを書いてみよう。

ここでは条件分岐について学ぶ。
If文の説明が書かれている。

Form1

上
左 右
下
話しかける
オス
ギブアップ
実行 提出

ドアを開閉しました。

class Test
{
 static void Main(string[] args)
 {
 Homemade homemade = new Homemade
 if(homemade.onoffCheck())
 {
 homemade.onoffPush();
 }
 }
}

スイッチのないところで
homemade.onoffPush()を実行し
たらゲームオーバーになるので、
homemade.onoffCheck()で真が
返ってきた時だけスイッチを押す
ようコードを書き、扉を開けたた
めゴールまで通れるようになっ
た！

▶このようにして、このステージでは条件分岐を学べた！



Form1

上
左 右
下
話しかける
オス

変数についての説明がされている！

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        System.Console.WriteLine("話しかける");
    }
}
```

話しかける

実行 提出

変数について学ぼう。
命令するところに
string hensuu;
hensuu = "hello";
System.Console.WriteLine(hensuu);
hensuu = "こんにちは";
System.Console.WriteLine(hensuu);
と入力してみよう。
string hensuu;の所で変数の宣言をしている。
hensuu = "hello";の所で変数に値を代入している。

Form1

Form1

上
左 右
下
話しかける
オス

入力を出力するコードを

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        string a=System.Console.ReadLine();
        System.Console.WriteLine(a);
    }
}
```

ギブアップ
右へ行け
実行 提出

提出したら、
モンスターが消えて、
ゴールまで通れるようになった！

私が文字列を入力するのでその文字列を出力しろ。
(System.Console.ReadLine()というメソッドは受け取った入力を返してくれる。)

▶このようにして、このステージでは変数と入力について学べた！

Form1

上

左 右

下

話しかける

オス

ギブアップ

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        //ここに命令を入力してください。
    }
}
```

実行 ボタン 提出

実行ボタンを押してください。

次は繰り返しについて学ぼう。
for(int i=0;i<5;i++)
{ System.Console.WriteLine("右へ行け");
}
これを命令するところに書くと右へ5回動いてくれる。
これは
int i=0.のところで変数を宣言する
i++はを1づつ足しているということだ。
i<5.はが5未満なら繰り返し続けるという意味だ。

For文についての説明
が書かれている！

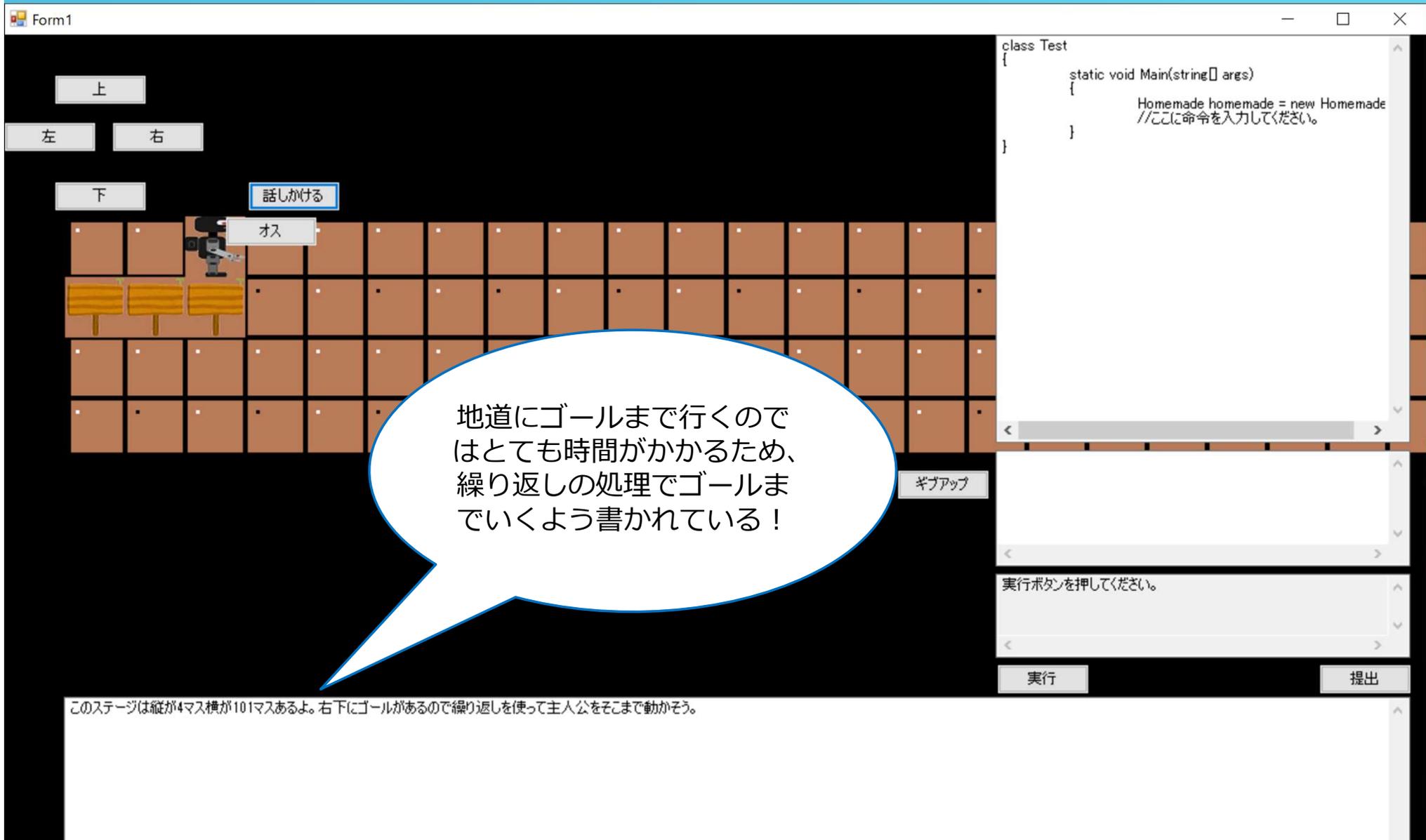
Form1

上
左 右
下
話しかける
オス
ギブアップ
実行 提出

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        //ここに命令を入力してください。
    }
}
```

実行ボタンを押してください。

このステージは縦が4マス横が101マスあるよ。右下にゴールがあるので繰り返しの処理を使って主人公をそこまで動かそう。



地道にゴールまで行くのはとても時間がかかるため、繰り返しの処理でゴールまでいくよう書かれている！

Form1

上
左 右
下
話しかける
オス

101回右へ移動するように指示されている。

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        //ここに命令を入力してください。
        for(int i=0;i<101;i++)
        {
            System.Console.WriteLine("右へ行け");
        }
    }
}
```

ギブアップ

右へ行け
右へ行け
右へ行け
右へ行け

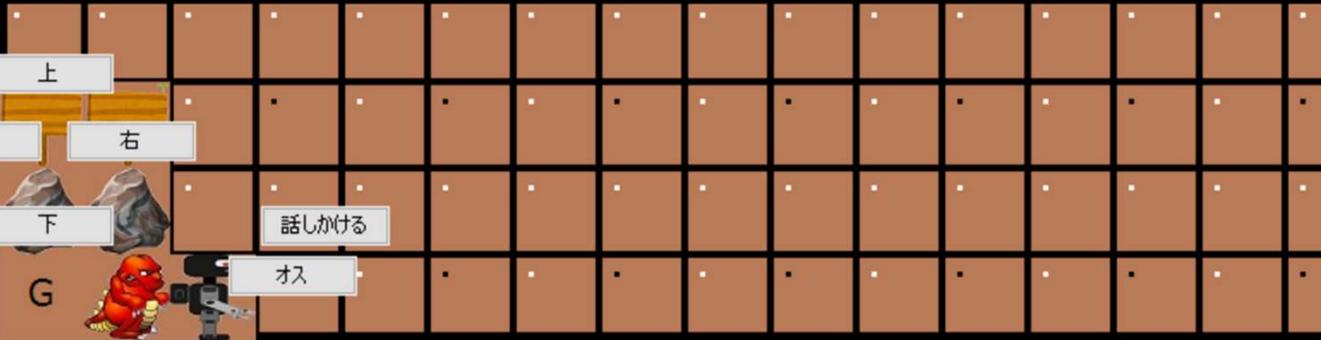
実行 提出

実行すると無事、101マス右へ移動してゴール手前までたどり着いた！

次は繰り返しについて学ぼう。
for(int i=0;i<5;i++)
{ System.Console.WriteLine("右へ行け");
}
これを命令するところに書くと右へ5回動いてくれる。
これは
int i=0.のところで変数を宣言する
i++はを1づつ足しているということだ。
i<5はiが5未満なら繰り返し続けるという意味だ。

- ▶このようにして、このステージでは繰り返し処理を学べるようになっている。
- ▶このステージはゴールまで横に101マスあるが、ステージ内容のデータを編集すれば簡単に、千マスにも1万マスにも変更できる。

Form1



上
左 右
下
話しかける
オス
G

このステージが何マスか数えるように指示された。

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        //ここに命令を入力してください。
        int cnt=0;
        for(int i=0;i<100;i++){
            cnt++;
            if(homemade.overCheck()) break;
        }
        System.Console.WriteLine(cnt);
    }
}
```

ギブアップ

こんにちは
こんにちは
こんにちは
こんにちは

実行 提出

このステージは横が何マスなのか調べて私に教えろ。(ただし、100マスは超えていない。)

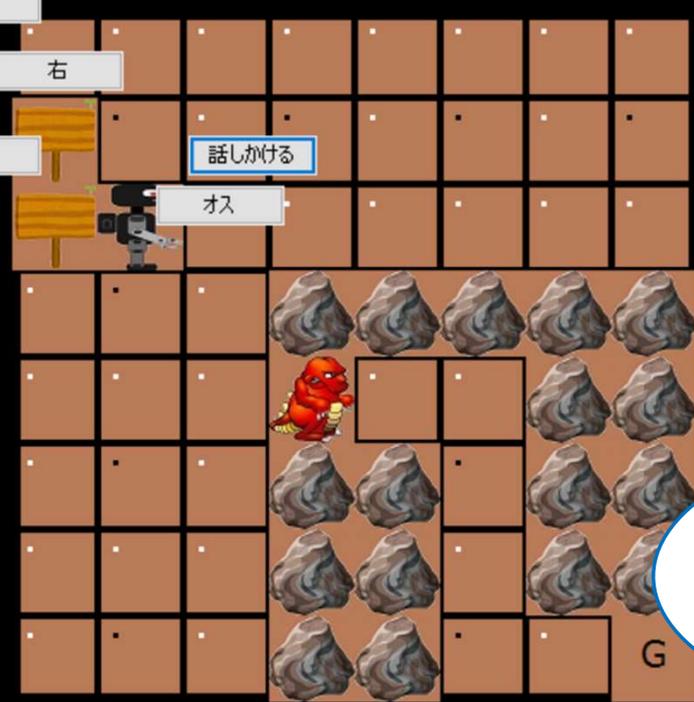
Form1

The bottom section is a console window with a 'ギョアップ' (Gyooapp) button on the left and an '実行' (Execute) button on the right. The console output shows the number '71'. A '提出' (Submit) button is located at the bottom right of the console area.

for文を駆使して何マスあるか数えて提出したら、ゴールまで通れるようになった！

▶このようにして、このステージでは
繰り返しについて再び学べた！

Form1



上
左 右
下
話しかける
オス
G

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        //ここに命令を入力してください。
    }
}
```

実行 ボタンを押してください。 提出

このステージでは配列について学ぼう。配列というのは複数の変数をひとまとめにする時に扱う。
int[] a = new int[5];
こういうコードを書いてみよう。
このコードは5個の変数をひとまとめにした配列aの宣言をしているよ。
次に
a[0]=34;
a[1]=82;
a[2]=56;
a[3]=39;
a[4]=21;

配列についての説明
が書かれている！

Form1

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        //ここに命令を入力してください。
    }
}
```

上
左 右
下
話しかける
オス
G

配列を使って解ける問題が課される!

実行ボタンを押してください。

実行 提出

私が君に命令を与えるから、君はその通りに動け。つまり、私が何回か入力を与えるから、それを改行して出力しろ。ただし、命令の終わりには0が来る。(0は出力しなくてよい。)

Form1

上
左 右
下
話しかける
オス

コードを書いて実行したら、無事、ゴールまでたどり着いた！

```
class Test
{
    static void Main(string[] args)
    {
        Homemade homemade = new Homemade
        string[] str = new string[10000];
        for(int i=0;i<10000;i++){
            str[i]=System.Console.ReadLine();
            if(str[i]=="0") break;
        }
        for(int i=0;i<10000;i++){
            if(str[i]=="0") break;
            System.Console.WriteLine(str[i]);
        }
    }
}
```

ギブアップ

右へ行け
右へ行け
右へ行け
下へ行け

実行 提出

私が君に命令を与えるから、君はその通りに動け。つまり、私が何回か入力を与えるから、それを改行して出力しろ。ただし、命令の終わりには0が来る。(0は出力しなくてよい。)

▶このようにして、このステージでは配列を学ぶことができた！

- ▶ xmlを読み取ってステージが作られるため、xmlを編集することで各ステージは自由に変更できる仕様になっている！

```
1 {?xml version="1.0" encoding="utf-8"?>
2 <things>
3   <heros>
4     <hero>
5       <x>0</x>
6       <y>0</y>
7       <direction>Right</direction>
8     </hero>
9   </heros>
10  <maps>
11    <map>
12      <numberOfGridWidth>7</numberOfGridWidth>
13      <numberOfGridHeight>7</numberOfGridHeight>
14      <gridWidth>50</gridWidth>
15      <gridHeight>50</gridHeight>
16      <code>
17        class Test{r\n
18          {r\n
19            static void Main(string[] args){r\n
20              {r\n
21                Homemade homemade = new Homemade();r\n
22                //ここに命令を入力してください。r\n
23                //主人公が右へ行く 右を上や下や左などの他の方向に変えて実行してみよう！r\n
24                System.Console.WriteLine("右へ行け");r\n
25                //主人公が看板やモンスターの前でこれをすると…r\n
26                System.Console.WriteLine("話しかける");r\n
27              }r\n
28            }r\n
29          }r\n
30        </code>
31      </map>
32    </maps>
33    <goals>
34      <goal>
35        <x>7</x>
36        <y>7</y>
37      </goal>
38    </goals>
39  </maps>
40 </things>
41 </xml>
```

- ▶ここで紹介したステージはあくまで1例であって、xmlにより自分でも簡単にステージは変更できるため、このゲームは無限の可能性が秘めている！

～終わり～