#プログラミング　#配列　#乱数　#条件判断　#ゲーム

Processing でプログラミングしよう(6)

Processing で配列を利用したプログラムをつくります。

材料（必要なもの）

Processing

1　配列を利用した簡単な宝探しゲーム

画面上に「＊」で表示された5つの箱があります。その中のどこか1箇所に宝が入っています。その宝が入っている箱をあてる宝探しゲームをつくりましょう。

```
＊ ＊ ＊ ＊ ＊
doko kana? input 0~4
2
hazure
```

2　配列を利用して5つの宝箱をつくります

配列とは，同じデータ型（整数，実数，文字列など）のデータを並べたものです。変数だと指定したデータ型のデータを1つしか記憶できませんが，配列を利用すれば複数のデータを記憶できます。配列に記憶された1つずつのデータを要素と呼び，要素の番号（添字と呼ぶ）で区別しています。

Processing では，配列を用意するとき，「データ型 [ ] 配列名 = new データ型 [要素数]」と記述するか，「データ型 [ ] 配列名 = {データ，データ，…}」と記述します。後者であれば，配列の各要素に初期値を記憶させることができます。配列の要素にデータを記憶するときは「配列名[添字] = データ」と記述します。なお添字の値は0から始まります。今回の宝探しゲームでは，宝の箱として5つの要素がある配列（配列名 takarabako）を用意し，初期値として0を記憶しておきます。プログラム例は次のようになります。なお，配列の要素の値0は宝が入っていないことを意味します。そのため，この takarabako の配列を表示したらどこにあたりが入っているかがわかってしまいます。そこで，画面に表示するための配列（配列名 hyoji）も用意し，初期値として「＊」を記憶しておきます。

```
1 int [] takarabako = {0, 0, 0, 0, 0};
2 String [] hyoji = {"*", "*", "*", "*", "*"};
```

3　乱数を利用して5つの宝箱のどこかに宝を入れます

乱数を利用して，宝箱の配列のどこかに宝を入れます。配列の初期値を0としたので，宝が入っている箱の値を1にします。画面の表示用の配列には，「A」と表示するようにします。

```
6  void draw() {
7    atari = int(random(0, 5));
8    takarabako[atari] = 1;
9  //  hyoji[atari] = "A";
10   println(hyoji);
11   println("doko kana? input 0~4");
12   noLoop();
13 }
```

右の図の4行目は「//」でコメントアウトしているので実行されない処理です。この「//」を削除すると，表示される配列のどこに宝が入っているかを確認できます。（宝が入っていることを確認するための処理です。）

右の図の6行目は，プログラムを実行したときに配列のどこを開けるか番号を入力するよう表示するプログラムです。

右の図の7行目は，この画面に表示する処理が1回だけ行われるようにするため，繰り返さないよう指示する処理です。これを入れないと次々と宝箱を表示します。忘れないよう気をつけてください。

4　キーボードで開ける宝箱を指定し，あたり判定をします

キーボードのキーを押したときに，当たり判定をします。まず，キーボードの操作があったときに処理を実行するために，keyPressed 関数を利用します。

開ける宝箱の指定には添字を利用します。5つの宝箱があるので，添え字は左から，0，1，2，3，4になります。そこで，キー入力した値が数値のときにあたり判定をします。Processing には key という変数が用意されており，これを利用すると押されたキーの文字コードが変数に記憶されます。Processing では0は 48，9は 57 という文字コードで処理されます。入力されたキーから 48 を引いて，入力された数値を0～9の値に変換しています。右のプログラムの4行目で，変換された値（押したキーの数字）を表示しています。変換がうまくできているかどうか，ここで確認ができます。

キー入力の処理ができたら，最後にあたり判定です。これは前回，前々回の学習で理解できているのではないでしょうか。プログラム例を見ないであたり判定のプログラムを記述できそうであれば，ぜひ挑戦してください。

なお，プログラム例は右上に記載してあります。あたり判定は，key2 の値が変数 atari と等しければ「atari」そうでなければ「hazure」と表示します。

```
15 void keyPressed() {
16   if(key >= 48 && key <=57) {
17     key2 = key - 48;
18     println(key2);
19   }
20   if(key2 == atari) {
21     println("atari");
22   }else{
23     println("hazure");
24   }
25 }
```

キーが押されたときで，さらにそのキーが数字キーだったときに，そのキーの値が０～９のどれかを判断しています

押されたキーの値と乱数で決めた宝が入っている要素の番号が等しいときに「atari」，そうでないときに「hazure」を表示します

5　完成したプログラム

```
1  int [] takarabako = {0, 0, 0, 0, 0};
2  String [] hyoji = {"*", "*", "*", "*", "*"};
3  int atari;
4  int key2;
5
6  void draw() {
7    atari = int(random(0, 5));
8    takarabako[atari] = 1;
9  //  hyoji[atari] = "A";
10   println(hyoji);
11   println("doko kana? input 0~4");
12   noLoop();
13 }
14
15 void keyPressed() {
16   if(key >= 48 && key <=57) {
17     key2 = key - 48;
18     println(key2);
19   }
20   if(key2 == atari) {
21     println("atari");
22   }else{
23     println("hazure");
24   }
25 }
```

配列を用意して初期値を記憶させます

atari は宝が入っている配列の要素の番号を記憶します
key2 はキーボードから入力した要素の番号を記憶します

乱数で０から５未満の値を整数化して変数 atai に記憶します
次に配列の要素の番号が変数 atai である要素にあたりを意味する値１を記憶します

コツ（留意点）

赤いバーにあたる，ゲームオーバーになる条件を，比較演算子や論理演算子を用いて表現することがポイントです。

作成者

北海道札幌北高等学校　前田健太朗

k_maeda@hokkaido-c.ed.jp