2021 年度 ソフトウェア実験 A2（関数その２(Processing)）

1. 目的

昨年度後半の 2 年プログラミング I では C 言語の関数の基礎について学んだ。関数は自分で新しく命令を作るものであり、本格的なソフトウェアを作る上で必ず必要になるものである。そこで本実験 1 週目では関数を用いたプログラムを作ることで、関数に対する理解を深める。また 1 年次では情報リテラシーで Processing を用いてグラフィカルなプログラム作成の基礎を学んだ。そこで特に今週は Processing での開発行程や関数について理解を深める。

2. 原理（Processing の関数について）

Processing の関数は C 言語の関数の仕組みと同様である。主な違いを以下に挙げる。

1. C 言語のポインタが使えない。

C 言語では 1 つの値を返す時には return、複数の値を返す時にはポインタを利用する方法を良く使うが、Processing では C 言語のような「*」や「&」を使ったポインタ使用法が使えない。

（C 言語でポインタを利用する関数の例）

```
void func(int *a, int *b){
	*a = 1;	*b = 2;
}
void main()[
	int a, b;
	func(&a, &b);
}
```

Processing では比較的小規模なプログラムを作ることが多く、大域変数を積極的に使う習慣がある。

```
int a, b;

void func(){
	a = 1;  b = 2;
}
void draw(){
	func();
}
```

2. 特殊な予約済み関数名がある。

Processing では keyPressed() や keyReleased()、mousePressed()、

mouseReleased()といった特殊な関数名が予約されている。これらの関数はユーザが所定の操作を行った際に実行されるものであり、その実行内容はプログラマが自由に実装することができる。例えば'a'キーを押すと長方形を描く関数を作りたければ以下の様にすれば良い。

（方法その１）

```
void keyPressed(){
    if (key == 'a'){
        rect(10, 10, 50, 50);
    }
}
```

（方法その２）

```
void keyReleased(){
    if (key == 'a'){
        rect(10, 10, 50, 50);
    }
}
```

（参考：方法その３）

```
void draw(){
    if (keyPressed == true){  //特殊な変数 keyPressed を利用
        if (key == 'a'){
            rect(10, 10, 50, 50);
        }
    }
}
```

3. 課題：　じゃんけんゲームの実装

3-1　概要

Processing を使い、以下に示す要領で一人のプレイヤーがコンピュータと対戦するじゃんけんゲームを作れ。

3-2　じゃんけんの手や勝敗について

前回の C 言語と同様のため省略する。

3-3　ゲームの状態遷移について

今回実装するじゃんけんではゲームの流れに沿って画面構成が変化する。そこでゲーム状

態がどのように遷移するかを図で設計する。以下の図１がその設計例である。

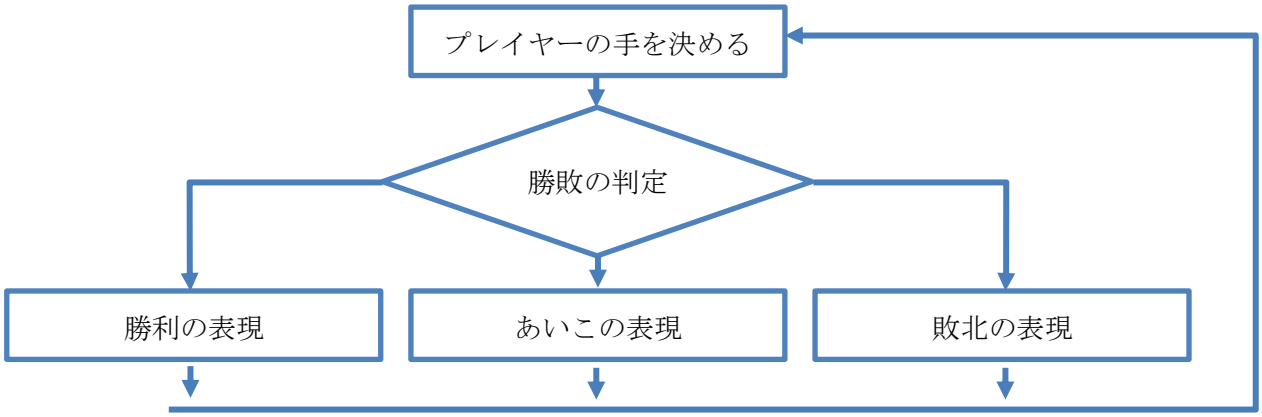

図１　ゲームの状態遷移

基本的には図１の四角ブロック毎に画面の表現が変わるものとする。これを Processing で実現するには、各状態に番号をふり、現在の状態を示す大域変数 status を用意して管理する方法がある。そのソース例を以下に示す。

```
int status = 0;

void draw(){
  switch (status) {
  case 0:
    状態 0 の時の処理
    break;
  case 1:
    状態 1 の時の処理
    break;
 (略)
}
```

ただしこの方法では状態をマジックナンバーで表しており、数値 0 がどんな状態なのか分かりづらい。こんな時に C 言語では#define を使って数値を文字列で表現するが、Processing の場合#define が無いので、以下のようにして数値に文字列を割り当てる。

```
public static final int MATCH = 0;
public static final int TIE   = 1;
public static final int LOSE  = 2;
public static final int WIN   = 3;
```

```
int status = MATCH;

void draw(){
  switch (status) {
  case MATCH:
    プレイヤーの手を決める処理
    break;
  case TIE:
    あいこの表現
    break;
  case LOSE:
    敗北の表現
    break;
  case WIN:
    勝利の表現
    break;
  }
}
```

3-4　実装する関数について

本課題では以下の関数を実装すること。関数は適宜増やすまたは仕様変更などして良いものとする。

≪最低限用意する大域変数≫

```
int status;          // ゲームの状態番号を入れる
int player;          // プレイヤーの手番号を入れる
int computer;        // コンピュータの手番号を入れる
```

≪互いに手を決める関数≫

関数の書式：

```
void match();
```

引数：

なし

返り値：

なし

機能：

乱数でコンピュータの手（0 or 1 or 2）を決め、大域変数 computer に入れる。

```
computer = (int)random(3);
```

またプレイヤーの手を入力し、大域変数 player に入れる。そして入力が終わったら勝敗判定の関数を呼ぶ。

≪勝負判定を行う関数≫

関数の書式：

```
void judge();
```

引数：

なし

返り値：

なし

機能：

プレイヤーの手(大域変数 player)とコンピュータの手(大域変数 computer)から勝敗を判定し、結果に応じて状態番号(大域変数 status)の値を変更する。

≪勝利を知らせる関数≫

関数の書式：

```
void win();
```

引数：

なし

返り値：

なし

機能：

プレイヤーが勝ったことを示す表示を行う。

≪あいこを知らせる関数≫

関数の書式：

```
void tie();
```

引数：

なし

返り値：

なし

機能：

あいこを示す表示を行う。

≪敗北を知らせる関数≫

関数の書式：

```
void lose();
```

引数：

なし

返り値：

なし

機能：

プレイヤーが負けたことを示す表示を行う。

≪雛形≫

```
001: //
002: // 手の画像素材：http://www.irasutoya.com/2013/07/blog-post_5608.html
003: //
004:
005: // ゲームの状態を示す番号と状態名の define
006: public static final int MATCH = 0;  // 手を決める状態
007: public static final int TIE   = 1;  // あいこの状態
008: public static final int LOSE  = 2;  // 敗北の状態
009: public static final int WIN   = 3;  // 勝利の状態
010:
011: // 大域変数の宣言
012: int status = MATCH;
013: int player   = 0;
014: int computer = 0;
015: PImage[] hand;
016: PFont font;
017:
018: //--------------------------------------
019: // 初期設定(実行時に 1 回実行される)
020: //
021: void setup() {
022:   // 画面の設定
023:   size(700, 500);
024:   smooth();
025:
026:   // 画像ファイルの設定
027:   hand = new PImage[3];
028:   hand[0] = loadImage("image/gu.png");
029:   hand[1] = loadImage("image/choki.png");
```

```
030:   hand[2] = loadImage("image/pa.png");
031:
032:   // フォントの設定
033:   // println(PFont.list());
034:   font = createFont("メイリオ", 32);   // フォントを設定する
035:   textAlign(CENTER, CENTER);
036: }
037:
038: //--------------------------------------
039: // 実行停止するまで何度も繰り返し実行される
040: //
041: void draw() {
042:   background(255, 255, 200);
043:   switch (status) {
044:   case MATCH:
045:     match();
046:     break;
047:   case TIE:
048:     tie();
049:     break;
050:   case LOSE:
051:     lose();
052:     break;
053:   case WIN:
054:     win();
055:     break;
056:   }
057: }
058:
059: //--------------------------------------
060: // 画面に両者の手を表示する
061: //
062: void showHand() {
063:   imageMode(CENTER);
064:   image(hand[player  ], width/4, height/2);
065:   image(hand[computer], width*3/4, height/2);
066:   fill(0);
067:   textFont(font);
```

```
068:   text("あなた", width/4, height*1/5);
069:   text("computer", width*3/4, height*1/5);
070: }
071:
072: //---------------------------------------
073: // 両者の手を決める状態
074: //
075: void match() {
076:   computer = (int)random(3);
077: //略
078: }
079:
080: //---------------------------------------
081: // 勝敗判定し status の値を変更する
082: //
083: void judge() {
084: }
085:
086: //---------------------------------------
087: // あいこの結果表示
088: //
089: void tie() {
090: }
091:
092: //---------------------------------------
093: // 敗北の結果表現
094: //
095: void lose() {
096: }
097:
098: //---------------------------------------
099: // 勝利の結果表現
100: //
101: void win() {
102: }
```

4. 完成度を高めるために

まずは最低限の仕様を満たすプログラムを完成させること。その上で以下の面を改造してより完成度を高めて欲しい。

1. プレイヤーの手をマウスクリックで選べるようにする。

2. 画面の色変更、文字のフォントサイズ、視覚効果を加える、音を付けるなど、見た目のゲームらしさを演出する。

5. レポートについて

今回作成した Processing プログラムについては、作成したものをレポートにまとめて WebClass に提出する。その際どんな点を工夫したか分かるようにすること。