# 2021 年度 ソフトウェア実験 A2（関数）

HI3　42 号　山口惺司

## 1. 目的

昨年度の 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。

## 2. 原理（関数）

関数は、プログラミング言語が提供する基本命令や数式を組み合わせて一つの処理を作り、関数名を命名して新たな命令として呼び出せるようにしたものである。実は printf()や scanf()もデータの入出力用にあらかじめ用意された関数である。

関数を利用することで以下のメリットが生まれる。

1. 一つの機能を一つの関数にまとめることで、プログラムを意味的に整理整頓する。
2. プログラムを複数の関数に分割することで、1 つ 1 つの長さを短くすることで、プログラムの見かけを整理整頓する。
3. プログラムで何度も呼び出される部分を一つにまとめることでプログラムを短くする。

関数はプログラミング言語によって「関数(function)」、「手続き(procedure)」、「サブルーチン(subroutine)」とも呼ばれる。

関数は以下の構造をしている。

```
返り値の型 関数名(引数){                    処理
        :                    :
        return 値;
}（例）
int sample(int x){
        int y = 2 * x;
        return y;
}
```

通常は return 文で処理結果を呼び出し元に返すが、複数の値を返すときなどは引数とポインタを利用する。上述の例をポインタで書き直すと以下のようになる。

```
void sample(int x, int *y){
        *y = 2 * x;
}
```

関数を上手に設計すると分かりやすいプログラムを作れる。逆に、関数を上手く設計できないと分かりづらいプログラムになってしまう。

3. 課題： じゃんけんゲームの実装

## 3-1 概要

C 言語を使い、以下に示す要領で一人のプレイヤーがコンピュータと 1 回（引き分けのときは勝負がつくまで繰り返し）対戦するじゃんけんゲームを作れ。

## 3-2 じゃんけんの手について

じゃんけんの手は以下のように整数値で表すものとする。

・ぐー　1

・ちょき 2

・ぱー　3

プレイヤーはキーボードから上記の値を入力する。コンピュータは乱数を使って手を決める（詳細は後述)。それぞれの手と勝敗を表 1 にまとめる。手のパターン数は全部で 9 通りあり、if～else if～else if～の繰り返し 9 個で実装可能だが、勝敗の規則性を利用すればもっと効率良く実装できる。

表 1 それぞれの手に対応する勝敗

## 3-3 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。

≪プレイヤーの手を入力する関数≫

関数の書式：

int setPlayerHand();

引数：

・なし

返り値：

プレイヤーの手（ぐー、ちょき、ぱーのいずれかを表す数値）

機能：

キーボードからプレイヤーの手を読み込んで返り値で返す。ぐー、ちょき、ぱー以外の数値が入力された場合は正しい入力が行われるまで再入力を促す。

≪コンピュータの手を決める関数≫

関数の書式：

int setComputerHand();

引数：

・なし

返り値：

コンピュータの手（ぐー、ちょき、ぱーのいずれかを表す数値）機能：

乱数（詳細後述）でコンピュータの手を決め、結果を返す。

≪勝負判定を行う関数≫

関数の書式：

int judge(int player, int computer);

引数：

・int player プレイヤーの手（ぐー、ちょき、ぱーのいずれかを表す数値）

・int computer コンピュータの手（ぐー、ちょき、ぱーのいずれかを表す数値）

返り値：

勝ち、負け、あいこのいずれかを表す数値

設計例：-1…プレイヤーの負け、0…あいこ、1…プレイヤーの勝ち

機能：

プレイヤーの手とコンピュータの手とを比較し、勝敗を判定し、その結果を返す。

≪結果を表示する関数≫

関数の書式：

void printResult(int result, int player, int computer);

引数：

・ int result　　勝敗結果を表す数値。関数 judge()からの返り値。

・ int player　　プレイヤーの手（ぐー、ちょき、ぱーのいずれかを表す数値）

・ int computer コンピュータの手（ぐー、ちょき、ぱーのいずれかを表す数値）

返り値：

・なし

機能：プレイヤーの手とコンピュータの手、勝敗結果を画面に表示する。

≪main 文≫

関数の書式：

int main();

引数：

・ なし

返り値：

return 0 で終了する。

機能：

じゃんけんを一回戦分行う。

実装例：

```
// 勝敗結果の定義
#define TIE  0     //あいこ
#define WIN  1     //プレイヤーの勝ち
#define LOSE -1    //プレイヤーの負け
int main(){   int player;    //プレイヤーの手   int computer;   //コンピュータの手   int result;    //勝敗結果
   srandom(time(0));              //乱数の初期化
   do {
     computer = setComputerHand();   //コンピュータの手を決める
る
     player = setPlayerHand();      //プレイヤーの手を決める      result = judge(player, computer);   //勝負判定      printResult(result, player, computer); //結果表示  } while (result == TIE);
  return 0;
}
```

## 3-4 実装

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <time.h>
5 #include <unistd.h>
6 #define TIE 0
7 #define WIN 1
8 #define LOSE -1
9
10 int setPlayerHand();
11 int setComputerHand();
```

```
12 int judge(int player, int computer);
13 void printResult(int result, int player, int computer);
14
15 int main(){
16    int player;
17    int computer;
18    int result;
19
20    srandom(time(0));
21
22    while(1){
23
24       computer = setComputerHand();
25
26       player = setPlayerHand();
27       result = judge(player, computer);
28       printResult(result, player, computer);
29       if(result != TIE){
30          break;
31       }
32    }
33
34    return 0;
35 }
36
37 int setComputerHand(){
38    long a;
39    a = random() % 3 + 1;
40
41    return a;
42 }
```

```
43
44 int setPlayerHand(){
45    char playerhand[80];
46    while(1){
47       system("clear");
48       puts("じゃんけんの手を決めてください(例：ぐー、ちょき、ぱー)：");
49       scanf("%s",playerhand);
50       if(strcmp(playerhand, "ぐー") == 0){
51          return 1;
52       } else if(strcmp(playerhand, "ちょき") == 0){
53          return 2;
54       } else if(strcmp(playerhand, "ぱー") == 0){
55          return 3;
56       } else {
57          puts("値が無効です");
58          sleep(1);
59       }
60    }
61 }
62
63
64 int judge(int player, int computer){
65    if(player == 1 && computer == 2 || player == 2 && computer == 3 || player == 3 && computer == 1){
66       return WIN;
67    } else if(player == 1 && computer == 3 || player == 2 && computer == 1 || player == 3 && computer
== 2){
68       return LOSE;
69    } else {
70       return TIE;
71    }
```

```
72 }
73
74 void printResult(int result, int player, int computer){
75   sleep(1);
76   if(player == 1){
77     puts("あなた：ぐー");
78   } else if(player == 2){
79     puts("あなた：ちょき");
80   } else if(player == 3){
81     puts("あなた：ぱー");
82   }
83   if(computer == 1){
84     puts("コンピュータ：ぐー");
85   } else if(computer == 2){
86     puts("コンピュータ：ちょき");
87   } else if(computer == 3){
88     puts("コンピュータ：ぱー");
89   }
90
91   if(result == LOSE){
92     puts("\e[34m\e[47m あなたの負けです\e[0m");
93   } else if(result == TIE){
94     puts("\e[33m 引き分けです\e[0m");
95     sleep(1);
96   } else if(result == WIN){
97     puts("\e[31m\e[47m あなたの勝ちです\e[0m");
98   }
99 }
```

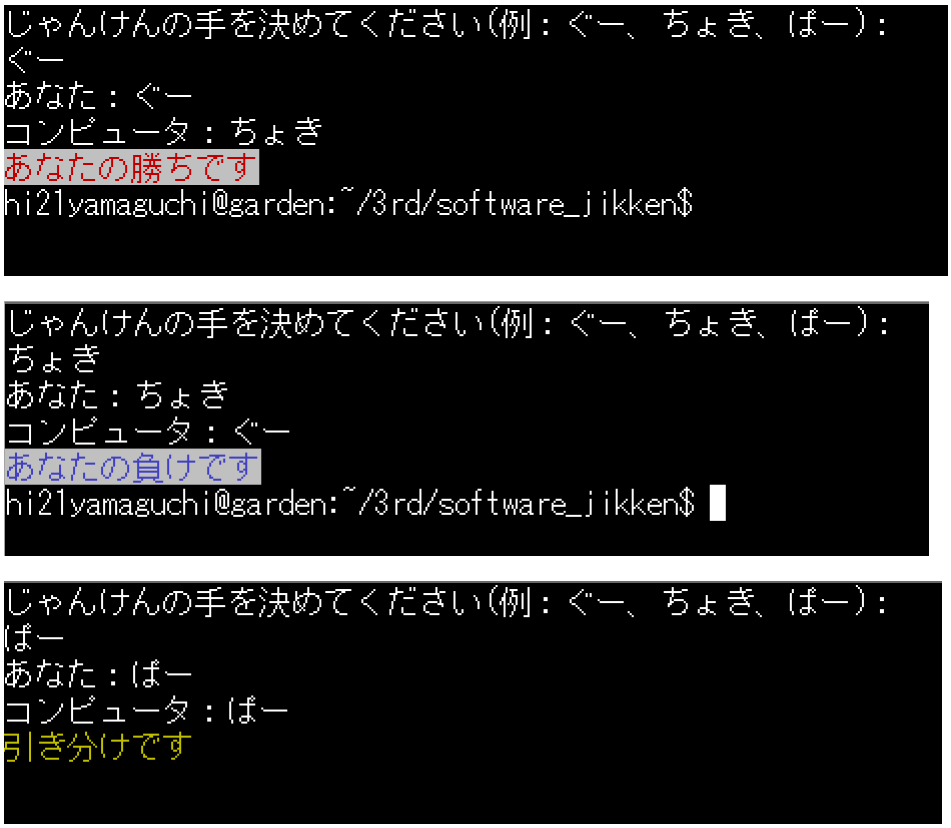


**図 3.1 実行結果写真 3 枚**

## 4 工夫した点

・ぐー、ちょき、ぱーの入力を 1,2,3 で行うのではなく、「ぐー」「ちょき」「ぱー」と入力させるようにした。

・表示させる文字の色や背景色を変え、見やすくした。

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

HI3　42 号　山口惺司

## 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。

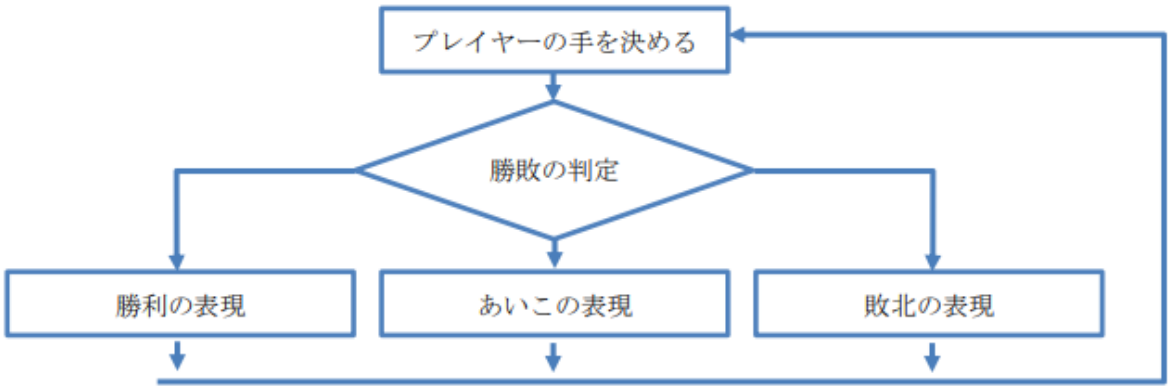


図 1　ゲームの状態遷移

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。

## 3-5 実装

```
//
// 手の画像素材：http://www.irasutoya.com/2013/07/blog-post_5608.html
//

// ゲームの状態を示す番号と状態名の define
public static final int MATCH = 0;  // 手を決める状態
public static final int TIE   = 1;  // あいこの状態
public static final int LOSE  = 2;  // 敗北の状態
public static final int WIN   = 3;  // 勝利の状態

// 大域変数の宣言
int status = MATCH;
int player   = 0;
int computer = 0;
PImage[] hand;
PFont font;

//--------------------------------------
// 初期設定(実行時に 1 回実行される)
//
void setup() {
  // 画面の設定
  size(700, 500);
  smooth();
```

```
  // 画像ファイルの設定
  hand = new PImage[3];
  hand[0] = loadImage("image/gu.png");
  hand[1] = loadImage("image/choki.png");
  hand[2] = loadImage("image/pa.png");

  // フォントの設定
  // println(PFont.list());
  font = createFont("メイリオ", 32);   // フォントを設定する
  textAlign(CENTER, CENTER);
}

//--------------------------------------
// 実行停止するまで何度も繰り返し実行される
//
void draw() {
  background(255, 255, 200);
  switch (status) {
  case MATCH:
    match();
    break;
  case TIE:
    tie();
    break;
  case LOSE:
    lose();
    break;
  case WIN:
    win();
    break;
  }
}

//--------------------------------------
// 画面に両者の手を表示する
//
void showHand() {
  imageMode(CENTER);
  image(hand[player  ], width/4, height/2);
  image(hand[computer], width*3/4, height/2);
  fill(0);
  textFont(font);
  text("あなた", width/4, height*1/5);
  text("computer", width*3/4, height*1/5);
}

//--------------------------------------
```

```
// 両者の手を決める状態
//
void match() {

  computer = (int)random(3);
  showHand();
  strokeWeight(5);

  //ぐー
  fill(255);
  if (mouseX >= 75 && mouseX <= 225 && mouseY >= 400 && mouseY <= 475) {
    fill(255, 255, 0);
  }
  rect(75, 400, 150, 75, 10);
  //ちょき
  fill(255);
  if (mouseX >= 275 && mouseX <= 425 && mouseY >= 400 && mouseY <= 475) {
    fill(255, 255, 0);
  }
  rect(275, 400, 150, 75, 10);
  //ぱー
  fill(255);
  if (mouseX >= 475 && mouseX <= 625 && mouseY >= 400 && mouseY <= 475) {
    fill(255, 255, 0);
  }
  rect(475, 400, 150, 75, 10);

  //勝負
  fill(255);
  if (mouseX >= 312.5 && mouseX <= 387.5 && mouseY >= 200 && mouseY <= 275) {
    fill(255, 255, 0);
  }
  rect(312.5, 200, 75, 75, 10);


  fill(0);
  text("ぐー", 150, 432.5);
  text("ちょき", 350, 432.5);
  text("ぱー", 550, 432.5);
  text("勝負", 350, 230);


  //略
}

//--------------------------------------
// 勝敗判定し status の値を変更する
//
```

```
void judge() {
  if (player == computer) {
    status = TIE;
  } else if (player == 0 && computer == 2 || player == 1 && computer == 0 ||
player == 2 && computer == 1) {
    status = LOSE;
  } else {
    status = WIN;
  }
}


//--------------------------------------
// あいこの結果表示
//
void tie() {
  showHand();
  fill(0);
  textSize(200);
  text("TIE", width/2+10, height/2-5);
  fill(0, 255, 0);
  textSize(200);
  text("TIE", width/2, height/2);
  nextGame();
}

//--------------------------------------
// 敗北の結果表現
//
void lose() {
  showHand();
  fill(0);
  textSize(200);
  text("LOSE", width/2+10, height/2-5);
  fill(0, 0, 255);
  textSize(200);
  text("LOSE", width/2, height/2);
  nextGame();
}

//--------------------------------------
// 勝利の結果表現
//
void win() {
  showHand();
  fill(0);
  textSize(200);
```

```
  text("WIN", width/2+10, height/2-5);

  fill(255,0,0);
  textSize(200);
  text("WIN", width/2, height/2);
  nextGame();
}

void nextGame() {
  fill(255);
  if (mouseX >= 275 && mouseX <= 425 && mouseY >= 400 && mouseY <= 475) {
    fill(255, 255, 0);
  }
  rect(275, 400, 150, 75, 10);
  fill(0);
  textSize(32);
  text("続ける", 350, 432.5);
}

void mouseClicked() {
  //match 状態
  if (status == MATCH) {
    //ぐー
    if (mouseX >= 75 && mouseX <= 225 && mouseY >= 400 && mouseY <= 475) {
      player = 0;
    }
    //ちょき
    if (mouseX >= 275 && mouseX <= 425 && mouseY >= 400 && mouseY <= 475) {
      player = 1;
    }
    //ぱー
    if (mouseX >= 475 && mouseX <= 625 && mouseY >= 400 && mouseY <= 475) {
      player = 2;
    }
    //勝負
    if (mouseX >= 312.5 && mouseX <= 387.5 && mouseY >= 200 && mouseY <= 275) {
      judge();
    }
  }

  //続ける
  if (status == TIE || status == LOSE || status == WIN) {
    if (mouseX >= 275 && mouseX <= 425 && mouseY >= 400 && mouseY <= 475) {
      status = 0;
    }
  }
}
```


図 3.1 実行結果写真 5 枚

## ４工夫した点

・プレイヤーの手を決めるときにボタンを採用した、ボタンの上にカーソルがあるとボタンを光らせ、選択している手がわかりやすいようにした。

・mouseClicked 関数を使って動作を安定させた。

・文字の色や陰にこだわり、見やすくした。