2021 年度 ソフトウェア実験 A2（関数）

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

		コンピュータの手		
		ぐー(1)	ちょき(2)	ぱー(3)
プレイヤーの手	ぐー(1)	あいこ	勝ち	負け
	ちょき(2)	負け	あいこ	勝ち
	ぱー(3)	勝ち	負け	あいこ

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
        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;
}
```

4. 乱数について

以下のソースコードは擬似乱数を 5 個表示するものである。乱数生成に最低限必要なのは赤字の 3 カ所である。

1. 乱数を使うために stdlib.h を include する。
2. 擬似乱数の出方を毎回変えるために、main 文冒頭で srandom()を一回だけ実行する。
3. random()を実行すると 0 から$2^{31}-1=2147483647$までの範囲の整数が返される。

```
#include <stdio.h>
#include <stdlib.h>          // random()、srandom()を利用可能にする

int main(void){
    long a;
    int i;

    srandom(time(0));        // 乱数生成パターンの初期化
    for (i = 0; i < 5; i++){
        a = random();         // 実行する度に 0 から pow(2,31)-1 までの乱数を生成する
        printf ("%d¥n", a);
    }
    return 0;
}
```

（実行例）

```
$ ./a.out
44493353
1520418501
1982776176
1206950013
775644769
```

random()から 0 から 2 までの整数を簡易に得るならば、演算子%を使って 3 で割った余りを求めればよい。

5. 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. プレイヤーの入力に対するエラーチェックを強化する。

   簡易には `scanf("%d", &hand);`のように数値入力させるのがシンプルだが、文字を入力した場合などに問題が生じる。この問題に対応する一方法として、文字配列を用意し、いったん scanf の%s で文字列として一行入力した上で atoi()関数を使って文字列を整数に変換する方法が挙げられる。文字列から整数への変換に失敗した場合 atoi()は 0 を返すので、この値を基にエラーかどうかを判定できる。

2. 結果表示など、随所でゲーム性を高める演出を加える。

   例えば結果がすぐに出ると味気ないので、sleep()命令を使い、プレイヤーを少しじらさせて結果を出すなど。