プログラミング３自由課題レポート

30114150 KIMHYUNWOO

12.23.2021

概要

本課題は重複していないｎ桁の数字を当てるゲームであるHitandblowと５個サイコロを１２回振って役を埋めた後総和で勝負を決めるゲームであるヨット(YACHT)を具現したプログラムである。

本課題で使用された主な技術には、制御構文、関数、ポインター、構造体、ファイルの操作、列挙体、文字列の操作、Mathライブラリー、ヘッダーファイル、入出力ポインター、構造体ポインター、Malloc&Freeがある。

本プログラムは３つのｃファイルと２つのヘッダファイルで分割していて、メイン関数のあるmain.c、ヨットゲームを実行するyacht.c、Hitandblowゲームを実行するHitandblow.cとゲームの関数を集めた各ゲームのヘッダファイルで構成されている。

本課題は概要、ソースコード付きファイルの関数、機能の解説、作動例、感想の順に記述されている。

Main.c

Main.cではバウンダリーチェックの関数実装、ファイルの読み取り、プレイヤーの名前入力、ゲームの選択などが行われている。バウンダリーチェックの関数をメイン関数に実装した理由は、他のゲーム関数で共通的に使用される機能の重複実装が防止のためである。

```
bool boundarycheck(int from, int to, int *target){
    return (*target < from || *target > to)?false:true;
}
```

仕組みとしてはまずファイルはAppend＋形式で読み取って、ゲームの結果を一番後ろに追加していく。もし、ファイルが確認できない場合にはファイルの消失と見なしてプログラムを終了させる。

```
    FILE *fp = fopen("history.txt","at+");
    if(fp == NULL) {
        printf("History file is detroyed, or doesn't exist!!");
        return -1;
    }
```

ファイルが正常に読み取れた場合、名前を入力してからプレイするゲームを選択、実行する。以後、ゲームが終了したらポインターを解放して終了する仕組みになっている。

```
    printf("Enter your name : ");
    scanf("%s", name);
    printf("1: Hit and blow\t\t2: Yacht Dice\n");
    printf("Select Game : ");
    scanf("%d", &gamemode);
    switch(gamemode){
        case 1:
            printf("Hit and blow Game is initialized\n");
            Hitandblow(name, fp);
            break;
        case 2:
            printf("YachtDice Game is initialized\n");
            YachtDice(name, fp);
            break;
    }
    //  最後にテキストファイルを閉じてからファイルポインターを解除する。
    fclose(fp);
```

Hitandblow.h

本ヘッダーファイルではHitandblow.cで具現した関数を宣言している。Boundarycheck関数は外部に定義されていることを示すためにextern識別子を付けている。

```
// Header file -> Hitandblow.h

#ifndef __HITANDBLOW_H__
#define __HITANDBLOW_H__

extern bool boundarycheck(int from, int to, int *target);
void Hitandblow(char *name, FILE *fp);

#endif
```

Hitandblow.c

Hitandblowの仕組みは桁数の指定→乱数の生成及び調整→当てる度にストライクやボールの判定→機会を尽くすまで正解が出なければ正解の公開、あるいは正解が出れば終了→内容の記録の５つの過程からなる。

まず、桁数の指定は一般的なscanf関数で読み取っていて、読み取った桁数を２倍した値で１を引いた値を最大の機会にしている。

```
    printf("How many digits do you want? ");
    scanf("%d", &digit);
    MAXCOUNT = 2*digit - 1;
```

以後、乱数生成の時にポインターを用いて桁数に合わせて１桁ずつ生成しているが、内部ループで以前生成された乱数と重複しないようにしている。

```
    for(target_index = target; target_index != target + digit;target_index++){
        int* loop_index;
        *target_index = rand()%10;
        for(loop_index = target; loop_index != target_index; loop_index++){
            if(*target_index == *loop_index){
                target_index--;
                break;
            }
        }
    }
    printf("\n%d digits number is created!", digit);
```

続くループの判定にはMAXCOUNTが 0 になるまで（機会を尽くすまで）ループごとに数字を読み取る。この数字は整数型であるため、配列型の乱数と桁を比較するためにインデクスを用いて 1 桁ずつ比較する方法を使用している。

```
while(MAXCOUNT > 0){
        printf("You have %d chances\n", MAXCOUNT);
        MAXCOUNT--;
        //do{
            printf("Enter the number : ");
            scanf("%d", &num);
```

ループごとにストライクやボールのカウンターを初期化し、配列のインデクスも配列長に合わせて後ろから探索できるように初期化する。以後、ループの最初に読み取った整数の残りを後ろから乱数配列と比較していきながら、値が含まれているかと桁も一致しているかを判断してストライクやボールのカウンターを操作している。

```
       strike = 0;
        ball = 0;
        cur_index = digit-1;
        while(num > 0){
            current_number = num % 10;
            num /= 10;
            for(index = digit-1; index != -1; index--){
                if(target[index] == current_number){
                    if(index == cur_index) strike++;
                    else ball++;
                }
            }
            cur_index--;
        }
```

全ての桁の比較が終わったら各カウンターを確認して結果を知らせる。 1 回もあってない場合にはOutと判定し、それ以外には各カウンターを表示する。もし、桁数とstrikeカウンターが一致すれば全ての乱数を当てたことになるため、褒め言葉を出力してループを終了する。

```
    if(strike == 0 && ball == 0) printf("Out!\n");
        else printf("%d strike %d ball\n",strike, ball);
        if(strike == digit){
            printf("Good jjob!");
            break;
        }
```

ループが終了したら MAXCOUNT を確認して 0 の場合、MAXCOUNT 回内に正解が出なかったことになるため、激励の言葉を出力して正解を公開する。

```
        if(MAXCOUNT == 0 && strike != digit){
         printf("You can do better than this!! \n");
         printf("The Answer was : ");
         for(index = 0; index<digit; index++){
             printf("%d",target[index]);
         }
         printf("\n");
    }
```

最後に calloc で用いたメモリー空間を解放して今回の結果を history ファイルに残してから終了する。結果の形式は、Hitandblow (Player's name) (Number of Attempt) (Year).(Month).(Day)である。

日付を示すために Time ヘッダーファイルの機能を用いて当日の日付を求めている。

```
    free(target);
    time_t now = time(NULL);
    struct tm *t = localtime(&now);
    if(fprintf(fp,"Hitandblow %s %d %d.%d.%d\n", name, 7-MAXCOUNT, t->tm_year
+ 1900,t->tm_mon+1,t->tm_mday) > 0) printf("History is saved successfully!");
    return;
```

Yacht.h

本ヘッダーファイルではYacht.cで具現した関数を定義している。特徴としては構造体をtypedefで宣言して、コンパイル時にYacht.cファイルで構造体が使えるようにしている。また、Hitandblow.cと同じくboundarycheck関数の外部定義を知らせている。

```
#ifndef __YACHT_H__
#define __YACHT_H__

typedef struct YachtBoard{
    int score;
    int categories[13];
    int dices[5];
    char name[20];
} YB;
extern bool boundarycheck(int from, int to, int *target);
void RollTheDice(YB *player, int* reroll);
void ShowTheDice(YB *player);
void SelectDice(YB *player, bool* mode);
void SetScore(YB *player);
void DisplayInfo(YB *player);
void YachtProcedure(YB *currentplayer, bool *mode);
void DisplayCategoryInfo(YB *player);
void initialize(YB *player, char* name);
void YachtDice(char *playername, FILE *fp);

#endif
```

Yacht.c

ヨットはプレイヤーのターン替えとサイコロの振りから点数の選択までの大きく２つ部分で分かれている。

Enum YachtCategories

関数を解説する前に本プログラムでよく使用される役（Category）を表すEnum列挙体を示す。役の意味は注釈に解説されていて、０から１１までのインデクスとして用いられる。

```
enum YachtCategories{
    Ones = 0,//１の目だけ足し合わせる。
    Twos, //２の目だけ足し合わせる。
    Threes,//３の目だけ足し合わせる。
    Fours,//４の目だけ足し合わせる。
    Fives,//５の目だけ足し合わせる。
    Sixes,//６の目だけ足し合わせる。
```

```
    Choice,//全ての目を足し合わせる。
    Four_of_a_Kind,//４つの目が同じである場合、全ての目を足し合わせる。
    Full_House,//３つの目が同じかつ前とは違う他の２つの目が同じである場合、全ての目を足し合わせる。
    Small_Straight,//４つの目が昇順並びになっている場合、１５点を与える。
    Large_Straight,//５つの目が昇順並びになっている場合、３０点を与える。
    Yacht//全ての目が同じである場合、５０点を与える。
};
```

YachtDice

まず、Yacht.cのメイン関数とも言えるYachtDice関数から見ると、オートモードであるかを示すauto mode変数と各プレイヤーを表す２つの構造体、その構造体を指す構造体ポインターがあることが分かる。以後、各構造体を名前で初期化する関数を呼び出して初期設定を行う。

```
    int turn;
    bool automode = false;
    YB p1;
    YB AI;
    YB *currentplayer;
    initialize(&p1,playername);
    initialize(&AI,"AI");
```

intialize関数ではプレイヤーのポインターと名前を用いて名前をコピーし、カテゴリーを－１に初期化して点数も０に初期化する作業を行っている。

```
void initialize(struct YachtBoard *player, char* named){
    strcpy(player->name, named);
    memset(player->categories, -1, sizeof(player->categories));
    player->score = 0;
}
```

以後、最後のターンまで奇数のターンにはプレイヤー１が、偶数のターンにはプレイヤー２がサイコロを振る作業を行う。現在サイコロを振っているプレイヤーを変えながらYachtProcedureで表されているサイコロを振る過程を実行している。一連の過程が終わるたびに両プレイヤーの点数を表示する。

```
       while(turn < 2*MAXTURN){
        if(turn % 2 == 0){
            currentplayer = &p1;
            automode = false;
        }
        else{
            currentplayer = &AI;
            automode = true;
```

```
        }
        printf("\nIt's %s's turn (%d/%d)\n", currentplayer->name, turn/2 + 1,
MAXTURN);
        YachtProcedure(currentplayer, &automode);
        printf("\nname : %s\t\tname : %s\nscore : %d\t\tscore : %d\n",p1.name,
AI.name,p1.score,AI.score);
        turn++;
    }
```

最後に、全てのターンが終わってからは各プレイヤーの点数を比較して勝負を決める。以後、各プレイヤーの点数を記録してファイルに残してから終了する。

```
if(p1.score > AI.score) printf("%s won !!", p1.name);
    else if(AI.score > p1.score) printf("AI won !!");
    else printf("Draw!!");

    // 記録をセーブする
    time_t now = time(NULL);
    struct tm *t = localtime(&now);

    if(fprintf(fp, "YachtDice %s %d %s %d %d.%d.%d\n", p1.name, p1.score,
AI.name, AI.score, t->tm_year + 1900, t->tm_mon + 1, t->tm_mday) > 0)
printf("History is saved successfully!");
```

YachtProcedure

続いて、サイコロを振る過程であるYachtProcedure関数を見ると、４つの過程からなることが分かる。各過程は、サイコロを振る、振った結果を表示する、リロールするサイコロを選ぶ、選んでいない役から点数を決めるという過程で構成されている。

```
void YachtProcedure(YB *currentplayer, bool *mode){
    Rollthedice(currentplayer, NULL);
    Showthedice(currentplayer);
    Selectdice(currentplayer, mode);
    Setscore(currentplayer);
}
```

RolltheDice

最初のRollthedice関数から見ると、パラメータとしてrerollしているかを受け入れていることが分かる。このrerollの可否がNULLである場合、リロールしていないことを示すため、全てのサイコロを振る。一方、rerollに値が存在する場合にはリロールするサイコロの目が記載されているため、その目だけを振る作業を行う。２つの場合ともdicesにセーブされる値は1~6の中で決められる。

```
void Rollthedice(YB *player, int* reroll){
    int diceindex, *ptr_index;
    srand((unsigned int)time(NULL));
    if(reroll == NULL){
        for(diceindex=0;diceindex<5;diceindex++){
            player->dices[diceindex] = (rand()%6 + 1);
        }
    }
    else{
        ptr_index = reroll;
        while(ptr_index < (reroll + 5) && *ptr_index != 0){
            player->dices[*ptr_index-1] = (rand()%6 + 1);
            ptr_index++;
        }
    }
}
```

Showthedice

パラメータで与えたプレイヤーのサイコロの目を表示する。

```
void Showthedice(YB *player){
    int diceindex;
    for(diceindex = 0;diceindex<5;diceindex++){
        printf("%d\t",diceindex+1);
    }
    printf("\n");
    for(diceindex = 0;diceindex<5;diceindex++){
        printf("%d\t",player->dices[diceindex]);
    }
    printf("¥n");
}
```

Selectdice

本関数の機能はリロールするサイコロを選択することである。２回の機会を与えて、リロールしたくない場合にはそのままループから抜けることができる。リロールする場合には最大５個のサイコロを選択してそのサイコロをリロールする作業を行う。リロールの際には上記したRolltheDice関数を呼び出して結果をShowthedice関数で表示する。

```
    for(i = 0; i<2; i++){
        printf("There are %d more chances \n",(2-i));
        memset(reroll,0,5*sizeof(int));
        do{
            printf("if you want to stop, enter 0. else if you want to reroll,
enter 1. ");
```

```
            scanf("%d",&again); getchar();
        }while(!boundarycheck(0,1,&again));
        if(again == false) break;
        printf("Please Enter the number of Dice which you want to reroll
without blank space : ");
        j = 0;
        while((input = getchar()) != '\n' && j < 5){
            value = input - '0';
            if(boundarycheck(1,5,&value)){
                reroll[j] = value;
                j++;
            }
            else {
                printf("Enter the right number!");
                printf("Please Enter the number of Dice which you want to
reroll without blank space : ");
            }
        }
        Rollthedice(player, reroll);
        printf("Result: \n");
        Showthedice(player);
    }
```

Setscore

本関数ではサイコロの目に基づいて点数の獲得が可能な役を探す作業を行い、点数に反映する作業を行う。ヨットゲームのルール上どの役にも点数を与えることができるが、条件に合わない点数付けは０点になるため、その条件を判断するための作業である。

まず、注釈に書いている通り各目が出た個数を把握するための配列numと、得点できる役を判断するための配列enableを生成する。この時、判断する役はCategories列挙体から見てFourofakindからYachtまでの役に限るため、５つの要素を持たせる。以後、numのインデクスをサイコロの目として使ってサイコロの目の個数を確認する。

```
    int num[6] = {0,};//各目の個数を把握するための配列。
    int enable[5] = {0,};//Fourofakind 以後のカテゴリーの条件を満足しているかを判
断するための配列。Fourofakind からヨットを 0 から 4 に対応させる。
    for(index = 0; index<5; index++){
        num[player->dices[index]-1] += 1;
    }
```

続いて、５つの役の条件を満たしているかを判断するための作業を行う。幸いに役の条件は目の個数で判断できるため、num配列を巡回しながら目の個数を把握することで目の個数が３個以上の場合と１連の目が昇順に並べている時の場合を考える。

Yachtの判定：５つの目が同じである場合

```
            if(num[index] == 5) {//同じ目が５つの場合
                enable[4] = 50;
                break;
            }
```

Fourofakindの判定：４つの目が同じである場合

→他の目を探して各目を覚えさせる。後で分解させるために桁で分離させる。

```
            else{
                for(additonal_index = 0; additonal_index < 6;
additonal_index++){
                    if(num[additonal_index] == 1)
{//同じ目が４つの場合、残りの１つの目を把握する。
                        enable[0] = 10*(index+1) +
(additonal_index+1);//後で２つの目で計算するために目を覚えさせる。
                        break;
                    }
                }
            }
            break;
        }
```

FullHouseの判定：３つの目が一緒で、２つの目がまた一緒の場合：

→Fourofakindと同じく各目を桁ごとに分離させる。

```
        else if(num[index] == 3){//フルハウス
            for(additonal_index = 0; additonal_index < 6; additonal_index++){
                if(num[additonal_index] == 2) {
                    enable[1] = 10*(index+1) +
(additonal_index+1);//３つの目が同じで他の２つの目も同じであれば、各目を覚えさせる
。
                    break;
                }
            }
        }
    }
```

SmallStraight,LargeStraightの判定：連続している４つ、５つの目が各々１個である場合

```
    for(index = 0; index<3; index++){
        bool flag = (num[index] == 1) && (num[index+1] == 1) && (num[index+2]
== 1) && (num[index+3] == 1);//{1234},{2345},{3456}
        if((flag == true) && index<2){
            enable[2] = 15;//連続した４つの目が存在すればスモールストレートである。
            if(flag == (num[index+4] == 1)){//{12345},{23456}
```

```
                enable[3] = 30;//スモールストレートで残りの１つの目も連続していれ
ば、ラージストレートである。
                break;
            }
            break;
        }
    }
```

最後に選択していない役を選択して各役の点数規則に合わせた得点を行う。(switch文は長文であるため別途の記載は行わない。)

```
do{
        DisplayInfo(player);
        printf("Select category that value is -1 : ");
        scanf("%d",&category);
    }while(player->categories[category] != -1
|| !boundarycheck(0,11,&category));
```

DisplayInfo

上記のカテゴリ選択の際に呼び出すDisplayInfo関数を見ると、現在プレイヤーの名前と点数、役の状態順に表示していることが分かる。また、役の情報は別途の関数で分離している。

```
void DisplayInfo(YB *player){
    printf("name : %s\n", player->name);
    printf("score : %d\n", player->score);
    DisplayCategoryInfo(player);
}
```

DisplayCategoryInfo

各役の表示と同時にボーナスを判定する。ボーナスはOnesからSixesまでの値の合計が６３以上の場合、３５点を付与されるもので、選択できない役であるため他の関数で管理している。

```
void DisplayCategoryInfo(YB *player){
    int index, sum = 0;
    if(player->categories[12] == -1){
        for(index = 0; index<6; index++){
            sum += player->categories[index];
        }
        if(sum >= 63){
            player->categories[12] = 35;
        }
    }
    printf("0. Ones : %d\n", player->categories[0]);
```

```
    printf("1. Twos : %d\n", player->categories[1]);
    printf("2. Threes : %d\n", player->categories[2]);
    printf("3. Fours : %d\n", player->categories[3]);
    printf("4. Fives : %d\n", player->categories[4]);
    printf("5. Sixes : %d\n", player->categories[5]);
    printf("Bonus!! : %d (if sum of score between Ones and Sixes is over 63,
take 35 points)\n", player->categories[12]);
    printf("6. Choices : %d\n", player->categories[6]);
    printf("7. Four of a kind : %d\n", player->categories[7]);
    printf("8. Full house : %d\n", player->categories[8]);
    printf("9. Small straight : %d\n", player->categories[9]);
    printf("10. Large straight : %d\n", player->categories[10]);
    printf("11. YACHT : %d\n", player->categories[11]);
}
```

作動例：

Hitandblow（3桁）：

Enter your name : asdf

1: Hit and blow　　2: Yacht Dice

Select Game : 1

Hit and blow Game is initialized

How many digits do you want? 3

3 digits number is created!You have 5 chances

Enter the number : 123

Out!

You have 4 chances

Enter the number : 456

0 strike 2 ball

You have 3 chances

Enter the number : 789

Out!

You have 2 chances

Enter the number : 065

0 strike 2 ball

You have 1 chances

Enter the number : 640

2 strike 0 ball

You can do better than this!!

The Answer was : 540

History is saved successfully!

Yacht（簡潔な結果のために MAXTURN を 2 回に制限している）：

Enter your name : asdf

1: Hit and blow 2: Yacht Dice

Select Game : 2

YachtDice Game is initialized

It's asdf's turn (1/2)

1 2 3 4 5

2 3 2 2 6

There are 2 more chances

if you want to stop, enter 0. else if you want to reroll, enter 1. 1

Please Enter the number of Dice which you want to reroll without blank space : 25

Result:

1 2 3 4 5

2 1 2 2 4

There are 1 more chances

if you want to stop, enter 0. else if you want to reroll, enter 1. 1

Please Enter the number of Dice which you want to reroll without blank space : 25

Result:

1 2 3 4 5

2 2 2 2 6

name : asdf

score : 0

0. Ones : -1

1. Twos : -1

2. Threes : -1

3. Fours : -1

4. Fives : -1

5. Sixes : -1

Bonus!! : -1 (if sum of score between Ones and Sixes is over 63, take 35 points)

6. Choices : -1

7. Four of a kind : -1

8. Full house : -1

9. Small straight : -1

10. Large straight : -1

11. YACHT : -1

Select category that value is -1 : 7

name : asdf name : AI

score : 14 score : 0

It's AI's turn (1/2)

1 2 3 4 5

6 6 2 2 5

There are 2 more chances

if you want to stop, enter 0. else if you want to reroll, enter 1. 1

Please Enter the number of Dice which you want to reroll without blank space : 345

Result:

1 2 3 4 5

6 6 5 3 2

There are 1 more chances

if you want to stop, enter 0. else if you want to reroll, enter 1. 1

Please Enter the number of Dice which you want to reroll without blank space : 345

Result:

1 2 3 4 5

6 6 4 4 3

name : AI

score : 0

0. Ones : -1

1. Twos : -1

2. Threes : -1

3. Fours : -1

4. Fives : -1

5. Sixes : -1

Bonus!! : -1 (if sum of score between Ones and Sixes is over 63, take 35 points)

6. Choices : -1

7. Four of a kind : -1

8. Full house : -1

9. Small straight : -1

10. Large straight : -1

11. YACHT : -1

Select category that value is -1 : 5

name : asdf name : AI

score : 14 score : 12

It's asdf's turn (2/2)

1 2 3 4 5

5 3 2 5 4

There are 2 more chances

if you want to stop, enter 0. else if you want to reroll, enter 1. 0

name : asdf

score : 14

0. Ones : -1

1. Twos : -1

2. Threes : -1

3. Fours : -1

4. Fives : -1

5. Sixes : -1

Bonus!! : -1 (if sum of score between Ones and Sixes is over 63, take 35 points)

6. Choices : -1

7. Four of a kind : 14

8. Full house : -1

9. Small straight : -1

10. Large straight : -1

11. YACHT : -1

Select category that value is -1 : 4

name : asdf name : AI

score : 24 score : 12

It's AI's turn (2/2)

1 2 3 4 5

5 3 2 1 4

There are 2 more chances

if you want to stop, enter 0. else if you want to reroll, enter 1. 0

name : AI

score : 12

0. Ones : -1

1. Twos : -1

2. Threes : -1

3. Fours : -1

4. Fives : -1

5. Sixes : 12

Bonus!! : -1 (if sum of score between Ones and Sixes is over 63, take 35 points)

6. Choices : -1

7. Four of a kind : -1

8. Full house : -1

9. Small straight : -1

10. Large straight : -1

11. YACHT : -1

Select category that value is -1 : 10

name : asdf name : AI

score : 24 score : 42

AI won !!History is saved successfully!

感想：

自分で２００桁を超えるプログラムを作成したことはありませんでしたので楽しいながら大変でした。勿論、２００桁とは言ってもグーグルのサービスコードは２０億桁を超えるコードで構成されているなど、他のプログラムに比べたら水ぼらしいコードに間違いないです。

しかし、２００～３００桁の中でも様々なバグが起こり、ポインターの間違えた使用とかインデクスの一貫性を持たなくて生じた不具合などの嫌な経験をたくさん味わうことができました。

今回のプログラムは他の人から見たら玩具でも使えないもので、実力の不足で修正できなかった不具合もまだ存在していますし（主にバウンダリーチェック）、当たり前ながら全ての不具合はプログラマーの自分の設計ミスでしたので言い訳もできませんでしたが、その分学んだこともあって楽しかったです。

機会があれば今度からはチームワークの時に使えるGitなどの有用なツールや技術を学んでみたいです。