プログラミング演習
～反復～

1　目的

C 言語における反復を理解し，ある処理を複数回繰り替えして実行する方法を会得する．例題としてバンディット問題 [1] を用いて実践し理解を深める．

前々回の基礎文法 (変数および命令文を順番に並べると順番に実行していく)，前回の条件分岐 (switch 命令と if 命令) と今回の反復によってほとんど全てのプログラムを記述する事が出来る．

2　製作対象　for 命令を用いたバンディット操作

これまで作成してきたプログラムは，一回バンディットを試行 (プレイ) すると終了していた．しかし，10 回連続で試行したい，自分の気が済むまで試行したい，という場合に反復命令を使うと実現できる．2.2 プログラムでは for 命令を用いて反復を実現する．

for 命令の説明の前に，バンディットに関して今までと異なる部分があるので先に述べる．15 行目が該当部分であり，対象としているバンディットの腕の数を取得している．これまでの bandit00.o では選択できる腕が 3 本であった．しかしどのバンディットも必ず 3 本の腕と決まっている訳ではない．そのため，異なるバンディットを試行する前に (例えば bandit04 など)，そのバンディットが何本の腕を持っているのかを知ることが必要となる．それが

num_arm = get_arm_num();

である．上記の関数は，いま対象にしているバンディットが何本の腕を持っているかを知るためのものであり，ここでは変数 num_arm に代入している．今回は，num_arm を 24 行目の printf で使い，腕の数が変わってもプレイヤーに腕の数を表示するように対応している．

さて，for 命令の基本的な使い方は

- for(初期化 ; 条件式 ; 変化式) 実行文

である．for によって 10 回なら 10 回実行文を繰り返すことができる．そのために，現在何回目の実行であるか，について記憶しておく必要がある．その現在の繰り返し回数を記憶する変数を初期化するのが，上記の”初期化”である．2.2 プログラムの 21 行目では，現在の回数を記憶する変数として loop 変数を用いており，初期化として loop を 0(回) にしていることが分かる．次に，現在繰り返している回数が，規定回数以上になっていないことを確認するのが”条件式”である．条件式は，条件分岐 [3] で述べた条件式のことである．この条件式が成り立っている間は for 命令による実行文の繰り返しが行われている．21 行目では，loop $<$ 10 となっており，現在の繰り返し回数が 10 回未満だと繰り返しが行われ，現在の繰り返し回数が 10 回を越えると for 命令による繰り返し

が終了することが分かる．次に，”変化式”は，”実行文”が一回行われるごとに一回処理される式である．21 行目のプログラムでは

- loop++ (loop = loop + 1 の略, loop に 1 を加える式)

となっており，一回実行文が実行されるごとに loop のカウントが+1 されていることが分かる．このような使い方により，希望回数だけ”実行文”を繰り返すことが実現できる．
ちなみに，実行したい”実行文”が複数行にわたる場合には，{ と } でくくってやる．

2.1 準備

ディレクトリ ”programming04”を作成する．今回の演習では，プログラムの作成や必要ファイルのダウンロードは，programming04 ディレクトリで行う．
はたおり虫より，以下のファイルをダウンロードする．

- bandit.h
- bandit00.o

余裕があれば，異なるバンディット (bandit01.o〜) を試してみること．

2.2 プログラム

バンディット関係の関数は，ガイダンス資料 [1] に説明があるので一読すること．

```
1  /*********************************************************
2            バンディットの for 命令を用いた反復
3  *********************************************************/
4
5  #include <stdio.h>  /* システムの用意した入出力        */
6  #include <stdlib.h> /* システムの用意した便利関数      */
7  #include "bandit.h" /* ユーザが用意したバンデット関連 */
8
9  int main(){
10   int select_arm, num_arm, loop;
11   double reward, total_reward;
12
13   init_bandit();          /* バンディットの初期化 */
14
15   num_arm = get_arm_num();/* バンディットの腕の数を取得 */
16   printf("このバンディットの腕の数は%d です\n",num_arm);
17
18
19   /* バンディット動作 10 回繰り返し*/
20   total_reward=0.0;
```

```
21      for(loop=0 ; loop < 10 ; loop++){
22
23        /* バンディットの腕の選択 */
24        printf("バンデットの腕の番号を選択してください [1-%d]:",num_arm);
25        scanf("%d", &select_arm);
26
27        / * バンディットの実行 */
28        reward = bandit(select_arm);
29        total_reward = total_reward + reward;
30      }
31
32      printf("総スコア:%lf\n",total_reward);
33
34      return 0;
35    }
```

2.3 コンパイル

上記のプログラムを作成し，ファイル名 gameplay-for.c として保存してコンパイルを行う．コンパイル後の名前を gameplay-for とする．

> gcc -o gameplay-for gameplay-for.c bandit00.o

2.4 動作実験

作成したプログラムを動作させる．腕を選択し，当たり/外れを観察する．

2.5 調べてみよう

for 命令の詳細について教科書等で調べてみよう．

2.6 やってみよう

2.6.1 繰り返し回数の変更

2.2 プログラムでは 10 回の繰り返し出会った．scanf を使用して，プレイヤーに試行回数を入力してもらい，その数だけ繰り返し実行できるようにしてみる．

2.6.2 総スコアと平均スコア

各試行で得られたバンディットからの報酬の和が総スコア (総獲得報酬) である．2.2 プログラムでは総スコアのみが出力されているが，一回当たり (一試行) の平均スコア (平均獲得報酬) を計算し，出力してみよう．

3　製作対象　whileを用いたバンディット操作

for 命令と同じく，while 命令を用いることで，ある処理の実行を規定回数繰り返すことが出来る．while 命令の基本的な使い方は

- while(条件式) 実行文

である．while によって，条件式が成り立っている間は実行文が繰り返し実行される．3.1 プログラムの 24 行目では select_arm に 0 が入力されるまで while の中身 ({ と } でくくられた領域，25 行目から 30 行目まで) の実行文が繰り返し実行される．

3.1　プログラム

```
1  /*********************************************************
2           バンディットの for 命令を用いた反復
3  *********************************************************/
4
5  #include <stdio.h>  /* システムの用意した入出力        */
6  #include <stdlib.h> /* システムの用意した便利関数      */
7  #include "bandit.h" /* ユーザが用意したバンデット関連 */
8
9  int main(){
10   int select_arm, num_arm, loop;
11   double reward, total_reward;
12
13   init_bandit();           /* バンディットの初期化 */
14
15   num_arm = get_arm_num();/* バンディットの腕の数を取得 */
16   printf("このバンディットの腕の数は%d です\n",num_arm);
17
18   /* バンディットの腕の選択 */
19   printf("バンデットの腕番号の選択 [1-%d] 終了したい場合には 0 入力:",num_arm);
20   scanf("%d", &select_arm);
21
22   /* バンディット動作 気が向くまで繰り返し*/
23   total_reward=0.0;
24   while(select_arm != 0){ /* select_arm に 0 が入っている場合には終了 */
25     reward = bandit(select_arm);
26     total_reward = total_reward + reward;
27
28     /* 次回のバンディット用に腕の選択 */
29     printf("バンデットの腕番号の選択 [1-%d] 終了したい場合には 0 入力:",num_arm);
30     scanf("%d", &select_arm);
31   }
```

```
32
33      printf("総スコア:%lf\n",total_reward);
34
35      return 0;
36    }
```

3.2 コンパイル

上記のプログラムを作成し，ファイル名 gameplay-while.c として保存してコンパイルを行う．コンパイル後の名前を gameplay-while とする．

> gcc -o gameplay-while gameplay-while.c bandit00.o

3.3 動作実験

作成したプログラムを動作させる．腕を選択し，当たり/外れを観察する．

3.4 調べてみよう

while 命令の詳細な使い方について教科書等で調べてみよう．

3.5 やってみよう

3.5.1 for と while と do-while の置き換え

while 命令の異なる形に do-while 命令がある．これも合わせて，反復の命令は相互に書き換えが可能である．下記の do-while の使い方をみて，3.1 プログラムを do-while を用いて書き換えてみよう．

- do{
- 実行文
- }while(条件式);

条件式が成り立っている間，実行文を繰り返す．while では，まず条件文が判定されて正しければ実行文の処理であったが，do-while では，まず実行文の処理が行われて条件式が判定される．

4 参考文献

参考文献

[1] プログラミング演習〜ガイダンス〜

[2] プログラミング演習〜基礎文法〜

[3] プログラミング演習〜条件分岐〜