プログラミング演習
～配列～

1　目的

C 言語における配列の使い方を理解する．通常の変数では一度に一つの値しか記憶できなかったものを，配列を使うことによって複数の値を扱うことを学習する．これをバンディット問題 [1] 等を例題にしながら実践し理解を深める．

2　製作対象 配列を利用したバンディットのスコア計算

前回 [2] まででバンディットを繰り返し試行 (プレイ) することが出きるようになった．これまでは総スコアは，その都度得られた報酬を加算していき求めていた．また平均スコアは総スコアを試行回数で除算し，求めていた．今回は，各試行で得られた報酬をすべて記憶しておき，必要に応じて総スコア・平均スコアを求める．

ここで，配列は下記のように定義される．

- double reward[10];

プログラム中では

- #define NUM_LOOP 10
- ...
- double reward[NUM_LOOP];

となっている．#define はコンパイル前の文字の置換を行っており，ここでは”NUM_LOOP”という文字列が”10”に置換されている．よって上記の二つの表現は同じである．

この場合，reward という変数は 10 個の数 (double 型, 実数) を記憶することができる．それぞれ，reward[0]～reward[9] まで”[”と”]”の間の数を変えることで別々の変数として扱うことができる．注意点としては，

- 定義するときには記憶しておきたい数，例:reward[10]
- 使用する時には，0 番めから始まる (reward[0]) ので，使える番号は記憶しておきたい数から 1 を引いた数 (reward[9])

である．

プログラム中で使用している部分は，29 行目と 30 行目，36 行目である．ここでは変数 loop を使って，reward[0], reward[1], reward[2], reward[3], reward[4], reward[5], reward[6], reward[7], reward[8], reward[9] に bandit からの報酬を記憶している．また，30 行目では各回でのスコアを表示しており，36 行目ではそれぞれ記憶した値を用いて総スコアを計算している．

2.1 準備

ディレクトリ ”programming05”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，programming05 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              配列を用いたバンディットスコア
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  #define NUM_LOOP 10 /* ループの回数                     */
9
10 int main(){
11   int select_arm, num_arm, loop;
12   double reward[NUM_LOOP];
13   double total_reward;
14
15   init_bandit();          /* バンディットの初期化 */
16
17   num_arm = get_arm_num();/* バンディットの腕の数を取得 */
18   printf("このバンディットの腕の数は%d です\n",num_arm);
19
20
21   /* バンディット動作 10 回繰り返し*/
22   for(loop=0 ; loop < NUM_LOOP ; loop++){
23
```

```
24        /* バンディットの腕の選択 */
25        printf("バンデットの腕の番号を選択してください [1-%d]:",num_arm);
26        scanf("%d", &select_arm);
27
28        /* バンディットの実行 */
29        reward[loop] = bandit(select_arm);
30        printf("%d 回目のスコア:%lf\n",loop+1, reward[loop]);
31      }
32
33      /* 総スコアの計算 */
34      total_reward=0.0;
35      for(loop=0 ; loop < NUM_LOOP ; loop++){
36        total_reward += reward[loop];
37      }
38
39      printf("総スコア:%lf\n",total_reward);
40
41      return 0;
42    }
```

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-array.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 gameplayary とする．

> gcc -o gameplayary gameplay-array.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，総スコアが正確に計算できていることを確認する．

2.5 調べてみよう

配列の使い方・注意方法について教科書を用いて調べてみよう．

2.6 やってみよう

2.6.1 獲得報酬一覧の表示

38 行目に，獲得した報酬の一覧 (1 回目の試行から 10 回目の試行まで) を表示するプログラムを追加してみよう．

一次元配列のイメージ

試行回数	1	2	3	4	5	6	7	8	9	10
記憶する場所	reward[0]	reward[1]	reward[2]	reward[3]	reward[4]	reward[5]	reward[6]	reward[7]	reward[8]	reward[9]

図 1: 一次元配列のイメージ

試行回数	1	2	3	4	5	6	7	8	9	10
選択した腕番号	1	3	2	3	1	2	3	3	2	3
得られた報酬	1	0	0	0	1	0	0	0	0	0

⇩

試行回数	1	2	3	4	5	6	7	8	9	10
選択した腕番号	reward[1][0]	reward[1][1]	reward[1][2]	reward[1][3]	reward[1][4]	reward[1][5]	reward[1][6]	reward[1][7]	reward[1][8]	reward[1][9]
得られた報酬	reward[0][0]	reward[0][1]	reward[0][2]	reward[0][3]	reward[0][4]	reward[0][5]	reward[0][6]	reward[0][7]	reward[0][8]	reward[0][9]

図 2: 多次元配列のイメージ

2.6.2 腕の選択を一括で

現在，腕の選択は 26 行目で行っており，bandit からの報酬を見ながら次の腕を選択することができる．これを 19 行目で事前に 10 回分の腕を選択し，結果を出すプログラムを作ってみよう．bandit からの報酬を見ながら腕を選択するのとどちらが難しい?

3 製作対象 多次元配列を使ったバンディット分析

bandit からの報酬を見ながら次の腕を考えるよりも，2.6.2 のように最初に腕の選択を一括で行って結果を出すことは難しい．これを一歩進めて，”どの腕を選択したときに，どのような報酬がもらえたのか?”の一覧を考える．この一覧によって，あるバンディットがどのような傾向を持つものか推測することが簡単になる．

ここで，記憶するべき値について考える．これまでは，試行回数毎に「その結果いくつ報酬がもらえたか?」のみを記憶していたために，各試行回数毎の報酬を記憶すればよかった (1 次元配列)．イメージとしては，1 次元配列が図 1 である．今回は，試行回数毎に,「どの腕を選んだか?」と「その結果いくつ報酬がもらえたか?」を記憶する．イメージとしては，図 2 である．

多次元を使用している部分は，12 行目 (定義)，27 行目および 30 行目 (代入・記憶)，31 行目および 37 行目 (使用) である．ここで 27 行目に注目する．多次元配列 reward[][] は double 型の変数である．一方，記憶すべき腕の番号は int 型の整数である．変数のサイズは $char > int > float > double$ であり，小さいサイズから大きいサイズへは変換が可能である．今回は，腕の番号を表す int 型の数値を無理やり double 型に変換して記憶している．変換の仕方は，27 行目のように，変数や数値の前に”(変換したい先の型)”と書けばよい．今回のように int 型変数 select_arm を double 型に変換する場合は，”(double)”を select_arm の前に書けばよい．

よりスマートな書き方は，構造体の演習時に行う．

3.1 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．

```
/*******************************************************
          多次元配列を用いたバンディットスコア
*******************************************************/

#include <stdio.h>  /* システムの用意した入出力        */
#include <stdlib.h> /* システムの用意した便利関数      */
#include "bandit.h" /* ユーザが用意したバンデット関連 */
#define NUM_LOOP 10 /* ループの回数                     */

int main(){
  int select_arm, num_arm, loop;
  double reward[2][NUM_LOOP];
  double total_reward;

  init_bandit();           /* バンディットの初期化 */

  num_arm = get_arm_num();/* バンディットの腕の数を取得 */
  printf("このバンディットの腕の数は%d です\n",num_arm);


  /* バンディット動作 10 回繰り返し*/
  for(loop=0 ; loop < NUM_LOOP ; loop++){

    /* バンディットの腕の選択 */
    printf("バンデットの腕の番号を選択してください [1-%d]:",num_arm);
    scanf("%d", &select_arm);
    reward[1][loop]=(double)select_arm;

    /* バンディットの実行 */
    reward[0][loop] = bandit(select_arm);
    printf("%d 回目の試行:",loop+1);
    printf("選択した腕番号:%lf:スコア:%lf\n",reward[1][loop], reward[0][loop]);
  }

  /* 総スコアの計算 */
  total_reward=0.0;
  for(loop=0 ; loop < NUM_LOOP ; loop++){
    total_reward += reward[0][loop];
  }

  printf("総スコア:%lf\n",total_reward);

  return 0;
```

```
44    }
```

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-multi-array.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-multiarray とする．

> gcc -o gameplay-multiarray gameplay-multi-array.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，選択した腕と報酬の一覧が出力されていることを確認する．

3.4 調べてみよう

多次元配列の使い方・注意方法について教科書を用いて調べてみよう．

3.5 やってみよう

3.5.1 腕とスコアの関係を見る

もう一つの多次元配列のイメージを図 3 に示す．この様に，二次元配列であれば平面のように配列を考え，各軸が対象にしたい要因 (この場合，要因 1:スコアに関するもの，要因 2:腕に関するもの) となる．この図のように，腕とスコアに関する多次元配列を実現しなさい．各腕に関する総スコアとは，その腕を選んだときに得た報酬の総和であり，平均スコアとは，総スコアをその腕を選んだ回数で除算し，その腕を選んだ場合の 1 回あたりの報酬である．

また，作成したプログラムを実行し，正しくスコアが表示されていることを確認し，多次元配列が正しく利用できていることを確認してみよう．

スコア / 腕番号	選択した回数	総スコア	平均スコア
1	info[0][0]	info[0][1]	info[0][2]
2	info[1][0]	info[1][1]	info[1][2]
3	info[2][0]	info[2][1]	info[2][2]

図 3: 腕とスコアの関係性

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〜反復〜