プログラミング　第 2 回　レポート

202212022 田島瑞起

2023/06/10

1　はじめに

今回の課題では,0 から n-1 までの範囲の乱数取得関数の作成 (設問 1),m から n までの範囲の乱数取得関数の作成 (設問 2), コマンドライン引数を用いた m から n までの乱数取得関数の作成 (設問 3) を満たしたレポートを latex を用いて作成する。

2　0 から n-1 までの範囲の乱数取得関数の作成 (設問 1)

2.1　課題内容の説明

0 から n-1 までの範囲の値をランダムに取得できるような関数 myrandN 関数を作成し、それを main 関数内でテストする。

2.2　課題への取り組み方針

通常の rand では値が 0 から RAND_MAX まで変化しうるので、myrandN はその値を 0 から n-1 まで圧縮できるような関数である必要がある。そこで、$0 <= rand() < RAND\_MAX$ の関係を用いて、この不等式の右辺が n になるよう式変形する。

2.3　解答結果

図 1　s2212022-1.c

```c
int myrandN(int n){
    return (int)( n * (rand()+1.0) / (RAND_MAX+1.0));
}

int main(int ac, char *av[]){
    int i;
    srand(time(NULL));

    for(i=0;i<20;i++){
```

```
10              printf("%d\n",myrandN(6));
11          }
```

図 1 にて、前述した圧縮する式は $(n-1-0+1)*(rand()+1.0)/(RAND\_MAX+1.0)$ と表現することができる。これによって、rand() の値を 0 から n-1 に圧縮することができる。後は結果を確かめるべく、main 関数の中で 20 回ほど myrand(6) を呼び出し、条件を満たしているか確認できるようになっている。また実行の際に乱数が変更されるように、srand(time(NULL)) も main 関数内に記述した。

2.4 確認

確認するべきポイントとしては、乱数が 0 から 5 に分布していることと、呼び出し時間によって乱数が異なることである。よって二回呼び出した結果のそれぞれ先頭 10 項を確認すると、

図 2 s2212022-1.out

```
2       3       4       1       5       1       1       1       4       1
0       5       3       3       3       5       0       1       1       4
```

上記二つの条件を満たしていることがわかる。

3 m から n までの範囲の乱数取得関数の作成 (設問 2)

3.1 課題内容の説明

設問 1 で作成したプログラムに変更を加えて、n から m までの範囲の乱数取得可能な関数を作成する。

3.2 課題への取り組み方針

設問 1 で作成した myarrayN() 関数を一般化し,n から m までの範囲の乱数を取得できるように、関数内部の $(n-1-0+1)*(rand()+1.0)/(RAND\_MAX+1.0)$ について変更を加える。

3.3 解答結果

図 3 s2212022-2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int myrandM2N(int m ,int n){
    return (int)(m + (n + 1 - m)* (rand()+1.0) / (RAND_MAX+1.0));
}

int main(int ac, char *av[]){
    int i;
```

```
11          srand(time(NULL));
12
13          for(i=0;i<25;i++){
14              printf("%d\n",myrandM2N(1,10));
15          }
```

図 3 にて、変更を加えた後の式は $(m + (n - m + 1) * (rand() + 1.0)/(RAND_MAX + 1.0))$ となっている。これは不等式である $0 <= rand() < RAND\_MAX$ を同値変形することによって得られる。文字を使わずに表すなら (最小値) <= 最小値 + (最大値 − 最小値 + 1) ∗ $(rand() + 1.0)/((RAND_MAX + 1.0)$ <= (最大値)) と表せる。これによって n から m までの範囲の値を乱数取得することが可能になる。

3.4 確認

設問 1 と同様に範囲と実行時間によって乱数が変化することが確認できれば良いが、今回の場合変更を加えたことによって確認する必要があるのは乱数の範囲についてだけである。

図 4 s2212022-2.out

```
1     1       10      6       9       3       1       10      3       7       9
2     4       1       7       6       9       9       4       1       2       6
3     3       6       3       2       3
```

1 から 10 まで乱数が取得できているので確認すべき条件を満たしている。

4 コマンドライン引数を用いた m から n までの範囲の乱数取得関数の作成 (設問 3)

4.1 課題内容の説明

設問 2 で作成したプログラムに変更を加えて, コマンドライン引数を用いた n から m までの範囲の乱数取得可能な関数を作成する。

4.2 課題への取り組み方針

設問 2 で作成したプログラムでは値の範囲をあらかじめ指定してから実行する形になっていたが、これをコマンドライン引数で指定できるように変更を加える。入力した値が char*型として格納されるため、数値に変換して値を取り扱えるようにする。また、入力数が 3 を満たしていて、m,n,c の制約を満たしている時のみ正常にプログラムが実行されるように、制御構文を用いて分岐させる。

4.3 解答結果

図 5 s2212022-3.c

```
1     #include <stdio.h>
2     #include <stdlib.h>
```

```
3     #include <time.h>
4
5     int myrandM2N(int m ,int n){
6         return (int)(m + (n + 1 - m)* (rand()+1.0) / (RAND_MAX+1.0));
7     }
8
9     int main(int ac, char *av[]){
10        int i;
11        int min;
12        int max;
13        int rep;
14        min = atoi(av[1]);
15        max = atoi(av[2]);
16        rep = atoi(av[3]);
17        srand(time(NULL));
18
19        if(ac != 4){
20            printf("Usage:a.out␣m␣n␣c\n␣0␣<␣m␣<␣n␣<␣100␣\n␣c␣is␣the␣number␣of␣repetition");
21            exit(1);
22        }
23
24        if(min > max|| max < 1 || max > 99 || min < 1 || min >99){
25            printf("Usage:a.out␣m␣n␣c\n␣0␣<␣m␣<␣n␣<␣100␣\n␣");
26            exit(1);
27        }
28
29        for(i=0;i<rep;i++){
30            printf("%d\n",myrandM2N(min,max));
31        }
```

図 5 にて、av[] に格納される文字列を数値型に変換するため atoi() 関数を使用した。また、入力数が 3 の制約を満たした時のみ実行されるよう、if(ac != 4) で分岐を実装した。(コマンドライン引数は入力数 3 +./s2212022-3.out の計 4 つであることに注意) 加えて、入力値の制約を満たした時のみ実行されるよう、if(min > max|| max < 1 || max > 99 || min < 1 || min >99) でも分岐を実装した。

4.4 確認

入力数が 3 の時のみプログラムが実行され、mnc が制約を満たしている時のみプログラムが実行されている必要がある。

図 6　s2212022-3.out 3 2 5

```
1     Usage:a.out m n c
2     0 < m < n < 100
```

図 7　s2212022-3.out 1 2 3 4

```
1     Usage:a.out m n c
2     0 < m < n < 100
3     c is the number of repetition
```

図 8　s2212022-3.out 2 5 10

```
1    4    3    2    5    5    2    3    3    2    5
```

図 6 では m>n の制約違反、図 7 では入力数の違反を犯している。図 8 にて、入力数 3 で m,n,c が制約を満たしている時のみプログラムが正常動作していることが確認された。

5　感想

今課題にて、設問 1 ではどうすれば乱数の範囲を圧縮できるかという部分が個人的な難所だった。0<=rand()<RAND_MAX の関係性から導けるということに気が付いてからはスムーズに実装することができた。設問 2 では設問 1 を一般化すればよいだけであったから特に詰まることなく実装できた。設問 3 ではコマンドライン引数の扱いについて苦労した。例えば./s2212022-3.out 1 2 3 を実行した際に、av[0] = ”./s2212022-3.out” が格納されるという関係性を知らなかったので取り扱いがうまくいかなかった。av[] の内容を for 文で出力してから関係性に気が付くことができ、わからない際とりあえず出力してみることの重要性を再認識した。latex でのレポート作成については前回ソースコードを適当に verbatim で張り付けたため参照ができなかったが、文献 [1] を参考にし、lstisting パッケージを用いて整形したソースコードを埋め込むことができるようになった。

参考文献

[1] tab0. Latex にソースコードを貼る方法, 2018. https://qiita.com/tab0/items/2619d5927492edbb5b03l.