情報処理特論I 第14回授業資料

＜注釈＞

* 以下の演習課題をwordファイルで作成すること（プログラムコードは別途添付でも構わない）
* プログラミングの出力結果はプリントスクリーン（スクリーンショット）でコピーしてwordファイル内に添付
* 「jfsv/5J書込可/matsuzaki/第14回（演習4）」フォルダ内に”4JXX\_name”（出席番号\_名前）フォルダを作成し、演習課題を提出すること（**最終締め切り2019/09/05 23:59**）

＜演習課題4＞

1. “Write a method to randomly generate a set of 𝑚 integers from an array of size 𝑛. Each element must have equal probability of being chosen.”（第12回実施）について、以下の演習を行ってください。
2. 授業時に考えた2つのアルゴリズム（2. & 4.）を実装し、それぞれが正しく出力されることを確認してください。

　授業時に考えた２つのアルゴリズムのソースコードを図１に，そのプログラムの実行結果を図２に示す。

|  |
| --- |
| #include <stdio.h> #include <math.h> #include <stdlib.h> #include <string.h>  void pickMRand (int\* original, int\* subset, int n, int m); int Rand (int lower, int higher); void pickMRand\_2 (int\* original, int\* subset, int n, int m);  int size = 10;  int main(){  int org[20] = {10,13,22,23,4,51,6,5,81,9,10,1,4,11,14,15,6,37,18,20};  int sub1[size];  int sub2[size];  pickMRand(org,sub1,20,size);  pickMRand\_2(org,sub2,20,size);  printf("課題 2 | 4\n");  for(int i=0;i<size;i++){  printf("%2d: %2d || %2d\n",i,sub1[i],sub2[i]);  }   }  void pickMRand (int\* original, int\* subset, int n, int m) {  for (int i = 0; i < m; i++) {  int index = (int)(rand() % n);  subset[i] = original[index];  } }  int Rand (int lower, int higher) {  return lower + (int)(rand() % (higher-lower +1)); }  void pickMRand\_2 (int\* original, int\* subset, int n, int m) {  int array[n];  memcpy(array, original, sizeof(int)\*n);  for (int i=0; i<m; i++) {  int index = Rand (i, n-1);  subset[i] = array[index];  array[index] = array[i];  } } |

図１　課題１の二つのアルゴリズムのソースコード

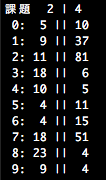


図2　課題１の二つのアルゴリズムの実行結果

1. 前問1-Aのそれぞれの出力結果から、𝑚配列内に生成される値がどのような傾向（重複頻度や配列構成などについて）があるか考察してください。

　図２の左のアルゴリズム（2.）は同じ数字（一つしかないもの）を複数回選んでいるのに対し，右のアルゴリズム（4.）は同じものは抽出していない。（４はデータ中に二つあるため）

1. 授業時の4.のアルゴリズムにおいて、関数内の一時配列array[]（第12回解答例参照）の中身がどのように変化するかを標準出力などに出力して確認してください。

キーボード, 電子機器, おろし金 が含まれている画像

自動的に生成された説明

図３　アルゴリズム4.の一時配列arrayの内部の変化

1. “Write a method to count the number of 2s between 0 and 𝑛.”（第13回実施）について、以下の演習を行ってください。
2. 授業で取り扱った再帰と繰り返しによる2種類のアルゴリズム（3. & 4.）をそれぞれ実装してください。

　実装したアルゴリズム（３．&４.）のソースコードを図４に示す。

|  |
| --- |
| int count2sR (int n) {  if (n==0) return 0;  int power = 1;  while (10 \* power < n) power \*= 10;  int first = n / power, remainder = n % power;  int nTwosFirst = 0;  if (first > 2) nTwosFirst += power;  else if (first == 2) nTwosFirst += remainder + 1;  int nTwosOther = first \* count2sR (power - 1) + count2sR (remainder);  return nTwosFirst + nTwosOther; }  int count2sI (int n) {  int countof2s = 0, digit = 0;  int num = n, seendigits = 0, position = 0, pow10\_pos = 1;  while (num > 0) {  digit = num % 10;  int pow10\_posMinus1 = pow10\_pos / 10;  countof2s += digit \* position \* pow10\_posMinus1;  if (digit == 2) countof2s += seendigits + 1;  else if (digit > 2) countof2s += pow10\_pos;  seendigits = seendigits + pow10\_pos \* digit;  pow10\_pos \*= 10;  position++;  num = num / 10;  }  return countof2s; } |

図４　授業で取り扱った再帰と繰り返しによる2種類のアルゴリズム（3. & 4.）のソースコード

1. “2s”の部分を任意の1桁の自然数（）で求められるようにアルゴリズムを変更してください。

|  |
| --- |
| #include <stdio.h>  int countR (int n, int a) {  if (n==0) return 0;  int power = 1;  while (10 \* power < n) power \*= 10;  int first = n / power, remainder = n % power;  int nTwosFirst = 0;  if (first > a) nTwosFirst += power;  else if (first == a) nTwosFirst += remainder + 1;  int nTwosOther = first \* countR (power - 1, a) + countR (remainder, a);  return nTwosFirst + nTwosOther; }  int countI (int n, int a) {  int countof2s = 0, digit = 0;  int num = n, seendigits = 0, position = 0, pow10\_pos = 1;  while (num > 0) {  digit = num % 10;  int pow10\_posMinus1 = pow10\_pos / 10;  countof2s += digit \* position \* pow10\_posMinus1;  if (digit == a) countof2s += seendigits + 1;  else if (digit > a) countof2s += pow10\_pos;  seendigits = seendigits + pow10\_pos \* digit;  pow10\_pos \*= 10;  position++;  num = num / 10;  }  return countof2s; }  int main(){  printf("countR = %d\n",countR(99, 2));  printf("countL = %d\n",countI(99, 2));  return 0; } |
|  |

図５　任意の1桁の自然数（）で求められるようにしたアルゴリズムのソースコード

1. “0~𝑛”の正の整数（自然数）において、利用される数字の数（例：345→3個、78→2個、0~10→12個）を 求めるアルゴリズムを作成してください。

|  |
| --- |
| #include <stdio.h>  int count\_num(int s, int e){  int buf, count=0;  for(int i=s; i<=e; i++){  buf = i;  do{  buf = buf / 10;  count++;  }while(buf > 0);  }  return count; }  int main(){  printf("345 =%d\n", count\_num(3, 5));  printf("78 =%d\n", count\_num(7, 8));  printf("0~10=%d\n", count\_num(0, 10));  return 0; } |

図６　課題2-Cのプログラムのソースコード

1. “0~𝑛”の任意の数字（）を含む数値を、すべて加算あるいは乗算するルゴリズムを作成してください。

|  |
| --- |
| #include <stdio.h>  int mul\_all(int num){  int i, ans=1;  int buf1 = num;  int buf2;  do{  buf2 = buf1 / 10;  ans \*= (buf1 - buf2\*10);  buf1 = buf2;  }while(buf1 >= 10);  ans \*= buf1;  return ans; }  int add\_all(int num){  int i, ans=0;  int buf1 = num;  int buf2;  do{  buf2 = buf1 / 10;  ans += (buf1 - buf2\*10);  buf1 = buf2;  }while(buf1 >= 10);  ans += buf1;  return ans; }  int main(){  printf("mul test %d\n", mul\_all(114514));  printf("add test %d\n", add\_all(114514));  return 0; } |

図７　課題2-Dのプログラムのソースコード