計算機基礎実習 I ・第 14 回

2017 年 7 月 21 日

今回の内容

14.1 大量のデータの格納

第 5 回の授業では、変数を使うことによって実数値や整数値を記憶しておき、その値を後で参照することができることを勉強しました。今回は記憶しなければならないデータが非常にたくさんある場合に便利な**配列**と呼ばれる仕組みを紹介します。

次のようなプログラムを作ることを考えます。

> 9 桁までの正の整数をキーボードから入力すると、0 〜 9 の各数字が何回現れているかを数えて次の実行例のように表示する

```
9桁までの自然数: 709404431
0 : 2個
1 : 1個
2 : 0個
3 : 1個
4 : 3個
5 : 0個
6 : 0個
7 : 1個
8 : 0個
9 : 1個
```

入力された正の整数が変数 n に格納されているとすると、その各桁に現れている数字は、n が正の間、次の手順を繰り返すことで取り出すことができます。

(1) `n % 10` を計算して 1 の位の数字を取り出す。

(2) `n /= 10;` を実行して、1 の位を取り除く。

入力された正の整数の桁に、0 〜 9 の各数字が何回現れているかを知るためには、出現回数を記憶する変数をそれぞれ用意しておき、上の手順 (1) で数字を取り出した際に、対応する変数の値を 1 つずつ増やしていかなければなりません。このため、出現回数を記憶する変数は 10 個必要になります。つぎのプログラム digittest.c は以上のような考え方で書いたものです。ただし、行頭の整数は説明のために付けた行番号で、このプログラムの一部ではありません。

digittest.c

```
1 #include <turtle.h>
2
3 main()
```

```
 4 {
 5     int n, d, c0, c1, c2, c3, c4, c5, c6, c7, c8, c9;
 6
 7     tPrintf("9桁までの正の整数: ");
 8     tScanf("%d", &n);
 9
10     c0 = c1 = c2 = c3 = c4 = c5 = c6 = c7 = c8 = c9 = 0;
11
12     while (n > 0) {
13         d = n % 10;
14
15         if (d == 0)
16             c0++;
17         else if (d == 1)
18             c1++;
19         else if (d == 2)
20             c2++;
21         else if (d == 3)
22             c3++;
23         else if (d == 4)
24             c4++;
25         else if (d == 5)
26             c5++;
27         else if (d == 6)
28             c6++;
29         else if (d == 7)
30             c7++;
31         else if (d == 8)
32             c8++;
33         else
34             c9++;
35
36         n /= 10;
37     }
38
39     tPrintf("0 : %d個\n", c0);
40     tPrintf("1 : %d個\n", c1);
41     tPrintf("2 : %d個\n", c2);
42     tPrintf("3 : %d個\n", c3);
43     tPrintf("4 : %d個\n", c4);
44     tPrintf("5 : %d個\n", c5);
45     tPrintf("6 : %d個\n", c6);
46     tPrintf("7 : %d個\n", c7);
47     tPrintf("8 : %d個\n", c8);
48     tPrintf("9 : %d個\n", c9);
49 }
```

このプログラムでは、c0、c1、···、c9 の 10 個の変数 (整数値) を用意しておき、12 行目から始まる while 文による繰り返しの処理の中で、取り出した数字 d によって場合分けを行い、その数字に対応する変数の値を 1 ずつ増やしています。この部分をもし、

```
do {
    d = n % 10;
    cd++;
    n /= 10;
}
```

のようにできれば簡単なのですが、cd++ と書いても「cd という名前の変数の値を 1 増やしなさい」

という意味でしかありませんからこのプログラムはうまく働きません。もとのプログラムにある c0、c1、···、c9 というのは単なる変数の名前であって、C プログラムとしてはそれらの間には何の関係もないのです。また cd も単なる変数の名前であって、c0、c1、···、c9 とは全く関係がありません。もしプログラムをこのように書き換えたとしたら cd という変数を宣言しておかないとコンパイルさえできないはずです。

メモ

配列　digittest.c では 10 通りの数字が現れた回数をそれぞれ数えていくために、c0、c1、···、c9 の 10 個の変数を用意していましたが、もし 1000 通りや 10000 通りなど、もっとたくさんのものを扱うとなれば、この方法で実現することはほとんど不可能になります。

そこで、C 言語を含めて、実用的なプログラミング言語には、digittest.c の c0、c1、···、c9 のように、関連したいくつかの変数を (関連づけたまま) まとめて扱えるような仕組みが用意されています。その仕組みは**配列**と呼ばれます。配列はいくつかの変数を並べてひとまとめにしたものです。配列を使うと digittest.c はつぎのプログラムのように書き換えることができます。

digitarray.c

```
 1 #include <turtle.h>
 2
 3 main ()
 4 {
 5     int n, d;
 6     int c[10];
 7
 8     tPrintf("9桁までの正の整数: ");
 9     tScanf("%d", &n);
10
11     for (d = 0; d < 10; d++)
12         c[d] = 0;
13
14     while (n > 0) {
15         d = n%10;
16         c[d]++;
17         n /= 10;
18     }
19
20     for (d = 0; d < 10; d++)
21         tPrintf("%d : %d個\n", d, c[d]);
22 }
```

配列の宣言　変数がそうであるように、配列はプログラムの中で使う前にまず宣言しておかなければなりません。この digitarray.c というプログラムでは、6 行目の

```
    int c[10];
```

という行でその宣言を行っています。c がその配列の名前で、[] の中に書かれた 10 がその配列の大きさ (いくつのデータを格納することができるか) です。ここでは、int を使って配列を宣言していますので、この配列 c には 10 個の整数値を記憶しておくことができます。実数値を記憶する配列を使いたければ、int の代わりに double を使って

```
double data[100];
```

のように宣言します。また、digitarray.c では、普通の変数である n や i とは別に配列 c の宣言を行っていますが、

```
int n, d, c[10];
```

のように、これらを 1 つにまとめて宣言することもできます。

メモ

配列の名前　配列の名前は、変数の場合と同じように選ぶことができます[1]。ただし、同じ名前の変数と配列を同時に宣言して使うことはできません。

メモ

配列の要素　配列はいくつかの変数を並べたもののように振舞います。大きさが 10 の配列であれば 10 個分の変数が用意されていることに相当します。その 1 つ 1 つは、配列名の後に [] で囲んだ**添字**を指定して、c[0] や c[3]、c[9] のようにして特定します。これらを、この配列 c の**要素**と呼びます。配列の各要素は通常の変数のようにそれぞれ代入[2]や参照をすることが可能です。

メモ

[1]つまり、英字で始まり、そのあとに英字か数字あるいは _ (下線記号) がいくつか続くものを使います。ただし、if、else、while、for、do、break、continue、int、double などの C プログラムの**予約語** (特別な意味を持っている単語) は変数名や配列名として使うことができません。

[2]tScanf("%d", &c[2]); のように、キーボードから入力した数値を配列の要素に格納することもできます

配列の添字の範囲　配列の添字は 0 から始まります。つまり、大きさが 10 の配列であれば、添字が 0 から 9 までの要素を使うことができます。たとえプログラムの中で `c[0]` を使わなかったとしても `c[10]` が使えるわけではありませんので注意が必要です。添字として 10 を使いたい場合は、配列の大きさを (少なくとも) 11 として宣言しておかなければなりません。また、定数に限らず、整数値を表すどんな数式でも配列の添字として `[]` の中に書くことができます。たとえば、`digitarray.c` の 15 行目と 16 行目は、まとめて、

```
c[n%10]++;
```

と書くこともできます。ただし、どんな場合でもその数式の値が許される添字の範囲を越えないように注意しなければなりません。たとえば、大きさが 10 として宣言された配列 `c` に対して `c[-1]` や `c[10]` を使ってしまった場合、そのプログラムの動作は全く予期することができません。配列の添字が許される範囲を越えないように常に注意してプログラムを書きましょう。

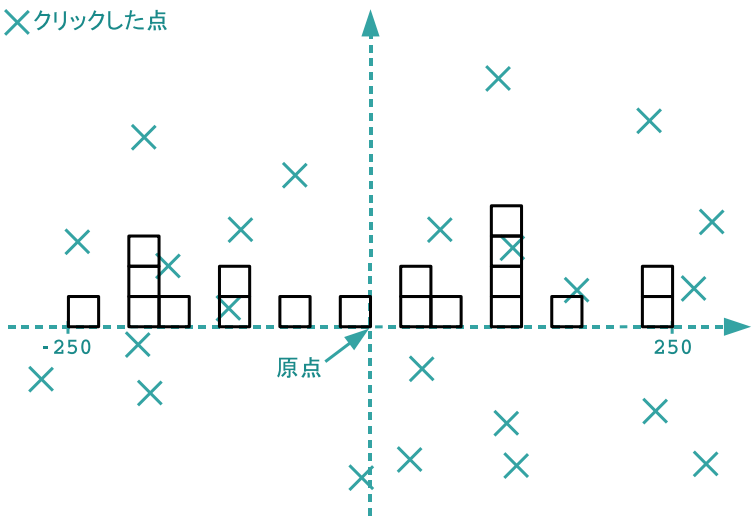
メモ

14.2　演習問題

1.　$-250 \leqq x < 250$ の範囲の x 座標を、幅 25 の 20 個の区間に分け、各区間が何回クリックされたかを、次の図のように (正方形を積み上げた) 棒グラフで表すプログラム `histogram.c` を作成し、コンパイル、実行して、正しく動作することを確認しなさい。20 個の区間は、

$$-250 + 25k \leqq x < -250 + 25(k+1) \qquad (k = 0,\ 1,\ 2,\ \cdots,\ 19)$$

となります。このプログラムは、$-250 \leqq x < 250$ の範囲がクリックされる度に、1 辺が 25 の正方形を一つ画面に追加します。$-250 \leqq x < 250$ の範囲以外がクリックされた場合は、そのクリックは無視するようにしてください。すばやく描画できるように `tSetSpeed` 関数を使って、カメの移動速度を速くしておいた方がよいでしょう。



ヒント： 大きさが 20 の配列を用意しておき、20 個の区間のそれぞれが何回クリックされたのか (整数値) を覚えておくようにしましょう。いずれかの区間がクリックされる度に、その区間に対応する配列の要素を調べて、正方形を描く際の (各頂点の) y 座標を求めます。クリックされた点の座標が (x, y) のとき、クリックされた区間に対応する配列要素の添字は $(x + 250)$ を 25 で割った商 (小数点以下切り捨て) とすることができます。

14.3 今回の実習内容

1. プリントをもう一度読み返しましょう。例題 `digitarray.c` のソースプログラムを作成し、コンパイル、実行してみましょう。プログラムが完成したら「課題の提出と確認」の Web ページから提出してください。
2. 演習問題に取り組みましょう。プログラムが完成したら、「課題の提出と確認」の Web ページからの提出を忘れないでください。
3. クイズに答えてください。「課題の提出と確認」の Web ページで「第 14 回 クイズ」を選択し「送信」のボタンをクリックしてクイズに答えてください。
4. 引き続き自由に魅力的なプログラムの作成を行って下さい。

    - ソースファイルの名前は `final.c` としてください。
    - マウスやキーボードからの入力を使っても ok です。ただし、どのように操作すればよいか、そのプログラムを実行しただけで分かるように工夫してください。
    - ソースファイルには他人が書いたプログラムが含まれていてはいけません。
    - この科目で紹介したことのない関数は (基本的には) 使用できません。もし、どうしても使用したい場合は早めに担当教員に相談してください。
    - 提出の締め切りは 7 月 31 日 (月) 18:30 です。第 15 回 (7 月 28 日) の授業開始以降、「課題の提出と確認」の Web ページから提出できるようになります。
    - みなさんが作ったプログラム (ソースプログラムとオブジェクトプログラム) は、8 月 2 日 (水) 12:00 から 8 月 4 日 (金) 18:30 の間、匿名で公開 (学内のみ) します。

- 公開期間中、1 人 10 件程度、他の受講者が作成したプログラムを評価して頂きます。この評価も課題の一部です。セルフラーニング室等を利用して、「課題の提出と確認」の Web ページから、8 月 4 日 (金) 18:30 までに忘れず評価を行ってください。
- 評価は、次のような観点で行い、自分が評価を担当していないプログラムも含めて、ごく平均的と思われるプログラムの点数が、満点の半分になるように採点してください。
  (a) プログラムを実行した時の動作や、その描く図形などは魅力的か (10 点満点)
  (b) ソースプログラムにこの科目で勉強したことが反映されているか (5 点満点)

次のフォルダに昨年度の作品が置いてありますので参考にしてください。

R:\a89023\計算機基礎実習I\2016年度自由課題

みなさんが作成した作品も、来年度以降の受講者に対して同様に紹介させていただきます。

計算機基礎実習 I ・第 14 回・終わり

14.4 付録：2次元の配列

3ページの例題 digitarray.c で使った配列は、整数値を記憶する要素を1次元的に10個並べたものでしたが、要素を2次元的に並べた配列を使うこともできます。たとえば、

```
int a[30][50];
```

のように配列 a を宣言すると、整数値を記憶する要素が2次元的に 30×50 個並んだ配列ができます。この配列の要素に対して代入や参照を行うには、a[13][37] のように2つの添字を指定します。最初の添字の範囲は0から29まで、2番目の添字の範囲は0から49までとなります。

次のプログラム tiles.c は $\left\{ (x, y) \,\middle|\, -250 \leqq x < 250 \text{ かつ } -250 \leqq y < 250 \right\}$ の領域を、間隔100の格子で 5×5 個の正方形の区画に分割し、各区画がクリックされる度に、クリックされた区画の色を、白 → 赤 → 青 → 白 → 赤 → 青 ··· の順に循環させていくものです。

tiles.c

```
 1 #include <turtle.h>
 2
 3 main ()
 4 {
 5     int i, j, x, y;
 6     int color[5][5];
 7
 8     tSetSpeed(0.0);     /* カメのスピードを最大にする */
 9     tHide();            /* カメを表示しないようにする */
10
11     /* 配列 color の各要素を 0 で初期化して、各区画の境界を描く */
12     for (i = 0; i < 5; i++) {
13         for (j = 0; j < 5; j++) {
14             color[i][j] = 0;
15             /* この区画(正方形)の左下の頂点の座標を求める */
16             x = i * 100 - 250;
17             y = j * 100 - 250;
18             /* この区画の境界を黒色で描く */
19             tPenUp();
20             tMoveTo(x, y);
21             tPenDown();
22             tMoveTo(x+100, y);
23             tMoveTo(x+100, y+100);
24             tMoveTo(x, y+100);
25             tMoveTo(x, y);
26         }
27     }
28
29     /* 左クリックされている間、繰り返す */
30     while (tGetClick() == 1) {
31         x = tClickX();
32         y = tClickY();
33         /* 領域外がクリックされたのならば無視する */
34         if (x < -250 || 250 <= x || y < -250 || 250 <= y)
35             continue;
36         /* クリックされた区画の添字を求める */
37         i = (x + 250) / 100;
38         j = (y + 250) / 100;
39         /* この区画の色を変える */
40         color[i][j] = (color[i][j] + 1) % 3;
41         /* クリックされた区画(正方形)の左下の頂点の座標を求める */
```

```
42          x = i * 100 - 250;
43          y = j * 100 - 250;
44          /* この区画の境界を黒色で描きなおす */
45          tPenUp();
46          tMoveTo(x, y);
47          tPenDown();
48          tSetColor(0.0, 0.0, 0.0);
49          tMark();
50          tMoveTo(x+100, y);
51          tMoveTo(x+100, y+100);
52          tMoveTo(x, y+100);
53          tMoveTo(x, y);
54          /* この区画の内部を新しい色で塗りつぶす */
55          if (color[i][j] == 0)
56              tSetColor(1.0, 1.0, 1.0);
57          else if (color[i][j] == 1)
58              tSetColor(1.0, 0.0, 0.0);
59          else
60              tSetColor(0.0, 0.0, 1.0);
61          tFill();
62      }
63 }
```

6行目で宣言されている2次元配列 color では、次の図のように、各区画の現在の色を配列の要素に対応させて記憶するようにしています。配列 color の要素の値が0であれば、対応する区画の色は白、1であれば赤、2であれば青を意味しています。

$(-250, 250)$ $(250, 250)$

color[0][4]	color[1][4]	color[2][4]	color[3][4]	color[4][4]
color[0][3]	color[1][3]	color[2][3]	color[3][3]	color[4][3]
color[0][2]	color[1][2]	color[2][2]	color[3][2]	color[4][2]
color[0][1]	color[1][1]	color[2][1]	color[3][1]	color[4][1]
color[0][0]	color[1][0]	color[2][0]	color[3][0]	color[4][0]

$(-250, -250)$ $(250, -250)$

tiles.c の 34 行目では、クリックされた座標が

$$\left\{ (x,\, y) \,\middle|\, -250 \leqq x < 250 \text{ かつ } -250 \leqq y < 250 \right\}$$

の領域に含まれているかどうかをチェックしています。このチェックを行わないと、領域外がクリックされた場合に、37 行目と 38 行目で計算される (配列 color の) 添字 i と j が、許される範囲 (0 から 4 まで) の外になってしまうためです。領域外がクリックされた場合、前回に登場した continue 文が実行されますので、36 行目から 61 行目の文はスキップされ、30 行目の while 文の条件式のチェックに進みます。

メモ

14.5 付録 : 配列を使った 4 桁の数当てゲーム

hitblow4.c

```
 1 #include <turtle.h>
 2 #include <stdlib.h>
 3 #include <time.h>
 4
 5 main()
 6 {
 7     int try, hit, blow;
 8     int x[4], g[4];         /* 一の位が[0]、千の位が[3] */
 9     int i, j;
10     int guess;
11
12     /* 現在時刻を乱数の種として設定 */
13     srand(time(0));
14
15     /* 正解(1〜9の数字4個の重複のない並び)を乱数で生成する */
16     i = 0;
17     while (i < 4) {
18         x[i] = rand()%9+1;              /* この位の数字を選ぶ */
19         for (j = 0; j < i; j++) {       /* 数字の重複をチェック */
20             if (x[i] == x[j])
21                 break;
22         }
23         if (j == i)                     /* 重複していなければ採用 */
24             i++;
25     }
26
27     tPenUp();
28     tMoveTo(-100, 200);
29
```

```
30      for (try = 1; try <= 10; try++) {
31          /* キーボードから4桁の数を入力 */
32          tPrintf("%d回目 ? ", try);
33          tScanf("%d", &guess);
34
35          /* 各桁を取り出す */
36          for (i = 0; i < 4; i++) {
37              g[i] = guess % 10;
38              if (g[i] <= 0)              /* 1〜9であることをチェック */
39                  break;
40              for (j = 0; j < i; j++) {   /* 重複をチェック */
41                  if (g[i] == g[j])
42                      break;
43              }
44              if (j < i)
45                  break;
46              guess /= 10;
47          }
48
49          /* 1〜9の数字の相異なる4個だったかどうかをチェックする */
50          if (i != 4 || guess != 0)
51              continue;
52
53          /* hit と blow の数を数える */
54          hit = blow = 0;
55          for (i = 0; i < 4; i++) {
56              for (j = 0; j < 4; j++) {
57                  if (g[i] == x[j]) {
58                      if (i == j)
59                          hit++;
60                      else
61                          blow++;
62                  }
63              }
64          }
65
66          /* 正解なら繰り返しを終了 */
67          if (hit == 4)
68              break;
69
70          /* hit と blow の数を表示 */
71          tPrintf("%dH%dB\n", hit, blow);
72      }
73
74      /* 10回以内で正解が入力されなかったら、正解を表示する。 */
75      if (try <= 10)
76          tPrintf("おめでとう。正解です。\n");
77      else {
78          tPrintf("残念でした。正解は ");
79          for (i = 3; i >= 0; i--)
80              tPrintf("%d", x[i]);
81          tPrintf(" です。\n");
82      }
83  }
```

付録：前回の演習問題のプログラム例

hitblow0.c

```
 1 #include <turtle.h>
 2 #include <stdlib.h>
 3 #include <time.h>
 4
 5 main ()
 6 {
 7     int try, hit, blow;
 8     int x1, x2, x3, g1, g2, g3;
 9     int guess;
10
11     /* 現在時刻を乱数の種として設定 */
12     srand(time(0));
13
14     /* 正解(0〜9の数字3個の重複のない並び)を乱数で生成する */
15     x1 = rand()%10;          /* 左端 */
16     do {
17         x2 = rand()%10;      /* 真中 */
18     } while (x2 == x1);      /* 同じ数だったらもう一度 */
19     do {
20         x3 = rand()%10;      /* 右端 */
21     } while (x3 == x1 || x3 == x2);     /* 同じ数だったらもう一度 */
22
23     tPenUp();
24     tMoveTo(-100, 200);
25
26     for (try = 1; try <= 10; try++) {
27         /* キーボードから3桁の数を入力 */
28         tPrintf ("%d回目 ? ", try);
29         tScanf("%d", &guess);
30
31         /* 各桁を取り出す */
32         g1 = guess/100;      /* 左端 */
33         g2 = guess/10%10;    /* 真中 */
34         g3 = guess%10;       /* 右端 */
35
36         /* 0〜9の数字3個の並びかどうかをチェックする */
37         if (g1 > 9 || g1 < 0 || g2 < 0 || g3 < 0)
38             continue;
39
40         /* その3個の数字に重複がないかどうかをチェックする */
41         if (g1 == g2 || g2 == g3 || g3 == g1)
42             continue;
43
44         /* hit と blow の数を数える */
45         hit = 0; blow = 0;
46         if (g1 == x1) hit++;
47         if (g2 == x2) hit++;
48         if (g3 == x3) hit++;
49         if (g1 == x2 || g1 == x3) blow++;
50         if (g2 == x3 || g2 == x1) blow++;
51         if (g3 == x1 || g3 == x2) blow++;
52
53         /* 正解なら繰り返しを終了 */
54         if (hit == 3)
55             break;
56
```

```
57          /* hit と blow の数を表示 */
58          tPrintf ("%dH%dB\n", hit, blow);
59      }
60
61      /* 10回以内で正解が入力されなかったら、正解を表示する。 */
62      if (try <= 10)
63          tPrintf ("おめでとう。 正解です。\n");
64      else
65          tPrintf ("残念でした。 正解は %d%d%d です。\n", x1, x2, x3);
66  }
67
```