プログラミングおよび実習II・第 3 回

2003 年 10 月 8 日

1　選択ソートの復習

1.1　実力チェック

前回の 5 ページにある関数 `selectsort` のプログラムを、次の「実現の方針」に従ってデータを大きい順 (降順) に並び替えるように変更しなさい。ただし、解答用紙のプログラム名は「`prog3-1.c`」とし、関数 `selectsort` の (変更後の) 定義を書きなさい。

実現の方針 (1) 配列中で最も小さな値を持つ要素を探し出し、これを最後尾の要素と交換する。

(2) 次に、最後尾の要素を除いて最も小さな値を持つ要素を探し出し、これを末尾から 2 番目の要素と交換する。

(3) さらに、末尾の 2 つの要素を除いて最も小さな値を持つ要素を探し出し、これを末尾から 3 番目の要素と交換する。

(4) さらに、末尾の 3 つの要素を除いて最も小さな値を持つ要素を探し出し、これを末尾から 4 番目の要素と交換する。

以下同様に、最も小さな値を探す領域が 1 個の要素になるまで繰り返す。

2　バブルソートのプログラミング

単純な整列 (ソーティング) アルゴリズムの 1 つである**バブル (bubble) ソート**と呼ばれる整列アルゴリズムのプログラミングをしてみましょう[1]。

2.1　バブルソートのアルゴリズム

配列中にいくつかの整数データが格納されているとします。これを配列の先頭側により小さいデータが来るように (昇順に) 並び替えたいとします。

実現の方針 (1) 配列中のデータの先頭から後方に向けて、隣り合った 2 つのデータを比較し、小さい方が前に来るように (必要なら交換) するという操作を、添字を 1 ずつずらしながら、最後尾まで繰り返す。この操作で、配列中の最も大きいデータが最後尾に移動する。

(2) 最後尾の要素だけを除いた残りの部分について、 (1) の操作をもう一度行えば、 2 番目に大きいデータがその部分の最後に移動してくる。 さらに、この (最後尾から 2 番目の) 要素も除外して (2) の操作を行い、次は最後尾から 3 番目の要素も除外して ... と繰り返せば、ついには元の配列全体を昇順に整列させることができる。

アルゴリズム (1) 整数データが格納されている配列を `data`、データの個数を `num` とする。

(2) 整数型変数 `last`、`i`、`tmp` を用意する。

[1] 教科書「C 言語によるアルゴリズムとデータ構造入門」の 91 ページからを適宜参照してください

(3) last に num − 1 を代入する。

(4) 0 < last が成り立っている限り、次の 3 つの作業を順に繰り返す。

(4-1) i に 0 を代入する。

(4-2) i < last が成り立っている限り、次の 2 つの作業を順に繰り返す。

(4-2-1) もし data[i] > data[i+1] ならば、data[i] と data[i+1] を交換する。 つまり、

- tmp へ data[i] の値を退避する。
- data[i] へ data[i+1] をコピーする。
- data[i+1] へ tmp をコピーする。

(4-2-2) i の値を 1 増やす。

(4-3) last の値を 1 減らす。

バブルソートを行う関数 bubblesort のプログラム例

```
 1 void bubblesort(int data[], int num)
 2 {
 3     int last, i;
 4     int tmp;
 5
 6     for (last = num-1; 0 < last; last--) {
 7         for (i = 0; i < last; i++) {
 8             if (data[i] > data[i+1]) {
 9                 tmp = data[i];
10                 data[i] = data[i+1];
11                 data[i+1] = tmp;
12             }
13         }
14     }
15 }
```

この関数 bubblesort の 6 行目と 7 行目の間に showdata の呼び出しを追加して、前回の演習問題 prog2-4.c で使った関数 selectsort をこの bubblesort で置き換えたプログラムを prog3-2.c とすると、その実行例はつぎのようなものとなります。

prog3-2.c の実行例

```
s1542h017% ./prog3-2
 1番目のデータ => 67
 2番目のデータ => 98
 3番目のデータ => 19
 4番目のデータ => 45
 5番目のデータ => 74
 6番目のデータ => 88
 7番目のデータ => 28
 8番目のデータ => 19
 9番目のデータ => 7
10番目のデータ => 55
11番目のデータ => .
 67, 98, 19, 45, 74, 88, 28, 19,  7, 55
 67, 19, 45, 74, 88, 28, 19,  7, 55, 98
 19, 45, 67, 74, 28, 19,  7, 55, 88, 98
 19, 45, 67, 28, 19,  7, 55, 74, 88, 98
```

```
 19, 45, 28, 19,  7, 55, 67, 74, 88, 98
 19, 28, 19,  7, 45, 55, 67, 74, 88, 98
 19, 19,  7, 28, 45, 55, 67, 74, 88, 98
 19,  7, 19, 28, 45, 55, 67, 74, 88, 98
  7, 19, 19, 28, 45, 55, 67, 74, 88, 98
  7, 19, 19, 28, 45, 55, 67, 74, 88, 98
s1542h017%
```

3 演習問題

以下の 3 つの演習問題に取り組みなさい。プログラムが完成したら、前回と同様に mprog2 コマンドを実行して、自分が作成したプログラムを提出してください。

3.1 prog3-1.c

前回作成した prog2-4.c を prog3-1.c にコピーし、「1.1 実力チェック」で行ったのと同様の変更をこのプログラム prog3-1.c に加え、実際に計算機上で作成、コンパイル、実行して、次の実行例のように動作することを確認しなさい。

prog3-1.c の実行例

```
s1542h017% ./prog3-1
 1番目のデータ => 67
 2番目のデータ => 98
 3番目のデータ => 7
 4番目のデータ => 19
 5番目のデータ => 45
 6番目のデータ => 74
 7番目のデータ => 21
 8番目のデータ => 5
 9番目のデータ => 28
10番目のデータ => 55
11番目のデータ => .
 67, 98,  7, 19, 45, 74, 21,  5, 28, 55
 67, 98,  7, 19, 45, 74, 21, 55, 28,  5
 67, 98, 28, 19, 45, 74, 21, 55,  7,  5
 67, 98, 28, 55, 45, 74, 21, 19,  7,  5
 67, 98, 28, 55, 45, 74, 21, 19,  7,  5
 67, 98, 74, 55, 45, 28, 21, 19,  7,  5
 67, 98, 74, 55, 45, 28, 21, 19,  7,  5
 67, 98, 74, 55, 45, 28, 21, 19,  7,  5
 74, 98, 67, 55, 45, 28, 21, 19,  7,  5
 98, 74, 67, 55, 45, 28, 21, 19,  7,  5
s1542h017%
```

注意: このプログラムは mprog2 コマンドで提出すると、教員か TA のチェックを受けるように要求されます。mprog2 コマンドが表示する指示に従ってください。

3.2 prog3-2.c

まず、前回作成した prog2-4.c を、prog3-2.c にコピーしなさい。次に、関数 selectsort を、2 ページの関数 bubblesort に置き換えるとともに、関数 bubblesort から showdata を呼び出すよう

にして、バブルソート法での整列の過程が 2 ページの実行例のように表示されるように prog3-2.c を変更しなさい。

3.3 prog3-3.c

prog3-2.c を、prog3-3.c にコピーし、次の実現の方針に従ってバブルソートを行うように、このプログラム prog3-3.c の関数 bubblesort の定義を書き換えなさい。

実現の方針 (1) 配列中のデータの最後尾から前方に向けて、隣り合った 2 つのデータを比較し、小さい方が前に来るように (必要なら交換) するという操作を、添字を 1 ずつずらしながら配列の先頭まで繰り返す。この操作で、配列中の最も小さいデータが配列の先頭に移動する。

(2) この配列から先頭要素だけを除いた残りの部分を 1 つの配列と見なし、(1) の操作をもう一度行えば、2 番目に小さいデータがその部分の先頭に移動してくる。 さらに、この先頭から 2 番目の要素も除外して (2) の操作を行い、次は 3 番目の要素も除外して ... と繰り返せば、ついには元の配列全体を昇順に整列させることができる。

prog3-3.c の実行例

```
s1542h017% ./prog3-3
 1番目のデータ => 67
 2番目のデータ => 98
 3番目のデータ => 19
 4番目のデータ => 45
 5番目のデータ => 74
 6番目のデータ => 88
 7番目のデータ => 28
 8番目のデータ => 19
 9番目のデータ => 7
10番目のデータ => 55
11番目のデータ => .
 67, 98, 19, 45, 74, 88, 28, 19,  7, 55
  7, 67, 98, 19, 45, 74, 88, 28, 19, 55
  7, 19, 67, 98, 19, 45, 74, 88, 28, 55
  7, 19, 19, 67, 98, 28, 45, 74, 88, 55
  7, 19, 19, 28, 67, 98, 45, 55, 74, 88
  7, 19, 19, 28, 45, 67, 98, 55, 74, 88
  7, 19, 19, 28, 45, 55, 67, 98, 74, 88
  7, 19, 19, 28, 45, 55, 67, 74, 98, 88
  7, 19, 19, 28, 45, 55, 67, 74, 88, 98
  7, 19, 19, 28, 45, 55, 67, 74, 88, 98
s1542h017%
```

余裕のある受講者は、次の実行例のように (関数 bubblesort が) 配列中のどの要素を入れ替えようとしたのかが分かるプログラム prog3-4.c を作ってみましょう[2] (このプログラムは mprog2 コマンドで提出する必要はありません)。^^^^ で示された範囲の要素が、次の行では右方向にそれぞれ 1 つずつずれている (右端の要素は左端に来ている) ことに注意してください。

[2]配列中の整数は高々 3 桁だと思って構いません。

prog3-4.c の実行例

```
s1542h017% ./prog3-4
 1番目のデータ => 67
 2番目のデータ => 98
 3番目のデータ => 19
 4番目のデータ => 45
 5番目のデータ => 74
 6番目のデータ => 88
 7番目のデータ => 28
 8番目のデータ => 19
 9番目のデータ => 7
10番目のデータ => 55
11番目のデータ => .
 67, 98, 19, 45, 74, 88, 28, 19,  7, 55
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  7, 67, 98, 19, 45, 74, 88, 28, 19, 55
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  7, 19, 67, 98, 19, 45, 74, 88, 28, 55
       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  7, 19, 19, 67, 98, 28, 45, 74, 88, 55
            ^^^^^^^^^^^^    ^^^^^^^^^^^^
  7, 19, 19, 28, 67, 98, 45, 55, 74, 88
                ^^^^^^^^^^^^
  7, 19, 19, 28, 45, 67, 98, 55, 74, 88
                   ^^^^^^^^^^^^
  7, 19, 19, 28, 45, 55, 67, 98, 74, 88
                              ^^^^^^^^
  7, 19, 19, 28, 45, 55, 67, 74, 98, 88
                                ^^^^^^^^
  7, 19, 19, 28, 45, 55, 67, 74, 88, 98

  7, 19, 19, 28, 45, 55, 67, 74, 88, 98
s1542h017%
```

プログラミングおよび実習Ⅱ・第 3 回・終り

付録

前回の演習問題 prog2-5.c のプログラム例

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3
 4 #define MAX_DATA        1000
 5
 6 void showdata (int data[], int num)
 7 {
 8     int i;
 9
10     if (num > 0) {
11         printf("%3d", data[0]);
12         for (i = 1; i < num; i++)
13             printf(",%3d", data[i]);
14     }
15     printf("\n");
16 }
```

```
17
18 void showexchange (int p, int q)
19 {
20     int i;
21
22     for (i = 0; i <= p || i <= q; i++) {
23         if (i == p || i == q)
24             printf("  ^ ");
25         else
26             printf("    ");
27     }
28     printf("\n");
29 }
30
31 void selectsort(int data[], int num)
32 {
33     int start, pos, i;
34     int min, tmp;
35
36     for (start = 0; start < num-1; start++) {
37         showdata(data, num);
38         min = data[start];
39         pos = start;
40         for (i = start+1; i < num; i++) {
41             if (data[i] < min) {
42                 min = data[i];
43                 pos = i;
44             }
45         }
46         showexchange(start, pos);
47         if (pos != start) {
48             tmp = data[start];
49             data[start] = data[pos];
50             data[pos] = tmp;
51         }
52     }
53 }
54
55 int main ()
56 {
57     int data[MAX_DATA];
58     int num;
59
60     for (num = 0; num < MAX_DATA; num++) {
61         printf("%2d番目のデータ => ", num+1);
62         if (scanf ("%d", &data[num]) != 1)
63             break;
64     }
65
66     selectsort(data, num);
67     showdata(data, num);
68     return EXIT_SUCCESS;
69 }
```