2005年度 プログラミング演習 II　演習12

2006年1月18日 (水)

演習 1　以下は「選択ソートによるソーティングプログラム」である。プログラミング演習テキストの表 9.1 (p.149) とプログラムの実行結果が一致するよう、空欄 ☐ を埋めプログラムを完成しなさい。

● 選択ソートによるソーティングプログラム　p2exercise1201.c

```
 1: #include <stdio.h>
 2:
 3: #define N 10
 4:
 5: void selectionsort(int a[]);
 6: void printarray(int a[]);
 7:
 8: int main(void)
 9: {
10:     int a[N+1] = {0, 4, 10, 5, 2, 1, 7, 8, 6, 3, 9};
11:
12:     selectionsort(a);
13:
14:     return 0;
15: }
16:
17: void selectionsort(int a[])
18: {
19:     int i, j, min, point;      ← 「point」は最小値「min」の位置を記憶する
20:
21:     for (i = 1; i <= N-1; i++) {
22:         min = a[i];
23:         point = i;
24:         for (j = i+1; j <= N; j++) {          ← 最小値を見つけるループ
25:             if (min > a[j]) {
26:                 [          ] ;
27:                 [          ] ;
28:             }
29:         }
30:         printarray(a);                        ← 交換前のデータ列を表示
31:         a[point] = a[i];
32:         a[i] = min;
33:         printarray(a);                        ← 交換後のデータ列を表示
34:         printf("¥n");
35:     }
36: }
```

```
37:
38: void printarray(int a[])
39: {
40:     int i;
41:
42:     for (i = 1; i <= N; i++) printf("%3d ", a[i]);
43:     printf("¥n");
44: }
```

演習 2　以下は「バブルソートによるソーティングプログラム」である。プログラミング演習テキストの表 9.2 (p.150) および表 9.3 (p.151) とプログラムの実行結果が一致するよう、空欄 □ を埋めプログラムを完成しなさい。ただし、交換が起こらない Step があった時点でソーティングは完了するので、その時点でプログラムがストップするように改良してある (Step 7 を終了した時点でプログラムはストップする)。

● バブルソートによるソーティングプログラム　　p2exercise1202.c

```
 1: #include <stdio.h>
 2:
 3: #define N 10
 4:
 5: void bubblesort(int a[]);
 6: void printarray(int a[]);
 7:
 8: int main(void)
 9: {
10:     int a[N+1] = {0, 4, 10, 5, 2, 1, 7, 8, 6, 3, 9};
11:
12:     bubblesort(a);
13:
14:     return 0;
15: }
16:
17: void bubblesort(int a[])
18: {
19:     int i, j, flag, tmp;                    ← 「flag」で交換の有無を判別
20:                                               0 なら交換無し、1 なら交換有り
21:     for (i = N-1; i >= 2; i--) {
22:         flag = 0;                                ← 交換無しにセットする
23:         printarray(a);                  ← 各 Step の交換前のデータ列を表示
24:         for (j = 1; j <= i; j++) {
25:             if ( [          ] ) {   ← 隣り合う要素の大小関係が逆ならば
26:                 tmp = a[j];
27:                 a[j] = a[j+1];
```

```
28:                 a[j+1] = tmp;
29:                 [          ] ;              ← 交換有りにセットする
30:             }
31:             printarray(a);      ← 交換の有無を確認するためにデータ列を表示
32:         }
33:         printf("¥n");
34:         if (flag == 0) break;        ← 交換が 1 度も無ければループを抜ける
35:     }
36: }
37:
38: void printarray(int a[])
39: {
40:     int i;
41:
42:     for (i = 1; i <= N; i++) printf("%3d ", a[i]);
43:     printf("¥n");
44: }
```

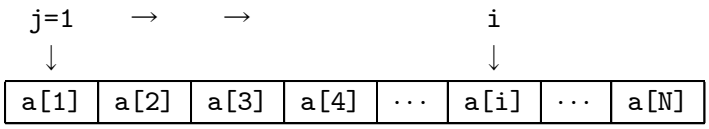
演習 3　以下は「挿入ソートによるソーティングプログラム」である。プログラミング演習テキストの表 9.4 (p.152) とプログラムの実行結果が一致するよう、空欄 ☐ を埋めプログラムを完成しなさい。

● 挿入ソートによるソーティングプログラム　p2exercise1203.c

```
 1: #include <stdio.h>
 2:
 3: #define N 10
 4:
 5: void insertionsort(int a[]);
 6: void printarray(int a[]);
 7:
 8: int main(void)
 9: {
10:     int a[N+1] = {0, 4, 10, 5, 2, 1, 7, 8, 6, 3, 9};
11:
12:     insertionsort(a);
13:
14:     return 0;
15: }
16:
17: void insertionsort(int a[])
18: {
19:     int i, j, k, tmp;
20:
```

```
21:     for (i = 2; i <= N; i++) {
22:         j = 1;
23:         printarray(a);                          ← 挿入前のデータ列を表示
24:         while (a[j] < a[i]) j++;      ← 1 から i の範囲で挿入位置を見つける
25:         tmp = a[i];                          ← 要素 a[i] を一時退避しておく
26:         for (k = i; k >= j+1; k--)  [          ] ;
 ↑ 挿入箇所 a[j] を空けるため、a[j] から a[i-1] の要素を後ろから順に後ろに移動する
                          a[i-1] → a[i], a[i-2] → a[i-1], ⋯ , a[j] → a[j+1]
27:         a[j] = tmp;       ← 見つけた挿入位置に退避しておいた要素を代入する
28:         printarray(a);                          ← 挿入後のデータ列を表示
29:         printf("¥n");
30:     }
31: }
32:
33: void printarray(int a[])
34: {
35:     int i;
36:
37:     for (i = 1; i <= N; i++) printf("%3d ", a[i]);
38:     printf("¥n");
39: }
```

ポイント：第 22 行〜第 24 行について (1 から i の範囲で挿入位置を見つける)



* 継続条件「a[j] < a[i]」より、i と j が等しくなった時点でループを抜ける (∵i=j より a[i]=a[j])。もし、i と j が等しくなった時点でループを抜けたなら、整列済みデータ列の最後尾に挿入することになる。言い換えれば、a[i] を元の位置に再代入することになる。