7. 選択法による並べかえ

前章で、配列を使った応用として、配列の要素を小さい順に並べかえるプログラムを示した。そのプログラムでは、隣り合う 2 つの要素の大きさを比べ、小さい方が後ろにあったらその 2 つを入れ換える、というアルゴリズム (bubble sort) を採用した。ここでは同じ並べかえを、選択法 (selection sort) という別のアルゴリズムで解いてみよう[1]。

選択法の概略

選択法の方針は、「一番小さいものを見つけて、それを先頭に持ってくる」となる。今、整数配列要素 a[0]〜a[3] に次のようなデータが入っているとしよう[2]。

	0	1	2	3
a	10	5	32	9
		↑		

これを小さい順に並べかえていくわけだが、選択法では、まず a[0] から a[3] の中で一番小さい要素を探す。図では a[1](矢印がついているところ) が 5 で、一番小さい。そして、一番小さい要素 a[1] を、先頭の要素 a[0] と入れ換える。すると、配列は次のように変わる。

	0	1	2	3
a	5	10	32	9

今の操作によって、a[0] に入れる数が確定する。何故なら、小さい順に入れた時、a[0] には全体で最も小さい数が入るはずだが、上の操作によって必ずそうなるからである。しかしまだ、a[1] 以降は確定していない。ただ a[0] が確定したので、あとは a[1] から a[3] を並べかえればよいことになる。そこで次に、a[1] から a[3] の中で一番小さい要素を探す。次の図で、a[3] (矢印がついている) が、それらの中で一番小さい。

	0	1	2	3
a	5	10	32	9
				↑

a[0] と a[1] の間を 2 重線にして、a[0] まで確定しているということをはっきりさせている。ここで、最初の操作の時と同じように、2 重線以降で最も小さい要素 a[3] と、2 重線以降の先頭の要素 a[1] を入れ換える。すると配列は次のようになる。

	0	1	2	3
a	5	9	32	10
				↑

これで、a[0]、a[1] が確定した。あとは a[2] から a[3] を並べかえればよいことになる。先ほどと同じように、a[2] から a[3] のなかで一番小さいものを探して (上で矢印がついているのが小さい)、それをまだ確定していない要素の先頭に持ってくると、配列は次のようになる。

	0	1	2	3
a	5	9	10	32

これで a[0] から a[2] が確定し、残るは a[3] だけ、となるわけだが、残りが一つしかなければ並べかえは起こらないので、これで並べかえは終了する。実際、配列 a は小さい順になっている!

[1] 一般に問題が複雑になると、それを解くアルゴリズムは複数出てくる。言い換えれば、正解は複数ある。そしてたいていはアルゴリズムによって、効率その他の性質が微妙に異なる。それらの中で絶対的に一番良いアルゴリズムが存在することもあるが、たいていは問題の性質によってどのアルゴリズムが良いかということも変わってくる。単に解ければよいというだけでなく、問題の性質に合ったアルゴリズムを選ぶことも重要である。

[2] C 言語では配列を a[4] と宣言すると a[0] から a[3] の 4 つの領域が取られる。それに合わせて、以下の説明においてもインデックスを 0 〜 3 としてある。

選択法のアルゴリズム

さて、ここまでは配列の要素の数を 4 として考えてきた。しかし、上に述べてきたことは、一般に配列の要素の数が N の時でも成り立つ。選択法の方針は、頭から一つずつ確定していき、残りが 1 つになったら終り、というものなので、配列の要素の数には関係なく正しく動く。

ここまでで選択法のおおまかな方針はわかったので、これをアルゴリズムとして記述してみよう。N に配列の要素の数が、a[0] から a[N-1] に並べかえるべきデータが与えられているものとする。

[ステップ 1] $i \leftarrow 0$ とする。

[ステップ 2] もし $i \leq N-2$ ならステップ 3 へ進む。そうでなければアルゴリズムは終了、この時配列 a は小さい順に並べかえられている。

[ステップ 3] $a[i], \ldots, a[N-1]$ の中で一番小さい要素のインデックスを $small$ とする ($a[small]$ が一番小さい要素となるように $small$ を定める)。

[ステップ 4] $a[i]$ と $a[small]$ の値を入れ換える。

[ステップ 5] $i \leftarrow i+1$ とし、ステップ 2 へ戻る。

上のアルゴリズムの中で、i は今何番目まで確定しているかを覚えているために使われている。最初は一つも確定していないので、$i \leftarrow 0$ にしている。また、$small$ は何番目の要素が最も小さいかを記憶しておくために使われている。最も小さい要素の値自身を記憶するのでなく、何番目が一番小さいかを示すインデックス (添字) を記憶させるのは、ステップ 4 で値を入れ換える時に「どことどこを入れ換えたらいいか」がわからないといけないからである。

さて、アルゴリズムの記述としてはこれで充分だが、ステップ 3、4 は、いくつかの操作を一口で言ってしまっていて、若干不親切である。実際、上の記述からプログラムにせよと言われると、考え込んでしまう人がいるかもしれない。そこで、ステップ 3、4 の手順をもう少し詳しく述べることにする。

ステップ 3 は、次の副問題を解くことになる。

[**副問題** 1] 配列の $a[i]$ から $a[N-1]$ の要素のうち、一番小さい要素のインデックス $small$ を求めよ

この問題を解くには、最初 $small \leftarrow i$ としておいて、別の変数 j を $i+1$ から $N-1$ まで動かしながら、もし $a[small] > a[j]$ なら (もっと小さいものが見つかったことになるので)、$small \leftarrow j$ と改める、という操作を繰り返すということになる。これをアルゴリズムで書くと次のようになる。

[ステップ 3-1] $small \leftarrow i$ とする。

[ステップ 3-2] $j \leftarrow i+1$ とする。

[ステップ 3-3] もし $j \leq N-1$ ならステップ 3-4 へ進む。そうでなければステップ 3 は終了、この時 $small$ は $a[i]$ から $a[N-1]$ の中の最小の要素のインデックスとなっている。

[ステップ 3-4] もし $a[j] < a[small]$ ならば、$small \leftarrow j$ とする。

[ステップ 3-5] $j \leftarrow j+1$ とし、ステップ 3-3 へ戻る。

これで副問題 1 が解け、ステップ 3 の詳細ができた。

次にステップ 4 の詳細だが、これは次の副問題に帰着する。

[**副問題** 2]
$a[i]$ と $a[small]$ の値を入れ換えよ。

プログラミング言語によっては、値の交換を行う命令があるものもあるが、残念ながら C 言語にはない。そのような言語の場合、値の交換は一つ別の変数を使わなければできない。そこでその変数を $work$ という名前にして、ステップ 4 の詳細を書くと次のようになる。

[ステップ 4-1] $work \leftarrow a[i]$

[ステップ 4-2] $a[i] \leftarrow a[small]$

[ステップ 4-3] $a[small] \leftarrow work$

このように、最初に $a[i]$ の値を $work$ に取っておいて、あとでそれを $a[small]$ に入れるようにする必要がある (もっとも最初に $a[small]$ の値を $work$ に取っておいてもいいが)。

以上をまとめると、選択法のアルゴリズムは次のようになる。

a[0] から a[N-1] を並べかえる選択法のアルゴリズム

[ステップ 1] $i \leftarrow 0$ とする。

[ステップ 2] もし $i \leq N-2$ ならステップ 3 へ進む。そうでなければアルゴリズムは終了、この時配列 a は小さい順に並べかえられている。

[ステップ 3]

　[ステップ 3-1] $small \leftarrow i$ とする。

　[ステップ 3-2] $j \leftarrow i+1$ とする。

　[ステップ 3-3] もし $j \leq N-1$ ならステップ 3-4 へ進む。そうでなければステップ 3 は終了、この時 $small$ は $a[i]$ から $a[N-1]$ の中の最小の要素のインデックスとなっている。

　[ステップ 3-4] もし $a[j] < a[small]$ ならば、$small \leftarrow j$ とする。

　[ステップ 3-5] $j \leftarrow j+1$ とし、ステップ 3-3 へ戻る。

[ステップ 4]

　[ステップ 4-1] $work \leftarrow a[i]$

　[ステップ 4-2] $a[i] \leftarrow a[small]$

　[ステップ 4-3] $a[small] \leftarrow work$

[ステップ 5] $i \leftarrow i+1$ とし、ステップ 2 へ戻る。

以上のように、問題をまず大まかに捉えて解き方の概略を考え、次にその概略の中の副問題を解いていく (必要なら更に副副問題に分割する) という、段階的な問題解決法を、トップダウン思考法、あるいはトップダウン問題解決という。問題が大きくなるとトップダウン思考は重要となる。

[練習 1] 以上の説明に基づいて、選択ソートを行なう C 言語のプログラムを実現しなさい。ただし入力データは全て整数であるとする。

[練習 2] 練習 1 のプログラムを、–1 が入力されるまでデータを配列に読み込み、それらを小さい順に並べかえ出力するようにしなさい。