プログラミングおよび実習II・第 2 回

2003 年 10 月 1 日

1 基本的なプログラミングの復習 (続き)

1.1 実力チェック

下の実現方針とアルゴリズムにしたがって、標準入力 (キーボード) から整数データをつぎつぎと読み込み、それらの最大値と、その最大値が何番目に入力されたデータであったかを、次の実行例のような形式で表示するような C プログラム prog2-1.c を作りなさい。ただし、最大値が複数回入力された場合は、最初のものを採用するようにしなさい。整数でない入力や、EOF を検出したら、そこで入力を終了するものとします。整数が 1 つも入力されなかった場合は、プログラムは何も出力せずに終了するようにしなさい。

prog2-1.c の実行例

```
s1542h017% ./prog2-1
1番目の整数を入力してください => 65
2番目の整数を入力してください => 35
3番目の整数を入力してください => 78
4番目の整数を入力してください => 213
5番目の整数を入力してください => 117
6番目の整数を入力してください => 98
7番目の整数を入力してください => .
最大値は4番目のデータ 213 です。
s1542h017% ./prog2-1
1番目の整数を入力してください => 53
2番目の整数を入力してください => .
最大値は1番目のデータ 53 です。
s1542h017% ./prog2-1
1番目の整数を入力してください => .
s1542h017%
```

実現の方針 (1) 常に「今までに読み込んだデータの個数」と「これまでの最大値」、「最大値が何番めのデータであったのか」を保持しておく。

(2) 新しいデータを読み込む度に、これら 3 つの情報を更新していく。

(3) すべてのデータを読み終えた段階で、もし「今までに読み込んだデータの個数」が 0 でなければ、「これまでの最大値」と「最大値が何番目のデータであったのか」を出力する。

アルゴリズム (1) 4 つの整数型変数 num、max、maxpos と new を用意する。

(2) num を 0 で初期化する。

(3) 次の 4 つの作業を順に繰り返す。

(3-1) num + 1 番目のデータの入力を促す。

(3-2) 入力された整数を new に読み込む。読み込めなければ (もう入力がなければ) 繰り返しを終了する。

(3-3) もし、 num = 0 あるいは new > max が成り立っていれば、

(3-3-1) max へ new を代入する。

(3-3-2) maxpos へ num + 1 を代入する。

(3-4) num を 1 増やす。

(4) もし num が 0 でなかったら、

(4-1) maxpos 番目のデータが最大値 max であったことを表示する。

1.2 標準偏差を求める

標準入力 (キーボード) からいくつかの数値データ (小数部があっても可) をつぎつぎと読み込んで、それらの平均値と標準偏差を小数点以下 2 桁まで出力するような C プログラム prog2-2.c を作りなさい。ただし、入力されるデータの個数は 1000 個以下であると仮定してよい。

ヒント (1) データの値が $x_1, x_2, x_3, \ldots, x_n$ で、それらの平均が m であった場合、これらのデータの標準偏差 σ は

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - m)^2}{n}}$$

で定義される。この計算には平均値が必要となるので、すべてのデータを読み込んだ後に計算せざるを得ない。また、すべてのデータの値が必要となるため、読み込まれた各データの値はどこかに記憶しておかなければならない。このために配列が必要となる。

実現の方針 (1) データの個数を数えながら、読み込んだデータをすべて配列中に記憶していく。

(2) 前回の「1-3 実力チェック」と同様の方針で、平均値を計算する。ただし、データを読み込みながら総和を求める代わりに、配列中に格納されたデータを順に参照しなが総和を求める。

(3) 得られた平均値と配列中の各データの値を参照して標準偏差を計算する。

(4) 平均値と標準偏差を出力する。

アルゴリズム (1) 大きさ 1000 の浮動小数点型配列 data と整数型変数 num、i、浮動小数点型変数 sum、mean、vsum、sigma を用意する。

(2) 配列 data にデータを読み込むために、次の作業を行う。

(2-1) num を 0 で初期化する。

(2-2) num < 1000 が成り立っている限り、次の 2 つ作業を順に繰り返す。

(2-2-1) 入力されたデータを data[num] に読み込む。
読み込めなければ (もうデータがなければ) 繰り返しを終了。

(2-2-2) num の値を 1 増やす。

(3) num が 0 なら全体の処理を終了する。

(4) 平均値を求めるために、次の作業を行う。

(4-1) sum を 0.0 で初期化する。

(4-2) i に 0 を代入する。

(4-3) i < num が成り立っている限り、次の 2 つの作業を順に繰り返す。

(4-3-1) data[i] を sum へ足し込む。

(4-3-2) i の値を 1 増やす。

(4-4) mean に $\frac{\texttt{sum}}{\texttt{num}}$ を代入する。

(5) 標準偏差を計算するために、次の作業を行う。

(5-1) vsum を 0.0 で初期化する。

(5-2) i に 0 を代入する。

(5-3) i < num が成り立っている限り、次の 2 つの作業を順に繰り返す。

(5-3-1) (data[i] − mean) の 2 乗を vsum へ足し込む。

(5-3-2) i の値を 1 増やす。

(5-4) sigma に $\sqrt{\frac{\texttt{vsum}}{\texttt{num}}}$ を代入。

(6) mean と sigma の値を、それぞれ平均値と標準偏差として出力する。

ポイント (1) 配列 data の大きさ、つまり入力できるデータの個数の限界 1000 は、

```
#define MAX_DATA 1000
```

のようにマクロ定義しておく。 # が行頭にないといけないことに注意。

(2) 配列 data は大域変数でも、main の局所変数でもよい。

(3) C では大きさ N の配列のインデックスは 0 から始まり、 $N-1$ 迄であることに注意する。 たとえば、

```
double data[1000];
```

と double 型の配列 data を定義した場合、利用できる配列の要素は data[0]、data[1]、...、data[999] となる。

(4) 上のアルゴリズム中での繰り返しの処理は、すべて for 文で書くことができる。

(5) 繰り返しを途中で終了するには break 文を使えばよい。

(6) $\sqrt{\frac{\texttt{vsum}}{\texttt{num}}}$ の値は、 C では sqrt(vsum/num) で計算できる。 ただし、ソースファイルの冒頭に

```
#include <math.h>
```

が必要であり、コンパイル (リンク) 時に

```
cc -o prog2-2 prog2-2.c -lm
```

のように、-lm オプションを付けなければならない。

prog2-2.c のプログラム例

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX_DATA        1000

int main ()
{
    double data[MAX_DATA], sum, mean, vsum, sigma;
    int i, num;

    for (num = 0; num < MAX_DATA; num++) {
        printf("%2d番目のデータ => ", num+1);
        if (scanf ("%lf", &data[num]) != 1)
            break;
    }

    if (num == 0)
        exit(EXIT_SUCCESS);

    sum = 0.0;
    for (i = 0; i < num; i++)
        sum += data[i];
    mean = sum/num;

    vsum = 0.0;
    for (i = 0; i < num; i++)
        vsum += (data[i]-mean)*(data[i]-mean);
    sigma = sqrt(vsum/num);

    printf ("平均値 = %.2f, 標準偏差 = %.2f\n", mean, sigma);
    return EXIT_SUCCESS;
}
```

2　選択ソートのプログラミング

配列などに格納された複数のデータを、何らかの基準によって並び替える (あるいは分類する) ことを**整列**、あるいは**ソーティング** (sorting) と呼びます。この節では、最も単純な整列法の 1 つである**選択 (selection) 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。

2.1　選択ソートのアルゴリズム

例えば、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並び替えたいとします[1]。

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、これを配列の先頭要素と交換する。

(2) 次に、先頭要素を除いて (配列中で) 最も小さな値を持つ要素を探し出し、これを配列の 2 番目の要素と交換する。

[1] これを「**昇順**に整列させる」といいます。逆に、より大きいデータが先に来ることを**降順**と呼びます。

(3) さらに、先頭の 2 つの要素を除いて (配列中で) 最も小さな値を持つ要素を探し出し、これを配列の 3 番目の要素と交換する。

(4) 先頭の 3 つの要素を除いて (配列中で) 最も小さな値を持つ要素を探し出し、これを配列の 4 番目の要素と交換する。

以下同様

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 start、pos、i、min、tmp を用意する。

(3) start を 0 で初期化する。

(4) start $<$ num $-$ 1 が成り立っている限り、次の 3 つ作業を順に繰り返す。

(4-1) 配列 data の添字 start 以降での最小要素の添字を pos に代入する。 このためには次の 3 つの手続きを行えばよい。

(4-1-1) min を data[start] で、pos を start でそれぞれ更新する。

(4-1-2) i に start $+$ 1 を代入する。

(4-1-3) i $<$ num が成り立っている限り、次の作業を繰り返す。

(4-1-3-1) もし、 data[i] $<$ min であれば、min を data[i] で、pos を i でそれぞれ更新する。

(4-2) もし、 pos $\neq$ start ならば、data[start] と data[pos] の値を交換する。 つまり、

(4-2-1) tmp へ data[start] の値を退避する。

(4-2-2) data[start] へ data[pos] をコピーする。

(4-2-3) data[pos] へ tmp をコピーする。

(4-3) start の値を 1 増やす。

選択ソートを行う関数 selectsort のプログラム例

```
 1 void selectsort(int data[], int num)
 2 {
 3     int start, pos, i;
 4     int min, tmp;
 5
 6     for (start = 0; start < num-1; start++) {
 7         min = data[start];
 8         pos = start;
 9         for (i = start+1; i < num; i++) {
10             if (data[i] < min) {
11                 min = data[i];
12                 pos = i;
13             }
14         }
15         if (pos != start) {
16             tmp = data[start];
17             data[start] = data[pos];
18             data[pos] = tmp;
19         }
```

```
20      }
21 }
```

3 演習問題

次の 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 prog2-1.c

「1.1 実力チェック」のプログラム prog2-1.c を実際に計算機上で作成、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。

注意: このプログラムを 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 prog2-3.c

5 ページの関数 selectsort を利用して、次の実行例のように、標準入力 (キーボード) から読み込んだ整数データ (1000 個以下) を、小さい順に並び替えて 1 行に表示する C プログラム prog2-3.c を作りなさい。

prog2-3.c の実行例

```
s1542h017% ./prog2-3
 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 => 19
 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 => .
  5,  7, 19, 19, 28, 45, 55, 67, 74, 98
s1542h017%
```

ただし、配列中の整数データを (先頭から順に)1 行に表示するためには、次のような関数 showdata を定義して用いなさい。

関数 showdata の定義

```
void showdata (int data[], int num)
{
    int i;

    if (num > 0) {
        printf("%3d", data[0]);
        for (i = 1; i < num; i++)
            printf(",%3d", data[i]);
    }
```

```
    printf("\n");
}
```

3.3 prog2-4.c

まず、prog2-3.c を prog2-4.c にコピーしなさい。次に、5 ページの関数 selectsort の定義を、6 行目と 7 行目の間で関数 showdata を呼び出すように変更して、次の実行例のように、配列の整列が進む途中の状況が表示されるようにしなさい。

prog2-4.c の実行例

```
s1542h017% ./prog2-4
 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 => 19
 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, 19,  5, 28, 55
  5, 98,  7, 19, 45, 74, 19, 67, 28, 55
  5,  7, 98, 19, 45, 74, 19, 67, 28, 55
  5,  7, 19, 98, 45, 74, 19, 67, 28, 55
  5,  7, 19, 19, 45, 74, 98, 67, 28, 55
  5,  7, 19, 19, 28, 74, 98, 67, 45, 55
  5,  7, 19, 19, 28, 45, 98, 67, 74, 55
  5,  7, 19, 19, 28, 45, 55, 67, 74, 98
  5,  7, 19, 19, 28, 45, 55, 67, 74, 98
  5,  7, 19, 19, 28, 45, 55, 67, 74, 98
s1542h017%
```

余裕のある受講者は、次の実行例のように (関数 selectsort の中で) 配列のどの要素とどの要素を交換しようとしているのかが分かるプログラム prog2-5.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)。

prog2-5.c の実行例

```
s1542h017% ./prog2-5
 1番目のデータ => 67
   :
10番目のデータ => 55
11番目のデータ => .
 67, 98,  7, 19, 45, 74, 19,  5, 28, 55
  ^                          ^
  5, 98,  7, 19, 45, 74, 19, 67, 28, 55
     ^   ^
  5,  7, 98, 19, 45, 74, 19, 67, 28, 55
        ^   ^
  5,  7, 19, 98, 45, 74, 19, 67, 28, 55
              ^                 ^
  5,  7, 19, 19, 45, 74, 98, 67, 28, 55
                    ^                 ^
  5,  7, 19, 19, 28, 74, 98, 67, 45, 55
                         ^                 ^
```

[2]配列中の整数は高々 3 桁だと思って構いません。

```
  5,  7, 19, 19, 28, 45, 98, 67, 74, 55
                          ^           ^
  5,  7, 19, 19, 28, 45, 55, 67, 74, 98
                              ^
  5,  7, 19, 19, 28, 45, 55, 67, 74, 98
                                  ^
  5,  7, 19, 19, 28, 45, 55, 67, 74, 98
s1542h017%
```

プログラミングおよび実習Ⅱ・第 2 回・終り

付録

前回の演習問題 prog1-4.c のプログラム例

```
#include <stdio.h>

int main ()
{
    double min, max, sum, new;
    int num;

    num = 0;
    sum = 0.0;
    while (1) {
        printf("数値を入力してください => ");
        if (scanf("%lf", &new) != 1)
            break;
        if (num == 0 || new < min)
            min = new;
        if (num == 0 || new > max)
            max = new;
        sum += new;
        num ++;
    }
    if (num > 0) {
        printf ("最小値 = %.3f, 平均値 = %.3f, 最大値 = %.3f\n",
                min, sum/num, max);
    }
    return 0;
}
```