プログラミング言語実験・C言語第一回実験レポート

1410149 吉田健太朗

2016年4月18日

1 目的

数値計算の際の情報落ちと線形リストの実装について確認し考察を加える．

2 方法

以下の手順により情報落ちを確認する．

1. データセットをファイルから入力する．
2. 文字列として読み込んだデータをdouble型の数値として変換する．
3. データを格納するコンテナとして配列と線形リストの両方を用意し，それぞれに絶対値についてソート行わないものと行うもので合計4種類のセットを用意する．
4. 4種のデータセットについて総和を取り，その計算の様子を確認する．

データセットは講義サイトに用意してあったものを使用した．

3 ソースコード

実験で用いたソースコード，assignment1-1.c，assignment1-2.cを以下に示す．それぞれ既知，未知の個数のデータの読み取りに対して配列と線形リストを用意し格納，そのデータの総和をとるプログラムとなっている．

ソースコード 1: assignment1-1.c

```
1
2  /*
3   *既知の要素数のデータに対し総和を取る
4   *(a)データを絶対値でソートせずに総和を取る
5   *(b)データを絶対値でソートして総和を取る
6   *2つの結果を比較する
7   */
8
9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <math.h>
12 #define N 21
13
14 int main(void)
15 {
```

```
16    FILE *fp;
17    char *fname = "num.dat";
18    char s[100];
19    double data1[N];
20    double data2[N];
21    int i = 0;
22
23    void print_array(int size, double * array);
24
25    void abssort(double *array);
26
27    double dsum(int size, double *array);
28
29    void check_sum(int size, double* array1, double* array2);
30
31    fp = fopen(fname, "r");
32    //ファイルオープン失敗
33    if(fp == NULL){
34      printf("%s の読み込みに失敗しました\n", fname);
35      return -1;
36    }
37
38    //成功->データ取り込み
39    printf("読み込んだデータ:\n");
40    while(fgets(s, 100, fp) != NULL){
41      data1[i] = atof(s);
42      data2[i] = atof(s);
43      printf("Data[%d]:␣%s", i++, s);
44    }
45    printf("\n");
46    printf("----------------------------------------\n");
47    fclose(fp);
48
49    //未ソートデータ表示
50    printf("ソートされていないデータ:\n");
51      print_array(N, data1);
52    printf("----------------------------------------\n");
53
54    //ソート後のデータ表示
55    abssort(data2);
56    printf("ソートしたデータ:\n");
57    print_array(N, data2);
58    printf("----------------------------------------\n");
59
60    //未ソートで総和を取る
61    double result1 = dsum(N, data1);
62    printf("ソートしない値の和:␣%f\n", result1);
63
64    //絶対値ソートしてから総和を取る
65    double result2 = dsum(N, data2);
66    printf("ソートされた値の和:␣%f\n", result2);
67    printf("----------------------------------------\n");
68
69    //和の様子の比較
70    check_sum(N, data1, data2);
71
72    return 0;
73
74  }
75
76  void print_array(int size, double* array){
77    for(int i = 0; i < size; i++){
78      printf("data[%d]:␣%f\n", i, array[i]);
79    }
80  }
81
82  double dsum(int size, double *array){
83    double result = 0;
```

```
84    for(int i = 0; i < size; i++){
85      result += array[i];
86    }
87    return result;
88  }
89
90  void abssort(double *array){
91    //絶対値でソート
92    for(int i = 0; i<20; i++){
93      for(int j = i+1; j<20; j++){
94        if(fabs(array[i]) > fabs(array[j])){
95          double tmp = array[j];
96          array[j] = array[i];
97          array[i] = tmp;
98        }
99      }
100   }
101 }
102
103 void check_sum(int size, double* array1, double* array2){
104   double sum1, sum2;
105   sum1 = sum2 = 0;
106
107   printf("ソートの有無で総和の経過を比較:\n");
108   for(int i = 0; i < size; i++){
109     sum1 += array1[i];
110     sum2 += array2[i];
111
112     printf("ソート無しデータの和 [%d 番目まで]:␣%f\n", i, sum1);
113     printf("ソート有りデータの和 [%d 番目まで]:␣%f\n", i, sum2);
114     if(i+1 != size){
115     printf("次に和を取る値 (ソート無し):␣%f\n", array1[i+1]);
116     printf("次に和を取る値 (ソート有り):␣%f\n", array2[i+1]);
117     }
118     printf("\n");
119   }
120 }
```

ソースコード 2: assignment1-2.c

```
1
2  /*
3   *未知の要素数のデータに対し総和を取る
4   *(a)データを絶対値でソートせずに総和を取る
5   *(b)データを絶対値でソートして総和を取る
6   *2つの結果を比較する
7   */
8
9
10
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <math.h>
14
15 typedef struct _data{
16   double dnum;
17   struct _data* next;
18 } data;
19
20 typedef struct{
21   data* head;
22   data* crnt;
23 } list;
24
25 void abs_sort(list* list);
26
27 void init_list(list* list);
28
```

```
29 void print_list(list* list);
30
31 double dsum(list list);
32
33 void check_sum(list list1, list list2);
34
35 int main(void)
36 {
37   FILE *fp;
38   char *fname = "num.dat";
39   char s[100];
40   int i = 0;
41   list list1;
42   list list2;
43
44   init_list(&list1);
45   init_list(&list2);
46
47   fp = fopen(fname, "r");
48   //ファイルオープン失敗
49   if(fp == NULL){
50     printf("%s の読み込みに失敗しました\n", fname);
51     return -1;
52   }
53
54   //成功->データ取り込み
55   printf("読み込んだデータ:\n");
56   while(fgets(s, 100, fp) != NULL){
57     data* next1 = (data*)malloc(sizeof(data));
58     data* next2 = (data*)malloc(sizeof(data));
59
60     next1->dnum = atof(s);
61     next2->dnum = atof(s);
62
63     if(list1.head == NULL) {
64       list1.head = next1;
65       list1.crnt = list1.head;
66     } else {
67       list1.crnt->next = next1;
68       list1.crnt = list1.crnt->next;
69     }
70     if(list2.head == NULL) {
71       list2.head = next2;
72       list2.crnt = list2.head;
73     } else {
74       list2.crnt->next = next2;
75       list2.crnt = list2.crnt->next;
76     }
77
78     printf("Data[%d]:␣%s", i++, s);
79   }
80   list1.crnt = list1.head;
81   list2.crnt = list2.head;
82   printf("\n");
83   printf("----------------------------------------\n");
84   fclose(fp);
85
86   //未ソートデータ表示
87   printf("ソートされていないデータ:\n");
88   print_list(&list1);
89   printf("----------------------------------------\n");
90
91   //ソート後のデータ表示
92   abs_sort(&list2);
93   printf("ソートしたデータ:\n");
94   print_list(&list2);
95   printf("----------------------------------------\n");
96
97   //未ソートで総和を取る
```

```
98    double result1 = dsum(list1);
99    printf("ソートしない値の和:␣%f\n", result1);
100
101   //絶対値ソートしてから総和を取る
102   double result2 = dsum(list2);
103   printf("ソートされた値の和:␣%f\n", result2);
104   printf("----------------------------------------\n");
105
106   //和の様子の比較
107   check_sum(list1, list2);
108
109   return 0;
110
111 }
112
113 void init_list(list* list){
114   list->head = NULL;
115   list->crnt = NULL;
116 }
117
118 void print_list(list* list){
119   int i = 0;
120   data* ptr = list->head;
121   while(ptr != NULL){
122     printf("data[%d]:␣%f\n", i++, ptr->dnum);
123     ptr = ptr->next;
124   }
125 }
126
127 double dsum(list list){
128   double result = 0;
129   data* data = list.head;
130   while(data != NULL){
131     result += data->dnum;
132     data = data->next;
133   }
134   return result;
135 }
136
137 void abs_sort(list *list){
138   data* ptr1 = list->head;
139   while(ptr1 != NULL){
140     data* ptr2 = ptr1->next;
141     while(ptr2 != NULL){
142       if(fabs(ptr1->dnum) > fabs(ptr2->dnum)){
143         double tmp = ptr2->dnum;
144         ptr2->dnum = ptr1->dnum;
145         ptr1->dnum = tmp;
146       }
147       ptr2 = ptr2->next;
148     }
149     ptr1 = ptr1->next;
150   }
151 }
152
153 void check_sum(list list1, list list2){
154   double sum1, sum2;
155   sum1 = sum2 = 0;
156   int i = 0;
157   data* data1 = list1.head;
158   data* data2 = list2.head;
159
160   printf("ソートの有無で総和の経過を比較:\n");
161   while(data1 != NULL && data2 != NULL){
162     if(data1 != NULL) sum1 += data1->dnum;
163     if(data2 != NULL) sum2 += data2->dnum;
164
165
166     printf("ソート無しデータの和 [%d 番目まで]:␣%f\n", i, sum1);
```

```
167       printf("ソート有りデータの和 [%d 番目まで]:␣%f\n", i, sum2);
168
169       if(data1->next != NULL && data2->next != NULL){
170         printf("次に和を取る値 (ソート無し):␣%f\n", data1->next->dnum);
171         printf("次に和を取る値 (ソート有り):␣%f\n", data2->next->dnum);
172       }
173
174       printf("\n");
175       i++;
176       data1 = data1->next;
177       data2 = data2->next;
178     }
179 }
```

以上のプログラムにより，次のデータファイルを読み込む．

ソースコード 3: num.dat

```
1  1.0e16
2  −1.0e2
3  23
4  −6.4
5  3.6e2
6  −0.01
7  8.0
8  −70
9  5.0e3
10 1.2e−2
11 −3.0e3
12 46
13 −1.7e3
14 10
15 −5.0e2
16 7.0
17 −2.0e−3
18 0.3
19 −30
20 3.1
21 −1.0e16
```

4 結果

上記のソースコードを実行した結果は以下のようになった．ログファイル名の付番は実行ファイルの付番と対応している．

ソースコード 4: result1-1.txt

```
1  読み込んだデータ
2  :
3  Data[0]: 1.0e16
4  Data[1]: −1.0e2
5  Data[2]: 23
6  Data[3]: −6.4
7  Data[4]: 3.6e2
8  Data[5]: −0.01
9  Data[6]: 8.0
10 Data[7]: −70
11 Data[8]: 5.0e3
12 Data[9]: 1.2e−2
13 Data[10]: −3.0e3
14 Data[11]: 46
15 Data[12]: −1.7e3
16 Data[13]: 10
17 Data[14]: −5.0e2
```

```
Data[15]: 7.0
Data[16]: −2.0e−3
Data[17]: 0.3
Data[18]: −30
Data[19]: 3.1
Data[20]: −1.0e16
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ソートされていないデータ:
data[0]: 10000000000000000.000000
data[1]: −100.000000
data[2]: 23.000000
data[3]: −6.400000
data[4]: 360.000000
data[5]: −0.010000
data[6]: 8.000000
data[7]: −70.000000
data[8]: 5000.000000
data[9]: 0.012000
data[10]: −3000.000000
data[11]: 46.000000
data[12]: −1700.000000
data[13]: 10.000000
data[14]: −500.000000
data[15]: 7.000000
data[16]: −0.002000
data[17]: 0.300000
data[18]: −30.000000
data[19]: 3.100000
data[20]: −10000000000000000.000000
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ソートしたデータ:
data[0]: −0.002000
data[1]: −0.010000
data[2]: 0.012000
data[3]: 0.300000
data[4]: 3.100000
data[5]: −6.400000
data[6]: 7.000000
data[7]: 8.000000
data[8]: 10.000000
data[9]: 23.000000
data[10]: −30.000000
data[11]: 46.000000
data[12]: −70.000000
data[13]: −100.000000
data[14]: 360.000000
data[15]: −500.000000
data[16]: −1700.000000
data[17]: −3000.000000
data[18]: 5000.000000
data[19]: 10000000000000000.000000
data[20]: −10000000000000000.000000
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ソートしない値の和: 54.000000
ソートされた値の和: 52.000000
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ソートの有無で総和の経過を比較:
ソート無しデータの和 [0番目まで]: 10000000000000000.000000
ソート有りデータの和 [0番目まで]: −0.002000
次に和を取る値 (ソート無し): −100.000000
次に和を取る値 (ソート有り): −0.010000

ソート無しデータの和 [1番目まで]: 9999999999999900.000000
ソート有りデータの和 [1番目まで]: −0.012000
次に和を取る値 (ソート無し): 23.000000
次に和を取る値 (ソート有り): 0.012000

```

ソート無しデータの和 [2番目まで]: 9999999999999924.000000
ソート有りデータの和 [2番目まで]: 0.000000
次に和を取る値 (ソート無し): −6.400000
次に和を取る値 (ソート有り): 0.300000

ソート無しデータの和 [3番目まで]: 9999999999999918.000000
ソート有りデータの和 [3番目まで]: 0.300000
次に和を取る値 (ソート無し): 360.000000
次に和を取る値 (ソート有り): 3.100000

ソート無しデータの和 [4番目まで]: 10000000000000278.000000
ソート有りデータの和 [4番目まで]: 3.400000
次に和を取る値 (ソート無し): −0.010000
次に和を取る値 (ソート有り): −6.400000

ソート無しデータの和 [5番目まで]: 10000000000000278.000000
ソート有りデータの和 [5番目まで]: −3.000000
次に和を取る値 (ソート無し): 8.000000
次に和を取る値 (ソート有り): 7.000000

ソート無しデータの和 [6番目まで]: 10000000000000286.000000
ソート有りデータの和 [6番目まで]: 4.000000
次に和を取る値 (ソート無し): −70.000000
次に和を取る値 (ソート有り): 8.000000

ソート無しデータの和 [7番目まで]: 10000000000000216.000000
ソート有りデータの和 [7番目まで]: 12.000000
次に和を取る値 (ソート無し): 5000.000000
次に和を取る値 (ソート有り): 10.000000

ソート無しデータの和 [8番目まで]: 10000000000005216.000000
ソート有りデータの和 [8番目まで]: 22.000000
次に和を取る値 (ソート無し): 0.012000
次に和を取る値 (ソート有り): 23.000000

ソート無しデータの和 [9番目まで]: 10000000000005216.000000
ソート有りデータの和 [9番目まで]: 45.000000
次に和を取る値 (ソート無し): −3000.000000
次に和を取る値 (ソート有り): −30.000000

ソート無しデータの和 [10番目まで]: 10000000000002216.000000
ソート有りデータの和 [10番目まで]: 15.000000
次に和を取る値 (ソート無し): 46.000000
次に和を取る値 (ソート有り): 46.000000

ソート無しデータの和 [11番目まで]: 10000000000002262.000000
ソート有りデータの和 [11番目まで]: 61.000000
次に和を取る値 (ソート無し): −1700.000000
次に和を取る値 (ソート有り): −70.000000

ソート無しデータの和 [12番目まで]: 10000000000000562.000000
ソート有りデータの和 [12番目まで]: −9.000000
次に和を取る値 (ソート無し): 10.000000
次に和を取る値 (ソート有り): −100.000000

ソート無しデータの和 [13番目まで]: 10000000000000572.000000
ソート有りデータの和 [13番目まで]: −109.000000
次に和を取る値 (ソート無し): −500.000000
次に和を取る値 (ソート有り): 360.000000

ソート無しデータの和 [14番目まで]: 10000000000000072.000000
ソート有りデータの和 [14番目まで]: 251.000000
次に和を取る値 (ソート無し): 7.000000
次に和を取る値 (ソート有り): −500.000000

ソート無しデータの和 [15番目まで]: 10000000000000080.000000
ソート有りデータの和 [15番目まで]: −249.000000

```
次に和を取る値 (ソート無し): −0.002000
次に和を取る値 (ソート有り): −1700.000000

ソート無しデータの和 [16番目まで]: 10000000000000080.000000
ソート有りデータの和 [16番目まで]: −1949.000000
次に和を取る値 (ソート無し): 0.300000
次に和を取る値 (ソート有り): −3000.000000

ソート無しデータの和 [17番目まで]: 10000000000000080.000000
ソート有りデータの和 [17番目まで]: −4949.000000
次に和を取る値 (ソート無し): −30.000000
次に和を取る値 (ソート有り): 5000.000000

ソート無しデータの和 [18番目まで]: 10000000000000050.000000
ソート有りデータの和 [18番目まで]: 51.000000
次に和を取る値 (ソート無し): 3.100000
次に和を取る値 (ソート有り): 10000000000000000.000000

ソート無しデータの和 [19番目まで]: 10000000000000054.000000
ソート有りデータの和 [19番目まで]: 10000000000000052.000000
次に和を取る値 (ソート無し): −10000000000000000.000000
次に和を取る値 (ソート有り): −10000000000000000.000000

ソート無しデータの和 [20番目まで]: 54.000000
ソート有りデータの和 [20番目まで]: 52.000000
```

ソースコード 5: result1-2.txt

```
読み込んだデータ
:
Data[0]: 1.0e16
Data[1]: −1.0e2
Data[2]: 23
Data[3]: −6.4
Data[4]: 3.6e2
Data[5]: −0.01
Data[6]: 8.0
Data[7]: −70
Data[8]: 5.0e3
Data[9]: 1.2e−2
Data[10]: −3.0e3
Data[11]: 46
Data[12]: −1.7e3
Data[13]: 10
Data[14]: −5.0e2
Data[15]: 7.0
Data[16]: −2.0e−3
Data[17]: 0.3
Data[18]: −30
Data[19]: 3.1
Data[20]: −1.0e16
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ソートされていないデータ:
data[0]: 10000000000000000.000000
data[1]: −100.000000
data[2]: 23.000000
data[3]: −6.400000
data[4]: 360.000000
data[5]: −0.010000
data[6]: 8.000000
data[7]: −70.000000
data[8]: 5000.000000
data[9]: 0.012000
data[10]: −3000.000000
data[11]: 46.000000
data[12]: −1700.000000
data[13]: 10.000000
```

```
data[14]: −500.000000
data[15]: 7.000000
data[16]: −0.002000
data[17]: 0.300000
data[18]: −30.000000
data[19]: 3.100000
data[20]: −10000000000000000.000000
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ソートしたデータ:
data[0]: −0.002000
data[1]: −0.010000
data[2]: 0.012000
data[3]: 0.300000
data[4]: 3.100000
data[5]: −6.400000
data[6]: 7.000000
data[7]: 8.000000
data[8]: 10.000000
data[9]: 23.000000
data[10]: −30.000000
data[11]: 46.000000
data[12]: −70.000000
data[13]: −100.000000
data[14]: 360.000000
data[15]: −500.000000
data[16]: −1700.000000
data[17]: −3000.000000
data[18]: 5000.000000
data[19]: 10000000000000000.000000
data[20]: −10000000000000000.000000
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ソートしない値の和: 54.000000
ソートされた値の和: 52.000000
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ソートの有無で総和の経過を比較:
ソート無しデータの和 [0番目まで]: 10000000000000000.000000
ソート有りデータの和 [0番目まで]: −0.002000
次に和を取る値 (ソート無し): −100.000000
次に和を取る値 (ソート有り): −0.010000

ソート無しデータの和 [1番目まで]: 9999999999999900.000000
ソート有りデータの和 [1番目まで]: −0.012000
次に和を取る値 (ソート無し): 23.000000
次に和を取る値 (ソート有り): 0.012000

ソート無しデータの和 [2番目まで]: 9999999999999924.000000
ソート有りデータの和 [2番目まで]: 0.000000
次に和を取る値 (ソート無し): −6.400000
次に和を取る値 (ソート有り): 0.300000

ソート無しデータの和 [3番目まで]: 9999999999999918.000000
ソート有りデータの和 [3番目まで]: 0.300000
次に和を取る値 (ソート無し): 360.000000
次に和を取る値 (ソート有り): 3.100000

ソート無しデータの和 [4番目まで]: 10000000000000278.000000
ソート有りデータの和 [4番目まで]: 3.400000
次に和を取る値 (ソート無し): −0.010000
次に和を取る値 (ソート有り): −6.400000

ソート無しデータの和 [5番目まで]: 10000000000000278.000000
ソート有りデータの和 [5番目まで]: −3.000000
次に和を取る値 (ソート無し): 8.000000
次に和を取る値 (ソート有り): 7.000000

ソート無しデータの和 [6番目まで]: 10000000000000286.000000
ソート有りデータの和 [6番目まで]: 4.000000
```

次に和を取る値 (ソート無し): −70.000000
次に和を取る値 (ソート有り): 8.000000

ソート無しデータの和 [7番目まで]: 10000000000000216.000000
ソート有りデータの和 [7番目まで]: 12.000000
次に和を取る値 (ソート無し): 5000.000000
次に和を取る値 (ソート有り): 10.000000

ソート無しデータの和 [8番目まで]: 10000000000005216.000000
ソート有りデータの和 [8番目まで]: 22.000000
次に和を取る値 (ソート無し): 0.012000
次に和を取る値 (ソート有り): 23.000000

ソート無しデータの和 [9番目まで]: 10000000000005216.000000
ソート有りデータの和 [9番目まで]: 45.000000
次に和を取る値 (ソート無し): −3000.000000
次に和を取る値 (ソート有り): −30.000000

ソート無しデータの和 [10番目まで]: 10000000000002216.000000
ソート有りデータの和 [10番目まで]: 15.000000
次に和を取る値 (ソート無し): 46.000000
次に和を取る値 (ソート有り): 46.000000

ソート無しデータの和 [11番目まで]: 10000000000002262.000000
ソート有りデータの和 [11番目まで]: 61.000000
次に和を取る値 (ソート無し): −1700.000000
次に和を取る値 (ソート有り): −70.000000

ソート無しデータの和 [12番目まで]: 10000000000000562.000000
ソート有りデータの和 [12番目まで]: −9.000000
次に和を取る値 (ソート無し): 10.000000
次に和を取る値 (ソート有り): −100.000000

ソート無しデータの和 [13番目まで]: 10000000000000572.000000
ソート有りデータの和 [13番目まで]: −109.000000
次に和を取る値 (ソート無し): −500.000000
次に和を取る値 (ソート有り): 360.000000

ソート無しデータの和 [14番目まで]: 10000000000000072.000000
ソート有りデータの和 [14番目まで]: 251.000000
次に和を取る値 (ソート無し): 7.000000
次に和を取る値 (ソート有り): −500.000000

ソート無しデータの和 [15番目まで]: 10000000000000080.000000
ソート有りデータの和 [15番目まで]: −249.000000
次に和を取る値 (ソート無し): −0.002000
次に和を取る値 (ソート有り): −1700.000000

ソート無しデータの和 [16番目まで]: 10000000000000080.000000
ソート有りデータの和 [16番目まで]: −1949.000000
次に和を取る値 (ソート無し): 0.300000
次に和を取る値 (ソート有り): −3000.000000

ソート無しデータの和 [17番目まで]: 10000000000000080.000000
ソート有りデータの和 [17番目まで]: −4949.000000
次に和を取る値 (ソート無し): −30.000000
次に和を取る値 (ソート有り): 5000.000000

ソート無しデータの和 [18番目まで]: 10000000000000050.000000
ソート有りデータの和 [18番目まで]: 51.000000
次に和を取る値 (ソート無し): 3.100000
次に和を取る値 (ソート有り): 10000000000000000.000000

ソート無しデータの和 [19番目まで]: 10000000000000054.000000
ソート有りデータの和 [19番目まで]: 10000000000000052.000000
次に和を取る値 (ソート無し): −10000000000000000.000000
次に和を取る値 (ソート有り): −10000000000000000.000000

```
174
175 ソート無しデータの和 [20番目まで]: 54.000000
176 ソート有りデータの和 [20番目まで]: 52.000000
```

5 考察

結果について，以下のことが見受けられた．

1. コンテナが配列であるか線形リストであるかは実行結果の差に寄与しない．

2. 絶対値ソートの有無は実行結果の差に寄与する．

ここで．絶対値ソートの有るデータセットと無いデータセットで和を取る様子を比較し，結果に差が生まれる原因を考察する．なお，result1-1.txt と result1-2.txt とでソートの有無による和の様子に変化がないため，result1-1.tex のみで考える．

90 行目のソート無しデータの和 [3 番目まで] を見ると，ここで 10000000000000000.000000 に 23.000000 が加えられた 10000000000000023.000000 が本来表示されるはずであるが，実際には 24.000000 が加わった 10000000000000024.000000 となっている．

また，95 行目のソート無しデータの和 [4 番目まで] を見ると，10000000000000024.000000 から 6.400000 が引かれた 10000000000000017.600000 になるところが 10000000000000018.000000 となっており．計算に誤差が生まれている．

いずれも左から 17 桁目で数値の切り上げによる丸めが行われていることから，C 言語における double 型の有効数字は 16 桁であることが考えられる．

絶対値昇順ソートが行われたデータセットの和は，絶対値が近い数値同士で計算することにより 16 桁を超えるような桁数の急激な増大が避けられ，更に昇順により巨大な数値の計算の後に有効桁数を超えた小さな値の計算が行われることを未然に防ぐことが可能になるため，情報落ちを起こす可能性が低くなると考えられる．

これ以降にも計算の誤差が存在する場面が存在し，その誤差の集積の結果として最後のデータまで足しきった際に 2.000000 の差がソート無しのデータセットと有りのデータセットで生まれている．ソートなしの結果のほうが値が大きいのは，丸めが切り上げによって行われているからである．