プログラミング言語実験・C 言語 第 1 回課題レポート

学籍番号：2210745 **氏名：**LEORA DAVID

2024 年 04 月 17 日

1　課題 1

A. データの個数が既知の場合（あらかじめデータの個数がわかっている場合）で、データとしてリスト構造をもたない、単なる配列を使用したとき。

a. データの入力順に総和を求めたとき。

ソースコード 1　kadai1aa.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    FILE *fptr;
    int n = 21;
    double myArray[21] = {0.0};
    double sum = 0.0;
    fptr = fopen("input2.txt", "r"); // ファイルを読み込む

    for(int i = 0; i < 21; i++){
        fscanf(fptr, "%lf", &myArray[i]);
        sum += myArray[i];
    }

    printf("The numbers are:\n");
    printf("--------------------------\n");
    for(int i = 0; i < n; i++){
        printf("%25lf\n", myArray[i]);
    }
    printf("-------------------------- + \n");
    fclose(fptr);
    printf("Sum: %lf\n", sum); // 合計を出力
    return 0;
}
```

課題 1.A.a の実行結果

入力：input.txt

```
The numbers are:
--------------------------
 10000000000000000.000000
              -100.000000
                23.000000
                -6.400000
               360.000000
                -0.010000
                 8.000000
               -70.000000
              5000.000000
                 0.012000
             -3000.000000
                46.000000
             -1700.000000
                10.000000
              -500.000000
                 7.000000
                -0.002000
                 0.300000
               -30.000000
                 3.100000
-10000000000000000.000000
-------------------------- +
Sum: 54.000000
```

入力：input2.txt

```
The numbers are:
--------------------------
                43.000000
                -0.003000
                 4.200000
-20000000000000000.000000
             2900.000000
          -400000.000000
                31.000000
          2900000.000000
                62.000000
                 1.000000
                -0.000200
                90.000000
```

```
                 1.800000
              -3000.000000
                 14.000000
                  8.400000
                 -0.007000
 10000000000000000.000000
      200000000000.000000
                 99.000000
                 -0.010000
-------------------------- +
Sum: -9999799997499748.000000
```

A. データの個数が既知の場合（あらかじめデータの個数がわかっている場合）で、データとしてリスト構造をもたない、単なる配列を使用したとき。

b. データ入力後、絶対値に関して昇順に並べ替えたあとで総和を求めたとき。

ソースコード 2　kadai1ab.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <math.h>
5
6  // ２つの要素を入れ替える関数
7  void swap(double *a, double *b){
8      double t = *a;
9      *a = *b;
10     *b = t;
11 }
12
13 // クイックソートのパーティション関数
14 int partition(double arr[], int low, int high){
15     double pivot = fabs(arr[high]);
16     int i = (low - 1);
17
18     for (int j = low; j < high; j++){
19         if (fabs(arr[j]) < fabs(pivot)){
20             i++;
21             swap(&arr[i], &arr[j]);
22         }
23     }
24     swap(&arr[i+1], &arr[high]);
25     return (i+1);
26 }
27
28 // クイックソート関数
29 void quickSort(double arr[], int low, int high){
30     if (low < high){
31         int pi = partition(arr, low, high);
32         quickSort(arr, low, pi-1);
33         quickSort(arr, pi+1, high);
34     }
35 }
36
37 int main(){
38     FILE *fptr;
39     int n = 21;
40     double myArray[21] = {0.0};
41     double sum = 0.0;
```

```
    fptr = fopen("input2.txt", "r"); // ファイルを読み込む

    for(int i = 0; i < 21; i++){
        fscanf(fptr, "%lf", &myArray[i]); // ファイルから数字を読み込む
    }

    quickSort(myArray, 0, n-1);

    printf("The numbers are:\n");
    printf("--------------------------\n");
    for(int i = 0; i < n; i++){
        printf("%25lf\n", myArray[i]); // ソートされた数字を出力
        sum += myArray[i];
    }
    printf("-------------------------- + \n");
    fclose(fptr);
    printf("Sum: %lf\n", sum); // 合計を出力
    return 0;
}
```

課題 1.A.b の実行結果

入力：input.txt

```
The numbers are:
--------------------------
                -0.002000
                -0.010000
                 0.012000
                 0.300000
                 3.100000
                -6.400000
                 7.000000
                 8.000000
                10.000000
                23.000000
               -30.000000
                46.000000
               -70.000000
              -100.000000
               360.000000
              -500.000000
             -1700.000000
             -3000.000000
              5000.000000
-10000000000000000.000000
```

```
 10000000000000000.000000
-------------------------- +
Sum: 52.000000
```

入力：input2.txt

```
The numbers are:
--------------------------
               -0.000200
               -0.003000
               -0.007000
               -0.010000
                1.000000
                1.800000
                4.200000
                8.400000
               14.000000
               31.000000
               43.000000
               62.000000
               90.000000
               99.000000
             2900.000000
            -3000.000000
          -400000.000000
          2900000.000000
      200000000000.000000
 10000000000000000.000000
-20000000000000000.000000
-------------------------- +
Sum: -9999799997499746.000000
```

B. データの個数が未知の場合（データの個数がわからない場合）で、データとしてポインタによる線形リスト（自己参照型のデータ構造）を使用したとき。

a. データの入力順に総和を求めたとき。

ソースコード 3　kadai1ba.c

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

typedef double data_type;

// ノードの構造体
typedef struct node_tag {
    data_type data; // ノードに格納するデータ
    struct node_tag *next; // 次のノードへのポインタ
} node_type;

// リストの要素を出力する関数
void printList(node_type *head){
    node_type *current = head;
    while(current != NULL){
        printf("%lf", current->data);
        if (current -> next != NULL)
            printf(" -> "); // 最後の要素以外は矢印を表示
        else
            printf("\n"); // 最後の要素は改行
        current = current->next; // 次のノードへ移動
    }
}

int main(){
    FILE *fptr;
    double sum = 0.0;

    node_type *head = NULL;
    fptr = fopen("input2.txt", "r"); // ファイルを読み込む

    data_type num;
    while(fscanf(fptr, "%lf", &num) != EOF){ // ファイルから数字を読み込む
        node_type *new_node = (node_type *)malloc(sizeof(node_type)); // 新しいノードを
            作成
        new_node->data = num; // ノードに数字を格納
        new_node->next = head; // 新しいノードの次のノードをhead に設定
        head = new_node; // head を新しいノードに設定
        sum += num; // 合計を計算
```

```
41      }
42
43      printList(head); // リストの要素を出力
44      fclose(fptr);
45      printf("The sum of the numbers is: %lf\n", sum); // 合計を出力
46      return 0;
47  }
```

課題 1.B.a の実行結果

入力：input.txt

```
-10000000000000000.000000 -> 3.100000 -> -30.000000 -> 0.300000 -> -0.002000 ->
    7.000000 -> -500.000000 -> 10.000000 -> -1700.000000 -> 46.000000 ->
    -3000.000000 -> 0.012000 -> 5000.000000 -> -70.000000 -> 8.000000 ->
    -0.010000 -> 360.000000 -> -6.400000 -> 23.000000 -> -100.000000 ->
    10000000000000000.000000
The sum of the numbers is: 54.000000
```

入力：input2.txt

```
-0.010000 -> 99.000000 -> 200000000000.000000 -> 10000000000000000.000000 ->
    -0.007000 -> 8.400000 -> 14.000000 -> -3000.000000 -> 1.800000 -> 90.000000
    -> -0.000200 -> 1.000000 -> 62.000000 -> 2900000.000000 -> 31.000000 ->
    -400000.000000 -> 2900.000000 -> -20000000000000000.000000 -> 4.200000 ->
    -0.003000 -> 43.000000
The sum of the numbers is: -9999799997499748.000000
```

B. データの個数が未知の場合（データの個数がわからない場合）で、データとしてポインタによる線形リスト（自己参照型のデータ構造）を使用したとき。

b. データ入力後、絶対値に関して昇順に並べ替えたあとで総和を求めたとき。

ソースコード 4　kadai1bb.c

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

typedef double data_type;

// ノードの構造体
typedef struct node_tag {
    data_type data;
    struct node_tag *next;
} node_type;

// リストの要素を出力する関数
void printList(node_type *head, double sum){
    node_type *current = head;
    while(current != NULL){
        sum += current->data; // 合計を計算
        printf("%lf", current->data); // ノードのデータを出力
        if (current -> next != NULL)
            printf(" -> "); // 最後の要素以外は矢印を表示
        else
            printf("\n"); // 最後の要素は改行
        current = current->next; // 次のノードへ移動
    }
    printf("The sum of the numbers is: %lf\n", sum); // 合計を出力
}

// リストの要素を検索する関数
int search_position(data_type x, node_type *p){
    int i = 0;
    while((p != NULL) && (fabs(x) > fabs(p->data))){
        p = p->next;
        i++;
    }
    return i;
}

int main(){
    FILE *fptr;
    int pos;
```

```
42      double sum = 0.0;
43
44      node_type *head = NULL;
45      fptr = fopen("input2.txt", "r");
46
47      data_type num;
48      while(fscanf(fptr, "%lf", &num) != EOF){ // ファイルから数字を読み込む
49          node_type *new_node = (node_type *)malloc(sizeof(node_type)); // 新しいノードを
                作成
50          new_node->data = num; // ノードに数字を格納
51          new_node->next = NULL; // 新しいノードの次のノードをNULL に設定
52
53          pos = search_position(num, head); // ノードの位置を検索
54
55          if (pos == 0 || head == NULL){ // ノードが先頭に追加される場合
56              new_node->next = head; // 新しいノードの次のノードをhead に設定
57              head = new_node; // head を新しいノードに設定
58          } else {
59              node_type *current = head; // ノードの位置を検索
60              for(int i=0; i < pos-1 && current->next != NULL; i++){ // ノードの位置を検
                    索
61                  current = current->next;
62              }
63              new_node->next = current->next; // 新しいノードの次のノードを
                    current の次のノードに設定
64              current->next = new_node; // current の次のノードを新しいノードに設定
65          }
66      }
67      printList(head, sum); // リストの要素を出力
68      fclose(fptr);
69      return 0;
70  }
```

課題 1.B.b の実行結果

入力：input.txt

```
1  -0.002000 -> -0.010000 -> 0.012000 -> 0.300000 -> 3.100000 -> -6.400000 ->
       7.000000 -> 8.000000 -> 10.000000 -> 23.000000 -> -30.000000 -> 46.000000 ->
       -70.000000 -> -100.000000 -> 360.000000 -> -500.000000 -> -1700.000000 ->
       -3000.000000 -> 5000.000000 -> -10000000000000000.000000 ->
       10000000000000000.000000
2  The sum of the numbers is: 52.000000
```

入力：input2.txt

```
1 -0.000200 -> -0.003000 -> -0.007000 -> -0.010000 -> 1.000000 -> 1.800000 ->
      4.200000 -> 8.400000 -> 14.000000 -> 31.000000 -> 43.000000 -> 62.000000 ->
      90.000000 -> 99.000000 -> 2900.000000 -> -3000.000000 -> -400000.000000 ->
      2900000.000000 -> 200000000000.000000 -> 10000000000000000.000000 ->
      -20000000000000000.000000
2 The sum of the numbers is: -9999799997499746.000000
```

課題１の考察

この結果から、配列と線形リストを使用してデータの総和を求めてみた。データを入力順に加算すると、合計は 54 になった。しかし、データを絶対値に基づいて昇順に並べ替えてから総和を求めると、合計は 52 になった。実際の正解は 51 である。第二の例では、データを入力順に加算した結果、合計は -9999799997499748 だった。一方、データを絶対値に基づいて昇順に並べ替えた後の総和は -9999799997499746 になった。しかし、実際の正しい結果は -9999799997499750 である。この違いは、絶対値の大きい数値と小さい数値が混在しているため、浮動小数点演算の限界により情報落ちが生じたと考えらる。

2　課題２

以下の条件を満たす、Ｃのプログラムを作成した。

● データの個数が未知で、各データが２つの文字列変数 firstname、lastname からなるデータセットがある。

● これらの文字列変数をメンバーとしてもつ自己参照型の構造体を定義して、ポインタによる線形リストを実現し、データ入力時に線形リストの各要素が lastname の辞書順となるようにデータを挿入する。データ入力終了後、この線形リストの全ての要素の内容を先頭から順に表示する。

● 続いて、もう１つの線形リストを用意して、 上記の lastname の辞書順にデータが並べられた線形リストからデータを取り出し、今度は firstname の辞書順となるようにデータを挿入する。その後、その線形リストの全ての要素の内容を先頭から順に表示する。

ソースコード 5　kadai2.c

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

typedef char firstName[50];
typedef char lastName[50];

// ノードの構造体
typedef struct name_tag {
    firstName firstName; // ノードに格納するFirstName
    lastName lastName; // ノードに格納するLastName
    struct name_tag *next; // 次のノードへのポインタ
} name_tag;

int main(){
    FILE *fptr;
    fptr = fopen("input2.txt", "r");
    if (fptr == NULL) {
        printf("Unable to open file\n");
        return -1;
    }

    name_tag *head = NULL, *temp = NULL;
    firstName first;
    lastName last;

    while(fscanf(fptr, "%49s %49s", first, last) == 2){ // ファイルから名前を読み込む
        name_tag *newNode = (name_tag *)malloc(sizeof(name_tag)); // 新しいノードを作成
        if (newNode == NULL) {
            printf("Unable to allocate memory\n");
            return -1;
```

```
33          }
34          strcpy(newNode->firstName, first); // ノードにfirstName を格納
35          strcpy(newNode->lastName, last); // ノードにlastName を格納
36          newNode->next = NULL; // 新しいノードの次のノードをNULL に設定
37
38          if(head == NULL || strcmp(newNode->lastName, head->lastName) < 0){ // ノードが
                先頭に追加される場合
39              newNode->next = head; // 新しいノードの次のノードをhead に設定
40              head = newNode; // head を新しいノードに設定
41          } else { // ノードが先頭以外に追加される場合
42              temp = head;
43              while(temp->next != NULL && strcmp(newNode->lastName, temp->next->lastName)
                     > 0){ // ノードの位置を検索
44                  temp = temp->next; // 次のノードへ移動
45              }
46              newNode->next = temp->next; // 新しいノードの次のノードをtemp->next に設定
47              temp->next = newNode; // temp の次のノードを新しいノードに設定
48          }
49      }
50      fclose(fptr);
51      printf("線形リストの各要素が Last Name の辞書順を先頭から順に出力する\n");
52      printf("-------------------------------------------------------\n");
53      temp = head;
54      while(temp != NULL){ // リストの要素を出力
55          printf("苗字: %10s, 名前: %10s\n", temp->lastName, temp->firstName);
56          temp = temp->next;
57      }
58      printf("\n");
59
60      name_tag *headOfFirstName = NULL, *temp2 = NULL; // First Name の辞書順に並び替える
            ためのリスト
61
62      temp = head;
63      while (temp != NULL) { // First Name の辞書順に並び替える
64          name_tag *newNode = (name_tag *)malloc(sizeof(name_tag));
65          if (newNode == NULL) {
66              printf("Unable to allocate memory\n");
67              return -1;
68          }
69          strcpy(newNode->firstName, temp->firstName); // ノードにfirstName を格納
70          strcpy(newNode->lastName, temp->lastName); // ノードにlastName を格納
71          newNode->next = NULL;
72          if(headOfFirstName == NULL || strcmp(newNode->firstName, headOfFirstName->
                firstName) < 0){ // ノードが先頭に追加される場合
73              newNode->next = headOfFirstName;
74              headOfFirstName = newNode;
```

```
75          } else { // ノードが先頭以外に追加される場合
76              temp2 = headOfFirstName;
77              while (temp2->next != NULL && strcmp(newNode->firstName, temp2->next->
                    firstName) > 0){
78                  temp2 = temp2->next;
79              }
80              newNode->next = temp2->next;
81              temp2->next = newNode;
82          }
83          temp = temp->next;
84      }
85
86      printf("線形リストの各要素が First Name の辞書順を先頭から順に出力する\n"); // First
            Name の辞書順に並び替えたリストを出力
87      printf("-------------------------------------------------------\n");
88      temp2 = headOfFirstName;
89      while(temp2 != NULL) { // リストの要素を出力
90          printf("苗字: %10s, 名前: %10s\n", temp2->lastName, temp2->firstName);
91          temp2 = temp2->next;
92      }
93
94      while (head != NULL) { // メモリの解放
95          temp = head;
96          head = head->next;
97          free(temp);
98      }
99
100     while(headOfFirstName != NULL){ // メモリの解放
101         temp2 = headOfFirstName;
102         headOfFirstName = headOfFirstName->next;
103         free(temp2);
104     }
105
106     return 0;
107 }
```

課題２の実行結果

入力：input.txt

```
線形リストの各要素が Last Name の辞書順を先頭から順に出力する
-------------------------------------------------------
苗字: Chiba, 名前: Mio
苗字: Inoue, 名前: Asuka
苗字: Ishikawa, 名前: Masaki
苗字: Ito, 名前: Yuuto
苗字: Kato, 名前: Nana
苗字: Kimura, 名前: Riku
苗字: Kobayashi, 名前: Kaito
苗字: Matsumoto, 名前: Sakura
苗字: Mori, 名前: Hana
苗字: Nakajima, 名前: Takeru
苗字: Nakamura, 名前: Yui
苗字: Saito, 名前: Daiki
苗字: Sasaki, 名前: Ren
苗字: Suzuki, 名前: Aoi
苗字: Takahashi, 名前: Hina
苗字: Tanaka, 名前: Haruto
苗字: Ueda, 名前: Yuna
苗字: Watanabe, 名前: Mei
苗字: Yamada, 名前: Hiro
苗字: Yamamoto, 名前: Souta

線形リストの各要素が First Name の辞書順を先頭から順に出力する
-------------------------------------------------------
苗字: Suzuki, 名前: Aoi
苗字: Inoue, 名前: Asuka
苗字: Saito, 名前: Daiki
苗字: Mori, 名前: Hana
苗字: Tanaka, 名前: Haruto
苗字: Takahashi, 名前: Hina
苗字: Yamada, 名前: Hiro
苗字: Kobayashi, 名前: Kaito
苗字: Ishikawa, 名前: Masaki
苗字: Watanabe, 名前: Mei
苗字: Chiba, 名前: Mio
苗字: Kato, 名前: Nana
苗字: Sasaki, 名前: Ren
苗字: Kimura, 名前: Riku
苗字: Matsumoto, 名前: Sakura
苗字: Yamamoto, 名前: Souta
```

```
苗字: Nakajima, 名前: Takeru
苗字: Nakamura, 名前: Yui
苗字: Ueda, 名前: Yuna
苗字: Ito, 名前: Yuuto
```

入力：input2.txt

```
線形リストの各要素が Last Name の辞書順を先頭から順に出力する
------------------------------------------------------------
苗字: Aldridge, 名前: Benjamin
苗字: Baxter, 名前: Yasmin
苗字: Carmichael, 名前: David
苗字: Dalton, 名前: William
苗字: Everhart, 名前: Finn
苗字: Fletcher, 名前: Uma
苗字: Gallagher, 名前: Henry
苗字: Hawthorne, 名前: Sophia
苗字: Irving, 名前: Jack
苗字: Jennings, 名前: Quinn
苗字: Kensington, 名前: Liam
苗字: Lancaster, 名前: Olivia
苗字: Maddox, 名前: Noah
苗字: Norwood, 名前: Mia
苗字: O'Connell, 名前: Parker
苗字: Paston, 名前: Kylie
苗字: Quincy, 名前: Ryan
苗字: Rutherford, 名前: Isabella
苗字: Sterling, 名前: Thomas
苗字: Thatcher, 名前: Grace
苗字: Underwood, 名前: Violet
苗字: Vaughn, 名前: Ella
苗字: Whitemore, 名前: Xander
苗字: Xavier, 名前: Charlotte
苗字: Yardley, 名前: Zoe
苗字: Zimmerman, 名前: Amelia

線形リストの各要素が First Name の辞書順を先頭から順に出力する
------------------------------------------------------------
苗字: Zimmerman, 名前: Amelia
苗字: Aldridge, 名前: Benjamin
苗字: Xavier, 名前: Charlotte
苗字: Carmichael, 名前: David
苗字: Vaughn, 名前: Ella
苗字: Everhart, 名前: Finn
苗字: Thatcher, 名前: Grace
苗字: Gallagher, 名前: Henry
苗字: Rutherford, 名前: Isabella
```

41 **苗字**: Irving, **名前**: Jack
42 **苗字**: Paston, **名前**: Kylie
43 **苗字**: Kensington, **名前**: Liam
44 **苗字**: Norwood, **名前**: Mia
45 **苗字**: Maddox, **名前**: Noah
46 **苗字**: Lancaster, **名前**: Olivia
47 **苗字**: O'Connell, **名前**: Parker
48 **苗字**: Jennings, **名前**: Quinn
49 **苗字**: Quincy, **名前**: Ryan
50 **苗字**: Hawthorne, **名前**: Sophia
51 **苗字**: Sterling, **名前**: Thomas
52 **苗字**: Fletcher, **名前**: Uma
53 **苗字**: Underwood, **名前**: Violet
54 **苗字**: Dalton, **名前**: William
55 **苗字**: Whitemore, **名前**: Xander
56 **苗字**: Baxter, **名前**: Yasmin
57 **苗字**: Yardley, **名前**: Zoe
