

# 北京交通大学

## 《数据结构（A）》第2章作业

专    业： 计算机科学与技术

班    级：                     

学生姓名：                     

学    号：                     

北京交通大学计算机与信息技术学院

2021 年 9 月 23 日



## 《数据结构 (A)》第 2 章作业<sup>①</sup>

每位学生都应该创建一个 `project` 来存放你的（这次作业）程序。如果数据结构定义与已实现的 `project` 相同，则一定的在已有的 `project` 之下实现，并要求：

- （1）单独书写新的一个 `.h` 文件；
- （2）一个题目程序书写成一个新的 `.cpp` 文件；
- （3）修改（也可以认为是）`main` 文件。

如果没有定义好数据结构，则要求重新建立一个新的 `project`，以上 3 个文件都需要自己写。

学生作业提交 `.docx` 文件：包括题目、思路、代码与注释、以及测试情况及其分析等（格式按模板要求）。

### 1 基本作业题目

2.1 （《数据结构题集（C 语言版）》，第 2 章，第 2.19 题）已知线性表中的元素以值递增有序排列，并以单链表作存储结构。试写一高效的算法，删除表中所有值大于 `mink` 且小于 `maxk` 的元素（若表中存在这样的元素），同时释放被删结点空间，并分析你的算法的时间复杂度（注意：`mink` 和 `maxk` 是给定的两个参变量，他们的值可以和表中相同，也可以不同）。

2.2 （《数据结构题集（C 语言版）》，第 2 章，第 2.21 题）试写一个算法，实现顺序表的就地逆置，即利用原表存储空间，将线性表  $(a_1, a_2, \dots, a_n)$  逆置为  $(a_n, a_{n-1}, \dots, a_2, a_1)$ 。

---

<sup>①</sup> 这是《数据结构 (A)》第 2 章的基本作业，最晚提交日期是 2021 年 09 月 26 日。

2.3 （《数据结构题集（C语言版）》，第2章，第2.22题）试写一个算法，对单链表实现就地逆置。

注：没有特别说明，链表均带头结点。

2.4 （《数据结构题集（C语言版）》，第2章，第2.38题）设有一个双向循环链表，每个结点中除有 `prior`、`data` 和 `next` 三个域外，还增设了一个访问频度域 `freq`。在链表被起用之前，频度域 `freq` 的值均初始化为零，而每当对链表进行一次 `locate(L,x)` 的操作后，被访问的结点（即元素值等于 `x` 的结点）中的频度域 `freq` 的值便增 1，同时调整链表中结点之间的次序，使其按访问频度非递增的次序顺序排列，以便始终保持被频繁访问的结点总是靠近表头结点。试编写符合上述要求的 `locate` 操作的算法。

## 2 基本作业题目解答

### 2.1

题目：（《数据结构题集（C语言版）》，第2章，第2.19题）已知线性表中的元素以值递增有序排列，并以单链表作存储结构。试写一高效的算法，删除表中所有值大于 `mink` 且小于 `maxk` 的元素（若表中存在这样的元素），同时释放被删结点空间，并分析你的算法的时间复杂度（注意：`mink` 和 `maxk` 是给定的两个参变量，他们的值可以和表中相同，也可以不同）。

思路：先在头文件中声明单链表，再在 `cpp` 文件中制作算法。算法分为三部分。首先创建一个链表，是为函数 `CreatLink`，由交互窗口输入结点的 `data`，然后输出输入内容，这里是 `Display_LinkList` 函数。`Display` 函数以 -1 作为结束标志符，最后创建 `Delete` 函数。

`Main` 函数中，先键入原单链表 `data` 信息，这里要求是升序的 `int` 型数据。因为 `chapter2` 所有题目为一个 `project`，所以在 `main` 中是以 `switch` 进行选择功能的，其中标志符为字符 1。再显示出来键入内容，接着输入 `mink` 和 `maxk` 值，再调用 `Delete` 函数进行处理，最后显示出修改后的 `data` 值。

Delete 函数具体，有三个参数，利用链表原本的升序和三个局部指针变量进行处理，利用不等式条件，先连接节点，再 free 地址。

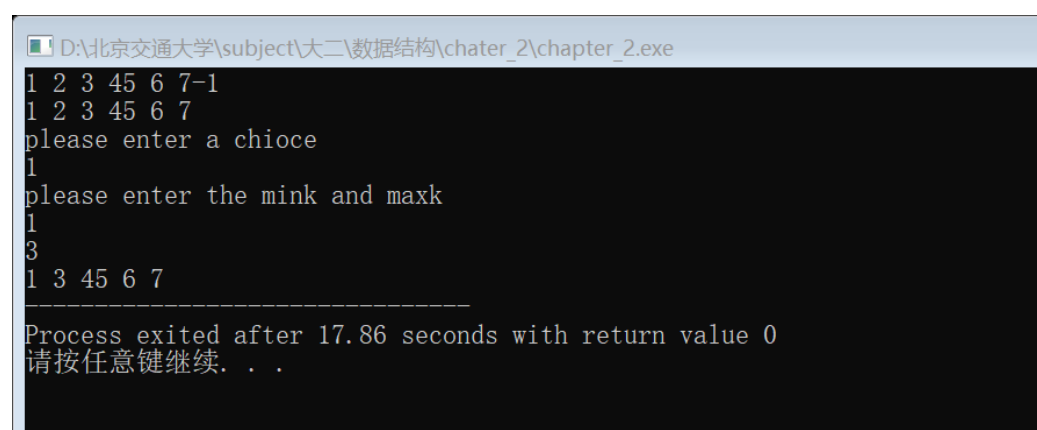
代码：

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4.
5.
6. struct LinkList {
7.     int data;
8.     struct LinkList *next;
9. };
10. //删除链表中小于maxk 并且大于mink的结点
11. int Delete_LinkList(struct LinkList *list, int mink, int maxk){
12.     struct LinkList*ptr1, *ptr2, *ptr3= NULL;
13.     ptr1 = list;
14.     ptr2 = ptr1;
15.     ptr1 = ptr1->next;
16.     while(ptr1!=NULL && ptr1->data < maxk){ //利用有序列，简化判断
17.         if(ptr1->data <= mink){
18.             ptr2 = ptr1;
19.             ptr1 = ptr1->next;
20.         }
21.         else{
22.             ptr2->next = ptr1->next;
23.             ptr3 = ptr1;
24.             ptr1 = ptr1->next;
25.             free(ptr3);
26.         }
27.     }
28.     return 1;
29. }//
30. 创建一个链表
31. struct LinkList*CreatLink(struct LinkList* head) {
32.     struct LinkList *p=NULL, *q=NULL;
33.     int data;
34.     p = (struct LinkList*)malloc(sizeof(struct LinkList));
35.     head = p;
36.     q = p;
37.     scanf("%d",&data);
```

```
38.     while(data != -1) {
39.         p = (struct LinkList*)malloc(sizeof(struct LinkList));
40.         p->data = data;
41.         q->next = p;
42.         q = p;
43.         scanf("%d",&data);
44.     }
45.     p->next = NULL;
46.     return head;
47. }
48.
49.
50. //显示链表data
51. void Display_LinkList(struct LinkList *head) {
52.     struct LinkList *h = head->next;//跳过头节点
53.     if (h == NULL) { //判空
54.         printf("The Link is null");
55.         return;
56.     }
57.     while (h != NULL) {
58.         printf("%d ", h->data);
59.         h = h->next;
60.     }
61. }
62.
63. int main(){
64.     struct LinkList*head=NULL;
65.     struct LinkList*linklist = CreatLink(head);
66.     Display_LinkList(linklist);
67.     printf("\n");
68.
69.     int mink, maxk;
70.     char chioce = '\0';
71.
72.     printf("please enter a chioce\n");
73.     scanf( "%s", &chioce);
74.
75.     switch (chioce) {
76.     case '1':
77.         printf("please enter the mink and maxk\n");
78.         scanf( "%d", &mink);
79.         scanf("%d", &maxk);
80.         Delete_LinkList(linklist, mink, maxk);
```

```
81.      Display_LinkList(linklist);
82.      break;
83.
84.
85.
```

测试情况:



```
D:\北京交通大学\subject\大二\数据结构\chater_2\chapter_2.exe
1 2 3 4 5 6 7-1
1 2 3 4 5 6 7
please enter a chioce
1
please enter the mink and maxk
1
3
1 3 4 5 6 7
-----
Process exited after 17.86 seconds with return value 0
请按任意键继续. . .
```

2.1

分析:

时间复杂度为  $O(n)$

## 2.2

题目:

(《数据结构题集 (C 语言版)》, 第 2 章, 第 2.21 题) 试写一个算法, 实现顺序表的就地逆置, 即利用原表存储空间, 将线性表  $(a_1, a_2, \dots, a_n)$  逆置为  $(a_n, a_{n-1}, \dots, a_2, a_1)$ 。

思路:

首先由交互窗口键入一个数组即顺序表, 再调用 `Oppose_List` 函数进行就

地转置。其中，`Oppose_List` 函数有两个参数，一个是数组，第二个数组长度。凭借已知的数组长度，进行对偶交换位置，即可装置。

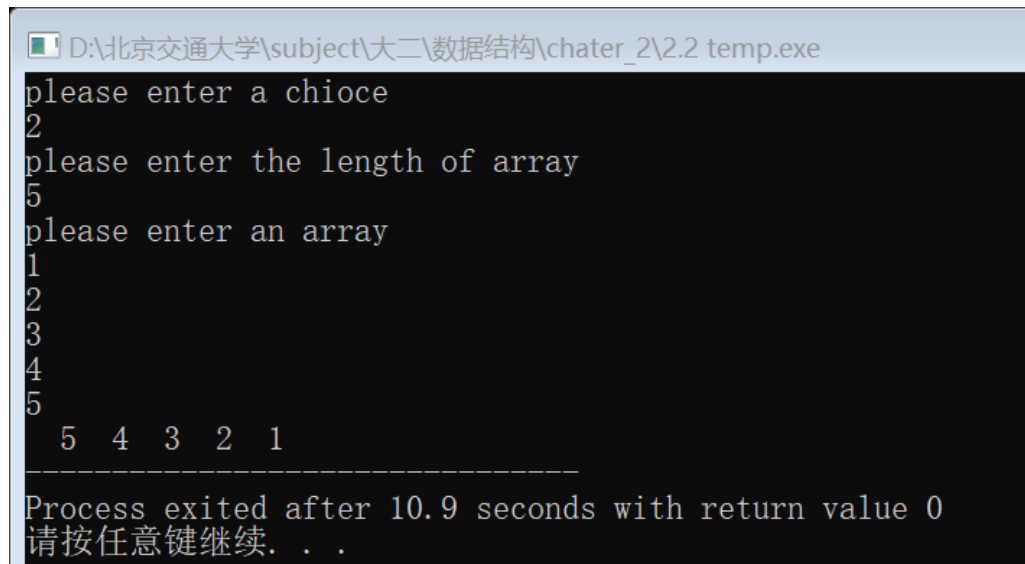
代码：

```
1. int Oppose_List(int *list, int length ) {
2.     int tem;
3.     for (int i = 0; i < length / 2; i++) {
4.         tem = list[i];
5.         list[i] = list[length - i - 1];
6.         list[length - i - 1] = tem;
7.     }
8.
9.     return 1;
10. }
11. int main() {
12. //  struct LinkList*head=NULL;
13. //  struct LinkList*linklist = CreatLink(head);
14. //  Display_LinkList(linklist);
15. //  printf("\n");
16. //
17. //  int mink, maxk;
18.
19.
20.     int length;
21.
22.     char chioce = '\0';
23.
24.     printf("please enter a chioce\n");
25.     scanf( "%s", &chioce);
26.
27.     switch (chioce) {
28. //  case '1':
29. //      printf("please enter the mink and maxk\n");
30. //      scanf( "%d", &mink);
31. //      scanf("%d", &maxk);
32. //      Delete_LinkList(linklist, mink, maxk);
33. //      Display_LinkList(linklist);
34. //      break;
35.     case '2':
36.         printf("please enter the length of array\n");
37.         scanf("%d", &length);
```



```
38.         int list[length];
39.         printf("please enter an array\n");
40.         for (int i = 0; i < length; i++) {
41.             scanf("%d", &list[i]);
42.         }
43.         Oppose_List(list, length);
44.         for (int j = 0; j < length; j ++){
45.             printf("%3d", list[j]);
46.         }
47.
48.         break;
49.     }
50. }
```

调试:



```
D:\北京交通大学\subject\大二\数据结构\chater_2\2.2 temp.exe
please enter a chioce
2
please enter the length of array
5
please enter an array
1
2
3
4
5
5 4 3 2 1
-----
Process exited after 10.9 seconds with return value 0
请按任意键继续. . .
```

### 2.3

题目：

（《数据结构题集（C语言版）》，第2章，第2.22题）试写一个算法，对单链表实现就地逆置。

思路：

接2.1题，由相同函数创建出一个单链表，再显示出来，调用 `Oppose_LinkList` 函数进行就地转置。其中，`Oppose_LinkList` 函数，仅仅是把成员 `next` 指针逆转，并没有改变物理位置。`Oppose_LinkList` 函数返回值为 `LinkList` 类型指针，即返回原链表最后一个结点的地址，此时它变为第一个结点。`Head` 变为最后一个结点，`head` 指向 `NULL`，但是输出控制条件是 `!=NULL`，则会输出 `head` 原来的 `data` 值，此值随机，所谓，可以把原链表的首结点 `next` 指向 `NULL`。完美输出。

代码：

```
1. #include "chapter2.h"
2. //仍然利用2.1中的Create函数和diplay函数
3. struct LinkList *CreatLink(struct LinkList *head) {
4.     struct LinkList *p = NULL, *q = NULL;
5.     int data;
6.     p = (struct LinkList *)malloc(sizeof(struct LinkList));
7.     head = p;
8.     q = p;
9.     scanf("%d", &data);
10.    while (data != -1) {
11.        p = (struct LinkList *)malloc(sizeof(struct LinkList));
12.        p->data = data;
13.        q->next = p;
```

```
14.         q = p;
15.         scanf("%d", &data);
16.     }
17.     p->next = NULL;
18.     return head;
19. }
20.
21. //显示链表data
22. void Display_LinkList(struct LinkList *head) {
23.     struct LinkList *h = head->next;//跳过头节点
24.     if (h == NULL) { //判空
25.         printf("The Link is null");
26.         return;
27.     }
28.     while (h != NULL) {
29.         printf("%d ", h->data);
30.         h = h->next;
31.     }
32. }
33. //单链表的就地转置
34. struct LinkList *Oppose_LinkList(struct LinkList *list) {
35.     struct LinkList *ptr1, *ptr2, *ptr3;
36.     ptr1 = list;
37.     ptr3 = ptr2 = NULL;
38.
39.     ptr2 = ptr1->next;
40.     ptr1->next = ptr3;
41.     ptr3 = ptr1;
42.     ptr1 = ptr2;
43.     ptr3 = NULL;
44. // ptr3 重新赋值为NULL 可避免输出head的值
45.
46.     while (ptr1) {
47.         ptr2 = ptr1->next;
48.         ptr1->next = ptr3;
49.         ptr3 = ptr1;
50.         ptr1 = ptr2;
51.     }
52.
53.     return ptr3;
54.
55. }
56. int main() {
```

```
57.     struct LinkList *head = NULL;
58.     struct LinkList *linklist = CreatLink(head);
59.     Display_LinkList(linklist);
60.     printf("\n");
61.     char chioce = '\0';
62.
63.     printf("please enter a chioce\n");
64.     scanf( "%s", &chioce);
65.
66.     switch (chioce) {
67. // case '1':
68. //     printf("please enter the mink and maxk\n");
69. //     scanf( "%d", &mink);
70. //     scanf("%d", &maxk);
71. //     Delete_LinkList(linklist, mink, maxk);
72. //     Display_LinkList(linklist);
73. //     break;
74. // case '2':
75. //     printf("please enter the length of array\n");
76. //     scanf("%d", &length);
77. //     int list[length];
78. //     printf("please enter an array\n");
79. //     for(int i=0; i<length; i++){
80. //         scanf("%d", list[i]);
81. //     }
82. //     Oppose_List(list, length);
83. //     for(int j=0; j<length; j++){
84. //         printf("%d\t", list[j]);
85. //     }
86. //
87. //     break;
88.     case '3':
89.         struct LinkList *tem = Oppose_LinkList(linklist);
90.         while (tem != NULL) {
91.             printf("%d ", tem->data);
92.             tem = tem->next;
93.         }
94.         break;
95.     }
96. }
```

调试:

```
1 2 3 4 5 6 -1
1 2 3 4 5 6
please enter a chioce
3
6 5 4 3 2 1
-----
Process exited after 9.368 seconds with return value 0
请按任意键继续. . .
```

### 2.3

### 2.4

题目:

(《数据结构题集 (C 语言版)》, 第 2 章, 第 2.38 题) 设有一个双向循环链表, 每个结点中除有 **prior**, **data** 和 **next** 三个域外, 还增设了一个访问频度域 **freq**。在链表被起用之前, 频度域 **freq** 的值均初始化为零, 而每当对链表进行一次 **locate(L,x)** 的操作后, 被访问的结点 (即元素值等于 **x** 的结点) 中的频度域 **freq** 的值便增 1, 同时调整链表中结点之间的次序, 使其按访问频度非递增的次序顺序排列, 以便始终保持被频繁访问的结点总是靠近表头结点。试编写符合上述要求的 **locate** 操作的算法。

思路:

首先创建一个双向循环链表, 由控制台输入长度和 **data** 的值, 再显示出来, 由 **switch-case '4'** 进入 **Locate** 功能块, 输入需要被定位的 **x** 值, 调用 **Locate** 函数, 再输出修改过顺序的链表。

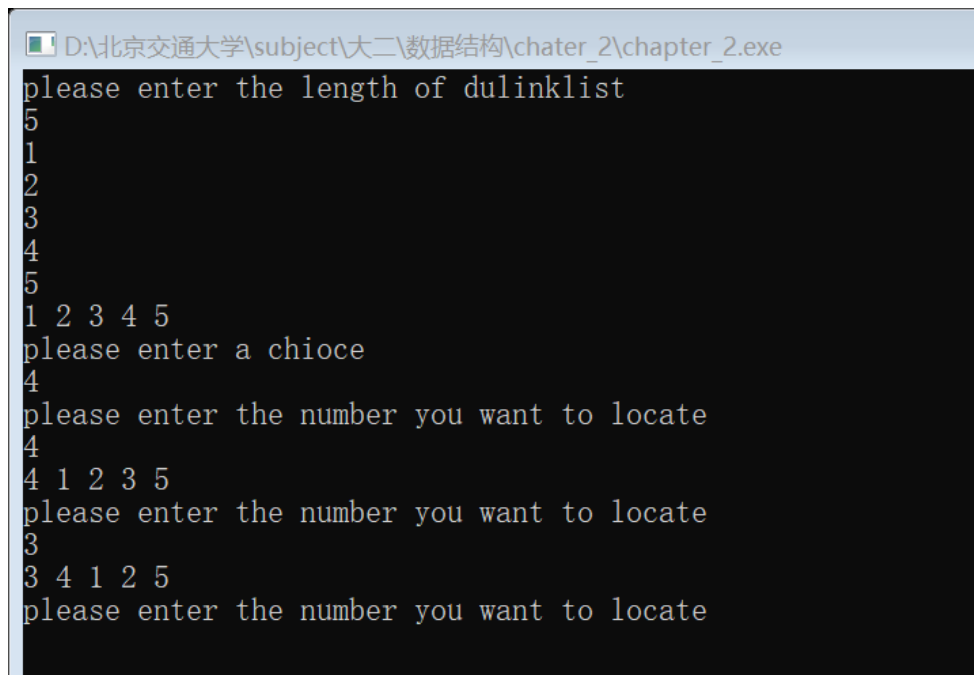
关于 **Locate** 函数, 首先遍历一遍以找到 **x**, 如找到则对应结点 **freq** 加 1。接着进行排序。因为 **freq** 每次只增加 1, 则只用和前边 **freq** 相同结点进行交换, 这里采取先删除再插入的方法。

```
1. #include "chapter2.h"
2. DuLinkedList *DuLinkedList_Locate(DuLinkedList *L, int x) {
3.     DuLinkedList *p = L->next, *q = L->next, *r;
4.     while (q != L && q->data != x) {
5.         q = q->next;
6.     }
```

```
7.     if (q == L) return NULL;
8.     q->freq++;
9.
10.    if (q->pre == L || q->pre->freq == q->freq) return NULL ;
11.
12.    q->pre->next = q->next;
13.    q->next->pre = q->pre; //删除结点q
14.    q->next = p;
15.    q->pre = p->pre;
16.    p->pre->next = q;
17.    p->pre = q;
18.
19.    return L;
20.    }
21. //建立双向链表
22. struct DuLinkedList *Createlist(int n) {
23.     DuLinkedList *head = (DuLinkedList *)malloc(sizeof(DuLinkedList));
24.     DuLinkedList *p, *q;
25.     int x;
26.     p = head;
27.     head->data = 0;
28.     head->freq = 0;
29.     for (int i = 0; i < n; i++) {
30.         q = (DuLinkedList *)malloc(sizeof(DuLinkedList));
31.         scanf("%d", &x);
32.         q->data = x;
33.         q->freq = 0;
34.         p->next = q;
35.         q->pre = p;
36.         p = q;
37.     }
38.     p->next = head;
39.     head->pre = p ;
40.     return head;
41. }
42.
43. //输出双向链表
44. void ShowDuLinkedList(struct DuLinkedList *linklist) {
45.     DuLinkedList *p = linklist;
46.     DuLinkedList *q = linklist->next;
47.     while (q != p) {
48.         printf("%d ", q->data);
49.         q = q->next;
```

```
50.     }
51.     printf("\n");
52. }
53.
54. int main() {
55.     int n, x;
56.
57.     printf("please enter the length of dulinklist\n");
58.     scanf("%d", &n);
59.     struct DuLinkList *linklist = CreateList( n);
60.     ShowDuLinkList(linklist);
61.     char chioce = '\0';
62.
63.     printf("please enter a chioce\n");
64.     scanf( "%s", &chioce);
65.
66.     switch (chioce) {
67.         case '4':
68.             while (chioce != 0) {
69.                 printf("please enter the number you want to locate\n");
70.                 scanf("%d", &x);
71.                 linklist = DuLinkList_Locate(linklist, x);
72.                 ShowDuLinkList(linklist);
73.             }
74.
75.             break;
76.
77.     }
78. }
```

调试:



```
D:\北京交通大学\subject\大二\数据结构\chater_2\chapter_2.exe
please enter the length of dulinklist
5
1
2
3
4
5
1 2 3 4 5
please enter a chioce
4
please enter the number you want to locate
4
4 1 2 3 5
please enter the number you want to locate
3
3 4 1 2 5
please enter the number you want to locate
```

2.4

分析:

调试情况正常, 但程序强健性有提升空间, 比如当 **data** 值重复时则每次只把相同的第一个结点 **freq** 增加, 这些都有待加强。