資料結構期末報告

鏈結串列

C111112102李步剛
C111112112蔡宗洛
C111112168王冠中

第七組

中華民國113年1月5日 星期五

目錄

01

特性介紹

為什麼要學鏈結串列

鏈結串列 VS 陣列 簡單比較

	鏈結串列	陣列
學習成本	高	低
新增、刪除、移動資料	**時間複雜度低** O(1)	**時間複雜度高** O(n)
存取資料	**時間複雜度高** O(n)	**時間複雜度低** O(1)
記憶體使用彈性	高	低
記憶體使用量	高	低

鏈結串列使用時機

1.需要頻繁新增、刪除及移動資料 => 時間複雜度低 O(1)

2.資料量大以至於無法得知確切數量 => 直接改變指標指向

02

陣列 v.s. 鏈結串列

陣列與鏈結串列的差異

內存分配方式：

- **陣列**：使用**連續**的記憶體空間來儲存元素，所有元素在記憶體中是相鄰的。
- **鏈結串列**：使用**非連續**的記憶體空間儲存元素，每個元素(節點)都包含下一個元素的地址，這樣的連接形成了一個串列。

陣列與鏈結串列的差異

大小動態性：

- **陣列：**具有**固定**大小，在創建時必須指定元素的數量。
- **鏈結串列：**具有**動態**大小，可以動態添加或刪除元素，無需事先知道元素的總數。

陣列與鏈結串列的差異

訪問速度：

- **陣列：**通過**索引**可以直接訪問元素，訪問速度**快**。
- **鏈結串列：**需要**從頭部開始尋訪節點**，無法直接通過索引訪問，訪問速度相對較**慢**。

陣列與鏈結串列的差異

插入和刪除操作：

- **陣列：** 插入和刪除操作可能涉及元素的移動，特別是在中間或開頭插入/刪除操作。
- **鏈結串列：** 插入和刪除操作更為**高效**，只需調整相鄰節點的指針，無需移動其他元素。

陣列與鏈結串列的差異

記憶體佔用空間：

- **陣列：**在元素數量**固定**的情況下，可能存在記憶體浪費的情況，特別是如果分配了大量的**固定**大小空間。
- **鏈結串列：**使用的記憶體空間相對更**靈活**，每個節點的大小可以**獨立分配**，**減少記憶體浪費**。

陣列與鏈結串列的差異

應用場景：

- **陣列：**適用於**固定**大小、需要**快速隨機訪問**的場景，例如**矩陣運算**。
- **鏈結串列：**適用於**動態**數據、需要**頻繁插入**和**刪除**操作的場景，例如**動態數據結構**和**高級數據結構**實現。

兩者之間的優缺點

特點	陣列	鏈結串列
隨機訪問速度	快速，使用索引直接訪問	較慢，需要從頭部遍歷節點
大小動態性	固定大小，無法動態調整	動態大小，可以動態添加或刪除
插入和刪除效率	低效，可能需要移動元素	高效，只需調整指針
記憶體使用彈性及用量	彈性:相對較低(固定大小依造成浪費) 用量:只存資料(較低)	彈性:相對較高(新增才建立空間，不用時可刪除) 用量:存資料和指標(較高)
實現複雜性	相對簡單	相對複雜，需要處理指標
適用場景	需要快速隨機訪問 已知資料量多寡	需要動態調整大小 頻繁插入和刪除操作 不需要快速隨機訪問

03

程式範例-功能說明

範例舉例說明

題目:電腦將產生一組為1~5且不重複的資料排序，使用者利用鏈結串列的概念新增及刪除來排序資料。

獲勝條件:使用者排序的資料至少有一個與電腦產生的資料排序相同。

敗北條件:使用者排序的資料均與電腦產生的資料排序不同。

功能說明-新增節點

新增節點資料

```
0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 1
請輸入需新增的資料: 5
```

輸入已達5個資料產生提醒

```
遊戲規則:排序1~5數字，若至少有一排序與電腦亂數排序相同則獲勝
電腦已排序完成，換你囉!(請排數值1~5之順序)
請輸入需新增的資料: 1

請輸入需新增的資料: 2

請輸入需新增的資料: 3

請輸入需新增的資料: 4

請輸入需新增的資料: 5

已完成排序囉!
```

輸入已達5個資料但仍選擇新增資料

```
遊戲規則:排序1~5數字，若至少有一排序與電腦亂數排序相同則獲勝
電腦已排序完成，換你囉!(請排數值1~5之順序)
請輸入需新增的資料: 1

請輸入需新增的資料: 2

請輸入需新增的資料: 3

請輸入需新增的資料: 4

請輸入需新增的資料: 5

已完成排序囉!

0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 1
輸入已達5個值，已完成排序囉!
```

功能說明-刪除節點、列印排序資料

刪除節點

```
0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 2
請輸入需刪除的資料: 5

資料 5 已被刪除
```

列印排序資料

```
0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 3
當前資料順序: 1  2  3  4  5
```

功能說明-與電腦亂數比較

先前有刪除節點但未新增新結點導致排序未滿5個

```
0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 2
請輸入需刪除的資料: 5

資料 5 已被刪除

0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 4


請重新操作(若要比較請填滿五個資料)
```

功能說明-與電腦亂數比較

達成獲勝條件

```
遊戲規則:排序1~5數字，若至少有一排序與電腦亂數排序相同則獲勝
電腦已排序完成，換你囉!(請排數值1~5之順序)
請輸入需新增的資料: 1

請輸入需新增的資料: 2

請輸入需新增的資料: 3

請輸入需新增的資料: 4

請輸入需新增的資料: 5

已完成排序囉!

0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 4
您第2位置值 2 與電腦之隨機亂數 2 相同
恭喜!至少有一個排序與電腦排序相同，您贏了!
電腦之隨機亂數排序: 4  2  5  3  1
```

達成敗北條件

```
遊戲規則:排序1~5數字，若至少有一排序與電腦亂數排序相同則獲勝
電腦已排序完成，換你囉!(請排數值1~5之順序)
請輸入需新增的資料: 1

請輸入需新增的資料: 2

請輸入需新增的資料: 3

請輸入需新增的資料: 4

請輸入需新增的資料: 5

已完成排序囉!

0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 4
不好意思，您所輸入的排序均與電腦之隨機亂數排序不同，您輸了
電腦之隨機亂數排序: 5  3  4  2  1
```

功能說明-再次遊戲、結束遊戲

獲勝後詢問

```
0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 4
您第2位置值 2 與電腦之隨機亂數 2 相同
恭喜!至少有一個排序與電腦排序相同，您贏了!
電腦之隨機亂數排序: 4  2  5  3  1

是否重新遊戲(繼續:1，結束:0)

1
遊戲規則:排序1~5數字，若至少有一排序與電腦亂數排序相同則獲勝
電腦已排序完成，換你囉!(請排數值1~5之順序)
請輸入需新增的資料:
```

```
0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 4
您第2位置值 2 與電腦之隨機亂數 2 相同
恭喜!至少有一個排序與電腦排序相同，您贏了!
電腦之隨機亂數排序: 5  2  4  3  1

是否重新遊戲(繼續:1，結束:0)

0

遊戲結束，歡迎下次再來!

--------------------------------
Process exited after 12.31 seconds with return value 0
請按任意鍵繼續 . . .
```

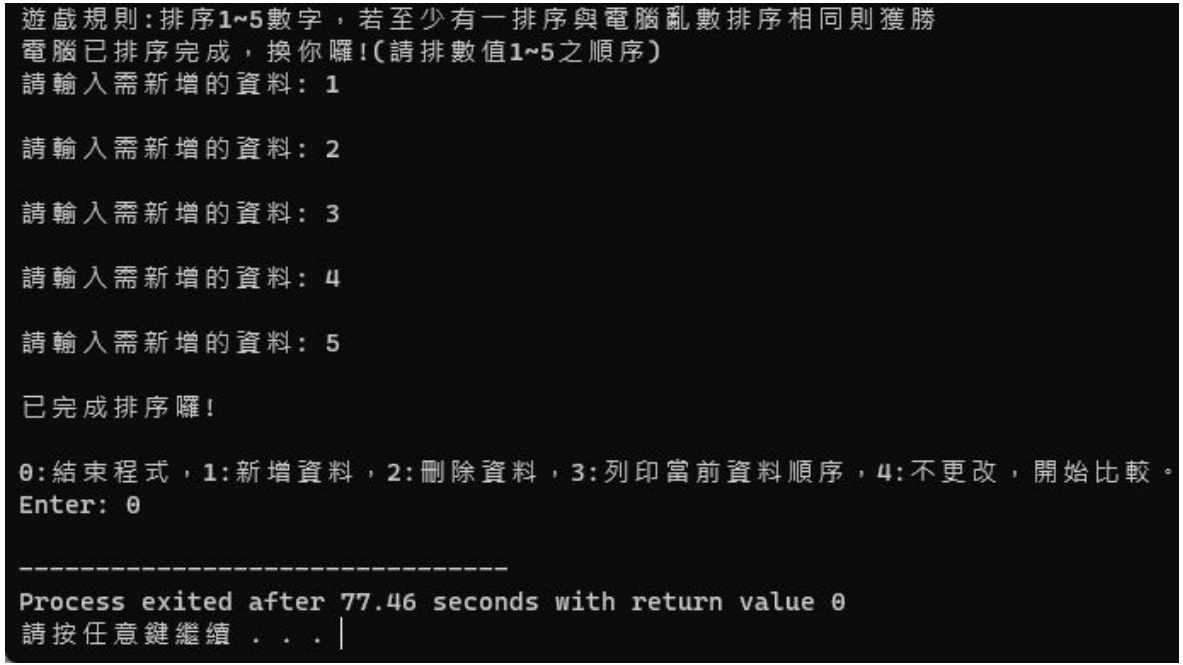
敗北後詢問

```
0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 4
不好意思，您所輸入的排序均與電腦之隨機亂數排序不同，您輸了
電腦之隨機亂數排序: 5  3  4  2  1

是否重新遊戲(繼續:1，結束:0)

1
遊戲規則:排序1~5數字，若至少有一排序與電腦亂數排序相同則獲勝
電腦已排序完成，換你囉!(請排數值1~5之順序)
請輸入需新增的資料:
```

```
0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。
Enter: 4
不好意思，您所輸入的排序均與電腦之隨機亂數排序不同，您輸了
電腦之隨機亂數排序: 3  4  2  5  1

是否重新遊戲(繼續:1，結束:0)

0

遊戲結束，歡迎下次再來!

--------------------------------
Process exited after 79.25 seconds with return value 0
請按任意鍵繼續 . . .
```

功能說明-結束遊戲

不比較結束



04

程式範例-程式解說

程式解說-定義

定義標頭檔(include)

```
#include<stdio.h>
#include<malloc.h>
#include<time.h>
```

standard input output.header(標準輸入輸出頭文件)

memory allocation(動態記憶體分配)

取得時間之標頭檔

```
typedef struct _NODE_
{
    int data;
    struct _NODE_ *next;
} NODE;

NODE *head,*pnew,*p1;
```

定義結構(內容及名稱)

```
void create(int n);
```

定義函式(建立節點用)

程式解說-定義(main內)

```
int ask;//功能
int c;//計數
int del;//刪除
int i,j,z,n,a,x,y;//判斷工具
int computer[5]={0};//亂數陣列
```

定義變數及亂數用陣列

```
srand(time(NULL));//抓取系統時間(沒有此項得出亂數會不變)
```

srand(產生亂數用)

程式解說-產生亂數(第一層do-while內(a==1))

```
ask=1;
c=-1;
a=-1;
for(j=0;j<5;j++)
{
    computer[j]=rand()%5+1;
    for(z=0;z<j;z++)
    {
        if(computer[j]==computer[z])
        {
            j--;
            break;
        }
    }
}
printf("遊戲規則:排序1~5數字，若至少有一排序與電腦亂數排序相同則獲勝\n");
printf("電腦已排序完成，換你囉!(請排數值1~5之順序)\n");
```

給定變數初始值

產生亂數排列

避免重複

程式解說-詢問功能(第二層do-while內(ask!=0))

```
if( ask < 0 || ask > 4 )
{
    printf("需輸入0~4之間的數字!\n");
    continue;
}
if(c!=-1)
{
    if(c==5)
    {
        printf("已完成排序囉!\n\n");
    }

    printf( "0:結束程式，1:新增資料，2:刪除資料，3:列印當前資料順序，4:不更改，開始比較。\nEnter: " );
    scanf( "%d", &ask );
}
```

程式解說-新增節點(case 1)

```c
do
{
    printf("請輸入需新增的資料: ");
    scanf(" %d", &n );

    if(n>0&&n<=5)
    {
        if(c==-1)
        {
            create(n);
            p1=head;
            c=c+2;
        }
        else
        {
            int p=0;
            p1=head;
            while(p1!=NULL)
            {
                if(p1->data==n)
                {
                    p=1;
                    printf("不可輸入重複之值\n\n");
                    break;
                }
                p1 = p1->next;
            }
```

新增第一個節點

判斷是否有重複

```c
        if(c==5)
        {
            printf("輸入已達5個值，");
            continue;
        }
                if(p1==NULL)
                {
                    create(n);
                    c++;
                }
            }
        }
        else
        {
            printf( "請輸入1~5之間的數字\n\n");
        }
    }while(n<=0||n>5||c<5);
    break;
```

新增第二個以後之節點

程式解說-圖示(新增節點)

```
void create(int n)
{
    pnew = (NODE*)malloc(sizeof(NODE));
    pnew->data=n;
    pnew->next=NULL;
    if (head == NULL)
    {
        pnew->next = head;
        head = pnew;
    }
```

新增節點
填入資料

找出一個節點A(其下一節點為空)

```
    else
    {
        p1=head;
        while (p1->next != NULL)
        {
            p1 = p1->next;
        }
        pnew->next = p1->next;
        p1->next = pnew;
    }
    printf("\n");
}
```

將新增之節點資料填入
節點A之下一節點

程式解說-刪除節點(case 2)

```c
if(c==0)
{
    printf("\n已無任何資料排序\n\n");
    continue;
}
NODE*p2=NULL;
p1=head;
printf("請輸入需刪除的資料: ");
scanf( "%d", &del );
while (p1 != NULL && p1->data != del)
{
    p2 = p1;
    p1 = p1->next;
}
```

迴圈刪除指定節點資料

```c
if (p1 == NULL)
{
    printf("\n%d 已被刪除，請輸入其他值\n\n",del);
    continue;
}
if (p2 == NULL)
{
    head = p1->next;
    free(p2);
}
else
{
    p2->next = p1->next;
    free(p1);
}
printf("\n資料 %d 已被刪除\n\n", del);
c--;
break;
```

需刪除節點在第一個

需刪除節點非第一個

程式解說-列印資料順序(case 3)

```
p1=head;
printf("當前資料順序:");
while(p1!=NULL)
{
    printf(" %d ",p1->data);
    p1 = p1->next;
}
printf("\n\n");
break;
```

迴圈列印節點資料

程式解說-比較資料順序(case 4)&再次遊戲

```
if(a==1||a==0)
{
    break;
}
```

不論是否繼續遊戲均跳出第二層do-while(條件:ask!=0)

```
if(y==0)
{
    printf("不好意思，您所輸入的排序均與電腦之隨機亂數排序不同，您輸了\n");
}
else
{
    printf("恭喜!至少有一個排序與電腦排序相同，您贏了!\n");
}
printf("電腦之隨機亂數排序:");
for(j=0;j<5;j++)
{
    printf(" %d ",computer[j]);
}
printf("\n\n是否重新遊戲(繼續:1，結束:0)\n\n");
scanf("%d",&a);
if(a==1)
{
    head=NULL;
}
break;
```

重新遊戲head恢復原始

```
p1=head;
x=0;
y=0;
if(c<5){
    printf("\n\n請重新操作(若要比較請填滿五個資料)\n\n");
    continue;
}
while(p1!=NULL)
{
    if(p1->data==computer[x])
    {
        printf("您第%d位置值 %d 與電腦之隨機亂數 %d 相同\n",x+1,p1->data,computer[x]);
        y++;
    }
    p1 = p1->next;
    x++;
}
```

迴圈比較節點資料

工作分配 05

李步剛-程式撰寫、簡報製作(理論及程式部分)

蔡宗洛-簡報製作(理論部分)

王冠中-程式修改、簡報製作(理論部分)

Thanks
For Listening