

## 19. 抽象資料型態

抽象資料型態(Abstract Data Type[ADT])是一組資料處理方法的集合，它只是一種概念，規範定義其該提供哪些相關的操作，但不限制其實作的方式。本章以常見的「鏈結串列(linked list)」為例進行相關討論，它可以用以管理不定數目的資料項目，figure 1是一個包含有四個資料項目的鏈結串列：

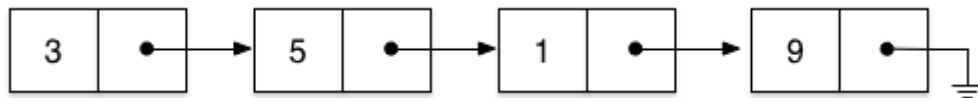


Fig. 1: 含有四筆資料項目的鏈結串列範例

並提供以下的操作：

- 取回第一筆資料
- 取回最後一筆資料
- 取回鏈結串列中的資料項目個數
- 該鏈結串列是否為空(沒有任何資料)
- 加入一筆資料項目到最前面
- 加入一筆資料項目到最後面
- 移除第一筆資料項目
- 移除最後一筆資料項目
- 搜尋特定的資料項目
- 搜尋特定的資料項目，並加以移除
- 搜尋特定的資料項目，並更新其資料
- 在某一筆資料項目的後面加入一筆新的資料項目
- 在某一筆資料項目的前面加入一筆新的資料項目

為了便於討論相關的實作，我們為上述的操作命名如下：

- getFirst: 取回第一筆資料
- getLast: 取回最後一筆資料
- getSize: 取回鏈結串列中的資料項目個數
- isEmpty: 該鏈結串列是否為空(沒有任何資料)
- addFirst: 加入一筆資料項目到最前面
- addLast: 加入一筆資料項目到最後面
- removeFirst: 移除第一筆資料項目
- removeLast: 移除最後一筆資料項目
- search: 搜尋特定的資料項目
- remove: 搜尋特定的資料項目，並加以移除
- replace: 搜尋特定的資料項目，並更新其資料
- addAfter: 在某一筆資料項目的後面加入一筆新的資料項目
- addBefore: 在某一筆資料項目的前面加入一筆新的資料項目

通常，鏈結串列的實作，是以指標配合結構體與動態記憶體配置等方法進行設計。一筆資料項目是以結構體的方式來定義，稱為節點(node)包含有一個存放資料的欄位，以及一個指向下一個節點的指標，

如figure 2所示：

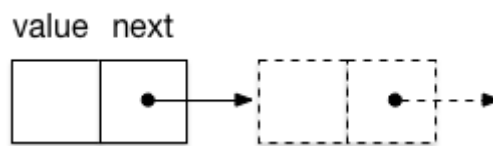


Fig. 2: 典型的鏈結串列的節點

## 19.1 資料節點定義

其程式碼宣告如下：

```
struct node
{
    int value;
    struct node *next;
};

typedef struct node Node;
```

其中第一個欄位為所要儲存的資料，你可以視需要改成所需的型態，或是宣告一個適當的結構體，用以儲存相關的資料。

```
typedef struct
{
    int x;
    int y;
} Point;

struct node
{
    Point point;
    struct node *next;
};
```

為了簡化討論，我們還是以前者做為討論的對象。

## 19.2 head與tail指標

我們通常會為一個鏈結串列建立以下的指標：

- head: 指向第一個節點
- tail: 指向最後一個節點

在開始時，因為鏈結串列中並沒有任何的節點，所以我們可以設定為NULL。相關程式碼如下：

```
struct node *head=NULL;
struct node *tail=NULL;
```

或是

```
Node *head=NULL;
Node *tail=NULL;
```

為了便利討論，我們也設計了以下兩個函式：

```
void showANode(Node *p)
{
    if(p)
        printf("%d", p->value);
}

void showAllNodes()
{
    Node *p=head;
    while(p)
    {
        showANode(p);
        printf(" -->");
        p=p->next;
    }
    printf("NULL\n");
}
```

## 19.3 getFirst()與getLast()實作

此兩函式的實作相當簡單，直接傳回head指標與tail指標即可：

```
Node *getFirst()
{
    return head;
}

Node *getLast()
{
    return tail;
}
```

## 19.4 getSize()實作

```
int getSize()
{
    int c=0;
    Node *p=head;

    while(p)
    {
        c++;
        p = p->next;
    }
    return c;
}
```

## 19.5 isEmpty()實作

```
int isEmpty()
{
    return !head;
}
```

## 19.6 addFirst()與addLast()實作

```
Node *createANode(int v)
{
    Node *newNode = malloc(sizeof(Node));
    newNode->value = v;
    newNode->next = NULL;
    return newNode;
}

void addFirst(int v)
{
    Node *newNode = createANode(v);
    if(head!=NULL)
        newNode->next = head;
    else
        tail = newNode;
    head = newNode;
}
```

```
void addLast(int v)
{
    Node *newNode = createANode(v);
    if(tail!=NULL)
        tail->next=newNode;
    else
        head=newNode;
    tail=newNode;
}
```

## 19.7 removeFirst()與removeLast()實作

```
void removeFirst()
{
    Node *p;

    if(head!=NULL)
    {
        if(head==tail)
        {
            free(head);
            head=tail=NULL;
        }
        else
        {
            p=head;
            head=head->next;
            free(p);
        }
    }
}
```

```
void removeLast()
{
    Node *p;

    if(tail!=NULL)
    {
        if(head==tail)
        {
            free(head);
            head=tail=NULL;
        }
        else
        {
            p=tail;
            while(p->prev!=NULL)
                p=p->prev;
            p->next=NULL;
            tail=p;
            free(p->next);
        }
    }
}
```

```
    p=head;
    while(p->next!=tail)
    {
        p=p->next;
    }
    free(tail);
    p->next=NULL;
    tail=p;
}
}
```

## 19.8 search()實作

```
Node *search(int v)
{
    Node *p=head;
    while(p)
    {
        if(p->value == v)
            return p;
        p=p->next;
    }
    return NULL;
}
```

## 19.9 remove()實作

此函式會搜尋欲刪除的資料節點，於刪除後傳回前一筆資料所在的位址。

```
Node *removeANode(int v)
{
    Node *p=head;
    Node *q;
    if((p!=NULL)&&(p->value==v))
    {
        head=head->next;
        free(p);
    }
    else
    {
        while(p->next)
        {
            if(p->next->value == v)
            {

```

```
        q=p->next;
        p->next=q->next;
        if(q==tail)
        {
            tail=p;
        }
        free(q);
        return p;
    }
    p=p->next;
}
}
return NULL;
}
```

## 19.10 replace()實作

如果成功代換了數值則傳回1，否則傳回0。

```
int replace(int v, int newV)
{
    Node *p=head;
    while(p)
    {
        if(p->value==v)
        {
            p->value=newV;
            return 1;
        }
        p=p->next;
    }
    return 0;
}
```

## 19.11 其它

- addAfter: 在某一筆資料項目的後面加入一筆新的資料項目
- addBefore: 在某一筆資料項目的前面加入一筆新的資料項目

From:

<https://junwu.nptu.edu.tw/dokuwiki/> - Jun Wu的教學網頁

國立屏東大學資訊工程學系

CSIE, NPTU

Total: 118849

Permanent link:

<https://junwu.nptu.edu.tw/dokuwiki/doku.php?id=c:adt>

Last update: **2019/07/02 15:01**

