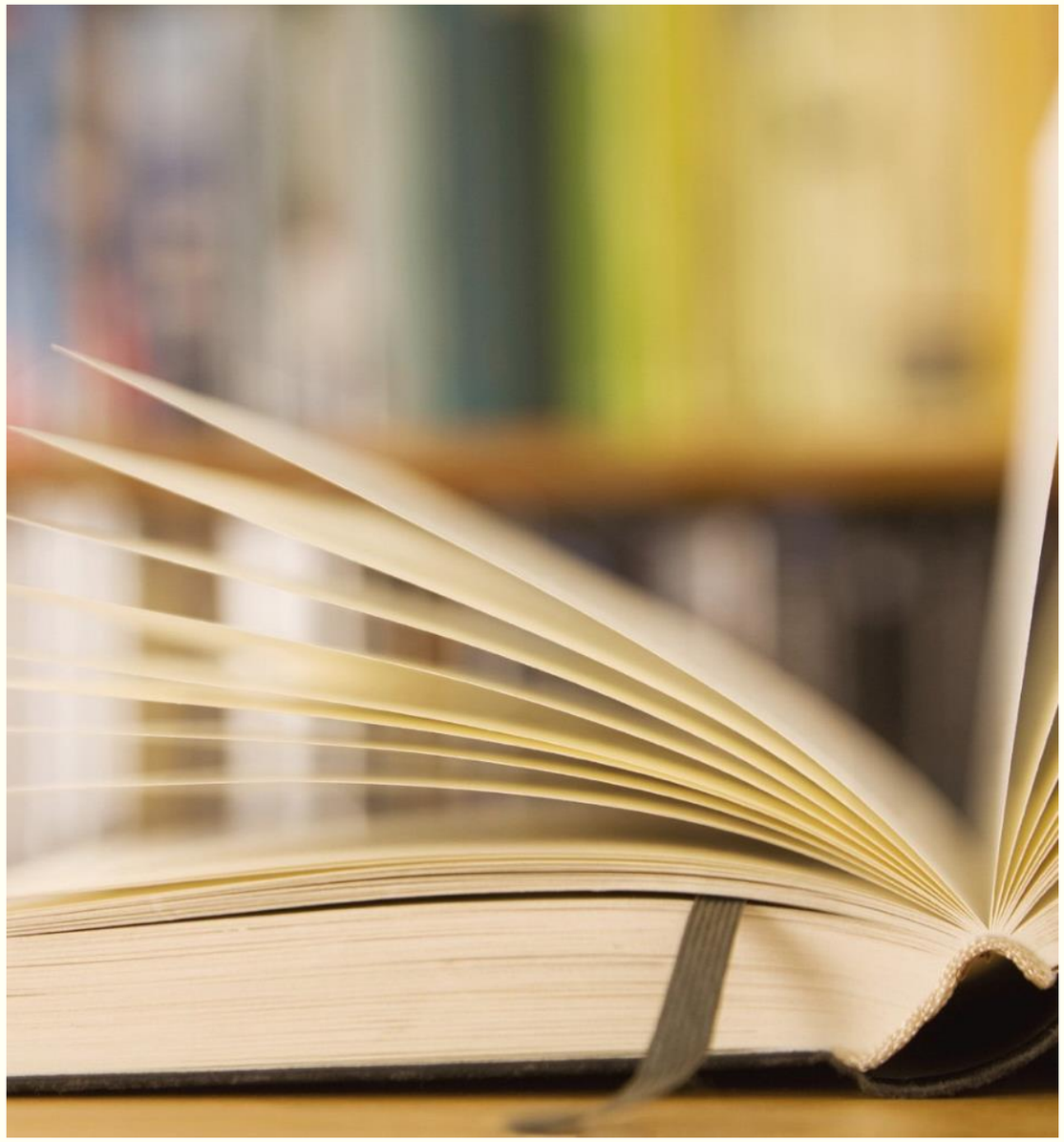
DANH SÁCH



Nội dung chính

- Danh sách tuyến tính
  - Khái niệm
  - Các phép toán trên danh sách
  - Cài đặt danh sách trên cơ sở mảng

- Danh sách liên kết
  - Khái niệm
  - Các phép toán trên danh sách
  - Cài đặt danh sách bằng con trỏ

Danh sách tuyến tính – Khái niệm

- Danh sách là một dãy hữu hạn các phần tử thuộc cùng một lớp các đối tượng nào đó.
- Danh sách tuyến tính là các phần tử của nó được sắp tuyến tính: nếu n>1 thì phần a_i ở trước phần tử a_{i+1}. Với a_i là phần tử ở vị trí thứ i của danh sách.

Danh sách tuyến tính – Khái niệm (tiếp)

- Gọi L là danh sách có n (n > 0) phần tử.
  - L = (a_1,a_2, …,a_n)
- Gọi n là độ dài của danh sách.
  - Nếu n = 0:
    - L là danh sách rỗng
  - Nếu n>1 :
    - a_1 là phần tử đầu tiên của danh sách
    - a_n là phần tử cuối cùng của danh sách

Danh sách tuyến tính – Khái niệm (tiếp)

- Một đối tượng có thể xuất hiện nhiều lần trong một danh sách.
- VD:
  - 1, 1, 2, 3, 5,8,13,21,34,55,89 là danh sách các số fibonacci
  - (31,28,31,30,31,30,31,31,30,31,30,31) là danh sách số ngày của các tháng trong một năm

Danh sách tuyến tính – Các phép toán

- Khởi tạo danh sách
- Xác định độ dài của danh sách
- Loại bỏ phần tử ở vị trí thứ p trong danh sách
- Xen phần tử x vào danh sách sau vị trí p
- Xen phần tử x vào danh sách trước vị trí p

Danh sách tuyến tính – Các phép toán (tiếp…)

- Tìm kiếm trong danh sách có chứa phần tử x hay không?
- Kiểm tra danh sách có đầy hay không?
- Kiểm tra danh sách có rỗng hay không?
- Duyệt danh sách

Danh sách tuyến tính – Cài đặt trên cơ sở mảng

- Sử dụng mảng để lưu trữ các phần tử của danh sách, trong đó mỗi thành phần của mảng sẽ lưu giữ một phần tử của danh sách.
- Các phần tử liên tiếp nhau của danh sách được lưu trữ trong các thành phần liên tiếp nhau của mảng

```
const int Max=30;
template <class Item>
struct     List{
        Item element[::Max];
        int count;
};
```

Danh sách tuyến tính – Cài đặt trên cơ sở mảng (tiếp…)

- Max chiều dài tối đa của danh sách
- Xây dựng cấu trúc List gồm 2 thành phần
  - element là một mảng các phần tử thuộc kiểu dữ liệu item
  - count lưu chỉ số của thành phần mảng lưu giữ phần tử cuối cùng của danh sách

Danh sách tuyến tính – Khởi tạo danh sách rỗng

- Khởi tạo danh sách rỗng, ta gán giá trị count = 0

```
template <class Item>
    void Initial(List<Item> &L){

        L.count=0;

}
```

Danh sách tuyến tính – Xác định số phần tử của danh sách

- Xây dựng hàm Length:
  - Input : Danh sách L
  - Output: Số phần tử của danh sách

```
// Lay do dai cua danh sach
template <class Item>
int GetLength(List<Item> L){
    return L.count;
}
```

Danh sách tuyến tính – Kiểm tra danh sách đầy

- Xây dựng hàm isFull:
  - Input : Danh sách L
  - Output:
    - 1: Danh sách đầy
    - 0: Danh sách chưa đầy

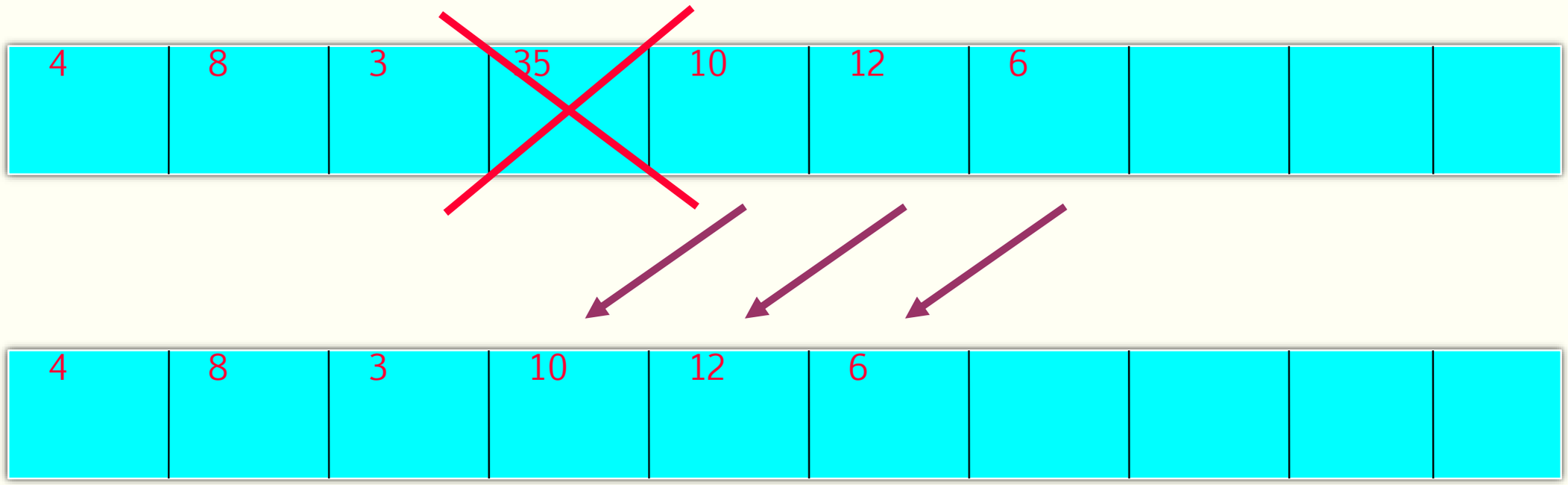
```
// Kiem tra danh sach day
    template <class Item>
    bool IsFull(List<Item> L){

         if (L.count==::Max) return true;
         else return false;

    }
```

Danh sách tuyến tính – Kiểm tra danh sách rỗng

- Xây dựng hàm isEmpty:
  - Input : Danh sách L
  - Output:
    - 1: Danh sách rỗng
    - 0: Còn phần tử trong danh sách

```
template <class Item>
bool IsEmpty(List<Item> L){

      if (L.count==0) return true;
      else return false;

}
```

Danh sách tuyến tính – Loại bỏ phần tử ở vị trí thứ p trong danh sách



Danh sách tuyến tính – Loại bỏ phần tử ở vị trí thứ p trong danh sách

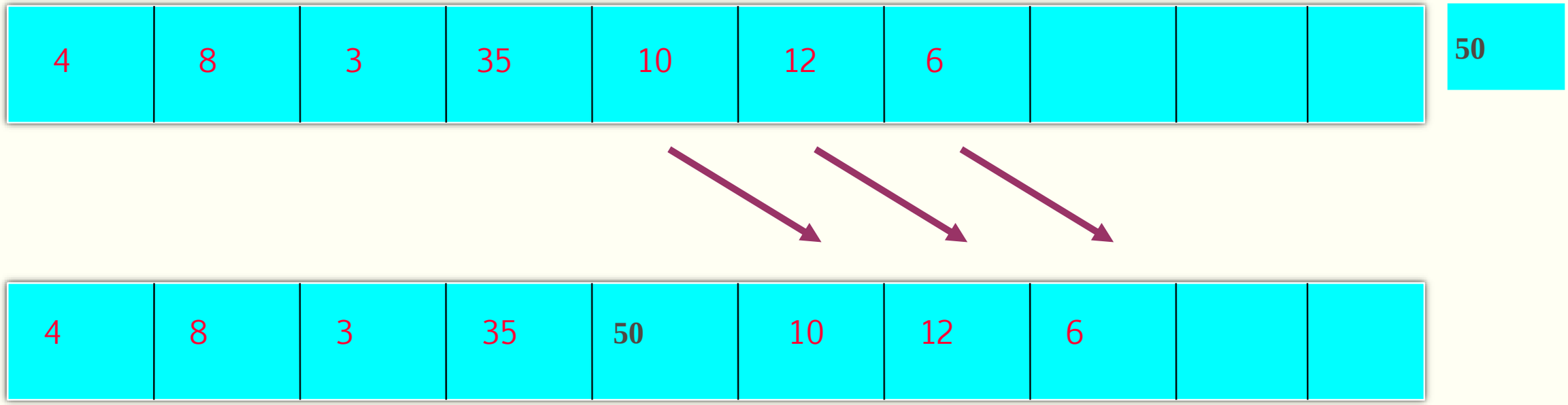
- Thuật toán:
  - Bước 1:Thực hiện kiểm tra nếu danh sách không rỗng và p chỉ vào một phần tử trong danh sách thì thực hiện bước 2, ngược lại thì dừng.
  - Bước 2: Ta tịnh tiến các phần tử ở các vị trí p+1, p+2,…đến các vị trí p,p+1… (tịnh tiến lên một vị trí)

Danh sách tuyến tính – Loại bỏ phần tử ở vị trí thứ p trong danh sách

- Xây dựng hàm Delete
  - Input:
    - Danh sách L
    - Vị trị phần tử cần xoá p
    - Tham biến q cho biết việc xoá có thành công hay không?
  - Output:
    - q =1 nếu việc xoá thành công
    - q = 0 nếu việc xoá không thành công

```
// Loai bo phan tu o vi tri p
template <class Item>
void Deleted(List<Item> &L, int p, bool &q) {

    if ((p<=0)||(p>L.count))
        q=false;
    else {

        for(int i=p+1; i<=L.count; i++)
            L.element[i]=L.element[i-1];
        L.count--;

    }
}
```

Danh sách tuyến tính – Chèn phần tử x vào danh sách sau vị trí p



Danh sách tuyến tính – Chèn phần tử x vào danh sách sau vị trí p

- Thuật toán:
  - Bước 1:Thực hiện kiểm tra nếu danh sách chưa đầy và p chỉ vào một phần tử trong danh sách thì thực hiện bước 2, ngược lại thì dừng.
  - Bước 2: Ta tịnh tiến các phần tử ở các vị trí L.count, L.count -1,…đến các vị trí L.count +1,L.count…
  - Bước 3: Chèn phần tử x vào vị trí p

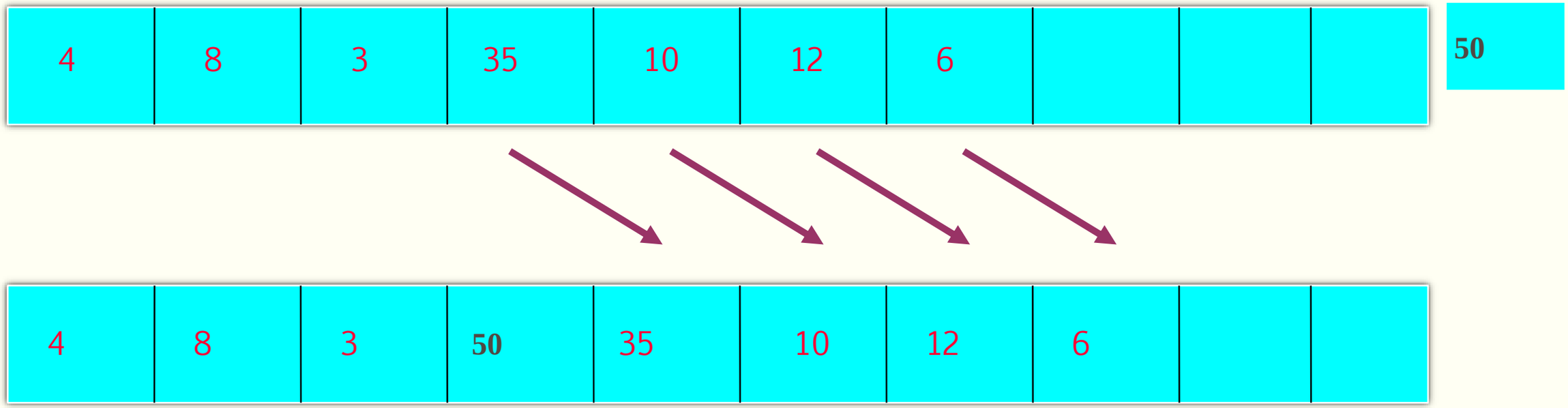
Danh sách tuyến tính – Chèn phần tử x vào danh sách sau vị trí p

- Xây dựng hàm InsertAfter
  - Input:
    - Danh sách L
    - Vị trị p
    - Phần tử x thuộc kiểu Item cần chèn
    - Tham biến q cho biết việc chèn có thành công hay không?
  - Output:
    - q =1 nếu việc chèn thành công
    - q = 0 nếu việc chèn không thành công

Danh sách tuyến tính – Chèn phần tử x vào danh sách sau vị trí p

```
// Chen mot phan tu vao sau vi tri p trong danh sach
template <class Item>
void insertAfter(List<Item> &L,Item x,int p,bool &q) {

    if(IsEmpty(L)) {
        L.count++;
        L.element[L.count]=x;

    }
    else
        if (IsFull(L)||(p<=0)||(p>L.count))
            q=false;
        else {
            L.count++;
            q=true;

            for (int i=L.count; i>p+1; i--)
                L.element[i]=L.element[i-1];
            L.element[p+1]=x;

        }

}
```

Danh sách tuyến tính – Chèn phần tử x vào danh sách trước vị trí p



Danh sách tuyến tính – Chèn phần tử x vào danh sách trước vị trí p

- Thuật toán:
  - Bước 1:Thực hiện kiểm tra nếu danh sách chưa đầy và p chỉ vào một phần tử trong danh sách thì thực hiện bước 2, ngược lại thì dừng.
  - Bước 2: Ta tịnh tiến các phần tử ở các vị trí L.count, L.count -1,…đến các vị trí L.count +1,L.count…
  - Bước 3: Chèn phần tử x vào vị trí p

Danh sách tuyến tính – Chèn phần tử x vào danh sách trước vị trí p

- Xây dựng hàm InsertBefore
  - Input:
    - Danh sách L
    - Vị trị p
    - Phần tử x thuộc kiểu Item cần chèn
    - Tham biến q cho biết việc chèn có thành công hay không?
  - Output:
    - q =1 nếu việc chèn thành công
    - q = 0 nếu việc chèn không thành công

Danh sách tuyến tính – Chèn phần tử x vào danh sách trước vị trí p

```
// Chen mot phan tu vao truoc vi tri p trong danh sach
template <class Item>
void InsertBefore(List<Item> &L, Item x,int p, bool &q) {
    int i;
    if(IsEmpty(L)) {
        L.count++;
        L.element[L.count]=x;

    } else {

        if (IsFull(L)||(p<=0)||(p>L.count))
            q=false;
        else {
            q =true;
            i = L.count+1;
            while(i>=p) {
                L.element[i] = L.element[i-1];
                i--;
            }
            L.element[p] = x;
            L.count = L.count +1;
        }
    }
}
```

Danh sách tuyến tính – Tìm kiếm phần tử x trong danh sách

- Ta sử dụng phương pháp tìm kiếm tuần tự xây dựng hàm Search
  - Input:
    - Danh sách L
    - Phần tử x
    - Tham biến found cho biết có tìm được hay không?
    - Tham biến p cho biết vị trí của phần tử x trong danh sách nếu x có trong danh sách

Danh sách tuyến tính – Tìm kiếm phần tử x trong danh sách (tiếp…)

- Output:
  - Found:
    - 1: nếu có x trong danh sách
    - 0: nếu không có x trong danh sách
  - P:
    - Nếu found =1 p trả lại vị trí của x trong danh sách
    - Nếu found =0, p =0;

Danh sách tuyến tính – Tìm kiếm phần tử x trong danh sách (tiếp…)

```
void Search(List &L, item x, int &found, long &p){
      found=0;p=1;
      while((!found) && (p<=L.count)){
            if (L.element[p] = = x) found =1;
            else p++;
      }
      if(!found) p=0;
}
```

Danh sách tuyến tính – Duyệt danh sách

- Trong nhiều bài toán chúng ta cần phải thao tác với các phần tử của danh sách. Do đó cần phải đi qua danh sách từ phần tử đầu tiên cho đến hết.
- Xây dựng hàm Traverse duyệt danh sách.

Danh sách tuyến tính – Duyệt danh sách (tiếp…)

- Input:
  - Danh sách L
- Output:
  - None

```
// Duyet danh sach tuyen tinh
template <class Item>

void  Travel(List<Item> L) {

    for (int i=1; i<=L.count; i++)
        cout<<L.element[i]<<"  ";
}
```

Danh sách tuyến tính – Bài toán quản lý sinh viên

- Bài toán quản lý sinh viên:
  - Kiểu dữ liệu Sinh viên gồm các trường Mã,Tên, Điểm Toán, Điểm Lý, Điểm Hoá.
  - Yêu cầu:
    - Xây dựng danh sách để quản lý điểm sv
    - Nhập 5 sinh viên
    - Bổ sung sinh viên vào danh sách
    - In danh sách sinh viên gồm tên, điểm trung bình
    - Tìm kiếm sinh viên theo mã

Danh sách tuyến tính – Bài toán quản lý sinh viên (tiếp…)

```
Struct sinhvien{
    int Ma;
    char[27] Ten;
    int dt,dl,dh;
};
```

Danh sách tuyến tính – Bài toán quản lý sinh viên (tiếp…)

```
Const max =N;
struct List{
    sinhvien element[max];
    long count;
} L;
```

Danh sách tuyến tính – Bài toán quản lý sinh viên (tiếp…)

```
void nhapmoi(){
      int i=1,d;  char[27] ten; sinhvien sv;
      do{
        printf(“\n Nhập ten sv”);gets(ten);
        sv.ma = i;strcpy(sv.ten,ten);
        printf(“\nDiem Toán:”);scanf(“%d”,&d);sv.dt =d;
        printf(“\nDiem Lý:”);scanf(“%d”,&d);sv.dl =d;
         printf(“\nDiem Hoá:”);scanf(“%d”,&d);sv.dh =d;
        InsertAfter(L,sv,i);
        i++;
      }while(i<=5);
}
```

Danh sách tuyến tính – Bài toán quản lý sinh viên (tiếp…)

```
void Inds(){
      int i=1; float dtb;
      while (i <=L.count){
          printf("Sinh vien %s" ,L.element[i].ten);
          dtb =(L.element[i].dt+ L.element[i].dl +
      L.element[i].dh)/3;
          printf("ĐTB :%4.2f", dtb);
         i++;
      }
}
```

Danh sách liên kết – Khái niệm

- Danh sách liên kết là danh sách, mỗi phần tử của nó gồm hai thành phần:
  - Phần chứa dữ liệu -Data
  - Phần chỉ vị trí của phần tử tiếp theo trong danh sách - Next
- Trường Next có giá trị từ 0 đến max (max :số phần tử tối đa của danh sách)

Danh sách liên kết

Chỉ số	Data	Next
1	?	?
2	B	4
3	?	?
4	C	0
5	?	?
6	?	?
7	A	2
8	?	?

Danh sách liên kết – Khái niệm

- Khác với danh sách tuyến tính các phần tử được cho bởi chỉ số của mảng. Phần tử đầu tiên được lưu ở vị trí 1, và phần tử tiếp theo của phần tử thứ i được tìm thấy ở vị trí i+1 của mảng.
- Danh sách liên kết chỉ cần xác định vị trí của phần tử đầu danh sách. Các phần tử tiếp theo được định vị thông qua trường Next.

Danh sách liên kết – Các phép toán

- Khởi động vùng lưu trữ tự do
- Lấy một node tự do
- Xoá một node
- Tạo danh sách liên kết rỗng
- Kiểm tra danh sách có rỗng hay không?
- Kiểm tra danh sách có đầy không?
- Chèn, Xoá, tìm kiếm
- Duyệt danh sách.

Xây dựng danh sách liên kết trên cơ sở con trỏ

```
6   #include<iostream>
7   using namespace std;
8   // Dinh nghia cau truc danh sach
9   template <class Item>
10  struct List{
11          Item element;
12          List *next;
13  };
```

Xây dựng danh sách liên kết trên cơ sở con trỏ (tiếp…)

- Tạo danh sách rỗng

```
template <class Item>
// Khoi tao danh sach lien ket
void Initial(List<Item> *&head,List<Item> *&tail){
 head=NULL;tail=NULL;
}
```

Xây dựng danh sách liên kết trên cơ sở con trỏ (tiếp…)

- Kiểm tra danh sách rỗng?

```
// Kiem tra danh sach rong
bool IsEmpty(List<Item> *head,List<Item> *tail){
      if (head==NULL) return true;
      else return false;
}
```

Danh sách liên kết – Chèn phần tử vào danh sách

```
// Chen mot phan tu vao danh sach
template <class Item>
void InsertLast(List<Item> *&head,List<Item> *&tail,Item x){

    List<Item> *p;
     p=new List<Item>;
     p->element= x;
     p->next=NULL;

     if (IsEmpty(head,tail)) {head=p;tail=p;}
     else {tail->next=p;tail=p;}
}
```

Danh sách liên kết – Chèn phần tử vào danh sách

```
// chen mot phan tu vao vi tri i
template <class Item>
void Insert(List<Item> *&head,List<Item> *&tail, int i, Item x){
      List<Item> *p;
      p=new List<Item>;
      p->element=x;
      p->next=NULL;
      if (i==1){
         if (IsEmpty(head,tail)){head=p;tail=p;}
         else {p->next=head;head=p;}
      }
      else {
            List<Item> *q=head;

            int count=1;
            while ((count<i-1)&&(q!=NULL)){
                  q=q->next;
                  count++;
            }
            if (q!=NULL) {p->next=q->next; q->next=p;}
      }
}
```

Danh sách liên kết – Xoá phần tử khỏi danh sách

```
Procedure Delete(var L: PointerType, PrevPtr: PointerType){
    var Temp : PointerType;
    Begin
    if isEmpty(L) then halt
    else
        begin
            if (PrevPtr = nil) then
                begin
                    Temp := L;
                    L := Temp^.Next;
            end
        else
            begin
                    Temp:= PrevPtr^.Next;
                    PrevPtr^.Next := Temp^.Next;
            end;
          dispose(Temp);
End;
```

Danh sách liên kết – Duyệt danh sách

```
59  // Hien thi danh sach
60  template <class Item>
61  void display(List<Item> *head,List<Item> *tail){
62        List<Item> *p;
63        p=head;
64        while (p!=NULL){
65              cout<<p->element<<"   ";
66              p=p->next;
67        }
68
69  }
```

Bài tập

- Nhập vào một số nguyên dương n, tiếp theo là n số nguyên của một dãy số, hãy cài đặt nó vào một danh sách liên kết đơn. Tiếp theo cho một số nguyên k, ($0 \le k < n$), hãy xóa phần tử ở chỉ số k và in ra màn hình danh sách đó, sau một phần tử có một khoảng trắng.
- Nhập vào một số nguyên dương n, hãy cài đặt một danh sách lên kết đôi để lưu các số từ n giảm về 1 và từ 1 tăng đến n. In ra danh sách liên kết đó, phía sau mỗi phần tử có một khoảng trắng.

Tổng kết

- Cấu trúc dữ liệu danh sách tuyến tính , liên kết
- Các phép toán thêm, loại bỏ, tìm kiếm trên danh sách

Q&A

Tiếp theo …

- Cấu trúc tập hợp