BÀI 2

DANH SÁCH

TÌNH HUỐNG DẪN NHẬP

Danh sách lớp lưu các thông tin của học viên như: họ tên, ngày tháng năm sinh, quê quán, giới tính, chỗ ở hiện nay… nhằm phục vụ trong công tác quản lí học viên;

Khi thêm học viên nào vào lớp hay có học viên chuyển sang lớp khác thì danh sách học viên này phải thay đổi bằng cách bổ sung hay xóa bớt đi thông tin về học viên.

DANH SÁCH HỌC VIÊN LỚP TIN HỌC - TNU

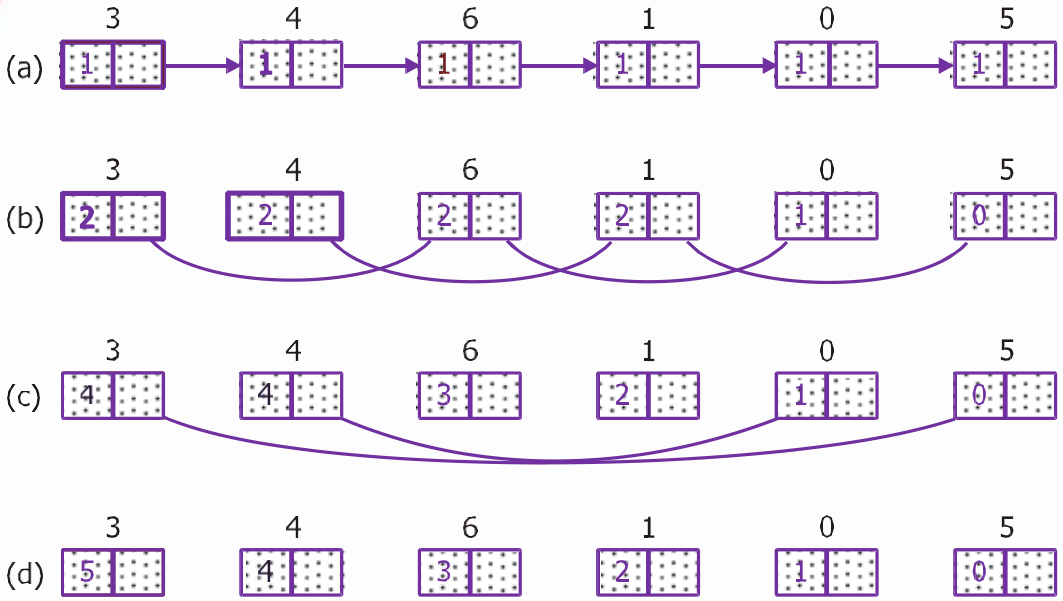
TT	Họ và tên	Ngày sinh	Giới tính	Quê quán
1	Đinh Mạnh Ninh	01/02/1990	Nam	Hà Nội
2	Trần Hà	01/01/1980	Nữ	Tuyên Quang
3	Hồ Ngọc Hà	01/02/1986	Nữ	Quảng Bình
4	Hồ Quỳnh Hương	01/05/1980	Nữ	Quảng Ninh
5	Tiêu Lam Trường	01/01/1978	Nam	Lâm Đồng

“Cách giải quyết bài toán này như thế nào?”

MỤC TIÊU

- Khái niệm về danh sách và danh sách liên kết, phân biệt được các dạng của danh sách liên kết;
- Các thao tác trên danh sách liên kết;
- Cách cài đặt các thao tác trên danh sách và danh sách liên kết;
- Vận dụng cấu trúc dữ liệu của danh sách vào giải quyết các bài toán thực tế.

NỘI DUNG



- Khái niệm danh sách và các thao tác trên danh sách;
- Biểu diễn danh sách bằng mảng;
- Danh sách liên kết;
- Ứng dụng danh sách.

1. KHÁI NIỆM DANH SÁCH VÀ CÁC THAO TÁC TRÊN DANH SÁCH

1. KHÁI NIỆM DANH SÁCH VÀ CÁC THAO TÁC TRÊN DANH SÁCH

Khái niệm:

- Danh sách là một cấu trúc dữ liệu gồm một hữu hạn các phần tử có kiểu dữ liệu xác định và giữa các phần tử có mỗi liên hệ với nhau;
- Trên phương diện toán học, danh sách là tập hợp hữu hạn các phần tử, có thứ tự và giữa chúng có mối liên hệ tuyến tính.

1. KHÁI NIỆM DANH SÁCH VÀ CÁC THAO TÁC TRÊN DANH SÁCH

Các thao tác trên danh sách:

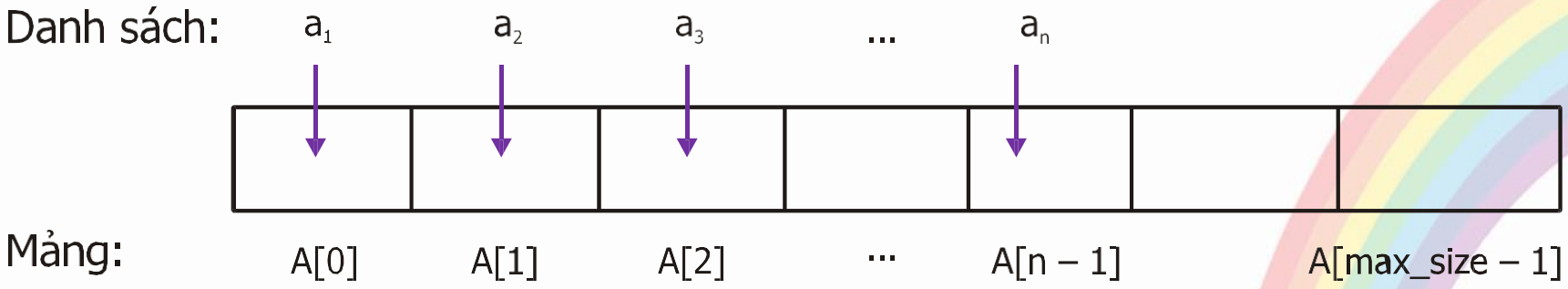
- Tạo mới một danh sách;
- Thêm một phần tử vào danh sách;
- Loại bỏ bớt một phần tử ra khỏi danh sách;
- Tìm kiếm một phần tử trong danh sách;
- Cập nhật (sửa đổi) giá trị cho một phần tử trong danh sách;
- Sắp xếp thứ tự các phần tử trong danh sách;
- Tách một danh sách thành nhiều danh sách;
- Nhập nhiều danh sách thành một danh sách;
- Sao chép một danh sách, hủy danh sách.

2. BIỂU DIỄN DANH SÁCH BẰNG MẢNG

2. BIỂU DIỄN DANH SÁCH BẰNG MẢNG

- Phương pháp tự nhiên nhất để cài đặt một danh sách là sử dụng mảng;
- Trong đó mỗi thành phần của mảng sẽ lưu giữ một phần tử nào đó của danh sách;
- Các phần tử kế nhau của danh sách được lưu giữ trong các thành phần kế nhau của mảng.

Việc sử dụng mảng để biểu diễn danh sách được thể hiện bằng hình dưới đây:



2. BIỂU DIỄN DANH SÁCH BẰNG MẢNG

Các khai báo biểu diễn danh sách bằng mảng:

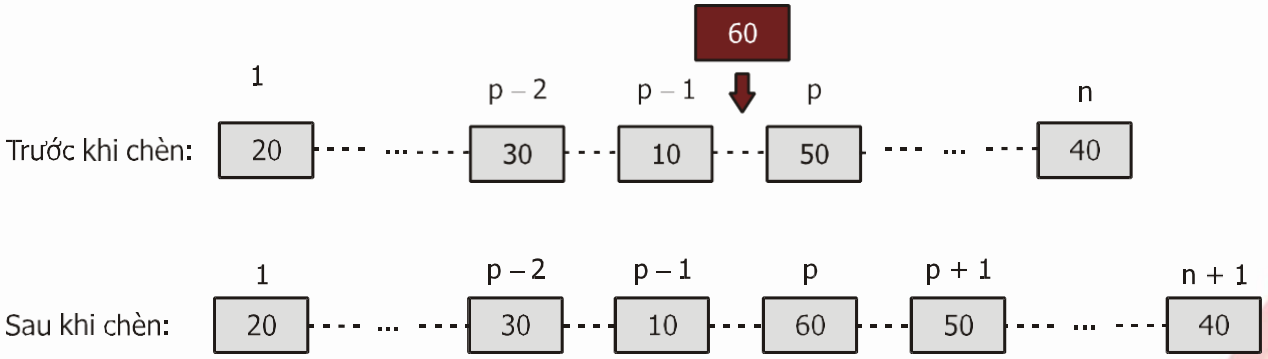
```
typedef      Kieu_du_lieu E_Type;
struct ListType
{
    E_Type Element[Max_Size];
    int Size;
} List;
```

- Để khởi tạo một danh sách rỗng: List.Size == 0;
- Độ dài của danh sách là List.Size.
- Danh sách đầy, nếu List.Size = Max_Size.

2. BIỂU DIỄN DANH SÁCH BẰNG MẢNG

Ví dụ 2.1

Chèn thêm một phần tử có giá trị 60 vào vị trí sau phần tử có giá trị 10 của danh sách.



2. BIỂU DIỄN DANH SÁCH BẰNG MẢNG

Ví dụ 2.1

Chèn thêm một phần tử có giá trị 60 vào vị trí sau phần tử có giá trị 10 của danh sách.

```
ListType Insert_E(int Pos,E_Type Item,ListType List)
{       int k,j;
        if (List.Size==Max_Size)
                printf("danh sach day khong the chen them");
        else    {     for (k = List.Size - 1; k >= Pos - 2;k --)
                        { j = k + 1;
                                if (k==Pos-2)List.Element[j]=Item;
                                else
 List.Element[j]=List.Element[k];
                        } List.Size=List.Size+1;
                }
         return List;
}
```

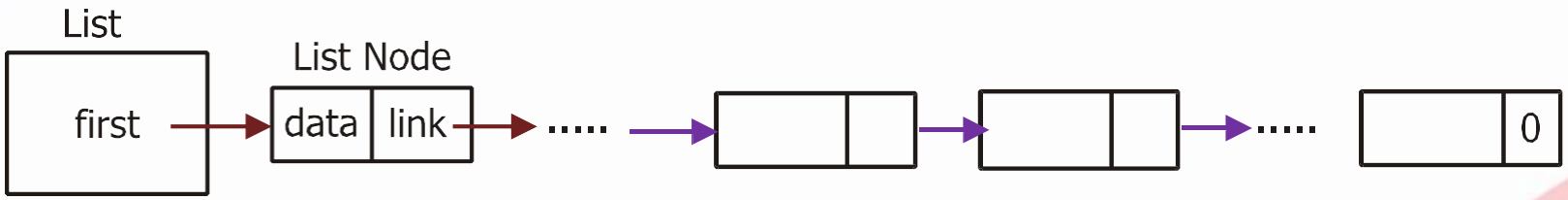
2. BIỂU DIỄN DANH SÁCH BẰNG MẢNG

- **Ưu điểm:** Truy cập trực tiếp đến phần tử ở vị trí bất kỳ trong danh sách một cách dễ dàng.
- **Nhược điểm:** Danh sách không thể phát triển quá cỡ của mảng, phép toán chèn vào sẽ không được thực hiện khi mảng đã đầy.

3. DANH SÁCH LIÊN KẾT

3. DANH SÁCH LIÊN KẾT

Khái niệm: Danh sách liên kết là tập hợp các phần tử gọi là nút (node), mỗi nút chứa dữ liệu về phần tử của danh sách và con trỏ chỉ vị trí của nút chứa phần tử khác trong danh sách. Con trỏ là địa chỉ trong bộ nhớ về nút tiếp theo của danh sách.



Trong đó:

- Phần **Data**: Chứa thông tin về phần tử của danh sách;
- Phần **Link**: Chứa địa chỉ về một nút khác trong danh sách;
- Riêng nút cuối cùng chứa 1 "địa chỉ đặc biệt", mang tính chất qui ước, dùng để đánh dấu nút kết thúc danh sách ta gọi nó là "địa chỉ null" hay "con trỏ rỗng" (null pointer).

3. DANH SÁCH LIÊN KẾT

Ví dụ 2.2

Ta có một danh sách tên các sinh viên bất kỳ.

Bây giờ ta muốn đưa ra danh sách theo thứ tự "từ điển" ta có thể tổ chức móc nối như sau:

3. DANH SÁCH LIÊN KẾT

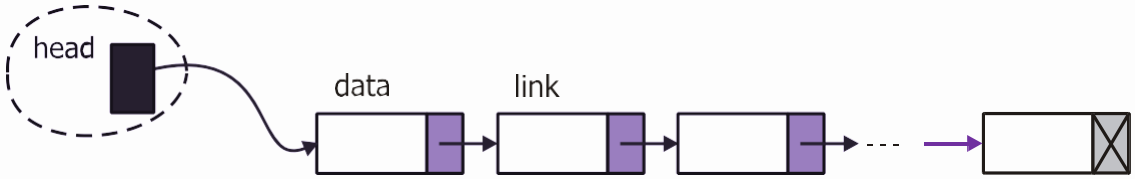
Chú ý: Không nên nhầm lẫn danh sách và danh sách liên kết.

- Danh sách là một mô hình dữ liệu, nó có thể được cài đặt bởi các cấu trúc dữ liệu khác nhau;
- Còn danh sách liên kết là một cấu trúc dữ liệu, ở đây nó được sử dụng để biểu diễn danh sách;
- Do vậy các thao tác trên danh sách đều có thể thực hiện được trên danh sách liên kết.

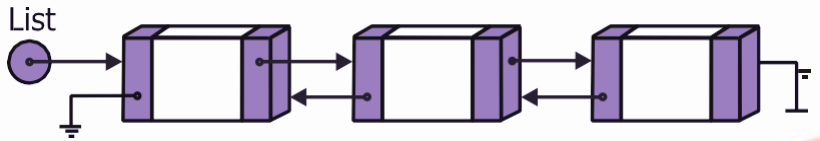
3. DANH SÁCH LIÊN KẾT

Danh sách liên kết có thể chia thành nhiều loại:

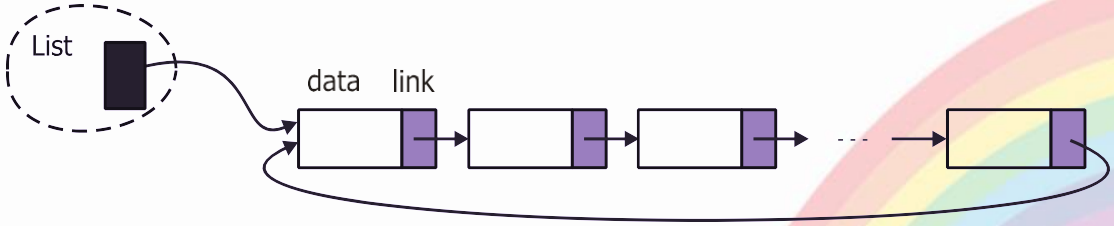
- Danh sách liên kết đơn



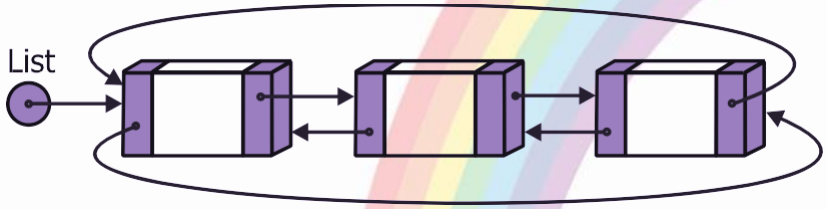
- Danh sách liên kết đôi/kép



- Danh sách liên kết vòng đơn



- Danh sách liên kết vòng kép



3. DANH SÁCH LIÊN KẾT

- Trong danh sách liên kết đơn, mỗi nút gồm 2 thành phần:
  - **Trường dữ liệu:** Chứa một phần tử của danh sách;
  - **Trường liên kết:** Chỉ đến nút tiếp theo trong danh sách hoặc là con trỏ rỗng.
- Cấu trúc của danh sách liên kết được khai báo như sau:

```
typedef Kieu_du_lieu ElementType;
        struct NodeType
                {
                    ElementType Data;
                    struct NodeType *Link;
                }
        NodeType *List;
```

3. DANH SÁCH LIÊN KẾT

Hàm thực hiện thao tác kiểm tra danh sách liên kết đơn có rỗng hay không được cài đặt như sau:

```
int Empty(NodeType *List)
        {
                if (List == NULL)
                        return 1;
                 else
                        return 0;
        }
```

3. DANH SÁCH LIÊN KẾT

Thao tác duyệt các phần tử trên danh sách liên kết đơn:

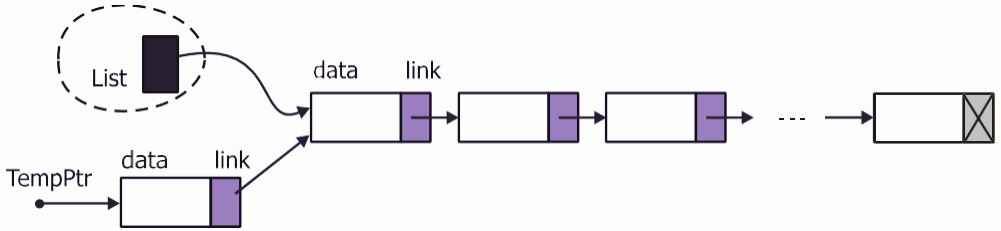
```
void Traverse( NodeType *List)
    {    int i=1;  NodeType
         *Currp;  Currp = List;
         while (Currp->Link!=NULL)
           {        printf("&f",Node[i].Data);
           Currp =Currp->Link;
                    i++;
           }
    }
```

3. DANH SÁCH LIÊN KẾT

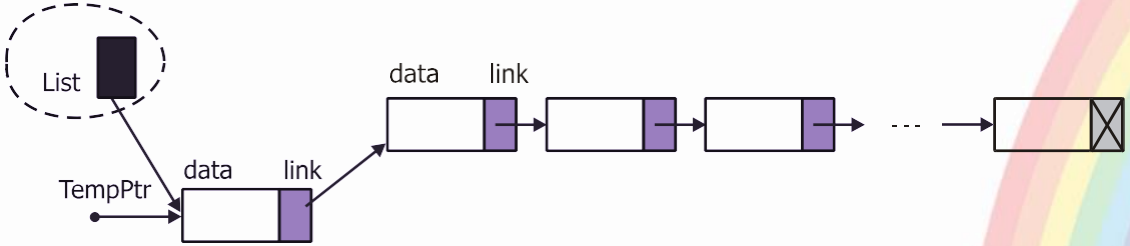
Xây dựng thao tác thêm một nút mới vào danh sách liên kết đơn:

- Tạo được một nút mới chứa phần tử cần chèn và ta truy cập được tới nút này;
- Thêm một nút mới vào đầu của một danh sách liên kết đơn được thực hiện như sau:

➢ **Bước 1**: Đặt Link(TempPtr)bằng List



➢ **Bước 2**: Đặt Listbằng TempPtr

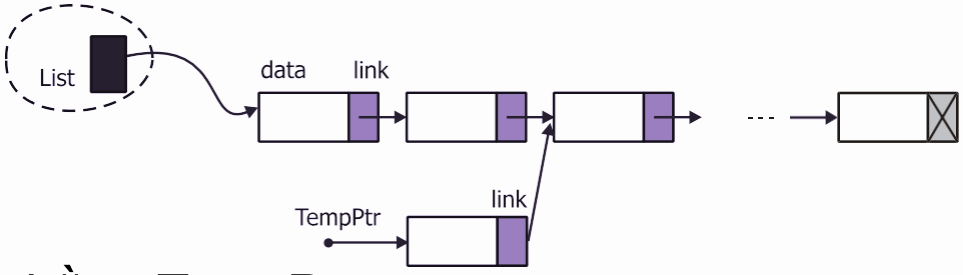


3. DANH SÁCH LIÊN KẾT

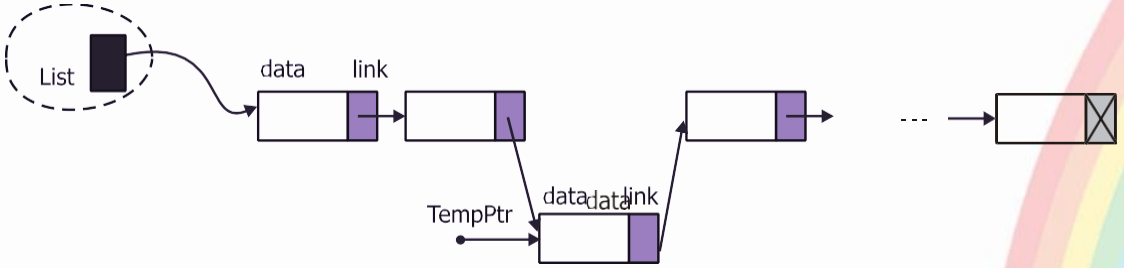
Xây dựng thao tác thêm một nút mới vào danh sách liên kết đơn:

- Tạo được một nút mới chứa phần tử cần chèn và ta truy cập được tới nút này;
- Khi chèn nút mới vào sau một nút bất kỳ trong danh sách liên kết đơn:

➢ **Bước 1**: Đặt Link(TempPtr)bằng List



➢ **Bước 2**: Đặt Listbằng TempPtr



3. DANH SÁCH LIÊN KẾT

Thuật toán xóa một nút trong danh sách liên kết đơn:

```
if (danh sách rỗng)
//Đưa ra thông báo về danh sách rỗng
else
{
//Bước 1:
        if      PrevPtr == NULL // Xóa nút đầu tiên
                {               TempPtr = List;
                                List = TempPtr->Link;
                }
    else // Xóa một nút có nút đứng trước nó
                {               TempPtr = PrevPtr-> Link;
                                PrevPtr->Link=TempPtr->Link;
                }
 //Bước 2: Tiến hành xóa nút
        free(TempPtr));;
}
```

3. DANH SÁCH LIÊN KẾT

Thuật toán thêm một nút mới vào một danh sách liên kết:

```
void Insert(NodeType * List, ElementType Item, NodeType
*PrevPtr)
{
        NodeType *TempPtr;
        TempPtr=GetNode(Item);
        //tạo nút mới để chèn vào danh sách
            if (List == NULL)
              {
                    List = TempPtr;
                    TempPtr -> Link = NULL;
              }
            else
              {
                    TempPtr -> Link = PrevPtr -> Link;
                     PrevPtr -> Link = TempPtr;
              }

}
```

3. DANH SÁCH LIÊN KẾT

Thủ tục cài đặt thuật toán này để thực hiện thao tác xóa một nút trong danh sách liên kết đơn trên cơ sở mảng được xây dựng như sau:

```
void Deletefromlist(NodeType *List,NodeType *PrevPtr)
    {
        NodeType * TempPtr;
        if (List==NULL)              printf("danh sach rong");
        else
        {           if      (PrevPtr == NULL)
                                    {
                                    TempPtr = List;
                                    List = TempPtr->Link;
                            }
                    else
                            {
                                    TempPtr = PrevPtr-> Link;
                                    PrevPtr->Link=TempPtr-Link;
                            }
                    DeleteNode(TempPtr);
        }
    }
```

3. DANH SÁCH LIÊN KẾT

Thuật toán tìm kiếm tuyến tính trong một danh sách liên kết đơn:

```
//Bước 1: Đặt
Currptr = List; PrevPtr = NULL; Found =0;
//Bước 2:
while ((Found == 0) && (CurrPtr != NULL))
      { if      (CurrPtr -> Data = Item) Found = 0;
          else
          {
                    PrevPtr = CurrPtr;
                    CurrPtr = CurrPtr->Link;
          }
    }
```

3. DANH SÁCH LIÊN KẾT

Thuật toán tìm kiếm tuyến tính trong một danh sách liên kết đơn:

```
int SearchOfList(NodeType*List,ElementType Item)
    {
     int Found;
     Found = 0;
     NodeType*PrevPtr; NodeType *CurrPtr;  CurrPtr
     = List; PrevPtr = NULL;
     while ((Found == 0)&&(CurrPtr != NULL))
        {
           if      (CurrPtr -> Data = Item) Found = 1;
           else
                        {
                                    PrevPtr = CurrPtr;
                                    CurrPtr = CurrPtr -> Link;
                        }
        } return Found;
     }
```

4. ỨNG DỤNG DANH SÁCH

4. ỨNG DỤNG DANH SÁCH

Các phép tính số học trên đa thức:

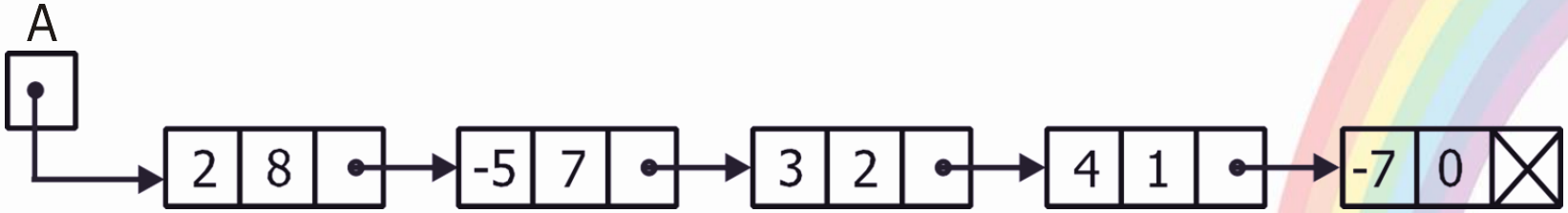
- Trong mục này ta sẽ xét các phép tính số học (cộng, trừ, nhân, chia) đa thức một ẩn;
- Các đa thức một ẩn là các biểu thức có dạng:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

- Đa thức biểu diễn dưới dạng danh sách liên kết đơn mà mỗi nút của nó có quy cách như sau: Mỗi nút có 3 trường:
  - Trường COEF chứa hệ số khác không của mỗi số hạng trong đa thức;
  - Trường EXP chứa số mũ tương ứng;
  - Trường LINK chứa địa chỉ nút tiếp theo trong danh sách.

A là con trỏ, trỏ tới nút đầu tiên của danh sách.

Ví dụ: $A(x) = 2x^8 - 5x^7 + 3x^2 + 4x - 7$



4. ỨNG DỤNG DANH SÁCH

Giải thuật thực hiện cộng 2 đa thức A(x) và B(x):

- Giả sử rằng danh sách liên kết biểu diễn chúng đã được tạo lập và đã được trỏ lần lượt bởi biến trỏ A và B;
- Ta sẽ gọi C là con trỏ, trỏ tới danh sách liên kết biểu diễn đa thức tổng;
- Dùng 2 biến trỏ p và q để thăm lần lượt các nút, khi duyệt qua 2 danh sách.

4. ỨNG DỤNG DANH SÁCH

Khi đó ta thấy có những tình huống như sau :

- Nếu EXP(p) = EXP(q) ta sẽ phải thực hiện cộng giá trị ở trường COEF của 2 nút đó. Nếu giá trị tổng khác không thì phải tạo ra nút mới để biểu diễn số hạng tương ứng và bổ sung vào (gọi tắt là "gắn vào") danh sách tổng;
- Nếu EXP (p) > EXP (q), nghĩa là trong danh sách B không có số hạng cùng số mũ với p như trong danh sách A (Ngược lại EXP(q) > EXP (p) thì xử lí cũng tương tự). Như vậy sẽ phải sao chép thông tin ở nút p vào 1 nút mới và gắn vào danh sách tổng;
- Nếu 1 trong 2 danh sách kết thúc trước thì các nút còn lại của danh sách kia sẽ được sao chép lần lượt vào nút mới và "gắn vào" danh sách tổng. Mỗi lần 1 nút mới được tạo ra thì nó được gắn vào sau nút cuối cùng của danh sách tổng (ta gọi tắt là nút "đuôi" của danh sách tổng). Như vậy, phải thường xuyên nắm được địa chỉ của nút đuôi này, ta gọi đó là d.

TÓM LƯỢC CUỐI BÀI

Sau khi học xong bài học này, học viên có thể:

- Nắm được khái niệm về danh sách, các thao tác trên danh sách;
- Nắm được cách biểu diễn danh sách bằng mảng;
- Nắm được khái niệm danh sách liên kết, các dạng danh sách liên kết;
- Nắm được cách khai báo, cài đặt các thao tác trên danh sách liên kết.