

BÀI THỰC HÀNH SỐ 4: DANH SÁCH LIÊN KẾT (tt)

(Số tiết: 3)

Mục đích :

1. Áp dụng cấu trúc dữ liệu danh sách liên kết vào việc giải quyết một số bài toán đơn giản.
2. Sắp xếp các phần tử trong danh sách.

Vấn đề 1: Áp dụng cấu trúc dữ liệu danh sách liên kết vào việc giải quyết một số bài toán đơn giản. [*Bài toán cộng trừ hai đa thức*].

Xét đa thức tổng quát: $P(x) = a_n x^n + a_{n-1} x^{n-1} + a_1 x^2 + \dots + a_0$

Cách đơn giản nhất để biểu diễn đa thức là dùng mảng lưu các hệ số của đa thức. Các phép toán như: cộng, trừ, nhân 2 đa thức,.. sẽ được thực hiện một cách dễ dàng nếu biểu diễn đa thức bằng mảng.

Giả sử ta có hai đa thức:

$$P1(x) = 3x^7 + 5x^6 + x^5 + 2x^3 - 7x + 9$$

$$P2(x) = 2x^7 + 3x^5 - 5x^4 + 2x^3 + x - 8$$

Cộng hai đa thức P1 và P2 ta được đa thức tổng:

$$T(x) = 5x^7 + 5x^6 + 4x^5 - 5x^4 + 4x^3 - 6x + 1$$

Có thể dùng hai mảng A, B và C để lưu hai đa thức P1, P2 và T như sau:

A:

3	5	1	0	2	0	-7	9
0	1	2	3	4	5	6	7

B:

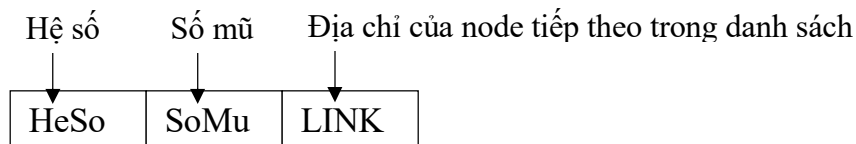
2	0	3	-5	2	0	1	-8
0	1	2	3	4	5	6	7

C:

5	5	4	5	4	0	-6	1
0	1	2	3	4	5	6	7

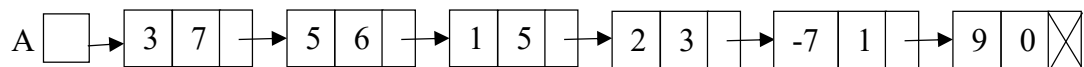
Tuy nhiên, biểu diễn đa thức bằng mảng có một hạn chế lớn là: khi đa thức có nhiều hệ số bằng 0 thì việc biểu diễn bằng mảng gây lãng phí bộ nhớ. Ví dụ: đa thức $x^{2017} + 1$ đòi hỏi 1 mảng 2018 phần tử.

Để khắc phục các nhược điểm ở trên, ta có thể dùng danh sách liên kết đơn để biểu diễn đa thức. Mỗi node của DSLK sẽ chứa các thông tin sau:



Như vậy, với đa thức $P1(x)$ ở trên thì danh sách biểu diễn nó có dạng:

$$P1(x) = 3x^7 + 5x^6 + x^5 + 2x^3 - 7x + 9$$



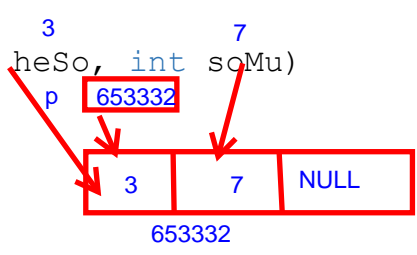
Ở đây A là con trỏ lưu địa chỉ nút đầu tiên của danh sách.

1. Định nghĩa cấu trúc **Node** với mỗi phần tử của Node gồm có 3 thành phần: hệ số, số mũ và địa chỉ của Node tiếp theo.

```
Struct Node{
    float heSo;
    int soMu;
    Node *link;
};
Struct List
{
    Node *first, *last;
};
```

2. Hàm thêm một phần tử vào danh sách

```
//Khởi tạo danh sách rỗng
Void init(List &L)
{
    L.first = L.last = NULL;
}
//Tạo một node mới
Node *GetNode(float heSo, int soMu)
{
    Node *p;
    p = new Node;
    if(p == NULL)
        return NULL;
    p->heSo = heSo;
    p->soMu = soMu;
    p->link = NULL;
    return p;
}
// Gắn node new_ele vào danh sách
void AddLast (List&L, Node *new_ele)
{
    if(L.first == NULL) //danh sách rỗng
    {
        L.first = L.last = new_ele;
    }
    else
    {
        L.last->link = new_ele;
        L.last = new_ele;
    }
}
//Thêm một node với dữ liệu là heSo và soMu vào danh sách
Void InsertLast (List&L, float heSo, int soMu)
{
    Node *new_ele=GetNode(heSo, soMu);
    if(new_ele==NULL)
        return;
    AddLast(L,new_ele);
}
```



The diagram shows a pointer variable `p` pointing to a newly created node. The node is represented as a structure with three fields: `heSo` (containing the value 3), `soMu` (containing the value 7), and `link` (set to NULL). The memory address 653332 is indicated next to the node structure.

3. Hàm nhập đa thức (hệ số và số mũ) bằng cách thêm vào cuối danh sách.

```
// Hàm nhập đa thức
void NhậpDaThuc(List&L)
{
    float heSo;
    int soMu;
    printf("\nBat dau nhap da thuc (nhap he so 0 de
ket thuc): \n");
    do
    {
        printf("\nNhap he so: ");
        scanf("%f", &heSo);
        if (heSo == 0)
            break;
        printf("\nNhap so mu:");
        scanf("%d", &soMu);
        InsertLast(L, heSo, soMu);
    } while (heSo != 0);
    printf("\nDa nhap da thuc xong: \n");
}
```

4. Xuất danh sách biểu diễn đa thức

```
void XuatDanhSach(List L)
{
    Node *p;
    p = L.first;
    printf("\n");
    while (p!= NULL)
    {
        printf("%.0f, %d      ", p->heSo, p->soMu);
        p = p->link;
    }
}
```

5. Cộng hai đa thức

```
//Cộng đa thức: d3 = d2 + d1
void CongDaThuc(List d1, List d2, List&d3)
{
    init(d3);
    Node *p = d1.first, *q = d2.first;
    float tongHeSo;
    while(p && q)
    {
        if(p->soMu == q->soMu) //Hai số mũ bằng nhau
        {
            tongHeSo = p->heSo + q->heSo;
            if(tongHeSo != 0)
                InsertLast(d3, tongHeSo, p->soMu);
            p = p->link;
            q = q->link;
        }
        else
        {
            if(p->soMu > q->soMu)
            {
                InsertLast(d3, p->heSo, p->soMu);
                p = p->link;
            }
            else
            {
                if(p->soMu < q->soMu)
                {
                    InsertLast(d3, q->heSo, q->soMu);
                    q = q->link;
                }
            }
        }
    }
    while(q) //biểu thức d1 kết thúc trước
    {
        InsertLast(d3, q->heSo, q->soMu);
        q = q->link;
    }
    while(p) //biểu thức d2 kết thúc trước
    {
        InsertLast(d3, p->heSo, p->soMu);
        p = p->link;
    }
}
```

6. Chương trình chính

```
int main()
{
    List d1, d2, d3;
    init(d1);
    init(d2);
    init(d3);
    NhapDaThuc(d1);
    printf("\nDanh sach bieu dien da thuc d1: ");
    XuatDanhSach(d1);
    NhapDaThuc(d2);
    printf("\nDanh sach bieu dien da thuc d2: ");
    XuatDanhSach(d2);
    CongDaThuc(d1, d2, d3);
    printf("\nDanh sach bieu dien đa thuc tong: ");
    XuatDanhSach(d3);
}
```

7. Chạy chương trình, nhập vào hai đa thức P1 và P2.

$$P1(x) = 3x^7 + 5x^6 + x^5 + 2x^3 - 7x + 9$$

$$P2(x) = 2x^7 + 3x^5 - 5x^4 + 2x^3 + x - 8$$

Ghi kết quả xuất ra màn hình ở đây:

.....

.....

.....

.....

Tương tự, hãy ghi kết quả xuất ra màn hình khi nhập hai biểu thức P3 và P4.

$$P3(x) = 5x^{15} - 3x^{12} - 2x^{10} + 5x^2$$

$$P4(x) = 2x^{30} + 6x^{21} - 4x^{10} + 5x - 2$$

.....

.....

.....

.....

Vấn đề 2: Sắp xếp các phần tử trong danh sách.

Với chương trình ở trên, nếu ta nhập vào hai đa thức P5 và P6 (theo thứ tự các hệ số và số mũ như bên dưới) thì kết quả tính tổng hai đa thức đúng hay sai?

$$P5(x) = 3x^7 - x^{10} + 2x^3 + 9x^5$$

$$P6(x) = 7x^5 + 6x^7 - 4x^{10} + x + 6$$

Câu trả lời là SAI. Vì với phương pháp cộng hai đa thức như trên thì DSLK biểu diễn đa thức phải được sắp xếp giảm dần theo số mũ. Nếu người dùng nhập không vào không đúng thứ tự thì ta cần tiến hành sắp xếp DSLK giảm dần theo số mũ trước khi thực hiện cộng hai đa thức.

Sắp xếp danh sách liên kết đơn có 2 phương pháp tiếp cận: Hoán vị dữ liệu của các phần tử trong danh sách, hoặc thay đổi các mối liên kết. Trong trường hợp này, mỗi node của danh sách chỉ có hai thành phần là hệ số và số mũ, kích thước của dữ liệu nhỏ, vì thế ta có thể chọn cách sắp xếp DSLK bằng cách hoán vị dữ liệu của các phần tử trong danh sách. Hàm bên dưới thực hiện sắp xếp bằng phương pháp đổi chỗ trực tiếp.

Hàm sắp xếp danh sách bằng phương pháp đổi chỗ trực tiếp

```
void InterchangeSort ( List&L)
{
    for ( Node* p=L.first ; p!=L.last ; p=p->link )
        for ( Node* q=p->link ; q!=NULL ; q=q->link )
            if ( p->soMu > q->soMu )
            {
                Swap1(p->heSo, q->heSo); //Hoán vị 2 số thực
                Swap2(p->soMu, q->soMu); //HV2 số nguyên
            }
}
```

BÀI TẬP VỀ NHÀ: (bắt buộc – sinh viên nộp vào đầu buổi thực hành sau)

Hãy viết hàm xuất đa thức dưới dạng như sau: Giả sử có đa thức là $6x^7 - 3x^5 + 9x^2 - 1$ thì sẽ xuất dưới dạng $6*x^7 - 3*x^5 + 9*x^2 - 1$.

BÀI TẬP LÀM THÊM: (sinh viên có thể nộp bài vào đầu buổi thực hành sau để lấy điểm cộng)

Viết tiếp chương trình để thực hiện việc :

1. Trừ hai đa thức.
2. Nhân hai đa thức
3. Chia hai đa thức