CHƯƠNG 4: DANH SÁCH LIÊN KẾT

Mục tiêu của chương

Nắm vững:

- Cách khai báo danh sách liên kết như: danh sách liên kết đơn**,** danh sách liên kết kép, danh sách liên kết vòng và danh sách liên kết đôi vòng .

- Cách thao tác trên các danh sách liên kết như: tạo node, chèn node, thêm node, loại node, sửa đổi thông tin node và hiển thị thông tin trên danh sách.

- Các giải thuật cơ bản trên danh sách liên kết.

- Vận dụng lý thuyết giải các bài tập danh sách liên kết.

Nội dung của chương

Nghiên cứu các bước thực hiện các thuật toán trên các danh sách liên kết đơn, danh sách liên kết kép, danh sách liên kết vòng và danh sách liên kết đôi vòng.

4.1. Giới thiệu

Danh sách liên kết (Linked List là một cấu trúc dữ liệu có kiểu truy cập tuần tự. Mỗi phần tử trong danh sách liên kết có chứa thông tin về phần tử tiếp theo, qua đó ta có thể truy cập tới phần tử này.

Danh sách liên kết là một cấu trúc dữ liệu bao gồm một tập các phần tử, trong đó mỗi phần tử là một phần của một nút có chứa một liên kết tới nút kế tiếp. Mỗi phần tử là một phần của một nút bởi vì mỗi nút ngoài việc chứa thông tin về phần tử còn chứa thông tin về liên kết tới nút tiếp theo trong danh sách. Có thể nói danh sách liên kết là một cấu trúc dữ liệu được định nghĩa kiểu đệ quy , vì trong định nghĩa một nút của danh sách có tham chiếu tới khái niệm nút. Thông thường, một nút thường có liên kết trỏ tới một nút khác, tuy nhiên nó cũng có thể trỏ tới chính nó.

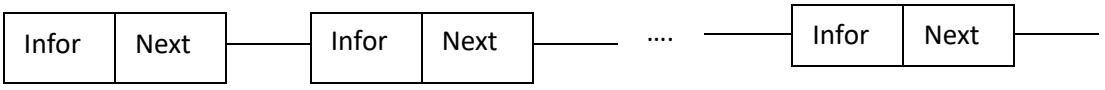
4.2. Danh sách liên kết đơn

4.2.1. Mô tả

Định nghĩa danh sách liên kết đơn: tập hợp các node thông tin (khối dữ liệu) được tổ chức rời rạc trong bộ nhớ. Trong đó, mỗi node gồm hai thành phần:

• Thành phần dữ liệu (infor): dùng để lưu trữ thông tin của node.

• Thành phần con trỏ (pointer): dùng để trỏ đến node dữ liệu tiếp theo.



Hình 4.1: Danh sách liên kết đơn

4.2.2. Khai báo

Sử dụng kiểu dữ liệu cấu trúc tự trỏ để định nghĩa mỗi node của danh sách liên kết đơn. Giả sử thành phần thông tin của mỗi node được định nghĩa như một cấu trúc Item:

typedef struct {

<Kiểu 1> <Thành viên 1>;

<Kiểu 2> <Thành viên 2>;

……

<Kiểu N> <Thành viên N>;

} Item;

Khi đó, mỗi con trỏ đến một node được định nghĩa như sau:

typedef struct node {

tem Infor; //Thông tin của mỗi node;

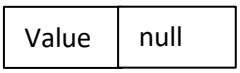
struct node *next;

} *List;

4.2.3. Các thao tác trên danh sách liên kết đơn (DSLKĐ)

4.2.1.1. Khởi tạo một node cho DSLKĐ:

node temp



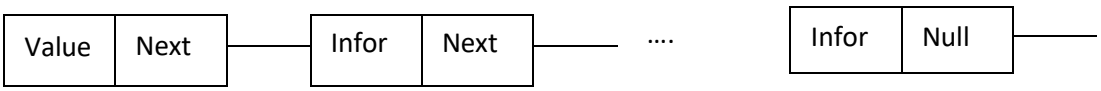
Hình 4.2: Cấu tạo một nút temp

node *Tao_nut_dslkdon(int value){

```
struct node *temp, *s; //Khai báo hai con trỏ node *temp, *s
temp = new(struct node); //Cấp phát miền nhớ cho temp
if (temp == NULL){ //Nếu không đủ không gian nhớ
    cout<<“Không đủ bộ nhớ để cấp phát"<<endl;
    return 0;
 }
 else{
    temp->info = value;//Thiết lập thông tin cho node temp
    temp->next = NULL; //Thiết lập liên kết cho node temp
    return temp;//Trả lại node temp đã được thiết lập.
     }
 }
```

4.2.1.2. Chèn node vào đầu DSLKĐ:

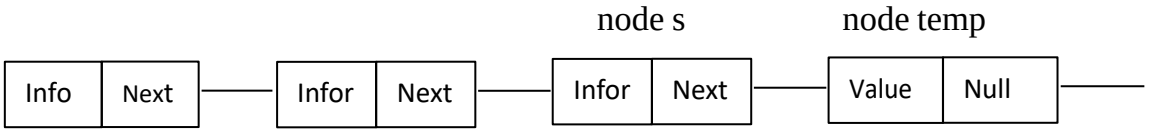
start->next=p start=temp



Hình 4.3: Chèn node vào đầu DSLKĐ

```
void  Chen_dau() {//Chèn node vào đầu DSLKĐ
    int value;
    cout<<“Nhập giá trị node:"; cin>>value; //Giá trị node cần chèn struct node
    *temp, *p; //Sử dụng hai con trỏ temp và p
    temp = create_node(value);/ /Tạo một node với giá trị value
        if (start == NULL){ //Nếu danh sách rỗng
            start = temp; //Danh sách chính là node temp
            start->next = NULL; //Không có liêt kết với node khác
        }
        else { //Nếu danh sách không rỗng
            p = start; //p trỏ đến node đầu của start
                start = temp; //start được trỏ đến temp
            start->next = p;//start trỏ tiếp đến gốc cũ
        }
        cout<<“Hoàn thành thêm node vào đầu DSLKĐ"<<endl;
}
```

4.2.1.3. Thêm node vào cuối DSLKĐ:

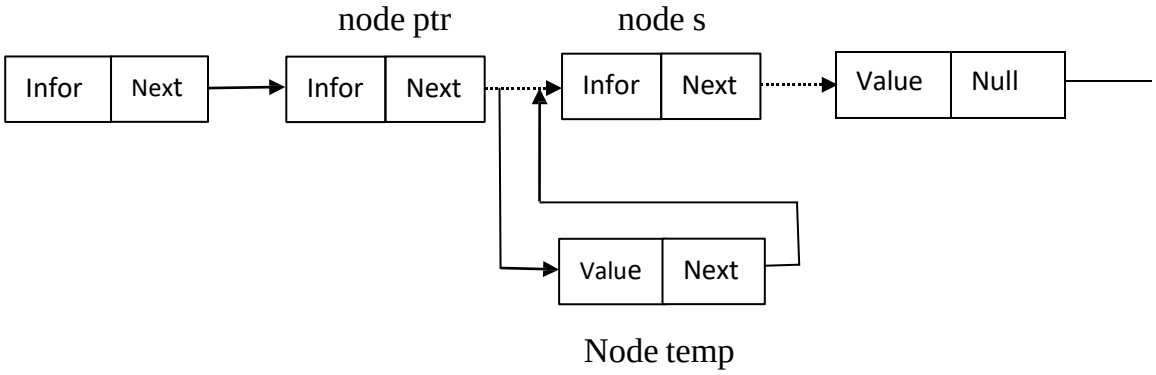


Hình 4.4: Thêm node vào cuối DSLKĐ

```
void Themnut_cuoi(){//Thêm node vào cuối DSLKĐ
    int value;
    cout<<“Nhập giá trị cho node: "; cin>>value; //Nhập giá trị node struct node
    *temp, *s; //Sử dung hai con trỏ temp và s
    temp = create_node(value);//Tạo node có giá trị value
    s = start; //s trỏ đến node đầu danh sách
    while (s->next != NULL){ //Di chuyển s đến node cuối cùng
         s = s->next;
    }
    temp->next = NULL; //temp không chỏ đi đâu nữa
    s->next = temp; //Thiết lập liên kết cho s
    cout<<“Hoàn thành thêm node vào cuối"<<endl;
```

}

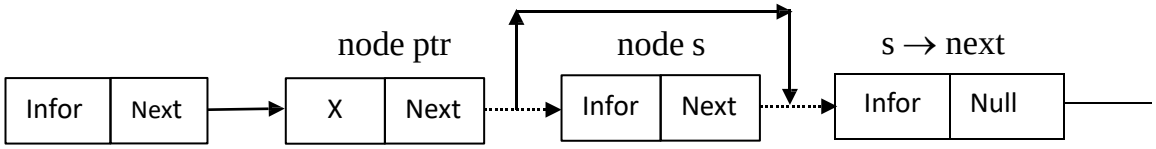
4.2.1.4. Thêm node vị trí pos:



Hình 4.5: Thêm node vị trí pos trong DSLKĐ

```
void Themnut_pos(){//Thêm node vào vị trí pos
    int value, pos, counter = 0;
    cout<<"Nhap gia tri node:"; cin>>value;
    struct node *temp, *s, *ptr;
    temp = create_node(value);//Tạo node
    cout<<“Nhập vị trí node cần thêm: "; cin>>pos;
    int i; s = start; //s trỏ đến node đầu tiên
    while (s != NULL){ //Đếm số node của DSLKĐ
            s = s->next; counter++;
    }
    if (pos == 1){ //Nếu pos là vị trí đầu tiên
            if (start == NULL){ //Trường hợp DSLKĐ rỗng
                    start = temp; start->next = NULL;
            }
            else { ptr = start; start = temp; start->next = ptr;  }
    }
    else if (pos > 1 && pos <= counter){ //Trường hợp pos hợp lệ
            s = start; //s trỏ đến node đầu tiên
            for (i = 1; i < pos; i++){ ptr = s; s = s->next; }
            ptr->next = temp; temp->next = s; //Thiết lập LK cho node
    }
            else { cout<<“Vượt quá giới hạn DSLKĐ"<<endl;}
}
```

4.2.1.5. Loại node ở vị trí pos:



Hình 4.6: Xóa node vị trí pos trong DSLKĐ

```
void Xoa_pos(){//Loại phần tử ở vị trí cho trước
    int pos, i, counter = 0;
    if (start == NULL){ cout<<“Không thực hiện được"<<endl; return; }
    cout<<“Vị trí cần loại bỏ:"; cin>>pos;
    struct node *s, *ptr;
    s = start; //s trỏ đến đầu danh sách
    if (pos == 1){//Nếu vị trí loại bỏ là node đầu tiên
            start = s->next; s->next=NULL; free(s);
    }
    else {
        while (s != NULL) { s = s->next; counter++; } //Đếm số node
        if (pos > 0 && pos <= counter){ //Nếu vị trí hợp lệ
        s = start;//s trỏ đến node đầu của danh sách
        for (i = 1;i < pos;i++){ ptr = s; s = s->next; }
        ptr->next = s->next; //Thiết lập liên kết cho node
        }
        else { cout<<"Vi tri ngoai danh sach"<<endl; }
        free(s);
        cout<<"Node da bi loai bo"<<endl;
        }
}
```

4.2.1.6. Sửa đổi nội dung node:

```
void Sua(){//Sửa đổi thông tin của node
    int value, pos, i;
    if (start == NULL){ //Nếu danh sách rỗng
        cout<<“Không thực hiện được"<<endl; return;
    }
    cout<<“Nhập vị trí node cần sửa:";cin>>pos;
```

```
    cout<<“Giá trị mới của node:";cin>>value;
    struct node *s, *ptr; //Sử dụng hai con trỏ s và ptr
    s = start; //s trỏ đến node đầu tiên
    if (pos == 1) { start->info = value;} //Sửa luôn node đầu tiên
    else { //Nếu không phải là node đầu tiên
        for (i = 0;i < pos - 1;i++){//Chuyển s đến vị trí pos-1
                if (s == NULL){//Nếu s là node cuối cùng
                        cout<<"Vị trí "<<pos<<" không hợp lệ“; return;
                }
                s = s->next;
        }
        s->info = value; //Sửa đổi thông tin cho node
    }
    cout<<“Hoàn thành việc sửa đổi"<<endl;
}
```

4.2.1.7. Tìm kiếm node trên DSLKĐ:

```
void Timkiem(){//Tìm kiếm node
    int value, pos = 0; bool flag = false;
    if (start == NULL){
        cout<<“Danh sách rỗng"<<endl;
        return;
    }
    cout<<“Nội dung node cần tìm:"; cin>>value;
    struct node *s; s = start; //s trỏ đến đầu danh sách
    while (s != NULL){ pos++;
        if (s->info == value){//Nếu s->infor là value
                flag = true;
                cout<<“Tìm thấy "<<value<<" tại vị trí "<<pos<<endl;
        }
        s = s->next;
    }
    if (!flag) {
        cout<<“Giá trị"<<value<<“không tồn tại"<<endl;
    }
```

```
}
```

4.2.1.8. Hiển thị nội dung DSLKĐ:

```
void Hienthi(){//Hiển thị nội dung DSLKĐ
    struct node *temp; //Sử dụng một con trỏ temp
    if (start == NULL){ // Nếu danh sách rỗng
        cout<<“Không có dữ liệu hiển thị"<<endl;
        return;
    }
    temp = start; //temp trỏ đến node đầu trong DSLKĐ
    cout<<“Nội dung DSLKĐ: "<<endl;
    while (temp != NULL) { //Lặp cho đến node cuối cùng
        cout<<temp->info<<"->"; //Hiển thị thành phần thông tin
        temp = temp->next; //Trỏ đến node kế tiếp
    }
    cout<<"NULL"<<endl; //Cuối cùng chắc chắn sẽ là NULL
}
```

4.2.1.9. Sắp xếp nội dung các node của DSLKĐ:

```
void Sapxep(){//Sắp xếp nội dung các node
    struct node *ptr, *s; //Sử dụng hai con trỏ ptr và s
    int value; //Giá trị trung gian
    if (start == NULL){//Nếu danh sách rỗng
        cout<<“Không có dữ liệu sắp xếp"<<endl;
        return;
    }
    ptr = start;//ptr trỏ đến node đầu danh sách
    while (ptr != NULL){ //Lặp nếu ptr khác rỗng
        for (s = ptr->next;s !=NULL;s = s->next){ //s là node kế tiếp
                if (ptr->info > s->info){
                        value = ptr->info; ptr-
                        >info = s->info; s-
                        >info = value;
                }
        }
        ptr = ptr->next;
```

```
    }
}
```

4.3. Danh sách liên kết đơn vòng (Circular single linked List)

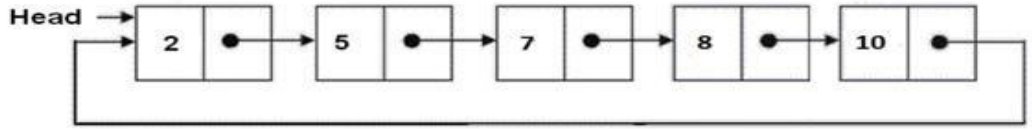
4.3.1. Mô tả

Định nghĩa danh sách liên kết đơn vòng là danh sách liên kết đơn trong đó tất cả các node liên kết với nhau thành một vòng tròn. Không có con trỏ NULL ở node cuối cùng mà được liên kết với node đầu tiên.

Một số tính chất của danh sách liên kết đơn vòng:

• Mọi node đều là node bắt đầu. Ta có thể duyệt tại bất kỳ node nào và chỉ dừng khi ta lặp lại một node đã duyệt.

• Dễ dàng phát triển các ứng dụng thực hiện vòng quang danh sách.



Hình 4.7: Danh sách liên kết đơn vòng

4.3.2. Khai báo

Khai báo danh sách liên kết vòng:

```
struct node {//Biểu diễn node của danh sách liên kết đơn vòng
        int info; //Thành phần thông tin của node
        struct node *next; //Thành phần con trỏ của node
}*last;
```

4.3.3. Các thao tác trên danh sách liên kết đơn vòng

4.3.3.1. Tạo node cho danh sách liên kết đơn vòng:

```
void Taonut_vong(int value){//Tạo danh sách liên kết vòng
    struct node *temp; //Khai báo con trỏ temp
    temp = new(struct node);//Cấp phát bộ nhớ cho con trỏ
    temp temp->info = value;//Thiết lập giá trị cho node temp
    if (last == NULL){//Nếu danh sách rỗng
    last = temp; //last chính là temp
    temp->next = last;//Temp trỏ vòng lại last
    }
    else  {//Nếu danh sách không rỗng
    temp->next = last->next; //Thiết lập liên kết cho temp  last-
    >next = temp; //Thiết lập liên kết cho last
    last = temp; //Thiết lập liên kết vòng cho last
```

```
    }
}
```

4.3.3.2. Chèn node vào đầu cho danh sách liên kết đơn vòng:

```
void Chendau_vong{//Chèn node vào đầu
    if (last == NULL){//Nếu danh sách rỗng
    cout<<“Chưa tạo node đầu cho danh sách"<<endl;
    return;
    }
    struct node *temp; //Khai bào con trỏ temp
    temp = new(struct node); //Cấp phát bộ nhớ cho node temp
    temp->info = value; //Thiết lập giá trị cho temp
    temp->next = last->next; //Thiết lập liên kết cho temp  last-
    >next = temp; //Thiết lập liên kết cho last
}
```

4.3.3.3. Chèn node vào sau vị trí pos:

```
void Chennut_pos_vong(int value, int pos){//Chèn node vào sau vị trí pos
    if (last == NULL){//Nếu danh sách rỗng
        cout<<“Không thể thực hiện được."<<endl;
        return;
    }
    struct node *temp, *s;
    s = last->next; //s trỏ đến node tiếp theo
    for (int i = 0;i < pos-1; i++){//Di chuyển đến vị trí pos-1
        s = s->next;
        if (s == last->next){ //Nếu s lại quay về đầu
                cout<<“Số node của danh sách bé hơn";
                cout<<pos<<"trong danh sách"<<endl;
                return;
        }
    }
    temp = new(struct node); temp->next = s->next;  temp-
    >info = value; s->next = temp;
    if (s == last){ //Nếu s là node cuối cùng
        last=temp;
```

```
    }
}
```

4.3.3.4. Loại bỏ node trong danh sách:

```
void Xoanut_vong(int value){//Loại node trong danh sách
    struct node *temp, *s; s = last->next;
    if (last->next == last && last->info == value){ //Nếu DS chỉ có một node
        temp = last; last = NULL; free(temp); return;
    }
    if (s->info == value) {//Nếu s là node đầu tiên
        temp = s; last->next = s->next; free(temp);return;
    }
    while (s->next != last){//Loai node ở giữa
        if (s->next->info == value) {
                temp = s->next; s->next = temp->next;free(temp);
                cout<<"Phần tử "<<value<<“đã loại bỏ”<<endl; return;
        }
        s = s->next;
    }
    if (s->next->info == value){ //Nếu s là node cuối cùng temp=s-
        >next;
        s->next=last->next;
        free(temp);
        last=s;return;
    }
    cout<<“Node"<<value<<" không tồn tại trong danh sách"<<endl;
}
```

4.3.3.5. Tìm node trong danh sách:

```
void Tim_vong(int value){//Tìm node trong DSLK vòng
    struct node *s; int counter = 0;
    s = last->next; //s là node tiếp theo
    while (s != last) { //Lặp trong khi s chưa phải cuối cùng
        counter++;
        if (s->info == value){ //Nếu node s có giá trị value
                cout<<“Tìm thấy node "<<value;
```

```
                cout<<" ở vị trí "<<counter<<endl;
                return;
            }
            s = s->next;
        }
        if (s->info == value){ //Nếu node cuối cùng là value
            counter++;
            cout<<“Tìm thấy node "<<value;
            cout<<" ở vị trí "<<counter<<endl;
            return;
        }
        cout<<“Giá trị "<<value<<" không có trong danh sách"<<endl;
}
```

4.3.3.6. Hiển thị nội dung các node trong danh sách:

```
void Hienthi_vong(){//Hiển thị nội dung các node trong DS
    struct node *s;
    if (last == NULL){
        cout<<“Không có gì để hiển thị"<<endl;
        return;
    }
    s = last->next; //s là node kế tiếp
    cout<<“Nội dung DSLKV: "<<endl;
    while (s != last){ //Lặp trong khi s chưa phải cuối cùng
        cout<<s->info<<"->"; //Hiển thị nội dung node s
        s = s->next; //s trỏ đến node tiếp theo
    }
    cout<<s->info<<endl; //Hiển thị node cuối cùng
}
```

4.3.3.7. Sửa đổi nội dung node:

```
void Sua_vong(){//Sửa đổi nội dung node
    int value, pos, i;
    if (last == NULL){ //Nếu danh sách rỗng
    cout<<“ không thực hiện được"<<endl; return;
    }
```

```
    cout<<“Nhập vị trí node cần sửa: ";cin>>pos;
    cout<<“Giá trị mới của node: ";cin>>value;
    struct node *s;
    s = last->next; //s là node tiếp theo
    for (i = 0;i < pos - 1; i++){//Chuyển đến vị trí pos-1
        if (s == last){ //Nếu s quay trở lại đầu
                cout<<“Số node nhỏ hơn "<<pos<<endl;
                return;
        }
        s = s->next;
    }
    s->info = value;
    cout<<“Node đã được sửa đổi"<<endl;
}
```

4.3.3.8. Sắp xếp nội dung node:

```
void Sapxep_vong(){//Sắp xếp nội dung các node
    struct node *s, *ptr; int temp;
    if (last == NULL){ //Nếu danh sách rỗng
    cout<<“Chưa có dữ liệu sắp xếp"<<endl;  return;
    }
    s = last->next; //s là node kế tiếp
    while (s != last){ //Lặp nếu s không phải là last
        ptr = s->next; //ptr là node kế tiếp của s
        while (ptr != last->next) { //Lặp đến node cuối cùng
                if (ptr != last->next) {
                        if (s->info > ptr->info) {
                                temp = s->info;
                                s->info = ptr->info;
                                ptr->info = temp;
                        }
                }
                else { break; }
                ptr = ptr->next;
        }
```

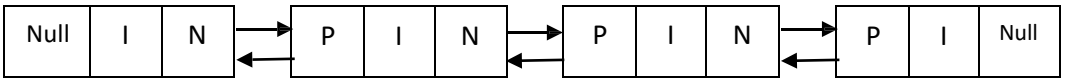
```
        s = s->next;
    }
}
```

4.4. Danh sách liên kết kép

4.4.1. Mô tả

Định nghĩa DSLK kép: Tập hợp các node (khối dữ liệu) được tổ chức rời rạc trong bộ nhớ. Trong đó, mỗi node gồm ba thành phần:

- Thành phần dữ liệu (infor): dùng để lưu trữ thông tin của node.
- Thành phần con trỏ prev: dùng để trỏ đến node dữ liệu sau nó.
- Thành phần con trỏ next: dùng để trỏ đến node dữ liệu trước nó



Hình 4.8: Danh sách liên kết kép

Biểu diễn danh sách liên kết kép

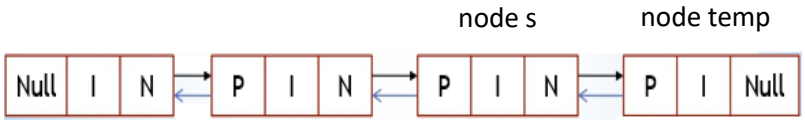
```
typedef struct node {
Item  Infor; //Thành phần dữ liệu của node
struct node *prev; //Thành phần con trỏ sau
struct node *next; //Thành phần con trỏ trước
}*L;
```

4.4.2. Khai báo

```
struct node { //Biểu diễn node
        int info; //Thành phần thông tin của node
        struct node *next; // Thành phần con trỏ đến node trước nó
        struct node *prev; //Thành phần con trỏ đến node sau nó
}*start; //Biến danh sách liên kết kép
```

4.4.3. Các thao tác trên danh sách liên kết kép

4.4.3.1. Tạo node cuối cùng cho danh sách liên kết kép:



Hình 4.9: Tạo node cuối cùng cho danh sách liên kết kép

```
void Taonutcuoi_kep(int value){ //Tạo node cuối cho DSLK kép
        struct node *s, *temp; //Sử dụng hai con trỏ s và temp
        temp = new(struct node); //Cấp phát miền nhớ cho temp
        temp->info = value; //Thiết lập thành phần thông tin cho temp
```

```
        temp->next = NULL; //Thiết lập liên kết tiếp theo cho temp
        if (start == NULL){ //Nếu danh sách rỗng
                temp->prev = NULL; //Thiết lập liên kết sau cho temp
                start = temp; //Node đầu tiên trong danh sách là temp
        }
        else { //Nếu danh sách không rỗng
                s = start; // s trỏ đến start
                while (s->next != NULL) //Lặp trong khi chưa đến node cuối
                s = s->next;
                s->next = temp; //Thiết lập liên kết tiếp theo cho node cuối
                temp->prev = s;//Thiết lập liên kết sau cho temp
        }
}
```

4.4.3.2. Loại node có thông tin value:

```
void Xoanut_kep(int value){//Loại node có giá trị value
        struct node *tmp, *q; //sử dụng hai con trỏ tmp và q
        if (start->info == value){//Nếu value là thông tin node đầu tiên
        tmp = start;  start = start->next;   start->prev = NULL;
        cout<<“Node đầu tiên đã bị loại bỏ"<<endl; free(tmp);  return
        }
        q = start; //q trỏ đến node đầu tiên
        while (q->next->next != NULL) { //Chuyển đến node trước của q->next if
                (q->next->info == value){ //Nếu node trước của q->next là value
                        tmp = q->next;
                        q->next = tmp->next;  tmp->next-
                        >prev = q;
                        cout<<“Node đã loại bỏ"<<endl;free(tmp); return;
                }
                q = q->next;
        }
        if (q->next->info == value){//Nếu value là node cuối cùng
                tmp = q->next;  free(tmp);
                q->next = NULL;
                cout<<“Node cuối cùng đã bị loại bỏ"<<endl;
```

```
            return;
        }
        cout<<"Node  "<<value<<" không có thực"<<endl;
    }
```

4.4.3.3. Hiển thị và đếm số node của danh sách:

```
    void Hienthi_kep(){//Hiển thị nội dung danh sách
        struct node *q;
        if (start == NULL){ //Nếu danh sách rỗng
        cout<<“Không có dữ liệu hiển thị"<<endl; return;
        }
        q = start; //Đặt q là node đầu tiên trong danh sách
        cout<<“Nội dung danh sách liên kết kép :"<<endl;
        while (q != NULL){ //Lặp cho đến node cuối cùng
            cout<<q->info<<" <-> "; //Hiển thị thông tin node
            q = q->next; //q trỏ đến node tiếp theo
        }
        cout<<"NULL"<<endl;
    }
    void Dem_kep(){ //Đếm số node của danh sách
        struct node *q = start;
        int cnt = 0;
        while (q != NULL){
            q = q->next;
            cnt++;
        }
        cout<<“Số node: "<<cnt<<endl;
    }
```

4.5. Danh sách liên kết kép vòng

4.5.1. Mô tả

Định nghĩa danh sách liên kết kép vòng: là một danh sách liên kết kép sao cho: con trỏ next của node cuối cùng trỏ đến node đầu tiên và con trỏ prev của node đầu tiên trỏ đến node cuối cùng.

4.5.2. Khai báo

Giống như biểu diễn của danh sách kép thông thường nhưng sử dụng con trỏ start luôn là node đầu tiên, con trỏ last luôn là node cuối cùng.

struct node {//Biểu diễn node

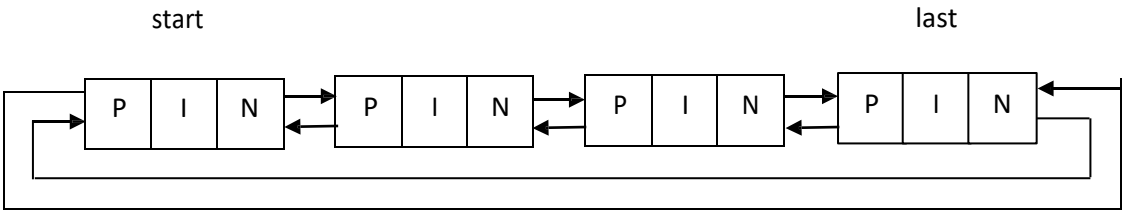
int info; //Thành phần thông tin của node

struct node *next; //Thành phần con trỏ next

struct node *prev; //Thành phần con trỏ prev

}*start, *last; //start là node đầu tiên, last là node cuối cùng

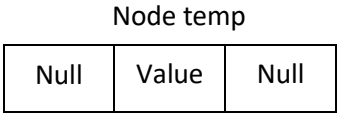
int counter =0; //ghi nhận số node của danh sách liên kết vòng



Hình 4.10: Danh sách liên kết kép vòng

4.5.3. Các thao tác trên danh sách liên kết kép vòng

4.5.3.1. Tạo node có giá trị value trên danh sách liên kết kép vòng:



Hình 4.11: Nút temp trong DSLK kép vòng

node* Taonut_kepvong(int value){//Tạo node có giá trị value

counter++; //Tăng số node

struct node *temp; //Sử dụng con trỏ temp

temp = new(struct node); //Cấp phát miền nhớ cho node temp->info = value; //Thiết lập giá trị cho node

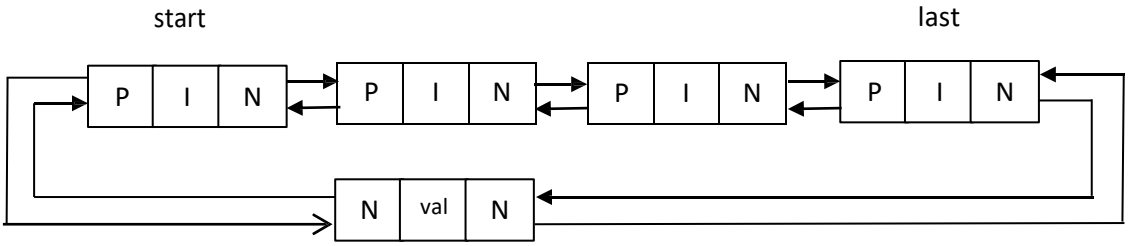
temp->next = NULL; //Thiết lập liên kết next

temp->prev = NULL; //Thiết lập liên kết prev

return temp;

}

4.5.3.2. Chèn node vào đầu danh sách liên kết kép vòng:



Hình 4.12: Chèn node vào đầu DSLK kép vòng

```
void Chennutdau_kepvong(){//Chèn node vào đầu DSLK kép vòng
    int value; cout<<endl<<"Nhập nội dung node: "; cin>>value;
    struct node *temp; // Khai báo con trỏ temp
    temp = create_node(value); // Tạo node có giá trị value
    if (start == last && start == NULL){//Nếu danh sách rỗng
        cout<<"Node được chèn là node duy nhất"<<endl;
        start = last = temp; //node đầu và nde cuối trùng nhau
        start->next = last->next = NULL;
        start->prev = last->prev = NULL;
    }
    else {//Thiết lập liên kết cho temp, start, last temp-
        >next = start;
        start->prev = temp;
        start = temp;
        start->prev = last; last-
        >next = start;
    cout<<"Node đã được chèn vào đầu"<<endl;
    }
}
```

4.5.3.3. Chèn node vào cuối danh sách liên kết kép vòng:

```
void Chennutcuoi_kepvong(){//Chèn node vào cuối
    int value;
    cout<<endl<<"Giá trị node chèn vào cuối: "; cin>>value;
    struct node *temp; //Khai báo node temp
    temp = create_node(value); //Tạo node temp có giá trị value
    if (start == last && start == NULL){ //Nếu danh sách rỗng
        cout<<"Chèn node vào danh sách rỗng"<<endl;
```

```
        start = last = temp;
        start→next = last→next = NULL;
        start→prev = last→prev = NULL;
        }
    else
        {
        last→next = temp;
        temp→prev = last;
        last = temp;
        start→prev = last;
        last→next = start;
        }
}
```

4.5.3.4. Chèn node tại vị trí bất kỳ trên danh sách liên kết kép vòng:

```
void Chennutbatky_kepvong(){ int value, pos, i;
    cout<<endl<<“Giá trị node cần chèn: ";cin>>value;
    cout<<endl<<“Vị trí node cần chèn: "; cin>>pos;
    struct node *temp, *s, *ptr;
    temp = create_node(value);
    if (start == last && start == NULL){//Nếu danh sách rỗng
        if (pos == 1){// Nếu vị trí cần chèn là 1
                start = last = temp;
                start->next = last->next = NULL;
                start->prev = last->prev = NULL;
        }
        else{ counter--; return; }
    }
    else {   if (counter < pos){ counter--; return; }
        s = start;
        for (i = 1;i <= counter;i++){ ptr = s; s = s->next;
                if (i == pos – 1) {ptr->next = temp;temp->prev = ptr; temp-
                        >next = s;
                        s->prev = temp;
                        cout<<“Hoàn thành"<<endl; break;
```

```
                }
            }
        }
    }
```

4.5.3.5. Loại node tại vị trí bất kỳ trên danh sách liên kết kép vòng:

```
void Xoanutbatky_kepvong(){ int pos, l; node *ptr, *s;
    if (start == last && start == NULL){
        cout<<“Không có dữ liệu”; return;
    }
    cout<<endl<<“Vị trí cần loại bỏ: "; cin>>pos;
    if (counter < pos){ cout<<“Vị trí không hợp lệ"<<endl;return;}
    s = start; //s là node đầu tiên
    if(pos == 1){//Nếu là vị trí đầu tiên
        counter--;
        last->next = s->next;  s-
        >next->prev = last;
        start = s->next; free(s);
        cout<<“Node đầu tiên đã bị loại"<<endl;
        return;
    }
    for (i = 0; i < pos - 1; i++ ){ //Di chuyển đến vị trí pos-1
        s = s->next;
        ptr = s->prev;
    }
    ptr->next = s->next;  s->next-
    >prev = ptr;
    if (pos == counter) { last = ptr; }  counter-
    -; free(s);
    cout<<“Node đã bị loại bỏ"<<endl;
}
```

4.5.3.6. Tìm kiếm node tại vị trí bất kỳ trên danh sách liên kết kép vòng:

```
void Tim_kepvong(){//Tìm kiếm node
    int pos = 0, value, i;
    bool flag = false;
```

```
    struct node *s;
    if (start == last && start == NULL){
        cout<<“Danh sách rỗng"<<endl;
        return;
    }
    cout<<endl<<“Nội dung node cần tìm: "; cin>>value;
    s = start; //s là node đầu tiên
    for (i = 0;i < counter;i++){ pos++;
        if (s→info == value){
                cout<<“Tìm thấy "<<value<<" tại vị trí: "<<pos<<endl;
                flag = true;
        }
        s = s→next;
    }
    if (!flag)
        cout<<“Không tìm thấy"<<endl;
}
```

4.5.3.7. Sắp xếp node trên danh sách liên kết kép vòng:

```
void Sapxep_kepvong(){//Sap xep
    struct node *temp, *s;  int value, i;
    if (start == last && start == NULL){
        cout<<“Danh sách rỗng"<<endl;
        return;
    }
    s = start; //s là node đầu tiên
    for (i = 0; i <counter; i++){
        temp = s→next;
        while (temp != start){
                if (s→info > temp→info){ value = s→info; s→info = temp→info;
                        temp→info = value;
                }
                temp = temp→next;
        }
        s = s→next;
```

```
    }
}
```

4.5.3.8. Hiển thị nội dung danh sách liên kết kép vòng:

```
void Hienthi_kepvong(){//Hiển thị nội dung danh sách
    int i;
    struct node *s;
    if (start == last && start == NULL){
        cout<<“Danh sách rỗng"<<endl;
        return;
    }
    s = start; //s là node đầu tiên
    for (i = 0;i < counter-1;i++){
        cout<<s→info<<"<->"; //Hiển thị thông tin của s
        s = s→next; //s trỏ đến node tiếp theo
    }
    cout<<s→info<<endl; //Hiển thị node cuối cùng
}
```

Bài tập vận dụng

1. Viết chương trình minh họa việc sử dụng danh sách liên kết đơn với các chức năng:
    a. Khởi tạo danh sách
    b. Thêm phần tử
    c. Xoá phần tử
2. Nêu các bước để thêm một nút vào đầu, giữa, và cuối danh sách liên kết đơn, danh sách liên kết vòng, danh sách liên kết kép.
3. Nêu các bước để xoá một nút ở đầu, giữa, và cuối danh sách liên kết đơn, danh sách liên kết vòng, danh sách liên kết kép.
4. Viết chương trình thực hiện việc sắp xếp 1 danh sách liên kết đơn bao gồm các phần tử là các số nguyên.
5. Nêu các bước tìm node tại vị trí bất kỳ trên danh sách liên kết đơn, danh sách liên kết kép, danh sách liên kết vòng.
6. Nêu các bước sửa đổi thông tin node và hiển thị thông tin trên danh sách liên kết đơn, danh sách liên kết kép, danh sách liên kết vòng.
7. Nêu các bước sắp xếp thông tin trên danh sách liên kết đơn, danh sách liên kết kép, danh sách liên kết vòng.
8. Hãy nêu các ưu và nhược điểm của danh sách liên kết so với mảng.

9. Cho ví dụ minh họa sắp xếp node trên danh sách liên kết kép vòng.

10. Cho danh sách nối đơn nút đầu tiên được trỏ bởi L.

Hãy lập các giải thuật thực hiện:

+ Tìm nút đầu tiên thỏa mãn info=X.

+ Chuyển nút thứ k thành nút cuối cùng của danh sách.

11. Cho danh sách nối đơn nút đầu tiên được trỏ bởi L. Giá trị của trường info trong các nút là các số khác nhau và đã được sắp xếp tăng dần. Hãy lập các giải thuật thực hiện:

+ Đếm số nút của danh sách.

+ Bổ sung một nút mới có info=X vào danh sách.

12. Cho danh sách các mặt hàng, thông tin về một mặt hàng gồm có: Tên hàng, giá.

a) Hãy biểu diễn CTDL của danh sách nói trên dưới dạng một DSLK đơn.

b) Dựa vào CTDL đã biểu diễn ở trên hãy viết các thủ tục thực hiện các yêu cầu sau:

+ Tính tổng giá trị của các mặt hàng có giá từ 5000 trở xuống.

+ Bổ sung một mặt hàng mới vào cuối danh sách với tên hàng và giá nhập từ bàn phím.