CHƯƠNG 3. Stack – Queue – Link List

NỘI DUNG:

3.1. Danh sách liên kết đơn (Single Linked List)

3.2. Danh sách liên kết đơn vòng (Circular Single Linked List)

3.3. Danh sách liên kết kép (Double Linked List)

3.4. Danh sách liên kết kép vòng (Circualar Double Linked List)

3.5. Ngăn xếp (Stack)

3.6. Hàng đợi (Queue)

3.7. Hàng đợi vòng (Curcular Queue)

3.8. CASE STUDY

3.1. Danh sách liên kết đơn

3.1.1. Định nghĩa. 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.

Một số vấn đề cần thảo luận:

- Tại sao phải xây dựng danh sách liên kết đơn:
  - Vấn đề bộ nhớ.
  - Vấn đề thêm phần tử.
  - Vấn đề loại bỏ phần tử.
- Khi nào sử dụng danh sách liên kết đơn?
- So sánh danh sách liên kết đơn và mảng?

3.1.2. Biểu diễn danh sách liên kết đơn

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 {
        Item    Infor; //Thông tin của mỗi node;
        struct  node *next;
} *List;
```

3.1.3. Các thao tác trên danh sách liên kết đơn

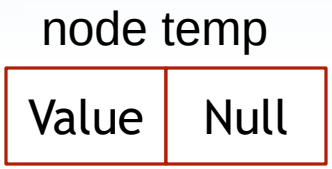
- Khởi tạo danh sách liên kết đơn: đưa trạng thái danh sách liên kết đơn về trạng thái rỗng. Ta gọi thao tác này là Init().
- Cấp phát miền nhớ cho một node: khi thực hiện thêm node vào danh sách thì node cần thêm vào cần trỏ đến một miền nhớ cụ thể thông qua các thao tác cấp phát bộ nhớ.
- Thêm node vào đầu bên trái danh sách liên kết đơn.
- Thêm node vào đầu bên phải theo chiều con trỏ next.
- Thêm node vào node giữa danh sách liên kết đơn.
- Loại node cuối bên trái danh sách liên kết đơn.
- Loại node cuối bên phải theo chiều con trỏ next.
- Loại node ở giữa danh sách liên kết đơn.
- Duyệt thông tin của danh sách liên kết đơn.
- Tìm node trên danh sách liên kết đơn.

Lớp các thao tác trên danh sách liên kết đơn (DSLKĐ):

```
struct node { // biểu diễn node
        int info; //thành phần thông tin của node
        struct node *next; //thành phần con trỏ của node
}*start; // danh sách liên kết đơn: *start.
class single_llist { //Biểu diễn lớp llist
    public:
        node* create_node(int);//Tạo một node cho danh sách liên kết đơn
        void insert_begin(); //Thêm node vào đầu DSLKĐ
        void insert_pos(); //Thêm node tại vị trí ch trước trên DSLKĐ
        void insert_last(); //Thêm node vào cuối DSLKĐ
        void delete_pos(); //Loại node tại vị trí cho trước trên DSLKĐ
        void sort(); //Sắp xếp nội dung các node theo thứ tự tăng dần
        void search(); //Tìm kiếm node trên DSLKĐ
        void update(); //Sửa đổi thông tin của node trên DSLKĐ
        void reverse(); //Đảo ngược danh sách liên kết đơn
        void display(); //Hiển thị nội dung DSLKĐ
        single_llist(){//Constructor của lớp llist.
                start = NULL;
        }
};
```

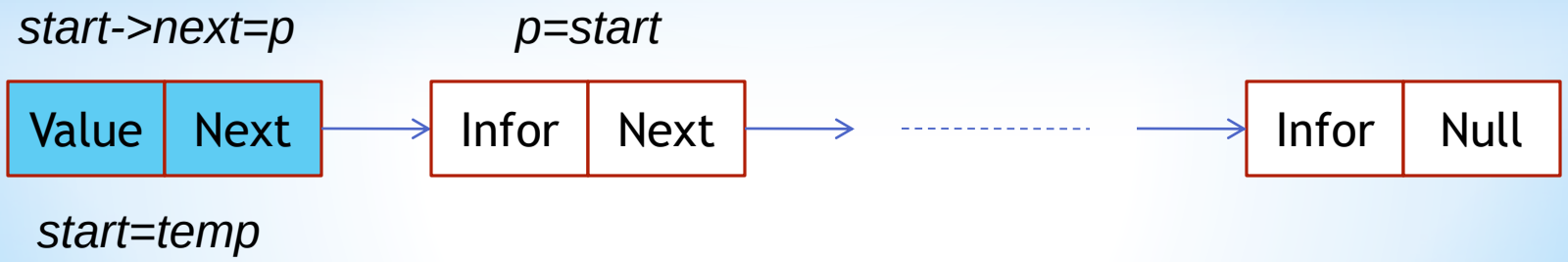
Khởi tạo một node cho DSLKĐ:

```
node *single_llist::create_node(int value){
        struct node *temp; //Khai báo hai con trỏ node *temp
        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 NULL;
        }
        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
        }
}
```



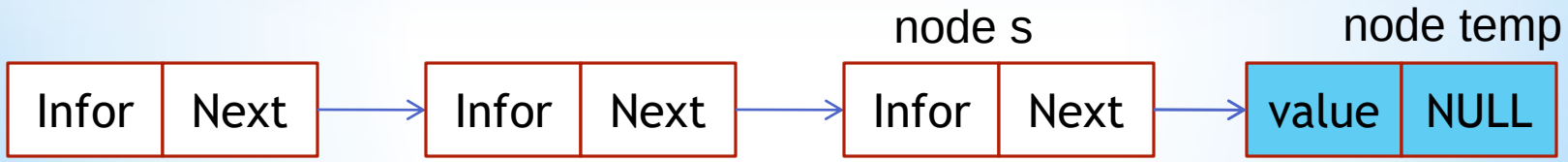
Chèn node vào đầu DSLKĐ:

```
void single_llist::insert_begin(){ //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;
}
```



Thêm node vào cuối DSLKĐ:

```
void single_llist::insert_last(){//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;
}
```



Thêm node vị trí pos:

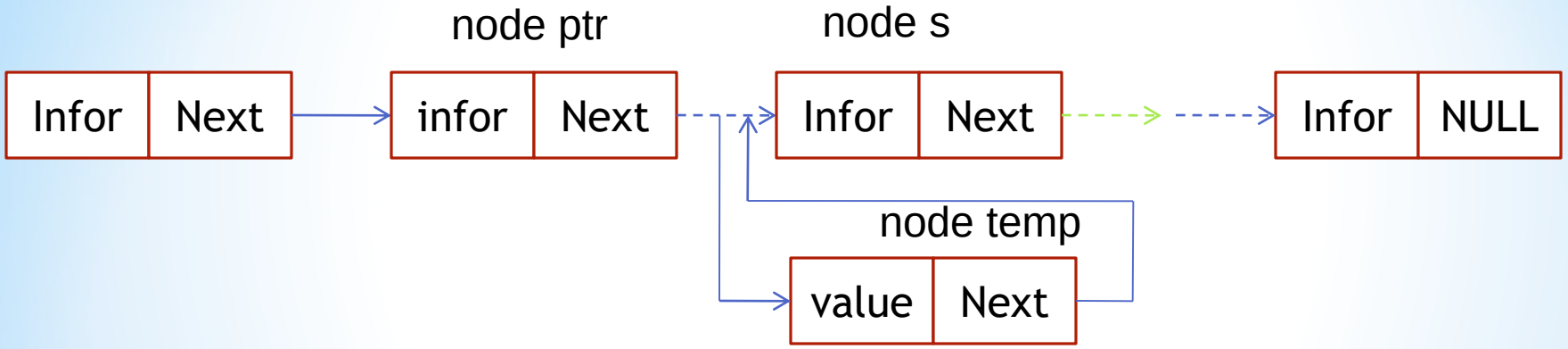
```
void single_llist::insert_pos(){//Thêm node vào vị trí pos
        int value, pos, counter = 0; cout<<"Nhap gia tri node:";cin>>value;
        struct node *temp, *s, *ptr; temp = create_node(value);//Tạo node
        cout<<“Nhập vị trí node cần thêm: ";cin>>pos;
        int i; s = start; //s trỏ 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; }
}
```

Loại node ở vị trí pos:

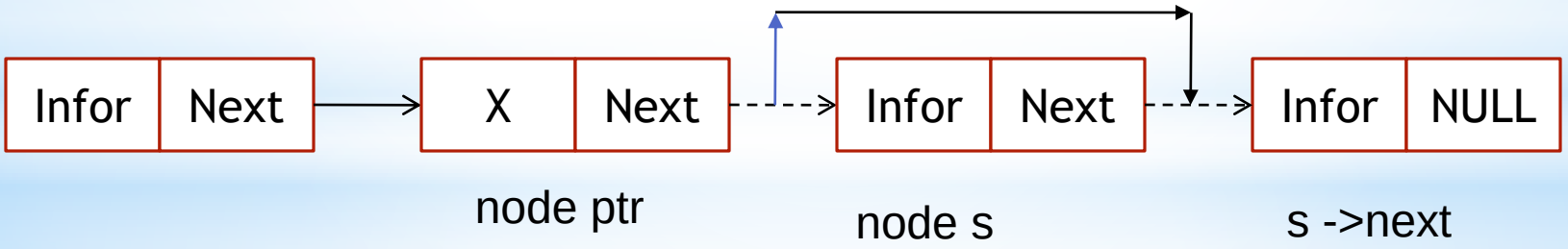
```
void single_llist::delete_pos(){//Loai phan tu o vi tri cho truoc
        int pos, i, counter = 0;
        if (start == NULL){  cout<<“Không thực hiện được"<<endl; return; }
        cout<<“Vị trí cần loại bỏ:";cin>>pos;
        struct node *s, *ptr;  s = start; //s trỏ đến đầu danh sách
        if (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;
        }
}
```

Giải thích thêm về hai thao tác chèn node và loại bỏ node:

Thêm node vào vị trí pos:



Loại node ở vị trí pos:



Sửa đổi nội dung node:

```
void single_llist::update(){//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;
}
```

Tìm kiếm node trên DSLKĐ:

```
void single_llist::search(){//Tìm kiếm node
        int value, pos = 0; bool flag = false;
        if (start == NULL){
                cout<<“Danh sách rỗng thì tìm cài gì?"<<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;
        }
}
```

Hiển thị nội dung DSLKĐ:

```
void single_llist::display(){//Hiển thị nội dung DSLKĐ
        struct node *temp; //Sử dụng một con trỏ temp
        if (start == NULL){ // Nếu danh sách rỗng
                cout<<"Có gì đâu mà 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
}
```

Sắp xếp nội dung các node của DSLKĐ:

```
void single_llist::sort(){//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<<"Có gì đâu mà 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;
        }
}
```

Đảo ngược các node trong DSLKĐ:

```
void single_llist::reverse(){//Đảo ngược danh sách
        struct node *ptr1, *ptr2, *ptr3; //Sử
        if (start == NULL) {//Nếu danh sách rỗng
                cout<<"Ta không cần đảo"<<endl; return;
        }
        if (start->next == NULL){//Nếu danh sách chỉ có một node
                 cout<<"Đảo ngược là chính nó"<<endl; return;
        }
        ptr1 = start; //ptr1 trỏ đến node đầu tiên
        ptr2 = ptr1->next;//ptr2 trỏ đến node kế tiếp của ptr1
        ptr3 = ptr2->next;//ptr3 trỏ đến nod kế tiếp của ptr2
        ptr1->next = NULL;//Ngắt liên kết ptr1
        ptr2->next = ptr1;//node ptr2 bây giờ đứng trước node ptr1
        while (ptr3 != NULL){//Lặp nếu ptr3 khác rỗng
                ptr1 = ptr2;  //ptr1 lại bắt đầu tại vị trí ptr2
                ptr2 = ptr3;  //ptr2 bắt đầu tại vị trí ptr3
                ptr3 = ptr3->next; //ptr3 trỏ đến node kế tiếp
                ptr2->next = ptr1; //Thiết lập liên kết cho ptr2
        }
        start = ptr2; //node đầu tiên bây giờ là ptr2
}
```

3.1.4. Ứng dụng của danh sách liên kết

- Xây dựng các lược đồ quản lý bộ nhớ:
  - Thuật toán Best Fit:
  - Thuật toán First Fit:
  - Thuật toán Best Availbale.
- Biểu diễn ngăn xếp :
  - Danh sách L + { Add-Top, Del-Top}.
  - Danh sách L + { Add-Bottom, Del-Bottom}.
- Biểu diễn hàng đợi:
  - Danh sách L + { Add-Top, Del-Bottom}.
  - Danh sách L + { Add-Bottom, Del-Top}.
- Biểu diễn cây.
- Biểu diễn đồ thị.
- Biểu diễn tính toán.

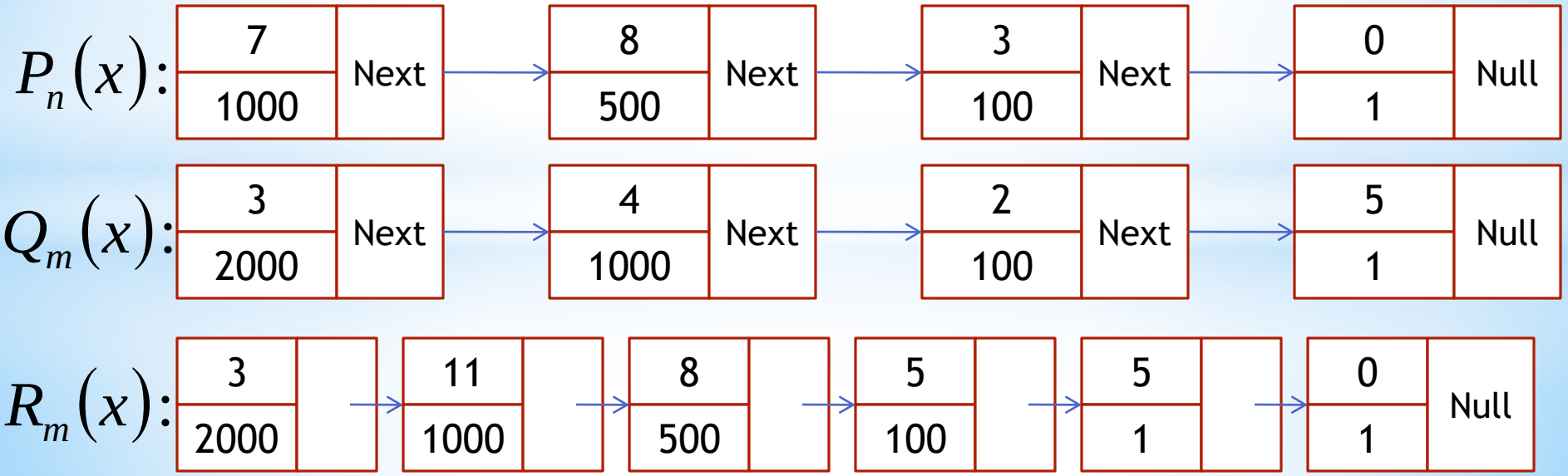
Ví dụ. Thuật toán cộng hai đa thức R = $P_n(x)$ + $Q_m(x)$.

$$P_n(x) = 7x^{1000} + 8x^{500} + 3x^{100} + 1$$

$$Q_m(x) = 3x^{2000} + 4x^{1000} + 2x^{100} + 5x$$

Biểu diễn mỗi số hạng của đa thức:

```
typedef struct node {
        float hsomueso;
        float ;
        struct node *next;
} *dathuc;
```



Thuật toán Cong_Dathuc (Dathuc *P, Dathuc *Q):

Bước 1 (Khởi tạo): R = ∅;

Bước 2 (lặp):

```
while ( P≠∅ && Q≠∅ ) {
        if( P→Somu > Q → Somu ) {
                R = R →P; P = P →next;
        }
        else if (P→Somu < Q → Somu ) {
                R = R →Q; Q = Q →next;
        }
        else {    P → heso = P → heso + Q → heso;
                R = R →P; P = P →next; Q = Q →next;
        }
}
```

Bước 3 (Hoàn chỉnh đa thức):

```
if (P ≠∅ ) R = R = R →P;
 if (Q ≠∅ ) R = R = R →Q;
```

Bước 4 (Trả lại kết quả):

```
Return (R);
```

3.2. Danh sách liên kết đơn vòng (Circualar single linked List)

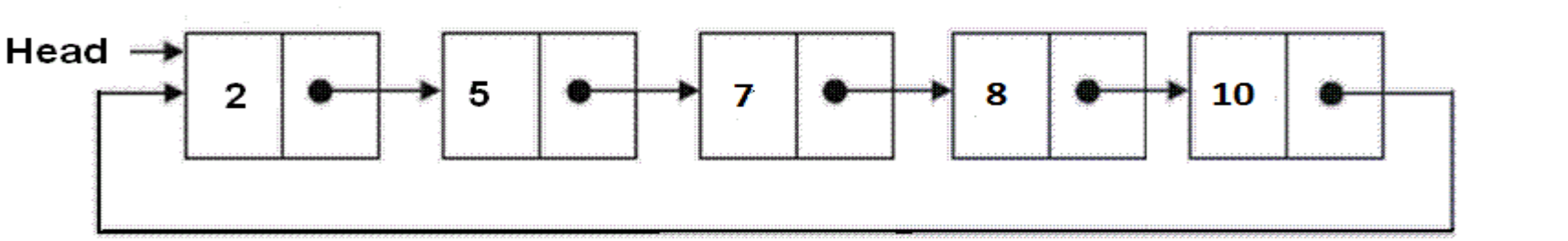
3.2.1. Định nghĩa. Là danh sách liên kết đơn trong đó tất cả các node liên kết với nhau thành một vòng tròn. Không có con trỏ NULL ở node cuối cùng mà được liên kết với node đầu tiên.

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

- Mọi node đều là node bắt đầu. Tao có thể duyệt tại bất kỳ node nào và chỉ dừng khi ta lặp lại một node đã duyệt.
- Dễ dàng trong việc cài đặt hàng đợi.
- Dễ dàng phát triển các ứng dụng thực hiện vòng quang danh sách.

3.2.2. Biểu diễn: Giống như biểu diễn của danh sách liên kết đơn.

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



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

- Tạo lập node cho danh sách liên kết đơn vòng.
- Thêm node đầu tiên cho sách liên kết đơn vòng.
- Thêm node vào sau một node khác.
- Loại bỏ node trên danh sách liên kết đơn vòng.
- Tìm kiếm node trên danh sách liên kết đơn vòng.
- Hiển thị nội dung danh sách liên kết đơn vòng.
- Sửa đổi nội dung node cho danh sách liên kết đơn vòng.
- Sắp xếp nội dung các node trên danh sách liên kết đơn vòng.

Khai báo danh sách liên kết vòng: giống như danh sách liên kết thông thường.

```
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;
```

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

```
class circular_llist{//Mô tả lớp danh sách liên kết đơn vòng
    public:
        void create_node(int value); //Tạo node cho DSLKĐ vòng
        void add_begin(int value); //Thêm node đầu tiên
        void add_after(int value, int position); //Thêm node sau vị trí pos
        void delete_element(int value); //Loại bỏ node
        void search_element(int value); //Tìm kiếm node
        void display_list(); //Hiển thị node
        void update(); //Sửa đổi nội dung node
        void sort(); //Sắp xếp node
        circular_llist(){//Constructor của lớp circular_llist
            last = NULL;
        }
};
```

Tạo node cho danh sách liên kết đơn vòng:

```
void circular_llist::create_node(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
        }
}
```

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

```
void circular_llist::add_begin(int value){//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
}
```

Chèn node vào sau vị trí pos:

```
void circular_llist::add_after(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;
        }
}
```

Loại bỏ node trong danh sách:

```
void circular_llist::delete_element(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;
}
```

Tìm node trong danh sách:

```
void circular_llist::search_element(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;
}
```

Hiển thị nội dung các node trong danh sách:

```
void circular_llist::display_list(){//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
}
```

Sửa đổi nội dung node:

```
void circular_llist::update(){//Sửa đổi nội dung node
        int value, pos, i;
        if (last == NULL){ //Nếu danh sách rỗng
                cout<<"Ta không làm gì 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;
}
```

Sắp xếp nội dung node:

```
void circular_llist::sort(){//Sắp xếp nội dung các node
        struct node *s, *ptr;  int temp;
        if (last == NULL){ //Nếu danh sách rỗng
                cout<<"Có gì đâu mà 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;
        }
}
```

Bài tập 1. Hoàn thành việc xây dựng các thao tác cơ bản trên danh sách liên kết đơn, bao gồm:

- Khởi tạo danh sách liên kết đơn.
- Chèn node vào đầu danh sách liên kết đơn.
- Chèn node vào cuối danh sách liên kết đơn.
- Chèn node vào vị trí xác định trong danh sách liên kết đơn.
- Loại node tại vị trí Pos trong danh sách liên kết đơn.
- Sửa đổi nội dung node trong danh sách liên kết đơn.
- Sắp xếp các node của danh sách liên kết đơn.
- Đảo ngược các node trong danh sách liên kết đơn.
- Tìm kiếm vị trí của node trong danh sách liên kết đơn.
- Hiển thị nội dung trong danh sách liên kết đơn.

Bài tập 2. Hoàn thành bài tập 1 sử dụng C++ STL .

Bài tập 3. Xây dựng tập thao tác trên đa thức dựa vào danh sách liên kết đơn.

Bài tập 4. Xây dựng các phép toán với số lớn bằng danh sách liên kết đơn.

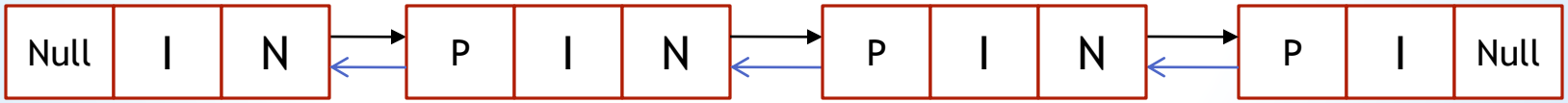
Bài tập 5. Hoàn thành các thao tác trên danh sách liên kết đơn vòng

- Tạo lập node cho danh sách liên kết đơn vòng.
- Thêm node đầu tiên cho sách liên kết đơn vòng.
- Thêm node vào sau một node khác.
- Loại bỏ node trên danh sách liên kết đơn vòng.
- Tìm kiếm node trên danh sách liên kết đơn vòng.
- Hiển thị nội dung danh sách liên kết đơn vòng.
- Sửa đổi nội dung node cho danh sách liên kết đơn vòng.
- Sắp xếp nội dung các node trên danh sách liên kết đơn vòng.

3.3. Danh sách liên kết kép

3.3.1. Định nghĩa. 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.



3.3.2. 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;
```

3.3.3. Các thao tác trên danh sách liên kết kép

- Khởi tạo danh sách liên kết kép.
- Cấp phát miền nhớ cho một node.
- Thêm node vào đầu bên trái danh sách liên kết kép.
- Thêm node vào đầu bên phải danh sách liên kết kép.
- Thêm node vào node giữa danh sách liên kết kép.
- Loại node cuối bên trái danh sách liên kết kép.
- Loại node cuối bên phải danh sách liên kết kép.
- Loại node ở giữa danh sách liên kết kép.
- Duyệt trái danh sách liên kết kép.
- Duyệt phải danh sách liên kết kép.
- Tìm node trên danh sách liên kết kép…

Dưới đây là một số thao tác cơ bản trên danh sách liên kết kép.

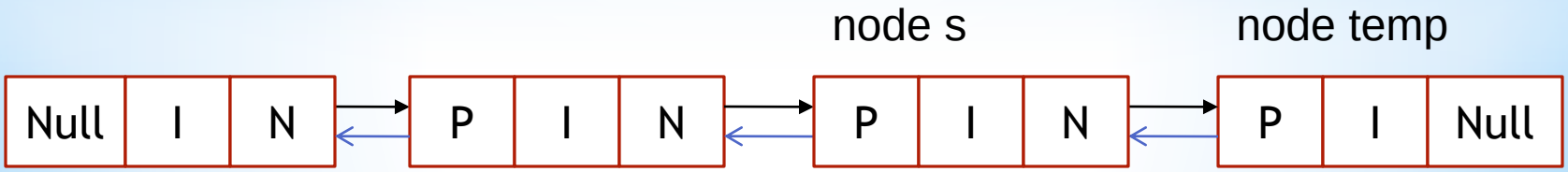
Tập thao tác trên danh sách liên kết kép:

```
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

class double_llist {//Mô tả lớp các thao tác
    public:
        void create_list(int value);//Tạo node cho DSLK kép
        void add_begin(int value);//Thêm node vào đầu danh sách
        void add_after(int value, int position);//Thêm node vào sau position
        void delete_element(int value);//Loại node có thông tin value
        void search_element(int value);//Tìm node có thông tin value
        void display_dlist();//Hiển thị danh sách liên kết kép
        void count();//Đếm số node
        void reverse();//Đảo ngược danh sách
        double_llist(){//Constructor của lớp
            start = NULL;
        }
};
```

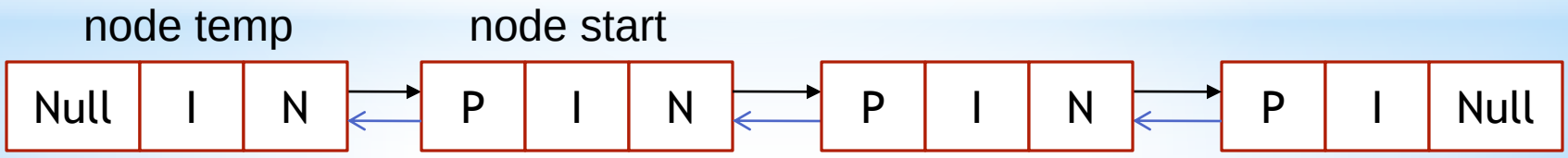
Tạo node cuối cùng cho danh sách liên kết kép:

```
void double_llist::create_list(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
        }
}
```



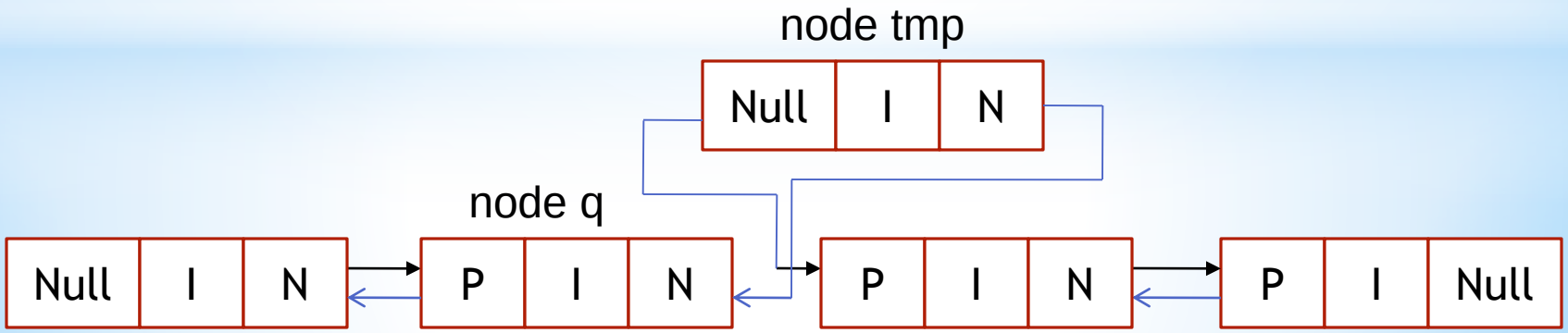
Thêm node vào đầu danh sách liên kết kép:

```
void double_llist::add_begin(int value){//Thêm node vào sau vị trí pos
        if (start == NULL){ //Nếu danh sách rỗng
                cout<<"Không phải làm gì."<<endl;
                return;
        }
        struct node *temp; //Sử dụng con trỏ temp
        temp = new(struct node); //Cấp phát miền nhớ cho temp
        temp->prev = NULL; //temp->prev được thiết lập là Null
        temp->info = value; //Thiết lập thông tin cho temp
        temp->next = start; // temp->next là start
        start->prev = temp; //start->prev là temp.
        start = temp; //Node đầu tiên bây giờ là temp
        cout<<"Node đã thêm vào đầu"<<endl;
}
```



Thêm node vào sau vị trí pos:

```
void double_llist::add_after(int value, int pos){//Thêm node vào sau vị trí pos
        if (start == NULL){ cout<<"Danh sách rỗng."<<endl;  return; }
        struct node *tmp, *q;   int i;  q = start;
        for (i = 0;i < pos - 1;i++) {//Chuyển q đến vị trí pos
                q = q->next;
                if (q == NULL) { cout<<"Số node nhỏ hơn "<<pos<<"<<endl;
                        return;
                }
        }
        tmp = new(struct node); tmp->info = value;//Thiết lập thông tin cho tmp
        if (q->next == NULL) { //Nếu q là node cuối cùng
                q->next = tmp; tmp->next = NULL;  tmp->prev = q;
        }
        else  { //Nếu q không phải node cuối cùng
                tmp->next = q->next;  tmp->next->prev = tmp;
                q->next = tmp; tmp->prev = q;
        }
}
```



Loại node có thông tin value:

```
void double_llist::delete_element(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;
}
```

Hiển thị và đếm số node của danh sách:

```
void double_llist::display_dlist(){//Hiển thị nội dung danh sách
        struct node *q;
        if (start == NULL){ //Nếu danh sách rỗng
                cout<<"Không có gì để 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 double_llist::count(){ //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;
}
```

Đảo ngược danh sách liên kết kép:

```
void double_llist::reverse(){//Đảo ngược danh sách liên kết kép
        struct node *p1, *p2; //Sử dụng hai con trỏ p1, p2
        p1 = start; //Đặt p1 trỏ đến node đầu tiên
        p2 = p1->next; //Đặt p2 trỏ đến node kế tiếp của p1
        p1->next = NULL; //Ngắt liên kết next của p1
        p1->prev = p2; //Thiết lập liên kết sau cho p1
        while (p2 != NULL) {
                p2->prev = p2->next; //Thiết lập liên kết sau cho p2
                p2->next = p1; //Thiết lập liên kết trước cho p2
                p1 = p2; //Đặt p1 vào p2
                p2 = p2->prev; //Thiết lập lại liên kết sau cho p2
        }
        start = p1; //Thiết lập node cuối cùng
        cout<<"Danh sách đã đảo ngược"<<endl;
}
```

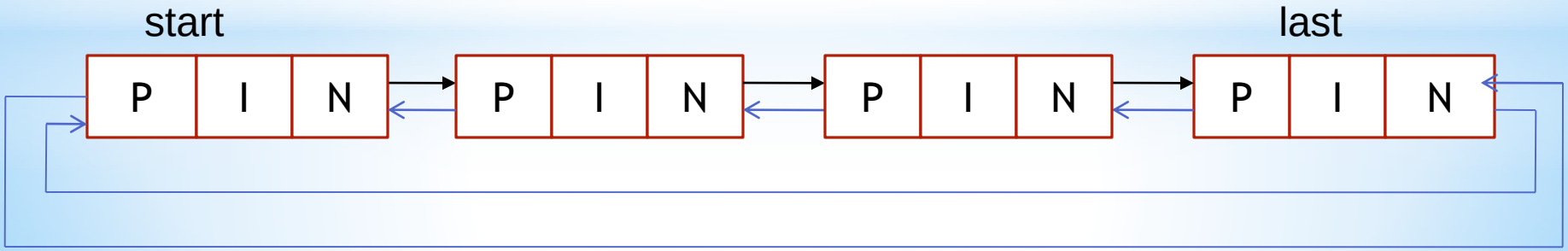
3.4. Danh sách liên kết kép vòng

3.4.1. 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.

3.4.2. Biểu diễn danh sách liên kết kép vòng

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
```



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

- Tạo node cho danh sách liên kết kép vòng.
- Chèn node vào đầu danh sách liên kết kép vòng.
- Chèn node vào cuối danh sách liên kết kép vòng.
- Chèn node giữa cuối danh sách liên kết kép vòng.
- Thêm node vào node giữa danh sách liên kết kép vòng.
- Loại node tại vị trí bất kỳ trên danh sách liên kết kép vòng.
- Tìm node tại vị trí bất kỳ trên danh sách liên kết kép vòng.
- Sửa đổi thông tin node trên danh sách liên kết kép vòng.
- Hiển thị thông tin trên danh sách liên kết kép vòng.
- Sắp xếp thông tin trên danh sách liên kết kép vòng.

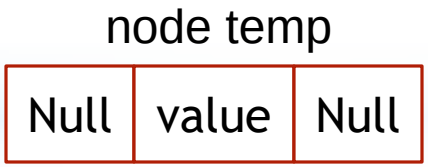
Dưới đây là một cách cài đặt cho các thao tác cơ bản trên danh sách liên kết kép vòng.

Mô tả lớp thao tác trên danh sách liên kết kép vòng:

```
class double_clist {//Mô tả lớp double-clisst
    public:
        node *create_node(int); //Tạo node có giá trị value
        void insert_begin(); //Chèn node vào đầu DSLK kép vòng
        void insert_last(); //Chèn node vào cuối DSLK kép vòng
        void insert_pos(); //Chèn node vào giữa DSLK kép vòng
        void delete_pos(); //Loại node tại vị trí bất kỳ
        void search(); //Tìm node tại vị trí bất kỳ
        void update(); //Sửa đổi thông tin node tại vị trí bất kỳ
        void display(); //Hiển thị nội dung DSLK kép vòng
        void reverse(); // Đảo ngược DSLK kép vòng
        void sort(); // Sắp xếp DSLK kép vòng
        double_clist() {//Constructor DSLK kép vòng
            start = NULL; last = NULL;
        }
};
```

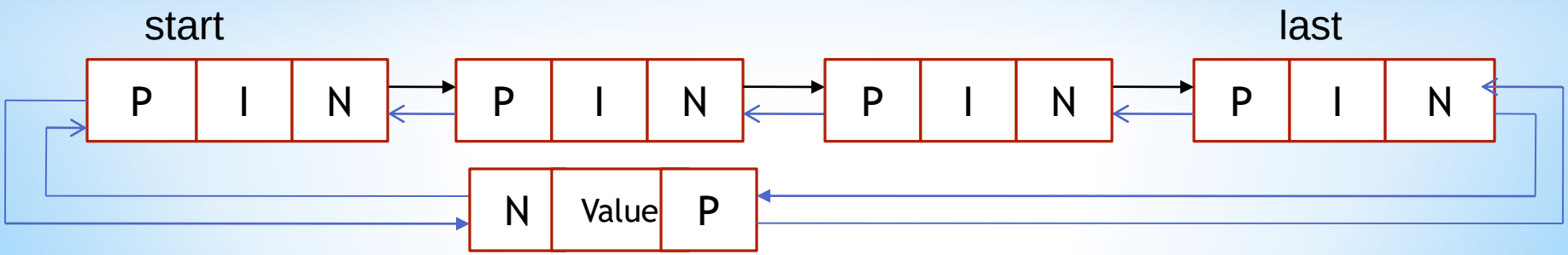
Tạo node có giá trị value trên danh sách liên kết kép vòng:

```
node* double_clist::create_node(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;
}
```



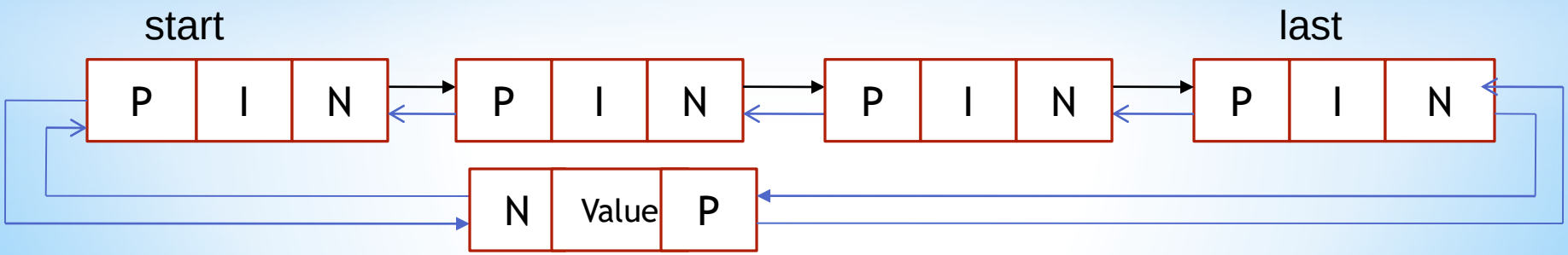
Chèn node vào đầu danh sách liên kết kép vòng:

```
void double_clist::insert_begin(){//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;
        }
}
```



Chèn node vào cuối danh sách liên kết kép vòng:

```
void double_clist::insert_last(){//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;
        }
}
```



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

```
void double_clist::insert_pos(){    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;
                        }
                }
        }
}
```

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

```
void double_clist::delete_pos(){ int pos, I; node *ptr, *s;
        if (start == last && start == NULL){
                cout<<“Không phải làm gì.”;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;
}
```

Sửa đổi node tại vị trí bất kỳ trên danh sách liên kết kép vòng:

```
void double_clist::update(){//Sửa đổi thông tin cho node
        int value, i, pos;
        if (start == last && start == NULL){
                cout<<"Danh sách rỗng"<<endl;return;
        }
        cout<<endl<<"Vị trí node cần sửa: ";cin>>pos;
        cout<<"Giá trị mới: ";cin>>value;
        struct node *s;
        if (counter < pos){ cout<<"Vị trí không hợp lệ"<<endl;return;}
        s = start;  // s là node đầu tiên
        if (pos == 1) {//Nếu vị trí sửa đổi là vị trí đầu tiên
                s->info= value;cout<<"Node đã được  sửa đổi "<<endl;return;
        }
        for (i=0;i < pos - 1;i++) {//Di chuyển s đến vị trí pos-1
                s = s->next;
        }
        s->info = value;// Cập nhật thông tin sửa đổi
        cout<<"Thông báo sửa thành công"<<endl;
}
```

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

```
void double_clist::search(){//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;
}
```

Sắp xếp node trên danh sách liên kết kép vòng:

```
void double_clist::sort(){//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;
        }
}
```

Hiển thị nội dung danh sách liên kết kép vòng:

```
void double_clist::display(){//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
}
```

Đảo ngược danh sách liên kết kép vòng:

```
void double_clist::reverse(){//Dao nguoc
        if (start == last && start == NULL){
                cout<<“Danh sách rỗng"<<endl;
                return;
        }
        struct node *p1, *p2;
        p1 = start;
        p2 = p1->next;
        p1->next = NULL;
        p1->prev = p2;
        while (p2 != start){
                p2->prev = p2->next;
                p2->next = p1;
                p1 = p2;
                p2 = p2->prev;
        }
        last = start; start = p1;
        cout<<“Danh sách đã được đảo ngược"<<endl;
}
```

Bài tập 6. Hoàn thành các thao tác trên danh sách liên kết kép

- Khởi tạo danh sách liên kết kép.
- Cấp phát miền nhớ cho một node.
- Thêm node vào đầu bên trái danh sách liên kết kép.
- Thêm node vào đầu bên phải danh sách liên kết kép.
- Thêm node vào node giữa danh sách liên kết kép.
- Loại node cuối bên trái danh sách liên kết kép.
- Loại node cuối bên phải danh sách liên kết kép.
- Loại node ở giữa danh sách liên kết kép.
- Duyệt trái danh sách liên kết kép.
- Duyệt phải danh sách liên kết kép.
- Tìm node trên danh sách liên kết kép…

Dưới đây là một số thao tác cơ bản trên danh sách liên kết kép.

Bài tập 7. Hoàn thành các thao tác trên danh sách liên kết kép vòng

- Tạo node cho danh sách liên kết kép vòng.
- Chèn node vào đầu danh sách liên kết kép vòng.
- Chèn node vào cuối danh sách liên kết kép vòng.
- Chèn node giữa cuối danh sách liên kết kép vòng.
- Thêm node vào node giữa danh sách liên kết kép vòng.
- Loại node tại vị trí bất kỳ trên danh sách liên kết kép vòng.
- Tìm node tại vị trí bất kỳ trên danh sách liên kết kép vòng.
- Sửa đổi thông tin node trên danh sách liên kết kép vòng.
- Hiển thị thông tin trên danh sách liên kết kép vòng.
- Sắp xếp thông tin trên danh sách liên kết kép vòng.

Dưới đây là một cách cài đặt cho các thao tác cơ bản trên danh sách liên kết kép vòng.