Chương 4:
Danh sách liên kết (tt)

4.3 Danh sách liên kết mở rộng

4.3.1 Danh sách liên kết vòng

- Mô tả
- Cài đặt
- Thao tác

4.3.2 Danh sách liên kết kép

- Mô tả
- Cài đặt
- Thao tác

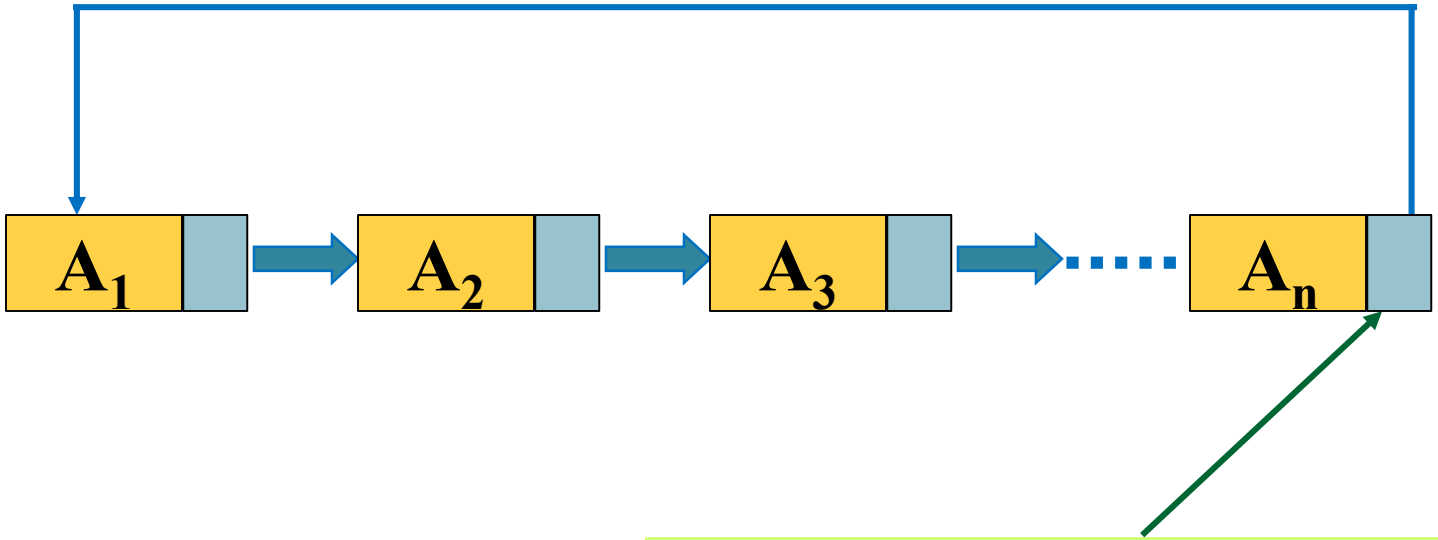
4.3.3 Danh sách liên kết đôi vòng

- Mô tả
- Cài đặt
- Thao tác

Circular Linked List

4.3.1 Circular Linked List - Mô tả

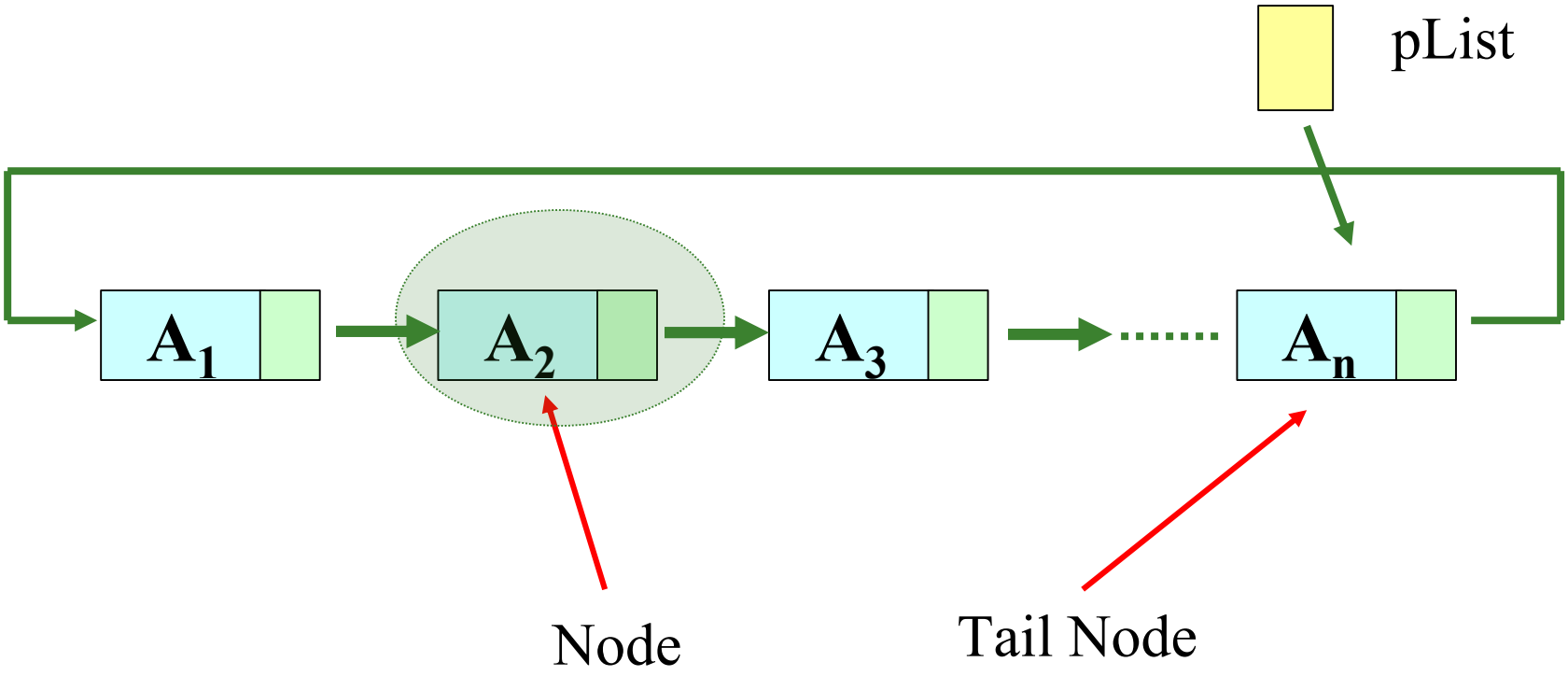
- Tương tự như danh sách liên kết đơn.
- Trường next của nút cuối chỉ đến đầu danh sách



Trường Next của nút cuối ko còn trỏ đến NULL, mà trỏ đến nút đầu

4.3.1 CLL - Mô tả

- Mô tả CLL
  - Sử dụng pList trỏ đến phần tử cuối của danh sách



4.3.1 CLL – Cài đặt

- Khai báo

```
typedef    struct     node
{
        DataType        info;
        node*           next;
} NODE;
typedef    NODE*      NodePtr;

NodePtr         pList;
```

pList trỏ nút cuối ds

4.3.1 CLL – Các thao tác

- Các thao tác cơ bản
  - Init
  - IsEmpty
  - ShowList

Phần minh hoạ sẽ dùng

DataType là int

- Bổ sung 1 phần tử mới vào danh sách
  - InsertFirst
  - InsertLast
  - InsertAfter
  - InsertBefore
- Loại bỏ 1 phần tử khỏi danh sách
  - DeleteFirst
  - DeleteLast
  - DeleteAfter
  - DeleteBefore
  - DeleteNode
- Các thao tác khác
  - Search
  - AddList
  - ClearList

4.3.1 CLL – Các thao tác

- **Init**: khởi tạo danh sách, ban đầu chưa có phần tử

```
void    Init(NodePtr      &pList)
{
        *pList = NULL;
}
```

- **IsEmpty:** Kiểm tra danh sách rỗng

```
int   IsEmpty(NodePtr      &pList)
{
      return (pList == NULL);
}
```

4.3.1 CLL – Các thao tác cơ bản

- **ShowList - Duyệt danh sách**
  - Duyệt từ đầu danh sách
  - Đến khi nào quay lại phần tử đầu thì dừng

4.3.1 CLL – Các thao tác cơ bản

- ShowList:

```
1. void    ShowList(NodePtr    &pList)
2. {  NodePtr   p;   //p con tro de duyet
3.    if (pList == NULL ) return;
4.    p = pList->next;   //duyet tu dau
5.    do
6.    {    ShowNode(p);//tac dong len nut
7.         p = p->next;//chuyen nut sau
8.    } while (p != pList->next);
9. }
```

4.3.1 CLL – Các thao tác cơ bản

- ShowNode: in ra trường info của nút

```
void      ShowNode(NodePtr  q)
{
      cout<< q->info<<"\t";
}
```

4.3.1 CLL – Bổ sung vào đầu ds

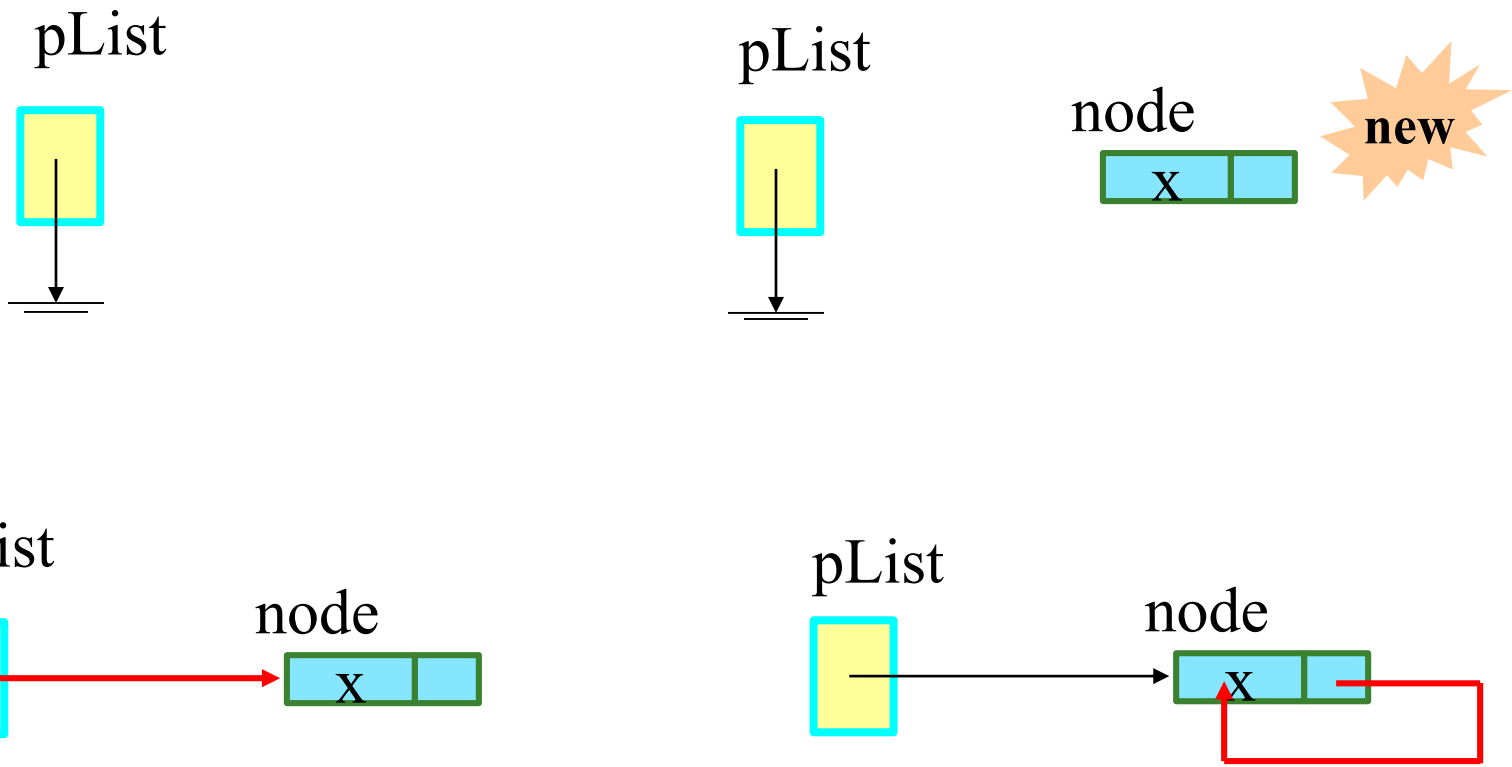
- Bổ sung vào đầu danh sách
  - Tạo nút mới

```
NodePtr  node;    //con tro tro nut moi
node = new    NODE; //xin cap phat dc
node -> info = x;  //gan gia tri
```

  - Nối vào danh sách

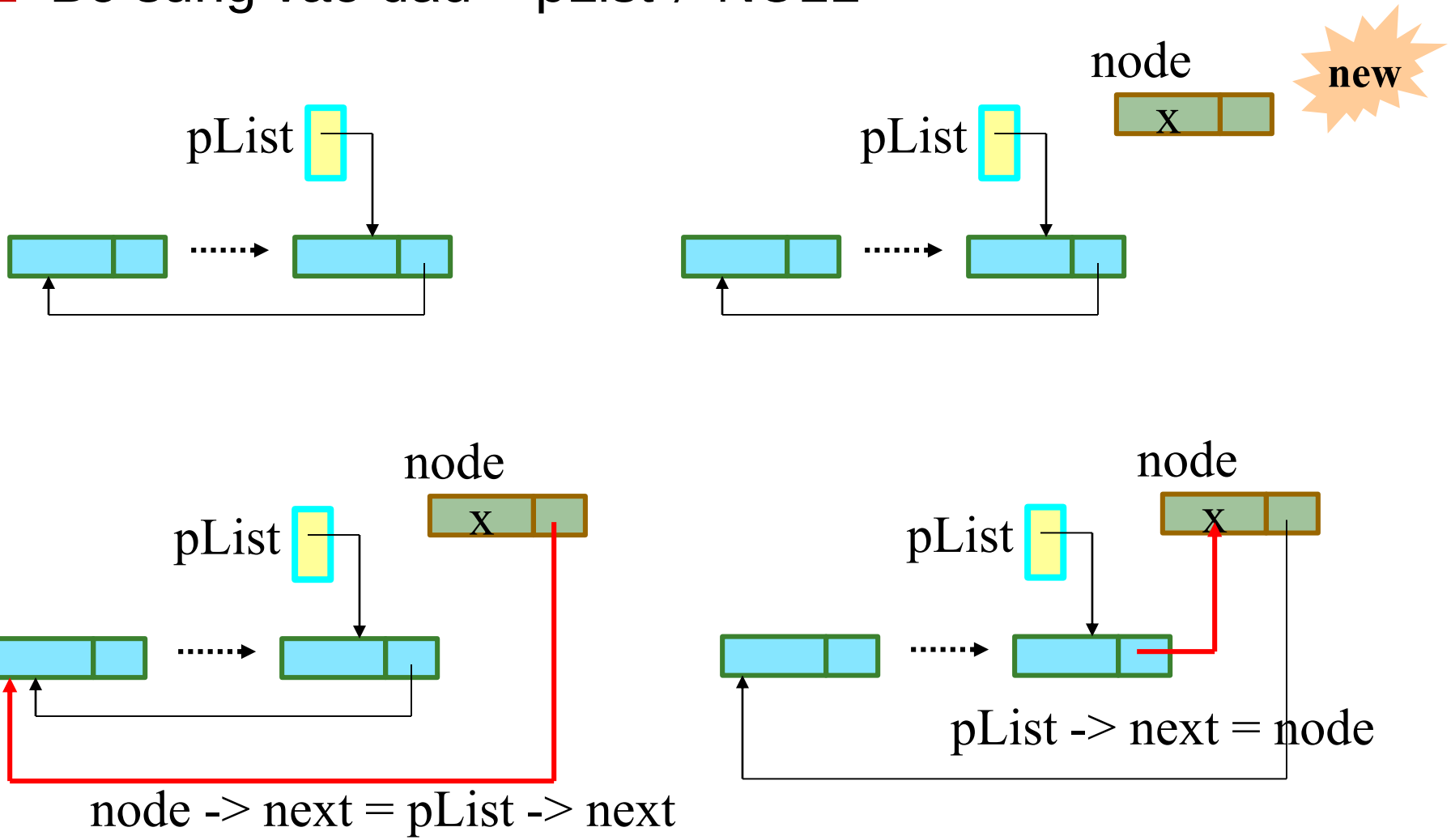
4.3.1 CLL – Bổ sung vào đầu

- Bổ sung vào đầu danh sách – pList = NULL



4.3.1 CLL – Bổ sung vào đầu

- Bổ sung vào đầu – pList ≠ NULL

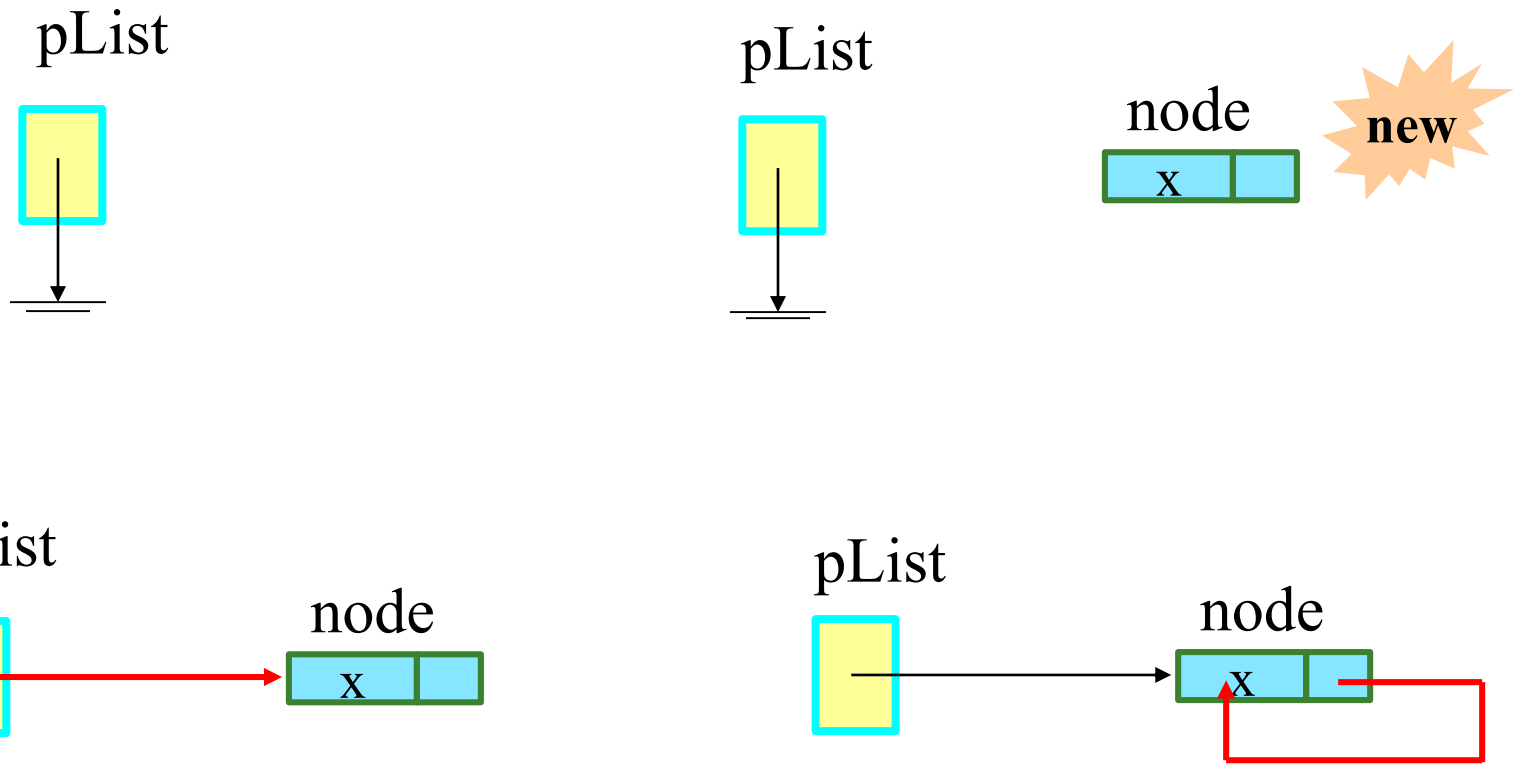


4.3.1 CLL – Bổ sung vào đầu ds

```
1. void    InsertFirst(NodePtr &pList, int x)
2. { NodePtr   node;
3.    node = new      NODE;
4.    node->info = x;
5.    if (pList == NULL)
6.    {    pList = node;
7.         pList->next = pList;      }
8.    else
9.    {    node->next = pList->next;
10.        pList->next = node;       }
11. }
```

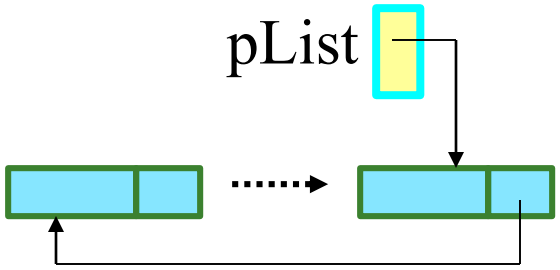
4.3.1 CLL - Bổ sung vào cuối ds

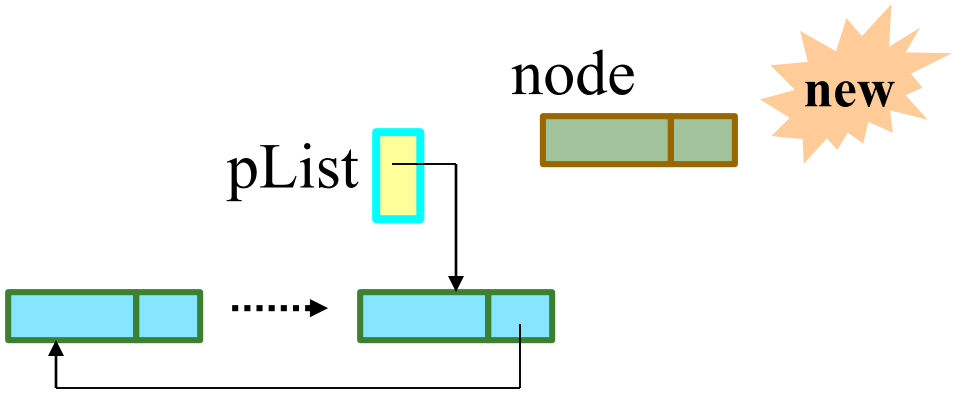
- InsertLast: pList = NULL

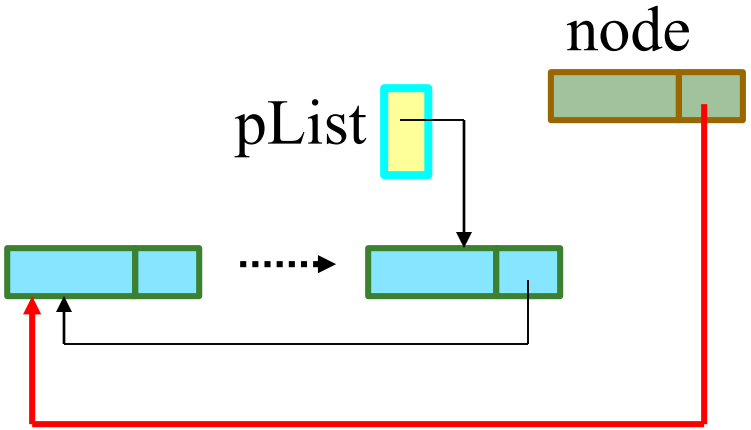


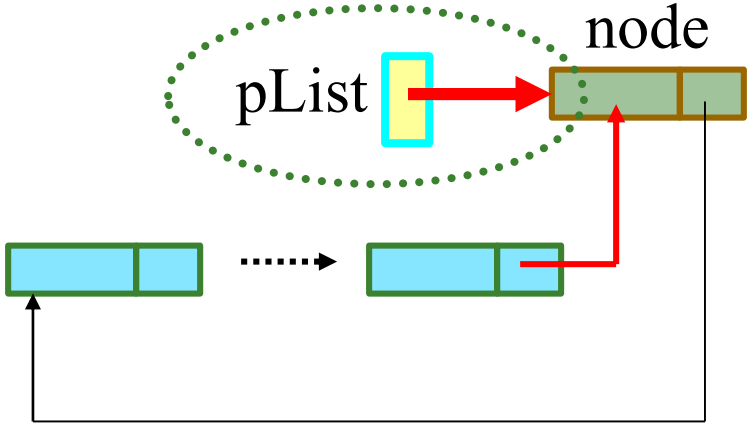
4.3.1 CLL - Bổ sung vào cuối ds

- InsertLast: pList ≠ NULL









4.3.1 CLL – Bổ sung vào cuối ds

```
1. void    InsertLast(NodePtr &pList, int x)
2. {  NodePtr    node;
3.    node = new  NODE;
4.    node->info = x;
5.    if (pList == NULL)
6.    {     pList = node;
7.          pList->next = pList;          }
8.    else
9.    {     node->next = pList->next;
10.         pList->next = node;
11.         pList = node;                 }
12.}
```

4.3.1 CLL – Bổ sung vào sau nút p

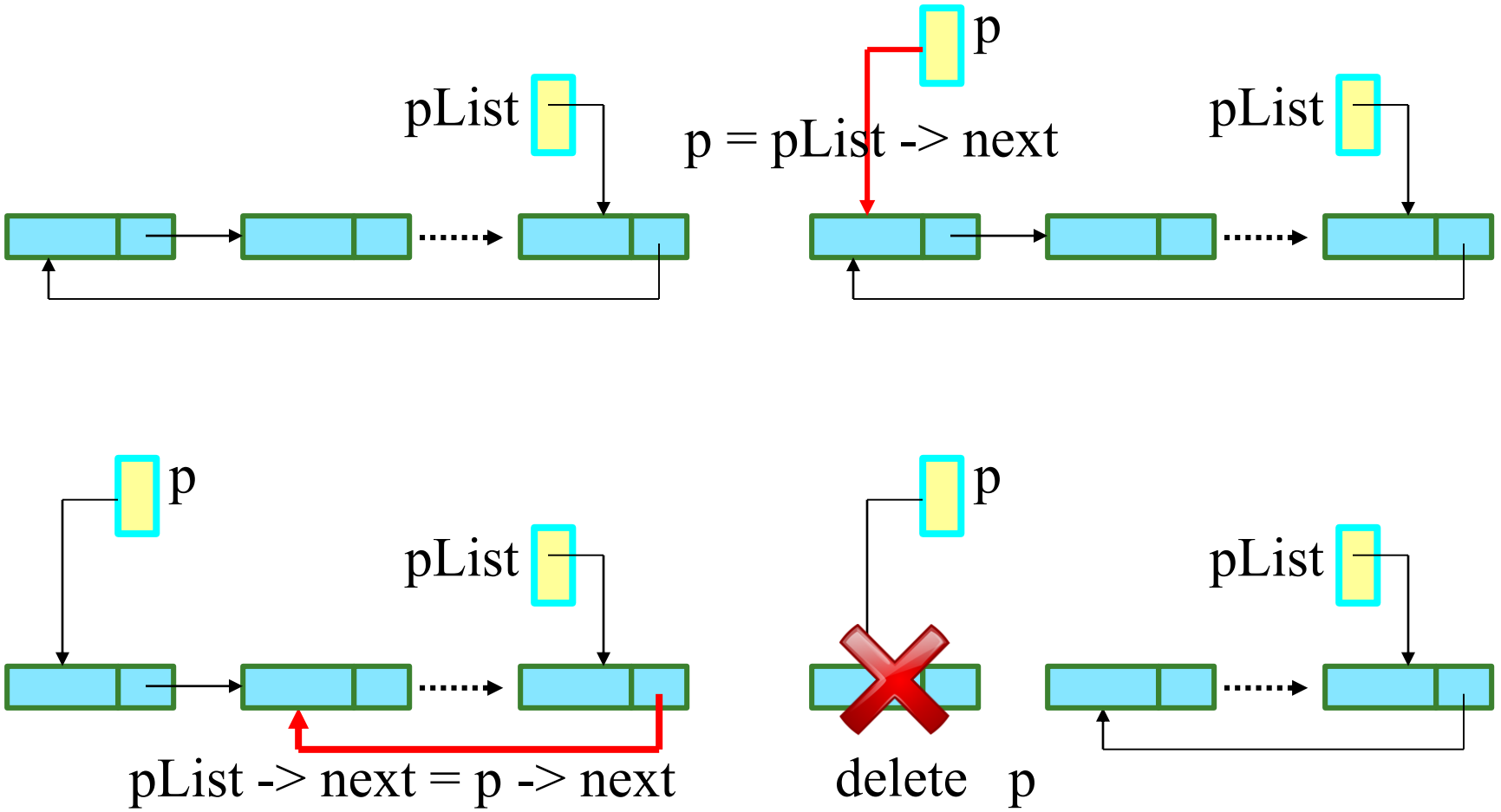
```
1. void    InsertAfter(NodePtr &p, int x)
2. {  NodePtr   node;
3.    node = new  NODE;
4.    node->info = x;
5.    if (p == NULL)
6.          return;
7.    else
8.    {     node->next = p->next;
9.          p->next = node;
10.   }
11.}
```

4.3.1 CLL – Loại bỏ nút đầu

- DeleteFirst: pList->next = pList

4.3.1 CLL – Loại bỏ nút đầu

- DeleteFirst: pList->next ≠ pList



4.3.1 CLL – Loại bỏ nút đầu

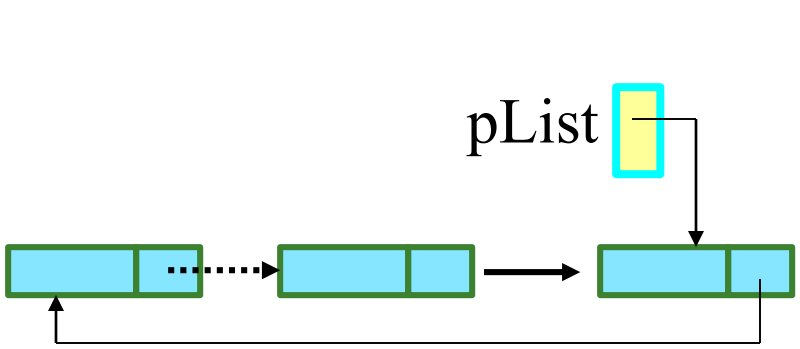
```
1. void DeleteFirst(NodePtr  &pList)
2. {  NodePtr p;        //p trỏ nút sẽ loại bỏ
3.    if (pList == NULL)          //ds rong
4.          return;
5.    else if (pList == pList->next) //co 1 nut
6.          {     delete     pList;
7.                pList = NULL;
8.          }
9.          else                  //co nhieu hon 1 nut
10.         {     p = pList->next;
11.               pList->next = p->next;
12.               delete     p;
13.         }
14.}
```

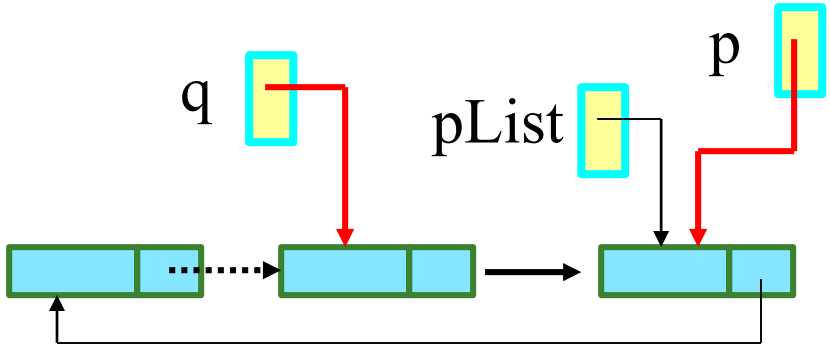
4.3.1 CLL – Loại bỏ nút cuối

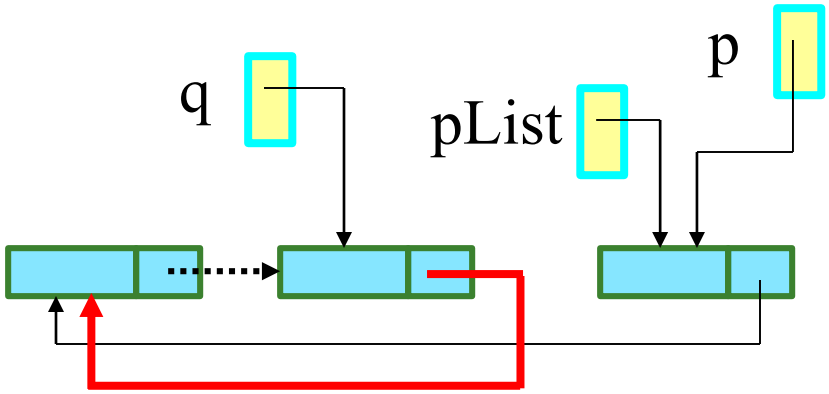
- DeleteLast: pList->next = pList

4.3.1 CLL – Loại bỏ nút cuối

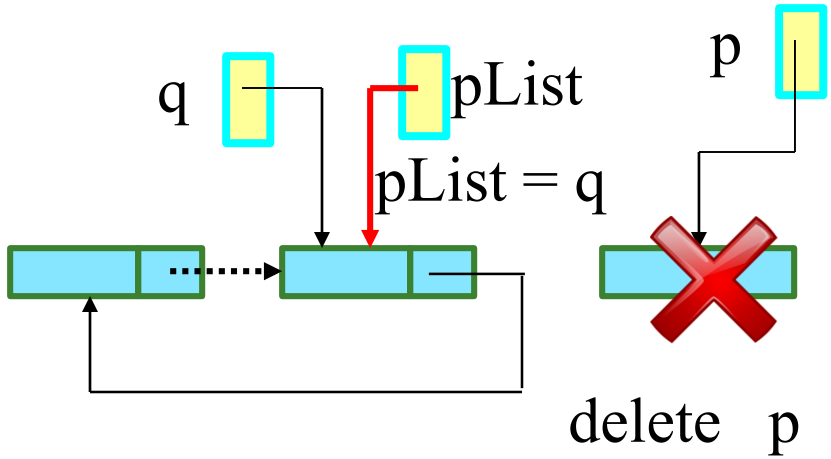
- DeleteLast: pList->next ≠ pList







q -> next = pList -> next



4.3.1 CLL – Loại bỏ nút cuối

```
1. void    DeleteLast (NodePtr &pList)
2. {  NodePtr    p; //p trỏ nút sẽ loại bỏ
3.    if (pList == NULL)     //ds rỗng
4.          return;
5.    else
6.          if (pList == pList->next) //co 1 nut
7.          {     delete     pList;
8.                pList = NULL;
9.          }
```

4.3.1 CLL – Loại bỏ nút cuối

```
10.   else
11.   {     p = pList;
12.         //tim den nut truoc nut cuoi
13.         NodePtr   q = pList -> next;
14.         while ( q -> next != pList)
15.               q = q -> next;
16.         //loai bo
17.         q -> next = pList -> next;
18.         pList = q;
19.         delete    q;
20.   }
21.}
```

4.3.1 CLL - Loại bỏ nút p

```
1. void   DeleteNode(NodePtr &pList, NodePtr &p)
2. {  if (p == NULL)          return;
3.    else  if ( p == pList)
4.              DeleteLast(pList);
5.          else      // tìm đến nút trước p
6.          {   NodePtr    q = pList->next;
7.              while ( q -> next != p)
8.                    q = q -> next;
9.              //loại bỏ
10.             q -> next = p->next;
11.             delete    p;                }
12.}
```

4.3.1 CLL – Loại bỏ nút sau p

```
void DeleteAfter(NodePtr &pList, NodePtr &p)
{



}
```

4.3.1 CLL - Loại bỏ nút trước p

```
void   DeleteBefore(NodePtr &pList, NodePtr &p)
{




}
```

4.3.1 CLL – Tìm kiếm

- Search:
  - Xuất phát từ đầu danh sách
  - Nếu tìm thấy trả về địa chỉ nút đó và dừng
  - Ngược lại qua phần tử tiếp theo
  - Điều kiện dừng khi quay lại phần tử đầu tiên
  - Không tìm thấy trả về NULL

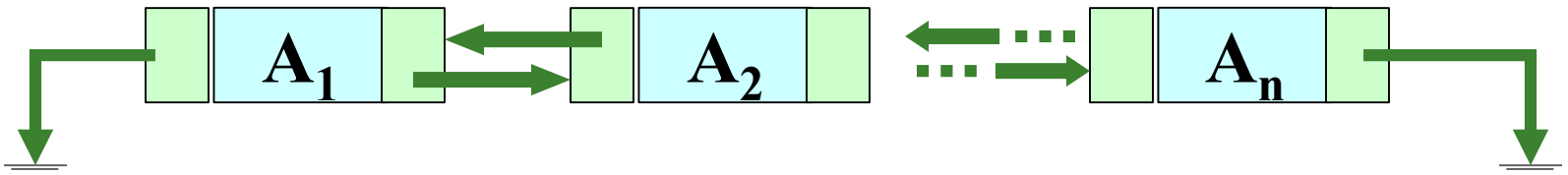
4.3.1 CLL – Tìm kiếm

```
1. NodePtr      Search (NodePtr &pList, int x)
2. {
       NodePtr p;  //p con trỏ tìm kiếm
3.     if (pList ==NULL) return NULL;
4.     p = pList->next;     //tìm từ đầu ds
5.     do {
6.           p = p->next;
7.     } while (p->info!=x && p!=pList>next);
8.     if ( p->info == x)   return p;
9.     return NULL;
10.}
```

Doubly Linked List

4.3.2 Doubly Link List - Mô tả

- Cho phép di chuyển 2 chiều đến nút trước và sau.
  - Liên kết nút trước là: prev
  - Liên kết nút sau là: next
- Nút đầu có prev là NULL
- Nút cuối có next là NULL



4.3.2 Doubly Linked List – Cài đặt

- Khai báo

```
typedef struct node
{
        DataType        info;
        struct node *   prev;
        struct node *   next;
}NODE;
typedef NODE *          NodePtr;

NodePtr         pHead;
```

trỏ đến nút trước

trỏ đến nút sau

pHead quản lý ds kép

4.3.2 DLL– Các thao tác

- Các thao tác cơ bản
  - Init
  - IsEmpty
- Duyệt danh sách: ShowList, ShowReverse
- Bổ sung 1 phần tử mới vào danh sách
  - InsertFirst
  - InsertLast
  - InsertAfter
  - InsertBefore
- Loại bỏ 1 phần tử khỏi danh sách
  - DeleteFirst
  - DeleteLast
  - DeleteAfter
  - DeleteBefore
  - DeleteNode
- Các thao tác khác
  - Search
  - ClearList

Phần minh hoạ sẽ dùng

DataType là int

4.3.2 DLL – Duyệt danh sách

- ShowList:
  - Duyệt từ đầu danh sách
  - Đến khi nào hết danh sách thì dừng

4.3.2 DLL – Duyệt danh sách

- ShowList:

```
1. void    ShowList(NodePtr    pHead)
2. {
3.    NodePtr   p;
4.    if (pHead == NULL ) return;
5.    p = pHead;
6.    while (p != NULL)
7.    {     ShowNode(p);
8.          p = p->next;
9.    }
10.}
```

4.3.2 DLL – Duyệt danh sách

- ShowReverse:
  - Duyệt từ cuối danh sách
  - Đến khi nào hết danh sách thì dừng

4.3.2 DLL – Duyệt danh sách

- ShowReverse:

```
1. void    ShowReverse(NodePtr pHead)
2. {  if (pHead == NULL ) return;
3.    NodePtr   p = pHead;
4.    while (p -> next != NULL) p = p->next;
5.    NodePtr   q = p;
6.    while (q != NULL)
7.    {     ShowNode(q);
8.          q = q->prev;
9.    }
10.}
```

4.3.2 DLL – Bổ sung vào đầu ds

```
1. void    InsertFirst(NodePtr &pHead, int x)
2. {  //tao nut moi
3.    NodePtr   node;
4.    node = new  NODE;
5.    node->info = x;
6.    //noi vao danh sach
7.    if (pHead == NULL) //t/hop ds rong
8.    {     pHead = node;
9.          pHead->prev =pHead->next = NULL;
10.   }
```

4.3.2 DLL – Bổ sung vào đầu ds

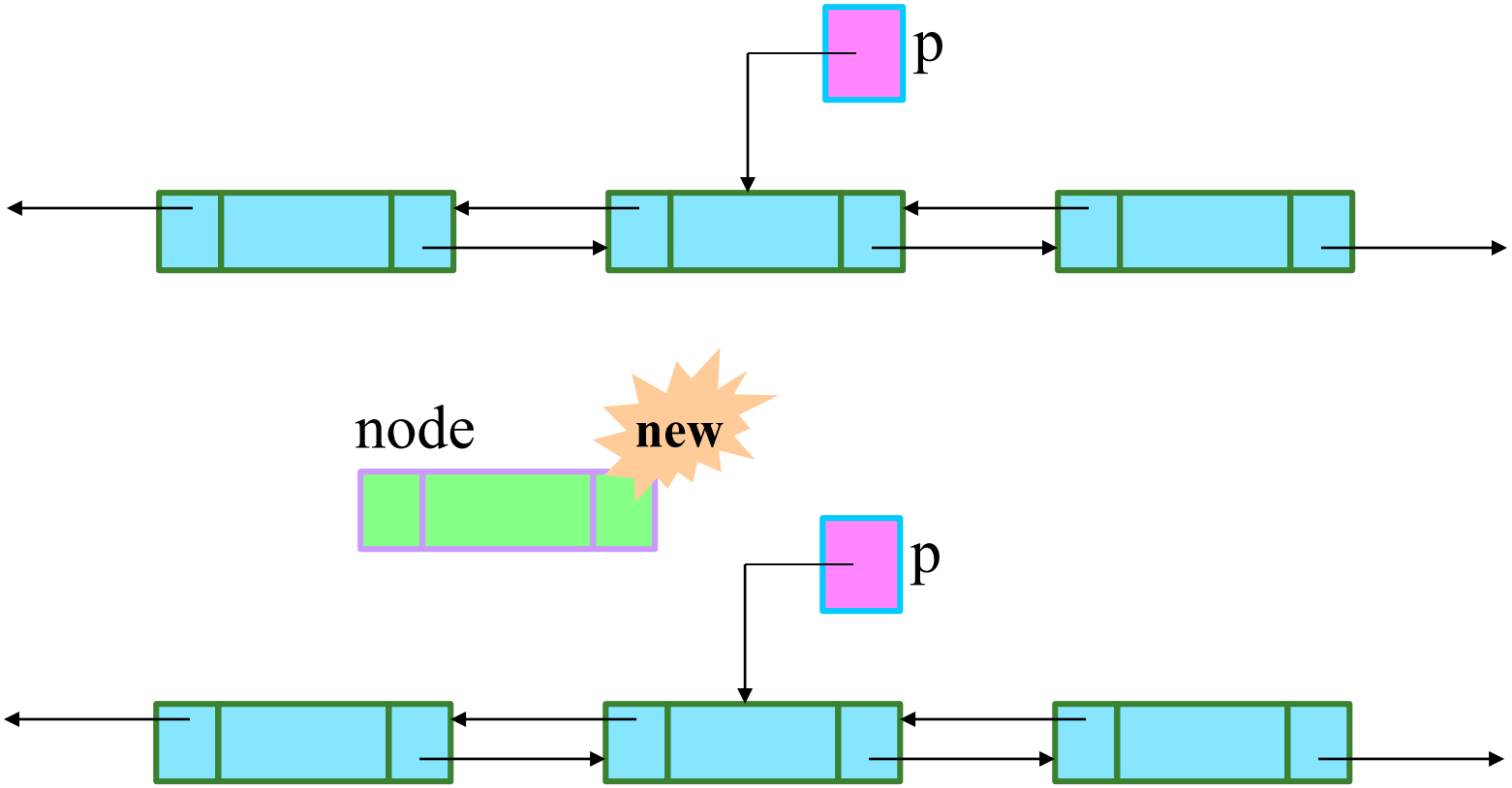
```
11.   else
12.   {     //noi node voi pHead
13.         node->next = pHead;
14.         pHead -> prev = node;
15.         node->prev = NULL;
16.         pHead = node;
17.   }
18.}
```

4.3.2 DLL - Bổ sung vào cuối ds

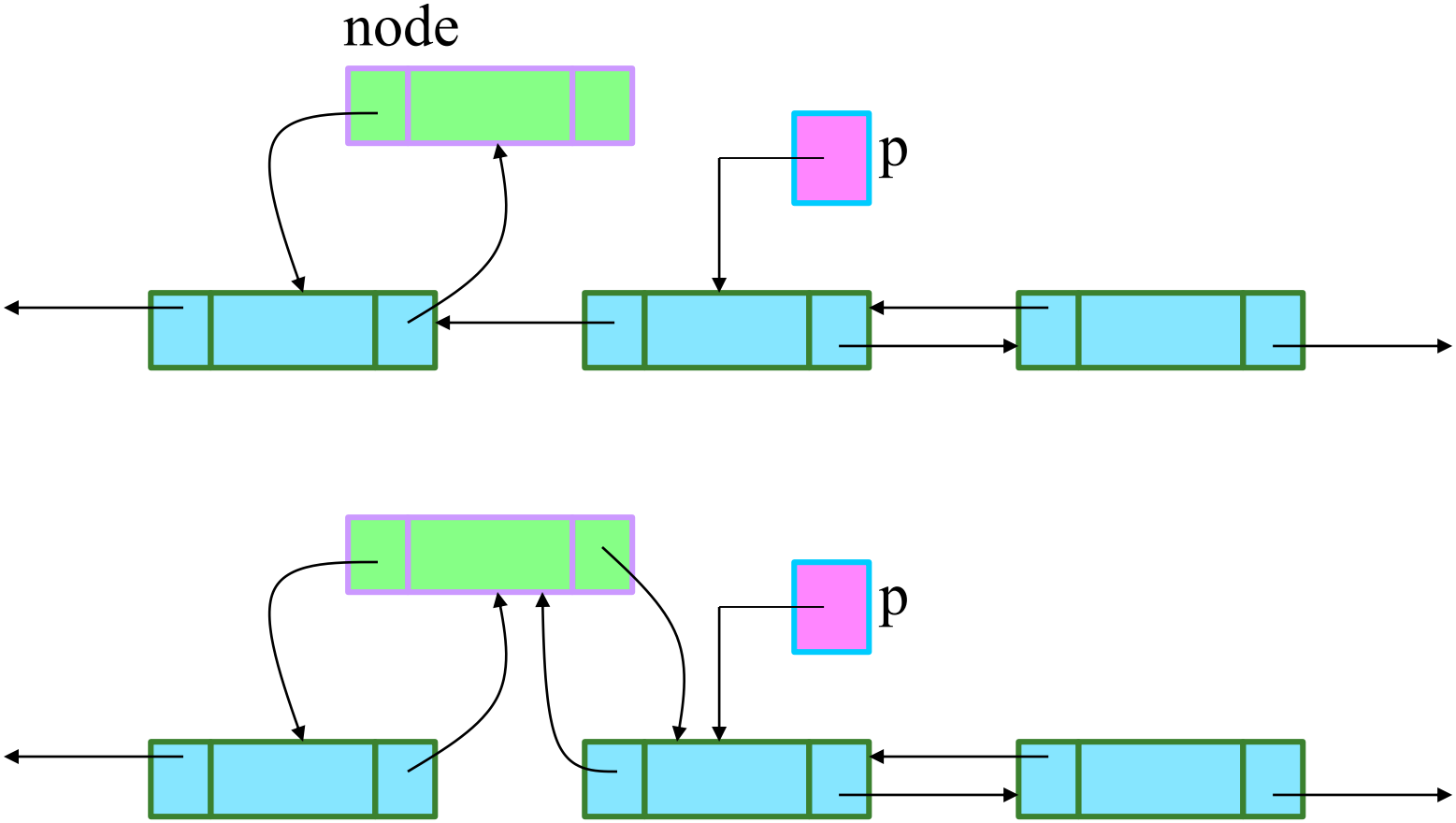
```
void InsertLast(NodePtr &pHead, int x)
{




}
```

4.3.2 DLL - Bổ sung vào trước nút p

- InsertBefore: chèn vào trước nút p



4.3.2 DLL - Bổ sung vào trước nút p



4.3.2 DLL - Bổ sung vào trước nút p

```
1. void InsertBefore(NodePtr &pHead, NodePtr
   &p, int x)
2. {  NodePtr    node, left;
3.    if (p == NULL) return;
4.    if (p == pHead)InsertFirst(pHead,x);
5.    else
6.    {     //tao nut moi
7.          node = new   NODE;
8.          node->info = x;
9.          left = p->prev;
```

4.3.2 DLL - Bổ sung vào trước nút p

```
10.         //noi left voi node
11.         left->next = node;
12.         node->prev = left;
13.         //noi node voi p
14.         node->next = p;
15.         p->prev = node;
16.   }
17.}
```

4.3.2 DLL - Bổ sung vào sau nút p

```
void InsertAfter(NodePtr &pHead, NodePtr &p,
int x)
{




}
```

4.3.2 DLL - Loại bỏ nút đầu ds

```
1. void    DeleteFirst (NodePtr &pHead)
2. {
3.    NodePtr  p;  //p tro nut loai bo
4.    if (pHead == NULL)
5.         return;
6.    else
7.         if (pHead ->prev == pHead->next)
8.         {
9.              delete    pHead;
10.             pHead = NULL;
11.        }
```

4.3.2 DLL - Loại bỏ nút đầu ds

```
12.   else
13.   {
14.         p = pHead;   //p tro nut dau
15.         pHead = pHead ->next;
16.         pHead -> prev = NULL;
17.         delete          p; //giai phong p
18.   }
19.}
```

4.3.2 DLL - Loại bỏ nút cuối

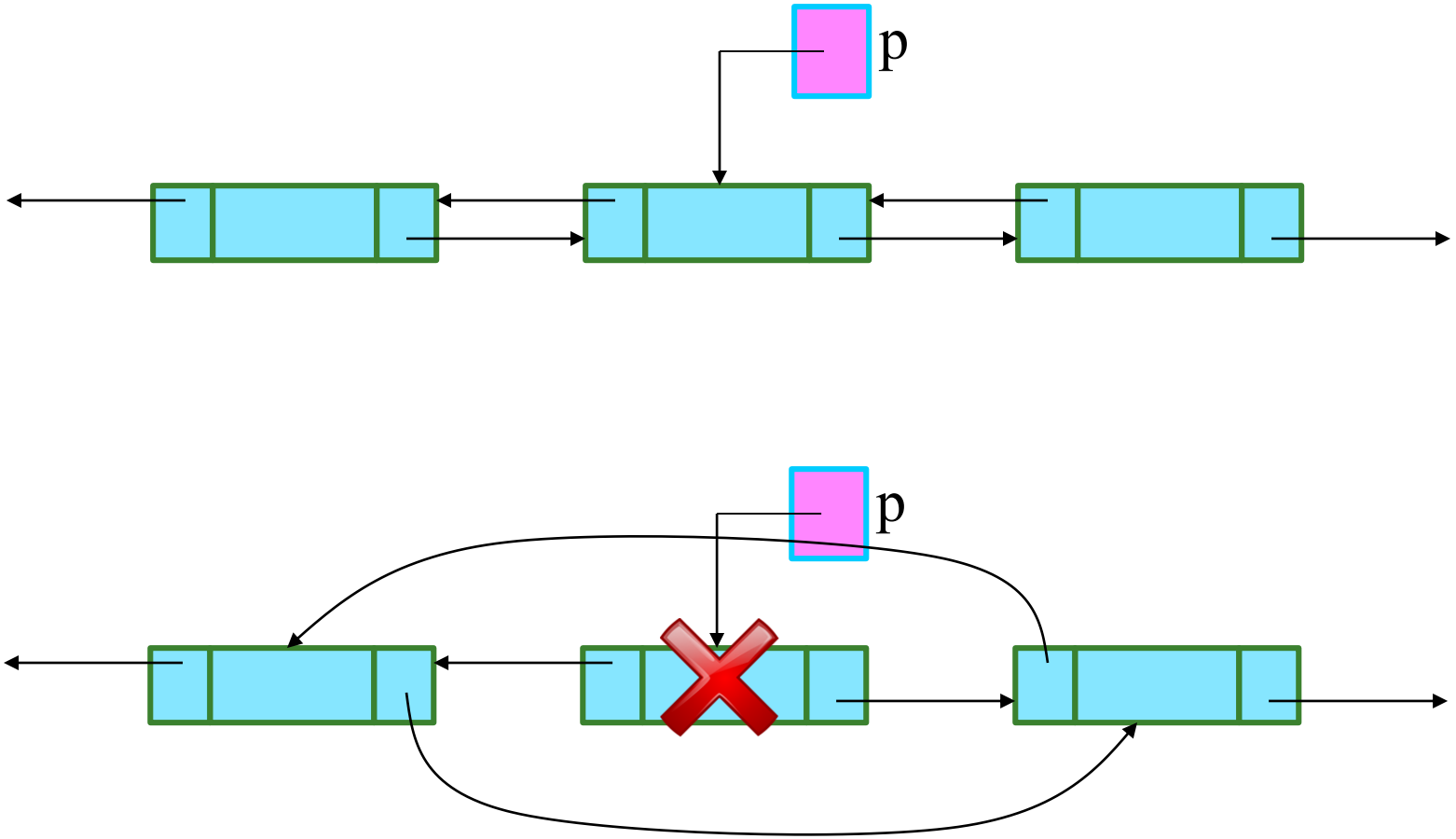
```
1. void    DeleteLast (NodePtr &pHead)
2. {  if (pHead == NULL)
3.          return;
4.    else
5.          if (pHead ->prev == pHead->next)
6.          {
7.                delete     pHead;
8.                pHead = NULL;
9.          }
10.         else
11.         {     NodePtr  p;
12.               p = pHead;
```

4.3.2 DLL - Loại bỏ nút cuối

```
13.         while (p-> next != NULL)
14.               p = p ->next;
15.         NodePtr   q = p -> prev;
16.         q -> next = NULL;
17.         delete    p;
18.   }
19.}
```

4.3.2 DLL – Loại bỏ nút p

- DeleteNode: xoá nút p



4.3.2 DLL - Loại bỏ nút p

```
1. void    DeleteNode(NodePtr &pHead, NodePtr &p)
2. {  NodePtr      left, right;
3.    if (p == NULL)     return;
4.    if (p==pHead)      DeleteFirst(pHead);
5.    else
6.    {     left = p ->prev;
7.          right = p->next;
8.          left->next = right;
9.          if (right != NULL)
10.               right->prev = left;
11.         delete    p;
12.   }
13.}
```

4.3.2 DLL – Tìm kiếm

- Search:
  - Xuất phát từ đầu danh sách
  - Nếu tìm thấy trả về địa chỉ nút đó
  - Ngược lại qua phần tử tiếp theo
  - Điều kiện dừng khi hết danh sách
  - Không tìm thấy trả về NULL

4.3.2 DLL – Tìm kiếm

```
1. NodePtr  Search (NodePtr pHead, int x)
2. {
3.    NodePtr   p;
4.    if (pHead ==NULL) return NULL;
5.    p = pHead;
6.    while (p->info != x && p != NULL)
7.          p = p->next;
8.    if ( p->info == x)  return    p;
9.    else return   NULL;
10.}
```

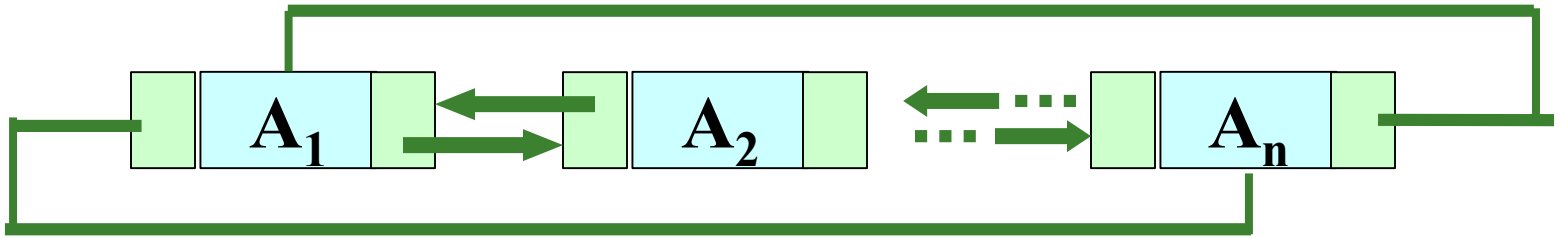
4.3.2 DLL – Các thao tác

- Các thao tác còn lại SV tự làm!

Circular Doubly Linked List

4.3.3 Circular Doubly Link List

- Cho phép di chuyển 2 chiều đến nút trước và sau.
  - Liên kết nút trước là: prev
  - Liên kết nút sau là: next
- Trường prev của nút đầu trỏ đến nút cuối
- Trường next của nút cuối trỏ đến nút đầu



4.3.3 CDLL – Cài đặt

- Khai báo

```
typedef struct node
{
        DataType        info;
        node*           prev;
        node*           next;
} NODE;
typedef   NODE *        NodePtr;

NodePtr         pHead;
```

trỏ đến nút trước

trỏ đến nút sau

pHead quản lý ds đôi vòng

4.3.3 CDLL – Các thao tác

- Các thao tác cơ bản
  - Init
  - IsEmpty
- Duyệt danh sách: ShowList, ShowReverse
- Bổ sung 1 phần tử mới vào danh sách
  - InsertFirst
  - InsertLast
  - InsertAfter
  - InsertBefore
- Loại bỏ 1 phần tử khỏi danh sách
  - DeleteFirst
  - DeleteLast
  - DeleteAfter
  - DeleteBefore
  - DeleteNode
- Các thao tác khác
  - Search
  - ClearList

Phần minh hoạ sẽ dùng

DataType là int

4.3.3 CDLL – Duyệt danh sách

- ShowList:
  - Duyệt từ đầu danh sách
  - Đến khi nào quay lại nút đầu thì dừng

4.3.3 CDLL – Duyệt danh sách

- ShowList:

```
1. void    ShowList(NodePtr    pHead)
2. {
3.    NodePtr   p;
4.    if (pHead == NULL ) return;
5.    p = pHead;
6.    do
7.    {     ShowNode(p);
8.          p = p->next;
9.    } while (p != pHead);
10.}
```

4.3.3 CDLL – Duyệt danh sách

- ShowReverse:
  - Duyệt từ cuối danh sách
  - Bắt đầu từ nút cuối quay trở về đầu
  - Đến khi nào quay lại nút cuối thì dừng

4.3.3 CDLL – Duyệt danh sách

- ShowReverse:

```
1. void   ShowReverse(NodePtr    pHead)
2.  {if (pHead == NULL ) return;
3.   NodePtr  p = pHead->prev;
4.   do
5.   {          ShowNode(p);
6.              p = p->prev;
7.   } while (p != pHead->prev);
8. }
```

4.3.3 CDLL – Bổ sung vào đầu ds

```
1. void    InsertFirst(NodePtr &pHead, int x)
2. {  //tao nut moi
3.    NodePtr   node;
4.    node = new      NODE;
5.    node->info = x;
6.    // noi vao danh sach
7.    if (pHead == NULL)
8.    {     pHead = node;
9.          pHead->prev =pHead->next = pHead;
10.   }
```

4.3.3 CDLL – Bổ sung vào đầu ds

```
11.   else
12.   {     NodePtr  q;  //q tro nut cuoi
13.         q = pHead -> prev;
14.         //noi node voi pHead
15.         node->next = pHead;
16.         pHead -> prev = node;
17.         //noi node voi q
18.         q -> next = node;
19.         node->prev = q;
20.         pHead = node;
21.   }
22.}
```

4.3.3 CDLL - Bổ sung vào cuối ds

```
void    InsertLast(NodePtr &pHead, int x)
{    //tao nut moi
     NodePtr   node;
     node = new   NODE;
     node->info = x;
     //noi vao danh sach
     if (pHead == NULL)
     {    pHead = node;
          pHead->prev = pHead->next = pHead;
     }
```

4.3.3 CDLL - Bổ sung vào cuối ds

```
    else
    {     NodePtr   q;   //q trỏ vào nút cuối
          q = pHead ->prev;
          node->next = pHead;
          pHead -> prev = node;

          q -> next = node;
          node->prev = q;
    }
}
```

4.3.3 CDLL - Bổ sung vào sau nút p

```
void  InsertAfter(NodePtr &pHead, NodePtr & p, int x)
{




}
```

4.3.3 CDLL - Loại bỏ nút đầu

```
void DeleteFirst (NodePtr &pHead)
{
      NodePtr p;
      if (pHead == NULL)
            return;
      else
            if (pHead ->prev == pHead->next)
            {
                  delete    pHead;
                  pHead = NULL;
            }
```

4.3.3 CDLL - Loại bỏ nút đầu

```
        else
        {
                p = pHead;
                NodePtr  q = pHead -> prev;
                pHead = pHead ->next;

                pHead -> prev = q;
                q -> next = pHead;
                delete    p;
        }
}
```

4.3.3 CDLL - Loại bỏ nút cuối

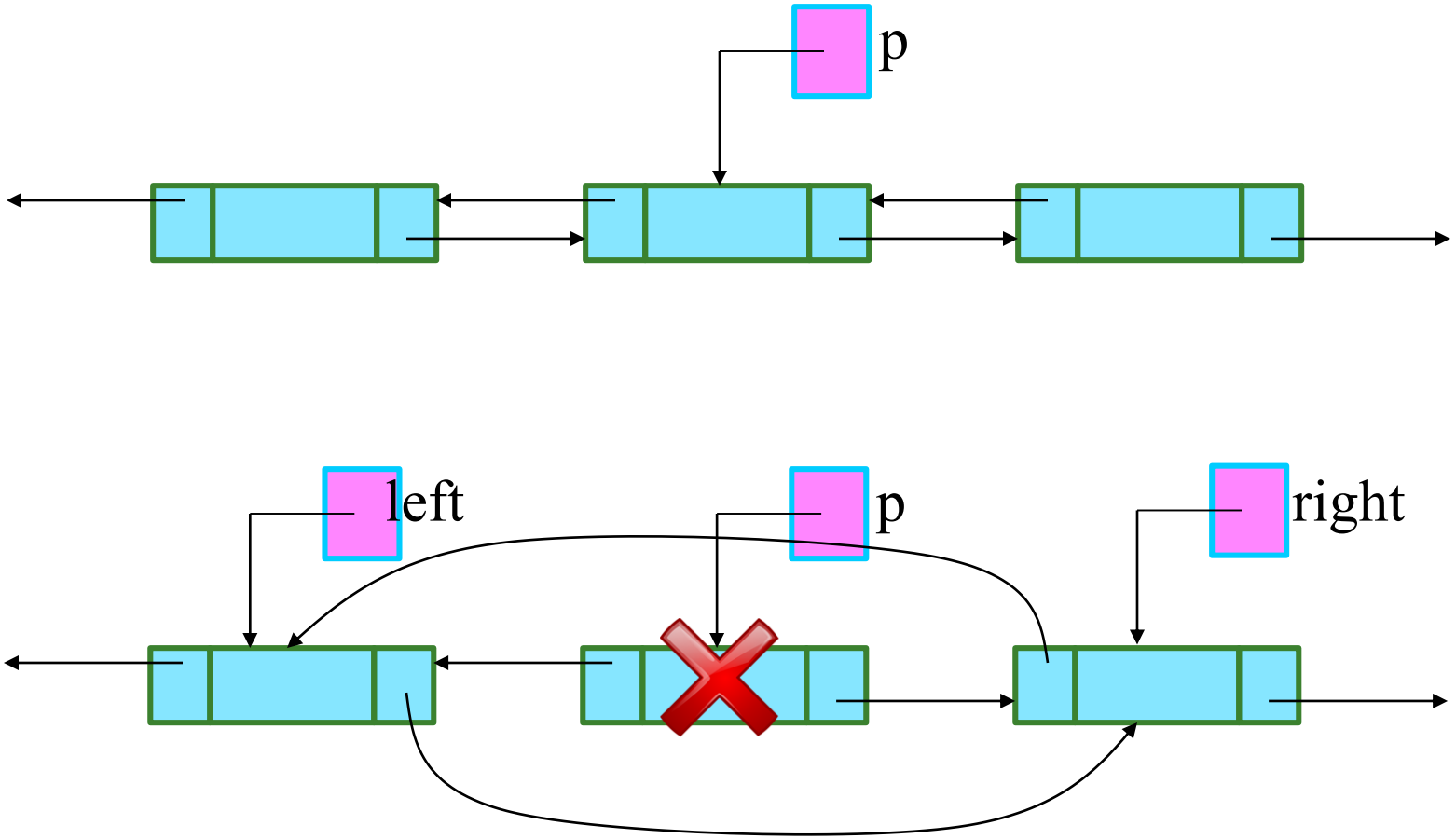
```
void DeleteLast (NodePtr &pHead) {
     if (pHead == NULL)
          return;
     else
          if (pHead ->prev == pHead->next)
          {
               delete    pHead;
               pHead = NULL;
          }
```

4.3.3 CDLL - Loại bỏ nút cuối

```
        else
        {     NodePtr  p = pHead -> prev;
              NodePtr  q = p -> prev;

              q -> next = pHead;
              pHead -> prev = q;
              delete    p;
        }
}
```

4.3.3 CDLL - Loại bỏ nút p

- DeleteNode: xoá nút p



4.3.3 CDLL - Loại bỏ nút p

```
void DeleteNode(NodePtr &pHead, NodePtr &p)
{
      NodePtr       left, right;
      if (p == NULL)     return;
      if (p==pHead)      DeleteFirst(pHead);
      else {
            left = p ->prev;
            right = p->next;
```

4.3.3 CDLL - Loại bỏ nút p

```
            left->next = right;
            if (right != NULL)
                  right->prev = left;
            delete    p;
      }
}
```

4.3.3 CDLL – Tìm kiếm

- Search:
  - Xuất phát từ đầu danh sách
  - Nếu tìm thấy trả về địa chỉ nút đó
  - Ngược lại qua phần tử tiếp theo
  - Điều kiện dừng khi quay lại nút đầu
  - Không tìm thấy trả về NULL

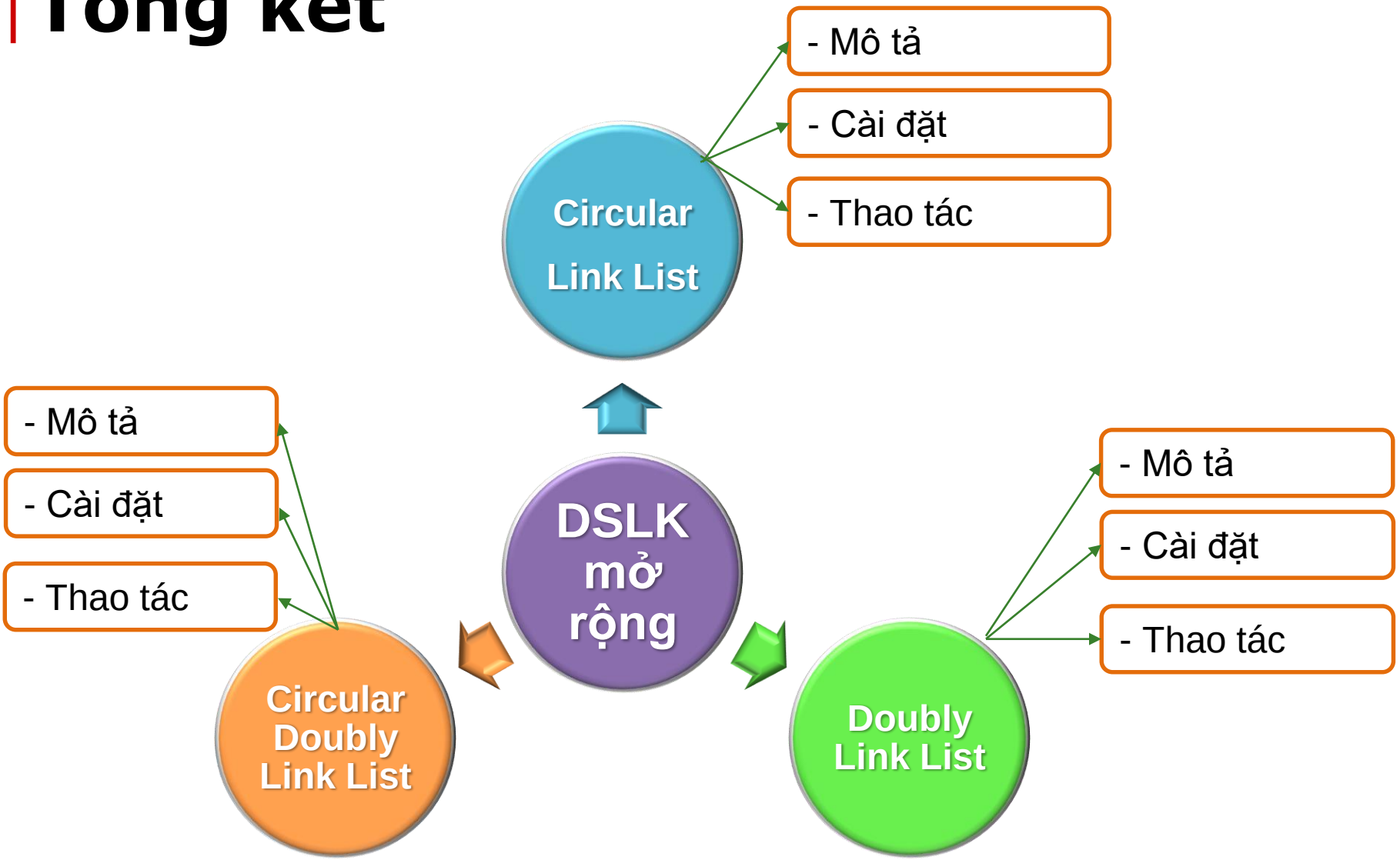
4.3.3 CDLL – Tìm kiếm

```
NodePtr  Search (NodePtr pHead, int x) {
      NodePtr  p;
      if (pHead ==NULL) return NULL;
      p = pHead;
      do {      p = p->next;
      } while (p->info != x && p != pHead);
      if ( p->info == x)      return   p;
      return NULL;
}
```

4.3.3 CDLL – Các thao tác

- Các thao tác còn lại SV tự làm!

Tổng kết


- Mô tả
- Cài đặt
- Thao tác
Circular Link List
- Mô tả
- Cài đặt
- Thao tác
DSLK mở rộng
- Mô tả
- Cài đặt
- Thao tác
Circular Doubly Link List
Doubly Link List

Bài tập

1. Cho một danh sách nối vòng có nút cuối được trỏ bởi pList. Trường info của các nút chứa giá trị nguyên.Viết giải thuật thực hiện các công việc sau:

a. Đếm số nút của danh sách.

b. Bổ sung một nút mới với thông tin x vào làm nút thứ k trong danh sách

c. Loại bỏ nút có giá trị y trong danh sách

Bài tập

2. Cho một danh sách nối vòng có nút cuối được trỏ bởi pList. Trường info của các nút chứa giá trị nguyên. p là con trỏ trỏ nút bất kỳ trong danh sách. Viết giải thuật thực hiện các công việc sau:

a. Bổ sung một nút mới với thông tin x vào trước nút p.

b. Loại bỏ nút sau nút p.

c. Tìm và in ra các nút chia 5 dư 2 trong danh sách

Bài tập

3. Cho hai danh sách nối vòng có nút cuối lần lượt được trỏ bởi pList1 và pList2. Trường info của các nút chứa giá trị nguyên. Viết giải thuật nối pList2 vào sau pList1 thành một danh sách nối vòng có nút cuối được trỏ bởi pList1.
4. Cho danh sách nối kép có nút đầu được trỏ bởi pHead. Trường info của các nút chứa giá trị thực và đã được sắp xếp tăng dần. Viết giải thuật bổ sung một nút mới với thông tin x vào danh sách sao cho vẫn đảm bảo thứ tự sắp xếp

Bài tập

5. Viết chương trình khai báo và thực hiện các thao tác trên danh sách nối kép, có nút đầu được trỏ bởi pHead, trường info của các nút chứa giá trị nguyên.
6. Viết chương trình khai báo và thực hiện các thao tác trên danh sách nối đôi vòng, có nút đầu được trỏ bởi pHead, trường info của các nút chứa giá trị nguyên.

Tài liệu tham khảo

[1]. Giáo trình Cấu trúc dữ liệu và giải thuật – Lê Văn Vinh, NXB Đại học quốc gia TP HCM, 2013

[2]. Cấu trúc dữ liệu & thuật toán, Đỗ Xuân Lôi, NXB Đại học quốc gia Hà Nội, 2010.

[3]. Trần Thông Quế, *Cấu trúc dữ liệu và thuật toán (phân tích và cài đặt trên C/C++)*, NXB Thông tin và truyền thông, 2018

[4]. Robert Sedgewick, *Cẩm nang thuật toán*, NXB Khoa học kỹ thuật, 2004 .

[5]. PGS.TS Hoàng Nghĩa Tý, *Cấu trúc dữ liệu và thuật toán*, NXB xây dựng, 2014