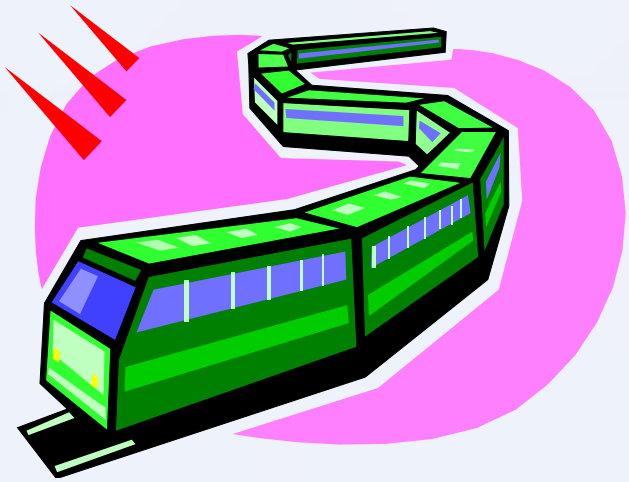
Bài 7

Danh sách liên kết (Linked List)

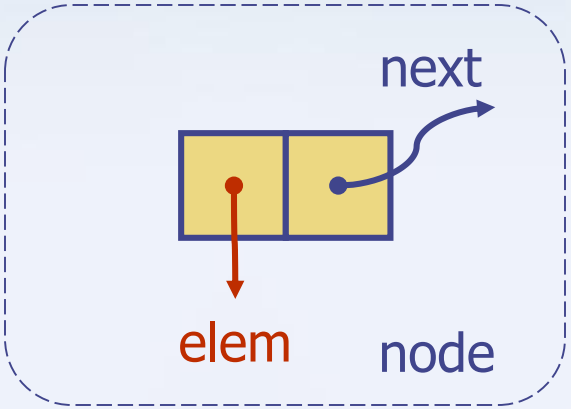
- Mô hình cấu trúc dữ liệu trừu tượng **Linked List** là một dãy các vị trí lữu trữ các đối tượng với số lượng tùy ý.
- Nó thiết lập một mối quan hệ **trước/sau** giữa các vị trí

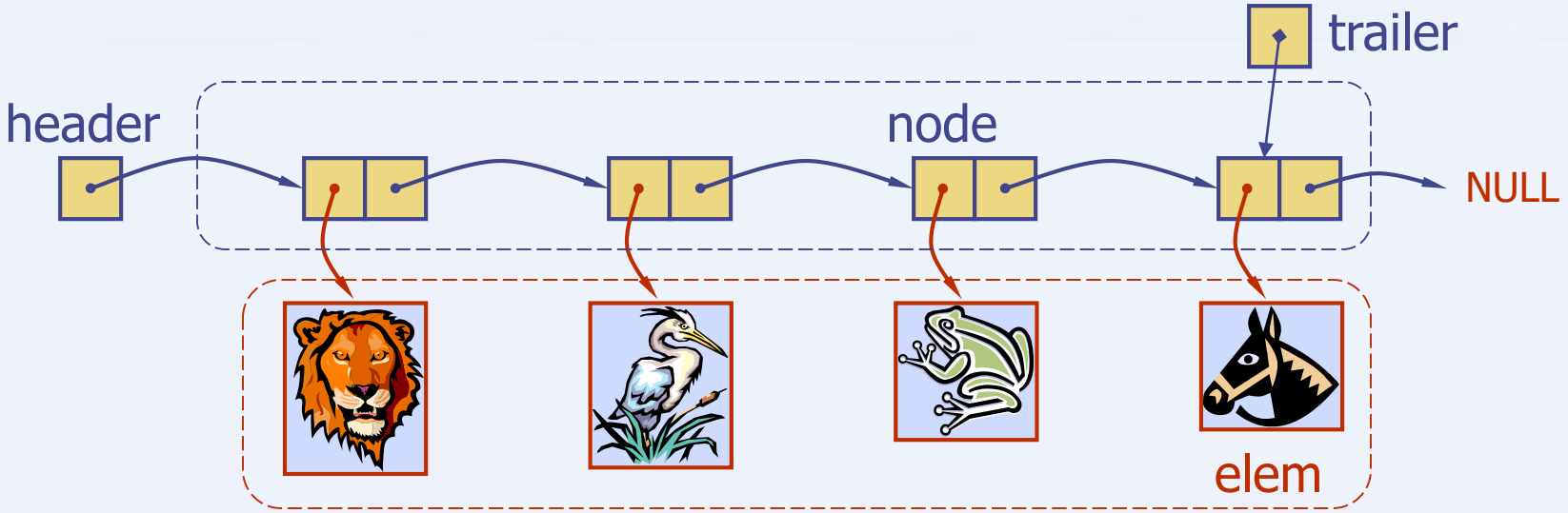
- Danh sách liên kết **đơn**
- Danh sách liên kết **kép**



Danh sách liên kết đơn

- Các nút (node) được cài đặt bao gồm:
  - Phần tử lưu trữ trong nó
  - Một liên kết đến nút kế tiếp
- Sử dụng một con trỏ header, trỏ vào node đầu danh sách và con trỏ trailer trỏ vào node cuối danh sách.

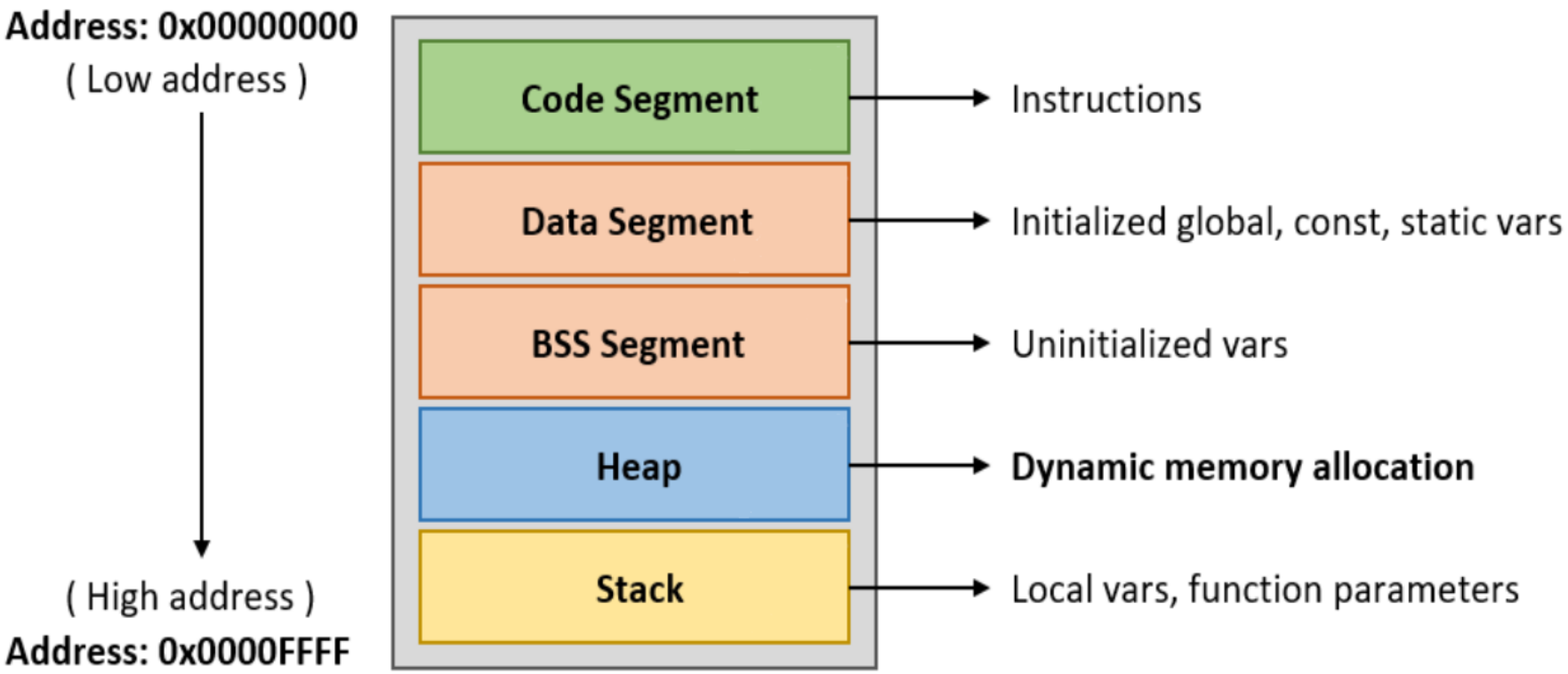




Tại sao phải dùng Linked List?

- Ưu và nhược điểm khi dùng mảng
  - Ưu điểm: truy xuất phần tử nhanh thông qua chỉ số
  - Nhược điểm: số phần tử cần được xác định trước → lãng phí ô nhớ khi chưa sử dụng đến. Cần dồn mảng khi xóa phần tử. Bị giới hạn kích thước mảng (64KB).
- Ưu và nhược điểm khi dùng Linked List
  - Ưu điểm: cấp phát động → không cần xác định trước số phần tử của danh sách. Không bị giới hạn kích thước, tận dụng các ô nhớ khi bộ nhớ bị phân mảnh
  - Nhược điểm: muốn truy xuất phần tử thì phải duyệt từ đầu danh sách

Tại sao phải dùng Linked List?



Phân loại vùng nhớ trong máy tính

Cấu trúc của một Node

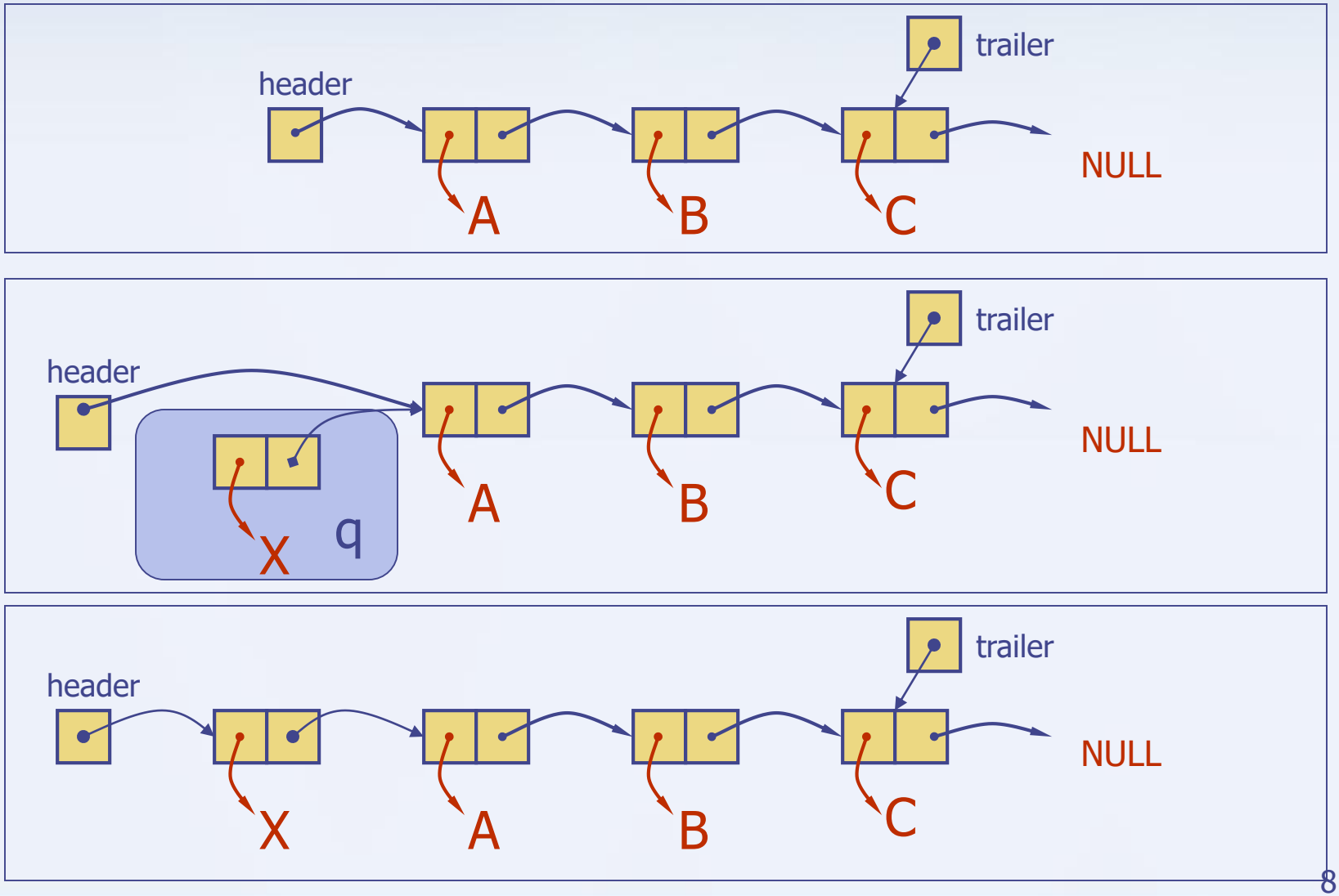
- Các thuộc tính
  - Element elem;
  - Node *next;
- Các phương thức
  - Node *getnext() - Trả lại địa chỉ của nút kế tiếp
  - Element getElem() - Trả lại giá trị phần tử lưu trữ trong nút
  - void setNext(Node *) - Gán địa chỉ cho thuộc tính next
  - void setElem(Element e) - Gán giá trị e cho thuộc tính elem

Cấu trúc danh sách liên kết đơn

- **Các thuộc tính:**
  - **Node *header**
  - **Node *trailer**
- **Các phương thức chung:**
  - int size(),
  - bool empty()
- **Các phương thức truy cập:**
  - front()
  - back()
- **Các phương thức cập nhật:**
  - void push_front(T e)
  - void push_back(T e)
  - void pop_front()
  - void pop_back()
- **Bộ lặp xuôi:**
  - begin
  - end
  - =
  - !=
  - ++
  - *
- **Chèn và xóa**
  - insert
  - erase

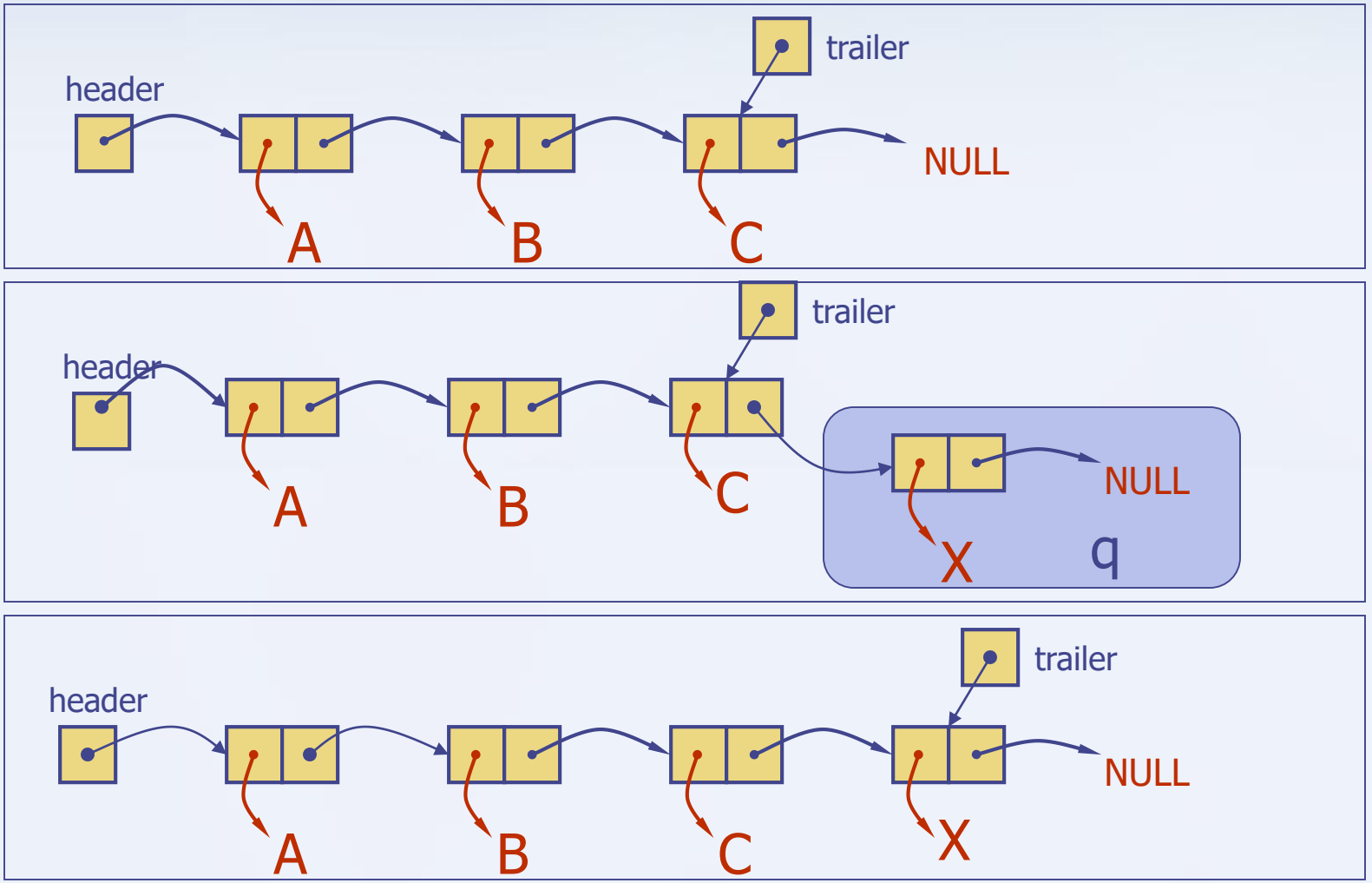
Thêm vào đầu

- Hình ảnh phép toán push_front(), phép toán trả lại vị trí q



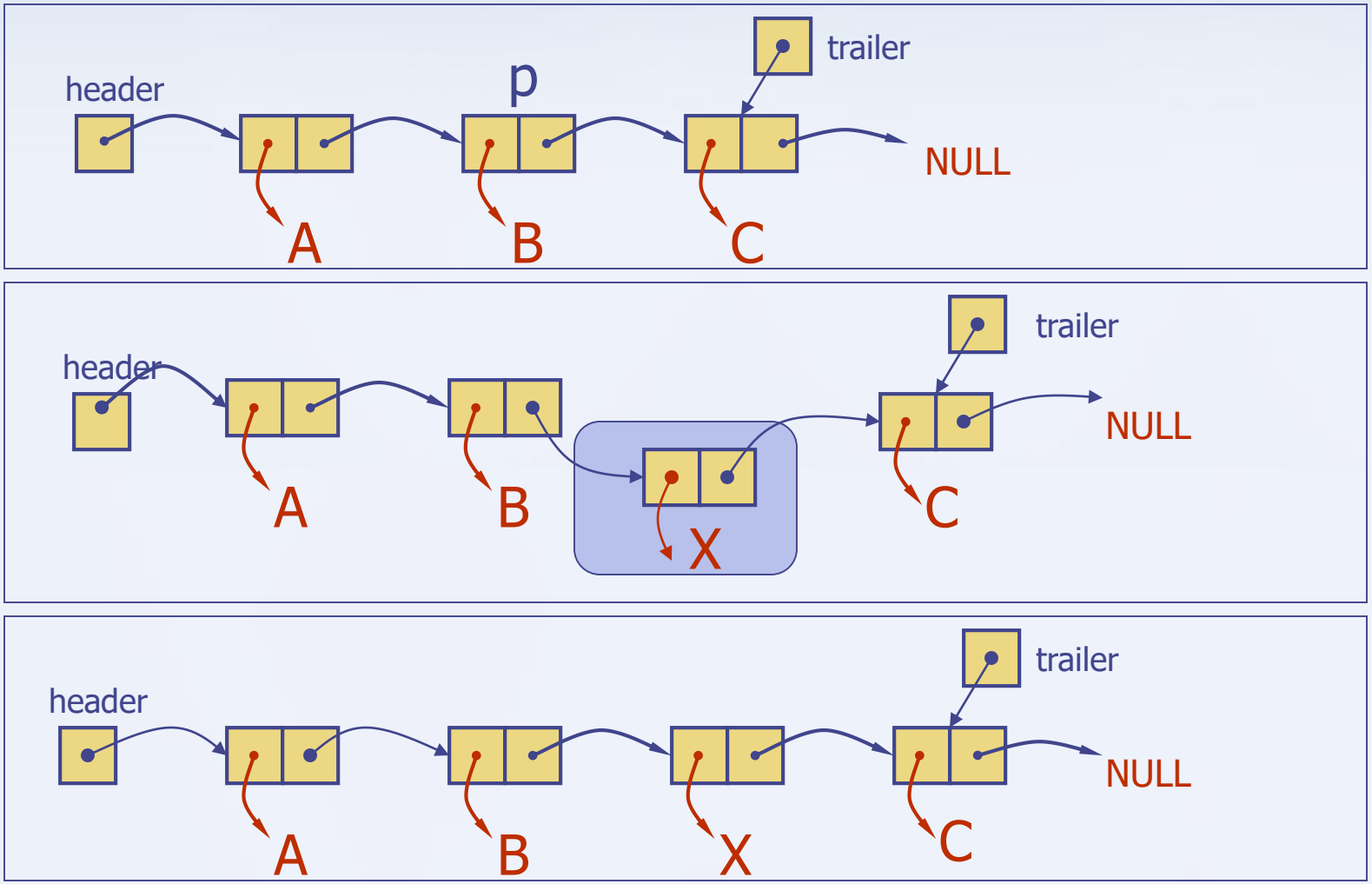
Thêm vào cuối

- Hình ảnh phép toán push_back(), phép toán trả lại vị trí q



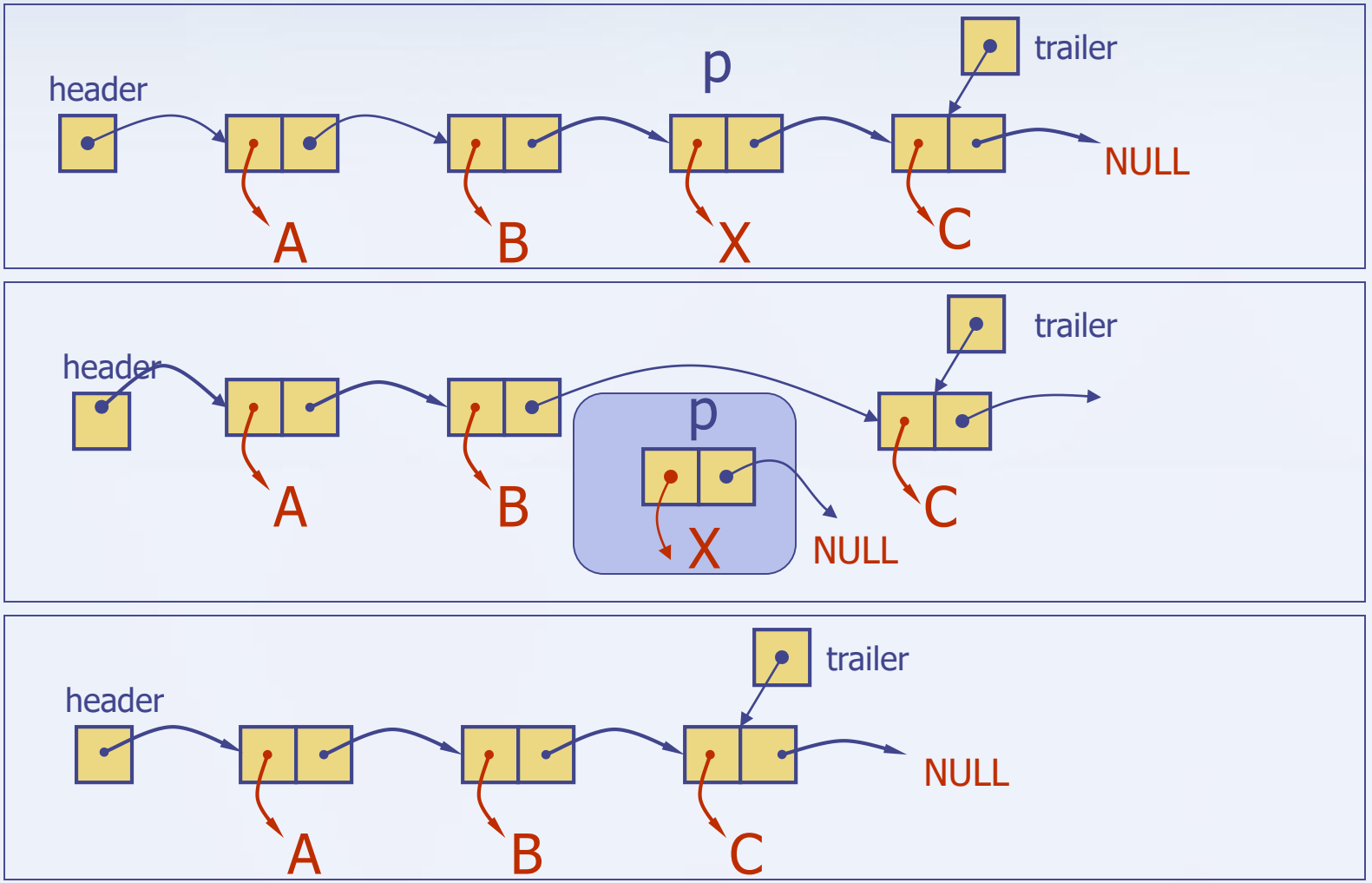
Chèn

- Hình ảnh phép toán insert(p, X), phép toán trả lại vị trí q



Xóa

- Hình ảnh phép toán erase(p)



Code C++

Danh sách liên kết đơn (Singer linker list)

- node.cpp: https://ideone.com/Udb9nc
- slist.cpp: https://ideone.com/3ajMD5
- slist_iterator.cpp: https://ideone.com/Cfied0
- Test: https://ideone.com/dgdlxb

Bài tập

Xây dựng lớp ứng dụng sử dụng lớp Danh sách liên kết đơn để lưu trữ 1 danh sách sinh viên. Mỗi sinh viên gồm các thông tin sau: MaSv, Hoten, Ngay, Thang, Nam sinh, gioi tinh, que quan.

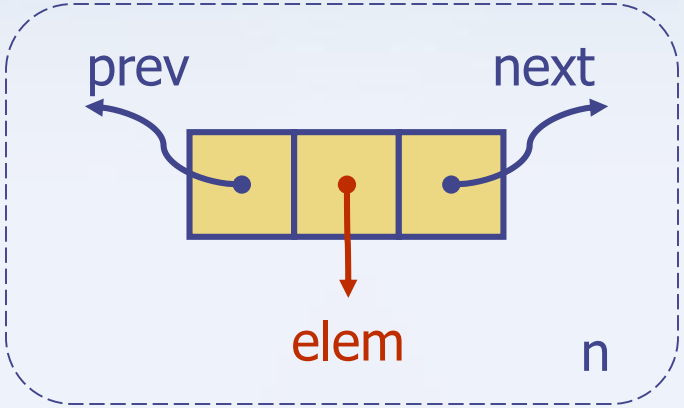
Yêu cầu lớp có các chức năng sau:

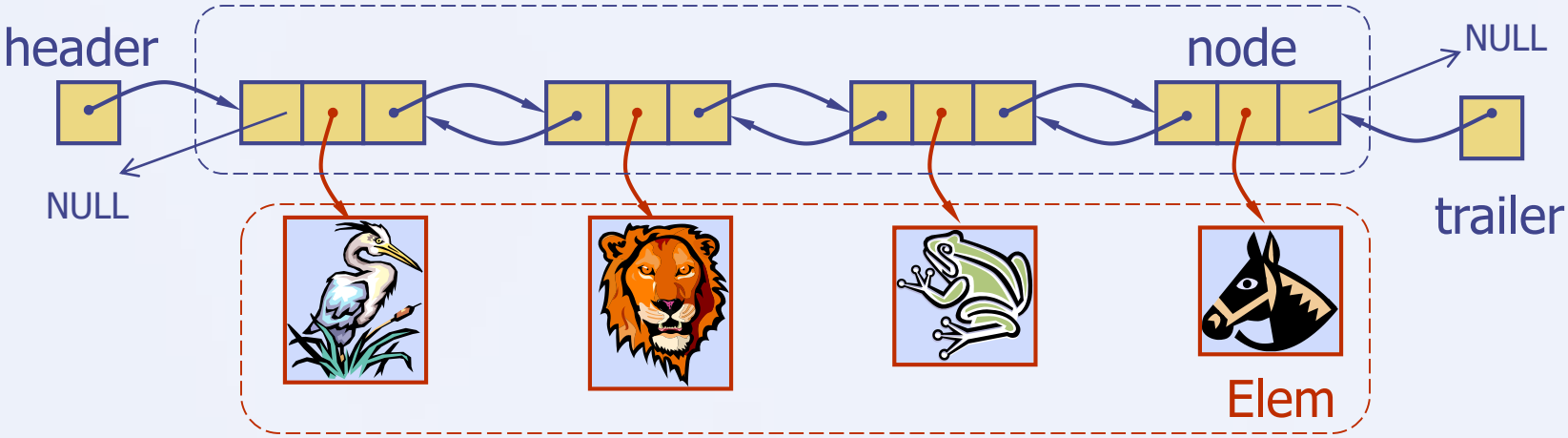
- Thêm một sinh viên vào cuối DS
- Thêm một sinh viên vào đầu DS
- Xóa bỏ một sinh viên thứ i khỏi DS
- Thay thế sinh viên thứ i bằng một sinh viên mới

Xây dựng chương trình để chạy lớp ứng dụng

Danh sách liên kết kép

- Các nút (node) được cài đặt bao gồm:
  - Phần tử lưu trữ trong nó
  - Một liên kết đến nút trước nó
  - Một liên kết đến nút kế tiếp
- Có hai nút đặc biệt là trailer và header





Cấu trúc của một Node

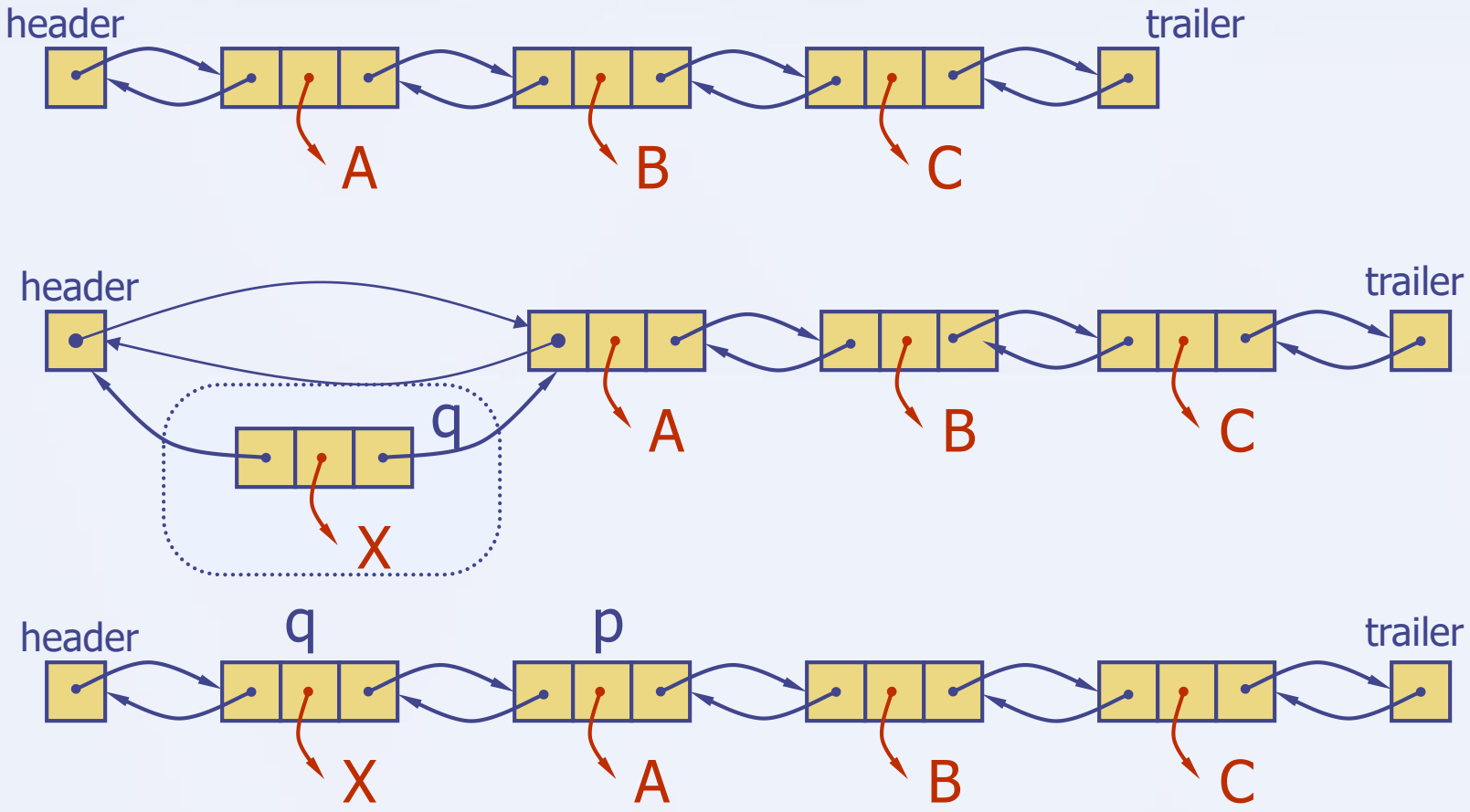
- Các thuộc tính
  - Element elem;
  - Node *next, *pre;
- Các phương thức
  - Node *getnext() - Trả lại địa chỉ của nút kế tiếp
  - Node *getPre() - Trả lại địa chỉ của nút trước đó
  - Element getElem() - Trả lại địa chỉ của phần tử lưu trong nút
  - void setNext(Node *) - Đặt thuộc tính next trỏ đến đ/c của phần tử là đối của phương thức
  - void setPre(Node *) - Đặt thuộc tính pre trỏ đến đ/c của phần tử là đối của phương thức
  - void setElem(Element e) - Đặt phần tử e vào nút

Cấu trúc danh sách liên kết kép

- **Các thuộc tính:**
  - **Node *header**
  - **Node *trailer**
- **Các phương thức chung:**
  - int size(),
  - bool empty()
- **Các phương thức truy cập:**
  - front()
  - back()
- **Chèn và xóa**
  - insert
  - erase
- **Các phương thức cập nhật:**
  - void push_front(T e)
  - void push_back(T e)
  - void pop_front()
  - void pop_back()
- **Bộ lặp xuôi:**
  - begin
  - end
  - =
  - !=
  - ++
  - *
- **Bộ lặp ngược:**
  - rbegin
  - rend
  - =
  - !=
  - ++
  - *

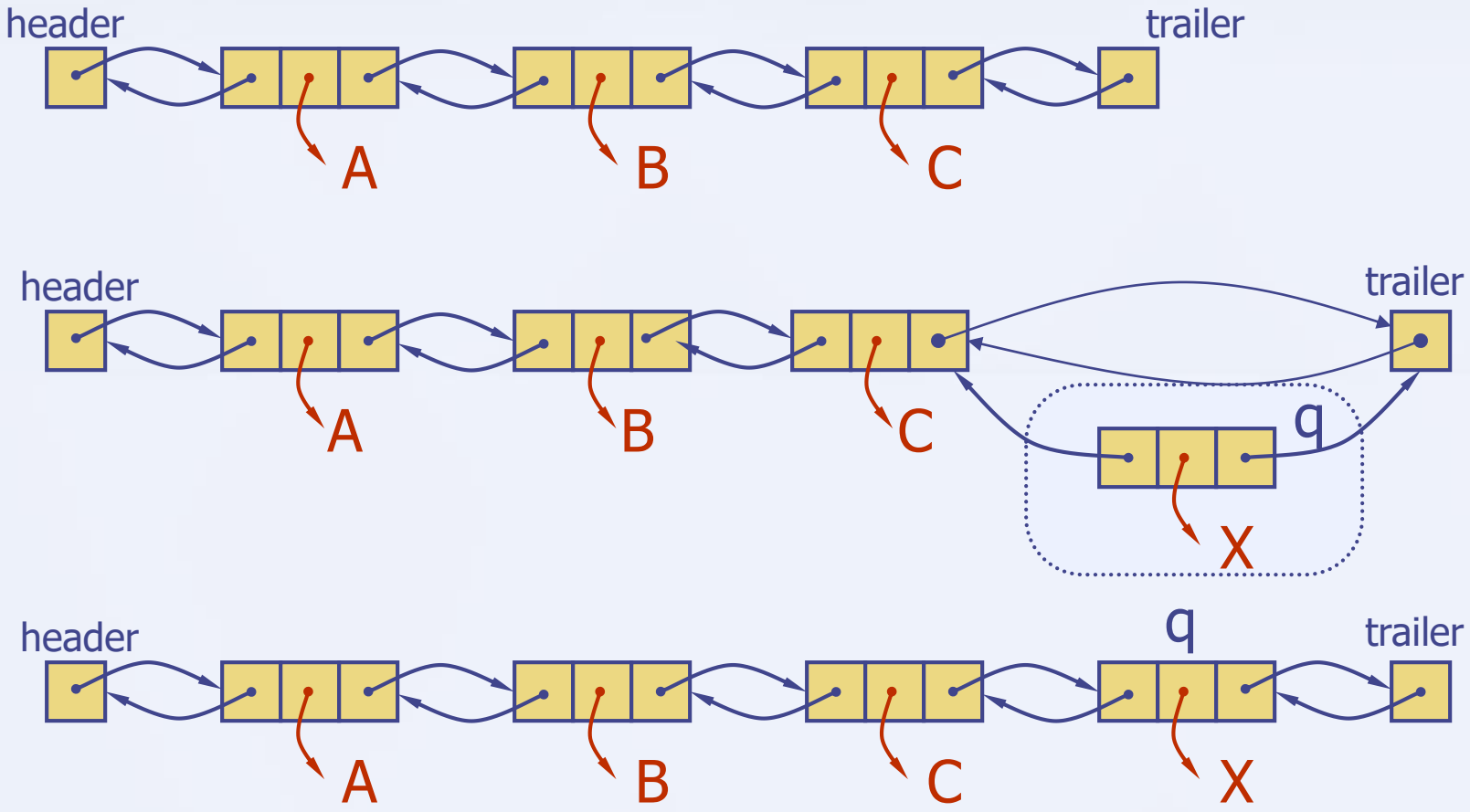
Thêm vào đầu

- Hình ảnh phép toán push_front(X), phép toán trả lại vị trí q



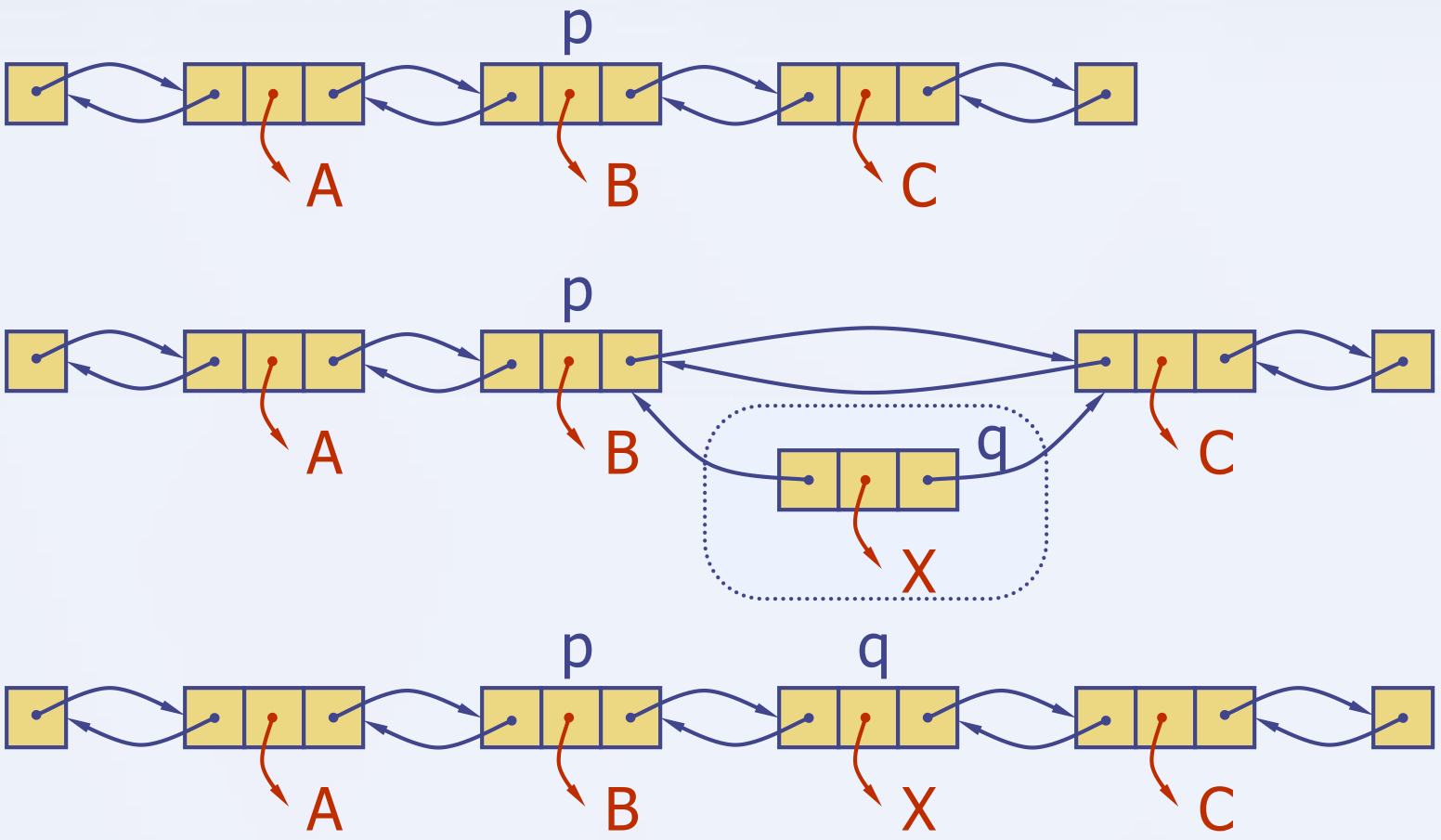
Thêm vào cuối

- Hình ảnh phép toán push_back(X), phép toán trả lại vị trí q



Chèn

- Hình ảnh phép toán Chèn(p, X), phép toán trả lại vị trí q



Thuật toán Chèn

Algorithm insert (*p,e*): //Bổ sung phần tử e vào phần tử nút p

Tạo ra một nút mới q

q->setElement(*e*) //Đặt gia trị e vào nút q

q->setNext(p->getNext())//liên kết với phần tử sau nó

p->getNext()->setPrev(q)//Liên kết phần tử sau p với q

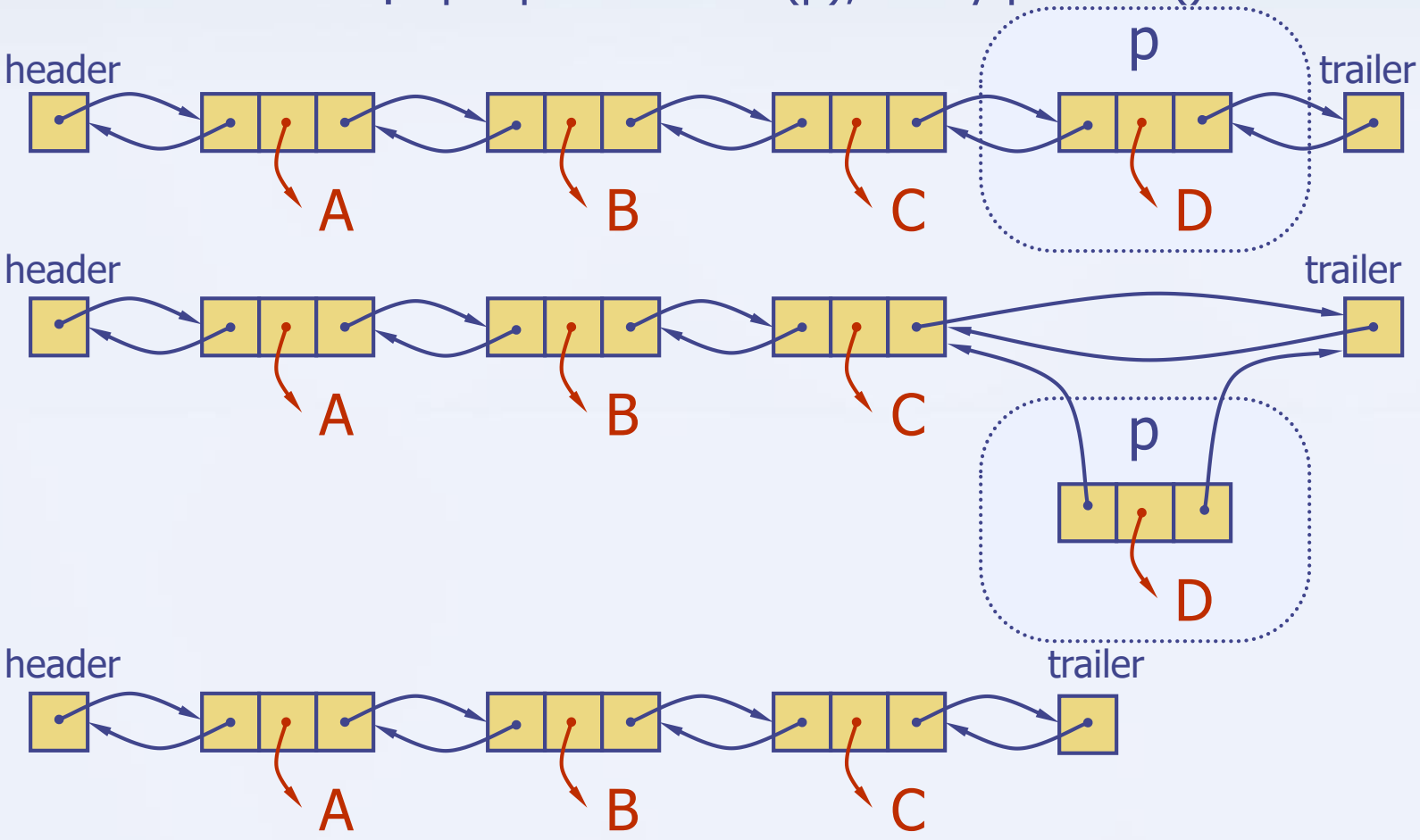
q->setPrev(p) //liên kết q với phần tử trước nó

p->setNext(q) //liên kết p với q

return q //trả lại vị trí của q

Xóa

- Hình ảnh minh họa phép toán erase(p), ở đây p = last()



Thuật toán erase

Algorithm erase(p):

```
//kết nối phần tử trước p với phần tử sau p
    p->getPre()->setNext(p->getNext())
//kết nối phần tử sau p với pần tử trước p
    p->getNext()->setPre(p->getPre())
//bỏ kết nối p với phần tử trước nó
    p->setPre(NULL)
    p->setNext(NULL)
    delete p
```

So sánh mảng và danh sách liên kết

Những đặc trưng của mảng

- Bộ nhớ sử dụng lưu trữ phụ thuộc vào việc cài đặt chứ không phải số lượng thực sự cần lưu.
- Mối quan hệ giữa phần tử đầu và các phần tử khác là rất ít
- Các phần tử được sắp xếp cho phép tìm kiếm rất nhanh
- Việc chèn và xóa phần tử đòi hỏi phải di chuyển các phần tử.

Những đặc trưng của danh sách liên kết

- Bộ nhớ sử dụng để lưu trữ tương ứng với số lượng các phần tử thực sự cần lưu tai bất kỳ thời điểm nào.
- Sử dụng một con trỏ để lưu phần tử đầu, từ đó đi đến các phần tử khác.
- Việc bổ sung và xóa bỏ các phần tử không phải di chuyển các phần tử
- Truy nhập đến các phần tử chỉ có thể thực hiện được bằng cách đi dọc theo chuỗi mắt xích từ phần tử đầu. Vì vậy đối với danh sách liên kết đơn thì thời gian tìm kiếm một phần tử sẽ là $O(n)$.

Danh sách liên kết kép (double linker list)

- node.cpp: https://ideone.com/aW3E0l
- dlist.cpp: https://ideone.com/FxJfsO
- dlist_iterator.cpp: https://ideone.com/4Gmbrr
- Test factorial : https://ideone.com/iDpnlP

Danh sách trong thư viện C++STL

```
#include <iostream>
#include <list>
using namespace std;
int main ()
{
  // constructors used in the same order as described above:
  list<int> first;                                // empty list of ints
  list<int> second (4,100);                       // four ints with value 100
  list<int> third (second.begin(),second.end());  // iterating through second
  list<int> fourth (third);                       // a copy of third

  // the iterator constructor can also be used to construct from arrays:
  int myints[] = {16,2,77,29};
  list<int> fifth (myints, myints + sizeof(myints) / sizeof(int) );
  cout << "The contents of fifth are: ";
  for (list<int>::iterator it = fifth.begin(); it != fifth.end(); it++)
    cout << *it << ' ';
  return 0;
}
```

Danh sách trong thư viện C++STL

- Xem phần tử
  - front
  - back
- Bộ lặp xuôi
  - begin
  - end

```
#include <iostream>
#include <list>
using namespace std;
int main ()
{
  list<double> L;           //tao danh sach rong
  for(double x:{3.5,6.7,1.2,4.8,0.5,2.3}) L.push_back(x); //them cac phan tu vao cuoi
  cout<<"\nCac phan tu trong L : ";
  for(auto x: L ) cout<<x<<" ";        //vong for C++11 tro di
  cout<<"\nPhan tu dau "<<L.front();
  cout<<"\nPhan tu cuoi "<<L.back();
  L.front()=9.0; L.back()=2.1;  //gan lai 2 phan tu dau va cuoi
  cout<<"\nCac phan tu trong L : ";
  for(auto x: L ) cout<<x<<" ";        //vong for C++11 tro di
}
```

Danh sách trong thư viện C++STL

```
#include <iostream>
#include <list>
using namespace std;
int main ()
{
  list<int> L(5,1);           //tao danh sach 5 so 1
  //Them x trong day sau neu chan vao cuoi, le vao dau list L
  for(int x:{3,7,2,4,5,6,8})   //vong for cho C++11 tro len
  x%2==0?L.push_back(x):L.push_front(x);
  list<int> ::iterator it;    //Khai bao bo lap xuoi
  cout<<"\nCac phan tu trong L : ";
  for(it=L.begin();it!=L.end();it++) cout<<*it<<" ";
  L.pop_back();  L.pop_back();  //xoa 2 phan tu cuoi
  L.pop_front();                //xoa 1 phan tu dau
  cout<<"\nCac phan tu trong L duyet nguoc tu cuoi ve dau : ";
  for(list<int>::reverse_iterator rit=L.rbegin();rit!=L.rend();rit++) cout<<*rit<<" ";
  it=L.begin(); it++;           //Bo lap tro vao phan tu thu 1 va chuyen sang pt thu 2
  L.erase(it);                  //Xoa phan tu khoi vi tri bo lap it
  it=L.begin(); it++; it++;     //Bo lap tro vao phan tu thu 3
  L.insert(it,9);               //Chen so 9 vao  vi tri bo lap
  cout<<"\nCac phan tu cua L : ";
  for(auto x:L) cout<<x<<" ";
}
```

Bài tập

- Xây dựng lớp Node
- Xây dựng lớp DblList
- Xây dựng lớp DblItr //Lớp bộ lặp
- Xây dựng lớp ứng dụng sử dụng lớp Danh sách liên kết kép để lưu trữ 1 danh sách sinh viên. Mỗi sinh viên gồm các thông tin sau: MaSv, Hoten, Ngay, Thang, Nam sinh, gioi tinh, que quan.

Yêu cầu lớp có các các chức năng sau:

- Thêm một sinh viên vào cuối DS
- Thêm một sinh viên vào đầu DS
- Xóa bỏ sinh viên thứ i khỏi DS
- Thay thế sinh viên thứ i bằng một sinh viên mới

Xây dựng chương trình để chạy lớp ứng dụng

Hết