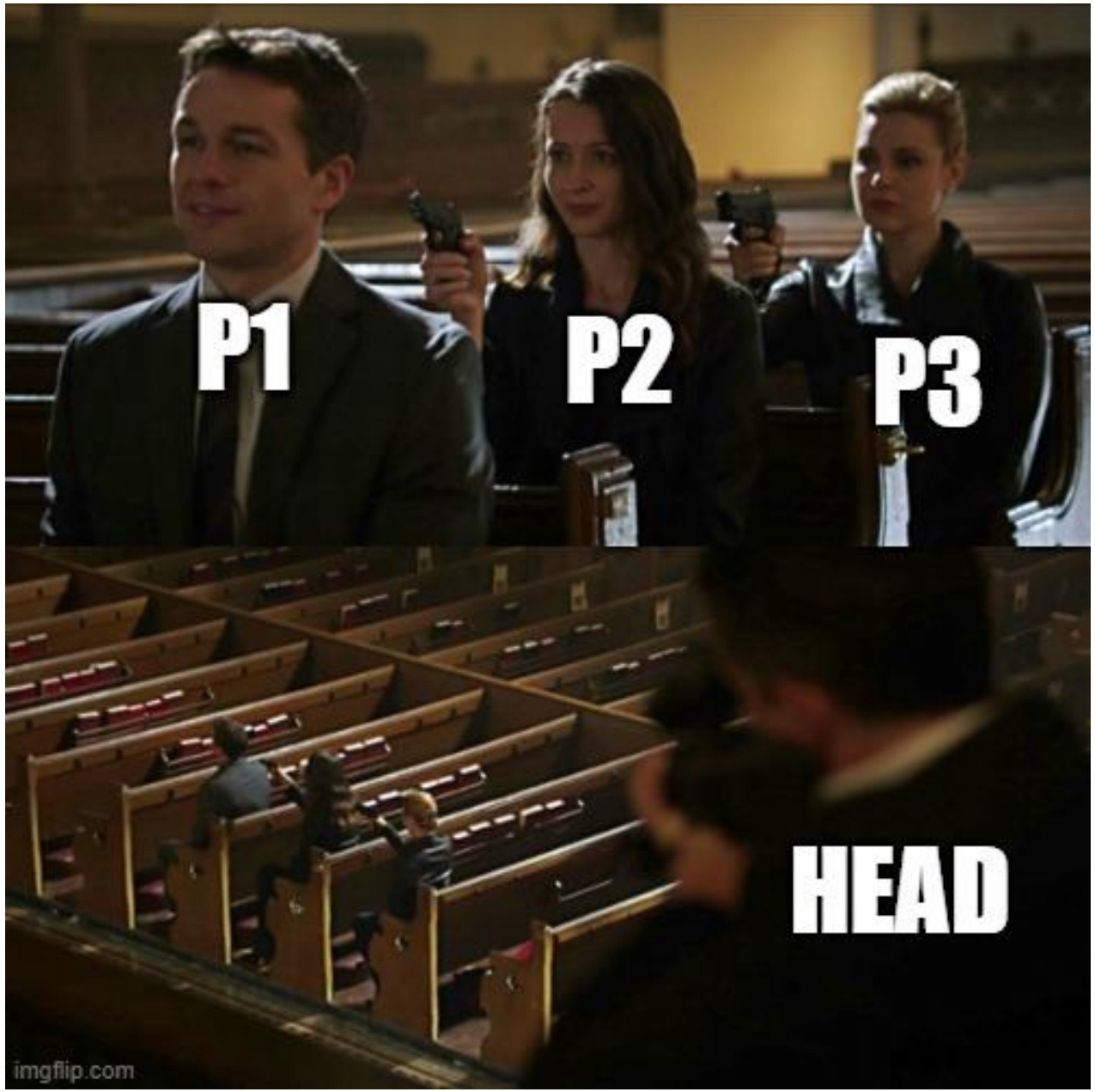
OOP & data struct

11. Linked list

BY SOMSIN THONGKRAIRAT


P1
P2
P3
HEAD
imgflip.com

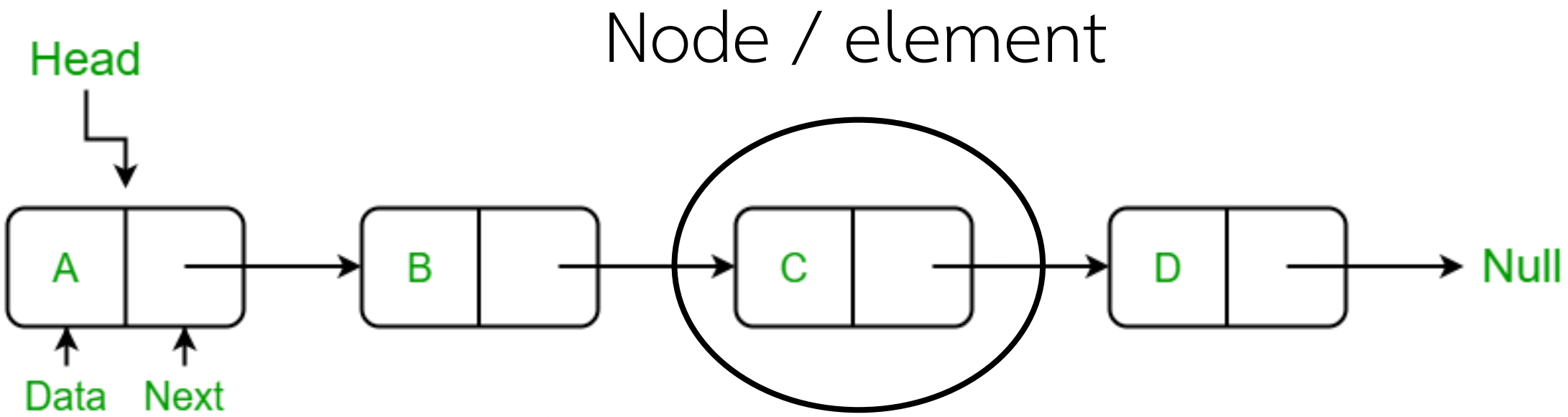
Linked list

- basic dynamic size structure
- easy to insert and delete member in structure
- use to build further advance data structure
- linear structure

Linked list

- เป็น structure พื้นฐานที่สามารถเปลี่ยนแปลงขนาดได้
- ง่ายต่อการ insert และ delete สมาชิก
- ใช้ในการสร้าง data structure ขั้นสูงต่อไป
- linear structure

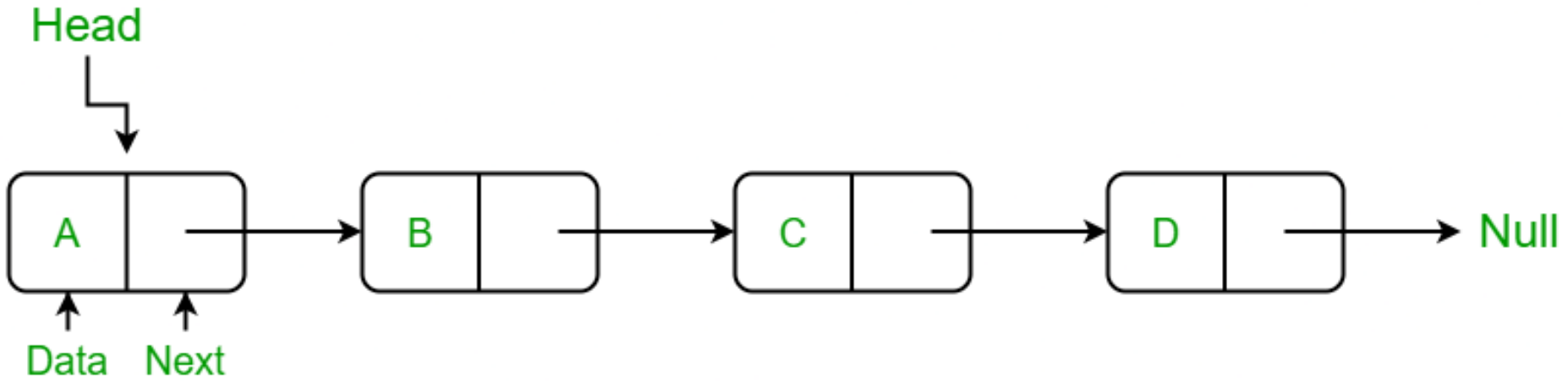
Concept



https://www.geeksforgeeks.org/what-is-linked-list//

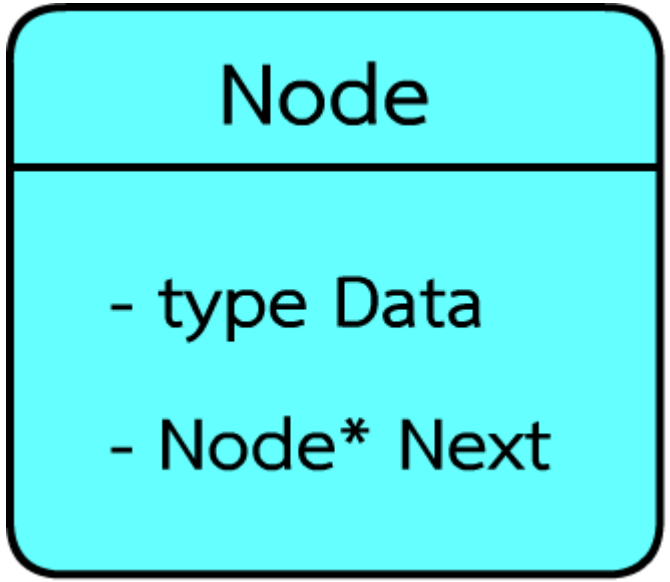
Linked list

- each node(object) contain data and representation to single next node
- แต่ละ node(object) จะเก็บ data และตัวแทนสำหรับข้อมูลถัดไป



object

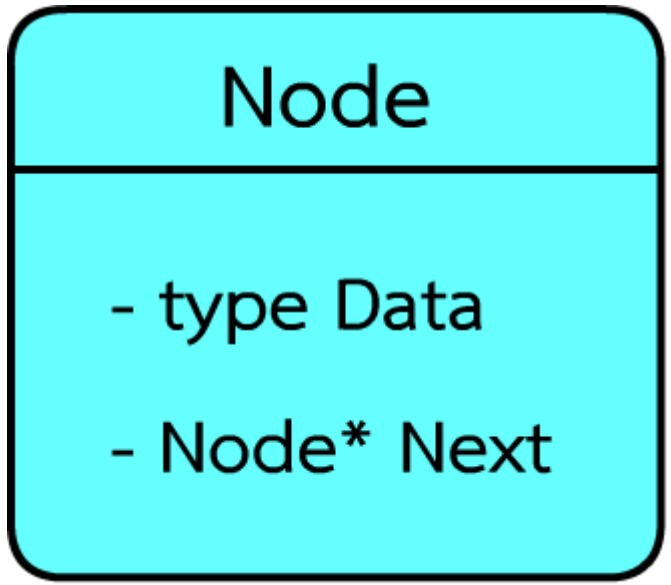
Each node(object) contain / แต่ละ node(object) จะประกอบไปด้วย



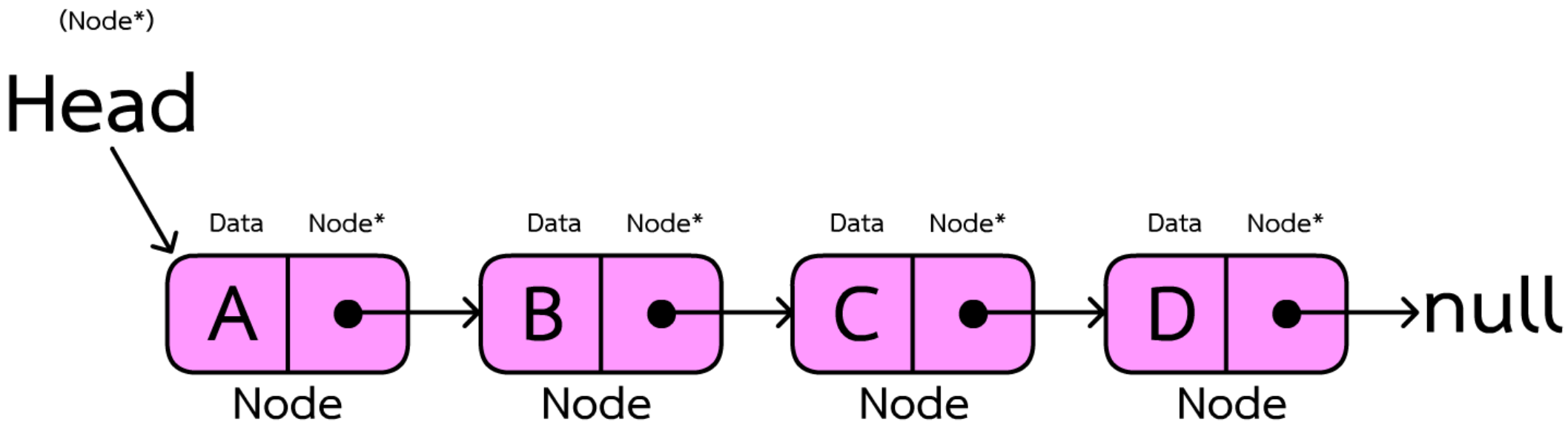
- Data
- Representer to next member
- ข้อมูล
- ตัวแทนไปยังข้อมูลถัดไป

Object (int Linked list)

```
class Node{
public :
     char data;
     Node* next;
};
```



Concept


(Node*)
Head
Data
Node*
A
Node
Data
Node*
B
Node
Data
Node*
C
Node
Data
Node*
D
Node
null

To create Link list (C++)

- start with empty structure / เริ่มจาก structure ที่ว่างป่าว

- use pointer to access first member of list called “Head”

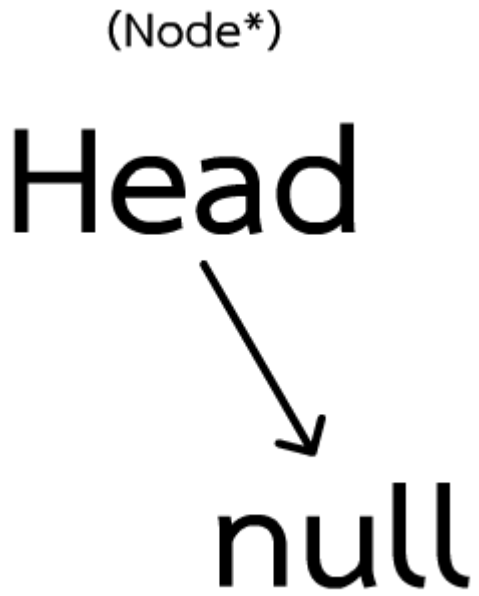
- ใช้ pointer ในการเข้าถึงตัวแรงของ list เรียกว่า “Head”

- at start Head point to null or empty

- ตอนเริ่มต้น Head ชี้ไปที่ null หรือ ความว่างป่าว

To create Link list (C++)

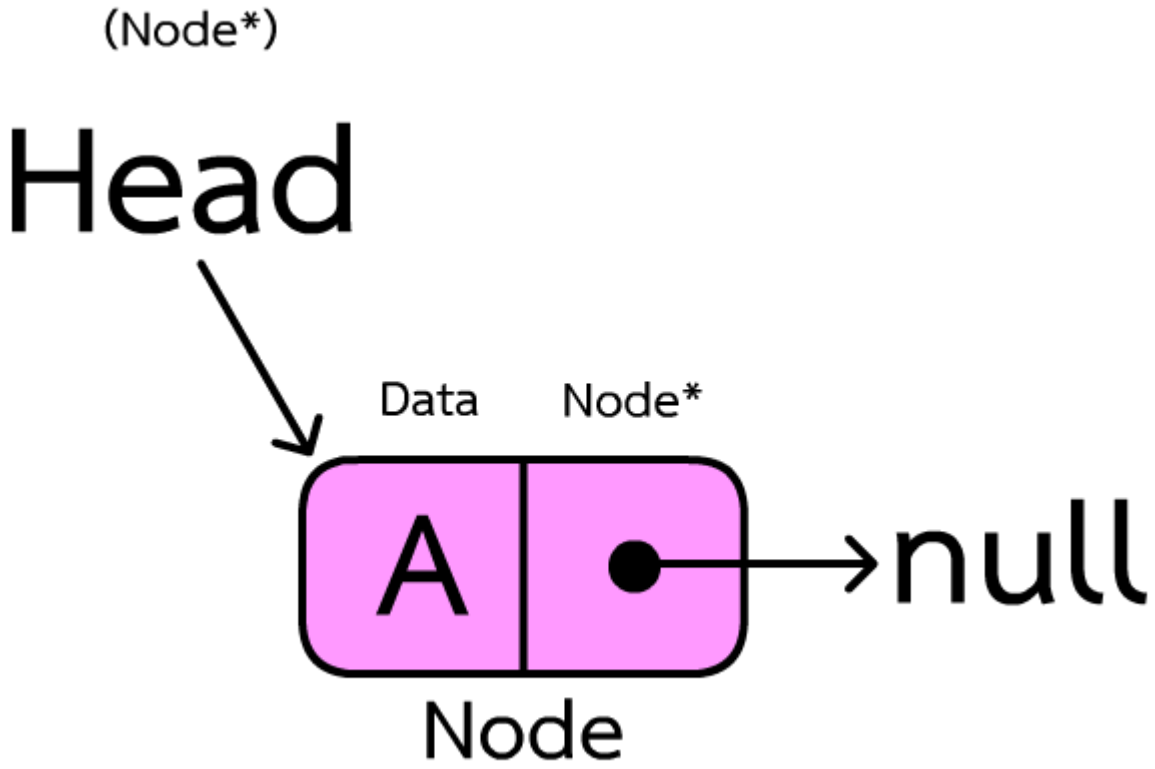
```
int main(){

    Node* head = nullptr;
    return 0;
}
```



* Using main just for convenient for explaining / ขออธิบายใน main ไปก่อนเพื่อความสะดวกนะครับ ^^

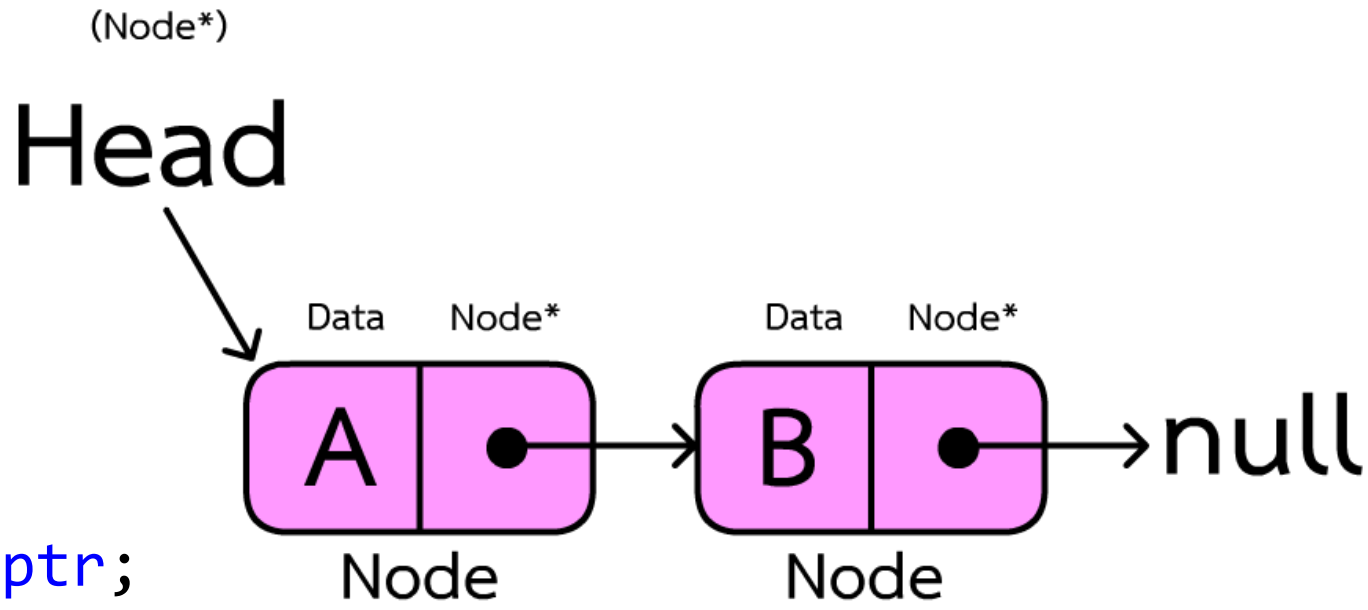
Create first element

```
int main(){

    Node* head = nullptr;

    head = new Node();
    head->data = 'A';
    head->next = nullptr;

    return 0;
}
```



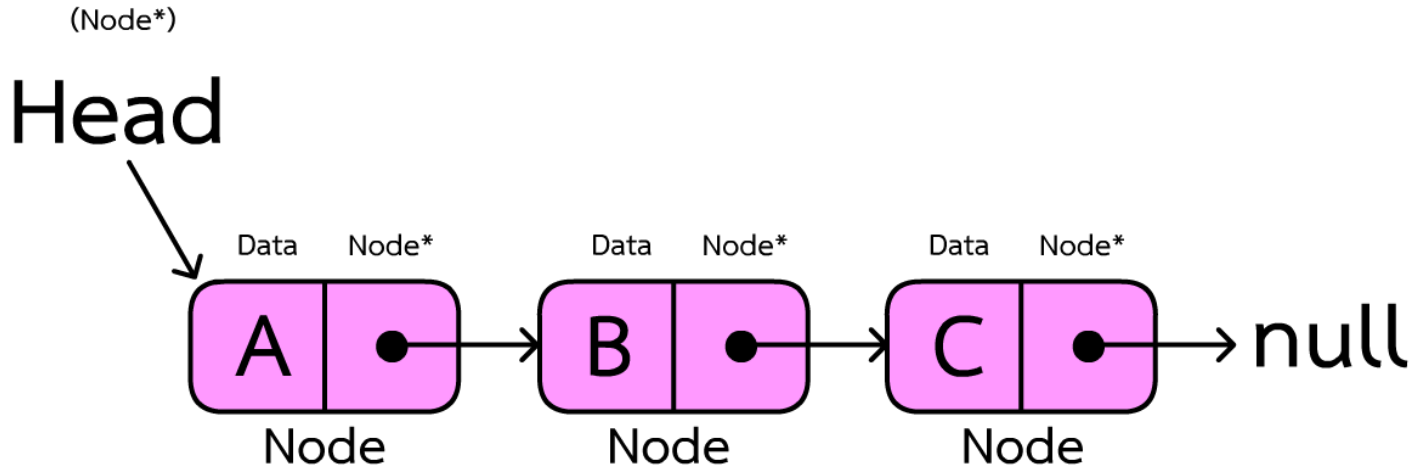
Create second element

```
head = new Node();
head->data = 'A';
head->next = nullptr;

head->next = new Node();
(head->next)->data = 'B';
(head->next)->next = nullptr;
```

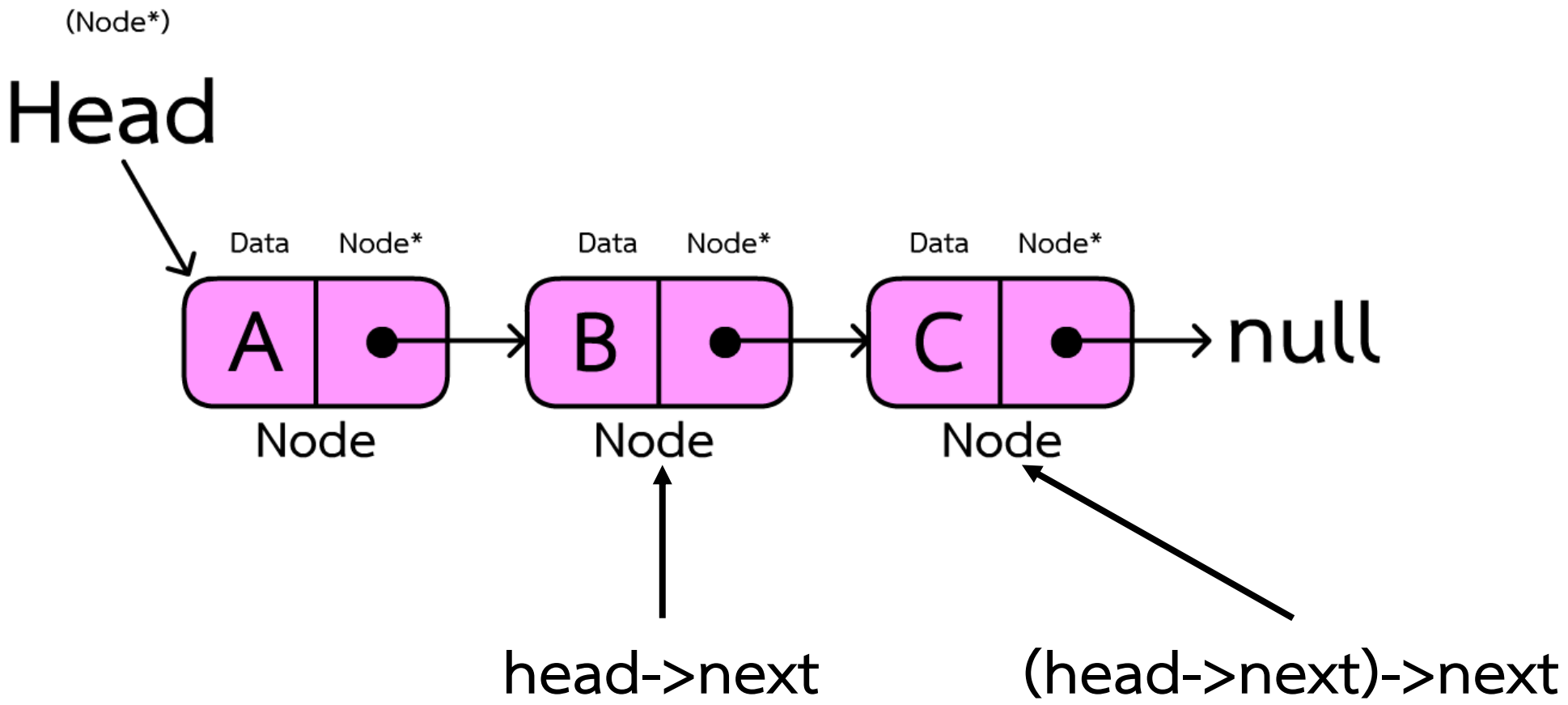


Create third element

```
head = new Node();
head->data = 'A';
head->next = nullptr;

(head->next) = new Node();
(head->next)->data = 'B’;
(head->next)->next = nullptr;

(head->next)->next = new Node();
((head->next)->next)->data = 'C';
((head->next)->next)->next = nullptr;
```



Access to items



Access to items

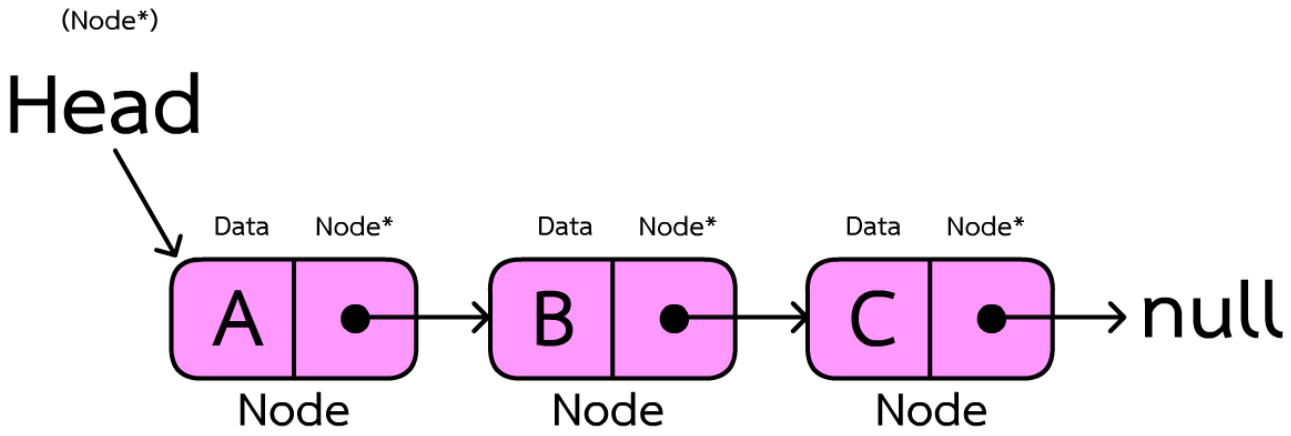
```
cout << head->data << endl;
cout << (head->next)->data << endl;
cout << ((head->next)->next)->data << endl;
```

Result:

A

B

C

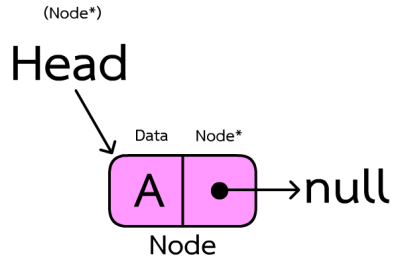


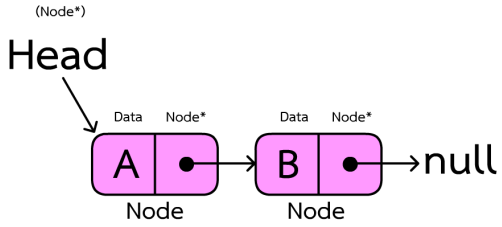
This is Linked list

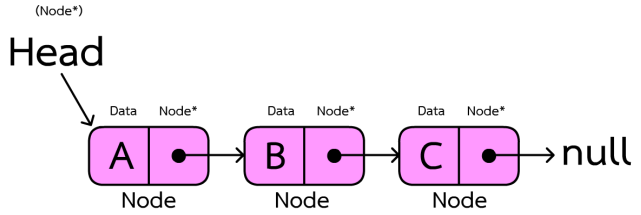
Happy!?

How to make it easy to add ?

อย่างไรให้ add ง่ายขึ้น ?







Make it more general add function

- traverse in list until found **null** then add new node

- ท่อง ไปใน list จนกว่าจะพบ **null** จากนั้นสร้าง node ใหม่

```
void add_node(Node* p,char item){
    if(p == nullptr){ // for empty head list
        p = new Node();
        p->data = item;
        p->next = nullptr;
    }
    else{
        while(p->next != nullptr){
            p = p->next;
        }
        p->next = new Node();
        (p->next)->data = item;
        (p->next)->next = nullptr;
    }
}
```

```
add_node(head,'D');

cout << head->data << endl;
cout << (head->next)->data << endl;
cout << ((head->next)->next)->data << endl;
cout << (((head->next)->next)->next)->data << endl;
```

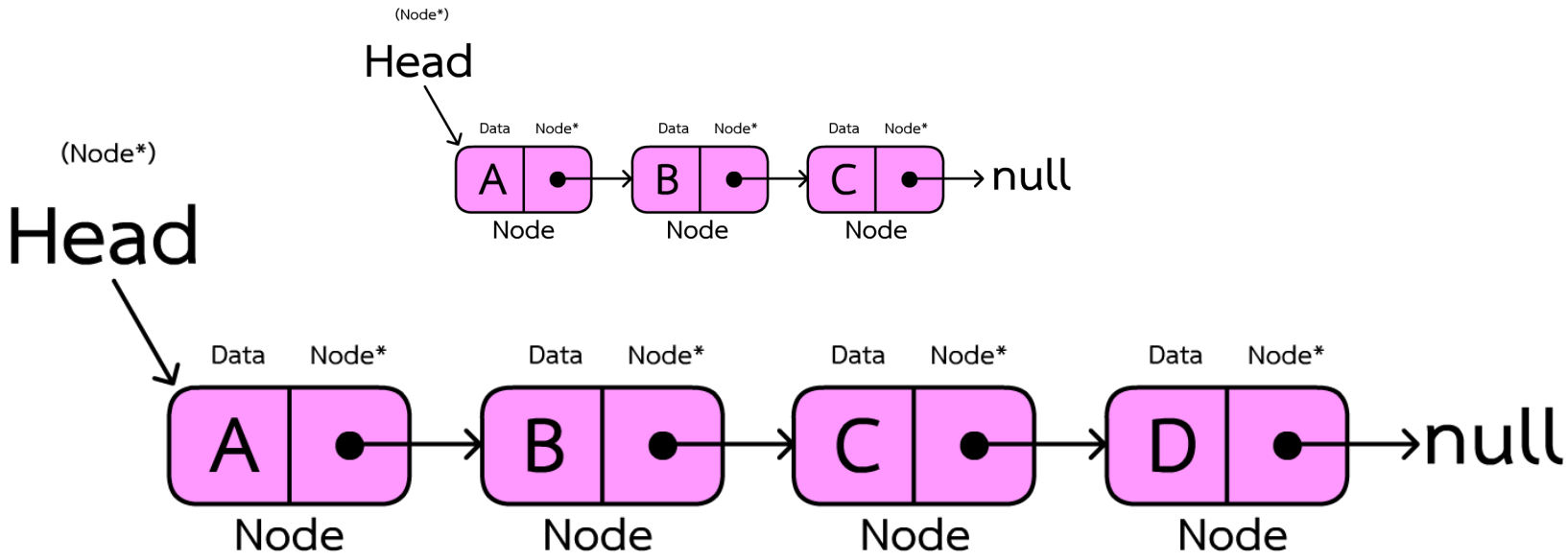
Result :

A

B

C

D

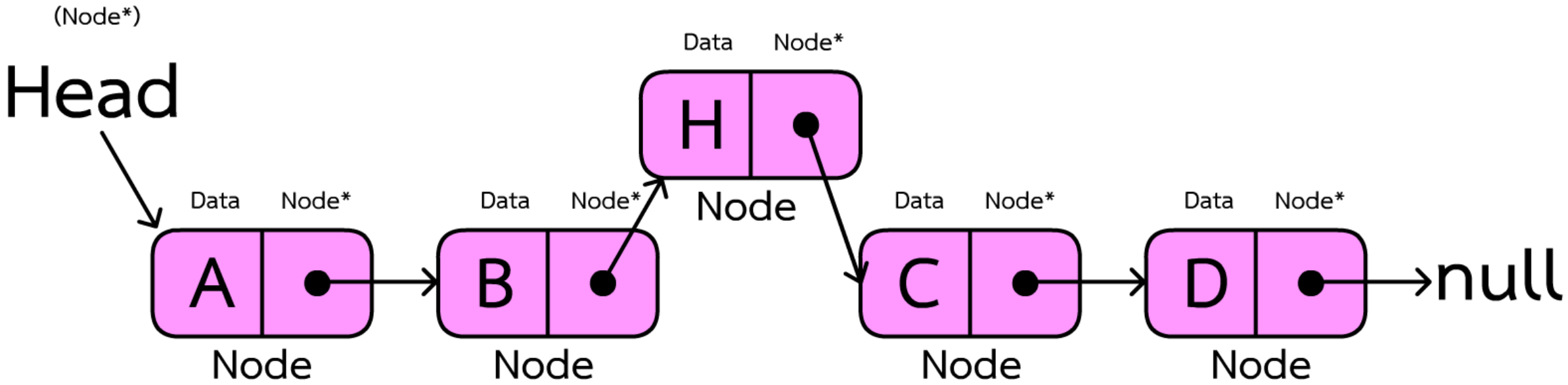


Quiz

Can we add in-between structure ?

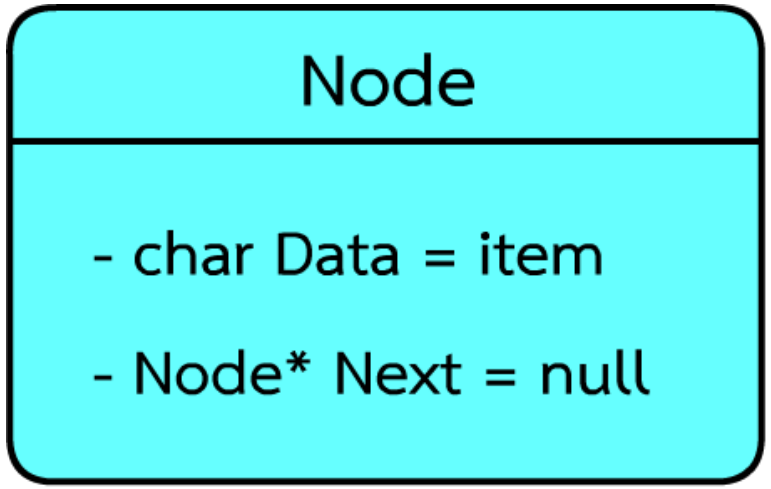
สามารถ add ตรงกรางระหว่าง structure ได้หรือไม่ ?

INSERT AT INDEX


(Node*)
Head
Data
Node*
A
Node
Data
Node*
B
Node
Data
Node*
H
Node
Data
Node*
C
Node
Data
Node*
D
Node
null

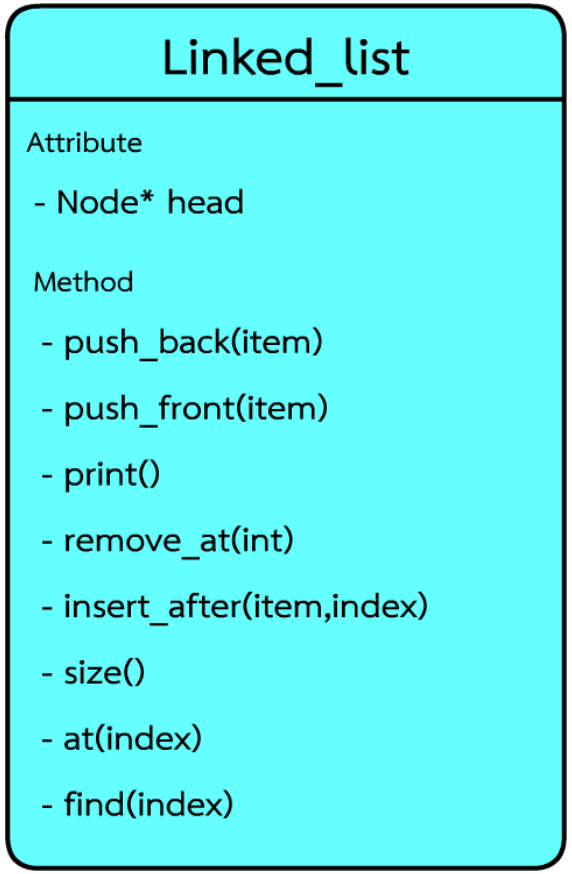
Make it OOP – Node constructor

```
class Node{
public :
    char data;
    Node* next;

    Node(char item){
        data = item;
        next = nullptr;
    }
};
```



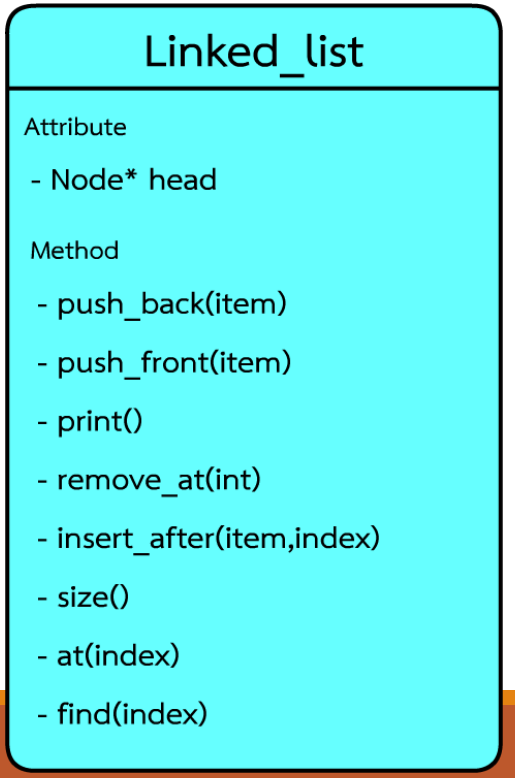
Link list Class

- attribute
  - Node* head
- method
  - push_back(item) / append(item) / insertLast / insertTail
    - Add item to last of list / เพิ่ม item ต่อจากสมาชิกตัวสุดท้าย
  - print() print all member in the list / print สมาชิกทุกตัวใน list
  - push_front(item) / push(item)
    - Add item to beginning of the list / เพิ่ม item เข้ามาเป็นตัวแรกสุดของ list



Link list Class (Continue)

- method
  - remove_at(index) remove member at index / ลบสมาชิกตัวที่ index
  - insert_after(item,index)
    - insert new member at index / แทรก item ตัวใหม่เข้าไปหลังจาก index
  - size() return number of items in list / return ค่าของจำนวนสมาชิกใน list
  - at(index) return member at index / return สมาชิกในตำแหน่งที่ index
  - find(item) find item in list / หา item ใน list



Constructor

- all new head must be null

- head ของ object ที่สร้างใหม่จะเริ่มต้นด้วย null เสมอ

```
class Linked_list{
    Node* head;

    Linked_list(){
        head = nullptr;
    }

};
```

push_back(item) Add item to last of list / เพิ่ม item ต่อจากสมาชิกตัวสุดท้าย

```
void push_back(char item){ // insert to last node
    Node* new_node = new Node(item);

    if(head == nullptr){ // for empty head list
        head = new_node;
    }
    else{
        Node* p = head;
        while(p->next != nullptr){
            p = p->next;
        }
        p->next = new_node;
    }
}
```

* Same as add_node(item)

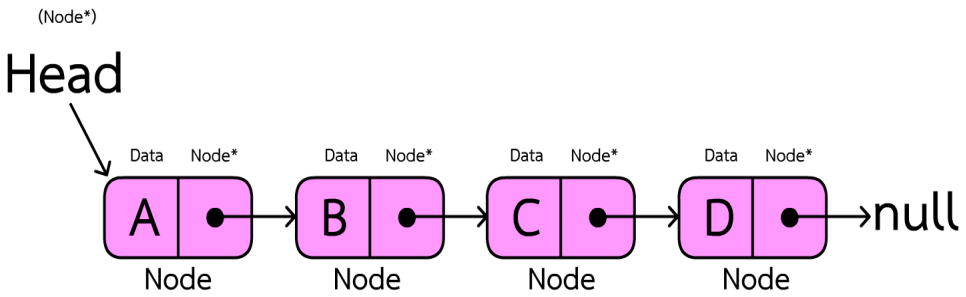
* เหมือนกันกับ add_node(item)

print()

- traverse in list and print until found **null**

- ท่อง ไปใน list และ print จนกว่าจะพบ **null**

```
void print(){
    Node* p = head;
    while(p!=nullptr){
        cout << p->data << "->";
        p=p->next;
    }
    cout << "null" << endl;
}
```

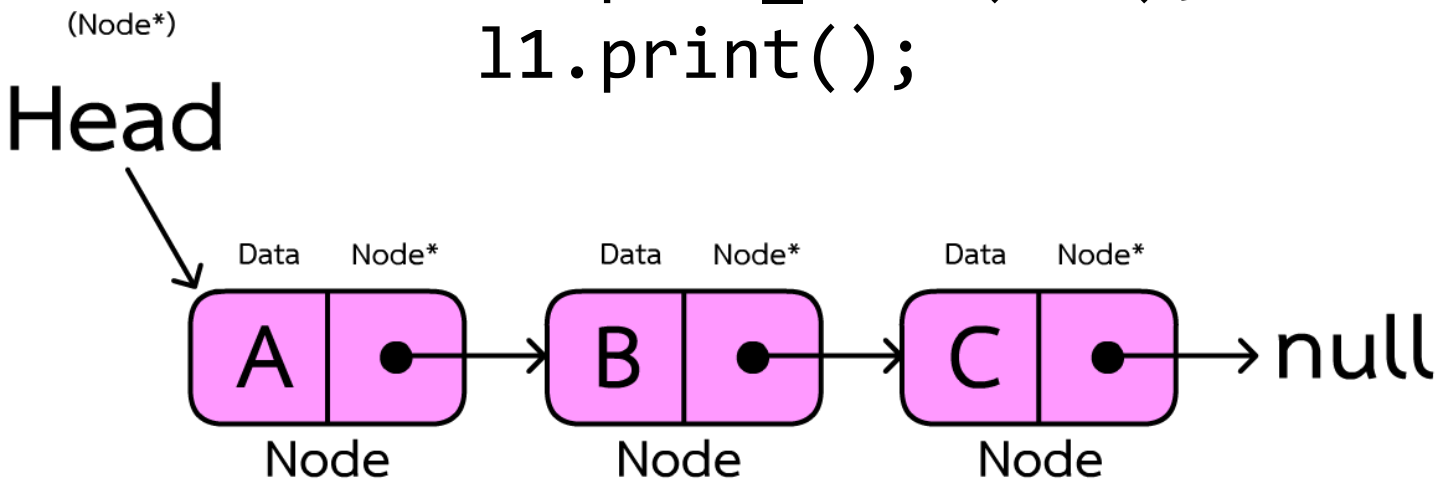


test

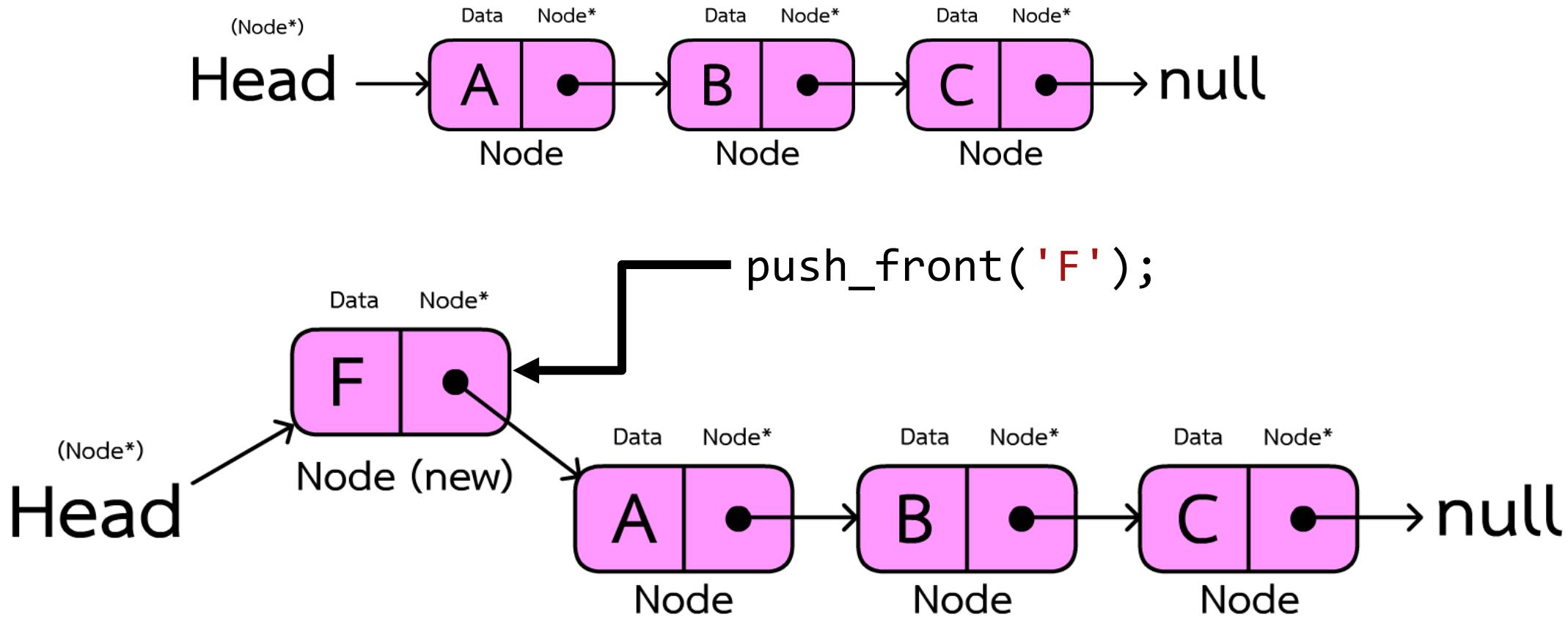
```
Linked_list l1;
l1.push_back('A');
l1.push_back('B');
l1.push_back('C');
l1.print();
```

Result :

A->B->C->null

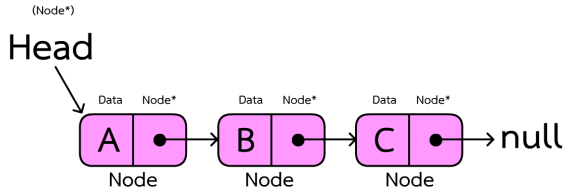


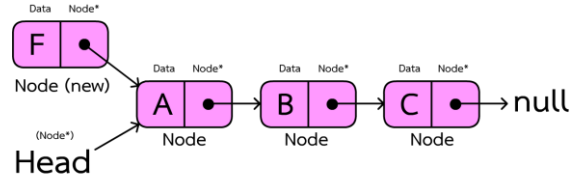
Push_front(item) Add item to beginning of the list / เพิ่ม item เข้ามาเป็นตัวแรกสุดของ list

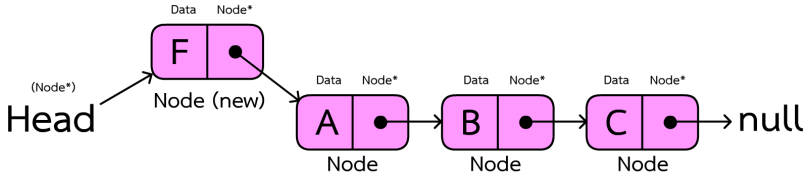


Push_front(item)

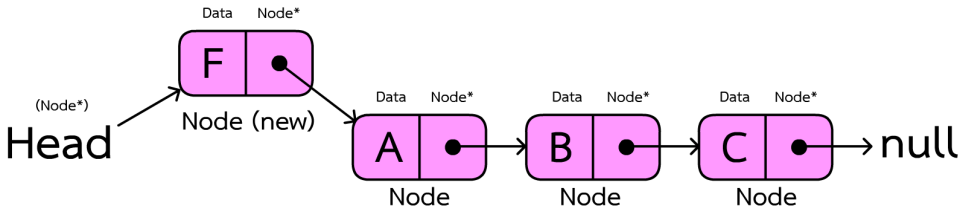
```
void push_front(char item){ // insert to first of list
    Node* new_node = new Node(item);
    if(head == nullptr){ // for empty head list
        head = new_node;
    }
    else{
        new_node->next = head; // step 1
        head = new_node; // step 2
    }
}
```







Check

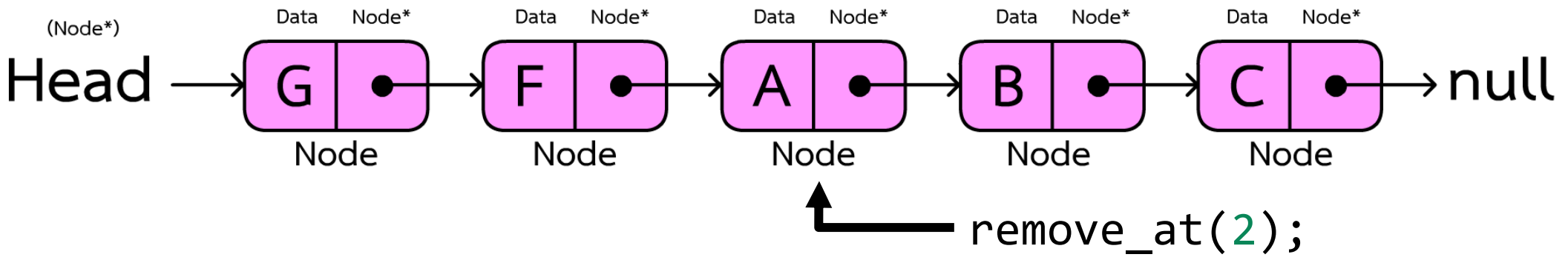


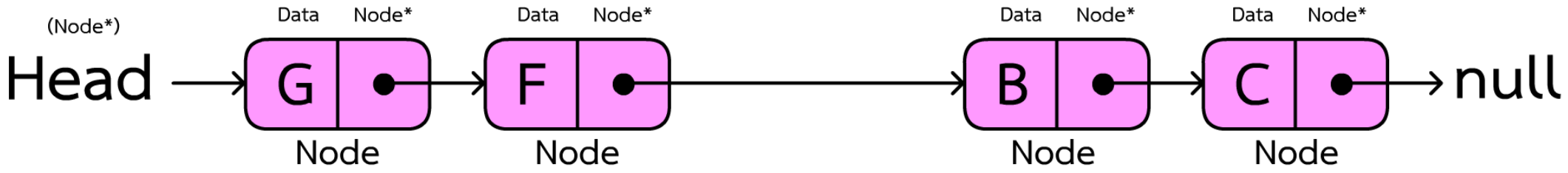

```
Linked_list l1;
l1.push_back('A');
l1.push_back('B');
l1.push_back('C');
l1.print();
l1.push_front('F');
l1.push_front('G');
l1.print();
```

Result:

G->F->A->B->C->null

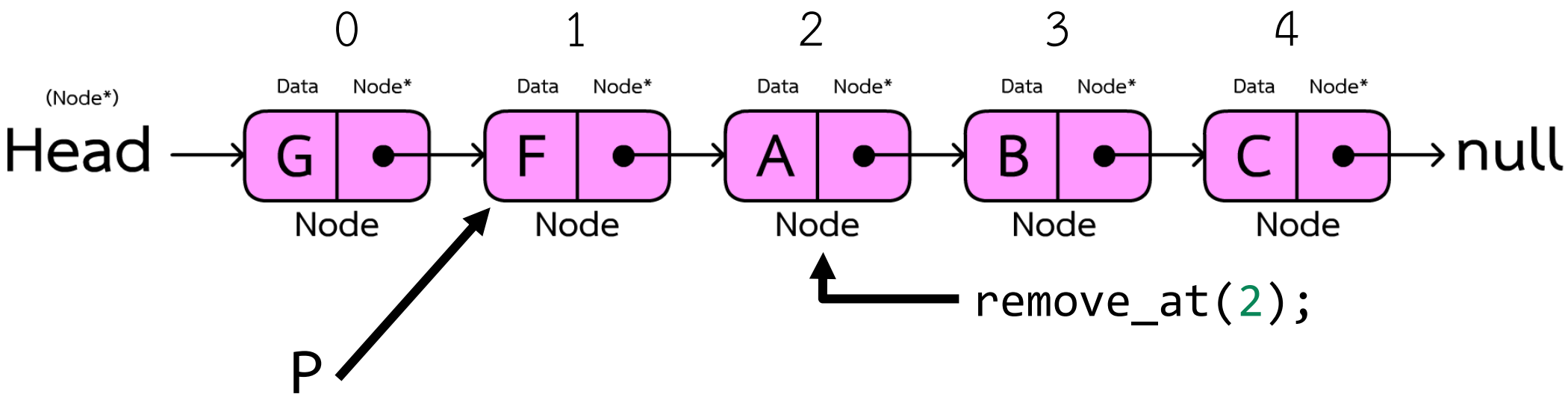
remove_at(index)





remove_at(index) step by step

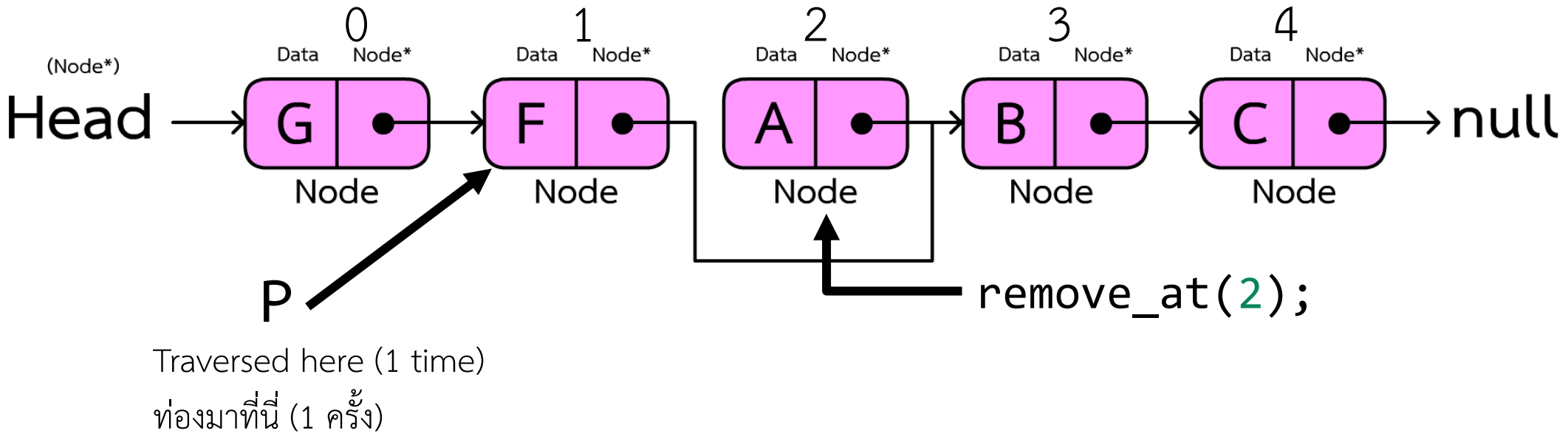
- traverse in list (index – 1) time / ท่อง ไปใน list นจำนวน (index – 1) ครั้ง



remove_at(index) step by step

Done!??

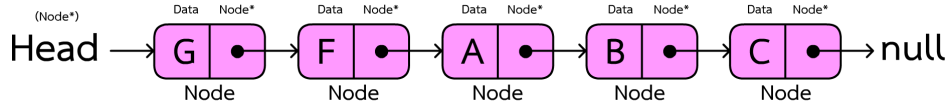
- point next of traversed member to the next of next /
- กำหนดค่าของ next ของตัวที่ ท่อง ไปล่าสุดให้มีค่าเท่ากับ next ของ ตัวถัดไป

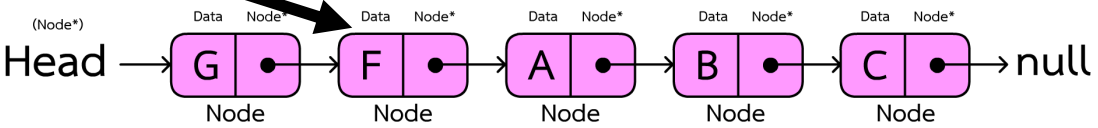


remove_at(index) step by step

- traverse in list (index – 1) time /

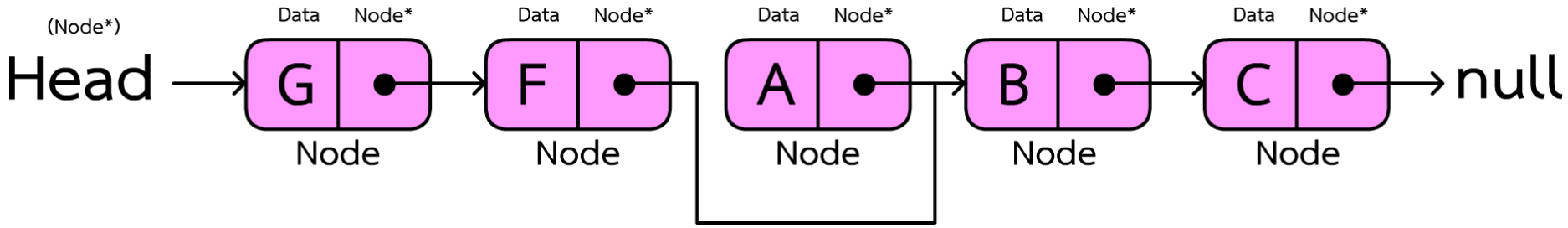
- ท่อง ไปใน list นจำนวน (index – 1) ครั้ง





- point next of traversed member to the next of next /

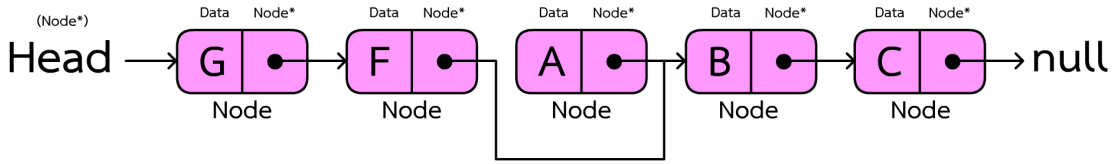
- กำหนดค่าของ next ของตัวที่ ท่อง ไปล่าสุดให้มีค่าเท่ากับ next ของ ตัวถัดไป



remove_at(index) step by step

```
void remove_at(int index){ // insert to first of list
    Node *p = head;
    for(int i=0;i<index-1;i++){ // traverse to index - 1
        p = p->next;
    }

    p->next = (p->next)->next; // step 2
}
```

Check




```
Linked_list l1;
l1.push_back('A');
l1.push_back('B');
l1.push_back('C');
l1.print();
l1.push_front('F');
l1.push_front('G');
l1.print();
l1.remove_at(2);
l1.print();
```

Result:

A->B->C->null

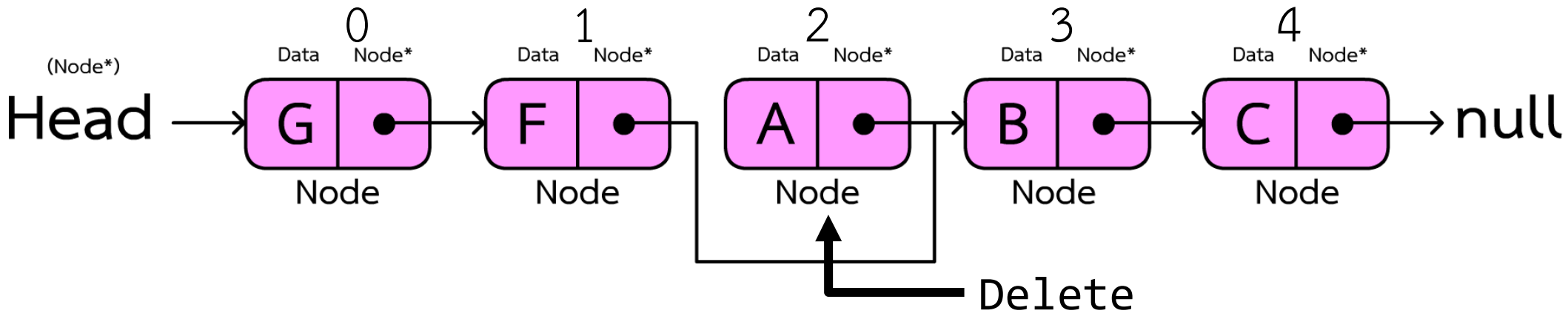
G->F->A->B->C->null

G->F->B->C->null

remove_at(index) step by step

Bad End

- delete to prevent memory leak
- delete เพื่อกันการกิน memory

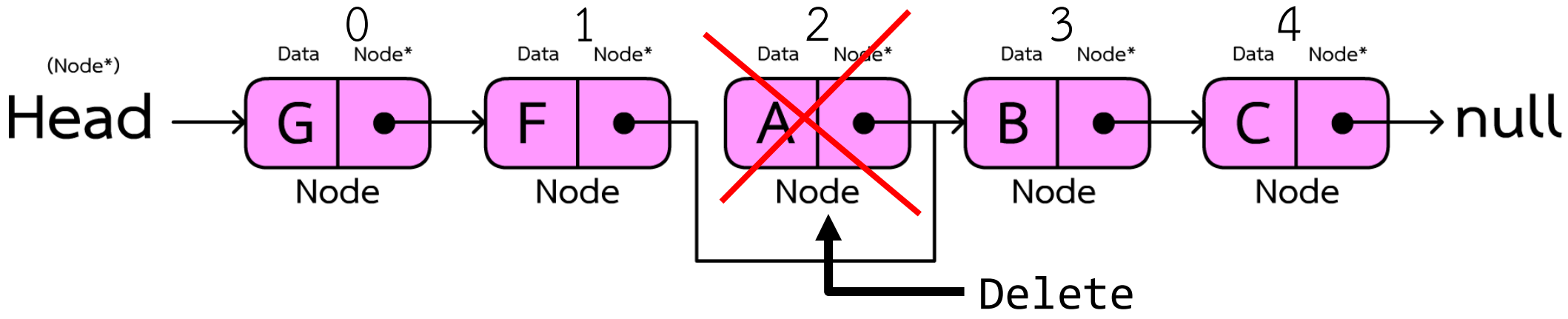


remove_at(index) step by step

Truly Done!

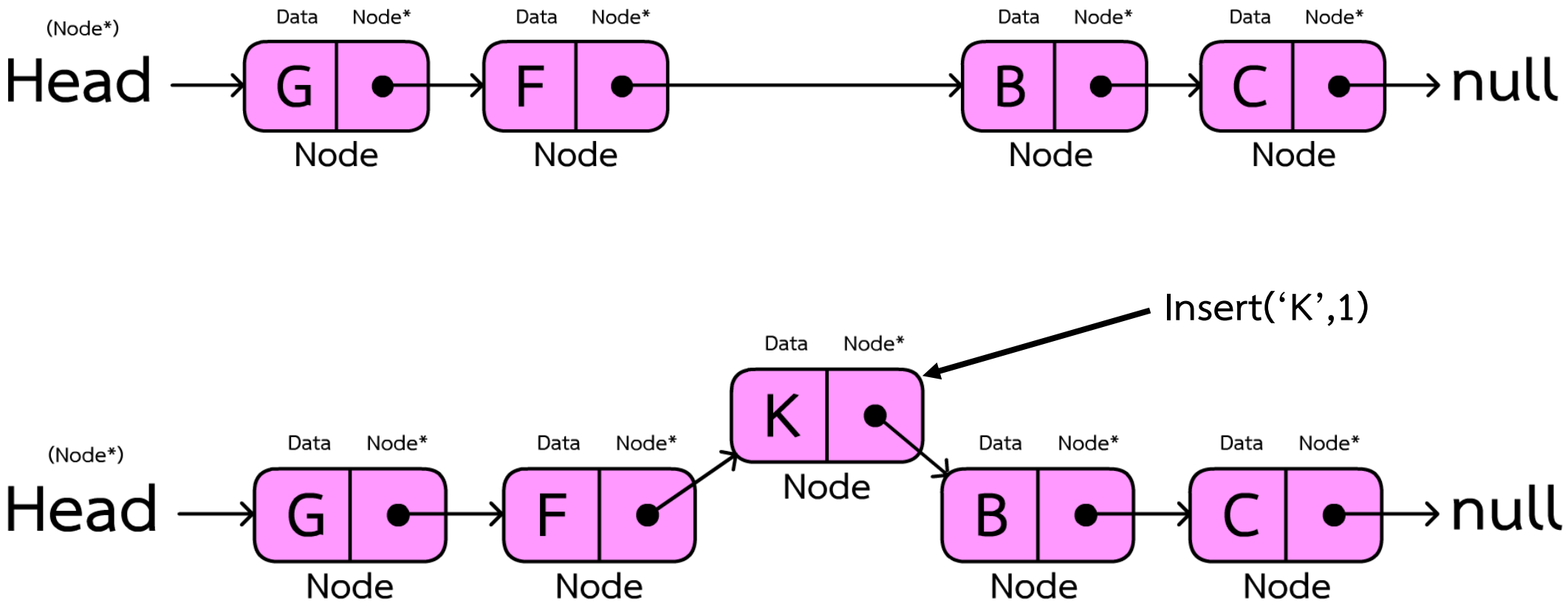
- delete to prevent memory leak
- delete เพื่อกันการกิน memory

```
void remove_at(int index){ // insert to first of list
    Node *p = head;
    for(int i=0;i<index-1;i++){ // traverse to index - 1
        p = p->next;
    }

    Node *old_node = p->next;
    p->next = (p->next)->next; // step 2
    delete old_node;
}
```



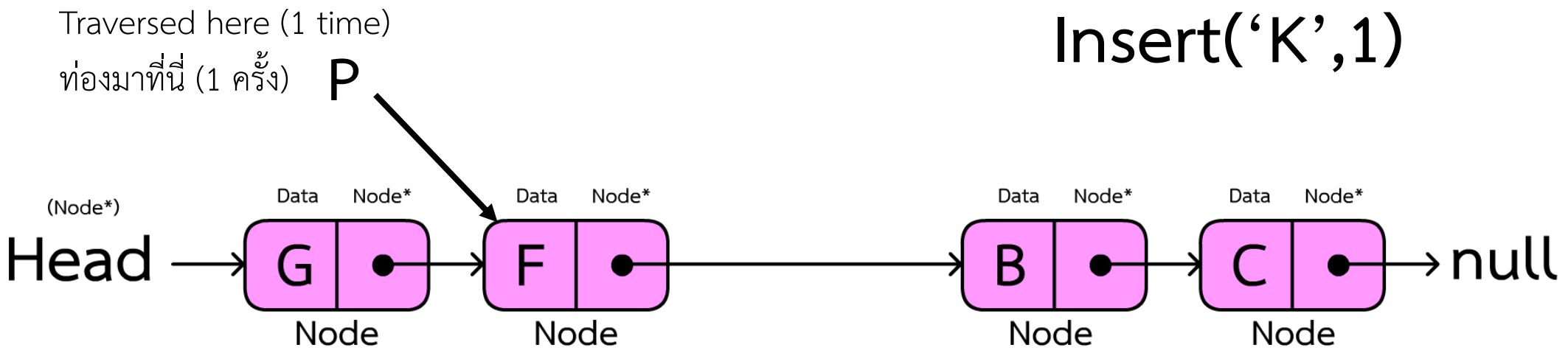
insert_after(item,index)

insert new member at index / แทรก item ตัวใหม่เข้าไปหลังจาก index



insert_after(item,index) step by step

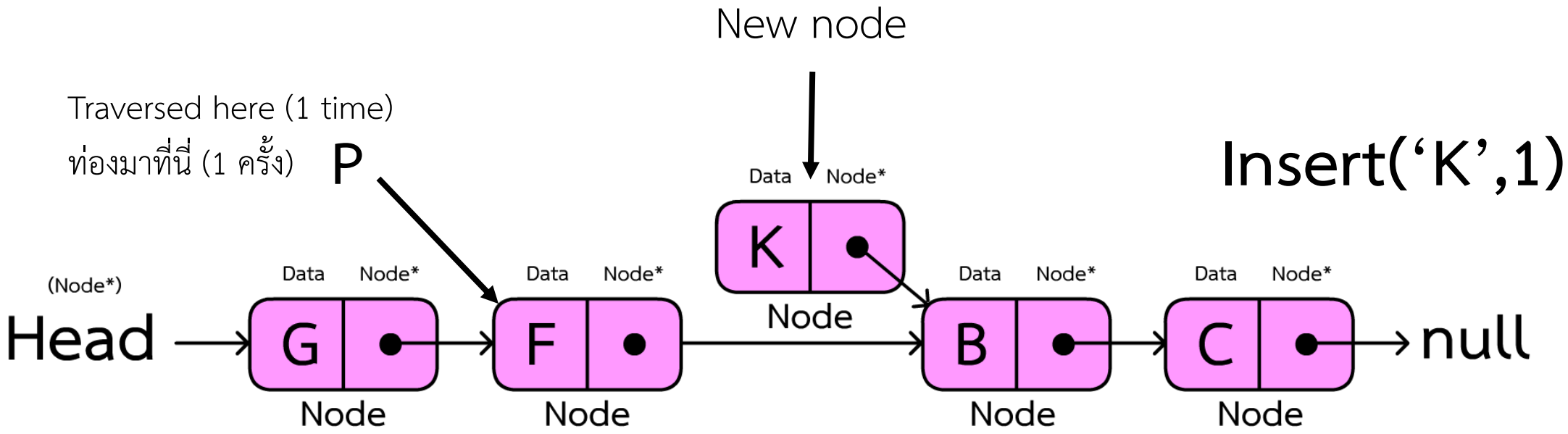
- traverse in list index time / ท่อง ไปใน list นจำนวน index ครั้ง



insert_after(item,index) step by step

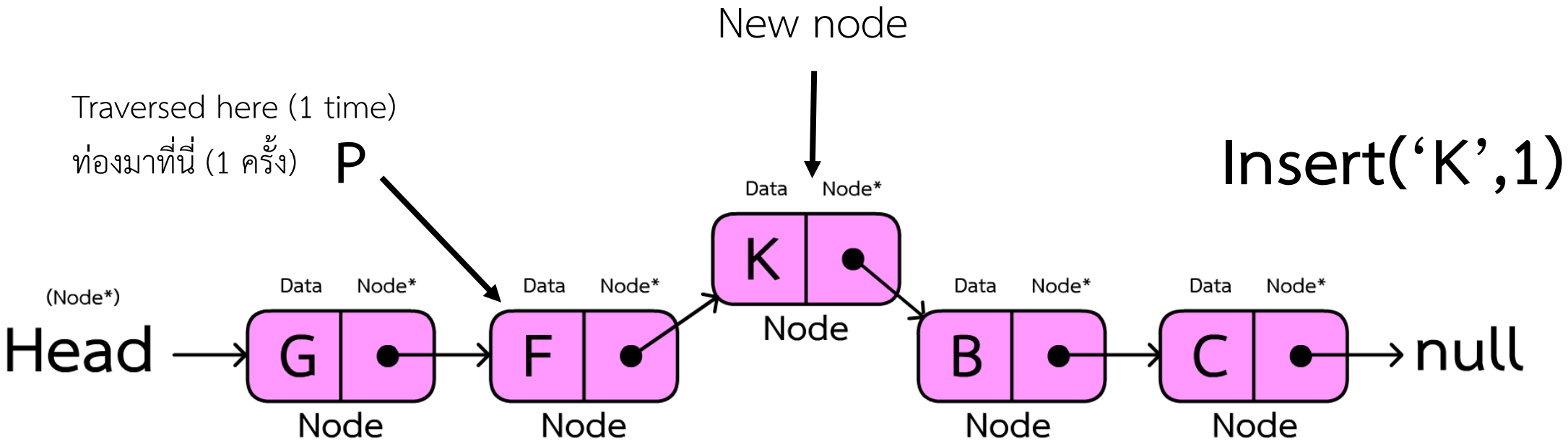
- create new node and point to the same p->next

- สร้าง node ใหม่และชี้ไปที่เดียวกันกับ p->next



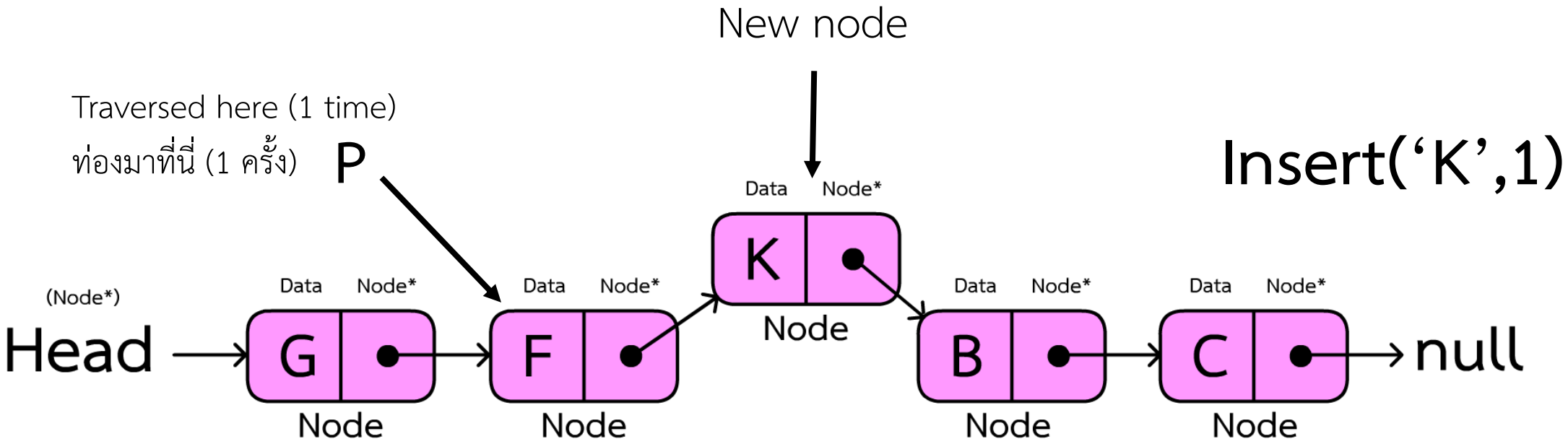
insert_after(item,index) step by step

- point Traversed member to new node
- กำหนดให้ member ที่ท่องไป ให้ชี้ไปที่ node ตัวใหม่

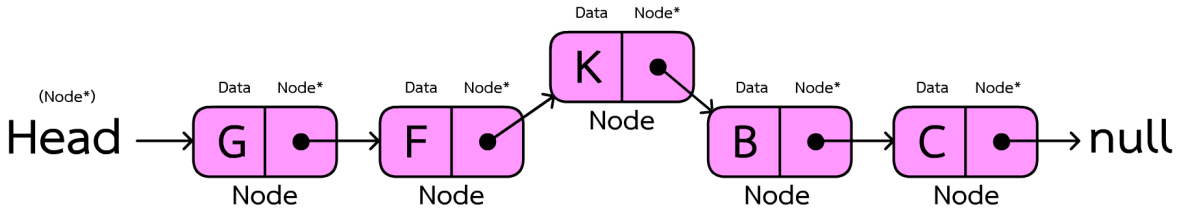


insert_after(item,index) step by step

- point Traversed member to new node
- กำหนดให้ member ที่ท่องไป ให้ชี้ไปที่ node ตัวใหม่



Check




```
Linked_list l1;
l1.push_back('A');
l1.push_back('B');
l1.push_back('C');
l1.print();
l1.push_front('F');
l1.push_front('G');
l1.print();
l1.remove_at(2);
l1.print();
l1.insert_after('K',1);
l1.print();
```

Result:

A->B->C->null

G->F->A->B->C->null

G->F->B->C->null

G->F->K->B->C->null

Advantage

- think about you want to insert item in middle of structure

- How to do with array ?

- How to do with linked list ?

- หากต้องการแทรกสมาชิกเข้าไปกลาง structure

- ถ้าใช้ array จะทำอย่างไร ?

- หากใช้ Linked list ต้องทำอย่างไร ?

Asymptotic notation

Insert_after

- Big O ->
- Big Omega ->

remove_at

- Big O ->
- Big Omega ->

push_front

- Big Theta ->

push_back

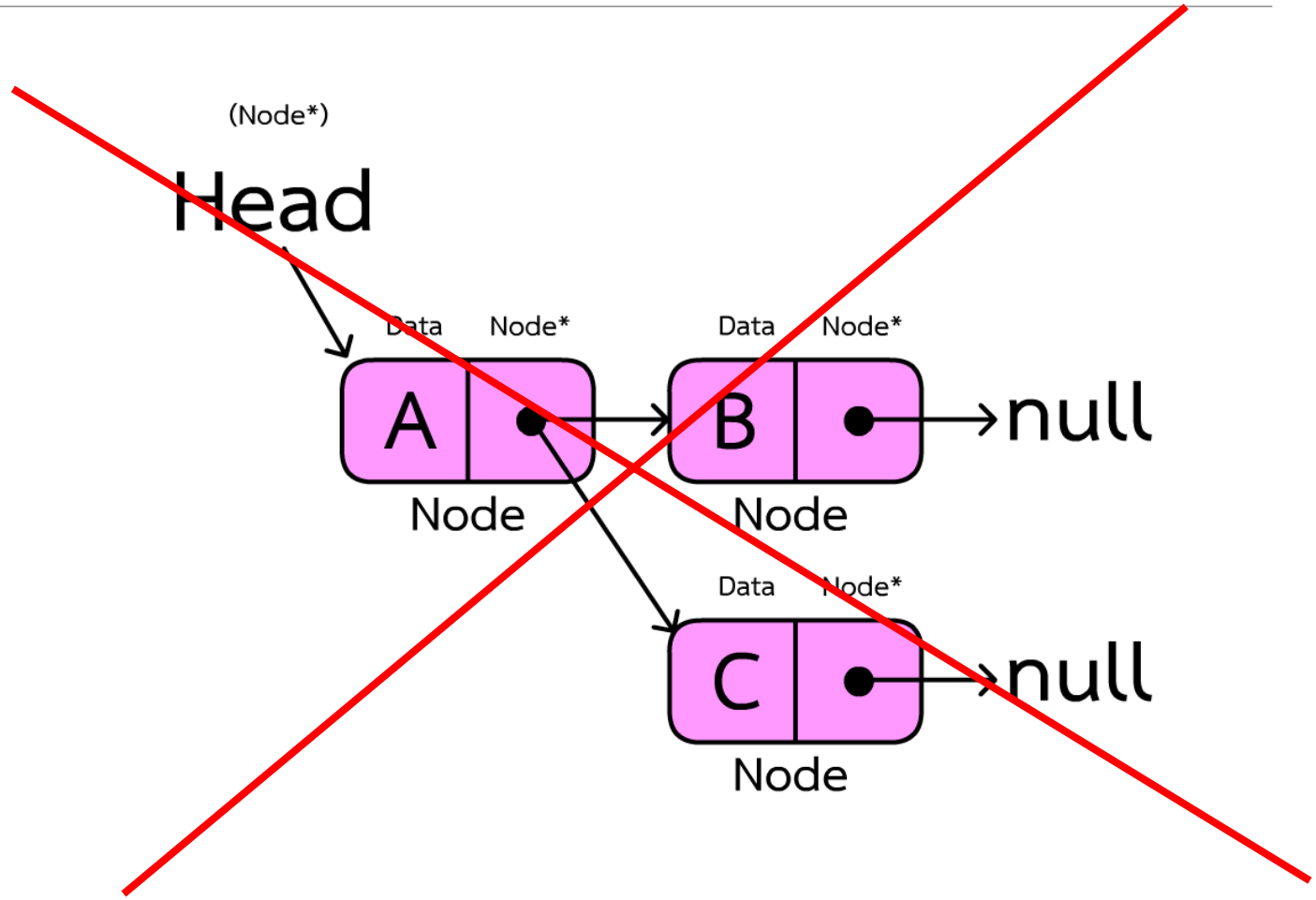
- Big Theta ->

remove_at

- Big O ->
- Big Omega ->

Not a binary tree

- Linked list is a linear data structure
- Linked list คือ linear data structure



LAB