


Data Structures

Chapter 4

1. Singly Linked List
 - **Pointer & Linking**
 - Singly Linked List (SLL)
 - SLL Basic Operations
 - SLL Advanced Operations
2. Doubly Linked List

A pair of black-rimmed glasses is placed on an open book. The book's pages are filled with text, but they are out of focus. The scene is lit with a warm, golden light, creating a soft and contemplative atmosphere.

내 아들들을 먼 곳에서 이끌며 내 딸들을 땅 끝에서 오게 하며 내 이름으로 불려지는 모든 자 곧 내가 내 영광을 위하여 창조한 자를 오게 하라 그를 내가 지었고 그를 내가 만들었노라 (사43:6-7)

수고하고 무거운 짐 진 자들아 다 내게로 오라 내가 너희를 쉬게 하리라 나는 마음이 온유하고 겸손하니 나의 멍에를 메고 내게 배우라 그리하면 너희 마음이 쉼을 얻으리니 이는 내 멍에는 쉽고 내 짐은 가벼움이라 하시니라 (마11:28-30)

Pointer reviewed – Example 1

```
int z = 25;    // define an int

int* p;        // declare an integer pointer
p = &z;        // p holds the address of z
               // p points z
```

??

z

Pointer reviewed – Example 1

```
int z = 25;    // define an int

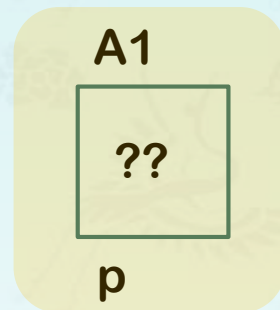
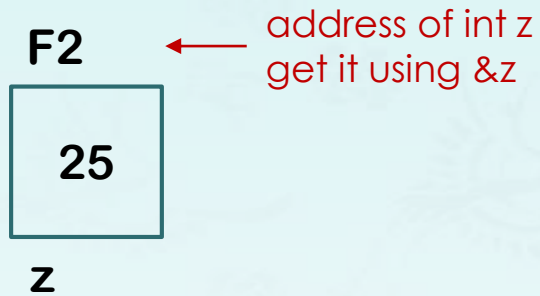
int* p;        // declare an integer pointer
p = &z;        // p holds the address of z
               // p points z
```



Pointer reviewed – Example 1

```
int z = 25;    // define an int
```

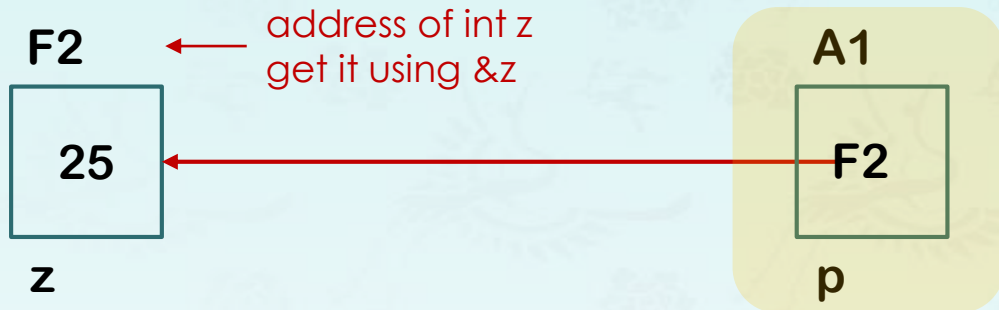
```
int* p;        // declare an integer pointer  
p = &z;        // p holds the address of z  
               // p points z
```



Pointer reviewed – Example 1

```
int z = 25;    // define an int

int* p;        // declare an integer pointer
p = &z;        // p holds the address of z
               // p points z
```

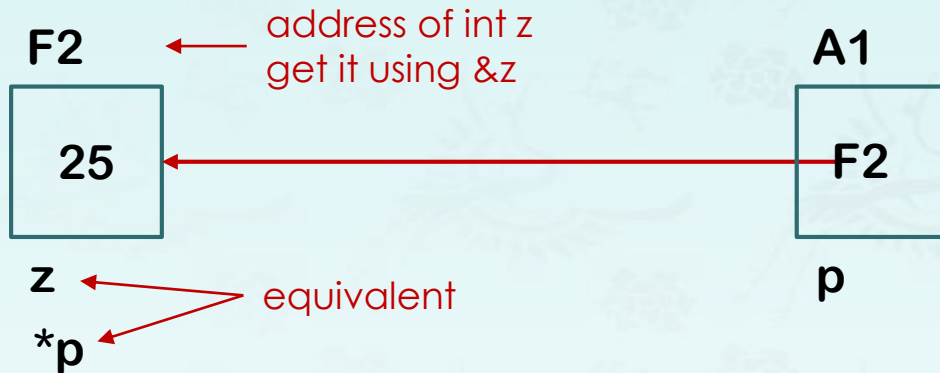


What is *p?

Pointer reviewed – Example 1

```
int z = 25;    // define an int

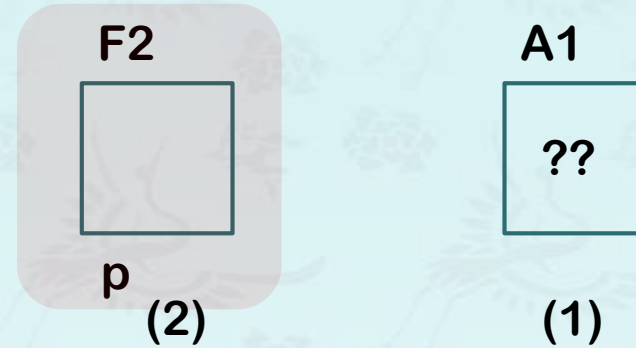
int* p;        // declare an integer pointer
p = &z;        // p holds the address of z
               // p points z
```



If **p** is a pointer, ***p** is the thing it is pointing at.
Therefore, ***p = 25**;

Pointer reviewed – Example 2

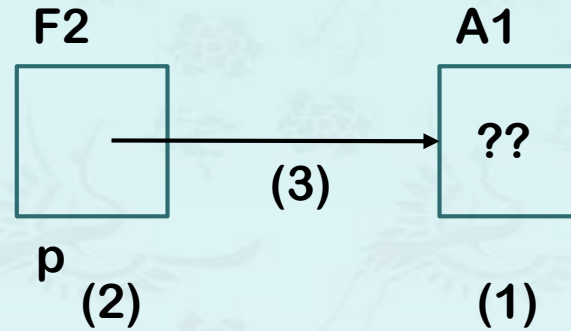
```
int* p = new int;
```



- 1) `new int;` declares an integer storage space in memory
- 2) `int *p` makes create a pointer to point an integer storage

Pointer reviewed – Example 2

```
int* p = new int;
```

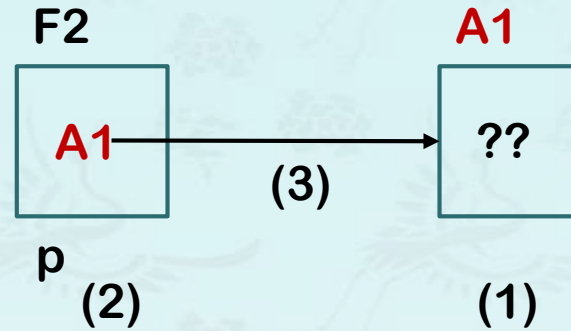


- 1) `new int;` declares an integer storage space in memory
- 2) `int *p` makes create a pointer to point an integer storage
- 3) `=` makes the pointer point at an integer storage.

What is really happening in (3)?

Pointer reviewed – Example 2

```
int* p = new int;
```

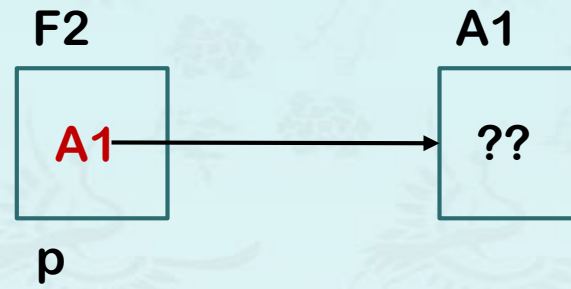


- 1) `new int;` declares an integer storage space in memory
- 2) `int *p` makes create a pointer to point an integer storage
- 3) `=` makes the pointer point at an integer storage.

What is really happening in (3)?

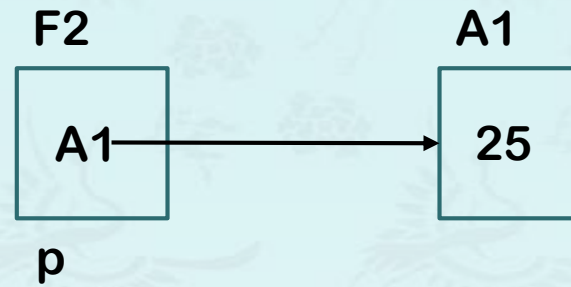
Pointer reviewed – Example 2

```
int* p = new int;  
*p = 25;
```



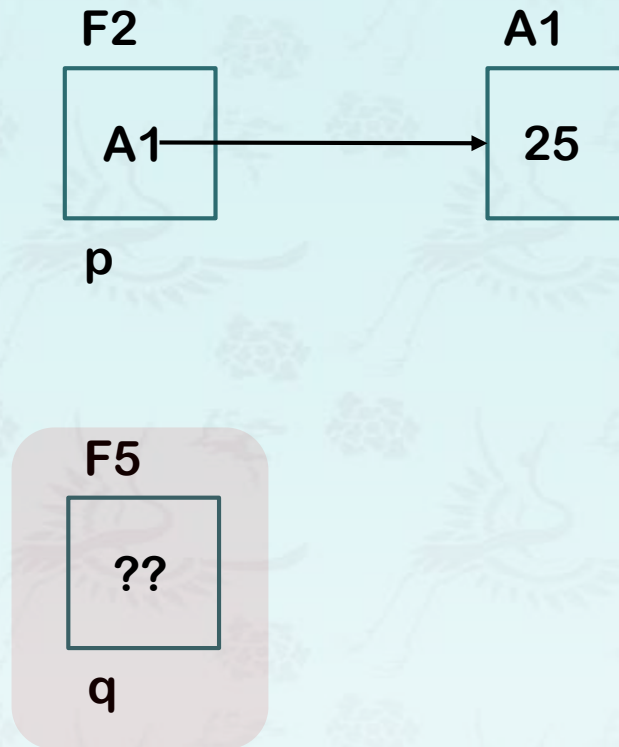
Pointer reviewed – Example 2

```
int* p = new int;  
*p = 25;  
cout << *p << endl;  
int* q;
```



Pointer reviewed – Example 2

```
int* p = new int;  
*p = 25;  
cout << *p << endl;  
int* q;
```

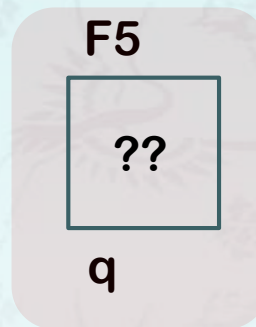


- 1) `int* q;` declares a pointer,
- 2) but it doesn't point anywhere (it's uninitialized) and
- 3) the statement doesn't assign any memory for the integer data.

Pointer reviewed – Example 2

```
int* p = new int;  
*p = 25;  
cout << *p << endl;  
int* q;
```

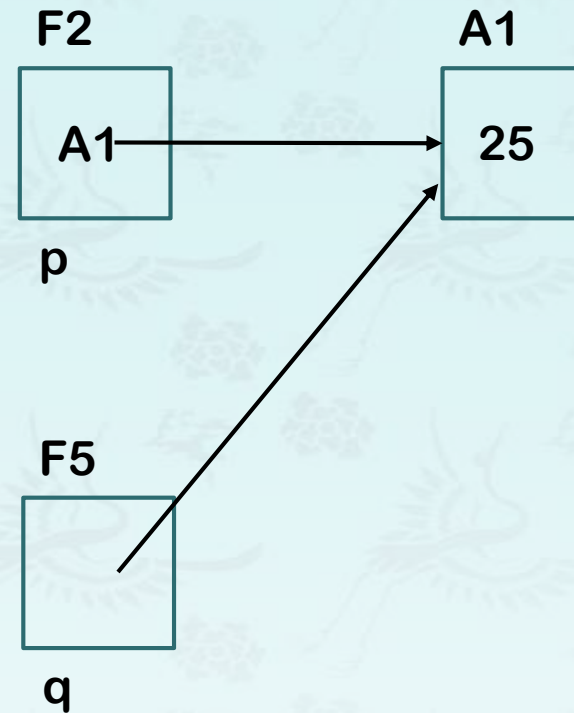
- 1) Is this valid?
 *q = 35;
- 2) Is this valid?
 int* q = p;



- 1) `int* q;` declares a pointer,
- 2) but it doesn't point anywhere (it's uninitialized) and
- 3) the statement doesn't assign any memory for the integer data.

Pointer reviewed – Example 2

```
int* p = new int;  
*p = 25;  
cout << *p << endl;  
int* q;  
q = p;
```

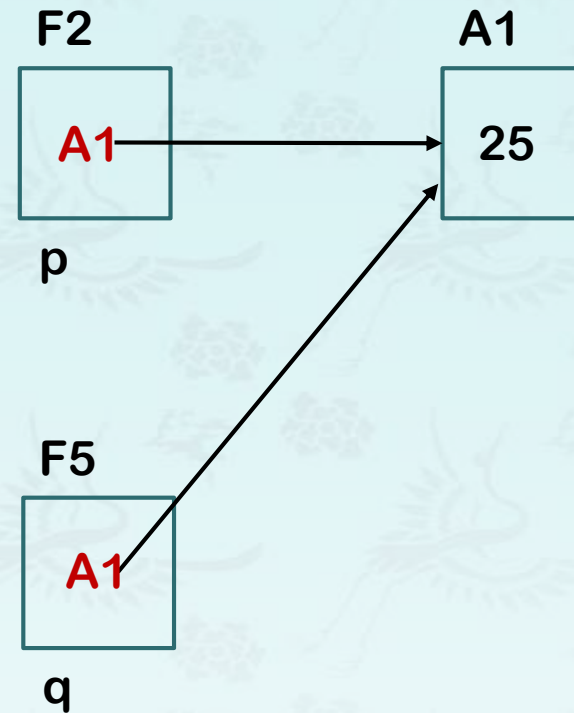


What is really happening in (`q = p`)?

- 1) it means that **q** is pointing to the same place **p** is pointing at.
- 2) it does **not** mean that **q** is pointing at **p**.

Pointer reviewed – Example 2

```
int* p = new int;  
*p = 25;  
cout << *p << endl;  
int* q;  
q = p;
```



What is really happening in `q = p`?

- 1) it means that **q** is pointing to the same place **p** is pointing at.
- 2) it does **not** mean that **q** is pointing at **p**.

Pointer reviewed

```
int* p = new int;  
*p = 25;  
cout << *p << endl;  
int* q;  
q = p;  
cout << *q;
```

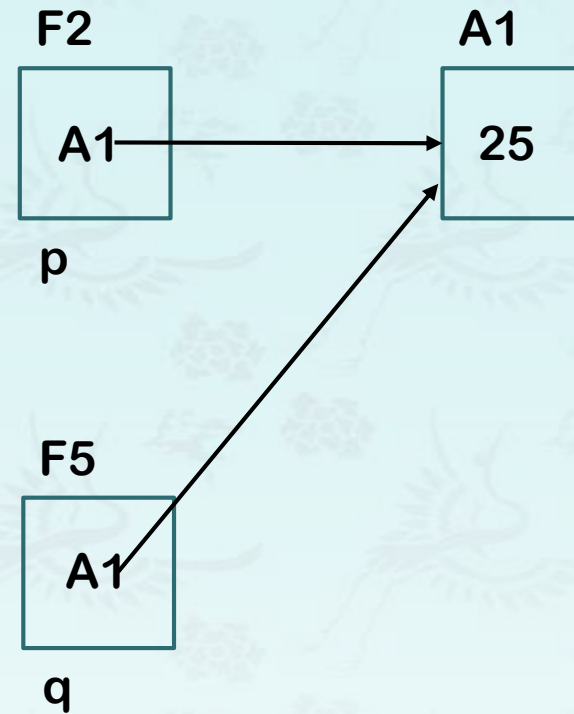


```
int* p = new int(25);  
  
cout << *p << endl;  
int* q = p;  
  
cout << *q;
```


Pointer reviewed – Quiz Time

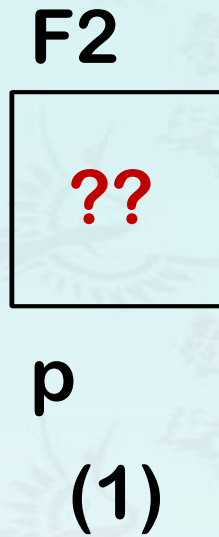
```
int* p = new int(25);  
cout << *p << endl;  
int* q = p;  
cout << *q;  
*q = 34;  
q = new int(56); // keep this line  
p = new int(78); // keep this line  
delete p;  
delete q;
```

Example 2



Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p   
    ...  
}
```



- (1) This code declares a **Node pointer, p**, and
- (2) allocate memory space for a **new Node** and
- (3) make **p point the Node**.

Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    new Node;  
    ...  
}
```

A6

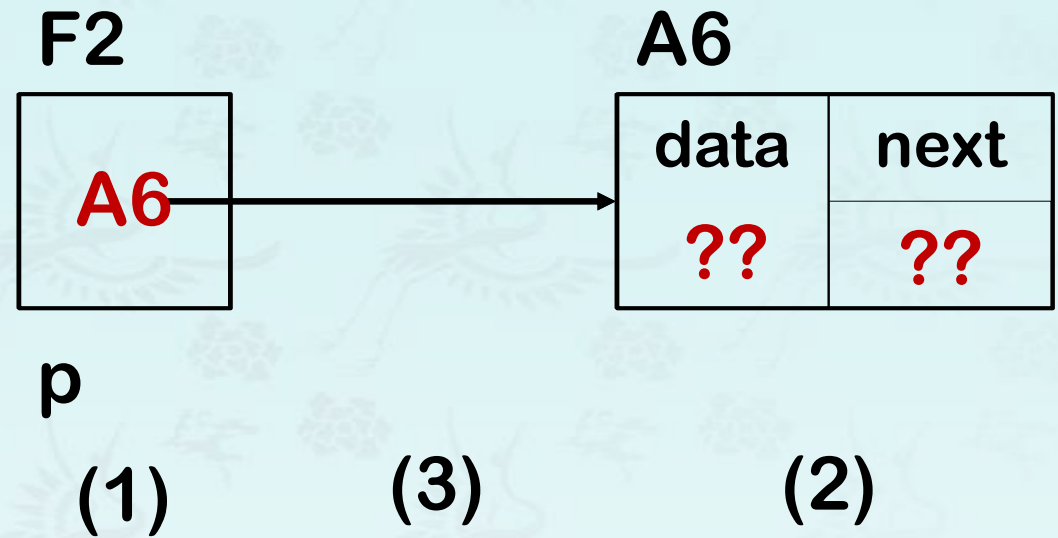
data	next
??	??

(2)

- (1) This code declares a **Node pointer, p**, and
- (2) allocate memory space for a **new Node** and
- (3) make **p point the Node**.

Pointers Linked

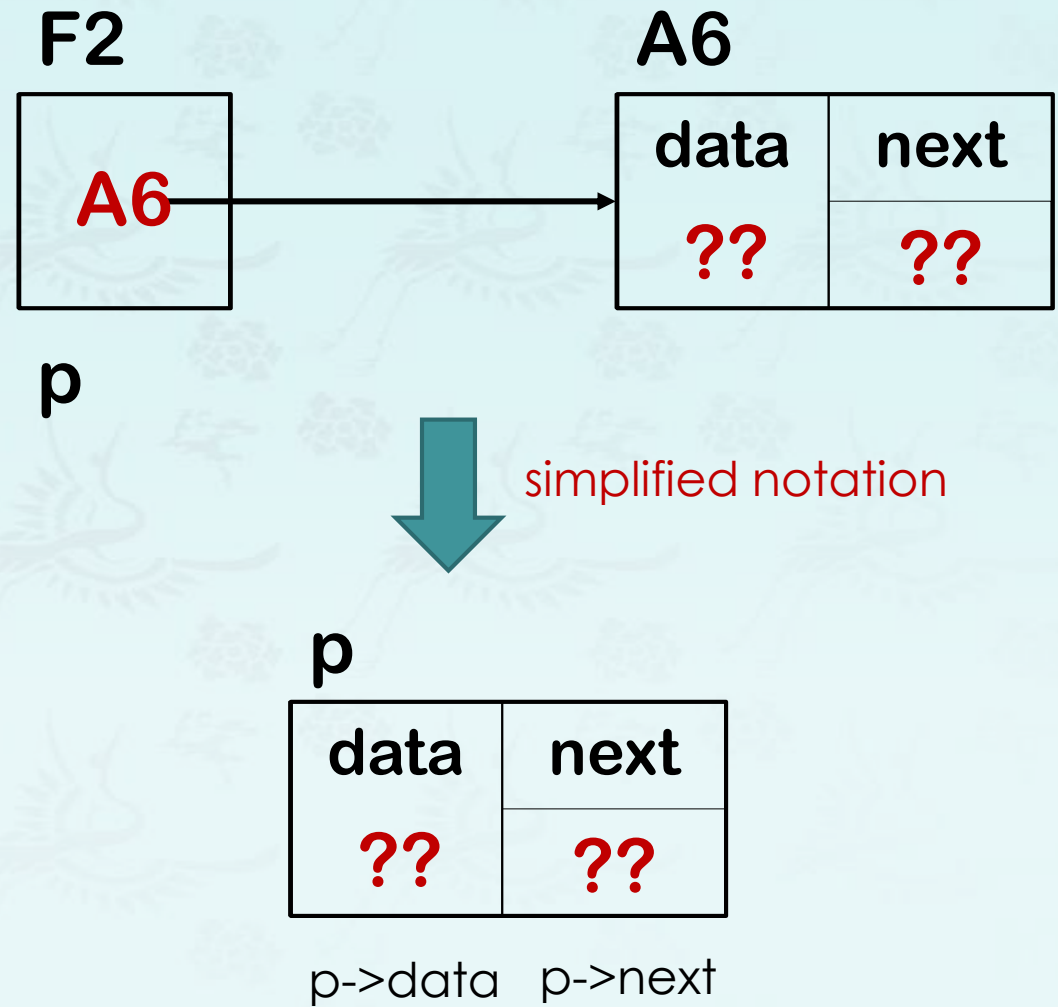
```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p = new Node;  
    ...  
}
```



- (1) This code declares a **Node pointer, p**, and
- (2) allocate memory space for a **new Node** and
- (3) make **p point the Node**.

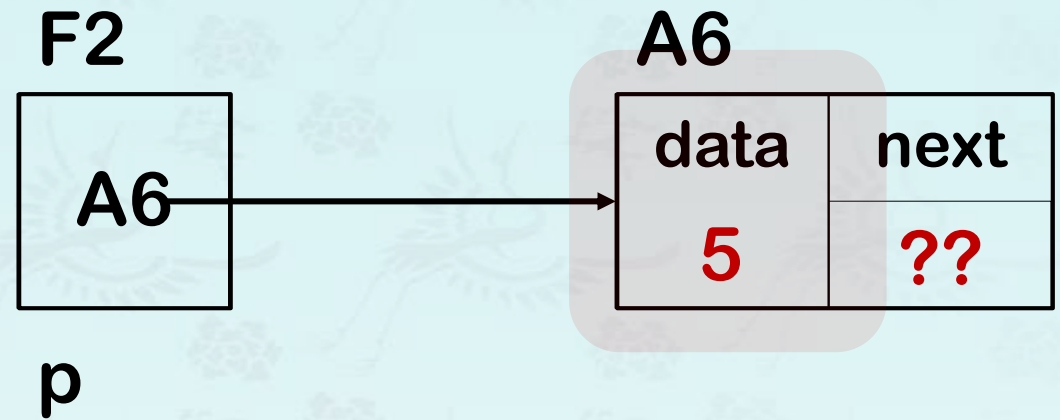
Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p = new Node;  
    p->data = 5;  
    p->next = nullptr;  
    Node* q = new Node;  
    q->data = 3;  
    q->next = nullptr;  
    p->next = q;  
}
```



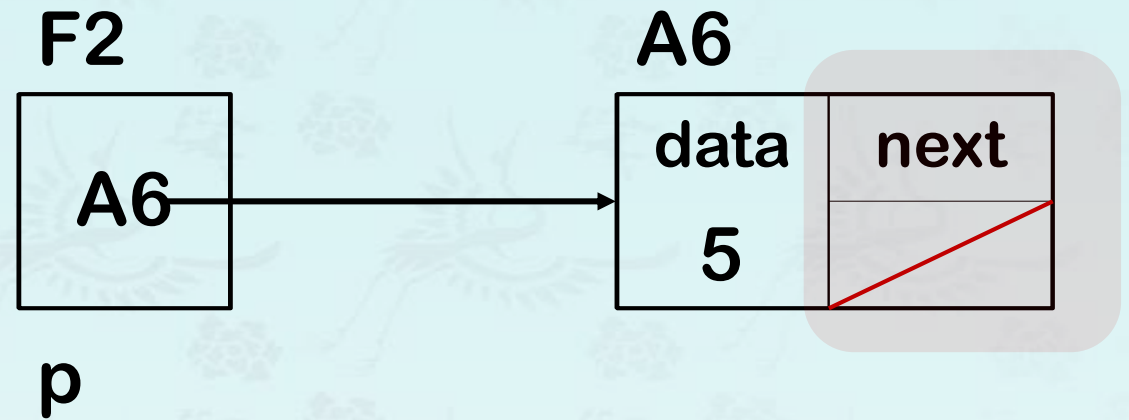
Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p = new Node;  
    p->data = 5;  
    p->next = nullptr;  
    Node* q = new Node;  
    q->data = 3;  
    q->next = nullptr;  
    p->next = q;  
}
```



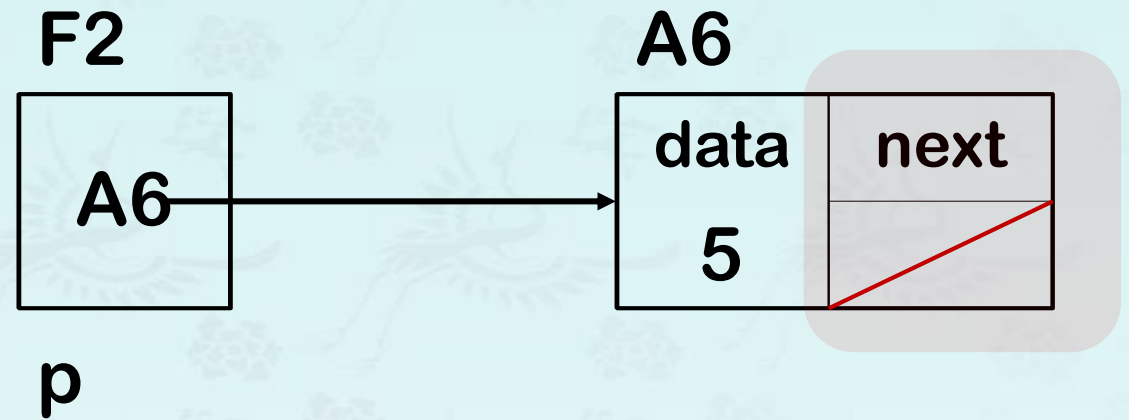
Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p = new Node;  
    p->data = 5;  
    p->next = nullptr;  
    Node* q = new Node;  
    q->data = 3;  
    q->next = nullptr;  
    p->next = q;  
}
```



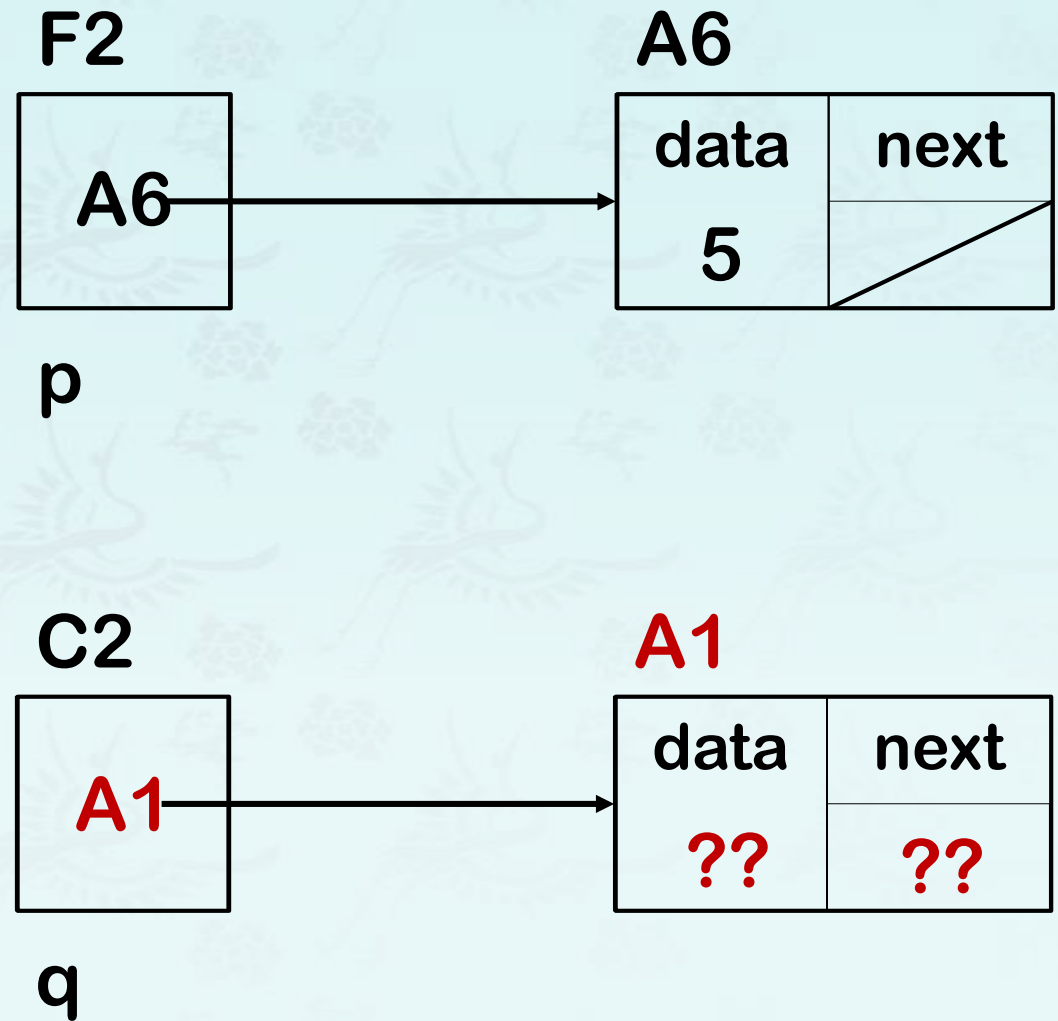
Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p = new Node;  
    p->data = 5;  
    p->next = nullptr;  
    Node* q = new Node;  
    q->data = 3;  
    q->next = nullptr;  
    p->next = q;  
}
```



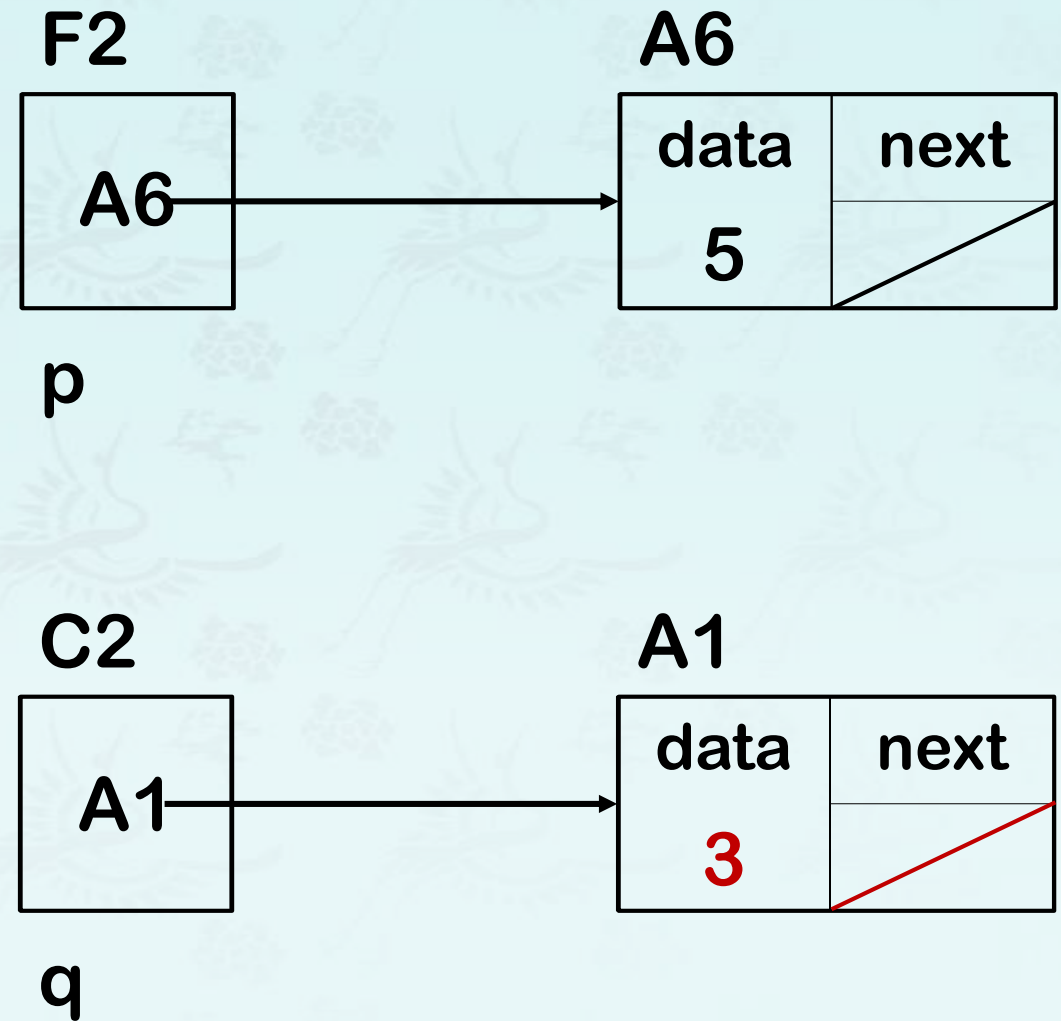
Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p = new Node;  
    p->data = 5;  
    p->next = nullptr;  
    Node* q = new Node;  
    q->data = 3;  
    q->next = nullptr;  
    p->next = q;  
}
```



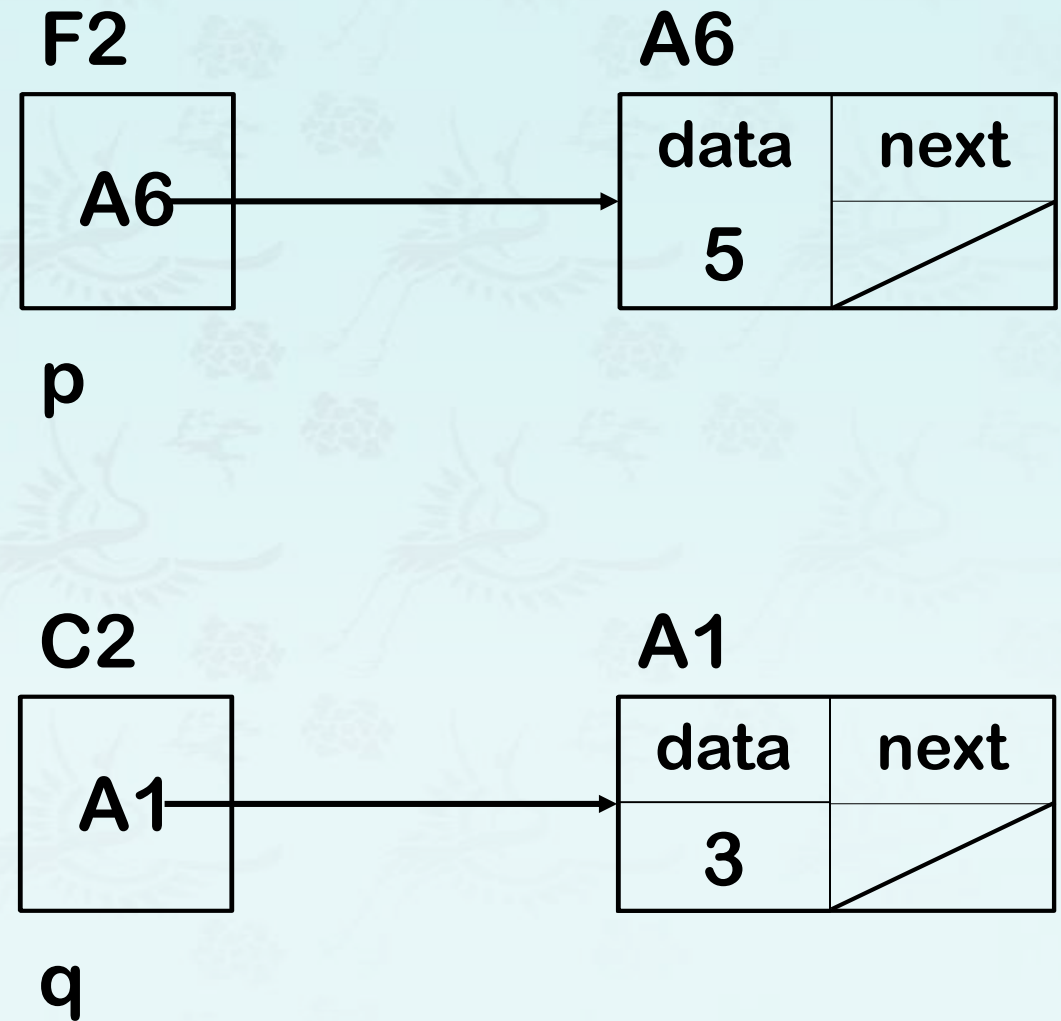
Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p = new Node;  
    p->data = 5;  
    p->next = nullptr;  
    Node* q = new Node;  
    q->data = 3;  
    q->next = nullptr;  
    p->next = q;  
}
```



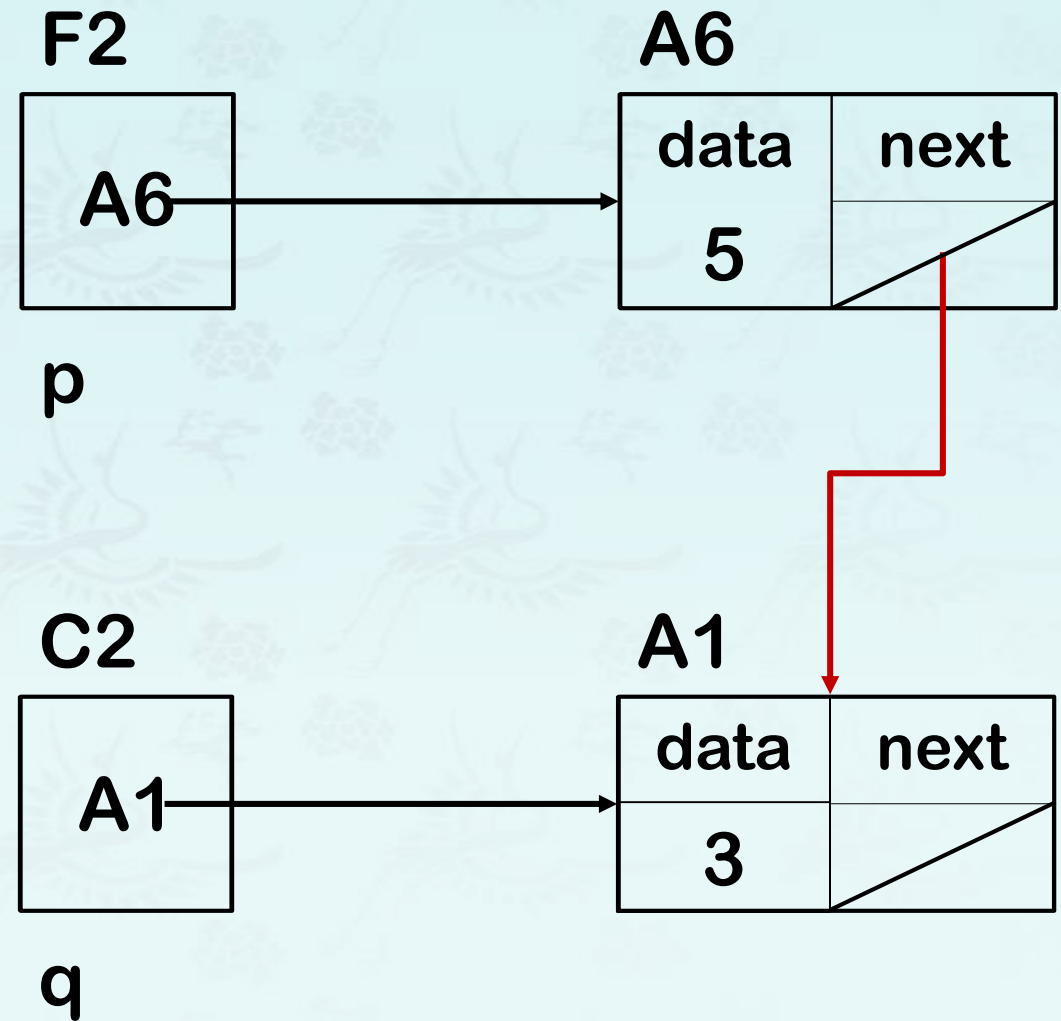
Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p = new Node;  
    p->data = 5;  
    p->next = nullptr;  
    Node* q = new Node;  
    q->data = 3;  
    q->next = nullptr;  
    p->next = q;  
}
```



Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* p = new Node;  
    p->data = 5;  
    p->next = nullptr;  
    Node* q = new Node;  
    q->data = 3;  
    q->next = nullptr;  
    p->next = q;  
}
```

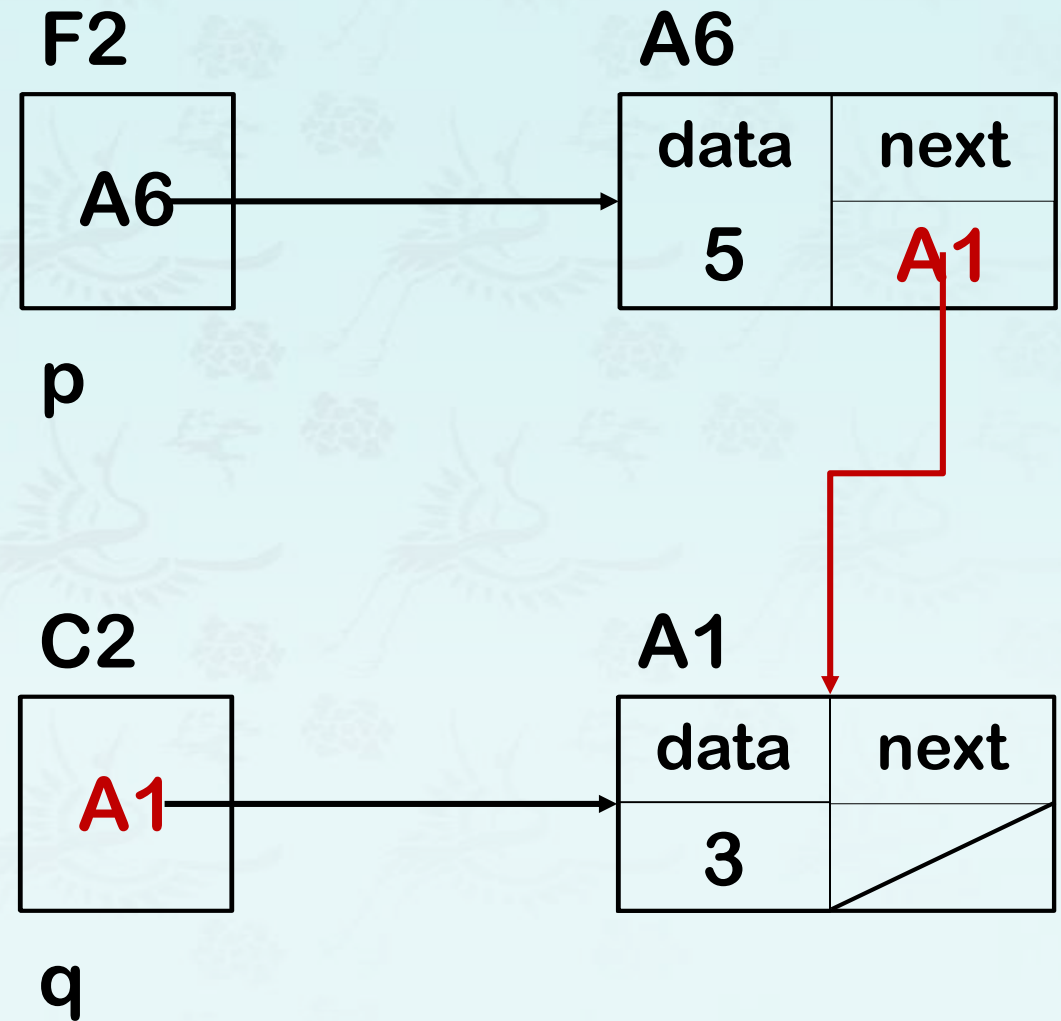


What should be update in the figure?

Pointers Linked

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

int main( ) {
    Node* p = new Node;
    p->data = 5;
    p->next = nullptr;
    Node* q = new Node;
    q->data = 3;
    q->next = nullptr;
    p->next = q;
}
```



What should be update in the figure?

p->next = A1; p->next = q;

Pointers Linked

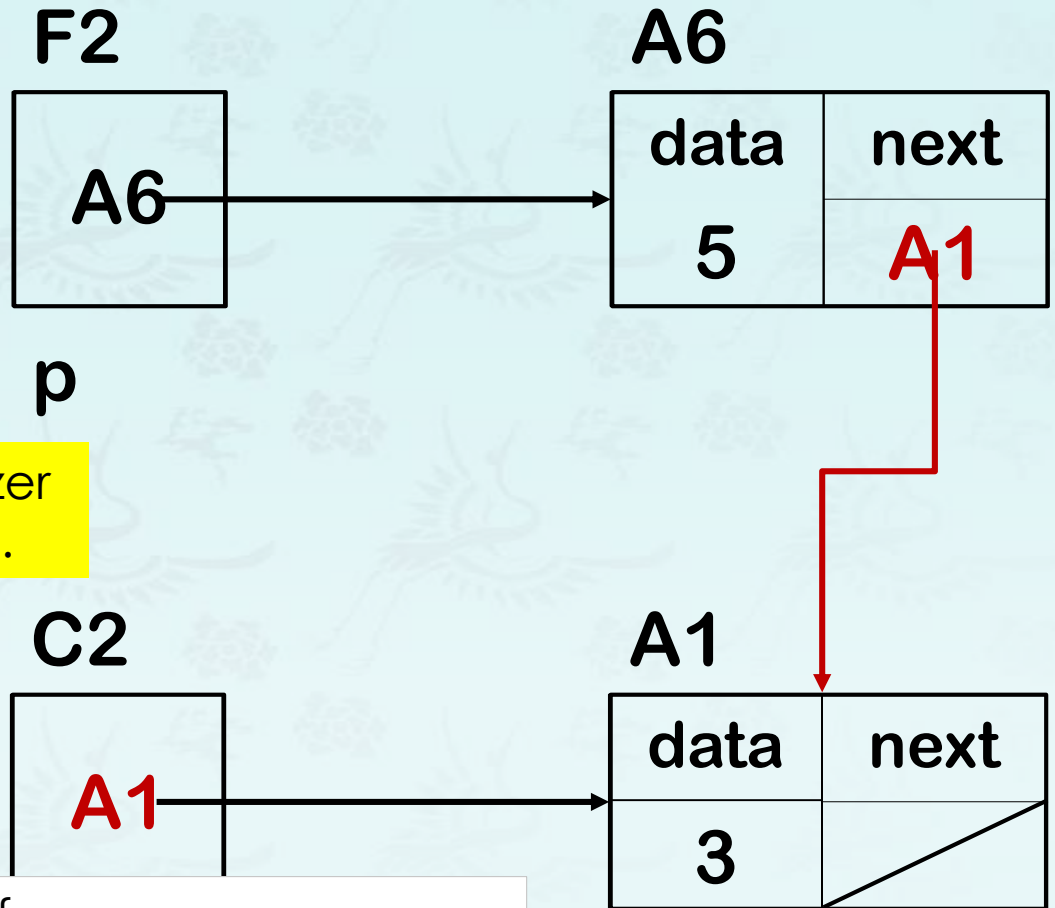
```
class Node {
public:
    int    data;
    Node* next;
};

int main( ) {
    Node* p = new Node;
    p->data = 5;
    p->next = nullptr;
    Node* q = new Node;
    q->data = 3;
    q->next = nullptr;
    p->next = q;
}
```

Recode this using initializer
(constructor) in two lines.

```
int main( ) {
    Node* p = new Node{5, nullptr};
    Node* q = new Node{3, nullptr};
    p->next = q;
}
```

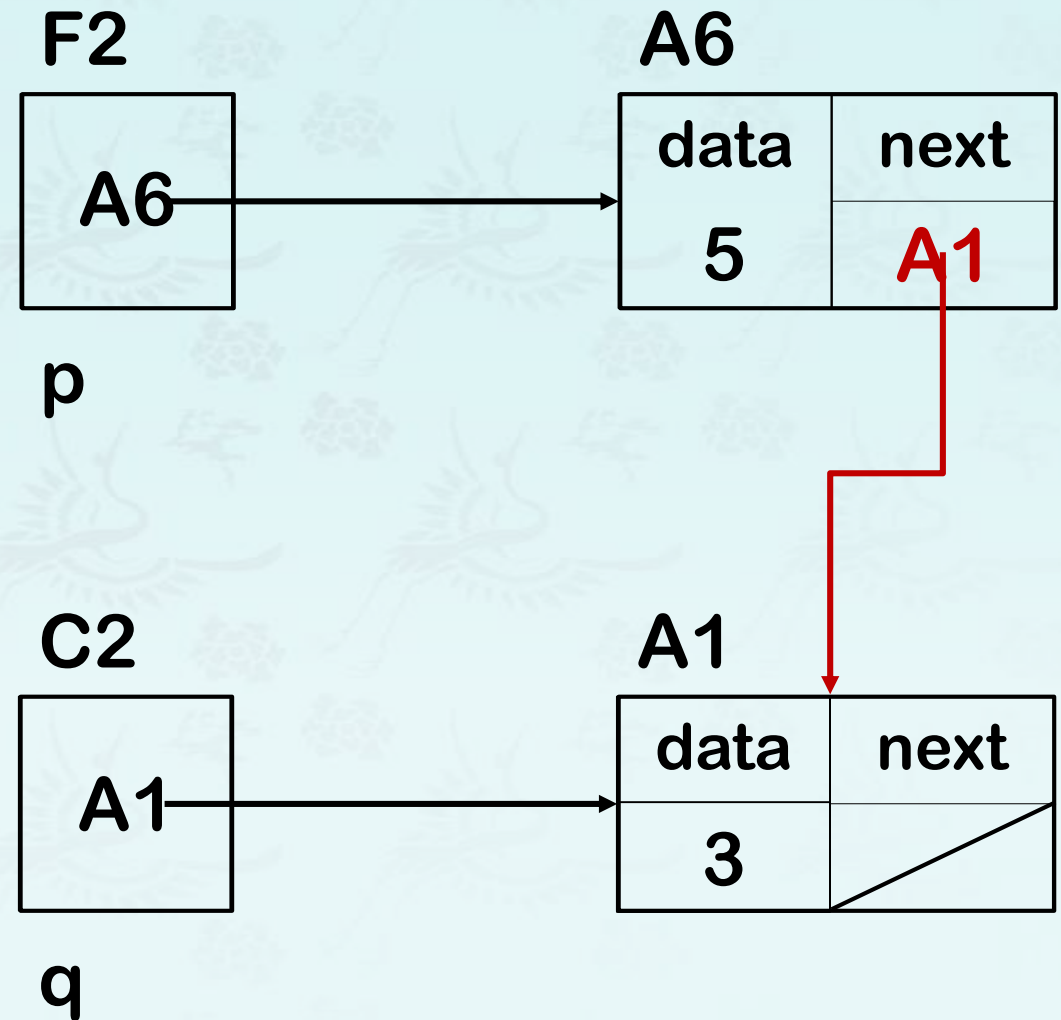
```
int main( ) {
    Node* q = new Node{3, nullptr};
    Node* p = new Node{5, q};
}
```



Pointers Linked

By stringing many of these Node objects together we can create a structure called a **singly-linked list**;

Hide p and q, and you may see a singly linked list clearly.

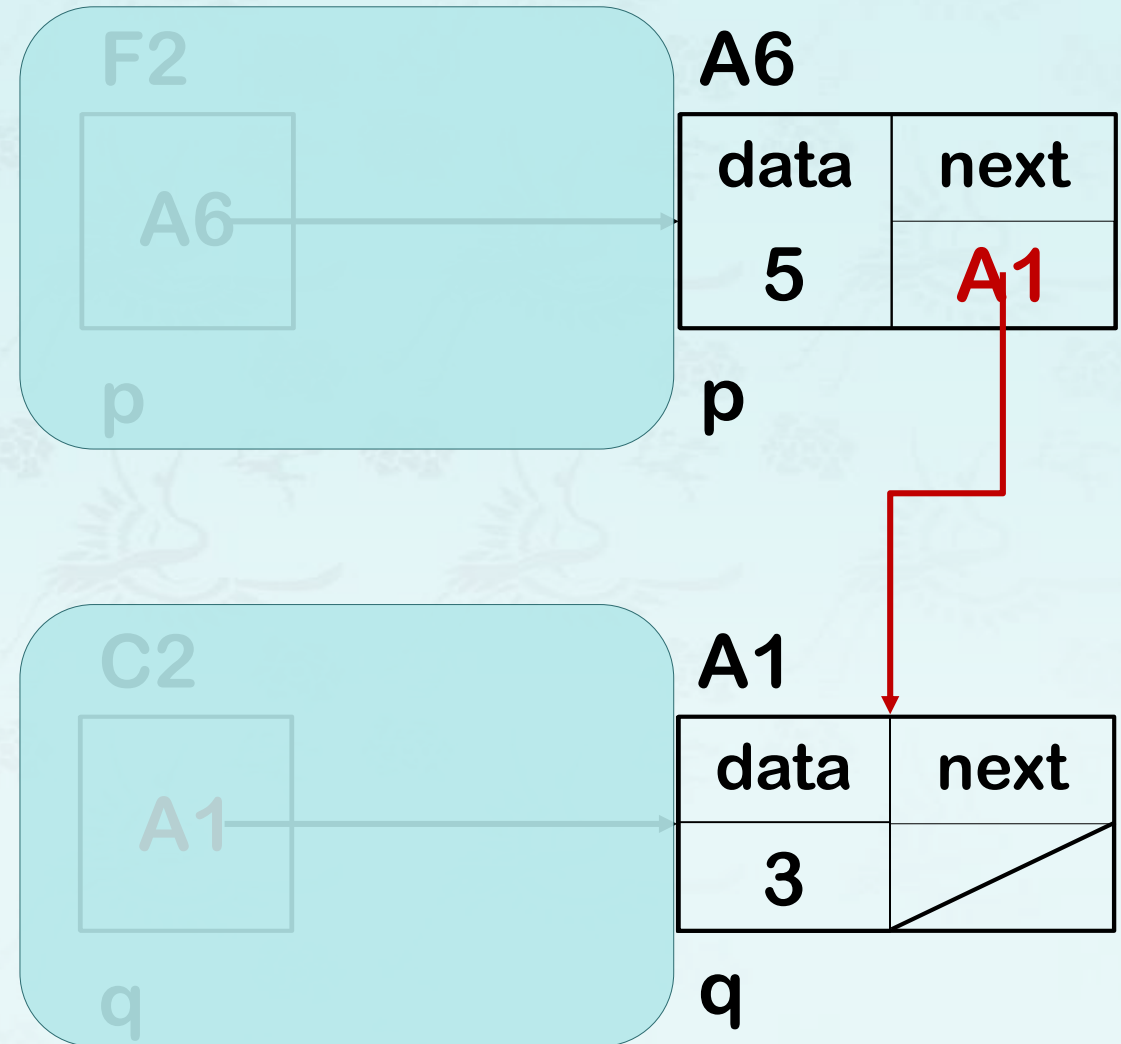


Pointers Linked

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

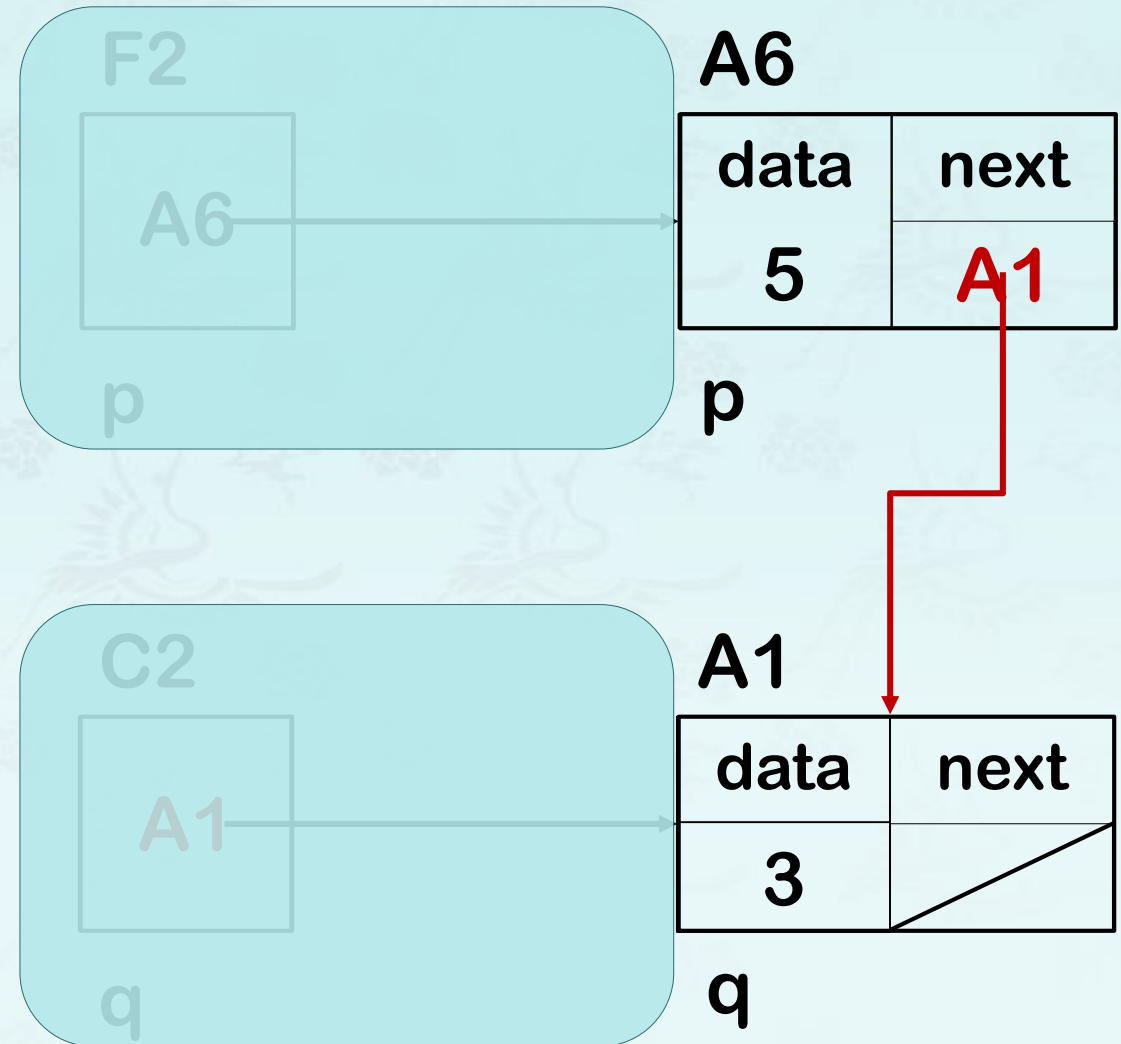
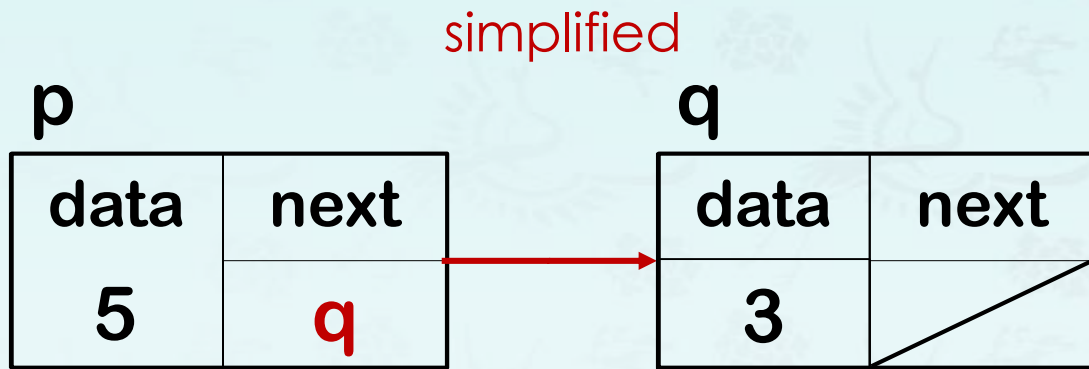
int main( ) {
    Node* q = new Node{3, nullptr};
    Node* p = new Node{5, q};
}
```

Hide p and q, and you may see a singly linked list clearly.



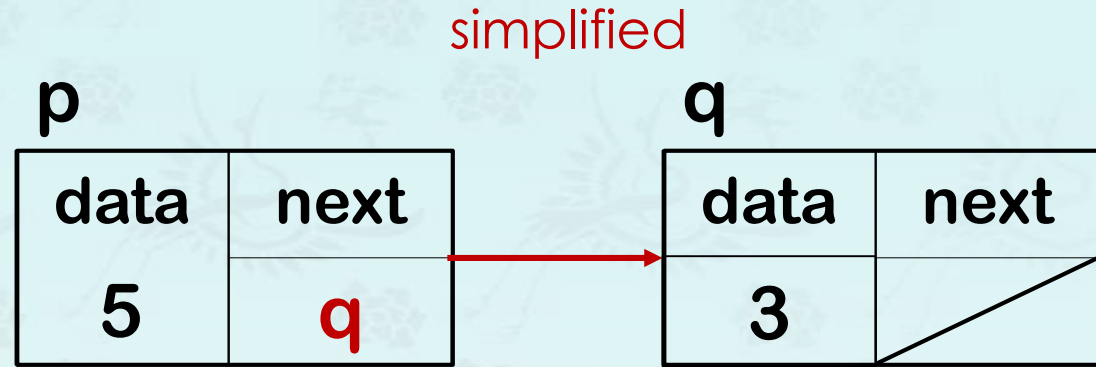
Pointers Linked

By stringing many of these Node objects together we can create a structure called a **singly-linked list**;



Pointers Linked

```
class Node {  
public:  
    int    data;  
    Node* next;  
};  
  
int main( ) {  
    Node* q = new Node{3, nullptr};  
    Node* p = new Node{5, q};  
}
```



Pointers Linked – Example

Link a, b and c nodes;

```
struct List {  
    string who;  
    char  data;  
    List *link;  
};
```

List **a, b, c**;

a.data = 'A';

b.data = 'B';

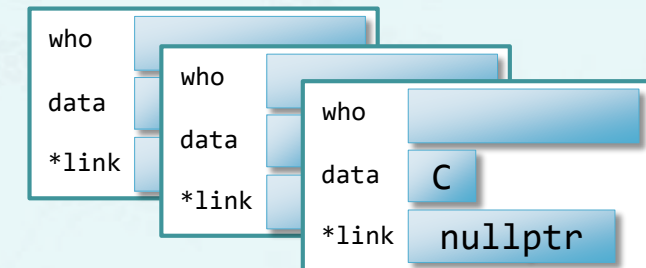
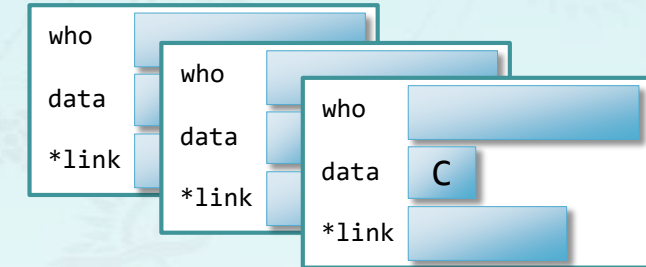
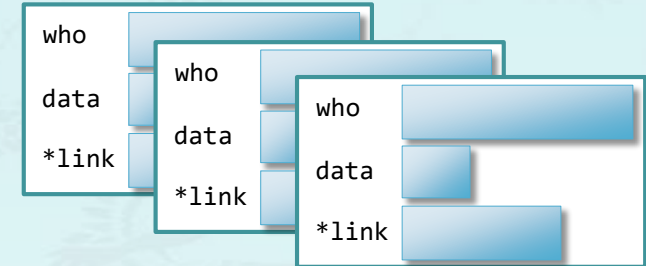
c.data = 'C';

a.link = b.link = c.link = nullptr;

List a, b, c;

a.data = 'A';
b.data = 'B';
c.data = 'C';

a.link = b.link = c.link = nullptr;



Pointers Linked – Example

Link a, b and c nodes;

```
struct List {  
    string who;  
    char  data;  
    List *link;  
};
```

List **a, b, c**;

a.data = 'A';

b.data = 'B';

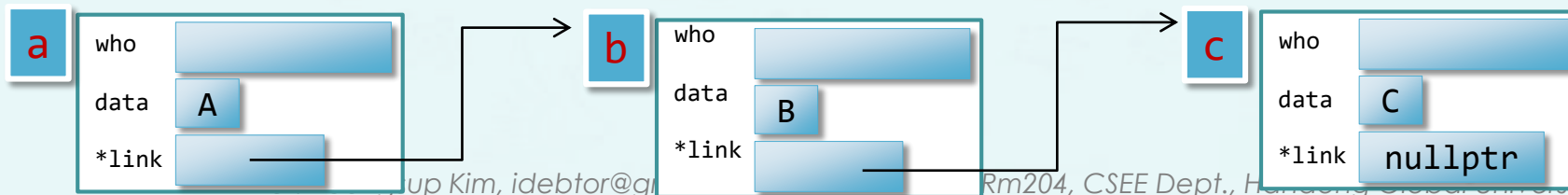
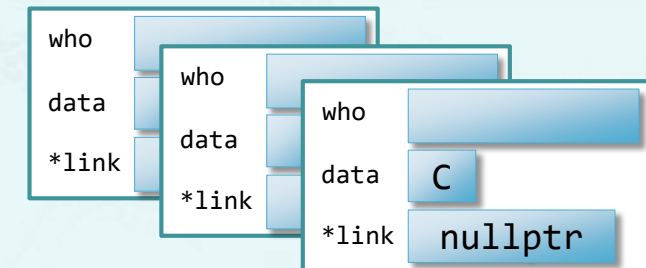
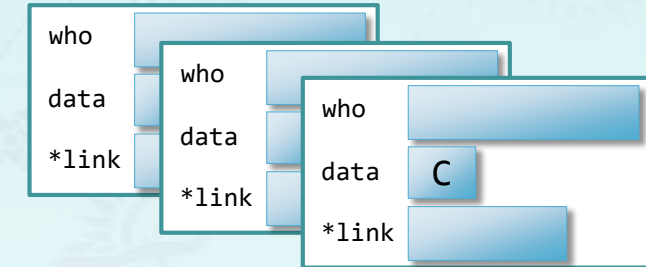
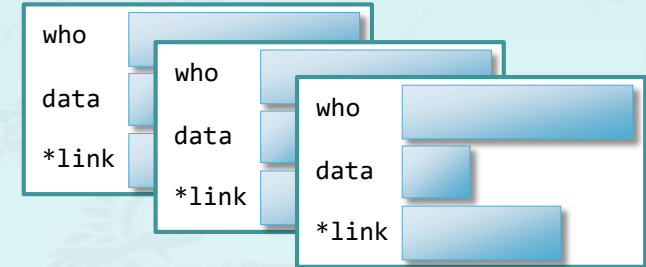
c.data = 'C';

a.link = b.link = c.link = nullptr;

List a, b, c;

a.data = 'A';
b.data = 'B';
c.data = 'C';

a.link = b.link = c.link = nullptr;



Pointers Linked – Exercise

Link a, b and c nodes;

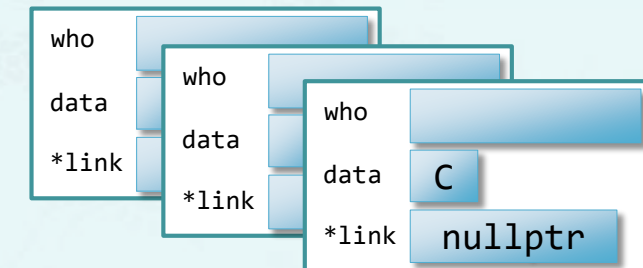
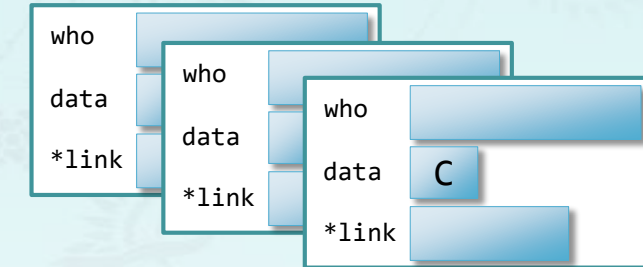
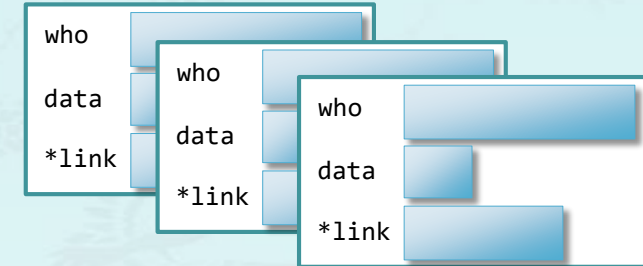
```
struct List {  
    string who;  
    char data;  
    List *link;  
};
```

```
List a, b, c;  
List *p, *q, *r;
```

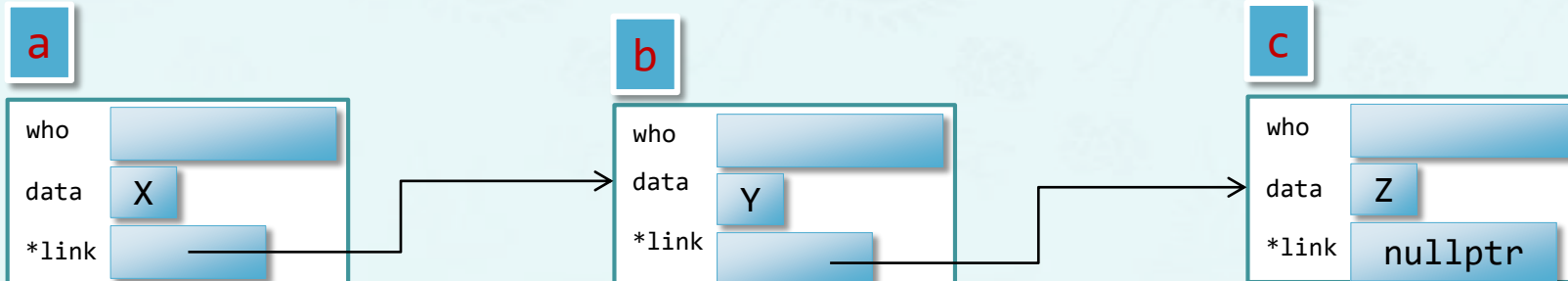
(1)

(2)

(3)



- (1) Let each p, q, and r point to a, b, and c;
- (2) Store each 'X', 'Y', and 'Z' in data using p, q, and r.
- (3) Connect them using p, q and r as shown below:



Pointers Linked

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

← constructor, destructor

```
int main( ) {  
    Node* p = new Node;  
    ...  
}
```

constructor →

destructor →

```
struct Node {  
    int    data;  
    Node* next;
```

```
    Node(int i=0, Node* n=nullptr){  
        data = i, next = n;
```

```
    }  
    ~Node() {};
```

```
};
```

```
int main( ) {  
    Node* p = new Node;
```

```
    ...
```

```
}
```

Data Structures

Chapter 4

1. Singly Linked List

- **Pointer & Linking**
- Singly Linked List (1)
- Singly Linked List (2)
- Singly Linked List Operations

2. Doubly Linked List

Summary &
quaestio quaestio 90 9 9 9 9

