

ED1021 - Introduction to computation and visualisation

L17 - Linked List

Ramanathan Muthuganapathy (<https://ed.iitm.ac.in/~raman>)

Course web page: <https://ed.iitm.ac.in/~raman/introcomp.html>

Moodle page: Available at <https://courses.iitm.ac.in/>

DMA for a structure

```
//Declaring a structure
struct AccountDetails {
    char name[20];
    int num;
    float bal;
};
typedef struct AccountDetails AD;
```

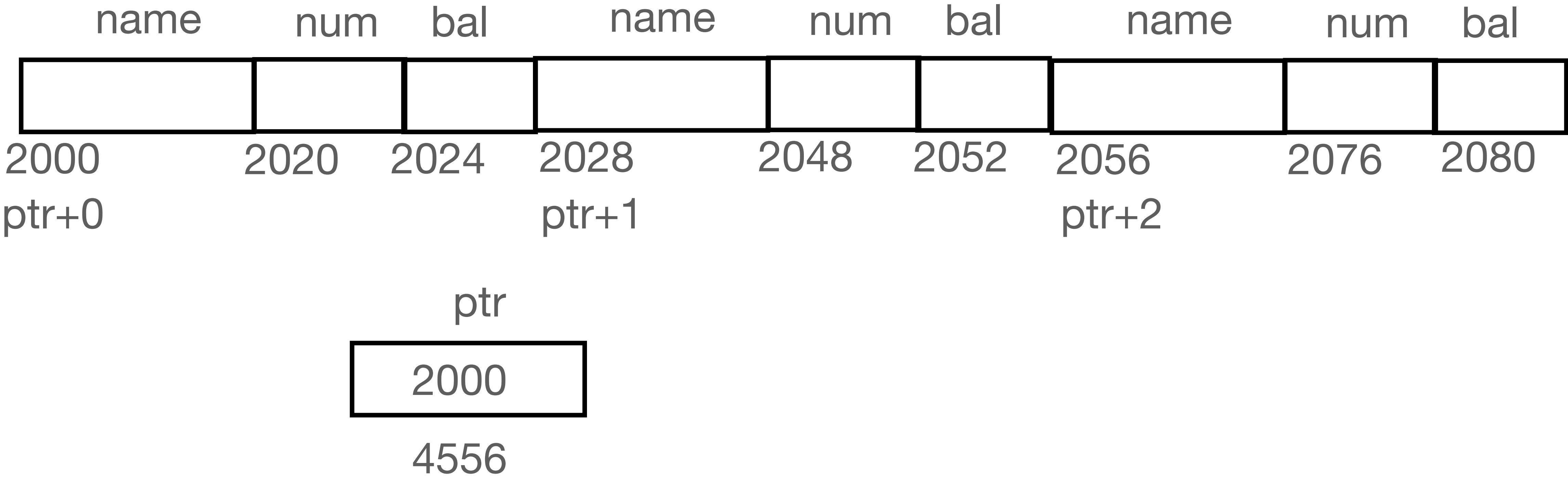
How do you do DMA for structures?

L16_DMA.c

```
AD *ptr; int m;  
  
scanf("%d", &m);  
  
ptr = ( AD *) malloc(m * sizeof(AD) );  
  
for ( ..... ) {  
  
    //Write scanf  
  
}
```

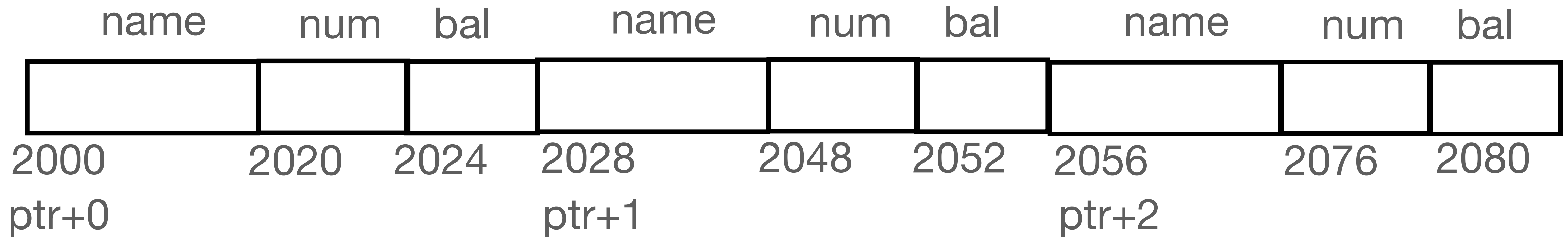
How does DMA for structures look?

if m is 3



How does DMA for structures look?

if m is 3



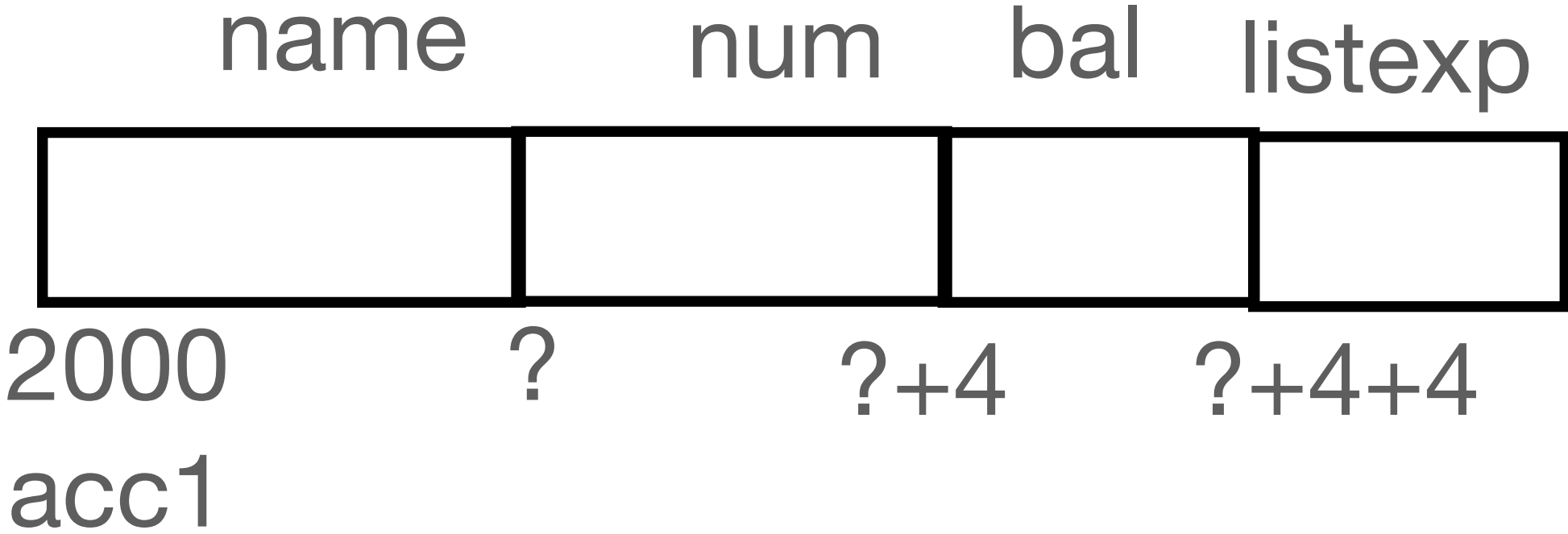
```
(ptr+i)->name  
&(ptr+i)->num  
&(ptr+i)->bal
```

```
(ptr+i)->name  
(ptr+i)->num  
(ptr+i)->bal
```

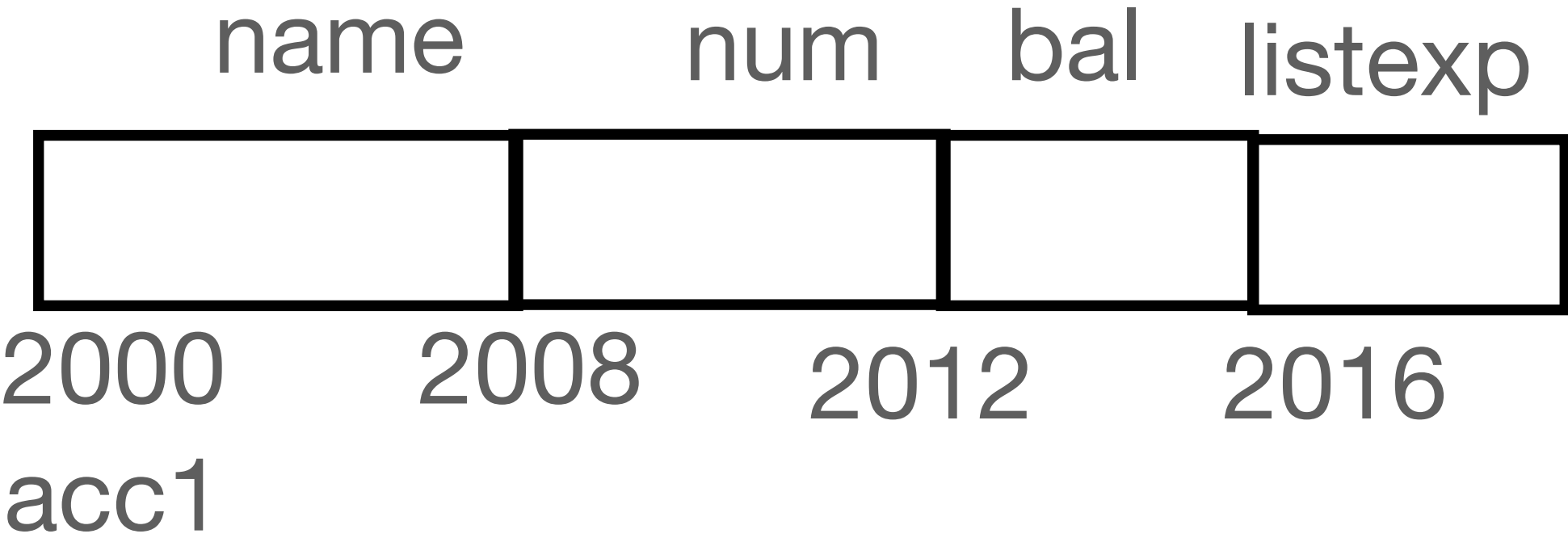
DMA for a structure member

```
//Declaring a structure
struct AccountDetails {
    char *name;
    int num;
    float bal;
    float *listexp;
};
typedef struct AccountDetails AD;
```

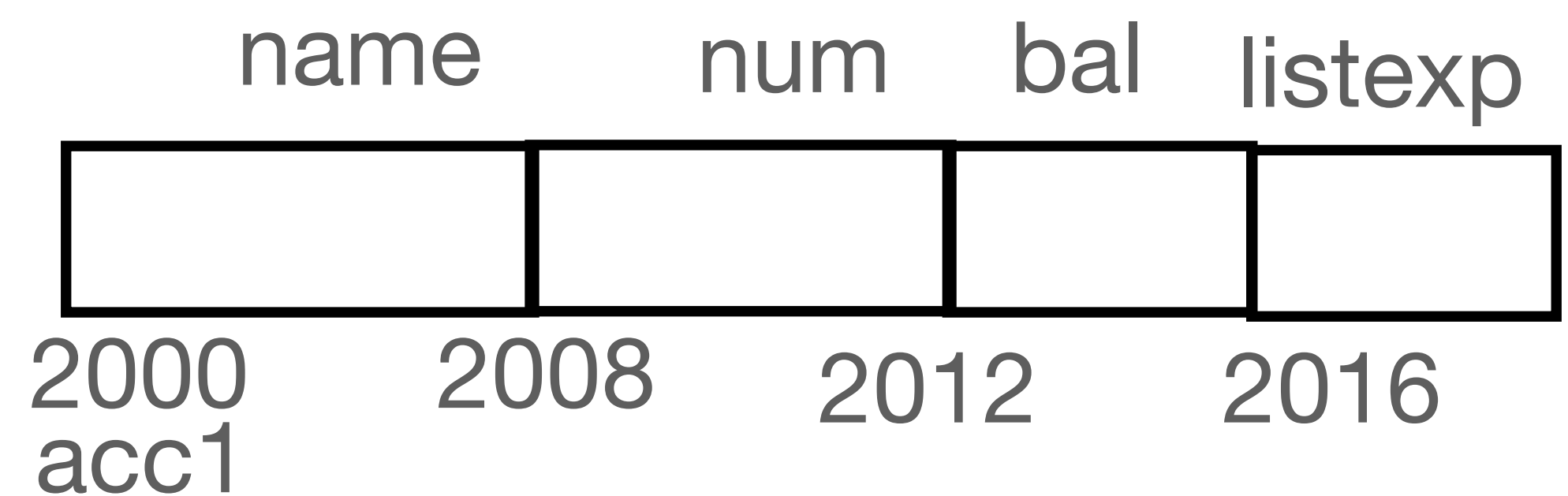
How does DMA for structure members look?



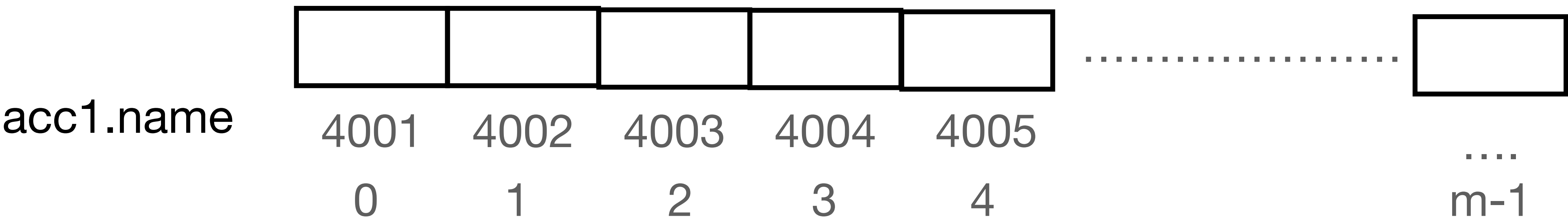
How does DMA for structure members look?



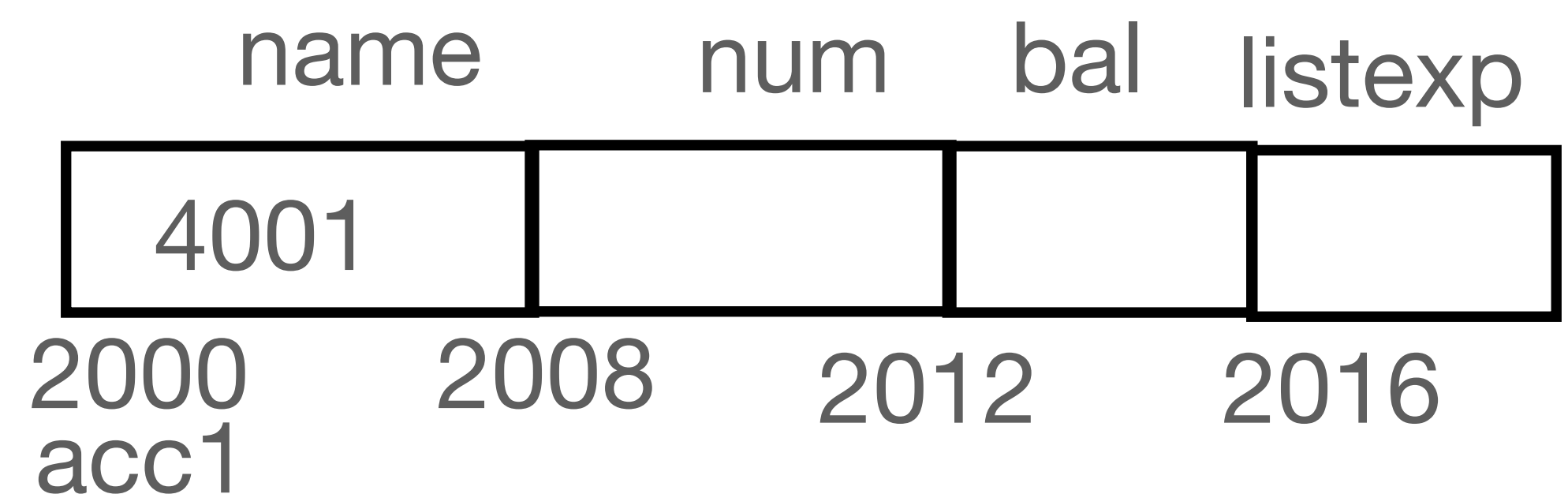
How does DMA for structure members look?



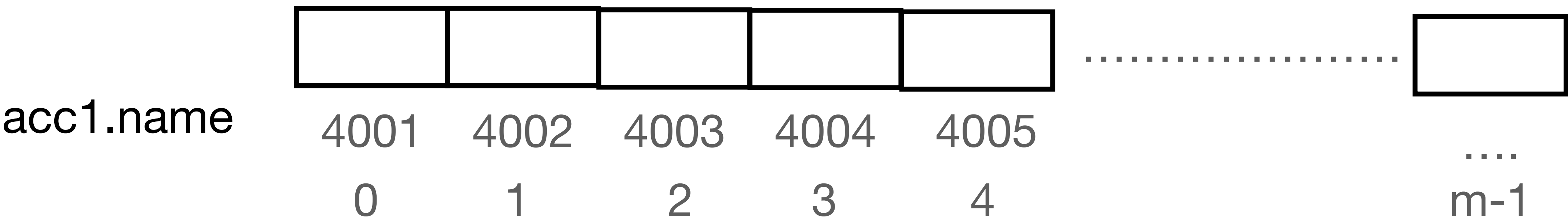
```
acc1.name = ( char *) malloc(m * sizeof(char) );
```



How does DMA for structure members look?



```
acc1.name = ( char *) malloc(m * sizeof(char) );
```



How do you do DMA for structure member?

L17_DMA.c

```
AD acc1; int m, n;

scanf("%d %d", &m, &n);

printf("%u %u %u", &acc1, acc1.name, acc1.listexp);

acc1.name = ( char *) malloc(m * sizeof(char) );

printf("%u %u %u", &acc1, acc1.name, acc1.listexp);

acc1.listexp = ( float *) malloc(n * sizeof(float) );

printf("%u %u %u", &acc1, acc1.name, acc1.listexp);

for ( ..... ) {

    //Write scanf

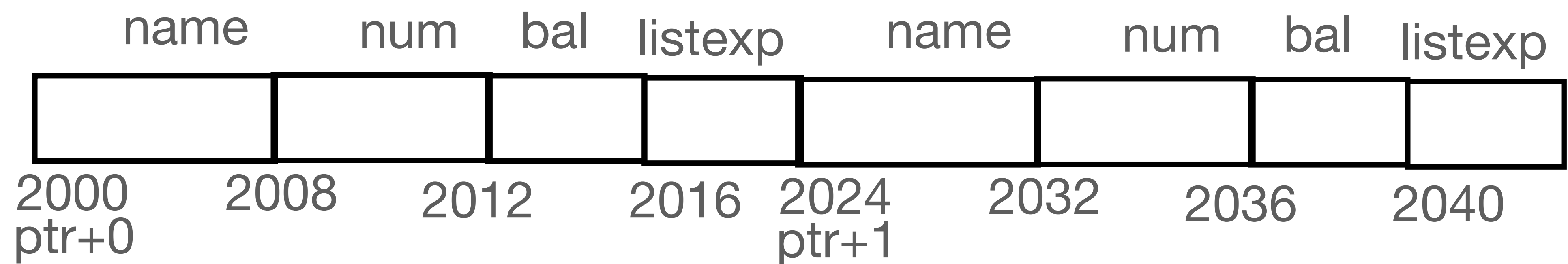
}
```

How do you do DMA for structure members?

L17_DMA.c

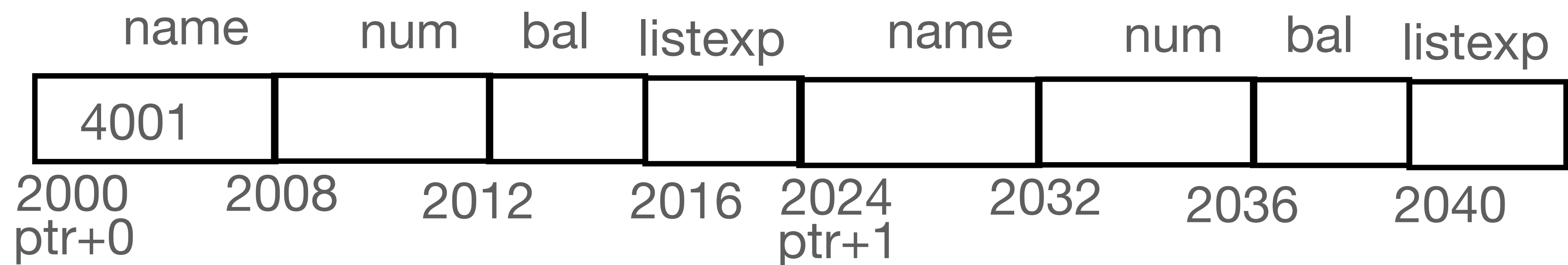
```
AD acc1; int m, n;  
  
scanf("%d %d", &m, &n);  
  
acc1.name = ( char *) malloc(m * sizeof(char) );  
acc1.listexp = ( float *) malloc(n * sizeof(float) );  
  
for ( ..... ) {  
  
    //Write scanf  
  
}
```

DMA for structure as well as its members

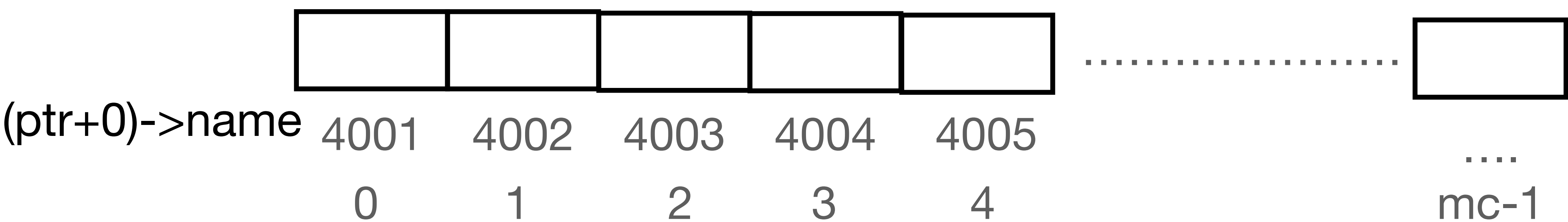


```
AD *ptr = ( AD *) malloc(m * sizeof(AD) );
```

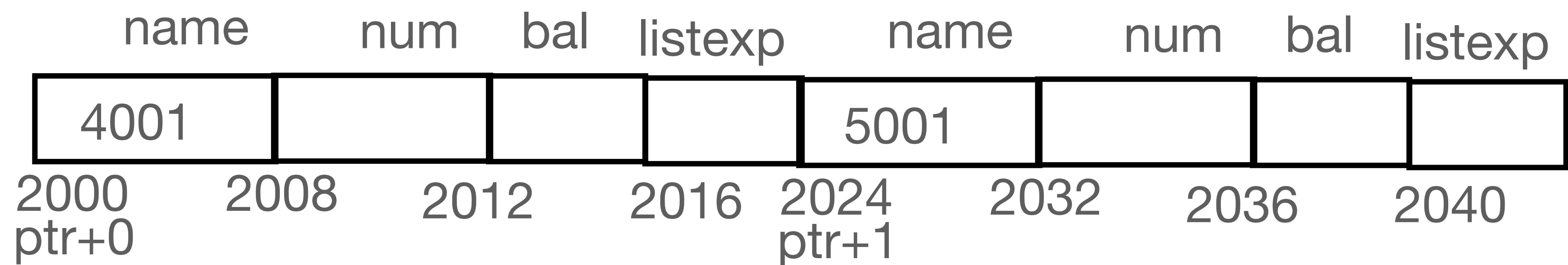
DMA for structure as well as its members



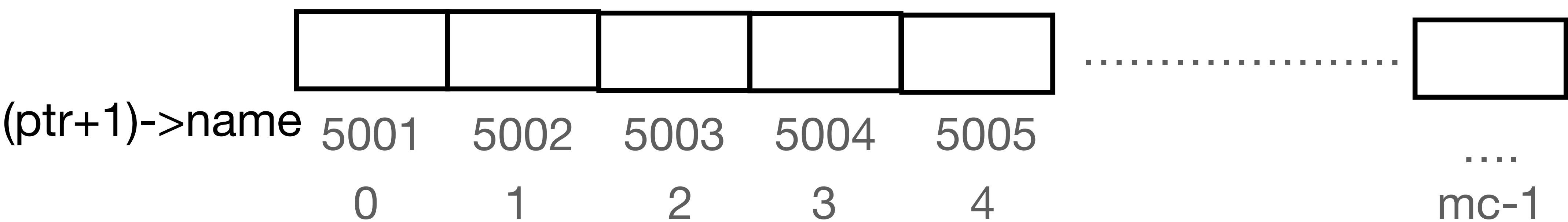
`(ptr+0)->name = (char *) malloc(mc * sizeof(char));`



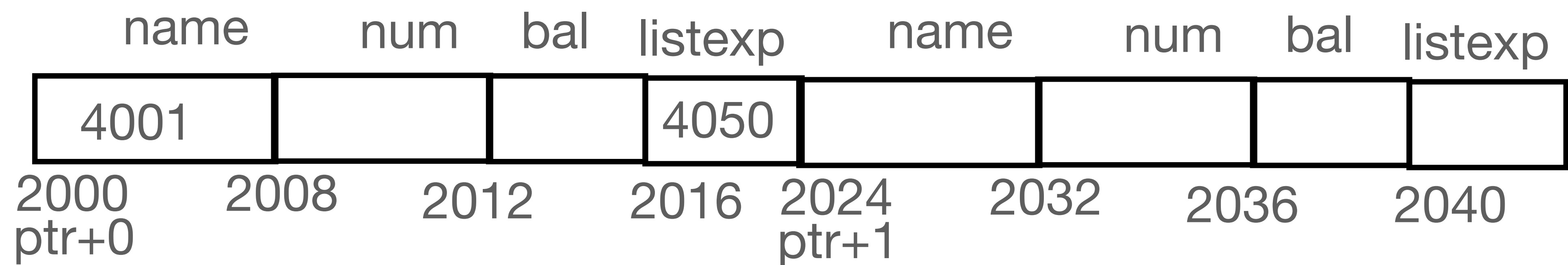
DMA for structure as well as its members



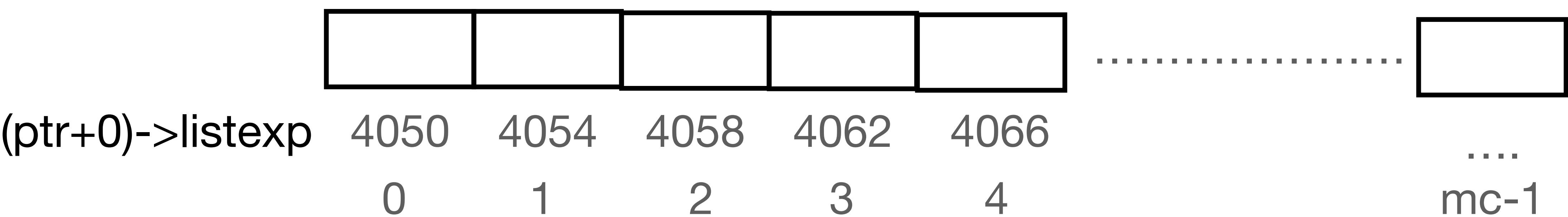
```
(ptr+1)->name = ( char *) malloc(mc * sizeof(char) );
```



DMA for structure as well as its members



```
(ptr+0)->listexp = ( float *) malloc(n * sizeof(float) );
```



DMA - key points

- allocate memory at run time
- Freeing and reallocation
- number of allocations has to be given as input

DMA - key points

- Suppose we want to add one more!
 - Freeing and reallocation
 - number of allocations has to be given as input
-
- Suppose we want to add one more location!

Number not known!

- Book railway tickets (we don't know the number!)
 - Number of people opening a bank account
 - Cancel the ticket
 - Deleting a bank account.
-
- Ideally, as we want to add / delete,

Number not known!

with addition / deletion possibilities

- Cancel the ticket
- Deleting a bank account.

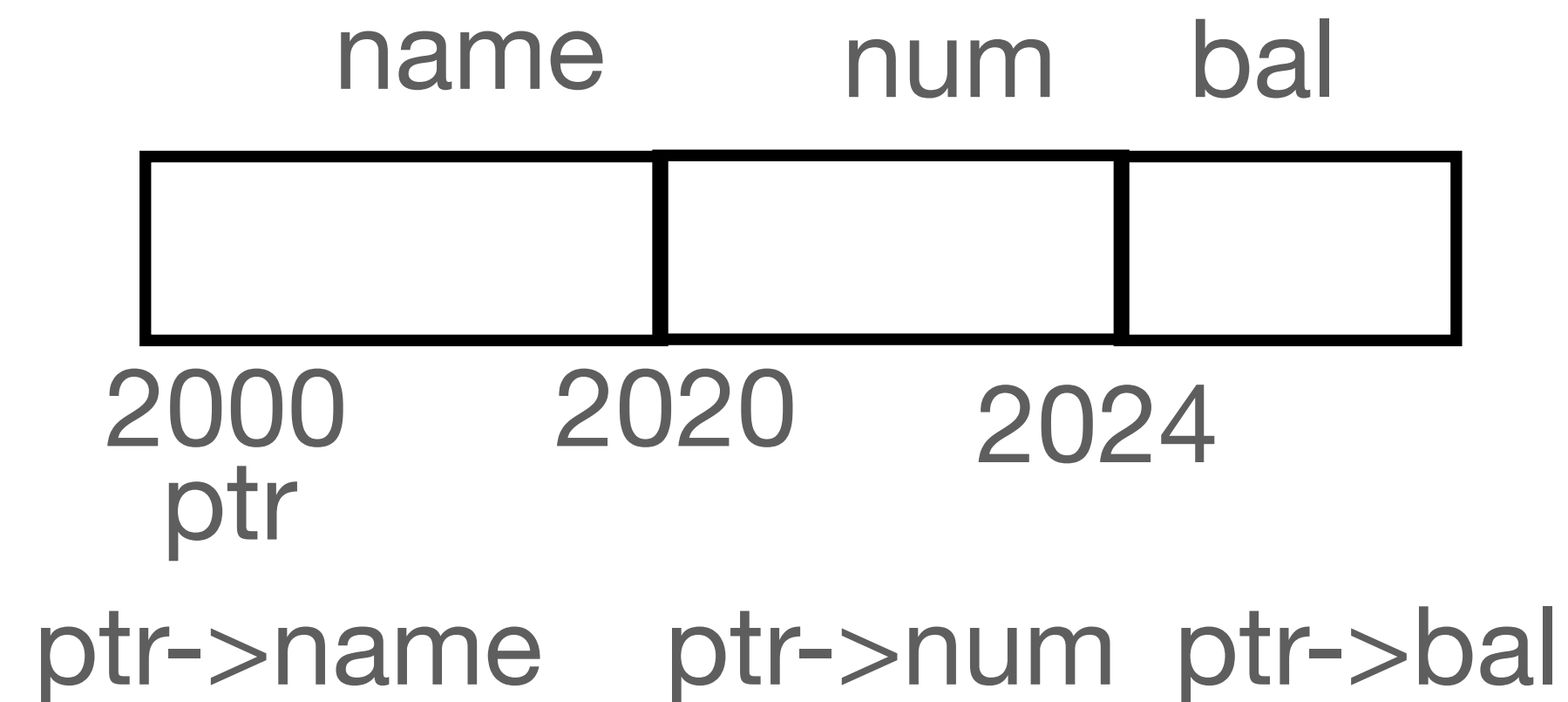
Ideally, as we want to add / delete as and when we need!

Adding one memory at a time

```
//Declaring a structure
struct AccountDetails {
    char name[20];
    int num;
    float bal;
};
typedef struct AccountDetails AD;
```

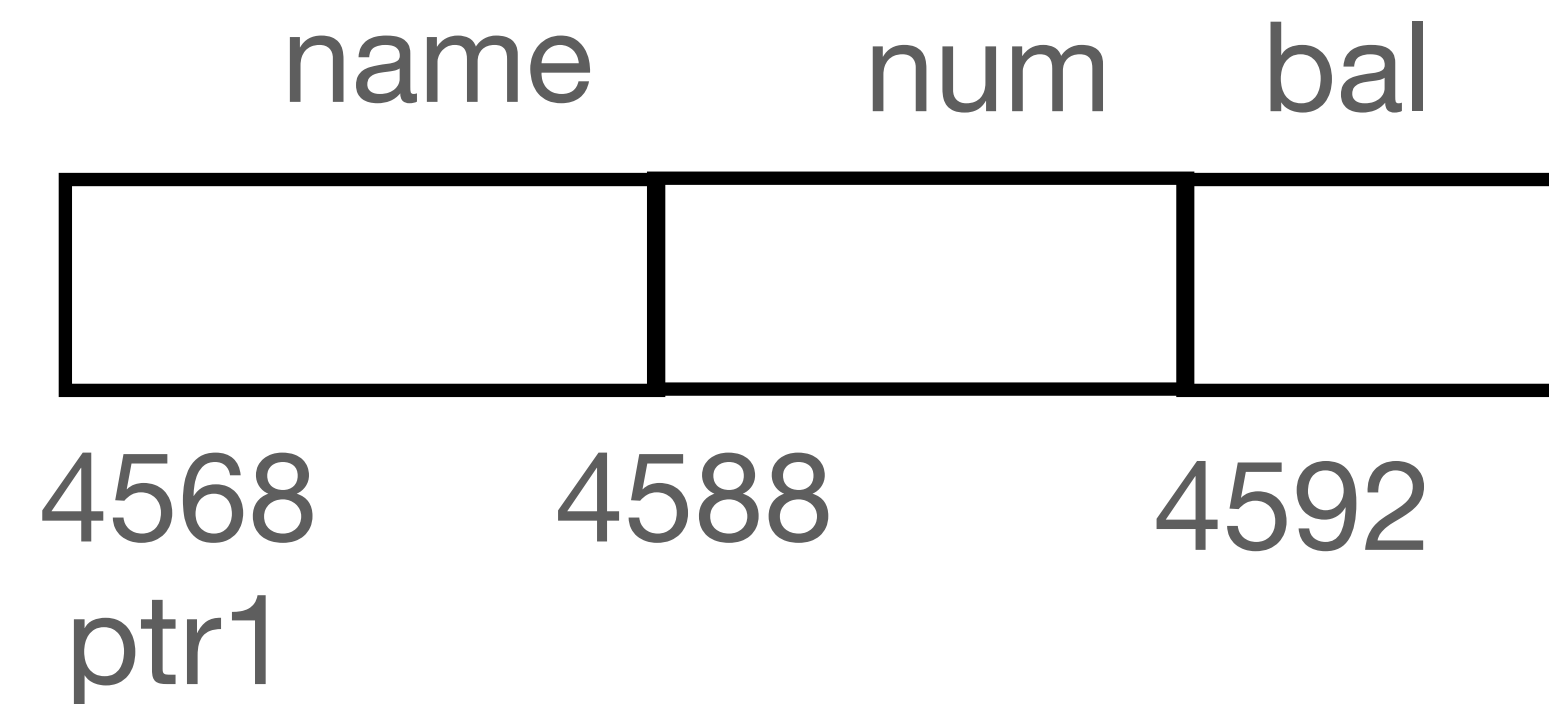
Adding a node (memory)

```
AD *ptr = ( AD *) malloc(1 * sizeof(AD) );
```



Adding another node (memory)

```
AD *ptr1 = ( AD *) malloc(1 * sizeof(AD) );
```



ptr1->num ptr1->num ptr1->bal

Adding one by one

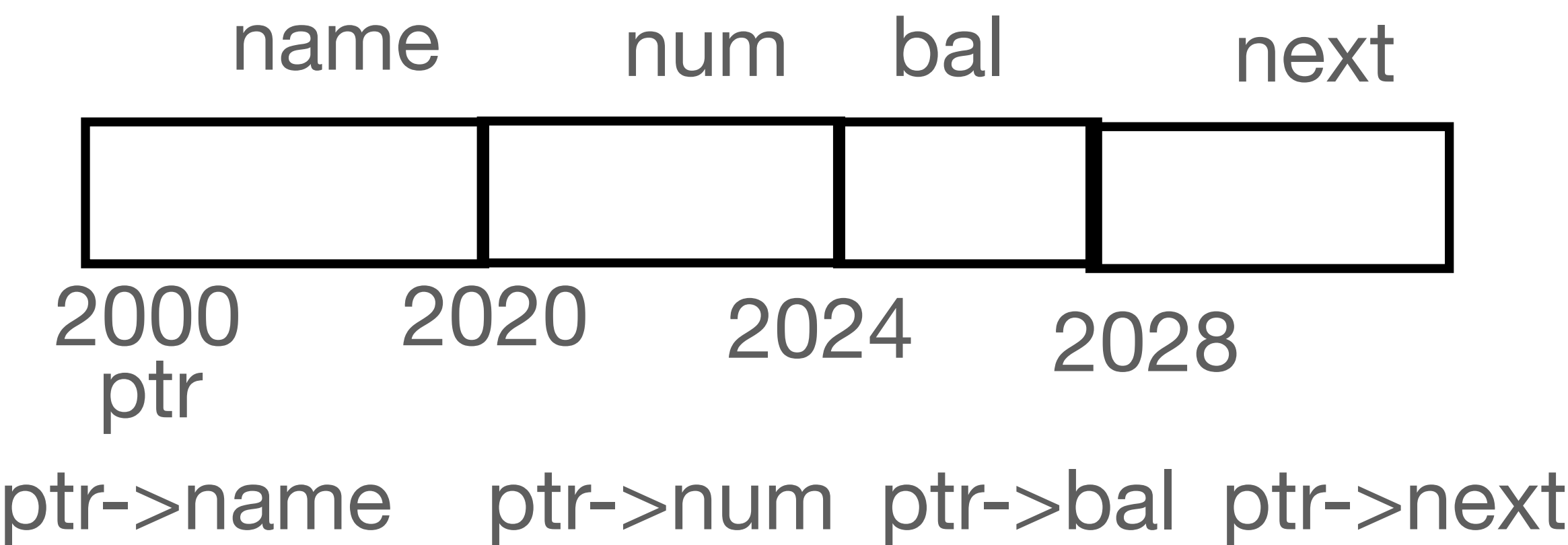
- Memory allocation is arbitrary
- $\text{ptr} = ?$
- $\text{ptr} + 1 = ?$ (not same as $\text{ptr}1$)
- No connection between ptr and $\text{ptr}1$
- Navigation from the starting is not even possible.

Self-referential structure

```
//Declaring a structure
struct AccountDetails {
    char name[20];
    int num;
    float bal;
    struct AccountDetails *next;
};
typedef struct AccountDetails AD;
```

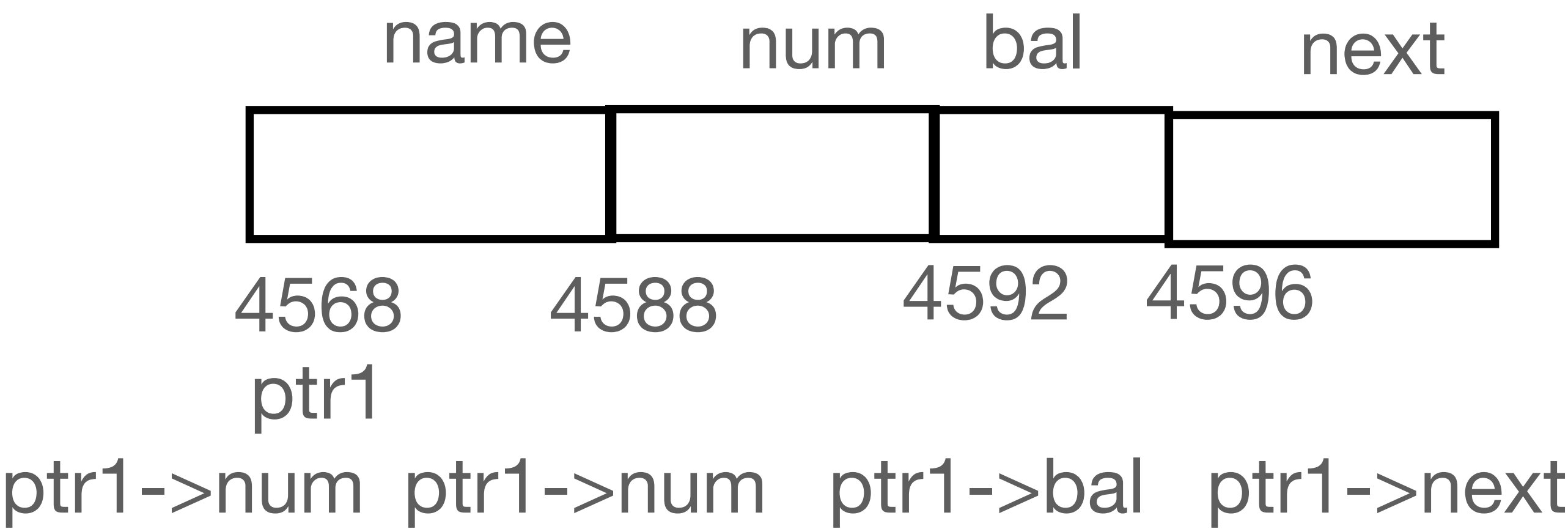
Self-referential structure details

```
AD *ptr = ( AD *) malloc(1 * sizeof(AD) );
```



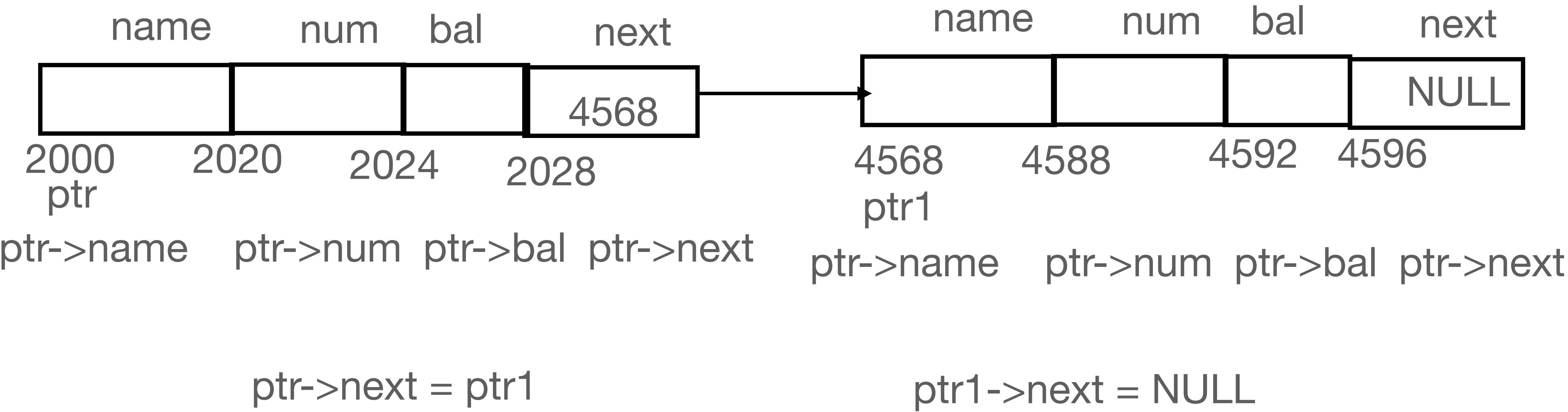
Self-referential structure details

```
AD *ptr1 = ( AD *) malloc(1 * sizeof(AD) );
```



Connecting (linking) the two nodes

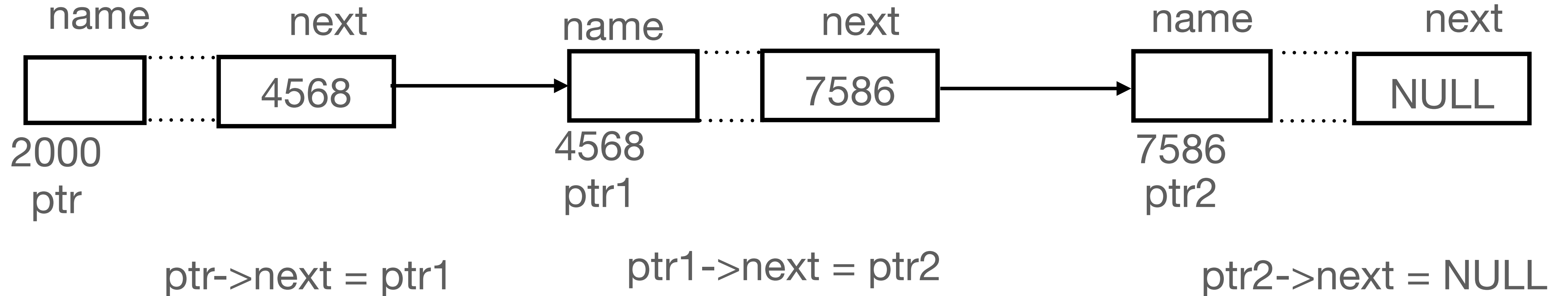
Linked List



Linking together with another node

Linked List

```
AD *ptr2 = ( AD *) malloc(1 * sizeof(AD) );
```



- It is infeasible to use ptr3, ptr4
- What is the alternative?
- HW: Try writing that.

Linked List Implementation

Start with a Headptr

```
int main(int argc, char **argv)
{
    AD *Headptr, *temp, *ptr;

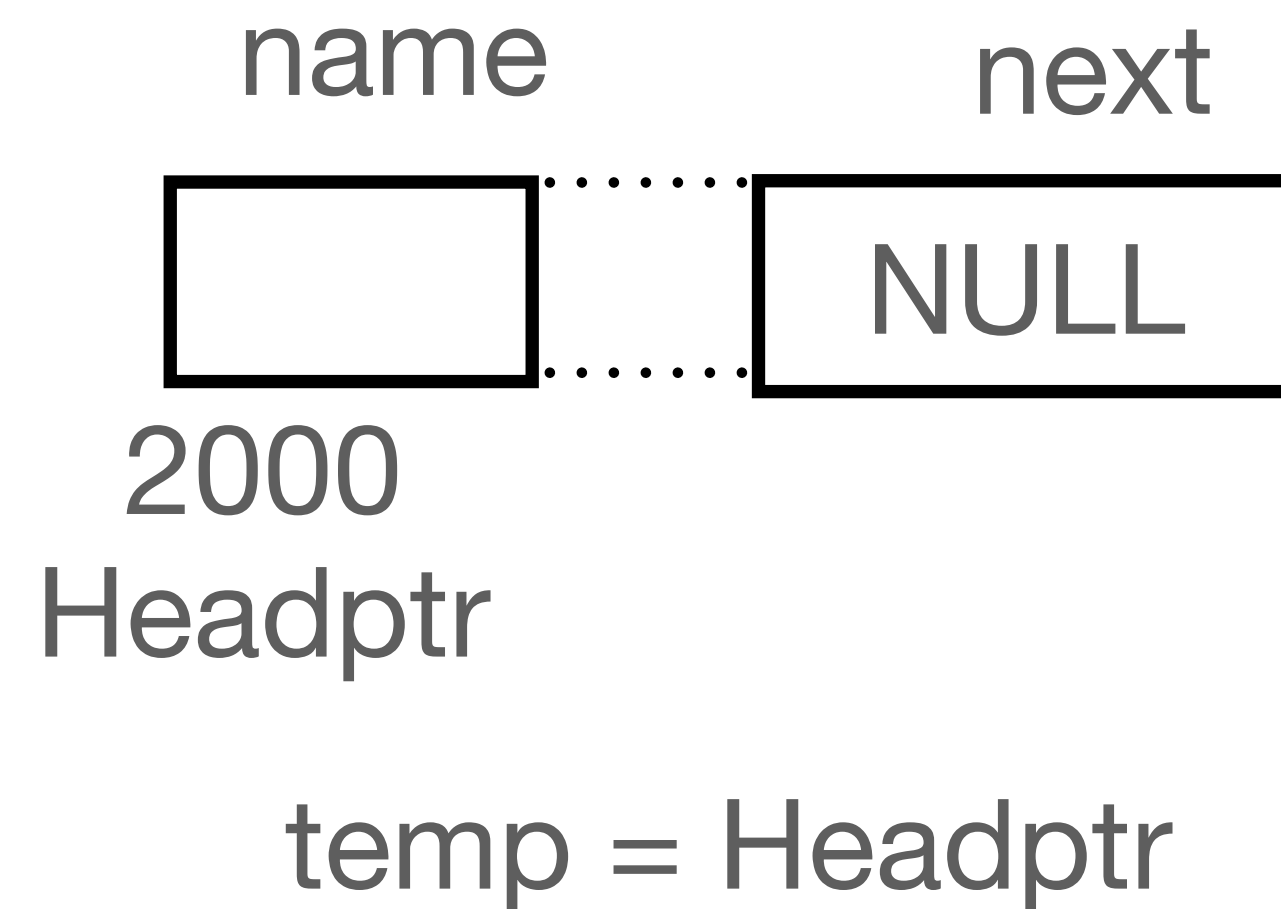
    Headptr = (AD *)malloc(sizeof(AD));
    //Inputs for Headptr
    Headptr->next = NULL;
    temp = Headptr;

}
```

Linking together with another node

Linked List

```
AD *Headptr = ( AD *) malloc(1 * sizeof(AD) );
```



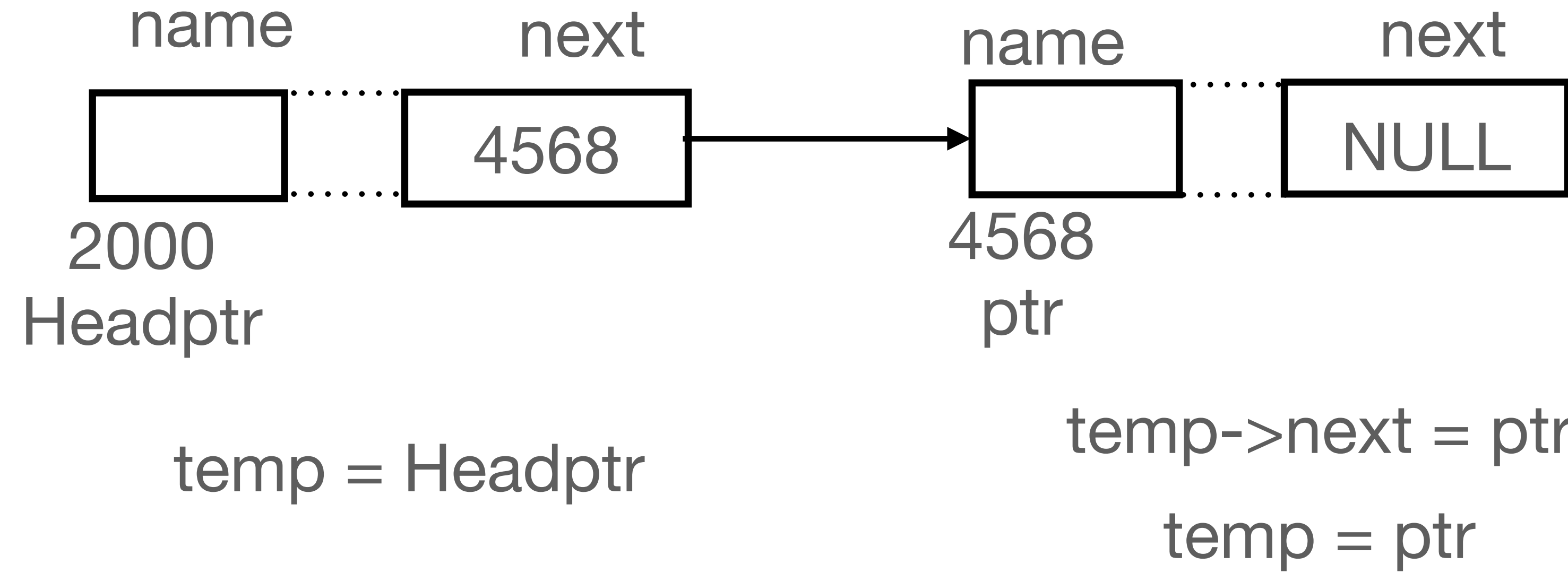
Taking one node (mem. block) at a time

```
while (ans == 'y') {  
    ptr = (AD *)malloc(sizeof(AD));  
    //Inputs for ptr  
    ptr->next = NULL;  
    temp->next = ptr;  
    temp = ptr;  
    scanf("%c", &dummy);  
    printf("Enter y to continue adding a node : ");  
    scanf("%c", &ans);  
    scanf("%c", &dummy);  
}  
printdetails(Headptr);
```

Linking together with another node

Linked List

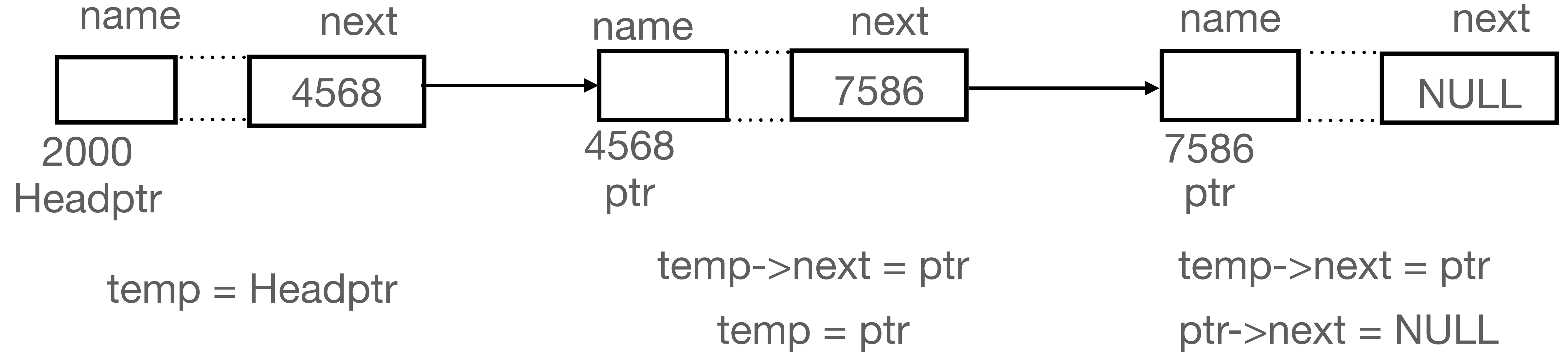
```
AD *ptr = ( AD *) malloc(1 * sizeof(AD) );
```



Linking together with another node

Linked List

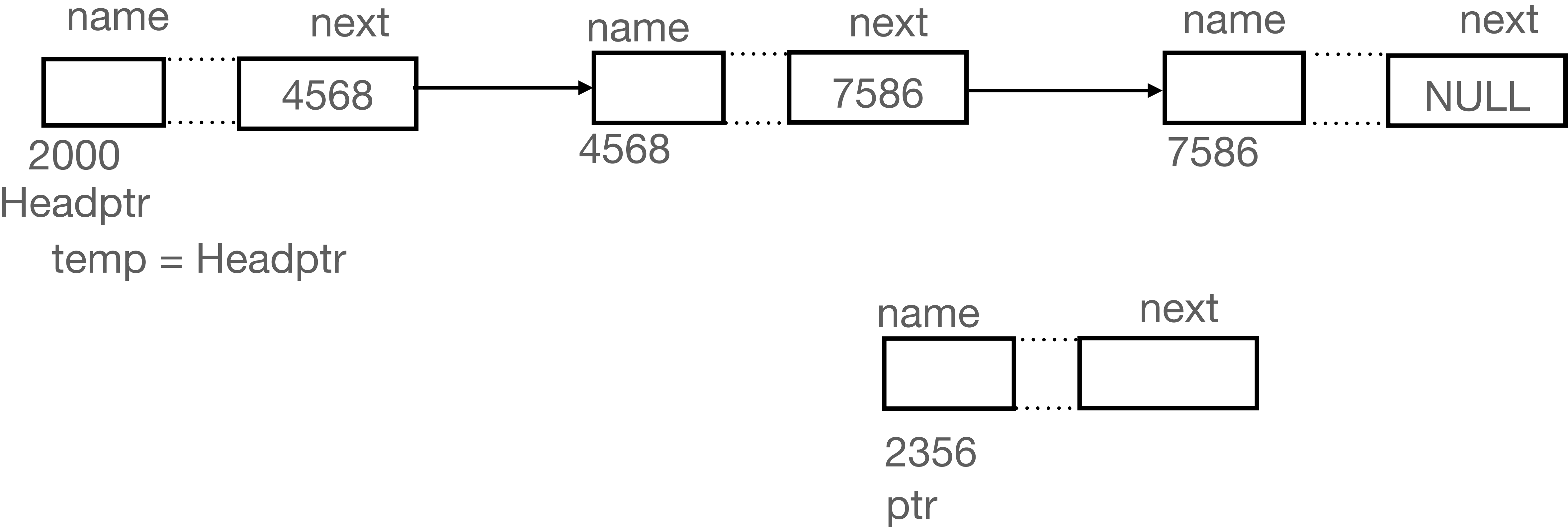
```
AD *ptr = ( AD *) malloc(1 * sizeof(AD) );
```



Inserting a node inbetween

Linked List

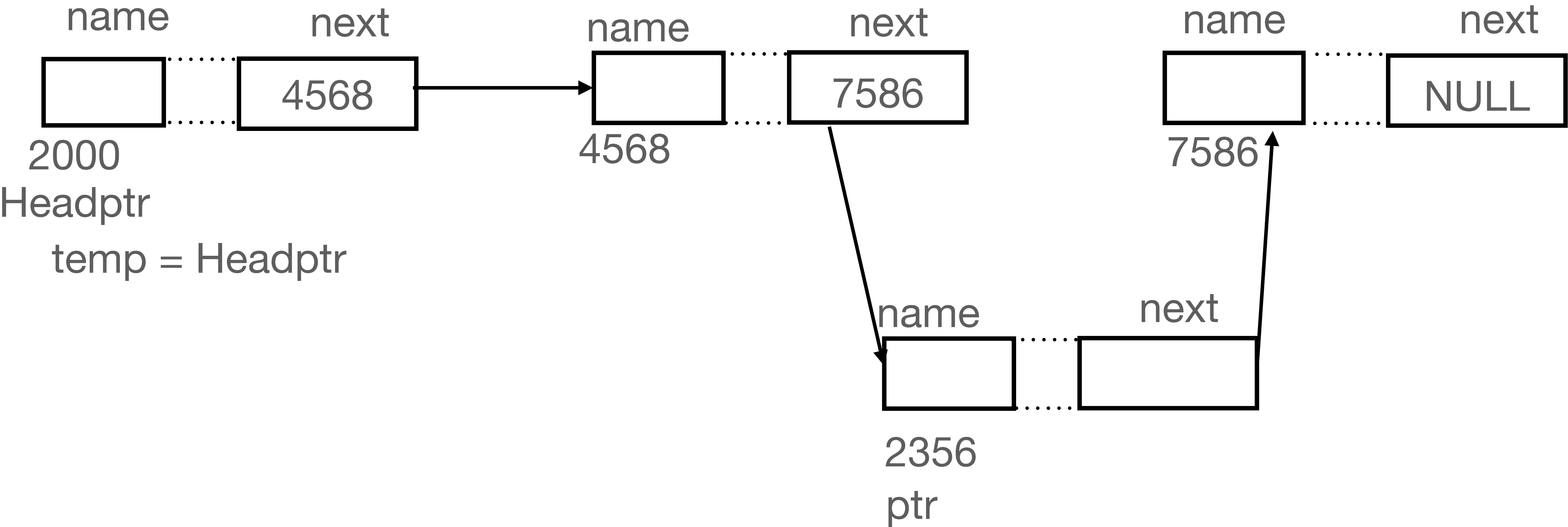
```
AD *ptr = ( AD *) malloc(1 * sizeof(AD) );
```



Inserting a node inbetween

Linked List

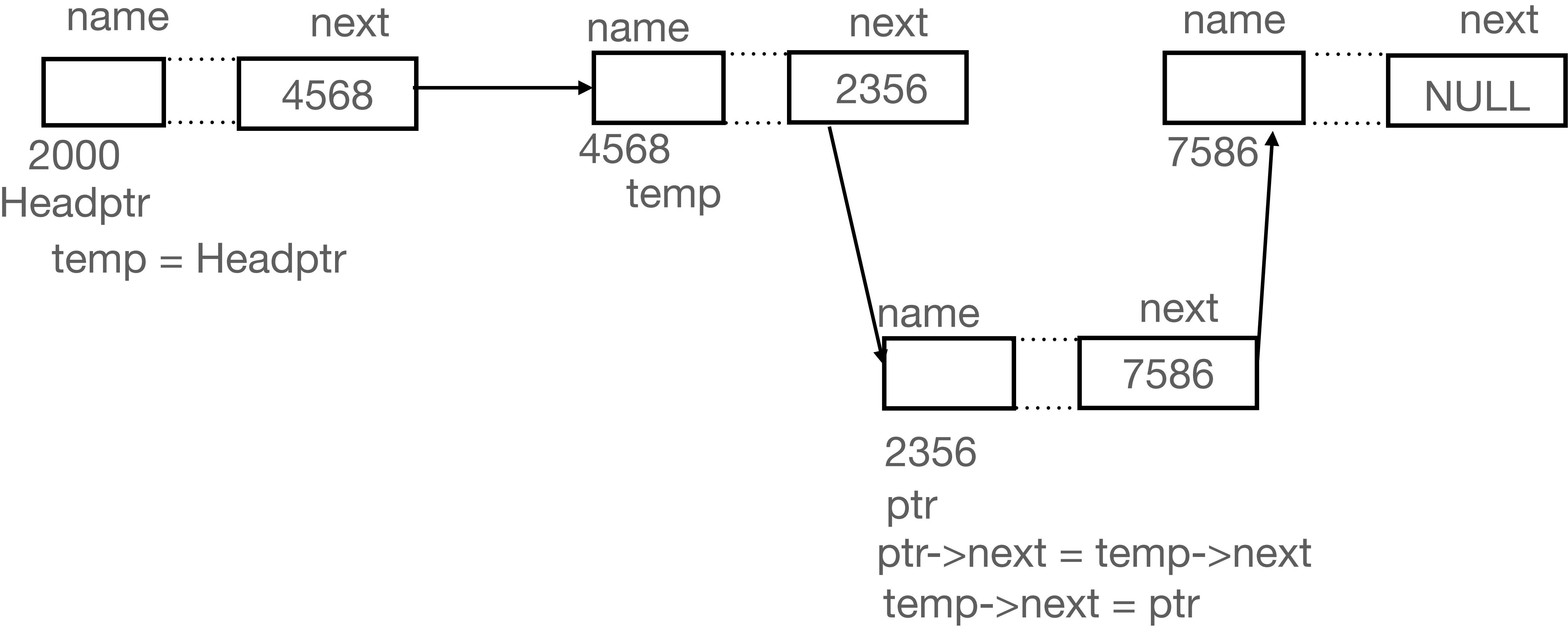
```
AD *ptr = ( AD *) malloc(1 * sizeof(AD) );
```



Inserting a node inbetween

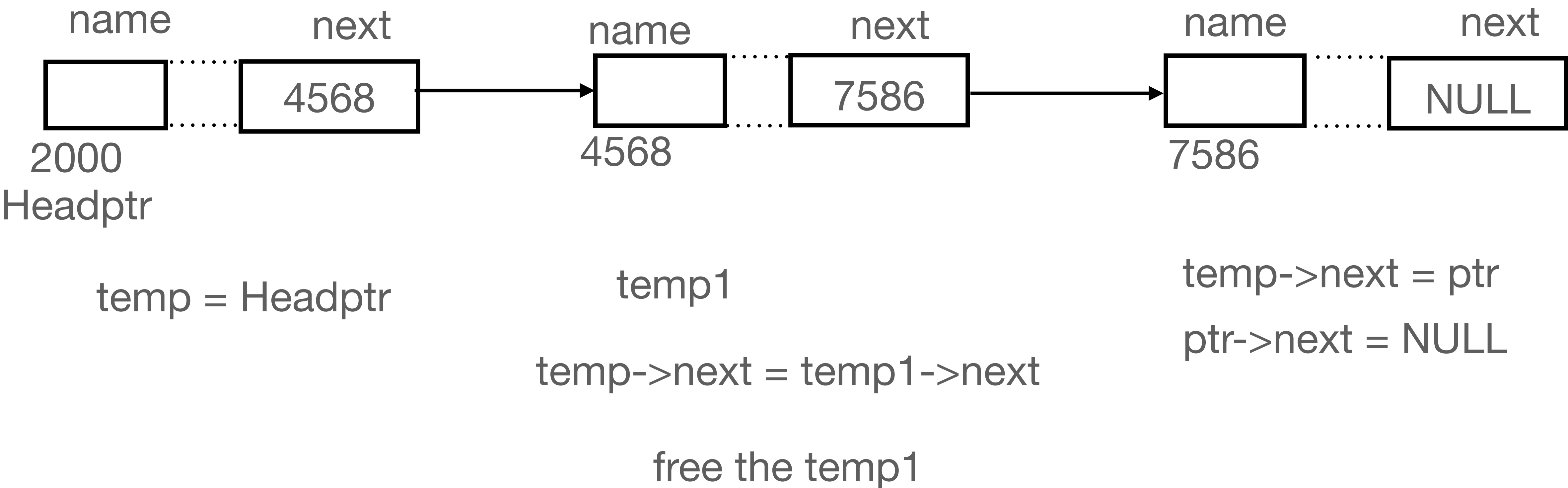
Linked List

```
AD *ptr = ( AD *) malloc(1 * sizeof(AD) );
```



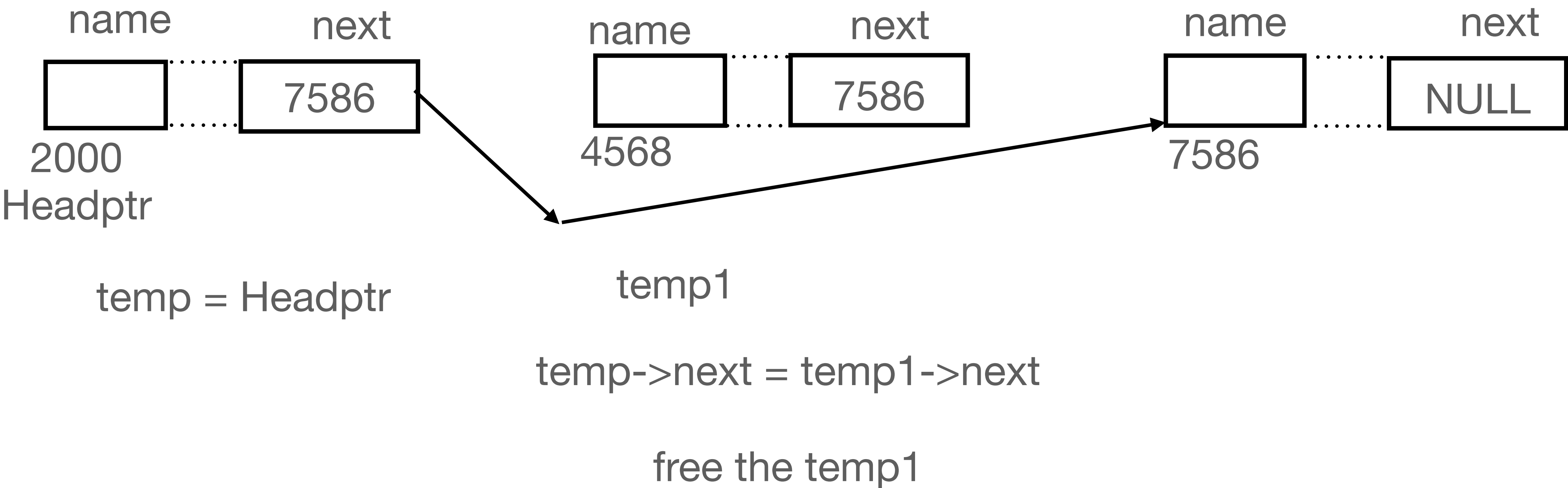
Deleting a node

Linked List



Deleting a node

Linked List



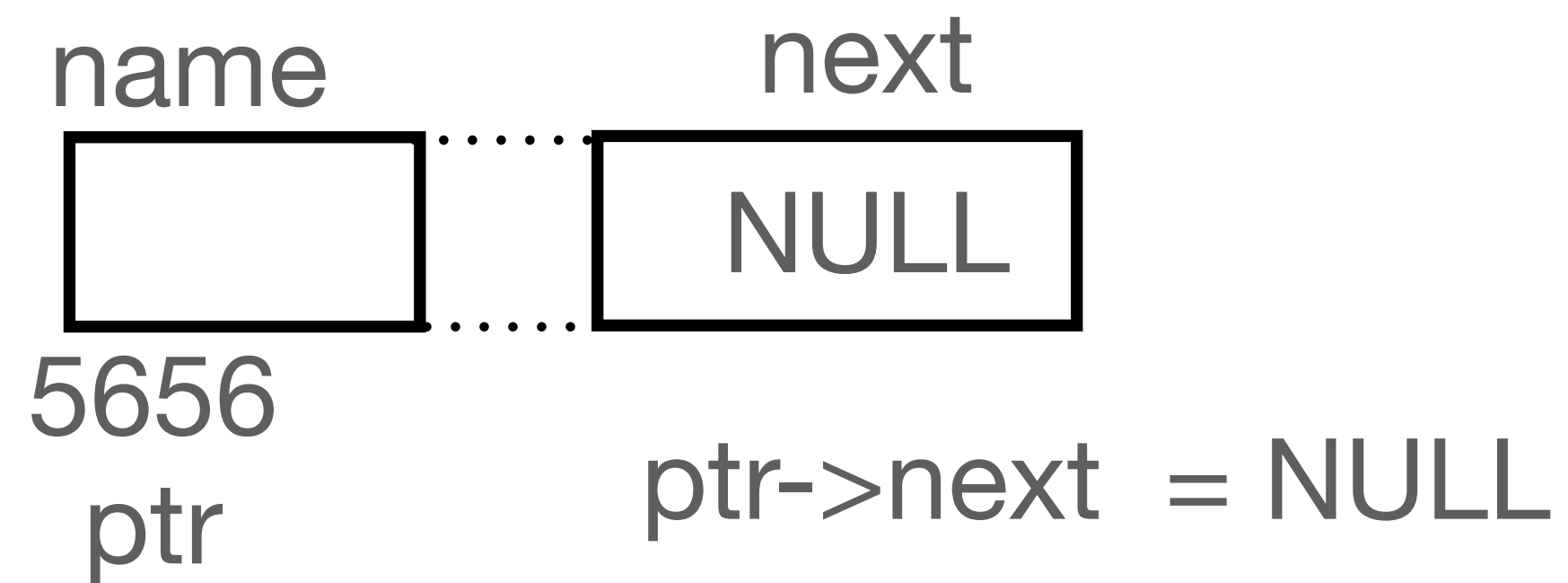
UDF's in a singly linked list

- AddingANodeAtEnd
- AddingANodeAtBegin
- InsertANode
- DeleteANode
- DeleteANodeAtEnd
- DeleteANodeAtBegin

Add at Begin

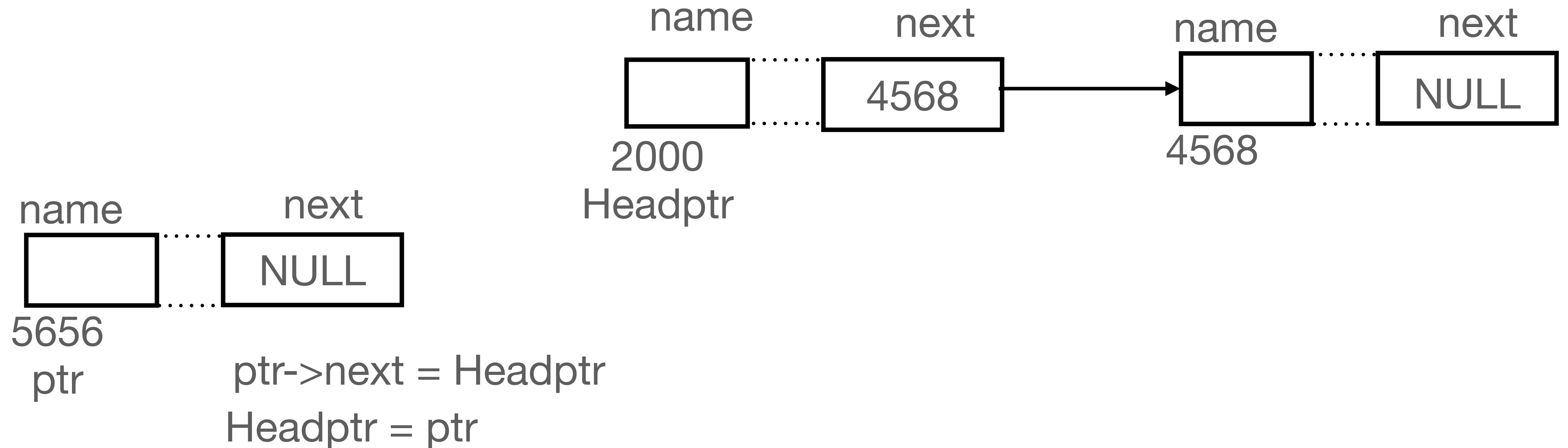
Linked List

```
AD *ptr = ( AD *) malloc(1 * sizeof(AD) );
```



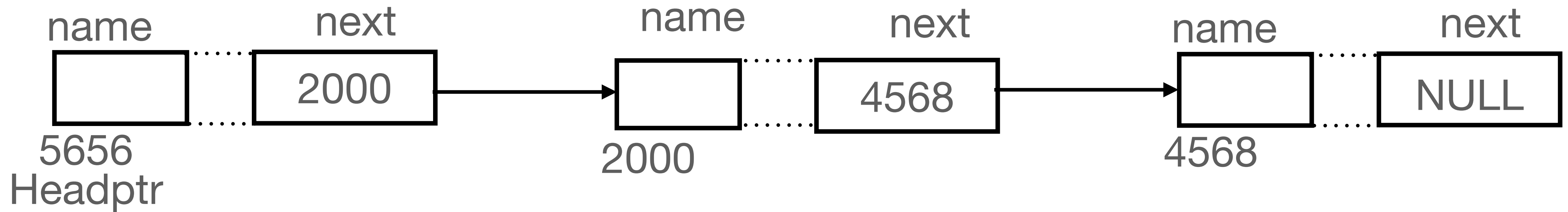
Add at Begin

Linked List



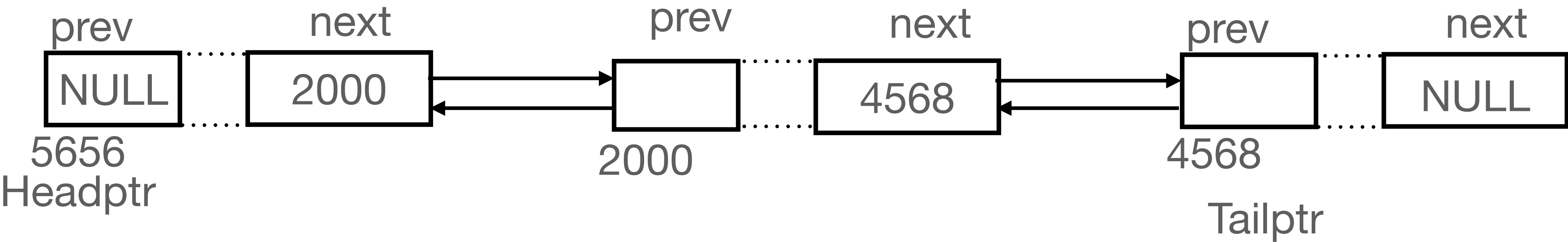
Add at Begin

Linked List



Doubly Link List

Linked List



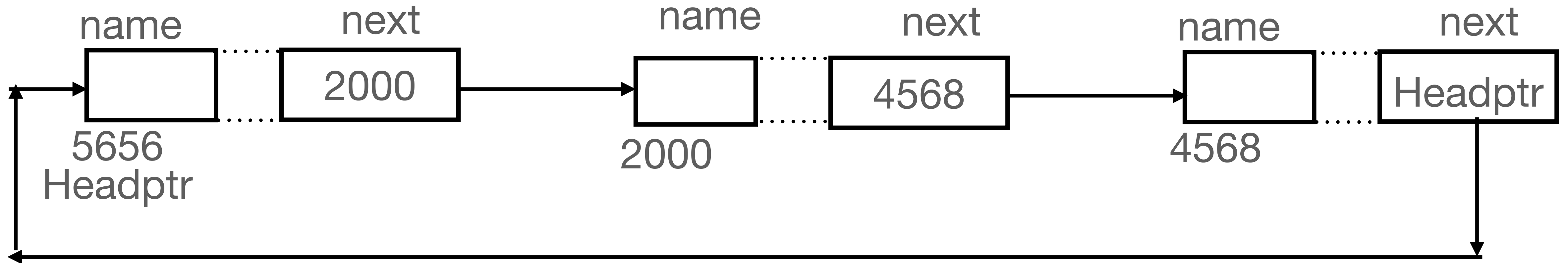
Headptr -> next

Tailptr -> prev

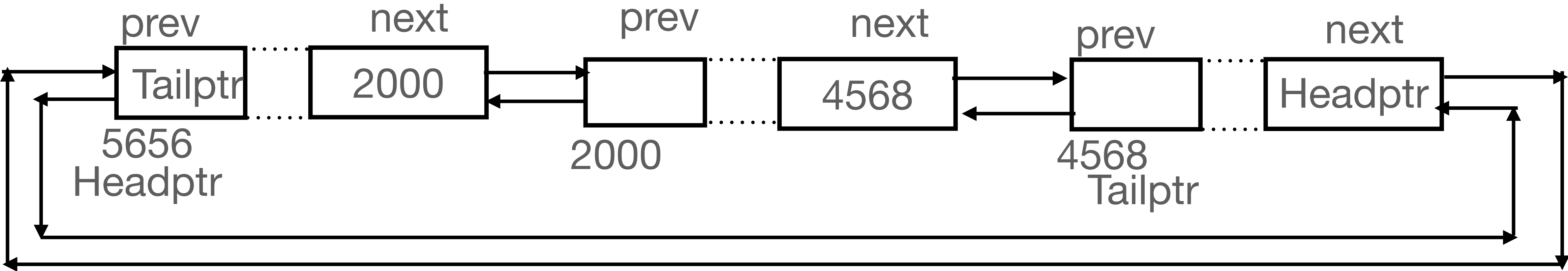
Structure for a doubly link list

```
//Declaring a structure
struct AccountDetails {
    struct AccountDetails *prev;
    char name[20];
    int num;
    float bal;
    struct AccountDetails *next;
};
typedef struct AccountDetails AD;
```

Singly circular Linked List

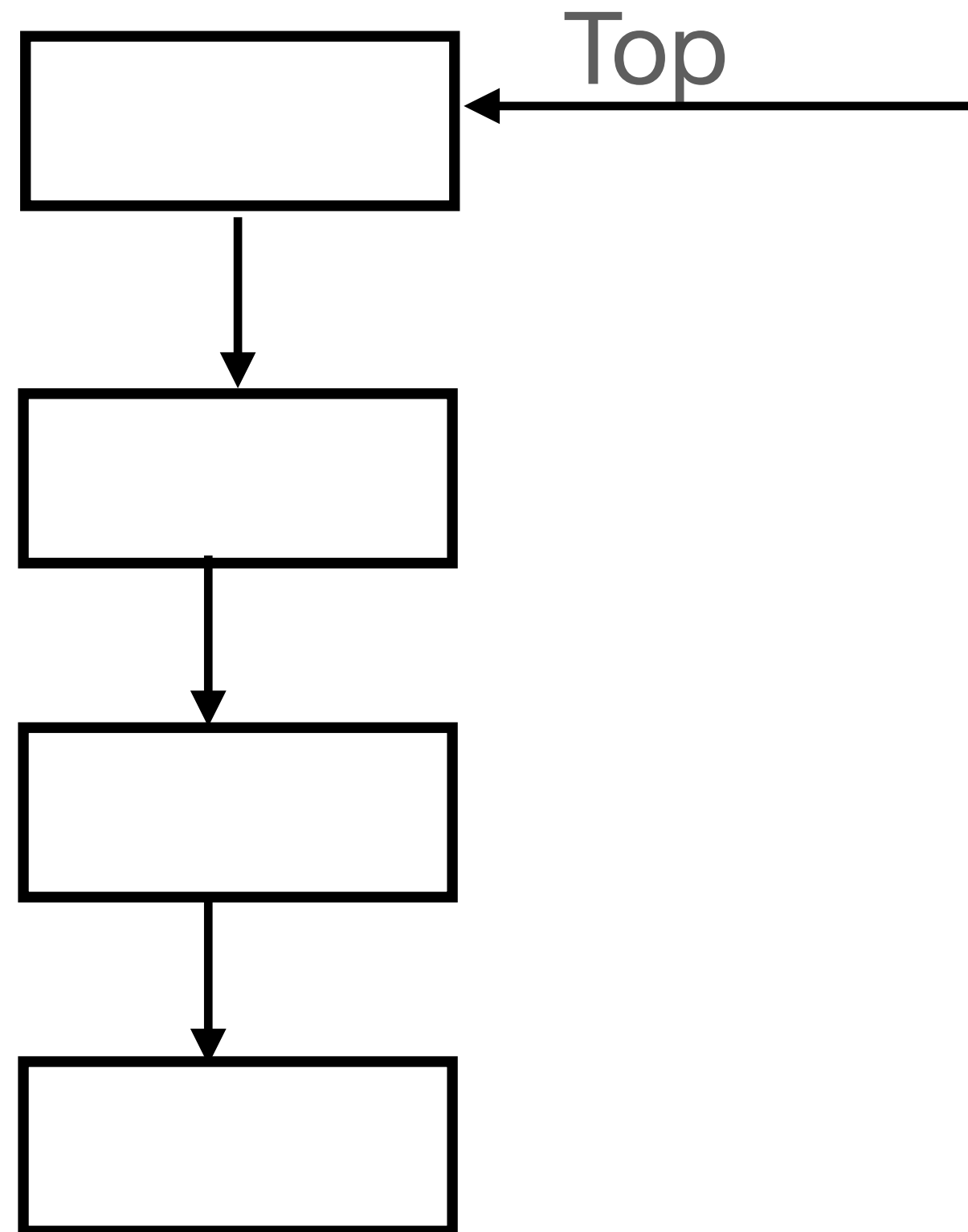


Doubly Circular Linked List



Stack (Last In First out, LIFO) - Rice bags

Variation in Linked List



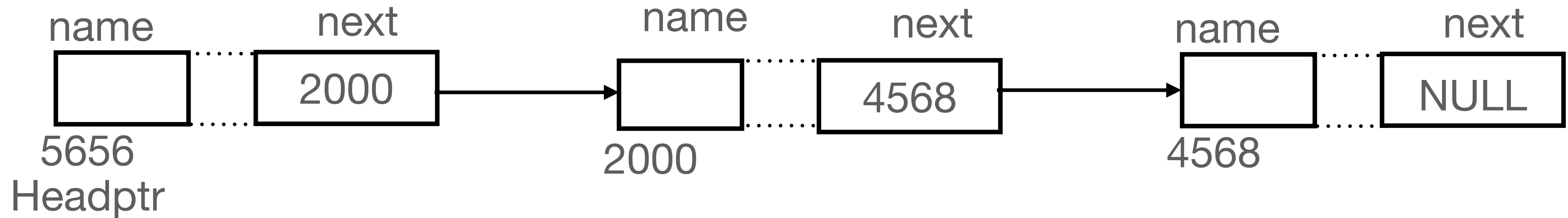
Only Two functions are need

1) Push (Add at the top)

2) Pop (Delete at the top)

Queue (First In First out, FIFO) - Typical queue

Also a linked List



Only Two functions are need

- 1) Add at the end
- 2) Delete at begin