

ED1021 - Introduction to computation and visualisation

L16 - Dynamic Memory Allocation

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

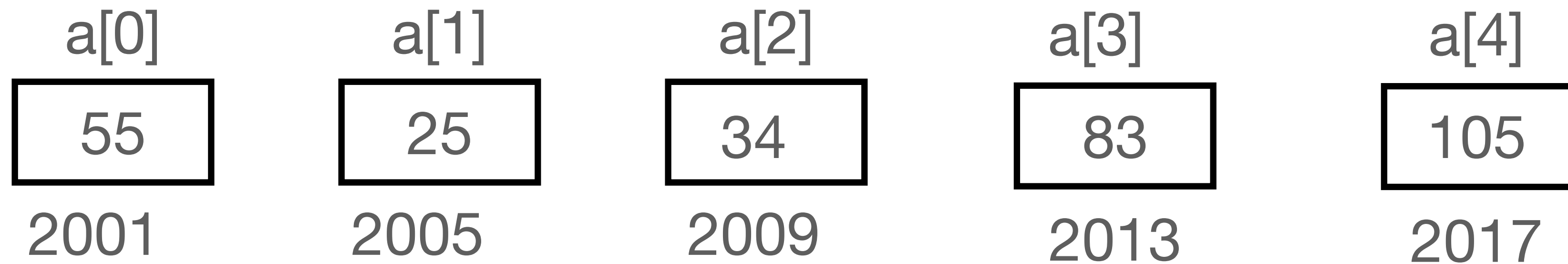
Course web page: <https://ed.iitm.ac.in/~raman/introcomp.html>

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

Array variable

integer array

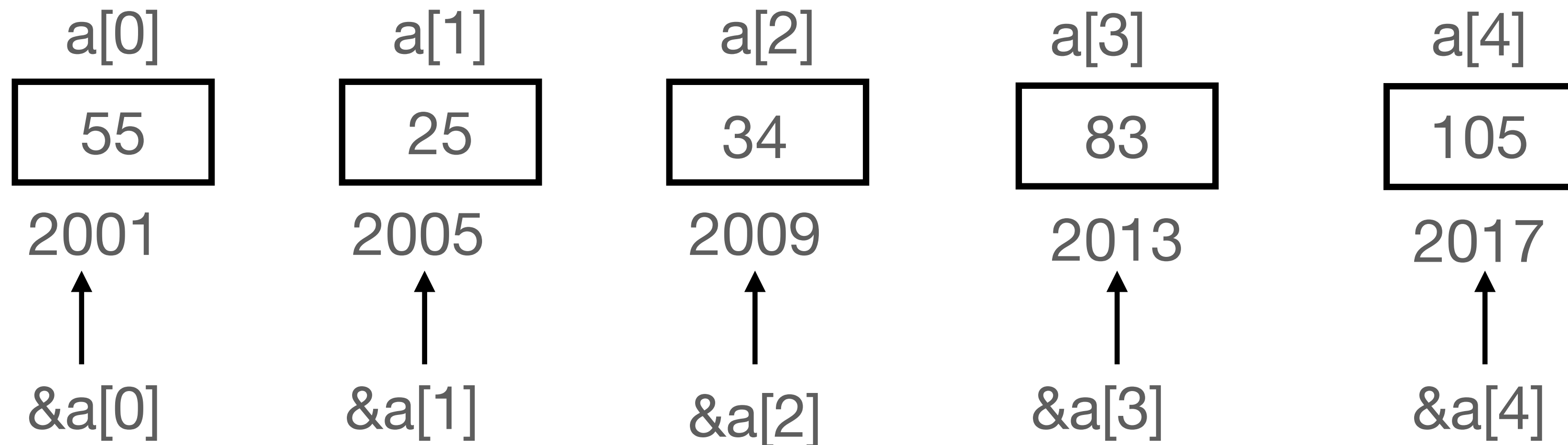
- `int a[5];`



Fetching the address of each array element

integer array

- `int a[5];`



Array variable

Better representation

- `int a[5];`

a[0]	a[1]	a[2]	a[3]	a[4]
55	25	34	83	105
2001	2005	2009	2013	2017

Array elements are always arranged in a contiguous manner i.e. the memory location or address will be having constant interval from one to the other.

Interval determined by sizeof(int)

sizeof(datatype)

- `int a[5];`

a[0]	a[1]	a[2]	a[3]	a[4]
55	25	34	83	105
2001	2005	2009	2013	2017

Suppose I want to change this array size

I can't allocate this number run time

- `int a[10];`

a[0]	a[1]	a[2]	a[3]	a[4]		a[9]
55	25	34	83	105		1005
2001	2005	2009	2013	2017		2037

Suppose I want to change this array size

I can't allocate this number run time

- `#define N 5`
- `int a[N];`

a[0]	a[1]	a[2]	a[3]	a[4]
55	25	34	83	105
2001	2005	2009	2013	2017

Suppose I want to change this array size

I can't allocate this number run time

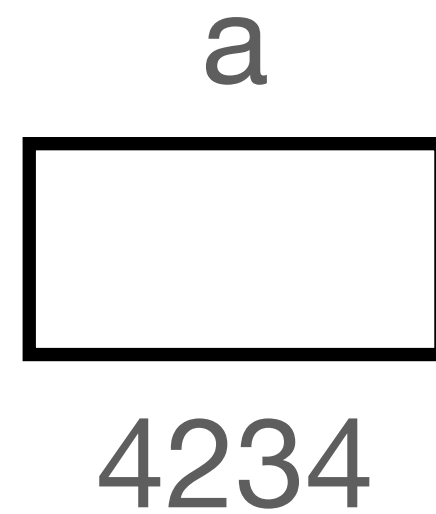
- `#define N 10`
- `int a[N];`



Allocation of array size during running

Dynamic memory allocation (#include <malloc.h>)

- `int *a;`



Allocation of array size during running

Dynamic memory allocation (#include <malloc.h> or <stdlib.h>)

- `int *a;`



- `int *a, b = 5;`

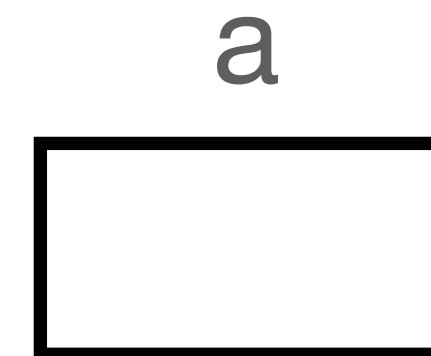
- `a = &b;`

- `b` has a mem. loc.

Allocating one memory location

Dynamic memory allocation using sizeof(datatype) and malloc function

- `int *a;`
- `a = () malloc(1 * sizeof(int));`



Allocating one memory location

malloc function returns (void *) and needs typecasting

- `int *a;`
- `a = (int *) malloc(1 * sizeof(int));`

a

2001

- How do you put value at the address?

Allocating one memory location

malloc function returns (void *) and needs typecasting

- `int *a;`
- `a = (int *) malloc(1 * sizeof(int));`

a

2001

- `*a = 10;`
- `printf("%d", *a);`

- `scanf("%d", a);`
- `printf("%d", *a);`

Allocating one memory location

malloc function returns (void *) and needs typecasting

- `int *a;`
- `a = (int *) malloc(1 * sizeof(int));`

- (remember `*a = *(a+0) = a[0]`)

- Effectively, malloc statement has created an array of size 1 and hence a is now base address.

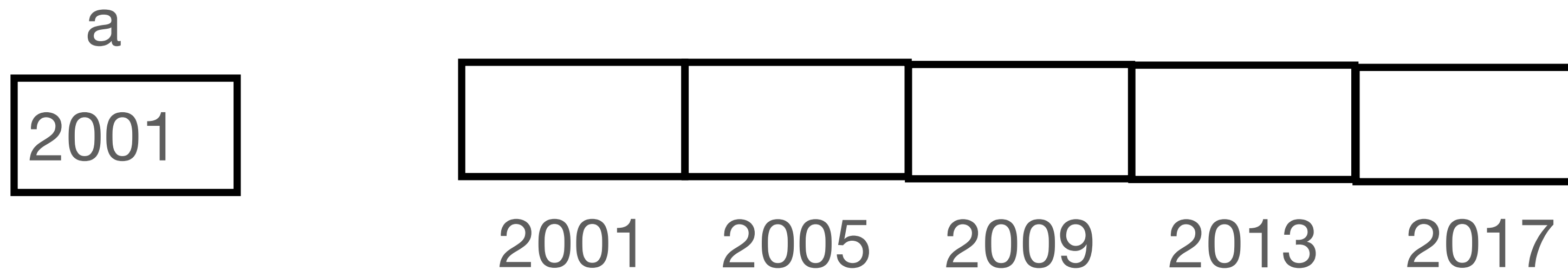
a
2001
4234

a[0]
10
2001

Allocating more memory locations

malloc and typecasting

- `int *a;`
- `a = (int *) malloc(5 * sizeof(int));`

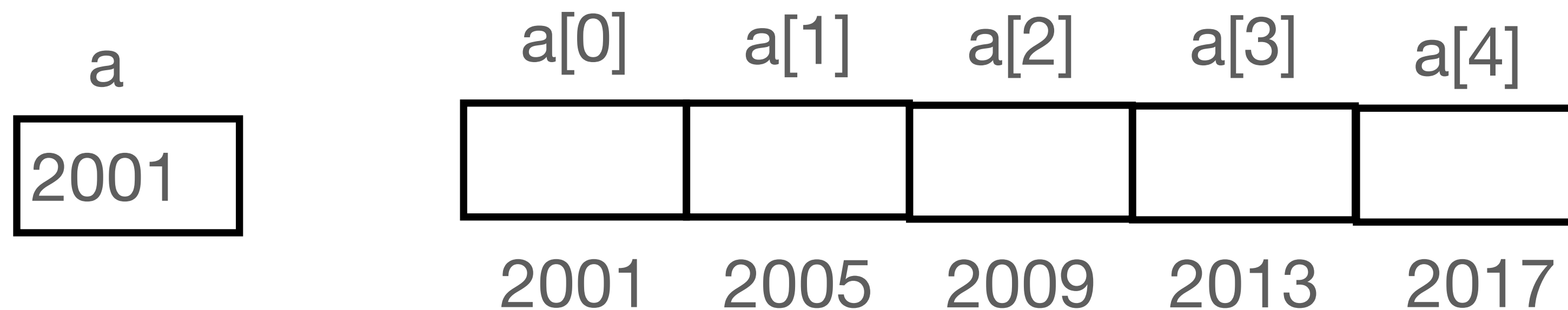


- 5 locations, a is now base address and they are also contiguous.

Allocating more memory locations

malloc and typecasting

- `int *a;`
- `a = (int *) malloc(5 * sizeof(int));`



- Effectively, malloc statement has created an array of size 5, `a` is now base address and are also contiguous.

Actually, no array notation anywhere!

Array indexing will still work

```
int *a;
```

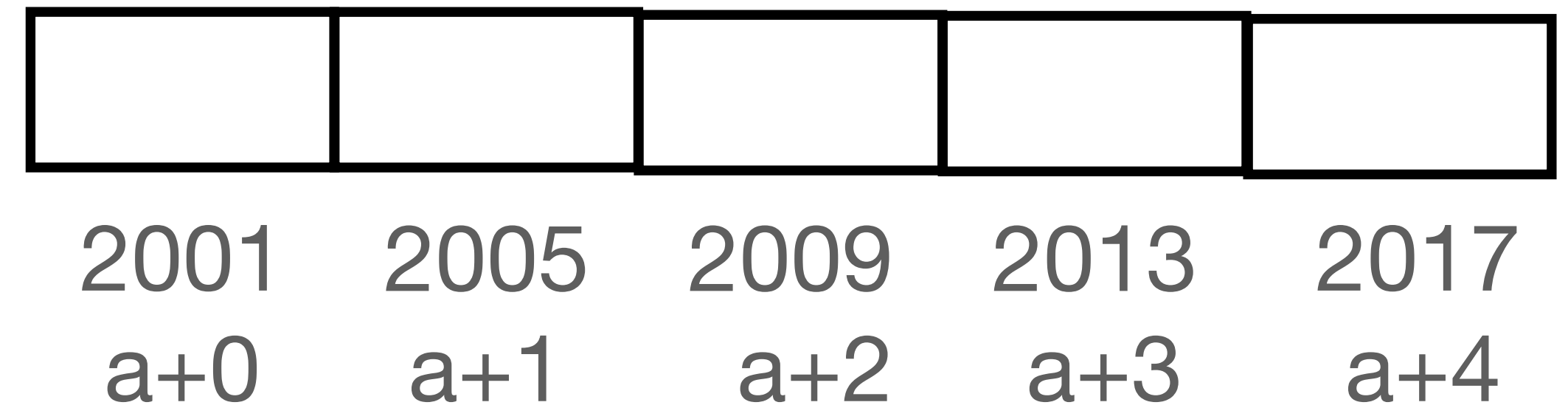
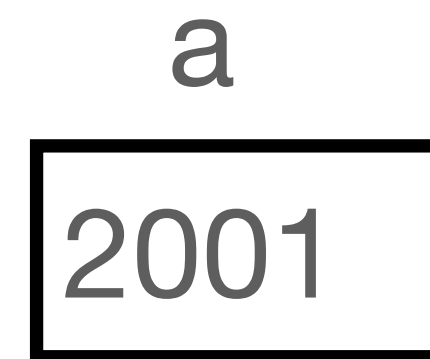
```
a = ( int *) malloc(5 * sizeof(int) );
```

```
for ( ..... ) {
```

```
    scanf("%d", a + i);
```

```
    printf("%d", *(a+i));
```

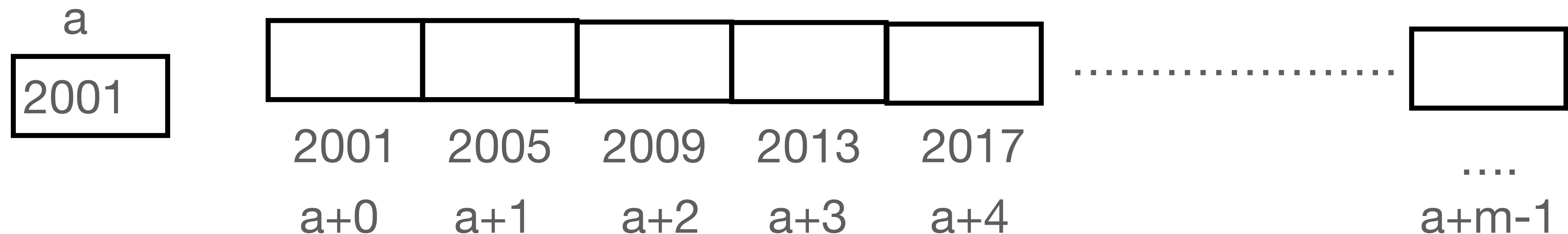
```
}
```



Allocating more memory locations

make it as a variable, which can be taken as input at run time

```
int *a, m;  
  
scanf("%d", &m);  
  
a = ( int *) malloc(m * sizeof(int) );  
  
for ( ..... ) {  
    scanf("%d", a + i);  
    printf("%d", *(a+i));  
}
```



CW for today

CW: Do the dynamic memory allocation for a float pointer.

Similar allocation for float pointer

sizeof(float)

```
float *a;
```

```
int m;
```

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

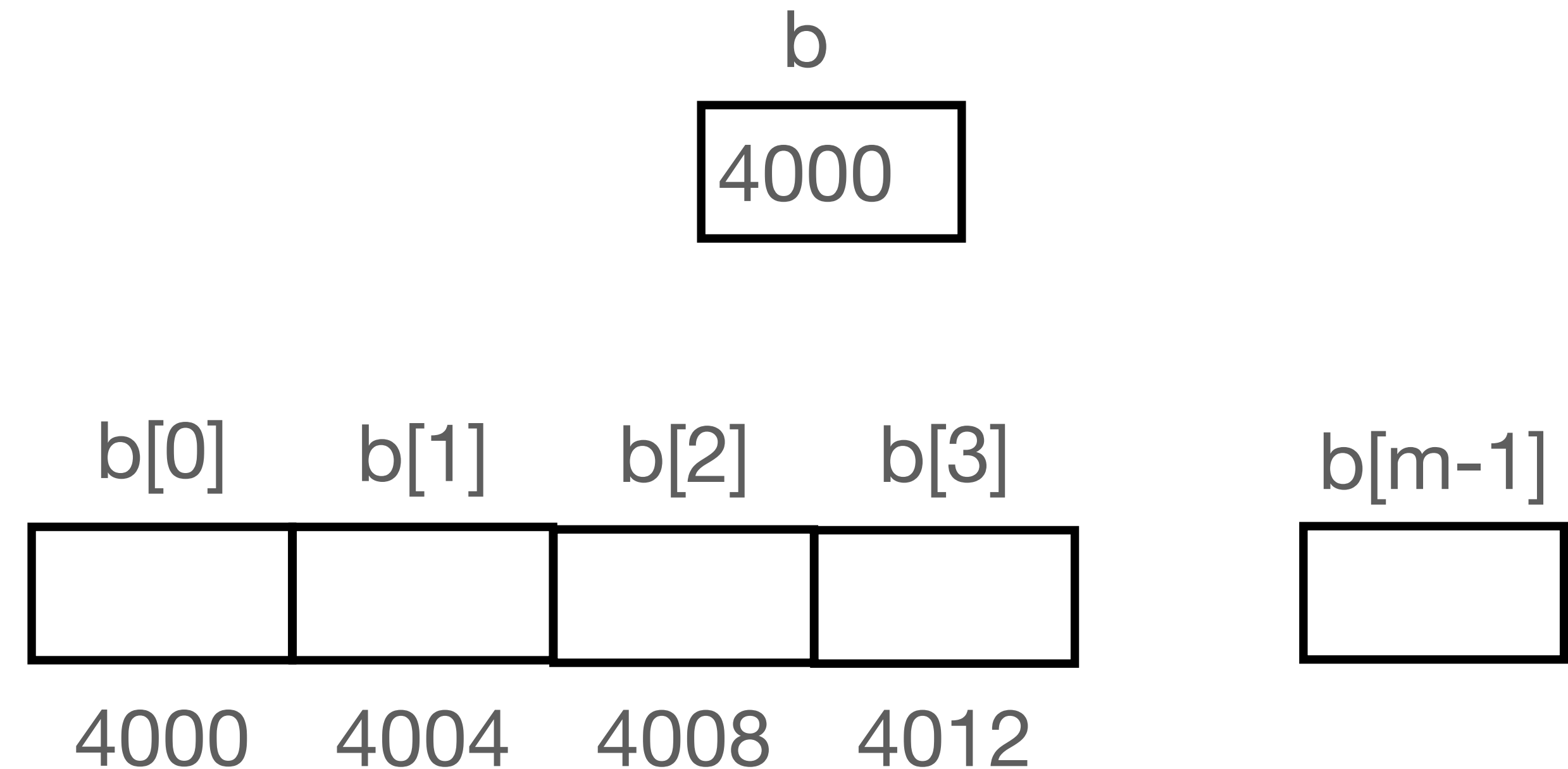
```
a = ( float *) malloc(m * sizeof(float) );
```

```
for ( ..... ) {
```

```
    scanf("%f", a + i);
```

```
    printf("%f", *(a+i));
```

```
}
```



HW for today

HW: Do the dynamic memory allocation for a char pointer.

DMA for a structure

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

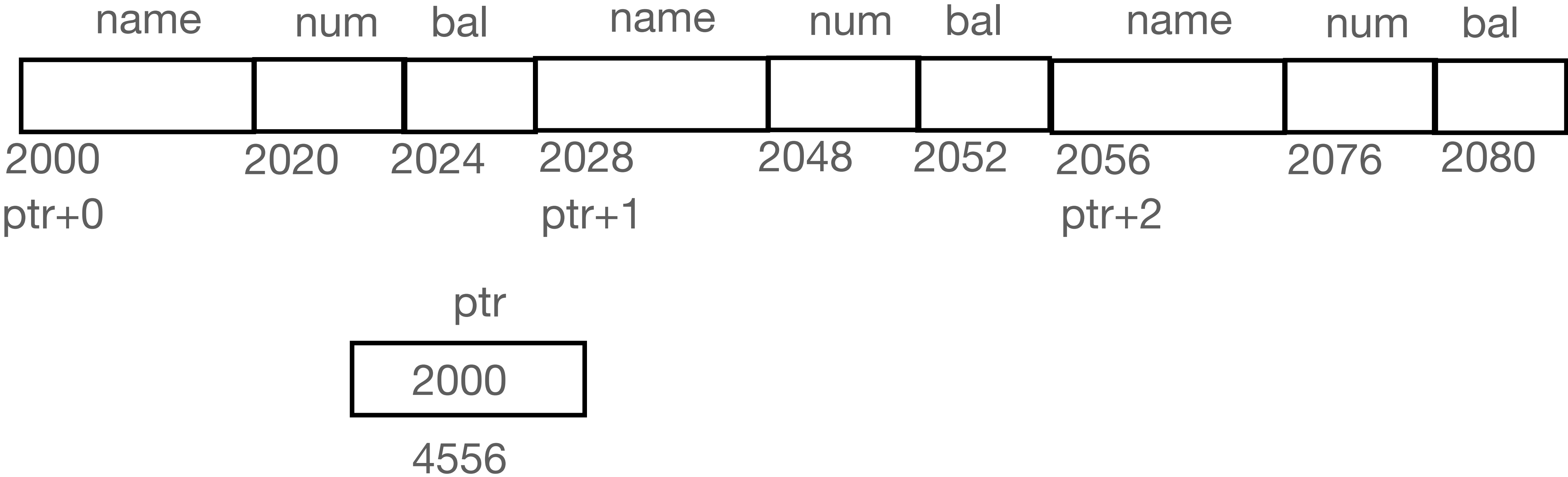
How do you do DMA for structures?

L16_DMA.c

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

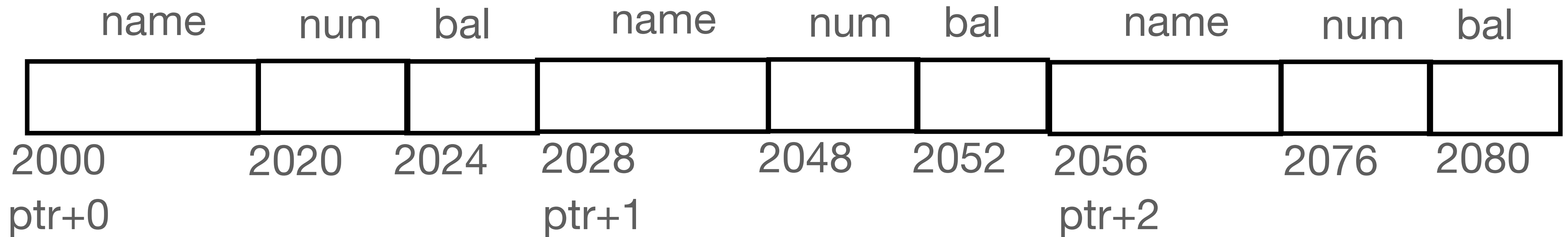
How does DMA for structures look?

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`