

ED1021 - Introduction to computation and visualisation

L15 - Structures in C

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/>

Structures

basics

- To include multiple data type together

Structures

Example of a passbook

- Account Name
- Account Number
- Account Balance
- Date of commencing, other details.

Structures

Example of a passbook

- Account Name
- Account Number
- Account Balance

Structures - User Define datatype

Example of a passbook

- Account Name
- Account Number
- Account Balance

Declaring variables

usual way

```
int main(void)
{
    // For one account holder
    char acc_name1[20];
    int acc_num1;
    float acc_bal1;

    // For another account holder
    char acc_name2[20];
    int acc_num2;
    float acc_bal2;

}
```

Structure way - User defined datatype

Declaring a structure

```
struct AccountDetails {  
    char name[20];  
    int num;  
    float bal;  
};
```

Structure way

Declaring a structure

```
struct NameoftheStructure {  
    Members / elements  
    of the structure with  
    datatype  
};
```

struct AccountDetails {
 char name[20];
 int acc_num;
 float acc_bal;
};

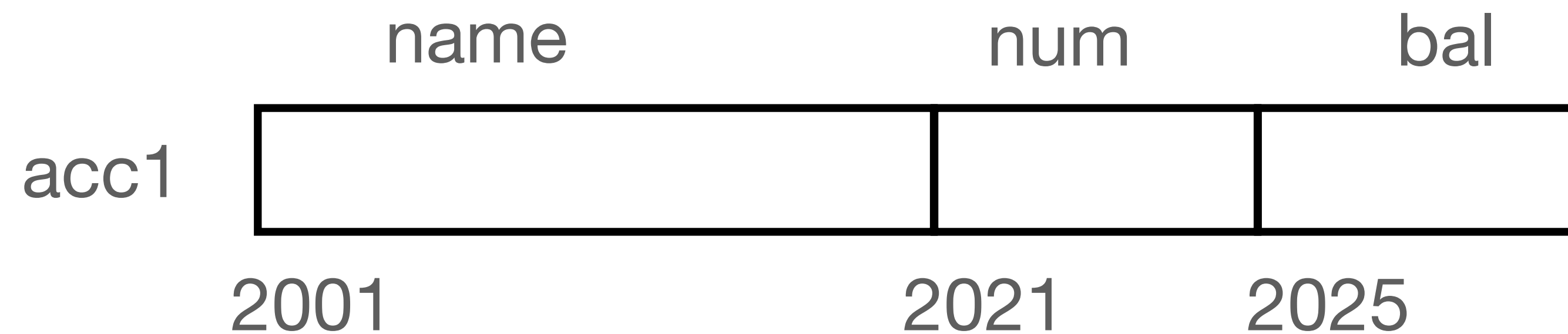
More examples?

CW: Come up with example situations and write the structure details for the same.

- NOTE: Merely writing a structure DOES NOT make it directly usable.
 - `int`
 - `a`
 - `int a;`

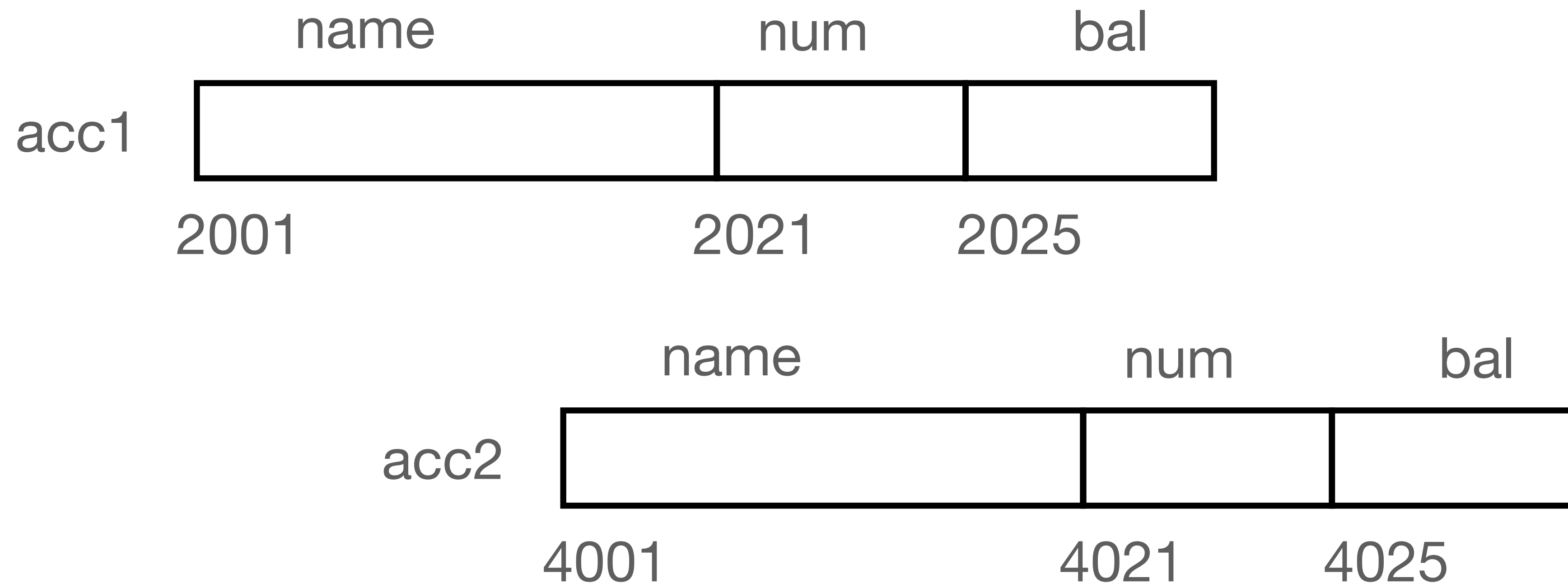
Declaring a variable of structure type

```
struct AccountDetails acc1;
```



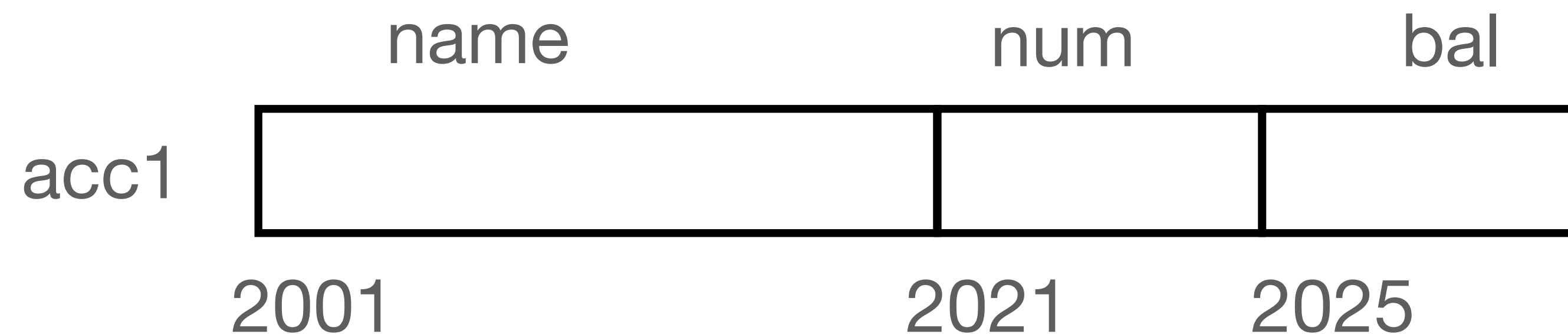
Declaring variables of structure type

```
struct AccountDetails acc1, acc2;
```



Accessing members of a structure using its variable using ‘.’

```
struct AccountDetails acc1;
```



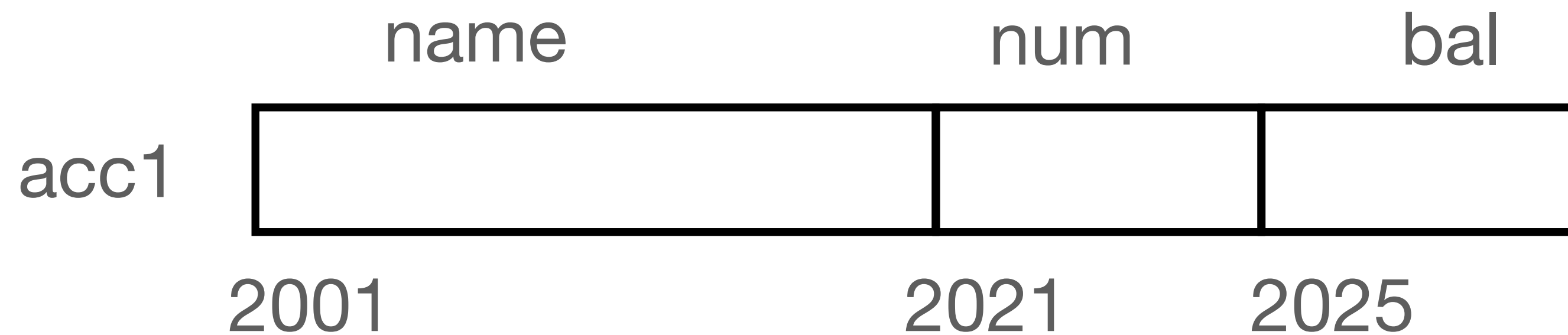
`acc1.name`

`acc1.num`

`acc1.bal`

Accessing members of a structure using its variable using ‘.’

```
struct AccountDetails acc1;
```



`acc1.name`

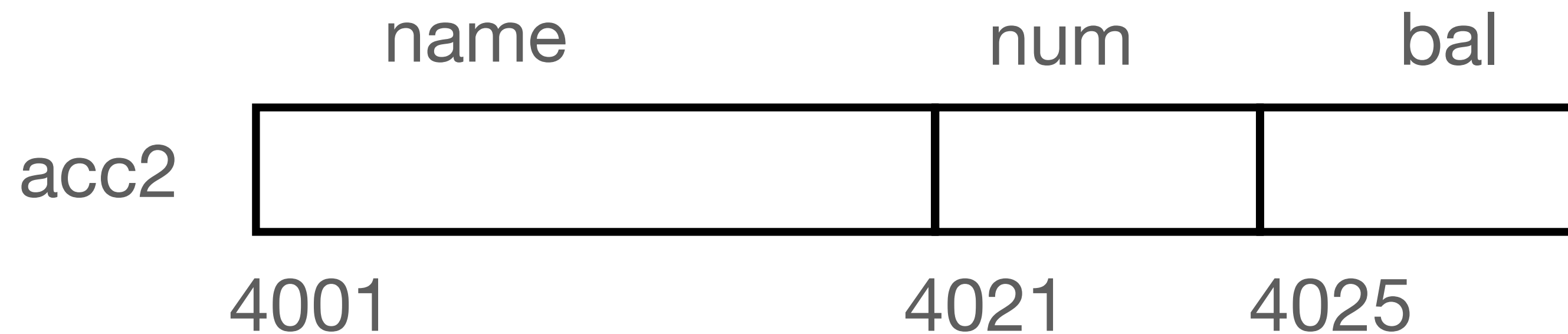
`acc1.num`

`acc1.bal`

- **RULE:**
 - Before the dot - structure variable
 - After the dot - member of the structure.

Declaring variables of structure type

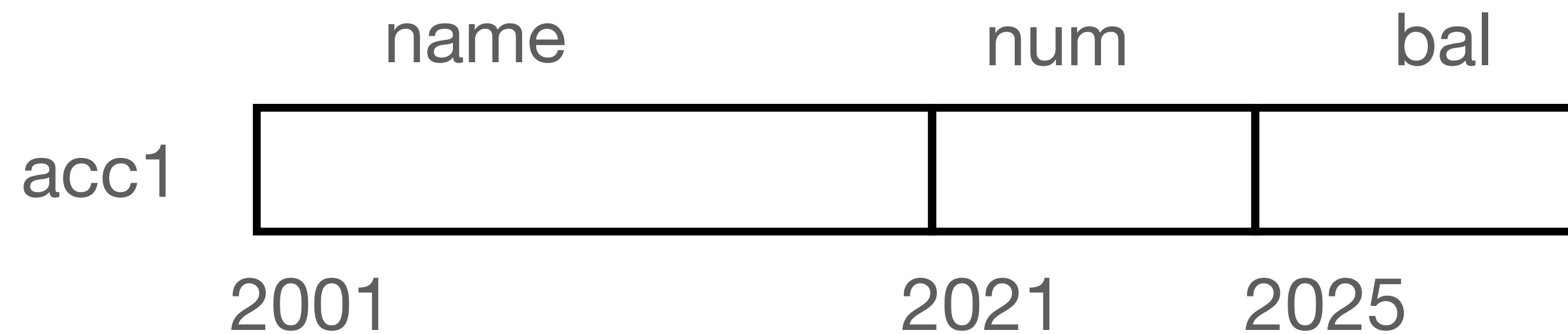
```
struct AccountDetails acc1, acc2;
```



CW: How do you access members using the variable `acc2`?

Printing the locations of the members

```
struct AccountDetails acc1;
```



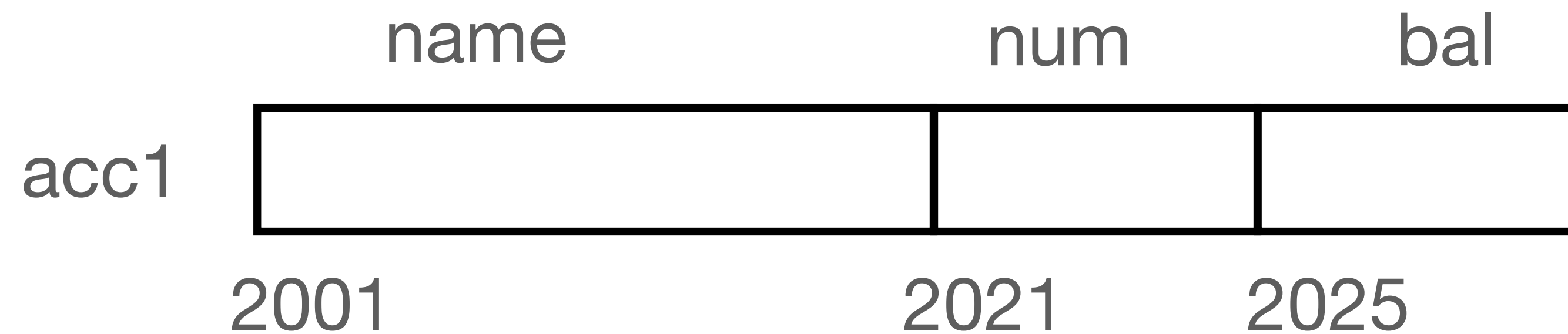
```
acc1.name
```

```
&acc1.num
```

```
&acc1.bal
```


Printing the locations of the members

```
struct AccountDetails acc1;
```



```
printf("printing the addresses of the structure variable acc1, %u %u %u\n",  
       acc1.name, &acc1.num, &acc1.bal);
```

Input and output for structure variable /member

L15_StructureBasics.c

```
struct AccountDetails acc1;
```

```
scanf("%s %d %f", acc1.name, &acc1.num, &acc1.bal);  
printf("Account name = %s, Account number = %d, Account balance = %f\n", acc1.name,  
acc1.num, acc1.bal);
```

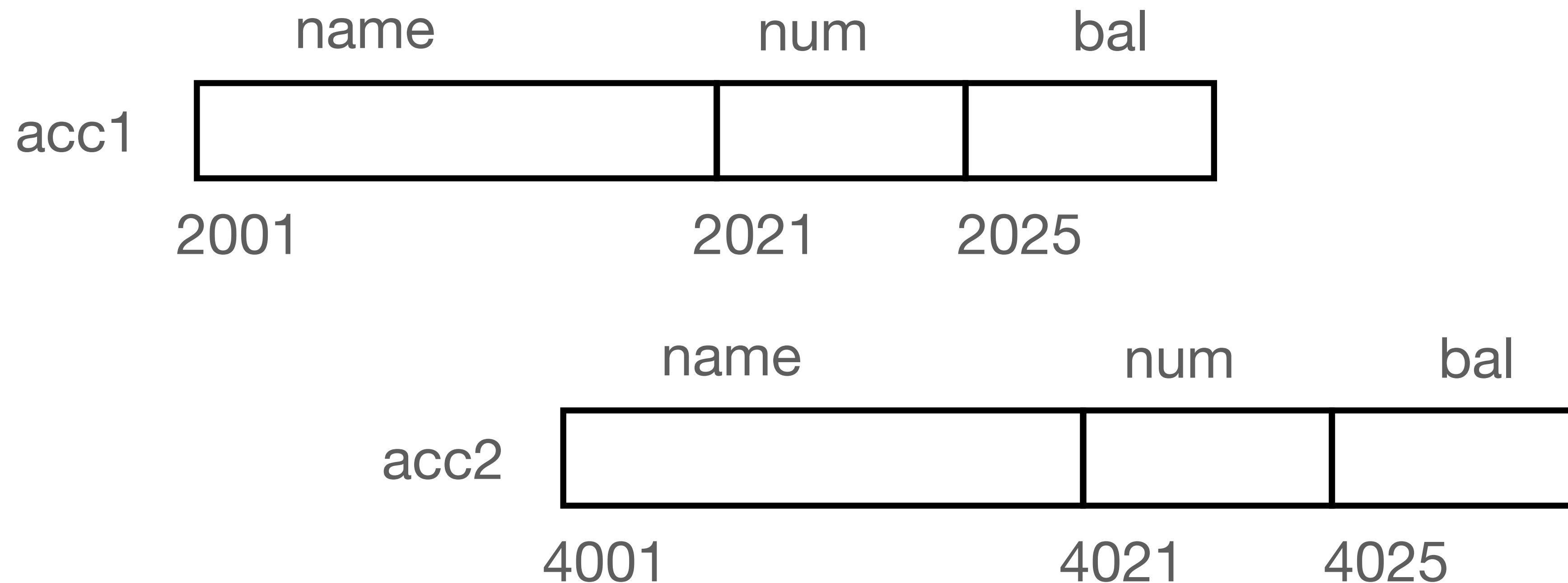
Homework for today

HW (WAP): For the example situations and write the structure details and take input and output for the same. At least three different examples are needed!

Array of Structures

Two users

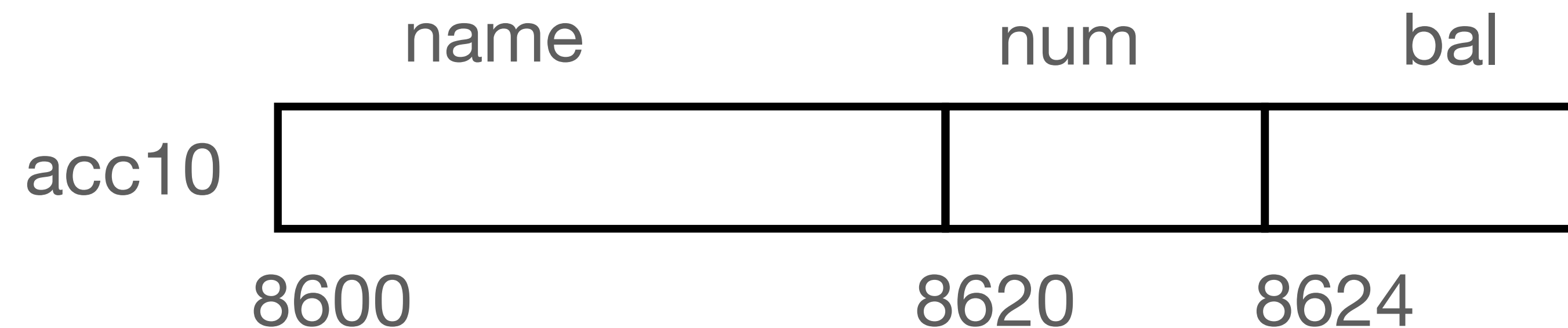
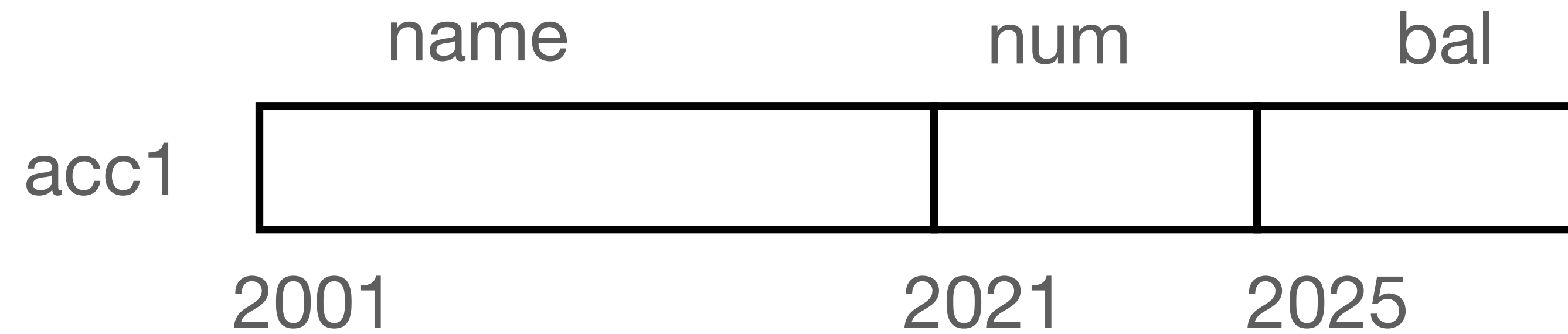
```
struct AccountDetails acc1, acc2;
```



Ten users

how will the declaration be?

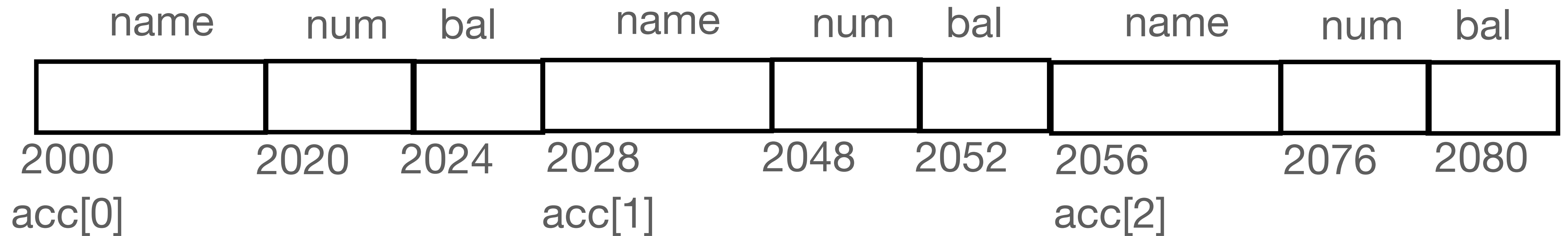
```
struct AccountDetails acc1, acc2, acc3,... , acc10;
```



Alternative?

Array of structures

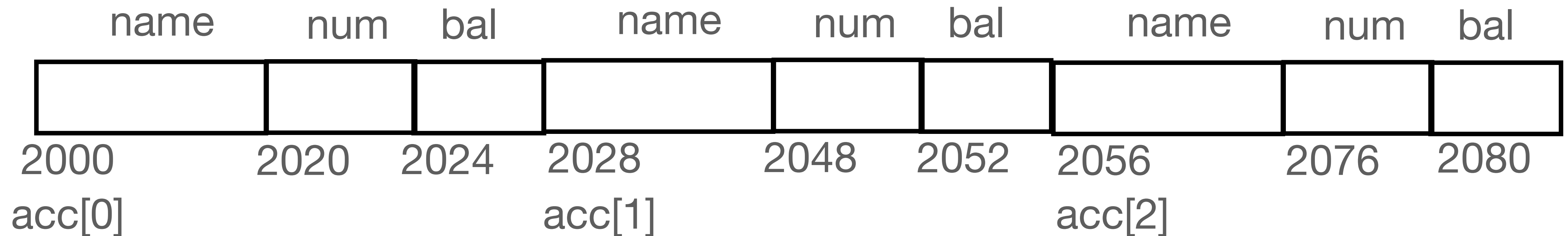
```
struct AccountDetails acc[10];
```



Alternative?

Array of structures

```
struct AccountDetails acc[10];
```

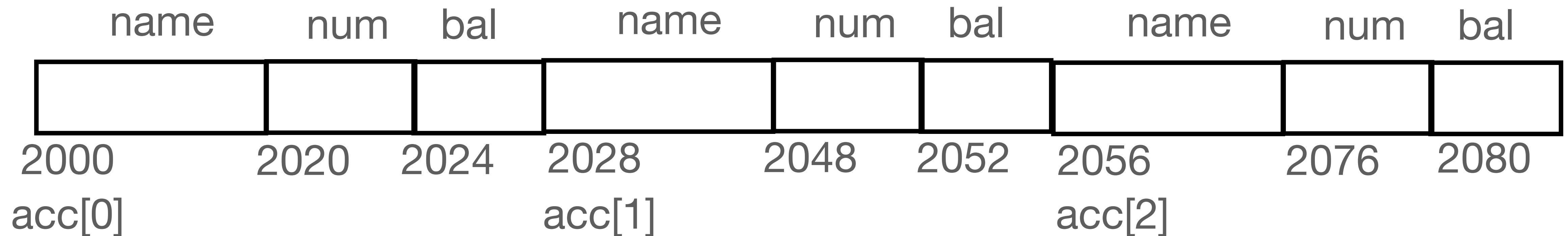


acc[0].name	acc[1].name	acc[2].name
acc[0].num	acc[1].num	acc[2].num
acc[0].bal	acc[1].bal	acc[2].bal

Array of structures

Input

```
struct AccountDetails acc[10];
```



acc[0].name	acc[1].name	acc[2].name
&acc[0].num	&acc[1].num	&acc[2].num
&acc[0].bal	&acc[1].bal	&acc[2].bal

CW for today

CW: WAP for an array of a structure and take inputs for the same and print them.

Using typedef

using as alias (substitute)

```
struct AccountDetails acc1, acc2;
```

```
typedef struct AccountDetails AD;  
typedef int INT;
```

Using typedef

using as alias (substitute) - L15_StructureBasics

```
struct AccountDetails acc1, acc2;
```

```
typedef struct AccountDetails AD;  
AD acc1, acc2; // Variables  
AD acc[5]; // Array of structure
```

CW for today

CW: Suppose you have 5 students in a class and each of them do five subjects. Compute the avg marks of each subject and % of marks for each student. Print the output along with each student name and roll number?

How do you approach the above problem?

Structure detail and variables

```
int main(void)
{
    struct StudentDetail{
        char name[20];
        int rollnum;
        float MarksInSubject[5];
    };
    typedef struct StudentDetail SD;
    SD Students[5];
}
```

Passing structures / members as arguments

Structures - Function arguments

- You can pass the entire structure
- You can pass individual members

Using the following structure

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

Passing the entire structure as argument

```
void OutputStruct(AD acc1);
```

```
// Entire structure as function argument
```

```
void OutputStruct(AD account)
{
    printf("Account name = %s, Account number = %d,
Account balance = %f\n", account.name, account.num, account.bal);
}
```

```
int main(void) {

    AD acc1;

    scanf("%s %d %f", acc1.name, &acc1.num, &acc1.bal);
    // Calling the function by passing the entire structure
    OutputStruct(acc1);
}
```

Passing the member(s) of a structure as argument - L15_StructureAndFunction.c

```
void changeName(char *name);

// Changing the user name using the member of the structure
void changeName(char *name)
{
    printf("Enter the name that is to be changed : ");
    scanf("%s", name);
}

int main(void) {

    AD acc1;

    scanf("%s %d %f", acc1.name, &acc1.num, &acc1.bal);
    // Calling the function by passing the entire structure
    OutputStruct(acc1);

    // Changing the name of an account by passing name (member)
    // of the account (inside the main)
    changeName(acc1.name);
}
```

CW for today

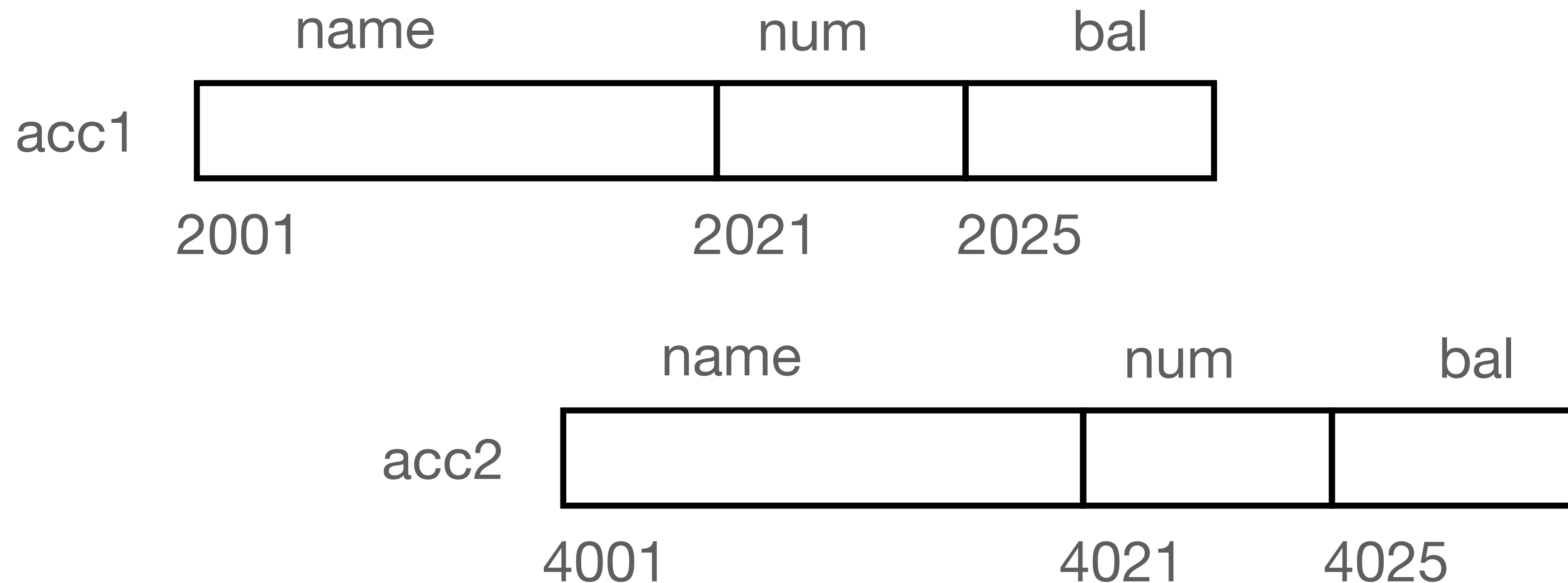
CW: Upgrade changeName function to change both name as well as the account number.

**HW: Using array of structures,
input the coordinates of a
polygon and display the same
using OpenGL.**

Pointer to a structure

Pointer to a structure

```
typedef struct AccountDetails AD;  
AD acc1, acc2;
```

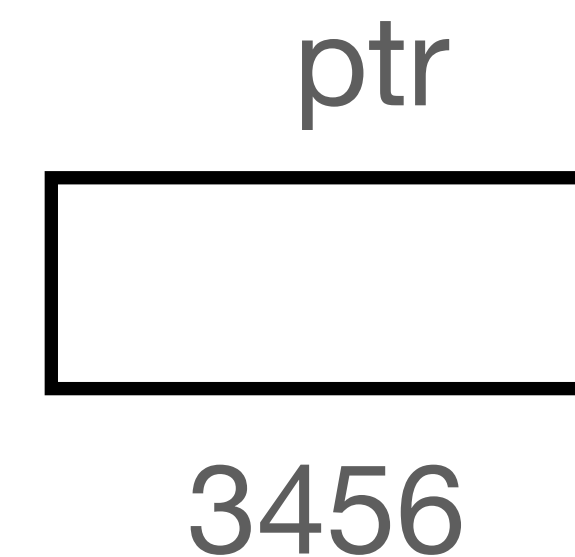
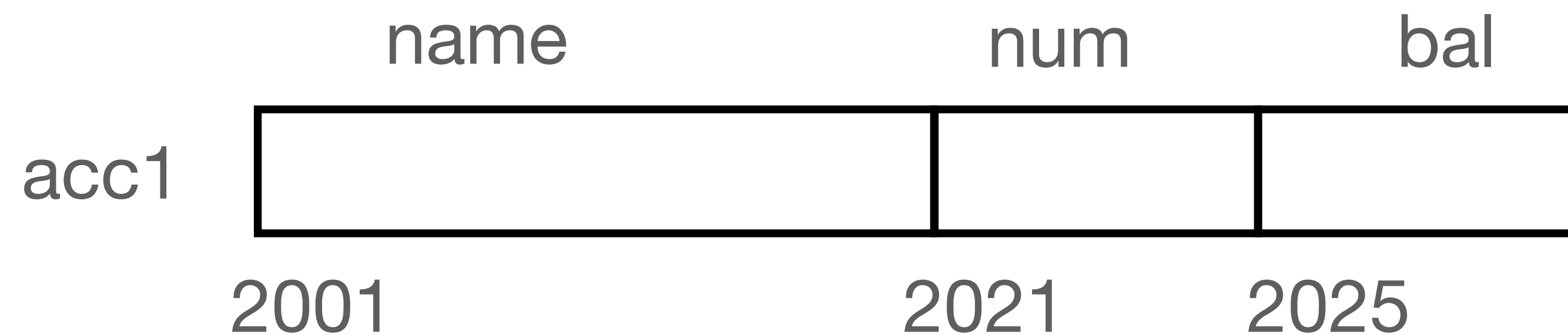


Pointer to a structure

Also called as structure pointer

AD acc1;

AD *ptr;



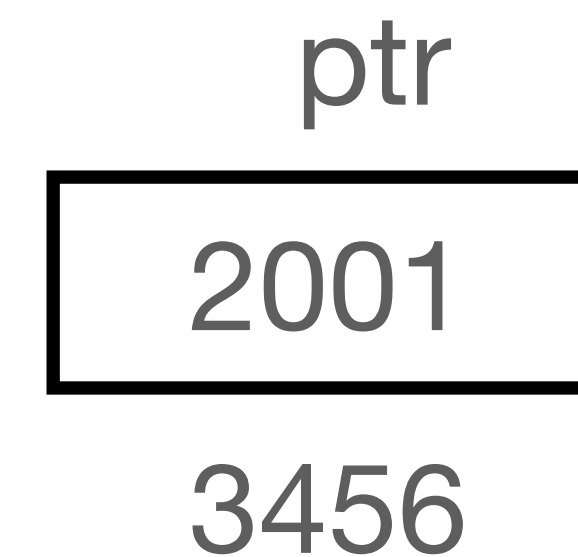
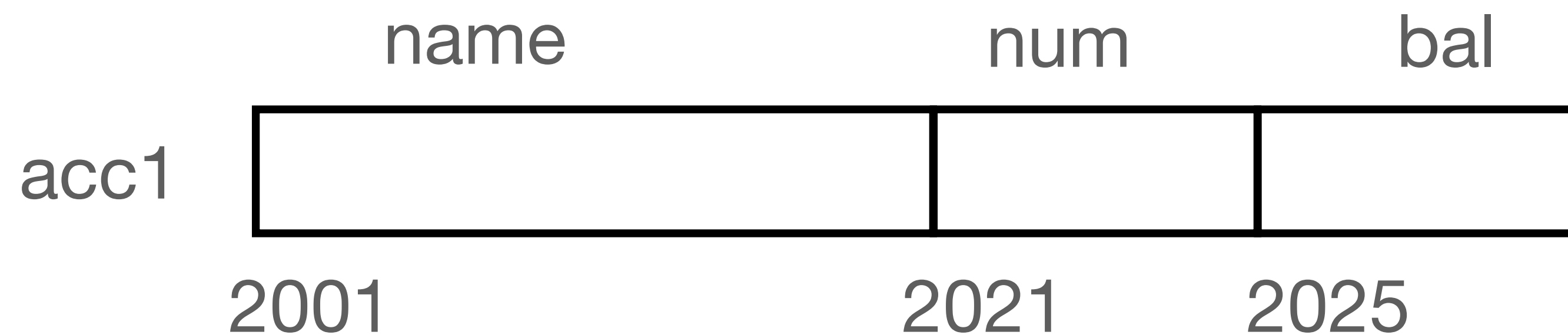
Pointer to a structure

Also called as structure pointer

```
AD acc1;
```

```
AD *ptr;
```

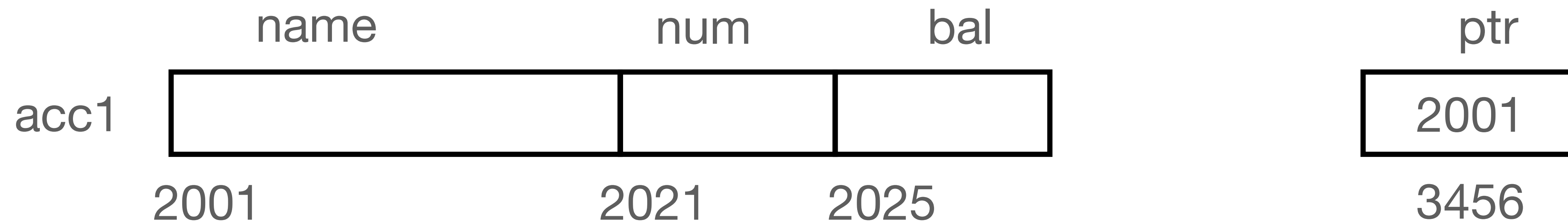
```
ptr = &acc1;
```



Accessing the members using pointer

Indirection operator

```
ptr = &acc1;
```



`acc1.name`

`acc1.num`

`acc1.bal`

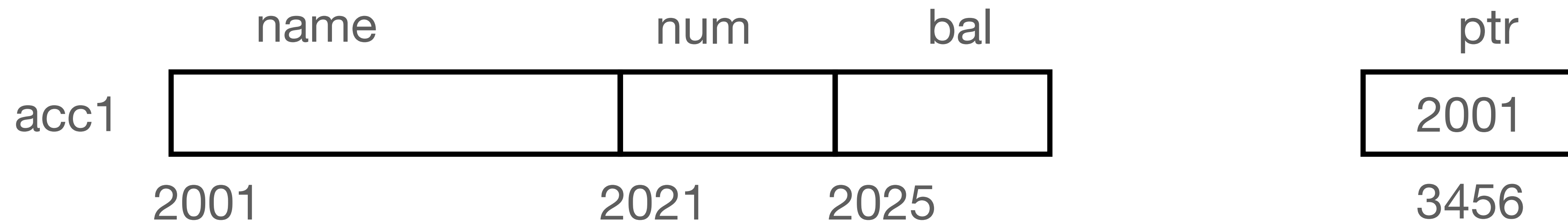
- **RULE:**
 - Before the dot - structure variable
 - After the dot - member of the structure.

What is the meaning of `*ptr`?

Accessing the members using pointer

Indirection operator

```
ptr = &acc1;
```



```
(*ptr).name
```

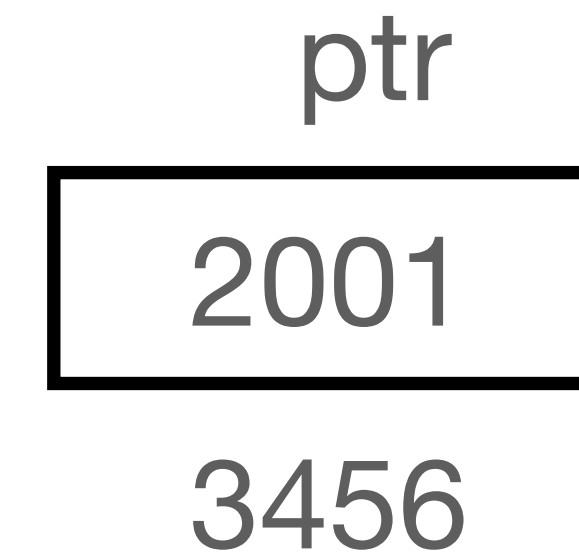
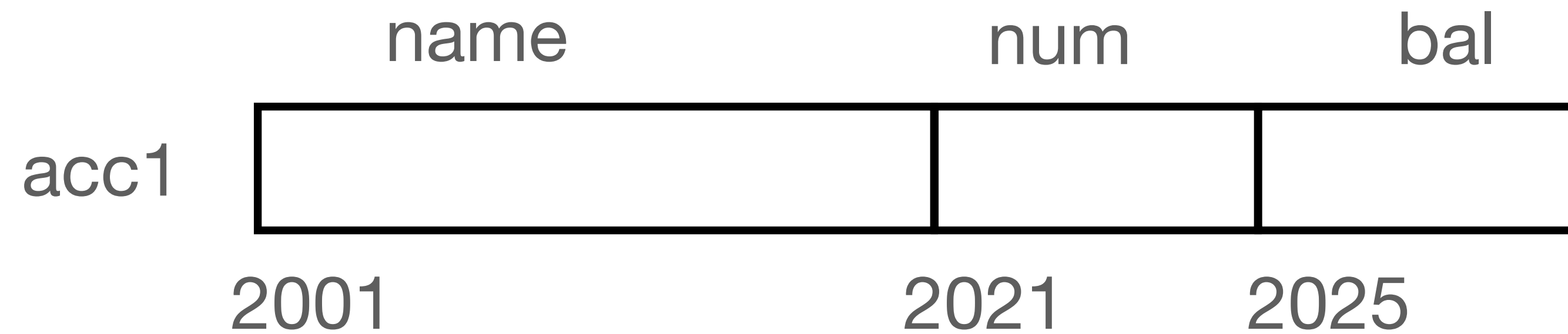
```
(*ptr).num
```

```
(*ptr).bal
```

Accessing the members using —>

Right arrow

```
ptr = &acc1;
```



```
ptr->name
```

```
ptr->num
```

```
ptr->bal
```

- **RULE:**
 - Before the `—>`, pointer to structure
 - After the `—>`, member of the structure.

CW for today

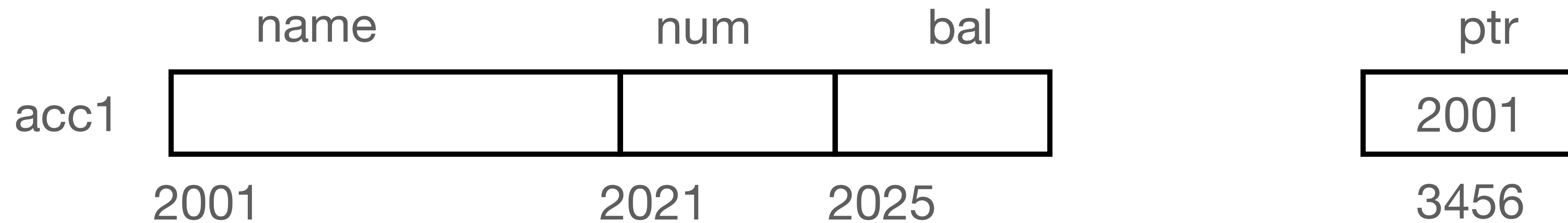
L15_StructurePointer.c

CW: Upgrade yesterday's program by including a pointer to a structure for acc1 and use scanf and printf with the pointer notation.

What to use in scanf when `—>` appears

Confusion?

```
ptr = &acc1;
```



```
ptr->name  
&ptr->num  
&ptr->bal
```

- **RULE:**
 - Before the `—>`, pointer to structure
 - After the `—>`, member of the structure.

CW for today

L15_StructurePointer.c

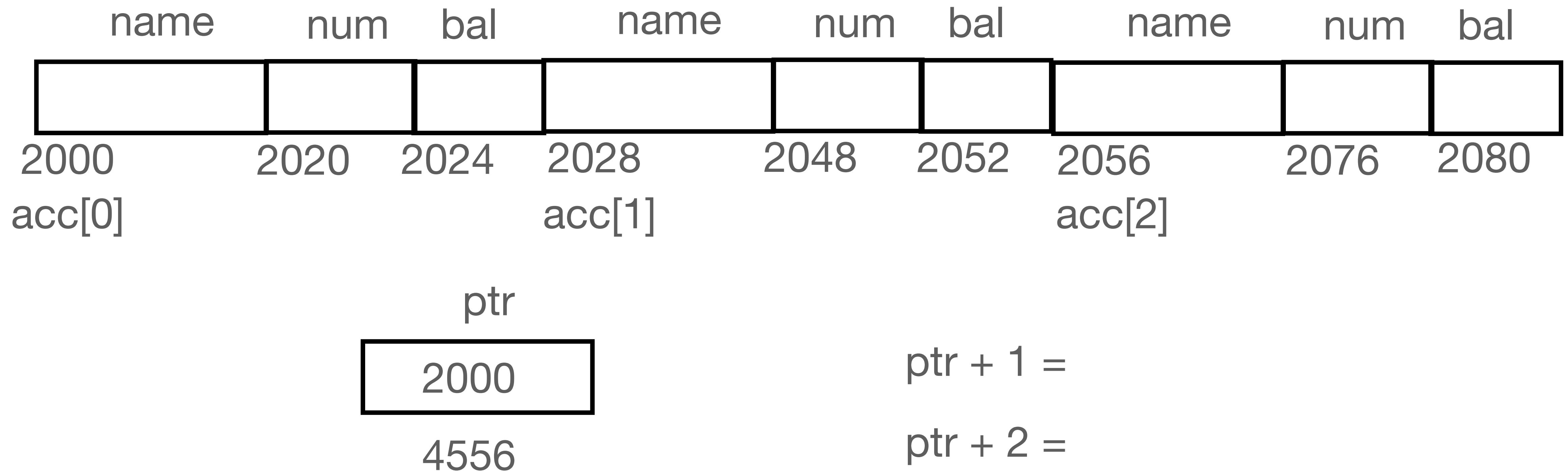
CW: Send the pointer to structure as an argument and put new values for its members inside a UDF. Print the output in the main.

Array of structures and pointer

Look carefully at the pointer arithmetic

AD `acc[3];`

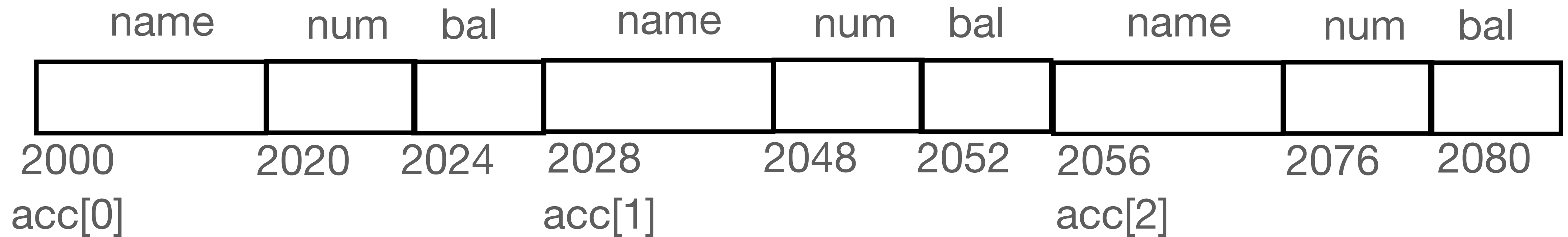
AD `*ptr = acc;`



Array of structures and pointer

AD acc[3];

AD *ptr = acc;



ptr+0 is &acc[0]

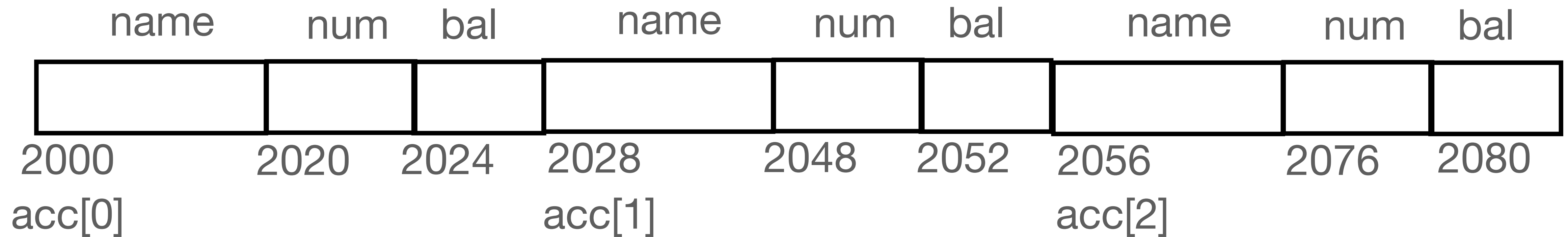
ptr+1 is &acc[1]

ptr+2 is &acc[2]

Array of structures and pointer

AD `acc[3];`

AD `*ptr = acc;`



`*(ptr+0)` is `acc[0]`

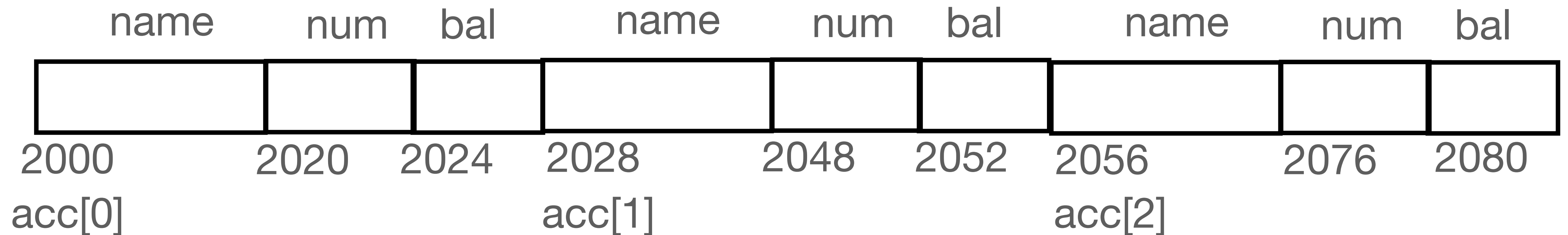
`*(ptr+1)` is `acc[1]`

`*(ptr+2)` is `acc[2]`

Array of structures and pointer

AD `acc[3];`

AD `*ptr = acc;`

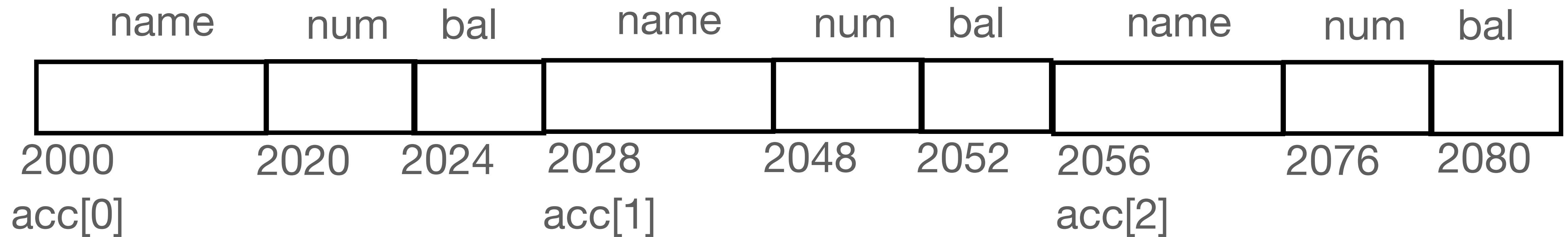


`acc[0].name` is `*(ptr+0).name` is `(ptr+0)->name`
`acc[0].num` is `*(ptr+0).num` is `(ptr+0)->num`
`acc[0].bal` is `(ptr+0)->bal`

Array of structures and pointer

AD acc[3];

AD *ptr = acc;



acc[1].name is (ptr+1)->name

acc[1].num is (ptr+1)->num

acc[1].bal is (ptr+1)->bal

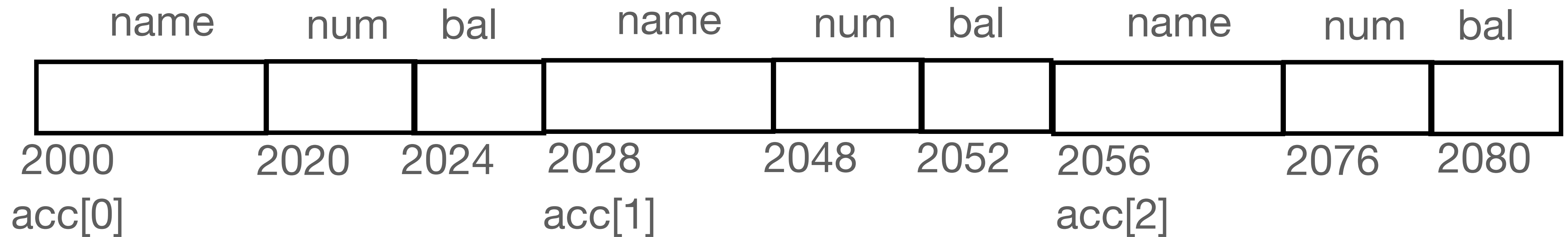
- RULE:

- Before the \rightarrow , pointer to structure
- After the \rightarrow , member of the structure.

Array of structures and pointer

AD acc[3];

AD *ptr = acc;



acc[2].name is (ptr+2)->name

acc[2].num is (ptr+2)->num

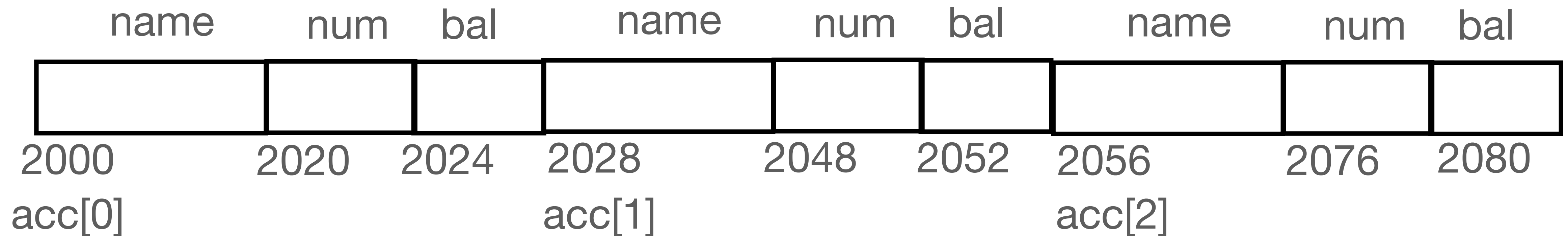
acc[2].bal is (ptr+2)->bal

Array of structures and pointer

Input

AD `acc[10];`

AD `*ptr = acc;`



`acc[i].name` is `(ptr+i)->name`

`&acc[i].num` is `&(ptr+i)->num`

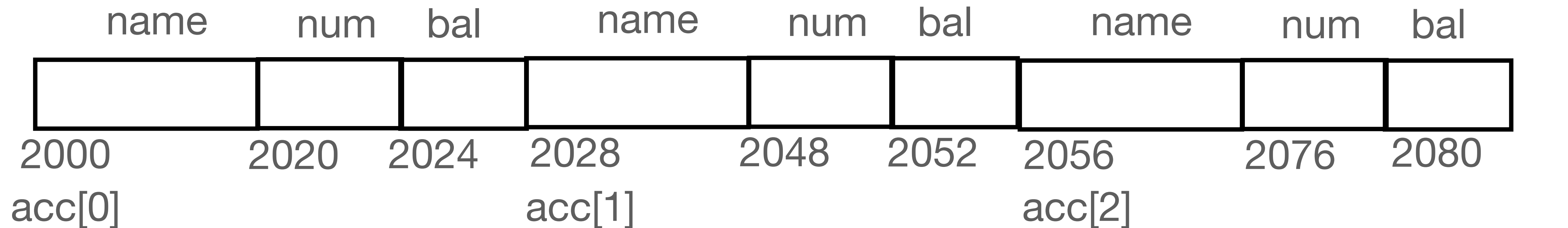
`&acc[i].bal` is `&(ptr+i)->bal`

Array of structures and pointer

Input

AD `acc[10];`

AD `*ptr = acc;`



`acc[i].name` is `(ptr+i)->name` is `(acc+i)->name`
`&acc[i].num` is `&(ptr+i)->num` is `&(acc+i)->num`
`&acc[i].bal` is `&(ptr+i)->bal` is `&(acc+i)->bal`

HW

L15_StructurePointer.c

HW: Instead of the array notation in yesterday's homework problem, use pointer notation.