POINTERS TO POINTERS - MIND BLOWINGLY BENT

```
int value1 = 10;
int value2 = 5;

int* ptr = &value1;
int** ptr_ptr = &ptr;
**ptr_ptr = 20;

// What will this print to screen? Specify the exact values.
printf("On editing dereferenced pointer to pointer:\n");
printf("value1: %d\n", value1);
printf("*ptr: %d\n", *ptr);
printf("*ptr_ptr: %d\n", **ptr_ptr);
```

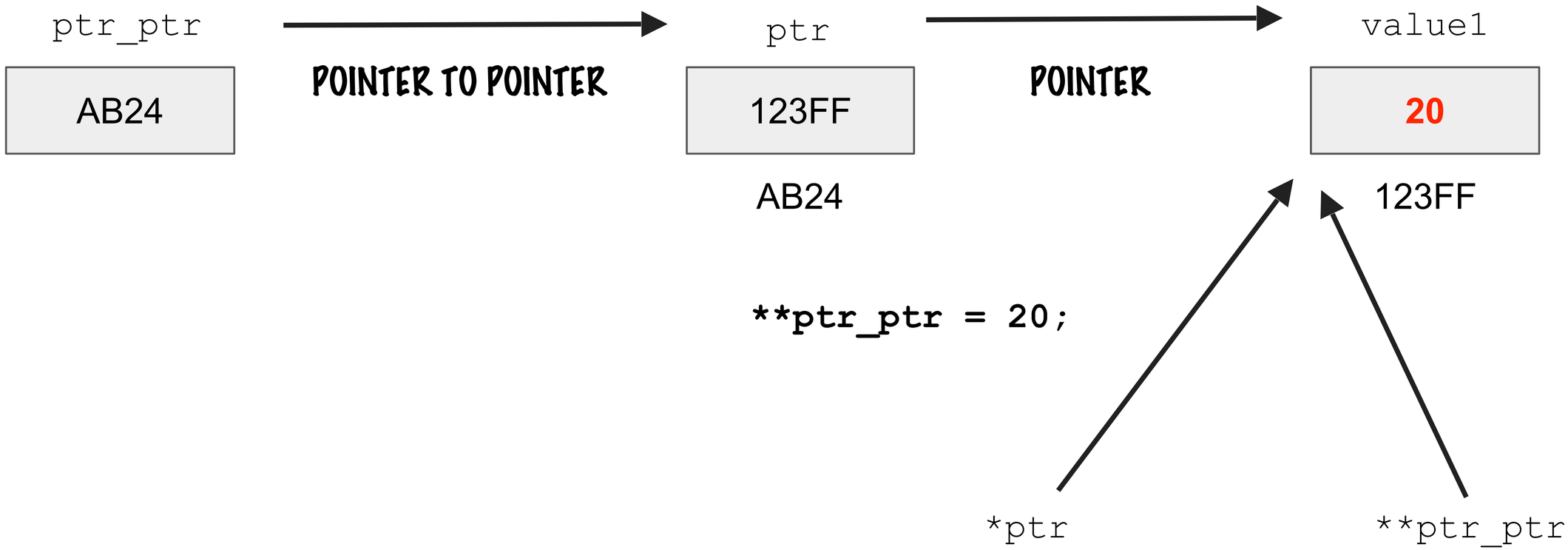
HINT: POINTER TO POINTER DEREFERENCED TWICE WHAT ACTUALLY GOT UPDATED? WHAT ELSE POINTS TO THE SAME MEMORY LOCATION?

ANSWER

```
On editing dereferenced pointer to pointer:
value1: 20
*ptr: 20
*ptr_ptr: 20
```

**PTR_PTR ACCESSES VALUE1, IF VALUE1 CHANGES THE VALUE POINTER TO BY PTR CHANGES AS WELL, ITS ALL THE SAME MEMORY LOCATION

POINTERS TO POINTERS MODIFICATION



POINTERS TO POINTERS - THE FINAL TOUCH

```c
int value1 = 10;
int value2 = 5;

int* ptr = &value1;
int** ptr_ptr = &ptr;

*ptr_ptr = &value2;

// What will this print to screen? Specify the exact values.
printf("On reassigning derefenced pointer:\n");
printf("*ptr: %d\n", *ptr);
printf("**ptr_ptr: %d\n", **ptr_ptr);
```

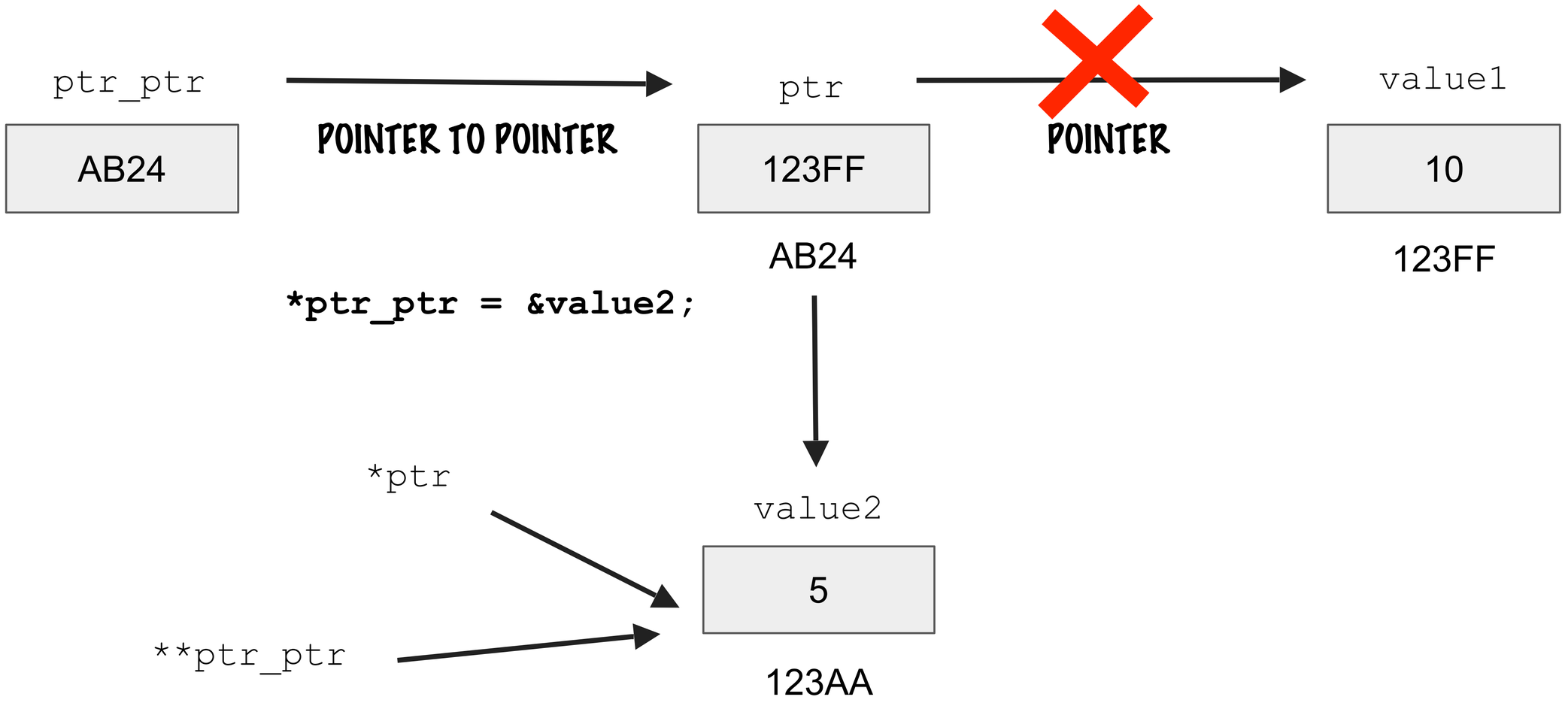
HINT: A SINGLE DEFERENCE HERE IS A POINTER WHICH IS NOW POINTING TO A DIFFERENT PLACE, WHAT OTHER POINTER IS BEING AFFECTED?

ANSWER

```
On reassigning derefenced pointer:
*ptr: 5
**ptr_ptr: 5
```

*ptr_ptr NOW POINTS TO VALUE2 I.E IT HAS THE ADDRESS OF VALUE2 IN IT
*ptr_ptr IS EQUIVALENT TO PTR - ITS THE SAME MEMORY LOCATION

POINTERS TO POINTERS REASSIGNMENT



Pointers to structures

Pointers work with primitive types in C, they also work with structures or user-defined types

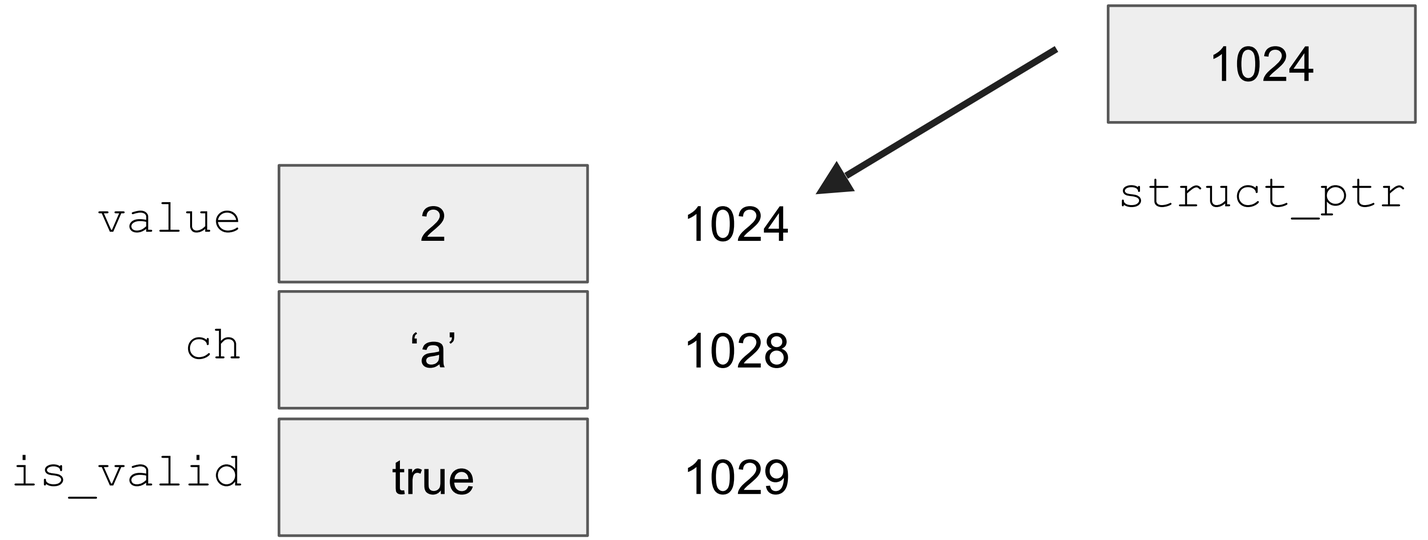
Structures in C are made up of different elements, they are laid out in memory one element after another

The pointer points to the very first element in a structure

Pointer arithmetic works exactly like with ints, chars etc, C knows the size of structs

POINTERS TO STRUCTURES - MEMORY LAYOUT

```
struct SomeData {
  int value;
  char ch;
  bool is_valid;
}
```




```
struct SomeData some_data;
some_data.value = 2;
some_data.ch = 'a';
some_data.is_valid = true;

struct SomeData *struct_ptr = &some_data;

some_data->value = 5;
```

NOTE THE "." WHEN ACCESSING STRUCTURE MEMBERS FROM AN INSTANCE VARIABLE

WHEN ACCESSING STRUCTURE MEMBERS THROUGH A POINTER USE THE "->"