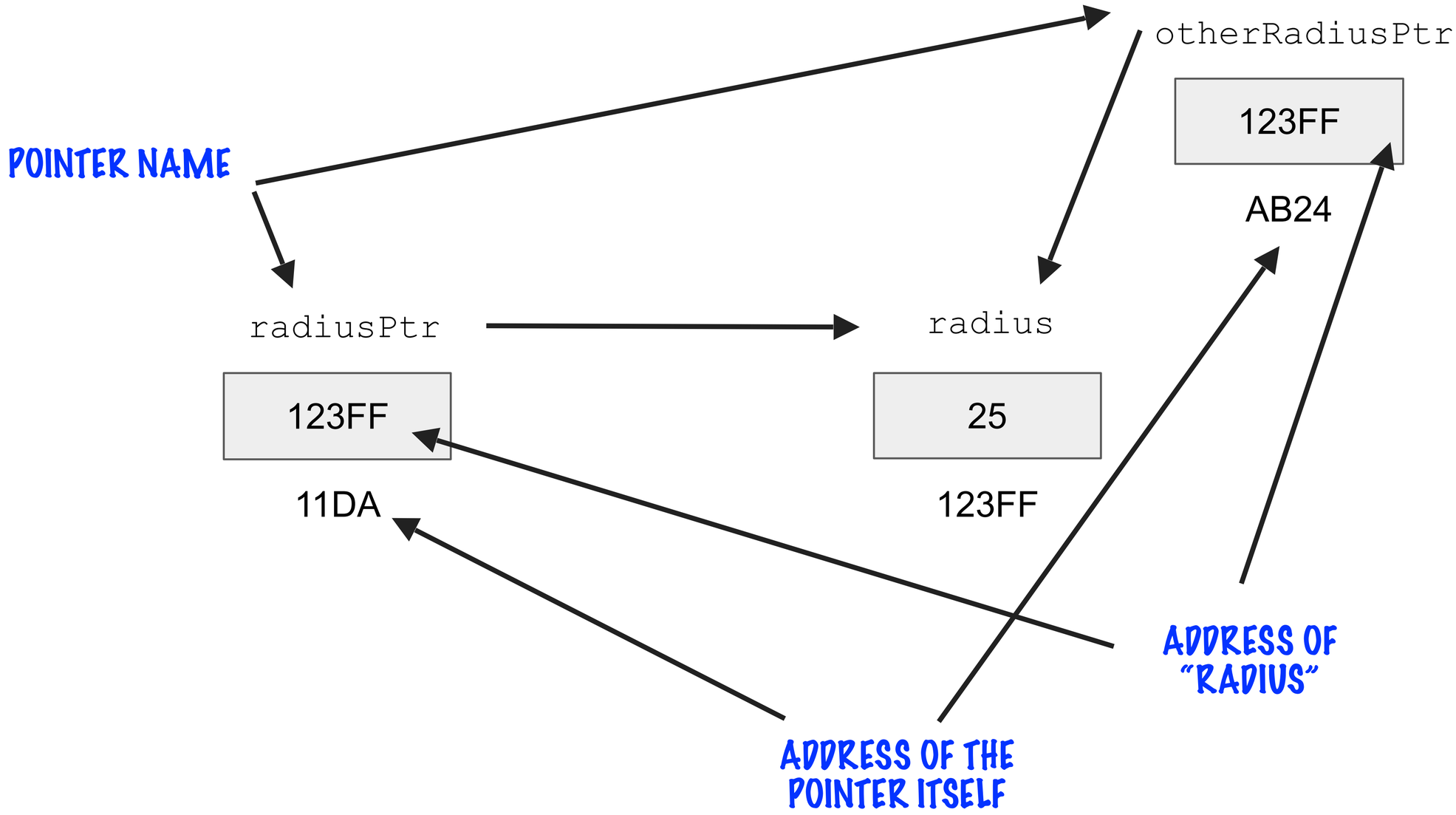
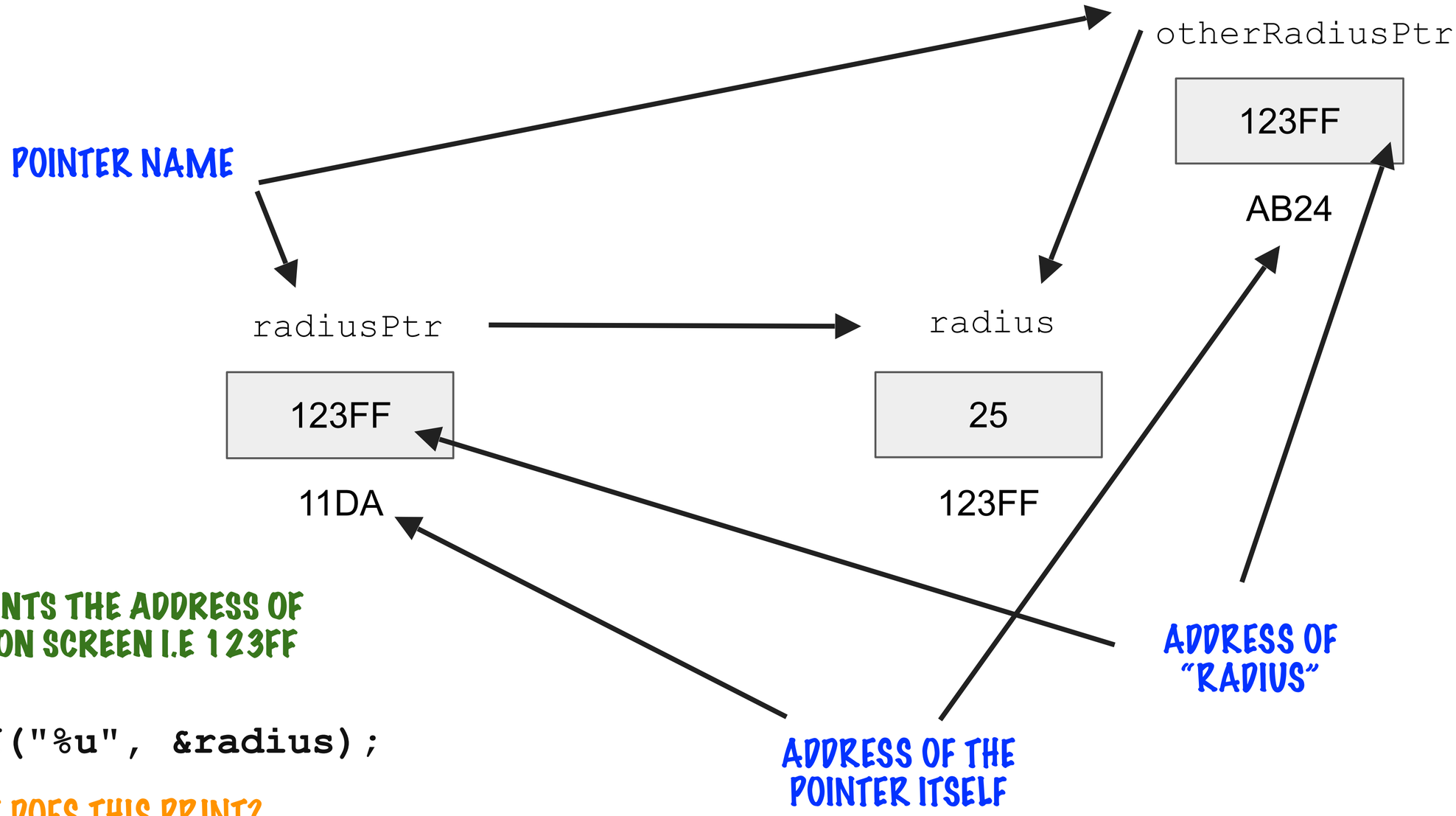
WORKING WITH POINTERS



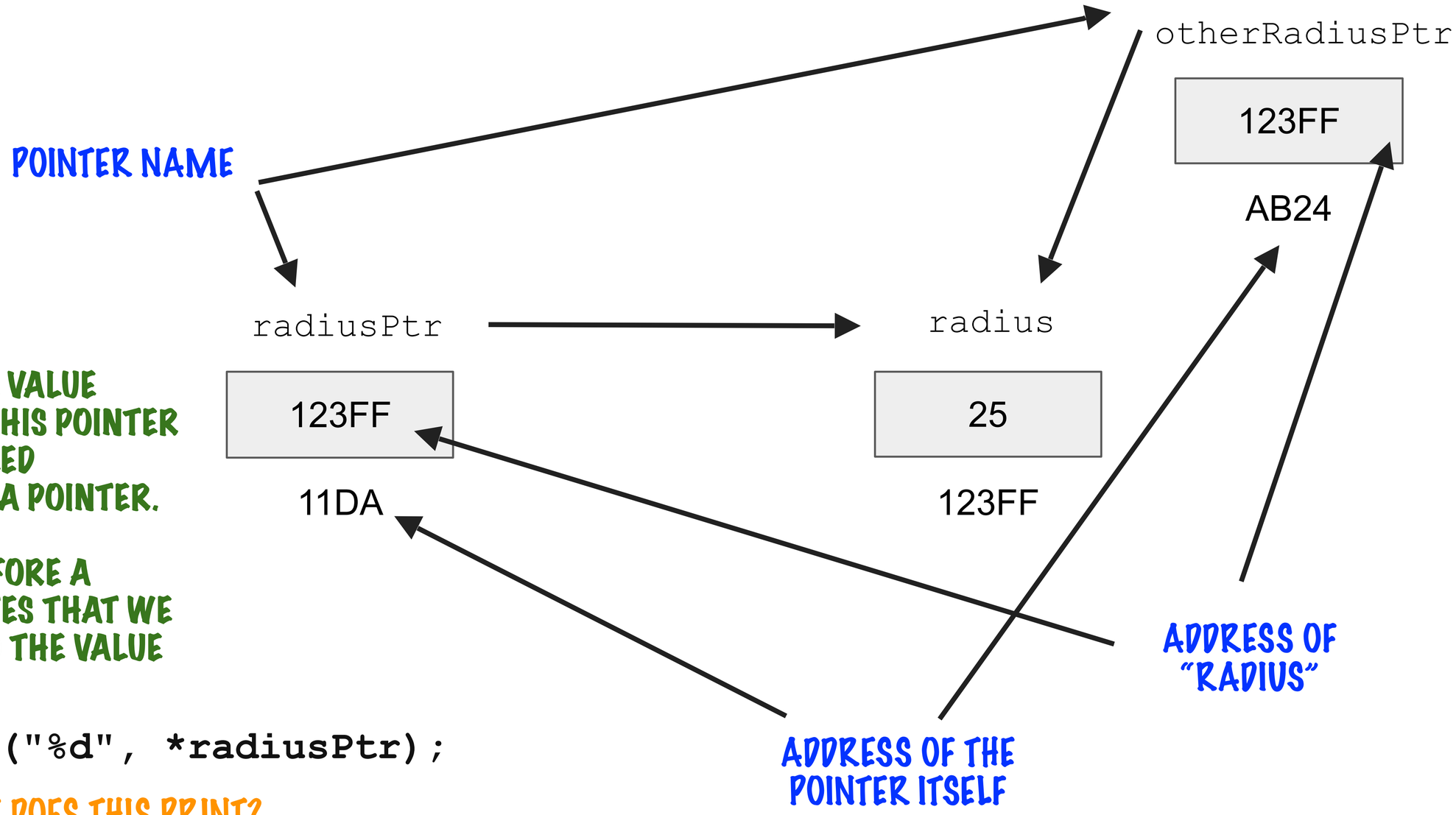
WORKING WITH POINTERS




```
printf("%u", &radius);
```

WHAT DOES THIS PRINT?

WORKING WITH POINTERS



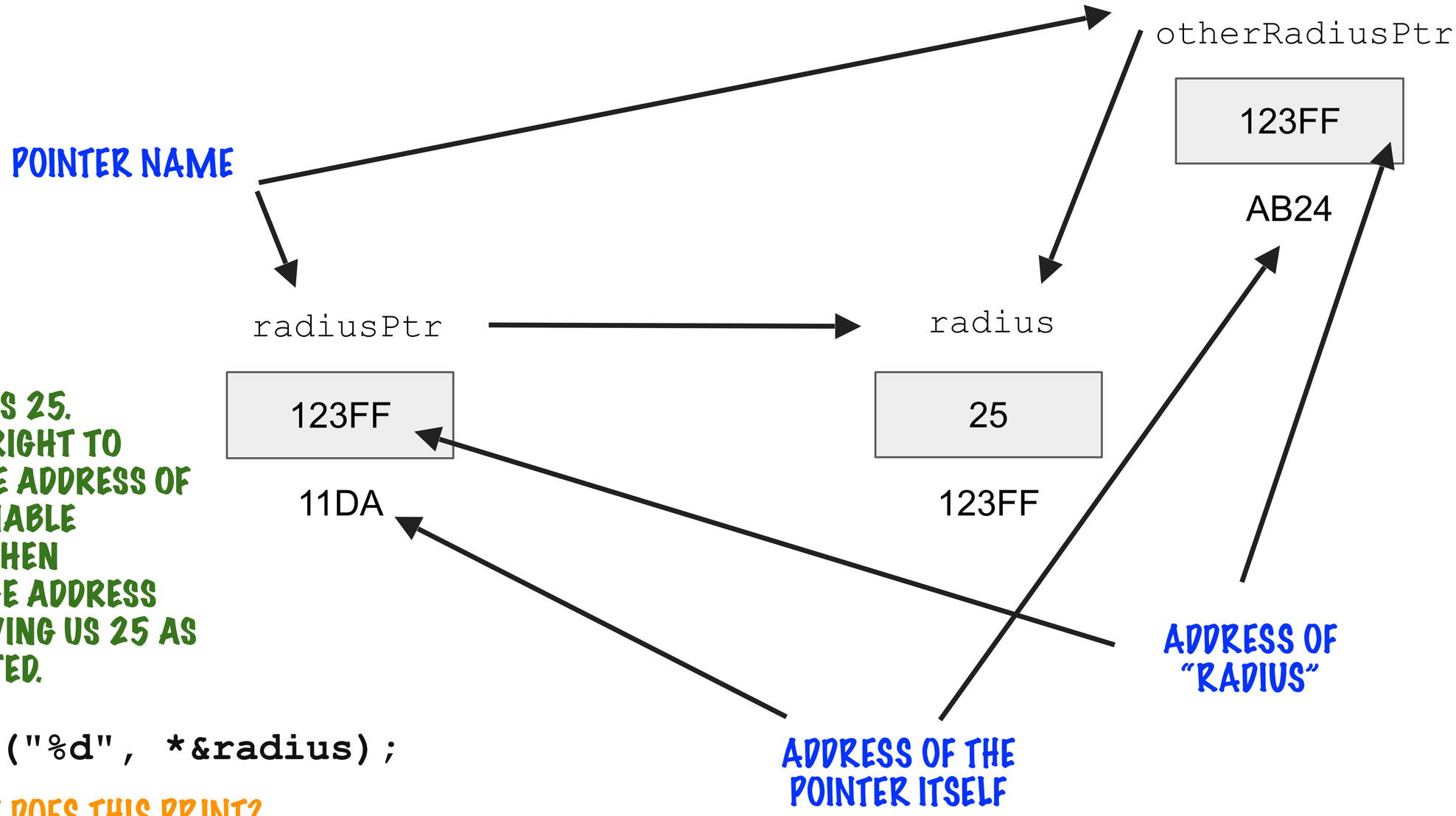
THIS PRINTS THE VALUE POINTED TO BY THIS POINTER I.E. 25. IT IS CALLED DEREFERENCING A POINTER.

THE USE OF * BEFORE A POINTER INDICATES THAT WE WANT TO GET TO THE VALUE POINTED TO.

```
printf("%d", *radiusPtr);
```

WHAT DOES THIS PRINT?

WORKING WITH POINTERS

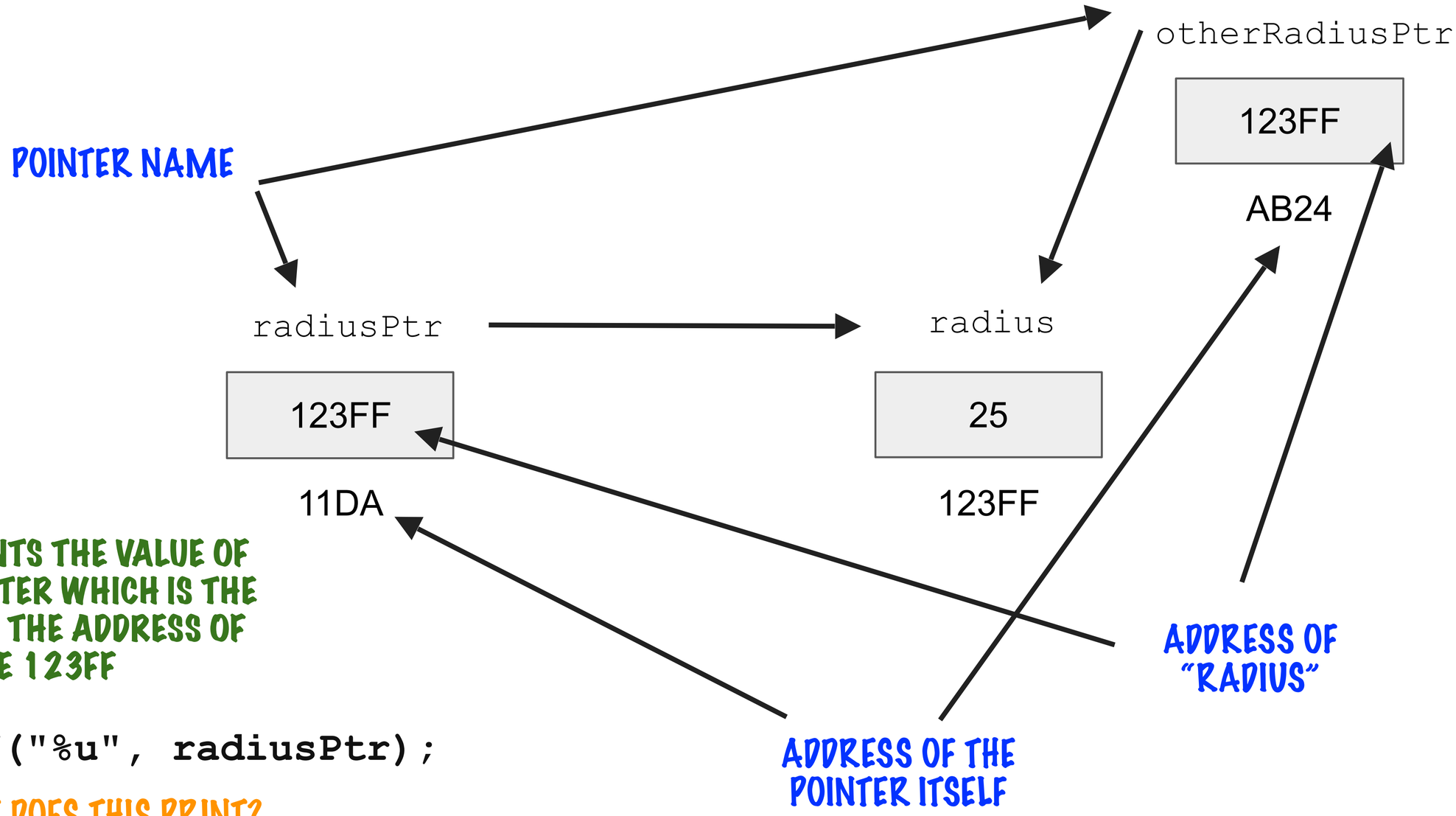


THIS ALSO PRINTS 25. READING FROM RIGHT TO LEFT, WE GET THE ADDRESS OF THE RADIUS VARIABLE (&RADIUS) AND THEN DEREFERENCE THE ADDRESS *(&RADIUS) - GIVING US 25 AS THE VALUE PRINTED.

```
printf("%d", *&radius);
```

WHAT DOES THIS PRINT?

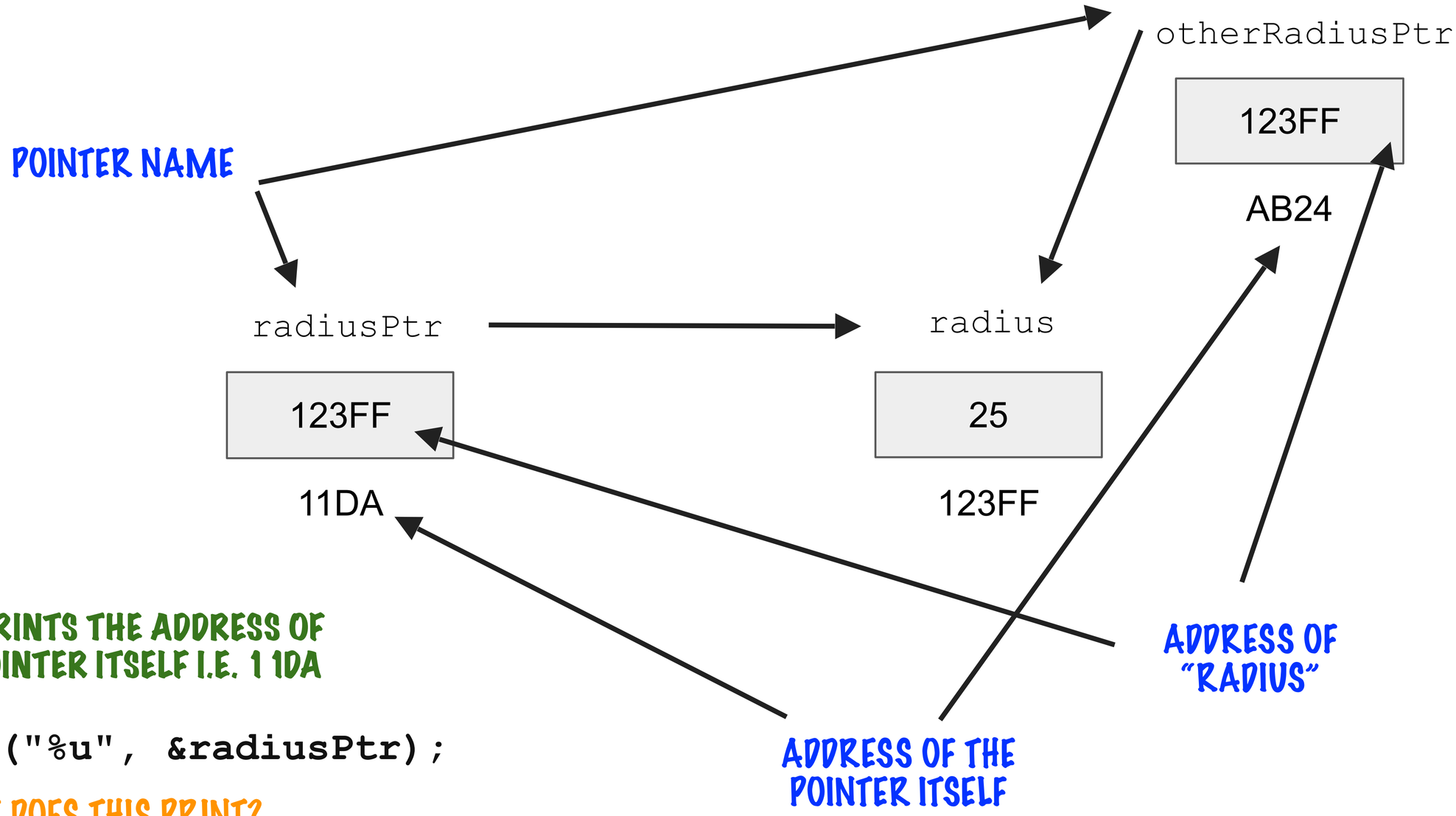
WORKING WITH POINTERS



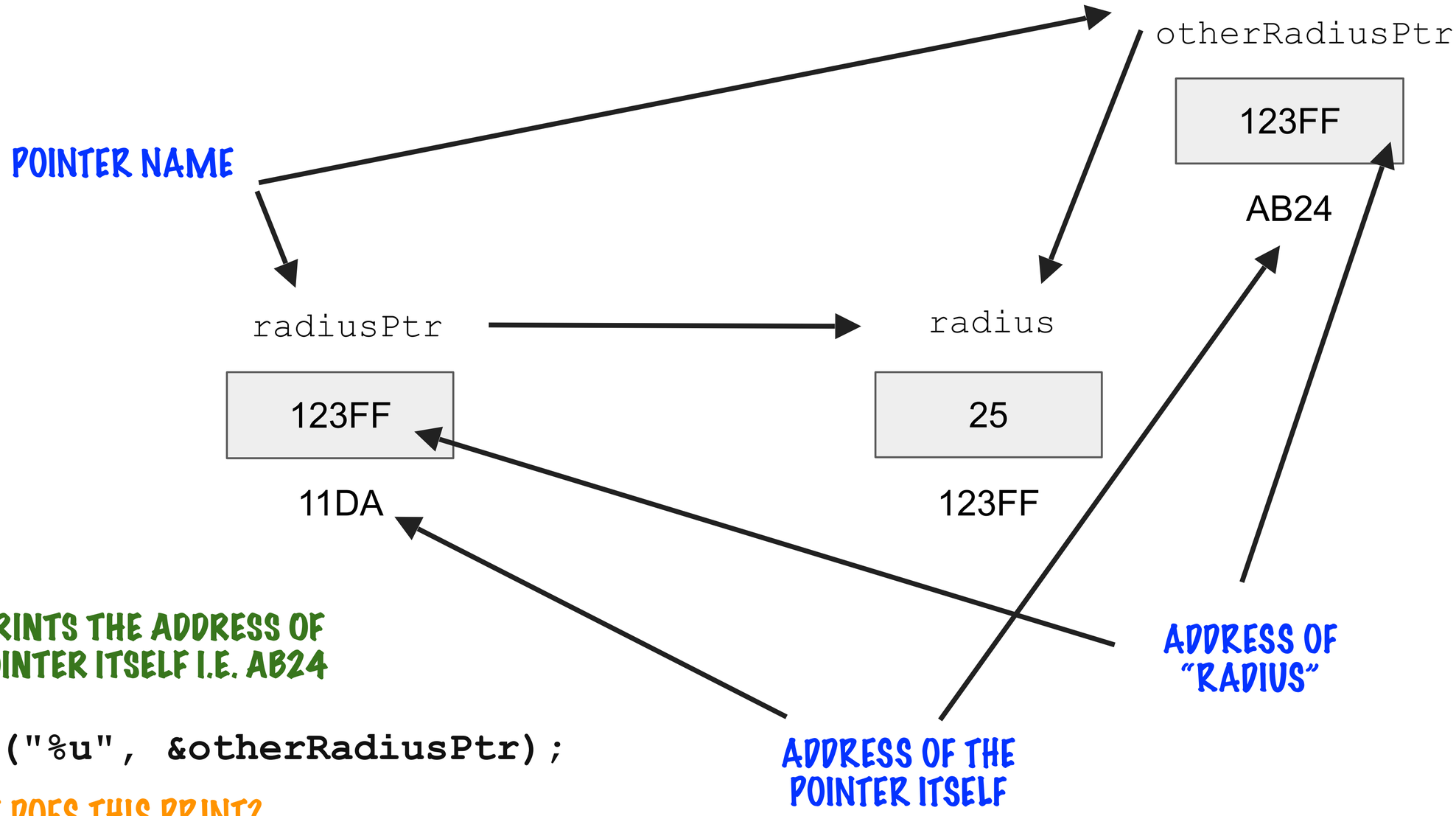

```
printf("%u", radiusPtr);
```

WHAT DOES THIS PRINT?

WORKING WITH POINTERS



WORKING WITH POINTERS




```
printf("%u", &otherRadiusPtr);
```

WHAT DOES THIS PRINT?

Arrays in C are actually pointers

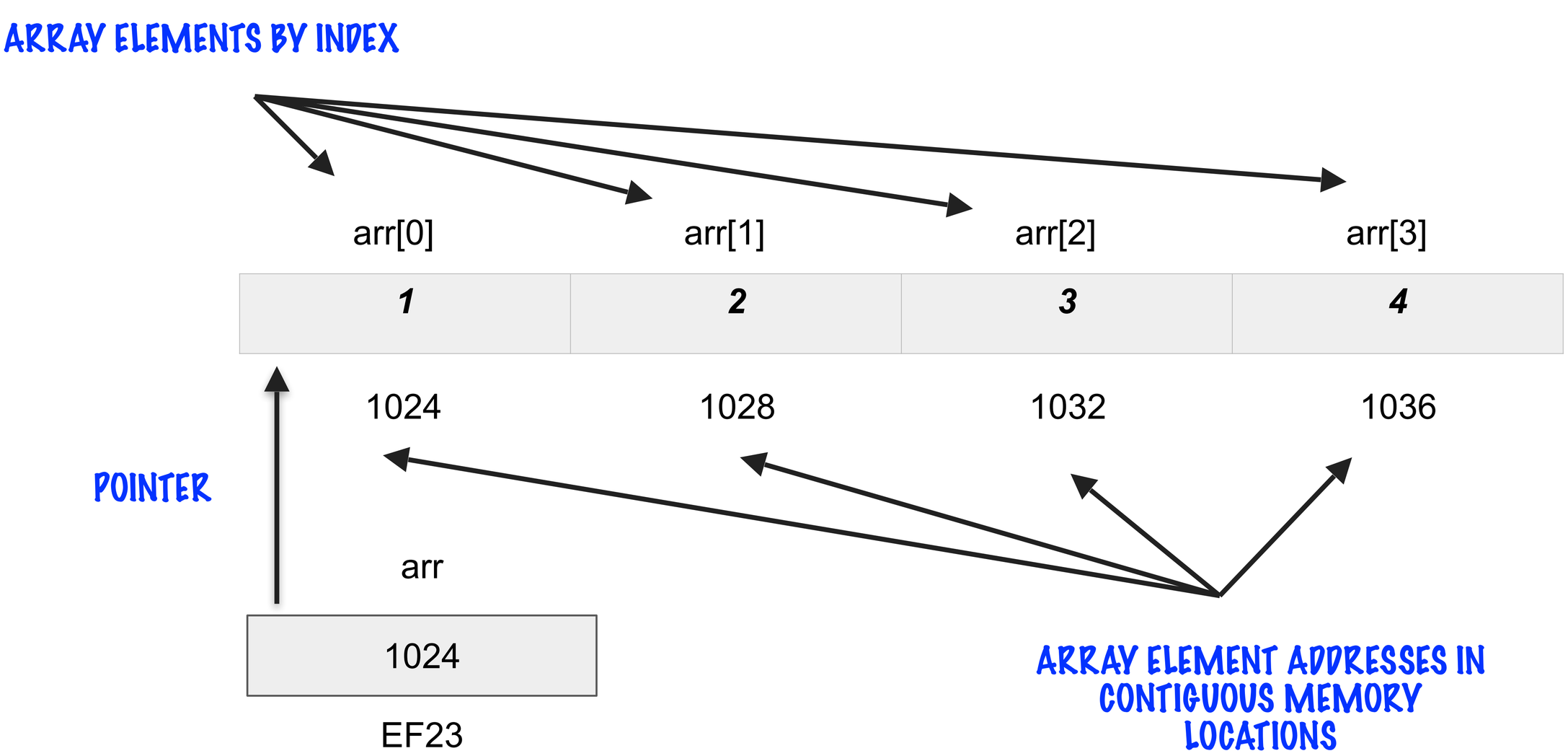
```
int arr[4] = { 1, 2, 3, 4 };
```

The way to access individual elements in the array is via indexes `arr[0]`…`arr[3]`

If you just refer to `arr` what kind of variable is it?

To answer this we must first see how arrays are laid out in memory

ARRAY LAYOUT IN MEMORY



Array variables are simply pointers

`arr` is a pointer to an integer and it points to the address location of the very first element in the integer array.

The type of `arr` is `int*`

Similarly if you have char array

```
char chararray[4] = { ‘a’, ‘b’, ‘c’, ‘d’ };
```

then chararray is of type **char***

```
int arr[4] = { 1, 2, 3, 4 };
```

“arr” will contain the address of the first element in the array which is `&arr[0]`

```
int *intptr = arr;
```

Let’s say we perform the increment operation on the variable “`intptr`” i.e. we call

```
intptr++;
```

What does this do?

This takes us squarely to pointer arithmetic