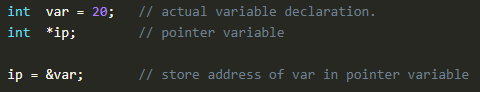
**Primitive Data Types**:

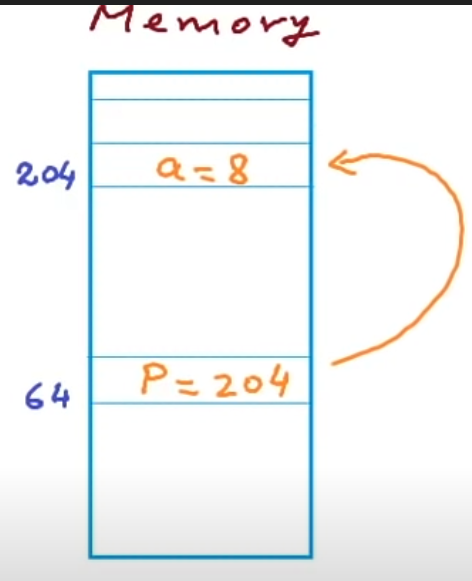
* Char = 1 byte || -128 to 127 or 0 to 255
* Bool = true/false
* Int = 4 bytes || -2147483648 to 2147483647
* Float = 4 bytes
* Double = 8 bytes

**Data type modifiers**:

* Signed
* Unsigned
* Short
* Long

**Pointers:** Pointer is a variable that stores address of another variable. Pointer variable must be same type as the variable it points to.





In this case,

int a=5;

int \*p; // or int\* p; this works too

p=&a;

now if we print the values will look like this

p // 204

&a // 204

&p // 64

\*p // 5 (this is knowns as dereferencing)

\*p = 8 (assigning 8 value to a via pointer)

**Pointer Operation**

1. **Incrementing/Decrementing:** Pointers points to the next/previous memory location of its type when incremented/decremented.

For example, int type pointer x points to location 200

* If we do x++ it will now point to location 204. If it was char type it would point to location 201.
* If we do x+3, it would point to location 212.
* process is similar for decrement.

1. **Subtracting one pointer from another:** If two pointer variables are of same array, we can subtract them. In this case it will give the number of elements in between the two pointer.

For example,

int num[10] = {1, 5, 9, 4, 8, 3, 0, 2, 6, 7};

int\* a=&num[2]; // points to location of 9 in array

int\* b=&num[6]; // points to location of 0 in array

a-b =4 // it will calculate the offset difference

\*a-\*b = 9 // normal operation

1. **Comparison of two pointers:** This one is kind of self-explanatory. We can compare for equality/inequality of two pointer (meaning if they point to same location or not).
2. **Not allowed Operations:** 
   * Addition of two pointer variables
   * Multiplication of a pointer with a constant value
   * Division of a pointer with a constant value

**Type cast:** If we want to assign different type of pointer to another pointer, we need to type cast. For example,

Int a=1025;

Int \*p=&a;

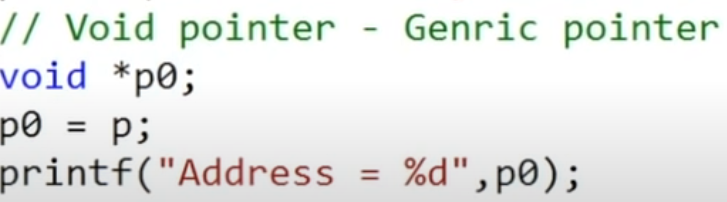
char \*p1 = (char\*)p;

now if we dereference p1, we will get its value 1(cause first byte of 1025 is 1)

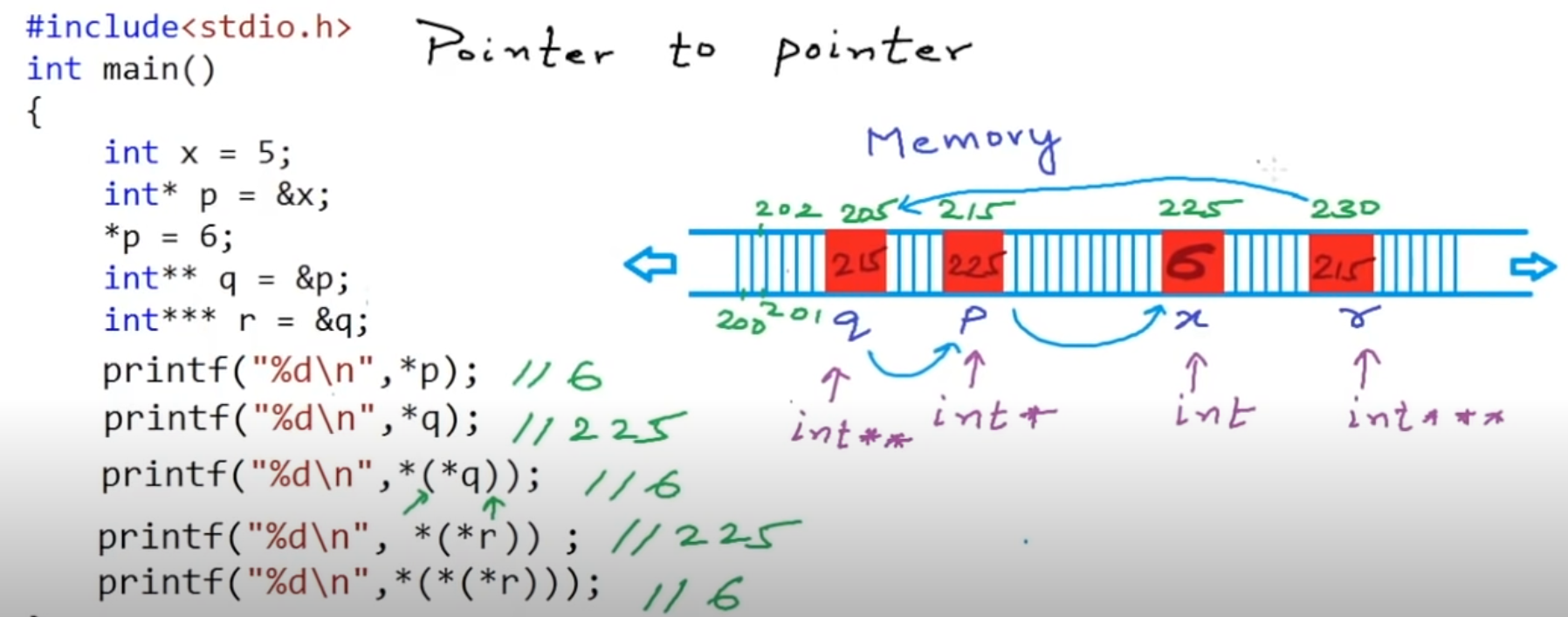


Now if we do p1++, we will get its value 4 when we dereference it.

**Void Pointer:** It is a generic type of pointer which doesn’t correspond to any particular data type. We don’t need explicit type casting when assigning other type of pointers to void. But we can’t dereference/perform arithmetic on the void pointers cause its not mapped to any particular data type.



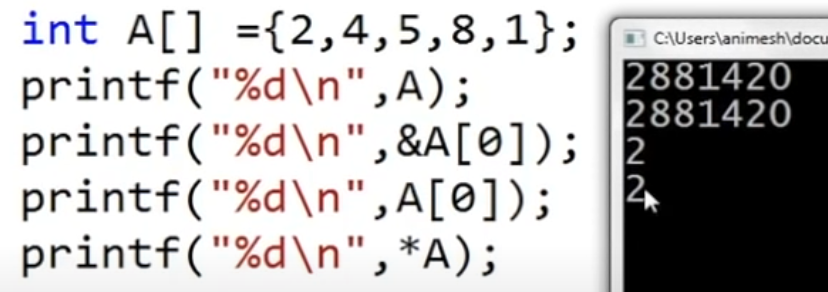
**Pointer to pointer:** the following picture demonstrate the pointer-to-pointer theory.



**Pointer and Array**

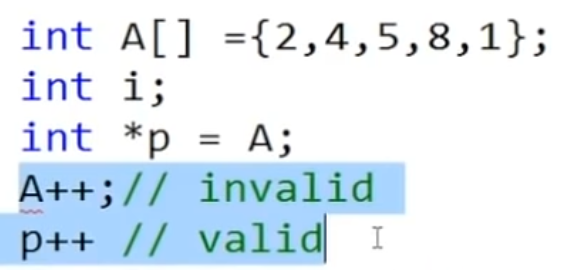
Array variable holds the starting address of the array.

* We can directly use it as a pointer. So, this is a special case, when we pass an array to a function, we pass it by reference always. For arrays there is no call by value, always call by reference.



**A** and **&A[0]** both points to base address. In turn, **A[0]** and **\*A** both will give first element.

* We can’t inc/dec A itself even though its reference to base. But we can do pointer operation once we assign A to another pointer (**A++** would be invalid).



**Pointer and Character Array**