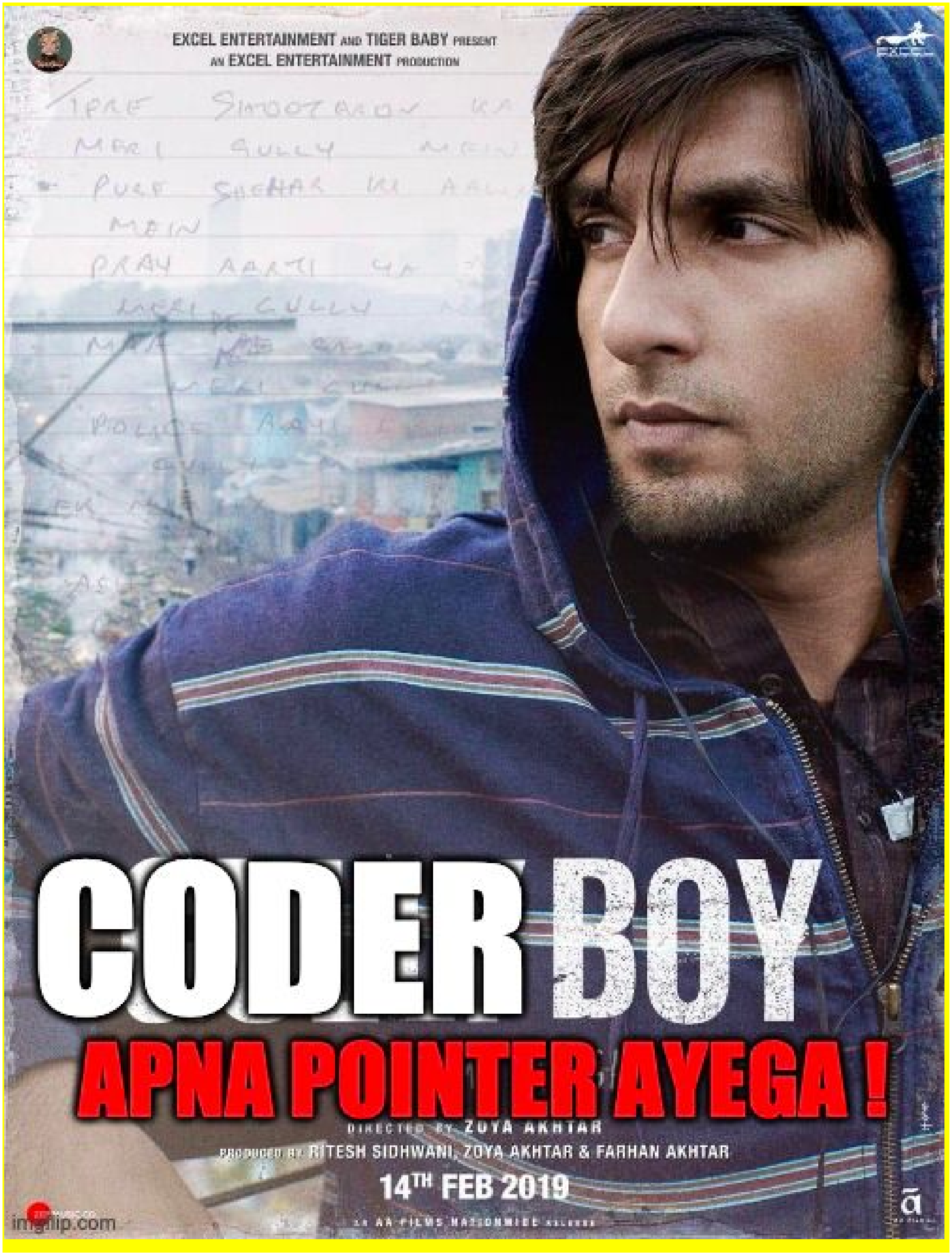
POINTERS 101

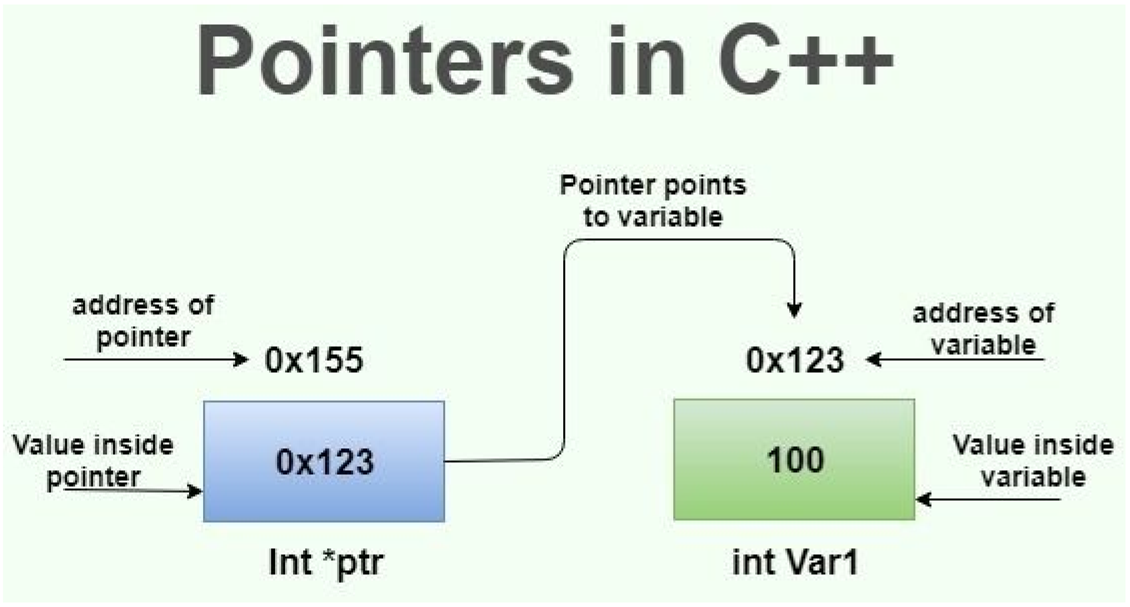

EXCEL ENTERTAINMENT AND TIGER BABY PRESENT
AN EXCEL ENTERTAINMENT PRODUCTION
CODER BOY
APNA POINTER AYEGA !
DIRECTED BY ZOYA AKHTAR
PRODUCED BY RITESH SIDHWANI, ZOYA AKHTAR & FARHAN AKHTAR
14TH FEB 2019
AA FILMS NATIONWIDE RELEASE
imgflip.com

POINTERS 101

What are pointers?

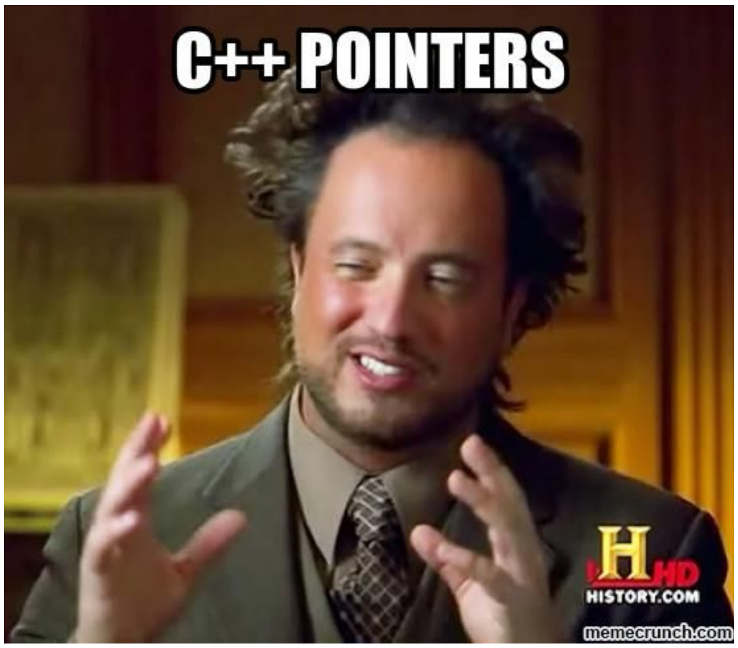
Pointers are one of the most powerful and confusing aspects of the C/C++ language.

```
datatype *var_name;
int *ptr;   //ptr can point to an address which holds int data
```



A pointer is a variable that holds the address of another variable. To declare a pointer, we use an asterisk between the data type and the variable name.

POINTERS 101



```
int *pnPtr; // a pointer to an integer value
double *pdPtr; // a pointer to a double value
int* pnPtr2; // also valid syntax
int * pnPtr3; // also valid syntax
```

Address of Operator (&)

Since pointers only hold addresses, when we assign a value to a pointer, the value has to be an address. To get the address of a variable, we can use the address-of operator (&)

```
int p = 5; int * q = &p; // assign address of p in q
```

Dereference Operator [*]

An interesting property of pointers is that they can be used to access the variable they point to directly. This is done by preceding the pointer name

with the dereference operator. The operator itself can be read as "value pointed to by”
Therefore the value pointed by q in previous example can be accessed as

```
int r = *q;
```

Null Pointer

Sometimes it is useful to make our pointers point to nothing. This is called a null pointer. We assign a pointer a null value by setting it to address 0.

```
double *p = 0;
```

Resource : https://www.learncpp.com/cpp-tutorial/6-7a-null-pointers/

Arithmetic Operators & Pointers

Addition, Multiplication, Division of two addresses doesn’t make any sense

Addition of an address by a constant integer value i.e. ptr +5 means address of cell which is 5 * sizeof(*ptr) away from ptr.

Similar for subtraction
Again Multiplying/Dividing an address with some constant value doesn’t make any sense

Subtracting two address of same type would give you number of elements between them

POINTERS 101

Arrays and Pointers

Just C/C++ things

Pointers and arrays are intricately linked in the C/C++ language.
An Array is actually a pointer that points to the first element of the array!
Because the array variable is a pointer, you can dereference it, which returns array element 0.

a[i] is same as *(a + i)

Its possible to pass part of an array to function.

Difference – Arrays & Pointers

<u>The sizeof operator:</u>
sizeof(array) returns the amount of memory used by all elements in array

POINTERS 101

sizeof(pointer) only returns the amount of memory used by the pointer variable itself

<u>The & operator</u>
&array is an alias for &array[0] and returns the address of the first element in array

&pointer returns the address of pointer String literal initialization of a character array

char array[] = “abc” sets the first four elements in array to ‘a’, ‘b’, ‘c’, and ‘\0′

char *pointer = “abc” sets pointer to the address of the “abc” string (which may be stored in read-only memory and thus unchangeable)

<u>Assignment/Re-assigment</u>
Pointer variable can be assigned a value whereas array variable cannot be.

<u>Arithmetic</u>

```
int a[10];
int *p;
p=a; /*legal*/ a=p; /*illegal*/
```

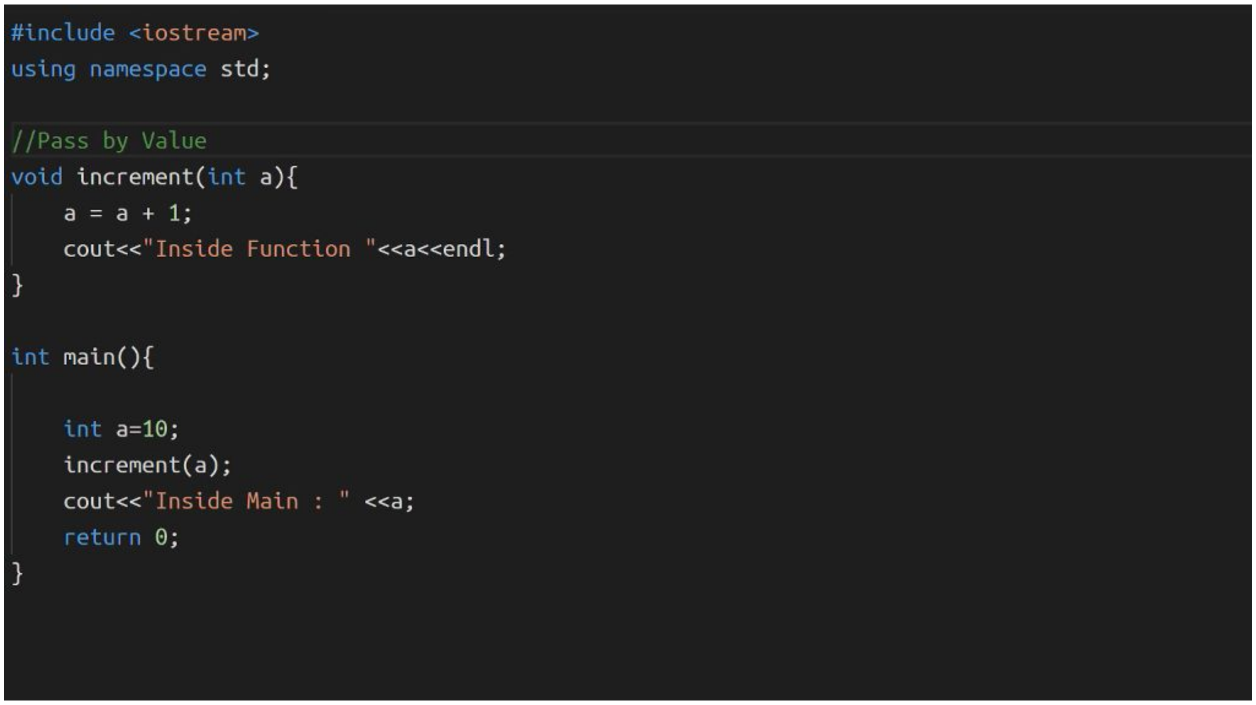
Arithmetic on pointer variable is allowed but array can’t be incremented/decremented.

```
p++; /*Legal*/
a++; /*illegal*/
```

USE CASE OF POINTERS : PASS BY REFERENCE

POINTERS 101

Lets understand what is Pass by Value first :

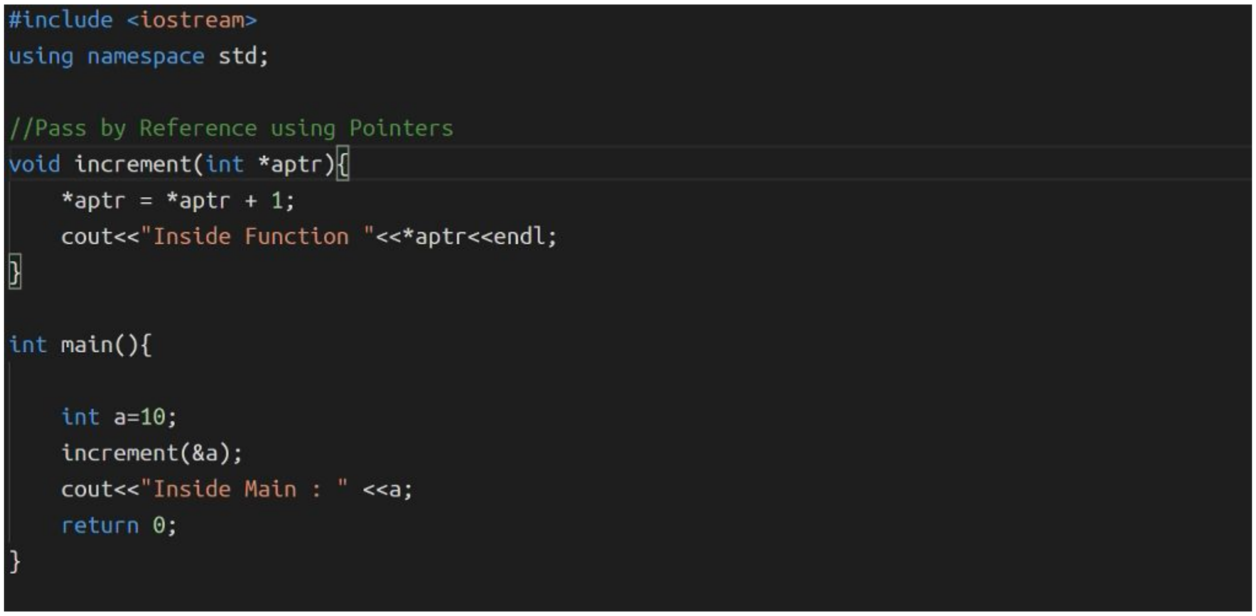


```cpp
#include <iostream>
using namespace std;

//Pass by Value
void increment(int a){
    a = a + 1;
    cout<<"Inside Function "<<a<<endl;
}

int main(){

    int a=10;
    increment(a);
    cout<<"Inside Main : " <<a;
    return 0;
}
```

Thus when we run this program , the a inside main is still 10 .Because the increment function is passed the value of a not its address /reference.

Now let us see how Pass by Reference works and how it is different from Pass by Value

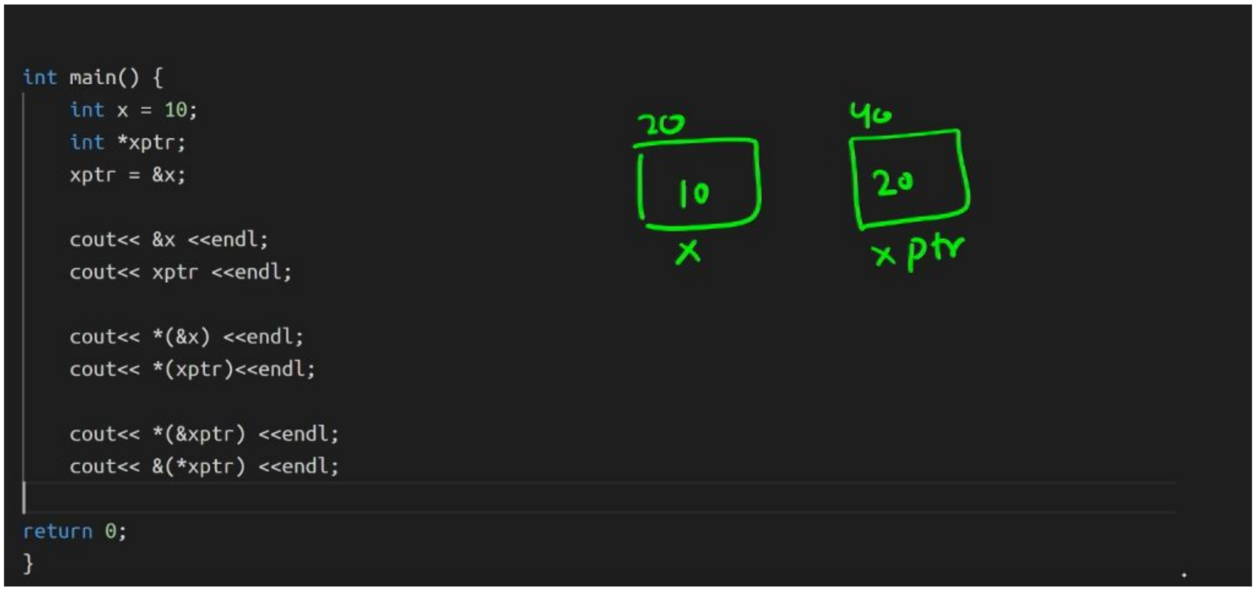
POINTERS 101



```
#include <iostream>
using namespace std;

//Pass by Reference using Pointers
void increment(int *aptr){
    *aptr = *aptr + 1;
    cout<<"Inside Function "<<*aptr<<endl;
}

int main(){

    int a=10;
    increment(&a);
    cout<<"Inside Main : " <<a;
    return 0;
}
```

Here address of a is passed from main to increment function .Hence any change done changes the original local variable in main .Thus a is 11 in main after the increment functionW call .

QUIZ TIME NOW



```
int main() {
    int x = 10;
    int *xptr;
    xptr = &x;

    cout<< &x <<endl;
    cout<< xptr <<endl;

    cout<< *(&x) <<endl;
    cout<< *(xptr)<<endl;

    cout<< *(&xptr) <<endl;
    cout<< &(*xptr) <<endl;

return 0;
}
```

POINTERS 101

Ans :
20
20
10
10
20
20

Resources for Pointers in C++ in depth

< Please make sure to finish the course exercises first before giving the resources mentioned below a try .
These resources are for in depth concept building and out of the scope of the course . >

https://www.javatpoint.com/cpp-pointers

http://www.cplusplus.com/doc/tutorial/pointers/

https://www.javatpoint.com/cpp-references

https://www.javatpoint.com/cpp-reference-vs-pointer

https://www.javatpoint.com/cpp-memory-management

POINTERS 101

http://cslibrary.stanford.edu/102/

NeXt : Introduction to OOPS