Pointers are variables storing the memory address of another data/variable/object.
Pointers are variables storing the memory address of another data/variable/object.

int x=5;
Pointers are variables storing the memory address of another data/variable/object.

- `int x=5;`
- `int *y = &x;`
- Operator `&` gives the memory address of a variable/object
Pointers are variables storing the memory address of another data/variable/object.

```
int x = 5;
int *y = &x;
Operator & gives the memory address of a variable/object
int* z = y;
```
- int *y = &x;
- int* z = y;
- What will happen if I write: *z = 0;?
int *y = &x;
int* z = y;
What will happen if I write: *z = 0;?
* is a dereferencing operator – gives the content of the memory address pointed by (or stored in) the pointer
MEMORY ALLOCATION – NEW OPERATOR

- new allocates space to hold the object.
- new calls the object’s constructor.
- new returns a pointer to that object.

- Point * A = new Point (10, 20);
For every call to new, there must be exactly one call to delete.

Point * A = new Point (10, 20); //allocates memory
...
delete A; //deallocates or frees the memory
An array can act as a pointer
Array name is a pointer to first element in the array
Pointer can be indexed like an array

```
int arr[5] = {1,2,3,4,5};
cout << arr[2] << "," << *(arr+2); //displays 3,3
```
DYNAMIC ALLOCATION & DEALLOCATION OF ARRAY

- Static allocation:
  - You must know the size of the array before hand
  - int arr[10];

- Dynamic Allocation
  - Size of the array can be passed as a variable
  - size_t sz = 10;
  - int* arr = new int[sz];
  - Deallocate : delete[] arr;
MEMORY

HEAP → MEMORY ← STACK
STACK VS. HEAP

- Heap – Dynamic Allocation
- Point *p = new Point(5,10);
- double *amount = new double[5];

- Stack – static allocation
- Point p(5,10);
- double amount[5];

What happens when p goes out of scope?
PAIR PROGRAMMING

- Driver – One at the keyboard
- Navigator – Helps direct driver
- Teamwork and communication are key here
- Switch roles frequently!