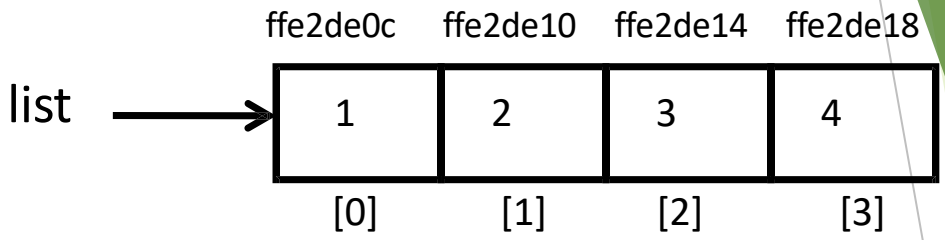
Pointers

Pointers and 1-D arrays

Array Name

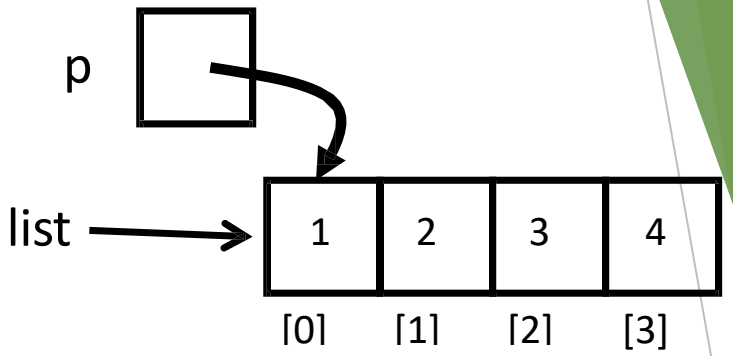
The array name is a pointer to the first element of the array




```
int list[] = {1,2,3,4};
printf("%x, %x, %d", list, &list[0], *list);
```

Output: **ffe2de0c ffe2de0c 1**

Pointers and Arrays

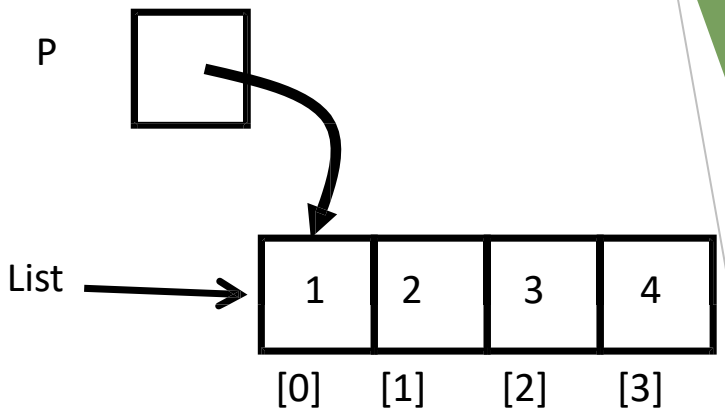



```
int *p,
int list[]={1,2,3,4};
p = list;                  /* equivalent to p = &list[0] */
printf("%d\n", *p);        /* prints the value "1"       */
```

You can use a pointer to access the array

Pointer and []

- Any pointer to a block of memory can use the [] syntax, even if it is not declared as an array!




```
int *p;
int list[]={1,2,3,4};
p = list;
printf("%d\n", p[2]);   // prints 3
```

Difference between pointer to array and pointer to int

```
int main()
{
    int *p; // pointer to int
    int (*parr)[5]; // pointer to an array of 5 integers
    int my_arr[5]; // an array of 5 integers

    p = my_arr;
    parr = my_arr;

    printf("Address of p = %u\n", p );
    printf("Address of parr = %u\n", parr );

    p++;
    parr++;

    printf("\nAfter incrementing p and parr by 1 \n\n");
    printf("Address of p = %u\n", p );
    printf("Address of parr = %u\n", parr );

    printf("Address of parr = %u\n", *parr );
     return 0;
}
```

output

- Address of p = 2293296
- Address of parr = 2293296
- After incrementing p and parr by 1
- Address of p = 2293300
- Address of parr = 2293316

Here p is a pointer which points to the 0th element of the array my_arr, while parr is a pointer which points to the whole array my_arr.

The base type of p is of type (int *) or pointer to int and base type of parr is pointer to an array of 5 integers.

Since the pointer arithmetic is performed relative to the base type of the pointer, that's why parr is incremented by 20 bytes i.e (5 x 4 = 20 bytes). On the other hand, p is incremented by 4 bytes only.