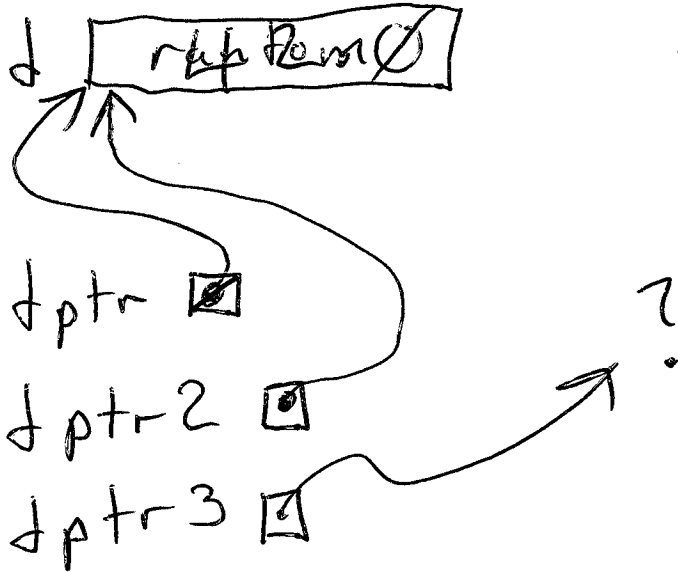


Pointers

Declare + Initialize

```
double d, *dptr = NULL, *dptr2 = &d, *dptr3;
```



`&`
address
operator

Assign

```
dptr = &d;
```

Use

```
*dptr = 42.0;
```

↑ reference

```
void copyString(char *s, char *d) {
```

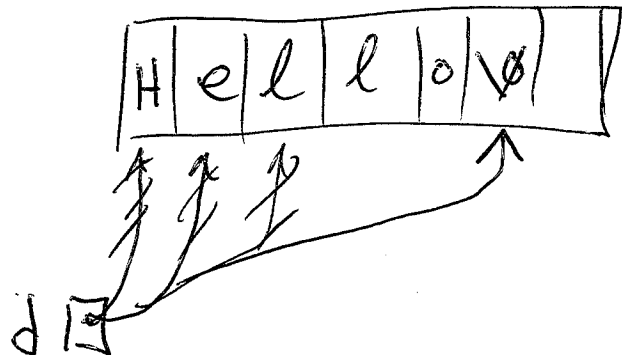
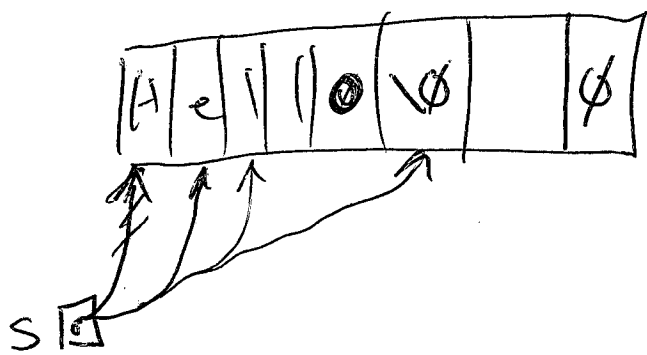
```
    while (*s)
```

```
        *d++ = *s++;
```

```
    *d = '\0';
```

```
    return;
```

```
}
```



`*d++` increments `d`

`(*d)++` " `*d`

`++*d`

`N = 3;`

`M = N++;`

↑
3

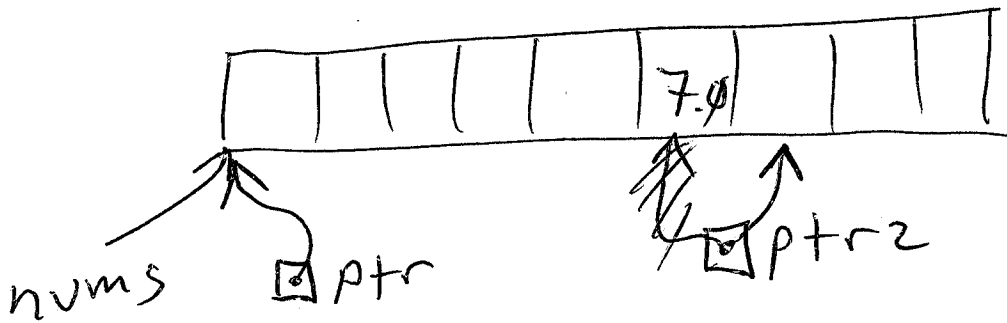
`N = 3;`

`M = ++N;`

↑
4

Arrays

double nums[10];



nums[5] = 7.0;

nums[i] is a double
nums is an address

ptr = nums;

ptr2 = &nums[5];

Pointer math is in units of data type size

nums ~~ptr~~[i] = 5;

* (ptr + i) = 5;

* (nums + i) = 5;

ptr[i] = 5;

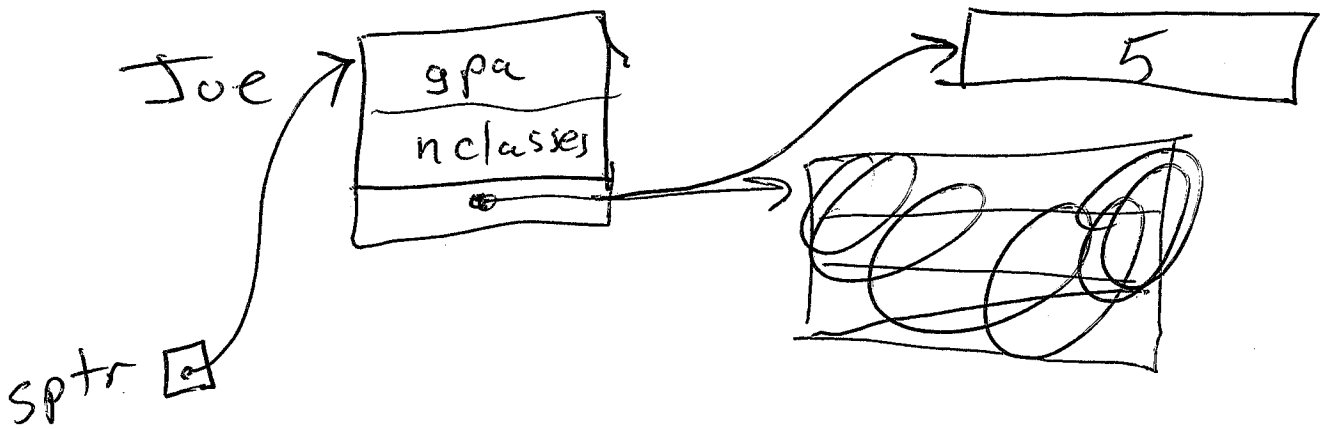
ptr2 ++;

} same

student Structs

```
struct { double gpa;  
        int nclasses,  
char student int *n };
```

```
(struct student) Joe,  
*sptr = &Joe;
```

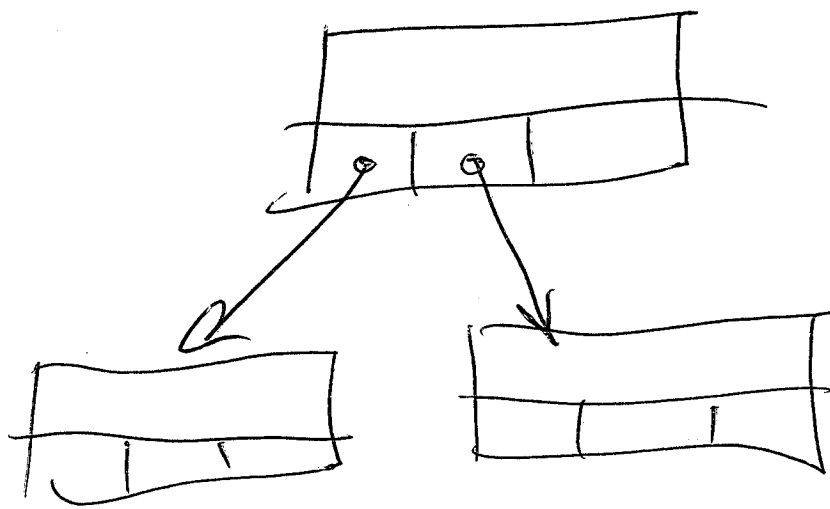
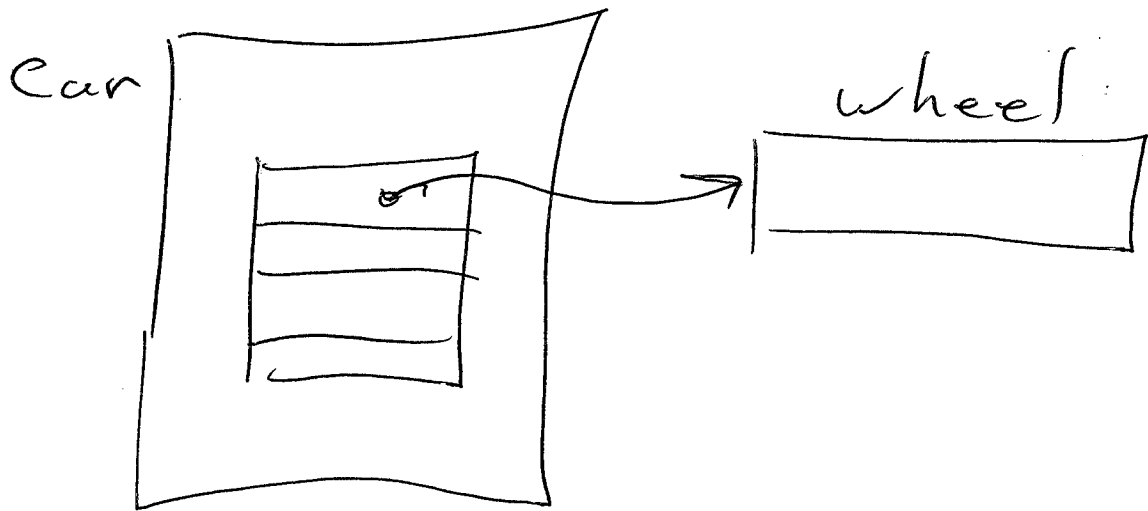


```
Joe.gpa = 4.0;
```

```
(*sptr).gpa = 4.0;
```

```
sptr -> gpa = 4.0;
```

```
*Joe.nclasses = 5;
```

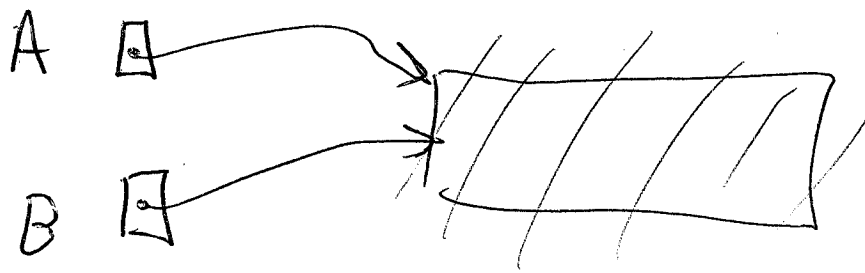


```
typedef struct { int wings } Plane;
```

```
Plane p;
```

```
struct Car { int doors };
```

```
Car c;
```



free(A)

A = NULL;