Structs are collections of variables under one name.

```
struct student {
    char name[15];
    char age;
    char credit_hours;
};
```

← does not create an instance of the thing

← note semicolon

```
struct student bertha = { {'B','e',...'a',0}, 18, 27 };
struct student bertha = { "Bertha", 18, 27 };
```

terminating zero

same

↑ string constant

↑ an instance of struct student

new operator

```
bertha.name ← "Bertha"
bertha.age  ← 18

printf("%s", bertha.name);

st = bertha
printf("%s", st.name);
```

→ `struct student studs[20] = { ... }`

average age of all the students

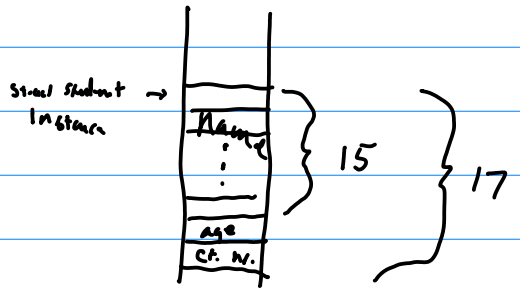
```
int sum = 0;
for (i=0; i < numstud; i++) {
    sum += studs[i].age;
}
int average = sum / numstud;
```

```
struct class {
    char num_students;
    struct student studs[30];
};


int getAvgAge(const struct class *c) {
    int sum = 0;
    for(int i=0; i<c->num_students; i++) {
        sum += c->studs[i].age;
    }
    int average = sum/c->num_students;
    return average;
}
```

---

c	← type	pointer to a const struct class
*c		const struct class
(*c).num_students		int
same → c->num_students		

```
struct student {
    char name[15];
    char age;
    char credit_hours;
} bertha {};

struct student ralph, rex;

struct class {
    char num_students;
    struct student studs[30];
} myclass;
```



c->studs

	type	# bytes
ralph	struct student	17
ralph.age	char	1
ralph.name	array of 15 char	15
myclass	struct class	1 + 30*17
myclass.studs	array of 30 struct student	30*17
myclass.studs[0] → *myclass.studs (same)	struct student	17
myclass.studs[0].name	array of 15 char	15
myclass.studs[0].name[2]	char	1

```
struct class *pclass = &myclass;
```

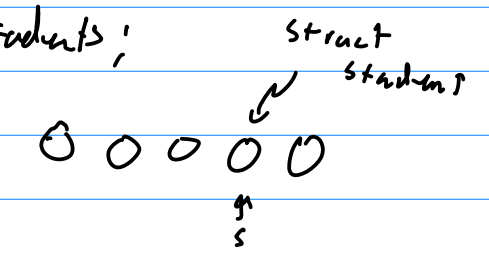
X pclass.num_students Wrong

X *pclass.num_students Wrong

(*pclass).num_students ✓

pclass->num_students ✓

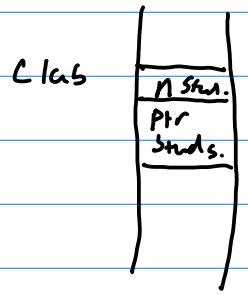
```
int getAvgAge (const struct class * c) {
    int sum = 0;
    for (int i = 0; i < c->num_students; i++) {
        sum += c->studs[i].age;
    }
    int average = sum / c->num_students;
    return average;
}
```




```
int getAvgAge (const struct class * c){
    int sum = 0;
    struct student *s = 0
    for (s = c->studs; s - c->studs < c->num_students; s++){
        sum += s->age;
    }
    int average = sum / c->num_students;
    return average;
}
```

```
struct club {
    char numstudents;
    struct student * studs;
} myclub, aclub[10], *pclub;
```

← initialized elsewhere




```
struct student threestuds[3] = {{"Bertha", 18, 21}, {"Rex", 19, 16}...}
struct club myclub2 = { 3, threestuds }
```

myclub # bytes = 1 + 4

* assuming pointer is 4 bytes

```
Struct club myclub3;
 myclub3.num_students = 50;
 myclub3.studs = calloc (50, sizeof (struct student));
                 malloc (50 *sizeof (struct student));   <- same

  free ( myclub3.studs );
```

```
Struct club *pclub = 0.
 pclub = calloc (1, sizeof (struct club));
pclub->num_students = 5;
pclub->studs = calloc (5, sizeof (struct student));

  for (int i=0; i<5; i++) {
     readfromfile (&pclub->studs[i]);
  }

  free ( pclub->studs );   <- in this order
  free ( pclub);
```

```
struct student {
    char name[15];
    char age;
    char credit_hours;
} bertha[3];

struct club {
    char numstudents;
    struct student *studs;
} myclub, aclub[10], *pclub;
```

17 bytes

pointer = 4 bytes

1 + 4 } 5 bytes

	type	# bytes
myclub	struct club	5
myclub.studs	pointer to struct student	4
same → * myclub.studs	struct student	17
→ myclub.studs[0]	struct student	17
(*myclub.studs).age	char	1
(*myclub.studs).name	array of 15 char	15
(*myclub.studs).name[2]	char	1

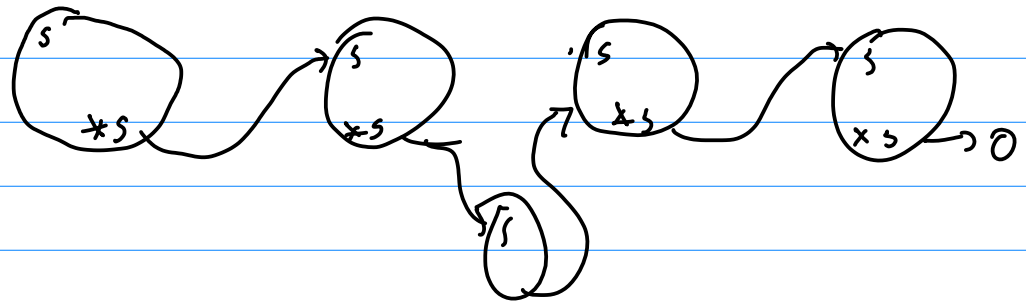
pclub	pointer to a struct club	4
pclub -> studs	pointer to a struct student	4
* pclub -> studs	struct student	17

aclub	array of 10 struct clubs	50 bytes

typedef

```
typedef struct song * Songlist;

Songlist mysongs;

typedef struct song Song;

typedef struct {
    char numnotes;
    struct note thenotes[30];
} Song;

Song stejrsong;
```

Structures cannot cannot contain an instance of themselves but may contain a pointer to an instance of themselves.



Only operations on structs:

assign to another of the same type

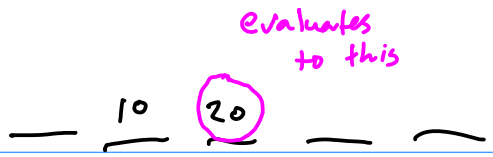
```
ralph = bertha;
```

Take the address of &

Access the members . or ->

sizeof operator

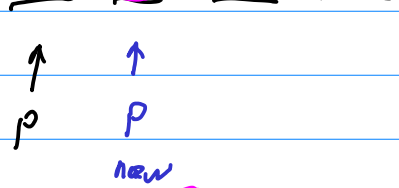
int *p;

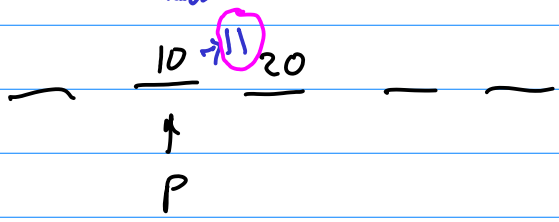


*++p

p increments

p+1

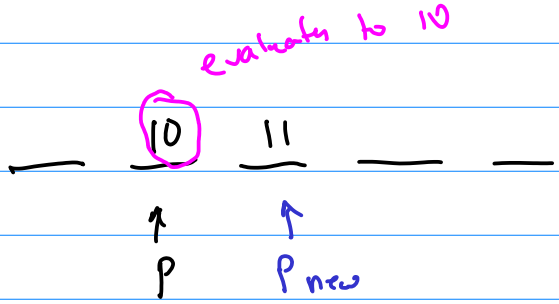




++*p

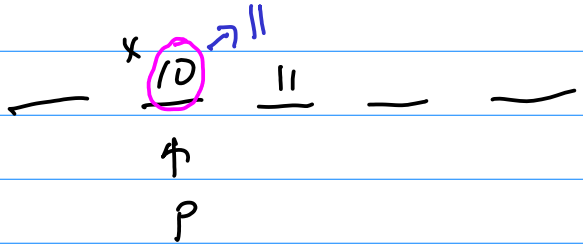
10

evaluates



⇒ *p++

p increments



(*p)++

x++