

Arrays of Structs

```
struct student class[50];  
  
strcpy(class[0].name, "Alice");  
class[0].age = 20;  
class[0].gpa = 4.0;  
  
strcpy(class[1].name, "Bob");  
class[1].age = 19;  
class[1].gpa = 3.1  
  
strcpy(class[1].name, "Cat");  
class[1].age = 21;  
class[1].gpa = 3.4
```

```
struct student {  
    char name[20];  
    int age;  
    float gpa;  
};
```

class:	'A'	'l'	'i'	'c'	'e'	'\0'	...	'B'	'o'	'b'	'\0'	...	'C'	'a'	't'	'\0'	...
	20							19					21				
	4.0							3.1					3.4				
	[0]												[1]				

Arrays of structs...think about type!

```
int main() {  
    struct student class[50];  
  
    strcpy(class[0].name, "Jo");  
    class[0].age = 18;  
    class[0].gpa = 3.4;  
    class[1] = class[0];    // structs are lvalues  
    class[1].name[0] = 'M';  
    class[1].gpa = 2.8;  
    strcpy(class[2].name, "So");  
    class[2].age = 20;  
}
```

	[0]				[1]				[2]				...
class:	'J'	'o'	'\0'	...	'M'	'o'	'\0'	...	'S'	'o'	'\0'	...	
	18				18				20				
	3.4				2.8								

Arrays of Structs

```
struct student {
    char name[20];
    int age;
    float gpa;
};

struct student classroom[50];

strcpy(classroom[0].name, "Alice");
classroom[0].age= 20;
classroom[0].gpa = 4.0;

// With a loop, create an army of Alice clones!
int i;
for (i = 0; i < 50; i++) {
    strcpy(classroom[i].name, "Alice");
    classroom[i].age=20;
    classroom[i].gpa = 4.0;
}
```

Arrays of structs parameters:

```
void test(struct student a[], int n) {  
    a[0].age = 20;  
}
```

```
int main() {  
  
    struct student class[50];  
    ...  
    test(class, 3);  
}
```

Changing value stored in bucket of an array parameter
(a[0].age = 20), changes the corresponding bucket value in
argument (class[0].age):

a and class refer to the same memory location

Arrays of structs parameters:

test:

a



main:

class:

0				1				
'J'	'o'	'\0'	...	'M'	'o'	'\0'
18				18				
3.4				2.8				

STACK