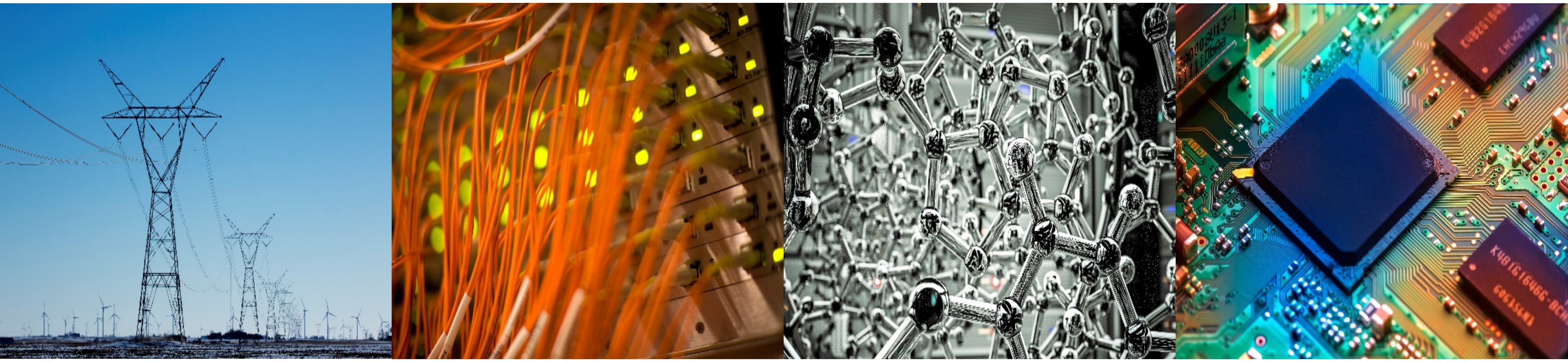
ECE 220 Computer Systems & Programming

Lecture 24 – Data Structures & Dynamic Memory Allocation

July 20, 2020



ILLINOIS

Electrical & Computer Engineering

GRAINGER COLLEGE OF ENGINEERING

- **Schedule MT2 with CBTF**

Pointer to Struct

```
student ece220[200];
student s1;
student *s_ptr, *s_ptr2;
s_ptr = ece220; /* pointer to a struct array */
s_ptr2 = &s1; /* pointer to a struct */

strncpy(s_ptr->Name, “Jane Doe”, sizeof(s1.Name));
s_ptr->UIN = 123456789;
s_ptr->GPA = 3.89;
```

- Which student record has been changed?

```
s_ptr++;
```

- where is s_ptr pointing to now?

- What is the difference between the following function calls?

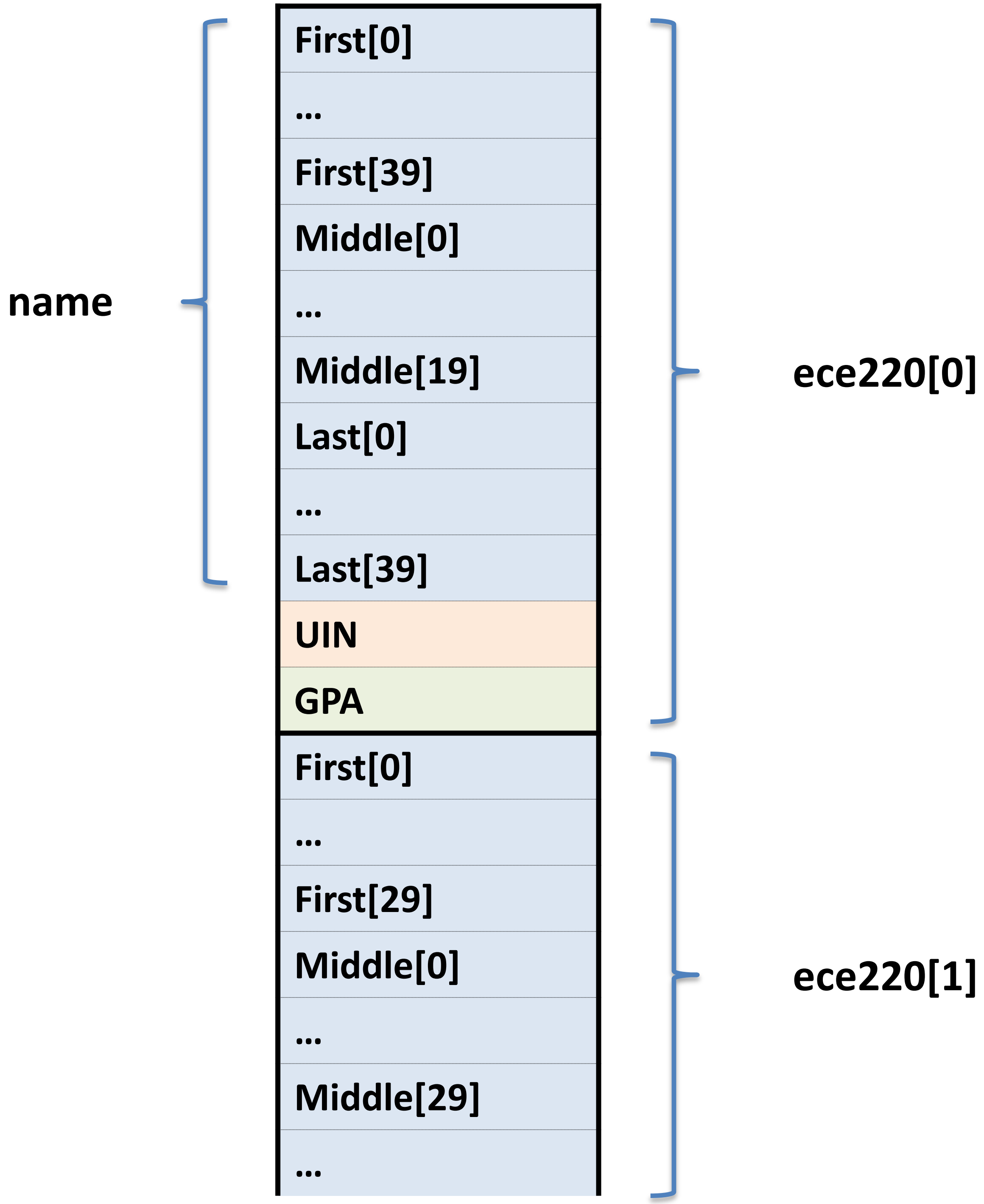
```
printname(s1);
PRINTNAME(&s1);
```

Struct within a Struct

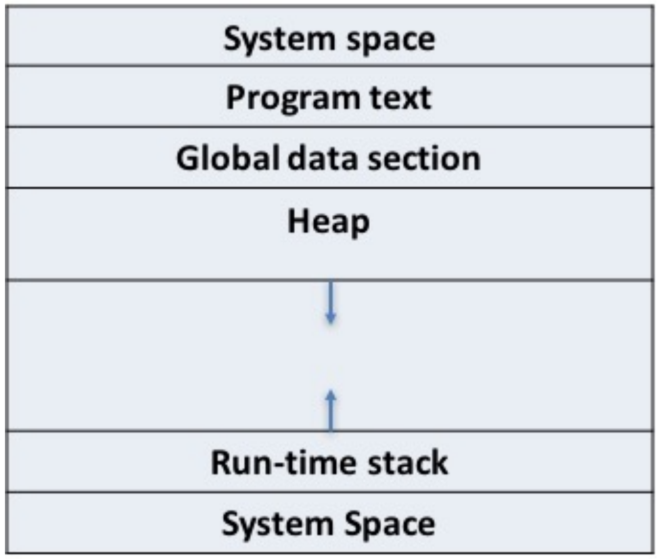
```
typedef struct StudentName{
   char First[40];
   char Middle[20];
   char Last[40];
}name;

typedef struct StudentStruct{
   name Name;
   int UIN;
   float GPA;
}student;

student ece220[200];
student *ptr;
ptr = ece220;
```

➢ How can we set the ‘First’ name in the first student record?

```
strncpy(                        , “Jane”,                              );
```


name
First[0]
…
First[39]
Middle[0]
…
Middle[19]
Last[0]
…
Last[39]
UIN
GPA
First[0]
…
First[29]
Middle[0]
…
Middle[29]
…
ece220[0]
ece220[1]

“Static” vs. Dynamic Memory Allocation



	“Static”	Dynamic
Mechanism of allocation		
Lifetime of memory		
Location of memory		
Size of allocation		

malloc & free

```
void *malloc(size_t size);
```

- allocates a contiguous region of memory on the heap
- size of allocated memory block is indicated by the argument
- returns a generic pointer (of type void *) to the memory, or NULL in case of failure
- allocated memory is not clear (there could be left over junk data!)

```
void free(void *ptr);
```

- frees the block of memory pointed to by ptr
- ptr must be returned by malloc() family of functions

Example using malloc & free:

```
int *ptr = (int *)malloc(sizeof(int));
if(ptr == NULL){
      printf("ERROR – malloc failure!");
      return 1;}
*ptr = 10;
free(ptr);
```

- How can we dynamically allocate space for an integer array with 10 elements?

- What is happening in this block of code?

```
int *ptr = (int *)malloc(sizeof(int));
*ptr = 5;
int *ptr_2 = (int *)malloc(sizeof(int));
*ptr_2 = 6;
ptr = ptr_2;
```

Exercise:

```
typedef struct studentStruct{
   char *NAME;
   int UIN;
   float GPA;
}student;
```

1. Dynamically allocate memory for 200 student records (**hint: you will also need to allocate an array of 100 chars to hold the name for each record**)
2. Initialize name to “To be set”, UIN to -1 and GPA to 0.0 for all 200 records