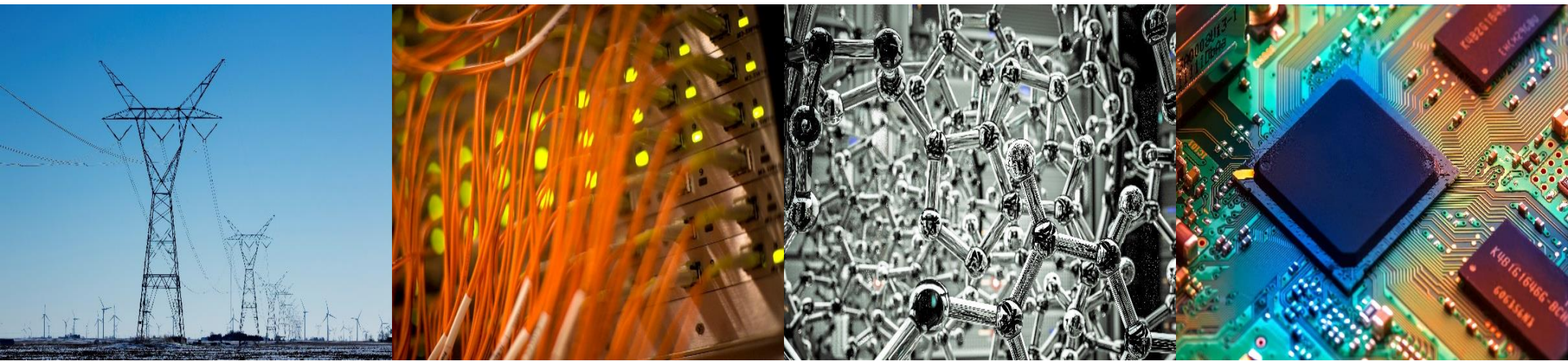


ECE 220 Computer Systems & Programming

Lecture 15 – Data Structures

October 16, 2025



- MT2 is scheduled for 10/30

I ILLINOIS

Electrical & Computer Engineering

GRAINGER COLLEGE OF ENGINEERING

The type journey

Objects

struct *

struct []

struct, typedef, enum

int *, char *, float *

int[], char[], float[]

int, char, float

Data Types

Fundamental data types: int, float/double, char

Arrays & Pointers

Enumerated data type:

- user-defined
- a list of related items (enumeration constants)
- count starts at 0 by default

```
/* Example: create an enumerated data type for days of the week */
```

```
enum Days {SUN, MON, TUE, WED, THU, FRI, SAT};
```

```
enum Days today;
```

```
today = THU;
```

➤ What is the value of today?

➤ What if we define 'Days' this way?

```
enum Days {SUN=1, MON, TUE, WED, THU, FRI, SAT};
```

Structures

- allow users to define a new type consists of a combination of fundamental and pre-defined data types (**aggregate** data type)

```
/* Example: use a struct to store student record */
```

```
struct StudentStruct{  
    char Name[100];  
    int UIN;  
    float GPA;  
};
```

```
struct StudentStruct student_var;  
strncpy(student_var.Name, "Jamie", sizeof(student_var.Name));  
student_var.UIN = 123456789;  
student_var.GPA = 3.89;
```

```
/* use just one line to declare and initialize variable */
```

```
struct StudentStruct student_var2 = {"Alex", 987654321, 3.98};
```

Using typedef

```
struct StudentStruct{
    char Name[100];
    int UIN;
    float GPA;
};
typedef struct StudentStruct student;
student s1, s2;
```

```
/* use 'typedef' in a slightly different way */
typedef struct StudentStruct{
    char Name[100];
    int UIN;
    float GPA;
}student;
student s1, s2;
```

Unions

- similar to struct, but members of the union share the same memory location(s)

```
typedef union StudentUnion{  
    char Name[100];  
    int UIN;  
    float GPA;  
}studentU;
```

➤ What is the size of studentU?

➤ What would happen if we execute the following code?

```
studentU su1;  
su1.UIN = 123456789;  
su1.GPA = 3.89;
```

Arrays of Structs

```
/* create an array of student struct */  
student ece220[200];
```

```
/* access each element of the array */  
ece220[0]  
ece220[1]
```

```
/* access individual fields in each element */  
ece220[0].Name[0] = 'J';  
ece220[0].Name[1] = 'a';  
ece220[0].Name[2] = 'm';  
ece220[0].Name[3] = 'i';  
ece220[0].Name[4] = 'e';  
ece220[0].UIN = 123456789;  
ece220[0].GPA = 3.89;
```

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, "Jamie", 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[32];  
    char Middle[32];  
    char Last[36];  
}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(                                , "Jamie",                                );
```

Name

First[0]

...

First[31]

Middle[0]

...

Middle[31]

Last[0]

...

Last[35]

UIN

GPA

First[0]

...

First[31]

Middle[0]

...

Middle[31]

...

ece220[0]

ece220[1]