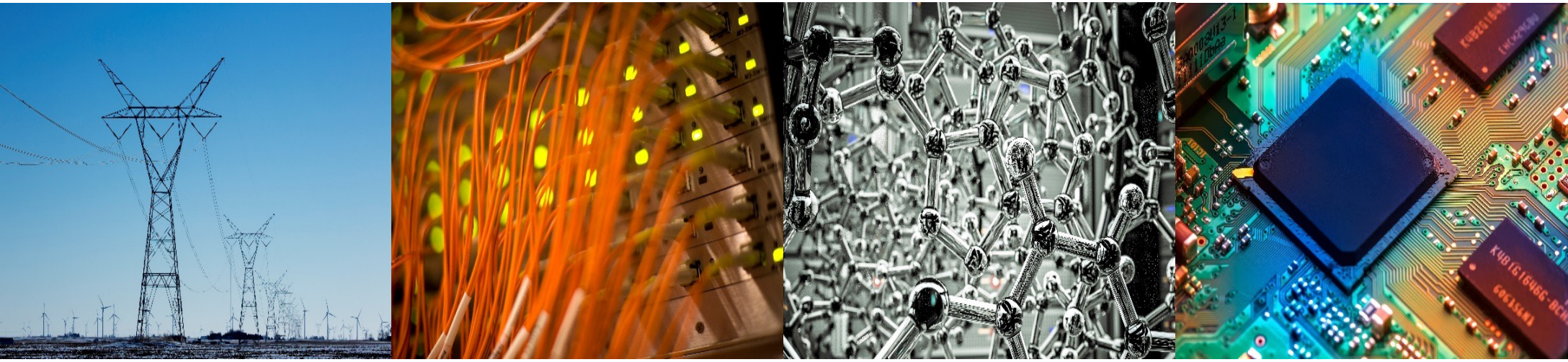
ECE 220 Computer Systems & Programming

Lecture 10 – Strings and Multidimensional Arrays

February 19, 2026



- **MT1 is next Thursday, conflict request deadline is Sunday, 2/22**

ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

Pointers Recap

- **Pass by Value** vs. **Pass by Reference** in C (assume x and y are declared as *int*)


```
swap(x,y);    swap(________,________);
```

- **Double Pointer**: a pointer to another pointer

```
int var=3;

int *var_ptr;

int **var_pptr;

var_ptr = ________________;

var_pptr = ________________;
```

Arrays Recap

- Array of size **n** has indices **0, 1, … , n-1**
- Arrays are **passed by reference** (pointer to the first element) in C
- Importance of array **bounds checking**

```
int array[2] = {3,5};
int *ptr = array;
int i;
for (i=0; i<2; i++, ptr++){
   *(ptr + 1) = *ptr + 1;
}
```

➢ What will be modified after execution of for loop?

Pointer Array Duality

```
char word[4];
char *cptr;
cptr = word;
```

	Using pointer ***cptr***	Using pointer notation on ***word***	Using array notation on ***word***
Address of the ***nth*** element in array *word*	**(cptr+n)**	**(word+n)**	**&word[n]**
Value of the ***nth*** element in array *word*	***(cptr+n)**	***(word+n)**	**word[n]**

```
int buffer[4] = {1,3,5,7};
int *ptr = buffer;
*(buffer+1) = 9;
ptr[3] = 11;
```

- What is the value of each element in ***buffer*** now?

Exercise: implement a function to reverse an integer array, such that the first element will become the last element, the second element will become the second to last element, and so on. This function takes two arguments: a pointer to an integer array and its size.

```
void array_reversal(int array[], int n){

}
```

Strings

- Allocate space for a string just like any other array

```
char outputString[16];
```

- Space for string must contain room for **terminating zero – '\0'**

- Special syntax for initializing a string:

```
char outputString[16] = "Result = ";
```

which is the same as:

```
outputString[0] = 'R';
outputString[1] = 'e';
outputString[2] = 's';
...
```

I/O with Strings

- **printf and scanf use "%s" format character for string**

- **printf:** print characters up to terminating zero

```
char outputString[20] = ”ECE 220 Roster";
printf("%s", outputString);
```

- **scanf:** read characters until whitespace, store result in string, and terminate with zero

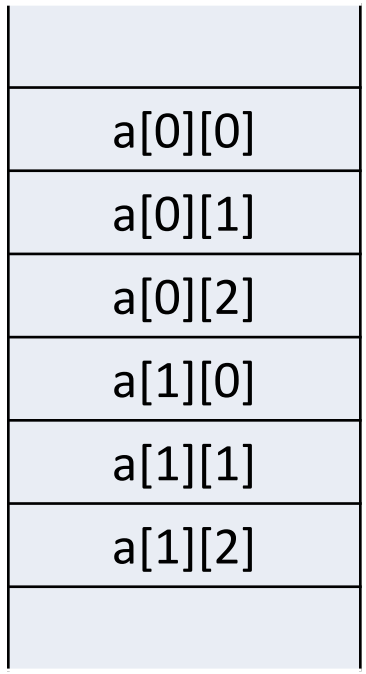
```
char inputString[20];
scanf("%s", inputString);
```

Multi-dimensional Arrays

int a [2][3];

	Column 0	Column 1	Column 2
Row 0	a[0][0]	a[0][1]	a[0][2]
Row 1	a[1][0]	a[1][1]	a[1][2]

In memory



❖ multi-dimensional arrays are stored in **row-major** order in memory

Declare and Initialize Multi-dimensional Array

```
/* initialize array with both row and column size */
int a[2][3] = {{1, 2, 3}, {4, 5, 6}};

/* initialize array with only column size */
int a[][3] = {{1, 2, 3}, {4, 5, 6}};

/* initialize array without specifying elements in each row */
int a[2][3] = {1, 2, 3, 4, 5, 6};
```

- Assume that array **a** has been declared first, can we reinitialize elements in this array using the {} syntax shown above?

Exercise: implement a function to exchange two rows of an MxN integer matrix. This functions takes three arguments: pointer to the matrix, row index x and row index y.

```
#define M 3 /* row size */
#define N 4 /* col size */
void row_exchange(int matrix[M][N], int row_x, int row_y){




}
```