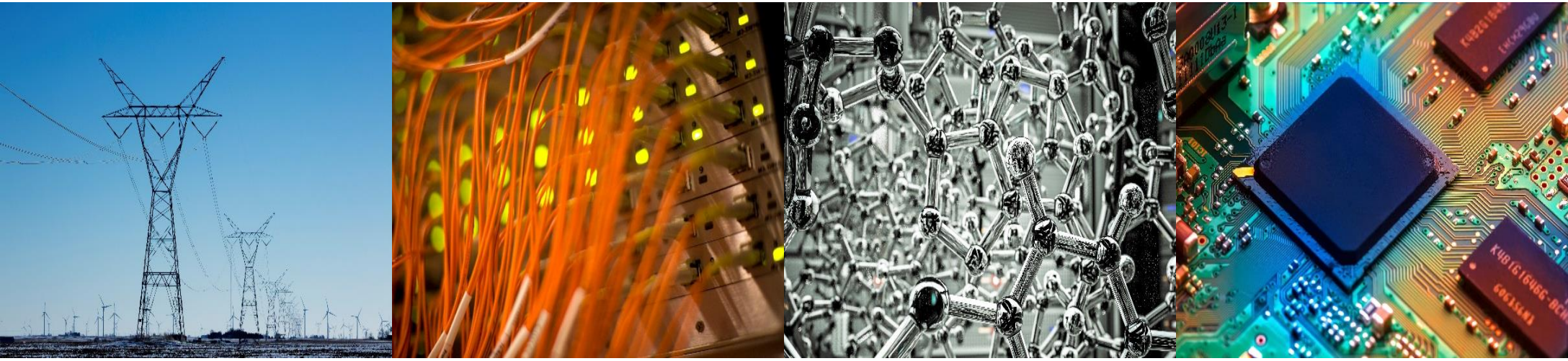
ECE 220 Computer Systems & Programming

Lecture 9 – Pointers and Arrays

September 24, 2024



- MT1 is this Thursday, 9/26; lecture will be cancelled
- Quiz2 should be taken next Monday through Wednesday @ CBTF

ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

Pointers and Arrays

Pointer

- Address of a variable in memory
- Allows us to indirectly access variables (in other words, we can talk about its **address** rather than its **value**)

Array

- A list of values arranged *sequentially* in memory
- Example: a list of telephone numbers
- Indices **start from 0**: **a[4]** refers to the 5th element of the array **a**

Pointers in C

Declaring Pointers

```
int *ptr;
char *cptr;
double *dptr;
```

Using address and dereference operators

& (address operator): &variable - returns the address of variable

***** (dereference operator): *ptr - returns the value pointed to by ptr

Example:

```
int variable = 4;
int *ptr;
ptr = &variable;
*ptr = *ptr + 1;
```

➢ What will be the value of **variable** and **ptr** at the end?

More on Pointers

NULL pointer (a pointer that points to nothing)

```
int *ptr;
ptr = NULL;
```

Syntax for using pointers

1. Declaring a pointer

```
type *ptr1;
type* ptr2;
```

2. Creating a pointer from a variable

```
type var;
type* var_ptr = &var;
```

3. Dereferencing a pointer

```
*var_ptr
**var_pptr
```

Arrays

- **Allocate a group of memory locations: character string, table of numbers**

- **Declaring and using arrays**

```
Example:
int grid[5] = {1,3,5,7,9};
grid[4] = grid[2] + 1;
int i;
for(i=0;i<2;i++){
   grid[i+1] = grid[i] + 2;
}
```

Passing Array as Argument

Arrays are passed by reference in C

- the address of the array (i.e., address of the first element) is written to the function's activation record
- otherwise, would have to copy each element

```
Example:
int main(){
   int array[10] = {2,4,6,8,10,12,14,16,18,20};
   int result;
   result = average(array, 10);
   return 0;
}

int average(int array[10], int size);
/* int average(int array[], int size); */
/* int average(int *array, int size); */
```

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 first element in array *word*	**cptr**	**word**	**&word[0]**
Address of the nth element in array *word*	**(cptr+n)**	**(word+n)**	**&word[n]**
Value of the first element in array *word*	***cptr**	***word**	**word[0]**
Value of the nth element in array *word*	***(cptr+n)**	***(word+n)**	**word[n]**

Exercise: implement a function to reverse an array

/* array_reversal(): reverses 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){
```

```
}
```