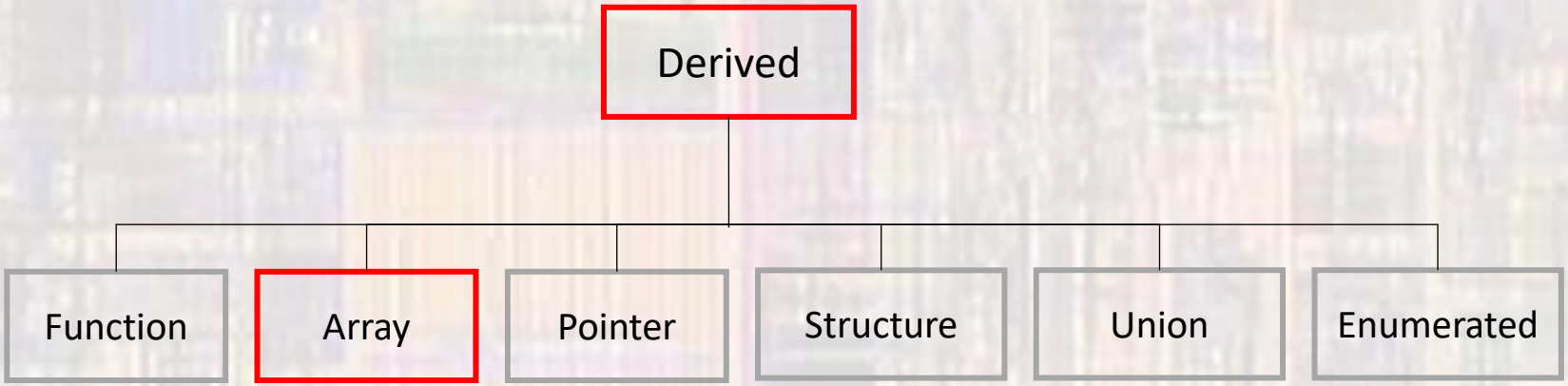
Arrays

Last updated 6/19/23

These slides introduce 1-dimensional arrays in C

Arrays

- C Types
  - Arrays are a Derived type



Arrays

- Arrays
  - Grouping of similar items

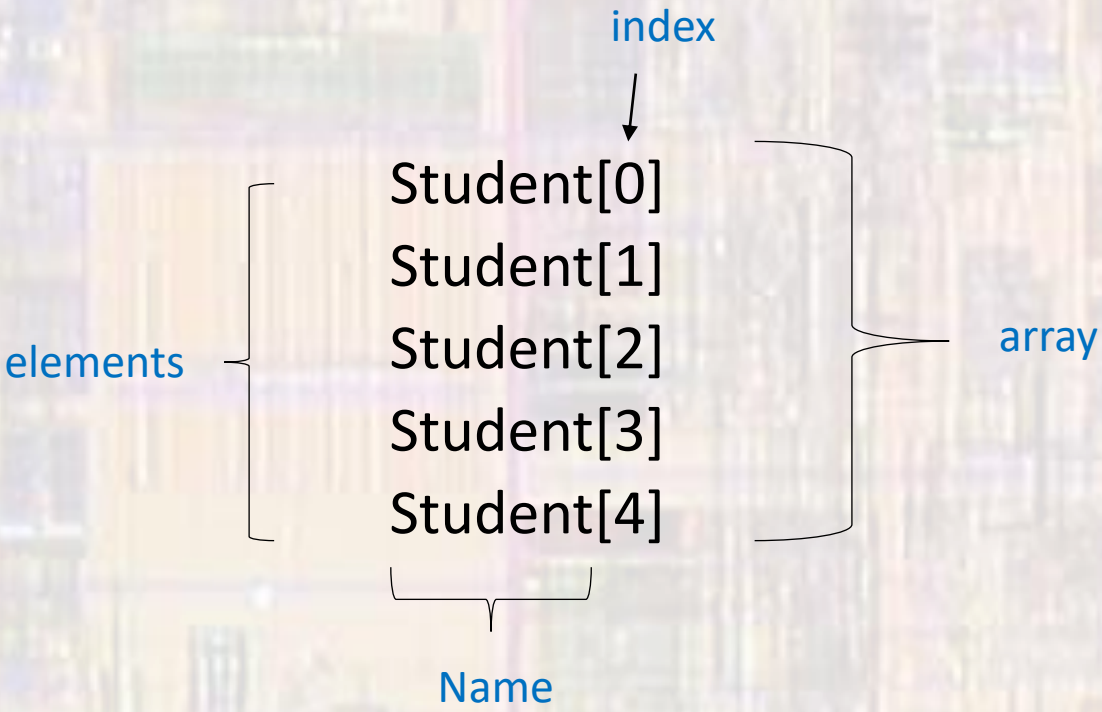
Student 0	$Student_0$	Student[0]
Student 1	$Student_1$	Student[1]
Student 2	$Student_2$	Student[2]
Student 3	$Student_3$	Student[3]
Student 4	$Student_4$	Student[4]

Arrays

- Arrays in C
  - All elements in the array must be of the same type

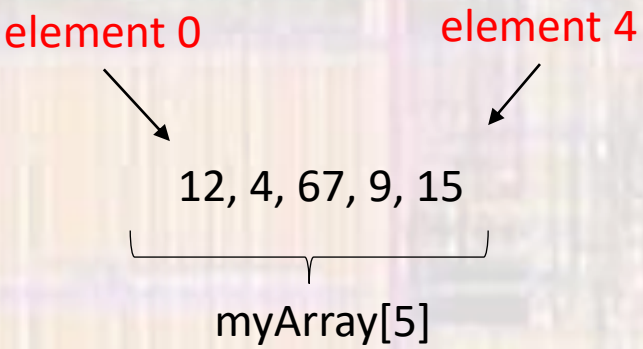
Arrays

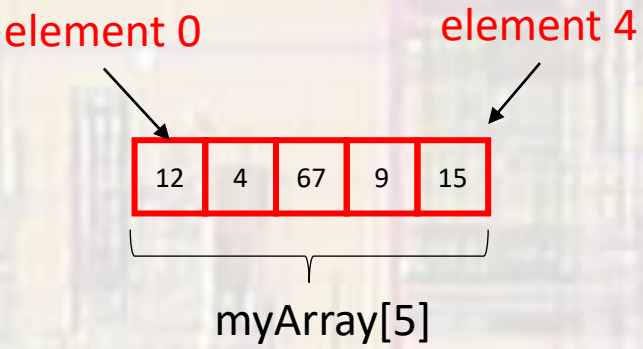
- Array notation



Arrays

- Array Notation – list form
  - First element [0] is left most element
  - Last element has index size - 1





Arrays

- Declaration

```
type arrayName[arraySize];
```

Fixed length array – size known during compilation

```
int scores[22];
char first_name[15];
```

Variable length array – size only known during execution

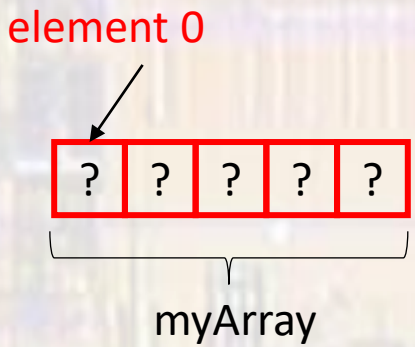
```
float testAve[classSize];
int numAs[gradesGE90];
```

where classSize and gradesGE90 are integral variables

Arrays

- Declaration
  - Un-initialized arrays contain garbage

```
type arrayName[arraySize] ;

int myArray[5];
```



Arrays

- Declaration with Initialization

```
type arrayName[arraySize] = {comma separated list};
```

```
int myArray[5] = {5,4,3,2,1};          // basic
```


```
int myArray[5] = {5,4};                // partial initialization
                                       // others are set to 0
```


```
int myArray[ ] = {5,4,3,2,1};          // size is taken from
                                       // initialization values
```


```
int myArray[5] = {0};                  // all set to 0
```

Arrays

- Variable length arrays

Variable length arrays **cannot** have an initialization

```
float testAve[classSize];
int numAs[gradesGE90];
```

Arrays

- Accessing Elements

myArray


```
foo = myArray[3];          // foo = 2
foo = myArray[foo];        // foo = 3
```

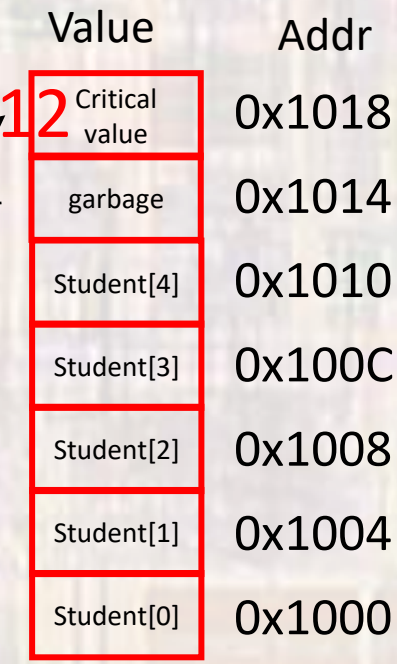
```
myArray[0] = 0;
```


```
myArray[foo + 1] = 6;
```

Arrays

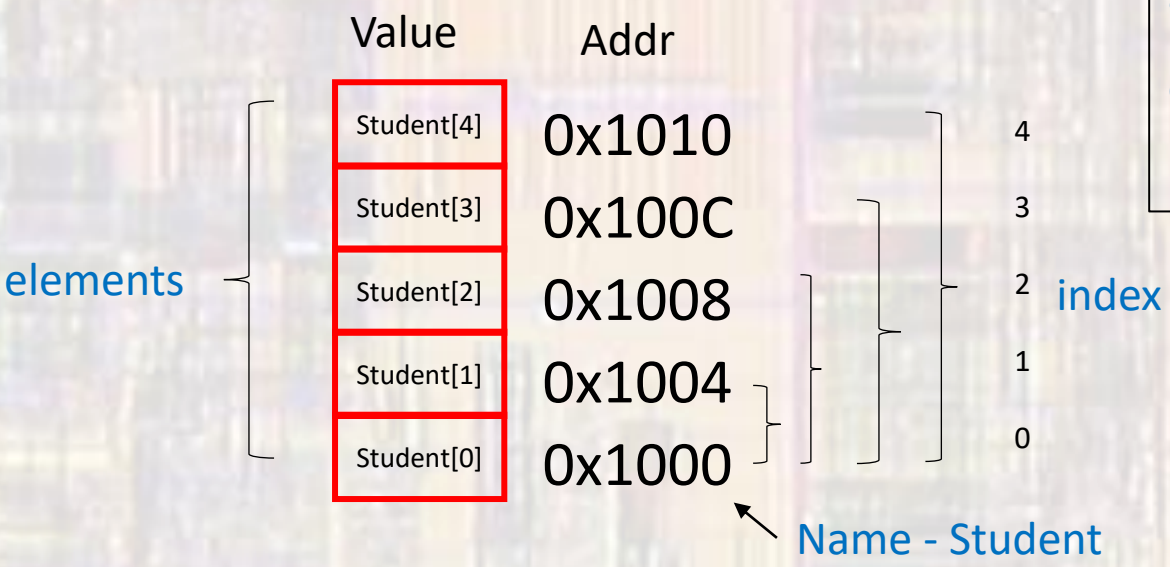
- Index Range Checking
  - C does NOT check array index ranges

```
int Student[5];
…

foo = Student[5];
    sets foo = garbage

Student[6] = 12;
    overwrites critical data value
```



Arrays

- Memory Structure
  - Name is actually the address of the beginning of the array (a pointer)
  - Index is the offset from the name address
    - not an address



addr offset in memory is calculated by the compiler to match the size of the element types

offset = size_of_type * index