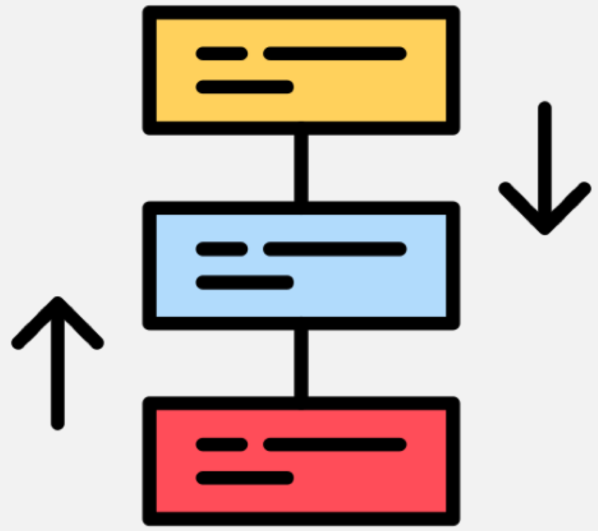
# COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

## *Week 9- Class I:* Arrays Part III



## Administrivia

- SPA 2 Grades out.
- Small Programming Assignment 3 will be out by tomorrow
- Exam 2 is on Wednesday, October 25th
    - We will review in class on Monday 🙂

# Today's Agenda

1. More Arrays!

## Quick Review

## Accessing Values

- Now that we have observe the stack space visualization of arrays, we now have to understand how values are accessed.
- Subscripted variable are variables followed by a subscript in brackets, designating an array element.
- Array subscript is a value or expression enclosed in brackets after the array name, specifying which array element to access.

Array x

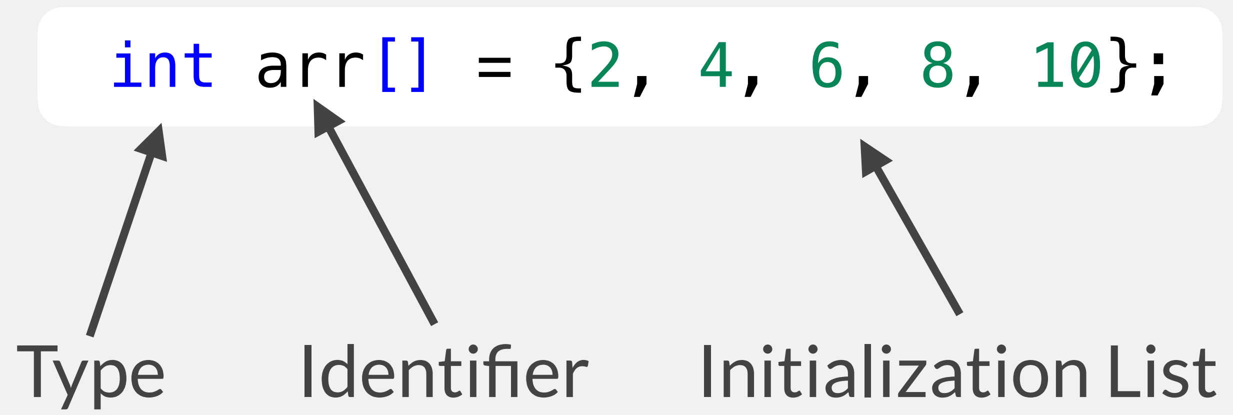
| 4 | 2 | 46 | 3 | 8 | 55 | 3 |
|---|---|---|---|---|---|---|
| x[0] | x[1] | x[2] | x[3] | x[4] | x[5] | x[6] |

## Useful Statements for Array Access

| Statement | Explanation |
|---|---|
| `printf(“%d,x[0]);` | Displays the stored value at `x[0]` |
| `x[3] = 1;` | Stores the value 1 in `x[3]` |
| `sum = x[0] + x[1];` | Stores the sum of `x[0]` and `x[1]` |
| `sum += x[2];` | Adds `x[2]` to sum |
| `x[3] +=13;` | Adds 13 to `x[3]` |
| `x[2] = x[0] + x[1]` | Adds the values stored in `x[0]` and `x[1]`. |

## Array Initialization List

- Like variables, arrays must be declared and initialize.
- In order to declare an array, programmers must specify the type of data it holds along with the predefined size.
- Programmers can also declare and initialize an array in one line of code (programmers don't have to include the size if this method is done).
- When an array is declared, what values are automatically stored?



```
int arr[] = {2, 4, 6, 8, 10};
```

# Array Initialization List

```
int arr[] = {2, 4, 6, 8, 10};
```

| Stack | Space |
|---|---|
| AA9 | |
| AA8 | |
| AA7 | |
| AA6 | |
| AA5 | |
| AA4 | 🟩 |
| AA3 | 🟩 |
| AA2 | 🟩 |
| AA1 | 🟩 |
| AA0 | 🟩 |

## Variable Length Arrays ☹️

```c
int size;

printf("Enter the number of elements: ");

scanf("%d", &size);

int arr[size]; // GROSS!
```

NEVER DO THIS!

## Arrays

# ArraySubscripts

- Subscript are used to access and manipulate array elements.
- It's very important to know how to manipulate array elements.

| Statement | Explanation |
|---|---|
| `x[i-1] = x[i];` | Assign the value stored at index i to index i-1 |
| `x[i] = x[i+1];` | Assignment the value stored at index i + 1 to index i |
| `x[i] -1 = x[i]` | Illegal! |

## Array Subscript Example

Here →

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 3 |
| AA1 | arr[1] = 2 |
| AA0 | arr[0] = 1 |

# Array Subscript Example

Here →

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 3 |
| AA1 | arr[1] = 2 |
| AA0 | arr[0] = 2 |

## Array Subscript Example

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

Here →

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 3 |
| AA1 | arr[1] = 2 |
| AA0 | arr[0] = 2 |

# Array Subscript Example

Here

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 3 |
| AA1 | arr[1] = 2 |
| AA0 | arr[0] = 2 |

# Array Subscript Example

Here

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 3 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

## Array Subscript Example

Here →
```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 3 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

# Array Subscript Example

Here →

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 3 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

# Array Subscript Example

Here

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 3 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

# Array Subscript Example

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

Here →

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 4 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

# Array Subscript Example

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

Here

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 4 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

## Array Subscript Example

Here

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 4 |
| AA2 | arr[2] = 4 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

# Array Subscript Example

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

Here →

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 5 |
| AA2 | arr[2] = 4 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

## Array Subscript Example

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

Here →

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 5 |
| AA2 | arr[2] = 4 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

## Array Subscript Example

Here

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 5 |
| AA3 | arr[3] = 5 |
| AA2 | arr[2] = 4 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

# Array Subscript Example

Here →

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 6 |
| AA3 | arr[3] = 5 |
| AA2 | arr[2] = 4 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

# Array Subscript Example

Here

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 6 |
| AA3 | arr[3] = 5 |
| AA2 | arr[2] = 4 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

## Array Subscript Example

Here

```
for (int x = 0; x < 5; x++){
    arr[x] = arr[x + 1];
}
```

| Stack Space | |
|---|---|
| AA9 | arr[9] = 10 |
| AA8 | arr[8] = 9 |
| AA7 | arr[7] = 8 |
| AA6 | arr[6] = 7 |
| AA5 | arr[5] = 6 |
| AA4 | arr[4] = 6 |
| AA3 | arr[3] = 5 |
| AA2 | arr[2] = 4 |
| AA1 | arr[1] = 3 |
| AA0 | arr[0] = 2 |

# `sizeof()` Operator

- In C, there's an operator that programmers can use to determine the exact size of the array.
- `sizeof()` is an operator that is used to determine the size of a variable allocated for memory.
  - Integer: 4 bytes
  - Double: 8 bytes
  - Character: 1 byte
  - Float (in Eustis): 4 bytes
  - Pointer: 8 bytes
- This operator can be used to determine the number of elements in a predefined array.

```c
int size = sizeof(arr)/sizeof(arr[0]);
```

## Using Loops to Access Arrays

- Loops are very useful to access all elements in a given array.
- Loops will handle all possibilities for accessing all elements in the array.

```
for (int i = 0; i < SIZE; i++){
    list[i] = val;
}
```

## UsingArray Elements as Function Arguments

- We understand how arrays are declared, initialize, and accessed.
- How can arrays be used with other functions?
- Like variables, programmers can pass arrays to other functions.
- Something interesting about arrays are that they are memory addresses.
- What kind of pass-by does that handle?

## Demo

## Acknowledgements

Slides adapted from Dr. Andrew Steinberg's COP 3223H course