COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

Week 8- Class III: Arrays Part II

Administrivia

- *Large Programming Assignment 1* is due today!!
- SPA 2 Grades out today/tomorrow.

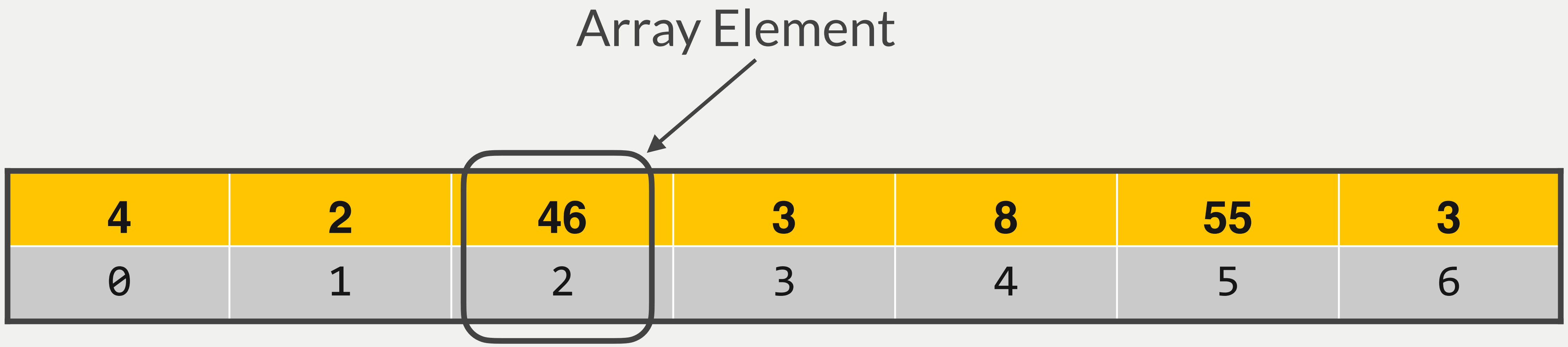
Today's Agenda

1. More Arrays!

Quick Review

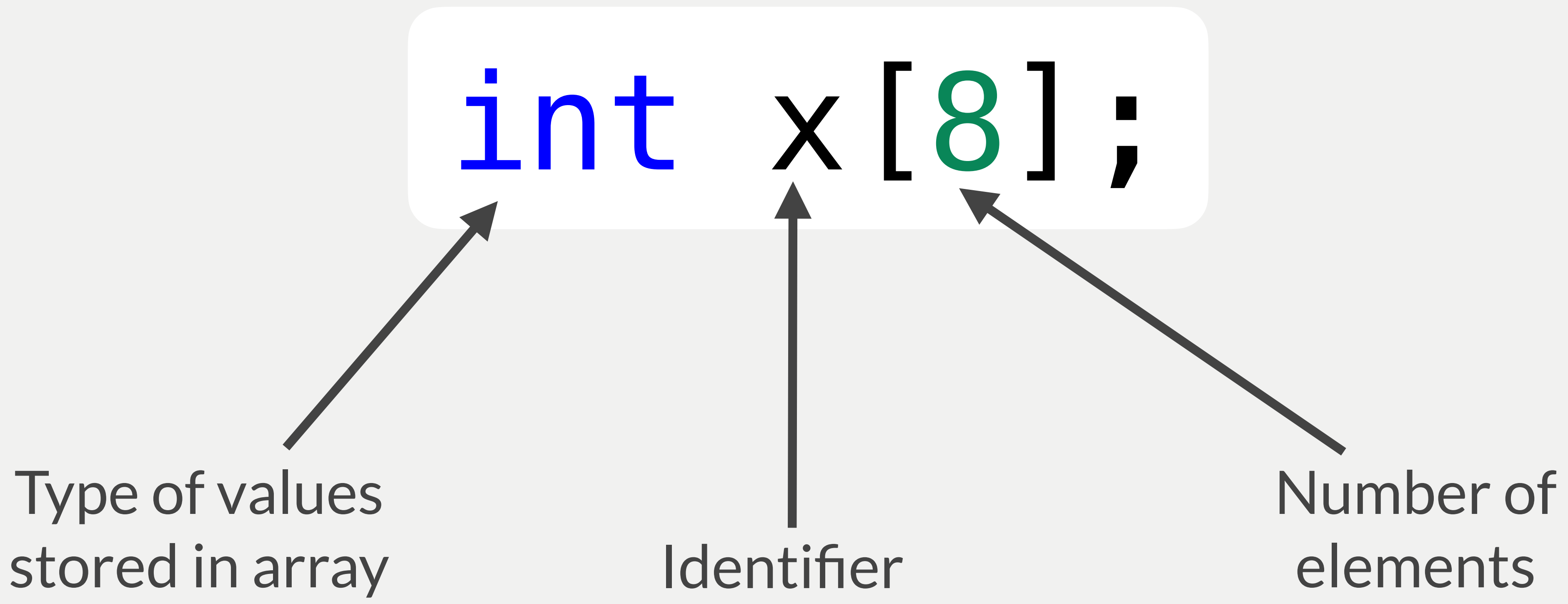
Arrays

- An Array is a collection of data items of the same type.
- An array element is a data item that is part of an array.
- An array is a collection of two or more adjacent memory cells.



4	**2**	**46**	**3**	**8**	**55**	**3**
0	1	2	3	4	5	6

Declaring an Array



```
int x[8];
```

Arrays and Stack Visualization

```
int x[8];
```

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

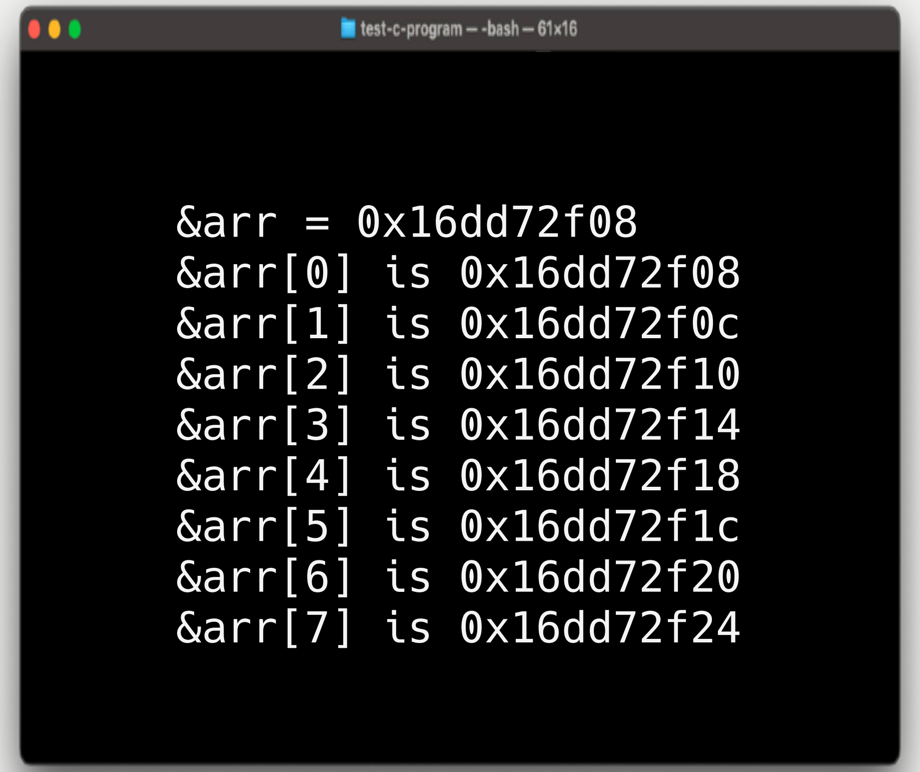
Here we have an array (called x) of 8 elements. That means there are 8 adjacent cells occupied.

Array Variables

- We have already observed the adjacent memory cells by displaying their addresses.
- What about the actual array variables?
- For example: Where do you think the variable `arr` itself is located?

```c
int arr[8];

printf("&arr = %p\n", &arr);

for(int x = 0; x < 8; x++){

    printf("&arr[%d] is %p\n", x, &arr[x]);

}
```

The first adjacent stack cell is the actual place where the array variable is stored in memory.



```
&arr = 0x16dd72f08
&arr[0] is 0x16dd72f08
&arr[1] is 0x16dd72f0c
&arr[2] is 0x16dd72f10
&arr[3] is 0x16dd72f14
&arr[4] is 0x16dd72f18
&arr[5] is 0x16dd72f1c
&arr[6] is 0x16dd72f20
&arr[7] is 0x16dd72f24
```

Arrays

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]

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

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

Arrays and Stack Visualization

Here

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = ??
AA1	arr[1] = ??
AA0	arr[0] = ??

Arrays and Stack Visualization

Here

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 0
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = ??
AA1	arr[1] = ??
AA0	arr[0] = ??

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here →

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 0
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = ??
AA1	arr[1] = ??
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 0
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = ??
AA1	arr[1] = ??
AA0	arr[0] = 0

Arrays and Stack Visualization

Here

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 1
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = ??
AA1	arr[1] = ??
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here →

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 1
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = ??
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 1
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = ??
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

Here

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 2
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = ??
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

        arr[x] = x * 3;

}
```

Here

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 2
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = 6
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 2
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = 6
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 3
AA4	arr[4] = ??
AA3	arr[3] = ??
AA2	arr[2] = 6
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here →

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 3
AA4	arr[4] = ??
AA3	arr[3] = 9
AA2	arr[2] = 6
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here →

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 3
AA4	arr[4] = ??
AA3	arr[3] = 9
AA2	arr[2] = 6
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

Here →

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 4
AA4	arr[4] = ??
AA3	arr[3] = 9
AA2	arr[2] = 6
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here →

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 4
AA4	arr[4] = 12
AA3	arr[3] = 9
AA2	arr[2] = 6
AA1	arr[1] = 3
AA0	arr[0] = 0

Arrays and Stack Visualization

```
int arr[5];

for(int x = 0; x < 5; x++){

    arr[x] = x * 3;

}
```

Here →

Stack	Space
AA9	
AA8	
AA7	
AA6	
AA5	x = 4
AA4	arr[4] = 12
AA3	arr[3] = 9
AA2	arr[2] = 6
AA1	arr[1] = 3
AA0	arr[0] = 0

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]`.

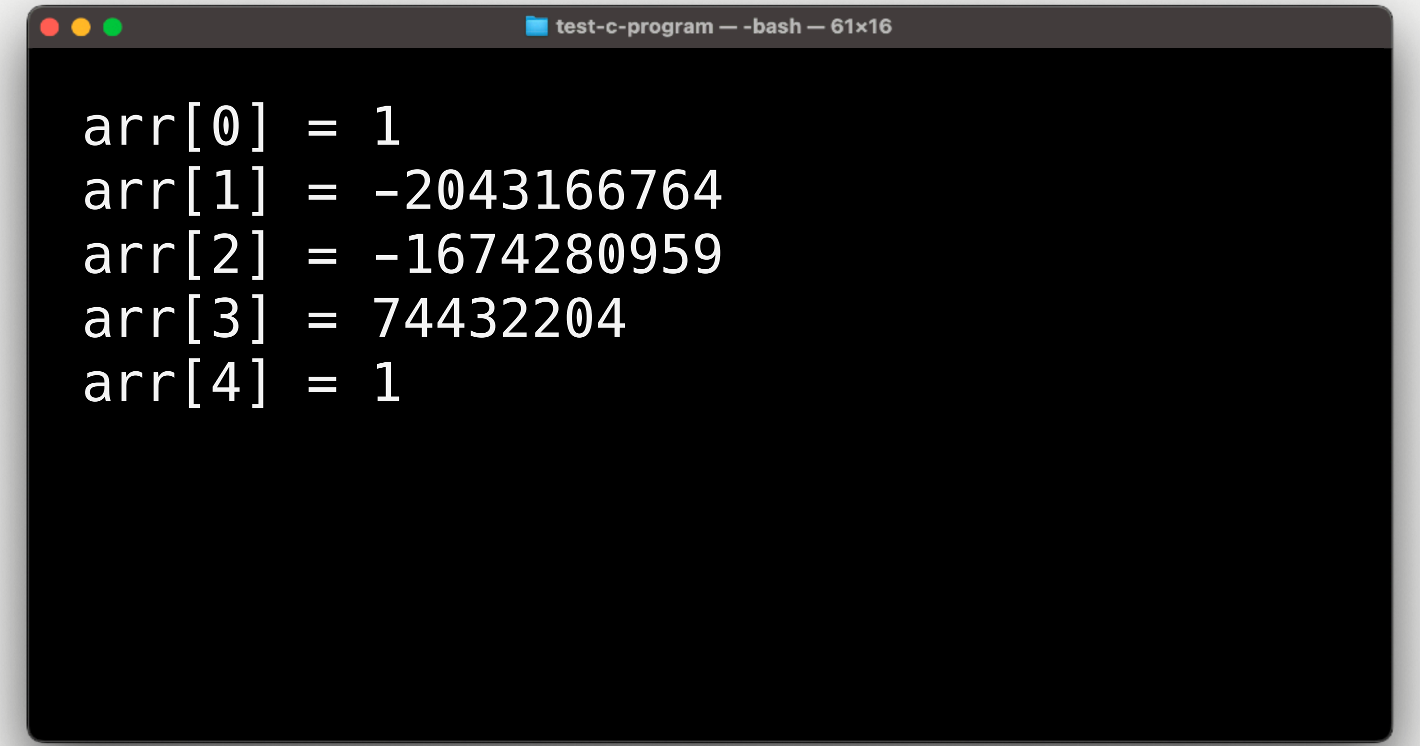
Array Initialization

- 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[5]; // What is stored inside memory after declaration
```

Array Initialization

```
int arr[5];

for(int x = 0; x< 5; x++){

    printf("arr[%d] = %d\n", x, arr[x]);

}
```

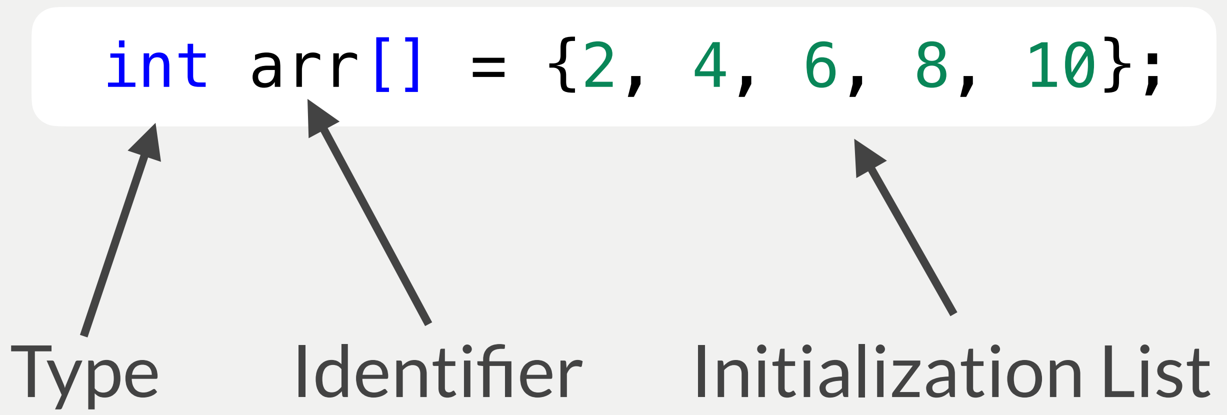


```
arr[0] = 1
arr[1] = -2043166764
arr[2] = -1674280959
arr[3] = 74432204
arr[4] = 1
```

How can we initialize an array when we create it?

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};
```

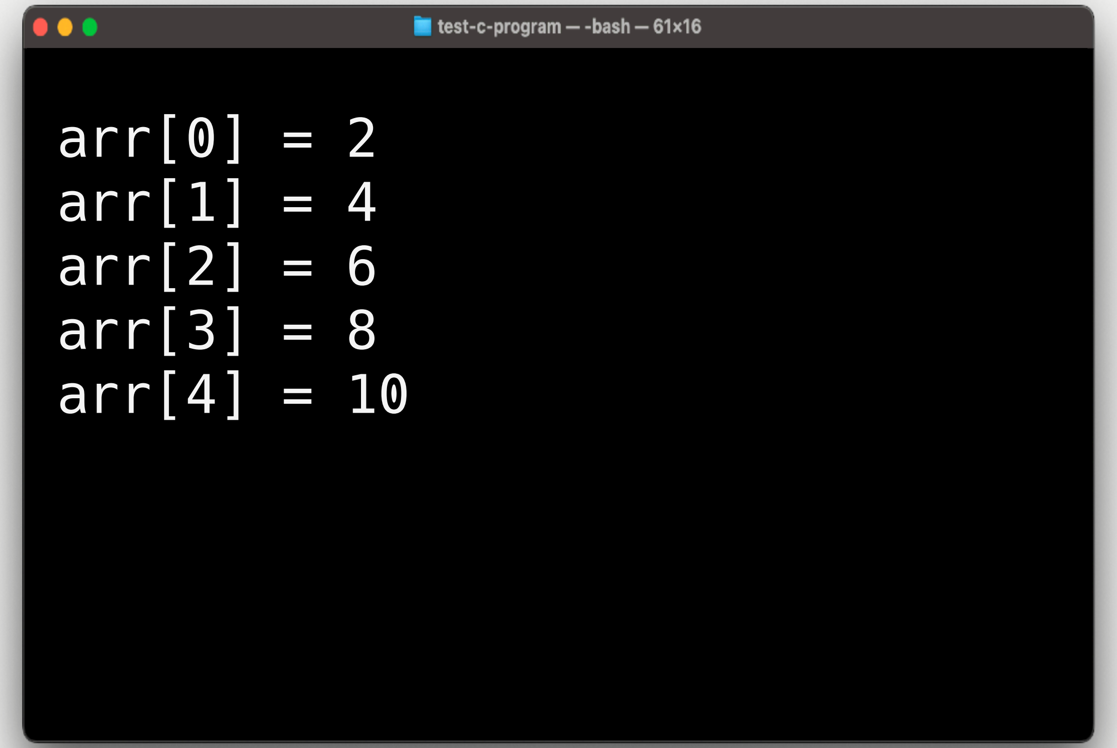
Array Initialization List

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

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

Array Initialization List

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

for(int x = 0; x < 5; x++){

    printf("arr[%d] = %d\n", x, arr[x]);

}
```

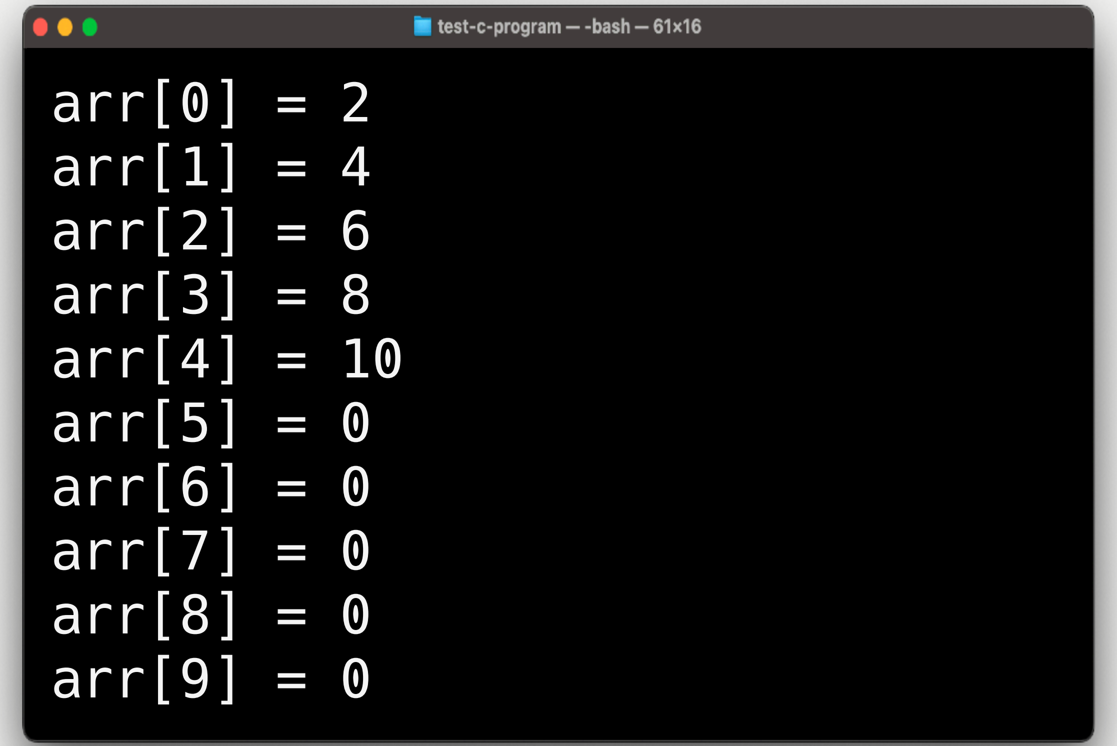

test-c-program — -bash — 61×16

```
arr[0] = 2
arr[1] = 4
arr[2] = 6
arr[3] = 8
arr[4] = 10
```

What if we put a number inside []?

Array Initialization List

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

for(int x = 0; x < 10; x++){

    printf("arr[%d] = %d\n", x, arr[x]);

}
```



```
arr[0] = 2
arr[1] = 4
arr[2] = 6
arr[3] = 8
arr[4] = 10
arr[5] = 0
arr[6] = 0
arr[7] = 0
arr[8] = 0
arr[9] = 0
```

Demo

Default Values for Different Data Types

- `int` - 0
- `double` 0.0
- `float` - 0.0
- `char` - ‘\0’ Null Character
- `pointer` - Null

Arrays of Characters

- Now we have seen array of ints and doubles.
- What about an array of characters?
- Programmers can have an array of characters.
- Array of characters are known as strings.
- Strings are a bit more unique than array of ints and doubles.
- We will cover this more in depth after Exam 2! ☺

Variable Length Arrays ☹️

- The arrays we are dealing with use static memory (stack space).
- Static means no flexibility in changing the size of memory required.
- Adding this flexibility results in dynamic memory
- We will study this at the end of the semester.
- Never use variables when declaring an array as you can have potential danger in what the value a variable can hold.
- VLAs pose danger if we accidentally change a value to a size that can't be properly handled in memory.

Variable Length Arrays ☹️

```c
int size;

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

scanf("%d", &size);

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

NEVER DO THIS!

Acknowledgements

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