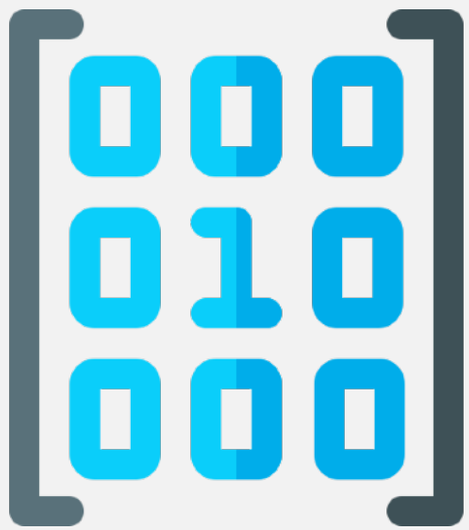
COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

Week 9- Class II: 2D-Arrays Part I



Administrivia

- *Small Programming Assignment 3 out.*
  - Due Friday October 27th.
  - We will discuss right now.
- *I have finalized the assignments for the semester.*
  - 5 small programming assignments
  - 3 Large Programming Assignments
- *Quiz 2 will be released today*
  - Due on Monday, Oct 23rd
- *Exam 2 is on Wednesday, October 25th*
  - We will review in class on Monday 🙂

Today's Agenda

1. Review Arrays passed to User Defined Functions
2. Introduce 2D-Arrays!

Review

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!

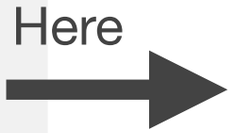
Array Subscript Example

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




```
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

`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.

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

Using Array 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?

Using Array Elements as Function Arguments

- Function prototype shows we are passing an array
- What does C pass arrays by reference?
- It is *Far* more efficient to always pass a pointer than to pass a copy of the entire array!

```c
#include<stdio.h>
# define SIZE 10

void fillArray(int list[], int val);

int main(void){

int list[SIZE];

fillArray(list, SIZE);

for(int i = 0; i < SIZE; i++){
    printf("arr[%d] = %d\n", i, list[i]);
}

return 0;
}

void fillArray(int list[], int val){

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

}
```

Using Array Elements as Function Arguments

- In this code, you might notice that `sizeof()` operators are being used to calculate the # of elements.

- However, there is an issue with this code and we will get a compiler warning!

```c
#include<stdio.h>

void displayArray(int list[]);

int main(void){

int list[5];

for(int i = 0; i < 5; i++){
    list[i] = i + 1;
}

displayArray(list);

return 0;
}


void displayArray(int list[]){

    for(int i = 0; i < sizeof(list)/sizeof(list[0]); i++){
        printf("list[%d] = %d\n", i, list[i]);
    }

}
```

```
warning: sizeof on array function parameter will return size of 'int *' instead of
'int[]' [-Wsizeof-array-argument]
```

Using Array Elements as Function Arguments

- In this code, you might notice that `sizeof()` operators are being used to calculate the # of elements.

- However, there is an issue with this code and we will get a compiler warning!

- Remember a pointer is 8 bytes, and an integer is 4 bytes.

```c
#include<stdio.h>

void displayArray(int list[]);

int main(void){

int list[5];

for(int i = 0; i < 5; i++){
    list[i] = i + 1;
}

displayArray(list);

return 0;
}


void displayArray(int list[]){

    for(int i = 0; i < sizeof(list)/sizeof(list[0]); i++){
        printf("list[%d] = %d\n", i, list[i]);
    }

}
```

```
warning: sizeof on array function parameter will return size of 'int *' instead of
'int[]' [-Wsizeof-array-argument]
```

Using Array Elements as Function Arguments

```c
#include<stdio.h>

void displayArray(int list[]);

int main(void){

int list[5];

for(int i = 0; i < 5; i++){
    list[i] = i + 1;
}

displayArray(list);

return 0;
}


void displayArray(int list[]){

    for(int i = 0; i < sizeof(list)/sizeof(list[0]); i++){
        printf("list[%d] = %d\n", i, list[i]);
    }

}
```

- What happens if we run this code?

```
test-c-program — -bash — 61×16

Legacy:code KevinMoran$ ./arrays
list[0] = 1
list[1] = 2
Legacy:code KevinMoran$
```

- What is going on??

Demo

2D-Arrays

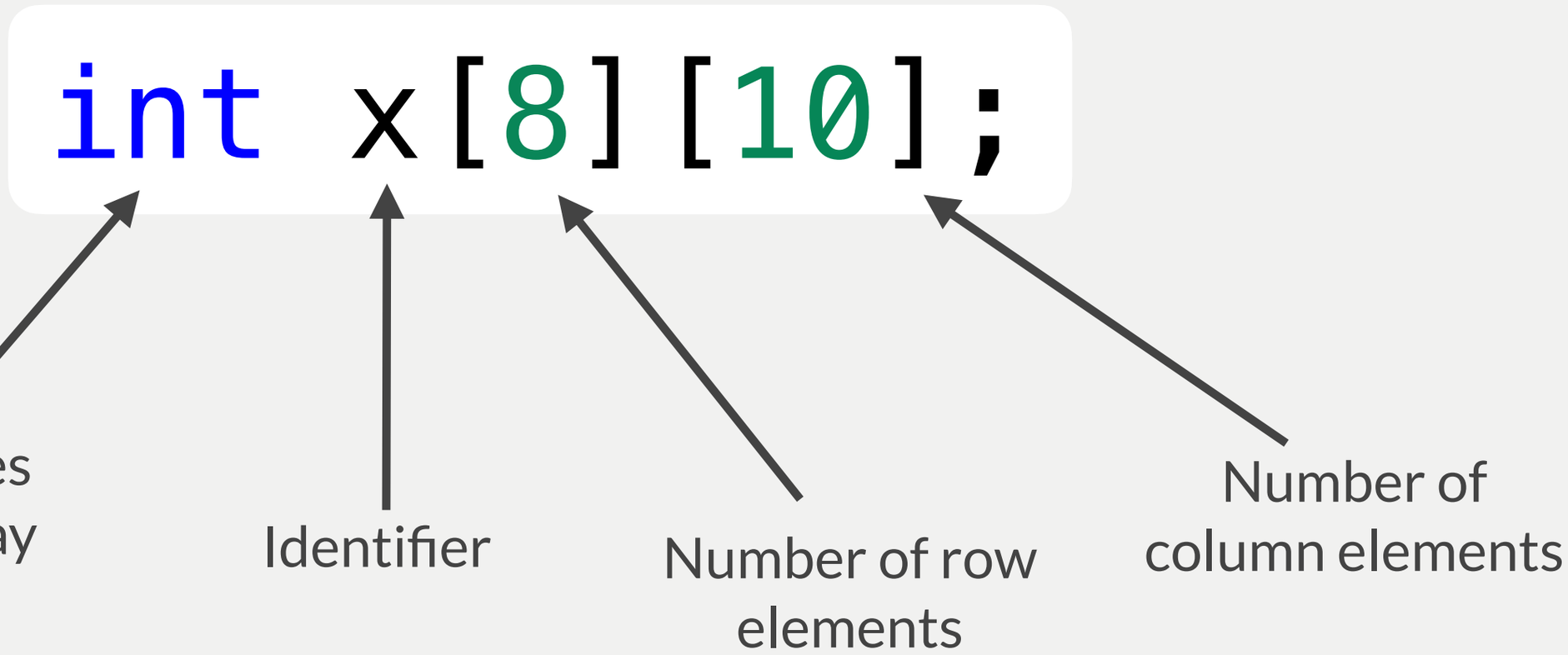
2D Arrays

- We have seen that arrays can be useful, but what if we need to store multidimensional data?
- 2D-Arrays to the rescue!
- 2D Arrays allow us to store information in a matrix-like format, as shown below.

	0	1	2	3
0	a	s	d	f
1	n	k	i	v
2	h	j	k	l
3	f	e	o	p

Example of a 2-D Array of Characters

Declaring a 2D Array



2D Array Initialization

```
int arr[3][3]  = { {24, 15, 34}, {26, 134, 194}, {67, 23, 345} };
```

	0	1	2
0	24	15	34
1	26	134	194
2	67	23	345

Accessing Array Elements

```
int arr[3][3]  = { {24, 15, 34}, {26, 134, 194}, {67, 23, 345} };
```

	0	1	2
0	24	15	34
1	26	134	194
2	67	23	345

```
int test_val = arr[1][0];

printf("First element in second row is: %d\n", test_val);
```

Demo

Acknowledgements

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