Week 4

XMUT-NWEN 241 - 2024 T2

Systems Programming

Mohammad Nekooei

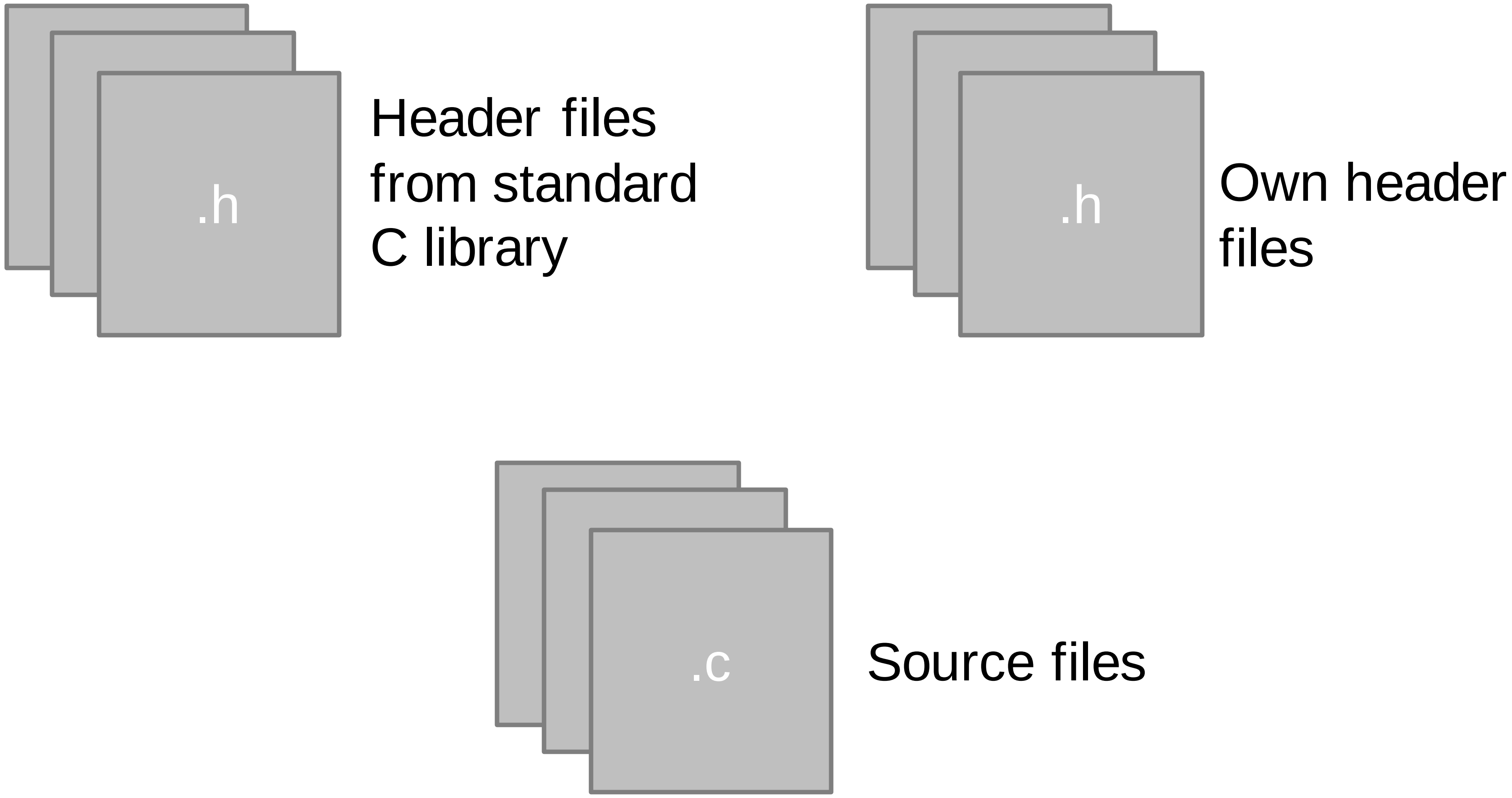
School of Engineering and Computer Science

Victoria University of Wellington

Content

- More on Arrays

Recall: Large C Program



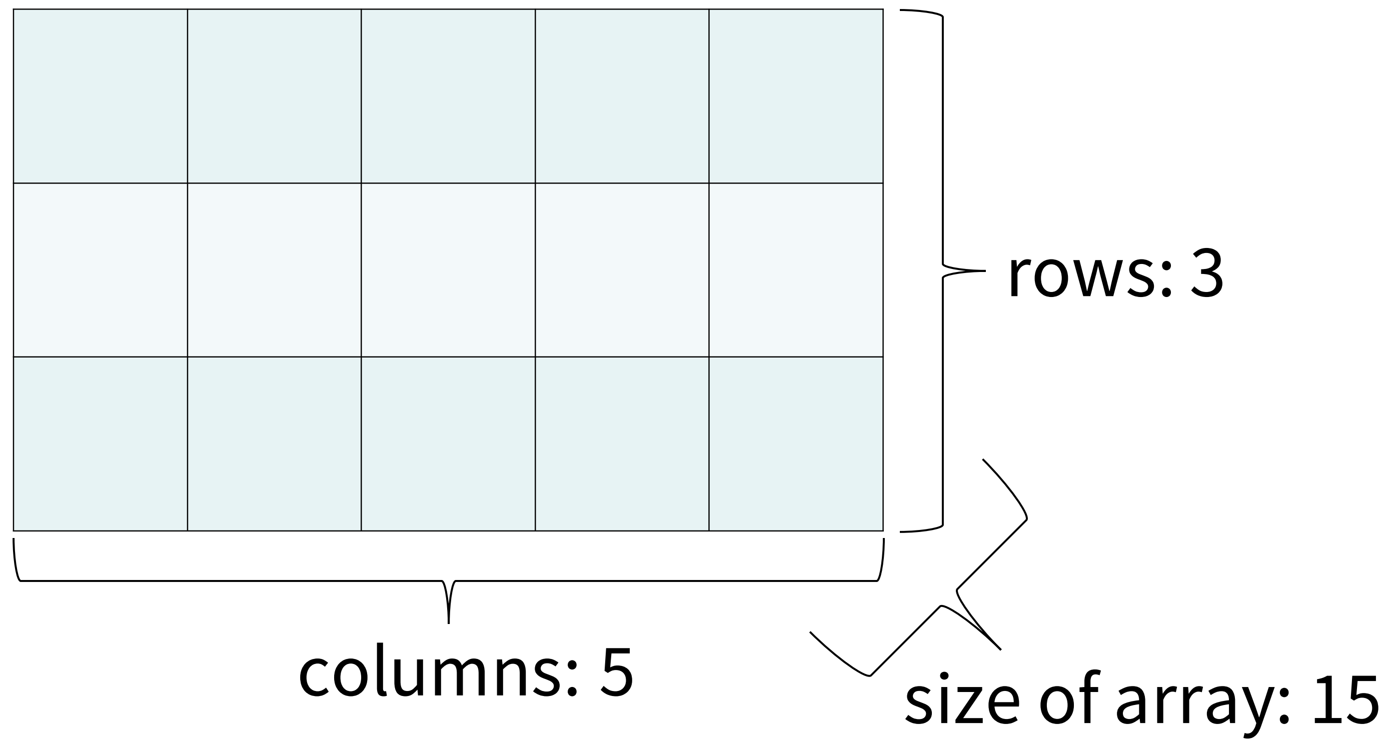
Multi-dimensional Arrays

Multi-dimensional Arrays

- In C, you can create array of an array known as multidimensional array

- The simplest interpretation of a multi-dimensional array is a table, i.e. a **two-dimensional array**
  - each **row** has the same number of **columns**

Two-Dimensional Arrays Overview (1)



Two-Dimensional Arrays Overview (2)

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

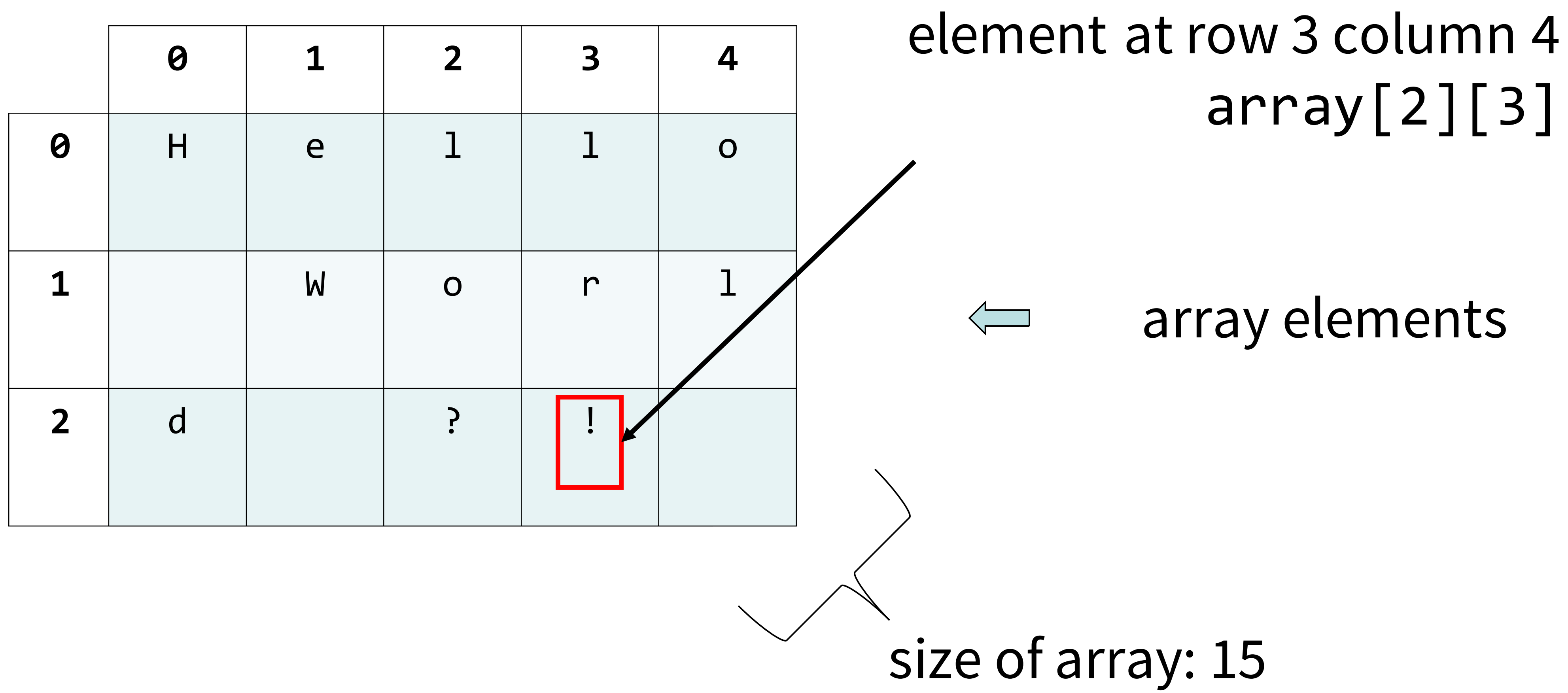
⇦ array of ints

Two-Dimensional Arrays Overview (3)

1.0	2.0	3.0	4.0	5.0
6.0	7.0	8.0	9.0	10.0
11.0	12.0	13.0	14.0	15.0

array of floats

Two-Dimensional Arrays Overview (4)



	0	1	2	3	4
0	H	e	l	l	o
1		W	o	r	l
2	d		?	!	

Two-Dimensional Arrays

- Declaring a char array with 3 rows and 5 columns

```
char two_d[3][5];
```

  – The array can hold 15 char elements

- Accessing a value

```
char ch;
ch = two_d[2][4];
```

- Modifying a value

```
two_d[0][0] = 'x';
```

- The array can be initialized in one of the following ways

```
int two_d[2][3] = {{5, 2, 1}, {6, 7, 8}};
int two_d[2][3] = {5, 2, 1 , 6, 7, 8};
int two_d[][3] = {{5, 2, 1}, {6, 7, 8}};
```

  – The number of columns must be explicitly stated. The compiler will find the appropriate amount of rows based on the initializer list

Passing 2D Arrays to Functions (1)

- Passing a **single array element** to a function
  - can be passed in a similar manner as passing a variable to a function

```
void display(int a) {
   printf("%d", a);
}

int main(void) {
   int age[2][3] = { {18, 19, 20}, {21, 22, 23} };

   display(age[1][2]); /* Passing element age[1][2] only */

   return 0;
}
```

Passing 2D Arrays to Functions (2)

- Passing an **entire array** to a function
  - When passing an array as an argument to a function, it is passed by its memory address (starting address of the memory area) and not its value(call-by-address)!
  - Because a function accesses the original array values, we must be very careful that we do not inadvertently (accidentally) change values in an array within a function.

```
void enterData(int d[][10]) {
	/* Code for reading and saving data into 2D array */
}

int main(void)
{
	int data[10][10];

	enterData(data);
}
```

Arrays

- Arrays are second class citizens

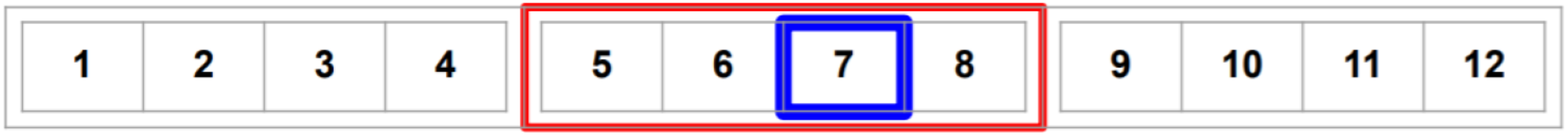
- With an array, you can NOT:
  - Change the size after initialization
  - Assign a new array using ‘=’

- In addition, arrays automatically ‘decay’ into pointers, losing information about their size (with few exceptions).
  - More on array decay (after you learn pointers)!

2D Arrays

- Multi-dimensional arrays are typically contiguous.

```
int arr[3][4] = {{1, 2, 3, 4},{5, 6, 7, 8},{9, 10, 11, 12}};
int i = arr[1][2];
```



- They also need additional information to index into the correct position. When passed to a function for example, it needs to know how many values to ‘skip’ to get to an inner array.

Next Lecture

- Strings