

CSc 352

C Programming
2D Arrays

Benjamin Dicken


2D Array

```
int main() {  
    int numbers[2][2] = { {1, 2}, {3, 4} };  
    for (int i = 0; i < 2; i++) {  
        for (int j = 0; j < 2; j++) {  
            printf("Element at address %p ",  
                &numbers[i][j]);  
            printf("is %d\n", numbers[i][j]);  
        }  
    }  
}
```

2D Array

numbers
(0x00....010)

```
int main() {  
    int numbers[2][2] = { {1, 2}, {3, 4} };  
    for (int i = 0; i < 2; i++) {  
        for (int j = 0; j < 2; j++) {  
            printf("Element at address %p ",  
                &numbers[i][j]);  
            printf("is %d\n", numbers[i][j]);  
        }  
    }  
}
```



address	value
0x00..10	1
0x00..14	2
0x00..18	3
0x00..1c	4

2D Array Memory Layout

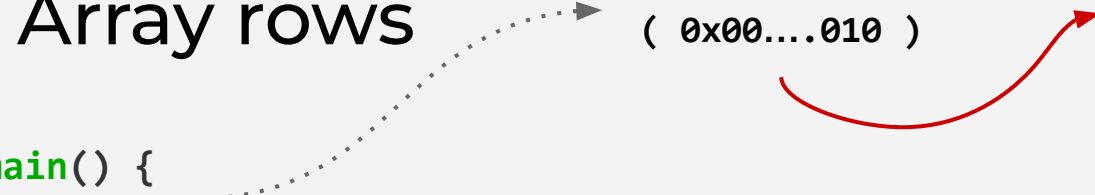
The memory layout for a 2D array is one, contiguous block of memory sized $\mathbf{N} * \mathbf{M} * \mathbf{T}$ where \mathbf{N} is the number of rows, \mathbf{M} is the number of columns, and \mathbf{T} is the size of the type of each element.

With these “basic” 2D arrays, each row is of same length, even if not given a value explicitly

2D Array rows

numbers
(0x00...010)

```
int main() {  
    int numbers[3][4] = { {1, 2, 3, 4},  
                          {50, 75}, {10, 20, 100} };  
    for (int i = 0; i < 3; i++) {  
        for (int j = 0; j < 4; j++) {  
            printf("Element at address %p ",  
                  &numbers[i][j]);  
            printf("is %d\n", numbers[i][j]);  
        }  
    }  
}
```



address	value
0x00..10	1
0x00..14	2
0x00..18	3
0x00..1c	4
0x00..20	50
0x00..24	75
0x00..28	0
0x00..2c	0
0x00..30	10
0x00..34	20
0x00..38	100
0x00..3c	0

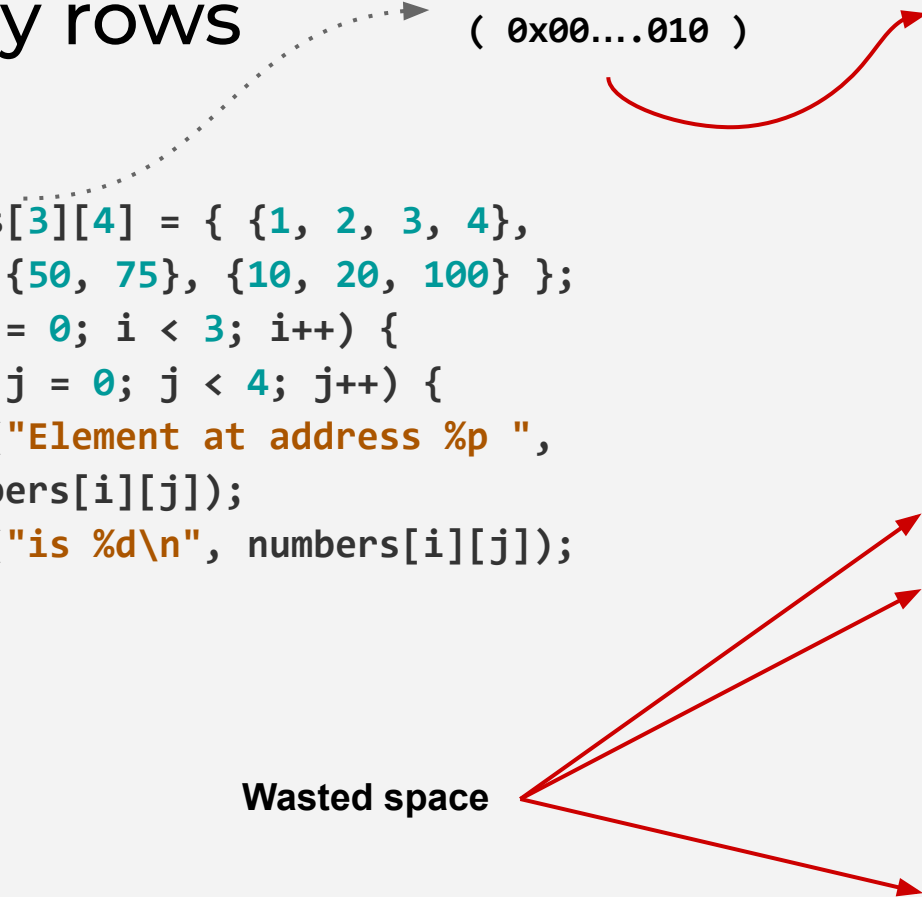
2D Array rows

numbers
(0x00...010)

```
int main() {  
    int numbers[3][4] = { {1, 2, 3, 4},  
                          {50, 75}, {10, 20, 100} };  
    for (int i = 0; i < 3; i++) {  
        for (int j = 0; j < 4; j++) {  
            printf("Element at address %p ",  
                  &numbers[i][j]);  
            printf("is %d\n", numbers[i][j]);  
        }  
    }  
}
```

address	value
0x00..10	1
0x00..14	2
0x00..18	3
0x00..1c	4
0x00..20	50
0x00..24	75
0x00..28	0
0x00..2c	0
0x00..30	10
0x00..34	20
0x00..38	100
0x00..3c	0

Wasted space



2D Array Indexing

To access an element at a pair of 2D indexes, such as:

```
int array[t][r] = {....}  
....  
printf("%d", array[x][y]);
```

The program can take the base address of **array** and add $((r*x)+y)$ to get to the address of the requested element.

What will this print?

```
int main() {
    int x[4] = {2, 1, 100, -1};
    int y[2] = {5, 7};
    int z[3] = {2, 8, -1};
    int* two_d[2] = {x, z};
    for (int i = 0; i < 2; i++) {
        for (int j = 0; two_d[i][j] != -1; j++) {
            printf("Element at address %p ",
                &two_d[i][j]);
            printf("is %d\n", two_d[i][j]);
        }
    }
    printf("%p %p %p %p", x, y, z, two_d);
}
```

Assuming that these arrays are placed in sequence on the stack.

Array of Pointers

two_d
(0x00....034)

```
int main() {  
    int x[4] = {2, 1, 100, -1};  
    int y[2] = {5, 7};  
    int z[3] = {2, 8, -1};  
    int* two_d[2] = {x, z};  
    for (int i = 0; i < 2; i++) {  
        for (int j = 0; two_d[i][j] != -1; j++) {  
            printf("Element at address %p ",  
                &two_d[i][j]);  
            printf("is %d\n", two_d[i][j]);  
        }  
    }  
    printf("%p %p %p %p", x, y, z, two_d);  
}
```

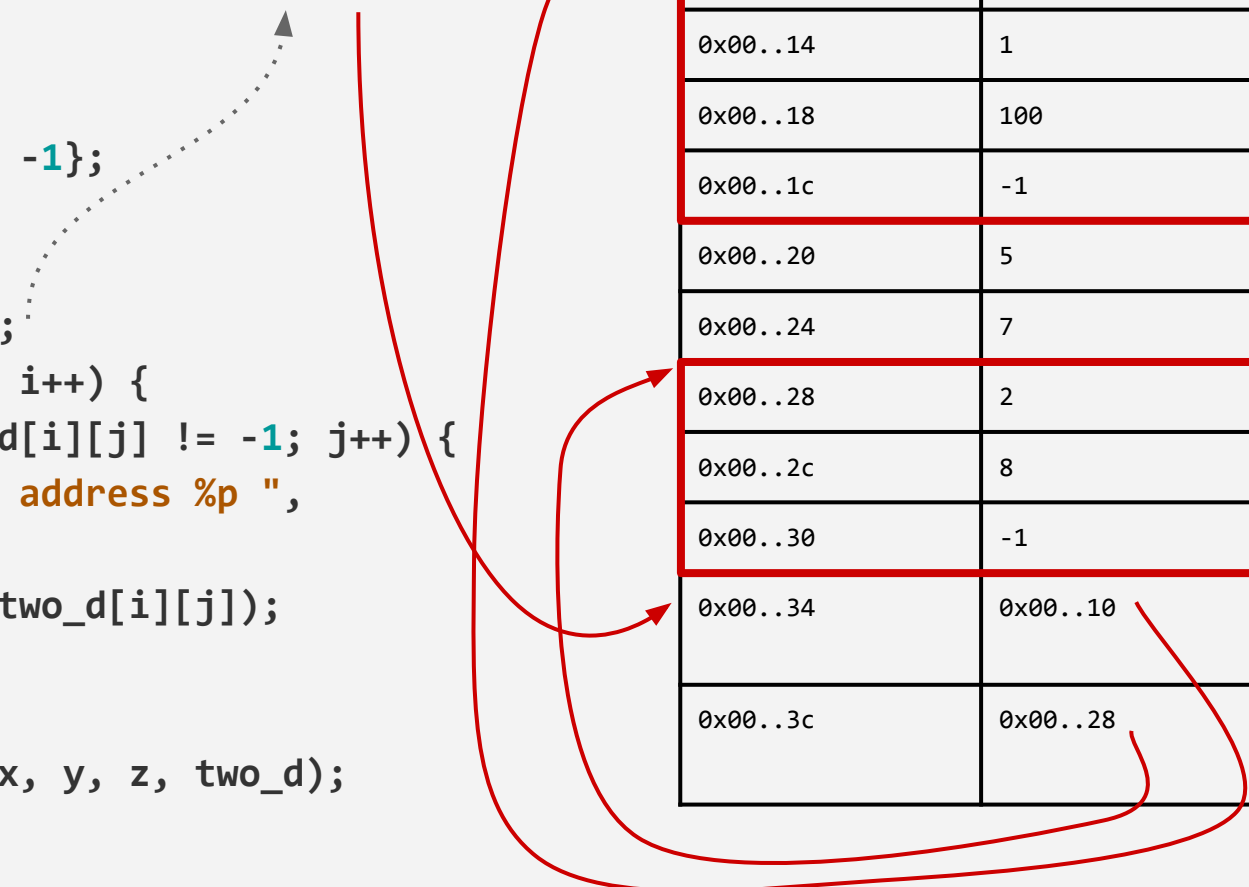
address	value
0x00..10	2
0x00..14	1
0x00..18	100
0x00..1c	-1
0x00..20	5
0x00..24	7
0x00..28	2
0x00..2c	8
0x00..30	-1
0x00..34	0x00..10
0x00..3c	0x00..28

Array of Pointers

two_d
(0x00...034)

```
int main() {  
    int x[4] = {2, 1, 100, -1};  
    int y[2] = {5, 7};  
    int z[3] = {2, 8, -1};  
    int* two_d[2] = {x, z};  
    for (int i = 0; i < 2; i++) {  
        for (int j = 0; two_d[i][j] != -1; j++) {  
            printf("Element at address %p ",  
                &two_d[i][j]);  
            printf("is %d\n", two_d[i][j]);  
        }  
    }  
    printf("%p %p %p %p", x, y, z, two_d);  
}
```

address	value
0x00..10	2
0x00..14	1
0x00..18	100
0x00..1c	-1
0x00..20	5
0x00..24	7
0x00..28	2
0x00..2c	8
0x00..30	-1
0x00..34	0x00..10
0x00..3c	0x00..28



Write the Program

- Write a program that reads paragraphs of text from standard input until EOF
- Paragraphs are separated by a blank line
- Program should determine the most-occurring word in each paragraph
- Should use a 2D array
- Max line width: 128 chars
- Max lines per paragraph: 100

story.txt

```
There once was a bear  
that lived by the sea in a tiny house.
```

```
He wanted to go into town to get some  
ice-cream. However, he knew people in town  
would be scared of him. Those town people  
are scared easily.
```

```
He came up with a plan to get around  
this. The plan was to dress up as a man.  
The plan worked, and he was able to get  
ice-cream.
```

output

```
Most-occurring word from paragraph 1: a  
Most-occurring word from paragraph 2: town  
Most-occurring word from paragraph 3: plan
```