Arrays

Suppose I need to compute statistics on class marks?

```c
int mark_student0, mark_student1, mark_student2,
mark_student0 = 73;
mark_student1 = 42;
mark_student2 = 99;
...
```

- cumbersome, need hundreds of individual variables
- can’t write while loop which executes for each student
- becomes unfeasible if dealing with a lot of values

**Solution** use an array

```c
int mark[930];
mark[0] = 73;
mark[1] = 42;
mark[2] = 99;
...
```
Arrays

Suppose I need to compute statistics on class marks?

```c
int mark_student0, mark_student1, mark_student2,
mark_student0 = 73;
mark_student1 = 42;
mark_student2 = 99;
...
```

- cumbersome, need hundreds of individual variables
- can’t write while loop which executes for each student

**Solution use an array**

```c
int mark[930];
mark[0] = 73;
mark[1] = 42;
mark[2] = 99;
...
```
C Arrays

- C array is a collection of variables called **array elements**.
- All array elements must be the same type.
- Array elements don’t have a name.
- Array elements accessed by a number called the **array index**.
- Valid array indices for array with $n$ elements are $0 .. n − 1$.
- Array can have millions/billions of elements.
- Array elements must be initialized.
- Can’t assign scanf/printf whole arrays.
- Can assign scanf/printf array elements.
// Declare an array with 10 elements
// and initialises all elements to 0
int myArray[10] = {0};
Arrays

// Declare an array with 10 elements
// and initialises all elements to 0.
int myArray[10] = {0};

// Put some values into the array.
myArray[0] = 3;

<table>
<thead>
<tr>
<th>myArray</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
// Declare an array with 10 elements and initialises all elements to 0.
int myArray[10] = {0};

// Put some values into the array.
myArray[0] = 3;
myArray[5] = 17;
### Arrays

// Declare an array with 10 elements and initialises all elements to 0.
int myArray[10] = {0};

// Put some values into the array.
myArray[0] = 3;
myArray[5] = 17;
myArray[10] = 42; // <-- Error

<table>
<thead>
<tr>
<th></th>
<th>myArray</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>
Scanf can’t read an entire array. This will read only 1 number:

```c
#define ARRAY_SIZE 42
...
int array[ARRAY_SIZE];
scanf("%d", &array);
```

Instead you must read the elements one by one:

```c
i = 0;
while (i < SIZE) {
    scanf("%d", &array[i]);
    i = i + 1;
}
```
printf can't print an entire array. This won't compile:

```c
#define ARRAY_SIZE 42
...
int array[ARRAY_SIZE];
printf("%d", array);
```

Instead must print the elements one by one:

```c
i = 0;
while (i < ARRAY_SIZE) {
    printf("%d\n", array[i]);
    i = i + 1;
}
```
Suppose we have the following:

```c
int array1[5] = {1, 2, 3, 4, 5};
int array2[5];
```

Array assignment not allowed in C. This won’t compile:

```c
array2 = array1;
```

Instead must must copy the elements one by one:

```c
i = 0;
while (i < 5) {
    array2[i] = array1[i];
    i = i + 1;
}
```
Suppose we have the following:

```c
int array1[5] = {1, 2, 3, 4, 5};
int array2[5];
```

Array assignment not allowed in C. This won’t compile:

```c
array2 = array1;
```

Instead must must copy the elements one by one:

```c
i = 0;
while (i < 5) {
    array2[i] = array1[i];
    i = i + 1;
}
```
Arrays of Arrays

- C supports arrays of arrays.
- Useful for multi-dimensional data.

```c
int matrix[3][3] = { {1, 2, 3},
                    {4, 5, 6},
                    {7, 8, 9} };

printf("%d\n", matrix[1][1]);
```
#define SIZE 42

... int matrix[SIZE][SIZE];
int i, j;

i = 0
while (i < SIZE) {
    j = 0;
    while (j < SIZE) {
        scanf("%d", &matrix[i][j]);
        j = j + 1;
    }
    i = i + 1;
}
while (i < SIZE) {
    j = 0;
    while (j < SIZE) {
        print("%d", &matrix[i][j]);
        j = j + 1;
    }
    printf("\n");
    i = i + 1;
}