COMP1511 17s2

— Lecture 7 —

Array of Sunshine

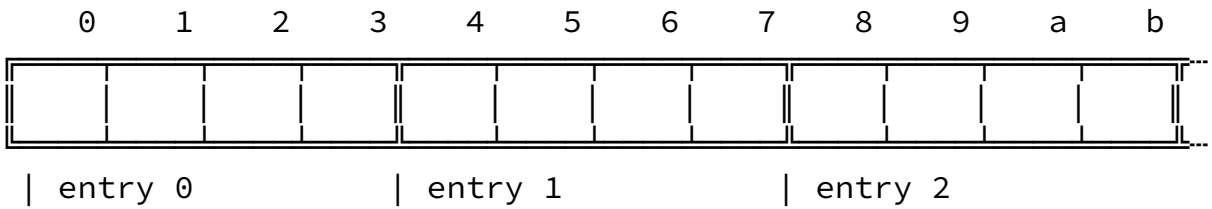
Andrew Bennett
<andrew.bennett@unsw.edu.au>

loops and repetition
arrays of data

arrays

Arrays

A series of boxes with a common type,
all next to each other



Why?

Suppose we need to compute statistics on class marks…

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

becomes unfeasible if dealing with a lot of values
… we'd need hundreds of individual variables!

Why?

Solution: Use an array!

```
int mark[550];
mark[0] = 73;
mark[1] = 42;
mark[2] = 99;
...
```

Arrays in C

a collection of array elements
each element must be the same type

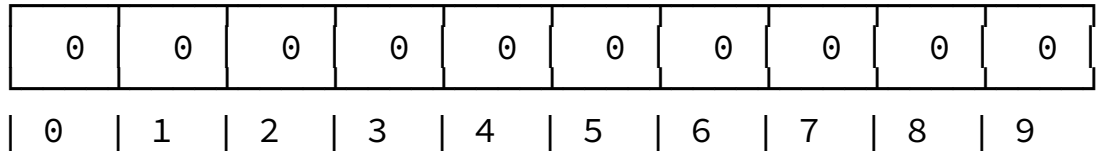
we refer to arrays by their index
valid indices for n elements are $0 \ldots n-1$

no real limit on number of elements

we cannot assign, scan, or print whole arrays…
but we can assign, scan, and print elements

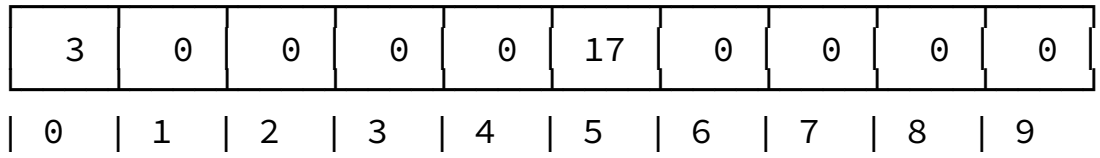
Arrays in C


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



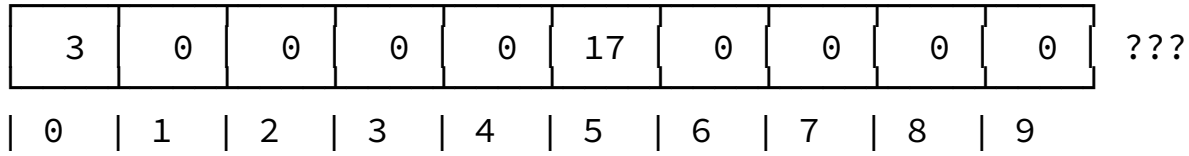
Arrays in C


```
int myArray[10] = {0};
// Put some values into the array.
myArray[0] = 3;
myArray[5] = 17;
```



Arrays in C


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



Reading an Array

`scanf()` can't read an entire array.
this will only read 1 number:

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

instead, you must read the elements one by one:

```
int i = 0;
while (i < ARRAY_SIZE) {
    scanf ("%d", &array[i]);
    i++;
}
```

Printing an Array

`printf()` also can't print an entire array.
this won't compile…

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

instead, you must print the elements one by one:

```
int i = 0;
while (i < ARRAY_SIZE) {
    printf ("%d", array[i]);
    i++;
}
```

Copying an Array

given:

```
#define ARRAY_SIZE 5
int array1[ARRAY_SIZE] = {1, 4, 9, 16, 25};
int array2[ARRAY_SIZE];
```

this won't compile...

```
array2 = array1;
```

instead, you must copy the elements one by one:

```
int i = 0;
while (i < ARRAY_SIZE) {
    array2[i] = array1[i];
    i++;
}
```

Array-ception

an array may have elements of any type...
including of array type!
we call these multi-dimensional arrays

here's an array of arrays of `int`:

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

it's a two-dimensional array.

```
printf ("%d\n", matrix[1][1]); // outputs... ?
```

Array-ception

the same caveats apply to multi-dimensional arrays:
we can't read, print, or copy the whole array,
but have to copy each non-array element…

this usually means we need to use nested loops

```
#define SIZE 42

int matrix[SIZE][SIZE] = { {0} };
int i = 0;
while (i < SIZE) {
    int j = 0;
    while (j < SIZE) {
        scanf ("%d", &matrix[i][j]);
        j++;
    }
    i++;
}
```