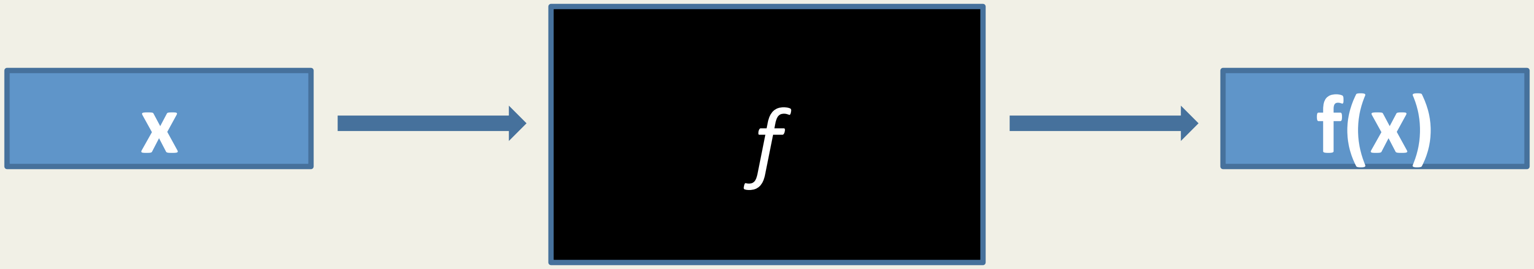
Week 2

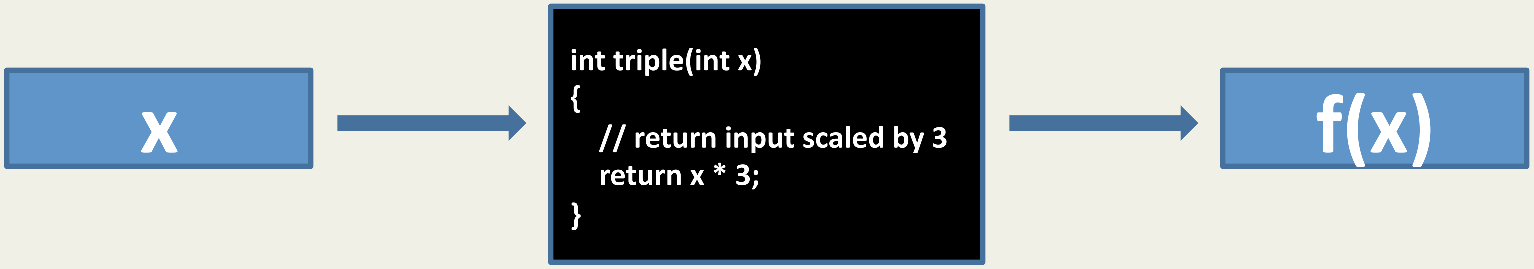
What's a Function?

- Grouped lines of code with a unified purpose.
- A 'black box'. Accepts input, returns output.



What’s a Function?

- Grouped lines of code with a unified purpose.
- A ‘black box’. Accepts input, returns output.



```
int triple(int x)
{
   // return input scaled by 3
   return x * 3;
}
```

Why Use a Function?

- Organization – related code ‘encapsulated’.

- Reusability – functions can be re-called!

Anatomy of a Function in C

```
<return type>
<function name> (arg1, ..., argn)
{
   // code goes here
}
```

Anatomy of a Function in C

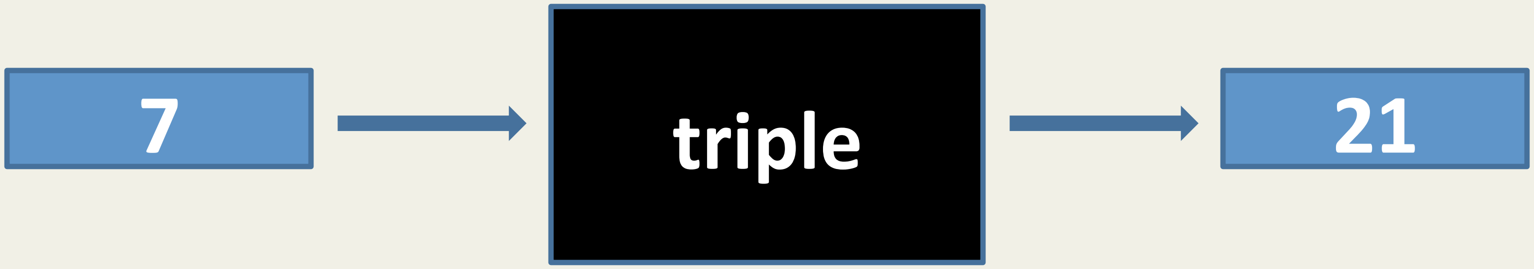
```
<return type>
<function name> (arg1, ..., argn)
{
   // code goes here
}
```

Header

Body

Anatomy of a Function in C

```
int
triple (int x)
{
   int y = x * 3;
   return y;
}
```



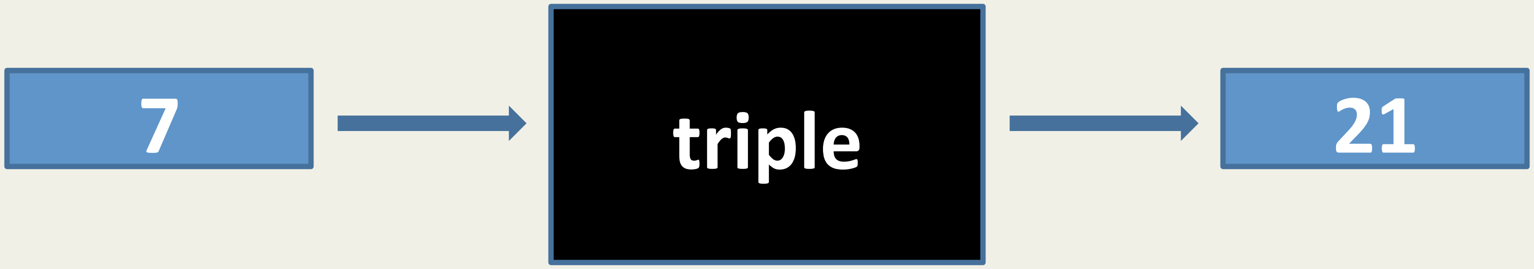
7

triple

21

Anatomy of a Function in C

```
int
triple (int x)
{
   return x * 3;
}
```



7

triple

21

Sample Function Call

```
int
main(int argc, char*
   argv[])
{
   int x = 5;
   int y = triple(x);
   // what is y now?
}
```

```
int
triple(int x)
{
   return x * 3;
}
```

Variable Scope

Two Types of Variables:

- Local Variables
  - Declared inside of a function.
  - Exist only within that function.

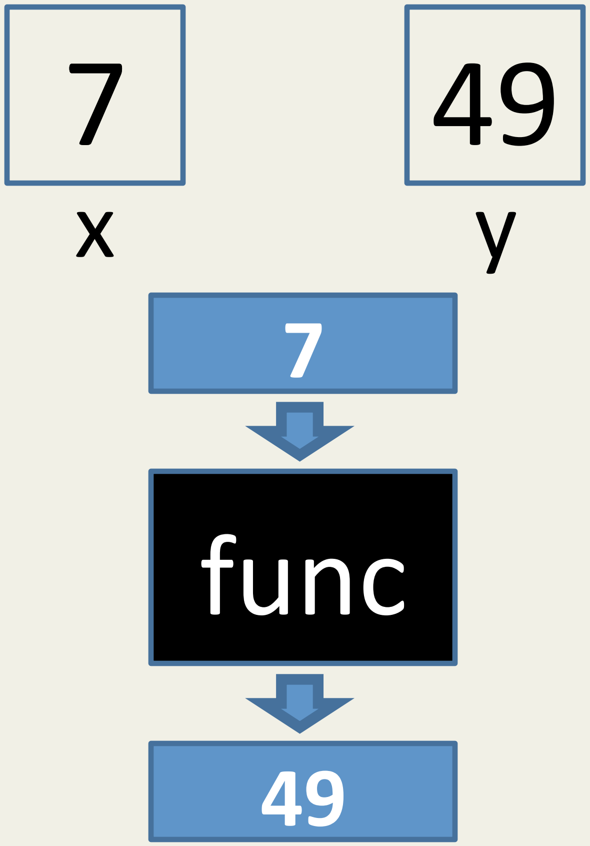
- Global Variables
  - Declared outside of *all functions*.
  - May be accessed or changed from anywhere!

Passing Variables to Functions

- Variables are passed to functions by value.

```
int x = 7;

int y = func(x);
```



7

x

49

y

7

func

49

Sample Function Call 2

```
int
main(int argc, char* argv
   [])
{
   int x = 5;
   int y = triple(x);
   // what is y now?
   // what about x?
}
```

```
int
triple(int x)
{
   x = x * 3;
   return x;
}
```

Sample Function Call 2

```
int
main(int argc, char* argv
   [])
{
   int x = 5;
   int y = triple(x);
   // what is y now?
   // what about x?
}
```

```
int
triple(int val)
{
   val = val * 3;
   return val;
}
```

Local variable is distinct; its name, whether re-used from main or not, is irrelevant!

Magic Numbers

‘Magic number’ – a constant value which is hard-coded into a program.

Magic Numbers

Magic is bad.

Image: Arrested Development, Fox

Magic Number

```
for(int i = 0; i < 8; i++)
{
    // do stuff
}
```

This is bad.

```
#define NUM_ITERS 8
...
for(int i = 0; i < NUM_ITERS; i++)
{
    // do stuff
}
```

This is better.

Arrays

- Data structures which hold sets of same types of values.

- Allows multiple related values to be stored under one name.

Arrays

```
int numbers[4];

numbers[0] = 7;
numbers[1] = 8;
numbers[3] = 2;
numbers[2] = 5;
```

7	8	5	2
0	1	2	3

numbers

Arrays

- Can also initialize an entire array at once:

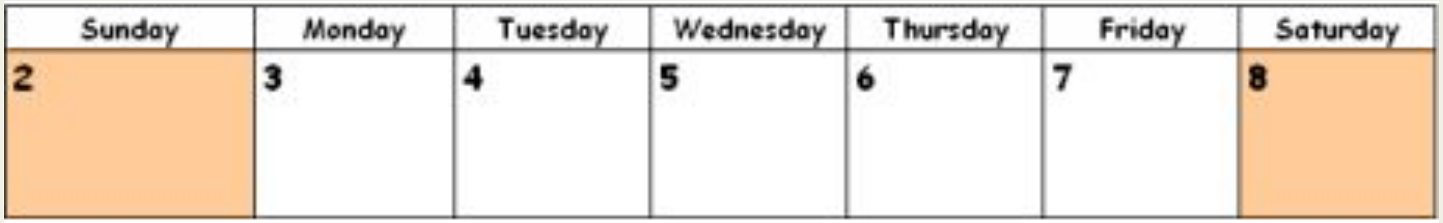
```
int numbers[4] = {7, 8, 5, 2};
```

Multi-Dimensional Arrays

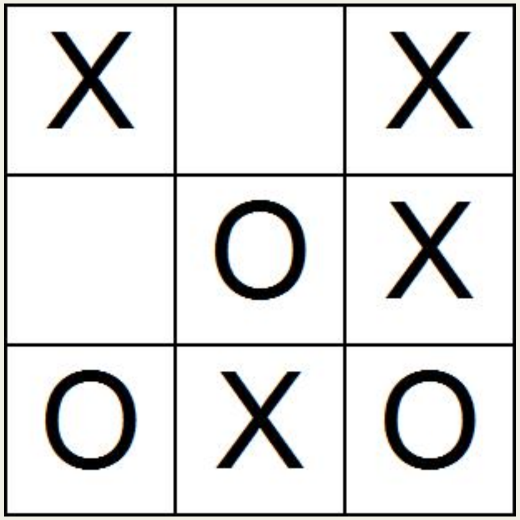
- In single-dimensional case, specify particular element of an array using one index value.
- With multi-dimensional arrays, elements are specified using multiple index values.

Multi-Dimensional Arrays

- Useful when it makes more sense to think of an array in terms of being multi-dimensional.



Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
2	3	4	5	6	7	8



X		X
	O	X
O	X	O

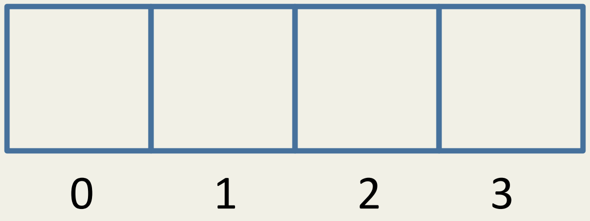
Multi-Dimensional Arrays

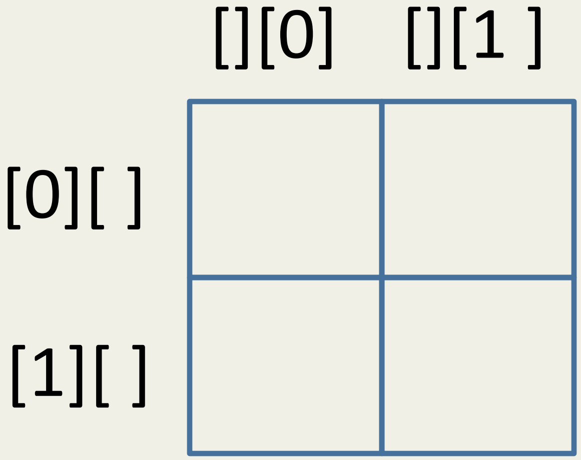
- array[0][0]
- array[0][1]
- array[1][0]
- array[1][1]

	[][0]	[][1]
[0][ ]		
[1][ ]		

Multi-Dimensional Arrays

- We can think of multi-dimensional arrays in geometric terms, but this is irrelevant to the computer.





```
array1D[0] == array2D[0][0]
array1D[1] == array2D[0][1]
array1D[2] == array2D[1][0]
array1D[3] == array2D[1][1]
```

New referencing method, same old data structure!

Passing Arrays to Functions

- Arrays are not primitive data types, rather they are data structures which contain them.
- An array does not have a ‘value’ in the same sense that a primitive data variable does.
- Arrays are passed to functions by ‘reference’, rather than by ‘value’.

Strings

- A string is just an array of chars!
- In C, strings are terminated (ended) by a null character ‘\0’ (backslash-zero).

String plaintext = “Ohai!”;

O	h	a	i	!	\0

Crypto

Caesar Cipher

- Rotate characters by n

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F

Example: Rot 6

Vigenere Cipher

- Given the ‘key’ xyz:

  To encode: Rotate 1st char by x, 2nd by y, 3rd by z, 4th by x, 5th by y…

  To decode: Rotate in the reverse direction.

- Example: Decode my secret password!
  - String: “zypkik2”
  - Key: “secret”

h	u	n	t	e	r	2