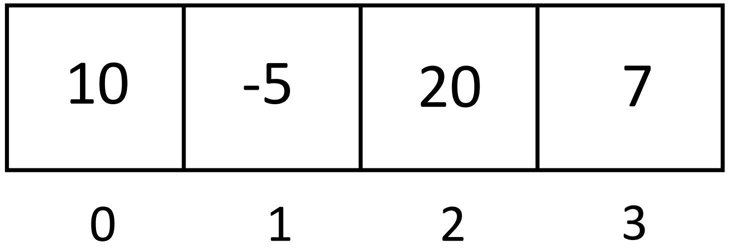
Arrays, Pointers, and Command Line Arguments

Arrays and Pointers



10	-5	20	7
0	1	2	3

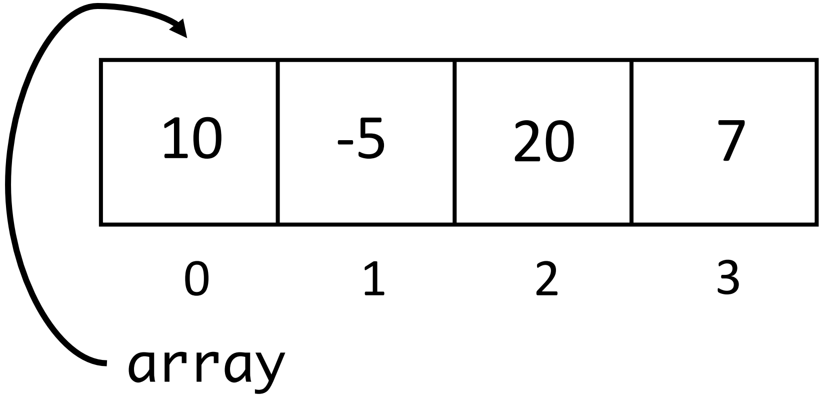
Arrays and Pointers

```
int array[] = { 10, -5, 20, 7 };
```

10	-5	20	7
0	1	2	3

Arrays and Pointers

```
int array[] = { 10, -5, 20, 7 };
```



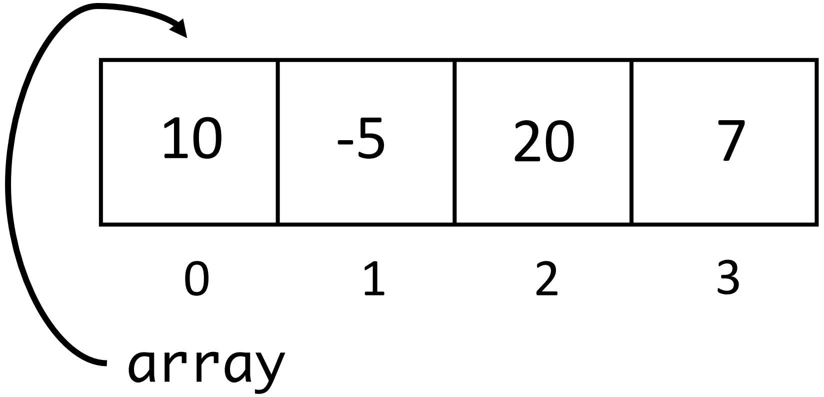
10	-5	20	7
0	1	2	3

array

- Variable array points to the first element of the array

Arrays and Pointers

```
int array[] = { 10, -5, 20, 7 };
```

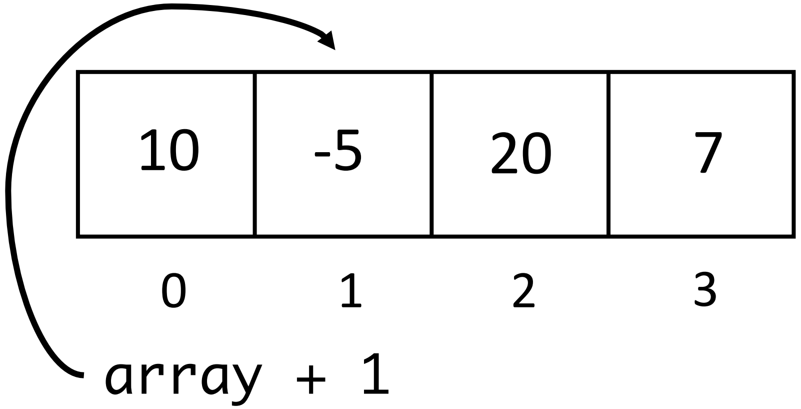


10	-5	20	7
0	1	2	3

- Variable `array` points to the first element of the array
- `*array` accesses the value 10

Arrays and Pointers

```
int array[] = { 10, -5, 20, 7 };
```



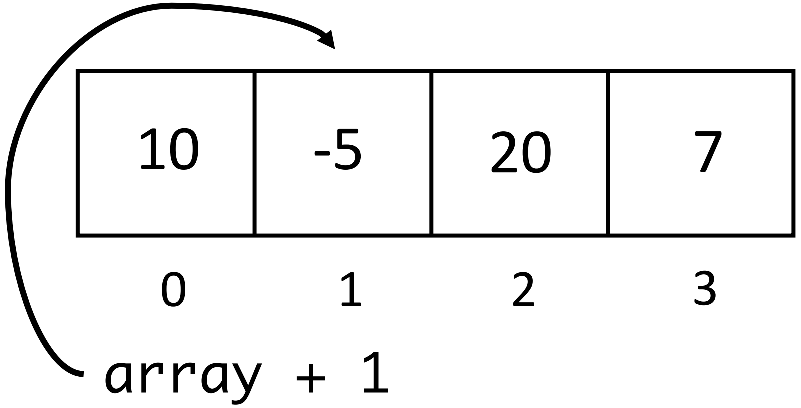
10	-5	20	7
0	1	2	3

`array + 1`

- Variable `array` points to the first element of the array
- `*array` accesses the value 10
- `array + 1` points to the second element of the array

Arrays and Pointers

```
int array[] = { 10, -5, 20, 7 };
```



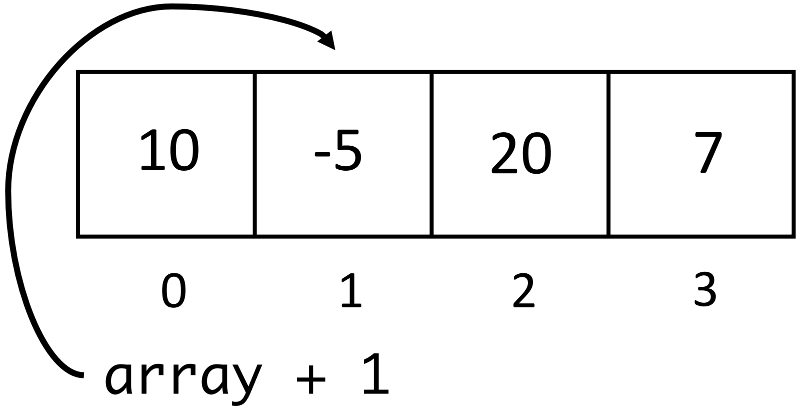
10	-5	20	7
0	1	2	3

`array + 1`

- Variable `array` points to the first element of the array
- `*array` accesses the value 10
- `array + 1` points to the second element of the array
- `*(array + 1)` accesses the value -5

Arrays and Pointers

```
int array[] = { 10, -5, 20, 7 };
```



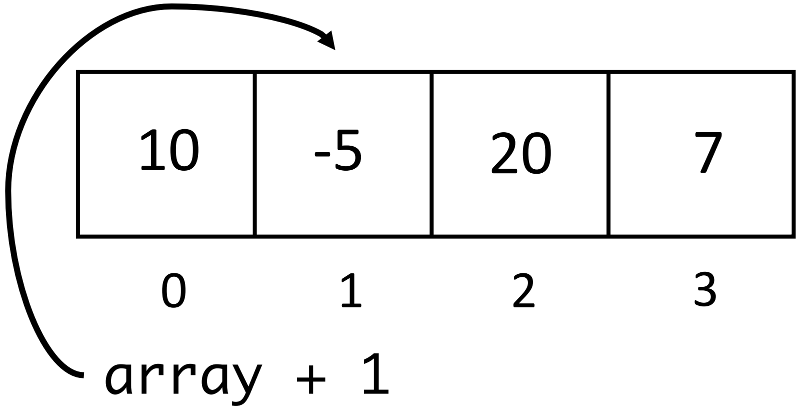
10	-5	20	7
0	1	2	3

```
array + 1
```

- Variable `array` points to the first element of the array
- `*array` accesses the value 10
- `array + 1` points to the second element of the array
- `*(array + 1)` accesses the value -5
  - Equivalent to array[1]

Arrays and Pointers

```
int array[] = { 10, -5, 20, 7 };
```



- Variable `array` points to the first element of the array
- `*array` accesses the value 10
- `array + 1` points to the second element of the array
- `*(array + 1)` accesses the value -5
  - Equivalent to array[1] ← Use this for arrays!

Command Line Arguments

- Providing data to a program when you run it
- `$ gcc prog.c`
  - This has 2 arguments
  - The values of the arguments are the strings “gcc” and “prog.c”
- $ ./prog
  - This has 1 argument
  - The value of the argument is the string “./prog”
- You always have at least one argument
- The operating system provides this information to the main function of our programs

The New main() Function

- In order to receive the data in our main function, we need two additional parameters.
- `int main(int argc, char *argv[]) { … }`
  - `argc` = how many arguments we have
  - `argv` = the string values of each of the arguments
- Strings are arrays of characters
  - `char *` is a pointer to a char data (zero or more characters)
  - `char *variable_name[] is an array of character pointers`
  - For argv, this behaves like a 2d array
    - an array of strings, or an array of character arrays

argv

Assume we run the following program:

```
$ gcc prog.c
```

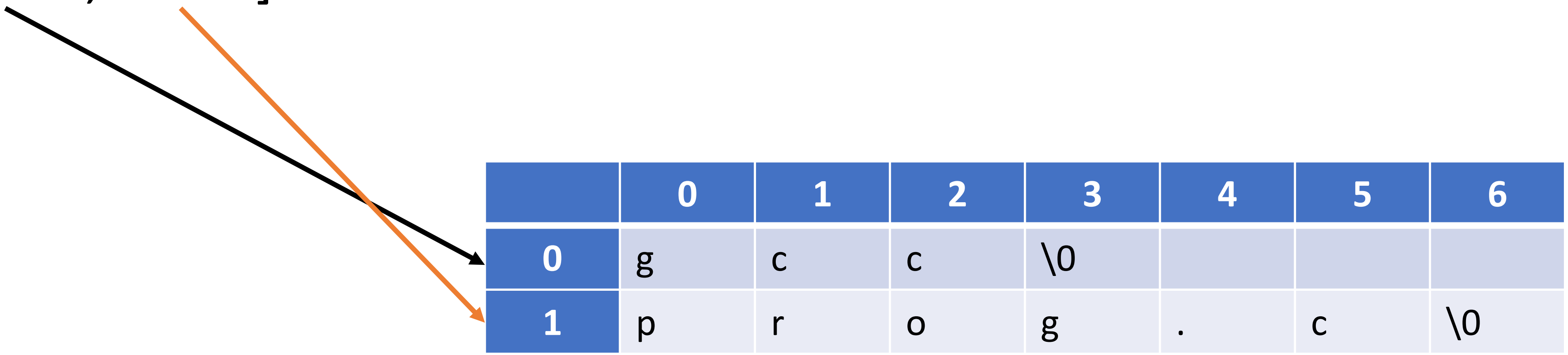
- argv = [“gcc\0”, “prog.c\0”]

argv

Assume we run the following program:

```
$ gcc prog.c
```

- argv = [char*, char*]



	0	1	2	3	4	5	6
0	g	c	c	\0			
1	p	r	o	g	.	c	\0

argv

Assume we run the following program:

```
$ gcc prog.c
```

- argv = [char*, char*]

- argv[0] = “gcc\0”

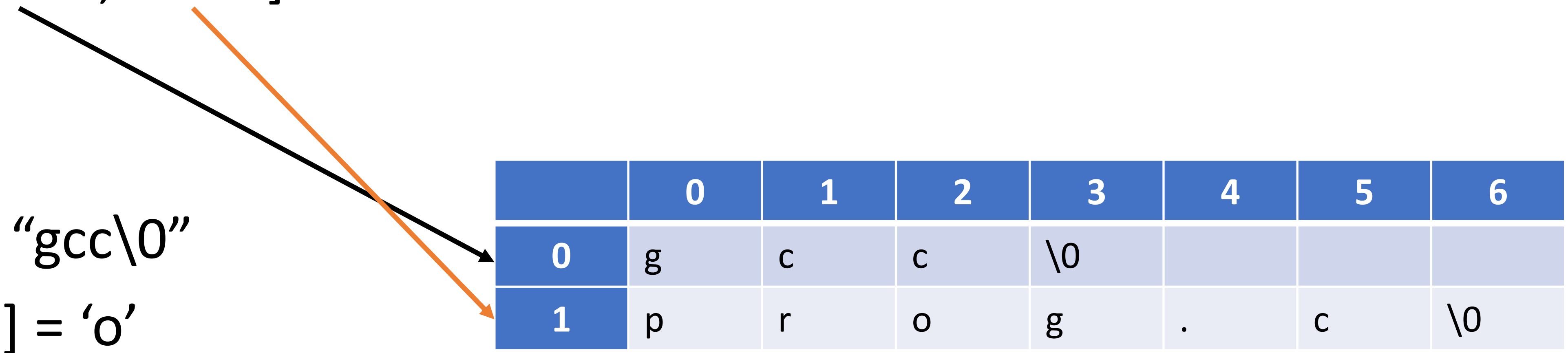
	0	1	2	3	4	5	6
0	g	c	c	\0			
1	p	r	o	g	.	c	\0

argv

Assume we run the following program:

```
$ gcc prog.c
```

- argv = [char*, char*]



	0	1	2	3	4	5	6
0	g	c	c	\0			
1	p	r	o	g	.	c	\0

- argv[0] = “gcc\0”
- argv[1][2] = ‘o’