Pointers, Arrays, Command-line Arguments

Call By Value

```c
#include <stdio.h>

void square_integer(int n);

int main() {
    int x = 10;

    square_integer(x);
    printf("The value of x is: %i\n", x);

    return 0;
}

void square_integer(int n) {
    n = n * n;
}
```

Call By Reference

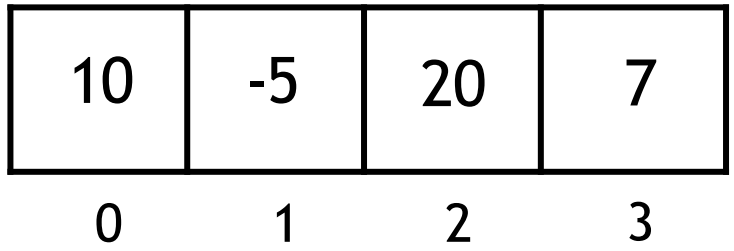
```c
#include <stdio.h>

void square_integer(int *n);

int main() {
    int x = 10;

    square_integer(&x);
    printf("The value of x is: %i\n", x);

    return 0;
}

void square_integer(int *n) {
    *n = (*n) * (*n);
}
```

Array Parameters

```c
#include <stdio.h>

void fill_with_squares(int array[], size_t size);

void print_array(const int array[], size_t size);

int main() {
    int array[] = {1, -1, 3};

    fill_with_squares(array, 3);

    printf("Here are the new contents of the array:\n");
    print_array(array, 3);
    printf("\n");

    return 0;
}
```

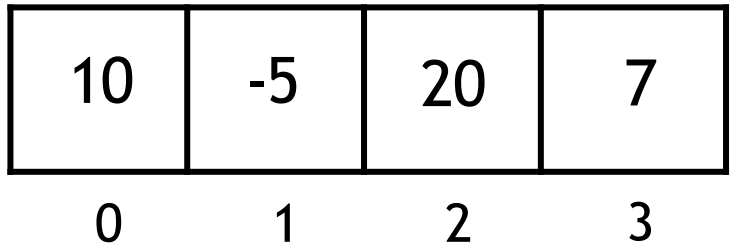
```c
void fill_with_squares(int array[], size_t size) {
    for (size_t i = 0; i < size; i++) {
        array[i] = i * i;
    }
}

void print_array(const int array[], size_t size) {
    for (size_t i = 0; i < size; i++) {
        printf("%i ", array[i]);
    }
}
```

Arrays and Pointers



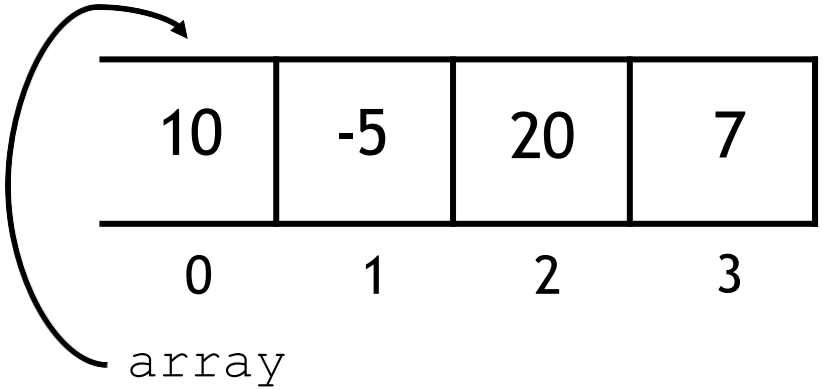
Arrays and Pointers

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



Arrays and Pointers

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

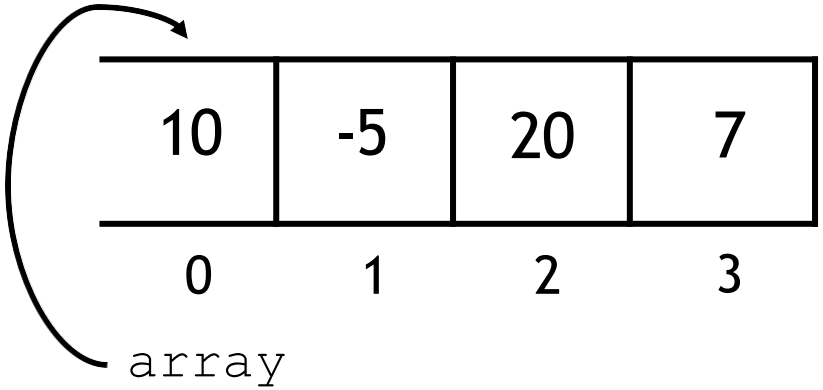


- Variable `array` points to the zeroth element of the array

```c
#include <stdio.h>

void fill_with_squares(int array[], size_t size);

void print_array(const int array[], size_t size);

int main() {
    int array[] = {1, -1, 3};

    fill_with_squares(array, 3);

    printf("Here are the new contents of the array:\n");
    print_array(array, 3);
    printf("\n");

    return 0;
}
```

Arrays and Pointers

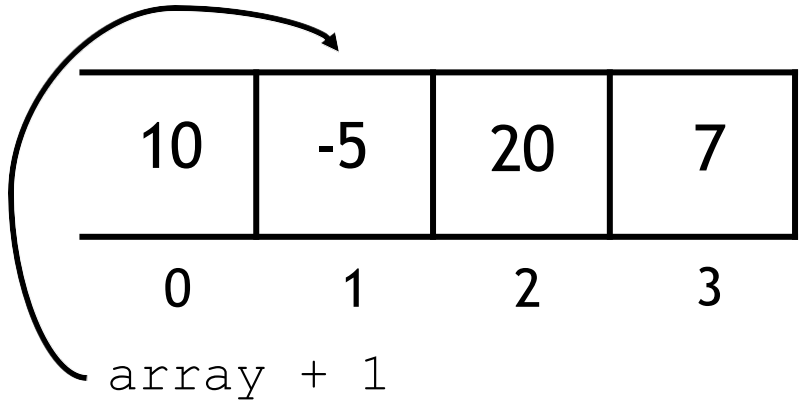
```
int array[] = { 10, -5, 20, 7 };
```



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

Arrays and Pointers

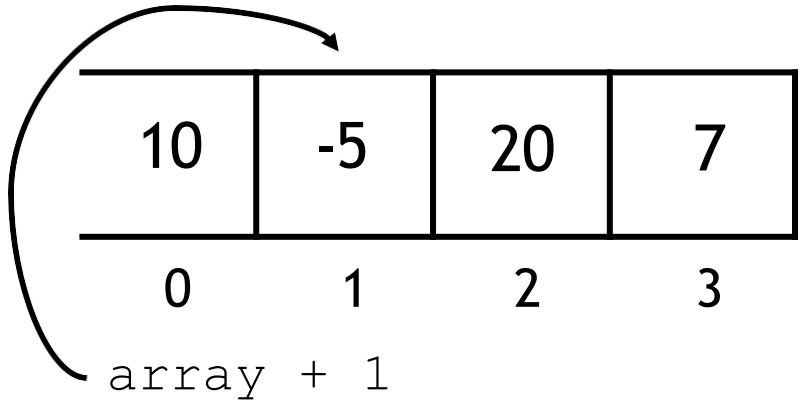
```
int array[] = { 10, -5, 20, 7 };
```



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

Arrays and Pointers

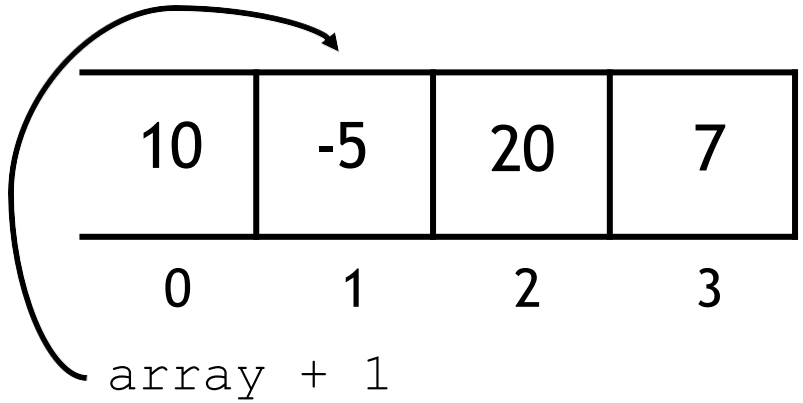
```
int array[] = { 10, -5, 20, 7 };
```



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

Arrays and Pointers

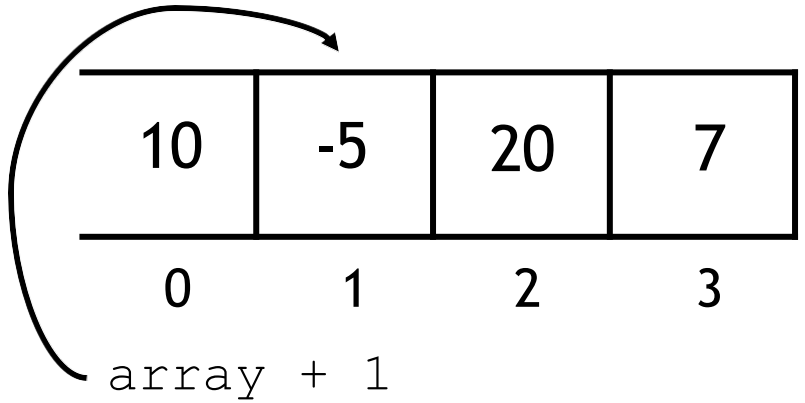
```
int array[] = { 10, -5, 20, 7 };
```



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

Array Parameters

```c
#include <stdio.h>

void print_array(int array[], size_t size);

int main() {
    size_t size = 4;
    int array[] = {10, -3, 14, 2};

    printf("The zeroth element is:%i\n", array[0]);
    printf("The first element is:%i\n", array[1]);

    print_array(array, size);
    printf("\n");

    return 0;
}
```

```c
void print_array(int array[], size_t size) {
    for (size_t i = 0; i < size; i++) {
        printf("%i ", array[i]);
    }
}
```

Array Parameters

```c
#include <stdio.h>

void print_array(int *array, size_t size);

int main() {
    size_t size = 4;
    int array[] = {10, -3, 14, 2};

    printf("The first element is %i\n", *array);
    printf("The second element is %i\n", *(array + 1));

    print_array(array, size);

    return 0;
}
```

```c
void print_array(int *array, size_t size) {
    for (size_t i = 0; i < size; i++) {
        printf("%i\n", array[i]);
    }
}
```

Command Line Arguments

```c
int main(int argc, char **argv) {
    printf("Here are the command line arguments:\n");
    for (int i = 0; i < argc; i++) {
        printf("%s\n", argv[i]);
    }

    printf("\n");

    printf("Here are the individual characters from argv[0]:\n");
    size_t length = strlen(argv[0]);
    for (size_t i = 0; i < length; i++) {
        printf("%c\n", argv[0][i]);
    }

    return 0;
}
```

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, so they are the type `char *`
  - `char*` is a pointer to a char
  - `char**` is a pointer to a pointer of a char
  - in effect a `char**` behaves like a 2d array
    - an array of strings, or an array of character arrays

argv

Assumewe run the following program:

```
$ gcc prog.c
```

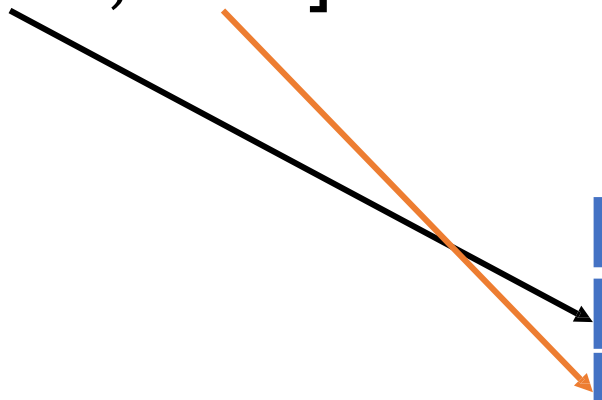
- argv = [“gcc\0”, “prog.c\0”]

argv

Assumewe run the following program:

```
$ gcc prog.c
```

- argv = [char*, char*]



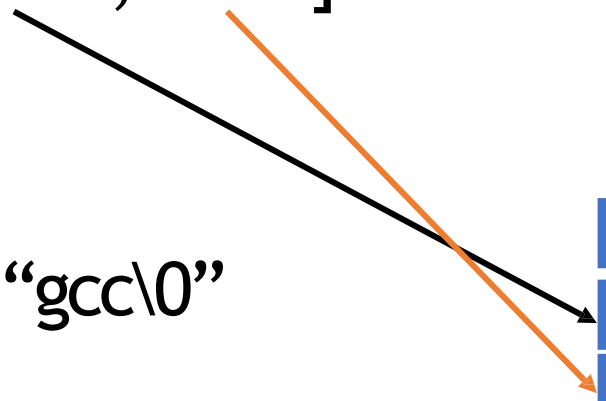
	0	1	2	3	4	5	6
0	g	c	c	\0			
1	p	r	o	g	.	c	\0

argv

Assumewe run the following program:

```
$ gcc prog.c
```

- argv = [char*, char*]
- argv[0] = “gcc\0”

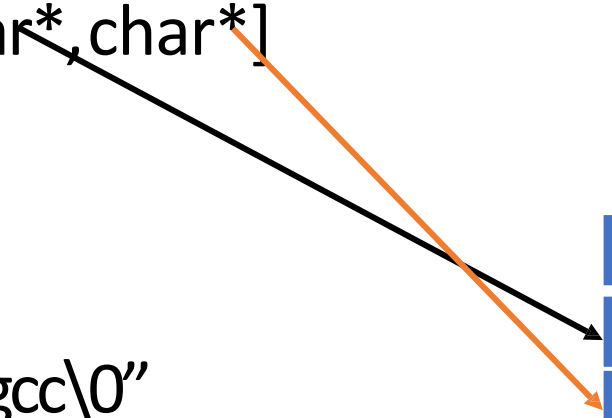


	0	1	2	3	4	5	6
0	g	c	c	\0			
1	p	r	o	g	.	c	\0

argv

- Assumewe 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’