arrays

Declare

```
int a[3];
```

Initialize

```
int a[3] = {10, 20 , 30};

// remaining elements initialized to zero
int b[4] = {40, 50};

// all elements zero
int c[5] = {0};
```

Other Ways to Initialize

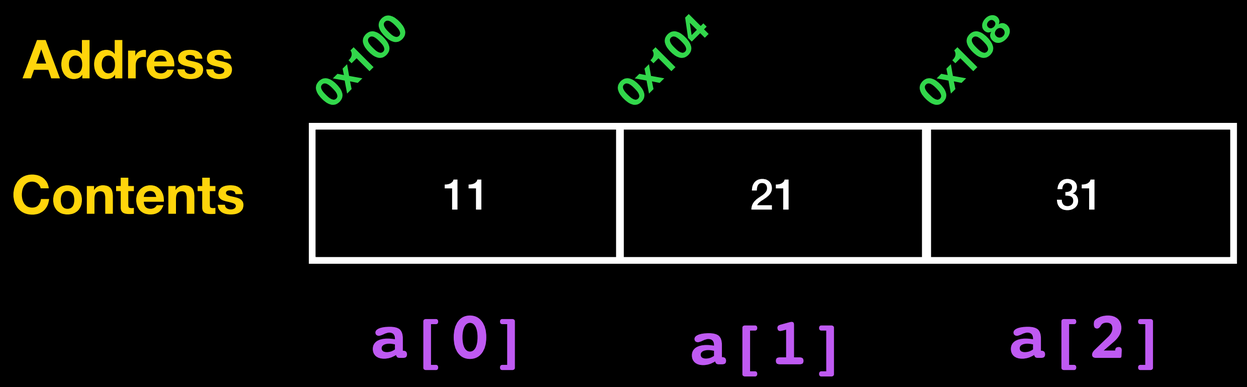
```
// compiler determines array size
int a[] = {1, 2, 3};

// initialize specific elements; all others zero'd
int b[10] = {[5] = 20, [7] = 40};

// explicitly zero
int c[20];
memset(c, 0x00, sizeof(c));
```

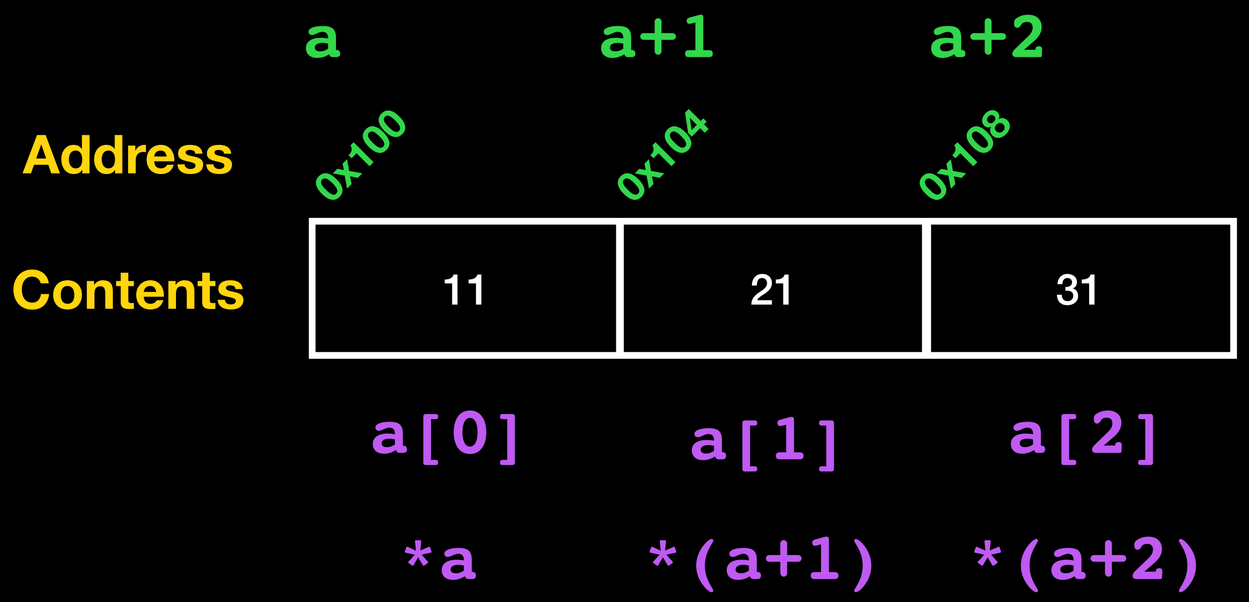
Memory representation

```
int a[3] = {11, 21, 31};
```



Array name == base address

```
int a[3] = {11, 21, 31};
```



Passing arrays to functions

```
void
inc_all(char a[], size_t n)
{
    size_t i;
    for (i = 0; i < n; i++)
        a[i] += 1;
}

int
main(void)
{
    int a[] = {11, 21, 31};

    inc_all(a, 3);
    /* … */
}
```

Passing arrays to functions

```
void
inc_all(char *a, size_t n)
{
    size_t i;
    for (i = 0; i < n; i++)
        a[i] += 1;
}

int
main(void)
{
    int a[] = {11, 21, 31};

    inc_all(a, 3);
    /* … */
}
```

Dynamic arrays

```
#include <stdlib.h>

int main(void)
{
    int *a, *b;

    a = malloc(10 * sizeof(int)); // 10 element array
    b = calloc(10, sizeof(int));  // same, but also zeroes

    free(a);
    free(b);

    /* … */
}
```

Arrays vs. pointers

```
#include <stdio.h>

int main(void)
{
    int a[5];
    int *p =a;


    printf(“%zu\n”, sizeof(a));  // 20
    printf(“%zu\n”, sizeof(p));  // 8

    /* … */
}
```