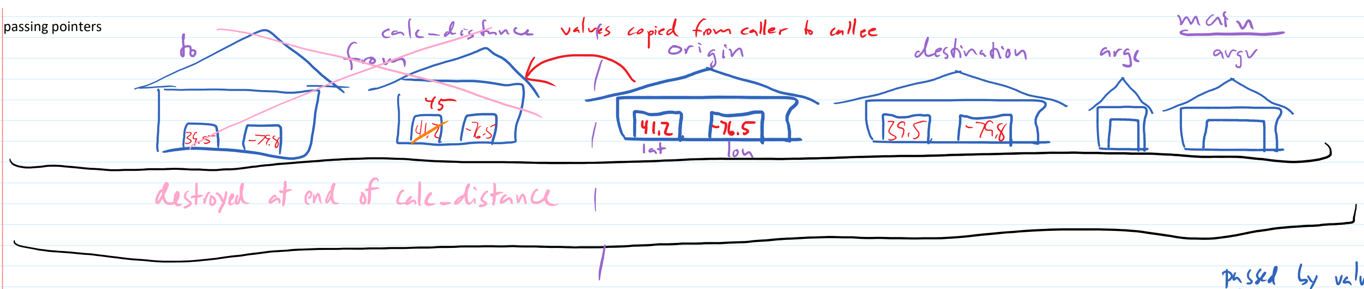
passing pointers



passed by value

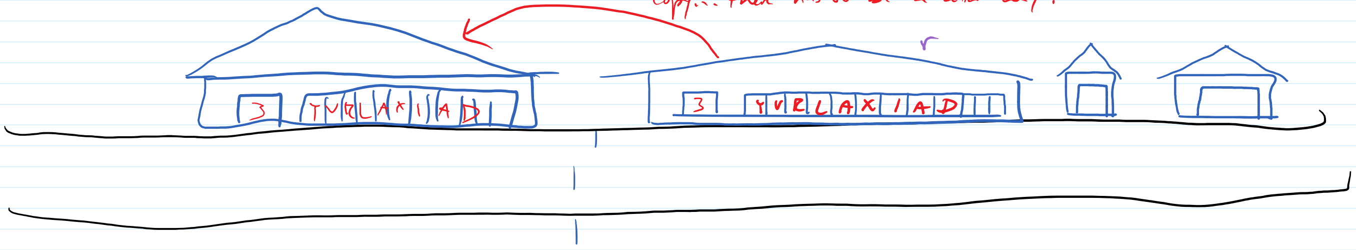
```
typedef struct {
  double lat;
  double lon;
} location;
```

```
int main()
{
  location origin = {41.2, -76.5};
  location destination = {39.5, -79.8};

  printf("%f\n", calc_distance(origin, destination));
}
```

```
double calc_distance(location from, location to)
{
  from.lat = 45;
  ...
}
```

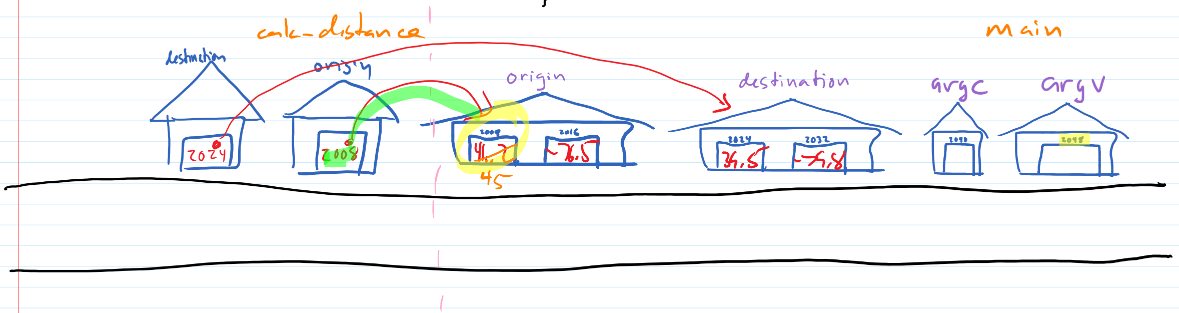
changes the copy, not the caller's original

it was expensive to build this copy... there has to be a better way!



```
typedef struct {
  int len;
  char segments[12];
} route;
```

```
void process(route rt);

int main()
{
  route r;
  ...
  process(r);
}
```



pass addresses of (pointers to) the structs

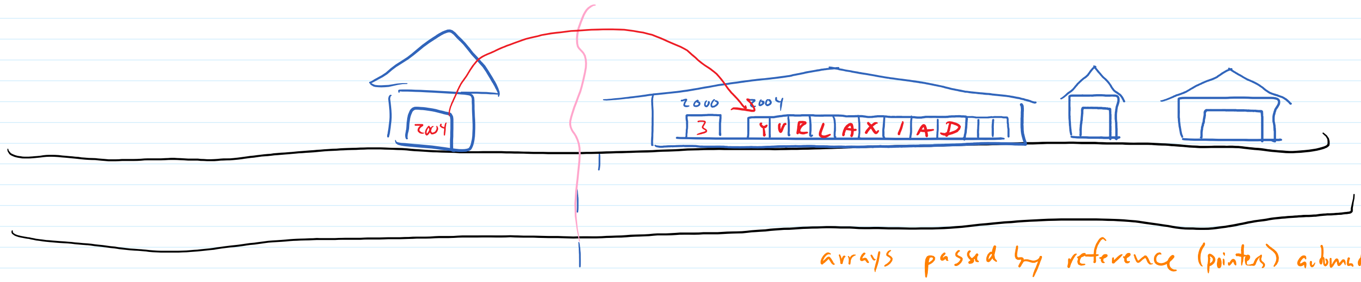
```
typedef struct {
  double lat;
  double lon;
} location;
```

```
int main()
{
  location origin = {41.2, -76.5};
  location destination = {39.5, -79.8};

  printf("%f\n", calc_distance(&origin, &destination));
}
```

```
double calc_distance(location *from, location *to)
{
  from->lat = 45;
}
```

changes the caller's original

follow the pointer and select lat field

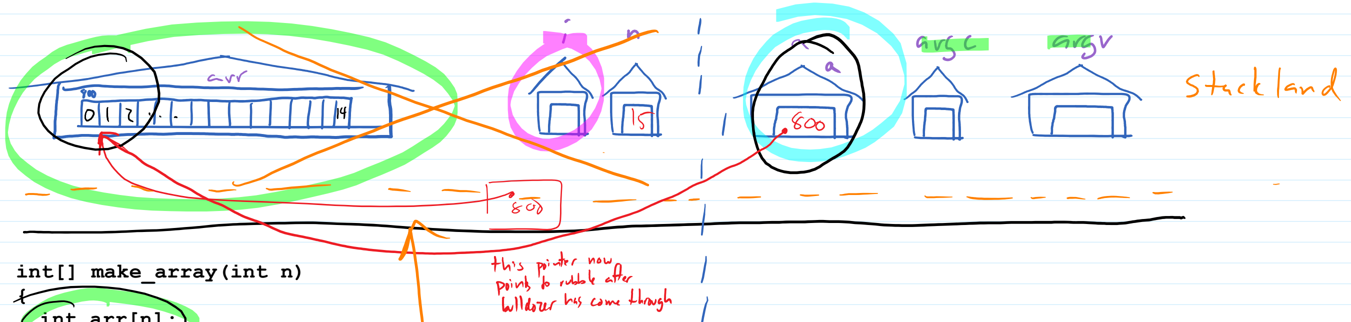


arrays passed by reference (pointers) automatically!

```
typedef struct {
  int len;
  char segments[12];
} route;
```

```
void process(char route[]);

int main()
{
  route r;
```

```
    ...
    process(r.segments);
  }
```

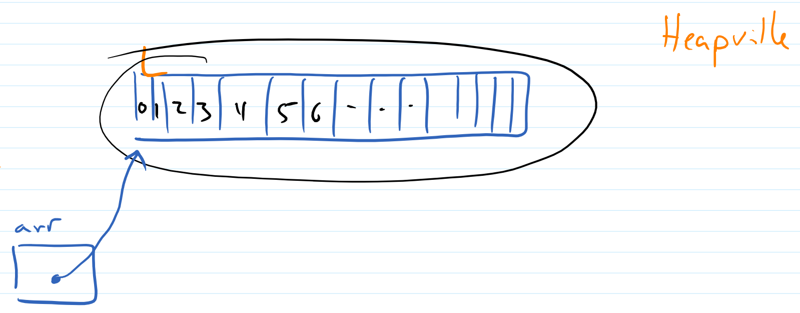


arr
900
0 1 2 . . 14
i
n
15
a
800
argc
argv
Stackland
800
this pointer now points to rubble after bulldozer has come through

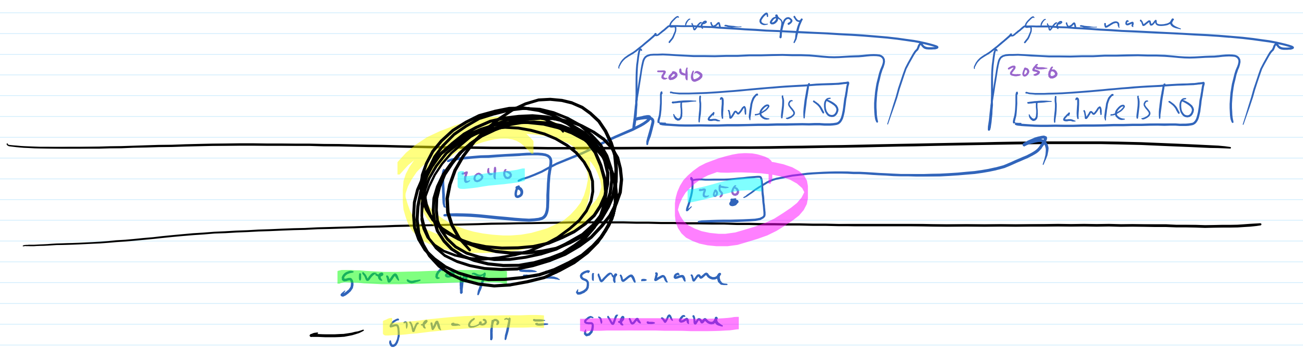
```
int[] make_array(int n)
{
  int arr[n];
  for (int i = 0; i < n; i++)
  {
    arr[i] = i;
  }

  return arr;
}
```

a bulldozer runs at } ending fxn to destroy fxn's stack frame

```
int main(int argc, char *argv[])
{
  int *a = make_array(15);
  printf("%d\n", sum1D(arr));
  free(arr);
}
```



Heapville
0 1 2 3 4 5 6 . . .
arr



given_copy
given_name
2040
J a m e s \0
2050
J a m e s \0
2040
0
2050
given_copy == given_name
given_copy = given_name