Structs:

- structs within structs
- design & implementation using structs
- structs as return values
- assignments to structs

Assignment to structs

```
struct Product {
    string name;
    float price;
} prod1;
```

You can do assignment field-by-field

```
prod1.name = "drone";
prod1.price = 249.59;
```

} Field assignment

Or you can do it in one assignment, using { , }

```
prod1 = { "drone", 249.59 };
Product prod2;
prod2 = prod1;   // Now prod2 has same values.
                 // as prod1.
```

struct assignment

However...

You can only use struct assignment if all fields of the struct can use =

Note: Can't use = on arrays. (except with literals)

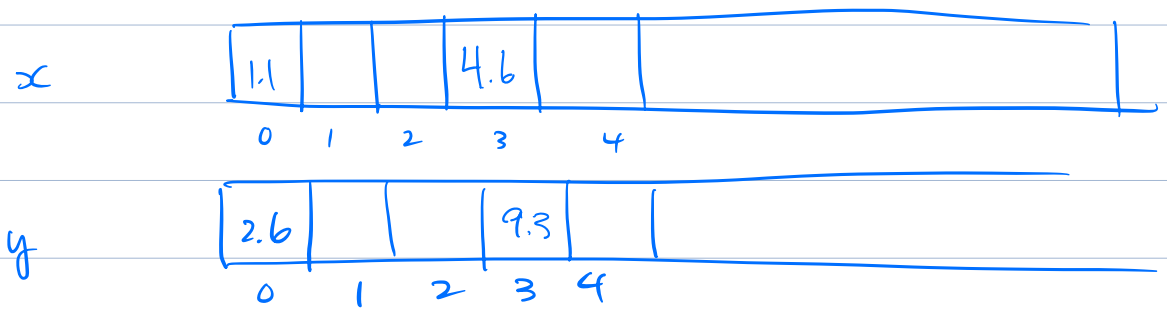
```
int arr[5] = { 1, 2, 3, 4, 5 };

int arr2[5];

arr2 = arr;
```

error: invalid array assignment

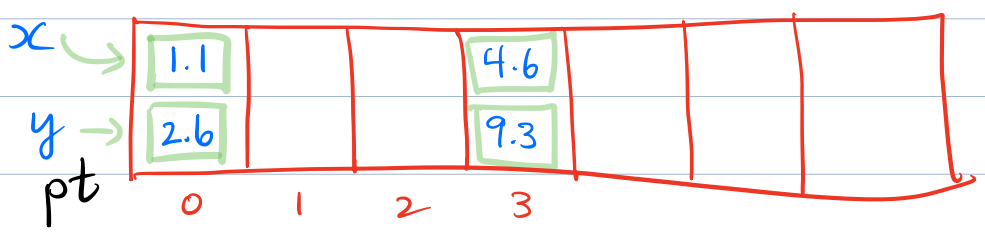
Breaking news: It didn't work an straight arrays, but it worked on structs that contain arrays! (contrary to expectation.)

Returning a struct from a function call

- Can be done
- part of usefulness of structs



// to access 0th point, refer to x[0], y[0]
//

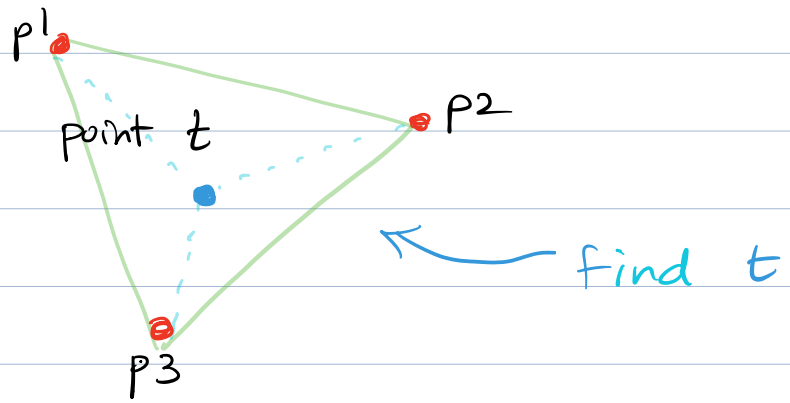


// to access 0th point, refer to pt[0].x, pt[0].y.

// or just pt[0]

```
struct Point {
    float x;
    float y;
};
```

```
Point triangulate (Point p1, Point p2, Point p3)
{
```




```
    return t;
}
```

in main ...

```
Point centre = triangulation(pt[0], pt[1], pt[2]);
```

```
void getPoint ( Point &p );

int main()
{
    Point x;
    getPoint(x);
    return 0;
}.

void getPoint(Point &p)
{
    cout << "Enter x-coordinate: ";
    cin >> p.x;
    cout << "Enter y-coordinate: ";
    cin >> p.y;
}
```