n triangles → a, b, c → sorted by their areas!

$$S = \sqrt{P \times (P-a) \times (P-b)(P-c)}$$

$$P = \frac{a+b+c}{2}$$

```
struct triangle {
    int a;
    int b;
    int c;
};
```
12 bytes

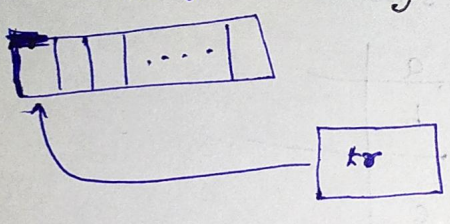
```
int main() {
    int n;
    scanf("%d", &n);
    triangle *tr = malloc(n * sizeof(triangle));
```

(3 * 12 bytes) → 36 bytes!

Pointer

triangle *tr
↳ 12 bytes.
So actually 3

```
    for (int i = 0; i < n; i++) {
        scanf("%d %d %d\n",
              &tr[i].a, &tr[i].b, &tr[i].c);
    }
```



(triangle *) tr → 12 bytes

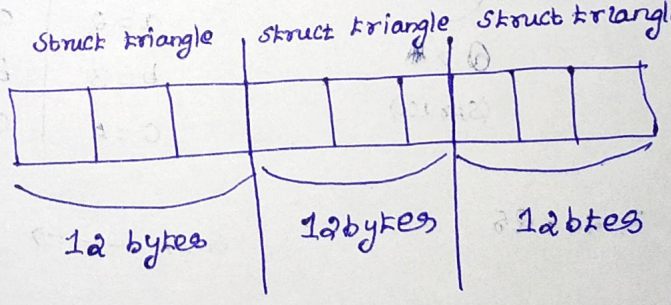
((triangle *)tr + 1) → 13th byte address

((triangle *)tr + 2) → 25th byte address.

tr[0] → 0th element (4 bytes - 1 element)
0 → (a)

tr[1] → 1st element
1 → (a)

tr[2] → 2nd element
2 → a



1st block → for struct 1
2nd block → for struct 2
3rd block → for struct 3.

triangle * → 1 block = 12 bytes

tr[0] → 1st block
tr[1] → 2nd block
tr[2] → 3rd block.

arr[0].a / ptr[0].a ⟶ a th data

we need — store in address scanf

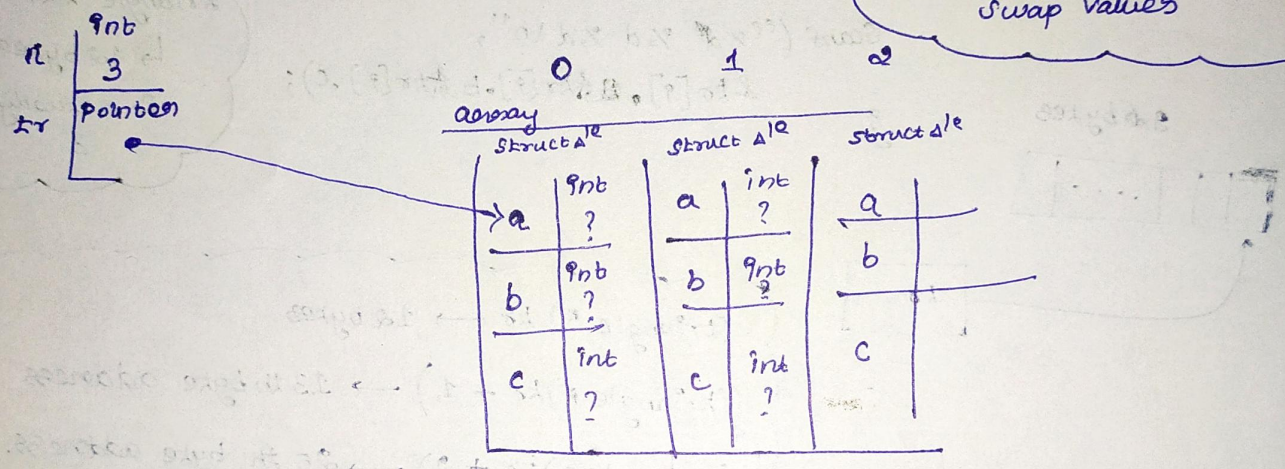
scanf ("%d %d %d \n", &ptr[i].a, &ptr[i].b, &ptr[i].c);

Sort_by_area (ptr, n);

---

* I'm getting ⟶ ptr as i/p

only choice swap!
Can't swap completely — swap values



Say:

a = 5	a = 2
b = 6	b = 3
c = 3	c = 4

a > b (struct)

Swap

a = 2	a = 5
b = 3	b = 6
c = 4	c = 3

---

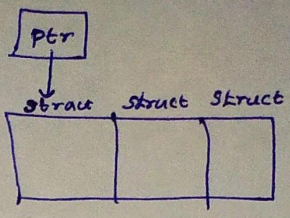
a = 7, 24, 25

Sort ⟶ area

Compare

↳ Swap if necessary!

```
float triangleArea (triangle *tr, int n)
{
    float p = (float)(tr[n].a + tr[n].b + tr[n].c)/2.0 ;
    float result = p*(p-tr[n].a) * (p-tr[n].b) *(p-tr[n].c);
    return (sqrt(result));
}
```



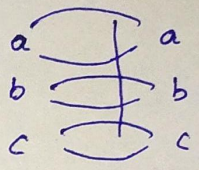
// $S = \sqrt{p*(p-a)*(p-b)*(p-c)}$

$$p = \frac{a+b+c}{3}$$

```
void swap (triangle *tr, int i, int j)
{
    int temp;
    temp = tr[i].a ;
    tr[i].a = tr[j].a ;
    tr[j].a = temp;

    temp = tr[i].b ;
    tr[i].b = tr[j].b;
    tr[j].b = temp;

    temp = tr[i].c ;
    tr[i].c = tr[j].c;
    tr[j].c = temp;
}
```




```
void sort_by_area (triangle *tr, int n) {
    float area1, area2 ;
    int i, j;
    for (i=0; i < n-1; i++)
    {
        for (j=i+1; j < n; j++)
        {
            area1 = triangleArea (tr, i);
            area2 = triangleArea(tr, j);
            if (area1 > area2)
                swap (tr, i, j) ;
        }
    }
```

done! for good!