01 Typedef, Struct Initialization

Struct without Typedef

Struct with Typedef

Passing using pointers

01 Typedef, Struct Initialization

Struct without Typedef

```c
#include<stdio.h>
struct rectangle {
    float length;
    float breadth;
};

float compute_area(struct rectangle r) {
    return r.length * r.breadth;
}

void print_rectangle(struct rectangle r) {
    printf("Rectangle with length %f and breadth %f\n", r.length, r.breadth);
}

int main()
{
    struct rectangle rect = { 1.5, 3.2 }; // Initializer
    print_rectangle(rect);
```

```
    printf("Area of the rectangle is %f \n", compute_area(rect) );
}
```

Struct with Typedef

```
#include<stdio.h>
typedef struct rectangle {
    float length;
    float breadth;
} rectangle;

float compute_area(rectangle r) {
    return r.length*r.breadth;
}

rectangle scale(rectangle r, float s) {
    r.length = r.length*s;
    r.breadth = r.breadth*s;
    return r;
}

int main()
{
    rectangle rect = {  .breadth = 1.0, .length = 3.0} /* {3.0, 1.0 }*/;
    // rect.length = 3.2;
    // rect.breadth = 1.2;
    printf("Area of the rectangle is %f \n", compute_area(rect));
    rectangle rp = scale(rect, 5);
    printf("Area of the rectangle is %f \n", compute_area(*rp));
    printf("Area of the rectangle is %f \n", compute_area(rect));
```

```
}
```

Passing using pointers

```
#include<stdio.h>
typedef struct rectangle {
    float length;
    float breadth;
} rectangle;

float compute_area(rectangle r) {
    return r.length*r.breadth;
}

rectangle* scale(rectangle* r, float s) {
    r->length = r->length*s;
    r->breadth = r->breadth*s;
    return r;
}

int main()
{
    rectangle rect = {  .breadth = 1.0, .length = 3.0} /* {3.0, 1.0 }*/;
    // rect.length = 3.2;
    // rect.breadth = 1.2;
    printf("Area of the rectangle is %f \n", compute_area(rect));
    rectangle* rp = scale(&rect, 5);
    printf("Area of the rectangle is %f \n", compute_area(*rp));
    printf("Area of the rectangle is %f \n", compute_area(rect));
}
```