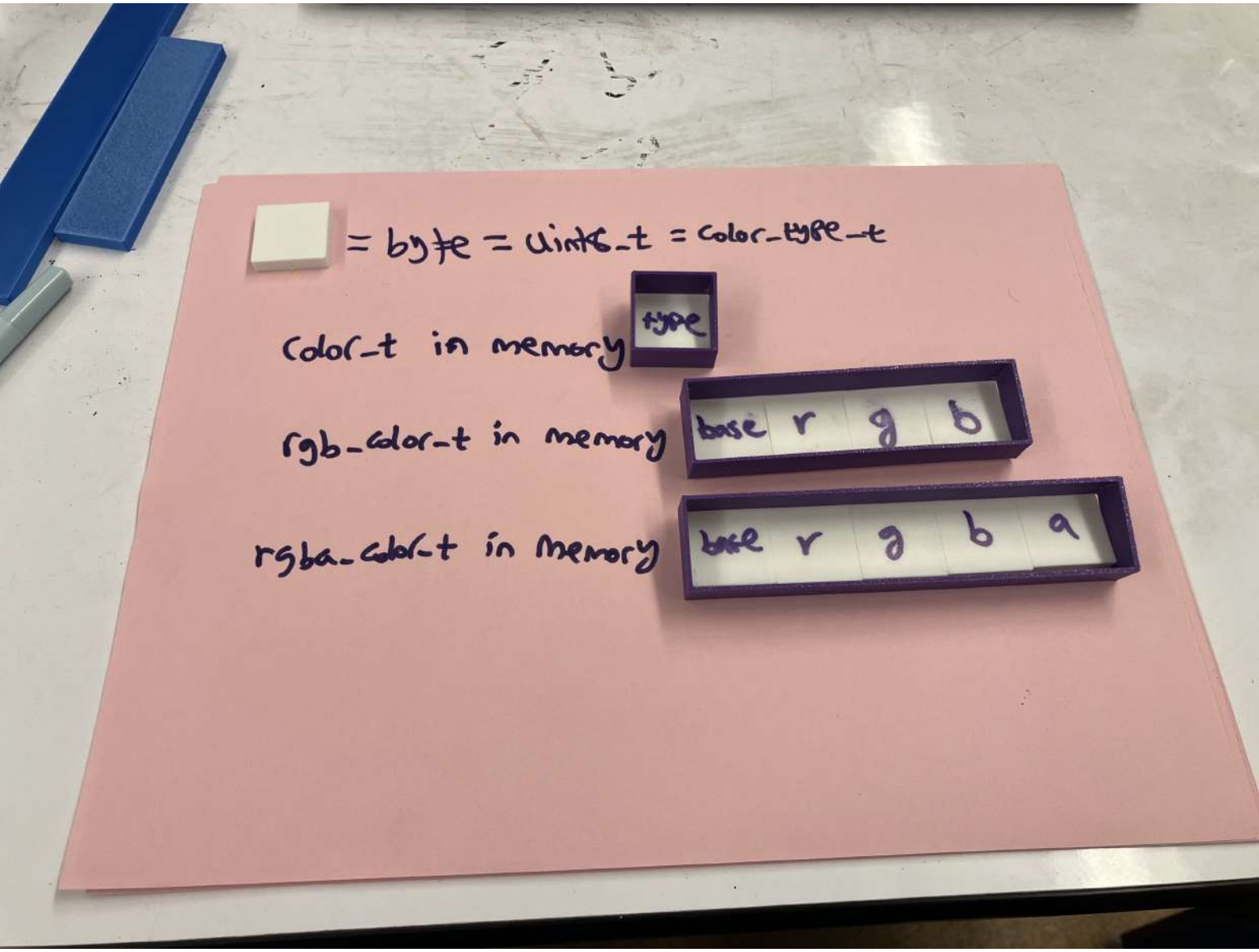
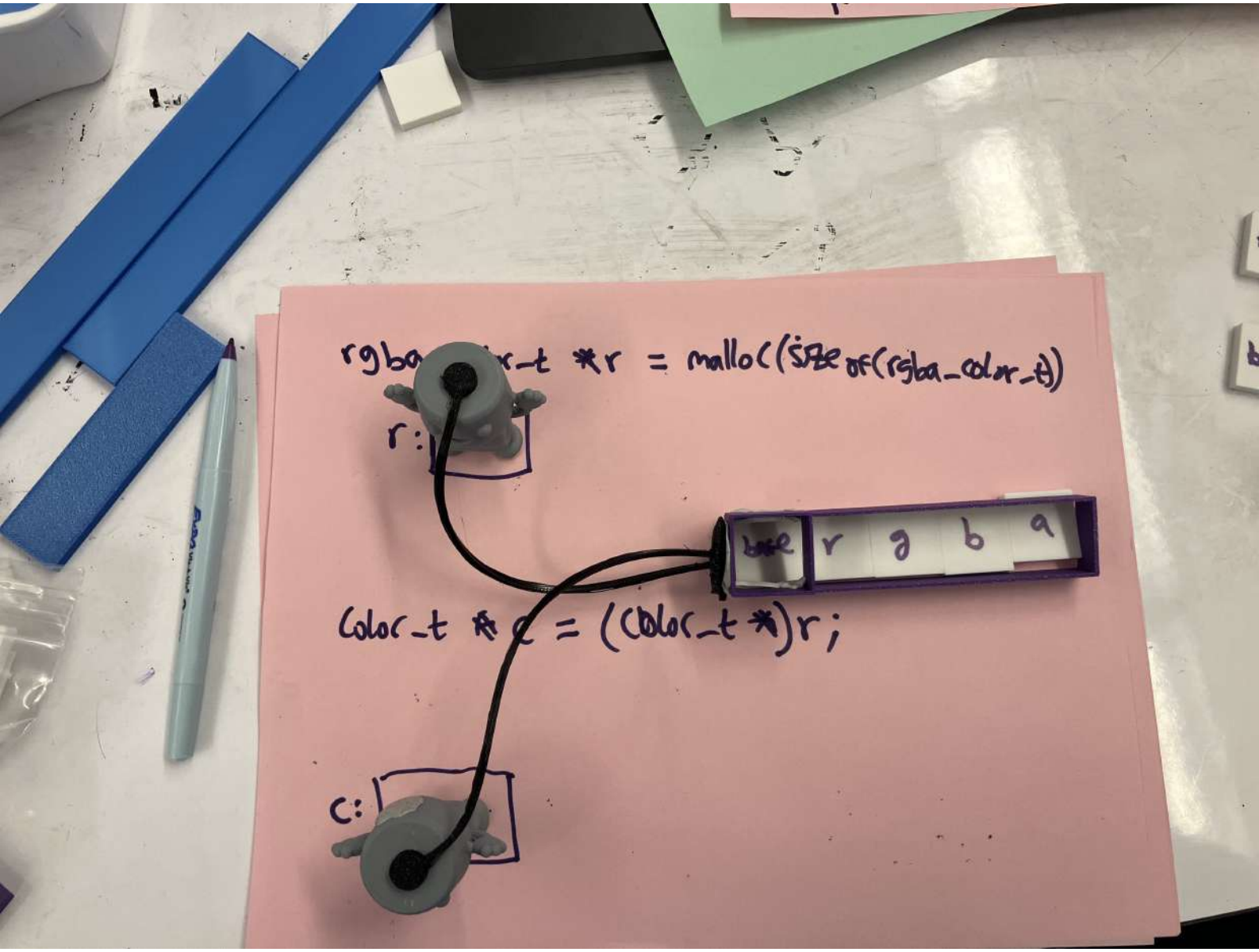

= byte = uint8_t = color_type_t
color_t in memory
type
rgb_color_t in memory
base r g b
rgba_color_t in memory
base r g b a


r:
base
r
g
b
a
Color_t *c = (Color_t *)r;
c:

CS 3: Introduction to Software Design

Faking Inheritence in C Exercises

color.h

```h
typedef enum { COLOR_RGB, COLOR_RGBA } color_type_t;

typedef struct {
  color_type_t type;
} color_t;

typedef struct {
  color_t base;
  uint8_t r;
  uint8_t g;
  uint8_t b;
} rgb_color_t;

typedef struct {
  color_t base;
  uint8_t r;
  uint8_t g;
  uint8_t b;
  uint8_t a;
} rgba_color_t;
```

Using `color_t` Inheritence

Option A

```c
rgba_color_t new = *(rgba_color_t *)&(color_get_random_in_range_rgb());
new.base.type = COLOR_RGBA;
new.a = COLOR_MAX;
```

Option B

```c
rgb_color_t temp = color_get_random_in_range_rgb();
rgba_color_t new = *(rgba_color_t *)&temp;
new.base.type = COLOR_RGBA;
new.a = COLOR_MAX;
```

Option C

```c
rgba_color_t *new = malloc(sizeof(rgba_color_t));
*new = color_get_random_in_range_rgb();
new->base.type = COLOR_RGBA;
new->a = COLOR_MAX;
```

CS 3: Introduction to Software Design

2D Arrays in C Exercises

image.c

```c
typedef struct image {
   size_t width;
   size_t height;
   rgba_color_t **pixels;
} image_t;

image_t *image_init(size_t width, size_t height) {
   image_t *new = malloc(sizeof(image_t));
   new->width = width;
   new->height = height;
   new->pixels = malloc(sizeof(rgba_color_t *) * new->height);
   for (size_t i = 0; i < new->height; i++) {
      new->pixels[i] = malloc(sizeof(rgba_color_t) * new->width);
      for (size_t j = 0; j < new->width; j++) {
         new->pixels[i][j] = (rgba_color_t){.r=0, .g=0, .b=0, .a=0};
         new->pixels[i][j].base.type = COLOR_RGBA;
      }
   }
   return new;
}

void image_free(image_t *img) {




}
```