22AIE101

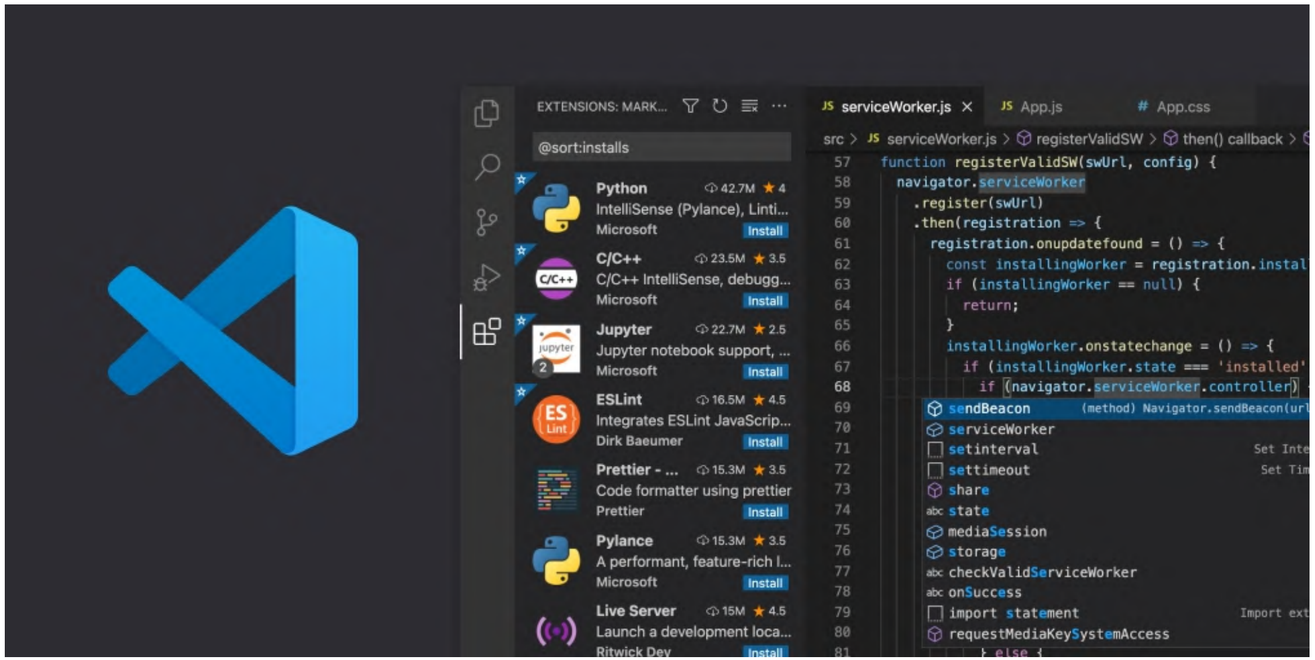
PROBLEM SOLVING AND C PROGRAMMING

PHOTO EDITOR

- Cropping an Image
- Extract the RGB Components of the Image
- Convert the Image to a Grayscale image
- Convert the image to Black and White Image
- Convert the image to Sepia colour format
- Interchange the components of the image

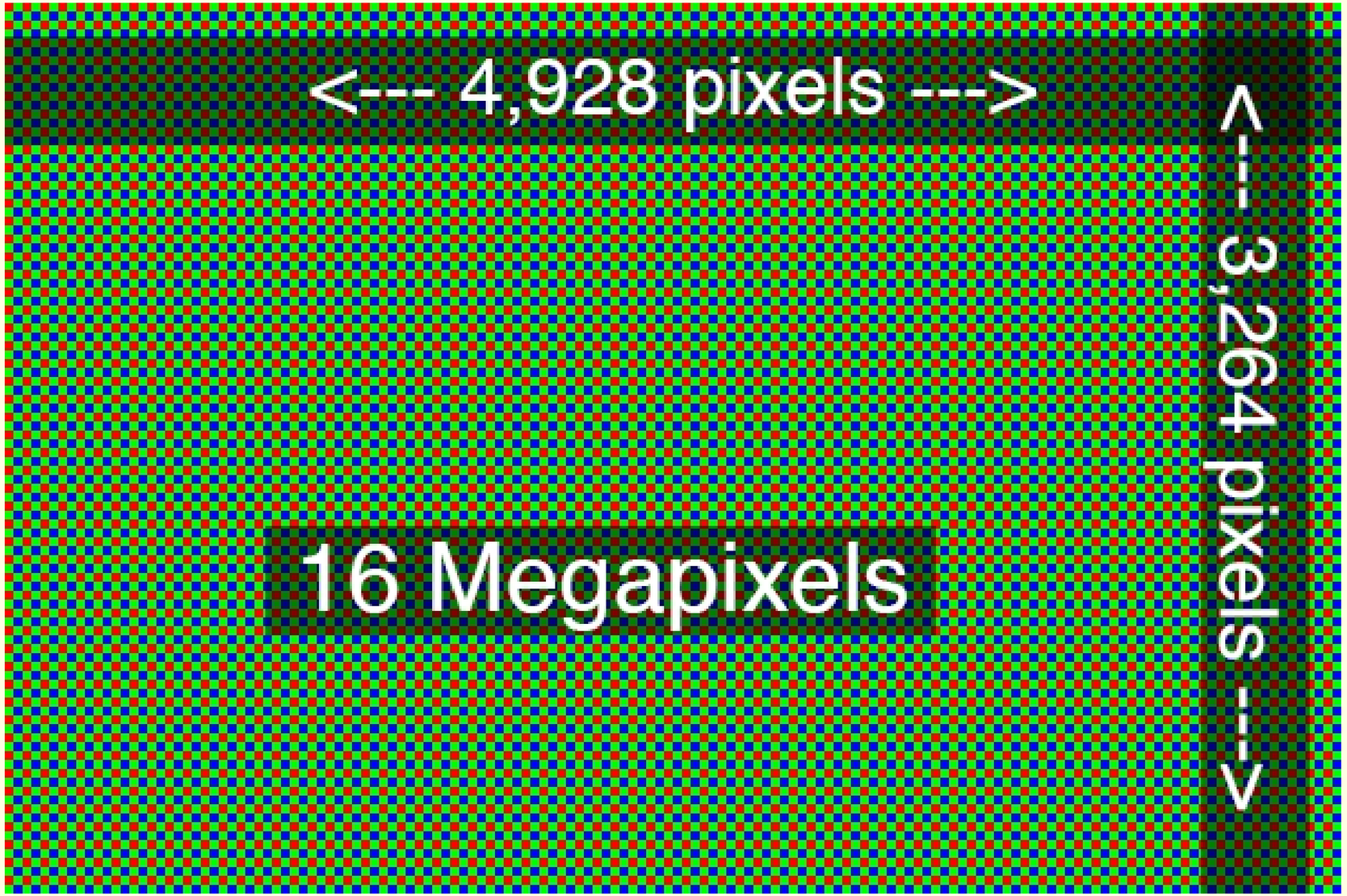
INTRODUCTION

- C Language is one of the most primitive and low-level programming languages used.
- The platform used here is VS Code



BASICS

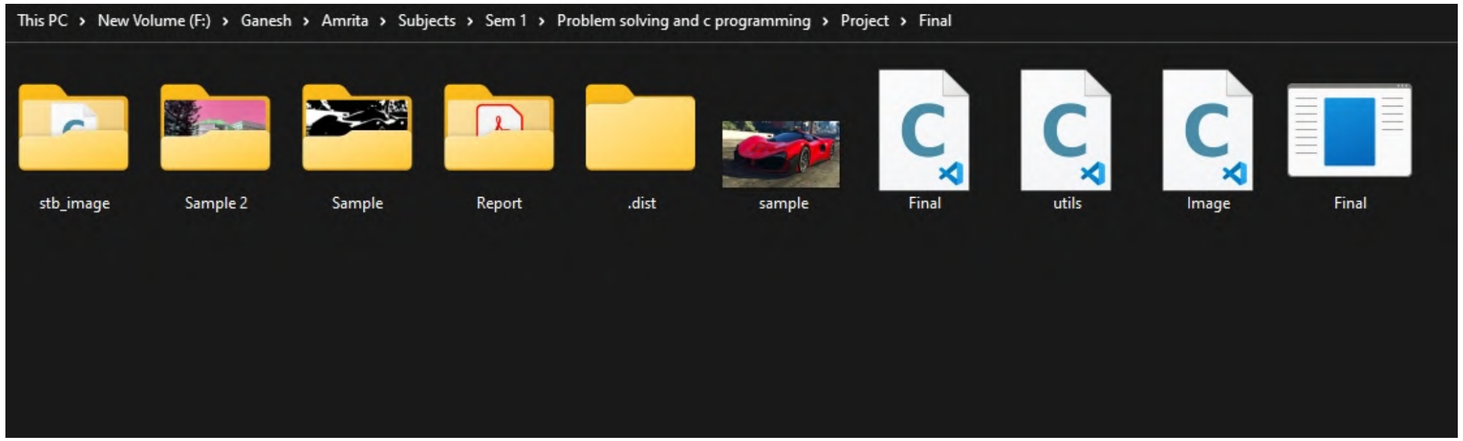
- What is an Image?
- Types of images:
- What are Channels


<--- 4,928 pixels --->
<--- 3,264 pixels --->
16 Megapixels

LIBRARIES AND HEADER FILES

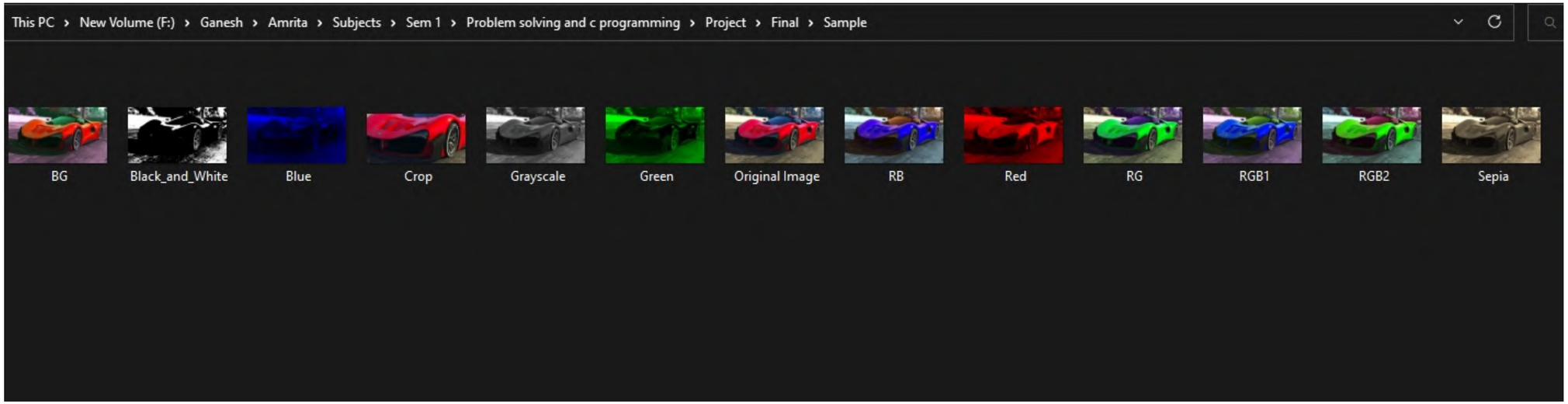
- <stdio.h>
- <stdlib.h>
- stb_image
- stb_image_write

IMPORTING



stbi_load("sample.png",&width,&height,&channels,0)

EXPORTING



stbi_write_jpg("Sepia.jpg", width, height, channels, sepia_img, 100)

stbi_write_png("Sepia.png", width, height, channels, sepia_img, width * channels)

ORIGINAL IMAGE

CROPPING

- Import the Image
- Get the values of the Coordinates
- Initialize size_t variables
- Allocate Memory
- for loop for cropping
- Write the Output
- Free those images

```
for(unsigned char *p = img,*pg=ima_si;p!=img_siz+img;p+=channels) {
    c+=1;
    c=c%(width);
    if(c==0){
        d+=1;
    }

    if(c>=a && c<=m && d>=b && d<=n){
    *pg = *p;
    *(pg+1) = *(p+1);
    *(pg+2) = *(p+2);

    if(channels==4){
        *(pg+3) = *(p+3);
    }

    pg+=channels;

    }
```

RGB COMPONENTS

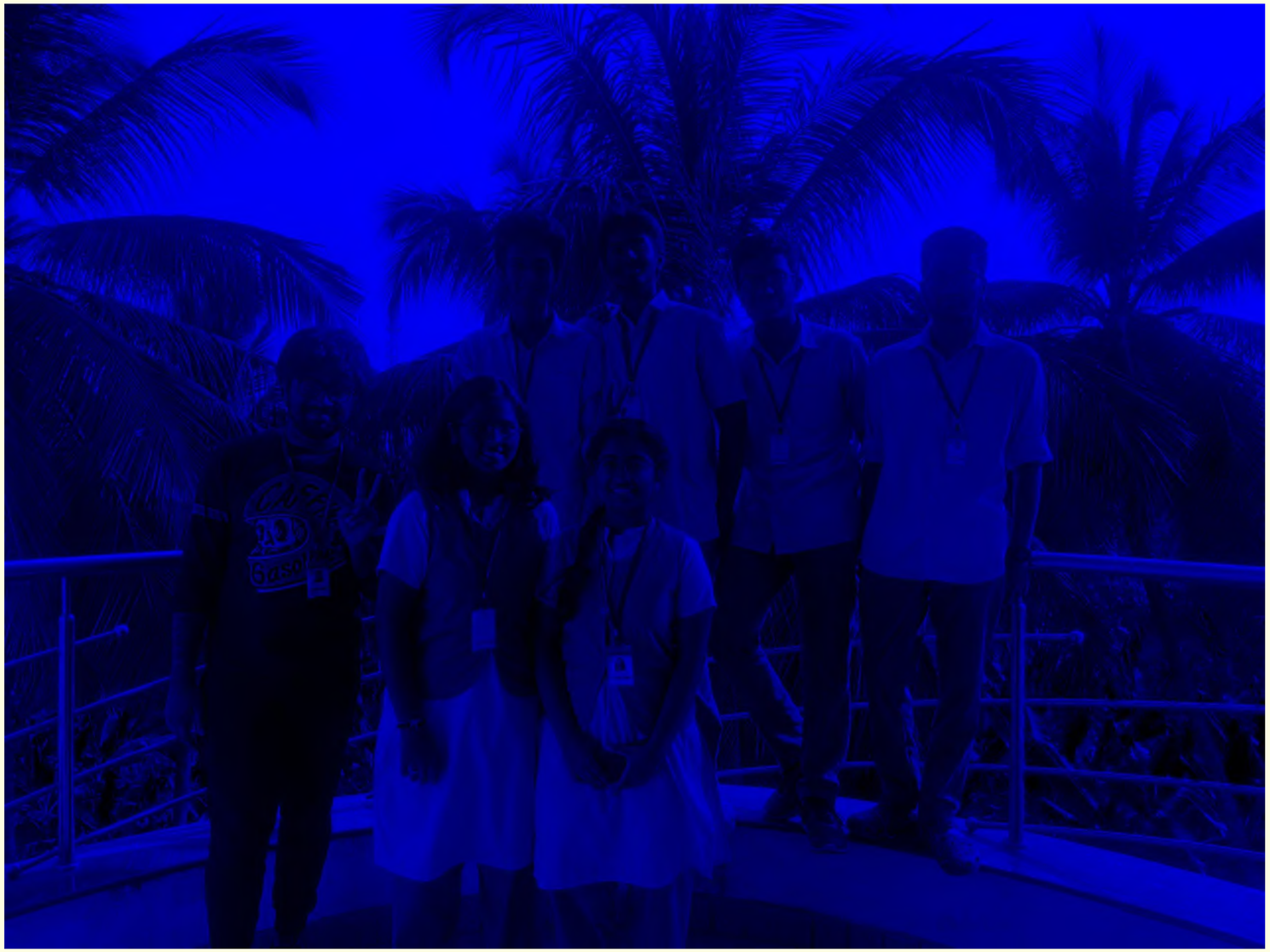
- Import the Image
- Initialize size_t variables
- Allocate Memory
- for loop for extracting components
- Write the Output
- Free those images

```
for(unsigned char *p = img, *pg = gray_img; p != img + img_size; p += channels, pg += channels) {
    *pg = *(p);
    *(pg+1) = (0);
    *(pg+2) = (0);

    if(channels == 4){
        *(pg+3) = *(p+3);
    }
```

```
for(unsigned char *p = img, *pg = gray_img; p != img + img_size; p += channels, pg += channels) {
    *pg = (0);
    *(pg+1) = *(p+1);
    *(pg+2) = (0);

    if(channels == 4){
        *(pg+3) = *(p+3);
    }

}
```

```
for(unsigned char *p = img, *pg = gray_img; p != img + img_size; p += channels, pg += channels) {
    *pg = (0);
    *(pg+1) = (0);
    *(pg+2) = *(p+2);

    if(channels == 4){
        *(pg+3) = *(p+3);
    }

}
```


Gasoline



GRAYSCALE

- Import the Image
- Initialize size_t variables
- Allocate Memory
- for loop for conversion
- Write the Output
- Free those images

```
for(unsigned char *p = img, *pg = gray_img; p != img + img_size; p += channels, pg += gray_channels) {
    *pg = (uint8_t)((*p)*0.299 + *(p + 1)*0.587 + *(p + 2)*0.114);
    if(channels == 4) {
        *(pg + 1) = *(p + 3);
    }
}
```

BLACK AND WHITE

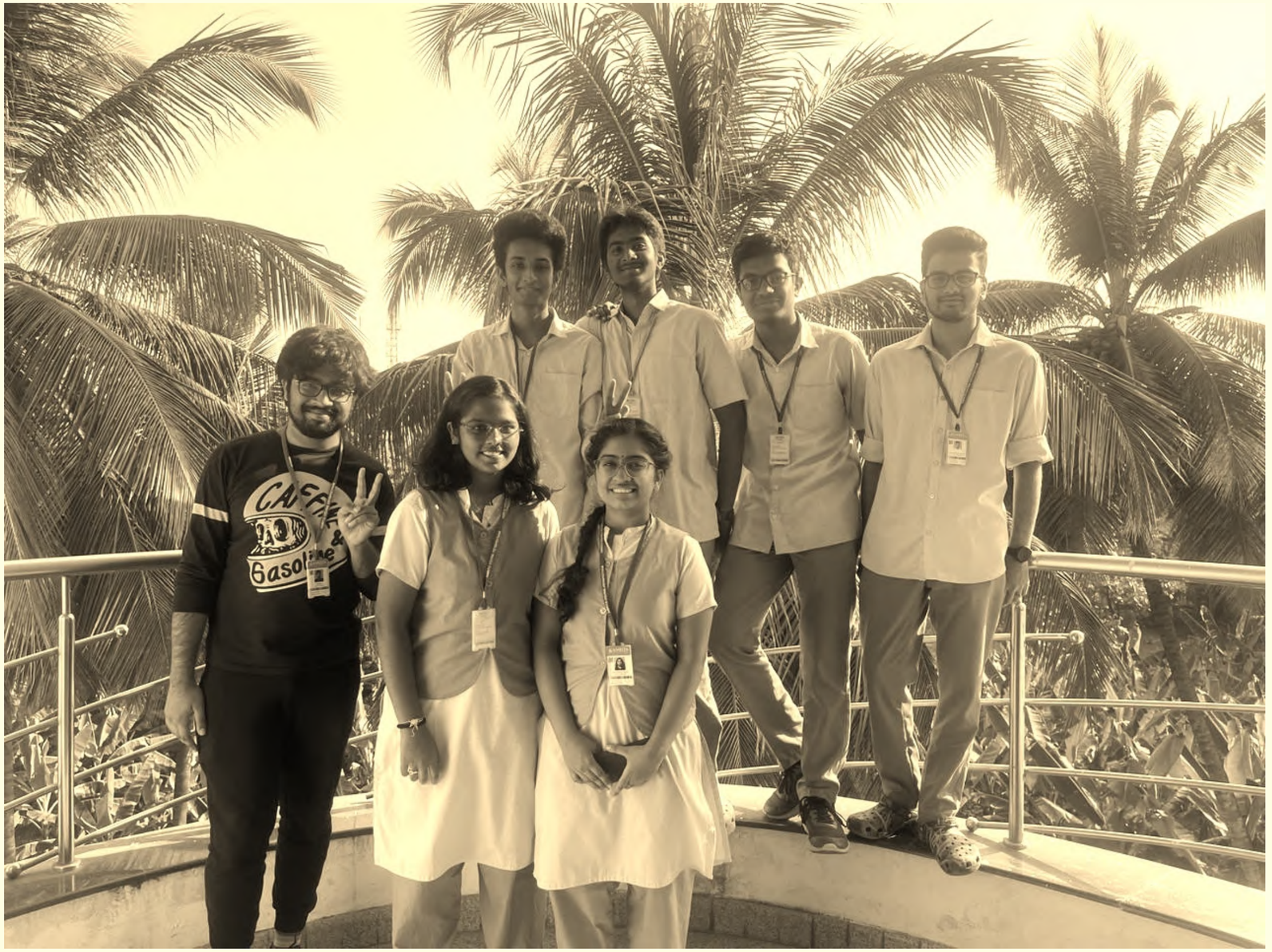
- Import the Image
- Initialize size_t variables
- Allocate Memory
- for loop for conversion
- Write the Output
- Free those images

```
for(unsigned char *a = img,*b = grey_img;a!= img+img_size;a+=channels,b+=bchannels){
        n = (*a + *(a+1) + *(a+2))/3;

        if(n>=128){
            *b = 255;
        }
        else{
            *b = 0;
        }

        if(channels == 4){
            *(b+1) = *(a+3);
        }
}
```

SEPIA

- Import the Image
- Initialize size_t variables
- Allocate Memory
- for loop for conversion
- Write the Output
- Free those images

```
for(unsigned char *p = img, *pg = sepia_img; p != img + img_size; p += channels, pg += channels) {
    *pg       = (uint8_t)fmin(0.393 * *p + 0.769 * *(p + 1) + 0.189 * *(p + 2), 255.0);
    *(pg + 1) = (uint8_t)fmin(0.349 * *p + 0.686 * *(p + 1) + 0.168 * *(p + 2), 255.0);
    *(pg + 2) = (uint8_t)fmin(0.272 * *p + 0.534 * *(p + 1) + 0.131 * *(p + 2), 255.0);
    if(channels == 4) {
        *(pg + 3) = *(p + 3);
    }
}
```



INVERTING COMPONENTS

- Import the Image
- Initialize size_t variables
- Allocate Memory
- for loop for inversion
- Write the Output
- Free those images

```
for(unsigned char *p = img, *pg = gray_img; p != img + img_size; p += channels, pg += channels) {
    *pg = *(p+1);
    *(pg+1) = *(p);
    *(pg+2) = *(p+2);

    if(channels == 4){
        *(pg+3) = *(p+3);
    }

}
```

```
for(unsigned char *p = img, *pg = gray_img; p != img + img_size; p += channels, pg += channels) {
    *pg = *(p+2);
    *(pg+1) = *(p+1);
    *(pg+2) = *(p);

    if(channels == 4){
        *(pg+3) = *(p+3);
    }

}
```

```
for(unsigned char *p = img, *pg = gray_img; p != img + img_size; p += channels, pg += channels) {
    *pg = *(p);
    *(pg+1) = *(p+2);
    *(pg+2) = *(p+1);

    if(channels == 4){
        *(pg+3) = *(p+3);
    }

}
```

```
for(unsigned char *p = img, *pg = gray_img; p != img + img_size; p += channels, pg += channels) {
    *pg = *(p+1);
    *(pg+1) = *(p+2);
    *(pg+2) = *(p);

    if(channels == 4){
        *(pg+3) = *(p+3);
    }

}
```

```
for(unsigned char *p = img, *pg = gray_img; p != img + img_size; p += channels, pg += channels) {
    *pg = *(p+2);
    *(pg+1) = *(p);
    *(pg+2) = *(p+1);

    if(channels == 4){
        *(pg+3) = *(p+3);
    }

}
```

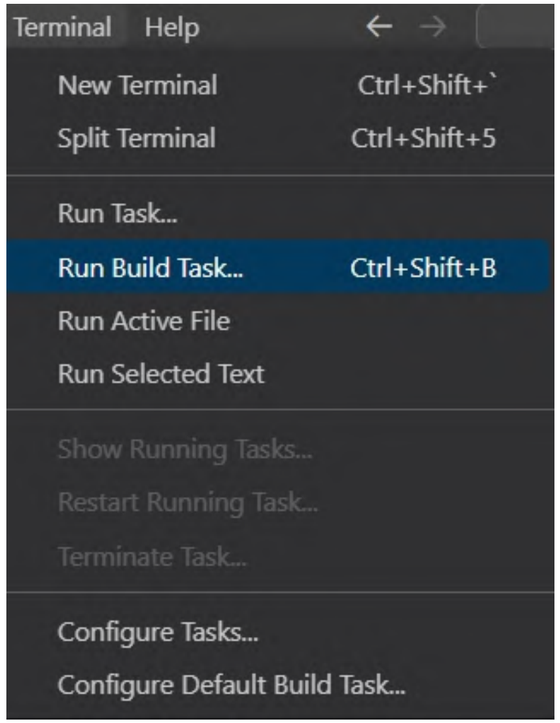
SWITCH

```
switch(number) {
case  C1: //do something
          break;
case C2 : //do something
          break;
default : //do something
}
```

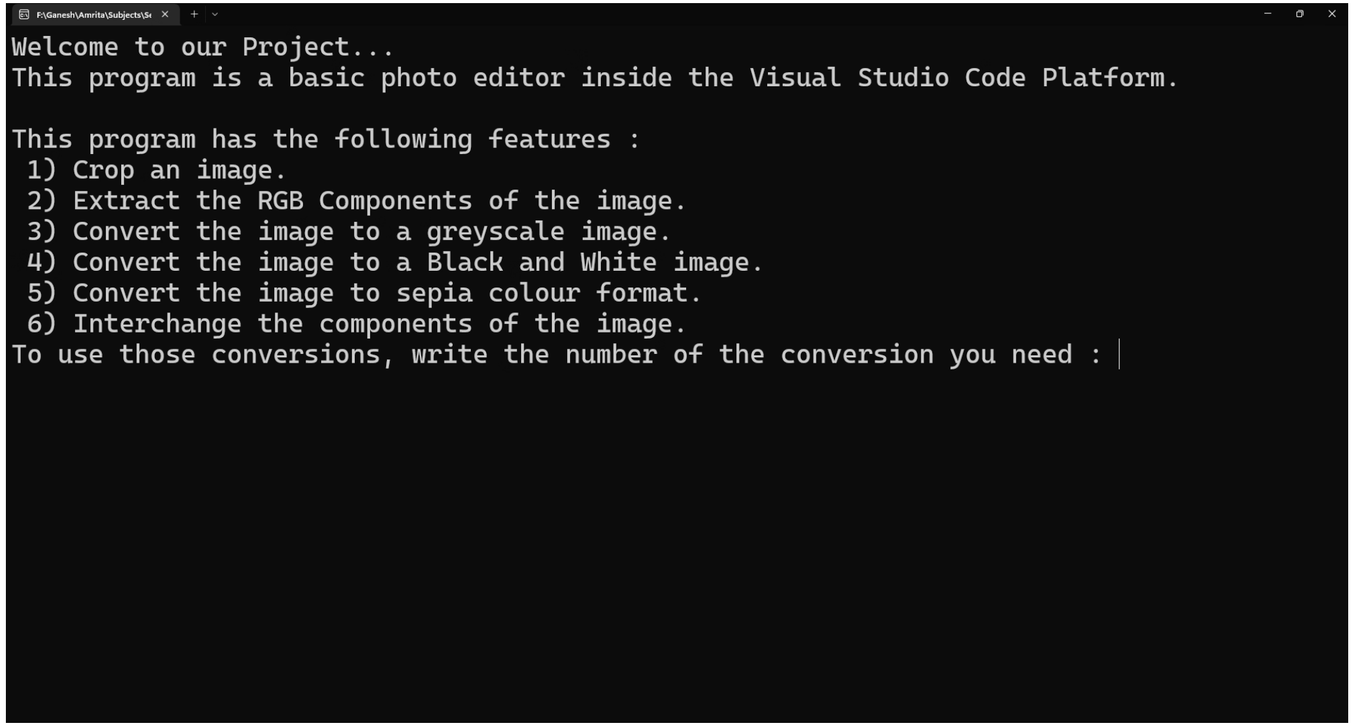
Switch Case for selecting between different conversions.

APPLICATION

Run Build task



Executable file



CONCLUSION

That's it!!

Thank you!