DOCUMENTATIE PROIECT

-PROGRAMARE PROCEDURALA-

```
typedef struct
{
    unsigned char R,G,B;

}pixel;
typedef struct
{
    unsigned int DIMENSIUNE_IMAGINE_PIXELI, LATIME, LUNGIME;

}DETALII_IMAGINE;
typedef struct
{
    int x,y,sters,cifra;
    double val;
}corelatii;
typedef struct
{
    int x,y;
}punct;
```

pixel: Structura ce retine 3 variabile de tipul unsigned char care le voi folosi pentru retinerea pixelilor

DETALII_IMAGINE: Structura ce retine informatii despre o anumita imagine

corelatii: Structura ce se va folosi in alocarea vectorului de detectii.Ea contine informatii cum ar fi x,y-reprezentand un punct in plan, sters-daca aceea detectie este sau nu stearsa, cifra-cifra pe care s-a aplicat detectia, val-valoarea corelatiei in urma aplicarii formulei

punct: Structura ce contine coordonatele unui punct in plan

```
void citire_imagine (char *NUME, unsigned char **RETINE_HEADER,DETALII_IMAGINE *y,
                     pixel **RETINE_CONTINUT_IMAGINE)
```

PARAMETRII:

-NUME: Numele imaginii ce urmeaza sa fie citita

-RETINE_HEADER: Pointer catre un vector de tipul unsigned char in care se va stoca header-ul imaginii

-y: variabila de tip DETALII_IMAGINE in car seva stoca informatii despre imagine

-RETINE_CONTINUT_IMAGINE: Pointer catre un vector de tipul pixel in care se va stoca pixelii imaginii

SCURTA DESCRIERE:

Functia deschide fisierul de unde o sa citesc imaginea.Daca reuseste sa-l deschida citeste header-ul si apoi pozitioneza pointer-ul de fisier convenabil pentru a culege diferite informatii despre imagine cum ar fi latime, lungime etc.Apoi calculeaza paddingul imaginii daca exista si citeste pixelii imaginii ignorand padding-ul.Imaginea se va retine rasturant.

```
void scriere_imagine(char *NUME, unsigned char *RETINE_HEADER,
                     DETALII_IMAGINE y, pixel *RETINE_CONTINUT_IMAGINE)
```

PARAMETRII:

-NUME: Numele fisierului unde urmeaza sa fie scris continutul imaginii

-RETINE_HEADER: Vectorul unde se retine header-ul imaginii

-y: Variabila care retine informatii despre imagine

-RETINE_CONTINUT_IMAGINE: Vectorul unde se retine pixelii imaginii

SCURTA DESRIERE:

Functia deschide fisierul unde se va stoca imaginea.In caz afirmativ copiaza header-ul in fisier si calculeaza padding-ul pentru respectiva imagine.Apoi scrie imaginea, rasturnata in fisier adaugand octetii de padding unde este nevoie.

```
void xorshift32(char *NUME, unsigned int **VECTOR, DETALII_IMAGINE y)
```

PARAMTERII:

-NUME: Numele fisierului unde se retine cheia secreta

-VECTOR: Pointer catre un vector cu elemente de tipul unsigned int unde se va retine numerele generate aleatoriu

-y: Variabila ce retine informatii despre imagine

SCURTA DESCRIERE:

Functia se foloseste de generatorul XORSHIFT32 propus de George Marsaglia astfel:

Deschide fisierul secret_key.txt si copiaza in VECTOR[0] cheia secreta.La pasul i se genereaza un numar aleatoriu pe baza elementului VECTOR[0] si cu ajutorul shiftarilor , apoi acest element se retine in VECTOR[i] si se actulaizeaza si VECTOR[0] cu aceasta valoare

```
void permutare(unsigned int *VECTOR, unsigned int **PERMUTARE,
               DETALII_IMAGINE y)
```

PARAMETRII:

-VECTOR: Vectorul unde se retin numerele generate aleatoriu

-PERMUTARE: Pointer catre un vector de tipul unsigned int care urmeaza sa retina permutarea generata pe baza vectorului VECTOR

-y: Varaiabila ce retine informatii despre imagine

SCURTA DESCRIERE:

Functia intializeaza vectorul PERMUTARE cu permutarea identica(1->1, 2->2 s.a).Apoi se permuta valorile intre ele cu ajutorul numerelor generate aleatoriu din functia xorshift32.

```
void permuta_pixeli(pixel *imagine_normala, pixel **imagine_modificata,
                    unsigned int *PERMUTARE,DETALII_IMAGINE y)
```

PARAMETRII:

-imagine_normala: Pointer catre un vector de tip pixel ce retine pixelii imaginii citite

-imagine_modificata: Pointer catre un vector de tip pixel ce urmeaza sa retina pixelii imaginii dupa permutarea lor

-PERMUTARE: Pointer catre vector de tip unsigned int ce retine permutarea generata cu ajutorul functiei permutare.

-y: Variabila ce retine informatii despre o imagine

SCURTA DESCRIERE:

Se permuta pixelii din imaginea_normala cu ajutorul vectorului PERMUTARE si noua imagine se retine in imagine_modificata

```
void cripteaza_imaginea(char *NUME_IMAGINE, char *NUME_IMAGINE_CRIPTATA,
                        char *NUME_CHEIA_SECRETA)
```

PARAMETRII:

-NUME_IMAGINE: Numele imaginei ce urmeaza sa fie criptata

-NUME_IMAGINE_CRIPTATA: Numele imaginei unde urmeaza sa se retina imaginea criptata

-NUME_CHEIA_SECRETA: Numele fisierului ce retine cheia secreta

SCURTA DESCRIERE:

Se apeleaza functiile de mai sus pentru a creea imaginea cu pixelii permutati.Apoi se incearca deschiderea fisierului secret_key.txt pentru a culege a 2-a valoare din fisier ce se va retine in SV. In caz afirmativ, se aplica formula prezentata in proiect de la criptare, iar la final se scrie imaginea criptata in fisier si se elibereaza toata memoria folosita.

```
void permutarea_inversa(unsigned int **PERMUTARE_INVERSA,
                        unsigned int *PERMUTARE, DETALII_IMAGINE y)
```

PARAMETRII:

-PERMUTAREA_INVERSA: Pointer catre un vector cu elemente unsigned int ce urmeaza sa retina permutarea inversa

-PERMUTAREA: Pointer catre un vector cu elemente de tip unsigned int ce retine permutarea genererata

-y: Variabila ce retine informatii despre imagine

SCURTA DESRIERE:

Se genereaza permutarea inversei a vectorului PERMUTAREA

```
void decripteaza_imaginea(char *NUME_IMAGINE_DECRIPTATA,
                          char *NUME_IMAGINE_CRIPTATA,
                          char *NUME_CHEIA_SECRETA)
```

PARAMETRII:

-NUME_IMAGINE_DECRIPTATA: Numele imaginii unde se va retine imaginea decriptata

-NUME_IMAGINE_CRIPTATA: Numele imaginii unde se retine imaginea criptata

-NUME_CHEIA_SECRETA: Numele fisierului unde se retine cheia secreta

SCURTA DESCRIERE:

Se citeste imaginea criptata, apoi se folosesc functiile de mai sus pentru a genera permutarea si permutarea inversa.Se deschide fisierul secret_key.txt si se culege a 2-a valoare ce se retine in SV.Apoi se aplica formula de decriptarea prezentata in proiect.

Dupa cu ajutorul functiei permuta_imaginea_decriptata se ajunge la imaginea initiala care se scrie in fisier iar mai apoi se elibereaza toata memoria utilizata

```
void permuta_imaginea_decriptata(pixel **imagine_decriptata,
                                 pixel *imagine_decriptata_partial,
                                 unsigned int *permutarea_inversa,
                                 DETALII_IMAGINE y)
```

PARAMETRII:

-imagine_decriptata: Pointer catre un vector cu elemente de tip pixel ce va retine imaginea decriptata

-imagine_decriptata_partial: Pointer catre un vector cu elemente de tip pixel ce retine imaginea_decriptata_partial(imaginea dupa ce s-a aplicat formula de xorare/decriptare)

-y: Variabila ce retine informatii despre imagine

-permutarea_inversa: Pointer catre un vector ce retine permutarea inversa

SCURTA DESCRIERE:

Pe baza permutarii inverse se permuta pixelii inapoi in imagine astfel acestia ajungand pe locul lor si reformand imaginea initiala

```
void chi_patrat(char *NUME_IMAGINE)
```

PARAMETRII:

-NUME_IMAGINE: Numele imaginii pe care urmeaza sa se aplice testul chi_patrat

SCURTA DESCRIERE:

Se citeste imaginea din fisier pe care urmeaza sa se aplice testul si se implementeaza formula prezentata in proiect.Valorile testului sunt afisate pe ecran

```
void citire_matrice(char *NUME,
                    pixel ***RETINE_CONTINUT_SABLON,
                    unsigned char **RETINE_HEADER,DETALII_IMAGINE *y)
```

PARAMETRII:

-NUME: Numele imaginii care urmeaza sa fie citie

-RETINE_CONTINUT_SABLON: Pointer catre o matrice alocata dinamic cu elemente de tipul pixel ce urmeaza sa retina pixelii imaginii

-RETINE_HEADER: Pointer catre un vector alocat dinamic cu elemente de tipul unsinged char ce retine header-ul imaginii

-y: Variabila ce retine informatii despre imagine

SCURTA DESCRIERE:

Se deschide fisierul de unde se citeste imaginea.In caz afirmativ se citeste header-ul, se calculeaza padding-ul pentru imagine si se pozitioneaza cursorul convenabil pentru a stoca informatii despre imagine in y.Apoi se citeste matricea, aceasta retinandu-se rasturnat in memorie.Padding-ul se ignora.

```
void afisare_matrice(char *NUME,
                     pixel **RETINE_CONTINUT_SABLON,
                     unsigned char *RETINE_HEADER,
                     DETALII_IMAGINE y)
```

PARAMETRII:

-NUME: Numele fisierului unde se va scrie noua imagine

-RETINE_CONTINUT_SABLON: Pointer catre o matrice alocata dinamic ce retine elemente de tip pixel.Aceasta retine pixelii imaginii

-RETINE_HEADER: Poiner catre un vector alocat dinamic ce retine elemenete de tipul unsinged char care retine header-ul imaginii

-y: Variabila ce retine informatii despre imagine

SCURTA DESCIRERE:

Se deschide fisierul.In caz afirmativ se scrie header-ul in fisier si se calculeaza paddingul.Apoi se scrie in fisier matricea rasturnata si se adauga octetii de padding unde este nevoie

```
void grayscale_image(char* nume_fisier_sursa,
                     pixel ***IMAGINE,
                     unsigned char **HEADER_IMAGINE,
                     DETALII_IMAGINE *img)
```

PARAMETRII:

-nume_fisier_sursa: Numele fisierului de unde citesc imaginea

-IMAGINE: Pointer catre o matrice alocata dinamic ce retine elemente de tipul pixel in care este stoacata imaginea

-HEADER_IMAGINE: Pointer catre un vector alocat dinamic cu elemente de tipul unsigned char ce retine header-ul imaginii

-img: Variabila ce retine informatii despre imagine

SCURTA DESCRIERE:

Se citeste din fisier imaginea cu ajutorului functiei citire_matrice.Apoi se transforma imaginea in grayscale

```
void template_matching(char *NUME_SABLON, float PS ,
                       pixel ***IMAGINE, pixel ***SABLON,
                       unsigned char **HEADER_SABLON ,
                       DETALII_IMAGINE DIM_IMG,
                       DETALII_IMAGINE *DIM_S, int *k, corelatii **d,
                       int indice)
```

PARAMETRII:

-NUME_SABLON: Numele sablonului care urmeaza sa fie citit

-PS: Pragul care indica daca o detectie e buna sau nu

-IMAGINE: Pointer catre o matrice alocata dinamic cu elemente de tipul pixel care retine pixelii imaginii

-SABLON: Pointer catre o matrice alocata dinamic cu elemente de tipul pixel care retine pixelii sablonului

-HEADER_SABLON: Pointer catre un vector alocat dinamic cu elemente de tipul unsigned char ce va retine header-ul sablonului

-DIM_IMG: Variabila ce retine informatii despre imagine

-DIM_S: Variabila ce urmeaza sa retina informatii despre sablon

-k: Variabila ce o sa retina numarul de detectii>PS

-d: Poiner ce retine un vector alocat dinamic cu elemente de tipul corelatii ce urmeaza sa retina detectiile si anumite informatii despre acestea

-indice: Variabila ce retine ce cifra a fost testata.Spre exemplu daca a fost testat sablonul(cifra0.bmp variabila indice va avea valoarea 0)

SCURTA DESCRIERE:

Functia citeste sablonul curent care urmeaza sa fie testat pe imaginea grayscale.Apoi merge pixel cu pixel pana la finalul imaginie si in fiecare punct apeleaza functia corelatie(care returneaza valoarea corelatiei).Daca este mai pare ca PS, aceasta detectie se retine in vectorul d.In final se elibereaza toata memoria utilizata

```
double corelatie(int x, int y, pixel **IMAGINE,
                 pixel **SABLON, DETALII_IMAGINE di, DETALII_IMAGINE ds)
```

PARAMETRII:

-x, y :Coordonatele punctului stanga sus al detectiei

-IMAGINE: Poineter catre o matrice alocata dinamic cu elemente de tipul pixel ce retine pixelii imaginii

-di: Variabila ce retine informatii despre imagine

-ds: Variabila ce retine informatii despre sablon

SCURTA DESCRIERE:

Functia implementeaza formula de corelatie prezentata in proiect.

```
void deseneaza_sablon(int x, int y,
                      pixel ***IMAGINE,
                      DETALII_IMAGINE d, pixel C)
```

PARAMTERII:

-x, y: Coordonatele punctului stanga sus al detectiei

-IMAGINE: Pointer catre o matrice alocata dinamic ce retine elemente de tipul pixel care urmeaza sa stocheze imagine dupa desenarea sabloanelor

-d: Variabila ce retine informatii despre imagine

-C: Variabila de tip pixel ce retine culoare cu care se va desenza conturul sablonului

SCURTA DESCIERE:

Functia deseneaza pe imagine conturul sabloanelor

```
int verifica_intersectia(punct l1, punct r1, punct l2, punct r2)
```

PARAMETRII:

-l1: Coltul stanga sus al primului dreptunghi

-r1:Coltul dreapta jos al primului dreptunghi

-l2: Coltul stanga sus al dreptunghiului 2

-r2:Coltul dreapta jos al dreptunghiului 2

SCURTA DESCIERE:

Functia implementeaza formula cu arie pentru cele 2 dreptunghiuri prezentata in proiect.Daca interesectia lor este mai mare de 0.2 atunci se returneaza 1(insemnand ca dreptunghiul trebuie sters) altfel se returneaza 0 ceea ce inseamna ca dreptunghiul trebuie pastrat momentan

```
void elim_suprapuneri(corelatii **d, int k, DETALII_IMAGINE s)
```

PARAMETRII:

-d: Vectorul unde se retin detectiile >PS

-k: Variabila ce retine numarul de detectii >PS

-s: Variabila ce retine informatii despre sablon

SCURTA DESCRIERE:

Functia porneste cu 2 for-uri pe vectorul de detectii.Unul cu i=0,k-1 si unul cu j=i,k.Apoi calculeaza pentru fiecare detectie i, j coltul din stanga sus, coltul din dreapta jos si apeleaza functia de verificare a intersectiei

```
 FILE *date=NULL;
 date=fopen("date.txt","rt");
 if (date==NULL)
 {
     printf("NU S-A PUTUT CITI FISIERUL DE DATE");
     exit(-1);
 }
RIMA PARTE
 char calea_imaginii_in[30], calea_imaginii_out[30],
 nume_cheia_secreta[30], nume_imagine_criptata[30];
 int i;

 fscanf(date,"%30s",calea_imaginii_in);
 fscanf(date,"%30s",nume_imagine_criptata);
 fscanf(date,"%30s",calea_imaginii_out);
 fscanf(date,"%30s",nume_cheia_secreta);

 cripteaza_imaginea(calea_imaginii_in, nume_imagine_criptata, nume_cheia_secreta);
 decripteaza_imaginea(calea_imaginii_out, nume_imagine_criptata, nume_cheia_secreta);
 printf("Chi-squared pentru peppers.bmp:\n");
 chi_patrat(calea_imaginii_in);
 printf("\nChi-squared pentru imaginea criptata(peppers_criptata.bmp):\n");
 chi_patrat(nume_imagine_criptata);
```

Explicatie:

In interiorul functiei main, pentru ambele parti deschidem un fisier de unde citim numele tuturor fisierelor care sunt implicate in program.Apoi pentru prima parte, facem declararile si initializarile necesare pentru criptare/decriptare.Dupa ce am terminat cu partea de initializare si alocare incepem criptarea si decriptarea.Apoi apelam functia chi_patrat pentru imaginea peppers_bmp si pentru imaginea peppers_criptata.bmp

```
int nr_detectii=0;
unsigned char *HEADER_SABLON, *HEADER_IMAGINE, *HEADER_IMAGINE_MODIFICATA;
corelatii *DETECTII;
pixel **SABLON, **IMAGINE, **IMAGINE_MODIFICATA;
pixel CULORI[10]={{255,0,0} , {255,255,0} , {0,255,0} , {0,255,255}, {255,0,255} ,
{0,0,255}, {192,192,192}, {255,140,0}, {128,0,128}, {128,0,0}};

DETALII_IMAGINE DIM_SABLON, DIM_IMG, DIM_IMG_MODIF;
char NUMELE_IMAGINII_IN[30], NUMELE_IMAGINII_OUT[30];
char NUMELE_SABLONULUI[10][12];
for (i=0;i<=9;i++)
   fscanf(date,"%30s",NUMELE_SABLONULUI[i]);
fscanf(date,"%30s",NUMELE_IMAGINII_IN);
fscanf(date,"%30s",NUMELE_IMAGINII_OUT);

grayscale_image(NUMELE_IMAGINII_IN, &IMAGINE, &HEADER_IMAGINE, &DIM_IMG);
citire_matrice(NUMELE_IMAGINII_IN, &IMAGINE_MODIFICATA, &HEADER_IMAGINE_MODIFICATA, &DIM_IMG_MODIF);
for (i=0;i<10;i++)
{
    printf("\nSe aplica template_matching pentru %s",NUMELE_SABLONULUI[i]);
    template_matching(NUMELE_SABLONULUI[i], 0.50, &IMAGINE, &SABLON,
                      &HEADER_SABLON, DIM_IMG, &DIM_SABLON, &nr_detectii, &DETECTII, i);
}
qsort(DETECTII, nr_detectii, sizeof(corelatii),cmp);
```

Explicatie:

Pentru a 2-a parte facem declararile si initializarile necesare.Consideram un vector care retine culori astfel, pentru valoare i apartinand [0,9] se retine o anumita culoare data in proiect.Apoi se citesc din fisierul date.txt numele fisierelor necesare si se transforma imaginea test.bmp in grayscale.Apoi se considera o alta imagine test.bmp(care nu este grayscale) pentru a desena pe aceasta conturul sabloanelor.Apoi se aplica operatia de template_matching pentru toate cele 10 sabloane.Dupa asta se foloseste functia qsort pentru a sorta in ordine descrescatoare detectiile

```
    elim_suprapuneri(&DETECTII, nr_detectii, DIM_SABLON);
    for (i=0;i<nr_detectii;i++)
        if(DETECTII[i].sters==0)
            deseneaza_sablon(DETECTII[i].x, DETECTII[i].y, &IMAGINE_MODIFICATA,
                             DIM_SABLON, CULORI[DETECTII[i].cifra]);
    afisare_matrice(NUMELE_IMAGINII_OUT,
                    IMAGINE_MODIFICATA, HEADER_IMAGINE_MODIFICATA, DIM_IMG_MODIF);

    free(HEADER_IMAGINE), free(HEADER_IMAGINE_MODIFICATA), free(DETECTII);
    for (i=0;i<DIM_IMG.LUNGIME;i++)
        free(IMAGINE[i]);
    for (i=0;i<DIM_IMG_MODIF.LUNGIME;i++)
        free(IMAGINE_MODIFICATA[i]);
    fclose(date);
    return 0;
}
```

Dupa ce avem sortate detectiile, apelam functia de eliminare a suprapunerilor.Apoi desenam sabloanele pentru detectiile pentru care campul sters=0.In cele din urma afisam matricea, dezalocam toata memoria folosita si inchidem fisierul date.txt