Explicatie cod proiect Programare Procedurala:

Pentru a intelege mai bine conceptul din spatele codului meu, am sa prezint mai intai structurile folosite:

-structura "pixel" (folosita la ambele parti ale proiectului) - retine valorile celor 3 canale de culoare ale unui pixel RGB, cu ajutorul a 3 variabile de tip unsigned char;
-structura "BMP_image"(folosita doar la partea de criptare de imagini) - retine dimensiunile unei imagini (inaltime si latimea) cu ajutorul a 2 variabile de tip unsigned int, precum si un vector de elemente de tip "pixel", care retine la randul lui toti pixelii imaginii respective, in forma unui vector liniarizat;
-structura "detectie"(folosita doar la partea de recunoastere de pattern-uri) - retine doi/doua indici/coodonate (IndiceI si IndiceJ) ale unei detectiie, ce reprezinta coordonatele pixelului din coltul din stanga sus ale detectiei, o variabila de tipa double "corelatie", care retine corelatia fiecare detectii calculata conform formulei din enuntul proiectului,iar variabila de tip "culoare" de tip "pixel",de care ma voi ajuta sa pot schimba culoare conturului pentru o detectie anume;
-structura "BMP_matrice"(folosita doar la partea de recunoastere de pattern-uri) - asemanatoare structurii "BMP_imagine", retine dimensiunile unei imagini (inaltime si latime) cat si un vector bidimensional de tip "pixel" care retine sub forma de matrice pixelii imaginii respective;

Prima parte a proiectului (Criptarea de imagini):

-functia "BMP_image* citire_si_liniarizare(char *nume_img)" :
	-primeste calea unei imagini;
	-deschide fisierul (binar,deoarece este imagine BMP);
	-afla dimensiunile imaginii primite prin informatiile din header;
	-creeaza vectorul liniarizat "BMP_imagine *img", continand toti pixelii imaginii;
	-inchide fisierul si returneaza vectorul liniarizat (variabila de tip BMP_imagine);

-functia "unsigned int* xorshift32(int seed, int size)" :
	-primeste un seed(int) si un size(int), reprezentand numarul de elemente are vectorului de numere generate cu ajutorul algoritmului XORSHIFT32 care urmeaza sa fie returnat;
	-implementeaza algoritmul XORSHIFT32, un generator de numere pseudo-aleatoare;
	-"populeaza" tabloul unidimensional "output" cu numerele generate;
	-returneaza tabloul "output";

-functia "BMP_image* permutare(BMP_image *img, unsigned int *aleatoare)" :
	-primeste o imagine(imaginea creeata cu ajutorul functiei "citire_si_liniarizare") si tabloul de numere aleatoare;
	-creeaza permutarea sigma initiala (intr-un tablou unidimensional de tip

unsigned int);
	-implementeaza algoritmul Durstenfeld;
	-creeaza si salveaza imaginea permutata in tabloul unidimensional "intermediare" de tip BMP_image*;
	-returneaza tabloul;

-functia "void criptare(BMP_image *img, unsigned int *aleatoare, char *secret_key)" :
	-primeste o imagine (imaginea creeata cu ajutorul functiei "permutare"), tabloul de numere aleatoare si calea cheii secrete, care urmeaza sa fie citita dintr-un fisier text;
	-modifica imaginea primita pe baza formulei de criptare prezentata in enuntul proiectului;

-functia "void creare_img(BMP_image *img, char *nume_img_sursa, char *nume_img_dest)" :
	-primeste o imagine, calea imaginii sursa,calea imaginii destiatie;
	-deschide cele 2 fisiere;
	-copiaza headerul din imaginea sursa in imaginea destinatie, iar in continuare, copiaza fiecare pixel al imaginii primite ca parametru, in fisierul "nume_img_destinatie";
	-face padding-ul dupa fiecare linie copiata;
	-inchide fisierele;

-functia "void testul_chi_patrat(BMP_image *img)" :
	-primeste o imagine;
	-calculeaza testul chi patrat si afiseaza pe ecran valorile pentru fiecare canal (R,G,B);

-functia "BMP_image* decriptare(BMP_image *img, unsigned int *aleatoare, char *secret_key)" :
	-primeste o imagine criptata, tabloul de numere aleatoare si calea cheii secrete, care urmeaza sa fie citita dintr-un fisier text;
	-modifica imaginea criptata primita pe baza formulei de decriptare prezentata in enuntul proiectului;
	-returneaza imaginea decriptata prin intermediul unei variabile de tip BMP_imagine, "img_decriptata";

A doua parte a proiectului (Recunoasterea de pattern-uri) :

-functia "void grayscale_image(char* nume_fisier_sursa, char* nume_fisier_destinatie)" :
	-primeste calea imaginii sursa si calea imaginii care va urma sa fie grayscale;
	-modifica valorile celor 3 canale (R,G,B) dupa o formula anume;
	-creeaza noua imagine grayscale;

-functia "BMP_matrice* creare_matrice(char *nume_img_sursa)" :
	-primeste calea unei imagini sursa;

-deschide fisierul (binar,deoarece este imagine BMP);
-afla dimensiunile imaginii primite prin informatiile din header;
-creeaza tabloul bidimensional "BMP_matrice *img", continand toti pixelii imaginii;
-inchide fisierul si returneaza tabloul bidimensional (variabila de tip BMP_matrice);

-functia "double deviatie_s(double S_mediu, BMP_matrice *sablon)" :
-primeste "S_mediu" (media valorilor intensităților grayscale a pixelilor în fereastra), care va fi folosit pentru a calcula deviatia sablonului "sablon";
-reprezintă deviația standard a valorilor intensităților grayscale a pixelilor în șablonul "sablon";
-calculeaza pe baza formulei din enuntul proiectului, deviatia standard s;
-returneaza "suma_s", reprezentand calculul deviatiei standard;

-functia "double deviatie_fi(int i, int j, double fi_mediu, BMP_matrice *sablon, BMP_matrice *img)" :
-primeste "fi_mediu" (media valorilor intensităților grayscale a pixelilor din fereastra), care va fi folosit pentru a calcula deviatia ferestrei fi;
-reprezintă media valorilor intensităților grayscale a pixelilor din fereastra fi;
-calculeaza pe baza formulei din enuntul proiectului, deviatia standard fi;
-returneaza "suma_fi", reprezentand calculul deviatiei standard;

-functia "void contur_fereastra(BMP_matrice **img, int W, int H, detectie *detectii, int i)" :
-primeste tabloul bidimensional "img", inaltimea si latimea sablonului (W si H),tabloul "detectii" de tip "detectie" si indicele i;
-cu ajutorul campului "culoare" al structurii "deviatie", rescrie valorile pixelilor conturului pe care trebuie sa il realizeze;

-functia "double calculeaza_S_mediu(BMP_matrice *sablon, double *S_mediu)" :
- media valorilor intensităților grayscale a pixelilor în fereastra "sablon" (fereastra S din formula din cerinta);

-functia "double calculeaza_fi_mediu(BMP_matrice *img, BMP_matrice *sablon, double *fi_mediu, int i, int j)" :
-media valorilor intensităților grayscale a pixelilor în fereastra "sablon" (fereastra fi din formula din cerinta);

-functia "double suprapunere(int DIindiceI, int DIindiceJ, int DJindiceI, int DJindiceJ, BMP_matrice *sablon)" :
-verifica daca 2 detectii se suprapun;
-calculeaza pe baza formulei din enuntul proiectului, aria care se intersecteaza dintre 2 detectii;
-returneaza rezultatul0

-functia "void eliminare_non_maxim(int *size_detectii, BMP_matrice *sablon, detectie **detectii)" :
-functia elimina toate detectiile care au gradul de suprapunere mai mare de

0.2,calculat cu ajutorul functiei "suprapunere";
	-actualizeaza tabloul "detectii" (elementele acestuia cat si numarul de elemente);

-functia "int comparator(const void *c1, const void *c2)" :
	-functie care compara 2 elemente de tip detectie;
	-va fi folosita in functia "sortare", fiind unul dintre parametrii functiei qsort;

-functia "void sorteaza(detectie *detectii, int size_detectii)" :
	-sorteaza tabloul "detectii" de tip detectie in functie de campul corelatiei al fiecarui element;
	-elementele vor fi sortate descrescator;

-functia "BMP_matrice* matching_si_corelatie(char *nume_img_sursa_GS, char **vector_sabloane, float prag, detectie **detectii, int *size_detectii, char *nume_img_sursa)" :
	-parcurge cele 10 sabloane primite prin adresa in tabloul "vector_sabloane";
	-calculeaza deviatiile fi/s, cat si S_mediu si fi_mediu pentru a calcula corelatia fiecarei detectii pe baza formulei din enuntul problemei;
	-verifica daca corelatia este mai mare decat un prag citit de la tastatura. Daca da,atunci aloca/realoca memorie pentru vectorul de detectii,si populeaza pe pozitia curenta vectorul, modificand fiecare camp,inclusiv valorile campurilor R,G,B;
	-sorteaza tabloul de detectii;
	-apeleaza functia "eliminare_non_maxim" pentru a elimina detectiile (ferestrele) care se suprapun;
	-apeleaza functia "contur_fereastra" pentru fiecare element al tabloului "detectii";

-functia "void afisare_matrice(BMP_matrice *img, char *nume_img_sursa, char *nume_img_dest)" :
	-primeste o imagine, calea imaginii sursa,calea imaginii destiatie;
	-deschide cele 2 fisiere;
	-copiaza headerul din imaginea sursa in imaginea destinatie, iar in continuare, copiaza fiecare pixel al imaginii primite ca parametru, in fisierul "nume_img_destinatie";
	-face padding-ul dupa fiecare linie copiata;
	-inchide fisierele;

FUNCTIA MAIN :

Functia main este structurata sub forma unui meniu (switch) cu 3 optiuni.
Primele 2 optiuni sunt suplimentare,reprezentand executarea primei,respectiv celei de-a doua parti. A treia optiune reprezinta programul COMPLET cum este precizat in enuntul proiectului.
OPTIUNEA NR.3 (PROGRAMUL COMPLET,CERUT) REALIZEAZA URMATOARELE :

-Declara si aloca tablourile necesare;
-Apeleaza functiile pentru : a creea vectorul liniarizat care stocheaza imaginea

pentru prima parte a proiectului,cripteaza o imagine color BMP si salveaza imagina criptata in memoria externa,afiseaza pe ecran valorile testului chi-patrat pentru imaginea initiala si imaginea criptata,pe fiecare canal de culoare,decripteaza o imagine color BMP criptata si salveaza imaginea decriptata in memoria externa,citeste si foloseste cheia secreta,ruleaza operatia de template matching pentru o imagine BMP si o coletie de sabloane color BMP,ruleaza functia de eliminare a non-maximelor pe tabloul bidimensional si deseneaza in imagine detectiile ramase folosind o culoare specifica detectiei respective;
-Dezaloca TOATA memoria alocata.