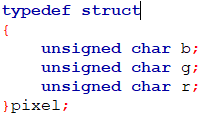
**Documentatie proiect programare procedurala**

Proiectul reuneste 2 mari teme ale informaticii: criptarea continutului unui document, respectiv procesarea digitala si recunoasterea unor pattern-uri dintr-o imagine.

In prima parte, vor fi detaliati pasii necesari criptarii unei imagini în format bmp.

**Criptare si decriptare**



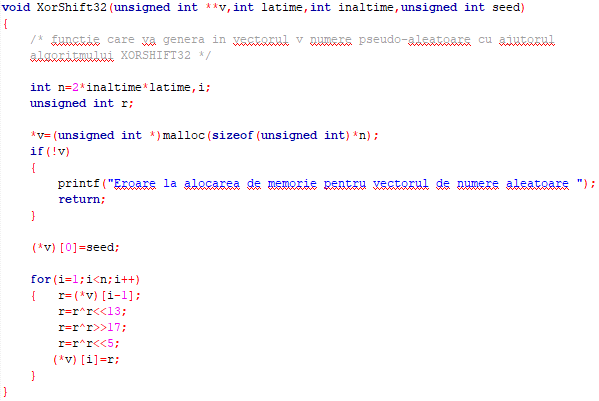
Am definit o structura “pixel” pentru a memora convenabil continutul unei imagini. Vor fi stocati cei 3 octeti ai unui pixel, pentru canalele rosu, verde şi albastru.

**XorShift32**

Funcţia primeste prin intermediul parametrilor:

* *\*v*- un tablou unidimensional alocat dinamic in care vor fi stocate numerele pseudo-aleatoare
* *latime* – latimea imaginii criptate
* *inaltime* – inaltimea imaginii criptate
* *seed* – valoarea initiala de la care se pleaca

si va genera o secventa de numere pseudo-aleatoare prin intermediul algoritmului XorShift32 propus de George Marsaglia.

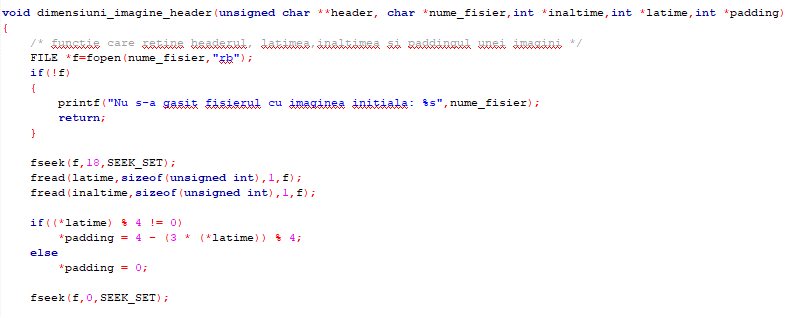


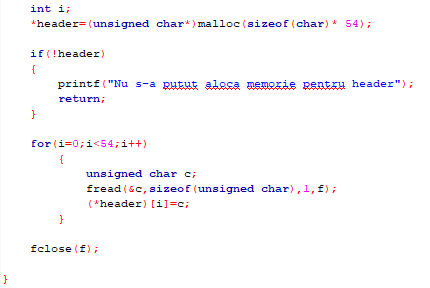
**dimensiuni\_imagine\_header**

Functia va primi ca parametri:

* *header* – un tablou unidimensional alocat dinamic in care va fi stocat headerul imaginii transmis prin parametrul nume\_fisier
* *nume\_fisier* – calea unei imagini
* *inaltime* – inaltimea imaginii transmisa prin parametrul “*nume\_fisier*”
* *latime* – latimea imaginii transmisa prin parametrul “*nume\_fisier*”
* *padding* – paddingul imaginii transmisa prin parametrul “*nume\_fisier*”

si retine principalele caracteristici ale unei imagini





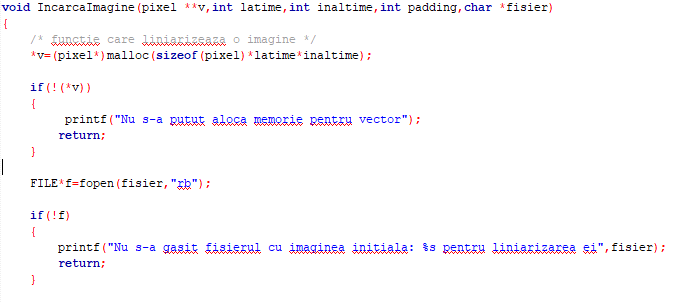
Functia deschide imaginea furnizata in modul “rb” si se pozitioneaza pe pozitia celui de al 18-lea octet pentru a citi latimea imaginii, iar in continuare inaltimea acestuia si calculeaza paddingul. Dupa care se pozitioneaza la inceput, si citeste primii 54 octeti, reprezentand header-ul imaginii.

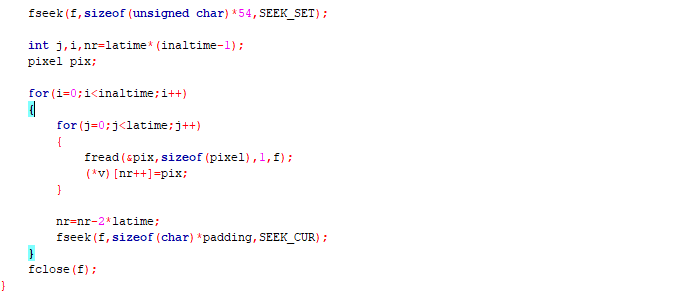
**IncarcaImagine**

Functia va avea ca parametri:

* *\*v* – tablou unidimensional structura de tip pixel
* *latime* – latimea unei imagini
* *inaltime* – inaltimea unei imagini
* *\*fisier* – sir de caractere ce reprezinta calea imaginii

si liniarizeaza imaginea transmisa prin parametrul *fisier*, incepand din coltul din stanga sus pana la cel din dreapta jos.





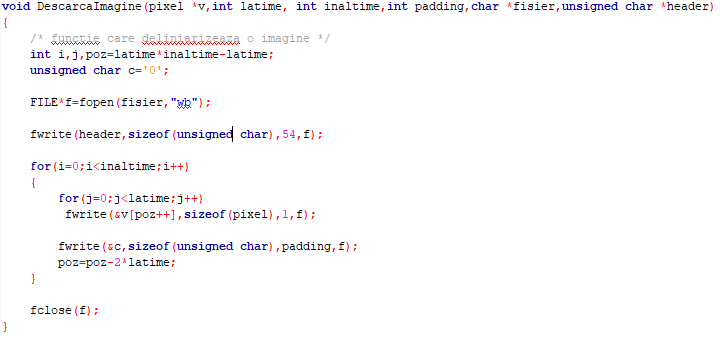
Functia sare peste primii 54 de octeti, ce reprezinta header-ul, dupa care citeste cate un pixel ( 3 octeti fara semn ), punandu-l pe pozitia nr, actualizata dupa fiecare atribuire, in tabloul unidimensional *\*v* .

**DescarcaImagine**

Functia va avea ca parametri:

* *\*v* – tablou unidimensional structura de tip pixel ce memoreaza o imagine in forma liniarizata
* *latime* – latimea unei imagini
* *inaltime* – inaltimea unei imagini
* *\*fisier* – sir de caractere ce reprezinta denumirea unei imagini
* *padding* – padding-ul imaginii
* *header* – headerul imaginii

si salveaza in memoria externa sub numele trimis ca parametru *fisier* imaginea memorata prin tabloul unidimensional *\*v*



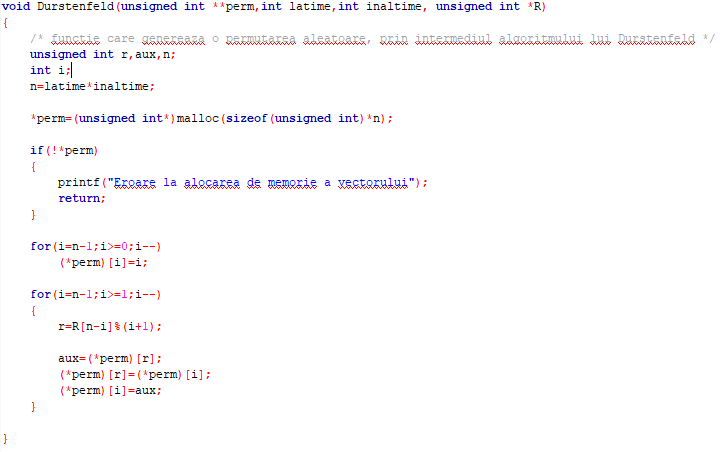
Functia este complementara celei numite *IncarcaImagine*.

**Durstenfeld**

Functia va avea ca parametri:

* *\*perm* – tablou unidimensional in care va fi stocata o permutare aleatoare
* *latime – latimea unei imagini*
* *inaltime – inaltimea unei imagini*
* *\*R – tablou unidimensional cu numere aleatoare*

si construieste prin intermediul algoritmului lui Durstenfeld o permutare aleatoare, bazandu-se pe valorile tabloului *R*.

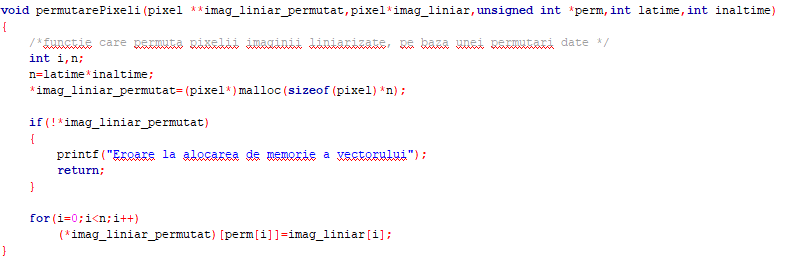


**permutarePixeli**

Functia va avea ca parametri:

* *\*imag\_liniar\_permutat -* tablou unidimensional structura de tip pixel care memoreaza pixelii unei imagini in forma liniara, dupa permutarea acestora
* *\*Imag\_liniar –* tablou unidimensional structura de tip pixel care retine o imagine in forma liniara
* *\*perm –* tablou unidimensional ce retine o permutare aleatoare
* *latime –* latimea unei imagini
* *inaltime –* inaltimea unei imagini

si permuta pixelii unei imagini (retinuta in forma liniarizata – *imag\_liniar*), pe baza unei permutari *perm* si vor fi stocati in tabloul *imag\_liniar\_permutat*.

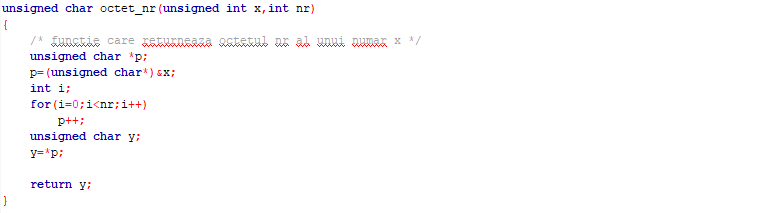
**

**octet\_nr**

Functia va avea ca parametri:

* *x* – un numar natural
* *nr* – un numar

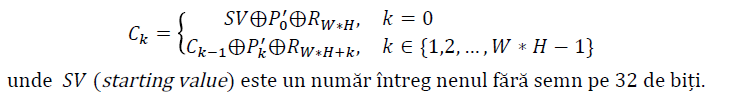
si returneaza al *nr*-lea octet al numarului *x*.



**xorFinal**

Functia va avea ca parametri :

* *\*imag\_liniar* - tablou unidimensional structura de tip pixel care memoreaza pixelii unei imagini in forma liniarizata
* *\*R* – tablou unidimensional in care este memorata o permutare aleatoare
* *latime –* latimea unei imagini
* *inaltime* – inaltimea unei imagini
* *SV* – numar natural (starting value )
* *\*imag\_liniar\_criptat* - tablou unidimensional structura de tip pixel care memoreaza pixelii unei imagini in forma liniara, dupa aplicarea operatiei de xorare pe baza formulei:

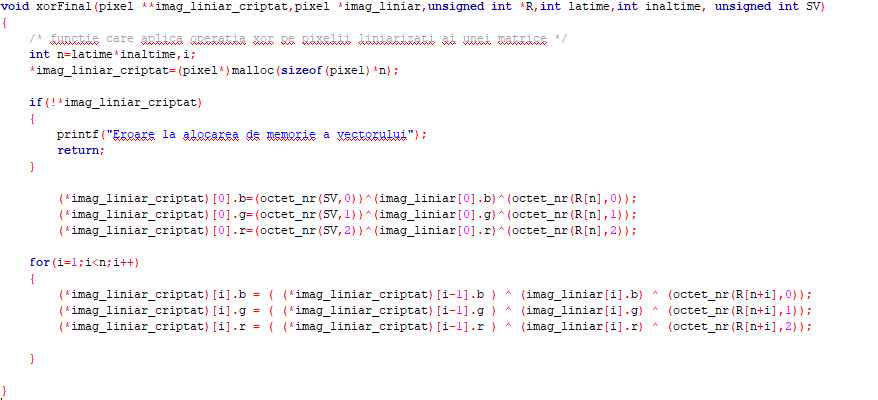
**

Unde C = \*imag\_liniar\_criptat

P’ = \*imag\_liniar

R = \*R

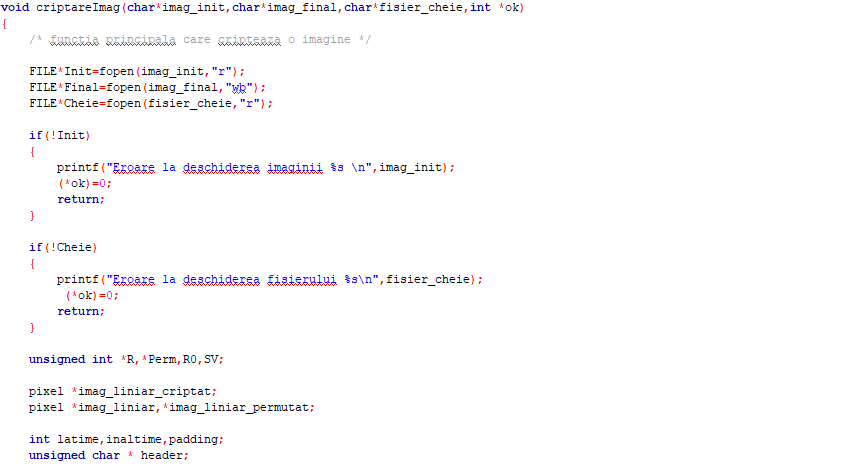
Iar  reprezinta operatia *xor* intre primii 3 octeti ai celor 3 componente

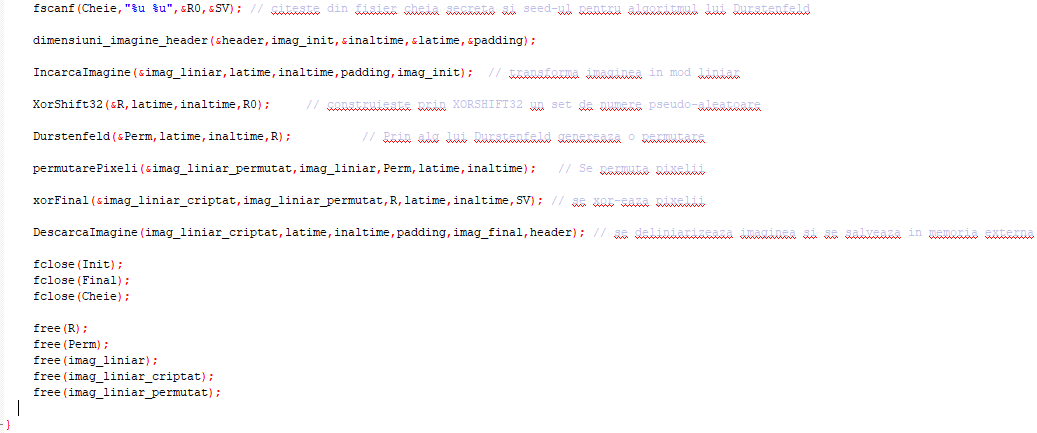


**criptareImag**

Functia va avea ca parametri:

* *\*imag\_init* – sir de caractere care furnizeaza calea imaginii initiale
* *\*imag\_final* – sir de caractere care furnizeaza calea imaginii finale
* *\*fisier\_cheie* – sir de caractere care furnizeaza numele fisierului care contine cheia secreta si seed-ul pentru functia de generare de numere aleatoare
* *ok* – valoare care furnizeaza 1 sau 0 daca s-au gasit fisierele *imag\_initial* si *fisier\_cheie*





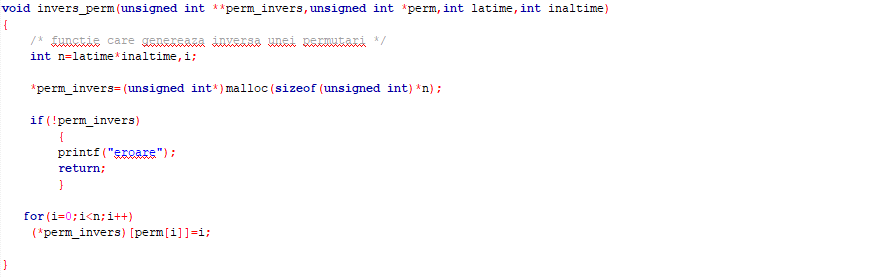
Functia deschide imaginea, respectiv fisierul cheie, iar daca nu reuseste, ok=0 si se opreste. Citeste din fisierul *fisier\_cheie* cheia secreta, repspectiv seed-ul, obtine caracteristicile imaginii *imag\_init* prin apelul *dimensiuni\_imagine\_header(&header,imag\_init,&inaltime,&latime,&padding),* liniarizeaza imaginea prin *IncarcaImagine ( &imag\_liniar, latime, inaltime, padding, imag\_init)* , genereaza un set de numere aleatoare prin algoritmul xorShift32 ( *XorShift32(&R, latime ,inaltime, R0)* ), genereaza o permutare aleatoare prin algoritmul lui Durstenfeld si pe baza numerelor aleatoare generate ( *Durstenfeld(&Perm ,latime, inaltime, R) )* permuta pixelii *permutarePixeli (&imag\_liniar\_permutat, imag\_liniar, Perm, latime, inaltime),* are loc xor-area *xorFinal(&imag\_liniar\_criptat,imag\_liniar\_permutat,R,latime,inaltime,SV)*, iar in final imaginea este deliniarizata *DescarcaImagine(imag\_liniar\_criptat,latime,inaltime,padding,imag\_final,header)* . Dupa salvarea in memoria externa a noii imagini obtinute, *imag\_final*, este eliberata memoria.

**invers\_perm**

Functia va avea ca parametri:

* *\*perm\_invers -* tablou unidimensional care memoreaza permutarea inversaa permutarii *perm*
* *\*perm* – tablou unidimensional care memoreaza o permutare
* *latime* – latimea unei imagini
* *inaltime* – inaltimea unei imagini

Programul primeste o permutare prim intermediul parametrului *perm* si memoreaza in *perm\_invers* permutarea inversa.

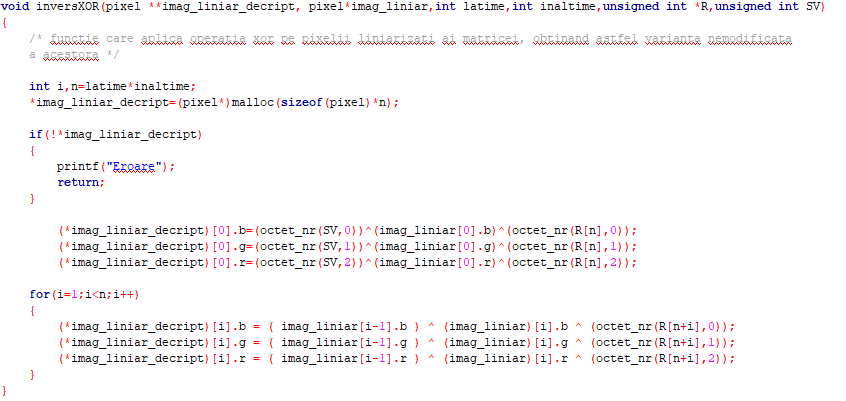


**inversXOR**

Functia primeste ca parametri:

* *\*imag\_liniar\_decript* - tablou unidimensional structura de tip pixel care memoreaza pixelii unei imagini in forma liniarizata, dupa aplicare operatiei de xor-are
* *\*imag\_liniar* - tablou unidimensional structura de tip pixel care memoreaza pixelii unei imagini in forma liniarizata
* *latime* - latimea unei imagini
* *inaltime* – inaltimea unei imagini
* *\*R* – tablou unidimensional care
* *SV* – numar natural (starting value)

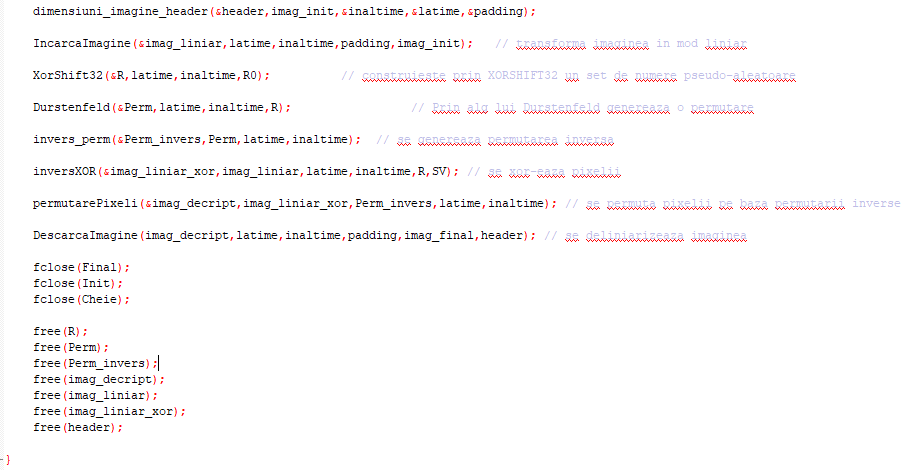
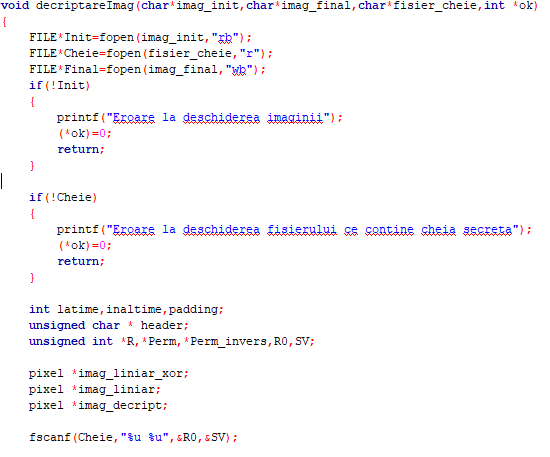
Si aplica operatia de xor-are pe pixelii unei imagini memorate in forma liniarizata



**decriptareImag**

Functia va avea ca parametri:

* *\*imag\_init* – sir de caractere care furnizeaza calea imaginii initiale
* *\*imag\_final* – sir de caractere care furnizeaza calea imaginii finale
* *\*fisier\_cheie* – sir de caractere care furnizeaza numele fisierului care contine cheia secreta si seed-ul pentru functia de generare de numere aleatoare
* *ok* – valoare care furnizeaza 1 sau 0 daca s-au gasit fisierele *imag\_initial* si *fisier\_cheie*



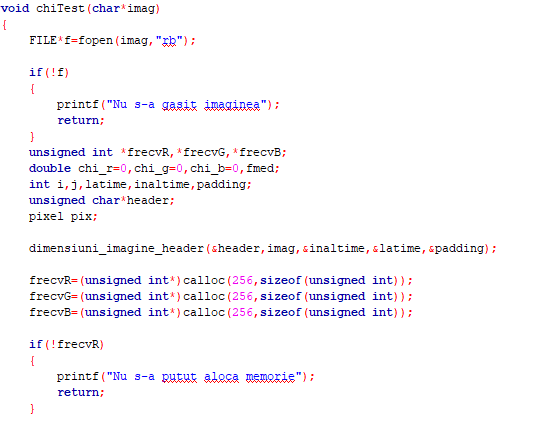
Functia deschide imaginea, respectiv fisierul cheie, iar daca nu reuseste, ok=0 si se opreste. Citeste din fisierul *fisier\_cheie* cheia secreta, repspectiv seed-ul, obtine caracteristicile imaginii *imag\_init* prin apelul *dimensiuni\_imagine\_header(&header,imag\_init,&inaltime,&latime,&padding),* liniarizeaza imaginea prin *IncarcaImagine ( &imag\_liniar, latime, inaltime, padding, imag\_init)* , genereaza un set de numere aleatoare prin algoritmul xorShift32 ( *XorShift32(&R, latime ,inaltime, R0)* ), genereaza o permutare aleatoare prin algoritmul lui Durstenfeld precum si inversa sa ( *invers\_perm(&Perm\_invers,Perm,latime,inaltime)* ) si pe baza numerelor aleatoare generate ( *Durstenfeld(&Perm ,latime, inaltime, R)* are loc xor-area inversa *inversXOR( &imag\_liniar\_xor, imag\_liniar, latime, inaltime, R, SV)*, se permuta pixelii pe baza permutarii inverse obtinute *permutarePixeli( &imag\_decript, imag\_ liniar\_xor, Perm\_invers, latime, inaltime)* iar in final imaginea este deliniarizata *DescarcaImagine(imag\_liniar\_criptat,latime,inaltime,padding,imag\_final,header)* . Dupa salvarea in memoria externa a noii imagini obtinute, *imag\_final*, este eliberata memoria.

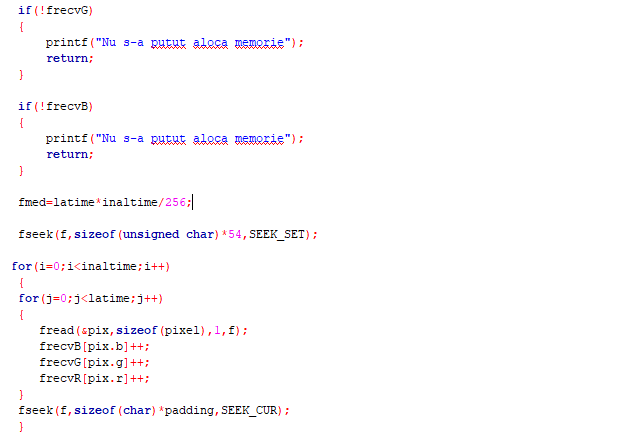
**chiTest**

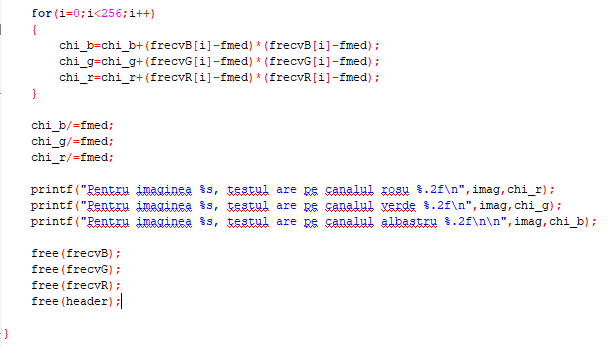
Functia va avea ca parametru:

* *\*imag* – sir de caractere care furnizeaza calea unei imagini

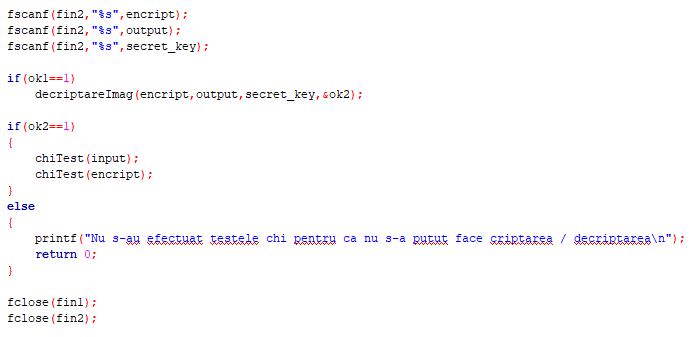
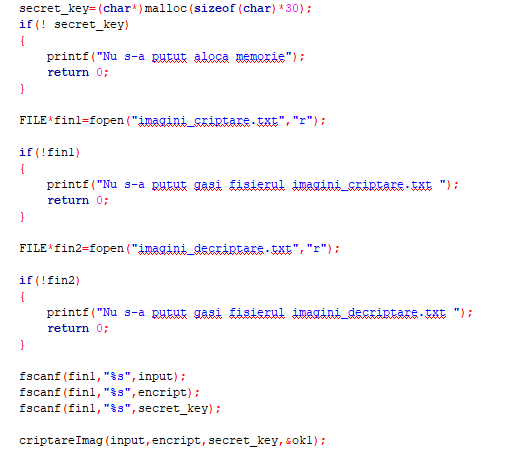
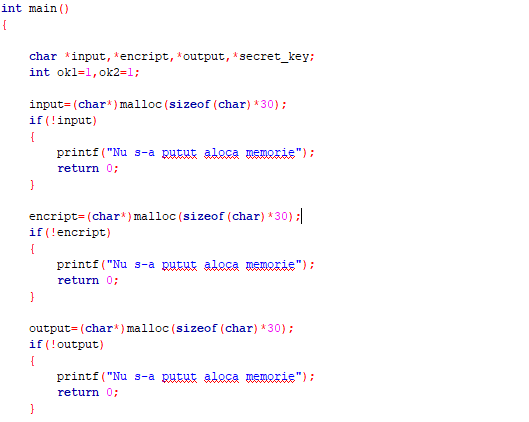
si afiseaza valorile testului chi pentru cele 3 canale RGB ale imaginii *imag*







In primare partea a main-ului, am apelat functiile de criptare si decriptare pentru o imagine citita dintr-un fisier:



Se aloca spatiu pentru sirurile de caractere ce vor memora numele fisierelor, se vor citi din fisierul imagini\_criptare.txt numele imaginii care va fi criptata , numele imaginii in forma criptata si fisierul ce contine cheia secreta si starting value. Se deschide fisierul imagini\_decriptare.txt ce contine numele imaginii in forma criptata, numele imaginii dupa ce a fost decriptata, precum si fisierul ce contine cheia secreta si starting value, iar in final se va afisa rezultatul testului chi pentru fiecare canal RGB.

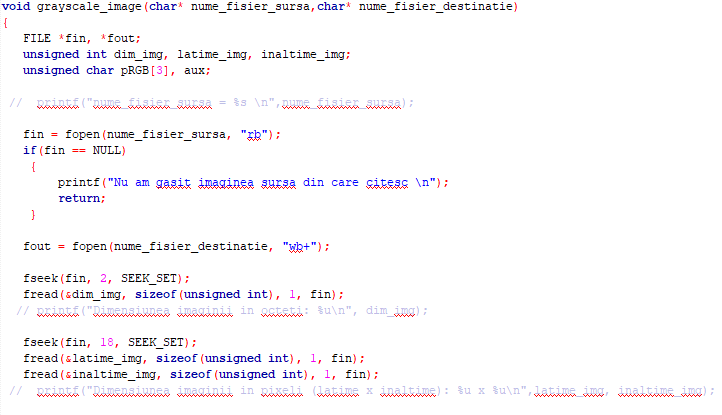
**Template matching**

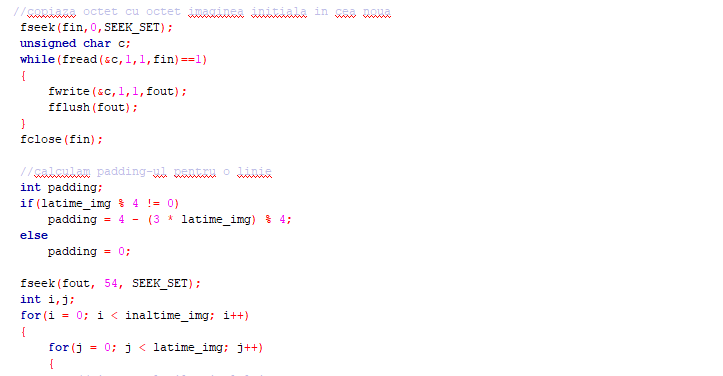
In cea de-a doua parte vor fi descrisi pasii prin care se doreste recunoasterea unor pattern-uri intr-o imagine.

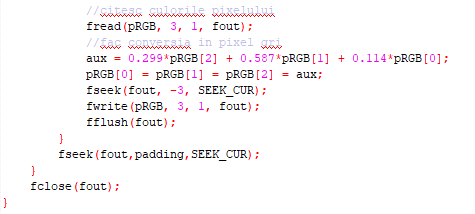
**grayscale\_image**

Functia va avea ca parametri:

* *\*nume\_fisier\_sursa* - sir de caractere care furnizeaza calea imaginii initiale
* *\*nume\_fisier\_destinatie* - sir de caractere care furnizeaza calea imaginii finale







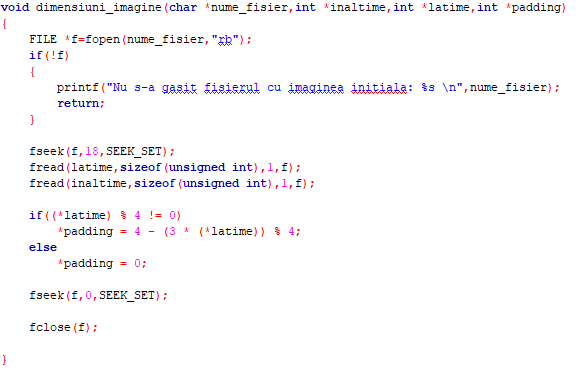
Functia transforama o imagine normala intr-una grayscale, folosindu-se de formula :

* R’ = noul octet pentru canlul rosu
* G’ = noul octet pentru canalul verde
* B’ = noul octet pentru canalul albastru

**dimensiuni\_imagine**

Functia va avea ca parametri:

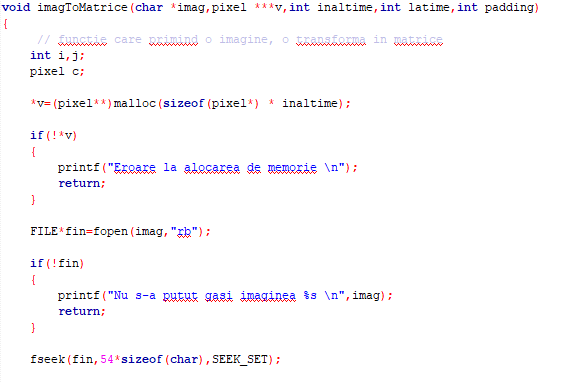
* *\*nume\_fisier* – sir de caractere care furnizeaza calea unei imagini
* *inaltime* - inaltimea imaginii
* *latime* - latimea imaginii
* *padding* – padding-ul imaginii

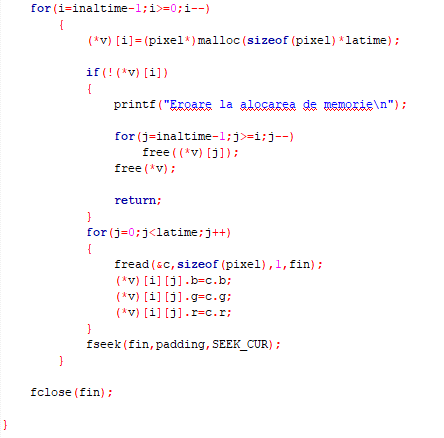


**imagToMatrice**

Functia va avea ca parametri:

* *\*imag* - sir de caractere ce furnizeaza calea unei imagini
* \*\*v – tablou unidimensional structura de tip pixel in care va fi retinuta imaginea (colt stanga sus -> dreapta jos)
* *inaltime* – inaltimea imaginii
* *latime* – latimea imaginii
* *padding* – padding-ul imaginii





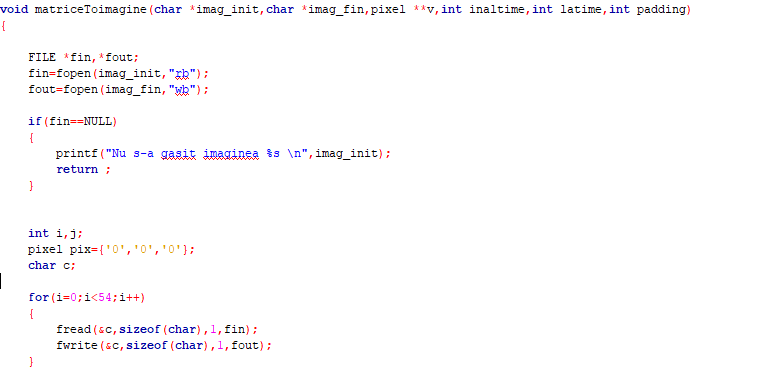
Functia va transforma o imagine intr-un tablou bidimensional pentru a o prelucra mai usor.

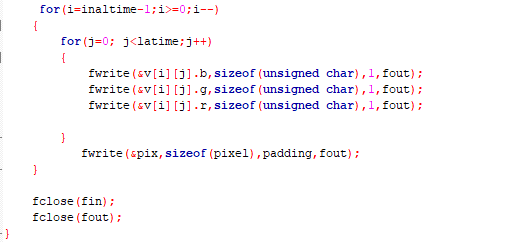
**matriceToimagine**

Functia va avea ca parametri:

* *imag\_init* – sir de caractere ce transmite calea imaginii initiale
* *imag\_final* – sir de caractere ce transmite numele imaginii dupa ce va fi transformata din matrice in imagine
* *\*\*v* – tablou bidimensional ce memoreaza o imagine
* *inaltime* – inaltimea imaginii
* *latime* - latimea imaginii
* *padding* – padding-ul imaginii

Functia este simetrica celei imagineToMatrice, salvand in memoria externa o imagine salvata sub forma unui tablou bidimensional de tip pixel. Este nevoie si de imaginea initiala, in forma normala, pentru header-ul acesteia.



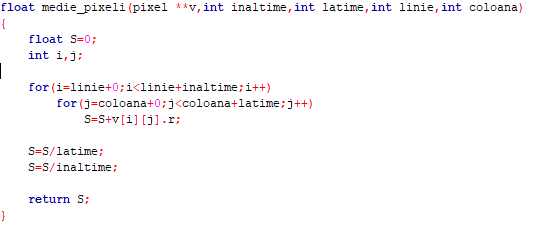


**medie\_pixeli**

Functia va avea ca parametri:

* *\*\*v* – tablou bidimensional ce memoreaza o imagine
* *latime* – latimea unei imagini
* *inaltime* – inaltimea unei imagini
* *linie* – linia tabloului de unde se va calcula media pixelilor
* *coloana* – coloana tabloului de unde se va calcula media pixelilor

Functia va calcula media intensitatii pixelilor imaginii memorate in tabloul *\*\*v* incepand cu coordonatele (linie,coloana) pentru o imagine de dimensiuni (latime,inaltime)

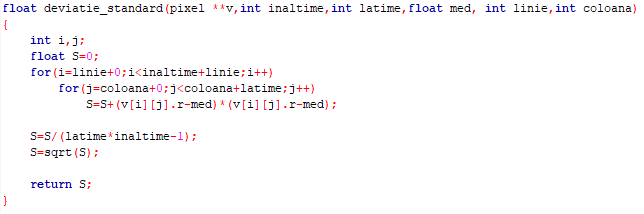


**deviatie\_standard**

* *\*\*v* - tablou bidimensional ce memoreaza o imagine
* *latime* – latimea unei imagini
* *inaltime* – inaltimea unei imagini
* *med* – media intensitatii pixelilor pentru portiunea respectiva
* *linie* – linia tabloului de unde se va calcula deviatia standard
* *coloana* – coloana tabloului de unde se va calcula deviatia standard

Functia calculeaza deviatia standard a unei portiuni dintr-o imagine, incepand cu coordonatele (linie,coloana) pentru un sablon de dimensiuni (inaltime,latime), cu formula:



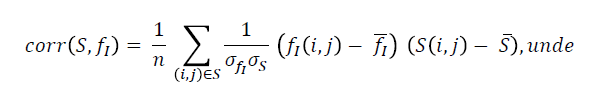


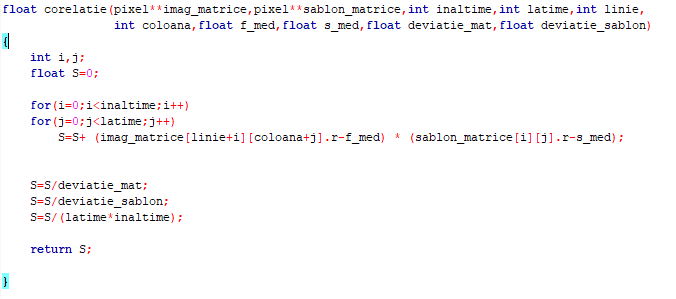
**corelatie**

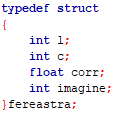
Functia va avea ca parametri:

* *\*\*imag\_matrice* – tablou bidimensional ce memoreaza o imagine
* *\*\*sablon\_matrice* – tablou bidimensional ce memoreaza o imagine
* *inaltime* – inaltimea imaginii *sablon\_matrice*
* *latime* – latimea imaginii *sablon\_matrice*
* *linie* – linia tabloului *imag\_matrice*  de unde se va calcula corelatia dintre cele 2 imagini
* *coloana* – coloana tabloului *imag\_matrice*  de unde se va calcula corelatia dintre cele 2 imagini
* *f\_med* – *intensitatea* medie a pixelilor a imaginii *imag\_matrice*
* s\_med – intensitatea medie a pixelilor a imaginii *sablon\_matrice*
* *deviatie\_mat* – deviatia standard a imaginii *imag\_matrice*
* *deviatie\_sablon* – deviatia standard a imaginii *sablon\_matrice*

Functia calculeaza corelatia dintre portiunea imaginii *imag\_matrice* incepand cu coordonatele (linie,coloana) cu imaginea *sablon\_matrice*, dupa formula:







Am definit o structura ce retine linia,coloana, corelatia si imaginea (cifra reprezentata) pentru o anumita fereastra

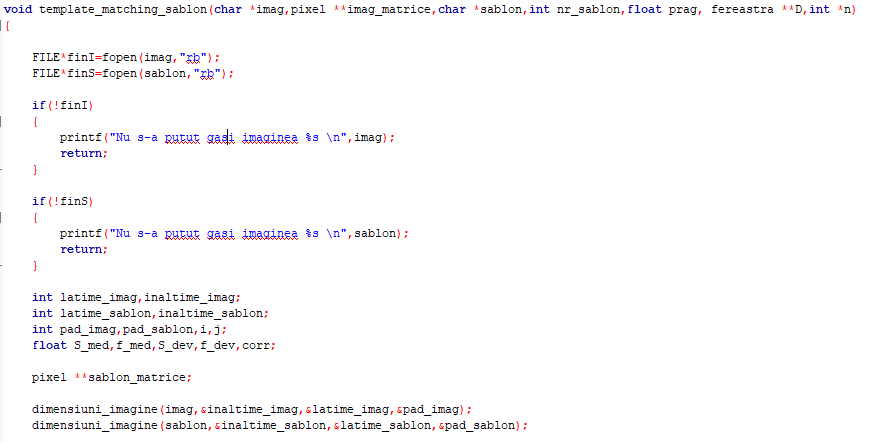
**template\_matching\_sablon**

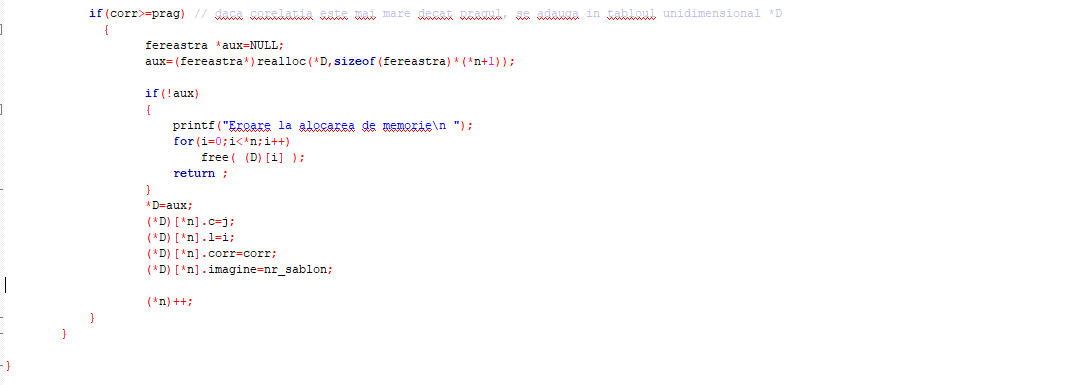
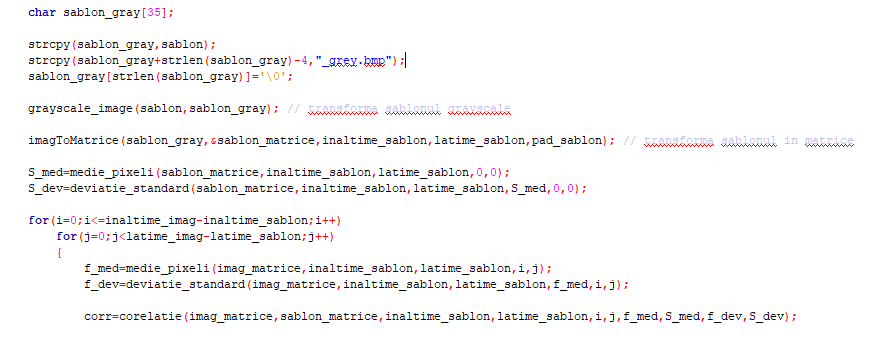
Functia va acea ca parametri:

* *\*imag* – sir de caractere care transmite calea unei imagini
* *\*\*imag\_matrice* – tabloul bidimensional de tip pixel in care este memorata imaginea *imag*
* *\*sablon* – sir de caractere care transmite calea unei imagini
* *nr\_sablon* – numar natural
* *prag* – numar real
* *\*D* – tablou unidimensional de tip *fereastra*
* *n* – numarul de elemente al tabloului *\*D*

Functia primeste ca parametru o imagine *imag*, si transformata in forma sa de matrice, precum si un sablon *sablon*, nr\_sablon care reprezinta un numar natural reprezentat in imaginea *sablon*, un numar real *prag* precum si un tablou unidimensional de tip *fereastra* in care vor fi stocate toate ferestrele pentru care corelatia dintre *sablon* si o portiune din imaginea *imag* este mai mare decat pragul respectiv.

Functia deschide fiecare imagine si obtine principalele caracteristici ale fiecare imagini, prin apelurile dimensiuni\_imagine(imag, &inaltime\_imag, &latime\_imag, &pad\_imag) si dimensiuni\_imagine ( sablon, &inaltime\_sablon, &latime\_sablon, &pad\_sablon). Transforma imaginea sablon in grayscale grayscale\_image(sablon,sablon\_gray) , transforma noua imagine grayscale intr-un tablou bidimensional imagToMatrice(sablon\_gray,&sablon\_matrice,inaltime\_sablon,latime\_sablon,pad\_sablon) , calculeaza intensitatea medie a pixelilor si deviatia standard a sablonului S\_med=medie\_pixeli(sablon\_matrice, inaltime\_sablon, latime\_sablon, 0, 0), S\_dev=deviatie\_standard(sablon\_matrice, inaltime\_sablon, latime\_sablon, S\_med, 0, 0), iar pentru fiecare pozitie a tabloului bidimensional *\*\*imag\_matrice* calculeaza corelatia cu sablonului grayscale corr=corelatie(imag\_matrice, sablon\_matrice, inaltime\_sablon, latime\_sablon, i, j, f\_med, S\_med, f\_dev, S\_dev) si daca este mai mare decat *prag*, se retine in tabloul \*D.



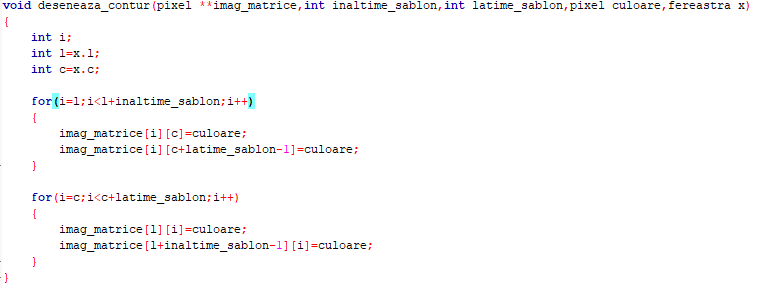


**deseneaza\_contur**

Functia va avea ca parametri:

* *\*\*imag\_matrice*- tablou bidimensional ce memoreaza o imagine
* *inaltime\_sablon*- inaltimea unui sablon (imaginile cu cifrele 0,1 ...9 )
* *latime\_sablon* – latimea unui sablon (imaginile cu cifrele 0,1…9)
* *culoare* – structura de tip pixel ce semnifica o culoare specificata printr-un triplet RGB
* *x* – structura de tip fereastra ce reprezinta o portiune a unei imagini

si deseneza in tabloul unidimensional *imag\_matrice* conturul unui sablon cu latimea si inaltimea date ca parametri si de culoarea *culoare*, plecand de la coordonatele initiale din fereastra X.

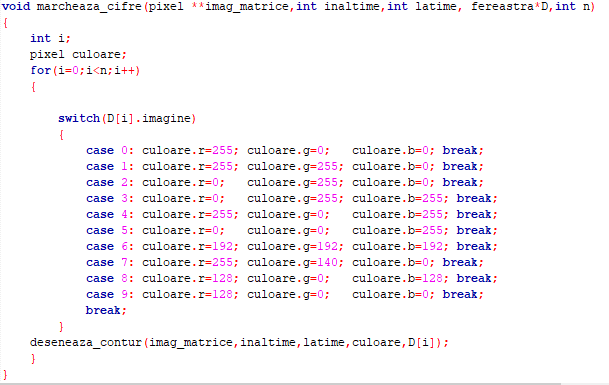


**marcheaza\_cifre**

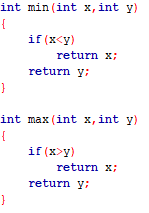
Functia va avea ca parametri:

* *\*\*imag\_matrice* – tablou bidimensional ce memoreaza o imagine
* *inaltime* – inaltimea unui sablon
* *latime* – latimea unui sablon
* *\*D* – tablou unidimensional structura de tip pixel in care sunt retinute ferestrele
* *n* – numarul de elemente al tabloului \*D

si pentru fiecare fereastra, va decide cu ce culoare va fi desenat conturul.



**min max**

****

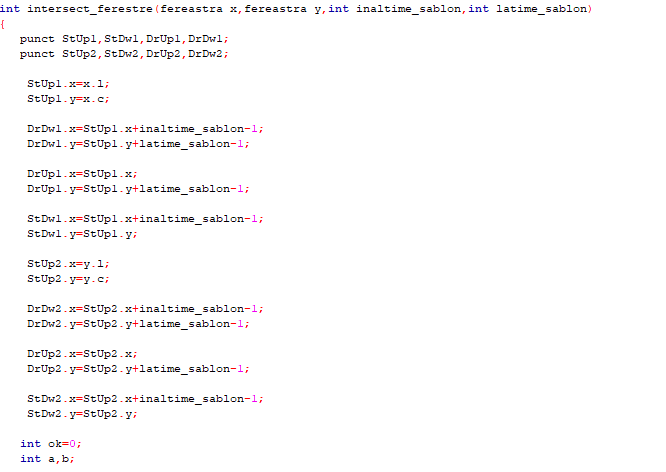
Functii care returneaza cel mai mic sau cel mai mare element dintre 2 numere date ca parametru

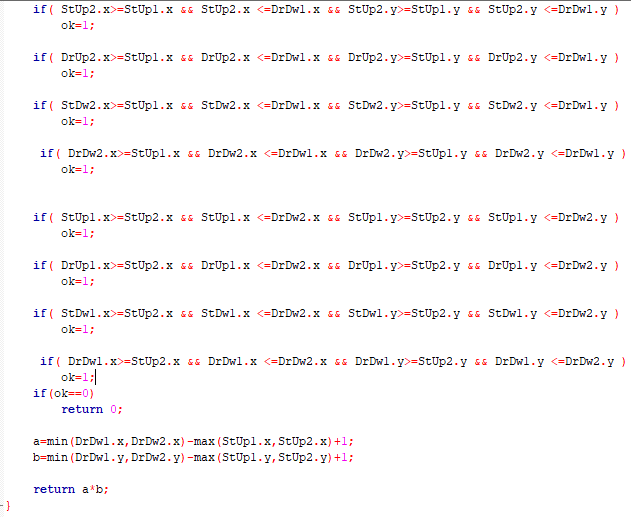
**intersect\_ferestre**

Functia va avea ca parametri:

* *x*  - variabila de tip fereastra
* *y* – variabila de tip fereastra
* *inaltime\_sablon* – inaltimea unui sablon
* *latime\_sablon* – latimea unui sablon

si va calcula aria de interesectie dintre doua ferestre. Functia calculeaza coordonatele celor 4 colturi ale fiecarei ferestre si decide daca cele 2 se intersecteaza ,iar in caz afirmativ, va retruna aria de intersectie.

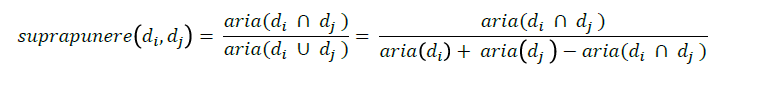




**suprapunere**

* *x* – variabila de tip fereastra
* *y* – variabila de tip fereastra
* *inaltime* – inaltime unui sablon
* *latime* – latimea unui sablon

si va returna suprapunerea spatiala a 2 ferestre, data de formula:

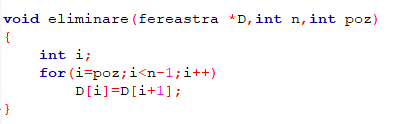


**eliminare**

Functia va avea ca parametri:

* *\*D* – tablou unidimensional structura de tip pixel in care sunt retinute ferestrele
* *n* – numarul de elemente al tabloului *\*D*
* poz – numar natural

si va elimina elementul de pe pozitia  *poz*.

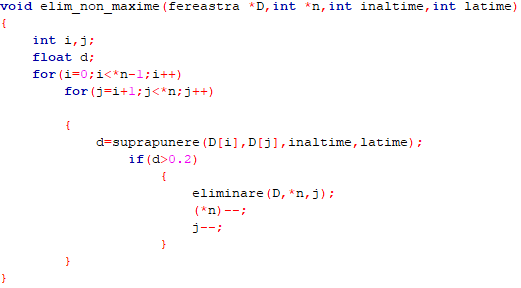


**elim\_non\_maxime**

Functia va avea ca parametri:

* *\*D* – tablou unidimensional structura de tip pixel in care sunt retinute ferestrele
* *n* – numarul de elemente
* *inaltime* – inaltimea unui sablon
* *latime* – latimea unui sablon

si compara cate 2 ferestre, iar in cazul in care suprapunerea spatiala este mai mare de 0.2, se elimina cea cu corelatia mai mica.

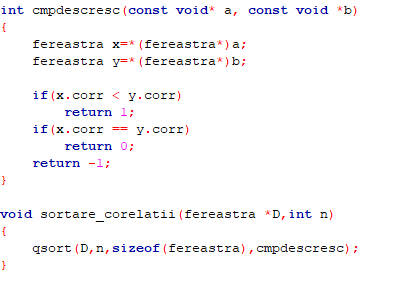


**sortare\_corelatii**

Functia avea ca parametri:

* *\*D* - tablou unidimensional structura de tip pixel in care sunt retinute ferestrele
* *n* – numarul de elemente

si sorteaza tabloul descrescator in functie de corelatii



In cea de-a doua parte a main-ului, are loc operatia de template matching dintre o imagine si diferite sabloane. Intai se deschide fisierul *imagini.txt* care contine numele imaginii, numele imaginii dupa ce aceasta a fost criptata, precum si numele fisierului ce contine cheia secreta. Se creeaza numele noii imagini in grayscale ( strcpy (imagineI\_gray, imagineI ), strcpy( imagineI\_gray + strlen ( imagineI ) -4, "\_gray.bmp ") )si se transforma in grayscale ( grayscale\_image(imagineI, imagineI\_gray) ). Se obtin dimensiunile imaginii dimensiuni\_imagine( imagineI\_gray, &inaltime, &latime, &pad) ) dupa care ambele imagini (initiala si grayscale) se transforma in matrice: imagToMatrice( imagineI\_gray, &imagineI\_matrice\_gray, inaltime, latime, pad) si imagToMatrice( imagineI, &imagineI\_matrice, inaltime, latime, pad). Pentru fiecare din cele 10 sabloane are loc operatia de template matching: se citeste si deschide fiecare imagine si apeleaza functia de template matching dintre imaginea grayscale si un sablon template\_matching\_sablon( imagineI\_gray, imagineI\_matrice\_gray, cifra\_sablon, i, prag, &D, &n) . Are loc operatia de sortare a corelatiilor ( sortare\_corelatii(D,n) ) si eliminare a non-maximelor ( elim\_non\_maxime(D, &n, inaltime\_sablon, latime\_sablon) ), se marcheaza pe matricea imaginii initiale conturul fiecarei ferestre ramase ( marcheaza\_cifre(imagineI\_matrice, inaltime\_sablon, latime\_sablon, D, n) )si are loc salvarea in memoria externa a noii imagini ( matriceToimagine(imagineI, "template\_final.bmp", imagineI\_matrice, inaltime, latime, pad) ), cu denumirea template\_final.bmp. In final, are loc eliberarea de memorie.