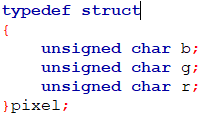
**Criptare imagine**

In cadrul acestui proeiect pentru facucltate, am realizat un program care primeste ca input o imagine in format .bmp si o cripteaza / decripteaza, aplicand transformari asupra pixelilor.

**Memorare**



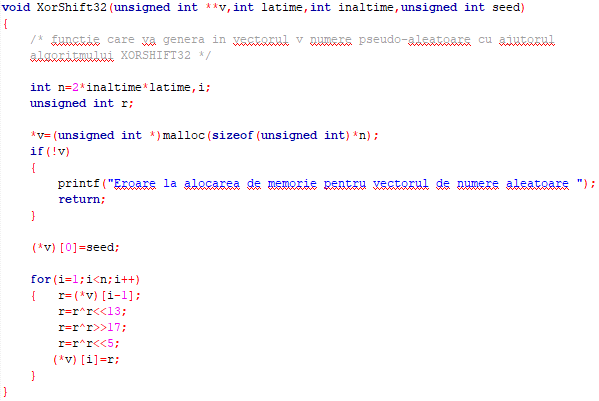
Am definit o structura “pixel” pentru a memora convenabil continutul unei imagini. Vor fi stocati cei 3 octeti ai unui pixel, pentru canalele rosu, verde şi albastru.

**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.

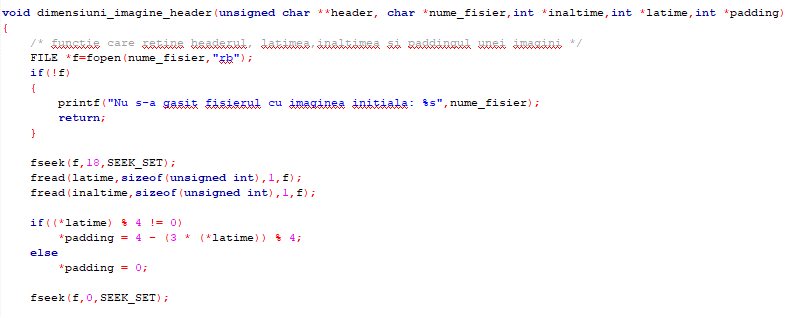


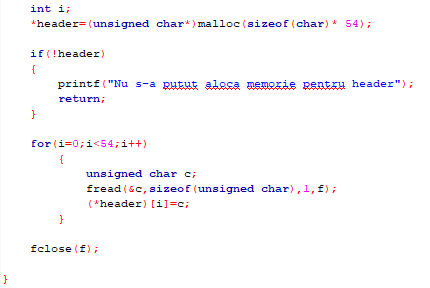
**dimensiuni\_imagine\_header**

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





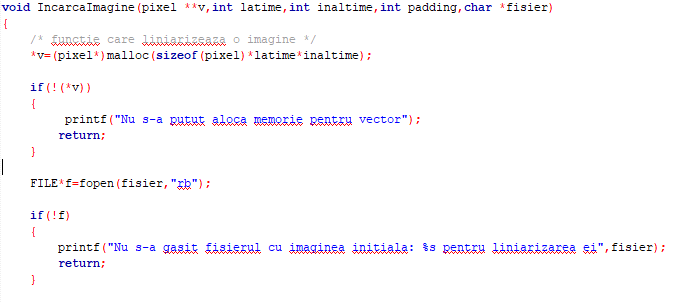
Functia deschide imaginea furnizata in modul “rb” si se pozitioneaza pe pozitia celui de al 18-lea octet pentru a citi latimea imaginii, iar in continuare inaltimea acestuia si calculeaza paddingul. Dupa care se pozitioneaza la inceput, si citeste primii 54 octeti, reprezentand header-ul imaginii.

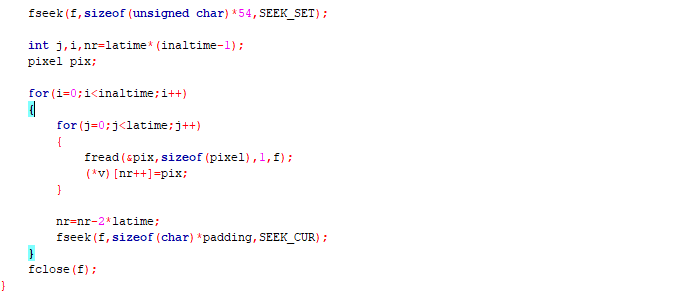
**IncarcaImagine**

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.





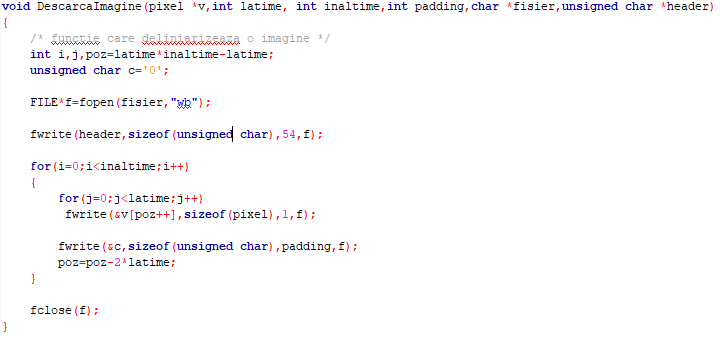
Functia sare peste primii 54 de octeti, ce reprezinta header-ul, dupa care citeste cate un pixel ( 3 octeti fara semn ), punandu-l pe pozitia nr, actualizata dupa fiecare atribuire, in tabloul unidimensional *\*v* .

**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*



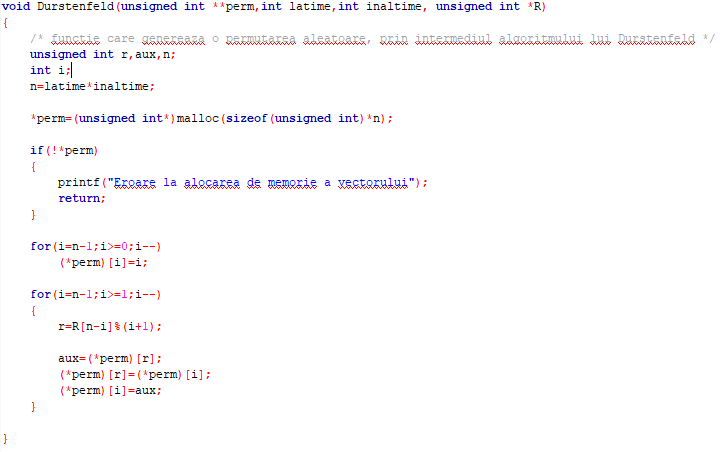
Functia este complementara celei numite *IncarcaImagine*.

**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*.

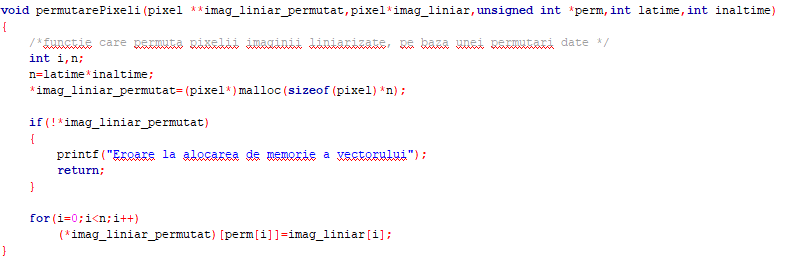


**permutarePixeli**

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*.

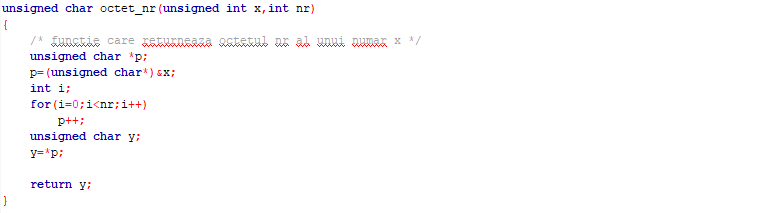
**

**octet\_nr**

Functia va avea ca parametri:

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

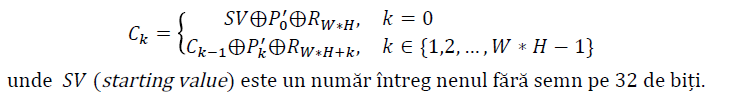
si returneaza al *nr*-lea octet al numarului *x*.



**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:

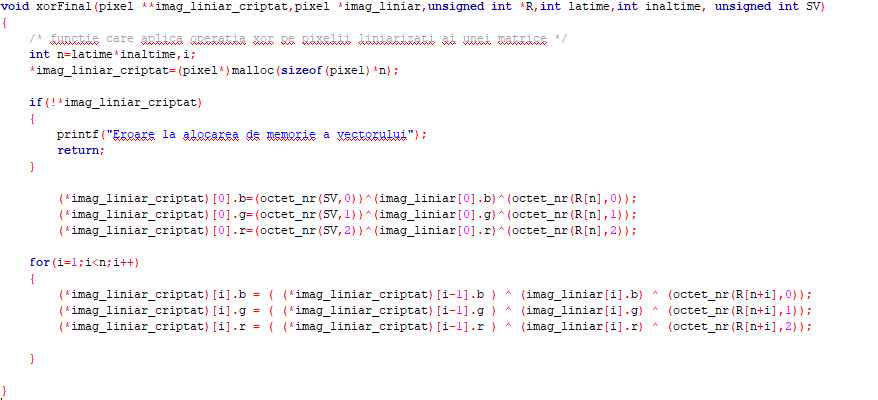
**

Unde C = \*imag\_liniar\_criptat

P’ = \*imag\_liniar

R = \*R

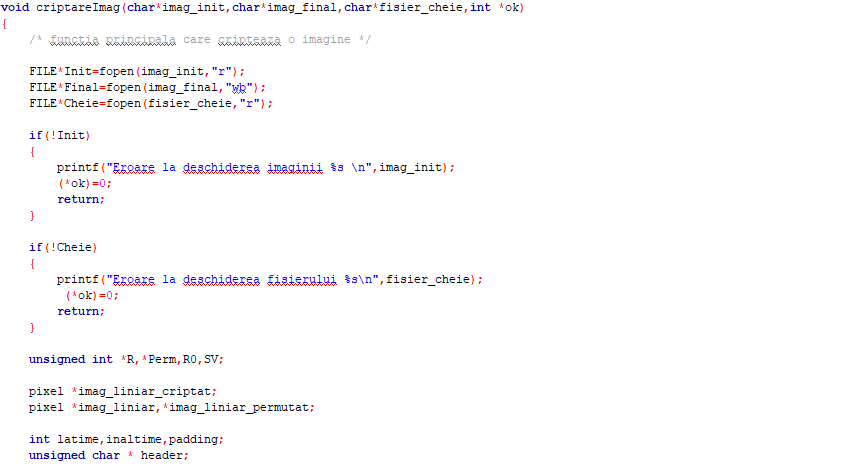
Iar  reprezinta operatia *xor* intre primii 3 octeti ai celor 3 componente

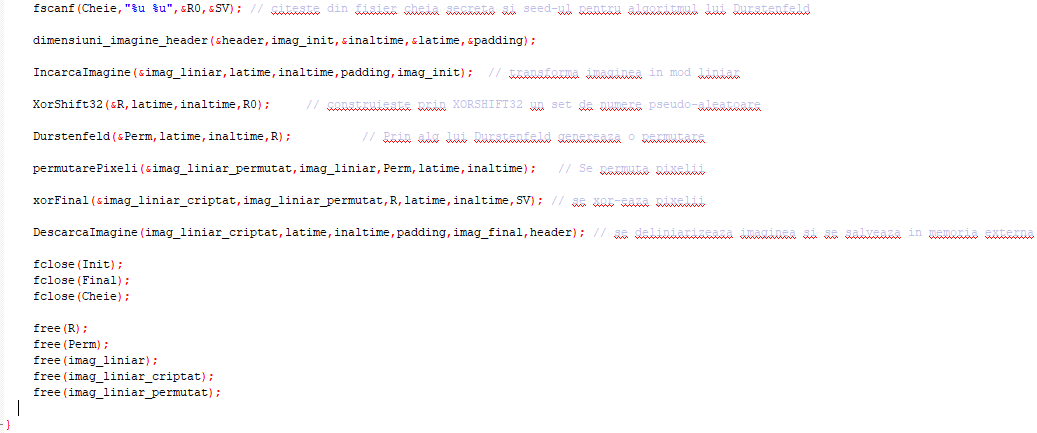


**criptareImag**

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*





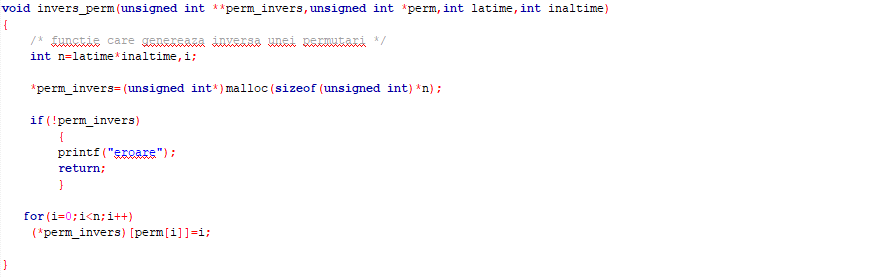
Functia deschide imaginea, respectiv fisierul cheie, iar daca nu reuseste, ok=0 si se opreste. Citeste din fisierul *fisier\_cheie* cheia secreta, repspectiv seed-ul, obtine caracteristicile imaginii *imag\_init* prin apelul *dimensiuni\_imagine\_header(&header,imag\_init,&inaltime,&latime,&padding),* 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.

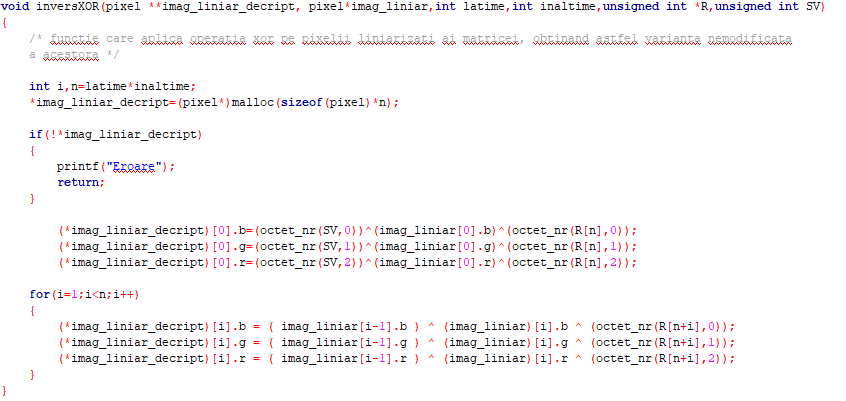


**inversXOR**

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)

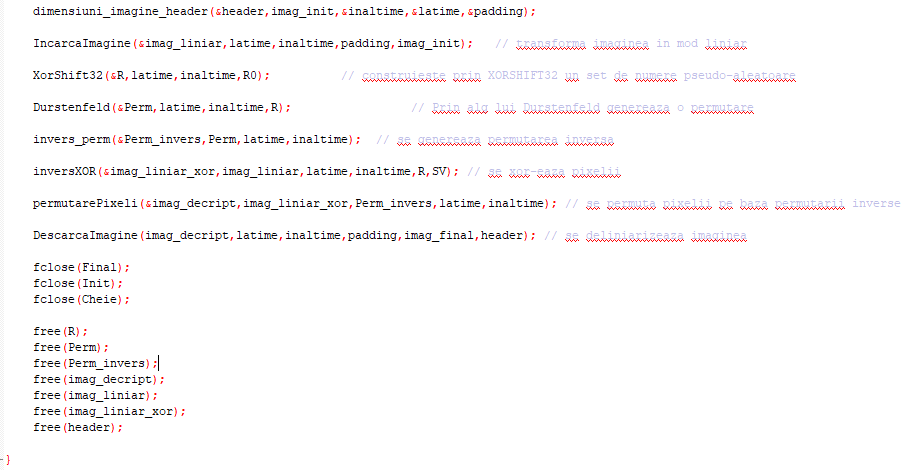
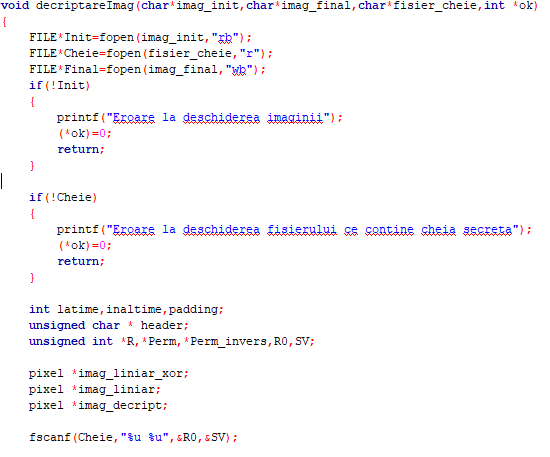
Si aplica operatia de xor-are pe pixelii unei imagini memorate in forma liniarizata



**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*



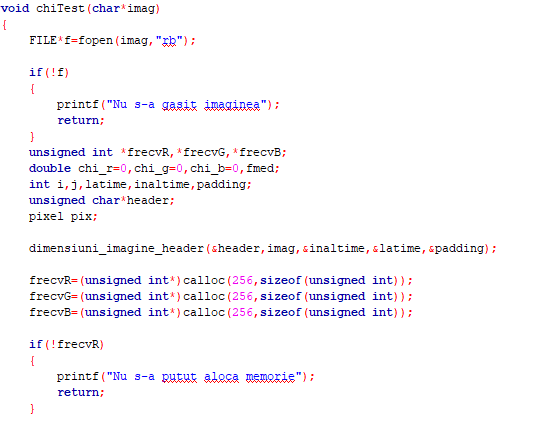
Functia deschide imaginea, respectiv fisierul cheie, iar daca nu reuseste, ok=0 si se opreste. Citeste din fisierul *fisier\_cheie* cheia secreta, repspectiv seed-ul, obtine caracteristicile imaginii *imag\_init* prin apelul *dimensiuni\_imagine\_header(&header,imag\_init,&inaltime,&latime,&padding),* 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.

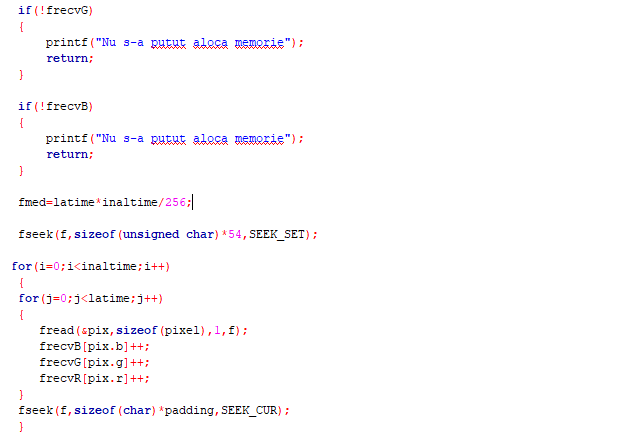
**chiTest**

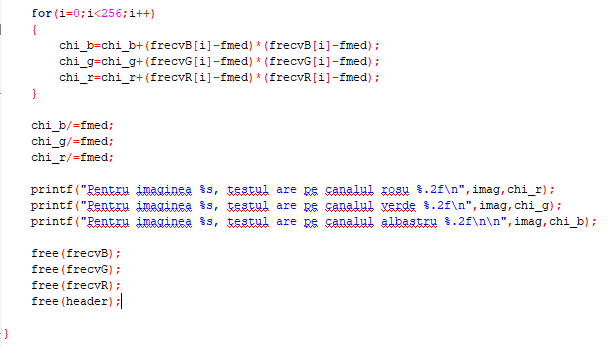
Functia va avea ca parametru:

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

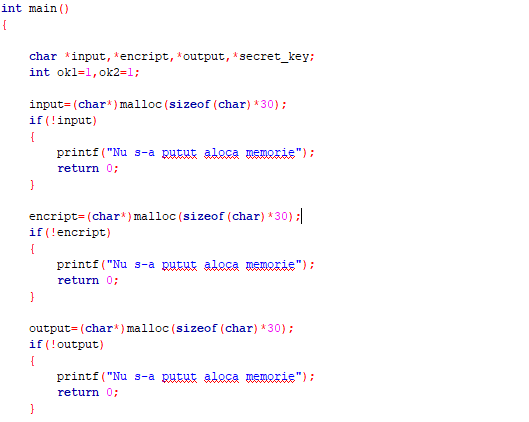
si afiseaza valorile testului chi pentru cele 3 canale RGB ale imaginii *imag.* Daca valorile testelor chi nu depasesc 293.15, inseamna ca procesul de criptare a avut loc cu success

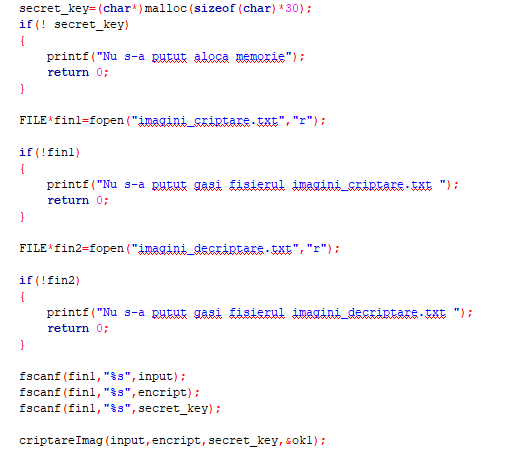


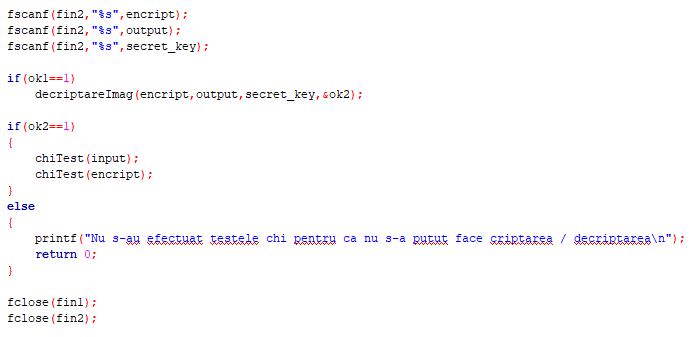




In primare partea a main-ului, am apelat functiile de criptare si decriptare pentru o imagine citita dintr-un fisier:







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.