Programare Procedurala 2018
-documentație proiect-

Stoleru Vlad-Ștefan

Criptarea unei imagini în format BMP

- În interiorul funcţiei ***main*** sunt alocate şi citite tablouri ce vor reprezenta căile către fişierele BMP folosite/ ce urmează sa fie create. În cazul în care nu se poate efectua alocarea, programul se termina prematur.
- După ce au fost alocate şi citite fiecare cale de acces, se apelează funcţia ***encryptBMP,*** aflată în fişierul-antet ***encryptions.h.***
- In interiorul acesteia se va încărca imaginea sub forma liniarizată în tabloul ***pixelArray*** prin intermediul funcţiei ***loadBMPLiniar***, aflată în fişierul-antet ***bitmap.h***. Dacă în cadrul încărcării au apărut erori (ce vor fi detaliate în cadrul funcţiei de încărcare), funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- Se deschide fişierul ce conţine cheia secreta şi se citesc cele doua numere reprezentând seed-ul pentru generator ***R0*** şi numărul cu care se iniţializează substituţia pixelilor, ***SV***. Dacă a apărut o eroare în cadrul deschiderii fişierului sau a citirii celor doua valori, funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- Se aloca memorie pentru tabloul de numere aleatoare (***randArray***) şi pentru permutare (***permArray***). Dacă alocarea nu s-a putut efectua, funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1. Numerele aleatoare vor fi generate prin intermediul funcţiei ***getRandArray*** având valoarea ***R0*** printre parametrii. Funcţia se află în fişierul-antet ***random.h*** şi furnizează valori aleatoare folosind algoritmul XORSHIFT32.
- Se iniţializează ***permArray*** cu valorile corespunzătoare permutării identice, urmând să se aplice transpoziţii utilizând numerele din ***randArray*** în cadrul algoritmului Durstenfeld: se porneşte de la ultima poziţie (*pixelNo - 1*) şi se transpune cu elementul de pe poziţia corespunzătoare din ***randArray*** (*pixelNo* - 1 - *iterator)*, luat modulo *iterator* + 1, astfel încât poziţia să se afle în intervalul (0*, iterator*).
- Se aloca un tablou ***auxArray*** în care se va reţine configuraţia rezultata în urma aplicării permutării ***permArray*** asupra tabloului ***pixelArray***. Similar, în cazul unei alocări nereuşite, funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- Se aplica substituţii la nivel de element în cadrul tabloului ***auxArray***, folosind

operaţii XOR la nivel de pixeli şi intre pixel şi un număr; primul element se va XOR-a cu valoarea ***SV*** şi numărul aleator de pe poziţia *pixelNo* - 1, următoarele elemente fiind XOR-ate cu valorile anterioare şi numărul aleator corespunzător.

- În final se salvează noua configuraţie la calea precizată prin al doilea parametru al funcţiei ***encryptBMP*** şi se verifică apariţia unor eventuale erori la salvare (ce vor fi detaliate în cadrul funcţiei de salvare). În cazul unei detecţii pozitive, funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- Se returnează valoarea 0, cu semnificaţia că funcţia nu a întâmpinat o eroare pe parcurs.
- La întoarcerea în funcţia ***main***, se verifica valoarea returnata de ***encryptBMP*****,** afişându-se un mesaj sugestiv în funcţie de aceasta.

Decriptarea unei imagini în format BMP

- În interiorul funcţiei ***main*** sunt alocate şi citite tablouri ce vor reprezenta căile către fişierele BMP folosite/ ce urmează sa fie create. În cazul în care nu se poate efectua alocarea, programul se termina prematur.
- După ce au fost alocate şi citite fiecare cale de acces, se apelează funcţia ***decryptBMP,*** aflată în fişierul-antet ***encryptions.h.***
- In interiorul acesteia se va încărca imaginea sub forma liniarizată în tabloul ***pixelArray*** prin intermediul funcţiei ***loadBMPLiniar***, aflată în fişierul-antet ***bitmap.h***. Dacă în cadrul încărcării au apărut erori (ce vor fi detaliate în cadrul funcţiei de încărcare), funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- Se deschide fişierul ce conţine cheia secreta şi se citesc cele doua numere reprezentând seed-ul pentru generator ***R0*** şi numărul cu care se iniţializează substituţia pixelilor, ***SV***. Dacă a apărut o eroare în cadrul deschiderii fişierului sau a citirii celor doua valori, funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- Se aloca memorie pentru tabloul de numere aleatoare (***randArray***) şi pentru permutare (***permArray***). Dacă alocarea nu s-a putut efectua, funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1. Numerele aleatoare vor fi generate prin intermediul funcţiei ***getRandArray*** având valoarea ***R0*** printre parametrii. Funcţia se află în fişierul-antet ***random.h*** şi furnizează valori aleatoare folosind algoritmul XORSHIFT32.
- Se aplica substituţii la nivel de element în cadrul tabloului ***auxArray***, folosind

operaţii XOR la nivel de pixeli şi intre pixel şi un număr, în ordine inversă faţă de algoritmul de criptare (de la ultimul element la primul element); primul element se va XOR-a cu valoarea ***SV*** şi numărul aleator de pe poziţia *pixelNo* - 1, următoarele elemente fiind XOR-ate cu valorile anterioare şi numărul aleator corespunzător.

- Se iniţializează ***permArray*** cu valorile corespunzătoare permutării identice, urmând să se aplice transpoziţii utilizând numerele din ***randArray*** în cadrul algoritmului Durstenfeld: se porneşte de la ultima poziţie (*pixelNo - 1*) şi se transpune cu elementul de pe poziţia corespunzătoare din ***randArray*** (*pixelNo* - 1 - *iterator)*, luat modulo *iterator* + 1, astfel încât poziţia să se afle în intervalul (0*, iterator*).
- Se calculează inversa permutării ***permArray***, ce va fi reţinută în tabloul ***inversePerm***, alocat dinamic. Dacă alocarea nu s-a putut efectua, funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- Se aloca un tablou ***auxArray*** în care se va reţine configuraţia rezultata în urma aplicării permutării ***inversePerm*** asupra tabloului ***pixelArray***. Similar, în cazul unei alocări nereuşite, funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- În final se salvează noua configuraţie la calea precizată prin al doilea parametru al funcţiei ***decryptBMP*** şi se verifică apariţia unor eventuale erori la salvare (ce vor fi detaliate în cadrul funcţiei de salvare). În cazul unei detecţii pozitive, funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- Se returnează valoarea 0, cu semnificaţia că funcţia nu a întâmpinat o eroare pe parcurs.
- La întoarcerea în funcţia ***main***, se verifica valoarea returnata de ***decryptBMP*****,** afişându-se un mesaj sugestiv în funcţie de aceasta.

Rularea testului Chi-Squared

- În interiorul funcţiei ***main*** se apelează funcţia ***chiSquared*** aflată în fişierul-antet ***chisquared.h*** cu calea fişierului ca parametru.
- În interiorul funcţiei ***chiSquared*** se încarcă în tabloul ***pixelArray*** imaginea aflată la calea transmisă ca parametru, sub formă liniarizată. Dacă în cadrul încărcării au apărut erori (ce vor fi detaliate în cadrul funcţiei de încărcare), funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.
- Se alocă tablourile ***redFreq***, ***blueFreq***, ***greenFreq*** ce vor stoca frecvenţa valorilor pixelilor din ***pixelArray***, pe canalul corespunzător. În cazul în care alocarea nu s-a putut efectua, funcţia afişează un mesaj sugestiv şi returnează valoarea 1.
- Se setează fiecare câmp din cele trei tablouri cu valoarea 0, ca iniţializare.
- Se parcurge tabloul ***pixelArray*** şi se actualizează frecventa valorilor de pe fiecare canal.
- Se calculează media presupusă pe fiecare canal (***pixelNo*** / 256).
- Se parcurge fiecare tablou de frecvenţă şi se calculează media reală a valorilor.
- Se afişează pe ecran cele 3 valori corespunzătoare mediei canalului roşu, canalului verde şi canalului albastru, cu o trunchiere la a 2-a zecimală.
- Se returnează valoarea 0, cu semnificaţia că funcţia nu a întâmpinat o eroare pe parcurs.
- La întoarcerea în funcţia ***main***, se verifica valoarea returnata de ***chiSquared*****,** afişându-se un mesaj sugestiv în funcţie de aceasta.

Funcţii şi structuri folosite

- Fişierul-antet ***encryptions.h***:
  - ***int encryptBMP(const char* originalBMPPath, const char* encryptedBMPPath, const char* keyFilePath)***
    - Funcţia rulează algoritmul de criptare pe imaginea de la adresa specificată.
    - Primeşte ca parametrii căile către fişierul ce trebuie criptat, fişierul criptat ce urmează a fi creat şi fişierul ce conţine cheie secretă.
    - Conţinutul funcţiei este explicat în detaliu în cadrul paginii de criptare.
    - Returnează valoarea 0 în cazul în care criptarea s-a efectuat fără a întâmpina erori. În caz contrar funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.

  - ***int decryptBMP(const char* encryptedBMPPath, const char* decryptedBMPPath, const char* keyFilePath)***
    - Funcţia rulează algoritmul de decriptare pe imaginea de la adresa specificată.
    - Primeşte ca parametrii căile către fişierul ce trebuie decriptat, fişierul decriptat ce urmează a fi creat şi fişierul ce conţine cheie secretă
    - Conţinutul funcţiei este explicat în detaliu în cadrul paginii de criptare.
    - Returnează valoarea 0 în cazul în care decriptarea s-a efectuat fără a întâmpina erori. În caz contrar funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.

- Fişierul-antet ***chisquared.h***:
  - ***int chiSquared(const char* testBMPPath)***
    - Funcţia rulează testul Chi-Squared pe imaginea aflată la adresa specificată.
    - Primeşte ca parametru calea către fişierul asupra căruia trebuie efectuat testul Chi-Squared.
    - Conţinutul funcţiei este explicat în detaliu în cadrul paginii de rulare a testului Chi-Squared.
    - Returnează valoarea 0 în cazul în care testul a fost rulat fără a întâmpina erori. În caz contrar funcţia va afişa un mesaj sugestiv şi apoi va returna valoarea 1.

- Fişierul-antet ***bitmap.h***:
  - ***typedef struct {unsigned char B, G, R;}PIXEL;***
    - Defineşte tipul de date PIXEL, ce va fi folosit pe tot parcursul programului pentru a reţine pixelii din imaginile în format BMP utilizate.

  - ***PIXEL pixelXORpixel(PIXEL pixel1, PIXEL pixel2)***
    - Realizează operaţia XOR între doi pixeli la nivel de canale de culoare (***pixel1.B^pixel2.B, pixel1.G^pixel2.G, pixel1.R^pixel2.R***).
    - Returnează pixelul rezultat în urma operaţiei XOR.

- ***PIXEL pixelXORu_int(PIXEL pixel, unsigned int u_int)***
  - Converteşte (prin ***cast***) numărul întreg la tipul PIXEL, prin vizualizarea adresei întregului (cei 4 octeţi 0,1,2,3) ca o adresă a unui pixel (vor fi vizibili doar octeţii 0,1,2, ale căror valori vor fi utilizate pentru canalele B,G,R ale unui pixel).
  - Se apelează funcţia ***pixelXORpixel*** pentru pixelul transmis ca parametru şi valoarea obţinută prin metoda de mai sus.

- ***int loadBMPLiniar(const char* loadPath, unsigned char** header, unsigned int* pixelNo, PIXEL** pixelArray)***
  - Funcţia încarcă în memoria internă imaginea de la adresa specificată.
  - Se deschide fişierul aflat la adresa ***loadPath*** prentru citire; în cazul unei erori la deschidere se afişează un mesaj sugestiv şi se returnează valoarea 1.
  - Se obţine extensia fişierului ***loadPath*** şi se verifică dacă este în format BMP, în caz contrar se afişează un mesaj sugestiv şi se returnează valoarea 1.
  - Se alocă spaţiu pentru tabloul ****header*** transmis ca parametru, în cazul în care nu se poate aloca memorie funcţia afişând un mesaj sugestiv şi întorcând valoarea 1. Se copiază în tablou octeţii corespunzători header-ului imaginii în format BMP.
  - Se obţin dimensiunile (***width, height***) imaginii pentru a se calcula padding-ul imaginii şi respectiv numărul de pixeli (parametru ****pixelNo***).
  - Se alocă tabloul ****pixelArray***, transmis prin parametru, ce va conţine pixelii imaginii în format BMP. În cazul în care alocarea nu s-a putut efectua, funcţia afişează un mesaj sugestiv şi returnează valoarea 1.
  - Se parcurge imaginea pixel cu pixel şi se copiază pixelul curent în tabloul ****pixelArray***, pe poziţia corespunzătoare - se ţine cont de ordinea inversată a liniilor.
  - La final se returnează valoarea 0, cu semnificaţia că nu a apărut nicio eroare în cadrul funcţiei de încărcare.

- ***int saveBMPLiniar(const char* savePath, unsigned char* header, unsigned int pixelNo, PIXEL* pixelArray)***
  - Funcţia salvează în memoria externă imaginea de la adresa specificată.
  - Se obţine extensia fişierului ***savePath*** şi se verifică dacă este în format BMP, în caz contrar se afişează un mesaj sugestiv şi se returnează valoarea 1.
  - Se deschide fişierul aflat la adresa ***savePath*** pentru scriere; în cazul unei erori la deschidere se afişează un mesaj sugestiv şi se returnează valoarea 1.
  - Se scrie în imagine conţinutul tabloului ***header*** transmis ca parametru. Acesta va reprezenta header-ul imaginii noi.
  - Se obţin dimensiunile (***width, height***) imaginii pentru a se calcula padding-ul imaginii.
  - Se salvează în imagine fiecare pixel din tabloul ***pixelArray*** la poziţia corespunzătoare, ţinându-se cont de ordinea inversată a liniilor, la finalul fiecărei linii adăugându-se octeţi de padding.
  - La final se returnează valoarea 0, cu semnificaţia că nu a apărut nicio eroare în cadrul funcţiei de salvare.

Pattern-matching într-o imagine în format BMP

- În interiorul funcţiei ***main*** sunt alocate şi citite tablouri ce vor reprezenta căile către fişierele BMP folosite/ ce urmează sa fie create. În cazul în care nu se poate efectua alocarea, programul se termină prematur.
- După ce au fost alocate şi citite fiecare cale de acces, se apelează funcţia ***startMatch*** aflată în fişierul-antet ***patterns.h***.
- În interiorul funcţiei ***startMatch*** se citeşte valoarea pentru ***treshold***, reprezentând pragul folosit în cadrul funcţiei de corelaţie. Dacă valoarea nu s-a putut citi, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
- Se citeşte apoi numărul de şabloane pe care se va rula algoritmul de matching (***patternNo***), având loc o verificare a citirii similară cu cea precizată anterior.
- Se alocă apoi un tablou de şabloane (***char **pattern*** - reprezentând de fapt un tablou cu căile de acces spre fiecare şablon). În cazul în care alocarea nu s-a putut efectua, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
- Se alocă şi se citesc căile de acces către şabloane. În cazul în care alocarea nu s-a putut efectua, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
- Se alocă şi se iniţializează un tablou de culori (folosind structura PIXEL) corespunzător culorilor specificate în enunţ. În cazul în care alocarea nu s-a putut efectua, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
- Se alocă un tablou ce va reţine calea către conversia grayscale a imaginii folosite ca şi planşă (***workplacePath - workplacePath_grayscale***). În cazul în care alocarea nu s-a putut efectua, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1. Se realizează calea pentru imaginea grayscale folosind numele imaginii originale plus sufixul ”_g”.
- Se apelează funcţia ***convertGrayscale*** aflată în fişierul-antet ***bitmap.h***, creându-se imaginea grayscale în memoria externă.
- Se declară tabloul ***detection*** ce va reţine chenarele a căror corelaţie cu imaginea planşă este mai mare sau egală cu ***treshold***.
- Se apelează funcţia ***match*** din fişierul-antet curent pentru fiecare şablon în parte. În cazul în care se întâmpină o eroare în cadrul funcţiei (se returnează valoarea 1), funcţia va afişa un mesaj sugestiv şi va returna valoarea 1
- În interiorul funcţiei ***match*** se încarcă în memoria internă (se foloseşte funcţia ***loadBMP*** aflată în fişierul-antet ***bitmap.h***) imaginea planşă şi şablonul, cel din urmă fiind mai întâi transformat în grayscale, urmând să se gliseze şablonul peste planşă şi să se calculeze corelaţia între cele două folosind funcţia ***getCorrelation*** din fişierul-antet curent. Dacă valoarea este mai mare sau egală cu ***treshold***, detecţia se adaugă în tabloul ***detection***, ce se alocă la fiecare pas cu memorie suficientă memorării. În cazul în care cele două imagini nu pot fi încărcate în memoria internă, nu se poate realiza transformarea grayscale a şablonului sau alocarea tabloului nu se poate efectua, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
- La întoarcerea în funcţia ***startMatch***, se apelează funcţia ***deleteNonMax*** aflată în fişierul-antet curent, reprezentând algoritmul de eliminare al non-maximelor.
- În funcţia ***deleteNonMax***, se sortează tabloul ***detection*** folosind funcţia ***qsort*** aflată în fişierul-antet ***stdlib.h***, cu argument funcţia ***compareDetections*** aflată în fişierul-antet curent.

- Se parcurge apoi tabloul ***detection***, urmând ca fiecare element să se compare cu cele următoare din punct de vedere al suprapunerii, folosind funcţia ***isCovering*** aflată în fişierul-antet curent. În cazul unei suprapuneri, se elimină elementul detectat şi se realocă tabloul ***detection*** cu dimensiunea scăzuta cu o unitate. În cazul în care realocarea nu s-a putut efectua, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
- La întoarcerea în funcţia ***startMatch*** se verifică dacă a fost întâmpinată o eroare în cadrul algoritmului de eliminare; în cazul în care s-a întâmpinat o eroare, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
- Se încarcă în memoria internă imaginea originală (planşa înainte de grayscale). În cazul în care nu s-a putut efectua încărcarea, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
- Se parcurge tabloul ***detection*** urmând să se coloreze fiecare detecţie cu un chenar în cadrul imaginii originale, folosind funcţia ***colorDetection*** aflată în fişierul-antet curent.
- Se salvează în memoria externă imaginea cu chenarele, folosind funcţia ***saveBMP*** aflată în fişierul-antet ***bitmap.h***. În cazul în care a apărut o eroare la salvare, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
- Se returnează valoarea 0, cu semnificaţia că funcţia nu a întâmpinat o eroare pe parcurs.
- La întoarcerea în funcţia ***main***, se verifica valoarea returnata de ***startMatch*****,** afişându-se un mesaj sugestiv în funcţie de aceasta.

Funcţii şi structuri folosite

- Fişierul-antet ***patterns.h***:
  - ***typedef struct {***
    struct {
    unsigned int line, collumn;
    }top_left, bottom_right;

    unsigned int area;
    double correlation;
    PIXEL color;
    }DETECTION;
    - Defineşte tipul de date DETECTION, ce va fi folosit pe tot parcursul programului pentru a reţine detecţiile furnizate de algoritmul de pattern-matching.
    - O detecţie constă din colţul stânga-sus, respectiv dreapta-jos al chenarului, dimensiunea acestuia, valoarea corelaţiei dintre şablon şi chenarul din imaginea planşă, culoarea cu care "va fi" colorat.

  - ***int compareDetections(const void* d1, const void* d2)***
    - Funcţie folosită în cadrul funcţiei ***qsort***.
    - Decide, în funcţie de valorile corelaţiilor, dacă prima detecţie trebuie să se situeze înaintea (returnează -1), după (returnează 1) sau aleator (returnează 0) faţă de a doua detecţie.

  - ***double isCovering(DETECTION d1, DETECTION d2)***
    - Funcţia calculează gradul de suprapunere dintre cele două detecţii, analizând intersecţia dintre cele două chenare.
    - Se calculează colţul din stânga-sus al intersecţiei ca fiind linia maximă a colţurilor stânga-sus ale celor două detecţii, respectiv coloana maximă a colţurilor stânga-sus ale celor două detecţii.
    - Se calculează colţul din dreapta-jos al intersecţiei ca fiind linia minimă a colţurilor dreapta-jos ale celor două detecţii, respectiv coloana minimă a colţurilor dreapta-jos ale celor două detecţii.
    - În cazul în care intersecţia are colţul stânga-sus mai jos sau mai la dreapta decât coltul dreapta-jos, se returnează valoarea 0, cele doua detecţii nefiind într-o situaţie de suprapunere.
    - Se calculează aria intersecţiei.
    - Se returnează valoarea raportului dintre aria intersecţiei şi suma dintre ariile celor două detecţii din care se scade aria intersecţiei. Matematic, numitorul reprezinta aria reuniunii celor două detecţii.

- ***int deleteNonMax(DETECTION** detection, unsigned int* detectionNo, double coverPercentage)***
  - Se sortează tabloul ***detecion*** descrescator în funcţie de corelaţie, apelând funcţia ***qsort*** cu funcţia ***compareDetections*** ca şi argument.
  - Se parcurge apoi tabloul ***detection***, urmând ca fiecare element să se compare cu cele următoare din punct de vedere al suprapunerii, folosind funcţia ***isCovering***.
  - În cazul unei suprapuneri, se elimină elementul detectat şi se realocă tabloul ***detection*** cu dimensiunea scăzuta cu o unitate. În cazul în care realocarea nu s-a putut efectua, funcţia va afişa un mesaj sugestiv şi va returna valoarea 1.
  - La final se returnează valoarea 0, cu semnificaţia că funcţia nu a întâmpinat o eroare pe parcurs.

- ***int startMatch(const char* workplacePath, const char* resultPath, FILE* input)***
  - Funcţia iniţializează algoritmul de pattern-matching şi reprezintă funcţia de bază a acestuia.
  - Conţinutul funcţiei este explicat în detaliu în cadrul paginii de pattern-matching.

- ***int match(const char* workplacePath_grayscale, const char* pattern, double threshold, unsigned int* detectionNo, DETECTION** detection, PIXEL color)***
  - Funcţia realizează algoritmul de pattern-matching între şablonul ***pattern*** şi planşa ***worplacePath_grayscale***.
  - Conţinutul funcţiei este explicat în detaliu în cadrul paginii de pattern-matching.
  - În timpul suprapunerii celor două imagini, detecţia curentă are ca linie are ca şi colţ stânga-sus poziţia curentă în planşă, iar colţul dreapta jos este poziţia relativă a colţului dreapta-jos al şablonului în planşă. Culoarea detecţiei primeşte valoarea ***color***, specifică şablonului curent.
  - Parcurgerea matricei se execută din colţul stânga-sus al planşei până în poziţia relativă a colţului stânga-sus al şablonului în interiorul planşei, în situaţia în care colţurile dreapta-jos coincid.
  - La finalul funcţiei au loc dealocările tablourilor folosite.

- ***void getCorrelation(PIXEL** workplacePixel, PIXEL** patternPixel, DETECTION* outline)***
  - Funcţia calculează corelaţia dintre planşă şi şablon, în poziţiile descrise în ***outline***, şi o va returna prin intermediul câmpului ***correlation*** al detecţiei.
  - Se calculează numărul de pixeli din chenar, folosind datele din ***outline***. Acesta este în cadrul proiectului constant şi egal cu 165 (11x15).
  - Se calculează media intensităţilor pe planşă, respectiv pe şablon.
  - Se calculează deviaţia standard a intensităţii pixelilor pentru planşă, respectiv pentru şablon.
  - Se aplică formula descrisă în enunţul proiectului.
- ***void colorDetection(PIXEL** workplacePixel, DETECTION outline)***
  - Funcţia colorează în planşă chenarul descris de câmpurile detecţiei ***outline***.
  - Se parcurg marginile chenarului şi se colorează cu culoarea specificată în câmpul ***color*** al detecţiei.

- Fişierul-antet ***bitmap.h***:
  - ***int loadBMP(const char* loadPath, unsigned char** header, unsigned int* width, unsigned int* height, PIXEL*** pixelMap)***
    - Funcţia încarcă în memoria internă imaginea de la adresa specificată.
    - Se deschide fişierul aflat la adresa ***loadPath*** prentru citire; în cazul unei erori la deschidere se afişează un mesaj sugestiv şi se returnează valoarea 1.
    - Se obţine extensia fişierului ***loadPath*** şi se verifică dacă este în format BMP, în caz contrar se afişează un mesaj sugestiv şi se returnează valoarea 1.
    - Se alocă spaţiu pentru tabloul ****header*** transmis ca parametru, în cazul în care nu se poate aloca memorie funcţia afişând un mesaj sugestiv şi întorcând valoarea 1. Se copiază în tablou octeţii corespunzători header-ului imaginii în format BMP.
    - Se obţin dimensiunile (***width, height***) imaginii pentru a se calcula padding-ul imaginii şi respectiv numărul de pixeli (parametru ****pixelNo***).
    - Se alocă tabloul ****pixelMap***, transmis prin parametru, ce va conţine adrese către fiecare linie a imaginii în format BMP. În cazul în care alocarea nu s-a putut efectua, funcţia afişează un mesaj sugestiv şi returnează valoarea 1.
    - Se alocă fiecare adresă din cadrul ****pixelMap*** ca şi tablou de pixeli, urmând să conţină pixelii aflaţi pe linia curentă.
    - Se parcurge imaginea pixel cu pixel şi se copiază pixelul curent în tabloul ****pixelMap***, pe poziţia corespunzătoare - se ţine cont de ordinea inversată a liniilor.
    - La final se returnează valoarea 0, cu semnificaţia că nu a apărut nicio eroare în cadrul funcţiei de încărcare.

- ***int saveBMP(const char* savePath, unsigned char* header, unsigned int width, unsigned int height, PIXEL** pixelMap)***
  - Funcţia salvează în memoria externă imaginea de la adresa specificată.
  - Se obţine extensia fişierului ***savePath*** şi se verifică dacă este în format BMP, în caz contrar se afişează un mesaj sugestiv şi se returnează valoarea 1.
  - Se deschide fişierul aflat la adresa ***savePath*** pentru scriere; în cazul unei erori la deschidere se afişează un mesaj sugestiv şi se returnează valoarea 1.
  - Se scrie în imagine conţinutul tabloului ***header*** transmis ca parametru. Acesta va reprezenta header-ul imaginii noi.
  - Se obţin dimensiunile (***width, height***) imaginii pentru a se calcula padding-ul imaginii.
  - Se salvează în imagine fiecare pixel din tabloul ***pixelMap*** la poziţia corespunzătoare, ţinându-se cont de ordinea inversată a liniilor, la finalul fiecărei linii adăugându-se octeţi de padding.
  - La final se returnează valoarea 0, cu semnificaţia că nu a apărut nicio eroare în cadrul funcţiei de salvare.

- ***int convertGrayscale(const char* originalPath, const char* resultPath)***
  - Funcţia creează o un fişier/ suprascrie fişierul aflat la adresa ***resultPath*** în care copiază imaginea aflată la adresa ***originalPath*** şi apoi aplică o conversie grayscale asupra acesteia.
  - În cazul în care fişierul de la adresa ***originalPath*** nu este în format BMP, se afişează un mesaj sugestiv şi funcţia returnează valoarea 1.
  - În cazul în care fişierul de la adresa ***originalPath*** nu se poate deschide, se afişează un mesaj sugestiv şi funcţia returnează valoarea 1.
  - În cazul în care fişierul de la adresa ***resultPath*** nu este în format BMP, se afişează un mesaj sugestiv şi funcţia returnează valoarea 1.
  - În cazul în care fişierul de la adresa ***resultPath*** nu se poate deschide, se afişează un mesaj sugestiv şi funcţia returnează valoarea 1.
  - Se parcurge imaginea pixel cu pixel, calculându-se intensitatea grayscale pentru pixelul curent.
  - Se înlocuiesc valorile de pe fiecare canal al pixelului curent cu intensitatea grayscale calculată.
  - La final se returnează valoarea 0, cu semnificaţia că nu a apărut nicio eroare în cadrul funcţiei de conversie.