Documentaţie

Gălăţan Alexandru-Cristian

Grupa 131

`unsigned int* XORSHIFT32(unsigned int seed, unsigned int n)`

Generează **n** numere la întâmplare mutând biţii numărului precedent, pornind de la **seed**.

`unsigned int* generatePermutation(unsigned int *R, unsigned int WH)`

Generează o permutare de dimensiunea **WH**, folosind numerele ce au fost generate prin **XORSHIFT** şi se află în **R**.

`void scrambleImage(image **imageToScramble, unsigned int *P)`

Permută pixelii unei imagini stocate intern şi liniarizate folosind o permutare **P**.

`void unScrambleImage(image **imageToScramble, unsigned int *P)`

Permută invers pixelii unei imagini stocate intern şi liniarizate folosind o permutare **P**.

`void grayscaleImage(image **imageToGrayscale)`

Transformă o imagine color stocată intern în imagine grayscale.

`void XORPX(pixel *PIX, unsigned int X)`

Execută operaţia sau-exclusiv (**XOR**) între pixelul **PIX** şi numărul **X**.

`void XORPP(pixel *P0, pixel *P1, pixel *P2)`

Execută operaţia sau-exclusiv (**XOR**) între pixelii **P1** şi **P2** şi salvează rezultatul în **P0**.

`void substitute(image **imageToSubstitute, unsigned int SV, unsigned int *R)`

Execută operaţii de substituţie asupra fiecărui pixel dintr-o imagine stocată intern.

`void invertedSubstitution(image **imageToSubstitute, unsigned int SV, unsigned int *R)`

Realizează inversa substituţiei, pentru a reveni la valorile iniţiale ale unei imagini criptate.

`void encryption(image **img, unsigned int key, unsigned int SV)`

Criptează integral o imagine **img** stocată intern, folosind o cheie, **key**, şi o valoare ce este folosită la prima operaţie sau-exclusiv (**XOR**), **SV**.

`void decryption(image **img, unsigned int key, unsigned int SV)`

Decriptează integral o imagine img stocată intern, folosind o cheie, key, şi o valoare ce este folosită la prima operaţie sau-exclusiv (**XOR**), **SV**.

`unsigned int getImageDetails(image** imageFile)`

Citeşte informaţiile din header şi le salvează în structura salvată intern. De asemenea, detectează ce padding a fost folosit.

`unsigned int liniarizeImage(image **imageFile)`

Citeşte pixelii unei imagini stocate extern şi îi salvează intern, în mod liniarizat.

`image* readImage(char *imagePath)`

Uneşte toate operaţiile necesare pentru a salva în mod integral o imagine în memoria internă.

`void outputImage(image* IMAGE, unsigned char *imagePath)`

Salvează extern o imagine stocată intern.

`void readString(unsigned char **readTo, char* showString)`

Afişează pe ecran mesajul **showString**, după care salvează informaţia transmisă de la tastatură în **readTo**.

`void chiTest(unsigned int *a, double f, char c)`

Calculează şi afişează rezultatul testului chi pe un anumit canal de culoare.

`void chi2(image *IMAGE)`

Contorizează fiecare apariţie a fiecărei culori în imagine, după care afişează testul chi pentru fiecare canal de culoare.

`pixel* S(image* IMG, int x, int y)`

Returnează pixelul ce se află pe poziţia (x, y) în imaginea stocată intern, **IMG**.

`float calculateCorelation(image *FI, image* TEMPLATE)`

Calculează corelaţia dintre două imagini stocate intern de dimensiuni identice.

`void copyImage(image* source, image* target, int x, int y)`

Copiază pixelii din imaginea stocată intern, **source**, în imaginea stocată intern, **target**.

`void drawRectangle(image *TARGET, detection *DETECTION)`

Desenează un dreptunghi în imaginea stocată intern, **TARGET**, folosind informaţiile ce se află în detecţia **DETECTION**.

`detection* detectNumber(image* IMG, image* TEMPLATE, float PS, unsigned char R, unsigned char G, unsigned char B)`

Generează un vector de detecţii a unei imagini stocate intern, **TEMPLATE**, într-o altă imagine stocată intern, **IMG**. Detecţiile găsite trebuie să treacă de pragul **PS**, iar dreptunghiurile generate de acestea la afişare vor avea culoarea în format (**R**, **G**, **B**).

`void freeImage(image **img)`

Eliberează spaţiul ocupat de o imagine stocată intern.

`int compareDetections (const void * a, const void * b)`

Compară corelaţia dintre două detecţii pentru a sorta vectorul folosind **qsort.**

`int overlap(detection *a, detection *b)`

Returnează **0** dacă cele două detecţii nu se intersectează şi **1** în cazul contrar.

`int min(int a, int b)`

Returnează numărul mai mic dintre **a** şi **b**.

`int max(int a, int b)`

Returnează numărul mai mare dintre **a** şi **b**.

`int areaOfIntersection(detection *a, detection *b)`

Calculează aria de intersecţie dintre două detecţii.

`int getDetectionArea(detection *a)`

Returnează aria unei detecţii.

`void eliminateNonMaxs(detection* detectionArray)`

Realizează operaţia de eliminare a non-maximelor din vectorul de detecţii **detectionArray**.

`detection* getDetectionArray(image **img)`

Generează şi returnează un vector de detecţii ale tuturor cifrelor în imaginea stocată intern, **img**.

`void templateMatchingProgram()`

Uneşte toate funcţiile necesare pentru recunoaşterea cifrelor într-o imagine.

`void cryptProgram()`

Uneşte toate funcţiile necesare pentru a cripta o imagine.

`void decryptProgram()`

Uneşte toate funcţiile necesare pentru a decripta o imagine.

`int main()`

Programul principal, prezintă un meniu utilizatorului ce permite executarea a celor 3 operaţii, în mod repetat: **criptarea unei imagini**, **decriptarea unei imagini**, **recunoaşterea cifrelor într-o imagine**.