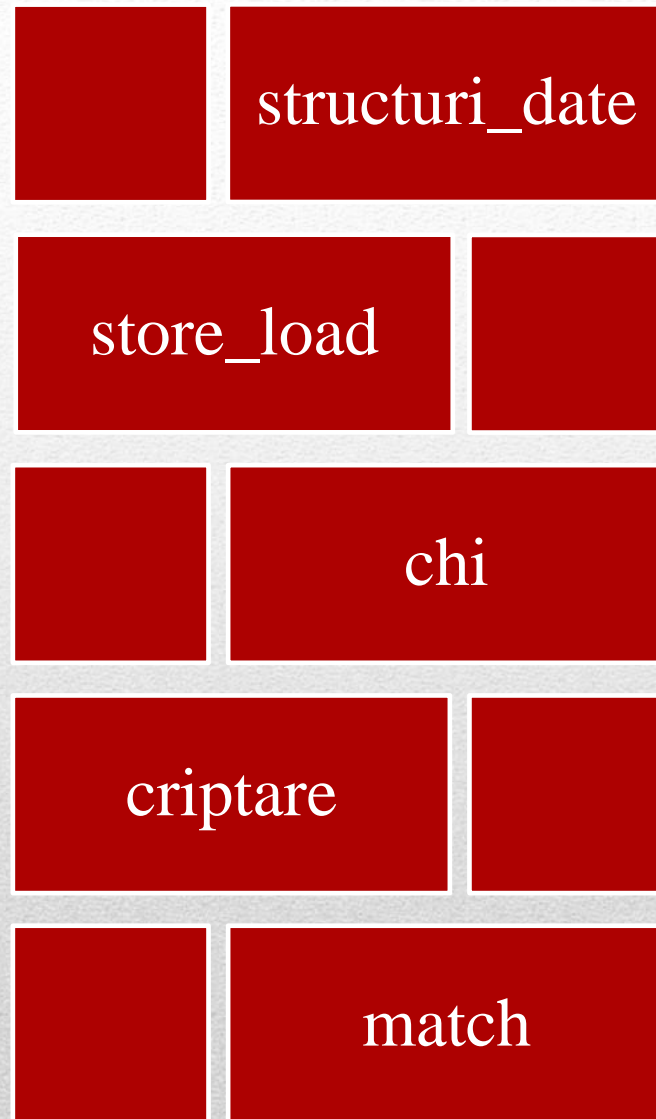




Proiect PP








Criptare si decriptarea unei imagini +Patter matching
– M. Daniel

Structura proiectului








Directoare

- cerinte
 - Contine un director care contine rezolvarea unuia dintre programele cerute
- headers
 - Fisiere surse ce contin functile folosite pentru rezolvarea cerintelor
- imagini_match
 - Imagini bmp folosite in procesul de pattern matching
- test-match + test-crypt
 - Fisiere pentru stocarea testelor
- cai_imagini_sabloane + secret
 - Fisiere text ce retin caile sabloanelor + cheia secreta

	cerinte	12/26/2018 2:24 PM	File folder	
	headers	12/1/2018 3:27 PM	File folder	
	imagini_match	12/2/2018 2:28 PM	File folder	
	test-match	12/26/2018 2:28 PM	File folder	
	test-crypt	12/26/2018 1:42 PM	File folder	
	cai_imagini_sabloane	12/8/2018 3:48 PM	Text Document	1 KB
	secret	11/30/2018 4:41 PM	Text Document	1 KB

headers

- Chi
 - Functiile pentru testu chi patrat
- Criptare
 - Functii pentru criptarea unei imagini
- Match
 - Functii pentru patter matching
- Store_load
 - Citire + scriere
- Structuri_Date
 - Definire tipuri de date

 chi	12/23/2018 3:57 PM	C source file	2 KB
 criptare	12/23/2018 4:07 PM	C source file	5 KB
 match	12/23/2018 3:44 PM	C source file	6 KB
 store_load	12/26/2018 2:36 PM	Header file	2 KB
 structuri_date	12/26/2018 2:37 PM	Header file	1 KB

structuri_date

```
typedef struct {
    int cifra, x, y;
    double scor;
} FEREASTRA;

union bytes{
    unsigned char b[4];
    unsigned int x;
};

typedef struct {
    unsigned char B, G, R;
} PIXEL ;

typedef struct {
    unsigned char Header[ 54 ];
    int n, m, pad;
    PIXEL *Img;
} IMAGE;
```

- Fereastră
 - Coltul stanga sus
 - Scorul inregistrat pentru detectie
- Bytes
 - Folosita pentru acces octeti
- Pixel
 - Codeaza un pixel prin RGB
- IMAGE
 - Header constant de dim 54
 - Dimesniunile imaginii (latime + lungime + padding)
 - Vector alocat dinamic pentru a memora pixelii

store_load

```
▢ IMAGINE LoadImg( char *path ) {
```

- Incarca in memoria interna o imagine in format BMP
 - Primeste ca parametru calea imaginii
 - Returneaza imaginea citita (de tip IMAGINE)
 - Aloca dinamic un vector de pixeli

```
▢ int StoreImg( IMAGINE Img, char *path ) {
```

- Stocheaza in memoria externa o imagine (de tip IMAGINE)
 - Parametrii : imaginea de stocat + calea de stocare
 - Returneaza 1 pentru stocare reusita, 0 altfel
 - Nu elimina imaginea din memorie
-

chi

```
void ChiPatrat( IMAGINE imag ) {
    int k, i, j, d, dim;
    dim = imag.n * imag.m;
    float xPatrat, valMedie;

    valMedie = 1.0*imag.n*imag.m/256;
    xPatrat = 0;

    for ( k = 0; k < 3; ++k ) {
        xPatrat = 0;
        for ( i = 0; i < 256; ++i ) {
            d = 0;
            for ( j = 0; j < dim; ++j ) {
                if ( *(&imag.Img[ j ].B+k) == i ) {
                    d++;
                }
            }
            xPatrat += ( d - valMedie)*( d - valMedie);
        }
        xPatrat /= valMedie;
        switch(k){
            case 0 : printf("B : "); break;
            case 1 : printf("G : "); break;
            case 2 : printf("R : "); break;
        }
        printf("%.2f | ", xPatrat);
    }
    printf("\n");
}
```

- Calculeaza valoarea testului chi patrat
- Parametru : imaginea pentru care se ruleaza testul
- Afiseaza valoarea testului pe fiecare canal de culoare (2 zecimale)
- Implementeaza urmatoare formula
 - f_i = frecventa lui I
 - \bar{f} = valoarea medie

$$\chi^2 = \sum_{i=0}^{255} \frac{(f_i - \bar{f})^2}{\bar{f}}$$

criptare

- Swap
 - Interschimba 2 valori de tip *unsigned int*
- XorShift32
 - Primesete calea catre un fisier secret ce contine o cheie dupa care se genereaza cu metoda XorShift 32 numerele pseudo-aleatoare + cate numere se vor genera
 - Returneaza un pointer catre o zona de memorie ce retine numere generate
- GetPermutare
 - Genereaza o pseudo-permutare pe baza numerelor aleatoare
 - Foloseste alg. Durstenfeld
 - Foloseste liniarizare modulo (numerele XorShift32 nu au un domeniu fixat)
 - Returneaza un pointer catre o zona de memorie ce retine permutarea
- SuffleImg
 - Amesteca un vector de pixeli folosind o permutare aleatoare (data de GetPermutare)

```
void swap( unsigned int *a, unsigned int*b );
unsigned int* GetPermutare( int dim, char *caleeFisier );
unsigned int* XorShift32( int nnumar_elemente, char *caleeFisier );
PIXEL* ShuffleImg( PIXEL *Img, int dim, char *caleeFisier );
IMAGINE CriptareImg( IMAGINE imag, char *caleeFisier );
IMAGINE DecriptareImg( IMAGINE imag, char *caleeFisier );
```


criptare

- CriptareImg
 - Primește o imagine (de tip IMAGE) pentru a fi criptata
 - Primește calea fisierului ce contine cheia secreta
 - Criptarea se realizeaza dupa algoritmul descris (Vezi fisieurl cu cerinta proiectului)
 - Returneaza imaginea criptata
- DecriptareImg
 - Primește o imagine (de tip IMAGE) pentru a fi decriptata
 - Primește calea fisierului ce contine cheia secreta
 - Decriptarea se realizeaza dupa algoritmul descris (Vezi fisieurl cu cerinta proiectului)
 - Returneaza imaginea decriptata

```
void swap( unsigned int *a, unsigned int*b );  
unsigned int* GetPermutare( int dim, char *caleeFisier );  
unsigned int* XorShift32( int nnumar_elemente, char *caleeFisier );  
PIXEL* ShuffleImg( PIXEL *Img, int dim, char *caleeFisier );  
IMAGE CriptareImg( IMAGE imag, char *caleeFisier );  
IMAGE DecriptareImg( IMAGE imag, char *caleeFisier );
```

match

- TemplateMatching
 - Intrare : Sablon + Imagine (ambele sunt IMAGINE) + prag detectie
 - Iesire : Vector de detectii de ferestre + dimensiunea vectorului
- getGreyValue
 - Valoarea Greyscale a unui pixel RGB
- Cmp
 - Functie comparator folosita la sortarea vectorului de detectie

Pentru mai multe detalii vezi cod

```
void TemplateMatching( IMAGINE S, IMAGINE I, double pS, FEREASTRA **detectie, int *dim );
unsigned char getGreyValue( PIXEL x );
double Suprapunere( FEREASTRA a, FEREASTRA b, int n, int m, int sn, int sm );
int cmp( const void *a, const void *b ) {
```

Example

```
0) Ieire
1) Criptare imagine
2) Decriptare imagine
3) Testul X^2
--> Introdu optiunea :
-->1
--->Calea imaginii: ../../test-crypt/peppers.bmp
--->Calea stocare: ../../test-crypt/peppers_enc.bmp
--->Calea secret: ../../secret.txt
0) Ieire
1) Criptare imagine
2) Decriptare imagine
3) Testul X^2
--> Introdu optiunea :
-->2
--->Calea imaginii: ../../test-crypt/peppers_enc.bmp
--->Calea stocare: ../../test-crypt/peppers_enc_dec.bmp
--->Calea secret: ../../secret.txt
0) Ieire
1) Criptare imagine
2) Decriptare imagine
3) Testul X^2
--> Introdu optiunea :
-->3
Cale imagine pentru chi:../../test-crypt/peppers_enc.bmp
B : 226.36 | G : 250.71 | R : 286.68 |
```

Cerinta de criptare a fost implemenata sub forma unui meniu ce permite operatile cerute

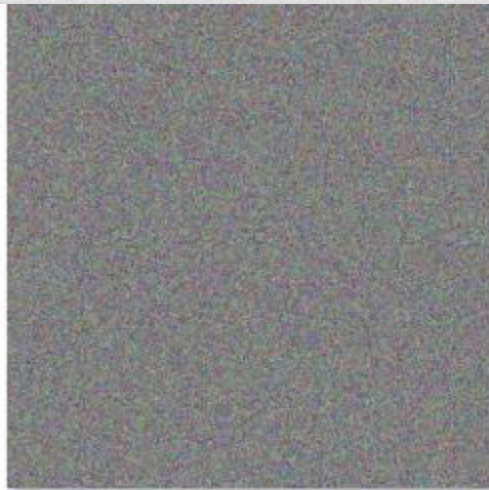
Exemplu de rulare a programului de criptare decriptare in stanga

Exemple

Rezultatul rularii anterioare:



peppers



peppers_enc



peppers_enc_dec

Exemple

Exemplu de rulare pentru pattern matchinga



test



test_match

```
0) Ieire
1) Criptare imagine
2) Decriptare imagine
3) Testul X^2
--> Introdu optiunea :
-->0
Cale imaginini fisier sabloane: ../../cai_imagini_sabloane.txt
Cale imagine de procesat: ../../test-match/test.bmp
Cale imagine de salvat: ../../test-match/test_match.bmp
```