

## Tema 1 - Image processor

### Changelog:

- 13 Dec 2023: deadline-ul a fost amanat pentru data de 17 soft si 20 hard
- 12 Dec 2023: update checker - toate imaginile sunt acum semnificativ mai mici
- 30 Nov 2023: update checker sa foloseasca pentru Valgrind inputurile 3-12 in loc de 5-14 pentru ca erau imagini prea mari
- 28 Nov 2023: update schelet ca sa nu dea mai dea erori de cplint pe bmp.h si imageprocessing.h

### Responsabili:



- Alin Popa

### Termen de predare:

- Deadline soft: **Duminica 17 Decembrie 2023, 23:55**
- Deadline hard: **Miercuri 20 Decembrie 2023, 23:55**

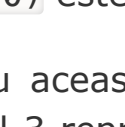
Pentru fiecare zi (24 de ore) de intarziere, se vor scadea 10 puncte din nota acordată, până la atingerea deadline-ului 17h10.

### Întrebări

Dacă aveți nelămuriri, puteți să ne contactați pe forumul dedicat  temei de casă nr. 1 sau pe  canalul Tema 1.

La orice întrebare vom răspunde în maxim 24 de ore.

Nu se acceptă întrebări în ultimele 24 de ore înainte de deadline.



Atenție:

- Citiți cu atenție tot enunțul temei.

### Intro imagini

La această temă ne propunem să construim un program de procesare de imagini. Vom reprezenta o imagine sub forma unei matrici de pixeli NxM (N linii, M coloane), unde pentru fiecare pixel avem 3 valori: R (Red), G (Green), B (Blue) reprezentând cele 3 componente de culoare. Astfel, o imagine va fi o matrice NxMx3 (spre exemplu, m[5][100][8] este componenta Red a pixelului de pe linia 5, coloana 100 din imaginea m). Fiecare din cele 3 componente de culoare poate avea doar valori întregi între 0 și 255 inclusiv. Spre exemplu, un pixel cu valorile (0,0,0) este negru; unul cu valorile (255,255,255) este alb; unul cu valorile (255,255,0) este galben etc.

Pentru această temă vom considera că o imagine are originea coordonatelor în colțul stânga-sus. De exemplu, rândul 3 reprezintă al treilea rând de pixeli ai imaginii numărât de sus în jos; coloana 3 reprezintă a treia coloană de pixeli numărată de la stânga la dreapta. W (Width) înseamnă dimensiunea imaginii pe orizontală (numărul de coloane); H (Height) înseamnă dimensiunea imaginii pe verticală (numărul de linii). La funcțiile care primesc ca parametru o pereche de coordonate (x,y), coordonata x este pe orizontală (între 0 și M) iar coordonata y este pe verticală (între 0 și N).



Toate matricile și coordonatele se consideră indexate de la 0 (e.g. coordonata (x=0,y=1) se referă la pixelul de pe prima coloană, a doua linie a imaginii).

### Schelet de cod

Pentru această temă trebuie să porniți de la scheletul de cod de aici:  tema1\_schelet.zip.

În scheletul de cod veți găsi următoarele:

- Un Makefile cu reguli care compilează și rulează main.c și respectiv interactive.c
- Un fișier imageprocessing.c în care voi va trebui să completați implementările funcțiilor pentru taskurile 1-6
- Un fișier interactive.c în care voi va trebui să completați implementarea programului pentru taskul 7
- Un fișier main.c care **nu trebuie trimis în arhiva cu tema**, scopul lui este să vă ofere un exemplu pentru rularea funcțiilor din celelalte fișiere (ca de exemplu, funcțiile read\_from\_bmp și write\_to\_bmp)
- Un fișier bmp.c ce conține implementările funcțiilor read\_from\_bmp și write\_to\_bmp ce trebuie folosite în temă.



Atenție! **Nu** aveți voie să:

- redenumiți fișierele temei (imageprocessing.c și interactive.c)
- schimbați numărul, tipul, sau ordinea parametrilor funcțiilor din imageprocessing.c și imageprocessing.h
- schimbați sau redenumiți comenzile sau regulile din Makefile



**Notă referitoare la alocarea dinamică.** Toate funcțiile din fișierul imageprocessing.c primesc ca parametru o imagine care se presupune că a fost deja alocată (dinamic) și returnează imaginea modificată. Dacă aplicați procesările direct pe imaginea primită ca parametru, este suficient să o returnați tot pe ea. Însă dacă alocăți o nouă matrice și aplicați procesările pe matricea nou creată, imaginea originală primită ca parametru trebuie dezalocată în aceeași funcție! Atenție: pentru funcțiile de procesare care necesită schimbarea dimensiunii imaginii (e.g. crop, extend), este obligatoriu ca la finalul funcției să obținem o imagine cu exact atăta memorie alocată cât este necesar (e.g. în urma aplicării unui crop de dimensiune 100x100 pe o imagine 800x600 trebuie să obținem o matrice alocată 100x100).

### Cerință

#### Task1 (5p) - Flip Horizontal

Implementați în fișierul imageprocessing.c funcția:

```
int ***flip_horizontal(int ***image, int N, int M)
```

Funcția trebuie să returneze imaginea obținută prin oglindirea pe orizontală a imaginii primite ca parametru.

Exemplu. Dacă imaginea originală avea pixelii:

$$A = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

atunci, după oglindire, trebuie să aiibă:

$$A = \begin{bmatrix} p_{13} & p_{12} & p_{11} \\ p_{23} & p_{22} & p_{21} \\ p_{33} & p_{32} & p_{31} \end{bmatrix}$$

#### Task2 (5p) - Rotate Left

Implementați în fișierul imageprocessing.c funcția:

```
int ***rotate_left(int ***image, int N, int M)
```

Funcția trebuie să returneze imaginea obținută prin rotirea cu 90 de grade la stânga a imaginii primite ca parametru.

Exemplu. Dacă imaginea originală avea pixelii:

$$A = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

atunci, după rotire la stânga, trebuie să aiibă:

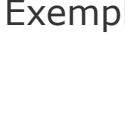
$$A = \begin{bmatrix} p_{13} & p_{23} & p_{33} \\ p_{12} & p_{22} & p_{32} \\ p_{11} & p_{21} & p_{31} \end{bmatrix}$$

#### Task3 (5p) - Crop

Implementați în fișierul imageprocessing.c funcția:

```
int ***crop(int ***image, int N, int M, int x, int y, int h, int w)
```

Funcția trebuie să returneze imaginea obținută prin crop (sub-matrice) care începe (adică colțul stânga-sus al cropului) la coordonatele (x,y), de dimensiune (h,w) (adică sub-matricea rezultată trebuie să aiibă h linii, w coloane).



Coordonata x este pe orizontală (denotă coloana) iar coordonata y este pe verticală (denotă linia).

Exemplu. Dacă imaginea originală avea pixelii:

$$A = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$$

atunci, după aplicarea unui crop cu x=2, y=1, h=2, w=2, trebuie să obținem:

$$A = \begin{bmatrix} p_{23} & p_{24} \\ p_{33} & p_{34} \end{bmatrix}$$

#### Task4 (5p) - Extend

Implementați în fișierul imageprocessing.c funcția:

```
int ***extend(int ***image, int N, int M, int rows, int cols, int new_R, int new_G, int new_B)
```

Funcția trebuie să aplice opusul operației de crop, adică să extindă imaginea cu rows linii atât deasupra cât și dedesubt, și cu cols coloane atât la stânga cât și la dreapta. Pixelii nou-creați trebuie să aiibă toți aceeași culoare, dată de parametrii new\_R, new\_G, new\_B.

Exemplu. Dacă imaginea originală avea pixelii:

$$A = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

atunci, după aplicarea unui extend cu rows=1, cols=2, imaginea devine:

$$A = \begin{bmatrix} p_0 & p_0 & p_0 & p_0 & p_0 & p_0 \\ p_0 & p_0 & p_{11} & p_{12} & p_0 & p_0 \\ p_0 & p_0 & p_{21} & p_{22} & p_0 & p_0 \\ p_0 & p_0 & p_0 & p_0 & p_0 & p_0 \end{bmatrix}$$

unde p\_(n) reprezintă pixeli cu valorile culorilor (new\_R, new\_G, new\_B).

#### Task5 (5p) - Copy Paste

Implementați în fișierul imageprocessing.c funcția:

```
int ***paste(int ***image_dst, int N_dst, int M_dst, int *** image_src, int N_src, int M_src,
```

Funcția trebuie să facă copy-paste de la imaginea sursă peste imaginea destinație, începând de la coordonatele (x,y) ale imaginii destinație. Dacă imaginea sursă este mai mare decât spațiul disponibil (adică ar depăși marginile imaginii destinație), atunci pixelii care depășesc trebuie ignorați. Atenție: imaginea sursă nu trebuie modificată și nici dezalocată în această funcție! Funcția trebuie să returneze pointer la imaginea destinație.



Coordonata x este pe orizontală (denotă coloana) iar coordonata y este pe verticală (denotă linia).

Exemplu. Dacă imaginea destinație avea pixelii:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

și imaginea sursă este:

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

atunci, după aplicarea paste la coordonatele x=1, y=1, trebuie să obținem:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & b_{11} & b_{12} \\ a_{31} & b_{21} & b_{22} \end{bmatrix}$$

#### Task6 (15p) - Apply Filter

Implementați în fișierul imageprocessing.c funcția:

```
int ***apply_filter(int ***image, int N, int M, float **filter, int filter_size)
```

Funcția trebuie să modifice imaginea prin aplicarea filtrului primit ca parametru pe toți pixelii imaginii. Un filtru este o matrice bidimensională de dimensiune (filter\_size x filter\_size) care definește modul în care fiecare pixel este modificat în funcție de valoarea sa și de valorile vecinilor lui. În practică, filtrele sunt folosite pentru tot felul de procesări pe imagini: blur, sharpen, emboss, sobel, edge detection etc.

De exemplu: fie  $(R_{22}, G_{22}, B_{22})$  un pixel din imagine care are următorii vecini:

$$\begin{bmatrix} (R_{11}, G_{11}, B_{11}) & (R_{12}, G_{12}, B_{12}) & (R_{13}, G_{13}, B_{13}) \\ (R_{21}, G_{21}, B_{21}) & (R_{22}, G_{22}, B_{22}) & (R_{23}, G_{23}, B_{23}) \\ (R_{31}, G_{31}, B_{31}) & (R_{32}, G_{32}, B_{32}) & (R_{33}, G_{33}, B_{33}) \end{bmatrix}$$

și fie filtrul de dimensiune 3 (atenție - valorile din filtru sunt numere (float), nu pixeli!):

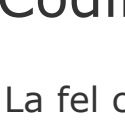
$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Atunci, în imaginea rezultată în urma aplicării filtrului A, pixelul  $(R_{22}, G_{22}, B_{22})$  va fi înlocuit cu  $(R'_{22}, G'_{22}, B'_{22})$  unde:

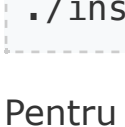
$$R'_{22} = \sum_{i=1}^3 \sum_{j=1}^3 R_{ij} f_{ij}$$

$$G'_{22} = \sum_{i=1}^3 \sum_{j=1}^3 G_{ij} f_{ij}$$

$$B'_{22} = \sum_{i=1}^3 \sum_{j=1}^3 B_{ij} f_{ij}$$



Filtrul are valori de tip float, iar după calculul sumei, valoarea trebuie transformată în înt prin rotunjire în jos (cast la int). Dacă în urma calculului uneia dintre cele 3 valori de mai sus, suma rezultată este < 0 atunci valoarea se va seta la 0. Asemănător, dacă suma este > 255 atunci componenta corespunzătoare din pixel se va seta la 255.



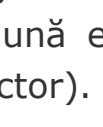
**Dacă un pixel are vecini în afara imaginii, la calculul sumei acești vecini vor fi considerați cu valoarea (0, 0, 0).**

Se garantează că filtrele primite ca parametru vor avea mereu dimensiune impară.

#### Task7 (30p) - Interactive Image Processor

Scrieți un program (în fișierul interactive.c) care execută procesări pe imagini în mod interactiv. Programul trebuie să funcționeze astfel:

- citește de la tastatură o comandă dată de utilizator
- în funcție de comanda primită, citește parametrii comenzii așa cum sunt definiți mai jos
- aplică procesările pe imagini conform cu comanda și parametrii primiți
- așteaptă următoarea comandă.



Imaginile create cu comanda **Load** trebuie să primească fiecare câte un index. Astfel, prima imagine creată va avea indexul 0; a doua imagine creată va avea indexul 1; a treia imagine creată va avea indexul 2 ș.a.m.d. Indexul este folosit în comenzile de procesare de imagini pentru a selecta imaginea pe care se execută procesarea. Similar, fiecare filtru creat va primi câte un index, care va fi apoi folosit pentru selectarea filtrului de aplicat.



Prin aplicarea oricărei procesări, trebuie modificată imaginea de la indexul respectiv, nu creată o nouă imagine. Spre exemplu, dacă imaginile încărcate sunt A (poză cu câțel) și B (poză cu pisică), atunci în urma aplicării Rotate pe imaginea cu index 1, trebuie să obținem pe indexul 0 o poză cu câțel și pe indexul 1 o poză cu pisică rotită.



La ștergerea unei imagini sau a unui filtru, imaginile/filtrele rămase trebuie să își schimbe indexul corespunzător. Exemplu: dacă am creat 3 imagini A,B,C (indexi 0,1,2) și 5 filtre Q,W,E,R,T (indexi 0,1,2,3,4) atunci dacă șterg imaginea 0 si filtrele 2 și 3, imaginile rămase trebuie să aiibă indexi B=0, C=1, iar filtrele rămase trebuie să aiibă indexi Q=0, W=1, T=2. De asemenea, **la ștergerea unei imagini sau a unui filtru, matricea asociată imaginii/filtrului trebuie dezalocată**



Atenție: Pentru operațiile **Load** (încărcarea unei imagini sub forma unei matrici NxMx3 dintr-un fișier .BMP) și **Save** (salvarea unei imagini date ca matrice într-un fișier .BMP) **trebuie** să folosiți funcțiile read\_from\_bmp și write\_to\_bmp puse la dispoziție în scheletul de cod.

Comenzile care trebuie implementate sunt următoarele:

Nume comandă	Parametri	Descriere
e	-	<b>Exit</b> - Închide programul
l	N M path	<b>Load</b> - alocă și încarcă imaginea de dimensiune NxM aflată la calea path
s	index path	<b>Save</b> - salvează imaginea de pe indexul index la calea specificată prin path
ah	index	<b>Apply Horizontal Flip</b> - aplică operația de flip pe orizontală imaginii de la indexul index
ar	index	<b>Apply Rotate</b> - aplică operația de rotație la stânga imaginii de la indexul index
ac	index x y w h	<b>Apply Crop</b> - aplică operația de crop cu parametrii dați imaginii de la indexul index
ae	index rows cols R G B	<b>Apply Extend</b> - aplică operația de extend cu parametrii dați imaginii de la indexul index
ap	index_dst index_src x y	<b>Apply Paste</b> - aplică operația de paste cu parametrii dați imaginii de la indexul index_dst
cf	size [list of values]	<b>Create filter</b> - alocă și creează un filtru de dimensiune size cu valorile date de lista de valori (exemplu: dacă se creează un filtru de dimensiune 3, atunci după size vor urma 9 valori)
af	index_img index_filter	<b>Apply filter</b> - aplică filtrul de pe indexul index_filter pe imaginea de pe indexul index_img
df	index_filter	<b>Delete filter</b> - șterge și dezalocă filtrul de pe indexul index_filter
di	index_img	<b>Delete image</b> - șterge și dezalocă imaginea de pe indexul index_img

Exemplu de utilizare (input al programului):



```
l 1 768 1024 ./cat.bmp
l 1 768 1024 ./dog.bmp
ah 0
ac 1 300 300 200 100
cf 3 0 1 0 1 1 1 0 1 0
af 1 0
ap 0 1 300 300
s 0 ./output.bmp
df 0
di 0
e
```

Se garantează că:

- path-urile nu vor avea lungime mai mare de 100 de caractere (puteți folosi o variabilă char path[100])
- path-urile nu vor se încrucișa (pot fi citite cu scanf ("%s", path))
- fișierul din care se încarcă o imagine există și este valid
- comenzile date sunt mereu valide (nu se va da ca input o comandă inexistentă)
- parametrii comenzilor sunt mereu valizi (e.g. nu se va cere aplicarea unei procesări pe un index care nu există, sau paste la niște coordonate care încep în afara imaginii destinație etc)



Atenție la următoarele lucruri:

- imaginile nu au neapărat aceleași dimensiuni, pot avea dimensiuni diferite
- filtrele nu au neapărat aceeași dimensiune, pot avea dimensiuni diferite (e.g. 1, 3, 5 etc)
- aceleși fișier .BMP poate fi folosit pentru **Load** de mai multe ori (se vor crea mai multe imagini, chiar dacă sunt identice)

#### Task8 (20p) - Clean Valgrind

Pentru acest task, trebuie să aveți punctajul maxim pe taskul 7 și să nu aveți memory leaks la verificarea folosind utilitarul  valgrind pe acea.

Utilitarul se va rula folosind următoarea comandă:

```
valgrind --tool=memcheck --leak-check=full --error-exitcode=1 ./interactive
```

### Coding Style

La fel ca la tema 0, există o depunțare de până la -20p pentru coding style. Checkerul verifică coding style-ul în mod automat.

### Validare locală temă

Pentru a vă ajuta în dezvoltarea temei, arhiva  checker.zip conține o copie a checkerului.

Pentru a instala depenzile necesare arhivării pentru **coding style** utilizați scriptul **install-linters.sh**:

```
./install-linters.sh
```

Pentru a rula checkerul local, copiați toate fișierele checkerului în același director în care aveți codul sursă, apoi folosiți comanda:

```
./check.sh
```



Checkerul trebuie rulat pe Linux! Trebuie să aveți instalat utilitarul Valgrind (care nu există pe Windows).

### Trimite temă

Tema va fi trimisă folosind Moodle, cursul **Programarea Calculatoarelor (CB & CD)**, activitatea "Tema 1".



Se va posta un anunț pe forum când se va deschide upload-ul.

Toate temele sunt testate în mod automat pe Moodle.


Arhiva temei se va încărca folosind formularul de submitse (butonul **Add submission**).

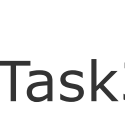
Rezultatele vor fi disponibile în secțiunea **Feedback** - nota apare la linia **Grade**, iar outputul checkerului și erorile apar la secțiunea **Feedback comments**. Dacă apare un buton albastru în formă de plus, trebuie să dați click pe el pentru a afișa întregul output al checkerului.

**Citiți cu atenție** informațiile afișate în **Feedback** pentru a vă asigura că tema a fost rulată cu succes; o eroare comună este dată de faptul că conținutul arhivei nu respectă structura dorită (ex. fișierele sunt într-un alt director).

**Punctajul final al temei** este afișat la linia **Grade** și la finalul outputului din checker.

Conținutul arhivei trebuie să fie următorul:

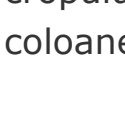
- Fișierele **imageprocessing.c**, **imageprocessing.h**
- Fișierele **bmp.c**, **bmp.h**
- Fișierele **interactive.c**
- Fișierul **Makefile**
- Un fișier  README în care descrieți rezolvarea temei.



Arhiva trebuie să fie de tipul **zip**.



Nu includeți fișierele checkerului în arhiva voastră.



În cazul în care testează pe trec local, insa pica pe vmchecker cel mai probabil aveți o sursa de "undefined behavior - wall". Pentru a va asigura ca scapati de aceste probleme, compilati cu flagul de compilare "-Wall" si rezolvati toate