Zmienne i struktury danych:

Zmienne

Nazwa zmiennej	Typ	Funkcja	Opis
MAIN	MAIN		
start	time_t	start zegara	Obliczanie czasu wykonywania operacji
end	time_t	stop zegara	
duration	time_t	obliczenie czasu trwania operacji	
kolor	int	ustalenie koloru bitmapy	1 - gdy kolorowa bitmapa, 0- gdy czarno-biała bitmapa
wybor	int	wybór opcji głównego menu	-
KczyD	int	ustalenie czy wykonać kompresję, czy dekompresję	1 - kompresja, 2 - dekompresja
KOMPRESJA	KOMPRESJA		
tabx	double*	wskaźnik na kwadrat 8x8	pokazuje na pierwsze pole tablicy t8x8[64], przechowującej uśredniony kwadrat 16x16, przekazywane do makefit
taby	UCHAR*	wskaźnik na kwadrat 16x16	pokazuje na pierwszy piksel kwadratu 16x16, przekazywane do makefit
size	int	rozmiar fraktala 8x8	-
dozapamietania	int	zapis wskaźnika na 16x16	liczba określająca położenie kwadratu 16x16 przed kompresją
zapamietaja	double	zapis współczynnika a	liczby a i b są współczynnikami równania Y[n] = aX[n] + b, które określa jak podobny jest obraz
zapamietajb	double	zapis współczynnika b	
t8x8[64]	double	przechowywanie uśrednionego 16x16	tablica przechowuje uśrednione wartości kwadratu 16x16
postep	int	wyświetlanie postępu operacji	-
indeks8	int	określanie położenia	określenie położenia kwadratu 8x8
errmin	double	porównywanie błędów	jeżeli meta.err jest mniejszy od errmin, dany kwadrat 16x16 powinien zostać zapisany
indeks16	int	określanie położenia	określenie położenia kwadratu 16x16
i16	int	określanie położenia	określa gdzie znajduje się mały kwadracik 4x4 w zadanym kwadracie 16x16 podczas uśredniania

Zmienne pomocnicze

c	int	c = getchar ()	użyta, aby wyczyścić bufor w razie błędnego podania danych przez użytkownika
RGB	int	pętla for()	określa jaki kolor jest przetwarzany w danym momencie
k	int	pętla for()	ilość kolumn po podziale na kwadraty 8x8
w	int	pętla for()	ilość wierszy po podziale na kwadraty 8x8
dk	int	pętla for()	ilość kolumn podczas podziału na kwadraty 16x16
dw	int	pętla for()	ilość wierszy podczas podziału na kwadraty 16x16
i8x8	int	iterowanie pola tablicy t8x8	-
jump2k	int	iterowanie	umożliwia skok o 2 kolumny, aby dostać się do oddalonego o tyle piksela
n	int	pętla for()	-
m	int	pętla for()	-
ile	int	pętla for()	wielokrotne powtórzenie operacji dekompresji
iter	int	iterowanie tablic output i outputcol	służy do wydobycia z tablic zapamiętanych danych dla każdego kwadratu 8x8 podczas dekompresji

Struktury:

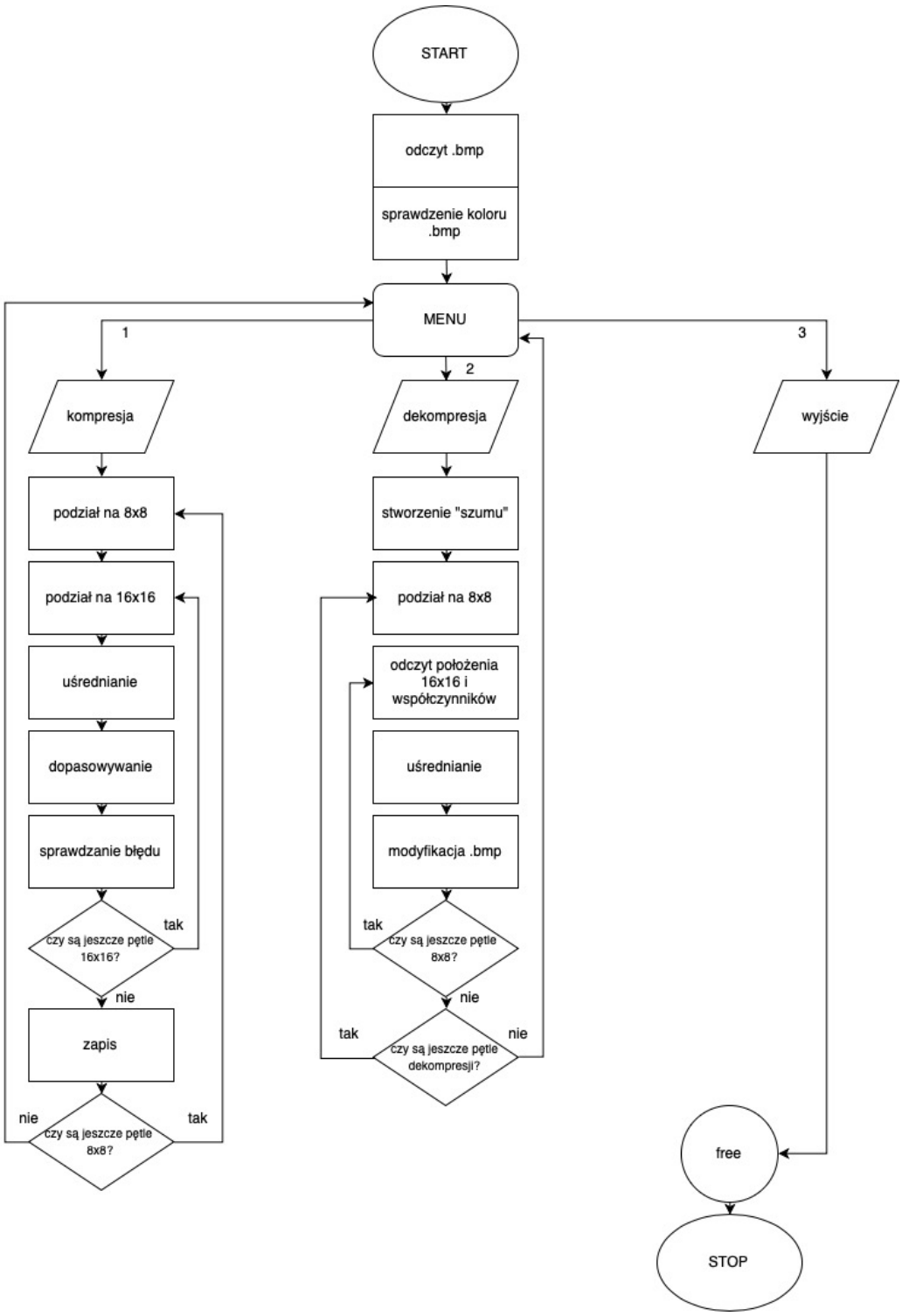
- struct lab_file:
  - UINT width - szerokość bitmapy
  - UINT height - wysokość bitmapy
  - UCHAR *r - kolor czerwony
  - UCHAR *g - kolor zielony
  - UCHAR *b - kolor niebieski
  - UCHAR *i - odcienie szarości
  - UCHAR is_indeks - określenie czy czarnobiała czy kolorowa bitmapa

- struct fit:
  - double a - współczynnik a
  - double b - współczynnik b
  - double err - błąd dopasowania

- struct zapamietane
  - double a - zapamiętany współczynnik a
  - double b - zapamiętany współczynnik b
  - int wskx - zapamiętane położenie 16x16

Zmienne strukturalne:
- struct zapamietane output[4096] - tablica zapamiętanych struktur dla czarno-białych obrazów
- struct zapamietane outputcol[12288] - tablica zapamiętanych struktur dla kolorowych obrazów
- struct lab_file input - wczytana bitmapa
- struct fit meta - dane wyjściowe funkcji makefit

Schemat algorytmu:



Modyfikacje:

1. Easter Egg
2. Zegar
3. Quick Compression

Opis działania programu:

Cel działania programu to skompresowanie (zapis bitmapy w postaci tablicy struktur zawierających 3 liczby: współrzędną położenia ‘wskx’, współczynnik ‘a’, współczynnik ‘b’ - współczynniki równania Y[n] = aX[n]+b), a następnie dekompresowanie tego zdjęcia na podstawie tych właśnie współczynników. Program wczytuje kwadratową bitmapę w kolorze lub w odcieniach szarości, następnie kompresuje je do wyżej wymienionych danych, a po tym odtwarza początkową bitmapę, korzystając z dowolnej tablicy pikseli o tym samym rozmiarze.

Opis czynności w punktach, [główne funkcje]:

- **Start programu [pętla for(; ;)]**
- **wczytanie z pliku (read_bmp)**
- **- Menu Główne: [switch( ) case]**
  - **1 - Kompresja/Szybka Kompresja [case 1/2]**
    - kolorowa/czarnobiała (if, else if)
    - podział na fraktale o rozmiarze 8x8 (nie mogą na siebie nachodzić)
    - podział na fraktale o rozmiarze 16x16 (mogą na siebie nachodzić)
    - uśrednianie fraktali 16x16 do rozmiaru 8x8
    - porównywanie i sprawdzanie podobieństwa każdego fraktala 8x8 z każdym 16x16
    - zapamiętywanie danych najlepszego dopasowania (dla najmniejszego błędu)
  - **2 - Dekompresja [case 3]**
    - tworzenie „szumu“, czyli losowanie wartości pikseli
    - wielokrotne powtórzenie tych operacji:
    - podział na fraktale o rozmiarze 8x8 (nie mogą na siebie nachodzić)
    - odtworzenie położenia zapamiętanego fraktala 16x16 dla każdego z 8x8
    - uśrednianie fraktali 16x16 do rozmiaru 8x8
    - odtworzenie za pomocą wzoru Y[n] = aX[n]+b
    - modyfikacja wylosowanego „szumu“, poprzez wstawienie nowych, obliczonych wyżej wartości
  - **3 - Easter Egg [case 3]**
  - **4 - wyjście z programu [case 4]**
- **Koniec programu**