

Operacje morfologiczne na obrazie
0.97

Wygenerowano przez Doxygen 1.8.16

1 Indeks struktur danych	1
1.1 Struktury danych	1
2 Indeks plików	3
2.1 Lista plików	3
3 Dokumentacja struktur danych	5
3.1 Dokumentacja struktury BITMAPHEADER	5
3.1.1 Opis szczegółowy	6
3.1.2 Dokumentacja pól	6
3.1.2.1 colors_importance_rotation	6
3.1.2.2 colors_used	6
3.1.2.3 comp	6
3.1.2.4 height	6
3.1.2.5 image_size	6
3.1.2.6 numBitPlanes	7
3.1.2.7 numBitsPerPlane	7
3.1.2.8 reserved	7
3.1.2.9 size	7
3.1.2.10 width	7
3.1.2.11 Xres	7
3.1.2.12 Yres	8
3.2 Dokumentacja struktury BMPFILEHEADER	8
3.2.1 Opis szczegółowy	8
3.2.2 Dokumentacja pól	8
3.2.2.1 offset	8
3.2.2.2 r1	9
3.2.2.3 r2	9
3.2.2.4 s1	9
3.2.2.5 s2	9
3.2.2.6 size	9
3.3 Dokumentacja struktury PIXEL	9
3.3.1 Opis szczegółowy	10
3.3.2 Dokumentacja pól	10
3.3.2.1 blue	10
3.3.2.2 green	10
3.3.2.3 red	10
3.4 Dokumentacja struktury pixel_structure	11
3.4.1 Opis szczegółowy	11
3.4.2 Dokumentacja pól	11
3.4.2.1 next	11
3.4.2.2 type	11

4 Dokumentacja plików	13
4.1 Dokumentacja pliku bmp.h	13
4.1.1 Opis szczegółowy	14
4.1.2 Dokumentacja definicji typów	14
4.1.2.1 BITMAPHEADER	14
4.1.2.2 BMPFILEHEADER	14
4.1.2.3 PIXEL	14
4.1.2.4 pixel_structure	14
4.2 Dokumentacja pliku functions.c	15
4.2.1 Opis szczegółowy	16
4.2.2 Dokumentacja funkcji	16
4.2.2.1 AllocateInputMemory()	16
4.2.2.2 AllocateOutputMemory()	16
4.2.2.3 change_pixel_color()	17
4.2.2.4 check_pixel_color()	17
4.2.2.5 closing()	18
4.2.2.6 deallocate_sieve()	18
4.2.2.7 DeallocateInputMemory()	18
4.2.2.8 DeallocateOutputMemory()	19
4.2.2.9 load_sieve()	19
4.2.2.10 main_point()	19
4.2.2.11 opening()	20
4.2.2.12 operation()	20
4.2.2.13 ReadPixels()	21
4.2.2.14 ReadToStructure()	21
4.2.2.15 SaveBMPFile()	21
4.2.2.16 WriteFromStructure()	23
4.3 Dokumentacja pliku functions.h	23
4.3.1 Opis szczegółowy	24
4.3.2 Dokumentacja funkcji	24
4.3.2.1 AllocateInputMemory()	24
4.3.2.2 AllocateOutputMemory()	25
4.3.2.3 change_pixel_color()	25
4.3.2.4 check_pixel_color()	26
4.3.2.5 closing()	26
4.3.2.6 deallocate_sieve()	26
4.3.2.7 DeallocateInputMemory()	27
4.3.2.8 DeallocateOutputMemory()	27
4.3.2.9 load_sieve()	28
4.3.2.10 main_point()	28
4.3.2.11 opening()	28
4.3.2.12 operation()	30

4.3.2.13 ReadPixels()	30
4.3.2.14 ReadToStructure()	31
4.3.2.15 SaveBMPFile()	31
4.3.2.16 WriteFromStructure()	32
4.4 Dokumentacja pliku main.c	32
4.4.1 Opis szczegółowy	32
4.4.2 Dokumentacja funkcji	32
4.4.2.1 main()	33
Indeks	35

Rozdział 1

Indeks struktur danych

1.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

BITMAPHEADER	Struktura przechowująca nagłówek DIB	5
BMPFILEHEADER	Pierwsza w kolejności podstruktura struktury nagłówkowej pliku mapy bitowej	8
PIXEL	Struktura przechowująca poszczególne składowe kolorów pikseli	9
pixel_structure	Struktura listy jednokierunkowej elementu strukturalnego. Składa się ona ze wskaźnika na następny element elementu strukturalnego, który zawiera znak. Ta struktura jest synchronizowana z siatką pikselu obrazu za pomocą odpowiednich funkcji	11

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

bmp.h	Plik nagłówkowy zawierający wszelkie struktury, potrzebne do działania programu	13
functions.c	Plik nagłówkowy zawierający definicje funkcji	15
functions.h	Plik nagłówkowy zawierający deklaracje funkcji	23
main.c	Plik źródłowy	32

Rozdział 3

Dokumentacja struktur danych

3.1 Dokumentacja struktury BITMAPHEADER

Struktura przechowująca nagłówek DIB.

```
#include <bmp.h>
```

Pola danych

- uint32_t [size](#)
Wartość ta wskazuje na ilość bajtów, z których składa się drugi podnagłówek pliku mapy bitowej.
- uint32_t [width](#)
Szerokość obrazu.
- uint32_t [height](#)
Wysokość obrazu.
- uint16_t [numBitPlanes](#)
Liczba używanych przez obraz płaszczyzn kolorów.
- uint16_t [numBitsPerPlane](#)
Ilość bitów na piksel.
- uint32_t [comp](#)
Parametr, wskazujący na typ kompresji (w tym przypadku 0).
- uint32_t [image_size](#)
- uint32_t [Xres](#)
Rozdzielczość pozioma.
- uint32_t [Yres](#)
Rozdzielczość pionowa.
- uint32_t [colors_used](#)
Ilość kolorów w palecie.
- uint16_t [colors_importance_rotation](#)
Ilość ważnych kolorów w palecie (w tym przypadku 0; wszystkie kolory są jednokowo ważne).
- uint16_t [reserved](#)
Bajty zarezerwowane przez aplikację.

3.1.1 Opis szczegółowy

Struktura przechowująca nagłówek DIB.

3.1.2 Dokumentacja pól

3.1.2.1 colors_importance_rotation

```
uint16_t colors_importance_rotation
```

Ilość ważnych kolorów w palecie (w tym przypadku 0; wszystkie kolory są jednokowo ważne).

3.1.2.2 colors_used

```
uint32_t colors_used
```

Ilość kolorów w palecie.

3.1.2.3 comp

```
uint32_t comp
```

Parametr, wskazujący na typ kompresji (w tym przypadku 0).

3.1.2.4 height

```
uint32_t height
```

Wysokość obrazu.

3.1.2.5 image_size

```
uint32_t image_size
```

3.1.2.6 numBitPlanes

```
uint16_t numBitPlanes
```

Liczba używanych przez obraz płaszczyzn kolorów.

3.1.2.7 numBitsPerPlane

```
uint16_t numBitsPerPlane
```

Ilość bitów na piksel.

3.1.2.8 reserved

```
uint16_t reserved
```

Bajty zarezerwowane przez aplikację.

3.1.2.9 size

```
uint32_t size
```

Wartość ta wskazuje na ilość bajtów, z których składa się drugi podnagłówek pliku mapy bitowej.

3.1.2.10 width

```
uint32_t width
```

Szerokość obrazu.

3.1.2.11 Xres

```
uint32_t Xres
```

Rozdzielczość pozioma.

3.1.2.12 Yres

```
uint32_t Yres
```

Rozdzielczość pionowa.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [bmp.h](#)

3.2 Dokumentacja struktury BMPFILEHEADER

Pierwsza w kolejności podstruktura struktury nagłówkowej pliku mapy bitowej.

```
#include <bmp.h>
```

Pola danych

- [uint8_t s1](#)
Pierwszy znak typu pliku w kodzie ASCII (powinien wskazywać 'M').
- [uint8_t s2](#)
Drugi znak typu pliku w kodzie ASCII (powinien wskazywać 'M').
- [uint32_t size](#)
Rozmiar pliku obrazu mapy bitowej (powinno być 70 bajtów).
- [uint16_t r1](#)
Bajt zarezerwowany dla aplikacji.
- [uint16_t r2](#)
Bajt zarezerwowany dla aplikacji.
- [uint32_t offset](#)
Numer bajtu, który rozpoczyna macierz pikseli.

3.2.1 Opis szczegółowy

Pierwsza w kolejności podstruktura struktury nagłówkowej pliku mapy bitowej.

3.2.2 Dokumentacja pól

3.2.2.1 offset

```
uint32_t offset
```

Numer bajtu, który rozpoczyna macierz pikseli.

3.2.2.2 r1

```
uint16_t r1
```

Bajt zarezerwowany dla aplikacji.

3.2.2.3 r2

```
uint16_t r2
```

Bajt zarezerwowany dla aplikacji.

3.2.2.4 s1

```
uint8_t s1
```

Pierwszy znak typu pliku w kodzie ASCII (powinien wskazywać 'M').

3.2.2.5 s2

```
uint8_t s2
```

Drugi znak typu pliku w kodzie ASCII (powinien wskazywać 'M').

3.2.2.6 size

```
uint32_t size
```

Rozmiar pliku obrazu mapy bitowej (powinno być 70 bajtów).

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [bmp.h](#)

3.3 Dokumentacja struktury PIXEL

Struktura przechowująca poszczególne składowe kolorów pikseli.

```
#include <bmp.h>
```

Pola danych

- `uint8_t blue`
Składowa niebieska koloru piksela.
- `uint8_t green`
Składowa zielona koloru piksela.
- `uint8_t red`
Składowa czerwona koloru piksela.

3.3.1 Opis szczegółowy

Struktura przechowująca poszczególne składowe kolorów pikseli.

3.3.2 Dokumentacja pól

3.3.2.1 blue

```
uint8_t blue
```

Składowa niebieska koloru piksela.

3.3.2.2 green

```
uint8_t green
```

Składowa zielona koloru piksela.

3.3.2.3 red

```
uint8_t red
```

Składowa czerwona koloru piksela.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [bmp.h](#)

3.4 Dokumentacja struktury pixel_structure

Struktura listy jednokierunkowej elementu strukturalnego. Składa się ona ze wskaźnika na następny element elementu strukturalnego, który zawiera znak. Ta struktura jest synchronizowana z siatką pikselu obrazu za pomocą odpowiednich funkcji.

```
#include <bmp.h>
```

Pola danych

- [pixel_structure * next](#)
Wskaźnik na następny element (znak) elementu strukturalnego.
- [uint8_t type](#)
Znak, który zawiera dany element elementu strukturalnego.

3.4.1 Opis szczegółowy

Struktura listy jednokierunkowej elementu strukturalnego. Składa się ona ze wskaźnika na następny element elementu strukturalnego, który zawiera znak. Ta struktura jest synchronizowana z siatką pikselu obrazu za pomocą odpowiednich funkcji.

3.4.2 Dokumentacja pól

3.4.2.1 next

```
pixel\_structure* next
```

Wskaźnik na następny element (znak) elementu strukturalnego.

3.4.2.2 type

```
uint8_t type
```

Znak, który zawiera dany element elementu strukturalnego.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [bmp.h](#)

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku bmp.h

Plik nagłówkowy zawierający wszelkie struktury, potrzebne do działania programu.

```
#include <inttypes.h>
```

Struktury danych

- struct [BMPFILEHEADER](#)
Pierwsza w kolejności podstruktura struktury nagłówkowej pliku mapy bitowej.
- struct [BITMAPHEADER](#)
Struktura przechowująca nagłówek DIB.
- struct [PIXEL](#)
Struktura przechowująca poszczególne składowe kolorów pikseli.
- struct [pixel_structure](#)
Struktura listy jednokierunkowej elementu strukturalnego. Składa się ona ze wskaźnika na następny element elementu strukturalnego, który zawiera znak. Ta struktura jest synchronizowana z siatką pikselu obrazu za pomocą odpowiednich funkcji.

Definicje typów

- typedef struct [BMPFILEHEADER](#) [BMPFILEHEADER](#)
Pierwsza w kolejności podstruktura struktury nagłówkowej pliku mapy bitowej.
- typedef struct [BITMAPHEADER](#) [BITMAPHEADER](#)
Struktura przechowująca nagłówek DIB.
- typedef struct [PIXEL](#) [PIXEL](#)
Struktura przechowująca poszczególne składowe kolorów pikseli.
- typedef struct [pixel_structure](#) [pixel_structure](#)
Struktura listy jednokierunkowej elementu strukturalnego. Składa się ona ze wskaźnika na następny element elementu strukturalnego, który zawiera znak. Ta struktura jest synchronizowana z siatką pikselu obrazu za pomocą odpowiednich funkcji.

4.1.1 Opis szczegółowy

Plik nagłówkowy zawierający wszelkie struktury, potrzebne do działania programu.

Autor

Aleksander Augustyniak

4.1.2 Dokumentacja definicji typów

4.1.2.1 BITMAPHEADER

```
typedef struct BITMAPHEADER BITMAPHEADER
```

Struktura przechowująca nagłówek DIB.

4.1.2.2 BMPFILEHEADER

```
typedef struct BMPFILEHEADER BMPFILEHEADER
```

Pierwsza w kolejności podstruktura struktury nagłówkowej pliku mapy bitowej.

4.1.2.3 PIXEL

```
typedef struct PIXEL PIXEL
```

Struktura przechowująca poszczególne składowe kolorów pikseli.

4.1.2.4 pixel_structure

```
typedef struct pixel_structure pixel_structure
```

Struktura listy jednokierunkowej elementu strukturalnego. Składa się ona ze wskaźnika na następny element elementu strukturalnego, który zawiera znak. Ta struktura jest synchronizowana z siatką pikselu obrazu za pomocą odpowiednich funkcji.

4.2 Dokumentacja pliku functions.c

Plik nagłówkowy zawierający definicje funkcji.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <inttypes.h>
#include "bmp.h"
#include "functions.h"
```

Funkcje

- `uint8_t *** AllocateInputMemory (const uint32_t *imgHeight, const uint32_t *imgWidth)`
Funkcja alokująca tyle pamięci, by pomieścić cały zakres kolorów (wysokość x szerokość x 3 składowe).
- `uint8_t *** DeallocateInputMemory (uint8_t ***image_in, const uint32_t *imgHeight, const uint32_t *imgWidth)`
Funkcja dealokująca całą pamięć zajęłą przez funkcję AllocateInputMemory.
- `void ReadPixels (uint8_t ***image_in, const uint32_t *imgHeight, const uint32_t *imgWidth, const FILE *input_file, const uint32_t *offset_byte)`
- `uint8_t *** AllocateOutputMemory (uint8_t ***image_in, const uint32_t *imgHeight, const uint32_t *imgWidth)`
Funkcja alokująca tyle pamięci, by pomieścić cały zakres kolorów dla pliku wyjściowego.
- `uint8_t *** DeallocateOutputMemory (uint8_t ***image_out, const uint8_t *imgHeight, const uint8_t *imgWidth)`
Funkcja dealokująca całą pamięć zajęłą przez funkcję AllocateOutputMemory.
- `void SaveBMPFile (const uint32_t *imgHeight, const uint32_t *imgWidth, const uint8_t ***image_out, const FILE *output_file)`
Funkcja zapisująca bajty składowych kolorów pikseli do pliku wyjściowego.
- `void ReadToStructure (const BMPFILEHEADER *FileHeader, const BITMAPHEADER *Header, const FILE *input_file)`
Funkcja zapisująca do struktury dane nagłówka pliku, zawierającego informacje nt. obrazu, m.in. rozdzielczość, wysokość, szerokość, waga pliku, etc.
- `void WriteFromStructure (const BMPFILEHEADER *FileHeader, const BITMAPHEADER *Header, const FILE *output_file)`
Funkcja zapisująca do pliku wyjściowego pozyskane przez funkcję ReadToStructure dane, zapisane do struktury.
- `uint8_t * change_pixel_color (uint8_t *pixel, uint8_t color)`
Funkcja zamienia kolor danego przez adres piksela.
- `uint8_t check_pixel_color (uint8_t *pixel)`
Funkcja sprawdza jakiego koloru jest podany piksel.
- `void operation (const uint8_t type, uint8_t ***image_in, uint8_t ***image_out, uint32_t width, uint32_t height, pixel_structure *sieve, uint32_t *coordinates)`
Główny algorytm, który odpowiada za wykonanie operacji morfologicznych na obrazie. Funkcja dopasowuje strukturę sita do piksela modyfikowanego tak, by można było sprawdzić warunki zamiany piksela modyfikowanego.
- `void opening (uint8_t ***image_in, uint8_t ***image_out, uint32_t width, uint32_t height, pixel_structure *sieve, uint32_t *coordinates)`
Funkcja otwarcia morfologicznego, która wykonuje na obrazie w odpowiedniej kolejności operacje w celu usunięcia szczegółów z obrazu (erozja -> dyatacja).
- `void closing (uint8_t ***image_in, uint8_t ***image_out, uint32_t width, uint32_t height, pixel_structure *sieve, uint32_t *coordinates)`

Funkcja zamknięcia morfologicznego, która wykonuje na obrazie w odpowiedniej kolejności operacje w celu "zamalowania" niewypełnionych do końca szczelin (dylatacja-> erozja). Wymagany efekt ściśle zależy od elementu strukturalnego.

- `pixel_structure * load_sieve (FILE *stream)`

Funkcja wczytująca do specjalnej struktury element strukturalny.

- `uint32_t * main_point (pixel_structure *sieve)`

Algorytm wykrywający współrzędne piksela modyfikowanego (głównego) elementu strukturalnego.

- `void deallocate_sieve (pixel_structure *sieve)`

Funkcja dealokująca pamięć, potrzebną by wczytać element strukturalny.

4.2.1 Opis szczegółowy

Plik nagłówkowy zawierający definicje funkcji.

Autor

Aleksander Augustyniak

4.2.2 Dokumentacja funkcji

4.2.2.1 AllocateInputMemory()

```
uint8_t*** AllocateInputMemory (
    const uint32_t * imgHeight,
    const uint32_t * imgWidth )
```

Funkcja alokująca tyle pamięci, by pomieścić cały zakres kolorów (wysokość x szerokość x 3 składowe).

Parametry

<code>imgHeight</code>	wskaźnik na wysokość obrazu.
<code>imgWidth</code>	wskaźnik na szerokość obrazu.

Zwraca

wskaźnik do adresu schowka na dane wejściowe lub NULL.

4.2.2.2 AllocateOutputMemory()

```
uint8_t*** AllocateOutputMemory (
    uint8_t *** image_in,
    const uint32_t * imgHeight,
    const uint32_t * imgWidth )
```

Funkcja alokująca tyle pamięci, by pomieścić cały zakres kolorów dla pliku wyjściowego.

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.

Zwraca

wskaźnik do adresu schowka na dane wyjściowe lub NULL.

4.2.2.3 change_pixel_color()

```
uint8_t* change_pixel_color (
    uint8_t * pixel,
    uint8_t color )
```

Funkcja zamienia kolor danego przez adres piksela.

Parametry

<i>pixel</i>	wskaźnik do piksela edytowanego.
<i>color</i>	współczynnik mówiący o tym, na jaki kolor ma być zamieniony dany piksel (1: biały / 0: czarny).

Zwraca

wskaźnik na piksel zmodyfikowany.

4.2.2.4 check_pixel_color()

```
uint8_t check_pixel_color (
    uint8_t * pixel )
```

Funkcja sprawdza jakiego koloru jest podany piksel.

Parametry

<i>pixel</i>	wskaźnik do piksela, którego poszczególne składowe koloru mają być sprawdzone.
--------------	--

Zwraca

0 jeżeli wszystkie składowe koloru piksela są równe zero. W przeciwnym wypadku 1.

4.2.2.5 closing()

```
void closing (
    uint8_t *** image_in,
    uint8_t *** image_out,
    uint32_t width,
    uint32_t height,
    pixel_structure * sieve,
    uint32_t * coordinates )
```

Funkcja zamknięcia morfologicznego, która wykonuje na obrazie w odpowiedniej kolejności operacje w celu "zamalowania" niewypełnionych do końca szczelin (dylatacja-> erozja). Wymagany efekt ściśle zależy od elementu strukturalnego.

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>width</i>	szerokość obrazu.
<i>height</i>	wysokość obrazu.
<i>sieve</i>	wskaźnik na strukturę zawierającą dane elementu strukturalnego.
<i>coordinates</i>	tablica zawierająca współrzędne położenia piksela modyfikowanego (głównego).

4.2.2.6 deallocate_sieve()

```
void deallocate_sieve (
    pixel_structure * sieve )
```

Funkcja dealokująca pamięć, potrzebną by wczytać element strukturalny.

Parametry

<i>sieve</i>	wskaźnik na dynamiczną listę jednokierunkową zawierającą element strukturalny.
--------------	--

4.2.2.7 DeallocateInputMemory()

```
uint8_t*** DeallocateInputMemory (
    uint8_t *** image_in,
    const uint32_t * imgHeight,
    const uint32_t * imgWidth )
```

Funkcja dealokująca całą pamięć zajętą przez funkcję AllocateInputMemory.

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.

Zwraca

NULL.

4.2.2.8 DeallocateOutputMemory()

```
uint8_t*** DeallocateOutputMemory (
    uint8_t *** image_out,
    const uint8_t * imgHeight,
    const uint8_t * imgWidth )
```

Funkcja dealokująca całą pamięć zajęta przez funkcję AllocateOutputMemory.

Parametry

<i>image_out</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.

Zwraca

NULL.

4.2.2.9 load_sieve()

```
pixel_structure* load_sieve (
    FILE * stream )
```

Funkcja wczytująca do specjalnej struktury element strukturalny.

Parametry

<i>stream</i>	wskaźnik do pliku, który zawiera element strukturalny.
---------------	--

Zwraca

wskaźnik na dynamiczną listę jednokierunkową zawierającą element strukturalny.

4.2.2.10 main_point()

```
uint32_t* main_point (
    pixel_structure * sieve )
```

Algorytm wykrywający współrzędne piksela modyfikowanego (głównego) elementu strukturalnego.

Parametry

<i>sieve</i>	wskaźnik na dynamiczną listę jednokierunkową zawierającą element strukturalny.
--------------	--

Zwraca

wskaźnik na tablicę współrzędnych punktu głównego elementu strukturalnego.

4.2.2.11 opening()

```
void opening (
    uint8_t *** image_in,
    uint8_t *** image_out,
    uint32_t width,
    uint32_t height,
    pixel_structure * sieve,
    uint32_t * coordinates )
```

Funkcja otwarcia morfologicznego, która wykonuje na obrazie w odpowiedniej kolejności operacje w celu usunięcia szczegółów z obrazu (erozja -> dylatacja).

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>width</i>	szerokość obrazu.
<i>height</i>	wysokość obrazu.
<i>sieve</i>	wskaźnik na strukturę zawierającą dane elementu strukturalnego.
<i>coordinates</i>	tablica zawierająca współrzędne położenia piksela modyfikowanego (głównego).

4.2.2.12 operation()

```
void operation (
    const uint8_t type,
    uint8_t *** image_in,
    uint8_t *** image_out,
    uint32_t width,
    uint32_t height,
    pixel_structure * sieve,
    uint32_t * coordinates )
```

Główny algorytm, który odpowiada za wykonanie operacji morfologicznych na obrazie. Funkcja dopasowuje strukturę sita do piksela modyfikowanego tak, by można było sprawdzić warunki zamiany piksela modyfikowanego.

Parametry

<i>type</i>	znak warunkujący operację, która ma być wykonana przez funkcję ('e' – erozja; 'd' – dylatacja).
-------------	---

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>image_out</i>	wskaźnik do początku tablicy na dane wyjściowe.
<i>width</i>	szerokość obrazu.
<i>height</i>	wysokość obrazu.
<i>sieve</i>	wskaźnik na strukturę zawierającą dane elementu strukturalnego.
<i>coordinates</i>	tablica zawierająca współrzędne położenia piksela modyfikowanego (głównego).

4.2.2.13 ReadPixels()

```
void ReadPixels (
    uint8_t *** image_in,
    const uint32_t * imgHeight,
    const uint32_t * imgWidth,
    const FILE * input_file,
    const uint32_t * offset_byte )
```

4.2.2.14 ReadToStructure()

```
void ReadToStructure (
    const BMPFILEHEADER * FileHeader,
    const BITMAPHEADER * Header,
    const FILE * input_file )
```

Funkcja zapisująca do struktury dane nagłówka pliku, zawierającego informacje nt. obrazu, m.in. rozdzielczość, wysokość, szerokość, waga pliku, etc.

Parametry

<i>FileHeader</i>	wskaźnik na podstrukturę mówiąca o tym, czy plik rzeczywiście jest plikiem mapy bitowej. Podstruktura mówi m.in. od którego bajtu zaczynają się piksele.
<i>Header</i>	wskaźnik na podstrukturę, której dane opisują wymiary obrazu. Podstruktura zawiera przede wszystkim informacje techniczne obrazu.
<i>input_file</i>	wskaźnik na plik wejściowy.

4.2.2.15 SaveBMPFile()

```
void SaveBMPFile (
    const uint32_t * imgHeight,
    const uint32_t * imgWidth,
```

```
const uint8_t *** image_out,  
const FILE * output_file )
```

Funkcja zapisująca bajty składowych kolorów pikseli do pliku wyjściowego.

Parametry

<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.
<i>image_out</i>	wskaźnik do początku tablicy na dane wyjściowe.
<i>output_file</i>	wskaźnik na plik wyjściowy.

4.2.2.16 WriteFromStructure()

```
void WriteFromStructure (
    const BMPFILEHEADER * FileHeader,
    const BITMAPHEADER * Header,
    const FILE * output_file )
```

Funkcja zapisująca do pliku wyjściowego pozyskane przez funkcję ReadToStructure dane, zapisane do struktury.

Parametry

<i>FileHeader</i>	wskaźnik na podstrukturę mówiąca o tym, czy plik rzeczywiście jest plikiem mapy bitowej. Podstruktura mówi m.in. od którego bajtu zaczynają się piksele.
<i>Header</i>	wskaźnik na podstrukturę, której dane opisują wymiary obrazu. Podstruktura zawiera przede wszystkim informacje techniczne obrazu.
<i>output_file</i>	wskaźnik na plik wyjściowy.

4.3 Dokumentacja pliku functions.h

Plik nagłówkowy zawierający deklaracje funkcji.

Funkcje

- `uint8_t *** AllocateInputMemory (const uint32_t *imgHeight, const uint32_t *imgWidth)`
Funkcja alokująca tyle pamięci, by pomieścić cały zakres kolorów (wysokość x szerokość x 3 składowe).
- `uint8_t *** DeallocateInputMemory (uint8_t ***image_in, const uint32_t *imgHeight, const uint32_t *imgWidth)`
Funkcja dealokująca całą pamięć zajęta przez funkcję [AllocateInputMemory](#).
- `void ReadPixels (uint8_t ***image_in, const uint32_t *imgHeight, const uint32_t *imgWidth, const FILE *input_file, uint32_t *offset_byte)`
Funkcja czytująca do struktury poszczególne składowe każdego piksela obrazu wejściowego. Jeżeli którykolwiek ze składowych koloru piksela ma wartość większą od 0 – wszystkie składowe są maksymalizowane.
- `uint8_t *** AllocateOutputMemory (uint8_t ***image_in, const uint32_t *imgHeight, const uint32_t *imgWidth)`
Funkcja alokująca tyle pamięci, by pomieścić cały zakres kolorów dla pliku wyjściowego.
- `uint8_t *** DeallocateOutputMemory (uint8_t ***image_out, const uint8_t *imgHeight, const uint8_t *imgWidth)`
Funkcja dealokująca całą pamięć zajęta przez funkcję [AllocateOutputMemory](#).

- void [SaveBMPFile](#) (const uint32_t *imgHeight, const uint32_t *imgWidth, const uint8_t ***image_out, const FILE *output_file)
Funkcja zapisująca bajty składowych kolorów pikseli do pliku wyjściowego.
- void [ReadToStructure](#) (const [BMPFILEHEADER](#) *FileHeader, const [BITMAPHEADER](#) *Header, const FILE *input_file)
Funkcja zapisująca do struktury dane nagłówka pliku, zawierającego informacje nt. obrazu, m.in. rozdzielczość, wysokość, szerokość, waga pliku, etc.
- void [WriteFromStructure](#) (const [BMPFILEHEADER](#) *FileHeader, const [BITMAPHEADER](#) *Header, const FILE *output_file)
Funkcja zapisująca do pliku wyjściowego pozyskane przez funkcję [ReadToStructure](#) dane, zapisane do struktury.
- uint8_t * [change_pixel_color](#) (uint8_t *pixel, uint8_t color)
Funkcja zamienia kolor danego przez adres piksela.
- uint8_t [check_pixel_color](#) (uint8_t *pixel)
Funkcja sprawdza jakiego koloru jest podany piksel.
- void [operation](#) (const uint8_t type, uint8_t ***image_in, uint8_t ***image_out, uint32_t width, uint32_t height, [pixel_structure](#) *sieve, uint32_t *coordinates)
Główny algorytm, który odpowiada za wykonanie operacji morfologicznych na obrazie. Funkcja dopasowuje strukturę sita do piksela modyfikowanego tak, by można było sprawdzić warunki zamiany piksela modyfikowanego.
- void [opening](#) (uint8_t ***image_in, uint8_t ***image_out, uint32_t width, uint32_t height, [pixel_structure](#) *sieve, uint32_t *coordinates)
Funkcja otwarcia morfologicznego, która wykonuje na obrazie w odpowiedniej kolejności operacje w celu usunięcia szczegółów z obrazu (erozja -> dylatacja).
- void [closing](#) (uint8_t ***image_in, uint8_t ***image_out, uint32_t width, uint32_t height, [pixel_structure](#) *sieve, uint32_t *coordinates)
Funkcja zamknięcia morfologicznego, która wykonuje na obrazie w odpowiedniej kolejności operacje w celu "za-malowania" niewypełnionych do końca szczelin (dylatacja-> erozja). Wymagany efekt ściśle zależy of elementu strukturalnego.
- [pixel_structure](#) * [load_sieve](#) (FILE *stream)
Funkcja wczytująca do specjalnej struktury element strukturalny.
- void [deallocate_sieve](#) ([pixel_structure](#) *sieve)
Funkcja dealokująca pamięć, potrzebną by wczytać element strukturalny.
- uint32_t * [main_point](#) ([pixel_structure](#) *sieve)
Algorytm wykrywający współrzędne piksela modyfikowanego (głównego) elementu strukturalnego.

4.3.1 Opis szczegółowy

Plik nagłówkowy zawierający deklaracje funkcji.

Autor

Aleksander Augustyniak

4.3.2 Dokumentacja funkcji

4.3.2.1 AllocateInputMemory()

```
uint8_t*** AllocateInputMemory (
    const uint32_t * imgHeight,
    const uint32_t * imgWidth )
```

Funkcja alokująca tyle pamięci, by pomieścić cały zakres kolorów (wysokość x szerokość x 3 składowe).

Parametry

<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.

Zwraca

wskaźnik do adresu schowka na dane wejściowe lub NULL.

4.3.2.2 AllocateOutputMemory()

```
uint8_t*** AllocateOutputMemory (
    uint8_t *** image_in,
    const uint32_t * imgHeight,
    const uint32_t * imgWidth )
```

Funkcja alokująca tyle pamięci, by pomieścić cały zakres kolorów dla pliku wyjściowego.

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.

Zwraca

wskaźnik do adresu schowka na dane wyjściowe lub NULL.

4.3.2.3 change_pixel_color()

```
uint8_t* change_pixel_color (
    uint8_t * pixel,
    uint8_t color )
```

Funkcja zamienia kolor danego przez adres piksela.

Parametry

<i>pixel</i>	wskaźnik do piksela edytowanego.
<i>color</i>	współczynnik mówiący o tym, na jaki kolor ma być zamieniony dany piksel (1: biały / 0: czarny).

Zwraca

wskaźnik na piksel zmodyfikowany.

4.3.2.4 check_pixel_color()

```
uint8_t check_pixel_color (
    uint8_t * pixel )
```

Funkcja sprawdza jakiego koloru jest podany piksel.

Parametry

<i>pixel</i>	wskaźnik do piksela, którego poszczególne składowe koloru mają być sprawdzone.
--------------	--

Zwraca

0 jeżeli wszystkie składowe koloru piksela są równe zero. W przeciwnym wypadku 1.

4.3.2.5 closing()

```
void closing (
    uint8_t *** image_in,
    uint8_t *** image_out,
    uint32_t width,
    uint32_t height,
    pixel_structure * sieve,
    uint32_t * coordinates )
```

Funkcja zamknięcia morfologicznego, która wykonuje na obrazie w odpowiedniej kolejności operacje w celu "zamalowania" niewypełnionych do końca szczelin (dylatacja-> erozja). Wymagany efekt ściśle zależy od elementu strukturalnego.

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>width</i>	szerokość obrazu.
<i>width</i>	wysokość obrazu.
<i>sieve</i>	wskaźnik na strukturę zawierającą dane elementu strukturalnego.
<i>coordinates</i>	tablica zawierająca współrzędne położenia piksela modyfikowanego (głównego).

4.3.2.6 deallocate_sieve()

```
void deallocate_sieve (
```



```
pixel_structure * sieve )
```

Funkcja dealokująca pamięć, potrzebną by wczytać element strukturalny.

Parametry

<i>sieve</i>	wskaźnik na dynamiczną listę jednokierunkową zawierającą element strukturalny.
--------------	--

4.3.2.7 DeallocateInputMemory()

```
uint8_t*** DeallocateInputMemory (
    uint8_t *** image_in,
    const uint32_t * imgHeight,
    const uint32_t * imgWidth )
```

Funkcja dealokująca całą pamięć zajętą przez funkcję AllocateInputMemory.

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.

Zwraca

NULL.

4.3.2.8 DeallocateOutputMemory()

```
uint8_t*** DeallocateOutputMemory (
    uint8_t *** image_out,
    const uint8_t * imgHeight,
    const uint8_t * imgWidth )
```

Funkcja dealokująca całą pamięć zajętą przez funkcję AllocateOutputMemory.

Parametry

<i>image_out</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.

Zwraca

NULL.

4.3.2.9 load_sieve()

```
pixel_structure* load_sieve (
    FILE * stream )
```

Funkcja wczytująca do specjalnej struktury element strukturalny.

Parametry

<i>stream</i>	wskaźnik do pliku, który zawiera element strukturalny.
---------------	--

Zwraca

wskaźnik na dynamiczną listę jednokierunkową zawierającą element strukturalny.

4.3.2.10 main_point()

```
uint32_t* main_point (
    pixel_structure * sieve )
```

Algorytm wykrywający współrzędne piksela modyfikowanego (głównego) elementu strukturalnego.

Parametry

<i>sieve</i>	wskaźnik na dynamiczną listę jednokierunkową zawierającą element strukturalny.
--------------	--

Zwraca

wskaźnik na tablicę współrzędnych punktu głównego elementu strukturalnego.

4.3.2.11 opening()

```
void opening (
    uint8_t *** image_in,
    uint8_t *** image_out,
    uint32_t width,
    uint32_t height,
```

```
pixel_structure * sieve,  
uint32_t * coordinates )
```

Funkcja otwarcia morfologicznego, która wykonuje na obrazie w odpowiedniej kolejności operacje w celu usunięcia szczegółów z obrazu (erozja -> dylatacja).

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>width</i>	szerokość obrazu.
<i>width</i>	wysokość obrazu.
<i>sieve</i>	wskaźnik na strukturę zawierającą dane elementu strukturalnego.
<i>coordinates</i>	tablica zawierająca współrzędne położenia piksela modyfikowanego (głównego).

4.3.2.12 operation()

```
void operation (
    const uint8_t type,
    uint8_t *** image_in,
    uint8_t *** image_out,
    uint32_t width,
    uint32_t height,
    pixel_structure * sieve,
    uint32_t * coordinates )
```

Główny algorytm, który odpowiada za wykonanie operacji morfologicznych na obrazie. Funkcja dopasowuje strukturę sita do piksela modyfikowanego tak, by można było sprawdzić warunki zamiany piksela modyfikowanego.

Parametry

<i>type</i>	znak warunkujący operację, która ma być wykonana przez funkcję ('e' – erozja; 'd' – dylatacja).
<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>image_out</i>	wskaźnik do początku tablicy na dane wyjściowe.
<i>width</i>	szerokość obrazu.
<i>width</i>	wysokość obrazu.
<i>sieve</i>	wskaźnik na strukturę zawierającą dane elementu strukturalnego.
<i>coordinates</i>	tablica zawierająca współrzędne położenia piksela modyfikowanego (głównego).

4.3.2.13 ReadPixels()

```
void ReadPixels (
    uint8_t *** image_in,
    const uint32_t * imgHeight,
    const uint32_t * imgWidth,
    const FILE * input_file,
    uint32_t * offset_byte )
```

Funkcja czytująca do struktury poszczególne składowe każdego piksela obrazu wejściowego. Jeżeli którykolwiek ze składowych koloru piksela ma wartość większą od 0 – wszystkie składowe są maksymalizowane.

Parametry

<i>image_in</i>	wskaźnik do początku tablicy na dane wejściowe.
<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.
<i>input_file</i>	wskaźnik na plik wejściowy.
<i>offset_byte</i>	numer bajtu, od którego zaczynają się dane związane z zawartością poszczególnych składowych pikseli.

4.3.2.14 ReadToStructure()

```
void ReadToStructure (
    const BMPFILEHEADER * FileHeader,
    const BITMAPHEADER * Header,
    const FILE * input_file )
```

Funkcja zapisująca do struktury dane nagłówka pliku, zawierające informacje nt. obrazu, m.in. rozdzielczość, wysokość, szerokość, waga pliku, etc.

Parametry

<i>FileHeader</i>	wskaźnik na podstrukturę mówiąca o tym, czy plik rzeczywiście jest plikiem mapy bitowej. Podstruktura mówi m.in. od którego bajtu zaczynają się piksele.
<i>Header</i>	wskaźnik na podstrukturę, której dane opisują wymiary obrazu. Podstruktura zawiera przede wszystkim informacje techniczne obrazu.
<i>input_file</i>	wskaźnik na plik wejściowy.

4.3.2.15 SaveBMPFile()

```
void SaveBMPFile (
    const uint32_t * imgHeight,
    const uint32_t * imgWidth,
    const uint8_t *** image_out,
    const FILE * output_file )
```

Funkcja zapisująca bajty składowych kolorów pikseli do pliku wyjściowego.

Parametry

<i>imgHeight</i>	wskaźnik na wysokość obrazu.
<i>imgWidth</i>	wskaźnik na szerokość obrazu.
<i>image_out</i>	wskaźnik do początku tablicy na dane wyjściowe.
<i>output_file</i>	wskaźnik na plik wyjściowy.

4.3.2.16 WriteFromStructure()

```
void WriteFromStructure (
    const BMPFILEHEADER * FileHeader,
    const BITMAPHEADER * Header,
    const FILE * output_file )
```

Funkcja zapisująca do pliku wyjściowego pozyskane przez funkcję ReadToStructure dane, zapisane do struktury.

Parametry

<i>FileHeader</i>	wskaźnik na podstrukturę mówiąca o tym, czy plik rzeczywiście jest plikiem mapy bitowej. Podstruktura mówi m.in. od którego bajtu zaczynają się piksele.
<i>Header</i>	wskaźnik na podstrukturę, której dane opisują wymiary obrazu. Podstruktura zawiera przede wszystkim informacje techniczne obrazu.
<i>output_file</i>	wskaźnik na plik wyjściowy.

4.4 Dokumentacja pliku main.c

Plik źródłowy.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <inttypes.h>
#include <ctype.h>
#include "bmp.h"
#include "functions.h"
```

Funkcje

- int `main` (int argc, char **argv)

4.4.1 Opis szczegółowy

Plik źródłowy.

Autor

Aleksander Augustyniak

4.4.2 Dokumentacja funkcji

4.4.2.1 main()

```
int main (
    int argc,
    char ** argv )
```


Indeks

AllocateInputMemory

functions.c, [16](#)

functions.h, [24](#)

AllocateOutputMemory

functions.c, [16](#)

functions.h, [25](#)

BITMAPHEADER, [5](#)

bmp.h, [14](#)

colors_importance_rotation, [6](#)

colors_used, [6](#)

comp, [6](#)

height, [6](#)

image_size, [6](#)

numBitPlanes, [6](#)

numBitsPerPlane, [7](#)

reserved, [7](#)

size, [7](#)

width, [7](#)

Xres, [7](#)

Yres, [7](#)

blue

PIXEL, [10](#)

bmp.h, [13](#)

BITMAPHEADER, [14](#)

BMPFILEHEADER, [14](#)

PIXEL, [14](#)

pixel_structure, [14](#)

BMPFILEHEADER, [8](#)

bmp.h, [14](#)

offset, [8](#)

r1, [8](#)

r2, [9](#)

s1, [9](#)

s2, [9](#)

size, [9](#)

change_pixel_color

functions.c, [17](#)

functions.h, [25](#)

check_pixel_color

functions.c, [17](#)

functions.h, [26](#)

closing

functions.c, [17](#)

functions.h, [26](#)

colors_importance_rotation

BITMAPHEADER, [6](#)

colors_used

BITMAPHEADER, [6](#)

comp

BITMAPHEADER, [6](#)

deallocate_sieve

functions.c, [18](#)

functions.h, [26](#)

DeallocateInputMemory

functions.c, [18](#)

functions.h, [27](#)

DeallocateOutputMemory

functions.c, [19](#)

functions.h, [27](#)

functions.c, [15](#)

AllocateInputMemory, [16](#)

AllocateOutputMemory, [16](#)

change_pixel_color, [17](#)

check_pixel_color, [17](#)

closing, [17](#)

deallocate_sieve, [18](#)

DeallocateInputMemory, [18](#)

DeallocateOutputMemory, [19](#)

load_sieve, [19](#)

main_point, [19](#)

opening, [20](#)

operation, [20](#)

ReadPixels, [21](#)

ReadToStructure, [21](#)

SaveBMPFile, [21](#)

WriteFromStructure, [23](#)

functions.h, [23](#)

AllocateInputMemory, [24](#)

AllocateOutputMemory, [25](#)

change_pixel_color, [25](#)

check_pixel_color, [26](#)

closing, [26](#)

deallocate_sieve, [26](#)

DeallocateInputMemory, [27](#)

DeallocateOutputMemory, [27](#)

load_sieve, [28](#)

main_point, [28](#)

opening, [28](#)

operation, [30](#)

ReadPixels, [30](#)

ReadToStructure, [31](#)

SaveBMPFile, [31](#)

WriteFromStructure, [32](#)

green

PIXEL, [10](#)

- height
 - BITMAPHEADER, 6
- image_size
 - BITMAPHEADER, 6
- load_sieve
 - functions.c, 19
 - functions.h, 28
- main
 - main.c, 32
- main.c, 32
 - main, 32
- main_point
 - functions.c, 19
 - functions.h, 28
- next
 - pixel_structure, 11
- numBitPlanes
 - BITMAPHEADER, 6
- numBitsPerPlane
 - BITMAPHEADER, 7
- offset
 - BMPFILEHEADER, 8
- opening
 - functions.c, 20
 - functions.h, 28
- operation
 - functions.c, 20
 - functions.h, 30
- PIXEL, 9
 - blue, 10
 - bmp.h, 14
 - green, 10
 - red, 10
- pixel_structure, 11
 - bmp.h, 14
 - next, 11
 - type, 11
- r1
 - BMPFILEHEADER, 8
- r2
 - BMPFILEHEADER, 9
- ReadPixels
 - functions.c, 21
 - functions.h, 30
- ReadToStructure
 - functions.c, 21
 - functions.h, 31
- red
 - PIXEL, 10
- reserved
 - BITMAPHEADER, 7
- s1
 - BMPFILEHEADER, 9
- s2
 - BMPFILEHEADER, 9
- SaveBMPFile
 - functions.c, 21
 - functions.h, 31
- size
 - BITMAPHEADER, 7
 - BMPFILEHEADER, 9
- type
 - pixel_structure, 11
- width
 - BITMAPHEADER, 7
- WriteFromStructure
 - functions.c, 23
 - functions.h, 32
- Xres
 - BITMAPHEADER, 7
- Yres
 - BITMAPHEADER, 7