Projekty ARKO (x86) – termin A, godz. 12:15

1. Kompresja T4 (RLE i kodowanie Huffmana)
   (*Krzysztof Kalkhoff, Karol Katerżawa*)

   Kompresja T4 (używana np. urządzeniach faksowych) jest przeznaczona dla obrazów czarno-białych, w których każdy piksel jest reprezentowany przez 1 (jeden) bit. Kompresja jest wykonywana linia po linii. Najpierw wyznacza się długości odcinków białych i czarnych pikseli w linii (co ważne, zawsze zaczyna się od odcinka białego). Długości tych odcinków są następnie zapisywane kodem Huffmana. Częstości występowania poszczególnych odcinków, a co za tym idzie ich kody) zostały już wyznaczone – tabele kodów znajdują się na końcu tego dokumentu. Proszę zwrócić uwagę, że kody są różne dla odcinków białych i czarnych (do długości 1728 pikseli). Bardzo istotne jest to, że odcinki o długości mniejszej niż 64 piksele są zapisywane zawsze jako jeden kod (*terminal code*). Przy większych długościach najpierw jest zapisywany kod częściowy (*make-up code*) a następnie kod terminalny. W przypadku linii dłuższych niż 2560 pikseli do zakodowania długości odcinka może być konieczne użycie więcej niż jednego kodu częściowego. Każda linia obrazu po kompresji rozpoczyna się na granicy bajtu – należy w razie potrzeby dostawić stosowną liczbę bitów do strumienia wyjściowego.

   Prototyp funkcji, którą mają Państwo zaimplementować, jest następujący:

   ```
   int T4_Comp (unsigned char * srcptr, int height, int width,
                unsigned char ** dstptr);
   ```

   gdzie:

   srcptr jest wskazaniem na początek pamięci obrazu o strukturze opisanej powyżej

   height jest wysokością obrazu (liczbą linii)

   width jest szerokością obrazu w pikselach

   dstptr jest wskazaniem na wskazanie na bufor skompresowanego obrazu (alokowany w funkcji T4_Comp)

   Wartością funkcji jest rozmiar obrazu po kompresji (w bajtach) lub 0, kiedy kompresja nie powiodła się (jedyną przyczyną jaką widzę może być błąd alokacji pamięci).

2. Dekompresja T4 (kodowanie Huffmana i RLE)
   (*Jakub Ładysz, Maciej Suchecki, Artur Smoleń*)

   Kompresja T4 (używana np. urządzeniach faksowych) jest przeznaczona dla obrazów czarno-białych, w których każdy piksel jest reprezentowany przez 1 (jeden) bit. Dekompresja jest wykonywana linia po linii. Najpierw odczytuje się kody Huffmana długości odcinków białych i czarnych w linii (co ważne, zawsze zaczyna się od odcinka białego). Tabele kodów znajdują się na końcu tego dokumentu. Proszę zwrócić uwagę, że kody są różne dla odcinków białych i czarnych (do długości 1728 pikseli). Bardzo istotne jest to, że odcinki o długości mniejszej niż 64 piksele są zapisywane zawsze jako jeden kod (*terminal code*). Przy większych długościach najpierw jest zapisywany kod częściowy (*make-up code*) a następnie kod terminalny. W przypadku linii dłuższych niż 2560 pikseli do zakodowania długości odcinka może być konieczne użycie więcej niż jednego kodu częściowego. Każda linia obrazu po kompresji

rozpoczyna się na granicy bajtu – należy w razie potrzeby pominąć stosowną liczbę bitów ze strumienia wejściowego. Po ustaleniu długości odcinka (lub odcinków) stosowną liczbę pikseli odpowiedniego koloru należy zapisać w obrazie wyjściowym.

Prototyp funkcji, którą mają Państwo zaimplementować, jest następujący:

```
int T4_Decomp (unsigned char * srcptr, int height, int width,
               unsigned char * dstptr);
```

gdzie:

srcptr jest wskazaniem na skompresowany zgodnie z powyższym opisem obraz

height jest wysokością obrazu (liczbą linii)

width jest szerokością obrazu w pikselach

dstptr jest wskazaniem na bufor obrazu (alokowany poza funkcją T4_Decomp)

Wartością funkcji jest 0, kiedy dekompresja powiodła się, i kod błędu w przeciwnym przypadku.

3. Filtr splotowy
(*Andrzej Niedźwiedź, Dawid Kaczmarek*)

   Napisać funkcję o prototypie w języku C:

```
uchar* ConvFilter(ImgDesc* pImg,
      int maskSize, int* mask, uchar* dstimg);
```

   gdzie:
   `pImg` jest wskazaniem na deskryptor obrazu, zawierający w szczególności informację o szerokości i wysokości obrazu oraz wskazanie na bufor obrazu źródłowego
   `dstimg` jest wskazaniem na wyjściowy bufor obrazu, w którym zostanie zapisany obraz po filtracji
   `maskSize` jest rozmiarem maski filtru splotowego
   `mask` jest wskazaniem na współczynniki wagowe maski (jest ich maskSize * maskSize).

   Wartością funkcji jest wskazanie na bufor obrazu wyjściowego.

   Uwaga: w przypadku pikseli „skrajnych”, dla których maska wystaje poza obraz źródłowy, należy przepisać te piksele źródłowe do obrazu wyjściowego bez zmian.

   Odczyt i zapis plików graficznych jest wykonywany poza zasadniczą funkcją. Szerokość i wysokość obrazu nie musi być odczytywana z pliku (podobnie, nie ma wymagania, żeby nazwę pliku odczytywać z konsoli) – można te wartości zawrzeć w kodzie programu testującego. Nie ma wymagań dotyczących formatu pliku, choć najprostszym wydaje mi się format BMP.

4. Filtr medianowy
(*Tomasz Szleter, Michał Stankiewicz*)

   Napisać funkcję o prototypie w języku C:

```
uchar* MedFilter(ImgDesc* pImg, int maskSize, uchar* dstimg);
```

gdzie:
`pImg` jest wskazaniem na deskryptor obrazu, zawierający w szczególności informację o szerokości i wysokości obrazu oraz wskazanie na bufor obrazu źródłowego
`dstimg` jest wskazaniem na wyjściowy bufor obrazu, w którym zostanie zapisany obraz po filtracji
`maskSize` jest rozmiarem maski filtru medianowego.
Wartością funkcji jest wskazanie na bufor obrazu wyjściowego.

Uwaga: w przypadku pikseli „skrajnych", dla których maska wystaje poza obraz źródłowy, należy przepisać te piksele źródłowe do obrazu wyjściowego bez zmian.

Odczyt i zapis plików graficznych jest wykonywany poza zasadniczą funkcją. Nie ma wymagań dotyczących formatu pliku, choć najprostszym wydaje mi się format BMP.

5. Rysowanie wielokąta kolorowego
(*Michał Piekarczyk, Adam Jakubowski*)

Napisać funkcję o prototypie w języku C:

```
uchar* DrawTriangle(ImgDesc* pImg, Vertex* triangle[3]);
```

gdzie:
`pImg` jest wskazaniem na deskryptor obrazu, zawierający w szczególności informację o szerokości i wysokości obrazu oraz wskazanie na bufor obrazu źródłowego
`triangle` jest tablicą wskazań na rekordy opisujące punkty trójkąta (struktura punktu zawiera współrzędne (x,y) oraz jego kolor (r,g,b)). Można przyjąć, że trójkąt jest w całości zawarty w obrazie.
Wartością funkcji jest wskazanie na bufor obrazu, w którym został narysowany trójkąt.

Odczyt i zapis plików graficznych jest wykonywany poza zasadniczą funkcją. Nie ma wymagań dotyczących formatu pliku, choć najprostszym wydaje mi się format BMP.

6. Kopiowanie prostokątnych obszarów obrazu czarno-białego
(*Michał Kraluk, Łukasz Marcinkowski*)

Napisać funkcję o prototypie w języku C:

```
ImageDesc* CopyRect(ImgDesc* pImg, Rect* pRect, Point* pDst);
```

gdzie

`pImg` jest wskazaniem na deskryptor obrazu, zawierający w szczególności informację o szerokości i wysokości obrazu oraz wskazanie na bufor obrazu źródłowego
`pRect` jest wskazaniem na strukturę opisującą prostokąt źródłowy (ten, z którego pochodzą kopiowane piksele)

`pDst` wskazanie na strukturę punktu wskazującą, w którym miejscu obrazu powinien znaleźć się kopiowany prostokąt obrazu.

Wartością funkcji jest wskazanie na źródłowy deskryptor obrazu (pImg).

Koniecznie proszę pamiętać, że obraz jest czarno-biały, tzn. każdy piksel obrazu jest zapisany na **jednym** bicie. Bardzo istotne jest rozsądne postępowanie w sytuacji, kiedy obszary źródłowy i docelowy zachodzą na siebie lub gdy prostokąt źródłowy „przeniesiony" na pozycję docelową nie mieści się w całości w obrazie.

Odczyt i zapis plików graficznych jest wykonywany poza zasadniczą funkcją. Nie ma wymagań dotyczących formatu pliku, choć najprostszym wydaje mi się format BMP.

7. Wypełnianie zalewowe
(*Jakub Borowski, Jacek Witkowski*)

   Napisać funkcję o prototypie w języku C:

   ```
   uchar* FloodFill(ImgDesc* pImg, int x, int y, int rgbFill);
   ```

   gdzie:
   `pImg` jest wskazaniem na deskryptor obrazu, zawierający w szczególności informację o szerokości i wysokości obrazu oraz wskazanie na bufor obrazu źródłowego
   `(x,y)` są współrzędnymi punktu ziarna, od którego rozpoczyna się wypełnianie obrazu (kolor tego punktu należy przyjąć jako kolor tła)
   `rgbFill` jest wartością koloru wypełnienia.
   Wartością funkcji jest wskazanie na bufor obrazu, w którym zostało wykonane wypełnianie.

   Odczyt i zapis plików graficznych jest wykonywany poza zasadniczą funkcją. Nie ma wymagań dotyczących formatu pliku, choć najprostszym wydaje mi się format BMP.

TABLE 2/T.6

Terminating codes

White run length	Code word	Black run length	Code word
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	11	0000101
12	001000	12	0000111
13	000011	13	00000100
14	110100	14	00000111
15	110101	15	000011000
16	101010	16	0000010111
17	101011	17	0000011000
18	0100111	18	0000001000
19	0001100	19	00001100111
20	0001000	20	00001101000
21	0010111	21	00001101100
22	0000011	22	00000110111
23	0000100	23	00000101000
24	0101000	24	00000010111
25	0101011	25	00000011000
26	0010011	26	000011001010
27	0100100	27	000011001011
28	0011000	28	000011001100
29	00000010	29	000011001101
30	00000011	30	000001101000
31	00011010	31	000001101001
32	00011011	32	000001101010
33	00010010	33	000001101011
34	00010011	34	000011010010
35	00010100	35	000011010011
36	00010101	36	000011010100
37	00010110	37	000011010101
38	00010111	38	000011010110
39	00101000	39	000011010111
40	00101001	40	000001101100
41	00101010	41	000001101101
42	00101011	42	000011011010
43	00101100	43	000011011011
44	00101101	44	000001010100
45	00000100	45	000001010101
46	00000101	46	000001010110
47	00001010	47	000001010111
48	00001011	48	000001100100
49	01010010	49	000001100101
50	01010011	50	000001010010
51	01010100	51	000001010011
52	01010101	52	000000100100
53	00100100	53	000000110111
54	00100101	54	000000111000
55	01011000	55	000000100111
56	01011001	56	000000101000
57	01011010	57	000001011000
58	01011011	58	000001011001
59	01001010	59	000000101011
60	01001011	60	000000101100
61	00110010	61	000001011010
62	00110011	62	000001100110
63	00110100	63	000001100111

TABLE 3/T.6

Make-up codes between 64 and 1728

White run length	Code word	Black run length	Code word
64	11011	64	0000001111
128	10010	128	000011001000
192	010111	192	000011001001
256	0110111	256	000001011011
320	00110110	320	000000110011
384	00110111	384	000000110100
448	01100100	448	000000110101
512	01100101	512	0000001101100
576	01101000	576	0000001101101
640	01100111	640	0000001001010
704	011001100	704	0000001001011
768	011001101	768	0000001001100
832	011010010	832	0000001001101
896	011010011	896	0000001110010
960	011010100	960	0000001110011
1024	011010101	1024	0000001110100
1088	011010110	1088	0000001110101
1152	011010111	1152	0000001110110
1216	011011000	1216	0000001110111
1280	011011001	1280	0000001010010
1344	011011010	1344	0000001010011
1408	011011011	1408	0000001010100
1472	010011000	1472	0000001010101
1536	010011001	1536	0000001011010
1600	010011010	1600	0000001011011
1664	011000	1664	0000001100100
1728	010011011	1728	0000001100101

Make-up codes between 1792 and 2560

Run length (black and white)	Make-up codes
1792	00000001000
1856	00000001100
1920	00000001101
1984	000000010010
2048	000000010011
2112	000000010100
2176	000000010101
2240	000000010110
2304	000000010111
2368	000000011100
2432	000000011101
2496	000000011110
2560	000000011111