# Grafika Komputerowa i Multimedia

## Projekt nr.6

„Należy utworzyć specyfikację rastrowego pliku graficznego rejestrującego obraz kolorowy (z wykorzystaniem 32 narzuconych i 32 dedykowanych barw) i w 32 stopniowej skali szarości we wszystkich przypadkach opierającego się na kompresji LZW. Alfabet wejściowy to wartość 5 bitowa. Należy napisać aplikacje, które zgodnie ze stworzoną specyfikacją dokonają filtracji danych wejściowych(przystosowanie danych do alfabetu wejściowego) i konwersji z pliku BMP do nowego rodzaju pliku graficznego oraz z nowego formatu pliku do rodzaju BMP. Użytkownik powinien mieć możliwość m.in. wyboru jednego z trzech trybów barwnych (paleta narzucona, paleta dedykowana, skala szarości).”

| | |
|---|---|
| Wydział | Wydział Inżynierii Elektrycznej i Komputerowej |
| Kierunek | Informatyka |
| Rok | 2 |
| Wykonawcy | Bernard Pigan<br>Dominik Tamiołło<br>Sebastian Smulski<br>Mateusz Młodochowski |
| Grupa | 22i |
| Data | 9.01.2020 |

# Specyfikacja

Nagłówek pliku

| Nazwa | Wartości | Rozmiar |
|---|---|---|
| **Wartość magiczna** | „BSDM” | 32 bity (4 bajty) |
| **Szerokość** | $0\text{-}2^{32}$ | 32 bity (4 bajty) |
| **Wysokość** | $0\text{-}2^{32}$ | 32 bity (4 bajty) |
| **Bity na pixel** | 5 lub 24 (w zależności od wyboru palety) | 8 bitów (1 bajt) |
| **Tryb koloru** | 0 – kolor<br>1 – skala szarości | 8 bitów (1 bajt) |
| **Tryb palety** | 1 - standardowa<br>0 - dedykowana | 8 bitów (1 bajt) |
| **Wielkość nagłówka** | 20 bajtów | 32 bity (4 bajty) |
| **Długość słowa LZW** | 0-255 | 8 bitów (1 bajt) |
| **Ilość kolorów w palecie (opcjonalne)** | 1-32 | 8 bitów (1 bajt) |

Rozszerzenie pliku: .bsdm

Na kolor każdego pixela jest przeznaczone 5 bitów(przy palecie standardowej) w konfiguracji RRGGB lub 24 bity (przy palecie dedykowanej). Obraz zapisywany jest od lewego górnego rogu po szerokości. Dane są kodowane przy użyciu algorytmu LZW. Podczas kompresji długość słowa, które jest zapisywane do pliku, zmienia się, dlatego jest zapisana w nagłówku.

Paleta jest zapisywana tylko gdy obraz jest tworzony z palety dedykowanej. Jest ona w całości zapisywana w pliku. Zatem maksymalnie 32 kolory po 24 bity na kolor dają łącznie 96 bajtów. Dla każdego pixela zapisujemy numer koloru, a nie sam kolor. Długość słowa również zależy od kompresji LZW.

Paleta szarości posiada 32 odcienie (pokazana w dalszej części specyfikacji), które są zapisane na 24 bitach każdy. Jest zaimplementowana w programie i nie jest zapisywana do pliku. Dla każdego pixela, do pliku, są zapisywane tylko indeksy palety, a nie wartości konkretnych kolorów. Długość słowa również zależy od kompresji LZW.

# Paleta standardowa:


# Paleta dedykowana:

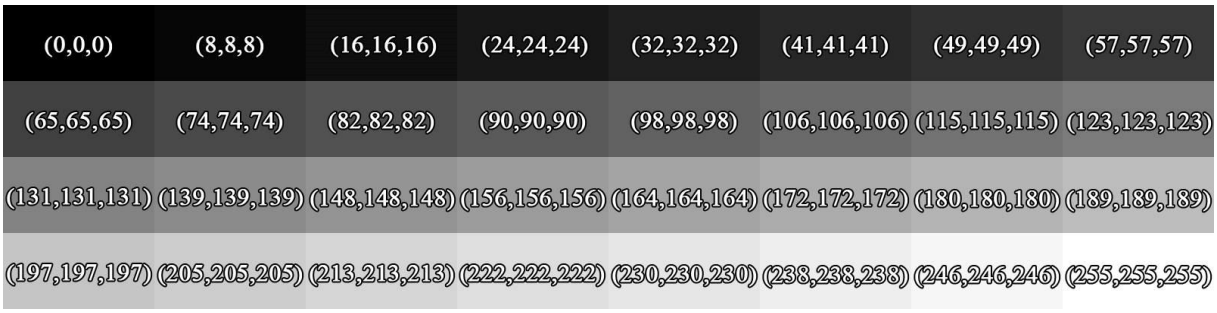
| Indeks | Wartość koloru |
|---|---|
| 0 | $Y_0$ |
| 1 | $Y_1$ |
| 2 | $Y_2$ |
| ... | ... |
| n | $Y_n$ |

Gdzie: $0 \leq n \leq 31$

# Paleta szarości:

Do obliczenia iluminacji piksela w 256 stopniowej skali szarości używamy wzoru:

$$Y = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B$$



# Kompresja LZW:

W skróconym opisie działanie algorytmu LZW polega na pobraniu liczby z podanego ciągu, dołączeniu jej do aktualnego słowa algorytmu i sprawdzenie czy podane słowo istnieje w słowniku, który został podany na początku działania algorytmu. Jeśli podane słowo nie istnieje zostaje dodany na koniec słownika, a słowo przed dodaniem znaku z ciągu wejściowego zostaje wypisane, jeśli jednak konkretne słowo istnieje w słowniku pobrany jest kolejny znak z ciągu wejściowego, który dołączamy do słowa sprawdzanego przez algorytm.

Lista kroków:

1. Wypełnij słownik alfabetem źródła informacji.
2. **c** := *pierwszy symbol wejściowy*
3. Dopóki są dane na wejściu:
   - Wczytaj znak **s**.
   - Jeżeli ciąg **c + s** znajduje się w słowniku, przedłuż ciąg **c**, tj. **c** := **c** + **s**
   - Jeśli ciągu **c + s** nie ma w słowniku, wówczas:
     - wypisz kod dla **c** (**c** znajduje się w słowniku)
     - dodaj ciąg **c + s** do słownika
     - przypisz **c** := **s**.
4. Na końcu wypisz na wyjście kod związany **c**.

Przykład działania:

Dla alfabetu wejściowego: 0 1 2 3 4 5 6 7 8 9

oraz ciągu wejściowego: 7539898753075398

kompresja wygląda w następujący sposób:

| Znak | Słowo | Słownik? | | Do słownika | | Wyjście |
|---|---|---|---|---|---|---|
| **7** | | 7 | Tak | | | |
| **5** | 7 | 75 | Nie | 75 | [10] | 7 |
| **3** | 5 | 53 | Nie | 53 | [11] | 5 |
| **9** | 3 | 39 | Nie | 39 | [12] | 3 |
| **8** | 9 | 98 | Nie | 98 | [13] | 9 |
| **9** | 8 | 89 | Nie | 89 | [14] | 8 |
| **8** | 9 | 98 | Tak | | | |
| **7** | 98 | 987 | Nie | 987 | [15] | [13] |
| **5** | 7 | 75 | Tak | | | |
| **3** | 75 | 753 | Nie | 753 | [16] | [10] |
| **0** | 3 | 30 | Nie | 30 | [17] | 3 |
| **7** | 0 | 07 | Nie | 07 | [18] | 0 |
| **5** | 7 | 75 | Tak | | | |
| **3** | 75 | 753 | Tak | | | |
| **9** | 753 | 7539 | Nie | 7539 | [19] | [16] |
| **8** | 9 | 98 | Tak | | | |
| **KONIEC** | 98 | 98 | Tak | | | [13] |

Wynik końcowy: 7,5,3,9,8,[13],[10],3,0,[16],[13]

W naszym przypadku alfabet wejściowy składa się z 32 liczb, od 0 do 31. Jest to spowodowane zapisem kolorów na 5 bitach(maksymalna wartość zapisana na 5 bitach wynosi 31) oraz fakt, że paleta dedykowana posiada 32 kolory (kompresowane są indeksy kolorów z palety, a nie konkretne kolory).

Przykład działania na 5-bitowej palecie kolorów

Alfabet wejściowy : 0-31

Ciąg wejściowy: 24 31 11 04 07 09 24 31 09 24 17 18 29 31 11 04

| Znak | Słowo | Słownik? | | Do słownika | | Wyjście |
|---|---|---|---|---|---|---|
| **24** | | 24 | Tak | | | |
| **31** | 24 | 24 31 | Nie | 24 31 | [32] | 24 |
| **11** | 31 | 31 11 | Nie | 31 11 | [33] | 31 |
| **04** | 11 | 11 04 | Nie | 11 04 | [34] | 11 |
| **07** | 04 | 04 07 | Nie | 04 07 | [35] | 04 |
| **09** | 07 | 07 09 | Nie | 07 09 | [36] | 07 |
| **24** | 09 | 09 24 | Nie | 09 24 | [37] | 09 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **31** | 24 | 24 31 | Tak | | | |
| **09** | 24 31 | 24 31 09 | Nie | 24 31 09 | [38] | [32] |
| **24** | 09 | 09 24 | Tak | | | |
| **17** | 09 24 | 09 24 17 | Nie | 09 24 17 | [39] | [37] |
| **18** | 17 | 17 18 | Nie | 17 18 | [40] | 17 |
| **29** | 18 | 18 29 | Nie | 18 29 | [41] | 18 |
| **31** | 29 | 29 31 | Nie | 29 31 | [42] | 29 |
| **11** | 31 | 31 11 | Tak | | | |
| **04** | 31 11 | 31 11 04 | Nie | 31 11 04 | [43] | [33] |
| **KONIEC** | 04 | 04 | Tak | | | 04 |

Ciąg końcowy: 24 31 11 04 07 09 [32] [37] 17 18 29 [33] 04

Ciąg skrócił się z 16 liczb do 13.

Dekompresja LZW:

Lista kroków:

1. Wypełnij słownik alfabetem źródła informacji.
2. **pk** := *pierwszy kod skompresowanych danych*
3. Wypisz na wyjście ciąg związany z kodem **pk**, tj. słownik[**pk**]
4. Dopóki są jeszcze jakieś słowa kodu:
   - Wczytaj kod **k**
   - **pc** := słownik[**pk**] – ciąg skojarzony z poprzednim kodem
   - Jeśli słowo **k** jest w słowniku, dodaj do słownika ciąg (**pc** + pierwszy symbol ciągu słownik[**k**]), a na wyjście wypisz cały ciąg słownik[**k**].
   - W przeciwnym razie (przypadek **scscs**) dodaj do słownika ciąg (**pc** + pierwszy symbol **pc**) i tenże ciąg wypisz na wyjście.
   - **pk** := **k**

Przykład odkodowania ciągu:

Alfabet: 0-31

Ciąg wejściowy: 24 31 11 04 07 09 [32] [37] 17 18 29 [33] 04 (ciąg, który został zakodowany w poprzednim przykładzie)

| Kod | Słowo | Znak | Do słownika | | Wyjście |
|---|---|---|---|---|---|
| **24** | 24 | | | | 24 |
| **31** | 24 | 31 | 24 31 | [32] | 31 |
| **11** | 31 | 11 | 31 11 | [33] | 11 |
| **04** | 11 | 04 | 11 04 | [34] | 04 |
| **07** | 04 | 07 | 04 07 | [35] | 07 |
| **09** | 07 | 09 | 07 09 | [36] | 09 |
| **[32]** | 09 | 24 | 09 24 | [37] | 24 31 |
| **[37]** | [32] | 09 | 24 31 09 | [38] | 09 24 |
| **17** | [37] | 17 | 09 24 17 | [39] | 17 |
| **18** | 17 | 18 | 17 18 | [40] | 18 |
| **29** | 18 | 29 | 18 29 | [41] | 29 |
| **[33]** | 29 | 31 | 29 31 | [42] | 31 11 |
| **04** | [33] | 04 | 31 11 04 | [43] | 04 |

Ciąg końcowy: 24 31 11 04 07 09 24 31 09 24 17 18 29 31 11 04