Wprowadzenie do programowania w języku C
grupa RKr, wtorek 16:15-18:00
lista nr 10 *(na pracownię 17.12.2019) (wersja 2)*

Zadanie 1. [15p na pracowni lub 10p po pracowni**]**

Napisz funkcję "**MemDescriptor loadBMP**(**const char*** bmpFileIn)", która wczyta plik w formacie BMP (v3). Napisz także funkcję "**void saveBMP**(**const char*** bmpFileOut, **MemDescriptor** m)", która zapisze plik w formacie bmp. Odczyt i zapis powinien być w trybie binarnym, a więc użyj funkcji *fopen(.)* z odpowiednimi parametrami. Do odczytu i zapisu będą potrzebne funkcje *fread(.)* i *fwrite(.)* oraz *fclose(.)*, można też użyć *fseek(.)* do ominięcia nagłówka BMP, etc. Zmodyfikuj typ **MemDescriptor** jeśli potrzeba, np. dodaj obsługę wymiarów obrazu i innych informacji, dynamicznej pamięci, etc.

Napisz funkcję „**MemDescriptor outtakeImg**(**MemDescriptor** img, **int** xA, **int** yA, **int** xB, **int** yB)", która stworzy nowszy obraz poprzez wycięcie z img prostokąta opisanego punktami (xA, yA) i (xB, yB). Format BMP wymaga aby jeden wiersz w pliku miał rozmiar (w bajtach) podzielny przez 4. Możesz założyć, że szerokość okna rozpinanego przez punkty A i B (ich odległość na osi X zwiększona o 1) jest podzielna przez 4. Oznacza to wprost, że liczba pikseli w wierszu będzie podzielna przez 4, co pozwoli uniknąć manualnego dodawania paddingu za ostatnim pikselem w każdym wierszu.

Wszystkie funkcje zaimplementuj w pliku bmpFiddler.c, stwórz także plik nagłówkowy bmpFiddler.h, zaprogramowane w takim sposób funkcje zawołaj/wykorzystaj w pliku main.c.
W SKOS znajdziesz plik „**JokerHarley**.bmp", wczytaj go, przytnij i zapisz jako „**JokerHarley_outtake**.bmp".
W SKOS dostępne są też definicje i funkcje pomocnicze dla formatu BPM, w pliku pC19gRK_lista10_toys.txt.
Na drugiej można obejrzeć nagłówek formatu BMP (dla wersji 3) z kolorowaniem składni.
- - - - - - -
Wczytaj obraz bmp o kształcie kwadratu, który ma 24bpp (bits-per-pixel), czyli jest bez kanału alpha. Zaokrąglij wymiary do najbliższych potęg dwójki, używając adaptacji makra ROUND_UP_toPOT(.) [lista 5]. Adaptacja jest w sensie liczby obsługiwanych bitów, zamiast 6, potrzebujemy 11 (dla rozmiaru 1200). Uzyskany z zaokrąglania padding wypełnij jakimś kolorem lub drobną szachownicą o polu 4x4 lub 8x8 pikseli. Następnie zmień układ pamięci obrazu, używając funkcji layToZTiled(.) [lista 9], modyfikując ją do obsługi innej liczby bitów. Podziel obraz na kwadranty korzystając z właściwości układu ZTiled, każdy kwadrant zapisz do nowego pliku. Jeśli oryginalny wymiar obrazu był niewiele większy od potęgi dwójki, to jego nowy wymiar będzie około dwukrotnie większy, a cała powierzchnia obrazu będzie około czterokrotnie większa. W takiej sytuacji możesz wykonywać operacje na najbardziej wypełnionym kwadrancie zamiast na całym obrazie. Wczytaj jeden kwadrant, a następnie zmień układ na Lined i zapisz do nowego pliku. Zwróć uwagę na to jaki układ pamięci ma każdy kwadrant w momencie tworzenia/wydzielania. Sprawdź w przeglądarce obrazów, czy finalny kwadrant jest poprawny.

W SKOS znajdziesz plik „**AmicorumSpectaculum**.bmp" posiadający formę kwadratu o wymiarach 1200x1200, wykonaj na nim opisane operacje.

Zadanie 2. [15p**]** Dostępne w serwisie SKOS.

```
typedef unsigned char  byte;
typedef signed char    si1B;
typedef signed short   si2B;
typedef signed int     si4B;

#pragma pack(push, 1)
typedef                 // ## BMPv3 HEADER ###################### //
  struct BmpHeaderTAG   // sizeof(BmpHeader) must be 54 [Byte]    //
{
//// FILE HEADER      //// the very basic (main) file header    ////
  si1B id[2];           /* 0x4D42 must be here, meaning 'B'and'M' */
  si4B fileSize;        /* (xDim * yDim * 3) + 54L + padding      */
  si2B reserved[2];     /* ((ignored))                            */
  si4B dataOffset;      /* 54 [Byte] ((expected))                 */
//// DIB HEADER       //// 'device-independent bitmap' header   ////
  si4B DIBHeaderSize;   /* should be 40 for us *//* 14 + 40 = 54  */
                        /* version-wise values                    */
                        /*   12: (BMPv2) Windows 2.x + OS/2 1.x   */
                        /*   40: (BMPv3) Windows 3.x + Windows NT */
                        /*  108: (BMPv4) Windows 4.x (Windows 95) */
  si4B pixWidth;        /* xDim, horizontal size ((variable))     */
  si4B pixHeight;       /* yDim, vertical ((variable))            */
  si2B biPlanes;        /*  1 assume((fixed)) */
  si2B bitsPerPixel;    /* 24 assume((fixed)) */
  si4B biCompression;   /*  0 assume((fixed)) */
  si4B biImageByteSize; /* ((ignored)) */
  si4B biXPixPerMeter;  /* ((ignored)) */
  si4B biYPixPerMeter;  /* ((ignored)) */
  si4B biClrUsed;       /* ((ignored)) */
  si4B biClrImportant;  /* ((ignored)) */
}
#if defined(__linux__)
  __attribute__((__packed__, aligned(1)))
#endif
  BmpHeader;
#pragma pack(pop)
```