Отчет по лабораторной работе №1

«BMP конвертор»

по дисциплине  
«ЭВМ и периферические устройства»

|  |  |
| --- | --- |
| Выполнили:  студенты гр. 3530904/80004 | Захарова А. А.  Федоров Д. В.  Селезнев В. А. |
| Преподаватель | А. В. Петров |
|  |  |

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2020 г.

**Задание на выполнение работы**

1. Напишите программу тестирования входного файла. Выведите размеры и число бит на пиксел (8 - 24 бита).

Используйте структуры для тестирования.

2. Напишите программу-конвертор BMP форматов (24-16, 24-8, 24-4 или другое). При бинаризации изображения (24->1) необходимо задание уровня порога. У каждого студента должны быть свои форматы данных

**Код программы**

#include <windows.h>

#include <iostream>

#include <string>

using namespace std;

void main()

{

string sFileName;

BITMAPFILEHEADER bmpFileHeader;

BITMAPINFOHEADER bmpInfoHeader;

int Width, Height;

RGBQUAD Palette[256];

RGBTRIPLE\* inBuf;

BYTE\* outBuf;

HANDLE hInputFile, hOutFile;

DWORD RW;

cout << "Enter the full name, please: ";

cin >> sFileName;

hInputFile = CreateFile(sFileName.c\_str(), GENERIC\_READ,

FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, 0, NULL);

if (hInputFile == INVALID\_HANDLE\_VALUE)

return;

int edge;

cout << "Enter the edge for picture binarisation: ";

cin >> edge;

hOutFile = CreateFile("Result.bmp", GENERIC\_WRITE, 0,

NULL, CREATE\_NEW, 0, NULL);

if (hOutFile == INVALID\_HANDLE\_VALUE)

{

CloseHandle(hInputFile);

return;

}

// Считываем инфу

ReadFile(hInputFile, &bmpFileHeader,

sizeof(bmpFileHeader), &RW, NULL);

ReadFile(hInputFile, &bmpInfoHeader,

sizeof(bmpInfoHeader), &RW, NULL);

// Установим указатель на начало растра

SetFilePointer(hInputFile, bmpFileHeader.bfOffBits, NULL,

FILE\_BEGIN);

Width = bmpInfoHeader.biWidth;

Height = bmpInfoHeader.biHeight;

// Выделим память

inBuf = new RGBTRIPLE[Width];

outBuf = new BYTE[Width];

// Заполним заголовки

bmpFileHeader.bfOffBits = sizeof(bmpFileHeader) + sizeof

(bmpInfoHeader) +1024;

bmpInfoHeader.biBitCount = 8;

bmpFileHeader.bfSize = bmpFileHeader.bfOffBits + Width \*

Height + Height \* (3 \* Width % 4);

// Запишем заголовки

WriteFile(hOutFile, &bmpFileHeader,

sizeof(bmpFileHeader), &RW, NULL);

WriteFile(hOutFile, &bmpInfoHeader,

sizeof(bmpInfoHeader), &RW, NULL);

// Палитра черно-белая

for (int i = 0; i < 256; i++)

{

Palette[i].rgbBlue = i;

Palette[i].rgbGreen = i;

Palette[i].rgbRed = i;

}

WriteFile(hOutFile, Palette, 256 \* sizeof(RGBQUAD), &RW,

NULL);

// Начнем преобразовывать

for (int i = 0; i < Height; i++)

{

ReadFile(hInputFile, inBuf, sizeof(RGBTRIPLE) \*

Width, &RW, NULL);

for (int j = 0; j < Width; j++)

{

outBuf[j] = (inBuf[j].rgbtRed + inBuf[j].rgbtGreen + inBuf[j].rgbtBlue) / 3 < edge ? 0 : 255;

}

WriteFile(hOutFile, outBuf, sizeof(BYTE) \* Width,

&RW, NULL);

// Пишем мусор для выравнивания

WriteFile(hOutFile, Palette, (3 \* Width) % 4, &RW,

NULL);

SetFilePointer(hInputFile, Width % 4, NULL,

FILE\_CURRENT);

}

delete[] inBuf;

delete[] outBuf;

CloseHandle(hInputFile);

CloseHandle(hOutFile);

cout << "Updating has come to the end successfully!";

}

**Результат работы программы**

