

Лабораторна робота №2 (2-й семестр)

Теоретичні відомості

BMP (від англ. Bitmap Picture) — формат зберігання растрових зображень.

З форматом **BMP** працює величезна кількість програм, так як його підтримка інтегрована в операційні системи Windows і OS/2. Файли формату **BMP** можуть мати розширення .bmp, .dib і .rle. Крім того, дані цього формату включаються в двійкові файли ресурсів RES і в PE-файли.

Глибина кольору в даному форматі може бути 1, 2, 4, 8, 16, 24, 32, 48 біт на піксель, максимальні розміри зображення 65535 × 65535 пікселів. Однак, глибина 2 біт офіційно не підтримується.

У форматі **BMP** є підтримка стиснення по алгоритму RLE, однак тепер існують формати з більш сильним стисненням, і через великий розмір **BMP** рідко використовується в Інтернеті, де для стиснення без втрат використовуються PNG і GIF.

BMP-файл складається з чотирьох частин:

1. Заголовок файлу (**BITMAPFILEHEADER**)
2. Заголовок зображення (**BITMAPINFOHEADER**, може бути відсутнім).
3. Палітра (може бути відсутня).
4. Зображення.

Структура **BITMAPFILEHEADER** містить інформацію про тип, розмір і представлення даних в файлі:

```
struct BitmapFileHeader {  
    WORD    bfType; // тип файла, символи «BM» (0x424d),  
                // тип WORD - unsigned short  
                // тип DWORD - unsigned long  
    DWORD   bfSize; // розмір всього файлу в байтах.  
    WORD    bfReserved1; // зарезервовані,  
    WORD    bfReserved2; // повинні містити нулі  
    DWORD   bfOffBits; // містить зміщення в байтах від початку  
//структури BITMAPFILEHEADER до безпосередньо битів зображення  
};
```

Найбільш простий варіант заголовку зображення **BITMAPINFOHEADER**:

```
struct BitmapInfoHeader {  
    DWORD   biSize; // розмір структури в байтах  
    LONG    biWidth; // ширина зображення в пікселях  
    LONG    biHeight; // висота зображення в пікселях  
    WORD    biPlanes; // містить одиницю  
    WORD    biBitCount; // число біт на піксель  
    DWORD   biCompression; // тип стиснення (0— без стиснення)  
    DWORD   biSizeImage; // розмір зображення в байтах  
    LONG    biXPelsPerMeter; // горизонтальна роздільна здатність в  
// пікселях на метр для цільового пристрою  
// тип LONG - long int  
    LONG    biYPelsPerMeter; // вертикальна роздільна здатність в  
// пікселях на метр для цільового пристрою  
    DWORD   biClrUsed; // кількість застосованих кольорових індексів в  
// палітрі
```

```
DWORD biClrImportant; // кількість індексів, необхідних
// для відображення зображення
};
```

Кольори в **.bmp** файлі задаються у форматі RGB (Red-Green-Blue) . Формат RGB- це набір правил, за допомогою яких будь-який колір можна представити у вигляді певного коду цифр і букв.

Формат RGB - це вказівка комп'ютеру як змішувати червоні, зелений і синій колір в різних поєднаннях, щоб отримати всі кольори веселки (див. рис.1). Кожен колір: червоний, зелений або синій, характеризується його інтенсивністю або насиченістю.

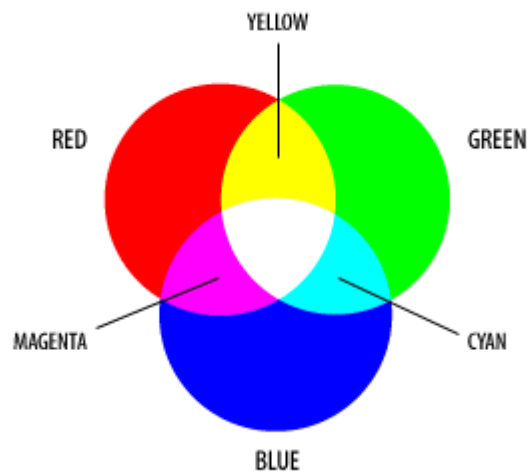


Рисунок 1.

Кількість кожного кольору може лежати в діапазоні від 0 до 255.

Абсолютно червоний колір матиме форму записи (255,0,0). Це означає, що кількість червоного кольору 255, зеленого 0 (тобто зеленої складової немає), синій 0 (синьої складової немає).

Абсолютно синій колір (0,255,0) і зелений (0,0,255).

При різних комбінаціях, починають вже утворюватися різні кольори веселки:

яскраво-фіолетовий - (255,0,255), чорний - (0,0,0)

Це форма запису (255,0,255) у вигляді десяткових чисел. Але колір RGB можна також представити у вигляді 16-річної системи. Такими числами легше оперувати комп'ютеру.

Якщо перетворити по черзі, кожне з чисел, яке відповідає певному кольору, в 16-річну систему, то ми отримаємо іншу форму запису кольору.

FFFFFF - (255,255,255) - білий колір

Де FF - число 255 в 16-річній системі числення.

000000 - (0,0,0) - чорний колір

Тобто колір в форматі RGB можна представити як в 16-річній, так і в 10-чній системі числення (див.рис.2).

Знак # повідомляє про те, що використовується саме 16-річна система.

У мові C ++ для подання 16-річних чисел # замінюється на **0x**

Named	Numeric	Color name	Hex rgb	Decimal
		<i>black</i>	#000000	0,0,0
		<i>silver</i>	#C0C0C0	192,192,192
		<i>gray</i>	#808080	128,128,128
		<i>white</i>	#FFFFFF	255,255,255
		<i>maroon</i>	#800000	128,0,0
		<i>red</i>	#FF0000	255,0,0
		<i>purple</i>	#800080	128,0,128
		<i>fuchsia</i>	#FF00FF	255,0,255
		<i>green</i>	#008000	0,128,0
		<i>lime</i>	#00FF00	0,255,0
		<i>olive</i>	#808000	128,128,0
		<i>yellow</i>	#FFFF00	255,255,0
		<i>navy</i>	#000080	0,0,128
		<i>blue</i>	#0000FF	0,0,255
		<i>teal</i>	#008080	0,128,128
		<i>aqua</i>	#00FFFF	0,255,255

Рисунок 2. Деякі стандартні кольори, якими може оперувати комп'ютер.

Приклад. Написати консольний додаток, в якому користувач має можливість задати ширину і висоту зображення; залити зображення червоним кольором і зберегти зображення у вигляді bmp-файлу (рис.3).

```
#include <fstream>
#include <iostream>

using namespace std;

typedef uint16_t WORD;
typedef uint32_t DWORD;
typedef int32_t LONG;
//щоб уникнути округлення розміру структури до максимального
//4х-байтного розміру
#pragma pack(push,1)
struct __attribute__((__packed__)) BitmapFileHeader {
    WORD    bfType;
    DWORD   bfSize;
    WORD    bfReserved1;
    WORD    bfReserved2;
    DWORD   bfOffBits;
};
#pragma pack(push,1)
struct __attribute__((__packed__)) BitmapInfoHeader {
    DWORD   biSize;
    LONG    biWidth;
    LONG    biHeight;
    WORD    biPlanes;
    WORD    biBitCount;
```

```

    DWORD biCompression;
    DWORD biSizeImage;
    LONG   biXPelsPerMeter;
    LONG   biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
};

//створення .bmp файла
//задання параметрів полям заголовка .bmp файла і зображення
void CreateBmp(const char *fileName, unsigned int **colors , int
height, int width)
{
    BitmapFileHeader bfh = {0};
    BitmapInfoHeader bih = {0};

    bfh.bfType = 0x4D42; //символи 'BM'
    bfh.bfOffBits = sizeof(bfh) + sizeof(bih); //
    bfh.bfSize =bfh.bfOffBits + sizeof(colors[0][0])*width*height;
    // розмір кінцевого файлу

    bih.biSize = sizeof(bih); // розмір структури
    bih.biBitCount = 32;      // 32 біт (4 байта) на піксель
    bih.biHeight = -height;   // Висота
    bih.biWidth = width;      // Ширина
    bih.biPlanes = 1;         // містить 1
    // інші поля дорівнюють нулю

    ofstream f;
    f.open(fileName, ios::binary); // Відкриваємо файл для запису

    f.write((char*)&bfh, sizeof(bfh)); // Записуємо заголовки
    f.write((char*)&bih, sizeof(bih));

    // Записуємо зображення
    for (int i = 0; i < height; i++)
        f.write((char*)colors[i], sizeof(colors[0][0]) * width);

    f.close(); // Закриваємо файл
}
#pragma pack (pop)

int main(int argc, char* argv[])
{
    int w = 0, h = 0;
    cout << "Input height in px" << endl;
    cin >> h;
    cout << "Input width in px" << endl;
    cin >> w;

    unsigned int** color = new unsigned int*[h];
    for (int i = 0; i < h; i++)
        color[i] = new unsigned int[w];
}

```

```
for (int i = 0; i < h; i++)
    for (int j = 0; j < w; j++)
        color[i][j] = 0xFF0000; //задання червоного коліру

CreateBmp("test.bmp",color, h, w);

for (int i = 0; i < h; i++)
    delete []color[i];
delete []color;
cout << "test.bmp has been successfully created; Press Enter
to exit";
cin.ignore(2);//
return 0;
}
```

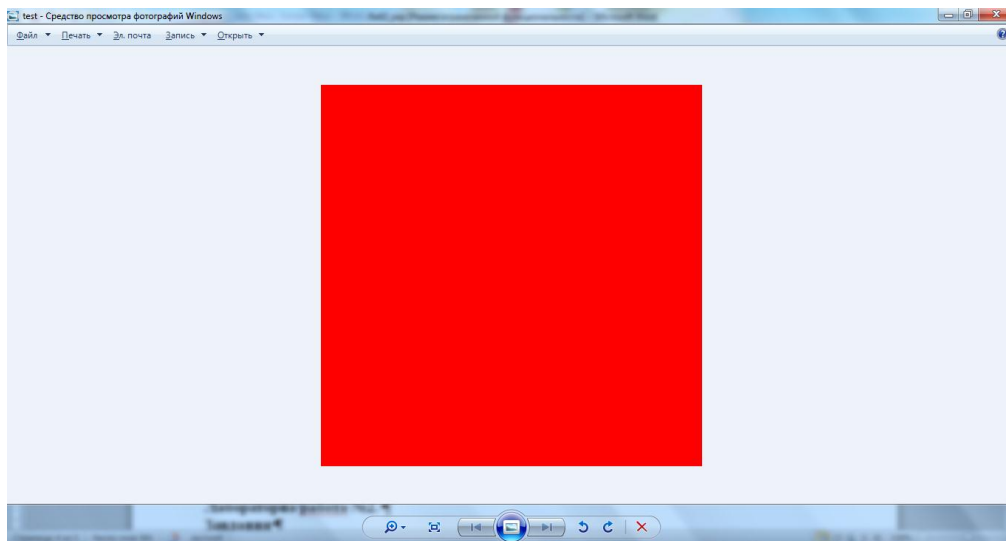



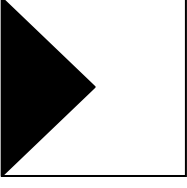
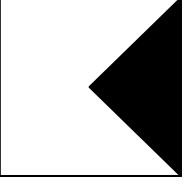
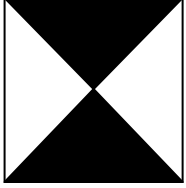
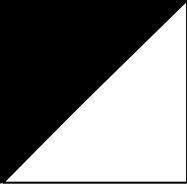
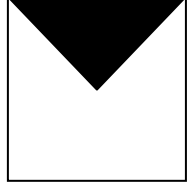
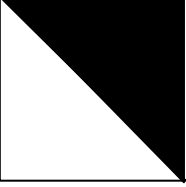
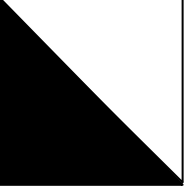
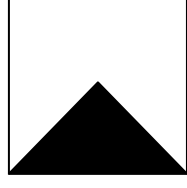
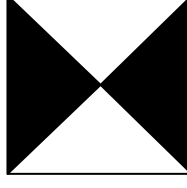
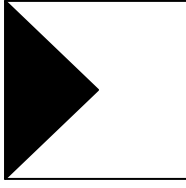

Рис.3. Збереження зображення в .bmp файлі

Лабораторна робота №2.

Завдання

Написати консольний додаток, в якому:

1. Користувач має можливість задати розмір зображення.
2. Зображення формується згідно рисунку, наведеному нижче для кожного варіанту.
Колір світлої та темної частини задає користувач.
3. Сформоване зображення зберігається у вигляді .bmp файлу.

Варіант №1	Варіант №2	Варіант №3	Варіант №4
			
Варіант №5	Варіант №6	Варіант №7	Варіант №8
			
Варіант №9	Варіант №10	Варіант №11	Варіант №12
			
Варіант №13	Варіант №14	Варіант №15	Варіант №16
