Формат bmp

Формат bmp(bitmap Picture) — формат хранения растровых изображений от компаниии Microsoft. Файлы формата BMP могут иметь расширения .bmp, .dib и .rle. по рещению разработчиков формат не привязан к конкретной аппаратной платформе. Файл состоит из трех или четырех частей, в зависимости от количества бит на один пиксел изображения. Если количество бит на один пиксел 16, 24, 32, то отсутствует таблица цветов, в противном случае таблица цветов присутствует. Структуру файла можно представить следующим образом

Структура BITMAPFILEHEADER

Описывает тип файла, размер, смещение битовой области изображения, задана в файле windows.h

```
typedef struct tagBITMAPFILEHEADER
 {  WORD    bfType;
    DWORD   bfSize;
    WORD    bfReserved1;
    WORD    bfReserved2;
     DWORD   bfOffBits;
  }   BITMAPFILEHEADER, *PBITMAPFILEHEADER;
```

где:

bfType определяет тип файла, всегда "BM"

bfTypebfSize - Обычно содержимое данного поля игнорируется.

bfReserved1 и bfReserved2 зарезервированы и равны нулю.

bfOffBits смещение области битового изображения относительно начала файла в байтах

Таким образом, в структуре BITMAPFILEHEADER важны **два поля - bfType и bfOffBits**, остальные поля можно игнорировать

Структура BITMAPINFO

Данная структура в файле windows.h определена следующим образом:

```
typedef struct tagBITMAPINFO
{    BITMAPINFOHEADER    bmiHeader;
     RGBQUARD                bmiColors[1];
}   BITMAPINFO;
```

т.е. внутри структуры находятся заголовок bmiHeader и способ задания цвета в битовом изображении.

Структура заголовка BITMAPINFOHEADER

Размер данной структуры всегда 40 байт.

```
typedef struct tagBITMAPINFOHEADER
{ DWORD  biSize;                          //размер структуры в байтах
  LONG      biWidth;                      //ширина битового изображения в пикселах;
  LONG      biHeight;                     //высота битового изображения в пикселах
  WORD     biPlanes;                      //количество плоскостей изображения (обычно
1)      WORD     biBitCount;              //количество бит на один пиксел. Может быть
1,2,4,8,16,24,32
  DWORD   biCompression;                  //метод сжатия изображения
  DWORD   biSizeImage;                    // размер несжатого изображения в байтах
  LONG      biXPelsPerMeter;
  LONG      biYPelsPerMeter;
  DWORD   biClrUsed;                      //количество используемых цветов, т.е. размер таблицы
RGBQUARD
  DWORD   biClrImportant;                 //количество важных цветов. Если 0, все цвета считаются
```

biXPelsPerMeter, biYPelsPerMeter - необходимое разрешение устройства для вывода битового изображения без искажения, задается в пикселах на метр по горизонтальи вертикали. Формат bmp - это по сути аппаратно-независимый растр. Но разрешение у устройств разное, и для того, чтобы построить наиболее подходящее изображение используют эти параметры. Если параметры имеют нулевые значения, используются параметры устройства, определяемые по умолчанию.

Сразу после структуры BITMAPINFOHEADER в файле может находиться таблица цветов. Эта таблица содержит массив структур типа RGBQUAD:

```
typedef struct tagRGBQUAD
{     BYTE rgbBlue ;
      BYTE rgbGreen;
      BYTE rgbRed  ;
      BYTE rgbReserved;
} RGBQUAD;
```

Первые три поля содержат красную, зеленую и синюю составляющую цвета. Последнее поле зарезервиро-вано, не используется.

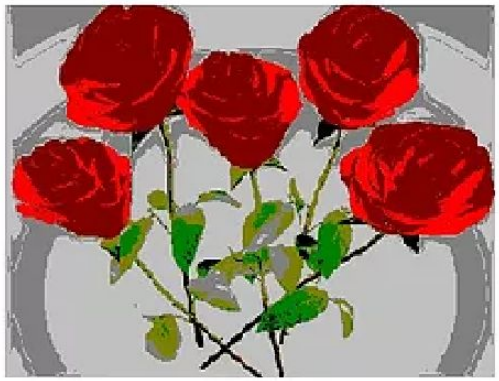
Все разновидности формата bmp условно можно разделить на два типа: палитровые и беспалитровые в зависимости от того, используется в данном формате палитра или нет. Палитра может быть даже в

biBitCount	*Тип формата*	*Макси-мальное число цветов*	*Комментарии*
1	Палитровый	2	Двуцветное, не обязательно черно-белое изображение
2	Палитровый	4	Четырехцветное изображение
4	Палитровый	16	Каждый пиксел изображения определяется 4 битами.
8	Палитровый	256	Один байт описывает один пиксел. Причем значение байта - это номер цвета в палитре.
16	Беспалитровый	2^{16}	Каждые два байта (16 битов) в растре однозначно определяют один пиксель. Здесь есть две варианта: 1) использовать не 16, а 15 битов, тогда на каждую компоненту цвета выходит по 5 бит. Таким образом мы можем использовать максимум 2^{15} = 32768 цветов. При этом теряется один бит из каждых 16 битов. 2) лишний бит можно отдать под зеленую компоненту, то есть использовать тройку R-G-B = 5-6-5, число цветов при этом 2^{16} = 65536.
24	Беспалитровый	2^{24}	Три байта определяют интенсивность каждой составляющей R, G, B для одного пиксела изображения.
32	Беспалитровый	2^{32}	4 байта определяют 3 компоненты. Но один байт, как правило не используется. Его можно использовать, например, для альфа-канала (прозрачности).

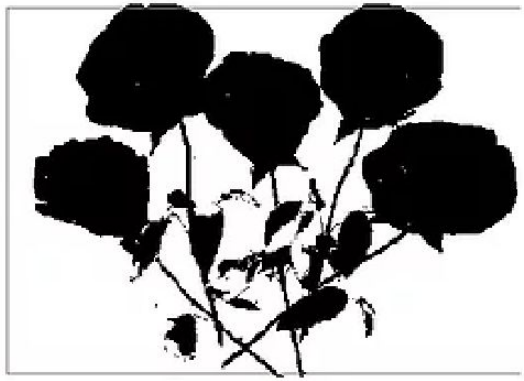
Изображение в битовой матрице записано по строкам: значения пикселов первой строки, значения пикселов второй строки, третьей и т д. Первая строка пикселов - НИЖНЯЯ строка изображения. Рисунок ниже показывает, как выглядит одно и тоже изображение в формате bmp с разной глубиной изображения.

Глубина изображения 24 бита
Размер 589 878 байтов

Палитра 256 цветов
Размер 197 688 байтов



Палитра 16 цветов
Размер 98 422 байтов



Палитра 2 цвета
Размер 24 638 байтов

Пример 1. Разбор графического файла с глубиной изображения 24 бита.

```cpp
#pragma once
#include <windows.h>
#include <conio.h>
#include <iostream>
#include <tchar.h>
#include <stdio.h>
using namespace std;

void Test()
{    setlocale(LC_ALL, "RUSSIAN");
     TCHAR title[128];
     ::GetConsoleTitle(title, 80);
     HWND hWnd = ::FindWindow(NULL, title);
     SetWindowPos(hWnd, 0, 0, 0, 750, 600, 0);
     HDC hdc = ::GetDC(hWnd);

     TCHAR fileName[] = L"C:\\Users\\Public\\DZ\\App_CPP\\Graph\\BMP24.bmp";
     wprintf(L"Попытка открытия файла  %s  \n", fileName, fileName);
     HANDLE hFile = CreateFile(fileName, GENERIC_READ, FILE_SHARE_READ, NULL,
                               OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
     if (hFile == INVALID_HANDLE_VALUE)
     {    wprintf(L"Не могу открыть файл %s\n", fileName); return;
     }

    DWORD fSizeHigh = 0,
          fSize     = GetFileSize(hFile, &fSizeHigh);
     BYTE* mmttrr    = new BYTE[fSize];
     DWORD nread;
     ReadFile(hFile, mmttrr, fSize, &nread, NULL);
     wprintf(L"Файл  %s  открыт, размер= %u байтов, получено =%d \n", fileName,
             fSize, nread);
```

```
//1. Разбор первого сегмента файла
    void * ptr = mmttrr;
    BITMAPFILEHEADER* fi = (BITMAPFILEHEADER*)mmttrr;
    DWORD offbits = fi->bfOffBits;
    char * p1 = (char*)(&fi->bfType);
    printf("Параметры заголовков \n1) Type = %c%c  OffBits = %d  size(Header)=
%d\n",
        p1[0], p1[1], offbits, sizeof(BITMAPFILEHEADER));

    //2  Разбор второго сегмента файла
    ptr = mmttrr + sizeof(BITMAPFILEHEADER);            //начало 2-го сегмента
    BITMAPINFO * pbi = (BITMAPINFO*)ptr;
    BITMAPINFOHEADER* pbh = (BITMAPINFOHEADER*)ptr;
    int w = pbh->biWidth;
    int h = pbh->biHeight;
    printf("2) %d  %d  %d  %d   w*h*3 = %d\n",
        pbh->biSizeImage, pbh->biBitCount,     w, h, w*h*3);

  //3  offbits - смещение битовой матрицы относительно начала файла
  //  Изображение выводится дважды - прямое изображение и повернутое на 90 град.

int X1 = 50;                  // Горизонтальные координаты в пикселах для 1-го и 2-го
изображений
int X2 = 400;
int Y = 150 + 256;            // Вертикальная координата экрана первой (нижней строки
изображений)
int ind_ = offbits;              // Начало битовой матрицы
int rest = 4-(3*w)%4;            // Число пикселов в строке всегда кратно 4(адрес строки
                                 // выранивается по двойному слову
```

```
while (1)
{ ind_ = offbits;                                   //ind_ - номер начального байта очередного пиксела
  for (int is = 0; is < h; is++)                    // is - номер строки битовой матрицы
  { for (int ic = 0; ic < w; ic++)                  // ic- номер столбца битовой матрицы
        {    r = mmttrr[ind_];                    // выделение цветовых составляющих пиксела
             g = mmttrr[ind_ + 1];
             b = mmttrr[ind_ + 2];
             ind_ += 3;                           // переход к следующему пикселу (следующей
тройке
                                                  //байтов  в матрице )
             ::SetPixel(hdc, ic+X1, -is+Y, COLORREF(RGB(b,g,r)));
                                                    // вывод на экран очередного пиксела в позицию
                                                    // (ic+X1, -is+Y)
             ::SetPixel(hdc, is + X2, -ic + Y, COLORREF(RGB(r, g, b)));
                                                  // вывод на экран очередного пиксела повернутого
                                                  //изображения. Кроме того, в повернутом
изображени
                                                  //поменялись цвета
             }ind_ += rest;
        }
        ::Sleep(500);                       // Циклический вывод картинки для сохранения
нормального     }                                   // вида, иначе при выполнении каких-либо
действий (например, }                                   // щелчок мышью по экрану и т.д.)
картинка ломается или
                                                  // пропадает
```

C:\Users\Public\DZ\App_CPP\Debug\Graph.exe

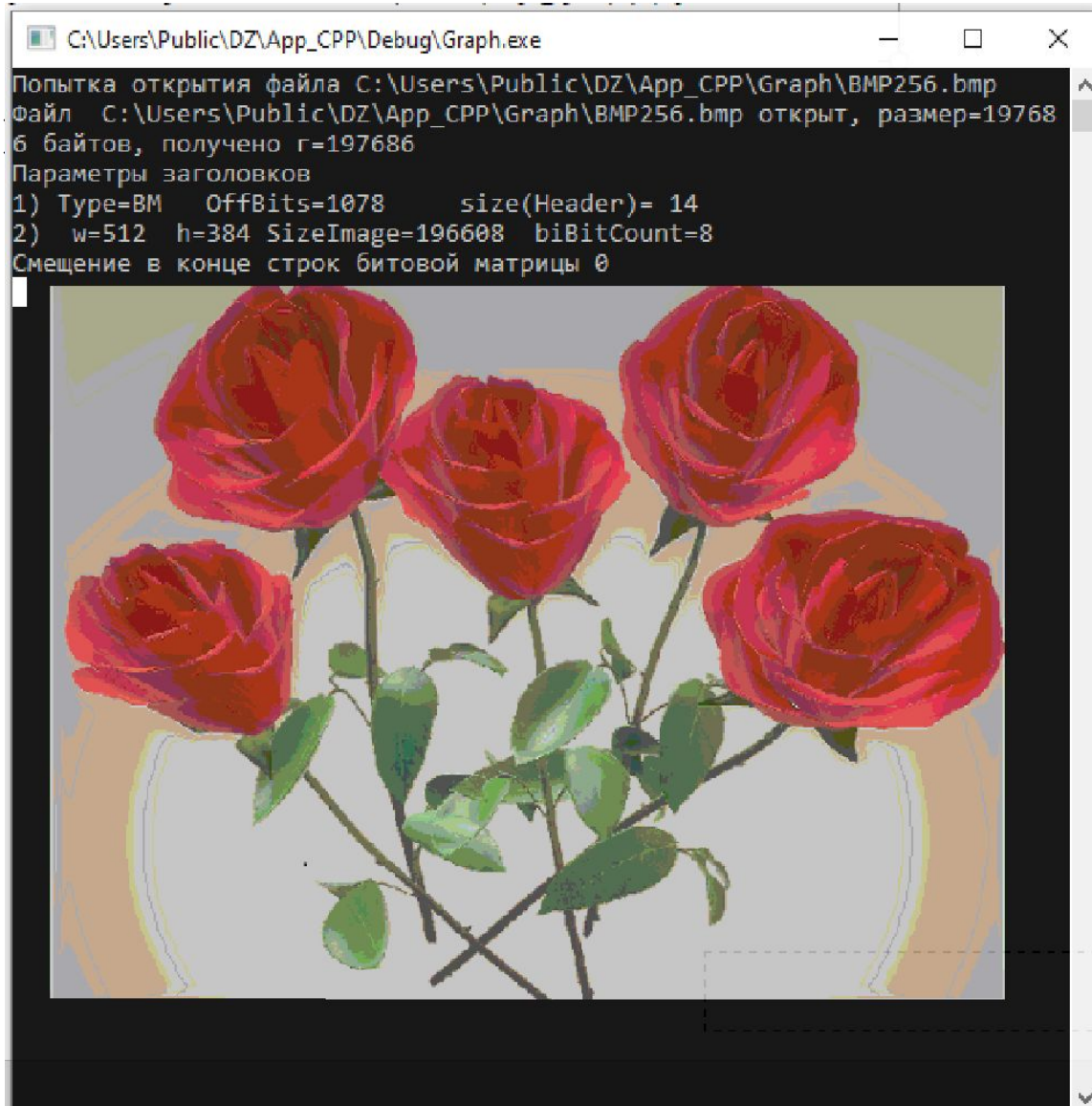
```
ФАйл  C:\Users\Public\DZ\App_CPP\Graph\BMP24.bmp  открыт, размер = 147510 байтов, получено
 =147510
Параметры заголовков
1) Type = BM   OffBits = 54  size(Header)= 14
2) 147456  24  256  192  wh3 = 147456
```

Пример 2. Разбор и вывод графического файла с глубиной изображения 256 цветов

```
#pragma once
#include <windows.h>
#include <conio.h>
#include <tchar.h>
#include <stdio.h>

void Test()
{    setlocale(LC_ALL, "RUSSIAN");
     TCHAR title[128];
     ::GetConsoleTitle(title, 80);
     HWND hWnd = ::FindWindow(NULL, title);
     SetWindowPos(hWnd, 0, 50, 50, 600, 600, 0);
     HDC hdc = ::GetDC(hWnd);

     TCHAR fileName[] = L"C:\\Users\\Public\\DZ\\App_CPP\\Graph\\BMP256.bmp";
     wprintf(L"Попытка открытия файла %s \n", fileName);
     HANDLE hFile = CreateFile(fileName, GENERIC_READ, FILE_SHARE_READ, NULL,
          OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
     if (hFile == INVALID_HANDLE_VALUE)
     {    wprintf(L"Не могу открыть файл %s\n", fileName); return;
     }

DWORD fSizeHigh, fSize= GetFileSize(hFile, &fSizeHigh);
     BYTE* mtr = new BYTE[fSize];
     DWORD nread;
     ReadFile(hFile, mtr, fSize, &nread, NULL);
     wprintf(L"Файл  %s открыт, размер %u байтов, получено %d \n", fileName, fSize,
          nread);
```

```
//1. Разбор первого сегмента файла
        void * ptr = mtr;
   BITMAPFILEHEADER* fi = (BITMAPFILEHEADER*)mtr;
        DWORD offbits = fi->bfOffBits;
        char * p1 = (char*)(&fi->bfType);
        printf("Параметры заголовков \n1) Type=%c%c   OffBits=%d    size(Header)= %u\n",
                        p1[0], p1[1], offbits,  sizeof(BITMAPFILEHEADER));

//2  Разбор второго сегмента файла
        ptr = mtr + sizeof(BITMAPFILEHEADER);
        BITMAPINFO * pbi = (BITMAPINFO*)ptr;
        BITMAPINFOHEADER* pbh = (BITMAPINFOHEADER*)(&pbi->bmiHeader);
        RGBQUAD * bmCol = (RGBQUAD*)(&pbi->bmiColors);

        long w =(long)pbh->biWidth;
        long h =(long)pbh->biHeight;
        printf("2)  w=%d  h=%d SizeImage=%d biBitCount=%d \n", w , h, pbh->biSizeImage,
                        pbh->biBitCount);

       int X1 = 20;
        int Y = 500;                    // координата  экрана первой (нижней) строки изображения
        int ind_ =  offbits;            // начало битовой матрицы
        BYTE r, g, b;
        int color;
        int w4 = w % 4;
        printf("Смещение в конце строк битовой матрицы %d\n", w4);
```

```
for (int is = 0; is < h; is++)
{    for (int ic = 0; ic < w; ic++)
     {    int  num_col = mtr[ind_]; ind_++;
          RGBQUAD col = bmCol[num_col];
          r = col.rgbRed;
          g = col.rgbGreen;
          b = col.rgbBlue;
      SetPixel(hdc, ic+X1, -is+Y,
                 COLORREF(RGB(r,g,b)
               );
     }
     ind_ += (4 - w4)%4;
}
_getche();
//Нажать любую клавишу для продолжения
}
```

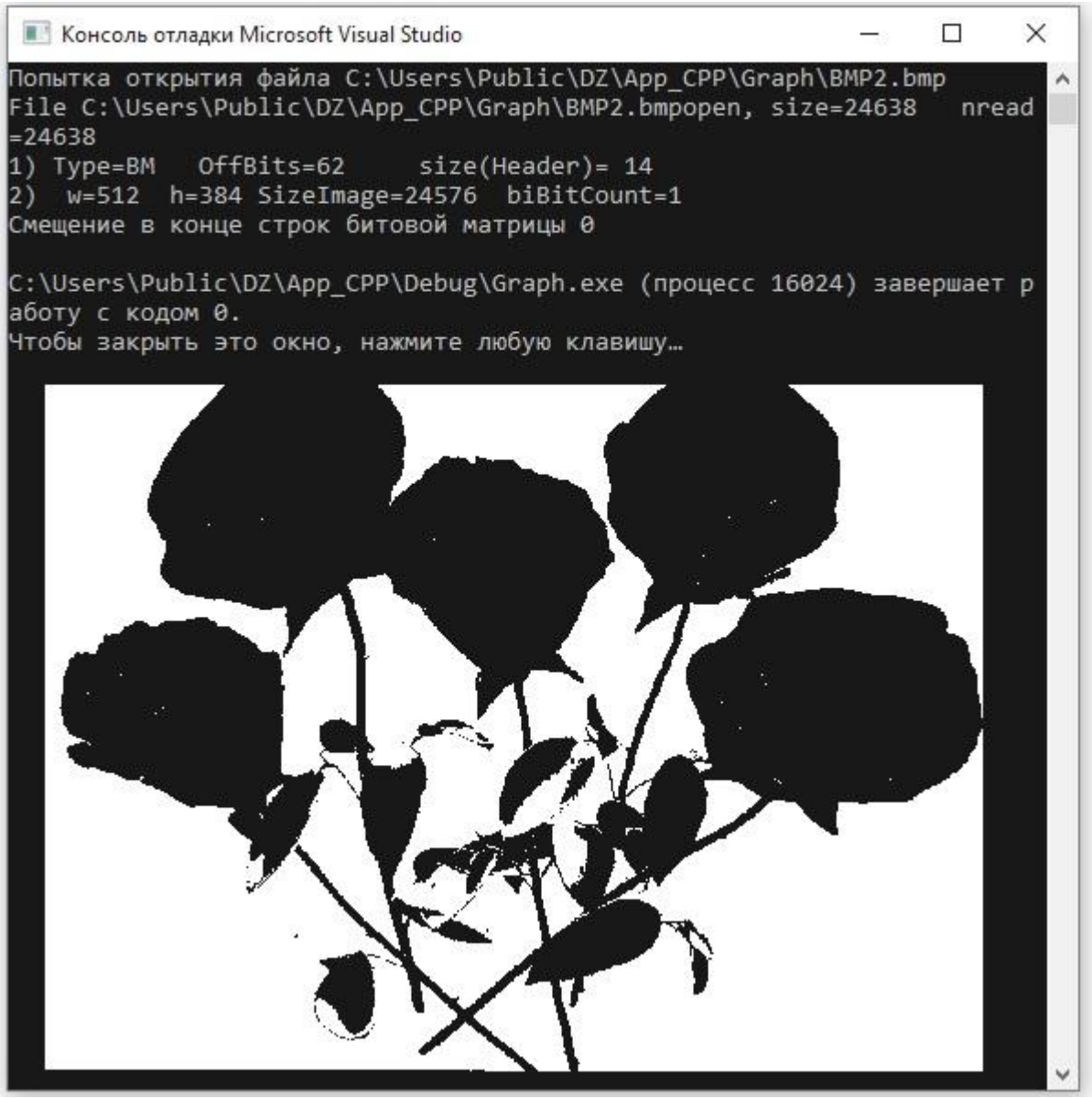


Пример 3. Разбор и вывод графического файла с глубиной изображения 1 (черно-белое изображение). Один пиксел занимает 1 бит. Бит равен 1 -белый цвет, бит=0 - черный. Таким образом, один байт битовой матрицы хранит 8 пикселов изображения.

```
#pragma once
#include <windows.h>
#include <conio.h>
#include <tchar.h>
#include <stdio.h>


void Test()
{    setlocale(LC_ALL, "RUSSIAN");
     TCHAR title[128];
     ::GetConsoleTitle(title, 80);
     HWND hWnd = ::FindWindow(NULL, title);
     SetWindowPos(hWnd, 0, 50, 50, 600, 600, 0);
     HDC hdc = ::GetDC(hWnd);
     TCHAR fileName[] = L"C:\\Users\\Public\\DZ\\App_CPP\\Graph\\BMP2.bmp";
     wprintf(L"Попытка открытия файла  %s  \n", fileName, fileName);

     HANDLE hFile = CreateFile(fileName, GENERIC_READ, FILE_SHARE_READ, NULL,
          OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
     if (hFile == INVALID_HANDLE_VALUE)
     {    wprintf(L"Не могу открыть файл %s\n", fileName); return;
     }

     DWORD fSizeHigh, fSize = GetFileSize(hFile, &fSizeHigh);
     BYTE* mtr = new BYTE[fSize];
     DWORD nread;
     ReadFile(hFile, mtr, fSize, &nread, NULL);
     wprintf(L"File %s open, size=%u   nread=%d \n", fileName, fSize, nread);
```

```
//1. Разбор первого сегмента файла
     void * ptr = mtr;
     BITMAPFILEHEADER* fi = (BITMAPFILEHEADER*)mtr;
     DWORD offbits = fi->bfOffBits;
     char * p1 = (char*)(&fi->bfType);
     printf("1) Type=%c%c   OffBits=%d     size(Header)= %u\n",
          p1[0], p1[1], offbits, sizeof(BITMAPFILEHEADER));

//2  Разбор второго сегмента файла
     ptr = mtr + sizeof(BITMAPFILEHEADER);
     BITMAPINFO * pbi = (BITMAPINFO*)ptr;
     BITMAPINFOHEADER* pbh = (BITMAPINFOHEADER*)(&pbi->bmiHeader);
     RGBQUAD * bmCol = (RGBQUAD*)(&pbi->bmiColors);
     long w = (long)pbh->biWidth;
     long h = (long)pbh->biHeight;
     printf("2) w=%d h=%d SizeImage=%d biBitCount=%d\n",w,h,pbh->biSizeImage,pbh->biBitCount);
     int X1 = 20;
     int Y = 550;                    // координата  экрана первой(нижней) строки изображения
     int ind_ = offbits;             // начало битовой матрицы
     int w4 = w%4;
     printf("Смещение в конце строк битовой матрицы %d\n", w4)
for (int is = 0; is < h; is++)
     {    for (int ic = 0; ic < w; ic+=8)
          {    int num_col = mtr[ind_];
               ind_++;
               int bb = 1;
               for ( int  ic1=0; ic1<8; ic1++)
               {    int bi = num_col & bb;   bb *= 2;
                    if (bi==0) SetPixel(hdc, ic+(7-ic1)+ X1, -is+Y, COLORREF(RGB(0,0, 0)));
                    else SetPixel(hdc, ic+(7-ic1)+ X1, -is+Y, COLORREF(RGB(255, 255, 255)));
                }
          }
```

```
    ind_ += (4-w4)%4;
  }
}
```


Консоль отладки Microsoft Visual Studio
Попытка открытия файла C:\Users\Public\DZ\App_CPP\Graph\BMP2.bmp
File C:\Users\Public\DZ\App_CPP\Graph\BMP2.bmpopen, size=24638 nread
=24638
1) Type=BM OffBits=62 size(Header)= 14
2) w=512 h=384 SizeImage=24576 biBitCount=1
Смещение в конце строк битовой матрицы 0
C:\Users\Public\DZ\App_CPP\Debug\Graph.exe (процесс 16024) завершает р
аботу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу…

Вопросы для самопроверки

1. Привязан ли формат BMP к конкретной платформе.
2. Хранит ли формат BMP изображение в сжатом виде.
3. Что такое глубина цвета изображения
4. Что такое палитровые цвета.
5 Сколько битов может занимать один пиксел при использовании палитры цветов.
6. Какая информация находится в байтах битовой матрицы изображения при использовании палитры из 256 цветов
7. Какая информация находится в байтах битовой матрицы изображения при использовании глубины цвета 24 бита.
8. Какая строка изображения - верхняя или нижняя - является первой битовой строкой матрицы.
9. Какова кратность строки битовой матрицы.

Задание для самостоятельной работы

1. Изучить данный материал.
2. Ответить на вопросы для самопроверки
3. Разработать программу на языке C++, которая выводит изображение глубины 24 и меняет цвета изображения.