Računarska grafika
(2OER7O02)

Bitmapa

Vežbe

Šta je bitmapa?

Bitmapa je grafički objekat koji se koriste za kreiranje i manipulaciju rasterskim podacima, kao i njihovo smeštanje na disku.

Organizacija

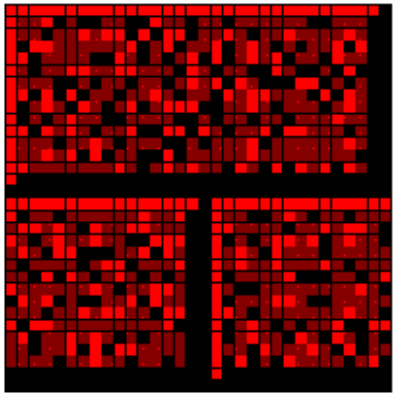
- **Bitmapa** je jedan od objekata koji se može selektovati u device context-u (DC).
- Sačinjavaju je:
  - **Zaglavlje** – definiše rezoluciju uređaja (device), veličinu datoteke, veličina polja sa bitovima slike, itd.
  - **Logička paleta**
  - **Polje bitova** – definiše karakteristike i odnos piksela bitmape i sadržaja logičke palete
- Vrste bitmapa su:
  - monohromatska ili
  - kolor
    - 16 boja – 4 bita po pixelu
    - 256 boja – 8 bita po pikselu
    - 16M boja – 24 bita po pikselu

Bottom-up i top-down bitmape

Palette-index

```
row 0, scanline 31  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
row 1, scanline 30  00 00 00 00 00 00 00 00 09 00 00 00 00 00 00 00
row 2, scanline 29  11 11 01 19 11 01 10 10 09 09 01 09 11 11 01 90
    .
    .
    .
row 31, scanline 0  99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 90
```

Pixel rectangle



Color-palette

0 Black
1 Dark red
2 Dark green
3 Dark yellow
4 Dark blue
5 Dark magenta
6 Dark cyan
7 Dark gray
8 Light gray
9 Light red
A Light green
B Light yellow
C Light blue
D Light magenta
E Light cyan
F White

Primer bottom-up bitmape sa 4-bitnom paletom (16 različitih boja)

Vrste bitmapa

- **Device-independent bitmaps (DIB)**. DIB format omogućuje da se bitmapa kreirana u jednom programu može učitati i prikazati u drugoj aplikaciji na isti način kako bi je prikazala i izvorna aplikacija.
- **Device-dependent bitmaps (DDB)** DDB nema informacija o rezoluciji, bojama i sl., tako da izgled zavisi od karakteristika DC-ja u koji se selektuje.

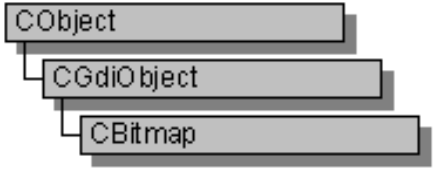
Device-dependent Bitmaps

Operacije

- Kreiranje
- Selekcija u DC
- Kopiranje u kompatibilan DC
- Uništavanje (brisanje)

- DDB su top-down bitmape

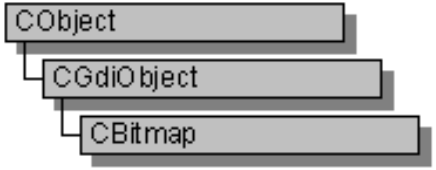
Klasa CBitmap



- Konstrukcija instance klase
- **CBitmap** – Konstruiše **CBitmap** objekat

- Inicijalizacija objekta
- **LoadBitmap** – Inicijalizuje objekat učitavanjem imenovanog resursa bitmape iz izvršne datoteke aplikacije i dodeljivanjem bitmape ovom objektu
- **LoadOEMBitmap** – Inicijalizuje objekat učitavanjem predefinisane Windows bitmape i dodeljivanjem bitmape ovom objektu
- **LoadMappedBitmap** – Učitava bitmapu i preslikava boje u trenutne sistemske boje
- **CreateBitmap** – Inicijalizuje objekat sa bitmapom iz memorije zavisnom od uređaja koja ima navedenu širinu, visinu i niz bitova
- **CreateBitmapIndirect** – Inicijalizuje objekat bitmapom širine, visine i nizom bitova (ako je naveden) zadatim pomoću BITMAP strukture
- **CreateCompatibleBitmap** – Inicijalizuje objekat sa bitmapom tako da je kompatibilan sa navedenim uređajem
- **CreateDiscardableBitmap** – Inicijalizuje objekat sa bitmapom za uklanjanje koja je kompatibilna sa navedenim uređajem.

Klasa CBitmap



- Atributi
- **GetBitmap** – Ispunjava BITMAP strukturu informacijama o bitmapi
- **Operator HBITMAP** – Vraća *handle* operativnog sistema Windows povezan sa objektom CBitmap

- Operacije
- **FromHandle** – Vraća pokazivač na CBitmap objekat na osnovu Windows HBITMAP *handle-a* bitmape
- **SetBitmapBits** – Postavlja niz bitova bitmape na navedene vrednosti
- **GetBitmapBits** – Kopira niz bitova bitmape u navedeni niz
- **SetBitmapDimension** – Postavlja širinu i visinu bitmape u jedinicama od 0,1 milimetara
- **GetBitmapDimension** – Vraća širinu i visinu bitmape. Pretpostavlja se da su visina i širina prethodno postavljene funkcijom člana **SetBitmapDimension**

Inicijalizacija bitmape

- Inicijalizacija objekta sa bitmapom tako da je kompatibilan sa navedenim uređajem

```
BOOL CBitmap::CreateCompatibleBitmap(CDC* pDC,
                                     int nWidth, int nHeight );
```

- ***pDC*** – kontekst uređaja
- ***nWidth*** – širina bitmape u pikselima
- ***nHeight*** – visina bitmape u pikselima

Inicijalizacija bitmape

- Inicijalizacija objekta sa bitmapom iz memorije zavisnom od uređaja koja ima navedenu širinu, visinu i niz bitova

  BOOL CBitmap::CreateBitmap(int *nWidth*, int *nHeight*,
  UINT *nPlanes*, UINT *nBitcount*, const void* *lpBits*);

- ***nWidth*** – širina bitmape u pikselima
- ***nHeight*** – visina bitmape u pikselima
- ***nPlanes*** – broj kolor ravni
- ***nBitcount*** – broj bitova po pikselu
- ***lpBits*** – pokazivač na niz bitova kojima se popunjava bitmapa
  - Ako je NULL, nova bitmapa ostaje neinicijalizovana
- Napomena: Ako su parametri ***nPlanes*** i ***nBitcount*** postavljeni na 1, kreira se monohromatska bitmapa.

Učitavanje bitmape iz resursa

- Inicijalizacija objekta učitavanjem imenovanog resursa bitmape iz izvršne datoteke aplikacije i dodeljivanjem bitmape ovom objektu

```
BOOL CBitmap::LoadBitmap( LPCTSTR pszResName );
BOOL CBitmap::LoadBitmap( UINT nIDResource );
```

- ***lpszResName*** – pokazivač na string koji sadrži ime resursa bitmape
- ***nIDResource*** – identifikator resursa bitmape.

- Primer

```
CBitmap bmpExercising;
bmpExercising.LoadBitmap(IDB_BITMAP1);
```

Pribavljanje atribute bitmape

- **GetBitmap** – Ispunjava BITMAP strukturu informacijama o bitmapi
- **Operator HBITMAP** – Vraća *handle* operativnog sistema Windows povezan sa objektom CBitmap

BITMAP struktura

```
typedef struct tagBITMAP {
        LONG        bmType;         // mora biti 0
        LONG        bmWidth;        // mora biti >0
        LONG        bmHeight;       // mora biti >0
        LONG        bmWidthBytes;   // br. bajtova u scan-liniji, mora biti deljiv sa 4
        WORD        bmPlanes;       // br. kolor-ravni
        WORD        bmBitsPixel;    // br. bitova po pikselu
        LPVOID      bmBits;         // pokazivač na bitove bitmape
} BITMAP;
```

- Svaka scan linija (vrsta u bitmapi) mora imati dužinu koja je celobrojni umnožak 32 bita
- Monohromatske bitmape imaju 1 ravan i 1 bit po pikselu.
  - Vrednost bita 1 predstavlja foreground boju, a 0 background boju

Pristup bitovima bitmape

- Postavljanje niza bitova bitmape na navedene vrednosti

```
DWORD CBitmap::SetBitmapBits( DWORD dwCount,
                              const void* lpBits );
```

- Kopiranje niza bitova bitmape u navedeni niz

```
DWORD CBitmap::GetBitmapBits( DWORD dwCount,
                              LPVOID lpBits ) const;
```

- ***dwCount*** – veličina niza ***lpBits*** u bajtovima
- ***lpBits*** – niz bitova koje treba postaviti u bitmapu ili iskopirati iz bitmape

Selekcija bitmape u DC

```
CPen* SelectObject( CPen* pPen );
CBrush* SelectObject( CBrush* pBrush );
CFont* SelectObject( CFont* pFont );
CBitmap* SelectObject( CBitmap* pBitmap );
int SelectObject( CRgn* pRgn );
```

Kopiranje bitmape

- Kopiranje bitmape iz izvornog u odredišni kontekst uređaja

  BOOL CDC::BitBlt(int *x*, int *y*, int *nWidth*, int *nHeight*,
  CDC* *pSrcDC*, int *xSrc*, int *ySrc*, DWORD *dwRop*);

- ***x*** – Logička x koordinata gornjeg levog ugla odredišnog pravougaonika
- ***y*** – Logička y koordinata gornjeg levog ugla odredišnog pravougaonika
- ***nWidth*** – Širina (u logičkim jedinicama) odredišnog pravougaonika i izvorne bitmape
- ***nHeight*** – Visina (u logičkim jedinicama) odredišnog pravougaonika i izvorne bitmape
- ***pSrcDC*** – Pokazivač na CDC objekat izvornog konteksta uređaja
  - Mora biti NULL ako dvRop predstavlja rastersku operaciju koja ne uključuje izvor
- ***xSrc*** – Logička x koordinata gornjeg levog ugla izvorne bitmape
- ***ySrc*** – Logička y koordinata gornjeg levog ugla izvorne bitmape

Rasterske operacije (parametar *dwROP*)

- **BLACKNESS** Pretvara sve piksele u crnu boju.
- **DSTINVERT** Invertuje odredišnu bitmapu.
- **MERGECOPY** Kombinuje obrazac i izvornu bitmapu koristeći logički I operator.
- **MERGEPAINT** Kombinuje obrnutu izvornu bitmapu sa odredišnim bitmapom koristeći logički ILI operator.
- **NOTSRCCOPY** Kopira obrnutu izvornu bitmapu na odredište.
- **NOTSRCERASE** Invertuje rezultat kombinovanja odredišne i izvorne bitmape koristeći logički ILI operator.
- **PATCOPY** Kopira obrazac na odredišnu bitmapu.
- **PATINVERT** Kombinuje odredišnu bitmapu sa obrasem koristeći logički ISKLJKUČIVO ILI operator.
- **PATPAINT** Kombinuje obrnutu izvornu bitmapu sa obrascem koristeći logički ILI operator. Kombinuje rezultat ove operacije sa odredišnim bitmapom koristeći logički ILI operator.
- **SRCAND** Kombinuje piksele odredišne i izvorne bitmape koristeći logički operator AND.
- **SRCCOPY** Kopira izvornu bitmapu u odredišnu bitmapu.
- **SRCERASE** Invertuje odredišnu bitmapu i kombinuje rezultat sa izvornom bitmapom koristeći logički operator AND.
- **SRCINVERT** Kombinuje piksele odredišne i izvorne bitmape koristeći logički ISKLJUČIVO ILI operator.
- **SRCPAINT** Kombinuje piksele odredišne i izvorne bitmape koristeći logički ILI operator.
- **WHITENESS** Pretvara sve piksele u crnu boju.

Kopiranje bitmape sa skaliranjem

- Kopiranje bitmape iz izvornog u odredišni kontekst uređaja sa skaliranjem

  BOOL CDC::StretchBlt(int *x*, int *y*, int *nWidth*, int *nHeight*,

  CDC* *pSrcDC*, int *xSrc*, int *ySrc*, int *nSrcWidth*, int *nSrcHeight*, DWORD *dwRop*);

- ***x*** – Logička x koordinata gornjeg levog ugla odredišnog pravougaonika
- ***y*** – Logička y koordinata gornjeg levog ugla odredišnog pravougaonika
- ***nWidth*** – Širina (u logičkim jedinicama) odredišnog pravougaonika i izvorne bitmape
- ***nHeight*** – Visina (u logičkim jedinicama) odredišnog pravougaonika i izvorne bitmape
- ***pSrcDC*** – Pokazivač na CDC objekat izvornog konteksta uređaja
  - Mora biti NULL ako dvRop predstavlja rastersku operaciju koja ne uključuje izvor
- ***xSrc*** – Logička x koordinata gornjeg levog ugla izvorne bitmape
- ***ySrc*** – Logička y koordinata gornjeg levog ugla izvorne bitmape
- ***nSrcWidth*** – Širina (u logičkim jedinicama) izvorne bitmape
- ***nSrcHeight*** – Visina (u logičkim jedinicama) izvorne bitmape

Iscrtavanje bitmape iz resursa

```
void CBitmapView::OnDraw(CDC* pDC) {
        // Ucitavanje bitmape iz resursa
        CBitmap bmpExercising;
        bmpExercising.LoadBitmap(IDB_BITMAP1);
        // Kreiranje memorijskog DC-ja kompatibilnog sa DC-jem prozora
        CDC* MemDCExercising = new CDC();
        MemDCExercising->CreateCompatibleDC(pDC);
        // Selekcija bitmape
        MemDCExercising->SelectObject(bmpExercising);
        // Kopiranje bitova iz memorijskog u prozorski DC
        pDC->BitBlt( 10, 10, 450, 400, MemDCExercising, 0, 0, SRCCOPY);
        MemDCExercising->DeleteDC();
        delete MemDCExercising;
        bmpExercising.DeleteObject();
}
```

Transparentni prikaz bitmape (1)
Učitavanje i pravljenje monohromatske

```
CBitmap bmpImage;
// Učitavanje bitmape iz resursa
BOOL suc = bmpImage.LoadBitmap(IDB_BITMAP1);

CBitmap bmpMask;
BITMAP bm;
bmpImage.GetBitmap(&bm);

// Pravljenje monohromatske bitmape iste veličine
bmpMask.CreateBitmap(bm.bmWidth,bm.bmHeight,1,1,NULL);
```

Transparentni prikaz bitmape (2)
Učitavanje bitmapa u DC

```
CDC* SrcDC = new CDC();
SrcDC->CreateCompatibleDC(pDC);
CDC* DstDC = new CDC();
DstDC->CreateCompatibleDC(pDC);

CBitmap* pOldSrcBmp = SrcDC->SelectObject(&bmpImage);
CBitmap* pOldDstBmp = DstDC->SelectObject(&bmpMask);
```

Transparentni prikaz bitmape (3)
Popunjavanje monohromatske bitmape

```
COLORREF clrTopLeft = SrcDC->GetPixel(0,0);
COLORREF clrSaveBk = SrcDC->SetBkColor(clrTopLeft);
DstDC->BitBlt(0,0,bm.bmWidth,bm.bmHeight,SrcDC,0,0,SRCCOPY);
```

SRCCOPY

SrcDC

DstDC

Transparentni prikaz bitmape (4)
Markiranje pozadine u izvornoj bitmapi

```
COLORREF clrSaveDstText = SrcDC->SetTextColor(RGB(255,255,255));
SrcDC->SetBkColor(RGB(0,0,0));
SrcDC->BitBlt(0,0,bm.bmWidth,bm.bmHeight,DstDC,0,0,SRCAND);
```

SRCAND

DstDC

SrcDC

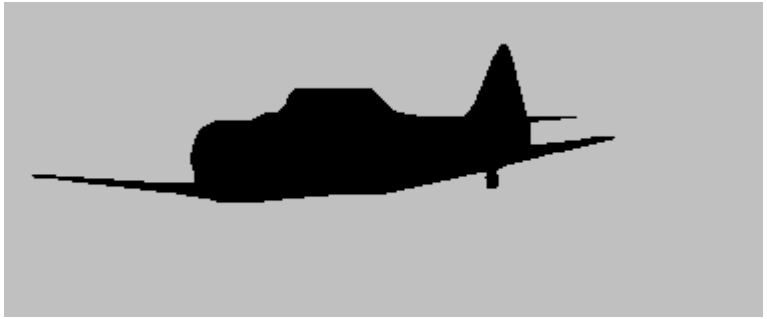
Transparentni prikaz bitmape (5)
Oslobađanje resursa

```
DstDC->SetTextColor(clrSaveDstText);
SrcDC->SetBkColor(clrSaveBk);
SrcDC->SelectObject(pOldSrcBmp);
DstDC->SelectObject(pOldDstBmp);
SrcDC->DeleteDC();
delete SrcDC;
DstDC->DeleteDC();
delete DstDC;
```

Transparentni prikaz bitmape (6)
Priprema prikaza

```
CDC* MemDC = new CDC();
MemDC->CreateCompatibleDC(pDC);
CBitmap* bmpOldT =MemDC->SelectObject(&bmpMask);
pDC->BitBlt(0,0,bm.bmWidth,bm.bmHeight,MemDC,0,0,SRCAND);
```

SRCAND



MemDC

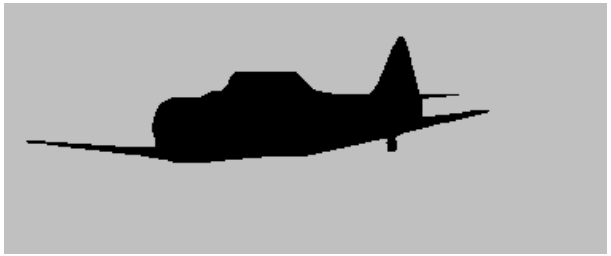
pDC

Transparentni prikaz bitmape (7)
Prikaz bitmape

```
MemDC->SelectObject(&bmpImage);
pDC->BitBlt(0,0,bm.bmWidth,bm.bmHeight,MemDC,0,0,SRCPAINT);
MemDC->SelectObject(bmpOldT);
delete MemDC;
```

MemDC

SRCPAINT



pDC

Kopiranje bitmape u oblast oblika paralelograma

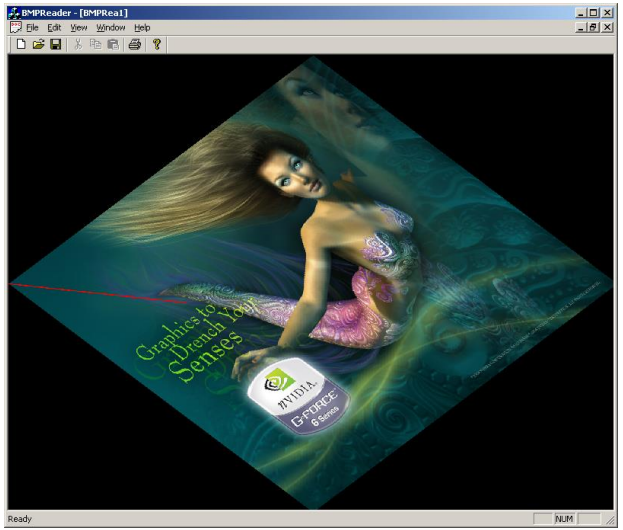
- Kopiranje bitmape iz izvornog u oblast oblika paralelograma odredišnog konteksta uređaja

  BOOL CDC::PlgBlt(POINT *lpPoint*, CDC* *pSrcDC*, int *xSrc*, int *ySrc*, int *nWidth*, int *nHeight*, CBitmap& *maskBitmap*, int *xMask*, int *yMask*);

- ***lpPoint*** – niz od 3 tačke (u logičkim koodinatama) odredišnog paralelograma. Gornji levi ugao izvornog pravougaonika mapira se u prvu tačku, gornji desni u drugu, a donji levi u treću. Četvrta tačka se izračunava tako da formira paralelogram.
- ***maskBitmap*** – Opciona monohromatska bitmapa koja se koristi kao maska
- ***xMask*** – x koordinata gornjeg levog ugla bitmape maske
- ***yMask*** – y koordinata gornjeg levog ugla bitmape maske

Kopiranje bitmape u oblast oblika paralelograma (primer)

```
POINT points[3]={{0,384},{512,0},{512,768}};
CBitmap pBmp;
MemDC->PlgBlt(points,MemDC,0,0,1024,768,pBmp,0,0);
```

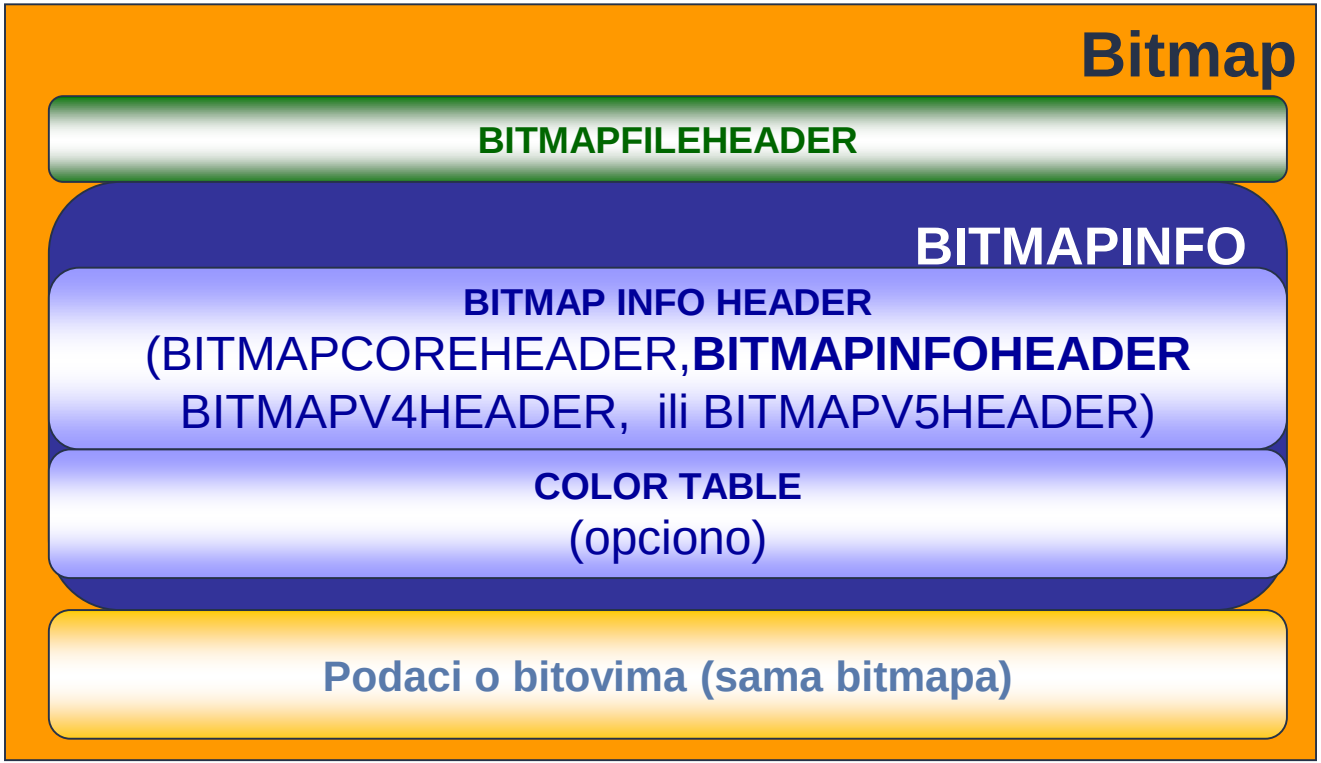


Device-Independent Bitmaps

Glavne osobine DIB-a

- DIB nije GDI objekat
- DIB ne može biti selektovan u DC
- DIB se iscrtava se korišćenjem posebnih GDI funkcija

Struktura bmp datoteke



BITMAP struktura

- Sadrži informacije o tipu, veličini i organizaciji DIB-a

```
typedef struct tagBITMAPFILEHEADER {
        WORD bfType;        // mora biti BM
        DWORD bfSize;       // veličina BMP fajla u B
        WORD bfReserved1; // mora biti 0
        WORD bfReserved2; // mora biti 0
        DWORD bfOffBits;    /* određuje pomeraj bitova bitmape u odnosu na
        početak BITMAPFILEHEADER strukturu - offset */
} BITMAPFILEHEADER;
```

BITMAPINFO struktura

- Sadrži informacije o formatu i veličini DIB-a

```
typedef struct tagBITMAPINFO {
        BITMAPINFOHEADER bmiHeader;      // zaglavlje
        RGBQUAD bmiColors[1];            // tabela boja (color table)
} BITMAPINFO;

typedef struct tagRGBQUAD {
        BYTE rgbBlue;
        BYTE rgbGreen;
        BYTE rgbRed;
        BYTE rgbReserved; // mora biti 0
} RGBQUAD;
```

BITMAPINFOHEADER struktura

- Sadrži informacije o formatu i veličini DIB-a

```
typedef struct tagBITMAPINFOHEADER{
        DWORD biSize; // veličina strukture u B
        LONG biWidth; // širina slike u pikselima
        LONG biHeight; // visina slike u pikselima
        WORD biPlanes; // 1
        WORD biBitCount ; // (0-jpg, 1, 4, 8, 16, 24 i 32)
        DWORD biCompression; // BI_RGB, BI_RLE8, BI_RLE4, ...
        DWORD biSizeImage; // veličina slike u B
        LONG biXPelsPerMeter; // horizontalna rezolucija (piksela po metru)
        LONG biYPelsPerMeter; // vertikalna rezolucija (piksela po metru)
        DWORD biClrUsed; // broj boja u tabeli koje se zaista koriste (ako je 0 koriste se sve boje iz tabele)
        DWORD biClrImportant; // broj neophodnih boja za prikaz
} BITMAPINFOHEADER;
```

Klasa DImage

```
class Dimage {
public:
        DImage(void);
        DImage(CBitmap& bmp);
        virtual ~DImage(void);
        bool Load(CString fileName);
        bool Save(CString fileName);
        void Draw(CDC* pDC, CRect rcImg, CRect rcDC);
        int Width();
        int Height();
        int BPP();
};
```

Klasa Dimage
Kreiranje na osnovu bitmape

```
DImage::DImage(CBitmap& bmp) {
        m_pBmp = new CBitmap();
        BITMAP bmpInfo;
        bmp.GetBitmap(&bmpInfo);
        m_nWidth = bmpInfo.bmWidth;
        m_nHeight = bmpInfo.bmHeight;
        m_pBmp->CreateBitmap( m_nWidth, m_nHeight, 1, 32, NULL);
        int BPP = bmpInfo.bmBitsPixel / 8;
        m_nBPP = 4;
        int size = m_nWidth*m_nHeight*BPP;
        m_pBuf = new unsigned char[size];
        bmp.GetBitmapBits(size, m_pBuf);
        Convert(BPP,m_nBPP);
        m_pBmp->SetBitmapBits(m_nWidth*m_nHeight*m_nBPP, m_pBuf);
}
```

Klasa Dimage
Učitavanje iz datoteke

```
bool DImage::Load(CString fileName) {
        CFile f;
        if (f.Open(fileName, CFile::modeRead)) {
                unsigned char* buf;
                unsigned long len = f.GetLength();
                buf = new unsigned char[len];
                f.Read(buf, len);
                if (m_pBuf) delete m_pBuf;
                m_pBuf = stbi_load_from_memory(buf, len,  &m_nWidth, &m_nHeight, &m_nBPP, 0);
                f.Close();
                delete buf;
```

stb_image.c

Klasa Dimage
Učitavanje iz datoteke

```
            if (m_pBmp) delete m_pBmp;
            m_pBmp = new CBitmap();
            m_pBmp->CreateBitmap( m_nWidth, m_nHeight, 1, 32, NULL);
            BITMAP bitmapInfo;
            m_pBmp->GetBitmap(&bitmapInfo);
            int curBPP = bitmapInfo.bmBitsPixel / 8;
            Convert(m_nBPP,curBPP);
            m_pBmp->SetBitmapBits(m_nWidth*m_nHeight*m_nBPP,  m_pBuf);
            return true;
        }
        return false;
}
```

Klasa Dimage
Snimanje u datoteku

```
bool DImage::Save(CString fileName) {
	CFile f;
	BOOL b = f.Open(fileName, CFile::modeCreate | CFile::modeWrite);
	if (b) {
		Flip();
		Save(f);
		f.Close();
	}
	return true;
}
```

Klasa Dimage
Snimanje u datoteku

```
void DImage::Save(CFile& file)
{
        BITMAPFILEHEADER bmfHdr;
        BITMAPINFO BMI;
        BMI.bmiHeader.biSize = sizeof(BITMAPINFOHEADER);
        BMI.bmiHeader.biWidth = m_nWidth;
        BMI.bmiHeader.biHeight = m_nHeight;
        BMI.bmiHeader.biPlanes = 1;
        BMI.bmiHeader.biBitCount = 32;
        BMI.bmiHeader.biCompression = BI_RGB;
        BMI.bmiHeader.biSizeImage = 0;
        BMI.bmiHeader.biXPelsPerMeter = 2835;
        BMI.bmiHeader.biYPelsPerMeter = 2835;
        BMI.bmiHeader.biClrUsed = 0;
        BMI.bmiHeader.biClrImportant = 0;
```

Klasa Dimage
Snimanje u datoteku

```
    bmfHdr.bfType = DIB_HEADER_MARKER;  // "BM"
    bmfHdr.bfSize = sizeof(BITMAPINFO) + sizeof(BITMAPFILEHEADER);
    bmfHdr.bfReserved1 = 0;
    bmfHdr.bfReserved2 = 0;
    bmfHdr.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFO);

    // Snimanje zaglavlja datoteke
    file.Write((LPSTR)&bmfHdr, sizeof(BITMAPFILEHEADER));

    // Snimanje zaglavlja bitmape
    file.Write(&BMI, sizeof(BITMAPINFO));

    // Snimanje sadržaja bitmape
    file.Write(m_pBuf, m_nWidth*m_nHeight*m_nBPP);
}
```

Klasa Dimage
Crtanje

```
void DImage::Draw(CDC* pDC, CRect rcImg, CRect rcDC) {
        CDC* pMemDC = new CDC();
        BOOL b = pMemDC->CreateCompatibleDC(pDC);
        pMemDC->SelectObject(m_pBmp);
        pDC->SetStretchBltMode(HALFTONE);
        b = pDC->StretchBlt(rcDC.left,rcDC.top,
                rcDC.Width(),rcDC.Height(),
                pMemDC,rcImg.left,rcImg.top,
                rcImg.Width(),rcImg.Height(), SRCCOPY);
        pMemDC->DeleteDC();
        delete pMemDC;
}
```

Domaći zadatak

- Korišćenjem **DImage** klase učitati proizvoljnu sliku sa diska u JPG formatu i prikazati je na ekranu
- Implementirati funkcije za zatamnjivanje i osvetljavanje učitane slike za datu vrednost.
  - Promenu osvetljaja vršiti skaliranjem RGB komponenti svakog piksela, ali tako da vrednost ostane u opsegu [0,255].
- Implementirati funkciju za rotaciju učitane slike za dati ugao
  - Koristiti funkciju **PlgBlt**
- Transformisanu sliku snimiti na disk pod drugim imenom

Pitanja