04 - Format Citra dan Struktur Data untuk Citra

IF4073 Interpretasi dan Pengolahan Citra

Oleh: Rinaldi Munir

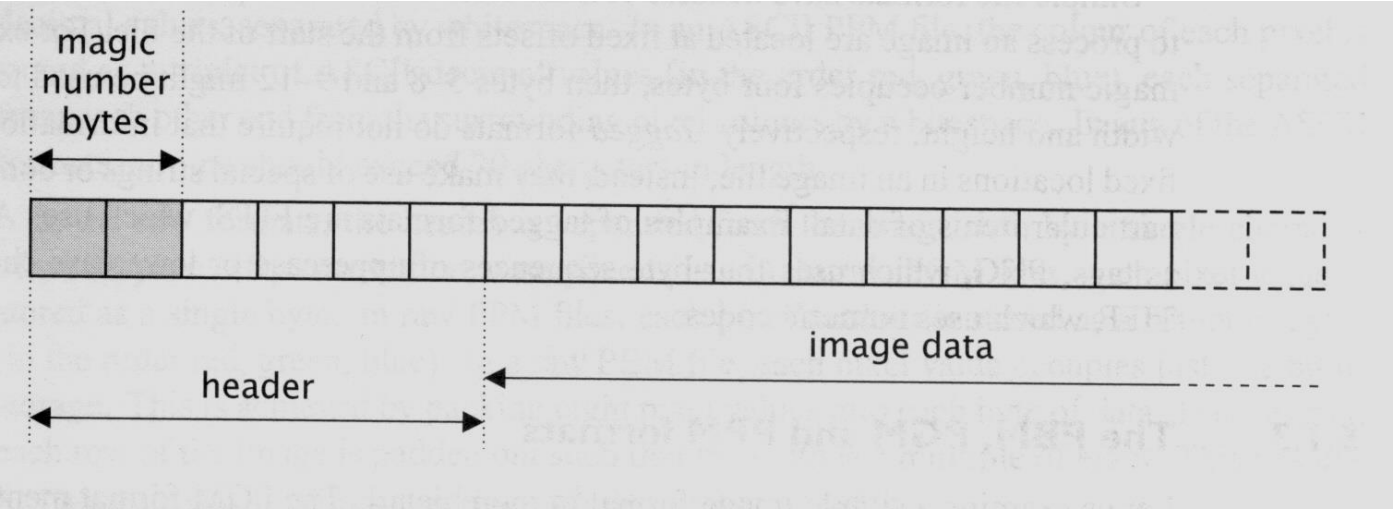
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

Format File Citra

- BMP (Bitmap)
- GIF (Graphic Interchange Format) -
- PNG (Portable Network Graphics)
- JPEG (Joint Photographic Experts Group)
- TIFF (Tagged Image File Format)
- PGM (Portable Gray Map)
- FITS (Flexible Image Transport System)

- File citra selalu diawali dengan *header* yang berisi informasi tentang data citra dan bagaimana cara membaa data citra

- Kebanyakan header diawali dengan **signature** atau “magic number” (yaitu deretan *byte* yang mengidentifikasi format file)



Sumber: Dr. George Bebis, *Image Formation and Representation*, CS485/685 Computer Vision

PBM/PGM/PPM format

- PGM = *Portable Graymap*
- PGM merupakan format yang popular untuk citra *grayscale* (8 bits/pixel).
- Masih satu keluarga dengan:
  - PBM (*Portable Bitmap*), untuk citra biner (1 bit/pixel)
  - PPM (*Portable Pixelmap*), untuk citra berwarna (24 bits/pixel)

Type	Magic number		Extension	Colors
	ASCII	Binary		
Portable BitMap[1]	P1	P4	.pbm	0–1 (white & black)
Portable GrayMap[2]	P2	P5	.pgm	0–255 (gray scale)
Portable PixMap[3]	P3	P6	.ppm	0–255 (RGB)

Sumber: wikipedia

- Contoh format PBM

Ukuran citra

```
P1
# This is an example bitmap of the letter "J"
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

Note: there is a newline character at the end of each line

Sumber: wikipedia

Citra yang terbentuk

- Contoh format PGM

Ukuran citra

Jumlah level keabuan

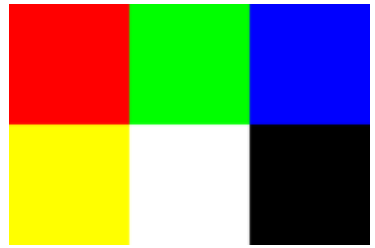
```
P2
# Shows the word "FEEP" (example from Netpbm man page on PGM)
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Citra yang terbentuk

Sumber: wikipedia

- Contoh format PPM untuk citra berwarna RGB

```
P3
3 2
255
# The part above is the header
# "P3" means this is a RGB color image in ASCII
# "3 2" is the width and height of the image in pixels
# "255" is the maximum value for each color
# The part below is image data: RGB triplets
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
```



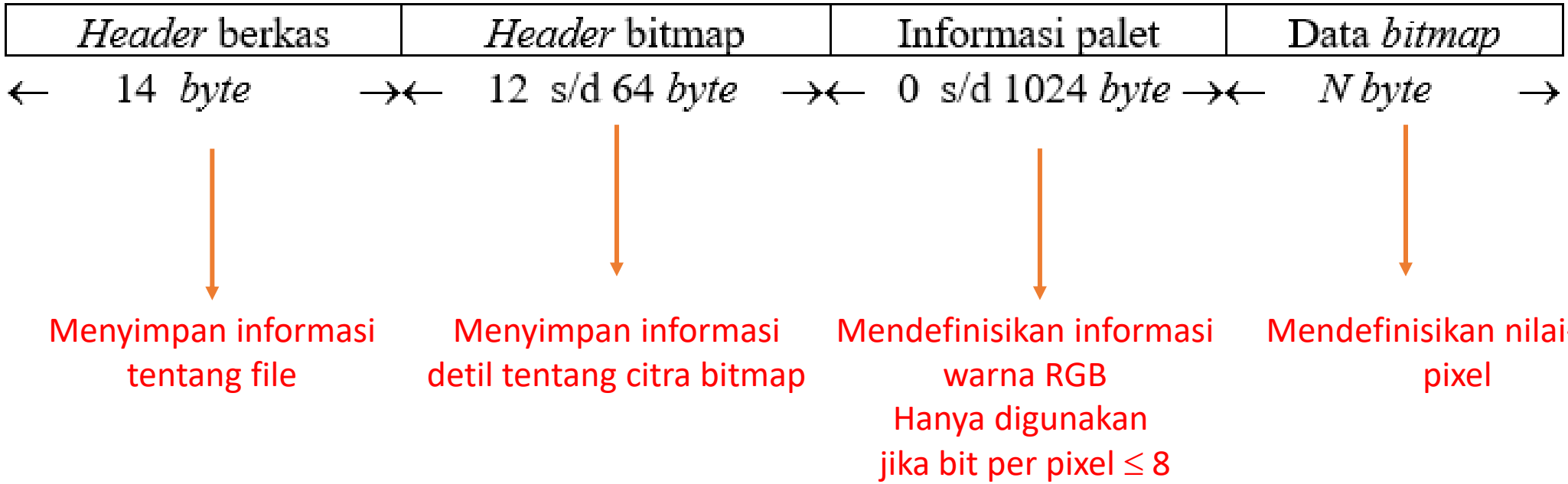
Citra yang terbentuk

Sumber: wikipedia

Format File BMP

- Dikenal juga dengan nama file citra bitmap atau *device independent bitmap (DIB) file format.*
- Merupakan format citra yang baku untuk sistem operasi Windows dan OS/2
- Tidak dimampatkan (*uncompressed image*)
- Kekurangan: membutuhkan memori yang besar
- Kelebihan: kualitas citranya lebih bagus daripada citra terkompresi (karena tidak ada informasi yang hilang)
- Citra format BMP ada tiga macam: citra biner, citra berwarna, dan citra hitam-putih (*grayscale*).

Struktur file citra BMP:



Tabel 1. *Header berkas bitmap (panjang = 14 byte)*

Byte ke-	Panjang (*byte*)	Nama	Keterangan
1 – 2	2	BmpType	Signature: BA = *bitmap array*, CI = *icon* BM = *bitmap*, CP = *color pointer* PT = *pointer*
3 – 6	4	*BmpSize*	Ukuran berkas *bitmap*
7 – 8	2	*XhotSpot*	Reserved
9 – 10	2	*YhotSpot*	Reserved
11 – 14	4	*OffBits*	Ofset ke awal data *bitmap* (dalam *byte*)

Tabel 2. *Header bitmap versi lama dari Microsoft Windows (12 byte)*

***Byte* ke-**	**Panjang (*byte*)**	**Nama**	**Keterangan**
1 – 4	4	HdrSize	Ukuran *header* dalam satuan *byte*
5 – 6	2	*Width*	Lebar *bitmap* dalam satuan *pixel*
7 – 8	2	*Height*	Tinggi *bitmap* dalam satuan *pixel*
9 – 10	2	*Planes*	Jumlah *plane* (umum- nya selalu satu)
11 – 12	2	*BitCount*	Jumlah bit per *pixel*

Tabel 3. *Header bitmap versi baru dari Microsoft Windows (40 byte)*

***Byte* ke-**	**Panjang (*byte*)**	**Nama**	**Keterangan**
1 – 4	4	HdrSize	Ukuran *header* dalam satuan *byte*
5 – 8	4	*Width*	Lebar *bitmap* dalam satuan *pixel*
9 – 12	4	*Height*	Tinggi *bitmap* dalam satuan *pixel*
13 – 14	2	*Planes*	Jumlah *plane* (umum- nya selalu satu)
15 – 16	2	*BitCount*	Jumlah bit per *pixel*
17 – 20	4	*Compression*	0=tak dimampatkan, 1=dimampatkan
21 – 24	4	*ImgSize*	Ukuran *bitmap* dalam *byte*
25 – 28	4	*HorzRes*	Resolusi horizontal
29 – 32	4	*VertRes*	Resolusi vertikal
33 – 36	4	*ClrUsed*	Jumlah warna yang digunakan
37 – 40	4	*ClrImportant*	Jumlah warna yang penting

Tabel 4. *Header bitmap versi baru dari IBM OS/2 (64 byte)*

Byte ke-	Panjang (*byte*)	Nama	Keterangan
1 – 4	4	HdrSize	Ukuran *header* dalam satuan *byte*
5 – 8	4	*Width*	Lebar *bitmap* dalam satuan *pixel*
9 – 12	4	*Height*	Tinggi *bitmap* dalam satuan *pixel*
13 – 14	2	*Planes*	Jumlah *plane* (umumnya selalu satu)
15 – 16	2	*BitCount*	Jumlah bit per *pixel*
17 – 20	4	*Compression*	0 = tak dimampatkan, 1 = dimampatkan
21 – 24	4	*ImgSize*	Ukuran *bitmap* dalam *byte*
25 – 28	4	*HorzRes*	Resolusi horizontal
29 – 32	4	*VertRes*	Resolusi vertikal
33 – 36	4	*ClrUsed*	Jumlah warna yang digunakan
37 – 40	4	*ClrImportant*	Jumlah warna yang penting
41 – 42	2	*Units*	Satuan pengukuran yang dipakai
43 – 44	2	*Reserved*	*Field* Cadangan
45 – 46	2	*Recording*	Algoritma perekaman
47 – 48	2	*Rendering*	Algoritma *halftoning*
49 – 52	4	*Size1*	Nilai ukuran 1
53 – 56	4	*Size2*	Nilai ukuran 2
57 – 60	4	*ClrEncoding*	Pengkodean warna
61 – 64	4	Identifier	Kode yang digunakan aplikasi

- Informasi palet warna terletak sesudah *header bitmap*. Panjangnya 0 sampai 1024 bit
- Informasi palet warna dinyatakan dalam suatu tabel *RGB*.
- Setiap *entri* pada tabel terdiri atas tiga buah *field*, yaitu *R* (*red*), *G* (*green*), dan *B* (*blue*).
- Untuk citra *grayscale*, nilai R = G = B.

- Penyimpanan data *bitmap* di dalam berkas disusun terbalik dari bawah ke atas dalam bentuk matriks yang berukuran *Height* $\times$ *Width*. Baris ke-0 pada matriks data *bitmap* menyatakan data *pixel* di citra baris terbawah, sedangkan baris terakhir pada matriks menyatakan data *pixel* di citra baris teratas.

- Untuk citra berwarna dengan 8 bit/*pixel*, data *bitmap* menyatakan indeks ke palet warna. Jumlah kombinasi warna yang dihasilkan adalah $2^8 = 256$ warna.

<*header berkas*>
<*header bitmap*>

<*palet warna RGB*>

	R	*G*	*B*
1	20	45	24
2	14	13	16
3	12	17	15
...			
256	46	78	25

<*data bitmap*>
2 2 1 1 1 3 5 ...

- Pada contoh di atas, warna *pixel* pertama dinyatakan oleh entri baris ke-2 pada palet warna (R = 14, G = 13, B = 16).

- Untuk citra *truecolor* (24-bit), tidak ada palet warna. Nilai *RGB* langsung diuraikan di dalam data *bitmap*.
- Setiap elemen data *bitmap* panjangnya 3 *byte*, masing-masing *byte* menyatakan komponen *R*, *G*, dan *B*.

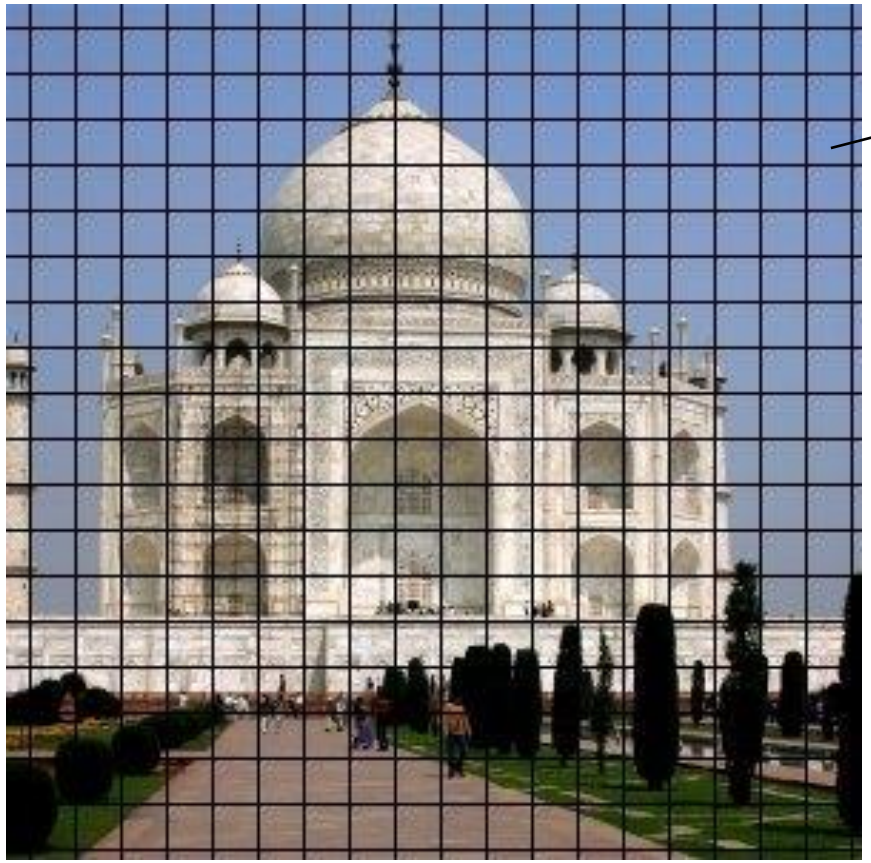
<*header* berkas>
<*header bitmap*>

<data *bitmap*>
20 19 21 24 24 23 24 ...

- *Pada contoh di atas, pixel* pertama mempunyai *R* = 20, *G* = 19, *B* = 21, *pixel* kedua mempunyai *R* = 24, *G* = 24, *B* = 23. Demikian seterusnya.

- Citra *truecolor* disebut juga citra 16 juta warna, karena ia mampu menghasilkan 2^{24} = 16.777.216 kombinasi warna.

Pada citra 24-bit (*real image* atau *truecolor image*), 1 *pixel* = 24 bit, terdiri dari komponen RGB (*Red-Green-Blue*) sepanjang 24 bit, masing-Masing 8 bit untuk setiap komponen.



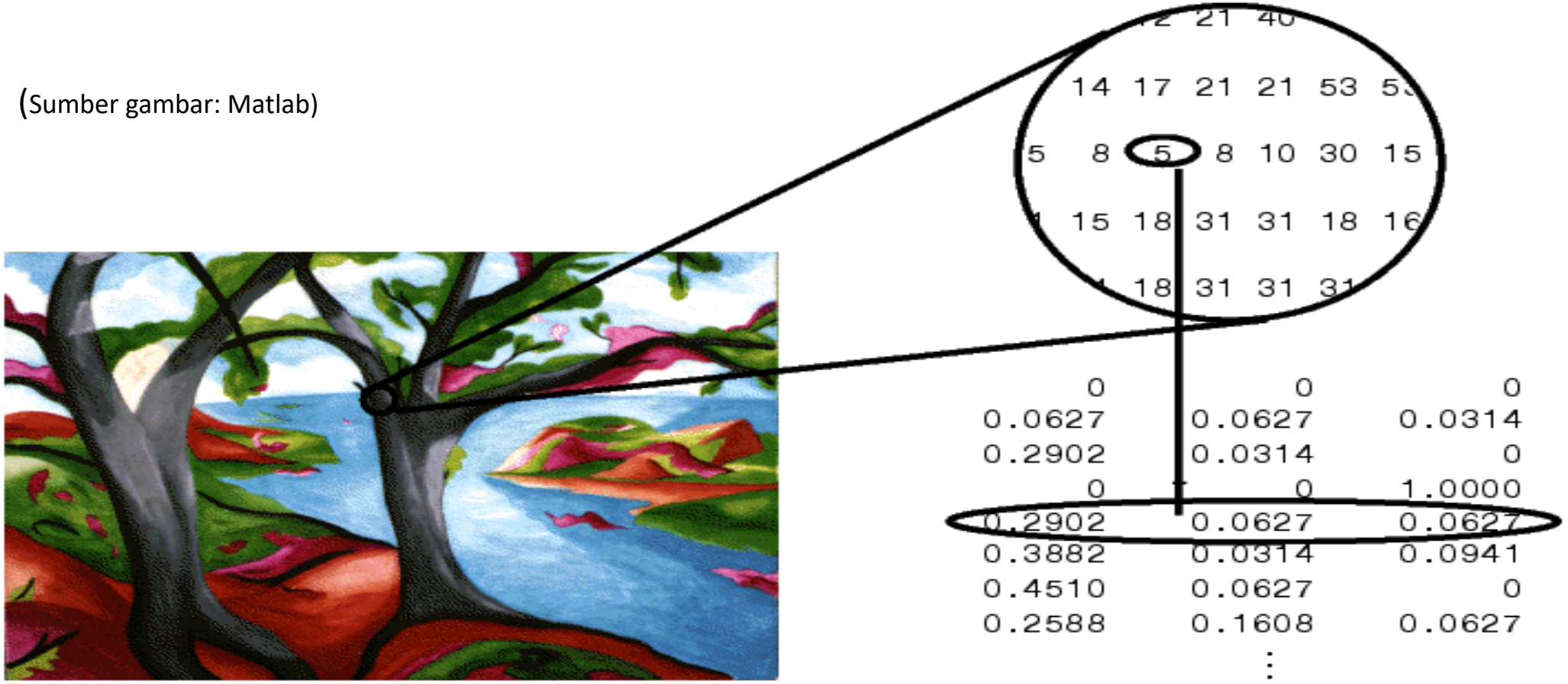
10010011100101001000101 0

R G B

Indexed Image

- GIF adalah salah satu *indexed image*
- Citra berformat GIF terdiri dari sebuah palet warna (*colormap*) dan sebuah matriks yang nilai elemennya menyatakan indeks ke sebuah baris di dalam palet.
- Palet adalah matriks *m* x 3 bertipe *floating-point* dengan nilai di dalam [0, 1]. Untuk citra dengan 256 warna, maka *map* berukuran 256 x 3.
- Tiap baris di dalam palet menspesifikasikan komponen *red*, *green*, dan *blue* dari sebuah warna tunggal.
- Warna sebuah *pixel* adalah kombinasi setiap komponen *red* (R), *green* (G), dan *blue* (B) pada baris palet tersebut.
- GIF cocok untuk menyimpan grafik yang hanya memiliki beberapa warna seperti diagram, logo, dan kartun. Ada juga *animated GIF* yang disusun oleh beberapa *frame*.

(Sumber gambar: Matlab)



Pada gambar di atas, nilai *pixel* 5 menunjuk ke baris ke-5 dari palet. Nilai komponen warna pada baris ke-5 adalah R = 0.2902, G = 0.0627, dan B = 0.0627. Itu artinya warna pada *pixel* bernilai 5 merupakan kombinasi dari ketiga komponen R, G, dan B tersebut.

Animated GIF

JPEG

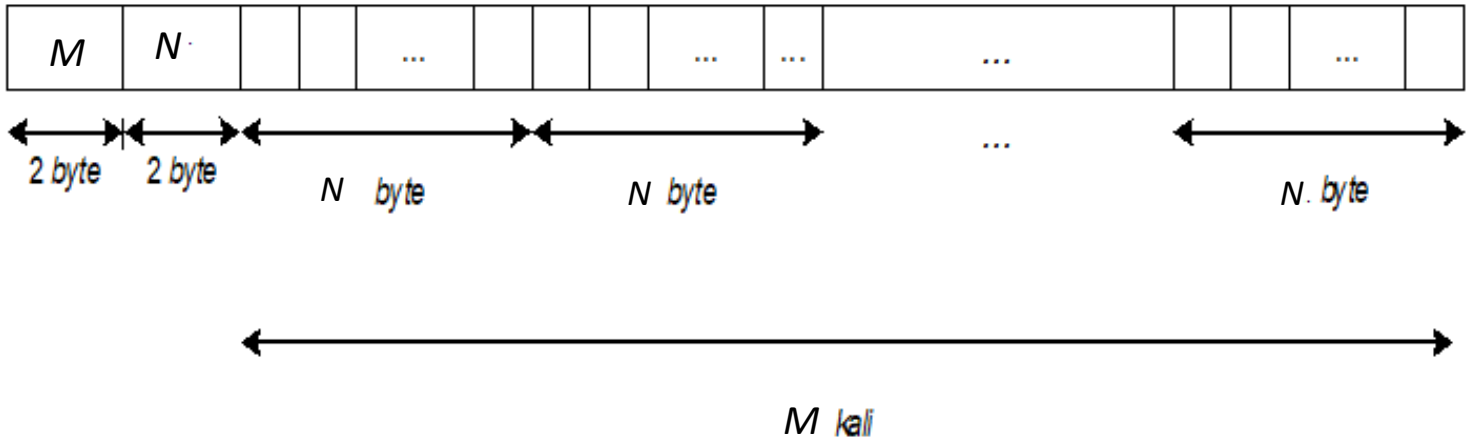
- JPEG adalah format citra terkompresi dengan metode kompresi yang *lossy,* yaitu DCT (*discrete cosine transform*).
- Ukuran file JPEG lebih kecil daripada file BMP tetapi dengan kompensasi penurunan kualitas secara visual.
- Ekstensi file: JPG atau JPEG
- Format citra umum untuk kamera digital
- Mendukung citra *grayscale* 8-bit dan citra berwarna 24-bit
- Varian dari JPEG adalah JPEG 2000 yang menggabungkan *lossless* dan *lossy compression*.

PNG

- PNG = Portable Network Graphics
- Merupakan format citra yang *free*, *open-source*, dan universal sehingga dapat digunakan secara luas di internet.
- Dibuat sebagai alternatif format GIF
- *Lossless compression*.
- Mendukung citra terindeks dengan palet 8-bit, 24-bit truecolor, dan 48-bit *truecolor*.

Format *Raw Image*

- *Raw image* atau citra mentah adalah format citra yang tidak mempunyai format spesifik.
- Hanya berisi informasi ukuran citra (tinggi x lebar atau *M* x *N*) dan data *bitmap*



- Citra mentah juga dapat disimpan di dalam file teks dalam format ASCII. Umumnya untuk citra *grayscale* 8-bit/*pixel.*
- Baris pertama menyatakan ukuran citra (M x N), baris-baris berikutnya menyatakan nilai-nilai *pixel*. Setiap nilai *pixel* dipisahkan dengan spasi.

```
9  11
148  162  175  182  189  194  195  193  195  195  197
148  164  174  176  185  189  191  191  196  194  195
144  159  167  176  178  185  188  191  196  194  197
128  147  157  168  173  179  182  184  191  191  192
119  134  148  160  164  170  179  176  181  189  185
145  124  142  151  160  168  169  174  180  182  183
172  120  140  153  157  169  171  178  180  182  182
196  120  129  144  152  158  167  170  177  176  178
204  144  116  134  142  149  155  165  165  170  171
```

Struktur Data untuk Citra Digital

- Di dalam kuliah ini tugas membuat program menggunakan Bahasa C/C++.
- Citra direpresentasikan sebagai matriks.
- Untuk citra dengan 256 derajat keabuan, nilai setiap elemen matriks adalah bilangan bulat di dalam selang [0, 255].
- Gunakan *unsigned char* untuk menyatakan tipe elemen matriks.
- Untuk citra yang berukuran M x N, deklarasi citra *f* adalah sebagai berikut:

```
#define M 500       /* Maksimum tinggi citra */
#define N 100       /* Maksimum lebar citra */
unsigned char f[M][N];
```

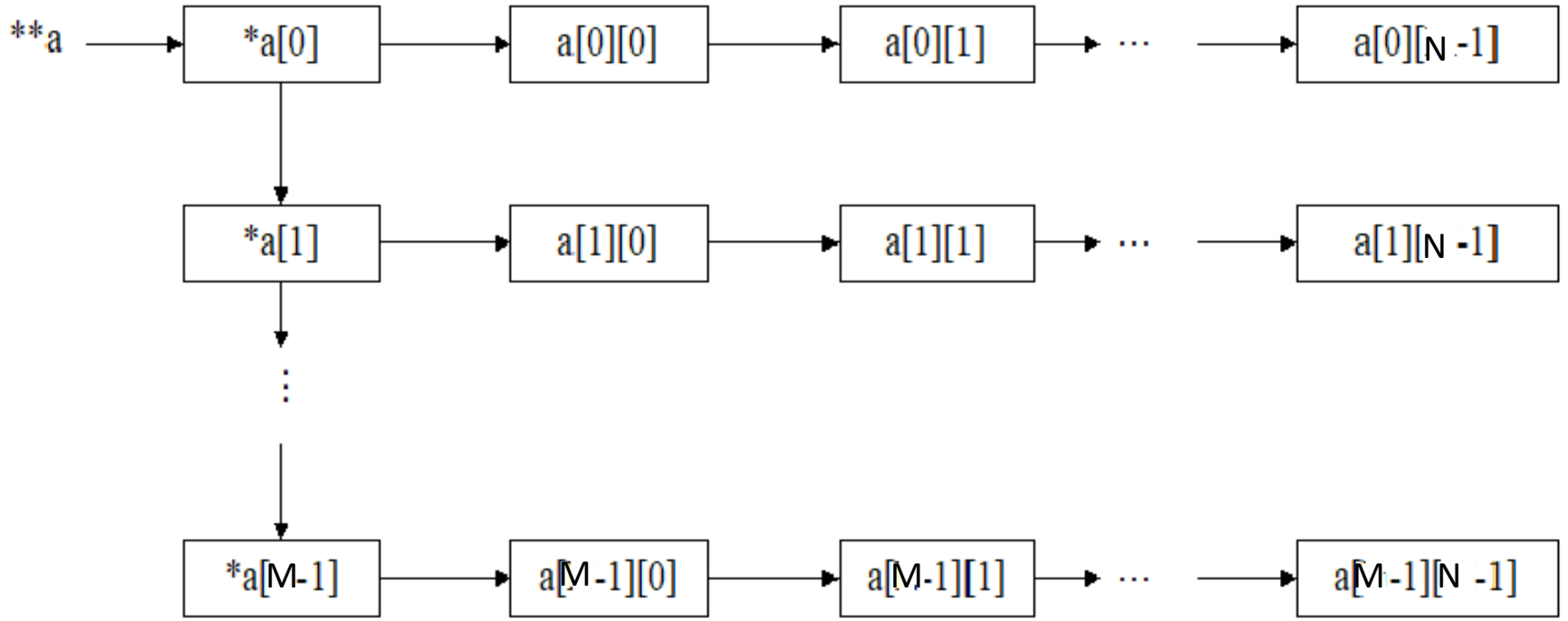
- Umumnya ukuran citra tidak diketahui sebelum *running program*, oleh karena itu struktur data yang cocok untuk citra adalah *pointer*.

- Di dalam Bahasa C, elemen larik `a[i]` ekivalen dengan `*(a+i)`. Tanda '*' menyatakan *pointer* yang menunjuk pada alamat suatu elemen di memori.

- Larik yang berukuran *M* elemen dapat dibuat secara dinamik pada saat *run-time* dengan prosedur alokasi `malloc`:

```
unsigned char *a;
a=(unsigned char*) malloc (M*sizeof(unsigned char));
```

- Matriks dapat dianggap sebagai larik satu matra (matra = dimensi) dari sebuah larik lain.



- *Pointer* **a menunjuk ke larik *pointer* *a[0], *a[1], …, *a[M-1], yang masing-masing larik menunjuk ke baris-baris dari citra

- Dengan demikian, deklarasi matriks dinamis untuk citra *f* adalah sebagai berikut ini:

```
unsigned char **f;
```

atau dengan menggunakan *user defined type*:

```
typedef unsigned char **citra;
citra f;
```

- Alokasi memori untuk matriks citra `f` yang berukuran *M* × N (M baris dan N kolom) dilakukan pada saat *run-time* dengan memanggil fungsi `alokasi` sebagai berikut:

```
citra alokasi(int M, int N)
/* Mengalokasikan memori untuk citra f yang berukuran M x N pixel. */
{
  int i;

  f=(unsigned char**)malloc(M * sizeof(unsigned char*));
  if (f==NULL) return(NULL);   /* memori habis */
  for (i=0; i<M; i++)
  {
    f[i]=(unsigned char*)malloc(N*sizeof(unsigned char));
    if (f[i]==NULL)
    { /* memori habis, dealokasi semua elemen baris matriks */
      dealokasi(f, N);
      return(NULL);
    }
  }
  return f;
}
```

- Cara pemanggilan fungsi `alokasi`:

```
f = alokasi(M, N)
```

- Untuk citra berwarna, yang mana setiap nilai intensitas merah, hijau, dan biru disimpan di dalam matriks `r`, `g`, dan `b`, maka kita harus mengalokasikan memori untuk ketiga buah matriks tersebut:

```
r = alokasi(M, N)
g = alokasi(M, N)
b = alokasi(M, N)
```

- Bila citra selesai diproses, maka memori yang dipakai oleh citra tersebut dikembalikan (dealokasi) kepada sistem.

- Dealokasi memori untuk matriks citra `f` dapat dilakukan dengan memanggil fungsi `dealokasi` berikut:

```
void dealokasi(citra f, int M)
/* Dealokasi memori dari citra f yang mempunyai M baris pixel */

{
  int i;

  for (i=0; i<M; i++)
  {
    free(f[i]); /* bebaskan memori semua elemen pada baris i */
  }
  free(f);
}
```

- Fungsi alokasi secara umum (tidak hanya untuk tipe *unsigned char*):

```
void **alokasi(int M, int N, int UkuranElemen)
/* Mengalokasikan memori untuk matriks M x N. Setiap elemen matriks berukuran UkuranElemen byte  */
{
  int i;
  void **larik = (void**)xalloc(M * sizeof(void *)); /* buat array M  elemen */
  for (i=0; i<M; i++)
    larik[i] = (void*)xalloc(N * UkuranElemen);
  return larik;
}

void *xalloc(unsigned ukuran)
/* Mengalokasikan memori dan memeriksa apakah alokasi memori berhasil */
{
  void *p = malloc(ukuran);
  if (p==NULL)
  {
    printf("Memori tidak cukup untuk alokasi matriks");
    exit(0);
  }
  return p;
}
```

Cara alokasi memori untuk citra *Image* bertipe *unsigned char* dan matriks *Mat* bertipe *float*, masing-masing berukuran $M \times N$ elemen:

```
Image = (unsigned char**) alokasi (M, N, sizeof(unsined char));
Mat = (float**) alokasi (M, N, sizeof(float));
```

Menampilkan Citra ke Layar

```
setpixel(unsigned char r,unsigned char g, unsigned char b,nt i,int j);
/* menampilkan pixel dengan komponen rgb pada koordinat i, j */
/* Perhatikan: setpixel bukan standard C */
```

Prosedur menampilkan citra ke layer:

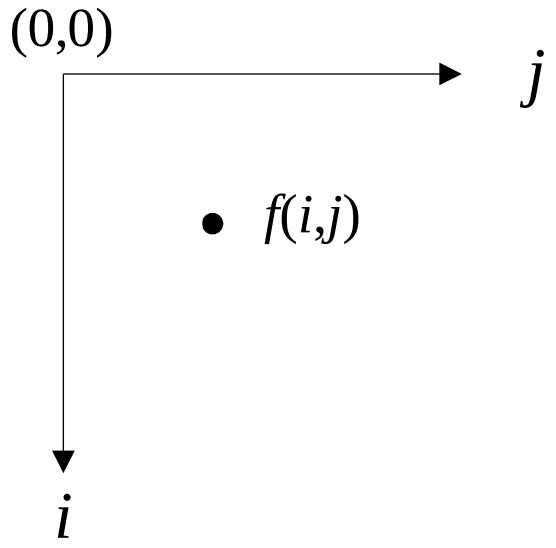
```
void tampilkan_citra(citra r,citra g,citra b, int M,int N)
/* Menampilkan citra yang berukuran M x N pixel ke layar. */
{ int i, j;

  for (i=0; i<M; i++)
     for (j=0; j<N; j++)
        setpixel(r[i][j],g[i][j],b[i][j],j,i);}
```

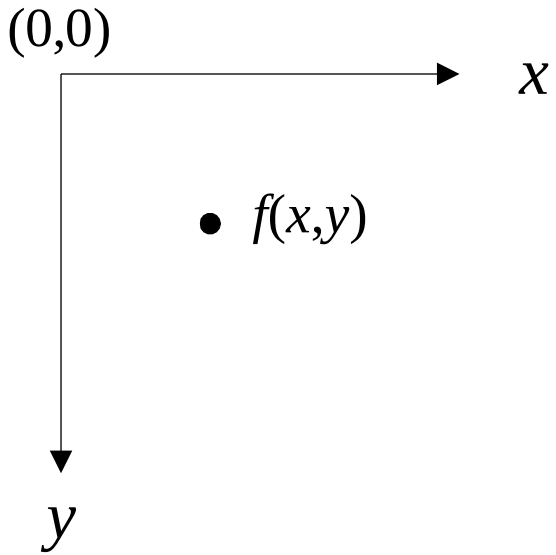
Jika citra yang ditampilkan adalah citra *grayscale*, maka perubahan yang dilakukan adalah pada:

```
setpixel(f[i][j],f[i][j],f[i][j], j, i);
```

- *Pixel* (*i*,*j*) ditampilkan pada posisi (*j*,*i*) di layar karena perbedaan sistem koordinat yang digunakan pada representasi citra dan layar peraga



Koordinat citra



Koordinat layar

- Menampilkan citra *grayscale* 8-bit pada *platform* Windows:

```
void WIN_tampilkan_citra(citra Image, int M, int N)
/* Menampilkan citra Image yang berukuran M x N di lingkungan Windows */
{
  HDC      MemDC;      /* Handle ke memory device context */
  HBITMAP  mbitmap;    /* Handle ke citra */
  HWND     hwnd;       /* Handle ke window */
  COLORREF TabelWarna[256];   /* Tabel warna (palet) */
  int i, j, palet;

  hwnd = GetActiveWindow();
  MemDC = CreateCompatibleDC(GetDC(hwnd));
  mbitmap = CreateCompatibleBitmap(GetDC(hwnd),N,M);
  SelectObject(MemDC,mbitmap);

  /* Definisikan palet */
  for (i=0; i<256; i++)
    TabelWarna[i]=GetNearestColor(MemDC, RGB(i,i,i));

  /* Isikan pixel ke memori device (layar) */
  for (i=0; i<M; i++)
    for (j=0; j<N; j++)
     {
       palet = Image[i][j];
       SetPixelV(MemDC,j,i,TabelWarna[palet]);
     }
   /* Tembakkan citra ke layar */
  BitBlt(GetDC(hwnd),0,0,N,M,MemDC,0,0,SRCCOPY);
}
```

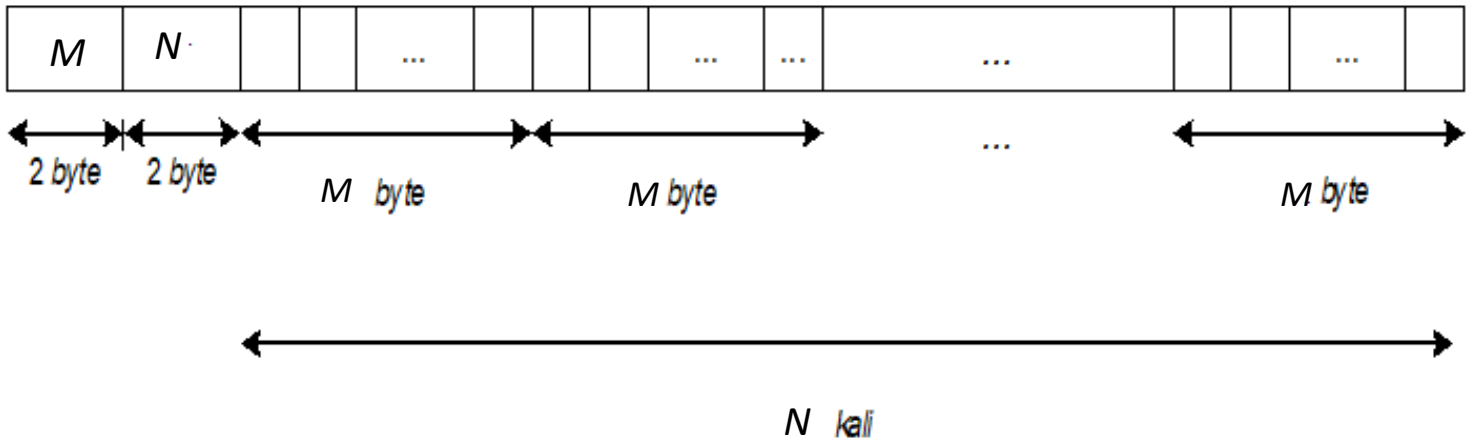
- Jika citra yang ditampilkan adalah citra berwarna (matriks intensitas *pixel*-nya masing-masing adalah *r*, *g*, dan *b*), maka perubahan yang dilakukan adalah sebagai berikut:

```
for (i=0; i<M; i++)
   for (j=0; j<N; j++)
    {
      palet = GetNearestColor(MemDC, RGB(r[i][j], g[i][j], b[i][j]);
      SetPixelV(MemDC, j, i, palet);
    }

/* Tembakkan citra ke layar */
BitBlt(GetDC(hwnd), 0, 0, M, N, MemDC, 0, 0, SRCCOPY);
```

Membaca Citra dari File Citra Mentah

Format file citra mentah:



```
void baca_citra_dari_arsip(char nama_arsip[], citra f)
/* Membaca citra dari file citra mentah. Citra hasil pembacaan disimpan di alam matriks f. */
{
  FILE *fp;
  int i, j;
  unsigned short int M, N;

  if((fp=fopen(nama_arsip, "rb"))==NULL)
  {
   printf("Arsip tidak ada");
   exit(0);
  }
  fread(&M, sizeof(unsigned short int), 1, fp);  /* baca tinggi citra */
  fread(&N, sizeof(unsigned short int), 1, fp);  /* baca lebar citra */

  f = alokasi(M, N)  /* alokasi memori matriks untuk citra f */
  if(f==NULL)
  { printf("Memori tidak cukup");
    exit(0);
  }

  /* baca data citra baris demi baris */
  for(i=0; i<M; i++)
  {
    /* baca data citra baris ke-i */
    fread(f[i], sizeof(citraunsigned char), N, fp);
  }
  close(fp);
}
```

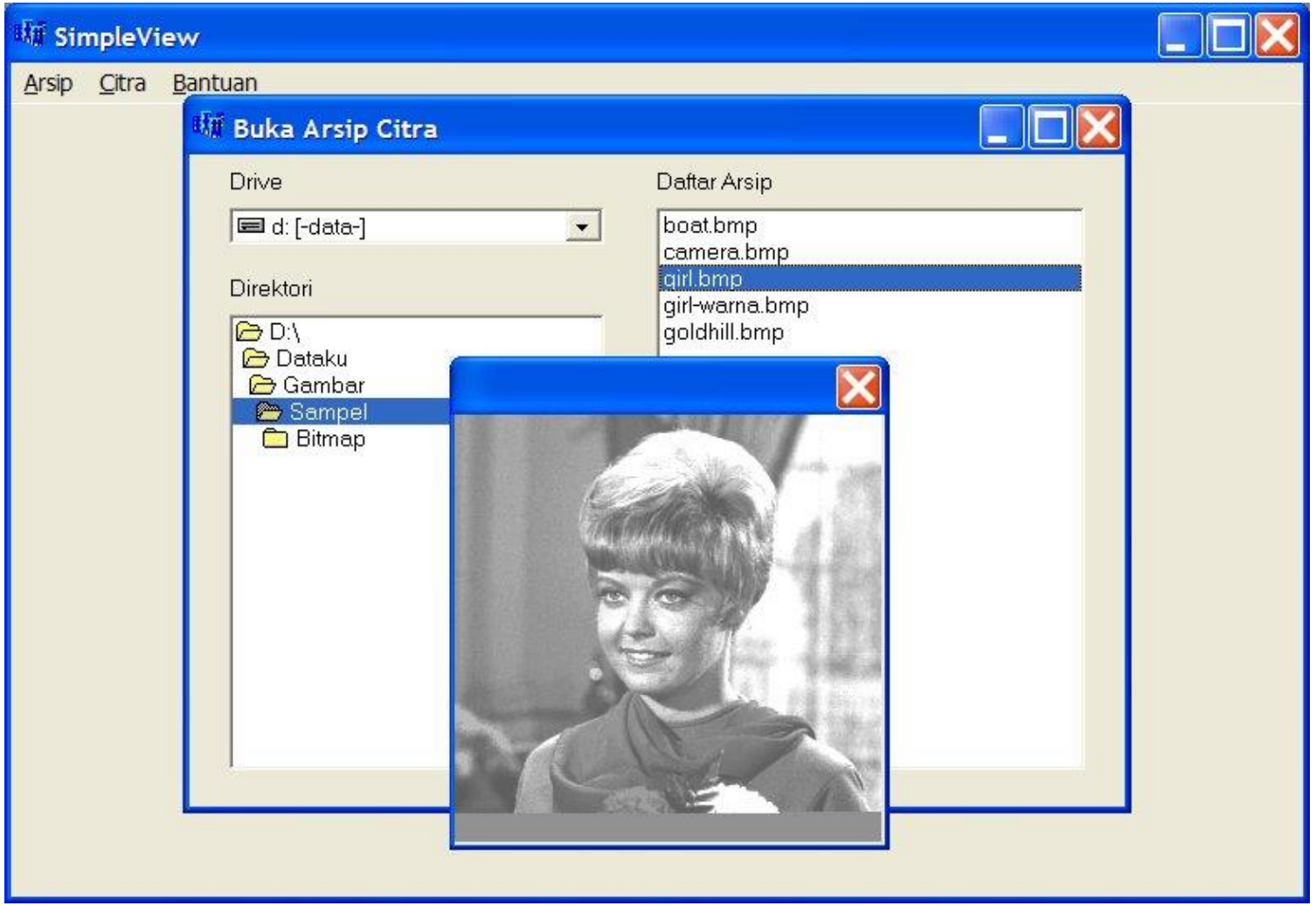
Menyimpan Citra ke dalam File Citra Mentah

```
void tulis_citra_ke_arsip(char nama_arsip[], citra f)
/* Menulis citra f ke dalam arsip nama_arsip. */
{
  FILE *fp;
  int i, j;
  unsigned short int M, N;

  if((fp=fopen(nama_arsip, "wb"))==NULL)
  {
    printf("Arsip tidak dapat dibuat");
    exit(0);
  }
  fwrite(M, sizeof(unsigned short int), 1, fp);  /* tulis tinggi citra */
  fwrite(N, sizeof(unsigned short int), 1, fp);  /* tulis lebar citra */

  /* baca data citra baris demi baris */
  for(i=0; i<M; i++)
  {
    /* tulis data citra baris ke-i */
    fwrite(f[i], sizeof(unsigned char), N, fp);
  }
  close(fp);
}
```

- Contoh antarmuka program membaca dan menampilkan citra yang ditulis dengan *Borland C++ Bulder* dan diberi nama *SimpleView*.



Pengolahan Citra dengan Matlab

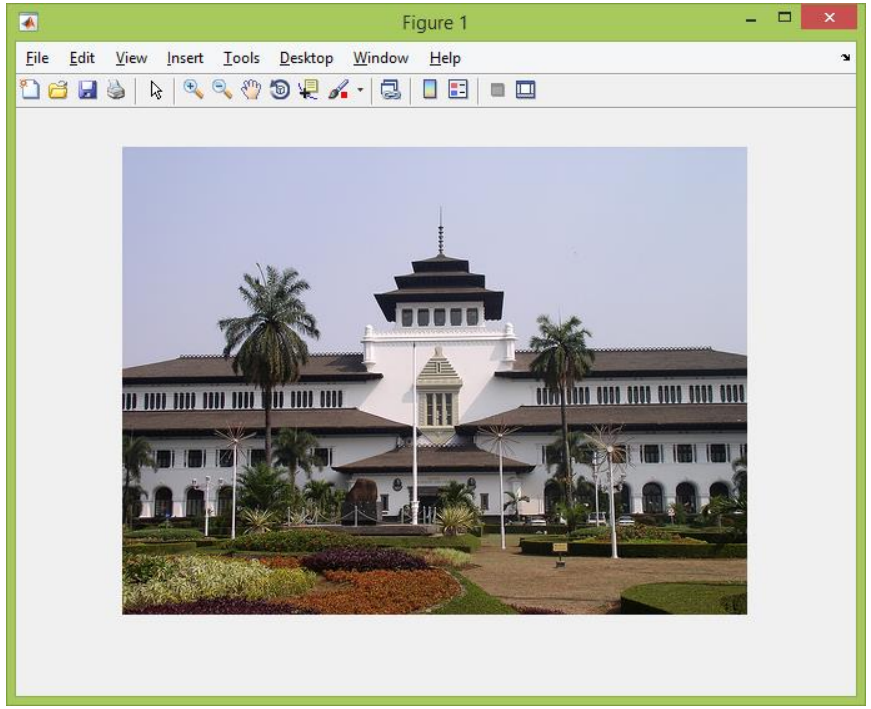
- Matlab memiliki *Image Processing Toolbox* yang dapat digunakan untuk pemrosesan citra digital.
- *Image processing toolbox* memiliki fungsi-fungsi *built-in* untuk pemrosesan citra.

- Di dalam Matlab, citra direpresentasikan dengan matriks M x N, namun indeks matriks dimulai dari 1, bukan dari nol. *Pixel* pada sudut kiri atas adalah pada posisi (1, 1).

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & \vdots & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix}$$

- Membaca citra dan menampilkan ke layar

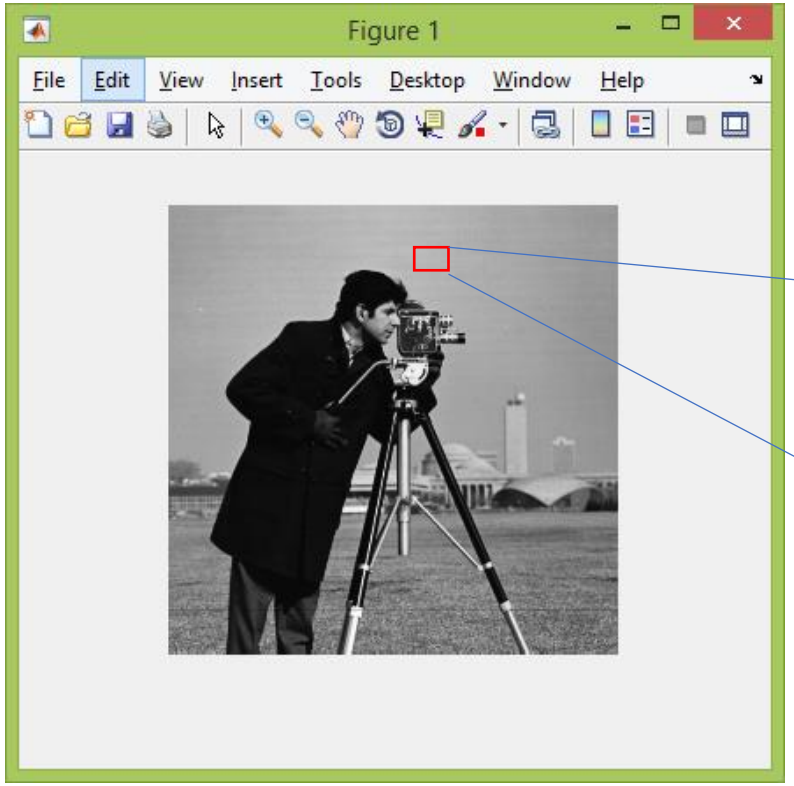
```
>> I = imread('gedung-sate.jpg');
>> imshow(I)
>> whos
Name          Size          Bytes    Class     Attributes
  I           374x500x3     561000   uint8
```



- Citra *grayscale*

```
>> img = imread('camera.bmp');
>> whos
  Name         Size                Bytes  Class    Attributes
  img        256x256               65536  uint8

>> imshow(img)
```



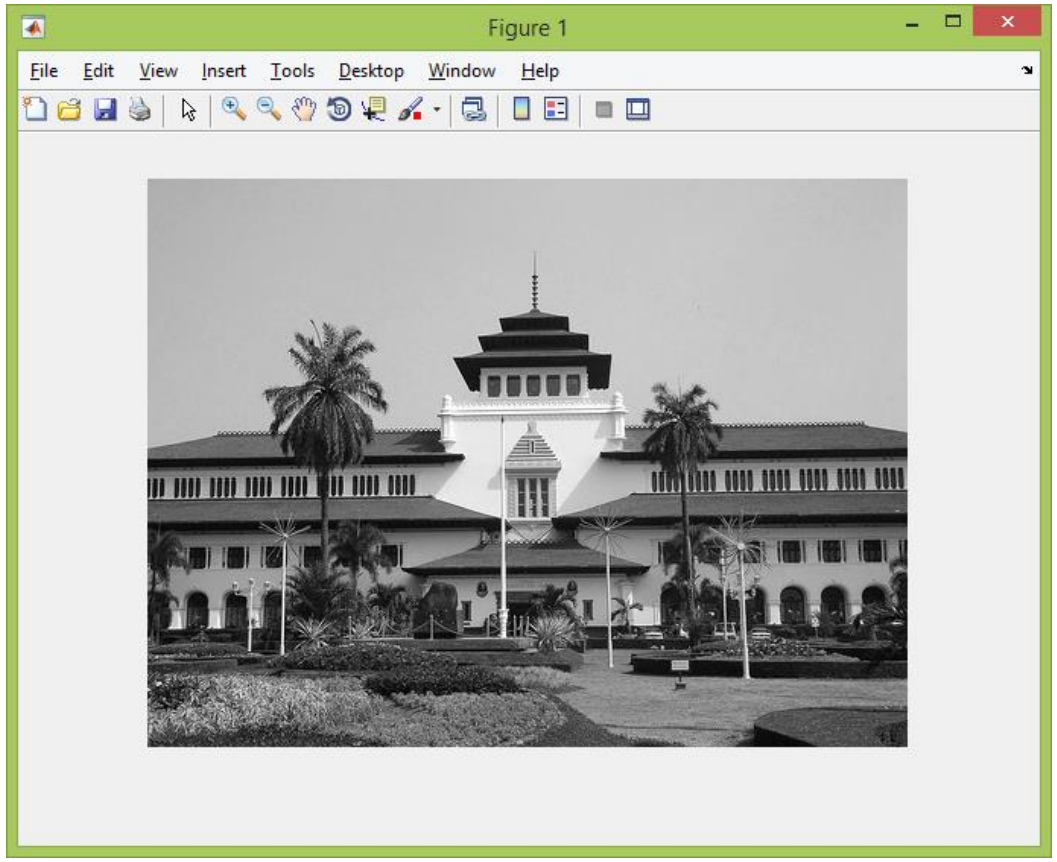

```
>> img(20:25, 100:110)

ans =

   183   184   184   183   185   185   185   188   189   187   188
   182   185   185   186   181   185   185   187   187   189   185
   184   183   185   185   179   187   186   184   185   189   187
   183   182   182   181   185   180   183   180   181   184   185
   180   183   183   182   189   184   186   185   186   186   185
   182   185   180   179   182   185   185   183   185   187   187
```

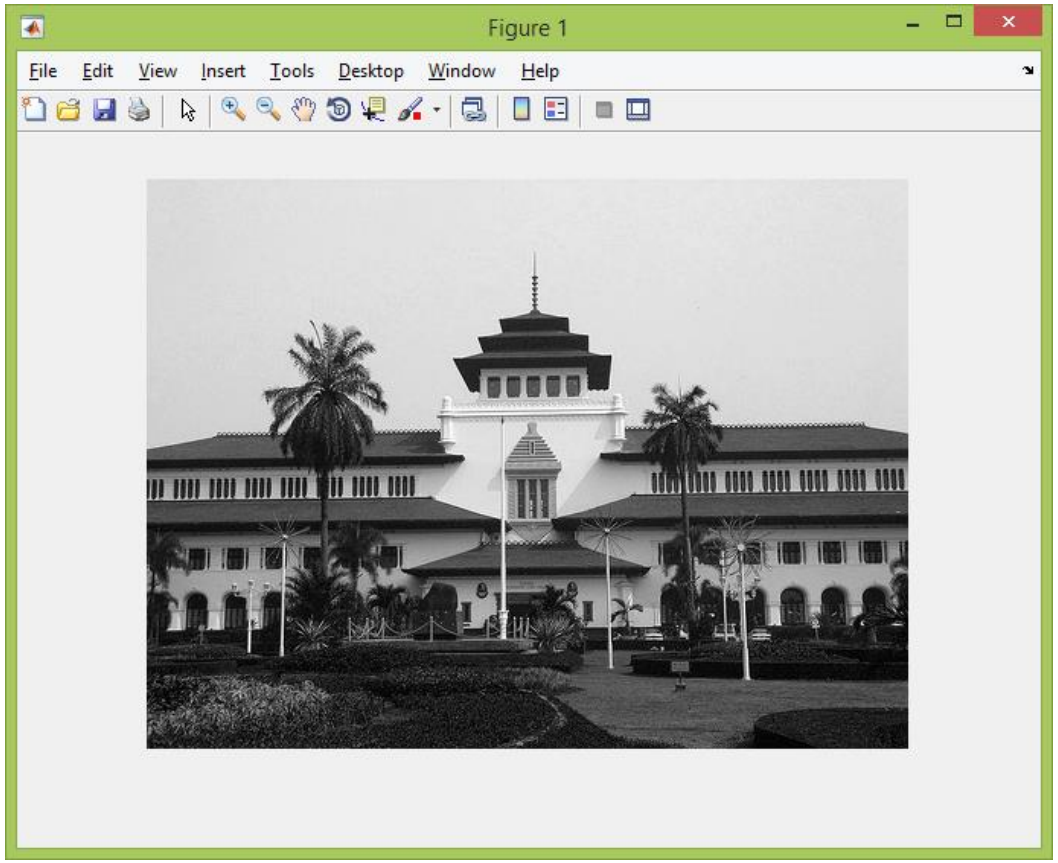
- Tampilkan hanya komponen *red* dari citra

```
>> imshow(I(:,:,1))
```



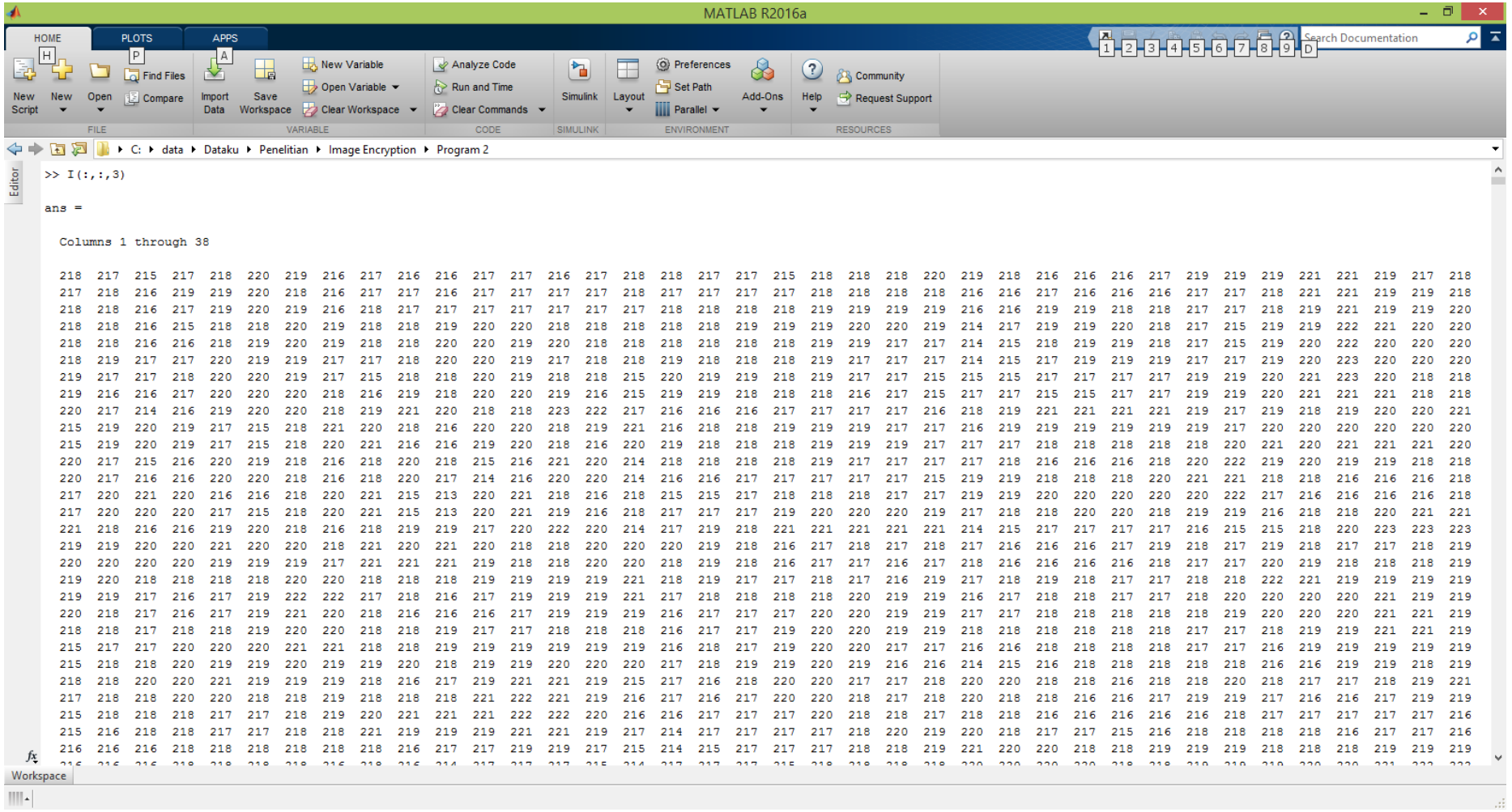
- Tampilkan komponen *blue*

```
>> imshow(I(:,:,3))
```



- Tampilkan data bitmap

```
>> I(:,:,3)
```

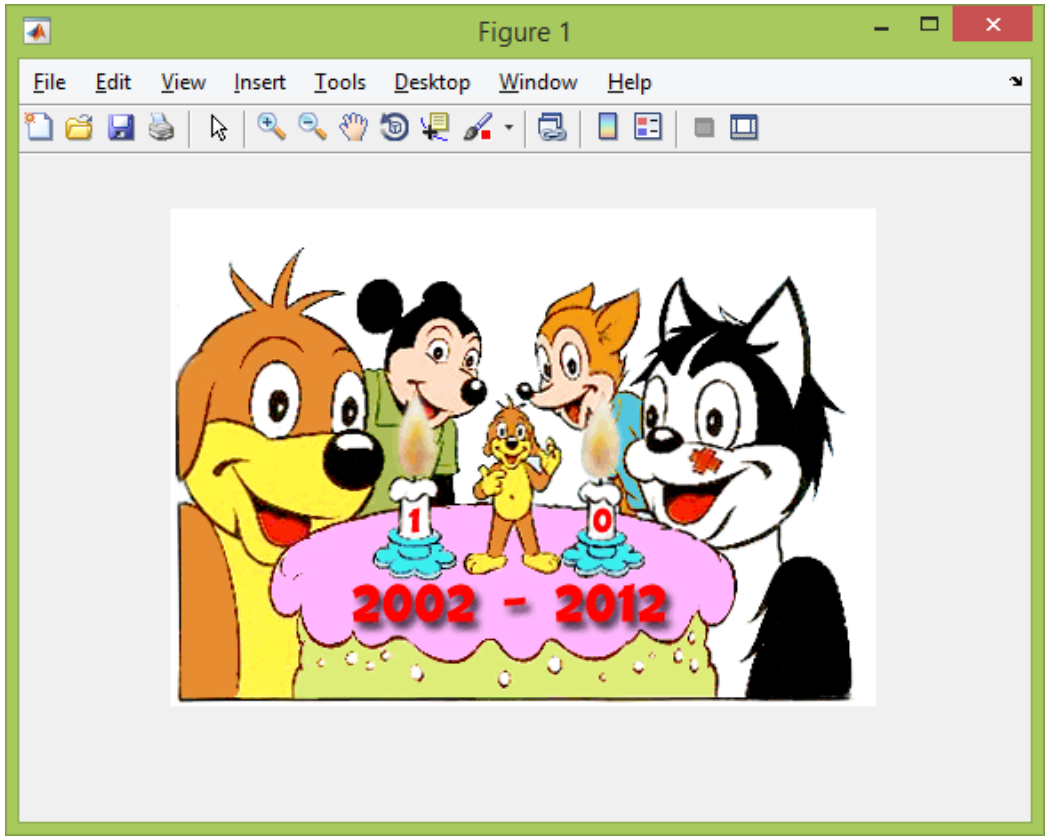


- Membaca citra GIF

```
>> [C, map] = imread('kartun.gif');
>> whos
  Name         Size                Bytes  Class     Attributes
  C          280x397             111160  uint8
  map        256x3                 6144  double
>> imshow(C, map)
>> map

map =

    0.9137    0.9608    0.5412
    0.0039         0         0
    0.8471    0.3882    0.0863
    0.9961    0.0039         0
    0.6980    0.6980    0.6902
    0.8078    0.5765    0.8039
    0.6627    0.8392    0.8824
    1.0000    0.8706    0.7843
    0.9647    0.6118    0.0627
    0.6784    0.7412    0.3490
    0.9608    0.9333    0.6902
       …         …         …
```



- Menyimpan citra

```
>> imwrite(img,'camera.jpg','jpg');
```

- Membaca citra *animated GIF*

```
[citra map]=imread('walk.gif', 'frames','all');

s = size(citra);
numframes=s(4);

for n=1:numframes;
    A = citra(:,:,:,n);
    figure; imshow(A,map);
end
```

- Frame-frame citra `walk.gif`