# BAB IV

# ANALISIS DAN PEMECAHAN MASALAH

## 4.1. Membaca File PCX

Struktur file PCX 16 warna terdiri dari dua bagian, yaitu: Header, dan Raster Data. Struktur dari file PCX tersebut adalah seperti yang tampak pada gambar 2.3. Dimana susunan file PCX tersebut adalah tetap dan tidak dapat dipertukarkan. Sehingga dalam proses pembacaan file PCX, urutan pembacaan selalu dimulai dari Header, diikuti dengan bagian Raster Data.

### 4.1.1. Membaca header

Header pada file PCX mempunyai ukuran sebesar 128 byte. Header pada file PCX adalah bagian dari file PCX yang berisi informasi mengenai data citra yang disimpan di dalamnya. Informasi tersebut antara lain mengenai versi dari file PCX, jumlah bit per piksel, resolusi citra yang disimpan dan sebagainya. Untuk itu, sebelum melakukan pembacaan terhadap data citra yang disimpan di dalam file PCX, terlebih dahulu harus dibaca Header dari file PCX tersebut. Adapun struktur dari Header pada file PCX adalah seperti yang tampak pada tabel 2.4.

Field-field pada Header file PCX seperti yang terdapat dalam tabel 2.4 hanyalah merupakan suatu nama variabel. Sehingga untuk pembacaan, nama-nama variabel yang digunakan untuk membaca field-field tidak harus sama dengan nama-nama variabel seperti yang disebutkan pada tabel di atas. Yang terpenting adalah bahwa setiap variabel yang digunakan untuk membaca field-field Header harus mempunyai jenis dan lebar data yang sama, sehingga tidak terjadi adanya kesalahan pada waktu pembacaan. Adapun deklarasi tipe data struktur tersebut harus sesuai dengan struktur yang dimiliki oleh header file PCX. Dalam bahasa C,

deklarasi tersebut adalah sebagai berikut:

```
#define MAXPALETTECOLORS      16
typedef unsigned char BYTE;

/* Definisi tipe data struktur untuk palette warna VGA */
typedef struct {
      BYTE   Red;              /* Komponen warna RGB */
      BYTE   Green;
      BYTE   Blue;
} ColorRegister;

struct PCXFileHeader {
      BYTE          Manufacturer;   /* Identitas file PCX */
      BYTE          Version;        /* Versi file PCX. Menentukan penggunaan palette */
      BYTE          Encoding;       /* Mode kompresi file */
      BYTE          BitPerPixel;    /* Jumlah bit per piksel */
      unsigned      XMin;           /* Ukuran citra*/
      unsigned      YMin;
      unsigned      XMax;
      unsigned      YMax;
      unsigned      HRes;           /* Resolusi horizontal */
      unsigned      VRes;           /* Resolusi vertikal */
      ColorRegister ColorMap[MAXPALETTECOLORS];      /* Palette warna VGA */
      BYTE          Reserved;       /* Dapat diabaikan */
      BYTE          NPlanes;        /* Jumlah bit plane */
      unsigned      BytesPerLine;   /* Jumlah byte per baris dalam citra */
      unsigned      PaletteType;    /* 1 = grayscale; 2 = warna */
      BYTE          Unused[58];     /* Melengkapi ukuran header menjadi 128 byte */
};
struct PCXFileHeader PCXHeader;     /* Variabel header file PCX */
FILE        *PCXFile;               /* Variabel File */
```

Setelah deklarasi tipe data struktur untuk header file PCX, dan deklarasi variabel

dengan tipe struktur tersebut maka untuk proses pembacaan header file PCX tersebut dalam bahasa C dapat dinyatakan dengan:

```
/* Membaca header file PCX */
if( fread(&PCXHeader,sizeof(struct PCXFileHeader),1,PCXFile)!=1) {
      printf("Error reading PCX file header");
      exit(1);
};
```

Dalam proses pembacaan header file PCX ini, dapat dipakai fungsi fread yang disediakan oleh bahasa C. Fungsi fread ini akan menghasilkan nilai jumlah elemen yang dibaca. Sehingga dengan pernyataan seperti tersebut di atas, jika tidak sama dengan nilai yang dihasilkan 1 (satu), berarti terjadi kesalahan pada saat proses pembacaan header file PCX. Kesalahan ini dapat disebabkan karena kerusakan pada bagian header file PCX atau yang lainnya. Jika terjadi kesalahan ini maka proses pembacaan file PCX akan segera dihentikan.

Setelah proses pembacaan header file PCX dilakukan dengan sukses, maka proses selanjutnya adalah memeriksa setiap field yang diperlukan dalam header tersebut.

Field yang pertama dari Header pada file PCX adalah Manufacturer atau kode pembuat. Selain extension pada nama file, hanya kode pembuat ini yang menandakan bahwa file yang dibaca adalah benar-benar file PCX. Kode pembuat yang mempunyai ukuran sebesar satu byte ini selalu berisi nilai 10 atau 0Ah, yang merupakan tanda bahwa file yang dibaca adalah benar-benar file PCX. Untuk memeriksa nilai dari field Manufacturer dalam bahasa C, dapat dinyatakan sebagai berikut:

```
/* Memeriksa identitas file PCX */
if(PCXHeader.Manufacturer != 0x0A) {
      printf("Error: Bukan file PCX\n");
      exit(1);
}
```

Dengan pernyataan seperti tersebut di atas, jika field Manufacturer tidak berisi 10 atau 0Ah maka file yang dibaca dianggap bukan merupakan file PCX, dan program akan menampilkan pesan bahwa file yang dibaca bukan file PCX dan dengan segera proses akan dihentikan. Dan jika kode pembuat atau Manufacturer berisi nilai 10 atau 0Ah, maka file yang dibaca merupakan file PCX dan akan dilanjutkan ke proses berikutnya.

Field berikutnya adalah field Version yang merupakan kode versi dari file PCX. Kode versi yang mempunyai ukuran sebesar 1 byte ini merupakan tanda dari jenis file PCX yang ada. Ada empat macam kode versi, yaitu:

0 untuk file PCX versi 2.5 PC Paintbrush

2 untuk file PCX versi 2.8 dengan palette

3 untuk file PCX versi 2.8 tanpa palette (menggunakan palette standar)

5 untuk file PCX versi 3.0 untuk file PCX high color atau true color

Field Version pada dasarnya hanya digunakan untuk menentukan penggunaan jenis palette pada saat menampilkan gambar, yaitu antara palette warna yang ada pada field ColorMap pada bagian header file atau Extended Color Palette yang disimpan pada bagian akhir dari file PCX. Karena untuk file PCX 16 warna tidak ada Extended Color Palette dan cukup hanya menggunakan palette yang ada pada field ColorMap dalam header file PCX, maka pemeriksaan field Version cukup

dilakukan dengan memeriksa apakah terdapat palette warna pada file PCX tersebut. File PCX yang memiliki palette warna adalah file PCX yang pada field Version-nya berisi nilai selain 3, yaitu file PCX yang tidak termasuk file PCX versi 2.8 tanpa palette.

Setelah field Version, field selanjutnya adalah Encoding. Field Encoding selalu berisi nilai 1, yang merupakan tanda bahwa data citra yang disimpan pada bagian Data Raster ditulis dengan menggunakan metode kompresi, dimana metode kompresi yang digunakan dalam file PCX ini adalah metode RLE (Run Length Encoding). Field Encoding ini mempunyai ukuran sebesar 1 byte.

BitsPerPixel adalah field yang berisi informasi mengenai jumlah bit per piksel. Jumlah bit per piksel dalam file PCX tergantung dari jumlah warna maksimum yang dapat ditampilkan oleh citra yang disimpan dalam file PCX. Untuk file PCX dengan 16 warna field BitsPerPixel ini berisi nilai satu. Field ini bersama-sama dengan field NPlanes digunakan untuk menentukan jumlah warna pada file PCX. Field BitsPerPixel ini mempunyai ukuran sebesar 1 byte.

Pada empat field berikutnya, yaitu XMin, YMin, XMax, dan YMax, adalah merupakan posisi citra saat ditampilkan. Dengan masing-masing dari setiap field tersebut mempunyai ukuran sebesar 2 byte. XMin dan YMin biasanya mempunyai nilai 0, yang berarti data citra yang disimpan akan ditampilkan mulai posisi 0,0 atau berada di pojok kiri atas layar. Sedangkan XMax dan YMax akan berisi nilai lebar dan tinggi citra yang disimpan. Jika XMin dan YMin berisi nilai 0, akan XMax dan YMax akan berisi nilai maksimum lebar dan tinggi dikurangi 1. Biasanya field ini bukan dipakai untuk menentukan posisi citra saat ditampilkan, tetapi dipakai untuk menentukan ukuran atau dimensi dari citra yang disimpan pada bagian data raster.

Dua field berikutnya, HRes dan VRes, merupakan field yang menyimpan

informasi mengenai resolusi citra yang disimpan, dimana HRes merupakan resolusi horizontal dan VRes merupakan resolusi vertikal dari citra. HRes dan VRes ini mempunyai ukuran masing-masing sebesar 2 byte. Kedua field HRes dan VRes ini biasanya dapat diabaikan dalam proses pembacaan.

ColorMap adalah field yang berisi palette warna untuk file PCX 16 warna atau kurang. Palette warna yang mempunyai ukuran sebesar 48 byte ini, terdiri dari 3 komponen warna Red, Green dan Blue, yang masing-masing komponen warna mempunyai ukuran sebesar 16 byte. Dengan ukuran sebesar 16 byte untuk masing-masing komponen warna Red, Green dan Blue, maka ColorMap dapat dipakai untuk menyimpan palette warna sebanyak 16 warna pilihan untuk setiap komponen warna Red, Green dan Blue.

Reserved yang menempati ruang sebesar 1 byte ini adalah field yang tidak dipakai dan dapat diabaikan.

NPlanes adalah field yang berisi informasi mengenai jumlah color plane. Color plane dalam adaptor tampilan jenis VGA merupakan halaman memori. Dalam adaptor tampilan jenis VGA terdapat empat buah halaman memori. Keempat color plane tersebut, masing-masing menyimpan sebuah bit informasi warna yang secara berurutan Intensitas warna, Red, Green, dan Blue. Dengan membaca setiap komponen warna dan menampilkan sesuai dengan komponen-komponen warna tersebut maka citra 16 warna akan dapat ditampilkan dengan lengkap. Field NPlanes ini menempati ruang sebesar 1 byte.

Field BytePerLine adalah field yang menyimpan informasi mengenai jumlah byte yang ada untuk setiap baris dalam setiap plane. Field ini bersama-sama dengan field NPlanes digunakan untuk menghitung ukuran dari bagian Data Raster. Dengan mengalikan nilai dalam field BytePerLine dan NPlanes akan dapat diketahui jumlah

byte data yang harus dibaca untuk setiap baris pada bagian data raster.

PaletteType adalah field yang memberitahukan apakah data citra akan ditampilkan sebagai citra gray scale atau citra berwarna. Untuk citra gray scale maka field ini akan berisi nilai 1, sedangkan untuk citra berwarna field ini akan berisi nilai 2. Field PaletteType ini mempunyai ukuran sebesar 2 byte. Karena pada tugas akhir ini menggunakan file PCX 16 warna, maka field ini dapat diabaikan.

Sisa dari bagian Header, yaitu field Unused, sebesar 58 byte terakhir adalah kosong dan dapat diabaikan.

### 4.1.2. Membaca data raster

Setelah proses pembacaan bagian Header file PCX dan pemeriksaan terhadap hasilnya selesai, proses pembacaan file PCX dilanjutkan dengan membaca data citra yang disimpan pada bagian Data Raster. Bagian ini mempunyai ukuran yang bervariasi untuk setiap file PCX, tergantung dari data citra yang disimpan. Besarnya ukuran penyimpanan suatu data citra dipengaruhi oleh ukuran atau dimensi dari citra yang disimpan dan jumlah warna yang dapat ditampilkan oleh citra yang disimpan tersebut.

Dalam pengolahan citra, karena data citra yang dibaca tidak hanya untuk ditampilkan saja melainkan juga perlu diproses, maka data citra yang dibaca harus di simpan ke dalam buffer yang dialokasikan dalam memori komputer. Dari buffer ini data citra yang ada dapat ditampilkan dan atau diproses sesuai dengan kebutuhan pengolahan citra. Adapun alokasi memori untuk buffer tersebut dalam bahasa C dapat dinyatakan sebagai berikut:

```
ImageMemory = (BYTE huge *) farcalloc((long) ImageWidth * ImageHeight, sizeof (BYTE));
```

Dalam kenyataannya, karena keterbatasan dari memori dan besarnya buffer yang dibutuhkan untuk penyimpanan data citra, maka pernyataan di atas belum cukup. Jika alokasi memori untuk buffer hanya menggunakan pernyataan seperti tersebut diatas, akan dapat mengakibatkan terjadinya kesalahan pada waktu eksekusi jika memori yang dibutuhkan untuk alokasi memori untuk buffer tidak cukup. Untuk mengatasi jika memori yang tersedia tidak mencukupi, maka pernyataan tersebut di atas dapat di sesuaikan sehingga menjadi seperti di bawah ini:

```
ImageMemory = (BYTE huge *) farcalloc((long) ImageWidth * ImageHeight, sizeof (BYTE));
if (ImageMemory == NULL) {
    printf("Error: Memory untuk buffer tidak cukup\n");
    exit(1);
}
```

Dengan memanfaatkan fungsi farcalloc yang akan menghasilkan sebuah pointer yang menunjuk ke lokasi buffer di memori jika memori yang tersedia cukup dan nilai NULL jika memori yang tersedia tidak mencukupi untuk alokasi buffer. Dengan pernyataan seperti tersebut di atas, maka jika memori yang dibutuhkan untuk alokasi buffer tidak mencukupi, program akan menampilkan pesan bahwa memori untuk buffer tidak cukup dan dengan segera akan menghentikan proses.

Penyimpanan data citra pada file PCX selalu menggunakan kompresi data dengan algoritma RLE. Adanya kompresi data ini ditandai dengan nilai 1 pada field Encoding pada bagian header file. Kompresi data pada file PCX ini dilakukan diterapkan untuk bagian Data Raster. Sehingga pada saat melakukan pembacaan pada bagian Data Raster ini, harus dilakukan dekompresi data dengan algoritma

RLE juga. Dengan dilakukannya dekompresi data untuk proses pembacaan bagian Data Raster ini akan dapat diperoleh hasil yang benar, sesuai dengan yang diharapkan.

Algoritma RLE adalah algoritma kompresi data yang melakukan kompresi data dengan membuat kode untuk byte data perulangan menjadi sebesar 2 byte. Byte yang pertama digunakan untuk menyimpan tanda perulangan dan jumlah perulangan, dimana dua bit tertinggi digunakan untuk menyimpan tanda perulangan dan enam bit terendah digunakan untuk menyimpan jumlah perulangan. Sedangkan byte kedua digunakan untuk menyimpan data yang diulang.

Tanda perulangan yang ditempatkan pada dua bit tertinggi dari byte yang pertama adalah berupa di-set-nya kedua bit tertinggi tersebut. Sedangkan enam bit terendah dari byte pertama adalah merupakan jumlah perulangan yang harus dilakukan untuk byte yang kedua. Dari enam bit terendah yang tersedia untuk jumlah perulangan ini hanya akan menghasilkan maksimum 63 kali perulangan untuk suatu byte data. Sehingga jika ada byte data yang membutuhkan lebih dari 63 kali perulangan akan membutuhkan lebih dari satu kode perulangan.

Kelemahan dari kompresi data dengan algoritma RLE ini terlihat jika terdapat sebuah byte data yang dua bit tertingginya di-set. Untuk kondisi seperti ini, data tersebut tidak dapat dikompresi melainkan akan dibentuk menjadi suatu perulangan sebanyak satu kali. Sehingga bukan kompresi data yang terjadi, melainkan pemekaran data dari satu byte menjadi dua byte data.

Deretan bit-bit data pada bagian Data Raster ini berisi nilai-nilai yang merupakan nilai intensitas dari piksel-piksel yang membentuk citra. Banyaknya jumlah bit yang dibutuhkan untuk membentuk sebuah piksel tergantung dari jumlah warna maksimum yang dapat ditampilkan oleh citra tersebut. Untuk file PCX 16 warna

setiap piksel diwakili oleh empat bit data. Dengan demikian jika untuk sekali pembacaan digunakan sebuah byte data, maka dalam sebuah byte data akan terdapat dua bagian yang membentuk dua piksel. Untuk itu hasil pembacaan bagian Data Raster dengan menggunakan byte data seperti tersebut di atas perlu dibentuk ulang agar setiap piksel yang ada hanya diwakili oleh satu buah byte data. Pembentukan ulang ini dapat dilakukan dengan cara mengambil setiap empat bit data dan meletakkannya dalam satu buah byte data. Pembentukan ulang ini diperlukan untuk mempermudah proses menampilkan dan proses pengolahan yang akan dilakukan terhadap data citra tersebut.

Untuk melakukan proses dekompresi pada setiap baris Data Raster dapat dibentuk fungsi yang dinyatakan dalam bentuk *pseudo-code* sebagai berikut:

```
BytesToRead = Jumlah Plane Warna * Jumlah Byte per Baris
InPtr = 0
WHILE InPtr < BytesToRead DO
    READ data Char
    IF (byte data AND 11000000)=11000000 THEN
            RepCount = Char AND 00111111
            WHILE RepCount > 0 DO
                ScanLine[InPtr] = Char
                InPtr = InPtr + 1
            ENDWHILE
    ELSE
            ScanLine[InPtr] = Char
            InPtr = InPtr + 1
    ENDIF
ENDWHILE
```

Di dalam file PCX bagian Data Raster ini adalah sama dengan deretan bit-bit data yang membentuk matriks dua dimensi. Dimana elemen kolom dari matriks tersebut

menandakan lebar citra dan elemen baris dari matriks menandakan tinggi dari citra yang disimpan. Sedangkan untuk membaca bit-bit data yang ada pada bagian Data Raster ini dapat digunakan perulangan dengan sebuah byte data yang digunakan sebagai penampung sementara. Digunakannya sebuah byte data sebagai penampung sementara ini untuk mempermudah proses pembacaan bagian Data Raster, yang jumlah setiap barisnya diketahui dalam satuan byte sebagai hasil dari perkalian jumlah color plane dan jumlah byte per baris. Sedangkan banyaknya baris adalah sama dengan nilai tinggi citra yang dapat diketahui dari pengurangan field YMax dan field YMin yang ada pada bagian Header dan banyaknya kolom dapat diketahui dari pengurangan field XMax dan field XMin. Adapun pernyataan untuk membaca bagian Data Raster dalam *pseudo-code* adalah sebagai berikut:

```
READ data XMin, XMax, YMin dan YMax dari citra
ImageW = XMax - XMin + 1
ImageH  = YMax - YMin + 1
DO Row = 0 TO ImageH
    READ Baris_Kompresi_Data
    Index = Row * ImageW
    DO Col = 0 TO ImageW
        ImageMemory[Index + Col] = Kompresi-Data
    ENDDO
ENDDO
```

Pembacaan bagian Data Raster ini dilakukan baris demi baris. Dalam proses membaca setiap baris tersebut juga dilakukan proses dekompresi baris data raster. Proses tersebut dilakukan hingga seluruh baris data raster yang ada selesai dibaca seluruhnya. Kemudian hasil dari pembacaan data raster tersebut diletakkan dalam variabel memori seperti yang sudah dideklarasi di atas.

Setelah data raster berada di dalam variabel pointer tersebut, dapat dilakukan proses selanjutnya terhadap citra yang tersimpan didalamnya, baik untuk proses menampilkan atau pun untuk pengolahan data citra.

## 4.2. Merekam Citra ke Bentuk File PCX

Merekam citra ke dalam bentuk file bitmap dengan format PCX harus sesuai dengan struktur yang dimiliki oleh file PCX, agar citra yang disimpan tersebut dapat dibaca dan ditampilkan kembali dengan benar. Seperti yang sudah kita ketahui file PCX 16 warna memiliki struktur yang terdiri dari dua bagian yaitu bagian Header, dan Raster Data. Oleh karena itu untuk merekam citra ke dalam file PCX harus sesuai dengan urutan mulai dari bagian Header, kemudian diikuti oleh bagian Data Raster.

Data citra yang akan direkam ke dalam file PCX disini, adalah data citra yang berasal dari proses pengolahan atau pembacaan citra sebelumnya. Oleh karena itu data citra yang akan direkam ke dalam file PCX ini adalah data citra yang berada di dalam buffer memori, baik data bagian header maupun data raster.

### 4.2.1. Merekam header

Dalam proses perekaman Header file PCX ini, urutan field-field Header harus sesuai dengan susunan header seperti pada tabel.

Yang pertama dimulai dari field Manufacturer yang harus diisi dengan nilai 10 atau 0Ah.

Berikutnya adalah field Version yang dapat diisi dengan nilai yang berasal dari field Version pada file sumber atau nilai version yang dimiliki file PCX selain nilai 3.

Field Encoding diisi dengan nilai 1, karena dalam penulisan file PCX ini selalu dilakukan kompresi data dengan algoritma RLE.

Field BitsPerPixel yang menandakan jumlah bit yang membentuk sebuah piksel ini akan selalu berisi nilai satu untuk file PCX 16 warna.

Untuk dua field berikutnya yaitu XMin dan YMin dapat diisi dengan nilai 0. Sedangkan untuk field XMax diisi dengan nilai lebar citra dikurangi satu dan YMax diisi dengan nilai tinggi citra dikurangi satu. Nilai lebar citra dan tinggi citra yang diisikan pada field XMax dan YMax disini adalah nilai lebar dan nilai tinggi citra tujuan. Nilai lebar citra dapat diketahui dari perkalian antara nilai lebar citra sumber dengan faktor skala, sedangkan nilai tinggi citra dihasilkan dari perkalian nilai tinggi citra sumber dengan faktor skala.

Field HRes dan field VRes yang merupakan resolusi horizontal dan vertikal citra diisi dengan nilai lebar dan tinggi citra dari citra tujuan.

Field ColorMap, yang berisi informasi palette warna dapat diisi dengan nilai ColorMap yang berasal dari citra sumber. Nilai ColorMap pada citra sumber sama dengan nilai ColorMap pada citra tujuan, karena dalam proses pengolahan citra pada tugas akhir ini hanya dilakukan untuk mengubah ukuran atau dimensi dari citra sumber. Sehingga semua nilai warna, baik yang terdapat dalam nilai piksel maupun dalam palette warna adalah tetap.

Field Reserved yang pada waktu proses pembacaan dapat diabaikan, dalam proses penulisan ini harus diisi dengan nilai 0.

Field NPlanes yang merupakan informasi jumlah color plane dapat diambilkan dari nilai NPlanes yang ada pada file PCX sumber.

Field BytePerLine adalah field yang berisi informasi mengenai jumlah byte dalam satu baris. Nilai ini dapat diperoleh dari file PCX sumber.

Field PaletteType diisi dengan nilai 2, yang menandakan bahwa citra yang disimpan dalam file PCX adalah citra berwarna.

Field terakhir yaitu field Unused yang mempunyai ukuran sebesar 58 byte dapat diisi dengan nilai 0. Dalam hal ini seluruh ruang yang ada pada field Unused harus diisi dengan nilai 0, agar field Unused tetap mempunyai ukuran sebesar 58 byte. Hal ini dilakukan agar header file yang terbentuk mempunyai ukuran sebesar 128 byte.

```
PCXData.Manufacturer    = 0x0A;
PCXData.Version         = 5;
PCXData.Encoding        = 1;
PCXData.BitPerPixel     = BitsPerPixel;
PCXData.XMin            = 0;
PCXData.YMin            = 0;
PCXData.XMax            = MaxX-1;
PCXData.YMax            = MaxY-1;
PCXData.HRes            = MaxX;
PCXData.VRes            = MaxY;
PCXData.Reserved        = 0;
PCXData.NPlanes         = Planes;
PCXData.BytesPerLine    = BytesPerLine;
getpalette(&palette);
for( Index = 0; Index<palette.size; Index++ ) {
      regs.h.ah = 0x10;
      regs.h.al  = 0x15;
      regs.x.bx = palette.colors[Index];
      int86(VIDEO,&regs,&regs);
      PCXData.ColorMap[Index].Red   = regs.h.dh <<= 2;
      PCXData.ColorMap[Index].Green = regs.h.ch <<= 2;
      PCXData.ColorMap[Index].Blue  = regs.h.cl <<= 2;
};
memset(&PCXData.Unused,'\0',sizeof(PCXData.Unused));
```

Setelah semua field yang ada sudah diinisialsasi, maka informasi header file tersebut dapat dituliskan ke dalam media penyimpanan. Dalam bahasa C penulisan

header tersebut dapat dilakukan dengan memanfaatkan fungsi fwrite. Adapun bentuk pernyataan dalam bahasa C tersebut adalah sebagai berikut:

```
if( fwrite(&PCXData,sizeof(struct PCXFileHeader),1,PCXFile)!=1 ) {
      printf("Unable to write PCX file header");
      exit(1);
}
```

Dengan pernyataan seperti tersebut di atas, jika terjadi kesalahan pada saat proses perekaman header file, maka proses akan dihentikan.
Jika perekaman header file tersebut sukses, maka proses dapat dilanjutkan dengan merekam data citra ke dalam bagian raster data dalam file PCX.

### 4.2.2. Merekam data citra

Dalam tugas akhir ini data citra yang ditulis ke dalam bentuk file PCX adalah data citra yang sudah berada di dalam buffer yang dialokasikan di dalam memori komputer. Untuk itu proses merekam data citra disini adalah sama dengan memindahkan data citra dari memori ke dalam media penyimpanan dalam bentuk file. Data citra tersebut dituliskan ke dalam file PCX pada bagian Data Raster. Tentu saja proses ini dilakukan setelah proses perekaman header file PCX selesai.

Data citra yang akan direkam ke dalam media penyimpanan harus dikompresi terlebih dahulu dengan algoritma RLE yang menjadi standar dari file PCX. Kompresi data dengan algoritma RLE adalah kompresi data yang melakukan kompresi dengan cara mengkodekan data yang diulang secara berurutan menjadi sebuah data sebesar 2 byte. Byte pertama digunakan untuk menyimpan tanda perulangan dan jumlah perulangan, dimana tanda perulangan ini menempati 2 bit tertinggi sedangkan jumlah perulangan menempati 6 bit terendah dari byte pertama. Sedangkan byte

yang kedua digunakan untuk menyimpan data yang diulang. Kompresi data dengan algoritma RLE ini hanya dapat dilakukan untuk data yang diulang maksimum sebanyak 63 kali, karena bagian perulangan hanya menempati ruang sebesar 6 bit. Sehingga untuk data yang diulang lebih dari 63 kali, akan dilakukan pengkodean lebih dari satu.

Untuk melakukan kompresi data dengan algoritma RLE ini, maka data yang akan dikenai kompresi, dibaca satu persatu, byte per-byte. Jika ada data yang diulang dan data tersebut lebih kecil atau sama dengan 63, maka akan dilakukan pengkodean terhadap data perulangan tersebut. Pengkodean dilakukan dengan mengisi 2 bit tertinggi dari byte pertama dengan 1, dan 6 bit berikutnya dengan jumlah perulangan yang ada. Kemudian byte data yang diulang diletakkan pada byte yang kedua. Hasil pengkodean ini kemudian disimpan ke dalam media penyimpanan. Jika byte data yang dibaca bukan merupakan suatu data perulangan, maka byte tersebut ditulis apa adanya ke dalam media penyimpanan.

Tanda perulangan pada kompresi data dengan algoritma RLE ini ditandai dengan 2 bit tertinggi yang diisi dengan nilai 1. Jika terdapat suatu byte data yang 2 bit tertingginya mengandung nilai 1 (11XXXXXX), maka byte data tersebut akan mendapat perlakuan khusus, yaitu dianggap sebagai suatu byte data yang diulang sebanyak 1 (satu) kali. Dengan adanya perlakuan khusus ini bukan kompresi data yang terjadi, melainkan suatu pemekaran data dari 1 (satu) byte menjadi 2 (dua) byte. Adapun fungsi untuk kompresi data citra yang dinyatakan dalam *pseudo-code* adalah sebagai berikut:

```
BytesToWrite = Jumlah Plane Warna * Jumlah Byte per Baris
OutPtr = 0
WHILE OutPtr < BytesToWrite DO
    Char = ScanLine[OutPtr]
    OutPtr = OutPtr + 1
    RepCount = 1
    WHILE ScanLine[OutPtr] = Char AND
            RepCount < MaxRepCount AND
            OutPtr < BytesToWrite DO
            RepCount = RepCount + 1
            OutPtr = OutPtr + 1
    ENDWHILE
    IF RepCount > 1 OR Char > 10111111 THEN
            RepCount = RepCount OR 11000000
            WRITE RepCount ke OutFile
    ENDIF
    WRITE Char ke OutFile
ENDWHILE
```

Setelah data yang akan dituliskan ke dalam media penyimpanan dikompresi, maka langkah selanjutnya adalah menuliskan data yang sudah terkompresi tersebut ke dalam media penyimpanan. Jika seluruh data citra yang ada sudah dituliskan ke dalam media penyimpanan, maka proses menulis citra ke dalam bentuk file PCX ini selesai.

## 4.3. Menampilkan Citra

Proses pengolahan citra adalah ilmu pengetahuan yang berhubungan dengan citra dan data citra. Menampilkan citra ke layar monitor adalah salah satu proses yang mutlak dilakukan dalam rangkaian proses penglahan citra. Dengan ditampilkannya citra ke layar monitor, maka kita dapat melakukan pengamatan terhadap citra, baik citra sebelum proses dan citra sesudah proses pengolahan.

Pada proses menampilkan citra ini, data citra yang akan ditampilkan sudah

berada di dalam buffer yang dialokasikan dalam memori komputer. Oleh karena proses menampilkan citra ke layar monitor ini merupakan suatu proses yang menuliskan setiap byte data yang mewakili nilai piksel ke layar monitor.

### 4.3.1. Memeriksa keberadaan adaptor tampilan

Untuk dapat menampilkan suatu citra grafis dibutuhkan adaptor tampilan yang mendukung mode grafik. Adapun adaptor tampilan yang dibutuhkan untuk menampikan citra disini adalah adaptor tampilan yang memiliki mode grafik dengan resolusi yang sesuai. Selain itu adaptor tampilan tersebut harus memiliki kemampuan dimana baik palette dan color register-nya dapat diatur sesuai keperluan. Seperti tampak pada tabel, maka hanya adaptor tampilan jenis VGA yang dapat digunakan untuk menampilkan citra grafis ke layar monitor.

Untuk mencegah terjadinya suatu kesalahan yang fatal pada proses menampilkan citra ke layar monitor, maka perlu diperiksa terlebih dahulu keberadaan adaptor tampilan jenis VGA sebelum proses menampilkan citra tersebut dilakukan.

Untuk memeriksa keberadaan adaptor tampilan jenis VGA, dapat digunakan fungsi yang dimiliki oleh video BIOS yang memanfaatkan interrupt 10h yaitu fungsi 1Ah. Dengan memanfaatkan fungsi 1Ah, maka fungsi untuk memeriksa keberadaan adaptor tampilan tersebut bila dituliskan dalam bahsa C adalah seperti berikut ini:

```
int CheckVGAAdapter( void ) {
    static union REGS regs;
    regs.h.ah = 0x1A;
    regs.h.al = 0;
    int86(0x10,&regs,&regs);
    if( (regs.h.al==0x1A) && (regs.h.bl==0x08) )
            return(1);
    else    return(0);
};
```

Dengan fungsi seperti tersebut di atas, maka jika adaptor tampilan terpasang pada sistem, fungsi tersebut akan mengembalikan nilai benar dan nilai salah untuk kondisi sebaliknya.

### 4.3.2. Mengaktifkan mode grafik

Setelah adaptor tampilan jenis VGA dapat dipastikan keberadaannya, maka proses selanjutnya adalah mengubah mode yang aktif menjadi mode grafik yang sesuai untuk menampilkan citra ke layar monitor. Dalam hal ini digunakan mode grafik dengan resolusi 640x480 16 warna.

Untuk mengubah ke mode grafik dengan resolusi 640x480 16 warna, dapat digunakan fungsi 00h dari interrupt 10h yang ada pada Video BIOS. Selain itu kompiler Turbo C juga menyediakan fungsi yang dapat dipakai untuk mengaktifkan mode grafik dengan resolusi 640x480 16 warna pada adaptor tampilan jenis VGA.

Fungsi yang disediakan oleh kompiler Turbo C dalam file pustaka graphics.lib tersebut adalah fungsi initgraph. Dengan memanfaatkan fungsi initgraph ini dapat dibentuk sebuah fungsi yang melakukan inisialisasi mode grafik dan sekaligus melakukan penanganan bila terjadi kesalahan pada saat proses inisialisasi mode grafik.

```
void InitGraphics( void ) {
    int GDriver, GMode, GErr;

    GDriver = GMode = GErr = 0;
    initgraph(&GDriver,&GMode,"");
    GErr = graphresult();
    if( GErr<0 ) {
            restorecrtmode();
            printf("Initgraph error: %s.\n",grapherrormsg(GErr)); exit(1);
    }
}
```

Dengan adanya fungsi InitGraphics seperti tersebut di atas, maka proses inisialisasi mode grafik dilakukan dengan cara mendeteksi adaptor tampilan yang ada. Dimana bila digunakan cara ini maka fungsi initgraph akan melakukan pemanggilan terhadap fungsi detectgraph. Dimana deteksi terhadap jenis adaptor tampilan yang ada dilakukan oleh fungsi detectgraph ini. Jika terjadi kesalahan pada proses inisialisasi mode grafik, maka fungsi InitGraphics di atas akan mengambalikan ke mode teks dan akan menampilkan pesan kesalahan proses inisialisasi grafik.

### 4.3.3. Mengaktifkan palette warna

Agar citra yang ditampilkan pada layar monitor memiliki warna-warna yang sesuai, maka palette warna yang disimpan dalam file PCX 16 warna harus diaktifkan. Cara mengaktifkan palette warna tersebut adalah dengan menuliskan informasi palette warna ke dalam memori adaptor tampilan. Hal ini dapat dilakukan dengan memanfaatkan fungsi yang disediakan oleh Video BIOS dan kompiler Turbo C.

Proses mengaktifkan paletter warna ini dapat dilakukan baik sebelum atau sesudah citra ditampilkan ke layar monitor. Jika proses mengaktifkan ini dilakukan sesudah menampilkan citra, maka waktu antara sesudah proses menampilkan citra dan sebelum proses mengaktifkan palette warna, akan terlihat citra di layar monitor dengan warna-warna yang tidak benar. Hal ini meskipun tidak mengganggu jalannya program, tetapi cukup mengganggu keindahan tampilan dari program. Lain halnya jika proses mengaktifkan palette warna dilakukan sebelum proses menampilkan citra. Pada saat citra tampil di layar monitor, citra tersebut sudah tampil dengan warna-warna yang benar, sehingga tidak akan mengganggu keindahan tampilan dari program.

Video BIOS menyediakan fungsi 10h sub fungsi 12h dari interrupt 10h. Fungsi

ini mendefinisikan register warna ke dalam DAC (Digital Analog Converter). Fungsi 10h subfungsi 12h dari interrupt 10h ini memiliki beberapa parameter, yaitu nomor fungsi yang diletakkan pada register AH yaitu 10h, nomor sub fungsi yang diletakkan pada register AL yaitu 12h, nomor awal dari DAC color register yang ditempatkan pada register BX, jumlah color register yang diaktifkan yang diletakkan pada register CX, serta pasangan alamat segment dan offset dari buffer, yang berisi register warna, yang ditempatkan pada register ES dan DX. Buffer yang berisi palette warna tersebut adalah field ColorMap yang terdapat dalam header file PCX. Seperti yang sudah dijelaskan sebelumnya, field ColorMap ini mempunyai lebar data sebanyak 48 byte, yang terdiri atas tiga bagian komponen warna Red, Green, dan Blue, yang masing-masing mempunyai lebar data sebesar 16 byte.

Kompiler Turbo C menyediakan fungsi setallpalette. Fungsi setallpalette ini memiliki sebuah parameter yang berisi informasi palette yang mempunyai tipe data struktur seperti yang didefinisikan dalam file pustaka graphics.lib. Adapun tipe data struktur untuk palette warna tersebut adalah:

```
struct palettetype {
    unsigned char size;
    signed character color[MAXCOLOR+1];
}
```

MAXCOLOR adalah konstanta nilai warna maksimum dalam palette warna yang didefinisikan dalam file pustaka graphics.lib.

Mengaktifkan palette warna untuk file PCX 16 warna harus dilakukan baik dengan fungsi 10h sub fungsi 12h dari interrupt 10h dan juga dengan fungsi setallpalette yang disediakan oleh kompiler Turbo C. Untuk itu dapat dibentuk satu

fungsi tersendiri yang akan mengaktifkan palette warna tersebut.

```
unsigned InstallPCXFilePalette( void ) {
    struct  palettetype palette;
    union   REGS regs;
    unsigned Index;

    if( PCXData.Version!=3 ) {
        palette.size = MAXPALETTECOLORS;
        for( Index = 0; Index<MAXPALETTECOLORS; Index++) {
            palette.colors[Index] = Index;
            PCXData.ColorMap[Index].Red   >>= 2;
            PCXData.ColorMap[Index].Green >>= 2;
            PCXData.ColorMap[Index].Blue  >>= 2;
        }
        regs.h.ah = 0x10;
        regs.h.al = 0x12;
        regs.x.bx = 0;
        regs.x.cx = MAXPALETTECOLORS;
        _ES = FP_SEG(&PCXData.ColorMap);
        regs.x.dx = FP_OFF(&PCXData.ColorMap);
        int86(VIDEO,&regs,&regs);
        setallpalette(&palette);
        return(TRUE);
    }
    else return(FALSE);
}
```

Nomor nilai warna dimasukkan dalam variabel palette yang mempunyai tipe data struktur palettetype dan diaktifkan dengan menggunakan fungsi yang disediakan oleh kompiler Turbo C, setallpalette. Sedangkan Informasi nilai warna yang berasal dari field ColorMap pada header file PCX diaktifkan dengan fungsi 10h sub fungsi 12h dari interrupt 10h.

Setelah proses mengaktifkan palette warna selesai, maka citra yang

ditampilkan akan memiliki warna-warna yang benar, yang sesuai warna-warna yang dimiliki oleh citra tersebut.

### 4.3.4. Menuliskan piksel ke layar

Pada proses menampilkan citra, yang terjadi sebenarnya adalah menuliskan setiap piksel pembentuk citra ke layar monitor. Untuk menuliskan piksel ke layar monitor dapat digunakan fungsi-fungsi baik yang disediakan oleh Video BIOS atau pun yang disediakan oleh kompiler Turbo C yang didefinisikan dalam file pustaka graphics.lib.

Fungsi yang disediakan oleh Video BIOS yaitu fungsi 0Ch dari interrupt 10h. Fungsi ini mempunyai beberapa parameter yaitu nomor fungsi yang diletakkan pada regiter AH, nomor halaman layar yang diletakkan pada register BH, baris layar yang diletakkan pada register DX, kolom layar yag diletakkan pada register CX, dan nilai piksel yang ditulis ke layar yang diletakkan pada register AL. Selain fungsi dari Video BIOS dapat juga dipakai fungsi yang disediakan oleh kompiler Turbo C, yaitu putpixel. Fungsi putpixel ini membutuhkan parameter kolom layar, baris layar dan nilai piksel yang akan ditulis ke layar monitor.

Piksel yang dituliskan ke layar monitor adalah piksel yang membentuk citra. Nilai piksel yang membentuk citra tersebut berasal dari proses pembacaan dari file PCX yang telah disimpan ke dalam variabel pointer. Dalam hal menuliskan piksel ke layar monitor cukup digunakan salah satu fungsi saja yaitu fungsi yang disediakan oleh Video BIOS atau fungsi yang disediakan oleh kompiler Turbo C. Untuk mempermudah proses implementasi dalam tugas akhir ini digunakan fungsi yang disediakan oleh Turbo C, yaitu putpixel. Adapun penggunaan fungsi putpixel tersebut di buat dalam satu fungsi yang mengerjakan proses untuk menampilkan

citra secara keseluruhan.

```
void DisplayImageInBuf( unsigned XStart, unsigned YStart, BYTE huge *Image,
                        unsigned WithPalette ) {
register unsigned ScanNum, PixelNum;
unsigned long Index;

if( WithPalette )
InstallPCXFilePalette();
for( ScanNum=0; ScanNum<ImageH; ScanNum++ )
    for( PixelNum=0; PixelNum<ImageW; PixelNum++ ) {
        Index  = ScanNum;
        Index *= ImageW;
        Index += PixelNum;
        putpixel(XStart+PixelNum,YStart+ScanNum,Image[Index]);
    }
}
```

Fungsi DisplayImageInBuf diatas adalah fungsi yang digunakan untuk mengerjakan proses menampilkan citra ke layar monitor. Dalam fungsi DisplayImageInBuf tersebut dilakukan juga pemanggilan terhadap fungsi InstallPCXFilePalette yang akan mengaktifkan palette warna untuk citra yang akan ditampilkan.

### 4.4. Pengolahan Citra

Pengolahan citra adalah ilmu pengetahuan yang berhubungan dengan citra dan data citra. Dalam tugas akhir ini disajikan pengolahan citra yang membahas cara mengubah ukuran citra. Poses mengubah ukuran citra disini adalah proses memperbesar dan memperkecil citra. Proses memperbesar dan memperkecil citra ini dapat digolongkan ke dalam proses geometri. Proses yang termasuk ke dalam golongan proses geometri ini adalah proses pengolahan citra yang melakukan proses pengaturan atau penempatan piksel dalam suatu citra berdasarkan pada transformasi

geometri.

Dalam tugas akhir ini baik untuk proses memperbesar dan atau proses memperkecil citra dilakukan terhadap data citra yang berada di dalam buffer yang dialokasikan ke dalam memori komputer. Untuk itu sebelum proses dilakukan, baik untuk proses memperbesar dan atau proses memperkecil, citra sumber yang berasal dari file PCX 16 warna terlebih dahulu harus dibaca dan dimasukkan ke dalam buffer. Setelah data citra berada di dalam buffer baru dapat dilakukan proses pengolahan citra yang diinginkan.

Selain membutuhkan masukkan berupa data citra, baik untuk proses memperbesar maupun proses memperkecil citra juga membutuhkan faktor skala sebagai masukan. Nilai faktor skala yang lebih besar dari satu digunakan untuk melakukan proses memperbesar citra, sedangkan nilai faktor skala yang lebih kecil dari satu digunakan untuk melakukan proses memperkecil citra. Dalam implementasi nilai ini dinyatakan dalam hitungan prosen untuk mempermudah pemakaian program. Nilai 1 dinyatakan sebagai 100%. Sehingga untuk nilai faktor skala yang lebih besar dari 100% digunakan untuk melakukan proses memperbesar citra dan nilai faktor skala yang lebih kecil dari 100% digunakan untuk melakukan proses memperkecil citra. Sedangkan jika dimasukkan nilai faktor skala sebesar 100% maka dianggap tidak memperbesar atau memperkecil citra.

Faktor skala baik untuk memperbesar maupun untuk proses memperkecil citra dibagi menjadi dua yaitu faktor skala horizontal dan faktor skala vertikal. Faktor skala horizontal mempengaruhi lebar citra, sedangkan faktor skala vertikal mempengaruhi tinggi citra. Kedua faktor skala yang ini dapat mempunyai nilai yang tidak sama, tetapi hal ini akan mengakibatkan citra hasil proses memiliki aspek rasio yang tidak proporsional. Untuk mendapatkan hasil berupa citra dengan ukuran

yang proporsional maka nilai masukan untuk faktor skala baik untuk skala horizontal atau pun skala vertikal harus sama.

Baik proses memperbesar maupun proses memperkecil citra dilakukan dengan jalan pemetaan. Pemetaan yang dilakukan ini merupakan pemetaan balik, yaitu pemetaan yang dilakukan dari sudut pandang citra tujuan. Dengan pemetaan balik, setiap piksel pada citra tujuan ditinjau satu persatu dan dicari nilainya berdasarkan pemetaan piksel-piksel yang berasal dari citra sumber. Setelah semua piksel selesai ditinjau dan dicari nilainya maka proses pengolahan citra selesai. Pemetaan balik ini dilakukan dengan tujuan agar setiap piksel pada citra tujuan dapat dipastikan mendapatkan sebuah nilai.

Untuk mencari nilai piksel yang ada di citra tujuan dilakukan berdasarkan metode yang dipakai untuk mengerjakan proses pengolahan citra. Dalam tugas akhir metode yang digunakan untuk melakukan proses mengubah ukuran citra adalah metode replikasi dan interpolasi linier.

### 4.4.1. Memperbesar citra

Proses memperbesar citra dapat dikerjakan dengan menggunakan beberapa metode, diantaranya metode replikasi dan metode interpolasi linier. Pada metode replikasi, setiap piksel yang ada pada citra sumber diperbesar menjadi *n*-piksel sesuai dengan faktor skala. Perbesaran ini disesuaikan dengan faktor skala yang bersangkutan. Jika faktor skala horizontal diperbesar, maka citra tujuan diperbesar sesuai dengan faktor skala horizontal. Sedangkan jika yang diperbesar adalah faktor skala vertikal, maka citra tujuan akan dipertinggi sesuai dengan faktor skala vertikal. Dalam bentuk *pseudo-code* penerapan metode replikasi ini adalah sebagai berikut:

```
DO DestRow = 0 TO DestHeight
    SRowAddr = DestRow / ScaleV
    SRowNum = bagian desimal dari SRowAddr
    DO DestCol = 0 TO DestWidth
        SColAddr = DestCol / ScaleH
        SColNum = bagian desimal dari SColNum
        OutImage[DestCol,DestRow] = InImage[SColNum,SRowNum]
    ENDDO
ENDDO
```

Pada metode interpolasi linier, setiap piksel pada citra tujuan yang nilainya dicari dihitung berdasarkan nilai-nilai piksel yang berada disekitarnya yang merupakan piksel-piksel pedukung, yang nilainya sudah diketahui dan dengan jarak antara piksel-piksel tersebut diketahui juga. Interpolasi linier menganggap bahwa piksel-piksel pendukung yang berada disekitar piksel yang nilainya dicari, berpengaruh secara langsung atau secara linier sesuai dengan jarak masing-masing piksel dari piksel yang nilainya dicari. Dengan kata lain piksel yang mempunyai jarak terdekat akan memiliki pengaruh yang terbesar dan sebaliknya pada piksel yang memiliki jarak paling jauh dari piksel yang dicari.

Interpolasi yang diterapkan untuk mencari sebuah nilai piksel Px pada citra tujuan adalah interpolasi yang dilakukan secara dua dimensi. Interpolasi yang dilakukan secara dua dimensi disini sebenarnya adalah interpolasi biasa yang dilakukan sebanyak 3 kali. Interpolasi yang pertama adalah interpolasi secara horizontal untuk piksel A dan B, dan interpolasi yang kedua adalah interpolasi secara horizontal untuk piksel C dan D, sedangkan interpolasi yang ketiga adalah interpolasi secara vertikal untuk hasil interpolasi horizontal A dan B dan hasil interpolasi horizontal C dan D.

Dalam bentuk *pseudo-code* penerapan metode replikasi ini adalah sebagai berikut:

```
DO DestRow = 0 TO DestHeight
    SRowAddr = DestRow / ScaleV
    SRowNum = bagian desimal dari SRowAddr
    RowDelta = SRowAddr - SRowNum
    DO DestCol = 0 TO DestWidth
        SColAddr = DestCol / ScaleH
        SColNum = bagian desimal dari SColNum
        ColDelta = SColAddr - SColNum
        PxA = InImage[SColNum,SRowNum];
        PxB = InImage[SColNum+1,SRowNum];
        PxC = InImage[SColNum,SRowNum+1];
        PxD = InImage[SColNum+1,SRowNum+1];
        IAB = ColDelta*((double)PxB - PxA) + PxA;
        ICD = ColDelta*((double)PxD - PxC) + PxC;
        OutImage[DestCol,DestRow] = IAB + (ICD - IAB) * RowDelta
    ENDDO
ENDDO
```

### 4.4.2. Memperkecil citra

Pada proses memperkecil citra dapat juga digunakan metode-metode yang telah digunakan pada proses memperbesar citra. Tentu saja metode-metode tersebut digunakan dengan cara yang dibalik.

Pada metode replikasi, setiap piksel ke-*n* dari citra sumber diambil dan diletakkan pada citra tujuan. Dimana *n* merupakan faktor skala. Jika faktor skala yang diperkecil adalah faktor skala horizontal, maka lebar citra akan diperkecil *n* kali. Sedangkan jika yang diperkecil adalah faktor skala vertikal, maka tinggi citra tujuan diperpendek *n* kali.

Dalam bentuk *pseudo-code* penerapan metode replikasi ini adalah sebagai berikut:

```
DO DestRow = 0 TO DestHeight
    SRowAddr = DestRow / ScaleV
    SRowNum = bagian desimal dari SRowAddr
    DO DestCol = 0 TO DestWidth
            SColAddr = DestCol / ScaleH
            SColNum = bagian desimal dari SColNum
            OutImage[DestCol,DestRow] = InImage[SColNum,SRowNum]
    ENDDO
ENDDO
```

Proses memperkecil citra dengan metode interpolas linier, merupakan kebalikan dari metode interpolasi linier yang diterapkan untuk proses memperbesar citra. Proses ini mirip dengan mengambil nilai rata-rata beberapa piksel yang berada di sekitar piksel yang nilainya dicari, dan hasilnya diletakkan pada piksel yang nilainya dicari pada citra tujuan. Dalam bentuk *pseudo-code* penerapan metode replikasi ini adalah sebagai berikut:

```
DO DestRow = 0 TO DestHeight
    SRowAddr = DestRow / ScaleV
    SRowNum = bagian desimal dari SRowAddr
    RowDelta = SRowAddr - SRowNum
    DO DestCol = 0 TO DestWidth
            SColAddr = DestCol / ScaleH
            SColNum = bagian desimal dari SColNum
            ColDelta = SColAddr - SColNum
            PxA = InImage[SColNum,SRowNum];
            PxB = InImage[SColNum+1,SRowNum];
            PxC = InImage[SColNum,SRowNum+1];
            PxD = InImage[SColNum+1,SRowNum+1];
            IAB = ColDelta*((double)PxB - PxA) + PxA;
            ICD = ColDelta*((double)PxD - PxC) + PxC;
            OutImage[DestCol,DestRow] = IAB + (ICD - IAB) * RowDelta
    ENDDO
ENDDO
```