ARMY RESEARCH LABORATORY

Reading, Writing, and Modifying BMP Files Using C++

by Robert J. Yager

ARL-TN-559 **August 2013**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005

ARL-TN-559 **August 2013**

Reading, Writing, and Modifying BMP Files Using C++

Robert J. Yager
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
August 2013	Final	05/2013

4. TITLE AND SUBTITLE
Reading, Writing, and Modifying BMP Files Using C++

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Robert J. Yager

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-WML-A
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-559

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
BMP files can be used to store images as arrays of pixels. This report presents a set of functions, written in C++, that can be used to read, write, and modify 24-bit, uncompressed BMP files.

15. SUBJECT TERMS
BMP, file, read, write, C++, pixel

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Robert J. Yager
Unclassified	Unclassified	Unclassified	UU	22	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6689

Contents

List of Figures v

List of Tables v

Acknowledgments vi

1. Introduction 1

2. BMP Headers 1

3. BMP Pixel Data 2

4. Creating New Images – the NewImage() Function 2

4.1 NewImage() Code 2

4.2 NewImage() Parameters 2

4.3 NewImage() Return Value 3

5. Reading BMP Files – the ReadBmpFile() Function 3

5.1 ReadBmpFile() Code 3

5.2 ReadBmpFile() Parameters 3

5.3 ReadBmpFile() Return Value 3

6. Writing BMP Files – the WriteBmpFile() Function 4

6.1 WriteBmpFile() Code 4

6.2 WriteBmpFile() Parameters 4

6.3 WriteBmpFile() Return Value 4

7. Accessing Individual Pixels – the GetPixel() Function 4

7.1 GetPixel() Code 4

7.2 GetPixel() Parameters 5

7.3 GetPixel() Return Value 5

8. Image Dimensions – GetWidth(), GetHeight(), and GetSize() Functions 5

8.1 GetWidth() Code 5
8.2 GetWidth() Parameters 5
8.3 GetWidth() Return Value 5
8.4 GetHeight() Code 5
8.5 GetHeight() Parameters 5
8.6 GetHeight() Return Value 6
8.7 GetSize() Code 6
8.8 GetSize() Parameters 6
8.9 GetSize() Return Value 6

9. Copying Images – The CopyImage() Function 6

9.1 CopyImage() Code 7
9.2 CopyImage() Parameters 7
9.3 CopyImage() Return Value 7

10. Example – Creating an Image of the Mandelbrot Set 7

11. Example – Adjusting Contrast Level 8

12. Summary 10

Distribution List 13

List of Figures

Figure 1. Source and destination images. 6
Figure 2. The BMP file mandelbrot.bmp, created using example code. 8
Figure 3. Incrementally increasing levels of contrast (top-left is the original image). 10

List of Tables

Table 1. BMP header values. 1

Acknowledgments

The author would like to thank Dr. Benjamin Breech of the U.S. Army Research Laboratory's Weapons and Materials Research Directorate. Dr. Breech provided technical and editorial recommendations that improved the quality of this report.

1. Introduction

BMP files can be used to store images as arrays of pixels. This report presents a set of functions, written in C++, that can be used to read, write, and modify 24-bit, uncompressed BMP files. A summary sheet is provided at the end of this report. It presents the yBmp namespace, which contains the eight functions that are described in this report.

2. BMP Headers

BMP files begin with header information that stores, among other things, the size of the image contained in the BMP file. BMP headers store numeric values in little-endian order. Bitwise-shift operators can be used to convert to and from little-endian format. Table 1 lists the contents of the BMP headers that are associated with the BMP files that are described in this report.

Table 1. BMP header values.

Offset	Size	Decimal Value (0 to 255)				Description
0	2	-	66	77	-	"BM"
2	4	N	N>>8	N>>16	N>>24	size, N, of a BMP file
6	2	-	0	0	-	application specific - not used by the yBmp namespace
8	2	-	0	0	-	application specific - not used by the yBmp namespace
10	4	54	0	0	0	number of bytes from beginning of file to beginning of pixel data
14	4	40	0	0	0	number of bytes from this point to the beginning of pixel data
18	4	w	w>>8	0	0	width, w, of image (in pixels, limited to 2^{16}-1=65,535)
22	4	h	h>>8	0	0	Height, h, of image (in pixels, limited to 2^{16}-1=65,535). Height values can be positive or negative. Negative values imply that the image is mirrored, top to bottom
26	2	-	1	0	-	number of color planes used
28	2	-	24	0	-	number of bits per pixel
30	4	0	0	0	0	no compression
34	4	n	n>>8	n>>16	n>>24	size, n, of image (n=h*(3*w+w%4)=N-54)
38	4	19	11	0	0	horizontal resolution of image (2835 pixels/meter)
42	4	19	11	0	0	vertical resolution of image (2835 pixels/meter)
46	4	0	0	0	0	number of colors in the palette
50	4	0	0	0	0	all colors are important

3. BMP Pixel Data

Three 8-bit integers, each with values 0 to 255, are used to define the color of each pixel in an image. The integers describe the intensity of blue, green, and red contributions to a pixel's color. For example, {0,0,255} is red, while {255,0,0} is blue. Pixels are ordered by column, then by row, beginning with either the bottom-left corner of the image for positive height values or the top-left corner of the image for negative height values. The end of each row of pixel data must be padded to ensure that the number of 8-bit integers stored in each row is evenly divisible by 4.

4. Creating New Images – the NewImage() Function

New images can be created using the NewImage() function.

Note that the NewImage() function uses the "new" command to allocate memory for the image that is pointed to by the return value. Thus, to avoid memory leaks, each use of the NewImage() function should be accompanied by a use of the "delete[]" operator.

4.1 NewImage() Code

```
inline unsigned char* NewImage(//<=========================CREATES A NEW IMAGE
    unsigned int w,int h,//<---------------WIDTH & HEIGHT (IN PIXELS) OF IMAGE
    unsigned char f=0){//---------------------FILL VALUE (0=>BLACK,255=>WHITE)
  unsigned int n=abs(h)*(3*w+w%4),N=n+54;//.............image size, & file size
  unsigned char H[54]={66,77,N,N>>8,N>>16,N>>24,0,0,0,0,54,0,0,0,40,0,0,0,w,
                       w>>8,0,0,h,h>>8,h>>16,h>>24,1,0,24,0,0,0,0,0,n,n>>8,
                       n>>16,n>>24,19,11,0,0,19,11,0,0,0,0,0,0,0,0,0,0};
  unsigned char* I=new unsigned char[N];/*<-*/memcpy(I,H,54),memset(I+54,f,n);
  return I;
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

4.2 NewImage() Parameters

w — **w** specifies the width of an image, in pixels (1 to 2^{16}–1).

h — **h** specifies the height of an image, in pixels (1 to 2^{16}–1). Values for **h** can be positive or negative. Positive height values imply that pixels are ordered beginning with the bottom-left corner of an image. Negative height values imply that pixels are ordered beginning with the top-left corner of an image.

f — **f** specifies a numeric value (0 to 255) that will be used to fill the pixel values. Use 0 to fill the image with black pixels, use 255 to fill the image with white pixels. The default value is 0.

4.3 NewImage() Return Value

The NewImage() function returns a pointer to a newly created image.

---

5. Reading BMP Files – the ReadBmpFile() Function

The ReadBmpFile() function can be used to create an image from a BMP file.

Note that the ReadBmpFile() function uses the "new" command to allocate memory for the image that is pointed to by the return value. Thus, to avoid memory leaks, each use of the ReadBmpFile() function should be accompanied by a use of the "delete[]" operator.

5.1 ReadBmpFile() Code

```
inline unsigned char* ReadBmpFile(//<======CREATES A NEW IMAGE FROM A BMP FILE
    const char* filename){//<-------------------------THE NAME OF THE BMP FILE
  FILE* f=fopen(filename,"rb");//..............file must be opened as a binary
  if(!f)printf("\nCan't open \"%s\".\n",filename),exit(1);//.is the file open?
  unsigned char H[6];/*<-*/fread(H,1,6,f),rewind(f);//read beginning of header
  unsigned int N=H[2]+(H[3]<<8)+(H[4]<<16)+(H[5]<<24);//.........get file size
  unsigned char* I=new unsigned char[N];/*<-*/fread(I,1,N,f),fclose(f);
  return I;
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

5.2 ReadBmpFile() Parameters

filename **filename** specifies the name of a 24-bit, uncompressed BMP file.

5.3 ReadBmpFile() Return Value

The ReadBmpFile() function returns a pointer to an image that contains the header and pixel data from the BMP file that was specified by the **filename** parameter.

If the file that is specified by **filename** cannot be opened, the ReadBmpFile() function calls the exit() function with status code of 1. Inability to open a file is typically the result of an incorrectly specified filename or path.

6. Writing BMP Files – the WriteBmpFile() Function

The WriteBmpFile() function can be used to write an image to a BMP file.

6.1 WriteBmpFile() Code

```
inline int WriteBmpFile(//<===========WRITES A BMP FILE (24-BIT, UNCOMPRESSED)
    const char* filename,//<-------------------------THE NAME OF THE BMP FILE
    const unsigned char* I){//<---THE IMAGE THAT WILL BE WRITTEN TO A BMP FILE
  FILE* f=fopen(filename,"wb");//.............file must be opened as a binary
  if(!f)printf("\nCan't open \"%s\".\n",filename),exit(1);//.is the file open?
  unsigned int N=*(I+2)+(*(I+3)<<8)+(*(I+4)<<16)+(*(I+5)<<24);//.....file size
  unsigned int n=fwrite(I,1,N,f);/*&*/fclose(f);//......write the file & close
  return n;//..............number of characters that were written to the file
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

6.2 WriteBmpFile() Parameters

filename **filename** specifies the name of a BMP file.

I **I** specifies a pointer to an image.

6.3 WriteBmpFile() Return Value

The WriteBmpFile() function returns the number of characters that were written to the output file.

If the file that is specified by **filename** cannot be opened, the WriteBmpFile() function calls the exit() function with status code of 1. Inability to open a file is often the result of an incorrectly specified filename or path. However, it can also be the result of the file being marked as read only, as may be the case if the file is open in another program.

7. Accessing Individual Pixels – the GetPixel() Function

The GetPixel() function returns a pointer to a pixel in an image. The pointer can be used to read or modify the pixel that it points to.

7.1 GetPixel() Code

```
inline unsigned char* GetPixel(//<================RETURNS A POINTER TO A PIXEL
    unsigned char* I,//<-------------A POINTER TO THE BEGINNING OF A BMP IMAGE
    unsigned int i,unsigned int j){//<-----------------# OF PIXELS RIGHT & UP
  return I+54+j*((*(I+18)+(*(I+19)<<8))*3+*(I+18)%4)+3*i;
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

7.2 GetPixel() Parameters

I **I** specifies a pointer to an image.

i **i** specifies a column in an image. Column numbers increase moving left to right. Allowable values for **i** are $0 \le \mathbf{i} < \mathbf{w}$, where **w** is the width of an image.

j **j** specifies a row in an image. Row numbers increase moving bottom to top for images with positive heights, and top to bottom for images with negative heights. Allowable values for **j** are $0 \le \mathbf{j} < |\mathbf{h}|$, where **h** is the height of an image.

7.3 GetPixel() Return Value

The GetPixel() function returns a pointer to a pixel.

---

8. Image Dimensions – GetWidth(), GetHeight(), and GetSize() Functions

---

The GetWidth() function can be used to determine the width (in pixels) of an image, the GetHeight() function can be used to determine the height (in pixels) of an image, and the GetSize() function can be used to determine the size (in bytes) of an image.

8.1 GetWidth() Code

```
inline unsigned int GetWidth(//<===============RETRIEVES THE WIDTH OF AN IMAGE
    const unsigned char* I){//<------A POINTER TO THE BEGINNING OF A BMP IMAGE
  return *(I+18)+(*(I+19)<<8);
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

8.2 GetWidth() Parameters

I **I** specifies a pointer to an image.

8.3 GetWidth() Return Value

The GetWidth() function returns the width of the input-specified image, in pixels.

8.4 GetHeight() Code

```
inline int GetHeight(//<======================RETRIEVES THE HEIGHT OF AN IMAGE
    const unsigned char* I){//<------A POINTER TO THE BEGINNING OF A BMP IMAGE
  return *(I+22)+(*(I+23)<<8)-(*(I+25)==255?65536:0);
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

8.5 GetHeight() Parameters

I **I** specifies a pointer to an image.

8.6 GetHeight() Return Value

The GetHeight() function returns the height of the input-specified image, in pixels. Values can be positive or negative. Positive height values imply that pixels are ordered beginning with the bottom-left corner of an image. Negative height values imply that pixels are ordered beginning with the top-left corner of an image.

8.7 GetSize() Code

```
inline unsigned int GetSize(//<====RETRIEVES THE SIZE OF AN IMAGE'S PIXEL DATA
    const unsigned char* I){//<------A POINTER TO THE BEGINNING OF A BMP IMAGE
  unsigned int w=GetWidth(I);
  return abs(GetHeight(I))*(3*w+w%4);
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

8.8 GetSize() Parameters

I **I** specifies a pointer to an image.

8.9 GetSize() Return Value

The GetSize() function returns the size of the pixel portion of an image, in bytes.

---

9. Copying Images – The CopyImage() Function

The CopyImage() function can be used to copy pixel data from one image to another. The two images need not be the same size and can be offset from each other, as shown in figure 1.

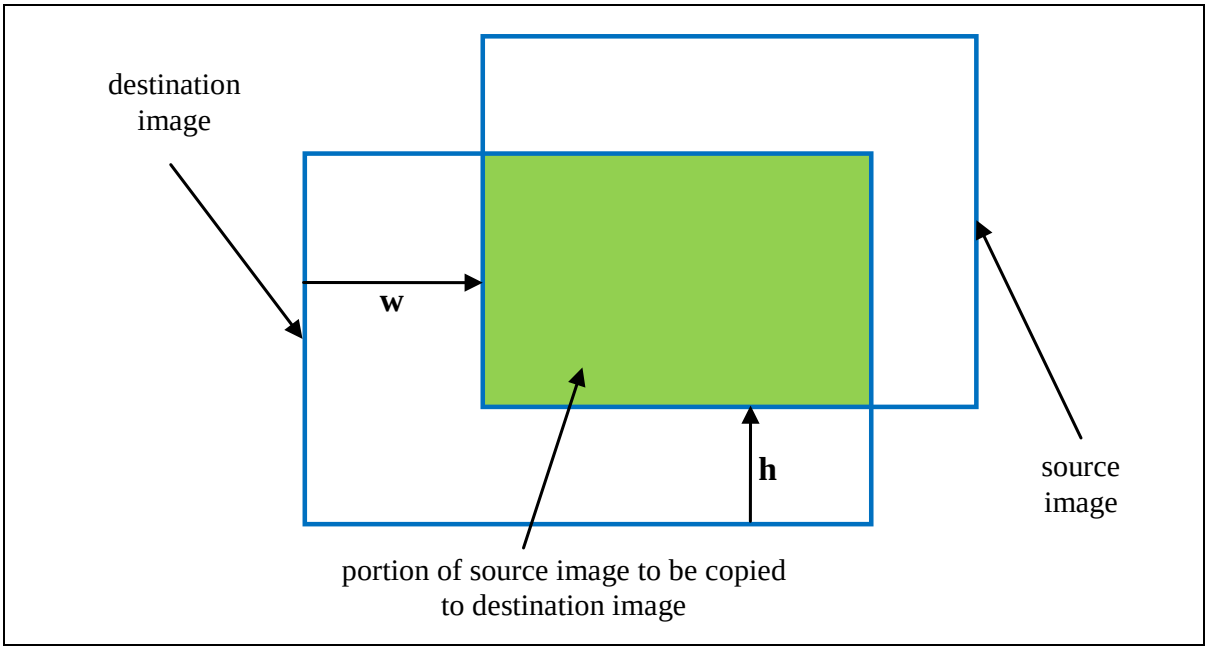


Figure 1. Source and destination images.

9.1 CopyImage() Code

```
inline unsigned char* CopyImage(//<======COPY IMAGE FROM SOURCE TO DESTINATION
    unsigned char* S,//<-----------------------------------------SOURCE IMAGE
    unsigned char* D,//<----------------------------------DESTINATION IMAGE
    int w, int h){//<--------------------------------WIDTH AND HEIGHT OFFSETS
  int wd=GetWidth(D),W=GetWidth(S)+w,hd=abs(GetHeight(D)),
    H=abs(GetHeight(S))+h,i=w>0?w:0,I=wd<W?wd:W;
  if(i<I)for(unsigned int j=h>0?h:0,J=hd<H?hd:H;j<J;++j)
    memcpy(GetPixel(D,i,j),GetPixel(S,i-w,j-h),3*(I-i));
  return D;
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

9.2 CopyImage() Parameters

S **S** specifies a pointer to a source image.

D **D** specifies a pointer to a destination image.

w **w** specifies a width offset (see figure 1).

h **h** specifies a height offset (see figure 1).

9.3 CopyImage() Return Value

The CopyImage() function returns a pointer to the destination image (**D**).

---

10. Example – Creating an Image of the Mandelbrot Set

The following example uses functions from the yBmp namespace to create an image of the Mandelbrot set. The example begins by creating an image that is 1400 pixels wide by 800 pixels high. The example then uses the escape-time algorithm[1] to generate an integer value (k) for each pixel location. Next, values of k are mapped to intensities of blue and the colors of individual pixels are set using the GetPixel() function. Finally, the WriteBmpFile() function is used to create a BMP file. The result is displayed in figure 2.

[1] Barnsley, M. F. *Fractals Everywhere, New Edition*; Dover Press: United States, 2012.

```
#include <cmath>//.....................................................hypot()
#include "y_bmp.h"
int main(){
  int w=1400,h=800;
  unsigned char* I=yBmp::NewImage(w,h);
  for(int i=0,j,k,N=1000;i<w;++i)for(j=0;j<h;++j){
    double x=-2.5+i*3.5/(w-1) , y=-1+j*2./(h-1) , re=0 , im=0 , t;
    for(k=0;k<N&&hypot(re,im)<4;k+=50) t=re*re-im*im+x , im=2*re*im+y , re=t;
    *yBmp::GetPixel(I,i,j)=k<N?k*255/N:0;}
  yBmp::WriteBmpFile("mandelbrot.bmp",I);
  delete I;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

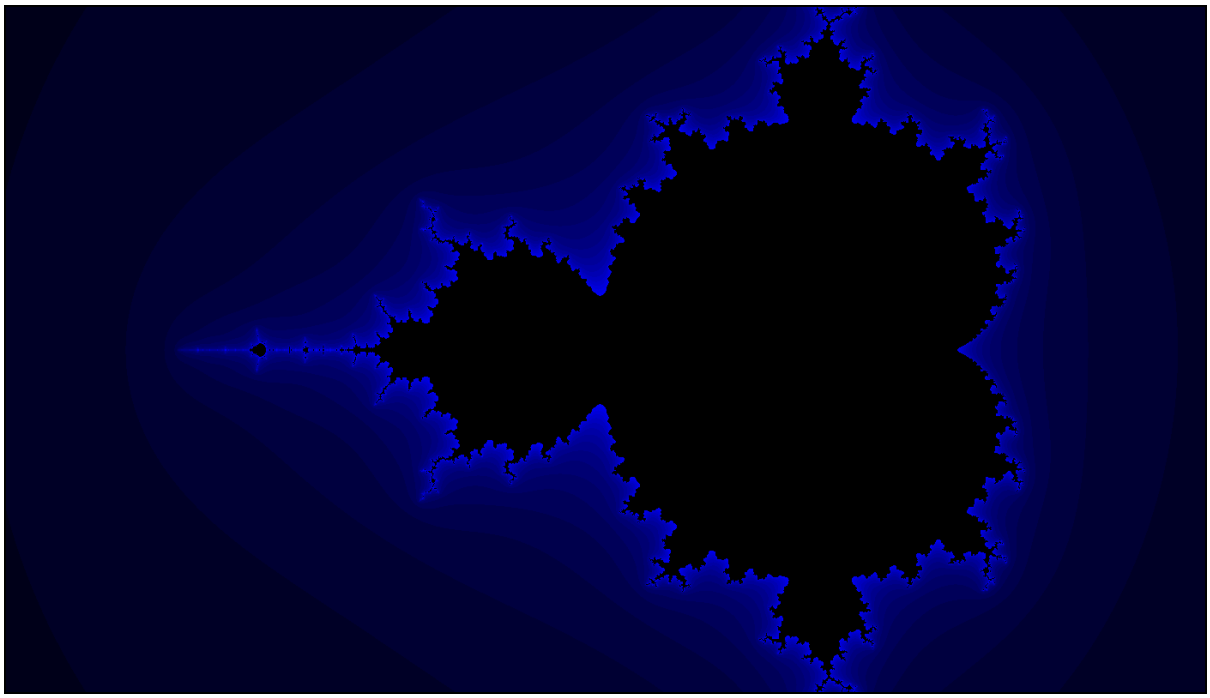


Figure 2. The BMP file mandelbrot.bmp, created using example code.

11. Example – Adjusting Contrast Level

The following example uses functions from the yBmp namespace, as well as the included AdjustContrast() function, to change the contrast of a photograph. The example begins by using the ReadBmpFile() function to convert a BMP file to an image. Next the NewImage() function is used to create an image that is twice as wide and twice as high as the original image. The original image is then copied into the top-left quadrant of the new image using the CopyImage() function. The AdjustContrast() function is then used to modify the original image. Modified images are then copied to the remaining three quadrants of the new image, proceeding clockwise. Finally,

the WriteBmpFile() function is used to convert the new image to a BMP file. Figure 3 displays the contents of the newly created file.

```
#include <algorithm>//.....................................................max()
#include "y_bmp.h"
void AdjustContrast(//<==================ADJUSTS THE CONTRAST RATIO OF AN IMAGE
    unsigned char* I,//<-------------------------------A POINTER TO A BMP IMAGE
    double contrast){//<----------------------CONTRAST SETTING (0 => No Change)
  for(unsigned int i=0,n=GetSize(I);i<n;++i)
    I[i+54]=unsigned char(std::max((1+contrast)*I[i+54]-contrast*255,0.));
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~13AUG2013~~~~~~
int main(){
  unsigned char* I=yBmp::ReadBmpFile("image.bmp");
  int w=yBmp::GetWidth(I),h=yBmp::GetHeight(I);
  unsigned char* I2=yBmp::NewImage(2*w,2*h);
  yBmp::CopyImage(I,I2,0,h);//.....................................top-left image
  AdjustContrast(I,.25),yBmp::CopyImage(I,I2,w,h);//.............top-right image
  AdjustContrast(I,.25),yBmp::CopyImage(I,I2,w,0);//..........bottom-right image
  AdjustContrast(I,.25);yBmp::CopyImage(I,I2,0,0);//...........bottom-left image
  yBmp::WriteBmpFile("composite_contrast.bmp",I2);
  delete I, delete I2;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

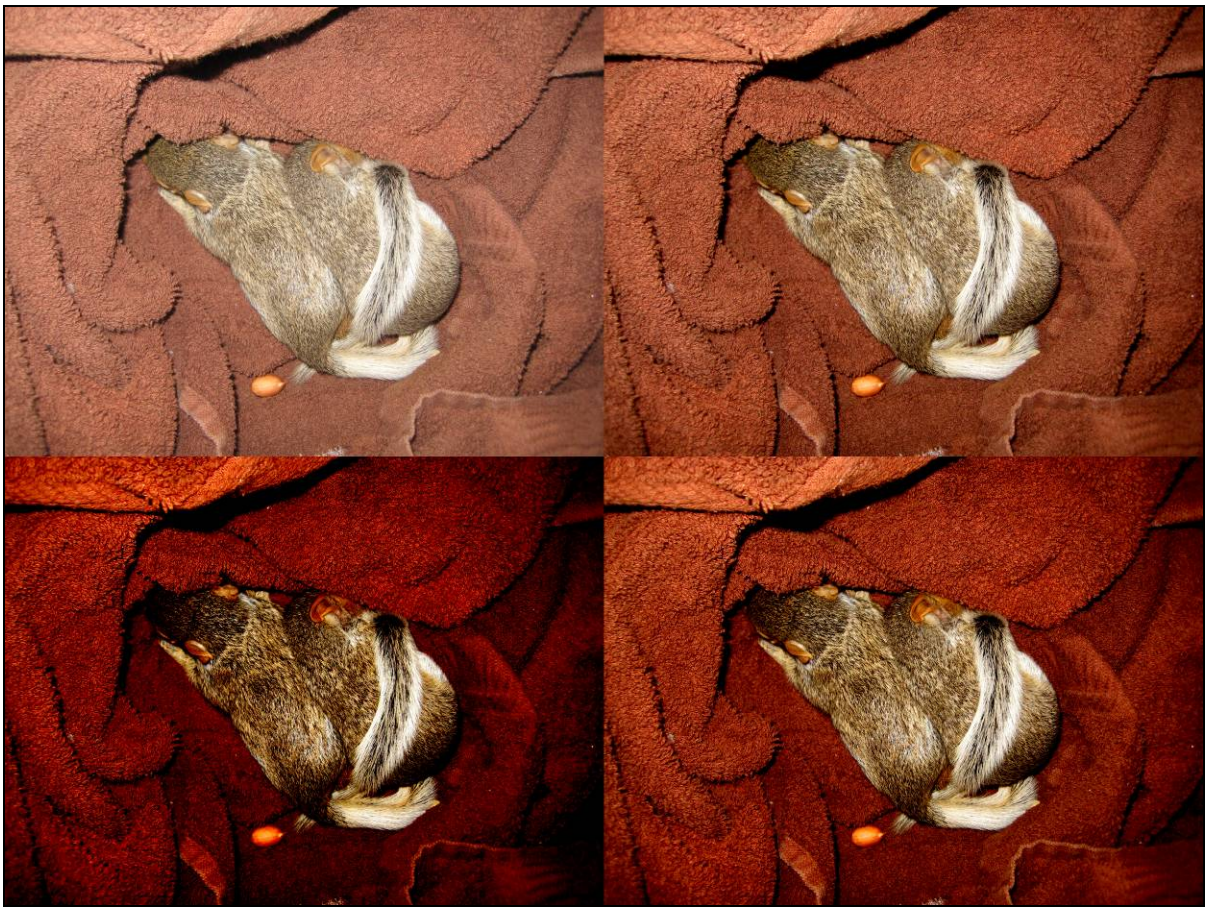


Figure 3. Incrementally increasing levels of contrast (top-left is the original image).

12. Summary

A summary sheet is provided at the end of this report. It presents the yBmp namespace, which contains the eight functions that are described in this report.

yBmp Summary

y_bmp.h

```
#ifndef Y_BMP_GUARD
#define Y_BMP_GUARD
#include <cstdio>//.....FILE,fopen(),fclose(),fread(),fwrite(),rewind(),printf()
#include <cstdlib>//............................................abs(),exit()
#include <cstring>//.......................................memcpy(),memset()
namespace yBmp{
  //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  inline unsigned char* NewImage(//<========================CREATES A NEW IMAGE
      unsigned int w,int h,//<---------------WIDTH & HEIGHT (IN PIXELS) OF IMAGE
      unsigned char f=0){//---------------------FILL VALUE (0=>BLACK,255=>WHITE)
    unsigned int n=abs(h)*(3*w+w%4),N=n+54;//...........image size, & file size
    unsigned char H[54]={66,77,N,N>>8,N>>16,N>>24,0,0,0,0,54,0,0,0,40,0,0,0,w,
                         w>>8,0,0,h,h>>8,h>>16,h>>24,1,0,24,0,0,0,0,0,n,n>>8,
                         n>>16,n>>24,19,11,0,0,19,11,0,0,0,0,0,0,0,0,0,0};
    unsigned char* I=new unsigned char[N];/*<-*/memcpy(I,H,54),memset(I+54,f,n);
    return I;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
  inline unsigned char* ReadBmpFile(//<======CREATES A NEW IMAGE FROM A BMP FILE
      const char* filename){//<-------------------------THE NAME OF THE BMP FILE
    FILE* f=fopen(filename,"rb");//.............file must be opened as a binary
    if(!f)printf("\nCan't open \"%s\".\n",filename),exit(1);//.is the file open?
    unsigned char H[6];/*<-*/fread(H,1,6,f),rewind(f);//read beginning of header
    unsigned int N=H[2]+(H[3]<<8)+(H[4]<<16)+(H[5]<<24);//.........get file size
    unsigned char* I=new unsigned char[N];/*<-*/fread(I,1,N,f),fclose(f);
    return I;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
  inline int WriteBmpFile(//<==========WRITES A BMP FILE (24-BIT, UNCOMPRESSED)
      const char* filename,//<-------------------------THE NAME OF THE BMP FILE
      const unsigned char* I){//<---THE IMAGE THAT WILL BE WRITTEN TO A BMP FILE
    FILE* f=fopen(filename,"wb");//.............file must be opened as a binary
    if(!f)printf("\nCan't open \"%s\".\n",filename),exit(1);//.is the file open?
    unsigned int N=*(I+2)+(*(I+3)<<8)+(*(I+4)<<16)+(*(I+5)<<24);//.....file size
    unsigned int n=fwrite(I,1,N,f);/*&*/fclose(f);//......write the file & close
    return n;//..............number of characters that were written to the file
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
  inline unsigned char* GetPixel(//<================RETURNS A POINTER TO A PIXEL
      unsigned char* I,//<-------------A POINTER TO THE BEGINNING OF A BMP IMAGE
      unsigned int i,unsigned int j){//<------------------# OF PIXELS RIGHT & UP
    return I+54+j*((*(I+18)+(*(I+19)<<8))*3+*(I+18)%4)+3*i;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
  inline unsigned int GetWidth(//<===============RETRIEVES THE WIDTH OF AN IMAGE
      const unsigned char* I){//<------A POINTER TO THE BEGINNING OF A BMP IMAGE
    return *(I+18)+(*(I+19)<<8);
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
  inline int GetHeight(//<=====================RETRIEVES THE HEIGHT OF AN IMAGE
      const unsigned char* I){//<------A POINTER TO THE BEGINNING OF A BMP IMAGE
    return *(I+22)+(*(I+23)<<8)-(*(I+25)==255?65536:0);
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
  inline unsigned int GetSize(//<====RETRIEVES THE SIZE OF AN IMAGE'S PIXEL DATA
      const unsigned char* I){//<------A POINTER TO THE BEGINNING OF A BMP IMAGE
    unsigned int w=GetWidth(I);
    return abs(GetHeight(I))*(3*w+w%4);
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
  inline unsigned char* CopyImage(//<======COPY IMAGE FROM SOURCE TO DESTINATION
      unsigned char* S,//<---------------------------------------------SOURCE IMAGE
      unsigned char* D,//<----------------------------------------DESTINATION IMAGE
      int w, int h){//<--------------------------------WIDTH AND HEIGHT OFFSETS
    int wd=GetWidth(D),W=GetWidth(S)+w,hd=abs(GetHeight(D)),
      H=abs(GetHeight(S))+h,i=w>0?w:0,I=wd<W?wd:W;
    if(i<I)for(unsigned int j=h>0?h:0,J=hd<H?hd:H;j<J;++j)
      memcpy(GetPixel(D,i,j),GetPixel(S,i-w,j-h),3*(I-i));
    return D;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
}//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#endif
```

Example 1 Output (rotated 90° clockwise)

Example 2 Output (rotated 90° clockwise)

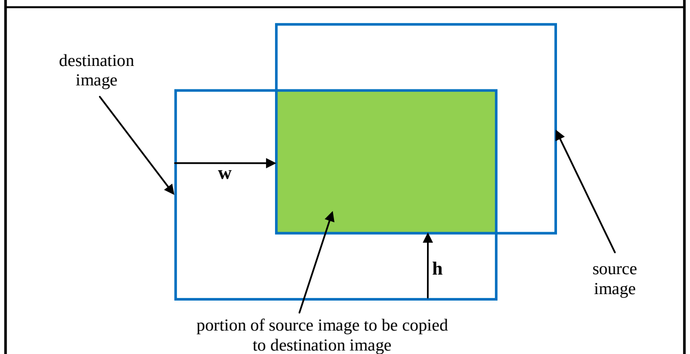
Example 1

```
#include <cmath>//.....................................................hypot()
#include "y_bmp.h"
int main(){
  int w=1400,h=800;
  unsigned char* I=yBmp::NewImage(w,h);
  for(int i=0,j,k,N=1000;i<w;++i)for(j=0;j<h;++j){
    double x=-2.5+i*3.5/(w-1) , y=-1+j*2./(h-1) , re=0 , im=0 , t;
    for(k=0;k<N&&hypot(re,im)<4;k+=50) t=re*re-im*im+x , im=2*re*im+y , re=t;
    *yBmp::GetPixel(I,i,j)=k<N?k*255/N:0;}
  yBmp::WriteBmpFile("mandelbrot.bmp",I);
  delete I;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

Example 2

```
#include <algorithm>//....................................................max()
#include "y_bmp.h"
void AdjustContrast(//<===================ADJUSTS THE CONTRAST RATIO OF AN IMAGE
    unsigned char* I,//<-------------------------------A POINTER TO A BMP IMAGE
    double contrast){//<----------------------CONTRAST SETTING (0 => No Change)
  for(unsigned int i=0,n=GetSize(I);i<n;++i)
    I[i+54]=unsigned char(std::max((1+contrast)*I[i+54]-contrast*255,0.));
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~13AUG2013~~~~~~
int main(){
  unsigned char* I=yBmp::ReadBmpFile("image.bmp");
  int w=yBmp::GetWidth(I),h=yBmp::GetHeight(I);
  unsigned char* I2=yBmp::NewImage(2*w,2*h);
  yBmp::CopyImage(I,I2,0,h);//...................................top-left image
  AdjustContrast(I,.25),yBmp::CopyImage(I,I2,w,h);//.............top-right image
  AdjustContrast(I,.25),yBmp::CopyImage(I,I2,w,0);//..........bottom-right image
  AdjustContrast(I,.25);yBmp::CopyImage(I,I2,0,0);//...........bottom-left image
  yBmp::WriteBmpFile("composite_contrast.bmp",I2);
  delete I, delete I2;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~14AUG2013~~~~~~
```

CopyImage() Parameters



INTENTIONALLY LEFT BLANK.

NO. OF
COPIES ORGANIZATION

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

2 (PDFS) DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT

1 (PDF) GOVT PRINTG OFC A MALHOTRA

ABERDEEN PROVING GROUND

1 (PDF) DIR USARL RDRL WML A R YAGER

INTENTIONALLY LEFT BLANK.