位图混合报告

（一）设计目标

编写一个程序，可以在命令行输入参数，完成参数所指定的两幅真彩色位图图像的混合运算，并存储混合后的结果到新文件，命令行参数如下:

blend x.bmp 80% y.bmp result.bmp

第 1 个参数( blend )为可执行程序名称；

第 2 个参数 x.bmp 为原始图像文件名；

第 3 个参数为混合比例（百分比），如 80%表示 x 图像亮度数据 \* 80% + y 图像亮度数据\* （1 - 80%） = 结果图像数据）；

第 4 个参数 y.bmp 第二幅图像文件名；

第 5 个参数 result.bmp 为结果图像文件名

（二）功能要求：

1.打开两个输入图像文件（argv[1]和argv[3]），读取它们的位图文件头和位图信息头。

2.从命令行参数中获取混合的百分比，并将其转换为小数形式。

3.打开输出图像文件（argv[4]），准备写入数据。

4.检查输入图像是否为BMP格式、位深度是否为24位真彩色、尺寸是否相同。

5.如果检查通过，则按照从下到上、从左到右的顺序逐像素读取两个6.输入图像的颜色数据

7.根据给定的混合百分比，计算混合后的像素值，并将结果存储在result数组中。

8.将位图文件头和位图信息头写入输出图像文件。

9.按照从下到上、从左到右的顺序逐像素将混合后的像素数据写入输出图像文件。

10.打印"执行成功"消息表示程序成功执行。

11.关闭输入图像文件和输出图像文件的文件指针。

（三）设计思路

程序使用全局数组 xx、yy 和 result 来存储图像的RGB颜色数据。这些数组分别用于存储两个输入图像的颜色像素值以及混合后的图像像素值。

1.定义了两个结构体：biHead 表示位图信息头，bfHead 表示位图文件头。

2.main 函数接受命令行参数：argc 表示参数个数，argv 是一个包含参数字符串的数组。

3.使用 fopen 打开由 argv[1] 和 argv[3] 指定的两个输入图像文件。文件以读写模式（r+b）打开。

4.从命令行参数 argv[2] 中提取混合百分比，并将其计算为十进制值。

程序使用 fopen 以写入和读取模式（w+b）打开由 argv[4] 指定的输文件。

5.使用 fread 读取两个输入图像的位图文件头和信息头。

程序检输入文件是否为正确类型（BM），是否具有24位的位深度（真彩色），以及是否具有相同的宽度和高度。

6.如果输入图像通过了检查，程序从文件中读取每个图像的颜色像素数据。像素数据从底部向顶部逐列读取。

7.在读取像素数据后，程序执行混合操作。它将第一个图像每个像素的颜色值乘以混合百分比，并将其与第二个图像相应像素的颜色值乘以 (1 - 混合百分比) 相加。

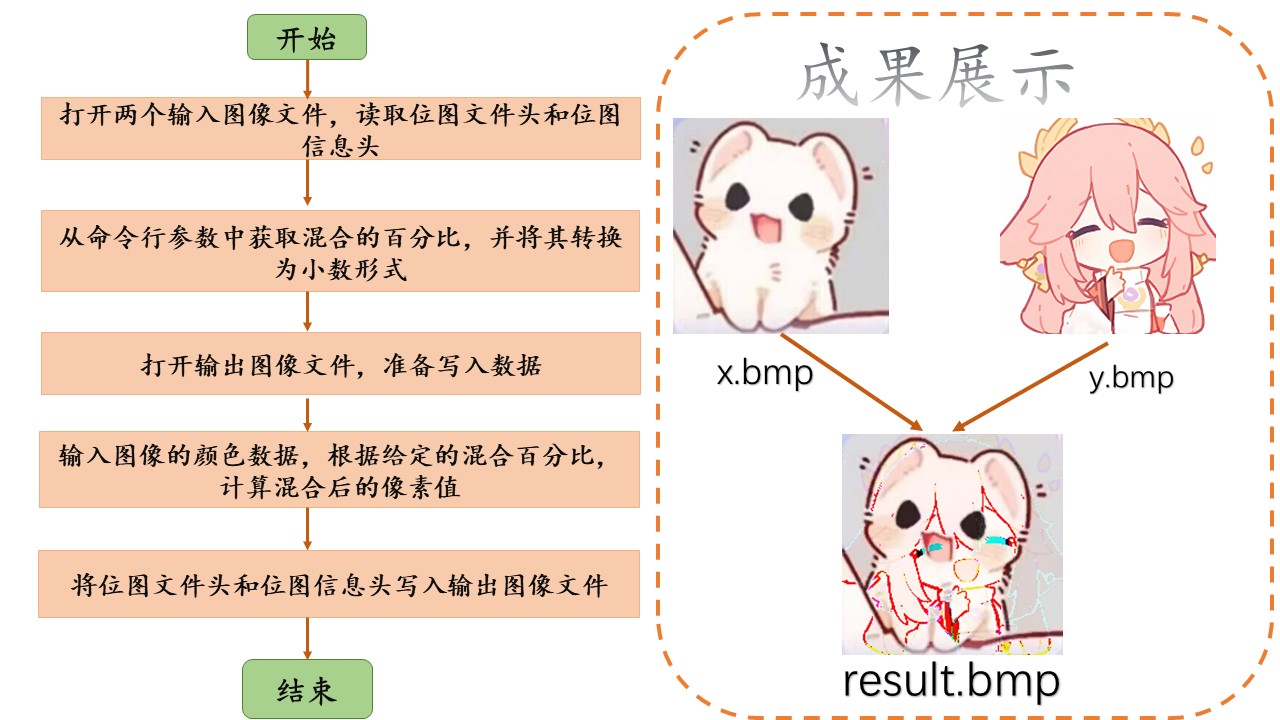
8.混合后的像素数据存储在 result 数组中。

9.程序使用 fwrite 将位图文件头和信息头写入输出文件。

10.最后，程序逐列从底部向顶部将混合后的像素数据写入输出文件。

11.如果混合过程成功，程序打印成功消息。

12.使用 fclose 关闭所有打开的文件。

（四）总体设计：

（五）功能设计

接受两个输入图像文件、一个混合百分比和一个输出文件名作为命令行参数。该程序从输入文件中读取图像数据，根据指定的百分比混合图像，并将混合后的图像数据写入输出文件中。

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. *//存放图片的颜色点阵数据RGB*
5. char xx[1000][1000][3];
6. char yy[1000][1000][3];
7. char result[1000][1000][3];

包括必要的头文件：<stdio.h>, <stdlib.h>, 和<string.h>。

定义了三个全局数组来存储图像的RGB颜色数据：xx、yy和 result。这些数组用于保存两个输入图像和混合图像的颜色像素值。

1. *//位图信息头*
2. struct biHead
3. {
4. long biSize;
5. long biWidth;   *//图的宽度*
6. long biHeight;  *//图的高度*
7. short biPlanes;
8. short biBitCount;   *//像素点位深度，24则为24位真彩图*
9. long biCompression;
10. long biSizeImage;
11. long biXPelsPerMeter;
12. long biYpelsPerMeter;
13. long biClrUsed;
14. long biClrImportant;
15. };
16. *//位图文件头*
17. struct bfHead
18. {
19. char bfType[2]; *//检查读入的文件是否是"BM"类型*
20. long bfSize;
21. long bfReserved;
22. long bfOffBits;
23. };

定义了两个结构：biHead代表位图信息头，bfHead代表位图文件头。

主函数接受命令行参数：argc（参数数目）和argv（参数字符串的数组）。

1. int main(int argc, char\* argv[])
2. {
3. struct bfHead bfSrc1, bfSrc2;
4. struct biHead biSrc1, biSrc2;
5. *//打开2个原始文件 供读取数据*
6. FILE \*fp1 = fopen(argv[1], "r+b");
7. FILE \*fp2 = fopen(argv[3], "r+b");

使用fopen打开由argv[1]和argv[3]指定的两个输入图像文件。这些文件是以 "读写 "模式（r+b）打开的。

1. *//混合的百分比*
2. double percent = 0;
3. int i;
4. for (i = 0; i < strlen(argv[2]) - 1; ++i)
5. percent = percent \* 10 + (argv[2][i] - '0');
6. percent /= 100;

从命令行参数argv[2]中提取混合百分比，并将其计算为十进制值。

1. *//打开目标文件 供写入数据*
2. *//这个文件可能是没有的，fopen会创建一个新的文件；如果该文件已经存在，fopen会将其覆盖掉*
3. FILE \*fp3 = fopen(argv[4], "w+b");
4. *//读取位图文件头 位图信息头*
5. fread(&bfSrc1, 14, 1, fp1);
6. fread(&bfSrc2, 14, 1, fp2);
7. fread(&biSrc1, 40, 1, fp1);
8. fread(&biSrc2, 40, 1, fp2);

以 "写和读 "模式（w+b）使用fopen打开argv[4]所指定的输出文件。

用fread读取两个输入图像的位图文件头和信息头。

1. *//检查是否为BM类型 是否为24位真彩图 照片尺寸是否相等*
2. if (bfSrc1.bfType[0]=='B'&& bfSrc1.bfType[1]=='M' && bfSrc2.bfType[0]=='B' && bfSrc2.bfType[1]=='M'
3. && biSrc1.biBitCount == 24 && biSrc2.biBitCount == 24
4. && biSrc1.biWidth == biSrc2.biWidth && biSrc1.biHeight == biSrc2.biHeight)
5. {
6. *//读取颜色点阵数据*
7. *//遍历列 从下向上*
8. int y;
9. for ( y = biSrc1.biHeight - 1; y >= 0; --y)
10. {
11. *//遍历行 从前向后*
12. int x;
13. for (x = 0; x < biSrc1.biWidth; ++x)
14. {
15. *//文件数据流中按照BGR的顺序存储*
16. int k;
17. for (k = 2; k >= 0; --k)
18. {
19. fread(&xx[x][y][k], 1, 1, fp1);
20. fread(&yy[x][y][k], 1, 1, fp2);
21. }
22. }
23. }

检查输入文件的类型是否正确（BM），比特深度是否为24（真彩色），以及宽度和高度是否相同。

1. *//混合像素到目标文件数据*
2. for (y = biSrc1.biHeight - 1; y >= 0; --y)
3. {
4. int x;
5. for (x = 0; x<biSrc1.biWidth; ++x)
6. {
7. int k;
8. for (k = 2; k >= 0; --k)
9. result[x][y][k] = xx[x][y][k] \* percent + yy[x][y][k] \* (1 - percent);
10. }
11. }

如果输入的图像通过了检查，程序就会从文件中读取每个图像的彩色像素数据。像素数据从下到上逐列读取。

读取像素数据后，程序执行混合操作，将第一幅图像中每个像素的颜色值乘以混合百分比，并将其与第二幅图像中像素的相应颜色值相乘（1-混合百分比）。

1. *//写目标文件*
2. fwrite(&bfSrc1,14,1,fp3);
3. fwrite(&biSrc1,40,1,fp3);
5. for (y = biSrc1.biHeight - 1; y >= 0; --y)
6. {
7. int x;
8. for (x = 0; x<biSrc1.biWidth; ++x)
9. {
10. int k;
11. for ( k = 2; k >= 0; --k)
12. fwrite(&result[x][y][k], 1, 1, fp3);
13. }
14. }
15. printf("成功混合！！！\n");
16. }

混合后的像素数据被存储在结果数组中。

程序使用fwrite将位图文件头和信息头写到输出文件中。

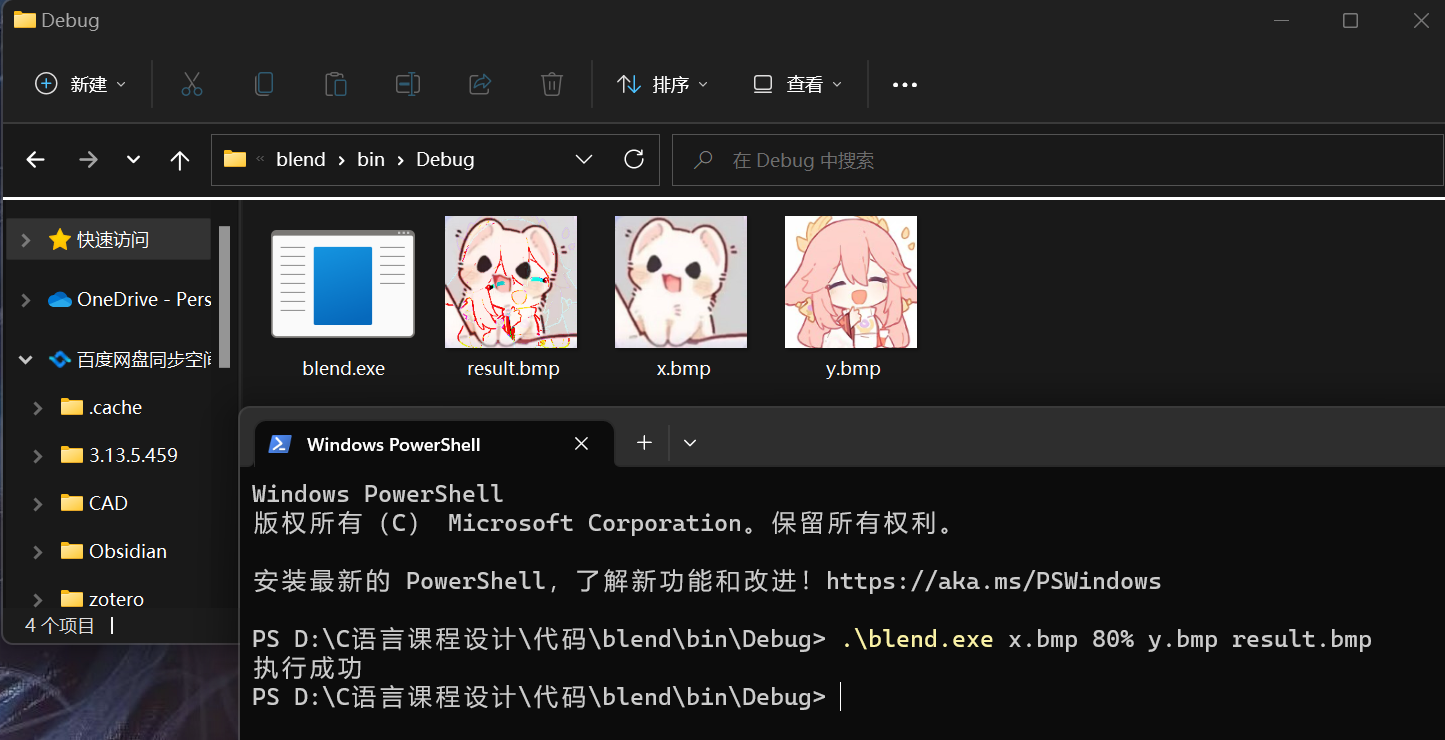
最后，程序将混合后的像素数据从下到上逐列写到输出文件中。

如果混合过程成功，程序会打印出一条成功信息

1. *//关闭文件*
2. fclose(fp1);
3. fclose(fp2);
4. fclose(fp3);
5. return 0;
6. }

程序用fclose关闭所有打开的文件。

返回0，表示成功执行。

（六）运行结果展示

（七）设计总结

本程序涉及到的操作有：

1.文件操作：通过使用标准C库中的文件操作函数（如fopen、fwrite和fclose），我学会了如何打开、读取和写入二进制文件。这对于处理图像数据非常有用，可以打开和操作不同格式的图像文件。

2.结构体的使用：在程序中使用了结构体来表示位图文件头和信息头，这使得处理和读取这些数据变得更加简洁和可读。结构体提供了一种组织和存储相关数据的方式，有助于更好地理解和维护代码。

3.图像处理：通过遍历图像的像素数据，我学会了如何处理图像的颜色数据。在这个程序中，我了解了如何按照BGR顺序读取和写入像素的RGB颜色值，并进行像素混合的计算。

4.命令行参数处理：通过使用argc和argv参数，我学会了如何接受和处理命令行输入。这使得程序可以在命令行中指定输入文件、输出文件和其他参数，增加了程序的灵活性和可扩展性。

5.错误处理：在程序中添加了一些简单的错误检查，例如检查文件类型、位深度和尺寸是否匹配。这有助于提高程序的健壮性，并避免因无效输入导致的错误。

6.数组操作：通过使用多维数组来存储图像的像素数据，我进一步加强了对数组的操作和索引的理解。这对于处理图像等二维数据非常有用。