**Predmet**: C/C++ programski jezik

Hidlo

LowLevel32

Contents

[1. Uvod 3](#_Toc448360256)

[2. Analiza 4](#_Toc448360257)

[2.1 Cilj Programa 4](#_Toc448360258)

[2.2. Elementi programa 4](#_Toc448360259)

[MAIN:C 4](#_Toc448360260)

[GETLINE.C 6](#_Toc448360261)

[IMAGEDATA:C 7](#_Toc448360262)

[WRITEIMAGEDATA.C 8](#_Toc448360263)

[GETBGR.C 9](#_Toc448360264)

[STRUCT.H 10](#_Toc448360265)

[FALPHABLEND.C 10](#_Toc448360266)

[FBINARY.C 11](#_Toc448360267)

[FBRIGHTNESS.C 12](#_Toc448360268)

[FGRAY.C 12](#_Toc448360269)

[FGREENREPLACE:C 13](#_Toc448360270)

[FMIRROR.C 14](#_Toc448360271)

[FSALTANDPAPER.C 14](#_Toc448360272)

[2.3 Izgled programa 16](#_Toc448360273)

[2.4 Prikaz obrađenih slika. 17](#_Toc448360274)

[3. Problemi programa 19](#_Toc448360275)

# 1. Uvod

Kod profesionalnih fotografa, veliku ulogu imaju fotoaparati, ali i njihov talenat zbog kojeg uspevaju da uhvate trenutke koje se ne mogu opisati rečima te nestaju zauvek i ne mogu se opet reprodukovati. Ipak, u današnje vreme, računari isto tako igraju veliku ulogu u svim oblastima, pa i u fotografiji. U večini slučajeva, fotografi ne mogu ne mogu ostvariti svoju viziju koristeći samo opremu i talenat već je neophodan i program koji će im omogućiti da svoje stavove i osobine još bolje iznesu.

Hidlo je program koji im baš to omogućava. Efekat stakla, osvetljenje, binarni efekat, transparentnost kao i crno bela slika. Svi ovi efekti omogućavaju bolji dozivljaj fotografije jer kao što je čuveni, fotoreporter rekao " Kada slikamo ljude u boji, mi slikamo njihovu odeću. A kada slikamo ljude u crno beloj boji, mi slikamo njihovu dušu. ", Ted Grant.

# 2. Analiza

## 2.1 Cilj Programa

Hidlo je program koji omogućava ne samo profesionalnim fotografima već i ostalima sirok spektar filtera radi boljeg doživljaja fotografije.

## 2.2. Elementi programa

Program se sastoji od 7 .c fajlova za filter kao i njhovi .h fajlovi. Pored toga se nalaze jos 4 .c fajlova kao i njihovi .h fajlovi koji uglavnom služe za ispis i upis podataka iz .bmp fajla. Postoji jos jedan .h fajl za strukturu podataka .bmp fajl formata i najvažniji main.c fajl.

### MAIN:C

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <math.h>
5. #include "struct.h"
6. //#include "mirror.h"
8. // Include getLine File for filename input
9. #include "getLine.h"
11. // Include imageData for loading from bmp file
12. #include "imageData.h"
14. /\* ############## HERE INCLUDE 'h' FILES FOR IMAGE FILTER ############## \*/
15. #include "fMirror.h"
16. #include "fGray.h"
17. #include "fBrightness.h"
18. #include "fSaltAndPaper.h"
19. #include "fGreenReplace.h"
20. #include "getBGR.h"

23. /\* ############## END OF INCLUDE 'h' FILES FOR IMAGE FILTER ############## \*/
25. // Write image Data to the file
26. #include "writeImageData.h"

29. **int** main() {
30. // Load Filename and image data in imagedata pointer
31. // Filename input
32. **char** \*filename;
33. filename = getln();
35. **char** \*filename2;
36. filename2 = getln();
38. // Load image data in imagedata pointer
39. **int**\* imagedata;
40. imagedata = imageData(filename);
42. **int**\* imagedata2;
43. imagedata2 = imageData(filename2);
45. // ==================================== //
46. // Input all information in struct
47. HEADER header;
48. HEADER header2;
49. INFOHEADER infoHeader;
50. INFOHEADER infoHeader2;
51. BGR BGR1;
52. BGR BGRA;
53. BGR2 BGR2;
55. // Data Offset
56. header2.offset = imagedata2[10]+imagedata2[11]+imagedata2[12]+imagedata2[13];
57. header.offset = imagedata[10]+imagedata[11]+imagedata[12]+imagedata[13];
58. printf("Data Offset: %d \n", header.offset);
59. printf("Data Offset: %d \n", header2.offset);
61. // InfoHeader Information    ( 40 Bytes )
62. infoHeader.size = imagedata[14]+imagedata[15]+imagedata[16]+imagedata[17];
63. infoHeader.width = imagedata[19]\*256 + imagedata[18];
64. infoHeader.height = imagedata[24 - 1]\*256 + imagedata[23 - 1];
65. infoHeader.bits = imagedata[28]+imagedata[29];
66. infoHeader.compression = imagedata[30]+imagedata[31]+imagedata[32]+imagedata[33];
67. infoHeader.imagesize = imagedata[35]\*128;
69. //InfoHeader Information 2
70. infoHeader2.size = imagedata2[14]+imagedata2[15]+imagedata2[16]+imagedata2[17];
71. infoHeader2.width = imagedata2[19]\*256 + imagedata2[18];
72. infoHeader2.height = imagedata2[24 - 1]\*256 + imagedata2[23 - 1];
73. infoHeader2.bits = imagedata2[28]+imagedata2[29];
75. printf("\n\n-------------------------\n\n");
76. printf("Info Header\n\n");
77. printf("Header size: %d \n", infoHeader.size);
78. printf("Image width: %d \n", infoHeader.width);
79. printf("Image height: %d \n", infoHeader.height);
80. printf("%d bit image: \n", infoHeader.bits);
81. printf("Compression: %d \n", infoHeader.compression);
83. // RowSize and Pixel Array Size
84. **int** rowSize = floor((infoHeader.bits \* infoHeader.width +31)/32)\*4;
85. **int** pixelArraySize = rowSize \* abs(infoHeader.height) + header.offset;
87. // RowSize and Pixel Array Size 2
88. **int** rowSize2 = floor((infoHeader2.bits \* infoHeader2.width +31)/32)\*4;
89. **int** pixelArraySize2 = rowSize2 \* abs(infoHeader2.height) + header2.offset;
91. // BGR Color
92. BGR1.bgr = getBGR(infoHeader.height, infoHeader.width, header.offset, pixelArraySize, imagedata);
93. // BGR Color 2
94. BGRA.bgr = getBGR(infoHeader2.height, infoHeader2.width, header2.offset, pixelArraySize2, imagedata2);
96. /\* ------------------------------------------------------------ CALLING FUNCTIONS FOR FILTERS ----------------------------------------------------------------------------\*/
98. //Mirror Filter
99. printf("\nIzaberite filter: \n");
100. printf("1. Mirror\n");
101. printf("2. Gray Scale\n");
102. printf("3. Brightness\n");
103. printf("4. Salt and Paper\n");
104. printf("5. Binary\n");
105. printf("6. Alpha Blend\n");
106. printf("7. Green Replace\n");
107. **int** ch;
108. scanf("%d", &ch);
109. **switch**(ch){
110. **case** 1:
111. BGR2.bgr2 = mirror(infoHeader.width, infoHeader.height, BGR1.bgr);
112. **break**;
113. **case** 2:
114. BGR2.bgr2 = gray(infoHeader.width, infoHeader.height, BGR1.bgr);
115. **break**;
116. **case** 3:
117. BGR2.bgr2 = brightness(infoHeader.width, infoHeader.height, BGR1.bgr, 50);
118. **break**;
119. **case** 4:
120. BGR2.bgr2 = saltAndPaper(infoHeader.width, infoHeader.height, BGR1.bgr);
121. **break**;
122. **case** 5:
123. BGR2.bgr2 = binary(infoHeader.width, infoHeader.height, BGR1.bgr);
124. **break**;
125. **case** 6:
126. BGR2.bgr2 = alphaBlend(infoHeader.width, infoHeader.height, BGR1.bgr, BGRA.bgr, 0.5);       //Alpha Channel from 0.0 - 1.0
127. **break**;
128. **case** 7:
129. BGR2.bgr2 = greenReplace(infoHeader.width, infoHeader.height, BGR1.bgr, BGRA.bgr);
130. **break**;
131. **default**:
132. printf("Niste uneli validan broj!");
133. }


137. /\* ------------------------------------------------------------ END OF CALLING FUNCTIONS FOR FILTERS ----------------------------------------------------------------------------\*/
139. // Write Pixels to the image
140. writeImageData(infoHeader.width, infoHeader.height, infoHeader.bits, BGR2.bgr2, imagedata, header.offset);



145. system("pause");
146. **return** 0;
147. }

### GETLINE.C

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "getLine.h"
6. **char**\* getln()
7. {
8. printf("Welcome LowLevel32\n");
9. printf("Please, enter the image name: ");
10. **char** \*line = NULL, \*tmp = NULL;
11. **int** size = 0, index = 0;
12. **int** ch = EOF;
14. **while** (ch) {
15. ch = getc(stdin);
17. // Check if we need to stop.
18. **if** (ch == EOF || ch == '\n')
19. ch = 0;
21. // Check if we need to expand.
22. **if** (size <= index) {
23. size += **sizeof**(**char**);
24. tmp = (**char** \*)realloc(line, size);
25. **if** (!tmp) {
26. free(line);
27. line = NULL;
28. **break**;
29. }
30. line = tmp;
31. }
33. // Actually store the thing.
34. line[index++] = ch;
35. }
37. **return** line;
38. }

### IMAGEDATA:C

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <math.h>
5. #include "imageData.h"

8. **int**\* imageData(**char** \*filename){
10. unsigned **int** \*imagedata;
11. // Do malloc for first 50 bytes
12. imagedata = malloc(**sizeof**(**int**)\*50);
13. **int** bytevalue;
14. **int** offset = -1;
15. **int** index = 0;
16. **int** increase = 1;
17. **FILE**\* file;
18. file = fopen(filename, "rb");
20. **if**(file){
21. **do**{
22. bytevalue = fgetc(file);
23. imagedata[index] = bytevalue;
24. index++;
26. **if**(index==13){
27. offset = imagedata[10];
28. }
29. **if**((index > offset)&&(offset!=-1)){
30. **break**;
31. }
33. /\* When index with mod 50 be 0 increase memory size by 50 \*/
34. **if**(index % 50 == 0){
35. increase++;
36. **int** \*reallocState = (**int** \*)realloc(imagedata, increase\*50\***sizeof**(**int**));
37. **if**(!reallocState){
38. printf("Could not allocate memory!");
39. free(imagedata);
40. }**else**{
41. imagedata = reallocState;
42. }
43. }
45. }**while**(bytevalue != EOF);
46. }**else**{
47. printf("Error Ocurred.");
48. }
49. //Width, Height and Bits
50. **int** width = imagedata[19]\*256 + imagedata[18];
51. **int** height =  imagedata[24 - 1]\*256 + imagedata[23 - 1];
52. **int** bits = imagedata[28]+imagedata[29];
54. // Row Size and PixelArraySize
55. **int** rowSize = floor((bits \* width +31)/32)\*4;
56. **int** pixelArraySize = rowSize \* abs(height) + offset;
57. **int** actualRow = width \* 3;
58. **int** \*reallocState = (**int** \*)realloc(imagedata, pixelArraySize\***sizeof**(**int**));
59. **if**(!reallocState){
60. printf("Could not allocate memory!");
61. free(imagedata);
62. }**else**{
63. imagedata = reallocState;
64. }
65. **int** row = 0;
66. **while**(index < pixelArraySize){
67. bytevalue = fgetc(file);
68. imagedata[index] = bytevalue;
69. index++;
71. row++;
72. **int** padding = rowSize - actualRow;
73. **if**(row == actualRow){
74. **while**(padding != 0){
75. bytevalue = fgetc(file);
76. padding--;
77. }
78. row = 0;
79. }
81. }
82. fclose(file);
84. **return** imagedata;
85. }

### WRITEIMAGEDATA.C

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "writeImageData.h"
6. **void** writeImageData(**int** width, **int** height, **int** bits, unsigned **short** \*\*bgr2, **int** \*imagedata, unsigned **int** offset){
7. **FILE** \*bmpCompression;
8. **int** actualRow = width \* 3;
9. **int** rowSize = floor((bits \* width +31)/32)\*4;
10. bmpCompression = fopen("maca3.bmp", "wb");
11. **int** row = 0;
12. **int** g = 0;
13. **while**(g<138){
14. fputc(imagedata[g], bmpCompression);
15. g++;
16. }
18. **int** index = 138;
19. **int** k;
20. **int** i;
21. **int** j;
22. **int** c;
23. **for**(i = 0; i < height; i++){
24. **for**(j = 0; j < width; j++){
25. **for**(c = 0; c < 3; c++){
26. imagedata[index] = bgr2[i\*width+j][c];
27. fputc(imagedata[index], bmpCompression);
28. index++;
29. }
30. }
31. **int** padding = rowSize - actualRow;
32. **while**(padding != 0){
33. imagedata[index] = 0;
34. fputc(imagedata[index], bmpCompression);
35. index++;
36. padding--;
37. }
38. }
40. fclose(bmpCompression);
41. }

### GETBGR.C

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "getBGR.h"
6. unsigned **short**\*\* getBGR(**int** height, **int** width, **int** offset, **int** pixelArraySize, unsigned **int** \*imagedata){
7. unsigned **short** \*\*bgr = malloc((height\*width) \* **sizeof**(**short**)\*3);
8. **int** h = offset;
9. **int** i;
10. **int** j;
11. **int** c;
12. **int** row = 0;
13. **for**(i = 0; i < height; i++){
14. **for**(j = 0; j < width; j++){
15. **if**((bgr[i\*width+j] = malloc(3 \* **sizeof**(**short**))) == NULL){
16. free(bgr);
17. **break**;
18. }
19. **for**(c = 0; c < 3; c++){
20. bgr[i\*width+j][c] = imagedata[h];
21. h++;
22. }
23. }
24. }
25. **return** bgr;
26. }

### STRUCT.H

1. // Struct for File Header ( 14 Bytes )
2. **typedef** **struct** {
3. unsigned **short** **int** type[2];                 /\* Magic identifier            \*/
4. unsigned **int** size;                       /\* File size in bytes          \*/
5. unsigned **short** **int** reserved1, reserved2;
6. unsigned **int** offset;                     /\* Offset to image data, bytes \*/
7. } HEADER;
8. // Struct for Bitmap Information ( 40 Bytes )
9. **typedef** **struct** {
10. unsigned **int** size;               /\* Header size in bytes      \*/
11. **int** width,height;                /\* Width and height of image \*/
12. unsigned **short** **int** planes;       /\* Number of colour planes   \*/
13. unsigned **short** **int** bits;         /\* Bits per pixel            \*/
14. unsigned **int** compression;        /\* Compression type          \*/
15. unsigned **int** imagesize;          /\* Image size in bytes       \*/
16. **int** xresolution,yresolution;     /\* Pixels per meter          \*/
17. unsigned **int** ncolours;           /\* Number of colours         \*/
18. unsigned **int** importantcolours;   /\* Important colours         \*/
19. } INFOHEADER;
21. // Struct for bitmap BGR ( Blue Green Red )
22. **typedef** **struct** {
23. unsigned **short** \*\*bgr;
24. }BGR;
25. **typedef** **struct** {
26. unsigned **short** \*\*bgr2;
27. }BGR2;

### FALPHABLEND.C

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "fAlphaBlend.h"
6. unsigned **short**\*\* alphaBlend(**int** height, **int** width, unsigned **short** \*\*bgr, unsigned **short** \*\*bgr1, **int** alpha){
7. unsigned **short** \*\*bgr2 = malloc((height\*width) \* **sizeof**(**short**)\*3);
8. **int** i;
9. **int** j;
10. **int** c;
11. **int** gray = 0;
12. **int** sum = 0;
13. **for**(i = 0; i < height; i++){
14. **for**(j = 0; j < width; j++){
15. // Malloc for bgr2 2 D array
16. **if**((bgr2[i\*width+j] = malloc(3 \* **sizeof**(**short**))) == NULL){
17. free(bgr2);
18. **break**;
19. }
21. **for**(c = 0; c < 3; c++){
22. bgr2[i\*width+j][c] = bgr1[i\*width+j][c]\*alpha + bgr[i\*width+j][c]\*(1.0-alpha);
23. }
25. }
26. }
27. **return** bgr2;
28. }

### FBINARY.C

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "fBinary.h"
6. unsigned **short**\*\* binary(**int** height, **int** width, unsigned **short** \*\*bgr){
7. unsigned **short** \*\*bgr2 = malloc((height\*width) \* **sizeof**(**short**)\*3);
8. **int** i;
9. **int** j;
10. **int** c;
11. **int** gray = 0;
12. **int** sum = 0;
13. **for**(i = 0; i < height; i++){
14. **for**(j = 0; j < width; j++){
15. // Malloc for bgr2 2 D array
16. **if**((bgr2[i\*width+j] = malloc(3 \* **sizeof**(**short**))) == NULL){
17. free(bgr2);
18. **break**;
19. }
20. /\* Gray Scale thing to do \*/
21. sum = bgr[i\*width+j][0] + bgr[i\*width+j][1] + bgr[i\*width+j][2];
22. gray = sum/3;
23. **if**(gray > 100){
25. **for**(c = 0; c < 3; c++){
26. bgr2[i\*width+j][c] = 255;
27. }
28. sum = 0;
29. gray = 0;
31. }**else**{
33. **for**(c = 0; c < 3; c++){
34. bgr2[i\*width+j][c] = 0;
35. }
36. sum = 0;
37. gray = 0;
39. }
41. }
42. }
43. **return** bgr2;
44. }

### FBRIGHTNESS.C

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include "fBrightness.h"
4. .
5. unsigned **short**\*\* brightness(**int** height, **int** width, unsigned **short** \*\*bgr, **int** brightness){

8. unsigned **short** \*\*bgr2 = malloc((height\*width) \* **sizeof**(**short**)\*3);
9. **int** i;
10. **int** j;
11. **int** c;
12. **for**(i = 0; i < height; i++){
13. **for**(j = 0; j < width; j++){
15. **if**((bgr2[i\*width+j] = malloc(3 \* **sizeof**(**short**))) == NULL){
17. free(bgr2);
18. **break**;
19. }
20. **for**(c = 0; c < 3; c++){
21. **int** newPixel = bgr[i\*width+j][c]+brightness;
22. **if**(newPixel>255){ newPixel = 255; }
23. **if**(newPixel<0){ newPixel = 0; }
24. bgr2[i\*width+j][c] = newPixel;
25. }
26. }
27. }
28. **return** bgr2;
30. }

### FGRAY.C

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "fGray.h"
6. unsigned **short**\*\* gray(**int** height, **int** width, unsigned **short** \*\*bgr){
7. unsigned **short** \*\*bgr2 = malloc((height\*width) \* **sizeof**(**short**)\*3);
8. **int** i;
9. **int** j;
10. **int** c;
11. **int** gray = 0;
12. **int** sum = 0;
13. **for**(i = 0; i < height; i++){
14. **for**(j = 0; j < width; j++){
15. // Malloc for bgr2 2 D array
16. **if**((bgr2[i\*width+j] = malloc(3 \* **sizeof**(**short**))) == NULL){
17. free(bgr2);
18. **break**;
19. }
20. /\* Gray Scale thing to do \*/
21. sum = bgr[i\*width+j][0] + bgr[i\*width+j][1] + bgr[i\*width+j][2];
22. gray = sum/3;
23. **for**(c = 0; c < 3; c++){
24. bgr2[i\*width+j][c] = gray;
25. }
26. sum = 0;
27. gray = 0;
28. }
29. }
30. **return** bgr2;
31. }

### FGREENREPLACE:C

1. #include <stdio.h>
2. #include <stdlib.h>
4. #include "fGreenReplace.h"
6. unsigned **short**\*\* greenReplace(**int** height, **int** width, unsigned **short** \*\*bgr, unsigned **short** \*\*bgr1){
7. unsigned **short** \*\*bgr2 = malloc((height\*width) \* **sizeof**(**short**)\*3);
8. **int** i;
9. **int** j;
10. **int** c;
11. **int** gray = 0;
12. **int** sum = 0;
13. **for**(i = 0; i < height; i++){
14. **for**(j = 0; j < width; j++){
15. // Malloc for bgr2 2 D array
16. **if**((bgr2[i\*width+j] = malloc(3 \* **sizeof**(**short**))) == NULL){
17. free(bgr2);
18. **break**;
19. }
20. **int** r = bgr1[i\*width+j][2];
21. **int** g = bgr1[i\*width+j][1];
22. **int** b = bgr1[i\*width+j][0];
23. /\* Green replace \*/
24. **int** green[3];
25. **if**(g > r+30 && g > b-20){
26. green[0] = bgr[i\*width+j][0];
27. green[1] = bgr[i\*width+j][1];
28. green[2] = bgr[i\*width+j][2];
29. }**else**{
30. green[0] = bgr1[i\*width+j][0];
31. green[1] = bgr1[i\*width+j][1];
32. green[2] = bgr1[i\*width+j][2];
33. }
35. **for**(c = 0; c < 3; c++){
36. bgr2[i\*width+j][c] = green[c];
37. }
38. sum = 0;
39. gray = 0;
40. }
41. }
42. **return** bgr2;
43. }

### FMIRROR.C

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include "fMirror.h"
5. unsigned **short**\*\* mirror(**int** height, **int** width, unsigned **short** \*\*bgr){

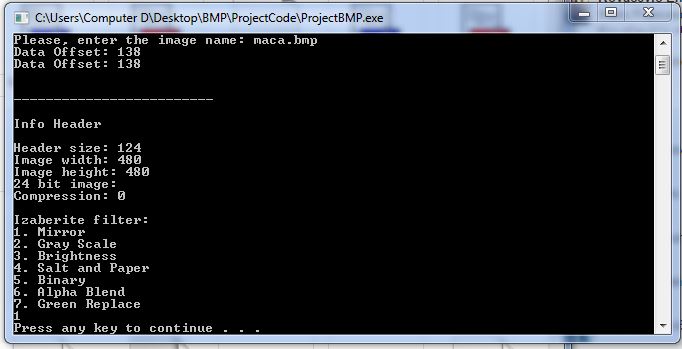
8. unsigned **short** \*\*bgr2 = malloc((height\*width) \* **sizeof**(**short**)\*3);
9. **int** i;
10. **int** j;
11. **int** c;
12. **for**(i = 0; i < height; i++){
13. **for**(j = 0; j < width; j++){
15. **if**((bgr2[i\*width+j] = malloc(3 \* **sizeof**(**short**))) == NULL){
17. free(bgr2);
18. **break**;
19. }
20. **for**(c = 0; c < 3; c++){
21. bgr2[i\*width+j][c] = bgr[i\*width+((width-1)-j)][c];
22. }
23. }
24. }
25. **return** bgr2;
27. }

### FSALTANDPAPER.C

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
4. #include "fSaltAndPaper.h"
6. unsigned **short**\*\* saltAndPaper(**int** height, **int** width, unsigned **short** \*\*bgr){
7. unsigned **short** \*\*bgr2 = malloc((height\*width) \* **sizeof**(**short**)\*3);
8. **int** i;
9. **int** j;
10. **int** c;
11. **int** gray = 0;
12. **int** sum = 0;
13. srand(time(NULL));
14. **for**(i = 0; i < height; i++){
15. **for**(j = 0; j < width; j++){
16. // Malloc for bgr2 2 D array
17. **if**((bgr2[i\*width+j] = malloc(3 \* **sizeof**(**short**))) == NULL){
18. free(bgr2);
19. **break**;
20. }
21. /\* Gray Scale thing to do \*/
22. sum = bgr[i\*width+j][0] + bgr[i\*width+j][1] + bgr[i\*width+j][2];
23. gray = sum/3;
24. **int** randNum1 = rand()%100;
25. **if**(randNum1 < 5){
26. **int** randNum2 = rand()%2;
28. **if**(randNum2 == 0){
29. **for**(c = 0; c < 3; c++){
30. bgr2[i\*width+j][c] = 0;
31. }
32. }**else**{
33. **for**(c = 0; c < 3; c++){
34. bgr2[i\*width+j][c] = 255;
35. }
36. }
37. }**else**{
38. **for**(c = 0; c < 3; c++){
39. bgr2[i\*width+j][c] = gray;
40. }
41. }
43. sum = 0;
44. gray = 0;
45. }
46. }
47. **return** bgr2;
48. }

## 2.3 Izgled programa

Program je veoma jednostavan za korišćenje. Unesemo naziv slike koju zelimo obraditi pa zatim željeni filter.



## 2.4 Prikaz obrađenih slika.



*Mirror slika 2*



*Originalna slika 1*



Gray Scale slika 3



Brightness slika 4



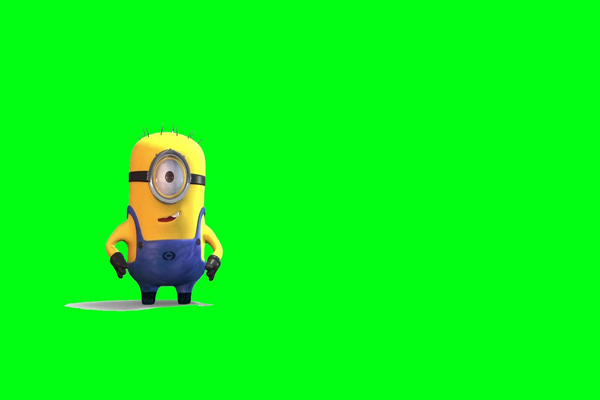
Binary slika 6



Salt and Paper slika 5



Originalna slika 8



Originalna slika 7



Alpha slika 10



Green Replace slika 9

# 3. Problemi programa

Samo molim Vas obratite paznju na upis slike. Trazice vam da dva puta unesete naziv slike. U koliko birate opcije od 1 - 5, sliku koju zelite da filtirate unesite na prvi ulaz, a na drugi ulaz unesite bilo koju postojecu sliku u folderu ( Ovde je bitno samo da ne pukne program, jer mora da uzme podatke nekog fajla). U koliko birate opciju 6 - 7 morate uneti 2 razliticite slike istih dimenzija. Izvinjavam se zbog toga, ali stvarno nemam vremena da to sredim.

I jos nesto ne radi. Mirror efekat ne radi kada se unese slika razlicitih dimenzija. Stvarno ne znam zasto, ali iskreno nisam ni pokusao da sredim zbog manjka vremena.

U sustini ko zeli da radi na ovom projektu i doprinese istom neka mi se javi kako bi mogli azurirati projekat na git - u.