LBYEC72 Final Project Overview

Mini-Image Processing Project

Melvin Kong Cabatuan

Objectives

- Introduce bitmaps and colour depth.
- Perform basic image operations, i.e. addition, subtraction, multiplication, etc.
- Convert RGB to Grey Scale Image.
- Provide a general overview in image processing in C using Cimg library.

Tools

- **DevC++ Compiler** - a full-featured Integrated Development Environment (IDE) for the C/C++ programming language.

  http://www.bloodshed.net/devcpp.html

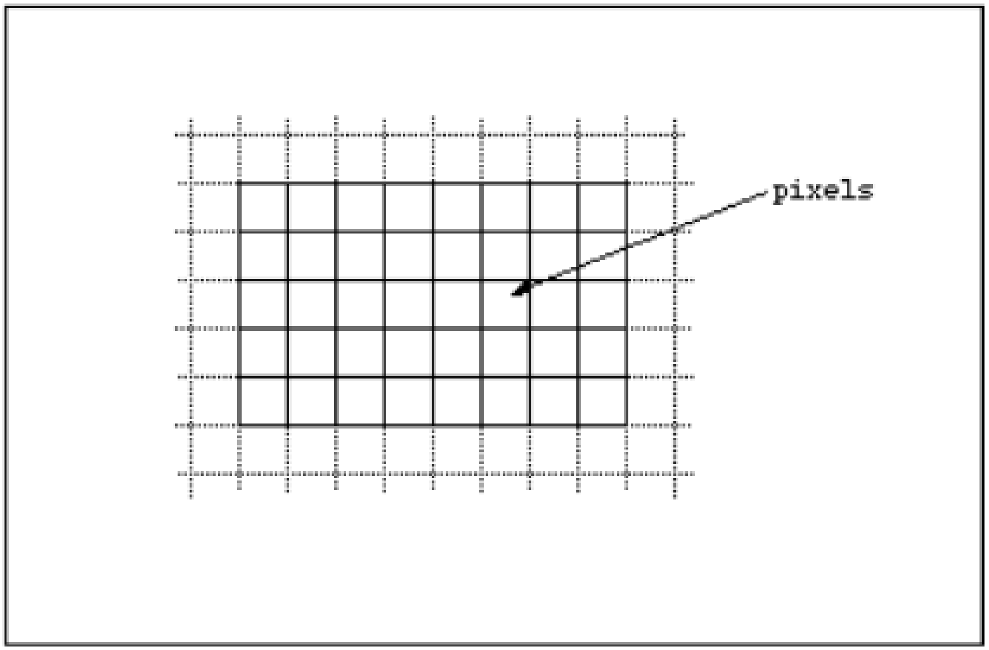
- **CImg library** - a small, open source, C++ toolkit for image processing. Choose CImg 1.2.5 version for compatibility with DevC++.

  http://cimg.sourceforge.net/news.shtml

- **Paint** - a simple graphics painting program that has been included with all versions of Microsoft Windows. We use this for editing pictures in 24 bit .bmp files. Note: You can use any other picture editing tool.

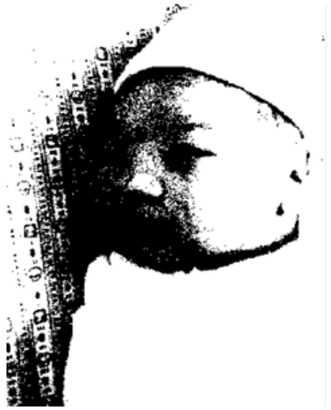
Introduction

- **Bitmaps** are defined as a regular rectangular mesh of cells called **pixels**, each pixel containing a colour value.

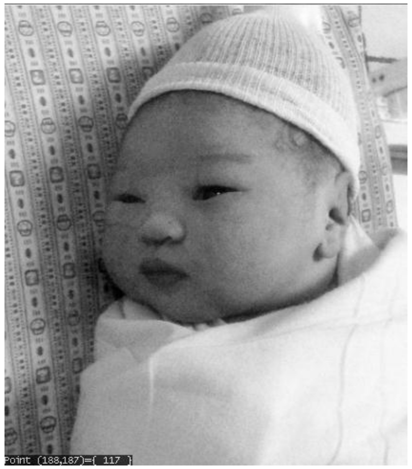


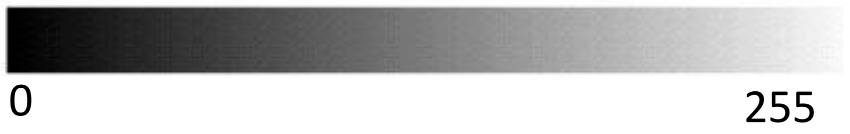
Colour Depth

- **1 bit (black and white)** – monochrome



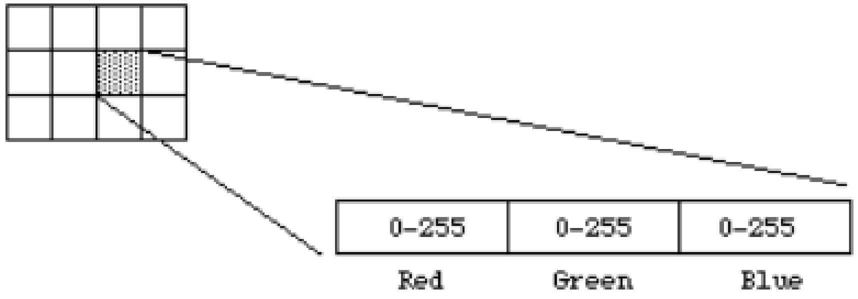
- **8 bit greys** - each pixel takes 1 byte (8 bits) of storage resulting in 256 different states.

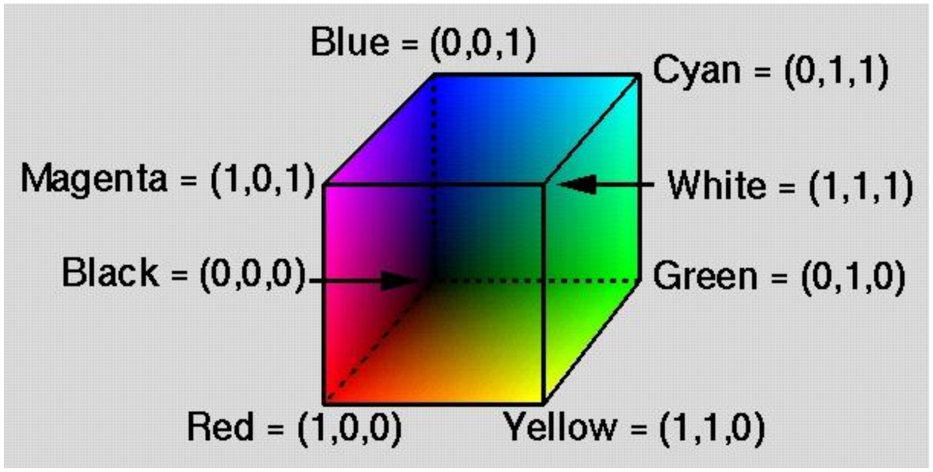




Colour Depth

- **24 bit RGB (true color)** - 8 bits allocated to each red, green, and blue component.





Example:

The height of the image is 24.
The width of the image is 28.

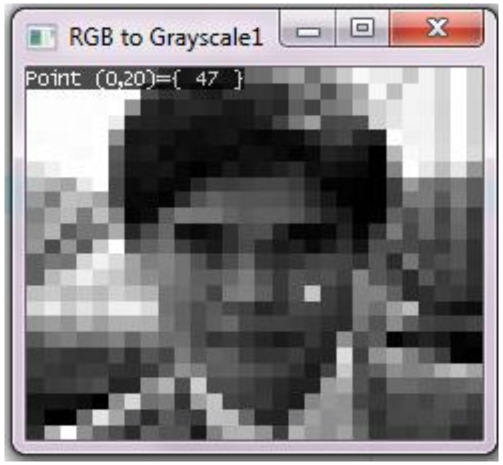
```
253 253 253 253 253 253 253 253 253 252 253 253 254 250 252 254 255 255 217 227
227 181 194 192 148 157 159 124 134 136 138 147 146 146 153 150 94 97 99 86 103
99 184 194 197 200 219 228 136 151 158 82 91 87 47 66 67 49 67 66 40 50 48 19 25
 21 27 29 25 44 47 46 253 253 253 253 253 253 253 253 253 253 253 253 253 252 25
3 252 255 254 237 244 242 167 181 179 113 125 124 161 170 171 173 183 182 70 80
78 116 122 124 174 191 190 194 205 208 223 241 247 147 162 170 112 121 119 51 70
 69 42 62 61 51 63 61 19 25 22 0 1 0 174 178 180 253 253 253 253 253 253 253 253
 253 253 253 253 252 252 253 254 252 252 255 255 255 155 169 169 185 196 194 177
 189 187 88 97 96 144 153 153 109 118 124 198 218 218 175 187 191 190 207 214 13
5 151 160 121 130 128 41 62 63 36 58 58 53 65 65 15 24 21 0 2 1 254 255 255 253
252 253 252 253 253 253 253 253 252 252 253 251 254 253 254 250 252 255 255 255
223 237 238 178 193 187 117 128 125 110 122 122 85 93 94 205 217 224 224 245 246
 188 200 204 215 231 238 166 183 193 102 113 112 37 59 61 46 69 69 59 73 74 9 17
 16 106 109 107 140 145 149 248 254 253 253 251 253 254 251 251 255 255 254 255
255 255 255 255 255 255 255 255 255 255 255 153 168 170 176 196 194 110 121 121
189 205 209 228 246 252 202 221 222 181 187 192 193 212 214 152 161 170 107 118
120 43 57 57 49 66 69 65 84 84 0 2 0 222 229 229 97 102 107 253 252 253 255 254
255 255 255 255 233 237 236 144 147 145 86 91 88 78 83 81 69 74 72 80 71 78 129
131 131 172 172 171 255 255 255 197 227 227 177 209 206 200 211 216 232 241 251
163 180 192 119 136 140 65 84 82 38 57 55 25 42 41 135 146 144 163 168 168 73 78
 84 255 254 255 255 255 255 142 151 150 30 42 43 15 24 23 19 28 27 16 22 23 9 13
 14 7 11 12 4 7 3 50 32 28 94 74 72 167 161 160 152 166 159 182 196 195 223 247
250 161 182 193 86 107 111 33 54 51 35 55 52 28 44 45 207 217 217 81 89 91 37 45
 51 255 255 255 68 80 76 29 42 43 28 36 39 18 28 27 24 31 31 18 19 21 20 19 21 8
 9 9 13 16 8 86 70 60 108 95 83 142 122 111 143 143 127 180 179 172 218 221 223
163 178 190 113 131 136 72 92 93 44 65 70 130 144 147 107 116 117 39 47 49 90 10
1 106 142 149 147 39 51 53 20 31 33 27 32 35 17 23 26 10 11 14 22 23 22 7 11 13
8 5 0 105 86 79 116 98 94 126 113 87 144 131 113 142 126 109 132 111 93 128 110
85 138 117 98 110 110 102 57 75 77 76 98 100 171 181 186 118 124 128 89 92 97 80
 84 93 69 76 74 46 56 59 33 43 45 27 34 36 10 15 18 16 17 19 7 9 8 10 10 7 55 38
 31 133 119 109 18 19 16 72 60 46 129 110 96 132 109 95 131 103 86 118 95 72 118
 95 68 128 107 84 130 124 110 136 148 150 69 80 87 43 54 57 65 73 74 79 85 82 88
 92 90 27 36 40 34 44 46 16 21 24 11 15 17 7 13 16 1 1 0 44 36 27 101 75 63 127
117 103 4 13 10 44 33 29 98 77 63 113 88 74 120 90 76 99 72 56 95 73 50 101 70 4
1 110 81 58 134 127 120 202 204 216 153 159 173 52 62 75 21 27 35 93 97 97 22 30
 33 30 38 42 15 20 23 18 21 22 9 14 18 12 10 7 76 59 45 114 95 77 132 113 99 20
17 15 32 23 16 65 47 32 111 87 74 89 61 49 109 85 69 109 89 79 71 48 32 119 93 7
3 113 89 64 87 73 62 195 194 197 183 194 207 120 130 144 77 77 72 47 54 52 19 27
 30 17 21 25 19 25 27 5 10 13 28 24 20 71 56 42 103 84 68 113 93 81 62 45 33 62
46 28 93 73 52 121 99 87 140 116 107 93 67 50 119 101 87 81 53 49 114 86 71 112
91 67 94 72 49 78 66 54 180 190 202 169 179 190 104 104 96 49 52 51 21 27 30 20
25 27 8 14 15 16 19 22 20 16 11 79 63 49 98 81 62 103 85 70 95 72 59 90 67 49 11
9 99 83 141 119 109 147 122 117 82 59 45 86 67 52 72 42 42 85 55 43 102 81 59 81
 65 48 78 56 34 108 105 99 173 182 191 113 109 100 53 54 51 43 46 46 32 33 31 3
8 8 15 17 19 28 20 16 83 67 51 97 81 63 109 90 78 95 75 64 60 38 24 70 51 39 96
78 70 110 89 82 55 33 24 81 61 48 56 31 30 70 47 35 90 70 52 62 52 40 78 52 26 4
9 35 21 144 149 154 153 146 136 47 48 42 37 39 38 67 66 64 9 13 13 9 11 13 39 30
 25 85 68 50 96 78 65 106 89 82 46 32 26 34 24 15 58 43 31 69 52 40 69 52 39 63
43 33 79 62 53 38 22 21 65 50 37 78 62 44 52 39 28 55 34 17 41 31 28 64 61 66 16
5 162 152 110 109 101 94 95 86 120 121 112 13 17 13 2 7 6 47 42 38 76 56 40 89 6
7 56 92 82 73 7 7 2 34 32 31 64 49 33 99 80 63 96 78 64 88 70 53 76 58 50 46 31
21 71 60 45 56 47 34 45 30 15 37 27 22 119 123 133 170 177 189 110 111 102 133 1
34 125 65 68 58 156 160 149 40 40 35 9 12 11 17 16 13 41 30 20 54 37 26 66 61 53
 5 6 2 13 12 8 76 63 52 94 72 59 199 196 196 89 63 48 73 52 38 70 52 35 49 35 21
 36 26 19 38 32 26 134 134 139 154 160 175 142 153 166 156 161 154 79 83 77 112
```

Example:

The height of the image is 24.
The width of the image is 28.

```
28 x 24 Image
( 0 , 0  ) = R253 G253 B253
( 0 , 1  ) = R253 G253 B253
( 0 , 2  ) = R253 G253 B253
( 0 , 3  ) = R253 G252 B253
( 0 , 4  ) = R248 G254 B253
( 0 , 5  ) = R253 G252 B253
( 0 , 6  ) = R255 G254 B255
( 0 , 7  ) = R255 G255 B255
( 0 , 8  ) = R142 G149 B147
( 0 , 9  ) = R69 G76 B74
( 0 , 10  ) = R88 G92 B90
( 0 , 11  ) = R93 G97 B97
( 0 , 12  ) = R77 G77 B72
( 0 , 13  ) = R104 G104 B96
( 0 , 14  ) = R113 G109 B100
( 0 , 15  ) = R153 G146 B136
( 0 , 16  ) = R165 G162 B152
( 0 , 17  ) = R110 G111 B102
( 0 , 18  ) = R156 G161 B154
( 0 , 19  ) = R199 G205 B203
( 0 , 20  ) = R247 G249 B248
( 0 , 21  ) = R235 G235 B235
( 0 , 22  ) = R235 G234 B237
( 0 , 23  ) = R202 G201 B206
( 0 , 24  ) = R243 G242 B246
( 0 , 25  ) = R214 G213 B218
( 0 , 26  ) = R255 G255 B255
( 0 , 27  ) = R210 G214 B216
( 1 , 0  ) = R253 G253 B253
( 1 , 1  ) = R253 G253 B253
( 1 , 2  ) = R253 G253 B253
( 1 , 3  ) = R252 G253 B253
( 1 , 4  ) = R253 G251 B253
( 1 , 5  ) = R255 G254 B255
( 1 , 6  ) = R255 G255 B255
( 1 , 7  ) = R68 G80 B76
( 1 , 8  ) = R39 G51 B53
( 1 , 9  ) = R46 G56 B59
( 1 , 10  ) = R27 G36 B40
( 1 , 11  ) = R22 G30 B33
( 1 , 12  ) = R47 G54 B52
( 1 , 13  ) = R49 G52 B51
( 1 , 14  ) = R53 G54 B51
( 1 , 15  ) = R47 G48 B42
( 1 , 16  ) = R110 G109 B101
( 1 , 17  ) = R133 G134 B125
( 1 , 18  ) = R79 G83 B77
( 1 , 19  ) = R130 G139 B135
( 1 , 20  ) = R166 G168 B172
( 1 , 21  ) = R249 G246 B251
( 1 , 22  ) = R190 G189 B194
( 1 , 23  ) = R244 G244 B243
( 1 , 24  ) = R243 G243 B242
( 1 , 25  ) = R214 G213 B218
( 1 , 26  ) = R255 G255 B255
( 1 , 27  ) = R212 G212 B216
( 2 , 0  ) = R253 G253 B253
( 2 , 1  ) = R253 G253 B253
( 2 , 2  ) = R253 G253 B253
( 2 , 3  ) = R253 G253 B253
( 2 , 4  ) = R254 G251 B251
( 2 , 5  ) = R255 G255 B255
( 2 , 6  ) = R142 G151 B150
( 2 , 7  ) = R29 G42 B43
( 2 , 8  ) = R20 G31 B33
( 2 , 9  ) = R33 G43 B45
( 2 , 10  ) = R34 G44 B46
( 2 , 11  ) = R30 G38 B42
( 2 , 12  ) = R19 G27 B30
( 2 , 13  ) = R21 G27 B30
( 2 , 14  ) = R43 G46 B46
( 2 , 15  ) = R37 G39 B38
( 2 , 16  ) = R94 G95 B86
( 2 , 17  ) = R65 G68 B58
( 2 , 18  ) = R112 G120 B111
( 2 , 19  ) = R122 G133 B129
( 2 , 20  ) = R176 G181 B185
( 2 , 21  ) = R209 G210 B213
( 2 , 22  ) = R228 G227 B232
( 2 , 23  ) = R244 G243 B243
( 2 , 24  ) = R242 G241 B242
( 2 , 25  ) = R213 G211 B216
( 2 , 26  ) = R255 G255 B255
( 2 , 27  ) = R212 G210 B214
( 3 , 0  ) = R252 G253 B253
( 3 , 1  ) = R253 G253 B253
…
…
( 22 , 13  ) = R108 G105 B99
( 22 , 14  ) = R49 G35 B21
( 22 , 15  ) = R41 G31 B28
( 22 , 16  ) = R119 G123 B133
( 22 , 17  ) = R154 G160 B175
( 22 , 18  ) = R132 G144 B159
( 22 , 19  ) = R134 G149 B161
( 22 , 20  ) = R82 G93 B101
( 22 , 21  ) = R44 G50 B54
( 22 , 22  ) = R66 G75 B75
( 22 , 23  ) = R67 G75 B77
( 22 , 24  ) = R68 G77 B80
( 22 , 25  ) = R72 G81 B84
( 22 , 26  ) = R60 G69 B72
( 22 , 27  ) = R52 G60 B63
( 23 , 0  ) = R44 G47 B46
( 23 , 1  ) = R174 G178 B180
( 23 , 2  ) = R254 G255 B255
( 23 , 3  ) = R140 G145 B149
( 23 , 4  ) = R97 G102 B107
( 23 , 5  ) = R73 G78 B84
( 23 , 6  ) = R37 G45 B51
( 23 , 7  ) = R90 G101 B106
( 23 , 8  ) = R80 G84 B93
( 23 , 9  ) = R79 G85 B82
( 23 , 10  ) = R21 G27 B35
( 23 , 11  ) = R120 G130 B144
( 23 , 12  ) = R169 G179 B190
( 23 , 13  ) = R173 G182 B191
( 23 , 14  ) = R144 G149 B154
( 23 , 15  ) = R64 G61 B66
( 23 , 16  ) = R170 G177 B189
( 23 , 17  ) = R142 G153 B166
( 23 , 18  ) = R134 G144 B156
( 23 , 19  ) = R153 G161 B178
( 23 , 20  ) = R98 G104 B117
( 23 , 21  ) = R48 G56 B60
( 23 , 22  ) = R56 G65 B65
( 23 , 23  ) = R54 G62 B63
( 23 , 24  ) = R65 G72 B75
( 23 , 25  ) = R62 G68 B72
( 23 , 26  ) = R40 G46 B50
( 23 , 27  ) = R45 G52 B56
```

8 bit greys

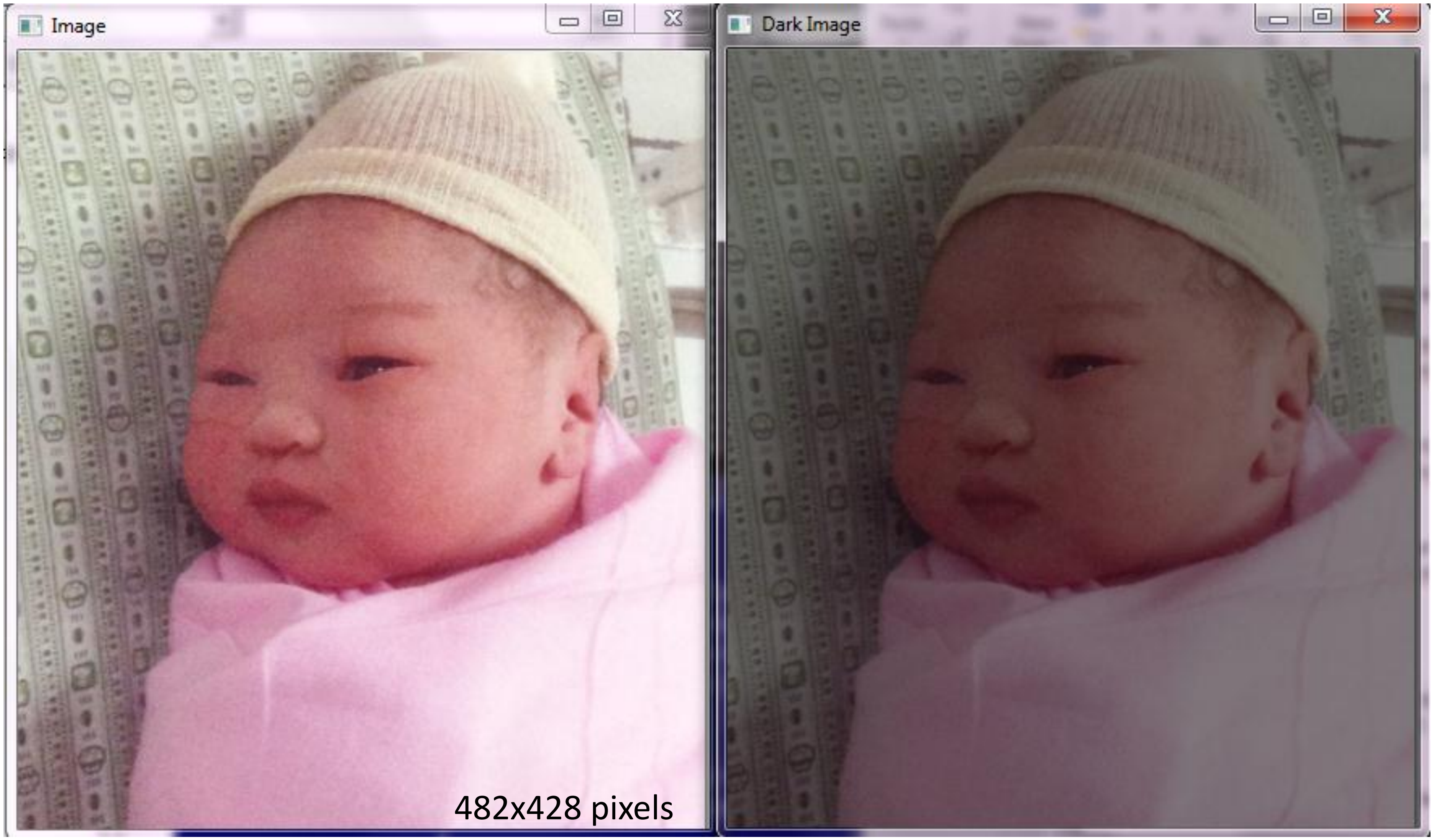


Point (0,20)={ 47 }

The height of the image is 24.
The width of the image is 28.

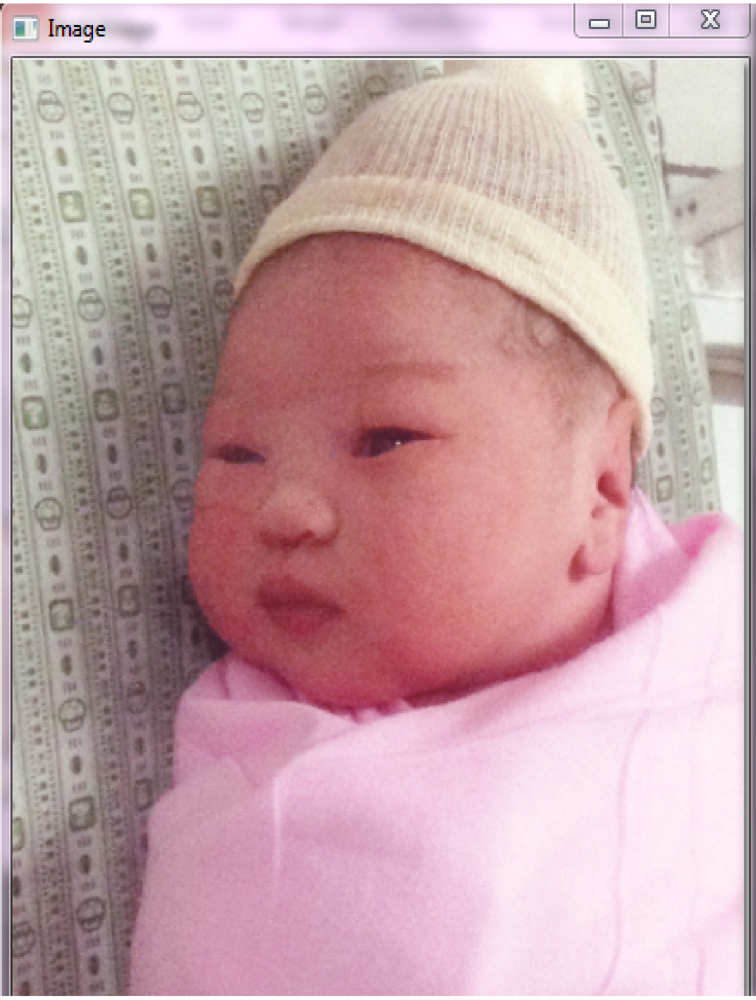
253 253 253 253 251 255 224 190 155 131 144 151 96 97 191 214 147 88 60 62 47 23 28 46 253 253
253 253 252 254 242 177 121 167 180 77 120 186 202 236 158 118 64 56 59 23 1 177 253 253 253
253 252 253 255 165 192 185 94 150 116 212 184 203 147 127 56 51 61 21 1 255 252 253 253 252
253 251 255 233 188 124 118 91 214 239 197 227 179 110 53 62 69 14 108 144 252 252 252 255 255
255 255 255 164 190 118 201 241 215 186 207 159 115 53 61 78 1 227 101 252 254 255 236 146 89
81 72 74 130 172 255 218 199 208 239 176 131 78 51 37 142 167 77 254 255 148 39 21 25 20 12 10
6 37 80 163 161 192 240 177 101 47 49 39 214 87 43 255 76 38 34 25 29 19 20 9 14 74 98 127 141
179 220 175 126 86 59 140 113 45 98 147 48 28 31 22 11 23 10 5 91 103 114 133 129 115 113 121
109 70 92 179 123 92 84 74 53 40 32 14 17 8 10 42 122 18 62 114 114 109 99 99 111 124 145 78 51
71 83 91 34 41 20 14 12 1 37 81 118 10 36 82 94 97 78 77 76 87 128 205 159 60 26 96 28 36 19 20
13 10 62 99 117 18 25 51 93 68 90 94 53 98 93 76 195 192 129 76 52 25 20 23 9 25 59 88 98 49 49
77 104 122 73 105 61 93 95 76 68 188 177 103 51 26 24 12 18 17 66 84 89 77 72 103 124 129 64 71
51 63 85 68 60 105 180 109 53 45 32 7 17 22 70 84 94 80 43 55 82 94 39 65 38 53 74 54 57 38 148
147 47 38 66 12 11 32 71 82 93 36 26 46 56 56 48 66 27 53 65 42 38 34 62 162 108 94 120 15 5 43
60 72 84 6 32 52 84 82 73 62 34 62 48 33 29 123 176 110 133 66 158 39 11 16 32 41 62 5 12 66 77
197 69 57 55 38 28 33 135 160 151 159 81 117 72 61 35 21 5 23 37 9 29 71 81 62 60 47 37 36 66 154
151 142 142 203 136 129 76 71 29 14 4 4 25 34 50 67 63 49 46 44 143 221 201 170 145 146 161 248
168 180 80 39 23 11 1 1 0 5 20 51 133 129 127 116 78 63 81 82 69 91 104 235 247 210 210 35 0 0 0
0 3 51 79 87 88 130 112 84 92 145 96 55 60 49 54 235 190 228 230 253 184 133 64 61 97 122 80 44
71 96 157 141 3 169 142 107 70 72 62 202 244 243 244 241 249 255 157 120 83 72 73 118 43 43 186
93 23 128 130 106 75 73 60 243 243 241 243 243 245 231 120 123 107 104 65 67 88 47 46 66 35 92
135 97 69 75 70 214 214 212 212 214 217 190 107 88 66 73 58 78 58 62 32 60 23 60 173 94 82 79 67
255 255 255 255 255 255 231 124 127 108 102 84 84 70 57 18 59 4 4 190 100 85 67 45 213 212 211
211 212 219 150 90 91 57 61 49 47 49 43 14 68 8 0 164 108 67 58 50

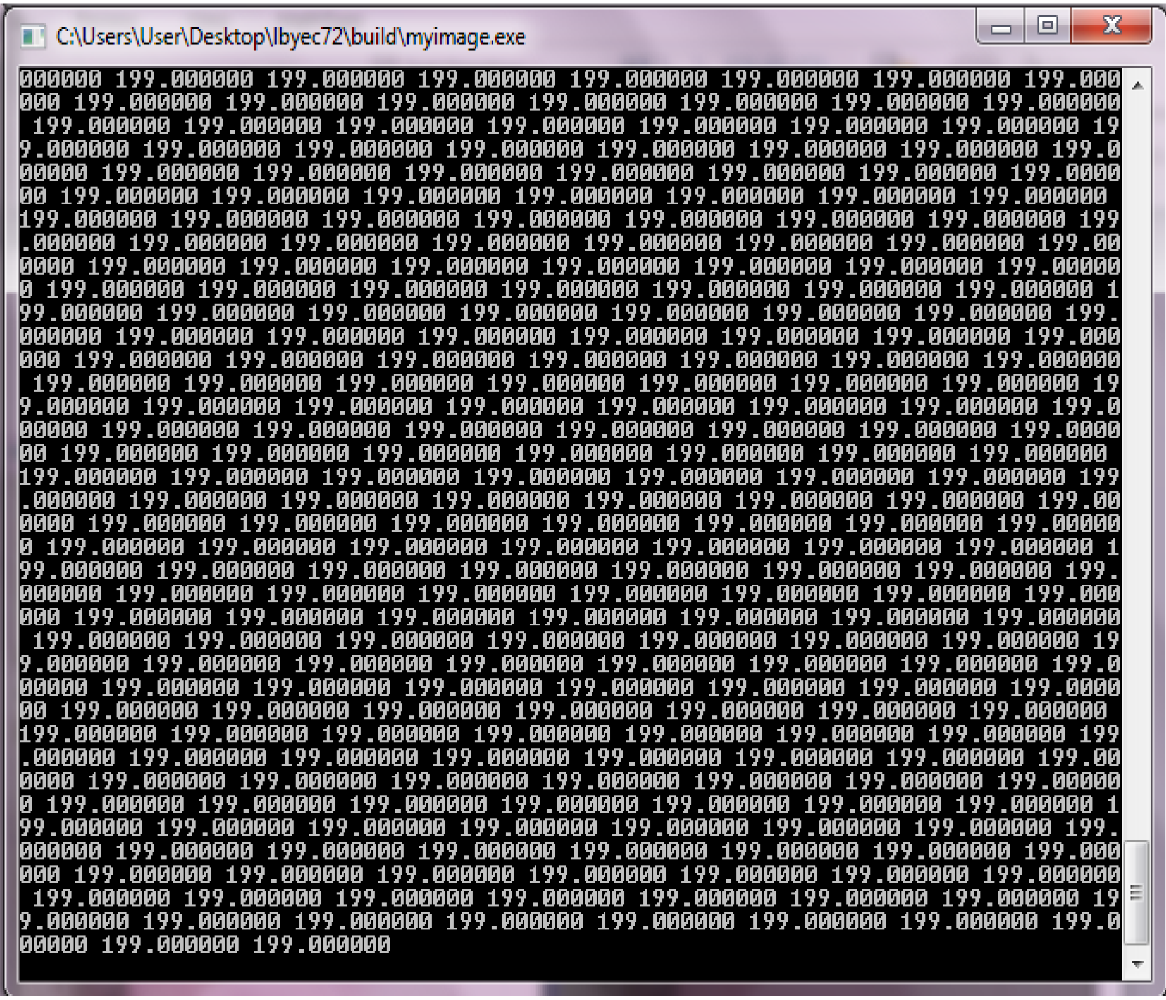
Simple Image Darkening



24 bit Bitmap Image (.bmp)

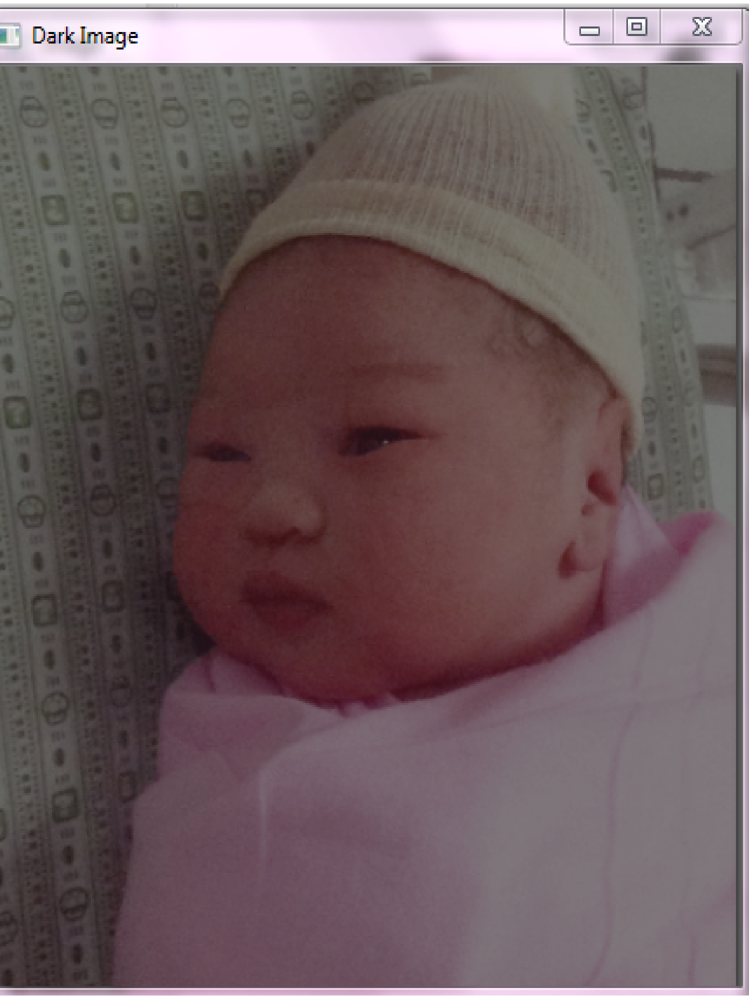
Original Image and its Array

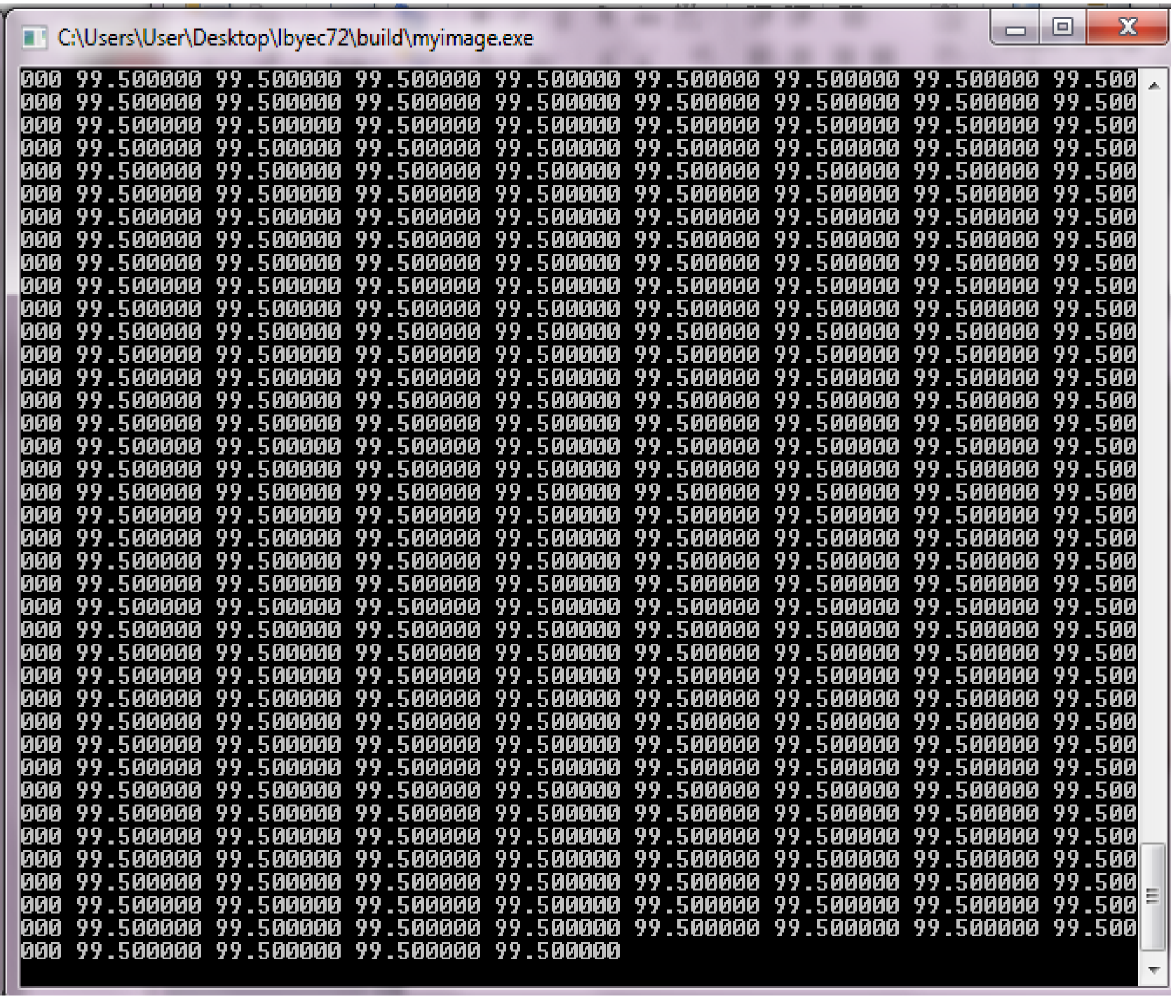




```
000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000
000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000
 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 19
9.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.0
00000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.0000
00 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000
199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199
.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.00
0000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.00000
0 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 1
99.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.
000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000
000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000
 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 19
9.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.0
00000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.0000
00 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000
199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199
.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.00
0000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.00000
0 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 1
99.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.
000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000
000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000
 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 19
9.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.0
00000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.0000
00 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000
199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199
.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.00
0000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.00000
0 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 1
99.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.
000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000
000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000
 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 19
9.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.000000 199.0
00000 199.000000 199.000000
```

Dark Image and its Array



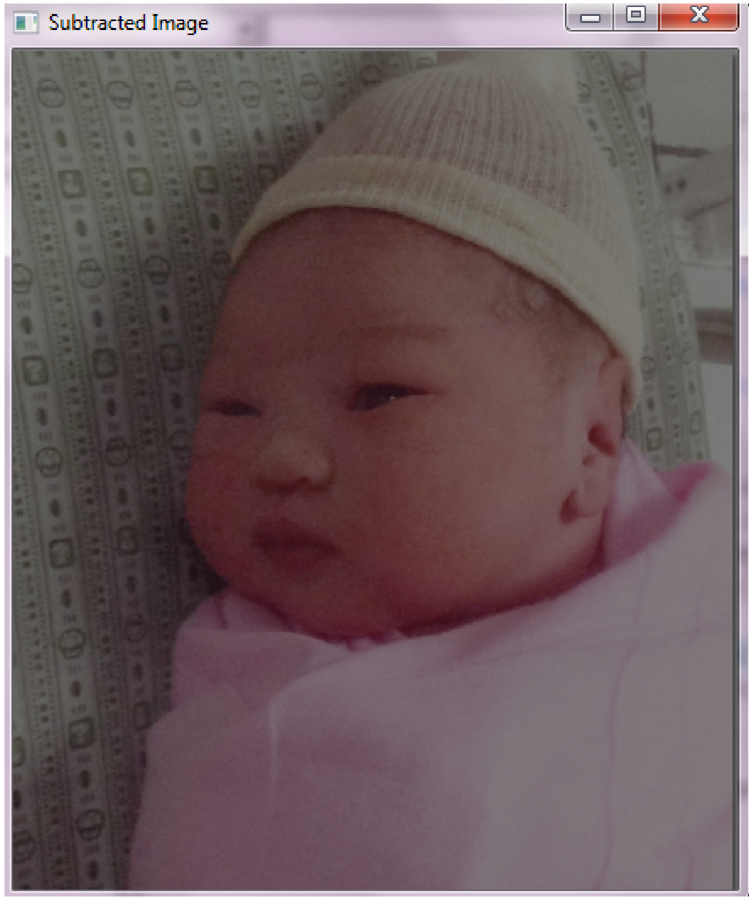


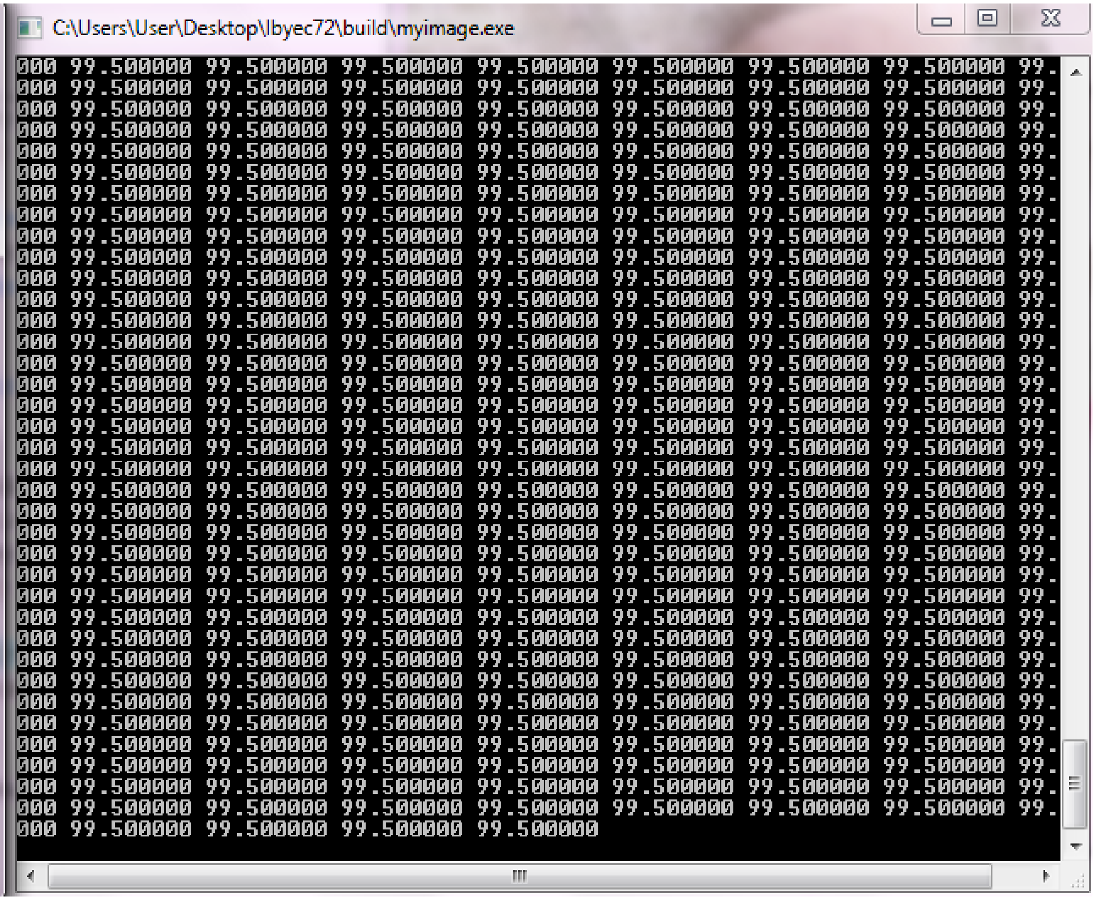
```
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500
000 99.500000 99.500000 99.500000 99.500000
```

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include "CImg.h"
using namespace cimg_library;
int main() {
   CImg<double> image("img/baby2.bmp");     // load baby2.bmp image
   const unsigned int width = image.dimx();     //retrieve image dimensions
   const unsigned int height = image.dimy();    //retrieve image dimensions

   printf("The height of the image is %d\n",height);  //display dimensions
   printf("The width of the image is %d\n",width);   //display dimensions
   system("PAUSE");

   CImgDisplay main_disp(image,"Image",0);      //display source image
   CImg<double> darkimage(image.dimx(),image.dimy(),1,3,0);
   for (int i=0;i<image.dimx();i++)
      for (int j=0;j<image.dimy();j++)
         for (int k=0;k<3;k++)
            {
            darkimage(i,j,0,k)=image(i,j,0,k)/2;
            printf("%f ",darkimage(i,j,0,k));
            }
         printf("\n");
   CImgDisplay dark_disp (darkimage,"Dark Image",0); //Display dark image

// Wait while the image is not closed

while (!main_disp.is_closed)
  main_disp.wait();

   return 0;
}
// EndOfMain
```

Subtracted Image and its Array




C:\Users\User\Desktop\lbyec72\build\myimage.exe

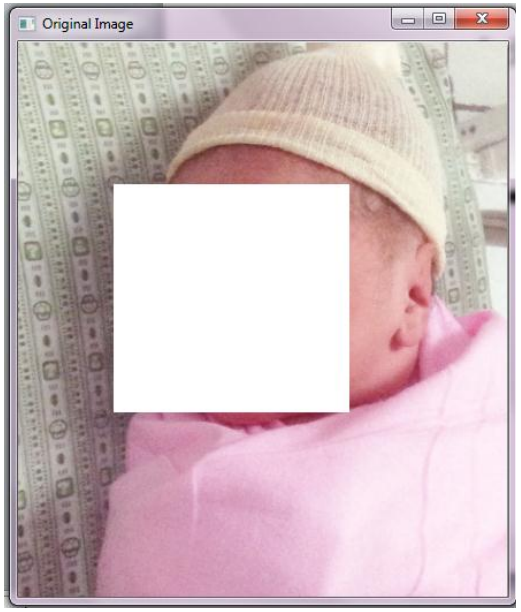
```
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.500000 99.
000 99.500000 99.500000 99.500000 99.500000
```

Note: CImg<T> is basically a 4th-dimensional array (width,height,depth,dim)

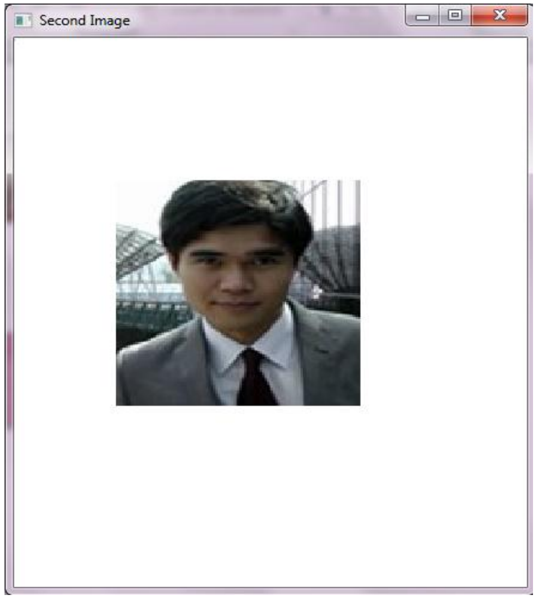
Subtracted Image Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include "CImg.h"
using namespace cimg_library;
int main() {

CImg<double> subimage(image.dimx(),image.dimy(),1,3,0);
   for (int i=0;i<image.dimx();i++)
      for (int j=0;j<image.dimy();j++)
         for (int k=0;k<3;k++)
            {
            subimage(i,j,0,k)=image(i,j,0,k) - darkimage(i,j,0,k);
            printf("%f ",subimage(i,j,0,k)); // print the dark image matrix
            }
         printf("\n");
   CImgDisplay sub_disp(subimage,"Subtracted Image",0); //display dark image

 // Wait while the image is not closed
      while (!main_disp.is_closed)
      main_disp.wait();
   return 0;
}
// EndOfMain
```

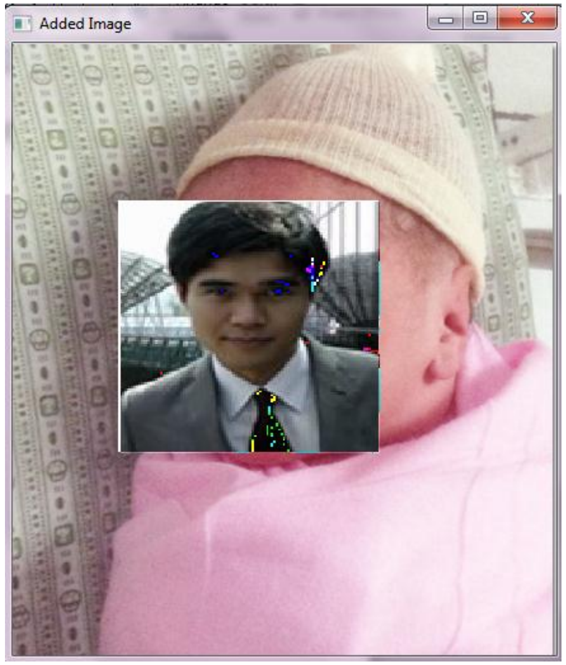
Image Addition



+



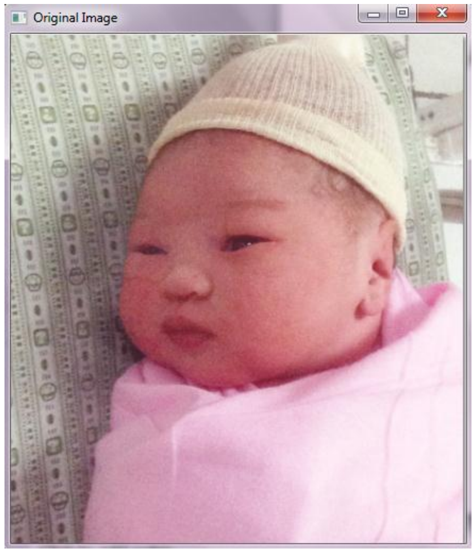
=



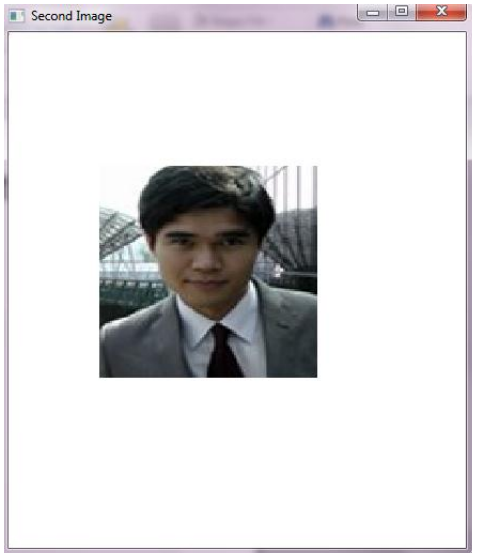
Addition Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include "../CImg.h"
using namespace cimg_library;

int main() {
   CImg<double> image("img/baby3.bmp"),image2("img/melvin_kyoto.bmp");
   CImgDisplay main_disp(image,"Original Image",0); //display the original image
   CImgDisplay second_disp(image2,"Second Image",0); //display second image

   CImg<double> sumimage(image.dimx(),image.dimy(),1,3,0); //storage of result
   for (int i=0;i<image.dimx();i++)
      for (int j=0;j<image.dimy();j++)
         for (int k=0;k<3;k++)
            {
            sumimage(i,j,0,k)=image(i,j,0,k) + image2(i,j,0,k);
            printf("%f ",sumimage(i,j,0,k)); // print the sum image matrix
            }
         printf("\n");
   CImgDisplay sum_disp(sumimage,"Added Image",0); //display the result

   while (!main_disp.is_closed) // Wait while the image is not closed
   main_disp.wait();
   return 0;
}
```

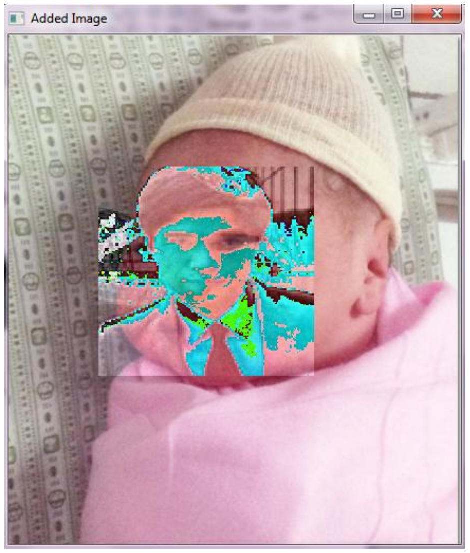
Addition Distortion



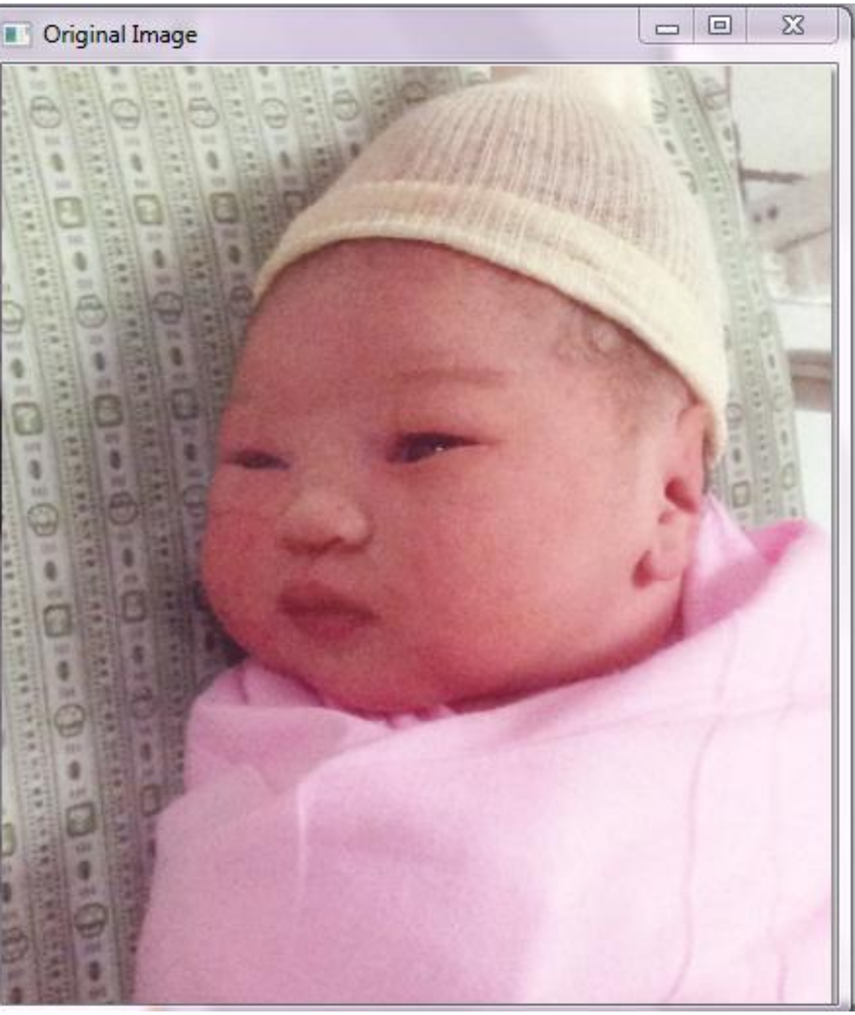
+



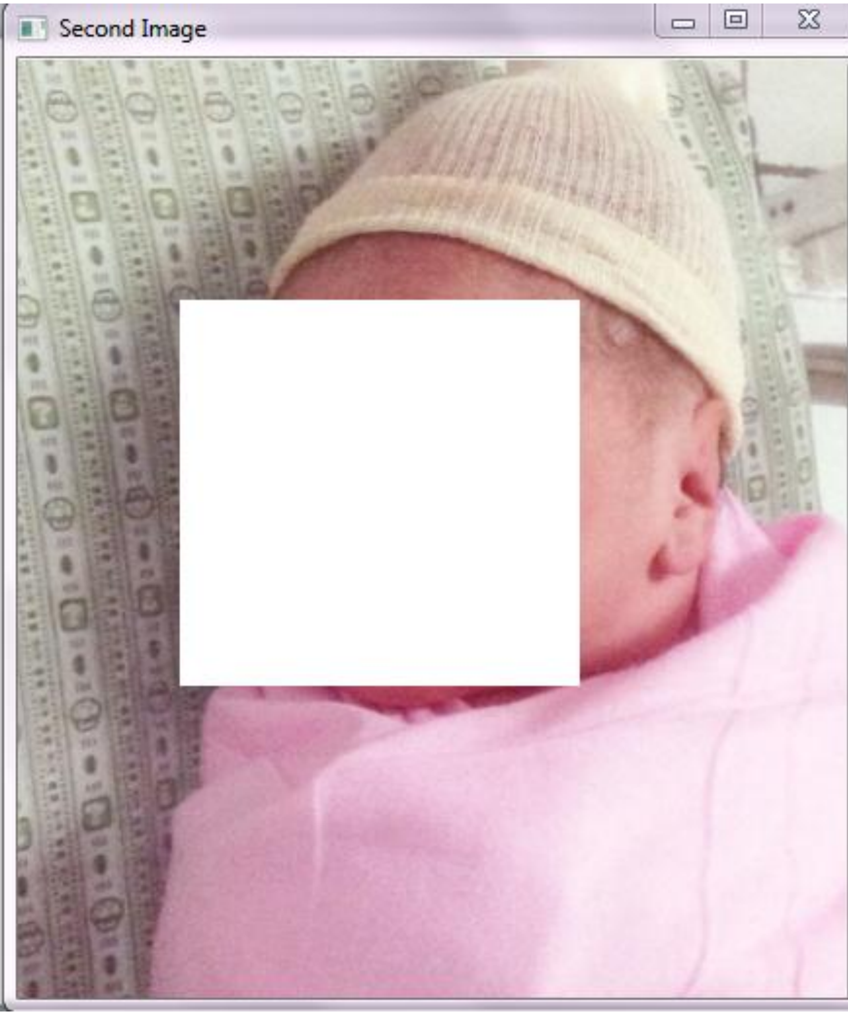
=



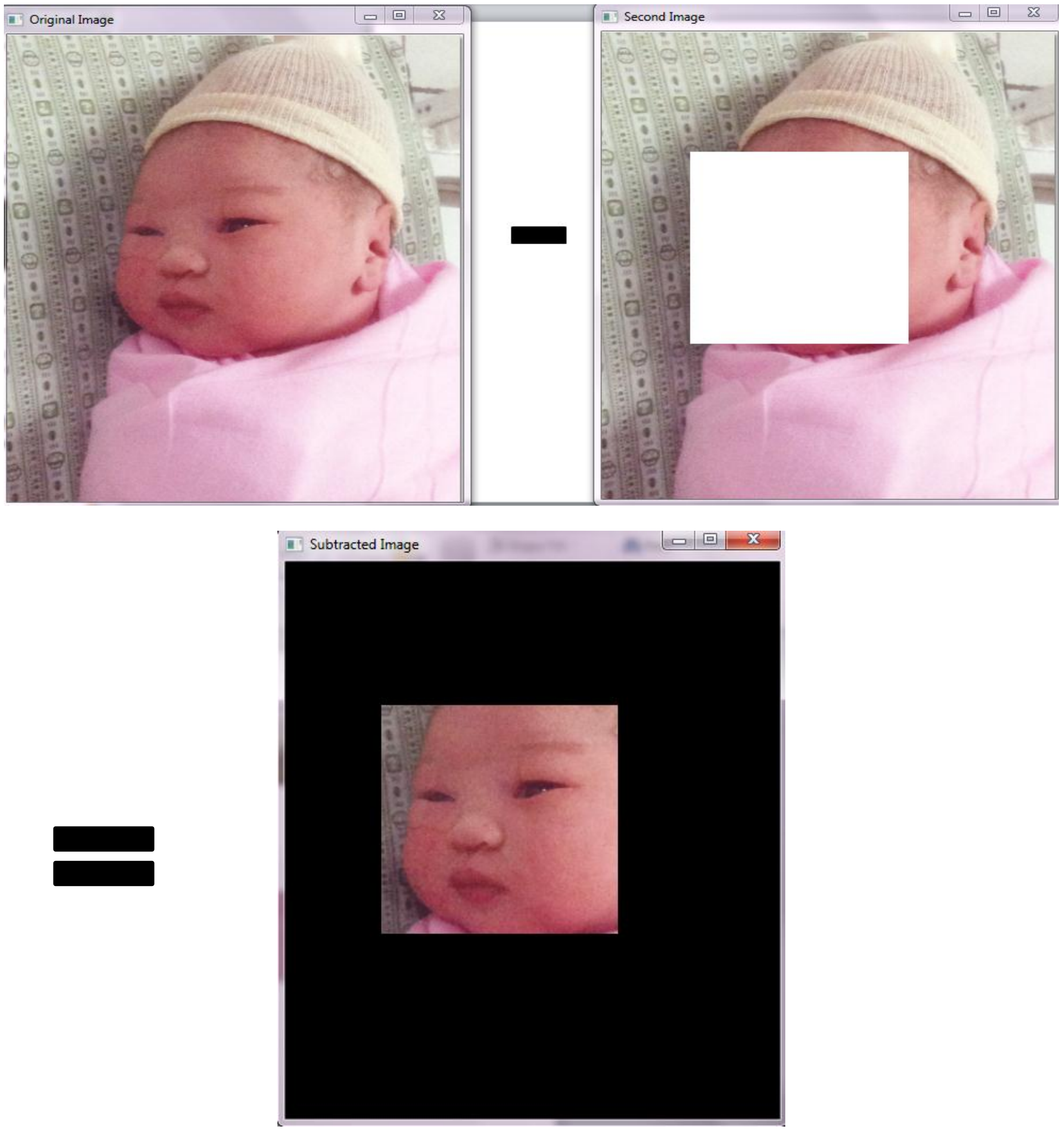
Subtraction



−



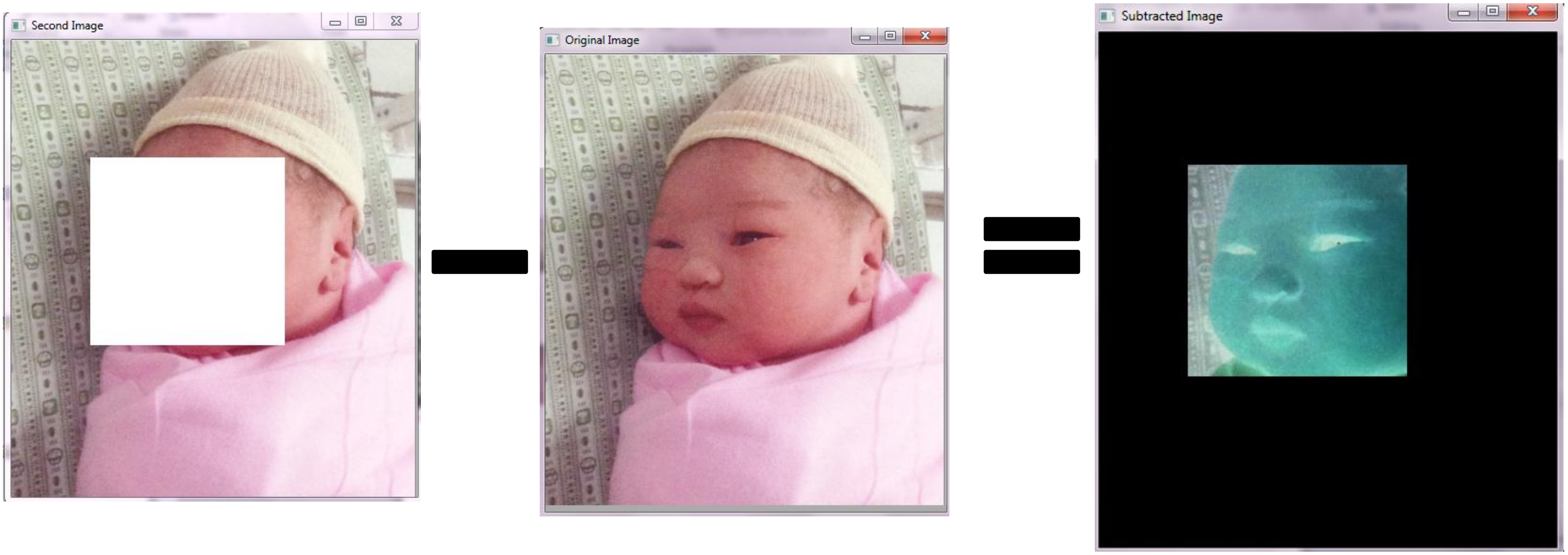
Subtraction



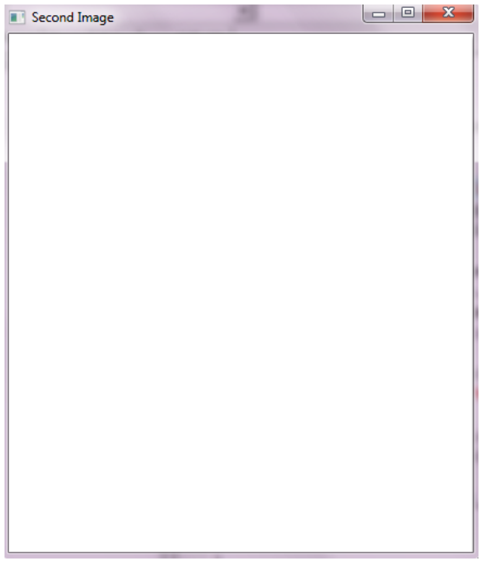
Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include "CImg.h"
using namespace cimg_library;

int main() {
    CImg<double> image("baby2.bmp"),image2("baby3.bmp");
    CImgDisplay main_disp(image,"Original Image",0); //display the original image
    CImgDisplay second_disp(image2,"Second Image",0); //display second image

    CImg<double> subimage(image.dimx(),image.dimy(),1,3,0); //storage of result
    for (int i=0;i<image.dimx();i++)
        for (int j=0;j<image.dimy();j++)
            for (int k=0;k<3;k++)
                {
                subimage(i,j,0,k)=image(i,j,0,k) - image2(i,j,0,k);
                printf("%f ",subimage(i,j,0,k)); // print the difference image matrix
                }
            printf("\n");
    CImgDisplay sub_disp(subimage,"Subtracted Image",0); //display second image

    while (!main_disp.is_closed) // Wait while the image is not closed
    main_disp.wait();
    return 0;
}
```

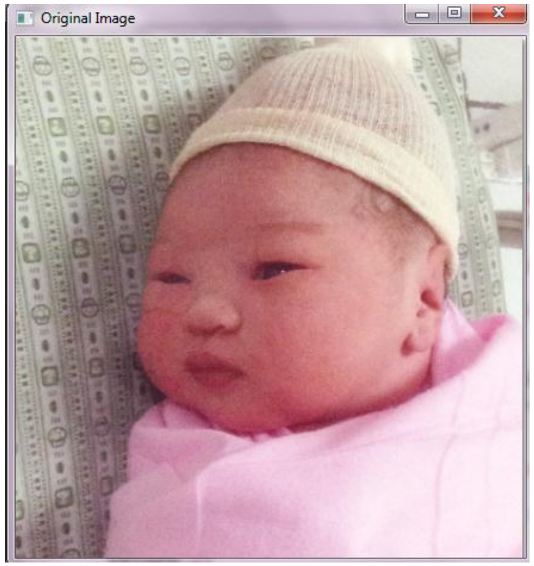
Negative



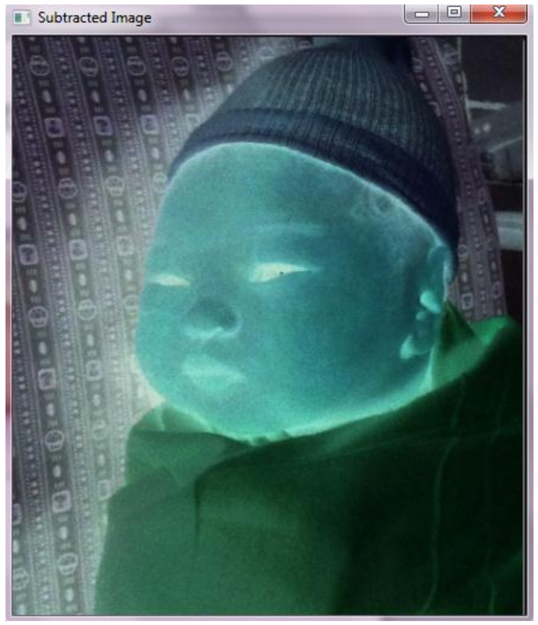
Negative



−



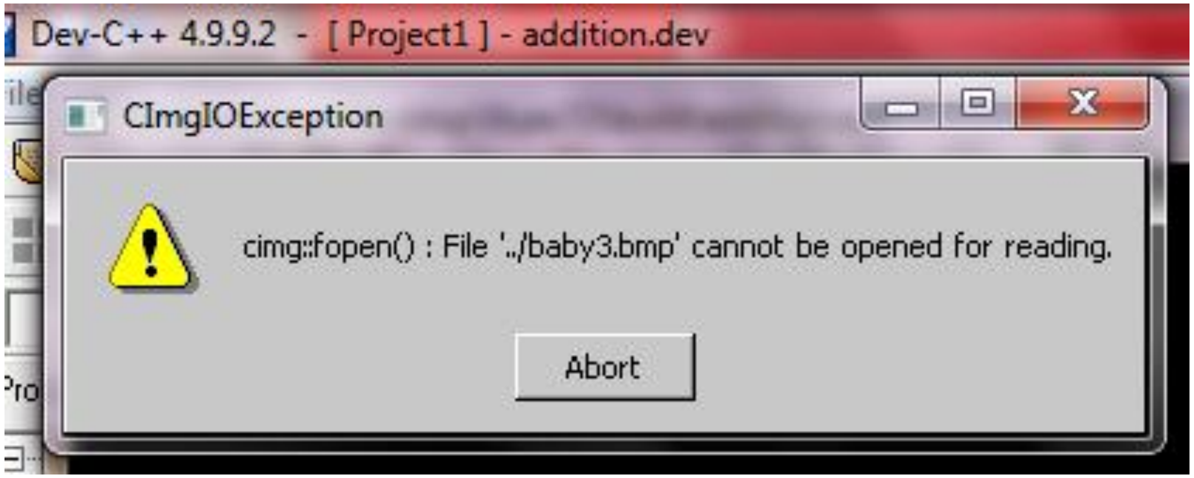
=



Addendum

Errors

- Devcpp cannot find the filename.bmp file.
- This error is caused by faulty directories.


Dev-C++ 4.9.9.2 - [Project1] - addition.dev

CImgIOException

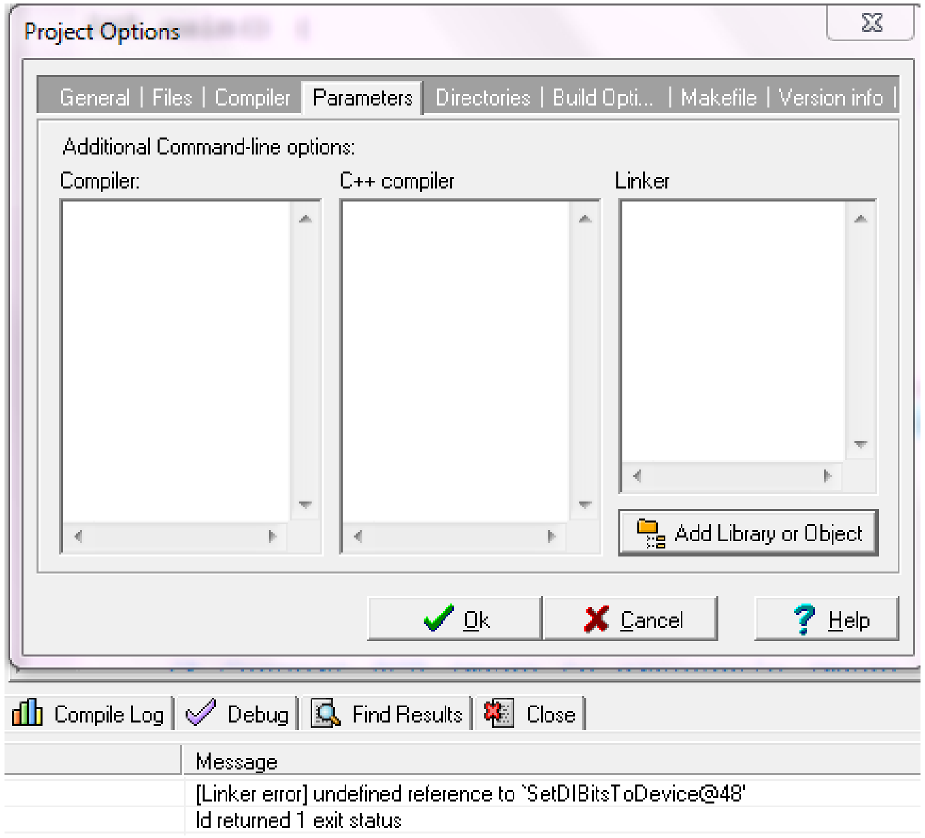
```
cimg::fopen() : File '../baby3.bmp' cannot be opened for reading.
```

Errors

- [*Build Error*] [main.o] *Error* 1
- This is a directory error experienced when you move the project .dev file in another directory from where it was created.

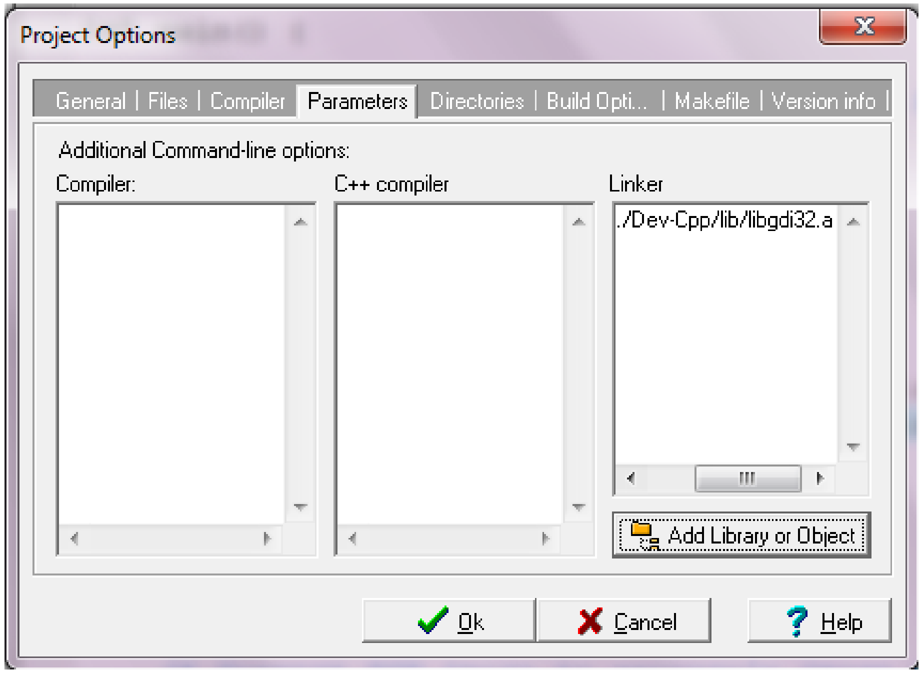
Errors

- Linker Error


Project Options
General | Files | Compiler | Parameters | Directories | Build Opti... | Makefile | Version info
Additional Command-line options:
Compiler:
C++ compiler
Linker
Add Library or Object
Ok
Cancel
Help
Compile Log
Debug
Find Results
Close

```
Message
[Linker error] undefined reference to `SetDIBitsToDevice@48'
ld returned 1 exit status
```

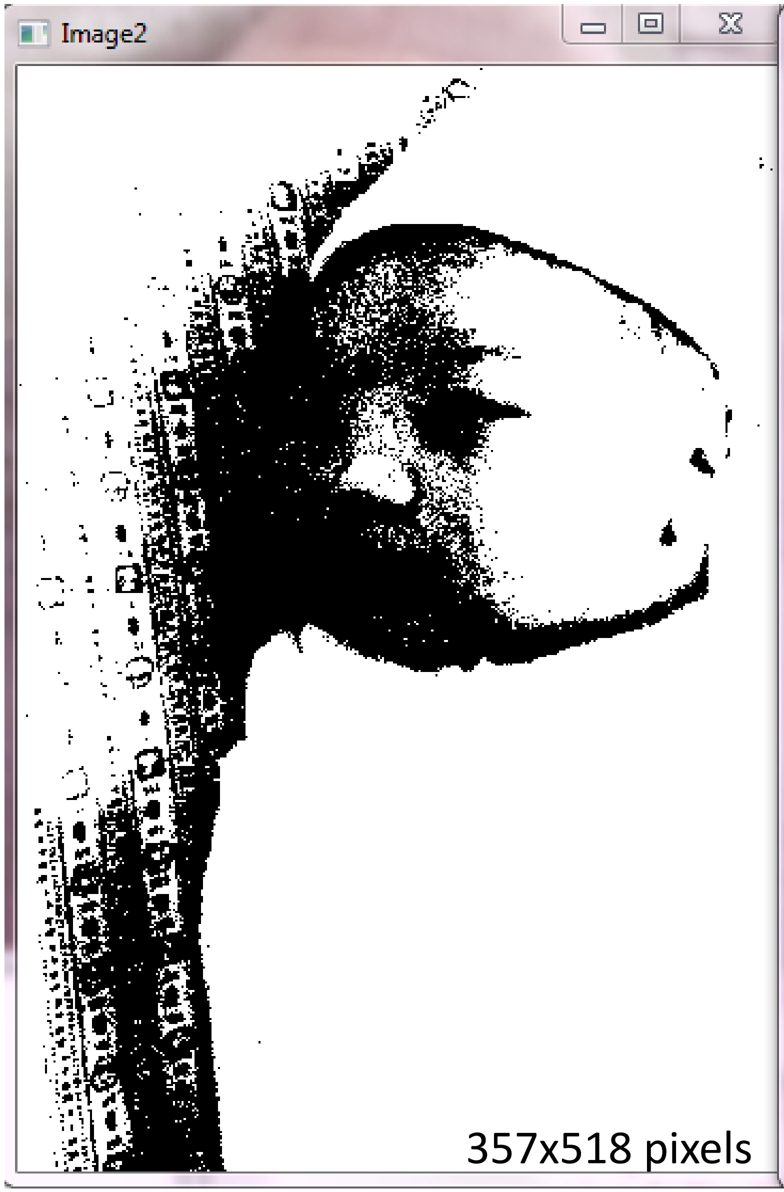
Solution:


Project Options
General | Files | Compiler | Parameters | Directories | Build Opti... | Makefile | Version info
Additional Command-line options:
Compiler:
C++ compiler
Linker
Add Library or Object
Ok
Cancel
Help

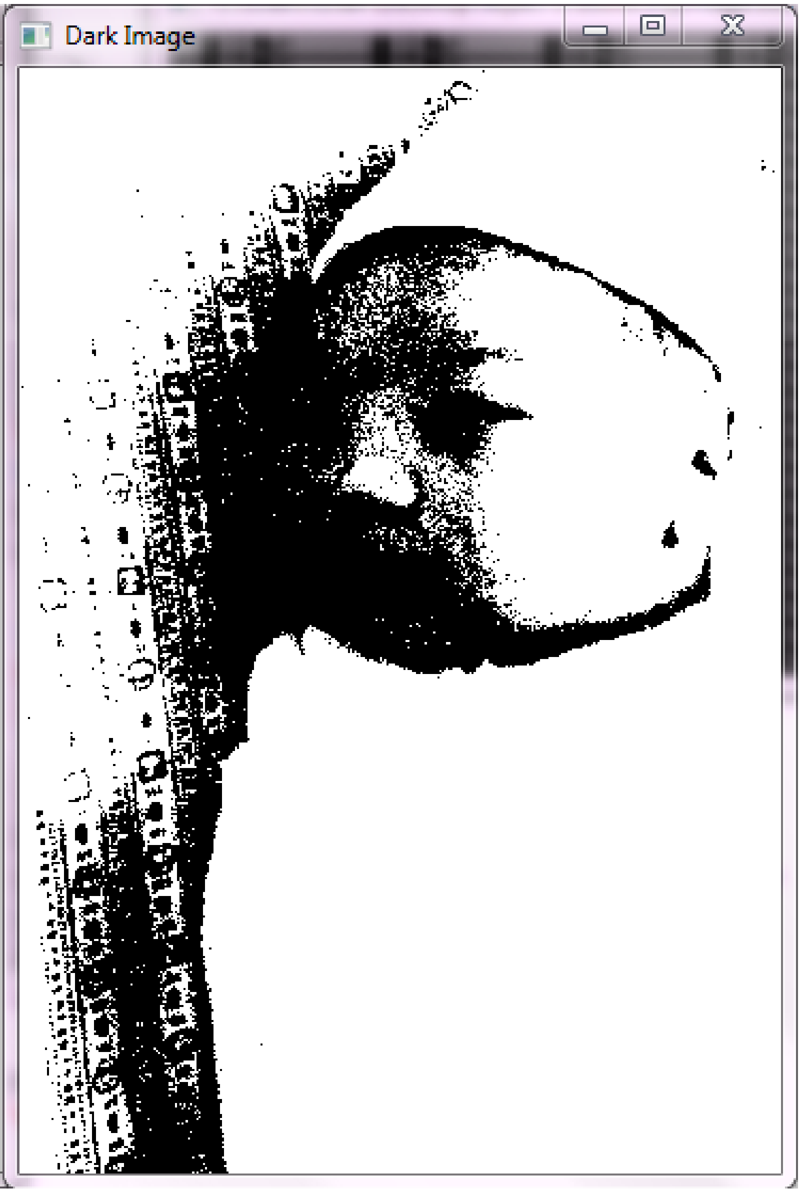
```
../Dev-Cpp/lib/libgdi32.a
```

Add Dev-Cpp/lib/libgdi32.a in parameters for the Linker In Project Options.

1 Bit Binary Bitmap Image (.bmp)



357x518 pixels



Source Code

```
#include <stdio.h>
#include "CImg.h"
using namespace cimg_library;

int main() {
CImg<double> image2("img/baby.bmp");
   CImgDisplay main_disp2(image2,"Image2",0);
   CImg<double> darkimage2(image2.dimx(),image2.dimy(),1,3,0);
   for (int i=0;i<image2.dimx();i++)
      for (int j=0;j<image2.dimy();j++)
         for (int k=0;k<3;k++)
            darkimage2(i,j,0,k)=image2(i,j,0,k);
   CImgDisplay dark_disp2(darkimage2,"Dark Image",0);
                                                                        //Display dark image

   while (!main_disp.is_closed) // Wait while the image is not closed
   main_disp.wait();
 return 0;
}
```

References

- The CImg 1.2.5 Library Reference Manual, Generated by Doxygen 1.5.3, 2007.
- Vicknesh Selvam, "An Introduction to the CImg Library“ A project conducted with the aid of Professor Todd Wittman.
- http://cimg.sourceforge.net/reference/