**Software Maintainability**

# **Part 1: Tool Demonstration**

***Tool Demonstration***

We analyze a problem that produces complex code due to its domain for this tool demonstration. It is the code for a SER334 homework assignment on Parallel Image Filtering. We removed all code related to reading the image and placed it in an external library, analyzing only the code about Image Filtering. We want to find out how maintainable this code is, and to do that; we used the reports M2 – Complexity Metrics, Cyclomatic, Interface, Total and *M2 – Total Quality Profile*. We will be analyzing the entire report, but the critical summary is in the following:

**~~ Project Functional Analysis ~~**

**Total Functions .......: 17 Total Physical Lines ..: 753**

**Total LOC .............: 691 Total Function Pts LOC : 5.4**

**Total eLOC ............: 563 Total Function Pts eLOC: 4.4**

**Total lLOC.............: 437 Total Function Pts lLOC: 3.4**

**Total Cyclomatic Comp. : 181 Total Interface Comp. .: 75**

**Total Parameters ......: 45 Total Return Points ...: 30**

**Total Comment Lines ...: 27 Total Blank Lines .....: 47**

The full report is included in the [Appendix](#_Appendix).

***Metrics Interpretation***

In addition to Complexity Metrics, we will be looking at the Project Quality Profile:

**~~ Project Quality Profile ~~**

**Type Count Percent Quality Notice**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1 38 11.52 Physical line length > 80 characters**

**5 1 0.30 Assignment "=" within "while" statement**

**7 8 2.42 Pre-increment operator "++" identified**

**14 1 0.30 "case" conditions do not equal "break"**

**17 14 4.24 Function comment content less than 10.0%**

**20 2 0.61 File comment content < 10.0%**

**22 12 3.64 if, else, for or while not bound by scope**

**23 10 3.03 "?" ternary operator identified**

**26 12 3.64 void \* generic type identified**

**27 5 1.52 Number of function return points > 1**

**28 8 2.42 Cyclomatic complexity > 10**

**29 1 0.30 Number of function parameters > 6**

**30 11 3.33 TAB character has been identified**

**43 1 0.30 Keyword "continue" has been identified**

**44 3 0.91 Keyword "break" identified outside a "switch" structure**

**46 12 3.64 Function/Class Blank Line content less < 10.0%**

**47 2 0.61 File Blank Line content < 10.0%**

**49 2 0.61 Function appears to have null or blank parameters**

**50 107 32.42 Variable assignment to a literal number**

**51 16 4.85 No comment preceding a function block**

**53 4 1.21 No comment preceding a struct block**

**55 8 2.42 Scope level exceeds the defined limit of 5**

**102 16 4.85 Dynamic memory using malloc is not initialized**

**105 3 0.91 A symbolic constant using #define**

**109 30 9.09 Double pointer indirection identified**

**125 3 0.91 A data member in the header file is not of the form m\_\***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The report shows that the Code is not of high quality. The first indication is the Average LOC, 40.65, and the Average eLOC, 33.12. Our Project Quality profile can retrieve some indications of where to focus. The highest Quality Notice is **Variable assignment to a literal number**, but since we are using C, we can safely assume that most of those come from lines like int i = 0, where i is an index. The more critical measures come from the Physical Line Length being more than 80 characters, and we will see why in a moment. We also noticed a high amount of double pointer indirection, and **dynamic memory using malloc is not initialized**. This application does not contain memory leaks, but it is essential to pay attention to that.

The most crucial metric from the report is Cyclomatic Complexity, and there's a high correlation between the length above 80 characters and the functions with high complexity. For the purpose of this report, we will focus on the functions compute\_holes, cheese\_filter, make\_slices, and colorShiftPixels.

***Metrics Report and Source Code***

Please refer to the full report included in the [Appendix](#_Appendix).

1. **int** \*\* compute\_holes(**struct** Pixel \*\* pArr, **int** width, **int** height) {
2. **int** radius = 0;
3. **int** all\_holes = 0;
4. **int** i = 0;
5. **int** j = 0;
6. **int** k = 0;
7. **srand**(**time**(NULL));
8. **if** (width < height) {
9. radius = (**int**)(width \* 0.08);
10. all\_holes = (**int**)(width \* 0.08);
11. } **else** {
12. radius = (**int**)(height \* 0.08);
13. all\_holes = (**int**)(height \* 0.08);
14. }
15. **if** (all\_holes == 0) {
16. **return** NULL;
17. }
18. **int** normal\_holes = all\_holes / 2;
19. **int** small\_holes = (**int**)(all\_holes - normal\_holes) \* 0.7;
20. **int** large\_holes = (all\_holes - normal\_holes - small\_holes);
21. all\_holes = normal\_holes + small\_holes + large\_holes;
23. // is not an exact science, but it works
24. **int** s\_radius = (**int**)radius \* 0.7;
25. **int** l\_radius = (**int**)radius \* 1.7;
26. **int** m\_radius = radius;
28. **if** (DEBUG) {
29. **printf**("normal\_holes: %d\n", normal\_holes);
30. **printf**("small\_holes: %d\n", small\_holes);
31. **printf**("large\_holes: %d\n", large\_holes);
32. **printf**("small radius: %d\n", s\_radius);
33. **printf**("medium radius: %d\n", m\_radius);
34. **printf**("large radius: %d\n", l\_radius);
35. }
37. **int** \*\* center = (**int** \*\* ) **malloc**(**sizeof**(**int** \* ) \* all\_holes);
38. **for** (i = 0; i != all\_holes; ++i) {
39. center[i] = (**int** \* ) **malloc**(**sizeof**(**int**) \* 2);
40. }
42. **for** (i = 0; i != all\_holes; ++i) {
43. **int** chunk\_width;
44. **int** chunk\_height;
45. **int** wMax = 0;
46. **int** hMax = 0;
47. **int** wMin = 0;
48. **int** hMin = 0;
49. **char** type[2] = {'\0', '\0'};
50. **if** (i < normal\_holes) {
51. **if** (!normal\_holes)
52. **return** NULL;
53. chunk\_width = width / normal\_holes;
54. chunk\_height = height / normal\_holes;
55. wMax = i \* chunk\_width;
56. hMax = i \* chunk\_height;
57. wMin = (i \* chunk\_width) + m\_radius;
58. hMin = (i \* chunk\_height) + m\_radius;
59. **if** (DEBUG)
60. type[0] = 'm';
61. } **else** **if** (i < normal\_holes + small\_holes) {
62. **if** (!small\_holes)
63. **break**;
64. chunk\_width = width / small\_holes;
65. chunk\_height = height / small\_holes;
66. wMax = i \* chunk\_width;
67. hMax = i \* chunk\_height;
68. wMin = (i \* chunk\_width) + s\_radius;
69. hMin = (i \* chunk\_height) + s\_radius;
70. **if** (DEBUG)
71. type[0] = 's';
72. } **else** {
73. **if** (!large\_holes)
74. **break**;
75. chunk\_width = width / large\_holes;
76. chunk\_height = height / large\_holes;
77. wMin = (i \* chunk\_width) + l\_radius;
78. hMin = (i \* chunk\_height) + l\_radius;
79. **if** (DEBUG)
80. type[0] = 'l';
81. }
82. **int** nRandonNumberWidth = **rand**();
83. **int** nRandonNumberHeight = **rand**();
84. **if** (**rand**() % 1) {
85. nRandonNumberWidth = **rand**()%((wMax+1)-wMin) + wMin;
86. } **else** {
87. nRandonNumberHeight = **rand**()%((hMax+1)-hMin) + hMin;
88. }
89. center[i][0] = nRandonNumberWidth % width;
90. center[i][1] = nRandonNumberHeight % height;
91. **if** (DEBUG) {
92. **printf**("%s %d - %d %d\n", type, i , center[i][0], center[i][1]);
93. }
94. }

97. **int** \*\* circles = (**int** \*\* ) **malloc**(**sizeof**(**int** \* ) \* height);
98. **for** (i = 0; i != height; i++) {
99. circles[i] = (**int** \* ) **malloc**(**sizeof**(**int**) \* width);
100. **for** (**int** j = 0; j < width; j++) {
101. circles[i][j] = 0;
102. }
103. }
105. **for** (k = 0; k != all\_holes; k++) {
106. **for** (i = 0; i != height; ++i) {
107. **for** (j = 0; j != width; ++j) {
108. **int** x1 = center[k][0];
109. **int** x2 = j;
110. **int** y1 = center[k][1];
111. **int** y2 = i;
112. **int** dist = **sqrt**(**pow**(x2 - x1, 2) + **pow**(y2 - y1, 2));
113. **if** (k < normal\_holes) {
114. **if** (dist < m\_radius) {
115. circles[i][j] = 1;
116. }
117. } **else** **if** (k < normal\_holes + small\_holes) {
118. **if** (dist < s\_radius) {
119. circles[i][j] = 1;
120. }
121. } **else** {
122. **if** (dist < l\_radius) {
123. circles[i][j] = 1;
124. }
125. }
126. }
127. }
128. }
130. **for** (i = 0; i < radius; i++) {
131. **free**(center[i]);
132. }
133. **free**(center);
135. **return** circles;
136. }
138. **void** cheese\_filter(**struct** Pixel \*\* pArr, **int** max\_threads, **int** width, **int** height) {
139. **printf**("Applying Cheese filter...\n");
140. **int** thread\_cap = max\_threads < width ? max\_threads : width;
141. **struct** Pixel\_Slice \*\* slices = make\_slices(pArr, thread\_cap, 0, width, height);
142. **int** \*\* circles = compute\_holes(pArr, width, height);
143. **if** (circles == NULL) {
144. **return**;
145. }

148. **int** i = 0;
149. **int** j = 0;
150. **int** k = 0;
151. **int** z = 0;
152. **for** (i = 0; i < max\_threads; i++) {
153. **int** \*\* slice\_circles = (**int** \*\* ) **malloc**(**sizeof**(**int**\*) \* height);
154. **for** (j = 0; j != height; j++) {
155. slice\_circles[j] = (**int** \* ) **malloc**(**sizeof**(**int**) \* slices[i]->width);
156. **for** (k = 0; k < slices[i]->width; k++) {
157. slices[i]->circles = slice\_circles;
158. }
159. }
160. }
161. **for** (i = 0; i < max\_threads; i++) {
162. **for** (j = 0; j != height; j++) {
163. z = 0;
164. **int** start = slices[i]->offset;
165. **int** end = slices[i]->offset + slices[i]->width;
166. **for** (k = start; k != end; k++) {
167. slices[i]->circles[j][z] = circles[j][k];
168. z++;
169. }
170. }
171. }
173. apply\_threading(pArr, slices, thread\_cap, width, height, \_cheese\_filter);
175. **if** (DEBUG)
176. **printf**("Deleting slices\n");
178. **for** (i = 0; i < height; i++) {
179. **free**(circles[i]);
180. }
181. **free**(circles);
182. circles = NULL;
184. **for** (i = 0; i < max\_threads; i++) {
185. **for** (j = 0; j != height; j++) {
186. **free**(slices[i]->circles[j]);
187. **free**(slices[i]->slice[j]);
188. }
189. **free**(slices[i]->slice);
190. **free**(slices[i]->circles);
191. **free**(slices[i]);
192. slices[i] = NULL;
193. }
194. **free**(slices);
195. slices = NULL;
196. }
198. **struct** Pixel\_Slice \*\* make\_slices(**struct** Pixel \*\* pArr, **int** threads, **int** with\_padding, **int** width, **int** height) {
199. **struct** Pixel\_Slice \*\* slices = **malloc**(**sizeof**(**struct** Pixel\_Slice) \* threads);
200. **int** i = 0;
201. **int** slice\_width = (**int**) width / threads;
202. **int** slice\_residue = width % threads;
203. **int** row = 0;
204. **int** col = 0;
205. // Build slices
206. **for** (i = 0; i < threads; i++) {
207. **struct** Pixel \*\* pixels = (**struct** Pixel \*\* ) **malloc**(**sizeof**(**struct** Pixel \* ) \* height);
208. **for** (row = 0; row != height; row++) {
209. **if** (i == threads - 1) {
210. **int** padding = with\_padding ? 1 : 0;
211. pixels[row] = (**struct** Pixel \* ) **malloc**(**sizeof**(**struct** Pixel) \* (slice\_width + slice\_residue + padding));
212. } **else** **if** (i == 0) {
213. **int** padding = with\_padding ? 1 : 0;
214. pixels[row] = (**struct** Pixel \* ) **malloc**(**sizeof**(**struct** Pixel) \* (slice\_width + padding));
215. } **else** {
216. **int** padding = with\_padding ? 2 : 0;
217. pixels[row] = (**struct** Pixel \* ) **malloc**(**sizeof**(**struct** Pixel) \* (slice\_width + padding));
218. }
219. }
220. slices[i] = **malloc**(**sizeof**(**struct** Pixel\_Slice));
221. slices[i]->slice = pixels;
222. slices[i]->height = height;
223. }
224. // Fill and configure slices
225. **for** (**int** i = 0; i < threads; i++) {
226. // first slice
227. **if** (i == 0) {
228. **int** padding = with\_padding ? 1 : 0;
229. **for** (**int** j = 0; j < height; j++) {
230. **for** (**int** k = 0; k < slice\_width + padding; k++) {
231. **if** (DEBUG && j == 0) {
232. **printf**("%d ", k);
233. }
234. slices[i]->slice[j][k].red = pArr[j][k].red;
235. slices[i]->slice[j][k].green = pArr[j][k].green;
236. slices[i]->slice[j][k].blue = pArr[j][k].blue;
237. }
238. **if** (DEBUG && j == 0) {
239. **printf**("\n");
240. }
241. }
242. slices[i]->offset = 0;
243. slices[i]->start = 0;
244. slices[i]->end = slice\_width + padding;
245. slices[i]->width = slice\_width + padding;
246. }
247. // last slice
248. **else** **if** (i == threads - 1) {
249. **int** padding = with\_padding ? 1 : 0;
250. **for** (**int** j = 0; j < height; j++) {
251. **for** (**int** k = 0; k < slice\_width + padding + slice\_residue; k++) {
252. **if** (DEBUG && j == 0) {
253. **printf**("%d ", k + i \* slice\_width - 1);
254. }
255. slices[i]->slice[j][k].red = pArr[j][k + i \* slice\_width - padding].red;
256. slices[i]->slice[j][k].green = pArr[j][k + i \* slice\_width - padding].green;
257. slices[i]->slice[j][k].blue = pArr[j][k + i \* slice\_width - padding].blue;
258. }
259. **if** (DEBUG && j == 0) {
260. **printf**("\n");
261. }
262. }
263. slices[i]->offset = i \* slice\_width - padding;
264. slices[i]->start = padding;
265. slices[i]->end = slice\_width + slice\_residue + padding;
266. slices[i]->width = slice\_width + slice\_residue + padding;
267. }
268. // mid slice
269. **else** {
270. **int** padding = with\_padding ? 2 : 0;
271. **int** start\_padding = with\_padding ? 1 : 0;
272. **for** (**int** j = 0; j < height; j++) {
273. **for** (**int** k = 0; k < slice\_width + padding; k++) {
274. **if** (DEBUG && j == 0) {
275. **printf**("%d ", k + i \* slice\_width - 1);
276. }
277. slices[i]->slice[j][k].red = pArr[j][k + i \* slice\_width - start\_padding].red;
278. slices[i]->slice[j][k].green = pArr[j][k + i \* slice\_width - start\_padding].green;
279. slices[i]->slice[j][k].blue = pArr[j][k + i \* slice\_width - start\_padding].blue;
280. }
281. **if** (DEBUG && j == 0) {
282. **printf**("\n");
283. }
284. }
285. slices[i]->offset = i \* slice\_width - start\_padding;
286. slices[i]->start = start\_padding;
287. slices[i]->end = slice\_width + padding;
288. slices[i]->width = slice\_width + padding;
289. }
290. }
292. **if** (DEBUG) {
293. **for** (i = 0; i < threads; i++) {
294. **printf**("%d: offset: %d, start: %d, end: %d, width: %d\n", i, slices[i]->offset, slices[i]->start, slices[i]->end, slices[i]->width);
295. }
296. }
298. **return** slices;
299. }
301. **void** colorShiftPixels(**struct** Pixel\*\* pArr, **int** width, **int** height, **int** rShift, **int** gShift, **int** bShift, **int** leave\_black) {
302. **for** (**int** i = 0; i != height; i++) {
303. **for** (**int** j = 0; j != width; j++) {
304. **if** (leave\_black) {
305. **if** (pArr[i][j].red == 0 && pArr[i][j].green == 0 && pArr[i][j].blue == 0) {
306. **continue**;
307. }
308. }
309. **if** (rShift != 0) {
310. **if** (pArr[i][j].red + rShift > 255) {
311. pArr[i][j].red = 255;
312. } **else** **if** (pArr[i][j].red + rShift < 0) {
313. pArr[i][j].red = 0;
314. } **else** {
315. pArr[i][j].red += rShift;
316. }
317. }
318. **if** (gShift != 0) {
319. **if** (pArr[i][j].green + gShift > 255) {
320. pArr[i][j].green = 255;
321. } **else** **if** (pArr[i][j].green + gShift < 0) {
322. pArr[i][j].red = 0;
323. } **else** {
324. pArr[i][j].green += gShift;
325. }
326. }
327. **if** (bShift != 0) {
328. **if** (pArr[i][j].blue + bShift > 255) {
329. pArr[i][j].blue = 255;
330. } **else** **if** (pArr[i][j].blue + bShift < 0) {
331. pArr[i][j].blue = 0;
332. } **else** {
333. pArr[i][j].blue += bShift;
334. }
335. }
336. }
337. }
338. }

# **Part 2: Software Maintainability Measure**

***Maintainability Measure Identification***

Software Maintainability has undergone many changes over the years. In the book Cyclomatic Complexity Metrics Revisited [1], the authors quote another paper citing Software maintenance as "work done on a software system after it becomes operational." They also mention three types of tasks:

1. Corrective - Correct discovered problem (e.g., Bug fixing)
2. Adaptive - Adapt to a changed or changing environment (e.g., new laws)
3. Perfective - Improve Performance or maintainability (e.g., new database driver)

We also found a fourth one recently shown in the talk "Building Maintainable Software" [2] and in the book Software Architect's Handbook [3]:

1. Preventive - Detect and correct latent faults (e.g., patch vulnerabilities)

In the book Cyclomatic Complexity Metrics Revisited [1], the authors also find that increasing complexity reduces maintenance productivity regardless of the measure. In the book Complete Guide to Test Automation [4], Arnon Axelrod explains how complexity comes with a cost and provides two points to keep a low maintenance cost:

1. *Make the cost of running the ever-growing regression test suite negligible.*
2. *Keeping the code very easy to maintain.*

We can find books that tackle the problem of keeping maintenance costs flat. Building Maintainable Software [5] provides a lot of critical points on how to do it with examples. Some of those are unrelated to the RSM metrics, such as automated tests. However, testing is a big part of software maintainability and is too big of a subject. In a newer book, Software Architecture Metrics, the authors describe the following metrics to address maintainability:

1. Average Component Dependency (ACD)
   * Average Component Dependency (ACD) is a software metric used to assess the coupling within a codebase. It measures, on average, how many other components a single component depends on.
2. Cyclicity and relative cyclicity
   * Cyclomatic complexity focuses on the decision points within a single function or method, measuring its inherent complexity based on loops, conditionals, and exits.
3. Structural Debt index
   * In summary, it involves analyzing the dependency relationships between different parts of the codebase. Factors like the number of dependencies that need to be broken to achieve a clean, modular design and the complexity of those dependencies likely contribute to the SDI score. The complete implementation is private.

In addition to these metrics, the authors go over other potentially valuable metrics to use as decision points:

1. Lines of Code
2. Cyclomatic Complexity
3. Indentation Debt
4. Change Frequency
5. Code Churn
6. Number of Authors
7. Component Tank
8. Lack of Cohesion of Methods

***Relation to RSM Tool Metrics***

Returning to the types of Maintenance tasks and the cost of maintenance, we see that Lines of Code, Cyclomatic Complexity, and Quality Profiles are all very related. On Building Maintainable Software [5], we can conclude that if we did their advice of writing simple units of code that are reusable, with simple interfaces and components loosely coupled but with high cohesion and a small codebase, we would obtain a small number of Lines of Codes per function, low Cyclomatic Complexity, as well as a small number of issues in the Quality Profiles.

We calculate Cyclomatic complexity by considering the number of decision points within a code block. This calculation includes loops, conditional statements (if/else), and exits. It counts the number of independent paths a program could take through the code, with a higher number indicating potentially harder-to-understand and maintain logic.

The metrics provide guidance, and it's very valid. The Cyclomatic Complexity and the Quality Profile can help refactor the code into something more accessible for reuse. But as with everything, there's the question of why this code already performs as optimally as possible in a multithreaded fashion. We know somebody will only reuse it if it is built for a specific image filter.

# **References**

|  |  |
| --- | --- |
| [1] | G. K. Gill, Cyclomatic complexity metrics revisited: An empirical study of software development and maintenance (classic reprint), London, England: Forgotten Books, 2022. |
| [2] | Ž. Obrenović, *Building Maintainable Software,* O'Reilly Media, Inc., 2016. |
| [3] | J. Ingeno, Software Architect's Handbook, Birmingham, England: Packt Publishing, 2018. |
| [4] | A. Axelrod, Complete guide to test automation, 1 ed., Berlin, Germany: APress, 2018. |
| [5] | J. Visser, S. Rigal, G. Wijnholds, P. van Eck and R. van der Leek, Building Maintainable Software, C# Edition, Sebastopol, CA: O'Reilly Media, 2016. |
| [6] | C. Ciceri, D. Farley, N. Ford, A. Harmel-Law, M. Keeling and C. Lilienthal, Software architecture metrics, Sebastopol, CA: O'Reilly Media, 2022. |

# **Appendix**

***Resource Standard Metrics Report***

***M2- Complexity Metrics, Cyclomatic, Interface, Total***

**Resource Standard Metrics™ for C, C++, C# and Java**

Version 7.75 - [mSquaredTechnologies.com](http://mSquaredTechnologies.com)

**License Type**: **Shareware Evaluation License**

**Licensed To** : **Shareware End User - Distribute Freely**

**License No**. : **SW1380 License Date**: Dec 05, 1998

**Build Date** : **Sep 2 2009 Run Date**: Apr 30, 2024

**©1996-2009 M Squared Technologies LLC™**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**License File: C:\Program Files (x86)\MSquared\M2 RSM\rsm.lic**

**Config. File: C:\Program Files (x86)\MSquared\M2 RSM\rsm.cfg**

**Command Line: -H -OC:\Users\claud\Documents\MastersLocal\CSE566\RSM\outp**

**ut.htm -c -fd -FC:\Users\claud\M2 RSM Wizard\input\rsm\_fil**

**e\_list.lst**

**~~ Function Metrics ~~**

**~~ Complexity Detail Analysis ~~**

**File:** [**C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.h**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.h)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**~~ Total File Summary ~~**

**LOC 6 eLOC 6 lLOC 2 Comment 24 Lines 35**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**End of File:** [**C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.h**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.h)

**File:** [**C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.h**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.h)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**------------------------------------------------------------------------**

**~~ Total File Summary ~~**

**LOC 9 eLOC 8 lLOC 5 Comment 9 Lines 21**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**End of File:** [**C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.h**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.h)

**File:** [**C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: blur\_filter**

**Parameters: (struct Pixel \*\* pArr, int max\_threads, int width, int heigh**

**t)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 2**

**Conditional if / else if: 1**

**Inlined if-else ( ? : ) : 1**

**Complexity Param 4 Return 1 Cyclo Vg 5 Total 10**

**LOC 20 eLOC 17 lLOC 15 Comment 0 Lines 22**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: cheese\_filter**

**Parameters: (struct Pixel \*\* pArr, int max\_threads, int width, int heigh**

**t)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 9**

**Conditional if / else if: 2**

**Inlined if-else ( ? : ) : 1**

**Complexity Param 4 Return 1 Cyclo Vg 13 Total 18**

**LOC 53 eLOC 42 lLOC 39 Comment 0 Lines 59**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: compute\_holes**

**Parameters: (struct Pixel \*\* pArr, int width, int height)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 8**

**Conditional if / else if: 18**

**Complexity Param 3 Return 3 Cyclo Vg 27 Total 33**

**LOC 126 eLOC 107 lLOC 84 Comment 1 Lines 136**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: apply\_threading**

**Parameters: (struct Pixel \*\* pArr, struct Pixel\_Slice \*\* slices, int max**

**\_threads, int width, int height, void \* (\* filter\_func)(void**

**\*)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 2**

**Conditional if / else if: 2**

**Complexity Param 6 Return 1 Cyclo Vg 5 Total 12**

**LOC 19 eLOC 15 lLOC 12 Comment 0 Lines 21**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: make\_slices**

**Parameters: (struct Pixel \*\* pArr, int threads, int with\_padding, int wi**

**dth, int height)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 10**

**Conditional if / else if: 11**

**Inlined if-else ( ? : ) : 7**

**Logical and ( && ) : 6**

**Complexity Param 5 Return 1 Cyclo Vg 35 Total 41**

**LOC 95 eLOC 73 lLOC 59 Comment 5 Lines 102**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: \_copy\_slices\_into\_image**

**Parameters: (struct Pixel \*\* pArr, struct Pixel\_Slice \*\* slices, int thr**

**eads, int width, int height)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 3**

**Conditional if / else if: 6**

**Inlined if-else ( ? : ) : 1**

**Logical or ( || ) : 1**

**Logical and ( && ) : 4**

**Complexity Param 5 Return 1 Cyclo Vg 16 Total 22**

**LOC 34 eLOC 24 lLOC 17 Comment 0 Lines 36**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: \_box\_blur\_filter**

**Parameters: (void \* arg)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 2**

**Conditional if / else if: 9**

**Logical and ( && ) : 4**

**Complexity Param 1 Return 1 Cyclo Vg 16 Total 18**

**LOC 99 eLOC 87 lLOC 77 Comment 8 Lines 112**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: \_cheese\_filter**

**Parameters: (void \* arg)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 2**

**Conditional if / else if: 1**

**Complexity Param 1 Return 1 Cyclo Vg 4 Total 6**

**LOC 15 eLOC 11 lLOC 9 Comment 0 Lines 17**

**------------------------------------------------------------------------**

**~~ Total File Summary ~~**

**LOC 483 eLOC 397 lLOC 326 Comment 22 Lines 547**

**------------------------------------------------------------------------**

**~~ File Functional Summary ~~**

**File Function Count....: 8**

**Total Function LOC.....: 461 Total Function Pts LOC : 3.8**

**Total Function eLOC....: 376 Total Function Pts eLOC: 3.1**

**Total Function lLOC....: 312 Total Function Pts lLOC: 2.5**

**Total Function Params .: 29 Total Function Return .: 10**

**Total Cyclo Complexity : 121 Total Function Complex.: 160**

**------ ----- ----- ------ ------ -----**

**Max Function LOC ......: 126 Average Function LOC ..: 57.63**

**Max Function eLOC .....: 107 Average Function eLOC .: 47.00**

**Max Function lLOC .....: 84 Average Function lLOC .: 39.00**

**------ ----- ----- ------ ------ -----**

**Max Function Parameters: 6 Avg Function Parameters: 3.63**

**Max Function Returns ..: 3 Avg Function Returns ..: 1.25**

**Max Interface Complex. : 9 Avg Interface Complex. : 4.88**

**Max Cyclomatic Complex.: 35 Avg Cyclomatic Complex.: 15.13**

**Max Total Complexity ..: 41 Avg Total Complexity ..: 20.00**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**End of File:** [**C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)

**File:** [**C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.c**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.c)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.c)**: colorShiftPixels**

**Parameters: (struct Pixel\*\* pArr, int width, int height, int rShift, int**

**gShift, int bShift, int leave\_black)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 2**

**Conditional if / else if: 11**

**Logical and ( && ) : 2**

**Complexity Param 7 Return 1 Cyclo Vg 16 Total 24**

**LOC 38 eLOC 27 lLOC 12 Comment 0 Lines 38**

**------------------------------------------------------------------------**

**~~ Total File Summary ~~**

**LOC 39 eLOC 28 lLOC 12 Comment 0 Lines 40**

**------------------------------------------------------------------------**

**~~ File Functional Summary ~~**

**File Function Count....: 1**

**Total Function LOC.....: 38 Total Function Pts LOC : 0.3**

**Total Function eLOC....: 27 Total Function Pts eLOC: 0.2**

**Total Function lLOC....: 12 Total Function Pts lLOC: 0.1**

**Total Function Params .: 7 Total Function Return .: 1**

**Total Cyclo Complexity : 16 Total Function Complex.: 24**

**------ ----- ----- ------ ------ -----**

**Max Function LOC ......: 38 Average Function LOC ..: 38.00**

**Max Function eLOC .....: 27 Average Function eLOC .: 27.00**

**Max Function lLOC .....: 12 Average Function lLOC .: 12.00**

**------ ----- ----- ------ ------ -----**

**Max Function Parameters: 7 Avg Function Parameters: 7.00**

**Max Function Returns ..: 1 Avg Function Returns ..: 1.00**

**Max Interface Complex. : 8 Avg Interface Complex. : 8.00**

**Max Cyclomatic Complex.: 16 Avg Cyclomatic Complex.: 16.00**

**Max Total Complexity ..: 24 Avg Total Complexity ..: 24.00**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**End of File:** [**C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.c**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.c)

**File:** [**C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: main**

**Parameters: (int argc, char \*\*argv)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 2**

**Selection case : 2**

**Conditional if / else if: 2**

**Logical or ( || ) : 3**

**Logical and ( && ) : 1**

**Complexity Param 2 Return 4 Cyclo Vg 11 Total 17**

**LOC 60 eLOC 53 lLOC 42 Comment 13 Lines 70**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: cli\_options\_create**

**Parameters: ()**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Conditional if / else if: 1**

**Complexity Param 0 Return 2 Cyclo Vg 2 Total 4**

**LOC 11 eLOC 9 lLOC 7 Comment 0 Lines 11**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: cli\_options\_destroy**

**Parameters: (Cli\_Options \*\* options)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 4 eLOC 3 lLOC 2 Comment 0 Lines 4**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: help**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 12 eLOC 11 lLOC 10 Comment 0 Lines 12**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: get\_cli\_options**

**Parameters: (int argc, char \*\*argv, Cli\_Options \*cli\_options)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops while / do : 1**

**Selection case : 5**

**Conditional if / else if: 8**

**Logical and ( && ) : 1**

**Complexity Param 3 Return 7 Cyclo Vg 16 Total 26**

**LOC 54 eLOC 46 lLOC 28 Comment 0 Lines 58**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: display\_user\_selections**

**Parameters: (ImageProcessorOptions\* options)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Conditional if / else if: 2**

**Complexity Param 1 Return 1 Cyclo Vg 3 Total 5**

**LOC 9 eLOC 6 lLOC 3 Comment 0 Lines 9**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: image\_processor\_create**

**Parameters: (Cli\_Options\* cli\_options)**

**Cyclomatic Complexity Vg Detail**

**Function Base : 1**

**Loops for / foreach : 1**

**Conditional if / else if: 7**

**Complexity Param 1 Return 2 Cyclo Vg 9 Total 12**

**LOC 38 eLOC 29 lLOC 19 Comment 0 Lines 42**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: image\_processor\_options\_destroy**

**Parameters: (ImageProcessorOptions \*\* options)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 4 eLOC 3 lLOC 2 Comment 0 Lines 4**

**------------------------------------------------------------------------**

**~~ Total File Summary ~~**

**LOC 225 eLOC 191 lLOC 134 Comment 40 Lines 306**

**------------------------------------------------------------------------**

**~~ File Functional Summary ~~**

**File Function Count....: 8**

**Total Function LOC.....: 192 Total Function Pts LOC : 1.8**

**Total Function eLOC....: 160 Total Function Pts eLOC: 1.5**

**Total Function lLOC....: 113 Total Function Pts lLOC: 1.0**

**Total Function Params .: 9 Total Function Return .: 19**

**Total Cyclo Complexity : 44 Total Function Complex.: 72**

**------ ----- ----- ------ ------ -----**

**Max Function LOC ......: 60 Average Function LOC ..: 24.00**

**Max Function eLOC .....: 53 Average Function eLOC .: 20.00**

**Max Function lLOC .....: 42 Average Function lLOC .: 14.13**

**------ ----- ----- ------ ------ -----**

**Max Function Parameters: 3 Avg Function Parameters: 1.13**

**Max Function Returns ..: 7 Avg Function Returns ..: 2.38**

**Max Interface Complex. : 10 Avg Interface Complex. : 3.50**

**Max Cyclomatic Complex.: 16 Avg Cyclomatic Complex.: 5.50**

**Max Total Complexity ..: 26 Avg Total Complexity ..: 9.00**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**End of File: [C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilter](file:///C:\\Code\\asu-mcs\\SER334\\Module6\\HW6\\RodriguezRodriguezFilters.c" \t "_blank)**

**[s.c](file:///C:\\Code\\asu-mcs\\SER334\\Module6\\HW6\\RodriguezRodriguezFilters.c" \t "_blank)**

**------------------------------------------------------------------------**

**~~ Total Metrics For 5 Files ~~**

**------------------------------------------------------------------------**

**~~ Total Project Summary ~~**

**LOC 762 eLOC 630 lLOC 479 Comment 95 Lines 949**

**Average per File, metric/5 files**

**LOC 152 eLOC 126 lLOC 95 Comment 19 Lines 189**

**------------------------------------------------------------------------**

**~~ Project Functional Metrics ~~**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: blur\_filter**

**Parameters: (struct Pixel \*\* pArr, int max\_threads, int width, int heigh**

**t)**

**Complexity Param 4 Return 1 Cyclo Vg 5 Total 10**

**LOC 20 eLOC 17 lLOC 15 Comment 0 Lines 22**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: cheese\_filter**

**Parameters: (struct Pixel \*\* pArr, int max\_threads, int width, int heigh**

**t)**

**Complexity Param 4 Return 1 Cyclo Vg 13 Total 18**

**LOC 53 eLOC 42 lLOC 39 Comment 0 Lines 59**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: compute\_holes**

**Parameters: (struct Pixel \*\* pArr, int width, int height)**

**Complexity Param 3 Return 3 Cyclo Vg 27 Total 33**

**LOC 126 eLOC 107 lLOC 84 Comment 1 Lines 136**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: apply\_threading**

**Parameters: (struct Pixel \*\* pArr, struct Pixel\_Slice \*\* slices, int max**

**\_threads, int width, int height, void \* (\* filter\_func)(void**

**\*)**

**Complexity Param 6 Return 1 Cyclo Vg 5 Total 12**

**LOC 19 eLOC 15 lLOC 12 Comment 0 Lines 21**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: make\_slices**

**Parameters: (struct Pixel \*\* pArr, int threads, int with\_padding, int wi**

**dth, int height)**

**Complexity Param 5 Return 1 Cyclo Vg 35 Total 41**

**LOC 95 eLOC 73 lLOC 59 Comment 5 Lines 102**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: \_copy\_slices\_into\_image**

**Parameters: (struct Pixel \*\* pArr, struct Pixel\_Slice \*\* slices, int thr**

**eads, int width, int height)**

**Complexity Param 5 Return 1 Cyclo Vg 16 Total 22**

**LOC 34 eLOC 24 lLOC 17 Comment 0 Lines 36**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: \_box\_blur\_filter**

**Parameters: (void \* arg)**

**Complexity Param 1 Return 1 Cyclo Vg 16 Total 18**

**LOC 99 eLOC 87 lLOC 77 Comment 8 Lines 112**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\FilterProcessor.c)**: \_cheese\_filter**

**Parameters: (void \* arg)**

**Complexity Param 1 Return 1 Cyclo Vg 4 Total 6**

**LOC 15 eLOC 11 lLOC 9 Comment 0 Lines 17**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\PixelProcessor.c)**: colorShiftPixels**

**Parameters: (struct Pixel\*\* pArr, int width, int height, int rShift, int**

**gShift, int bShift, int leave\_black)**

**Complexity Param 7 Return 1 Cyclo Vg 16 Total 24**

**LOC 38 eLOC 27 lLOC 12 Comment 0 Lines 38**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: main**

**Parameters: (int argc, char \*\*argv)**

**Complexity Param 2 Return 4 Cyclo Vg 11 Total 17**

**LOC 60 eLOC 53 lLOC 42 Comment 13 Lines 70**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: cli\_options\_create**

**Parameters: ()**

**Complexity Param 0 Return 2 Cyclo Vg 2 Total 4**

**LOC 11 eLOC 9 lLOC 7 Comment 0 Lines 11**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: cli\_options\_destroy**

**Parameters: (Cli\_Options \*\* options)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 4 eLOC 3 lLOC 2 Comment 0 Lines 4**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: help**

**Parameters: ()**

**Complexity Param 0 Return 1 Cyclo Vg 1 Total 2**

**LOC 12 eLOC 11 lLOC 10 Comment 0 Lines 12**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: get\_cli\_options**

**Parameters: (int argc, char \*\*argv, Cli\_Options \*cli\_options)**

**Complexity Param 3 Return 7 Cyclo Vg 16 Total 26**

**LOC 54 eLOC 46 lLOC 28 Comment 0 Lines 58**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: display\_user\_selections**

**Parameters: (ImageProcessorOptions\* options)**

**Complexity Param 1 Return 1 Cyclo Vg 3 Total 5**

**LOC 9 eLOC 6 lLOC 3 Comment 0 Lines 9**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: image\_processor\_create**

**Parameters: (Cli\_Options\* cli\_options)**

**Complexity Param 1 Return 2 Cyclo Vg 9 Total 12**

**LOC 38 eLOC 29 lLOC 19 Comment 0 Lines 42**

[**Function**](file:///C:\Code\asu-mcs\SER334\Module6\HW6\RodriguezRodriguezFilters.c)**: image\_processor\_options\_destroy**

**Parameters: (ImageProcessorOptions \*\* options)**

**Complexity Param 1 Return 1 Cyclo Vg 1 Total 3**

**LOC 4 eLOC 3 lLOC 2 Comment 0 Lines 4**

**Total: Functions**

**LOC 691 eLOC 563 lLOC 437 InCmp 75 CycloCmp 181**

**Function Points FP(LOC) 5.4 FP(eLOC) 4.4 FP(lLOC) 3.4**

**------------------------------------------------------------------------**

**~~ Project Functional Analysis ~~**

**Total Functions .......: 17 Total Physical Lines ..: 753**

**Total LOC .............: 691 Total Function Pts LOC : 5.4**

**Total eLOC ............: 563 Total Function Pts eLOC: 4.4**

**Total lLOC.............: 437 Total Function Pts lLOC: 3.4**

**Total Cyclomatic Comp. : 181 Total Interface Comp. .: 75**

**Total Parameters ......: 45 Total Return Points ...: 30**

**Total Comment Lines ...: 27 Total Blank Lines .....: 47**

**------ ----- ----- ------ ------ -----**

**Avg Physical Lines ....: 44.29**

**Avg LOC ...............: 40.65 Avg eLOC ..............: 33.12**

**Avg lLOC ..............: 25.71 Avg Cyclomatic Comp. ..: 10.65**

**Avg Interface Comp. ...: 4.41 Avg Parameters ........: 2.65**

**Avg Return Points .....: 1.76 Avg Comment Lines .....: 1.59**

**------ ----- ----- ------ ------ -----**

**Max LOC ...............: 126**

**Max eLOC ..............: 107 Max lLOC ..............: 84**

**Max Cyclomatic Comp. ..: 35 Max Interface Comp. ...: 10**

**Max Parameters ........: 7 Max Return Points .....: 7**

**Max Comment Lines .....: 13 Max Total Lines .......: 136**

**------ ----- ----- ------ ------ -----**

**Min LOC ...............: 4**

**Min eLOC ..............: 3 Min lLOC ..............: 2**

**Min Cyclomatic Comp. ..: 1 Min Interface Comp. ...: 1**

**Min Parameters ........: 0 Min Return Points .....: 1**

**Min Comment Lines .....: 0 Min Total Lines .......: 4**

**------------------------------------------------------------------------**

**~~ File Summary ~~**

**C Source Files *\*.c* ....: 3 C/C++ Include Files *\*.h*: 2**

**C++ Source Files *\*.c\** .: 0 C++ Include Files *\*.h\** : 0**

**C# Source Files *\*.cs* ..: 0 Java Source File *\*.jav\**: 0**

**Other Source Files ....: 0**

**Total File Count ......: 5**

**Shareware evaluation licenses process only 20 files.**

**Paid licenses enable processing for an unlimited number of files.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***M2 – Total Quality Profile***

**Resource Standard Metrics™ for C, C++, C# and Java**

Version 7.75 - [mSquaredTechnologies.com](http://mSquaredTechnologies.com)

**License Type**: **Shareware Evaluation License**

**Licensed To** : **Shareware End User - Distribute Freely**

**License No**. : **SW1380 License Date**: Dec 05, 1998

**Build Date** : **Sep 2 2009 Run Date**: Apr 30, 2024

**©1996-2009 M Squared Technologies LLC™**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**License File: C:\Program Files (x86)\MSquared\M2 RSM\rsm.lic**

**Config. File: C:\Program Files (x86)\MSquared\M2 RSM\rsm.cfg**

**Command Line: -H -OC:\Users\claud\Documents\MastersLocal\CSE566\RSM\outp**

**ut.htm -TN -Tp -FC:\Users\claud\M2 RSM Wizard\input\rsm\_fi**

**le\_list.lst**

**UDQN File : C:\Program Files (x86)\MSquared\M2 RSM\rsm\_udqn.cfg**

**------------------------------------------------------------------------**

**~~ Total Metrics For 5 Files ~~**

**------------------------------------------------------------------------**

**~~ Project Quality Profile ~~**

**Type Count Percent Quality Notice**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1 38 11.52 Physical line length > 80 characters**

**5 1 0.30 Assignment "=" within "while" statement**

**7 8 2.42 Pre-increment operator "++" identified**

**14 1 0.30 "case" conditions do not equal "break"**

**17 14 4.24 Function comment content less than 10.0%**

**20 2 0.61 File comment content < 10.0%**

**22 12 3.64 if, else, for or while not bound by scope**

**23 10 3.03 "?" ternary operator identified**

**26 12 3.64 void \* generic type identified**

**27 5 1.52 Number of function return points > 1**

**28 8 2.42 Cyclomatic complexity > 10**

**29 1 0.30 Number of function parameters > 6**

**30 11 3.33 TAB character has been identified**

**43 1 0.30 Keyword "continue" has been identified**

**44 3 0.91 Keyword "break" identified outside a "switch" structure**

**46 12 3.64 Function/Class Blank Line content less < 10.0%**

**47 2 0.61 File Blank Line content < 10.0%**

**49 2 0.61 Function appears to have null or blank parameters**

**50 107 32.42 Variable assignment to a literal number**

**51 16 4.85 No comment preceding a function block**

**53 4 1.21 No comment preceding a struct block**

**55 8 2.42 Scope level exceeds the defined limit of 5**

**102 16 4.85 Dynamic memory using malloc is not initialized**

**105 3 0.91 A symbolic constant using #define**

**109 30 9.09 Double pointer indirection identified**

**125 3 0.91 A data member in the header file is not of the form m\_\***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**330 100.00 Total Quality Notices**

**~~ Quality Notice Density ~~**

**Basis: 1000 (K)**

**Quality Notices/K LOC = 433.1 ( 43.31%)**

**Quality Notices/K eLOC = 523.8 ( 52.38%)**

**Quality Notices/K lLOC = 688.9 ( 68.89%)**

**------------------------------------------------------------------------**

**~~ File Summary ~~**

**C Source Files *\*.c* ....: 3 C/C++ Include Files *\*.h*: 2**

**C++ Source Files *\*.c\** .: 0 C++ Include Files *\*.h\** : 0**

**C# Source Files *\*.cs* ..: 0 Java Source File *\*.jav\**: 0**

**Other Source Files ....: 0**

**Total File Count ......: 5**

**Shareware evaluation licenses process only 20 files.**

**Paid licenses enable processing for an unlimited number of files.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**