

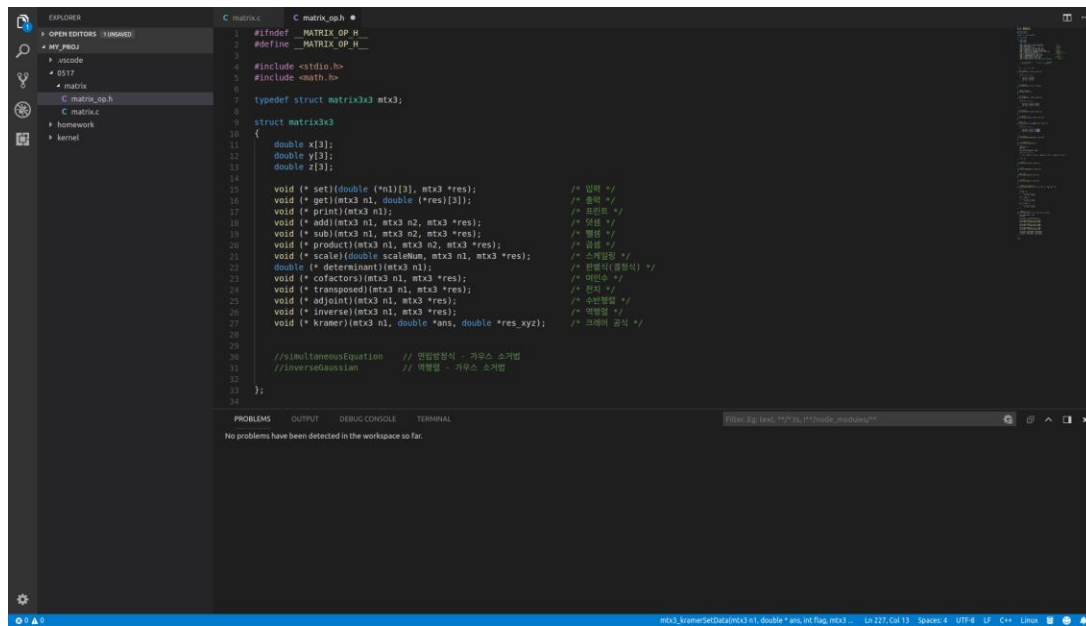


TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

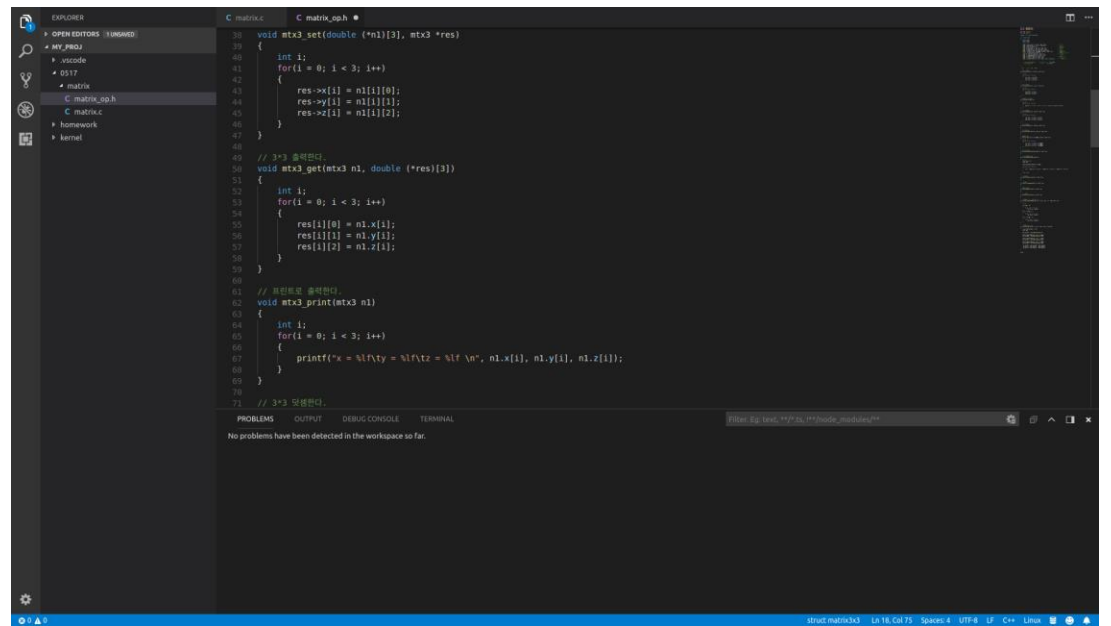
Innova Lee(이상훈) 강사
gcccompil3r@gmail.com

은태영 학생
zero_bird@naver.com

수학

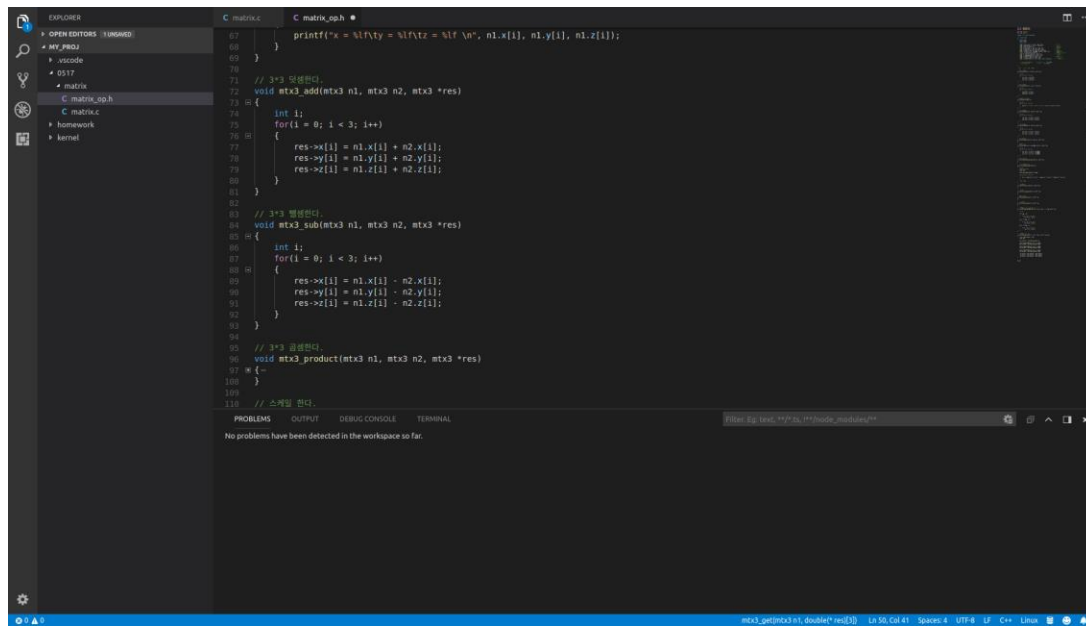


```
1 #ifndef MATRIX_OP_H
2 #define MATRIX_OP_H
3
4 #include <stdio.h>
5 #include <math.h>
6
7 typedef struct matrix3x3 mtx3;
8
9 struct matrix3x3
10 {
11     double x[3];
12     double y[3];
13     double z[3];
14
15     void (* set)(double (*n1)[3], mtx3 *res); /* 입력 */
16     void (* get)(mtx3 n1, double (*res)[3]); /* 출력 */
17     void (* print)(mtx3 n1); /* 표현 */
18     void (* add)(mtx3 n1, mtx3 n2, mtx3 *res); /* 덧셈 */
19     void (* sub)(mtx3 n1, mtx3 n2, mtx3 *res); /* 뺄셈 */
20     void (* product)(mtx3 n1, mtx3 n2, mtx3 *res); /* 곱셈 */
21     void (* scale)(double scaleNum, mtx3 n1, mtx3 *res); /* 스케일링 */
22     double (* determinant)(mtx3 n1); /* 행렬식(결정식) */
23     void (* cofactors)(mtx3 n1, mtx3 *res); /* 여인수 */
24     void (* transposed)(mtx3 n1, mtx3 *res); /* 전치 */
25     void (* adjoint)(mtx3 n1, mtx3 *res); /* 수반행렬 */
26     void (* inverse)(mtx3 n1, mtx3 *res); /* 역행렬 */
27     void (* kramer)(mtx3 n1, double *ans, double *res_xyz); /* 크래머 공식 */
28
29     // simultaneousEquation // 연립방정식 - 가우스 소거법
30     // inverseGaussian // 역행렬 - 가우스 소거법
31 };
32
33 #endif
```



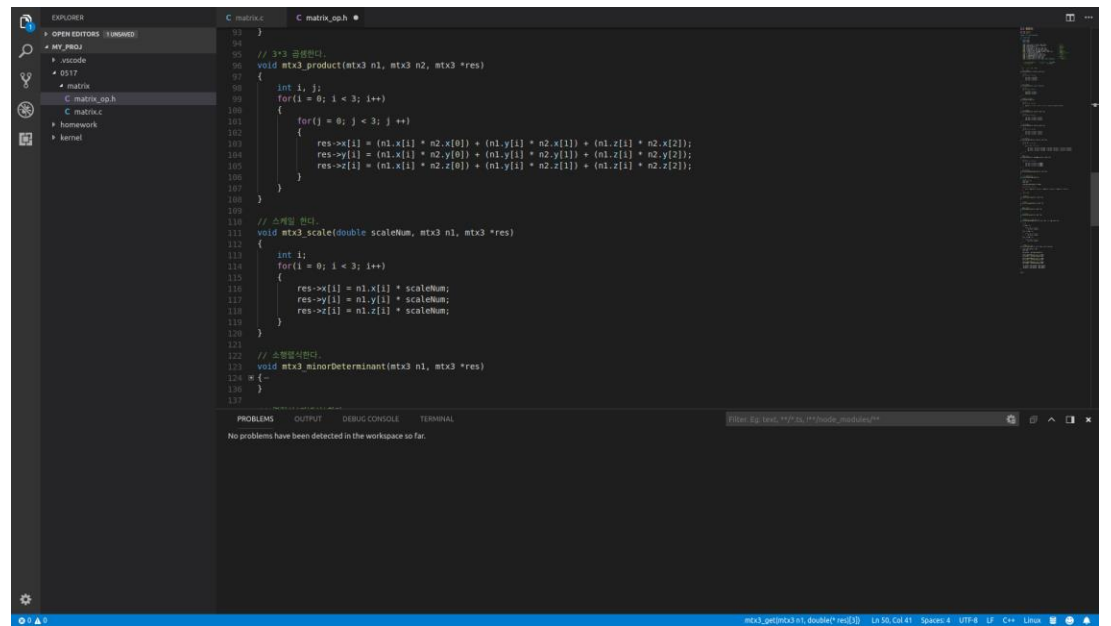
```
30 void mtx3_set(double (*n1)[3], mtx3 *res)
31 {
32     int i;
33     for(i = 0; i < 3; i++)
34     {
35         res->x[i] = n1[i][0];
36         res->y[i] = n1[i][1];
37         res->z[i] = n1[i][2];
38     }
39 }
40
41 // 3*3 출력한다.
42 void mtx3_get(mtx3 n1, double (*res)[3])
43 {
44     int i;
45     for(i = 0; i < 3; i++)
46     {
47         res[i][0] = n1.x[i];
48         res[i][1] = n1.y[i];
49         res[i][2] = n1.z[i];
50     }
51 }
52
53 // 표현으로 출력한다.
54 void mtx3_print(mtx3 n1)
55 {
56     int i;
57     for(i = 0; i < 3; i++)
58     {
59         printf("x = %lf\ty = %lf\tz = %lf\n", n1.x[i], n1.y[i], n1.z[i]);
60     }
61 }
62
63 // 3*3 되돌린다.
```

수학



VS Code editor showing C code for matrix operations. The Explorer sidebar on the left shows a project structure with files: MY_PROJ, .vscode, .gitignore, matrix, C_matrix_op.h, C_matrix.c, homework, and kernel. The main editor displays the content of C_matrix.c, which includes functions for printing a 3x3 matrix, adding, subtracting, multiplying, and scaling matrices. The code is written in C and uses standard library functions like printf and scanf. The bottom status bar shows the file path: mxx_gethx7a's doubleC\rec(3) and the current line is 110.

```
67     printf("x = %f\ty = %f\tr = %f\n", n1.x[i], n1.y[i], n1.z[i]);
68 }
69
70
71 // 3*3 덧셈한다.
72 void mtx3_add(mtx3 n1, mtx3 n2, mtx3 *res)
73 {
74     int i;
75     for(i = 0; i < 3; i++)
76     {
77         res->x[i] = n1.x[i] + n2.x[i];
78         res->y[i] = n1.y[i] + n2.y[i];
79         res->z[i] = n1.z[i] + n2.z[i];
80     }
81 }
82
83 // 3*3 뺄셈한다.
84 void mtx3_sub(mtx3 n1, mtx3 n2, mtx3 *res)
85 {
86     int i;
87     for(i = 0; i < 3; i++)
88     {
89         res->x[i] = n1.x[i] - n2.x[i];
90         res->y[i] = n1.y[i] - n2.y[i];
91         res->z[i] = n1.z[i] - n2.z[i];
92     }
93 }
94
95 // 3*3 곱셈한다.
96 void mtx3_product(mtx3 n1, mtx3 n2, mtx3 *res)
97 {
98     // 소괄호를 쓴다.
99     // 소괄호를 쓴다.
100 }
101
102 // 소괄호를 쓴다.
103 void mtx3_minorDeterminant(mtx3 n1, mtx3 *res)
104 {
105     // 소괄호를 쓴다.
106     // 소괄호를 쓴다.
107 }
108
109 // 소괄호를 쓴다.
110 void mtx3_scale(double scaleNum, mtx3 n1, mtx3 *res)
111 {
112     int i;
113     for(i = 0; i < 3; i++)
114     {
115         res->x[i] = n1.x[i] * scaleNum;
116         res->y[i] = n1.y[i] * scaleNum;
117         res->z[i] = n1.z[i] * scaleNum;
118     }
119 }
120
121 // 소괄호를 쓴다.
122 void mtx3_minorDeterminant(mtx3 n1, mtx3 *res)
123 {
124     // 소괄호를 쓴다.
125     // 소괄호를 쓴다.
126 }
127
```



VS Code editor showing C code for matrix operations. The Explorer sidebar on the left shows a project structure with files: MY_PROJ, .vscode, .gitignore, matrix, C_matrix_op.h, C_matrix.c, homework, and kernel. The main editor displays the content of C_matrix.c, which includes functions for printing a 3x3 matrix, adding, subtracting, multiplying, and scaling matrices. The code is written in C and uses standard library functions like printf and scanf. The bottom status bar shows the file path: mxx_gethx7a's doubleC\rec(3) and the current line is 110.

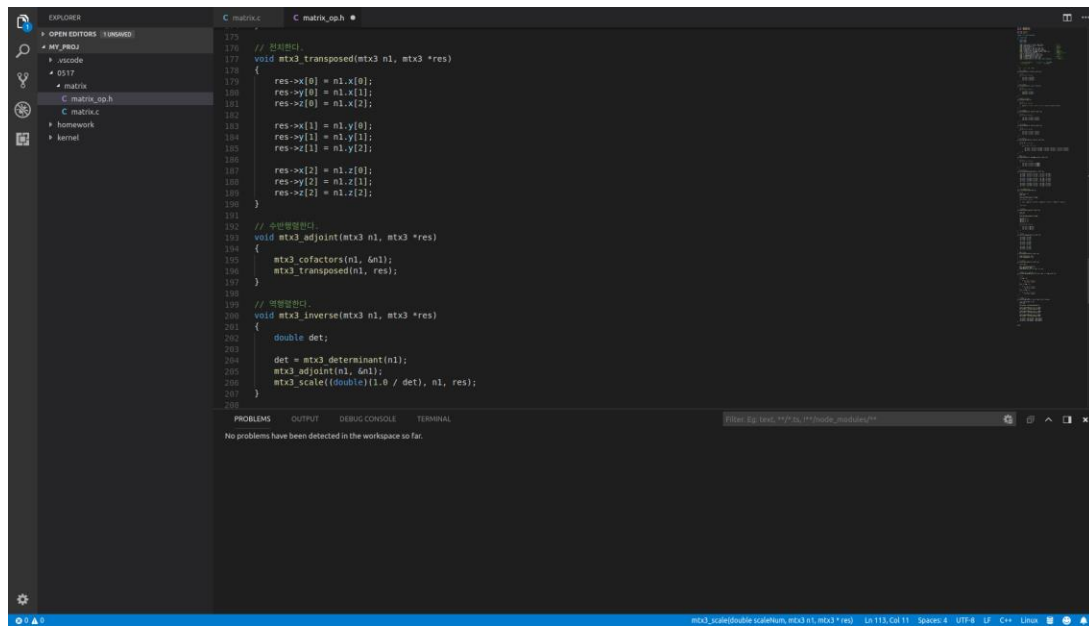
```
93 }
94
95 // 3*3 곱셈한다.
96 void mtx3_product(mtx3 n1, mtx3 n2, mtx3 *res)
97 {
98     int i, j;
99     for(i = 0; i < 3; i++)
100     {
101         for(j = 0; j < 3; j++)
102         {
103             res->x[i] = (n1.x[i] * n2.x[0]) + (n1.y[i] * n2.x[1]) + (n1.z[i] * n2.x[2]);
104             res->y[i] = (n1.x[i] * n2.y[0]) + (n1.y[i] * n2.y[1]) + (n1.z[i] * n2.y[2]);
105             res->z[i] = (n1.x[i] * n2.z[0]) + (n1.y[i] * n2.z[1]) + (n1.z[i] * n2.z[2]);
106         }
107     }
108 }
109
110 // 소괄호를 쓴다.
111 void mtx3_scale(double scaleNum, mtx3 n1, mtx3 *res)
112 {
113     int i;
114     for(i = 0; i < 3; i++)
115     {
116         res->x[i] = n1.x[i] * scaleNum;
117         res->y[i] = n1.y[i] * scaleNum;
118         res->z[i] = n1.z[i] * scaleNum;
119     }
120 }
121
122 // 소괄호를 쓴다.
123 void mtx3_minorDeterminant(mtx3 n1, mtx3 *res)
124 {
125     // 소괄호를 쓴다.
126     // 소괄호를 쓴다.
127 }
```

수학

```
122 // 소정렬식이다.
123 void mtx3_minorDeterminant(mtx3 n1, mtx3 *res)
124 {
125     res->x[0] = (n1.y[1] * n1.z[2]) - (n1.z[1] * n1.y[2]);
126     res->y[0] = (n1.x[1] * n1.z[2]) - (n1.z[1] * n1.x[2]);
127     res->z[0] = (n1.x[1] * n1.y[2]) - (n1.y[1] * n1.x[2]);
128
129     res->x[1] = (n1.y[0] * n1.z[2]) - (n1.z[0] * n1.y[2]);
130     res->y[1] = (n1.x[0] * n1.z[2]) - (n1.z[0] * n1.x[2]);
131     res->z[1] = (n1.x[0] * n1.y[2]) - (n1.y[0] * n1.x[2]);
132
133     res->x[2] = (n1.y[0] * n1.z[1]) - (n1.z[0] * n1.y[1]);
134     res->y[2] = (n1.x[0] * n1.z[1]) - (n1.z[0] * n1.x[1]);
135     res->z[2] = (n1.x[0] * n1.y[1]) - (n1.y[0] * n1.x[1]);
136 }
137
138 // 클랑식(반행식)이다.
139 double mtx3_determinant(mtx3 n1)
140 {
141     int i;
142     double res = 0;
143     mtx3 tmp;
144     mtx3_minorDeterminant(n1, &tmp);
145     for(i = 0; i < 3; i++)
146     {
147         res += (tmp.x[i] * n1.x[i]) + (tmp.y[i] * n1.y[i]) + (tmp.z[i] * n1.z[i]);
148     }
149     return res;
150 }
151
152 // 여인수한다.
```

```
151
152     return res;
153 }
154
155 // 여인수한다.
156 void mtx3_cofactors(mtx3 n1, mtx3 *res)
157 {
158     mtx3 tmp;
159     int i;
160
161     mtx3_minorDeterminant(n1, &tmp);
162     tmp.y[0] *= -1;
163     tmp.x[1] *= -1;
164     tmp.z[1] *= -1;
165     tmp.y[2] *= -1;
166     tmp.x[2] *= -1;
167     for(i = 0; i < 3; i++)
168     {
169         res->x[i] = tmp.x[i];
170         res->y[i] = tmp.y[i];
171         res->z[i] = tmp.z[i];
172     }
173 }
174
175 // 전치한다.
176 void mtx3_transposed(mtx3 n1, mtx3 *res)
177 {
178     res->x[0] = n1.y[0];
179     res->y[0] = n1.x[0];
180 }
181
182 // 수반행렬식이다.
183 void mtx3_adjoint(mtx3 n1, mtx3 *res)
184 {
185     //
186 }
```

수학



The image shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project structure with folders 'MY_PROJ', '.vscode', '.git', 'matrix', 'C', 'homework', and 'kernel'. The 'C' folder is expanded, showing 'matrix.c' and 'matrix_op.h'. The main editor area displays the contents of 'matrix.c', which includes functions for matrix transposition, adjoint calculation, and inversion. The code is written in C and uses standard mathematical operations. The bottom status bar shows the current file is 'mtx3_scale(double scale, mtx3 n1, mtx3 *res)' at line 115, column 11, with 5 spaces and UTF-8 encoding.

```
175
176 // 전치한다.
177 void mtx3_transposed(mtx3 n1, mtx3 *res)
178 {
179     res->x[0] = n1.x[0];
180     res->y[0] = n1.x[1];
181     res->z[0] = n1.x[2];
182
183     res->x[1] = n1.y[0];
184     res->y[1] = n1.y[1];
185     res->z[1] = n1.y[2];
186
187     res->x[2] = n1.z[0];
188     res->y[2] = n1.z[1];
189     res->z[2] = n1.z[2];
190 }
191
192 // 수반행렬한다.
193 void mtx3_adjoint(mtx3 n1, mtx3 *res)
194 {
195     mtx3 cofactors(n1, &n1);
196     mtx3_transposed(n1, res);
197 }
198
199 // 역행렬한다.
200 void mtx3_inverse(mtx3 n1, mtx3 *res)
201 {
202     double det;
203     det = mtx3_determinant(n1);
204     mtx3_adjoint(n1, &n1);
205     mtx3_scale((double)(1.0 / det), n1, res);
206 }
207
208
```

수학

```
204 det = mtx3_determinant(n1);
205 mtx3_adjoint(n1, &n1);
206 mtx3_scale((double)(1.0 / det), n1, res);
207
208 // 크래머의 공식을 위한 선언이다.
209 void mtx3_kramerSetData(mtx3 n1, double *ans, int flag, mtx3 *res)
210 {
211     int i;
212
213     if(flag == 0)
214     {
215         for(i = 0; i < 3; i++)
216             res->x[i] = ans[i];
217     }
218     else if(flag == 1)
219     {
220         for(i = 0; i < 3; i++)
221             res->y[i] = ans[i];
222     }
223     else if(flag == 2)
224     {
225         for(i = 0; i < 3; i++)
226             res->z[i] = ans[i];
227     }
228 }
229
230 // 크래머의 공식이다.
231 void mtx3_kramer(mtx3 n1, double *ans, double *res_xyz)
232 {
233     double det_xyz[4] = {0};
234     mtx3 tmp;
235     det_xyz[3] = mtx3_determinant(n1);
236
237     mtx3_kramerSetData(n1, ans, 0, &tmp);
238     det_xyz[0] = mtx3_determinant(tmp);
239     mtx3_kramerSetData(n1, ans, 1, &tmp);
240     det_xyz[1] = mtx3_determinant(tmp);
241     mtx3_kramerSetData(n1, ans, 2, &tmp);
242     det_xyz[2] = mtx3_determinant(tmp);
243
244     res_xyz[0] = det_xyz[0] / det_xyz[3];
245     res_xyz[1] = det_xyz[1] / det_xyz[3];
246     res_xyz[2] = det_xyz[2] / det_xyz[3];
247 }
```

```
225 {
226     for(i = 0; i < 3; i++)
227         res->z[i] = ans[i];
228 }
229
230 // 크래머의 공식이다.
231 void mtx3_kramer(mtx3 n1, double *ans, double *res_xyz)
232 {
233     double det_xyz[4] = {0};
234     mtx3 tmp;
235     det_xyz[3] = mtx3_determinant(n1);
236
237     mtx3_kramerSetData(n1, ans, 0, &tmp);
238     det_xyz[0] = mtx3_determinant(tmp);
239     mtx3_kramerSetData(n1, ans, 1, &tmp);
240     det_xyz[1] = mtx3_determinant(tmp);
241     mtx3_kramerSetData(n1, ans, 2, &tmp);
242     det_xyz[2] = mtx3_determinant(tmp);
243
244     res_xyz[0] = det_xyz[0] / det_xyz[3];
245     res_xyz[1] = det_xyz[1] / det_xyz[3];
246     res_xyz[2] = det_xyz[2] / det_xyz[3];
247 }
```