

```

1 #ifndef SELECTION
2 #define SELECTION
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <ctype.h>
7 #include <string.h>
8 #include <stdbool.h>
9
10 #include "../datatypes/enum.h"
11
12 #include "print.h"
13
14 char *toLower(char *string)
15 {
16     unsigned char *char_ptr = (unsigned char *)string;
17
18     while (*char_ptr)
19     {
20         *char_ptr = tolower(*char_ptr);
21         char_ptr++;
22     }
23 }
24
25 bool ShapeSelection(enum shape *shape)
26 {
27     char *input;
28
29     ShapeSelectionInstructions();
30
31     while (true)
32     {
33         if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
34         {
35             NoMemoryAlert();
36             exit(1);
37         }
38         fgets(input, 100 * sizeof(char), stdin);
39         toLower(input);
40
41         if (strcmp(input, "rectangle\n") == 0 || strcmp(input, "1\n") == 0)
42         {
43             free(input);
44             *shape = Rectangle;
45             return true;
46         }
47         else if (strcmp(input, "square\n") == 0 || strcmp(input, "2\n") == 0)
48         {
49             free(input);
50             *shape = Square;
51             return true;
52         }
53         else if (strcmp(input, "circle\n") == 0 || strcmp(input, "3\n") == 0)
54         {
55             free(input);
56             *shape = Circle;
57             return true;
58         }
59         else if (strcmp(input, "back\n") == 0)
60         {
61             free(input);
62             return false;
63         }
64         else if (strcmp(input, "exit\n") == 0)
65         {
66             free(input);
67             exit(0);
68         }
69         else
70         {
71             WrongShapeInput();
72         }
73     }
74 }
75
76 bool ObjectSelection(enum shape *shape)
77 {
78     char *input;
79
80     ObjectSelectionInstructions();
81
82     while (true)
83     {
84         if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
85         {
86             NoMemoryAlert();
87             exit(1);
88         }
89         fgets(input, 100 * sizeof(char), stdin);
90         toLower(input);
91
92         if (strcmp(input, "cuboid\n") == 0 || strcmp(input, "1\n") == 0)
93         {
94             free(input);

```

```

95     *shape = Cuboid;
96     return true;
97 }
98 else if (strcmp(input, "cube\n") == 0 || strcmp(input, "2\n") == 0)
99 {
100     free(input);
101     *shape = Cube;
102     return true;
103 }
104 else if (strcmp(input, "sphere\n") == 0 || strcmp(input, "3\n") == 0)
105 {
106     free(input);
107     *shape = Sphere;
108     return true;
109 }
110 else if (strcmp(input, "cone\n") == 0 || strcmp(input, "4\n") == 0)
111 {
112     free(input);
113     *shape = Cone;
114     return true;
115 }
116 else if (strcmp(input, "back\n") == 0)
117 {
118     free(input);
119     return false;
120 }
121 else if (strcmp(input, "exit\n") == 0)
122 {
123     free(input);
124     exit(0);
125 }
126 else
127 {
128     WrongObjectInput();
129 }
130 }
131 }
132
133 bool GeometrySelection(enum shape *shape, int dimension)
134 {
135     switch (dimension)
136     {
137     case 2:
138         return ShapeSelection(&(*shape));
139         break;
140     case 3:
141         return ObjectSelection(&(*shape));
142         break;
143     }
144     return false;
145 }
146
147 void DimensionSelection(int *dimension)
148 {
149     char *input;
150
151     DimensionSelectionInstructions();
152
153     while (true)
154     {
155         if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
156         {
157             NoMemoryAlert();
158             exit(1);
159         }
160         fgets(input, 100 * sizeof(char), stdin);
161         toLower(input);
162
163         if (strcmp(input, "2d\n") == 0 || strcmp(input, "1\n") == 0)
164         {
165             free(input);
166             *dimension = 2;
167             return;
168         }
169         else if (strcmp(input, "3d\n") == 0 || strcmp(input, "2\n") == 0)
170         {
171             free(input);
172             *dimension = 3;
173             return;
174         }
175         else if (strcmp(input, "exit\n") == 0)
176         {
177             free(input);
178             exit(0);
179         }
180         else
181         {
182             WrongDimensionInput();
183         }
184     }
185 }
186
187 void UnitSelection(enum unit *unit)
188 {
189     char *input;

```

```

190
191 UnitSelectionInstructions();
192
193 while (true)
194 {
195     if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
196     {
197         NoMemoryAlert();
198         exit(1);
199     }
200     fgets(input, 100 * sizeof(char), stdin);
201     toLower(input);
202
203     if (strcmp(input, "m\n") == 0 || strcmp(input, "1\n") == 0)
204     {
205         *unit = m;
206         free(input);
207         return;
208     }
209     else if (strcmp(input, "dm\n") == 0 || strcmp(input, "2\n") == 0)
210     {
211         *unit = dm;
212         free(input);
213         return;
214     }
215     else if (strcmp(input, "cm\n") == 0 || strcmp(input, "3\n") == 0)
216     {
217         *unit = cm;
218         free(input);
219         return;
220     }
221     else if (strcmp(input, "mm\n") == 0 || strcmp(input, "4\n") == 0)
222     {
223         *unit = mm;
224         free(input);
225         return;
226     }
227     else
228     {
229         WrongUnitInput();
230     }
231 }
232 }
233
234 bool ProcessSelection()
235 {
236     char *input;
237
238     ProcessSelectionInstructions();
239
240     while (true)
241     {
242         if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
243         {
244             NoMemoryAlert();
245             exit(1);
246         }
247         fgets(input, 100 * sizeof(char), stdin);
248         toLower(input);
249
250         if (strcmp(input, "history\n") == 0 || strcmp(input, "1\n") == 0)
251         {
252             free(input);
253             return true;
254         }
255         else if (strcmp(input, "calculate\n") == 0 || strcmp(input, "2\n") == 0)
256         {
257             free(input);
258             return false;
259         }
260         else if (strcmp(input, "exit\n") == 0 || strcmp(input, "3\n") == 0)
261         {
262             free(input);
263             exit(0);
264         }
265         else
266         {
267             WrongProcessInput();
268         }
269     }
270 }
271
272 void ShapeAndObjectSelection(enum shape *shape)
273 {
274     char *input;
275
276     ShapeAndObjectSelectionInstructions();
277
278     while (true)
279     {
280
281         if ((input = (char *)malloc(100 * sizeof(char))) == NULL)
282         {
283             NoMemoryAlert();
284             exit(1);

```

```

285     }
286     fgets(input, 100 * sizeof(char), stdin);
287     toLower(input);
288
289     if (strcmp(input, "rectangle\n") == 0 || strcmp(input, "1\n") == 0)
290     {
291         free(input);
292         *shape = Rectangle;
293         return;
294     }
295     else if (strcmp(input, "square\n") == 0 || strcmp(input, "2\n") == 0)
296     {
297         free(input);
298         *shape = Square;
299         return;
300     }
301     else if (strcmp(input, "circle\n") == 0 || strcmp(input, "3\n") == 0)
302     {
303         free(input);
304         *shape = Circle;
305         return;
306     }
307     else if (strcmp(input, "cuboid\n") == 0 || strcmp(input, "4\n") == 0)
308     {
309         free(input);
310         *shape = Cuboid;
311         return;
312     }
313     else if (strcmp(input, "cube\n") == 0 || strcmp(input, "5\n") == 0)
314     {
315         free(input);
316         *shape = Cube;
317         return;
318     }
319     else if (strcmp(input, "sphere\n") == 0 || strcmp(input, "6\n") == 0)
320     {
321         free(input);
322         *shape = Sphere;
323         return;
324     }
325     else if (strcmp(input, "cone\n") == 0 || strcmp(input, "7\n") == 0)
326     {
327         free(input);
328         *shape = Cone;
329         return;
330     }
331     else if (strcmp(input, "exit\n") == 0)
332     {
333         free(input);
334         exit(0);
335     }
336     else
337     {
338         WrongShapeAndObjectInput();
339     }
340 }
341 }
342
343 #endif

```