```zig
const std = @import("std");

const dds = @import("dds");

// keyboard
const kbd = @import("cursed").kbd;

// panel
const pnl = @import("forms").pnl;
// button
const btn = @import("forms").btn;
// label
const lbl = @import("forms").lbl;
// menu
const mnu = @import("forms").mnu;
// flied
const fld = @import("forms").fld;
// line horizontal
const lnh = @import("forms").lnh;
// line vertival
const lnv = @import("forms").lnv;

// grid
const grd = @import("grid").grd;

// full delete for produc
const forms = @import("forms");
const allocator = std.heap.page_allocator;

pub fn SavJson(XPANEL: std.ArrayList(pnl.PANEL)) !void {
    var out_buf: [20480]u8 = undefined;
    var slice_stream = std.io.fixedBufferStream(&out_buf);
    const out = slice_stream.writer();

    var w = std.json.writeStream(out, .{ .whitespace = .indent_2 });

    const Ipanel = std.enums.EnumIndexer(pnl.Epanel);
    try w.beginObject();
    try w.objectField("PANEL");
    var nbrPnl: usize = XPANEL.items.len;
    var np: usize = 0;
    while (np < nbrPnl) : (np += 1) {
        try w.beginArray();
        try w.beginObject();
        var p: usize = 0;
        while (p < Ipanel.count) : (p += 1) {
```

```
47:                 switch (Ipanel.keyForIndex(p)) {
48:                     .name => {
49:                         try w.objectField(@tagName(pnl.Epanel.name));
50:                         try w.print("\"{s}\"", .{XPANEL.items[np].name});
51:                     },
52:                     .posx => {
53:                         try w.objectField(@tagName(pnl.Epanel.posx));
54:                         try w.print("{d}", .{XPANEL.items[np].posx});
55:                     },
56:                     .posy => {
57:                         try w.objectField(@tagName(pnl.Epanel.posy));
58:                         try w.print("{d}", .{XPANEL.items[np].posy});
59:                     },
60:                     .lines => {
61:                         try w.objectField(@tagName(pnl.Epanel.lines));
62:                         try w.print("{d}", .{XPANEL.items[np].lines});
63:                     },
64:                     .cadre => {
65:                         try w.objectField(@tagName(pnl.Epanel.cadre));
66:                         try w.print("{s}", .{@tagName(XPANEL.items[np].frame.cadre)});
67:                     },
68:                     .title => {
69:                         try w.objectField(@tagName(pnl.Epanel.title));
70:                         try w.print("{s}", .{XPANEL.items[np].frame.title});
71:                     },
72:                     .button => {
73:                         const Ibutton = std.enums.EnumIndexer(btn.Ebutton);
74:                         var nbrBtn: usize = XPANEL.items[np].button.items.len;
75:                         var bp: usize = 0;
76:                         try w.objectField("button");
77:                         try w.beginArray();
78:                         while (bp < nbrBtn) : (bp += 1) {
79:                             try w.beginObject();
80:                             var b: usize = 0;
81:                             while (b < Ibutton.count) : (b += 1) {
82:                                 switch (Ibutton.keyForIndex(b)) {
83:                                     .name => {
84:                                         try w.objectField(@tagName(btn.Ebutton.name));
85:                                         try w.print("\"{s}\"", .{XPANEL.items[np].button.items[bp].name});
86:                                     },
87:                                     .key => {
88:                                         try w.objectField(@tagName(btn.Ebutton.key));
89:                                         try w.print("\"{s}\"", .{@tagName(XPANEL.items[np].button.items[bp].key)});
90:                                     },
91:                                     .show => {
92:                                         try w.objectField(@tagName(btn.Ebutton.show));
```

```
93:                                     try w.print("{d}", .{@intFromBool(XPANEL.items[np].button.items[bp].show)});
94:                                 },
95:                                 .check => {
96:                                     try w.objectField(@tagName(btn.Ebutton.check));
97:                                     try w.print("{d}", .{@intFromBool(XPANEL.items[np].button.items[bp].check)});
98:                                 },
99:                                 .title => {
100:                                    try w.objectField(@tagName(btn.Ebutton.title));
101:                                    try w.print("\"{s}\"", .{XPANEL.items[np].button.items[bp].title});
102:                                },
103:                            }
104:                        }
105:                        try w.endObject();
106:                    }
107:
108:                    try w.endArray();
109:                },
110:                .label => {
111:                    const Ilabel = std.enums.EnumIndexer(lbl.Elabel);
112:                    var l: usize = 0;
113:                    var nbrLbl: usize = XPANEL.items[np].label.items.len;
114:
115:                    var lp: usize = 0;
116:                    try w.objectField("label");
117:                    try w.beginArray();
118:                    while (lp < nbrLbl) : (lp += 1) {
119:                        try w.beginObject();
120:                        l = 0;
121:                        while (l < Ilabel.count) : (l += 1) {
122:                            switch (Ilabel.keyForIndex(l)) {
123:                                .name => {
124:                                    try w.objectField(@tagName(lbl.Elabel.name));
125:                                    try w.print("\"{s}\"", .{XPANEL.items[np].label.items[lp].name});
126:                                },
127:                                .posx => {
128:                                    try w.objectField(@tagName(lbl.Elabel.posx));
129:                                    try w.print("{d}", .{XPANEL.items[np].label.items[lp].posx});
130:                                },
131:                                .posy => {
132:                                    try w.objectField(@tagName(lbl.Elabel.posy));
133:                                    try w.print("{d}", .{XPANEL.items[np].label.items[lp].posy});
134:                                },
135:                                .text => {
136:                                    try w.objectField(@tagName(lbl.Elabel.title));
137:                                    try w.print("\"{s}\"", .{XPANEL.items[np].label.items[lp].text});
138:                                },
```

```
139:                                .title => {
140:                                    try w.objectField(@tagName(lbl.Elabel.title));
141:                                    try w.print("{d}", .{@intFromBool(XPANEL.items[np].label.items[lp].title)});
142:                                },
143:                            }
144:                        }
145:                        try w.endObject();
146:                    }
147:
148:                    try w.endArray();
149:                },
150:                .field => {
151:                    const Ifield = std.enums.EnumIndexer(fld.Efield);
152:                    var f: usize = 0;
153:                    var nbrFld: usize = XPANEL.items[np].field.items.len;
154:
155:                    var fp: usize = 0;
156:                    try w.objectField("field");
157:                    try w.beginArray();
158:                    while (fp < nbrFld) : (fp += 1) {
159:                        try w.beginObject();
160:                        f = 0;
161:                        while (f < Ifield.count) : (f += 1) {
162:                            switch (Ifield.keyForIndex(f)) {
163:                                .name => {
164:                                    try w.objectField(@tagName(fld.Efield.name));
165:                                    try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].name});
166:                                },
167:                                .posx => {
168:                                    try w.objectField(@tagName(fld.Efield.posx));
169:                                    try w.print("{d}", .{XPANEL.items[np].field.items[fp].posx});
170:                                },
171:                                .posy => {
172:                                    try w.objectField(@tagName(fld.Efield.posy));
173:                                    try w.print("{d}", .{XPANEL.items[np].field.items[fp].posy});
174:                                },
175:                                .reftyp => {
176:                                    try w.objectField(@tagName(fld.Efield.reftyp));
177:                                    try w.print("{s}", .{@tagName(XPANEL.items[np].field.items[fp].reftyp)});
178:                                },
179:                                .width => {
180:                                    try w.objectField(@tagName(fld.Efield.width));
181:                                    try w.print("{d}", .{XPANEL.items[np].field.items[fp].width});
182:                                },
183:                                .scal => {
184:                                    try w.objectField(@tagName(fld.Efield.scal));
```

```
185:                                     try w.print("{d}", .{XPANEL.items[np].field.items[fp].scal});
186:                                 },
187:                                 .requier => {
188:                                     try w.objectField(@tagName(fld.Efield.requier));
189:                                     try w.print("{d}", .{@intFromBool(XPANEL.items[np].field.items[fp].requier)});
190:                                 },
191:                                 .protect => {
192:                                     try w.objectField(@tagName(fld.Efield.protect));
193:                                     try w.print("{d}", .{@intFromBool(XPANEL.items[np].field.items[fp].protect)});
194:                                 },
195:                                 .edtcar => {
196:                                     try w.objectField(@tagName(fld.Efield.edtcar));
197:                                     try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].edtcar});
198:                                 },
199:                                 .errmsg => {
200:                                     try w.objectField(@tagName(fld.Efield.errmsg));
201:                                     try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].errmsg});
202:                                 },
203:                                 .help => {
204:                                     try w.objectField(@tagName(fld.Efield.help));
205:                                     try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].help});
206:                                 },
207:                                 .procfunc => {
208:                                     try w.objectField(@tagName(fld.Efield.procfunc));
209:                                     try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].procfunc});
210:                                 },
211:                                 .proctask => {
212:                                     try w.objectField(@tagName(fld.Efield.proctask));
213:                                     try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].proctask});
214:                                 },
215:                             }
216:                         }
217:                         try w.endObject();
218:                     }
219:
220:                     try w.endArray();
221:                 },
222:                 else => {},
223:             }
224:         }
225:         try w.endObject();
226:         try w.endArray();
227:     }
228:     try w.endObject();
229:
230:     const result = slice_stream.getWritten();
```

```
231:
232:     var my_file = try std.fs.cwd().createFile("Zdspf.txt", .{ .read = true });
233:     _ = try my_file.write(result);
234:     my_file.close();
235:
236:     _ = kbd.getKEY();
237:     slice_stream.reset();
238: }
```