```
 1: const std = @import("std");
 2:
 3:
 4: /// terminal Fonction
 5: const term = @import("cursed");
 6: // keyboard
 7: const kbd = @import("cursed").kbd;
 8:
 9: // error
10: const dsperr = @import("forms").dsperr;
11:
12: // full delete for produc
13: const forms = @import("forms");
14:
15: // frame
16: const frm = @import("forms").frm;
17:
18: // panel
19: const pnl = @import("forms").pnl;
20:
21: // button
22: const btn = @import("forms").btn;
23:
24: // label
25: const lbl = @import("forms").lbl;
26:
27: // flied
28: const fld = @import("forms").fld;
29:
30: // line horizontal
31: const lnh = @import("forms").lnh;
32:
33: // line vertival
34: const lnv = @import("forms").lnv;
35:
36: // grid
37: const grd = @import("grid").grd;
38:
39: // menu
40: const mnu = @import("menu").mnu;
41:
42: // tools utility
43: const utl = @import("utils");
44:
45: // tools regex
46: const reg = @import("match");
```

```
47:
48: // tools execve Pgm
49: const mdl = @import("modul");
50:
51: /// Errors
52: pub const Error = error{
53:     main_function_Enum_invalide,
54: };
55:
56:
57: /// ------------------------------------------------
58: /// Exemple defined Panel Label Field Button Menu Grid
59: /// ------------------------------------------------
60:
61: pub fn Panel_Fmt01() *pnl.PANEL {
62:
63:     //------------------------------------------------
64:     // Panel
65:     // Name Panel, Pos X, Pos Y,
66:     // nbr Lines, nbr columns
67:     // Attribut Panel
68:     // Type frame, Attribut frame
69:     // Title Panel, Attribut: Title
70:     var Panel    : *pnl.PANEL = pnl.newPanelC("Fmt01",
71:                     1, 1,
72:                     32,
73:                     132,
74:                     forms.CADRE.line1,
75:                     "TITLE");
76:
77:     //------------------------------------------------
78:     // Label
79:     // Name , pos X, pos Y,
80:     // Text , Attribut Text
81:     Panel.label.append(lbl.newLabel("free",2,2,"Text-Free.................:")
82:     ) catch unreachable ;
83:
84:     Panel.label.append(lbl.newLabel("full",3,2,"Text-Full.....protect.......:")
85:     ) catch unreachable ;
86:
87:     Panel.label.append(lbl.newLabel("cb01",3,62,"Fonction 01..:")
88:     ) catch unreachable ;
89:
90:     Panel.label.append(lbl.newLabel("cb02",4,62,"Fonction 02..:")
91:     ) catch unreachable ;
92:
```

```
 93:     Panel.label.append(lbl.newLabel("alpha",5,2,"Text-Alpha.................:")
 94:     ) catch unreachable ;
 95:
 96:     Panel.label.append(lbl.newLabel("alphaupper",6,2,"Text-Alpha-Uppercase........:")
 97:     ) catch unreachable ;
 98:
 99:     Panel.label.append(lbl.newLabel("alphanumeric",7,2,"Text-Alpha-Numeric..........:")
100:     ) catch unreachable ;
101:
102:     Panel.label.append(lbl.newLabel("alphanumericupper",8,2,"Text-Alpha-Numeric-Upercase.:")
103:     ) catch unreachable ;
104:
105:     Panel.label.append(lbl.newLabel("password",10,2,"Text-Password..............:")
106:     ) catch unreachable ;
107:
108:     Panel.label.append(lbl.newLabel("yesno",11,2,"Text-Yes or No.............:")
109:     ) catch unreachable ;
110:
111:     Panel.label.append(lbl.newLabel("udigit",13,2,"Text-Unsigned.Digit.........:")
112:     ) catch unreachable ;
113:
114:     Panel.label.append(lbl.newLabel("digit",14,2,"Text-signed.Digit...........:")
115:     ) catch unreachable ;
116:
117:     Panel.label.append(lbl.newLabel("udecimal",15,2,"Text-unsigned.Ddecimal......:")
118:     ) catch unreachable ;
119:
120:     Panel.label.append(lbl.newLabel("decimal",16,2,"Text-signed.Ddecimal........:")
121:     ) catch unreachable ;
122:
123:     Panel.label.append(lbl.newLabel("dateiso",18,2,"Text-Date-ISO..............:")
124:     ) catch unreachable ;
125:
126:     Panel.label.append(lbl.newLabel("datefr",19,2,"Text-Date-FR...............:")
127:     ) catch unreachable;
128:
129:     Panel.label.append(lbl.newLabel("dateus",20,2,"Text-Date-US...............:")
130:     ) catch unreachable;
131:
132:     Panel.label.append(lbl.newLabel("telephone",22,2,"Text-Telephone..US.........:")
133:     ) catch unreachable;
134:
135:     Panel.label.append(lbl.newLabel("telephone2",24,2,"Text-Telephone..Standard....:")
136:     ) catch unreachable;
137:
138:     Panel.label.append(lbl.newLabel("mail",26,2,"Text-Mail...................:")
```

```
139:     ) catch unreachable;
140:
141:     Panel.label.append(lbl.newLabel("switch",28,2,"Text-Switch.................:",)
142:     ) catch unreachable;
143:
144:     Panel.label.append(lbl.newTitle("TITLE",29,70,"Title ex : FACTURE",)
145:     ) catch unreachable;
146:
147:     //example: option specific
148:     Panel.label.items[1].attribut.styled[0] = @intFromEnum(term.Style.styleItalic);
149:     Panel.label.items[1].attribut.styled[1] = @intFromEnum(term.Style.notStyle);
150:
151:
152:  // Field
153:
154:
155:     Panel.field.append(fld.newFieldTextFree("free",2,32,         // Name , posx posy
156:                                         30,                     // width
157:                                         "free",                 // text
158:                                         true,                   // tofill
159:                                         "required",             // error msg
160:                                         "please enter text",    // help
161:                                         "",                     // regex
162:                                         )
163:     ) catch unreachable ;
164:
165:     Panel.field.append(fld.newFieldTextFull("full",3,32,         // Name , posx posy
166:                                         30,                     // width
167:                                         "full",                 // text
168:                                         true,                   // tofill
169:                                         "required",             // error msg
170:                                         "please enter text",    // help
171:                                         "",                     // regex
172:                                         )
173:     ) catch unreachable ;
174:
175:     fld.setProtect(Panel,1,true) catch unreachable;
176:
177:     Panel.field.append(fld.newFieldAlpha("alpha",5,32,              // Name , posx posy
178:                                         30,                     // width
179:                                         "abcd",                 // text
180:                                         true,                   // tofill
181:                                         "required",             // error msg
182:                                         "please enter text Alpha crtl+p call Exemple",  // help
183:                                         "^[a-zA-Z]{1,}$",           // regex
184:                                         )
```

```
185:     ) catch unreachable ;
186:
187:     fld.setCall(Panel,fld.getIndex(Panel,"alpha") catch unreachable,"exCallpgm") catch unreachable; // test appel pgm
188:
189:     Panel.field.append(fld.newFieldAlphaUpper("alphaU",6,32,                    // Name , posx posy
190:                                         30,                                    // width
191:                                         "ABCD",                                // text
192:                                         true,                                  // tofill
193:                                         "required",                            // error msg
194:                                         "please enter text Alpha Uppercase",   // help
195:                                         "",                                    // regex
196:                                         )
197:     ) catch unreachable ;
198:
199:     Panel.field.append(fld.newFieldAlphaNumeric("alphaN",7,32,                 // Name , posx posy
200:                                         30,                                    // width
201:                                         "abcd12345",                           // text
202:                                         true,                                  // tofill
203:                                         "required",                            // error msg
204:                                         "please enter text Alpha NumÃ©ric", // help
205:                                         "^[a-zA-Z]{1,1}[a-zA-Z0-9]{0,}",       // regex
206:                                         )
207:     ) catch unreachable ;
208:
209:     Panel.field.append(fld.newFieldAlphaNumericUpper("alphaNU",8,32,           // Name , posx posy
210:                                         30,                                    // width
211:                                         "ABCD12345",                           // text
212:                                         true,                                  // tofill
213:                                         "required",                            // error msg
214:                                         "please enter text Alpha NumÃ©ric", // help
215:                                         "^[A-Z]{1,1}[A-Z0-9]{0,}",             // regex
216:                                         )
217:     ) catch unreachable ;
218:
219:     Panel.field.append(fld.newFieldPassword("password",10,32,                  // Name , posx posy
220:                                         30,                                    // width
221:                                         "SECRET",                              // text
222:                                         true,                                  // tofill
223:                                         "required",                            // error msg
224:                                         "please enter text Alpha NumÃ©ric", // help
225:                                         "",                                    // regex
226:                                         )
227:     ) catch unreachable ;
228:
229:     Panel.field.append(fld.newFieldYesNo("yesno",11,32,      // Name , posx posy
230:                                         "N",                 // text
```

```
231:                                        true,                 // tofill
232:                                        "required Y or N",    // error msg
233:                                        "",                   // help default "to validate Y or N "
234:                                        )
235:     ) catch unreachable ;
236:
237:     Panel.field.append(fld.newFieldUDigit("udigit",13,32,            // Name , posx posy
238:                                        5,                            // width
239:                                        "00102",                      // text
240:                                        true,                         // tofill
241:                                        "Invalide value",             // error msg
242:                                        "value numeric not signed",   // help
243:                                        "",                           // regex default standard
244:                                        )
245:     ) catch unreachable ;
246:
247:     Panel.field.append(fld.newFieldDigit("digit",14,32,          // Name , posx posy
248:                                        5,                            // width
249:                                        "+00102",                     // text
250:                                        true,                         // tofill
251:                                        "Invalide value",             // error msg
252:                                        "value numeric signed",       // help
253:                                        "",                           // regex default standard
254:                                        )
255:     ) catch unreachable ;
256:
257:     Panel.field.append(fld.newFieldUDecimal("udecimal",15,32,         // Name , posx posy
258:                                        10,                           // width
259:                                        2,                            // scal
260:                                        "001.02",                     // text
261:                                        true,                         // tofill
262:                                        "Invalide value",             // error msg
263:                                        "",                           // help default
264:                                        "",                           // regex default standard
265:                                        )
266:     ) catch unreachable ;
267:
268:     Panel.field.append(fld.newFieldDecimal("decimal",16,32,      // Name , posx posy
269:                                        10,                           // width
270:                                        2,                            // scal
271:                                        "+001.02",                    // text
272:                                        true,                         // tofill
273:                                        "Invalide value",             // error msg
274:                                        "",                           // help default
275:                                        "",                           // regex default standard
276:                                        )
```

```
277:     ) catch unreachable ;
278:
279:     Panel.field.append(fld.newFieldDateISO("dateiso",18,32,      // Name , posx posy
280:                                         "1951-10-12",           // text
281:                                         true,                   // tofill
282:                                         "required",             // error msg
283:                                         "",                     // help default
284:                                         )
285:     ) catch unreachable ;
286:
287:     Panel.field.append(fld.newFieldDateFR("datefr",19,32,        // Name , posx posy
288:                                         "12/10/1951",           // text
289:                                         true,                   // tofill
290:                                         "required",             // error msg
291:                                         "",                     // help default
292:                                         )
293:     ) catch unreachable ;
294:
295:     Panel.field.append(fld.newFieldDateUS("dateus",20,32,            // Name , posx posy
296:                                         "07/04/1776",               // text
297:                                         true,                       // tofill
298:                                         "required",                 // error msg
299:                                         "",                         // help default
300:                                         )
301:     ) catch unreachable ;
302:
303:     Panel.field.append(fld.newFieldTelephone("telephone",22,32,      // Name , posx posy
304:                                         25,                         // width
305:                                         "+(001)451 452 453 545",    // text
306:                                         true,                       // tofill
307:                                         "required or invalide",     // error msg
308:                                         "ex:+(001)456.123.789",     // help
309:             "^[+]{1,1}[(]([0-9]{3,3})[)]([-. ]?[0-9]{3}){2,4}$",    // regex US default standard
310:                                         )
311:     ) catch unreachable ;
312:
313:     Panel.field.append(fld.newFieldTelephone("telephone2",24,32,            // Name , posx posy
314:                                         25,                                 // width
315:                                         "+(33)6 01 02 03 04",               // text
316:                                         false,                              // tofill
317:                                         "required or invalide",             // error msg
318:                                         "ex:+(33)6.12.34.56.78",            // help
319: "^[+]{1,1}[(]{0,1}[0-9]{1,3}[)]([0-9]{1,3}){1,1}([-. ]?[0-9]{2,3}){2,4}$"   // regex default standard fr
320:                                         )
321:     ) catch unreachable ;
322:
```

```
323:     Panel.field.append(fld.newFieldMail("mail",26,32,            // Name , posx posy
324:                                         100,                     // width
325:                                         "gloups@gmail.com",      // text error
326:                                         true,                    // tofill
327:                                         "required",              // error msg
328:                                         "",                      // help default
329:                                         )
330:     ) catch unreachable ;
331:
332:     Panel.field.append(fld.newFieldSwitch("Switch",28,32,        // Name , posx posy
333:                                         true,                    // switch
334:                                         "required",              // error msg
335:                                         "",                      // help
336:                                         )
337:     ) catch unreachable ;
338:
339:      Panel.field.append(fld.newFieldFunc("cb01",3,76,            // Name , posx posy
340:                                         20,                      // width
341:                                         "Amis",                  // text
342:                                         true,                    // tofill
343:                                         "comboFn01",             // Process for FUNC
344:                                         "required",              // error msg
345:                                         "select combo",          // help
346:                                         )
347:     ) catch unreachable ;
348:
349:     fld.setCall(Panel,fld.getIndex(Panel,"cb01") catch unreachable,"exCallpgm") catch unreachable; // test appel pgm
350:
351:     Panel.field.append(fld.newFieldFunc("cb02",4,76,             // Name , posx posy
352:                                         20,                      // width
353:                                         "",                      // text
354:                                         false,                   // tofill
355:                                         "comboFn02",             // Process for FUNC
356:                                         "required",              // error msg
357:                                         "select combo",          // help
358:                                         )
359:     ) catch unreachable ;
360:
361:
362:     // button-------------------------------------------------
363:     Panel.button.append(btn.newButton(
364:                         kbd.F3,                  // function
365:                         true,                    // show
366:                         false,                   // check field
367:                         "Exit"                   // title
368:                         )
```

```
369:     ) catch unreachable ;
370:
371:     Panel.button.append(btn.newButton(
372:                         kbd.F2,                 // function
373:                         true,                   // show
374:                         true,                   // check field
375:                         "test"                  // title
376:                         )
377:     ) catch unreachable ;
378:
379:     Panel.button.append(btn.newButton(
380:                         kbd.F4,                 // function
381:                         true,                   // show
382:                         true,                   // check field
383:                         "test window"           // title
384:                         )
385:     ) catch unreachable ;
386:
387:     Panel.button.append(btn.newButton(
388:                         kbd.F5,                 // function
389:                         true,                   // show
390:                         false,                  // check field
391:                         "Menu"                  // title
392:                         )
393:     ) catch unreachable ;
394:
395:     Panel.button.append(btn.newButton(
396:                         kbd.F8,                 // function
397:                         true,                   // show
398:                         false,                  // check control to Field
399:                         "Grid"                  // title
400:                         )
401:     ) catch unreachable ;
402:
403:     Panel.button.append(btn.newButton(
404:                         kbd.F12,                // function
405:                         true,                   // show
406:                         false,                  // check control to Field
407:                         "ClearPanel"            // title enrg record
408:                         )
409:     ) catch unreachable ;
410:     Panel.button.append(btn.newButton(
411:                         kbd.F24,                // function
412:                         true,                   // show
413:                         false,                  // check control to Field
414:                         "Refresh"               // title enrg record
```

```
415:                         )
416:     ) catch unreachable ;
417:     return Panel;
418: }
419:
420:
421:
422: pub fn Panel_Fmt0X() *pnl.PANEL {
423:
424:     //-----------------------------------------------
425:     // Panel
426:     // Name Panel, Pos X, Pos Y,
427:     // nbr Lines, nbr columns
428:     // Attribut Panel
429:     // Type frame, Attribut frame
430:     // Title Panel, Attribut: Title
431:     var Panel    : *pnl.PANEL = pnl.newPanelC("Fmt01",
432:                     1, 1,
433:                     8,
434:                     70,
435:                     forms.CADRE.line1,
436:                     "TEST WINDOW");
437:
438:     //-----------------------------------------------
439:     // Label
440:     // Name , pos X, pos Y,
441:     // Text , Attribut Text
442:     Panel.label.append(lbl.newLabel("free",2,2,"Text-Free.................:")
443:     ) catch unreachable ;
444:
445:     Panel.label.append(lbl.newLabel("full",3,2,"Text-Full.....protect.......:")
446:     ) catch unreachable ;
447:
448:     // button-------------------------------------------------
449:     Panel.button.append(btn.newButton(
450:                         kbd.F12,                // function
451:                         true,                   // show
452:                         false,                  // check field
453:                         "Return"                // title
454:                         )
455:     ) catch unreachable ;
456:     return Panel;
457: }
458: //-----------------------------------------------
459: //the menu is not double buffered it is not a Panel
460: pub fn Menu01() mnu.MENU {
```

```
461:     const  m01 = mnu.newMenu(
462:                     "Menu01",                  // name
463:                     2, 2,                      // posx, posy
464:                     mnu.CADRE.line1,           // type line fram
465:                     mnu.MNUVH.vertical,        // type menu vertical / horizontal
466:                     &.{"Open..",               // item
467:                     "List..",
468:                     "View..",
469:                     "Delete",
470:                     "New..",
471:                     "Src...",
472:                     "Exit.."}
473:                     );
474:     return m01;
475: }
476:
477:
478: // combo---------------------------------
479: fn comboFn01( vpnl : *pnl.PANEL , vfld :* fld.FIELD) void {
480:     var cellPos:usize = 0;
481:
482:     const Xcombo : *grd.GRID =  grd.newGridC(
483:                     "Combo01",
484:                     3, 75,
485:                     4 ,
486:                     grd.gridStyle,
487:                     grd.CADRE.line1,
488:                     );
489:     defer grd.freeGrid(Xcombo);
490:     defer grd.allocatorGrid.destroy(Xcombo);
491:
492:     grd.newCell(Xcombo,"Choix",15,grd.REFTYP.TEXT_FREE,term.ForegroundColor.fgGreen);
493:     grd.setHeaders(Xcombo) ;
494:
495:     grd.addRows(Xcombo , &.{"---"});
496:     grd.addRows(Xcombo , &.{"Famille"});
497:     grd.addRows(Xcombo , &.{"Amis"});
498:     grd.addRows(Xcombo , &.{"Professionel"});
499:     grd.addRows(Xcombo , &.{"Docteur"});
500:
501:     if (std.mem.eql(u8,vfld.text,"---") == true)         cellPos = 0;
502:     if (std.mem.eql(u8,vfld.text,"Famille") == true)     cellPos = 1;
503:     if (std.mem.eql(u8,vfld.text,"Amis") == true)        cellPos = 2;
504:     if (std.mem.eql(u8,vfld.text,"Professionel") == true) cellPos = 3;
505:     if (std.mem.eql(u8,vfld.text,"Docteur") == true)     cellPos = 4;
506:
```

```
507:
508:     // Interrogation
509:     var Gkey :grd.GridSelect = undefined ;
510:     defer Gkey.Buf.deinit();
511:
512:     Gkey =grd.ioCombo(Xcombo,cellPos);
513:     pnl.rstPanel(grd.GRID,Xcombo, vpnl);
514:
515:
516:     if ( Gkey.Key == kbd.esc )    return ;
517:     vfld.text = Gkey.Buf.items[0];
518:     return ;
519: }
520:
521: fn comboFn02( vpnl : *pnl.PANEL , vfld :* fld.FIELD) void {
522:     var cellPos:usize = 0;
523:
524:     const Xcombo : *grd.GRID =  grd.newGridC(
525:                                         "Combo02",
526:                                         4, 75,
527:                                         4 ,
528:                                         grd.gridStyle,
529:                                         grd.CADRE.line1,
530:                                         );
531:
532:     defer grd.freeGrid(Xcombo);
533:     defer grd.allocatorGrid.destroy(Xcombo);
534:
535:     grd.newCell(Xcombo,"Choix",15, grd.REFTYP.TEXT_FREE, term.ForegroundColor.fgGreen);
536:     grd.setHeaders(Xcombo) ;
537:
538:     grd.addRows(Xcombo , &.{"---"});
539:     grd.addRows(Xcombo , &.{"Informaticien"});
540:     grd.addRows(Xcombo , &.{"sportif"});
541:
542:     if (std.mem.eql(u8,vfld.text,"---") == true)             cellPos = 0;
543:     if (std.mem.eql(u8,vfld.text,"Informaticien") == true)   cellPos = 1;
544:     if (std.mem.eql(u8,vfld.text,"sportif") == true)      cellPos = 2;
545:
546:     // Interrogation
547:     var Gkey :grd.GridSelect = undefined ;
548:     defer Gkey.Buf.deinit();
549:
550:     Gkey =grd.ioCombo(Xcombo,cellPos);
551:     pnl.rstPanel(grd.GRID,Xcombo, vpnl);
552:
```

```
553:     if ( Gkey.Key == kbd.esc )  return ;
554:     vfld.text = Gkey.Buf.items[0];
555:     return ;
556: }
557:
558: /// run emun Function ex: combo
559: pub const FnEnum = enum {
560:     comboFn01,
561:     comboFn02,
562:     none,
563:
564:     pub fn run(self: FnEnum, vpnl : *pnl.PANEL, vfld: *fld.FIELD ) void {
565:         switch (self) {
566:             .comboFn01 => comboFn01(vpnl,vfld),
567:             .comboFn02 => comboFn02(vpnl,vfld),
568:             else => dsperr.errorForms(vpnl, Error.main_function_Enum_invalide),
569:         }
570:     }
571:
572:     fn searchFn ( vtext: [] const u8 ) FnEnum {
573:     var i    :usize = 0;
574:     const max :usize = @typeInfo(FnEnum).Enum.fields.len;
575:         while( i < max ) : (i += 1) {
576:         if ( std.mem.eql(u8, @tagName(@as(FnEnum,@enumFromInt(i))), vtext)) return @as(FnEnum,@enumFromInt(i));
577:         }
578:         return FnEnum.none;
579:
580:     }
581: };
582: var callFunc: FnEnum = undefined;
583:
584:
585: /// run emun Function ex: combo
586: pub const FnProg = enum {
587:     exCallpgm,
588:     none,
589:
590:     pub fn run(self: FnProg, vpnl : *pnl.PANEL, vfld: *fld.FIELD ) void {
591:         switch (self) {
592:             .exCallpgm=> {
593:
594:              mdl.callPgm("APPTERM",vfld.progcall)
595:                         catch |err | switch(err)  {
596:                                 mdl.ErrChild.Module_Invalid => {
597:                                 const msgerr  = std.fmt.allocPrint(utl.allocUtl,
598:                                 " module {s} invalide appeller service Informatique ",
```

```
599:                                    .{vfld.progcall}) catch unreachable;
600:                                    defer utl.allocUtl.free(msgerr);
601:                                    forms.debeug(9999,msgerr);
602:                                    },
603:                                    else => unreachable,
604:                            };
605:
606:                },
607:                else => dsperr.errorForms(vpnl, Error.main_function_Enum_invalide),
608:            }
609:        }
610:
611:        fn searchFn ( vtext: [] const u8 ) FnProg {
612:        var i     :usize = 0;
613:        const max :usize = @typeInfo(FnProg).Enum.fields.len;
614:            while( i < max ) : (i += 1) {
615:            if ( std.mem.eql(u8, @tagName(@as(FnProg,@enumFromInt(i))), vtext)) return @as(FnProg,@enumFromInt(i));
616:            }
617:            return FnProg.none;
618:
619:        }
620: };
621: var callProg: FnProg = undefined;
622:
623:
624: pub fn deinitWrk() void {
625:     term.deinitTerm();
626:     grd.deinitGrid();
627:     utl.deinitUtl();
628: }
629:
630: //test ---------- pas de sortie output
631:
632: test "test" {
633:     var infox : []const u8 = "";
634:     infox = utl.concatStr("Info : ", infox );
635:     std.debug.print("{s}",.{infox});
636: }
637:
638:
639:
640: // main---------------------------------
641: pub fn main() !void {
642:
643:     // open terminal and config and offMouse , cursHide->(cursor hide)
644:     term.enableRawMode();
```

```
645:     defer term.disableRawMode() ;
646:
647:     // define Panel
648:     var pFmt01 = Panel_Fmt01();
649:
650:
651:     var mMenu01:mnu.MENU = Menu01();
652:
653:
654:     // defines the receiving structure of the keyboard
655:     var Tkey : term.Keyboard = undefined ;
656:
657:     // work Panel-01
658:     term.resizeTerm(pFmt01.lines,pFmt01.cols);
659:
660:
661:     while (true) {
662:         // clean works
663:         term.deinitTerm();
664:         grd.deinitGrid();
665:         utl.deinitUtl();
666:
667:         Tkey.Key = pnl.ioPanel(pFmt01);
668:
669:         switch (Tkey.Key) {
670:
671:             .func => {
672:             callFunc = FnEnum.searchFn(pFmt01.field.items[pFmt01.idxfld].procfunc); // User clicks "increment"
673:             callFunc.run(pFmt01, &pFmt01.field.items[pFmt01.idxfld]);
674:             },
675:
676:             .call => {
677:             callProg = FnProg.searchFn(pFmt01.field.items[pFmt01.idxfld].progcall); // call programe ex: Exemple
678:             callProg.run(pFmt01, &pFmt01.field.items[pFmt01.idxfld]);
679:             },
680:
681:             .F2 => {
682:                 // test control chek field
683:                 pnl.msgErr(pFmt01,"le test de la saisie est OK");
684:             },
685:             .F4 => {
686:                 const pFmt0X = Panel_Fmt0X();
687:                 _= pnl.ioPanel(pFmt0X);
688:                 pnl.rstPanel(pnl.PANEL,pFmt0X, pFmt01);
689:                 pnl.freePanel(pFmt0X);
690:                 forms.allocatorForms.destroy(pFmt0X);
```

```
691:             },
692:             .F5 => {
693:                 const nitem = mnu.ioMenu(mMenu01,0);
694:                 pnl.rstPanel(mnu.MENU,&mMenu01, pFmt01);
695:                 std.debug.print("nÂ°item {}",.{nitem});
696:             },
697:             .F8 => {
698:                 var Gkey :grd.GridSelect = undefined ;
699:                 Gkey.Key = term.kbd.none;
700:                 Gkey.Buf = std.ArrayList([]const u8).init(grd.allocatorGrid);
701:
702:                 // Grid ----------------------------------------------------------------
703:                 var Grid01 : *grd.GRID =    grd.newGridC(
704:                             "Grid01",           // Name
705:                             20, 62,             // posx, posy
706:                             7,                  // numbers lines
707:                             grd.gridStyle,      // separator │ or   space
708:                             grd.CADRE.line1,    // type line 1
709:                             );
710:
711:
712:                 if (grd.countColumns(Grid01) == 0) {
713:
714:                     grd.newCell(Grid01,"ID",2,grd.REFTYP.UDIGIT,term.ForegroundColor.fgCyan) ;
715:                     grd.newCell(Grid01,"Name",15,grd.REFTYP.TEXT_FREE,term.ForegroundColor.fgYellow);
716:                     grd.newCell(Grid01,"animal",20,grd.REFTYP.TEXT_FREE,term.ForegroundColor.fgWhite) ;
717:                     grd.newCell(Grid01,"prix",10,grd.REFTYP.DECIMAL,term.ForegroundColor.fgWhite) ;
718:                     grd.setCellEditCar(&Grid01.cell.items[3],"â\202¬");
719:                     grd.newCell(Grid01,"HS",1,grd.REFTYP.SWITCH,term.ForegroundColor.fgRed) ;
720:                     //grd.newCell(&Grid01,"Password",10,grd.REFTYP.PASSWORD,term.ForegroundColor.fgGreen) ;
721:                     grd.setHeaders(Grid01);
722:                     grd.printGridHeader(Grid01);
723:                 }
724:
725:                 grd.resetRows(Grid01);
726:
727:                 grd.addRows(Grid01 , &.{"1", "Adam","Aigle","+1000.00","1","tictac"});
728:                 grd.addRows(Grid01 , &.{"2", "Eve", "poisson","-1001.00","1","tictac2"});
729:                 grd.addRows(Grid01 , &.{"3", "Rouge","Aigle","1002.00","0","tictac3"});
730:                 grd.addRows(Grid01 , &.{"4", "Bleu", "poisson","100.00","0","tictac"});
731:                 grd.addRows(Grid01 , &.{"5", "Bleu5", "poisson","100.00","0","tictac"});
732:                 grd.addRows(Grid01 , &.{"6", "Bleu6", "poisson","100.00","0","tictac"});
733:                 grd.addRows(Grid01 , &.{"7", "Bleu7", "poisson","100.00","1","tictac"});
734:                 grd.addRows(Grid01 , &.{"8", "Bleu8", "poisson","100.00","0","tictac"});
735:                 grd.addRows(Grid01 , &.{"9", "Bleu9", "poisson","100.00","0","tictac"});
736:                 grd.addRows(Grid01 , &.{"10", "Bleu10", "poisson","100.00","0","tictac"});
```

```
737:                grd.addRows(Grid01 , &.{"11", "Bleu11", "poisson","100.00","0","tictac"});
738:                grd.addRows(Grid01 , &.{"12", "Bleu12", "Canard","100,00","0","tictac"});
739:
740:                //grd.dltRows(&Grid01 , 5) catch |err| {dsperr.errorForms(err); return;};
741:                while (true ){
742:                    Gkey =grd.ioGrid(Grid01,true);
743:
744:                    if ( Gkey.Key == kbd.enter and pFmt01.idxfld == 0) {
745:                    fld.setText(pFmt01,0,Gkey.Buf.items[2]) catch |err| {dsperr.errorForms(pFmt01,err); return;};
746:                    // exemple key reccord hiden
747:                    //fld.setText(&pFmt01,0,Gkey.Buf.items[5]) catch |err| {dsperr.errorForms(err); return;};
748:                    break;
749:                    }
750:                    if ( Gkey.Key == kbd.esc) {
751:                    break;
752:                    }
753:
754:                    if (Gkey.Key    == kbd.pageDown) {
755:                    grd.resetRows(Grid01);
756:                    grd.addRows(Grid01 , &.{"13", "Bleu13", "poisson","100,00","0","tictac"});
757:                    grd.addRows(Grid01 , &.{"14", "Bleu14", "Vache","100,00","0","tictac"});
758:                    }
759:                    if (Gkey.Key    == kbd.pageUp) {
760:                    grd.resetRows(Grid01);
761:                    grd.addRows(Grid01 , &.{"1", "Adam","Aigle","1000,00","1","tictac"});
762:                    grd.addRows(Grid01 , &.{"2", "Eve", "poisson","1001,00","1","tictac"});
763:                    grd.addRows(Grid01 , &.{"3", "Rouge","Aigle","1002,00","0","tictac"});
764:                    grd.addRows(Grid01 , &.{"4", "Bleu", "poisson","100,00","0","tictac"});
765:                    grd.addRows(Grid01 , &.{"5", "Bleu5", "poisson","100,00","0","tictac"});
766:                    grd.addRows(Grid01 , &.{"6", "Bleu6", "poisson","100,00","0","tictac"});
767:                    grd.addRows(Grid01 , &.{"7", "Bleu7", "poisson","100,00","1","tictac"});
768:                    grd.addRows(Grid01 , &.{"8", "Bleu8", "poisson","100,00","0","tictac"});
769:                    grd.addRows(Grid01 , &.{"9", "Bleu9", "poisson","100,00","0","tictac"});
770:                    grd.addRows(Grid01 , &.{"10", "Bleu10", "poisson","100,00","0","tictac"});
771:                    grd.addRows(Grid01 , &.{"11", "Bleu11", "poisson","100,00","0","tictac"});
772:                    grd.addRows(Grid01 , &.{"12", "Bleu12", "Canard","100,00","0","tictac"});
773:                    }
774:                }
775:                pnl.rstPanel(grd.GRID,Grid01, pFmt01);
776:                // if you have several grids please do a freeGrid on exit and a reloadGrid on enter
777:                grd.freeGrid(Grid01);
778:                grd.allocatorGrid.destroy(Grid01);
779:                Gkey.Buf.deinit();
780:                grd.deinitGrid();
781:                // for debug control memoire in test CODELLDB
782:                // _= kbd.getKEY();
```

```
783:                 },
784:
785:                 .F12 => {
786:                 // function test clean
787:                     deinitWrk();
788:                     pnl.clearPanel(pFmt01);
789:                     pnl.printPanel(pFmt01);
790:                 },
791:                 .F24 => {
792:                 // function enrg file record
793:                     pnl.freePanel(pFmt01);
794:                     forms.deinitForms();
795:                     deinitWrk();
796:                     pFmt01 = Panel_Fmt01();
797:                     pnl.printPanel(pFmt01);
798:                 },
799:                 else => {},
800:             }
801:             if (Tkey.Key == kbd.F3) break; // end work
802:         }
803: }
```