Victima	Wingest carrera 2008
Compilador	Visual FoxPro 9
Herramientas	UniExtract, Refox XI+, UnRefox, vfp v9 portable
Fecha	Entre Enero y Junio de 2009
Cracker	ZiKaTRiZ
Dificultad	Newbie +

Hola, no recuerdo cuando termine, este proyecto. Hay dos formas de hacer esto, una es instalar el wingest, ejecutando el setup.exe y dejar que termine, y

la otra es: ejecutar UniExtract.exe, cargar el archivo setup.exe darle a aceptar, cuando termine, solo hay que colocar en la carpeta {app} todo lo que salga en la carpeta {sys}.

Utilizaré la segunda ya que no tengo ganas de instalar nada.

Comencemos:

Ejecutaremos carrera,1.exe o carrera,2.exe, que para el caso que nos ocupa es lo mismo.

Bueno crea todos los archivos de la base de datos y demás.

Para poder registrar este proyecto, debemos seguir varias pestañas:

Introducir valores y en el lugar de número de serie, dejarlo con un 1 o en su caso poner un 2 ó un 3, dependiendo de la cantidad de usuarios que lo utilicen.
Pulsemos siguiente y nos aparece la segunda pantalla

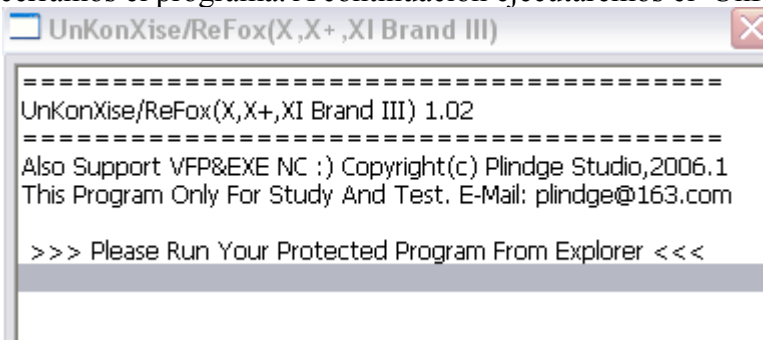
Las licencias de red están relacionadas con las diferentes soluciones.
Pulsemos siguiente y nos aparece el código de registro.
Si se selecciona Fax, busca la impresora para imprimirlo, ni caso, le dais a cancelar a la impresora y pulsáis siguiente al programa.

Veamos de donde sale el código: el “ 1 ” lo podemos encontrar en la primera pantalla de registro, el “ 3 ” es porque la solución Premium está seleccionada, el “ 15 “ son las licencias de red, el numero

hexadecimal " 07FFFFDB " son los módulos de gestión, pasar este numero a binario y lo veréis y por ultimo " 768 ", es el tiempo que ha transcurrido desde medianoche hasta que se ejecuta la aplicación.

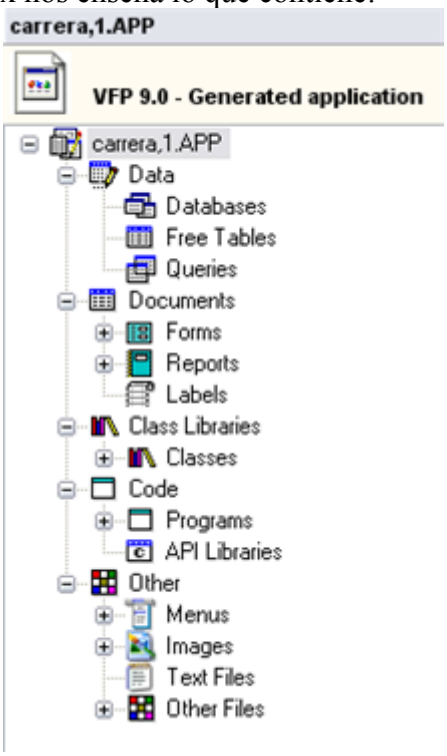
Pulsando siguiente, nos queda la zona en donde introducir el contracódigo, si es correcto nos felicitará y si es erróneo nos dirá, " Lo siento ".

demos a cancelar, y cerramos el programa. A continuación ejecutaremos el UnRefox



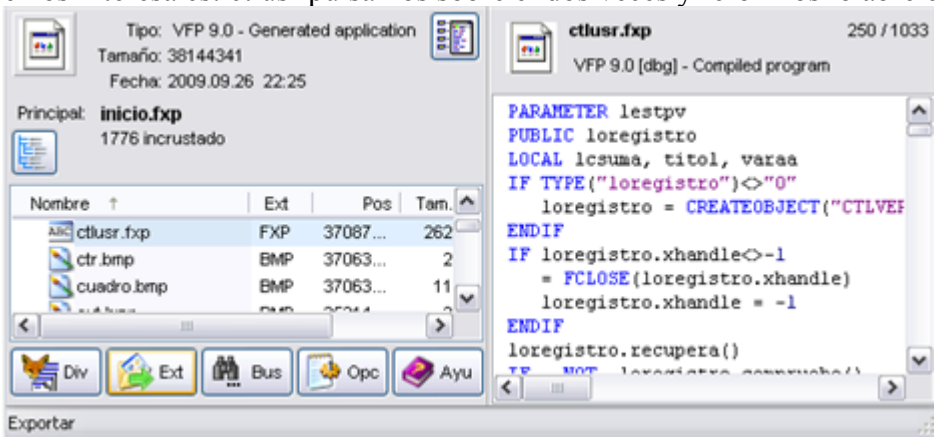
y como el mismo nos pide ejecutaremos el carrera,1.exe dejando que termine su trabajo. Ahora tenemos el ejecutable de vfp con el nombre carrera,1.app , este lo abriremos con Refox XI+.

Seleccionamos el archivo, y refox nos enseña lo que contiene:

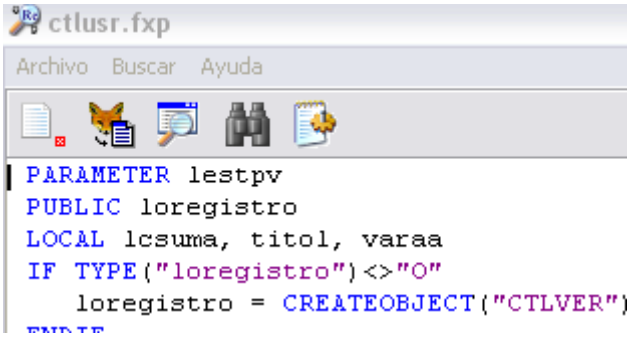


Aquí pulsamos sobre el + de Programs y tendremos el código fuente en diferentes archivos.

El primero que nos interesa es: ctlusr pulsamos sobre él dos veces y refox nos lo abre en la pantalla



Pulsando dos veces sobre las lineas del código fuente, refox nos muestra otra pantalla con todo el código completo de este archivo.



Desde esta otra pantalla, se ve mucho mejor.

variable publica “ loregistro “, esto me suena, vamos por buen camino.

Buscando más abajo encontramos esta linea de código:

loregistro.xcontra = generacodigo(ALLTRIM(loregistro.xcodigo))

loregistro.xcodigo es el código que nos da el programa.

ALLTRIM limpia espacios en blanco de inicio y fin de código.

Bien colocare en color rojo las llamadas a las diferentes funciones, para que nadie se pierda.

Generacodigo es la función que principalmente nos interesa.

```
FUNCTION GeneraCodigo
 PARAMETER ccodigo
 LOCAL cclave, ccharvalidos, nsuma, i, nlongitudcontraclave, cchar, cletra,
cletra2
 LOCAL aresul, contador, distancia, nposini, ccharini, ccontraclave, npos,
nsalto
 cclave = sha1(ccodigo)
 ccharvalidos = ""
 cclave = cclave+sha1(ccharvalidos, @ccharvalidos)
 ccodigo = ALLTRIM(ccodigo)
 nsuma = 0
 FOR i = 1 TO LEN(ALLTRIM(ccodigo))
    nsuma = nsuma+ASC(SUBSTR(ccodigo, i, 1))
 ENDFOR
 ccodigo = ccodigo+ALLTRIM(STR(nsuma))
 nlongitudcontraclave = 24
 cchar = "@"
 IF LEN(cclave)>LEN(ccodigo)
    ccodigo = PADR(ccodigo, LEN(cclave), cchar)
 ELSE
    cclave = PADR(cclave, LEN(ccodigo), cchar)
 ENDIF
 cletra = ""
 cletra2 = ""
 DIMENSION aresul(1)
 contador = 0
 FOR i = 1 TO LEN(cclave)
    cletra = SUBSTR(cclave, i, 1)
    IF i+1>LEN(cclave)
       cletra2 = SUBSTR(cclave, LEN(cclave)-i+1, 1)
    ELSE
       cletra2 = SUBSTR(cclave, i+1, 1)
    ENDIF
    contador = contador+1
    DIMENSION aresul(contador)
    distancia = 0
    distancia = distancia+(ABS(ASC(cletra)-ASC(cletra2)))
    aresul(contador) = distancia
 ENDFOR
 AINS(aresul, 1)
 aresul(1) = 0
 nposini = ROUND(RAND()*LEN(ccharvalidos), 0)
 ccharini = SUBSTR(ccharvalidos, nposini, 1)
 ccharvalidos = RIGHT(ccharvalidos, LEN(ccharvalidos)-
nposini+1)+LEFT(ccharvalidos, nposini-1)
 ccontraclave = ""
 npos = 1
 FOR i = 1 TO ALEN(aresul)
    nsalto = aresul(i)
```

```
    npos = npos+nsalto
    IF npos>LEN(ccharvalidos)
       npos = nsalto+1
    ENDIF
    cletra = SUBSTR(ccharvalidos, npos, 1)
    ccontraclave = ccontraclave+cletra
 ENDFOR
 RETURN PADR(SUBSTR(ccontraclave, 1, nlongitudcontraclave),
nlongitudcontraclave, "0")
ENDFUNC
```

En la función generacódigo tenemos otra función que es sha1 y dentro de esta dos funciones mas: hsh y pddmsg , después en hsh y pddmsg abran llamadas a otras funciones que por supuesto están en el archivo proce.

Antes de continuar, debo deciros que la variable lasletras, la correcta es la "9JMOW6HITYKZ42ENXR1A17QGLB05DVCS38D", por lo tanto a la hora de crear el keygen deberemos dejar solamente la variable lasletras = "9JMOW6HITYKZ42ENXR1A17QGLB05DVCS38D"

```
FUNCTION Sha1( hash1_mensage, lasletras)
 IF parcual="S"
    lasletras = "2UIHYANJ7EQ9V5WMK148LTC3S0XODFGBR6PZ"
 ELSE
    lasletras = "9JMOW6HITYKZ42ENXR1A17QGLB05DVCS38D"
 ENDIF
 IF TYPE('hash1_mensage')='C'
    hash1_mensage = hsh(pddmsg(hash1_mensage))
    RELEASE ALL LIKE 'hash_*'
    RETURN hash1_mensage
 ELSE
    RETURN .F.
 ENDIF
ENDFUNC
```

Vamos a dividirlo y decompilar el archivo para verlo mejor.

Pulsaremos sobre el boton

y despues

pulsamos en aceptar y dejamos que termine.

Una vez que tenemos todos los archivos en disco ya podemos cerrar refox.

Buscaremos el archivo proce.prg lo abriremos con el bloc de notas y buscaremos todas las funciones que nos hacen falta, para generar el keygen.

```
FUNCTION Hsh( hash_msgh)
 LOCAL hash_r, hash_t, hash_k1, hash_k2, hash_k3, hash_k4, hash_temp, hash_232, hash_k,
```

```
hash_f
 LOCAL hash_a, hash_b, hash_c, hash_d, hash_e, hash_h1, hash_h2, hash_h3, hash_h4,
hash_h5
 LOCAL hash_blokes, hash_cw
 hash_k1 = "5A827999"
 hash_k2 = "6ED9EBA1"
 hash_k3 = "8F1BBCDC"
 hash_k4 = "CA62C1D6"
 hash_h1 = "67452301"
 hash_h2 = "EFCDAB89"
 hash_h3 = "98BADCFE"
 hash_h4 = "10325476"
 hash_h5 = "C3D2E1F0"
 hash_232 = 4294967296
 hash_blokes = LEN(hash_msgh)/128
 LOCAL hash_m(hash_blokes), hash_w(80)
 hash_cw = 0
 FOR hash_bk = 1 TO hash_blokes
    hash_m[hash_bk] = SUBSTR(hash_msgh, 1+(hash_cw*128), 128)
    hash_cw = hash_cw+1
 ENDFOR
 RELEASE hash_msgh
 FOR hash_bk = 1 TO hash_blokes
    hash_a = hash_h1
    hash_b = hash_h2
    hash_c = hash_h3
    hash_d = hash_h4
    hash_e = hash_h5
    hash_cw = 0
    FOR hash_cc = 1 TO 16
       hash_w[hash_cc] = SUBSTR(hash_m(hash_bk), 1+(hash_cw*8), 8)
       hash_cw = hash_cw+1
    ENDFOR
    FOR hash_cc = 17 TO 80
       hash_w[hash_cc] = rotar(bxor32(bxor32(hash_w(hash_cc-3), hash_w(hash_cc-8)),
bxor32(hash_w(hash_cc-14), hash_w(hash_cc-16))), 1)
    ENDFOR
    FOR hash_t = 1 TO 80
       DO CASE
          CASE hash_t<=20
             hash_k = hash_k1
             hash_f = f1(hash_b, hash_c, hash_d)
          CASE hash_t<=40
             hash_k = hash_k2
             hash_f = f2(hash_b, hash_c, hash_d)
          CASE hash_t<=60
             hash_k = hash_k3
             hash_f = f3(hash_b, hash_c, hash_d)
          CASE hash_t<=80
             hash_k = hash_k4
             hash_f = f2(hash_b, hash_c, hash_d)
       ENDCASE
       hash_temp = sumas(sumas(sumas(rotar(hash_a, 5), hash_e, hash_232),
sumas(hash_w(hash_t), hash_f, hash_232), hash_232), hash_k, hash_232)
       hash_e = hash_d
       hash_d = hash_c
       hash_c = rotar(hash_b, 30)
       hash_b = hash_a
       hash_a = hash_temp
    ENDFOR
    hash_h1 = sumas(hash_h1, hash_a, hash_232)
    hash_h2 = sumas(hash_h2, hash_b, hash_232)
    hash_h3 = sumas(hash_h3, hash_c, hash_232)
    hash_h4 = sumas(hash_h4, hash_d, hash_232)
    hash_h5 = sumas(hash_h5, hash_e, hash_232)
 ENDFOR
 hash_r = TRIM(" ")
 hash_r = hash_h1+hash_h2+hash_h3+hash_h4+hash_h5
 RETURN hash_r
ENDFUNC
**
```

```
FUNCTION F1( hash_f1_b, hash_f1_c, hash_f1_d)
 LOCAL hash_f1_r
 hash_f1_r = bor32(band32(hash_f1_b, hash_f1_c), band32(bnot32(hash_f1_b), hash_f1_d))
 RETURN hash_f1_r
ENDFUNC
**
FUNCTION F2( hash_f2_b, hash_f2_c, hash_f2_d)
 LOCAL hash_f2_r
 hash_f2_r = bxor32(bxor32(hash_f2_b, hash_f2_c), hash_f2_d)
 RETURN hash_f2_r
ENDFUNC
**
FUNCTION F3( hash_f3_b, hash_f3_c, hash_f3_d)
 LOCAL hash_f3_r
 hash_f3_r = bor32(bor32(band32(hash_f3_b, hash_f3_c), band32(hash_f3_b, hash_f3_d)),
band32(hash_f3_c, hash_f3_d))
 RETURN hash_f3_r
ENDFUNC
**
FUNCTION sumas( hash_su_x, hash_su_y, hash_su_c)
 LOCAL hash_suma_r
 hash_suma_r = MOD(hexdec(hash_su_x)+hexdec(hash_su_y), hash_su_c)
 RETURN wordsh(hash_suma_r)
ENDFUNC
**
FUNCTION band32( hash_ba_cad1, hash_ba_cad2)
 hash_ba_cad1 = wordsb2(hexdec(hash_ba_cad1))
 hash_ba_cad2 = wordsb2(hexdec(hash_ba_cad2))
 LOCAL hash_band32_r
 hash_band32_r = TRIM(" ")
 FOR hash_ba_i = 1 TO LEN(hash_ba_cad1)
    IF SUBSTR(hash_ba_cad1, hash_ba_i, 1)="1" .AND. SUBSTR(hash_ba_cad2, hash_ba_i,
1)="1"
       hash_band32_r = hash_band32_r+"1"
    ELSE
       hash_band32_r = hash_band32_r+"0"
    ENDIF
 ENDFOR
 hash_band32_r = wordsh(lbin(hash_band32_r))
 RETURN hash_band32_r
ENDFUNC
**
FUNCTION bor32( hash_br_cad1, hash_br_cad2)
 hash_br_cad1 = wordsb2(hexdec(hash_br_cad1))
 hash_br_cad2 = wordsb2(hexdec(hash_br_cad2))
 LOCAL hash_bor32_r
 hash_bor32_r = TRIM(" ")
 FOR hash_br_i = 1 TO LEN(hash_br_cad1)
    IF SUBSTR(hash_br_cad1, hash_br_i, 1)="0" .AND. SUBSTR(hash_br_cad2, hash_br_i,
1)="0"
       hash_bor32_r = hash_bor32_r+"0"
    ELSE
       hash_bor32_r = hash_bor32_r+"1"
    ENDIF
 ENDFOR
 hash_bor32_r = wordsh(lbin(hash_bor32_r))
 RETURN hash_bor32_r
ENDFUNC
**
FUNCTION bxor32( hash_bx_cad1, hash_bx_cad2)
 LOCAL hash_bxor32_r, hash_bx_1, hash_bx_2
 hash_bx_cad1 = wordsb2(hexdec(hash_bx_cad1))
 hash_bx_cad2 = wordsb2(hexdec(hash_bx_cad2))
 hash_bxor32_r = TRIM(" ")
 hash_bx_1 = TRIM(" ")
 hash_bx_2 = TRIM(" ")
 FOR hash_bx_i = 1 TO LEN(hash_bx_cad1)
    hash_bx_1 = SUBSTR(hash_bx_cad1, hash_bx_i, 1)
    hash_bx_2 = SUBSTR(hash_bx_cad2, hash_bx_i, 1)
    DO CASE
       CASE hash_bx_1="0" .AND. hash_bx_2="0"
```

```
            hash_bxor32_r = hash_bxor32_r+"0"
         CASE hash_bx_1="1" .AND. hash_bx_2="0"
            hash_bxor32_r = hash_bxor32_r+"1"
         CASE hash_bx_1="0" .AND. hash_bx_2="1"
            hash_bxor32_r = hash_bxor32_r+"1"
         CASE hash_bx_1="1" .AND. hash_bx_2="1"
            hash_bxor32_r = hash_bxor32_r+"0"
      ENDCASE
   ENDFOR
   hash_bxor32_r = wordsh(lbin(hash_bxor32_r))
   RETURN hash_bxor32_r
ENDFUNC
**
FUNCTION bnot32( hash_bo_cadena)
   hash_bo_cadena = wordsb2(hexdec(hash_bo_cadena))
   LOCAL bnot32_r
   hash_bnot32_r = TRIM(" ")
   FOR hash_bo_i = 1 TO LEN(hash_bo_cadena)
      IF SUBSTR(hash_bo_cadena, hash_bo_i, 1)="1"
         hash_bnot32_r = hash_bnot32_r+"0"
      ELSE
         hash_bnot32_r = hash_bnot32_r+"1"
      ENDIF
   ENDFOR
   hash_bnot32_r = wordsh(lbin(hash_bnot32_r))
   RETURN hash_bnot32_r
ENDFUNC
**
FUNCTION rotar( hash_rt_cadena, hash_rt_num)
   LOCAL hash_rt_lcad
   hash_rt_cadena = wordsb2(hexdec(hash_rt_cadena))
   hash_rt_lcad = LEN(hash_rt_cadena)
   RETURN wordsh(lbin(SUBSTR(hash_rt_cadena, hash_rt_num+1, hash_rt_lcad-hash_rt_num)
+SUBSTR(hash_rt_cadena, 1, hash_rt_num)))
ENDFUNC
**
FUNCTION binl( hash_bl_num_c)
   LOCAL hash_binl_r, hash_bl_num_a
   hash_bl_num_a = hash_bl_num_c
   hash_binl_r = TRIM(" ")
   IF hash_bl_num_a=0
      hash_binl_r = "00000000"
   ELSE
      IF hash_bl_num_a=1
         hash_binl_r = "00000001"
      ELSE
         DO WHILE hash_bl_num_a>=2
            hash_binl_r = ALLTRIM(STR(MOD(hash_bl_num_a, 2)))+hash_binl_r
            hash_bl_num_a = hash_bl_num_a/2
            hash_bl_num_a = INT(hash_bl_num_a)
         ENDDO
         hash_binl_r = "1"+hash_binl_r
      ENDIF
   ENDIF
   RETURN hash_binl_r
ENDFUNC
**
FUNCTION DecHex( hash_dh_num)
   IF hash_dh_num<10
      RETURN ALLTRIM(STR(hash_dh_num))
   ELSE
      LOCAL hash_dechex_r, hash_dh_c
      LOCAL hash_dh_nhex(16)
      hash_dh_nhex[11] = "A"
      hash_dh_nhex[12] = "B"
      hash_dh_nhex[13] = "C"
      hash_dh_nhex[14] = "D"
      hash_dh_nhex[15] = "E"
      hash_dh_nhex[16] = "F"
      IF hash_dh_num<16
         hash_dechex_r = hash_dh_nhex(hash_dh_num+1)
```

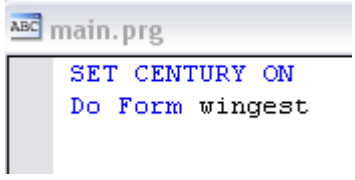
```
      RETURN hash_dechex_r
   ELSE
      hash_dh_c = 1
      FOR hash_dh_i = 0 TO 9
         hash_dh_nhex[hash_dh_c] = ALLTRIM(STR(hash_dh_i))
         hash_dh_c = hash_dh_c+1
      ENDFOR
      IF hash_dh_num<=255
         hash_dechex_r = hash_dh_nhex(1+INT(hash_dh_num/16))
+hash_dh_nhex(1+MOD(hash_dh_num, 16))
      ELSE
         LOCAL hash_dh_n
         hash_dh_n = hash_dh_num
         hash_dechex_r = TRIM(" ")
         DO WHILE hash_dh_n>=16
            hash_dechex_r = hash_dh_nhex(1+MOD(hash_dh_n, 16))+hash_dechex_r
            hash_dh_n = hash_dh_n/16
            hash_dh_n = INT(hash_dh_n)
         ENDDO
         hash_dechex_r = hash_dh_nhex(1+hash_dh_n)+hash_dechex_r
      ENDIF
      RETURN hash_dechex_r
   ENDIF
 ENDIF
ENDFUNC
**
FUNCTION HexDec( hash_hd_num)
 LOCAL hash_hexdec_r, hash_hd_nascii, hash_hd_exp
 hash_hexdec_r = 0
 hash_hd_exp = 0
 FOR hash_hd_i = LEN(hash_hd_num) TO 1 STEP -1
    hash_hd_nascii = ASC(SUBSTR(UPPER(hash_hd_num), hash_hd_i, 1))
    DO CASE
       CASE hash_hd_nascii>47 .AND. hash_hd_nascii<58
          hash_hexdec_r = hash_hexdec_r+((hash_hd_nascii-48)*(16**hash_hd_exp))
       CASE hash_hd_nascii>64 .AND. hash_hd_nascii<71
          hash_hexdec_r = hash_hexdec_r+((hash_hd_nascii-55)*(16**hash_hd_exp))
    ENDCASE
    hash_hd_exp = hash_hd_exp+1
 ENDFOR
 RETURN hash_hexdec_r
ENDFUNC
**
FUNCTION lbin( hash_lb_num)
 LOCAL hash_lbin_r, hash_lb_exp
 hash_lbin_r = 0
 hash_lb_exp = 0
 FOR hash_lb_i = LEN(hash_lb_num) TO 1 STEP -1
    hash_lbin_r = hash_lbin_r+(VAL(SUBSTR(hash_lb_num, hash_lb_i, 1))*(2**hash_lb_exp))
    hash_lb_exp = hash_lb_exp+1
 ENDFOR
 RETURN hash_lbin_r
ENDFUNC
**
FUNCTION Pddmsg( hash_pdd_msg)
 LOCAL hash_pddmsg_r, hash_pdd_bag, hash_pdd_lmsg, hash_pdd_lbmsg, hash_pdd_mlbmsg,
hash_pdd_bpag
 LOCAL hash_pdd_lmsgh, hash_pdd_lmsgag, hash_pdd_bmsg, hash_pdd_bpdd, hash_pdd_c
 hash_pdd_bpdd = ""
 hash_pdd_bmsg = TRIM(" ")
 hash_pdd_lmsg = LEN(hash_pdd_msg)
 FOR hash_pdd_i = 1 TO hash_pdd_lmsg
    hash_pdd_bmsg = hash_pdd_bmsg+wordsb(ASC(SUBSTR(hash_pdd_msg, hash_pdd_i, 1)))
 ENDFOR
 hash_pdd_lbmsg = LEN(hash_pdd_bmsg)
 hash_pdd_lmsgh = dechex(hash_pdd_lmsg*8)
 IF LEN(hash_pdd_lmsgh)<16
    FOR hash_pdd_i = 1 TO 16-LEN(hash_pdd_lmsgh)
       hash_pdd_lmsgh = "0"+hash_pdd_lmsgh
    ENDFOR
 ENDIF
```

```
 hash_pdd_bmsg = hash_pdd_bmsg+"1"
 hash_pdd_lbmsg = hash_pdd_lbmsg+1
 hash_pdd_mlbmsg = MOD(hash_pdd_lbmsg, 512)
 IF hash_pdd_mlbmsg<>448
    hash_pdd_bhash_pdd = ""
    IF hash_pdd_mlbmsg<448
       hash_pdd_bpag = 448-hash_pdd_mlbmsg
    ELSE
       hash_pdd_bpag = 512-hash_pdd_mlbmsg+448
    ENDIF
    FOR hash_pdd_p = 1 TO hash_pdd_bpag
       hash_pdd_bpdd = hash_pdd_bpdd+"0"
    ENDFOR
    hash_pdd_bmsg = hash_pdd_bmsg+hash_pdd_bpdd
 ENDIF
 hash_pdd_c = 0
 hash_pddmsg_r = TRIM(" ")
 FOR hash_pdd_i = 1 TO LEN(hash_pdd_bmsg)/32
    hash_pddmsg_r = hash_pddmsg_r+wordsh(lbin(SUBSTR(hash_pdd_bmsg, 1+(32*hash_pdd_c),
32)))
    hash_pdd_c = hash_pdd_c+1
 ENDFOR
 hash_pddmsg_r = hash_pddmsg_r+hash_pdd_lmsgh
 RETURN hash_pddmsg_r
ENDFUNC
**
FUNCTION wordsb( hash_wb_nbin)
 LOCAL hash_wordsb_r, hash_wb_ceros
 hash_wb_ceros = "0000000"
 hash_wordsb_r = binl(hash_wb_nbin)
 IF MOD(LEN(hash_wordsb_r), 8)<>0
    hash_wordsb_r = SUBSTR(hash_wb_ceros, 1, 8-MOD(LEN(hash_wordsb_r), 8))+hash_wordsb_r
 ENDIF
 RETURN hash_wordsb_r
ENDFUNC
**
FUNCTION wordsh( hash_wh_nhex)
 LOCAL hash_wordsh_r, hash_wh_ceros
 hash_wh_ceros = "00000000"
 hash_wordsh_r = dechex(hash_wh_nhex)
 IF MOD(LEN(hash_wordsh_r), 8)<>0
    hash_wordsh_r = SUBSTR(hash_wh_ceros, 1, 8-MOD(LEN(hash_wordsh_r), 8))+hash_wordsh_r
 ENDIF
 RETURN hash_wordsh_r
ENDFUNC
**
FUNCTION wordsb2( hash_wb2_nbin)
 LOCAL hash_wordsb_r, hash_wb2_ceros
 hash_wb2_ceros = "00000000000000000000000000000000"
 hash_wordsb2_r = binl(hash_wb2_nbin)
 IF LEN(hash_wordsb2_r)<32
    hash_wordsb2_r = SUBSTR(hash_wb2_ceros, 1, 32-LEN(hash_wordsb2_r))+hash_wordsb2_r
 ENDIF
 RETURN hash_wordsb2_r
ENDFUNC
**
```

Una vez que ya tenemos todas las funciones necesarias, solo nos queda construir el keygen.

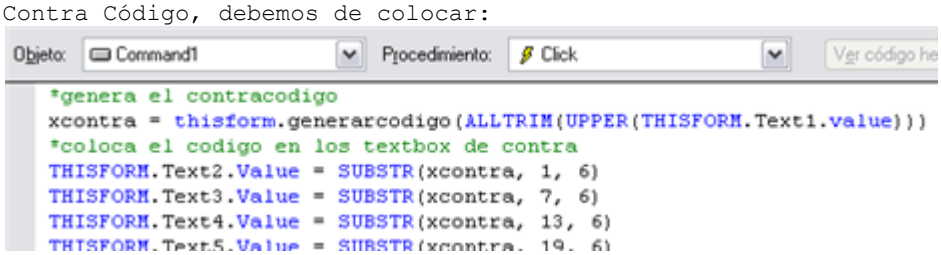
Hay que colocar la palabra THISFORM en todas las llamadas a los procedimientos, ya que si no se pierde y provoca errores.

Creamos el proyecto y en Codigo, Programas, Main colocamos:

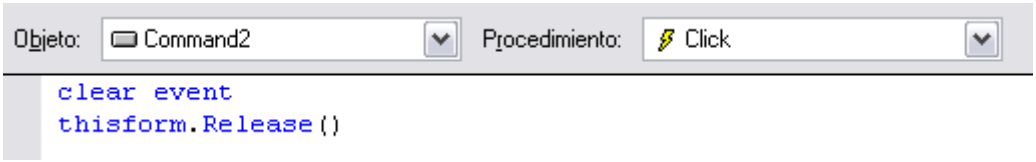


Dibujamos el formulario:

En el botón Contra Código, debemos de colocar:

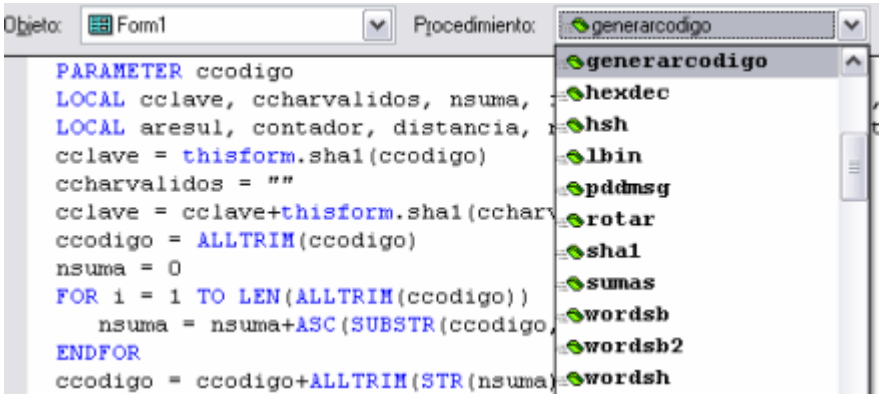


En el botón Salir colocaremos:

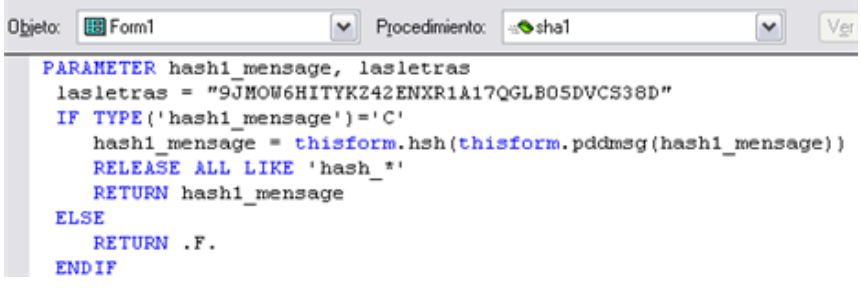


Crearemos las funciones de la siguiente manera:

En vfp, barra del menú Formulario, Nuevo método, le ponemos nombre he introducimos el código que ya hemos ido localizando.



No olvidarse que la función sha1 debe quedar de la siguiente manera:



Por ultimo
el proyecto terminado:

PD: Es posible que tengáis razón,

```
varaa = "Ya sabemos que sois muy buenos crackeando el wingest"
varaa = "pero no creeis que deberias dedicaros a algo un pelin mas dificil"
varaa = "por que nosotros somos pardillos pero crackear a pardillos no es muy"
varaa = "inteligente. "
```

pero no sabéis la cantidad de cosas que he aprendido.

Gracias especiales a los componentes de CracksLatinos.

ZiKaTRiZ.