به نام خدای بر حق .....

مجموعه مقالات آموزشی دلفی فروم تیم امنیتی ویرانگر (هکرهای جوان ایران)

تهیه فایل : پیمان شاهکار(Desperado)

Email : Peyman.shahkar@Gmail.com

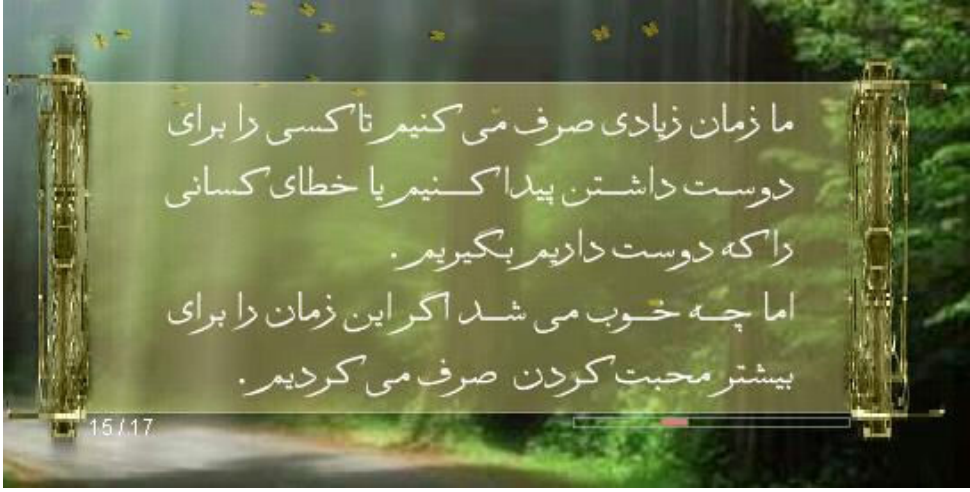
Yahoo Id : Peyman_shahkar

۱۳۸۷/۶/۸

www.Desperado.coo.ir

با تشکر از تمامی اعضای فروم تیم امنیتی ویرانگر که برای پیشرفت این تیم زحمت می کشند

تقدیم به پویندگان راه علم .....



Virangar Under
Gr0und Security
Team
Www.Virangar.net

نکات مفید برای کار در محیط دلفی

محیط دلفی برای برنامه نویسی یکی از بهترین محیطهای برنامه نویسی است گذشته از کارکرد داخلی و کمپایلر آن که بسیار قوی و سریع است، محیط آن یعنی IDE آنهم قدرت بسیار زیادی دارد که باعث شده یکی از بهترین ادیتورها باشد. در این مقاله من سعی بر این داشته ام تا با ارائه یک سری از نکات و کلیدهای میانبر که می توانند برای کار در دلفی بسیار مفید و کارا باشند، کمک کنم تا شما بتوانید با قدرت بیشتر به برنامه نویسی و کار در این محیط قدرتمند ادامه دهید.

در قسمت اول مقاله که در حال حاضر در مقابل شماست من یک سری از کلیدهای میانبر و ترکیبی مورد استفاده در IDE دلفی را بصورت لیست وار و همراه یک توضیح کوچک آورده ام. با توجه به اینکه این مطالب حاصل تجربه های خودم در کار با دلفی (از ۱ تا ۷) بوده ممکنه که یک سری موارد دیگری هم باشد که من تا حالا برخورد نداشتم که دوستان عزیز میتونند به این لیست اضافه کنند و نام خود را در انتهای این مقاله ذکر کنند و حتما یک نسخه از آن را برای من ارسال کنند.

امیدوارم که این سری مقالات با کمک شما در اثر مرور زمان بهتر و مفید تر شود.
دوستان عزیز برنامه نویس ممکنه که شما مدتها با دلفی مشغول برنامه نویسی بوده باشید اما من یقین دارم که در این لیست نکات و روشهای جدیدی را خواهید آموخت.

قسمت اول - کلیدهای میانبر و ترکیبی:

جستجو در متن بصورت مستقیم:
برای اینکار کلیدهای Ctrl+E را بفشارید و بدنبال آن شروع به تایپ کلمه مورد نظر کنید نتیجه آن را خود ببینید. برای اینکه به کلمه بعدی بروید کافیست کلید F3 را بزنید.

ایجاد فرورفتگی در کد:
بعضی اوقات - که خیلی هم پیش می‌آید - لازم است که یک مقداری از متن را بصورت بلوک شده به جلو و یا عقب ببریم. منظور دندانه دار کردن متن است که به خوانایی برنامه کمک می کند. برای اینکار می تونید از کلید Ctrl +Shift+I برای جلو بردن و Ctrl+Shift+U برای عقب برگرداندن متن بلوک شده استفاده کنید.

پرش به قسمت تعریف یک شی (Object):
برای اینکه ببنید شی مورد نظرتون (از قبیل VCL, Procedure, Function,...) در کجا و چطور تعریف شده می توانید کلید Crtl رو پایین نگه داشته و روی شی مورد نظر Click کنید.

برای تغییر حالت کاراکترها:
شما می توانید یک قسمت از متن (که ممکن است با حروف بزرگ و یا کوچک تایپ شده باشد) را انتخاب کنید و با زدن کلیدهای Ctrl+o+u به ترتیب تمامی حروف کوچک آن قسمت از متن را به حروف بزرگ و تمامی حروف بزرگ آنرا به حروف کوچک تبدیل کنید.

برای تعییر حالت یک کلمه نیز میتوانید روی کلمه مورد نظر رفته و کلیدهای Ctrl+k+f برای بزرگ کردن و کلیدهای ctrl+k+e را برای کوچک کردن حروف آن کلمه بکار برد.

درست کردن ماکرو متنی:

این امکان بسیار مفید است و می توانید بسیاری از کارهای نوشتاری را کاهش دهد با اینکار شما میتوانید یک سری از کارهای تکراری که روی متون انجام می دهید را بصورت ماکرو در آورده و از آنها به راحتی استفاده کنید. برای شروع به ضبط ماکرو کلیدهای ctrl+shift+r را بفشارید و آن سری کارهایی را که می خواهید را انجام دهید و سپس برای اینکه به کار ضبط ماکرو پایان دهید کلیدهای ctrl+shift+r را دوباره بزنید. حال برای استفاده از ماکرو کافیست در هر جا که لازم بود کلیدهای Ctrl+Shift+P را بفشارید.

انتخاب متن بصورت مربعی:

اگر شما از کهنه کارهای کامپیوتر باشید حتما از زمان داس یادتون هست که برنامه ای بود به نام PE2 که یکی از امکانات بسیار جالبش این بود که یک مربع از متن رو میتوانستین انتخاب

کنید و آنرا کپی یا حذف کنید. بله درست متوجه شدید در محیط دلفی هم شما اینکار را میتوانید انجام دهید اما نه به مشکلی PE2 بلکه اینکار را میتوانید فقط با گرفتن کلید Alt و کشیدن

موس روی متن انجام دهید. هر چند ممکن است در نگاه اول زیاد این امکان مفید به نظر نیاید ولی بعضی وقتهای خیلی کار را راحت میکنه، که حتماً تجربه خواهید کرد.

گذاشتن علامت روی متن:

این کار که به BookMark معروف است بسیار مفید و کارا می باشد. در هنگامی که شما روی قسمتی از متن برنامه کار میکنید و می خواهید به یک قسمت دیگر بروید ممکن

است برای برگشتن به مکان اول خود کمی مشکل پیدا کنید. ولی شما میتوانید با زدن چند دکمه به محل مورد نظرتون باز گردید. برای اینکار در خطی که قصد دارید علامت بگذارید کلیدهای Ctrl+Shift+0..9 را بفشارید. منظور اینست که کلیدهای ctrl+Shift را نگه دارید و یکی از اعداد ۰ تا ۹ را وارد کنید تا آن خط به همان شماره علامت گذاری شود و سپس هر جا که خواستید بروید و سپس هر بار که کلید Ctrl را نگه دارید و شماره مورد نظر را وارد کنید به همان خط باز خواهید گشت. البته توجه داشته باشید که فقط می توانید ۱۰ خط را با این روش علامت گذاری بکنید و برای برداشتن علامت ها کافیست روی همان خط دوباره کلید Ctrl+shift و شماره‌ای که برای آن خط وارد کرده اید را بفشارید با اینکار علامت آن خط برداشته می شود.

ایجاد کلاس مورد نظر :

شما هنگامی که در قسمت Private و یا Public یک type، روال یا تابع درست کردید لازم دارید که قسمتی را برای قرار دادن کدهای مربوط به آن روال یا تابع را ایجاد کنید. برای اینکار شما پس از اینکه نام تابع را تایپ کردید می توانید کلیدهای Ctrl+Shift+C را فشار دهید تا دلفی یک قسمت برای نوشتن کدهای مورد نظرتان ایجاد کند.

ظاهر کردن پنجره Code insight :

شما حتما به اهمیت و مفید بودن این قسمت دلفی واقفید که در هنگام کد نویسی تا چه حد می تواند کارها را راحت کند. بله در هنگام وارد کردن کدها بعد از وارد کردن نام یک کلاس و یا Object با زدن یک نقطه (.) پنجره Code Insight ظاهر می شود. حال در بعضی وقتها شما ممکن است که نقطه را قبلا وارد کرده باشید و یا در مواقع دیگر این پنجره ظاهر نشود. در

این صورت برای اینکه پنجره را ظاهر کنید باید دوباره نقطه را وارد کنید ولی راه اسانتری هم وجود دارد و آن اینست که کلیدهای Ctrl+Speacebar را فشار دهید.

ظاهر کردن پنجره Code Parameter:

همانند بالا در هنگام ظاهر شدن Hint مربوط به راهنمای توابع که معمولاً بعد از گذاشتن پرانتز مربوط ظاهر میشود و در مورد پارامترهای لازم می باشد نیز می توانید از کلیدهای Ctrl+Shift+SpaceBar استفاده کنید.

رفتن از قسمت تعریف توابع و روالها به قسمت کد آنها:

همیشه این نیاز وجود خواهد داشت که شما در هنگامی که دارید به دنبال یک روال در قسمت type میگردید بعد از پیدا کردن نام آن می خواهید که خود آن تابع یا روال را نیز ببنید. برای اینکار خوب حتما نام آن را جستجو میکنید ولی یک راه آسانتر اینست که شما روی نام آن تابع قرار گیرید و کلیدهای Ctrl+Shift+Up/Down را بزنید. در اینحالت اگر روی کد تابع باشید به قسمت تعریف آن خواهید رفت.

نکات ریز و درشت در دلفی

چگونه تمامی رویدادهای یک شیء را در زمان اجرا به Nil تنظیم کنیم؟

در این مقاله روشی را جهت اینکه تمامی رویدادهای یک شیء تعریف شده در دلفی را در زمان اجرا به Nil تنظیم کنید برای شما بازگو می‌کنیم. شما می‌توانید از RTTIها جهت رسیدن به اهداف خود استفاده کنید اما فقط برای زمان طراحی و اجرا و این امکان برای رویدادها وجود ندارد. استفاده از RTTI، تا حدودی پیچیده است بنابراین من رویه‌ای را برای نسبت دادن Nil به یک شیء موجود در زمان اجرای یک برنامه در دلفی آورده‌ام که نحوه انجام این کار را به شما نشان می‌دهد.

```
;unit uNilEvent

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
;Dialogs, StdCtrls

type
(TForm1 = class(TForm
;Button1: TButton
```

```
;(procedure Button1Click(Sender: TObject
private
{Private declarations}
public
{Public declarations}
;end

var
;Form1: TForm1

implementation

{R *.DFM$}

uses
;TypInfo

;(procedure NilEvents(Instance: TObject
var
;TypeInfo: PTypeInfo
;I, Count: Integer
;PropList: PPropList
;PropInfo: PPropInfo
;Method: TMethod
begin
;TypeInfo := Instance.ClassInfo
;Method.Code := nil
;Method.Data := nil
;(Count := GetPropList(TypeInfo, [tkMethod], nil
;((GetMem(PropList, Count * SizeOf(Pointer
try
;(GetPropList(TypeInfo, [tkMethod], PropList
```

```
for I := 0 to Count -1 do
begin
PropInfo := PropList^[i];
SetMethodProp(Instance, PropInfo, Method);
end;
finally
FreeMem(PropList);
end;
end;

procedure TForm1.Button1Click(Sender: TObject);
const
sText = 'The 2nd time you click Button1 the event will not fire';
begin
NilEvents(Button1);
ShowMessage(sText);
end;

end.
```

چگونه رویداد مربوط به تمامی اجزاء درون برنامه خود را تغییر دهیم؟

گاهی لازم می‌شود تا ما فرامینی را برای یکی از رویدادهای برنامه خود به تمامی اشیاء استفاده شده در برنامه نسبت دهیم اینکار می‌تواند با استفاده از RTTI دلفی صورت گیرد شما مثالی را می‌بینید که این کار را شبیه‌سازی کرده است.

```
uses
TypInfo;
```

```
procedure TFrmRTTIOnChange.Button1Click(Sender: TObject);
var
  propInfo: PPropInfo;
  thisEvent: TNotifyEvent;
begin
  propInfo := GetPropInfo(Memo1.ClassInfo, 'OnChange');
  if propInfo <> nil then
  begin
    thisEvent := Memo1AltChange;
    SetOrdProp(Memo1, PropInfo, integer(@thisEvent));
  end;
end;

procedure TFrmRTTIOnChange.Memo1Change(Sender: TObject);
begin
  Caption := 'Normal On Change';
end;

procedure TFrmRTTIOnChange.Memo1AltChange(Sender: TObject);
begin
  Caption := 'Alternate On Change';
end;
```

چگونه با استفاده از شیوه Boyer-Moore جستجو کنیم؟

راه حل اول:

```
unit BMSearch;

interface

type
{IFDEF WINDOWS$}
size_t = Word;
{ELSE$}
size_t = LongInt;
{ENDIF$}

type
TTranslationTable = array[char] of char; { translation table }
TSearchBM = class(TObject)
private
FTranslate: TTranslationTable; { translation table }
FJumpTable: array[char] of Byte; { Jumping table }
FShift_1: integer;
FPattern: pchar;
FPatternLen: size_t;
public
procedure Prepare(Pattern: pchar; PatternLen: size_t; IgnoreCase: Boolean);
procedure PrepareStr(const Pattern: string; IgnoreCase: Boolean);
function Search(Text: pchar; TextLen: size_t): pchar;
function Pos(const S: string): integer;
end;
```

```
implementation

uses
;SysUtils

{Ignore Case Table Translation}

procedure CreateTranslationTable(var T: TTranslationTable;
;(IgnoreCase: Boolean
var
;c: char
begin
for c := #0 to #255 do
;T[c] := c
if not IgnoreCase then
;exit
for c := 'a' to 'z' do
;(T[c] := UpCase(c

{ Mapping all accented characters to their uppercase equivalent }

;'T['Á'] := 'A
;'T['À'] := 'A
;'T['Ä'] := 'A
;'T['Â'] := 'A

;'T['á'] := 'A
;'T['à'] := 'A
;'T['ä'] := 'A
;'T['â'] := 'A

;'T['É'] := 'E
```

```
;'T['È'] := 'E
;'T['Ë'] := 'E
;'T['Ê'] := 'E

;'T['é'] := 'E
;'T['è'] := 'E
;'T['ë'] := 'E
;'T['ê'] := 'E

;'T['Í'] := 'I
;'T['Ì'] := 'I
;'T['Ï'] := 'I
;'T['Î'] := 'I

;'T['í'] := 'I
;'T['ì'] := 'I
;'T['ï'] := 'I
;'T['î'] := 'I

;'T['Ó'] := 'O
;'T['Ò'] := 'O
;'T['Ö'] := 'O
;'T['Ô'] := 'O

;'T['ó'] := 'O
;'T['ò'] := 'O
;'T['ö'] := 'O
;'T['ô'] := 'O

;'T['Ú'] := 'U
;'T['Ù'] := 'U
;'T['Ü'] := 'U
;'T['Û'] := 'U

;'T['ú'] := 'U
;'T['ù'] := 'U
```

```
;'T['ü'] := 'U
;'T['û'] := 'U

;'T['ñ'] := 'Ñ
;end

{Preparation of the jumping table}

procedure TSearchBM.Prepare(Pattern: pchar; PatternLen: size_t;
;(IgnoreCase: Boolean
var
;i: integer
;c, lastc: char
begin
;FPattern := Pattern
;FPatternLen := PatternLen
if FPatternLen < 1 then
;(FPatternLen := strlen(FPattern

{This algorythm is based on a character set of 256}
if FPatternLen > 256 then
;exit
{Preparing translating table .۱}

;(CreateTranslationTable(FTranslate, IgnoreCase

{Preparing jumping table .۲}
for c := #0 to #255 do
;FJumpTable[c] := FPatternLen
for i := FPatternLen - 1 downto 0 do
begin
;[[c := FTranslate[FPattern[i

if FJumpTable[c] >= FPatternLen - 1 then
;FJumpTable[c] := FPatternLen - 1 - i
;end
;FShift_1 := FPatternLen - 1
```

```
;[[lastc := FTranslate[Pattern[FPatternLen - 1
for i := FPatternLen - 2 downto 0 do
if FTranslate[FPattern[i]] = lastc then
begin
;FShift_1 := FPatternLen - 1 - i
;break
;end
if FShift_1 = 0 then
;FShift_1 := 1
;end

procedure TSearchBM.PrepareStr(const Pattern: string; IgnoreCase:
;(Boolean
var
;str: pchar
begin
if Pattern <> '' then
begin
{IFDEF Windows$}
;[str := @Pattern[1
{ELSE$}
;(str := pchar(Pattern
{ENDIF$}
;(Prepare(str, Length(Pattern), IgnoreCase
;end
;end

{Searching Last char & scanning right to left}

;function TSearchBM.Search(Text: pchar; TextLen: size_t): pchar
var
;shift, m1, j: integer
;jumps: size_t
```

```
begin
;result := nil
if FPatternLen > 256 then
;exit
if TextLen < 1 then
;(TextLen := strlen(Text
;m1 := FPatternLen - 1
;shift := 0
;jumps := 0
{Searching the last character}
while jumps <= TextLen do
begin
;(Inc(Text, shift
;[[^shift := FJumpTable[FTranslate[Text
while shift <> 0 do
begin
;(Inc(jumps, shift
if jumps > TextLen then
;exit
;(Inc(Text, shift
;[[^shift := FJumpTable[FTranslate[Text
;end
{ Compare right to left FPatternLen - 1 characters }
if jumps >= m1 then
begin
;j := 0
while FTranslate[FPattern[m1 - j]] = FTranslate[(Text - j)^] do
begin
;(Inc(j
if j = FPatternLen then
begin
;result := Text - m1
;exit
;end
;end
```

```
;end
;shift := FShift_1
;(Inc(jumps, shift
;end
;end

;function TSearchBM.Pos(const S: string): integer
var
;str, p: pchar
begin
;result := 0
if S <> '' then
begin
{IFDEF Windows$}
;[str := @S[1
{ELSE$}
;(str := pchar(S
{ENDIF$}
;((p := Search(str, Length(S
if p <> nil then
;result := 1 + p - str
;end
;end

.end
```

راه حل دوم:

این یک دمو برای الگوریتم جستجوی Boyer-Moore است. ایده بسیار ساده است. در ابتدا یک فهرست از رشته‌های مورد نظر جهت جستجو ساخته و سپس روتین BMsearch را صدا می‌کنیم. فراموش نکنید که در برنامه نهایی خود Range Checking را با سوییچ {$R-} خاموش کنید. در غیر اینصورت جستجو ۲ الی ۳ ساعت بیشتر از زمان عادی طول می‌کشد.

```
.Public-domain demo of Boyer-Moore search algorithm}
{.Guy McLoughlin - May 1, 1993

;program DemoBMSearch

{Boyer-Moore index table data definition}
type
;BMTable = array[0..127] of byte

{.Create a Boyer-Moore index table to search with}

;(procedure Create_BMTable(Pattern: string; var BMT: BMTable
var
;Index: byte
begin
,fillchar(BMT
```

DataSnap, WebSnap

پس از اینکه بورلند تکنولوژی MIDAS را در جهت انجام برنامه‌های چند لایه و تحت Web ارائه کرد در ویرایش جدید دلفی تکنولوژی را مطرح کرده که می‌توان گفت تاثیر زیادی بر دنیای برنامه‌نویسی Web خواهد داشت. تکنولوژی Websnap در واقع پایه ویرایش WebBroker در دلفی ۵ مطرح کرده است که ویژگیهای بسیار زیادی همراه آن می‌باشد:

- Multiple Modules
- New wizards
- Server side scripting
- New component
- Surface design

Multiple Modules :

یک برنامه منفی Websnap می‌تواند از چند مدل وب پشتیبانی کند، در این حالت چندین پیاده‌ساز توانایی کاربردهای یک یا چند پروژه مشابه را دارند بطوری که کمترین تداخل بین آنها صورت گیرد برای این منظور یک Wizard جهت ایجاد Websnap ایجاد شده است که شما براحتی می‌توانید، انواع مدلها از قبیل Webappage، WebAppData، Webpage و WebData را ایجاد کنید. مدل WebData همانند یک ظرف برای اجزاء که شما بین برنامه Web خود به اشتراک می‌گذارید می‌باشد.

مدل Webpage برای شما یک صفحه Web را در برنامه مورد نظر شما ایجاد می‌کند که دارای فرمت و تواناییهای HTML می‌باشد و شما توسط Websnap surface designer توانایی نمایش Scriptهای خاصی خود را دارید.

Server-side scripting :

Websnap دارای اسکریپت نویسی سرویس دهنده می‌باشد یک زبان ساده جهت طراحی و برنامه‌نویسی یک صفحه Web که شامل:

- یک اسکریپت که توانایی دسترسی به Componentهای دلفی را دارد.

- از زبانهای Script همانند VB یا JavaScript حمایت می‌کند.

- ولی می‌تواند ترکیبی در اسکریپت و HTML باشد.

DataSnap :

این تکنولوژی اجازه می‌دهد تا بانکهای اطلاعاتی چندلایه ایجاد شوند که برنامه‌های دلفی توانایی اتصال سرویس گیرنده‌ها را به این لایه‌ها فراهم می‌کنند.بوسیله این تکنولوژی شما توانایی جستجو و کار با بانکهای اطلاعاتی در روی سرورهای مختلف در چند نقطه را خواهید داشت بعنوان مثال می‌توانید از ایران روی سرور در ژاپن Query بزنید و نتیجه آن را در Query دیگری در سرور آلمان شرکت دهید. البته این قابلیت می‌تواند در محیطهای غیر Web نیز استفاده شود

دلفی و اسمبلی

برنامه نویسان دوران DOS با tasm.exe و tlink.exe آشنایی دارند.ایندو از دیرباز رقیبان masm.exe و link.exe بوده و هستند.بله tasm و برادرکوچکش tlink از اولین ابزارهای برنامه نویسان حرفه ای هستند.اما بعد از ظهور Turbo Pascal، Turbo C/C++ و متعاقبا کامپایلر های تحت ویندوز ، کم کم محبوبیت خود را از دست دادند.هم اکنون کمتر صحبتی در مورد این افسانه های قدیمی پردازنده های x86 میشود.

آیا این نشانه مرگ tasm است؟

آیا میدانید هر بار که Delphi یا C++Builder را نصب میکنید ، نمونه ۳۲ بیتی این اسمبلر باوفا نیز میهمان دیسک سخت شما میشود؟

آیا میدانید مهمترین بخشهای VCL توسط اسمبلی و tasm32.exe نوشته شده است؟

آیا می خواهید روش استفاده از tasm در Delphi را یاد بگیرید؟

اگر جواب شما به سوال آخر بله است ، با من همراه شوید تا یک برنامه کاملا ابتدایی و آموزشی با ترکیب قدرت tasm و Delphi بسازیم.اگر جواب شما به سوال آخر خیر است لطفا موس خود را به سمت گوشه بالا و راست این پنجره برده و روی آن دکمه ضربدری شکل کلیک کنید!

چه کسانی میتوانند از این مقاله استفاده کنند؟
قبل از هر چیز خواننده باید با دستورات اسمبلی آشنا باشد.با یک جستجوی ساده در اینترنت میتوانید به مقالات آموزشی زیادی دسترسی پیدا کنید.
آشنایی با نحوه پیاده سازی ویندوز نیز کمک زیادی بشما خواهد کرد اما برای استفاده این مقاله لازم نیست.در این مورد، شاید مقاله دیگر من بنام "اسمبلی تحت ویندوز" بتواند بشما کمک کند.

آستینها رابالا بزنید!!
برنامه ای که خواهیم ساخت کار بسیار ساده ای انجام میدهد.
این برنامه یک Message box را نمایش داده که دارای دو کلید OK و Cancel است.هر کدام از این کلید ها که click شوند توسط یک Message box دیگر به کاربر گزارش داده میشود.

برای شروع یک فایل با پسوند dpr ساخته و کد زیر را در آن کپی کنید:

```
;program DelphiAsm
;uses windows

var
Title :PChar = 'Inline assembly with Reza'#0;//bar hasbeh adat!!
;Mes1 : PChar = 'Please press one of the following buttons!!'#0
;Mes2 : PChar = 'You pressed OK button!!'#0
;Mes3 : PChar = 'You pressed Cancel button!!'#0

label
;PRINTPROC

begin
asm
push 1
push Title
push Mes1
push 0
call MessageBoxA
cmp eax , IDOK
je PRINTPROC
push Mes3
pop Mes2

PRINTPROC:
push 0
push Title
push Mes2
push 0
call MessageBoxA
```

```
mov eax,0
call ExitProcess
;end
.end
```

نکته:در دلفی برای نوشتن دستورات اسمبلی از بلوک asm ... end; استفاده میشود.

قبل از هر چیز به نحوه تعریف یک لیبل توسط کلمه کلیدیLabel توجه کنید.چه در tasm و چه در masm احتیاجی به اعلان کردن لیبل ها ندارید ولی از آنجایی که Delphi یک کامپایلر است و قوانین خواص خود را دارد ، قبل از استفاده از هر لیبل باید آنرا اعلان کنید.

نکته دوم مسئله calling convention است.باید بدانید که Delphi از روش Fastcall برای فراخوانی توابع خود استفاده میکند ولی windows از روش stdcall بهره میبرد.

Stdcall یعنی آرگومانها را از چپ به راست در پشته قرار می گیرند و callee (فراخوانده) مسئول pop کردن آرگومانها است.(برای اطلاعات بیشتر در مورد انواع calling convention ها به MSDN یا مقاله دیگر من بنام "اسمبلی تحت ویندوز" مراجعه کنید.)

برای یافتن Prototype اولین تابع مورد استفاده نگاهی به MSDN بیندازید و تابع MessageBox (و نه MessageBoxA یا MessageBoxW) را search کنید :

```
int MessageBo
HWND hWnd, // handle to owner window
LPCTSTR lpText, // text in message box
LPCTSTR lpCaption, // message box title
UINT uType // message box style
);
```

اکنون میدانید که

۱.Prototype تابع MessageBox چیست.

۲.calling convention استاندارد یا stdcall چگونه عمل میکند.

پس دست بکار شده و آرگومانها را از چپ به راست در stack بچپانید!! سپس تابع MessageBox را فراخوانی کنید.

همانطور که در Prototype تابع MessageBox میبینید مقدار بازگشتی آن یک Integer است که نشان دهده دکمه کلیک شده در Message box است.

نکته مهم:

توابع stdcall مقدار بازگشتی خود را در eax قرار میدهند.

آقا یادم رفت بگم ، اسامی چون IDOK, IDCANCEL, MessageBox و... در یونیت windows اعلان شده اند.

پس تا اینجا موفق شدیم یک MessageBox با متن
Please press one of the following buttons!! نمایش دهیم .بر اساس مقدار بازگشتی این تابع تشخیص میدهیم که کدام دکمه(button) کلیک شده است و با یک جابجایی ساده و بدون تولید کد اضافی پیام مناسب را توسط یک Message box دیگر به کاربر نمایش میدهیم.حتما میدانید که متغییری از نوع رشته در ویندوز ، یک اشاره گر به اولین کاراکتر آن رشته است.(علت این است که windows بوسیله زبان C پیاده سازی شده است.)

در انتها نیز با فراخوانی ExitProcess تمامی thread های فعال (که در این برنامه یکی بیشتر نیست!!) بسته میشوند ، هر چند که انجام این کار ضروری نیست، دلفی کد لازم را تولید خواهد کرد.می توانید دو خط آخر را حذف کنید و نتیجه کار را ببینید.

در این برنامه برای ساده کردن کار از VCL که نقطه عطف Delphi است استفاده نکردیم اما شما میتوانید در برنامه های عادی خود VCL و اسمبلی را باهم بکار گرفته و برنامه هایی با بازدهی بسیار بالا بسازید.از حالا به بعد اگر شنیدید که :"آره بابا با دلفی نمیشه فلان کارو انجام داد ، باید با اسمبلی کد بنویسی" میدانید چه جوابی بدهید.

زبان برنامه‌نویسی دلفی

زبان برنامه‌نویسی دلفی

دِلفی (Delphi) یا به تعبیری ویژوآل پاسکال – یک زبان برنامه‌نویسی است و بستری برای توسعهٔ نرم‌افزار که شرکت بورلند آن را تولید کرده است. این زبان، در بدو انتشار خود در سال ۱۹۹۵، به عنوان یکی از نخستین ابزارهایی مطرح شد که از توسعهٔ نرم‌افزار بر مبنای متدولوژی RAD پشتیبانی می‌کردند؛ یعنی تولید و توسعهٔ سریع برنامه‌های کاربردی.

این نرم افزار بر مبنای پاسکالشی‌گرا بوده و از این زبان مشتق شده است. البته بورلند نسخه‌ای از دلفی و سی‌پلاس‌پلاس‌بیلدر را برای لینوکس به نام کایلیکس (Kylix) ارائه کرد که مورد استقبال توسعه دهندگان نرم‌افزارهای لینوکس قرار نگرفت. نرم‌افزارهای دلفی در ابتدا به صورت مستقیم از کتابخانه‌های ویندوز و کتابخانهٔ مخصوص خود به نام VCL استفاده می‌کرد، اما پس از نسخه ۶ دلفی، امکانات استفاده از دات‌نت هم به آن اضافه شد. در حال حاضر می‌توان دلفی را یکی از رایج‌ترین زبان‌های ممکن در ایران دانست.

زبانِ دلفی که پیشتر بنام Object-Pascal یا پاسکال شیءگرا خوانده می‌شد و برای طراحی نرم‌افزاهای تحت ویندوز به کار می‌رفت، امروزه چنان توسعه یافته است که برای تولید نرم‌افزارهای تحت سیستم‌عاملِ لینوکس و دات‌نت نیز به کار می‌آید. بیشترین کاربرد دلفی

در طراحی برنامه‌های رومیزی و پایگاه داده‌ها است، اما به عنوان یک ابزارِ «چند- منظوره»، برای طراحی انواع گوناگونی از پروژه‌های نرم‌افزاری نیز مورد استفاده قرار می‌گیرد.
RAD=Rapid Application Development

دلفی ۲۰۰۶

شرکت بورلند در سال ۲۰۰۶ نرم‌افزار جدید خود را با ویژگیهای جدید به بازار ارائه کرد. این برنامه جدید امکان برنامه نویسی با دلفی و سی پلاس پلاس و همچنین سی‌شارپ را بطور هم‌زمان ارائه می‌دهد. بدین ترتیب برنامه نویسانی که با ابزارهای مختلفی کار می‌کنند براحتی می‌توانند در این محیط جدید برنامه نویسی کنند. ویژگی مهم این نگارش نسبت به نگارش ۲۰۰۵ بحث مدیریت حافظه است. در نگارش ۲۰۰۵ ضعفهائی در این زمینه وجود داشت که در این نسخه حل شده است.

آرایه‌ها در دلفی

دلفی به ما امکان می دهد آرایه‌هایی از هر نوع متغییری را ایجاد کنیم. برای تعریف آرایه به صورت زیر عمل میکنیم:

کد:

```
[align=left]var
[array[indexType1, ..., indexTypen] of baseType;[/align
```

در این تعریف برای نامگذاری آرایه، از قانون نامگذاری متغییرها استفاده میکنیم و مقدار اولیه را نیز درون یک جفت کروشه قرار می دهیم.

نکته: شما می توانید به جای استفاده از کروشه [] از ترکیب پرانتز نقطه استفاده کنید:

کد:

```
[align=left]d(.i.):= 3 + i; // Equivalent d[i]:= 3 + i; [/align]
```

نکته: وقتی که شما یک آرایه را تعریف می کنید احتیاجی ندارید که به آن مقدار کمترین یا بیشترین بدهید:

کد:

```
[align=left]var
```

```
;A : array [Boolean] of integer
begin
;A[True] := 50
;A[False] := 100
;end
```

نکته: توابع Low و High کران‌های پایین وبالای یک متغییر آرایه‌ای یا نوعی یا ترتیبی را بر میگردانند:

[for I := 0 to High(X) do S := S + X[i]; [/align

آرایه‌های ثابت:

آرایه های ثابت می توانند توسط ساختار ثابت نوع دلفی تعریف شود. نوع ثابت که همیشه با عبارت Const تعریف می شود، نه تنها مانع تغییر مقدار پارامتر می شود، بلکه کدهای بهینه بیشتری برای رشته‌ها و رکوردهای رد شده به توابع تولید می کند. ما هنگامی از این نوع استفاده می کنیم که نخواهیم مقدار رد شده به یک تابع تغییر کند.

کد:

align=left]type[

```
TDay = (Sunday, Monday, Tuesday, Wednesday, Thursday, Friday,
;(Saturday

const
DayNames : array [TDay] of String[9] = ('Sunday', 'Monday',
,"Tuesday

,'Wednesday', 'Thursday'
```

```
;('Friday', 'Saturday'
var
;Today : TDay

begin
;(Today := TDay(DayOfWeek(Date) - 1
;('!' + [ShowMessage('Today is ' + DayNames[Today
[end;[/align
```

آرایه‌های دینامیکی:

آرایه‌های دینامیکی، آرایه‌های تحلیلی پویایی هستند که ابعاد آنها موقع کامپایل شدن شناخته شده نیست. برای اعلان آنها کافی است یک آرایه بدون بعد تعریف کنید:

کد:

```
[align=left]var MyFlexibleArray: array of Real; [/align]
```

قبل از به کار گیری آرایه‌های دینامیکی، ابتدا باید از رویه SetLength برای تخصیص حافظه آرایه استفاده کرد:

کد:

```
[align=left]SetLength (MyFlexibleArray, 2; ([/align]
```

نکته: آرایه‌های دینامیکی همیشه مبتنی بر صفر می باشند.

نکته: شما می‌توانید آرایه‌های دینامیکی را قبل از رسیدن به ترک قلمرو از حافظه خارج کنید:

کد:

[align=left]MyFlexibleArray := nil; [/align]

نکته: مقدار حافظه‌ای که در اختیار آرایه قرار میگیرد، به طول آرایه ونوع عناصر آن بستگی دارد. به عنوان مثال اگر آرایه‌ای از نوع صحیح به طول ۱۰ داشته باشیم ۴*۱۰ بایت حافظه به آن اختصاص می‌یابد.

فشرده‌سازی آرایه‌ها:

در دلفی شما هنگامی که ساختار خود را تعیین کردید می‌توانید با استفاده از کلمه کلیدی packed اطلاعات ذخیره شده خود را متراکم کنید:

کد:

[align=left]type TNumbers = packed array[1..100] of Real; [/align]

نکته: استفاده از packed سرعت دسترسی به اطلاعات را کند میکند. در مورد آرایه‌ای از کاراکترها این مورد سازگارتر می‌باشد.

آرایه‌های چند بعدی دینامیکی:

برای تعریف آرایه‌های چند بعدی دینامیکی، تنها کافی است array of... را در ساختار خود تکرار کنید. به طور مثال:

کد:

align=left]type]
;TMessageGrid = array of array of string

var
[Msgs: TMessageGrid;[/align

این تعریف یک آرایه دو بعدی از رشته‌ها می باشد. سپس باید به آرایه خود فضا نسبت داد:

کد:

[align=left]SetLength(Msgs, I, J);[/align]

شما می‌توانید آرایه‌های چند بعدی دینامیکی خود را به صورت غیر مستطیلی (Not Rectangular) ایجاد کنید. ابتدا رویه SetLength را صدا زده و پارامتر بعد اول را بدهید:

کد:

align=left]var]
;Ints: array of array of Integer

[SetLength(Ints, 10);[/align

ما ۱۰ سطر به آرایه خود اختصاص دادیم. از این پس، شما می توانید ستونهای خود را در هر زمان (با اندازه‌های مختلف) تخصیص دهید:

کد:

[align=left]SetLength(Ints[2], 5);[/align]

بررسی فشردن کلید توسط کاربر در هنگام اجرای یک حلقه

زمانی که شما از حلقه ها یا همان Loop در برنامه خود استفاده می کنید در برخی مواقع شاید لازم باشد که کاربر بتواند این حلقه را در هر لحظه ای متوقف سازد. در اکثر برنامه های دیده شده این عمل انجام نمی شود و کاربر باید زمان طولانی برای رسیدن به آخر حلقه صبر کند.

به طور مثال فرض کنید برنامه شما می خواهد یک سری اعداد اتفاقی تولید کند و به کاربر نمایش دهد. تعداد این اعداد شاید به صدهزار عدد برسد. حال فرض کنید حلقه ای بسازید که برای کاربر ، این اعداد را یکی یکی نشان دهد.

شاید کاربر شما در حین کار خسته شود و بخواهد این عمل را لغو کند.

اما اگر به صورت معمول این حلقه را اجرا کنید ، وی مجبور است تا انتهای حلقه

صبر کند و این کار بسیار خسته کننده ای می باشد.

این بدان خاطر است که دلفی ، دستورات را یکی پس از دیگری اجرا می کند.
یعنی تا زمانی که حلقه مذکور تمام نشود ، دلفی دستور بعدی را اجرا نمی کند.

اما راه حل چیست ؟ آیا راهی وجود دارد که به کاربر اجازه دهد تا عملیات را در هر زمانی لغو کند ؟

جواب این سوال در استفاده از دستور ProcessMessages خلاصه می شود.
به کار بردن این دستور باعث می شود تا دلفی متوجه باشد که دیگر دستورات برنامه را حین عملیات زیر نظر بگیرد و آنها را اجرا کند.

به طور مثال با به کار بردن این دستور ، دلفی می تواند فشار دادن یک کلید توسط کاربر را هنگام اجرای حلقه متوجه شود و آن را اجرا کند.

اما در صورتی که از این دستور استفاده نکنیم ، دلفی فشاردادن کلید را متوجه می شود ، اما صبر می کند تا حلقه تمام شود و سپس دستورات بعدی را اجرا می کند.

این دستور ، به صورت Application.ProcessMessages به کار می رود.

حال به کد زیر دقت فرمایید :

```
procedure TForm1.Button1Click(Sender: TObject);
var
LoopAborted: Boolean;
```

```
;i: Integer
begin
;LoopAborted := False
;i := 0
repeat

;(Label1.Caption := IntToStr(i
;Application.ProcessMessages

if GetKeyState(VK_Escape) and 128 = 128 then
begin
;LoopAborted := True
;Break
;end

;(Inc(i
;until i = 100000
if LoopAborted then
;('ShowMessage('User has aborted the loop
;end
```

همان طور که ملاحظه می کنید دستور ذکر شده در داخل حلقه گنجانده شده تا فشار دادن کلید ESC توسط کاربر در زمان اجرای حلقه شناخته شود.

این دستور به کاربر این امکان را می دهد تا هر لحظه ای که تمایل داشت اجرای حلقه را لغو کند.

توجه داشته باشید که به کار بردن دستور ProcessMessages تنها محدود به

حلقه ها نمی باشد و در کل هر جایی که لازم باشد تا چندین دستور به همراه هم اجرا شوند ، این دستور نقش اصلی را ایفا می کند.

موفق باشید