رجیستری

رجیستر رو در زبان عامیانه قلب ویندوز ان تی می دانند . رجیستری یک مر کز بسیار مهم در ویندوز می باشد که در انجا تمامی برنامه ها و مقدار ها و فرمت ها و تنظیمات ثبت میشوند در رجیستری می توان به تنظیمات ویندوز دست پیدا کرد و با استفاده از آن بدون استفاده از کنترل پنل تنظیمات ویندوز رو عوض کرد

شما برای کار با رجیستری در دلفی نیاز به تعریف و نوشتن هیچ کلاس و متد و آبجکت ندارید شرکت بورلند برای کار با رجیستر یه یونیت با همین نام Registry ارائه داده که تمام نیاز های ملت رو برای کار با رجیستر تامین می کنه که شامل کلاس TRegistry می باشد که دارای روال ها و تابع های خاصی مثلCreateKey یا WriteString یا DeleteKey یا MoveKey یا ReadFloat و....میباشد.
نحو اصلی کار با رجیستری و ثبت کردن مقدار ایجاد یک کلید و.... به این صورت می باشد ابتدا یونیت رجیستری رو به لیست Uses یونیتمون اضافه می کنیم به شکل زیر :

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,Registry;
```

خب حالا یه متغیر باید از نوع کلاس TRegistry تعریف کنیم تا بتوانیم از توابع آن ها استفاده کنیم. به شکل زیر

```
Var
B:TRegistry;
```

سپس باید یک نوع از کلاس را بوجود بیاوریم که طبق معمول باید از constructor کریییت (Create) مر بوط به اکثر کلاس ها استفاده کنیم به شکل زیر

```
b:=TRegistry.Create;
```

برای کار با رجیستری باید به یک لانه مربوط در آن اشاره کنیم که عبارتند از :

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

HKEY_PERFORMANCE_DATA

HKEY_CURRENT_CONFIG

HKEY_DYN_DATA

خواص و رویدادهای شیئ رجیستری

Rootkey

برای این که یکی از این مقدار ها را انتخاب کنیم باید از پراپرتی RootKey به شکل زیر استفاده کنید :

```
B.RootKey:=HKEY_LOCAL_MACHINE;
```

OpenKey

سپس باید آدرس مربوط به کلیدی که می خواهیم در آن اعمال انجام دهیم رو با استفاده از تابع OpenKey مشخص کنیم به شکل زیر :

```
B.OpenKey( آدرس کلید مربوطه , اجازه بوجود آوردن کلید)
```

در مورد آدرس شما باید آدرس کلید مربوطه رو بدون کی روت آن تایپ کنید و به صورت یک استرینگ در مورد اجازه بوجود آوردن که با یک مقدارFalse یا True ست میشود این صورت است که شما می توانید با ست کردن آن به صورت True به آن اجازه می دهید تا در صورت وجود نداشتن ان ادرس یک کلید با آدرس که شما تایپ کردید بوجود بیاورد البته ما به شما پشنهاد می کنیم که مقدار مربوطه را همیشه False کنید چون شما می توانید با تابع CreateKey یک کلید بسازید پس بهتره که شما از تابع اپن کیOpenKey آدرس کلید مربوطه تان را لود کنید .

حذف یک کلید

با استفاده از متد DeleteKey میتوان مسیر مورد نظر را حذف نمود

```
b.deletekey(ادرس کلید مربوطه);
```

ایجاد یک کلید

با استفاده از متد CreateKey میتوان مسیر مورد نظر را ایجاد کرد

```
b.createkey(ادرس کلید مربوطه);
```

openKeyreadOnly

باز کردن یک مسیر برای خواندن

KeyExisits

چک کردن وجود یک مسیر که یک مقدار از نوع string میگیرد که همان ادرس مورد نظر ماست

GetKeyNames

لیست کردن زیر کلید های یک مسیر یک مقدار از نوع Strings میگیرد و در ان لیست زیر کلیدها را قرار میدهد

GetKeyValues

لیست کردن مقادیر یک کلید یک مقدار از نوع Strings میگیرد و در ان لیست مقادیر را قرار میدهد

HasSubKey

چک کردن وجود زیر کلید مقدار خروجی ان در صورت وجود True و در غیر این صورت False خواهد بود

CurrentKey

خروجی این متد کلید جاری که باز است می باشد

CurrentPath

خروجی این متد مسیر کلیدی که باز است میباشد

توابع خواندن مقادیر از رجیستری

نام تابع (نام عبارتی که باید خوانده شود)

ReadString

خواندن یک مقدار از نوعstring

ReadInteger

خواندن یک مقدار از نوع integer

ReadBool

خواندن یک مقدار از نوع boolean

ReadDateTime

خواندن یک مقدار از نوع DateTime

ReadDate

خواندن یک مقدار از نوع Date

ReadCurrency

خواندن یک مقدار از نوع Currency

ReadBinaryData

خواندن یک مقدار از نوع باینری

ReadFloat

خواندن یک مقدار از نوع اعشاری

ReadTime

خواندن یک مقدار از نوع زمان

توابع نوشتن در رجیستری

نام تابع (نام مقداری که باید از نو ثبت شود و یا با نویسی شود , مقدار)

WriteString

نوشتن یک مقدار از نوع String

WriteExtendedString

نوشتن یک مقدار از نوع Extended string

WriteInteger

نوشتن یک مقدار از نوع integer

WriteBool

نوشتن یک مقدار از نوع boolean

WriteDateTime

نوشتن یک مقدار از نوع DateTime

WriteDate

نوشتن یک مقدار از نوع Date

WriteTime

نوشتن یک مقدار از نوع زمان

WriteFloat

نوشتن یک مقدار از نوع اعشاری

WriteCurrency

نوشتن یک مقدار از نوع currency

WriteBinaryData

نوشتن یک مقدار از نوع باینری

DeleteValue

حذف یک مقدار این متد یک مقدار از نوع Stringمیگیرد که نام مقدار می باشد

GetDataInfo

گرفتن اطلاعات یک مقدار

GetDataSize

گرفتن طول یک مقدار

GetDataName

گرفتن نام یک مقدار

GetKeyInfo

گرفتن اطلاعات یک کلید

LoadKey

بارگزاری یک کلید این متد ۲ پارامتر دارد اولی نام کلید و دومی ادرس فایلی که کلید باید از ان لود شود

MoveKey

انتقال یک کلید به مکان دیگر ۲ پارامتر از نوع Stringمیگیرد اولی ادرس قدیم و دومی ادرس جدید

UnloadKey

خالی کردن یک کلید یک پارامتر از نوع String که کلید مورد نظر در ان قرار دارد

RenameValue

تغییر یک مقدار به مقدار جدید این تابع ۲ پارامتر دارد اولی مقدار قدیم و دومی مقدار جدید

ReplaceKay

تغییر مکان ۲ کلید با هم

RestoreKey

باز یابی یک کلید این متد ۲ پارامتر دارد اولی نام کلید دومی نام فایلی که کلید باید از ان خوانده و بازیابی شود

GetDataAsString

گرفتن اطلاعات به صورت رشته

SaveKey

ذخیره یک کلید این متد ۲ پارامتر دارد اولی نام کلید دومی نام فایلی که کلید باید در ان ذخیره شود

CloseKey

بستن یک کلید

برای مثال به کد زیر یک نگاه بیندازید این کد تسک منیجر (ctrl+Alt+Delete) را غیر فعال می کند:

```
var
dbm: Tregistry;
begin
DbM:= tregistry.Create;
```

ایجاد یک کلاس از نوع registry

```
DbM.RootKey:= HKEY_CURRENT_USER;
```

باز کردن شاخه روت

```
DbM.OpenKey('Software\MicrosoFt\Windows\CurrentVersion\Policies\system\',false);
```

رفتن به کلید مورد نظر

```
DbM.WriteInteger('DisableTaskMgr',0);
```

نوشتن در کلید مورد نظر

```
DbM.Free;
```

ازاد سازی فضای اختصاص داده شده به کلاس Registry بعد از اتمام کار

```
end;
```

در اینجا چند مثال دیگر برای کار با رجیستری می اوریم

بدست اوردن برنامه های نصب شده روی سیستم

```
procedure TForm2.FormCreate(Sender: TObject);
const
  UNINST_PATH = 'SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall';
var
  Reg: TRegistry;
  SubKeys: TStringList;
  i: integer;
  sDisplayName, sUninstallString: string;
begin
  Reg := TRegistry.Create;
        Reg.RootKey := HKEY_LOCAL_MACHINE;
```

باز کردن مسیر مورد نظر رای خواندن

```
        if Reg.OpenKeyReadOnly(UNINST_PATH) then
        begin
          SubKeys := TStringList.Create;
            Reg.GetKeyNames(SubKeys);
```

لیست کردن زیر کلید ها در یک Tstringlist

```
            Reg.CloseKey;
```

```
        for i := 0 to subKeys.Count - 1 do
        begin
          if (Reg.OpenKeyReadOnly(UNINST_PATH+'\'+SubKeys[i])) and
(Reg.ReadString('DisplayName')<>'') then
                                    چک کردن وجود مسیر و داشتن مقدار DisplayName
            listbox1.Items.Add(Reg.ReadString('DisplayName'));
                                    در اینجا مقدار displayName در Listboxقرار میکیرد
              reg.CloseKey;
            end;
          end;
        SubKeys.Free;
    reg.Free;
                پس از پایان کار فضای اختصاص داده شده به Tstringlist و Tregistry را ازد میکینم
  end;
```

بدست اوردن CpuUsage یک Label و یک timer روی قورم قرار دهید

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
  CPUU : integer ;
begin
  if started then
  begin
    Reg.OpenKey('PerfStats\StatData',false);
    Reg.ReadBinaryData('KERNEL\CPUUsage',CPUU,SizeOf(Integer));
```

در اینجا از ادرس 'KERNEL\CPUUsage' درون متغییر CPUU به سایز (SizeOf(Integer ریخته می شود البته میتوان بجای CPUU از یک ارایه دینامیک استفاده کرد و طول مقدار باینری را با متد GetValueSize بدست اورد

```
    Reg.CloseKey ;
    Label1.Caption:=IntToStr(CPUU)+ '%' ;
  end ;
```

```
end;
```

فصل ۱۷

TStrings چیست ؟

TStrings کلاسی است برای نگهداری مجموعه ای از رشته ها همراه با Index و شماره مخصوص هر رشته که امکان ویرایش و دسترسی آسان آنها را فراهم میکند ، علاوه بر این میتوان در یک TStrings ، یک Object وابسته به یک رشته را نیز ثبت کرد و از آن استفاده نمود . البته به علت زیادی متدها و خصوصیات ، به صورت موردی و کاربردی آنها را بررسی میکنیم ...

ثبت اطلاعات در یک TStrings :

برای ثبت اطلاعات در یک کلاس TStrings ابتدا باید آن را Create کرده و سپس از متدهایی مثل Add، AddObject و ... استفاده نماییم ...

Create کردن یک TStrings :

با استفاده از متد Create میتوانیم یک TStrings ایجاد نماییم :

```
var
 Strings : TStrings;
begin
 Strings := TStrings.Create;
end;
```

اگر نیاز داریم که تنها یک رشته را در یک TStrings ثبت نماییم ، میتوانیم از دستور Add استفاده نماییم :

```
var
 Strings : TStrings;
 S : String;
begin
```

```
 S := 'String';
 Strings := TStrings.Create;
 Strings.Add(S);
end;
```

دستور Add در کد بالا مقدار متغیر S را در Strings که یک مقدار TStrings است ثبت می نماید ...

گاهی اوقات نیاز داریم که تعدادی Object (مثل TabSheet های یک PageControl) را در یک TStrings ثبت نماییم ، بدین منظور باید از دستور AddObject استفاده نماییم

البته توجه کنید که یک متغیر از نوع TStrings نمی تواند یک Object را در خود نگاه دارد (در زمان اجرا خطای AbStract نمایش داده خواهد شد)

باید Object در آیتمهای یک شی دیگر که از نوع TStrings است قرار گیرد ، برای مثال می توانید در آیتم های یک ListBox و یا یک ComboBox آبجکت مورد نظر را Add کنید

برای مثال :

```
ListBox1.Items.AddObject('MyTabSheet', PageControl1.Pages[0]);
```

پس از ثبت این Object میتوانید برای مثال به صورت زیر از آن استفاده نمایید :

```
PageControl1.ActivePage := TTabSheet(ListBox1.Items.Objects[1]);
```

کد بالا صفحه فعال PageControl را برابر با TabSheet موجود در لیست (که در کد مشخص شده) قرار می دهد ...

دستور دیگری که برای ثبت اطلاعات در TStrings استفاده می شود ، دستور AddStrings است که یک مقدار TStrings را به لیست رشته ها اضافه خواهد کرد ...

مثال :

```
var
 Strings : TStringList;
 S : String;
begin
 S := 'String;'
 Strings := TStringList.Create;
```

```
 Strings.Add(S);
 ListBox1.Items.AddStrings(Strings);
end;
```

کد بالا ، رشته های موجود در Strings (که همان S است) را در ListBox1 قرار خواهد داد ...

دستور دیگری که برای اضافه کردن اطلاعات به یک TStrings به کار میرود ، دستور Append است ، این دستور یک مقدار String را به آخر لیست رشته ها در TStrings قرار می دهد ...

مثال :

```
var
 S : TStrings;
begin
 S := TStrings.Create;
 S.Append('Hello');
end;
```

راه دیگر برای اضافه کردن یک رشته به TStrings استفاده از متد Insert است ...

این متد ۲ پارامتر دارد ، ۱. Index که شماره Index رشته ای که اضافه می شود را تعیین میکند و ۲. S که مقدار رشته ای که باید اضافه شود را دربر می گیرد ..

مثال :

```
var
 S : TStrings;
begin
 S := TStrings.Create;
 S.Insert(0, 'Hello');
end;
```

کد بالا رشته Hello را به عنوان اولین مقدار در لیست قرار می دهد (به عنوان آیتمی که Index صفر دارد)

کلاس TStrings همچنین توابع دیگری به نام های LoadFromFile و LoadFromStream دارد که امکان وارد کردن رشته ها از یک فایل متنی و از Stream را میدهد ...

این دستورات یک پارامتر دارند که نام فایل و نام Stream را دربر می گیرد ...

مثال:

```
var
 S : TStrings;
begin
 S := TStrings.Create;
 S.LoadFromFile(FileName);
 S.LoadFromStream(Stream);
end;
```

حذف اطلاعات :

برای حذف اطلاعات از یک TStrings چندین راه وجود دارد که بررسی میکنیم

استفاده از متد Delete :

این متد یک رشته (آیتم) را از لیست حذف می نماید ، این متد یک پامتر دارد که باید Index مربوط به آیتمی که قصد حذف کردن آن را دارید را در آن وارد نمایید :

```
var
;S : TStrings
begin
;S := TStrings.Create
;(S.Delete(0
;end
```

استفاده از متد Clear :

این متد کل رشته ها (آیتمهای) موجود در TStrings را حذف خواهد کرد ...

```
var
;S : TStrings
```

```
begin
;S := TStrings.Create
;S.Clear
;end
```

سایر متدها و توابع پرکاربرد :

متد Exchange :

این متد مقدار (یا جایگاه) دو آیتم را با هم عوض میکند ، این متد ۲ پارامتر دارد با نامهای Index1 و Index2 ، بعد از اجرای این متد جای Index1 با Index2 عوض خواهد شد ...

برای مثال فرض کنیم در یک TStrings، یک آیتم با Index صفر داریم و یک آیتم دیگر با Index یک که مقدار رشته اولی برای با " Hello " و دومی برابر با " Bye " است ، پس اجرای این متد با Index های صفر و یک ، مقدار Index برای رشته " Hello " برابر با ۱ و مقدار Index برای رشته " Bye " صفر خواهد بود :

```
S.Exchange(0, 1);
```

تابع Equals :

این تابع مقادیر TStrings فعلی را با مقادیر یک TStrings دیگر که به عنوان پارامتر دریافت می کند مقایسه میکند و در صورت همسان بودن ، مقدار True و در غیر این صورت مقدار False را برمی گرداند :

```
S.Equals(Strings : TStrings);
```

متد Move :

این متد یک رشته را از یک Index به یک Index دیگر منتقل میکند ، به عنوان مثال فرض کنیم که یک TStrings با ۴ رشته (آیتم) داریم ، اگر بخواهیم مثلا آیتم سوم را از جای خود به آیتم اول ببریم باید به صورت زیر از متد Move استفاده نماییم :

```
var
;S : TStrings
begin
;S := TStrings.Create
```

```
;(S.Move(2, 0
;end
```

توابع IndexOf و IndexOfObject :

به وسیله این توابع می توانید Index یک رشته و یا یک Object را بدست آورید ، این توابع هر کدام یک پارامتر دارند ، تابع IndexOf پارامتری از نوع String دارد که رشته را گرفته و Index آن را به عنوان خروجی برگشت میدهد و تابع IndexOfObject هم پارامتری از نوع TObject دارد که Object مورد نظر را به عنوان ورودی دریافت کرده و شماره Index آن را به عنوان خروجی برگشت میدهد ...

```
S.IndexOf('Hello');
S.IndexOfObject(PageControl1.Pages[0]);
```

تابع Strings :

با استفاده از این تابع می توانید یک رشته از TStrings را با دادن Index آن استخراج کنید :

```
;[MyStr := S.Strings[0
```

خاصیت Capacity :

این خاصیت از نوع Integer بوده و به وسیله آن میتوانید محدودیت تعداد رشته (آیتم) هایی که می توان در TStrings مورد نظر ثبت کرد را مشخص نمایید ...

مثلا اگر این خصوصیت بر روی ۶ تنظیم شده باشد ، TStrings مورد نظر تنها ۶ آیتم دریافت خواهد کرد ...

خاصیت Count :

این خاصیت از نوع Integer بوده و تعداد آیتمهای موجود در TStrings را به ما می دهد

این نکته رو هم ذکر کنم که این مقاله کامل کامل نبود یعنی در این مقاله کل توابع و متد ها و خصوصیتهای کلاس TStrings بررسی نشد ، کلاس TStrings چندین خاصیت و تابع داره که کاربردهای خاص دارند و به صورت معمول استفاده نمی شن ، از این جمله میشه به خصوصیت CommaText اشاره کرد ، با استفاده از این Property می توان رشته های درون TStrings را به صورت (SDF) یا System Data Format داشت

همچنین در این مبحث از توضیح برخی توابع مانند InsertObject که عملکردی شبیه تابع Insert دارند اجتناب شد ...

Tstringlist چیست

TStringlist نیز همانند Tstringsکلاسی است برای نگهداری مجموعه ای از رشته ها همراه با Index و شماره مخصوص هر رشته که امکان ویرایش و دسترسی آسان آنها می باشد ، علاوه بر این میتوان در یک TStrings ، یک Object وابسته به یک رشته را نیز ثبت کرد و از آن استفاده نمود . البته به علت زیادی متدها و خصوصیات ، به صورت موردی و کاربردی آنها را بررسی میکنیم ...

Create

ایجاد یک کلاس از نوع TstringList

Add

اضافه کردن یک ایتم به لیست یک پارامتر از نوع String که نشان دهنده ایتم مورد نظر ما می باشد

Clear

خالی کردن کلاس از ایتمها

Delete

حذف یک ایتم که یک پارامتر از نوع Integer دارد که نشان دهنده index ایتم مورد نظر برای حذف می باشد

Exchange

جابجایی ۲ ایتم باهم این متد دارای ۲ پارامتر از نوع Integer میباشد که نشان دهنده index ایتمهای مورد نظر برای جابجایی می باشد

Find

جستجو بین ایتمها این متد دارای ۲ پارامتر میباشد اولی از نوع string و نشان دهنده ایتم مورد نظر و دومی از نوع Integer که index ایتم در صورت یافت شدن در ان قرار میگیرد و خروجی تابع مقدار true میگیرد این تابع حالت پیش رونده رو به جلو دارد

IndexOf

این تابع index ایتم مورد نظر را به ما میدهد و یک پارامتر از نوع String دارد که بیان کننده ایتم مورد نظر می باشد

Insert

این هم همانند متد Add برای اضافه کردن یک ایتم به لیست در مکان دلخواه میباشد و دارای ۲ پارامتر میباشد اولی نام ایتم و دومی index مکانی که میخواهیم ایتم در انجاقرار گیرد

Sort

ردیف کردن ایتمها بر اساس حرف الفبا

CustomSort

ردیف کردن ایتمها به شیوه ای که مد نظر برنامه نویس میباشد

Append

این تابع نیز برای اضافه کردن یک ایتم به انتهای لیست می باشد

Addstring

اضافه کردن یک String به لیست

Onchange

این رویداد بعد از انجام تغییر در لیست فراخوانی می شود

OnChanging

این رویداد در هنگام تغییر در لیست فراخوانی می شود

LoadFromfile

لیست را از یک فایل بارگزاری میکند یک پارامتر از نوع String دارد که بیانگر ادرس فایل مورد نظر برای بارگزاری می باشد

SavetoFile

لیست را در یک فایل ذخیره می کند یک پارامتر از نوع String دارد که بیانگر ادرس فایل مورد نظر برای ذخیره می باشد

Count

ایتمهای داخل لیست را شمارش میکند

LoadFromStream

لیست را از یک Stream بارگزاری میکند

SavetoStream

لیست را در یک stream قرار میدهد

مثال

```
var
  Form2: TForm2;
list:tstringlist;
```

تعریف یک Tstringlist به صورت عمومی تا در سراسر یونیت قابل استفاده باشد

```
implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
```

```
begin
list.Add(edit1.Text);
listbox1.Items.Add(list.Strings[list.Count-1])
```

اضافه کردن یک ایتم به انتهای لیست و اضافه کردن اخرین ایتم لیست به Listbox

```
end;

procedure TForm2.Create(Sender: TObject);
begin
list:=TStringList.Create;
```

ایجاد یک کلاس TstringList

```
list.LoadFromFile('c:\file.txt');
```

بارگزاری کلاس از فایل

```
List.sort;
```

ردیف کردن لیست بر اساس حروف الفبا

```
listbox1.Items:=list;
```

پر کردن Listbox از لیست

```
end;

procedure TForm2.Button3Click(Sender: TObject);
begin
ListBox1.ItemIndex:=list.IndexOf(Edit2.Text);
end;
```

جستجو در لیست و نشان دادن ان در Listbox

```
procedure TForm2.Button4Click(Sender: TObject);
begin
list.Delete(listbox1.ItemIndex);
listbox1.Items.Delete(listbox1.ItemIndex);
end;
```

حذف ایتم از لیست و Listbox بر اساس index ایتم انتخاب شده در listbox

کلاس Tlist

این کلاس یکی از کلاسهای مهم دلفی می باشد که همانند ListArray در دات نت عمل میکند باکمی تفاوت این کلاس یک متد بنام add دارد که از نوع اشاره گر می باشد و میتواند به هر چیزی اشاره کند از تغییر گرفته تا کلاس ابجکت ادرس توابع و ...

شما میتوانید هر چیزی در این لیست قرار دهید

برخی از توابع پر کاربرد این کلاس

Count

شمارش تعداد ایتم های موجود در لیست

Items

این متد مقداری از نوع integer که معرف index ایتم ما در لیست می باشد را دریافت کرده و مقداری از نوع اشاره گر که محلی است که به ان اشاره میکند را برمیگرداند

List

ایتمهای موجود در لیست را به یک ارایه تبدیل میکند

Add

اضافه کردن یک ایتم به انتهای لیست این متد ۱ پارامتر از نوع اشاره گر دارد

Assign

پرکردن محتویات ۱ لیست از لیست دیگر

Clear

خالی کردن محتویات یک لیست

Delete

حذف یک ایتم از لیست که یک پارامتر از نوع integer دارد که index ایتم مورد نظر می باشد

Move

انتقال یک ایتم موقعیت مکانی دیگر این تابع ۲ پارامتر از نوع integer دارد اولی موقعیت مکانی جاری ایتم و دومی موقعیت جدیدی که میخواهید ایتم به انجا منتقل شود

Insert

اضافه کردن یک ایتم به محل مورد نظر این متد ۲ پارامتر دارد اولی از نوع اشاره گر که به محتویات ما اشاره میکند دومی از نوع integer که موقعیت مکانی در لیست می باشد

Exchange

جابجایی ۲ ایتم باهم که ۲ پارامتر از نوع integer دارد که نشان دهنده index ۲ ایتم میباشد

First

اشاره به اولین ایتم در لیست دارد

Last

اشاره به اخرین ایتم در لیست دارد

IndexOf

با دادن مقدار مورد نظر انرا در لیست جستجو میکند در صورت یافت index انرا برمیگرداند

مثال:

```
var
  Form2: TForm2;
  list:tlist;
```

ایجاد یک متغییر عمومی از کلاس tlist

```
implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
var
p:pinteger;
i:integer;
begin
i:=StrToInt(Edit1.Text);
```

قرار دادن رشته عددی در متغییر عددی

```
new(p);
```

تخصیص فضا برای اشاره گر از نوع integer

```
p^:=i;
```

مقدار دهی P از I

```
list.Add(p);
```

اضافه کردن P به لیست

```
end;

procedure TForm2.Button2Click(Sender: TObject);
var
i,j:integer;
```

```
begin
j:=0;
i:=strtoint(edit2.Text);
```

قرار دادن رشته عددی در i

```
if i<= list.Count-1 then
begin
```

چک کردن قرار داشتن i در محدوده تعداد ایتمها

```
j:=pinteger(list.Items[i])^;
```

قرار دادن مقدار موجود در ایتم مورد نظر در j

```
showmessage(inttostr(j));
end;
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
list:=tlist.Create;
end;
```

تخصیص فضا به متغییر list

```
procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
list.Free;
end;
```

ازاد کردن فضای تخصیص داده شده به list

این هم مثالی دیگر

قرار دادن کنترل در Tlist

```
var
  Form2: TForm2;
  list:tlist;
implementation
```

```
{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
var
i:integer;
button:tbutton;
begin
for I := 0 to 10 do
begin
button:=tbutton.Create(self);
button.Parent:=Form2;
button.Caption:='Button'+inttostr(i);
button.Left:=i*50;
button.Top:=1*30;
list.Add(button);
end;
end;
procedure TForm2.Button2Click(Sender: TObject);
begin
ShowMessage(tbutton(list.Items[StrToInt(edit1.Text)]).Caption);
end;

procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
list.Free;
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
list:=tlist.Create;
end;
```