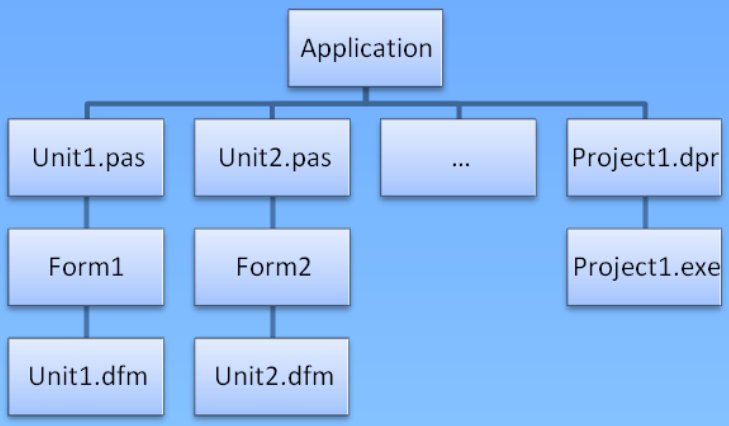
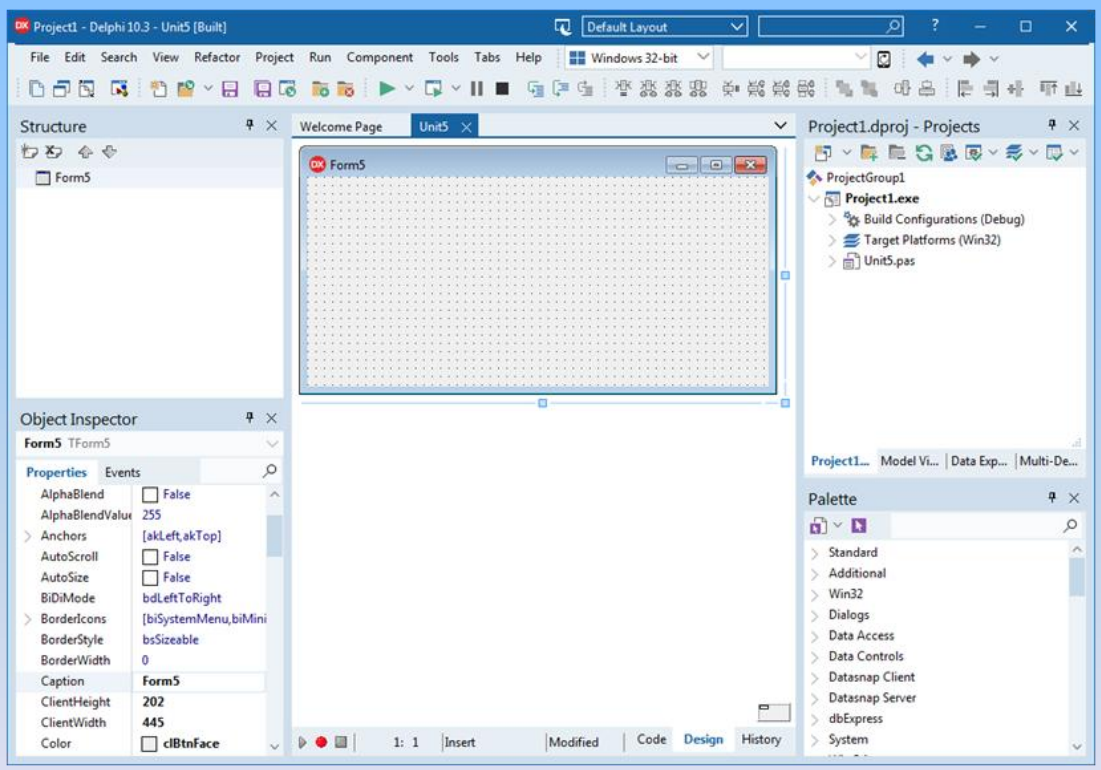
جزوه آموزشی دلفی

نیمسال دوم ۹۷-۹۸





علی سلیمانی

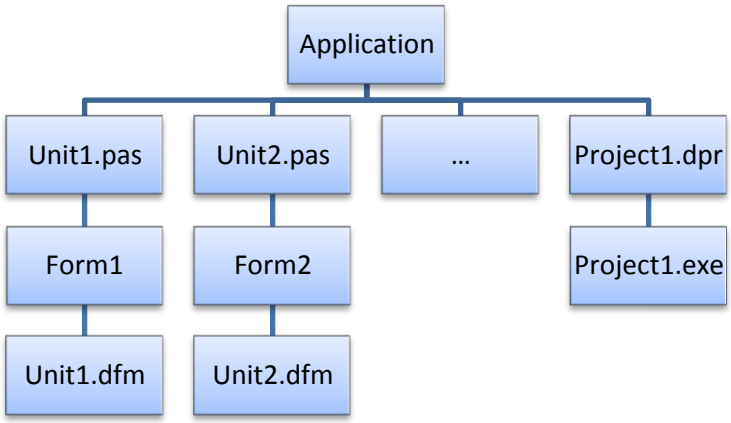
عضو هیئت علمی دانشگاه آزاد واحد اصفهان (خوراسگان)

فهرست مطالب

مقدمه ای بر دلفی

ساختار پروژه

پیش از وارد شدن به بخش طراحی و کدنویسی لازم است با ساختار یک پروژه یا اپلیکیشن آشنا شویم. هر اپلیکیشن از تعدادی یونیت بدون فرم، تعدادی یونیت دارای فرم و یک فایل پروژه تشکیل میشود. فرم، پنجره‌ای برای طراحی و چیدن کامپوننتها و یا ترسیمهای گرافیکی است. یونیت یک فایل متنی برای کدنویسی است و فایل پروژه هم یک فایل متنی دیگر برای کدنویسی است که اغلب IDE دلفی آن را کنترل میکند. در فایل پروژه دستورهایی برای تخصیص حافظه به فرمها و اجرای اپلیکیشن دیده میشود. نمودار زیر ساختار یک اپلیکیشن را نشان میدهد (فرض اولیه بر این است که همه یونیتها دارای فرم باشند):



هر یونیت (بگیرید Unit1) در فایلی با نام خود یونیت و پسوند pas (Unit1.pas) بر روی دیسک ذخیره میشود. و چنانچه یونیت دارای فرم باشد مشخصات گرافیکی فرم و کامپوننتهای درج شده بر آن در فایلی با نام یونیت (و نه فرم) و پسوند dfm (Unit1.dfm) ذخیره میشوند. فایل پروژه (بگیرید Project1) هم فایلی با پسوند dpr است (Project1.dpr). هنگامی که یک پروژه ترجمه یا کامپایل میشود به ازای هر یونیت و فرم متناظر با آن فایلی با پسوند dcu بر روی دیسک ایجاد میشود (Unit1.dcu). سپس این فایلها در هم ادغام شده و کد اجرایی در فایلی با نام فایل پروژه و پسوند exe (Project1.exe) بر روی دیسک ایجاد میشود.

روشهای نامگذاری

در یک اپلیکیشن، فرم و یونیتها باید نام یکتا داشته باشند. برای نمونه یک پروژه نمیتواند دو فرم با نام Form1 یا دو یونیت با نام Unit1 داشته باشد و چنانچه Form1 یکی از فرمهای پروژه باشد هیچ یونیتی (از جمله یونیت متناظر با آن) نمیتواند با نام Form1 ذخیره شود. از طرفی چون هر یونیت و فرم متناظر با آن برای هدف مشترکی ایجاد میشوند، بهتر است در نامگذاری بخش مشترکی داشته باشند که ارتباط آنها با هم مشخص شود. IDE دلفی در ابتدا این ارتباط را با شماره یکسان مشخص میکند (Form3, Unit3) ولی هنگام ذخیره‌سازی بهتر است از نامهای معنی‌دار برای بخش مشترک استفاده کنید. در یک شیوه نامگذاری میتوانید یک نام مشترک برای یونیت و فرم متناظر با آن انتخاب کرده و به انتهای هر کدام Unit یا Form بچسبانید (MainForm, MainUnit) و در شیوه دیگر میتوانید یونیتها را با U و فرمها را با F آغاز کرده و سپس بخش مشترک را به کار بگیرید (FMain, UMain).

ساختار یونیت

هر شناسه باید با یکی از واژه‌های کلیدی type ، const ، var ، procedure و function **به عنوان نوع، ثابت، متغیر، پروسجر و تابع تعریف شود تا بعدا در تعریف شناسه‌های بعدی یا در بخش بدنه یک زیربرنامه یا متد به کار گرفته شود. یونیت مکانی برای تعریف شناسه‌ها و بخشهای کدنویسی است و از دو بخش اصلی واسط (interface) و پیاده‌سازی (implementation) تشکیل میشود. ساختار یک یونیت در ساده‌ترین حالت معادل UnitA و در حالت پیشرفته‌تر با افزودن بخشهای اختیاری (uses ، initialization و finalization) شبیه UnitB است:**

```
unit UnitA;
interface
تعریف شناسه‌های عمومی (قابل رویت)
implementation
تعریف شناسه‌های خصوصی (پنهان) و بخش کدنویسی
end.
```

```
unit UnitB;
interface
uses لیست یونیتها;
تعریف شناسه‌های عمومی (قابل رویت)
implementation
uses لیست یونیتها;
تعریف شناسه‌های خصوصی (پنهان) و بخش کدنویسی
initialization
بخش مقداردهی اولیه
finalization
بخش خاتمه‌سازی
end.
```

بخش واسط تعریف شناسه‌های عمومی (قابل رویت) را شامل میشود به طوری که سایر یونیتها با به کارگیری عبارت uses (یونیتهای میزبان) میتوانند به آنها دسترسی داشته باشند. و بخش پیاده‌سازی هم تعریف شناسه‌های خصوصی (پنهان) و همچنین بخشهای کدنویسی را شامل میشود. سایر یونیتها نمیتوانند به شناسه‌های خصوصی دسترسی داشته باشند. دو بخش اختیاری مقداردهی اولیه و خاتمه‌سازی فقط کدنویسی‌ها را شامل هستند و به ترتیب در آغاز و پایان پروژه اجرا میشوند.

در حالت پیش فرض تعریفهای بخش واسط بین دو واژه کلیدی interface و implementation و تعریفها و کدنویسی‌های بخش پیاده‌سازی هم بین دو واژه کلیدی implementation و end قرار میگیرند. ولی چنانچه یونیتها دارای بخشهای اختیاری باشند تعریفهای بخش واسط باید بعد از جمله uses و تعریفها و کدنویسی‌های بخش پیاده‌سازی هم باید بعد از جمله uses و قبل از واژه initialization قرار بگیرند. هر شناسه باید قبل از استفاده تعریف شود. بنابراین تا زمانی که این قانون رعایت شود شناسه‌ها در هر دو بخش واسط و پیاده‌سازی با هر ترتیب دلخواهی میتوانند تعریف شوند.

معرفی زیربرنامه‌ها در یونیت

تعریف یک زیربرنامه بدون بخش بدنه (بخش عنوان) یک تعریف معتبر است و در دو حالت لازم است عنوان یک زیربرنامه برای سایر بخشهای برنامه‌نویسی معرفی شود. در چنین حالتی باید بخش عنوان همانند سایر شناسه‌ها در یکی از دو بخش واسط یا پیاده‌سازی تعریف شده و متن کامل آن در بخش پیاده‌سازی و بعد از معرفی عنوان قرار بگیرد.

۱- به اشتراک گذاری: چنانچه بخواهید از یک زیربرنامه در سایر یونیتها استفاده کنید، باید یک کپی از عنوان آن را در بخش واسط قرار دهید:

```
unit Unit1;

interface

procedure MyProc;

implementation

procedure MyProc;
begin
  // Some sode
end;

end.
```

۲- فراخوانی چرخه‌ای: فرض کنید دو یا چند زیربرنامه بخواهند همدیگر را فراخوانی کنند. از آنجاییکه هر شناسه قبل از استفاده باید معرفی شود پس تعریف هر کدام باید قبل از دیگری انجام بگیرد که امکانپذیر نیست. برای رفع این مشکل باید عنوان برخی از آنها زودتر در بخش واسط یا پیاده‌سازی معرفی شود. معرفی عنوانها در بخش واسط اگرچه مشکل را حل میکند ولی آنها را برای سایر یونیتها قابل رویت میکند. بنابراین در صورتی که نیاز به اشتراک‌گذاری نباشد باید عنوانها در بخش پیاده‌سازی با واژه کلیدی forward معرفی شوند:

```
implementation
```

```
....

procedure Q; forward;

procedure P;
begin
  // Some code that calls Q
end;

procedure Q;
begin
  // Some code that calls P
end;
```

وابستگی یونیتها

با توجه به ساختار یونیت، جمله uses **میتواند در هر دو بخش واسط و پیاده‌سازی به کار گرفته شود. اکنون فرض کنید یونیت** A **(به عنوان میزبان) میخواهد با جمله** uses **بخشی از شناسه‌های تعریف شده در بخش واسط یونیت** B **را به کار بگیرد. در این صورت اگر دست کم یکی از آن شناسه‌ها در بخش واسط یونیت** A **(در تعریف شناسه دیگری) استفاده شود، لازم است جمله "uses B" در بخش واسط یونیت** A **قرار بگیرد وگرنه هر دو بخش واسط و پیاده‌سازی برای قرارگیری این جمله مجاز هستند. وقتی یونیت** A **از تعریفهای** B **استفاده میکند،** A **به** B **وابسته است و بخش واسط** B **باید قبل از جمله "uses B" کامپایل شود.**

اکنون فرض کنید یونیت B **هم بخواهد با جمله** uses **بخشی از شناسه‌های تعریف شده در بخش واسط یونیت** A **را به کار بگیرد. در این حالت** B **هم به** A **وابسته است و بخش واسط** A **باید قبل از جمله "uses A" کامپایل شود. اینگونه وابستگیها یک چرخه ایجاد میکند. چرخه‌ها میتوانند مجاز و یا غیر مجاز باشند. برای تشخیص مجاز بودن چرخه فرض کنید همه وابستگیها در یک گراف وابستگی قرار بگیرد، در این صورت اگر گراف دارای چرخه باشد و همه گره‌های آن از بخش واسط گذر کرده باشند آن چرخه غیرمجاز وگرنه مجاز است. به بیان دیگر اگر گراف وابستگی با نادیده گرفتن بخش پیاده‌سازی یونیتها و فقط از روی بخش واسط آنها ساخته شده و دارای چرخه باشد آن چرخه غیر مجاز است. برای نمونه چرخه زیر غیر مجاز است:**

```
unit A;

interface

uses B;

implementation

end.
```

```
unit B;

interface

uses C;

implementation

end.
```

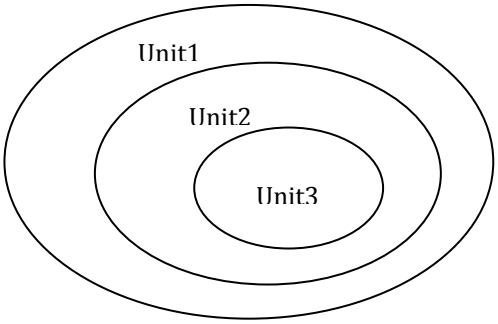
```
unit C;

interface

uses A;

implementation

end.
```

برای مجاز کردن چرخه کافی است یکی از جمله‌های uses **در گراف وابستگی به بخش پیاده‌سازی انتقال یابد. از آنجاییکه تنها وابستگیهای بخش واسط میتوانند منجر به چرخه غیرمجاز شوند توصیه میشود در موارد غیر ضروری جمله** uses **در بخش پیاده‌سازی قرار بگیرد.**

ترتیب یونیتها در جمله uses

ترتیبی قرارگیری یونیتها در جمله uses روی ترتیب اجرای بخشهای مقداردهی اولیه و ترتیب شناسایی شناسه‌ها اثر میگذارد. بخشهای مقداردهی اولیه با همان ترتیب آمده در لیست اجرا میشوند ولی ترتیب شناسایی شناسه‌ها برعکس ترتیب لیست است. برای نمونه برای جمله uses زیر که در یونیت Unit3 (میزبان) به کار رفته میدان دید شناسه‌ها به صورت زیر است:

```
unit Unit3;
...
uses Unit1, Unit2;
```



بنابراین اگر شناسه‌ای مثل X که در یونیت میزبان به کار رفته در هر دو یونیت و یا در یونیت میزبان و یکی از یونیتها تعریف شده باشد، نزدیکترین تعریف آن در نظر گرفته میشود. برای دسترسی به شناسه دورتر باید نام یونیت را به آن اضافه کنید (UnitName.X).

```
unit Unit1;

interface

var A: Integer;

implementation

end.
```

```
unit Unit2;

interface

var A, B: Char;

implementation

end.
```

```
unit Unit3;

interface

uses Unit1, Unit2;

var B: Boolean;

implementation

initialization
  A:= 'K';  // in Unit2
  Unit1.A:= 5;
  B:= True; // in Unit3
  Unit2.B:= 'G';
end.
```

یونیتهای بدون فرم و دارای فرم

همانگونه که اشاره شد یونیتها میتوانند دارای فرم و یا بدون فرم باشند. برای ایجاد یک یونیت بدون فرم میتوانید از منوی File گزینه New→Unit استفاده کنید، در حالی که یونیتهای دارای فرم همزمان با فرم و با گزینه New→Form ایجاد میشوند. یونیتهای بدون فرم برای ساختن کتابخانه و به اشتراک گذاشتن شناسه‌ها و زیربرنامه‌ها مفید هستند ولی یونیتهای دارای فرم اغلب برای بخش کدنویسی فرم و دنبال کردن نیازهای مرتبط با فرم ایجاد میشوند. ساختار یک یونیت دارای فرم در آغاز کار به صورت زیر است:

```
unit Unit1;

interface

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

type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```

همانگونه که مشاهده میشود دلفی تعریفهای خود را در بخش واسط وارد میکند، بنابراین بهتر است شما تعریف‌های خود را در بخش پیاده‌سازی انجام دهید.

مدیریت پروژه

در اینجا یک اپلیکیشن شامل ۴ فرم را ایجاد میکنیم تا عملیات اولیه بر روی پروژه و فرمهای آن را یاد بگیرید.

ایجاد پروژه: **از منوی فایل گزینه New→VCL Application را انتخاب کنید تا یک پروژه جدید شامل یک فرم و یونیت متناظر با آن ایجاد شود.**

تاگل کردن بین فرم و یونیت: **با کلید F12 یا آیکن Toggle (روی نوار ابزار) میتوانید بین فرم و یونیت متناظر با آن تاگل کنید.**

افزودن فرم جدید به پروژه: **با گزینه New→VCL Form بر منوی فایل سه فرم دیگر هم به پروژه اضافه کنید.**

مشاهده لیست فرمها و یونیتها: **برای مشاهده لیست فرمها و یافتن آنها از کلید Shift-F12 یا آیکن View Form (روی نوار ابزار) و برای مشاهده لیست یونیتها و یافتن آنها از کلید Ctr-F12 یا آیکن View Unit (روی نوار ابزار) استفاده کنید. از آنجاییکه هر یونیت و فرم متناظر با آن اغلب با بخش مشترک نامگذاری میشوند (FMain, UMain) حفظ کردن یکی از این دو کلید یا آیکن کافی است (برای یافتن جفت آن میتوانید تاگل کنید). در چنین حالتی کلید Ctr-F12 مفیدتر است چون همه یونیتها (دارای فرم و بدون فرم) را لیست میکند.**

اجرای پروژه: **برای اجرای پروژه (به همراه دیباگ) از کلید F9 یا آیکن Run (روی نوار ابزار) و برای اجرای بدون دیباگ پروژه از کلید Shift-Ctr-F9 یا آیکن Run Without Debugging (روی نوار ابزار) استفاده کنید. در حالت «با دیباگ» امکان تریس کردن برنامه برای یافتن خطاهای زمان اجرا و همچنین امکان توقف برنامه (در حالتهای ضروری**

مثل هنگ کردن برنامه) وجود دارد. با اجرای برنامه، اولین فرمی که ایجاد کرده‌اید (فرم اصلی) نمایش داده میشود. برای خروج از اجرا باید فرم اصلی را ببندید (روی آیکن × فرم کلیک کنید).

کمینه‌سازی IDE در اجرا: اغلب دانشجویان مبتدی بدون خروج از اجرا (بستن فرم اصلی) شروع به توسعه پروژه میکنند. در این موارد دو اشکال ممکن است پیش بیاید: چنانچه اجرای قبلی بدون دیباگ باشد، در اجرای بعدی امکان ساختن کد اجرایی (exe) از بین رفته و خطای زیر اعلام میشود:

```
Could not create output file "…\Project1.ext"
```

و چنانچه اجرای قبلی با دیباگ باشد، هر دو آیکن Run خاموش شده و کلیدهای مربوطه هم عمل نمیکنند. برای پرهیز از این اشتباه (توسعه بدون خروج) بهتر است IDE دلفی را برای هر اجرا کمینه کنید تا هنگام بازگشت مجدد به IDE متوجه بستن فرم اصلی بشوید. برای این منظور گزینه «Minimize on run» را در پنجره Tools→Options جستجو کرده و آن را علامت بزنید.

خروج اضطراری: چنانچه به دلیل هنگ کردن پروژه نتوانید از اجرای با دیباگ برنامه خارج شوید روی IDE دلفی متمرکز شده و کلید Ctr-F2 را فشار دهید یا روی آیکن Program Reset (روی نوار ابزار) کلیک کنید. و چنانچه نتوانید از اجرای بدون دیباگ برنامه خارج شوید در این حالت به دلیل اینکه کنترل اجرا از دست دلفی خارج شده و در اختیار ویندوز قرار گرفته است باید با Winodw Task Manager پروژه خود را یافته و به اجرای آن خاتمه دهید.

ذخیره‌سازی پروژه: پیش از ادامه کار با پروژه لازم است آن را بر روی دیسک ذخیره کنید. برای این منظور ابتدا یک فلدر با نام مناسب در مکانی از دیسک ایجاد کنید و سپس به کمک کلید Shift-Ctr-S یا آیکن Save All (روی نوار ابزار) آن را در فلدر ایجاد شده ذخیره کنید. مشاهده خواهید کرد که ۵ نام از شما پرسیده میشود. برای هر فرم و یونیت متناظر با آن یک نام پرسیده میشود (مثل Unit1) و برای فایل پروژه نیز یک نام پرسیده میشود (مثل Project1). (توجه داشته باشید که کلید Ctr-S یا آیکن Save روی نوار ابزار تنها یک فرم و یونیت متناظر با آن را ذخیره میکند).

حذف فرم از پروژه: چنانچه بخواهید فرمی را از پروژه حذف کنید از آیکن Remove file from project بر روی نوار ابزار استفاده کنید. توجه داشته باشید که این آیکن تنها فرم را از پروژه حذف میکند ولی آن را از روی دیسک حذف نمیکند. شما فرم چهارم را از پروژه حذف کنید سپس با کلیدهایی که اشاره شد لیست فرمها و یونیتها را مشاهده کنید.

افزودن فرم موجود به پروژه: چنانچه بخواهید فرمی که از پروژه حذف شده است را دوباره به پروژه برگردانید و یا فرمی که از یکی از دوستانتان دریافت کرده‌اید را به پروژه خود اضافه کنید از آیکن Add file to project روی نوار ابزار استفاده کنید. شما فرم چهارم که از پروژه حذف کردید را دوباره به پروژه برگردانید سپس با کلیدهایی که اشاره شد لیست فرمها و یونیتها را مشاهده کنید.

بستن پروژه: هنگامی که کار کردن با یک پروژه تمام میشود باید آنرا ببندید. برای بستن یک پروژه از گزینه Close All در منوی File استفاده کنید. اگر پروژه ذخیره نشده باشد پیش از بستن برای ذخیره‌سازی آن پرسش میکند (توجه داشته باشید که گزینه Close در منوی File تنها یک فرم و یونیت متناظر با آن را ذخیره میکند).

بازکردن پروژه: برای باز کردن پروژه‌ای که از پیش بسته شده باشد در بخش Open Recent صفحه Welcome Page نام پروژه خود را پیدا کرده و روی آن کلیک کنید. در مقابل نام هر پروژه در بخش Open Recent گزینه "Add to favorites" دیده میشود. برای پروژه‌های دلخواه و پرکاربرد خود روی این گزینه کلیک کنید تا نام پروژه به بخش Favorite Projects انتقال یابد. پس از انتقال، پروژه خود را از این بخش انتخاب کنید. همچنین برای باز کردن پروژه‌ای که آن را قبلا در IDE باز نکرده‌اید و یا پروژه‌ای که آن را از دوست خود دریافت کرده‌اید از آیکن Open Project (روی نوار ابزار) استفاده کنید. با کلیک روی این آیکن فایل dpr از شما پرسیده میشود. توجه داشته باشید که آیکن Open روی نوار ابزار هر نوع فایلی را باز میکند و روش مناسبی برای باز کردن پروژه نیست (چنانچه اشتباها فایل یونیت را به جای فایل پروژه باز کنید هر دو آیکن Run بر روی نوار ابزار خاموش شده و امکان اجرای پروژه وجود ندارد).

مشاهده فایل پروژه: برای مشاهده فایل پروژه از منوی Project گزینه View Source را انتخاب کنید. فایل پروژه‌ای که شامل ۴ فرم باشد، ساختاری به صورت زیر دارد:

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1},
  Unit2 in 'Unit2.pas' {Form2},
  Unit3 in 'Unit3.pas' {Form3},
  Unit4 in 'Unit4.pas' {Form4};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm4, Form4);
  Application.Run;
end.
```

هر یک از دستورهای CreateForm حافظه‌ای را به یکی از فرمها تخصیص میدهد. فرمی که زودتر از همه تخصیص حافظه داده میشود فرم اصلی است و هنگام اجرا نمایش داده میشود. سایر فرمها باید توسط فرم اصلی فراخوانی و نمایش داده شوند.

تعیین فرم اصلی: اگر بخواهید فرمی دیگری را به عنوان فرم اصلی انتخاب کنید باید دستورهای CreateForm را جابجا کنید. ولی روش بهتر این است که گزینه «Main Form» را در پنجره Project→Options جستجو کرده و فرم دلخواه خود را به آن نسبت دهید.

کامپوننتها

اشیایی که بر روی فرم درج میکنید کامپونت نام دارند. کامپوننتها به دو دسته ویژوال و غیرویژوال تقسیم‌بندی میشوند. کامپوننتهای ویژوال که کنترل هم نام دارند آنهایی هستند که در زمان طراحی و اجرا نمایش مخصوص خود را دارند (همانند Button, Label, Edit). و کامپوننتهای غیرویژوال آنهایی هستند که در زمان طراحی به صورت یک آیکن دیده

میشوند و در زمان اجرا پنهان میشوند (همانند MainMenu, ActionList, OpenDialog). کنترلها به دو دسته پنجره‌ای و گرافیکی تقسیم میشوند. کنترل‌های پنجره‌ای آنهایی هستند که در زمان اجرا میتوان به کمک صفحه کلید بر روی آنها متمرکز شد (همانند Button, Edit, CheckBox) یا میتوانند در برگیرنده کنترلهای دیگر به نام کنترلهای فرزند باشند (همانند Form, Panel, ToolBar). کنترلهایی که مشخصات پیش را نداشته باشند کنترلهای گرافیکی هستند. کنترلهای گرافیکی سریعتر هستند و دارای منابع کمتری میباشند (همانند Image, Label, Shape, SpeedButton). برای درج کامپوننت بر روی فرم از پنجره ابزار پالت (Palette) یا نوار ابزار کامپوننت (Component) استفاده میشود که در بخشهای بعدی به جزئیات آنها خواهیم پرداخت.

روشهای انتخاب چند کامپوننت

گاهی لازم است عملیات مشترکی را روی چند کامپوننت درج شده بر روی فرم و یا یک کنترل پنجره‌ای انجام دهیم (مثل: تغییر خواص مشترک، تنظیم فاصله، تنظیم مکان و تراز کردن). در این موارد لازم است قبل از انجام عملیات آنها را انتخاب کنیم. دو روش برای این منظور وجود دارد:

انتخاب همزمان: با کشیدن ماوس روی یک ناحیه مستطیلی از فرم کامپوننتهای درون آن ناحیه را همزمان انتخاب کنید. چنانچه کامپوننتها روی یک کنترل پنجره‌ای درج شده باشند هنگام کشیدن ماوس کلید Ctrl را پایین نگه دارید. بستگی به روش انتخاب یکی از کامپوننتها سرگروه میشود.

انتخاب تک تک: اولین کامپوننت را با کلیک ماوس انتخاب کرده و بقیه را با ترکیب Shift-Click به لیست انتخابی اضافه کنید. اولین کامپوننت انتخاب شده سر گروه است.

اولین کامپوننت در لیست انتخاب شده سرگروه است و در انجام عملیاتی مثل تغییر خواص مشترک و تراز کردن محور قرار میگیرد.

روشهای تغییر مکان کامپوننت

دو روش برای تغییر مکان یک یا چند کامپوننت انتخاب شده وجود دارد:

با ماوس: با کشیدن ماوس کامپوننت مورد نظر را روی شبکه نقاط جابجا کنید. برای جابجایی آزاد (از شبکه) کلید Alt را پایین نگه دارید.

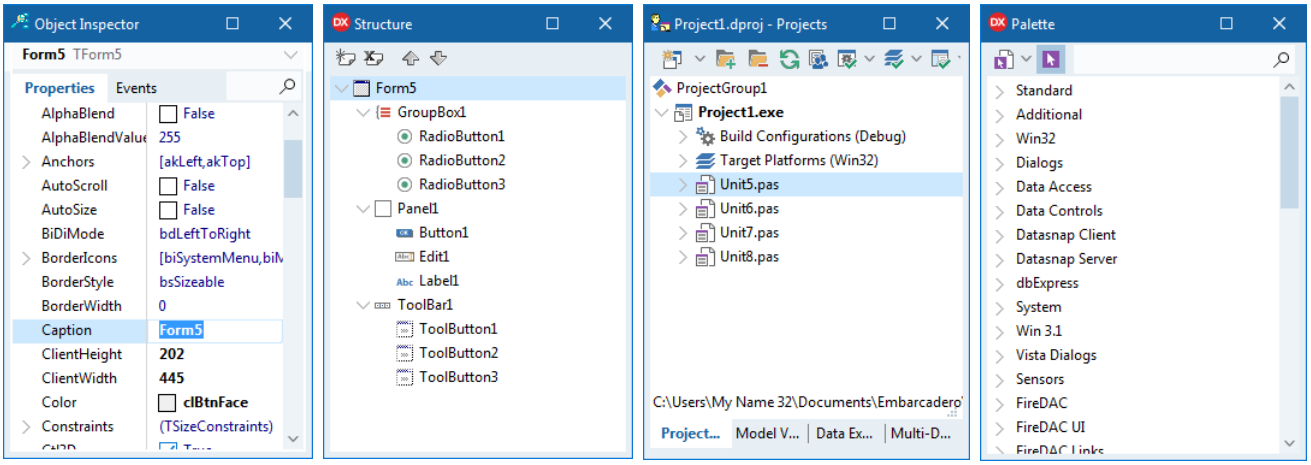
با صفحه کلید: با ترکیب Shift-Ctrl-Arrows کامپوننت مورد نظر را روی شبکه نقاط و با ترکیب Ctrl-Arrows آن را به صورت آزاد (از شبکه) جابجا کنید.

روشهای تغییر اندازه کامپوننت

دو روش برای تغییر اندازه یک یا چند کامپوننت انتخاب شده وجود دارد: با کشیدن لبه‌های کامپوننت و یا با ترکیب Shift-Arrows کامپوننت مورد نظر را تغییر اندازه دهید.

پنجره ابزارهای مهم

در این بخش پنجره ابزارهای مهمی که در مدیریت و توسعه یک پروژه به کار میرند را معرفی میکنیم. خوشبختانه همه آنها به صورت پیش فرض آشکار هستند. با اینحال در صورت پنهان بودن یک پنجره ابزار میتوانید از گزینه Tool windows در منوی View برای نمایش آن استفاده کنید.



پنجره ابزار ناظر اشیاء

پس از درج یک کامپوننت بر روی فرم اغلب لازم است مشخصات آن را تغییر دهید و یا برای محاوره با کاربر عملیاتی را بر روی آن تعریف کنید. برای این دو منظور از پنجره ابزار ناظر اشیاء (Object Inspector) استفاده میشود که در صورت پنهان بودن با کلید F11 نمایش داده میشود. این پنجره از دو تب Properties (خواص) و Events (رویدادها) تشکیل میشود.

صفحه خواص: در صفحه خواص که مشخصات کامپوننت مورد نظر را لیست میکند نکات زیر قابل توجه است:

- در کنار برخی از خاصیتها (همانند BorderIcons, Font) علامت ">" (در نسخه‌های قبلی "+") دیده میشود. این خاصیتها خود دارای ویژگیهایی هستند که برای تغییر آنها باید باز شوند. خاصیتهایی که مقدار آنها از یک مجموعه محدود گرفته میشود (همانند Position, WindowState) به صورت جعبه‌های کامبو باز میشوند. این خاصیتها با دو بار کلیک ماوس تغییر مقدار میدهند. روبروی برخی از خاصیتها علامت «...» دیده میشود. با کلیک کردن روی این علامت پنجره‌ای برای تغییر مقدار آن باز میشود.
- هرگاه گروهی از کامپوننتها را با هم انتخاب کرده باشید خاصیتهای مشترک آنها در این پنجره لیست میشود به طوری که تغییر مقدار یک خاصیت مشترک، همه کامپوننتهای انتخاب شده را تحت تاثیر قرار میدهد.
- هنگامی که مقدار یک خاصیت تغییر میکند فونت آن برای شناسایی تغییر رنگ میدهد.

صفحه رویدادها: به کمک رویدادها میتوانید به کلیکهای ماوس، ضربه‌های صفحه کلید، گذشت زمان و سایر رویدادهایی که در زمان اجرا به وجود میآید پاسخ‌گویی کنید. برای پاسخ‌گویی به رویدادی از یک کامپوننت باید نخست روی کامپوننت مورد نظر متمرکز شوید سپس در جعبه روبروی آن رویداد دوبار کلیک کنید. با انجام این کار

یک زیربرنامه با بدنه خالی در یونیت متناظر با آن فرم ایجاد میشود. شما باید فرمانهای خود را در بخش بدنه و مابین دو کلمه begin و end وارد کنید.

البته برخی از رویدادها پیش فرض هستند و میتوانید مستقیما با دوبار کلیک بر روی کامپوننت مورد نظر برای آنها زیر برنامه‌ای ایجاد کنید. برای مثال رویداد OnClick دگمه (Button) یا رویداد OnCreate فرم رویدادهای پیش فرض هستند.

پیش از ادامه کار لازم است زیربرنامه‌ای را برای پاسخ‌گویی به رویدادی از یک کامپوننت ایجاد کنید. شما یک دگمه بر روی فرم درج کنید و به روشی که گفته شد زیر برنامه‌ای برای رویداد OnMouseDown آن ایجاد کنید و فرمان زیر را برای آن وارد کنید:

```
procedure TForm1.Button1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  Color:= clRed;
end;
```

سپس زیر برنامه دیگری برای رویداد OnMouseUp آن ایجاد کنید و فرمان زیر را برای آن وارد کنید:

```
procedure TForm1.Button1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  Color:= clBlue;
end;
```

اکنون پروژه را اجرا کنید و در زمان اجرا روی دگمه درج شده ماوس را فشار داده و رها کنید. مشاهده خواهید کرد که با هر بار فشار ماوس رنگ فرم قرمز و با هر بار رها کردن آن رنگ فرم آبی میشود.

برای حذف یک رویداد باید فرمانهایی که در بخش بدنه زیربرنامه مربوط به آن نوشته‌اید را پاک کنید و سپس یک بار پروژه را با کلید Ctr-F9 کامپایل کرده و یا آن را ذخیره نماید.

پنجره ابزار ساختار

به کمک پنجره ابزار ساختار (Structure) میتوانید ارتباط درختی کامپوننتهای درج شده بر روی فرم را مشاهده کنید. برخی از کامپوننتها (کنترلهای پنجره‌ای) میتوانند در برگیرنده کنترلهای دیگر باشند (مثل Panel، ToolBar و GroupBox). در پنجره ساختار این نوع کنترلها با آیکن ویژه‌ای مشخص میشوند و در صوتی که دربرگیرنده فرزندی باشند در کنار آنها علامت ">" یا "∨" دیده میشود. میتوانید با کشیدن ماوس یک کنترل را از یک کنترل پنجره‌ای به کنترل پنجره‌ای دیگر انتقال دهید. پنجره ساختار همچنین برای پیدا کردن کنترلهایی که در سطح فرم دیده نمیشوند و زیر کنترلهای دیگر قرار دارند به کار میرود. در صورت پنهان بودن این پنجره میتوانید با کلید Alt-Shift-F11 آن را نمایش دهید.

پنجره ابزار پروژه‌ها

به کمک پنجره ابزار پروژه‌ها (Projects) میتوانید ساختار اپلیکشین خود را که از تعدادی یونیت (دارای فرم و بدون فرم) و یک فایل پروژه تشکیل میشود، مشاهده کرده و آن را مدیریت کنید. برخی از عملیاتی که توسط این پنجره یا

آیکنها و گزینه‌های زیرمنوی این پنجره قابل انجام است عبارتند از: مشاهده فرم و یونیت در ویرایشگر، افزودن فرم جدید به پروژه، کامپایل بدون اجرای پروژه، اجرای پروژه، ذخیره‌سازی پروژه، حذف فرم از پروژه، افزودن فرم موجود به پروژه، مشاهده فایل پروژه، مشاهده فایل در مرورگر ویندوز، تغییر نام یونیت یا فایل پروژه.

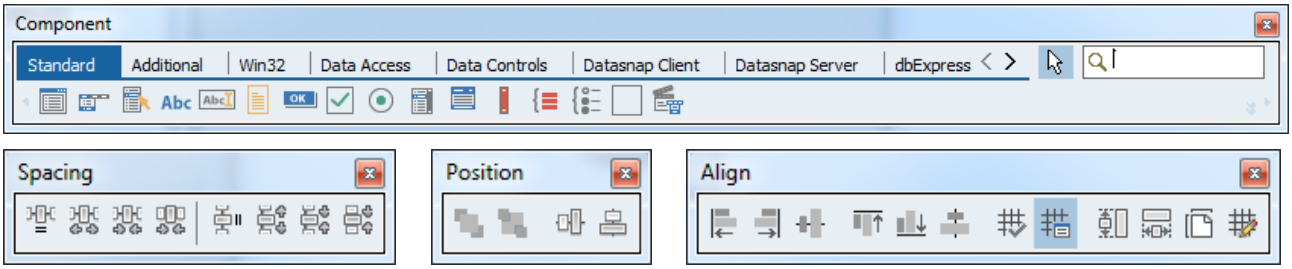
به کمک این پنجره همچنین میتوانید یک گروه پروژه از پروژه‌های موجود ایجاد کرده و آن را مدیریت کنید. مزیت اینکار این است که بدون بستن پروژه فعلی میتوانید از یک پروژه به پروژه دیگر تاگل کنید. و در مواردی که پروژه‌ها از یونیتهای مشترک استفاده کرده باشند با هر تغییر در یونیتهای مشترک میتوانید همه آنها را همزمان کامپایل کنید. در صورت پنهان بودن این پنجره میتوانید با کلید Alt-Ctrl-F11 آن را نمایش دهید.

پنجره ابزار پالت

یک روش برای درج کامپوننت بر روی فرم (یا کنترلهای پنجره‌ای دیگر) استفاده از پنجره ابزار پالت است. کامپوننتها در این پنجره تحت عنوانهایی دسته‌بندی شده‌اند. برای درج یک کامپوننت بر روی فرم نام آن را جستجو کرده و یا روی آن کلیک کنید سپس با کلیک در مکان دلخواهی از فرم کامپوننت مورد نظر در آن مکان درج میشود. چنانچه بخواهید تعداد زیادی از یک کامپوننت را بر روی فرم درج کنید، ابتدا روی کامپوننت مورد نظر Shift+Click کنید سپس با کلیک در هر مکان از فرم یک نمونه از آن کامپوننت در آن مکان درج میشود. برای پایان کار از آیکن "↑" بر روی پالت استفاده کنید. در صورت پنهان بودن این پنجره میتوانید با کلید Alt-Ctrl-P آن را نمایش دهید.

نوار ابزارهای مهم

در این بخش نوار ابزارهای مهمی که در مدیریت و توسعه یک پروژه به کار میرند را معرفی میکنیم. برخی از این نوار ابزارها ممکن است به صورت پیش فرض و به دلیل کمبود فضای بالای پنجره IDE پنهان باشند، ولی اغلب مهم و پرکاربرد هستند. در صورت پنهان بودن یک نوار ابزار میتوانید از گزینه Toolbars در منوی View برای نمایش آن استفاده کنید. البته پس از نمایش اغلب نیاز است با کشیدن ماوس موقعیت آنها را در بخش نوار ابزارها تثبیت کنید.



نوار ابزار کامپوننت

یک روش دیگر برای درج کامپوننت بر روی فرم (یا کنترلهای پنجره‌ای دیگر) نوار ابزار کامپوننت (Component) است. کامپوننتها در این پنجره در چندین تب دسته‌بندی شده‌اند. عمل درج یک یا یک گروه از کامپوننتها از طریق این نوار ابزار همانند پنجره ابزار پالت است. در حالت پیش فرض این نوار ابزار پنهان است ولی میتوانید با گزینه Toolbars→Component در منوی View آن را نمایش داده و از آن استفاده کنید.

نوار ابزار تنظیم فاصله

به کمک نوار ابزار تنظیم فاصله (Sapcing) میتوانید عملیات زیر را بر روی چند کامپوننت انتخاب شده بر روی فرم انجام دهید: یکسان کردن فاصله روی محور افقی یا عمودی، افزایش (یا کاهش) فاصله روی محور افقی یا عمودی و حذف فاصله روی محور افقی یا عمودی. توجه داشته باشید که قبل از استفاده از این نوار ابزار باید تعدادی از کامپوننتهای درج شده بر روی فرم را انتخاب کنید. در صورت پنهان بودن این نوار ابزار میتوانید با گزینه Toolbars→Spacing در منوی View آن را نمایش داده و از آن استفاده کنید.

نوار ابزار تنظیم مکان

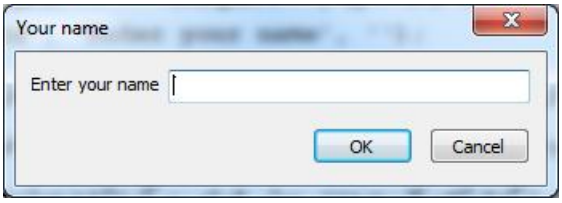
به کمک نوار ابزار تنظیم مکان (Position) میتوانید عملیات زیر را بر روی یک یا چند کامپوننت انتخاب شده بر روی فرم انجام دهید: زیر و رو کردن کامپوننتها نسبت به هم و وسطچین کردن آنها نسبت به محور افقی یا عمودی فرم (یا کنترل پنجره‌ای والد). توجه داشته باشید که عمل وسطچین کردن چند کامپوننت انتخاب شده، بر روی ناحیه مستطیلی حاصل از اجتماع آنها انجام میشود. در صورت پنهان بودن این نوار ابزار میتوانید با گزینه Toolbars→Position در منوی View آن را نمایش داده و از آن استفاده کنید.

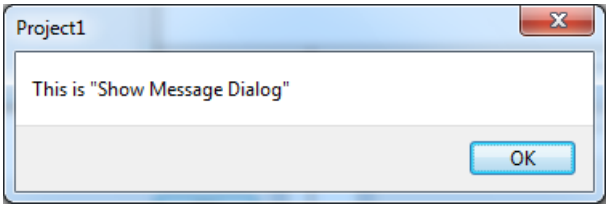
نوار ابزار ترازبندی

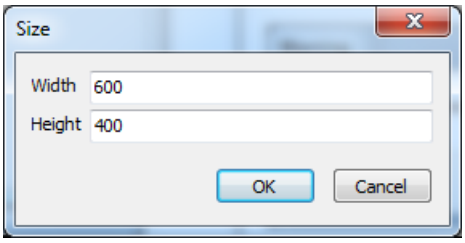
به کمک نوار ابزار ترازبندی (Align) میتوانید عملیات زیر را بر روی یک یا چند کامپوننت انتخاب شده بر روی فرم انجام دهید: چپ چین، راست چین، بالا چین، پایین چین، وسط چین (افقی و عمودی)، هم پهنا کردن، هم ارتفاع کردن، هم اندازه کردن (پهنا و ارتفاع) و تراز کردن نسبت به شبکه نقاط روی فرم. توجه داشته باشید که عملیات چیدن و هم اندازه کردن نسبت به سرگروه (اولین کامپوننت انتخاب شده) انجام میگیرد. در صورت پنهان بودن این نوار ابزار میتوانید با گزینه Toolbars→Align در منوی View آن را نمایش داده و از آن استفاده کنید.

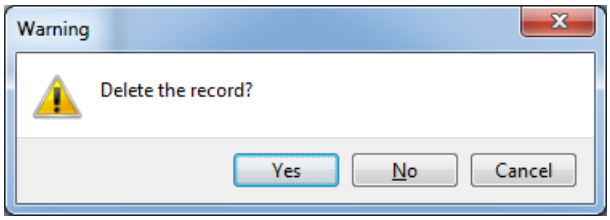
انواع دیالوگ

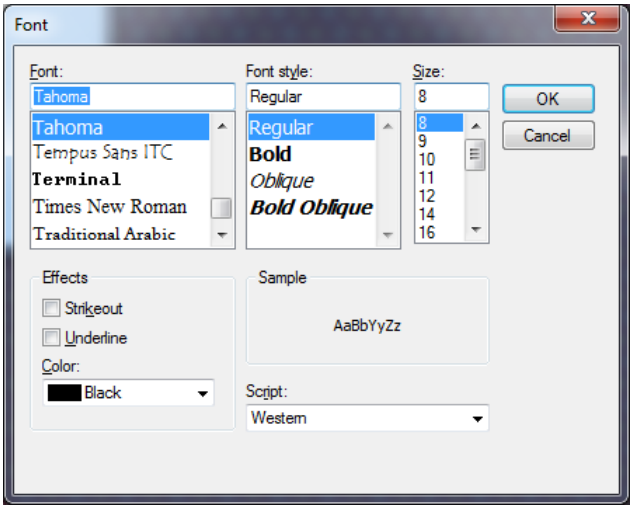
دیالوگها پنجره‌هایی هستند که در زمان اجرا برای محاوره با کاربر (مثل چاپ پیام، دریافت داده و کسب اجازه برای انجام یک پردازش خاص) با فراخوانی تابع و یا با اجرای یک کامپوننت غیر ویژوال نمایش داده میشوند. در این بخش با دیالوگهای مهم و پرکاربرد آشنا میشوید.

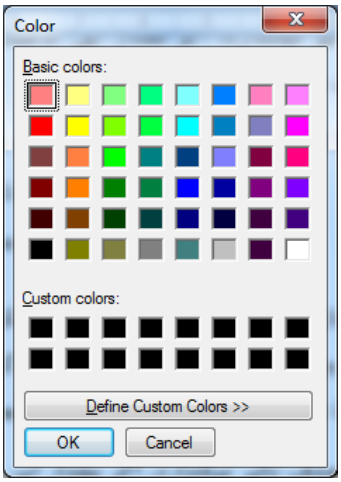


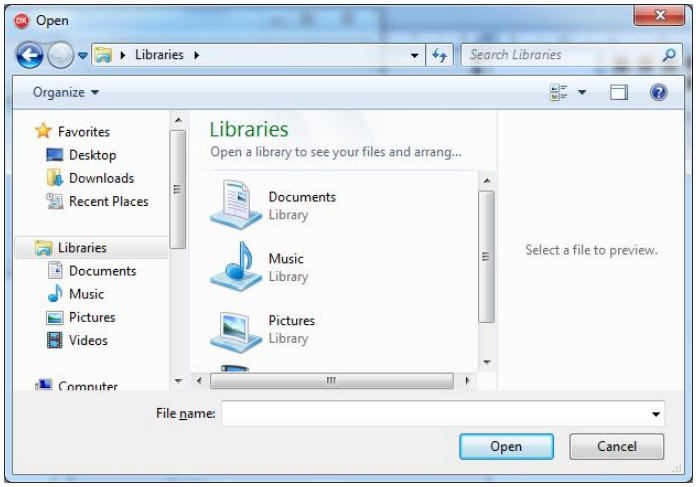












دیالوگ چاپ پیام

ShowMessage ساده‌ترین نوع دیالوگ است که برای چاپ یک پیام ساده به کار رفته و با فرمان زیر فراخوانی میشود:

```
ShowMessage (یک پیام ساده) ;
ShowMessage('This is "Show Message Dialog"');
```

پس از اجرای این فرمان پنجره‌ای با عنوان پروژه و پیام مورد نظر و یک دگمه تایید نمایش داده میشود.

دیالوگ دریافت داده

دیالوگ InputBox برای دریافت یک داده رشته‌ای استفاده شده و با فرمان زیر فراخوانی میشود:

```
StrVar:= InputBox(عنوان پنجره , پیغام مناسب برای ورود داده , مقدار پیش فرض);
S:= InputBox('Your name', 'Enter your name', '');
```

پس از اجرای این فرمان پنجره‌ای با عنوان و پیام مورد نظر، یک جعبه ویرایش شامل مقدار پیش فرض و دگمه‌های تایید و انصراف نمایش داده میشود. اگر کاربر رشته خود را جایگزین رشته پیش فرض کرده و دگمه تایید را انتخاب کند رشته جدید برگشت داده میشود وگرنه با انتخاب دگمه انصراف ورودی پیش فرض برگردانده میشود. بنابراین چنانچه پس از فراخوانی تابع، رشته برگشت شده رشته غیر پیش فرض باشد باید پردازش دلخواه برنامه‌نویس انجام شود.

```
if S<> '' then // Do something
```

از آنجاییکه این دیالوگ تنها داده‌های رشته‌ای را دریافت میکند برای دریافت داده‌های عددی باید یک تبدیل نوع انجام بگیرد:

```
N:= InputBox('Your number', 'Enter your number', '0').ToInteger;
if N<> 0 then // Do something
```

دیالوگ دریافت چند داده

دیالوگ InputQuery **برای دریافت یک یا چند داده رشته‌ای استفاده شده و با فرمان زیر فراخوانی میشود:**

```
BoolVar:= InputQuery(Caption, Prompts, Values);
```

Caption **عنوان پنجره،** Prompts **آرایه‌ای از پیامها و** Values **آرایه‌ای از مقادیر پیش فرض است. طول دو آرایه** Prompts **و** Values **باید یکسان باشد و از آنجاییکه خروجی در** Values **برگشت داده میشود** Values **باید یک متغیر باشد. پس از اجرای این فرمان پنجره‌ای با عنوان** Caption**، پیامهای** Prompts **و برای هر پیام یک جعبه ویرایش با مقدار پیش فرض متناظر در** Values **و دگمه‌های تایید و انصراف نمایش داده میشود. با انتخاب دگمه تایید رشته‌های تغییر یافته در جعبه‌های ویرایش در متغیر** Values **برگشت داده شده و تابع مقدار** True **برمیگرداند و با انتخاب دگمه انصراف مقدار** Values **تغییر نکرده و** False **برگشت داده میشود. بنابراین چنانچه پس از فراخوانی تابع خروجی** True **باشد باید پردازش دلخواه برنامه‌نویس انجام شود.**

```
var Size: array of String;
...
Size:= ['600', '400'];
if InputQuery('Size', ['Width', 'Height'], Size) then // Do Something
```

از آنجاییکه این دیالوگ تنها داده‌های رشته‌ای را دریافت میکند برای دریافت داده‌های عددی باید یک تبدیل نوع روی مقادیر Values **انجام بگیرد:**

```
Size[0].ToInteger
Size[1].ToInteger
```

دیالوگ مجوز گرفتن

دیالوگ MessageDlg **هم برای چاپ یک پیام ساده (اطلاعاتی و خطا) و هم برای کسب مجوز (هشدار و تایید) به کار رفته و با فرمان زیر فراخوانی میشود:**

```
IntVar:= MessageDlg(پیغام , نوع پیغام , مجموعه دگمه‌ها , شماره هلپ);
```

مجموعه دگمه‌ها در یک جفت کروشه معرفی شده و از مقدارهای زیر تشکیل میشود:

```
mbYes, mbNo, mbOK, mbCancel, mbAbort, mbRetry, mbIgnore, mbAll, mbNoToAll,
mbYesToAll, mbHelp, mbClose
```

و خروجی تابع دگمه انتخابی کاربر است که در یک متغیر از نوع صحیح برگردانده میشود تا با یکی از مقدارهای زیر مقایسه شود:

```
mrYes, mrNo, mrOK, mrCancel, mrAbort, mrRetry, mrIgnore, mrAll, mrNoToAll,
mrYesToAll, mrHelp
```

پارامتر آخر شناسه Help طراحی شده را مشخص میکند به طوری که کاربر با کلیک روی دگمه Help و یا فشردن کلید F1 میتواند آن را نمایش دهد. شما در این قسمت مقدار صفر را وارد کنید.

و نوع پیغام هم میتواند یکی از موارد زیر باشد:

نوع خطا: از این نوع پیغام که با مقدار mtError مشخص میشود برای اعلام خطا استفاده کنید. پیغام نوع خطا اغلب با دگمه Ok به کار میرود.

```
MessageDlg('";" expected', mtError, [mbOk], 0);
```

نوع اطلاعات: از این نوع پیغام که با مقدار mtInformation مشخص میشود برای اطلاع رسانی به کاربر استفاده کنید. پیغام نوع اطلاعات اغلب با دگمه Ok به کار میرود.

```
MessageDlg('Record count = '+ N.ToString, mtInformation, [mbOk], 0);
```

نوع هشدار: از این نوع پیغام که با مقدار mtWarning مشخص میشود برای کسب مجوز برای انجام پردازشی که ممکن است خطری به همراه داشته باشد (مثل حذف اطلاعات) استفاده کنید. پیغام نوع هشدار اغلب با مجموعه دگمه‌های Yes، No و Cancel به کار میرود.

```
Ans:= MessageDlg('Delete the record?', mtWarning, mbYesNoCancel, 0);
if Ans= mrYes then // Do Something
```

در حالت عادی مجموعه دگمه‌ها باید در یک جفت کروشه معرفی شوند، ولی برای برخی از ترکیبهای پرکاربرد ثابتهایی مثل زیر تعریف شده‌اند:

```
mbYesNoCancel = [mbYes, mbNo, mbCancel];
mbOKCancel = [mbOK, mbCancel];
```

نوع تایید: از این نوع پیغام که با مقدار mtConfirmation مشخص میشود برای کسب مجوز برای انجام پردازشهای کم خطر (مثل نصب نرم افزار یا خروج از پروژه) استفاده کنید. پیغام نوع هشدار اغلب با مجموعه دگمه‌های Yes، No و Cancel به کار میرود.

```
Ans:= MessageDlg('Install application?', mtConfirmation, mbYesNoCancel, 0);
if Ans= mrYes then // Do Something
```

نوع دلخواه: که با مقدار mtCustom مشخص میشود و یک پنجره بدون نماد که نام پروژه اجرایی در عنوان آن دیده میشود را باز میکند.

دیالوگ دریافت فونت

دیالوگ دریافت فونت برای دریافت مشخصات فونتی به کار میرود. برای استفاده از این دیالوگ ابتدا باید یک نمونه از کامپوننت غیرویژوال FontDialog **را روی فرم درج کنید. این کامپوننت دارای خاصیت اصلی** Font **و متد اصلی** Execute **است. خاصیت** Font **فونت دیالوگ را قبل و پس از فراخوانی آن نشان میدهد و** Execute **هم متدی برای فراخوانی دیالوگ است:**

```
BoolVar:= FontDialog1.Execute;
```

پس از اجرای این فرمان، دیالوگ با فونت ذخیره شده در خاصیت Font **و دگمه‌های تایید و انصراف نمایش داده میشود. با انتخاب دگمه تایید، فونت تغییر یافته در خاصیت** Font **برگشت داده شده و تابع مقدار** True **برمیگرداند و با انتخاب دگمه انصراف مقدار** Font **تغییر نکرده و** False **برگشت داده میشود. بنابراین چنانچه پس از فراخوانی تابع خروجی** True **باشد باید پردازش دلخواه برنامه‌نویس انجام شود. برای نمونه کد زیر فونت مربوط به** Edit1 **را در دیالوگ نشان داده و تغییرات را به** Edit1 **برمیگرداند:**

```
FontDialog1.Font.Assign(Edit1.Font);
if FontDialog1.Execute then
  Edit1.Font.Assign(FontDialog1.Font);
```

دیالوگ دریافت رنگ

دیالوگ دریافت رنگ برای دریافت یک رنگ به کار میرود. برای استفاده از این دیالوگ ابتدا باید یک نمونه از کامپوننت غیرویژوال ColorDialog **را روی فرم درج کنید. روش به کارگیری و فراخوانی این دیالوگ شبیه دیالوگ دریافت فونت است با این تفاوت که به جای خاصیت** Font **از خاصیت** Color **استفاده میشود. برای نمونه کد زیر رنگ پس زمینه** Edit1 **را در دیالوگ نشان داده و تغییرات را به** Edit1 **برمیگرداند.**

```
ColorDialog1.Color:= Edit1.Color;
if ColorDialog1.Execute then
  Edit1.Color:= ColorDialog1.Color;
```

دیالوگهای انتخاب نام فایل

دیالوگهای انتخاب نام فایل برای دریافت نام کامل یک فایل به کار میروند. برای استفاده از این دیالوگها ابتدا باید یکی از کامپوننتهای غیرویژوال زیر را روی فرم درج کنید (تفاوت این دیالوگها اغلب در شکل و نام پنجره است):

OpenDialog , SaveDialog , OpenPictureDialog , SavePictureDialog , OpenTextFileDialog , SaveTextFileDialog

روش به کارگیری و فراخوانی این دیالوگها شبیه دیالوگ دریافت فونت است با این تفاوت که به جای خاصیت Font **از خاصیت** FileName **استفاده میشود. توجه داشته باشید که دیالوگهای دریافت نام فایل فقط نام فایل را برمیگردانند و عملیات ذخیره، بازیابی و ویرایش فایل باید جداگانه کدنویسی و یا انجام شوند. برای نمونه کد زیر پس از دریافت نام فایل محتوای آن را برای ویرایش در پنجره** Memo1 **نمایش میدهد:**

```
if OpenDialog1.Execute then
  Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
```

و کد زیر هم پس از دریافت نام فایل خطوط ویرایش شده در پنجره Memo1 را در فایل مورد نظر ذخیره و یا رونویسی میکند.

```
if SaveDialog1.Execute then
  Memo1.Lines.SaveToFile(SaveDialog1.FileName);
```

میانبرهای مهم

در این بخش کلیدهای میانبر مفید و پرکاربرد را معرفی میکنیم.

افزایش و کاهش فونت: کلید Ctr-Num+ و Ctr-Num- به ترتیب باعث افزایش و کاهش فونت ویرایشگر (برای همه یونیتها) میشود. فونت جدید برای استفاده‌های بعدی به طور خودکار ذخیره میشود.

کامنت کردن: کلید Ctr-/ سطر و یا سطرهای انتخاب شده را کامنت کرده و یا در صورت کامنت بودن از کامنت در میآورد.

انتخاب کامل سطر: در حالت پیش فرض عمل دوبار کلیک کلمه‌ای که مکان نما بر روی آن قرار دارد را انتخاب میکند. ولی میتوانید آن را برای انتخاب یک سطر تنظیم کنید. برای این منظور گزینه «Double click line» را در پنجره Tools→Options یافته و علامت بزنید.

فرمت کردن کد: با کلید Ctr-D میتوانید کدهای یونیت و یا کدنویسی‌های بخش انتخاب شده را برای ایجاد فاصله و تورفتگیهای پیش فرض و حذف سطرهای اضافی فرمت کنید. برای تغییر روش فرمت به بخش Formatter در پنجره Tools→Options رجوع کنید.

افزایش و کاهش تورفتگی: کلید Tab و Shift-Tab به ترتیب باعث افزایش و کاهش تورفتگی در دستورهای انتخاب شده میشود.

نمایش تعریف شناسه: کلید Ctr-Click یا Alt-Up چنانچه بر روی یک شناسه انجام بگیرد تعریف آن شناسه را نمایش میدهد، حتی اگر تعریف آن شناسه در یک یونیت دیگر و یا یونیتهای خود دلفی باشد.

تاگل بین عنوان و متن زیربرنامه: کلید Shift-Ctr-Up ، Shift-Ctr-Down یا Ctr-Click بین عنوان زیربرنامه یا متد (اغلب در بخش واسط) و تعریف کامل آن (در بخش پیاده‌سازی) تاگل میکند.

ایجاد متن و یا عنوان متد: کلید Shift-Ctr-C چنانچه بر روی یک متد بدون متن فشرده شود، متن کامل آن را در بخش پیاده‌سازی ایجاد میکند و چنانچه روی یک متد بدون عنوان فشرده شود، عنوان آن را در بخش تعریف متد اضافه میکند.

مکمل کدنویسی: کلید Ctr-Space لیست شناسه‌هایی را نشان میدهد که با حروف تایپ شده شروع میشوند. برای نمونه برای نوشتن InputBox کافی است حروف ابتدایی آن (مثل Inp) را تایپ کرده و Ctr-Space را فشار دهید. IDE دلفی شناسه‌های با پیشوند Inp را در یک پنجره لیست میکند.

الگوی کدنویسی: کلید Ctr-J لیست الگوهای کدنویسی را نشان میدهد که با حروف تایپ شده شروع میشوند. برای نمونه برای نوشتن پروسجر کافی است حروف ابتدایی آن (مثل pr) را تایپ کرده و Ctr-J را فشار دهید. IDE دلفی الگوهای کدنویسی با پیشوند pr را در یک پنجره لیست میکند. با انتخاب الگوی procedure متن کامل آن ایجاد میشود. برای تعریف یک الگوی کدنویسی جدید باید از پنجره ابزار الگوسازی استفاده کنید. در صورت پنهان بودن این پنجره با گزینه Tool Windows→Templates در منوی View آن را نمایش دهید.

یافتن شناسه: کلید Shift-Ctr-Enter چنانچه بر روی یک شناسه انجام بگیرد تعریف شناسه و همه موارد کاربرد آن را در همه یونیتهای پروژه نمایش میدهد.

تغییر نام شناسه: کلید Shift-Ctr-E تعریف شناسه و همه موارد کاربرد آن را در همه یونیتهای پروژه یافته و نام جدید را جایگزین نام قدیم آن میکند.

ایجاد متد جدید: کلید Shift-Ctr-M کد انتخاب شده از یک زیربرنامه (یا متد) را در یک زیربرنامه (یا متد) جدید وارد کرده و جمله فراخوانی زیربرنامه (یا متد) جدید را جایگزین کد انتخاب شده میکند.

حذف سطر: کلید Ctr-Y سطری که مکان نما بر روی آن قرار دارد را حذف میکند.

فولد کردن: کلید Shift-Ctr-K-C همه کلاسها و رکوردهای یونیت (اغلب بخش واسط) و کلید Shift-Ctr-K-M همه متدهای بخش پیاده‌سازی را فولد میکند. فولد کردن کلاسها یا متدها مرور کردن روی یونیت را برای یافتن یک کلاس یا متد خاص آسان میکند. همچنین حذف و یا انتخاب یک سطر فولد شده همه خطوط آن کلاس یا متد را تحت تاثیر قرار میدهد. از کلید Shift-Ctr-K-A برای باز کردن همه بخشهای فولد شده استفاده کنید.

جابجایی به اندازه یک کلمه: کلید Ctr-Right و Ctr-Left مکان نما را به اندازه یک کلمه به سمت راست یا چپ جابجا میکند.

مرور کردن یونیت: کلید Ctr-Up ، Ctr-Down یا نوار وسط ماوس بدون جابجایی مکان‌نما عمل Scroll را روی یونیت انجام داده و خطوط را در پنجره یونیت بالا و پایین میبرد.