منبع : مرکز اطلاع رساني و پژوهش هاي برنامه نويسي ايران (irAsp.Net)
آموزش:مصطفي جلمباداني "آرش"
ايميل: arash.j13@gmail.com

عنوان: **دلفي چيست...؟**

دلفی چیست؟

در سال ۱۹۹۴ شرکت بورلند تصمیم گرفت تا ابزاری برای ساخت سریع اپلیکشن ارائه کنه
چون در اون زمان برنامه نویسان برای نوشتن برنامه ها یا باید به زبان ها قدیمی که غیر بصر بودن پناه می بردن یا از دو زبان ویژوال VB و VC یکی رو انتخاب می کردند که اولی برای نوشتن برنامه های حرفه ای واقعاً نا مناسب بود و دومی هم بسیار سخت و شاید نوشتن یه برنامه با اون مدت بسیار زیادی طول می کشید در میان بورلند تصمیم گرفت یک ابزارRAD (rapid application development)رو ارائه کنه اولین چیزی که برا ی این منظور نیاز بود یه هسته بود که باید یه زبان سطح بالا ی برنامه نویسی بود و از اونجایی که مهمترین محصول بورلند کامپایلر قدرتمند پاسکال بود بورلند از این زبان به عنوان قلب سیستم استفاده کرد البته نه از نسخه ای که در کامپایلر توربو پاسکال استفاده شده چون قبل از ارائه دلفی نسخه ی جدید از پاسکال
به وجود اومده بود در این نسخه پاسکال به دنیای برنامه نویسی شی گرا قدم گشوده بود و ابجکت پاسکال
متولد شده بود دلفی در هسته ی خودش یه کامپایلر ابجکت پاسکال رو قرارا داد و با اون خودش رو به دنیای برنامه نویسی اضافه کرد البته دلفی برای رسیدن به هدفش که همون طراحی سریع اپلیکشن بود
فقط یه کامپایلر با خودش نیاورد بلکه یه کتابخانه بسیار عظیم به نام VCL (Visual component library)
یه محیط مناسب اشکال زدای عالی هم داره دلفی در همو ن نسخه اولیه توانست بسیاری از برنامه نویسان
VB3رو به طرف خودش جذب کنه
الان نسخه های از دلفی که معمولا پر کاربرد ترن دلفی ۷ که می شه گفت یه شاهکاره و نسخه ی جدید دلفی
Delphi 2005هستند که از لحاظ ساختار زبانی فرق چندانی ندارند اما از لحاظ ظاهری فرق بسیاری دارند
من هر جا لازم باشه به هر دو نسخه اشاره می کنم

برای یاد گیری دلفی شما باید به زبان پاسکال آشنا باشید اما اگه آشنایی ندارید من سعی می کنم طی چند قسمت
مهمترین بخش زبان پاسکال رو بگم
امه در دلفی
دلفی یه زبان ساخت یافته بلاکی هست به ایم معنا که ار بلوک ها خاص تشکیل شده هر بلوک در دلفی با
دو کلمه ی کلید Begin و end تعریف می شه
متغییر ها در دلفی
در دلفی هر متغییر باید از قبل تعریف بشه و نوع اون هم همونجا مشخص بشه کامپایلر دلفی در این مورد بسیار
سخت گیره بسیار در تطابق نوع ها دقت می کنه متغییر ها همیشه قبل از بلوک اصلی یه تابع تعریف می شه
و برای این کار از کلمه ی کلید var استفاده می شه طریقه ی تعریف یه متغییر این جوریه

```
Var
Myvaribe : DataType
```

که در اون myvaribe نام متغیر و datatype نوع اون هست

انواع متغییر در دلفی
1عددی که خودش دو نوع
الف عددی صحیح
آ shortint -127.تا ۱۲۷
ب smalint -32768 .تا ۳۲۷۶۸
پinteger -2147483647.تا2147483647
ت int64 .منهای دو به توان ۶۳ تا دو به توان ۶۳ منهای یک
ث byte 0.تا ۲۵۶
ج word 0 .تا ۶۵۵۳۵
چ cardinal 0.تا ۴۲۹۴۹۶۶۷۲۹۵
ب عددهای عشاری
آ single 4 .بایتی
ب real48 6 .بایتی
پ real 8 .بایتی
ت comp .بایتی
ث currency 8 .بایتی
ج extended 10 .بایتی (عددهای در حدود ۱۰ به توان(4900
.2کاراکتری
الف char کارکتر های قدیمی
ب ansichar .کاراکتر های استانداردansi
پ Widechar .کراکتر ها با یکتای شانزده بیتی (یونی کد(
.3رشته ها
الف Shrotstring .رشته های کوتاه
این رشته ها حد اکثر می توانند ۲۵۵ کاراکتر باشند و بدو روش معرفی می شوند
.1بعد از کلمه یstring طول رشته درون [] ذکر شود
.2ازکلمه ی shortstring برای ایجاد رشته ای به طول ۲۵۵ کاراکتر استفاده شود
توجه کنید حافظه مربوط به این رشته ها در زمان تعریف اختصاص می یابد و قابل تغییر نیست ولی سرعت دستکاری در انها بسیار بالا ست
ب String .که گاهی اوقات AnsiString هم نامیده می شود حافظه ی مربوط به این رشته ها به صورت پویا اختصاص می یابد و سرعت دستکاری در انها کمتر از رشته های کوتاه است
پ wideString.رشته هایی با کاراکتر های گسترش یافته از این رشته معمولا در استفاده از توابع API ویندوز استفاده می شود وحافظه مربوط به انها نیز پویا ست
.3اشاره گرها
متغییر های هستند که محلی در حافظه اشاره می کنند و خود دو نوع دارند
آ.بدون نوع
با کلمهی pointer معرفی می شوند و مقدار فظایی که به ان آشاره می کند نامشخص است و توسط برنامه نویس استفاده کننده مشخص می شود
ب .نوع دار
به نوع خصی از داده اشاره می کندو این گونه تعریف می شود

```
Var
P:^Datatype
```

بعد در باره کاربرد و طریقه ی استفاده از اشاره گر مفصل بحت خواهد شد
.4نوع ها گسترش یافته
شامل
class

Record
مجموعه ها
انواع شمارشی
انواع تعریف شده
Interfaceمی باشد
که درباره همه ی آنها به موقع بحث می کنم
ادامه دارد...

عنوان: **دلفي:::درس ۲**

ابتدا بگم ممکن به به عباراتی بر بخورید که معنی اونها ر ندونید اگه این عبارات در سطح همین با شند همین جا
و گرنه در جای خودشون توضیح می دم پس نگران نباشید که منعی بعضی چیز ها رو نمی دونید

یونیت ها در دلفی
یونیت فایلی است که که کد های دلفی در آن نوشته می شود
در دلفی چهار نوع یونیت وجود دارد
1.یونیت برنامه
2.یونیت کتابخانه
3.یونیت پکیج
4.یونیت کد

1.یونیت برنامه
یونیتی است که با کلمه ی کلیدی Program آغاز می شود و کد ها اصلی برنامه یا به عبارتی قسمت شروع
برنامه در این یونیت خواهد بود هر برنامه فقط یک یونیت از این نوع دارد حاصل برنامه ای که این یونیت یونیت اصلی ان باشد یک فایل اجرایی با پسوند های EXE. یا SCR. خواهد بود

2.یونیت کتابخانه
این یونیت هم مانند یونیت برنامه است با این تفاوت که حاصل ان یک کتابخانه با پسوندها ی CPL. OCX. DLL. خواهد بود و با کلمهی library معرفی می شود

3.یونیت پکیج
این یک فایل پکیج برای دلفی می سازد فایل های پکیج در دلفی کاربرد های زیادی دارند که در قسمت پیشرفته با آنها آشنا خواهید شد این فایل با

4.یونیت کد
این نوع پر کاربرد ترین نوع یونیت در دلفی است و معمولا شما با این یونیت کار دارید این یونیت نمی تواند به تنهایی در برنامه به کار رود بلکه باید با یکی از سه نوع دیگر استفاده شود

ساختار یونیت ها

هر چند شما معمولا با ساختار یونیت ها کاری ندرید اما اگر ساختار انها را بدانید بهتر است

یونیت برنامه

این یونیت به طور خودکار توسط دلفی ساخته می شود و معمولا از دید شما پنهان است برای دیدن ان به منو پروژه رفته و گزینه ی View Source را انتخاب کنید اکنون شما یونیتی را می بیند که یونیتی اصلی برنامه است و برنامه از ان آغاز می شود در قسمت بالایی ان کلمه ی Program و بعد نام یونیت که نام برنامه هم هست را می بینید در خط بعد کلمه ی Uses و سپس لیست uses را می بینید این لیست
معمولا به این حال نیست و لی در یونیت اصلی به این شکل نوشته می شود در این لیست نام یونیت های
کدی را می بینید که یونیت حاضر به آنها ارجاع دارد این لیست می تواند تحت یک کلمه ی uses یا چندین
Usesبیاید
بعد از لیست uses رهنمود کامپایلر {$R *.RES} را می بینید این رهنود به لینکر می گویید که فایل ریسورس پروژه را در این جا ادغام کند فعلا با آن کاری نداریم بعدا در باره فایل های ریسورس توضیح می دم بعد قسمت اصلی یونیت که همان قسمت اجرایی آن است شروع می شود قسمت اصلی یک یونیت
برنامه با کلمهی Begin شروع می شود و به .End ختم می شود تو جه کنید که end با نقطه ی پایانی نشانه پایان یونیت هست و هرچه بعد از ان باشد توسط کامپایلر نادیده گرفته می شود و تحت یه اخطار گزارش
می شود شما می توانید begin و End های بسیاری داشته باشید اما هیچکدام غیر از end پایانی یونیت
نباید نقطه داشته باشد
در میان بلوک اصلی چند خط کد می بیند که باعث اجرای برنامه می شود که بعدها آنها را بررسی می کنیم

یونیت کد
این یونیت با کلمه ی unit و سپس نام یونیت آغاز می شود این یوینت شما چهار بخش اصلی است که دو
بخش آن اختیاری است
بخش interface
این بخش بخش معرفی برای خارج از کلاس است یعنی شما معمولا کلاس ها متغییر ها و توابعی رو در این
بخش معرفی می کنید که در بیرون یونیت باید قابل دسترسی باشد این بخش تا رسید به بخش بعدی که
Implementationاست ادامه دارد

بخش implementation
این بخش با همین کلمه ی کلید آغاز می شود و تا رسید به بخش بعدی یا پایان یونیت ادامه پیدا می کنید
در این بخش توابع و متدهایی از کلاس را که در بخش قبل تعریف کرده بودیم پیاده سازی می کنیم مثلا

```
unit Unit2;interfaceprocedure Test;implementationprocedure Test;begin //here codeend;end.
```

بخش initialization
این بخش در هنگام بار گذاری یونیت در حافظه اجرا می شود

بخش finalization
این بخش در هنگام خارج شدن یونیت از حافظه اجرا می شود و برای آزاد کردن حافظه ها ی تخصیص یافته
می توان از آن استفاده کرد

ادامه دارد.....

عنوان: **آموزش دلفی درس۳**

عملگر ها در دلفی

.1عملگر های ریاضی
این عمل گر ها رو فقط لیست می کنم
) div / * - +تقسیم صحیح)mod (باقی مانده (

.2عملگر های مقایسه ای
<و >و)<>نامساوی)و=و >=و <=

.3عملگر های منطقی
And , or, xor, not,

.4عملگرهای بیتی
با این عملگر ها می توان به بیت های حافظه دست یافت و تقریبا سطح پایین ترین کارها را انجام داد
Or, and, xor, not

.5عملگر جایگزینی
از این عملگر برای مقدار دهی متغییر ها استفاده می شد
:=
توجه کنید که دو نقطه الزامی است و باعث تفاوت عملگر با مساوی می شود

در باره ی این عملگر ها که در پایین می آید بعدا بحث می شود
.6عملگر های آدرس دهی
@,^

.7عملگر های کلاس ها
As, is
8عملگر مجموعه ها
In

توجه کنید که از عملگر مقایسه ای برای بررسی کاراکتر ها و رشته هم می توان استفاده کرد

توابع و پروسیجر ها

دلفی یه زبان ابجکت اریتتد (OOP)است که از قابلیت های قدیمی پاسکال پشتیبانی می کند
در دلفی می توان از روال ها استفاده کرد
برای معرفی یه تابع که مقدار بازگشتی دارد از روش زیر استفاده می شود

```
Function
func_name(argument1 :datatype1;argument2:datatype2;…):Result-data-
type
```

```
Begin
//function code
End;
```

که باید در ان لیست آرگومان ها را با علامت سمی کالن; جدا کرد
برای فراخوانی آن می توان از روش زیر استفاده کرد
Result-variable:=Function-name(parameter1,paramtr2...);
که در ان باید لیست پارامتر ها با کاما از هم جدا کرد و استفاده از متغییر برای نگه داری مقدار برگشتی هم اختیاری است

برای معرفی روالی که مقداری را برنمی گرداند باید از کلمه ی procedure استفاده کرد و شکل کار دقیقا به صورت بالا است

ادامه دارد.....

عنوان: **آموزش دلفی درس۴**

در تمام این مقاله منظور ار تابع function ها و procedure ها هستند

پارامتر توابعج
در دلفی دو جور می شه پارامتر رو به توابع فرستاد یک با مقدار و دیگری با مرجع
وقتی شما با مقدار یک پارامتر رو ارسال می کنید دیگر هر تغییر در پارامتر رسیده به تابع انجام شود تاثیری در مقدار متغییر اصلی ندارد ولی وقتی با مرجع پارامتر رو ارسال کنید هر تغییر در متغییر در داده بشه در متغییر اصلی هم اعمال خواهد شد دلفی به طور پیش فرض متغییر ها رو با مقدار ارسال می کنه ولی اگر بخواهید صراحتا اعلام کنید که باید با مقدار ارسال بشه می تونید قبل از نام متغییر از کلمه ی const استفاده کنید مثلا

```
Procedure Test(const a:integer);
Begin
//code
End;
```

و اگر بخواهید از فرستادن با مقدار استفاده کنید می توانید از کلمهی var قبل از نام متغییر استفاده کنید

```
Procedure Test(var a:integer);
Begin
//code
End;
```

البته راه دیگری هم هست که بدون استفاده از فرستادن با مرجع در متغییر دستکاری کرد که استفاده از

اشاره گر ها است که بعدا در بحث اشاره گر ها توضیح داده می شود

Method Overloading

ممکن است شما چند تابع داشته باشید که کار یکسانی را برای چند نوع متغییر انجام می دهد یا یکی پارامتر نمی خواهد و دیگری می خواهد مثلا ضرب را برای اعداد صحیح و حقیقی پیاده سازی کنید حالا یا شما باید از چند نام جداگانه استفاده کنید یا از method overloading استفاده کنید برای method overloading تمام توابعی رو که می خواهید معرفی کنید سپس بعد از تعریف اونها از کلمهی overload استفاده کنید
مثلا

```
Function multiply(a,b:integer):integer;overload;
Function multiply (a,b:real):real;overload;
……
Function multiply(a,b:integer):integer;
Begin
  Result:=a*b
End;

Function  Function multiply(a,b: real):real;
Begin
  Result:=a*b
End;
```

پارامتر پیش فرض

ممکنه شما تابعی داشته باشید که معمولا یه پارامتر ان مقدار ثابتی است و کمتر تغییر می کند شما می توانید برای این پارامتر مقدار پیش فرض در نظر بگیرید تا در مواقعی که لازم است پارامتر را به تابع نفرستید و خود دلفی آن را به تابع ارسال کند
مثلا

```
Procedure Test(a:integer;b:Boolean=true);
```

در این گونه موارد اگه شما به آرگومان b مقداری ندهید و تابع را این گونه فراخوانی کنید

```
TEST(1);
```

دلفی به صور خودکار مقدار true را برای b در نظر می گیرد
توجه کنید که اگر تابعی دارای پارامتر اختیاری(پارامتر با مقدار پیش فرض) باشد باید این آرگومان را در تعریف تابع بعد از پارامتر های الزامی بیاورید و اگر چندین پارامتر اختیاری دارید مثلا
Function test(a: integer=2;b:byte=13;f:integer=9):integer;
اگر بخواهید مثلا به f مقدار دهی کنید باید به تمام پارامتر های قبل از آن هم مقدار بدهید و دگیر دلفی به
پارامتر ها اختیاری قبل از آن مقدار پیش فرض را اختصاص نمی دهد

بحث توابع تموم نشده اما فعلا چون نمیخوام زیاد وارد بحث هایی بشم که شما رو گیج کنه بقیه بحث توابع رو به بعد از یه سری مطالب دیگه موکول می کنم
موفق باشید

عنوان: **درس ۵**

توضیحات
در دلفی توضیحات با سه علامت مشخص می شه
1.هر عبارتی که بین {} قرار داشته باشه مثلا{it is comment}
2.هر عبارتی که بین (**) قرار داشته باشه مثلا(* it is comment*)
3.هر خط یا قسمتی از خط که بعد از // قرار بگیرد مثلا //it is comment

تبدیل نوع
همون طور که قبلا هم گفتم کامپایلر دلفی در یکسان بودن نوع متغییر ها یا ثابت هایی که باهم مقایسه می سن
یا مقدار دهی می شن بسیار سخت گیره مثلا شما به راحتی در C یه کاراکتر رو در متغییر از نوع Byte
می ریزید اما در دلفی اگر اینکار رو بکنید با خطای کامپایل مواجه می شید برای جلو گیری از این خطا ها باید از تبدیل نوع استفاده کرد که به دو روش صورت می گیره
1.خیلی ساده شما نام متغییر رو درون یه پرانتز می نویسید و نوعی که قرار به اون تبدیل بشه رو پشت پرانتز قرار می دی
مثلا

```
Var
  B:byte;
  C:char;
Begin
  B:=65;
 C:=char(b);    //now c='A'
End;
```

از این روش بیشتر برای تبدیل نوع کاراکتر به اعداد و برعکس و تبدیل اشاره گر ها به نوع خاصی از اشاره گر استفاده می شه یا تبدیل مقدار ریفرنس یه اشاره گر بدون نوع به نوعی خاص(درباره اشاره گرها بعد توضیح داده می شود)
و همچنین تبدیل رشته های معمولا به رشته های مختوم به تهی

توجه کنید که در بعضی موارد نیاز به این دستورات نیست و خود دلفی تبدیل نوع را انجام می دهد
مثلا
تبدیل انواع عدد های صحیح یا اعشاری به هم(صحیح به صحیح و اعشاری به اعشاری) تبدیل رشته
های معمولی به هم و هم چنین تبدیل عدد صحیح به اعشاری و چند نوع دیگر

تبدیل به کمک توابع بعضی از نوع ها را نمی توان به کمک تبدیل نوع صریح تبدیل کرد مثلا یه رشته به عدد یا برعکس برای این گونه تبدیل ها توابعی در کتابخانه ی زمان اجرا RTL موجود می باشد که

مهمترین توابع به شرح زیر است
IntToStrتبدیل عدد صحیح به رشته
StrToIntتبدیل رشته به عدد صحیح
StrToFloatرشته به عدد حقیقی
FloatToStrتبدیل عدد حقیقی به رشته
Intتبدیل عدد حقیقی به صحیح
StrToDateتبدیل رشته به تاریخ
DateToStrتبدیل تاریخ به رشته
...

همچنین یک عملگر هم برای تبدیل اشیا به هم وجود داره که در جای خودش بحث می شه
)عملگر(as

توابع کار با رشته ها

مانند هر زبانی دلفی هم توابعی برای کار با رشته ها داره
Lengthبدست آوردن طول رشته
Copyقسمتی از یه رشته رو بر کی گردونه
پارمتر اول رشته رو مشخص می کنه پارمتر دوم محل شروع و پارمتر شوم طول رشته برگشتی که در صورتی که مقداری به اون ندهید باقی رشته رو برمی گردونه
مثال

```
Var
  S1,s2:string;
Begin
  S1:='learning Delphi on IrAsp.Net';
   S2:=copy( s1,9,6);   //now s2 = 'Delphi'
End;
```

Posاین تابع محل شروع یه رشته رو درون رشته دیگه پیدا می کنه
پارمتر اول رشته ی اصلی ارمتر دوم رشته ای که باید به دنبال اون گشت
مثلا

```
Var
  S:string;
  I:integer
Begin
  S:='learning Delphi on IrAsp.Net';
   i:=post( s,'Delphi');   //now i = 9
End;
```

Uppercaseتبدیل رشته به حروف بزرگ
Lowercaseتبدیل رشته به حروف کوچک

Strpcopyکپی کردن یه رشته مختوم به تهی در یکی دیگر مثلا

```
Var
  S1,string;
  S2:pchar
Begin
  S1:='IrAsp.Net';
  Strpcopy(s2,s1);//now s2 = 'IrAsp.Net'
End;
```

StringOfCharیه رشته با طول مشخص که از کاراکتر خاصی پر شده رو برمی گردونه

توابع کار با رشته بسیار زیاد که به طوری که در این مقاله نمی گنجه اما شما می تونید به مراجعه به هلپ
دلفی اونها رو پیدا کنید
موفق باشید

عنوان: **ساختار های کنترل درس۶**

ساختار های کنترلی
در دلفی ساختار های کنرلی دقیا از اسکال به ارث برده شده و تغییر نکرده پس اگه با پاسکال آشنا هستی ان مقاله برای شما چیز چندان جدید نداره
ساختار شرطی
ساختار سرطی بسیار ساده است پس فقط یه مثال می زنم

```
if b>10 then
s:=1
else if b>20 then
s:=2
else
begin
s:=3;
end;
```

توجه کنید که دستورات می تواند ساده یا یه بلوک باشد همچنین نباید قبل از else سمی کالن به کار برد

ساختار انتخاب

```
case varible of
  label1: action1;
  lable2: action2;
  ...
  lable n: action n
  else
    default action ;
end;
```

توجه کنید که متغییر که مورئد انتخاب قرار می گیره باید یه مغییر عددی کارکتری یا شمارشی باشه

هم lable می تواند شامل چنیدی مقدار باشد که با یه , از هم جدا میشن یا یه محدوده باشن که با نوشتن ابتدا و گذاشت دو نقطه و نوشت انتها اون رو مشخص کرد
و دستورات هم می تونن مرکب یا ساده باشند
مثال

```
procedure test(i:integer);
var
  s:integer;
begin
   case i of
     1:  s:=1;
     2,3: s:=2;
     4..10 : s:=3;
     11,12,13:
     begin
        s:=4;
     end;
    else
       s:=5;
   end;
// other action
end;
```

حلقه ها

1.حلقه های بدون شمارنده
حلقه یwhile
این حلقه تا زمانی که شرط جلوی while برقرار باشد دستور بعد از do رو اجرا می کنه
این دستو می تواند مرکب باشد
مثال

```
i:=0;
While i<10 do
begin
   sum:=sum+i;
   inc(i);  // i:=i+1;
end;
```

حلقه یrepeat
این حلقه بسیار شبیه حلقه ی while است با این تفاوت که حلقه تا زمانی ادامه پیدا می کنه که شرط برقرار شود و در ضمن شرط هم در انتهای حلقه کنترل می می شود
و برای بیش از یک دستو رهم نیازی به نوشتن انها در یک بلوک نیست چون کلیه کدهای بین repeat و untilیه بلوک به حساب می آید
مثال
[left]

```
i:=0;
repeat
  sum:=sum+i;
  inc(i);//i:=i+1;
until i=10;
```

حلقه های با شمارنده
حلقه یfor to do
ساختار
[left]
for LooopVar:=StartVlaue to EndValue do

که أر آن دستورات یعد از do تا زمانی اجرا می شود که مقدرا LoopVar به enaValue برسد و در هر بار آجرا یک واحد به آن اضافه می شود
دستور بعد از do می تواند مرکب باشد
البته می توان به جای to از downto استفاده کرد در اینصورت هر بار اجرای حلقه یک واحد از متغییر حلقه کم می شود متغییر حلقه باید عدد صحیح که کوچکتر از ۳۲ بیت باشد یا متغییر کارکتری باشد
مثال

[left]

```
for i:=0 to 10 do
     sum:=sum+i ;
```

حلقه ی for in do این حلقه فقط در delphi 2005 پیشتیبانی می شود
این حلقه برای برررسی عناصر درون آرایه استفاده می شود و در مبحث آرایه ها بحث می شود

موفق باشید

عنوان: **انواع گسترش یافته درس۷**

آنواع گسترش بافته
دلفی به جز دیتا تایپ ها معمولی نوع ها گسترش یافته ای که توسط کاربر مشخص می شود را نیز شامل می شود
در این مقاله به بررسی این انواع داده می پردازم
1. نوع محدوده شما می توانید دیتا تایپی را تعریف کنید که محدوده خاصی از عادد صحیح یا کارکتری را شمل شود این بازه باید کران دار باشد
مثلا

```
Type
  TExample = 1..100;
```

حالا شما می توانید متغییر هایی از نوع TExample تعریف کنید که شامل اعداد یک تا صد می شود

نوع شمارشی
این نوع فقط مقداری خاصی که با ثابتی هایی مشخص می شود را می گیرد
مثلا

```
TExample = (vaue1,value2,value3);
```

در این حالت اولین مقدار برابر صفر و به همین ترتیب مقدار آنها افزایش می یابد اما تغییر نوع TExampleفقط می تواند یکی از این ثابت ها یا مقدار برابر آنها را بگیرد

مجموعه ها

بزارید اول یه مثال از مجموعه بگم بدون چک همه ی شما تا حالا تایپ کردید و می دونید که هر فونت خواصی مثل پرنگ بود ایتالیک بودن یا زیرخط دار بودن رو می گیره
و می تونه تعدادی یا همه یا هیچکدوم رو داشته باشه در اصل یه مجموعه می تونه چند تا عضو داشته باشه
در دلفی ما مجموعه ها رو از روی نوع ها شمارشی به کمک کلمهی set تعریف می کنییم
مثلا

```
Type
TExample = (bold,Italic,Underline);
Set TFontStyle =TExample
```

در این حالت یه مجوعه تعریف کردیدمی تونید از اون به این شکل استفاده کنید

```
var
  font: TFontStyle;
begin
   font:=[bold];
  font:=font+[intalic];
```

```
  font:=font-[bold];
end;
```

همچنین می توانیداز عملهای اجتمال+ وتفاضل - و اشتراک * هم استفاده کنید
اگر مبحث عملگر های رو به خاطر بیارید عملگری بود که گفتم مخصوص مجموعه هاست و در بخش خودش توضیح می دم
عملگر in با این علگر کنترل می کنیم که آیا عضو خاصی متعلق به مجموعه هست یا نه
به این صورت

```
var
  font: TFontStyle;
begin
   font:=[bold,italic];
   if italic IN font then
        //here code for ture
  else
      //here code for false
end;
```

رکوردها
این داده ها خود شمال چندید متغییر هستند به یه نمونه توجه کنید

```
type
TExample = record
      fristname : string;
      lastname : string;
end;
```

هان طور که ی بینید این رکورد شمال دو فیلد است تو جه کنید که لازم نیست نوع فیلد ها یکسان باشد و از هر نوع حتی یه رکورد دیگر و یا همان رکورد نیز می تواند باشد
برای دسترسی به فیلد ها رکورد این گونه عمل می کنیم ابتدا نام متغیر رکود سپس یه نقطه و بعد نام فیلد

توجه کنید که رکورد ها در فراخوانی توابع API ویندوز بسیار پر کاربرد می باشند

نوع تابع
شما میوتوانید متغییر هایی ار نوع تابع تعریف کنید این متغییر ها را این گونه تعریف می کنند

```
type
  TFunction=function(ageument):resultType ;
```

همچنین شما می توانید توابع را مجبور کنید که عضو کلاس باشند برای این کار در آنتهای تعیر از کلمهی of object استفاده کینند
مثلا

```
type
```

```
  TFunction=function(ageument):resultType of Object;
```

این متغییر ها در VCL کار برد دارند وبرای ایجاد کنترل کننده های رویداد استفاده می شوند

آرایه ها
من فرض می کنم همه مفهوم آرایه رو می دونند
در دلفی یه آرایه اینگونه تعریف می شود
[code]

```
Type
  a = array[low..higth] of data type;
```

البته می توان این تعریف را در هنگام تعریف متغیی آورد مثلا
[code]

```
var
  a:array[low..higth] of data type
```

این آرایه ها ارایه های استاتیک هستند که حد بالای آنها در هنگام تعریف مشخص می شود آرایه های پویا هم وجود دارد که در مقاله ی بعد در باره آنها صحبت می کنم
ادامه دارد...
موفق باشد
آرش

عنوان: **آرایه های پویا درس۸**

آرایه های پویا
این آرایه ها بر خلاف آرایه ها ی استاتیک می تونن اندازشون در زمان اجرا تعیین بشه و به صورت دینامیکی تغییر اندازه بدهند
این آرایه هم مانند آرایه های معمولی استفاده می شن با این تفاوت که قسمت مربوط به طول آرایه رو نمی نویسید و به این صورا معرفی می شه

```
type
TArary = array of data-type;
یا
var
 a:array of data-type;
```

در این ارایه برای استفاده باید ابتدا صول ارایه رو مشخص کنیم برای اینکار از تابع setlengthاستفاده می شه که طول آرایه رو تنظیم می کنه و می تونید هر زمان نیاز به تغییر طول ارایه داشتید اون تغییر بدید
البته اگر ارایه رو کوچیک کنید داده ها یی که در خانه های اخر ارایه بوده اند از بین می رود

توابع کار با آرایه ها

setlengthاین تابع همو طور که گفتم طول آرایه رو تنظیم می کندفقط در آرایه های پویا(

)copyآرایه ها ی پویا(
با این تابع می تونید چند ایندکس از یه ارایه رو کپی کنید و آرایه جدید بسازید رون یه ارایه دیگه بریزید
پامتر ها
1.پارامتر اول ارایه منبع
2.پارامتر دوم ایندکس شروع
3.تعدا ایندکس هایی که باید کپی بشه

این تابع یه ارایه پویا بز می گردونه

آرایه های چند بعدی
1.استاتیک
برای تعریف این ارایه ها دو راه داریم
یک

```
var
  a:array [0..10] of array [0..20] of integer;
یا
var
 a:array [0..10 , 0..20] of integer;
```

برای دسترسی به عناصر این ارایه این گونه عمل می کنیم

```
a[1][2]:=1;
یا
a[1,2]:=3;
```

تو جه کنید که اگر فقط یه اندیس بگذلرید مثل a[1] مقدار موجود خود یک ارایه هست که دارای ۲۰ ایندکس است

2پویا
برای ارایه های پویا تعداد بعد ها ی آرایه را در زمان تعریف تعیین کنیم
البته می تونان دیتا تایپ را برابر pointeger گذاشت تا به اشاره گر اشاره کند انگه می تواند تعدا بعد ها را دزمان اجرا تعیین کرد
که بحث در باره ی آن در این مقاله نمی گنجد

```
var
  a:array of array:integer;
  i:integer;
begin
  setlengh(a,10);
  for i:=0 to 9 do
     setlength(a[i],20);

end;
```

حلقه ی کار با ارایه ها(فقط در دلفی ۲۰۰۵)
از این حلقه می توان به جای حلقه for to do برای دسترسی به ایندکس های ارایه استفاده کرد
با یه مثال طریقه کار رو نشون می دم

```
var
  a:array[0..10] of char;
  c:char;
begin
  strpcopy(a,'irasp.net');
  for  c in a do
    c:='A';
// now a='AAAAAAAAAAA'
end;
```

این حلقه دسترسی به عناصر ارایه را بسیار ساده کرده است

عنوان: **آشاره گر ها درس۹**

آشاره گر ها
این مطلب یکی از مهمترین بحث ها در دلفی است اشاره گر ها در دلف بسیار پر کاربرد هستند هر چند این نوع در پاسکال هم وجود داشته ولی هیچ گاه کاربردش به اندازه دلفی نبوده
حتی می تونم ادعا کنم که کاربرد اشاره گرها در دلفی بسیار بیشتر از c++ بوده و تقریبا همون قدرتی رو که برنامه نویسان C++ در کار با حافظه دارند رو برنامه نویسان دلفی هم دارن
ولی به شکلی راحت تر

اشاره گر چیست
حافظه اصلی کامپوتر رو می شه مثل یه شهر در نظر گرفت که هرخونه یه آرس داره ما می تونیم این آدرس ها رو در متغییر هایی گه اری کینم که با اشاره گر موسوم هستند
در واقع اشاره گرها آدس دیگر مغییر ها رو نگه داری می کنند و خودشون مقدار با ارزشی ندارن

اشاره گر ها انواع گونا گون دارن ولی ما فعلا با اشاره گری کار می کنیم که به اشاره بدون نوع معروف ایین اشاره گر به کمک کلمه ی pointer تعریف می شه

```
var
  p: pointer;
```

خب حالا چه جوری ادرس یه متغییر رو بدست بیاریم
اگه از قسمت عملگر ها به خاطر داشته بلشید دو عملگر مخصوص اشاره گر ها موجود بود عملگر ادرس و عملگر ریفرنس
که عکس هم عمل می کنند برای بدست آوردن آدرس یه متغییر از عملگر ادرس استفاده می کنیم به این صورت

```
var
```

```
  p: pointer;
  i:integer;
begin
  i:=10;
  p:=@i;
end;
```

ما انواع دیگری نیز از اساره گر ها داریم که به اشاره گرها نوع دار معروف هستند و هر کدام فقط می توانند آدریس نوع خاصی را نگه داری کنند
برای تعریف اونها این جوری عمل می کنیم

```
type
  PType = ^ TType
```

مثال

```
type
  PInteger =^integer;
```

البته توجه کنید که برای بیشتر انواع اصلی اشاره گر ها در یونیت سیستم دلفی تعریف شده و نیازی به تعریف مجدد اونها ندارید می تونید با اضافه کردن یه p به اول نام اون نوع از اشاره گر به اون نوع استفاده کنید
مثلا انواع Pinteger, PCardinal, PDouble معتبر هستند

تعریف اشاره گر به انواع دیگه خصوصا رکورد ها و ارایه ها بسیار کاربرد داره مثلا رشته های معمولی دلفی اشاره گری به ارایه ای از کارکتر ها هستند
ادامه دارد...
آرش

عنوان: کار با اشاره گر ها درس۱۰

واکشی اشاره گر ها
فرض کنید که اشاره گر p به یک عدد صحیح اشاره می کنه ما می تونیم به راحتی عددی که p بهش اشاره ی کنه رو بخونیم یا تغییر بدیم برای اینکار از عملگر ریفرنس استفاده می کنیم
مثلا این یه برنامه ی کنسول هست

```
var
  p:Pinteger;
  i:integer;
begin
  i:=10;
  p:=@i;
  P:=13; //now i=13
  writeln(i);
  readln;
end;
```

از این روش می توان برای ارسال پارمتر به توابع نیز استفاده کرد یعنی شما می توانید اشاره گری به تابع ارسال کنید تا تابع بتواند در آن مقدار تغییر ایجاد نماید

توجه
اشاره گری که به هیچ داده ای اشاره نکند با مقدار ویژه nil شخص می شود و شما می تواتندی با تنظیم کردن مقدار اشاره گر به این مقدار آن را وادار کنید تا به جایی اشاره نکند

تخصیص حافظه شما می توانید از تخصیص حافظه پیا استفاده کنید البته ای روش بیشتر در ارایه بزرگ یا رکود های با فیلد های زیاد کاربرد دارد ولی گاهی اوغات برای رکورد های معمولی هم کاربرد
پیدا می کردند هرچند که فقط کارایی آن به رکود ها و آرایه ها محدود نمی شود به اتمم انواع تعمیم می یابد مخصوصا اشیا که فقط باید به روش پیا تخصیص حافظه یابند

شما وقتی یه متغییر تعریف می کنید دلفی به آن مقداری حافظه اختصاص می دهد که غیر قابل تغییر ولی در روش پویا شما از شاره گرها استفاده می کنید و هر گاه نیاز داشتید حافظه را تخصیص می دهید
هر گاه نیاز داشتید آن را آزاد می کنید یا مقدار آن را تغییر می دهید

برای تخصیص حافظه از توابع getmem allocmem getmemory و newاستفاده می شود که معمولا شما با allocmem استفاده میی کنید
ان تواع معمولا مقدار حافظه مورد نیاز رو به بایت گرفته و یه اشاره گر به حتفظه تخصیص یافته بر م گردونند
بری تغییر مقدار حافظه می توان از ReallocMem و freemem استفاده کرد که اولی مقداری اشاره گر به حافظه و مقداری که باید به اشاره گر اختصاص یابد رو دریافت
و دومی اشاره گری به حافظه و مقداری که باید آزاد گردد رو دریافت می کنه توجه کنید که freememنمی تونه مقدار بیش ازمقدار کنونی رو برای اشاره گر در نظر بگیره

آزاد کردن حافه به کمک توابع freemem ReallocMem Dispose آنجام می شه که برای ازاد کردن حافظه با reallocmem باید یه اشاره گر به حافظه و مقدار صفر و برای استفاده از دو تای دیگه فقط یه اشاره گر به حافظه ارسال کنید
مثلا باید با هم یه رکورد پیا بسازیم
این کد رو من نوشتم

```
Type
  PInfo=^TInfo;
  TInfo=record
    name:string;
    old: integer;
  end;

var
  P:PInfo;
  T:TInfo;
begin
  p:=AllocMem(sizeof(TInfo));
  p.name:='amin';
  p.old:=20;
  t:=p^;
  freemem(p);
  writeln(t.name);
  Writeln(t.old);
  readln;
end.
```

در قسمت type ابتدا رکور و اشاره گر به رکورد رو تعریف می کنیم تو جه کنید که استثنائا تعریف اشاره گر به نوع قبل از تعریف نوع آورده می شود
سپس در قسمت var دو متغییر کی از نوع رکور و یکی از نوع اشاره گر به رکورد تعریف می کنیم
و در قسمت کد اصلی ابتدا به اشاره گر فضا تخصیص می دهیم تو جه کنید که تابع sizeof مقدار حافظه مورد نیاز برای ساختار یا متغیر رو بر می گردونه
در خط بعد به فیلد name و old مقدار دهی می کنیم تو جه کنید استثناوا می توان برای دسترسی به فیلد های یه رکورد پویا از متغییر اشاره گر همانند متغییر رکورد استفاده کرد
در خط بعد یعنی این عبارت t:=p^ مقدار رون اشاره گرا واکشی می کنیم تو جه کند اگر می نوشتیم t:=p غلط بود و کامپایلر خطا می گرفت چون t از نوع رکورد وp
از نوع اشاره گر است ولی وقتی از عملگر ^ استفاده می کنیم و این عملگر مقداری رو که بر می گردونه از نوع رکورد خواهد بود
در خط بعد حافظه رو ازاد م کنیم من اینجا تاکید می کنم که باید هر حافظه ای رو که تخصیص می هید خودتون آزاد کنید چون دلف فقط حافظه هایی رو که خودش تخصیص داده آزاد می کنه
.اگر شما این کار رو انجام ندید اون قسمت از حافظه بلا استفاد خواهد موند
در خطوط آخر هم مقادیر t چا پ می شوند و می بیند که همون مقادیری که به p دادیم هستند
ادامه دارد

عنوان: **تخصیص حافظه درس۱۱**

تخصیص حافظه
در دلفی تخصیص حافظه به دو گونه است یا از stack استفاده می شه ه این کار رو خود دلفی و در زمان طراحی انجام می ده یعنی مقدار stack غیر قابل تغییر
و مقدار stack بسیار کوچیکه و غیر قابل تغییر ولی بقیه حافظه کامپوتر در اصطلاح heap می گن این حافظه مقدار فظای اشغال نشده توسط سایر برنامه ها است
که هنوز استفاده نشده در ویندوز حداقل ۱۰۰ مگابایت می باشد اگر شما برنامه نویس باشید می دانید که این مقدار حافظه بسیار عظیمی است
خب حالا اگه شما از این حافظه با توابعی که در قبل گفتم استفاده کنید و حافظه تخصیص بدید خودتون وظیفه دارید که این حافظه رو ازاد کنید و دلفی براتون این کار
رو نمی کنه این ضعف دلفی نیست که حافظه رو ازاد نمی کنه بلکه این یه قدرته چون اگه قرار بود مدریت حافظه در دست دلفی قرار بگیره اون وقت دلفی هم باید
در روشی مانند جاوا شاره گر ها رو حذف می کرد و این یعنی یه برنامه نویس بدون اسلحه و بی قدرت چون شما با کمک اشاره گرها می تونید بر سیستم حکمرانی کنید
و در هر جای حافظه که خواستید چیزی بنویسید یا چیز بخونید و هر کاری که خواستید انجام بدید ولی مدریت حافظه هم در دست شماست البته دلفی باز هم شما رو تنها
نگذاشته و با کمک روش های شی گرا شما رو در مدریت حافظهئ کمک می کنه ولی این دلخواه که از اون سیستم استفاده کنید یا از سیستم مدریت حافظه خودتون
بعدها در معرفی کلاس vcl با این کلاس که مدریت حافظه رو انجام می ده اشنا خواهیم شد ولی فعلا شما باید بدونید که اگر شما حافظه ای رو تخصیص دادید
باید خودتون و فقط خودتون اون رو ازاد کنید این مطلب بسیار مهمی است یکی از اصول یهبرنامه خوب اینکه حتی یه بایت از حافظه ای که به برنامه تخصیص یافته رو
نیز بلا استفاده نزاره و اون ازاد کنه

خب حالا که فهمید تخصیص حافظه ی پویا چیه بهتره چند تا از نقاط ضعفش رو هم بدونید اول این تخصیص حافظه کمی کند تر صورت می گیره چون نیاز به سربار عملیات داره
که البته این کندی در حالت عادی مشاهده نمی شه و فقط در حلق های طولانی دیده می شه

خب فکر این فصل هم تموم شده ولی اگر مشکلی داشتید من در تالار اماده ی پاسخ گویی به شما هستم
موفق باشید
آرش