زبان پاسکال :

پاسکال یک زبان ساختار یافته و منظم است ، یعنی برای کنترل جریان دستورالعمل های برنامه فقط از سه ساختار زیر پیروی می کند :

- ترتیب : یعنی دستورات یکی پس از دیگری استفاده می شود .
- انتخاب : یعنی در صورت برقرار یک شرط ، عملی انجام می شود و گرنه عمل دیگری یا عملی انجام نمی شود
- تکرار : یعنی اجرای مداوم یک یا چند دستورالعمل تحت شرایط خاص .

ساختار زبان پاسکال به صورت کلی :

۱. عنوان برنامه
۲. بخش تعاریف و اعلانات
۳. بلوک اصلی

```
Program
Uses
Label
Const
Type
Var
Procedure
Function

Begin
.....
.....
End.
```

عناصر و اجزای اصلی زبان پاسکال :

مجموعه و علائم و نشانه های مجاز - ثابتها - متغیرها - شناسه ها - کلمات از قبل رزرو شده پاسکال- دستورات - توابع و پردازه ها

- **مجموعه و علائم و نشانه های مجاز** : مانند : اعداد ۰ تا ۹ - حروف کوچک یا بزرگ A تا Z و * / () و ...
- **متغیر** : مکانی با نام معین از حافظه برای نگهداری داده ها که در طول برنامه قابل تغییرند.

```
Var نام متغیر : نوع متغیر ;
Var A : Integer;
```

- **ثابتها** : مکانی با نام معین از حافظه برای نگهداری داده ها که در طول برنامه قابل تغییر نیستند .

نکته مهم : ثابتها به دو دسته بدون نوع و نوع دار تقسیم می شوند . **تفاوت** این دو در این است که مقدار ثوابت نوع دار در طول برنامه قابل تغییر است.

```
Const نام ثابت = مقدار;          یا          Const نام ثابت : نوع ثابت = مقدار ثابت;
Const P=۳,۱۴;                              Const N:Byte=۱۰۰;
```

- **شناسه** : اسمی است که برنامه نویس به دلخواه خود برای نام گذاری متغیرها ، ثابتها ، عنوان برنامه ها و ... انتخاب می کند .

نکته ۱ : حداکثر طول یک شناسه در تورپو پاسکال ۶۳ کاراکتر می باشد اما در پاسکال(قدیم) ۱۴ کاراکتر بود.

نکته ۲ :شناسه می تواند فقط شامل حروف الفبایی و ارقام ۰ تا ۹ و علامت زیر خط (_) باشد . بکاربردن علائم ویژه مجاز نیست

نکته ۳ : اولین حرف نباید از نوع عددی باشد .

نکته ۴ :استفاده از فاصله غیر مجاز است .

نکته ۵ : شناسه می تواند شامل نام تمامی توابع و دستورات در پاسکال غیر از String باشد . مانند :Reed,Sqr,write,crt
فقط نباید جزو اسامی از قبل رزرو شده پاسکال باشد .

نکته ۶: پاسکال در استفاده نام شناسه از حروف کوچک یا بزرگ تفاوتی قائل نمی شود .

عملگرها و انواع آنها :

عملگر : نماد هایی هستند برای انجام عملیات خاص – عملگرها برای انجام اعمال خاص روی عملوندها بکار می روند.

انواع عملگرها	
محاسباتی	+ - * / Div Mod
رابطه ای	= <> < <= > >= In
منطقی	And Or Xor Not
بیتی	Shr Shl And Or Xor Not

تقدم عملگرها	
۱	@ Not
۲	* / div mod and shl shr
۳	+ - or xor
۴	= <> < > <= >= in

نکته : عملگر @ آدرس یک متغیر

و عملگر ^ محتویات یک محل از حافظه را مشخص می کنند .

به عبارتی عملگر ^ برای دسترسی غیر مستقیم به حافظه بکار می رود .

مثال : محتویات A برابر ۱۲ A:= ۱۲ ;

آدرس متغیر A در B قرار می گیرد. B:= @A;

محتویات جایی که آدرس آن در B است . C:=B^;

انواع داده ها در پاسکال :

عددی صحیح		
نوع	محدوده	میزان حافظه مصرفی
Byte	۰ تا ۲۵۵	۱بایت ۸ بیت
Shortint	-۱۲۸ تا ۱۲۷	۱بایت ۸ بیت
Integer	-۳۲۷۶۸ تا ۳۲۷۶۷	۲بایت ۱۶ بیت
Word	۰ تا ۶۵۵۳۵	۲بایت ۱۶ بیت
Longint	-۲۱۴۷۴۸۳۶۴۸ تا ۲۱۴۷۴۸۳۶۴۸	۴بایت ۶۴ بیت

عددی اعشاری		
نوع	تعداد ارقام معنی دار	میزان حافظه مصرفی
Single	۷-۸	۴ بایت
Real	۱۱-۱۲	۶ بایت
Double	۱۵-۱۶	۸ بایت
Extended	۱۹-۲۰	۱۰ بایت
Comp	۱۹-۲۰	۸ بایت

نکته : اعداد اعشاری را می توان به دو صورت ممیز شناور و نماد علمی نشان داد . در نمایش اعداد اعشاری به صورت ممیز شناور ، حداقل یک رقم قبل و بعد از ممیز باید وجود داشته باشد . (۰,۲۵ درست) - (.۲۵ ۲۵. نادرست).

در نمایش اعدا اعشاری به صورت نماد علمی ، عدد از دو جزء مانتیس و نما تشکیل شده است . مانتیس باید حداقل یک رقم + یا – باشد . بعد از مانتیس حرف E و سپس نما قرار می گیرد . نما باید یک عدد صحیح + یا – باشد . (۲۵E-۳)

غیر عددی :

نوع داده		توضیح	میزان حافظه
منطقی	Boolean	فقط می تواند دو ارزش True و False را بپذیرد. **نکته** : این نوع داده را نمی توان از ورودی خواند.	۱ بایت
کاراکتری	Char	تک کاراکتری یا تک حرفی – می تواند در برگیرنده یکی از علائم مندرج در جدول اسکی باشد.	۱ بایت
رشته ای	String	مجموعه ای از کاراکترها یا چند حرفی- **نکته** : در صورتیکه طول رشته را تعیین نکرده باشید ، طول این نوع داده ۲۵۵ کاراکتر میباشد باشد.درصورتیکه طول رشته را معین نمایید: (میزان حافظه مصرفی = طول رشته + ۱). Var a: string; بدون محدودیت(۲۵۶ بایت) Var a:string[۱۰]; با محدودیت (۱۱ بایت)	

نوع داده	توضیح
اسکالر	به نوعی از داده ها می گویند که مقادیر تشکیل دهنده آن دارای ترتیب و شمارش پذیر باشد . یعنی هر عضو آن نوع دارای یک مقدار قبلی یا بعدی باشد به ا ستثنای عضو اول و آخر. **توجه** : این نوع داده به انواع زیر تقسیم می شود : عددی صحیح – کاراکتری – منطقی – زیر قلمرو – شمارشی
زیر محدوده یا زیر قلمرو	این نوع از داده ، امکان تعریف محدوده دلخواهی را برای برنامه نویس به وجود می آورد. مثلاً Var a:۱..۲۰; **نکته** : حد بالا و پائین محدوده باید از نوع اسکالر باشند . Var b:'H'..'M'; Var c:False..True;
مجموعه ای	امکان تعریف گروهی ، تعداد محدودی از داده ها که از نظر نوع یکسان هستند را به برنامه نویس می دهد . مانند : روزهای هفته – ماه های سال و داده های نوع اسکالر (**نکته** : تعداد عناصر محدود به حداکثر ۲۵۶ می باشد .)
آرایه ای	آرایه : **مجموعه ای از خانه های(متغیرهای) پشت سر هم و هم نوع تحت یک نام** .که هریک از متغیرهای درون آرایه با یک شماره که به آن اندیس می گوییم از یکدیگر تمیز داده می شوند . (کاربرد : ایجاد ماتریس) Var A: Array[۱..۵] of Integer; A ۱ ۲ ۳ ٤ ٥
اشاره گر	امکان نگهداری آدرسهای حافظه را به وجود می آورد .

یادآوری : تبدیل هر عدد از مبنای ۲ به ۱۰ یا بلعکس

برای تبدیل کافی است مضربهای ۲ را از عدد ا بنویسید (۱ ۲ ۴ ۸ ۱۶ ۳۲ ۶۴ ۱۲۸ و ...)

سپس عددی را که می خواهید از مبنای ۲ به ۱۰ تبدیل کنید رقم به رقم در زیر مضربها یادداشت کنید حال اعدادی را که در زیر آنها رقم ۱ می باشد را بایکدیگر جمع کنید :

$(۱۰۰۱۱)_۲ = (\ ۱۹\ )_{۱۰}$ $(\ ۱۹\ )_{۱۰} = (۱۰۰۱۱)_۲$

۳۲	۱۶	۸	۴	۲	۱
	۱	۰	۰	۱	۱
۱+۲+۱۶=۱۹					

برای تبدیل عدد از مبنای ۱۰ به ۲ نیز به این صورت عمل می کنیم که مضربهای ۲ را از عدد ا را می نویسیم (۱ ۲ ۴ ۸ ۱۶ ۳۲ ۶۴ ۱۲۸ و ...) سپس از سمت چپ (بزرگترین) زیر اعدادی را که اگر آنها را با هم جمع کنیم عدد مورد نظر بدست می آید را ۱ می گذاریم .

آشنائی با کاربرد عملگر های : (Div Mod And Or Xor Not Shl Shr)

نحوه استفاده	توضیح	مثال
A Div B	خارج قسمت صحیح A را بر B رابدست می آورد .	۱۷ Div ۳ =۵
A Mod B	باقیمانده تقسیم صحیح A را بر B رابدست می آورد.	۱۷ Mod ۳ =۲
	نکته ۱ : در عملگرهای / و Div و Mod عملوند سمت راست (B) حتماً باید غیر صفر باشد. **نکته ۲** : در عملگرهای Div و Mod عملوندها باید از نوع صحیح باشند. **نکته ۳** : اگر یکی از عملوندها دارای مقدار منفی بود حاصل نیز منفی می باشد . **نکته ۴** : اگر هر دو عملوند + یا – بود جواب + خواهد بود .	-۱۷ Div ۳ = -۵ -۱۷ Div -۳ =۵

توجه : برای استفاده از عملگرهای بیتی بایستی عملوند های AوB به مبنای ۲ تبدیل شود. و عملگر بیت به بیت اعمال شود .

نکته * : میزان بیت های اشغال شده عدد در مبنای ۲ بر طبق میزان حافظه مصرفی نوع متغیر می باشد .

بطور مثال اگر عدد A=۲۱ فرض بگیریم(۱۰۱۰۱)۲ و در صورتیکه جنس A تعریف نشده باشد چون می تواند جزء مجموعه Byte یا Shortint باشد (پائین ترین مجموعه در نظر گرفته می شود) بنابراین فضای ۱ بایت معادل ۸ بیت را اشغال می کند پس خواهیم داشت : $(۰۰۰۱۰۱۰۱)_۲ = (۲۱)_{۱۰}$

نحوه استفاده	توضیح	مثال
A And B	حاصل And زمانی True می باشد که هر دو شرط برقرار باشد . نکته : در عملگرهای بیتی And و Or و Xor (**نکته** *)ضرورتی ندارد . چرا ؟ برای درک مطلب : **به توضیحات سر کلاس به دقت توجه کنید.** مثال : ۲۰ and ۱۴ = ۴	۱۶ ۸ ۴ ۲ ۱ A ۱ ۰ ۱ ۰ ۰ B . ۱ ۱ ۱ ۰ A and B ۰ ۰ ۱ ۰ ۰
A Or B	حاصل Or زمانی True می باشد که حداقل یکی از شرایط برقرار باشد . مثال : ۱۹ or ۹ = ۲۷	۱۶ ۸ ۴ ۲ ۱ A ۱ ۰ ۰ ۱ ۱ B . ۱ ۰ ۰ ۱ A Or B ۱ ۱ ۰ ۱ ۱
A Xor B	حاصل Xor زمانی True می باشد که فقط یکی از شرایط برقرار باشد . مثال : ۲۸ xor ۱۵ = ۱۹	۱۶ ۸ ۴ ۲ ۱ A ۱ ۱ ۱ ۰ ۰ B ۰ ۱ ۱ ۱ ۱ A XOr B ۱ ۰ ۰ ۱ ۱
Not(A)	Not هر عدد برابر (همان عدد + ۱) با تغییر علامت . در Not ، Shl ، Shr توجه به **نکته** * ضرورت دارد . برای درک مطلب : **به توضیحات سر کلاس به دقت توجه کنید** .	Not(۲۰)= -۲۱ Not(-۱۷)= ۱۶ Not(۱۲۵)= -۱۲۶
A Shl B	A را B بیت به سمت چپ شیفت می دهد . **به عبارتی** : A را B مرتبه در ۲ ضرب می کند . یا حاصل = $A*2^B$	۱۲ shl ۲ = ۴۸ ۱۲ shl ۳ = ۹۶ ۸ shl ۴ = ۱۲۸
A Shr B	A را B بیت به سمت راست شیفت می دهد . **به عبارتی** : A را B مرتبه بر ۲ تقسیم صحیح می کند . حاصل = $A \text{ div } 2^B$	۲۵۰ shr ۳ = ۳۱ ۶۵ shr ۴ = ۴ ۸۱ shr ۲ = ۲۰

نکته : در TP می توانیم از مقادیر عددی مبنای ۱۶ نیز بطور مستقیم استفاده کنیم . در این صورت قبل از عدد مورد نظر از علامت $ استفاده می کنیم . مثلاً : $(۱۶)_{۱۰}$ = ۱۰ $

آشنایی مختصر با توابع مرتبط باداده ای ترتیبی ، زیر قلمرو و شمارش پذیر :

Pred	تعیین عنصر قبلی از همان نوع داده	Pred("F") = "E"
Ord	شماره ترتیب آن داده (جواب به صورت عددی). شماره ترتیب طبق جول اسکی باشد	Ord("F")=۷۰
Succ	تعیین عنصر بعدی از همان نوع داده	Succ("F") = "G"
Chr(n)	کاراکتر معادل کد n در جدول اسکی را برمی گرداند . n عددی بین ۰ تا ۲۵۵ می باشد .	Chr(۷۰) = "F"

توجه : برای نوشتن توضیحاتی در مورد برنامه ، که در آینده فراموش نشود می توان توضیحات را در { } یا (* *) یادداشت نمود. توضیحات در اجرای برنامه هیچ تاثیری ندارد .

توانایی کار با دستورات خروجی :

Write( )	از این فرمان برای نمایش پیغامها و محتوای متغیرها و نتیجه عملیات محاسباتی استفاده می شود . مقدار متغیر یا عبارت رشته ای مورد نظر را چاپ کرده و اشاره گر در همان سطر باقی می ماند .
Writeln( )	از این فرمان برای نمایش پیغامها و محتوای متغیرها و نتیجه عملیات محاسباتی استفاده می شود . مقدار متغیر یا عبارت رشته ای مورد نظر را چاپ کرده و اشاره گر را به سطر بعد منتقل می کند .

نکته : در فرامین Writeln , Write نمی توان دادهای شمارشی و یا مجموعه ها را چاپ کرد .

به مثالهای زیر توجه کنید : **با فرض وجود : a=۱۲ b=۱۷,۴۵**	خروجی	مثال
عبارت مابین ' ' را عیناً چاپ می نماید .	Hadi	Write ('Hadi');
محتویات متغیر a را چاپ می نماید .	۱۲	Write(a);
اعداد اعشاری در پاسکال در صورت عدم تعریف محدوده صحیح به صورت نماد علمی با ۱۰ رقم اعشار چاپ می شوند . منظور از E۰۱ یعنی 10^{1}	۱,۷۴۵۰۰۰۰۰۰E۰۱	Write(b);
ابتدا عبارت مابین ' ' را عیناً چاپ می نماید ، سپس محتویات متغیرa	a=۱۲	Write('a=',a);
حاصل عبارت مورد بررسی را چاپ می نماید .	True	Write((۱۲<۱۴) and (۱۳>۵));

نکته ۱: برای قالب بندی چاپ رشته ها ، کاراکترها و اعداد صحیح کافی است یک میدان صحیح مشخص کنیم ;(n , عدد)Write

n میدان صحیح مورد نظر می باشد . یعنی از محل اشاره گر n فضا را ایجاد کن و سپس عدد مورد نظر را در آن محدوده چاپ کن . **البته** دقت داشته باشید که اگر n کمتر از طول رشته یا عدد مورد نظر باشد عین عبارت عددی یا رشته چاپ می شود **اما** اگر n از طول رشته یا عدد مورد نظر بزرگتر باشد قبل رشته یا عدد (از سمت چپ) فضای خالی ایجاد می شود .

به مثالهای زیر توجه کنید : **با فرض وجود : a=۲۴۵**	خروجی	مثال
چون محدوده کوچکتر از طول عدد می باشد ، محدوده در نظر گرفته نمیشود	۲۴۵	Write(a:۱);
چون محدوده کوچکتر از طول عدد می باشد ، محدوده در نظر گرفته نمیشود	۲۴۵	Write(a:۲);
* منظور از b فضای خالی (Space) می باشد .	bbbb۲۴۵	Write(a:۷);
نکته : در توربو پاسکال	hadi	Write("hadi":۱);
نکته : در پاسکال استاندارد	h	Write("hadi":۱);
کلمه hadi در محدوده مورد نظر چاپ می شود .	bbbbhadi	Write("hadi":۸);

نکته ۲: برای قالب بندی اعداد اعشاری (محدود کردن تعداد ارقام اعشار یا به عبارتی جلوگیری از عدم نمایش عدد اعشار بصورت نماد علمی) کافی است یک میدان نیز برای محدوده اعشار تعریف کنیم تا عدد مورد نظر بصورت ممیز شناور چاپ شود .

به مثالهای زیر توجه کنید :	خروجی	مثال
عدد با دقت یک رقم اعشار چاپ می شود .	۱۷۹۱,۲	Write(۱۷۹۱,۱۷۳۲:۵:۱);
عدد با دقت دو رقم اعشار چاپ می شود .	۱۷۹۱,۱۷	Write(۱۷۹۱,۱۷۳۲:۵:۲);
عدد با دقت یک رقم اعشار در فضای تعیین شده چاپ می شود	bbbbbb۱۷۹۱,۲	Write(۱۷۹۱,۱۷۳۲:۱۲:۱);
عدد با دقت ۶ رقم اعشار گرد شده و در فضای تعیین شده چاپ می شود	۱۷۹۱,۱۷۳۲۰۰	Write(۱۷۹۱,۱۷۳۲:۵:۶);

به مثالهای زیر توجه کنید :	خروجی	مثال
عدد با دقت یک رقم اعشار گرد می شود .	۲۷,۲	Write(۲۷,۱۷۵۶:۱:۱);
عدد با دقت دو رقم اعشار گرد می شود .	۲۷,۱۸	Write(۲۷,۱۷۵۶:۱:۲);
عدد با دقت یک سه رقم اعشار گرد می شود .	۲۷,۱۷۶	Write(۲۷,۱۷۵۶:۱:۳);
عدد با دقت یک رقم اعشار گرد شده و در فضای تعیین شده چاپ می شود	bbbbb۲۷,۲	Write(۲۷,۱۷۵۶:۹:۱);

نکته مهم در گرد شدن اعداد :

اگر در فرمان Write اعشار با دقت m رقم تعریف شده باشد ، شما بایستی m رقم از اعشار را جدا کرده و سپس با m رقم بعد از آن مقایسه کنید .

اگر ۱ رقم اعشار تعیین شده بود شما بایستی به ۱ رقم را جدا کرده سپس آن را با یک رقم بعد مقایسه کنید اگر آن یک رقم مساوی یا بزرگتر از ۵ بود آن عدد با ۱ رقم بالاتر گرد می شود. مثال Write(۲۵,۲۶:۱:۱) ابتدا ۱ رقم از اعشار را جدا کرده و سپس با ۱ رقم بعد از آن مقایسه می کنیم و چون رقم بعد از ۵ بزرگتر می باشد عدد با ۱ رقم بالاتر گرد می شود . درنتیجه :جواب ۲۵,۳ است. برای درک مطلب از استاد خود راهنمایی بگیرید . (برای ۲ رقم با ۵۰ برای ۳ رقم با ۵۰۰ و ... مقایسه را انجام دهید).

توانایی کار با دستورات ورودی :

Read( )	از این فرمان برای خواندن ورودیها استفاده می شود . مقدار متغیر مورد نظر را دریافت کرده و اشاره گر در همان سطر باقی می ماند .
Readln( )	از این فرمان برای خواندن ورودیها استفاده می شود . مقدار متغیر مورد نظر را دریافت کرده و اشاره گر را به سطر بعد منتقل می کند .

نکته ۱: داده های نوع منطقی (Boolean) ، شمارشی ، ثابتهای حقیقی ، متغیرهای مجموعه ای را نمی توان از ورودی دریافت کرد .

نکته ۲: نوع داده های ورودی باید با نوع متغیر متناظرشان در دستورات ورودی همخوانی (مطابقت) داشته باشد. یعنی وارد کردن اعداد اعشاری و یا کاراکتری برای متغیر عددی صحیح مجاز نیست ، در این حالت برنامه با صدور پیغام خطای زمان اجرا (Runtime Error) متوقف می شود .

نکته ۳: اگر برای متغیر کاراکتری عدد وارد شود ، عدد به عنوان کاراکتر ذخیره می شود .

نکته ۴: برای متغیر های اعشار می توان عدد صحیح وارد کرد اما کاراکتر مجاز نیست .

نکته ۵: در هنگام وارد کردن مقدار برای متغیر های عددی سیستم از هر Tab ، Blank و یا Enter صرفنظر می کند.
در زمان مقدار دهی به متغیر های عددی فضای خالی یا Enter به عنوان جدا کننده مقادیر در نظر گرفته می شود .

نکته ۶:مقادیر اعشار می توانند هم بصورت نماد علمی و هم بصورت ممیز شناور وارد شوند .

نکته ۷: در این دستورات نمی توان پیغام یا فرمت خاصی قرار داد .

دستور انتساب :=	شکل کلی بکارگیری دستور : a:=۲۴;	;مقدار یا عبارت مناسب =: نام متغیر
نکته ۱:	نوع داده موجود در سمت چپ و راست باید با یکدیگر سازگار ی داشته باشند .	
نکته ۲:	در مقدار دهی مستقیم به یک متغیر ، مقدار انتسابی نباید خارج از محدوده مجاز متغیر باشد .	
نکته ۳:	در مقدار دهی غیر مستقیم به یک متغیر ، مقدار انتسابی می تواند خارج از محدوده مجاز متغیر باشد .	

راجع به کلید Enter بیشتر بدانیم :

کلید Enter دارای **۲** کد است : ۱۳# (رفتن به ابتدای سطر جاری) و ۱۰# (رفتن به سطر بعد)

در حقیقت ترکیب این دو کد کار کلید Enter می باشد .یعنی بازدن کلید Enter مکان نما به ابتدای سطر بعد منتقل می شود .

(در زمان حل تست ها بیشتر بحث می کنیم)

نکاتی در مورد مقدار دهی به متغیر های کاراکتری (Char) :

نکته ۱ : در وارد کردن مقدار برای متغیر های کاراکتری ، جدا کننده خاصی (مثل Blank یا Enter) مطرح نمی شود .

نکته ۲ : در وارد کردن مقدار برای متغیرهای کاراکتری نباید از کاراکترهای " " یا ' ' استفاده کرد .

مثال : برنامه زیر را در نظر بگیرید :

```
Var
   M , N , O : char;
Begin
   Read ( M,N,O);
End.
```

مقدار ورودی	متغیر M	متغیر N	متغیر O	توضیحات
ABC	A	B	C	توجه به نکته ۱
A B	A	فاصله	B	فاصله به عنوان یک کاراکتر در نظر گرفته می شود .
'A'	'	A	'	توجه به نکته ۲
۱۲۳	۱	۲	۳	هر رقم به عنوان یک کاراکتر در نظر گرفته می شود .
B←	B	# ۱۳	# ۱۰	همانطور که گفتیم کلید Enter دارای ۲ کد است .

ساختارهای کنترلی : گاهی اوقات لازم است که در یک برنامه یک یا چند عمل را انتخاب نماییم ، یا از بین دو حالت ممکن ، یکی را انتخاب نماییم بنابراین نیازمند دستورالعملهای تکرار و انتخاب هستیم .

دستور شرطی IF :

استفاده از این دستور زمانی مفید است که بخواهیم از بین دو حالت ممکن یکی را انتخاب نماییم ، کاربرد آن به صورت زیر می باشد

IF شرط مورد ارزیابی Then دستور ۱ Else دستو ر ۲

نحوه کار بدین صورت است که پس از ارزیابی شرط منطقی ، اگر حاصل درست (True) بود دستور ۱ اجرا می شود و دستور ۲ اجرا نخواهد شد ولی چنانچه حاصل شرط نادرست(False) باشد دستور ۲ اجرا می گردد و دستور ۱ اجرا نخواهد شد.

نکته ۱ : دستورات ۱و ۲ نیز می توانند به صورت یک بلوک نیز باشند ، یعنی بجای اجرای فقط یک دستور می توان گفت در صورت برقرای شرط یا بلعکس ، مجموعه ای از دستورات اجرا شوند

```
IF شرط مورد ارزیابی Then
  Begin
  ..... ;
  ..... ;
  End;
Else
  Begin
  ..... ;
  ..... ;
  End;
```

```
IF شرط مورد ارزیابی Then

دستور ۱

Else

دستو ر ۲ ;
```

نکته ۲ : استفاده از قسمت Else اختیاری است .

نکته ۳ : اگر از Else استفاده می کنید نباید بعد از دستور مربوط به Then از سمیکالن **(;)** استفاده نمایید .

دستور If تو در تو : اگر در بررسی شرط بیش از دوحالت قابل انتخاب داشتیم بایستی از Ifهای تو در تو استفاده نماییم .

مثال : برنامه ای بنویسید که عددی را از ورودی دریافت نماید و تشخیص دهد که عدد مثبت ، منفی یا صفر است :

```
Program Sample۱
Uses Crt;
Var n:integer;
Begin
Clrscr;
Write('Enter Number :');
Readln(n);
If n>۰ then
Write('Positive')
Else
If n<۰ then
Write('Negetiv')
Else
Write ('Zero');
Readln;
End.
```

Else If : وقتی تعداد انتخاب ها زیاد باشد (بیش از ۳ انتخاب) کنترل If های تو در تو چندان ساده نیست بنابراین از ساختار Else If استفاده می کنیم :

مثال : برنامه ای بنویسید که نمره دانشجویی را از ۱۰۰ نمره دریافت و طبق جدول زیر رتبه آن را چاپ نمایید :

A	نمره > = ۹۰
B	۸۰≤ نمره < ۹۰
C	۷۰≤ نمره < ۸۰
D	۶۰≤ نمره < ۷۰
F	نمره < ۶۰

```
Program Sample۲;
Uses Crt;
Var Grade:Real;
Begin
Clrscr;
Write('Enter Grade :');
Readln(Grade);
If Grade >= ۹۰ then
   Weite('A')
Else if Grade >= ۸۰ then
   Write('B')
Else if Grade >=۷۰ then
   Write('C')
Else if Grade >=۶۰ then
  Write('D')
Else
  Write('F');
Readln;
End.
```

دستور Case : با استفاده از دستور Case شما قادر به بررسی شرط یا شرایط خاصی خواهید بود . در حقیقت عملکرد این دستور معادل استفده از چندین دستور If است . شکل کلی بکار گیری این دستور به صورت زیر است :

```
Case عبارت مورد ارزیابی Of
  مقدار۱ : دستور۱ ;
  مقدار ۲ : دستور۲ ;
    .
    .
Else
  دستور;
End;
```

توجه : مقدار ها می توانند به صورت تک مقداری یا چند مقدار (با استفاده از کاما) یا محدوده ای باشند .

نکته ۱: عبارتی که مورد ارزیابی قرار می گیرد و همچنین مقادیر استفاده شده ، حتماً باید از نوع ترتیبی باشند . استفاده از نوع Real غیر مجاز است .

نکته ۲ : موارد بکار رفته نباید تکراری باشند .

نکته ۳ : اگر مقادیر انتخابی مربوط به یک مورد پراکنده بود آنها را با کاما (,) از یکدیگر جدا می کنیم

نکته ۴ : اگر مقادیر انتخابی مربوط به یک مورد به دنبال هم و پیوسته بود می توانیم آنها را به صورت زیر قلمرو و با استفاده از علامت (..) مشخص نماییم .

نکته ۵ : وجود قسمت Else اختیاری است .

نکته ۶ : عبارت مقابل Case و مقادیر استفاده شده باید هم نوع باشند .

نکته ۷ : هر کدام از دستورات می توانند یک بلوک باشند .

نکته ۸ : عبارت مورد ارزیابی می تواند به صورت یک عمل محاسباتی یا منطقی باشد .

نکته ۹ : نوع String یک نوع ترتیبی نیست و نمی توان آنرا مقابل عبارت case قرار داد .

برای درک مطلب نکات به مثالهای زیر توجه نمایید :

```
Var N:integer;
Begin
Readln(N);
Case (N<۲۰) Of
   True :  Writeln(' ok ');
   False : writeln('No');
End;
```

```
Var N:integer;
Begin
Readln(N);
Case (N*۲+۸)  Of
   ۵   : Writeln('  ');
   ۷،۱۱،۱۵ : writeln(' ');
   ۱۸..۲۲  :  writeln(' ');
   Else
   Writln(' ');
End;
```

```
Var N:integer;
Begin
Readln(N);
Case N Of
   ۶   :  Writeln('  ');
   ۷،۱۱،۱۵ : writeln(' ');
   ۱۸..۲۲  :  writeln(' ');
   Else
   Writln(' ');
End;
```

آشنایی با حلقه های تکرار :

در بسیاری از موارد لازم است که یک یا چند دستورالعمل به دفعات معین و یا وابسته به شرطی ، تکرار شوند . از این رو حلقه های مختلفی در پاسکال وجود دارند که با آنها آشنا می شویم . درکل حلقه ها به دو دسته : معین و نامعین تقسیم می شوند:

حلقه های معین : تعداد دفعات انجام آنها مشخص می باشد . (For)

حلقه های نامعین: تعداد دفعات آنها نا مشخص و وابسته به شرط می باشد . که این نوع حلقه ها نیز به دو دسته مثبت و منفی تقسیم میشوند.

منظور از حلقه تکرار شرطی مثبت : انجام عملیات تا هنگامی که شرط برقراراست .(While)

منظور از حلقه تکرار شرطی منفی : انجام عملیات تا هنگامی که شرط برقرارنیست. (Repeat)

حلقه تکرار معین (For) :

این دستور با استفاده از یک شمارنده (Counter) ، یک دستور یا بلوکی از دستورات را به تعداد دلخواه ما تکرار می کند .
از این حلقه به دو صورت افزایشی یا کاهشی می توان استفاده کرد .
شکل کلی دستور به صورت زیر است :

```
For نام شمارنده := مقدار اولیه To مقدار نهائی Do
Begin
.....
.....
End;
```

```
For نام شمارنده := مقدار اولیه To مقدار نهائی Do
دستور;
```

مثال :

```
Var I:integer;
Begin
For I:= ۱۰ downto ۱do
Write(i);
End.
```

```
Var I:integer;
Begin
For I:= ۱ to ۱۰
do
Write(i);
End.
```

```
Var lname:string;
    I:integer;
Begin
Write('Enter Last name:');
Readln('lname');
For I:= ۱ to ۱۰ do
Write('lname');
End.
```

نکته ۱ : در حلقه For افزایشی برای اینکه دستورات بدنه حلقه ، حداقل یک بار تکرار شود ، مقدار اولیه باید کوچکتر یا مساوی مقدار نهائی باشد . اما درحلقه For کاهشی مقدار اولیه باید بزرگتر یا مساوی مقدار نهائی باشد.

نکته ۲ : در حلقه For شرط حلقه ابتدای ورود به حلقه مورد بررسی قرار میگیرد .

نکته ۳ : متغیر حلقه حتماً باید از نوع ترتیبی باشد (عددی صحیح ، کاراکتری ، منطقی) .استفاده از متغیر نوع Real یا String مجاز نیست .

نکته ۴ : مقدار اولیه و نهایی باید با یکدگر سازگاری داشته باشند .

نکته ۵ : بهتر است مقدار متغیر حلقه For در ، درون حلقه تغییر داده نشود ، البته با تغییر متغیر درون حلقه پیغام خطایی از سوی کامپایلر داده نمی شود ولی ممکن است اجرای برنامه دچار مشکل شود .

نکته ۶ : تغییر مقدار اولیه یا نهائی ، درون حلقه For هیچ تأثیری بر روند اجرای برنامه ندارد .

نکته ۷ : در انتهای دستور For نباید علامت سمی کالن (;) استفاده شود . چون در اینجا حلقه نمی تواند دستورات بعد از Do را اجرا کند چرا که با رسیدن به سمی کالن دستور پایان یافته است . البته استفاده از سمی کالن در انتهای دستور سبب ایجاد حلقهای تاخیر می شود .

نکته ۸ :در صورتی که بدنه حلقه بیس از یک دستور داشته باشد بایستی دستورات بین بلاک (Begin End;) قرار گیرند.

نکته ۹ : تعداد تکرار حلقه For از فرمول زیر محاسبه می شود .

در حلقه For افزایشی = (مقدار نهائی – مقدار اولیه) + ۱

در حلقه For کاهشی = (مقدار اولیه – مقدار نهائی) + ۱

نکته ۱۰ : برای محاسبه تعداد دفعات تکرار در حلقهای تکرار تو در تو : تعداد دفعات هریک را جدا گانه طبق نکته قبل محاسبه و سپس حاصل را در یکدگر ضرب کنید .

حلقه تکرار شرطی مثبت (While) :

هرگاه بخواهیم دستور یا دستوراتی تا برقرار بودن شرطی خاص تکرار شوند از این نوع حلقه استفاده می کنیم که شکل کلی آن به صورت زیر است :

While **شرط مورد ارزیابی** **Do**
دستور;

While **شرط مورد ارزیابی** **Do**
Begin
……;
……;
End;

While **شرط مورد ارزیابی** **Do**
دستور;

حلقه تکرار شرطی منفی (Repeat) :

هر گاه بخواهیم چند دستوربه صورت مداوم اجرا شوند از دستور حلقه سازی Repaear استفاده می نماییم و تفاوت آن با حلقه While در این است که تا زمانی که یک شرط منطقی خاص برقرار نشده است حلقه ادامه دارد . شکل کلی دستور به صورت زیر می باشد :

Repeat دستور Until شرط یا عبارت منطقی ;

نحوه کار بدین صورت است که ابتدا یکبار بدون بررسی شرط ، دستورات حلقه انجام انجام می شود سپس شرط یا عبارت منطقی ارزیلبی می گردد .
اگر نتیجه نادرست باشد حلقه یکبار دیگر تکرار می شود و اگر شرط درست باشد حلقه پایان می یابد .

نکته ۱ : در حلقه While اجرای حلقه منوط به برقراری شرط است ولی در حلقه Repeat اجرای حلقه منوط به برقرار نبودن شرط است .

نکته ۲ : در دستور While چنانچه دستورات بدنه حلقه بیش از یکی باشد ، بایستی درون یک بلوک قرار گیرد ولی در دستور Repeat نیازی به بلوک نیست .

نکته ۳ : در دستور While شرط حلقه در همان ابتدای ورود به حلقه بررسی می شود ولی در repeat دستورات درون Repeat یکبار اجرا شده سپس شرط مورد ارزیابی قرار می گیرد .

نکته ۴ : حلقه های متداخل باید به صورتی باشد که یکی کاملاً درون دیگری قرار بگیرد ، نه آنکه یکدیگر را قطع نمایند .

نکته ۵ : در حلقه های متداخل ابتدا بیرونی ترین حلقه ، اولین دور خود را آغاز می کند ولی از بقیه حلقه ها دیر تر پایان می یابد .

نکته ۶ : برای محاسبه تعداد دفعات تکرار در حلقهای تکرار تو در تو : تعداد دفعات هریک را جدا گانه طبق نکته قبل محاسبه و سپس حاصل را در یکدگر ضرب کنید .

یادآوری : دستورات درون بدنه Repeat نیازی به Begin…End; ندارند .

خروج اضطراری از حلقه به کمک دستور Break :

برخی مواقع ممکن است لازم باشد تا بسته به یک شرط خاص ، کار حلقه پیش از موئد خاص پایان پذیرد . در این صورت از دستور Break برای متوقف کردن حلقه استفاده می کنیم و کنترل برنامه به سطر بعد از انتهای حلقه منتقل می شود . Break;

دستور Continue :

این دستور سبب می شود تا برنامه از اجرای بقیه دستورات درون حلقه صرفنظر نموده و با رفتن به ابتدای حلقه دور جدیدی را آغاز نماید .

نکته : دستورات Break و Continue فقط درون حلقه ها کاربرد دارد و در توربو پاسکال ۷ وارد شده اند .

آرایه

به تعدادی متغیر هم نوع و پشت سر هم ، تحت یک نام در حافظه اصلی اطلاق می شود. هریک از متغیرهای درون آرایه با یک شماره که به آن اندیس می گویند از یکدیگر تمیز داده می شوند . (آرایه را می توان مانند یک جدول یا یک ماتریس در نظر داشت که این جدول یا ماتریس می تواند دارای ابعاد مختلف باشد . یک بعدی ، دو بعدی و ...).

نحوه تعریف آرایه در پاسکال : Var نام متغیر : Array [کران پایین..کران بالا] Of نوع ;

مثال : آرایه A را می توان معادل یک جدول شامل ۱ ستون و ۵ سطر تصور کرد .

```
Var A:Array[۱..۵] Of Integer;
```

مثال : آرایه B را می توان معادل یک جدول شامل ۵ ستون و ۴ سطر تصور کرد .

```
Var B:Array[۱..۵،۱..۴] Byte;
```

آرایه ۱ بعدی

A	۱
۱	
۲	
۳	
۴	
۵	

آرایه دو بعدی

B	۱	۲	۳	۴	۵
۱					
۲					
۳					
۴					

مثال : برای دسترسی به عناصر آرایه کافی است نام آرایه و سپس اندیس آرایه مورد نظر را بنویسیم :

```
A[۲] = ۱۷ ;
B[۲،۴] = ۲۵;
```

مثال : تکه برنامه زیر نمرات ۵ درس دانشجوئی را از ورودی دریافت و در یک آرایه ذخیره می کند :

```
Var N:Array[۱..۵] of Integer;
    i:Integer ;
Begin
For i:= ۱ to ۵ do
Readln(N[i]);
End.
```

مثال : تکه برنامه زیر نمرات ۵ درس دانشجوئی را از ورودی دریافت و در یک آرایه ذخیره و حاصل جمع و معدل وی را محاسبه و چاپ می نماید :

```
Var N:Array[۱..۵] of Integer;
    i,S:Integer ;
Begin
For i:= ۱ to ۵ do
    Begin
    Readln(N[i]);
    S:=S+I;
    End;
Writeln('Sum=',S);
Writeln('Avg=',S/۵ : ۱ :۲ );
End.
```

مثال : برنامه زیر نمرات ۷ درس دانشجوئی را از ورودی دریافت و دریک آرایه ذخیره می کند و سپس بالاترین نمره وی را چاپ می نماید :

```
Var N:Array[۱..۷] of Integer;
    i,max:Integer ;
Begin
For i:= ۱ to ۷ do Readln(N[i]);
Max:=N[۱];
For i:=۲ to ۷ do
   If N[i]>Max then  Max:=N[i];
Writeln('Max:=',Max);
End.
```

نکته ۱ : امکان انتساب دو آرایه هم نوع و هم اندازه در پاسکال وجود دارد :

```
Var A,B:Array[۱۷..۲۵] of Byte;
For i:= ۱۷ to ۲۵ do A[i]:=B[i];
```

اگر بخواهیم محتوای آرایه B را درون آرایه A قرار دهیم :

نکته ۲ : از طرفی چون دو آرایه Aو B هم نوع و هم اندازه هستند می توان با یک دستور این کار را انجام داد :

```
A:=B;
```

نکته ۳ : آرایه دو بعدی را نیز می توان بصورت زیر هم تعریف کرد :

```
Var A:array[۱..۵] of  B:Array [۱..۵] Of  Byte;
```

نحوه محاسبه حافظه مصرفی آرایه های یک بعدی : میزان حافظه مصرفی بر حسب نوع * (۱ + (کران پایین – کران بالا))

نحوه محاسبه حافظه مصرفی آرایه های چند بعدی :

میزان حافظه مصرفی بر حسب نوع * (۱ + (کران پایین – کران بالا)) * (۱ + (کران پایین – کران بالا))

بعد دوم | بعد اول

به عبارتی : تعداد عناصر آرایه * میزان حافظه مصرفی بر حسب نوع

مثال : میزان حافظه مصرفی آرایه X عبارت است از :

```
Var X:Array[۱..۱۰،۳..۹،۰..۵] of  Integer;
```

با توجه به اینکه هر متغیر از نوع Integer ۲ بایت از حافظه را اشغال $(۱۰-۱)+۱ * (۹-۳)+۱ * (۵-۰)+۱ * ۲ = ۸۴۰$ می کند بنابراین این آرایه ۸۴۰ بایت از حافظه را اشغال میکند .

نکاتی در مورد اندیس آرایه ها :

اندیس آرایه هر نوع داده ترتیبی می تواند باشد . (البته محدودیتهای نیز دارد که به شرح آنها خواهیم پرداخت)

یادآوری : داده های ترتیبی در پاسکال عبارتند از : Integer, Longint , Shortint, Byte ,Char, Boolean، زیر قلمرو

نکته ۱ : نوع های char, Byte, Boolean و شمارشی و زیر قلمرو ، مستقیماً می توانند در قسمت اندیس آرایه ها بکار روند .

```
Var n:Array[chare] of  Real;
Var m:Array[byte] of  Integer;
Var x:Array['a' .. 'f'] of integer;
```

نکته ۲: استفاده مستقیم از نوع Integer در قسمت اندیس غیرمجاز است .

نکته ۳ : اندیس آرایه نمی تواند از نوع Longint ، Real یا String باشد .

نکته ۴: حداکثر سقف مجاز متغیرهای حافظه های سراسری در پاسکال ۶۴K است . بنابراین متغیرهای که حافظه بیشتری را اشغال کنند با یکی از پیغامهای خطای زیر مواجه می شوند :

- Too many Variables
- Data Segment too large
- Structure to large

بطور مثال تعریف آرایه زیر باعث به وجود آمدن یکی از خطاها می شود : Var n:Array[۱..۲۵۰۰۰] of Real;

چرا ؟ چون حافظه مصرفی اشغال شده برابر با **۱۵۰۰۰۰** = ۶ * ۱ + (۲۵۰۰۰-۱) می شود که از حد مجاز ۶۴۰۰۰ تجاوز میکند .

نکته ۵ : کران پایین یک آرایه نمی تواند کمتر از ۳۲۷۶۸- باشد .

نکته ۶ : کران بالای یک آرایه نیز اگر کران پائین مثبت با صفر باشد نمی تواند بیشتر از ۶۵۵۳۶ باشد .

نکته ۷ : کران بالای یک آرایه ، اگر کران پائین منفی باشد نمی تواند بیشتر از ۳۲۷۶۷ باشد.

نکته ۸ : اندیس آرایه از هرنوع و در هر محدوده ای که تعریف شود، در طول برنامه نیز اندیس باید از همان نوع و در همان محدوده باشد . به برنامه مقابل توجه کنید :

```
Type Color=(Red,Green,Blue);
Var A:Array[False .. True] of  Byte;
    B:Array['A' .. 'C'] of  ۱..۱۰۰;
    C:Array[Red .. Blue] of Color;
Begin
A[۲۵<۷۰]:= ۲۰;
B['A']:=۳۵;
C[Red]:= red;
End.
```

روشهای جستجو و مرتب سازی :

تعریف لیست:

لیست از تعدادی داده تشکیل شده است که با یکدیگر در خاصیتی مشترک هستند و تعریف لیست بسیار شبیه تعریف مجموعه در ریاضی است با این تفاوت که:

- در مجموعه(Set) عضو تکراری وجود ندارد اما در لیست ممکن است داده تکراری وجود داشته باشد .
- در مجموعه ترتیب اعضا اهمیت ندارد اما در لیست ترتیب داده ها اهمیت دارد .

نکته: یک مجموعه می تواند حتما لیست باشد اما یک لیست همیشه نمی تواند یک مجموعه باشد .

نکته: داده های موجود در یک لیست می تواند از چند نوع داده متفاوت باشد مثلا (۱۸ , ۱۹,۵ , ۱۲ , Hadi)

دقت شود که داده های یک لیست در خاصیتی باید مشترک باشند . (مثلاً نوع داده)

نکته: رکورد و آرایه می توانند یک لیست باشند.

جستجودر لیست: در مواردی که می خواهیم یک داده خاصی را در یک لیست پیدا کنیم ، به داده مورد جستجو **کلید** می گوییم .

انواع روش های جستجو در لیست ها وجود دارد که ۲ به مورد از آنها اشاره می کنیم :

روش جستجوی خطی(ترتیبی)

در این روش ، کلید(داده ایی که قرار است جستجو شود) با اولین دادهٔ لیست مقایسه می شود اگر پیدا شد کار به پایان می رسد در غیر این صورت با دومین دادهٔ لیست مقایسه می شود و این کار تا پایان لیست ادامه می یابد.

مثال) اگر کلید ۱۲ باشد در لیست زیر چند عمل مقایسه انجام می شود تا کلید پیدا شود؟ (به روش خطی)

۱	۲۳	۱۸	۱۹	۲۰	۱۲	۱۴	۱۹	۸	۱۲

نکته: اگر در لیست چند مورد تکراری از کلید باشد در جستجو به روش خطی فقط اولین کلید را پیدا می کند .

در این برنامه پس از عمل جسجو در صورت یافتن کلید ، اندیس (فیلد) لیست را برمی گرداند و کار جستجو تمام می شود. در مثال بالا دو عدد ۱۲ وجود دارد اما اولین ۱۲ که در خانه ششم است پیدا می شود و کار تمام می شود.

نکته: در روش جستجوی خطی لازم نیست که لیست ما مرتب شده باشد یعنی داده ها در لیست صعودی یا نزولی باشند.

نکته: در روش جستجوی خطی در صورتیکه لیست نامرتب باشد، تعداد مقایسه ها زیاد است و جستجو به کندی صورت می گیرد.

نکته: در بهترین حالت جستجوی خطی عمل مقایسه یکبار انجام می شود و در بدترین حالت جستجوی خطی به تعداد خانه های لیست عمل مقایسه انجام می شود. بنابراین اگر لیست N تا عنصر داشته باشد N عمل مقایسه انجام می شود. در حالت متوسط N/۲ عمل مقایسه انجام می شود.

```
Var   A: Array[۱..۱۰] of integer;
      Key , i:integer;  T:Boolean;
Begin
  For  i:=۱  to  ۱۰ do
     Readln( A[i]);
  Readln(Key);
  T:=False;
  For i:=۱  to  ۱۰  do
     IF  A[i] = Key  then
          Begin
               T:=True;
               Break;
          End;
  If  T=False  then
      Writeln(' Not Found ')
  Else
      Writeln(' Find ' , i);
End.
```

روش جستجوی دودویی (باینری): در این روش باید لیست از قبل مرتب باشد. پس این روش مخصوص لیست هایی است که از قبل مرتب(صعودی یا نزولی) شده اند. فرض کنید که یک آرایه مرتب صعودی (از کوچک به بزرگ) داریم و می خواهیم کلیدی را در آن به روش باینری جستجو کنیم : آرایه را از وسط نصف می کنیم و کلید را با عنصر وسط آرایه مقایسه می کنیم که سه حالت ممکن است رخ دهد:

الف) کلید از عنصر وسط بزرگتر باشد Key > A[i] : در این حالت با توجه به اینکه لیست مرتب (صعودی) است حتما کلید در نیمه سمت راست است و ما باید نیمه سمت راست را جستجو کنیم و نیمه سمت چپ را جستجو نخواهیم کرد.

	نیمه سمت راست	وسط	نیمه سمت چپ	

ب) کلید از عنصر وسط لیست کوچکتر است Key < A[i] : در این حالت با توجه به اینکه لیست به صورت صعودی مرتب است حتما کلید در نیمه سمت چپ است و باید نیمه سمت چپ را باید جستجو کرد و نیمه سمت را جستجو نخواهیم کرد.

ج) در بهترین حالت کلید مساوی عنصر وسط لیست باشد. در این حالت کلید پیدا شده است و عمل جستجو متوقف می شود.

اگر هر کدام از حالت های **الف** یا **ب** رخ دهد عمل جستجو را در نیمه سمت چپ یا راست ادامه می دهیم. به عبارت دیگر نیمه سمت چپ یا نیمه سمت راست را دوباره از وسط نصف می کنیم و دوباره عمل بالا را روی آن انجام می دهیم.

```
Var   A:array[۱..۲۰] of integer;
  Key , I , J , N , S : integer;
F:Boolean;
Begin
  For   J:= ۱  to ۲۰  do
        Readln( A[ J ] )   ;
   Sort(A);  { یک زیربرنامه جهت مرتب سازی}
S:=۱ ;   N : =۲۰;
Readln(Key);
F:=False;
While  (S<= N)  and ( Not  F)  Do
  Begin
      I:= ( S+N) Div ۲;
      IF  A[ I]  < Key  then
          S:=I+۱
     Ese
      IF  A[I] > Key  then
           N:=I-۱
       Else
      IF      A[I]=Key then
          F:=True;
  End;
IF  F=False then
  Writeln(' Not Found')
Else
   Writeln(' Find in index' , I);
End.
```

نکته: در بهترین حالت جستجوی دودویی یک عمل مقایسه انجام می شود و در بدترین حالت جستجوی دودویی یک لیستی که دارای N عنصر باشد تعداد عمل جستجو برابر با $\mathbf{[Log_2(N)] + 1}$ است.

نکته: جستجوی اطلاعات غیر عددی(رشته یا کاراکتر) دقیقا مانند اطلاعات عددی است.

نکته: معمولا در حالتهایی که تعداد عناصر لیست زیاد باشد جستجوی دودویی بسیار بهتر عمل می کند.

روشهای مرتب سازی لیست:

مرتب سازی: منظور از مرتب سازی داده ها تغییر دادن موقعیت آنها در یک لیست است تا داده ها در لیست به طور منظم صعودی یا نزولی قرار گیرند.

صعودی : از کمتر به بیشتر ۱،۵،۹،۱۵،۲۰،۲۵,..

نزولی : از بیشتر به کمتر ۹۰،۷۸،۳۴،۲۵،۱,..

توجه : ((درهمه مثالهای زیر فرض بر این است که می خواهیم لیست را بصورت صعودی مرتب کنیم)).

روش مرتب سازی حبابی: Bubble Sort

در این روش در دور اول (بار اول) اولین عدد لیست با دومین عدد لیست مقایسه می شود و اگر ترتیب آنها درست نبود جای آنها را عوض می کنیم.(یعنی اینکه عدد اولی از دومی بزرگتر باشد) سپس عدد دومی را با عدد سومی مقایسه می کنیم و اگر ترتیب آنها درست نبود جای آنها را عوض می کنیم. و این کار را تا انتهای لیست (فرض بر این است که N تا سلول دارد) ادامه می دهیم. عدد بزرگتر به خانه N ام می رود (مانندیک حباب) **سپس در دوردوم** (حلقه دوم) از خانه اول شروع می کنیم و خانه اول را با خانه دوم مقایسه می کنیم و اگر ترتیب آنها درست نبود جای آنها را عوض می کنیم و سپس عدد دوم وعدد سوم مقایسه می شوند و ..... و این کار تا خانه N-۱ ادامه پیدا می کند. **درمرحله سوم** نیز مانند مرحله دوم است با این تفاوت که تا خانه N-۲ انجام می شود. و الی آخر

سئوال : تا چند مرحله کار را ادامه می دهیم :

جواب : اگر لیست داری N تا سلول باشد N-۱ مرحله خواهیم داشت.

چرا !!! چون عناصر مرتب شده مجدداً مورد بررسی قرار نگیرند و سرعت عمل با لاتر رود. برای درک مطلب به **مثال ۲** توجه کنید :

مثال۱ : برنامه ایی بنویسید که یک آرایه ۶ عنصری را دریافت کندو به روش حبابی مرتب صعودی کند و نتیجه را نیز چاپ کند؟

```
Var
  A:Array[۱..۶] of integer;
  I, J , Tmp :Integer;
Begin
 For I:=۱ To ۶ do
    Readln(A[I]);
.....................................................
 For I:=۵ downto ۱ do
   For J:=۱ To I Do
     If A[ j] > A[j+۱] then
         Begin { جابجایی }
             Tmp:=A[j];
             A[j]:=A[j+۱];
             A[j+۱]:=Tmp;
          End;
.....................................................
For I:=۱ to ۶ do
   Writeln(A[I] );
End.
```

*** نکته بسیار مهم در ارتباط با ساختن حلقه مورد نیاز :**

با داشتن یک آرایه N عنصری

- **چون در بعضی از حلقه ها عنصر های مرتب شده نیز مجددا مورد بررسی بیهوده قرار می گیرند مانند :**

```
For  I:= ۱ to N do
  For J:= ۱ to  I  do
```

- **از این حلقه برای جلوگیری از عمل تکرار مقایسه استفاده می کنیم :**

```
For  I:= ۱ to N-۱ do
  For J:= i downto  ۱  do
```

یا

```
For  I:= N-۱ downto ۱ do
  For J:= ۱ to  I  do
```

- حلقه I تعیین کننده تعداد مراحل بررسی
- حلقه J تعیین کننده تعداد مقایسه در هر دور

((برای درک مطلب به مثال زیر توجه کنید))

مثال ۲: آرایه روبرو را به روش حبابی مرتب صعودی کنید؟

۹	۱۳	۱۷	۱۰	۱۸	۱۲

۶	۵	۴	۳	۲	۱
۱۸	۹	۱۳	۱۷	۱۰	۱۲
۱۸	۹	۱۳	۱۷	۱۲	۱۰
۱۸	۹	۱۳	۱۷	۱۲	۱۰
۱۸	۹	۱۷	۱۳	۱۲	۱۰
۱۸	۱۷	۹	۱۳	۱۲	۱۰

مرحله ۲ : چهار عمل مقایسه و سه جابجایی رخ داده است.

۶	۵	۴	۳	۲	۱
۹	۱۳	۱۷	۱۰	۱۸	۱۲
۹	۱۳	۱۷	۱۰	۱۸	۱۲
۹	۱۳	۱۷	۱۸	۱۰	۱۲
۹	۱۳	۱۸	۱۷	۱۰	۱۲
۹	۱۸	۱۳	۱۷	۱۰	۱۲
۱۸	۹	۱۳	۱۷	۱۰	۱۲

مرحله ۱ : ۵ عمل مقایسه و چهار جابجایی رخ داده است.

۶	۵	۴	۳	۲	۱
۱۸	۱۷	۹	۱۳	۱۲	۱۰
۱۸	۱۷	۹	۱۳	۱۲	۱۰
۱۸	۱۷	۹	۱۳	۱۲	۱۰
۱۸	۱۷	۱۳	۹	۱۲	۱۰

مرحله ۴ : دو عمل مقایسه و یک جابجایی رخ داده است.

۶	۵	۴	۳	۲	۱
۱۸	۱۷	۹	۱۳	۱۲	۱۰
۱۸	۱۷	۹	۱۳	۱۲	۱۰
۱۸	۱۷	۹	۱۳	۱۲	۱۰
۱۸	۱۷	۱۳	۹	۱۲	۱۰

مرحله ۳ : سه عمل مقایسه و یک عمل جابجایی رخ داده است .

۶	۵	۴	۳	۲	۱
۱۸	۱۷	۱۳	۱۲	۹	۱۰
۱۸	۱۷	۱۳	۱۲	۱۰	۹

مرحله ۵ : یک مقایسه و یک جابجایی رخ داده است **.**

در این مثال مجموعا ۱۵ عمل مقایسه و ۱۰ عمل جابجایی صورت گرفته است .

- **نکته ۱:** بطور کلی در روش مرتب سازی حبابی اگر لیست دارای N تا عنصر باشد به تعداد **N(N-۱) / ۲** مقایسه انجام می شود.
- **نکته ۲:** تعداد جابجایی ها در روش حبابی را دقیقا نمی توان محاسبه کرد و بستگی به بی نظمی آرایه دارد ولی می توان گفت که حداکثر تعداد عمل جابجایی ها به اندازه تعداد مقایسه ها است یعنی هر مقایسه ایی که انجام می شود یک جابجایی نیز رخ دهد.
- **نکته ۳ :** اگر آرایه ایی نزولی باشد و بخواهیم آن را به روش حبابی صعودی کنیم تعداد جابجایی ها برابر تعداد مقایسه ها است.

روش مرتب سازی انتخابی (Selection Sort) :

فرض کنید می خواهیم یک لیست شامل N خانه را به روش انتخابی مرتب کنیم(صعودی):

- **مرحله۱ یا دور اول :** از ابتدا تا انتهای لیست را جستجو کرده و کوچکترین عدد را پیدا کرده و محل آن را با اولین عنصر لیست عوض می کنیم.
- **مرحله۲ یا دور دوم :** از خانه دوم تا انتهای لیست را جستجو کرده و کوچکترین عدد را پیدا کرده و محل آن را با خانه دوم عوض می کنیم.
- **مرحله۳ یا دور سوم** :از خانه سوم تا انتهای لیست را جستجوکرده و کوچکترین عدد را پیدا کرده و محل آن را با سومین عنصر لیست عوض می کنیم.
- این کار را تا انتهای لیست انجام می دهیم .
- دقت کنید که در هر مرحله محل واقعی یکی از اعداد مشخص می شود.
- اگر لیست دارای N تا عنصر باشد N-۱ گذر یا فاز خواهیم داشت و در هر مرحله فقط یک جابجایی رخ می دهد.

آرایه روبرو را به روش انتخابی مرتب صعودی کنید؟

۱۲	۱۸	۱۰	۱۷	۱۳	۹

۹	۱۰	۱۲	۱۷	۱۳	۱۸

گذر ۳ : ۳ مقایسه و یک جابجایی

۹	۱۰	۱۸	۱۷	۱۳	۱۲

گذر ۲ : ۴ مقایسه و یک جابجایی

۹	۱۸	۱۰	۱۷	۱۳	۱۲

گذر ۱ : ۵ مقایسه و یک جابجایی

۹	۱۰	۱۲	۱۳	۱۷	۱۸

گذر ۵ : ۱ مقایسه و یک جابجایی

۹	۱۰	۱۲	۱۳	۱۷	۱۸

گذر ۴ : ۲ مقایسه و یک جابجایی

- در این مثال ۱۵ عمل مقایسه و چهار عمل جابجایی رخ داده است.
- همانطور که مشاهده می کنید همین آرایه به روش حبابی ۱۰ عمل جابجایی داشت و به روش انتخابی ۴ عمل جابجایی داشت.
- در روش مرتب سازی انتخابی تعداد مقایسه ها برابراست با $N (N-۱) /۲$

$(N-۱)+(N-۲)+(N-۳)+\ldots.+۲+۱ = N*(N-۱) /۲$

- در روش انتخابی تعداد جابجایی ها همیشه برابر N-۱ تا است.
- تعداد مقایسه ها در روش انتخابی و حبابی یکسان است.
- تعداد جابجایی ها در روش انتخابی کمتر از روش حبابی است به همین دلیل روش انتخابی سریعتر از حبابی است.
- مرتب سازی لیست های غیر عددی نظیر رشته ها دقیقا مانند لیست های عددی است.