" به نام دادار پاک "

مقدمه :

این کتاب به منظور ارتقا۶ سطح برنامه نویسی در بین کاربران کامپیوتر به صورت الکترونیک و به همراه نرم افزار حل تمرین در مورد نامگذاری متغیرها و خلاصه درس با فرمت جار (**jar**) که توسط اکثر گوشی های موبایل پشتیبانی میشود به صورت کاملا رایگان توزیع شده است .
در نوشتن این **pdf** بیشتر سعی در به وجود آوردن مرجعی تقریبا کامل برای زبان پاسکال بود. برای رسیدن به این هدف بیشتر تلاش براین شد تا فصولی که در اینترنت به طور کامل در موردشان مطلبی نیست کاملا پوشش داده شوند .

بعد از اکثر فصول مهم کتاب تمرینات مهم تحت عنوان کارگاه حل تمرین ارائه شده اند که صرف وقت برای این قسمت توسط خواننده خالی از سود نیست در متن کتاب هم سعی شده است تا با ایجاد محیطی دوستانه روند یادگیری شما بهبود پیدا کند همچنین شخصیتی مجازی به نام غضنفر در قسمت هایی از کتاب مخاطب من بود که همین جا از کسانی که از این شیوه ناراحت شده اند عذر خواهی میکنم .
در پایان ازجناب دانشجو مدیر وبلاگ **garmdareh1386.blogfa.com** که با نوشتن مطالب فصل آرایه ها این حقیر را در سریع تر پایان دادن کتاب یاری کردند تشکر میکنم .

با تشکر : بابک حاج عظیم زنجانی **(babak_cyclops)**

فهرست مطالب موجود :

مطالب

1 . الگوریتم
2 . الگوریتم های حلقوی
3 . برنامه نویسی چیست ؟
4 . انواع زبان های برنامه نویسی ؟
5 . مقدمه ای بر پاسکال
6 . مترجم های زبان پاسکال
7 . نکاتی در مورد برنامه های پاسکال
8 . مشخصات برنامه ی مطلوب

نویسنده

9 . اجزای تشکیل دهنده ی یک برنامه در پاسکال
10 . شرح قسمت اعلان
11 . نام برنامه و اعلان آن
12 . متغیر ها واعلان آنها
13 . عملگرها
14 . ثابت ها و اعلان آنها
15 . برچسب ها و اعلان آنها
16 . دستورات ورودی و خروجی
17 . فرمت خروجی
18 . حلقه ها و شروط
19 . انواع دستور if
20 . دستور select و کاربرد آن
21 . حلقه های تکرار
22 . کارگاه حل تمرین حلقه ها (تشریحی)
23 . پروسیجرهای کاربردی در حلقه ها

نویسنده

24 . آرایه ها
25 . آرایه های یک بعدی
26 . آرایه های چند بعدی
27 . مرتب سازی آرایه ها
28 . جستجو در آرایه های مرتب

مطالب

29 . توابع و روال های کتابخانه ای
30 . توابع برای اعداد صحیح و اعشاری
31 . توابع از نوع کاراکتری
32 . روالهای استاندارد
33 . کارگاه حل تمرین توابع و روال های کتابخانه ای (تستی)
34 . کاراکترها و رشته ها
35 . توابع و روال های کتابخانه ای پاسکال برای کار با رشته ها
36 . کارگاه حل تمرین کاراکترها و رشته ها (تستی تشریحی)

نویسنده

37 . برنامه های فرعی (زیر برنامه ها)
38 . روال ها
39 . توابع

مطالب

40 . توابع بازگشتی
41 . تعریف نوع جدید
42 . مجموعه ها و داده های شمارشی
43 . مجموعه ها
44 . داد های شمارشی
45 . رکوردها
46 . بدست آوردن حجم یک رکورد
47 . آرایه ای از رکوردها

نویسنده

48 . فایل ها
49 . فایل های متنی
50 . خواندن و نوشتن پرونده ها
51 . توابع Eoln و Eof
52 . کارگاه حل تمرین فایل ها
53 . آخر خط
54 . ضمیمه ها
- چند دستور اضافی اما کاربردی
- کلی مثال و تمرین
- جدول اسکی
- فهرست منابع

الگوریتم : (Algorithm)

برای حل یک مسئله لازم است عملیاتی را قدم به قدم انجام داد تا نهایتا به حل کامل مسئله دست یافت کلیه این عملیات گام به گام را الگوریتم گویند.

بعبارتی هر دستورالعملی که مراحل انجام کاری را به زبان دقیق و با جزئیات کافی بیان نماید بطوریکه ترتیب مراحل و شرط خاتمه در آن کاملا مشخص شده باشد را الگوریتم گویند.

منظور از زبان دقیق ، آن است که الگوریتم دقیقا به همان صورتیکه مورد نظر نویسنده است اجرا گردد.

منظور از جزئیات کافی ، آن است که در طول اجرای الگوریتم عملیات ناشناخته پیش نیامده و باعث انحراف از مسیر و هدف اصلی نگردد.

منظور از ترتیب مراحل ، آن است که مراحل اجرای الگوریتم قدم به قدم و با رعایت تقدم وتاخر مشخص شده باشد.

منظور از شرط خاتمه ، پایان پذیر بودن الگوریتم می باشد و به هر حال الگوریتم باید در زمانی دلخواه و تحت شرط یا شروط داده شده خاتمه پذیرد.

الگوریتم خواندن کتاب

1. شروع
2. کتاب را باز کرده و از صفحه اول شروع می کنیم
3. یک خط از این صفحه کتاب را می خوانیم
4. اگر به پایان کتاب نرسیده ایم به مرحله 3 باز می گردیم
5. به صفحه بعد می رویم
6. اگر به پایان کتاب نرسیده ایم به مرحله 3 بازمی گردیم
7. پایان

تعریف متغیر : قسمتی از حافظه اصلی کامپیوتر است که مقدار آن می تواند تغییر کند

الگوریتم ماکزیمم (بزرگترین) دو عدد

روش اول

1. شروع
2. عدد اول را بگیر (دریافت کن) و در متغیر **A** قرار بده
3. عدد دوم را بگیر و در متغیر **B** قرار بده
4. اگر عددی که در متغیر **A** است بزرگتر از عددی که در متغیر **B** است (**A>B**) محتویا ت متغیر **A** را چاپ کن در غیر اینصورت محتویات متغیر **B** را چاپ کن(نمایش بده)
5. پایان

روش دوم با استفاده از متغیر کمکی بنام (**Max**)

1. شروع
2. عدد اول را بگیر (دریافت کن) و در متغیر **A** قرار بده
3. عدد دوم را بگیر و در متغیر **B** قرار بده
4. اگر **A>A** درغیراینصورت **B** است **Max B| Max**
5. مقدار **Max** رابعنوان بزرگترین عدد نمایش بده
6. پایان

الگوریتم تعیین میانگین سه عدد

1. شروع
2. عدد اول و دوم و سوم را بخوان و بترتیب در متغیرهای **A** و **B**و **C**قرار بده
3. مجموع سه متغیر را در متغیر دیگری |(**A+B+C**) بنام **Sum** قرار بده **(Sum**
4. متغیر **Sum** را تقسیم بر سه کن و در متغیر |(**Sum/3 Avg** قرار بده **(Avg**
5. مقدار متغیر **Avg** را بعنوان میانگین چاپ کن
6. پایان

الگوریتم مساحت و محیط مستطیل

1. شروع
2. طول و عرض مستطیل را بخوان و در **A** و **B**قرار بده
3. |(**A*B**حاصل عبارت **A*B** را در **C** قرار بده**C**)

4. حاصل عبارت 2 (A+B)* را در D قرار بده
5. مقادیر C و D را بعنوان مساحت و محیط مستطیل نمایش بده
6. پایان

الگوریتم جابجایی دو متغیر

1. شروع
2. دو عدد بخوان و در متغیرهای A و B قرار بده
3. مقدار A را در متغیر کمکی Temp قرار |(A بده Temp)
4. |(Bمقدار B را در A قرار بده A)
5. مقدار Temp را در B قرار بده
6. مقادیر A و Bرا نمایش بده
7. پایان

الگوریتم های حلقوی:

حلقه : (Loop) یکی از کاربردهای مهم کامپیوتر انجام اعمال تکراری می باشد . مراحلی از الگوریتم که چندین بار اجرای آنها تکرارمی گردد تشکیل یک حلقه را می دهند.

برای ساختن یک حلقه از یک متغیر کمکی استفاده می گردد ، این متغیر را قبل از شروع حلقه با یک مقدار اولیه آماده می سازیم و سپس معمولا در انتهای حلقه و قبل از بازگشت به ابتدای حلقه مقداری را به آن اضافه کرده و تحت یک شرط خاص به مراحل قبل پرش می نمائیم.

مقداری که قبل از شروع حلقه به متغیرحلقه داده می شود را مقدار اولیه یا شرط اولیه گویند.

مقداری که پس از یک بار اجرای مراحل حلقه به متغیر حلقه اضافه می شود را مقدار اضافه شونده می نامند.

مقدار یا شرطی را که پس از یک بار انجام مراحل حلقه با تعیین مقدار آن به ابتدای حلقه بازمی گردیم را مقدار یا شرط پایانی گویند.

الگوریتم نمایش اعداد 1 تا 100

1. شروع
2. |(1عدد 1 را در I بگذار I)
3. مقدار I را نمایش بده
4. |(I+1یک واحد به I اضافه کن I)
5. اگر I<=100 است به مرحله 3 برگرد
6. پایان

الگوریتم نمایش اعداد 100 تا 1

1. شروع
2. عدد 100 را |(1 در I بگذار)00 I
3. مقدار I را نمایش بده
4. |I-1یک واحد از I کم کن(I)
5. اگر I>=1 است به مرحله 3 برگرد
6. پایان

الگوریتم نمایش اعداد 500 تا 1000

1. شروع
2. |(500مقدار 500 را در I قرار بدهI)
3. مقدار I را نمایش بده
4. |(I+1یک واحد به I اضافه کن I)
5. اگر I<=1000 است به مرحله 3 برگرد
6. پایان

الگوریتم اعداد زوج 0 تا 200

راه اول

1. شروع

2. (0عدد 0 را در I قرار بده)
3. مقدار I را نمایش بده
4. (I+2 دو واحد به I اضافه کن)
5. اگر I<=200 است به مرحله 3 برگرد
6. پایان

راه دوم

1. شروع
2. (0عدد 0 را در I قرار بده)
3. اگر باقیمانده I تقسیم بر دو برابر 0 است مقدار I را نمایش بده
4. یک واحد به (I I+1 اضافه کن)
5. اگر I<=200 است به مرحله 3 برگرد
6. پایان

الگوریتم فاکتوریل یک عدد

توضیح : فاکتوریل یک عدد یعنی حاصل ضرب اعداد 1 تا آن عدد و آن را با ! مشخص می کنیم .

مثال : 0!=1,1!=1,2!=1*2=2,3!=1*2*3=6

1. شروع
2. (1مقدار 1 را در Fact قرار بده)
3. یک عدد بخوان و در N بگذار
4. (1مقدار 1 را در I قرار بده)
5. اگر I>N است به مرحله 8 برو
6. (Fact*I حاصل عبارت Fact * I را در Fact قرار بده)
7. یک (I+1 و به مرحله 5 برگرد واحد به I اضافه کن)
8. مقدار Fact را نمایش بده
9. پایان

برنامه نویسی چیست؟

تا به حال به این فکر کرده اید که برنامه هایی که با آن کار می کنید چگونه ساخته شده اند؟ بله همانطور که همه ی شما میدانید برنامه ها را برنامه نویسان به زبان برنامه نویسی می نویسند و کامپایلرها آن را به زبان **0**و**1** که زبان کامپیوتر است ترجمه می کنند. حال بیاییم بررسی کنیم که منظور از نوشتن برنامه چیست؟ منظور همان نوشتن است. البته در خیلی از زبان های شی گرا نصف برنامه را شما و بقیه را کامپیوتر می نویسد. اما چون ما هنوز به این مبحث نرسیده ایم فعلا به زبانی مثل پاسکال می پردازیم که زبانی به نسبت کاربردی و ساده است و برای شروع برنامه نویسی به نظر من بهترین زبان است. شما در برنامه نویسی به کامپیوتر دستور می دهید و کامپیوتر آن را اجرا می کند. اما در هنگام برنامه نویسی باید همواره به یک اصل توجه داشته باشید که کامپیوتر خر است. این بدین منظور است که وقتی می خواهید برنامه بنویسید ابتدا آن را به زبان خودتان بنویسید و سپس آن را به زبان خرها ترجمه کنید. البته هرچه زبانها پیشرفته تر میشوند کمتر نیاز است تا به زبان خری به کامپیوتر دستور دهید. برای مثال در زبان اسمبلی که ابتدایی ترین زبان است، برای اینکه برنامه تان یک چیز را در صفحه بنویسد باید سه خط (دستور) برنامه بنویسید ولی در برنامه ی پاسکال فقط کافی است یک دستور بنویسید. اما به هرحال باز هم درجه ی خریت در کامپیوتر زیاد است. برای مثال اگر ما بخواهیم به یک کامپیوتر که پا دارد (در زبانهای ابتدایی) دستور بدهیم راه برو باید بگوییم:

پای راستت را بردار_ پای راستت را جلو ببر_ پای راستت را زمین بگذار_ پای چپت را بردار_ پای چپت را جلو ببر_ پای چپت را زمین بگذار_ دوباره از مرحله ی اول شروع کن

البته این مثال کمی خر بودن کامپیوتر را زیادی بزرگ کرده است چون زبان های جدیدکمی کار را آسان کرده اند. ما به تاریخچه ی این زبان ها نمی پردازیم و فقط طرز کار با آنها را به شما

آموزش می دهیم. حال اگر دوست دارید برنامه هایتان را خودتان بنویسید و مسئله هایتان را با برنامه نویسی به کامپیوتر بدهید حل کند، دروس را دنبال کنید.

انواع زبان های برنامه نویسی:

زبان های نسل اول : زبان ماشین **010110** (کارت های پانچی)

زبان های نسل دوم : نزدیک به سخت افزار ولی بهتر از زبان ماشین مثل اسمبلی

زبان های نسل سوم : پاسکال، **C T3, PL1 , Ada , Basic , GL** ...

زبان های نسل چهارم : دلفی ، ویژوال بیسیک ، جاوا ، **Asp.net**

زبان های نسل پنجم : زبان های هوشمند

برای برنامه نویسی به سه چیز نیاز داریم : **1**) الگوریتم **2**) کامپیوتر **3**) نوشتن برنامه

مقدمه ای بر پاسکال:

این زبان که به افتخار بلز پاسکال دانشمند فرانسوی قرن هفدهم میلادی(مخترع اولین ماشین حساب مکانیکی)، پاسکال نامگذاری شده است، در اواخر سال **1960** و اوایل **1970** توسط پروفسور نیکلاس ویژت در انستیتو فنی فدرال سوئیس مطرح گردید.

پاسکال یکی از زبانهای سطح بالای موجود است و از جمله پر کاربردترین زبان هایی است که در حال حاظر برای تدریس پایه ای برنامه نویسی استفاده می گردد و امروزه در زمینه های مختلفی از جمله برنامه های گرافیکی ، حسابداری ، انبارداری و غیره بخوبی از آن استفاده می شود.

پاسکال زبان برنامه نویسی همه منظوره ای است که از زبان برنامه نویسی **Algol** گرفته شده است و دستورالعملهای آن بر اساس عبارات و برخی کلمات انگلیسی می باشد . همچنین پاسکال

دارای برخی خصوصیات ویژه است که بخصوص امکان طرح برنامه نویسی سازمان یافته را میسر می سازد یعنی برنامه نویسی با روش کاملا مرتب و منظم که وضوح ، کارایی و خالی از خطا بودن برنامه ها را افزایش می دهد.

یکی از مزایای زبان پاسکال این است که به ما اجازه می دهد جملات برنامه را با شباهت زیاد به زبان انگلیسی ، بنویسیم. مثلا دیگر لازم نیست عملی را که دستور **write** انجام می دهد را حفظ کنید زیرا با کمی آگاهی از زبان انگلیسی متوجه می شوید که دستور **write** برای نوشتن به کار می رود ولی زبان انگلیسی برای اینکه بدانید دستور **write** چگونه مینویسد واز چه قواعدی پیروی می کند کمکی به شما نمی کند و برای درک جزئیات آن نیاز به مطالعه و عمل کردن دارید.(دو اصلی که در تمامی قسمت ها و شاخه های کامپیوتر مهمترین عوامی منجر به یادگیری هستند)
بدین ترتیب شما با مطالعه ی یک برنامه به صورت خیلی سطحی می توانید بدون تسلط خاصی به برنامه نویسی پاسکال قسمت های نسبتا زیادی از برنامه را متوجه شوید.
(اگه هنوزم پاسکال رو ندارید از این آدرس دانلود کنید
(**http://www.freepascal.org/download.html**

مترجمهای زبان پاسکال (Pascal Compilers)

زبان پاسکال دارای مترجمهای گوناگونی که عبارتند از **Ms Pascal** : ، **Ansi Pascal**، **Microsoft Quick Pascal**، **Used Pascal** ، **Free Pascal** ، **Turbo Pascal** که **Turbo Pascal** یکی از پرکاربردترین مترجمهای زبان پاسکال می باشد . تمامی مترجمها جنبه های پاسکال استاندارد (**Ansi Pascal**) را برآورده می سازند.

نکاتی در مورد برنامه های پاسکال:

1. کامپایلر زبان پاسکال فرقی بین حروف کوچک و بزرگ نمی گذارد . یعنی **label** و **LABEL** از نظر پاسکال یکی می باشند.

2. در پایان هر خط از دستورات در زبان پاسکال یک علامت **;** (سمی کالون) گذاشته می شود که کار این علامت جدا نمودن یک دستور از دستور دیگر است.

3. در هر خط از برنامه می توان بیش از یک دستور نوشت.

4. مجموعه دستوراتی که با **Begin** شروع و با **End** تمام می شوند یک بلاک نام دارند . در تمام جاهایی که میتوانیم از دستورات تکی استفاده کنیم میتوانیم از مجموعه ای از دستورات تحت عنوان بلاک دستورات استفاده کنیم .

بعد از کلمه **Begin** سمی کالون نمی خواهد . بعد از **End** انتهای بلوک اصلی برنامه باید نقطه گذاشت (**End.**) و بعد از سایر **End** ها در برنامه معمولا سمی کالون می گذاریم **(End;)**

5. تمام متغیرهایی که در برنامه استفاده می شوند باید در قسمت **Var** تعریف شوند.

6. پاسکال دارای تعدادی کلمه کلیدی یا رزرو شده است که از این کلمات کلیدی نمی توان به عنوان متغیر استفاده نمود.

مشخصات برنامه مطلوب:

1. صحت و دقت : برنامه باید پاسخی درست و نتیجه ای مطلوب که از آن انتظار می رود ارائه نماید.

2. وضوح : اگر یک برنامه به طور واضح نوشته شده باشد باید برای یک برنامه نویس دیگر ممکن باشد که بدون سعی اضافی منطق برنامه را دنبال نماید. برای افزودن توضیحات به منظور افزایش خوانایی برنامه توضیحات آنها را بین {} و یا (* *) قرار می دهیم.(کامپایلر این خطوط را ترجمه نمی کند.)

3. سادگی و کارآیی : همیشه باید منطقی را در نوشتن برنامه هایتان برگزینید که بتوانید برنامه هایتان را به ساده ترین شکل ممکن بنویسید.

4. برنامه نویسی مدولی **: (Module)** در برنامه های ساده بدنه اصلی برنامه در داخل بلوک اصلی قرار می گرفت اما در برنامه های بزرگ و پیچیده تر اغلب ضرورت ایجاب می نماید در چند جای برنامه از گروه یکسانی دستورات استفاده شود و به هنگام روبرو شدن با مسائل پیچیده و بزرگ متوجه خواهید شد عموما ساده ترین کار ، تقسیم این قبیل مسائل به برنامه های کوچک تر است که هر کدام می توانند بطور مجزا حل شوند . هر یک از این قطعات و تکه برنامه ها ، یک زیر برنامه یا ماجول نامیده می شود . روش تقسیم برنامه به وظایفی مجزا را برنامه نویسی ساخت یافته می نامند . استفاده بهینه از زیربرنامه ها ضمن ایجاد نظم در برنامه و سهولت در اشکال زدائی سبب درک و فهم عملکرد برنامه می شود . هر زیربرنامه می تواند توسط برنامه اصلی یا یک زیربرنامه دیگر فراخوانی یا صدازده شود.

اجزای تشکیل دهنده ی یک برنامه در پاسکال :

در پاسکال برنامه ها از دو جزء کلی تشکیل می شوند که هر قسمت از خطوطی از کدها تشکیل شده اند:

قسمت اول : "قسمت اعلان برنامه" :

این قسمت بالاتر از تمام کد های دیگر است و فقط از قسمت های زیر تشکیل شده است:(این قسمت ها به ترتیب عبارتند از)

1) اعلان نام برنامه
2) اعلان نوع هایی غیر از نوع های استاندارد(**user type** ها)
3) اعلان ثابت ها
4) اعلان متغیر ها
5) اعلان برچسب ها
6) اعلان توابع
7) اعلان پروسیجر ها

تمام قسمت های قبل از **begin** اصلی برنامه جزء قسمت اعلان برنامه محسوب می شوند. تمامی متغیر ها ، توابع و ... و به طور کلی تمامی چیزهایی که در طول برنامه از آن ها استفاده می شود باید قبل از **begin** برنامه و در قسمت اعلان به برنامه معرفی شوند و در غیر اینصورت برنامه با خطا یا **Error** همراه خواهد بود.(به جز توابع و روال های استاندارد پاسکال که نیازی به تعریف ندارند و در هر قسمت برنامه می توان از آن ها استفاده کرد.)

قسمت دوم : "بدنه ی اصلی برنامه" :

این قسمت از بالا به یک **begin** و از استها به یک **end** و یک نقطه **(end.)** محدود میشود. دستورات اصلی برنامه در این قسمت قرار می گیرند و در واقع در این قسمت ما از اجزائی که در قسمت اعلان برنامه تعریف کردیم ، استفاده می کنیم.

شمای کلی یک برنامه در پاسکال :

```
Program prog_name ;
Type
TypeName= …… ;
Const
Constant=value;
Var
Variable_list : type ;
Function name (list of parameters) : function type;
.
.
.
Procedure name (parameters list) ;
.
.
.
Begin {main program}
دســـــتـــــورات
.
.
.
End.
```

اگر تا اینجای کار زیاد متوجّه اصطلاحات و کلمات به کار رفته نشدید(روال وثابت و ...) جای نگرانی نیست چون تمامی آن ها به طور مجزّا توضیح داده خواهد شد پس تا اینجا ما فقط از شما انتظار داریم بین این دو قسمت تفاوت قائل بشید همین وبس.!!!!!!!!!!!!!!!!!

در تمامی مدت زمانی که برنامه نویسی میکنید به خاطر داشته باشید که کامپیوتر چیزی جز یه خر اصلاح نژاد شده نیست و فقط باید به زبان خودش باهاش حرف بزنید (زبونش زبون خری نیست نترسید !!☺) و اگر نه یا **Error** میده و اصلا برنامه رو ترجمه نمی کنه یا برنامه را ترجمه میکند و مشکلاتی دیگر مثل خطای زمان اجرا و خطاهای منطقی و ... رخ میده پس باید از گرامر حکم فرما در هر زبان تبعیّت کرد و اگر نه برنامه مشکل خواهد داشت.

شرح قسمت اول در برنامه های پاسکال (قسمت اعلان) :

اصطلاحات مورد استفاده در این قسمت :

1) کلمات رزرو شده (reseved words)

کلماتی که توسط شرکت سازنده ، تعریف شده اند و در کامپایلر معنای خاصی دارند.(به همین دلیل نمی توان از این کلمات برای اهدافی غیر از اهداف تعیین شده استفاده کرد)

لیست کلمات رزرو شده در پاسکال در زیر آمده است:

and	exports	mod	shr
asm	file	nil	string
array	for	not	then
begin	function	object	to
case	goto	of	type
concat	if	or	unit
constructor	implementation	packed	until
destructor	in	procedure	uses
div	inherited	program	var
do	inline	record	with
downto	interface	repeat	while
else	label	set	xor

2) نوع (type)

به اسامی اطلاق میشود که بیانگر محدوده ی (مجموعه ی) خاصی از داده ها هستند.

3) برچسب (label)

کلمه ای رزرو شده برای علامت زدن یک قسمت از کد برای مراجعه های بعدی از طریق کد (در صورت لزوم)

4) شناسه ها (identifier)

شناسه در پاسکال برای نامگذاری ثابتها، تایپها، پروسیجروها، توابع، میدانهای یک رکورد، برنامه و همچنین یونیت مورد استفاده قرار می‌گیرد.دو نوع شناسه وجود دارد :

الف) **id** های استاندارد: این نوع **id** ها از قبل در زبان پاسکال تعریف شده‌اند و در برنامه‌ها، معنای خاصی دارند . (همون کلمات رزرو شده)

ب) **id** های غیراستاندارد: این نوع **id** ها بوسیله کاربر بطور مجزا تعریف می‌شوند و اصطلاحاً به آنها **userdefined** گفته می‌شود.(در تعریف شناسه ها قواعد نامگذاری باید رعایت شوند که در مورد بعدی توضیح داده میشود)

5) قواعد نامگذاری:

این قواعد تقریبا به طور یکسان در تمام زبان های برنامه نویسی وجود دارد در زمان تعریف هر جزئی در قسمت اعلان برنامه ها در پاسکال به خاصیتی با عنوان نام آن جزء مواجه میشویم (نام متغیر ، نام تابع و ...) تمام این نام ها باید از قواعد نامگذاری تبعیت کنند این قواعد عبارتند از:

الف) حروف **a_z** و **A_Z** مجاز است.

ب) ارقام **0** تا **9** مجاز است.

پ) کاراکتر اول نباید رقم باشد.

ت) نمی توان از کلمات ذخیره شده استفاده کرد.

ث) از جای خالی (**space**) بین کاراکترها نمی توان استفاده کرد.

ج) از علامت **under line** (_) میتوان بین کاراکتر ها استفاده کرد.

چ) طول یک شناسه می تواند طولانی باشد (نهایتا **63**) اگرچه بعضی از نسخه های پاسکال فقط اولین هشت کاراکتر را شناسایی می نمایند.

هر چیزی غیر از اینا غیر مجازه.

{اصطلاحات دیگه زیاد چیز خاصی نداره همینطور که جلو میریم در موردش میگم}

الف : نام برنامه و اعلان آن :

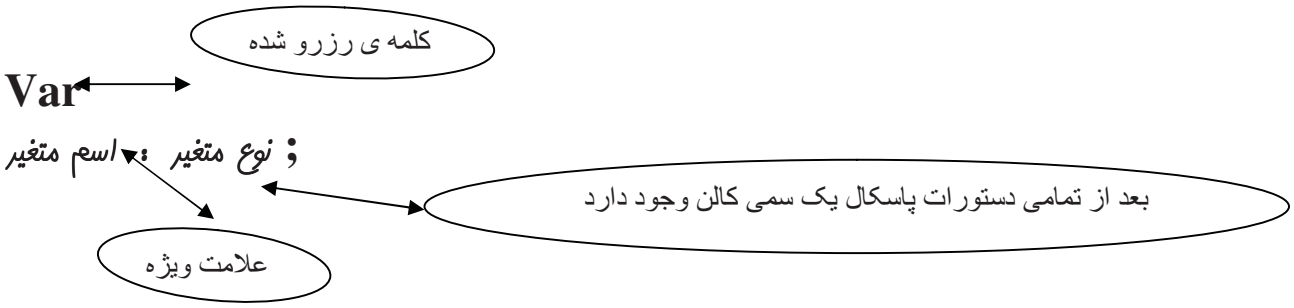
در سطر اول برنامه بعد از کلمه ی رزرو شده ی **program** نام دلخواه (بهتره اگر با معنی باشه) را به عنوان نام برنامه انتخاب میکنیم که البته در بعضی نسخه ها اختیاری می باشد .

مثال :

Program garmdareh;

ب : متغیرها و اعلان آنها :(variables)

متغیر یعنی قسمتی از حافظه که معمولا در طول برنامه مقادیر مختلفی را به خود می پذیرد در واقع متغیر ها مانند ظروفی با حجم مشخص شده توسط ما هستند برای نگهداری مقادیر. برای استفاده ازهر متغیر باید ابتدا آن را در قسمت اعلان ها اعلان کرد به این ترتیب که :



خب یه سوال :

اسم متغیر که گفتیم از قواعد نامگذاری تبعیت میکند و در اون مورد مشکلی نیست ولی "نوع متغیر" دیگه چیه؟

این همون حجم ظرفست که قرار بود ما تعیین کنیم (به تعریف دوباره یه نگاه بندازین) ما با تعیین کردن نوع به کامپیوتر اینو میگیم که خر خوبم اون قسمتی از حافظه که اسمش رو گذاشتی فلان

(همون اسمی که در هنگام تعریف به عنوان اسم متغیر و با استفاده از قواعد نامگذاری نوشتیم)
فقط میتونی از این شکل داده توش بریزی خب اون شکله که میگیم میشه نوع اون متغیر مثلا
معنی تعریف متغیر زیر رو با خود تعریف مقایسه کنید ببینید اونی که فکر میکنید هست ؟؟؟

```
Var
Test : char ;
```

خب حدس میزنید چی باشه ؟؟؟
Char رو میگم ؟؟؟!!!!!
اگه یه ذره زبان رو که گفته بودم بلد باشید شاید حدس زده باشید که همون کاراکتر خودمونه
(character)
این تعریف اینو میگه که کامپیوتر هر قسمت از حافظه **(Ram)** که خالیه و خودت صلاح میدونی رو اسمش رو بزار **Test** و از این به بعد اجازه بده برنامه ی من توی این قسمت از حافظه داده هایی از نوع کاراکتر رو بریزه در غیر این صورت (یعنی چیزی غیر از کاراکتر بود) **Error** بده با تشکر از زحمات جناب الاغ.
حالا بریم ببینیم کلا دست ما به عنوان برنامه نویس چقدر باز گذاشته شده و ما میتونیم از چه نوع های راحت و بی درد سر استفاده کنیم.
نوع های اصلی و پر کاربرد تر در پاسکال اینا هستند :

- نوع کاراکتری **(Char type)**

متغیری از این نوع میتواند یکی از کاراکتر های موجود در جدول اسکی را به عنوان داده نگهداری کند کلا **256** کاراکتر داریم و هر متغیر از نوع کاراکتر یک بایت فضا را در حافظه ی اصلی **(Ram)** اشغال میکند.
معمولا کاراکتر را بین ‘’ قرار میدهند. مثال:
هر یک از موارد زیر یک کاراکتر محسوب میشوند.

‘+’ ‘=’ ‘d’ ‘D’ ‘\’ ‘|’ ‘>’ ‘<’ ‘;’ ‘:’ ‘{’ ‘}’ ‘[’ ‘]’
‘*’ ‘!’ ‘.’ ‘?’ ‘x’ ‘a’ ‘ ’

- صحیح (integer)

یک متغیر از نوع صحیح میتواند مقادیر عددی جزء محدوده ی خودش رو به عنوان داده قبول کنه.
جدول زیرتمام اطلاعات مورد نیاز رو داره خوب بهش دقت کن.

نوع	محدوده	اندازه بر حسب بایت
Byte	از 0 تا 255	1
Shortint	از 128- تا 127	1
Integer	از 32768- تا 32767	2
Word	از 0 تا 65535	2
Longint	از 2147483648- تا 2147483647	4

مثال :

```
Var
I,j,k : byte ;
Test : shortint ;
```

نکته:
میشه همزمان چند متغیر رو از یک نوع تعریف کرد
برای این کار اونا رو با ویرگول از هم جدا میکنیم.

در مثال بالا متغیر های **I** و **j** و **k** میتونند مقادیر صحیح بین **0** تا **255** رو بگیرن و متغیر **test** هم مقادیر صحیح بین **128**- تا **127** رو به عنوان داده قبول میکنه.

- **اعشاري (حقيقي)**

یک متغیر از نوع اعشاری میتواند مقادیر عددی جزء محدوده ی خودش رو به عنوان داده قبول کنه.

جدول زیرتمام اطلاعات مورد نیاز رو داره خوب بهش دقت کن.

نوع	محدوده	تعداد ارقام معني دار	اندازه برحسب بایت
Real	2.2 e – 39 … 1.7 e 38	11-13	6
Single	1.5 e -45 … 3.4 e 37	7-8	4
Double	5.0 e -324 … 1.7 e 308	15-16	8
Extended	3.4 e -4932 … 1.1 e 4932	19-30	10
Comp	-9.2 e 18 … 9.2 e 18	19-30	8

- نوع رشته ای **(String type)**

متغیری از نوع رشته ای میتواند مجموعه ای از کاراکتر ها را یکجا نگهداری کند (مثل یک کلمه ، یک جمله و ...) بیشنه ی طول یک رشته **255** کاراکتر می باشد و معمولا رشته ها را بین "" قرار میدهند.

چون هر کاراکتر یک بایت فضا نیاز دارد یک رشته که مجموعه ای از کاراکتر هاست به تعداد کاراکترها فضا نیاز دارد (هر کاراکتر یک بایت) بعلاوه ی یک بایت اضافه که طول رشته را نگه میدارد پس:

1 + تعداد کاراکتر = فضای مورد استفاده برای یک رشته

(عدد به دست آمده بر حسب بایت)

- نوع منطقی **(Boolean type)**

متغیری از نوع بولین تنها میتواند یکی از مقادیر **True** یا **False** را بپذیرد و این نوع متغیر برای کنترل حلقه و کاربرد های زیاد دیگری در برنامه نویسی مورد استفاده قرار می گیرد.

یک متغیر از نوع بولین یک بایت حافظه را در مورد استفاده قرار می دهد.

. داده های زیربردی یا زیربازه :

اگر بخواهیم در محدوده خاصی از اعداد و یا حروف و یا در قلمرو یک مجموعه کار کنیم متغیرها را از نوع زیربردی تعریف می کنیم . مثال :

Var a : 1..100;

عملگر ها :

خب حال یه سوال پیش میآد (کی میدونه ؟؟؟)

حالا گیریم که ما متغیر رو تعریف کردیم هرفشم فهمیدیم که واسه ی نگهداری یه سری مقادیره خب این مقادیر رو چه جوری داخلش بریزیم؟

جواب غضنفر) اینم شد سوال خودت گفتی که مثل ظرفه خوب مثلا چه جوری تو لیوان آب میریزیم همون جوریه دیگه آره؟؟؟

Cyclops) ولی نـــــــه واسه ی این کار از یه عملگر استفاده میکنیم.

جواب غضنفر) خب عملگر دیگه چیه ؟؟؟ (از یه عملگر؟؟ مگه چند تا عملگر داریم؟؟؟)

Cyclops) آهان اینم یه سوال خوب.

تعریف خودمونی : عملگرها مثل یه جعبه ی جادویی میمونن که مثلا ما شعبه بازشیم ما چند تا وسیله میزاریم توش و بعد اون جعبه یه ورد روش میخونه (چه جالب ورد سر خوده) اینجوری اون چیزایی که گزاشته بودیم توش یه وسیله ی جدیدی هم به ما میده.

تعریف علمی : عملگر ها تو همه ی زبون ها وجود دارند و نمادهایی هستند که روی عملوند هاشون (همون وسایلی که میریزیم تو جعبه) کار خاصی انجام میدن.

(اینقدری که پیچوندمش سفت نیست ☺)

یه سری عملگر داریم که یه سری عملیات ریاضی رو روی عملوند ها انجام میدن به اونا میگیم عملگر های محاسباتی

اینم جدولش:

ردیف	عملگر	نام	مثال
1	+	جمع	x + y
3	-	تفريق و منهاي يكاني	x – y , -x
3	*	ضرب	x * y
4	/	تقسیم	x / y
5	div	تقسیم	a div b
6	mod	باقیمانده تقسیم	a mod b

نکته : در زبان پاسکال عملگر توان وجود ندارد و برای آن از ضربهای متوالی استفاده می شود.

مثلا تو جدول بالا ردیف یک "x" و"y" عملوند هستند و "+" عملگر.

غضنفر) خب این که نشد جواب اون سوال اولی بالاخره چه جوری یه مقدار رو توی یه متغیر میریزیم.

Cyclops) خوب شد گفتی خب یه عملگر هم واسه این کار داریم اونم عملگر انتساب یا دستور انتساب هست اصلا اگه این عملگر نبود ما محاسباتی که انجام میدادیم رو چی کار میکردیم؟؟ (رو کاغذ نگهداری میکردیم؟؟)

فرمت این دستور که یه جاهایی هم بهش دستور جایگزینی هم میگن اینجوریه:

مقدار =: اسم متغیر

مثلا: **x := x+1** این عبارت مقدار **x** رو یه دونه افزایش میده(در قسمت سمت راست)و اونو درون متغیر **x** میریزه (در قسمت سمت چپ) اگه مقدار متغیر **x** قبلا **10** بوده باشه بعد از اجرای دستور مقدارش برابر **11** خواهد بود.

یه سری عملگر داریم که عملوندهای خودشون رو با هم مقایسه میکنن به اونا میگیم عملگر های مقایسه ای:

مثال	نام	عملگر
x > y	بزرگتر	>
x >=y	بزرگتر مساوي	>=
x < y	کوچکتر	<
x <= y	کوچکتر مساوي	<=
x = y	مساوي بودن	=
x <> y	نامساوي	<>

نتیجه ی این عملگرها **true** و یا **false** است به همین دلیل معمولا جاهایی که باید شرطی بررسی بشه زیاد ازشون استفاده میشه(بعدا بیشتر توضیح میدم)

یه سری عملگر هم داریم که روی عملوند هایی از نوع **true** و **false** کار میکنند به اونا هم میگیم عملگر های بولی اینم جدولش:

مثال	نام	عملگر
a>y and y<x	و	And
x>y or y<x	یا	OR
Not (x)	نقیض	Not

یه سری عملگر هم داریم که به شما اجازه میدهند که روی بیت به بیت داده ها کار کنید به اینا هم میگن عملگرهای بیتی :

عملگر	نوع عمل
AND	و
OR	یا
XOR	یا انحصاري
NOT	نقیض
Shl	انتقال به سمت چپ
Shr	انتقال به سمت راست

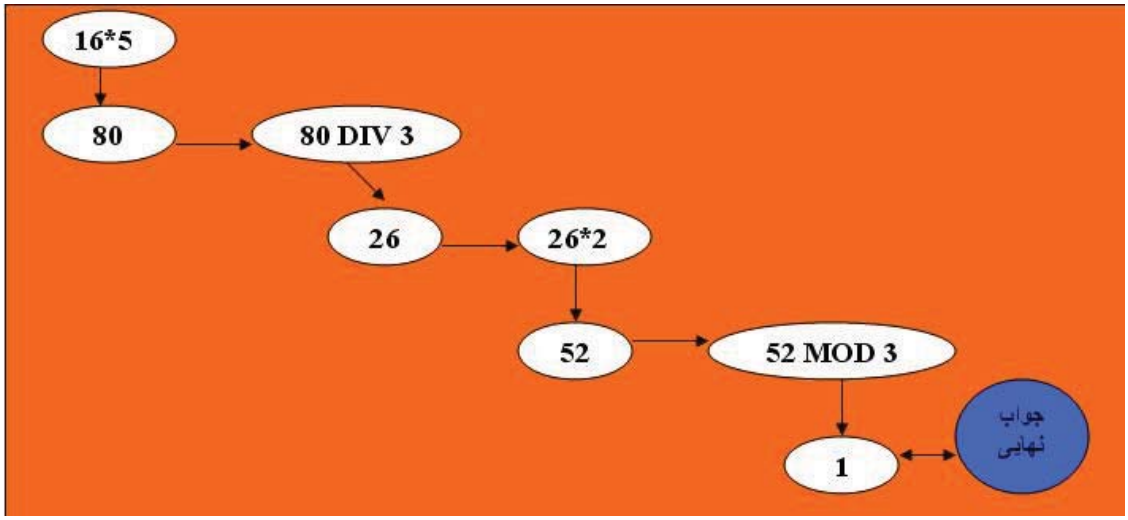
خب حالا جاهایی که کلی عملگر و عملوند بود چی کار کنیم مثلا اگه اینجوری بود:

16*5 DIV 3*2 MOD 3

همچین جاهایی از جدول زیر کمک می گیریم:

بالاترین تقدم ( )
Not
* div / mod
+ -
Shl shr
< <= >= >
= < >
And
XOR
OR

مثلا جواب همون عبارت طولانی بالا اینجوری قسمت بندی میشه:



که البته پرانتز میتونه تمام تقدم ها رو زیر پا بزاره یعنی بالاتریت تقدم با اونه.

ج : ثابت ها و اعلان آنها :

ثابت ها کمیت هایی هستند که مقدار آنها در طول اجرای برنامه تغییر نکند . برای معرفی ثابت ها در برنامه پاسکال از کلمه کلیدی **Const** استفاده می کنیم . مثال **Const P=3.1415;**

د : برچسب ها و اعلان آنها :

برچسبها شماره و یا نامهایی هستند که از آنها برای مشخص نمودن و یا نامگذاری خطی از برنامه استفاده می شود و به همراه دستور **goto** عمل می کنند . به کمک دستور **goto** می توان کنترل برنامه را از خطی به خط دیگر که دارای یک برچسب از پیش اعلان شده می باشد ، منتقل نمود . مثال **Label lb1,lb2;**

برای رجوع به **lb1** مینویسیم **goto lb1;** و اینطوری کنترل برنامه به جایی که ما **lb1** رو گذاشتیم برمیگرده.

دستورات ورودی و خروجی :

برای خواندن مقادیر از ورودی (معمولا کیبرد) و یا نوشتن مقادیری روی خروجی (معمولا مانیتور) از دستورات ورودی و خروجی استفاده می کنیم.

دستورات ورودی:

دستور Read

از این دستور برای خواندن داده ها از ورودی و اختصاص آنها به متغیری از نوع صحیح ، اعشاری و یا کاراکتری استفاده می شود ولی متغیر نمی تواند از نوع **Boolean** باشد.

توجه : هنگام ورود داده ها توسط دستور **Read** به ورودی ، باید بین مقادیر متغیرها توسط کلید **Space** فاصله ایجاد نمود..

دستور Readln

همانند دستور **Read** عمل می نماید با این تفاوت که دستور **Readln** باعث می شود که پس از انتقال داده به متغیر ، کنترل به خط بعد انتقال یابد درنتیجه دستور **Read** یا **Readln** بعدی ، داده ها را از خط بعدی خواهند خواند ، درصورتیکه دستور **Read** ، پس از انتقال داده به متغیر مربوطه باعث می گردد که دستور **Read** یا **Readln** بعدی ، داده ها را از همان خط جاری بخوانند.

دستورات خروجی :

دستور Write

این دستور برای نمایش یا چاپ کردن داده ها بر روی خروجی مورد استفاده قرار می گیرد . داده های خروجی می توانند ثابت های عددی ، مقادیر متغیرها ، عبارات و یا رشته ها باشند . داده ها می توانند از نوع صحیح ، حقیقی ، کاراکتری و یا منطقی (**Boolean**) باشند . رشته ها باید در بین دو علامت " قرار گیرند اگر داده ها بیش از یکی باشند باید با علامت , از یکدیگر مجزا گردند.

مثال:

Write (‘Input number : ‘ , a);

دستور **Writeln**

این دستور همانند دستور **Write** عمل می نماید با این تفاوت که بعد از اجرا کنترل را به ابتدای سطر بعد منتقل می کند در نتیجه موجب چاپ داده های بعدی ، در ابتدای سطر بعدی می گردد.

نکته : دستور **Writeln ;** باعث نمایش یک خط خالی در خروجی خواهد شد.

مثال : برنامه ای بنویسید که دو عدد را از ورودی دریافت و حاصلجمع آن دو را چاپ کند .

```
Program add;
Var
a,b:integer;
begin
num writeln(“enter 2”);
readln(a,b);
c:=a+b;
writeln('result is =',c);
end.
```

مثال : برنامه ای بنویسید که حقوق پایه وتعداد فرزندان یک کارگر را از ورودی دریافت و مبلغ حقوق وی را بر اساس فرمول زیر حساب کنید.

10*تعداد فرزندان + حقوق پایه =حقوق کل

```
Program test;
Var
Salary:longint;
Child:byte;
kole:integer;
Begin
Writeln('enter salary and number of child);
Readln(salary,child);
```

```
Kole := salary + child *10;
Writeln("kole is", kole);
END.
```

فرمت خروجی:

1) فرمت بندی اعداد صحیح : برای اعداد صحیح جلوی هر متغیر یک علامت : می گذاریم و بعد از آن تعداد خانه های موردنیاز یا طول میدان برای چاپ عدد را ذکر می کنیم.

مثال: a:=300; :

```
Writeln (a:4);
```

خروجی این دستورات چون عدد 3 رقمی می باشد و فرمت خروجی را 4 مشخص نموده ایم ابتدا یک فضای خالی و بعد چاپ 300 است . خروجی ⌗300

نکته ی1) اگر طول میدان از طول ارقام عدد صحیح بیشتر تعریف شود، عدد در منتهی‌الیه سمت راست میدان نوشته می‌شود.

نکته ی2) اگر طول میدان از طول ارقام عدد صحیح کمتر تعریف شود، طول میدان به اندازه تعداد ارقام در نظر گرفته می‌شود و طول میدان تعریف شده بی‌اثر خواهد بود.

2) فرمت بندی اعداد اعشاری : برای اعداد اعشاری از دو علامت : استفاده می شود . اولی برای قسمت صحیح عدد و دومی برای قسمت اعشاری.

مثال a:=32.112;

```
Write (a:2:2) ;
```

نتیجه این دستورات مقدار 32.11 خواهد بود پس بوسیله این دستور عدد را با دورقم اعشار نمایش داده ایم.

```
A:=35.118;
```

Write (a:2:2);

خروجی : **35.12**

نکته ی **1**) اگر طول میدان بزرگتر از تعداد ارقام عدد ذکر شود، عدد در منتهی الیه سمت راست میدان چاپ می‌شود.

نکته ی **2**) اگر فقط طول میدان ذکر شود، عدد به صورت نماد علمی در طول میدان مشخص شده چاپ می شود .

نکته ی **3**) از آنجائی که برای نمایش اعداد در نماد علمی حداقل **8** محل مورد نیاز است، لذا هنگامی که تنها طول میدان ذکر شده باشد، اگر از **8** رقم کمتر باشد، حداقل **8** رقم در نظر گرفته می‌شود.

نکته ی **4**) هنگامی که طول میدان همراه با تعداد ارقام بعد از ممیز ذکر شود، اگر طول میدان کوچکتر از مقدار عدد باشد، پاسکال تنها طول میدان را به اندازه‌ای که مورد نیاز است تصحیح کرده و آنرا برابر اندازه واقعی که عدد در آن قرار می‌گیرد، اصلاح می‌کند.

نکته ی **5**) اگر تعداد ارقام بعد از ممیز زیاد باشد و تعداد ارقام بعد از ممیز ذکر شده در طول میدان کمتر از تعداد ارقام اعشاری عدد باشد، تعداد ارقام اعشار مطابق درخواست برنامه‌نویس نشان داده خواهد شد و رقم آخر اعشار آن نسبت به عدد بعدی گرد می‌شود.

3) فرمت بندی رشته ها : مقادیر رشته ای در سمت راست طول میدان قرار می گیرند . بنابراین اگر فضایی که مقدار رشته ای باید قرار گیرد ، از طول رشته بیشتر باشد ، فضای خالی در سمت چپ میدان قرار می گیرد . اگر طول میدانی که برای رشته ذکر می شود ، کوچکتر از طول رشته باشد ، طول میدان نادیده گرفته خواهد شد .مثال: **write ('test':4);** :

خروجی **test** :

Write ('test':5);

خروجی **test** ⌗ :

مثالی دیگر :

X:= 3200 ;

```
A: = 12  ;
B: = 217  ;
Write ( X:3 , A:5  ,  B:5 )  ;
```

خروجی :

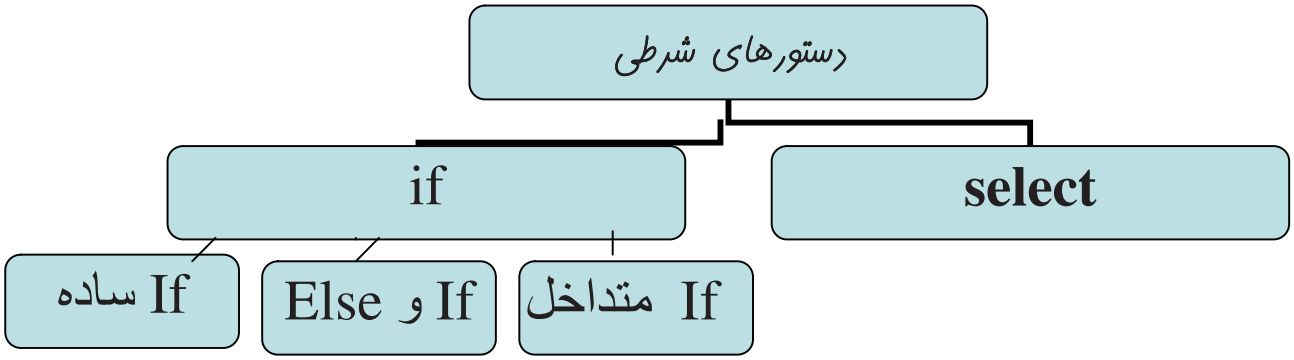
3200 □ □ □ 12 □ □ 217

نکته : در توربو پاسکال، کلیه موارد گفته شده در مورد اعداد صحیح برای رشته‌ها نیز صادق است.

حلقه ها و شروط :

شروط:

بطور کلی توسط اینگونه دستورات می‌توان بر حسب شرایط مختلف، تصمیمات متفاوتی را اتخاذ نمود و برحسب برقرار بودن یا نبودن شرایط دستورات متفاوتی را اجرا نمود.



انواع دستور if :

If – then :

در این نوع دستور شرطی اگر شرط خاصی تحقق یافته باشد، عمل یا اعمال خاصی انجام می‌شود. در غیر اینصورت برنامه روال عادی خود را طی می‌کند،در صورتی که شرط برقرار باشد ارزش منطقی **Ture** به خود می‌گیرد و اگر شرط برقرار نباشد، ارزش منطقی **False** به خود خواهد گرفت شکل کلی دستور **if** بصورت زیر می‌باشد:

If شرط یا شروط **Then**

دستـــورات ;

مثال : خروجی برنامه ی زیر چیست؟

```
Program if_exam ;
Var
Number: integer ;
Begin
Write ( 'Please enter Number: ' ) ;
Readln ( Number ) ;
if Number > 0 then
Write ( ' Number is positive ' ) ;
End.
```

خروجی برنامه بالا بصورت زیر است:

```
Please enter Number: 12
Number is positive
```

مثال: برنامه ای که یک عدد را از ورودی خوانده اگر آن عدد بزرگتر از **200** بود پیغامی چاپ کند .

```
Program If_then_Example;
Var a:integer;
Bagin
Readln (a);
If a>200 then
Writeln (' This number is big ' , a );
End.
```

If_Then_Else :

در این دستور ابتدا شرط بررسی می‌شود، در صورتی که شرط برقرار باشد، عمل یا اعمال خاصی را انجام می‌دهد و در صورتی که شرط برقرار نباشد، عمل یا اعمال بخصوص دیگری را انجام خواهد داد. اینگونه دستورات در واقع حالت توسعه یافته دستورات **if** می‌باشند.

در این دستور ابتدا شرط بررسی می‌شود، در صورتی که شرط برقرار باشد، عمل یا اعمال خاصی را انجام می‌دهد و در صورتی که شرط برقرار نباشد، عمل یا اعمال بخصوص دیگری را انجام خواهد داد. اینکونه دستورات در واقع حالت توسعه یافته دستورات **if** می‌باشند.

نکته مهم : قبل از **else** علامت **;** نمی خواهد.

شکل کامل دستور **if_Then_Else** :

If عبارت منطقی **Then**

Begin

دستور 1 **;**

دستور 2 **;**

.

.

.

End

Else

Begin

دستور 1 **;**

دستور 2 **;**

.

.

.

End ;

مثال : برنامه ای که یک عدد را از ورودی خوانده اگر آن عدد بزرگتر از **200** بود پیغامی چاپ کند.

```
Program If_then_Example;
Var a:integer;
```

Bagin
Readln (a);
If a>200 then
Writeln (‘ This number is big ‘ , a);
End.

مثال : برنامه ای بنویسید که تعداد فرزندان وحقوق پایه و رتبه یک کارگر را از ورودی دریافت و حقوق کل وی را بر اساس فرمول زیر حساب کند.

کسریها – مزایا +بیمه = حقوق کل

5*رتبه +**1000*** تعداد فرزند = مزایا

مالیات + بیمه =کسریها

بیشتر کمتر یا مساوی **2** تعداد فرزند

تعداد فرزندان **1000** بیمه

هر چیزی دیگر **20** تا **0** رتبه

حقوق پایه حقوق پایه***10/100** مالیات

```
begin
writeln(“enter salary and grade and number of
child”);
num,grade, readln(salary);
if num <=2 then;
bimeh:=100
else
bimeh :=num*500;
if (grade>=0) and (grade <=20) then
net := 10/100*salary
else
then 20< if grade
begin
net:=20/100*salary;
```

```
mazaya:=num*1000+grade*50;
kasry:=bimeh+net;
mazaya-kasry + kol :=salary;
end;
writeln(kol);
end.
```

مثال : در یک ترکیب شیمیایی 4 عنصر شرکت دارند مقدار مجاز برای تهیه ماده ای به نام asxd به این صورت است.

```
A 0 ~ 50
S 0.5 ~ 0.83
901 ~ X 81 ~ 92 or 824
D -100 ~ 100
```

با دریافت مقادیر a,s,x,d از ورودی به ما بگوید که آیا می توان این ماده را تولید کرد یا نه؟

```
Begin
Num:=0;
Writeln("please enter a s x d");
(Readln(a,s,x,d);
if (a > 0) and (a < 50) then
Num:=num+1;
if (s > 0.5) and (s < 0.83) then
Num:=num+1;
if ((x>81) and (x < 92)) or ((x > 824) and (x < 901))
then
Num:=num+1;
If (d > -100) and (d <100) then
Num:=num+1;
If num=4 then
Writeln("yes can")
Else
```

```
Writeln(“you can not”);
End.
```

مثال : برنامه ای بنویسید که با دریافت دو عدد بزرگترین آنها را چاپ کند.

```
Program test;
Var
a,b:integer;
 if a>b then
begin
writeln(a);
end
else
(writeln(b);
end.
```

مثال : برنامه ای بنویسید که با دریافت سه عدد به عنوان ضرایب **y=ax2+bx+c** معادله درجه دو را حل کند(این برنامه را با یک **If** بنویسید).

```
Var
A,b,c:integer;
D,x1,x2:real;
Begin
Writeln(“enter a,b,c”);
Readln(a,b,c);
D:=b*b-4*a*c;
if (d > =0)then
Begin
X1:=(-b+sqrt(d))/2*a);
X1:=(-b-sqrt(d))/2*a);
```

If های متداخل :

برای استفاده از **if** های متداخل دو راه وجود دارد من اینجا راحت ترین راه رو میگم که توی زبون های دیگه هم معادل داره و بیشتر توسط برنامه نویسا استفاده میشه.

If عبارت شرطی **1** **Then**

دستـــورات **1**

Else If عبارت شرطی **2** **Then**

دستـــورات **2**

Else If عبارت شرطی **3** **Then**

دستـــورات **3**

.

.

.

Else

دستـــورات

مثال : برنامه‌ای بنویسید که نمره دانشجوئی را از ورودی دریافت کرده، با توجه به مقدار نمره یکی از خروجی های زیر را نمایش دهد.

Grade	رتبه ی نمره
17 – 20	A
14 – 17	B
12 – 14	C
10 – 12	D
0 – 10	F

```
Var
      Grade   : Real  ;
Begin
```

```
Write ( ' please  enter  a  Real  Number : ' ) ;
Readln ( Grade ) ;
      if  Grade > = 17.0  Then
                Writeln ( ' Grade  is  A ' )
        Else  If  Grade > = 14.0  Then
                Writeln ( ' Grade  is  B ' )
        Else  If  Grade > = 12.0  Then
                Writeln ( ' Grade  is  C ' )
        Else  If  Grade > = 10
                Writeln ( ' Grade  is  D ' )
Else
                Writeln ( ' Grade  is  F ');
        Writeln ( ' Press  any  Key  ... '  : 30 ) ;
        Readln  ;
End . { end  of  program }
```

دستور **select** و کاربرد آن :

Case برای انتخاب یک حالت از چند گزینش میباشد ، در پاسکال علاوه بر دستور **If** دستور **Case** نیز وجود دارد . وقتی که گزینش بر پایه مقدار یک متغیر یا یک عبارت ساده انجام می شود ، استفاده از دستور **Case** بسیار مفید خواهد بود.

فرمت این دستور اینگونه می باشد :

Case عبارت **Of**

مقدار **1** : دستور **1** ;

مقدار **2** : دستور **2** ;

مقدار **3** : دستور **3** ;

.

.

.

```
Otherwise
        ;  دستور
End  ; { End  of  case }
```

نکته : **CASE** تنها دستوری است که **BEGIN** ندارد ولی **END** دارد.

مثال : برنامه‌ای بنویسید که دو عدد به همراه یک عملگر را از ورودی دریافت کرده، کار یک ماشین حساب ساده را شبیه‌سازی نماید.

```
Var
a , b: Real  ;
op: char ;
Begin
Write ( ' please  enter  two  Numbers: ' )  ;
Readln ( a , b )  ;
Write ( ' please  Enter  A  operator: ' )
Readeln ( op )  ;
Case  op  of
' + ':  Writeln ( ' Sum = ' , ( a + b ): 6: 2 )  ;
' - ':  Writeln ( ' Subtract = ' , ( a - b ): 6: 2 )  ;
' * ':  Writeln ( ' Multiple = ' , ( a * b ): 6: 2 )

' / ':  Writeln ( ' divide = ' , ( a / b ): 6: 2 )  ;
End; { End  of  Case  }
End. { end  of  program }
```

مثال : برنامه ای بنویسید که یک عدد را از ورودی دریافت (**0** یا **1** یا **2**) وتلفظ آنها را چاپ کند.

```
BEGIN
```

```
READLN(X);
CASE X OF
0 : WRITELN('ZERO:');
1 : WRITELN('ONE:') ;
2 : WRITELN('TWO:');
ELSE
WRITELN('ERROR')
END;
END.
```

حلقه های تکرار :

در اکثر مواقع میخواهیم دستور ویا دستور هایی رو به تعدادی بیش از یکبار انجام دهیم حالا یا این تعداد مشخصه یا تا وقتی یه شرطی برقرار بشه تکرار ادامه پیدا می کنه.

تقریبا میشه گفت یکی از مهمترین هنر های کامپیوتر هم همینه یعنی اجام کارهای تکراری به جای اینکه انسان اونا رو انجام بده و این وسط انسان فقط باید فکر کنه همین و بس.

معمولا حلقه ها از **2** قسمت مهم تشکیل شده اند :

الف) شرطی برای پایان دادن به حلقه (یا به نوعی شمارنده ای برای شمردن تعدادی که دستورات تکرار شده برای پایان دادن به موقع به حلقه)

ب) دستوراتی که باید تکرار شوند.

در زبان پاسکال **3** نوع حلقه داریم (بر خلاف **VB** که به **5** صورت حلقه در اختیار ما میزاره)

1) حلقه های با تعداد مشخص ← حلقه های **for**

فرمت این حلقه : (اندیس افزاینده)

For اندیس := مقدار اولیه **To** مقدار نهایی **Do**

دستور (بلوک دستورات) ;

فرمت این حلقه : (اندیس کاهنده)

Downto : در مواردی به کار میرود که مقدار اولیه از مقدار نهایی بیشتر باشد.

For اندیس := مقدار اولیه **DownTo** مقدار نهایی **Do**

دستور (بلوک دستورات) ;

فرمت این حلقه : (متداخل یا تو در تو)

For اندیس := مقدار اولیه **To** مقدار نهایی **Do**

For اندیس := مقدار اولیه **To** مقدار نهایی **Do**

دستور (بلوک دستورات) ;

و یا

For اندیس := مقدار اولیه **DownTo** مقدار نهایی **Do**

For اندیس := مقدار اولیه **DownTo** مقدار نهایی **Do**

دستور (بلوک دستورات) ;

و یا هر ترکیب دیگه ای (**to** و **downto** _ **downto** و **to** و ...)

2) حلقه های با تعداد تکرار نامشخص (شرط در ابتدا) ← حلقه های **while**

در این دستور تا زمانی که شرطی برقرار باشد اجرا میشود در این نوع حلقه برخلاف نوع بعد شرط حلقه در ابتدا بررسی میگردد یعنی این امکان وجود دارد که در صورت درست نبودن شرط حلقه حتی یک بار هم اجرا نشود.

فرمت این دستور :

While عبارت منطقی **Do**

دستور (بلوک دستورات) ;

3) حلقه های با تعداد تکرار نامشخص (شرط در انتها) ← حلقه های **Repeat_until**

در این دستور تا زمانی که شرطی برقرار نباشد اجرا میشود در این نوع حلقه شرط در انتها بررسی میشود بنابراین حلقه حداقل یک بار اجرا میشود.

فرمت این دستور :

Repeat
1 دستور ;
2 دستور ;
3 دستور ;
.
.
.
Until شرط یا شروط ;

نکته : دستور **Repeat** نیاز به بلوک ندارد و همراه **Until** ظاهر می‌شود.

کارگاه حل تمرین حلقه ها :

مثال های **For** :

مثال **1** :

```
Program garmdareh;
var
i:byte;
Begin
for i:=1 to 50 do
begin
writeln ('visit our site : www.garmdareh.tk');

end;

End.
```

که در این دستور ما ابتدا متغیر **i** را از نوع **byte** تعریف کردیم . سپس با استفاده از دستور **for** از کامپیوتر خواستیم تا **50** بار دستور **writeln** را انجام دهد که نتیجه آن **50** بار نوشته شدن متن **visit our site : www.garmdareh.tk** است .

مثال 2 :

برنامه ای بنویسید که 5 عدد را بگیرد و معدل را بدهد .

```
Program ibpersianblogcom
var
i:integer;
num:real;
average:real;
Begin
for i:=1 to 5 do
begin
writeln('please type your number');
read(num);
avarage:=avarage+num
end;
writeln (avarage/5);
End.
```

توجه : در اینجا در دستور writeln ما avarage/5 را بین دو علامت ' ' نگذاشتیم چون :

فقط وقتی از ' ' استفاده می کنیم که بخواهیم دقیقا" متن نوشته شده توسط ما چاپ شود یعنی در این مثال اگر 'avarage/5' را می نوشتیم خود avarage/5 چاپ می شد ولی اکنون که از ' ' استفاده نکرده ایم نتیجه ی تقسیم مقدار avarage به عدد 5 نمایش داده خواهد شد .

مثال 3 :

برنا مه ای بنویسید که یک عدد بگیرد و فاکتوریل آن را حساب کند و در آخر نتیجه را نمایش دهد .

```
program faktoriel;
var
fact,n,i : longint;

begin
writeln('enter an integer number');
```

```
readln(n);
fact:=1

for i:=1 to n do
    fact:=fact*i        ;
end;

writeln('factorial=',fact);
end.
```

الگوریتم هم که بسیار ساده است ابتدا عدد را از کاربر می گیرد بعد ۱ یا همان **fact** را به تعداد دفعات بزرگی عدد در آن عدد ضرب می کند که حاصل با راه حل ریاضی نیز هماهنگی دارد .

راه حل دوم با فرمت اندیس کاهنده ی دستور **For** :

```
Var
      i , n , Fact : integer ;
Begin
      Fact : = 1  ;
     Write ( ' please  enter  A  Number  ' )  ;
      Readln ( n )  ;
      For    I : = n      downto    1     do
               Fact : = Fact  *  i  ;
      Writeln ( ' Fact = ' , Fact )  ;
End .
```

مثال **4** :

برنامه‌ای بنویسید که **100** عدد از ورودی دریافت کرده، مجموع **100** عدد را محاسبه و چاپ نماید.

```
Var
      i , number , Sum: integer ;
Begin
       Writeln ( ' please  enter  100  Numbers: ' )  ;
       For    I: = 1     to     100     do
       Begin
              Readln ( number )  ;
              Sum:= Sum + number  ;
       End  ;
       Writeln ( ' Sum = ' , Sum )  ;
End. { end  of  program }
```

مثال 5 :

خروجی برنامه ی زیر چیست ؟

```
For   i: = 1     to    3    do
 Begin
       For   j: = 1     to    3    do
                Write ( ' pascal ': 8 )  ;
       Writeln  ;
 End  ;
```

خروجی :

```
Pascal      Pascal      Pascal مرحله اول  ( i = 1 )
 Pascal      Pascal      Pascal مرحله دوم  ( i = 2 )
 Pascal      Pascal      Pascal مرحله سوم  ( i = 3 )
```

مثال 6 : برنامه ای بنویسید که توان دوم اعداد **100** تا **200** را چاپ کند .

```
Begin
For i:=100 to 200 do
Begin
Writeln(i*i);
End;
End.
```

نکته : اجازه ندارید مقدار متغیر حلقه را در داخل بدنه عوض کنید.

مثال 7 : برنامه ای بنویسید که حاصلجمع اعداد **1..100** را چاپ کند .

```
Begin
Sum:=0;
For i:=1 to 10 do
Begin
Sum:=sum+i
End;
Write(sum);
End.
```

مثال 8 : برنامه ای بنویسید که حاصلجمع مضارب **5** بین **0** تا **100**را به دست آورد.

```
Begin
Sum:=0;
For I:=1 to 20 do
Begin
Sum:=sum+i*5;
End;
Writeln(sum);
End.
```

مثال **9** : برنامه ای بنویسید که دو عدد را از ورودی دریافت و اعداد ما بین این دو عدد را چاپ کند.

```
begin
Writeln('enter a,b');
Readln(a,b);
For I:= a to b do
Writeln(i);
End.
```

مثال **10** : برنامه ای بنویسید که با یک حلقه **for** اعداد بین **1..100** و **353..738** و **940..950** را چاپ کند

```
Begin
For i:=1 to 950 do
Begin
If ( i>=1) and (i<=100) then
Writeln(i);
If (i>353) and (i<=738) then
Writeln(i);
If (i>=940) and (i<=950) then
Writeln(i);
End;
End.
```

معادل برنامه بالا با استفاده از دستور **case** :

```
Begin
For i:=1 to 950 do
Case I of
1..100,353..738,940..950: writeln(i);
end;
end.
```

مثال **11** : برنامه ای بنویسید که مضارب **7** بین **1** تا **100** را چاپ کند .

```
Begin
For i:= 1 to 100 do
Begin
If I mod 7 = 0 then
Writeln(i);
End;
End.
```

مثال **12** : برنامه ای بنویسید که تعداد مضارب **5** و **7** را در بازه **846~231** چاپ کند .

```
Begin
K:=0;
C:=0;
For i:=846 downto 231 do
Begin
If I mod 5 = 0 then
C:=c+1;
If I mod 7 = 0 then
K:=k+1;
End;
Writeln('5=',c,'7=',k);
End;
End.
```

مثال **13** : برنامه ای بنویسید که حقوق پایه وتعداد فرزندان **100** کارگر را از ورودی دریافت و مبلغ حقوق آنها را بر اساس فرمول زیر حساب کند.

حقوق کل = حقوق پایه + تعداد فرزند ***1000**

```
Begin
For i:=1 to 100 do
Begin
Writeln('enter salary and number of children');
readln(salary,num);
kol := salary + num * 1000;
writeln(kol);
end;
end.
```

مثال **14** : برنامه ای بنویسید که **100** عدد را از ورودی دریافت و میانگین و حاصلجمع آنها را به ما بدهد .

```
Begin
Sum:=0;
For i:= 1 to 100 do
Begin
Writeln('enter num');
Readln(num);
Sum:=sum+num;
End;
Ave:=ave/100;
Writeln(sum,ave);
End.
```

مثال **15** : برنامه‌ای بنویسید که **n** عدد را از ورودی دریافت کرده، **n** جمله سری فیبوناچی را چاپ نماید.

```
Var
      F1 , F2 , F3 , N: integer ;
 Begin
      Write ( ' please  enter  A  Number: ' ) ;
       Readln ( N )  ;
       F1: = 0 ; F2: = 1  ;
                        Write ( F1: 5 , F2: 5 )  ;
      For    i: = 3    to    N    do
      Begin
            F3: = F1 + F2;
            If ( i  mod  10 ) = 0  Then
                  Writeln  ;
           Write ( F3: 5 )  ;
           F1: = F2  ;
           F2: = F3  ;
      End  ; { end  of  for }
End. { end  of  program }
```

مثال های **while** :

مثال **1** : برنامه ای بنویسید که بزرگترین مقسوم علیه دو عددی که از ورودی وارد شده اند را محاسبه کند.

```
Var
      m , n , r: integer ;
Begin
      Writeln ( ' Please  Enter  Two  Numbers ' )  ;

       Readln ( m, n ) ;
       While ( m   Mod   n )  < > 0   Do

        Begin
               r: = m   MOD   n  ;
               m: = n  ;
              n: = r  ;
End  ; { end  of  while }

       Writeln ( ' B. M. M = ' , n )  ;
End. { end  of  program }
```

مثال **2** : برنامه ای بنویسید که اعداد **1..100** را چاپ کند .

```
Begin
I:=1;
While i<=100 do
Begin
Write(i);
I:=i+1;
End;
```

این کار رو با حلقه های **for** هم می شد به راحتی انجام داد پس :

نکته : حلقه های **for** حالت خاص حلقه های **while** هستند درواقع تمام کارایی که میشه با **for** انجام داد میتونید راحت با **while** شبیه سازیش کنید.

مثال **3** : برنامه ای بنویسید که تعدادی اسم را از ورودی دریافت و به ما بگوید چند بار اسم **ali** تکرار شده است(آخرین اسم **end** است)

```
Var
S:integer;
St:string;
Begin
Read(st);
While st<> 'end' do
Begin
If st='ali' then
```

```
S:=s+1;
Read(st);
End;write(s);
End.
```

مثال **4** : برنامه ای بنویسید که تعدادی عدد که آخرین آنها صفر است را از ورودی دریافت و حاصلجمع آنها را چاپ کند ▪

```
Begin
Readln(num);
T:=1;
While num<>0 do
Begin
Sum:=sum+num;
Readln(num);
T:=t+1;
End;
Writeln('s=',s,'average',s/t);
End.
```

مثال **5** : برای دانش آموزان یک کلاس برنامه ای بنویسید که نام آنها به همراه تعداد نمرات و سپس نمرات آنها را به فرم زیر از ورودی دریافت کند.تعداد دانش آموزان کلاس نامعین است ولی آخرین نفر نامش **end** است .برنامه میبایستی پس از دریافت هر یک از اطلاعات پیامی به شکل زیر چاپ کند ▪

```
Average: Sum: Ali Name:
3
18
20
13.5

program garmdareh;
var
name:string[10];
ave,sum,nomreh:real;
num,i:integer;
begin
writeln('enter name ' );
readln(name);
```

```
while name <> 'end' do
begin
writeln('enter num:');
readln(num);
for i:=1 to num do
begin
writeln('enter nomreh: ');
readln(nomreh);
sum:=sum+nomreh;
end;
ave:=sum/num;
writeln('name:',name,'sum:',sum,'ave:',ave);
writeln('enter name ');
readln(name);
end;
end.
```

مثال **6** : میخواهیم برای برنامه های خود یک منو تهیه کنیم این منو شامل موارد زیر است:

1) load
2) save
3) edit
4) exit

please select a number :

```
program test;
var
num:integer;
begin
writeln(' 1) load');
writeln(' 2) save');
writeln(' 3) edit');
writeln(' 4) exit');
writeln(please select a number : ');
readln(num);
while num<>4 do
begin
case num of
1:writeln('loading');
2:writeln('saving');
3:writeln('editing');
end;
```

```
end;
end.
```

مثال **7** : برنامه ای بنویسید که دو عدد را از ورودی دریافت و طبق جدول زیر نسبت به مقدار آن واکنش نشان دهد .

3 2 1 a
واکنشB+...+3+2+1 1..b b..1

```
Begin
Writeln (' enter a,b :');
Readln(a,b);
Case a of
1:begin
for i:=1 to b do
write(i);
end;
2:begin
or I;=b downto 1 do
write(i);
end;
3:begin
sum:=0;
for i:=1 to b do
sum:=sum+I;
write(sum);
end;
end.
```

مثال های **repeat_until** :

مثال **1** : برنامه ای بنویسید که مجموع و میانگین تعدادی عدد صحیح مثبت را محاسبه نماید.

```
Var
      i , Sum , Number : integer ;
      ave: Real  ;
Begin
     Writeln ( ' please  enter  Numbers While  is  Not  Negative ' ) ;
      Sum: = 0  ;
      Ave: = 0  ;
      Repeat
            Readln ( Number )  ;
            Sum: = sum + Number  ;
```

```
            i: = i + 1  ;
        Until  number = 0  ;  { End  of  Repeat }
         i: = i - 1
        ave: = Sum / i  ;
       Writeln ( ' Sum = ' , Sum , ' average = ': 12 , ave: 7: 2 )  ;
end.{ end  of  program }
```

مثال **2** : برنامه‌ای بنویسید که یک عدد صحیح در مبنای ده را از ورودی دریافت کرده، به یک عدد در مبنای **2** ببرد.

```
Var
       Number , N , Power , R: integer  ;
 Begin
        Power: = 1  ;
        N: = 0  ;
        Write ( '  enter  A  Number: ' ) ;
        Readln ( Number )  ;
        Repeat
              R: = Number  MOD  2  ;
              Number: = Number  DIV  2  ;
              N: = N + Power * R  ;
              Power: = Power * 10  ;
        Until  Number < 2  ;
        N: = N + Number * Power  ;
        Writeln ( ' Number  In  Base  2 =  ' , N )  ;
End. { end  of  program }
```

خب یه زحمت هم واسه شما سعی کنید مثال های **while** رو با **repeat** حل کنید.

پروسیجر های کاربردی در حلقه ها :

پروسیجرها یا زیر روال‌ها قسمت‌های مستقلی از برنامه اصلی می‌باشند که به تنهایی اعمال خاصی را انجام داده و وظایف مستقل و بخصوصی بر عهده آنها گذاشته می‌شود. یک مزیت بزرگ پروسیجرها اینست که می توان بدون تعریف در قسمت های مختلف از آن ها کرد و از اصول برنامه‌نویسی ساخت یافته‌است.

Exit :

هدف: انتقال کنترل برنامه به خارج از بلوک فعلی

```
Procedure   Exit  ;
```

استفاده از این پروسیجر در هر بلوک از برنامه باعث می‌شود که کنترل برنامه بلافاصله به خارج از آن بلوک انتقال یابد.

break :

هرف: خاتمه دادن به اجرای یک حلقه

Procedure Break ;

استفاده از پروسیجر فوق باعث می‌شود که اجرای یک حلقه خاتمه یافته و کنترل برنامه به دستور العمل بعدی انتقال یابد.

Continue :

هرف: بازگشت به ابتدای حلقه

Procedure continue ;

وقتی این پروسیجر در حلقه ظاهر می‌شود کنترل برنامه به اول حلقه انتقال می‌یابد و دستورات بعد از پروسیجر اجرا نمی‌شوند.

دانشجو می نویسد :
(www.garmdareh1386.blogfa.com)
(garmdareh1386@yahoo.com)

آرایه ها (Arrays)

یه توصیه دوستانه: امیدوارم تا به اینجا کسی در باره انتخاب اسامی برای هر گونه متغیر صحبت کرده باشند اگر هم نه... دیگه نگران نباشید من می گم البته یه داستان کوچولو داره یه کم تحمل کنید آره آره من می دونم شما مشتاق درسین و تحمل دوری درس را حتی برای یک لحظه رو هم ندارید یکی بود یکی نبود...

من یک روز داشتم مبانی می خوندم و با توربوپاسکال برنامه می نوشتم که یک دفعه یک فرشته آسمانی که بر حسب اتفاق مهندس کامپیوتر بود(اونم از نوع نرم افزار) اومد پیشم وبه من گفت که خیلی افتضاح برنامه نویسی می کنی، اصلاً معلوم نیست که کدوم متغییر مربوط به کدام نوع است برای همین به من گفت که از این به بعد بهتر بنویس تا دفعه بعد حالتو نگیرم و تشویقت کنم و توصیه اش از این قرار بود که :

بابا اسم مُفته ، تو وَردار یه اسم برای متغیرت انتخاب کن یه متر، کسی بهت گیر نمی ده ولی به (همون کاراکترهایی که ما رسیدی متوجه می شی هر کدوم از **a,b,...** جاش وقتی وسط برنامه اِسما برای کدام نوع و متغییرِ ،و دیگه کوکبجه نمی مشخص می کنیم) گیرین.

آرایه و انواع آن

خانه های پشت سر هم از حافظه که همنوع بوده و توسط یک اسم معرفی می شوند، آرایه نام دارد. نحوه دسترسی به هر یک از اعضاء آرایه از طریق اندیس آرایه امکان پذیر است.

اکنون که پی بردید در بعضی از مسله ها ی برنامه نویسی ،به آرایه ها نیاز است، تعریف آنها را در پاسکال می آموزیم.آرایه ها در پاسکال به صورت زیر می توان تعریف کرد.

نام آرایه:**array**[نوع اندیس]**of** نوع عناصر آرایه

در این تعریف برای نامگذاری آرایه ،از قانون نامگذاری **array**کلمه کلیدی پاسکال است.

برای متغیرها استفاده می کنیم.

نکته: نوع اندیس می تواند کاراکتر،منطقی،شمارشی و زیر بازه باشد

نکته: انواع و نمی توانند،به عنوان لندیس انتخاب شوند

نکته: عناصر آرایه می تواند از هر نوعی باشد.

مثال:

```
var
        no:array[1..5] of integer;
        id:array[1..20] of byte;
        m:array[-4..3] of real;
```

آرایه های تعریف شده در بالا هر کدام دارای طول ها و نوع های **no,m** از نوع زیر بازه است.

مختلف می باشد.

m[-4]	m[-3]	m[-2]	m[-1]	m[0]	m[1]	m[2]	m[3]

در شکل اعدادی که در جلوی نام آرایه، در داخل[] آمده ،اندیس آرایه نام دارد.

آرایه ها را در پاسکال ،به روش دیگر نیز می توان تعریف کرد.در این روش ،ابتدا باید نوعی به نام نوع آرایه ایجاد کرد و سپس متغیری از نوع آرایه تعریف کرد.مانند مثال زیر:

type

نام نوع آرایه**=array[**نوع اندیس**]of** نوع عناصر آرایه

انتخاب نام نوع آرایه ،مثل انتخاب نام متغیرهاست. نوع اندیس مشخص می کند که اندیس آرایه چه مقادیری را کی پذیرد.عناصر آرایه می توانند هر یک از انواع موجود در پاسکال باشند.

مثال: تعریف نوع آرایه و تعریف متغیرهایی از نوع آرایه.

```
type
        intarray = array[1..5] of integer;
        rarray = array[-5..5] of real;
var
        y : intarray;
        x : rarray;
```

نکته: قبلاً در مورد ساختار برنامه پاسکال ،گفته شده که ابتدا ،ثوابت و سپس متغیرها تعریف می شوند. چنانچه در برنامه نوع جدیدی مثل زیر بازه یا نوع آرایه ایجاد کی شوند، ترتیب آنها به این صورت است: تعریف برچسبها، تعریف ثوابت، تعریف انواع جدید، تعریف متغیرها.

آرایه های یک بعدی

آرایه های یک بعدی به صورت زیر تعریف می شوند:

Name : array [1..length] of type;

اسم آرایه کلمه ذخیره شده طول آرایه کلمه ذخیره شده نوع آرایه

مقدار دهی آرایه ها مثل متغیر ها به دو صورت امکان پذیر است:

1.با استفاده از دستورات ورودی

2.مقداردهی در طول برنامه در سطر بعدی طریقه دسترسی به عناصر آرایه به صور زیر می باشد.

Name [index]

اسم آرایه اندیس آرایه

معمولاً اندیس آرایه که هنگام تعریف برنامه لحاظ می شود، از یک شروع می شود، ولی می تواند از یک هم شروع نشود. لذا در حالت کلی برای دسترسی به عناصر آرایه همراه اندیس ،محتویات خانه **i**ام را نمایش می دهد.

آرایه های دو بعدی

آرایه های دو بعدی متداولترین نوع آرایه ها هستند.آرایه های دو بعدی را می توان جدول یا ماتریس نیز می نامند. در هر آرایه دو بعدی، بعد اول را سطر و بعد دوم را ستون می نامیم.مانند نمونه ؛

Type

```
twodim = array[1..3,1..5] of   integer;
var
x:twodim;
```

x	ستون1	ستون2	ستون3	ستون4	ستون5
سطر1					
سطر2					
سطر3					

البته همانطور که قبلاً گفته شد ،می توان بدون تعریف نوع آرایه نیز، متغیر آرایه را تعریف کرد. در این صورت آرایه به صورت زیر تعریف می شود:

```
Var
X : array[1..3,1..5] of  integer;
```

آرایه های چند بعدی

آرایه هایی که تاکنون به کار گرفته شده اند، تعریف آنها طوری بود که برای دسترسی به عناصر آنها ، از یک اندیس استفاده می شد. این آرایه ها را آرایه های یک بعدی می نامیم .اما آرایه هایی وجود دارند که برای دستیابی به عناصر آنها ،باید از بیش از یک اندیس استفاده کرد .اینها را آرایه های چند بعدی می نامیم. به عنوان مثال، برای یافتن عددی در جدول ضرب، باید سطر و ستون آن عدد مشخص باشد. یعنی برای دسترسی به عنصری از جدول ضرب، باید سطر و ستون آن عدد مشخص باشد. به عبارت دیگر، برای دسترسی به عنصری از جدول ضرب از دو اندیس سطر و سطون استفاده می شود. نوع آرایه های چند بعدی را می توان به صورت زیر تعریف کرد:

Type

نام نوع آرایه = **array[1**اندیس**,2**اندیس**, ... ,n**اندیس**]** **of** نوع عناصر

پس از تعریف نوع آرایه چند بعدی باید متغیری از نوع آرایه تعریف و استفاده کرد:

Var نام آرایه : نوع آرایه

آرایه هایی که در اندیس داشته با شند، آرایه های دو بعدی، آرایه **n**بعد داشته باشندآرایه های هایی سه اندیس داشته باشند، آرایه های سه بعدی و آرایه هایی که **N**بعدی نام دارند.

جستجو در آرایه

یکی از اعمالی که به وفور در آرایه ها انجام می شود ،جستجو ی عنصری از آرایه است. فرض کنید ،معدل تعدادی از دانشجویان در آرایه قرار دارد و می خواهیم بیشترین معدل را پیدا کنیم. در این گونه موارد عناصر آرایه باید مورد جستجو قرار گیرند.

مثال: برنامه ای بنویسید که معدل ده دانشجو را در آرایه ای قرار می دهد و سپس بزرگترین معدل و محل وجود آن را در آرایه مشخص کرده، چاپ می کند.

حل)

```
program test;
        const n=10;
        type    arr=array[1..n] of real;
        var     ave:arr;
                i,p:integer;
Begin
        write('enter 10 average:');
        for     i:=1 to n do
        read(ave[i]);
        max := ave[1];
        p:=1;
        for     i:=2 to n do
                if      ave[i]>max    then
                begin
                        max:=ave[i];
                        p:=I;
                end;
        write('max average is:',max:5:2,'position is:',p);
End.
```

خروجی

```
Enter 10 average : 12.5 17.5 14 16.5 18.5 19 13 12 11 10
Max average is:19.00position is:6
```

مثال: برنامه ای بنویسید که معدل ده دانشجو را خوانده، در آرایه ای قرار می دهد و سپس مشخص می کند که چه معدلی بیش از همه تکرار شده است.اگر معدل تکراری وجود نداشت پیامی صادر کند.

حل:

```
Program test;
        const n=10;
        type    arr=array[1..n] of real;
        var     ave:arr;
                mean:real;
                i,j:integer;
```

```
        count,maxcount:integer;
Begin
    write('Enter 10 average:');
    for   i:=1  to  n  do
        read(ave[i]);
    maxcount:=0;
    for    i:=1 to n do
    begin
        count:=0;
        for    j:=1 to n do
            if ave[j]=ave[j] then
                count:=count+1;
        if  count>maxcount  then
        begin
            maxcount:=count;
            mean:=ave[i];
        end;
    end;
    if  maxcount>1 then
        write('Average',mean:5:2,'has',maxcount,'ineration.')
    else
        write('No ineration found!');
End.
```

خروجی یک:

```
Enter 10 average: 15.5 12.5 13 15.5 13 15.5 17 18 13.5 15.5
Average 15.5 has 4 ineration.
```

خروجی دو:

```
Enter 10 average: 11.5  15  13  11  14  15.5  17.5  14.5  16  18
No ineration found!
```

سوال: کار پردازه X چیست؟ {سال 85-86}

(a آرایه ای عمومی به طول n است)

```
Procedure x(n:integer);
var
    i,j:integer;
Begin
    for i:=1 to n-1 do
        for j:=i+1 to n do
            If a[i]>a[j] then
                swap(a[i],a[j]);
end;
```

الف) معکوس نمودن آرایه A

ب)مرتب نمودن آرایه A

ج)مقایسه آرایه عناصر A

د)موارد الف و ج

مرتب سازی آرایه ها

یکی دیگر از اعمال متداول در پاسکال که با آرایه انجام می شود ،مرتب سازی عناصر آرایه است. مرتب سازی به روشهای گوناگونی انجام می شود وقصد نداریم که آنها را در اینجا مورد بررسی قرار بدهیم و تنها به دو مورد از آنها اشاره می کنیم.

مر تب سازی ممکن است به طور صعودی یا نزولی انجام شود. در مرتب سازی صعودی ،عناصر از کوچک به بزرگ قرار می گیرند و مرتب سازی نزولی، عناصر، از بزرگ به کوچک قرار می گیرند. در زر نمونه ای از این دو شیوه مرتب سازی آمده اند:

مرتب سازی صعودی **4 9 11 19 25 30**

مرتب سازی نزولی **50 30 25 18 12 9**

مرتب سازی حبابی

روشهای مرتب سازی، بسیار فراوانند، اما در اینجا از روش مرتب سازی ساده ای به نام مرتب سازی حبابی استفاده شده است. در این روش دو عنصر متوالی با یکدیگر مقایسه می شوند ،چنانچه اولی از دومی بزرگتر باشد ،جای آنها در آرایه عوض می شود. وقتی یک بار تمام عناصر ،دو به دو با هم مقایسه شدند، بزرگترین عنصر به انتهای آرایه می رود. دفعه بعد که عناصر آرایه دو به دو با هم مقایسه می شوند ،دومین عدد از نظر بزرگی ،قبل از بزرگترین عنصر ،در انتهای آرایه قرار می گیرد.این روند تا مرتب سازی نهایی ادامه می یابد .برای روشن شدن موضوع ،مرتب شدن اعداد زیر را در نظر بگیرید:

عناصر اولیه **9 6 4 3**

مرحله اول **6 4 3 9**

مرحله دوم **4 3 6 9**

3 4 6 9

مرحله سوم

در مرحله اول نه با شش مقایسه شده جای آنها عوض شد.سپس نه با چهار مقایسه شد ،جای آنها عوض شد و سپس نه با سه هم مقایسه شد و جای آنها عوض شد و نه در انتهای آرایه قرار گرفت. این روند ادامه یافت تا عناصر آرایه مرتب شدند

مثال: برنامه ای بنویسید که ده عدد را خوانده ،آنها را به روش حبابی صعودی مرتب کند.

حل:

```
Program test;
        uses    crt;
        const  n=10;
        type
                arr=array[1..n] of integer;
        var     num:arr;
                temp,I,j:integer;
Begin
        clrscr;
        write('Enter 10 integer number');
        for i:=1   to   n  do
                read(num[i]);
        for   i:=n   downto   1  do
                for  j:=1  to   i-1   do
                        if  num[j]>num[j+1]  then
                        begin
                                temp  :=num[j];
                                num[j]:=num[j+1];
                                num[j+1]:=temp;
                        end;
        write('sort date is:');
        for  i:=1  to  n  do
                write(num[i]:3);
End.
```

خروجی:

Enter 10 integer number:15 11 14 13 16 19 10 21 9 4

Sort data is:4 9 10 11 13 14 15 16 19 21

جستجو در آرایه های مرتب

همانطور که دیدید،با جستجوی ترتیبی ،می توان عناصر مورد نظر را در آرایه جستجو کرد .در این روش ،مقدار مورد نظر ،به ترتیب با همه عناصر آرایه مقایسه می شود. اگر آرایه مرتب باشد ،روش دیگری برای جستجو در آن وجود دارد. این روش را باینری (دودویی) گویند.در این روش،وسط آرایه پیدا می شود و مقدار مورد نظر ،با عنصر وسط آرایه مقایسه می گردد. اگر با عنصر وسط آرایه برابر باشد،مقدار مورد نظر پیدا شده است و عمل جستجو خاتمه می یابد .چنانچه با عنصر وسط برابر نباشد، یا کوچکتر از آن است و یا بزرگتر از آن، در این صورت ،آرایه از وسط به دو بخش تقسیم می شود.اگر مقدار مورد جستجو کمتر از عنصر وسط باشد، جستجو در بخش سمت چپ انجام می شود وگرنه در بخش سمت راست انجام می گیرد. برای جستجو در بخشی از آرایه نیز، مثل آرایه کامل عمل می شود، و این روند ادامه می یابد و هر بار آرایه به دو بخش تقسیم می شود، تا مقدار مورد نظر پیدا شود یا حد پایینی آرایه از حد بالایی آرایه بزرگتر شود.

مثال:

برنامه ای بنویسید که در عدد را به طور صعودی مرتب هستند، از ورودی می خواند و سپس مقداری را از ورودی خوانده، آن را به روش باینری در آرایه جستجو می کند و پیام مناسبی صادر می نماید.

```
Program   test;
        uses   crt;
        const  n=10;
        type     arr=array[1..n] of  integer;
        var
                num : arr;
                mid,value,low,high,I : integer;
                find : Boolean;
Begin
        find:=false;
        clrscr;
        low:=1;
        high:=n;
        writeln('Enter 10 sorted integer number:');
        for    i:=1   to    n  do
```

```
            read(num[i]);
        writeln;
        write('Enter a number to search:');
        read(value);
        while(high>low) and (not find) do
        begin
            mid:=(low+high) diy 2;
            if num[mid]=value then
                find:=true
            else if value>num[mid] then
                low:=mid+1
            else
                high:=mid-1;
        end;{while}
        if find then
            write('The value',value,'exist in list.')
        else
            writeln('The value',value,'not exist in list.');
End.
```

خروجی یک:

```
Enter 10 sorted integer number:
1   3   5   7   8   19   20   23   25   27
Enter a number to search:40
The value 40 not in list.
```

خروجی دو:

```
Enter 10 sorted integer number;
12   34   37   39   45   56   67   78   97   100
Enter a number to search:56
The value 56 exist in list.
```

مثال:برنامه ای بنویسید که هر یک از کجوعه ها در آرایه قرار گرفتند و عناصر مجومه دوم مقایسه شدند.اگر عنصری در هر دو مجموعه وجود داشت، آن را به آرایه سوم منتقل کرده، سپس در خروجی چاپ شده اند.

(* بابا یه ذره هم خودتون فکر کنید فِرت پایین رو نگاه می کنید که چی بشه؟!؟!! هر چی خودتون فکر کنید بهتره *)

حل:

```
Program test;
        uses crt;
        const n=10;
        type   arr=array[1..n] of integer;
```

```
var
        x,y,m:arr;
        i,j,k:integer;
        find:Boolean;
Begin
        find:=false;
        k:=0;
        clrscr;
        write('Enter first array:');
        for i:=1 to n do
                read(x[i]);
        write(Enter second array:);
        for i:=1 to n do
                read(y[i]);
        for i:=1 to n do
                for j:=1 to n do
                        if x[i]=y[j] then
                        begin
                                k:=k+1;
                                m[k]:=x[i];
                                find:=true;
                        end;
        writeln;
        if find then
        begin
                writeln('Number in both array are:');
                for i:=1 to k do
                        write(m[i]);
        end
        else
                writeln('No elements are common!');
End.
```

خروجی:

```
Enter first array:2   6   10   8   4   16   18   24   20   36
Enter second array:2   19   17   27   48   16   8   20   18   6

Number in both array are:
2   6   8   16   18   20
```

مثال:برنامه ای بنویسید که عناصر دو آرایه پنج عنصری را که به طور صعودی مرتب هستند، از ورودی می خواند و آنها را طوری با هم ادغام می کند که به ترتیب صعودی عناصر حفظ شود.سپس عناصر مرتب شده ی نهایی به خروجی منتقل شوند

مثال:برنامه ای بنویسید که اعداد اول یک تا بیست را پیدا کرده در خروجی چاپ کند.این برنامه با استفاده از این تعریف نوشته شود:عددی اول است که بر هیچکدام از اعداد اول قبل از خودش قابل قسمت نباشد.

این دوتا هم هدیه من به شما ست چون حتماًخودتون می تونید دیگه حلشون کنیدسعی خودتونو بکنید حتماً می تونید یه کمی هم به خودتون متکی باشید حتماً می تونین... ای ول اعتـــــــــــــــماد به نفس همگی

اگه در این فصل کوتاهی شده مرا ببخشید ولی من سعی خودمو کردم با تشکر دانشجو

(www.garmdareh1386.blogfa.com)
(garmdareh1386@yahoo.com)

توابع و روال های کتابخانه ای :

ساختار تابع :

در کل هدف تابع اینست که متغیر یا متغیرهایی را بعنوان پارامتر از برنامه اصلی دریافت کرد.
عمل خاصی را روی پارامترها انجام داده و نتیجه را به برنامه اصلی برگرداند.
شکل کلی فراخوانی توابع در برنامه اصلی بصورت زیر می‌باشد:

ساختار تابع:

Function Name (parameters) : Type

کلمه ذخیره شده	اسم تابع	پارامترها	نوع تابع

توابعی برای اعداد صحیح و اعشاری :

✓ تابع Abs

هدف: باز گرداندن قدر مطلق (**absolute**) پارامتری که به آن ارسال می‌شود:

```
Function  Abs ( x: Integer ): Integer  ;
Function  Abs ( x: Real ): Real ;
```

این تابع یک عبارت از نوع حقیقي یا صحیح را بعنوان آراگون دریافت کرده سپس قدر مطلق آن را محاسبه و حاصل را برگرداند.

مثال :

```
Var
      f : Real  ;
      I : Integer  ;
 Begin
       F : = Abs ( - 191.15 ) ;
       I : = Abs ( - 171 )  ;
       Writeln ( ' f = '  ,  f : 8 : 2  ,  ' I = '  ,  I )  ;
  End .
```

خروجی حاصل :

F = 191.15   I = 171

✓ تابع Sin

هدف: باز گرداندن سینوس یک عدد از نوع اعشاري

شکل تابع: **Function Sin (X: Real): Real ;**

X یک عبارت یا عدد از نوع اعشاري بوده و حاصل مقدار سینوس X مي‌باشد.

مثال :

```
Var
   X : Real  ;
Begin
     X : = sin ( 10 )  ;
  Write ( ' sin ( 10 ) = '  ,  x :8 : 2 )  ;
End.
```

✓ تابع Cos

هدف: باز گرداندن کسینوس یک عدد از نوع اعشاري.

شکل تابع: Function cos (X: Real): Real ;

X یک عبارت با عدد از نوع اعشاري بوده و حاصل مقدار کسینوس X مي‌باشد.

مثال :

```
Var
   X : Real  ;
Begin
      X : = cos ( 10 )  ;
    Write ( ' cos ( 10 ) = ' ,  x : 8 : 2 )  ;
End .
```

✓ تابع ArcTan

هدف: بازگرداندن آرک تانژانت یک عدد از نوع اعشاري

شکل تابع: Function ArcTan (X: Real): Real ;

X یک عبارت یا عدد از نوع اعشاري بوده حاصل مقدار آرک تانژانت X مي‌باشد.

توجه: در صورتی که در توابع مثلثاتی زاویه از نوع درجه ارائه شود می‌توان با فرمول زیر معادل رادیان آن را محاسبه کرد:

```
Real = Dey * 3.14159 / 180
```

✓ تابع Exp

هدف: عدد نپر (e = 2.71828 ...) را به توان یک عدد می‌رساند.

شکل تابع: **Function Exp (X: Real): Real ;**

X عبارت یا متغیري از نوع اعشاري بوده و حاصل تابع نیز یک عدد اعشاري مي‌باشد

این تابع مقدار eبه توان X را محاسبه مي‌کند.

مثال :

```
Var
      X : Real ;
       i : integer ;
Begin
      For    i : = 1    to    10    do
      Begin
            X : = exp ( i ) ;
            Writeln (  x : 8 : 2  ) ;
      End ;
End .
```

✓ تابع frac

هدف: قسمت اعشاري يک عدد اعشاري را برمي‌گرداند.

شكل تابع: ; Function frac (X: Real): Real

X عددي از نوع اعشاري و حاصل تابع يک عدد اعشاري كه قسمت اعشاري عدد X است مي باشد و بعبارت ديگر اين تابع قسمت اعشاري عدد ورودي را به عنوان خروجي باز مي‌گرداند.

مثال :

```
Var
      Y , X : Real  ;
Begin
      X : = frac ( 24.769 )  ;
      Y : = frac ( - 12.75 )  ;
      Write ( ' x = ' ,  8 : 3 , ' y = ' , 8 : 2   )  ;
 End  .
```

خروجی :

```
X =    0.769    Y =    - 0.75
```

✓ تابع Int

هدف: قسمت صحيح يک عدد اعشاري را برمي‌گرداند.

شكل تابع: ; Function Int (X: Real): Real

X يک عبارت يا متغير از نوع اعشاري و خروجي تابع فيزيک عدد اعشاري است اين تابع مقدار صحيح يک عدد اعشاري در خروجي نشان مي‌دهد.

مثال :

```
Var
      y , x : Real  ;
Begin
       X : = Int ( 2.87 )  ;        { 2.0 }
       Y : = Int ( - 8.76 )  ;      { - 8.0 }

End .
```

✓ تابع Ln

هدف: محاسبه لگاریتم یـک عـدد اعـشاری در مبنای e.

شکل تابع: `; Function  Ln ( X: Real ): Real`

X یک عبارت یا متغیر ازنوع اعشاري بوده وحاصل تابع فیزیک عدداعشاري مي‌باشد.

مثال :

```
Var
      X : Real  ;
Begin
X : = Ln ( 2.87 )  ;        { x = 3.73767 }
Write ( ' x = ' , x : 10 : 5 )  ;
End .
```

تابع Odd

هدف: فرد بودن عدد صحیح را بررسی می کند.

شکل تابع: ; Function odd (X: logint): Booleam

X یک عبارت از نوع longint است و تابع ،فرد بودن عبارت را بررسي مي كند اگر مقدار خروجي تابع True باشد X فرد است و اگر مقدار خروجي تابع False باشد X فرد نيست.

مثال :

```
Var
      i : integer  ;
 Begin
      For    i : = 1    to    100    do
            If  odd ( i )  Then
           Writeln ( ' i = ' ,  j : 3 )  ;
End .
```

تابع Ord

هدف: غالباً براي پيدا كردن كد اسكي يک متغير كاراكتري بكار مي‌رود.

شکل تابع: ; Function Ord (x: char): Longint

عبارت x از نوع کارکتری را بعنوان پارامتر دریافت و کد اسکی آن را برمی گرداند. اگر x از نوع اسکالر باشد تابع بعنوان خروجی ترتیب قرار گرفتن x را در مجموعه ای که ابتدا به عنوان اسکالر اعلان شده، باز می‌گرداند.

مثال :

```
Var
     ch : char  ;
Begin
      For    ch : = ' A '    to    ' Z '    do
```

```
            Write ( ord ( ch ) : 5 ) ;
End .
```

خروجی :

کد اسکی ' A ' تا ' Z ' را درخروجی چاپ می‌کند

کد اسکی ' A ' عدد 65 می‌باشد.

✓ تابع pi

هدف: عدد پي را بر مي‌گرداند.

شکل تابع: `; Function pi: Real`

اين تابع براي بازگرداندن عددي پي (3.141592 ...) مورد استفاده قرار مي‌گيرد.

✓ تابع Pred

هدف: مقدار قبل مقدار پارامتر را بر می گرداند.

شکل تابع: `Function pred ( x ): < same type of parameter > ;`

پارامتر تابع مي تواند از هر نوع باشد و با توجه به نوع پارامتر تابع نياز از هم نوع مي‌باشد و خروجي تابع مقدار قبل از x مي‌باشد.

مثال :

```
{ ch = ' c ' }          Ch: = pred ( ' d ' ) ;
{ i = 14 }              i: = pred ( 15 ) ;
{ flag = false }        flag: = pred ( True ) ;
{ i = - 31 }            i: = pred ( - 30 ) ;
```

✓ تابع Random

هدف: براي توليد عدد تصادفي

شكل تابع:

- Function Random :Real ;

اگر تابع Random به شكل يک يعني بدون آرگومان مورد استفاده قرار گيرد يک عدد تساوي از نوع اعشاري بين صفر و يک توليد مي كند و اگر به شكل دو بكار رود باعث توليد يک عدد تصادفي از نوع word كه بزرگتر يا مساوي صفر و كوچكتر از x است خواهد شد.

✓ تابع Round

هدف: براي گرد كردن اعداد اعشاري بكار مي رود.

شكل تابع: Function Round (x: Real): Longint ;

X يک عبارت يامتغير اعشاري بوده وخروجي تابع يک عدد ازنوع Longint مي‌باشد كه نتيجه گرد كردن X مي‌باشد. اين تابع X را به نزديكترين مقدارصحيح گرد مي‌كند.

مثال :

```
i : = Round ( 57.4 )        { i = 57 }
```

```
i : = Round ( 59.5 )        { i = 60 }
i : = Round ( 12.7 )        { i = 13 }
i : = Round ( 12.25 )       { i = 12 }
i : = Round ( 17.75 )       { i = 18 }
i : = Round ( 17.45 )       { i = 18 }
i : = Round ( -2.5 )        { i = -3 }
```

✓ تابع sqr

هدف: براي محاسبه مجذور يک عدد صحيح يا اعشاري بكار مي‌رود.

شكل تابع:

```
Function  sqr ( x: Integer ): Integer  ;
Function  sqr ( x: Real ): Real  ;
```

X عبارتي يا متغييري از نوع صحيح يا اعشاري بوده و خروجي تابع فيزيک يک عدد صحيح يا اعشاري مي باشد اين تابع مجذور X را بعنوان خروجي بر مي‌گرداند.

مثال :

```
Var
      i : integer  ;
Begin
      For    i : = 1    to    10    do
                Writeln ( ' I ^ 2 = '  ,  sqr ( I ) )  ;
End .
```

✓ تابع sqrt

هدف: براي محاسبه جذر يک عدد بکار ميرود.

شکل تابع: `Function sqrt ( x: Real ): Real ;`

X يک عبارت از نوع اعشاري بوده و خروجي تابع فيزيک عدد اعشاري مي‌باشد اين تابع جذر X را بعنوان خروجي بر مي‌گرداند.

مثال :

```
Var
     Y , X : Real ;
Begin
     Y : = sqrt ( 64 ) ;        { y = 8 }
     Z : = sqrt ( 0.16 ) ;      { z = 0.4 }
     Write ( ' x = ' , x : 8 : 2 , ' y = ' , y : 8 : 2 ) ;
End .
```

✓ تابع succ

هدف: مقدار بعد از مقدار فعلي را برمي گرداند.

شکل تابع: `Function succ ( x ): same type of parameters ;`

X يک عبارت از نوع صحيح، اولين و غيره بوده و خروجي تابع نيز از همان نوع X مي باشد اين تابع مقدار بعد از X را بعنوان خروجي بر مي‌گرداند.

مثال :

```
ch : = suce ( ' a ' )        { ch = ' b ' }

ch : = suce ( ' A ' )        { ch = ' B ' }

 i : = suce ( 15 )           { i = 16 }

flag : = suce ( false )      { flag = True }

flag : = suce ( True )       { تعریف نشده }
```

✓ تابع Trunc

هدف: قسمت صحيح يک عدد اعشار را بر مي‌گرداند.

شکل تابع: `Function Trunc ( x: Real ): Longint ;`

Xيک عبارت يا متغير از نوع اعشاري بوده و خروجي تابع يک عدد از نوع Longintمي‌باشد. اين تابع قسمت صحيح عدد اعشاري X را بعنوان خروجي بر مي‌گرداند.

مثال :

```
Var
      x , i , j : Longint  ;
Begin
      i : = trunc ( 12.5 )  ;       { i = 12 }
      j : = trunc ( 12.4 )  ;       { i = 12 }
      k : = trunc ( 13.5 )  ;       { k = 13 }
      Writeln ( ' I = ' , i : 5  , ' j = ' , j : 5 , ' k = ' , k : 5 )  ;
 End .
```

توابع از نوع کاراکتری

در این بخش توابعی را بررسی می کنیم که خروجی آنها از نوع کاراکتری باشد.

تابع chr

هدف: معادل كاراكتري يک كد اسكي را بر مي‌گرداند.

شکل تابع: **Function chr (X: Byte): char ;**

X یک عبارت يا متغير از نوع بايت بوده و خروجي تابع يک كاركتر مي‌باشد.

اين تابع كد اسكي را دريافت كرده و معادل كاركتري آن را بر مي‌گرداند.

مثال :

```
Var
      i : Byte ;
Begin
      For    i : = 65    to    91    do
             Writeln ( chr ( I ) ) ;
End .
```

خروجی :

برنامه بالا حروف **A** تا **Z** را در خروجی چاپ می‌کند.

✓ تابع Upcase

هدف: برای تبدیل یک کاراکتر به حرف بزرگتر بکار می‌رود.

شکل تابع: **Function Upcase (ch: char): char ;**

ch یک عبارت یا متغیر از نوع کاراکتر بوده و خروجي تابع فیزیک کاراکتري باشد این تابع حرف کوچک را به حرف بزرگ تبدیل کرده و بعنوان خروجي حروف بزرگ را بر مي‌گرداند.

مثال :

```
Var
        Sent: array [ 1.. 80 ] of  char  ;
        i , N: Byte  ;
Begin
       Write ( ' enter  Numbers: ' )  ;
        Readln ( N )  ;
        For   i: = 1   to   N   do
              Read ( Sen [ i ] )  ;
        Writeln  ;
        For   i: = 1   to   N   do
              If ( sen [ i ] > = ' a ' )  and  ( sen [ i ] < = ' z ' ) Then  ;
                    sen [ i ]: =  Upcase ( sen [ i ] )  ;
        Writeln ( ' Out pot  of  program ' )  ;
         For   i: = 1   to   N   do
              Writeln ( sen [ i ] )  ;
End. { End  of  program }
```

روال‌های استاندارد :

در بخش های قبل دیدید که در توابع پارامترها به تابع ارسال می شود و تابع نیز مقداری را بعنوان خروجی برمی‌گرداند روالها نیز مشابه توابع عمل می‌کنند با این تفاوت که خروجی روالها از طریق پارامتر برمی‌گردانده می‌شود یا اعلان به سیستم عامل می‌باشد. بنابراین روالها بدون نوع هستند.

شکل کلی روال‌ها بصورت زیر می‌باشد:

Procedure Name (Parameturs) ;

کلمه ذخیره شده اسم روال پارامترهای روال

✓ روال Dec

هدف: یک یا چند واحد از پارامتر ارسالي كم مي كند.

شکل روال:

Procedure Dec (Var X: longint) ;
Procedure Dec (Var X: longint , N: longint; (

Xیک متغیر از نوع longint و بصورت متغيري مي‌باشد اين روال يک واحد از پارامتر ارسالي كم مي‌كند.

مثال :

```
Var
        N : integer
Begin
        N : = 1201 ;
        Dec ( N ) ;       { N = 1200 }
        Writeln ( N ) ;
        Dec ( N , 200 ) ;       { W = 1000 }
        Writeln ( N ) ;
End .
```

✓ روال Exit

هدف: كنترل برنامه را به خارج از بلوک جاري منتقل مي كند.

شكل روال: ; Procedure Exit

اين روال باعث مي شود كه كنترل برنامه از بلوک جاري خارج شود. اگر اين روال در برنامه اصلي بكار رود باعث خروج از برنامه مي‌شود. و اگر در يک روال يا تابع بكار رود باعث خروج از روال يا تابع شده و كنترل برنامه به برنامه اصلي منتقل مي‌شود.

مثال :

```
Var
i , j : integer  ;
Begin
        i : = 100 ;
        j : = 20  ;
        Dec ( i , j )  ;
        Write ( i )  ;
        Exit   ;
End .
```

✓ روال Halt

هـدف: خاتمـه دادن بـه اجـراي برنامه

شكل روال: ; Procedure Halt

اين روال باعث خاتمه اجراي برنامه شده و كنترل برنامه به سيستم عامل بر مي‌گردد.

✓ روال Inc

هدف: اضافه کردن یک یا چند واحد به یک متغیر

شکل تابع:

Procedure Inc (Var X: longint)

Procedure Inc (Var X: longint , N: longint) ;

X یک عبارت یا متغیر از نوع Longint مي باشد اين روال به مقدار متغير x يک يا چند واحد اضافه مي‌کند.

مثال :

```
Var
     N , i , j : integer  ;
Begin
      i : = 100 ;
      j : = 200  ;
      N : = 10  ;
      inc ( i )  ;      { i = 101 }
      inc ( j , N )  ;      { j = 210 }
      Writeln (' i = ' , i , ' j = ' , j )  ;
 End .
```

✓ روال Randomize

هدف: باعث تغير نحوه توليد اعداد تصادفي مي شود.

شکل روال: **Procedure Randomize ;**

وقتي در برنامه از تابع Random استفاده مي‌کنيم اعداد تصادفي توليد شده در اجراهاي مختلف يکسان مي باشد براي جلوگيري از اين وضعيت قبل از استفاده از تابع Random روال Rondomize را بکار مي‌بريم. تا باعث توليد اعداد تصادفي متفاوت در اجراهاي مختلف گردد.

مثال :

```
Var
      i : word  ;
Begin
       Randomize;
       For    i : = 1    to    10    do
             Writeln ( Random ( 50 ) )  ;
End .
```

کارگاه حل تمرین توابع و روال های کتابخانه ای (تستی) :

تست 1) دستور **pred('c')** چه کاری انجام می دهد ؟

الف : علامت مربوط به کد اسکی **c**

ب : علامت قبل از حرف **c** را برمیگرداند

ج : علامت بعد از حرف **c** را بر می گرداند

د : کد اسکی علامت **c** را برمی گرداند

تست 2) مقدار عبارت زیر چند است؟

5+2/8*3-sqrt(8/2)+2 mod Round(6 div 4)

الف : **-0.375**

ب : **3.75**

ج : **4.75**

د : **5.25**

تست 3) کد زیر چه چیز را روی مانیتور چاپ میکند؟

Writeln (sqrt(abs(-4)));

الف : **-4**

ب : **2**

ج : **-2**

د : برنامه خطا دارد و چیزی نمی نویسد.

تست 4) نوع خروجی و ورودی کدام تابع زیر با هم برابر است؟

الف : **sqr**

ب : **sqrt**

ج : **odd**

د : **trunk**

تست **5**) خروجی دستور زیر را از بین گزینه های ارائه شده پیدا کنید ؟ **(e ~ 2)**

```
Writeln(trunc(exp(3*ln(2))));
```

الف : **6**

ب : **5**

ج : **8**

د : **9**

تست **6**) نوع پارامتر ورودی و خروجی تابع **odd(x)** چیست ؟

الف : اعشاری ، منطقی

ب : اعشاری ، اعشاری

ج : صحیح ، منطقی

د : صحیح ، صحیح

تست **7**) خروجی این تکه برنامه چیست ؟ **x** و **y** متغیرهای صحیح و **k** متغیر کاراکتری می باشد.

```
For x:=3 to 4 do
    Begin
        Write(x-1);
        For k:='c' to 'd' do
        Begin
            For y:=5 to 6 do;
                Write(y);
            Write(pred(k));
        End;
    End;
```

الف : **356b56c456b56c**

ب : **26b6c36b6c**

ج : **256b56c356b56c**

د : برنامه خطا دارد .

<< پاسخ ها و راه حل ها >>

1) پاسخ صحیح گزینه ی **"ب"**

2) پاسخ صحیح گزینه ی **"ب"**

بر اساس جدول تقدم عملگرها داریم :

5+2/8*3-sqrt(8/2)+2 mod round(6 div 4)=
5+2/8*3-sqrt(4)+2 mod round(6 div 4)=
5+2/8*3-2+2 mod round(6 div 4)=
5+2/8*3-2+2 mod round(1)=
5+2/8*3-2+2 mod 1=
5+0.25*3-2+2 mod 1=
5+0.75-2+2 mod 1=
5+0.75-2+0=
5.75-2 = 3.75

3) پاسخ صحیح گزینه ی **"ب"**

Writeln(sqrt(abs(-4))) → writeln(sqrt(4)) → writeln(2) →2

4) پاسخ صحیح گزینه ی **"الف"**

5) پاسخ صحیح گزینه ی **"ج"**

با فرض اینکه مقداری تقریبی **e** برابر **2** باشد داریم :

Writeln(trunc(exp(3*ln(2)))) → Writeln(trunc(exp(3*1))) →
Writeln(trunc(8)) → writeln(8) → 8

6) پاسخ صحیح گزینه ی **"ج"**

7) پاسخ صحیح گزینه ی **"د"**

k	y	خروجی
c → 6		
d → 6		**26b6c36b6c**

کاراکترها و رشته ها :

کاراکتر ها :

همون طوری که تو قسمت تعریف متغیر ها گفتم . . .

یـــادآوری :

کاراکتر یکی از انواع از پیش تعریف شده ی پاسکال هست و یک متغیر از این نوع میتواند یکی از کاراکتر های موجود در جدول اسکی را به عنوان داده نگهداری کند کلا **256** کاراکتر داریم و هر متغیر از نوع کاراکتر یک بایت فضا را در حافظه ی اصلی (**Ram**) اشغال میکند.

معمولا کاراکتر را بین ‘’ قرار میدهند. مثال:

هر یک از موارد زیر یک کاراکتر محسوب میشوند.

‘+’ ‘=’ ‘d’ ‘D’ ‘\’ ‘|’ ‘>’ ‘<’ ‘;’ ‘:’ ‘{’ ‘}’ ‘[’ ‘]’

‘a’ ‘x’ ‘?’ ‘.’ ‘!’ ‘*’ ‘ ’ ‘ ’

در مورد کار با مغیر های کاراکتری همیشه باشد چند برابر مواقع دیگه دقت کنید چون همون طور که تو آخرین مثال بالا می بینید (‘’) حتی یه فضای خالی یا یک **enter** هم یه کاراکتر محسوب میشه

مثلا به مثال زیر نگاه کنید بگید توی هر متغر بعد از اجرای دستور چه مقادیری ذخیره شده ؟؟؟؟؟؟؟؟؟

مثال :

```
Var
      i,j:integer;
      ch1,ch2:char;
begin
      writeln(‘please enter 2 integer variable & 2
character now , thank you user !!!!’);
      read (ch1,ch2,i,j);
.
.
.
End.
```

خب برنامه رو اجرا کنید حالا برنامه از شما میخواد مقادیر رو وارد کنید خب مثلا وارد میکنیم :

Gp 4 13

خب کاملا مشخصه که بعد از اجرای دستور مقدار متغیر ها اینجوری هستند :

ch1=G , ch2=p , i=4 , j=13

خب اینجا نیازی به دقت نبود چون از نکته ای که یه ذره پایین تر میگم استفاده کردیم ولی حالا شما تو کد برنامه جای **read(ch1,ch2,I,j)** یه چیز دیگه بنویسید مثلا

read(i,j,ch1,ch2) خب دیگه درست از آب در نمی آد چون اگه مثل قبل متغیر ها توسط کاربر مقدار دهی بشن مقادیر غلط میشه چون کد اسکی **G** و **p** به ترتیب توی **I** و **j** ذخیره میشه و همچنین کاراکتر های معادل کد اسکی **4** و **13** بترتیب تو متغیر های **ch1** و **ch2** قرار میگیرن که ما همچین چیزی رو نمیخواستیم پـــــــــــــس :

نکته : همیشه برای خوندن متغیر های کاراکتری از ورودی یا اونا رو با دستورات **read/readln** جدا میخونیم یا اونا رو قبل از متغیر های دیگه تو دستورات **read/readln** مشترک می خونیم (مثل مثال بالا)

مقایسه ی کاراکتر ها با هم :

تمام کاراکتر ها را میتوان با یکدیگر مقایسه کرد این مقایسه بر حسب کد اسکی کاراکتر ها انجام می شود بدین ترتیب که هر کاراکتری که عدد کد اسکی آن بزرگتر باشد بزرگتر است و بلعکس.

(برای دیدن کد اسکی کاراکتر ها به ضمیمه ی آخر همین کتاب مراجعه کنید)

رشته ها :

آرایه ای از کاراکتر ها را یک رشته می نامیم (دقیقا میتوان یک رشته را با آرایه ای از کاراکتر ها شبیه سازی کرد به این صورت که **name : array [1...length] of char**)

همون طور که قبلا گفتم . . .

یـــــــادآوری :

متغیری از نوع رشته ای میتواند مجموعه ای از کاراکتر ها را یکجا نگهداری کند (مثل یک کلمه ، یک جمله و ...) بیشنه ی طول یک رشته **255** کاراکتر می باشد و معمولا رشته ها را بین "" قرار میدهند.

چون هر کاراکتر یک بایت فضا نیاز دارد یک رشته که مجموعه ای از کاراکتر هاست به تعداد کاراکترها فضا نیاز دارد (هر کاراکتر یک بایت) بعلاوه ی یک بایت اضافه که طول رشته را نگه میدارد پس:

1 + تعداد کاراکتر = فضای مورد استفاده برای یک رشته

(عدد به دست آمده بر حسب بایت)

میتوانیم طول یک رشته را هنگام تعری محدود کنیم مثلا :

```
Var
     Str : string [5];
```

این تعریف باعث میشه یه رشته تعریف کنیم که فقط **5** کاراکتر رو میتونه تو خودش نگه داره حالا یه سوال این رشته چقدر فضا از حافظه اشغال میکنه ؟؟؟؟؟؟

غضنفر : خب **5** تا دیگه .

Cyclops : ای بابا این فرمول بالا پس چیه؟؟؟

5 تا کاراکتر + یه بایت هم که طول رشته رو نگه داره با هم میشه **6** بایت. حله ؟

غضنفر :**ouk**

Cyclops : خب غضی جون یه سوال اگه **5** رو نگیم یعنی طول رشته رو تعیین نکنیم از کجا بفهمیم چقدر حافظه میگیره.؟؟

غضنفر : خب بستگی داره چه رشته ای توش میریزیم بعد مثل از فرمول بالا حساب می کنیم دیگه

Cyclops : نه زود تو تله نیفت !!

نکته : تو اکثر کامپایلر های پاسکال اگه طول رو مشخص نکنیم کامپایلر بیشینه ی طول ممکن(**255**) رو به رشته اختصاص میده.

پس اگه طول رشته ای رو میدونیم بهتره اونو مشخص کنیم تا برنامه بهینه تر باشه و حافظه ی کمتری هدر بره.

در ضمن همون طوری که ما تونستیم آرایه ای از کاراکتر ها رو تعریف و استفاده کنیم (گفتیم دیگه میتونیم رشته رو با آرایه ای از کاراکتر ها شبیه سازی کنیم) میتونیم آرایه ای از رشته ها رو تعریف کنیم که این هم کاربرد های زیادی داره .مثلا میخوایم یه برنامه واسه یه کتاب فروشی بنویسیم که اسم کل کتاباش رو وارد کنه و بعد تو خروجی چاپ کنه یا حتی کاربردی تر از توی اون لیست کتابی رو که میخواد پیدا کنه و یا هر قابلیت دیگه همچین جاهایی یکی از راه های نگه داری موقت اسمی کتاب همین آرایه ی رشته هاست (راه نگه داری دائم اطلاعات استفاده از فایل هاست که بعدا مفصلا توضیح میدم)

آرایه ای از رشته ها رو دقیقا به همون روش معمول تعریف میکنیم فقط بعد از **of** تو تعریف آرایه ها می نویسیم **string** که اینو به کامپیوتر بفهمونیم که به اون تعداد که گفتیم رشته میخوایم با یه اسم تحت عنوان یه آرایه.

مقایسه ی رشته ها هم مثل کاراکتر ها برحسب جدول اسکی انجام میشه فقط اینجا این نکته رو فراموش نکنید که :

نکته : هر رشته ای کد اسکی کاراکتر اولش عدد کمتری باشه از رشته ی دیگه کوچکتره و بلعکس.

مثلا **BaBAk** از **babak** کوچیکتره و یا **Cyclops** از **4u** بزرگتره.

از این مقایسه های بزرگ و کوچیکی رشته ها میشه راحت برای مرتب کردن رشته ها بر حسب هر موردی که احتیاج دارین تو یه آرایه یا جاهای دیگه استفاده کرد با توجه به همین نکته برنامه ی دفتر تلفن رو شبیه سازی کنید ببینم بلدید ؟؟ (برنامه ای که **20** تا اسم بگیره و **20** تا شماره و بعد لیست اسم ها و تلفن ها رو طوری مرتب کنه که مثلا شماره تلفن شخص **esm[1]** تو **shomare[1]** ذخیره شده باشه جواب هاتون رو ایمیل کنید براتون صحیح کنم البته در اسرع وقت)

توابع و روال های کتابخانه ای پاسکال برای کار با رشته ها :

تابع **length** :

آرگومان(= مقدار ورودی هر تابع را گویند) ورودی این تابع یک رشته است و خروجی این تابع یک عدد صحیح میباشد که بیانگر طول رشته ی ورودی است.

شکل تابع: **Function length (Str: string): Integer ;**

تابع **pos** :

آرگومان های ورودی این تابع **2** رشته می باشند و این تابع رشته ی اول را در رشته ی دوم جستجو میکند و خروجی این تابع یک عدد صحیح است که بیانگر اولین مرتبه ای که رشته ی اول در رشته دوم تکرار شده است می باشد. اگر تابع مقدار **0** را به عنوان خروجی برگرداند به این معنی است که رشته ی اول در رشته ی دوم وجود ندارد از این نکته با استفاده از یک شرط **if** میتوان به راحتی برای کنترل نتیجه ی جستجو اسفاده کرد .

شکل تابع:

```
Function  pos ( Str1: string  ;  Str2: string ): Byte;
```

روال **str** :

این روال یک عدد را دریافت می کند (صحیح یا اعشاری) و آن را به رشته تبدیل می کند و درون متغیری که ما تعیین کرده ایم قرار می دهد .

شکل روال:

```
1:  Procedure  Str( I: integer: format , Str: string ) ;
2:  Procedure  Str ( F: Real: format , Str: string )  ;
```

روال **val** :

این روال درست برعکس روال فبلی است یعنی رشته را به عدد تبدیل می کند . این روال توسط متغیری به نام **Error** کنترل میشود اگر رشته ی ما قابل تبدیل به عدد نباشد مقدار **Error** محلی را که فرایند تبدیل متوقف شده را برمیگرداند در غیر این صورت (تبدیل کامل انجام شده باشد) مقدار **Error** برابر با **0** خواهد بود (باز هم با یه شرط **if** ساده میتونید کنترل نسبتا کاملی روی عمل تبدیل داشته باشید و در صورت وجود هر مشکلی کار مورد نیاز رو انجام بدید)

تابع **copy** :

این تابع **3** تا آرگومان داره که یکیش یه رشتس و **2** تا هم عدد این تابع از کاراکتر مورد نظر به

تعداد دلخواه کپی میکنه و ما میتونیم این مقدار رو تو یه متغیر از نوع رشته ذخیره کنیم (**paste** کنیم)

شکل تابع:

Function copy (S:string ; Index:Integer; count:Integer): string;

روال **Delete** :

این روال از محل **index** رشته ی **string** به تعداد **length** کاراکتر را حذف میکند.

شکل روال:

Procedure delete (Var str: string; Index:inte ger;length: integer);

روال **insert** :

این روال رشته ی اول را از محل مشخص شده درون رشته ی دوم اضافه می کند .

شکل روال:

Procedure Insert (Str1: string ;Var Str2: string ; Index: Byte) ;

تابع **concat** :

این تابع از جمله توابعی است که به تعداد نامحدود و دلخواه (هر تعداد بیش از **2** تا) آرگومان میپذیرد و یک خروجی را برمیگرداند که پیوند خورده و به هم چسبیده ی تمام رشته های ورودی است .

شکل تابع: **Function concat (S1 , S2 , … , Sn): string ;**

در فصل بعد نحوه ی نوشتن توابع و روال های شخصی خودتان برای استفاده در برنامه هایتان را می آموزید .

کارگاه حل تمرین کاراکتر ها و رشته ها : (تستی _ تشریحی)

مثال 1 :

بعد از اجرای بر نامه ی زیر مقدار **m** و **n** و **f** چیست ؟

```
Var
        St: string  ;
        E , N: integer   ;
        F: Real  ;
Begin
       St = '0912'  ;
       Val ( St , m , E )  ;
       St = '  919'  ;
       Val ( St , n , E )  ;
       St = '3.0935'  ;
       Val ( St , f , E )  ;
End.
```

مثال 2 :

مقدار **str** در هر خط چقدر است ؟

```
Var
          Number , Str: String  ;
Begin
          Str ( 543: 5 , Str )  ;     { 1 }
          Str ( 12.25: 7: 2 ,Str )  ;     { 2 }
          Str ( 127: 3: Str )  ;     { 3 }
          Str ( 3.1239: 7: 3 , Str )  ;     {4}
End.
```

مثال 3 :

در خروجی مقدار **searchResult** برابر با چه مقداری چاپ میشود ؟

```
Var
          Str1 , Str2:  string  ;
          searchResult: integer  ;
```

```
Begin
        Str1: = 'cyclops'  ;
        Str2: = 'babak  cyclops'  ;
        searchResult: = Pos ( Str1 , Str2 )  ;
        Write ( ' searchResult = ' , searchResult )  ;
End.
```

مثال 4 :

در خروجی مقدار lengthResult برابر با چه مقداری چاپ میشود ؟

```
Var
        St:  string  ;
        lengthResult: integer  ;
Begin
        St: = 'babak Cyclops(babak haj azim zanjani)'  ;
        lengthResult: = length ( St )  ;
       Write ( ' The  length  of  string  is: ' , lengthResult)  ;
End.
```

مثال 5 :

حاصل روال **insert** زیر چیست ؟

```
Var
      Str1 , str2:  string  ;
Begin
        Str1: = ' garmdareh '  ;
        Str2: = 'university'  ;
        Insert ( Str1 , Str2 , 6 )  ;
End.
```

مثال 6 :

حاصل روال **Delete** زیر چیست (چه چیزی در خروجی چاپ میشود) ؟

```
Var
        St :  string  ;
Begin
        St: = 'univer garmdareh sity'  ;
        Delete ( St , 7 , 11 )  ;
```

```
        writeln ( St ) ;
End.
```

مثال 7 :

بعد از اجرای برنامه ی زیر چه مقداری جایگزین **str1** شده است ؟

```
Var
        Str , str1:  string  ;
 Begin
        Str: = 'babak.cyclops@gmail.com'  ;
        Str1: = Copy ( Str , 14 , 9 )  ;
        write ( Str1 )  ;
End.
```

مثال 8 :

خروجی این برنامه چیست ؟

```
Var
    Str1 , str2 , str3 :  string
Begin
    Str1 : = 'garmdareh'  ;
    Str2 : = '.tk'  ;
    Str3 : = Concat ( Str1 , Str2 )  ;
    write ( Str3 )  ;
End.
```

مثال 9 :

در عبارت زیر تحت زبان توربو پاسکال مقدار **s[0]** برابر با کدام گزینه ی زیر است ؟

```
Const s:string = '123'  ;
```

الف : **#1**

ب : **3**

ج : **#3**

د : سوال سر کاریه .

مثال 10 :

نتیجه ی کدام مقایسه ی زیر **true** می باشد ؟

الف : 'TEST > 'test

ب : 'TEST' > 'test

ج : TEST >= 'test'

د : 'TEST' = 'test'

<< پاسخ ها و راه حل ها >>

1)

m=0912 و (مقدار دهی نشد و e=3) و n = و f=3.0935

2)

خط 1 = '543' ____ خط 2 = '12.25' ____ خط 3 = '127'

خط 4 = ' 3.124'

3)

searchResult =7

4)

lengthResult = 37

5)

حاصل از این قرار است : univer garmdareh sity

6)

خروجی از این قرار است : university

7)

Str = gmail.com

8)

خروجی : **garmdareh.tk**

9)

پاسخ صحیح گزینه ی **"ج"**

10)

پاسخ صحیح گزینه ی **"د"**

برنامه های فرعی (زیر برنامه ها) :

زیر برنامه ها یا برنامه های فرعی همان طور که از اسمشان پیداست تقریبا مثل یک برنامه ی کامل هستند با چند تا تفاوت که مهمترینش اینه که زیر برنامه ها به تنهایی کار فایده ی خاصی ندارند و در صورتی مفیدند که از درون برنامه ی اصلی فراخوانی شوند و مورد استفاده قرار گیرند .
مثلا همین توابع استاندارد پاسکال تا زمانی که تو برنامه ی اصلی اونا را فراخوای نکنید و آرگومان بهش ارسال نکنید کاری انجام نمیدهند ولی برنامه اصلی بلافاصله بعد از اینکه برنامه رو **run** کردید اجرا میشه البته در واقع اونم یه حالت خاصه ما با **run** کردن برنامه (کامپایل و **run**) اونو فراخوانی می کنیم .
خب حالا که یه ذهنیتی پیدا کردید یه ذره بیشتر توضیح میدم (سعی کنید این فصل رو خیلی خوب یاد بگیرید این فصل خیلی مهم و کاربردیه) در واقع زیر برنامه ها برنامه های کوچکی هستند که بار خاصی رو از دوش برنامه ی اصلی بر میدارند و خودشون اونو انجام میدن به یه بیان دیگه اگه برنامه رو مثل یه ماشین تصور کنید کلی جزء داره مثل پیچ و مهره و تسمه و ... تو برنامه متغیر ها هستند ثابت ها و چیزای دیگه و از جمله زیر برنامه ها . خب مسلما میدونید که این پیچ و مهره ها و اجزای دیگه هر کدوم تو ماشین نقش خاص خودشون رو بازی می کنن و اگه همه کارشون رو درست انجام بدن ماشین راه میره .
مثل دینام ماشین که توسط تسمه فراخوانی میشه و کاری که وظیفشه یعنی تولید برق رو انجام میده و ماشین (برنامه ی اصلی) از نتیجه ی تلاش دینام استفاده میکنه .
دیگه نمیدونم به چه زبونی بگم . . .
تو برنامه ها ی فرعی این پارامتر ها هستند که نقش تسمه رو بازی میکنن یعنی ما بدون فرستادن پارامتر ها نمیتونیم یه زیر برنامه رو فراخوانی کنیم و رابط ما با زیر برنامه ها پارامتر ها هستند(به جز زیر برنامه های خاصی که به آرگومان ورودی احتیاج ندارن که اگه یادم بمونه مثالش رو مینویسم)

ما دو نوع زیر برنامه داریم : روال ها و توابع

روال ها :

روال ها نوعی از زیر برنامه ها هستند که قابلیت انعطاف بیشتری نسبت به توابع دارند چون هم میتونن با استفاده از پارامتر های ارسالی و یا بدون اون کار های مورد نظر رو انجام بدن و هم میتونن روی پارامتر های ارسالی تغییرات مورد نظر رو اعمال کنن واسه اینکه بهتر متوجه بشین اول طریقه ی تعریف کردن روال ها رو می گم بعد هم انواع روش ها برای استفاده از پارامترها .
ما روال ها رو بعد از اعلان نام برنامه و قبل از **begin** اصلی برنامه تعریف میکنیم
بدین شکل :

Program اسم برنامه ی اصلی ;

تعاریف برنامه اصلی ;

Procedure Name (parameters list) ;

کلمه ذخیره شده — اسم روال — لیست پارامترها

```
Var
      { List of locad variable }
Begin
          .
            .
              { Procedure Body }
       .
       .
End ; { End of procedure }
Begin   { main program }
                .
                .
                .
```

فراخوانی روال ها

.

.

End. { End of program }

ما بسته به نوع استفادمون و کاری که میخوایم انجام بدیم زمان تعریف روال ها از دو نوع پارامتر میتونیم استفاده کنیم :

الف : پارامتر های مقداری (**value**)

ب : پارامتر های متغیری (**variable**)

الف : پارامتر های مقداری :

همون طور که از اسم این نوع معلومه در این حالت فقط مقدار پارامترها زمان فراخوانی روال به روال ارسال میشه و در واقع فقط اینجوری فکر کنین که یک کپی از اون تهیه میشه و ما تو روالمون هر تغییری رو روی اون مقادیر انجام بدیم این تغییر روی کپی اعمال میشه نه روی متغیر اصلی مثال زیر رو خوب ببینین همه چی رو متوجه میشین .

راستی برای اینکه پارامتر ها از نوع مقداری وارد روال بشن (یه صورت مجازی ازشون کپی تهیه بشه) تو قسمت داخل پرانتز (لیست پارامتر ها)اینجوری مینویسم :

Procedure Name (var1 : type ; var2 : type , ...) ;

کلمه ذخیره شده — اسم روال — لیست پارامترهای از نوع مقداری

یعنی فقط اسم متغیر بعلاوه ی : و نوعش فقط همین .

بزارین اون یکی نوع رو هم بگم بعد مثال اینجوری جالب تر میشه .

ب : پارامتر های متغیری :

اگه روالی که فراخوانی میشه موقع تعریف پارامترهاش از نوع متغیری تعریف شده باشن اون

موقع هر تغییری رو در مسیر کدهای روال روی اون پارامتر انجام بدیم در متغیر برنامه ی اصلی هم اعمال میشه واسه ی تعیین یک پارامتر از نوع متغیری مثل نوع قبل عمل میکنیم با این تفاوت که قبل از اسم پارامتر کلمه ی رزرو شده ی **var** رو مینویسیم .

خب بریم مثال :

خروجی برنامه ی زیر چیست ؟

```
Procedure m(x:integer ; var y:integer) ;
Begin
x:x+y ; y:x-y ; x:y*2 ;
end ;
var x,y :integer ;
begin
x:5 ; y:2; m(x,y) ;
write(x,””,y) ;
end.
```

اولا یه نکته

شاید به نظرتون ظاهر برنامه یه ذره عجیب به نظر بیاد ولی باید اونو بدونید که میشه تو یک خط چند تا دستور رو نوشت ولی نمیتونین از سمی کالن ها فاکتور بگیرید و حذفشون کنید .

نکته ی بعدی اینکه این سبک برنامه نویسی خیلی بده همین فاصله نذاشتن بین دستورات رو میگم این میتونه تو برنامه های یه ذره حجیمتر مشکل زا بشه و باعث گیج شدن شرید شما بشه مخصوصا وقتی از دستورات تو در تو استفاده میکنید .

برای اینکه زمان برنامه نویسی راحت تر از فاصله ها استفاده کنید به جای **space** از کلید **tab** بهره ببرید تا هم نظم و هم راحتی رو یک جا داشته باشین .

همچنین یک نکته ی دیگه هم اینکه خوبه آگه از اسامی با معنایی برای متغیر ها ، روال ها و . . . استفاده کنین مثلا تو مثال بالا به جای روال **m** خوبه آگه بنویسیم **cal** تا این مفهوم رو برسونه که یه محسابه ای انجام میشه .

با رعایت کردن این چند مورد از گیج شدن خودتون جلوگیری میکنید همچنین درک برنامه واسه سایرین راحت تر میشه و خودتون هم بیشتر از برنامه نویسی لذت می برید همچنین میتونید از

{} و یا (* *) برای اضافه کردن توضیحات استفاده کنید .

البته باید اینو بدونید تو تست ها و آزمون ها طراح سوال معمولا اینجوری تر وتمیز واستون سوال نمینویسه و بیشتر از همون روش شلخته ی بالا استفاده می کنن تا طرف یه ذره گیج بشه .

پس خوبه اگه برنامه ی بالا رو اینجوری بنویسیم :

```
Program babak_cyclops  ;          {program name}
Procedure cal(x:integer ; var y:integer) ;
Begin
      x:x+y ;
       y:x-y ;
       x:y*2 ;
end ;                             {end of procedure}
var x,y :integer ;                {var define}
begin                             {main program}
           x:5 ;
           y:2;
           cal(x,y) ;
           write(x,””,y) ;
end.                                  {end of main program}
```

خب باز برگردیم سر سوال پرسیده بودم خروجی این برنامه چیست ؟

خب بچه ها نظر شما چیه ؟

غضنفر **1** :

x و **y** هر دو **5** چاپ میشه .

Cyclops :

خب دیگه ؟؟؟

غضنفر **2** :

x و **y** هر دو **2** چاپ میشه .

غضنفر 3 :

x برابر 2 چاپ میشه و y هم 4

غضنفر 4 :

نه نه برعکس این میشه 4 و 2

Cyclops :

خب دیگه نبود ؟؟؟؟

خب اونایی که خوب درس رو متوجه شدن اگه یه ذره فکر کنن متوجه میشن که غضنفر 1 درست میگه یعنی هر دو 5 چاپ میشن .

حالا چرا ؟؟؟؟؟؟؟؟؟؟؟

همیشه تو همچین سوالایی اول برین سراغ برنامه ی اصلی بعد خط به خط کدها رو دنبال کنین هر وقت روال یا به طور کلی زیر برنامتون فراخوانی شد برین به زیر برنامه تمام کد ها رو دنبال کنین و باز به برنامه ی اصلی برگردین و ادامه ی کد ها رو دنبال کنین تا آخر برنامه ؛ تا حساب کار از دستتون در نره خب یعنی چی ؟؟؟

من پیش خودم اینجوری میگم که :

تو برنامه ی اصلی به x مقدار اولیه ی 5 و به y مقدار اولیه ی 2 داده میشه بعد روال فراخوانی میشه یعنی باید در صورت وجود پارامتر ها به روال ارسال بشن اونم از نوعی که تو زیر برنامه معین شده خب به خط اول زیر برنامه نگاه میکنیم و میبینیم که x رو از نوع مقداری و y رو از نوع متغیری احتیاج داره پس ما هم اینجوری عمل میکنیم که از x موجود تو برنامه اصلی (که مقدار دهی شد با 5) یه کپی میگیریم و میریزیم تو یه x دیگه من واسه اینکه قاطی نکنین x و y تو روال رو از این به بعد با قرمز مینویسم .

مقدار x (یعنی 5) کپی میشه تو x

و خود متغیر y میره جای y پس هر تغییری روی y در زیر برنامه اعمال بشه در واقع روی y (اصلی تو برنامه)اعمال شده پس من اجازه دارم به جای y ها بنویسم y پـــــــــــــس :

تو خط اول زیر برنامه 5 با 2 جمع میشه و حاصل(7) به جای x ذخیره میشه .

تو خط دوم زیر برنامه 7 از 2 کم میشه و حاصل(5) به جای y قرار میگیره .

تو خط سوم زیر برنامه 5 ضربدر 2 میشه و حاصل(10) جای x ذخیره میشه .

بعد زیر برنامه(روال) تموم میشه و ما باز به برنامه برمیگردیم تا همون طور که گفتم ادامه ی برنامه اصلی رو دنبال کنیم

خب میرسیم به دستور **write**

ولی میخوایم چه چیزی رو چاپ کنیم (x و y) یا (x و y)یا (x و y) و یا حتی چیز دیگه ای غیر از اینا ؟؟؟

اینو باید بدونین به محض اینکه از زیر برنامه یا همون روالمون اومدیم بیرون دیگه x و y نداریم و اونا از بین میرن پس ما الان یه x داریم که هنوز مقدار اولیش رو داره (همون5) و یه y داریم که مقدار اولیش 2 بود ولی تو زیر برنامه عوض شد و الان مقدارش 5 هست پس ما خروجی جناب غضنفر 1 رو میبینیم یعنی هر دو 5 چاپ میشن .

اگه حجم این مثال زیاد شد دلیل بر سخت بودنش نیست اولا توضیح ها زیاد بود و خواستم حداقل این یه مثال رو خوب بفهمین دوما در مورد نوشتن برنامه طوری که تر و تمیز تر باشه صحبت کردیم پس اگه یه کم سردرگم شدید واسه جمع بندی یه بار فقط روند حل مساله رو نگاه کنین بدون توضیح های اضافیش بازم مشکلی بود ایمیل کنید در خدمتیم .

خب ادامه ی بحث روال ها :

به طور کلی تو زیر برنامه ها ممکنه با دو تا اصطلاح برخورد کنین متغیر های محلی و سراسری که اگه بخوایم به زبون اجنبی ها بگیم میشن **local & global variable** خب اینا چی هستند ؟

خیلی سادس ببینین گفتم زیر برنامه ها خودشون مثل یه برنامه ی کامل هستند با یه کم تفاوت یعنی ما حتی میتونیم متغیرهایی رو که تو زیر برنامه ها احتیاج داریم همون موقع تو زیربرنامه تعریف کنیم خب در این صورت اینگونه متغیر ها فقط تو همون زیر برنامه میتونن استفاده بشن و اصطلاحا بهشون محلی میگیم در واقع متغیر های محلی هر جا تعریف بشن تو فقط تو همون بلاک مورد استفاده هستند ولی متغیر های دیگه که تو قسمت **var** برنامه ی اصلی تعریف میشن رو میتونیم همه جا استفاده کنیم به اینگونه متغیر ها هم اصطلاحا میگیم سراسری استفاده ی محلی تر از متغیر ها اکثرا بهتره چون این مزایا رو داره :

1) میتونیم یه اسم رو چند جا استفاده کنیم (زمانی که بخوایم از اسامی با معنی استفاده کنیم یه موقع هایی اسامی با معنا کم می یاریم همچین مواقعی دو تا راه داریم اول اینکه به اسم مورد نظرمون پسوند اضافه کنیم دوم هم این که متغیر ها رو محلی تر استفاده کنیم)

2) اسفاده ی بهینه تر از حافظه (یعنی به جای اینکه به اندازه ی همه ی متغیر ها اول برنامه از حافظه فضا بگیریم هر زمانی که احتیاج داشتیم این کار رو انجام میدیم)

آخرین مورد مهم در مبحث روال ها همون استفاده از روال های بدون پارامتره که خدا رو شکر یادم مونده تا براتون بگم یه توضیح کوچولو بگم و بعد مثال :

این روال ها معمولا جاهایی بدرد می خورن که میخواید یه کار خاص و همیشه یه جور رو یه جاهایی از برنامه انجام بدین همیشه اونو بدونین که قدرت یه روال به پارامترهای ورودیشه و روال بدون پارامتر تقریبا بیشتر جنبه ی تزیین برنامه رو داره (مثل همین مثال پایین) اگه بخوام مثل همون مثال ماشینه تو اول بحث روال ها بگم اینجوری میشه که روال بدون پارامتر مثل دینام بدونه تسمه میمونه که نمیتونه نیروی موتور رو به برق تبدیل کنه .

مثال :

```
Program       babak_cyclops ;
Procedure     Head   ;
 Begin
       WriteLn('garmdareh = az Tehran ke miri karaj  hodudan 10
       kilumetr munde be karaj mishe garmdareh');
       WriteLn('……...');
  End;
Begin{Main}
          WriteLn( ' garmdareh university '  ) ;
          Head;
End.
```

خب همون طور که متوجه شدید این روال فقط هرجا فراخوانی بشه آدرس حدودی گرمدره رو مینویسه فقط همین . . .

تبریک میگم روال ها یعنی یکی از مهمترین مباحث برنامه نویسی تموم شد اگه خوب یاد گرفته باشین خیلی راحت تر و سریع تر توابع رو یاد می گیرین .

توابع :

توابع نوع دیگری از زیر برنامه ها هستند با چند تفاوت جزئی :

1) پارامترهای ارسالی به توابع همیشه از نوع مقداری هستند ولی در روال ها میتوانستیم هم از مقداری استفاده کنیم هم از متغیری .

2) تابع ها در برنامه نویسی دقیقا مثل توابع در ریاضی فقط یک خروجی دارند ولی روال ها از این نظر محدودیتی ندارند .

3) تابع یک **data** تولید میکند که معمولا در انتهای تابع به اسم تابع نسبت داده میشود ولی در روال ها اینگونه نبود .

توابع هم مانند روال ها درست بعد از قسمت تعریف برنامه ی اصلی و قبل از **begin** برنامه ی اصلی تعریف میشوند بدین صورت که :


```
Function  Name ( List Of Parameters  ):Function Type ;
Var
.............
Begin
.
.
.
      Name := Result of fanction ;
End;              {end fanction}
Begin    {main program}
.
.
End.     {end of program}
```

مثال :

برنامه‌ای بنویسید که یک عدد از ورودی دریافت کرده سپس توسط تابعی بنام **fact** ، فاکتوریل عدد را محاسبه در برنامه اصلی چاپ نماید.

```
Program      garmdareh_tk ;
Var          N:  Integer  ;
Function     Fact (  M :  Integer  ) :   longint ;
Var          P , I : Integer ;
Begin
        P:= 1 ;
        For     i:= 2    To     M    do
                P:= P * I ;
         Fact:= P ;
End ;
Begin { Main  }
       Write ( ' Enter Number = ' ) ;     Readln ( N ) ;
       Writeln (' Factorial  is = ' , Fact ( N ) ) ; {Call Function}
End.
```

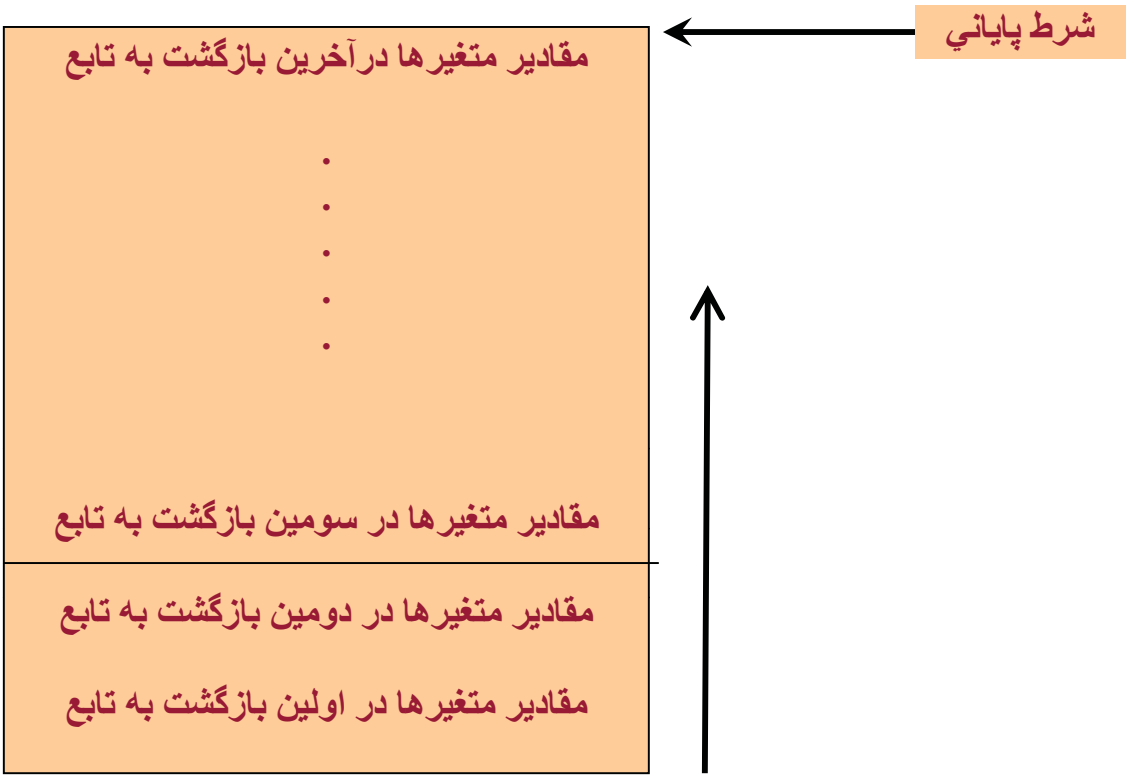
توابع بازگشتی **(Recursion Functions)** :

در پاسکال یک تابع یا روال می‌تواند، خودش را فراخوانی نماید. پیمانه‌ای که خودش را فراخوانی می‌کند یک پیمانه بازگشتی نام دارد. این نوع توابع در بدنه خود، خودشان را فراخوانی می‌کنند. این فراخوانی تا برقراری یک شرط خاص که غالباً به یک عدد ثابت ختم می‌شود، ادامه پیدا می‌کند.

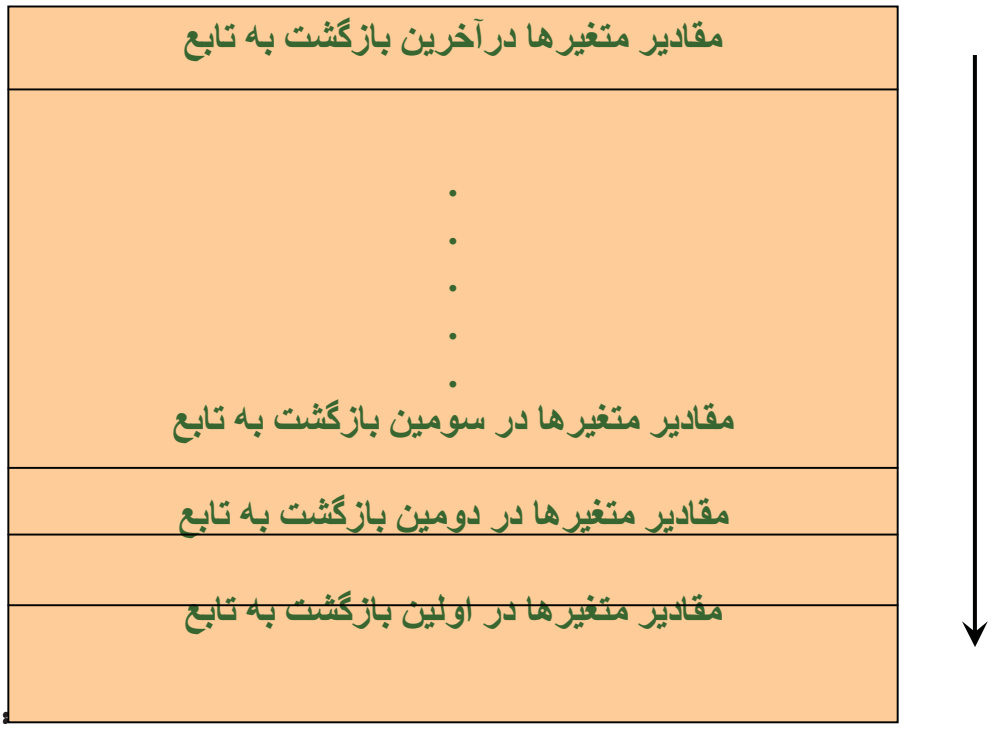
سپس مقدار تابع از پایین به بالا محاسبه می‌شود و در نهایت نتیجه تابع حاصل می‌شود.

هنگامی که تابع بازگشتی به خود مراجعه می‌کند، مقادیر فعلی متغیرهای خود را در محلی از حافظه بنام پشته **(stack)** قرار می‌دهد، اگر بازگشت به تابع بازگشتی مجدداً صورت گیرد، مقادیر فعلی متغیرها مجدداً بدنبال مقادیر قبلی و اصطلاحاً در پشت مقادیر اولیه قرار می‌گیرند. هنگامی که شرط پایانی در تابع بازگشتی رخ می دهد، در اولین بازگشت مقادیری را که هنگام مراجعه به خود، در پشته نگهداری کرده، مجدداً در دسترس قرار می‌دهد و بهمین ترتیب در بازگشت‌های بعدی این عمل تکرار می‌شود تا مقدار تابع محاسبه شود.

شکل زیر نحوه قرار گفتن مراجعه‌های یک تابع بازگشتی به پشته را نمایش می‌دهد.



ترتیب دسترسی به مقادیر هنگام بازگشت از تابع بازگشتی از بالا به پایین می‌باشد و می‌توان آنرا بصورت زیر نمایش داد



مثال :

برنامه‌أی بنویسید که با استفاده از توابع بازگشتی فاکتوریل عدد صحیح **N** را محاسبه نماید.

```
Program      Example ;
Var          N : integer ;
Function     Fact  (  N : integer  ) :  Longint ;
Begin
        If   N = 1  Then
               Fact:= 1
        Else
               Fact:=  N  *  Fact ( N – 1 ) ;
End ;
Begin
        Writeln ( ' Enter   Number ' ) ;
        Writeln ( ' Factorial  is = ' ,  Fact ( N ) ); {Call Function}
 End  .
```

تعریف نوع (type) جدید :

همون طوری که گفته بودم این قسمت هم تو قسمت اعلان برنامست ما علاوه بر نوع های استانداردی که تو خود پاسکال داریم (**byte , integer , string , ...**) میتونیم نوع های مورد استفاده ی خودمون رو تعریف کنیم و پس از اون هر کجای برنامه لازم شد ازشون استفاده کنیم .

اینجوری نوع جدید تعریف میکنیم :

Type شناسه = تعریف داده ساخت یافته

مثلا در مبحث زیربرنامه ها خیلی نیازه که برای فرستادن آرایه ها به توابع و روال ها از این روش استفاده کنیم و درواقع مهمترین کاربردش همین جاست .

خب یه مثال از همین کاربردش : (مثال 1)

Type Name = array [1...20] of string ;

تو مثال بالا ما یه نوع جدید تعریف کردیم که اینجوری متغیری رو از این نوع تعریف میکنیم :

```
Var
        Name_array: Name ;
```

و معنیش اینه که :

متغیر **name_array** از نوع آرایه ی **20** تایی از رشته هاست .

مثال **2** :

برنامه ای بنویسید که با استفاده از روال ها یک عدد را از ورودی بخواند و به همان تعداد اسم دانشجو از ورودی دریافت کند .

```
Program new_type ;
        Type    stu_name = array [1..20] of string ;
Var
        Student : stu_name ;
        I , k : integer ;
Procedure input (var a:stu_name ; i:integer) ;
        Var
                j : integer ;
        begin
                for ( j:=1 to I ) do
                        read (a[j]) ;
        end ;
{pqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqpqp}
begin       {main program}
        read (i) ;
        input ( student , I ) ;
        for ( k:=1 to I ) do
                writeln(student[k]) ;
End. {main program}
```

مجموعه ها و داده های شمارشی :

مقدمه :

تعداد محدودی از داده‌ها که از نظر نوع یکسان هستند، در غالب مجموعه و یا گونه‌های شمارشی نگهداری می‌شوند که مفهومی شبیه در ریاضیات دارند. برای استفاده در مواردی خاص نظیر روزهای هفته و یا نوع ماشینها و... که ترتیبی هستند و یا یک مجموعه از داده‌های پشت سر هم می‌باشند، استفاده از این ساختارها کار را بسیار راحت می‌کند. هر چند وجود آنها به عنوان ساختارهای داده‌ای، الزامی نیست. بهرحال به عنوان ابزارهایی از زبان پاسکال هستند که در مواقعی، ضروری بنظر می‌رسند و مسئله را به صورتی قابل‌فهم و راحت حل می‌کند.

مجموعه‌ها **(Sets)** :

در زبان پاسکال مجموعه، مفهومی شبیه به مفهوم مجموعه در ریاضیات جدید دارد. متغیری است که شامل لیستی از اعداد صحیح، کاراکتر، بولین و یا از نوع شمارشی می‌باشد که دارای تعداد عناصر محدود به حداکثر **256** تا می‌باشد. از این جهت بسیار شبیه به یک آرایه در زبان پاسکال است که شامل داده‌هایی از یک نوع می‌باشد، ولی آرایه دارای عناصر محدودی نیست و در ضمن مانند آرایه تعریف نمی‌شود.

تعریف مجموعه :

برای تعریف یک مجموعه از کلمات کلیدی **set of** بصورت زیر استفاده می‌کنیم:

```
Type
    Name= Set of  Type Of  Set;
```

مثال **1** :

```
Type
    Digit_type = 0..9;
    Digit = Set of  Digit_type
```

مثال 2 :

```
Var
      ch: set of char;
      bool: set of boolean;
      num: set of byte;
      x: set of 100..200;
      y: set of 'a'..'z';
      z: set of '0'..'9';
```

عملیات روی مجموعه‌ها :

در ریاضیات عمل عضویت وجود دارد که به این معنی است که متغیری عضو مجموعه می‌باشد یا خیر. این عمل در زبان پاسکال با کلمه کلیدی **in** صورت می‌گیرد. اگر عضویت صحیح باشد جواب **True** وگرنه **False** می‌باشد. همچنین دو مجموعه را می‌توان با علامات شرطی =، <>، <= و >= مقایسه کرد که همگی دارای خروجی درست یا غلط می‌باشند. ولی علامات شرطی > و < در مورد مجموعه‌ها کاربردی ندارد.

مثال :

```
Var
      a: set of byte;
Begin
      a:= [1..3,4];
      if  3 in a then write('3 is in set a')          چاپ می‌شود
      else write('3 is not in set a');
      write(3 in a);                                  True
      write(8 in a);                                  False
      write([1,2,3,4,] = a);                          True
      write([1,2,3,4] = [2,2,1,1,1,3,4,4]);           True
      write([1,2] = [1,2,3]);                         False
      write([1,2] <= [1,2,3]);                        True
      write([] >= [1]);                               False
End.
```

عملگرها روی مجموعه‌ها :

در ریاضیات می‌توانیم مجموعه‌ها را با هم اجتماع، اشتراک و تفاضل کنیم که این عملیات در پاسکال با عملگرهای +، * و - به ترتیب می‌باشد. اجتماع دو مجموعه ترکیبی از همه عضوهای آنها است و اشتراک یعنی عضوهایی که در هر دو مجموعه مشترک است و تفاضل یعنی اعضایی که در مجموعه اول می‌باشد و در مجموعه دوم وجود ندارد.

مثال :

```
Var
    a: set of byte;
    b: set of 0..10;
    c, d, e: set of byte;
Begin
    a:=[0,1,2,3];
    b:=[2,3,4,5];
    c:= a+b;                    c=[0,1,2,3,4,5]=[0..5]
    d:= a*b;                    d=[2,3]
    e:= a-b;                    e=[0,1]
End.
```

در استفاده از مجموعه ها به نکات زیر توجه کنید :

1. آرگومان ورودی روالها می‌تواند مجموعه باشد که قبلاً در تایپ تعریف شده باشد، نه اینکه مستقیم در روال به عنوان آرگومان بیاید. ولی خروجی یک تابع نمی‌تواند از نوع مجموعه باشد.
2. برای نوشتن یا خواندن مجموعه‌ها باید عضو به عضو عملیات صورت بگیرد و مستقیماً توابع **Read ،Write** روی آنها کار نمی‌کنند.

مثال : برنامه‌ای بنویسید که تعداد محدودی عدد مختوم به **-1** را از ورودی بخواند و در یک مجموعه از اعداد صحیح قرار بدهد. سپس اعضای این مجموعه را با توجه به مجموعه ساخته شده در خروجی چاپ کند:

```
Program  test ;
Var
   num , temp : set of byte;
```

```
    I , c : integer ;
Begin
     Writln  ('Enter numbers: ');
     Readln( I ) ;
     Num := [ ] ;
     c := 0 ;
   while  ( i  > -1)  do
    begin
          c := c + 1 ;
          temp := [ I ] ;
          num := num + temp ;
          Readln( I ) ;
    end;
    for  I := 0  to  255  do
         if   ( i  in  num )  then
              write( i:5 ) ;
End.{ End  Of   Program  }
```

داده‌های شمارشی (**Enumeration**) :

داده‌های شمارشی، یک مجموعه مرتب از اعداد است که در برنامه، هر عدد دارای نام بخصوصی است. این نامها در داخل دو پرانتز باز و بسته قرار می‌گیرند و بترتیب از صفر مقدار می‌گیرند مگر اینکه برنامه‌نویس به آنها مقدار مخصوصی بدهد.

به مثال زیر توجه کنید :

```
Type
      Cars_type = (Peykan, Pride, Pegout, PK);
Var
      Cars: cars_type;
```

حال در برنامه **cars** می‌تواند مقادیر داخل داده‌های شمارشی را بگیرد :

```
cars:= Pride;
cars:= PK;
```

مثال :

```
Type
  Days_type  = ( sat, sun, mon, tue, wed, thu, fri );
Var
  Day: Days_type;
```

```
Begin
   Day:= Mon;
   i:= ord (sat) + ord (day) + (succ(sat));
   j:= ord ( pred (sun) + ord (pred (fri));
  writeln(i:5, j:5);
End.
```

خروجی :

i = 3 j = 5

نکته ی مهم :

متغیر داده شمارشی شبیه مجموعه‌ها حداکثر دارای **256** عضو می‌تواند باشد که از صفر تا **255** شماره‌گذاری می‌شود و لذا یک بایت حافظه را اشغال می‌کند.

عملیات روی داده‌های شمارشی :

داده‌های شمارشی همانند هر نوع تایپی می‌توانند در **type** برنامه تعریف شود و به عنوان آرگومانهای روالها و یا خروجی توابع می‌تواند درنظر گرفته شود ولی حتماً باید در **type** تعریف شده باشد و مستقیماً نمی‌توان بکار برد. این عمل مشابه مجموعه‌ها و آرایه‌ها نیز می‌باشند، لذا ابتدا در **type** تعریف می‌شود، سپس به عنوان ورودی و یا خروجی روالها استفاده می‌شود وگرنه خطای کامپایلری پیش خواهد آمد. همچنین می‌توان در آرایه‌ها از داده‌ها شمارشی استفاده کرد.

مثال : مشابه مجموعه‌ها نمی‌توان داده‌های شمارشی را مستقیماً **Read** یا **Write** کرد.
برنامه‌ای بنویسید که اعداد صفر تا شش را به مراتب دریافت کرده و به کمک داده‌های شمارشی روزهای متناظر با آنها را از شنبه تا جمعه را حساب کند.

```
Program test:
Type
     days_type = ( sat,sun,mon,tue,wed,thu,fri ) ;
Var
     day: days_type;
     i: byte;
Begin
```

```
      Write('Enter your days number: ' );
      Readln(i);
      While (i>=0) and (i<=6) do begin
      Case   I      Of
      0: day:= sat;
      1: day:= sun;
      2: day:= mon;
      3: day:= tue;
      4: day:= wed;
      5: day:= thu;
      6:day:= fri;
  end;
  Case    day    Of
    sat: writeln ('your day is sat');
    sun: writeln ('your day is sun');
         :
  end;
  Readln(i);
End; {while}
 End.
```

رکورد ها (Records) :

مقدمه :

نوع داده ساخت‌یافته‌ای که در اینجا مطرح می‌شود، رکورد نام دارد که جهت نگهداری داده‌های مختلف نظیر نام، نام خانوادگی، سن و آدرس برای یک دانشجو بکار می‌رود. رکوردها برخلاف آرایه‌ها که دارای عناصر از یک جنس و نوع هستند، دارای عناصر از انواع مختلف می‌باشند. اطلاعات ازنوع مختلف را نمی‌توان در آرایه نگهداری کرد چرا که دارای جنس مشابه نیستند و لذا از ساختاری به نام رکورد استفاده می‌شود.

تعریف رکوردها :

نوع داده ساخت‌یافته ای که از یک سری داده ها یا اطلاعات مرتبط به هم تشکیل می شود.
هر کدام از اطلاعات را یک فیلد می نامند .
بطور کلی تعریف نوع رکورد در زیر آمده است :

```
Type
      Name = Record
            Field1-list : type1;
            Field2-list : type2;
            ..
            ..
            Fieldn-list : typen;
End;
```

برای ساختن رکورد، از کلمه کلیدی **Record** استفاده می‌شود که مطابق زیرمی‌باشد :

```
Type
    Student = Record
        Name: String [10];
        Family: String [15];
        Age: integer;
        Address: String;
    End;
```

در این تعریف **Student** از نوع **Record** است. حال می‌توان در **var** برنامه تعریف کرد :

```
Var
   S: Student;
```

دسترسی به فیلدهای رکورد :

برای دسترسی به فیلدهای رکورد ازعلامت '.' استفاده می‌شود. یعنی بصورت‌زیر :

نام فیلد . نام متغیر رکورد

مثال: برنامه‌ای بنویسید که رکوردی از نوع اعداد ایجاد کرده و مقادیر آنرا مقدارهی کند.

```
Type
      Numbers = record
           a, b, c = integer;
           x, y, z = Real;
end;
Var
      Num = Numbers;
begin
      Num. a: = 2;
      Num. b: = 3;
      Num. x: = 2.5;
      Num. y: = 3.5;
      Num. z: = 10;
Write (Num.a + Num.b + Num.x + Num.y + Num.z);
End.
```

مثال : برنامه‌ای بنویسید که رکورد یک دانشجو را داشته باشد و با توجه به جنس او کلمه آقا و یا خانم را به همراه نام و نام خانوادگی‌اش را چاپ کند.

```
Type
   Student = record
      Id: integer;
      Name: string[10];
      Family: string[15];
      Sex: char;
      Age: integer;
   End;
Var
   Stu: student;
Begin
```

```
        Stu.id:= 80132;
        Stu.name:='Ali';
        Stu.family:='Ahmadi';
        Stu.sex:='M';
        Stu.sge:=18;
        Case   sex   of
          'M': write('Mr.');
          'F': write('Mrs.');
        end;
       write(stu.name, ' ' ,    stu.family);
End.
```

بدست آوردن حجم یک رکورد :

برای بدست آوردن فضای اشغال شده توسط رکورد ابتدا باید فضای اشغال شده توسط تمامی فیلدها را بدست آورده و سپس باهم جمع کنیم.

مثال زیر را درنظر بگیرید :

```
Type
        List1 = Array [1… 5] of integer;
        List2 = Array [1… 5] of char;
        Rectype = Record
        A, B: Real;                    →          2*6
        C, D: String [10];             →          1+10
        F: Array [1… 10] of Boolean;   →          10*1
        G: list1;                      →          5*2
End;
Var
        x: Rectype;
```

12+11+10+10=43 بایت فضا اشغال می‌شود.

رکوردهای تودرتو :

فیلدهای یک رکورد می‌توانند از هر نوعی باشد، از جمله می‌توانند از نوع رکورد دیگری باشند. در اینجا نیز مشابه قبل دسترسی به همان صورت می‌باشد فقط به تعداد رکوردهای تودرتو، '.' پیش می آید.

به مثال زیر توجه کنید:

```
Type
      Rec = record
            a, b: integer;
            c: char;
            x: Record
                 p: integer;
                 q: integer;
             End;
  End;
Var
      r: Rec;
```

دراینجا یک رکورد تودرتو به نام **Rec** تعریف شده است که متغیر **r** از آن نوع تعریف شده است. سپس برای دسترسی به فیلدهای **a , b ,c** می‌توان به صورت زیر عمل کرد :

```
r.a
r.b
```

ولی برای دسترسی به فیلدهای **p ,q** چون متعلق به رکورد **x** نیز هستند داریم:

```
r.x.p
r.x.q
```

آرایه‌ای از رکوردها :

هنگامی‌که ما تعدادی داده مشابه داریم ولی در هریک، داده‌های مختلفی وجود دارد می‌توانیم یک رکورد تعریف کرده، سپس آرایه‌ای از آن تعریف کنیم.

```
Type
   Student = record
      Name: string[10];
      Id: integer;
      Age: integer;
   End;
   Arr_stu: array[1..10] of student;
Var
   S: Arr_stu;
```

در بالا ابتدا یک رکورد دانشجو تعریف شده است، سپس به تعداد **10** نفر دانشجو تحت آرایه **Arr_stu** شکل گرفته است و سرانجام متغیری به نام **S** از نوع آن تعریف شده است.
مشابه ساختار آرایه، چیزی عوض نشده است و فقط هر عنصر آرایه، یک رکورد می‌باشد که دارای سه فیلد مطابق جدول فوق می‌باشد.

برای دسترسی به آرایه فوق داریم:

S[1].name
S[1].id
S[1].age
S[2].name
S[2].id
..
..
..

ارسال رکورد به زیربرنامه‌ها :

رکوردها را می‌توان مشابه انواع تعاریف ساده دیگر به صورت پارامترهای متغیر و مقدار به زیربرنامه ارسال کرد. ولی نوع برگشتی توابع نمی‌تواند از نوع رکورد باشد، یعنی حتماً باید از نوع ساده نظیر **integer، char** و... باشد و از انواع ترکیبی نظیر رکورد، آرایه، مجموعه و فایل نمی‌تواند باشد.
اگر بخواهیم رکوردی را بصورت پارامتر به زیربرنامه ارسال کنیم، ابتدا باید آنرا در **type** تعریف کرده و سپس ارسال شود وگرنه کامپایلر خطا صادر می‌کند.
مثال : می‌خواهیم برنامه‌ای بنویسیم که رکورد **stu** را برای **30** دانشجو ایجاد کرده، به کمک زیربرنامه مقداردهی شده و چاپ شوند.

```
Program test;
Const     no = 30;
Type
        Stu-rec = Record
                Name: string [10];
                ID: integer;
                Age: record
                    Day: integer;
                    Month: integer;
                    Year: integer;
                End;
```

```
End;
Stu_arr=array[1..no] of stu_rec;
Var
        Stu: stu_rec;
Procedure ReadStu(Var S: Stu_rec);
Var    i: integer;
Begin
   for i:= 1 to no do
        Readln(S[i].name, S[i].id, S[i].age.day, S[i].age.month,
S[i].age.year);
End;
Procedure WriteStu(S: Stu_rec);
Var    i: integer;
Begin
   for i:= 1 to no do
Writeln(S[i].name,' ' , S[i].id, ' ' , S[i].age.day,' ' , S[i].age.month,' ' ,
S[i].age.year);
End;
Begin
   ReadStu(Stu);
   WriteStu(Stu);
End.
```

فایلها (files) :

فایل ها یکی از مهمترین و سخت ترین قسمت زبان های برنامه نویسی محسوب میشن البته اولش یعنی اگه خوب یاد بگیرین تقریبا دیگه تو زبون های دیگه هم مشکلی نخواهید داشت منم واقعا تمام سعی خودم رو میکنم بهترین بیان ها و روش هایی که به ذهنم میرسه یا تو آرشیوم دارم براتون بنویسیم .

خب بریم سر درس :

ببینین تا حالا از خودتون سوال کردین حالا کیریم که یه برنامه نوشتیم که اسم **100** تا دانشجو رو به همراه نمراتشون بگیره و کارنامه کامل اونا شامل : نمره مستمر ، نمره ی ترم ، معدل و غیره رو به ما بده خب که چـــــــــــی ؟؟؟؟؟؟؟؟

همچین برنامه ای بدون وجود یه تایپیست مخصوص این برنامه تو اون دانشگاه چه جوری می خواد کار کنه ؟؟؟

غضنفر :

خب ببین برنامه هم واسه خودش یه شرایطی داره دیگه نمیشه که فقط اون نیازهای ما رو برطرف کنه ما هم باید حداقل نیازهاش یعنی مقادیری که نیاز داره رو بهش بدیم .

Cyclops :

این چه حرفیه غضی جون خب پس بزار زودتر راه علمیش رو برات بگم تا از جهل خاصله بگیری .

ببین قبول دارم که ما واسه ی همچین برنامه ای لازمه یه سری مقادیر وارد برنامه بشن ولی ما حداقل میتونیم مقادیر تکراری مثل اسم و به طور کلی مشخصات دانشجو(یا هر اطلاعات دیگه ای رو) رو به جای اینکه هر دفعه تو برنامه تایپ کنیم یک بار تو یه فایل متنی ذخیره کنیم و دیگه هر دفعه از اون استفاده کنیم تازه آخه کدوم دانشگاه رو دیدی که فقط **100** نفر دانشجو داشته باشه عملا این برنامه باید کار رو واسه ی **500** نفر به بالا انجام بده بعد تازه تصورش رو بکن یه تایپیست بشینه همه رو تایپ کنه بعد یه دفعه یه **run time error** یا هر مشکل دیگه ای

تو برنامه پیش بیاد اون موقع هست که جون برنامه نویس به خطر می یفته تازه در بیشتر موارد این اطلاعات فقط برای یه برنامه استفاده نمیشه

مثلا تو یه برنامه ؛ با فایل متنی اطلاعات دانشجو ها و نمرات کار میکنه و کارنامه صادر می کنه یه برنامه با همین دو تا فایل (اطلاعات و نمرات) کار میکنه و دانشجو ها رو بر اساس معدل به ترتیب تو یه لیست می چینه یه برنامه ی دیگه با همین دو تا فایل کار میکنه و بر حسب معدل به دانشجوها رتبه ی **A_F** میده و .....

خب این دو تا فایل تکراری رو بدیم آخر ترم هی تایپیست تو این برنامه بنویسه بعد تو برنامه ی بعدی وارد کنه ؟؟؟؟ نه راه منطقی و درستش استفاده از فایل هاست

حالا اینا مزایای فایل ها برای ورود اطلاعات هست بحث و کاربرد مهمتر فایل ها برای ذخیره ی ثابت و طولانی مدت خروجی برنامست همون طوری که حتما میدونید خروجی حاصل از برنامه توسط پاسکال روی مانیتور نمایش داده میشه و این خروجی ها روی حافظه ی رم کامپیوتر ذخیره شدن یعنی اگه کامپیوتر رو خاموش کنین همه چی پاک میشه نه تنها خروجی برنامه ی شما بلکه همه ی اطلاعات موجود روی رم و کاربرد دیگه ی فایل ها هم واسه رفع همین مشکله ما میتونیم خروجی رو روی یه فایل متنی تو هارد ذخیره کنیم و همیشه به اون دسترسی داشته باشیم که این خیلی واسه ما مهمتره مثلا تو همون برنامه دانشجوها خب بعد از این که برنامه کارنامه رو صادر کرد ، دانشجو ها رو بر اساس نمرات مرتب کرد ، به همشون رتبه داد و هر کار دیگه خب معمولا ما نیاز داریم این نتیجه رو یه جا ذخیره کنیم تا بعدا سر فرصت ازش پرینت بگیریم و رو برد دانشگاه نصب کینم و یا لازمه کارنامه ی هر فرد رو همون موقع که برای دریافتش مراجعه کرد پرینت کنیم و بهش بدیم برای همچین کارایی بهترین راه استفاده از فایلهاست.

پس من به طور خلاصه مزایای استفاده از فایل ها رو همین پایین لیست میکنم (البته همه رو بالا گفتم ولی منسجم تر به دور از غضنفر باشه بهتره) :

1) یک بار تایپ کن **n** بار تو یه برنامه و **n** بار تو **n** برنامه استفاده کن .

2) کاهش احتمال خطا (اگه مجبور باشین واسه **n** برنامه **n** بار تایپ کنین احتمال خطا بالاتره یا یه بار با دقت تایپ کنین و بعد استفاده کنین ؟؟؟)

3) ذخیره ی دائم اطلاعات رو حافظه های جانبی از جمله هارد .

4) (این یکی جدیده) عدم نیاز به پاسکال برای دیدن خروجی (فایل متنی رو میتونید تو هر کامپیوتری باز کنید)

5) صرفه جویی در وقت و انرژی تایپیست ها

6) بیمه ی برنامه نویس در صورت کشف خطا در برنامه

و

.

.

.

خب مفصل مزایای فایل ها رو گفتم تا از این به بعد رو مشتاقانه بخونید و بدونید دارید یکی از پرکاربرد ترین مطالب رو یاد میگیرین .

فایل های متنی(**Text Files**) :

یک فایل متنی ، مجموعه ای از کاراکترها است که تحت یک نام روی دیسک ذخیره میشوند . ما میتوانیم کلیه داده هایی که توسط برنامه پردازش میشوند را قبل از اجرای برنامه در یک فایل ذخیره کنیم ، سپس به برنامه دستور خواندن داده ها را از فایل متن بجای صفحه کلید بدهیم . فایل ورودی فایلی است که شامل اطلاعات ورودی برای برنامه باشد.

یکی از مزایای فایل داده متنی این است که میتوانید با استفاده از یک برنامه ویراستار به آن دسترسی پیدا کنید و خطاها را قبل از اینکه فایل توسط برنامه مورد پردازش قرار گیرد ، تصحیح کنید . میتوانید به برنامه دستور نوشتن نتایج خود را در یک فایل متن ، بجای صفحه نمایش بدهید که این کار ، نسخه ثابتی از نتایج برنامه روی دیسک به شما میدهد و سپس فایل روی دیسک میتواند به عنوان یک فایل داده برای همان برنامه و یا یک برنامه دیگر مورد استفاده قرار گیرد.

یک فایل خروجی فایلی است که شامل نتایج برنامه است یعنی یک برنامه اطلاعات مورد نیاز خود را در آن فایل ثبت مینماید.

در زبان پاسکال **Text** نوعی است که برای تعریف پرونده ها بکار میرود . پرونده هایی که بصورت **Text** اعلان میگردند بصورت خطوطی از کاراکترها سازماندهی میشوند . بعبارت دیگر یک پرونده متنی پرونده ای است که از تعدادی کاراکتر بصورت جملات و یا کلمات تشکیل شده است

که هر خط از آن پرونده با علامتی خاص که علامت انتهای خط است پایان میپذیرد.

توجه : در ویندوز بیشتر دو فرمت **txt** و **dat** بیانگر فایل های متنی هستند .

معرفی یک پرونده متنی :

قبل از کار با پرونده **Text** باید شناسه ای برای آن تعریف شود.

```
Var
شناسه : Text ;
```

نام های خارجی و داخلی پرونده های متنی :

پیش از استفاده از پرونده درون برنامه ، لازم است پیوندی بین برنامه و پرونده ای که باید مورد پردازش قرار گیرد ایجاد نمایید . درون برنامه از یک شناسه معتبر برای نام پرونده استفاده میشود که آنرا نام داخلی پرونده میگویند .

خارج از برنامه باید از نام فایل با قوائد نامگذاری فایلهای **Dos** استفاده کنید . این نام نام خارجی پرونده نامیده میشود . بنابراین از آنجائیکه یک پرونده متنی باید روی دیسک با نام خاصی که خودمان تعیین میکنیم (نام خارجی) ذخیره گردد لذا نیاز به دستوری است که نام فایل روی دیسک را (نام خارجی) به متغیر پرونده معرفی شده در داخل برنامه (نام داخلی) نسبت دهد . برای اجرای این کار از رویه یا دستور **Assign** استفاده میکنیم .

```
Assign ( نام داخلی پرونده , ’ نام خارجی پرونده ‘ ) ;
```

مثال :

```
Assign ( F1 , ‘ C:\tp\Student.Dat’) ;
```

خواندن و نوشتن پرونده ها :

در زبان پاسکال پرونده ها به شکل پیوسته هستند یعنی اطلاعات باید از اول پرونده و بصورت متوالی خوانده و نوشته شوند . کلیه پرونده ها قبل از پردازش (خواندن و نوشتن) باید آماده شوند

برای آماده سازی پرونده ورودی از دستور **Reset** و برای پرونده های خروجی از دستور **Rewrite** استفاده میشود.

دستورالعمل**Reset** :

بعد از معرفی متغیر از نوع فایل متنی یه اسم به اون نسبت میدیم (توسط دستور**Assign**) ، برای آماده کردن فایل از رویه و یا دستور **Reset** استفاده می کنیم. دستورالعمل **Reset** فایل دیسک را باز نموده و آنرا بعنوان یک فایل ورودی آماده میسازد . در مورد پرونده ای که بعنوان ورودی باز شده است صرفا میتوان از دستورات ورودی استفاده کرد . چنانچه روی یک پرونده متنی ورودی چیزی نوشته شود ، موجب بروز خطای **I/O (Input/Output)** خواهد شد.

نکته : دستور **Reset** برای فایلهایی که بر روی دیسک وجود دارند قابل استفاده است.

دستور **Reset** یک اشاره گر پرونده را در اول یک پرونده قرار میدهد در نتیجه خواندن اطلاعات از اول فایل شروع شده و از آنجا به سمت جلو حرکت خواهد کرد.

مثالا دستور زیر فایل **f1** رو برای خوندن باز میکنه .

Reset (F1) ;

دستورالعمل های **Rewrite** و **Append :** این دو دستورالعمل یک پردازنده متنی را برای خروجی ما به گونه های مختلف آماده مینمایند ، چنانچه دستورالعمل یا رویه **Rewrite** در مورد یک پرونده متنی موجود اجرا شود محتوای آن پرونده پاک شده و اشاره گر پرونده در ابتدای پرونده قرار داده میشود یعنی پرونده آماده برای نوشتن است ، حال اگر پرونده موجود نباشد ، دستورالعمل **Rewrite** یک پرونده یا فایل با نام ذکر شده در دستورالعمل **Assign** تشکیل میدهد و سپس برای عمل نوشتن و یا عمل خروجی دیگر ، آنرا آماده میسازد.

دستور **Append** نیز پرونده را بصورت خروجی باز میکند . این دستورلعمل محتوای پرونده را حفظ کرده و اشاره گر پرونده را در آخر پرونده قرار میدهد درنتیجه به انتهای پرونده میتوان اطلاعات را اضافه نمود.

نکته : دستور **Append** نمی تواند در مورد پرونده هایی که وجود ندارند بکار رود.

دستورالعمل **Close :** از این دستورالعمل برای بستن پرونده های ورودی یا خروجی استفاده میشود . پس از استفاده از پرونده ها حتما باید آنها را بست .
وقتی که پرونده ها توسط دستورالعمل **Close** بسته شد ، دیگر نمیتوان از آن برای ورودی یا خروجی استفاده کرد مگر اینکه توسط دستورات ورودی یا خروجی مجددا باز شود . البته پس از **Close** پیوند میان نام متغیر پرونده و نام پرونده روی دیسک باقی خواهد ماند و نیازی به اجرای مجدد دستورالعمل **Assign** نخواهد بود.

خواندن محتویات یک پرونده متنی :

استفاده **Readln** و **Read** اطلاعات موجود در پرونده متنی از دستورالعملهای برای خواندن میشود .

Read (پرونده ورودی , متغیرها) ;
Readln (پرونده ورودی , متغیرها) ;
مثال : Read (F1 , a) ;
Readln (F1 , a);

خواندن چند رشته از روی یک خط :

Readln (F1 , a , b , c) ;

توابع **Eoln** و **Eof** برای فایلهای متنی:

تابع **Eof (End Of File)** انتهای یک پرونده متنی را مشخص میکند .

Eof (نام پرونده مورد نظر) ;

میتونید برای اینکه کی به آخر فایل میرسین خیلی راحت از این تابع تو یه شرط **if** استفاده کنید .
تابع **Eoln (End Of LiNe)** انتهای یک خط را در یک پرونده متنی مشخص میکند . اگر اشاره گر پرونده به علامت انتهای خط اشاره کند ارزش تابع **Eoln** درست بوده و درغیراینصورت مقدار نادرست رابرمیگرداند.

Eoln (نام پرونده مورد نظر) ;

نوشتن در پرونده های متنی :

با استفاده از دو دستور **Rewrite** و **Append** میتوانید یک پرونده متنی را بعنوان خروجی آماده سازید .

برای نوشتن روی یک پرونده متنی میتوانید از دستورالعملهای **Write** و **Writeln** استفاده نمایید.

Write (پرونده , متغیرها) ;
Writeln (پرونده و متغیرها) ;
مثال : **Write (F1 , a) ;**
Writeln (F1 , a) ;

نکته : در دستورالعملهای **Write** و **Writeln** پس از متغیر پرونده میتوانید هر نوع گزاره را قرار دهید (اعداد صحیح و اعشاری ، کاراکتر ها و رشته ها) . بدین ترتیب اگر پرونده مورد نظر با دستور **Rewrite** باز شده باشد اشاره گر پرونده در ابتدای پرونده قرار میگیرد و درصورتیکه پرونده پیش از این دارای محتویاتی بوده باشد کلیه محتویات از بین رفته و عبارت موردنظر در ابتدای آن نوشته میشود . چنانچه پرونده قبلا موجود نباشد با دستورالعمل **Rewrite** روی دیسک بوجود خواهد آمد و عبارت ذکر شده در دستور **Writeln** در ابتدای آن نوشته میشود.

برای دستیابی به پرونده های متنی می توان از روش دستیابی ترتیبی استفاده نمود که این روش تنها روش ممکن برای پرونده های متنی است . به خاطر ترتیبی بودن ساختار پرونده می توانید هر یک از داده ها را به ترتیب بنویسید و یا بخوانید بنابراین در هر لحظه تنها به یک قلم از داده های پرونده دستیابی دارید . بعنوان مثال در دستیابی ترتیبی برای اینکه بخواهیم به اطلاعات رکورد خاصی در فایل دستیابی داشته باشیم لازم است که ابتدا رکوردهای قبلی را بخوانیم تا به رکورد مورد نظر برسیم.

نکته : در زبان پاسکال پرونده های دارای نوع و فاقد نوع نیز وجود دارند که به پرونده های تصادفی مشهورند یعنی رکوردهای موجود در آنها ممکن است بصورت پشت سرهم خوانده نشوند.

کارگاه حل تمرین فایلها :

مثال **1** : برنامه‌ای بنویسید که از ورودی **10** عدد دریافت کند و آنها را به ترتیب هر کدام دریک خط ازیک فایل خروجی بریزد و در انتها تعداد و مجموع آنها را بنویسد.

```
Var
        f: Text;
        A: Array [ 1.. 10] of integer;
        Sum, i: integer;
Begin
        Sum: = 0;Writeln (' Enter 10 numbers: ');
        For i: = 1 to 10 do begin
              Read (A [i]);
              Sum: = sum + A[i];
        end;
        Assign (f, 'out.dat');
         Rewrite (f);
     For    i: = 1     to    10     do
           Writeln (f, A[i]);
     Writeln (f, Sum);
     Close (f);
End.
```

مثال **2** : بافرض موجود بودن فایل متنی **'test.dat'** می‌خواهیم اعداد داخل فایل را که پنج تا هستند را از فایل خوانده باهم جمع کنیم :

```
Program    usefile ;
Var
        f: Text;
        str1, str2: string; a, b, c, d, e: integer;
begin
        Assign (f, ' Test. dat ');
        Reset (f);
        Readln (f, str1);
        Readln (f, a, b, c);
```

```
        Readln (f, d, e, str2);
        Write (str2, ': ' , a + b + c + d + e );
        Close (f);
End.
```

مثال **3** : برنامه‌ای بنویسید که یک فایل متنی شامل چند جمله را از ورودی دریافت کرده و یک کپی از فایل درخروجی بسازد.

```
var
     f1,f2: Text;
     str1,str2 : string ; ch : char ;
begin
    write('Enter   Input file: ');
    readln(str1);
    write('Enter output (copy) file: ');
    readln(str2);
    assign(f1,strl);
    reset(f1);
    assign(f2,str2);
    rewrite(f2);
   while  not  eof ( f1 )  do
   begin
        while  not  eoln( f1 )  do
         begin
              read(f1,ch);
              write(f2,ch);
         end;
         readln(f1);
         writeln(f2);
    end;
    Close(f1);
    Close(f2);
End.
```

مثال **4** :

```
Var
      Bf2: file of integer;
      Bf2: file of integer;
      Bf1: file of char;
```

فایل دودویی از نوع صحیح

فایل دودویی از نوع صحیح

فایل دودویی از نوع کاراکتر

Bf6: file of set of 'A'..'Z'; فایل دودویی از مجموعه **A** تا **Z**

مثال **5** : برنامه‌ای بنویسید که تعداد رشته (حداکثر **10** تایی) از ورودی خوانده و در یک فایل دودویی بنویسید.

```
Const n= 10;
Var
     Bf: file of strin[10]; Str: string [10];
     i: integer;
Begin
    Assign (Bf,'out.dat');
    Rewrite (Bf);
    For i:=1 to n do  Begin
          Readln (str);
          Write (Bf,str);
    End;
    Close (Bf);
End.
```

مثال **6** : برنامه‌ای بنویسید که تعداد خطوط یک فایل متنی را بدست آورد.

```
Var
     F: text ;
     ch : char ;
     str: string[20];
     count: integer;
Begin
    Write ('Enter  the  file   name: ');
     Readln(str);
     Assign(f,str);
     Reset(f);
      count:=0;
      While not eof(f) do
      begin
            While not eoln(f) do
                  Read(f,ch);
            Readln(f);
            count:=count+1;
      end;
     Write('the number of lines in file is: ',count);
```

```
End.
```

مثال 7 : رکوردی از کتاب‌ها در نظر گرفته، تعداد10 کتاب را در یک فایل نوع دار ذخیره کنید.

```
Const
    n = 10;
Type
    Booktype = Record
    name: string[20];
    ID: integer;
 end;
Var
    Bf: file of Booktype;
    I , ID : integer;
    name: string[20];
    Book: Book type;
Begin
    Assign (Bf,'book.dat');
    Rewrite (Bf);
    For  I := 1   to  n   do
    begin
          Write ('Enter the name & ID of book: ');
           Readln (name, ID);
           Book.name:= name;
           Book.ID:= ID;
           Write (Bf, Book);
    End;
    Close (Bf);
End.
```

****

آخر خط :

خب این قسمت هم تموم شد و عملا پاسکال از نظر کاربردی تموم شد قبول دارم واسه این سه فصل آخر باید بیشتر وقت میذاشتم و بهتر توضیح میدادم شرمنده ولی به دلیل گرفتاری های جورواجور که هر روزم بیشتر میشه وقت نشد بی تعارف میگم اگه از سر و ته این سه فصل آخر یه ذره کم نمیکردم کتاب تموم نمی شد و همون جوری نصفه کاره گوشه ی درایو f کامپیوترم ویروس میگرفت و کپک میزد .

قرار بر همکاری کل بچه های کامپیوتر دانشگاه بود ولی افسوس که فقط یکی از دانشجو ها تو این راه منو کمک کرد .

درضمن از فرصت استفاده میکنم و از تمامی عزیزانی که علاقه دارند با تالیف کتاب به صورت الکترونیک و یا ساخت هرگونه نرم افزار آموزشی به صورت رایگان گامی در جهت افزایش سطح معلومات هموطنان خودشون بردارند دعوت میکنم تا از طریق آدرس

babak.cyclops@gmail.com

آمادگی خودشون رو به این حقیر اطلاع بدهند .

ضمیمه ها :

چند دستور اضافی اما کاربردی :

دستور **gotoxy** :

جهت دادن مختصات چاپ بکار می رود .

دستور **CLRscr** :

جهت پاک کردن صفحه ی نمایش معمولا **Begin** بعد از قرار می گیرد .

دستور **Textbackground** :

با استفاده از این دستور می توانیم رنگ زمینه را با استفاده از نام خود رنگ ها تغییر دهیم .(توجه : این دستور در توربو پاسکال تحت ویندوز کار نمی کند)

دستور **Textcolor** :

جهت تغییر رنگ نوشته به کار می رود(توجه : این دستور در توربو پاسکال تحت ویندوز کار نمی کند)

از **blink** همچنین جهت چشمک زدن متن استفاده می شود .

کلی مثال و تمرین :

در این قسمت هرچی مثال و برنامه تو کامپیوترم دارم براتون میزارم شاید واسه یه نفر مفید باشه اکثرش حاصل گشتن و **search** های خودم تو اینترنته که از سایت ها و وبلاگهای مختلف پیدا کردم که اسم اونایی رو که داشتم تو قسمت منابع و ماخذ آخر کتاب نوشتم تا مدیون کسی نمونم .
امیدوارم واستون مفید باشه :

مثال **1** :

برنامه ای بنویسید که دو عدد را از ورودی دریافت و چنانچه عدد اول در بازه **0** تا **5** بود توان دوم عدد دوم را چاپ کند چنانچه **68** بود حاصلضرب دو عدد را چاپ کند و در غیر اینصورت مقادیر مجاز را چاپ کند.

```
BEGIN
WRITELN('ENTER TWO NUMBER');
READLN(A,B);
CASE A OF
0..5 :WRITELN(B*B);
68:WRITELN(A*B)
ELSE
WRITELN('ENTER 0..5 OR 68');
END;
END.
```

معادل برنامه بالا با دستور **If** :

```
Begin
Writeln('enter 2 number');
Readln(a,b);
If (a>=0)and(a<=5) then
Writeln(b*b)
Else
If a=68 then
```

```
Writeln(a*b)
Else
Writeln('enter 0..5 or68');
End.
```

مثال **2** :

برنامه ای بنویسید که نمرات **100** دانش آموز را از ورودی دریافت و در یک آرایه بریزد.

```
Var
A:array[1..100] of real;
Begin
For i:=1 to 100 do
Readln(a[i]);
```

مثال **3** :

برنامه ای بنویسید که نمرات **100** دانش آموز را از ورودی دریافت و **50** تای اول را چاپ کند سپس تعداد نمرات **20** را مشخص کند.

```
Var
A:array[1..100] of real;
Begin
For I;=1 to 100 do
Writeln('enter no(',I,')');
Readln(a[i]);
For i:=1 to 50 do
Writeln(a[i]);
S:=0;
For i:=1 to 100 do
If i:=1 to 100 do
If a[i]=20 then
S:=s+1;
```

```
Writeln(s);
End.
```

مثال 4 :

برنامه ای بنویسید که با دریافت نمرات 100 دانش آموز بالاترین و پایین ترین نمره را در بین این 100 نفر پیدا کند.

```
Var
A:array[1..100] of real;
I:integer;
Begin
For i:=1 to 100 do
Readln(a[i]);
Min:=a[1];
Max:=a[1];
For i:=2 to 100 do
Begin
If a[1]< min then
Min:=a[i];
If a[i]>max then
Max := a[i];
End;
Writeln('max is :',max,'min is :',min);
End.
```

مثال 5 :

برنامه ای بنویسید که با دریافت 100 اسم به ما بگوید تعداد تکرار اسم ali چند بار است وآیا اسم hassan بیشتر ذکر شده یا اسم reza ؟

```
Var
A:array [1..100] of string;
Begin
```

```
For i:=1 to 100 do
Readln(a[i]);
Ali:=0;
Reza:=0;
Hassan:=0;
For i:=1 to 100 do
Begin
If a[i]='ali' then
Ali:=ali+1;
If a[i]='reza' then
Reza:=reza+1;
If a[i]='hassan' then
Hassan:=hassan+1;
Endd
Writeln(ali is :',ali);
If reza>hassan then
Writeln('reza')
Else
Writeln('hassan');
End.
```

مثال **6** :

برنامه ای بنویسید که با دریافت نام و نمره **100** دانش آموز تنبل ترین فرد کلاس را مشخص کند.

```
Var
Name:array[1..100] of string;
No:array[1..100] of real;
Begin
For i:=1 to 100 do
Begin
Readln(name[i]);
Readln(no[i]);
End;
Min:=no[1];
Namemin:=name[1];
```

```
For i:=2 to 100 do begin
If no[i]
Begin
Min:=no[i];
Namemin:=name[i];
Writeln(namemin);
End.
```

مثال **7** :

برنامه ای بنویسید که نمرات **4** کلاس که هر یک **20** دانش آموز دارد را از ورودی دریافت وتنبل ترین فرد را در این **4** کلاس مشخص کند.

```
Var
A:array[1..20,1..4] of real;
Begin
For i:=1 to 4 do
Begin
For j:=1 to 20 do
Readln(a[j,i]);
End;
Min:=a[1,1];
For i:=1 to 4 do
For j:=1 to 20 do
If a[j,i]< min then
Min:=a[j,i];
Writeln(min);
End.
```

مثال **8** :

در یک آرایه **100*20**دوهزار کاراکتر نگهداری میشود برنامه ای بنویسید که تعداد تکرار حرف **A** را با **a** مقایسه کند و همچنین به ما بگوید آیا کاراکتر **z** در این ماتریس وجود دارد ؟

```
Var
A:array[1..20,1..100] of character;
Bool:Boolean;
B,b1:integer;
Begin
For i:=1 to 100 do
For j:=1 to 20 do
If a[I,j]='A' then
B:=b+1;
If a[I,j]='a' then
B1:=b1+1;
If a[I,j]='z' then
Bool:=true;
End;
If b>b1 then writeln('A>a');
If b<a?);
If b=b1 then writeln('A=a');
If true then writeln('''z''exist');
End.
```

مثال **9** :

میخواهیم نام و نمره تعدادی دانش آموز یک کلاس را از ورودی دریافت کنیم و مشخص کنیم که تنبل ترین و زرنگ ترین فرد کلاس کیست.چند نفر نمره زیر **10** گرفتند؟نام کسانی که نمره آنها در بازه **17** تا **15** می باشد ب همراه نمره آنها چاپ شود آخرین نفر نامش **end** است حداکثر تعداد دانش آموزان **100** نفر است.

```
Var

a:array [1..100] of string ;
b:array[1..100] of real;
begin
i:=1;
read(a[i]);
while a[i]<>'end' do begin
readln(b[i]);
```

```
i:=i+1;
readln(a[i]);
end;
j:=i-1;
min:=a[1];
max:=a[1];
for i:= 2 to j do
begin
if b[i] >max then
begin
max:=b[i];
name:a[i];
end;
end
writeln(name);
count:=0;
for i:=1 to j do
if b[i]<=10 then
count:=count+1;
writeln(count);
for i:=1 to j do
if (b[i]<17) and (b[i]>15) then
write(b[i],a[i]);
end.
```

مثال **10** :

برنامه ای بنویسید که حاصل این عبارت را حساب کند.

i/i!=1/1!+2/2!+3/3!+4/4!+5/5!

N!= 1*2*3*4*…*n

```
Begin
Sum:=0;
For n:=1 to 5 do
Begin
For m:=1 to n do
F:=f*m;
```

```
Sum:=sum+n/f;
End;
Writeln(sum);
End.
```

مثال **11** :

برنامه ای بنویسید که نمرات **80** دانش آموز را که در چهار کلاس دسته بنده شده اند را از ورودی دریافت کند سپس به سوالات زیر جواب دهد.

1-معدل هر یک از کلاسها

2-نمره تنبل ترین و زرنگترین شخص در هر کلاس به طور جداگانه

```
Var
A:array [1..4,1..20] of real;
Begin
For i:=1 to 4 do
For j:=1 to 20 do
Begin
Writeln('please enter nomreh');
Readln(a[I,j]);
End;
For i:=1 to 4 do
Begin
Sum:=0 ;
For j:=1 to 20 do
Begin
Sum:=a[I,j]+sum;
End;
Writeln(sum/20,'average of this class');
End;
For i:=1 to 4 do
Begin
Max :=a[1,1];
Min:=a[1,1];
For j:=2 to 20 do
```

```
Begin
If a[I,j]>max then
Max :=a[I,j];
End;
Writeln(I,min,max);
End;
End.
```

مثال 12 :

برنامه ای بنویسید که یک رشته (متن) را از کاربر بگیرد و آخرین حرف رشته را ستاره بگذارد.

```
Program exam_12;
var
i:integer;
s:string;
begin
string writeln('enter');
readln(s);
i:=lenght(s);
s[i]:='*';
writeln(s);
end.
```

مثال 13 :

برنامه ای بنویسید که عدد از کاربر بگیرد و بگوید که عدد متقارن است و اگر نبود بگوید که عدد متقارن نیست

```
Program exam_13;
var
i,len:integer;
s:string;
begin
number writeln('enter');
readln(i);
str(i,s);
len:=lenght(s)
```

```
then 0=2 if len mod
begin
write('عد متقارن نیست');
halt;
;end
len/2 do for i:=1 to
begin
s[len-i+1] then<>[if s[i
begin
writeln('عد متقارن نیست');
halt
end
writeln('عد متقارن هست');
End.
```

مثال 14 :

برنامه ای بنویسید که یک رشته (متن) را از کاربر بگیرد و طول رشته (تعداد حروف آن) را چاپ کند .

```
program exam_14 ;

var
s:string;

begin;

readln(s);
length(s , ' = ' , writeln( s) );

end.
```

مثال 15 :

برنامه ای بنویسید که سانتی متر را بگیرد و بر حسب اینچ چاپ کند .

cm => inch

اطلا عات مورد نیاز :

2.54 * inch = 1cm1

```
Program exam_15 ;
var
n:real;

begin;
number writeln('enter one real:');
readln(n);

n:=n*2.54;

writeln('n=' , n);
end.
```

مثال **16** :

با استفاده از دستور **for** برنامه ای بنویسید که مرتبا حروف از کاربر بگیرد و اگر کاراکتر مورد نظر **Q** بود برنامه خاتمه پیدا کند .

```
program exam_16 ;

var
i:integer;
c:char;

begin
character writeln ('Enter');

for i:=1 to 28 do
begin
readln (c);
```

```
c='Q' then if
halt;
end;
end;

end.
```

مثال 17 :

برنامه ای بنویسید که مرتبا حروف را از کاربر بگیرد و اگر کاراکتر مورد نظر **Q** بود از برنامه خارج شود

```
program exam_17;
var
    char:c;

begin
clrscr;
readln(c);

do 'a'<>while c
begin
readln(c);
end;

end;
```

مثال 18 :

برنامه ای بنویسید که **N** را از ورودی بگیرد و نتیجه این سری را محاسبه کند .

N+....+4+3+2+1

```
program exam_18;

var

i,m,n : integer;

begin
```

```
number writeln('Enter one');
readln(n);
m:=0;
for i:=1 to n do
begin
m:=m+i;
end;
number:',m writeln('Enter one);
End.
```

مثال 19 :

برنامه ای بنویسید که اسامی ۱۰ نفر را بگیرد و آن کدام که حرف اولش (با توجه به ترتیب حروف الفبای انگلیسی) بزرگتر است را چاپ کند .

```
Program exam_19 ;
var
n,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10:string;
begin
writeln('Enter Name');
readln(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10);
n then>if s1
begin
n:=s1;
```

```
end;

n then>if s2

begin

n:=s2;

end;
```

به این ترتیب تا s10 می نویسیم

```
writeln('n=',n);

end.
```

مثال **20** :

در این مثال که از دستور **while** کمک گرفتیم کامپیوتر مرتبا' یک حرف را از کاربر می گیرد و اگر آن حرف برابر **q** بود متوقف می شود .

```
program exam_20;
var
char : c;

begin
clrscr;
readln(c);

do 'Q'<>while c
begin
   readln(c);
end;

end.
```

مثال **21** :

برنا مه ای بنویسید که یک عدد بگیرد و فاکتوریل آن را حساب کند و در آخر نتیجه را نمایش دهد .

```
program faktoriel;
var
fact,n,i : longint;

begin
writeln('enter an integer number)';
readln(n);
fact:=1

n do for i:=1 to
    fact:=fact*i
end;

writeln('factorial=',fact);
end.
```

مثال **22** :

در مثال زیر یک عدد از کاربر می گیریم و اگر آن عدد از **100** بزرگتر بود **yes** و اگر مساوی **100** بود **yesno** و اگر کوچکتر از **100** بود **no** را در جواب نمایش میدهد .

```
Program exam_22;

var

a:integer;

Begin

clr scr;

than  write('enter greade of our weblog more than 100 or 100 or less');
100

readln(a);

then 100<if a

begin
```

```
writeln ('yes');
end;
if a=100 then
begin
writeln ('yesno');
end;
then 100>if a
begin
writeln ('no');
end;
End.
```

مثال **23** :

این برنامه یرای مثال ابتدا **2** عدد از کاربر می گیرد و آن ها را جمع می کند ، سپس نتیجه را نمایش می دهد .

همچنین دستور **readln** داده ها را از کاربر در خط بعد می گیرد .

```
Program exam_23;
var
a:integer;
b:integer;
Begin
```

```
write('enter two numbers to add');
readln(a,b);
a:=a+b;
write ('a=',a);
End.
```

جدول اسکی :

Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20		64	40	@	96	60	'
^A	1	01		SOH	33	21	!	65	41	A	97	61	a
^B	2	02		STX	34	22	"	66	42	B	98	62	b
^C	3	03		ETX	35	23	#	67	43	C	99	63	c
^D	4	04		EOT	36	24	$	68	44	D	100	64	d
^E	5	05		ENQ	37	25	%	69	45	E	101	65	e
^F	6	06		ACK	38	26	&	70	46	F	102	66	f
^G	7	07		BEL	39	27	'	71	47	G	103	67	g
^H	8	08		BS	40	28	(	72	48	H	104	68	h
^I	9	09		HT	41	29	)	73	49	I	105	69	i
^J	10	0A		LF	42	2A	*	74	4A	J	106	6A	j
^K	11	0B		VT	43	2B	+	75	4B	K	107	6B	k
^L	12	0C		FF	44	2C	,	76	4C	L	108	6C	l
^M	13	0D		CR	45	2D	-	77	4D	M	109	6D	m
^N	14	0E		SO	46	2E	.	78	4E	N	110	6E	n
^O	15	0F		SI	47	2F	/	79	4F	O	111	6F	o
^P	16	10		DLE	48	30	0	80	50	P	112	70	p
^Q	17	11		DC1	49	31	1	81	51	Q	113	71	q
^R	18	12		DC2	50	32	2	82	52	R	114	72	r
^S	19	13		DC3	51	33	3	83	53	S	115	73	s
^T	20	14		DC4	52	34	4	84	54	T	116	74	t
^U	21	15		NAK	53	35	5	85	55	U	117	75	u
^V	22	16		SYN	54	36	6	86	56	V	118	76	v
^W	23	17		ETB	55	37	7	87	57	W	119	77	w
^X	24	18		CAN	56	38	8	88	58	X	120	78	x
^Y	25	19		EM	57	39	9	89	59	Y	121	79	y
^Z	26	1A		SUB	58	3A	:	90	5A	Z	122	7A	z
^[	27	1B		ESC	59	3B	;	91	5B	[	123	7B	{
^\	28	1C		FS	60	3C	<	92	5C	\	124	7C	|
^]	29	1D		GS	61	3D	=	93	5D	]	125	7D	}
^^	30	1E	▲	RS	62	3E	>	94	5E	^	126	7E	~
^-	31	1F	▼	US	63	3F	?	95	5F	_	127	7F	⌂

* ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL + BKSP key.

Dec	Hex	Char
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	₧
159	9F	ƒ

Dec	Hex	Char
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	░
177	B1	▒
178	B2	▓
179	B3	│
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	║
187	BB	╗
188	BC	╝
189	BD	╜
190	BE	╛
191	BF	┐

Dec	Hex	Char
192	C0	└
193	C1	┴
194	C2	┬
195	C3	├
196	C4	─
197	C5	┼
198	C6	╞
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	█
220	DC	▄
221	DD	▌
222	DE	▐
223	DF	▀

Dec	Hex	Char
224	E0	α
225	E1	ß
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	µ
231	E7	τ
232	E8	Φ
233	E9	Θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	φ
238	EE	ε
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	n
253	FD	2
254	FE	■
255	FF	

فهرست منابع :

برنامه نویسی به زبان پاسکال نویسنده : بلفورد / لیو

Power point مبانی کامپیوتر و برنامه نویسی نویسنده : جعفر تنها / مهدی یوسف خانی

وبسایت های :

www.irandevelopers.com

www.dostan.net

www.iritn.com

وبلاگ های :

www.mobinb.persianblog.ir

www.sherman2000.brovehost.com

www.ib.persianblog.ir

پایان