

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

دانشگاه پیام نور
دانشکده فناوری
اطلاعات (گروه
مهندسی
کامپیوتر)

مبانی کامپیوتر

مؤلفین :

جعفر تنها - مهدی یوسف خانی

ارائه دهنده :

جعفر تنها

اهداف کلی این درس

دانشجو پس از مطالعه این درس باید بتواند:

الگوریتمی برای حل مسئله ارائه دهد

اصول و مبانی اولیه نرم افزار و سخت افزار را بشناسد

اهداف و مفاهیم زبان های برنامه نویسی را بداند

مفاهیم اولیه برنامه نویسی ساخت یافته را بداند و اصول لازم را در
مرحله اجراء بکار ببرد

دستورات زبان پاسکال را در برنامه ها بکار ببرد

از توابع و روال های استاندارد زبان پاسکال در صورت لزوم استفاده
نماید

- از توابع ، روال ها برای جدا کردن قطعات برنامه استفاده کند
- فایل ها را به عنوان محلی برای ذخیره داده ها در برنامه استفاده کند
- با استفاده از قابلیت های زبان پاسکال برنامه ای را برای یک سیستم بنویسد.

❖ جایگاه این درس در رشته مهندسی کامپیوتر

این درس اولین درس دانشگاهی رشته می باشد و نقطه شروعی برای ورود به دنیای جالب برنامه نویسی و علم و فن کامپیوتر هست . بنابراین یاد گیری اصول اولیه برنامه نویسی در این درس از جایگاه ویژه ای برخوردار است . این درس پایه و اساس برنامه نویسی که جزء اصول این رشته می باشد را به فرگیران یاد می دهد .

بنابراین یادگیری دقیق این درس به همراه ارائه پروژه های عملی که لازمه این درس می باشد جزء اهم مسائل می باشد .

طرح درس

توصیه می شود این درس در بیست جلسه بصورت ذیل ارائه شود :

بخش اول: (در پنج جلسه)

اهداف درس و فصل اول و دوم کتاب درسی

بخش دوم:

جلسه ششم:

فصل سوم کتاب درسی

جلسه هفتم:

فصل چهارم کتاب درسی

جلسه هشتم :

فصل پنجم کتاب درسی

جلسه نهم :

فصل ششم کتاب درسی

جلسه دهم و یازدهم :

فصل هفتم کتاب درسی

جلسه دوازدهم :

فصل هشتم کتاب درسی

جلسه سیزدهم :

فصل نهم کتاب درسی

جلسه چهاردهم :

فصل دهم کتاب درسی

جلسه پانزدهم و شانزدهم :

فصل یازدهم کتاب درسی

جلسه هفدهم :

فصل دوازدهم کتاب درسی

جلسه هیجدهم :

فصل سیزدهم کتاب درسی

جلسه نوزدهم :

فصل چهاردهم کتاب درسی

جلسه بیستم :

فصل پانزدهم کتاب درسی و

ارائه پروژه نهائی درس

الگوریتم‌ها Algorithm

هدفهای کلی

شناخت حل مسئله و ارائه الگوریتم
شناخت اجزاء لازم برای حل مسئله
بررسی صحت الگوریتم

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

الگوریتمی را برای حل مسئله ارائه دهد. ■

الگوریتم های مختلف برای یک مسئله را مقایسه کند. ■

شرط ها و حلقه ها را در الگوریتم بکار ببرد. ■

مقدمه

در زندگی روزمره، انسان با مسائل مختلفی روبروست و برای هر کدام از این مسائل (حل مشکلات) راه حلی و روشی را برمی‌گزیند. مسائلی از قبیل راه رفتن، غذا خوردن، خوابیدن و غیره که بشر تقریباً هر روز آنها را پیش روی خود دارد.

همه این مسائل نیاز به روشی برای حل کردن دارند مثلاً راه رفتن باید با ترتیب خاصی و مراحل معینی انجام شود. تا مسئله راه رفتن برای بشر حل شود. اصطلاحاً روش انجام کار یا حل مسئله را الگوریتم آن مسئله می‌نامند

❖ تعریف الگوریتم

الگوریتم مجموعه‌ای از دستورالعمل‌ها، برای حل مسئله می‌باشد که شرایط زیر را باید دارا باشد:

- دقیق باشد
- جزئیات کامل حل مسئله را داشته باشد.
- پایان پذیر باشد.



❖ مراحل الگوریتم

برای حل یک مسئله باید الگوریتم آن مسئله را مشخص کنیم (یا بیابیم). که اصطلاحاً طراحی الگوریتم برای آن مسئله نامیده می‌شود. در طراحی الگوریتم معمولاً سه مرحله زیر را از هم جدا می‌کنند:

• خواندن داده‌ها

• انجام محاسبات

• خروجی‌ها

مثال : الگوریتمی بنویسید که دو عدد از ورودی دریافت کرده مجموع دو عدد را محاسبه و چاپ نماید.

ورودیها

a , b

انجام محاسبات

جمع دو عدد

خروجیها

مجموع دو عدد

0- شروع

1- a , b را بخوان.

2- مجموع a , b را محاسبه و در sum قرار بده.

3- sum را در خروجی چاپ کن

4- پایان

مثال: الگوریتمی بنویسید که سه عدد از ورودی دریافت کرده مجموع و میانگین سه عدد را محاسبه و چاپ کند.

| <u>ورودیها</u> | <u>انجام محاسبات</u> | <u>خروجیها</u> |
|----------------|----------------------|----------------|
| a | محاسبه مجموع | چاپ مجموع |
| b | محاسبه میانگین | چاپ میانگین |

c

0- شروع

1- سه عدد از ورودی بخوان

2- مجموع سه عدد را محاسبه و در sum قرار بده.

3- sum را بر سه تقسیم کرده، در ave قرار بده.

4- ave , sum را در خروجی چاپ کن.

5- پایان.

معمولا درک یک الگوریتم با شکل راحتتر از نوشتن آن بصورت متن می باشد. لذا الگوریتم را با فلوجارت (**flowchart**) نمایش می دهند. فلوجارت از شکل های زیر تشکیل می شود.

● علامت های شروع و پایان: که معمولا از یک بیضی استفاده می کنند:



● علامت های ورودی و خروجی: که معمولا از متوازی الاضلاع استفاده می شود:



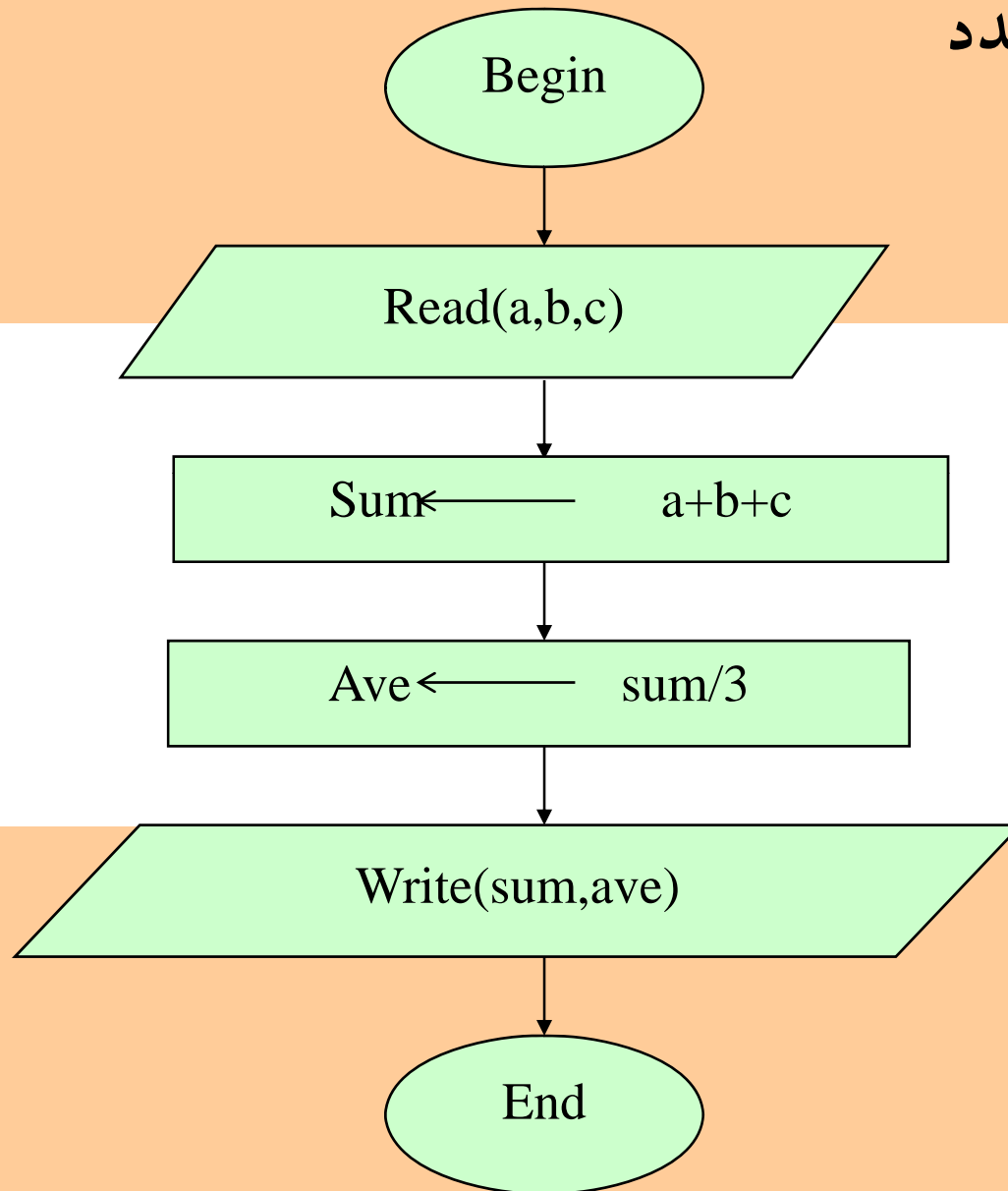
● علامتهای محاسباتی و جایگزینی: برای نمایش دستورات جایگزینی و محاسباتی از مستطیل استفاده می کنند:

جایگزین یا
محاسبات

● علامت شرط: برای نمایش شرط از لوزی استفاده می شود.

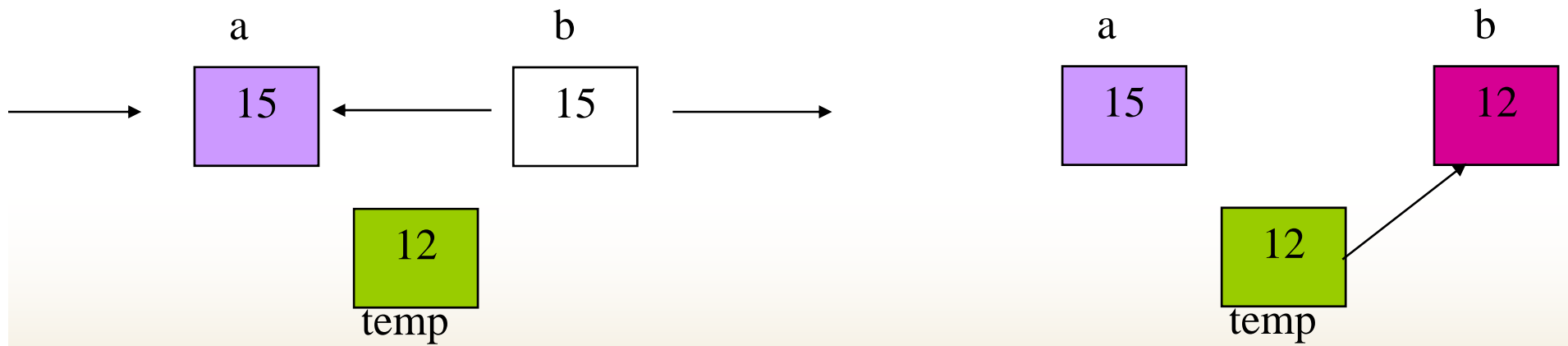
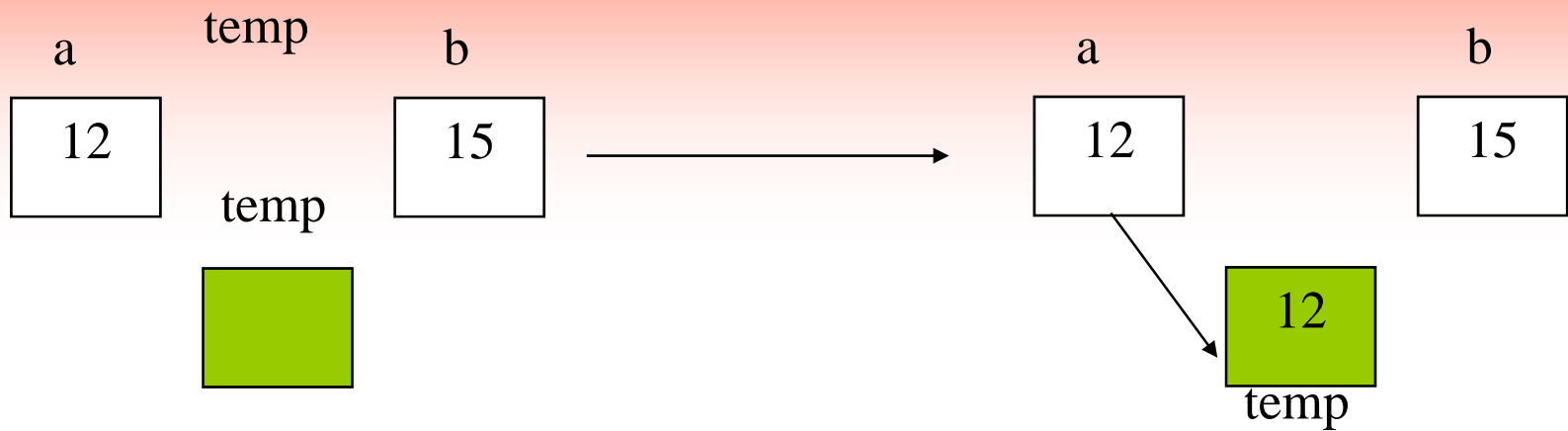
● علامت اتصال: برای اتصال شکل های مختلف بهم از فلش های جهت دار استفاده می کنند.

• فلوجارت مجموع سه عدد

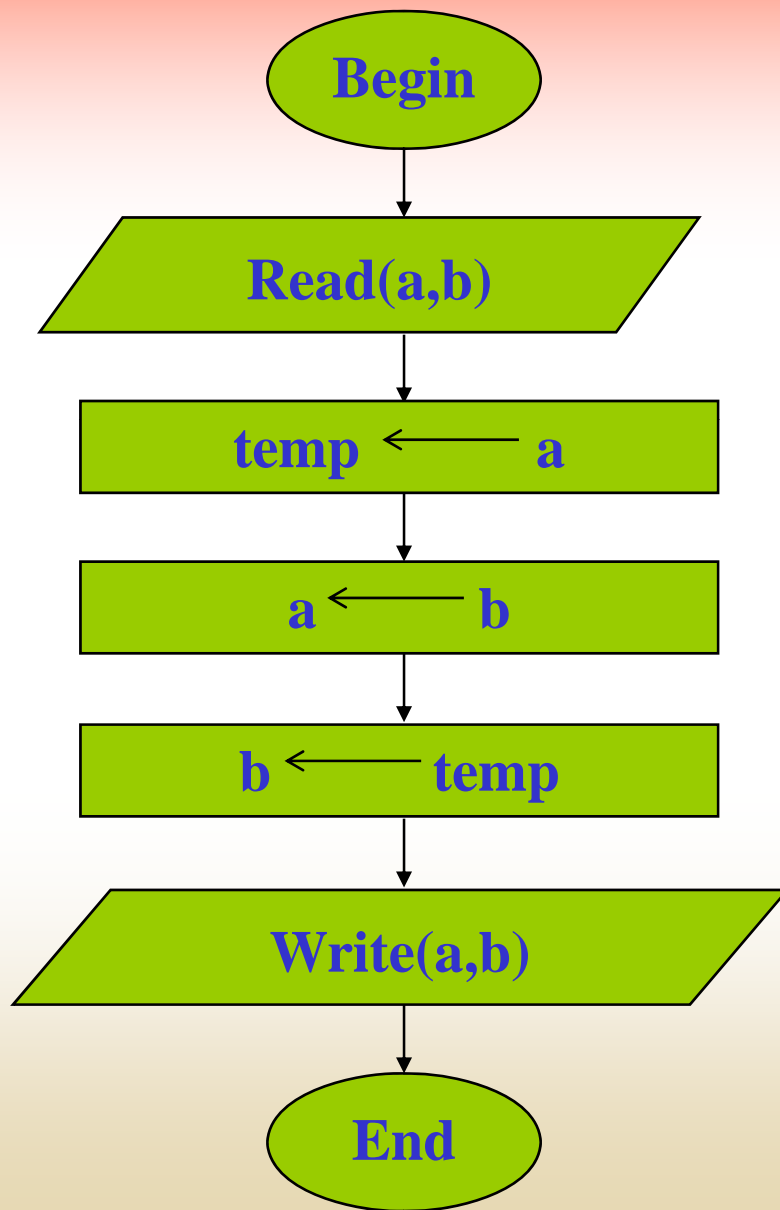


مثال: فلوجارتی رسم نمائید که دو عدد از ورودی دریافت کرده سپس محتویات دو عدد را با هم جابجا نماید.

برای حل این مسئله a , b را دو متغیر که در آنها دو عدد خوانده شده، قرار می گیرند در نظر می گیریم. سپس با استفاده از یک متغیر کمکی محتویات این دو عدد را جابجا می کنیم :



فلوچارت مسئله بالا بصورت زیر خواهد بود:



❖ تمرین

۱- فلوجارتی رسم نمائید که طول و عرض مستطیل را از ورودی دریافت کرده محیط و مساحت آنرا محاسبه و چاپ کند.

۲- فلوجارتی رسم نمائید که شعاع دایره‌ای را از ورودی دریافت کرده، محیط و مساحت آنرا محاسبه و چاپ نماید.

۳- فلوجارتی رسم کنید که سه عدد **Third , second, first** را از ورودی دریافت کرده، محتویات آنها را جابجا نموده، حاصل را در خروجی چاپ کند.

۴- فلوجارتی رسم نمائید که دو عدد از ورودی دریافت کرده، سپس محتویات دو عدد را بدون استفاده از متغیر کمکی جابجا کند.

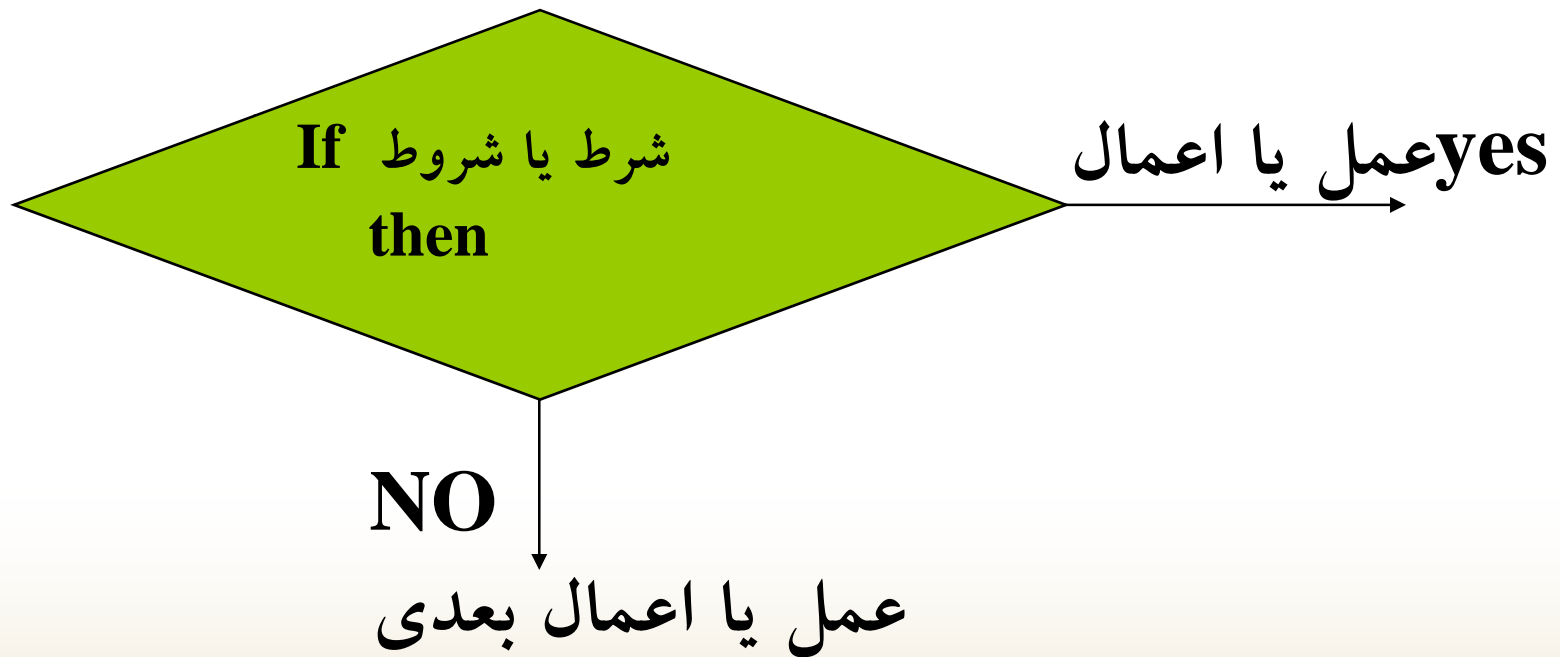
۵- فلوجارتی رسم نمائید که عددی (درجه حرارت برحسب سانتیگراد) را از ورودی دریافت کرده سپس آنرا به درجه فارنهایت تبدیل کند.

❖ دستورالعمل‌های شرطی

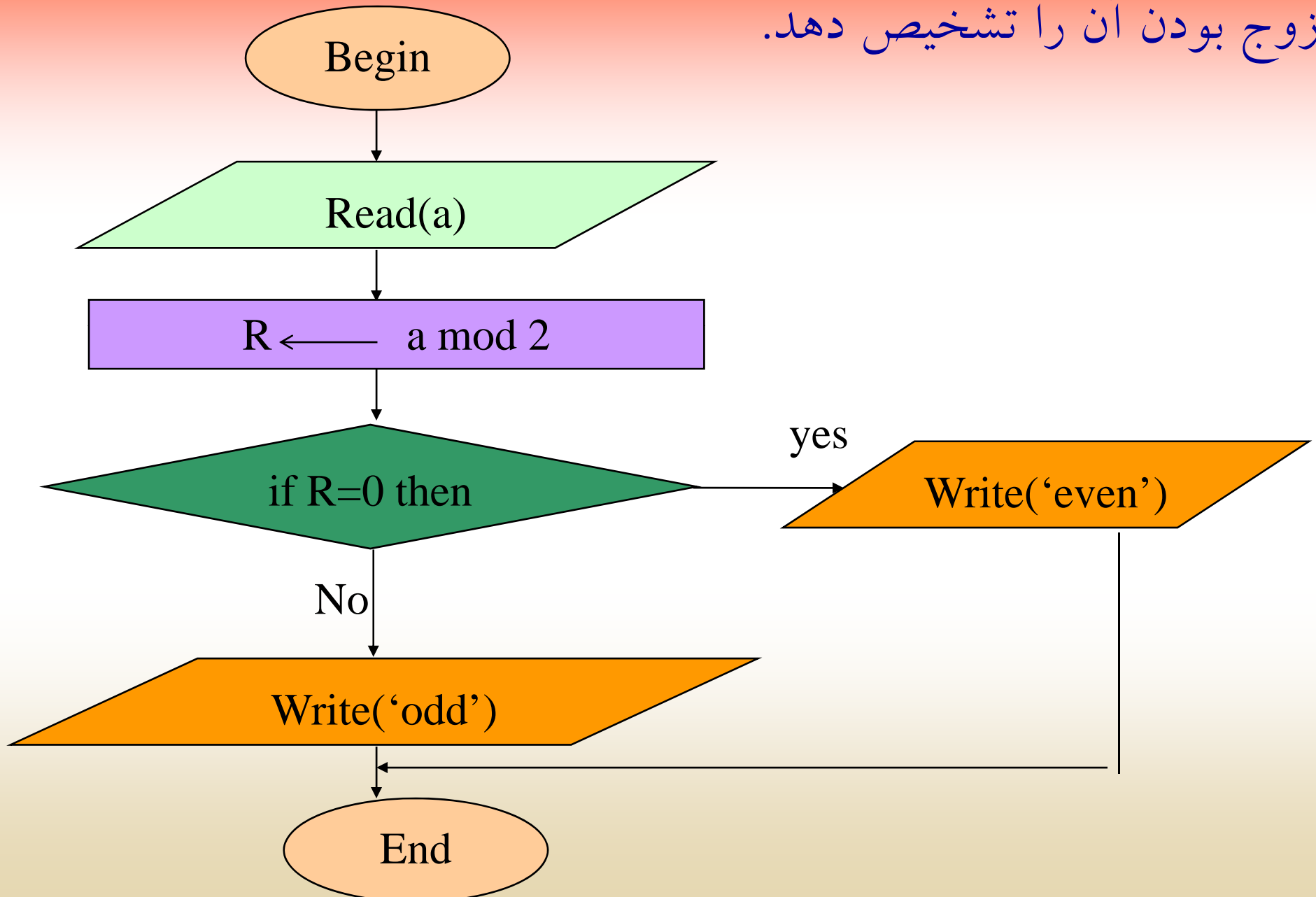
در حل بسیاری از مسائل یا تقریباً تمام مسائل نیاز به استفاده از شروط جزء، نیازهای اساسی محسوب می‌شود. همانطور که ما خودمان در زندگی روزمره با این شرط‌ها سرکار داریم. بطور مثال اگر هوا ابری باشد ممکن است چنین سخن بگوییم:

- اگر هوا بارانی باشد سپس چتری برمی‌دارم.
در غیر اینصورت چتر بر نمی‌دارم.

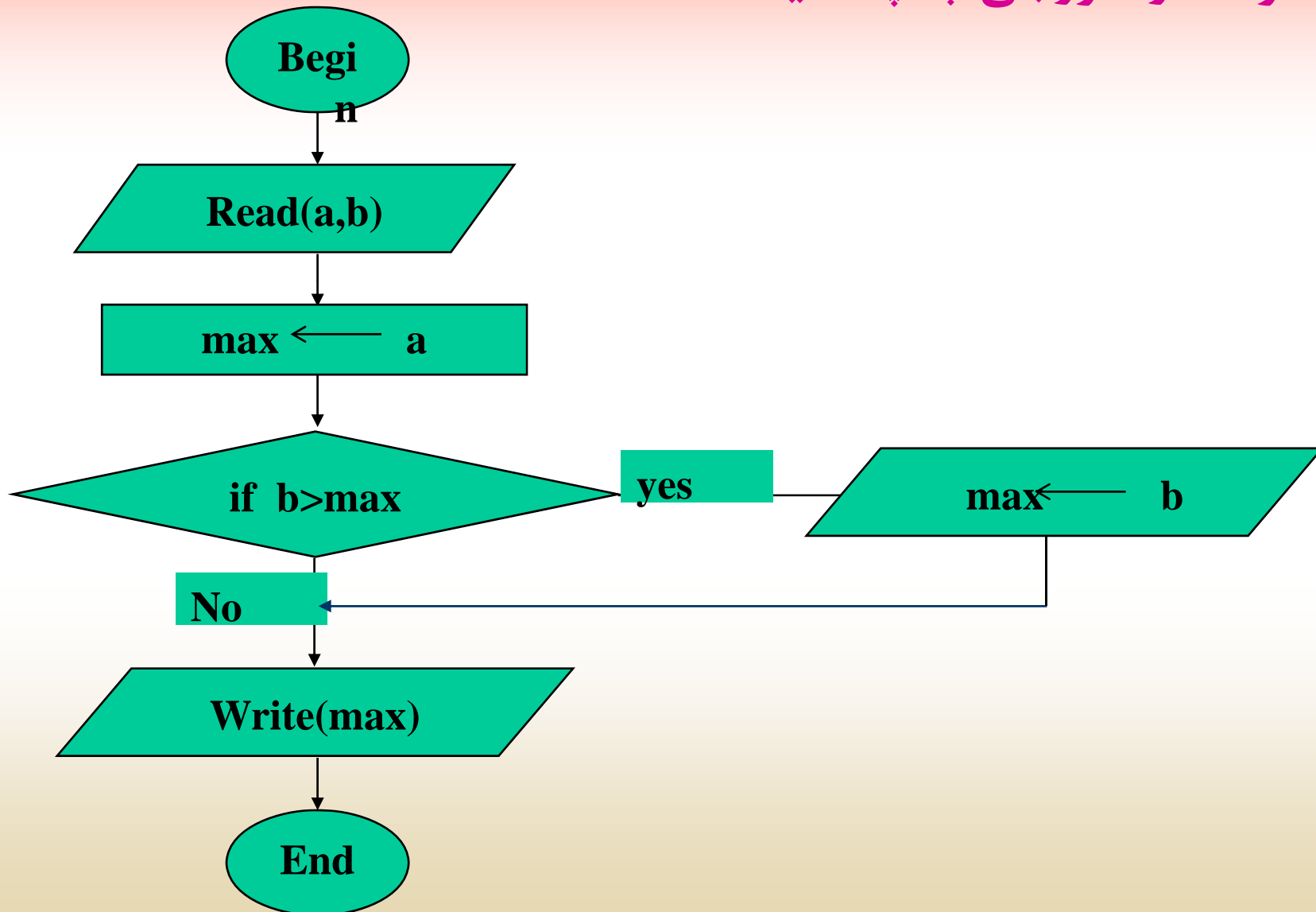
در حالت کلی شرط را بصورت زیر نمایش می دهند:



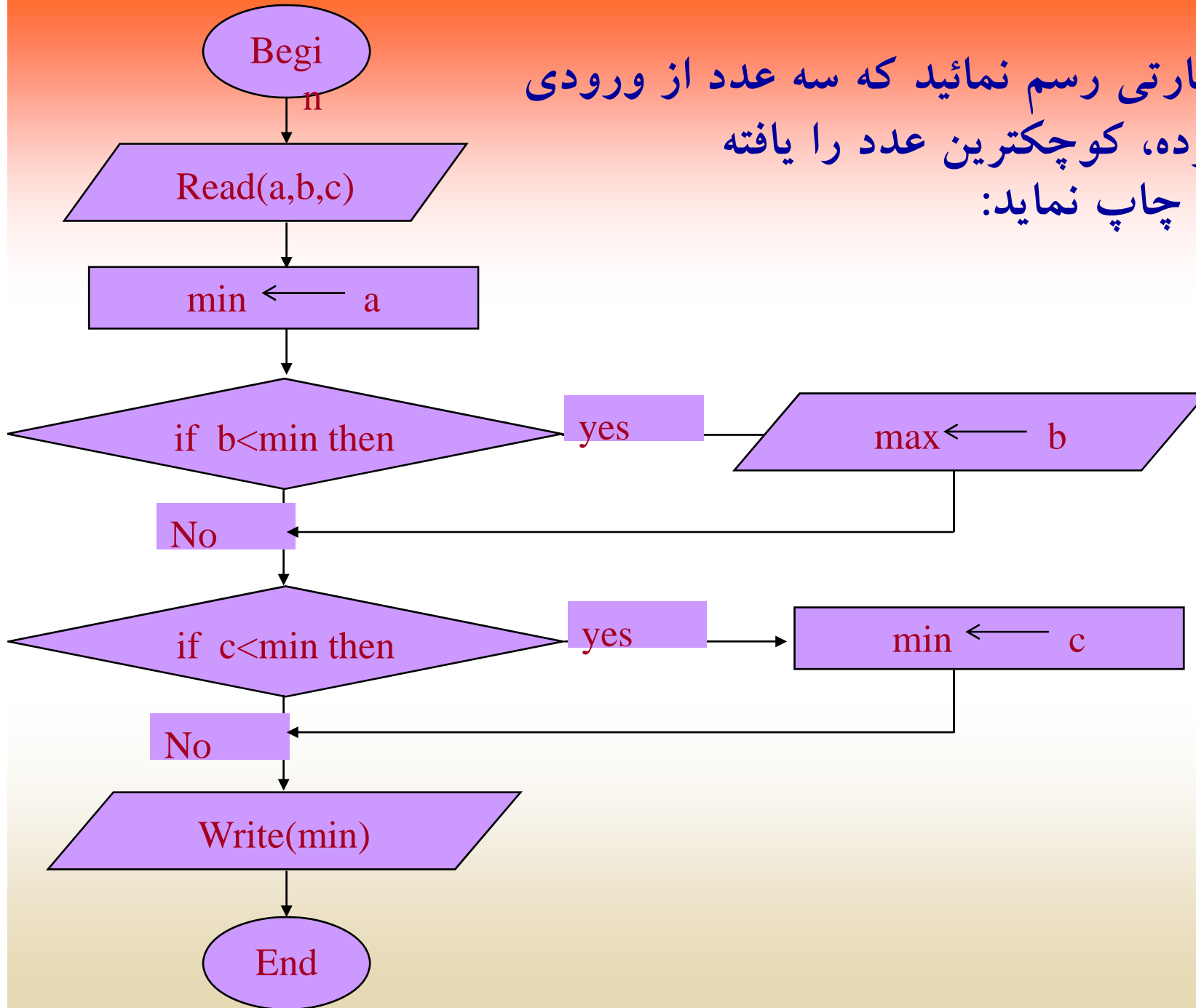
مثال : فلوچارتی رسم نمائید که عددی را از ورودی دریافت کرده، فرد یا زوج بودن آن را تشخیص دهد.



مثال : فلوچارتی رسم کنید که دو عدد از ورودی دریافت کرده بزرگترین عدد را پیدا کرده در خروجی چاپ نماید.



مثال : فلوچارتی رسم نمائید که سه عدد از ورودی دریافت کرده، کوچکترین عدد را یافته در خروجی چاپ نماید:



نمونه اجرای فلوچارت بالا بصورت زیر می باشد:

| | a | b | c | Min | خروج |
|---|----|----|----|-----|------|
| 1 | 12 | 11 | 17 | | 11 |
| 2 | | | | 12 | |
| 3 | | | | 11 | |
| 4 | | | | 11 | |
| 5 | | | | 11 | |

❖ تمرین

۱- فلوچارتی رسم کنید که عددی را از ورودی دریافت کرده، قدر مطلق عدد را در خروجی چاپ کند.

۲- فلوچارتی رسم نمائید که عددی از ورودی دریافت کرده مثبت، منفی یا صفر بودن عدد را تشخیص داده، در خروجی با پیغام مناسب چاپ کند.

۳- فلوچارتی رسم نمائید که عددی را از ورودی دریافت کرده، بخشپذیری آن بر ۳ و ۵ را بررسی نماید.

۴- فلوچارتی رسم نمائید که ضرایب یک معادله درجه دوم را از ورودی دریافت کرده، ریشه‌های آن را محاسبه در خروجی چاپ کند.

❖ حلقه‌ها

در حل بسیاری از مسائل با عملیاتی روبرو می‌شویم ، که نیاز به تکرار دارند و عمل تکرار آنها به تعداد مشخصی انجام می‌گیرد. فرض کنید، بخواهیم میانگین ۱۰۰ عدد را محاسبه کنیم، در اینصورت منطقی بنظر نمی‌رسد که ۱۰۰ متغیر مختلف را از ورودی دریافت کنیم سپس آنها را جمع کنیم.

• انواع حلقه ها

• حلقه های با تکرار مشخص

• حلقه های با تکرار نامشخص

• حلقه های با تکرار مشخص

در این نوع حلقه ها تعداد تکرار مشخص می باشد این حلقه از اجزاء زیر تشکیل می شود:

۱- اندیس حلقه

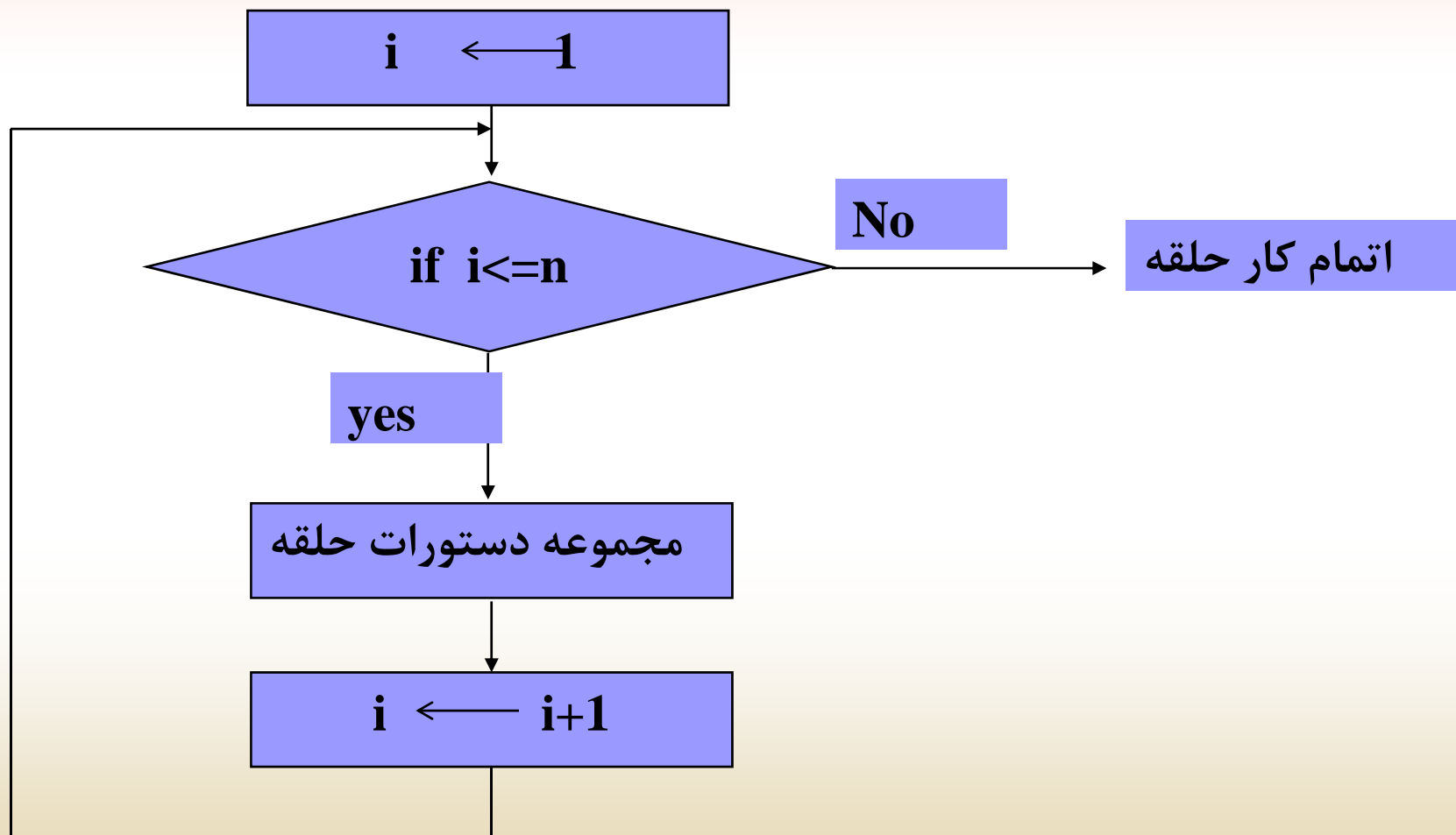
۲- مقدار اولیه برای اندیس حلقه

۳- مقدار افزایشده برای اندیس حلقه (معمولا یک واحد در هر مرحله)

۴- مقدار نهایی (تعداد تکرار حلقه)

۵- شرطی برای کنترل تعداد تکرار حلقه

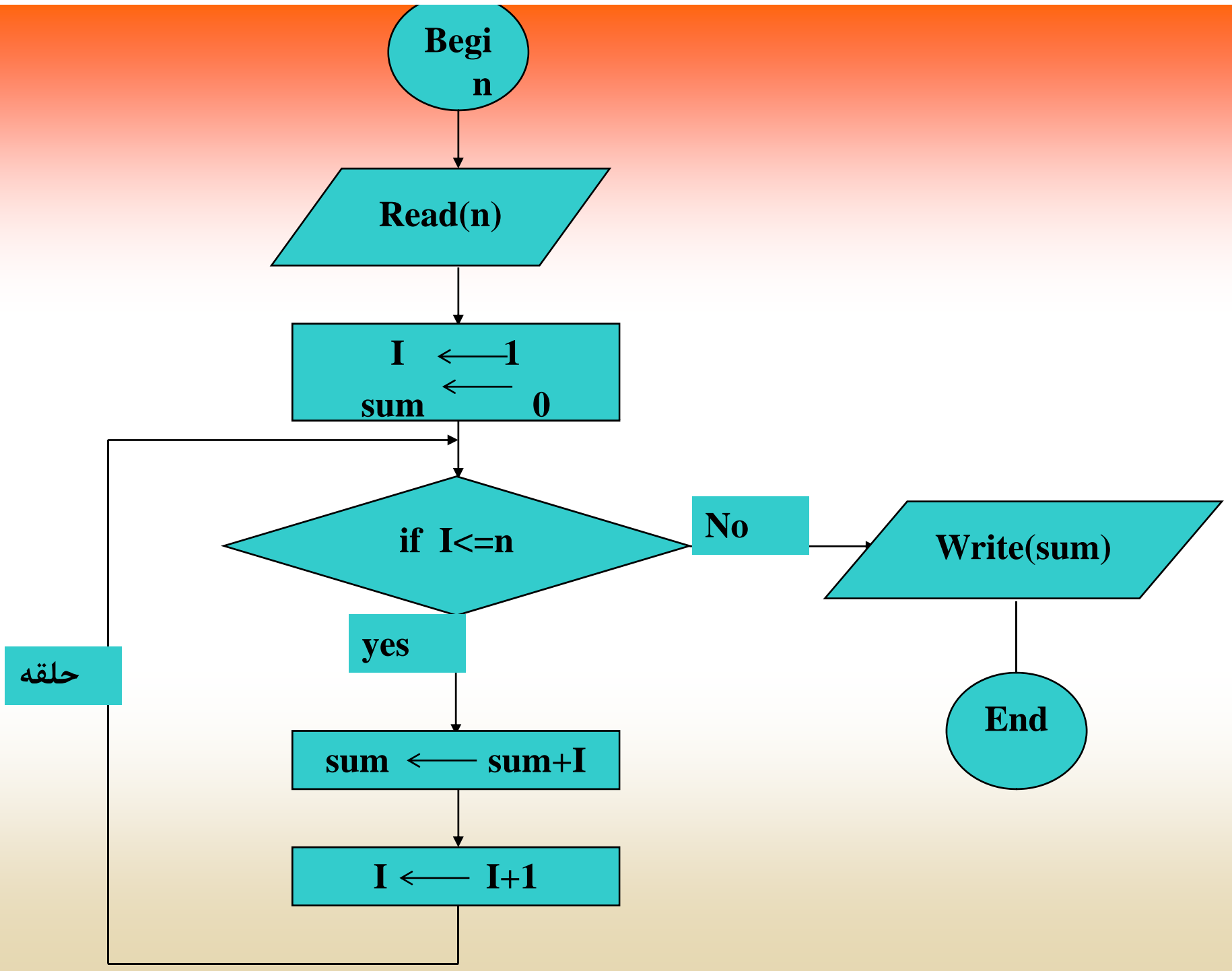
این حلقه‌ها را غالباً با فلوجارت بصورت زیر نمایش می‌دهند:



مثال : فلوجارتی رسم نمائید کہ عدد n را از ورودی دریافت کرده،
مجموع اعداد از یک تا n را محاسبه کند.

n مقدار نهایی

i اندیس حلقه



نمونه اجرای فلوجارت بالا بصورت زیر است:

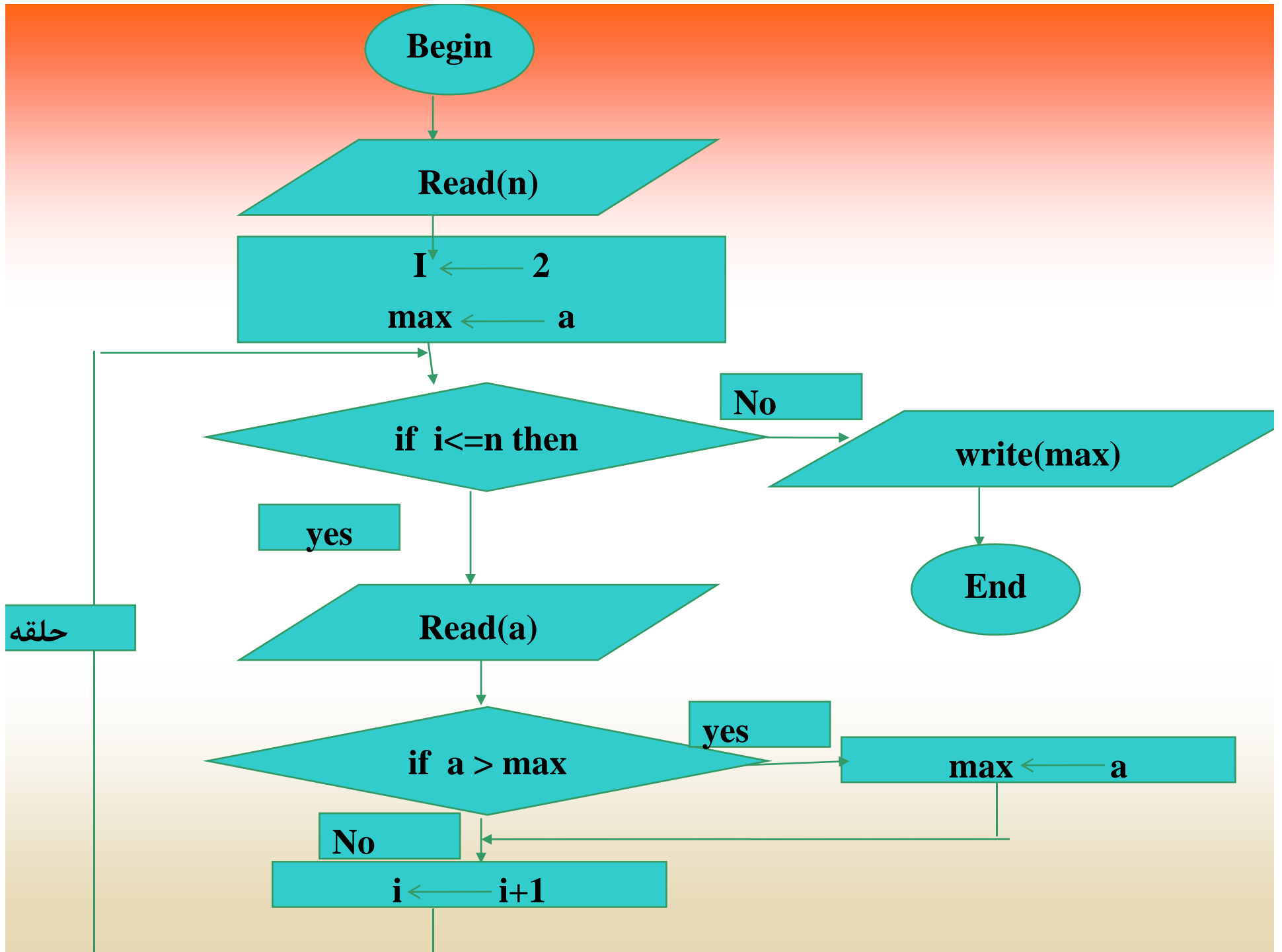
| | N | I | sum | خروجی |
|---|---|---|-----|-------|
| 1 | 5 | 1 | 0 | 15 |
| 2 | | 2 | 1 | |
| 3 | | 3 | 3 | |
| 4 | | 4 | 6 | |
| 5 | | 5 | 10 | |
| 6 | | 6 | 15 | |
| 7 | | | | |

مثال : فلوچارتی رسم کنید که n عدد از ورودی دریافت کرده،
بزرگترین مقدار از بین n عدد را پیدا کرده در خروجی چاپ نماید.

i اندیس حلقه

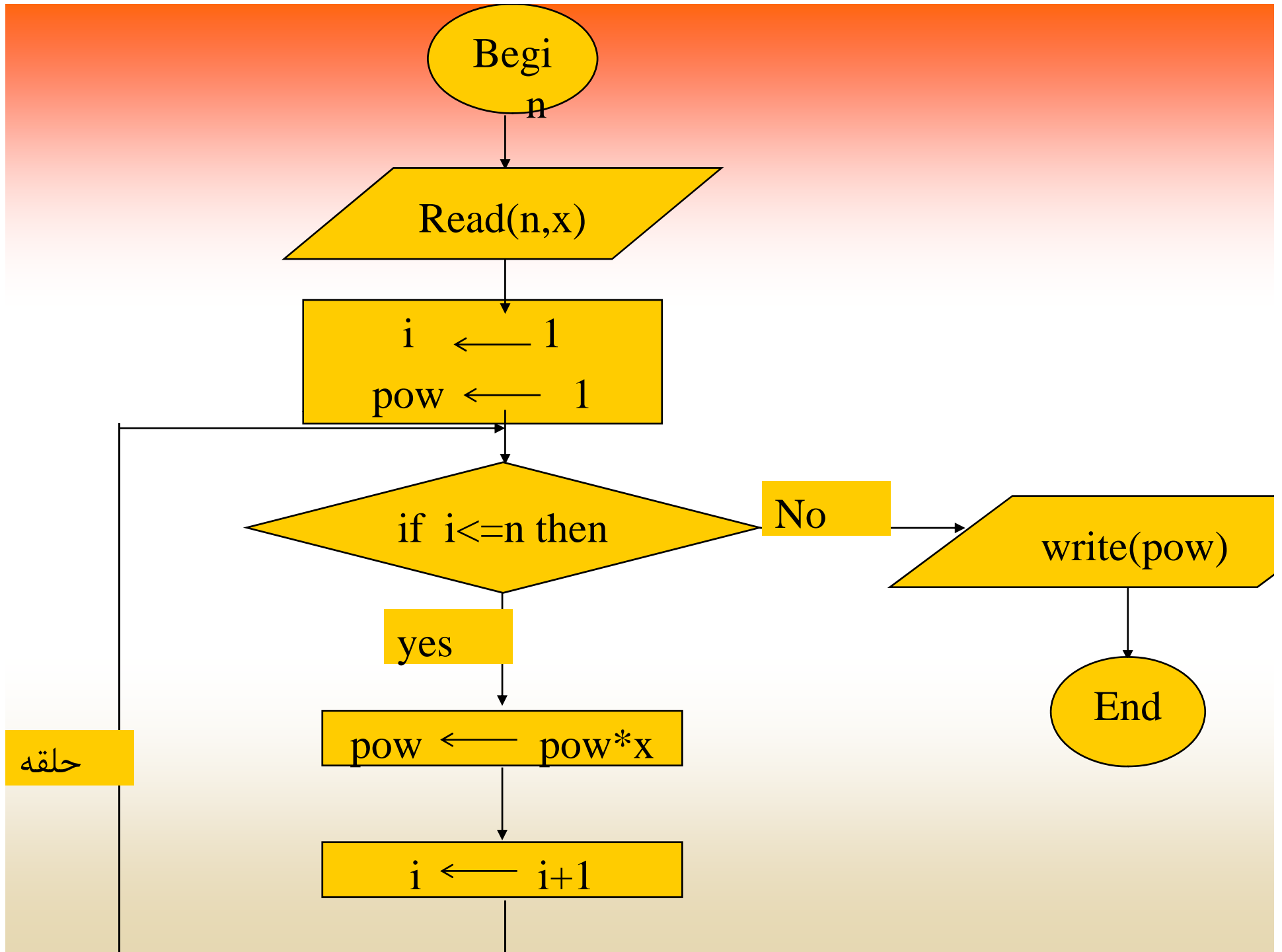
n مقدار نهایی

Max بزرگترین مقدار



مثال : فلوچارتی رسم نمائید که X و n ، دو عدد صحیح مثبت را از ورودی دریافت کرده سپس X به توان n را محاسبه کند.

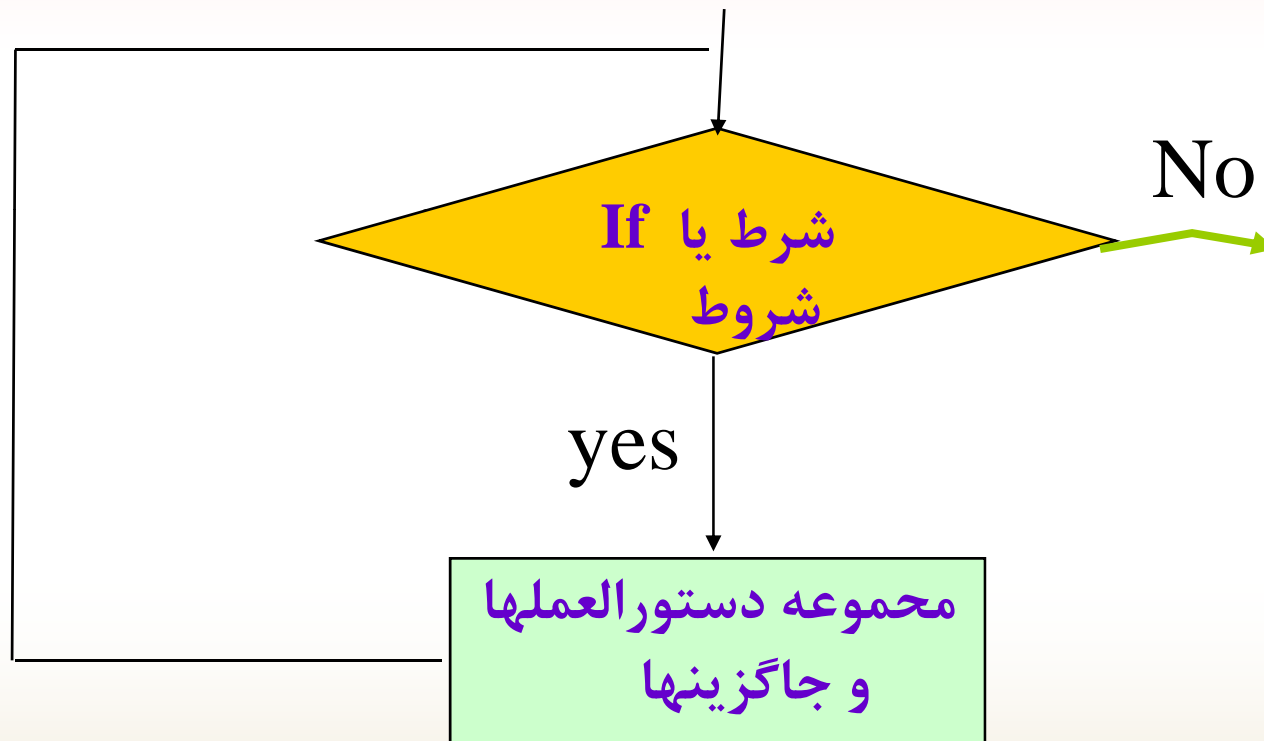
- اندیس حلقه i
- مقدار نهایی n
- عدد به توان n pow



• حلقه‌هایی که تعداد تکرار آنها مشخص نیست
(در پاسکال به حلقه while مشهورند.)

در این حلقه‌ها با توجه به ورودی، تعداد تکرار مشخص می‌شود. و دقیقاً
نمی‌توان تعداد تکرار حلقه را بدون ورودی معین کرد. این حلقه‌ها فقط
شامل شرطی هستند که تا زمانیکه برقرار باشد حلقه اجرا می‌شود.

در حالت کلی این نوع حلقه‌ها بصورت زیر نمایش داده می‌شوند:



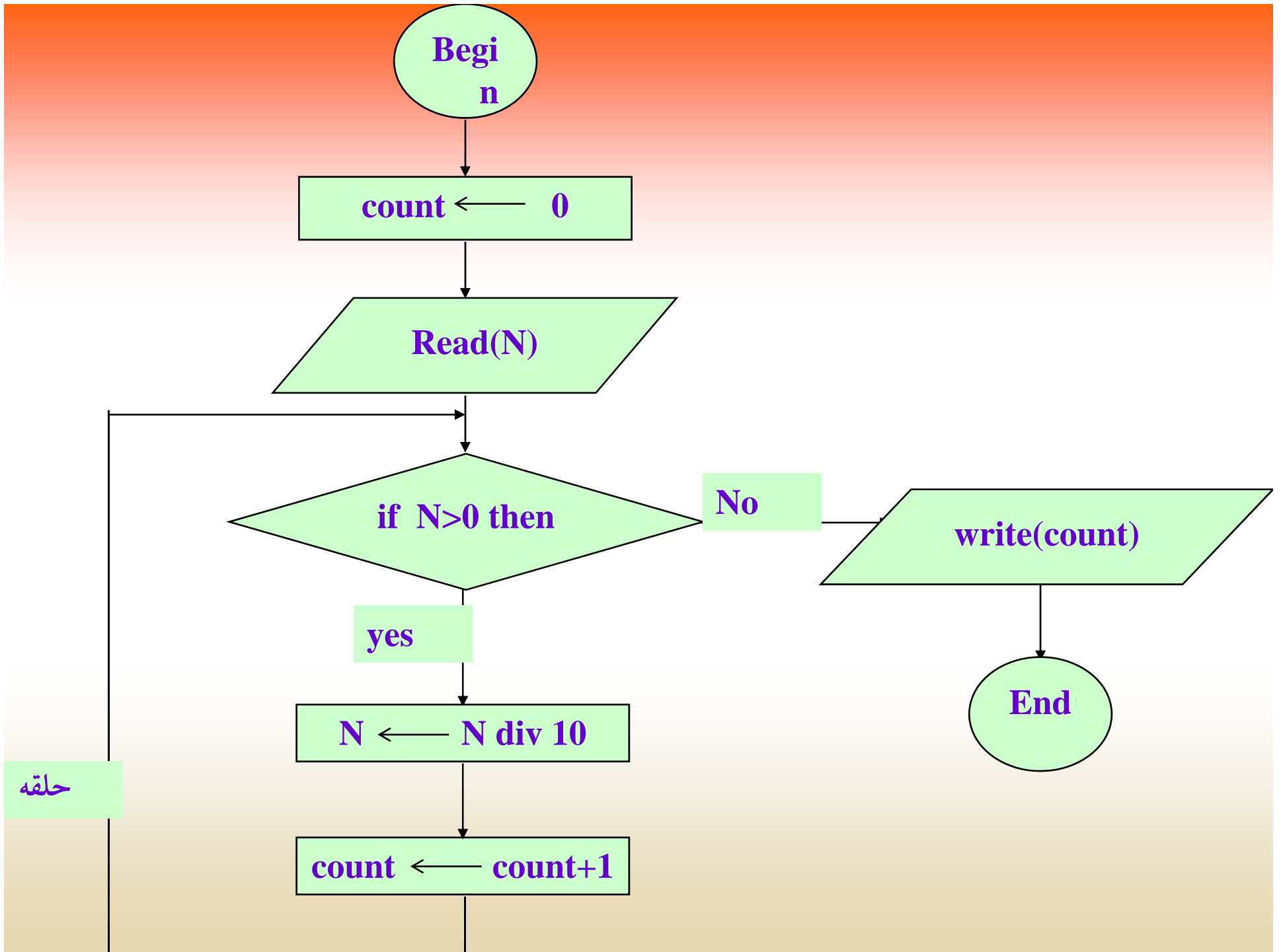
مثال: فلوجارتی رسم کنید که عددی را از ورودی دریافت کرده سپس تعداد ارقام آن را شمرده در خروجی چاپ نماید.

N

• عدد خوانده شده

count

• تعداد ارقام



مثال : فلوچارتی رسم نمائید که عددی از ورودی دریافت کرده، سری فیبوناچی قبل از آنرا تولید نماید.

در حالت کلی جملات سری بصورت:

$$f_k = f_{k-1} + f_{k-2}$$

N

عدد خوانده شده

f₁

جمله اول سری

f₂

جمله دوم سری

f₃

جمله سوم سری

Begin

$f1 \leftarrow 0$

$f2 \leftarrow 1$

Read(N)

$f3 \leftarrow f1 + f2$

if $f3 > N$ then

No

End

yes

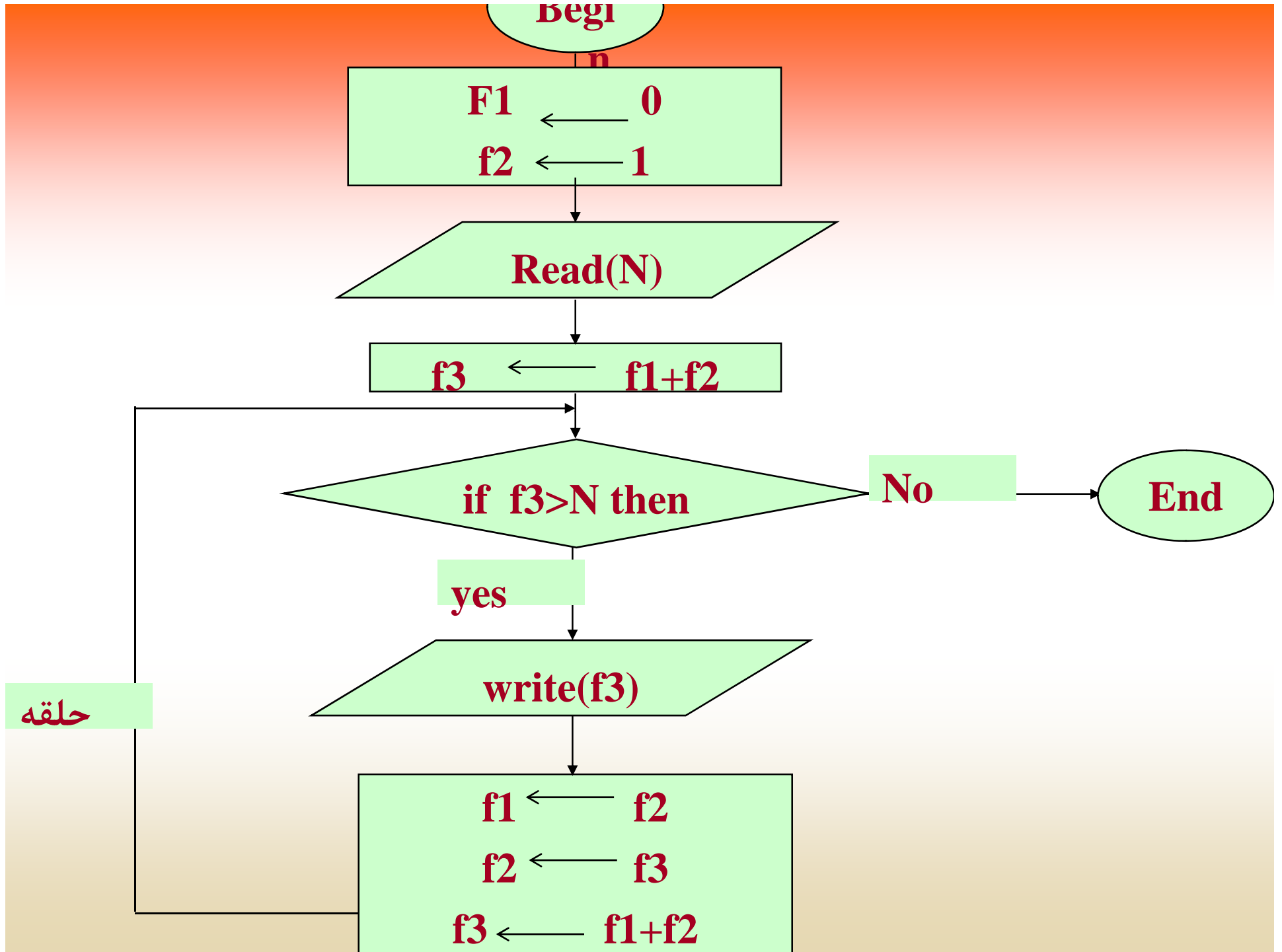
write($f3$)

$f1 \leftarrow f2$

$f2 \leftarrow f3$

$f3 \leftarrow f1 + f2$

حلقه



❖ تمرین

۱- فلوجارتی رسم نمائید که عددی از ورودی دریافت کرده، کامل بودن آنرا بررسی نماید. (عدد کامل، عددی است که مجموع مقسوم‌علیه‌های آن با خودش برابر باشد).

۲- فلوجارتی رسم کنید که N را از ورودی دریافت کرده، N جمله سری فیبوناچی را تولید نماید.

۳- فلوجارتی رسم نمائید که دو عدد M و N را از ورودی خوانده، بزرگترین مقسوم‌علیه مشترک دو عدد را محاسبه و چاپ کند.

❖ حلقه‌های تودرتو

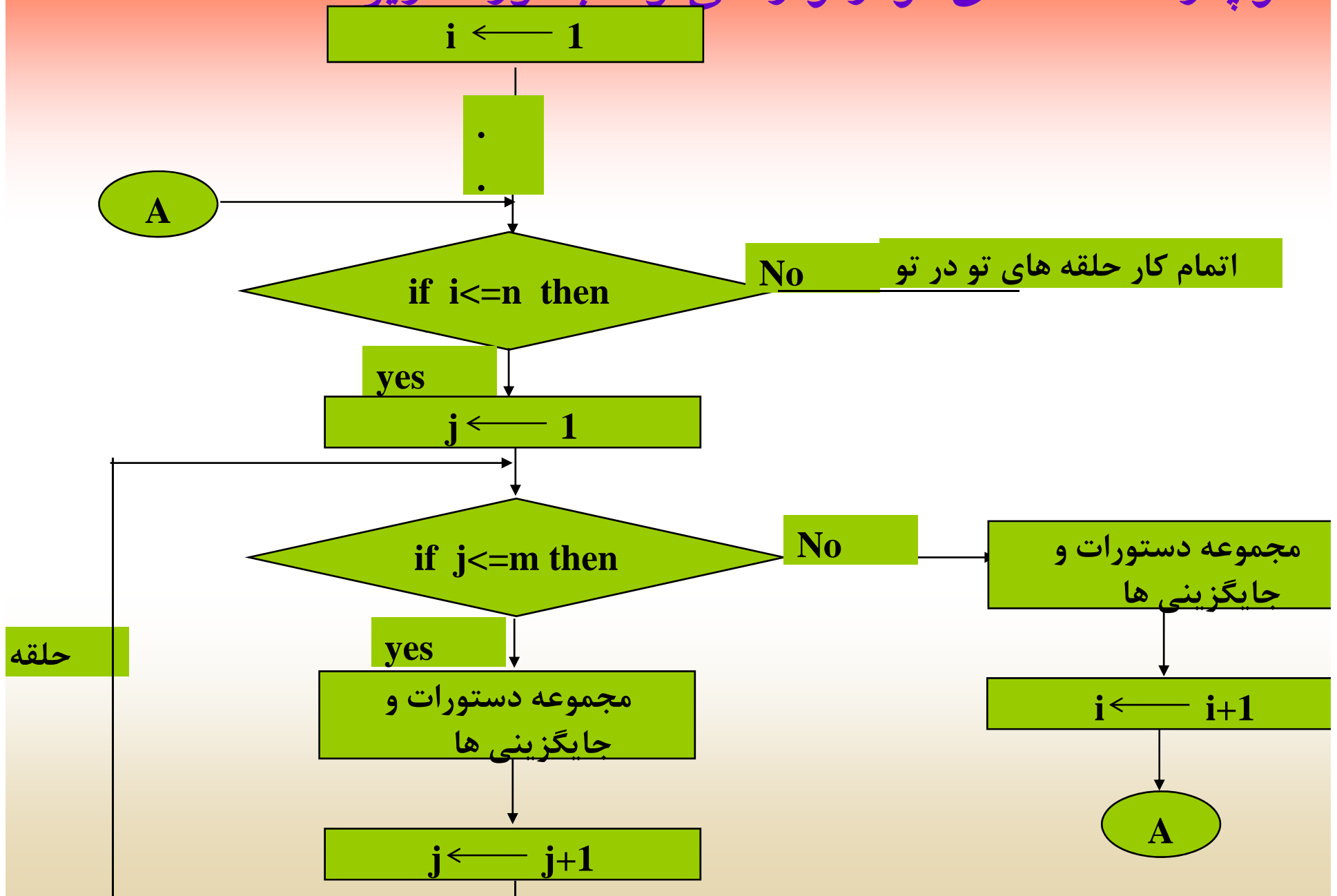
الگوریتم‌هایی که تا حال بکار بردیم، فقط شامل یک حلقه بودند. در صورتی که در بسیاری از مسائل ممکن است نیاز به استفاده از چند حلقه در داخل هم باشیم. در این نوع حلقه‌ها باید دقت بیشتری به خرج دهیم، تا مشکلی پیش نیاید. اگر از حلقه‌های نوع اول بصورت تودرتو استفاده کنیم در اینصورت برای هر حلقه شرط نهایی و اندیس اولیه جداگانه باید تعریف کنیم.

در حلقه‌های تو در تو به ازای یکبار تکرار حلقه اولیه، حلقه داخلی به اندازه مقدار نهایی خود تکرار می‌شود. در کل اگر حلقه اولیه n بار تکرار شود و حلقه داخلی m بار، در این صورت کل حلقه :

$$n \times m$$

بار تکرار خواهد شد.

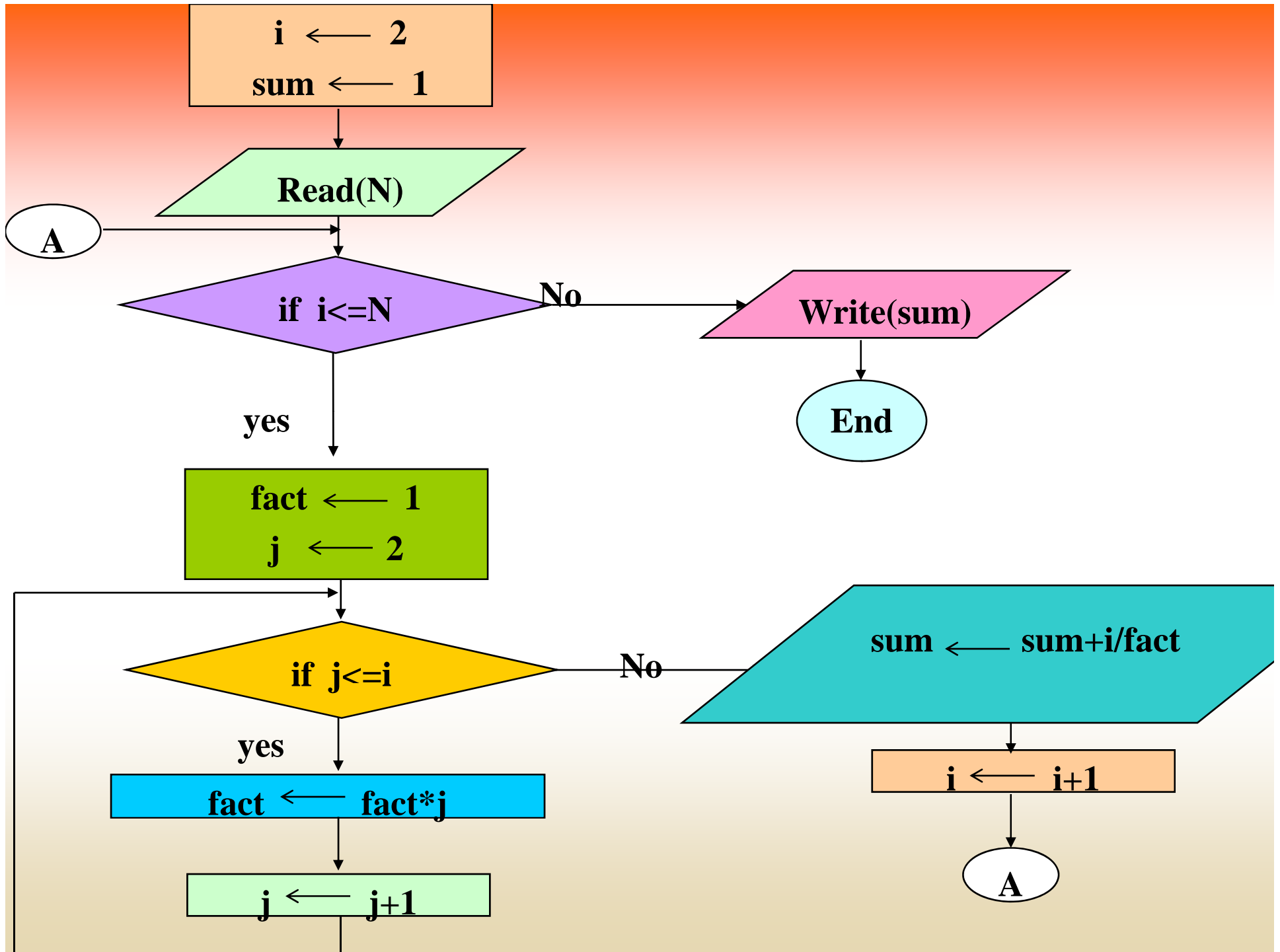
فلوچارت حلقه‌های تو در تو را می‌توان بصورت زیر نشان داد:



مثال : فلوچارتری رسم نمائید که N را از ورودی دریافت کرده، مجموع سری زیر را محاسبه نماید:

$$S = 1 + \frac{2}{2!} + \frac{3}{3!} + \dots + \frac{N}{N!}$$

| | |
|-------------|--------------------|
| I | • اندیس حلقه اول |
| N | • ورودی |
| fact | • محاسبه فاکتوریل |
| j | • اندیس حلقه داخلی |
| Sum | • مجموع |



❖ تمرینات آخر فصل

۱- فلوجارتی رسم نمائید که N عدد از ورودی دریافت کرده تعداد اعداد اول و کامل را شمرده در خروجی چاپ نماید.

۲- فلوجارتی رسم نمائید که N و X را از ورودی خوانده مقدار سری زیر را محاسبه کند:

$$S = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + \frac{x^n}{N!}$$

۳- فلوجارتی رسم نمائید که عددی را از ورودی دریافت کرده مقلوب عدد را محاسبه و در خروجی چاپ کند.

۴- فلوجارتی رسم کنید که تاریخ تولد شخصی را از ورودی خوانده، سن شخص را با تاریخ روز، محاسبه نموده در خروجی چاپ کند.

۵- فلوجارتی رسم نمائید که N, M ($m > n$) را از ورودی دریافت کرده سری فیبوناچی بین N, M را تولید کرده، در خروجی چاپ کند.

کاربرد آرایه ها در الگوریتم ها

هدفهای کلی

شناخت آرایه ها و مفهوم آن

شناخت الگوریتم های لازم برای جستجو و مرتب سازی

مقایسه انواع روش های جستجو با هم

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- از آرایه ها در حل مسئله استفاده کند .
- با استفاده از آرایه ها لیستی را مرتب نماید .
- در صورت لزوم در لیستی جستجو انجام دهد .

مقدمه

فرض کنید بخواهیم اطلاعات ۱۰۰ کارمند را از ورودی بخوانیم و سپس آنها را مرتب کنیم، در اینصورت باید ورودیها را در جایی از حافظه ذخیره کنیم. در زبانهای برنامه‌نویسی معمولاً از آرایه برای ذخیره اطلاعات در حافظه استفاده می‌کنند. در آرایه‌ها ما با توجه به تعداد ورودیها، طول آن را مشخص می‌کنیم. سپس داده‌ها را خوانده در آن قرار می‌دهیم.

❖ تعریف آرایه

خانه‌های پشت سر هم از حافظه، که ممنوع بوده و توسط یک اسم معرفی می‌شوند، آرایه نام دارد.

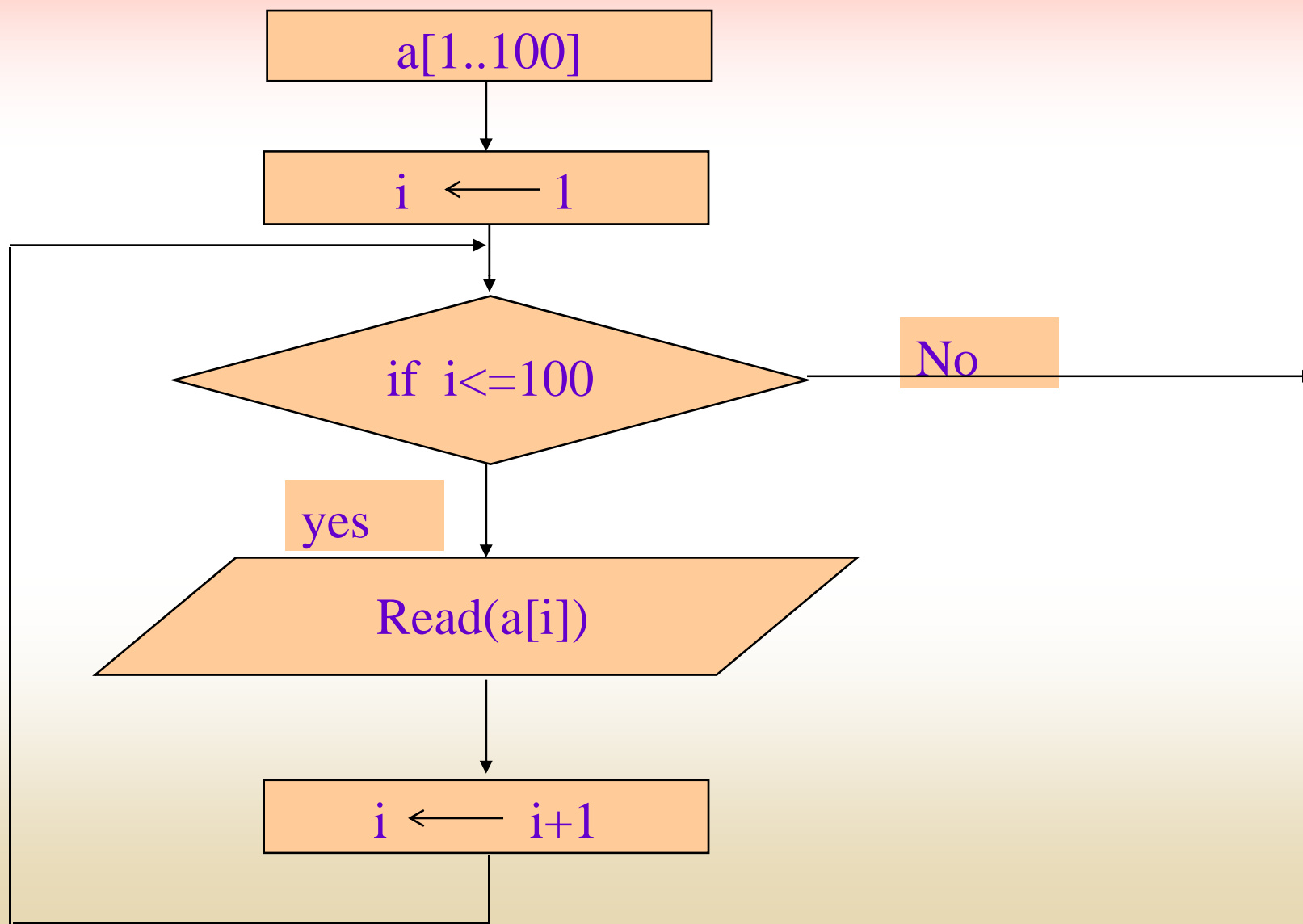
نحوه دسترسی به هر یک از اعضاء آرایه، از طریق اندیس آرایه امکانپذیر است. برای تعریف آرایه ابتدا طول آرایه که در حقیقت تعداد خانه‌های آن را مشخص می‌کند، معین می‌کنیم. سپس نوع خانه‌ها باید معین شوند.

در فلوجارت‌ها آرایه‌ها را بصورت زیر نمایش می‌دهیم:

Name[1 .. Length]

طول آرایه اسم آرایه

برای خواندن یک آرایه از ورودی از حلقه‌ها استفاده می‌کنیم. فلوجارت خواندن آرایه از ورودی بصورت زیر می‌باشد:



با توجه به فلوجارت بالا برای دسترسی به عنصر `Nam` آرایه در حالت کلی بصورت :

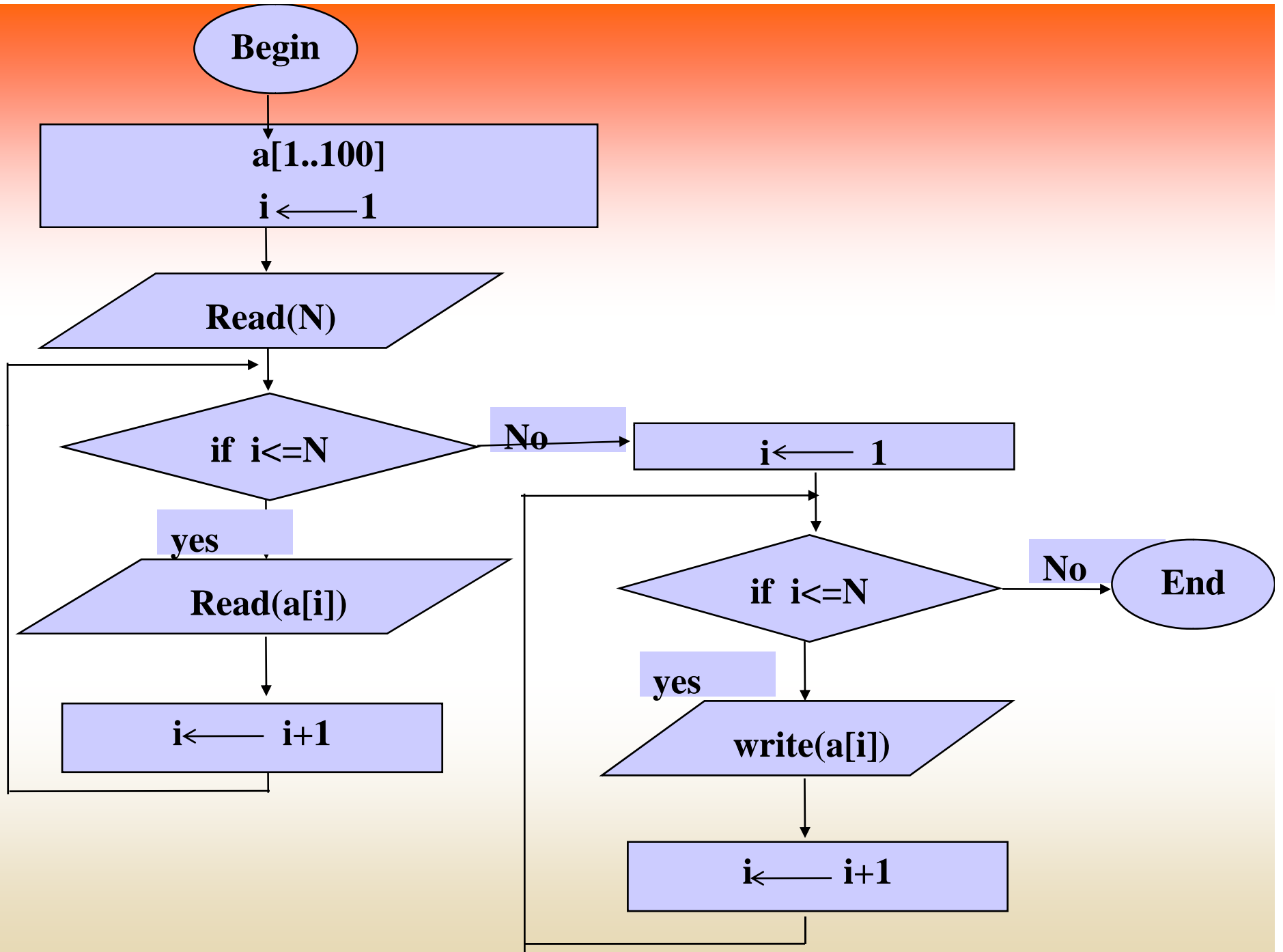
Nam [index]

اندیس آرایه اسم آرایه

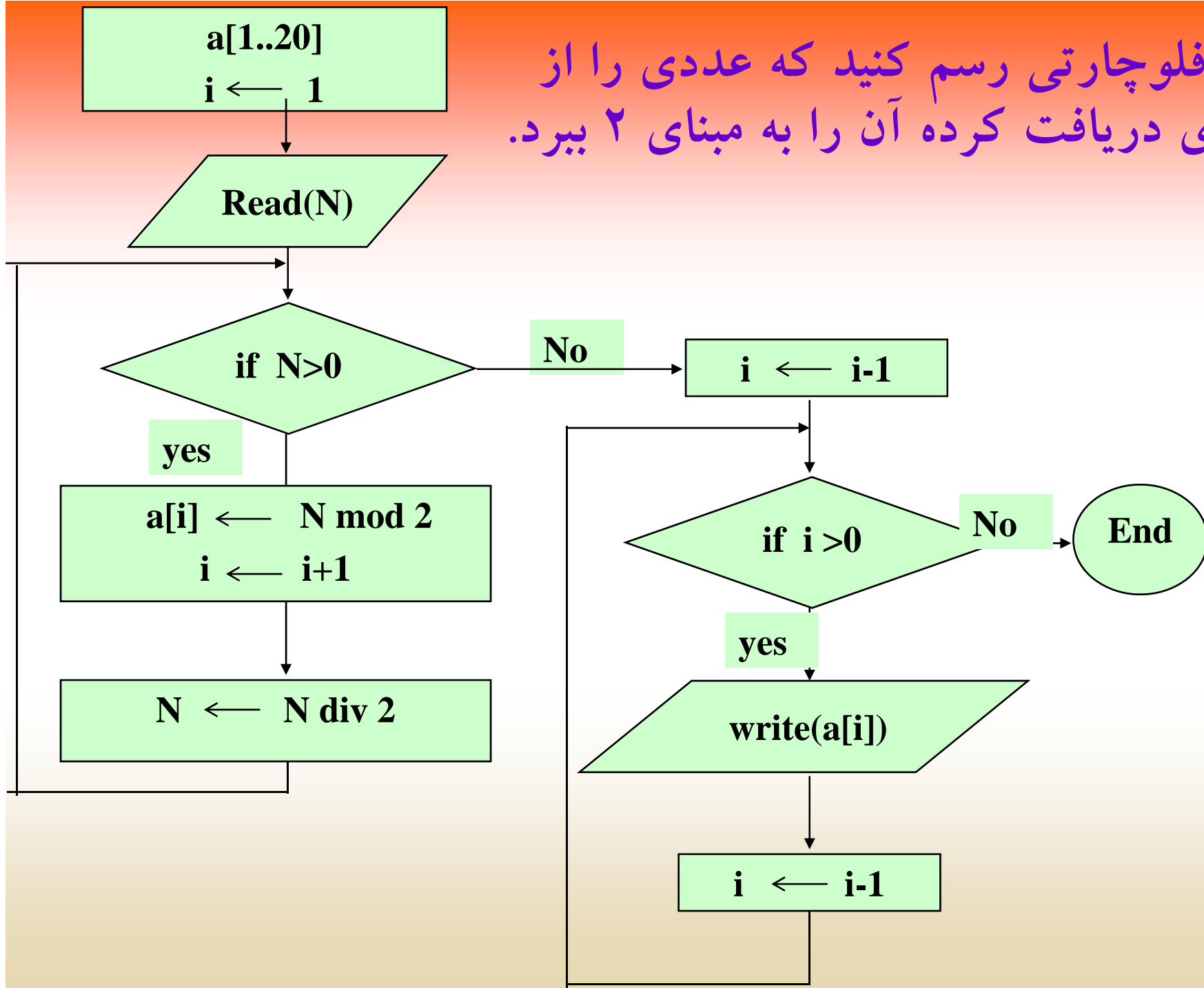
عمل می کنند.

مثال: فلوجارتی رسم کنید که یک آرایه حداکثر ۱۰۰ عنصری را از ورودی

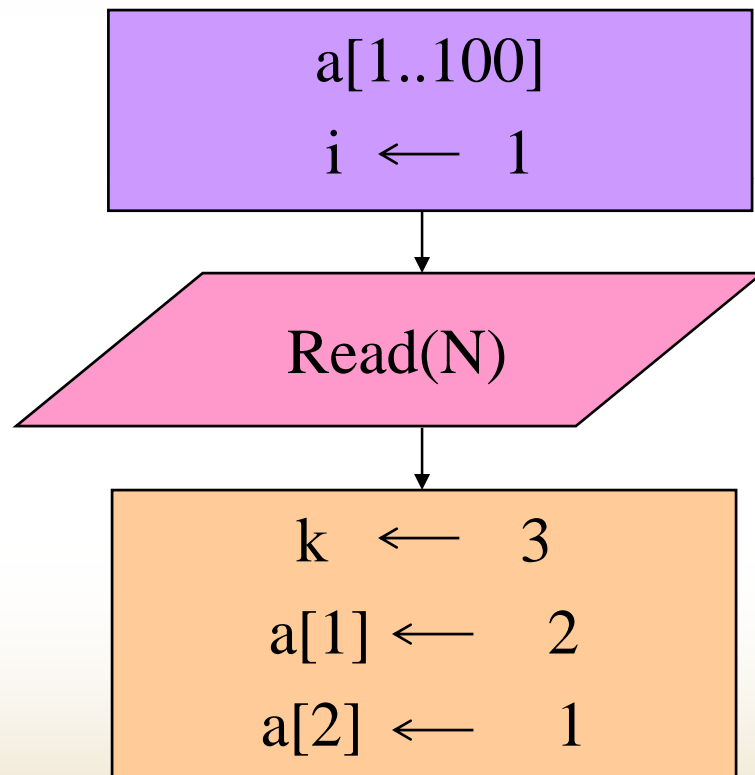
دریافت کرده، سپس آن را خروجی نمایش دهد.

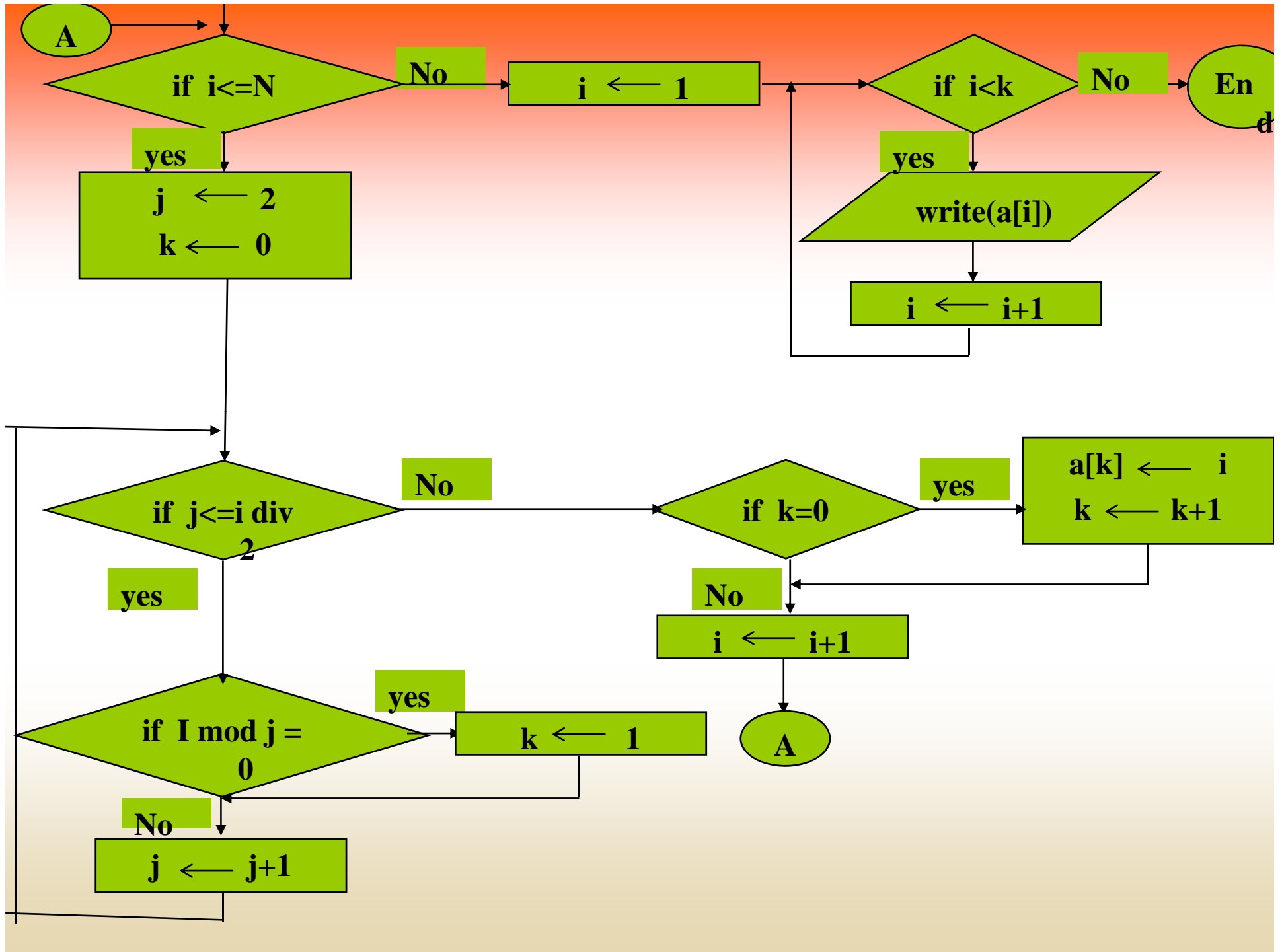


مثال : فلوجارته رسم کنيد که عددي را از ورودی دریافت کرده آن را به مبناي ۲ ببرد.



مثال : فلوچارتی رسم نمائید که عددی از ورودی دریافت کرده سپس اعداد اول قبل از آن را تولید نموده ، در یک آرایه قرار دهد.





❖ جستجو و مرتب سازی (search and sort)

یکی از مسائلی که در بحث طراحی الگوریتم بسیار مهم است، بحث مرتب سازی و جستجو می باشد. منظور از جستجو اینست که یک مقداری را از یک لیست جستجو کنیم و منظور از مرتب سازی اینست که یک لیست مرتب از داده ها را تولید کنیم.

برای جستجو و مرتب سازی الگوریتم های مختلفی وجود دارد در زیر الگوریتم های اولیه، برای جستجو و مرتب سازی را بررسی می کنیم.

❖ الگوریتم های جستجو (search)

دو الگوریتم زیر غالباً برای جستجو بکار می روند:

• جستجوی خطی *linear search*

• جستجوی دودوئی *binary search*

• جستجوی خطی *linear search*

در جستجوی خطی عبارت مورد جستجو را به ترتیب با اولین، دومین

و ... عنصر آرایه مقایسه می کنیم اگر عنصر مورد جستجو پیدا شد

اندیس آن را نمایش می دهیم.

فلوچارت جستجوی خطی بصورت زیر است:

list[1..100]

$i \leftarrow 1$

index $\leftarrow 0$

flag $\leftarrow 0$

Read(x)

$i \leftarrow 1$

if ($i \leq N$) and
(flag=0)

No

write(index)

End

yes

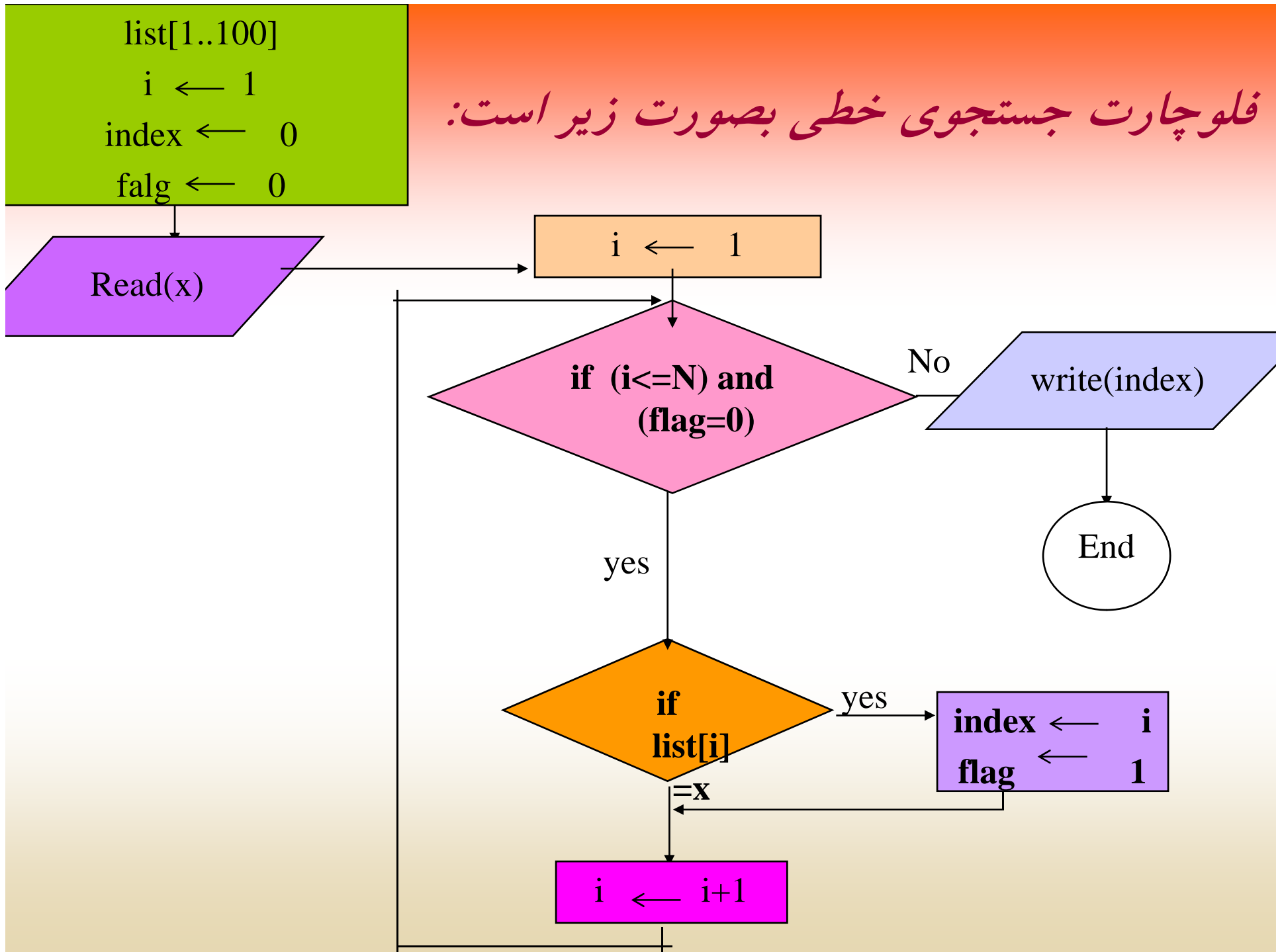
if
list[i]

yes

index $\leftarrow i$
flag $\leftarrow 1$

=x

$i \leftarrow i+1$



• جستجوی دودوئی *binary search*

در جستجوی دودوئی ، لیست مورد جستجو، مرتب می باشد. لذا برای جستجو اعمال زیر انجام می شود:

۱- عنصر X با عنصر وسط آرایه که اندیس آن برابر

middle ← $(low+high)/2$

مقایسه می شود.

۲- اگر x از عنصر وسط کوچکتر باشد، عنصر مورد نظر احتمالاً در قسمت بالای لیست قرار دارد. لذا آرایه با اندیس، جدید در نظر گرفته می‌شود و قسمت پایین لیست از فضای جستجو حذف می‌شود.

۳- اگر x از عنصر وسط بزرگتر باشد قسمت بالای لیست حذف می‌شود و فضای جستجو، قسمت پایین آرایه خواهد بود.

۴- اگر x برابر عنصر وسط باشد عمل جستجو خاتمه می‌پذیرد.

❖ مرتب سازی (sort)

مرتب سازی بحث بعدی این فصل می باشد. برای مرتب کردن داده ها نیز الگوریتم های مختلفی وجود دارد، که هر کدام مزایا و معایب خاص خود را دارد. بحث مفصل در این مورد را به فصل های بعد واگذار می کنیم.

❖ تمرین

فلوچارتی رسم نمائید که عددی از ورودی دریافت کرده، اعداد کامل قبل از خود را تولید و در یک آرایه قرار دهد.

فلوچارتی رسم نمائید که یک آرایه حداکثر ۱۰۰ عنصری از ورودی دریافت کرده، عناصری از آن که اول هستند را با صفر کردن حذف نماید.

فلوچارتی رسم نمائید که یک عدد حداکثر ۲۰ رقمی را توسط آرایه‌ای از ورودی دریافت نماید. سپس یک عدد تک رقمی را از ورودی خوانده در عدد ۲۰ رقمی ضرب نموده، حاصل را در خروجی چاپ نماید.

آشنایی با کامپیوتر



دانشگاه پیام نور
دانشکده فناوری
اطلاعات (گروه
مهندسی
کامپیوتر)

هدفهای کلی

شناخت کامپیوترهای نسل قدیم و امروزی
شناخت سخت افزارهای لازم برای کامپیوترهای شخصی
بررسی نرم افزارها و انواع آن

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

کامپیوترهای نسل جدید را با کامپیوترهای نسل قدیم مقایسه کند. ■

سخت افزارهای لازم برای کامپیوترهای شخصی را بشناسد. ■

انواع حافظه، مزایا و معایب آنها را شناخته و با هم مقایسه نماید. ■

سیستم عامل و انواع آن را مقایسه نماید. ■

نرم افزار و زبانهای برنامه نویسی را تعریف کند. ■

❖ کامپیوترهای قدیمی

اولین کامپیوتر بزرگ (Super Computer) همه منظوره دیجیتال الکترونیک، تحت عنوان ENIAC در سال ۱۹۴۶ میلادی در دانشگاه پنسیلوانیا ساخته شد. این کامپیوتر با سرمایه ارتش آمریکا طراحی شد. وزن این کامپیوتر ۳۰ تن و ابعاد آن ۳۰×۵۰ فوت بود. این کامپیوتر برای محاسبه جدول پرتابه‌ها، پیش‌گویی وضع آب و هوا و محاسبات انرژی اتمی بکار می‌رفت.





در کامپیوترهای اولیه از لامپهای خلاء بعنوان عنصر الکترونیکی پایه استفاده می کردند. در این ماشین ها ۱۹۰۰۰ لامپ خلاء استفاده شده بود و برای انرژی مصرفی لامپها و همچنین دستگاههای تهویه و خنک کننده ماشین حدود ۱۳۰ kW انرژی الکتریکی مصرف می شد. این ماشینها دارای حجم زیادی بودند و سطحی را معادل ۹۰۱۵ مترمربع اشغال می کردند. این کامپیوترها به کامپیوترهای نسل اول معروف شدند.

❖ کامپیوترهای امروزی

کامپیوترهای امروزی با بکارگیری ریزپردازنده به کامپیوترهای نسل چهارم معروفند. البته نسل‌های جدید دیگر کامپیوترها نیز به بازار ارائه می‌شود.

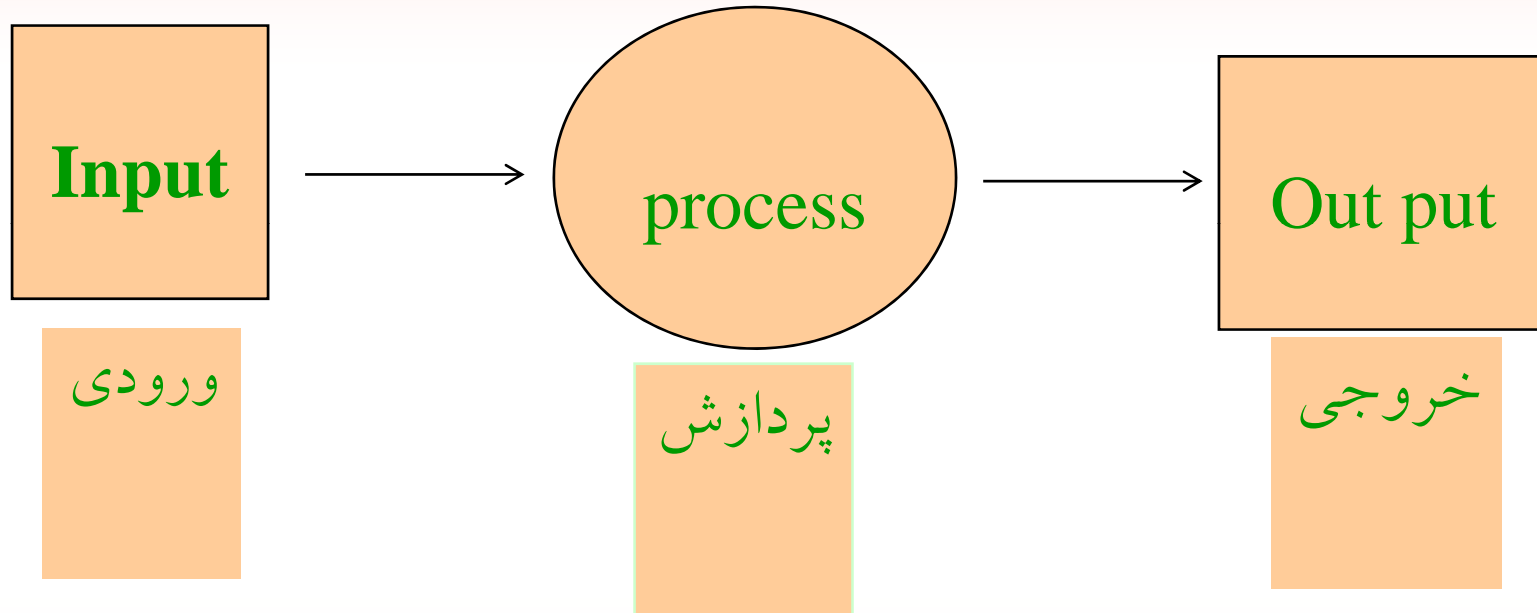
در کامپیوترهای امروزی سرعت پردازش بسیار بالا، حجم اجزاء سخت‌افزاری بسیار کوچک، حجم حافظه بالا و غیره آنها را از نسل‌های دیگر متمایز می‌سازد.

اجزاء تشکیل دهنده کامپیوتر عبارتند از :

❖ سخت افزار

❖ نرم افزار

❖ سخت افزار



کامپیوترهای امروزی معمولاً از قطعات زیر تشکیل می‌شوند:

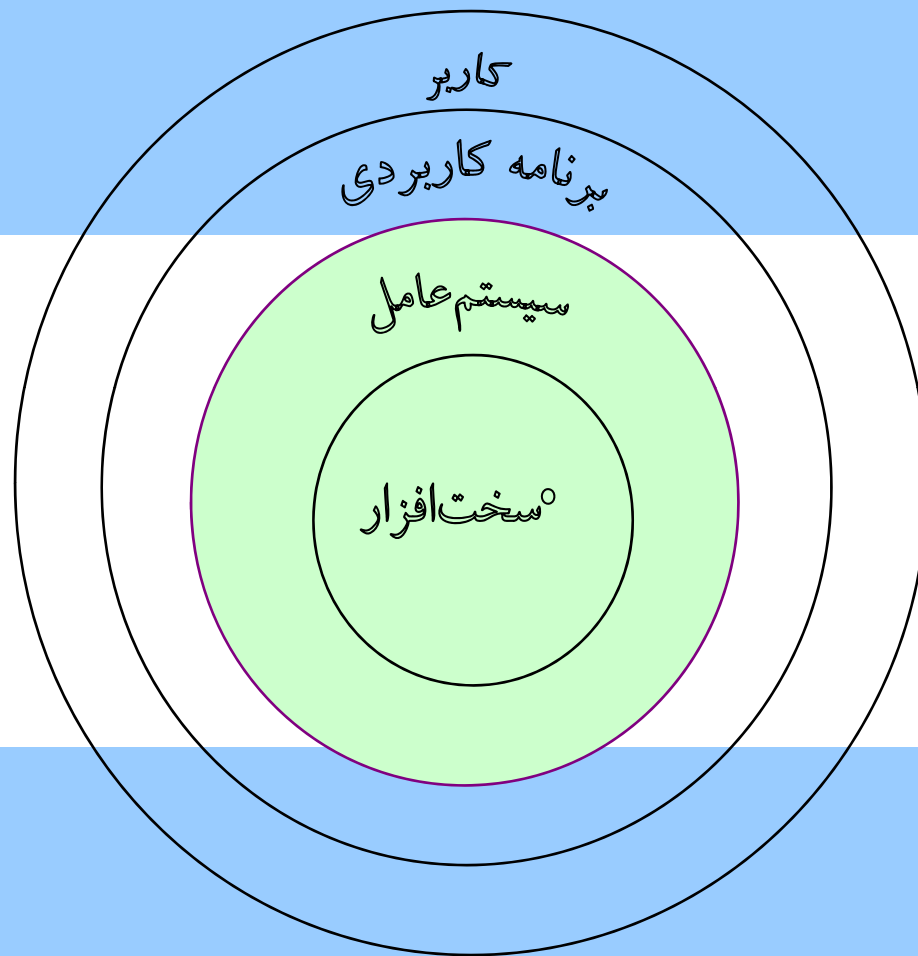
- دستگاههای ورودی
- حافظه‌های جانبی
- حافظه‌های اصلی
- واحد پردازشگر مرکزی
- دستگاههای خروجی

❖ نرم افزار

نرم افزار یکی از بخش های اساسی کامپیوتر به شمار می آید، که در واقع سخت افزار را بکار می گیرد. بعبارت دیگر رابط بین کاربر و سخت افزار را نرم افزار می نامند. نرم افزار در حقیقت روح و جان یک کامپیوتر است، که به سخت افزار هویت می بخشد.

نرم افزار سیستم عامل

- سیستم عامل (OS: Operating System) مشهورترین نوع نرم افزارهای سیستمی می باشد. که مدیریت منابع سیستمی را بر عهده دارد. سیستم عامل، همچنین ارتباط بین کاربر و اجزاء سخت افزاری و نرم افزاری دیگر را برقرار می کند.



کاربر

برنامه کاربردی

سیستم عامل

سخت افزار

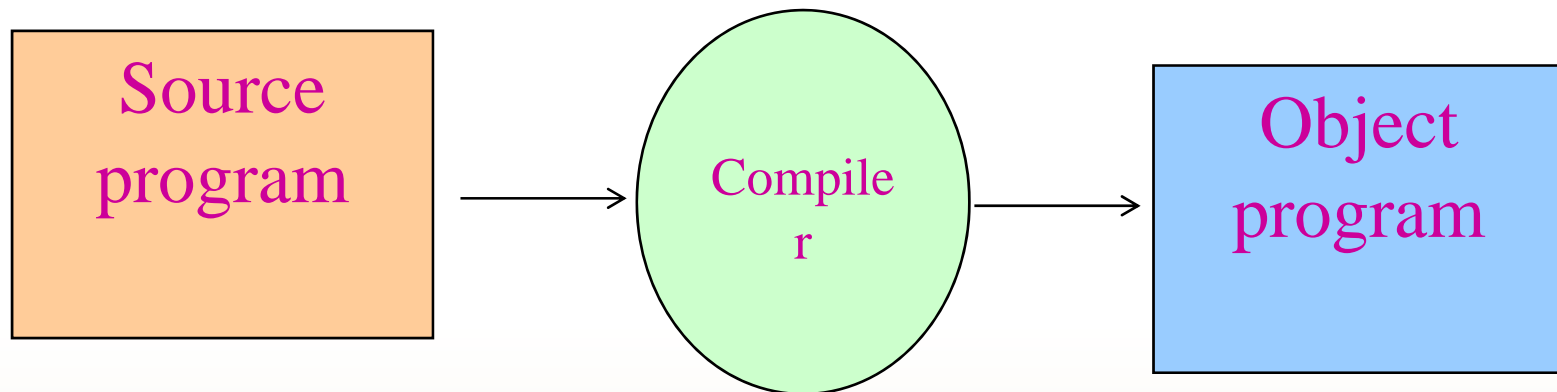
❖ زبانهای برنامه‌نویسی

- نرم‌افزارها توسط زبانهای برنامه‌نویسی نوشته می‌شوند. زبانهای برنامه‌نویسی، یک سیستم ارتباطی هستند که توسط آنها می‌توان دستورات لازم را به ماشین انتقال داد.
- هر زبان برنامه‌نویسی به مجموعه‌ای از علائم، قواعد و دستورات عمل‌ها گفته می‌شود که امکان ارتباط با کامپیوتر را جهت بیان کاری یا حل مسئله‌ای فراهم می‌کند.

در حالت کلی زبانهای برنامه نویسی را به سه دسته زیر تقسیم بندی می کنند:

- زبانهای سطح بالا
- زبانهای سطح پایین
- زبانهای سطح میانی

کامپایلر برنامه نوشته در یک زبان سطح بالا را به برنامه مقصد تبدیل می کند.



زبان Pascal

● در این کتاب زبان پاسکال (Pascal) را برای آموزش و نوشتن برنامه‌ها انتخاب کردیم. این زبان که به افتخار بلز پاسکال دانشمند فرانسوی قرن هفدهم میلادی، پاسکال نامگذاری شده است، در اواخر سال ۱۹۶۰ و اوایل ۱۹۷۰ توسط پروفیسور نیکلاس ویژت در انستیتو فنی فدرال سوئیس مطرح گردید

ساختار برنامه در زبان پاسکال

هدفهای کلی

- شناخت اجزای تشکیل دهنده یک برنامه
- شناخت ساختار یک برنامه در زبان پاسکال
- بررسی دستگاههای خروجی و دستورات لازم در زبان پاسکال برای تولید خروجی

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

■ اجزاء لازم برای نوشتن برنامه در زبان پاسکال را بداند.

■ یک شناسه صحیح در زبان پاسکال را تعریف کند.

■ ساختار یک برنامه در زبان پاسکال و اعلانهای مربوط به برنامه را تعریف نماید.

■ یک برنامه ساده به زبان پاسکال که فقط خاصیت خروجی دارد، بنویسد.

❖ اجزای تشکیل دهنده یک برنامه

- کلمات ذخیره شده (Reserved Words)

- شناسه ها (identifier)

کلمات ذخیره شده (Reserved Words)

کلمات ذخیره شده، کلماتی هستند که مترجم زبان آنها را می شناسد و معنای خاصی برای زبان دارند. مترجم زبان به محض مشاهده این کلمات اعمال خاصی را انجام می دهد. هر زبان دارای تعداد مشخصی کلمات ذخیره شده می باشد و این تعداد قابل افزایش توسط برنامه نویس نیست.

لیست کلمات ذخیره شده در پاسکال عبارتند از:

| | | | |
|-------------|----------------|-----------|--------|
| 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 |
| end | library | shl | |

شناسه‌ها (identifier)

- شناسه در پاسکال برای نامگذاری ثابتها، تایپها، پروسیجروها، توابع، میدانهای یک رکورد، برنامه و همچنین یونیت مورد استفاده قرار می‌گیرد.



در حالت کلی دو نوع شناسه وجود دارد :

➤ **id** های استاندارد: این نوع **id** ها از قبل در زبان پاسکال تعریف شده‌اند و در برنامه‌ها، معنای خاصی دارند .

➤ **id** های غیراستاندارد: این نوع **id** ها بوسیله کاربر بطور مجزا تعریف می‌شوند و اصطلاحاً به آنها **userdefined** گفته می‌شود.

❖ ساختار برنامه در زبان پاسکال

اجزاء اصلی یک برنامه به زبان پاسکال بصورت زیر می باشد:

عنوان برنامه

قسمت تعاریف برنامه

قسمت دستور العملها

قسمت تعاریف برنامہ

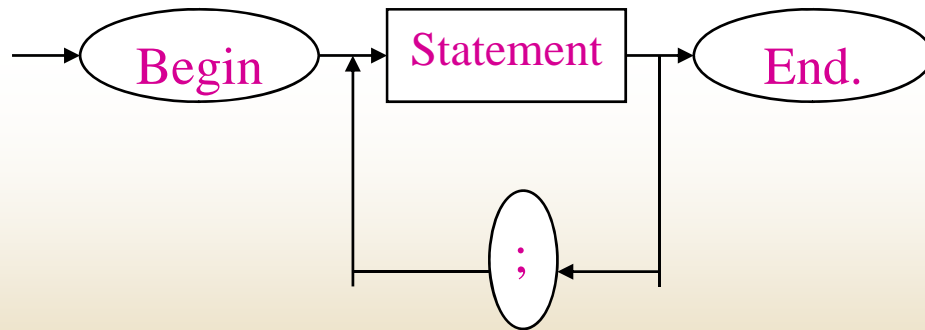
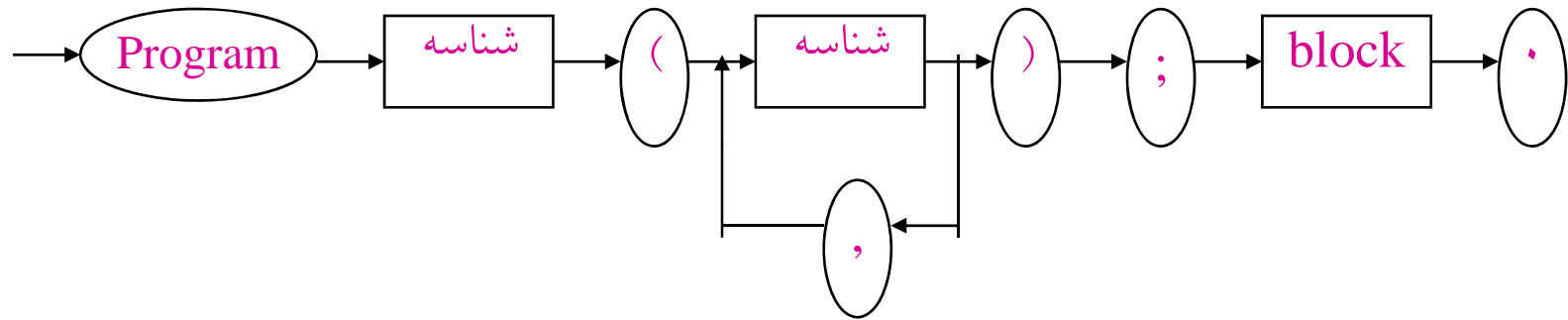
- اعلان ثابتہا Constant Declaration
- اعلان انواع Type Declaration
- اعلان متغیرها Variable Declaration
- اعلان برچسبها Label Declaration

در حالت کلی می توان شکل یک برنامه در زبان پاسکال را بصورت زیر بیان کرد:

- استفاده از کلمه ذخیره شده **Program** و اسم برنامه (که می تواند بکار برده نشود)
- قسمت تعاریف شناسه ها
- بلوک اصلی برنامه که با **Begin** شروع و به **End** همراه نقطه (.) ختم می شود.
- هر دستور در پاسکال به (;) ختم می شود.



نمودارهای یک برنامه و بلوک بترتیب به صورت زیر می باشد:



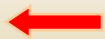
❖ خروجی (Output)

مشهورترین دستگاههای خروجی عبارتند از:

| | |
|------------|-----------|
| صفحه نمایش | Monitor |
| چاپگر | Printer |
| ترمینال | Terminale |
| رسام | Plotter |

❖ مثال های حل شده

```
Program    Print ( output ) ;  
Begin  
    Writeln ( ' Pascal Language ' )  
;          Writeln ( ' Hello ' ) ;  
End.
```



خروجی برنامه بالا بصورت زیر می باشد:

Pascal Language
Hello

فصل 5

انواع عملگرها و داده‌ها در زبان پاسکال

هدفهای کلی

- معرفی انواع عملگرها در زبان پاسکال
- شناخت انواع داده‌ها
- بررسی اولویت عملگرها
- معرفی دستورات جایگزینی در پاسکال

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- انواع عملگرها در زبان پاسکال را بکار ببرد
- انواع داده‌ها برای یک برنامه را تعریف کند.
- اولویت عملگرها در یک عبارت را تشخیص دهد.
- یک برنامه ساده با عملیات معمولی را بنویسد.

❖ عملگرها

عملگرها نمادهایی هستند که برای انجام اعمال خاصی مورد استفاده قرار می‌گیرند. عملگرها برای انجام اعمال خاصی روی عملوندها (Operands) بکار می‌روند. با توجه به نوع عملگر ممکن است یک یا دو عملوند وجود داشته باشد. عملگرها در زبان پاسکال از تنوع زیادی برخوردارند.

در پاسکال چهار دسته عملگر وجود دارند :

محاسباتی

رابطه ای

منطقی

بیتی

عملگرهای محاسباتی

| ردیف | عملگر | نام | مثال |
|------|-------|---------------------|--------------------|
| 1 | + | جمع | $x + y$ |
| 3 | - | تفریق و منهای یکانی | $x - y$, $-x$ |
| 3 | × | ضرب | $x * y$ |
| 4 | / | تقسیم | x / y |
| 5 | div | تقسیم | $a \text{ div } b$ |
| 6 | mod | باقیمانده تقسیم | $a \text{ mod } b$ |

عملگرهای رابطه ای

| عملگر | نام | مثال |
|-------|--------------|----------|
| > | بزرگتر | $x > y$ |
| >= | بزرگتر مساوی | $x >= y$ |
| < | کوچکتر | $x < y$ |
| <= | کوچکتر مساوی | $x <= y$ |
| = | مساوی بودن | $x = y$ |
| <> | نامساوی | $x <> y$ |

عملگرهای منطقی

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

عملگرهای بیتی

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

تقدم عملگرها

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

❖ انواع داده‌ها (data types)

➤ داده‌های ساده (Simple data type)

➤ داده‌های ساخت‌یافته (Structural Data Types)

➤ داده‌های اشاره‌گر (Pointer Data Types)

داده‌های ساده (Simple data type) ➤

- نوع کاراکتری (Char type)
- صحیح (integer)
- اعشاری (حقیقی)
- نوع رشته ای (String type)
- نوع منطقی (Boolean type)

• صحیح (integer)

| 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 |

اعشاری (حقیقی)

| نوع | محدوده | تعداد ارقام معنی دار | اندازه بر حسب بایت |
|----------|---|----------------------|--------------------|
| Real | $2.2 \times 10^{-39} \dots$ 1.7×10^{38} | 11-13 | 6 |
| Single | $1.5 \times 10^{-45} \dots$ 3.4×10^{37} | 7-8 | 4 |
| Double | $5.0 \times 10^{-324} \dots$ 1.7×10^{308} | 15-16 | 8 |
| Extended | $3.4 \times 10^{-4932} \dots$ 1.1×10^{4932} | 19-30 | 10 |
| Comp | $-9.2 \times 10^{18} \dots$ 9.2×10^{18} | 19-30 | 8 |

➤ داده‌های ساخت‌یافته (Structural Data Types)

انواع داده‌های ساخت‌یافته عبارتند از:

آرایه‌ها
رکوردها
مجموعه‌ها
فایلها

➤ داده‌های اشاره‌گر (Pointer Data Types)

ممکن است در نوشتن برنامه، نوع داده‌های بحث شده در بالا به دلایل مختلف از جمله مشخص نبودن تعداد ورودیهای مسئله و غیره مشخص نباشد لذا نیاز به متغیرهایی هست که بتوانند آدرس متغیرهای دیگر را در خود نگه دارند، این نوع داده‌ها، داده‌های اشاره‌گر نام دارند .

❖ متغیرها (Variables)

متغیر، محلی از حافظه است که دارای نوع و اسم می باشد. نوع متغیر همان نوع داده بوده و اسم متغیر از قواعد اسم گذاری شناسه تبعیت می کند.

در پاسکال برای معرفی متغیرها بصورت زیر عمل می کنند:

Var (کلمه ذخیره شده)

نوع متغیر: اسم متغیر ;

❖ ثابتها (Constants)

یک ثابت نام شناسه‌ای است که در آغاز یک برنامه، یک مقدار در آن جاگزین می‌شود. درست مانند متغیرها. ثابت‌ها را می‌توان بعنوان خانه‌هایی از حافظه در نظر بگیریم که مقدار داده‌ها در آنها ذخیره می‌شود ولی مقدار ثابت مشخص می‌باشد، طوری که نمی‌توان مقدار یک ثابت را در برنامه خود بوسیله یک دستور تغییر داد.

برای تعریف یک ثابت بصورت زیر عمل می‌کنیم :

مقدار ثابت = اسم متغیر `Const`

❖ دستور جایگزینی

برای قرار دادن یک مقدار یا مقدار یک متغیر داخل یک متغیر دیگر، از دستور جایگزینی استفاده می کنند.

شکل کلی یک دستور جایگزینی در پاسکال بصورت زیر است:

عبارت محاسباتی = : اسم شناسه
یا
عبارت قیاسی
عبارت منطقی

❖ افزودن توضیحات به برنامه (Comment)

افزودن مطلب توضیحی در درون خود برنامه عملی پسندیده و مطلوب است بدین ترتیب که بعد از مدتی امکان فراموشی کار با برنامه از بین می‌رود و در کل می‌توان گفت که نوشتن توضیحات در برنامه خوانایی آن را بالا می‌برد.

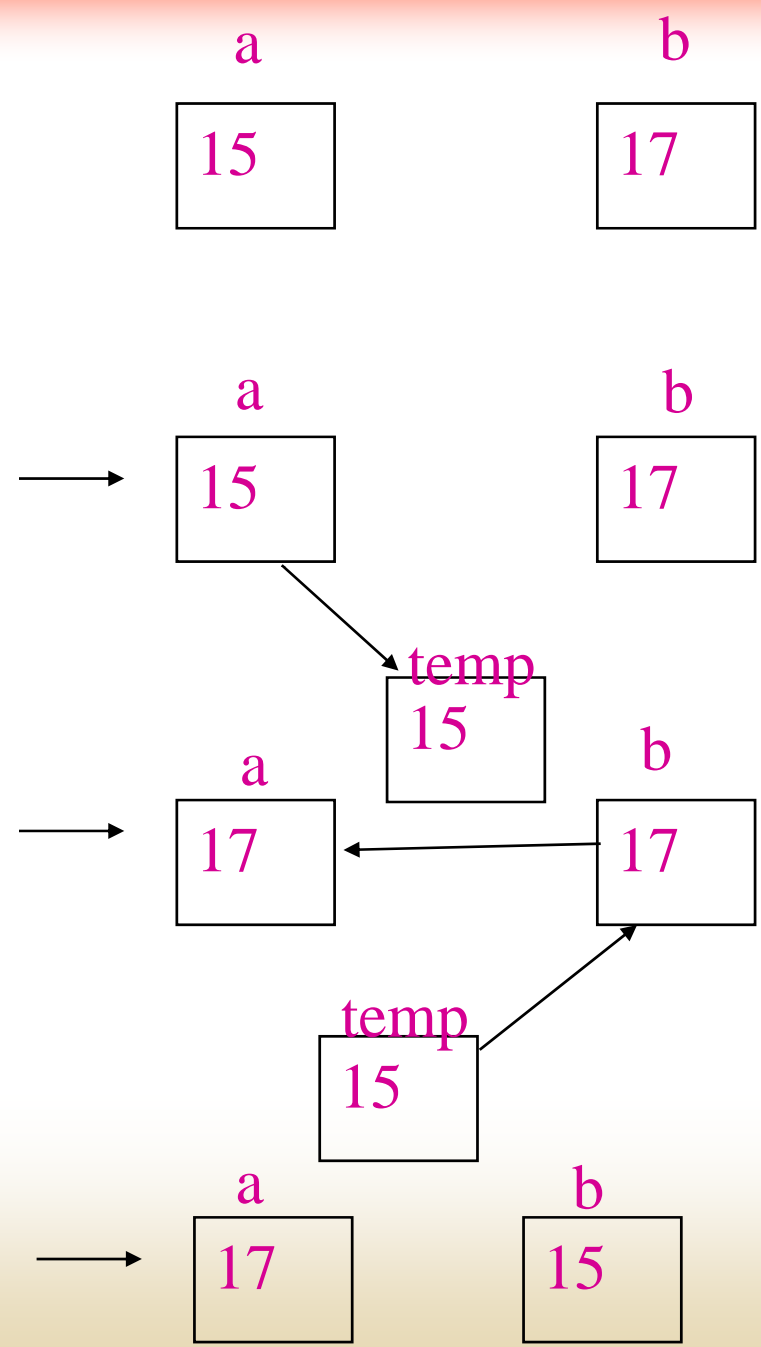
در پاسکال توضیحات بین دو آکولاد محصور می‌شوند.

```
{ This is comment }
```

```
    { This program written by A. Pascal }
```

مثال ❖

جابجا کردن مقدار دو متغیر



```
Program      Example4 ;  
  Var  
    a , b , temp : integer  
Begin  
  a := 15 ; b := 17 ; temp := 0 ;  
  temp := - a ;  
  a := b ;  
  b := temp ;  
  Writeln ( ' a = ' , a , ' b = ' , b ) ;  
End . { End . of program }
```



خروجی برنامه :

a = 17 b = 15

❖ نکاتی چند در مورد برنامه‌نویسی

- استفاده از اسامی با مفهوم برای متغیرها
- استفاده از دستور **Const** در صورتی که مقدار ثابت در برنامه وجود داشته باشد.
- سوال جوابی بودن برنامه (ورودی‌ها و خروجی‌ها باید دارای پیغام مناسب باشند)
- نوشتن برنامه با فرمت مناسب (رعایت قرار گرفتن خطوط مختلف برنامه زیر هم و فاصله گذاشتن آنها از

❖ تمرینات

حاصل عبارتهای زیر را بدست آورید:

A) $3 * 13 \text{ Mod } 3 \text{ div } 3 = ?$

B) $7.3 * 5 / 3 = ?$

C) $(3 + 4 < 6) \text{ and } (4 + 7 < 13) = ?$

D) $33 - 8 * 3 \text{ div } 3 \text{ mod } (5 \text{ div } 3) = ?$

E) NOT (((3 - 4 MOD 3) < 5) and ((6 div 4) < > 3))



حاصل عبارتهای منطقی را به ازاء مقادیر زیر مشخص کنید :

A: = true ; B: = false ; C: = true ;

A) (A AND B) OR (A AND C) = ?

B) (A OR NOT B) and (Not A OR C) = ?

C) A OR B AND C = ?

D) NOT (A OR B) AND C = ?

❖ تمرینات برنامه‌نویسی

- برنامه‌ای بنویسید که ابعاد مثلث که عبارتند از 1.3 , 3 را در نظر گرفته محیط و مساحت آن را محاسبه و با پیغام مناسب در خروجی چاپ کند.
- برنامه‌ای بنویسید که دو متغیر صحیح با مقادیر 15 , 3 را در نظر گرفته محتویات دو عدد را بدون استفاده از متغیر کمکی جابجا نماید.
- برنامه‌ای بنویسید که سه عدد بنام های First , Second , Third بترتیب با مقادیر 13 , 15 , 17 را در نظر گرفته بطور چرخشی مقادیر آنها را جابجا نموده در خروجی با پیغام مناسب چاپ کند.



فصل 6

ورودی و خروجی

دانشگاه پیام نور
دانشکده فناوری
اطلاعات (گروه
مهندسی
کامپیوتر)

هدفهای کلی

معرفی دستورات خروجی `WriteLn`

معرفی خروجی فرمت بندی شده

بررسی دستورات ورودی `ReadLn` و `Read`

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- عبارات مورد نظر را در خروجی نمایش دهد.
- عبارات خروجی را با فرمت مناسب چاپ کند.
- متغیرهای مورد نیاز برنامه را از ورودی دریافت نماید.
- برنامه‌های ساده به زبان پاسکال بنویسد.

❖ خروجی با دستور Write

این دستور برای نوشتن اطلاعات در خروجی بکار می‌رود. اطلاعات خروجی می‌توانند ثابت‌های عددی، مقادیر متغیرها، عبارات و غیره باشند. شکل دستور در حالت کلی بصورت زیر است:

(..... و متغیر ۲ و متغیر ۱) write

یا (..... و 'عبارت ۲' و 'عبارت ۱')

یا (..... و ثابت ۲ و ثابت ۱)

مثال ❖

Var

A , B: integer ;

Ch: char ;

R: Real ;

Begin

A: = 10 ; B: = 15 ;

Ch: = ' T ' ;

R: = 12.25

Write (' A = ' , A , ' B = ' , B) ;

Write (' ch = ' , ch , ' R = ' , R) ;

Write (' sum of A and B = ' , A + B) ;

End. { end of program }

بعد از اجرای برنامه فوق در خروجی خواهیم داشت:

A = 10 B = 15 ch = T R = 1.2250000000 e + 01 sum of A and B = 25

❖ خروجی با دستور *Writeln*

این دستور همانند دستور **Write** عمل می کند 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 بعد می شود.

مثال ❖

```
Var
  A , B: integer ;
  Ch: char ;
  R: Real ;
Begin
  A: = 10 ; B: = 15 ;
  Ch: = ' T ' ;
  R: = 12.25 ;
  Writeln ; { new line }
  Writeln ( ' A = ' , A , ' B = ' , B ) ;
  Writeln ( ' Ch = ' , ch , ' R = ' , R ) ;
  Writeln ( ' Sum of A and B = ' , A + B ) ;
End. { End of program }
```

خروجی برنامه بصورت زیر می باشد:

```
A = 10   B = 15
Ch = T   R = 1.225000000 e + 01
Sum of A and B = 25
```

❖ خروجی فرمت‌بندی شده

اگر بخواهیم اطلاعات با فاصله های مشخص یا در مکان مشخصی در صفحه نمایش قرار گیرند، باید فرمت چاپ را در دستورات بیان شده مشخص کنیم.

طریقه تعیین فرمت چاپ برای اعداد صحیح

فرمت اعداد صحیح بصورت زیر مشخص می شود:

Write یا **WriteLn** (طول میدان: داده صحیح)

در تعریف طول میدان برای متغیرها یا داده‌هایی از نوع صحیح به نکات زیر توجه کنید:

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

❖ مثال

```
X:= 3200 ;  
A: = 12 ;  
B: = 217 ;  
Write ( X:3 , A:5 , B:5 ) ;
```

خروجی :

3200 12 217

• طول میدان اعداد اعشاری

برای نمایش اعداد اعشاری بصورت دلخواه، می توان با تعریف طول میدان و تعداد ارقام اعشاری، عدد مزبور را نمایش داد

در حالت کلی طول میدان را می توان به صورت زیر تعریف کرد :

Write (تعداد ارقام بعد از ممیز: طول میدان: متغیر اعشاری)
یا **Writeln**



در تعریف فرمت برای اعداد اعشاری به نکات زیر باید توجه کرد:

اگر طول میدان بزرگتر از تعداد ارقام عدد ذکر شود، عدد در منتهی الیه سمت میدان چاپ می شود. راست

اگر فقط طول میدان ذکر شود، عدد به صورت نماد علمی در طول میدان مشخص شده چاپ می شود .

از آنجائی که برای نمایش اعداد در نماد علمی حداقل ۸ محل مورد نیاز است، لذا هنگامی که تنها طول میدان ذکر شده باشد، اگر از ۸ رقم کمتر باشد، حداقل ۸ رقم در نظر گرفته می شود.

✓ 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.

✓ 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.

• طول میدان کاراکترها و رشته‌ها

برای نمایش رشته‌ها و کاراکترهای با طول میدان بصورت زیر عمل می‌کنیم.

Write (طول میدان: متغیر یا عبارت رشته‌ای یا کاراکتری)

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

❖ ورودی با Readln , Read

از این دستور برای خواندن داده ها و اختصاص آنها به متغیرها استفاده می شود. در خواندن داده ها به دو موضوع باید دقت شود:

۱- منبع داده ها یعنی دستگاه ورودی که از آن داده ها خوانده می شود.

۲- متغیری که داده های خوانده شده در آن قرار می گیرد .

❖ شکل کلی دستور ورودی Read بصورت زیر می باشد:

Read (..... و متغیر ۲ و متغیر ۱)

این دستور عمل خواندن داده‌ها و ذخیره آنها در متغیرها را انجام می‌دهد

و پس از اتمام عمل خواندن کنترل را برای خواندن و نوشتن‌های بعدی در همان خط نگه می‌دارد .

مثال جابجا کردن محتویات دو عدد

```
program Example_1 ( input , output ) ;  
Var  
    first , second , temp: integer ;  
Begin  
    Writeln ;  
    Writeln ( 'Please Enter two numbers ' ) ;  
    Readln ( first , second ) ;  
    Temp: = first ;  
    First: = second ;  
    Second: = temp ;  
    Write ( ' first = ' , first , ' second = ': 10 , second ) ;  
End. { end of program }
```



خروجی برنامه بالا به صورت زیر می باشد:

Please Enter two numbers

15 17

First = 17 second = 15

❖ تمرینات

خروجی قطعه برنامه زیر را تعیین کنید :

```
Value1: = 27.3 ;
```

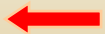
```
Value2: = -8.5 ;
```

```
Writeln ( ' Value1 is ' , Value1 ) ;
```

```
Writeln ( ' Value2 is ' , Value2 ) ;
```

```
Sum: = Value1 + Value2 ;
```

```
Writeln ( ' Sum of Two Values - ' , Sum: 6: 2 ) ;
```



اگر متغیر X از نوع real و مقدار آن 12.235 و متغیر 3 از نوع صحیح و مقدار آن 100 باشد خروجی دستورات زیر را تعیین کنید؟

```
Writeln ( ' X is ': 10 , X: 6: 2 , ' I is ': 4 , I: 5 ) ;
```

```
Writeln ( ' I is ': 10 , I: 1 ) ;
```

```
Writeln ( ' X is ': 10 , X: 2: 1 ) ;
```

```
Writeln ( ' X is ': 15 , X: 7: 1 ) ;
```

```
Writeln ( ' I is ': 10 , ' X is ': 10 , X: 7: 3 ) ;
```

❖ تمرینات برنامه نویسی

برنامه‌ای بنویسید که دو عدد را از ورودی دریافت کرده و محتویات آنها را بدون استفاده از متغیر کمکی جابجا نماید.

برنامه‌ای بنویسید که سه عدد صحیح **Third , Second , first** را از ورودی با پیغام مناسب دریافت کرده سپس محتویات این سه متغیر را بصورت چرخشی جابجا نموده با پیغام مناسب در خروجی چاپ کند.



برنامه‌ای برای یک حسابدار اداره جمع آوری مالیات بنویسید که صورت حسابهای مالیات را محاسبه نماید.

ورودی:

شماره شناسایی مالیات دهنده

بهای ارزیابی شده

نرخ مالیات

خروجی:

صورت حساب بافرمت مناسب شامل تمام داده‌های ورودی و میزان بدهی

فصل 7

ساختارهای شرطی و کنترلی

هدفهای کلی

معرفی دستور شرطی **If and Else**

معرفی دستور **case**

بررسی دستورات تکرار **for** ، **while** ، **repeat until**

معرفی دستورات شرطی متداخل

بررسی چند تابع و روال استاندارد زبان پاسکال

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- برنامه‌هایی را بنویسد که در آنها نیاز به استفاده از شرط وجود دارد.
- تفاوت‌های بین دستورات مختلف با `if` و `if-else` را تشخیص دهد.
- برنامه‌هایی که نیاز به تکرار تعدادی عملیات داشته باشند را بنویسد.
- در صورت نیاز بتواند در برنامه‌ها، از روالها و توابع استاندارد زبان استفاده نماید

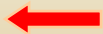
❖ دستورات شرطی

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

دستورات شرطی در حالت کلی به دو نوع تقسیم می شوند:

دستور `if`

دستور `Case`



• دستور if

هرگاه در طول برنامه نیاز به استفاده از شرط یا شروط داشته باشیم، از دستور **If** استفاده می‌کنیم.

دستور **if** بطور کلی به سه شکل بر حسب نیاز ممکن است ظاهر شود :

if ساده

if همراه **Else**

ifهای متداخل

If – then •

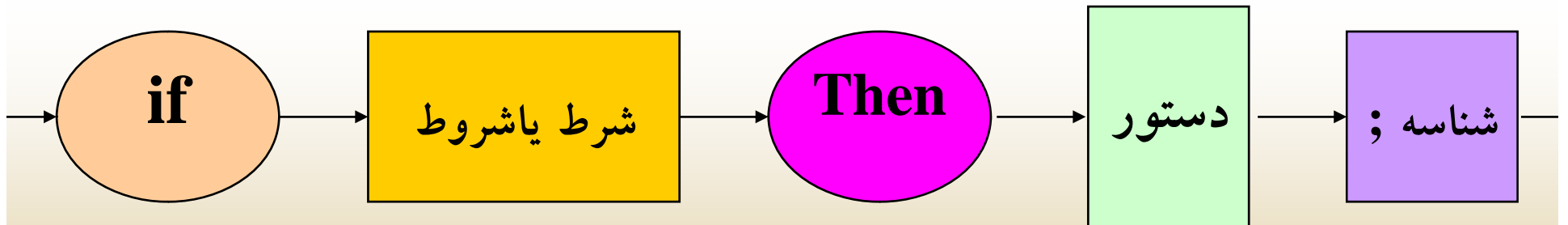
در این نوع دستور شرطی اگر شرط خاصی تحقق یافته باشد، عمل یا اعمال خاصی انجام می‌شود. در غیر اینصورت برنامه روال عادی خود را طی می‌کند، در صورتی که شرط برقرار باشد ارزش منطقی **Ture** به خود می‌گیرد و اگر شرط برقرار نباشد، ارزش منطقی **False** به خود خواهد گرفت.

• If – then (ادامه)

شکل کلی دستور if بصورت زیر می باشد:

if شرط یا شروط then
; دستور

دیاگرام دستور بالا بصورت زیر می باشد:



• مثال

```
Program Example ;  
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
```

• دستور if همراه 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 حالت‌ها را تعیین می‌کنند.

شکل کلی این دستور بصورت زیر است:

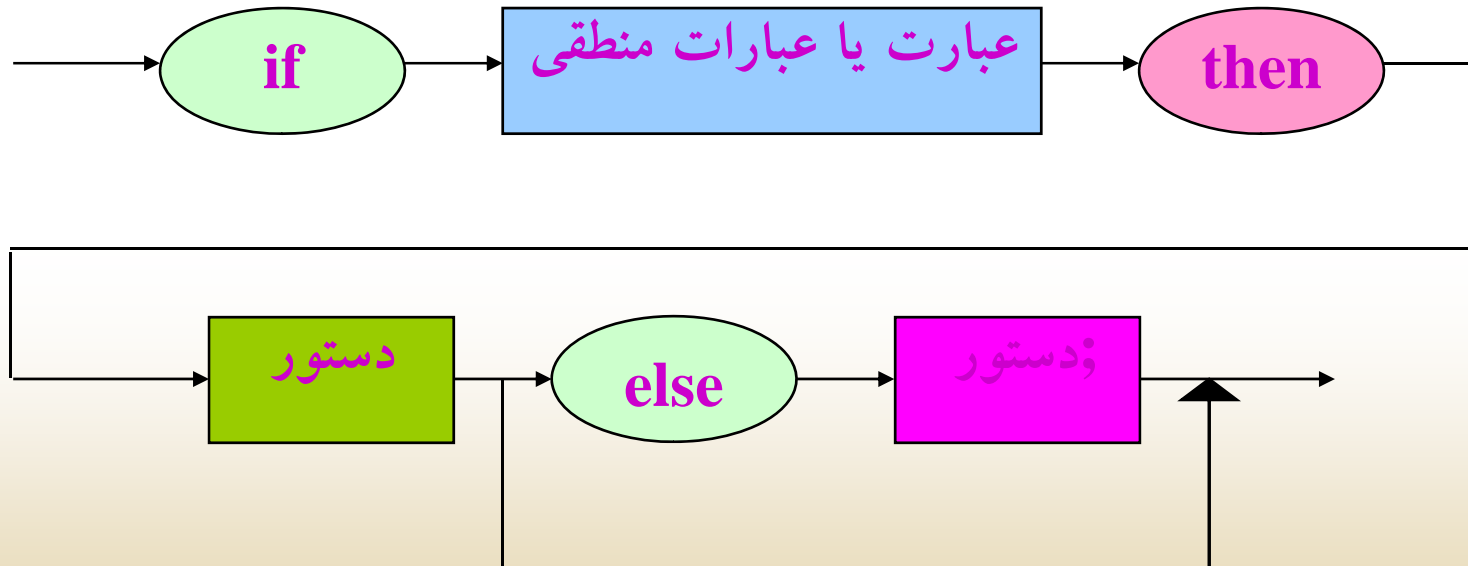
if عبارت یا عبارات منطقی **then**

؛ دستور ۱

else

؛ دستور ۲

دیاگرام دستور بصورت:



• مثال

```
Program      Example ;  
Var  
    Number: integer ;  
Begin  
    Writeln ( 'Please enter Number: ' ) ;  
    Readln ( Number ) ;  
    if ( Number mod 2 ) = 0 Then  
        Write ( ' even ' )  
    else  
        Write ( ' odd ' ) ;  
End. { end of program }
```

خروجی برنامه بالا بصورت زیر است:

```
Please enter Number:  
17  
odd
```

If عبارت منطقی Then
Begin

دستور 1 ;

دستور 2 ;

•
•
•

End

Else

Begin

دستور 1 ;

دستور 2 ;

•
•
•

End ;

صورت دیگر دستور



• If های متداخل

هرگاه در نوشتن برنامه نیاز به انتخاب یک شرط از بین چند شرط داشته

باشیم، معمولاً از **If** متداخل استفاده می کنند. در چنین مواقعی استفاده از **If**

متداخل کارائی برنامه را بالا می برد زیرا بجای کنترل تمام شروط فقط تا

زمانیکه شرط برقرار نشده، **If** ها بررسی می شوند. بعد از برقرار شدن یکی

از شروط، کنترل برنامه به بعد از **If** منتقل می شود و این در بهبود کارائی

یک برنامه می تواند بسیار موثر باشد.

در حالت کلی If متداخل به صورت های زیر ممکن است، در برنامه ظاهر شود.

| | |
|----------------------------|---------------------------------|
| if عبارت شرطی 1 then | ۲) if عبارت شرطی 1 then |
| if عبارت شرطی ۲ then | دستور 1 |
| دستور 1 | else if عبارت شرطی 2 then |
| else | دستور 2 |
| ; دستور 2 | else If عبارت شرطی 3 then |
| | دستور 3 |
| | else |
| | . |
| | . |
| | . |



مثال : برنامه‌ای بنویسید که نمره دانشجویی را از ورودی دریافت کرده، با توجه به مقدار نمره یکی از خروجی‌های زیر را نمایش دهد:

| Grade | خروجی |
|----------------|--------------|
| 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 Case ➤

زبان پاسکال دستور Case را بصورت زیر در نظر می‌گیرد:

```
Case      عبارت      Of
  مقدار 1 : دستور 1 ;
  مقدار 2 : دستور 2 ;
  مقدار 3 : دستور 3 ;
  .
  .
  .
  Otherwise
      دستور ;
End ; { End of case }
```

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: ')

Readln (op) ;

Case op of

' + ': Writeln (' Sum = ' , (a + b): 6: 2) ;

' - ': Writeln (' Subtract = ' , (a - b): 6: 2) ;

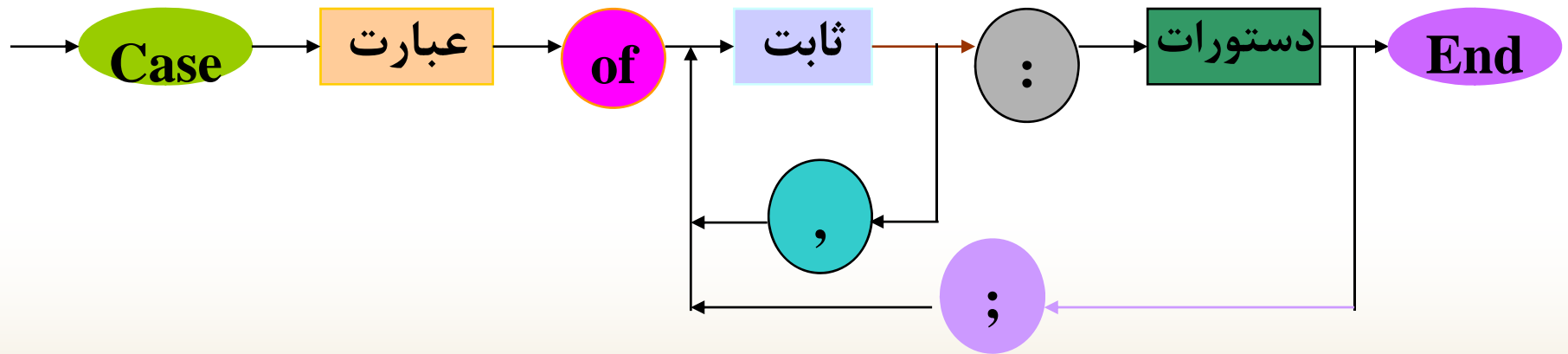
' * ': Writeln (' Multiple = ' , (a * b): 6: 2)

' / ': Writeln (' divide = ' , (a / b): 6: 2) ;

End; { End of Case }

End. { end of program }

دیاگرام دستور Case بصورت زیر می باشد:



❖ ساختارهای کنترلی

بسیاری از مواقع لازم است عمل یا اعمال به تعداد دفعات معین یا نامعین انجام شوند.

در چنین مواقعی زبانهای برنامه‌نویسی دستوراتی دارند که می‌توان این اعمال تکراری

را انجام داد. در حالت کلی ساختارهای کنترلی شامل یک یا چند شرط و همچنین

متغیر یا اصطلاحاً شمارنده‌ای برای پایان دادن به شرط می‌باشند

شکل کلی حلقه بصورت زیر می باشد:

Do مقدار نهایی To مقدار اولیه =: اندیس For

; دستور

باید توجه داشته باشید که در حلقه تعداد تکرار کاملاً مشخص است و حلقه دقیقاً به تعداد تکرار مشخص اجرا می شود.

✓ حلقه for

این دستور برای انجام عمل یا اعمالی مشخص به تعداد تکرار معین بکار برده می‌شود.

حلقه for شامل یک اندیس (index) مقدار اولیه (initial value) مقدار نهایی

(final value) و مقدار افزاینده می‌باشد. این حلقه با قرار دادن مقدار اولیه در

اندیس حلقه شروع شده و بعد از هر تکرار یک واحد به اندیس حلقه اضافه می‌کند تا

در نهایت به مقدار نهایی برسد. شکل کلی حلقه بصورت زیر می‌باشد:

مثال : 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 }
```

دستور for را بصورت زیر هم می توان بکار برد.

do مقدار نهایی downto مقدار اولیه = : اندیس for

; دستور

از این شکل از دستور For ابتدا مقدار اولیه در اندیس حلقه قرار داده می شود .
بعد از آن در هر تکرار حلقه یک واحد از اندیس حلقه کم می شود تا به
مقدار نهایی برسد .

مثال : برنامه‌ای بنویسید که عدد صحیحی را از ورودی دریافت کرده و فاکتوریل آن را محاسبه نماید.

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 .

For های متداخل

do مقدار نهایی to مقدار اولیه = : اندیس 1 for

do مقدار نهایی to مقدار اولیه = : اندیس 2 for

; دستور

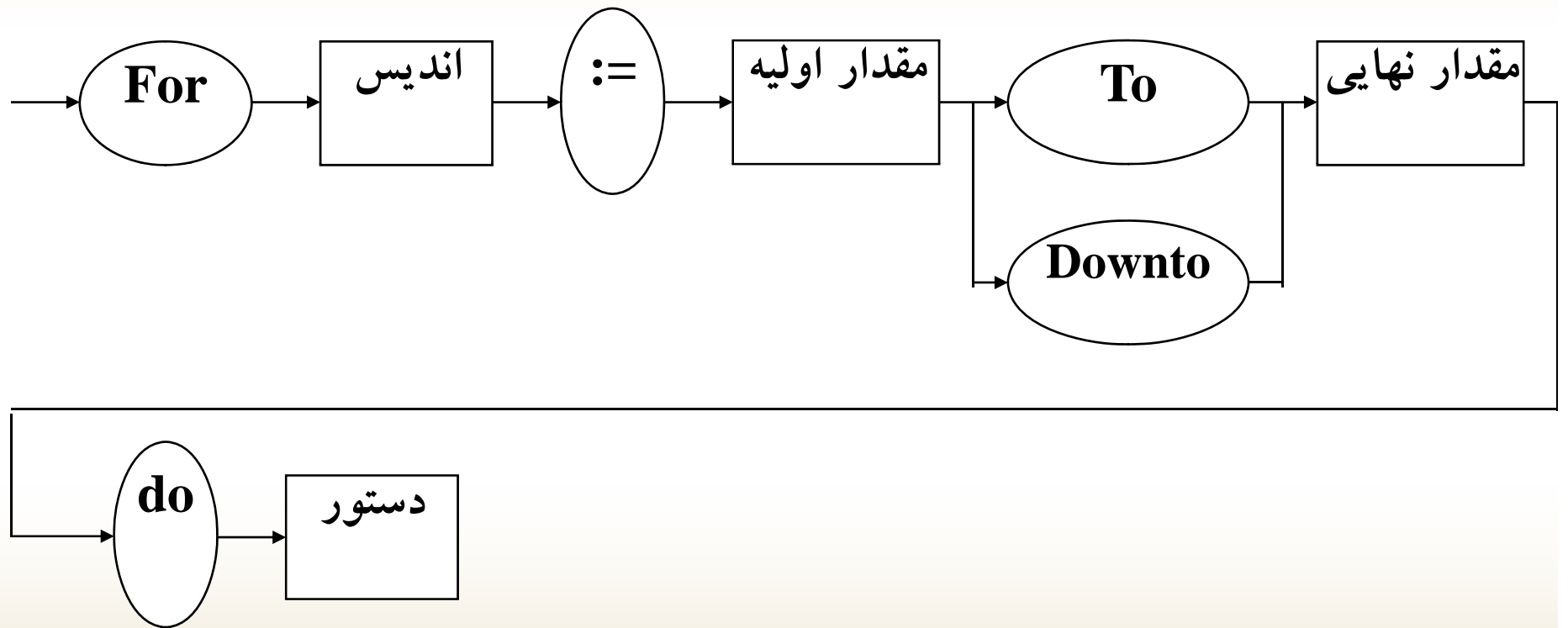
مثال :

```
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) |

✓ دیاگرام دستور for



✓ حلقه While

در حالت کلی هدف از بکار بردن این دستور انجام عملیاتی مشخص به تعداد دفعات نامعین است

این حلقه به صورت زیر بکار برده می شود:

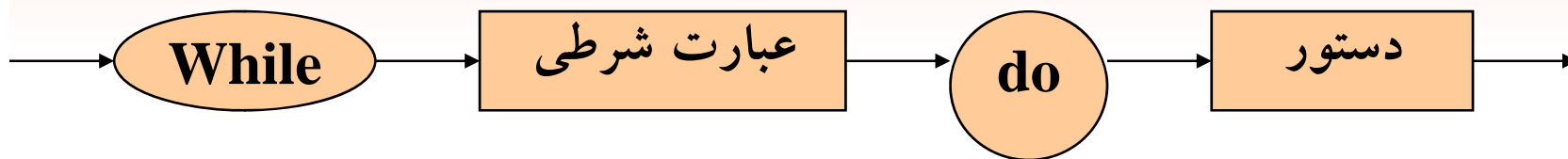
Do عبارت منطقی **While**

؛ دستور

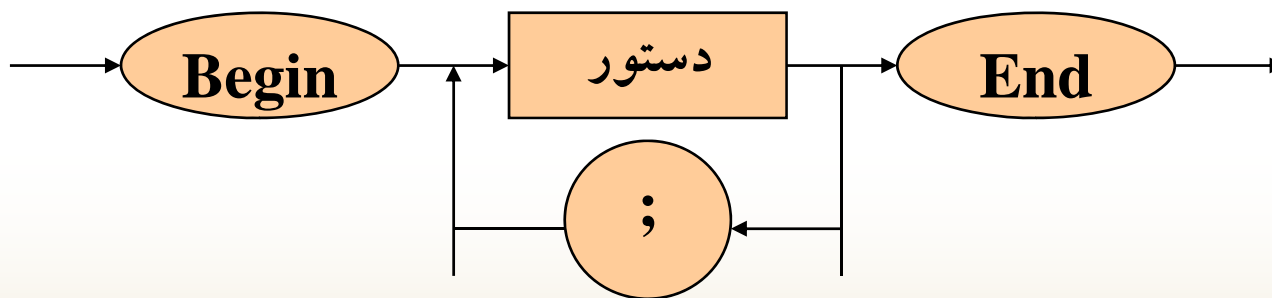
مثال : بزرگترین مقسوم علیه دو عدد

```
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 }
```

✓ **دیاگرام دستور While بصورت زیر می باشد:**



و در صورتی که دستور مرکب باشد دیاگرام بصورت زیر است :



✓ دستور Repeat

این دستور نیز از نوع دستورات تکراری می باشد و به کمک آن می توان یک یا چند دستور را به تعداد نامعین بار اجرا کرد.

این دستور مشابه دستور **while** است، با تفاوت هایی که در زیر عنوان می کنیم:

(۱) در دستور **Repeat** برعکس دستور **While**

شرط حلقه در انتهای حلقه بررسی می شود لذا حلقه حداقل یکبار اجرا می شود.





(۲) دستور **Repeate** تا زمانی اجرا می شود که شرط خاصی تحقق پیدا نکرده است در حالیکه دستور **While** تا زمانی که شرط برقرار باشد، اجرا می شود.

(۳) دستور **Repeat** نیاز به بلوک ندارد و همراه **Until** ظاهر می شود.

❖ شکل کلی این دستور بصورت زیر می باشد:

Repeat

دستور 1 ;

دستور 2 ;

.

.

until شرط یا شروط ;

برنامه مجموع و میانگین تعدادی عدد صحیح مثبت

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 }

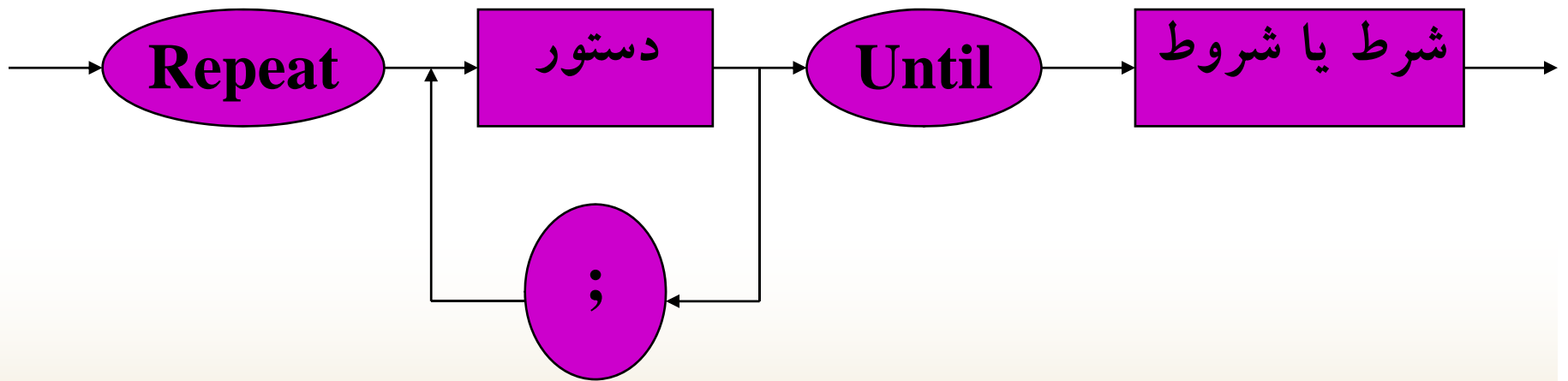
 i: = i - 1

 ave: = Sum / i ;

Writeln (' Sum = ', Sum , ' average = ': 12 , ave: 7: 2) ;

end.{ end of program }

دیاگرام دستور Repeat بصورت زیر می باشد:



❖ معرفی چند پروسیجر (Procedure)

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 یکبار در برنامه گنجانده شده ولی در محل‌های مختلف از آن استفاده به عمل می‌آید و از اصول برنامه‌نویسی ساخت یافته‌است.

✓ پروسیجر Exit

هدف: انتقال کنترل برنامه به خارج از بلوک فعلی

Procedure Exit ;

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

پروسیجر Break ✓

هدف: خاتمه دادن به اجرای یک حلقه

Procedure Break ;

استفاده از پروسیجر فوق باعث می شود که اجرای یک حلقه خاتمه یافته و کنترل برنامه به دستورالعمل بعدی انتقال یابد.

✓ پروسیجر continue

هدف: بازگشت به ابتدای حلقه

Procedure continue ;

وقتی این پروسیجر در حلقه ظاهر می شود کنترل برنامه به اول حلقه انتقال می یابد و دستورات بعد از پروسیجر اجرا نمی شوند.

ارائه چند مثال از کاربرد حلقه ها و شرطها

مثال : برنامه‌ای بنویسید که یک عدد صحیح در مبنای ده را از ورودی دریافت کرده، به یک عدد در مبنای ۲ 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 }



مثال : برنامه‌ای بنویسید که n عدد را از ورودی

دریافت کرده، n جمله سری زیر که به سری فیبوناچی معروف است 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 }

تمرینات ❖

a)

```
Sum := 0 ;  
While i <= 120 do  
Begin  
    Sum := Sum + i ;  
    i := i + 1 ;  
end ;
```

۱- خروجی قطعه برنامه‌های زیر را تعیین کنید:

b)

```
i := 0 ;  
Sum := 0 ;  
While i <= 20 do  
Begin  
    i := i + 1 ;  
    Sum := Sum + i ;  
end ;
```



c)

$b := 5$;

Repeat

 Writeln (b , ($b \text{ div } 5$) : 3) ;

$b := b - 1$;

Until ($b \text{ div } 3$) = 5 ;

d)

Count := 0 ;

Stop := 4

;

While Count < Stop Do

Begin

 For K := 1 to Count Do

 Write (K : 3) ;

 Writeln ;

 Count := Count + 1 ;

End .

❖ تمرینات برنامه نویسی

۱- برنامه‌ای بنویسید که با استفاده از حلقه‌ها خروجی زیر را تولید کند.

```
          1
        1 2 1
      1 2 3 2 1
    1 2 3 4 3 2 1
  1 2 3 4 5 4 3 2 1
```

۲- برنامه‌ای بنویسید که تعدادی عدد از ورودی دریافت کرده مجموع ارقام هر عدد را در خروجی چاپ نماید. (پایان داده ها به ۱- ختم می شود)

۳- برنامه‌ای بنویسید که دو عدد صحیح را از ورودی دریافت کرده سپس: اعداد فیبوناچی بین این دو عدد را چاپ کند.

۴- برنامه‌ای بنویسید که یک اسکناس ۱۰۰۰ تومانی را به حالت‌های مختلف یعنی به اسکناس ۲۰۰ تومانی، ۱۰۰ تومانی، ۵۰ تومانی، ۲۰ تومانی، ۱۰ تومانی و سکه های ۵ تومانی و ۲ تومانی و یک تومانی خرد نماید.

فصل 8

آرایه ها Array

هدفهای کلی

- شناخت لزوم استفاده از ساختار داده ای به نام آرایه
- شناخت انواع آرایه ها و موارد استفاده از آنها
- شناخت مفاهیم مرتب سازی و جستجو
- شناخت الگوریتمهای مرتب سازی و جستجو

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- آرایه‌ها یک بعدی را در برنامه‌های خود بکار ببرد.
- ماتریسها را پیاده‌سازی نماید.
- عمل جستجو در آرایه انجام دهد.
- یک لیست را توسط روشهای مرتب سازی حبابی، انتخابی و غیره مرتب کند.

❖ آرایه و انواع آن

خانه‌های پشت سرهم از حافظه که هم‌نوع بوده و توسط یک اسم معرفی می‌شوند، آرایه نام دارد. نحوه دسترسی به هر یک از اعضاء آرایه از طریق اندیس آرایه امکان‌پذیر است.

برای تعریف آرایه ابتدا طول آرایه که در حقیقت تعداد خانه‌های آنرا مشخص می‌کند، معین می‌گردد. سپس نوع خانه‌هایی که داده‌ها در آن قرار خواهند گرفت را تعیین می‌کنند

❖ آرایه‌های یک بعدی

آرایه‌های یک بعدی بصورت زیر تعریف می‌شوند:

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

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

برای مثال :

Var

No: Array [1.. 50] of integer ;

id: Array [1.. 20] of Byte ;

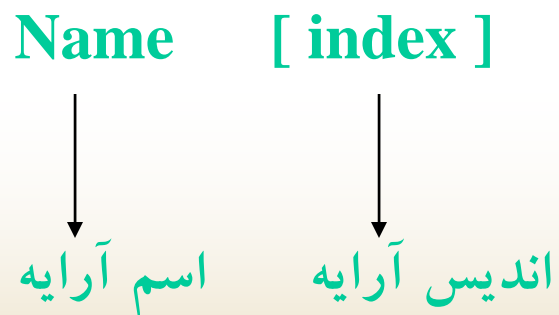
Name , Fam: Array [1.. 30] of char ;

مقداردهی آرایه‌ها مثل متغیرها به دو صورت امکان‌پذیر است :

۱- با استفاده از دستورات ورودی

۲- مقداردهی در طول برنامه

طریقه دسترسی به عناصر آرایه بصورت زیر می‌باشد:



مثال : برنامه‌ای بنویسید که ۱۰۰ عدد صحیح از ورودی دریافت کرده، بیشترین مقدار و محل وقوع آن را در خروجی چاپ نماید.

Var

No: array [1.. 100] of integer ;Max , i , index: integer ;

Begin

Writeln (' please enter TEN Numbers ') ;

For i:= 1 to 100 do

 Readln (No [i]) ;

 Max:= No [1] ; Index:= 1 ;

 For i:= 2 to 100 do

 If No [i] > Max Then

 Begin

 Max:= No [i] ;

 Index:= i ;

 End ;

 Writeln (' The Maximum is = ' , Max) ;

 Writeln (' And Index = ' , Index) ;

End. { End of program }

مثال: برنامه‌ای بنویسید که یک آرایه 100 عنصری را از ورودی دریافت کرده عناصر آرایه را معکوس نماید.

Var

i , temp : integer ; a : array [1 .. 100] of integer ;

Begin

Writeln (' please enter 100 Numbers : ') ;

For i := 1 to 100 do Begin

 Read (a [i]) ;

 If (i Mod 10) = 0 Then

 Writeln ;

End ;

For i := 1 to 50 do

Begin

 Temp := a [101 - i] ;

 a [101 - i] := a [i] ;

 a [i] := temp ;

End ;

End .

آرایه‌های دو بعدی

برای نمایش ماتریس در حافظه معمولاً از آرایه‌هایی بنام آرایه‌های دوبعدی استفاده می‌کنند
برای درک بیشتر این آرایه‌ها آنها را بصورت ماتریس در نظر می‌گیرند.

آرایه‌های دو بعدی بصورت زیر معرفی می‌شوند:

Name : array [1 .. row , 1 .. column] of type ;



مثال : برنامه‌ای بنویسید که یک ماتریس 5×5 را از ورودی دریافت کرده، سپس آنرا در خروجی چاپ نماید.

Var

```
a : array [ 1 .. 5 , 1 .. 5 ] of integer ;  
i , j : integer ;
```

Begin

```
For i := 1 to 5 do
```

```
Begin
```

```
For j := 1 to 5 do
```

```
Read ( a [ i , j ] ) ;
```

```
Writeln ;
```

```
End ;
```

```
Writeln ;
```

```
For i := 1 to 5 do Begin
```

```
For j := 1 to 5 do
```

```
Write ( a [ i , j ] : 5 ) ;
```

```
Writeln ;
```

```
End ;
```

```
End.
```

آرایه‌های چند بعدی

می‌توان آرایه‌هایی با ابعاد بیشتر از دو نیز تعریف کرد. بطور کلی برای معرفی یک آرایه چند بعدی می‌توان بصورت زیر عمل کرد :

Name : array [1.. length1] of array [1.. length2]

Of array [1.. lengthN] of Type

و یا

Name: array [1.. length1 , 1.. length2 , 1.. lengthN] of Type

❖ نکاتی چند در مورد آرایه‌ها

✓ تعریف آرایه با محدوده منفی

برای مثال:

A: array [-10.. 10] of Real ;

✓ تعریف آرایه از نوع منطقی (Boolean)

برای مثال:

A: array [1.. 20] of Boolean ;



✓آرایه‌های با محدوده منطقی:
برای مثال



A: array [Boolean] of integer;

✓آرایه‌ای با محدوده کاراکتری:
برای مثال

A: array ['A'..'Z'] of Real ;

✓آرایه‌ای که برای اعضای آن محدودیت قائل شویم
برای مثال:

A: array [1.. 20] of 1.. 30

❖ جستجو و مرتب‌سازی (Search and Sort)

یکی از مسائلی که در بحث طراحی الگوریتم بسیار مهم است، بحث مرتب‌سازی و جستجو می باشد. منظور از جستجو اینست که یک مقداری را از یک لیست جستجو کنیم و منظور از مرتب سازی اینست که یک لیست مرتب از داده ها را ایجاد کنیم. حال تعدادی الگوریتم که برای مرتب‌سازی و جستجو بکار می‌روند را بررسی می‌کنیم بخصوص زمانیکه ساختار داده ما یک آرایه باشد .

▪ جستجو در آرایه

در کل دو نوع عمل جستجو را در این کتاب بررسی می کنیم :

Linear search ➤ جستجوی خطی

Binary search ➤ جستجو دودویی

جستجوی خطی Linear search

در جستجوی خطی عبارت مورد جستجو را نخست با اولین عضو آرایه مقایسه می‌کنیم، اگر برابر بود عمل جستجو با موفقیت همراه بوده و عمل جستجو خاتمه می‌یابد در غیر اینصورت روند را ادامه داده و عبارت مورد جستجو را بترتیب با عضو دوم، سوم ... مقایسه می‌کنیم تا اینکه حالت تساوی حاصل شود و اگر این حالت حاصل نشد، عباریت مورد جستجو در لیست قرار ندارد.

قطعه برنامه زیر را می توان در حالت کلی برای جستجوی خطی بکار برد:

```
Flag: = True ;
```

```
i: = 0 ;
```

```
While ( i <= N ) and ( flag ) Do
```

```
    Begin
```

```
        i: = i + 1 ;
```

```
        if A [ i ] = x Then
```

```
            Begin
```

```
                Index: = i ;
```

```
                Flag: = false ;
```

```
            End ;
```

```
        End ;
```

```
    If flag Then
```

```
        Writeln ( ' The Element is found ' )
```

```
    Else
```

```
        Writeln ( ' The Element is not found ' ) ;
```

➤ جستجو دودویی Binary search

در جستجوی دودویی لیست اولیه باید مرتب باشد. برای جستجو در چنین آرایه‌ای نخست اندیس وسط آرایه را پیدا می‌کنیم و عنصر واقع در این اندیس را با عبارت مورد جستجو مقایسه می‌کنیم و حالات زیر ممکن است حاصل شود:

(**Low** اندیس ابتدای آرایه و **upper** اندیس آخرین عناصر آرایه و **middle** اندیس عنصر وسط می‌باشد.)



حالت اول :

if A [middle] < X Then

Low := middle

در اینصورت

و مقدار جدید **middle** را که عبارتست از:

Middle := (low + middle) / 2

حالت دوم :

if A [middle] > X Then

upper := middle

در اینصورت

و مقدار جدید **middle** را که عبارتست از:

Middle := (low + middle) / 2

حالت سوم :

if A [middle] = X Then

Write (' The Element is found ')

در اینصورت

در صورتی که حالت‌های الف یا ب اتفاق بیفتد، عمل جستجو را تا زمانی که $Low < upper$ می باشد ادامه می دهیم و در هر مرحله که حالت سوم رخ دهد عمل جستجو خاتمه می یابد.

مرتب‌سازی ➤

برای مرتب‌سازی داده‌ها روش‌های متفاوتی وجود دارد . تفاوت روش‌های مرتب‌سازی در زمان اجرای آنها می باشد. در حالت کلی با توجه به تعداد ورودیها (داده ها) و نوع مسئله مرتب‌سازی می توان از انواع روش‌های مرتب‌سازی استفاده نمود.

حال بعضی از روش های مرتب سازی عمومی را بررسی می کنیم.

- مرتب‌سازی حبابی (**Bubble sort**)
- مرتب‌سازی انتخابی (**Selection sort**)

• مرتب‌سازی حبابی (Bubble sort)

ساده‌ترین روش مرتب‌سازی روش مرتب‌سازی حبابی می‌باشد. یکی از خصوصیات بارز این نوع مرتب‌سازی این است که فهم آن ساده بوده و برنامه‌نویسی آن به سهولت انجام می‌گیرد. مرتب‌سازی حبابی نخست عنصر اول و دوم را با هم مقایسه می‌کند و در صورت نیاز، آنها را جابجا می‌کند، سپس عنصر دوم و سوم را مقایسه می‌کند. این عمل را تا زمانیکه به انتهای آرایه نرسیده تکرار می‌کند، در پایان مرحله اول بزرگترین عنصر در آخرین خانه آرایه قرار عمل بالا را انجام می‌دهد. این روند را $N - 1$ می‌گیرد. در مرحله دوم از خانه اول تا خانه تا زمانیکه تمام عناصر آرایه مرتب نشده‌اند ادامه می‌دهد و در نهایت یک لیست مرتب شده بصورت صعودی در خروجی تولید می‌شود.





```
For i:= 1 to n do
  For j:= 1 to n - 1 do
    If x [ j ] > x [ j + 1 ] Then
      Begin
        Temp:= x [ j ] ;
        X [ j ]:= x [ j + 1 ] ;
        X [ j + 1 ]:= temp ;
      End ;
```

تعداد مقایسه‌ها در این روش بصورت زیر محاسبه می‌شود:

$$n - 1 + n - 2 + \dots + 1 = N(N-1)/2$$

• مرتب‌سازی انتخابی (Selection sort)

در این روش مرتب‌سازی نخست کوچکترین عنصر را در کل آرایه پیدا کرده در خانه اول آرایه قرار می‌دهیم سپس عنصر کوچکتر بعدی را یافته در خانه دوم قرار می‌دهیم این روند را تا زمانی که کل آرایه مرتب نشده ادامه می‌دهیم



قطعه برنامه‌ای که برای مرتب‌سازی انتخابی می‌توان نوشت بصورت زیر می‌باشد:

```
For i: = 1 to n do
Begin
    Min: = x [ i ] ;
    Index: = i ;
    For j: = i + 1 to n do
        If x [ j ] < Min Then
            Begin
                Min: = x [ j ] ;
                Index: = j ;
            End ; { find The smallest Element }
    X [ index ]: = x [ i ] ;
    X [ i ]: = Min ; { swap Minimum With other Element }
End ; { end of selection sort }
```

❖ چند مثال در مورد آرایه‌ها

مثال : برنامه‌ای بنویسید که یک عدد صحیح از ورودی دریافت کرده سپس اعداد اول قبل از آن را تولید و در آرایه قرار دهد.

Var

a : array [1 .. 50] of integer ;

i , j , N : integer ;

k : Byte ;

flag : Boolean ;

Begin

Write (' Enter A Number : ') ;

Readln (N) ; a [1] := 2 ; a [2] := 3 ;

flag := True ;

k := 2 ;



```

For i := 4 to N do
Begin
    For j := 2 to (i div 2) do
        If (i mod j) = 0 Then
            Flag := false ;
    End ;
    If flag Then
        Begin
            K := k + 1 ;
            A [ k ] := i ;
        End ;
        Flag := True ;
    End ; { end of for }
Writeln ( ' The prime Numbers before N ' ) ;
For j := 1 to K do
    Writeln ( a [ i ] : 5 ) ;
End . { End of program }

```

مثال : برنامه‌ای بنویسید که یک ماتریس 3×3 را از ورودی دریافت کرده و مجموع هر سطر

را انتهای همان سطر به همراه خود ماتریس در خروجی چاپ نماید.

Var

```
a: array [ 1.. 3 , 1.. 3 ] of Real ; sum: Real ; i , j: byte ;
```

Begin

```
Writeln ( ' Enter Array ' )
```

```
;                                     for i:= 1
```

```
do
```

```
    for j:= 1 to 3 do
```

```
        Read ( a [ i , j ] ) ;
```

```
Writeln ; Writeln ( ' The result matrix ' ) ;
```

```
For i:= 1 to 3 do Begin
```

```
    For j:= 1 to 3 do Begin
```

```
        Write ( a [ i , j ]: 8: 2 ) ;
```

```
        Sum:= sum + a [ i , j ];{ calculate sum of any row }
```

```
    End ;
```

```
Writeln ( sum: 8: 2 ) ; Sum := 0 ;
```

```
End ;
```

```
End { End of program }
```


مثال : برنامه‌ای بنویسید که از ادغام دو آرایه مرتب، آرایه
سومی بنام L3 ایجاد کند بطوریکه آرایه سوم مرتب باشد.

Var

```
L1 , L2: array [ 1.. 30 ] of integer ;  
L3: array [ 1.. 60 ] of integer ;  
M , N , i , j , k: Byte ;
```

Begin

```
Write ( ' Enter Dimention of arrays: ' ) ;  
Readln ( N , M ) ;  
Writeln ( ' Enter first Array ' ) ;  
for i:= 1 to N do  
    Read ( L1 [ i ] ) ;  
Writeln ( ' Enter second Array ' ) ;  
for j:= 1 to M do  
    Read ( L2 [ i ] ) ;
```



```
i := 1 ; j := 1 ; k := 1 ;
while ( i <= N ) and ( j <= M ) Do      Begin
    If L1 [ i ] > L2 [ j ] Then      Begin
        L3 [ k ] := L2 [ j ] ;
        J := j + 1 ;
    End
    Else if L1 [ i ] < L2 [ j ] Then  Begin
        L3 [ k ] := L1 [ i ] ;
        i := i + 1 ;
    End
    Else      Begin
        L3 [ k ] := L1 [ i ] ;
        k := k + 1 ;
        L3 [ k ] := L2 [ j ] ;
        i := i + 1 ;
        j := j + 1 ;
    end ;
    k := k + 1 ;
End ; { end of while }
```



```
if i <= N Then
    for p:=i to n do Begin
        L3 [ k ]:= L1 [ p ] ;
        k:= k + 1 ;
    end
Else if j <= M Then
    for p:=j to M do Begin
        L3 [ k ]:= L2 [ p ] ;
        k:= k + 1 ;
    end ;
Writeln ;
Writeln ( ' The result of merge is ' ) ;
for i:=1 to k - 1 do Begin
    Write ( L3 [ i ] : 5 ) ;
    If ( i mod 10 ) = 0 Then
        Writeln ;
    End ; { end of merge }
End. { End of program }
```



مثال : برنامه ای بنویسید که یک جمله حداکثر 80 کاراکتری را از ورودی دریافت کرده و سپس کاراکترهای فضای خالی را با کاراکتر ستاره جایگزین کند.

```
Var      state: array [ 1.. 80 ] of char ;  
        i , N : integer ;  
  
Begin  
    Writeln ( ' Enter Number of sentence: ' ) ;  
    Readln ( N ) ;  
    Writeln ( ' Enter sentence ' ) ;  
    for i:= 1 to N do  
        Read ( state [ i ] ) ;  
    for i:= 1 to N do  
        if state [ i ] = ' ' Then  
            state [ i ] := ' * ' ;  
    Writeln ;  
    Writeln ( ' The output sentence ' ) ;  
    for i:= 1 to N do  
        Write ( state [ i ] ) ;  
  
End. { End of program }
```

❖ تمرینات

- کدامیک از دستورات زیر در مورد اعلان زیر صحیح است:

a: array [' A '.. ' Z '] of char

الف- **a [' A '] := ' Z '**

ب- **a [' a '] := ' A '**

ج- **a [' A '] := 1**

د- **a [' I '] := 12**

ح- **a [' i '] := 12**

-تعداد بایتهایی که هر کدام از اعلانهای زیر اشغال می کنند را محاسبه نمایید.

a: array [-20.. 10] of char •

a: array [-20.. -20 , 0.. -20] of ' A '.. ' Z ' •

a: array [Boolean] of char •

a: array [1.. 10 , ' A '.. ' Z '] of integer •

❖ تمرینات برنامه‌نویسی

- برنامه‌ای بنویسید که که یک آرایه حداکثر ۵۰ عنصری را از ورودی دریافت کرده و سپس عناصری از آرایه که اول هستند را با صفر جایگزین کرده آرایه حاصل را در خروجی چاپ کند.

برنامه‌ای بنویسید که عددی از ورودی دریافت کرده سپس آن را به عامل‌های اول تجزیه نماید و حاصل را بصورت زیر در خروجی چاپ نماید:

$$21 = (3^1) * (7^1)$$

برای مثال:

- برنامه‌ای بنویسید که یک عدد از ورودی دریافت کرده سپس در صورت وجود صفرهای آن را حذف نموده، نتیجه را در خروجی چاپ نماید.
- برنامه‌ای بنویسید که یک ماتریس $5 * 5$ را از ورودی دریافت کرده سپس مجموع هر سطر را انتهای همان سطر و مجموع هر ستون را در انتهای همان ستون چاپ نماید.

• برنامه‌ای بنویسید که یک ماتریس حداکثر $10 * 10$ را از ورودی دریافت کرده ماتریس هم بر حسب ستون و هم بر حسب سطر مرتب نموده به همراه ماتریس اول در یک سطر چاپ نماید.

• برنامه‌ای بنویسید که یک آرایه 200 عنصری از نوع صحیح که 150 عنصر مرتب در آن قرار می‌گیرد را از ورودی دریافت کرده سپس آرایه دومی با 50 عنصر را از ورودی بخواند.

فصل 9

توابع و روال‌های کتابخانه‌ای

هدفهای کلی

شناخت ساختار تابع و روال

شناخت توابع و روالهای استاندارد برای نوعهای صحیح

شناخت توابع و روالهای استاندارد برای نوعهای اعشاری و کارکتری

شناخت توابع و روالهای استاندارد ریاضی

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- توابع و روالهای استاندارد را در صورت نیاز در برنامه استفاده کند.
- ساختار توابع و خروجیهای آنها را تشخیص داده و در برنامه خود از آنها استفاده کند.
- ساختار روالها و خروجیهای آنها را تشخیص داده و نتایج حاصل از آنها را مورد استفاده قرار دهد.
- توابع ریاضی را برای محاسبه عبارت ریاضی در برنامه بکار ببرد.

❖ 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 می باشد:

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 می شود:

Function Abs (x: Integer): Integer ;

Function Abs (x: Real): Real ;

این تابع یک عبارت از نوع حقیقی 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 اعشاری

Function Sin (X: Real): Real ;

شکل تابع:

X یک عبارت یا عدد از نوع اعشاری بوده و حاصل مقدار سینوس X می باشد.

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 اعشاری.

Function $\cos (X: \text{Real}): \text{Real} ;$

شکل تابع:

X یک عبارت با عدد از نوع اعشاری بوده و حاصل مقدار کسینوس X می باشد.

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 اعشاری

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

X یک عبارت یا عدد از نوع اعشاری بوده حاصل مقدار آرک تانژانت X می باشد.

Var

X : Real ;

Begin

X := ArcTan (10) ;

Write (' ArcTan (10) = ' , x : 8 : 2) ;

End .

توجه: در صورتی که در توابع مثلثاتی زاویه از نوع درجه ارائه شود

می توان با فرمول زیر معادل رادیان آن را محاسبه کرد:

$$\text{Real} = \text{Dey} * 3.14159 / 180$$

✓ تابع Exp

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

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

X عبارت یا متغیری از نوع اعشاری بوده و حاصل تابع نیز یک عدد اعشاری می‌باشد
این تابع مقدار e به توان X را محاسبه می‌کند.

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 اعشاری یک عدد اعشاری را برمی گرداند.

شکل تابع: **Function `frac (X: Real): Real ;`**

X عددی از نوع اعشاری و حاصل تابع یک عدد اعشاری که قسمت اعشاری عدد **X** است می باشد و بعبارت دیگر این تابع قسمت اعشاری عدد ورودی را به عنوان خروجی باز می گرداند.

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

Int تابع ✓

هدف: قسمت صحیح یک عدد اعشاری را برمی گرداند.

Function Int (X: Real): Real ;

شکل تابع:

X یک عبارت یا متغیر از نوع اعشاری و خروجی تابع فیزیکی عدد اعشاری است این تابع مقدار صحیح یک عدد اعشاری در خروجی نشان می دهد.

مثال :

Var

y , x : Real ;

Begin

X := Int (2.87) ; { 2.0 }

Y := Int (- 8.76) ; { - 8.0 }

End .

✓ تابع Ln

هدف: محاسبه لگاریتم یک عدد اعشاری در مبنای e .

Function $\text{Ln} (X: \text{Real}): \text{Real} ;$

شکل تابع:

X یک عبارت یا متغیر از نوع اعشاری بوده و حاصل تابع فیزیک عدد اعشاری می باشد.

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: longint): Boolean ;**

X یک عبارت از نوع **longint** است و تابع، فرد بودن عبارت را بررسی می کند
اگر مقدار خروجی تابع **True** باشد **X** فرد است و اگر مقدار خروجی تابع **False**
باشد **X** فرد نیست.

مثال :

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

تابع Ord ✓

هدف: غالباً برای پیدا کردن کد اسکی یک متغیر کاراکتری بکار می‌رود.

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

عبارت **x** از نوع کاراکتری را بعنوان پارامتر دریافت و کد اسکی آن را برمی‌گرداند.

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 در خروجی چاپ می کند
کد اسکی 'A' عدد 65 می باشد.

✓ تابع pi

هدف: عدد پی را بر می گردانند.

Function pi: Real ;

شکل تابع:

این تابع برای بازگرداندن عددی پی (... 3.141592) مورد استفاده قرار می گیرد.

تابع Pred

هدف: مقدار قبل مقدار پارامتر را بر می گرداند.

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

پارامتر تابع می تواند از هر نوع باشد و با توجه به نوع پارامتر تابع نیاز از هم نوع می باشد و خروجی تابع مقدار قبل از x می باشد.

مثال :

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

✓ تابع Random

هدف: برای تولید عدد تصادفی

شکل تابع:

- 1) **Function Random: Real ;**
- 2) **Function Random (x: word): word ;**

اگر تابع **Random** به شکل یک یعنی بدون آرگومان مورد استفاده قرار گیرد یک عدد تساوی از نوع اعشاری بین صفر و یک تولید می کند و اگر به شکل دو بکار رود باعث تولید یک عدد تصادفی از نوع **word** که بزرگتر یا مساوی صفر و کوچکتر از **x** است خواهد شد.

مثال :

```
Var  
    i : integer ;  
Begin  
    For i := 1 to 10 do  
        Begin  
            Writeln ( Random : 8 : 7 ) ;  
            Writeln ( Random ( 20 ) : 8 ) ;  
        End ;  
    End .
```

خروجی :

10 عدد تصادفی بین صفر و یک،
و 10 عدد تصادفی بین 0 و 40 تولید نماید.

تابع Round ✓

هدف: برای گرد کردن اعداد اعشاری بکار می رود.

شکل تابع: **Function Round (x: Real): Longint ;**

X یک عبارت یا متغیر اعشاری بوده و خروجی تابع یک عدد از نوع **Longint** می باشد که نتیجه گرد کردن **X** می باشد. این تابع **X** را به نزدیکترین مقدار صحیح گرد می کند.

مثال :

$i := \text{Round} (57.4)$ $\{ i = 57 \}$

$i := \text{Round} (59.5)$ $\{ i = 60 \}$

$i := \text{Round} (12.7)$ $\{ i = 13 \}$

$i := \text{Round} (12.25)$ $\{ i = 12 \}$

$i := \text{Round} (17.75)$ $\{ i = 18 \}$

$i := \text{Round} (17.45)$ $\{ i = 18 \}$

$i := \text{Round} (-2.5)$ $\{ i = -3 \}$

✓ تابع sqr

هدف: برای محاسبه مجذور یک عدد صحیح یا اعشاری بکار می رود.

شکل تابع:

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

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

X عبارتی یا متغیری از نوع صحیح یا اعشاری بوده و خروجی تابع فیزیکی یک عدد صحیح یا اعشاری می باشد این تابع مجذور **X** را بعنوان خروجی بر می گرداند.

مثال :

Var

i : integer ;

Begin

For i := 1 to 10 do

Writeln (' I ^ 2 = ' , sqr (I)) ;

End .

✓ تابع sqrt

هدف: برای محاسبه جذر یک عدد بکار میرود.

Function sqrt (x: Real): Real ;

شکل تابع:

X یک عبارت از نوع اعشاری بوده و خروجی تابع فیزیکی عدد اعشاری می باشد این تابع جذر X را بعنوان خروجی بر می گرداند.

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 فعلی را برمی گرداند.

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

X یک عبارت از نوع صحیح، اولین و غیره بوده و خروجی تابع نیز از همان نوع **X** می باشد این تابع مقدار بعد از **X** را بعنوان خروجی بر می گرداند.

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 صحیح یک عدد اعشار را بر می گرداند.

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

X یک عبارت یا متغیر از نوع اعشاری بوده و خروجی تابع یک عدد از نوع **Longint** می باشد. این تابع قسمت صحیح عدد اعشاری **X** را بعنوان خروجی بر می گرداند.

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 کاراکتری یک کد اسکی را بر می گرداند.

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

X یک عبارت یا متغیر از نوع بایت بوده و خروجی تابع یک کاراکتر می باشد. این تابع کد اسکی را دریافت کرده و معادل کاراکتری آن را بر می گرداند.

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 می کند.

✓ تابع Uppcase

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

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

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

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 ]:= Uppcase ( sen [ i ] ) ;  
Writeln ( ' Out pot of program ' ) ;  
For i:= 1 to N do  
    Writeln ( sen [ i ] ) ;  
End. { End of program }
```

❖ روال‌های استاندارد

در بخش‌های قبل دیدید که در توابع پارامترها به تابع ارسال می‌شود 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 (Parameters)

↓ ↓ ↓
کلمه ذخیره شده اسم روال پارامترهای روال

Dec روال ✓

هدف: یک یا چند واحد از پارامتر ارسالی کم می کند.

شکل روال:

Procedure Dec (Var X: longint) ;

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

X یک متغیر از نوع **longint** و بصورت متغیری می باشد این روال یک واحد از پارامتر ارسالی کم می کند.

Var

N : integer

Begin

N := 1201 ;

Dec (N) ; { N = 1200 }

Writeln (N) ;

Dec (N , 200) ; { W = 1000 }

Writeln (N) ;

End .

Exit روال ✓

هدف: کنترل برنامه را به خارج از بلوک جاری منتقل می کند.

Procedure Exit ;

شکل روال:

این روال باعث می شود که کنترل برنامه از بلوک جاری خارج شود. اگر این روال در برنامه اصلی بکار رود باعث خروج از برنامه می شود. و اگر در یک روال یا تابع بکار رود باعث خروج از روال یا تابع شده و کنترل برنامه به برنامه اصلی منتقل می شود.

مثال :

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

خروجی :

بعد از اتمام عملیات تابع **Exit** باعث خروج از برنامه می شود.

Halt روال ✓

هدف: خاتمه دادن به اجرای برنامه

Procedure Halt ;

شکل روال:

این روال باعث خاتمه اجرای برنامه شده و کنترل برنامه به سیستم عامل بر می‌گردد.

Inc روال ✓

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** می باشد این روال به مقدار متغیر **x** یک یا چند واحد اضافه می کند.

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 تغییر نحوه تولید اعداد تصادفی می شود.

Procedure Randomize ;

شکل روال:

وقتی در برنامه از تابع **Random** استفاده می کنیم اعداد تصادفی تولید شده در اجراهای مختلف یکسان می باشد برای جلوگیری از این وضعیت قبل از استفاده از تابع **Random** روال **Randomize** را بکار می بریم. تا باعث تولید اعداد تصادفی متفاوت در اجراهای مختلف گردد.

مثال :

Var

i : word ;

Begin

Randomize;

For i := 1 to 10 do

Writeln (Random (50)) ;

End .

❖ حل چند مثال برنامه نویسی

مثال : برنامه ای بنویسید که یک ماتریس مربع $n * n$ ($n \leq 10$) از مقادیر صحیح را از ورودی دریافت کرده آنگاه عناصری که مربع کامل نیستند را صفر کرده و در نهایت ماتریس حاصل را در خروجی چاپ می کند.

Var

n , i , j : integer

a : array [1 .. 10 , 1 .. 10] of integer ;

Begin

Write (' enter Number : ') ;

For i := 1 to N do

For j := 1 to N do

Read (a [i , j]) ;





```
For i := 1 to N do
  For j := 1 to N do
    If a [ i , j ] sqr ( Trunc ( sqrt ( Abs ( a [ i , j ] ) ) ) ) < > 0 Then
      a [ i , j ] := 0 ;
  Writeln ( ' The oupput Mafrix ' ) ;
  For i := 1 to N do
    Begin
      For j := 1 to N do
        Write ( a [ i , j ] : 5 ) ;
      Writeln ;
    End ;
End . { End of program }
```

مثال : برنامه‌ای بنویسید که یک جمله از ورودی دریافت کرده سپس حروف بزرگ جمله را به حروف کوچک تبدیل نماید.

Var

```
sen : array [ 1 .. 80 ] of char ; Ch : char ; i : word ;
```

Begin

```
Write ( ' enter sentence ' ) ;
```

```
Repeat
```

```
    Read ( sen [ i ] ) ;
```

```
    Inc ( i ) ;
```

```
Until sen [ i ] = '.' ;
```

```
Writeln ;
```

```
n := Dec ( i ) ;
```

```
For i := 1 to n do
```

```
    If ( sen [ i ] >= ' A ' ) and ( sen [ i ] <= ' B ' ) Then
```

```
        sen [ i ] = chr ( ord ( sen [ i ] ) + ( ord ( ' a ' ) - ord ( ' A ' ) ) )
```

```
Writeln ( ' The result sentence ' ) ;
```

```
For i := 1 to n do
```

```
    Write ( sen [ i ] ) ;
```

```
End . { End of program }
```


تمرینات ❖

a)

```
Var
    i , j : integer
Begin
    i := 100 ; j := 20 ;
    inc ( i ) ;
    Dec ( j , 10 ) ;
    inc ( i , j ) ;
    Writeln ( i : 5 , j : 5 ) ;
End .
```

b)

```
Var
    y : integer ;
Begin
    y :=Round ( 18.31 ) MoD 5 ;
    Writeln ( y ) ;
End .
```

1. خروجی تمرینات زیر را تعیین کنید:

2. خروجی دستورات زیر را محاسبه نمایید:

a)

Ch: = ' A ' ;

Ch: = chr (ord (ch) + 3) ;

Write (ch) ;

b)

X: = 12 ;

Succ (x) ;

Ch: = ' B ' ;

Pred (ch) ;

Write (x , ch) ;

❖ تمرینات برنامه‌نویسی

- ۱- برنامه‌ای بنویسید که تعداد ۱۰۰۰ شماره حساب بانکی ۷ رقمی بطور تصادفی بین ۵۱۱۹۴۳۲ و ۹۹۸۱۷۱۱ را تولید کرده و چاپ کند.
- ۲- برنامه‌ای بنویسید تا تعداد ۳۸ عدد تصادفی صحیح بین ۱۰ تا ۹۹ را تولید کرده، سپس آن را در خروجی به شکل مربع وسط صفحه نمایش چاپ کند.
- ۳- برنامه‌ای بنویسید که یک عدد ۲۰ رقمی از ورودی دریافت کرده آن را در یک عدد تک رقمی ضرب نموده و حاصل را در خروجی چاپ نماید.

فصل 10

متغیرهای کاراکتری و رشته‌ها (String)

هدفهای کلی

بررسی ساختار آرایه‌هایی از نوع کاراکتر
معرفی نوع داده جدید به نام رشته
مقایسه آرایه‌ای از کاراکتر و رشته
شناخت توابع و روالهای استاندارد برای رشته‌ها

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- آرایه‌ای از کاراکترها را در برنامه بکار ببرد.
- آرایه‌ای از کاراکترها را با رشته مقایسه کند.
- اسمی افراد و غیره را با استفاده از آرایه‌ای از رشته مرتب نماید.
- توابع و روالهای استاندارد مربوط به رشته‌ها را در برنامه خود بکار ببرد.

❖ متغیرهایی از نوع کاراکتر

متغیرهای کاراکتری ظرفیت پذیرش یک کاراکتر (شامل یک رقم، یک حرف از حروف و یا یک کاراکتر دیگر) را دارا می‌باشند.
مقدار دهی این متغیرها به صورت زیر می‌باشد :

```
ch: = ' A ' ;
```

اگر بخواهیم متغیرهای کاراکتری را از ورودی بخوانیم باید دقت بیشتری به خرج دهیم برای اینکه فضاهای خالی کاراکتر محسوب می‌شوند.

مثال : خروجی قطعه برنامه زیر را تعیین کنید.

Ch: = ' A ' ;

C: = ' a ' ;

If ch = c Then

Writeln (' equal ')

Else

Writeln (' Not equal ') ;

خروجی حاصل از قطعه برنامه بالا بصورت زیر است:

Not equal

در کل می توان کاراکترها را از لحاظ اردینال (مرتبه) بصورت زیر دسته بندی کرد:

۱- رقم ها بصورت زیر مقایسه می شوند:

'0' < '1' < '2' < '3' < ... < '9'

رقم ها از لحاظ کد اسکی پشت سر هم قرار گرفته اند.

۲- حروف بصورت زیر مقایسه می شوند:

'B' < ... < 'Z' < ... < 'a' < 'b' < ... < 'z'

❖ متغیرهای رشته‌ای (String)

تعریف: مجموعه‌ای از کاراکتر را یک رشته نامیده و متغیر از نوع آن را یک متغیر رشته‌ای می‌نامند.

این متغیر نیز مانند سایر متغیرها در قسمت تعاریف متغیرها (Var) معرفی می‌شود. در تعریف یک متغیر رشته‌ای معمولاً طول آن را مشخص می‌کنند. اگر طول تعیین نشود بطور قراردادی کامپایلر حداکثر طول را برای آن در نظر می‌گیرد. حداکثر طول رشته ۲۵۵ کاراکتر می‌باشد.

نحوه تعریف متغیر رشته ای بصورت زیر می باشد:

Name : string [length]



اسم رشته کلمه ذخیره شده طول رشته

متغیرهای رشته‌ای را بصورت زیر می‌توان مقداردهی کرد:

Var

```
S : string [ 10 ] ;
```

-
-
-

Begin

```
•  
•  
S := ' pascal ' ;  
Write ( s ) ;
```

-
-

End .

برای خواندن متغیرهای رشته‌ای از دستور Read یا Readln بصورت زیر می‌توان استفاده کرد:

Read (s) ;

هنگام خواندن متغیرهای رشته‌ای اگر طول رشته ورودی از طول تعریف شده بیشتر باشد فقط به اندازه طول تعریف شده خوانده می‌شود و اگر طول رشته ورودی کمتر از طول تعریف شده باشد، رشته ورودی در منتهی‌الیه سمت چپ متغیر رشته‌ای قرار گرفته و بقیه متغیر رشته‌ای بدون محتوا باقی می‌ماند،

برای چاپ یک متغیر رشته‌ای نیز از دستور Write یا
writeln می‌توان استفاده کرد:

Write (s) ;

مثال : برنامه‌ای بنویسید که شماره دانشجویی، اسم و فامیل دانشجویی را
از ورودی دریافت کرده سپس در خروجی نمایش دهد.

Var

```
Name , family: string [ 40 ] ;  
Id: longint
```

Begin

```
writeln ( ' Enter student number ' ) ;  
Readln ( Id ) ;  
writeln ( ' Enter Name ' ) ;  
Readln ( Name ) ;  
writeln ( ' Enter family ' ) ;  
Readln ( family ) ;  
writeln ( ' Id Name Family ' ) ;  
writeln ( Id: 7 , Name , Family ) ;
```

End.

برای خواندن چنین آرایه‌ای بصورت زیر عمل می‌کنیم:

```
For i := 1 to 50 do  
  Readln ( Name [ i ] ) ;
```

برای نمایش آرایه در خروجی بصورت:

```
For i := 1 to 50 do  
  Writeln ( Name [ i ] )
```


مثال : برنامه‌ای بنویسید که اسم 10 نفر از ورودی دریافت کرده و آنها را بر حسب حروف الفبا مرتب نماید.

Var

```
Name : Array [ 1 .. 10 ] of string [ 10 ] ;  
string [ 10 ] ; temp : i , j : Byte;
```

Begin

```
writeln ( ' Enter Ten Names ' ) ;  
for i := 1 to 10 do  
    Readln ( Name [ i ] ) ;  
for j := 1 to 9 do  
    for j := i + 1 to 10 do  
        if Name [ i ] > Name [ j ] Then Begin  
            Temp := name [ i ] ;  
            Name [ i ] := Name [ j ] ;  
            Name [ j ] := temp ;  
        End ; { sort Algrithm }  
writeln ( ' The sorted list ' ) ;  
for i := 1 to 10 do  
    writeln ( Name [ i ] ) ;  
end . { End of program }
```

مثال : برنامه‌ای بنویسید که اسم، فامیلی و شماره دانشجویی حداکثر ۵۰ دانشجوی را از ورودی دریافت نماید. سپس با خواندن اسمی از ورودی، سایر اطلاعات اسم خوانده شده را در صورت وجود در خروجی نمایش دهد.

Var


```
Name , Family : array [ 1 .. 50 ] of string [ 30 ] ;  
id : array [ 1 .. 50 ] of longint ;  
Nam : string [ 30 ] ;
```

Begin

```
write ( ' Enter Number : ' ) ;  
Readln ( N ) ;  
for i := 1 to N Do Begin  
    write ( ' Enter Name : ' ) ;  
    Readln ( Name ) ;  
    write ( ' Enter family : ' ) ;  
    Readln ( family ) ;  
    write ( ' Enter student Number : ' ) ;  
    Readln ( Id ) ;  
end ;  
writeln ;  
write ( ' Enter Name : ' ) ;  
Readln ( Nam ) ;  
i := 0 ;  
flag := Ture ;
```



```
while ( i <= n ) and flag Do
Begin
    inc ( i ) ;
    if Name [ i ] = Nam then
        flag := false ;
end ;
If flag then
    writeln ( ' Not found ' ) ;
Else
Begin
    writeln ( '   Id           Name           family   ' )
    write ( Id , Name : 30 , family : 30 ) ;
end ;
End . { End of program }
```



❖ توابع و روالهای کتابخانه‌ای برای متغیرهای رشته‌ای

✓ تابع Concat

هدف: الحاق دو یا چند رشته به یکدیگر

شکل تابع: `Function concat (S1 , S2 , ... , Sn): string ;`

S_1, S_2, \dots, S_n متغیرهایی از نوع رشته هستند و خروجی تابع نیز یک متغیر رشته‌ای است این تابع دو یا چند تابع را به هم پیوند داده، و رشته حاصل را برمی‌گرداند.

مثال :

```
Var  
    Str3 , str1 , str2 : string  
Begin  
    Str1 := ' Pascal ' ;  
    Str2 := ' Book ' ;  
    Str3 := Concat ( Str1 , Str2 ) ;  
    write ( Str3 ) ;  
End.
```

با اجرای برنامه فوق، عبارت **Pascal Book** نشان داده خواهد شد.

✓ تابع Copy

هدف: استخراج یک زیر رشته (`substring`) از یک رشته

شکل تابع: `function copy (S:string ; Index:Integer; count: Integer): string;`

یک عبارت یا متغیر رشته ای که می خواهیم از آن زیر رشته ای که نقطه `S` می باشد جدا کنیم. لذا زیر رشته `Count` و طول آن `Index` شروع آن است، می باشد. `Count` حاصل، یک رشته که طول آن به اندازه

مثال :

Var

Str , str1: string ;

Begin

Str: = ' Pascal Book ' ;

Str1: = Copy (Str , 7 , 4) ; { Str1 = Book }

write (Str1) ;

End.

Delete روال ✓

هدف: حذف یک زیر رشته از یک رشته

شکل روال: **Procedure delete (Var str: string; Index:integer;length: integer) ;**

Str یک متغیر رشته ای، Index یک عبارت یا متغیر صحیح و Length نیز یک عبارت یا متغیر صحیح می باشد. روال Delete یک زیر رشته را از یک رشته حذف می کند این روال از محل Index بطول Length از رشته Str حذف می کند و رشته حاصل بعنوان خروجی روال برگردانده می شود.

مثال :

```
Var  
    St : string ;  
Begin  
    St := 'Pascal.Book' ;  
    Delete ( St , 7 , 5 ) ;  
    writeln ( St ) ;  
End.
```

خروجی حاصل از برنامه فوق Pascal می باشد.

روال Insert ✓

هدف: درج (وارد کردن) یک رشته در یک رشته دیگر

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

Str1 عبارت یا متغیر رشته ای، **Str2** متغیر رشته ای و **Index** عبارتی عدد از نوع صحیح می باشد. روال **Insert**، رشته **Str1** را در رشته **Str2** از خانه **Index** درج می کند و رشته حاصل خروجی روال خواهد بود.

مثال :

Var

Str1 , str2: string ;

Begin

Str1: = ' Pascal 7' ;

Str2: = ' Turbo' ;

Insert (Str1 , Str2 , 6) ;

End.

خروجی حاصل از برنامه بالا Turbo pascal 7 خواهد بود.

✓ تابع Length

هدف: محاسبه طول رشته

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

Str یک عبارت رشته ای بوده و خروجی تابع یک عدد صحیح می باشد.
این تابع طول رشته ورودی را محاسبه و بعنوان خروجی بر می گرداند.

مثال :

Var

St: string ;

n: integer ;

Begin

St: = 'Turbo Pascal 7' ;

n: = length (S4) ; { n = 14 }

Write (' The length of string is: ' , n) ;

End.

✓ تابع Pos

هدف: برای جستجوی یک رشته داخل رشته دیگر

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

`Str1` یک عبارت یا متغیر رشته ای و `Str2` نیز یک عبارت یا متغیر رشته ای می باشد و خروجی تابع یک عدد صحیح می باشد. این تابع محل اولین وقوع رشته `Str1` در رشته `Str2` را بعنوان خروجی بر می گرداند.

مثال :

Var

Str1 , Str2: string ;

i: integer ;

Begin

Str1: = ' Book ' ;

Str2: = ' Pascal Book ' ;

i: = Pos (Str1 , Str2) ; { i = 8 }

Write (' i = ' , i) ;

End.

نکته: توجه کنید که اگر عمل جستجو با موفقیت انجام نشود (یعنی رشته اول در رشته دوم وجود نداشته باشد) تابع مقدار صفر بر می گرداند.

✓ روال Str

هدف: برای تبدیل عدد به یک رشته عددی بکار می رود.

شکل روال: 1: Procedure Str(I: integer: format , Str: string) ;

2: Procedure Str (F: Real: format , Str: string) ;

در شکل 1، I یک عبارت یا متغیر عددی بوده و Str یک متغیر رشته‌ای می‌باشد این روال یک عدد صحیح با فرمت مشخص را به یک رشته عددی تبدیل می‌کند.

در شکل 2، F یک عبارت یا متغیر رشته‌ای و Str یک متغیر رشته‌ای می‌باشد.

مثال :

Var

Number , Str: String ;

Begin

Str (543: 5 , Str) ; { Str = ' 543' }

Str (12.25: 7: 2 ,Str) ; { Str = ' 12.25' }

Str (127: 3: Str) ; { Str = '127' }

Str (3.1239: 7: 3 , Str) ; { Str = ' 3.124' }

End.

✓ روال Val

هدف: تبدیل یک رشته عددی به یک عدد

1: Procedure Val (S: String ; Var N: integer ; شکل روال:
Var Error: integer) ;

2: Procedure Val (S: String ; Var N: Real ; Var Error: integer) ;

S یک عبارت یا متغیر رشته ای، N یک متغیر از نوع صحیح یا اعشاری و Error نیز یک متغیر از نوع صحیح می باشد. این روال یک رشته عددی را به یک عدد صحیح یا اعشاری تبدیل می کند و آن را توسط متغیر N برمی گرداند. اگر عمل تبدیل بطور صحیح انجام شود، مقدار متغیر Error برابر صفر در غیر اینصورت محل وجود اشکال را مشخص می کند.

مثال :

Var

St: string ;
E , N: integer ;
F: Real ;

Begin

St = ' 475 ' ;
Val (St , N , E) ; { N = 475 , E = 0 }
St = ' 475' ;
Val (St , F , E) ; { F = , E = 3 }
St = '3.1716' ;
Val (St , F , E) ; { F = 3.1716 , E = 0 }

End.

❖ ارائه چند مثال در مورد رشته‌ها و کاراکترها

مثال : برنامه‌ای بنویسید که یک جمله از ورودی دریافت کرده سپس در صورتی که کلمه **IS** وجود داشته باشد آنها را به **are** تبدیل نماید و در نهایت رشته حاصل را در خروجی نمایش دهد.

توجه کنید که ممکن است بیش از یک بار کلمه **IS** تکرار شده باشد در اینصورت همه کلمات **IS** را با **are** تبدیل کند.

Var

St : string ;

Begin

Writeln (' Enter sentence ') ;

Readln (St) ;

Repeat

i := Pos ('is' , St) ;

if i > 0 Then

Begin

Delete (St , i , 2) ;

Insert ('are' , St , i) ;

end ;

Until i = 0 ;

Writeln (' The result sentence ') ;

Write (St) ;

End . { End of program }

مثال : برنامه‌ای بنویسید که یک رشته از ورودی دریافت کرده، سپس معکوس رشته را بدست آورده به‌مراه خود رشته در دو سطر جداگانه چاپ نماید.

Var

```
St1 , St2 : string ;  
i , j : integer ;
```

Begin

```
Writeln ( ' Enter sentence ' ) ;  
Readln ( St1 ) ; j := 1 ;  
For i := length ( St1 ) downto 1 do Begin  
    St2 [ j ] := St1 [ i ] ;  
    Inc ( j ) ;  
End ;  
Writeln ( ' The result ' ) ;  
Writeln ( St1 ) ; Writeln ( St2 ) ;  
End . { End of program }
```

❖ تمرینات

- با فرض اینکه $S1$, $S2$, $S3$ متغیرهای رشته ای هستند خروجی عبارتهای زیر را تعیین کنید.

a) $S3 = \text{copy} (S1 , 1 , 6) ;$

b) $S3 = \text{concat} (S3 , S2 , S1) ;$

c) $S3 = \text{copy} (S2 , 1 , \text{pos}(S1 , S2) - 1) ;$

d) $\text{Delete} (S2 , \text{pos}(S1 , S2) , \text{length} (S1)) ;$

• خروجی قطعه برنامه زیر را تعیین کنید:

Var

S1: string ;

i: integer ;

Begin

S1 := 'ABCDEF' ;

For i := 1 to length (S1) ;

 Delete (S1 , i , 1) ;

Writeln (S1) ;

.End

• خروجی قطعه برنامه زیر را تعیین کنید:

Var

St: string ;

i: integer ;

Begin

Readln (St) ;

For i := 1 to length (St) Do

 If (St [i] >= ' 0 ') and (St [i] <= ' 9 ') then

 Delete (St , i , 1) ;

Writeln (St) ;

.End

• با فرض اینکه $S2 = \text{'Book'}$, $S1 = \text{'pascal Book'}$ هستند خروجی عبارت زیر را تعیین کنید.

- a) $\text{Insert} (S1 , S2 , \text{pos} (S1 , S2)) ;$
- b) $S3 := \text{copy} (S2 , \text{pos} (S1 , S2) , \text{length} (S1)) ;$
- c) $S3 := \text{copy} (S2 , \text{pos} (S1 , S2) , \text{length} (S2)) ;$
- d) $\text{Delete} (S1 , \text{pos} (S2 , S1) , \text{length} (S2));$
- e) $S1 [0] := \# 6 ;$

❖ تمرینات برنامه‌نویسی

• برنامه‌ای بنویسید که یک پاراگراف را از ورودی دریافت کرده (حداکثر ۱۰ خط) سپس:

- الف) تعداد کلمات هر سطر را شمرده و انتهای سطر نمایش دهد.
- ب) تعداد حروف صدا دار را شمرده چاپ نماید.
- ج) تعداد خطوط برنامه را شمرده در خروجی چاپ کند.
- د) تعداد جملات هر خط را محاسبه و در سطرهای جداگانه نمایش دهد.

• برنامه‌ای بنویسید که یک عدد در مبنای مشخص را از ورودی دریافت کرده سپس آن را به مبنای m که از ورودی خوانده می‌شود ببرد.

• برنامه‌ای بنویسید که یک پاراگراف را از ورودی دریافت کرده کلمات تکراری هر سطر را حذف نموده و پاراگراف حاصل را به‌مراه پاراگراف اولیه در خروجی با پیغام مناسب نمایش دهد.

• برنامه‌ای بنویسید که دو عدد ۲۰ رقمی (بصورت رشته باید خوانده شود) را از ورودی دریافت کرده سپس مجموع و حاصل ضرب این دو عدد را محاسبه و در خروجی با پیغام مناسب چاپ نماید.

• برنامه‌ای بنویسید که یک رشته از ورودی دریافت کرده سپس کلمات داخل آن را بصورت عمودی کاراکتر به کاراکتر نمایش دهد:

• برنامه‌ای بنویسید که یک پاراگراف با حداکثر ۵ خط را از ورودی دریافت نماید. سپس مجموع ارقام یا اعداد (در صورت وجود) هر خط را محاسبه نموده در انتهای همان سطر نمایش دهد.

برنامه‌های فرعی

هدفهای کلی

- شناخت اجزاء تشکیل دهنده توابع و روالها
- بررسی انواع پارامترها و متغیرها در برنامه‌های فرعی
- شناخت تفاوت‌های روالها و توابع
- معرفی مزایای استفاده از زیر برنامه‌ها

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

■ برنامه خود را به چندین زیربرنامه تقسیم‌بندی نماید.

■ زیربرنامه‌ها را با توجه به قوانین موجود، در ساختار برنامه جا دهد.

■ با توجه به مزایای استفاده از آنها را در برنامه خود بکار ببرد.

❖ روالها

روالها نوعی از برنامه‌های فرعی هستند، که به طور مستقل و جداگانه وظیفه یا وظایف خاصی از برنامه اصلی را انجام می‌دهند. روالها در صورت نیاز اطلاعات خود را از طریق پارامترها دریافت و همچنین در صورت نیاز نتایج را از طریق پارامترها به برنامه اصلی باز می‌گردانند. پارامترها در حقیقت خطوط ارتباطی بین برنامه اصلی و برنامه‌های فرعی هستند. پارامترها باعث می‌شوند، که توابع و روالها روانتر عمل کنند، زیرا آنها به برنامه‌های فرعی این قابلیت را می‌دهند که با هر فراخوانی، داده‌های مختلفی را مورد پردازش قرار دهند.

شکل کلی روالها بصورت زیر می باشد :

Procedure Name (parameters list) ;

↓ ↓ ↓
کلمه ذخیره شده اسم روال لیست پارامترها
Var

{ List of local variable }

Begin

.

.

{ Procedure Body }

.

.

End ; { End of procedure }

در حالت کلی روال ها در برنامه اصلی بصورت زیر ظاهر می شوند:

Program اسم برنامه اصلی ;

تعاریف برنامه اصلی ;

Procedure ;

Begin { main program }

.

.

.

فراخوانی روال ها

.

.

End. { End of program }

پارامترها با توجه به محل وقوع آنها به دو دسته تقسیم می شوند:

1. پارامترهای صوری (Formal parameters)

2. پارامترهای واقعی (Actual parameters)

از پارامترهای صوری هنگام اعلان روال و از پارامترهای واقعی هنگام فراخوانی روال ها استفاده می شود .

**پارامترهای صوری در حالت کلی 2 نوعند، که با توجه به نوع روال و
تصمیم برنامه نویس مورد استفاده قرار می گیرند :**

✓ پارامترهای مقداری (Value parameters)

✓ پارامترهای متغیری (Variable parameters)

مثال : به برنامه زیر توجه کنید:

Program main ;

Var

x1 , x2 , y1 , y2 : integer ;

Procedure test (A1,A2,B1,B2:integer);{formal parameters

.....
Begin { main }
.....
test (x1 , x2 , y1 , y2) ; { Actual parameters }
.....
End. { End of program }

The diagram illustrates the relationship between formal and actual parameters. Four diagonal arrows point from the actual parameters 'x1', 'x2', 'y1', and 'y2' in the procedure call to the corresponding formal parameters 'A1', 'A2', 'B1', and 'B2' in the procedure definition. A horizontal curly brace is positioned below the actual parameters, indicating they are grouped together as the 'Actual parameters'.

نکته: پارامترهایی که در روال بکار برده می شود، هیچ ارتباطی به پارامترهای ارسالی از برنامه اصلی ندارند و فقط مقادیر این متغیرها از برنامه اصلی ارسال می شود. لذا اسامی پارامترهای صوری ممکن است هم اسم با پارامترهای واقعی برنامه اصلی انتخاب شوند، این به معنای این نیست که این پارامترها (پارامترهای صوری) همان پارامترهای واقعی هستند.

در مثال بالا این تناظر برقرار است:

پارامترهای صوری

X1

X2

Y1

Y2

متناظر است با

پارامترهای واقعی

A1

A2

B1

B2





متغیرهای **Var1** , **Var2** , ... پارامترهای مقداری هستند. توجه کنید، که

این پارامترها صوری هستند و نوع و ترتیب آنها باید با نوع و ترتیب

پارامترهای واقعی در تناظر یک به یک باشند.

مثال : به برنامه زیر توجه کنید:

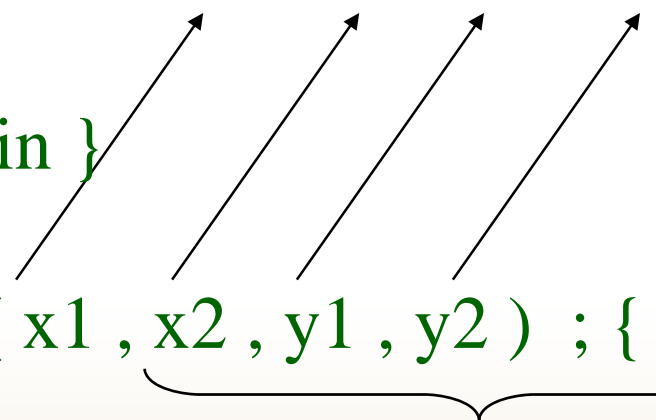
Program main ;

Var

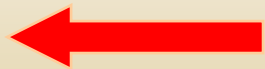
x1 , x2 , y1 , y2 : integer ;

Procedure test (A1 , A2 , B1 , B2: integer); { formal parameter

.....
Begin { main }
....
test (x1 , x2 , y1 , y2) ; { Actual parameters }
.....
End. { End of program }



The diagram consists of four parallel arrows pointing from the actual parameters (x1, x2, y1, y2) in the test procedure call to the corresponding formal parameters (A1, A2, B1, B2) in the test procedure definition. A horizontal curly brace is positioned below the actual parameters, and a vertical line connects its center to the center of the first arrow pointing to A1.





همانطور که ملاحظه می کنید، روالی بنام **test** با ۴ پارامتر مقداری در برنامه استفاده شده است. متغیرهای **x1**, **x2**, **y1**, **y2** از برنامه اصلی به روال **test** ارسال شده و بترتیب مقادیر این متغیرها در متغیرهای با همان نوع در **A1**, **A2**, **B1**, **B2** قرار می گیرند.

✓ پارامترهای متغیری (Variable parameters)

یک پارامتر متغیری، تغییرات یک پارامتر واقعی را بعنوان خروجی به برنامه اصلی برمی گرداند. تفاوت این گونه پارامترها با پارامترهای مقداری در ارسال تغییرات ایجاد شده در پارامترها می باشد. وقتی نیاز است، که نتایج روال به برنامه اصلی برگردد، از پارامترهای متغیری استفاده می کنند.

اعلان پارامترهای متغیری به صورت زیر می باشد:

```
procedure Name ( var id1 : type ; var id2 : type , ... );
```

↓ کلمه ذخیره شده ↓ اسم روال لیست پارامترهای متغیری

متغیرهای **id1** , **id2** و ... پارامترهای متغیری می باشند که نتایج روال، توسط آنها به برنامه اصلی بازگردانده می شود.

مثال : روالی بنام ComputeSumAve بنویسید که مجموع و میانگین دو عدد را محاسبه و نتیجه را در برنامه اصلی چاپ نماید.

Program Example;

Var

Num1, Num2: integer; Total, average: Real;

Procedure ComputeSumAve(Num1 , Num2: integer ;

Var sum , ave : Real);

Begin

Sum:=Num1+Num2; Ave:=sum/2;

End; {End of procedure}

Begin {Main}

WriteLn(' Enter Two Numbers');ReadLn(Num1,Num2);

ComputeSumAve (Num1, Num2, total, average);

**WriteLn('The Sum is= ', sum:8:2 ,
' The average is= ', average:8:2);**

End. {End of program }





پارامترهای واقعی و صوری عبارتند از:

| پارامترهای صوری | متناظر است با | پارامتر واقعی |
|-----------------|---------------|---------------|
| Num1 | | Num1 |
| Num2 | | Num2 |
| Total | | sum |
| Average | | ave |

Num1 , Num2 پارامترهای مقداری و ave , sum پارامترهای متغیری می باشند، که توسط آنها نتایج به برنامه اصلی برگردانده می شود.

✓ متغیرهای محلی و سراسری (Local and Global Variable)

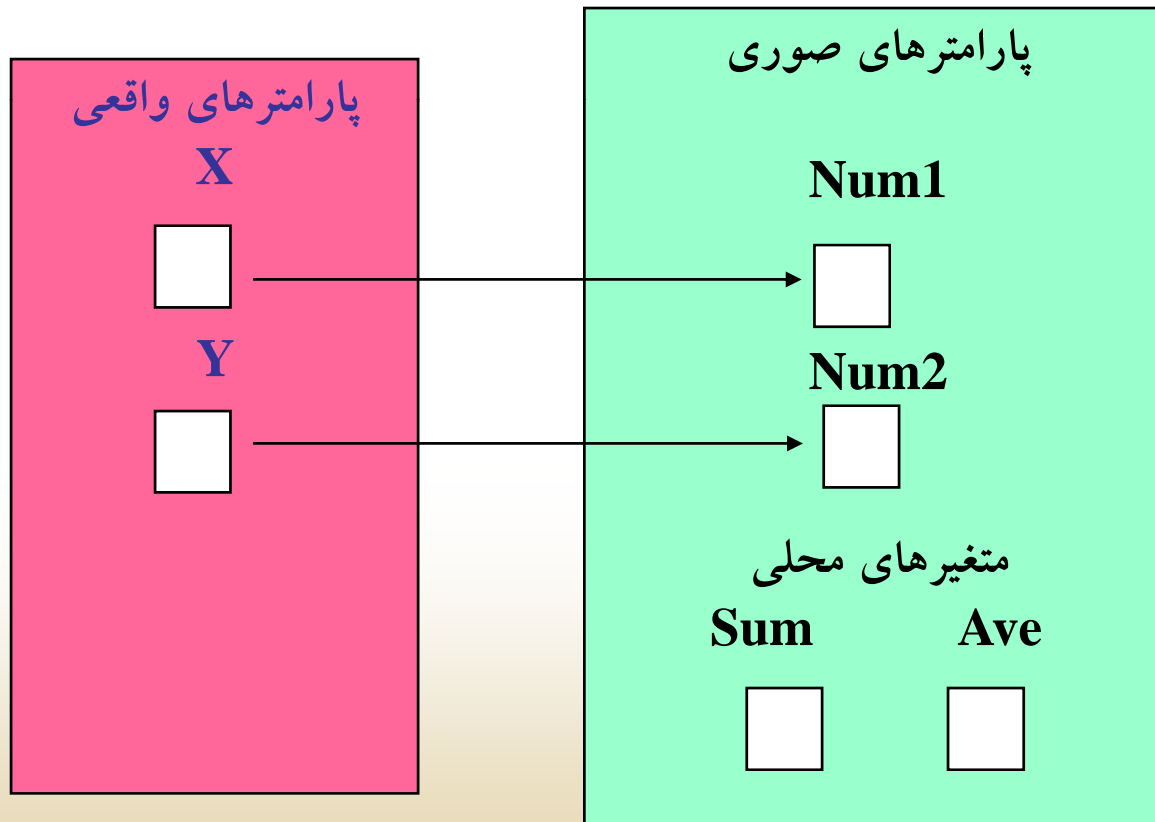
در برنامه‌های فرعی دو نوع متغیر علاوه از پارامترها مورد استفاده قرار می‌گیرند. این متغیرها متغیرهای محلی یا متغیرهای سراسر هستند. متغیرهای محلی، متغیرهایی هستند که در بلاک مربوط به خود قابل استفاده هستند. ولی متغیرهای سراسری در تمام برنامه‌های فرعی قابل دسترس می‌باشند.

متغیرهای محلی در داخل برنامه‌های فرعی در قسمت تعاریف معرفی می‌شوند و در خارج از روال قابل دسترسی نیستند.

مثال : روالی برای محاسبه مجموع و میانگین دو عدد بنویسید.

```
Procedure   CalcSumAve ( Num1 , Num2: Read );  
Var  
           Sum , Ave: Real ;  
Begin  
           Sum:= Num1 + Num2 ;  
           Ave:= Sum / 2 ;  
           WriteLn(' The Sum is =' , sum:8:2 ) ;  
           WriteLn(' The average is =' ,ave:8:2);  
End;{ End of procedure }
```


در مثال بالا `sum` , `Ave` متغیرهای محلی می‌باشند. برای فراخوانی روال `CalcSumAve(x.y);` که در آن دو متغیر از نوع `Real` می‌باشند. به شکل زیر توجه کنید:



مثال : خروجی برنامه زیر را تعیین کنید.

```
Program      Example6;  
Var          A , b , c: integer ;    { Global Variables }  
Procedure   test(var b: integer ; a: integer );  
Var         D: integer ;    {local variable}  
Begin  
            D:= 12 ; a:= b + d ; b:= a + c ; c:= c+ 2 ;  
End;  
Begin {Main}  
    A:=1; b:=2; c:=3;  
    Test(a,b);  
    WriteLn( ' a = ' , a , ' b = ' , b , ' c = ' , c ) ;  
End. {End of program }
```

خروجی برنامه بالا بصورت زیر است:

a =16 b=2 c=5

✓ روال‌ها معمولاً به سه شکل ظاهر می‌شوند :

بکارگیری روال‌های بدون پارامتر

بکارگیری روال همراه پارامترهای با خاصیت ورودی

بکارگیری روال همراه پارامترهای با خاصیت ورودی و خروجی

✓ بکارگیری روال‌های بدون پارامتر

گاهی لازم است تا برنامه فرعی کاملاً مستقل (بی‌نیاز از مقادیر برنامه اصلی) در بخش‌های مختلف یک برنامه اجرا شود. در این صورت نیاز به استفاده از پارامتر بی‌مفهوم می‌باشد و از روال‌های بدون پارامتر استفاده می‌کنند.

غالباً زمانی که بخواهیم پیغام‌های خاصی را در قسمت‌های مختلف برنامه نمایش دهیم، این پیغام‌ها را در یک روال قرار داده و در صورت نیاز، روال مربوطه را فراخوانی می‌کنیم.

مثال : روال Head بدون دریافت پارامتری فراخوانی می شود.

```
Program      Example ;
Procedure    Head  ;
  Begin
    WriteL('Name   Family   Age   No');
    WriteLe('.....');
  End;
Begin{ Main }
  WriteLn( ' open university ' ) ;
  Head;
End.
```

✓ بکارگیری روال همراه پارامترهای با خاصیت ورودی

همانطور که قبلاً اشاره کردیم، هدف از بکارگیری پارامترها انتقال مقادیر از برنامه اصلی به روال‌ها می‌باشد. اگر این انتقال یک طرفه باشد یعنی فقط از برنامه اصلی به روال باشد، این نوع پارامترها فقط خاصیت ورودی خواهند داشت. (قبلاً در این مورد توضیح داده شده است) در این نوع روال‌ها از پارامترهای مقداری استفاده می‌کنند.

مثال : روالی بنویسید که توسط آن مربعات اعداد ۱ تا N را در خروجی چاپ نماید.

```
Program Example ;
```

```
Var
```

```
    N , I: integer ;
```

```
Procedure sq ( M: integer ) ;
```

```
    Begin
```

```
        WriteLn ( ' sqart is = ' , M * M ) ;
```

```
    End;
```

```
Begin {Main}
```

```
    Write(' Enter Number = ' ) ;
```

```
    RealLn ( N ) ;
```

```
    For I:= 1 to N do
```

```
        Sq (i) ;
```

```
End. {End of program}
```

✓ بکارگیری روال همراه پارامترهای با خاصیت ورودی و خروجی

در این نوع روالها پارامترها دو خاصیت مهم دارند یکی انتقال داده‌ها از برنامه اصلی به روال و دیگری انتقال یا ارجاع نتایج از روال به برنامه اصلی می‌باشد. در این نوع روالها از پارامترهای متغیری استفاده می‌کنند.

مثال : روالی بنام **change** بنویسید که توسط آن دو عدد بعنوان پارامتر دریافت کرده، مقادیر این دو متغیر را جابجا نموده و نتیجه را در برنامه اصلی چاپ نماید.

```
Program Example ;
```

```
Var
```

```
    A , b: integer ;
```

```
Procedure change (var x , y: integer) ;
```

```
var
```

```
    temp:integer;
```

```
Begin
```

```
    Temp:=x;
```

```
    X:=y;
```

```
    Y:=temp;
```

```
End;
```





Begin { Main }

 ReadLn(a , b);

 Change(a , b);

 WriteLn (' a= ' , a , ' b= ' , b);

End. {End of program }

❖ ارتباط روال‌ها با یکدیگر

در پاسکال زیر برنامه‌ها نه تنها از طریق برنامه اصلی بلکه از داخل یکدیگر نیز فراخوانی می‌شوند.

فراخوانی روال‌ها از داخل یکدیگر تابع قوانین کلی زیر است:

✓ قانون اول

قانون دوم

قانون سوم

✓ قانون اول

از هر برنامه (اصلی یا فرعی) به برنامه فرعی در صورتی می توان، دسترسی داشت، که در بخش تعاریف آن برنامه (اصلی یا فرعی) قرار داشته باشد.

بطور مثال در شکل زیر برنامه اصلی M می تواند، به کلیه برنامه های فرعی p_1, p_2 و ... که در بخش تعاریف برنامه اصلی قرار دارند، مراجعه کند

Main

Proc1

Proc2

.

.

.

ProcN

برنامه اصلی

برنامه های فرعی

مثال : به برنامه زیر توجه کنید:

```
program Main;  
var      .....  
procedure proc1;  
var      .....  
Begin  
    .....  
End;  
Procedure proc2;  
Var      .....  
Begin  
    .....  
end;  
Begin {Main}  
    Proc1;  
    Proc2;  
End.
```





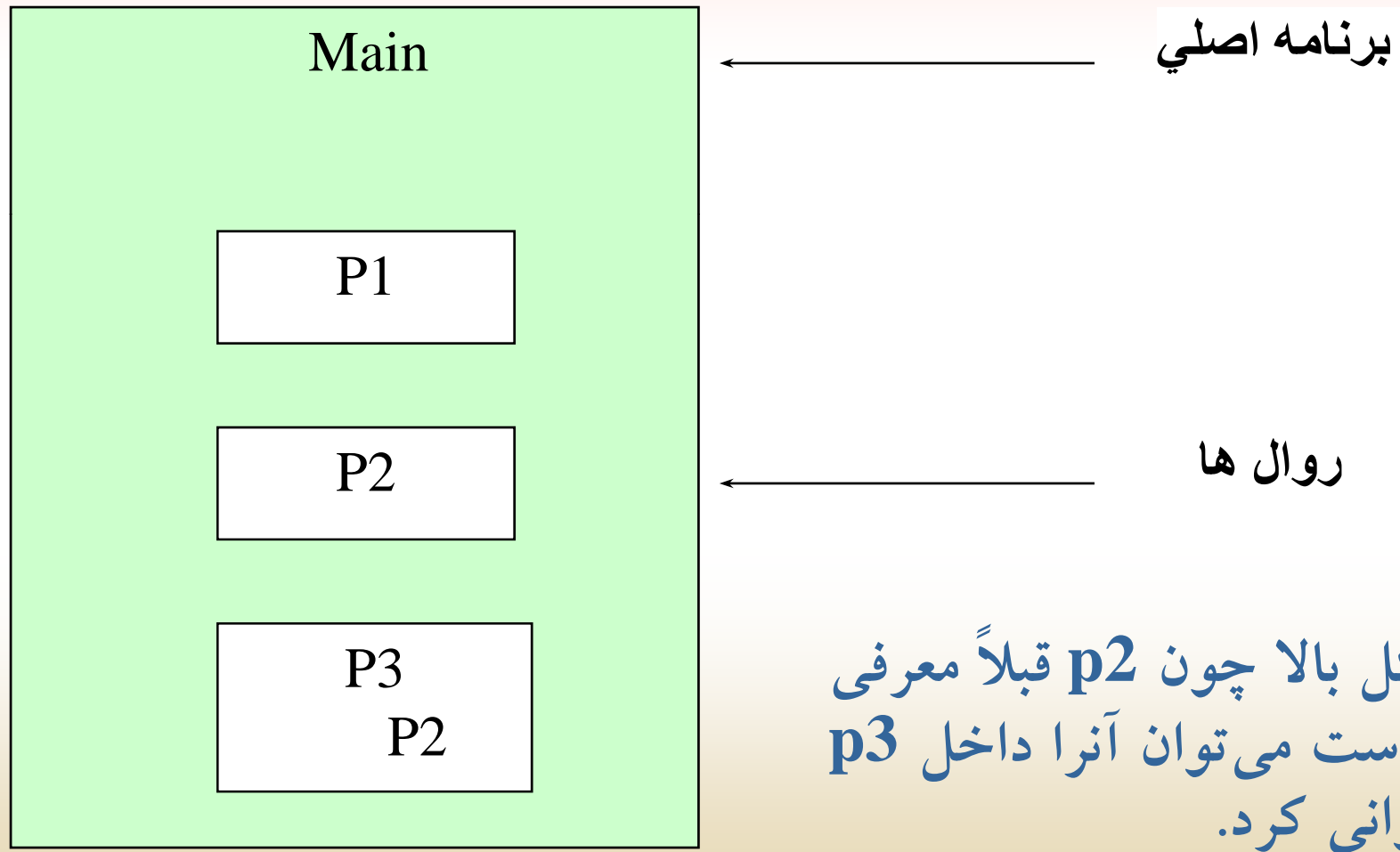
روال‌های `proc1` , `proc2` داخل برنامه اصلی `Main` تعریف شده‌اند. لذا
براحتی می‌توان در برنامه اصلی به آنها دسترسی پیدا کرد.

روال‌ها خود نیز داخل هم می‌توانند قرار بگیرند. دقیقاً مثل روال‌های معمولی
با این تفاوت که در داخل روال باید تعریف شوند. به اینگونه روالها، اصطلاحاً
روال‌های متداخل یا تودرتو می‌گویند.

✓ قانون دوم

اگر روال‌ها در بخش تعاریف یک برنامه به موازات یکدیگر تعریف شوند و نه در داخل هم، در روال‌های بعدی امکان رجوع به روال‌های قبلی وجود خواهد داشت. بعبارت دیگر به روالی می‌توان دسترسی پیدا کرد که قبلاً تعریف شده باشد.

بطور مثال اگر $p1, p2, p3$ سه روال باشند در این صورت می توان بصورت زیر عمل کرد:



✓ قانون سوم

اگر روال‌ها در بخش تعاریف برنامه‌ای بصورت متداخل باشند، در اینصورت از روال درونی می‌توان به روال‌های بیرونی که قبلاً تعریف شده، مراجعه کرد و همچنین از روال درونی به روال‌های بیرونی که به موازات یکدیگر قرار گرفته‌اند، نیز می‌توان دسترسی پیدا کرد.

بطور مثال در شکل زیر روال $p1$ می‌تواند به روال‌های $p2, p3$ دسترسی پیدا کند. و همچنین روال درونی $p4$ می‌تواند به روال‌های $p2, p3$ مراجعه کند.



برنامه اصلي



Main

P3

P2

P1

روال ها



❖ اعلان روال‌ها به روش forward

همانطور که قبلاً اشاره کردیم از روالی می‌توان در روال دیگر استفاده کرد، که قبلاً تعریف شده باشد. در توربوپاسکال نقیصه فوق به کمک اعلان **forward** قابل حل است. بدین صورت که اگر روالی به هنگام تعریف با اعلان **forward** همراه باشد، بدون رعایت از پیش تعریف شدن می‌تواند، در روالهای دیگر ظاهر شود.

شکل اعلان به روش forward بصورت زیر می باشد:

```
Procedure Name ( parameters ) ; forward ;
```



کلمه ذخیره شده
شده

اسم روال

لیست پارامترها

کلمه ذخیره

```
Program forward_Main ;
Var .....
Procedure proc3 ; forward ;
Procedure proc1;
var .....
Begin
.....
end;
Procedure proc2;
var .....
Begin
.....
proc3;
.....
end;
```

مثال :



```
Procedure    proc3;  
var  
    .....  
Begin  
    .....  
End;  
Begin { Main program }  
    .....  
    proc1;  
    proc2;  
    proc3;  
    .....  
End. { End of program }
```



همانطور که ملاحظه می کنید با اعلان روال `proc3` بصورت `forward`، بقیه روالها می توانند به آن دسترسی پیدا کنند.

❖ توابع (Functions)

نوع دیگری از برنامه‌های فرعی، توابع می‌باشند. توابع مانند روال‌ها،

پیمانه‌های مستقلی هستند. با این تفاوت که روال‌ها می‌توانند، تعدادی

خروجی داشته باشند، در حالی که توابع فقط یک خروجی دارند. در روال‌ها

معمولاً خروجی‌ها توسط پارامتر به برنامه اصلی ارجاع داده می‌شود. ولی

در توابع اینکار به نحو دیگر انجام می‌گیرد.

شکل کلی اعلان یک تابع بصورت زیر می باشد:

Function Name (List Of Parameters) : Function Type ;



کلمه

اسم تابع

لیست پارامترها

نوع تابع

ذخیره شده

نکته :

تابع فقط می تواند، یک خروجی داشته باشد. نوع خروجی تابع، همان نوع تابع محسوب می شود. لذا با توجه به نوع خروجی تابع، نوع تابع تشخیص داده می شود. ذکر این نکته خالی از لطف نیست که، نوع پارامترهای صوری توابع معمولاً مقداری هستند چرا که توابع نوعی برنامه های فرعی هستند که فقط یک خروجی برمی گردانند. لذا استفاده از پارامترهای از نوع متغیری پسندیده نمی باشد. (اشکال کامپایلری در توربوپاسکال ندارد ولی در پاسکال استاندارد این کار اشکال کامپایلری دارد.)

مقدار خروجی توابع توسط اسم تابع برگردانده می شود. توابع مثل روالها بعد از قسمت تعاریف برنامه اصلی ظاهر می شود.

شکل کلی توابع بصورت زیر می باشد:

Function Name (Parameters) : Type ;

Var

.....

Begin

.....

.....

.....

Name := Result Of Function ;

End ;

توجه به نکات زیر در بکارگیری توابع ضروری بنظر می‌رسد:

تمام پارامترهای تابع باید از نوع مقداری باشند.

نوع داده نتیجه تابع در انتهای عنوان تابع و بعد از لیست پارامترهای صورتی قرار می‌گیرد.

در بدنه تابع، خروجی تابع با نسبت دادن مقدار به نام تابع مشخص می‌شود.

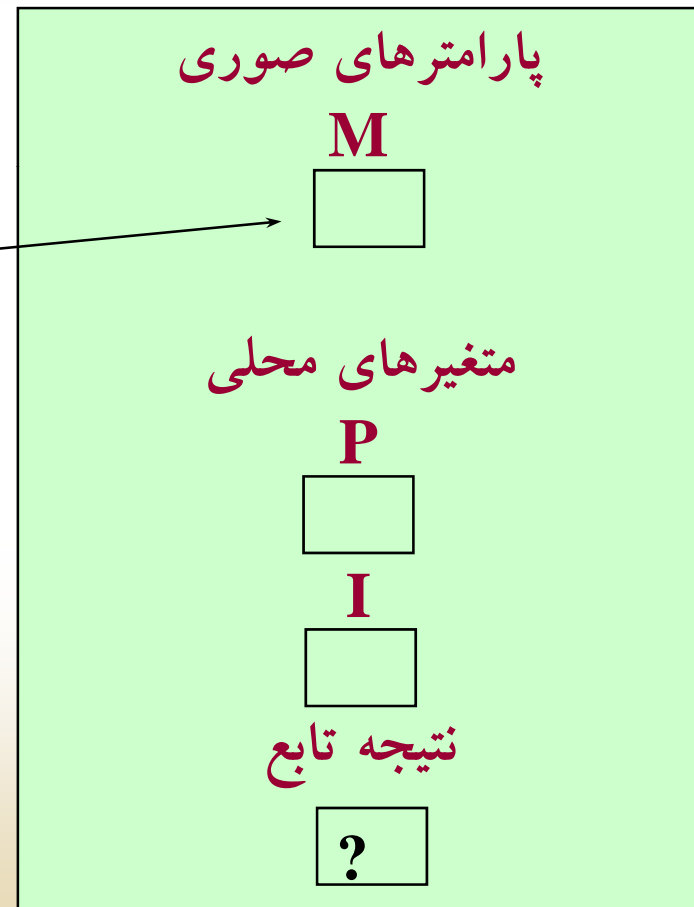
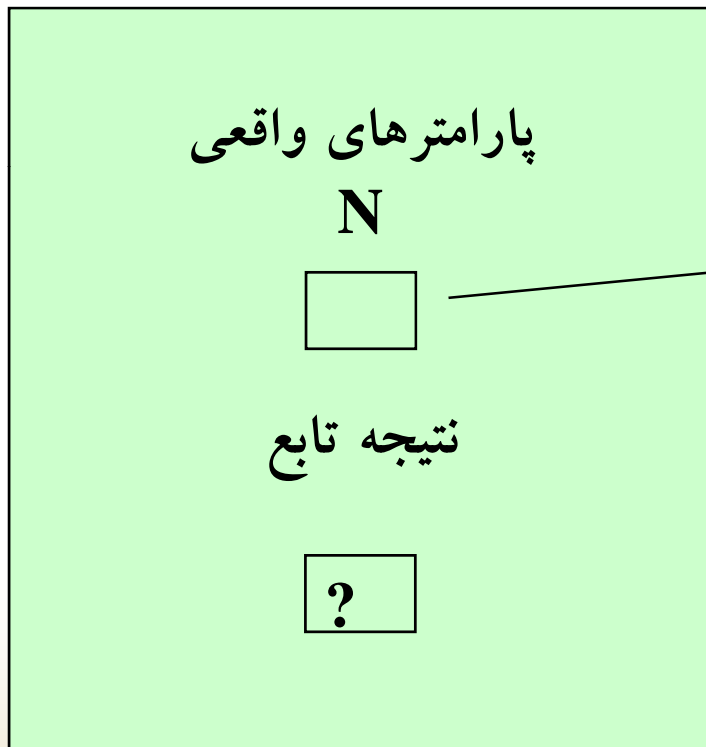
مثال : برنامه‌ای بنویسید که یک عدد از ورودی دریافت کرده سپس
توسط تابعی بنام fact، فاکتوریل عدد را محاسبه
در برنامه اصلی چاپ نماید.

```
Program      Example ;
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.
```

ناحیه داده‌ها بعد از فراخوانی بشکل زیر می‌باشد:

ناحیه داده‌های

Fact ناحیه داده‌های اصلی



❖ 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 می کند، مقادیر فعلی متغیرهای خود را در محلی از حافظه بنام پشته (stack) قرار می دهد، اگر بازگشت به تابع بازگشتی مجدداً صورت گیرد، مقادیر فعلی متغیرها مجدداً بدنبال مقادیر قبلی و اصطلاحاً در پشت مقادیر اولیه قرار می گیرند. 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 می دهد و بهمین ترتیب در بازگشت های بعدی این عمل تکرار می شود تا مقدار تابع محاسبه شود.

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

شرط پایانی

مقادیر متغیرها در آخرین بازگشت به تابع

- .
- .
- .
- .
- .

مقادیر متغیرها در سومین بازگشت به تابع

مقادیر متغیرها در دومین بازگشت به تابع

مقادیر متغیرها در اولین بازگشت به تابع

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

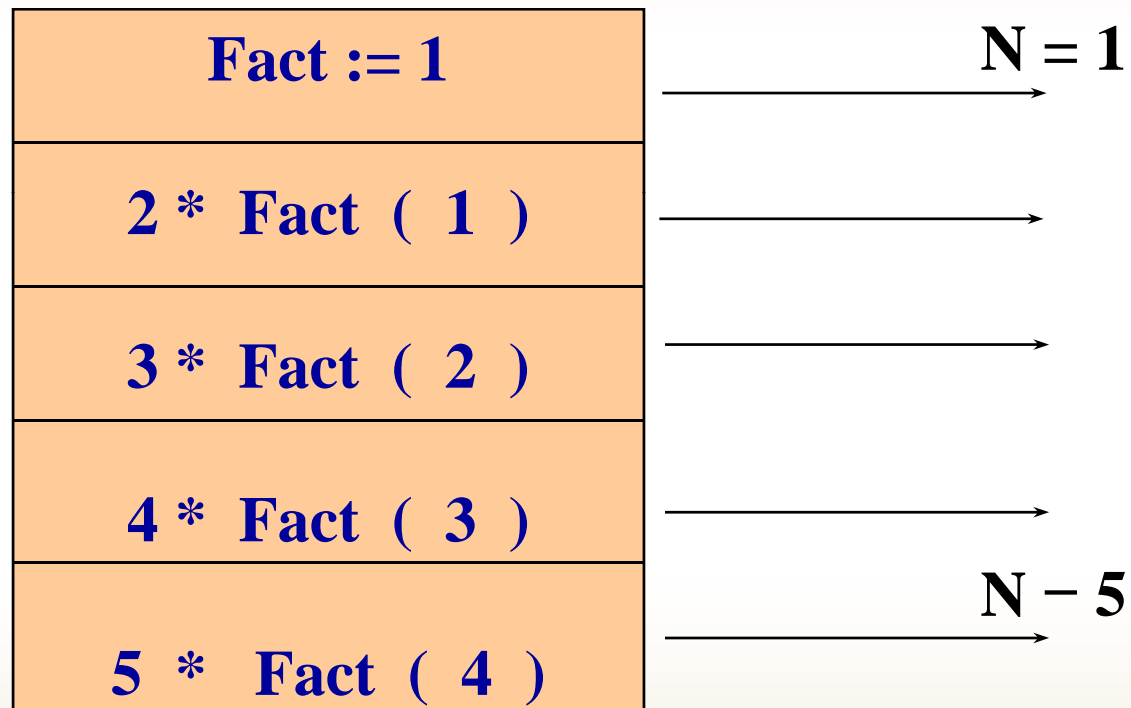
| |
|---|
| مقادیر متغیرها در آخرین بازگشت به تابع |
| <ul style="list-style-type: none">••••• |
| مقادیر متغیرها در سومین بازگشت به تابع |
| مقادیر متغیرها در دومین بازگشت به تابع |
| مقادیر متغیرها در اولین بازگشت به تابع |



مثال : برنامه‌ای 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
```

در برنامه بالا ترتیب قرار گرفتن مقادیر در پشته بصورت زیر می باشد:



مثال : تابعی برای محاسبه بزرگترین مقسوم علیه مشترک دو عدد صحیح N, M بنویسید.

```
Function    GCD ( N , M: integer ) : Integer ;  
Begin  
    If ( N <= M ) and ( M Mod N = 0 ) Then  
        GCD:= N  
    Else  
        GCD:= GCD ( M ,N mod M ) ;  
End ;
```

❖ مقایسه توابع و روال‌ها

توابع و روال‌ها هر دو برنامه‌های فرعی هستند، که بطور مستقل وظایفی را بر عهده دارند. ولی در این میان از بعضی جنبه‌ها متفاوت می‌باشند که عبارتند از:

۱. نحوه فراخوانی آنها با هم متفاوت است. روال‌ها از طریق عبارات روال فراخوانا می‌شوند، در صورتی که فراخوانی تابع توسط عبارات مقایسه‌ای و یا تخصیص نتیجه به یک متغیر صورت می‌گیرد.

۲. هنگام اعلان یک تابع، نوع تابع یا نوع نتیجه حاصل از تابع باید ذکر شود، در صورتی که روال‌ها نیازی به این کار ندارند.

۳. توابع فقط یک خروجی برمی‌گردانند، ولی روال‌ها می‌توانند، چندین خروجی برگردانند. در ضمن نتیجه توابع توسط اسم تابع فرستاده می‌شود ولی روال‌ها از طریق پارامترها، نتایج را برمی‌گردانند.

❖ **طریقه ارسال آرایه‌ها به توابع و روال‌ها**

آرایه‌ها خود مجموعه‌ای از داده‌ها می‌باشند، لذا برای ارسال آنها نمی‌توانیم از روش معمولی ارسال پارامترها استفاده کنیم. و بطور مستقیم نمی‌توان آنها را به برنامه‌های فرعی انتقال داد. برای ارسال آرایه‌ها به عنوان پارامتر به برنامه‌های فرعی از دستور **Type** استفاده می‌کنند. دستور **Type** قبل از تعاریف برنامه اصلی بکار می‌رود و توسط این دستور در واقع یک نوع ساده‌سازی در تعاریف فراهم می‌شود. و با این ساده‌سازی می‌توان داده‌های ساخت یافته را به برنامه‌های فرعی منتقل کرد.

فرم کلی دستور **Type** در بخش تعاریف بصورت زیر است:

Type تعریف داده ساخت یافته = شناسه

برای مثال یک آرایه از نوع صحیح را تعریف می کنیم:

```
Type      No = array [1...100] of integer ;
```

در اینصورت تعریف آرایه به، **No** نسبت داده می شود. اگر بخواهیم متغیری از نوع آرایه بالا تعریف کنیم، آنرا از نوع **No** تعریف می کنیم:

```
Var
```

```
    Number: No ;
```


مثال : برنامه‌ای بنویسید که یک آرایه حداکثر ۱۰۰ عنصری را از ورودی دریافت کرده، سپس عدد را از ورودی خوانده، توسط تابعی بنام **Search** محل وقوع عدد در آرایه را نمایش دهد.

Program Example ;

Type a = array [1.. 100] of integer ;

Var

B: a ; N: Word ;

Function Search (b:a; N: Word ; x: integer) : Word ;

Var

I , Index: Word ; Flag: Boolean ; { local variable }

Begin


Index:= 0 ;

Flag:= TRUE ;



```
While ( I <= N ) And ( flag ) do
    Begin
        If b[ I ] = x Then
            Begin
                Index:= I ;
                Flag:= FALSE ;
            End ;
            Inc ( I ) ;
        End ;
    Search:= Index ;
End ;
{*****}
```





```
Begin { Main }
  Writeln ( ' Enter Number ' ) ;
  Readln ( N ) ;
  For I:= 1 To N Do
    Readln ( b[ I ] ) ;
  Writeln ( ' Enter Number ' ) ;
  Readln ( x ) ;
  If Search ( b , N , x ) = 0 Then
    Writeln ( ' Not Found ' ) ;
  Else
    Writeln ( Search( b , N , x ) ) ;
End. { End Of Program }
```

❖ تمرینات

• روال Down را در نظر بگیرید:

```
Procedure    Down ( N: Integer ) ;  
Begin  
    While N > 0 do  
        Begin  
            Write ( N: 3 ) ;  
            Dec ( N ) ;  
        End ;  
    End ;  
End ;
```

الف) وقتی روال بصورت **down (5)** فراخوانی شود، چه چیزی چاپ می شود.
ب) مقدار پارامتر واقعی **N** بعد از اجرای روال چیست؟

• تابع بازگشتی زیر را در نظر بگیرید:

```
Function      MyStery ( M , N: Integer ) : integer ;  
Begin  
    If  N = 1 Then  
        MyStery:= 1  
    Else  
        MyStery:= M * MyStery ( M , N – 1 ) ;  
End ; { End Of Function }
```

الف) تابع بازگشتی بالا را به ازاء مقادیر $M=5$, $N=4$ فراخوانی کنید.

ب) تابع بازگشتی بالا را به ازاء مقادیر $M=3$, $N=5$ فراخوانی کنید.

❖ تمرینات برنامه‌نویسی

▪ برنامه‌ای بنویسید که اطلاعات حداکثر ۱۰۰ دانشجو که عبارتند از:

فامیلی
اسم
شماره دانشجویی

Family
Name
St_No

را از ورودی دریافت کرده سپس:

الف) توسط روالی بنام **sort** اطلاعات را برحسب شماره دانشجویی مرتب کند.

ب) توسط تابعی بنام **search** شماره دانشجویی فردی را از ورودی دریافت نماید، در صورتی که شخص موردنظر در لیست باشد، سایر اطلاعات آنرا نمایش دهد.

■ آرایه‌ای از نوع صحیح با حداکثر ۱۰۰ عنصر را در نظر بگیرید.

برنامه‌ای بنویسید که ابتدا آرایه را از ورودی دریافت کرده سپس با استفاده از یک تابع بازگشتی بیشترین مقدار آرایه را محاسبه نماید.

■ دو آرایه مرتب حداکثر ۱۰۰ عنصری که شامل اسامی افراد می‌باشد را در نظر بگیرید.

برنامه‌ای بنویسید که نخست دو آرایه را از ورودی دریافت نموده سپس توسط روالی بنام **merge** این دو آرایه در هم ادغام نموده و در آرایه سومی قرار دهد.

فصل 12

مجموعه‌ها و داده‌های شمارشی

هدفهای کلی

- مفهوم مجموعه و داده‌های شمارشی در زبان پاسکال
- مجموعه و داده‌های شمارشی به عنوان متغیر
- استفاده از مجموعه‌ها و داده‌های شمارشی در برنامه

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

■ مفهوم مجموعه و داده‌های شمارشی را درک کند.

■ بتواند در صورت لزوم از مجموعه و داده‌های شمارشی استفاده کند.

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. بهر حال به عنوان ابزارهایی از زبان پاسکال هستند که در مواقعی، ضروری بنظر می‌رسند و مسئله را به صورتی قابل فهم 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 ۲۵۶ 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 می کنیم:

Type

Name= Set of Type Of Set;

مثلا :

Type

Digit_type = 0..9;

Digit = Set of Digit_type

مثال :

Var

ch: set of char;

→ **#0..#255**

bool: set of boolean;

→ **false..true**

num: set of byte;

0..255

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** و گرنه **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.

Var

a: set of byte;

Begin

a:= [1..3,4];

if 3 in a then write('3 is in set a')

چاپ مي شود

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 اعضای 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;

d:= a*b;

e:= a-b;

End.

c=[0,1,2,3,4,5]=[0..5]

d=[2,3]

e=[0,1]

در استفاده از مجموعه ها به نکات زیر توجه کنید :

۱. آرگومان ورودی روالها می تواند مجموعه باشد که قبلاً در تایپ تعریف شده باشد، نه اینکه مستقیم در روال به عنوان آرگومان بیاید. ولی خروجی یک تابع نمی تواند از نوع مجموعه باشد.

۲. برای نوشتن یا خواندن مجموعه ها باید عضو به عضو عملیات صورت بگیرد و مستقیماً توابع **Read**، **Write** روی آنها کار نمی کنند.

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  
    Writeln ('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 می تواند مقادیر داخل داده های شمارشی را بگیرد:

```
cars:= Pride;
```

```
cars:= PK;
```

به مثال زیر توجه کنید :

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 :

متغیر داده شمارشی شبیه مجموعه‌ها حداکثر دارای ۲۵۶ عضو می‌تواند باشد که از صفر تا ۲۵۵ شماره‌گذاری می‌شود و لذا یک بایت حافظه را اشغال می‌کند.

✓ عملیات روی داده‌های شمارشی

داده‌های شمارشی همانند هر نوع تایی می‌توانند در **type** برنامه تعریف شود

و به عنوان آرگومانهای روالها و یا خروجی توابع می‌تواند در نظر 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 و گرنه

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.
```

❖ تمرینات

▪ اگر $A=[1,3,5,7]$ و $B=[2,4,6]$ و $C=[1,2,3]$ باشد عبارات زیر را ارزیابی کنید.

الف- $A + (B - C)$

ب- $A + (B * C)$

ج- $A + B + C$

د- $2 \text{ in } A$

■ برنامه‌ای بنویسید تا مجموعه‌ای از کاراکترهای کوچک را گرفته و به یک مجموعه از کاراکترهای حروف بزرگ متناظر با آن تبدیل کند.

■ به کمک داده‌های شمارشی، برنامه‌ای بنویسید که نام دانشجویان یک کلاس را در برگرد و هر دانشجو نیز یک شماره داشته باشد. سپس با دریافت نام او شماره او در نمایشگر چاپ شود.

فصل 13

رکوردها (Records)

هدفهای کلی

- مفهوم رکورد و اجزای آن
- انواع رکوردها و استفاده از آن در برنامه
- معرفی مزایای رکوردها در برنامه

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- مواقع لزوم رکورد را تشخیص دهد.
- از رکوردها در برنامه‌اش استفاده کند.
- برنامه‌های بزرگ و با داده‌های زیاد بنویسد.

مقدمه

نوع داده ساخت یافته‌ای 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 از نوع 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 ساخت یافته 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 می شود که مطابق زیر می باشد:

Type

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

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

Var

```
S: Student;
```

✓ دسترسی به فیلدهای رکورد

برای دسترسی به فیلدهای رکورد از علامت '!' استفاده می‌شود. یعنی بصورت زیر:

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 در نظر 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;

۱۲+۱۱+۱۰+۱۰=۴۳ بایت فضا اشغال می شود.

❖ رکوردهای تودرتو

فیلدهای یک رکورد می توانند از هر نوعی باشد، از جمله می توانند از نوع رکورد دیگری باشند. در اینجا نیز مشابه قبل دسترسی به همان صورت می باشد 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;

در اینجا 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** می توان 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

هنگامی که ما تعدادی داده مشابه داریم ولی در هر یک، 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 ۱۰ 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 می باشد که دارای سه فیلد مطابق جدول فوق می باشد.

`S[1].name`

`S[1].id`

`S[1].age`

`S[2].name`

`S[2].id`

..

..

..

برای دسترسی به آرایه فوق داریم:

❖ 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 `char`، `integer` 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 : می خواهیم برنامه ای بنویسیم که رکورد stu را برای ۳۰ 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.
```


❖ تمرینات

■ برای گونه‌های حیوانات یک پارک نظیر پرندگان وحشی رکوردی شامل نام، دسته، محیط زندگی، سن بسازید. در نظر بگیرید که در پارک ۱۰۰ گونه از یک پرنده وحشی وجود دارد (راهنمایی: آرایه‌ای از رکورد با فیلدهای گفته شده بسازید).

■ برنامه‌ای بنویسید که برای نگهداری تاریخ به صورت روز، ماه و سال برای ۱۰۰ سال بکار رود. در واقع به صورت تقویم باشد و جمله‌ای در مورد آن تاریخ را نگهداری کنید؟ (راهنمایی: آرایه‌ای ۳۶۶ تایی از روز به همراه آرایه‌ای از ۱۲ ماه و یک سال و یک رشته را به صورت یک رکورد در نظر گرفته سپس برای ۱۰۰ سال بصورت یک آرایه صدتایی در نظر بگیرید.)

فصل 14

فایلها (Files)

هدفهای کلی

- مفهوم فایل و انواع آن
- موارد استفاده از فایلها

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- در برنامه‌اش از فایلها استفاده کند..
- برای برنامه‌های با داده‌های زیاد از فایل استفاده کند..
- با فایل‌های داده‌ای و بازی برنامه‌نویسی کند.

مقدمه

برای ذخیره دائمی داده‌ها از ساختاری به نام فایل (File) استفاده می‌کنیم. تاکنون همه عملیات لازم در حافظه اصلی انجام می‌گرفت که گذرا و فقط به زمان اجرای برنامه و روشن بودن کامپیوتر بستگی داشت. ولی در فایلها چنین نیست، بلکه داده‌ها در فایلهایی قرار دارند که حتی بعد از خاموش کردن کامپیوتر، بعدها قابل دسترسی است.

❖ فایل‌های متنی (Text)

یک فایل متنی از تعداد کاراکتر تشکیل شده است که با یک اسم در روی دیسک ذخیره شده است. چون فایل از نوع متنی است می‌توانید داده‌های داخل آنرا مشاهده کنید.

ساختار داده‌ها در یک فایل متنی بدین صورت است که تعدادی خط وجود دارد که به علامت **EOLN** ختم می‌شوند و در انتهای فایل نیز علامت **EOF** قرار دارد. طول خطوط نامشخص است و اندازه فایل ممکن است بسیار بزرگ یا کوچک و یا تهی باشد.

✓ طریقه ایجاد یک فایل متنی

1. تعریف یک متغیر فایلی
2. نسبت دادن اسم فایل به متغیر فایلی
3. **Rewrite** ایجاد فایل با دستور

تعریف یک متغیر فایلی

Name : Text;

نسبت دادن اسم فایل به متغیر فایلی

Assign(FileVar , 'NameFile')

Rewrite ایجاد فایل با دستور

Rewrite(FileVar)

مثال : برنامه‌ای بنویسید که از ورودی ۱۰ عدد دریافت کند و آنها را به ترتیب هر کدام در یک خط از یک 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.
```

بعد از نوشته شدن اطلاعات در فایل ، فایل باید بسته شود برای اینکار از
روال زیر استفاده می شود :

```
Close(FileName)
```

✓ طریقه خواندن اطلاعات از یک فایل متنی

1. تعریف یک متغیر فایلی
2. نسبت دادن اسم فایل به متغیر فایلی
3. **Reset** باز کردن فایل برای خواندن با دستور

باز کردن فایل برای خواندن با دستور

Reset (FileName)

دستور **Readln** و **Read** برای خواندن اطلاعات از فایل بکار می روند .

Readln بعد از خواندن داده‌های مورد نظر به سطر بعد می‌رود. این عمل

برای **Writeln** نیز چنین می‌باشد. ولی قبل از استفاده از این دستورات باید

فایل متنی برای خواندن یا نوشتن باز شود. دستور **Reset** فایل متنی را برای

خواندن باز می‌کند و مکان نما را ابتدای فایل می‌برد.

دستور **Rewrite** فایل متنی را برای نوشتن در آن باز کرده، و مکان نما را به

ابتدای آن می‌برد.

حال با فرض موجود بودن فایل متنی 'test.dat' می‌خواهیم اعداد داخل فایل را که پنج تا هستند را از فایل خوانده با هم 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 : برنامه‌ای بنویسید که یک فایل متنی شامل چند جمله را از ورودی دریافت کرده و یک کپی از فایل در خروجی 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,str1);

reset(f1);

assign(f2,str2);

rewrite(f2);

در این برنامه ابتدا دو فایل به نامهایشان نسبت داده می‌شوند و ترتیب برای خواندن و نوشتن باز می‌شوند. سپس توسط **eof** شرط خاتمه فایل ورودی یعنی **f1** بررسی می‌شوند و در یک حلقه تو در تو **while** شرط خاتمه خطوط نیز توسط **eoln** بررسی شده و کاراکترها از فایل ورودی خوانده شده و در فایل خروجی نوشته می‌شوند.

```
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.
```

❖ فایل‌های دودویی و نوع‌دار (Binary & Typed)

این نوع فایل‌ها در زبان پاسکال از تایپ‌های مختلف `record`, `array`

`char`, `integer`, `read` و ... تشکیل شده است که نیاز به پردازش متنوع

شبه مرتب‌سازی، جستجو، حذف و ... دارند. این فایل‌ها پس از ایجاد توسط

برنامه، قابل رؤیت توسط ویرایشگرها نیستند، بلکه به صورت کدهای اسکی

می‌باشند یعنی دودویی می‌باشند. نحوه دسترسی به اطلاعات آنها نیز به

صورت تصادفی (Random) است.

✓ طریقه ایجاد یک فایل نوع دار

1. تعریف یک متغیر فایلی
2. نسبت دادن اسم فایل به متغیر فایلی
3. Rewrite ایجاد فایل با دستور

تعریف یک متغیر فایلی

Name : File Of FileType ;

نسبت دادن اسم فایل به متغیر فایلی

Assign(FileVar , 'NameFile')

Rewrite ایجاد فایل با دستور

Rewrite(FileVar)

نحوه تعریف یک فایل دودویی نوع دار از انواع مختلف در زیر آورده شده است:

```
Const n=100;
```

```
Type
```

```
    Student = Record
```

```
        Name: string[10];
```

```
        Family: String[15];
```

```
        Age: integer;
```

```
        ID: integer;
```

```
    end;
```

```
Sarray = Array [1..n] of student;
```

مثال :

Var

Bf2: file of integer;

Bf3: file of student;

Bf1: file of char;

Bf6: file of set of 'A'..'Z';

فایل دودویی از نوع صحیح

فایل دودویی از نوع رکورد دانشجو

فایل دودویی از نوع کاراکتر

فایل دودویی از مجموعه A تا Z

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

```
Const n= 10;
```

```
Var
```

```
    Bf: file of string [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.
```

در جدول صفحه بعد تمام روالهای کتابخانه‌ای پاسکال جهت کار با فایلها آورده شده است که به همراه توضیحات لازمه جهت کار با آنها می‌باشد. همچنین برای هر تابع و رویه محل مورد استفاده آن در ستونی مشخص شده است .

| فایل‌های نوع دار | فایل‌های متنی | عملیات مربوطه | نام دستور | ردیف |
|------------------------|------------------|--|----------------|------|
| + | + | نام فایل را به متغیر فایل نسبت می‌دهد. | Assign | 1 |
| + | + | فایل مربوطه را برای خواندن باز می‌کند. | Reset | 2 |
| + | + | فایل مربوطه را برای نوشتن باز می‌کند. | Rewrite | 3 |
| + | + | خواندن از فایل | Read | 4 |
| × | + | خواندن از فایل متنی | Readln | 5 |
| + | + | نوشتن در فایل | Write | 6 |
| × | + | نوشتن در فایل متنی | Writeln | 7 |
| × | + | مشخص نمودن انتهای خط | Eoln | 8 |
| + | + | مشخص نمودن انتهای فایل | Eof | 9 |
| × | + | بازنمودن فایل متنی جهت اضافه نمودن | Append | 10 |

| | | | | | |
|---|---|---|---|-------------------|----|
| + | + | + | بستن فایل | Close | 11 |
| + | + | × | انتقال و حرکت مکان‌نما در فایل دودویی | Seek | 12 |
| × | × | + | مشخص نمودن رسیدن به انتهای خط فایل متنی | Seekoln | 13 |
| × | × | + | مشخص نمودن رسیدن به انتهای فایل متنی | Seekeof | 14 |
| + | + | × | شماره رکورد از انتهای فایل | Filefos | 15 |
| + | + | × | تعداد رکوردهای فایل جاری را برمی‌گرداند | Filesize | 16 |
| + | + | × | حذف یک رکورد از انتهای فایل | Truncate | 17 |
| + | + | + | برای حذف فایلها بکار می‌رود. | Erase | 18 |
| + | + | + | برای تغییر نام فایلها بکار می‌رود. | Rename | 19 |
| × | × | + | محتویات بازفایل متنی را به دیسکت انتقال می‌دهد. | Flush | 20 |
| × | × | + | اندازه بافر برای فایلهای متنی را برمی‌گرداند. | SetTextbuf | 21 |
| + | × | × | خواندن یک بلاک از فایل بدون نوع | Blockread | 22 |
| + | × | × | نوشتن یک بلاک از فایل بدون نوع | Blockwrite | 23 |

❖ مثال های حل شده

مثال : برنامه ای 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 : رکوردی از کتابها در نظر گرفته، تعداد 10 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.

❖ تمرینات

■ برنامه‌ای بنویسید که از یک فایل متنی از اعداد حقیقی که در سطرها و ستونها مختلف قرار دارند، میانگین داده‌ها را بدست آورده و در مانیتور چاپ کند.

■ برنامه‌ای بنویسید که در انتهای یک فایل متنی، همان فایل متنی را اضافه کند.

■ برنامه‌ای بنویسید که در یک فایل نوع دار از کاراکترها، اطلاعات آماری هر کاراکتر را بدست آورده، یعنی تعداد هر کاراکتر را بدست آورد. مثلاً کاراکتر 'A' به تعداد ۴۵ تا و کاراکتر 'B' به تعداد ۳۶ تا و...

فصل 15

تحلیل الگوریتمها

هدفهای کلی

- مفهوم و تعریف الگوریتم
- مفهوم کارایی یک الگوریتم
- مرتبه یک الگوریتم

هدفهای رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

■ برنامه خود را تحلیل زمانی نماید.

■ مرتبه الگوریتم را بدست آورد.

■ الگوریتمهای بازگشتی را تحلیل نماید.

مقدمه

تحلیل یک الگوریتم یعنی ارزیابی روشهای مختلف حل آن مسئله، بررسی و محاسبه بهترین و بدترین حالتها، بصورتی که با توجه به شرایط بهترین حالت را بتوان انتخاب کرد. الگوریتم در واقع تعداد محدودی از دستورالعملها می باشد که بترتیب اجرا می شوند و هدف خاصی را دنبال می کنند

❖ تعریف مرتبه یا پیچیدگی الگوریتم (O بزرگ)

برای بدست آوردن بدترین حالت اجرای یک الگوریتم یا مرتبه پیچیدگی الگوریتم از

O بزرگ استفاده می کنیم و بنا به تعریف عبارتست از $f(n)=O(g(n))$ و

می خوانیم $f(x)$ از مرتبه $g(x)$ می باشد، اگر و تنها

$$\exists c > 0, \forall n > n_0 \Rightarrow f(x) < c.g(x)$$

مثال : $f(n) = 2n + 1$ می توان با در نظر گرفتن $g(n) = n$ و $c = 3$ رابطه زیر حاصل می شود :

$$f(n) = 2n + 1 \leq cg(n) = 3n \quad n \geq 1$$

لذا می توان گفت، پیچیدگی زمانی $f(n)$ از مرتبه $O(n)$ می باشد.

❖ بدست آوردن مرتبه الگوریتمها

برای بدست آوردن مرتبه اجرای الگوریتمها باید به دقت بررسی شود و تعداد

تکرار آن الگوریتم بدست آورده شود. یعنی در واقع تعداد تکرارها بدست

آورده شود و سپس با هم جمع شده و مطابق مثالهای قبلی به پیچیدگی زمان

واقعی رسید. برای این منظور از الگوریتمهای ساده شروع کرده و به پیچیده

می‌رسیم.

فرض کنید الگوریتمی دارای حلقه‌های ساده به شکل زیر باشد:

```
x:= 0;  
For I:=1 to n do  
    x:=x+1;
```

ه در آن n تعداد ورودی می‌باشد. یعنی حلقه **for** بستگی به پارامتر n دارد. همان طور که از تعریف حلقه **for** برمی‌آید تعداد $1+1-n$ تکرار یعنی n تکرار وجود دارد و لذا قطعه برنامه فوق از مرتبه $O(n)$ می‌باشد.

مثال : مرتبه تابع زیر را بدست آورید:

```
Function fact ( n : integer ) : integer ;  
var  
    f,i: integer;  
begin  
    f:=1;  
    for I := 1 to n do  
        f:=f*i;  
    fact:=f;  
end;
```

تابع فوق فاکتوریل عدد n را بر می گرداند. همانطور که می بینید تابع شامل یک حلقه `for` ساده بوده که از یک تا n متغیر بوده و لذا n بار تکرار می شود.

۲ جمله ساده نیز وجود دارد که زمان های ثابتی دارند.

بنابراین مرتبه تابع بالا برابر است با :

$O(n)$

❖ تمرینات

✓ پیچیدگی زمانی توابع زیر را بدست آورید:

$$1. f_1(n) = n + 2n^2 + 100$$

$$2. f_2(n) = n^3 + n^{\log n} + n + 2$$

$$3. f_3(n) = 2^n + 5^n + \log n! + n^n$$

$$4. f_4(n) = \log n + \sqrt{n} + 1$$

$$5. f_5(n) = n^2 \log n + n^{\log n} + (\log n)!$$

✓ پیچیدگی زمانی تابع زیر را بدست آورید:

```
Function test( n : integer ) : integer ;
```

```
var
```

```
    i,x: integer;
```

```
Begin
```

```
    x:=0;
```

```
    for i:=1 to n do
```

```
        x:=x+1;
```

```
    for i:=1 to n do
```

```
        for j:=1 to n do
```

```
            x:=x+1;
```

```
    test:=x;
```

```
End.
```


پاپان

www.salampnu.com

سایت مرجع دانشجوی پیام نور

- ✓ نمونه سوالات پیام نور : بیش از ۱۱۰ هزار نمونه سوال همراه با پاسخنامه
- تستی و تشریحی
- ✓ کتاب ، جزوه و خلاصه دروس
- ✓ برنامه امتحانات
- ✓ منابع و لیست دروس هر ترم
- ✓ دانلود کاملاً رایگان بیش از ۱۴۰ هزار فایل مختص دانشجویان پیام نور

www.salampnu.com