مبانی کامپیوتر و برنامه نویسی

دانشگاه آزاد اسلامی واحد کرج

مدرس : مهدی خدری

❖ 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 .

❖ اهداف 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 مفا هیم 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++c را در برنامه ها بکار ببرد.
- ✓ از توابع و روال های استاندارد زبان++c در صورت لزوم استفاده نماید.
- ✓ از توابع ، روال ها برای جدا کردن قطعات برنامه استفاده کند

❖ فصل اول: مفاهیم پایه کامپیوتر

کامپیوتراز دو بخش اصلی تشکیل شده است:

- **سخت افزار (Hardware):** اجزای فیزیکی (قابل لمس ودیدنی) در کامپیوتر ، مانند مانیتور ، ماوس ، کیبورد و...
- **نرم افزار(Software):** مجموعه ای از یک یا چند برنامه که برای انجام کار خاصی توسط یک برنامه نویس نوشته شده است.

❑ نرم افزاهای سیستمی:

○ **سیستم عامل(Operating System)** : در واقع ارتباط بین سخت افزار و نرم افزار کاربردی را کنترل می نماید. به عبارت دیگر به عنوان واسط سخت افزار و برنامه های کاربردی انجام وظیفه می کند. مثال: **Dos , Windows ,Unix ,...**

○ **مترجم(compiler):** زبان برنامه نویسی را به زبان قابل فهم کامپیوترترجمه میکند.

❑ نرم افزارهای کاربردی: **Photoshop ,AutoCAD ,Word,...**

❖ واحد های اندازه گیری داده در کامپیوتر

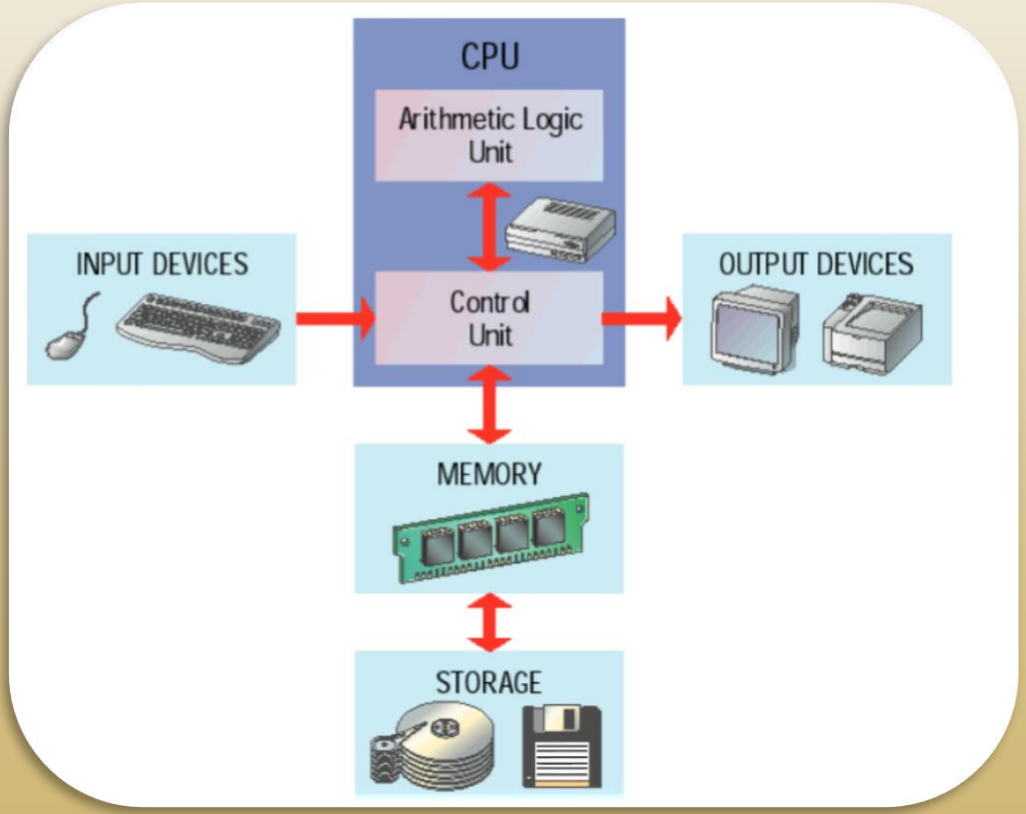
- ✓ **بیت:** کوچکترین واحد اطلاعاتی در کامپیوتر که می تواند مقادیر ۰ یا ۱ را داشته باشد.
- ✓ **بایت:** به مجموع ۸ بیت یک بایت گوییم.

  مثلاً: ۰۱۰۰۱۰۱۱
- ✓ **۱ کیلوبایت:** ۱۰۲٤ بایت = $۱۰^{۳}$ = $۲^{۱۰}$
- ✓ **۱ مگابایت:** ۱۰۲٤ کیلوبایت = $۱۰^{٦}$ = $۲^{۲۰}$
- ✓ **۱ گیگابایت:** ۱۰۲٤ مگابایت = $۱۰^{۹}$ = $۲^{۳۰}$
- ✓ **۱ ترابایت:** ۱۰۲٤ گیگابایت = $۱۰^{۱۲}$ = $۲^{٤۰}$

❖ اجزای اصلی یک کامپیوتر

هر سیستم کامپیوتری از چهار قسمت اصلی تشکیل می شود:

واحد پردازش مرکزی، واحد ورودی ، واحد خروجی و حافظه



❖ اجزای اصلی یک کامپیوتر

واحد پردازش مرکزی(Central Processing Unit): تراشه ی الکترونیکی است که انجام عملیات پردازشی، منطقی، ریاضی و کنترلی را بر عهده دارد. این قسمت اصلی ترین بخش و در واقع مغز کامپیوتر محسوب می شود.

واحد ورودی : واحدی که داده ها را از دستگاه های ورودی دریافت کرده و جهت تبدیل آن ها به داده های قابل فهم کامپیوتر، آن را به واحد کنترل (CU) تحویل می دهد .قابل ذکر است که واحد کنترل بخشی از**CPU** می باشد

دستگاه های ورودی شامل موس، کیبورد، اسکنر، وسایل بازی و .... هستند.

❖ اجزای اصلی یک کامپیوتر

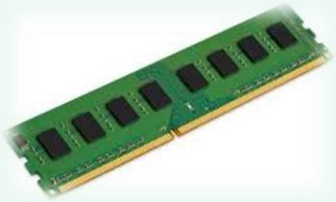
واحد خروجی: واحدی است که اطلاعات تولید شده توسط کامپیوتر را از حافظه اصلی دریافت کرده و به دستگاه های خروجی منتقل می نماید.

دستگاه های خروجی عبارتند از: چاپگر، پلاتر یا نقشه کش، مونیتور یا صفحه نمایش، اسپیکر و ...

❖ اجزای اصلی یک کامپیوتر

حافظه ها: حافظه مکانی است که اطلاعات موقتاً یا دائماً در آن قرار می گیرد.

- **حافظه اصلی(Main Memory): هر برنامه برای اجرا ابتدا بایستی در حافظه اصلی قرار بگیرد و سپس توسط CPU اجرا شود.**

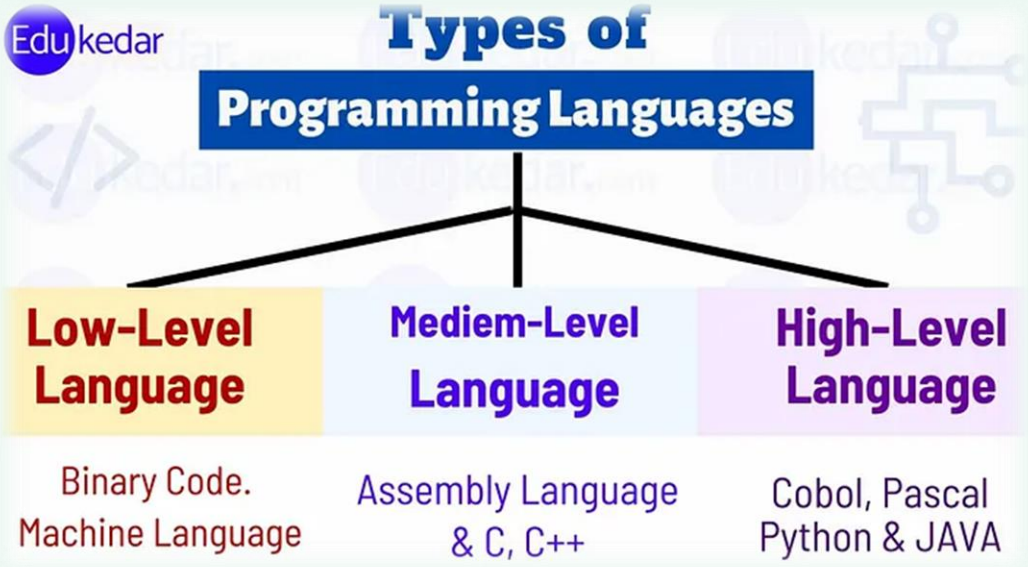


- **حافظه جانبی (Secondary Memory): پس از اجرای برنامه، برای ذخیره اطلاعات برنامه، آن ها را روی حافظه ی جانبی ذخیره می کنند.**

❖ زبان های برنامه نویسی

هر زبان برنامه نویسی به مجموعه أی از علایم، قواعد و دستورالعمل ها گفته می شود که امکان ارتباط با کامپیوتر را جهت بیان کاری یا حل مسئله ای فراهم می کند.

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



- کامپایلر برنامه نوشته شده در یک زبان سطح بالا را به برنامه قابل فهم کامپیوتر تبدیل می کند.

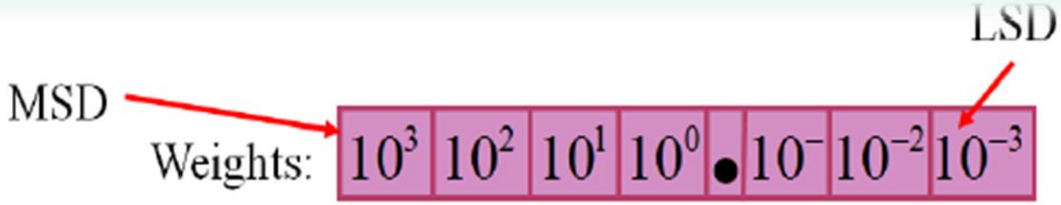
❖ آشنایی با سیستم اعداد(مرور سیستم دهدهی)

اعداد: ۰ تا۹ **پایه: ۱۰**

برای اعداد بزرگتر از ۹، یک رقم با اهمیت تر به سمت چپ اضافه کنید .

مثال: ۹<۱۹

هر محل دارای یک وزن است:



به عنوان مثال عدد ۱۹۳۶,۲۵ را می توان به صورت زیر نمایش داد:

$$1\times10^3+9\times10^2+3\times10^1+6\times10^0+2\times10^{-1}+5\times10^{-2}$$

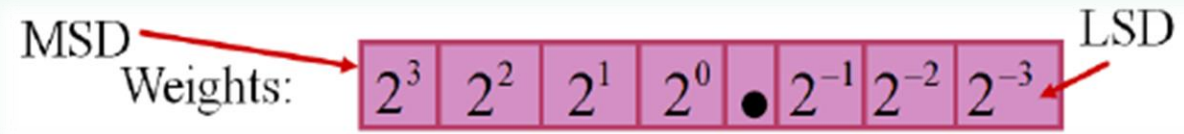
❖ آشنایی با سیستم اعداد(سیستم دودویی)

❑ تبدیل مبنای ۲ به مبنای ۱۰

اعداد: ۰ و ۱ **پایه: ۲**

برای اعداد بزرگتر از ۱، یک رقم با اهمیت تر به سمت چپ اضافه کنید. مثال: ۱<۱۰

هر محل دارای یک وزن است:



به عنوان مثال تبدیل مبنای ۲ به مبنای ۱۰ برای عدد ۱۰۱۱۱,۰۱ :

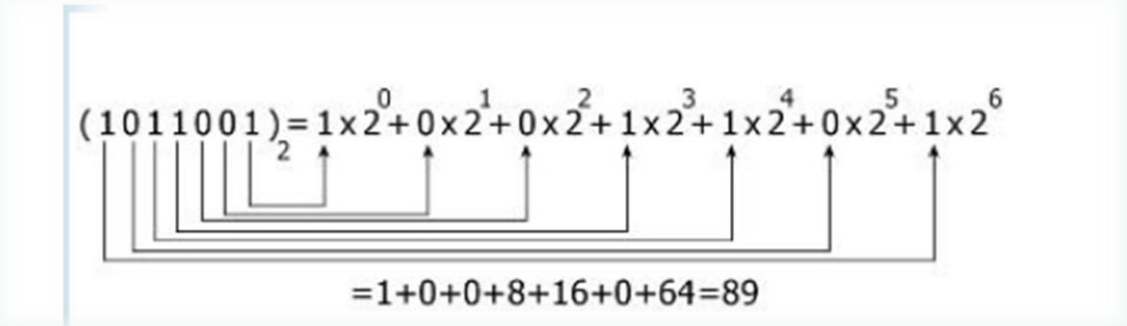
$$1\times2^4+0\times2^3+1\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}=$$

$$=1\times16+0\times8+1\times4+1\times2+1\times1+0\times0.5+1\times0.25=23.25$$

❖ آشنایی با سیستم اعداد(سیستم دودویي)

❑ تبدیل مبناي ۲ به مبناي ۱۰

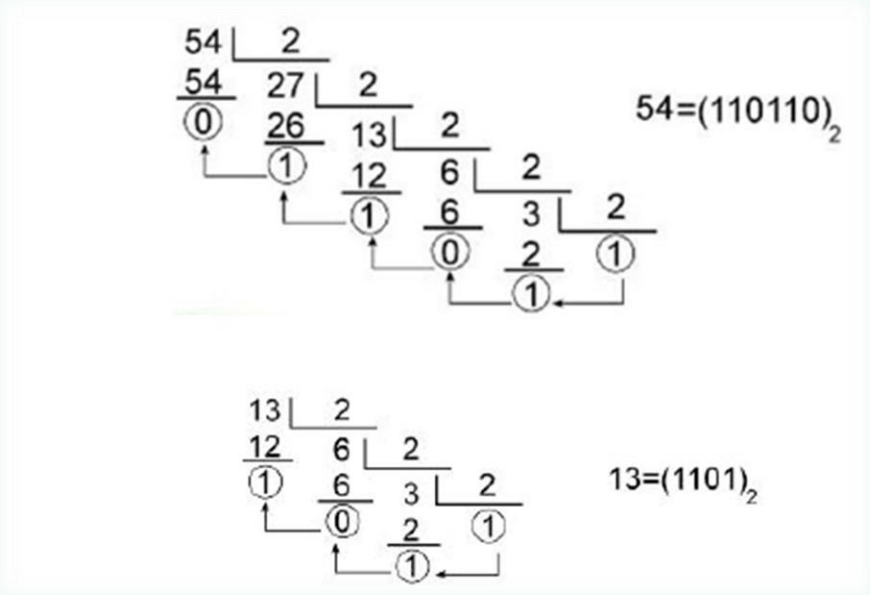
مثال دیگری از تبدیل مبنای ۲ به مبنای ۱۰:



❖ آشنایی با سیستم اعداد (سیستم دودویی)

❖ تبدیل مبنای ۱۰ به مبنای ۲ (معکوس)

برای عدد ٥٤ و ۱۳:



❖ آشنایی با سیستم اعداد(مبنای ۸)

❑ تبدیل مبنای ۸ به مبنای ۱۰

اعداد: ۰ تا۷ پایه: ۸

به عنوان مثال تبدیل مبنای ۸ به مبنای ۱۰ برای عدد ۲۳٦,٤ :

$$(236.4)_8 = (158.5)_{10}$$

روش حل:

$$2\times8^2 + 3\times8^1 + 6\times8^0 + 4\times8^{-1} = 158.5$$

❖ **توجه :** تبدیل مستقیم بنای ۱۰ به مبنای ۸ (معکوس) نداریم!

❖ برای اینکار میتوان عدد را به مبنای ۲ ببریم وبعد به ۸ تبدیل کنیم....

❖ آشنایی با سیستم اعداد(مبنای ۱۶)

❑ تبدیل مبنای ۱۶ به مبنای ۱۰

اعداد: ۰ تا۹ و **F, E,D,C,B,A** به ترتیب برای نمایش رقمهای ۱۰و۱۱و۱۲و۱۳و۱٤و۱۵

پایه: 16

به عنوان مثال تبدیل مبنای ۱٦ به مبنای ۱۰ برای عدد **D63FA**:

$$(D63FA)_{16} = (877562)_{10}$$

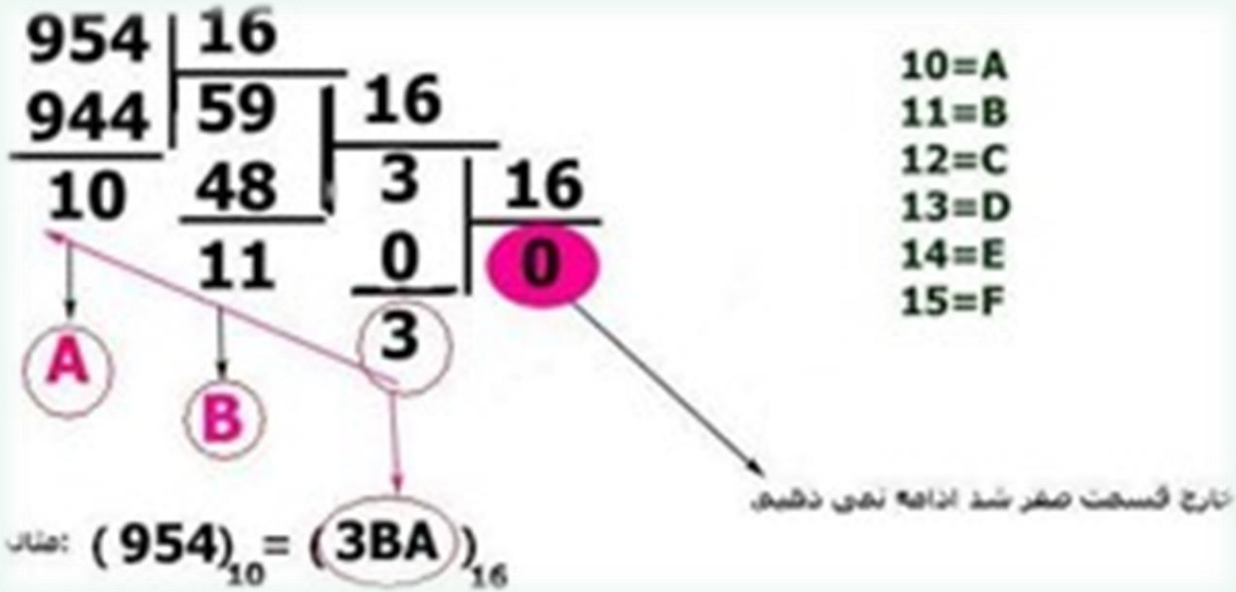
روش حل:

$$13\times16^4 + 6\times16^3 + 3\times16^2 + 15\times16^1 + 10\times16^0 = 877562$$

❖ آشنایی با سیستم اعداد(مبنای ۱۶)

❖ تبدیل مبنای ۱۰ به مبنای ۱۶(معکوس)

مثال) برای عدد ۹۵٤ :



❖ جمع دودویی

1 + 1 = 10	1 + 0 = 1	0 + 1 = 1	0 + 0 = 0

در جمع آخری ۱ رقم نقلی است که با بیت های بعدی جمع می شود.

مثال) جمع عدد ۲۹+۱۷= ٤٦ با سیستم دودویی:

عدد ۲۹ در مبنای ۲ میشود: ۰۰۰۱۱۱۰۱ عدد ۱۷ در مبنای ۲ میشود: ۰۰۰۱۰۰۰۱

روش حل:

$$\begin{array}{r} {\scriptstyle 1\quad 1} \\ 00011101 \\ +\ 00010001 \\ \hline (00101110)_2 \end{array}$$

امتحان نتیجه:

$$(00101110)_2 = 2+4+8+32 = 46$$

❖ فصل دوم : الگوریتم ها و فلوچارت

هدفهای کلی:

✓شناخت حل مسئله و ارائه الگوریتم

✓شناخت اجزای لازم برای حل مساله

✓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 بر می‌گزیند. 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 مثلا 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 می نامند.

❖ تعریف الگوریتم:

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

- دقیق باشد
- جزئیات کامل حل مسئله را داشته باشد.
- پایان‌پذیر 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 می کنند:

- ❑ *خواندن داده‌ها*
- ❑ *انجام محاسبات*
- ❑ *خروجی‌ها*

❖ مثال ۱ :

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

خروجی‌ها	انجام محاسبات	ورودیها
مجموع دو عدد	جمع دو عدد	a , b

❖ **الگوریتم:**

۱) شروع
۲) a, b را بخوان.
۳) مجموع a , b را محاسبه و در sum قرار بده.
۴) sum را در خروجی چاپ کن
۵) پایان

❖ مثال ۲ :

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

ورودیها	انجام محاسبات	خروجی‌ها
a, b, c	محاسبه ی مجموع ومیانگین	چاپ مجموع و میانگین

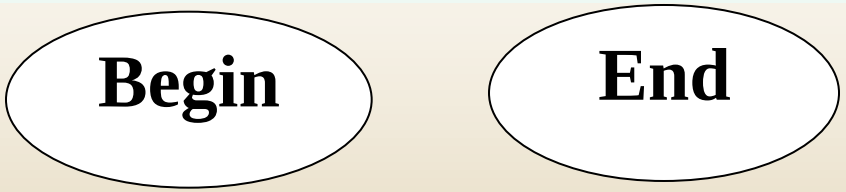
❖ الگوریتم:

۱) ۰ـ شروع
۲) ۱ـ سه عدد از ورودی بخوان
۳) ۲ـ مجموع سه عدد را محاسبه و در sum قرار بده.
۴) ۳ـ sum را بر سه تقسیم کرده،در ave قرار بده.
۵) ٤ـ ave , sum را در خروجی چاپ کن.
۶) ۵ـ پایان.

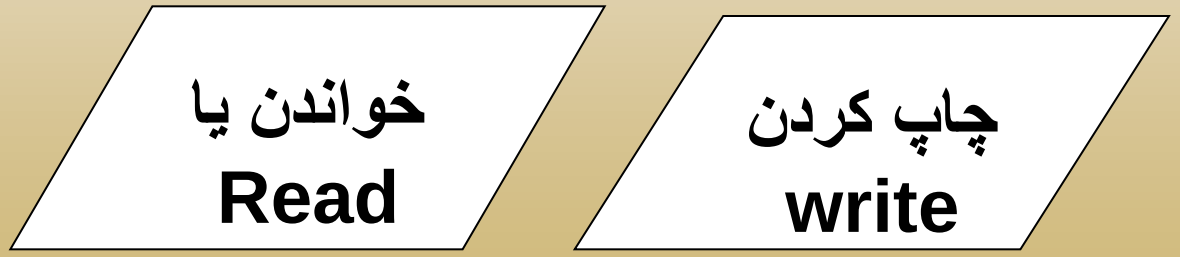
❖ فلوچارت

معمولا درك يك الگوريتم با شكل راحتر از نوشتن آن بصورت متن می‌باشد. لذا الگوريتم را با فلوچارت (Flowchart) نمايش می‌دهند.
فلوچارت از شكل‌های زير تشكيل می‌شود

۱- علامت‌های شروع و پايان :كه معمولا از يك بيضی استفاده می‌كنند:



۲- علامتهای ورودی و خروجی: كه معمولا از متوازی‌الاضلاع استفاده ميشود:

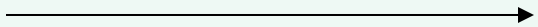


۳- علامتهای محاسباتی و جایگزینی: برای نمایش دستورات جایگزینی و محاسباتی از مستطیل استفاده می کنند:

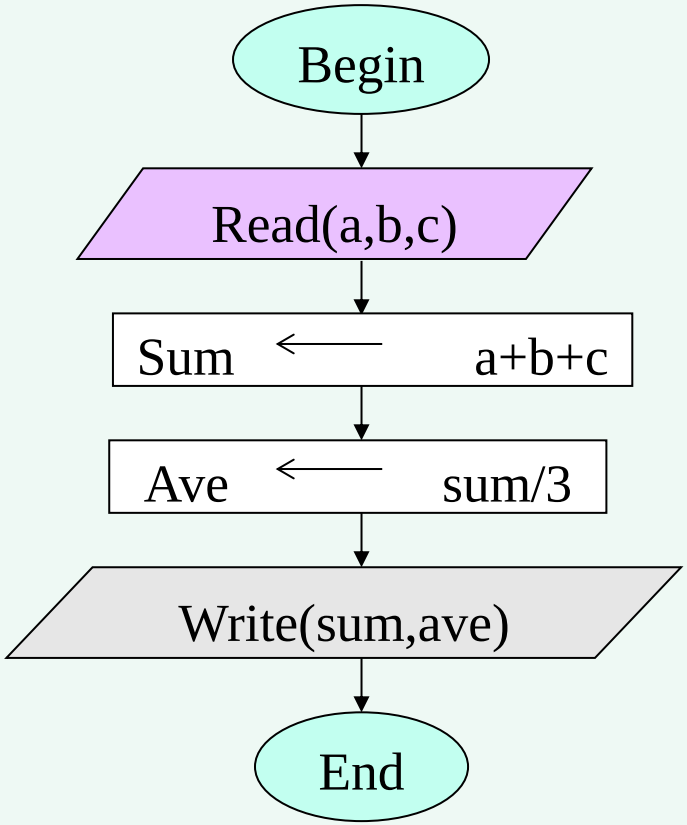
جایگزینی یا
محاسبات

۴- علامت شرط : برای نمایش شرط از لوزی استفاده می‌شود.

۵- علامت اتصال: برای اتصال شکل‌های مختلف بهم از فلش‌های جهت‌دار استفاده می‌کنند.



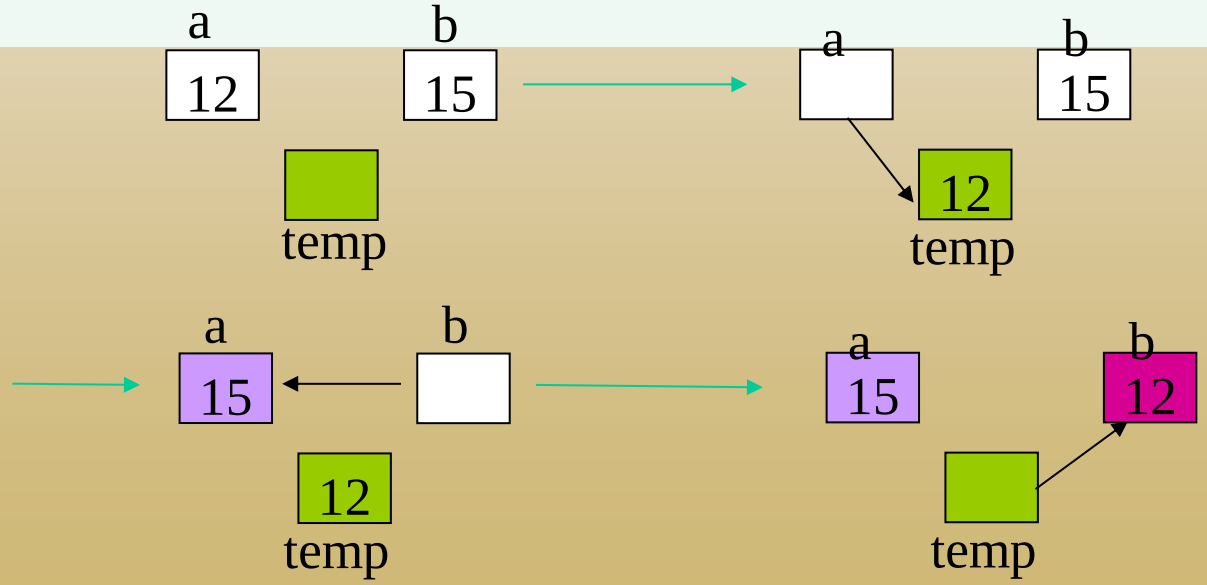
❖ مثال ۱) فلوچارت مجموع سه عدد:



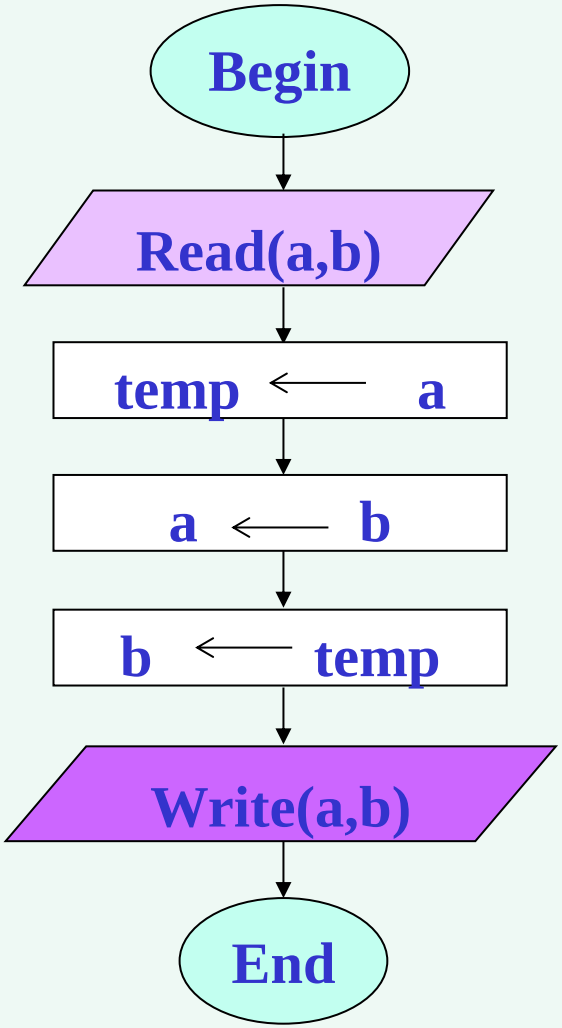
❖ مثال ۲)

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

روش حل: برای حل این مسئله a , b را دو متغیر که در آنها دو عدد خوانده شده، قرار مي‌گيرند در نظر مي‌گيريم.
سپس با استفاده از يك متغير كمكی محتويات اين دو عدد را جابجا می‌كنيم:



فلوچارت مسئله بالا بصورت زیر خواهد بود:



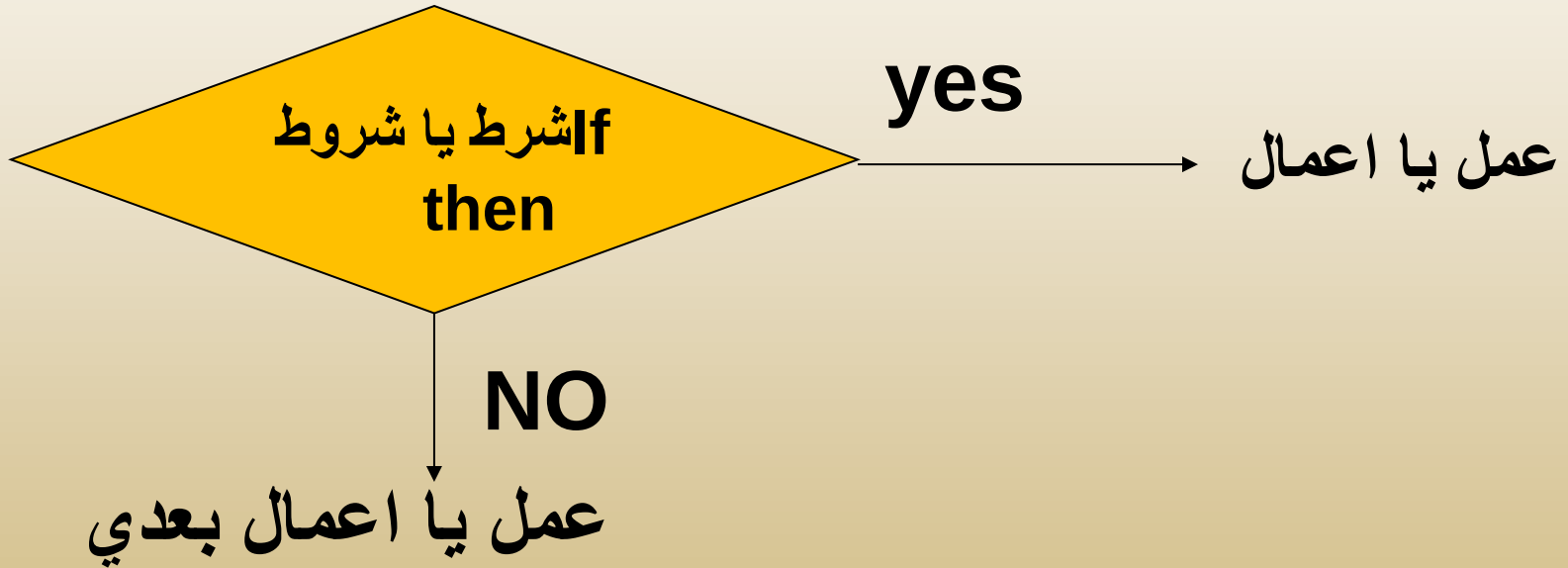
❖ دستورالعمل‌های شرطی

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.

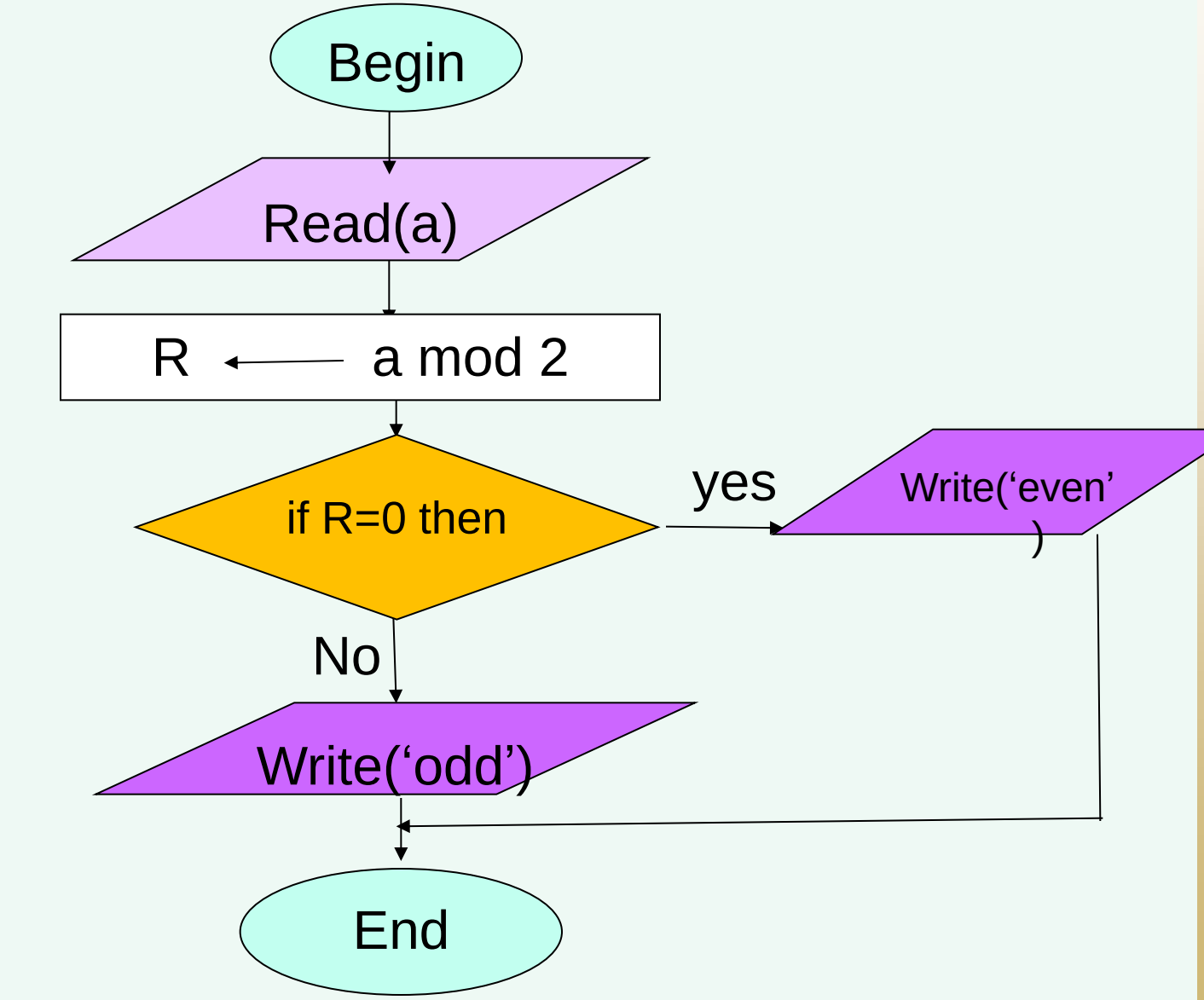
بطور مثال اگر هوا ابری باشد ممکن است چنین سخن بگوییم:

اگر هوا بارانی باشد سپس چتری برمی‌دارم.
در غیر اینصورت چتر برنمی‌دارم.

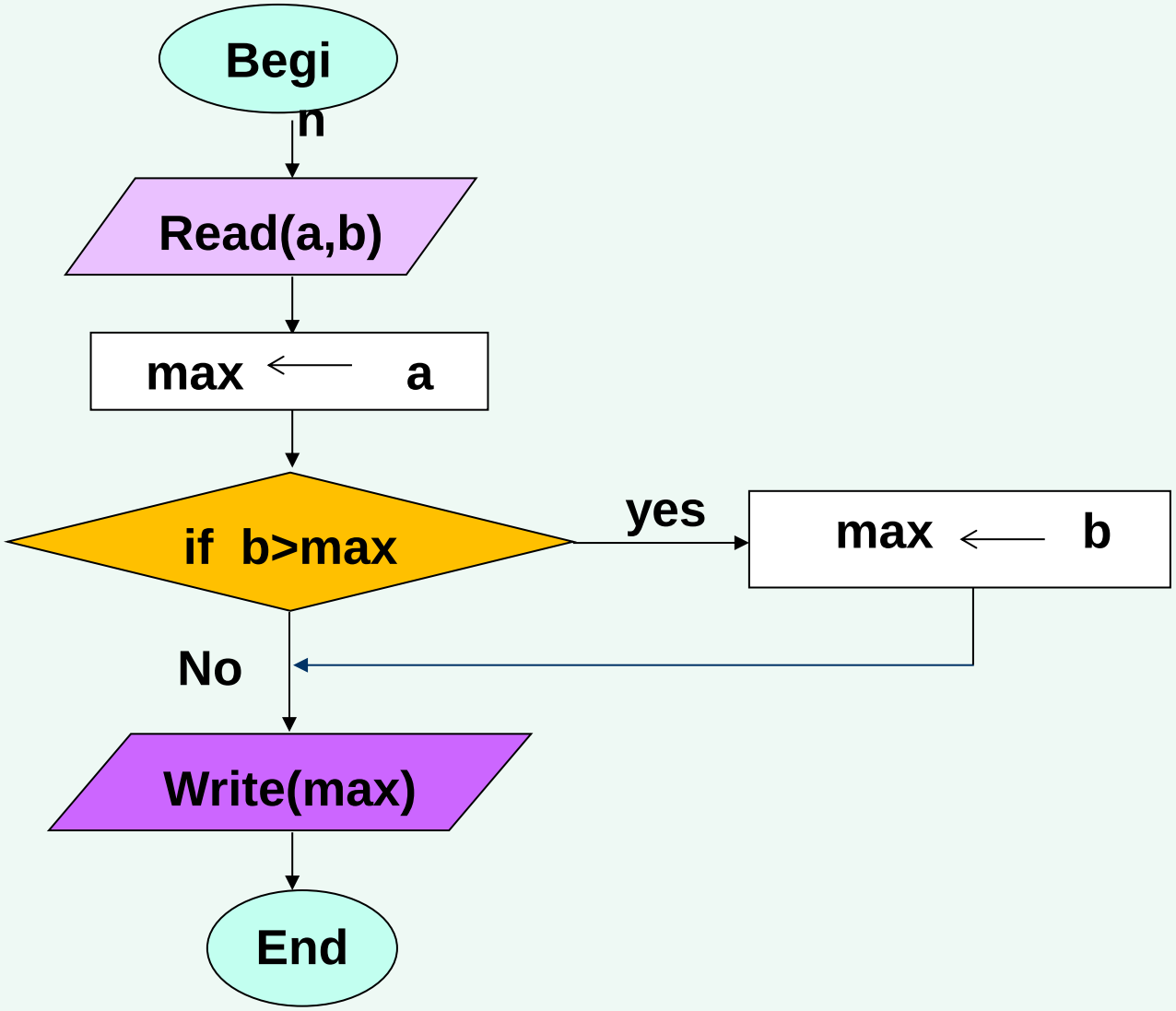
در حالت کلی شرط را بصورت زیر نمایش می‌دهند:



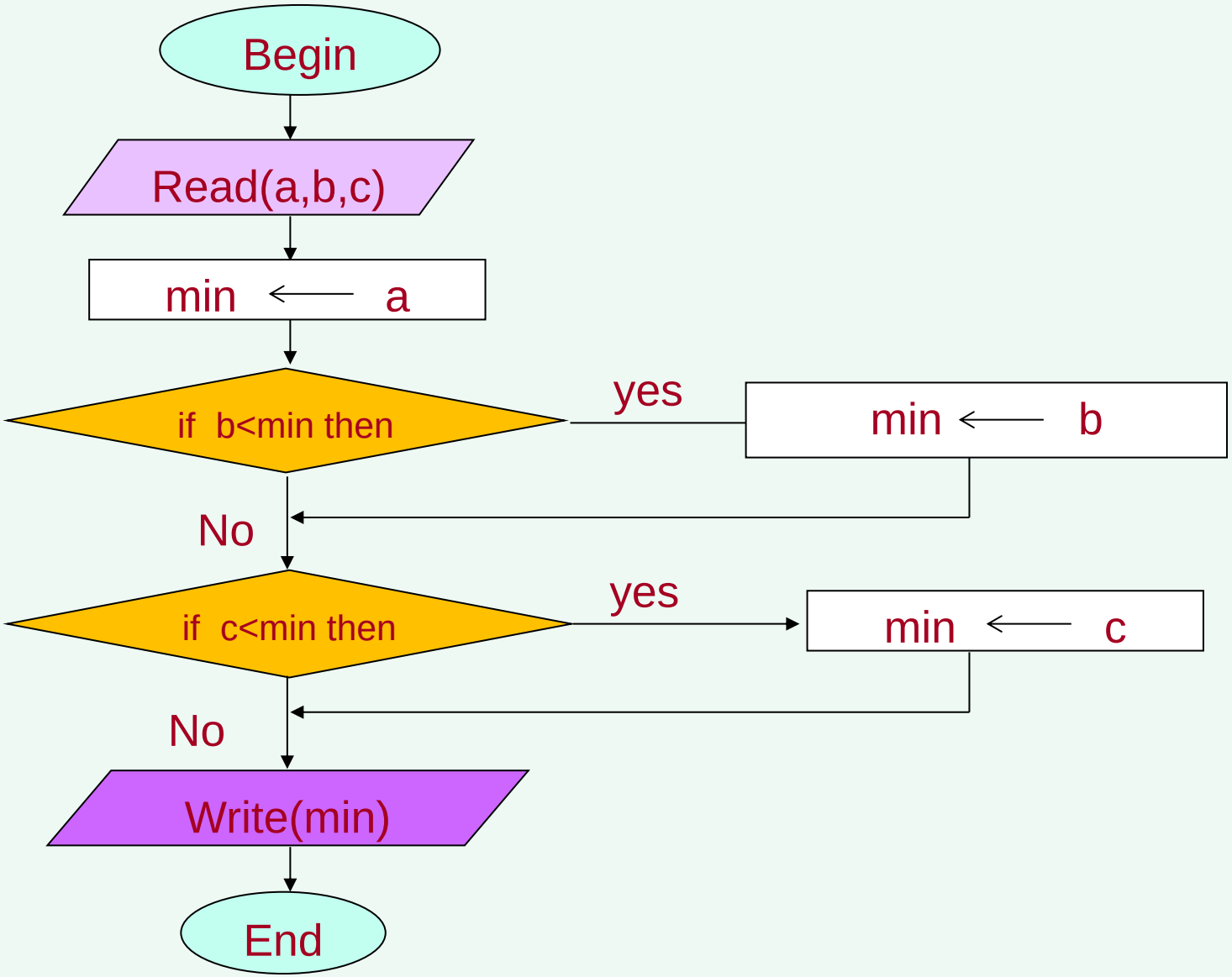
❖ مثال۱) فلوچارتی رسم نمائید که عددی را از ورودی دریافت کرده، فرد یا زوج بودن آن را تشخیص دهد.



مثال۲) فلوچارتی رسم کنید که دو عدد از ورودی دریافت کرده بزرگترین عدد را پیدا کرده در خروجی چاپ نماید.



مثال۳) فلوچارتی رسم نمائید که سه عدد از ورودی دریافت کرده، کوچکترین عدد را یافته در خروجی چاپ نماید:



مثال۳) نمونه اجرای فلوچارت بالا بصورت زیر مي‌باشد:

	a	b	c	Min	خروجی
1	12	11	17		11
2				12	
3				11	

❖ حلقه‌ها

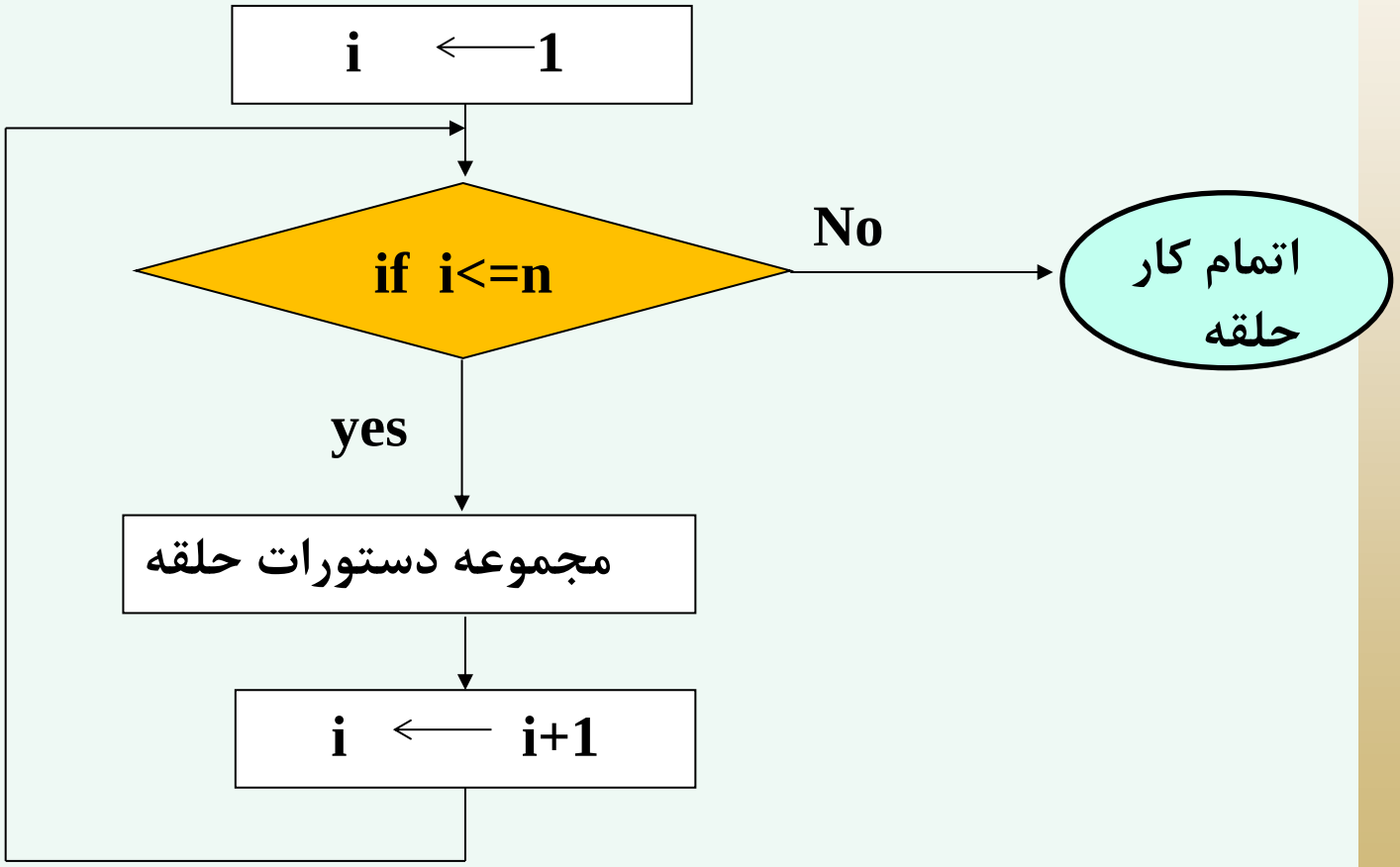
در حل بسیاری از مسائل با عملیاتی روبرو می شویم ، که نیاز به تکرار دارند و عمل تکرار آنها به تعداد مشخصی انجام می گیرد.

فرض کنید، بخواهیم میانگین ۱۰۰ عدد را محاسبه کنیم، در اینصورت منطقی بنظر نمی رسد که ۱۰۰ متغیر مختلف را از ورودی دریافت کنیم سپس آنها را جمع کنیم.

اجراي حلقه ها يا به تعداد تكرار مشخص می‌باشد يا تا زمان درست بودن يك شرط ادامه پيدا می كند.
يك حلقه معمولا شامل موارد زير می باشد:

۱ـ انديس حلقه
۲ـ مقدار اوليه برای انديس حلقه
۳ـ مقدار افزاينده براي انديس حلقه (معمولا يك واحد در هر مرحله)
۴ـ مقدار نهايی (تعداد تكرار حلقه)
۵ـ شرطي برای كنترل تعداد تكرار حلقه

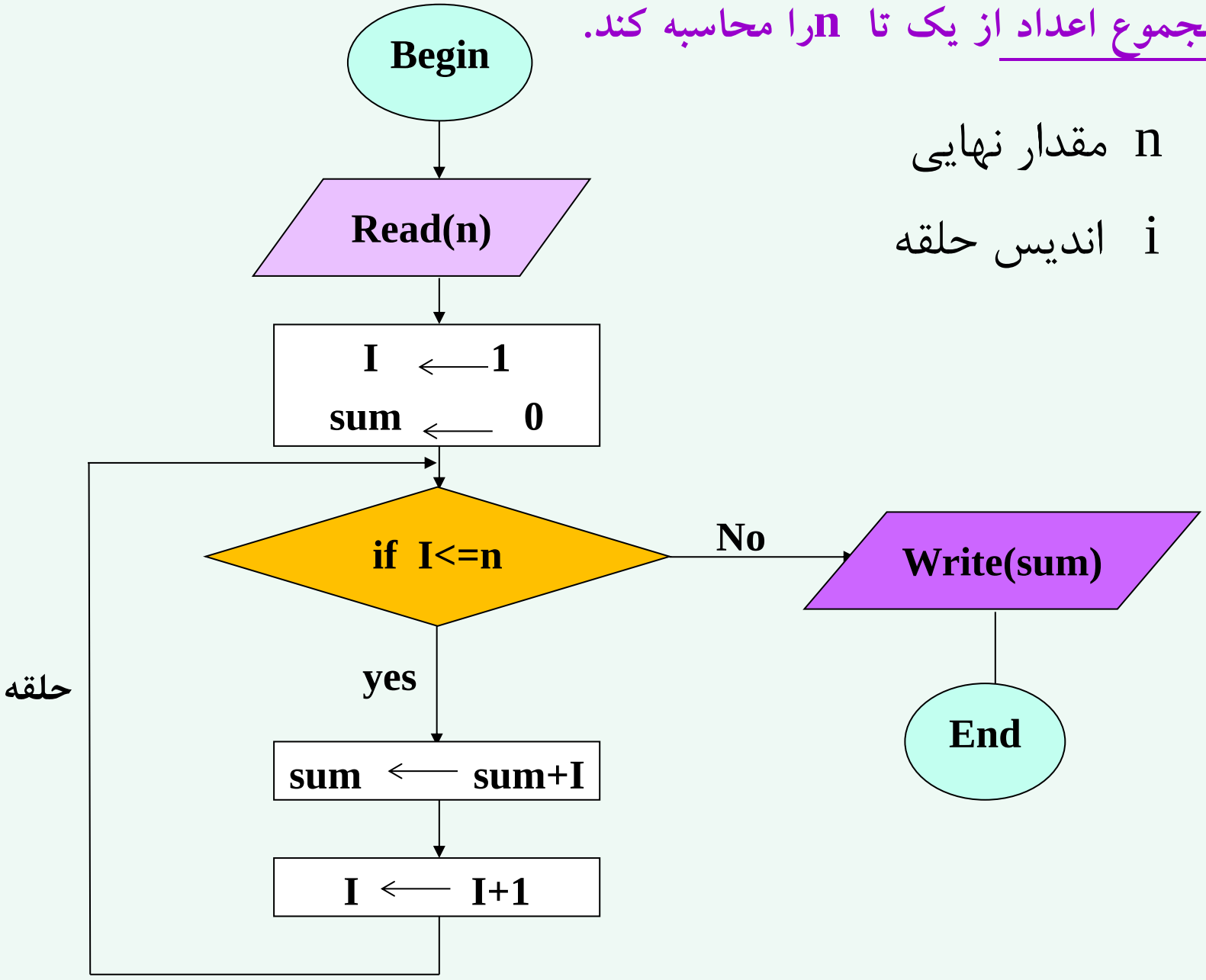
حلقه‌ها را غالباً با فلوچارت بصورت زیر نمایش می دهند:



مثال : فلوچارتی رسم نمائید که عدد nرا از ورودی دریافت کرده، مجموع اعداد از یک تا nرا محاسبه کند.

n مقدار نهایی

i اندیس حلقه



نمونه اجرای فلوچارت بالا بصورت زیر است:

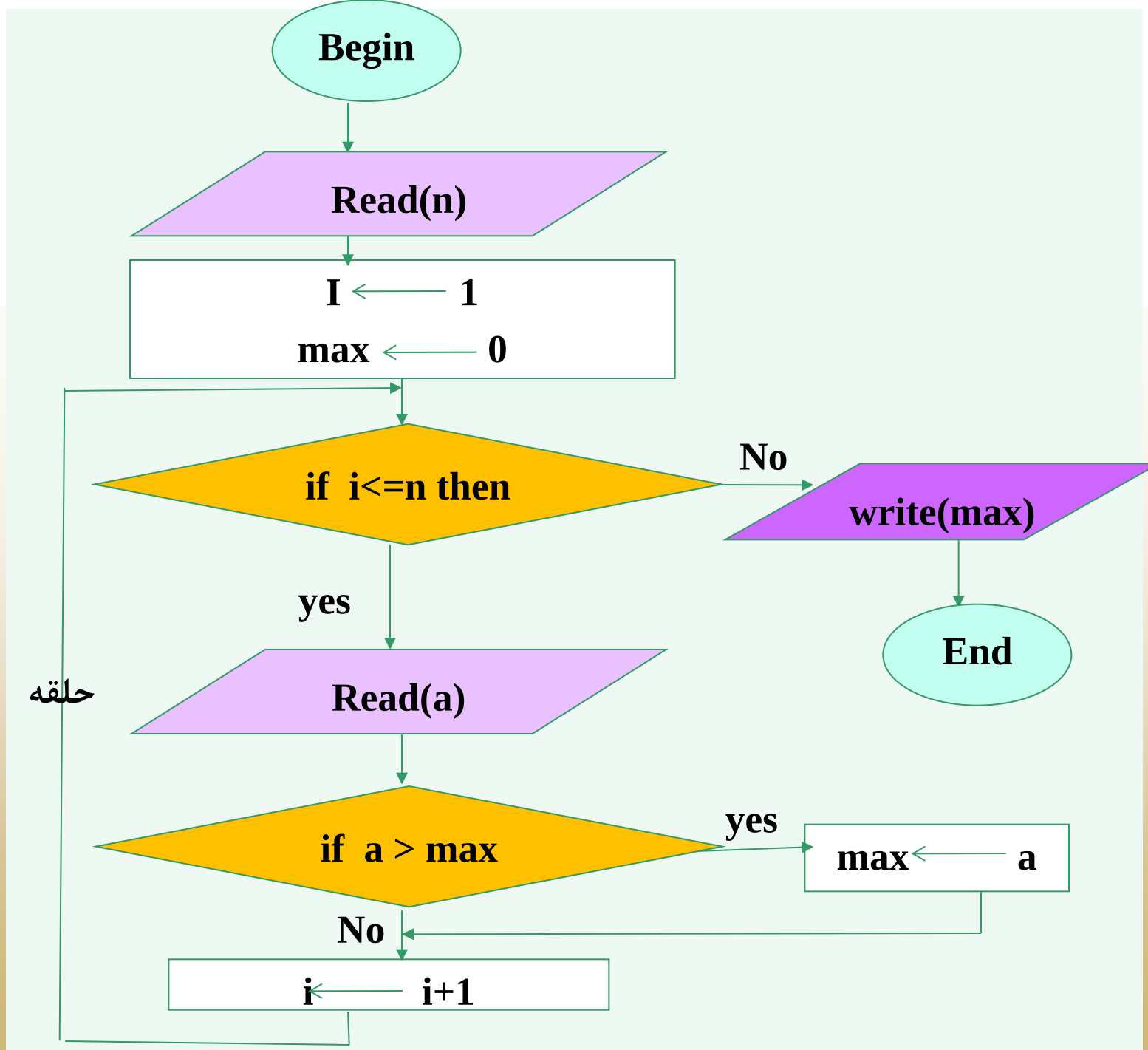
	N	I	sum	خروجی
1	5	1	0	15
2		2	1	
3		3	3	
4		4	6	
5		5	10	
6		6	15	
7				

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

n مقدار نهایی

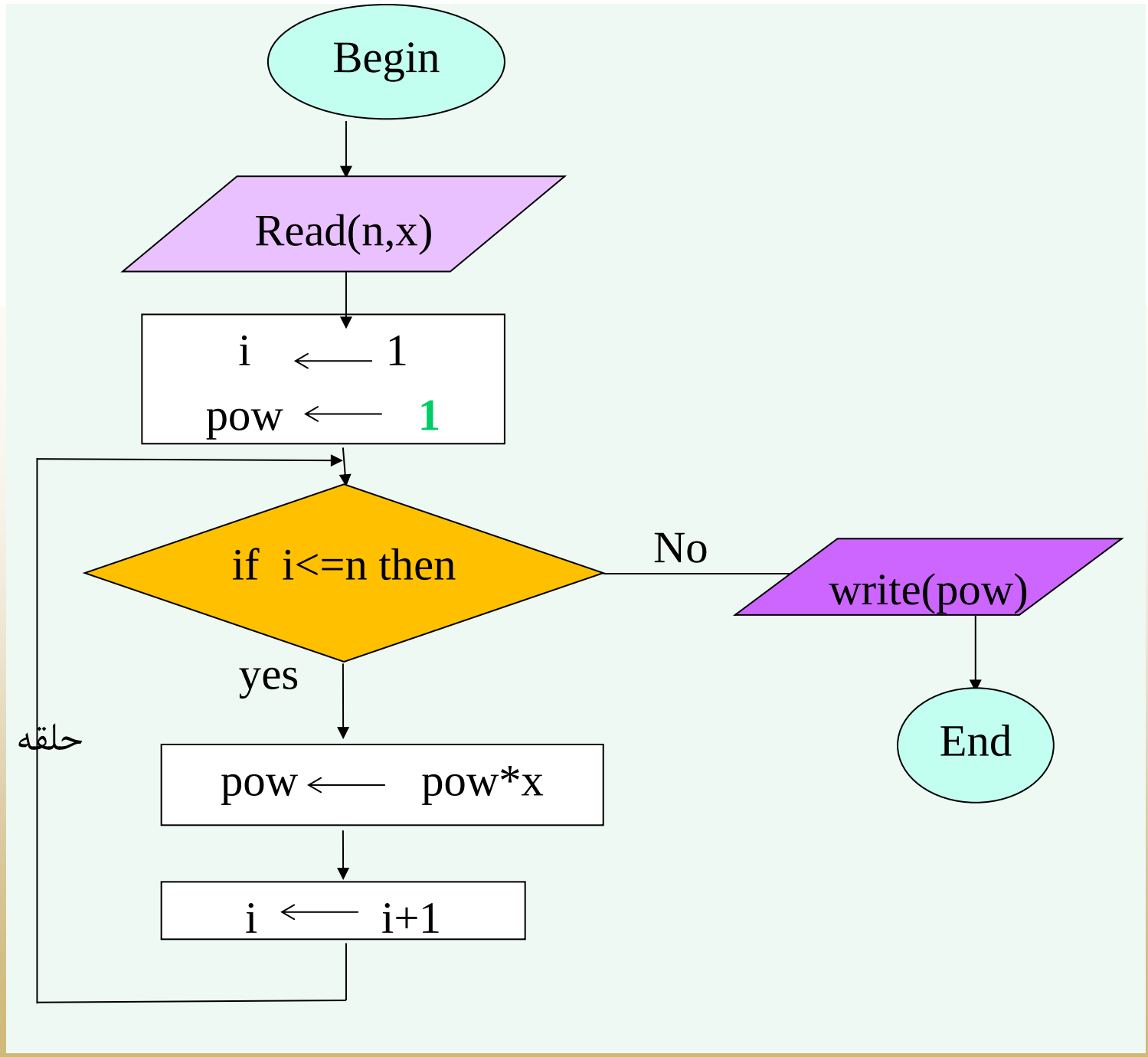
i اندیس حلقه

Max بزرگترین مقدار


Begin
Read(n)
I ← 1
max ← 0
if i<=n then
No
write(max)
End
yes
Read(a)
حلقه
if a > max
yes
max ← a
No
i ← i+1

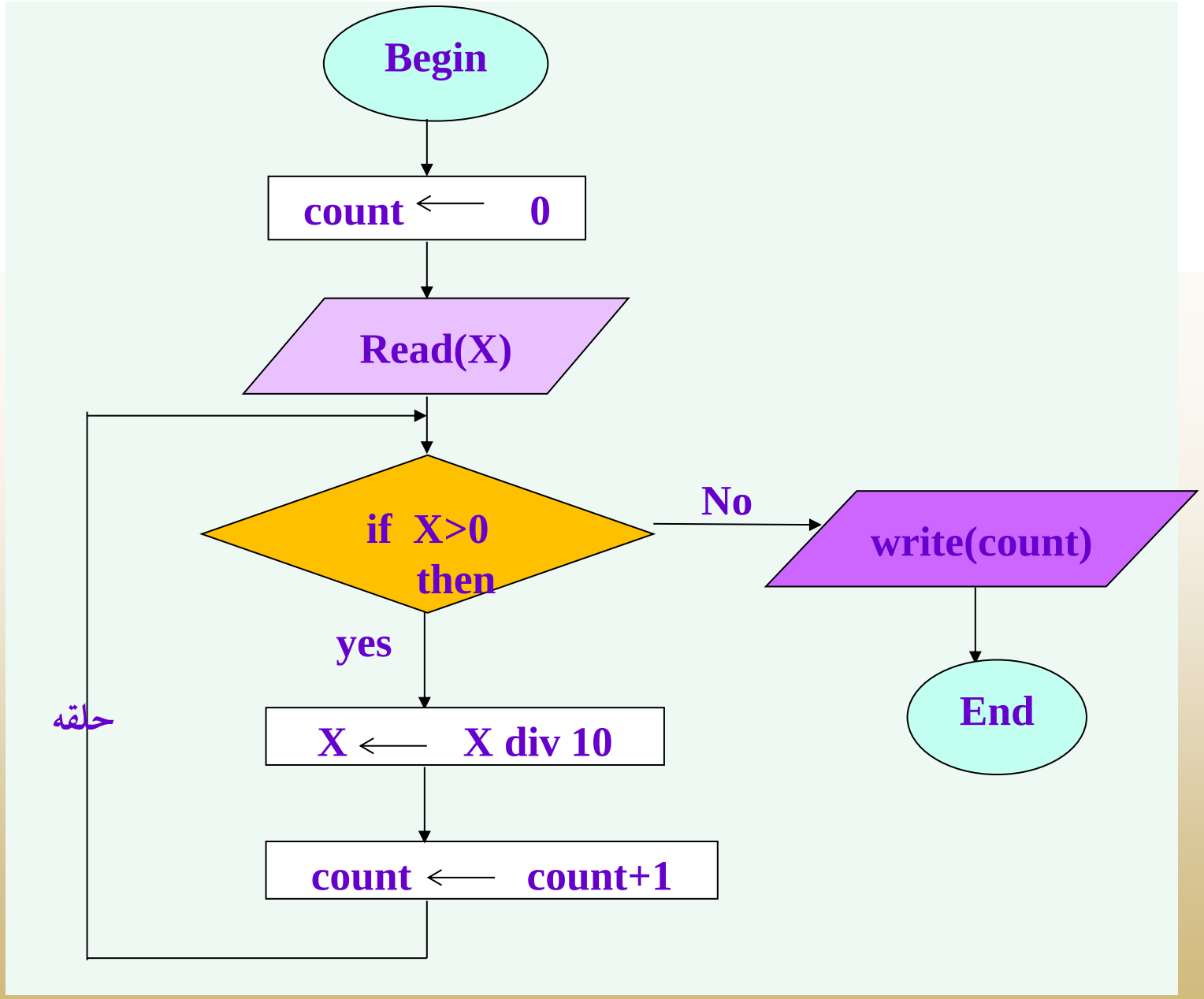
مثال : فلوچارتی رسم نمائید که x , n، دو عدد صحیح مثبت را از ورودی دریافت کرده سپس xبه توان nرا محاسبه کند.

اندیس حلقه	i
مقدار نهايي	n
عدد به توان n	
pow	


Begin
Read(n,x)
i ← 1
pow ← 1
if i<=n then
No
write(pow)
End
yes
حلقه
pow ← pow*x
i ← i+1

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

عدد خوانده شده	X
تعداد ارقام	$count$


Begin
count ← 0
Read(X)
if X>0 then
No
write(count)
End
yes
X ← X div 10
count ← count+1
حلقه

❖ حلقه‌هاي تودرتو

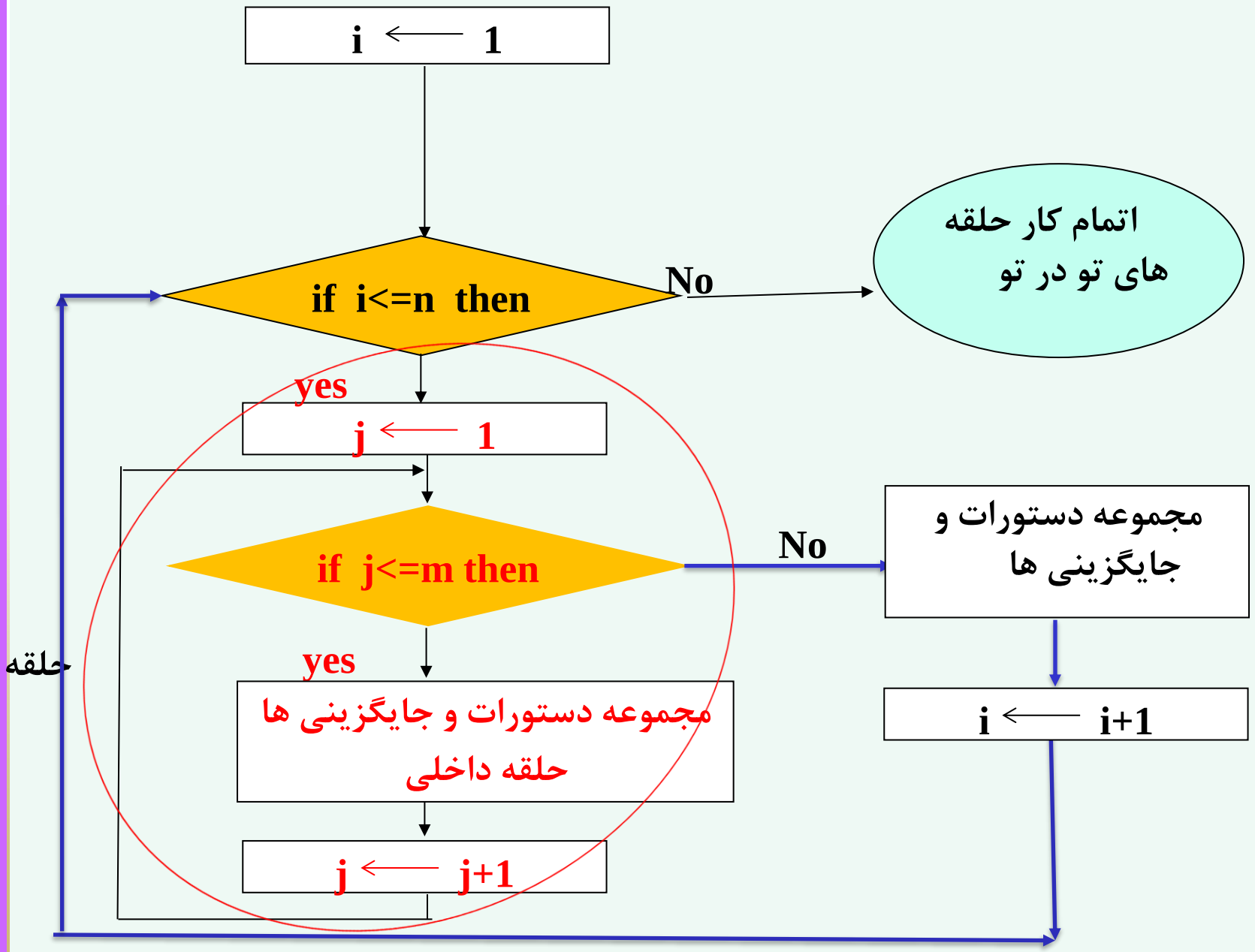
الگوریتم‌هایی که تا حال بکار بردیم، فقط شامل یک حلقه بودند.
در صورتی که در بسیاری از مسائل ممکن است نیاز به استفاده از چند حلقه در داخل هم باشیم. در این نوع حلقه‌ها باید دقت بیشتری به خرج دهیم، تا مشکلی پیش نیاید. در این حلقه ها معمولا برای هر حلقه شرط نهایی و اندیس اولیه جداگانه باید تعریف کنیم .

در حلقه‌های تودرتو 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 اینصورت کل حلقه :

$$n \times m$$

بار تکرار خواهد شد.

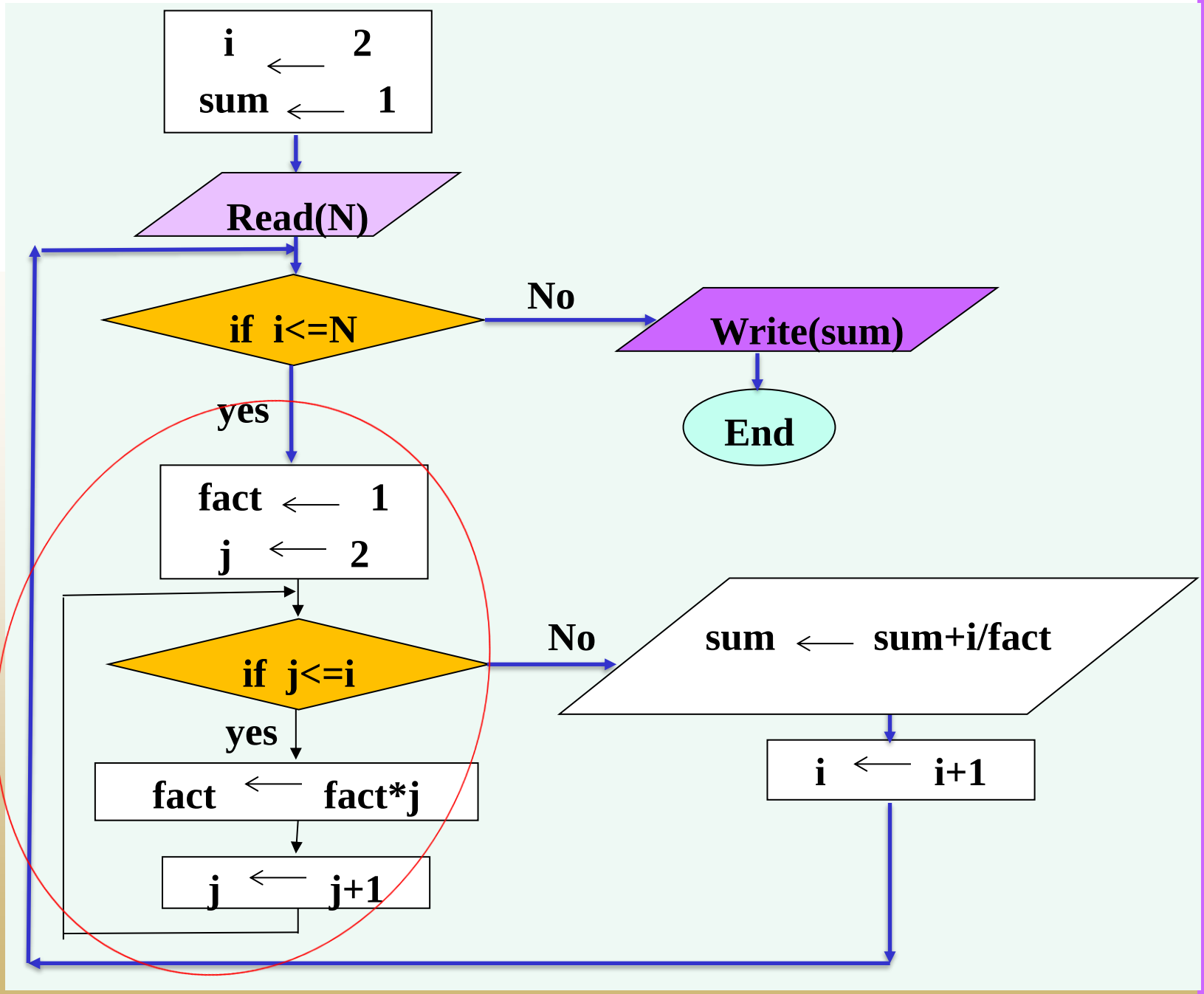
❖ فلوچارت حلقه‌های تودرتو را می‌توان بصورت زیر نشان داد:



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

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

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


i ← 2
sum ← 1
Read(N)
if i<=N
No
Write(sum)
End
yes
fact ← 1
j ← 2
if j<=i
No
sum ← sum+i/fact
yes
fact ← fact*j
i ← i+1
j ← j+1

Δ تمرین اول:

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

۲ـ فلوچارتی رسم نمائید که دو عدد از ورودی دریافت کرده، سپس محتویات دو عدد را بدون استفاده از متغیر کمکی جابجا کند.

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

٤-فلوچارتی رسم نمائید که عددی (درجه حرارت برحسب سانتیگراد) را از ورودی دریافت کرده سپس آن را به درجه فارنهایت تبدیل کند.

$$f = \frac{9}{5}c + 32$$

❖ فصل سوم : مقدمات زبان C++

معرفی زبان برنامه نویسی C++

- این زبان در اوائل دهه ۱۹۸۰ در آزمایشگاه بل طراحي شده است.
- این زبان عملاً توسعه یافته زبان برنامه نویسي C می باشد که امکان نوشتن برنامه‌هاي ساخت یافته شئ گرا را مي‌دهد.
- این زبان قادر است در کلیه ی پلتفرم های موجود مثل مکینتاش، توزیع های لینوکس و همچنین ویندوز، نصب و اجرا شود.
- زبان C ++ به عنوان یک زبان کامپایل شده، چند منظوره شناخته می شود که به حروف بزرگ و کوچک هنگام برنامه نویسی حساس است.

C ++ را یک زبان سطح متوسط معرفی کردیم چون به صورت ترکیبی از زبان سطح پایین و سطح بالا است.

- C ++ بعد از زبان C شکل گرفت که آن را می توان مجموعه ای تکمیل شده از زبان C دانست چون هر برنامه ای که با زبان C برنامه نویسی شده است را هم با زبان C ++ می توان کدنویسی کرد.

در C++چهار نوع کد داریم:

- شناسه ها
- کلمات کلیدی
- انواع داده ها
- عملگر ها

Δ شناسه ها:

شناسه‌ها نامی است که به یک قسمت از برنامه مانند متغیر تابع ثابت و یا ... داده می‌شود در زبان C++ برای انتخاب شناسه‌ها می‌توان از علائم زیر استفاده کرد.

-حروف انگلیسی کوچک و بزرگ(a...z A...z)

- ارقام(9,.....,۰,۱)

-علامت خط پایین یا _

البته یک شناسه نمی تواند با یک رقم شروع شود. مثلا شناسه های number ، number1 number_one مجاز هستند اما شناسه های 1number، number one مجاز نیستند. البته در برنامه نویسی امروزی پیشنهاد می شود بجای شناسه هایی همانند number_one از numberOne استفاده گردد. یعنی بجای اینکه در شناسه ها از _ برای جداکننده استفاده کنیم، اولین حرف هر قسمت را بصورت بزرگ بنویسیم.

نکته مهم دیگری که باید به آن اشاره کرد آنست که زبان C++ برخلاف بسیاری از زبانهای دیگر به کوچک و بزرگی حروف حساس است .(case sensitive) در نتیجه شناسه های زیر با یکدیگر متفاوتند:

Number1 ≠ number1

این مسئله معمولاً در هنگام برنامه نویسی باعث ایجاد بعضی خطاها می‌شود.

نکته: زبان C++هیچ محدودیتی در طول شناسه ها قرارنداده است.

نکته: بهتر است از شناسه‌های با معنی در برنامه‌هایمان استفاده کنیم.

نکته بهتر است از اختصار استفاده نکنیم تا درک عملکرد برنامه برای دیگران قابل فهم باشد.

نکته: درهنگام انتخاب شناسه نمی‌توانید از کلمات کلیدی که برای منظورهای خاص در زبان C++رزرو شده‌اند استفاده کنید.

زبان C++دارای کلمات کلیدی زیادی است که در زیر برخی از آنها را نشان می‌دهیم.

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while	and	and_eq asm	bitand bitor
bool	catch	class	compl	const_cast
delete	dynamic_cast	explicit	export	false
friend	inline	mutable	namespace	new
not	not_eq	operator	or	or_eq
private	protected	public	reinterpret_cast	static_cast
template	this	throw	true	try
typeid	typename	using	virtual	wchar_t
xor	xor_eq			

❖ انواع داده ها:

یک متغیر، مکانی از حافظه است که یک مقدار می‌تواند در آن ذخیره شود هر متغیر پیش از آنکه در یک برنامه استفاده گردد ابتدا باید اعلان گردد اعلان یک متغیر بدین معنی است که نوع آن متغیر را مشخص شود در زیر برخی نوع‌های مورد استفاده درسی C++را نشان می‌دهیم

نوع داده	توضیح	اندازه(بیت)	دامنه
char	کاراکتر	۸	-۱۲۸ تا ۱۲۷
int	عدد صحیح	۱۶	-32768 تا ۳۲۷۶۷
float	عدد اعشاری	۳۲	3.4e-38 تا 3.4e+38
double	عدد اعشاری با دقت مضاعف	۶۴	1.7e-308تا1.7e308

برای تعریف متغیرها به شکل روبروعمل میکنیم : <type><variable-list>;

که type یکی از نوع داده‌های گفته شده و variable-list لیستی از متغیرهاست که با کاما از یکدیگر جدا شده‌اند به عنوان مثال:

```
int number;
float average;
double a,b,c;
```

علاوه بر این می‌توان در هنگام تعریف متغیر به آن مقدار اولیه نیز داد مثال:

```
int d=0;
```

چند مثال از اعلان متغیر ها :

- ✓ براي اعلان متغيرx از نوع int :

```
int   x;
```

- ✓ برای اعلان متغیرهای p و q از نوع float که هر کدام چهار بایت از حافظه را اشغال می‌کنند :

```
float  p , q  ;
```

- ✓ برای اعلان متغیر next از نوع کراکتر که می‌توان یکی از ۲۵۶ کرکتر را به آن تخصیص داد و يك بايت را اشغال می کند.

```
char next ;
```

تخصیص مقادیر به متغیر ها :

با استفاده از عملگر = می توان به متغیرها مقدار اولیه تخصیص نمود

- ✓ در دستورالعمل `int x=26;`
- ✓ X را از نوع int با مقدار اولیه 26 اعلان نموده .

- ✓ در دستورالعمل `long a=67000 , b=260;`

متغیرهاي b و a را از نوع long int تعریف نموده با مقادیر بترتیب 260 و 67000.

داده‌های از نوع کراکتر :

- ✓ برای نمایش داده‌های از نوع char در حافظه کامپیوتر از جدول ASCII استفاده می شود. جدول اسکی به هر یك از ۲۵۶ کراکتر یك عدد منحصر بفرد بین ۰ تا ۲۵۵ تخصیص می‌دهد.

کراکترهای مخصوص:

- ✓ کامپیلر C++ بعضی از کراکترهای مخصوص که در برنامه می‌توان از آنها برای فرمت بندی استفاده کرد را تشخیص می‌دهد. تعدادی از این کراکترهای مخصوص به همراه کاربرد آنها در اسلاید بعد آورده شده است.

کراکترهای(کنترلی) مخصوص :

\n	Newline خط جدید
\t	Tab
\b	Backspace
\a	Beep sound
\”	Double quote
\’	Single quote
\0	Null character
\?	Question mark
\\	Back slash

بعنوان مثال از کرکتر \a می‌توان برای ایجاد صدای beep استفاده نمود.

```
char        x = '\a' ;
```

رشته‌ها :

رشته یا string عبارتست از دنباله‌ای از کرکترها که بین " " قرار داده می‌شود. در حافظه کامپیوتر انتهای رشته‌ها بوسیله 0\ ختم می‌گردد.

مثال ۱:

"BOOK STORE" یک رشته ده کرکتری می‌باشد که با توجه به کرکتر 0\ که به انتهای آن در حافظه اضافه می شود جمعاً یازده بایت را اشغال می‌کند.

مثال ۲ :

دقت نمایید که "w" یک رشته می‌باشد که دو بایت از حافظه را اشغال می‌کند در حالیکه 'w' یک کرکتر می‌باشد که یک بایت از حافظه را اشغال می‌نماید.

نمایش مقادیر داده‌ها :

برای نمایش داده‌ها بر روی صفحه مانیتور از cout که بدنبال آن عملگر درج یعني >>قید شده باشد استفاده می‌گردد. بایستی توجه داشت که دوکرکتر < پشت سر هم توسط C++ بصورت يك كركتر تلقی می‌گردد

مثال :

✓ برای نمایش پیغام good morning بر روی صفحه نمایش :

```
cout << "good morning";
```

✓ برای نمایش مقدار متغیر x بر روی صفحه نمایش :

```
cout << x ;
```

دریافت مقادیر متغیرها:

به منظور دریافت مقادیر برای متغیرها در ضمن اجرای برنامه از صفحه کلید، از cin که بدنبال آن عملگر استخراج یعنی >> قید شده باشد می‌توان استفاده نمود.

مثال :

```
int x;
cout << "Enter a number:" ;
cin >> x;
```

عملگر انتساب :

عملگر انتساب = می‌باشد که باعث می‌گردد مقدار عبارت در طرف راست این عملگر ارزیابی شده و در متغیر طرف چپ آن قرار گیرد.

مثال :

```
x=a+b;
x=35 ;
x=y=z=26 ;
```

از عملگرهای انتساب چندگانه نیز می‌توان استفاده نمـود.
که مقدار سه متغیر z و y و x برابر با 26 میشود.

عملگرهای محاسباتی :

در C++ پنج عملگر محاسباتی وجود دارد که عبارتند از :

+	جمع
–	تفریق
*	ضرب
/	تقسیم
٪	باقیمانده

این عملگرها دو تائی می‌باشند زیرا روی دو عملوند عمل می نمایند. از طرف دیگر عملگرهای + و – رامی توان بعنوان عملگرهای یکتائی نیز در نظر گرفت.

مثال ۱ :

در حالتي كه هر دو عملوند عملگرهای % ، / ، * ، + ، – از نوع صحيح باشد نتيجه عمل از نوع صحيح می‌باشد.

عبارت	نتيجه
5 + 2	7
5 * 2	10
5 – 2	3
5 % 2	1
5 / 2	2

مثال ۲ :

در صورتیکه حداقل یکی از عملوندهای عملگرهای / ، * ، – ، + از نوع اعشاری باشد نتیجه عمل از نوع اعشاری می‌باشد.

عبارت	نتیجه
5. 0 +2	7. 0
5* 2. 0	10.0
5. 0 / 2	2.5
5. 0 -2	3. 0
5. 0 / 2. 0	2.5

عملگر های افزایش و کاهش :

در C++ ، افزایش یك واحد به مقدار یك متغیر از نوع صحیح را افزایش و بطور مشابه کاهش یك واحد از مقدار یك متغیر از نوع صحیح را کاهش می‌نامند..

عملگر های افزایش و کاهش :

عملگر کاهش را با – – و عملگر افزایش را با ++ نمایش می‌دهند. چون عملگرهای ++ و – – فقط روی یک عملوند اثر دارند این دو عملگر نیز جزء عملگرهای یکتائی می‌باشند.

عملگر + + : مقدار یک متغیر را یک واحد افزایش می دهد.

عملگر – – : مقدار یک متغیر را یک واحد کاهش می دهد.

مثال :

سه دستور العمل :

```
++x;
```

(اول X یک واحد افزایش پیدا میکند و بعد در محاسبات شرکت میکند)

```
x++;
```

(اول X در محاسبات شرکت میکند و بعد یک واحد اضافه میشود)

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

بطريق مشابه سه دستورالعمل زير نيز معادل مي باشند.

```
-  - y ;
y- - ;
y=y-1;
```

از عملگرهاي ++ و -- مي‌توان بدو صورت پیشوندی و پسوندی استفاده نمود.
در دستورالعمل‌های پیچیده عملگر پیشوندی قبل از انتساب ارزیابی میشود و عملگرپسوندی بعد از انتساب ارزیابی می‌شود.

مثال :

```
int x=5;
y=++x * 2;
```

پس از اجرای دستورالعملهای فوق :

y=12

```
int  x=5;
y=x++ * 2;
```

پس از اجراي دستورالعملهای فوق :

y=11

عملگر sizeof:

Sizeof ازعملگرهای یکتائی می باشد و مشخص کننده تعداد بایت هائی است که یك نوع داده اشغال می‌کند

مثال :

```
int   x;
cout << sizeof x ;
```

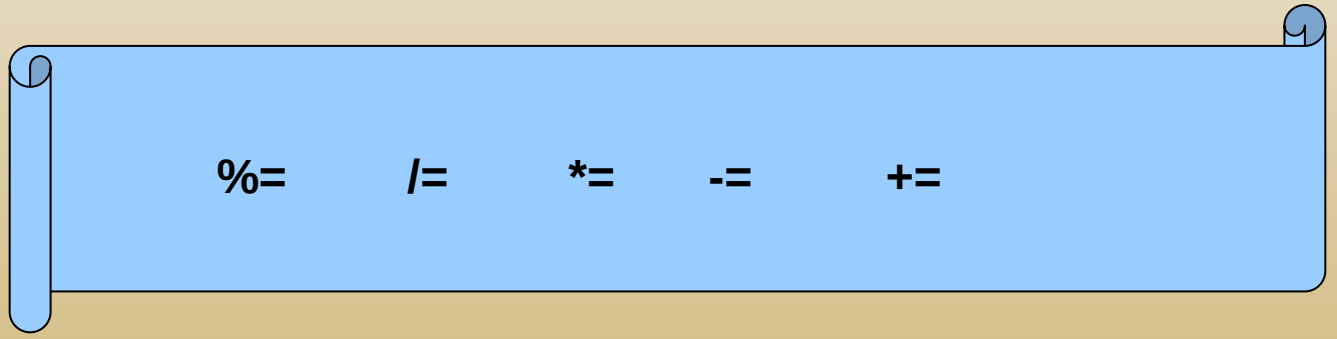
مقدار ۲ نمایش داده می شود

```
cout << sizeof(float) ;
```

مقدار ٤ نمایش داده می شود.

عملگرهای جایگزینی محاسباتی :

برای ساده‌تر نوشتن عبارتها در C++ ، می‌توان از عملگرهای جایگزینی محاسبـــاتی استفاده نمود.



الویت عملگرها :

ارزیابی مقدار یك عبارت ریاضی براساس جدول اولویت عملگرها انجام می‌گردد. در ذیل جدول اولویت عملگرها براساس به ترتیب از بیشترین اولویت به کمترین اولویت داده شده است.

چپ به راست	پرانتزها	( )
راست به چپ	عملگرهای یکتایی	sizeof ++ -- + -
چپ به راست	عملگرهای ضرب و تقسیم و باقیمانده	* / %
چپ به راست	عملگرهای جمع و تفریق	+ –
چپ به راست	عملگرهای درج و استخراج	>> <<
راست به چپ	عملگرهای جایگزینی و انتساب	= += -= *= /= %=

مثال ۱ :

(5+2) *(6+2*2)/2=

با توجه به جدول اولویت عملگرها داریم که :

7 *(6+2*2)/2

7*(6+4)/2

7* 10 /2

70 /2

35

توضیحات (Comments) :

توضیحات در برنامه باعث خوانائی بیشتر و درك بهتر برنامه میشود. بنابراین توصیه بر آن است که حتی الامکان در برنامه‌ها از توضیحات استفاده نمائیم. در C++، توضیحات به دو صورت انجام می‌گیرد که در اسلایدهای بعد به آن اشاره شده است.

الف: این نوع توضیح بوسیله // انجام می‌شود. که کامپیوتر هر چیزی را که بعد از // قرار داده شود تا انتهای آن خط اغماض مینماید.

مثال :

```
c=a+b;//c is equal to sum of a  and b
```

ب:توضیح نوع دوم با */ شروع شده و به /* ختم می‌شود و هر چیزی که بین* / و /* قرار گیرد اغماض می‌نماید .

مثال :

```
/ * this is a program
   to calcufate sum of
   n integer numbers */
```

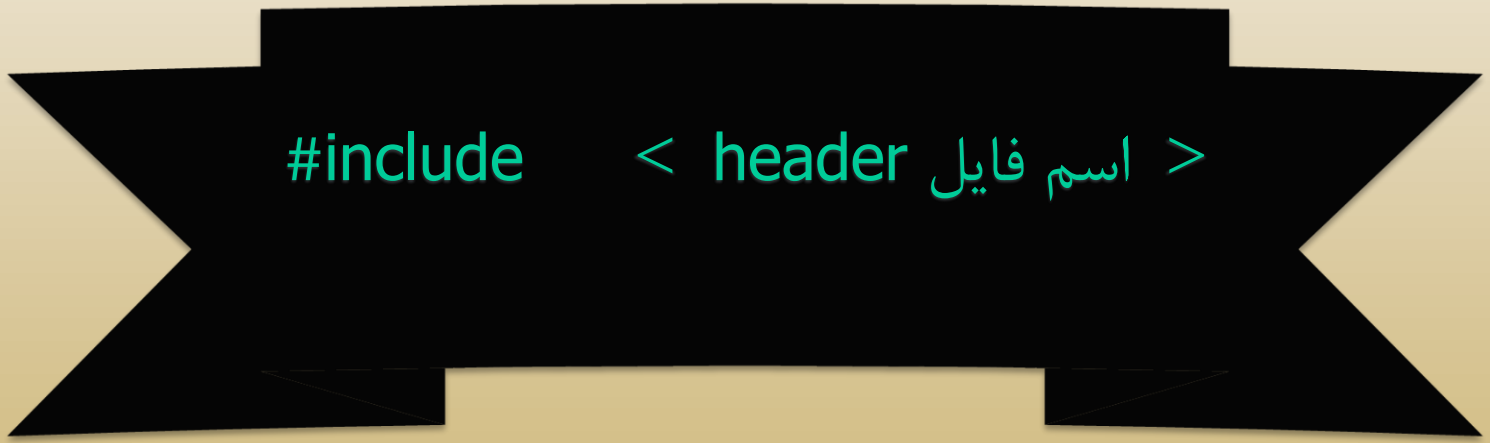
توابع کتابخانه :

زبان C++ مجهز به تعدادی توابع کتابخانه می‌باشد. به عنوان مثال تعدادی توابع کتابخانه برای عملیات ورودی و خروجي وجود دارند. معمولاً توابع کتابخانه مشابه ، به صورت برنامه‌های هدف (برنامه ترجمه شده به زبان ماشین) در قالب فایلهای کتابخانه دسته بندی و مورد استفاده قرارمي‌گیرند.

این فایلها را فایلهای **header** می‌نامند و داراي پسوند **h.** مي‌باشند.

نحوه استفاده از توابع کتابخانه ای:

براي استفاده از توابع کتابخانه خاصی بایستـــی نام فایل header آنرا در ابتدای برنامه در دستور #include قرار دهیم.



تابع	نوع	شرح	فایل هیدر
abs(i)	*int*	قدرمطلق i	stdlib.h
cos(d)	*double*	کسینوس d	math.h
exp(d)	*double*	e^x	math.h
log(d)	*double*	$\log_e d$	math.h
log10(d)	*double*	$\text{Log}_{10} d$	math.h
sin(d)	*double*	سینوس d	math.h
sqrt(d)	*double*	جذر d	math.h
strlen(s)	*int*	تعداد کرکترهای رشته s	string.h
tan(d)	*double*	تانژانت d	math.h
toascii(c)	*int*	کداسکی کرکتر c	stdlib.h
tolower(c)	*int*	تبدیل به حروف کوچک	stdlib.h
toupper(c)	*int*	تبدیل به حرف بزرگ	stdlib.h

برنامه در C++:

اکنون باتوجه به مطالب گفته شده قادر خواهیم بود که تعدادی برنامه ساده و کوچك به زبان C++ بنویسیم. براي نوشتن برنامه بایستی دستورالعملها را در تابع ( )main قرار دهیم و برای اینکار می‌توان به یکی از دو طریقی که در اسلایدها آمده است ، عمل نمود.

روش اول :

```
#include     <       >
int main( )
{
دستورالعمل ۱ ;
دستورالعمل ۲ ;
.
.
.
دستورالعمل n;
return 0 ;
}
```

برنامه در C++:

روش دوم :

```
#include      <          >
void main( )
{
دستورالعمل ۱ ;
دستورالعمل ۲ ;
.
.
.
دستورالعمل n;
}
```

برنامه ای که پیغام C++ is an object oriented language را روی صفحه مانیتور نمایش می دهد.

```
#include    <iostream.h>
int main( )
{
cout <<"C++ is an object oriented language \n" ;
return 0 ;
}
```

برنامه ای که پیغام welcome to c++! را روی صفحه مانیتور نمایش می دهد.

```
#include  <iostream.h>//allows program to output data to the screen
#include  <conio.h>

Using namespace std;

int main( )
{
cout <<" welcome to c++! "; //display message
getch();
return 0; //indicate that program ended successfully
}
```

برنامه ای که پیغام welcome to c++! را روی صفحه مانیتور نمایش می دهد.

```
1  # include <iostream> // allows program to output data to the screen
2  # include <conio.h>
3
4  using namespace std;
5
6  int main()
7  {
8      cout<< "welocome to c++!"; // display message
9
10     getch();
11     return 0; // indicate that program ended successfully
12 }
13
14
```

توضیحات کدهای مثال قبل:

خط ۱و ۲ دستورهای پیش پردازنده است،یعنی قبل از کامپایل(کار تبدیل کدهای برنامه به دستورهای قابل فهم کامپیوتر)برنامه این خط اجرا می شود،به عبارت دیگر هر خطی که با # شروع می شود یک دستور پیش پردازنده است.

دستور خط ۱ به پیش پردازنده می گوید که محتوای سرفایل جریان ورودی/خروجی را در برنامه قراردهد.

عبارت سبز رنگی که با علامت // شروع می شود،توسط کامپایلر نادیده گرفته می شود،زیرا از این نوع خطوط برای وضوح بخشیدن به برنامه استفاده می شود/همانطور که مشاهده می کنید،روبروی خط ۱ توضیحی برای علت استفاده از این خط نوشته شده است.(در این باره نکات تکمیلی توضیح می دهم).

خط ۳ یک خط خالی است که کامپایلار آن را نادیده در نظر میگیرد،در کل خطوط خالی در کامپایل برنامه خللی ایجاد نمی کند و به خوانایی کد کمک می کند.

توضیحات کدهای مثال قبل:

خط ۴ به این معنی است که ما از دستورهای استاندارد استفاده می کنیم که باید در تمام برنامه ها وجود داشته باشد.

از خط ۶ تا ۱۲، بدنه اصلی برنامه را تشکیل می دهد،که هر برنامه ما باید حداقل یک برنامه اصلی داشته باشد.

دستور main() یک تابع است که نمایانگر بدنه ی اصلی برنامه است،که گیومه خالی به این علت است که برنامه مقداری را برنمی گرداند،همین طور بدنه ی اصلی باید بین دو کروشه باز و بسته { } (خطوط ۷ و۱۲) قرارگیرد.(در حالت کلی دستور های شامل گیومه تابع هستند.

خط ۸ خطی است که جمله ی Welcom to c++! را در مانیتور نمایش می دهد. دستور coutبا علامت >>قسمت بعد از این علامت را نشان میدهد.

توضیحات کدهای مثال قبل:

خط ۱۰ به این علت به کارمی رود تا پنجره ای که خروجی برنامه را نمایش می دهد تا زمانی که ما کاری برای بستن آن انجام ندهیم،بسته نشود.بسته نشدن یعنی چی ؟به علت اینکه حاصل نتایج کار شما بر روی یک صفحه سیاه! نمایش داده می شود و خروجی برنامه تحت dos است شما باید از پایه تمام کارهای برنامه را بنویسید.

خط ۱۱،دستوری است که نشان می دهد برنامه به پایان رسیده است و عملیات ترجمه کد باید خاتمه یابد،البته این خط بیشتر زمانی به کار می رود که قرار است شما برنامه ی خود را بر روی فضایی استفاده کنید که دارای حافظه بسیار محدود است،ولی با توجه به اینکه فضای کامپیوتر بسیار زیاد است می توانید این کد را ننویسید.

برنامه زیر یك حرف انگلیسی كوچك را به حرف بزرگ تبدیل می‌نماید.

```
using namespace std;
#include<iostream>
#include <stdlib. h>
int main( )
{
 char c1 , c2;
cout << "Enter a lowercase letter:“;
cin >> c1;
c2 = toupper(c1);
cout << c2 << endl;
return 0;
}
```

دو عدد از نوع اعشاری را گرفته مجموع و حاصلضرب آنها را محاسبه و نمایش می‌دهد.

```
using namespace std;
#include<iostream>
int main( )
{
float  x,y,s,p ;
cin >> x >> y ;
s= x+y ;
p=x*y;
cout << s <<endl << p;
return 0 ;
}
```

❖ فصل چهارم : ساختارهای تصمیم گیری و تکرار

فهرست مطالب:

عملگرهای رابطه ای:

int

از این عملگرها براي تعيين اينكه آيا دو عدد با هم معادلند يا يكی از ديگری بزرگتر يا كوچكتر می‌باشد استفاده می‌گردد.

عملگرهای رابطه‌ای عبارتند از:

مساوی	= =
مخالف	! =
بزرگتر	>
بزرگتر يا مساوی	> =
كوچكتر	<
كوچكتر يا مساوی	<=

عملگر شرطی :

شکل کلی عملگر شرطی بصورت زیر می‌باشد:

```
expression _ test ? expression _ true : expression _ false
```

عملگر شرطی تنها عملگری در C++ می‌باشد که دارای سه عملوند می‌باشد.

مثال ۱ :

```
int x=10,y=20,b;
b=(x>y) ? x : y ;
```

این دو دستور العمل باعث میشوند که ماکزیمم مقادیر y و x در b قرار بگیرد.

مثال ۲ :

```
x>=10 ? cout << "passed" : cout << "failed" ;
```

اگر مقدار x بزرگتر یا مساوي ده باشد رشتة passed در غیر اینصورت رشته failed نمایش داده میشود.

دستورالعمل شرطی if :

توسط این دستور شرطی را تست نموده و بسته به آنکه شرط درست یا غلط باشد عکس‌العمل خاصی را نشان دهیم.

```
if( عبارت )
{
 دستورالعمل 1 ;
.
 دستورالعمل n;
}
else
{
 دستورالعمل 1 ;
.
 دستورالعمل n;
}
```

مثال ۱ :

```
if(x != y)
{
cout << x ;
++ x ;
}
else
{
cout << y ;
- - y ;
}
```

مثال ۲ :

برنامه زیر یک عدد اعشاری را از ورودی گرفته جذر آنرا محاسبه می‌نماید.

```
#include<iostream.h>
#include<math.h>
using namespace std;
int main( )
{
float  x,s;
cin >> x ;
if( x < 0 )
cout << " x is negative" << endl ;
else
{
s = sqrt(x) ;
cout << s << endl ;
}
return 0;
}
```

عملگر کاما :

تعدادی عبارت را می‌توان با کاما بهم متصل نمود و تشکیل یك عبارت پیچیده‌تری را داد.این عبارتها به ترتیب از چپ به راست ارزیابی شده و مقدار عبارت معادل عبارت nمي‌باشد.

(عبارت n , ... , عبارت 3 , عبارت 2 , عبارت 1)

مثال :

اگر داشته باشیم `int a=2 , b=4 , c=5 ;` **عبارت زیر را در نظر بگیرید:**

```
(++ a , a+b, ++ c, c+b)
```

مقدار عبارت برابر است با b+c که معادل 10 می‌باشد.

عملگرهای منطقی:

با استفاده از عملگرهای منطقی می‌توان شرطهای ترکیبی در برنامه ایجاد نمود. عملگرهای منطقی عبارتست از :

AND

OR

NOT

که در C++ به ترتیب بصورت زیر نشان داده میشود.

&&

||

!

جدول درستی سه عملگر شرطی :

And

a	b	a && b
true	true	True
true	false	False
false	true	False
false	false	False

Or

a	b	a || b
true	true	True
true	false	True
false	true	True
false	false	False

Not

a	!a
true	False
false	True

چند مثال :

اگر x برابر با 5 یا y مخالف صفر باشد مقدار x نمایش داده شود .

```
if ((x= = 5) || (y != 0))
    cout << x << endl ;
```

اگر مقدار x مخالف صفر باشد، آنگاه x برابر با صفر شود .

```
if(x !=0)
 x = 0 ;
```

برنامه زیر طول سه پاره‌خط را از ورودی گرفته مشخص می‌نماید که آیا تشکیل یك مثلث میدهد یا خیر؟

```
#include< iostream >
int main( )
{
float  a, b, c;
cout << "Enter three real numbers" << endl ;
cin >> a >> b >> c; //
if(( a < b + c) &&(b < a+c) &&(c < a+b))
cout << "It is a triangle" ;
else
cout << "Not a triangle" ;
return  0 ;
}
```

دستورالعمل for :

از دستور العمل for برای تکرار دستورالعملها استفاده می شود.
شکل کلی دستور for بصورت زیر می‌باشد:

```
for (عبارت 1 ; عبارت 2 ; عبارت 3)
 {
  دستورالعمل 1 ;
  دستورالعمل 2 ;
  .
  .
  .
  دستورالعمل n;
 }
```

ساختار for :

int

```
معرفی کنترل گر حلقه ;
for (گام حرکت;شرط حلقه;مقداردهی اولیه کنترل گر حلقه)
{
مجموعه دستورات بدنه حلقه;
}

int  i;
for (i=1;  i<=3;i++)
{
cout<<“hello \n”;
}
```

مثال :

```
void main()
{
cout<< ” hello \n”;
cout<< ” hello \n”;
cout<< ” hello \n”;
}
```

عبارت بالا برابراست با عبارت :

```
void main()
{
int i;
for(i=1;i<=3;i++)
}
cout<< ” hello \n”;
```

نکات:

لزومی ندارد که کنترل گر حلقه حتماً از1 شروع شود.

```
int i;
for(i=5;i<=7;i++)
{
cout<< ” hello \n”;
}
```

مقدار دهی اولیه کنترل گر حلقه می تواند خارج از دستور for باشد:

```
int i=1;
for(i ;i<=3;i++)
{
cout<< ” hello \n”;
}
```

نکات:

گام حرکت می تواند دربدنه دستور for تعریف شود.

```
int i=1;
for(i;i<=3;    )
{
cout<<" hello \n";
i++;
}
```

در دستور for اگرقسمت شرط خالی باشد،حلقه همیشه اجرا خواهد شد.به عبارتی شرطی برای توقف نداریم:

```
for(int i=1 ;   ;i++)
{
cout<<" hello \n";
}
```

نکات:

معرفی کنترل گر حلقه می تواند در داخل for باشد.

```
for(int i=1;i<=3;i--)
{
cout<<" hello \n";
}
```

لزومی ندارد که گام حلقه افزایشی باشد،بلکه می تواند کاهشی باشد:

```
for(int i=3;i>=1;i--)
{
cout<<" hello \n";
}
```

مثال: اعداد فرد سه رقمی را به صورت کاهشی نشان دهید.

```
using namespace std;
#include<iostream>
int main()
{
int i;
for(i=999;i>=101;i=i-2)
{
Cout<<i<<" \t ";
}
return 0;
}
```

مثال:اعداد فرد ۱ تا ۱۰۰ را نشان دهید.

```
using namespace std;
#include<iostream>
int main()
{
int i;
for(i=1;i<=100;i=i+2)
{
Cout<<i<<" \t ";
}
return 0;
}
```

برنامه زیر عدد صحیح و مثبت n را از ورودی گرفته فاکتوریل آنرا محاسبه ونمایش می‌دهد.

```
using namespace std;
#include<iostream>
int main( )
{
int  n, i ;
long fact = 1 ;
cout << "Enter a positive integer number";
cin >> n;
for( i=1; i<=n; ++i)
 fact *= i;
 cout << fact << endl;
return 0 ;
}
```

برنامه زیر مجموع اعداد صحیح و متوالی بین ۱ تا n را محاسبه نموده و نمایش می‌دهد.

```
using namespace std;
#include <iostream>
int main( )
{
int n, i=1 ;
long s = 0 ;
cin >> n ;
for(; i<=n; i++)
s += i;
cout << s ;
return  0 ;
 }
```

برنامه زیر ارقام 0 تا 9 را نمایش می‌دهد.

```
using namespace std;
#include <iostream>
int main( )
{
int  j=0 ;
for( ; j <= 9 ; )
    cout << j++ << endl;
return 0 ;
}
```

برنامه زیر کلیه اعداد سه رقمی که با ارقام 1 ، 2 ، 3 ایجاد می‌شوند را نمایش می‌دهد.

```
using namespace std;
#include <iostream>
int main( )
{
int  i,j,k,n;
for(i=1; i<=3; ++i)
for(j=1; j<=3; ++j)
for(k=1; k<=3; ++k)
{
n=i*100 + j*10+k;
cout << n << '\n' ;
}
return 0 ;
}
```

کاربرد دستور break در دستور for:

اگر در بدنه for از break استفاده شود،ادامه اجرای حلقه متوقف شده و حلقه خاتمه می یابد.

```
int i,x;
for(i=1;i<=100;i++)
{
cin>>x;
if(x==50)break;
}
```

قطعه کد فوق حداکثر ۱۰۰ عدد صحیح از ورودی می گیرد ولی اگر در بین اعداد ورودی عدد ۵۰ وارد شود بدون بررسی شرط حلقه از ادامه اجرای دستورات for اجتناب کرده و از حلقه خارج می شود.

مثال:قطعه کدی که تعدادی کاراکتر از صفحه کلید خوانده،بعد از فشردن دکمه ی E تعداد آن ها را مشخص کند.

```
using namespace std;
#include<iostream>
int main()
{
char ch;
int i;
for(i=0;    ;i++)
{
cin>>ch;
If(ch== ’ E’)break;
}
Cout<<i;
return 0;
}
```

حلقه for تودرتو :

int

می توان داخل بدنی دستور forهر دستور دلخواه دیگری نوشت.به عنوان مثال می توان از یک دستور for در بدنه دستور for استفاده کرد.

در حلقه های for تودرتو به ازای هر بار اجرایforبیرون حلقه forدرونی یکبار به طور کامل انجام می شود.

ساختار for تودرتو به این شکل است:

قطعه کد زیر عبارت Hello را ۱۵ بار اجرا می کند:

```
for (i=1;i<=5;i++)
        {
         for (j=1;j<=3;j++)
                cout<<“hello \n”;
         }
```

مثال:برنامه زیر کلیه اعداد سه رقمی که با ارقام 1,2,3 ایجاد می شوند را نمایش می دهد.

```
using namespace std;
#include<iostream>
int main()
{
int i,j,k,n;
for(i=1;i<=3;++i)
      for(j=1;j<=3;++j)
           for(k=1;k<=3;++k)
           {
             n=i*100+j*10+k;
              cout<<n<<"\n";
            }
return 0;
}
```

مثال:چاپ جدول ضرب اعداد

```
using namespace std;
#include<iostream>
int main()
{
int i,j;
for(i=1;i<=10; i ++)
      for(j=1;j<=10;j++)
           {
            cout<<i*j<<" ";
            if(j==10)
            cout<<"\n";
           }
return 0;
}
```

مثال:برنامه ای بنویسید که از ۱ تا ۲۰ عددی را از ورودی خوانده و برای هر کدام مجموع اعداد ۱ تا آن عدد را محاسبه کند.

```
using namespace std;
#include<iostream>
int main()
{
int i,j,x,sum;
for(i=1;i<=20; i ++)
        {
         sum=0;
         cin>>x;
         for(j=1;j<=x;j++)
         sum+=j;
         cout<<"sum of 1 to "<<x<<sum<<endl;
         }
return 0;
}
```

کاربرد حلقه for با دو اندیس :

مثال:برنامه ای بنویسید که ستون اعداد زیر را چاپ کند:

```
1,20
2,19
3,18
4,17
 .
 .
20,1
```

روش اول:

```
for(int i=1;i<=20;i++)
cout<<i<<" ,"<<21-i<<endl;
```

روش دوم:

```
for(int
   i=1;j=20;j>=1;i<=20;i++;j--)
cout<<i<<" ,"<<j<<endl;
```

دستور continue; در for :

اگردستور continue; درحلقه for استفاده شود،جملاتی از حلقه که هنوز اجرا نشده اند،بدون اجرا مانده و ادامه اجرا از انتهای حلقه آغاز خواهد شد:

```
#include <iostream>
    using namespace std;
    int main() {
    for (int i = 0; i < 8; i++) {
    if (i == 6) {
    continue;
    }
    cout << i << "\n";
    }
    return 0;
    }
```

خروجی

```
1
2
3
4
5
7
```

- **فصل پنجم : سایر ساختارهای تکرار**

فهرست مطالب:

دستورالعمل while :

از این دستور العمل مانند دستورالعمل for براي تكرار يك دستورالعمل ساده یا ترکیبی استفاده می‌گردد. شکل کلی این دستور العمل به صورت زیر می‌باشد.

```
while( شرط)
{
 دستورالعمل ۱ ;
 دستورالعمل ۲ ;
    .
    .
 دستورالعمل n ;
}
```

تفاوت دستور های while و for:

int

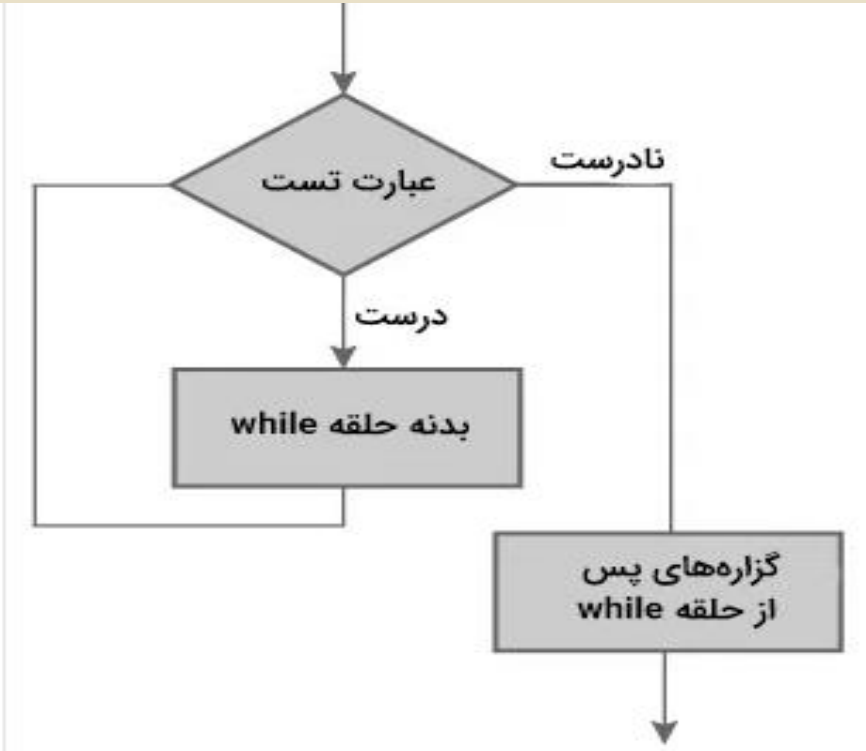
توسط این دستور شرطی را تست نموده و بسته به آنکه شرط درست یا غلط باشد ،عکس‌العمل خاصی را نشان دهیم.

2

دستورالعمل for زمانی استفاده میشود که تعداد دفعات تکرار از قبل مشخص و معین باشد. در صورتیکه تعداد دفعات تکرار مشخص نباشد بایستی از دستورالعمل while استفاده نمود.

حلقه while چگونه کار می کند:

- حلقه **while** عبارت شرط (تست)را ارزیابی می‌کند.
- اگر عبارت تست درست باشد، کد درون بدنه حلقه **while** مورد ارزیابی قرار می‌گیرد.
- سپس عبارت تست مجدداً ارزیابی می‌شود. این فرایند تا زمانی که عبارت تست نادرست شود ادامه می‌یابد.
- هنگامی که عبارت تست نادرست شود، حلقه **while**خاتمه می‌یابد.

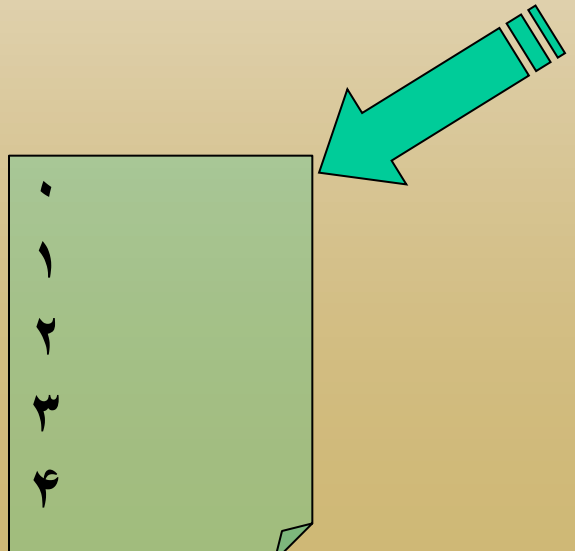


مثال :

توسط این دستور شرطی را تست نموده و بسته به آنکه شرط درست یا غلط باشد عکس‌العمل خاصی را نشان دهیم.

```
int x=0
while(x<5)
cout << x ++<< endl;
```

با اجرای قطعه برنامه فوق مقادیر زیر نمایش داده می شود :



```
#include<iostream>
using namespace std;
int main()
{
    int number , i=1 , fact=1;
        cout<<"Enter a positive integer:";
         cin>>number;
        while(i<=number)
       {
        fact *=i;
        i++;
        }
        cout<<"factorial of"<<number <<" = "<<fact;
       }
       return 0;
       }
```

```
Enter a positive integer: 4
Factorial of 4 = 24
```

در این برنامه از کاربر تقاضا می‌شود که یک عدد صحیح مثبت وارد کند که در متغیر number ذخیره می‌شود.
فرض کنید کاربر مقدار ۴ را وارد کند.
سپس حلقه whileشروع به اجرای کد می‌کند. طرز کار حلقه whileبه صورت زیر است:
۱.در ابتدا i=1است و عبارت تست یعنی i <= numberدرست است، از این رو مقدار فاکتوریل برابر با ۱ خواهد بود.
۲.عبارت iبه مقدار ۲ به‌روزرسانی می‌شود، عبارت تست trueاست، مقدار فاکتوریل برابر با ۲ می‌شود.
۳.عبارت iبه مقدار ۳ به‌روزرسانی می‌شود، عبارت تست trueاست، مقدار فاکتوریل برابر با ۶ می‌شود.
۴.عبارت iبه مقدار ۴ به‌روزرسانی می‌شود، عبارت تست trueاست، مقدار فاکتوریل برابر با ۲۴ می‌شود.
۵.عبارت iبه مقدار ۵ به‌روزرسانی می‌شود، عبارت تست falseمی‌شود و حلقه خاتمه می‌یابد.

برنامهٔ زیر n مقدار از نوع اعشاری را گرفته میانگین آنها را محاسبه و در متغیر avg قرار می‌دهد.

```
#include  <iostream. >
using namespace std;
int main( )
{
int count = 0 , n;
float  x, sum = 0 , avg ;
cin >> n ; /*   تعداد مقادیر ورودی n*/
while(count < n)
{
cin >> x ;
sum += x ;
++ count ;
}
avg = sum / n ;
cout << avg << endl;
return 0 ;
 }
```

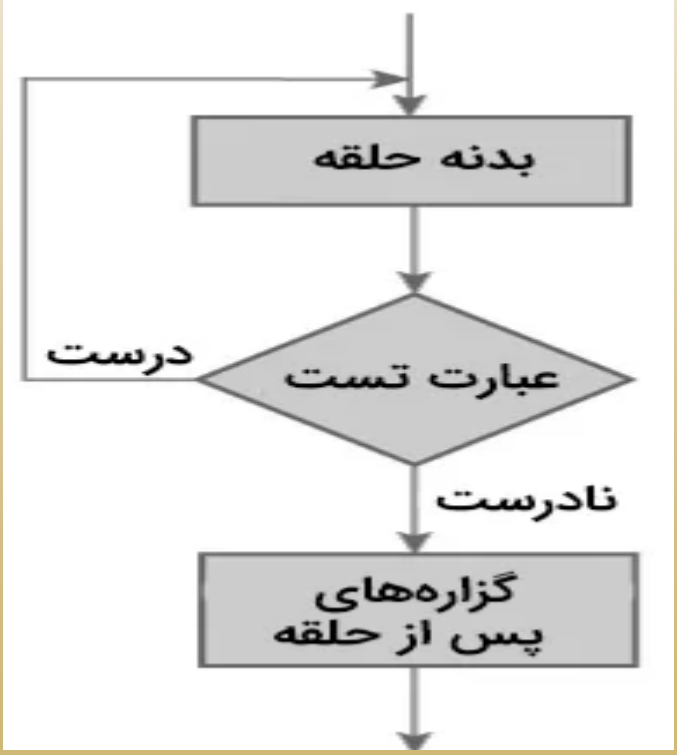
دستورالعمل do while :

این دستور العمل نیز برای تکرار یك دستورالعمل ساده یا ترکیبی استفاده می‌شود. شکل کلی این دستورالعمل بصورت زیر می‌باشد.

```
do
{
 دستورالعمل ۱ ;
 دستورالعمل ۲ ;
    .
    .
 دستورالعمل n ;
} while( شرط);
```

حلقه do… while چگونه کار می کند:

- کدهای درون بدنه حلقه دست‌کم یک بار اجرا می‌شوند. سپس تنها عبارت شرط(تست) بررسی می‌شود.
- اگر عبارت تست درست باشد، بدنه حلقه اجرا می‌شود. این فرایند تا زمانی ادامه می‌یابد که عبارت نادرست شود.
- هنگامی که عبارت تست نادرست باشد، حلقه do…while خاتمه می‌یابد.



تفاوت دستورهای do while و while :

در دستورالعمل while ابتدا مقدار شرط ارزیابی شده اما در دستورالعمل do while ابتدا دستورالعمل حداقل یکبار اجرا شده سپس مقدار شرط ارزیابی می‌گردد. بنابراین دستورالعمل do while حداقل یک بار انجام میشود.

مثال :

ارقام 0 تا 9 را روی ده خط نمایش می‌دهد .

```
#include   <iostream.h>
int main( )
{
int count = 0;
do
cout << count ++<<endl ;
while(count <= 9);
return 0 ; }
```

مثال :

```
#include<iostream>
using namespace std;
int main()
{
    float number,sum=0.0;
    do{
        cout<<"Enter a number:";
        cin>>number;
        sum+=number;
       }
While(number!=0.0);
          cout<<"Total sum="<<sum;
          }
return 0;
}
```

```
Enter a number: 2
Enter a number: 3
Enter a number: 4
Enter a number: -4
Enter a number: 2
Enter a number: 4.4
Enter a number: 2
Enter a number: 0
```

دستورالعمل break:

این دستورالعمل باعث توقف دستورالعملهای تکرار(while ,do while , for) شده و کنترل به خارج از این دستورالعملها منتقل می‌نماید.

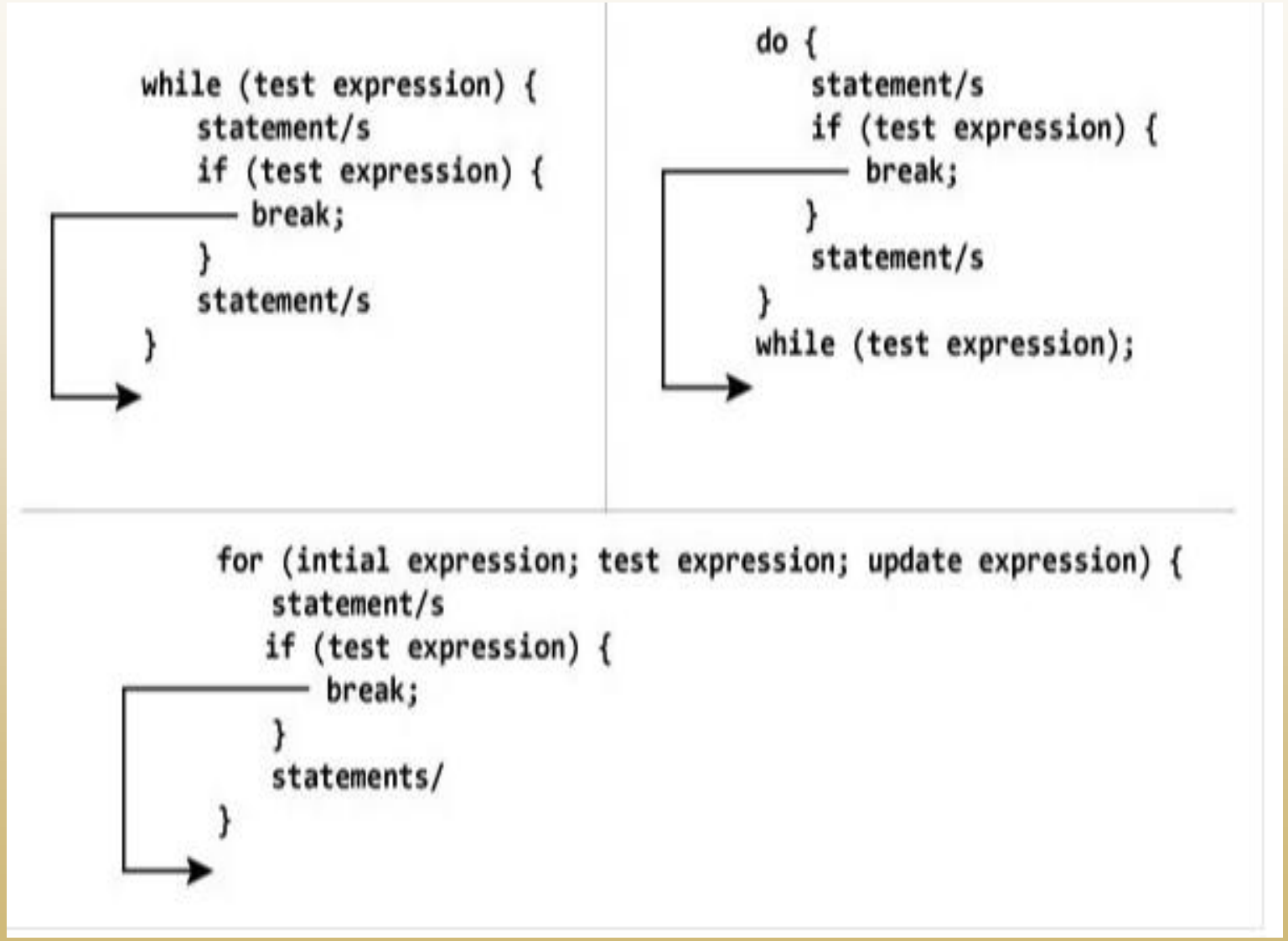
دستور العمل break:

اگر در بدنه for از break استفاده شود،ادامه اجرای حلقه متوقف شده و حلقه خاتمه می یابد.

```
int i,x;
for(i=1;i<=100;i++)
{
cin>>x;
if(x==50)break;
}
```

قطعه کد فوق حداقل ۱۰۰ عدد را از ورودی میگیرد ولی اگر در بین اعداد ورودی عدد ۵۰ وارد شود بدون بررسی شرط حلقه از ادامه اجرای دستورات forاجتناب کرده و از حلقه خارج می شود.

نحوه اعمال دستور العمل break:



مثال 1:برنامه C ++ برای افزودن همه اعداد وارد شده از سوی کاربر تا زمانی که کاربر عدد ۰ وارد نماید:

```
#include   <iostream.h>
using namespace std;
int main( ){
Float number,sum=0.0;
while(true)
{
cout << " Enter a number : ";
Cin>>number;
if(number !=0.0){
Sum +=number;
}
else
{
break;
}
}
cout << " Sum=    " << sum ;
return 0 ;
}
```

```
Enter a number: 4
Enter a number: 3.4
Enter a number: 6.7
Enter a number: -4.5
Enter a number: 0
Sum = 9.6
```

در برنامه فوق، عبارت تست همواره صحیح است. از کاربر تقاضا می‌شود که عدد دیگری را وارد کند هنگامی که کاربر مقدار ۰ وارد می‌کند، عبارت تست درون گزاره if نادرست است و بدنه else اجرا می‌شود که موجب خاتمه حلقه می‌شود. در نهایت مجموع نمایش پیدا می‌کند.

مثال ۲ :

جمع تعدادی عدد که بین ۰ و ۱۰۰۰ هستند.اگربین اعداد وارد شده عدد منفی وارد شود،بواسطه breakحلقه خاتمه می یابد.
اگرعدد وارد شده بزرگتر از ۱۰۰۰ باشد نیز شرط حلقه برآورده نشده و حلقه خاتمه می یابد.

```
#include   <iostream.h>
using namespace std;
int main( )
{
float x, s=0.0 ;
cin >> x ;
while(x <= 1000.0)
{
if(x < 0.0){
cout << "Error-Negative Value" ;
break;
}
s += x ;
cin >> x ;}
cout << s << endl ;
return 0 ;
}
```

مثال ۳:

int

```
#include   <iostream.h>
using namespace std;
int main( )
{
int count = 0 ;
while( 1 )
{
count ++ ;
if(count > 10 )
break ;
}
cout << "counter :    " << count << "\n";
return 0 ;
}
```

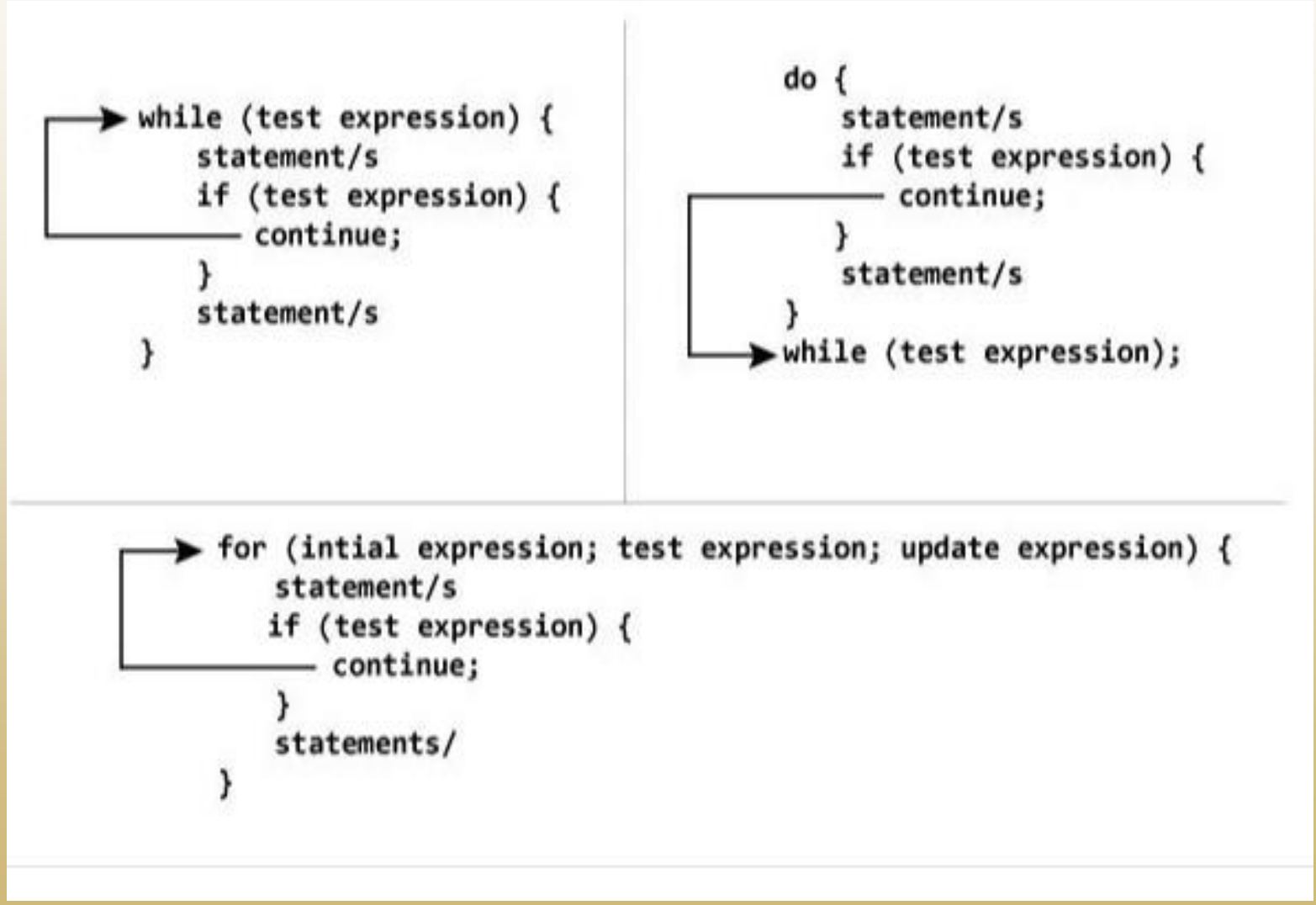
خروجی

Counter=11

دستورالعمل continue :

از دستورالعمل continue می‌توان در دستورالعملهای تکرار do while ، while ، for استفاده نمود. این دستورالعمل باعث می‌شود که کنترل به ابتدای دستورالعملهای تکرار منتقل گردد.

نحوه اعمال دستور العمل continue:



مثال :

```
#include <iostream>
using namespace
 std;int main()
{
for (int i = 1; i <= 10; ++i)
   {
  if ( i == 6 || i == 9)
     {
    continue;
        }
  cout << i << "\t";
  }
return 0;
}
```

```
1 2 3 4 5 7 8 10
```

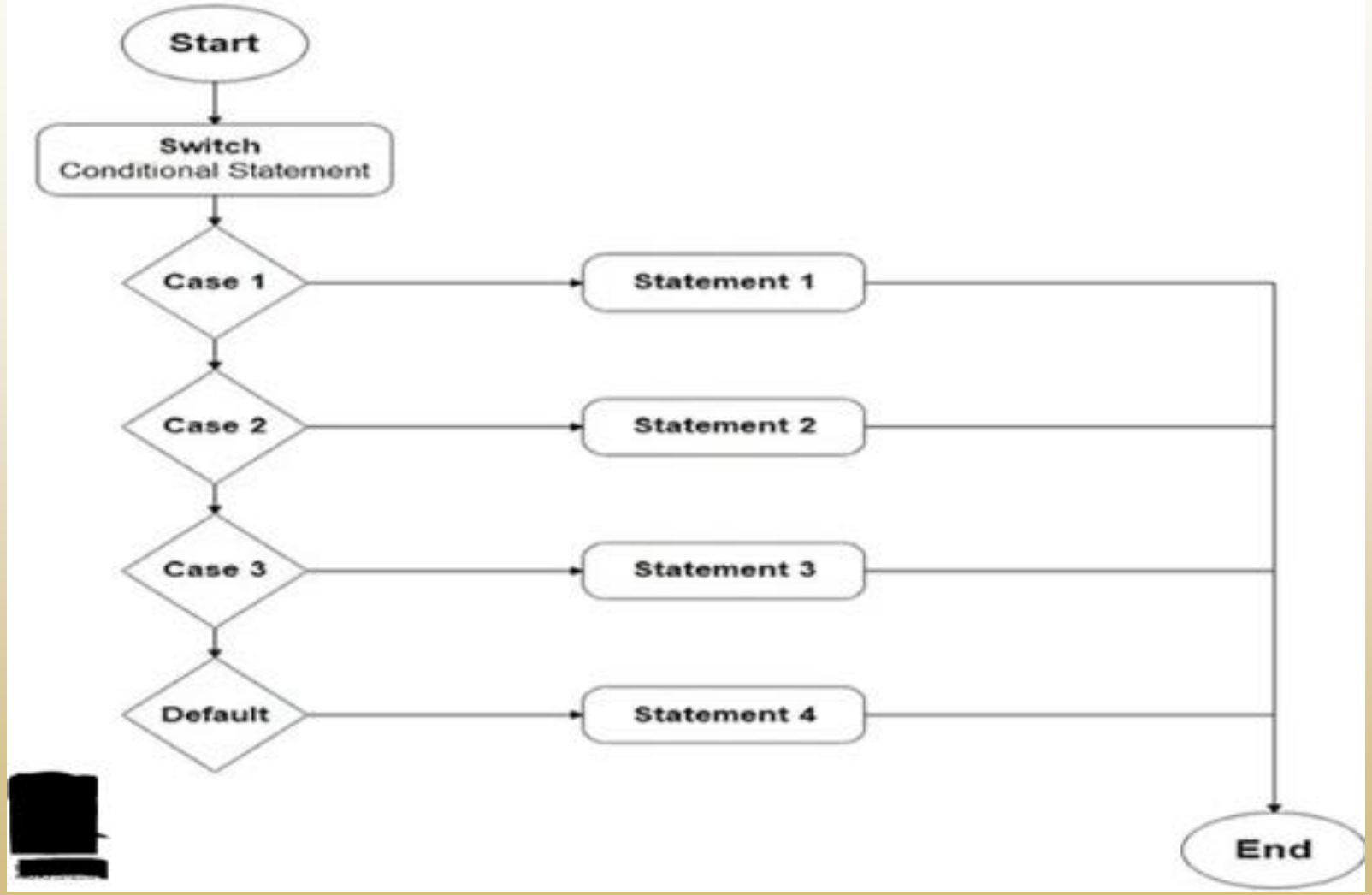
در برنامه فوق، زمانی که iبرابر با ۶ یا ۹ باشد، اجرای گزاره زیر درون حلقه با استفاده از گزاره ; Continueرد می‌شود:

دستورالعمل switch:

همانطور که می دانید از دستورالعمل شرطی(if else) می توان بصورت تودرتو استفاده نمود ولی از طرفی اگر عمق استفاده تو در تو از این دستورالعمل زیاد گردد، درك آنها مشکل میشود . براي حل اين مشكل C++ ، دستورالعمل switch كه عملاً يك دستورالعمل چند انتخابی مي‌باشد را ارائه نموده است.

```
switch(عبارت)
{
case    valueone : statement;
                   break;
case     valuetwo: statement;
                  break;
.
.
.
case       valuen : statement;
                    break;
default: statement ;
}
```

نحوه اعمال دستور العمل switch …case:



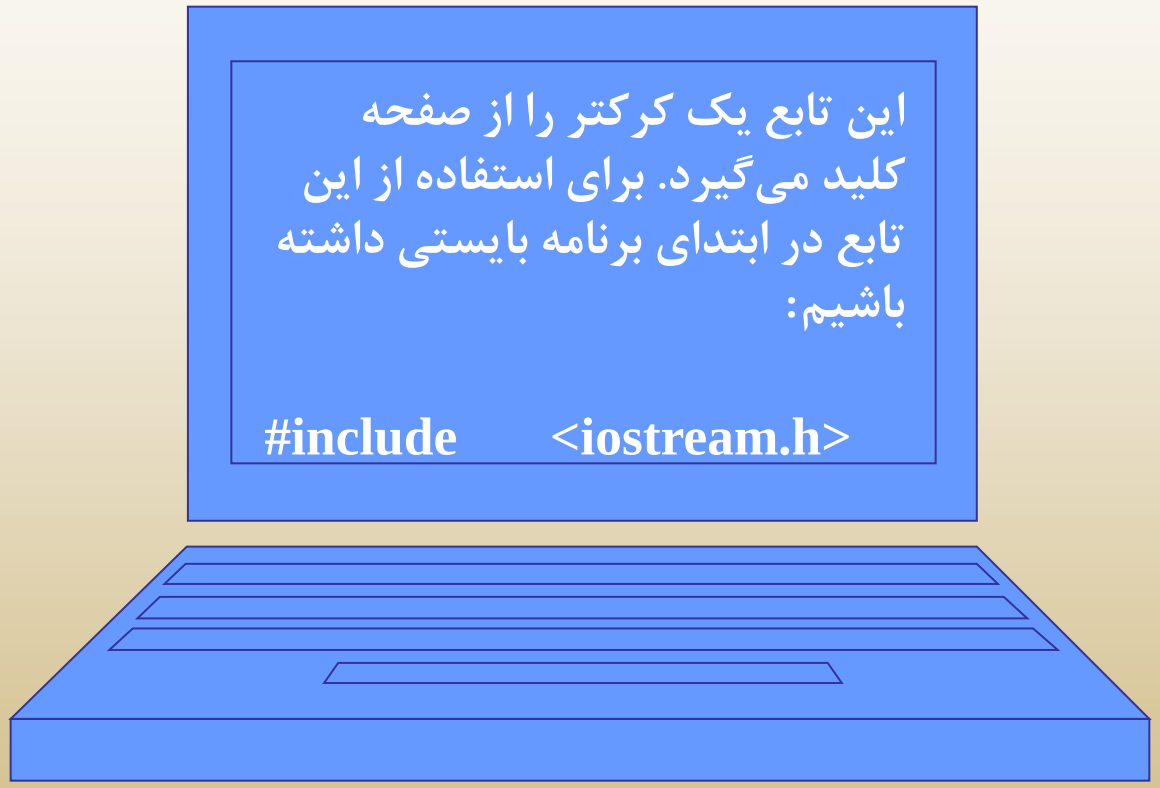
مثال ۱ :

```
#include   <iostream.h>
void main( )
{
unsigned int n ;
cin >> n;
switch(n)
{
    case   0:
           cout << "ZERO" << endl ;
           break;
    case   1:
           cout << "one" << endl ;
           break ;
    case 2:
           cout << "two" << endl ;
           break;
    default :
           cout << "default" << endl;
}       / * end of switch statement * /
}
```

مثال ۲ :

```
#include   <iostream.h>
void main( )
{
unsigned  int  n;
cin >> n ;
switch(n) {
case 0 :
case 1:
case 2:
        cout << "Less Than Three" << endl;
         break;
case 3:
         cout << "Equal To Three" << endl ;
          break;
default:
         cout << "Greater Than Three" << endl;
           }
}
```

تابع cin.get() :


این تابع یک کرکتر را از صفحه کلید می‌گیرد. برای استفاده از این تابع در ابتدای برنامه بایستی داشته باشیم:
#include <iostream.h>

قطعه برنامه ذیل یک کرکتر را از صفحه کلید گرفته و نمایش می‌دهد.

```
char  x;
x = cin.get( );
cout << x;
```

در قطعه برنامه ذیل ازتابع cin.get() و دستور switch استفاده شده است.

```
char x;
x = cin.get( );
switch(x)  {
case  ' r ' :
case  ' R ' :
            cout << "RED" << "\n" ;
            break ;
case  ' b ' :
case  ' B ' :
             cout << "BLUE" << endl ;
             break ;
case  ' y ' :
case  ' Y ' :
            cout <<  "YELLOW" << endl;
```

❖ فصل ششم : اعداد تصادفی

فهرست مطالب:

اعداد تصادفی:

مقادیر تصادفی یا شانسی در اکثر برنامه‌های کاربردی در زمینه شبیه سازی و بازیهای کامپیوتری نقش مهمی را ایفا می‌نمایند. برای ایجاد یک عدد تصادفی صحیح بین ۰ و ۳۲۷۶۷ بایستی از تابع rand( ) استفاده نمائیم.

rand( )

برنامه زیر 10 عدد تصادفی بین 0 و 32767 را ایجاد می‌نماید.

```
#include  <stdlib.h>
#include  < iostream.h>
using namespace std;
int main( )
{
for(int j=1; j<=10; ++j)
cout << rand(  ) << '\n' ;
return 0 ;
}
```

نـکته :

اگر برنامهٔ فوق را چندبار اجرا نمائیم جواب یکسانی را از کامپیوتری‌می گیریم.
برای تصادفی کردن اعداد می‌بایستی از تابع ( )srand استفاده نمائیم.
این تابع به یک آرگومان صحیح از نوع unsigned نیاز دارد.
به این آرگومان seed گفته می‌شود.

در اسلاید بعد برنامه قبلی را با تابع ( )srand نوشته ایم.

برنامه زیر 10 عدد تصادفی بین 0 و 32767 را ایجاد می‌نماید.(srand)

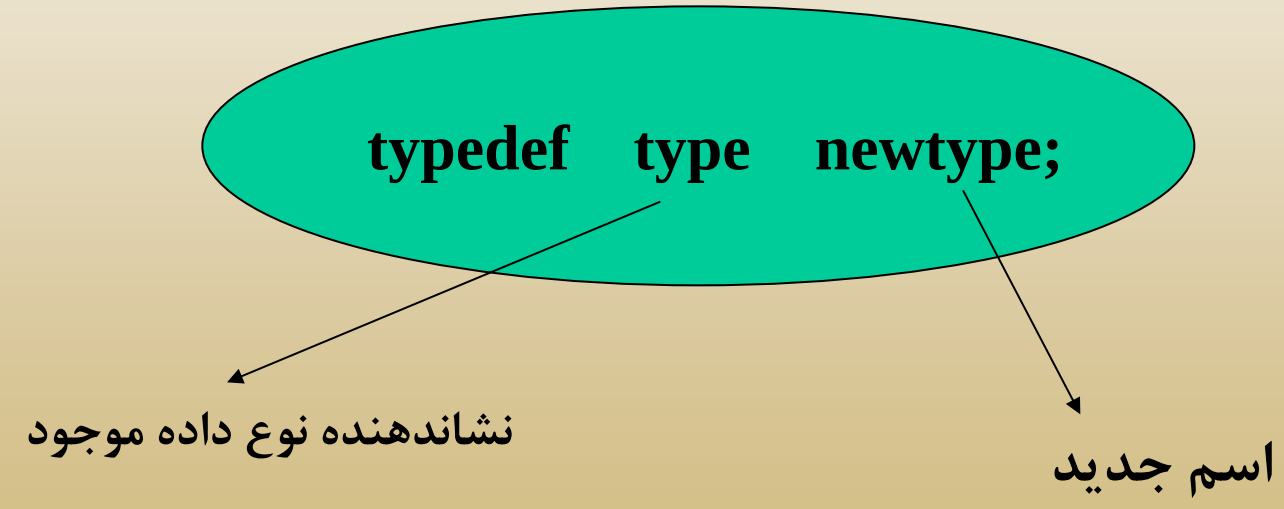
```
#include<stdlib.h>
#include< iostream.h>
using namespace std;
int main( )
{
unsigned     seed;
cout << "Enter seed value :  " ;
cin >> seed ;
srand(seed);
for(int  j=1; j<=10; ++j)
cout << rand( ) << ′ \n ′;
return 0  ;
}
```

برنامه زیر نتیجه پرتاب دو تاس را نمایش می‌دهد.

```
#include< iostream.h>
#include<stdlib.h>
using namespace std;
int main(  )
{
unsigned   seed, d1, d2;
cout << "Enter seed: " ;
cin >> seed ;
srand(seed) ;
d1= 1+rand( )% 6 ;
d2= 1+rand( )% 6 ;
cout << d1 << "   "<< d2 ;
return 0 ;
}
```

تعریف نوع داده (typedef):

از typedef می‌توان برای تعریف نوع داده‌های جدید که معادل نوع داده‌های موجود باشد استفاده نمود. شکل کلی عبارتست از :



مثال ۱:

```
typedef  int  integer;
```

حال می‌توان y و x را بصورت زیر تعریف نمود :

```
integer  x,y;
```

در زبان C++ ، کلمه کلیدی **typedef** برای تعریف یک نام جدید برای یک نوع **(type)** موجود استفاده می‌شود. این کار باعث می‌شود خوانایی کد افزایش یابد یا تغییر نوع در آینده راحت‌تر انجام شود.

مثال ۲:

```
#include <iostream>
using namespace std;
typedef unsigned int uint;
int main()
{
 uint a = 10;
  cout << "a = " << a << endl;
  return 0;
}
```

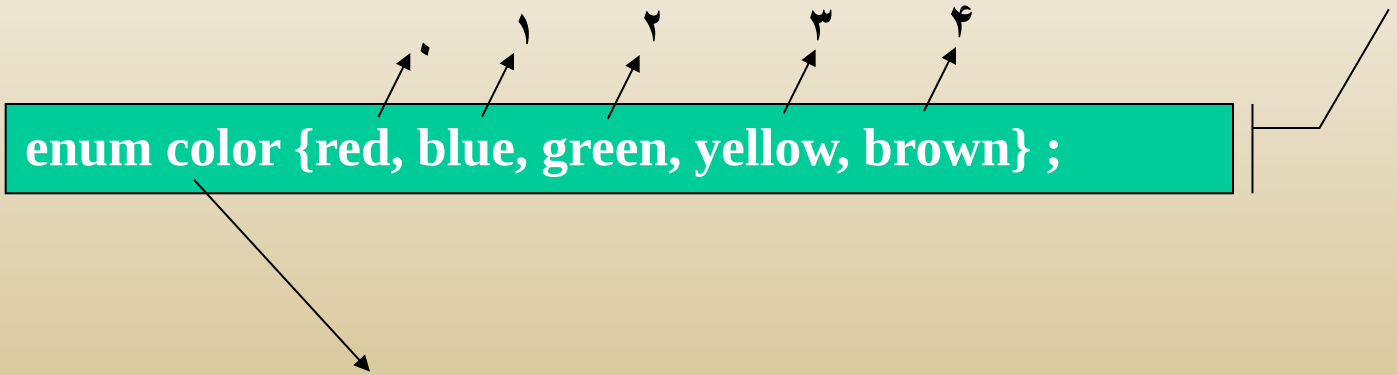
در این مثال:

- **با استفاده از typedef unsigned int uint; یک نام جدید uint برای نوع unsigned int تعریف شده است.**
- **در نتیجه می‌توان به‌جای unsigned int از uint استفاده کرد.**

داده‌های از نوع شمارشی :

به منظور معرفی داده‌های از نوع شمارشی از کلمه **enum** استفاده می‌گردد.

مثال :



color یک نوع داده شمارشی می‌باشد.

در زبان C++ ، دستور **enum** مخفف (*enumeration*) برای تعریف مجموعه‌ای از مقادیر ثابت صحیح استفاده می‌شود. این دستور معمولاً برای تعریف حالت‌ها، گزینه‌ها یا انواع مشخص قابل شمارش استفاده می‌شود.

چند مثال :

```
emum status {married, devorced, vidow, single};
status  a ;
a= single ;
```

```
enum days {sat, sun, mon, tue, wed, thr, fri};
```

```
enum bread {lavash, fantezi, taftoon, barbari};
```

```
enum color { yellow, red=2, brown, white };
color  x=brown;
```

استفاده در برنامه:

```
#include <iostream>
using namespace std;
enum Color {    Red,    Green,    Blue};
int main()
 {
 Color myColor = Green;
   if (myColor == Green)
{
  cout << "The color is green." << endl;
  }
 return 0;
}
```

فرمتهای مختلف مقادیر خروجی:

include <iomanip.h>

مقدار x بطور غیر علمی با نقطه اعشار ثابت نمایش داده می شود.

```
double   x=1050 ;
cout << setiosflags(ios : : fixed | ios: : showpoint ) << setw(23)
     << setprecision(2) << x << endl ;
```

مقدار x با طول میدان ۲۳ نمایش داده می شود.

مقدار x با دو رقم اعشار نمایش داده می شود.

بنابراین مقدار x بصورت زیر نمایش داده می شود :

1050.00 شانزده ستون خالی

توضیح اجزای کد:

۱. double x = 1050;
   - تعریف یک متغیر عددی از نوع اعشاری (دابل) با مقدار ۱۰۵۰.
۲. setiosflags(ios::fixed | ios::showpoint)
   - ios::fixed: نمایش عدد به‌صورت اعشاری ثابت (نه علمی).
   - ios::showpoint: نمایش نقطه اعشار حتی اگر عدد عدد صحیح باشد (مثل 1050.00).
۳. setw(23)
   - تنظیم عرض فیلد چاپ به ۲۳ کاراکتر. اگر خروجی کوتاه‌تر باشد، از فضای خالی (پیش‌فرض راست‌چین) برای پر کردن استفاده می‌شود.
۴. setprecision(2)
   - تنظیم دقت (تعداد رقم‌های بعد از اعشار) به ۲ رقم.

❖ فصل هفتم : آرایه ها

فهرست مطالب:

آرایه یک بعدی :

آرایه یك فضای پیوسته از حافظه اصلی كامپیوتر می‌باشد كه می‌تواند چندین مقدار را در خود جای دهد.

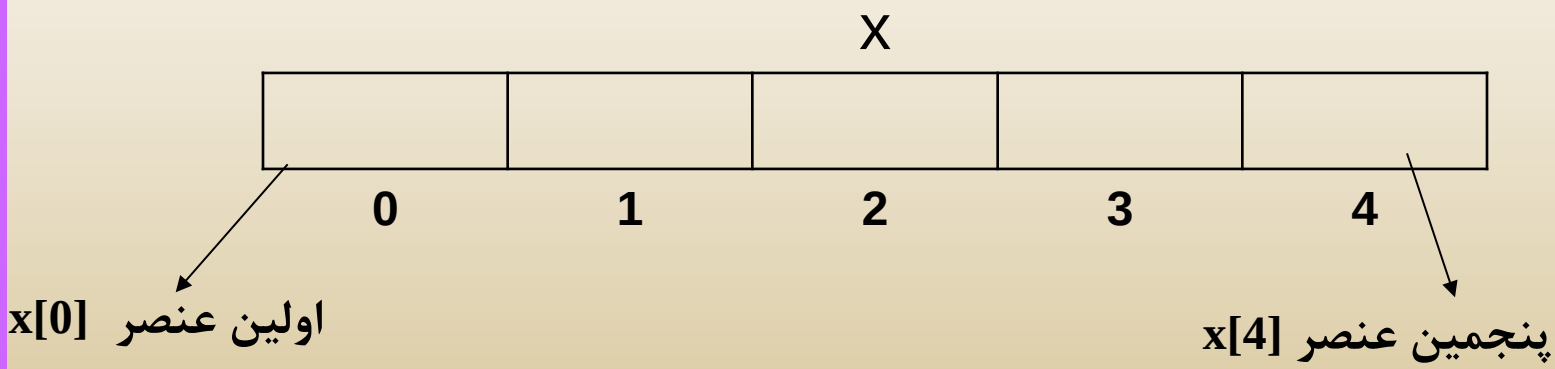
- **كلیه عناصر یک آرایه از یک نوع می‌باشند.**
- **عناصر آرایه بوسیله اندیس آنها مشخص می‌شوند.**
- **در C++ ، اندیس آرایه از صفر شروع می‌شود.**

کاربرد آرایه ها:

آرایه‌ها در برنامه‌نویسی در مواردی کاربرد دارند که بخواهیم اطلاعات و داده‌ها را در طول اجرای برنامه حفظ نمائیم.

آرایه یک بعدی از نوع int:

```
int   x[5] ;
```



تخصیص مقادیر اولیه به عناصر آرایه :

آرایه ها را می‌توان مثل متغیرهای معمولی تعریف و استفاده کرد. با این تفاوت که آرایه یک متغیر مرکب است و برای دستیابی به هر یک از خانه‌های آن باید از ایندکس استفاده نمود.

```
int  x[5]= {4, 2, 5, 17, 30};
```

x

4	2	5	17	30
0	1	2	3	4

پردازش آرایه ها در C++

مثال: دستیابی مستقیم به عناصر آرایه برنامه ساده زیر یک آرایه سه عنصری را تعریف می‌کند و سپس مقادیری را در آن قرار داده و سرانجام این مقادیر را چاپ می‌کند.

```
#include <iostream>
using namespace std;
int main()
{
int a[3];
a[2] = 55;
a[0] = 11;
a[1] = 33;
cout << "a[0] = " << a[0] << endl;
cout << "a[1] = " << a[1] << endl;
cout << "a[2] = " << a[2] << endl;
}
```

دریافت مقادیر عناصر آرایه :

```
int  x[5];
for(int i=0; i<=4; ++i)
cin >> x[ i ] ;
```

نمایش مقادیر عناصر آرایه :

```
for(int   i=0; i<=5; ++i)
cout << x[ i ] ;
```

اگر تعداد مقادیر اولیه کمتر از تعداد عضوهای آرایه باشد عضوهای باقیمانده بطور اتوماتیك، مقدار اولیه صفر می‌گیرند.

```
int   x[5] = {12, 5, 7};
```

x

12	5	7	0	0
0	1	2	3	4

بایستی توجه داشت که آرایه‌ها به صورت ضمنی مقدار اولیه صفر نمی‌گیرند. برنامه نویس باید به عضو اول آرایه، مقدار اولیه صفر تخصیص دهد تا عضوهای باقی مانده بطور اتوماتیك، مقدار اولیه صفر بگیرند.

```
int   x[5] = {0} ;
```

X

0	0	0	0	0
0	1	2	3	4

دستور زیر یك آرایه یك بعدی شش عنصری از نوع float ایجاد می‌نماید.

```
float x[  ] = {2.4, 6.3, -17.1, 14.2, 5.9, 16.5} ;
```

x

2.4	6.3	-17.1	14.2	5.9	16.5
0	1	2	3	4	5

مثال


```
#include <iostream>
using namespace std;
 int main()
{
// Creating an integer array named arr of size 10.
  int arr[10];    // accessing element at 0 index and
     setting its value
   // to 5.
   arr[0] = 5;
    // access and print value at 0 index we get the
     output
  // as 5.
 cout << arr[0];
  return 0;}
```

در خط هفتم از کدهای فوق، آرایه‌ای با اندازه ۱۰ ایجاد شده است و سپس در خط دهم از این مثال عدد ۵ به اندیس شماره صفر این آرایه تخصیص داده می‌شود. در نهایت در خط ۱۳ام از برنامه، خروجی آرایه ایجاد و چاپ شده است. تنها عدد ۵ در خروجی این برنامه نشان داده می‌شود.

برنامه زیر 100 عدد اعشاری و مثبت را گرفته تشکیل یك آرایه میدهد سپس مجموع عناصر آرایه را مشخص نموده نمایش می‌دهد.

```
#include  <iostream.h>
#include   <iomanip.h>
int main( )
{
const int arrsize = 100 ;
float x[ arrsize], tot = 0.0 ;
for(int   j=0; j<arrsize; j++)
cin >> x[ j ];
for(j=0; j<arrsize; j++)
cout << setiosflags(ios::fixed  ios :: showpoint ) << setw(12) <<
    setprecision(2) << x[ j ] << endl;
for(j=0; j<arrsize; j++)
tot += x[ j ] ;
cout << tot ;
return 0 ;
}
```

برنامه ذیل 20 عدد اعشاری را گرفته تشکیل یك آرایه داده سپس کوچکترین عنصر آرایه را مشخص و نمایش می‌دهد.

```
#include  <iostream.h>
#include   <conio.h>
int main( )
{
float   x[20], s;
int  j ;
clrscr( ) ;
for(j=0; j<20 ; ++j)
 cin >> x[ j ];
s = x[0 ] ;
for(j=1; j<20; ++j)
if(x[ j] <s)
s = x[ j ];
cout << s << endl;
return 0;
}
```

آرایه‌های دوبعدی (ماتریس‌ها):

ماتریس ها بوسیله آرایه‌های دوبعدی در کامپیوترنمایش داده میشوند.

```
int a[3][4];
```

	ستون 0	ستون 1	ستون 2	ستون 3
سطر 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
سطر 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
سطر 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

تخصیص مقادیر اولیه به عناصر آرایه :

```
int a[3][4]={ {1,2,3,4}, {5,6,7,8}, {9,10,11,12} } ;
```

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

نکته ۱:

```
int   a[3][4]= { {1}, {2,3} , {4,5,6} } ;
```

	0	1	2	3
0	1	0	0	0
1	2	3	0	0
2	4	5	6	0

نکته ۲:

```
int   a[3][4]= {1, 2, 3, 4,5 } ;
```

	0	1	2	3
0	1	2	3	4
1	5	0	0	0
2	0	0	0	0

نکته ۳:

در یك آرایه دواندیسی، هر سطر، در حقیقت آرایه‌ای یك اندیسی است. دراعلان آرایه‌های دواندیسی ذکر تعداد ستونها الزامی است.

```
int   a[  ][4]={1,2,3,4,5};
```

	0	1	2	3
0	1	2	3	4
1	5	0	0	0

❖ فصل هشتم : توابع

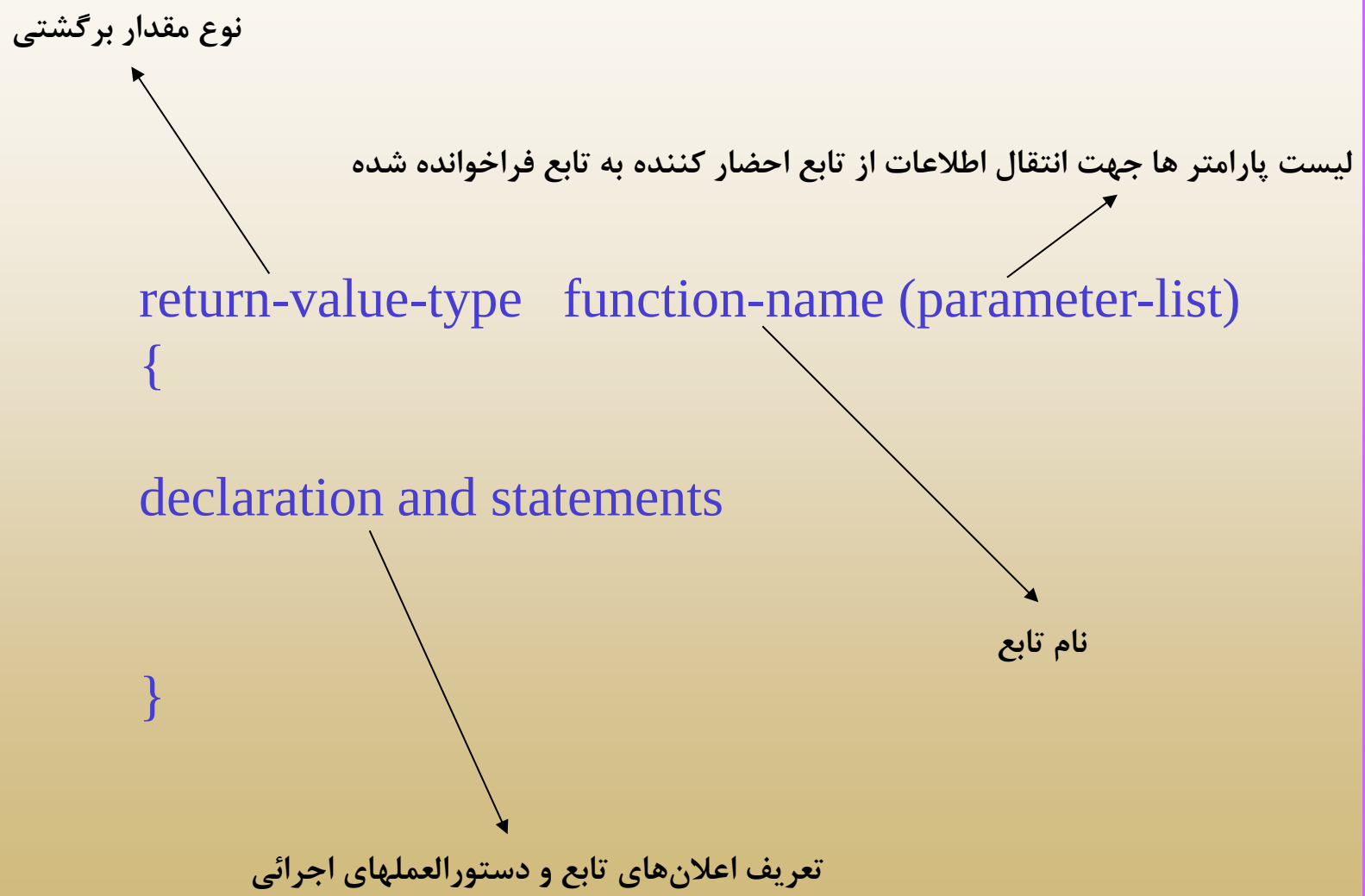
فهرست مطالب:

تعریف توابع:

استفاده از توابع در برنامه‌ها به برنامه‌نویس این امکان را می‌دهد که بتواند برنامه‌های خود را به صورت قطعه قطعه برنامه بنویسد. تا کنون کلیه برنامه‌هائی که نوشته‌ایم فقط از تابع ( )main استفاده نموده‌ایم .

در زبان C++، توابع (Functions) قطعه‌کدی هستند که برای انجام کار خاصی تعریف می‌شوند و می‌توانند چندین بار در برنامه فراخوانی شوند. استفاده از توابع باعث ساختارمند شدن برنامه، افزایش خوانایی، و کاهش کد تکراری می‌شود.

شکل کلی توابع به صورت زیر می باشد:



مثال ۱: تابعی که دو عدد صحیح را جمع می‌کند

```
int add(int a, int b)
{
int result = a + b;
    return result;
}
```

تابع زیر یک حرف کوچک را به بزرگ تبدیل می‌نماید.


```
char   low_to_up(char c1)
{
char c2;
c2 = (c1>= ′ a ′ && c1<= ′ z ′)?(′ A ′ + c1- ′ a ′): c1;
(c2) ; return
}
```

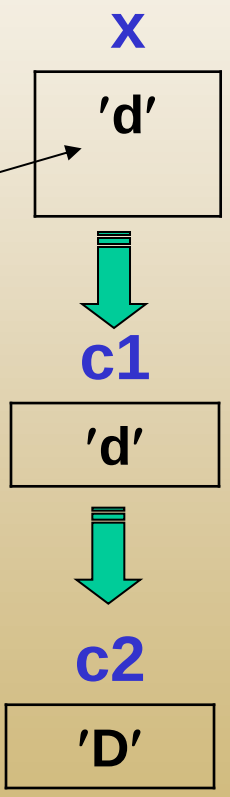
اگر c1 یک حرف کوچک بین 'a’ تا 'z’ باشد:

- فاصله‌ی عددی بین 'a'و 'A'را استفاده می‌کنیم تا حرف بزرگ متناظر را بسازیم.
- مثلاً اگر c1 = 'e’ باشد، آنگاه:
  - 'A' + ('e' - 'a') = 'A' + 4 = 'E'

- در غیر این صورت، خود c1 بدون تغییر بازگردانده می‌شود.

برنامه کامل که از تابع قبل جهت تبدیل یک حرف کوچک به بزرگ استفاده می‌نماید.

```
#include  <iostream.h>
char low_to_up(char c1)
{
char  c2;
c2=(c1 >= ′ a ′ && c1 <= ′ z ′)?(′ A ′ +c1 -′ a ′) : c1;
return c2;
}
int main( )
{
char x;
x=cin.get( );
cout << low_to_up(x) ;
return 0;
 }
```



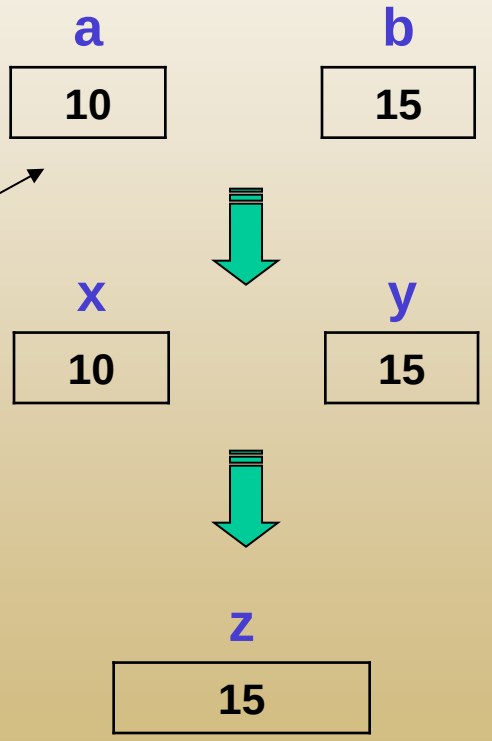
آرگومان

تابع maximum دو مقدار صحیح را گرفته بزرگترین آنها را برمیگرداند.

```
int   maximum(int x, int y)
{
int z ;
z=(x >= y)? x : y;
return z;
}
```

برنامه کامل که از تابع maximum جهت یافتن ماکزیمم دو مقدار صحیح استفاده می نماید.

```
#include  <iostream.h>
int   maximum(int x , int y)
{
int z ;
z=(x > y)? x : y ;
return z;
}
int main( )
{
int a, b ;
cin >> a >> b ;
cout << maximum(a,b);
return 0;
}
```



آرگومانهای تابع maximum a, b

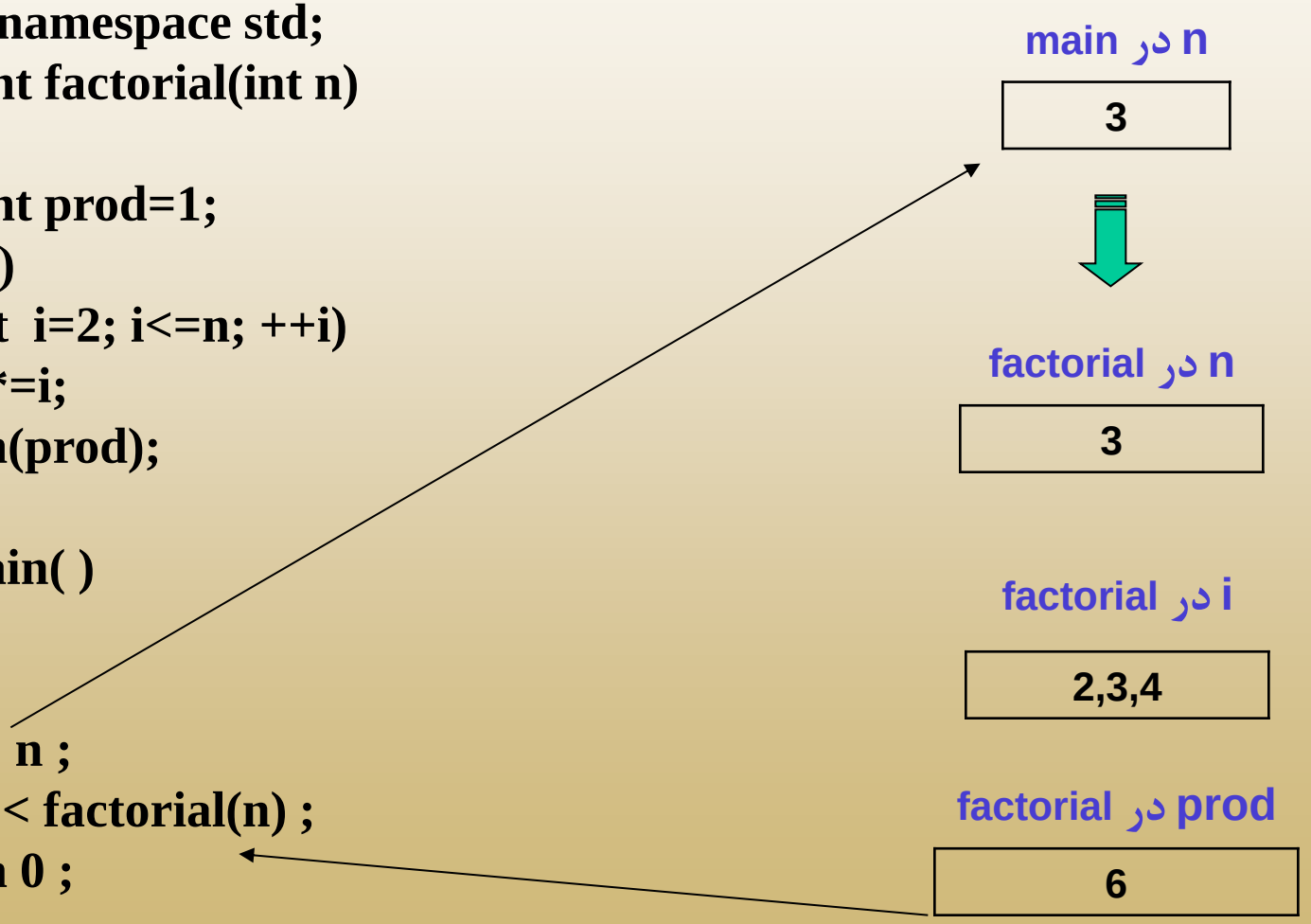
نکته ۱ :

اسامی پارامترها و آرگومانهای یک تابع می‌توانند همنام باشند.

برنامه زیر یک مقدار مثبت را گرفته فاکتوریل آنرا محاسبه نموده نمایش می‌دهد.

$$x!=1*2*3*4*...*(x-1)*x$$

```
#include  <iostream.h>
using namespace std;
long int factorial(int n)
{
long int prod=1;
if(n>1)
for(int  i=2; i<=n; ++i)
prod *=i;
return(prod);
}
int main( )
{
int n;
cin >> n ;
cout << factorial(n) ;
return 0 ;
}
```



نکته ۲ :

وقتی در تابعی، تابع دیگر احضار می‌گردد بایستی تعریف تابع احضار شونده قبل از تعریف تابع احضار کننده در برنامه ظاهر گردد.

نکته ۳ :

اگر بخواهیم در برنامه‌ها ابتدا تابع main ظاهر گردد بایستی تابع prototype یعنی پیش نمونهٔ تابع که شامل نام تابع، نوع مقدار برگشتی تابع، تعداد پارامترهائی را که تابع انتظار دریافت آنرا دارد و انواع پارامترها و ترتیب قرارگرفتن این پارامترها را به اطلاع کامپیلر برساند.

در اسلاید بعد مثالی در این زمینه آورده شده است.

مثالی از نکته ۲:

```
#include   <iostream.h>
#include   <conio.h>
long int factorial(int); // function prototype
int main( )
{
int n;
cout << "Enter a positive integer" << endl;
cin >> n;
cout << factorial(n) << endl;
return 0 ;
}
long int factorial(int n)
{
long int prod = 1;
if(n>1)
for(int i=2; i<=n; ++i)
prod *= i;
return(prod);}
```

نکته ۶ :

در صورتی که تابع مقداری بر نگرداند نوع مقدار برگشتی تابع را void اعلان می‌کنیم. ودر صورتیکه تابع مقداری را دریافت نکند بجای parameter- list از void یا ( ) استفاده میگردد.

در اسلاید بعد مثالی در این زمینه آورده شده است.

مثالی از نکته ۶:

```
#include<iostream.h>
void maximum(int,int) ;
int main( )
{ int x, y;
cin >> x >> y;
maximum(x,y);
return 0;
}
void maximum(int x, int y)
{
int z ;
z=(x>=y) ? x : y ;
cout << "max value \n" << z<< endl;
return ;
}
```

تابع مقداری را بر نمی گرداند.