الگوریتم و فلوچارت

(مبانی برنامه نویسی)

الگوریتم یک کلمه فارسی است که از دانشمند شهیر ایرانی، خوارزمی گرفته شده است. الگوریتم به معنی تشریح مراحل انجام کار است. مثلا الگوریتم مسواک زدن را در نظر بگیرید؛ ابتدا مسواک را برمی‌داریم، با خمیر دندان آغشته می‌کنیم، کمی خیس می‌کنیم، و آن‌را بروی دندان‌های خود می‌کشیم، در آخر دهان خود را شستشو می‌دهیم. همان‌طور که می‌بینید شما روزانه الگوریتم‌های زیادی را اجرا می‌کنید، البته به خاطر داشته باشید که در اینجا بیشتر روی الگوریتمهای ریاضی تمرکز خواهیم کرد. این الگوریتم‌ها بیشتر شبیه به یافتن راه‌حل گام به گام برای مسائل ریاضی است. مثلاً پیدا کردن کوچکترین عدد از میان تعدادی اعداد یا یافتن بزرگترین مقسوم علیه مشترک دو عدد.

الگوریتم مجموعه‌ای از **دستورالعمل‌ها** است که برای حل مسائل تعریف می شود. الگوریتم باید ویژگی های زیر را داشته باشد :

1- به زبان دقیق بیان شود.

2- جزئیات به طور کامل بیان گردد.

3- ترتیب مراحل الگوریتم مشخص باشد.

4- شرط پایان الگوریتم تعیین گردد.

به طور کلی هر الگوریتم از سه قسمت کلی تشکیل شده است که شامل **دریافت داده‌های ورودی مسئله ، انجام محاسبات** روی آنها و **نمایش نتایج در خروجی** است.

به طور دقیق‌تر مراحل طراحی الگوریتم عبارتند از :

1- ابتدا مسئله به خوبی درک شود.

2- ورودی‌های مسئله مشخص گردند.

3- خروجی‌های مسئله تعیین شوند.

4- روش های حل مسئله به صورت گام به گام یادداشت شود

5- از بین روش های یافته شده مناسب‌ترین روش انتخاب گردد.

6- با استفاده از داده‌های آزمایشی، الگوریتم اجرا شود.

7- با بررسی نتایج اجرای الگوریتم، اشکالات آن برطرف گردد.

الگوریتمی که معدل 3 درس دانشجو را محاسبه می‌نماید.

حل : مراحل طراحی الگوریتم باید به‌روی مسئله فوق اعمال گردد. بنابراین خواهیم داشت :

1. **نیازمندی‌های این مسئله معدل دانشجو است.**
2. **ورودی‌های مسئله نمره سه درس دانشجو است.**
3. **خروجی‌های مسئله معدل سه درس است.**
4. **برای حل مسئله نمره سه درس را با هم جمع می‌کنیم و حاصل را بر تعداد دروس تقسیم می‌کنیم. عدد حاصل معدل دانشجو است.**

اجرای الگوریتم

برای اجرای الگوریتم ابتدا دستور اول الگوریتم را می‌خوانیم و خواسته آن دستور را اجرا می‌کنیم، همین کار را برای دستورات بعدی به ترتیب اجرا کرده تا به پایان الگوریتم برسیم.

روشهای بیان الگوریتم

- **بیان الگوریتم به زبان فارسی**

در این روش دستورات الگوریتم به زبان فارسی نوشته شده و قبل از هر دستور شماره آن قرار می‌گیرد.

- **بیان ریاضی الگوریتم**

جهت رفع مشکل عدم خوانایی الگوریتم به زبان فارسی برای کامپیوتر، روش دیگری برای بیان الگوریتم مطرح شده است. این روش، بیان ریاضی الگوریتم است.

ویژگیهای روش ریاضی بیان الگوریتم

روش ریاضی یک رویه قانونمند برای نوشتن الگوریتم است که دارای ویژگی‌های زیر است :

- **تبدیل این شیوه بیان به برنامه کامپیوتری آسان است.**
- **الگوریتم با دستور شروع، آغاز می‌گردد و با دستور پایان، خاتمه می‌یابد.**
- **از نمادهایی برای نگهداری ورودی‌ها و خروجی‌ها استفاده می‌شود که به آنها نماد متغیر، گفته می‌شود.**
- **عبارات محاسباتی باید با فرمول‌های ریاضی بیان شوند. مثلا جمع دو عدد A و B به صورت A+B بیان می‌گردد.**
- **برای قرار دادن مقداری در متغیر از علامت انتساب (←) استفاده می‌گردد. مثلا A←10**
- **برای دریافت اطلاعات از عبارت "از ورودی بخوان" استفاده شود.**
- **برای نمایش اطلاعات از عبارت "در خروجی چاپ کن" استفاده می‌گردد.**

مثال : نوشتن الگوریتم با بیان ریاضی

الگوریتمی که نمره سه درس دانشجو را دریافت می‌نماید، سپس معدل دانشجو را محاسبه کرده و آن‌را در خروجی نمایش می‌دهد.

حل : ورودی‌های این مسئله، سه نمره دانشجو است. بنابراین باید ورودی‌ها را در متغیرها ذخیره کنیم. متغیر A را برای نمره اول، متغیر B را برای نمره دوم و متغیر C را برای نمره سوم در نظر می‌گیریم. همچنین برای نگهداری معدل از متغیر Avg استفاده می‌کنیم. حال باید الگوریتم را بنویسیم.

۱. شروع
۲. A ، B و C را از ورودی بخوان
۳. Avg←(A+B+C)/3
۴. Avg را چاپ کن.
۵. پایان

با اجرای الگوریتم بالا و قرار دادن نمره‌های دانشجو در متغیرها، معدل دانشجو محاسبه می‌گردد.

☐ عملگرهای ریاضی

به کمک این عملگرها می‌توان عملیات ریاضی را به‌روی اعداد و متغیرها انجام داد. در این کتاب از عملگرهای مختلفی استفاده می‌گردد که در جدول ۳-۱ بیان شده است.

عملگر	توضیح	مثال
+	جمع دو عدد	A+13
-	تفریق دو عدد	X-Y
/	تقسیم دو عدد برهم	X/4
*	ضرب دو عدد در هم	X*Y
Mod	باقیمانده تقسیم دو عدد بر هم	A Mod B
^	عددی به توان عددی دیگر	6^3
-	منهای عدد (تفریق یکاتی)	-A

آشنایی با فلوچارت

فلوچارت روش بیان الگوریتم توسط اشکال است. اشکال و نمادها باعث افزایش خوانایی دستورات می‌شوند و الگوریتم‌ها را واضح‌تر می‌کنند.

نماد شروع و پایان

این دو علامت به شکل بیضی هستند، بدین ترتیب از سایر نمادهای فلوچارت متمایز می‌شوند.

نماد دستورات محاسباتی و انتساب

مهمترین بخش الگوریتم دستورات محاسباتی و انتساب است. این دستورات به شکل مستطیل در فلوچارت ظاهر می‌شوند.

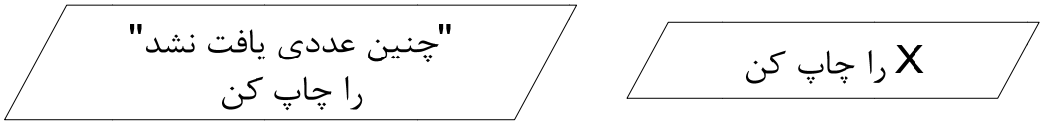
در نمادهای بالا دستور X ← 200 مقدار 200 را درون متغیر X قرار می‌دهد، همچنین Sum ← A + B حاصل جمع دو عدد A و B را در متغیر Sum قرار می‌دهد.

نماد دستورات خواندن از ورودی

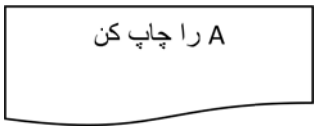
معمولاً پس از شروع الگوریتم، مقادیر به عنوان ورودی از کاربر دریافت می‌گردد. برای نمایش این دستورات از متوازی الاضلاع استفاده می‌گردد و در درون آن دستور ورودی قرار می‌گیرد.

نماد دستورات چاپ در خروجی

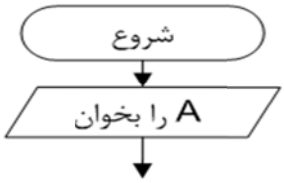
معمولاً قبل از اتمام الگوریتم، نتایجی در خروجی چاپ می‌گردد تا به کاربر نمایش داده شود. برای نمایش دستورات خروجی هم از متوازی الاضلاع استفاده می‌گردد و در داخل آن دستورات خروجی درج می‌شود.



گاهی اوقات از علامت زیر جهت چاپ کردن به‌روی کاغذ علاوه بر نماد بالا استفاده می‌شود.



نماد اتصال



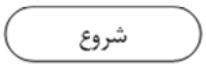
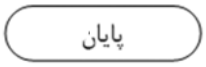
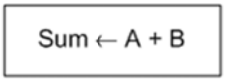
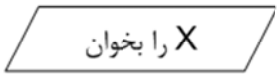
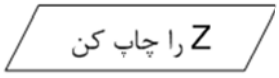
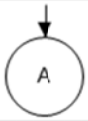
نماد ادامه

با زیاد شدن دستورات و استفاده مکرر از نماد اتصال، فلوچارت پیچیده می‌شود. برای رفع این مشکل از نماد ادامه استفاده می‌گردد. برای استفاده از این نماد ابتدا باید محل ادامه مانند شکل روبرو مشخص گردد.

تبدیل دستورات اصلی الگوریتم به فلوچارت و برعکس

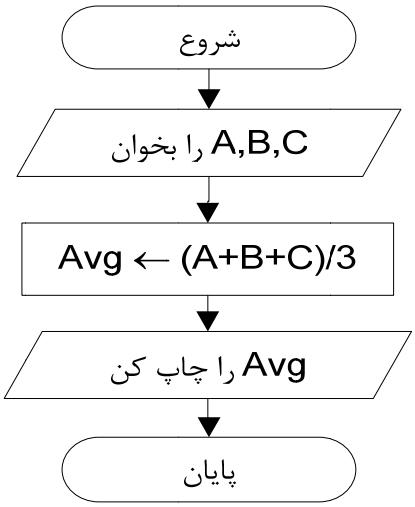
با جدول زیر می‌توانید الگوریتم‌های ساده را به فلوچارت تبدیل کنید و برعکس فلوچارت‌های ساده را به الگوریتم تبدیل نمایید.

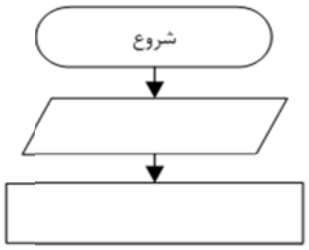
جدول ۳-۲ : تبدیل دستورات اصلی الگوریتم به فلوچارت و برعکس

نام دستور	مثالی از دستور در الگوریتم	نماد معادل دستور در فلوچارت
دستور شروع	شروع	
دستور پایان	پایان	
دستورات محاسباتی و انتساب	Sum ← A + B	
دستورات ورودی	X را بخوان	
دستورات خروجی	Z را چاپ کن	
ادامه اجرای الگوریتم	برو به A	

مثال: نوشتن فلوچارت به کمک الگوریتم

فلوچارتی رسم کنید که نمره سه درس دانشجو را دریافت نماید، سپس معدل دانشجو را محاسبه کرده و آن‌را در خروجی نمایش دهد.





بعضی الگوریتم‌ها ساده هستند و از سه قسمت اصلی تشکیل شده‌اند. این الگوریتم‌ها ابتدا داده‌هایی را دریافت می‌کنند، سپس محاسباتی را به‌روی آنها انجام می‌دهد و در آخر نتایج را چاپ می‌کنند. برای ایجاد این نوع الگوریتم‌ها ابتدا نماد شروع، نماد خواندن از ورودی و نماد محاسبات را رسم کنید و دستورات درون آنها را خالی بگذارید.

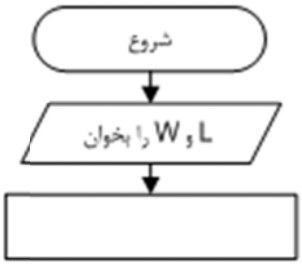
سپس ورودی‌ها و خروجی‌های مسئله را مشخص کنید و در جدول متغیرها یادداشت کنید. بعنوان مثال الگوریتمی را در نظر بگیرد که مساحت مستطیل را محاسبه می‌کند. این الگوریتم از این قاعده تبعیت می‌کند : مساحت مستطیل = طول × عرض

بنابراین ورودی‌های مسئله طول و عرض مستطیل و خروجی آن، مساحت مستطیل است.

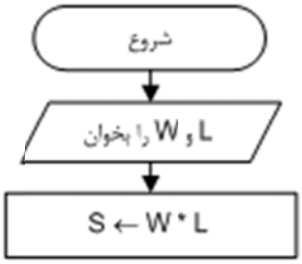
جدول متغیرها

طول	L	ورودی
عرض	W	ورودی
مساحت	S	خروجی

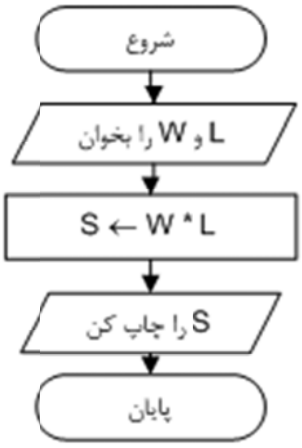
پس از تشکیل جدول متغیرها، خانه‌های خالی مربوط به دریافت ورودی را با متغیرهای ورودی جدول پر کنید. با یک دستور ورودی می‌توانید چندین متغیر را بخوانید و در متغیرها قرار دهید.



حال نوبت به قسمت دوم الگوریتم، یعنی محاسبات می‌رسد. الگوریتم‌های ساده فقط یک دستور محاسباتی دارند. این مثال هم به همین شکل است یعنی طبق فرمول محاسبه مساحت، طول در عرض ضرب می‌شود. البته برخی الگوریتمها شامل چندین بخش محاسباتی هستند، در این حالت به تعداد بخش‌های محاسباتی، نماد محاسبات در فلوچارت زیر هم رسم می‌گردد و در هر خانه یک بخش از محاسبات صورت می‌پذیرد.



در گام آخر نماد خروجی و پایان را رسم کرده و در داخل نماد خروجی، متغیرهای خروجی را چاپ نمایید.



به این ترتیب فلوچارت تکمیل می‌گردد.

مثال: الگوریتم تعویض مقدار دو متغیر به کمک متغیر کمکی

الگوریتمی که دو مقدار را از ورودی خوانده و درون متغیر A و B قرار می‌دهد. سپس مقادیر دو متغیر را با هم عوض کرده و در خروجی آنها را چاپ می‌کند.

دید الگوریتمی : یک لیوان نوشابه سیاه و یک لیوان نوشابه زرد داریم، چطور می توانیم این نوشابه‌ها را جابجا کنیم؟ حتما به فکر یک لیوان خالی دیگر می‌افتید تا ابتدا نوشابه سیاه را در آن بریزید. حال که لیوان نوشابه سیاه خالی شد، نوشابه زرد را درون آن ریخته و در گام آخر نوشابه سیاهی که در لیوان کمکی قرار دارد را در لیوانی که نوشابه زرد قبلاً در آن قرار داشت، بریزید. به این ترتیب محتویات این دو لیوان با استفاده از یک لیوان کمکی تعویض شده‌اند. در این مثال هم از این تکنیک استفاده می‌کنیم. ابتدا مقدار اول را خوانده و در A قرار می‌دهیم، سپس مقدار دوم را خوانده و در متغیر B قرار می‌دهیم. در گام بعد یک متغیرکمکی به نام T می‌گیریم و مقدار متغیر A را درون آن منتقل می‌کنیم، سپس مقدار B را درون متغیر A قرار می‌دهیم و در گام آخر مقدار T را درون B قرار می‌دهیم.

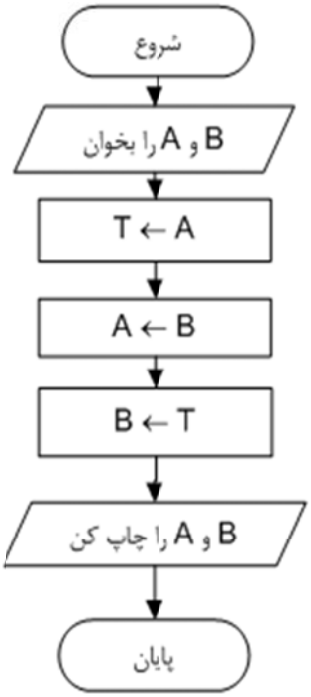
جدول متغیرها

عدد اول	A	ورودی/خروجی
عدد دوم	B	ورودی/خروجی
متغیر کمکی	T	کمکی

الگوریتم

۱. شروع
۲. A و B را بخوان.
۳. T ← A
۴. A ← B
۵. B ← T
۶. A و B را چاپ کن
۷. پایان

فلوچارت



بسیاری از دستورات الگوریتم را عبارت‌های محاسباتی شکل می‌دهند. اگر این عبارت‌ها طولانی شوند تعیین مقدار حاصل دشوار خواهد شد. بعنوان مثال دستور زیر را در نظر بگیرید :

$$X \leftarrow 5 \times 3 + 12 / (8 - 4)$$

با اجرای این دستور چه مقداری در متغیر X قرار خواهد گرفت؟ پاسخ سوال به این نکته بستگی دارد که عملگرهای محاسباتی را با چه اولویتی اعمال کنیم. یعنی اگر از سمت چپ ابتدا ضرب، سپس جمع و بعد تقسیم و تفریق را اعمال کنیم نتیجه کسب شده متفاوت خواهد بود با حالتی که از سمت راست عملگرها را اعمال کنیم. برای رفع این گونه ابهامات از قواعد زیر در تعیین نتیجه عبارت محاسباتی استفاده می‌شود :

- اولویت اول با عبارت‌های داخل پرانتز است.
- اولویت دوم با ضرب و تقسیم است (در صورت وجود عملگر ضرب و تقسیم در کنار هم، عملگری که سمت چپ قرار دارد از اولویت بالاتری برخوردار است.)
- اولویت آخر با جمع و تفریق است (در صورت وجود عملگر جمع و تفریق در کنار هم، عملگری که سمت چپ قرار دارد از اولویت بالاتری برخوردار است.)

با قواعد بالا نتیجه محاسبات بدین صورت خواهد شد :

$$X \leftarrow 5 \times 3 + 12 / (8 - 4)$$
$$\downarrow$$
$$X \leftarrow 5 \times 3 + 12 / 4$$
$$\downarrow$$
$$X \leftarrow 15 + 12 / 4$$
$$\downarrow$$
$$X \leftarrow 15 + 3$$
$$\downarrow$$
$$X \leftarrow 18$$

مثال: الگوریتم محاسبه یک عبارت ریاضی به کمک قوانین تقدم عملگرها

الگوریتمی که دو مقدار را از ورودی خوانده و در متغیرهای X و Y قرار می‌دهد. سپس با استفاده از آنها حاصل عبارت زیر را در خروجی چاپ می‌کند.

$$Z = \frac{X + XY}{X^2 + Y^2}$$

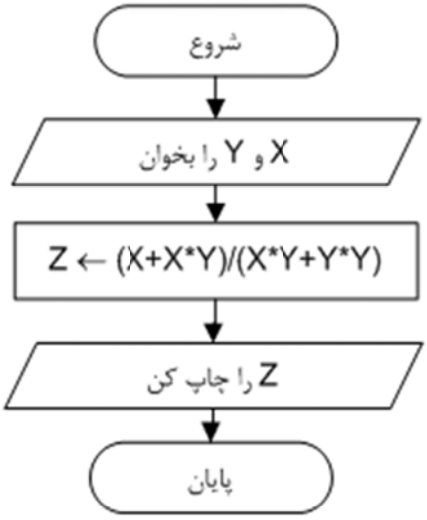
حل : ابتدا دو مقدار را از ورودی خوانده و در X و Y قرار می‌دهیم. عبارت محاسباتی بالا را با پرانتز گذاری مناسب تبدیل به عبارت محاسباتی قابل قبول الگوریتم می‌کنیم؛ طبق دستور زیر :

$$Z = (X + X \times Y) / (X \times X + Y \times Y)$$

جدول متغیرها

عدد اول	X	ورودی/خروجی
عدد دوم	Y	ورودی/خروجی
متغیر کمکی	T	کمکی

فلوچارت



الگوریتم

۱. شروع
۲. X و Y را بخوان
۳. Z ← (X+X*Y)/(X*Y+Y*Y)
۴. Z را چاپ کن
۵. پایان

دستورات شرطی

الگوریتمی که دو مقدار را از ورودی خوانده و در متغیرهای X و Y قرار می‌دهد. سپس با استفاده از آنها حاصل عبارت زیر را در خروجی چاپ می‌کند.

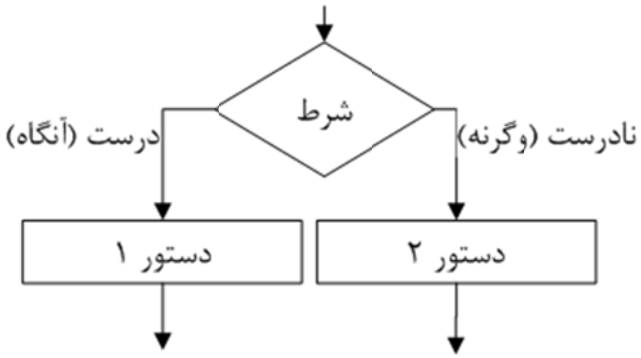
الگوریتم‌هایی که تا به‌حال بیان شد کاملاً سر راست و مستقل از شرایط برنامه بودند. ولی گاهی اوقات الگوریتم با توجه به شرایط، دستورات خاصی را اجرا می‌کند. بعنوان مثال برای تعیین علامت یک عدد می‌گوییم : *اگر عدد بزرگتر از صفر بود، مثبت را در خروجی چاپ کن، اگر عدد کوچکتر از صفر بود، منفی را در خروجی چاپ کن، وگرنه صفر را در خروجی چاپ کن.* دستورات شرطی در الگوریتم بدین شکل بیان می‌شود :

اگر شرط آنگاه دستور

در صورتی‌که شرط دو حالته باشد، یعنی در صورت غلط بودن شرط باید دستور دیگری انجام شود از دستور شرطی زیر استفاده می‌شود :

اگر شرط آنگاه دستور ۱ وگرنه دستور ۲

نماد فلوچارت دستور شرطی فوق بدین شکل است.



مثال :الگوریتم تعیین علامت اعداد

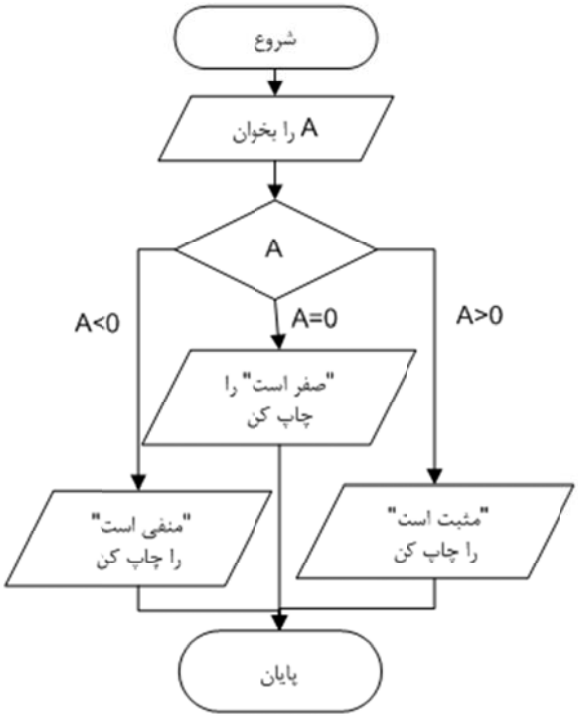
الگوریتمی که یک عدد را از ورودی خوانده و آن‌را تعیین علامت می‌کند.

حل : این الگوریتم از دستور شرطی سه حالته استفاده می‌کند تا تعیین علامت را انجام دهد. به دستورات خروجی دقت کنید که مثلاً جمله "مثبت است" در داخل دابل کتیشن محصور شده است، این کار باعث می‌شود که این جمله عیناً در خروجی چاپ شود.

جدول متغیرها

عدد مورد تعیین علامت	A	ورودی

فلوچارت

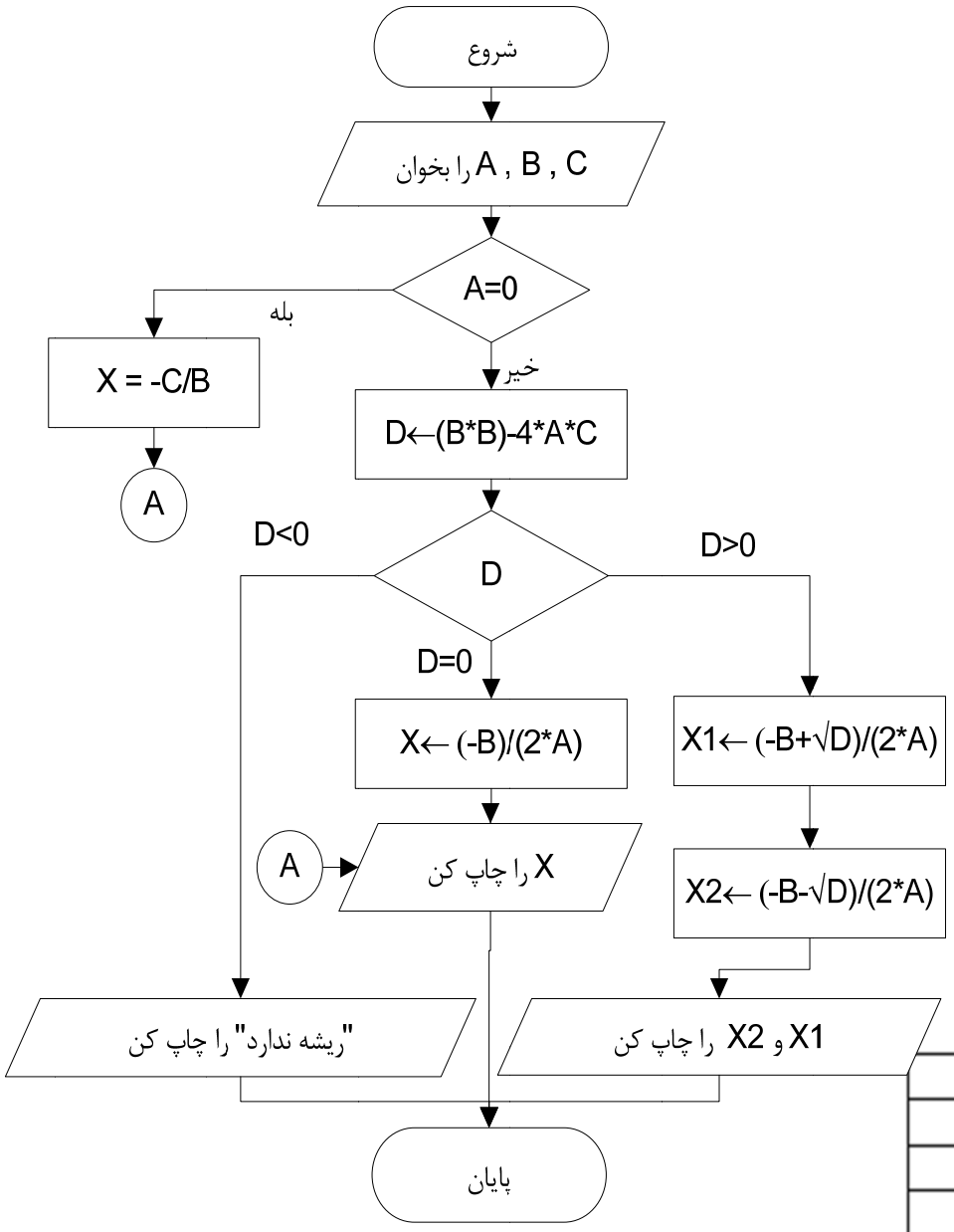


الگوریتم

۱. شروع
۲. A را بخوان.
۳. اگر A>0 آنگاه چاپ کن "مثبت است"
وگرنه، اگر A<0 آنگاه چاپ کن "منفی است"
وگرنه چاپ کن "صفر است"
۴. پایان

مثال: الگوریتم تعیین جواب‌های معادله درجه دوم

الگوریتمی که ضرایب یک معادله درجه دوم را دریافت کرده و جواب‌های آن‌را تعیین می‌نماید.



جدول متغیرها

ضریب x^2	A	ورودی
ضریب x	B	ورودی
عدد ثابت معادله	C	ورودی
دلتا	D	محاسباتی
ریشه اول	X1	خروجی
ریشه دوم	X2	خروجی
ریشه مضاعف	X	خروجی

آزمایش الگوریتم (اثبات درستی الگوریتم)

اثبات درستی الگوریتم را آزمایش الگوریتم، ردگیری اجرای الگوریتم (Trace) و بررسی صحت اجرای الگوریتم نیز می‌گویند.

مراحل انجام این کار به‌روی مثال قبل (تعیین جواب معادله درجه دوم) بررسی می‌شود. مراحل آزمایش الگوریتم بدین شرح است

1. تشکیل جدول مقادیر متغیرها:

ضریب x^2	A	ورودی	
ضریب x	B	ورودی	
عدد ثابت معادله	C	ورودی	
دلتا	D	محاسباتی	
ریشه اول	X1	خروجی	
ریشه دوم	X2	خروجی	
ریشه مضاعف	X	خروجی	

2. دستورات ورودی :

ضریب x^2	A	ورودی	1
ضریب x	B	ورودی	5
عدد ثابت معادله	C	ورودی	6
دلتا	D	محاسباتی	
ریشه اول	X1	خروجی	
ریشه دوم	X2	خروجی	
ریشه مضاعف	X	خروجی	

3- دستورات شرطی : در این دستورات ابتدا شرط را بررسی کنید و در صورت درستی بخش آنگاه شرط را اجرا نمایید. در صورت نادرست بودن شرط ،دستور خط بعد را اجرا کنید.

4- دستورات محاسباتی :

D← (5*5)-4*1*6 = 25- 24 = 1

ضریب x^2	A	ورودی	1
ضریب x	B	ورودی	5
عدد ثابت معادله	C	ورودی	6
دلتا	D	محاسباتی	1
ریشه اول	X1	خروجی	
ریشه دوم	X2	خروجی	
ریشه مضاعف	X	خروجی	

5- تکمیل جدول مقادیر متغیرها :

دستورات بعدی الگوریتم را اجرا کرده و هر تغییری را در جدول مقادیر متغیرها اعمال می‌کنیم.

ضریب x^2	A	ورودی	1
ضریب x	B	ورودی	5
عدد ثابت معادله	C	ورودی	6
دلتا	D	محاسباتی	1
ریشه اول	X1	خروجی	-3
ریشه دوم	X2	خروجی	-2
ریشه مضاعف	X	خروجی	

دستورات خروجی :

متغیرهایی که باید در خروجی چاپ شوند را در جدول مقادیر متغیرها با کشیدن دایره دور آنها مشخص می‌کنیم.

ضریب x^2	A	ورودی	1
ضریب x	B	ورودی	5
عدد ثابت معادله	C	ورودی	6
دلتا	D	محاسباتی	1
ریشه اول	X1	خروجی	-3
ریشه دوم	X2	خروجی	-2
ریشه مضاغف	X	خروجی	

بررسی نتایج : متغیرهای خروجی الگوریتم را با مقادیر ورودی مقایسه کرده و صحت اجرای الگوریتم را بررسی می‌کنیم.

$$A X^2 + B X + C = 0 \qquad A = 1 , B = 5 , C = 6 ,$$

$$D = B \times B - 4 \times A \times C = 5 \times 5 - 4 \times 1 \times 6 = 25 - 24 = 1$$

$$X1 = \frac{-B - \sqrt{D}}{2A} = \frac{-5 - \sqrt{1}}{2 \times 1} = -3$$

$$X2 = \frac{-B + \sqrt{D}}{2A} = \frac{-5 + \sqrt{1}}{2 \times 1} = -2$$

حلقه‌های تکرار

حلقه‌های تکرار راهکاری برای انجام دستورات تکراری در الگوریتم است.

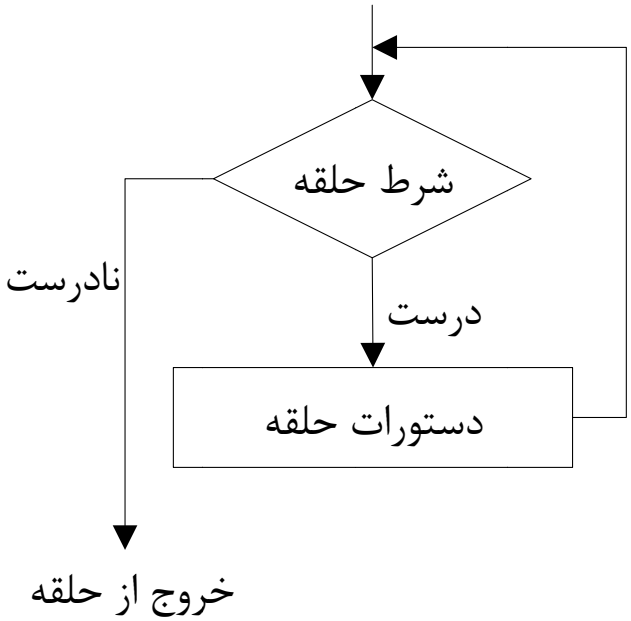
برای این‌کار باید چهار چیز را مشخص کنیم :

1. **شمارنده حلقه تکرار : متغیری است که مشخص می‌کند تاکنون چند بار حلقه اجرا شده‌است.**
2. **مقداردهی اولیه شمارنده حلقه : قبل از آغاز حلقه، شمارنده مقدار دهی می‌شود.**
3. **گام افزایش شمارنده حلقه : پس از هر بار اجرای حلقه، شمارنده باید افزایش یابد. یعنی I ← I + 1**
4. **شرط حلقه تکرار : تعیین می‌کند که چند بار باید دستورات تکرار شوند. مثلا شرط I<=N**

نحوه ایجاد حلقه‌های تکرار

به دو صورت می‌توان حلقه تکرار را ایجاد کرد.

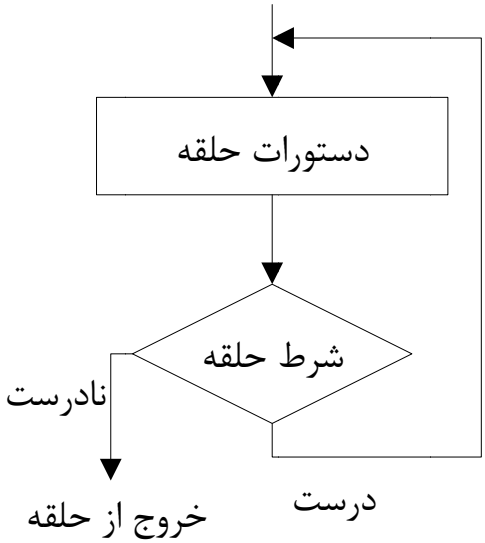
روش اول: ابتدا شرط حلقه بررسی می‌گردد، سپس دستورات حلقه اجرا می‌شوند.



الگوریتم

۱. .....
۲. تا زمانی‌که شرط حلقه درست است دستورات ۳ تا ۴ را اجرا کن
۳. دستورات حلقه
۴. .....
۵. پایان حلقه (شروع از دستور ۲)
۶. ادامه دستورات الگوریتم
۷. .....

روش دوم: ابتدا دستورات حلقه اجرا می‌شوند و سپس شرط حلقه بررسی می‌شود.



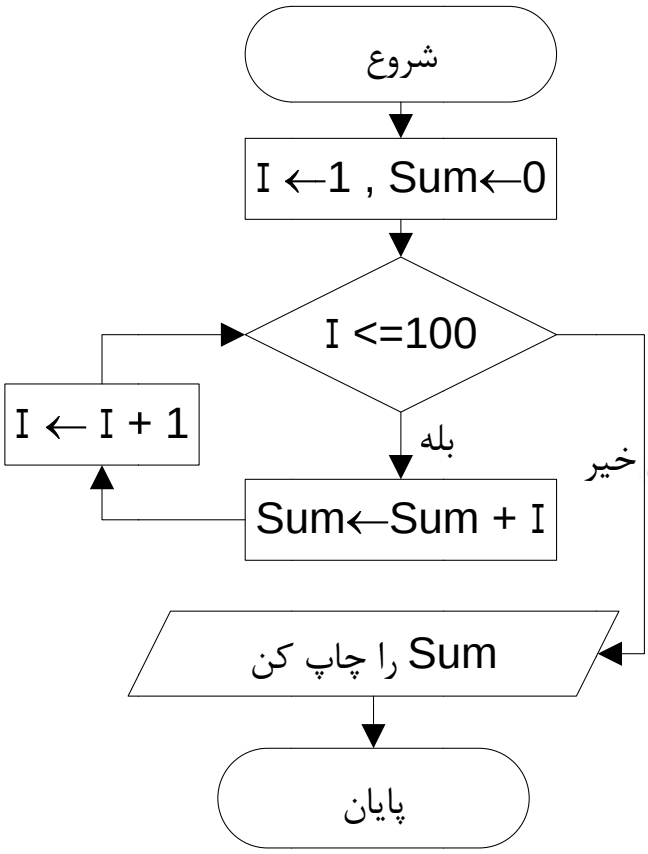
الگوریتم

۱. .....
۲. دستورات حلقه
۳. .....
۴. تا زمانی‌که شرط حلقه درست است دستورات ۲ تا ۴ را اجرا کن
۵. ادامه دستورات الگوریتم
۶. .....

مثال : الگوریتم مجموع اعداد 1 تا 100

الگوریتمی که مجموع اعداد 1 تا 100 را محاسبه کرده و چاپ می‌نماید.

دید الگوریتمی : برای جمع اعداد 1، 2، ... ، 100 از یک حلقه استفاده می‌شود. بنابراین چهار مورد اصلی حلقه باید مشخص گردد. نام شمارنده را I در نظر می‌گیریم. چون می‌خواهیم از عدد 1 شروع به جمع اعداد کنیم مقدار اولیه شمارنده را برابر با 1 قرار می‌دهیم. گام افزایش را نیز 1 تعیین می‌کنیم چون باید همه اعداد 1 تا 100 را با هم جمع کنیم. اگر قرار بود اعداد فرد را با هم جمع کنیم گام افزایش را 2 در نظر می‌گرفتیم زیرا با این کار مقدار Iدو واحد دو واحد زیاد می‌شد و اعداد فرد را پوشش می‌داد. شرط حلقه را I<=100 تعریف می‌کنیم تا هر 100 عدد را با هم جمع کند.



جدول متغیرها

شمارنده حلقه	I	شمارنده
مجموع اعداد	Sum	خروجی

الگوریتم

۱. شروع
۲. I ← 1
۳. Sum ← 0
۴. تا زمانی‌که I<=100 است، دستورات ۴ تا ۵ را تکرار کن.
۵. Sum ← Sum + I
۶. I ← I + 1
۷. پایان حلقه (شروع از دستور ۴)
۸. Sum را چاپ کن.
۹. پایان

مثال: الگوریتم محاسبه مجموع N عدد

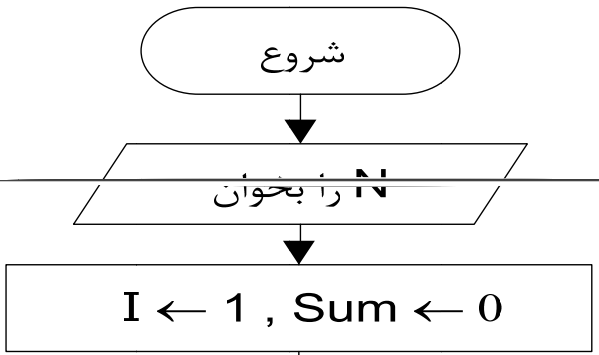
الگوریتمی که N بار اعدادی را از ورودی خوانده، مجموع آنها را محاسبه کرده و در خروجی چاپ می‌کند.

جدول متغیرها

عدد خوانده شده	X	ورودی
شمارنده حلقه	I	شمارنده
مجموع اعداد	Sum	خروجی
تعداد اعداد	N	ورودی

الگوریتم

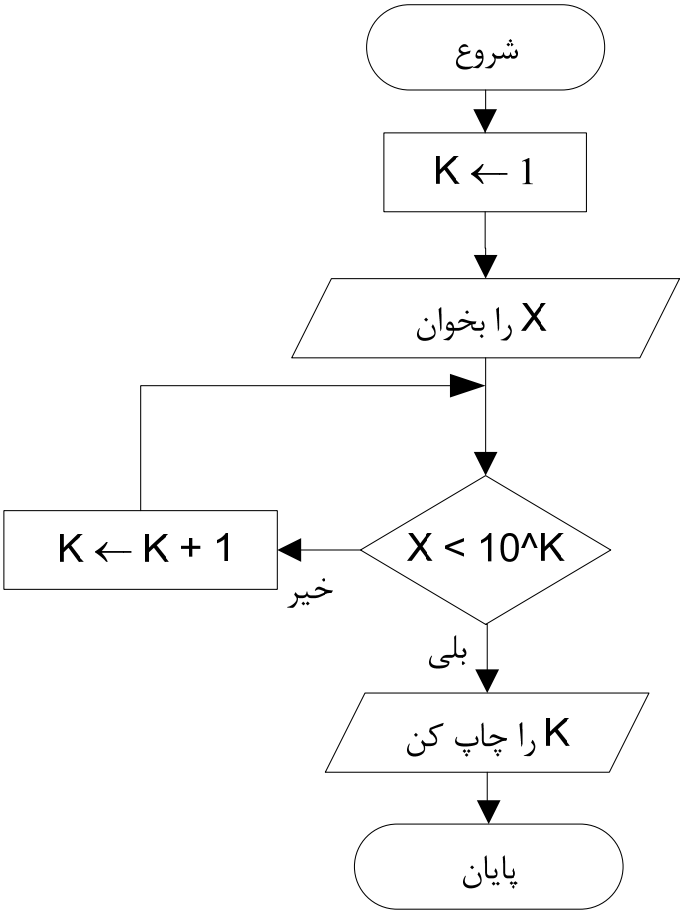
۱. شروع
۲. N را بخوان.
۳. SUM ←0 , I ←1
۴. تا زمانی که I<= N است، دستورات ۴ تا ۶ را تکرار کن.
۵. X را بخوان
۶. SUM ←X + SUM
۷. I ← I + 1
۸. پایان حلقه
۹. SUM را چاپ کن
۱۰. پایان



مثال: الگوریتم محاسبه تعداد ارقام یک عدد

الگوریتمی که عددی را از ورودی خوانده و تعداد ارقام آن‌را در خروجی چاپ می‌کند.

دید الگوریتمی : اگر عدد یک رقمی باشد از 10^1کوچکتر است، اگر دو رقمی باشد از 10^2 کوچکتر است و به همین ترتیب الی آخر. از ایده فوق برای تعیین تعداد ارقام عدد استفاده می‌گردد. در یک حلقه تکرار هر بار عدد مورد نظر با عدد 10 به توان شمارنده حلقه، مقایسه می‌شود، اگر عدد بزرگتر بود یک واحد به شمارنده اضافه شده و مجدداً مقایسه صورت می‌گیرد وگرنه شمارنده حلقه به عنوان تعداد ارقام عدد در خروجی چاپ می‌گردد.



حلقه‌های تکرار تو در تو

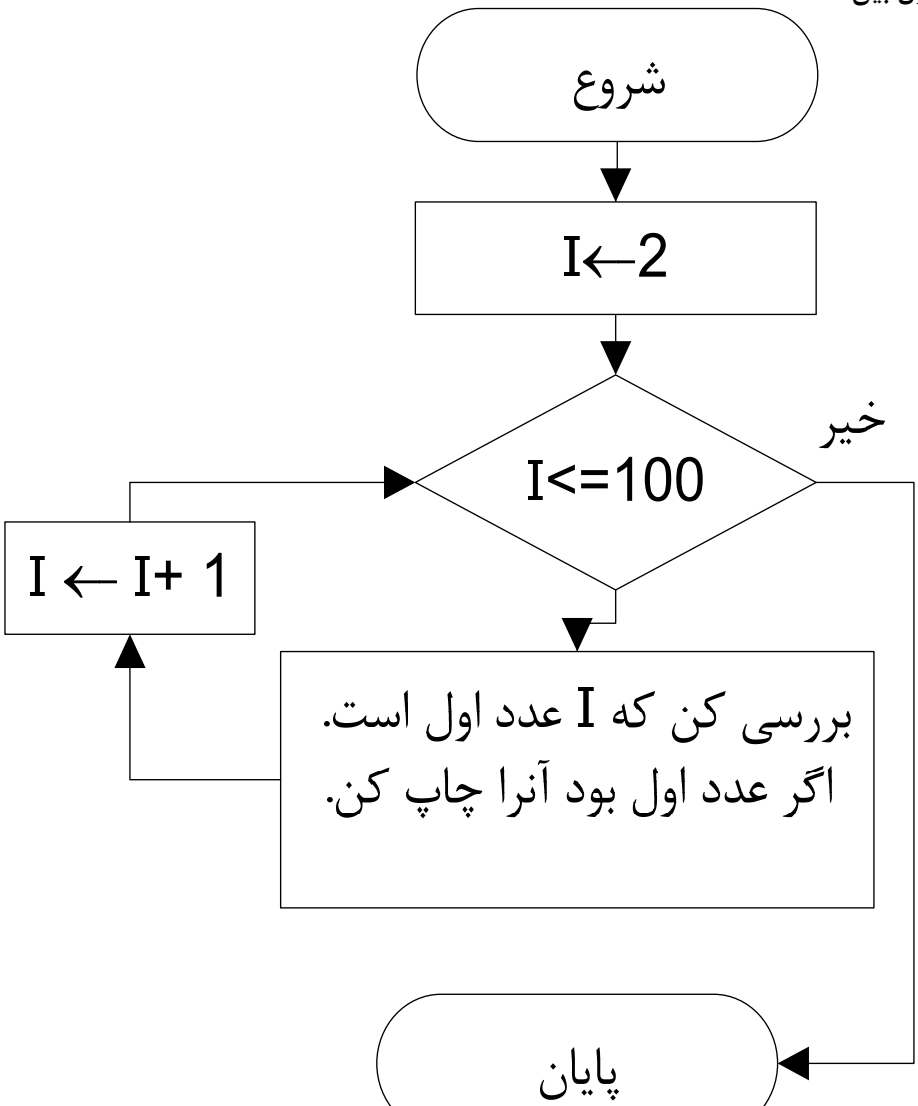
حلقه‌های تکرار تودرتو به حالتی گفته می‌شود که در الگوریتمی یک حلقه تکرار درون حلقه تکرار دیگری قرار داشته باشد.

حلقه‌های تکرار تو در تو، کاربردهای زیادی در طراحی الگوریتم‌های مرتب سازی، جستجو و بهینه سازی دارند.

دید Top-Down : یک روش مفید برای نوشتن الگوریتمهای پیچیده

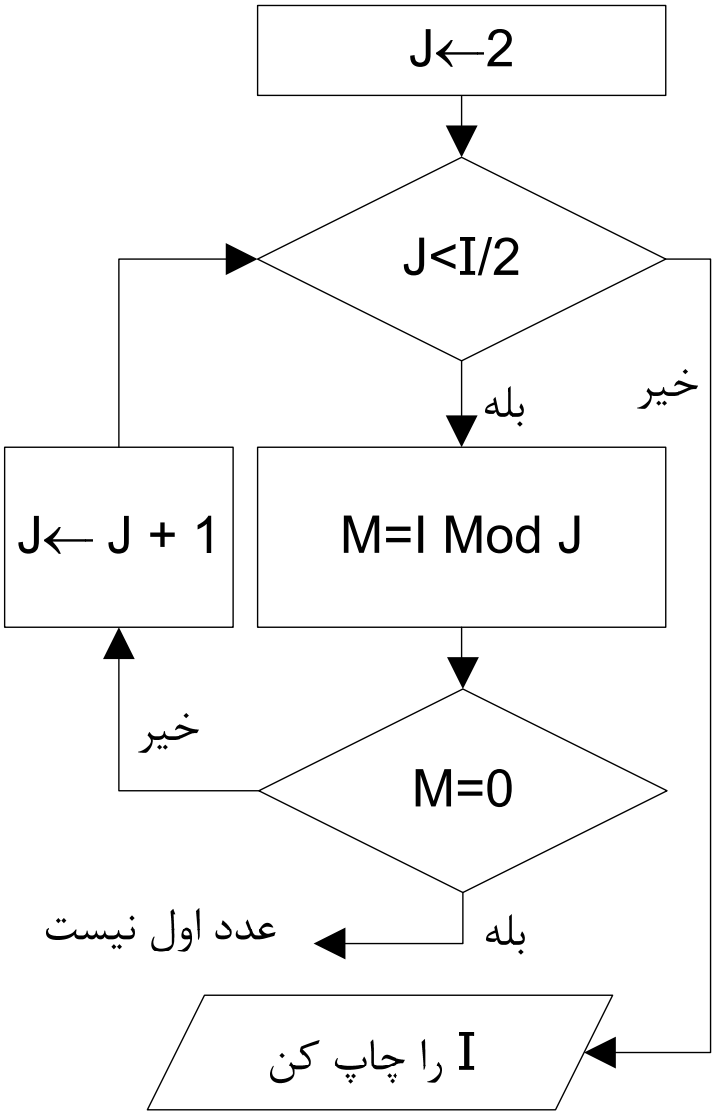
برای این‌که یک برنامه نویس حرفه‌ای شوید باید دید انتزاعی به مسائل داشته باشید. منظور از دید انتزاعی، طراحی کل به جز یا (Top-down) است.

مثال: الگوریتم تعیین اعداد اول بین 2 تا 100

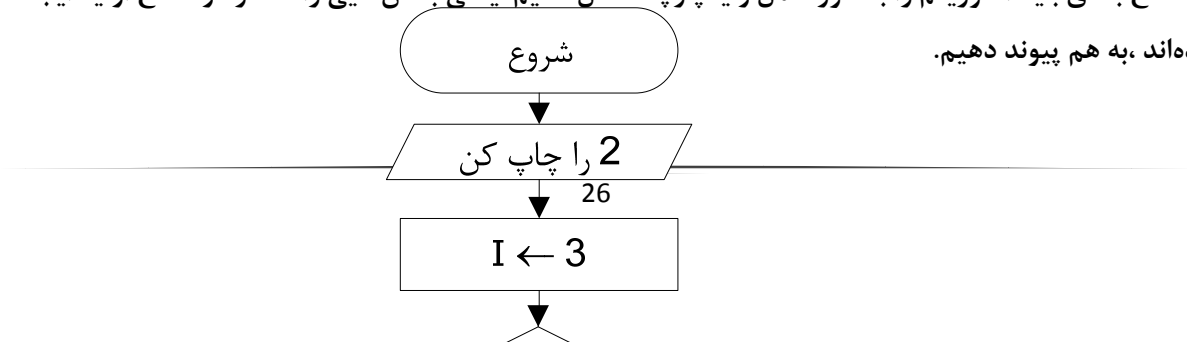


همان‌طور که مشاهده می‌کنیم **الگوریتم کلی بدون در نظر گرفتن جزئیات** گرفته است. این شیوه طراحی الگوریتم دو مزیت مهم دارد :

- **در ابتدا دست به کار می‌شوید و حداقل تفکر کلی خود نسبت به مسئله را روی کاغذ می‌آورید.**
- **پی خواهید برد که کدام بخش الگوریتم پیچیده‌تر است.**

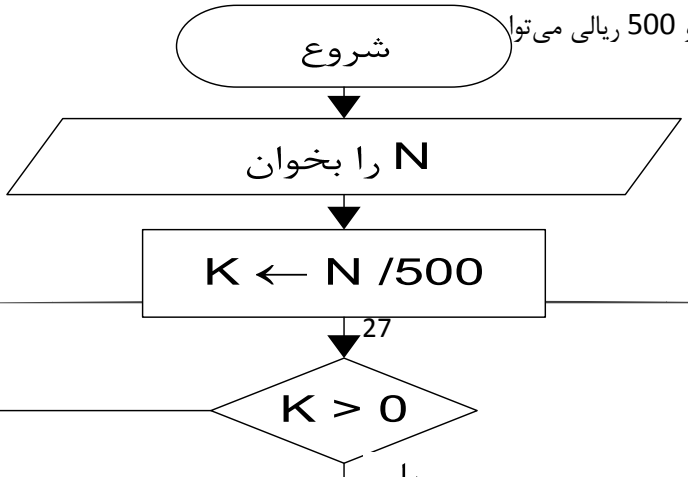


در سطح بعدی باید الگوریتم را به طور کامل و یکپارچه شکل دهیم. یعنی بخش‌هایی را که در دو سطح اولیه ایجاد شده‌اند ،به هم پیوند دهیم.



مثال: خرد کردن پول با سکه‌های 1، 250 و 500 ریالی

الگوریتمی که عددی را به عنوان پول از ورودی دریافت می‌کند، سپس آن‌را طوری خرد می‌کند طوری که تعداد سکه ها حداقل باشد. فقط از سکه‌های 1، 250 و 500 ریالی می‌توا



مثال: بررسی کارایی دو الگوریتم برای یک مسئله واحد

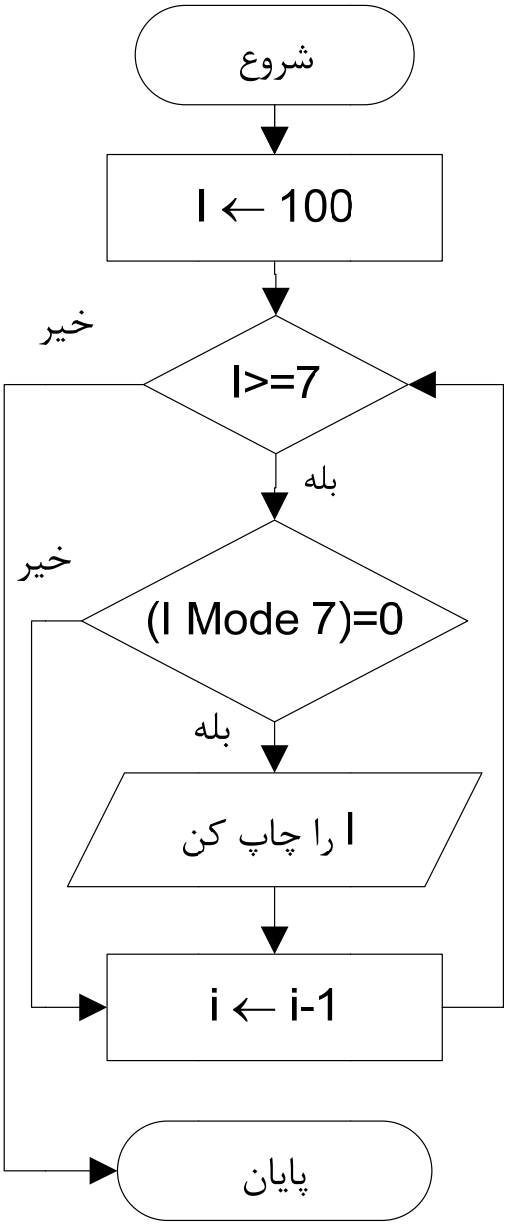
الگوریتمی که با دو روش مختلف فهرست اعداد مضرب 7 و کمتر از 100 را چاپ می‌کند و کارایی هر دو روش را بررسی می‌کند.

حل : این الگوریتم به دو روش قابل حل است :

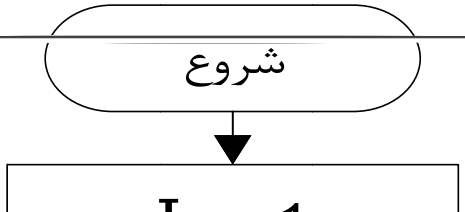
روش اول : در یک حلقه تکرار، شمارنده I با ۱۰۰ مقدار دهی می‌شود. در هر بار اجرای حلقه باقیمانده تقسیم I بر ۷ محاسبه شده و در صورتی‌که صفر باشد آن عدد در خروجی چاپ می‌گردد و در اجرای بعدی حلقه یک واحد از I کم شده تا به ۷ می‌رسد.

جدول متغیرها

شمارنده حلقه	I	شمارنده



روش دوم : می‌دانیم که با حاصل‌ضرب عدد ۷ در اعداد ۱ ، ۲ ، ۳ و ... می‌توانیم مضرب‌های ۷ را بسازیم.
با این ایده الگوریتمی می‌نویسیم که مسئله فوق را حل کند.



مقایسه کارایی دو الگوریتم :

زمان اجرای روش نخست = 93+93+15+93 = **294**

زمان اجرای روش دو = 15+15+15= **45**

زیرالگوریتم

پیچیدگی برخی از الگوریتم‌ها باعث می‌شود که نوشتن آنها در غالب یک الگوریتم دشوار شده و خوانایی آنها مشکل گردد. در مثال های قبل، الگوریتم های کوچک و متوسط مورد بررسی قرار گرفتند؛ ولی وقتی که الگوریتمی بیش از 50 دستور داشته باشد

نوشتن آن در یک صفحه دشوار است. حتی اگر از دستور انتقال کنترل اجرای الگوریتم استفاده گردد. برای رفع این مشکل زیرالگوریتم ها معرفی شده‌اند. زیرالگوریتم امکان تقسیم بندی الگوریتم را به بخش‌های کوچکتر فراهم می‌کند.

معرفی زیرالگوریتم

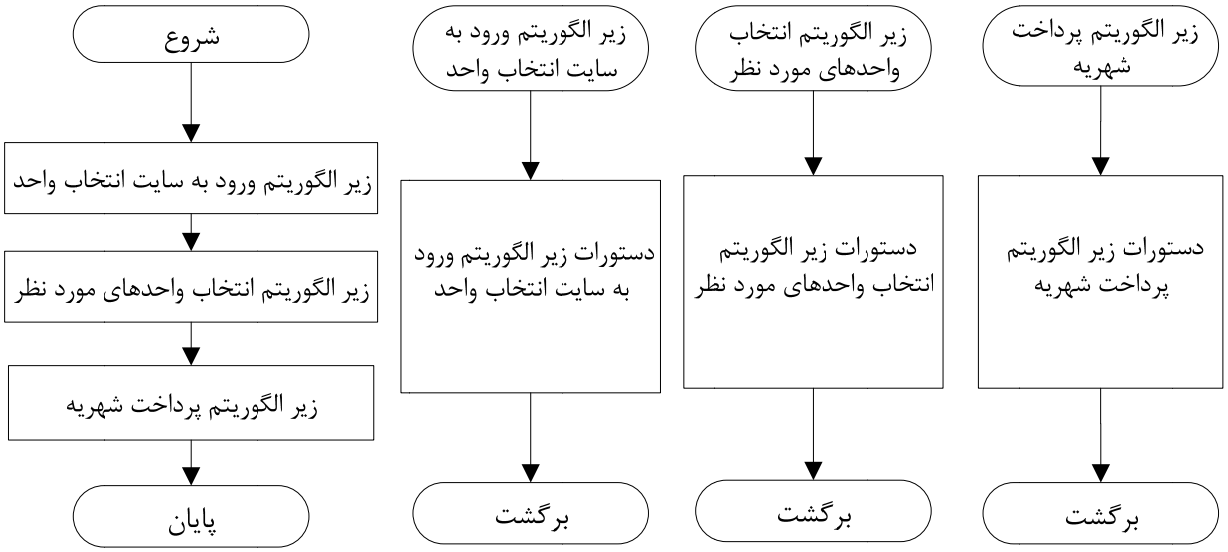
زیرالگوریتم، بخشی از الگوریتمی اصلی است که به صورت جداگانه نوشته می‌شود و توسط الگوریتم اصلی فراخوانی می‌شود.

کاربرد زیرالگوریتم

برای آشنایی با کاربرد زیرالگوریتم‌ها، مسئله انتخاب واحد دانشجو را در نظر بگیرید. این الگوریتم شامل چند مرحله است که عبارتند از :

- ورود به سایت انتخاب واحد
- انتخاب واحدهای مورد نظر
- پرداخت شهریه

سه مرحله فوق دارای الگوریتم‌های خاص خود هستند و در کنار هم الگوریتم اصلی انتخاب واحد را شکل می‌دهند. بنابراین آنها را به سه زیرالگوریتم تقسیم کرده و با نوشتن یک الگوریتم کلی، آنها را به هم پیوند می‌دهیم. همان طور که در شکل ۵-۱ نشان داده شده است.



مزایای زیرالگوریتم

زیرالگوریتم‌ها باعث افزایش خوانایی برنامه شده و پیچیدگی آن را کاهش می‌دهد. سایر مزایای زیرالگوریتم‌ها عبارتند از :

- کوچک کردن حجم الگوریتم ها

- همکاری افراد در نوشتن الگوریتم
- تسهیل رفع اشکال الگوریتم

انواع زیرالگوریتم

اکثر زیرالگوریتم‌ها، محاسباتی را انجام داده و نتیجه ای را به الگوریتم اصلی برمی‌گردانند. از لحاظ نحوه برگشت دادن مقدار، زیرالگوریتم ها به دو دسته تقسیم می‌شوند.

- **زیرالگوریتم های تابع (function)** : فقط یک مقدار را به الگوریتم اصلی برگشت می‌دهد. مزیت زیر الگوریتم‌های تابع، قابلیت انتساب آنها در عبارات محاسباتی است که در ادامه فصل مورد بررسی قرار می‌گیرد.
- **زیرالگوریتم های زیر روال (subroutine)** : این الگوریتم ها می‌توانند هیچ مقداری را برگشت ندهند و یا بیش از یک مقدار را به تابع اصلی برگشت دهند. از زیر روال‌ها در عبارات محاسباتی نمی‌توان استفاده کرد.

نحوه تعریف و فراخوانی زیرالگوریتم

در تعریف زیرالگوریتم، نام متغیرهای ورودی و دستورات آن نوشته می‌شوند.

پس از تعریف ، در الگوریتم اصلی برنامه باید از زیرالگوریتم استفاده نمود که اصطلاحاً به این عمل فراخوانی زیر الگوریتم گفته می شود.

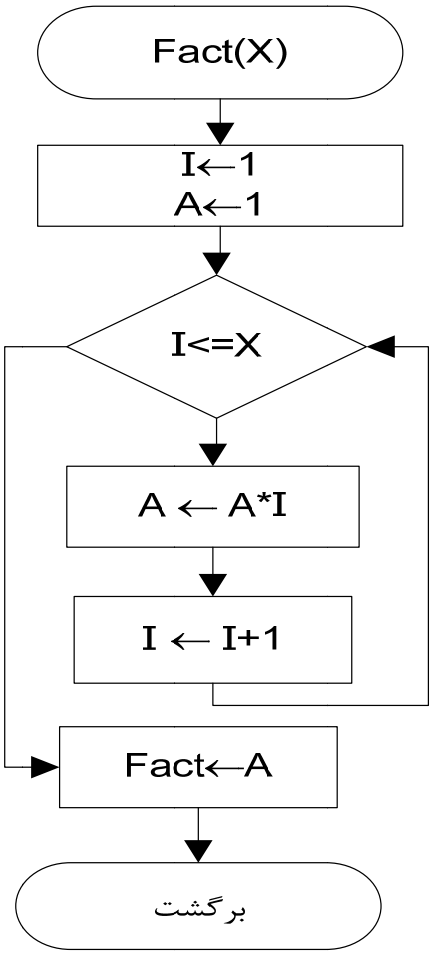
تعریف زیرالگوریتم

هر زیرالگوریتم با یک نام مشخص می‌گردد که معرف هدف آن است. مثلاً زیرالگوریتمی که عمل فاکتوریل را انجام می‌دهد با نام Fact تعریف می‌گردد.

مثال: زیرالگوریتم فاکتوریل یک عدد

زیرالگوریتمی که فاکتوریل یک عدد را محاسبه می‌کند.

حل : زیرالگوریتم فاکتوریل یک تابع است زیرا فاکتوریل یک عدد را برگشت می‌دهد. این مقدار با دستور Fact ← F به تابع اصلی برگشت داده می‌شود. در شکل زیر دستورات الگوریتم و نمادهای فلوچارت را برای زیرالگوریتم فاکتوریل مشاهده می‌کنید.



الگوریتم

زیرالگوریتم Fact(X)

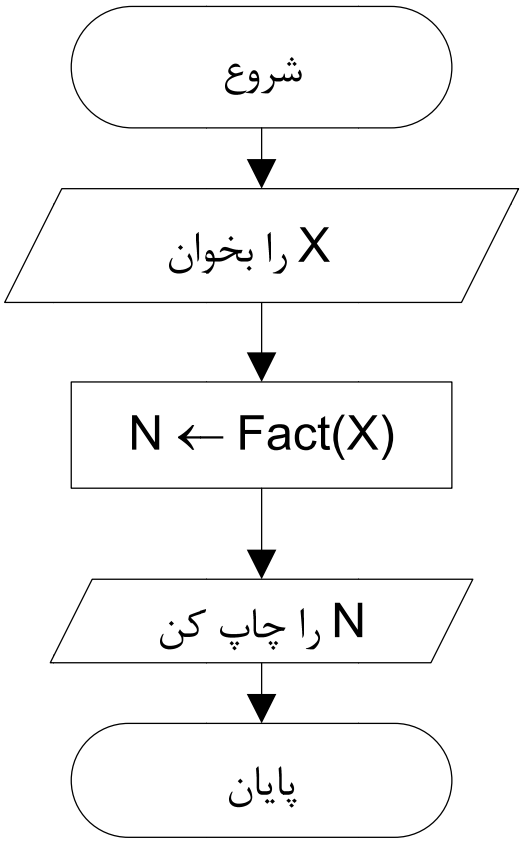
۱. I ← 1
۲. A ← 1
۳. تا زمانی‌که I<=X دستورات ۴ تا ۵ را تکرار کن
۴. A ← A*I
۵. I ← I +1
۶. پایان حلقه (شروع از دستور ۳)
۷. Fact ← A
۸. برگشت

فراخوانی زیرالگوریتم

فراخوانی زیرالگوریتم یعنی به کار گیری آن در الگوریتم اصلی

مثال: فراخوانی زیرالگوریتم فاکتوریل در الگوریتم اصلی

الگوریتمی که عددی را از ورودی خوانده و فاکتوریل آن‌را با استفاده از زیرالگوریتم فاکتوریل چاپ می‌کند.



الگوریتم

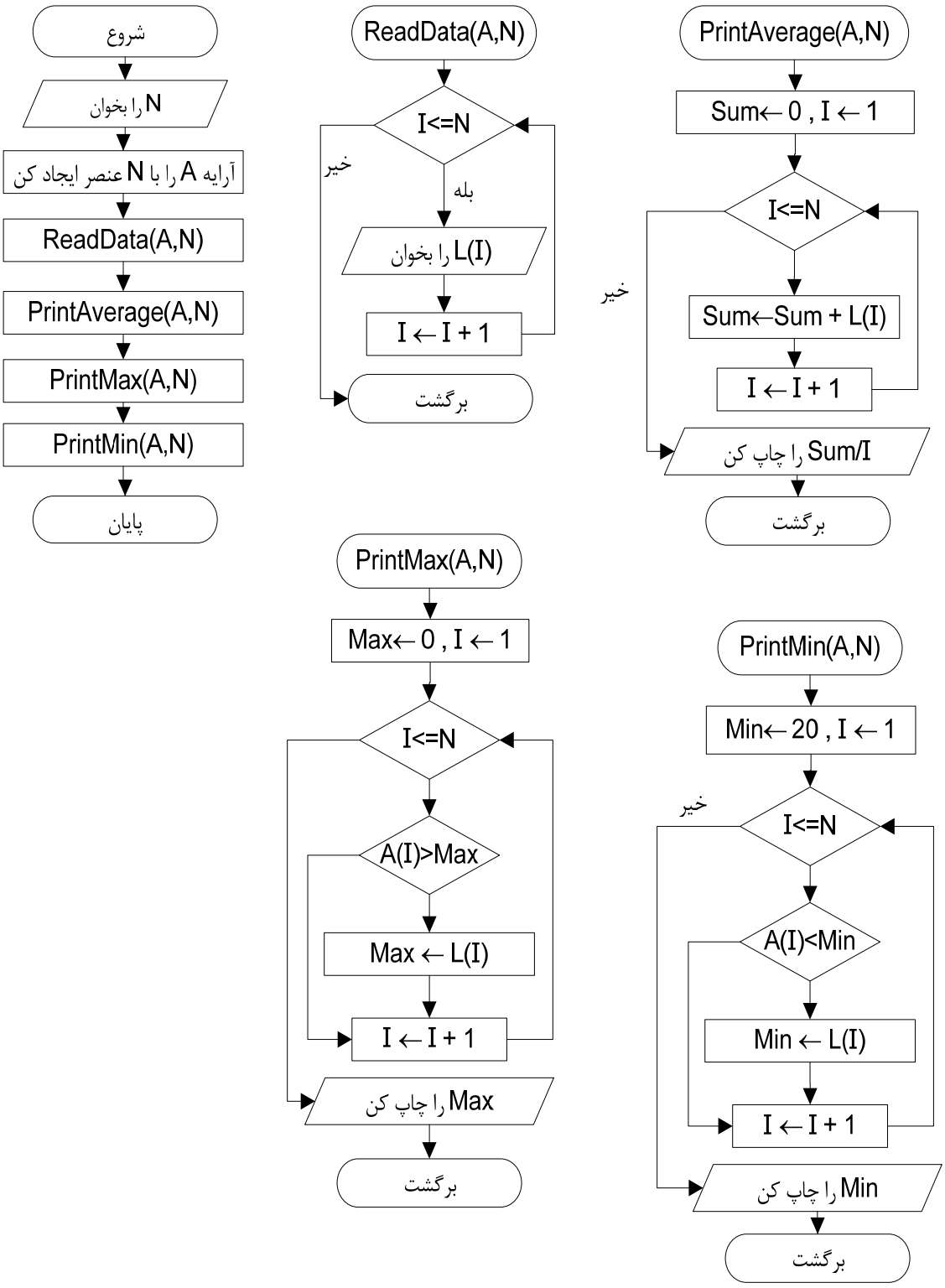
۱. شروع
۲. X را بخوان
۳. N ← Fact(X)
۴. N را چاپ کن
۵. پایان

مثال : محاسبه معدل دانشجویان به همراه بالاترین و پایین ترین نمره

الگوریتمی که معدل N دانشجو را خوانده و در آرایه A قرار می‌دهد، سپس معدل کل دانشجویان، بالاترین نمره به همراه شماره ردیف آن و پایین ترین نمره به همراه شماره ردیف آن را در خروجی چاپ کند.

دید الگوریتمی : مسائلی که از چندین خواسته متفاوت تشکیل شده اند به بهترین نحو با زیرالگوریتم‌ها قابل پیاده سازی هستند. در این مسئله چهار زیرالگوریتم تعریف کنید که شامل خواندن نمرات دانشجویان، محاسبه معدل، بالاترین نمره و پایین ترین نمره را باشد. به این ترتیب الگوریتم پیچیده‌ای به چهار الگوریتم ساده تقسیم می‌شود.

فلوچارت محاسبه معدل دانشجویان به همراه بالاترین و پایین ترین نمره



الگوریتم‌های تکمیلی

الگوریتم‌هایی که در فصول قبل مورد بررسی قرار گرفته‌اند بیشتر روی تکنیک‌های طراحی الگوریتم متمرکز بودند و دسته بندی درستی از الگوریتم‌ها بیان نمی‌کردند. در این فصل، الگوریتم‌ها به صورت موضوعی دسته بندی شده‌اند و هر دسته شامل چندین الگوریتم کاربردی و مفید است. موضوعاتی چون محاسبات عددی و هندسی، تبدیلات تاریخ و زمان و سری‌های ریاضی بخشی از این دسته بندی را شکل می‌دهند.

الگوریتم‌های ریاضی

الگوریتم‌های ریاضی پایه‌ای برای نوشتن الگوریتم‌های پیچیده‌تر هستند. این الگوریتم‌ها اکثراً روی فرمولهای عددی تمرکز کرده و آن‌ها را تبدیل به الگوریتم‌های کامپیوتری می‌کند.

توابع ریاضی پرکاربرد

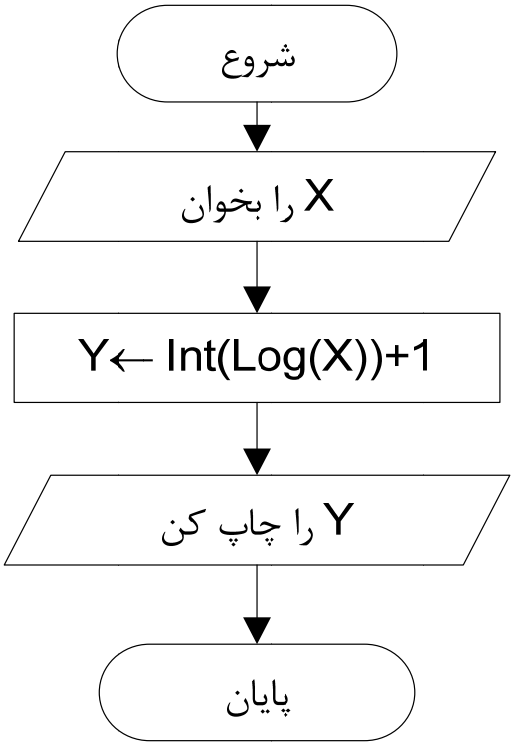
تابع معروفی چون قدر مطلق (Abs)، جزء صحیح (Int)، سینوس زوایه (Sin)، کسینوس زوایه (Cos) و لگاریتم (Log) در طراحی الگوریتم‌ها کاربرد فراوانی دارند. البته با نحوه نوشتن هر یک از این توابع در فصول قبل آشنا شدیم، بنابراین در اینجا فقط این توابع را فراخوانی کرده و از آنها در ایجاد الگوریتم های پیشرفته‌تر استفاده می‌کنیم.

مثال: الگوریتم محاسبه تعداد ارقام یک عدد

الگوریتمی که با استفاده از تابع لگاریتم تعیین می‌کند که یک عدد چند رقمی است.

دید الگوریتمی : در مثال های قبل با تعیین ارقام عدد با استفاده از یک حلقه و مقایسه عدد با توان‌هایی از عدد 10 آشنا شدید. با استفاده از تابع لگاریتم می‌توان تعداد ارقام هر عدد را تعیین کرد. کافی است که لگاریتم عدد را گرفته و قسمت صحیح آن‌را با یک جمع کنید. نتیجه حاصل تعداد ارقام عدد اصلی را نشان می‌دهد.

توضیح : عدد X را از ورودی خوانده و جزء صحیح لگاریتم آن‌را با یک جمع کرده و در متغیر Y قرار می‌دهیم. با چاپ متغیر Y در خروجی، تعداد ارقام متغیر نمایش داده خواهد شد.



جدول متغیرها

عدد ورودی	X	ورودی
تعداد ارقام	Y	خروجی

الگوریتم

۱. شروع
۲. X را بخوان
۳. Y ← Int (Log (X)) + 1
۴. Y را چاپ کن
۵. پایان

مثال: الگوریتم تعیین مثلث بودن سه ضلع

الگوریتمی که سه عدد را به عنوان اضلاع مثلث می‌خواند، سپس بیان می‌کند که آیا آنها می‌توانند یک مثلث را تشکیل دهند یا خیر.

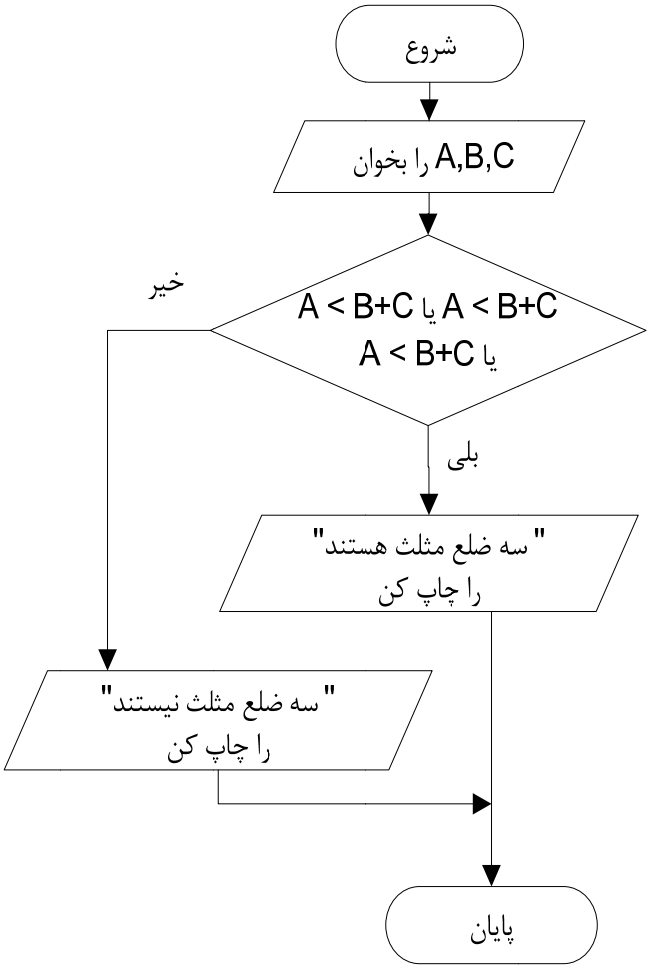
توضیح : ابتدا سه ضلع از ورودی خوانده شده و در متغیرهای اضلاع مثلث قرار می‌گیرد. سپس با دستور شرطی، سه شرط فوق با هم ترکیب می‌شوند. در صورتی‌که یکی از سه شرط درست باشند پیغامی مبنی بر مثلث بودن سه ضلع چاپ می‌شود و در غیر این صورت پیغام مثلث نبودن در خروجی چاپ می‌گردد.

جدول متغیرها

زاویه بر حسب رادیان	X	ورودی
زوایه بر حسب درجه	Y	خروجی

الگوریتم

۱. شروع
۲. A,B,C را بخوان
۳. اگر A < B+C یا A < B+C یا A < B+C
آنگاه " سه ضلع مثلث هستند" را چاپ کن
وگرنه " سه ضلع مثلث نیستند" را چاپ کن
۴. پایان



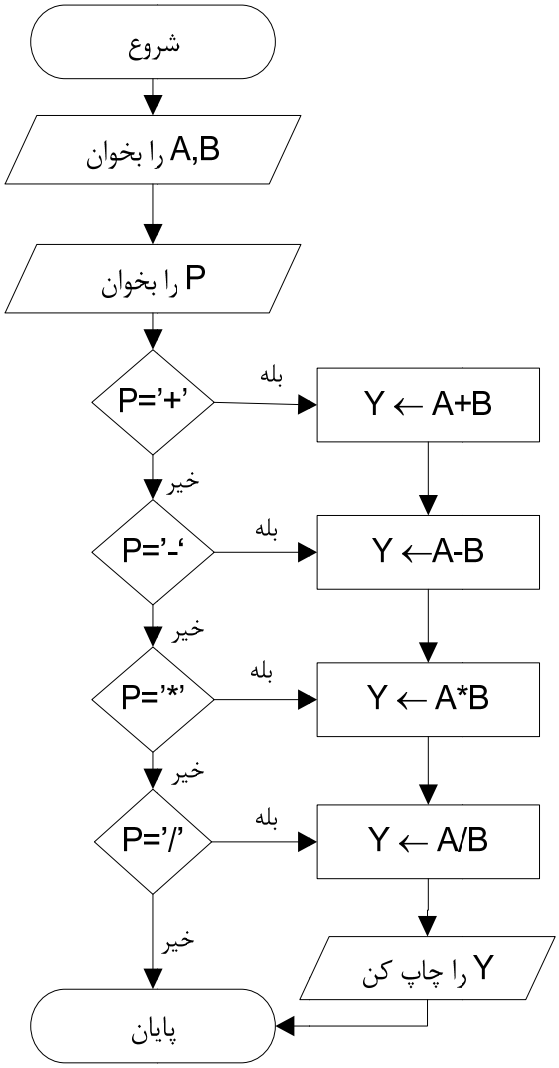
مثال: الگوریتم شبیه سازی یک ماشین حساب

الگوریتمی که دو عدد و یک عملگر از ورودی دریافت کرده و یک ماشین حساب ساده را شبیه سازی می‌کند.

دید الگوریتمی : سه متغیر از ورودی دریافت می شود و با کمک دستورات شرطی مقدار متغیر سوم (عملگر) با کاراکترهای '+' ، '-' ، '*' و '/' مقایسه می شود. در صورت برابری با هر یک از آنها، عملیات درخواستی به‌روی دو متغیر اول اجرا شده و در نهایت نتیجه در خروجی چاپ می شود.

جدول متغیرها

عدد	A,B	ورودی
عملگر	P	ورودی
ماشین حساب ساده	Y	خروجی



الگوریتم

۱. شروع
۲. A,B را بخوان
۳. P را بخوان
۴. اگر P='+' آنگاه Y ← A+B
۵. اگر P='-' آنگاه Y ← A-B
۶. اگر P='*' آنگاه Y ← A*B
۷. اگر P='/' آنگاه Y ← A/B
۸. Y را چاپ کن
۹. پایان

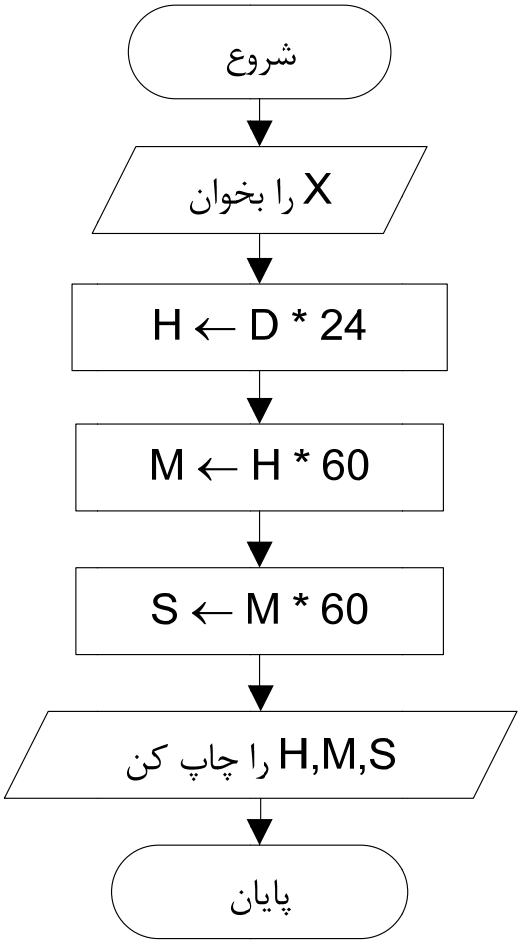
الگوریتم‌های تاریخ و زمان

تاریخ و زمان از عناصر مهم زندگی روزمره انسان‌ها است. تبدیل تاریخ‌ها از میلادی به هجری شمسی و برعکس بسیار پر کاربرد است. همچنین تبدیل واحدهای زمان به هم دیگر از دیگر کاربردهای الگوریتم‌های تاریخ و زمان است.

مثال: تبدیل کننده روزهای ماه به ساعت، دقیقه و ثانیه

الگوریتمی که تعداد روزهای سپری شده از سال جاری را دریافت کرده و آن‌را برحسب ساعت، دقیقه و ثانیه در خروجی چاپ می‌کند.

توضیح : تعداد روزها از ورودی خوانده شده و در متغیر X قرار می‌گیرد. با ضرب X در 24 تعداد ساعات محاسبه شده و در متغیر H قرار می‌گیرد. به همین ترتیب Mو S نیز محاسبه گشته و در خروجی چاپ می‌شوند.



جدول متغیرها

تعداد روزهای سال	X	ورودی
مقدار ساعت	H	خروجی
مقدار دقیقه	M	خروجی
مقدار ثانیه	S	خروجی

الگوریتم

۱. شروع
۲. X را بخوان
۳. H ← D * 24
۴. M ← H * 60
۵. S ← M * 60
۶. H,M,S را چاپ کن
۷. پایان

مثال: تعیین تعداد روزهای سپری شده از تاریخ روز

الگوریتمی که با دریافت تاریخ روز، تعداد روزهای سپری شده از سال را در خروجی چاپ می‌کند.

دید الگوریتمی : این مسئله دقیقاً عکس الگوریتم بالا است. بنابراین باز هم با یک مسئله تبدیل واحد سروکار داریم و باید فرمول تبدیل را به‌دست آوریم.

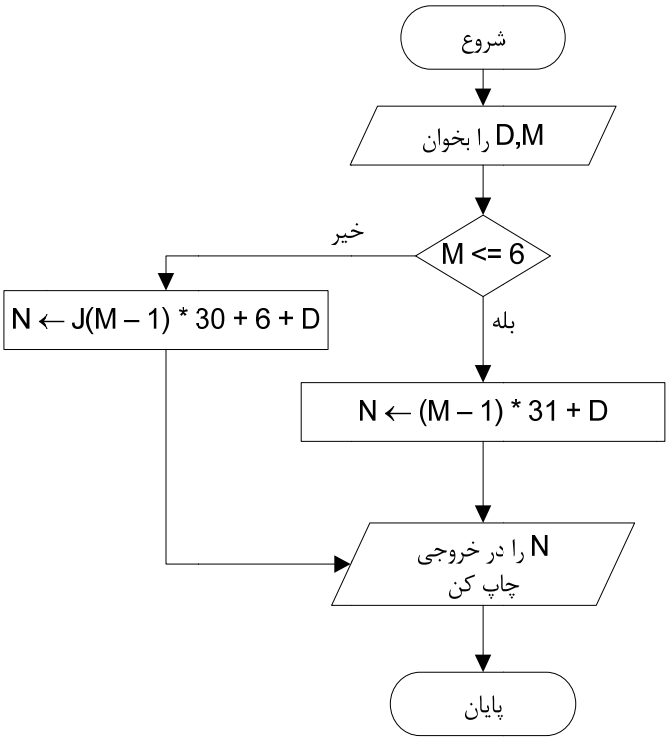
توضیح : ابتدا ورودهای مسئله که روز و ماه است از ورودی دریافت شده و در متغیر D و M قرار می‌گیرد. سپس با دستور شرطی بررسی می‌شود که تعداد ماه از 6 کمتر است یا خیر. اگر کمتر بود از فرمول زیر استفاده می‌شود :

$$N = (M - 1) * 31 + D$$

اگر کمتر نبود از فرمول زیر استفاده می‌گردد :

$$N = 6 * 31 + (M - 7) * 30 + D$$

در انتها N در خروجی چاپ می‌شود.



الگوریتم

۱. شروع

۲. D,M را بخوان

۳. اگر M <= 6 آنگاه

$N \leftarrow (M - 1) * 31 + D$

وگرنه

$N \leftarrow J(M - 1) * 30 + 6 + D$

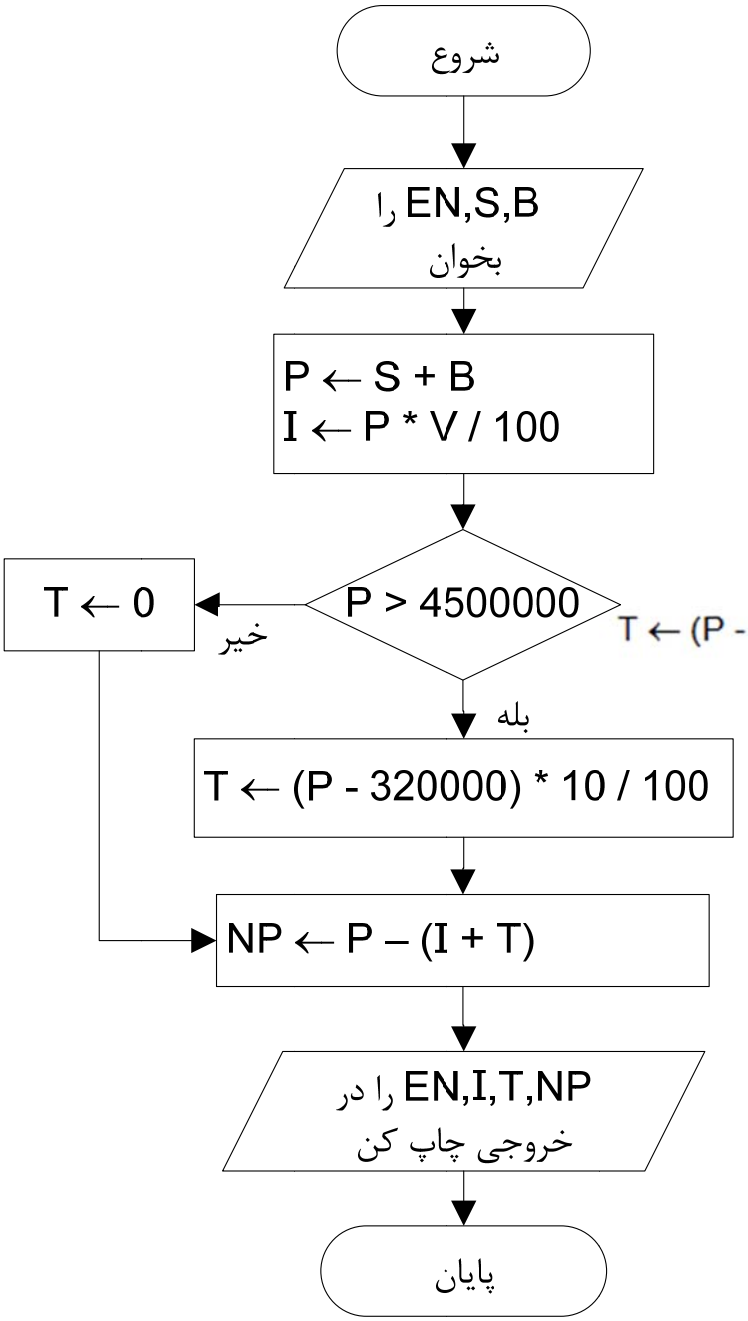
۴. N را در خروجی چاپ کن

۵. پایان

مثال: محاسبه حقوق پرداختی به کارمندان

الگوریتمی که کد کارمندی، حقوق ماهیانه و اضافه کارمندی را دریافت می‌کند و ضمن کسر 7 درصد حقوق و مزایای هر کارمند بابت حق بیمه، چنانچه مجموع حقوق و مزایای کارمند بیش از 4.500.000 ریال باشد، 10 درصد مازاد بر این رقم را به عنوان مالیات کسر نماید، سپس کد کارمندی، حق بیمه، مالیات و خالص پرداختی را چاپ نماید.

توضیح : EN شمارنده کارمند, S حقوق, B مزایا, I بیمه کسر شده, T مالیات کسر شده و NP مجموع حقوق و مزایای خالص است.



الگوریتم

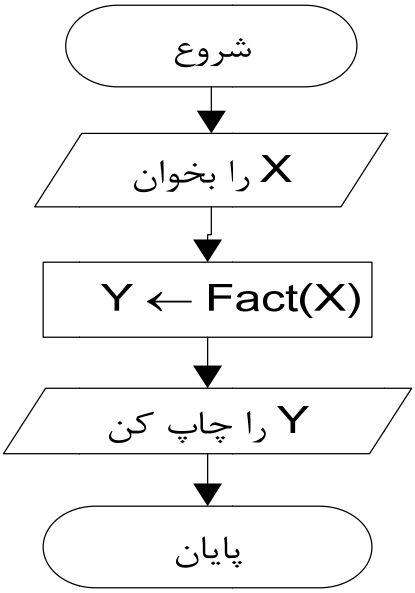
۱. شروع
۲. C ← 1
۳. EN,S,B را بخوان
۴. P ← S + B
۵. I ← P * V / 100
۶. اگر P > 4500000
آنگاه T ← (P - 320000) * 10 / 100
وگرنه T ← 0
۷. NP ← P – (I + T)
۸. EN,I,T,NP را چاپ کن
۹. پایان

الگوریتم‌های بازگشتی

در بحث الگوریتم یکی از موارد مهم و کمی پیچیده، الگوریتم‌های بازگشتی هستند. در الگوریتم‌های بازگشتی یک زیر الگوریتم خودش را فراخوانی می‌کند و در واقع الگوریتم به خودش بازگشت می‌کند و ممکن است بارها اجرا شود. درک نحوه کارکرد این نوع الگوریتم‌ها بسیار مهم است؛ زیرا مهمترین روش‌های جستجو، مرتب سازی و بهینه سازی از الگوریتم‌های بازگشتی استفاده می‌کنند.

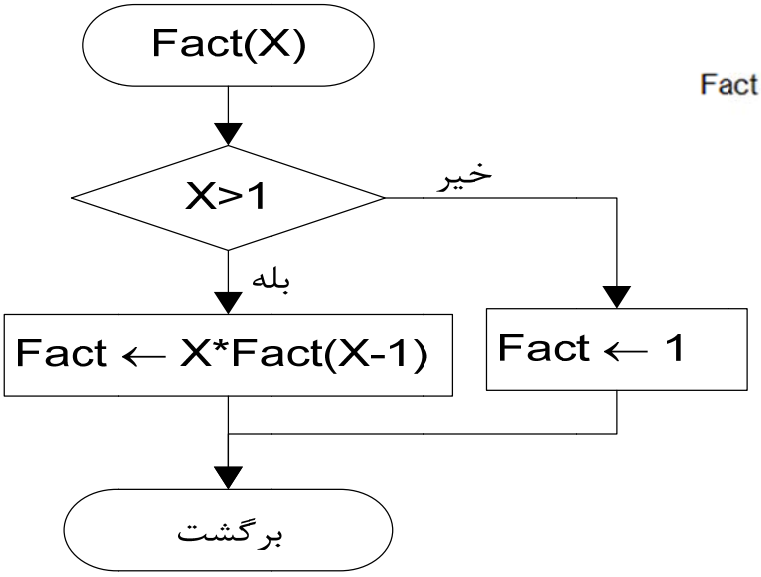
مثال: الگوریتم فاکتوریل به روش بازگشتی

الگوریتمی که فاکتوریل یک عدد را به روش بازگشتی محاسبه می‌کند.



الگوریتم

۱. شروع
۲. X را بخوان
۳. Y ← Fact(X)
۴. Y را چاپ کن
۵. پایان



زیر الگوریتم **Fact(X)**

۱. اگر X>1 آنگاه Fact ← X*Fact(X-1)
وگرنه Fact ← 1
۲. برگشت