## الگوریتم ها

این مبحث اساس و پایه ی برنامه نویسی است. الگوریتم و فلوچارت تنها چیزهایی هستند که به طور کامل میان تمامی زبان های برنامه نویسی مشترک هستند.

شما برای تمامی کارهای زندگی خود یک روال خاص را طی می کنید تا آن کار انجام شود. برای مثال وقتی که کاری مثل غذا خوردن را انجام می دهید مراحل غذاخوردن به این صورت است : با استفاده از قاشق غذای خود را برمی دارید غذا را در داخل دهان قرار می دهید قاشق را در جای اول خود قرار می دهید و خوب غذا را آسیاب می کنید (می جویید!). پس شما برای رفع مشکل گرسنگی که راه حل اون غذا خوردن هست حتما باید این مراحل را قدم به قدم انجام دهید تا (مشکل) گرسنگی شما برطرف شود. این یک روال و یا الگوریتم شما و هر انسانی برای غذاخوردن است. یعنی شما در تمامی کارهای خود یک روال و برنامه را طی می کنید هرچند که شاید در بعضی موارد جزئی با هم تفاوت داشته باشند اما موارد اصلی را حتما استفاده می کنید.

در برنامه نویسی رایانه هم دقیقا این موارد حاکم است. شما برای حل یک مسئله و یا (مشکل) باید مرحله به مرحله مثل یک کودک به کامپیوتر یاد بدهید که آن مشکل را چگونه حل کند. در واقع هربرنامه ای که می نویسید باید یک مسئله را حل کند و برنامه ی شما مراحل قدم به قدم را به کامپیوتر نشان می دهد تا آن مسئله را حل کند. به این مراحل قدم به قدم که برای حل (مسئله) به کار گرفته می شوند ، الگوریتم می گویند. برای مثال وقتی که می خواهید برنامه ای بنویسید که فاکتوریل عدد ۷ را محاسبه کند اولا باید بدانید که این یک (مسئله) است و برای هر مسئله ای باید راه حلی به نام الگوریتم به کامپیوتر معرفی نمود. کامپیوتر خود به خود نمی تواند فاکتوریل یک عدد را حساب کند مگر این که فرمول محاسبه ی فاکتوریل یک عدد را با استفاده از یک برنامه به او بدهید.

بنابراین با توجه به مطالب گفته شده یک تعریف جامع و کلی از الگوریتم ذکر می کنیم :

**الگوریتم مجموعه دستورالعمل های مشخصی است که مراحل انجام یک کار و یا مسئله را با زبانی دقیق و با جزییات کافی که چگونگی ترتیب کامل عملیات و کارها را ذکر می کند.**

✔ **نکته :** کلمه ی الگوریتم از دانشمند بزرگ و پرآوازه یعنی الخوارزمی گرفته شده است.

الخوریسم

ابو محمداله بن موسی خوارزمی

از دانشمندان بزرگ ریاضی جهان و اهل خوارزم بود که در حدود سالهای ۱۲۹ تا ۱۵۹ شمسی متولد شد و در حدود سال ۲۲۹ شمسی در گذشت او اولین کسی است که علم جبر را کشف کرد و کتاب جبر و المقابله را نوشت اروپاییان روش او را مورد استفاده قرار دادند و چون اولین بار به زبان لاتین به نام الخوریسم یا **algorism** چاپ شد نام الگوریسم و لگاریتم به رشته ای از علم حساب که

خوارزمی کاشف آن بود اطلاق گردید این نام در تمام فرهنگنامه های جهان و در دانش ریاضی ثبت شده است سیستم محاسبه ارقام ریاضی اروپاییان از خوارزمی گرفته شده و مدت ۴۰۰ سال کتاب ریاضی وی جز کتب مطرح در دانشگاه های اروپا بود به افتخار این دانشمند ایرانی نیمه اول قرن ۹ میلادی را عصر خوارزمی نامیده اند

کتاب جبر و مقابله و المجمع و التفریق وزیج خوارزمی از کتابهای معروف اوست و کتاب الرخامه درباره محاسبات ظل سایه آفتاب و تعیین اوقات است که پایه و اساس محاسبات مثلثات کروی گردید علم کامپیوتر علم مطالعه الگوریتم هاست الگوریتم برگرفته از نام خوارزمی است به همین دلیل دانشمندان علم کامپیوتر در هزاره خوارزمی او را پدر برنامه نویسی نامیده اند

## اما راستی چرا می گوییم که الگوریتم در بین همه ی زبان های برنامه نویسی مشترک است ؟

به این دلیل که کافی است با دستورات زبان برنامه نویسی موردنظر خود آشنا باشید و فقط عبارات فارسی موجود در الگوریتم بالا را به زبان برنامه نویسی موردنظر ترجمه کنید، کامپیوتر برنامه ی موردنظر شما را اجرا خواهد کرد و مجموع دو عدد را برای شما چاپ خواهد کرد.

هنگامی که یک الگوریتم را می نویسید دقت کنید که موارد زیر را حتما رعایت کنید :

۱ - آغاز و پایان الگوریتم به طور دقیق مشخص باشد.

۲- مراحل دارای جزییات کافی باشند.

۳- مراحل با زبانی دقیق نوشته شوند. مثلا عبارت "حدود ظهر است" برای کامپیوتر نامفهوم است باید دقیق ذکر شود که مثلا "ساعت ۱۱:۵۲ است".

٤- مراحل به ترتیب و درست نوشته شوند.

الگوریتمی که دارای ویژگی های فوق باشد الگوریتم درستی است و برای کامپیوتر به طور کامل قابل ترجمه و تفهیم است.

لازم است بدانید که هر الگوریتم دارای سه بخش اصلی است : آغاز - دستورالعمل ها - پایان که ترتیب این سه جزء مهم است.

## انواع دستورالعمل ها :

**۱- دستورالعمل های محاسباتی و انتسابی :** در این نوع دستورالعمل ها می توانید مقداری را به یک متغیر نسبت دهید و یا عملیات محاسباتی را انجام دهید.

مثلا دستور (first = hello) مقدار hello را به متغیر first نسبت می دهد. و یا دستور ( sec = 2*5) ابتدا عدد ۲ را در ۵ ضرب می کند و سپس آن را در متغیر sec قرار می دهد.

**۲- عبارات توضیحی :** برای اضافه کردن توضیح به برنامه و یا الگوریتم استفاده می شود. که برای جداکردن آن از دستورالعمل ها در داخل پرانتز قرار می گیرد.

**۳- دستورالعمل های شرطی :** بوسیله ی این دستورالعمل ها می توان شرطی را بررسی کرد در صورتی که آن شرط درست باشد عبارت بعد از آن اجرا می شود. برای مثال دستور "اگر ۲>۳ باشد آنگاه چاپ کن درست است " تنها در صورتی عبارت "درست است" را چاپ می کند (در برنامه نویسی معمولا منظور از چاپ کردن نمایش در صفحه ی نمایش است) که ۳ از ۲ بزرگتر باشد و چون این عبارت همیشه درست است در نتیجه همواره در هنگام اجرای برنامه عبارت "درست است" چاپ می شود.

**٤- دستورالعمل های خروجی :** به صورت "چاپ کن مقدار موردنظر" مورد استفاده قرار می گیرد.

## الگوریتم نویسی

برای حل مسائل مختلف توسط کامپیوتر باید این مسائل را به صورت مراحل عملیاتی و تصمیم گیری ساده ای که کامپیوتر قادر به اجرای آن باشد تبدیل کرد. بدین ترتیب لیست مرتبی از مراحل عملیاتی بدست می آید که اجرای مرتب آنها منجر به حل مسئله توسط کامپیوتر می شود. این لیست مرتب از مراحل عملیاتی و تصمیم گیری ، الگوریتم نامیده می شود.

در حالت کلی الگوریتم ها باید ویژگی های زیر را داشته باشند:

**الف)** الگوریتم باید ما را به نتیجه مورد نظر برساند.
**ب)** در زمان محدود پایان یابد.
**ج)** دستورالعملها باید به ترتیب منطقی پشت سرهم قرار گیرند.
**د)** جملات الگوریتم ها باید به صورت امری ، سؤالی باشند.
**ه)** هر الگوریتم باید نقطه آغاز و پایان داشته باشد.

یکی از توانایی هایی که در کامپیوتر وجود دارد استفاده از خانه های حافظه است که می توان در آن اطلاعات را قرار داد و در هر لحظه از اجرای الگوریتم می توان محتویات آن را تغییر داده و مقدار جدیدی را در آن قرار دهیم این ویژگی کارایی ما را برای حل مسائل پیچیده تر افزایش می دهد.

**مثال :** الگوریتم تعویض چرخ پنچر شده یک اتومبیل.

```
۰- شروع.
۱- جک را زیر اتومبیل بگذارید.
۲- پیچهای چرخ پنچر شده را باز کنید.
۳- چرخ را خارج کنید.
۴- چرخ یدک را به جای چرخ پنچر شده بگذارید.
۵- پیچها را ببندید.
۶- اگر پیچها سفت نشده اند به مرحله ۵ برو.
۷- جک را پایین بیاورید.
۸- چرخ پنچر شده را در صندوق عقب اتومبیل بگذارید.
۹- پایان.
```

**مثال :** الگوریتمی بنویسید که دو عدد از ورودی دریافت شود و سپس تعیین شود که مجموع دو عدد بزرگتر از ۲۰ است یا نه.

```
۰- شروع .
۱- دو عدد a و b را از ورودی در یافت کن.
۲- a+b را محاسبه کن.
۳- آیا ۲۰<a+b است؟ اگر بلی به مرحله ۶ برو.
۴- بنویس خیر.
۵- به مرحله ۷ برو.
۶- بنویس بلی.
۷- پایان.
```

با برنامه ریزی و ساماندهی دقیق می توان به راه حلی مناسب جهت حل یک مسئله به کمک کامپیوتر رسید. هرگونه کم توجهی و بی دقتی در نوشتن الگوریتم ضمن بروز مشکلات بسیار، برنامه نویس را نیز از هدف خود دور خواهد کرد؛ لذا برای به هدف رسیدن باید درک صحیح و کاملی از صورت مسئله داشت و سپس راه حل مورد نظر را به صورت الگوریتم بنویسیم. و در نهایت الگوریتم مورد نظر را به زبان برنامه نویسی مورد نظر تبدیل کنیم. برای درک بهتر شیوه حل مسائل و نوشتن الگوریتم به مثالهای زیر توجه کنید:

**مثال :** الگوریتمی بنویسید که مجموع اعداد طبیعی مضرب ۷ و کوچکتر از ۵۰ را حساب کند.

برای نوشتن این الگوریتم به دو خانه حافظه نیاز داریم.

```
۰- شروع.
۱- در خانه حافظه sum عدد صفر را قرار بده.
۲- در خانه حافظه index عدد ۷ را قرار بده.
۳- مقدار index را با مقدارsum جمع کن
      و حاصل را در sum قرار بده.
۴- مقدار ۷ را با مقدار index جمع کن
      و حاصل را در index قرار بده.
۵- آیاindex بزگتراز ۵۰ است،اگر خیر به مرحله ۳ برو.
۶- محتوای sum را چاپ کن.
۷- پایان.
```

**مثال :** الگوریتمی بنویسید که ۱۰۰۰ عدد را از ورودی دریافت کرده و کوچکترین را چاپ کند.

فرض کنید که به شما لیستی از اعداد را می دهند، برای پیدا کردن کوچکترین عدد در لیست اولین عدد را به عنوان کوچکترین در نظر می گیرید سپس عدد بعدی را با آن مقایسه می کنید، اگر عدد جدید از عدد قبلی کوچکتر بود عدد جدید را به عنوان کوچکترین در نظر می گیرید و گر نه همان عدد قبلی کوچکترین خواهد بود. این روند را تا انتهای لیست ادامه می دهید؛ در پایان عددی که در هر بررسی به عنوان کوچکترین عدد بود، جواب مورد نظر ما خواهد بود. توجه کنید که در این روال شما همواره یک عدد را در ذهن خود در نظر گرفته بودید، برای نوشتن الگوریتم مورد نظر ما یک خانه حافظه را به کوچکترین عدد درهر مرحله اختصاص می دهیم.

```
۰- شروع.
۱- min را دریافت کن.
۲- i =1 .
۳- a را دریافت کن.
۴- اگر min>a آنگاه min = a .
۵- i = i + 1 .
۶- اگر i=<۱۰۰۰ به مرحله ۸ برو.
۷- به مرحله ۳ برو.
۸- min را چاپ کن.
۹- پایان.
```

الگوریتم های قبلی به صورت جملات فارسی بودند که سبب طولانی و حجیم شدن الگوریتم می شدند. ولی الگوریتم اخیر بیشتر به صورت جملات ریاضی بود. این شیوه سبب راحتی درک الگوریتم و ساده شدن نگارش آن می شود. از این به بعد نیز الگوریتم ها را به شیوه جدید نگارش خواهیم کرد. شما نیز سعی کنید از این شیوه استفاده کنید.

**مثال :** الگوریتمی بنویسید که سه عدد از ورودی دریافت شود و تعیین شود که این اعداد می توانند اضلاع مثلث باشند یا خیر.

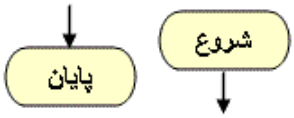
```
۰- شروع.
۱- a وb وc را از ورودی بگیر.
۲- اگر b+c<a به ۷ برو.
۳- اگر a+c<b به ۷ برو.
۴- اگرa+b<c به ۷ برو.
۵- بنویس " بلی ".
۶- به ۸ برو.
۷- بنویس " خیر ".
۸- پایان.
```

## فلوچارت

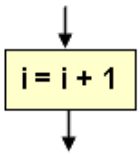
در عمل برای نمایش الگوریتم از یک فلوچارت ( شمای جریان عملیات ) استفاده می شود. در حقیقت فلوچارت روش تصویری و استاندارد نمایش الگوریتم است.

در رسم فلوچارت علائم و نمادهای استانداردی به کار می رود که هر کدام دارای معانی ویژه ای هستند.

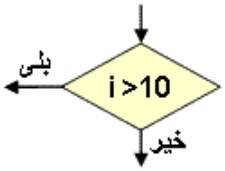
از شکل بیضی افقی برای شروع و پایان عملیات استفاده می شود.



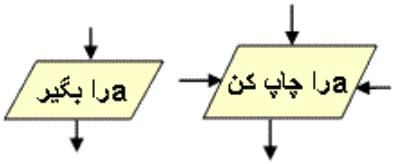

از شکل مستطیل برای نمایش مراحل پردازشی استفاده می شود و در داخل آن عمل مورد نظر نوشته می شود. این نماد ممکن است چندین ورودی داشته باشد ولی تنها یک خروجی دارد.



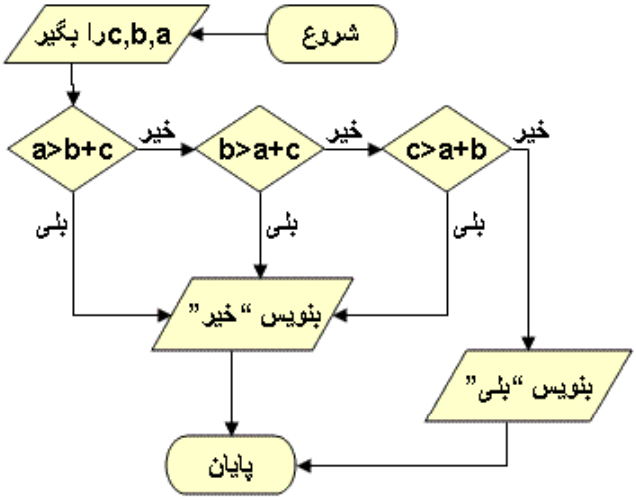

از نماد لوزی برای نشان دادن مراحل تصمیم گیری استفاده می گردد و شرط یا سؤال مورد نظر در داخل لوزی نوشته می شود.




از متوازی الاضلاع برای نشان دادن ورودی یا خروجی استفاده می شود.



**مثال :** فلوچارت الگوریتم اضلاع مثلث در مثال قبل به صورت زیر می باشد.




**پایان**