متخصص الگوریتم و منطق برنامه‌نویسی شوید!

آموزش گام‌به‌گام الگوریتم نویسی به زبان ساده

همیار آی‌تی

به شما تبریک می‌گوییم که اقدام به یادگیری صحیح برنامه‌نویسی کرده و سعی دارید دانش خود را در این زمینه افزایش دهید، مطمئنا این‌کار به شما کمک می‌کند تا از این پس با دید بازتری به دنیای برنامه‌نویسی نگاه کنید.

همانطوری که بارها در مجموعه آموزش‌های برنامه‌نویسی بیان کرده‌ایم، یکی از مهم‌ترین گام‌های یادگیری برنامه‌نویسی آشنایی صحیح و اصولی با علم الگوریتم و الگوریتم‌نویسی می‌باشد، شما با دانستن الگوریتم به خوبی با پایه، اساس و منطق برنامه‌نویسی آشنا شده و می‌توانید هر برنامه‌ای را بنویسید.

فرقی نمی‌کند به چه زبان برنامه‌نویسی علاقه‌مند باشید، شما با دانستن منطق الگوریتمی خواهید توانست در هر زبانی برنامه‌نویسی کنید، چراکه منطق و اساس برنامه‌های کامپیوتری در تمام زبان‌های برنامه‌نویسی یکسان است، به بیان ساده‌تر آشنایی با مفهوم الگوریتم پیش‌نیاز ورود به دنیای برنامه‌نویسی است.

این آموزش به شما کمک می‌کند به سادگی و به کمک مثال‌های کاربردی در عرض چند ساعت با مفاهیم مهم الگوریتم‌نویسی آشنا شوید، به شما توصیه می‌کنیم هر مثال را با دقت خوانده و سعی کنید در ابتدا خودتان به آن پاسخ دهید، سپس نگاهی به راه‌حل انداخته و آن‌را با پاسخ خودتان مقایسه کنید.

در پایان لطفا توجه داشته باشید که این کتاب الکترونیک یک محصول پریمیوم از همیار آی‌تی بوده و مخصوص عزیزانی است که آن را مستقیما از سایت خریداری کرده‌اند.

لطفا در صورتی که این محصول را خودتان نخریده‌اید، هم‌اکنون اقدام به خرید آن کنید تا از علاوه‌بر پشتیبانی ویژه از پاسخگویی به سوالات الگوریتم‌نویسی‌تان نیز برخوردار شوید، پاسخ‌گویی به سوالات تنها شامل دوستانی می‌شود که این محصول را مستقیما از سایت خریداری کرده‌اند.

امیدواریم از مطالعه‌ی این آموزش ویژه لذت ببرید

(Hamyarit.com)

راهنمای فصل‌های کتاب

فصل اول

فصل دوم

فصل سوم

فصل چهارم

فصل پنجم

فصل اول

الگوریتم دقیقا چیست؟

آشنایی با مفهوم الگوریتم

الگوریتم مفهومی است که مسائل را با استفاده از دستورالعمل‌های پشت سرهم و به کمک تحلیل‌های ریاضی و منطقی مورد بررسی قرار داده و راه حل مناسبی برای آن ارائه می‌کند، الگوریتم به ما کمک میکند مراحل حل مسئله را به زبان رایانه نزدیک‌تر کرده و در نهایت آن را به کدهای قابل فهم کامپیوتر تبدیل کنیم.

برای نوشتن یک الگوریتم در ابتدا باید ۳ عامل اصلی را در صورت مسئله بیابید!

مقادیر معلوم

اطلاعاتی که در اختیار ما قرار داده شده و باید به کمک آن‌ها به حل مسئله بپردازیم (داده‌ها)

خواسته‌های مسئله

نتایجی که در اثر انجام محاسبات بر روی داده‌های مسئله حاصل می‌شود (مقادیر مجهول)

عملیات محاسباتی

دستورات و روابط منطقی که برای رسیدن به خواسته‌های مسئله بر روی داده‌ها و مقادیر مجهول انجام می‌شود.

نکاتی که باید در نوشتن الگوریتم رعایت کنید

- مراحل الگوریتم را به ترتیب و پشت‌سرهم بنویسید (اجرا از بالا به پایین)
- مراقب باشید بخش‌های ضروری مسئله را از قلم نیندازید
- تا حد امکان الگوریتم خود را ساده و در عین حال کامل بنویسید
- الگوریتم‌ها فقط و فقط یک نقطه‌ی شروع دارند ولی می‌توانند چند پایان داشته باشند
- الگوریتم شما باید جامع باشد، در حالت‌های خاص به مشکل نخورده و بتواند پاسخ‌گوی انواع حالات مختلف باشد (همه‌ی جوانب را در نظر بگیرید)
- حتما اولویت عملگرهای ریاضی را در نظر داشته باشید، که به ترتیب اولویت عبارتند از:
  - پرانتز
  - توان
  - ضرب/تقسیم/باقی‌مانده
  - جمع/تفریق

به عنوان مثال حاصل عبارت زیر برابر با ۱۹ است.

$$۵ + ۳^{۲} * ۲ - ۸ \div ۲$$

اگر در یک سطر اولویت‌ها یکسان باشند عملیات به ترتیب از چپ به راست انجام خواهد شد.

یک الگوریتم می‌تواند شامل بخش ورودی و خروجی باشد، به همین دلیل می‌توانید یک نام برای متغیرهای ورودی و خروجی خود انتخاب کنید، به عنوان مثال به جای اینکه بگویید عدد را بخوان، عدد را نمایش بده، بگویید عدد a را بخوان، عدد a را نمایش بده.

تخصیص یک نام اختصاصی به متغیرها به شما کمک می‌کند راحت‌تر چندین متغیر ورودی و خروجی را کنترل کرده و آن‌ها را مدیریت کنید.

دستورات مورد استفاده در الگوریتم‌ها

در حالت کلی یک الگوریتم می‌تواند شامل ۵ نوع دستور مختلف باشد، این دستورات عبارتند از:

- **دستورات ورودی**
- **دستورات خروجی**
- **دستورات محاسباتی**
- **دستورات شرطی**
- **دستورات تکرار**

در ادامه‌ی فصل‌ها با همه‌ی این موارد آشنا خواهیم شد.

فصل دوم

دستورات ورودی، خروجی و محاسباتی

همانطور که گفتیم، الگوریتم‌ها از اجرای پشت‌سرهم چند دستور ساده تشکیل شده‌اند، شما باید با صبر و حوصله این دستوارت را به دقت نوشته تا به یک الگوریتم صحیح برسید، در گام اول دستوارت ورودی، خروجی و محاسباتی را باهم بررسی می‌کنیم.

دستورات ورودی

این دستورات برای دریافت داده‌های ورودی استفاده می‌شوند، شما میتوانید یک نام اختصاری را به متغیرهای ورودی تخصیص دهید. (بخوان، دریافت کن، بگیر و... جزو این نوع دستورات می‌باشند)

دستورات خروجی

این دستورات برای نمایش نتایج الگوریتم یا نمایش پیام‌های مورد نیاز به کار می‌روند (نمایش بده، چاپ کن و... از جمله دستورات خروجی هستند)

دستورات محاسباتی

این دستورات نحوه‌ی ارائه و محاسبات دستورات را با فرمول‌های ریاضی بیان کرده و به زبان ریاضی نیز نوشته میشوند که شامل ۳ جز اصلی میباشد:

- **متغیر:** عنوان‌هایی متشکل از حرف و عدد که مقدار آن‌ها قابل تغییر است (a, b, n, s۲)
- **عملگرهای محاسباتی:** عملگرهای ریاضی از پیش تعریف شده در دستگاه الگوریتم (= , % , / , * , – , +)
- **عملوندهای محاسباتی:** اعداد و عباراتی که محاسبات بر روی آن‌ها انجام می‌شود (شامل اعداد و متغیرها)

به عنوان مثال فرض کنید وزن مقداری سیب و قیمت هر کیلوگرم از آن به ما داده شده است، از ما خواسته می‌شود قیمت کل سیب‌ها را محاسبه کنیم.

در این مثال:

داده‌ها: وزن سیب‌ها (W) و قیمت هر کیلوگرم سیب (P)

خواسته‌ها: محاسبه‌ی قیمت کل سیب‌ها (T)

عملیات محاسباتی: قیمت کل = وزن سیب‌ها * قیمت هر کیلوگرم

مراحل حل این مسئله به صورت زیر است:

$$T=W\times P$$

الگوریتمی بنویسید که مراحل پختن یک کیک را به خوبی نمایش می‌دهد.

۱. شروع
۲. تهیه‌ی آرد
۳. تهیه‌ی شکر
٤. تهیه‌ی تخم‌مرغ
٥. تهیه‌ی آب
٦. مخلوط کردن، آرد، شکر، تخم‌مرغ و آب
۷. ریختن محتویات در ظرف مخصوص
۸. قرار دادن ظرف در فر یا مایکروویو
۹. روشن کردن حرارت
۱۰. صبر کردن تا پخت کامل کیک
۱۱. خارج کردن کیک از فر
۱۲. برش کیک به قطعات دلخواه
۱۳. پایان

همانطوری که مشاهده کردید، این مراحل در نهایت سادگی باید کاملا به اجزای کوچک‌تر تبدیل شده و به ترتیب قرار گیرند، اگر تنها یک مرحله را از قلم بیندازید کل فرایند پخت کیک با شکست مواجه خواهد شد!

الگوریتم های برنامه‌نویسی نیز دقیقا به همین شکل هستند، اگر حتی یک مرحله را به درستی یا ترتیب ننویسید، کامپیوتر به شدت به دردسر افتاده و خروجی صد درصد اشتباهی را برای شما تولید خواهد کرد!

پس در اولین گام سعی کنید، مراحل انجام کار را دقیقا و مرحله به مرحله بنویسید، به جرئت می‌توان گفت مهم‌ترین گام برای نوشتن یک الگوریتم خوب، فرایند شکستن کار به چند مرحله‌ی کوچک‌تر است، شما باید بتوانید این مراحل را به خوبی تشخیص داده و آن‌ها را از هم تفکیک کنید.

حال باهم یک مثال ساده‌ی دیگر را بررسی می‌کنیم.

الگوریتمی بنویسید که مراحل کاشت یک گیاه را نمایش دهد.

۱. شروع
۲. تهیه‌ی یک ظرف یا گلدان
۳. تهیه‌ی بذر گیاه مورد نظر
٤. ریختن خاک در گلدان
٥. قرار دادن بذر گیاه در میان خاک
٦. آب دادن به بذر کاشته شده
۷. پایان

همانطور که می‌بینید مراحل انجام این فرایند نسبت به الگوریتم قبلی از گام‌های کمتری برخوردار است اما همچنان تک‌تک مراحل کامل و پشت سرهم نوشته شده است.

بیایید مثال بعد را کمی به دنیای کامپیوتر نزدیک‌تر کنیم.

الگوریتمی بنویسید که یک عدد را از ورودی خوانده و دقیقا همان را در خروجی نمایش دهد.

۱. شروع
۲. عدد را بخوان
۳. عدد را نمایش بده
٤. پایان

ساده‌تر از آن چیزی بود که فکرش را می‌کردید، اینطور نیست؟! پس مثال بعدی را کمی پیچیده‌تر می‌کنیم!

الگوریتمی بنویسید که دو عدد را از ورودی خوانده و آن‌ها را باهم جمع کند.

۱. شروع
۲. عدد a را بخوان
۳. عدد b را بخوان
٤. مقدار a+b را محاسبه کن
٥. حاصل را در خروجی نمایش بده
٦. پایان

به نظر شما این الگوریتم را به شکل‌های دیگری نیز می‌توان نوشت؟

به عنوان مثال به جای اینکه در مرحله‌ی ٤ فقط اعداد را جمع کنیم، به کامپیوتر می‌گوییم، **مقدار a+b را نمایش بده**، یعنی عمل جمع و نمایش حاصل را در یک مرحله انجام بده، بدین ترتیب ۱ شماره از مراحل الگوریتم ما کم خواهد شد و به عبارتی الگوریتم کوتاه‌تر و در عین حال کمی پیچیده‌تر می‌شود.

نکته: با توجه به مثال بالا درمیابیم راه‌های گوناگونی برای نوشتن یک الگوریتم وجود دارد، تنها کافیست پاسخ نهایی درست باشد، شما آزادید الگوریتم خود را هرطور که دوست دارید بنویسید، اما به شرط صحیح بودن نتیجه‌ی الگوریتم!

پس ذهن خود را آزاد بگذارید، شاید راه حلی که برای نوشتن یک الگوریتم به ذهن شما می‌رسد خیلی بهتر و بهینه‌تر از راه حلی باشد که به ذهن یک مهندس کامپیوتر رسیده است، تنها کافیست تمام جوانب را در نظر گرفته و سعی کنید نتیجه الگوریتم شما صحیح باشد و روش پیشنهادی‌تان به خوبی کار کند.

الگوریتمی بنویسید که طول (L) ، عرض (W) و ارتفاع (H) یک مکعب مستطیل را از ورودی دریافت، حجم آن‌ها محاسبه کرده و نمایش دهد.

۱. شروع
۲. L و W و H را دریافت کن
۳. L*W*H را نمایش بده
٤. پایان

آیا می‌توانید پاسخ‌های متنوع دیگری نیز برای این الگوریتم بیابید؟

الگوریتمی بنویسید که شعاع یک دایره را دریافت کرده و محیط و مساحت آن را نمایش دهد.

۱. شروع
۲. R را دریافت کن
۳. مقدار R*R*۳٫۱٤ را در masahat قرار بده
٤. مقدار R*۲*۳٫۱٤ را در mohit قرار بده
٥. masahat و mohit را نمایش بده

سوال بالا، یک سوال محاسباتی بسیار ساده است، تنها کافیست رابطه‌ی به‌دست آوردن محیط و مساحت دایره را پیدا کنید، یعنی یک جستجوی ساده در اینترنت!

همیشه این نکته را به یاد داشته باشید، برنامه‌نویس خوب کسی است که بتواند به خوبی در اینترنت جستجو کند، پس سعی کنید هرگاه پاسخ سوالی را نمی‌دانستید، قبل از هرکار دیگری، آن را در اینترنت جستجو کنید.

مثل همین رابطه‌ی محیط و مساحت دایره که با یک جستجوی ساده در اینترنت آن را خواهید یافت، مساحت دایره برابر است با (شعاع*شعاع*۳٫۱٤) و محیط دایره نیز برابر است با (شعاع*۲*۳٫۱٤)

الگوریتمی بنویسید که ۳ عدد را به ترتیب از ورودی خوانده و آن‌ها را به همان ترتیب نمایش دهد.

۱. شروع
۲. عدد a را بخوان
۳. عدد b را بخوان
٤. عدد c را بخوان
٥. عدد a را نمایش بده
٦. عدد b را نمایش بده
۷. عدد c را نمایش بده
۸. پایان

آیا می‌توانید یک راه حل دیگر برای این الگوریتم ارائه دهید؟

به عنوان راهنمایی به شما می‌گوییم، بلافاصله بعد از دریافت عدد، آن را در خروجی نیز نمایش دهید، مثلا:

- ...
- عدد a را بخوان
- عدد a را نمایش بده
- ...

حال فرض کنید، قصد داریم در الگوریتمی ۱۰۰۰ عدد را به همین ترتیب از ورودی خوانده و آن‌ها را نمایش دهیم، آیا منطقی است ۱۰۰۰ بار دستور بخوان را بنویسیم و ۱۰۰۰ بار نیز دستور نمایش بده؟ در چنین شرایطی مجبوریم این دستورات ساده را ۲۰۰۰ بار بنویسیم!

مسلما پاسخ خیر است، در اینگونه مواقع باید از حلقه‌های برنامه‌نویسی استفاده کنیم که در فصل چهارم این آموزش به آن‌ها خواهیم پرداخت، اما قبل از آن در ابتدا الگوریتم دستورات شرطی را بررسی خواهیم کرد.

فصل سوم

الگوریتم‌های دستورات شرطی

هنگامی که قصد داریم در الگوریتم خود صحیح بودن یک شرط را بررسی کنیم از الگوریتم‌های دستورات شرطی استفاده خواهیم کرد، این‌گونه الگوریتم‌ها در ابتدا یک شرط را مورد بررسی قرار می‌دهند و درصورت صحیح بودن شرط دستورات خاصی را انجام می‌دهند.

قالب کلی دستورات شرطی به شکل زیر است:

اگر شرط برقرار بود آنگاه دستوری خاص را انجام بده، در غیر این‌صورت دستور دیگری را انجام بده.

از چه دستوراتی میتوانیم در شرط استفاده کنیم؟

- عملگرهای مقایسه‌ای (کوچکتر، بزرگتر، مساوی، نامساوی و...)
- عملگرهای منطقی (and, or, not)

عملگرهای مقایسه‌ای

== عملگر تساوی - مثال ۴ == ۴

=< عملگر بزرگ‌تر مساوی - مثال ۱ =< ۵

=> عملگر کوچک‌تر مساوی - مثال ۵ => ۱

> عملگر بزرگ‌تر - مثال ۲ > ۳

< عملگر کوچک‌تر - مثال ۴ < ۲

!= عملگر نامساوی - مثال ۲ =! ۳

عملگرهای منطقی

اگر بخواهیم چند شرط و مقایسه را با هم ترکیب کنیم باید از عملگرهای منطقی استفاده کنیم، که شامل دستورات زیر میشوند:

and: عملگر "و" منطقی (and منطقی)

این عملگر شرط‌ها را بررسی می‌کند و اگر **تمام آن‌ها درست باشند** دستورات شرط برقرار می‌شود، اما اگر تنها یکی از شرط‌ها برقرار نباشد، دستورات شرط اجرا نخواهند شد.

or: عملگر "یا" منطقی (or منطقی)

این عملگر شرط‌ها را بررسی می‌کند و اگر **تنها یکی از آن‌ها نیز درست باشد** دستورات شرط برقرار می‌شود، فقط در صورتی که هیچ کدام از شرایط برقرار نباشد دستورات اجرا نمی‌شود.

not: عملگر "نقیض"

این عملگر تمام دستورات را نقض میکند، یعنی عبارات صحیح غلط و عبارات غلط صحیح می‌شوند.

(در ادامه مثال‌هایی از این مفاهیم خواهیم زد تا به صورت کاربردی با آن‌ها آشنا شوید)

الگوریتمی بنویسید که یک عدد را دریافت کند، اگر عدد بزرگ‌تر از ۱۰ بود عبارت قبول را نمایش دهد و اگر کوچک‌تر از ۱۰ بود عبارت مردود را نمایش دهد.

۱. شروع
۲. عدد را دریافت کن
۳. اگر عدد بزرگ‌تر از ۱۰ بود "قبول" را نمایش بده
٤. اگر عدد کوچک‌تر از ۱۰ بود "مردود" را نمایش بده
۵. پایان

به نظر شما الگوریتم بالا را به چه شکل‌های دیگری می‌توان نوشت؟ به نظر شما الگوریتم بالا در ازای دریافت عدد ۱۰ چه واکنشی از خود نشان می‌دهد؟

در دنیای واقعی شما باید تمام جوانب و شرایط یک الگوریتم را در نظر داشته باشید، اما در این سوال چون در صورت مسئله اشاره‌ای به خود عدد ۱۰ نشده است، در نوشتن الگوریتم نیز آن را نادیده گرفتیم، اما شما می‌توانید یک شرط دیگر به الگوریتم‌تان اضافه کرده و برای اعداد مساوی ۱۰ نیز یک عبارت منحصر به فرد نمایش دهید.

الگوریتمی بنویسید که یک عدد را دریافت کند، اگر عدد بزرگ‌تر از ۱۰ بود عبارت قبول را نمایش دهد و در غیر این‌صورت عبارت مردود را نمایش دهد.

۱. شروع
۲. عدد را دریافت کن
۳. اگر عدد بزرگ‌تر از ۱۰ بود "قبول" را نمایش بده و در غیر این‌صورت "مردود" را نمایش بده.
٤. پایان

به نظر شما الگوریتم بالا دقیقا همانند مثال قبلی نیست؟ این الگوریتم به ازای اعداد بالای ۱۰ عبارت قبول را نمایش می‌دهد و به ازای سایر اعداد (که شامل اعداد کوچک‌تر از ۱۰ هستند) عبارت مردود را نشان خواهد داد.

هرچند نحوه‌ی نوشتن الگوریتم‌ها تا حدودی متفاوت بود اما دقیقا کار یکسانی را انجام می‌دادند (البته اگر از رفتار الگوریتم در ورودی عدد ۱۰ صرف نظر کنیم)

به نظر شما این دو الگوریتم در ورودی عدد ۱۰ چه تفاوت‌هایی با هم دارند؟

الگوریتمی بنویسید که یک عدد را از ورودی دریافت کرده و زوج یا فرد بودن آن را تشخیص دهد.

۱. شروع
۲. عدد را دریافت کن
۳. اگر باقی‌مانده تقسیم عدد بر ۲ برابر ۰ بود عبارت "زوج" را نمایش بده در غیر این‌صورت عبارت "فرد" را نمایش بده.
٤. پایان

توضیح این الگوریتم: می‌دانیم که اعداد زوج اعدادی هستند که در صورت تقسیم شدن بر ۲ باقی‌مانده‌ی آن‌ها همیشه صفر خواهد شد، به عنوان مثال عدد ٤ را بر ۲ تقسیم کنید، مشاهده خواهید کرد که باقی‌مانده همیشه ۰ است، این فرایند را برای عدد ١٦ تکرار کنید، همچنان باقی‌مانده‌ی تقسیم ١٦/٢ برابر صفر خواهد بود.

حال به عنوان مثال عدد ۷ را بر ۲ تقسیم کنید (تقسیم صحیح و بدون اعشار) مشاهده می‌کنید که باقی‌مانده ۱ خواهد شد و این‌بار ۰ نیست، این یعنی عدد ۷ یک عدد فرد است، یا به عنوان مثالی دیگر عدد ۸۵ را بر ۲ تقسیم کنید، همچنان مشاهده می‌کنید که باقی‌مانده ۱ می‌شود.

این رابطه برای تمام اعداد صحیح موجود در دنیا صدق می‌کند، کافیست خودتان اعداد بیشتری را امتحان کنید، پس همواره این نکته را به یاد داشته باشید که برای نوشتن یک الگوریتم صحیح باید بتوانید یک رابطه‌ی منطقی و یا ریاضی برای پارامترهای آن پیدا کنید، این رابطه چندان سخت یا پیچیده نخواهد بود، بلکه فقط احتیاج به اندکی تفکر دارد، چرا که قبلا به شما گفتیم برای نوشتن یک الگوریتم آن را به بخش‌های کوچک‌تر تبدیل کنید، این بخش‌های کوچک از روابط بسیار ساده‌ی ریاضی و منطقی تشکیل شده‌اند و شما به راحتی قادر به پیدا کردن آن‌ها خواهید بود.

الگوریتمی بنویسید که دو عدد را دریافت کرده و عدد بزرگ‌تر را در خروجی نمایش دهد، همچنین اگر هر دو عدد باهم مساوی بودند، عبارت "مساوی" را نمایش دهد.

۱. شروع
۲. عدد a و b را دریافت کن
۳. اگر a>b عدد a را در خروجی نمایش بده
٤. اگر b>a عدد b را در خروجی نمایش بده
٥. اگر a=b عبارت "مساوی" را نمایش بده
٦. پایان

الگوریتمی بنویسید که ۳ عدد را دریافت کرده و آن‌ها را مقایسه کند، سپس عددی که کوچک‌تر است را در خروجی نمایش دهد.

۱. شروع
۲. عدد a و b و c را دریافت کن
۳. a را در min قرار بده
٤. اگر b<min آنگاه b را در min قرار بده
٥. اگر c<min آنگاه c را در min قرار بده
٦. min را نمایش بده
۷. پایان

توضیح این الگوریتم: در چنین سوالی ما نمی‌توانیم هر ۳ عدد را همزمان با هم مقایسه کنیم، بنابراین باید آن‌ها را ۲ به ۲ با هم مقایسه کرده و نتیجه را اعلام کنیم، برای این منظور یک متغیر کمکی (min) در نظر می‌گیریم و فرض می‌کنیم عدد a کوچک‌ترین عدد ورودی است (شاید این فرض اشتباه باشد)

بنابراین عدد a را در متغیر min قرار داده و تا اینجای کار min را کمترین عدد موجود در نظر می‌گیریم، سپس در گام بعدی مقدار min را با عدد b مقایسه می‌کنیم، چراکه شاید فرض اولیه‌ی ما (a کوچک‌ترین عدد است) اشتباه بوده باشد، بنابراین هنگامی که مقدار min (که حالا برابر با a است) را با b مقایسه کنیم، می‌توانیم فرض خود را آزموده و بین دو عدد a و b کوچک‌ترین را بیابیم، اگر b<a بود، یعنی فرض اولیه اشتباه بوده و عدد b کوچک‌تر است، بنابراین مقدار b را در متغیر min جایگذاری خواهیم کرد و تا اینجای کار b را کوچک‌ترین عدد موجود فرض می‌کنیم (هرچند این فرض نیز می‌تواند اشتباه باشد)

سپس همین فرایند را برای متغیر c نیز تکرار می‌کنیم و این بار مقدار min (که این‌بار برابر b است) را با c مقایسه می‌کنیم، اگر c از متغیر min کوچک‌تر بود، یعنی فرض‌های قبلی اشتباه بوده و

کوچک‌ترین عدد c است، وگرنه فرض قبلی صحیح بوده است، بدین ترتیب با انجام این گام‌ها کوچک‌ترین عدد موجود پیدا شده و می‌توانیم آن را در خروجی نمایش دهیم.

الگوریتمی بنویسید که ۳ عدد را دریافت کرده، سپس کوچک‌ترین و بزرگ‌ترین مقدار را مشخص کند.

۱. شروع
۲. a و b و c را دریافت کن
۳. a را در min قرار بده
٤. a را در max قرار بده
٥. اگر b<min آنگاه b را در min قرار بده
٦. اگر b>max آنگاه b را در max قرار بده
۷. اگر c<min آنگاه c را در min قرار بده
۸. اگر c>max آنگاه c را در max قرار بده
۹. min و max را نمایش بده
۱۰. پایان

توضیح این الگوریتم: این مثال نیز دقیقا مشابه با مثال قبلی است، با این تفاوت که یک متغیر max نیز به آن اضافه می‌کنیم تا فرایند تشخیص عدد بزرگ‌تر را انجام دهد، بدیهی است که برای پیدا کردن عدد بزرگ‌تر باید دقیقا برعکس حالت قبل عمل کنیم.

الگوریتمی بنویسید که نمرات چهار درس ریاضی، فیزیک، زبان و ورزش یک دانش‌آموز را دریافت کرده، سپس میانگین آنها را محاسبه کند و با توجه به شرایط زیر رتبه‌ی دانش‌آموز را محاسبه کرده و نمایش دهد .

- **میانگین بزرگ‌تر از ۱۸ برابر رتبه‌ی A**
- **میانگین بین ۱۶ تا ۱۸ و ریاضی بزرگ‌تر از ۱۷ برابر رتبه‌ی B**
- **میانگین بین ۱۴ تا ۱۶ و ریاضی یا فیزیک بزرگ‌تر از ۱۵ برابر رتبه C**
- **میانگین کمتر از ۱۲ و ریاضی و فیزیک کمتر از ۱۲ برابر رتبهE**

۱. شروع
۲. riazi و fizik و zaban و varzesh را دریافت کن
۳. riazi + fizik + zaban + varzeshرا در sum قرار بده
٤. sum/٤ را در avg (میانگین) قرار بده
٥. اگر (avg>۱۸) رتبه‌ی A را نمایش بده
٦. اگر (avg>۱٦) و (avg<=۱۸) و (riazi>۱۷) رتبه‌ی B را نمایش بده
۷. اگر (avg>۱٤) و (avg<=۱٦) و (riazi>۱٥) یا (fizik>۱٥) رتبه‌ی C را نمایش بده
۸. اگر (avg<۱۲) و (riazi<۱۲) و (fizik<۱۲) رتبه‌ی E را نمایش بده
۹. پایان

توضیح این الگوریتم: یکی از الگوریتم‌های کاربردی که یادگیری آن می‌تواند در برنامه‌نویسی به شما کمک کند، الگوریتم مثال بالاست، فرض کنید در حال نوشتن یک برنامه‌ی محاسبه‌ی معدل برای یک مدرسه هستید، این الگوریتم دقیقا این فرایند را برای شما انجام می‌دهد.

شاید در نگاه اول کمی گیج‌کننده به نظر برسد، اما هنگامی که در یک الگوریتم حرف از و (and منطقی) یا (or منطقی) به میان آید شما باید همزمان چندین شرط را بررسی کنید.

یکی از این شرایط زمانی است که قصد داریم یک بازه‌ی عددی را بررسی کنیم، به عنوان مثال وقتی می‌گوییم یک عدد بین ١٦ تا ١٨ باشد، یعنی به طور همزمان از ١٦ بزرگ‌تر و در عین حال از ١٨

کوچک‌تر باشد، پس در چنین شرایطی برای اینکه بتوانیم به طور همزمان دو حالت را بررسی کنیم باید از and منطقی استفاده کرده تا تنها در صورت برقرار بودن هر دو شرط دستور مورد نظر اجرا شود.

الگوریتمی بنویسید که ضرایب یک معادله‌ی درجه‌ی ۲ را دریافت کرده و ریشه‌های آن را محاسبه کند.

۱. مقادیر a و b و c را دریافت کن (ضرایب معادله)
۲. مقدار $b^{۲}$-٤ac را در D قرار بده (منظور از D دلتاست)
۳. اگر D<۰ معادله ریشه‌ی حقیقی ندارد
٤. اگر D=۰ مقدار -b/۲a را نمایش بده
۵. اگر D>۰ مقدار -b+√D/۲a و -b-√D/۲a را نمایش بده
٦. پایان

می‌دانیم که ظاهر یک معادله‌ی درجه‌ی ۲ به شکل $ax^{۲}+bx+c$ است، ما تنها با داشتن مقادیر a و b و c می‌توانیم پاسخ‌های معادله را به دست آوریم، برای این منظور تنها کافیست مقدار دلتا را محاسبه کرده و با توجه به آن تصمیم بگیریم، اگر دلتا منفی شد بدین معناست که معادله‌ی ما جواب ندارد، اگر دلتا برابر با ۰ شد، یعنی معادله یک ریشه دارد که آن‌هم برابر است با **(-b/۲a)** و اگر مقدار دلتا مثبت شد، یعنی معادله ۲ ریشه دارد، که عبارتند از **(-b+√D/۲a)**و **(-b-√D/۲a)**

فرمول و روابط مورد نیاز برای حل این الگوریتم را نیز می‌توانید به راحتی با یک جستجو در اینترنت به‌دست آورید، پیدا کردن راه حل ریاضی مسائل جزو وظایف یک برنامه‌نویس نیست، برنامه‌نویس‌ها باید راه‌حل خلق کنند!

فصل چهارم

الگوریتم دستورات تکرار (حلقه‌ها)

به زبان ساده حلقه‌ها دستوراتی هستند که یک فرایند به خصوص را به تعداد دلخواه ما تکرار می‌کنند، بدین ترتیب لازم نیست یک دستور بسیار ساده و ابتدایی را ۲۰۰۰ بار بنویسیم، بلکه تنها یک بار آن را نوشته و عملیات تکرار را به خود کامپیوتر می‌سپاریم.

دستور تکرار (حلقه) از ۴ بخش اصلی تشکیل می‌شود:

شمارنده: متغیری عددی که تعداد دفعات تکرار را کنترل می‌کند و مقدار آن در هر بار اجرای حلقه افزایش یا کاهش می‌یابد.

مقدار اولیه: مقداری که قبل از شروع حلقه برای شمارنده تعیین می‌شود.

شرط حلقه: شمارنده را کنترل کرده و پایان تکرار (خروج از حلقه) را مشخص می‌کند.

دستورات حلقه: دستورات مورد نظر ما که در حلقه اجرا (و تکرار) می‌شوند.

الگوریتمی بنویسید که ۱۰۰۰ بار عبارت Hello را نمایش دهد.

۱. شروع
۲. عدد i را برابر ۱ قرار بده
۳. عبارت Hello را نمایش بده
٤. i+۱ را در i قرار بده (یک واحد به مقدار i اضافه کن)
٥. اگر i<=۱۰۰۰ به مرحله‌ی ۳ برو
٦. پایان

توضیح این الگوریتم: مثال بالا یک نمونه الگوریتم بسیار ساده از دستورات تکرار است، در چنین شرایطی به‌جای اینکه یک دستور را ۱۰۰۰ بار تکرار کنیم، از حلقه استفاده می‌کنیم.

ساز و کار حلقه‌ها به این صورت است که، یک شمارنده برای آن‌ها در نظر می‌گیریم (در این مثال متغیر i) و یک مقدار اولیه به آن می‌دهیم، سپس دستور مورد نظر که قصد داریم تکرار شود را در گام بعد قرار می‌دهیم، حال برای اینکه حلقه شروع به کار کند، در گام بعد ۱ واحد به آن اضافه می‌کنیم (i++) و پس از آن بررسی می‌کنیم که آیا مقدار شمارنده به عدد تکرار مورد نظر ما رسیده است؟ اگر به مقدار مد نظر نرسیده بود، مجددا نقطه‌ی شروع الگوریتم را به دستور تکراری باز می‌گردانیم تا این مراحل مجددا تکرار شده و دستور مورد نظر اجرا شود.

اگر هم مقدار شمارنده به تعداد مورد نظر رسیده باشد، بدین معناست که دستورات به تعداد دلخواه تکرار شده‌اند و می‌توانیم به ادامه‌ی الگوریتم بازگردیم.

الگوریتمی بنویسید که عدد N را از ورودی دریافت کرده به همان تعداد عبارت Hello را نمایش دهد.

۱. شروع
۲. عدد N را دریافت کن
۳. عدد i را برابر ۱ قرار بده
٤. عبارت Hello را نمایش بده
٥. i+۱ را در i قرار بده (یک واحد به مقدار i اضافه کن)
٦. اگر i<=N به مرحله‌ی ٤ برو
۷. پایان

الگوریتمی بنویسید که اعداد ۱ تا ۱۰۰۰ را نمایش دهد.

۱. شروع
۲. عدد i را برابر ۱ قرار بده
۳. مقدار i را نمایش بده
٤. یک واحد به مقدار i اضافه کن
۵. اگر i<=۱۰۰۰ به مرحله‌ی ۳ برو
٦. پایان

این الگوریتم تفاوت چندانی با مثال قبل ندارد، فقط در اینجا به‌جای یک عبارت ثابت خود متغیری که به عنوان شمارنده در نظر گرفته‌ایم را در خروجی نمایش می‌دهیم، هنگامی که در مرحله‌ی ٤ یک واحد به عدد i اضافه می‌کنیم، هنگام بازگشت به مرحله‌ی ۳ عدد i جدید نمایش داده خواهد شد و در هربار اجرای حلقه یک واحد به شمارنده اضافه شده و مجددا نمایش داده می‌شود، بدین ترتیب با این روش به هدف الگوریتم خواهیم رسید.

الگوریتمی بنویسید که اعداد ۱۰۰۰ تا ۱ را نمایش دهد.

۱. شروع
۲. عدد i را برابر ۱۰۰۰ قرار بده
۳. مقدار i را نمایش بده
٤. یک واحد از مقدار i کم کن
٥. اگر i>=۱ به مرحله‌ی ۳ برو
٦. پایان

مثال بالا نیز تفاوت چندانی با الگوریتم‌های قبلی ندارد، تنها نحوه‌ی کار شمارنده را تغییر داده و به جای اینکه از ۱ شروع کنیم از ۱۰۰۰ شروع خواهیم کرد و در مرحله‌ی ٤ نیز یک واحد از مقدار آن کم می‌کنیم، همچنین در مرحله‌ی چک شرط چون اعداد از مقدار زیاد به کم حرکت می‌کنند پس تا زمانی دستورات را تکرار می‌کنیم که مقدار شمارنده از ۱ کمتر شود.

الگوریتمی بنویسید که ۱۰۰ عدد مختلف را دریافت کرده و زوج یا فرد بودن آن‌ها را بررسی کند.

۱. شروع
۲. عدد i را برابر ۱ قرار بده
۳. مقدار x را از ورودی دریافت کن
٤. اگر باقی‌مانده‌ی x/۲ برابر صفر بود عبارت "زوج" را نمایش بده، در غیر این‌صورت عبارت "فرد" را نمایش بده.
۵. یک واحد به مقدار i اضافه کن
٦. اگر i<=۱۰۰ آنگاه به مرحله‌ی ۳ برو
۷. پایان

توضیح این الگوریتم: حتما لازم نیست در هربار اجرای حلقه تنها یک دستور انجام شود، شما می‌توانید در بدنه‌ی حلقه‌ی خود هر تعداد دستور تکراری را بنویسید، به عنوان مثال اینجا دریافت یک عدد جدید و چک کردن زوج یا فرد بودن آن جزو دستورات تکراری به حساب می‌آیند و می‌توانیم تمام آن‌ها را در بدنه‌ی حلقه بنویسیم تا به صورت خودکار ۱۰۰ بار اجرا شوند.

الگوریتمی بنویسید که تا زمانی که کاربر عددی منفی را وارد نکرده همچنان از او عدد گرفته و عبارت "مثبت" را نمایش دهد.

۱. شروع
۲. عدد n را دریافت کن
۳. اگر n>=۰ آنگاه "مثبت" را نمایش بده و به مرحله‌ی ۲ برو
٤. پایان

در بعضی از حلقه‌ها احتیاجی به شمارنده نداریم، بلکه شرط توقف با توجه به نیاز الگوریتم تعیین می‌شود، مانند مثال بالا که شرط توقف آن وارد کردن عدد منفی توسط کاربر است، اگر کاربر تا ابد عدد مثبت وارد کند این حلقه هیچگاه از حرکت نایستاده و تا بی‌نهایت ادامه خواهد داشت.

الگوریتمی بنویسید که اعداد ۵۰۰- تا ۵۰۰+ را دوتا دوتا نمایش دهد.

۱. شروع
۲. مقدار i را برابر ۵۰۰- قرار بده
۳. عدد i را نمایش بده
٤. دو واحد به i اضافه کن
۵. اگر i<=۵۰۰ به مرحله‌ی ۳ برو
٦. پایان

مقدار افزایش شمارنده‌ی حلقه حتما نباید ۱ واحد افزایش داشته باشد، با توجه به نیاز الگوریتم شما باید مقدار افزایش یا کاهش را تشخیص داده و آن‌را روی شمارنده‌ی حلقه اعمال کنید، به عنوان مثال در الگوریتم بالا لازم داشتیم که شمارنده‌ی حلقه هربار ۲ واحد افزایش پیدا کند.

الگوریتمی بنویسید که جمع اعداد ۱ تا ۱۰۰۰ را محاسبه کرده و نمایش دهد.

۱. شروع
۲. مقدار sum را برابر ۰ قرار بده
۳. مقدار i را برابر ۱ قرار بده
٤. مقدار i+sum را در sum قرار بده
۵. یک واحد به i اضافه کن
٦. اگر i<=۱۰۰۰ به مرحله‌ی ٤ برو
۷. مقدار sum را نمایش بده
۸. پایان

توضیح این الگوریتم: در مثال بالا باید یک متغیر کمکی در نظر بگیریم تا در هر مرحله مقادیر محاسبه شده را در آن نگهداری کرده و مقدار جدید شمارنده را نیز به آن اضافه کنیم، این متغیر کمکی sum (به معنای جمع) نام دارد، در هربار اجرای حلقه، مقدار i را با مقدار قبلی sum جمع کرده و مجددا آن را در خود متغیر sum ذخیره می‌کنیم، بدین ترتیب با اجرای کامل حلقه کل اعدادی که توسط شمارنده تولید می‌شود را با هم جمع کرده‌ایم و می‌توانیم مجموع آن‌ها را نمایش دهیم.

الگوریتمی بنویسید که مجموع و تعداد اعداد طبیعی مضرب ۳ و کوچک‌تر از ۱۰۰ را نمایش دهد.

۱. شروع
۲. مقدار sum و count را برابر ۰ قرار بده
۳. مقدار i را برابر ۳ قرار بده
٤. مقدار sum+i را در sum قرار بده
٥. یک واحد به مقدار count اضافه کن
٦. مقدار i+۳ را در i قرار بده
۷. اگر i<۱۰۰ به مرحله‌ی ٤ برو
۸. مقدار count و sum را نمایش بده
۹. پایان

توضیح این الگوریتم: بیایید در ابتدا مضرب‌های اعداد ۳ را باهم مرور کنیم:

این اعداد عبارتند از: ۳ - ٦ - ۹ - ۱۲ - ۱۵ - ۱۸ و...

چه رابطه‌ای میان این اعداد می‌بینیم؟

۳ واحد ۳ واحد افزایش دارند، این مهم‌ترین نکته‌ی این الگوریتم است، با دانستن این نکته به راحتی می‌توانید الگوریتم صحیح را بنویسید.

پس اگر در ابتدای امر نوشتن الگوریتم برایتان دشوار بود، سعی کنید آن را روی یک کاغذ نوشته و یک رابطه‌ی منطقی بین اجزای آن بیابید، مطمئن باشید با این روش به راحتی می‌توانید از پس حل تمام الگوریتم‌های موجود در دنیای کامپیوتر برآیید، تنها کافیست یک رابطه‌ی منطقی صحیح پیدا کنید.

در این الگوریتم از متغیر sum برای نگهداری مجموع اعداد و از متغیر count برای نگهداری اعداد شمارش شده استفاده می‌کنیم و در هربار اجرای حلقه ۱ واحد به آن می‌افزاییم تا تعداد اعداد موجود در این بازه را به دست آوریم.

آیا شما نیز می‌توانید یک راه حل جدید و مبتکرانه برای حل این الگوریتم پیشنهاد دهید؟!

راهنمایی :تمام اعداد مضرب سه، بر ۳ بخش‌پذیرند.

الگوریتمی بنویسید که ۲ عدد را از ورودی دریافت کرده و عدد اول را به توان عدد دوم برساند (از علامت توان استفاده نکنید)

۱. شروع
۲. عدد a و b را دریافت کن
۳. مقدار i را برابر ۱ قرار بده
٤. مقدار result را برابر ۱ قرار بده
٥. مقدار result*a را در result قرار بده
٦. یک واحد به i اضافه کن
۷. اگر i<=b به مرحله‌ی ٥ برو
۸. مقدار result را نمایش بده
۹. پایان

توضیح این الگوریتم: توان یعنی ضرب‌های متوالی یک عدد! به عنوان مثال $۲^{٤}$ برابر است با ۲*۲*۲*۲ یعنی ۱٦، تمام عداد توان‌دار با این شیوه محاسبه می‌شوند.

در این الگوریتم قصد داریم حاصل a^{b} را به‌دست آوریم، بنابراین باید b بار عدد a را در خودش ضرب کنیم. به همین علت یک حلقه در نظر گرفته و هربار در آن a را در خودش ضرب می‌کنیم و مقدار آن را در result ذخیره می‌کنیم تا در دور بعد حلقه مجددا بتوانیم مقدار a را در مقادیر قبلی ضرب کنیم، این فرایند را b بار تکرار می‌کنیم تا به نتیجه‌ی صحیح برسیم!

نکته: اگر دقت کرده باشید در این مثال مقدار اولیه result را برابر ۱ قرار دادیم، اما در مثال‌های قبلی مقدار اولیه‌ی sum را برابر ۰ قرار دادیم، تفاوت این دو عدد در این است که در مثال بالایی درحال انجام عملیات ضرب هستیم و اگر مقدار اولیه result برابر ۰ قرار بگیرد، حاصل تمام ضرب‌های بعدی ۰ شده و الگوریتم نتیجه‌ی نادرستی را تولید می‌کند، اما در مثال قبل‌تر مقدار sum را برابر ۰ می‌گذاریم، چراکه در حال انجام عملیات جمع هستیم.

پس این نکته را به یاد داشته باشید که عدد ۰ عضو خنثی در جمع و عدد ۱ عضو خنثی در ضرب می‌باشد.

اما به عنوان آخرین مثال این فصل، به سراغ الگوریتم محاسبه‌ی فاکتوریل می‌رویم.

الگوریتمی بنویسید که مقدار !n (n فاکتوریل) را محاسبه کرده و نمایش دهد.

۱. شروع
۲. عدد n را دریافت کن
۳. مقدار result را برابر ۱ قرار بده
٤. مقدار i را برابر ۱ قرار بده
٥. مقدار result*i را در result قرار بده
٦. یک واحد به i اضافه کن
۷. اگر i<=n به مرحله‌ی ٥ برو
۸. مقدار result را نمایش بده
۹. پایان

توضیح این الگوریتم: در ابتدا بیایید نگاهی به روش حل فاکتوریل داشته باشیم، فاکتوریل هر عدد از حاصل ضرب کل اعداد قبلی به دست می‌آید.

به عنوان مثال فاکتوریل عدد ٥ برابر است با: ۱*۲*۳*٤*٥ = ۱۲۰

اگر به نمونه‌ی بالا دقت کنید، در میابید که اعداد هر بار ۱ واحد افزایش داشته‌اند و در هر مرحله در عدد قبل از خود ضرب شده‌اند، این فرایند تا عدد ٥ (یعنی مقدار n ورودی) تکرار شده است، و این دقیقا سازوکار یک حلقه‌ی برنامه‌نویسی است!

فصل پنجم

آشنایی با مفهوم فلوچارت (روندنما)

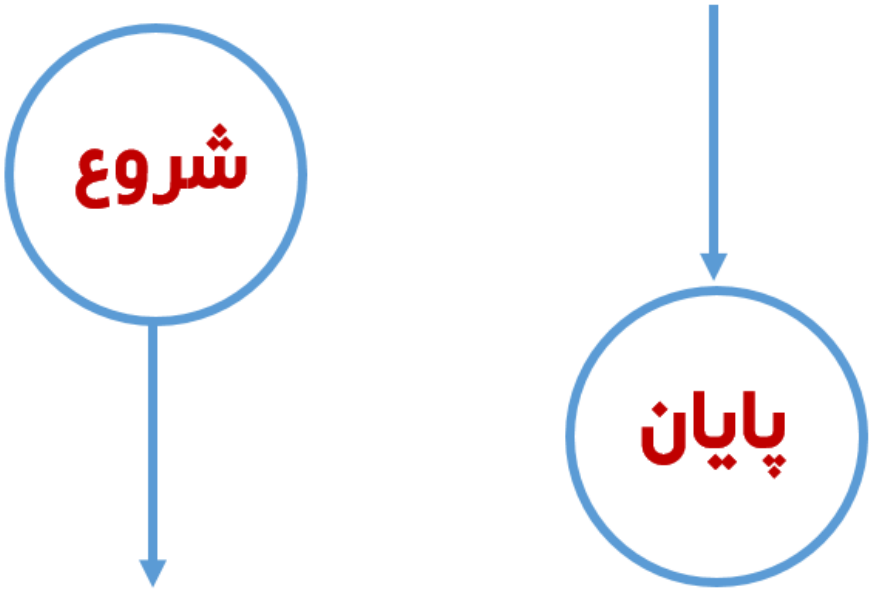
فلوچارت مجموعه‌ای از شکل‌های قراردادی است که دستورالعمل‌ها و ترتیب اجرای آن‌ها را مطابق با الگوریتم مورد نظر نمایش می‌دهد.

در حقیقت فلوچارت نمای تصویری الگوریتم است، لازمه‌ی رسم یک فلوچارت این است که شما به خوبی با مفهوم الگوریتم آشنا باشید، پس اگر فصل‌های قبل را مطالعه نکرده‌اید در ابتدا آن‌ها خوانده و سپس ادامه‌ی بخش فلوچارت را مطالعه کنید.

همانطور که گفتیم فلوچارت‌ها مجموعه ای از اشکال قراردادی هستند، یعنی شکل‌هایی ثابت که هرکدام معنا و مفهوم به خصوصی دارند، این اشکال عبارتند از:

- دایره
- متوازی‌الاضلاع
- مستطیل
- لوزی
- چهارضلعی منحنی‌دار

دایره

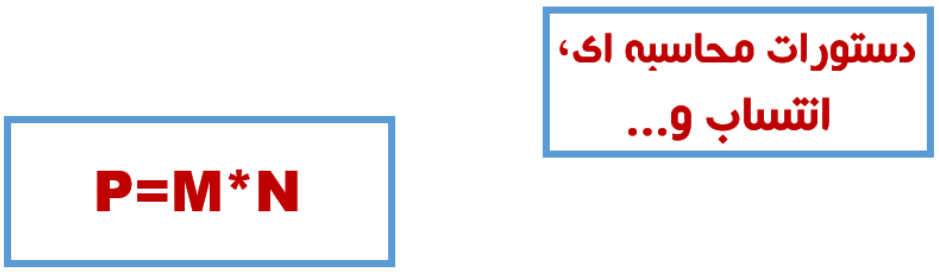


هنگامی که قصد دارید نقطه‌ی آغاز یا پایان یک فلوچارت را نمایش دهید باید از نماد دایره استفاده کنید، به یاد داشته باشید که در فلوچارت تنها باید یک نقطه‌ی شروع وجود داشته باشد، اما در تعداد نقاط پایان محدودیتی برای شما وجود ندارد، چراکه یک برنامه می‌تواند چندین حالت مختلف برای پایان داشته باشد اما همواره تنها یک حالت شروع خواهد داشت.

متوازی‌الاضلاع

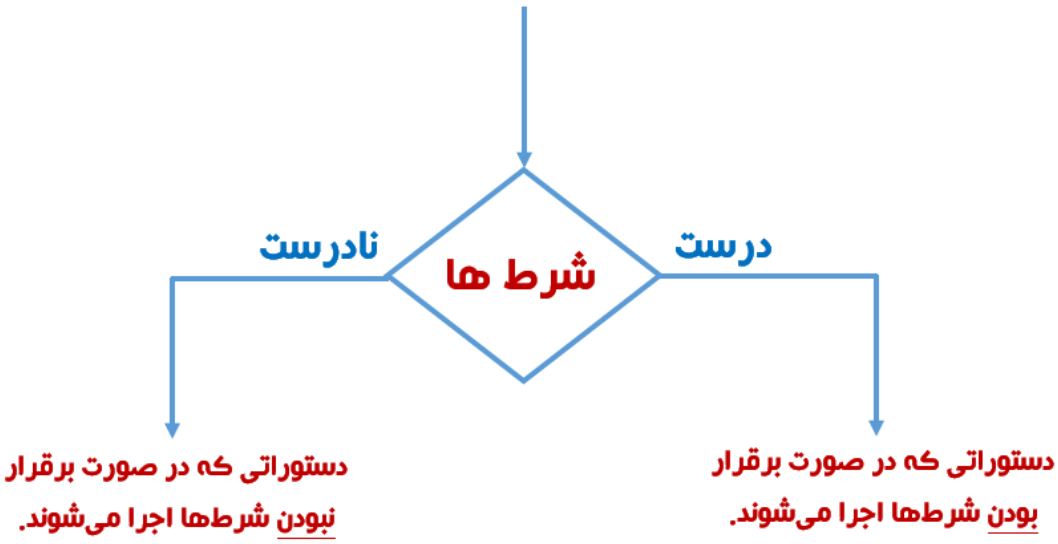
هنگامی که قصد دارید، مقداری را از کاربر یا جایی در خارج از برنامه دریافت کنید باید نام آن متغیرها را در علامت متوازی‌الاضلاع نمایش دهید.

مستطیل



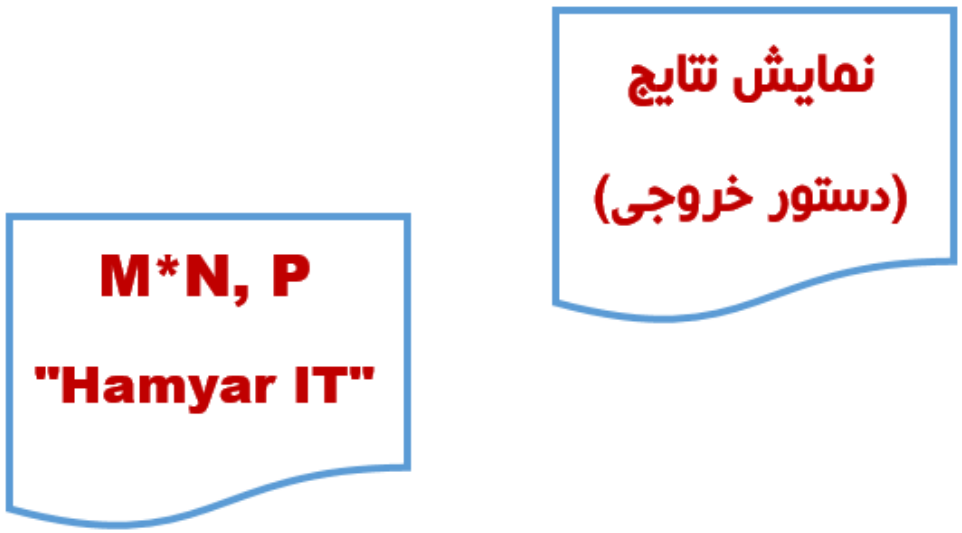
یک برنامه‌ی کامپیوتری پر از دستورات محاسباتی و انتساب (Assign) است، بنابراین احتمالا به دفعات فراوان از تصویر مستطیل در فلوچارت خود استفاده خواهید کرد!

لوزی



دستورات شرطی نیز معمولا زیاد در یک برنامه‌ی کامپیوتری استفاده می‌شوند، همانطور که در تصویر اشاره کردیم حلقه‌های تکرار نیز در برنامه‌نویسی با استفاده از همین لوزی‌ها نمایش داده می‌شوند.

چهارضلعی منحنی‌دار



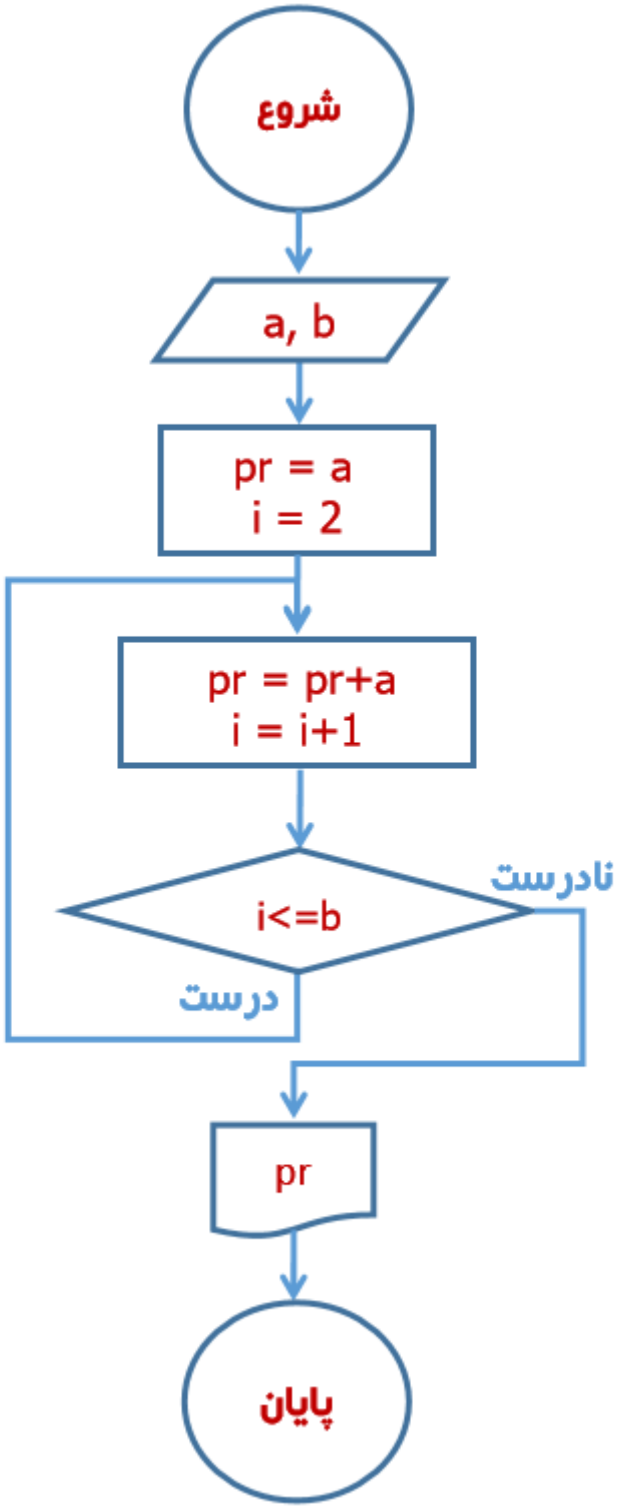
در نهایت هدف اصلی یک برنامه نمایش مقدار خروجی به کاربر است، بنابراین معمولا پس از انجام محاسبات فراوان و پردازش‌های مختلف روی داده‌های وروری یک مقدار خروجی تولید خواهد شد، در هنگام رسم فلوچارت این مقادیر را با شکل بالا (یک چهار ضلعی منحنی‌دار) نمایش می‌دهیم.

حال که با مفاهیم ابتدایی فلوچارت آشنا شدیم، بیایید چند نمونه الگوریتم را در قالب فلوچارت رسم کنیم.

مثال: الگویتم و فلوچارتی بنویسید که عمل ضرب دو عدد طبیعی a*b را به کمک عملیات جمع محاسبه کرده و نمایش دهد.

- شروع
- a و b را دریافت کن.
- pr=a (a را در pr قرار بده)
- i=۲ (۲ را در i قرار بده)
- pr=pr+a (مجموع pr و a را در pr قرار بده)
- i=i+۱ (یک واحد به مقدار i اضافه کن)
- اگر i<=b به مرحله‌ی ۵ برو
- pr را نمایش بده
- پایان

پس از اینکه الگوریتم گام‌به‌گام یک مسئله را نوشتید می‌توانید به راحتی آن را به فلوچارت تبدیل کرده و در قالب اشکال نمایش دهید، فلوچارت این مثال را در صفحه‌ی بعد مشاهده کنید.

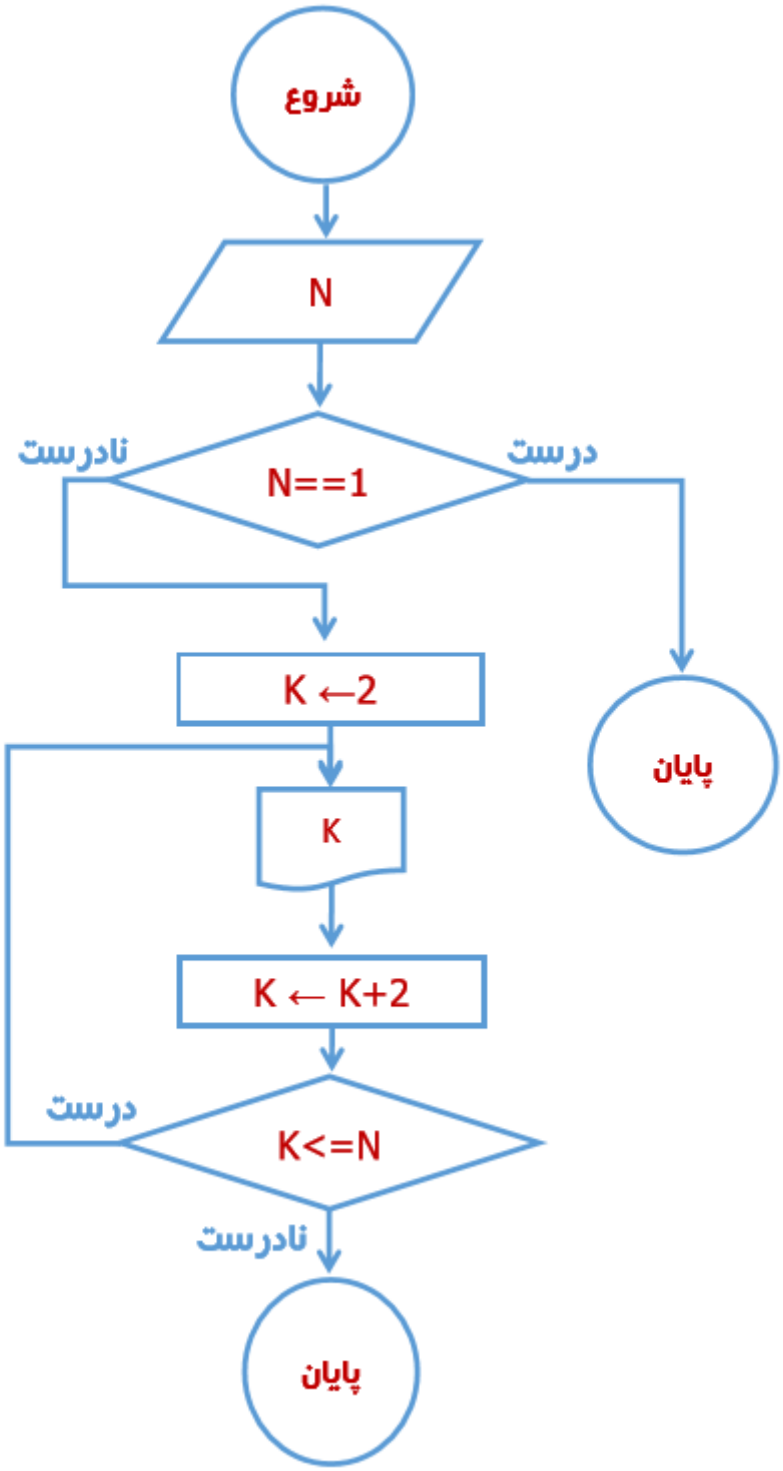

شروع
a, b
pr = a
i = 2
pr = pr+a
i = i+1
i<=b
نادرست
درست
pr
پایان

مثال: الگوریتم و فلوچارتی بنویسید که اعداد زوج کوچکتر یا مساوی عدد طبیعی N را نمایش دهد.

- شروع
- عدد N را دریافت کن
- اگر N==۱ (یعنی اگر N برابر با ۱ بود) آنگاه پایان
- K=۲
- K را نمایش بده
- K=K+۲
- اگر K<=N آنگاه به مرحله‌ی ۵ برو
- پایان

همانطور که پیش‌تر نیز گفتیم، پیش‌نیاز رسم یک فلوچارت آشنایی با مفهوم الگوریتم است، بنابراین اگر نتوانید به خوبی یک الگوریتم بنویسید در رسم فلوچارت نیز دچار مشکل خواهید شد.

فلوچارت مثال بالا را در صفحه‌ی بعد ببینید.


شروع
N
نادرست
N==1
درست
K ←2
پایان
K
K ← K+2
درست
K<=N
نادرست
پایان

بسیار خب، به پایان این آموزش ویژه رسیدیم، امیدواریم مطالعه‌ی این مسائل کمکی هرچند کوچک به بهبود جامعه‌ی برنامه‌نویسان آینده‌ی ایرانی کرده باشد، این تضمین را به شما می‌دهیم که اگر همین الگوریتم‌ها را چندین بار مطالعه کرده و خودتان آن‌ها را حل کنید تا آخر عمرتان برای شما مفید بوده و همواره در برنامه‌نویسی به کمک شما خواهند آمد.

چراکه شما با یادگیری این مباحث به صورت پایه‌ای با منطق برنامه‌نویسی آشنا شده‌اید و همواره می‌توانید دانش خود را در این زمینه به کار گرفته و به راحتی با هر زبان برنامه‌نویسی که به آن علاقه دارید کار کنید، به یاد داشته باشید پایه و اساس کار برنامه‌های کامپیوتری دقیقا همین الگوریتم‌های ساده و گام‌به‌گام است، بدون وجود الگوریتم‌ها هیچ برنامه‌ی بزرگی وجود نخواهد داشت.

در پایان اگر سوالی از این آموزش یا سایر آموزش‌های همیار آی‌تی دارید حتما به سایت به آدرس **Hamyarit.com** مراجعه کرده و دیدگاه‌های خود را با ما به اشتراک بگذارید، ما همواره پاسخ‌گوی شما عزیزان هستیم.