الگوریتم و فلوچارت

مولفان:
عبدالحمید جهانگیری
افشین امینی

به نام آن که جان را فکرت آموخت

الگوریتم و فلوچارت

عبدالحمید جهانگیری، افشین امینی

سرشناسه : جهانگیری، عبدالحمید، ۱۳۶۶ -
عنوان و نام پدیدآور : الگوریتم و فلوچارت/ تالیف عبدالحمید جهانگیری، افشین امینی.
مشخصات نشر : قم: سپهر حکمت، ۱۳۹۶.
مشخصات ظاهری : ۱۳۷ ص.؛ ۱۴/۵×۲۱/۵ س‌م.
شابک : ۹۷۸-۶۰۰-۹۹۶۶۷-۱-۴
وضعیت فهرست نویسی : فیپا
موضوع : الگوریتم‌ها
موضوع : Algorithms
موضوع : نمودار جریان کار
موضوع : Flow charts
شناسه افزوده : امینی، افشین، ۱۳۶۷ -
رده بندی کنگره : QA۵۸/۹/ج۹ف۷لا ۱۳۹۶
رده بندی دیویی : ۰۰۵/۱
شماره کتابشناسی ملی : ۵۰۶۳۳۸۳

مقدمه مولفین

در عصر حاضر بیشتر مسائلی که با آن روبرو هستیم بوسیله کامپیوتر حل می‌شود. سوالی که مطرح می‌شود این است که چگونه کامپیوتر می‌تواند این مسائل را حل کند؟

در پاسخ باید گفت که کامپیوتر بر اساس روش گام به گام حل مساله که به آن الگوریتم گفته می‌شود، مسائل را حل می‌کند. در نگاه اول شاید ممکن است به نظر برسد که کامپیوترها از انسانها باهوش‌ترند که می‌توانند همه‌ی مسائل را حل کنند. اما باید این نکته را در نظر داشت که برنامه‌های کامپیوتری توسط انسانها نوشته می‌شوند. انسانها با هوش و با درایت خویش مراحل گام‌به‌گام حل مساله را تدوین می‌کنند و کامپیوترها با ویژگی سرعت و دقت بالا دستورات را اجرا می‌کنند. بنابراین برای داشتن یک برنامه قدرتمند به یک الگوریتم دقیق نیاز دارید. کتاب حاضر چگونگی حل یک مساله و بیان مرحله به مرحله آن آموزش داده شده است. توصیه می‌شود که مطالب این کتاب را با دقت مطالعه کنید و همیشه این نکته را در نظر داشته باشید که پایه‌ی حل هر مساله‌ای در کامپیوتر، نوشتن الگوریتم آن است.

این کتاب حاصل تجربه سالها تدریس مولفین در درس الگوریتم و فلوچارت است و در نگارش آن سعی شده است که از همه‌ی تکنیکهای آموزشی برای انتقال بهتر مطالب به خواننده استفاده کند. از تمامی اساتید و دانشجویان عزیز خواهشمندیم که بر ما منت نهاده و از راهنماییهای خود جهت رفع عیوب کتاب دریغ ننمایند.

عبدالحمید جهانگیری – افشین امینی

سخنی با خوانندگان

در کتاب پیش‌رو سعی بر آن شده است که تمامی مفاهیم موردنیاز برای ورود به دنیای برنامه‌نویسی بیان شود و در این راه از مثالها و تمرینهای زیادی کمک گرفته شده است.

در ترتیب مثالها این هدف دنبال شده است که خواننده از ساده‌ترین مثال شروع به خواندن کند و هرچه جلوتر می‌رود با مفاهیم کاربردی‌تر در قالب مثالهای سخت‌تر آشنا شود. پس در صورتی که مثالی از این کتاب را متوجه نشدید برای خواندن مثالهای بعدی عجله نکنید. الگوریتم ها و فلوچارت‌های این کتاب به قدری از لحاظ آموزشی استاندارد نوشته شده است که با چند بار خواندن آنها و داشتن دقت و تمرکز کافی بر روی مثالها متوجه آنها خواهید شد.

در برنامه‌نویسی توانایی دنبال کردن الگوریتم می‌تواند به اندازه‌ی نوشتن الگوریتم اهمیت داشته باشد. به این معنی که اگر شما به الگوریتم برنامه‌ای دسترسی داشته باشید ولی هیچ توضیح نوشتاری در مورد الگوریتم در اختیارتان نباشد باید توانایی خواندن و یادگیری الگوریتم را داشته باشید. پس چنانچه در این کتاب مثالی را دیدید که توضیحات آن برای شما ناکافی به نظر می‌رسد می‌توانید به عنوان یک چالش به آن نگاه کنید! چالشِ دنبال کردن الگوریتم و یادگیری الگوریتم از روی فلوچارت آن.

فهرست مطالب

۱

فصل اول الگوریتم چیست؟

در این فصل با مفهوم کلی الگوریتم در قالب چند مثال ساده آشنا می‌شوید.

الگوریتم چیست؟

واژه الگوریتم بر گرفته از نام ابو جعفر محمد بن موسی الخوارزمی، ریاضی دان ایرانی است. الگوریتم، "روشی گام به گام برای حل مسئله" است. بسیاری از کارهای روزمره مانند وضو گرفتن، غذا پختن و ... از یک الگوریتم استفاده می کنند.[۱]

الگوریتم باید بدون ابهام نوشته شود تا توسط هر فرد یا کامپیوتری درک یکسانی صورت گیرد. به این منظور در نوشتن الگوریتم باید معیارهای زیر را در نظر گرفت:

۱. **ورودی:** یک الگوریتم باید صفر یا چند ورودی داشته باشد. به طور مثال در زمان کار با ماشین حساب، برای محاسبه حاصل ۲+۳ اعداد ۲ و ۳ و عمل جمع ورودی الگوریتم به شمار می روند.

۲. **خروجی:** الگوریتم باید صفر یا چند خروجی داشته باشد. به عنوان مثال خروجی مثال قبل ۵ است.

۳. **قطعیت:** دستورات باید با زبانی دقیق و بی ابهام نوشته شوند.

۴. **محدودیت:** الگوریتم باید دارای شروع و پایان مشخص باشد.

به این شیوه ی قدم به قدم انجام کار که همان الگوریتم است دقت کنید. اکنون شما قادر خواهید بود صدها مثال از زندگی روزمره خود را با استفاده از الگوریتم بیان نمایید.

---

[۱] هر چند تا قبل از این بدون دانستن نام الگوریتم قادر به حل یا انجام آنها بودید.

مثال ۱ الگوریتم وضو گرفتن

۱. شروع
۲. با دست راست، دو بار صورت را از بالا به پایین بشویید
۳. با دست چپ، دست راست را از آرنج تا نوک انگشتان بشویید.
۴. با دست راست، دست چپ را از آرنج تا نوک انگشتان بشویید.
۵. با دست راست، مسح سر را انجام دهید.
۶. با دست راست، مسح پای راست را بکشید.
۷. با دست چپ، مسح پای چپ را بکشید.
۸. پایان.

مثال ۲ الگوریتم تهیه چای

۱. شروع
۲. به اندازه یک قاشق چایخوری، چای خشک در قوری بریزید.
۳. یک یا دو لیوان آب جوش به آن اضافه کنید.
۴. قوری را روی حرارت ملایم قرار دهید.
۵. چند دقیقه صبر کنید.
۶. پایان.

در ادامه مثال های بیشتری مشاهده می کنید:

مثال ۳ الگوریتم جمع دو عدد

۱. شروع.
۲. عدد اول را بگیر و در x قرار بده.
۳. عدد دوم را بگیر و در y قرار بده.
۴. حاصل x+y را نمایش بده.
۵. پایان.

مثال ۴

الگوریتم برنامه ای بنویسید که دو عدد را دریافت نماید و بزرگترین آنها را نمایش دهد.

۱. شروع.
۲. عدد اول را بگیر و در x قرار بده.
۳. عدد دوم را بگیر و در y قرار بده.
۴. اگر $x \geq y$ آنگاه x را چاپ کن.
۵. اگر $y > x$، آنگاه y را چاپ کن.
٦. پایان.

همانطور که در این مثال دیدید، می توانیم در نوشتن الگوریتم ها از شرط‌ها (در این مثال اگر $x\geq y$ و $y>x$ شرط هایمان بودند) استفاده کنیم. به عنوان مثال در الگوریتم خروج خانه می گوییم اگر هوا بارانی بود چتر را ببر و اگر نبود نبر. حال الگوریتم مثال قبل را به صورت زیر بازنویسی می کنیم که با یکی از پر کاربردترین روش های الگوریتم آشنا شوید.

۱. شروع.
۲. عدد اول را بگیر و در x قرار بده.
۳. عدد دوم را بگیر و در y قرار بده.
۴. اگر $x\geq y$ آنگاه x را چاپ کن و برو به مرحله ی ۶.
۵. y را چاپ کن.
۶. پایان.

در اینجا، روند مثال آخر را دنبال می کنیم و به عنوان ورودی ۲۰ و ۱۱ را به برنامه می دهیم.

۱. شروع
۲. $x\leftarrow ۲۰$
۳. $y\leftarrow ۱۱$
۴. $x\geq y$
۵. این خط اجرا نمی شود، چون در مرحله ی قبل به برنامه گفته شده به مرحله ۶ برو.
۶. پایان.

تاکنون با الگوریتم‌های ساده‌ای آشنا شده‌اید. ادامه‌ی بحث که شامل حلقه ها و آرایه ها و ... می شود را بعد از آشنایی با فلوچارت مورد بررسی قرار می‌دهیم.

۲

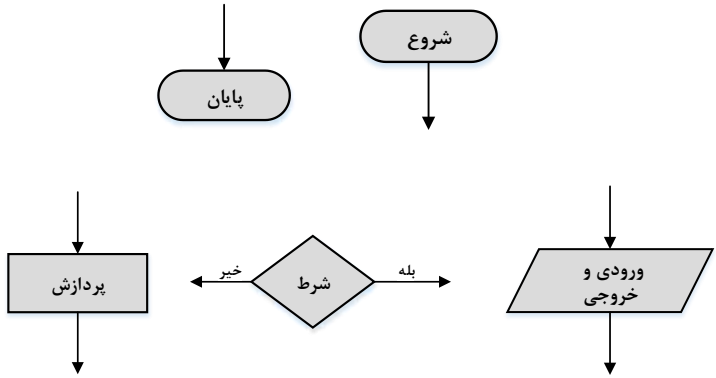
فصل دوم

فلوچارت

در این فصل نمادهای لازم برای معرفی فلوچارت بیان می‌شود و مثالهای زیادی برای درک بهتر موضوع مطرح می‌شود.

به دلیل قطعیت داشتن الگوریتم و بی ابهام بودن زبان ریاضی، برای بیان الگوریتم ، فلوچارت ساخته شد که با استفاده از نمادهای ریاضی سعی در بیان الگوریتم می‌نماید.

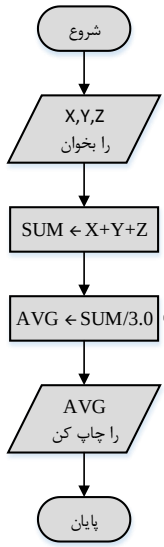
این نمودار جریان کار و روند دقیق حرکت داده ها را قدم به قدم نشان می دهد.



این نمادها دارای استانداردهای مختلفی می‌باشند. ممکن است در کتب دیگر، اشکال دیگری نیز ببینید.

مثال ۵

فلوچارت برنامه‌ای را رسم کنید که ۳ عدد به عنوان ورودی دریافت کند و میانگین آنها را چاپ کند.

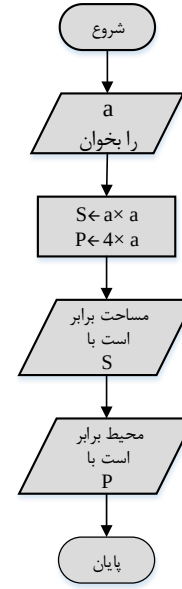


در مخرج کسر به این دلیل SUM را بر 3.0 تقسیم کردیم که نشان دهیم منظورمان تقسیم اعشاری است.

مثلا حاصل عبارت 11/3 که یک تقسیم صحیح است برابر 3 و حاصل عبارت 11/3.0 که یک تقسیم اعشاری است برابر 3.67 خواهد بود.

مثال ۶

فلوچارت برنامه‌ای را رسم کنید که طول ضلع مربعی را دریافت و محیط و مساحتش را چاپ کند.



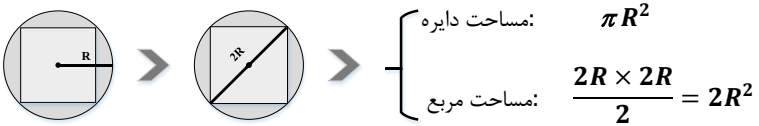
تمرین ۱

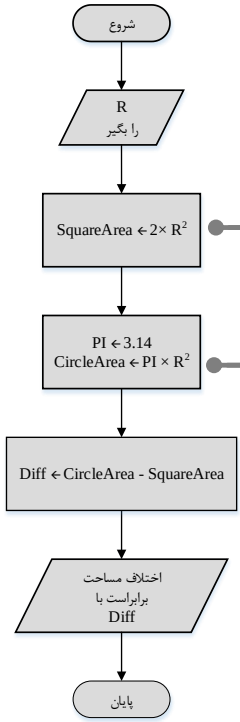
۱. فلوچارت برنامه‌ای را رسم کنید که طول و عرض مستطیلی را دریافت و محیط و مساحتش را چاپ کند.
۲. فلوچارت برنامه‌ای را رسم کنید که شعاع دایره ای را دریافت و محیط و مساحتش را چاپ کند.
۳. فلوچارت برنامه‌ای را رسم کنید که چهار عدد از ورودی دریافت کند و مجموع مکعبات آنها را چاپ کند.
۴. فلوچارت برنامه‌ای را رسم کنید که قاعده و ارتفاع یک مثلث قائم الزاویه را دریافت کند و محیط و مساحت مثلث را چاپ کند.

مثال ۷

فلوچارت برنامه‌ای را رسم کنید که شعاع دایره‌ای را دریافت و اختلاف مساحت دایره و مربع محاط شده را در آن چاپ کند.

ابتدا مساحت دایره و مربع محاط شده را جداگانه محاسبه می‌کنیم و در ادامه این دو مقدار را از هم کم می‌کنیم. (برای محاسبه مساحت مربع می‌توان به آن به صورت لوزی نگاه کرد! یعنی ضرب دو قطر تقسیم بر دو)





به مساحت مربع اشاره می‌کند.

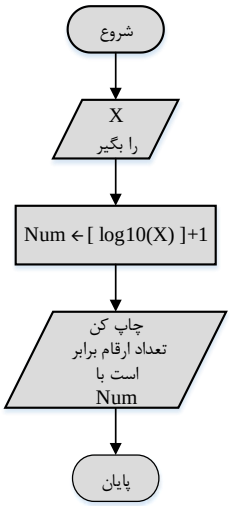
توجه داشته باشید که لزومی ندارد که اسامی فقط یك حرفی باشند. به این اسامی در برنامه نویسی متغیر گفته می‌شوند

به مساحت دایره اشاره می‌کند.

یکی از دغدغه دانشجویان در ایتدای راه برنامه‌نویسی انتخاب نام مناسب برای متغیرهاست. برنامه شما چه با نام متغیر SquareArea و چه با نام S به درستی کار می‌کند اما در برنامه های بزرگتر انتخاب نام مناسب خوانایی برنامه شما را به شدت بالا می‌برد. در مثالهای بعدی به نام متغیرها دقت کنید و سعی کنید در برنامه‌هایتان از نامهایی استفاده کنید که نام متغیر گویای محتوای متغیر باشد

مثال ۸

فلوچارت برنامه‌ای را رسم کنید که تعداد ارقام یک عدد را چاپ کند.

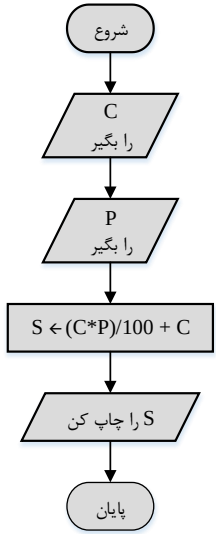


مثال ۹

فلوچارت برنامه‌ای را رسم کنید که قیمت خرید یک جنس و همچنین درصد سود فروشنده را دریافت و قیمت فروش را چاپ کند.

مثال ۱۰

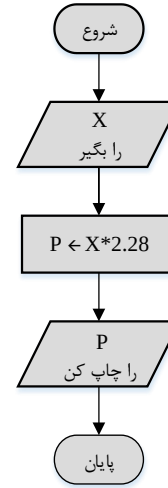
می دانیم که هر یک کیلوگرم معادل ۲/۲۰۸پوند است. فلوچارت برنامه‌ای بنویسید که وزن فرد را به کیلوگرم دریافت و مقدار آن را به پوند چاپ کند.



C نمایانگر قیمت خرید
P نمایانگر درصد سود
S نمایانگر قیمت فروش

فلوچارت مثال ۹

فلوچارت مثال ۱۰



تمرین ۲

۱. فلوچارت برنامه‌ای را رسم کنید که وزن را به پوند دریافت و مقدار آن را به کیلوگرم چاپ کند.

۲. فلوچارت برنامه‌ای را رسم کنید که قد فرد را به سانتیمتر دریافت و به اینچ چاپ کند. (هر سانتیمتر ۰/۳۹۴ اینچ است.)

۳. فلوچارت برنامه‌ای را رسم کنید که قد فرد را به اینچ دریافت و به سانتیمتر چاپ کند.

۴. فلوچارت برنامه‌ای را رسم کنید که درجه ی حرارت را از فارنهایت به سانتیگراد تبدیل کند. (می دانیم $\frac{C}{۱۰۰} = \frac{F-۳۲}{۱۸۰}$)

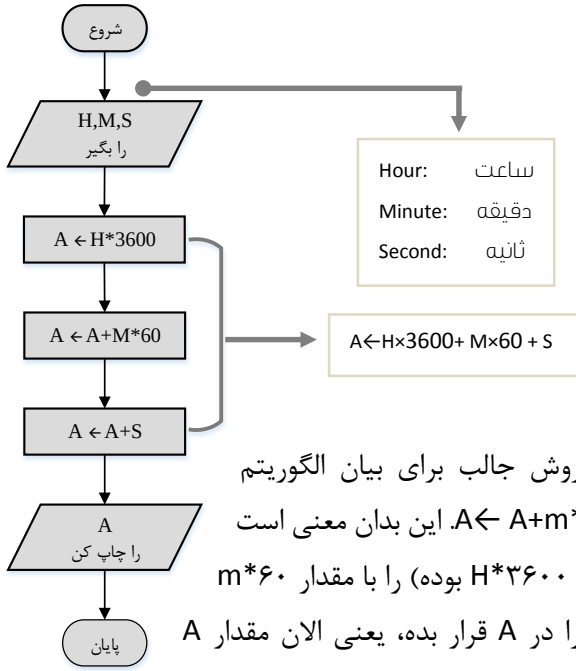
۵. فلوچارت برنامه‌ای را رسم کنید که درجه ی حرارت را از سانتیگراد به فارنهایت تبدیل کند.

۶. فلوچارت برنامه‌ای را رسم کنید که اختلاف مساحت دایره و مربع محیطی را رسم کند.

۷. فلوچارت برنامه‌ای را رسم کنید که اندازه ی قاعده و ارتفاع مثلث قائم الزاویه را بخواند، سپس مساحت مثلث را چاپ نماید.

مثال ۱۱

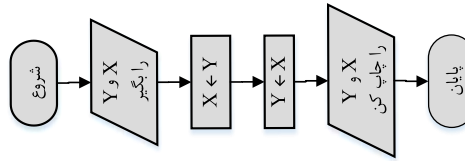
فلوچارت برنامه ای را رسم کنید که یک زمان مشخص بر حسب ساعت، دقیقه و ثانیه بگیرد و مشخص کند چند ثانیه از شروع روز گذشته است.



در مثال قبل یک روش جالب برای بیان الگوریتم مشاهده نمودیم. A← A+m*۶۰. این بدان معنی است که مقدار A (که قبلا H*۳۶۰۰ بوده) را با مقدار m*۶۰ جمع کن و حاصل را در A قرار بده، یعنی الان مقدار A برابر H*۳۶۰۰+M*۶۰ می باشد.

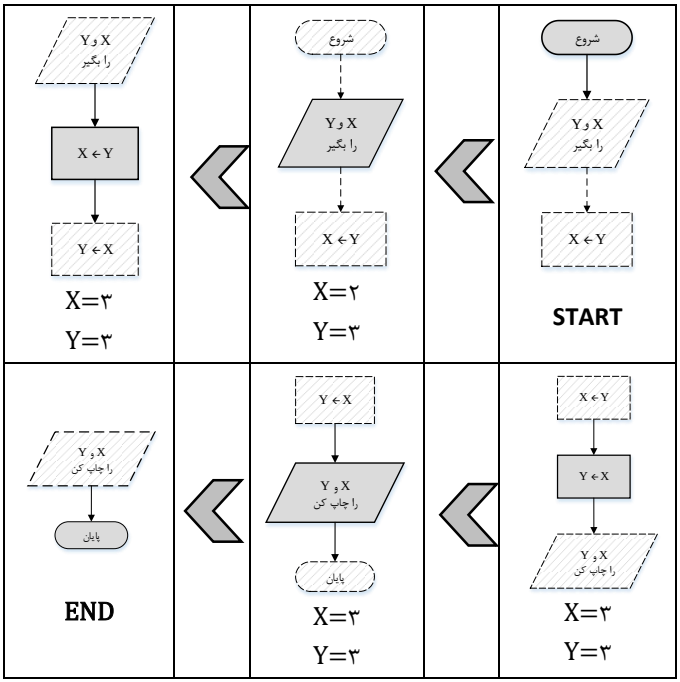
مثال ۱۲

فلوچارت برنامه‌ای را رسم کنید که دو مقدار را از ورودی خوانده و در x و y قرار دهد. سپس مقادیر x و y را با هم جا به جا کند.

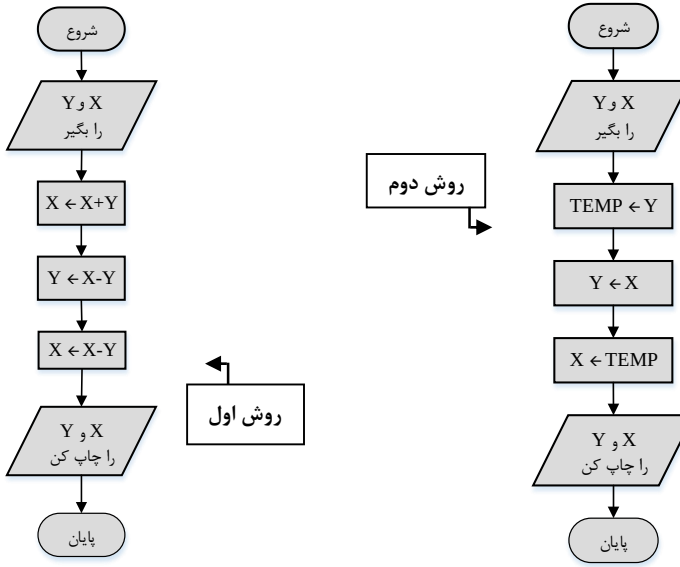


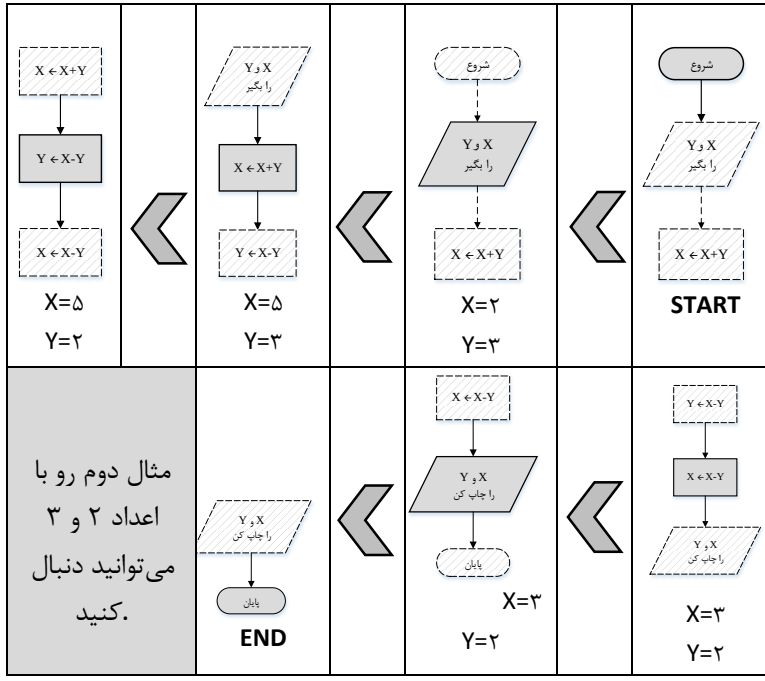
آیا فلوچارت بالا درست است؟

همانگونه که در ادامه خواهید دید، فلوچارت صفحه قبل اشتباه است! فلوچارت صفحه قبل را قدم به قدم از راست به چپ دنبال کنید:



در روند بالا، اعداد ۲ و ۳ از ورودی دریافت شده است و به ترتیب در متغیرهای X و Y ذخیره شده اند. این فلوچارت در انتها نمی‌تواند به درستی مقادیر X و Y را با هم عوض کند. برای رفع این مشکل، می‌توان این مساله را از دو روش حل کرد. روش اول با استفاده از یک متغیر کمکی و روش دوم بدون استفاده از متغیر کمکی و با استفاده از عملگر جمع و منها.


شروع
X و Y را بگیر
X ← X+Y
Y ← X-Y
X ← X-Y
Y و X را چاپ کن
پایان
روش اول
روش دوم
شروع
X و Y را بگیر
TEMP ← Y
Y ← X
X ← TEMP
Y و X را چاپ کن
پایان


START
X=۳
Y=۲
X=۲
Y=۳
X=۵
Y=۳
X=۵
Y=۲
END
X=۳
مثال دوم رو با اعداد ۲ و ۳ می‌توانید دنبال کنید.

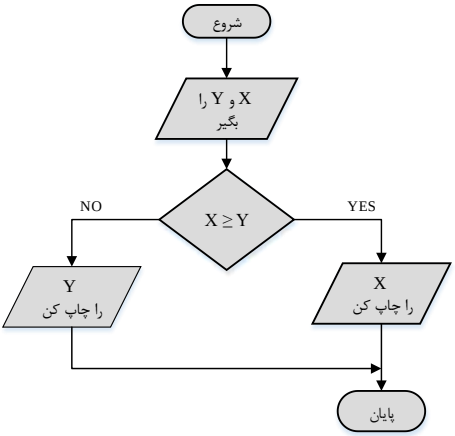
ممکن است این سوال برای شما پیش بیاید که چه نیازی است که دو متغیر دریافت کنیم و مقادیر این دو متغیر را جابجا کنیم؟!

در پاسخ باید گفت خیلی از مسائلی که الان مطرح می‌شوند به تنهایی ارزشی ندارند، اما در حل مسائل بزرگتر و سخت‌تر می‌توانند خیلی مفید واقع می‌شوند.

مثلا مساله جابجا کردن مقادیر دو متغیر به تنهایی یک مساله‌ی بی ارزش به نظر می رسد، اما همانطور که در ادامه خواهید دید در مساله مرتب سازی اعداد از این الگوریتم کاربردی استفاده می‌کنیم.

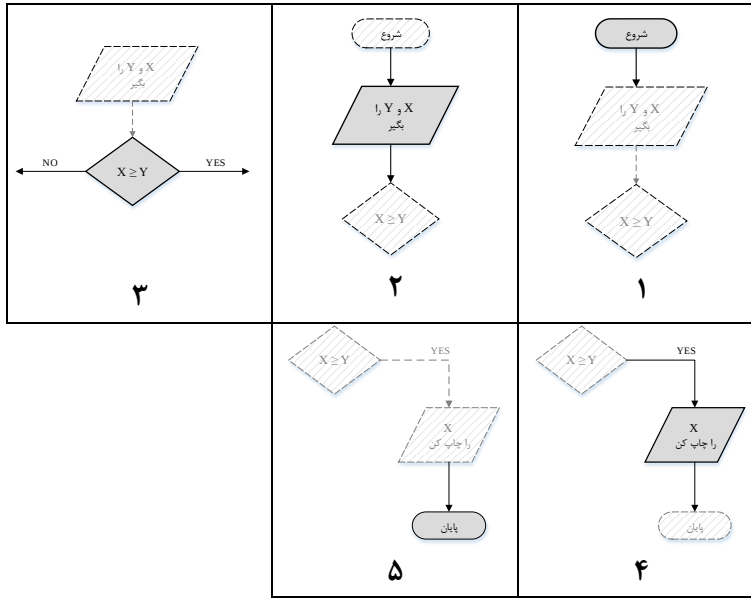
مثال ۱۳

فلوچارت مثال ۴ را رسم کنید.



در فلوچارت بالا به نحوه استفاده از شرط دقت کنید!

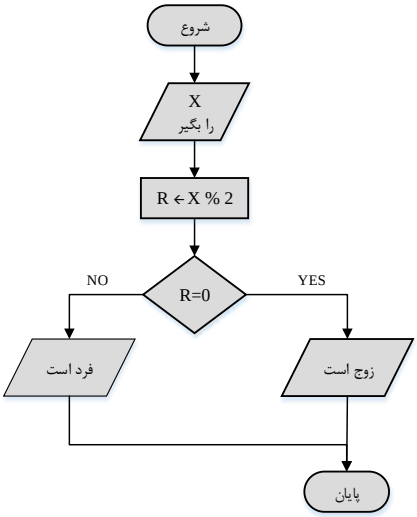
روند اجرای فلوچارت این مثال (در حالتی که$x \geq y$ باشد) به صورت زیر است:



مثال ۱۴

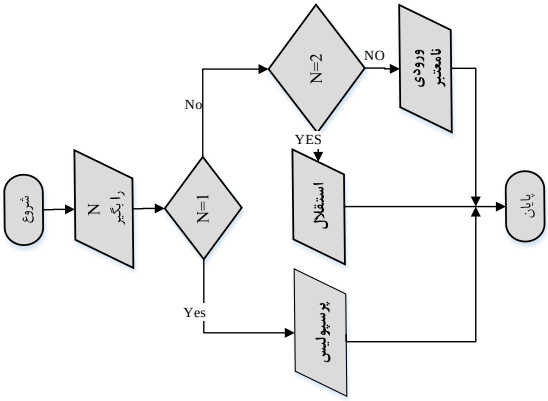
فلوچارت برنامه‌ای رسم کنید که یک عدد از ورودی بگیرد و زوجیت آن عدد را چاپ کند. (یعنی اگر عدد زوج است در خروجی "زوج است" را چاپ کند و اگر عدد فرد است در خروجی "فرد است" را چاپ کند.)

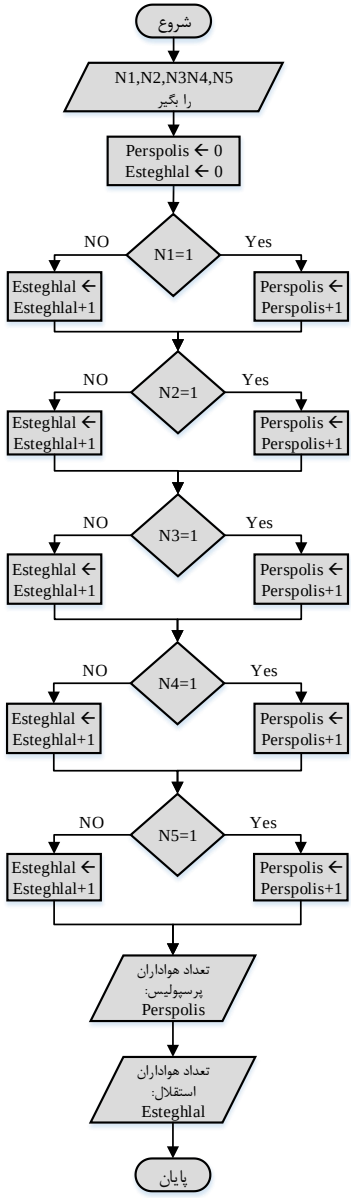
در این قسمت از عملگر % استفاده می‌کنیم که باقیمانده‌ی دو مقدار را به ما می‌دهد. مثلا ۳ % ۱۷ یعنی باقیمانده‌ی تقسیم عدد ۱۷ بر ۳ که برابر است با ۲. در بعضی از زبان‌های برنامه‌نویسی با MOD و در بعضی زبانها با % پیاده‌سازی شده است. همچنین توجه داشته باشید که در A%B، A و B حتما باید اعداد صحیح باشند و این عملگر برای اعداد اعشاری تعریف نشده است.



مثال ۱۵

فلوچارت برنامه‌ای رسم کنید که اگر کاربر عدد ۱ را وارد کرد در خروجی چاپ کند "پرسپولیس" و اگر ۲ را وارد کرد چاپ کند "استقلال" و اگر عددی به غیر از این وارد کند، در خروجی "ورودی نامعتبر است" را چاپ کند.

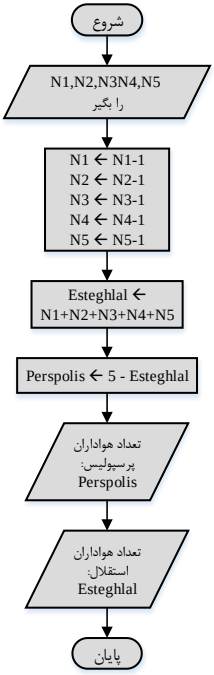




مثال ۱۶

فلوچارت برنامه‌ای رسم کنید که در آن پنج کاربر اعداد ۱ یا ۲ را به برنامه می‌دهند که ۱ یعنی هوادار پرسپولیس است و ۲ یعنی هوادار استقلال است. در نهایت تعداد طرفداران پرسپولیس و استقلال را چاپ کند.

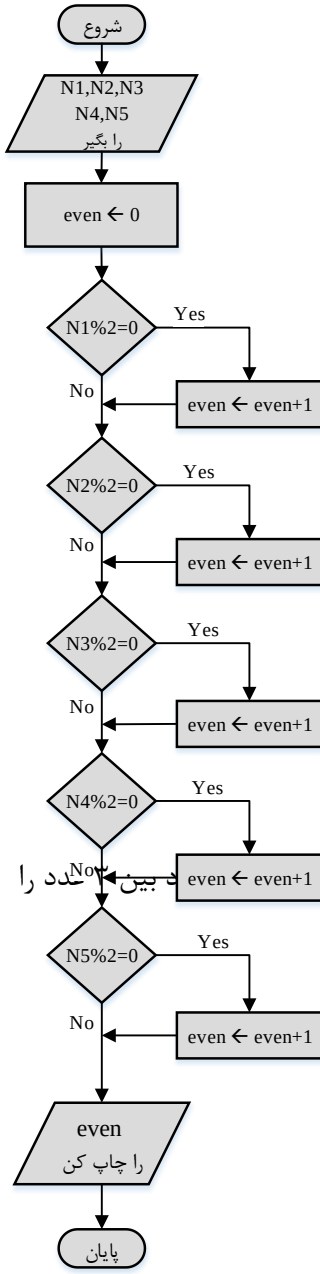
البته برای این مثال خاص، می‌توانستیم از یک ایده‌ی جالب هم استفاده کنیم که بدون هیچ شرطی جواب این مساله را بیابیم. ابتدا از n۱ تا n۵ هر کدام یک واحد کم کنیم، آن‌گاه مجموع n۱+...+n۵ برابر تعداد طرفداران استقلال و ۵-(n۱+...+n۵) برابر تعداد طرفداران پرسپولیس است. (چرا؟)



مثال ۱۷

فلوچارت برنامه‌ای رسم کنید که پنج عدد را از ورودی دریافت کند و تعداد اعداد زوج آنها را چاپ کند.

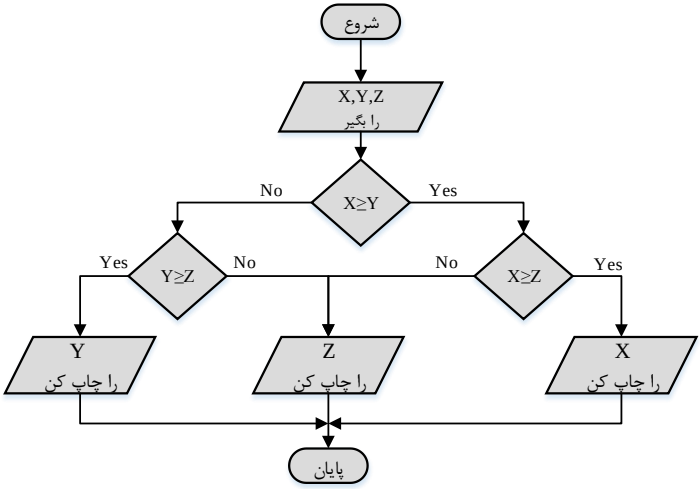
(even به معنی زوج و odd به معنی فرد است.)



مثال ۱۸

فلوچارت برنامه‌ای رسم کنید که ۲ عدد بین ۱ تا ۱۰ را چاپ کند.

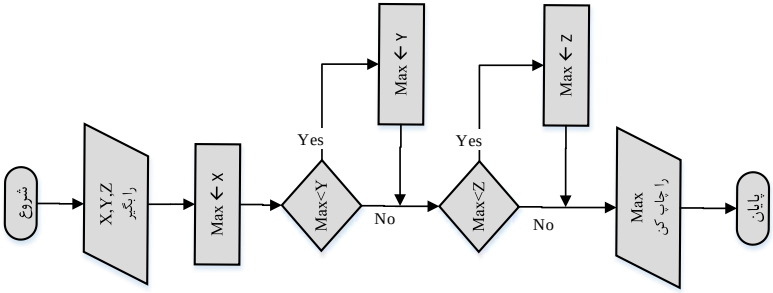
در مثال ۱۶ و ۱۷ برای پیدا کردن تعداد هواداران و تعداد اعداد زوج به متغیرهایی برای شمردن آنها نیاز داشتیم. اصطلاحا به این متغیرها **شمارنده** نیز گفته می‌شود و می‌توان با نام COUNTER نیز از آن استفاده کرد.



البته برای این مثال می‌توان از قانون زیر استفاده کرد و فلوچارت کوچکتری بکشیم. (این قانون کاربرد زیادی در برنامه‌نویسی دارد.)

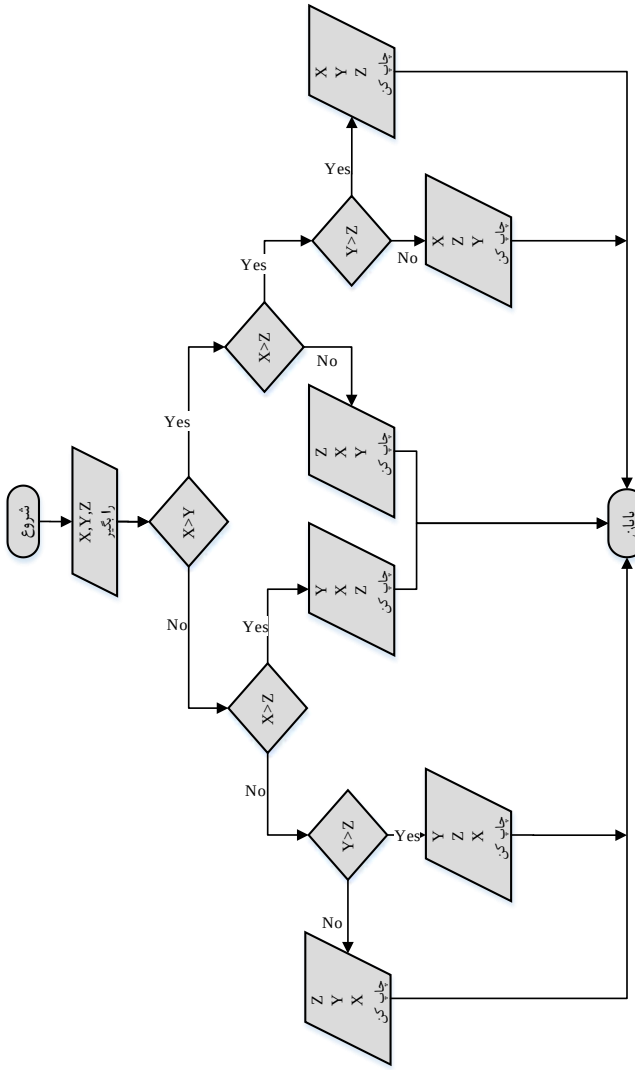
قانون: فرض می‌کنیم یک حکم درست است مگر نقیض آن ثابت شود.

حال برای این مثال فرض می‌کنیم x بزرگترین است مگر نقیض آن ثابت شود.



مثال ۱۹

فلوچارت برنامه‌ای رسم کنید که ۳ عدد را دریافت کند و به صورت نزولی مرتب و چاپ کند.


شروع
X,Y,Z را بگیر
X>Y
Yes
No
X>Z
Yes
No
Y>Z
Yes
No
X>Z
Yes
No
Y>Z
Yes
No
چاپ کن X Y Z
چاپ کن X Z Y
چاپ کن Z X Y
چاپ کن Y X Z
چاپ کن Y Z X
چاپ کن Z Y X
پایان

تمرین ۳

۱. فلوچارت برنامه‌ای رسم کنید که بزرگترین عدد بین ۴ عدد را چاپ نماید.
۲. فلوچارت برنامه‌ای رسم کنید که کوچکترین عدد بین ۳ عدد را چاپ نماید.
۳. فلوچارت برنامه‌ای رسم کنید که ۳ عدد را از ورودی دریافت و به صورت صعودی مرتب کند.
۴. فلوچارت برنامه‌ای رسم کنید که ۴ عدد را دریافت و به‌صورت نزولی مرتب و چاپ نماید.

همانطور که در سوال آخر این تمرین دیدید، با افزایش تعداد اعداد، رسم فلوچارت مربوط به مرتب‌سازی اعداد دشوارتر به نظر می‌رسد. در ادامه این کتاب و با یادگیری مفاهیم آرایه و ساختار تکرار، الگوریتمی ارائه می‌شود می‌شود که افزایش تعداد اعداد تاثیری در رسم فلوچارت ندارد.

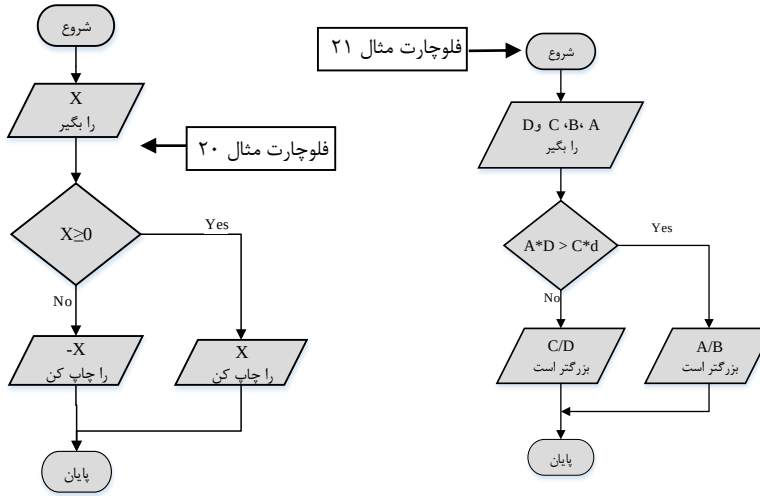
مثال ۲۰

فلوچارتی رسم کنید که قدرمطلق یک عدد را چاپ نماید.

مثال ۲۱

فلوچارت برنامه‌ای رسم کنید که دو کسر $\frac{C}{D}$ و $\frac{A}{B}$ را دریافت نماید و مشخص کند که آیا $\frac{A}{B}$ بزرگتر است یا $\frac{C}{D}$.

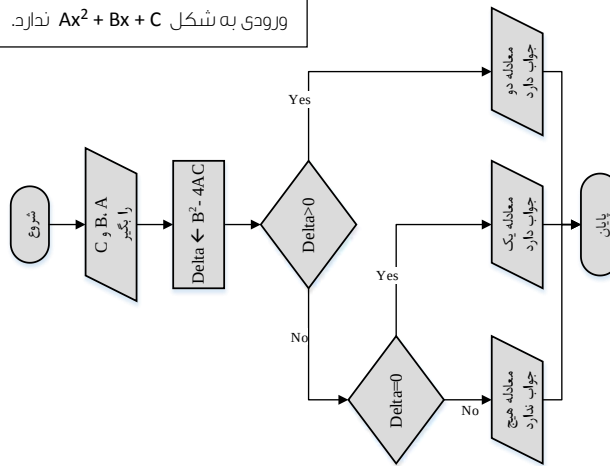
توجه داشته باشید که در مثال ۲۱، کاربر اعداد A و B و C و D را به ترتیب وارد می‌کند و کاربر نیازی به وارد کردن ورودی به شکل $\frac{A}{B}$ ندارد.



مثال ۲۲

فلوچارت برنامه‌ای رسم کنید ضرایب یک معادله درجه دوم را دریافت کند و تعداد ریشه‌های آن راچاپ کند.

توجه داشته باشید که در ورودی معادله‌ی $Ax^2 + Bx + C$ مقادیر A، B و C را وارد می‌کند و نیاز به وارد کردن ورودی به شکل $Ax^2 + Bx + C$ ندارد.

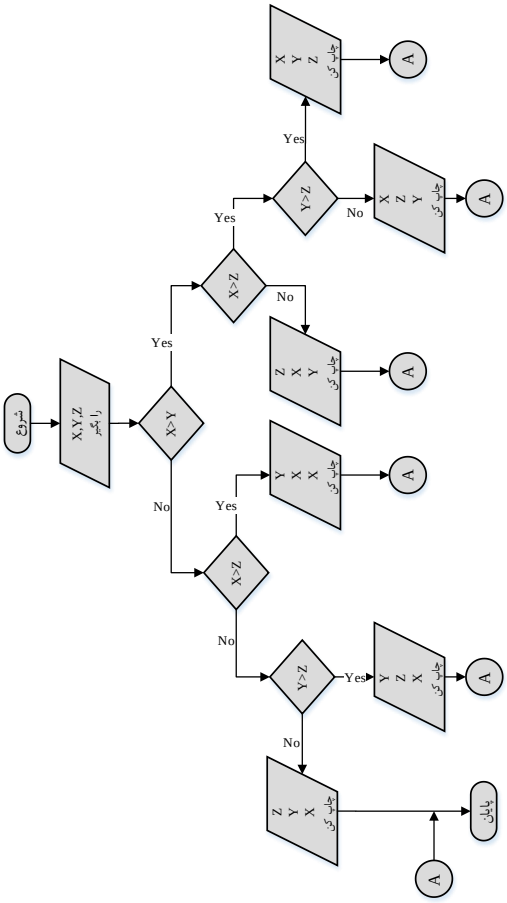


تمرین ۴

فلوچارت برنامه‌ای رسم کنید که جوابهای یک معادله درجه دو را چاپ کند.

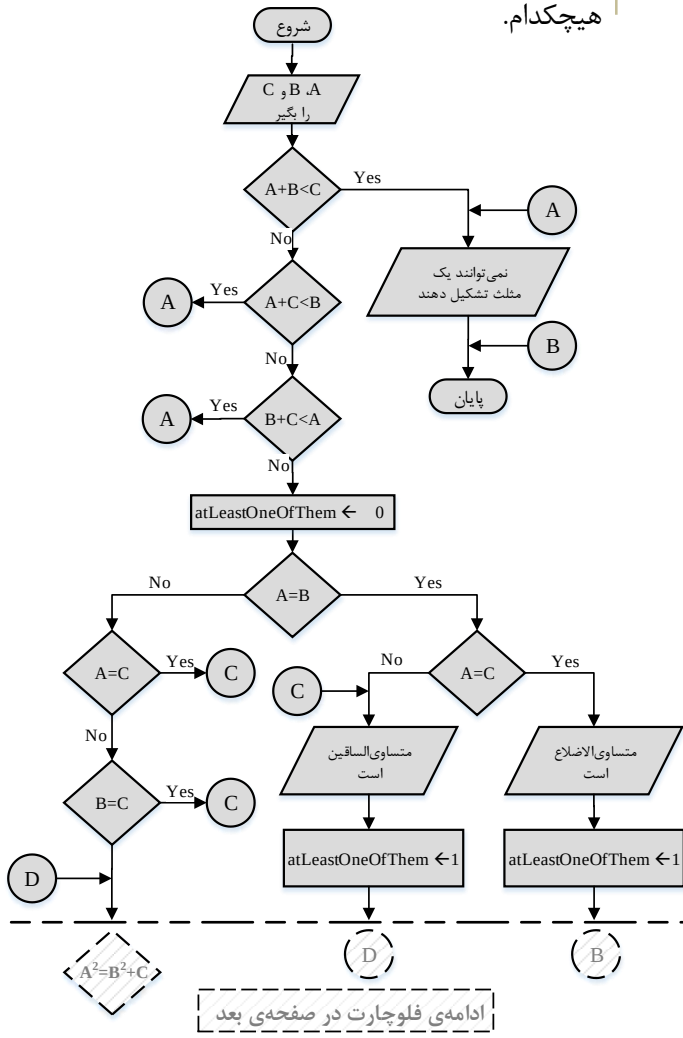
گاهی اوقات برای رفع پیچیدگی و جلوگیری از رسم خطوط زیاد، از نمادهایی مثل (A)→ در نوشتن فلوچارت استفاده می‌کنیم.

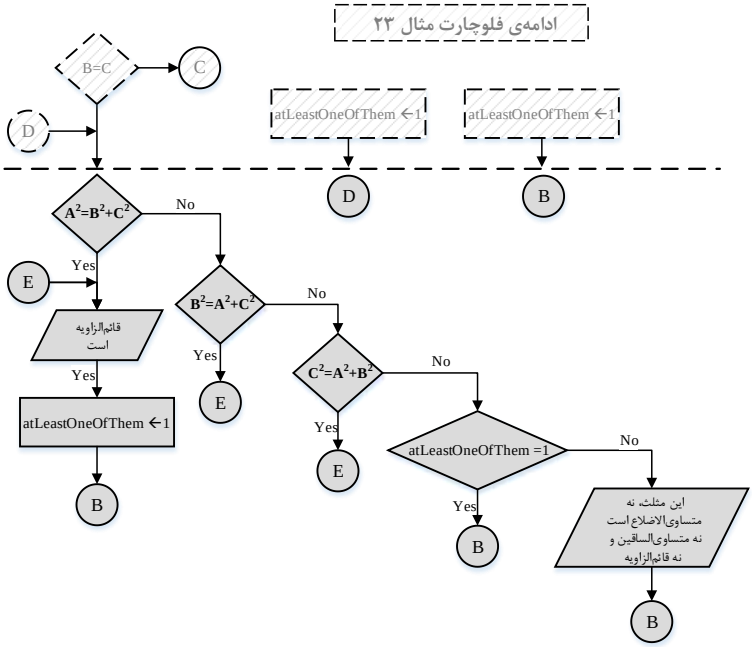
برای آشنا شدن بیشتر با این نماد، مثال ۱۹ را با استفاده از این نماد ببینید:



مثال ۲۳

فلوچارت برنامه‌ای رسم کنید که سه عدد از کاربر بگیرد و مشخص کند که آیا این سه عدد می‌توانند طول اضلاع یک مثلث باشند. اگر جواب آری بود مشخص کند که آیا آن مثلث متساوی‌الساقین است یا متساوی‌الاضلاع یا قائم‌الزاویه با هیچکدام.





تمرین ۵

۱. فلوچارت برنامه‌ای رسم کنید که وزن کشتی‌گیر را دریافت کند و مشخص کند که در هریک از وزن‌های کمتر از ۶۰، ۶۰-۷۰، ۷۰-۸۰، ۸۰-۹۰، ۹۰-۱۰۰ و بالای ۱۰۰ چه تعداد کشتی‌گیر قرار دارد.

۲. فلوچارت برنامه‌ای رسم کنید که یک عدد صحیح از ورودی دریافت و مشخص کند برای خورد کردن چنین مقدار پولی با سکه‌های یک تومانی، ۲ تومانی و ۱۰ تومانی به حداقل چه تعداد سکه نیازمندیم.

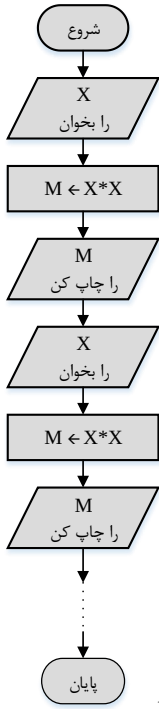
۳

فصل سوم ساختار تکرار

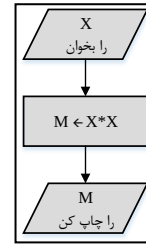
در این فصل با اصول و قواعد طراحی ساختار تکرار در فلوچارتها آشنا می‌شوید و مثالهای کاربردی و مفیدی بیان می‌شود.

ساختار تکرار

گاهی اوقات در نوشتن الگوریتم، باید عملیاتی را به صورت تکراری انجام دهیم. به طور مثال برنامه‌ای را در نظر بگیرید که ده عدد را از ورودی دریافت کند و پس از دریافت هر عدد مجذور آن عدد را چاپ کند. فلوچارت برنامه به شکل زیر می‌شود:



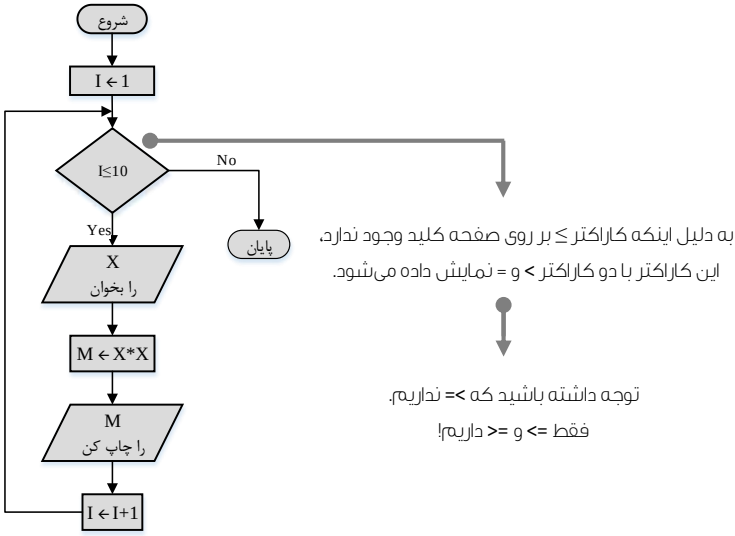
همانطور که می‌بینید سه مرحله‌ی زیر ده مرتبه باید در الگوریتم نوشته شود:



حال اگر به جای ده، ۱۰۰۰ مرتبه بخواهیم این کار را تکرار کنیم، کشیدن فلوچارت دشوار خواهد شد.

با استفاده از یک شمارنده کار را به راحتی انجام می‌دهیم.

(I مخفف Index و به معنی اندیس است.)



در نوشتن و استفاده از ساختار تکرار همیشه این نکته را مد نظر داشته باشید که یک ساختار تکرار از سه قسمت تشکیل شده است.

۱. مقداردهی اولیه به شمارنده
۲. شرط
۳. به‌روز رسانی اندیس شمارنده

به عنوان مثال در فلوچارت بالا اندیس شمارنده (که I نام دارد) با یک مقدار دهی اولیه شده است. شرط I<=۱۰ وجود دارد و در انتهای ساختار تکرار با دستور I=I+۱ شمارنده به روزرسانی می‌شود.

نکته دیگری که باید در نظر داشته باشید این است که اگر هرکدام از این سه قسمت را فراموش کنید که در فلوچارت ذکر کنید، فلوچارت شما با اشکال جدی روبرو می‌شود. هر سه مورد را با هم بررسی می‌کنیم:

۱. اگر فراموش کنید که اندیس شمارنده را مقدار دهی اولیه کنید، در به‌روز رسانی اندیس با مشکل مواجه می‌شوید. چون در به‌روز رسانی اندیس شما می‌خواهید یکی به اندیس اضافه کنید. حال اگر اندیس شما مقداری نداشته باشد چگونه یک واحد به آن اضافه شود؟
۲. قسمت اصلی ساختار تکرار شرط است. مثلا در فلوچارت قبلی شرط این جمله را بیان میکرد که تا I از ۱۰ کو چکتر است دستورات را انجام بده. حال اگر شرط وجو نداشته باشد دستورات فقط یک بار انجام می‌شود.
۳. اگر به‌روز رسانی اندیس صورت نگیرد ساختار تکرار تا بی‌نهایت ادامه پیدا خواهد کرد و الگوریتم هیچگاه متوقف نمی‌شود. (چرا؟)

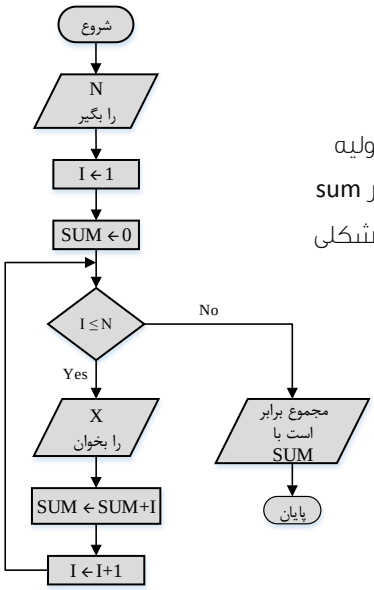
پس همیشه در استفاده از ساختار تکرار به این سه مورد توجه داشته باشید.

مثال ۲۴

فلوچارت برنامه‌ای را رسم کنید که با دریافتN ، مجموع ۱+۲+…+Nرا چاپ کند.

توجه داشته باشید که این مساله را بدون حلقه و با استفاده از فرمول $۱ + ۲ + \cdots + N = \frac{N(N+۱)}{۲}$ به راحتی می‌توان محاسبه کرد. ولی ما در این مثال برای اینکه چگونگی استفاده از ساختار تکرار را نمایش دهیم از این فرمول استفاده نمی‌کنیم.

برای رسم فلوچارت این سوال از متغیری به نام sum استفاده می‌کنیم. sum به معنی مجموع است و در این متغیر قرار است مجموع اعداد ۱ تا N ذخیره شود.



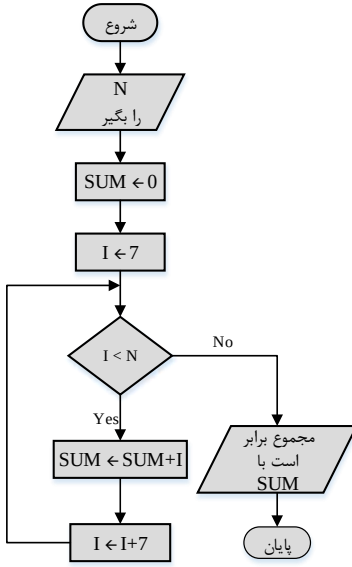
نکته مهم این الگوریتم مقداردهی اولیه کردن متغیر sum است. به نظرتان اگر sum را با صفر مقداردهی اولیه نکنیم چه مشکلی پیش می‌آید؟

این الگوریتم به چه صورت انجام می‌پذیرد؟ با استفاده از یک ساختار تکرار که در این ساختار تکرار با استفاده از متغیر I در هر دور تکرار اعداد ۱ تا N تولید می‌شوند و به متغیر sum اضافه می‌شود. توجه داشته باشید که که بین دستور sum= i و دستور sum = sum + i تفاوت زیادی وجود دارد. در دستور اول در هر مرحله i در sum ذخیره می‌شود و طبیعتا مقدار قبلی sum از بین می‌رود ولی در دستور دوم در هر مرحله i به sum اضافه می‌شود.

این مثال را با دقت زیاد بخوانید. در مثالهای بعد از ایده‌ی این مثال به وفور استفاده می‌شود!

مثال ۲۵ فلوچارت برنامه‌ای را رسم کنید که مجموع اعداد طبیعی ضریب ۷ کوچکتر از N را حساب کند. به طور مثال اگر N برابر ۴۰ باشد، حاصل ۳۵+۲۸+۲۱+۱۴+۷ را چاپ کند.

این مثال را با استفاده از ایده‌ی مثال قبل اینگونه می‌توان حل کرد که در یک ساختار تکرار همه‌ی اعداد ۱ تا N را تولید می‌کنیم و در صورتی که عدد بر ۷ بخشپذیر باشد به sum اضافه میکنیم. اما در اینجا از الگوریتم بهتری استفاده شده است:

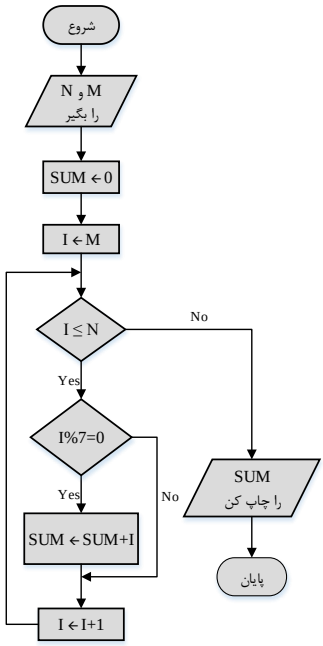


در این فلوچارت مقداردهی اولیه I با ۷ صورت گرفته است و در هر تکرار به جای اینکه یک واحد به I اضافه شود ۷ واحد اضافه شده است. بنابراین تضمین می‌کند که همه‌ی اعداد تولید شده در ساختار تکرار بر ۷ بخشپذیر است.

آیا می‌توانید بگویید که چرا این الگوریتم بهتر است؟

مثال ۲۶

فلوچارتی رسم کنید که مجموع اعداد ضرایب ۷ بین m و n را چاپ کند. (با فرض m<n)



تمرین۶

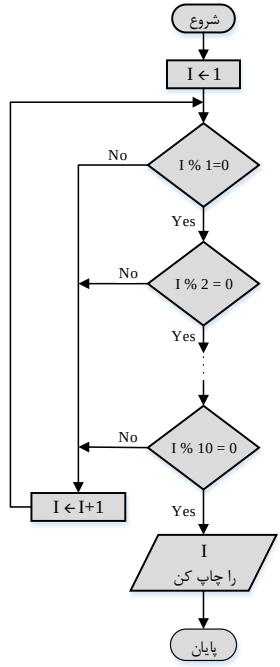
۱. فلوچارتی رسم کنید که مجموع اعداد ۳ رقمی که ضریب ۷ یا ۸ هستند را چاپ کند.

۲. فلوچارتی رسم کنید که مجموع اعداد ۳ رقمی که ضریب ۷ و ۸ هستند را چاپ کند.

مثال ۲۷

فلوچارت برنامه‌ای رسم کنید که کوچکترین عددی که بر اعداد۱ تا ۱۰ بخشپذیر است را چاپ کند.

برای حل این مساله به اندیس I نیاز داریم که از مقدار یک تا زمانیکه به جواب برسیم یک واحد یک واحد اضافه می‌شود. چنانچه I هم بر ۱ (که طبیعتا همه‌ی اعداد بر یک بخشپذیرند) هم بر ۲ و هم بر بقیه اعداد تا ۱۰ بخشپذیر باشد به این معنیست که به جواب رسیدیم و با چاپ عدد به پایان الگوریتم می‌رسیم. توجه داشته باشید الگوریتمهای بهتر و ساده‌تری برای این مثال وجود دارد که ذکر این ایده به این دلیل است که سعی شده است ساده‌ترین ایده‌ای که می‌تواند به ذهن خواننده برسد بیان شود

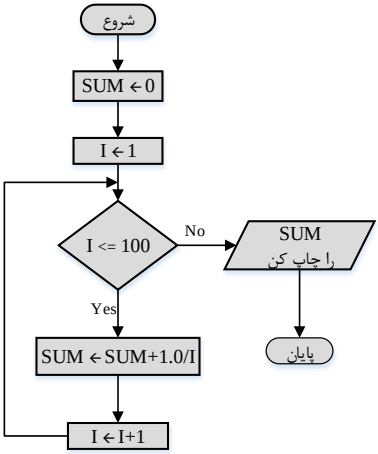


مثال ۲۸

فلوچارتی رسم کنید که مجموع ۱۰۰ جمله‌ی ابتدایی مجموع زیر را چاپ کند.

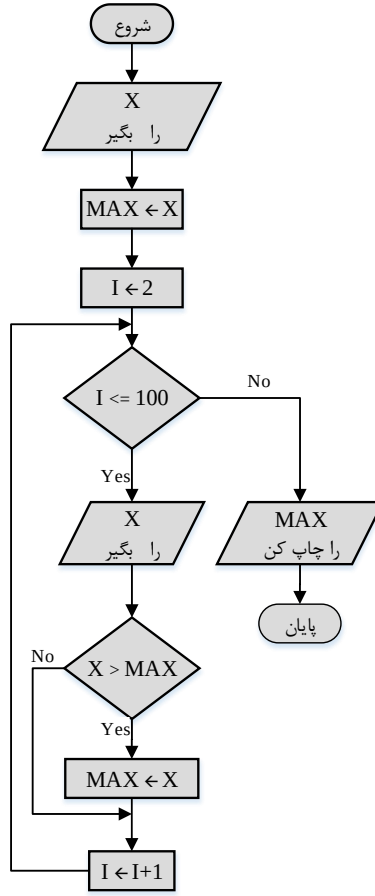
$$۱ + \frac{۱}{۲} + \frac{۱}{۳} + \frac{۱}{۴} + \cdots + \frac{۱}{n} + \frac{۱}{n+۱} + \cdots$$

رسم فلوچارت این مثال مشابه به مثال ۲۴ است. با این تفاوت که در مثال ۲۴ در هر مرحله باید I را به SUM اضافه می‌کردیم ولی در این مثال در هر مرحله باید ۱/I را به SUM اضافه کنیم. نکته قابل توجه استفاده از کسر اعشاری برای رسم فلوچارت است.



مثال ۲۹

فلوچارت برنامه‌ای رسم کنید که بزرگترین عدد را بین ۱۰۰ عددی که کاربر وارد می‌کند چاپ کند



در این مثال برای پیدا کردن بزرگترین عدد از متغیر max به صورت زیر استفاده می‌کنیم:

عدد اول را از ورودی دریافت می‌کنیم و در متغیر max ذخیره میکنیم. (به این معنی که فرض میکنیم که عدد اول بزرگترین عدد است مگر اینکه خلافش ثابت شود.)

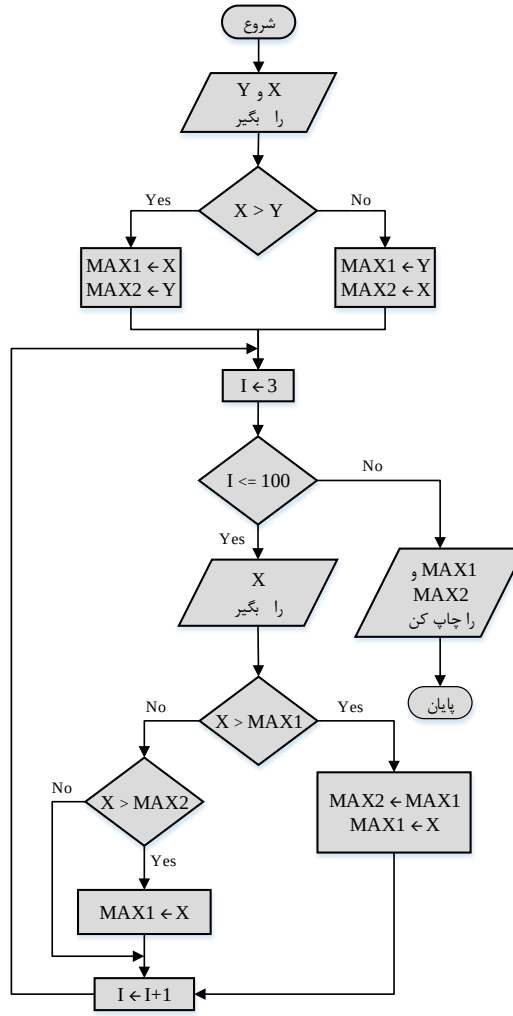
سپس در یک ساختار تکرار از I=۲ تا I=۱۰۰ حرکت میکنیم، عدد جدید دریافت میکنیم و با max مقایسه میکنیم. چنانچه x بزرگتر از max باشد، آن را در max ذخیره میکنیم.

مثال ۳۰

فلوچارت برنامه ای رسم کنید که بزرگترین و دومین بزرگترین عدد (عددی که فقط از بزرگترین عدد کوچکتر و از بقیه اعداد بزرگتر است.) را بین ۱۰۰ عددی که کاربر وارد می‌کند چاپ نماید.

از ایده مثال قبل استفاده میکنیم با این تفاوت که به دو متغیر نیاز داریم. max۱ برای ذخیره بزگترین عدد و max۲ برای ذخیره دومین بزرگترین عدد.

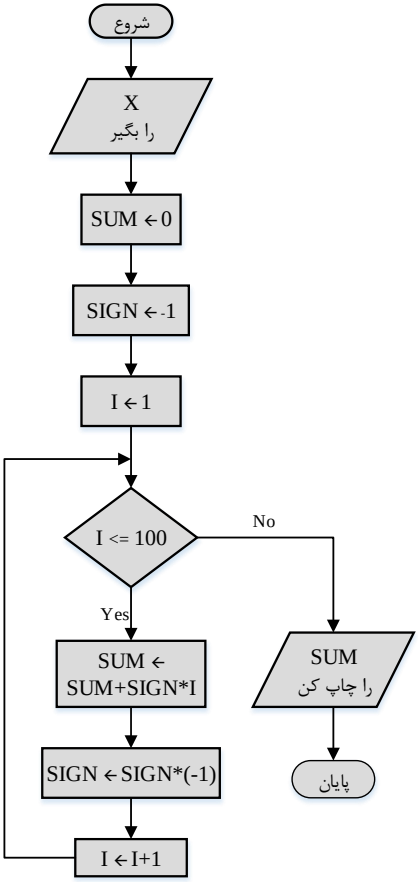
در مثال قبل فرض میکردیم که اولین عدد بزرگترین است (مگر خلافش ثابت شود) در این مثال ابتدا دو عدد از ورودی دریافت میکنیم و با مقایسه هم max۱ و max۲ رو مقداردهی اولیه میکنیم. ادامه الگوریتم را در فلوچارت زیر به دقت دنبال کنید.


شروع
X و Y
را بگیر
X > Y
Yes
No
MAX1 ← X
MAX2 ← Y
MAX1 ← Y
MAX2 ← X
I ← 3
I <= 100
No
Yes
X
را بگیر
MAX1 و
MAX2
را چاپ کن
پایان
X > MAX1
No
Yes
X > MAX2
No
Yes
MAX2 ← MAX1
MAX1 ← X
MAX1 ← X
I ← I+1

مثال ۳۱

فلوچارت برنامه‌ای رسم کنید که ۱۰۰ جمله‌ی ابتدایی مجموع زیر را محاسبه کند.

$-۱+۲-۳+۴-۵+۶-۷+...$

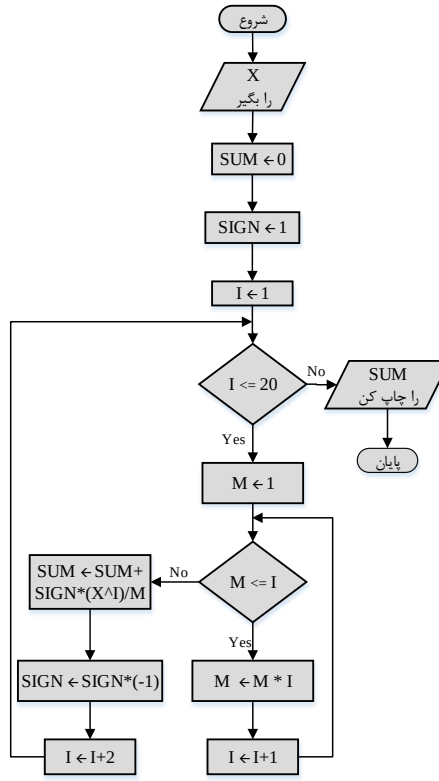


مثال ۳۲

سری محاسبه کننده‌ی sin(x) به صورت زیر می‌باشد:

$$\boldsymbol{sin(x)} = \boldsymbol{x} - \frac{\boldsymbol{x^3}}{\boldsymbol{3!}} + \frac{\boldsymbol{x^5}}{\boldsymbol{5!}} - \frac{\boldsymbol{x^7}}{\boldsymbol{7!}} + \frac{\boldsymbol{x^9}}{\boldsymbol{9!}} - \cdots$$

فلوچارتی رسم کنید که به ازای x ای که از ورودی دریافت می‌شود، مجموع ۲۰ جمله‌ی اول سری بالا را محاسبه کند.

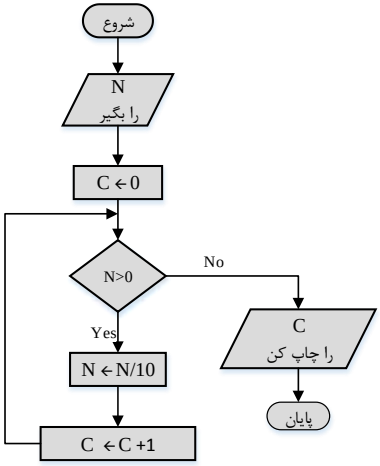


توجه داشته باشید که در ساختار تکرار فلوچارت بالا، در مرحله‌ی Iام، ابتدا I! محاسبه می‌شود در متغیر M ذخیره می‌شود و سپس $\frac{\boldsymbol{x^I}}{\boldsymbol{M}}$ به SUM اضافه می‌شود.

مثال ۳۳

فلوچارت برنامه‌ای را رسم کنید که یک عدد مثبت را دریافت و و بدون استفاده از log تعداد ارقام آن را چاپ کند.

برای حل این مساله از ایده‌ی ساده شمارش استفاده می‌کنیم. یعنی باید تک تک ارقام عدد را بشماریم. همانطور که می‌دانید، کامپیوتر مانند انسان نمی‌تواند عدد را نگاه کند و ارقام آن را بشمارد! پس برای رسیدن به این منظور، باید در الگوریتم تا جایی که عدد مورد نظر صفر نشده است آن را بر ده تقسیم کنیم و تعداد این مراحل را بشماریم (متغیر c مخفف counter و وظیفه آن شمردن این مراحل است.)

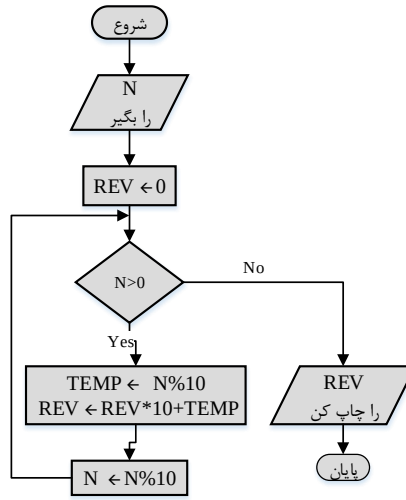


برای درک بهتر الگوریتم فرض کنید کاربر N را برابر ۱۴۲ وارد می‌کند. شما مراحل الگوریتم را قدم به قدم با این عدد دنبال کنید.

مثال ۳۴

فلوچارت برنامه‌ای رسم کنید که معکوس یک عدد مثبت را چاپ کند. (توجه کنید که معکوس ۲۴۰۰ برابر ۴۲ می‌باشد نه ۰۰۴۲)

برای رسیدن به الگوریتم این مساله بیایید با هم عدد ۳۴۱ را معکوس کنیم! شما در ذهن خود عدد معکوس را این گونه می‌سازید که ابتدا ۱ را در عدد معکوس قرار می‌دهید سپس ۴ را در سمت راست آن قرار می‌دهید و در نهایت ۳ را در سمت راست ۴ قرار می‌دهید. با این پیش زمینه فلوچارت زیر را نگاه کنید

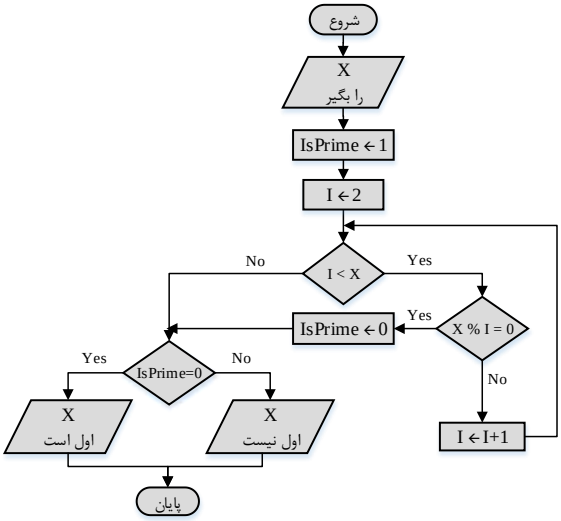


در فلوچارت بالا عددی که از ورودی خوانده می‌شود N و عددی که به عنوان عدد معکوس ساخته می‌شود REV نام دارد. در ساختار تکرار، در هر مرحله سمت راست‌ترین رقم با دستور N٪۱۰ در متغیر TEMP ذخیره می‌شود و با دستور REV = REV*۱۰+TEMP رقم به سمت راست REV اضافه می‌شود. برای درک بهتر، فلوچارت را برای N=۳۴۱ قدم‌به‌قدم دنبال کنید.

مثال ۳۵

فلوچارتی رسم کنید که با دریافت x از ورودی، بررسی کند که آیا x یک عدد اول است یا خیر.

همانطور که می‌دانید عدد اول عددی طبیعی بزرگ‌تر از ۱ است که بر هیچ عددی به جز خود و ۱ بخش‌پذیر نباشد. پس در تشخیص اول بودن عدد N به ساختار تکراری نیازی داریم که همه ی اعداد بزرگتر از یک و کوچکتر از N را تولید کند و چک کند که آیا N بر این اعداد بخش‌پذیر است یا خیر. اگر حداقل یک عدد یافت شد که N بر آن بخش‌پذیر است آنگاه نتیجه می‌گیریم که N اول نیست و درصورتی که هیچ عددی یافت نشد نتیجه می‌گیریم N اول است.



متغیر IsPrime به صورت پیش فرض ۱ است (و به این معنی است که عدد اول است.) چنانچه در ساختار تکرار عدد Iای پیدا شد که X%I برابر صفر شود متغیر IsPrime را برابر صفر قرار می‌دهد. (و به این معنی که عدد اول نیست)

بد نیست بدانیم...

۱

کاربرد

عدد اول

یکی از کاربردهای عدد اول رمزنگاری است. برای اینکه با ایده استفاده از اعداد اول در رمزنگاری آشنا شوید بیایید با هم به چند صد سال قبل برگردیم!

فرض کنید شما میخواهید قطعه ای طلا از شهر A برای دوستتان به شهر B ارسال کنید. برای اینکه طلاها از راهزنان احتمالی در امان باشند گاوصندوقی بسیار محکم در اختیار دارید که باز شدن آن با هیچ روشی به جز داشتن کلید امکان پذیر نیست. شما چطور میخواهید قطعه طلا را به آن شهر بفرستید؟ پاسخ های احتمالیتان را با هم مرور میکنیم:

-اگر طلا را در گاوصندوق قرار دهید و بخواهید گاوصندوق را به شهر B بفرستید مجبورید کلید را هم به همراه آن بفرستید. همراه بودن کلید با صندوق همان و به تاراج رفتن طلاهایتان نیز همان!

-اگر بخواهید کلید را در بار مخفی کنید و امید داشته باشید که راهزن کلید را پیدا نمی‌کند هم کار اشتباهی مرتکب شده اید. راهزنان مو به مو بار شما را خواهند گشت.

-اگر بخواهید صندوق را از مسیر ۱ و کلید را از مسیر ۲ به مقصد B بفرستید در کمال شرمندگی باید بگوییم باز هم اشتباه کرده اید. در همه مسیرها یکی از همکاران راهزنان منتظر شما هستند تا کلید را از شما بگیرند!

راه حل این مساله چندان هم سخت نیست. کافی است دوست شما از شهر B یک صندوق با قفل باز برای شما بفرستد (و کلیدِ قفل دست دوست شما باقی بماند) شما طلاها را در صندوق قرار دهید و قفل را ببندید و به مقصد بفرستید. همانطور که میبینید دیگر نیازی نیست که کلید را با صندوق ارسال کنید و کلید در اختیار گیرنده قرار دارد.

همه مطالبی که عنوان شد اساس و پایه کلید عمومی و کلید خصوصی در رمزنگاری است. این بار بیایید با هم وارد سایت جیمیل شویم و ببینیم برای ورود با نام کاربری و رمز عبورمان چه اتفاقی می افتد:

بد نیست بدانیم...

وقتی شما وارد سایت gmail می شوید، مرورگر شما از سرور جیمیل درخواست یک کلید عمومی می‌کند (همان گاوصندوق در باز که در داستان بالا گفته شد). سرور به مرورگر کلید عمومی را ارسال می‌کند و مرورگر شما با استفاده از کلید عمومی داده های شما که همان نام کاربری و رمزعبور است را رمزنگاری می‌کند و به سرور ارسال میکند. حال اگر در میان راه هکرها (یا همان راهزنها) به این داده ها دسترسی پیدا کنند اصلا جای نگرانی نیست، زیرا تا زمانی که کلید خصوصی در اختیار نداشته باشند نمی‌توانند پیام را رمزگشایی کنند. (کلید خصوصی در اختیار چه کسی است؟ سرور جیمیل! پس فقط سرور جیمیل می تواند داده های شما را رمزگشایی کند)

اگر تا اینجای مطلب را با دقت خوانده باشید حتما این سوال برایتان پیش می‌آید که عدد اول کجای داستان قرار دارد و کاربرد این اعداد در رمزنگاری چیست؟ در پاسخ باید گفت که مبنای ساخت کلیدهای عمومی و خصوصی اعداد اول هستند. نکته‌ی جالبی که می‌توان در مورد اعداد اول به آن اشاره کرد، امکان تقسیم تمام اعداد، به اعداد اول است. یعنی می‌توان اعداد اول را به گونه‌ای در هم ضرب کرد، تا حاصلضرب، عددی باشد که در نظر داریم. برای مثال عدد ۱۳۸ را می‌توان از ضرب سه عدد ۲ در ۳ در ۲۳ که همگی عدد اول هستند به دست آورد. این موضوع در مورد اعداد بزرگ‌تر نیز صادق است، برای مثال می‌توان عدد ۳۴۷۴۲۳۹۳۰ را با ضرب کردن اعداد ۲ در ۵ در ۷ در ۱۹ در ۹۷ در ۲۶۹۳ به دست آورد. به این عمل قانون فاکتورگیری عدد اول گفته می‌شود. علاوه بر این هر عدد طبیعی را فقط می‌توان به یک مجموعه عدد اول تجزیه کرد. نکته اساسی اینجاست که اگر از بهترین الگوریتم موجود به منظور تقسیم یک عدد ۲۰۰ رقمی یا ۵۰۰ رقمی به فاکتورهای اول آن استفاده کنیم، بهترین سوپرکامپیوتر موجود نیز به زمان بسیار بسیار زیادی برای اتمام کار خود نیاز خواهد داشت، شاید معادل عمر کره زمین! پس بیایید با هم یک بار دیگر مرور کنیم:

کلید عمومی و خصوصی با استفاده اعداد اول تولید می‌شوند. پیامها با استفاده از کلید عمومی گیرنده کدگذاری می‌شود و با استفاده از کلید خصوصی که فقط گیرنده دارد رمزگشایی می‌شود. اگر در بین راه کلید عمومی توسط هکرها به اعداد اول شکسته شوند با استفاده از اعداد اول تولیدی می‌توانند به کلید خصوصی دسترسی پیدا کنند و پیام را رمزگشایی کنند. اما همانطور که پیش تر گفته شد شکسته شدن اعداد بزرگ به فاکتورهای اول بسیار زمانبر است و عملا امکان پذیر نیست.

نبود راهی بهینه برای یافتن فاکتورهای اول یک عدد بزرگ، بنیان رمزنگاری در کامپیوترها است

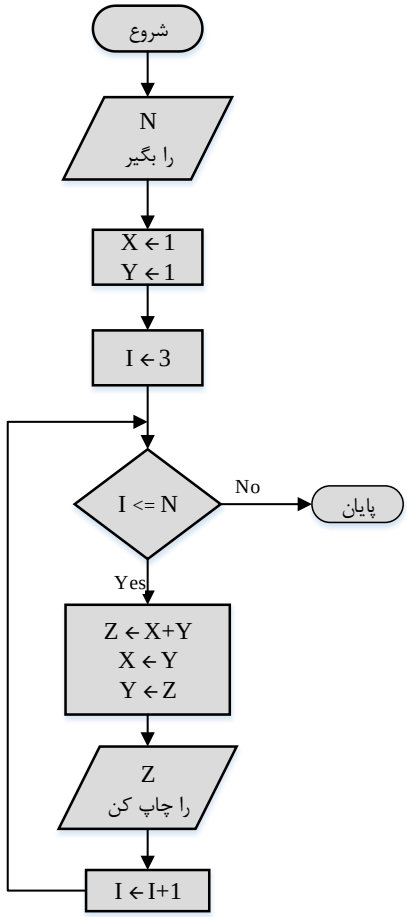
چنانچه به رمزنگاری علاقه‌مند شدید در آدرس jahangirics.ir
کاربرد اعداد اول را جستجو کنید

مثال ۳۶

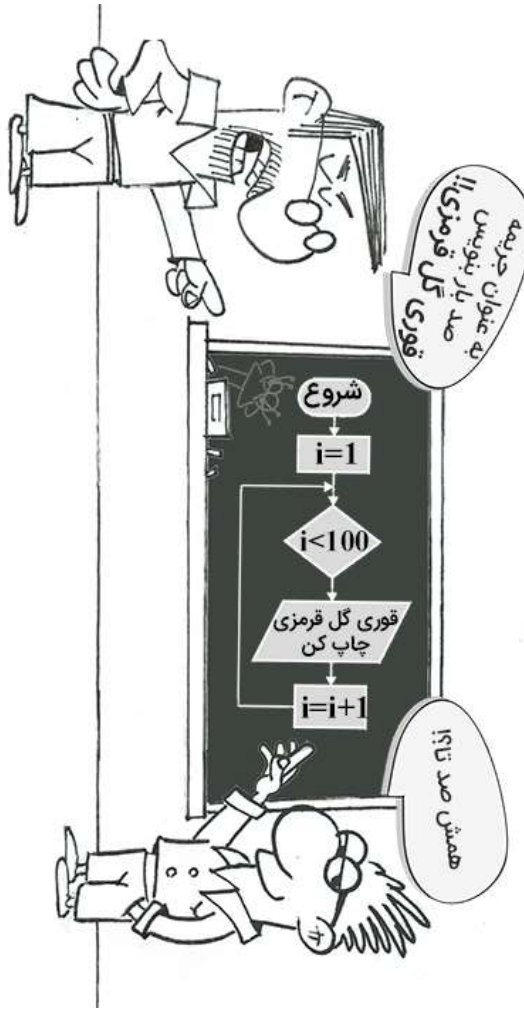
دنباله‌ی اعداد فیبوناتچی به صورت زیر است:
۱، ۱ ، ۲، ۳، ۵، ۸ ،۱۳، ...
که در آن به جز دو جمله‌ی اول، مابقی حاصل جمع دو جمله‌ی قبل خود هستند. فلوچارتی رسم کنید که با دریافت N، N جمله‌ی اول دنباله فیبوناتچی را چاپ کند.

برای این کار از سه متغیر X و Y و Z استفاده می‌کنیم که X و Y در ابتدا با عدد یک مقداردهی می‌شوند. همانطور که در جدول زیر می‌بینید همیشه Z از جمع دو عدد X و Y بدست می‌آید و X و Y از Y و Z مرحله قبل! به عنوان مثال در گام اول X و Y برابر ۱ و Z برابر جمع این دو عدد است. در گام دوم، X برابر مقدار Y در گام اول (یعنی ۱) و Y برابر مقدار Z در گام اول (یعنی ۲) و Z برابر مجموع X و Y (یعنی ۳) است.

گام اول	۱,	۱,	**۲**			
	X	Y	Z=**X+Y**			
گام دوم	۱,	۱,	۲,	**۳**		
		X	Y	Z=**X+Y**		
گام سوم	۱,	۱,	۲,	۳,	**۵**	
			X	Y	Z=**X+Y**	



در فلوچارت بالا I شمارنده‌ای است که قرار است اعداد دنباله را بشمارد و چون X و Y با ۱ مقدار دهی اولیه شده‌اند پس اولین عددی که در ساختار تکرار ساخته می‌شود در واقع سومین عدد دنباله ی فیبوناتچی است و به همین خاطر I با عدد ۳ مقداردهی شده است.


به عنوان جریمه صد بار بنویس قوری گل قرمزی!!!
شروع
i=1
i<100
قوری گل قرمزی چاپ کن
i=i+1
همش صد تا؟!

۴

فصل چهارم زیرالگوریتم‌ها

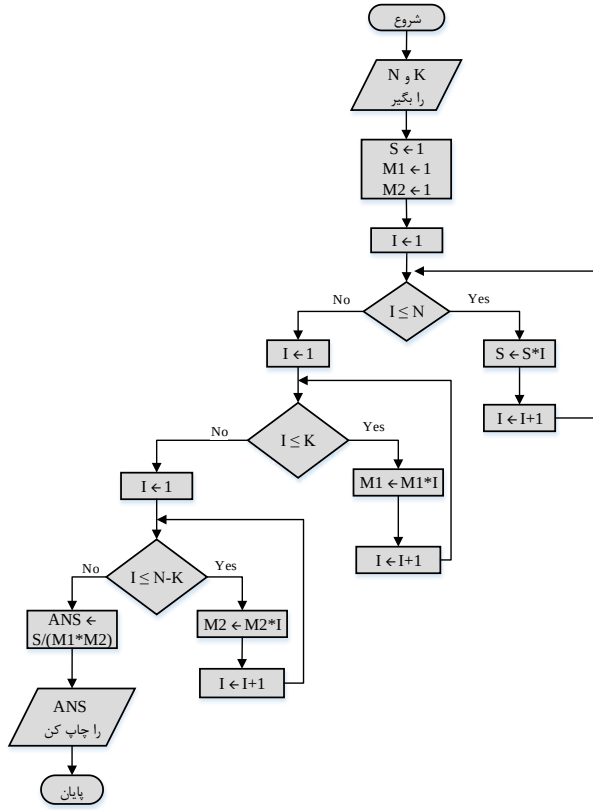
ممکن است بدون زیرالگوریتم بتوانید هر مساله‌ای را حل کنید، اما لازمه‌ی نوشتن یک الگوریتم قابل فهم و زیبا ، استفاده درست از زیرالگوریتم است.

زیرالگوریتم‌ها (یا معادل توابع در برنامه‌نویسی) از پرکاربردترین مفاهیم الگوریتمها و برنامه‌نویسی به شمار می‌رود. هدف اصلی زیرالگوریتمها کاهش اشتباه و کاهش نوشتن تعداد خطوط الگوریتم است. در بیشتر مواقع استفاده از زیرالگوریتم، خوانایی الگوریتم را به شدت افزایش می‌دهد. ابتدا در قالب یک مثال بیان می‌کنیم که چرا به زیر الگوریتم نیاز داریم و در ادامه نحوه استفاده و قالب نوشتن زیرالگوریتم را توضیح می‌دهیم.

مثال ۳۷ فلوچارت برنامه‌ای رسم کنید که با دریافت k و n مقدار $\binom{n}{k}$ را چاپ کند.

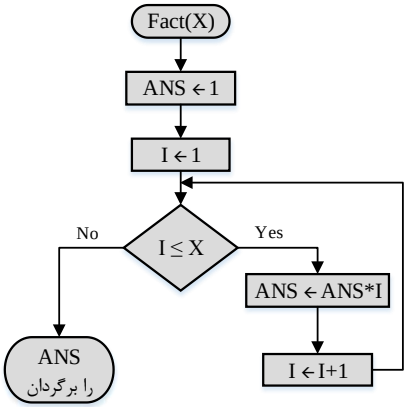
یادآوری: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$

همانطور که در فلوچارت صفحه بعد می‌بینید ، مقادیر !n ، !k و !(n-k) هر سه باید محاسبه شوند که مقدار ورودی یکبار n، یکبار k و یکبار n-k می‌باشد و نوشتن الگوریتم فاکتوریل برای هر سه مقدار کاری غیر منطقی به نظر می‌رسد!



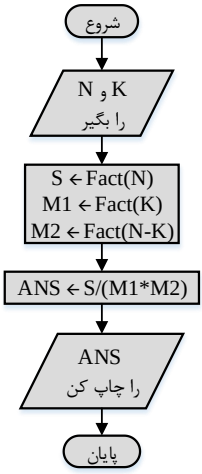
به جای اینکه الگوریتم فاکتوریل را سه بار در فلوچارت ذکر کنیم می‌توانیم الگوریتم فاکتوریل را در قالب یک زیر الگوریتم بیان کنیم و هر موقع به محاسبه‌ی فاکتوریل نیاز داشتیم زیر الگوریتم فاکتوریل را **صدا** بزنیم.

در زیرالگوریتم، به جای (شروع) اسم زیرالگوریتم و در داخل پرانتز ورودی آن را مشخص می‌کنیم. همچنین به جای (پایان) مقدار محاسبه شده را برمی‌گردانیم. زیرالگوریتم فاکتوریل را ببینید:



در زیرالگوریتم بالا، به جای نوشتن شروع نام تابع به همراه پارامتر ورودی (یعنی Fact(X)) و به جای پایان مقدار بازگشتی (یعنی ANS) را برمی‌گرداند.

بعد از نوشتن زیر الگوریتم به راحتی می‌توان با نام زیرالگوریتم دستورات زیر الگوریتم را فراخوانی کرد:

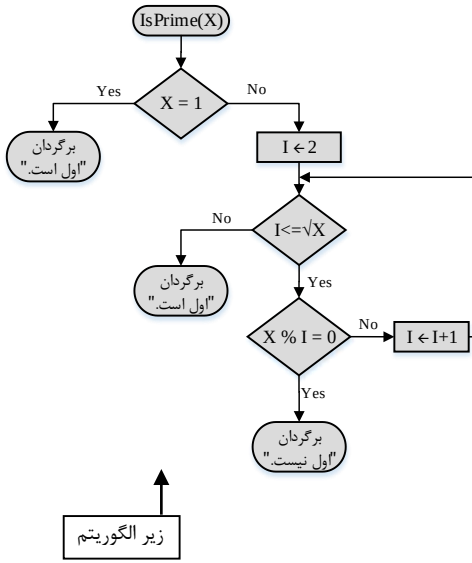


فلوچارت چگونه کار می‌کند؟

از ورودی N و K را دریافت می‌کند. دستور S=Fact(N) زیر الگوریتم Fact را صدا می‌زند و به عنوان ورودی به زیر الگوریتم N را ارسال می‌کند. زیر الگوریتم !N را محاسبه می‌کند و به فلوچارت اصلی برمی‌گرداند (یعنی همانجایی که زیر الگوریتم را صدا زده‌ایم) و در نتیجه مقدار !N در S ذخیره می‌شود. همین روال برای دو دستور بعدی نیز اجرا می‌شود.

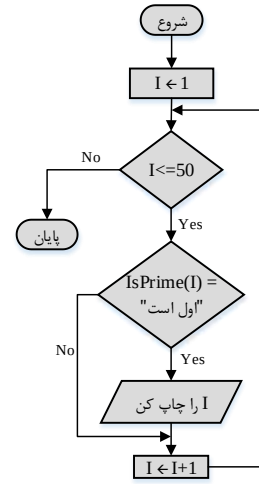
مثال ۳۸ با استفاده از زیرالگوریتمی برای تشخیص یک عدد اول، اعداد اول بین ۱ تا ۵۰ را چاپ کنید.

ابتدا زیر الگوریتم IsPrime(X) را می‌نویسیم. این زیرالگوریتم X را به عنوان ورودی دریافت می‌کند و درصورتی که X اول باشد **اول است** و در غیراینصورت **اول نیست** را برمی‌گرداند.



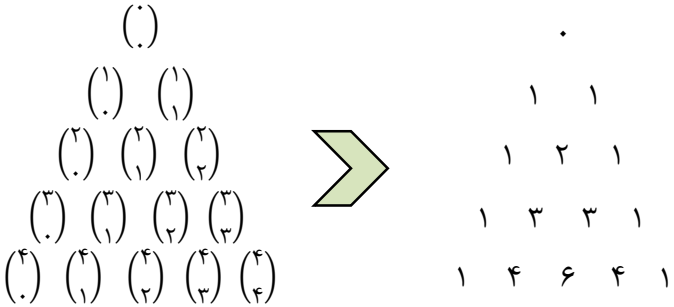
زیر الگوریتم

فلوچارت اصلی



مثال ۳۹

مثلث خیام به شکل زیر می‌باشد:



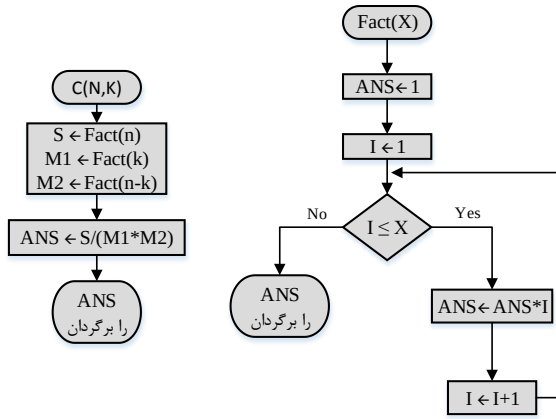
فلوچارتی رسم کنید که اعداد این مثلث را تا سطر N ام (که N را از کاربر دریافت می‌کند) چاپ کند.

برای حل این مساله به دو زیرالگوریتم احتیاج داریم:

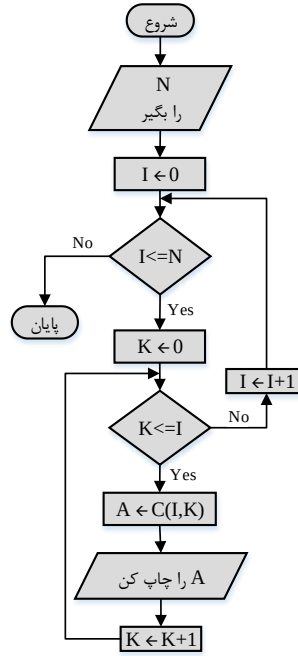
۱. زیرالگوریتم محاسبه‌ی فاکتوریل.

۲. زیر الگوریتم محاسبه‌ی $\binom{N}{K}$.

زیرالگوریتم محاسبه‌ی فاکتوریل را با نام Fact(X) معرفی می‌کنیم و زیرالگوریتم ترکیب K از N را با C(N,K) معرفی میکنیم.



فلوچارت اصلی برنامه به صورت زیر خواهد بود:



در این فلوچارت از دو ساختار تکرار تو در تو استفاده شده است. در ساختار تکرار اول، اندیس I از 0 تا N حرکت می‌کند. برای درک بهتر ادامه‌ی مسیر بیایید قدم به قدم حرکت کنیم:

بعد از اینکه I با صفر مقداردهی شد، با چک کردن شرط I<=N وارد ساختار تکرار می‌شود. سپس اندیس K با صفر مقدار دهی می‌شود و آماده‌ی ورود به ساختار تکرار دوم می‌شود. قبل از اینکه وارد ساختار دوم شویم تاکید می‌کنیم که مقدار I برابر صفر است. شرط K<=I به این معناست که ساختار تکرار دوم از K=0 تا K=I تکرار خواهد شد. (بنابراین چون I برابر صفر است در این مرحله ساختار تکرار دوم فقط یک بار اجرا می‌شود.) و در هر مرحله C(I,K) را محاسبه و چاپ کند. (پس فقط C(0, 0) اجرا می‌شود.) بعد از اینکه ساختار تکرار دوم به پایان رسید یک واحد به I اضافه می‌شود و به سراغ شرط ساختار اول میرود. چنانچه شرط برقرار باشد وارد ساختار تکرار می‌شود و این بار به ازای I=0 وارد ساختار تکرار دوم می‌شود و ساختار تکرار دوم به ازای K=0 و K=۱ اجرا می‌شود. پس C(۰, ۰) و C(۱, ۰) محاسبه و چاپ می‌شوند. این روند ادامه خواهد داشت تا I به مقدار N برسد و دستوراتش را اجرا کند.

پس در حالت کلی در صورتی که دو ساختار تکرار تو در تو مانند این مثال داشته باشید، باید به این نکته توجه کنید که اندیس I (که مربوط به ساختار تکرار اول است.) در ساختار تکرار دوم مقداری ثابت دارد و مقدارش بعد از اتمام ساختار تکرار دوم تغییر می‌کند.

حتما این فلوچارت را با دقت مطالعه کنید و اگر در فهم این الگوریتم با مشکل مواجه شدید به ازای N=۳ فلوچارت را قدم به قدم دنبال کنید.

مثال ۴۰

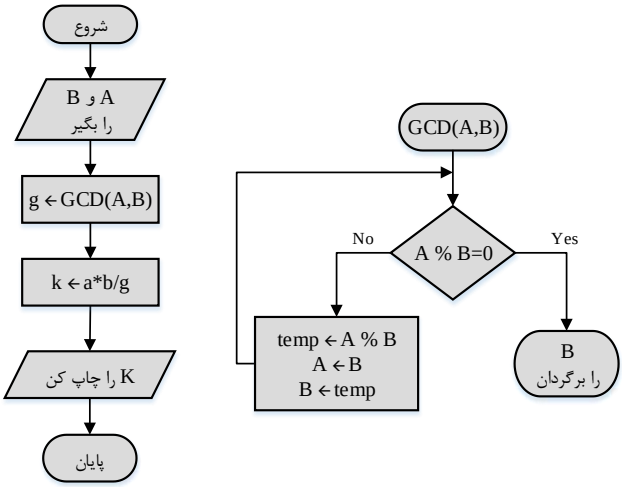
فلوچارت برنامه‌ای رسم کنید که با استفاده از زیرالگوریتمی که ب.م.م دو عدد را محاسبه می‌کند، ک.م.م دو عدد را چاپ کند.

برای درک بهتر از نحوه‌ی نوشتن زیرالگوریتم ب.م.م.، ب.م.م. دو عدد ۲۸ و ۱۶ را به روش نردبانی بدست می‌آوریم:

	۱	۱	۳	
۲۸	۱۶	۱۲	۴	۰
۱۶	۱۲	۱۲		

ب.م.م دو عدد ۲۸ و ۱۶

در گام اول باقیمانده ۲۸ بر ۱۶ (که ۱۲ است) محاسبه می‌شود. در گام دوم باقیمانده ۱۶ بر ۱۲ (که ۴ است) محاسبه می‌شود. و در گام آخر که باقیمانده ۱۲ بر ۴ صفر می‌شود به منزله‌ی اتمام الگوریتم است و ۴ به عنوان ب.م.م ۲۸ و ۱۶ محاسبه می‌شود. پس با دقت به روند اشاره شده متوجه خواهید شد که در هر مرحله به ۳ متغیر نیاز دارید. زیرالگوریتم GCD(A,B) ب.م.م دو عدد A و B را برمیگرداند:



مثال ۴۱

یک عدد را متقارن گویند، اگر از دو طرف به یک شکل خوانده شود. به طور مثال ۹۰۹ و ۸۱۱۸ و ۹ همگی متقارن هستند، ولی ۶۴۴ متقارن نیست. فلوچارتی رسم کنید که بزرگترین عدد متقارنی که حاصل‌ضرب دو عدد دو رقمی می‌باشد را چاپ نماید.

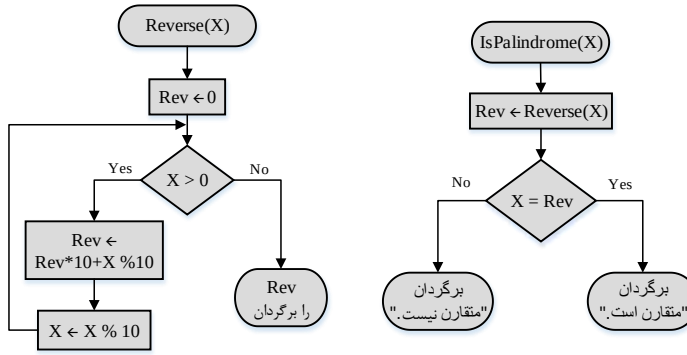
در مواجه با چنین سوالاتی ابتدا آرامش خود را حفظ کنید!
در ظاهر ممکن است با سوال سختی روبرو باشید، اما همیشه به یاد داشته باشید که هنر یک برنامه‌نویس می‌تواند ساده کردن مساله اصلی به چند مساله ساده‌تر باشد.

برای حل این مساله باید همه‌ی اعدادی که حاصلضرب دو عدد دو رقمی هستند را پیمایش کنید و از بین آنها اعداد متقارن را پیدا کنید و از بین اعداد متقارن بزرگترین عدد را پیدا کنید. پس یکی از زیرالگوریتمهای مهمی که در

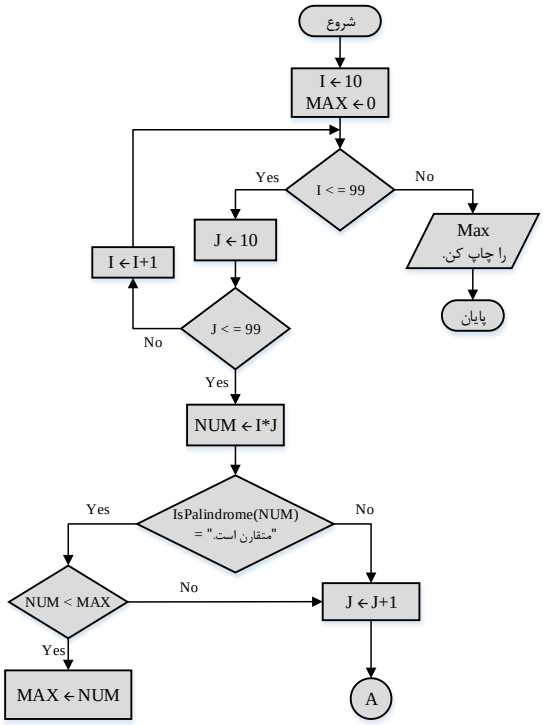
این مساله به آن نیاز دارید زیرالگوریتم تشخیص عدد متقارن است. چه عددی متقارن است؟ عددی که با معکوس خود برابر باشد. پس به دو زیر الگوریتم نیاز داریم:

۱- زیرالگوریتم معکوس کردن عدد (Reverse(X))

۲-زیرالگوریتم تشخیص عدد متقارن (IsPalindrome(X))



در ادامه باید همه اعدادی که حاصلضرب دو عدد دو رقمی هستند تولید کنیم. چگونه؟ با استفاده از دو ساختار تکرار تو در تو و دو اندیس I و J . به این صورت که در ساختار تکرار اول اندیس I از ۱۰ تا ۹۹ حرکت می‌کند و در دلِ ساختار تکرارِ اول، ساختار تکرار دوم با اندیس J از ۱۰ تا ۹۹ حر کت می‌کند (پس به ازای هر مقدار I، متغیر J از ۱۰ تا ۹۹ حرکت می‌کند.) بنابراین مقدار متغیر NUM برابر همه مقادیری خواهد شد که حاصلضرب دو عدد دو رقمی است. بعد از اینکه NUM تولید شد با استفاده از زیرالگوریتم IsPalindrome در صورتی که عدد متقارن بود و از max بزرگتر بود در max ذخیره کنیم.



مثال ۴۲

اعداد مثلثی به اعدادی گویند که از مجموع ۱ تا n بدست می‌آید. این اعداد به شکل زیر هستند:

۱: **۱**

۱+۲: **۳**

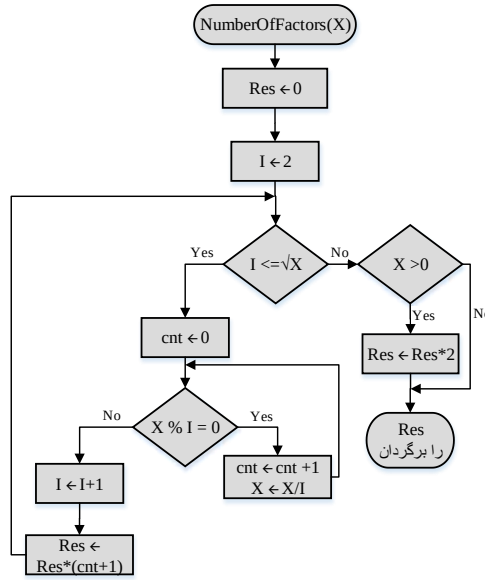
۱+۲+۳: **۶**

۱+۲+۳+۴: **۱۰**

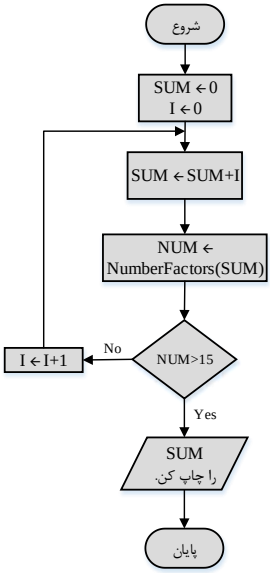
۱+۲+۳+۴+۵: **۱۵**

فلوچارتی رسم کنید که اولین عدد مثلثی که تعداد مقسوم‌علیه‌های آن بیشتر از ۱۵ باشد را بیابد.

در زیرالگوریتم NumberOfFactor(X) تعداد مقسوم‌علیه‌های عدد X برگردانده می‌شود



در فلوچارت اصلی برنامه، باید اعداد مثلثی تولید کنیم و چنانچه تعداد مقسوم علیه‌های عدد مثلثی (که با استفاده از زیر الگوریتم NumberOFFactor محاسبه می‌شود.) بیشتر از ۱۵ باشد آن عدد را چاپ کند.



تمرین ۷

۱. فلوچارتی رسم کنید که ۱۳۹۶مین عدد اول را چاپ کند. (راهنمایی: یک زیرالگوریتم برای تعیین عدد اول رسم کنید.)

۲. فلوچارتی رسم کنید که مجموع ارقام $۲^{۱۰۰۰}$ راچاپ کند. (راهنمایی: یک زیر الگوریتم برای محاسبه‌ی مجموع ارقام یک عدد رسم کنید. همچنین با رسم یک زیرالگوریتم برای به توان رساندن یک عدد فلوچارت خواسته شده را بهتر می‌توان رسم کرد.)

۳. فلوچارتی رسم کنید که مجموع ارقام ۱۰۰! را چاپ کند. (راهنمایی: یک زیرالگوریتم برای فاکتوریل و یک زیرالگوریتم هم برای محاسبه‌ی مجموع ارقام یک عدد رسم کنید.)

۴. اگر d(a) برابر مجموع تمام مقسوم‌علیه‌های a به غیر از خود a باشد و داشته باشیم d(a)=d(b) و a≠b، آنگاه a و b را یک جفت مهربان گویند. فلوچارتی رسم کنید که تعداد جفت‌های مهربان کمتر از ۱۰۰۰ را چاپ کند. (راهنمایی: یک زیرالگوریتم برای محاسبه‌ی d(a) رسم کنید.)

۵. یک عدد، خوب نامیده می‌شود اگر مجموع تمام مقسوم‌علیه‌های (غیرخودش) آن برابر خود عدد باشد، یعنی d(a)=a. فلوچارتی رسم کنید که تعداد اعداد خوبِ کوچکتر از ۱۰۰ را چاپ کند. (راهنمایی: یک زیرالگوریتم برای محاسبه‌ی مجموع مقسوم‌علیه‌های یک عدد رسم کنید.)

۶. برای عدد ۱۶۳۴ داریم: ۱۶۳۴=۱۴+۶۴+۳۴+۴۴

فلوچارتی رسم کنید که مجموع تمام اعدادی که دارای این خاصیت هستند (یعنی آن عدد که برابر مجموع توان ۴ام ارقام خودش است) را چاپ کند.

۷. یک عدد جالب‌انگیز است،اگر مجموع فاکتوریل ارقام آن با خود عدد برابر باشد، به طور مثال: ۱۴۵ = ۱! + ۴! + ۵!

فلوچارتی رسم کنید که مجموع تمام اعداد جالب‌انگیز را چاپ کند.

۸. یک عدد جالب نامیده می‌شود، اگر حاصلضرب دو عدد دیگر باشد و در هر سه‌ی این اعداد هر رقم فقط یک بار ظاهر شود.

به طور مثال: ۷۲۵۴ = ۱۸۶ × ۳۹

همانطور که می‌بینید ۷۲۵۴ جالب است، زیرا ارقام ۱ تا ۹ در رابطه‌ی بالا هر کدام یکبار (فقط یک بار) ظاهر شده‌اند. فلوچارتی رسم کنید که مشخص کند که آیا یک عدد جالب است یا خیر.

فصل پنجم آرایه‌ها

شما بدون آرایه‌ها قادر به نوشتن برنامه‌های بزرگ نیستید. در این فصل با آرایه‌های یک بعدی و دو بعدی آشنا می‌شوید.

تا کنون با فلوچارت‌های مهم و نحوه‌ی رسم آن آشنا شده‌اید. در این قسمت به بررسی یک نوع پر کاربرد و مهم برای متغیرها خواهیم پرداخت. سعی می‌کنیم به سوالاتی مانند "چطور می‌توان یک میلیون متغیر تعریف کرد؟" پاسخ دهیم.

ممکن است این سوال پیش بیاید که چرا ممکن است به یک میلیون متغیر نیاز پیدا کنیم؟ در پاسخ باید گفت که شما ممکن است در مساله‌ای نیاز به تحلیل نمرات دانشجویان کشور بر اساس چند پارامتر خاص داشته باشید و طبیعتا نیاز به ذخیره همه ی نمرات آنها داشته باشید.

برای تعریف یک میلیون متغیر باید از یک سری اصول استفاده کرد و هم‌چنین این متغیرها را باید طوری طراحی کنیم که کار با آنها خسته‌کننده نباشد. برای این کار از ایده‌ای که معمولا در ریاضیات به کار می‌برند استفاده می‌کنیم. در ریاضیات، زمانی که متغیرها زیاد باشد از اندیس استفاده می‌کنند. مثلا x_1 و x_2 و ... و x_n؛ یعنی n متغیر. در برنامه نویسی هم دقیقا از همین ایده استفاده می‌شود. فقط چون نمی‌توانیم اندیس را مانند دست‌خط زیر متغیر بنویسیم و بر روی صفحه کلید، کلیدی برای این کار تعریف نشده است از [ ] استفاده می‌کنیم. یعنی x[۱] و x[۲] و ...و x[n].

در واقع آرایه ساختاری است که می‌تواند چندین مقدار در خود ذخیره کند. (بر خلاف متغیر که فقط می‌تواند یک مقدار ذخیره کند.) مقادیر آرایه در بیشتر زبانهای برنامه‌نویسی باید از یک نوع باشد. (یعنی همه‌ی مقادیر یا باید عدد باشند یا همه‌ی مقادیر باید از نوع حروف باشند) بنابراین در این کتاب نیز قرارداد میکنیم که همه عناصر آرایه باید از یک نوع باشند. هر آرایه نامی دارد که با استفاده از نام آرایه و اندیس آرایه می‌توان به عناصر آرایه دسترسی پیدا کرد. برای درک بهتر، آرایه‌ی زیر را در نظر بگیرید:

X	۵	۱۲	۳	۹	۱	۱۴	۱۰	۲

آرایه‌ی A شامل ۸ مقدار است که برای دسترسی به مقادیر آن باید از اندیس خانه‌های آن استفاده کرد. مثلا عدد ۵ در اندیس (خانه) اول آرایه قرار دارد پس برای دسترسی به آن می‌توان نوشت: X[۱] .

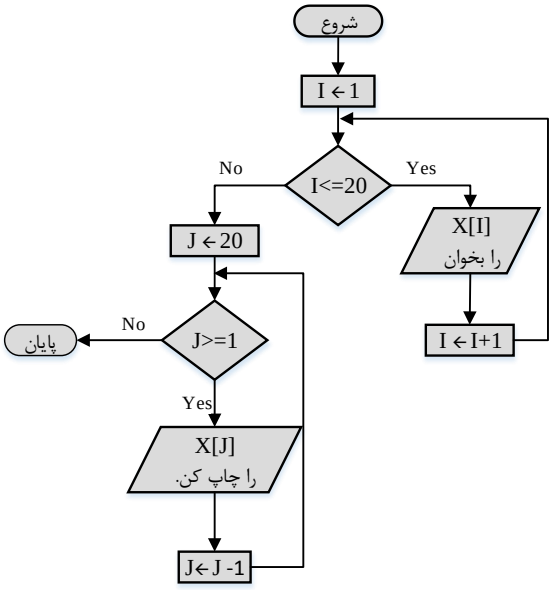
پس برای اینکه آرایه X را از ورودی بخوانید باید در الگوریتم به صورت زیر نوشت:

X [۸], X [۷], X [۶], X [۵], X [۴], X [۳], X [۲], X [۱] را بخوان.

واضح است که اینگونه نوشتن الگوریتم کار درستی نیست. به این دلیل که اگر تعداد عناصر آرایه زیاد باشد نوشتن الگوریتم غیر ممکن به نظر می‌رسد. در قالب مثال بعدی نحوه ی کار با آرایه‌ها برای خواندن و چاپ کردن عناصر آرایه را توضیح می‌دهیم.

مثال ۴۳

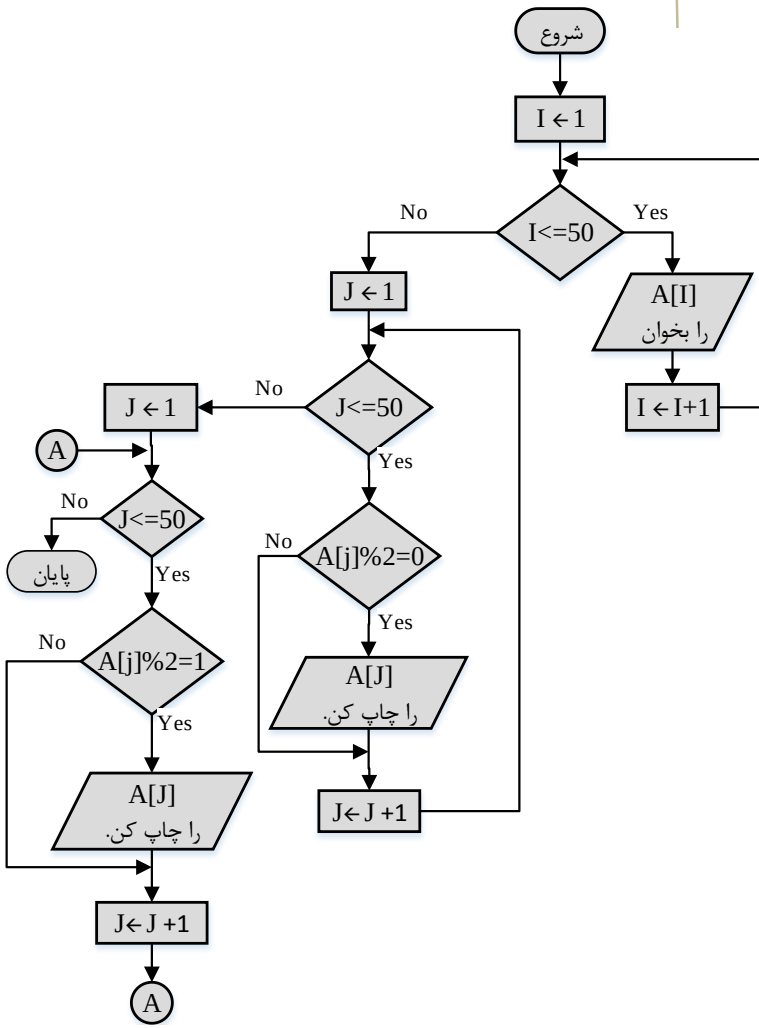
فلوچارت برنامه‌ای را رسم کنید که ۲۰ عدد از ورودی دریافت و اعداد را برعکس ترتیب ورودی آن‌ها را چاپ نماید. (مثلا اگر اعداد به ترتیب ۵، ۱۱، ۳ وارد شود در خروجی ۳، ۱۱، ۵ را



به نحوه‌ی خواندن عناصر آرایه دقت کنید. با استفاده از یک ساختار تکرار، شمارنده I از ۱ تا ۲۰ حرکت می‌کند و در هر مرحله تکرار X[I] را می‌خواند. پس وقتی I=۱ است در واقع X[۱] را می‌خواند، وقتی یک واحد به I اضافه می‌شود در مرحله‌ی بعد X[۲] را می‌خواند و ....

مثال ۴۴

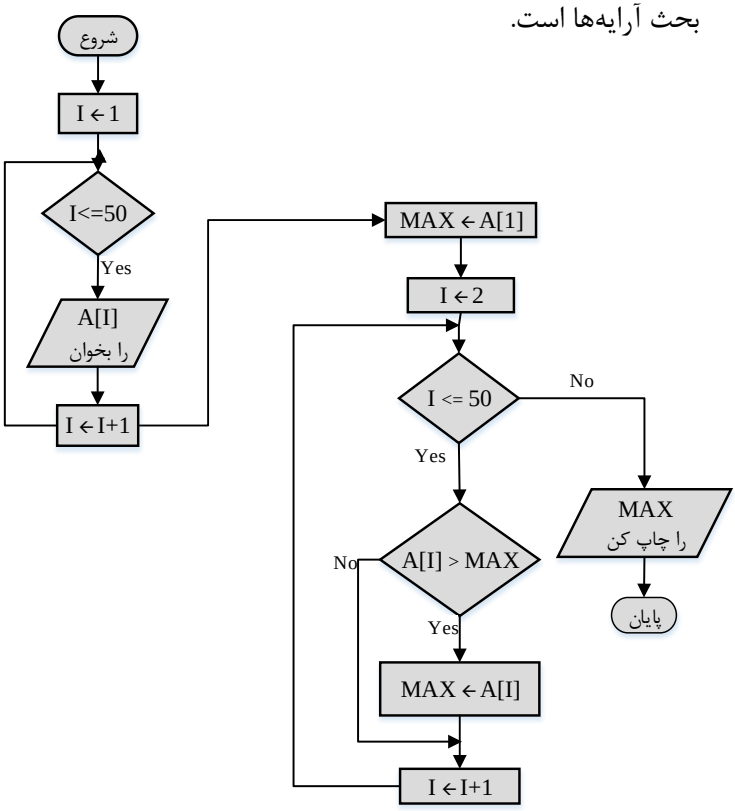
فلوچارت برنامه‌ای را رسم کنید که ۵۰ عدد از ورودی دریافت کند و پس از آن ابتدا اعداد زوج و سپس اعداد فرد را چاپ کند.



مثال ۴۵

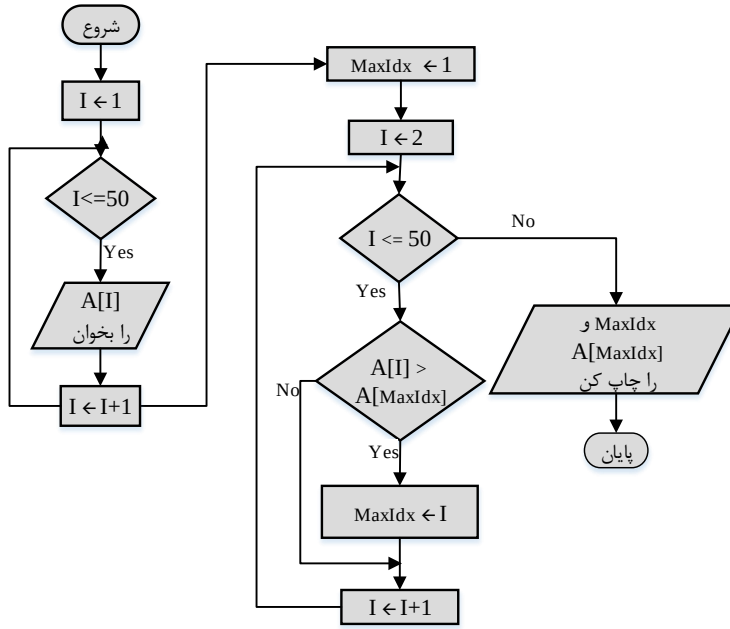
فلوچارت برنامه‌ای را رسم کنید که ۱۰۰ عدد از ورودی دریافت و بزرگترین عدد را بیابد. (با استفاده از آرایه)

این سوال را قبلا بدون استفاده از آرایه حل کردیم. با مقایسه فلوچارت این سوال و سوال ۲۹ متوجه خواهید شد که فرق چندانی بین این دو فلوچارت نیست و ذکر این مثال در اینجا صرفا جهت تمرین بیشتر در بحث آرایه‌ها است.



در روش بالا فرض را بر این گذاشتیم که عدد اول بزرگترین عدد است و از خانه دوم تا آخر آرایه حرکت کردیم و صحت ادعا را بررسی کردیم. یعنی اگر عددی بزرگتر از max ملاقات کردیم، آن عدد را در max ذخیره کردیم.

در روش دوم میخواهیم علاوه بر بزرگترین عدد، اندیس خانه ای که بزرگترین عدد در آن ذخیره شده است هم مشخص کنیم. بنابراین از متغیری به نام MaxIdx استفاده میکنیم و در ابتدا آن را برابر یک قرار می‌دهیم، به این معنی که فرض میکنیم که بزرگترین عدد در خانه اول قرار دهد و میخواهیم صحت این ادعا را بررسی کنیم.

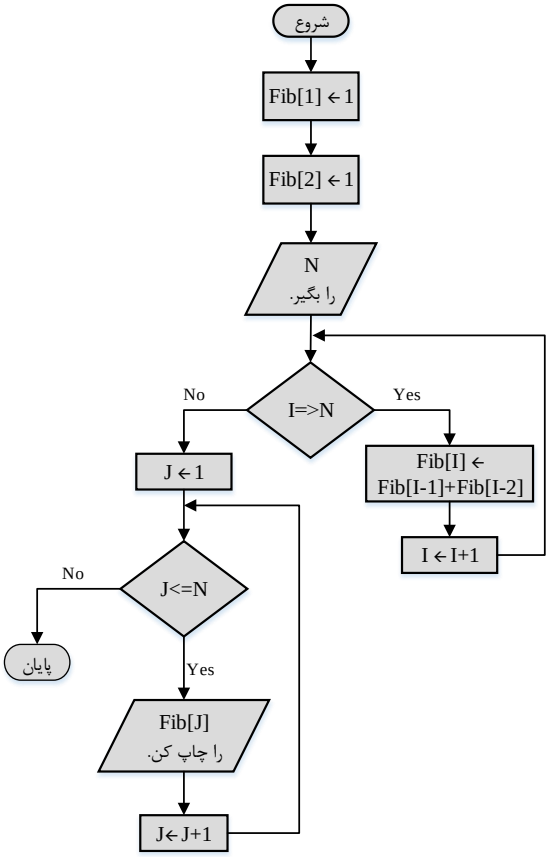


توجه داشته باشید که در MaxIdx اندیس بزرگترین عدد ذخیره شده است نه خود عدد. یعنی اگر بزرگترین عدد آرایه در خانه ۲۵ باشه در این صورت MaxIdx=۲۵ و بزرگترین عدد در A[MaxIdx] یا همان A[۲۵] ذخیره شده است.

مثال ۴۶

فلوچارت برنامه‌ای را رسم کنید که جملات اول تا nام جمله‌ی فیبوناتچی را در یک آرایه محاسبه کند.

در مثال ۳۶ بدون استفاده از آرایه این مثال حل شده است. در این مثال با استفاده از آرایه، دنباله‌ی فیبوناتچی را چاپ می‌کند.



بد نیست بدانیم...

۲

فیبوناچی
دنباله جادویی

لئوناردو فیبوناچی ، ریاضیدان ایتالیایی قرن سیزدهم با حل مسئله خرگوش‌ها به دنباله‌ای از اعداد رسید که خواص شگفت انگیز و کاربردهای فراوان آن نه تنها ریاضیدانان بلکه دانشمندان بسیاری از رشته‌های دیگر را به خود جلب کرده است. دنباله فیبوناچی زیر را در نظر بگیرید:

۱, ۱, ۲, ۳, ۵, ۸, ۱۳, ۲۱, ۳۴, ۵۵, ۸۹, ۱۴۴, ۲۳۳, ۳۷۷, ۶۱۰, ...

نکته شگفت انگیز در این دنباله، حاصل تقسیم هر جمله از این دنباله بر جمله قبل از خود است. یعنی اگر در دنباله بالا هر جمله را بر جمله قبل از خود تقسیم کنیم به دنباله زیر می‌رسیم:

۱, ۲, ۱/۵, ۱/۶۶۶, ۱/۶ , ۱/۶۲۵, ۱/۶۱۵, ۱/۶۱۹, ۱/۶۱۷, ۱/۶۱۸, ۱/۶۱۸, ۱/۶۱۸, ...

همانطور که می‌بینید حاصل این تقسیم به ۱/۶۱۸ همگراست. اما این عدد چیست که شگفت انگیزی فیبوناچی را نشان می‌دهد؟ اگر این عدد را در گوگل جستجو کنید به Golden ratio یا همان عدد طلایی می‌رسید. بد نیست کمی از عدد طلایی صحبت کنیم.

اگر از شما این سوال پرسیده شود که هرم فراعنه و نقاشی مونالیزای داوینچی چه وجه اشتراکی با توییتر و پپسی دارند؟ تا حدی عجیب به نظر می‌رسد. ولی پاسخ سریع این سوال این است: همه آنها با استفاده از نسبت طلایی طراحی شده‌اند. نسبت طلایی زمانی به دست می‌آید که یک پاره‌خط به دو بخش تقسیم شود و اگر بخش طولانی‌تر (a) را بر بخش کوتاه‌تر (b) تقسیم کنیم، برابر با تقسیم مجموع (a) + (b) بر (a) باشد که هر دو مقدار برابر با ۱/۶۱۸ هستند.

در طبیعت به وفور از نسبت طلایی استفاده شده است. شاخ و برگ درختان، دانه ها آفتابگردان، بدن انسان، گردبادها و منظومه‌ها و خیلی موارد دیگر.

مغز انسان طوری طراحی شده است که اشیا و تصاویری را ترجیح می‌دهد که در آنها نسبت طلایی رعایت شده است.

بد نیست بدانیم...

ساختار مغز (که تمایل دارد اشیایی را ببینید که در آنها نسبت طلایی رعایت شده است)، دانشمندان و محققین را قانع می‌کند که توجه ویژه ای به این نسبت داشته باشند. از اینرو نسبت طلایی در بازاریابی، عکاسی، معماری و حتی دندانپزشکی کاربرد دارد.

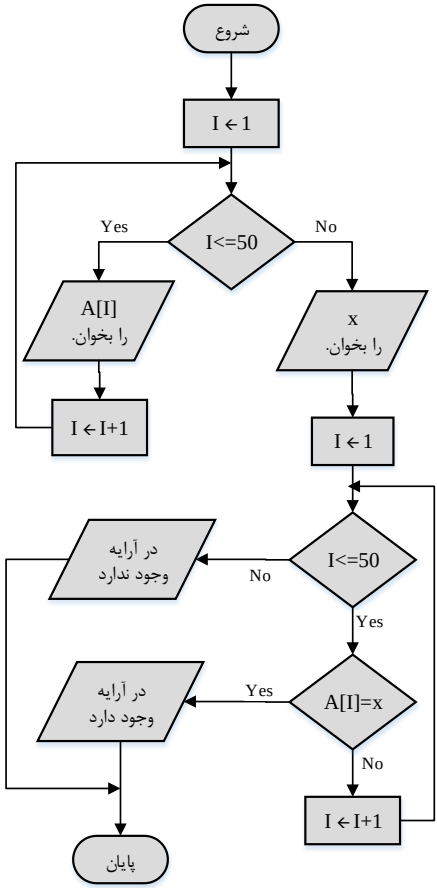
حال به دنباله فیبوناچی بازگردیم! یکی از کاربردهای مهم دنباله فیبوناچی در بورس است. پایه تحلیل تکنیکال در بورس دنباله فیبوناچی است. در این تحلیلها تنها به تقسیم هر جمله به جمله قبل از خود توجه نمی‌شود، بلکه تقسیم هر جمله به جمله بعد از خود، دو جمله بعد از خود، دو جمله قبل از خود.... حائز اهمیت است. مثلا اگر بیایید هر جمله فیبوناچی را بر دو جمله قبل از خود تقسیم کنید به عدد ۲/۶۱۸ می‌رسید.

بد نیست بدانید تحلیلگران بورس اعداد ۰/۳۸۲، ۰/۶۱۸، ۱/۶۱۸ و ۲/۶۱۸ را به ترازهای فیبوناچی می‌شناسند و در تحلیل های خود به وفور از این اعداد استفاده می‌کنند. شاید الانی که شما در حال خواندن این نوشته هستید آنها با همین اعداد در حال برداشت میلیونها تومان (و چه بسا میلیاردها تومان) سود هستند! بد نیست شما هم دانش خود را در این زمینه بالا ببرید. اگر علاقه‌مند به وردو به دنیای بورس هستید در آپارت احسان کامیابی‌نیا را جستجو کنید.

چنانچه به اعداد فیبوناچی و یا بورس علاقه‌مند شدید در آدرس jahangirics.ir مطالب مربوط را جستجو کنید

مثال ۴۷

فلوچارت برنامه‌ای را رسم کنید که یک آرایه از اعداد از ورودی دریافت کند (۵۰ عدد) سپس عدد دیگری از ورودی دریافت کند. در صورتی که عدد در آرایه وجود داشته باشد پیغام مناسبی چاپ کند.



در فلوچارت روبرو ابتدا ۵۰ عدد را در آرایه A قرار می‌دهد. سپس x را از ورودی می‌خواند و در کل آرایه جستجو میکند. (با مقایسه تک تک عناصر آرایه A با x) چنانکه خانه ای پیدا شد که مقدار داخل خانه با x برابر باشد در آرایه وجود دارد چاپ می‌کند در غیراینصورت به سراغ خانه بعدی می‌رود. چنانچه کل آرایه پیمایش شد و عددی یافت نشد در آرایه وجود ندارد چاپ می‌شود.

مثال ۴۸

فلوچارت برنامه‌ای را رسم کنید که با استفاده از روش غربال، اعداد اول کمتر از ۵۰ را بیابد و چاپ کند.

الگوریتم غربال: اعداد یک تا ۵۰ را به صورت زیر پشت سرهم بنویسید

۱- عدد یک را خط بزنید.

۲- دور عدد ۲ خط بکشید و همه مضرب‌هایش را خط بزنید

۳- دور عدد ۳ خط بکشید و همه مضرب‌هایش را خط بزنید

۴- عدد ۴ قبلا خط خورده است. پس به سراغ عدد بعدی بروید

۵- دور عدد ۵ خط بکشید و همه مضرب‌هایش را خط بزنید

.

.

این روند را تا عدد ۵۰ ادامه دهید تا همه‌ی اعداد غیر اول خط بخورند (در ریاضیات قضیه‌ای وجود دارد که بیان می‌کند که می‌توانید روند تشخیص عدد اول را تا $\sqrt{۵۰}$ دنبال کنید کافیست.)

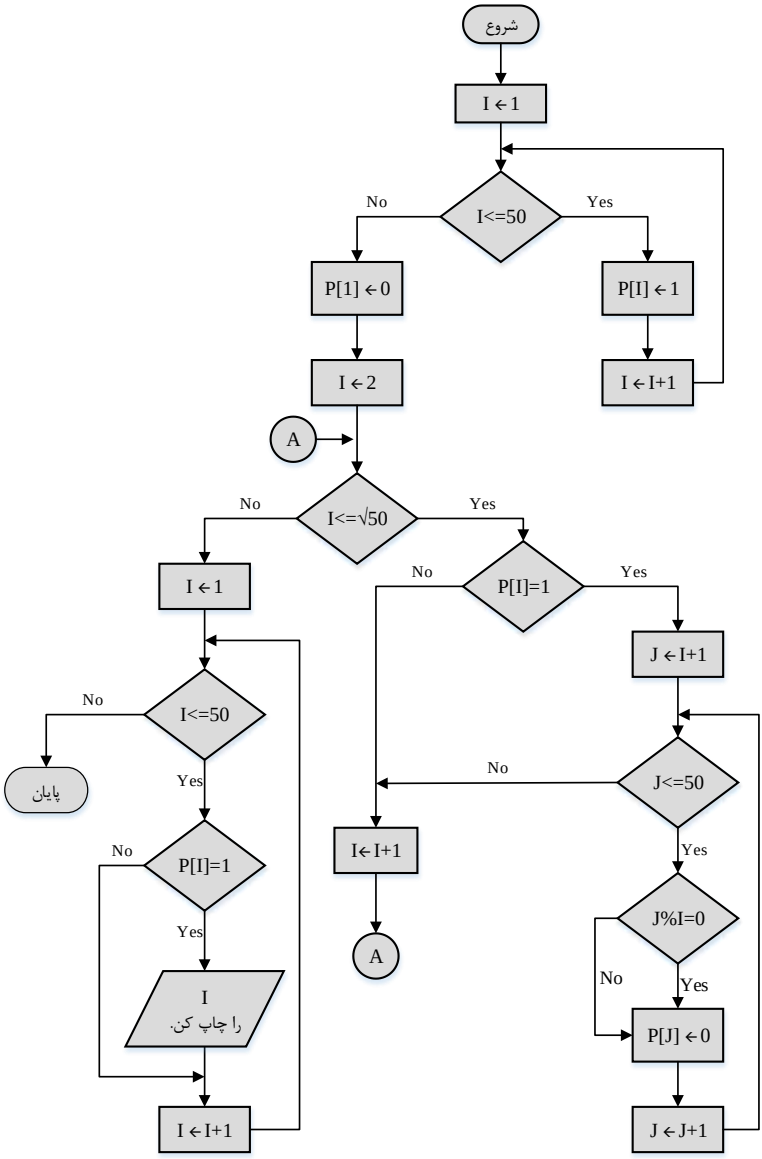
۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
۱۱	۱۲	۱۳	۱۴	۱۵	۱۶	۱۷	۱۸	۱۹	۲۰
۲۱	۲۲	۲۳	۲۴	۲۵	۲۶	۲۷	۲۸	۲۹	۳۰
۳۱	۳۲	۳۳	۳۴	۳۵	۳۶	۳۷	۳۸	۳۹	۴۰
۴۱	۴۲	۴۳	۴۴	۴۵	۴۶	۴۷	۴۸	۴۹	۵۰

سوالی که ممکن است برای شما پیش بیاید این است که چگونه می‌توانیم در فلوچارت خط زدن عدد یا دور عدد خط کشیدن را نمایش دهیم. برای حل این مشکل از یک ایده جالب استفاده می‌کنیم. به این صورت که یک آرایه به نام P تعریف می‌کنیم که دو مقدار می‌تواند داشته باشد. صفر یا یک.

اگر P[x] برابر یک باشد به این معنی است که x عدد اول است و اگر P[x] برابر صفر باشد به این معنی است که x اول نیست. بنابراین هر وقت به خط زدن یک عدد نیاز داشته باشید کافیست عنصر متناظر با آن عدد در آرایه را پیدا کنید و برابر صفر قرار دهید یعنی اگر میخواهید عدد x را خط بزنید P[x] را برابر صفر قرار می‌دهید.

نکته‌ی دیگری که باید به آن توجه کنید این است که ابتدا همه‌ی عناصر آرایه را با یک مقداردهی می‌کنیم. به این معنی که فرض را بر این می‌گذاریم که همه اعداد اول است. سپس خانه‌ی اول آرایه را صفر میکنیم. (معادل خط زدن در الگوریتم) و در ادامه هر جا که نیاز به خط زدن عدد باشد مقدار عنصر آرایه را برابر صفر قرار می‌دهیم.

فلوچارت صفحه‌ی بعد را به دقت بخوانید و در صورتی که در فهم آن با مشکل مواجه شدید به الگوریتم صفحه قبل مراجعه کنید و قدم به قدم با فلوچارت مطابقت دهید.

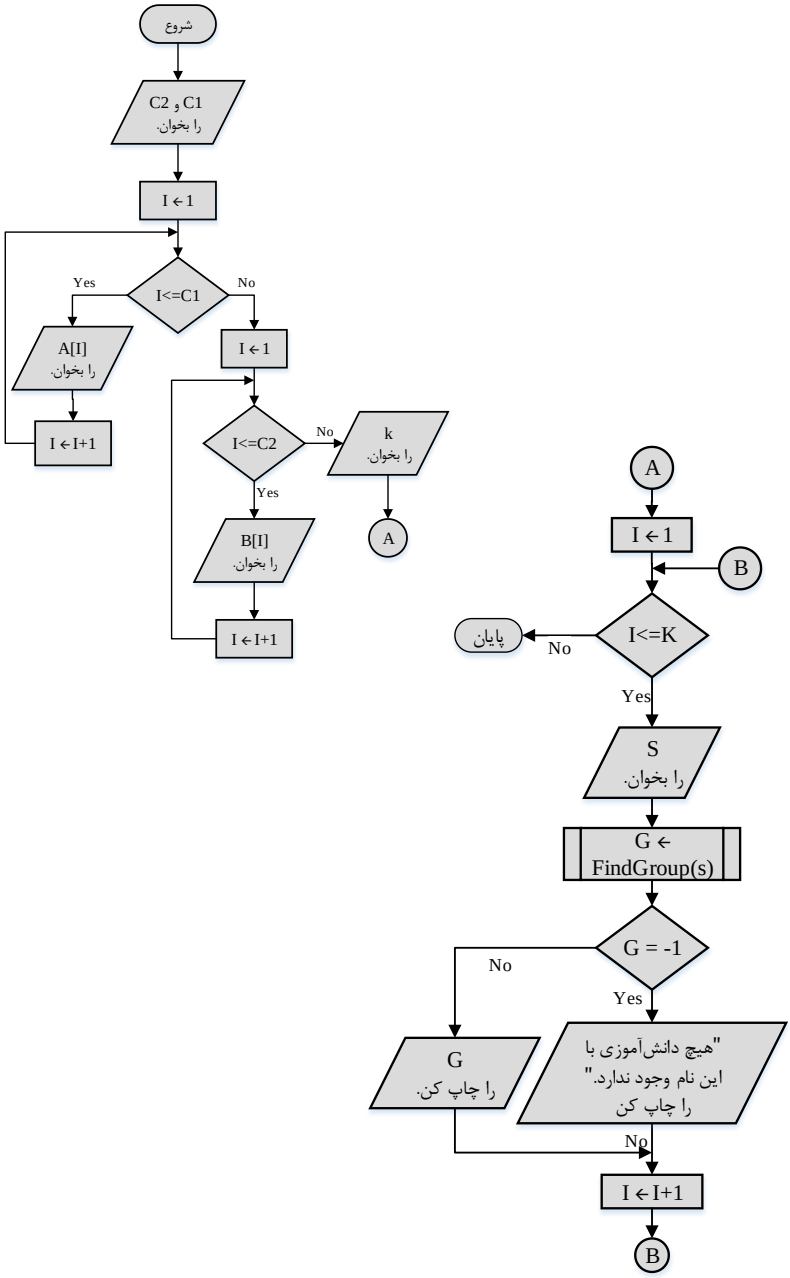

شروع
I ← 1
I<=50
No
Yes
P[1] ← 0
P[I] ← 1
I ← 2
I ← I+1
A
I<=√50
No
Yes
I ← 1
P[I]=1
No
Yes
J ← I+1
I<=50
No
پایان
Yes
J<=50
No
Yes
P[I]=1
I← I+1
J%I=0
No
Yes
Yes
I
را چاپ کن.
A
P[J] ← 0
I ← I+1
J ← J+1

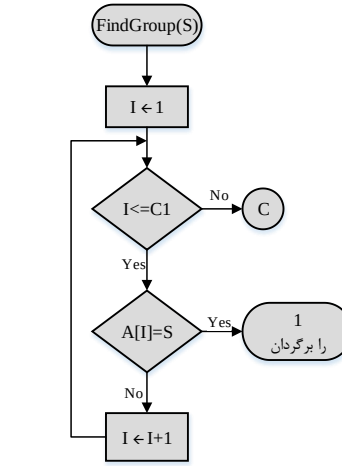
مثال ۴۹

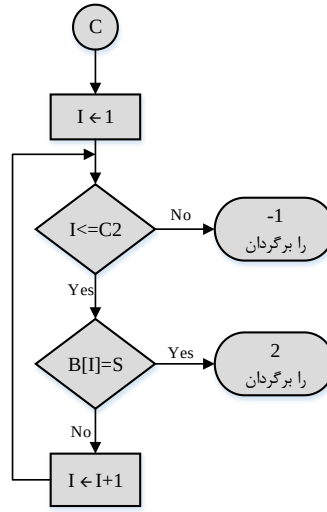
فلوچارت برنامه‌ای رسم کنید که اسامی افراد دو کلاس به همراه تعداد افراد هر کلاس را از ورودی بخواند و سپس k تا اسم (که k را از ورودی دریافت می‌کند) را از کاربر گرفته و مشخص کند که آن شخص متعلق به کدام کلاس است. (فرض کنید نام هیچ دو فردی یکسان نباشد.)

در حل این فلوچارت ابتدا تعداد افراد کلاس اول را در متغیر C۱ و تعداد افراد کلاس دوم را در متغیر C۲ از ورودی دریافت می‌کنیم و در ادامه اسامی افراد این دو کلاس را در دو آرایه A و B ذخیره می‌کنیم. سپس k اسم از ورودی دریافت می‌شود و با استفاده از زیر الگوریتم FindGroup تشخیص می‌دهیم که این اسم در کدام آرایه (کلاس) وجود دارد.

توجه داشته باشید زیرالگوریتم FindGroup(S) اسم S را ابتدا در آرایه A جستجو می‌کند. اگر در آرایه A باشد عدد یک را برمیگرداند و اگر در آرایه A نباشد به سراغ آرایه B می‌رود. اگر در آرایه B باشد ۲ را برمیگرداند و اگر در آرایه B نباشد ۱- را برمیگرداند که به این معنیست که این اسم در هیچکدام از آرایه‌ها (یا کلاسها) وجود ندارد.


شروع
C1 و C2
را بخوان.
I ← 1
I<=C1
Yes
No
A[I]
را بخوان.
I ← I+1
I ← 1
I<=C2
No
k
را بخوان.
Yes
B[I]
را بخوان.
I ← I+1
A
A
I ← 1
B
I<=K
پایان
No
Yes
S
را بخوان.
G ← FindGroup(s)
G = -1
No
Yes
G
را چاپ کن.
"هیچ دانش‌آموزی با این نام وجود ندارد."
را چاپ کن
No
I ← I+1
B


FindGroup(S)
I ← 1
I<=C1
No
C
Yes
A[I]=S
Yes
1
را برگردان
No
I ← I+1


C
I ← 1
I<=C2
No
-1
را برگردان
Yes
B[I]=S
Yes
2
را برگردان
No
I ← I+1

مثال ۵۰

فلوچارت برنامه‌ای رسم کنید که از ورودی یک آرایه به صورت صعودی مرتب دریافت کند. سپس از ورودی x دریافت کند و یه صورت دودویی x را در آرایه جستجو کند.

در مثال ۴۷ جستجوی ترتیبی را بررسی کردیم و در این مثال به دنبال فلوچارت جستجوی دودویی هستیم.

وقتی که یک آرایه مرتب باشد، بسیار سریعتر می‌توان بررسی کرد که آیا یک عنصر در آرایه قرار دارد یا خیر. قبل از بررسی الگوریتم به حل روند یک بازی خواهیم پرداخت!

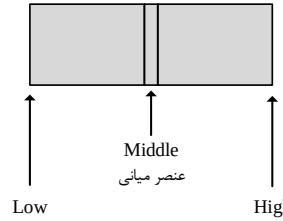
فرض کنید دو نفر این‌طور بازی می‌کنند که نفر اول یک عدد بین ۱ تا ۳۰۰ در نظر می‌گیرد و نفر دوم باید با پرسش اعدادی انتخابی ، بتواند آن عدد را بیابد. اگر نفر دوم عددی را بگوید و با عدد مورد نظر برابر باشد، نفر اول اعلام می‌کند و بازی تمام می‌شود. ولی اگر از آن بزرگتر باشد، نفر اول می‌گوید که آن عدد کوچکتر است و بر عکس.

یکی از بهترین روش‌های حل این مسئله این گونه است که ابتدا عدد میانی یعنی ۱۵۰ باید پرسیده شود. با پاسخ به این سوال از جانب نفر اول نصف اعداد برای پرسش بعدی حذف می‌شود. بدون کاسته شدن از کلیات فرض کنید که عدد موردنظر بیشتر از ۱۵۰ باشد. پس عدد موردنظر بین ۱۵۱ تا ۳۰۰ است. عدد میانی این بازه اعداد برابر ۲۲۵ است و نفر دوم آن را می‌پرسد که با شنیدن پاسخ این سوال نصف اعداد باقیمانده نیز حذف می‌شوند. این کار را

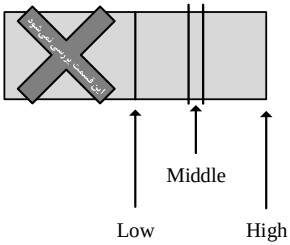
همین طور ادامه می‌دهیم تا فقط یک عدد باقی بماند. (یا یکی از اعداد میانی عدد مورد نظر باشد)

با این مقدمه به مساله اصلی برمیگردیم:

در جستجوی ترتیبی آرایه را از ابتدا تا انتها پیمایش میکنیم و x را با تک تک عناصر آرایه مقایسه می‌کنیم. ولی در جستجوی دودویی از مرتب بودن اعداد کمک می‌گیریم و ا بتدا عنصر میانی را با x مقایسه می‌کنیم، اگر برابر بود که x در آرایه وجود دارد، اگر کوچکتر بود نیمه‌ی سمت چپ و اگر بزرگتر بود نیمه‌ی سمت راست را مورد بررسی قرار می‌دهیم.



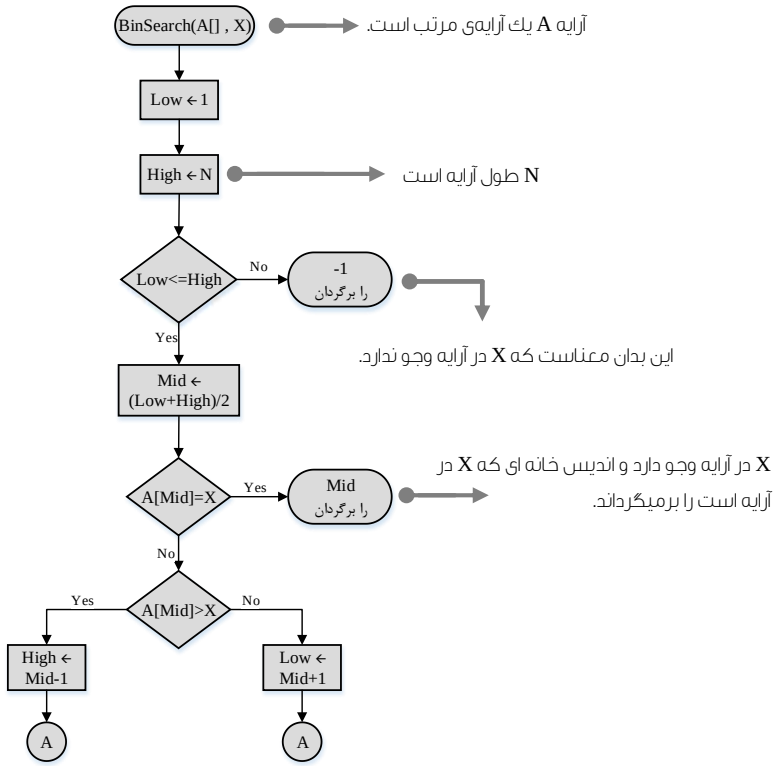
برای این که بتوان هر تکه از آرایه را مورد بررسی کنیم باید عنصر ابتدایی و انتهایی آن را در دو متغیر نگه داریم. (low,high)



در شکل بالا فرض را بر آن گذاشتیم که x از عنصر میانی بزرگتر است و بنابراین باید نیمه‌ی سمت راست را در ادامه ی الگوریتم بررسی کنیم. دقیقا همین روند برای سمت راست تکرار می‌شود. پس عنصر وسط را پیدا می‌کنیم و x را با آن مقایسه می‌کنیم.

اگر دقیق‌تر بخواهیم به الگوریتم نگاه کنیم در هر مرحله ما به یک عنصر میانی نیاز داریم (که اسم آن را middle انتخاب کردیم.) عنصر میانی با استفاده از عنصر ابتدایی (low) و عنصر انتهایی (high) بدست می‌آید. توجه داشته باشید که در هر مرحله با توجه به مقدار x و عنصر میانی یکی از دو متغیر low یا high تغییر مقدار می‌دهند. به این معنی که اگر x از عنصر میانی بزرگتر باشد پس باید نیمه سمت چپ حذف شود و بنابراین مقدار high ثابت و مقدار low به mid + ۱ تغییر کند. (چرا؟) با استدلال مشابه اگر x از عنصر میانی کوچکتر باشد پس بازه‌ی جستجوی ما نیمه ی سمت چپ آرایه است و مقدار high برابر mid-۱ خواهد شد.

ذکر این نکته ضروریست که مقادیر low,middle,high اندیس خانه های ابتدایی، میانی و انتهایی را نشان می‌دهند نه مقدار داخل آنها را.



زیرالگوریتم بالا آرایه A و عدد X به عنوان ورودی دریافت می‌کند و در صورتی که عدد در آرایه باشد اندیس آن و در صورتی که عدد در آرایه نباشد عدد ۱- را برمی‌گرداند. فلوچارت اصلی مثال را به عنوان تمرین رسم کنید.

مثال ۵۱

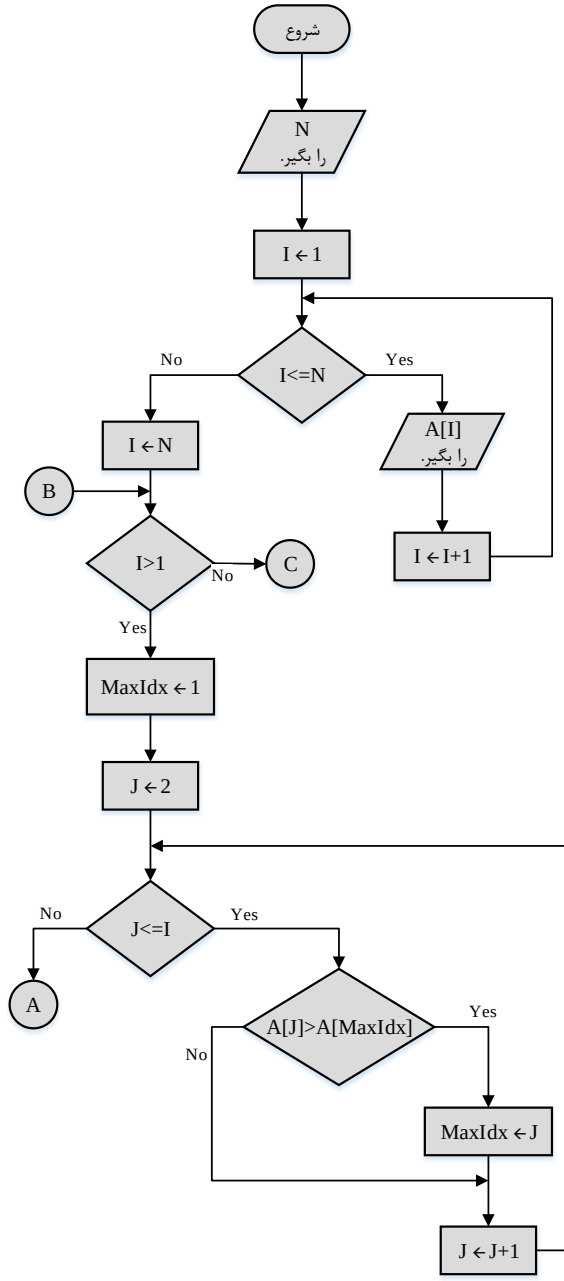
فلوچارت برنامه‌ای رسم کنید که از ورودی یک آرایه نامرتب دریافت کند و آرایه را به صورت صعودی مرتب کند..

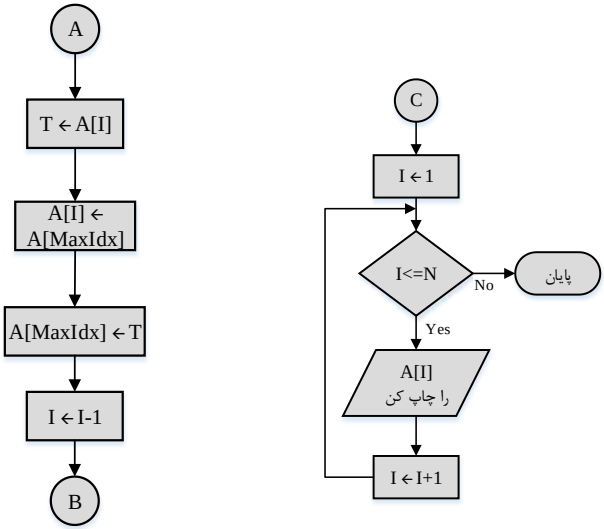
یکی از مسائلی که در برنامه‌نویسی خیلی به آن توجه شده است و الگوریتمهای زیادی برای آن نوشته شده است، مساله مرتب‌سازی است. در ادامه یکی از این الگوریتمها با نام مرتب‌سازی انتخابی توضیح داده می‌شود.

برای مرتب سازی یک آرایه با N عنصر در مرحله‌ی اول عنصر ماکزیمم را یافته و آن را با عنصر N ام آرایه جابجا میکنیم. (پس در مرحله‌ی اول بزرگترین عنصر آرایه در خانه‌ی آخر قرار می‌گیرد.) در مرحله‌ی دوم از بین عناصر ۱ تا N-۱ عنصر ماکزیمم را پیدا می‌کنیم و در خانه N-۱ قرار می‌دهیم. (پس در مرحله‌ی دوم، دومین بزرگترین عدد در خانه‌ی N-۱ ام قرار می‌گیرد.) به همین ترتیب ادامه می‌دهیم تا عناصر ماکزیمم به ترتیب در خانه‌های nام، n-۱ام، ... ۳ام و ۲ام قرار گیرد و آرایه مرتب شود.

علت نامگذاری روش انتخابی نیز به این دلیل است که در هر مرحله بزرگترین عدد انتخاب می‌شود و به سمت راست آرایه فرستاده می‌شود.

برای درک بهتر الگوریتم این روش، روش دوم مثال ۴۵ (MaxIdx) را دوباره با دقت بخوانید


شروع
N
را بگیر.
I ← 1
I<=N
No
Yes
I ← N
A[I]
را بگیر.
B
I>1
No
C
I ← I+1
Yes
MaxIdx ← 1
J ← 2
J<=I
No
Yes
A
A[J]>A[MaxIdx]
Yes
No
MaxIdx ← J
J ← J+1



برای درک بهتر الگوریتم، الگوریتم را قدم به قدم روی آرایه زیر دنبال می‌کنیم:

۱	۲	۳	۴	۵
۵۰	۹۰	۱۰۰	۸۰	۲۰

با اجرای الگوریتم مرتب‌سازی انتخابی بر روی آرایه‌ی بالا، عنصر اول (عدد ۵۰) به عنوان عنصر ماکزیمم به صورت پیش‌فرض انتخاب می‌شود. در پیمایش آرایه از اندیس J=۲ تا اندیس I (عنصر آخر) بزرگترین عنصر آرایه پیدا شده . در خانه‌ی I ام(عنصر آخر) قرار میگیرد. پس جای عدد ۱۰۰ که بزرگترین عدد آرایه است در اولین مرحله الگوریتم (که شامل پیمایش آرایه از خانه ۱ تا خانه N است) انتخاب می‌شود و با خانه‌ی N جابجا می‌شود.

N	I	J	MaxIdx	A[J]	A[MaxIdx]
۵	۵	۲	۱	۹۰	۵۰
۵	۵	۲	۲	۹۰	۹۰
۵	۵	۳	۲	۱۰۰	۹۰
۵	۵	۳	۳	۱۰۰	۱۰۰
۵	۵	۴	۳	۸۰	۱۰۰
۵	۵	۵	۳	۲۰	۱۰۰
۵	۵	۶	۳		۱۰۰

بعد از تکرار دور اول و پیدا شدن اندیس (جای) بزرگترین عدد (یا همان MaxIdx) مقادیر خانه MaxIdx و خانه I (که در این مرحله ۵ است) جابجا می‌شود:

۱	۲	۳	۴	۵
۵۰	۹۰	۲۰	۸۰	۱۰۰

در مرحله‌ی بعد یک واحد از I کم شده است و اینبار باید بزرگترین عدد در بازه ۱ تا N-۱ پیدا شود و جای بزرگترین عدد با خانه N-۱ عوض شود:

N	I	J	MaxIdx	A[J]	A[MaxIdx]
۵	۴	۲	۱	۹۰	۵۰
۵	۴	۲	۲	۹۰	۹۰
۵	۴	۳	۲	۲۰	۹۰
۵	۴	۴	۲	۸۰	۹۰
۵	۴	۵	۲		۹۰

و آرایه به صورت زیر در می‌آید:

۱	۲	۳	۴	۵
۵۰	۸۰	۲۰	۹۰	۱۰۰

مرحله ۳:

N	I	J	MaxIdx	A[J]	A[MaxIdx]
۵	۳	۲	۱	۸۰	۵۰
۵	۳	۲	۲	۸۰	۸۰
۵	۳	۳	۲	۲۰	۸۰
۵	۳	۴	۲		۸۰

آرایه :

۱	۲	۳	۴	۵
۵۰	۲۰	۸۰	۹۰	۱۰۰

مرحله‌ی ۴:

N	I	J	MaxIdx	A[J]	A[MaxIdx]
۵	۲	۲	۱	۲۰	۵۰
۵	۲	۳	۱		۵۰
۵	۱				

و در نهایت آرایه به صورت زیر مرتب می‌شود:

۱	۲	۳	۴	۵
۲۰	۵۰	۸۰	۹۰	۱۰۰

بد نیست بدانیم...

۳ مرتبه زمانی

در مثال آخر، الگوریتم مرتب‌سازی یک آرایه نامرتب را با نام مرتب سازی انتخابی بررسی کردیم. بد نیست بدانید که الگوریتم های زیادی برای مسئله مرتب سازی نوشته شده است. الگوریتم حبابی، درجی، ادغامی، سریع و هرمی از دیگر الگوریتم های مرتب‌سازی هستند. سوالی که پیش می‌آید این است که علت تنوع این الگوریتم‌ها چیست؟ آیا خروجی این الگوریتم‌ها با هم متفاوت است؟ اگر همه این الگوریتم‌ها خروجی یکسان تولید می‌کنند برتری هر کدام از اینها بر دیگری چیست؟

در جواب باید گفت خروجی همه این الگوریتم‌ها یکسان است ولی تفاوت در سرعت محاسبات و رسیدن به جواب است. فرض کنید الگوریتم ۱ در کمتر از یک ثانیه جواب درست را تولید کند و الگوریتم ۲ در یک دقیقه همان جواب را تولید کند. به نظر شما کدام الگوریتم مناسب تر است؟!

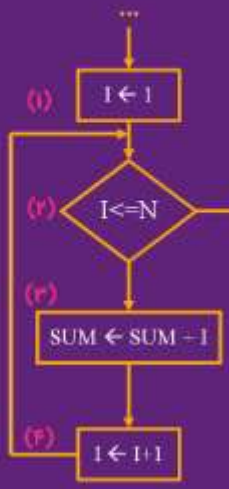
واضح است که الگوریتمی که در زمان کمتری به جواب می‌رسد الگوریتم مناسب‌تری است اما به نظر شما آیا این کار منطقی است که هر دو الگوریتم را پیاده‌سازی کنیم و بعد از پیاده سازی زمان اجرای آنها را اندازه‌گیری کنیم تا الگوریتم سریعتر را شناسایی کنیم؟ از آنجایی که سرعت اجرای الگوریتم به سرعت پردازنده کامپیوتر بستگی دارد آیا مقایسه دو الگوریتم متفاوت بر روی دو کامپیوتر با پردازنده های متفاوت روش درستی است؟

در مبحث مرتبه زمانی و پیچیدگی زمانی به دنبال بیان اصولی هستیم که بتوانیم الگوریتم‌ها را فارغ از قدرت پردازنده‌های کامپیوتری بررسی کنیم.

یک روش برآورد کارایی الگوریتم، شمارش تعداد عملیات اصلی مورد نیاز تا رسیدن به خروجی است

بد نیست بدانیم...

یک روش برآورد کارایی الگوریتم، شمارش تعداد عملیات اصلی مورد نیاز تا رسیدن به خروجی است. قطعه الگوریتم ساده‌ی زیر را در نظر بگیرید که مجموع اعداد ۱ تا n را محاسبه می‌کند. میخواهیم بررسی کنیم هر دستور الگوریتم چند بار اجرا می‌شود. با هم بررسی می‌کنیم:



دستور (۱) یک دستور ساده است که فقط یک بار اجرا می‌شود، دستور (۲) شرط ساختار تکرار است و N+1 بار اجرا می‌شود (قبل از اینکه علتش را بخوانید به چرایی اش فکر کنید) به این دلیل که در این ساختار I از ۱ تا N حرکت می‌کند، بنابراین دستورات این ساختار Nبار اجرا می‌شود اما توجه داشته باشید شرط ساختار به ازای Nبار درست است و وارد ساختار تکرار می‌شود و به ازای شرط I=N+1 نادرست است و از ساختار خارج می‌شود. بنابراین دستور شرط (۲) N+1 بار اجرا می‌شود

در ادامه دستور (۳) و (۴) که در ساختار تکرار وجود دارند هرکدام N بار اجرا می‌شوند. پس در مجموع ۳N+۲ عمل جمع و مقایسه و انتساب انجام می‌شود.

فرض کنید این مساله روش دیگری برای حل داشته باشد که با شمارش تعداد عملیات اصلی آن الگوریتم به فرمول $۳N^{۲}+۲$ برسیم به نظرتان کدام الگوریتم سریعتر است؟ الگوریتم ۱ یا الگوریتم ۲؟ برای درک بهتر مساله فرض کنید N=۱۰ باشد. الگوریتم ۱ به ۳۲ عمل جمع و مقایسه نیاز دارد و الگوریتم ۲ به ۳۰۲ عمل. واضح است که الگوریتم ۱ سریعتر است. همانطور که می‌بینید معیاری ساده برای مقایسه دو الگوریتم فارغ از سخت افزار کامپیوتر ارائه کردیم. بحث مرتبه زمانی یک بحث مفصل و مهم در علوم کامپیوتر و برنامه نویسی است که در درس های ساختمان داده و طراحی الگوریتم به آن پرداخته می‌شود. اگر به این مبحث علاقه‌مند شدید «پیچیدگی زمانی» را در گوگل جستجو کنید.

تمرین ۸

۱. فلوچارت برنامه‌ای رسم کنید که نمره ده دانشجو را از ورودی دریافت کند و در آرایه ذخیره کند و اعمال زیر را انجام دهد:

الف- میانگین نمرات دانشجویان را چاپ کند.

ب- بالاترین نمره و کمترین نمره را چاپ کند.

ج- تعداد دانشجویانی که نمره‌ی کمتر از ۱۲ گرفته‌اند را چاپ کند.

۲. فلوچارت برنامه‌ای رسم کنید که یک آرایه از ورودی بگیرد و جای عناصر آرایه را به صورت زیر با هم جابجا کند:

جای عنصر اول را با عنصر آخر عوض کند.
جای عنصر دوم را با عنصر یکی مانده به آخر عوض کند.
...
عنصر وسط در جای خود باقی می‌ماند.

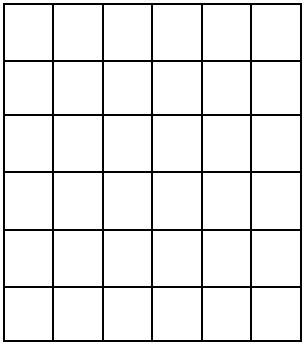
۵	۱۰	۳	۱	۲	۶	۹

ورودی

۹	۶	۲	۱	۳	۱۰	۵

خروجی

آرایه می‌تواند **چندبعدی** هم باشد، مثلا آرایه دو بعدی به شکل زیر در نظر بگیرید:



آرایه A ۶ سطر و ۶ ستون دارد. برای دسترسی به هر کدام از خانه‌های آرایه به دو اندیس نیاز داریم. اندیس اول شماره سطر و اندیس دوم شماره ستون را مشخص می‌کند. مثلا A[۱][۱] خانه‌ی واقع شده در سطر اول و ستون اول دارد و A[۲][۳] خانه واقع شده در سطر دوم ستون سوم را دارد.

در آرایه‌ی دو بعدی (مانند آرایه یک بعدی) مساله پیمایش آرایه اهمیت زیادی دارد. منظور از پیمایش آرایه این است که بتوانیم همه عناصر آرایه (و یا بخشی از عناصر آرایه) را ملاقات کنیم. در ساده‌ترین حالت آرایه دو بعدی را می‌توان به دو صورت سطری و یا ستونی پیمایش کرد. پیمایش سطری به این معناست که ابتدا سطر اول به صورت کامل از چپ به راست پیمایش شود. سپس سطر دوم و به این ترتیب همه سطرها به صورت کامل پیمایش می‌شوند. در پیمایش ستونی همین روند را برای ستونهای آرایه دنبال میکنیم. یعنی ابتدا ستون

اول، سپس ستون دوم و این کار را تا ستون آخر ادامه می‌دهیم. برای اطمینان از درک درست از این موضوع ماتریس (آرایه دوبعدی) زیر را در نظر بگیرید:

$$A = \begin{bmatrix} ۲ & ۱ & ۳ \\ ۷ & ۴ & ۸ \\ ۵ & ۶ & ۹ \end{bmatrix}$$

اگر ماتریس A را به صورت سطری پیمایش کنیم و عناصر آنها را به ترتیب در خروجی چاپ کنیم، خروجی به صورت زیر خواهد بود: (از چپ به راست)

۲, ۱, ۳, ۷, ۴, ۸, ۵, ۶, ۹

همچنین خروجی ماتریس A در صورتی که پیمایش به صورت ستونی انجام شود به شکل زیر خواهد بود: (از چپ به راست)

۲, ۷, ۵, ۱, ۴, ۶, ۳, ۸, ۹

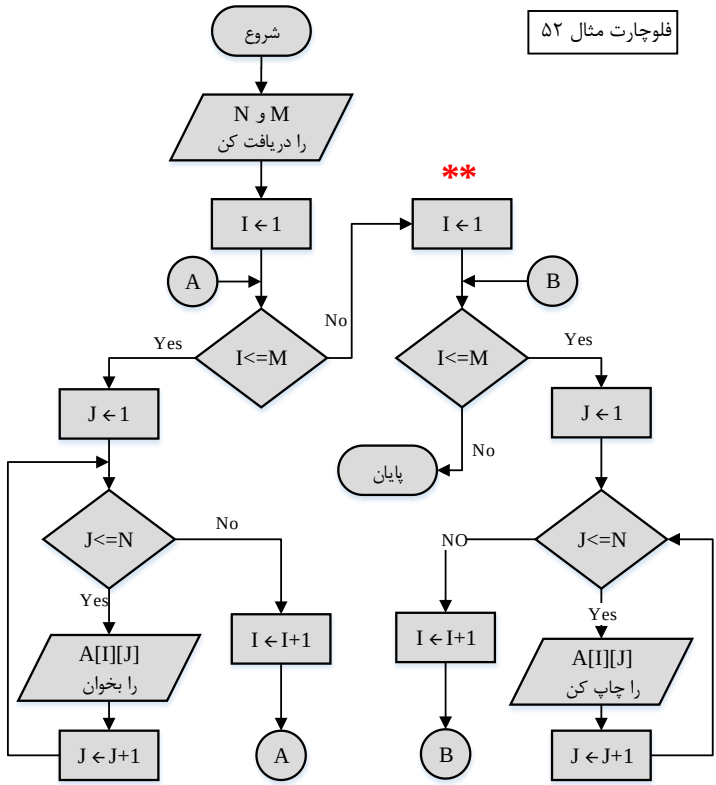
در ادامه با ذکر چند مثال آرایه‌ی دو بعدی را بیشتر توضیح می‌دهیم.

مثال ۵۲

فلوچارت برنامه‌ای رسم کنید که از ورودی یک ماتریس M×N دریافت کند و عناصر آن را به صورت سطری چاپ کند.

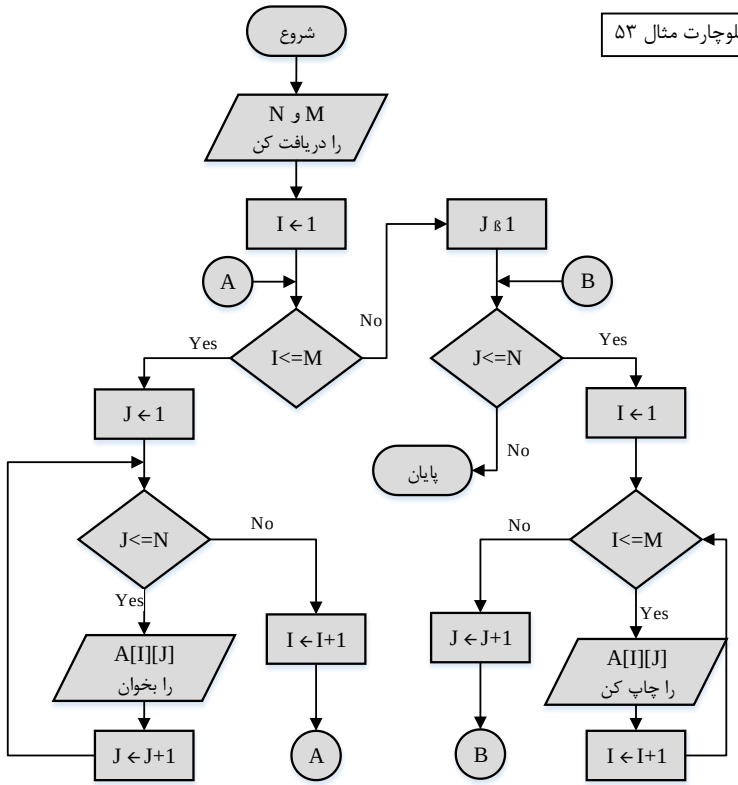
مثال ۵۳

فلوچارت برنامه‌ای رسم کنید که از ورودی یک ماتریس M×N دریافت کند و عناصر آن را به صورت ستونی چاپ کند.



در فلوچارت بالا ماتریس A به صورت سطری از ورودی دریافت می‌شود و در ادامه به صورت سطری ماتریس را چاپ می‌کند. اما پیمایش سطری چگونه انجام می‌شود؟ (برای درک بهتر، فرض کنید ماتریس را از ورودی دریافت کرده ایم توضیحات فلوچارت را از ** دنبال کنید) اندیس I ابتدا با ۱ مقدار دهی می‌شود و با مقدار ۱ وارد ساختار تکرار (I<=M) می‌شود. با مقدار ثابت I=۱ ساختار تکرار دوم شکل می‌گیرد که J از ۱ تا N تغییر می‌کند. پس I=۱ ثابت است و J از ۱ تا N تغییر می‌کند. بنابراین قابل تصور است که A[I][J] به ازای

I های ثابت (و برابر یک) و J های متغیر همه‌ی اول عناصر سطر را چاپ می‌کند. بعد از اینکه J>N شد از ساختار تکرار دوم خارج می‌شود به I یک واحد اضافه می‌شود و اینبار با مقدار ثابت I=۲ مراحل قبلی تکرار می‌شود و اینبار همه عناصر سطر دوم چاپ می‌شود. این روند تا سطر آخر ادامه می‌یابد.



پیمایش ستونی بسیار شبیه پیمایش سطری است. با این تفاوت که در پیمایش سطری به ازای Iهای ثابت وارد ساختار تکرار دوم می‌شدیم، ولی در پیمایش ستونی به ازای J های ثابت وارد ساختار تکرار دوم می‌شویم و در ساختار تکرار دوم مقادیر I تغییر می‌کند.

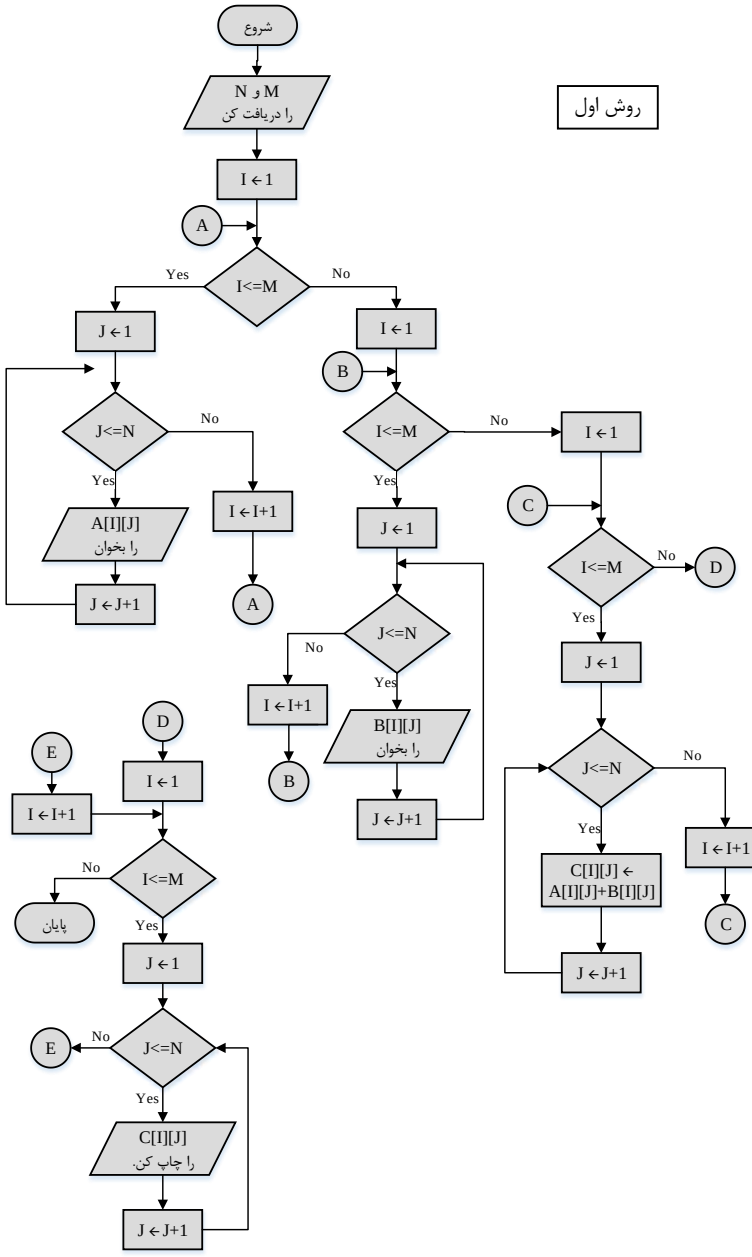
مثال ۵۴

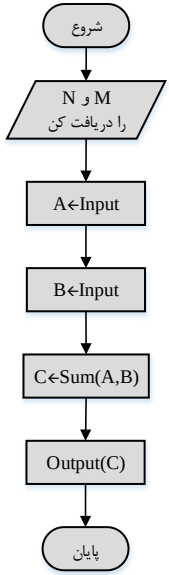
فلوچارت برنامه‌ای رسم کنید دو ماتریس M×N را از ورودی دریافت و مجموع آن‌ها را در ماتریس C ذخیره و چاپ نماید.

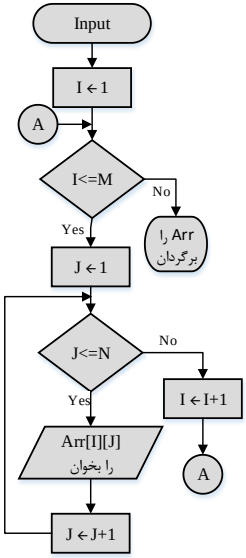
برای رسم فلوچارت این سوال باید دو ماتریس از ورودی دریافت کنیم (فلوچارت دریافت ماتریس از ورودی در مثال قبل توضیح داده شده است) و مجموع دو ماتریس را در ماتریس C ذخیره کنیم. الگوریتم جمع دو ماتریس به این صورت است که باید درایه‌های نظیر به نظیر دو ماتریس A و B را با هم جمع کنیم و در دارایه متناظر با ماتریس C قرار دهیم. برای این کار کافیست ماتریس را به صورت سطری پیمایش کنیم، در هر مرحله این عمل را انجام دهیم: C[I][J]=A[I][J] + B[I][J]

پس برای رسم کل این فلوچارت به ۴ پیمایش سطری جداگانه نیاز داریم. دو پیمایش برای خواندن ماتریس A و B، یک پیمایش برای محاسبه ماتریس C و یک پیمایش برای چاپ ماتریس C.

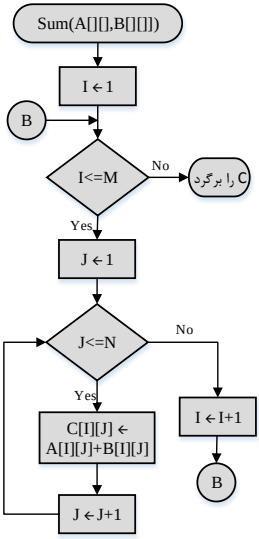
فلوچارت این مثال را با دو روش حل میکنیم. یکی بدون استفاده از زیر الگوریتم و دیگری با استفاده از زیر الگوریتم. با مقایسه این دو روش متوجه خواهید شد که نوشتن فلوچارت با زیرالگوریتم چقدر می‌تواند به قابل فهم شدن الگوریتم کمک کند.

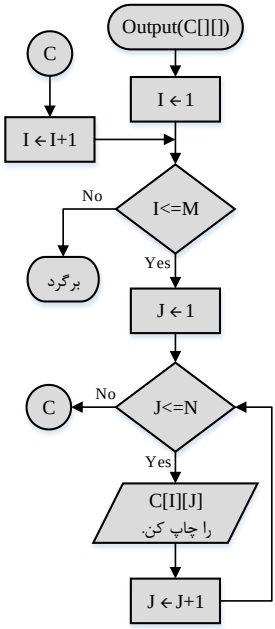

روش اول
شروع
M و N را دریافت کن
I ← 1
A
I<=M
Yes
No
J ← 1
J<=N
No
Yes
A[I][J] را بخوان
J ← J+1
I ← I+1
A
I ← 1
B
I<=M
No
Yes
J ← 1
J<=N
No
Yes
B[I][J] را بخوان
J ← J+1
I ← I+1
B
I ← 1
C
I<=M
No
D
Yes
J ← 1
J<=N
No
Yes
C[I][J] ← A[I][J]+B[I][J]
J ← J+1
I ← I+1
C
D
I ← 1
E
I ← I+1
I<=M
No
پایان
Yes
J ← 1
E
No
J<=N
Yes
C[I][J] را چاپ کن.
J ← J+1


شروع
M و N
را دریافت کن
A←Input
B←Input
C←Sum(A,B)
Output(C)
پایان


Input
I ← 1
A
I<=M
No
Yes
Arr را
برگردان
J ← 1
J<=N
No
Yes
I ← I+1
Arr[I][J]
را بخوان
A
J ← J+1

روش دوم

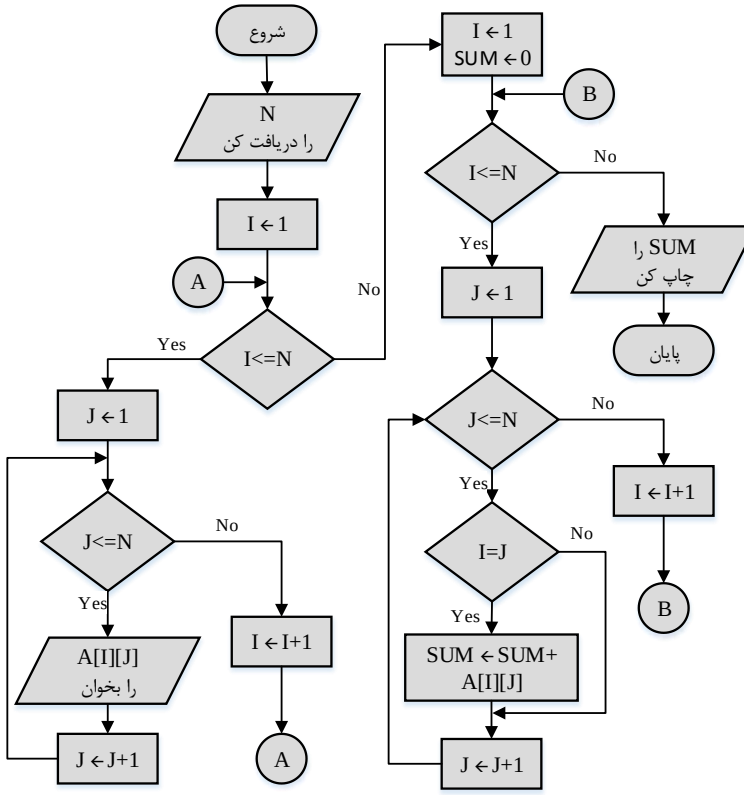

Sum(A[][],B[][])
I ← 1
B
I<=M
No
C را برگرد
Yes
J ← 1
J<=N
No
Yes
I ← I+1
C[I][J] ←
A[I][J]+B[I][J]
B
J ← J+1


Output(C[][])
C
I ← 1
I ← I+1
No
I<=M
برگرد
Yes
J ← 1
C
No
J<=N
Yes
C[I][J]
را چاپ کن.
J ← J+1

مثال ۵۵

فلوچارت برنامه‌ای رسم کنید که ماتریس A که یک ماتریس مربعی N×N است را از ورودی دریافت کند و مجموع عناصر قطر اصلی را چاپ کند.

برای رسم این فلوچارت می‌توانیم از ایده‌ی پیمایش سطری استفاده کنیم و در صورتی که I=J باشد A[I][J] را به متغیر sum اضافه کنیم:



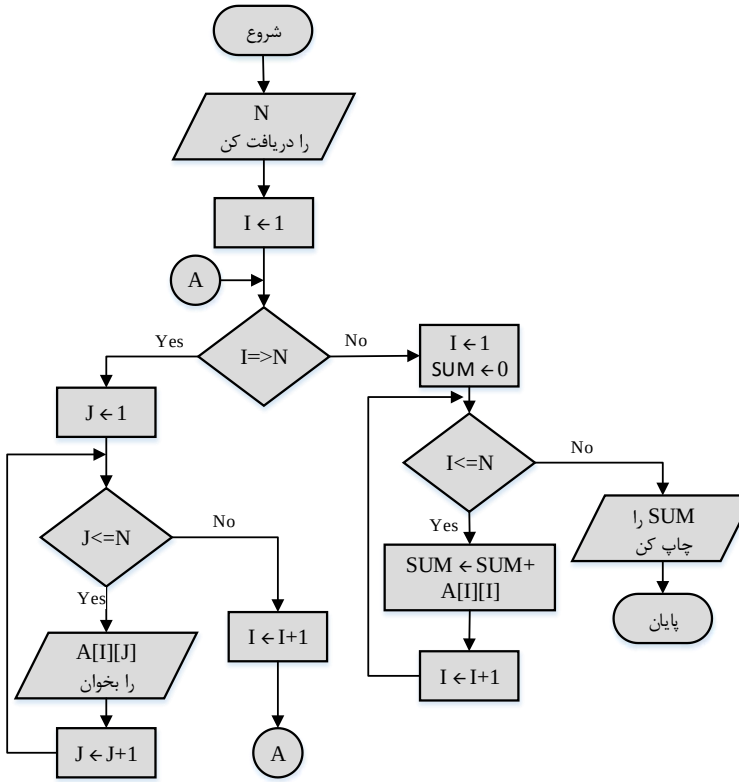
اما با یک ایده‌ی بهتر می‌توان فلوچارت ساده‌تری را رسم کرد. در ایده‌ی قبل الگوریتم کل ماتریس را پیمایش می‌کرد و فقط عناصر قطر اصلی را به SUM اضافه می‌کرد، که اگر ماتریس N×N داشته باشیم، در واقع $N^{۲}$-N دسترسی به عنصر اضافی انجام می‌پذیرد. چرا؟ به این دلیل که الگوریتم کل عناصر ماتریس که $N^{۲}$ عنصر است را پیمایش می‌کند و فقط به عناصر قطر اصلی (که N عنصر است) نیاز دارد. در روش بعد به دنبال راهکاری هستیم که فقط عناصر قطر اصلی را پیمایش کند. اگر ماتریس شما N×N باشد عناصر قطر اصلی ماتریس عبارت خواهند بود از :

A[۱][۱], A[۲][۲], A[۳][۳], …. , A[N][N]

با دقت به عناصر بالا متوجه میشوید که برای پیمایش قطر اصلی فقط به یک اندیس (مانند I) نیاز دارید و A[I][I] همان عناصر قطر اصلی هستند.

در صفحه بعد فلوچارت روش دوم رسم شده است.

همانطور که در فلوچارت صفحه‌ی بعد می‌بینید فلوچارت دومی از فلوچارت اولی کوتاهتر است. اما همیشه به خاطر داشته باشید که صرفا کوتاه نوشته شدن یک الگوریتم به تنهایی نمی‌تواند ملاک خوبی برای خوب تر بودن یک الگوریتم باشد. در این مثال فلوچارت دوم از فلوچارت اول بهتر است به این دلیل که عناصر کمتری را پیمایش می‌کند و درنتیجه سریعتر اجرا می‌شود. فلوچارت اول $N^{۲}$ عنصر را پیمایش می‌کند و فلوچارت دوم N عنصر.



در این کتاب سه پیماش سطری، ستونی و قطری بررسی شد. اما در حالت کلی پیمایش‌های دیگری نیز وجود دارند. مانند بالامثلثی، پایین مثلثی و قطر فرعی. در تمرینهای پیش‌رو به این پیمایشها توجه شده است.

تمرین ۹

۱- فلوچارت برنامه‌ای رسم کنید که یک ماتریس از ورودی دریافت کند و ترانهاده‌ی ماتریس را چاپ کند.

۲-فلوچارت برنامه‌ای رسم کنید که یک ماتریس از ورودی دریافت کند و تشخیص دهد که آیا ماتریس متقارن است یا خیر. (ماتریسی متقارن است که خودش با ترانهاده‌اش برابر است.)

۳- فلوچارت برنامه‌ای رسم کنید که یک ماتریس از ورودی دریافت کند و بزرگترین عنصر ماتریس را چاپ کند.

۴- فلوچارت برنامه‌ای رسم کنید که یک ماتریس از ورودی دریافت کند و تشخیص دهد که ماتریس قطری است یا خیر (یعنی همه‌ی عناصر به جز قطر اصلی صفر باشد)

۵- فلوچارت برنامه‌ای رسم کنید که یک ماتریس از ورودی دریافت کند و تشخیص دهد که ماتریس بالامثلثی است یا خیر. (یعنی همه عناصر زیر قطر اصلی صفر باشد.)

۶- فلوچارت برنامه‌ای رسم کنید که دو ماتریس A و B را از ورودی دریافت کند و ضرب دو ماتریس را در ماتریس C ذخیره کند.


دو تا کانتر جا انداختی
یه ساختار تکرار کم نوشتی
به آرایه هم که کلا اعتقاد نداری
جمعا میشه هفتصد هزارتومن
مگه خطی چند حساب کردی؟!

۶

فصل ششم

سخن پایانی

در این فصل نکات کلیدی و مهم برای ورود به دنیای برنامه نویسی مطرح می‌شود.

ورود به دنیای برنامه‌نویسی

اگر مطالبی که در این کتاب بیان شده است را به خوبی درک کرده باشید، باید بگوییم که شما آماده ورود به دنیای برنامه نویسی هستید. اولین سوالی که ممکن است در ذهن شما مطرح شود این است که چه زبان برنامه نویسی یاد بگیریم؟ پاسخ به این سوال ساده نیست و نیاز به طرح چند موضوع دارد.

اول اینکه شما باید هدفتان را مشخص کنید. اینکه میخواهید طراح سایت شوید؟ اپلیکیشن گوشی بنویسید یا نرم افزارهای کامپیوتری طراحی کنید؟ شاید یکی از سخت‌ترین سوالهایی که در ابتدای راه برنامه نویسی می‌شود از شما پرسید همین سوال است. فارغ از اینکه ممکن است شما به هر سه حوزه (یعنی برنامه نویسی وب، برنامه نویسی موبایل و برنامه نویسی دسکتاپ) علاقه داشته باشید از پیش نیازها و زبانهایی که برای هر حوزه نیازمند است اطلاعی ندارید. در ادامه به معرفی اجمالی چند زبان برنامه نویسی و حوزه فعالیت آن می‌پردازیم که با شناخت بهتری بتوانید به انتخاب آنها بپردازید.[۲]

این نکته هم در نظر داشته باشید که نه فقط شما بلکه هیچ توسعه دهنده ای دوست ندارد خودش را محدود به فقط یک حوزه کند، ولی برای شما که ابتدای راه برنامه نویسی هستید بهتر است ابتدا در یک حوزه با تجربه شوید و سپس برای یادگیری سایر حوزه ها اقدام کنید

[۲] مطالب این نوشته از سایت stackoverflow.com و learn.com۷ گرفته شده است.

جاوا اسکریپت یکی از محبوب‌ترین زبان‌های برنامه نویسی در دنیاست! می‌توان جاوا اسکریپت را یک زبان همه فن حریف دانست. در گذشته جاوا اسکریپت یک زبان اسکریپت‌نویسی برای صفحات وب بود، اما امروزه به یکی از پرکاربردترین زبان‌های برنامه‌نویسی تبدیل شده است. امروزه جاوا اسکریپت در پلتفرم‌های بسیار زیادی قابل استفاده است و دیگر تنها به صفحات وب خلاصه نمی‌شود، در حوزه‌ی اینترنت اشیا، بلاکچین، برنامه‌نویسی سمت سرور، برنامه‌نویسی اندروید، آی‌اواس، ویندوز و... تقریبا می‌توان رد پای این زبان محبوب را در هرجایی مشاهده کرد.

بد نیست بدانید کمبود برنامه نویس جاوااسکریپت در بازار به خوبی احساس می‌شود.

"سادگی و آسانی" ویژگی اصلی زبان پایتون است. یک گزینه مناسب برای کسانی که می‌خواهند برای اولین بار برنامه نویسی را یاد بگیرند. در واقع سادگی به این معنی است که شما با کمترین کدنویسی ممکن می توانید به سرعت به برنامه مورد نظرتان دسترسی داشته باشید و این سادگی به هیچ وجه قدرت این زبان را کم نمی کند. درون سازی کد در پایتون بسیار قوی است به این معنی که شما از کدهای جاوا و یا c++ می توانید درون زبان پایتون استفاده کنید. کتابخانه های زبان برنامه نویسی پایتون بسیار گسترده است و تقریبا می توان گفت که در هر موضوعی که بخواهید برنامه بنویسید کتابخانه پایتون در آن خصوص وجود دارد.

اگر به برنامه نویسی تحت وب علاقه مندید به احتمال زیاد در دو راهی PHP یا ASP.NET قرار خواهید گرفت PHP .با وجود فریمورک‌های قدرتمند و محبوبی همچون لاراول و سیمفونی حدود ۸۰ درصد از سهم بازار وب را به خود اختصاص داده است. ولی بد نیست بدانید که برنامه نویسان حرفه ای

انتقادهای جدی نسبت به این زبان برنامه نویسی داشتند. ولی این زبان با بروزرسانی خوبی که در سالهای اخیر داشته است به خوبی توانسته است نقاط ضعف خود را پوشش دهد. بد نیست بدانید سهم قابل توجهی از وب سایتهای موجود در ایران مانند دیجی کالا، اسنپ، اسنپ فود، نت برگ و ... زبان php را به عنوان زبان اصلی خود انتخاب کرده‌اند.

جاوا مناسب ترین انتخاب برای کسانی است که عاشق یک زبان برنامه نویسی قانونمند با ساختاری استاندارد هستند. طبق آخرین آمار انجمن برنامه نویسان TIOBE ، جاوا به لحاظ کارایی و استفاده، دارای بالاترین امتیاز در سرتا سر دنیا می باشد و رتبه اول را از آن خود نموده است. اگر قصد ورود به دنیای برنامه نویسی، مخصوصا برنامه نویسی اپلیکیشن های اندرویدی را دارید و می خواهید به یک متخصص اندروید تبدیل شوید جاوا گزینه بسیار مناسبی است.

از محصولات مایکروسافت که در ابتدای راه از لحاظ ساختاری شبیه به c++ بود و کم کم راه خود را از آن جدا کرد و توسعه یافت تا جایی که تفاوت‌های آنها بیشتر و بیشتر شد تا به امروز که می توان ادعا کرد که اگر شما C# را به عنوان زبان برنامه نویسی خود انتخاب کنید، میتوانید از آن به عنوان یک میانبر به دنیای برنامه نویسی استفاده کنید چرا که C# این امکان را برای شما فراهم می کند که در حوزه های مختلف نرم‌افزاری از جمله ساخت اپلیکیشن های اندرویدی، ساخت اپلیکیشن های مبتنی بر iOS ، برنامه نویسی وب وحتی نرم افزارهای دسکتاپی برنامه نویسی کنید. جالب است بدانید که در ایران تعصب مدیران شرکتهای کامپیوتری به سی شارپ از مدیران مایکروسافت بیشتر است!

یکی از زبانهای قدیمی که به عنوان الگویی موفق و الهام بخش برای دیگر زبان های برنامه نویسی یاد می شود و بسیاری از زبان های حال حاضر از آن مشتق گرفته شده اند. این زبان سطح میانی توانسته است در طی سال های فعالیت خود از طریق به کارگیری در شرکت های بزرگی همچون مایکروسافت، اوراکل،

ادوبی و... به شهرت فراوانی دست یابد. در دانشگاهها یادگیری برنامه نویسی تحت عنوان درسی آموزش برنامه نویسی سی پلاس پلاس به دانشجویان در رشته های فنی مهندسی تدریس می شود. تقریبا تمام سیستم های سطح پایین مانند سیستم عامل، سیستم های ماشینی و ... با استفاده از زبان C++ نوشته شده اند. پس اگر می خواهید یک برنامه نویس در سطح سیستم باشید و یا به برنامه نویسی در حوزه بازی های رایانه ای بپردازید، زبان C++ برای شما گزینه بسیار مناسبی است.

از دیگر زبانهای برنامه نویسی می‌توان به سوئیفت (توسعه نرم افزارهای مبتی بر سیستم عامل های اپل)، کاتلین (شبیه به زبان جاوا در توسعه نرم افزارهای اندرویدی) و Go (زبان برنامه نویسی بهینه در زمینه های وب، سرور و شبکه) اشاره کرد.

در این نوشته سعی کردیم به دور از هرگونه تعصب نسبت به هر زبان برنامه نویسی به معرفی آنها بپردازیم. ترتیب معرفی زبانها به هیچ وجه برتری آنها را نشان نمی‌دهد. همانطور که در ابتدای بحث گفتیم شما باید ابتدا هدفتان را مشخص کنید و در راستای اهدافتان به دنبال زبان برنامه نویسی مناسب باشید.

اما اگر با خواندن این سطور هنوز در انتخاب زبان برنامه نویسی مردد هستید پیشنهاد میکنیم خیلی به خودتان سخت نگیرید. این تردید از دانشِ کم شما در برنامه نویسی ناشی می‌شود. طبیعی است شمایی که در ابتدای راه برنامه

نویسی هستید با دیدن این همه زبان برنامه نویسی گیج شوید ولی بد نیست این نکته را در نظر داشته باشید که قرار نیست هر زبانی را که انتخاب کنید تا آخر با آن زبان ادامه بدید. مشکل همیشگی دانشجویان برنامه نویسی این است که بیشتر انرژیشان در ابتدای راهِ یادگیری، صرف انتخاب زبان برنامه نویسی می‌کنند. (و چه بسا درهمین مرحله خسته می شوند و بیخیال برنامه نویسی می شوند) اگرشما به یک زبان برنامه نویسی مسلط باشید یادگیری زبان برنامه نویسی جدید دشوار نیست.

و در آخر اگر هنوز به یک زبان برنامه نویسی واحد برای شروع یادگیری نرسیدید ما به شما پایتون رو پیشنهاد می‌دهیم.[۳]

[۳] نکات و راهنمایی های بیشتر در مورد شروع برنامه نویسی را در آدرس jahangirics.ir دنبال کنید. کافی است "شروع برنامه‌نویسی" را در این سایت جستجو کنید.

۷

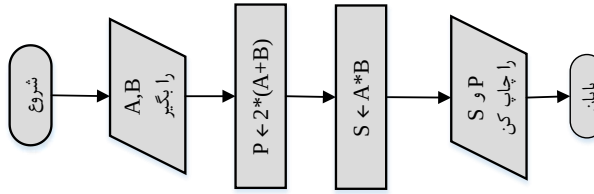
فصل هفتم حل تمرینهای منتخب

در این فصل به تعدادی از تمرینهای مطرح شده در فصول مختلف کتاب پاسخ داده می‌شود.

تمرین ۱

سوال ۱

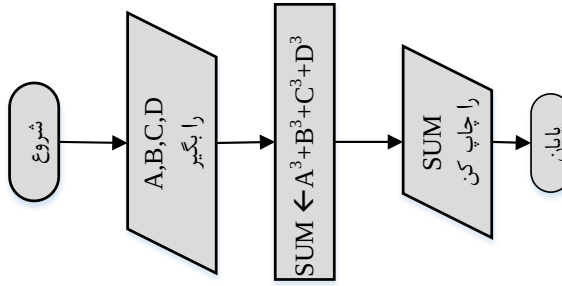
فلوچارت برنامه‌ای رسم کنید طول و عرض مستطیلی دریافت کند و محیط و مساحت آن را چاپ کند.



تمرین ۱

سوال ۳

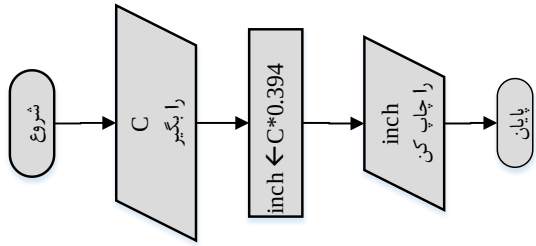
فلوچارت برنامه‌ای رسم کنید ۴ عدد دریافت کند و مجموع مکعبات آنها را چاپ کند.



تمرین ۲

سوال ۲

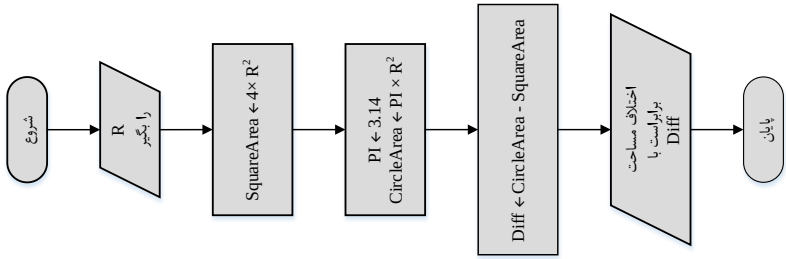
فلوچارت برنامه‌ای رسم کنید قد فرد را به سانتیمتر دریافت کند و به اینچ چاپ کند. (هر سانتیمتر ۰/۳۹۴ اینچ است.



تمرین ۲

سوال ۶

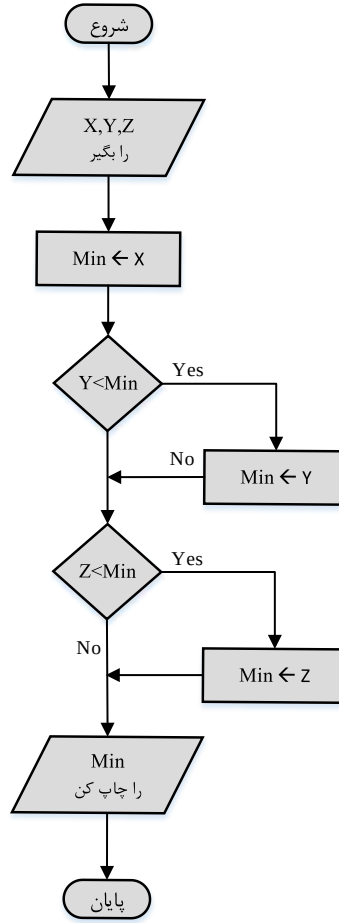
فلوچارت برنامه‌ای رسم کنید اختلاف مساحت دایره و مربع محیطی را رسم کند.



تمرین ۳

سوال ۲

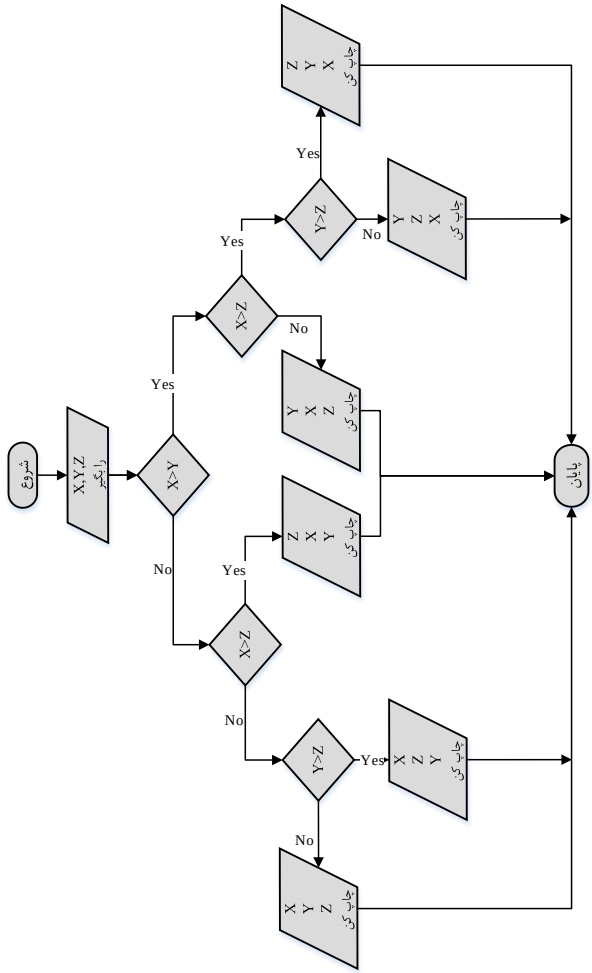
فلوچارت برنامه‌ای رسم کنید که کوچکترین عدد بین ۳ عدد را چاپ کند



تمرین ۳

سوال ۳

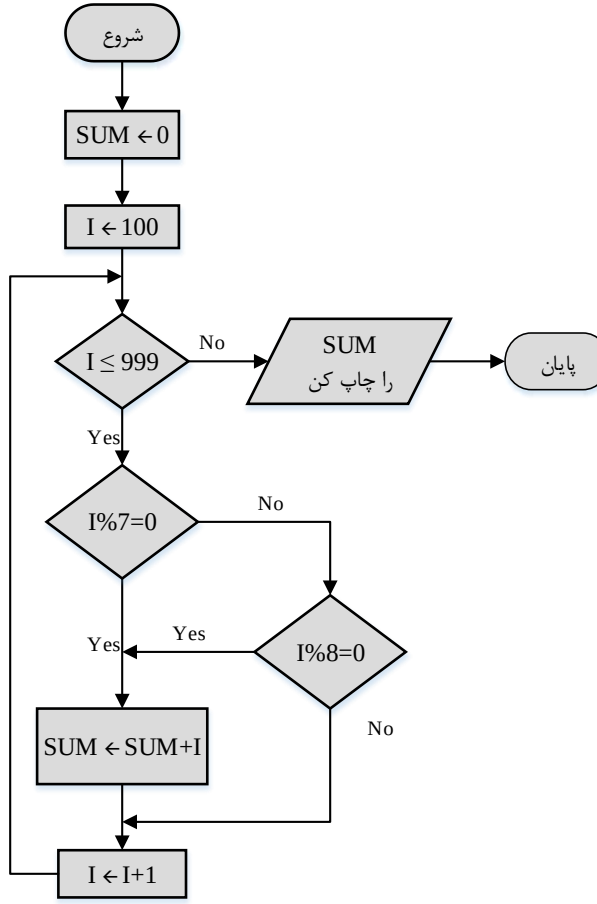
فلوچارت برنامه‌ای رسم کنید که ۳ عدد از ورودی دریافت کند و اعداد را به صورت صعودی چاپ کند.



تمرین ۶

سوال ۱

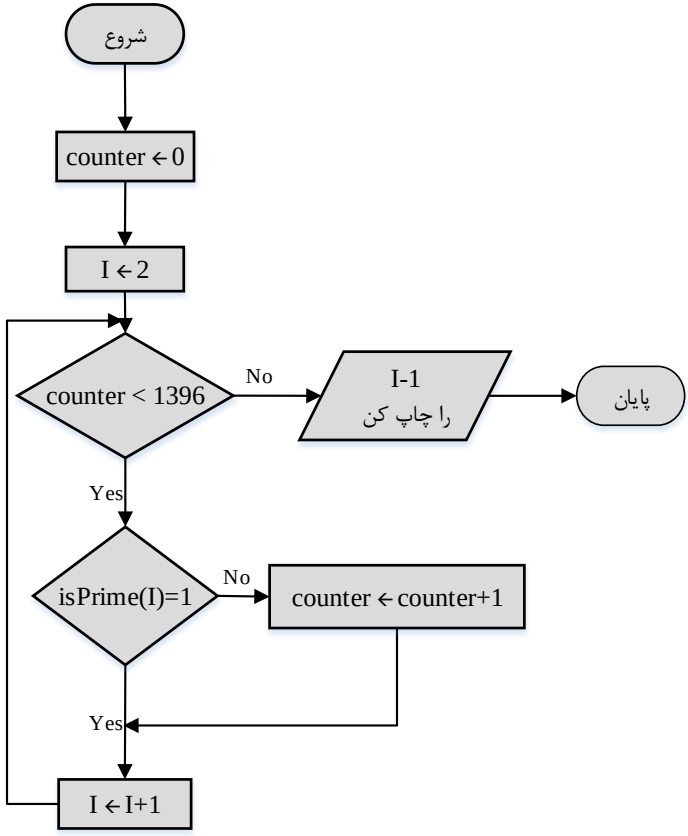
فلوچارت برنامه‌ای رسم کنید مجموع اعداد ۳ رقمی ضریب ۷ یا ضریب ۸ هستند را چاپ کند.



تمرین ۷

سوال ۱

فلوچارت برنامه‌ای رسم کنید که ۱۳۹۶ امین عدد اول را چاپ کند

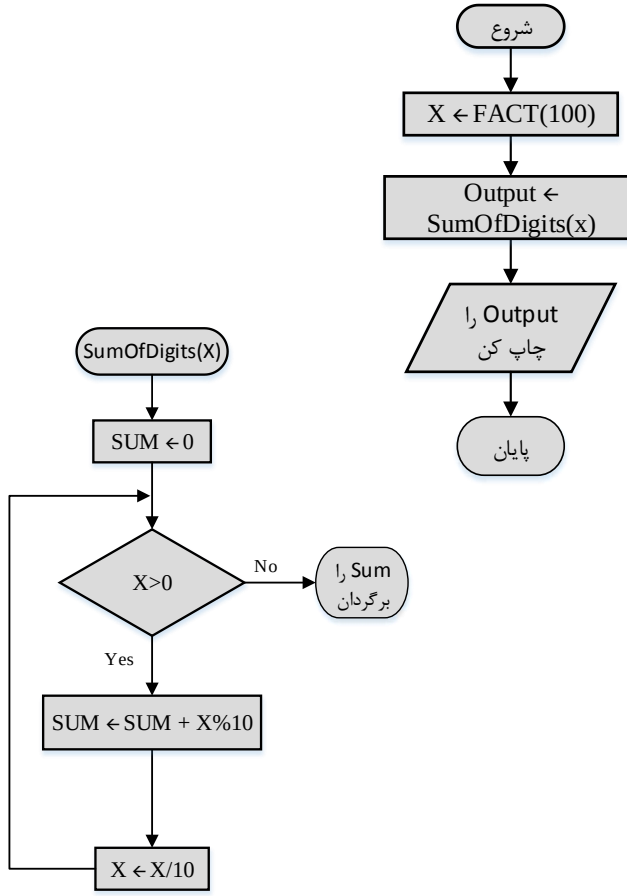


زیرالگوریتم isPrime در مثال ۳۸ پیاده سازی شده است. همچنین توجه کنید که در انتها، I-۱ به عنوان جواب چاپ می‌شود. (چرا؟)

تمرین ۷

سوال ۳

فلوچارت برنامه‌ای رسم کنید مجموع ارقام ۱۰۰! را چاپ کند

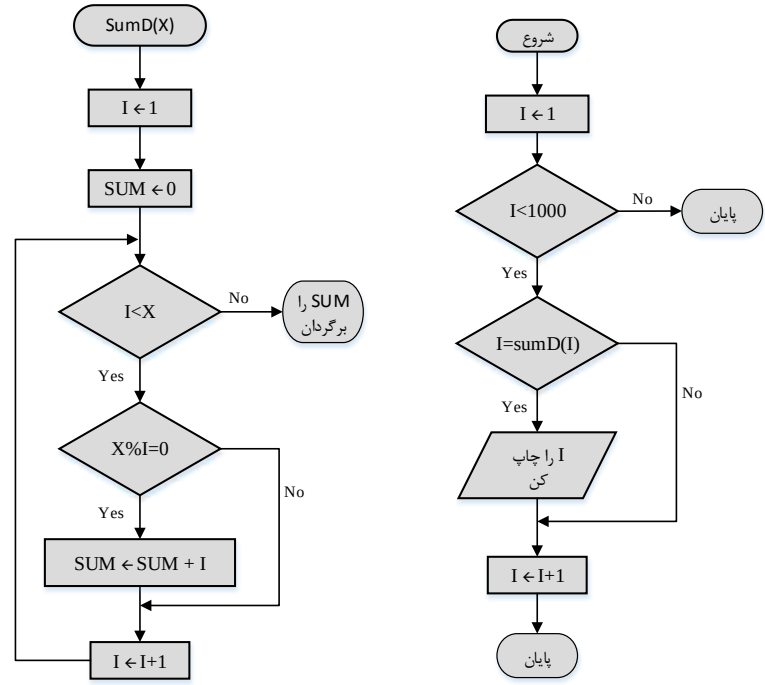


زیرالگوریتم FACT در مثال ۳۷ پیاده‌سازی شده است.

تمرین ۷

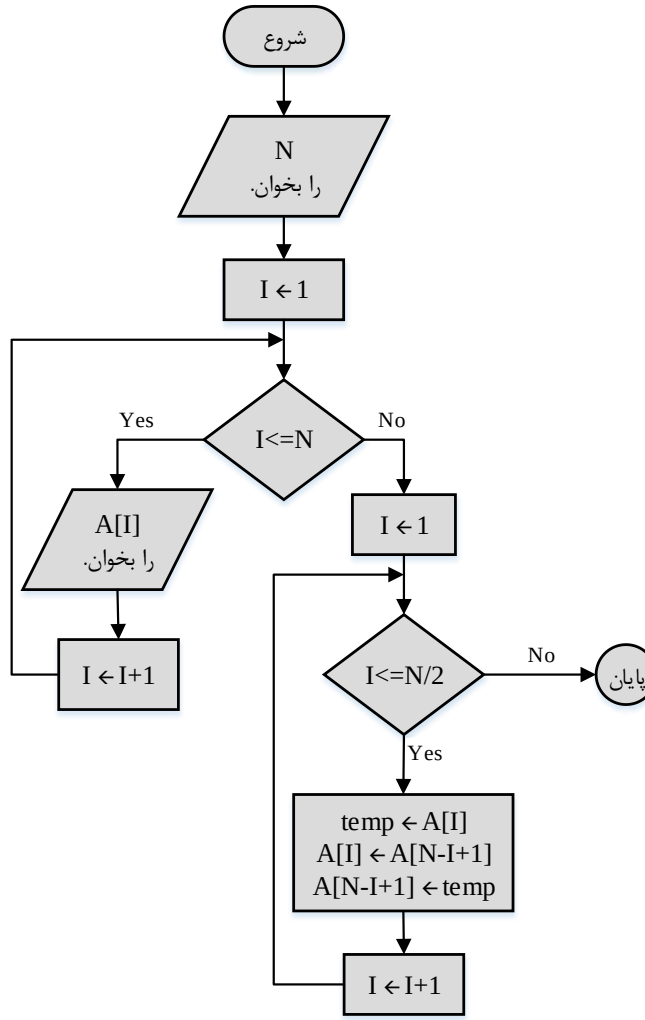
سوال ۵

فلوچارت برنامه‌ای رسم کنید که تعداد اعداد خوبِ کوچکتر از ۱۰۰۰ چاپ کند



تمرین ۸

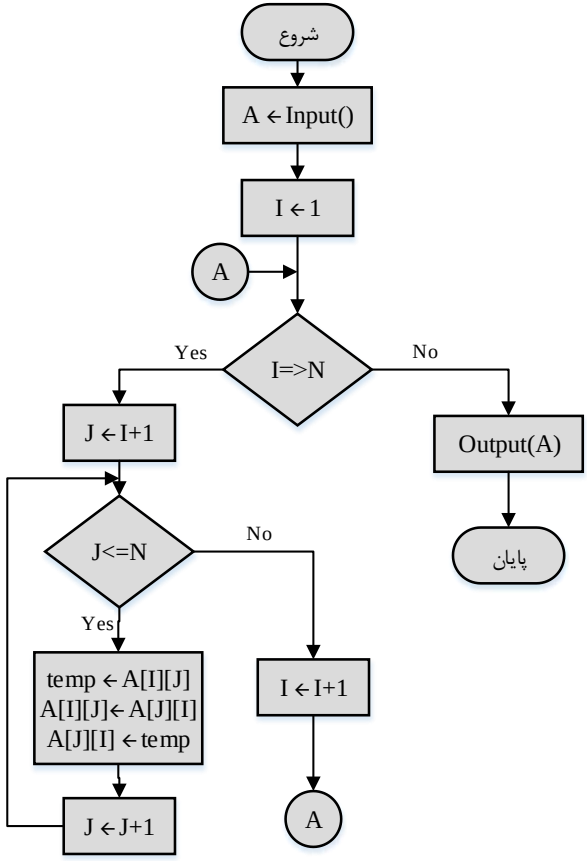
سوال ۲



تمرین ۹

سوال ۱

فلوچارت برنامه‌ای رسم کنید که یک ماتریس از ورودی دریافت کند و ترانهاده‌ی ماتریس را چاپ کند.

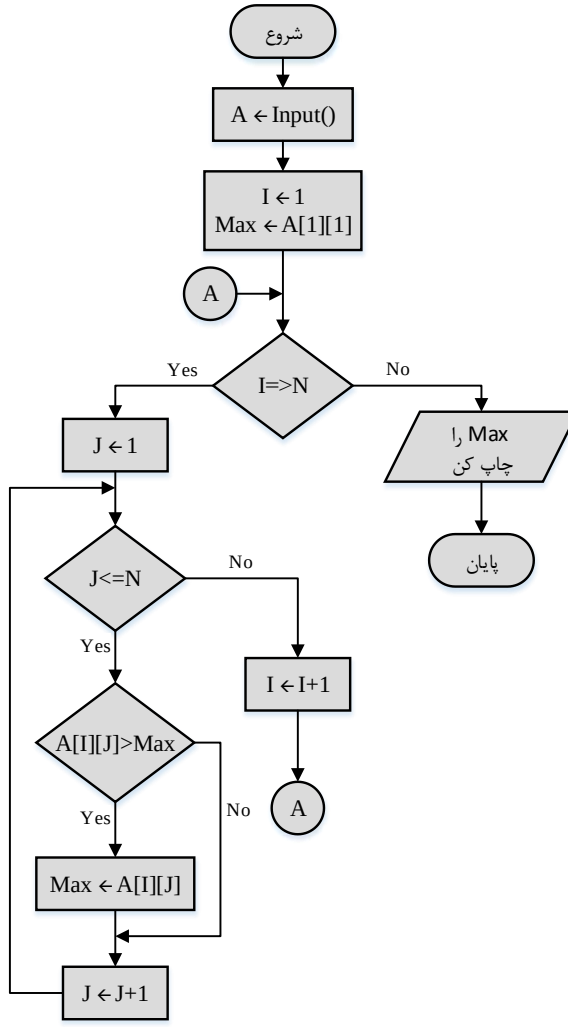


در مثال ۵۴ زیر الگوریتم Input و Output پیاده‌سازی شده است. همچنین در فلوچارت بالا به دستور J=I+۱ دقت کنید و نحوه پیمایش ماتریس را تحلیل کنید. (در این مثال پیمایش بالامثلثی صورت گرفته است.)

تمرین ۹

سوال ۳

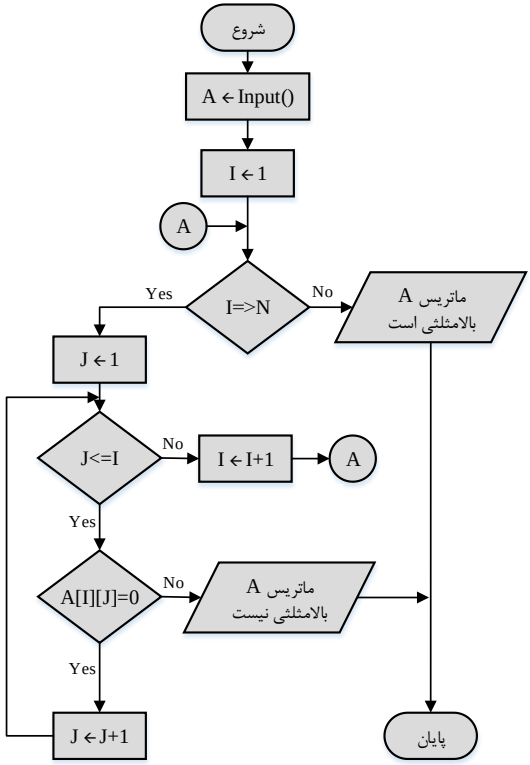
فلوچارت برنامه‌ای رسم کنید که یک ماتریس از ورودی دریافت کند و بزرگترین عنصر آن را چاپ کند.



تمرین ۹

سوال ۵

فلوچارت برنامه‌ای رسم کنید که یک ماتریس از ورودی دریافت کند و تشخیص دهد که ماتریس بالامثلثی است یا خیر.



توجه داشته باشید که در این مثال پیمایش پایین مثلثی صورت گرفته است.

موفق باشید