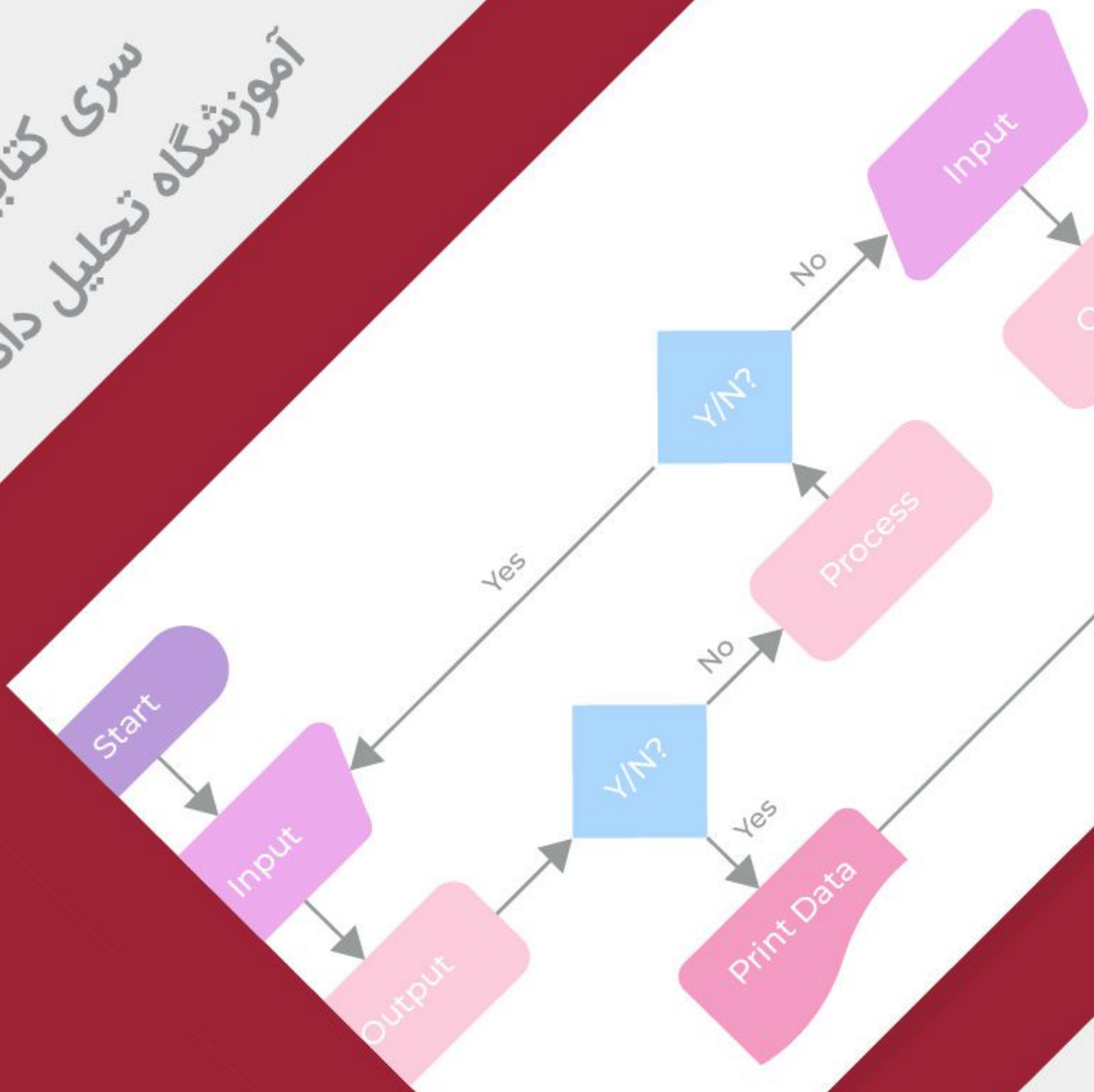
سری کتابهای آموزشی
آموزشگاه تحلیل داده



TAHLILDADEH EBOOK

آموزش
الگوریتم و فلو چارت

نویسنده : افشین رفوآ

www.tahlildadeh.com

کتابچه آموزش الگوریتم و فلوچارت

آموزشگاه تحلیل داده

فهرست محتوا

مقدمه

مهندسین نرم افزار از زبان‌های برنامه نویسی مختلفی استفاده می‌کنند تا بتوانند برنامه‌های متفاوتی را ایجاد بکنند. اما مهمتر از هر چیزی این نکته را نباید فراموش کرد که برنامه نویسان قبل از شروع به کدنویسی، نیاز دارند تا یکسری روال کاری را برای خودشان طراحی کنند تا در صورت برخورد با مشکلات بتوانند به صورت درست با آن برخورد نمایند. نداشتن این روال کاری و رویکرد از پیش آماده شده شانس اینکه نرم افزار با خطاهای بسیار زیادی مواجه شود را افزایش می‌دهد.

الگوریتم و فلوچارت دو ابزار و رویکرد قدرتمند در فرایند یادگیری برنامه نویسی و همچنین پیاده‌سازی روال کاری یک نرم افزار هستند. یک الگوریتم در واقع آنالیزهای یک کار را به صورت قدم به قدم بر عهده دارد، در حالیکه فلوچارت هر کدام از مراحل توسعه یک نرم افزار را به صورت گرافیکی نشان می‌دهد. این دو به عنوان مکمل‌های همدیگر شناخته می‌شوند و توانایی پیاده‌سازی هر دو این موارد در کنار همدیگر، نشان از قدرتمند بودن بیشتر نرم افزاری‌ست که در آینده قرار است درست شود.

برای افراد مبتدی و کسانی که قصد ایجاد یک نرم افزار را از اول دارند پیشنهاد می‌شود که در ابتدا فلوچارت و الگوریتم نرم افزار خود را طراحی کنند تا بدانند دقیقا قرار است چه چیزی را خلق کنند. در این حالت اگر به مشکلی نیز برخورد کردند می‌توانند به سرعت براساس نقشه راهی که فلوچارت و الگوریتم به آن‌ها می‌دهد، راهکار مناسب را پیدا کنند.

البته بسیاری از افراد مبتدی توانایی بالایی در طراحی و ساخت الگوریتم و فلوچارت ندارند و به همین دلیل این نقطه از مهندسی نرم افزار را چالش برانگیز می‌بینند. به همین دلیل بسیاری از اوقات شاهد آن بوده‌ایم که نرم افزاری بدون ایجاد فلوچارت و الگوریتم طراحی و ساخته شده است و در بیشتر مواقع نیز با شکست روبرو شده است.

این نکته را نیز در نظر بگیرید که الگوریتم یک رویکرد بسیار روشمند برای حل کردن مشکلات است. از این جهت افراد مختلف ممکن است در فرایند ساخت الگوریتم‌های‌شان از روش‌های متفاوتی استفاده کرده و لزوما الگوریتم و فلوچارت افراد مختلف مشابه همدیگر نخواهند بود.

در این آموزش از آموزشگاه تحلیل داده قصد داریم شما را با اصول اولیه الگوریتم‌ها و فلوچارت‌ها آشنا کنیم تا بتوانید درک محکم و اصولی از این دو موضوع را پیدا کنید. همچنین از آنجایی که مثال‌های متعددی در این نوشته وجود دارد شما می‌توانید به صورت عملی رویکرد پیاده‌سازی یک الگوریتم و فلوچارت را مشاهده بکنید.

الگوریتم

کلمه الگوریتم مربوط به دانشمند و ریاضی‌دان ایرانی «خوارزمی» است. این اسم نشان‌گر انجام یک کار با استفاده از یک پلن یا نقشه راه قدم به قدم است. دانشمندان و مهندسان علوم کامپیوتر از الگوریتم برای حل کردن خطاها، مشکلات و برنامه ریزی برای ایجاد یک نرم افزار استفاده می‌کنند.

یک الگوریتم شامل مجموعه‌ای از دستورات و اقدامات است که باید به صورت ترتیبی انجام شود تا بتوان به یک نتیجه نهایی دست پیدا کرد. مسلما این نتیجه نهایی چیزی خواهد بود که پیش از تعریف الگوریتم تعیین شده است.

هر الگوریتمی دارای چهار اصل و ویژگی پایه‌ای‌ست که در زیر این چهار مورد را به صورت لیست می‌توانید مطالعه کنید:

- ورودی: هر الگوریتم می‌تواند شامل ورودی باشد. البته این مورد همیشه درست نبوده و در برخی از حالت‌ها الگوریتم‌ها هیچ ورودی خاصی ندارند.
- خروجی: از هر الگوریتمی انتظار می‌رود که یک یا چند خروجی را برای ما تولید کند. الگوریتمی که نتواند خروجی تولید کند یک الگوریتم صحیح نبوده و باید مشکل آن را پیدا کرد.
- تعریف: مراحل و فرایندی که برای یک الگوریتم ایجاد می‌شود باید دارای تعریف واضح و روشنی باشد. در غیر اینصورت الگوریتمی که خود قرار است به شما کمک کند تا مشکلی را حل کنید، خود به یک معضل تبدیل می‌شود.
- پایان پذیر: در تعریف هر الگوریتم ممکن است یک دستور ثابت توسط شروط و حلقه‌ها چندین بار تکرار شود. یک الگوریتم نیاز دارد که نقطه پایانی داشته باشد و نباید در یک حلقه از تکرارهای پایان ناپذیر قرار بگیرد.

هم الگوریتم و هم فلوچارت دارای یکسری ساختارهای کنترلی هستند که برای کنترل کردن نوع انجام کارها مورد استفاده قرار می‌گیرد. در زیر این ساختارهای کنترلی را می‌توانید مطالعه کنید:

- ترتیب: ساختار کنترلی ترتیب به این موضوع اشاره می‌کند که یک الگوریتم از یک نقطه مبدا شروع شده و به صورت قدم به قدم به نقطه پایان می‌رسد. الگوریتم‌ها نمی‌توانند از وسط آغاز شوند و نمی‌توانند بدون پایان نیز باشند.
- شاخه‌ها: در برخی از الگوریتم‌ها و فلوچارت‌ها شما باید دو مسیر را تعیین کنید. در این حالت قبل از اینکه ساختار ترتیب وارد یکی از دو مسیر شود باید یک شرط بررسی شود. در صورت درست بودن شرط ساختار ترتیب براساس برنامه ریزی که شما انجام داده‌اید وارد یکی از مسیرها خواهد شد. به این ساختار کنترلی در الگوریتم و فلوچارت Branching گفته می‌شود.
- حلقه یا تکرار: ساختار کنترلی حلقه یا تکرار به شما این اجازه را می‌دهد تا یک دستور خاص در الگوریتم را بیش از یکبار اجرا کنید. این حلقه یا تکرار نیز می‌تواند باید براساس یک شرط عمل کند چرا که بدون شرط، حلقه تا بی نهایت ادامه خواهد داشت. همانطور که پیشتر اشاره کردم، الگوریتم باید پایان پذیر باشد.

مزایای استفاده از الگوریتم

- تبدیل کردن یک مشکل بزرگ به یکسری مشکلات کوچکتر که به صورت قدم به قدم می‌توان آن را حل کرد یک مزیت بزرگ در جهت درک مشکلات و حل کردن آن هاست که الگوریتم به شما می‌دهد.
- الگوریتم از یک رویکرد تعریف شده استفاده کرده و مشکلات بوجود آمده را می‌توان از پیشتر، پیش‌بینی کرد.
- وابستگی به یک زبان برنامه نویسی خاص نداشته و نسبت به اینکه با چه زبان و فریمورکی آن پیاده‌سازی کنید مستقل است.
- رفع عیب کردن و حل کردن مشکلات و خطاها در الگوریتم به دلیل ماهیت توضیحی بسیار ساده است.

چگونه یک الگوریتم را بنویسیم؟ (همراه با یک مثال ساده)

قدم اول: ورودی‌های الگوریتم خود را تعیین کنید. همانطور که گفته شد یک الگوریتم می‌تواند هیچ یا چند ورودی داشته باشد. برای مثال زمانی که شما قصد دارید مساحت یک مستطیل را حساب کنید ورودی‌های شما طول و عرض آن مستطیل خواهد بود.

قدم دوم: تعیین متغیرهای یک الگوریتم در مرحله دوم انجام می‌شود. زمانی که شما متغیر تعریف کنید می‌توانید مقادیر داده‌ای مختلفی را به کار ببرید. در همان مثال مساحت مستطیل، متغیرهای شما دو مورد خواهد بود که می‌توانید با نام‌های Height و Width آن‌ها را نامگذاری کنید. به عنوان یک نکته مهم این موضوع را همواره در نظر داشته باشید که برای تعیین نام

متغیرهای‌تان از اسامی با مفهوم استفاده کنید و از نام‌هایی مانند H و W خودداری کنید.

قدم سوم: عملیاتی که در الگوریتم باید انجام شود را تعریف کنید. برای مثال ضرب دو متغیر در نهایت می‌تواند به شما مساحت درست مستطیل را بدهد. در این حالت شما دو عملوند (متغیر) و یک عملگر خواهید داشت. عملگر این مثال ما ضرب خواهد بود. عملیاتی که شما در نظر دارید می‌تواند بسیار پیچیده و دارای Branch یا شاخه‌های مختلفی باشد و یا می‌تواند مانند این مثال یک عملیات ساده و سرراست باشد.

قدم چهارم: خروجی خود را تعیین کنید. پس از انجام عملیات شما با یک مقدار جدید روبرو هستید از حاصلضرب طول و عرض یک مستطیل است. اما باید این مقدار جدید را در یک متغیر جدید نیز تعریف کرده و به عنوان خروجی نهایی الگوریتم خود آن را معرفی کنید. این متغیر می‌تواند Area نامیده شود.

فلوچارت

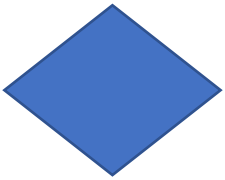
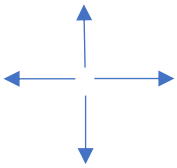
فلوچارت همان روش حل مسئله به صورت قدم به قدم است با این تفاوت که در این حالت شما به جای استفاده از متن خام، از یکسری اشکال استفاده می‌کنید. فلوچارت اولین بار در سال ۱۹۴۵ توسط John Von Neumann ایجاد شد و تلاشی بود برای اینکه با کمک گرفتن از نشانه‌های مختلف، یک روش بهتر برای توسعه راه حل‌ها ایجاد

شود. تنها با یک نگاه کلی به یک فلوچارت در بیشتر مواقع ممکن است شما از اینکه دقیقا چه کاری قرار است انجام شود یک ذهنیت را پیدا می‌کنید. در این حالت شما می‌توانید بسیار سریع‌تر از حالت الگوریتم، به یک مسئله پی ببرید.

مزایای بهره‌گیری از فلوچارت

- از آنجایی که فلوچارت از المان‌های گرافیکی استفاده می‌کند فرایند درک ساده‌تری داشته و بهتر می‌توان منطق برنامه نویسی را با آن ایجاد کرد.
- آنالیز و بررسی چیستی یک فلوچارت به مراتب سریعتر از یک الگوریتم انجام می‌شود.
- ایجاد تغییر و ویرایش کردن فلوچارت ساده‌تر بوده و رویکردی بصری دارد.
- تبدیل کردن یک فلوچارت به یک کد واقعی در یک زبان برنامه نویسی ساده‌تر از انجام همان کار با الگوریتم است.

همانطور که گفته شد فلوچارت براساس یکسری نماد به شما کمک می‌کند تا فرایند طراحی نرم‌افزارتان را انجام دهید. در ادامه ما با این نمادها آشنا خواهیم شد.

نام نماد	نماد	کارکرد
بیضی		بیضی نمایانگر شروع و پایان یک فلوچارت است.
متوازی الاضلاع		متوازی الاضلاع برای نمایش ورودی‌ها و خروجی‌ها استفاده می‌شود.
مستطیل		مستطیل برای نمایش پردازش‌ها و محاسبات نشان داده می‌شود. عملیات‌های ریاضی یکی از این محاسبات است.
الماس یا لوزی		لوزی برای تصمیم‌گیری و ایجاد برنچ‌ها و شاخه‌ها استفاده می‌شود. از این قسمت می‌توان براساس درست یا غلط بودن یک شرط، تصمیمات مختلف را شکل داد.
جهات		جریان قدم به قدم فلوچارت با استفاده از جهات نشان داده خواهد شد.

دایره		به جای خط های طولانی در یک فلوچارت قرار می گیرند. این نماد با نام On-page Connector نیز شناخته می شود.
Off Page Connector		مانند دایره است با این تفاوت که ادامه منطق را به صفحه دیگری منتقل می کند. به عبارتی دقیقا برخلاف On-page Connector است.

البته به غیر از موارد بالا اشکال دیگری نیز وجود دارند که البته استفاده ای زیاد نداشته و اشکالی که توضیح داده شد مهم ترین آن‌ها بودند.

نکته مهم: زبانی که برای نوشتن الگوریتم استفاده می‌شود دقیقا همان زبان روزمره انسان‌هاست و پیچیدگی گرامری یا املایی در آن وجود ندارد. با این حال از چند علامت ریاضیاتی استفاده می‌شود که کمی کاربردشان با حالت عادی متفاوت است.

متغیر: در دنیای برنامه نویسی و الگوریتم‌ها برای ذخیره یک مقدار داده‌ای ما آن را در یک متغیر ذخیره خواهیم کرد. برای این موضوع نیز ما نیاز داریم که روی متغیرمان اسم خاصی را بگذاریم.

مساوی: علامت مساوی در دنیای برنامه نویسی و الگوریتم به عنوان علامت Assignment یا نسبت دادن شناخته می‌شود. در واقع زمانی که شما

بخواهید مقداری را در داخل یک متغیر قرار دهید باید از علامت مساوی استفاده کنید.

جمع کردن: دستورالعمل C=A+B به معنی این است که مقدار متغیرهای A و B با همدیگر جمع شده و در نهایت داخل C قرار بگیرد.

علامت + در این حالت را یک عملگر تعریف می‌کنیم. در زیر می‌توانید لیستی از عملگرهای اصلی/ریاضیاتی دنیای برنامه نویسی و الگوریتم را مشاهده کنید.

عملگرهای ریاضی

عملگر	معنا	مثال
+	جمع	A + B
-	تفریق	A - B
*	ضرب	A * B
/	تقسیم	A / B
^	توان	A ^ B
٪	باقی مانده	A % B

عملگرهای رابطه‌ای

در دسته‌بندی عملگرهای رابطه‌ای، عملگرهایی وجود دارد که با استفاده از آن‌ها شما می‌توانید رابطه میان دو متغیر را با همدیگر بررسی کنید. نتیجه تمام این موارد در نهایت دو مقدار True یا False خواهد بود.

عملگر	معنا	مثال

< >	کوچکتر و بزرگ‌تر	B>A یا A<B
<=	کوچکتر/بزرگ‌تر یا مساوی	A<=B
==	بررسی مساوی بودن	A == B
!=	بررسی مساوی نبودن	A != B

عملگرهای منطقی

در دسته‌بندی عملگرهای منطقی، عملگرهایی وجود دارد که در زمان بررسی یک شرط کاربرد خواهند داشت. در این حالت شما زمانی که بخواهید چند شرط را به صورت همزمان بررسی کنید از عملگرهای منطقی بهره می‌گیرید.

عملگر	کارایی	مثال
AND	برای بررسی این که چند شرط به صورت همزمان درست باشند از AND استفاده می‌شود.	A < B AND B < C
OR	برای بررسی این که در بین چند شرط حداقل یک شرط درست باشد از OR استفاده می‌شود.	A< B OR B < C
NOT	هر چه که نتیجه یک شرط باشد با عملگر NOT نفی آن به خروجی ارسال می‌شود.	NOT (A >B)

دستورات کنترلی

از دستورات کنترلی در زمانی که بخواهیم یک شرط را بررسی کنیم استفاده می‌شود. برای مثال اگر باقی مانده یک عدد تقسیم بر دو برابر صفر باشد نتیجه می‌گیریم که این عدد یک مقدار زوج است، در غیر اینصورت عدد فرد خواهد بود.

در واقع دستورات کنترلی مجموعه‌ای از «اگر ها» و «در غیر اینصورت...» هایی‌ست که با همدیگر بخش بزرگی از منطق اصلی برنامه را ایجاد می‌کنند.

ساختار دستورات کنترلی به صورت زیر خواهند بود:

```
If condition:
    Do Something
else:
    Do Another Thing
```

منظور از عبارت Condition یک شرط است. در مثالی که بالا در ارتباط با اعداد زوج گفتیم، شرط ما عبارتی شبیه به number % 2 == 0 خواهد بود. تنها زمانی قسمت else اجرا خواهد شد که شرط اولیه درست نباشد.

اما اگر عدد ما برابر با صفر باشد چه؟ صفر نه زوج است و نه فرد! در این حالت می‌توان از حالت زیر استفاده کرد:

```
if condition:
```

```
        Do Something
else if condition:
        Do Something else
else:
        Do Another Thing
```

همانطور که مشاهده کردید با ترکیب else و if در کنار همدیگر می‌توانید یک شرط دیگر را نیز بررسی کنید.

دستورات حلقه یا تکرار

زمانی که شما بخواهید یک کار را برای چندین بار انجام دهید می‌توانید از یک دستور حلقه استفاده کنید. دستور حلقه شما می‌تواند در دو حالت اجرا شود. یا اینکه به صورت مشخص به حلقه گفته شود که چند بار اجرا شو! یا اینکه براساس یک شرط اینکار را انجام دهد.

در مثال شرطی برای مثال تصور کنید که قصد دارید تمام اعداد یک رقمی مثبت را چاپ کنید. در این حالت عملیات چاپ کردن را هر بار در حلقه انجام خواهید داد و در کنار آن به متغیرتان که مقدار اولیه صفر دارد یک عدد اضافه می‌کنید. اما شرط این قضیه تا زمانی خواهد بود که عدد شما به مقدار ۱۰ برسد. در واقع زمانی که عدد به ۱۰ رسید دیگر حلقه اجرا نمی‌شود چرا که شرط کوچکتر بودن از ۱۰ دیگر درست نخواهد بود.

WHILE (Condition):

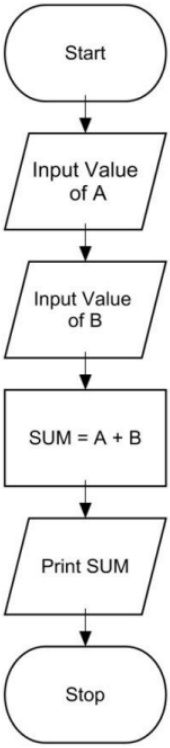
Do Somthing

در شبه کد بالا همانطور که مشاهده می‌کنید ساختار یک حلقه ساده نشان داده شده است.

حال با در نظر گرفتن این موضوعات ما می‌توانیم با همدیگر یکسری مثال و نمونه را حل کنیم تا ذهن الگوریتمی شما شکل گرفته و بتوانید به درستی این موارد را با حل مسئله متوجه شوید.

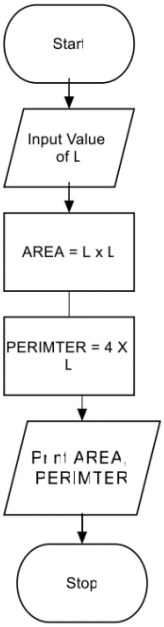
مسئله اول: الگوریتم و فلوچارتی بنویسید که دو عدد را با همدیگر جمع بکند.

حل مسئله: همانطور که از صورت مسئله پیداست شما باید حاصل جمع دو عدد را بدست بیاورید. به همین دلیل در ابتدای کار نیاز دارید تا این دو عدد را وارد الگوریتم خود بکنید. پس از انجام چنین کاری می‌توانید حاصل جمع آن‌ها را در یک متغیر با نام مثلا Sum ذخیره کرده و در نهایت آن را نمایش دهید.

الگوریتم	فلوچارت
1. شروع کن 2. اولین عدد را وارد کن A 3. دومین عدد را وارد کن B 4. Sum = A + B (متغیری با نام Sum ایجاد شده که حال جمع A و B در آن قرار گرفته است.) 5. Sum را چاپ کن (نشان بده) 6. پایان	

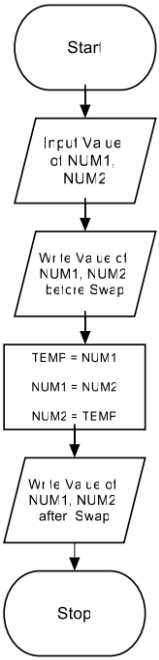
مسئله دوم: الگوریتمی بنویسید که با استفاده از آن مساحت و محیط یک مربع را بدست بیاورید.

حل مسئله: برای بدست آوردن محیط و مساحت یک مربع باید اندازه یک ضلع آن را بدانیم. همانطور که می‌دانید اندازه تمام اضلاع یک مربع برابر بوده و در نتیجه دانستن یک ضلع آن کافی خواهد بود. برای بدست آوردن مساحت یک مربع نیاز است تا اندازه یک ضلع را در خودش ضرب کنیم. برای بدست آوردن محیط نیز نیاز است تا اندازه یک ضلع مربع را در عدد ۴ ضرب کنیم. در نهایت باید اندازه‌های بدست آمده را در متغیرهای جداگانه‌ای ذخیره کرده و به کاربر نشان بدهیم.

الگوریتم	فلوچارت
1. شروع کن 2. اندازه یک ضلع مربع را وارد کن L 3. Area = L * L (اندازه یک ضلع L اندازه یک ضلع مربع و متغیر Area برابر با مساحت مربع خواهد بود) 4. Perimeter = L * 4 (محیط مربع) 5. Perimeter و Area را چاپ کن (نشان بده) 6. پایان	

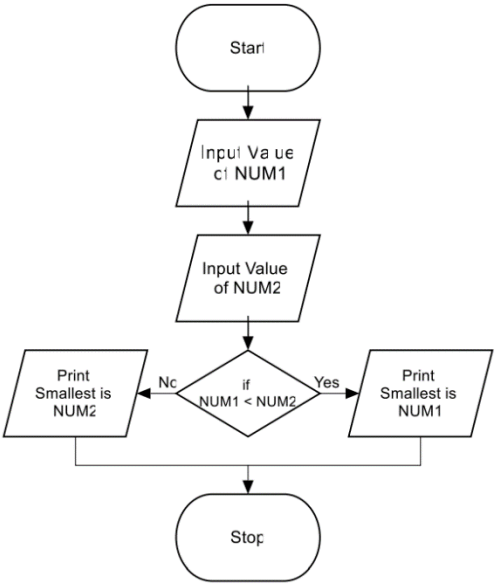
مسئله سوم: الگوریتمی بنویسید که با استفاده از آن جای دو متغیر با استفاده از یک متغیر سوم تغییر پیدا کند.

حل مسئله: زمانی که شما دو متغیر داشته باشید و بخواهید مقادیر داده‌ای آن‌ها را به همدیگر نسبت بدهید نیاز دارید که از یک متغیر سوم استفاده کنید. در این حالت شما مقدار متغیر اول را در متغیر سوم قرار داده و سپس مقدار متغیر دوم را در متغیر اول قرار می‌دهید و در نهایت مقدار متغیر سوم را به متغیر دوم نسبت خواهید داد. در این حالت مقدار اولیه متغیر اول در متغیر دوم و مقدار اولیه متغیر دوم در متغیر اول قرار می‌گیرد.

الگوریتم	فلوچارت
1. شروع کن 2. دو عدد را وارد کنید. Num1 و Num2 3. مقادیری را در دو متغیر Num1 و Num2 وارد کنید. 4. متغیری با نام Swap ایجاد کن و مقدار Num1 را در آن قرار بده: Swap = Num1 5. مقدار Num2 را در Num1 قرار بده: Num1 = Num2 6. مقدار Swap را در Num2 قرار بده: Num2 = Swap 7. مقادیر Num2 و Num1 را نشان بده 8. پایان	

مسئله چهارم: الگوریتمی بنویسید که دو عدد را دریافت کند و عدد کوچکتر را نشان بدهد.

حل مسئله: در این الگوریتم شما نیاز به استفاده از ساختارهای کنترلی و شرطی دارید. برای این کار در قدم اول دو عدد مورد نیاز را وارد کنید و سپس براساس یک شرط، کوچک یا بزرگ بودن یک عدد به نسبت عدد دیگر را بررسی کنید. برای مثال در صورتی که عدد اول از عدد دوم بزرگ‌تر بود، عدد دوم را چاپ کن، اگر عدد اول از عدد دوم بزرگ‌تر بود، عدد دوم را چاپ کن.

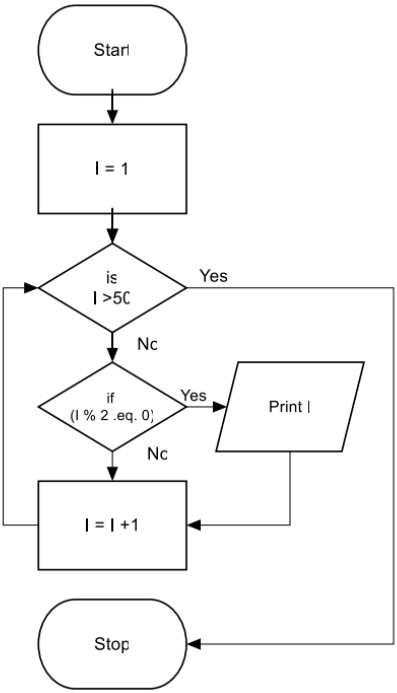
الگوریتم	فلوچارت
1. شروع کن 2. عدد اول را وارد کن: Num1 عدد دوم را وارد کن: Num2 4. اگر Num1 از Num2 کوچکتر بود، Num1 را چاپ کن. 5. در غیر اینصورت Num2 را چاپ کن. 6. پایان	

نکته: همانطور که در فلوچارت صفحه قبل مشاهده می‌کنید، با استفاده از شکل لوزی، ما جریان ساده فلوچارت را که پیشتر داشتیم تغییر داده و حال باید تصمیم‌گیری کنیم. برای اینکار با از دستور if که یک دستور شرطی است استفاده کرده‌ایم. همین مسئله را می‌توانیم برای نمایش عدد بزرگ‌تر نیز استفاده کنیم، کافی‌ست در منطق الگوریتم کمی دستکاری کنید.

مسئله ششم: الگوریتمی بنویسید که اعداد زوج ما بین ۱ تا ۵۰ را پیدا کند.

حل مسئله: در این مسئله ما نیاز به استفاده از یک ساختار حلقه‌ای نیاز داریم تا بتوانیم اعدادی که می‌خواهیم را تولید کنیم. برای این کار در ابتدا یک متغیر جدید درست کرده و مقدار ۱ را در آن قرار می‌دهیم. در مرحله بعدی براساس یک شرط، یکسری کارها را تکرار خواهیم کرد. شرط ما کوچک بودن مقدار متغیر i از عدد ۵۰ است. تا زمانی که این شرط درست باشد ما باقی‌مانده تقسیم متغیر بر ۲ را بررسی می‌کنیم. اگر صفر باشد i را به عنوان یک عدد زوج چاپ می‌کنیم و به i یک عدد اضافه می‌کنیم. اگر نباشد، تنها به سراغ مرحله اضافه کردن عدد خواهیم رفت.

نکته: همین الگوریتم را برای اعداد فرد نیز می‌توانید ایجاد کنید. تنها باید قسمت بررسی باقی‌مانده عدد بر ۲ برابر با یک باشد و نه صفر. در این صورت متغیر i اعداد فرد ما بین ۱ تا ۵۰ را در خود ذخیره خواهد کرد.

الگوریتم	فلوچارت
1. شروع کن 2. متغیری با نام i ایجاد کن و آن را برابر مقدار ۱ قرار بده. اول را وارد کن: Num1 3. تا زمانی که مقدار i از عدد ۵۰ کوچکتر است در یک حلقه کارهای مراحل بعدی را انجام بده، در غیر اینصورت به مرحله ۶ برو. 4. اگر باقی‌مانده تقسیم i بر ۲ برابر صفر است، i را به عنوان یک عدد زوج چاپ کن. 5. به i یک عدد اضافه کن و به مرحله ۳ بازگرد. 6. پایان	

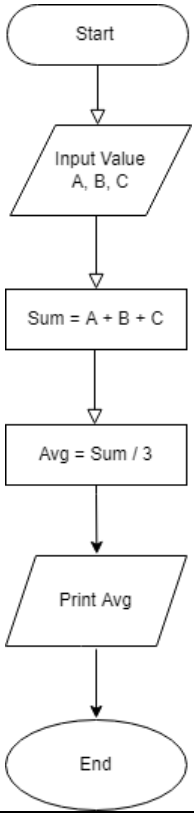
مسئله هفتم: الگوریتمی بنویسید که میانگین سه عدد دریافتی را پیدا کند

حل مسئله: برای حل این مسئله شما به سه عدد نیاز دارید که آن‌ها را در ابتدا باید وارد کنید. در مرحله بعدی نیاز است که سه عدد را جمع کرده و در یک متغیر جدید قرار دهید. در مرحله نهایی محاسبات نیز باید حاصل جمع بدست آمده را تقسیم بر تعداد اعداد که در این مثال ۳ است کرده و در نهایت آن را چاپ کنیم.

الگوریتم

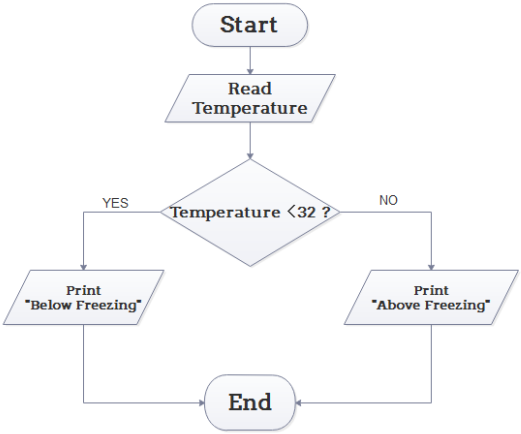
1. شروع کن
2. سه عدد اول را با نام‌های A و B و C وارد کنید.
3. حاصل جمع سه عدد را در متغیر Sum قرار بده.
4. حاصل تقسیم Sum بر عدد ۳ را در متغیر Avg قرار دهید.
5. Avg را چاپ کن.
6. پایان

فلوچارت



مسئله هشتم: الگوریتمی بنویسید که تشخیص دهد دما در حد انجماد است یا خیر.

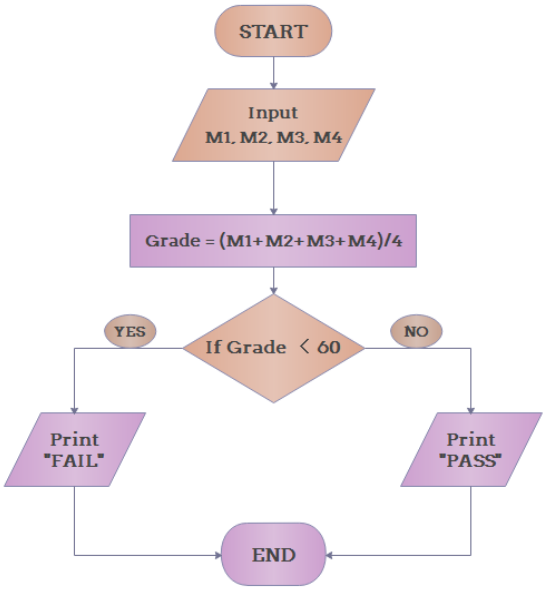
حل مسئله: برای حل این مسئله باید بدانید که دما انجماد صفر و زیر صفر است. در حل این مسئله ما از واحد فارنهایت استفاده کرده‌ایم. دما ۳۲ درجه فارنهایت برابر با صفر درجه سانتی‌گراد است. در نتیجه ما تنها یک شرط را بررسی می‌کنیم: اگر دما داده شده از ۳۲ درجه کوچکتر بود در نتیجه دما زیر صفر است و در غیر اینصورت بالای صفر.

الگوریتم	فلوچارت
1. شروع کن 2. مقدار دما را وارد کنید 3. آیا دما از ۳۲ درجه فارنهایت کمتر است؟ 4. بله: چاپ کن: دما در حد انجماد است 4. خیر: چاپ کن: دما در حد انجماد نیست. 5. پایان	

نکته: ۳۲ فارنهایت برابر همان ۰ درجه سانتی گراد است.

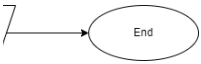
مسئله نهم: الگوریتمی بنویسید که براساس چند نمره یک دانش آموز، قبول شدن یا رد شدن وی را مشخص کند.

حل مسئله: برای حل این مسئله نیاز است تا ۴ نمره دانش آموز را دریافت کرده و حاصل جمع آن‌ها را بر عدد ۴ تقسیم کنیم و در نهایت معدل را بدست بیاوریم. اگر معدل از ۶۰ (معادل ۱۲ در سیستم آموزشی ایران) کمتر بود، رد شدن دانش آموز چاپ خواهد شد و در غیر اینصورت قبول شدن وی.

الگوریتم	فلوچارت
1. شروع کن 2. ۴ نمره را وارد کنید. M1, M2, M3, M4 3. معدل نهایی را با جمع کردن چهار عدد و تقسیم‌شان بر ۴ بدست بیاورید. Grade 4. حاصل Grade را نمره ۶۰ مقایسه کنید. 6. بزرگتر: چاپ کن: دانش آموزش قبول شده است. 6. کوچکتر: چاپ کن: دانش آموزش رد شده است. 7. پایان	

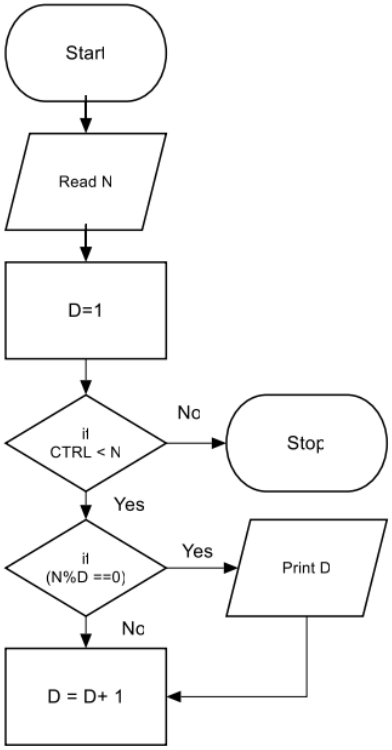
مسئله دهم: الگوریتمی بنویسید که حاصل جمع اعداد ۱ تا ۵۰ را محاسبه کند.

حل مسئله: برای حل این مسئله شما به دو متغیر نیاز دارید. یک متغیر در یک حلقه وظیفه دارد تا اعداد ۱ تا ۵۰ را در خود نگهداری کند و در همین حال متغیر بعدی جمع مقادیری که در متغیر اول قرار می‌گیرد را در خود ذخیره می‌کند. برای این کار هر دو متغیر در ابتدا مقدار صفر را در خود ذخیره خواهند کرد. تا زمانی که متغیر اول به مقدار ۵۰ نرسد هر بار یک عدد به آن اضافه شده و با مقدار Sum که متغیر دوم است جمع می‌شود. زمانی که متغیر اول به عدد ۵۰ رسید نتیجه نهایی چاپ شده و الگوریتم تمام خواهد شد.

الگوریتم	فلوچارت
1. شروع کن 2. دو متغیر N و Sum را تعریف کن و مقدار صفر را برای آنان در نظر بگیر. 3. به N یک عدد اضافه کن. 4. متغیر Sum را برابر حاصل جمع Sum + N قرار بده. 7. آیا مقدار N برابر با ۵۰ است؟ 8. خیر: به مرحله ۳ برگرد. 8. بله: متغیر Sum را چاپ کن. 9. پایان	

مسئله یازدهم: الگوریتمی بنویسید که تمام اعداد بخش پذیر یک عدد را پیدا کند.

حل مسئله: برای اینکه مشخص شود یک عدد بر عدد دیگری بخش پذیر است باید باقی مانده تقسیم آن‌ها برابر با صفر باشد. در نتیجه در ابتدا عدد مورد نظر را وارد متغیری به نام n کنید. سپس یک متغیر که اعداد ۱ تا n را تولید می‌کند در نظر بگیرید. تا زمانی که n از متغیر دوم بزرگ‌تر است عملیات تقسیم و چاپ اعداد بخش پذیر ادامه پیدا خواهد کرد. اگر شرط بزرگ‌تر بودن درست نباشد برنامه در نهایت متوقف خواهد شد.

الگوریتم	فلوچارت
1. شروع کن. 2. عدد را در متغیر N قرار بده. 3. یک متغیر جدید برابر ۱ قرار دهید. D 4. تا زمانی که D از N کوچکتر است به مرحله پنج برو، در غیر اینصورت به مرحله ۶ برو. 5. اگر باقی‌مانده N بر D برابر صفر است، D را چاپ کن و یک عدد به D اضافه کن. در غیر اینصورت تنها یک عدد به D اضافه کن و به مرحله ۴ بازگرد. 6. پایان برنامه.	

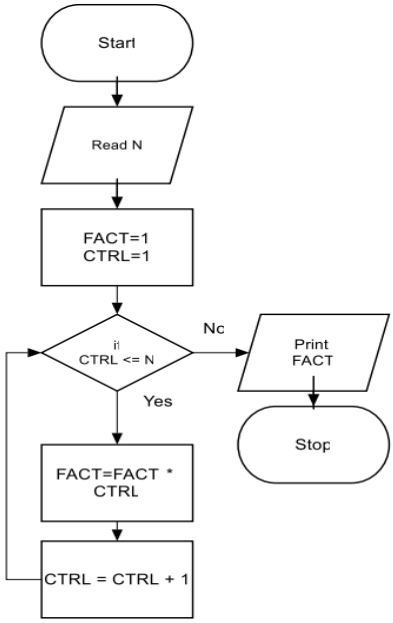
مسئله دوازدهم: الگوریتمی بنویسید که فاکتوریل یک عدد را بدست بیاورد.

حل مسئله: برای حل این مسئله باید با شیوه بدست آوردن فاکتوریل یک عدد آشنا باشید. در این حالت مشاهده خواهید کرد که برای چنین کاری به یک عدد در ابتدا نیاز دارید که بخواهید فاکتوریل آن را بدست بیاورید. برای اینکار ما مقدار N را در نظر گرفته‌ایم. در مرحله بعدی یک متغیر برای ذخیره مقدار فاکتوریل N در نظر گرفته‌ایم که برابر با Fact است و در مرحله بعدی متغیری دیگر که مقادیر کوچک‌تر از N را در خود نگه می‌دارد که CTRL نام دارد.

الگوریتم

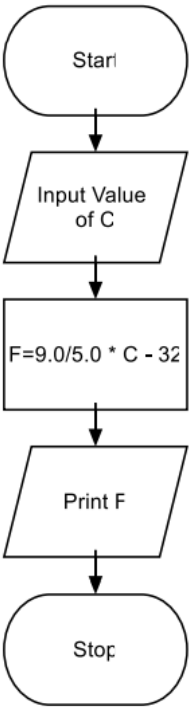
1. شروع کن.
2. عدد را در متغیر N قرار بده.
3. دو متغیر با نام Fact و CTRL را تعریف کنید و مقدار ۱ را در آن‌ها قرار دهید.
4. تا زمانی که CTRL از N کوچکتر یا مساوی آن است کارهای مرحله پنجم را انجام دهید. در غیر اینصورت به مرحله ۷ برو.
5. Fact را برابر Fact * CTRL کنید.
6. CTRL = CTRL + 1 به قدم ۴ برگرد.
7. Fact را چاپ کن.
8. پایان.

فلوچارت



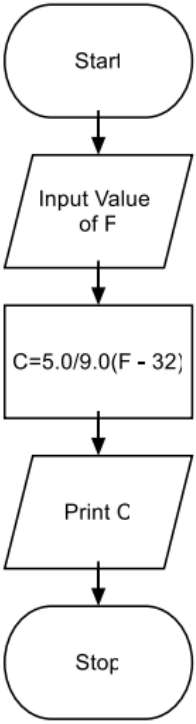
مسئله سیزدهم: الگوریتمی بنویسید که سانتی گراد را به فارنهایت تبدیل بکند.

حل مسئله: شما برای حل این مسئله باید فرمول لازم برای تبدیل سانتی‌گراد به فارنهایت را بلد باشید. در جدول زیر می‌توانید این فرمول را مشاهده کرده و به سادگی سانتی گراد را به فارنهایت تبدیل بکنید.

الگوریتم	فلوچارت
1. شروع کن. 2. دما سانتی گراد را وارد کن: C 3. فرمول F = (9.0/5.0 x C) + 32 را اجرا کن. 4. مقدار F را چاپ کن. 5. پایان.	

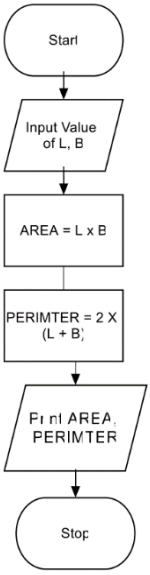
مسئله چهاردهم: الگوریتمی بنویسید که فارنهایت را به سانتی گراد تبدیل بکند.

حل مسئله: شما برای حل این مسئله باید فرمول لازم برای تبدیل فارنهایت به سانتی گراد را بلد باشید. در جدول زیر می‌توانید این فرمول را مشاهده کرده و به سادگی فارنهایت را به سانتی گراد تبدیل بکنید.

الگوریتم	فلوچارت
1. شروع کن. 2. دما فارنهایت را وارد کن: F 3. فرمول (C = 5.0/9.0 (F - 32 را اجرا کن. 4. مقدار C را چاپ کن. 5. پایان.	

مسئله پانزدهم: الگوریتمی بنویسید که با استفاده از آن مساحت و محیط یک مستطیل را بدست بیاورید.

حل مسئله: برای بدست آوردن مساحت و محیط یک مستطیل نیاز است که اندازه ضلع‌های طول و عرض آن را داشته باشیم. در صورتی که این دو مورد را داشته باشیم می‌توانیم با ضرب این دو مورد مساحت و با جمع دو مورد ضربدر ۲ محیط آن را بدست بیاوریم.

الگوریتم	فلوچارت
1. شروع کن 2. اندازه طول مستطیل را L و عرض آن را B نامگذاری می‌کنیم. 3. Area = L * B (اندازه طول L اندازه مستطیل و B اندازه عرض آن و متغیر Area برابر با مساحت مستطیل خواهد بود) 4. Perimeter = 2 (L + B) (محیط مستطیل) 5. Perimeter و Area را چاپ کن (نشان بده) 6. پایان	

در پایان

یکی از مهمترین موضوعاتی که مهندسین نرم افزار باید آن را در نظر داشته باشند یادگیری الگوریتم و فلوچارت است. در این کتابچه کوتاه ما سعی کردیم شما را به خوبی با این دو موضوع همراه با مثال‌های عملی آشنا کنیم.