مبانی کامپیوتر و برنامه‌نویسی

(جزوه درسی)

رشته مهندسی کامپیوتر

کمال محمّدی‌اصل

پیشگفتار

با توجه به لزوم آموزش مفاهیم اولیه قبل از شروع برنامه‌نویسی، بر آن شدیم تا با ارائه جزوه پیش رو ابتدا به بیان مقدماتی از مبانی کامپیوتر و اصول حل مسئله با استفاده از فلوچارت بپردازیم تا دانشجویان گرانقدر قبل از ورود به مبحث شیرین برنامه‌نویسی، با این مقدمات و اصول اولیه آشنا شوند.

در این جزوه که در سه فصل، نگاشته شده است، ابتدا به بیان تاریخچه‌ای کوتاه از کامپیوتر پرداخته شده و در فصل دوم، مفاهیم تبدیلات مبناهای عددی به صورتی خلاصه و قابل استفاده در درس مبانی کامپیوتر توضیح داده شده است. در فصل سوم، حل مسئله با استفاده از الگوریتم و فلوچارت مورد بررسی قرار گرفته است.

لازم به ذکر است جزوه پیش‌رو به صورت آزمایشی در نیمسال جاری به صورت ضمیمه قبل از کتاب درسی معرفی شده ارائه می‌گردد و نسخه بازبینی شده کتاب که حاوی مطالبی از این جزوه و ویرایشی از کتاب موجود برای درس برنامه نویسی است در نیمسال بعد به حضور همکاران و دانشجویان محترم عرضه خواهد شد.

امید است این تلاش مورد قبول همه عزیزان قرار گیرد و همه عزیزان با ارسال پیشنهادات و انتقادات، ما را در بهبود نسخه جدید کتاب «مبانی کامپیوتر و برنامه‌سازی» یاری بفرمایند.

کمال محمدی‌اصل

mohammaddiasl@pnu.ac.ir

فصل اول

اصول و مبانی کامپیوتر

هدف کلی

آشنایی با مفاهیم پایه‌ای سیستم‌های کامپیوتری و اجزاء تشکیل دهنده آنها

هدف‌های رفتاری

انتظار می‌رود پس از مطالعه این فصل بتوانید:

ـ تاریخچه پیدایش کامپیوترهای امروزی را شرح دهید.

ـ نسل‌های کامپیوترهای امروزی را نام برده و مشخصه‌های اصلی هر نسل را بیان نمایید.

ـ دسته‌بندی‌های مختلف کامپیوترها را برشمارید و ویژگی‌های هر دسته را عنوان کنید.

ـ اجزاء تشکیل‌دهنده سازمان یک کامپیوتر امروزی را شناسایی و بیان نمایید.

ـ با انواع سخت‌افزارها آشنا شده و بتواند آنها را دسته‌بندی نمایید.

ـ با مفهوم حافظه و واحدهای اندازه‌گیری آن آشنا شوید.

ـ با مفهوم نرم‌افزار و انواع آن آشنا شده و انواع نرم‌افزارها را شناسایی نمایید.

ـ با سیستم عامل و وظایف آن آشنا شوید.

ـ با انواع زبانهای برنامه‌سازی و سطوح آنها آشنا شده و مفاهیم کامپایلر و مفسر را بشناسید.

مقدمه

فهم درست اصول و مفاهیم اولیه یک سیستم کامپیوتری، در یادگیری مبانی برنامه‌نویسی تأثیری بسیار مهم خواهد داشت. این فصل به بیان تاریخچه مختصری از کامپیوتر می‌پردازد و سپس دسته‌بندی‌های کامپیوترهای امروزی را برمی‌شمرد. در ادامه به بررسی اجزاء یک سیستم کامپیوتری می‌پردازیم و آن را از جنبه های سخت‌افزاری و نرم‌افزاری مورد بررسی قرار می‌دهیم. انواع زبان‌های برنامه‌نویسی و مراحل ایجاد و توسعه یک سیستم نرم افزاری نیز به صورتی خلاصه مورد بحث واقع خواهد شد.

۱-۱ تاریخچه پیدایش کامپیوتر

واژه «*کامپیوتر*[۱]» به معنی محاسبه‌گر یا محاسبه‌کننده می‌باشد. از زمان‌های بسیار دور، انسانها به دنبال راه‌های مختلفی جهت تسهیل محاسبات خود بوده‌اند. استفاده از انگشتان دست در محاسبات، خط نشان‌ها و اختراع *چرتکه*[۲] نشان از این تلاش دارد. بسیاری از محقّقان، تاریخچه پیدایش اولین ابزارهای محاسباتی را حدود سال‌های ۵۰۰ الی ۳۰۰ قبل از میلاد مسیح می‌دانند. شاید بتوان چرتکه را اولین ابزار محاسباتی دانست که به دست انسان ساخته شد. در سال ۱۶۴۲، *بلیز پاسکال*[۳] یک ماشین حساب مکانیکی ساخت که قادر بود اعمال جمع و تفریق را انجام دهد. در سال ۱۸۳۴، *چارلز بابیج*[۴] یک ماشین تحلیلی را پیشنهاد نمود که در زمان خودش ساخته نشد. در سال ۱۸۴۳، *آدا بایرون*[۵] برنامه ای برای ماشین تحلیلی چارلز بابیج نوشت و نام اولین برنامه نویس کامپیوتر را به خود اختصاص داد. *جان وی. آتاناسوف*[۶] در سال ۱۹۳۹ نمونه اولیه‌ای از یک کامپیوتر دیجیتال الکترونیکی را توسعه داد. سال‌ها بعد، به پاس زحمات این افراد، زبانهای برنامه نویسی پاسکال و ادا نیز توسعه یافتند.

۱-۲ نسل‌های کامپیوترهای امروزی

کامپیوترهای امروزی در طول دوره تکامل خود، نسل‌های مختلفی را پشت سر گذاشته‌اند. از کامپیوترهای الکترونیکی اولیه تا تبلت‌ها و ربات‌های انسان‌نمای امروزی، این صنعت شاهد تحولات بسیار زیادی بوده است. صنعت کامپیوتر، این پیشرفت را مرهون توسعه صنعت الکترونیک بوده است. در ادامه، نسل‌های کامپیوترهای امروزی را از اولین نسل کامپیوترهای الکترونیکی مورد بررسی قرار می‌دهیم.

---

[۱] Computer
[۲] Abacus
[۳] Blaise Pascal
[۴] Charles Babbage
[۵] Ada Byron
[۶] John v. Atanasoff

نسل اول (۱۹۵۹-۱۹۴۵)

در نیمه اول قرن بیستم، *لامپ‌های خلأ*[۱] غالبترین تکنولوژی ساخت لوازم الکترونیکی بودند و اغلب لوازم الکترونیکی مانند رادیو، تلویزیون و غیره در طول این سالها با استفاده از لامپ‌های خلأ ساخته می شدند. بدین ترتیب نسل اول کامپیوترها با ساخته شدن اولین کامپیوتر الکترونیکی همه‌منظوره در سال ۱۹۴۶ در دانشگاه پنسیلوانیای آمریکا، شکل گرفت. این کامپیوتر ENIAC[۲] نام داشت و توسط *جی. پرسپر اکرت*[۳] و *جان دبلیو. موچلی*[۴] ساخته شد. انیاک ۱۸۰۰۰ لامپ خلأ، ۷۰۰۰۰ مقاومت، ۱۰۰۰۰ خازن، ۶۰۰۰ سوئیچ و ۱۵۰۰ رله داشت و ۱۶۲ مترمربع فضا اشغال می‌نمود. انیاک ۳۰ تن وزن داشت و ۱۶۹ کیلووات توان نیاز داشت و ارزش آن بیش از ۴۸۶۰۰۰ دلار بود. انیاک قادر بود ۵۰۰۰ عمل را در ثانیه انجام دهد که برای آن زمان قدرت محاسباتی فوق العاده‌ای محسوب می‌شد. انیاک هر عمل جمع را در ۰/۲ میلی ثانیه، هر عمل ضرب را در ۲/۸ میلی‌ثانیه و هر عمل تقسیم را در ۲۴ میلی‌ثانیه انجام می‌داد.

نسل دوم (۱۹۶۴-۱۹۶۰)

اختراع ترانزیستور توسط *جان باردین*[۵]، *والتر اچ. براتین*[۶] و *ویلیام ب. شاکلی*[۷] در *آزمایشگاه‌های بل*[۸] در سال ۱۹۴۷ باعث شد تا تحولی دیگر در صنعت الکترونیک به وجود آید. این اختراع باعث شد تا لامپ‌های خلأ بکار رفته در ابزارهای الکترونیکی با ترانزیستور جایگزین شوند. این امر باعث شد تا در عین حال که حجم و توان مصرفی این ابزارها کاهش می‌یافت، سرعت و قابلیت‌های پردازشی آنان نیز به مراتب افزایش یابد. یکی از اولین ماشین‌های محاسباتی که بر پایه ترانزیستور در این نسل ساخته شد، IBM 7090 نام داشت و توسط شرکت IBM ارائه گردید. علاوه بر مسائل مربوط به حجم و قدرت پردازش، قیمت دستگاه‌های محاسباتی نیز رو به کاهش رفت و این امر باعث افزایش عمومیّت آنها گردید. در این نسل، از زبان‌های کوبول و فرترن برای برنامه‌نویسی انواع سیستم‌های تجاری و علمی استفاده گردید. همچنین دیسک‌ها و نوارهای مغناطیسی برای ذخیره داده‌ها بکار گرفته شدند.

نسل سوم (۱۹۷۰-۱۹۶۴)

---

[۱] Vacuum Tubes
[۲] Electronic Numerical Integrator And Computer
[۳] J. Presper Eckert
[۴] John W. Mauchly
[۵] John Bardeen
[۶] William B. Shockley
[۷] Walter H. Brattain
[۸] Bell Laboratories

جک *کیلبی*[۱] و *رابرت نویس*[۲] در سال ۱۹۵۹ اولین مدار مجتمع جهان را ساختند. با مجتمع‌سازی ترانزیستورها در داخل *مدارات مجتمع*[۳]، حجم مدارات مبتنی بر ترانزیستورها به طور چشمگیری کاهش یافت، بنحوی که کامپیوترها توانستند موقعیت بی‌چون و چرای خود را در اکثر سازمانها به عنوان اجزاء ضروری تثبیت نمایند. ICها بسته به تکنولوژی ساخت می‌توانستند ده‌ها، صدها و حتی هزاران ترانزیستور، مقاومت و اجزای الکترونیکی دیگر را در خود جای دهند و این نوید خوبی برای توسعه هرچه بیشتر کامپیوترها در عرصه‌های مختلف بود. در نسل سوم، عمومیت و محبوبیت کامپیوترها بیش از پیش افزایش یافت و تقریباً اکثر مراکز علمی، بازرگانی و نظامی از کامپیوترها بهره می‌جستند. تاریخ پیدایش اولین شبکه‌های ارتباطی و حتی اینترنت (البته نه به معنای امروزی آن) به همین نسل برمی‌گردد.

نسل چهارم (تاکنون ـ ۱۹۷۰)

نسل چهارم در ادامه پیشرفتهای حاصله در تکنولوژی ساخت IC متولد شد. پس از ایجاد و توسعه ICها در نسل سوم، و با توسعه صنعت میکروالکترونیک و امروزه نانوالکترونیک میزان مجتمع سازی از ده‌ها، صدها و هزاران جزء الکترونیکی به میلیونها و ده‌ها میلیون ترانزیستور و سایر اجزاء الکترونیکی در یک چیپ[۴] رسید. تکنولوژی مورد استفاده در این نسل را اصطلاحاً VLSI و UVLSI می‌نامند که متناظر با *مجتمع سازی در مقیاس بسیار زیاد*[۵] و *مافوق بسیار زیاد*[۶] می‌باشد. این پیشرفت متناظر با کوچک‌تر شدن هر چه بیشتر کامپیوترها و سرازیر شدن کامپیوترها به خانه‌ها و میزهای کاری کارکنان ادارات، بانکها و شرکتها بود. کامپیوترهای قابل حمل[۷]، کامپیوترهای جیبی و تبلت‌ها همگی از محصولات این نسل هستند.

و اما در حال حاضر

در حال حاضر کامپیوترهای امروزی به آن مرحله از تکنولوژی دست یافته‌اند که دیگر کوچک سازی از نظر مقیاس، دغدغه اصلی محققین نیست. بلکه الان دغدغه های دیگری ذهن محققین را به خود مشغول نموده است. از جمله این دغدغه‌ها، می‌توان به قادر ساختن کامپیوترها به داشتن احساسات، ادراک و قدرت استنباط اشاره نمود.

۱-۲-۱ تاریخچه مختصری از کامپیوتر در ایران

[۱] Jack Kilby
[۲] Robert Noyce
[۳] Integrated Circuits (ICs)
[۴] Chip
[۵] Very Large-Scale Integration
[۶] Ultra Very Large-Scale Integration
[۷] Laptop Computers

اولین کامپیوتر حدود ۱۰ سال بعد از پیدایش اولین کامپیوترهای تجاری در جهان، و ۲۲ سال بعد از اختراع حدود سال ۱۳۴۱ شمسی وارد ایران گردید. بانک ملی و شرکت نفت نخستین نهادهایی بودند که کار با رایانه را شروع کردند. پس از آن در سال ۱۳۴۳ دانشگاه تهران نیز به جمع استفاده‌کنندگان از رایانه پیوست. بعد از پیروزی انقلاب شکوهمند اسلامی و در دهه ۶۰ شرکت‌های کامپیوتری زیادی ایجاد و شروع به فعالیت نمودند. در اواخر دهه ۷۰ و اوایل دهه ۸۰ نیز اینترنت در کشور به صورت عمومی مورد استفاده قرار گرفت.

۱-۳ دسته‌بندی کامپیوترها

کامپیوترها را می‌توان بر اساس قدرت پردازش و حجم محاسبات قابل انجام آنها به گروه‌های مختلفی تقسیم‌بندی نمود. ابرکامپیوترها، کامپیوترهای بزرگ، کامپیوترهای کوچک، ریزکامپیوترها، تبلت‌ها و غیره همه و همه نمونه هایی از انواع مختلف کامپیوترهایی هستند که امروزه در جهان موجودند.

ابرکامپیوترها[۱] کامپیوترهایی با قدرت پردازش بسیار بالا، حجم ذخیره‌سازی داده‌های بسیار زیاد و برای کاربردهای خاص می‌باشند. هر ساله لیستی از ۵۰۰ ابرکامپیوتر برتر جهان توسط مؤسسه بین‌المللی تاپ۵۰۰[۲] منتشر می‌شود. در چند سال اخیر، ابرکامپیوترهای چینی در صدر این جدول قرار گرفته‌اند. ابرکامپیوترهایی از آمریکا، آلمان، فرانسه و ژاپن نیز در این لیست دیده می شوند. ایران نیز در سال های اخیر، تلاش‌های بسیاری در جهت ساخت ابرکامپیوترهای ملی انجام داده که از آن جمله می‌توان به دو پروژه بزرگ که در دانشگاه صنعتی اصفهان و دانشگاه صنعتی امیرکبیر اجرا گردیده و در حال انجام است اشاره نمود. از ابرکامپیوترها در کاربردهای نظامی، فضایی و هواشناسی استفاده می‌کنند.

کامپیوترهای بزرگ[۳]، کامپیوترهایی هستند که قدرت پردازشی آنها بسیار بالاست ولی نسبت به ابرکامپیوترها توان محاسباتی پایین‌تری را دارا می‌باشند. این دسته از کامپیوترها نیز فضای زیادی را اشغال می‌کنند و قادرند محاسبات بسیار زیادی را در زمانی اندک انجام دهند. این دسته قیمت پایین تری نسبت به ابرکامپیوترها دارند.

کامپیوترهای کوچک[۴] در مقایسه با کامپیوترهای بزرگ، قیمت کمتری دارند و به همین نسبت نیز از توان محاسباتی و قدرت پردازشی پایین‌تری برخوردارند. این نوع کامپیوترها فضای کمتری اشغال می کنند و در کاربردهای تجاری و بازرگانی کاربرد دارند. با اینحال قدرت آنها از ریزکامپیوترها بالاتر است.

---

[۱] Supercomputers
[۲] Top500
[۳] Mainframes
[۴] MiniComputers

ریزکامپیوترها[۱] کوچکترین نوع کامپیوترها هستند. کامپیوترهای خانگی، شخصی، قابل حمل، تبلت ها و غیره جزو این دسته از کامپیوترها محسوب می‌شوند. امروزه ریزکامپیوترها را در هر جایی می‌توان دید. در مراکز آموزشی، ادارات، بانک ها، پایانه های مسافربری، فرودگاه‌ها و هر جای دیگری که تصور آن را بکنید. شاید مهمترین ویژگی این دسته از کامپیوترها نیز همین باشد: دسترس پذیری و در دسترس بودن. این گروه از کامپیوترها قدرت پردازش محدودی دارند گرچه همین قدرت پردازش آنها نیز هر روز رو به افزایش است.

۱-۴ ساختار یک سیستم کامپیوتری

هر سیستم کامپیوتری از دو بخش عمده تشکیل شده است:. نرم افزار[۲] و سخت افزار[۳]. سخت افزار را می‌توان چنین تعریف نمود:

«کالبد فیزیکی کامپیوتر را سخت افزار می‌گویند.»

«هرآنچه از یک کامپیوتر که قابل لمس و قابل مشاهده باشد، سخت افزار می‌گویند.»

به طور کلی، همه قسمتهای فیزیکی و قابل لمس یک کامپیوتر، از مدارات الکترونیکی و مانیتور و صفحه کلید و ماوس را سخت‌افزار می‌گوییم. در مقابل مفهوم نرم‌افزار را داریم. نرم‌افزار یکی از بخش‌های اساسی کامپیوتر به شمار می‌آید، که در واقع سخت‌افزار را بکار می‌گیرد. بعبارت دیگر رابط بین کاربر و سخت‌افزار را نرم‌افزار می‌نامند. نرم‌افزار در حقیقت روح و جان یک کامپیوتر است، که به سخت‌افزار هویت می‌بخشد. نرم‌افزارها انواع مختلفی دارند، که مشهورترین آنها نرم‌افزارهای سیستمی و کاربردی را می‌توان نام برد.

در تعریف نرم‌افزار، چنین می‌توان گفت:

«نرم‌افزار، به تمام برنامه‌ها و داده‌هایی که از سخت‌افزار بهره برده و با کمک سخت‌افزار، قابلیت اجرایی می‌یابند و می‌توانند قابلیتهای سخت‌افزار را به کارگیرند، اطلاق می‌شود.»

بدیهی است که هیچ کدام از این دو بدون همدیگر کامل نیستند و نمی‌توانند اهداف یک کاربر را برآورده نمایند.

۱-۵ اجزاء سیستم کامپیوتری

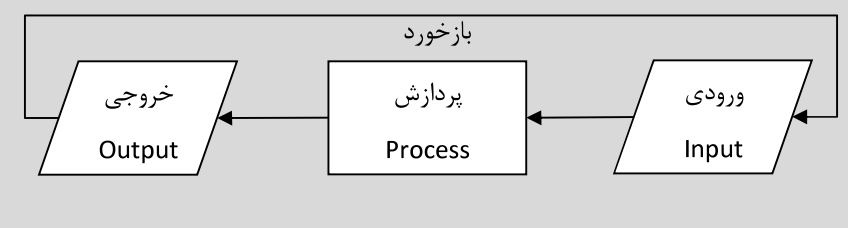
هر سیستم به طور کلی از مجموعه‌ای از اجزاء تشکیل شده است که در نهایت برای رسیدن به یک یا چند هدف با یکدیگر در تعامل هستند. نمودار کلی یک سیستم معمولا به شکل زیر می‌باشد:

---

[۱] MicroComputers

[۲] Software

[۳] Hardware



شکل ۱-۱ **نمودار سیستم**

برخی اوقات، یک بازخورد نیز از خروجی به ورودی فرستاده می‌شود تا عمل کنترل به طور بسته صورت پذیرد. این نمودار در مورد هر سیستم کامپیوتری نیز صدق می‌کند. اگر از جنبه سخت‌افزاری نگاه کنیم، ابزارهای ورودی، ابزارهای پردازشی و ابزارهای خروجی، اجزاء این نمودار را تشکیل می‌دهند. یعنی ورودی از طریق یک سری ابزارهای ورودی همانند: صفحه کلید، ماوس و غیره به داخل سیستم هدایت می‌شوند. عمل پردازش توسط مجموعه ابزارهای پردازشی (واحد پردازش مرکزی، حافظه اصلی و غیره) صورت گرفته و در نهایت خروجی توسط ابزارهای خروجی همانند: صفحه نمایش، چاپگر و سایر ابزارهای خروجی به کاربر اعلام می‌شود.

اگر از جنبه نرم‌افزاری نگاه کنیم، هر نرم‌افزار کامپیوتری دارای یک سری ورودی می‌باشد. بخش پردازشی هر برنامه روی داده‌های ورودی پردازش انجام می‌دهد و در نهایت یک سری اطلاعات به‌عنوان خروجی تولید می‌شود. به عنوان مثال: در برنامه سیستم آموزشی دانشگاه، داده‌های ورودی می‌تواند شامل: مشخصات دانشجویان، دروس انتخابی و نمرات آنها باشد. عمل پردازش شامل محاسبه معدل و واحدهای گذرانده می‌باشد و در نهایت یک خروجی همانند کارنامه کل از این روند حاصل خواهد شد.

۱-۶ سخت افزار

سخت‌افزارهای ورودی

رایج‌ترین سخت‌افزار ورودی هر سیستم کامپیوتری، صفحه‌کلید می‌باشد. صفحه‌کلیدها به دلیل دارابودن دکمه‌های حرفی، عددی و کلیدهای ویژه قابلیت انعطاف بیشتری در ورود داده‌ها دارند. پس از آن می‌توان ماوس[۱]، قلم نوری[۲]، گوی غلطان[۳]، صفحه لمسی[۴] و دسته‌های بازی[۵] را نیز نام برد که این

[۱] Mouse
[۲] Light Pen
[۳] Tracker Ball
[۴] Touch Pad
[۵] Joystick

گروه معمولاً با عنوان *ابزارهای اشاره‌گری*[۱] شناخته می‌شوند. ابزارهای ورودی دیگری نیز همچون اسکنر[۲]، میکروفون[۳] و دوربین وب[۴] نیز موجود هستند که برای ورود تصویر و صدا به کامپیوتر استفاده می‌شوند.

سخت‌افزارهای خروجی

رایج‌ترین سخت‌افزار خروجی، صفحه نمایش یا همان مانیتور می‌باشد. صفحه‌های نمایش با استفاده از تکنولوژی‌های مختلفی مانند CRT، LED،LCD و پلاسما ساخته و عرضه می‌شوند. چاپگرها[۵]، پلاترها[۶] و ویدئوپروژکتورها[۷] نیز نمونه های دیگری از ابزارهای خروجی سیستم های کامپیوتری هستند.

سخت‌افزارهای پردازشی

در واحد پردازشی یک سیستم کامپیوتری، اجزاء مختلفی با یکدیگر در تعامل هستند. مهمترین آنها، *واحد پردازشگر مرکزی*[۸] می‌باشد که وظیفه اجرای دستورالعمل‌های برنامه‌های کامپیوتری را برعهده دارد. این واحد وظیفه کنترل و مدیریت ابزارهای ورودی، خروجی را نیز بر عهده دارد. گرچه برای کنترل این ابزارها به اجزاء کمکی دیگری نیز نیاز می‌باشد. به عنوان مثال واحد پردازش مرکزی دستورالعمل‌های موردنیاز اجرا را از *حافظه اصلی*[۹] فرا می‌خواند. این حافظه که از نوع *حافظه قابل خواندن/نوشتن*[۱۰] می باشد، برنامه‌های در حال اجرا را در خود جا می‌دهد. این نوع حافظه از نوع *حافظه با دستیابی تصادفی*[۱۱] می‌باشد.

حافظه[۱۲]

حافظه‌ها، محل نگهداری داده‌ها هستند. دو نوع عمده حافظه عبارتند از: حافظه اصلی و حافظه جانبی یا ثانویه. حافظه‌های اصلی برای اجرای برنامه‌ها و حافظه‌های ثانویه برای نگهداری بلندمدت برنامه‌ها بکار گرفته می‌شوند. معمولا حافظه‌های جانبی تحت عنوان رسانه‌های ذخیره‌سازی مورد بررسی قرار می‌گیرند. دو نوع حافظه اصلی مورد استفاده در کامپیوتر عبارتند از RAM و ROM. برنامه‌ها برای اجرا در RAM قرار می‌گیرند. بنابراین حافظه RAM حاوی برنامه‌های در حال اجرا می‌باشد. از آنجا که این نوع حافظه

---

[۱] Pointing Devices
[۲] Scanner
[۳] Microphone
[۴] Webcam
[۵] Printers
[۶] Plotters
[۷] Video Projectors
[۸] Central Processing Unit (CPU)
[۹] Main Memory
[۱۰] Read/Write Memory (RWM)
[۱۱] Random Access Memory (RAM)
[۱۲] Memory

کوتاه‌مدت می‌باشد با قطع برق، محتویات آن از بین می‌رود. حافظه ROM برای نگهداری برنامه مورد نیاز یک سیستم کامپیوتری برای شروع اولیه در هنگام روشن نمودن کامپیوتر مورد استفاده قرار می گیرد. واحد اندازه‌گیری ظرفیت حافظه‌ها واحدی به نام Byte می‌باشد. هر بایت توانایی ذخیره سازی ۸ بیت را دارد. هر بیت[۱]، یکی از دو مقدار صفر یا یک را داراست. برای حافظه‌های با ظرفیت بالاتر معمولاً واحدهای بزرگتری مانند کیلوبایت، مگابایت، گیگابایت، ترابایت و پتابایت را نیز بکار می‌گیرند. رابطه بین این مقادیر در جدول ۱-۱ آمده است.

جدول ۱-۱ **واحدهای اندازه گیری ظرفیت حافظه‌ها**

واحد اندازه‌گیری	نشانه یا سمبل	برابری با دیگر واحدها
بایت	B	۸ بیت
کیلو بایت	KB	۱۰۲۴ بایت
مگا بایت	MB	۱۰۲۴ کیلو بایت
گیگا بایت	GB	۱۰۲۴ مگا بایت
ترا بایت	TB	۱۰۲۴ گیگا بایت
پتا بایت	PB	۱۰۲۴ ترا بایت
اگزا بایت	EB	۱۰۲۴ پتا بایت
زتا بایت	ZB	۱۰۲۴ اگزابایت

رسانه های ذخیره سازی

رسانه‌های ذخیره‌سازی برای نگهداری بلندمدت انواع داده‌ها و اطلاعات در سیستم‌های کامپیوتری مورد استفاده قرار می‌گیرند. این رسانه‌ها می‌توانند از تکنولوژی‌های مختلفی همچون: مغناطیسی، نوری، الکترومغناطیسی، نوری-مغناطیسی بهره گیرند. فلاپی‌دیسک‌ها، نوارهای مغناطیسی، دیسک‌های سخت، دیسک های نوری، DVDها، حافظه‌های فلش و دیسک‌های اشعه آبی[۲] انواع مختلفی از رسانه های ذخیره سازی هستند که در حال حاضر مورد استفاده قرار می‌گیرند یا قبلا مورد استفاده قرار گرفته‌اند. در ارزیابی رسانه های ذخیره سازی معمولا پارامترهای ظرفیت و سرعت انتقال داده مورد توجه قرار می‌گیرد.

۱-۷ نرم‌افزار

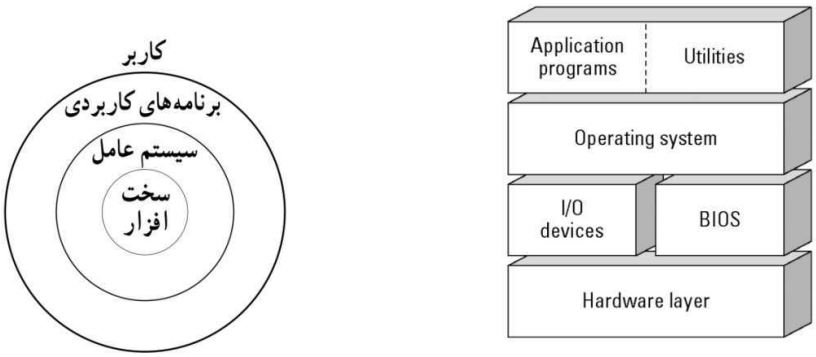
همانطور که گفته شد، نرم‌افزار به مجموعه برنامه‌ها و داده‌هایی گفته می‌شود که به سخت افزار روح و جان می‌بخشند. در حقیقت نرم‌افزار حلقه واسط بین کاربر و سخت افزار می‌باشد. کاربر به راحتی نمی تواند از سخت‌افزار بهره گیرد و این کار را از طریق نرم‌افزار انجام می‌دهد. نرم‌افزارها را به دو گروه عمده تقسیم‌بندی می‌کنند: نرم‌افزارهای سیستمی و نرم‌افزارهای کاربردی.

[۱] Bit (*Bi*nary Dig*it*)

[۲] Blue Ray Discs

۱-۷-۱ نرم‌افزارهای سیستمی

نرم‌افزارهای سیستمی با سخت‌افزار کامپیوتر (شامل واحدهای ورودی، خروجی، حافظه و پردازش مرکزی) ارتباط مستقیم دارند و عملیات مربوطه از طریق این نرم‌افزارها هدایت و کنترل می‌شوند. نرم افزارهای سیستمی معمولا به عنوان رابط بین سخت‌افزارها، نرم‌افزارهای کاربردی و کاربران عمل می کنند. این نرم‌افزارها به وسیله برنامه‌نویسان حرفه‌ای طراحی و به بازار عرضه می‌شوند. سیستم‌عامل، مهمترین نرم‌افزار سیستمی محسوب می‌شود. شکل ۱-۲ جایگاه نرم‌افزارها را در یک سیستم رایانه‌ای نشان می‌دهد.



شکل ۱-۲ سلسله مراتب نرم‌افزارها در سیستم

سیستم عامل[۱]

سیستم عامل یکی از مهمترین نرم‌افزارها در کامپیوتر است و به عنوان نرم‌افزار رابط بین کاربر و سخت‌افزار با روشن شدن رایانه فعال شده و پس از آغاز به کار، محیط را برای کار با نرم‌افزارهای کاربردی آماده می‌کند. هنگام خاموش کردن نیز سیستم عامل پس از بستن همه برنامه‌ها، به عنوان آخرین نرم افزار کار خود را به اتمام می‌رساند. سیستم عامل با سازماندهی، مدیریت و کنترل منابع سخت افزاری امکان استفاده بهینه از آنها را فراهم می‌کند. اکثر کامپیوترها برای کار به یک سیستم عامل نیاز دارند و معمولا سیستم عامل اولین نرم‌افزاری است که در کامپیوتر نصب می‌شود.

وظایف اصلی هر سیستم عامل عبارتند از:

- مدیریت منابع
- ایجاد سهولت کار با کامپیوتر
- اجرای برنامه های کاربردی

[۱] Operating System

مدیریت منابع: منابع یک سیستم کامپیوتری عبارتند از: واحد پردازنده مرکزی، حافظه اصلی، وسایل ورودی/خروجی، حافظه های جانبی، داده‌ها و دستورالعمل‌ها، که سیستم عامل وظیفه مدیریت آنها را بر عهده دارد.

ایجاد سهولت جهت کار با کامپیوتر سیستم عامل نقش یک رابط را برای ماشین و کاربر ایفا می‌کند. رابط کاربر قسمتی از سیستم عامل است که توسط کاربر قابل کنترل بوده و به او اجازه می‌دهد از طریق آن با سیستم عامل ارتباط برقرار کند. رابط کاربر تعیین کننده شیوه دریافت دستورات از کاربر است. رابط ها به دو شکل دستوری و گرافیکی می‌باشند. رابط دستوری کاربر را ملزم می‌سازد که دستور مورد نظرش را مستقیما با کد یا کلمات تایپ نماید. اما کار با رابط گرافیکی ساده‌تر و جذاب‌تر از رابط دستوری می‌باشد. رابط گرافیکی به کاربر اجازه می‌دهد که با استفاده از اشکال گرافیکی، عملیاتی از قبیل اجرای برنامه‌ها، نمایش لیستی از فایل‌ها و غیره را انجام دهد.

اجرای برنامه‌های کاربردی برنامه‌های کاربردی بدون وجود سیستم عامل قابل اجرا نیستند. سیستم عامل، محیط مناسب برای اجرای برنامه‌های کاربردی را فراهم می‌کند. به عنوان مثال، در اجرای یک برنامه واژه پرداز، نیاز به سیستم‌عاملی داریم که داده ورودی را بگیرد، آن را روی دیسک ذخیره کرده و متن تایپ شده را چاپ نماید.

مترجم‌های زبان‌های برنامه‌نویسی

مترجم‌های زبان‌های برنامه‌نویسی نیز جزء نرم‌افزارهای سیستمی به حساب می‌آیند. زبان‌های برنامه-نویسی برای حل مسایل توسط کامپیوتر بکار می‌روند. برای حل یک مسئله با استفاده از کامپیوتر ابتدا باید الگوریتم مناسبی برای آن طراحی نمود و سپس با استفاده از قواعد و اصول آن زبان برنامه‌نویسی آن را به یک برنامه کامپیوتری تبدیل نمود. زبانهای برنامه نویسی را به سطوح مختلفی تقسیم‌بندی می‌کنند. یک تقسیم‌بندی رایج عبارت است از: زبان های سطح بالا، زبان های سطح میانی و زبان‌های سطح پایین. زبان‌های برنامه‌نویسی سطح بالا بیشتر به زبان‌های محاوره‌ای انسانی نزدیکتر است و زبان‌های سطح پایین به زبان ماشین. زبان‌های سطح میانی نیز ترکیبی از این دو سطح هستند یعنی در برخی موارد بسیار نزدیک به زبان ماشین و تا حدودی نیز نزدیک به زبان محاوره‌ای می‌باشند. به عنوان مثال: زبان برنامه نویسی پاسکال یک زبان سطح بالا و زبان اسمبلی یک زبان سطح پایین محسوب می‌شود.

زمانیکه شما یک برنامه را با یک زبان برنامه‌نویسی سطح بالا می‌نویسید، کامپیوتری که فقط صفر و یک را متوجه می‌شود، درکی از برنامه شما و کدهای درون آن نخواهد داشت. بنابراین شما به ابزاری نیاز دارید که این برنامه سطح بالا را به زبانی تبدیل کند که برای کامپیوتر قابل فهم باشد. اینجا درست زمانی است که *کامپایلر*[۱]ها و *مفسر*[۲]ها به کمک ما می‌آیند و هر دوی آنها یک کار را برای ما انجام می‌دهند،

[۱] Compiler

[۲] Interpreter

آنها زبان سطح بالا را به زبانی که کامپیوتر متوجه شود ترجمه می‌کنند. مهمترین تفاوتی که بین یک کامپایلر و یک مفسر وجود دارد روشی است که آنها کد اجرایی برنامه را اجرا می‌کنند. مفسر کد برنامه را خط به خط برای ترجمه و اجرا به کامپیوتر ارسال می‌کند در حالیکه کامپایلر برنامه شما را به یکباره و به طور کامل به کد قابل اجرا ترجمه نموده و کد ترجمه‌شده را در اختیار کامپیوتر قرار می‌دهد.

غیر از بحث ترجمه یکباره و خط به خط کد برنامه، یکی دیگر از مهمترین تفاوت‌هایی که بین کامپایلر و مفسر وجود دارد و مهمترین تفاوت این دو نوع مترجم نیز می‌باشد بحث وابستگی به برنامه است. برنامه یا کد نرم‌افزاری که توسط یک زبان برنامه‌نویسی مفسری نوشته شده است برای اینکه بتواند بر روی یک سیستم اجرا شود حتما نیاز به این دارد که مفسر مورد نظر از قبل روی سیستم نصب شده باشد و تا اینکار انجام نشود اجرای برنامه امکانپذیر نیست. بنابراین نرم‌افزارهایی که به زبان‌های برنامه‌نویسی مفسری نوشته می‌شوند برای اجرا شدن حتما به مفسر مورد نظر نیاز دارند. اما بر خلاف مفسرها، کامپایلر یکبار برای همیشه یک برنامه را به زبان اجرایی ماشین تبدیل می‌کند و در اصطلاح یکبار برنامه را به همراه کدهای اجرایی آن کامپایل می‌کند، خروجی یک کامپایلر یک یا چند فایل است که فارغ از وجود کد اصلی برنامه و یا کامپایلر، قادر به اجرا شدن بر روی هر سیستمی هستند و در واقع هیچ وابستگی به کامپایلر بعد از تبدیل کد وجود نخواهد داشت.

از معروف‌ترین زبان‌های مفسری که بیشترین استفاده را دارند می‌توان زبان‌های BASIC، Jscript، MATLAB، Perl، PHP، Python، PostScript، Ruby، VBScriptو PowerShell را نام برد. در عین حال از معروف‌ترین زبان‌های کامپایلری که بیشترین استفاده را دارند می‌توان به زبان‌های ALGOL ، GCC، Visual C++، Borland C++، Visual C# و Turbo Pascal اشاره کرد.

۱-۷-۲ برنامه های کاربردی

گروه دیگر نرم‌افزارها، نرم‌افزارهای کاربردی هستند. نرم‌افزارهای کاربردی، برای کاربردهای متفاوتی تولید می‌شوند. واژه‌پردازها، نرم‌افزارهای گرافیکی، نرم‌افزارهای چندرسانه‌ای، بازیها و غیره، همه و همه نمونه هایی از نرم‌افزارهای کاربردی هستند. برنامه‌های کاربردی بر پایه سیستم عامل اجرا می‌گردند. بنابراین قبل از اینکه بتوان یک برنامه کاربردی را اجرا نمود، باید سیستم عامل متناسب با آن برنامه بر روی سیستم کامپیوتری نصب شده باشد.

۱-۸ مراحل ایجاد و توسعه یک برنامه

هر برنامه یا سیستم نرم‌افزاری، در طول حیات خود مراحلی را طی می‌کند. این مراحل ابتدا با طرح مسئله شروع می‌شود. سپس باید برای مسئله موردنظر راه حلی یافت. این راه حل، الگوریتم حل مسئله نام دارد. الگوریتم حل مسئله می تواند دارای سطوح پیچیدگی متفاوتی باشد. پس از حل مسئله، آن را با یکی از زبان‌های برنامه‌نویسی، کدنویسی می‌کنند. بعد از این مرحله، مرحله تست یک برنامه صورت

می‌پذیرد تا خطاهای آن کشف و رفع گردند. تست و خطایابی روش‌های مختلفی دارد که در طول کتاب با آنها آشنا خواهید شد.

خودآزمایی

۱. نسل‌های کامپیوترهای امروزی را نام برده و مشخصه‌های بارز هر نسل را نام ببرید.

۲. تقسیم‌بندی کامپیوترها را بر اساس قدرت پردازشی و حجم محاسبات قابل انجام آنها نام برده و توضیح دهید.

۳. نرم افزار و سخت افزار کامپیوتر را تعریف کرده و تفاوت آنها را بیان کنید.

۴. چند نمونه از سخت افزار ورودی و سخت افزار خروجی کامپیوتر را نام ببرید.

۵. انواع حافظه را نام برده و توضیح دهید.

۶. واحدهای اندازه گیری حافظه را نام برده و تبدیلات زیر را انجام دهید.

1 TB = ? KB _

10 KB = ? B _

32 GB = ? MB _

16 EB = ?GB _

۷. تقسیم بندی زبانهای برنامه نویسی را نام برده و هر کدام را توضیح دهید.

۸. تفاوت مترجم و مفسر را بیان نمایید.

۹. انواع مترجم‌های زبان‌های برنامه‌نویسی را نام برده و تفاوت‌های آنها را بیان نمائید.

۱۰. مراحل ایجاد و توسعه یک برنامه را نام برده و توضیح دهید.

فصل دوم

محاسبات در کامپیوتر

هدف کلی

آشنایی با سیستم‌های عددی و محاسبات رایج در کامپیوتر

هدف‌های رفتاری

انتظار می‌رود پس از مطالعه این فصل بتوانید:

ـ با انواع سیستم های عددی آشنا شوید.

ـ با اعداد در مبناهای عددی دو، هشت، ده و شانزده آشنا شوید.

ـ روش تبدیل یک عدد در مبنای متداول را به مبناهای دیگر را فرا گیرید.

ـ اعمال ابتدایی بر روی اعداد با مبناهای مختلف را فرا گیرید.

مقدمه

آشنایی با سیستم اعداد مختلف برای فهم درست آنچه در یک سیستم کامپیوتری اتفاق می‌افتد، لازم و ضروری می‌نماید. از آنجا که سیستم‌های دیجیتال کامپیوتری بر پایه سیستم‌های عددی دودویی بنا نهاده شده‌اند، آشنایی با مبنای دو یک الزام برای یک دانشجوی کامپیوتر می‌باشد. در عمل سیستم‌ها با سیستم عددی دودویی کار می‌کنند ولی برای سهولت، سیستم‌های عددی شانزده‌تایی و هشت‌تایی نیز در سطوح بالاتر برنامه‌نویسی مورد استفاده قرار می‌گیرند. در این فصل، با سیستم‌های عددی دودویی، هشت‌تایی و شانزده تایی و نحوه تبدیل آنها آشنا خواهیم شد.

۲-۱ سیستم اعداد

در سیستم‌های عددی معمولی، موقعیت مکانی هر رقم دارای ارزش معینی است. در چنین سیستم هایی می‌توان هر عدد را به صورت های زیر نمایش داد:

$$N = (a_{n-1}a_{n-2}...a_1a_0a_{-1}a_{-2}...a_{-m})_B$$

$$N = a_{n-1}B^{n-1} + ... + a_0B^0 + a_{-1}B^{-1} + a_{-2}B^{-2} + ... + a_{-m}B^{-m}$$

$$N = \sum_{k=-m}^{n-1} a_k B^k$$

در نمایش فوق:

B، مبنای سیستم اعداد $0 < a_k < B-1$، m، تعداد ارقام اعشاری n، تعداد ارقام صحیح و $a_0, a_1, a_2, ...$، ضرایب می باشند.

هر عدد N در مبنای B به صورت $(N)_B$ نمایش داده می‌شود. اگر مبنای عدد مشخص نگردد، به طور پیش فرض مبنای ۱۰ در نظر گرفته می‌شود.

۲-۱-۱ سیستم اعداد دهدهی (عدد در مبنای ۱۰)

سیستم اعداد دهدهی یکی از سیستم‌های متداول است که همگان روزانه با آن سر و کار دارند. این سیستم را دهدهی گویند زیرا از ۱۰ رقم استفاده می‌کند یعنی هر عدد می‌تواند ترکیبی از ارقام ۰ تا ۹ باشد و ضرایب در توانی از ۱۰ ضرب می‌شوند.

$$N = \sum_{k=-m}^{n-1} a_k (10)^k = a_{n-1}10^{n-1} + ... + a_0 10^0 + a_{-1}10^{-1} + ... + a_{-m}10^{-m}$$

مثال ۲-۱ عدد ۱۲۳٫۴۵ را در مبنای ۱۰ بسط دهید.

$$123.45 = 1\times10^2 + 2\times10^1 + 3 + 4\times10^{-1} + 5\times10^{-2}$$

۲-۱-۲ سیستم اعداد دودویی (عدد در مبنای ۲)

سیستم دودویی، یک سیستم اعداد متفاوت است. ضرایب سیستم دودویی فقط دو مقدار ممکن را دارند: (هر عدد ترکیبی از ۰ و ۱ می‌باشد). هر ضریب a_x در 2^x به‌صورت زیر ضرب می‌گردد.

$$N=\sum_{k=-m}^{n-1} a_k(2)^k = a_{n-1}2^{n-1}+\ldots+a_0 2^0+a_{-1}2^{-1}+\ldots+a_{-m}2^{-m}$$

مثال ۲-۲ عدد ۱۱۰۱۱/۰۱ را در مبنای ۲ بسط دهید.

$$11011.01 = 1\times 2^4+1\times 2^3+0\times 2^2+1\times 2^1+1+0\times 2^{-1}+1\times 2^{-2}$$

۲-۱-۳ سیستم اعداد هشتایی (اُکتال) (عدد در مبنای ۸)

در این سیستم، مبنای اعداد ۸ است. (B=۸) لذا هر عدد در این سیستم می‌تواند ترکیبی از ارقام ۰ تا ۷ باشد.

$$N=\sum_{k=-m}^{n-1} a_k(8)^k = a_{n-1}8^{n-1}+\ldots+a_0 8^0+a_{-1}8^{-1}+\ldots+a_{-m}8^{-m}$$

۲-۱-۴ سیستم اعداد شانزدهی (هگزا دسیمال)

در این سیستم مبنای اعداد ۱۶ است (B=۱۶). لذا می‌توان از ۱۶ رقم در نوشتن اعداد این سیستم استفاده کرد. چون اعداد ۹ به بالا را به عنوان یک رقم نمی‌شناسیم، برای نمایش ارقام ۱۰ تا ۱۵ از علائم A تا F استفاده می‌کنیم. لذا ارقام مبنای ۱۶ عبارتند از :

0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , A , B , C , D , E , F

$$N=\sum_{k=-m}^{n-1} a_k(16)^k = a_{n-1}(16)^{n-1}+\ldots+a_0(16)^0+a_{-1}(16)^{-1}+\ldots+a_{-m}(16)^{-m}$$

اعداد B1A و ABC و 98D اعدادی در مبنای ۱۶ هستند.

جدول ۱-۲ اعداد با مبناهای متفاوت را نشان می‌دهد.

جدول ۱-۲ **اعداد با مبناهای متفاوت**

شانزده تایی (مبنای ۱۶)	هشتایی (مبنای ۸)	دودویی (مبنای ۲)	دهدهی (مبنای ۱۰)
۰	۰۰	۰۰۰۰	۰۰

۱	۰۱	۰۰۰۱	۰۱
۲	۰۲	۰۰۱۰	۰۲
۳	۰۳	۰۰۱۱	۰۳
۴	۰۴	۰۱۰۰	۰۴
۵	۰۵	۰۱۰۱	۰۵
۶	۰۶	۰۱۱۰	۰۶
۷	۰۷	۰۱۱۱	۰۷
۸	۱۰	۱۰۰۰	۰۸
۹	۱۱	۱۰۰۱	۰۹
A	۱۲	۱۰۱۰	۱۰
B	۱۳	۱۰۱۱	۱۱
C	۱۴	۱۱۰۰	۱۲
D	۱۵	۱۱۰۱	۱۳
E	۱۶	۱۱۱۰	۱۴
F	۱۷	۱۱۱۱	۱۵

۲-۲ تبدیل مبناها

چون در سیستم‌های کامپیوتری با مبناهای ۲ و ۸ و ۱۶ سرو کار داریم، لازم است چگونگی تبدیل مبناهای مذکور را به یکدیگر مطالعه کنیم. تبدیل از مبنای ۲ به مبناهای ۸ و ۱۶ و بالعکس نقش عمده ای در کامپیوترهای دیجیتال دارند. تبدیلاتی که مورد بحث قرار می‌گیرند عبارتند از دهدهی به دودویی و بالعکس، هشتایی به دودویی و بالعکس و در آخر هگزا دسیمال به دودویی و بالعکس.

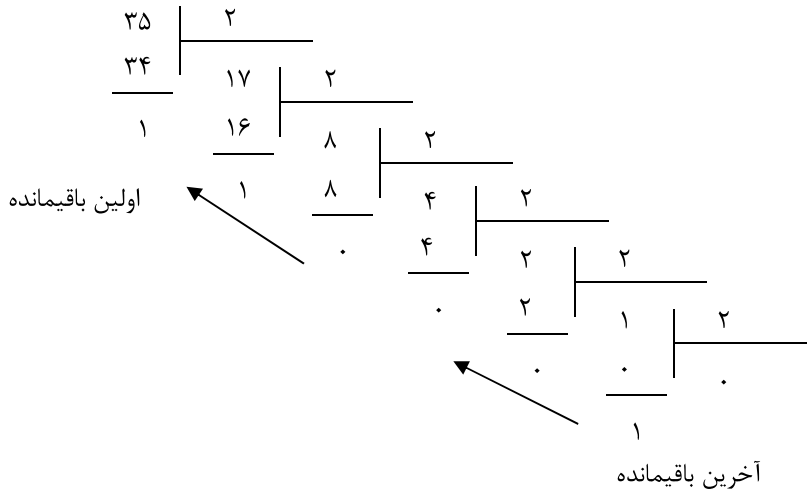
۲-۲-۱ تبدیل اعداد دهدهی به دودویی و بالعکس

تبدیل اعداد دهدهی را از دو جهت صحیح و اعشاری بودن مورد بررسی قرار می‌دهیم.

۲-۲-۱-۱ تبدیل اعداد صحیح دهدهی به دودویی و بالعکس

برای تبدیل اعداد صحیح دهدهی به دودویی از روش تقسیمات متوالی عدد دهدهی بر ۲ استفاده می شود و باقیمانده و خارج قسمت محاسبه می‌گردند. این تقسیم تا صفر شدن خارج قسمت ادامه می‌یابد. باقیمانده‌های ایجاد شده از تقسیم نگهداری می‌شوند و در آخر از آخرین باقیمانده به اولین باقیمانده در کنار هم نوشته می‌شوند. عدد حاصل، عدد در مبنای ۲ خواهد بود.

مثال ۳-۲ عدد ۳۵ را به مبنای ۲ تبدیل نمایید.



$(۱۰۰۰۱۱)_۲ = ۳۵$

روش دیگری که می‌توان برای تبدیل یک عدد دهدهی به دودویی بکار برد، این است که وزن مربوط به هر یک از مکانهای عدد دودویی را در بالای آن بنویسیم و سپس عدد مورد نظر را با استفاده از تفریق‌های متوالی (که نسبت به عمل تقسیم‌های متوالی ساده‌تر می‌باشد) به مبنای ۲ تبدیل نماییم. به این ترتیب که از بالاترین ارزش موجود در صورت شامل بودن وزن مربوطه در خانه مربوط به آن وزن عدد ۱ و در غیر اینصورت عدد صفر قرار دهیم. در زیر، ارزش مکانی هر یک از مکانهای یک عدد دودویی نمایش داده شده است:

$۲^۷$	$۲^۶$	$۲^۵$	$۲^۴$	$۲^۳$	$۲^۲$	$۲^۱$	$۲^۰$

یا

۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱

برای آشنایی بیشتر عدد ۳۵ را که در مثال بالا ذکر کردیم با استفاده از این روش به عدد دودویی تبدیل می‌کنیم.

توضیح از آنجا که عدد ۳۵ به ترتیب از بالاترین ارزش شامل یک عدد ۳۲، یک عدد ۲ و یک عدد ۱ می‌باشد. لذا در خانه‌های مربوط به ارزش‌های مکانی ذکر شده ۱ و در بقیه ۰ (صفر) قرار می‌دهیم.

۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱
۰	۰	۱	۰	۰	۰	۱	۱

$(۱۰۰۰۱۱)_۲ = ۳۵$

برای تبدیل اعداد در مبنای ۲ به مبنای ۱۰ باید این عدد را با ضرب هر یک از ارقام در توانی از دو بسط داد. به عنوان مثال بسط عدد $(۱۰۰۰۱۱)_۲$ به صورت زیر است :

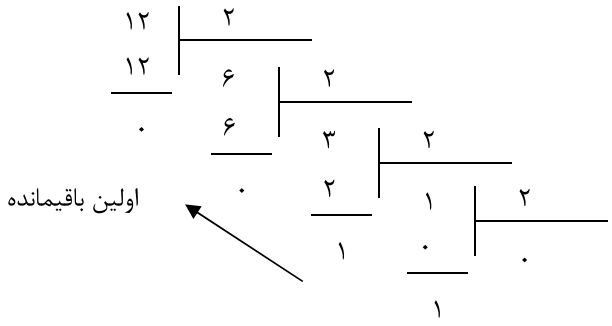
$$(۱۰۰۰۱۱)=۱\times۲^۵+۰\times۲^۴+۰\times۲^۳+۰\times۲^۲+۱\times۲^۱+۱=۳۲+۲+۱=۳۵$$

روش ساده برای ساده تر شدن کار، می توان فقط مقادیر بیت های دارای مقدار یک را باهم جمع نمود. چون مقادیر بیت های صفر در توان های دو ضرب شده و تأثیری در حاصل نخواهند داشت. در مثال ذکر شده ۳ عدد یک داریم لذا دهدهی مربوطه، جمع ۳ توان از ۲ می باشد.

۲-۲-۱-۲ تبدیل اعداد اعشاری دهدهی به دودویی و بالعکس

برای تبدیل اعداد اعشاری مبنای ۱۰ به ۲ باید قسمت صحیح و اعشاری را جداگانه به مبنای ۲ تبدیل کرد. برای تبدیل قسمت صحیح از روش تقسیمات متوالی و یا تفریقات متوالی و برای تبدیل قسمت اعشاری از ضرب متوالی در ۲ استفاده می‌گردد. در روش ضرب متوالی در ۲، قسمت اعشار در ۲ ضرب شده، قسمت صحیح حاصل از ضرب، نگهداری می‌شود و این روند برای قسمت اعشاری حاصل ادامه می یابد تا قسمت اعشار به صفر برسد. سپس قسمت‌های صحیح حاصل را به ترتیب در کنار هم می‌نویسیم. عدد حاصل، در مبنای دو خواهد بود. با تلفیق قسمت اعشاری و قسمت صحیح، عدد به طور کامل به مبنای ۲ تبدیل می‌شود.

مثال ۴-۲ عدد ۱۲٫۲۵ را به مبنای ۲ تبدیل نمایید.



$(۰.۲۵)\times۲=۰.\underline{۵}$

$۰.۵\times۲=۱.\underline{۰}$

$(۰.۲۵)_{۱۰}=(۰.۰۱)_۲$

با تلفیق قسمت‌های صحیح و اعشاری در مبنای دو، عدد ۱۲٫۲۵ به شکل زیر حاصل می‌شود:

$$(12.25)_{10}=(1100.01)_2$$

اگر با ضرب‌های متوالی قسمت اعشار به صفر نرسد، باید عمل ضرب را تا پر شدن کلمه حافظه ادامه داد.
برای تبدیل اعداد اعشاری مبنایِ ۲ به مبنایِ ۱۰ از روش بسط که قبلاً توضیح داده شد، استفاده می‌کنیم.

مثال ۲-۵ تبدیل $(۱۱۱۰/۰۱)_۲$ به مبنای ۱۰ به صورت زیر می‌باشد:

$$1110.01 = 1\times2^3 + 1\times2^2 + 1\times2^1 + 0 + 0\times2^{-1} + 1\times2^{-2} = 8+4+2+0+\frac{1}{4} = 14.25$$

۲-۲-۳ تبدیل اعداد مبنای ۲ به مبنای ۱۶ و ۸ و بالعکس

برای تبدیل اعداد در مبنای ۸ به مبنای ۲ مشابه تبدیل اعداد دهدهی به دودویی از تقسیمات متوالی بر ۲ استفاده می شود و برعکس برای تبدیل اعداد دودویی به اعداد مبنای ۸ از روش بسط می توان استفاده کرد. ولی ما از روش ساده تری برای این تبدیلات استفاده می کنیم به این ترتیب که :

چون۸ = $۲^۳$ و ۱۶ = $۲^۴$ است، لذا هر رقم در مبنای ۸ معادل با ۳ رقم دودویی و هر رقم در مبنای ۱۶ معادل با ۴ رقم دودویی می‌باشد.

تبدیل عدد دودویی به هشت هشتی به سادگی با تفکیک عدد دودویی به گروه های ۳ رقمی در دو طرف نقطه اعشار دودویی به دست می‌آید، و سپس به هر گروه یک رقم مبنای ۸ تعلق می‌گیرد و همچنین برای تبدیل اعداد مبنای ۸ به مبنای ۲، باید به جای هر رقمِ مبنای ۸ سه رقمِ مبنای ۲ را قرار داد.

مثال ۲-۶ تبدیل عدد $(۱۰۰۱۱,۱۱۰۱)_۲$ به مبنای ۸

صفرهای اضافه شده در قسمت اعشار (۰۱۱۰۰ . ۱۱۱۰۰) صفر اضافه شده در قسمت صحیح

چون صفر بعد از اعشار و قبل از صحیح ارزش مکانی ندارد لذا می‌توان از هر دو طرف صفر اضافه کرد. اینکار صرفاً جهت تسهیل در تفهیم مطلب می‌باشد.

مثال ۲-۷ تبدیل عدد $(۲۵/۳۴)_۸$ به مبنای ۲

$$(25.34)_8 = (010101011100)_2 = (101010111)_2$$

- تبدیل از مبنای ۲ به مبنای ۱۶ نیز مشابه با روند تبدیل هشت هشتی است با این تفاوت که عدد دودویی به گروه هایی چهار رقمی تفکیک می‌شود و بر عکس در تبدیل مبنای ۱۶ به ۲ هر رقم معادل چهار رقمِ مبنای ۲ می‌باشد.

مثال ۲-۸ تبدیل عدد $(۱۱۱۱۱۰۱,۰۱۱۰)_۲$ به مبنای ۱۶

$$(۱۱۰۱۰۱۱۱ . ۰۱۱۰)_۲ = (۷D۶)_{۱۶}$$

مثال ۲-۹ تبدیل عدد $(F25.03)_{16}$ به مبنای ۲

$$(F۲۵.۰۳)_{۱۶} = (۱۱۱۱۰۰۱۰۰۱۰۱.۰۰۰۰۰۰۱۱)_{۲}$$

۲-۲-۴ انجام محاسبات در مبنای ۲ و ۱۶ (دودویی و هگزا دسیمال)

نمایش اطلاعات در کامپیوتر با استفاده از مبناهای ۲ و ۱۶ انجام می گیرد لذا آشنایی با نحوه محاسبات آنها ضروری است.

۲-۲-۴-۱عمل جمع دودویی

عمل جمع در مبنای ۲ مانند عمل جمع در مبنای ۱۰ است. همانطور که در سیستم دهدهی، ده بر یک داریم، در سیستم دودویی، دو بر یک خواهیم داشت.

توضیح ***دو بر یک:*** در حالتی که ۱ با ۱ جمع می‌شود (یعنی حاصل ۲ میباشد)، به جای آن صفر قرار گرفته (مانند حالتی از جمع دهدهی که حاصل ده میباشد) و ۱، به رقم با ارزش بالاتر منتقل می‌شود. این عمل را دو بر یک گویند. همچنین اگر بر اثر ۲ بر یک، عددِ یک به رقم بعدی منتقل شود و حاصل جمع دو رقم قبلی نیز صفر باشد، عددِ یک را نوشته و یکِ دیگر را به عنوان دو بر یک به رقم بعدی منتقل می کنیم.

مثال ۱۰ – ۲ حاصل جمع دو عدد دودویی 100111 و 111010 را محاسبه نمایید.

```
  1  1  1  1  1
     1  0  0  1  1  1
+    1  1  1  0  1  0
-------------------------
  1  1  0  0  0  0  1
```

۲-۲-۴-۲ عمل تفریق دودویی

انجام عمل تفریق در مبنای ۲ نیز تقریباً مشابه تفریق دهدهی است و لذا توجه به نکات زیر ضروری است.

تفریق	y	x
۱	۰	۱
۰	۰	۰
۰	۱	۱
*	۱	۰

برای تفریق ۱ از صفر (۱-۰) باید یک را از مبنای عدد (در اینجا ۲) کم کرد و در مرحله بعد به حساب آورد. (عمل قرض گرفتن از رقم قبلی)

مثال ۲–۱۱ انجام چند عمل تفریق دودویی

۰ ۲ ۰ ۲	۱۱۱۱۱-	۱۱۱۱۱-
~~۲~~ ~~۰~~ ~~۲~~ -	۱۰۱۰۱	۱۰۰۰
۱۰۰۱	۰۱۰۱۰	۱۰۱۱۱
۱۰۰۱		

۲-۲-۴-۳ ضرب و تقسیم دودویی

ضرب و تقسیم در مبنای ۲ مانند ضرب و تقسیم در مبنای ۱۰ می باشد.

مثال ۲–۱۲ انجام چند عمل ضرب

۱۱۱۱۰۱×	۱۰۱×
۱۰۱	۱۱
۱۱۱۱۰۱	۱۰۱
۰۰۰۰۰۰	۱۰۱
۱۱۱۱۰۱	۱۱۱۱
۱۰۰۱۱۰۰۰۱	

۲-۲-۴-۴ جمع و تفریقِ هگزا دسیمال

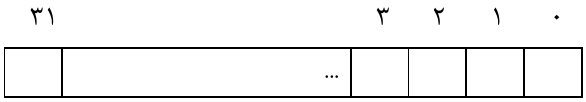
جمع و تفریق در این مبنا نیز مانند جمع و تفریق در مبنای ۲ و ۱۰است، با این تفاوت که در این مبنا، به جای ۱۰ بر یک و ۲ بر یک، ۱۶ بر یک خواهیم داشت. یعنی وقتی حاصل جمع دو رقم مساوی یا بیشتر از ۱۶ باشد ، عدد ۱۶ را از آن کم می‌کنیم و عدد هگزا دسیمال معادل آن را بدست می‌آوریم، و یک بیت نقلی برای اضافه کردن به رقم بعدی در نظر می گیریم. تفریق در مبنای ۱۶ نیز مشابه تفریقِ مبنای ۲ و ۱۰ می باشد.

مثال ۲–۱۳ انجام چند عمل جمع و تفریق در مبنای ۱۶

	1	
BFCA۲-	ABE۱۲+	۱A۵۳+
۱۴A۲	۳۵۴	۳۷۱
BE۸۰۰	AC۱۶۶	۱DC۴

۲-۳ نحوه ذخیره‌سازی اعداد صحیح مثبت و منفی در کامپیوتر

اعداد به صورت دودویی (باینری) در حافظه نگهداری می‌شوند. طول کلمات در کامپیوترهای مختلف ممکن است متفاوت باشد ولی معمولاً توانی از ۲ می‌باشد، مثل ۱۶ بیت، ۳۲ بیت و ۶۴ بیت. برای اعداد علامت دار، بزرگترین یا باارزش ترین بیت را، بیت علامت s[1] در نظر می گیرند. مثلاً اگر عدد ۸ بیتی را در نظر بگیریم، بیت آخر، بیت علامت و هفت بیت دیگر، قدر مطلق عدد است.

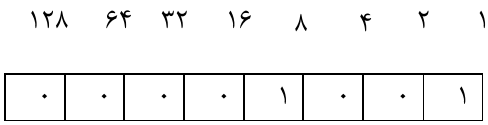


بیت علامت(S)

برای نمایش عدد صحیح مثبت، باید آن را به مبنای ۲ تبدیل کرده، کلمات ماشین را از سمت راست به چپ پر کرد و بیت علامت را برابر با صفر قرار داد. بیت های باقیمانده نیز با صفر پر می شوند.

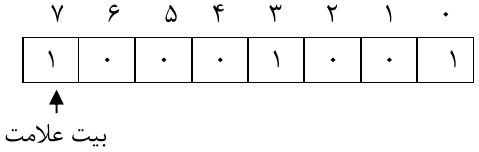
نگهداری اعداد منفی نیز همچون اعداد مثبت می باشد با این تفاوت که در بیت علامت مقدار یک قرار می گیرد.

مثال ۲-۱۴ نمایش اعداد ۹+ و ۹- در متغیری به طول یک بایت (یک بایت معادل ۸ بیت می باشد.)



$+۹ = (۰۰۰۰۱۰۰۱)_۲$

$-۹ = (-۰۰۰۰۱۰۰۱)_۲$



این نوع روش نگهداری برای اعداد منفی دارای دو اشکال عمده است :

۱- برای صفر منفی و صفر مثبت دو نمایش جداگانه وجود دارد.

۲- برای عمل تفریق باید مدار جداگانه ای طراحی شود.

[1] Sign bit

۲-۳-۱ نمایش صفر منفی و صفر مثبت در متغیری به طول یک بایت

۷	۶	۵	۴	۳	۲	۱	۰
۱	۰	۰	۰	۰	۰	۰	۰

نمایش صفر منفی

۷	۶	۵	۴	۳	۲	۱	۰
۰	۰	۰	۰	۰	۰	۰	۰

نمایش صفر مثبت

۲-۳-۲ روش متمم ۱

برای نمایش اعداد به روش متمم ۱، عدد را به صورت دودویی نوشته، نمایش مثبت ان را مشخص می کنیم و در نمایش حاصل، تمام صفرها را به یک و تمام یک ها را به صفر تبدیل می کنیم و یا به عبارت دیگر تمام ارقام را از ۱ کم می کنیم. (چون عدد در مبنای ۲ است ۱=۱-۲)

مثال ۲-۱۵ **نمایش عدد ۱۹- در متغیری به طول یک بایت به روش متمم ۱**

$-19=(-10011)_2$

نمایش عدد ۱۹ =۰۰۰۱۰۰۱۱

متمم عدد ۱۹- به روش متمم ۱ =۱۱۱۰۱۱۰۰

در این روش نیز برای صفر مثبت و منفی دو نمایش مختلف وجود دارد برای انجام عمل تفریق نیاز به مدار جداگانه ای نیست، یعنی روش متمم ۱، اشکال دوم روش علامت و مقدار را برطرف می کند.

نمایش صفر مثبت و منفی در روش متمم ۱، در متغیری به طول یک بایت

۷	۶	۵	۴	۳	۲	۱	۰
۱	۱	۱	۱	۱	۱	۱	۱

نمایش صفر منفی

۷	۶	۵	۴	۳	۲	۱	۰
۰	۰	۰	۰	۰	۰	۰	۰

نمایش صفر مثبت

۲-۶ روش متمم ۲

این روش هر دو اشکال روش علامت و مقدار را حل می‌کند. در این روش باید به صورت زیر عمل کرد :

۱- نمایش مثبت عدد

۲- پیدا کردن متمم ۱

۳- افزودن یک واحد به عدد حاصل

مثال ۲-۱۶ نمایش عدد ۱۹- در متغیری به طول یک بایت به روش متمم۲

$-۱۹=(-۱۰۰۱۱)_۲$

۱۹ نمایش عدد =۰۰۰۱۰۰۱۱

۱۹- متمم اعداد =۱۱۱۰۱۱۰۰

۱۹- متمم ۲ عدد =۱۱۱۰۱۱۰۱

* متمم ۲ یک عدد = متمم ۱ عدد + ۱

۲-۷ نحوه ذخیره اعداد اعشاری

اعداد اعشاری در هر مبنایی را می توان به صورت ممیز شناور نشان داد. به عنوان مثال ، عدد ۷۴۵,۰ را می توان به صورت های زیر نمایش داد :

۷۴۵.۰×۱۰^{۰}

۷۴.۵×۱۰^{۱}

۷.۴۵×۱۰^{۲}

۰.۷۴۵×۱۰^{۳}

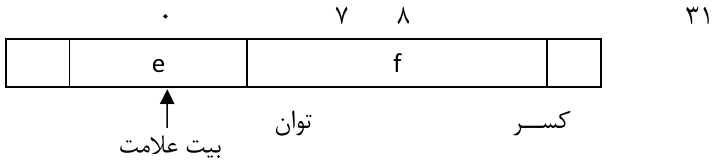
۰.۰۷۴۵×۱۰^{۴}

همان طور که ملاحظه می شود ممیز جای ثابتی ندارد. به همین دلیل آن را ممیز شناور گویند.هر عدد اعشاری را می توان به صورت زیر بیان کرد :

$$\pm f \times b^{\pm e}$$

در این نمایش f مقداری کسری، b مبنای عدد و e توان است. قسمت های کسری و توان می توانند مثبتو یا منفی باشند. به عنوان مثال در عدد ۰.۷۴۵×۱۰^{۳} f برابر با ۰,۷۴۵ ، b برابر با ۱۰ و e برابر با ۳ است.

اگر در نمایش $\pm f \times b^{\pm e}$ شرط $0.1 < f < 1$ برقرار باشد، عدد را نرمال گویند. در مبنای ۱۶، در عدد نرمال باید شرط $\frac{۱}{۱۶} < f < ۱$ برقرار باشد. اعداد ۰.۷۴۵×۱۰^{۳}, $۰.ADF\times۱۶^{۴}$ نرمال می باشند. برای نمایش اعداد اعشاری، باید f,b و e مشخص باشند. چون هر کامپیوتر با مبنای معینی کار می کند، لزومی به ذخیره عدد b نیست. لذا برای نمایش اعداد اعشاری کافی است قسمت کسر و توان را ذخیره کرد. کلمات کامپیوتر برای ذخیره اعداد اعشاری به سه قسمت تقسیم می شوند.



بیت علامت، مربوط به علامت قسمت کسری می باشد و علامت قسمت توان به شکل خاصی که در ادامه بحث می شود، اعمال می گردد. توان عدد در بیت های ۱ تا ۷ ذخیره می شودو قسمت کسری در بیت های ۸ تا ۳۱ ذخیره می گردد. توان عدد در ۷ بیت قرار می گیرد که می تواند $۲^{۷}$=۱۲۸ حالت مختلف داشته باشد. چون باید بتوان اعداد مثبت و منفی را نمایش داد، لذا e می تواند از ۶۴- تا ۶۳ را اختیار کند، اما e نه به صورتعلامت و مقدار و نه به صورت متمم۱ و نه به صورت متمم ۲ نمایش داده می شود، بلکه به روش خاصی بنام روش افزونی ۶۴ نمایش داده می شود. در این روش، به توان واقعی، که از ۶۴- تا ۶۳ است، ۶۴ واحد اضافه می شود تا توان جدید (توان ظاهری) از ۰ تا ۱۲۷ تغییر کند. f در ۲۴ بیت که معادل ۶ رقم مبنای ۱۶ است ذخیره می گردد.

مثال ۲-۱۷ نمایش عدد ۱۲۱,۸۴- در کامپیوتری به طول کلمات ۳۲ بیت.

$$۱۲۱ = (۱۱۱۱۰۰۱۱)_{۲}$$
$$۰.۸۴ = (۰.۱۱۰۱۰۱)_{۲}$$
$$-۱۲۱.۸۴ = (-۱۱۱۱۰۰۱.۱۱۰۱۰۱)_{۲}$$

تبدیل به مبنای ۱۶

$$-(۰۱۱۱۱۰۰۱.۱۱۰۱۰۱۰۰)_{۲} = -۷۹.D۴$$

تبدیل به عدد نرمال

$$-۷۹D۴ = ۰.۷۹D۴\times۱۶^{۲}$$

در این عدد نرمال داریم :

$$f = ۰.۷۹D۴$$
$$b^{۲} = ۱۶$$

۲= توان واقعی

توان ظاهری ۶۶=۶۴+۲

۶۶=(۱۰۰۰۰۱۰)$_{۲}$

همانطور که ملاحظه می شود، در قسمت کسری، ۶ رقم مبنای ۱۶ قرار می گیرد(هر رقم مبنای۱۶ ، چهار رقم مبنای ۲ است). اگر طول قسمت کسری از طول کلمه ماشین کمتر باشد، به تعداد لازم، صفر در سمت راست قرار می گیرد.

۱-۷-۲ نمایش اعداد به صورت کد بی- سی- دی (BCD[۲])

در روش‌های قبلی نمایش اعداد، هر عدد به صورت یک کمیت مستقل در نظر گرفته می‌شد. برای نمایش عدد به صورت *بی سی دی* هر رقم آن به طور مستقل به مبنای ۲ تبدیل می‌شود و هر رقم عدد، به چهار بیت تبدیل می گردد. به عنوان مثال، عدد ۱۲ به صورت ۰۰۰۱۰۰۱۰ در می‌آید. علت در نظر گرفتن چهار بیت برای هر رقم این است که بزرگترین رقم مبنای ده، عدد ۹ است که اگر به مبنای ۲ تبدیل شود عدد ۱۰۰۱ حاصل می‌گردد که به چهار بیت نیاز دارد.

مزیت این روش در سهولت تبدیل اعداد مبنای ۱۰ به BCD و سهولت انتقال اطلاعات عددی به حافظه و از حافظه به خروجی است. در این روش، مدارات لازم برای انجام محاسبات ریاضی و منطقی پیچیده‌تر خواهند بود.

برای ذخیره اعداد صحیح مثبت و منفی، چهار بیت دیگر به سمت راست عدد اضافه می‌شود. این چهار بیت، علامت عدد را مشخص می‌کنند.

کـــد	علامت عدد
۱۱۱۱	فاقد علامت
۱۱۰۰	مثبت
۱۱۰۱	منفی

مثال ۲-۱۸ نمایش عدد ۱۸ به روش کد BCD

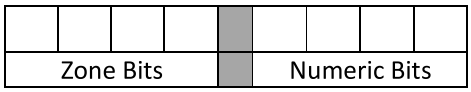
۱۸=۰۰۰۱۱۰۰۰

۰۰۰۱	۱۰۰۰	۱۱۱۱

[۲] Binary Coded Decimal

۲-۷-۲ نمایش اطلاعات به صورت کد اسکی

در اوایل دوران تولید کامپیوتر، برای نمایش اطلاعات کاراکتری، کدهای مختلفی طراحی شد، به طوری که هر شرکت سازنده کامپیوتر، طراحی کد مربوط خود را به کار می‌گرفت. انتخاب کد در ساخت کامپیوتر مؤثر بوده و بعد از ساختن کامپوتر تغییر کد به سادگی ممکن نبود. به دلیل عدم وجود یک کد استاندارد، انتقال اطلاعات از کامپیوتری به کامپیوتر دیگر، با مشکل مواجه بود. برای رفع این مشکل کمیته استاندارد تبادل اطلاعات آمریکا (ASCII) تصمیم گرفت کد استانداردی را تولید کند. این کد، کد اسکی نام گرفت. این کد در ابتدا از ۷ بیت تشکیل می‌شد. یعنی هر کاراکتر ۷ بیت را اشغال می‌کرد. لذا با ۷ بیت ۱۲۸ کاراکتر مختلف قابل نمایش است. (۱۲۸=$۲^{۷}$). با توسعه این سیستم، کد اسکی ۸ بیتی بوجود آمد. با این کد، ۲۵۶ کاراکتر مختلف قابل نمایش است. جدول زیر بخشی از کدهای اسکی را نمایش می دهد. چهار بیت سمت چپ هر کاراکتر، به نام zone bit و چهار بیت سمت راست به نام numeric bit است :



Zone bits						Numeric bits
۰۱۰۰	۰۱۰۱	۰۰۱۰	۰۰۱۱	۰۱۱۰	۰۱۱۱	
@	P	sp	۰	/	p	۰۰۰۰
A	Q	!	۱	a	q	۰۰۰۱
B	R	..	۲	b	r	۰۰۱۰
C	S	#	۳	c	s	۰۰۱۱

خودآزمائی

۱- تبدیلات زیر را انجام دهید.

۱.۵۲۲ = $(?)_{۲}$

۲.۱۲۵.۲۵ = $(?)_{۲}$

۳.(۱۱۰۰۱۱۱.۱۰۰۰۱) = $(?)_{۱۰}$

۴.(۱۰۱۰۱۱.۱۱۱۰۱) = $(?)_{۸}$

۵.(۱۱۰۰۱۱۱۰۰۰.۰۱۱۱۰۱) = $(?)_{۱۶}$

۶.$(۷۸۲۱۴۵)_{۸}$ = $(?)_{۱۶}$

۷.$(FAD۱۲E)_{۱۶}$ = $(?)_{۸}$

۸.$(AB۵۴۷)_{۱۶}$ = $(?)_{۲}$

۹.$(۶۷۲۱۳)_{۸}$ = $(?)_{۲}$

۲- اعداد زیر به صورت متمم ۲ در متغیری به طول یک بایت نمایش دهید.

۱۶-، ۴۳-، ۲۵-، ۵۶-

۳- شکل نرمال اعداد زیر را بنویسید.

۱.$(۳۷.۴۲۸)_{۱۰}$

۲.$(۶۱.۵۴)_{۱۶}$

۳.$(ABCD)_{۱۶}$

۴- اعمال زیر را انجام دهید.

۲- ۳ D ACF

۱ A ۹F

---

۱ ۰ ۱ ۰ ۰ ۱-

۱ ۰ ۱ ۱ ۰

---

۱ ۱ ۱ ۰ ۰ ۱ ۱+

۱ ۱ ۱ ۱

B۱A ۲ ۳+

۱ ۳ ۲ ۴

فصل سوم

حل مسئله (الگوریتم و فلوچارت)

هدف کلی

آشنایی با نحوه تعریف دقیق مسئله، بیان الگوریتم و رسم فلوچارت برای حل مسئله

هدف‌های رفتاری

انتظار می‌رود پس از مطالعه این فصل بتوانید:

ـ با نحوه تعریف دقیق یک مسئله آشنا شوید.

- الگوریتم را تعریف نموده و ویژگیهای اساسی آن را برشمارید.

ـ با نحوه طراحی الگوریتم برای مسائل مختلف آشنا شوید.

ـ قراردادهای استاندارد رسم فلوچارت را شناخته و به کار گیرید.

ـ برای مسائل مختلف، الگوریتم مناسب طراحی نموده و فلوچارت رسم نمایید.

ـ با مقادیر داده‌ای مناسب، مراحل اجرای فلوچارت‌های مختلف را دنبال نمایید.

مقدمه

برای حل یک مسئله، ابتدا باید آن مسئله را به طور دقیق تعریف نمود. بیان دقیق مسئله و فهم درست آن اولین گام در حل یک مسئله می‌باشد. در بیان دقیق مسئله، باید ورودی و خروجی مسئله تعیین شوند، بعبارتی باید تعیین نمود که مسئله چه داده‌هایی در اختیار ما قرار می‌دهد و چه اطلاعاتی را به عنوان خروجی انتظار دارد. گام بعدی ارائه یک راه حل برای مسئله است. این راه حل، تحت عنوان الگوریتم حل مسئله شناخته می‌شود. لغت الگوریتم از نام دانشمند ایرانی، *محمد بن موسی خوارزمی*[۱] برگرفته شده است. خوارزمی، ریاضیدان پرآوازه ایرانی در قرن سوم هجری بوده است که مفهوم الگوریتم را در ریاضیات ارائه داد.

در این فصل، ابتدا به بیان مفهوم الگوریتم خواهیم پرداخت. سپس با تعریف دقیق مسئله و نحوه بیان راه حل آن به زبان الگوریتم آشنا خواهیم شد. سپس به معرفی فلوچارت و نمادهای استاندارد آن خواهیم پرداخت و در ادامه فصل، مثال‌های متعددی را مورد بررسی قرار خواهیم داد.

۳-۱ الگوریتم

برای اینکه بتوانیم یک مسئله را با استفاده از یک برنامه کامپیوتری حل نماییم، ابتدا باید راه‌حلی برای آن پیدا نماییم. این راه‌حل باید قابل بیان به زبانی ساده و قابل تبدیل به اعمال و دستورالعمل‌های کامپیوتری باشد. راه حل پیشنهادی را الگوریتم[۲] حل مسأله می‌نامیم. پس *الگوریتم، راه حل مسئله است که به صورت قدم به قدم بیان شود بطوری که با دنبال نمودن قدم های آن، مسئله مورد نظر حل شود.* الگوریتم را می‌توان با جملات زبان متعارف بیان نمود ولی معمولاً برای اینکه این جملات برای همه افرادی که در این حوزه فعالیت می‌نمایند، قابل فهم باشد نیاز است تا یک سری قواعد استانداردی برای بیان آن تعریف و بکار گرفته شود. قبل از این با الگوریتمهای زیادی در دوران دبیرستان آشنا شده‌اید. الگوریتم تشخیص زوج و فرد بودن یک عدد، تشخیص اول بودن یک عدد، الگوریتم بدست آوردن بزرگترین مقسوم علیه مشترک دو عدد، الگوریتم بدست آوردن ریشه های معادله درجه دو و . . . نمونه هایی از الگوریتم‌های آشنایی هستند که قبل از این از آنها استفاده نموده‌اید.

۳-۱-۱ مشخصات الگوریتم

هر الگوریتم می‌بایستی ۵ ویژگی زیر را دارا باشد:

---

[۱] Muḥammad ibn Mūsā al-Khwārizmī

[۲] Algorithm

۱) *پایان پذیری*[1]. هر الگوریتم باید پس از تعداد مراحلی متناهی، پایان پذیرد.

۲) *تعریف پذیری*. هر یک از گام‌های الگوریتم باید بطور دقیق تعریف شده باشد. به بیان دیگر، جزئیات حل مسئله در الگوریتم بیان شده باشد. البته اینکه چه سطحی از جزئیات باید در بیان الگوریتم بیان گردد به عوامل مختلفی بستگی دارد.

۳) *ورودی*[2]. هر الگوریتم تعدادی ورودی دارد. ممکن است الگوریتمی ورودی نداشته باشد و برای بدست آوردن خروجی مشخصی برای وضعیتی معین طرح و حل شود.

۴) *خروجی*[3]. هر الگوریتم حداقل یک یا تعداد بیشتری خروجی دارد. خروجی الگوریتم نتیجه اعمال پردازشی انجام شده بر روی ورودی ها یا وضعیت مشخص شده می‌باشد.

۵) *انجام پذیربودن*. هر یک از عملهای الگوریتم باید به اندازه کافی پایه‌ای باشد به طوریکه در یک زمان متناهی و با استفاده از قلم و کاغذ توسط یک شخص قابل انجام باشد.

به عنوان نمونه، الگوریتم محاسبه قدرمطلق یک عدد را در نظر بگیرید. برای محاسبه قدرمطلق یک عدد، ابتدا باید بررسی نماییم آن عدد، کوچکتر از صفر (منفی) است که در این صورت قرینه آن و در غیر اینصورت (یعنی اگر عدد منفی نباشد که یا مثبت است و یا صفر) خود آن عدد به عنوان مقدار قدرمطلق آن عدد در نظر می‌گیریم.

به عنوان مثالی دیگر، الگوریتم یافتن ریشه‌های حقیقی معادله درجه ۲ را در نظر بگیرید. یک معادله درجه دو به فرم کلی $Ax^2+Bx+C=0$ می‌باشد. ورودی مسئله ضرایب A، B و C می باشد. همانطور که از دوران دبیرستان به یاد دارید، یک روش حل معادله درجه دو، روش موسوم به روش دلتا (Δ) می باشد. ابتدا مقدار دلتا را محاسبه نموده و سپس از روی مقدار بدست آمده برای دلتا، برای حالت $\Delta>0$ ریشه ها را از رابطه $X=\frac{-b\pm\sqrt{\Delta}}{2\times a}$ و برای حالت $\Delta=0$ ریشه مضاعف را از رابطه $X=\frac{-b}{2\times a}$ بدست می‌آوردیم و برای حالت $\Delta<0$ معادله ریشه حقیقی ندارد. در حقیقت این نوع بیان حل مسئله، یک الگوریتم برای حل معادله درجه دو می‌باشد.

برای حل یک مسئله، ابتدا باید تعریف دقیقی از آن مسئله داشته باشیم. تعریف دقیق مسئله و فهم آنچه که مسئله به دنبال آن است، اولین قدم در شروع طرح الگوریتم و حل آن مسئله می‌باشد. قدم بعدی شناسایی ورودی و خروجی‌های مسئله می‌باشد. اینکه چه داده‌هایی به عنوان ورودی مسئله داده خواهد شد و چه خروجی‌هایی از مسئله مورد انتظار است، گام دوم در حل مسئله به شمار می‌رود. در نهایت یافتن یک راه حل درست با استفاده از دانش مربوط به مسئله و طرح آن با گام های درستی از

---

[1] Finiteness

[2] Input

[3] Output

الگوریتم قدم آخر حل مسئله می‌باشد. بهتر است برای اطمینان از صحت الگوریتم، با دنبال‌نمودن گام‌های الگوریتم آن را مورد آزمون قرار داد.

در مثال زیر، الگوریتم یافتن ریشه‌های معادله درجه، آمده است.

مثال ۱-۳ الگوریتمی بنویسید که ضرایب یک معادله درجه دو را دریافت نموده و ریشه‌های آن را محاسبه و نمایش دهد.

مسئله: با در دست داشتن ضرایب یک معادله درجه دو، ریشه‌های آن را بیابید.

ورودی: ضرایب معادله درجه دو به فرم $Ax^2+Bx+C=0$ شامل : A , B , C

خروجی: ریشه(های) معادله در صورت وجود.

الگوریتم حل:

گام اول: دریافت ضرایب معادله (A و B و C) از کاربر

گام دوم: محاسبه دلتا طبق رابطه $\Delta = B^2 - (4 \times A \times C)$

گام سوم: اگر مقدار Δ عددی منفی است، عبارت «*معادله ریشه حقیقی ندارد*» را چاپ کن و به *گام هفتم* برو.

گام چهارم: اگر مقدار Δ برابر صفر است، معادله فقط یک ریشه حقیقی دارد که از رابطه $X = \frac{-b}{2 \times a}$ بدست می‌آید.

گام پنجم: اگر مقدار Δ عددی مثبت است، معادله دو ریشه حقیقی متمایز دارد که از رابطه $X = \frac{-b \pm \sqrt{\Delta}}{2 \times a}$ بدست می‌آید .

گام ششم: چاپ مقدار (مقادیر) ریشه معادله.

گام هفتم: پایان

■

معمولاً در الگوریتم‌ها، گامهایی بعنوان *گام شروع*[۱] و *گام پایان*[۲] نیز افزوده می‌شود که ابتدا و انتهای الگوریتم را مشخص نماید. همانطور که می‌بینید، در بیان الگوریتم به زبان متعارف، می‌توان از انواع لغات و جملات مورد استفاده در زبان محاوره‌ای استفاده نمود. این امر باعث ناسازگاری در الگوریتم‌های نوشته شده به زبانهای محاوره‌ای می‌شود. به عنوان مثال برای دریافت مقادیر، می‌توان از جملات «بگیر»، «دریافت کن»، «از کاربر بگیر»، «وارد کردن...» استفاده نمود. برای یکسان نمودن این عبارات در داخل فلوچارت ها از استاندارد یکسانی استفاده خواهیم نمود.

مثال ۲-۳ الگوریتمی برای تشخیص اول بودن یک عدد ارائه دهید.

یادآوری: *عدد اول*[۳]، عددی است که به غیر از عدد یک و خودش، هیچ مقسوم علیه دیگری نداشته باشد. بعبارت دیگر عددی، اول است که فقط دو مقسوم‌علیه داشته باشد. الگوریتم زیر، تعداد مقسوم‌علیه‌های

---

[۱] Start Step
[۲] Finish Step
[۳] Prime Number

عدد را می‌شمارد و اگر تعداد مقسوم‌علیه‌های عدد برابر ۲ بود آن عدد را به عنوان عدد اول و در غیر اینصورت به عنوان عددی غیر اول معرفی می‌کند.

مسئله: تشخیص اول بودن یک عدد
ورودی: یک عدد صحیح مثلاً Number
خروجی: پیغام «*اول است*» اگر Number اول باشد و «*اول نیست*» اگر Number اول نباشد.

الگوریتم حل:

گام اول: شروع
گام دوم: دریافت عدد Number از کاربر
گام سوم: مقدار Index را برابر یک و Counter را برابر صفر قرار بده
گام چهارم: اگر Index <= Number است، به *گام پنجم* و در غیر اینصورت به *گام هفتم* برو
گام پنجم: اگر باقیمانده تقسیم صحیح Number بر Index برابر صفر است، مقدار Counter را یک واحد افزایش بده
گام ششم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو
گام هفتم: اگر Counter برابر دو باشد، عبارت «*اول است*» را نمایش بده در غیر اینصورت عبارت «*اول نیست*» را نمایش بده.
گام هشتم: پایان

■

در الگوریتم فوق، عدد دریافتی از کاربر در متغیری با نام Number نگه‌داری می‌گردد. همچنین دو متغیر دیگر نیز در الگوریتم مورد استفاده قرار گرفته‌اند. متغیر Counter که برای نگه‌داری تعداد مقسوم علیه‌های عدد Number و متغیر Index برای نگه‌داری مقدار شمارنده‌ای که از یک تا عدد Number اضافه خواهد شد. در واقع در این الگوریتم از این واقعیت که هر عدد اول، فقط دو مقسوم علیه دارد، بهره می‌گیریم. پس از دریافت عدد، Index را برابر یک قرار می‌دهیم و در مراحل الگوریتم آن را افزایش می‌دهیم تا به مقدار Number برسد. زمانیکه از مقدار Number بزرگتر شد، یعنی شمارش به اتمام رسیده و کافی است بررسی نماییم تعداد مقسوم علیه های یافته شده برابر ۲ است یا نه؟ اگر تعداد مقسوم علیه های عدد برابر ۲ باشد بدین معنی است که آن عدد اول است در غیر اینصورت آن عدد اول نیست. برای یافتن مقسوم علیه‌های عدد، هر بار آن را به Index که از یک تا خود عدد در حال شمارش است، تقسیم نموده و باقیمانده تقسیم را مورد بررسی قرار می‌دهیم. اگر باقیمانده تقسیم Number بر Index برابر صفر باشد، یعنی Number بر Index بخشپذیر است و لذا Index یک مقسوم‌علیه برای Number محسوب می‌شود و در این حالت به مقدار Counter یک واحد افزوده می‌شود. الگوریتم‌های دیگری نیز برای تشخیص اول بودن یک عدد موجود هستند. به عنوان مثال، می‌توان همین الگوریتم را به نحوی تغییر داد تا زمانی که تعداد مقسوم‌علیه‌های یافته‌شده (مقدار متغیر Counter) به بیش از دو رسید، ادامه بررسی مقسوم‌علیه‌ها خاتمه پذیرفته و با نمایش عبارت «*اول نیست*» الگوریتم پایان پذیرد.

مثال ۳-۳ الگوریتمی برای تشخیص مربع کامل بودن یک عدد صحیح ارائه دهید.

یادآوری: عدد مربع کامل، عددی است که جذر آن، عددی صحیح باشد.

مسئله: تشخیص مربع کامل بودن یک عدد.

ورودی: یک عدد صحیح مثلاً Number

خروجی: نمایش پیغام «*مربع کامل است*» اگر Number مربع کامل باشد و «*مربع کامل نیست*» اگر مربع کامل نباشد.

الگوریتم حل:

گام اول: شروع

گام دوم: دریافت عدد Number از کاربر.

گام سوم: مقدار متغیر Index را برابر صفر قرار بده.

گام چهارم: اگر Index × Index < Number است به *گام پنجم* برو در غیر اینصورت به *گام ششم* برو.

گام پنجم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو.

گام ششم: اگر Index × Index = Number است، پیغام «*مربع کامل است*» را نمایش بده در غیر اینصورت پیغام «*مربع کامل نیست*» را چاپ کن.

گام هفتم: پایان

■

در این الگوریتم، شمارنده Index در ابتدا برابر صفر در نظر گرفته می‌شود. در هر مرحله از تکرار الگوریتم، مقدار Index × Index با Number بررسی می‌شود. اگر مقدار عبارت Index × Index از Number کوچکتر باشد با اضافه کردن یک واحد به مقدار Index کار را دنبال می‌کنیم. در صورتی که شرط برقرار نباشد از تکرار خارج می‌شویم. طبیعی است که زمانی از تکرار خارج خواهیم شد که مقدار Index×Index یا بزرگتر یا مساوی مقدار Number شده باشد. اگر Index×Index برابر Number باشد پس Number مربع کامل بوده و جذر آن نیز برابر Index می‌باشد، در غیر اینصورت Index×Index بزرگتر از Number بوده و Number مربع کامل نمی‌باشد و ادامه تکرار بی‌فایده است.

تحقیق الگوریتم فوق را برای دو عدد ۶۰ و ۸۱ مورد بررسی قرار دهید.

۲-۳ فلوچارت[۱]

برای استاندارد نمودن و همچنین بصری نمودن[۲] الگوریتمها از اشکال هندسی استانداردی استفاده می‌شود که روند اجرای الگوریتم را نشان دهد. این اشکال، شامل: بیضی، مستطیل، متوازی‌الاضلاع و لوزی هستند. همچنین برای نشان دادن روند اجرا و توالی اعمال در فلوچارت‌ها از اتصال‌دهنده‌های جهت‌دار (بردارهای جهت‌دار) استفاده می‌شود. برای اجرای الگوریتم در فلوچارت کافی است روند اجرا را از طریق مسیر بردارهای جهت‌دار دنبال نماییم.

---

[۱] Flowchart

[۲] Visualization

جهت یکسان‌سازی و استانداردنمودن نمادهای مورد استفاده در فلوچارت‌ها از قراردادهای زیر استفاده خواهیم نمود.

قراردادهای شبه کد برای استفاده در فلوچارت‌ها

ـ برای انتساب یک مقدار یا نتیجه یک محاسبه به یک متغیر از علامت ← استفاده خواهیم نمود.

ـ برای بررسی تساوی مقدار دو متغیر یا عبارت از علامت == استفاده خواهیم نمود.

ـ برای شروع از لغت Start و برای پایان از لغت Finish استفاده خواهیم نمود.

ـ برای دریافت مقادیر متغیرها از کاربر، از لغت Input و برای نمایش مقادیر در خروجی از لغت Output استفاده خواهیم نمود.

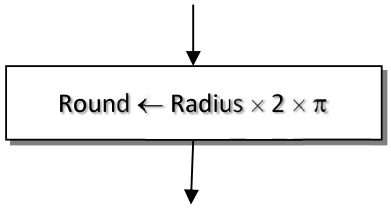
حالات شروع و پایان

برای نمایش حالات شروع و پایان یک فلوچارت از شکل بیضی استفاده می‌کنیم. در داخل بیضی حالت شروع، از لغت Start و در داخل بیضی حالت پایان از لغت Finish استفاده می‌کنیم.

شکل ۳ـ۱ نمادهای شروع و پایان در فلوچارت‌ها

محاسبات و جایگذاری و انتساب

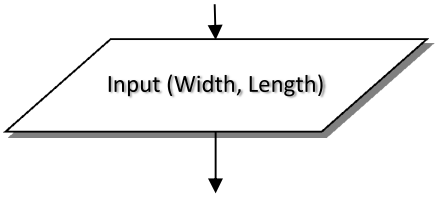
برای نشان دادن محاسبات و جایگذاری‌ها از شکل مستطیل استفاده می‌کنیم. عبارت محاسباتی در داخل مستطیل نوشته می‌شود و برای جایگذاری مقدار یا حاصل یک عبارت محاسباتی در یک متغیر از علامت ← استفاده می‌کنیم. در چنین حالاتی، حتماً متغیر در سمت چپ عبارت جایگذاری قرار می گیرد و مقدار یا عبارتی که قرار است حاصل آن در متغیر قرار بگیرد، در سمت راست عبارت قرار می گیرد.



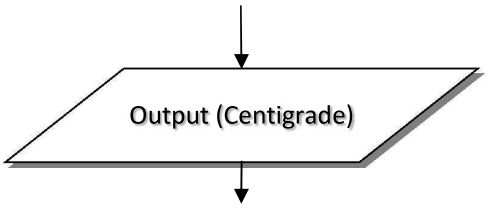
شکل ۳ـ۲ نماد محاسبات و جایگذاری در فلوچارت‌ها

ورودی و خروجی در فلوچارت

برای نمایش ورودی/خروجی در فلوچارت‌ها از شکل متوازی‌الاضلاع استفاده می‌کنیم. در داخل متوازی‌الاضلاع، اگر به عنوان ورودی مورد استفاده قرار گیرد لغت Input قرار می‌گیرد و بعد از آن نام متغیر یا متغیرهایی که قصد داریم آنها را از کاربر دریافت نماییم را ذکر می‌کنیم و چنانچه متوازی‌الاضلاع به عنوان خروجی مورد استفاده قرار گیرد از لغت Output در داخل آن و در ادامه پیغام یا عبارت یا متغیر خروجی برای نمایش قرار می‌گیرد. اگر از متوازی‌الاضلاع برای نمایش پیغامی استفاده می‌کنیم آن را در داخل گیومه و در غیر اینصورت برای حالتی که قصد نمایش مقدار یک متغیر یا حاصل یک عبارت را داریم خود آنها را عیناً ذکر می‌کنیم.



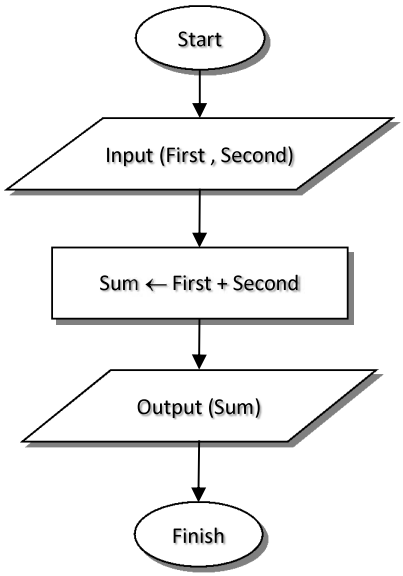
شکل ۳ـ۳ متوازی‌الاضلاع به عنوان ورودی



شکل ۳ـ۴ متوازی‌الاضلاع به عنوان خروجی

در ادامه فصل، مثال‌های متعددی برای آشنایی بیشتر شما با بیان دقیق مسئله و رسم فلوچارت ارائه گردیده است. در هر یک از مثال‌ها، الگوریتم حل مسئله به صورت بیان دقیق مسئله، استخراج ورودی و خروجی مسئله و فلوچارت حل آن برای آشنایی و فهم دقیق مفهوم الگوریتم و فلوچارت بیان شده است.

مثال ۳ـ۴ فلوچارتی رسم نمایید که دو عدد را از کاربر دریافت کرده و مجموع آنها را محاسبه و نمایش دهد.



مسئله: محاسبه مجموع دو عدد دریافتی از کاربر

ورودی: دو عدد (First و Second)

خروجی: حاصل جمع اعداد دریافتی از کاربر

الگوریتم حل:

گام اول: شروع.

گام دوم: دو عدد First و Second را از کاربر دریافت کن.

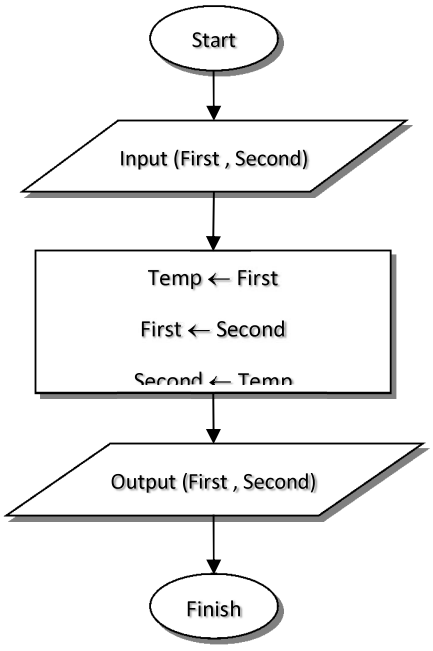
گام سوم: First + Second را محاسبه و در متغیر Sum قرار بده.

گام چهارم: مقدار Sum را نمایش بده.

گام پنجم: پایان.

■

مثال ۳–۵ فلوچارتی رسم نمایید که دو عدد را از کاربر دریافت کرده و مقادیر آنها را با استفاده از یک متغیر کمکی جابجا کند.



مسئله: جابجا نمودن مقادیر موجود در دو متغیر

ورودی: دو عدد (First و Second)

خروجی: مقادیر دو متغیر دریافتی (جابجا شده)

الگوریتم حل:

گام اول: شروع.

گام دوم: اعداد First و Second را از کاربر دریافت کن.

گام سوم: مقدار First را در متغیر Temp قرار بده.

گام چهارم: مقدار Second را در متغیر First قرار بده.

گام پنجم: مقدار Temp را در متغیر Second قرار بده.

گام ششم: مقادیر First , Second را چاپ کن.

گام هفتم: پایان.

■

مثال ۳,۶ فلوچارتی رسم نمایید که شعاع دایره‌ای را دریافت نموده و محیط و مساحت آن را محاسبه و نمایش دهد.

مسئله: محاسبه و نمایش محیط و مساحت دایره

ورودی: شعاع دایره

خروجی: محیط و مساحت دایره

الگوریتم حل:

گام اول: شروع

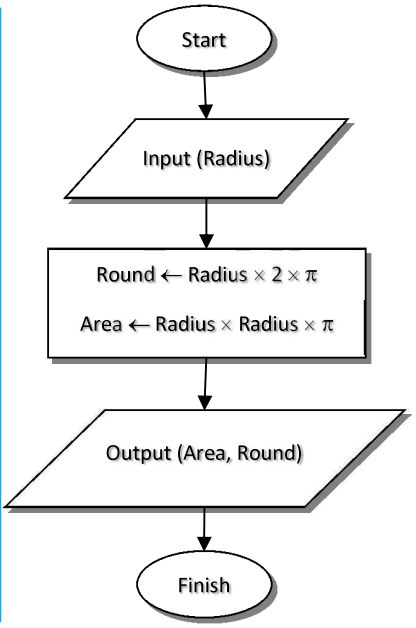
گام دوم: شعاع دایره را از کاربر دریافت کن و در متغیر Radius ذخیره کن.

گام سوم: محیط دایره را با فرمول $\pi \times 2 \times$ Radius محاسبه کرده و در متغیر Round ذخیره کن

گام چهارم: مساحت دایره را با فرمول $\pi \times$ Radius $\times$ Radius محاسبه کرده و در متغیر Area ذخیره کن.

گام پنجم: مقادیر Area, Round را چاپ کن.

گام ششم: پایان



در فلوچارت بالا، Radius متغیری برای نگهداری شعاع دایره است که از کاربر دریافت خواهد گردید. متغیر Round برای نگهداری محیط و متغیر Area برای نگهداری مساحت بکار رفته‌اند.

مثال۳,۷ فلوچارتی رسم نمایید که طول و عرض مستطیلی را دریافت نموده و محیط و مساحت آن را محاسبه و نمایش دهد.

مسئله: محاسبه محیط و مساحت مستطیل

ورودی: طول و عرض مستطیل

خروجی: محیط و مساحت مستطیل

الگوریتم حل:

گام اول: شروع

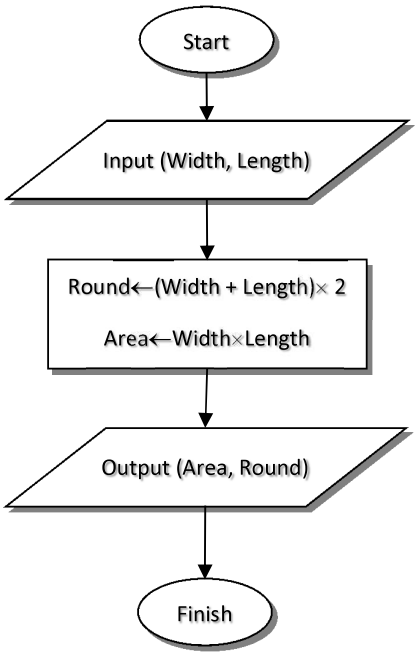
گام دوم: طول و عرض مستطیل را از کاربر دریافت کن و در متغیرهای Width , Length ذخیره کن.

گام سوم: محیط مستطیل را با فرمول $(\text{Width} + \text{Length}) \times 2$ محاسبه کرده و در متغیر Round ذخیره کن.

گام چهارم: مساحت مستطیل را با فرمول Width $\times$ Length محاسبه کرده و در متغیر Area ذخیره کن.

گام پنجم: مقادیر Area, Round را چاپ کن.

گام ششم: پایان



مثال ۸,۳ فلوچارتی رسم نمایید که دمای یک جسم را بر حسب فارنهایت دریافت نموده و آن را به سانتی‌گراد تبدیل نموده و نمایش دهد.

مسئله: تبدیل دمای فارنهایت به سانتی‌گراد

ورودی: دما بر حسب فارنهایت

خروجی: دما بر حسب سانتی‌گراد

الگوریتم حل:

گام اول: شروع

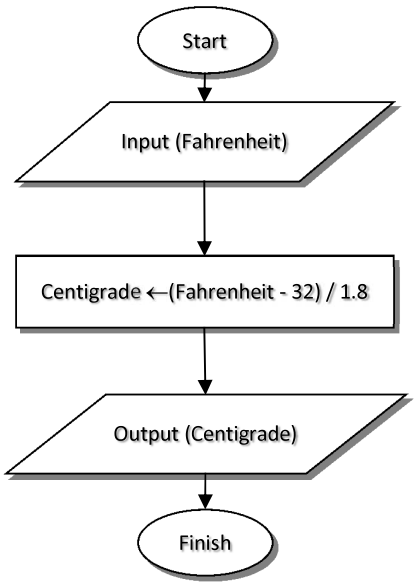
گام دوم: دما را بر حسب فارنهایت از کاربر دریافت کن و در متغیر Fahrenheit ذخیره کن

گام سوم: با استفاده از رابطه (Fahrenheit - 32) / 1.8 دما را به سانتی گراد تبدیل و در متغیر Centigrade ذخیره کن.

گام چهارم: مقدار Centigrade را چاپ کن.

گام پنجم: پایان

■



در فلوچارت بالا، Fahrenheit متغیری برای نگهداری دما بر حسب فارنهایت است که از کاربر دریافت خواهد گردید. متغیر Centigrade برای نگهداری دما بر حسب سانتی‌گراد بکار رفته است.

مثال ۹,۳ فلوچارتی رسم نمایید که دو عدد را دریافت نموده و میانگین آنها را حساب کند .

مسئله: محاسبه و نمایش میانگین دو عدد

ورودی: دو عدد

خروجی: میانگین دو عدد دریافتی

الگوریتم حل:

گام اول: شروع

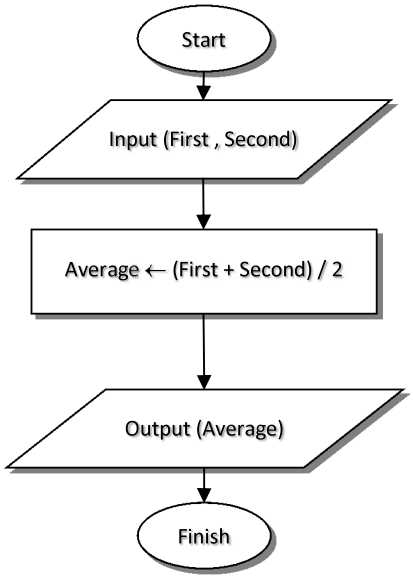
گام دوم: اعداد First , Second از کاربر دریافت کن.

گام سوم: میانگین دو عدد را با فرمول (First + Second)/2 محاسبه کرده و در متغیر Average ذخیره کن.

گام چهارم: مقدار Average را چاپ کن.

گام پنجم: پایان

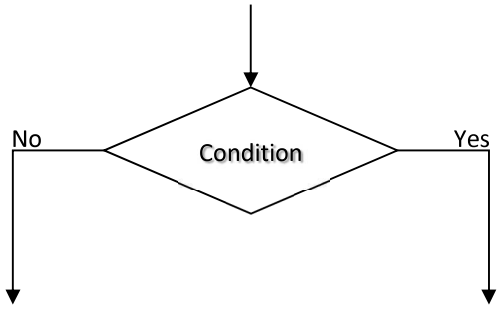
■



۳-۲-۱ شرطهای تصمیم

اغلب اوقات در حل مسائل، نیاز است که بر اساس شرطی خاص، عمل مشخصی را انجام دهیم. به عنوان مثال فرض کنید قصد داریم عددی را دریافت نموده و قدر مطلق آن را چاپ نماییم. لازم است مقدار عدد دریافتی را بررسی نماییم تا در صورتی که عدد دریافتی مقداری منفی داشته باشد، قرینه آن

و در غیر اینصورت خودش را به عنوان قدر مطلق نمایش دهیم. در چنین حالاتی، برای بررسی شرط در فلوچارت‌ها از شکل لوزی استفاده می‌کنیم. در هنگام استفاده از لوزی، شرط یا عبارت بررسی شونده را در داخل لوزی می‌نویسیم. این عبارت باید عبارتی منطقی باشد که جواب آن درست یا نادرست خواهد بود. پس با این توصیف دو خروجی از لوزی خواهیم داشت که یکی برای حالت درست (برقرار) بودن شرط و دیگری برای حالت نادرست (عدم برقراری) شرط می‌باشد.



شکل ۵,۳ نماد شرط در فلوچارت‌ها

مثال ۱۰,۳ فلوچارتی رسم نمایید که عددی را دریافت نموده و قدر مطلق[۱] آنرا محاسبه و نمایش دهد.

[۱] Absolute value

مسئله: نمایش قدرمطلق عدد دریافتی

ورودی: عدد Number

خروجی: قدرمطلق عدد Number

الگوریتم حل:

گام اول: شروع

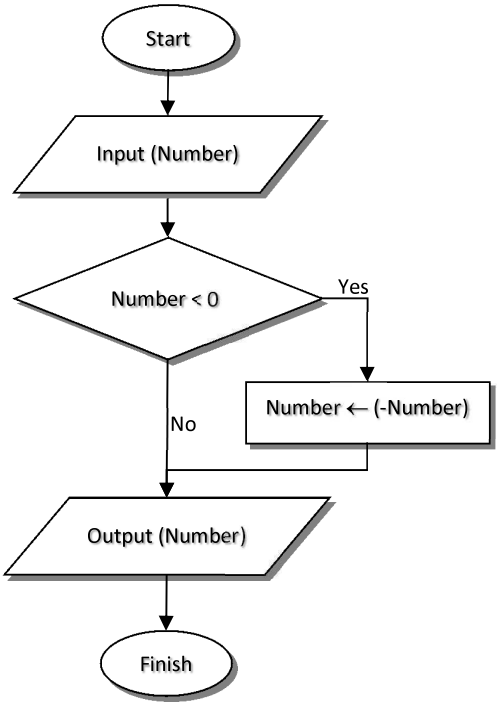
گام دوم: عدد Number را از کاربر دریافت کن

گام سوم: اگر Number از صفر کوچکتر باشد قرینه آن را در متغیر Number قرار بده

گام چهارم: مقدار Number را چاپ کن

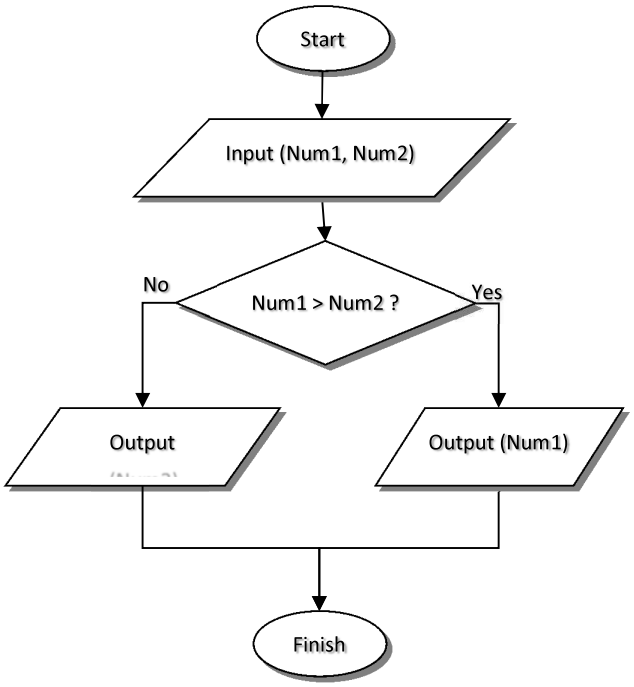
گام پنجم: پایان

■



در الگوریتم بالا، ابتدا یک عدد از کاربر دریافت و در متغیر Number قرار داده می‌شود. سپس بررسی می‌شود که عدد دریافتی (Number) از صفر کوچکتر است یا نه؟ این بررسی در فلوچارت توسط لوزی انجام می‌شود. اگر عدد دریافتی کوچکتر از صفر باشد، قرینه عدد Number در Number قرار می‌گیرد. (قرینه عدد منفی، مثبت خواهد بود.) اگر Number منفی نباشد، نتیجه بررسی در لوزی نادرست بوده و روند اجرا مستقیماً به گام چهارم رفته و مقدار Number نمایش داده خواهد شد. فلوچارت بالا را یکبار برای عدد ۱۰- و یکبار برای عدد ۲۵ مورد بررسی قرار می‌دهیم. اگر مقدار ۱۰- از ورودی برای عدد Number وارد شود، نتیجه بررسی شرط در لوزی درست و خروجی Yes دنبال می‌شود و کنترل به مستطیل ((Number←-(Number)) منتقل می‌شود. در داخل مستطیل، قرینه عدد ۱۰- در Number قرار می‌گیرد یعنی مقدار Number به ۱۰ تغییر می‌یابد و ادامه اجرا به متوازی الاضلاع برای نمایش مقدار Number یا همان ۱۰ منتقل می‌شود. در حالتی که ورودی عدد ۲۵ باشد، خروجی شرط (لوزی) به سمت No هدایت شده و مستقیما و بدون تغییر مقدار ۲۵ نمایش داده می‌شود.

مثال ۱,۳ ۱ فلوچارتی رسم نمایید که دو عدد دریافت نموده و عدد بزرگتر را نمایش دهد.



مسئله: نمایش عدد بزرگتر از بین دو عدد دریافتی

ورودی: دو عدد (Num1 و Num2)

خروجی: نمایش عدد بزرگتر

الگوریتم حل:

گام اول: شروع

گام دوم: اعداد Num1 و Num2 را از کاربر دریافت کن.

گام سوم: اگر Num1 از Num2 بزرگتر باشد مقدار Num1 را به عنوان عدد بزرگتر چاپ کن و در غیر اینصورت، مقدار Num2 را به عنوان عدد بزرگتر چاپ کن.

گام چهارم: پایان

■

مثال ۱,۳ ۲ فلوچارتی رسم نمایید که دو عدد را دریافت نموده و آنها را به ترتیب ابتدا بزرگتر و سپس کوچکتر نمایش دهد.

مسئله: مرتب سازی دو عدد

ورودی: دو عدد

خروجی: نمایش اعداد به ترتیب نزولی

الگوریتم حل:

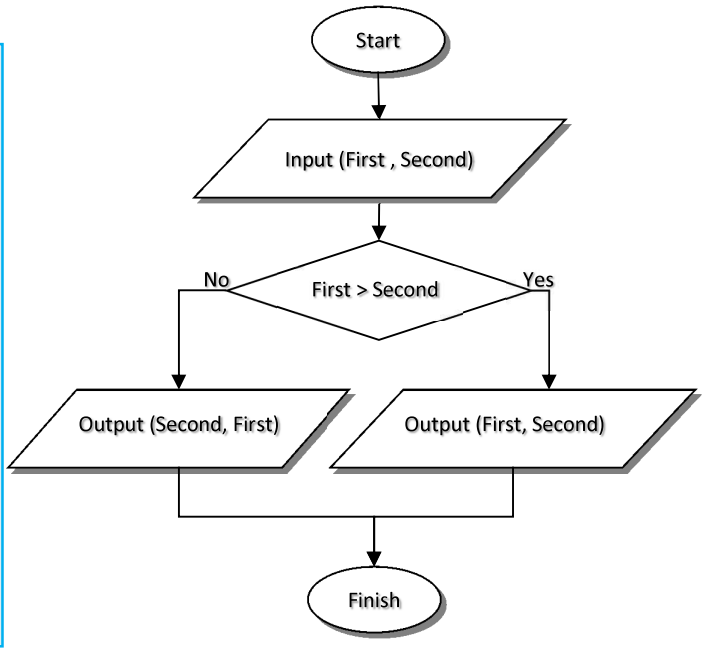
گام اول: شروع

گام دوم: اعداد First و Second را از کاربر دریافت کن

گام سوم: اگر First بزرگتر از Second است ابتدا مقدار First و سپس مقدار Second را چاپ کن، در غیر اینصورت ابتدا مقدار Second و سپس مقدار First را چاپ کن

گام چهارم: پایان

■



مثال ۳-۱۳ فلوچارتی رسم نمایید که عددی را دریافت نموده و قدرمطلق آن را محاسبه و نمایش دهد.

مسئله: نمایش قدرمطلق یک عدد

ورودی: عدد Number

خروجی: قدر مطلق عدد Number

الگوریتم حل:

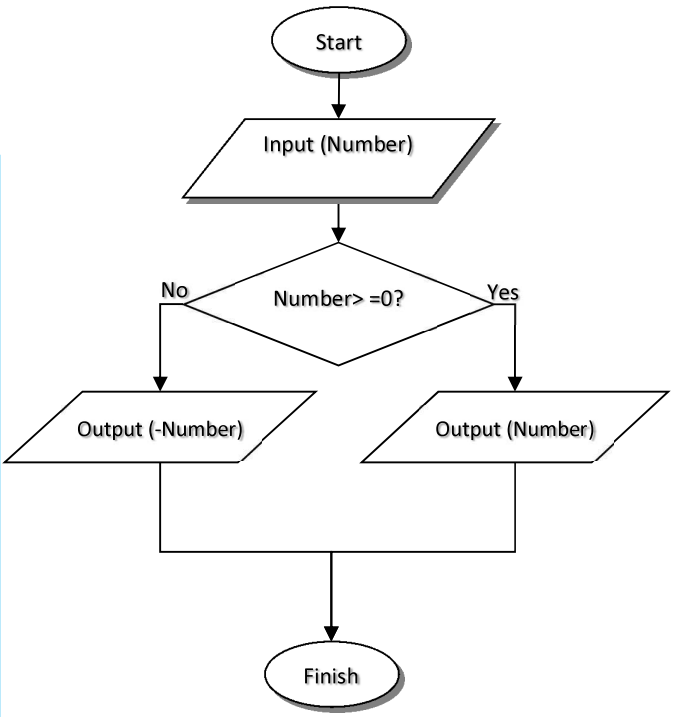
گام اول: شروع

گام دوم: عددNumber را از کاربر دریافت کن

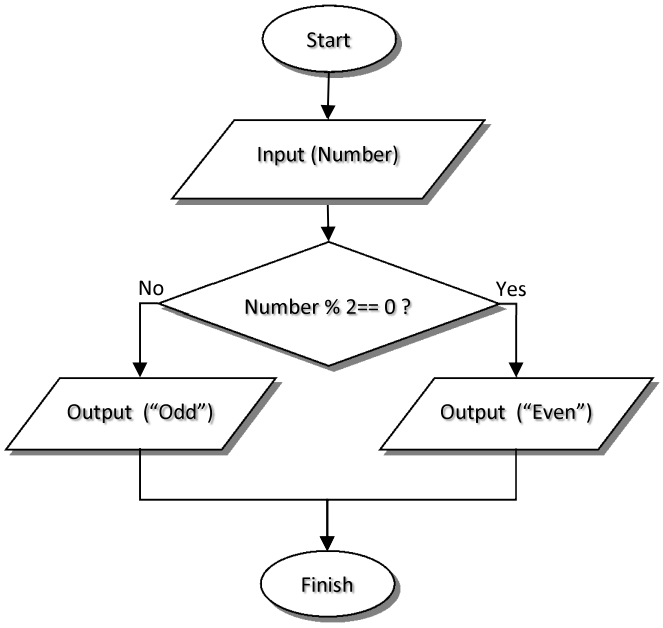
گام سوم: اگر Number بزرگتر یا مساوی صفر باشد مقدار Number را نمایش بده و در غیر اینصورت قرینه مقدار آن (Number-) را نمایش بده.

گام چهارم: پایان

■



مثال ۳-۱۴ فلوچارتی رسم نمایید که عددی را دریافت نموده و زوج یا فرد بودن آن را تشخیص دهد .



مسئله: تشخیص زوج یا فرد بودن عدد

ورودی: عدد Number

خروجی: پیغام "*زوج*" یا "*فرد*"

الگوریتم حل:

گام اول: شروع

گام دوم: دریافت عدد Number

گام سوم: اگر باقیمانده تقسیم Number بر ۲ مساوی صفر است پیغام «*عدد زوج است.*» را نمایش بده و در غیر اینصورت پیغام «*عدد فرد است.*» را نمایش بده.

گام چهارم: پایان

■

۳-۲-۲ حلقه‌های تکرار

حلقه‌های تکرار، را می‌توان اجزاء جدایی‌ناپذیر الگوریتمها و برنامه‌های کامپیوتری دانست. حلقه‌های تکرار در الگوریتم‌هایی که به تکرار مجموعه‌ای از دستورالعمل‌ها به تعدادی مشخص یا تکرار تا محقق شدن شرطی خاص نیاز دارند، کاربرد دارند. به عنوان نمونه فرض کنید می‌خواهید الگوریتمی طرح نمایید که دو عدد صحیح را دریافت نموده و بزرگترین مقسوم علیه مشترک آن دو را به دست آورده و نمایش دهد. *روش اقلیدسی*[۱] شاید ساده‌ترین روشی باشد که بتوان آن را به یک الگوریتم تبدیل نمود.

مثال ۳-۱۵ الگوریتمی برای محاسبه بزرگترین مقسوم‌علیه مشترک دو عدد دریافتی بنویسید.

مسئله: با در دست داشتن دو عدد صحیح، بزرگترین مقسوم علیه مشترک آن دو را محاسبه و نمایش دهد.

یادآوری: بزرگترین مقسوم علیه مشترک دو عدد، بزرگترین عدد صحیح مثبتی است که هر دو عدد بر آن بخشپذیر باشند.

ورودی: دو عدد

خروجی: بزرگترین مقسوم علیه مشترک دو عدد دریافتی

الگوریتم اقلیدس برای حل:

گام اول: شروع

گام دوم: دو عدد از کاربر دریافت نموده و آنها را در m و n قرار دهید.

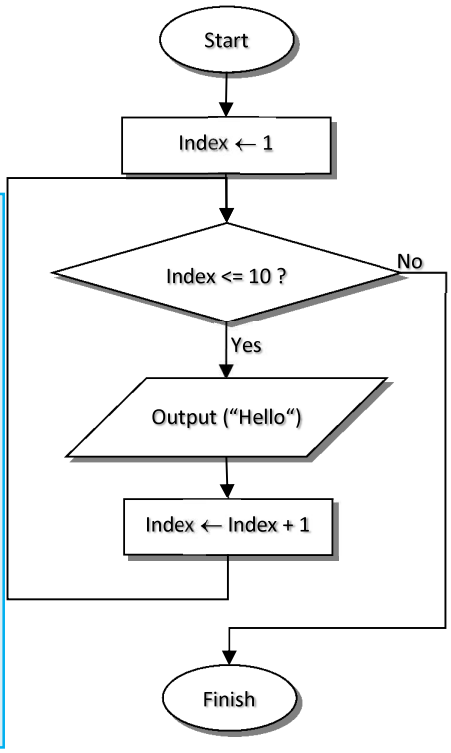
گام سوم: m را بر n تقسیم نموده و باقیمانده را در r قرار دهید.

[۱] Euclid's Method

گام چهارم: آیا r=0 است. الگوریتم خاتمه می یابد. n جواب است. به *گام ششم* بروید.
گام پنجم: n را در m و r را در n قرار دهید (m←n , n←r) و به *گام سوم* برگردید.
گام ششم: n را نمایش بده.
گام هفتم: پایان
■

می‌خواهیم الگوریتم فوق را برای دو عدد ورودی مفروض ۴۸ و ۹۰ اجرا نمائیم. پس از شروع، در گام دوم با وارد نمودن عدد ۴۸ برای m و ۹۰ برای n به گام سوم می‌رسیم. در گام سوم m را بر n تقسیم می‌کنیم و باقیمانده تقسیم را که عدد ۴۸ می‌باشد در r قرار می‌دهیم. گام چهارم بررسی می‌کند که r برابر صفر است یا نه؟ از آنجا که این شرط برقرار نیست، پس دستور مقابل آن را اجرا نمی‌کنیم و به گام بعدی (گام پنجم) می‌رویم. در گام پنجم مقدار n (۹۰) را در m و r (۴۸) را در n قرار داده و به گام سوم برمی‌گردیم. در گام سوم دوباره m را بر n تقسیم می‌کنیم. باقیمانده این تقسیم (۴۸ ÷ ۹۰) برابر ۴۲ خواهد بود که آن را در r قرار می‌دهیم و بررسی می‌کنیم که آیا صفر است یا نه؟ چون مقدار r هنوز صفر نشده، مقدار n (۴۸) را در m و مقدار r (۴۲) را در n قرار داده به گام سوم برمی‌گردیم. دوباره m (۴۸) را بر n (۴۲) تقسیم نموده و باقیمانده تقسیم (۶) را در r قرار می دهیم. چون r برابر صفر نیست، مقدار n (۴۲) را در m و مقدار r (۶) را در n قرار داده و به گام سوم برمی‌گردیم. در گام سوم باقیمانده تقسیم m بر n (۶ ÷ ۴۲) را محاسبه می‌کنیم (صفر) و آن را در r قرار می‌دهیم. در گام چهارم بررسی می‌کنیم که مقدار r برابر صفر است؟ بله. پس جواب مقدار n است که در حال حاضر مقدار n برابر ۶ می باشد. به گام ششم رفته مقدار n (عدد ۶) را به عنوان حاصل بزرگترین مقسوم علیه مشترک دو عدد ۴۹ و ۹۰ نمایش داده و به گام پایان می‌رویم. الگوریتم خاتمه می‌یابد. این الگوریتم روند اجرای حلقه را نشان می‌دهد. این حلقه وابسته به شرط بود یعنی تا زمانی که مقدار r برابر صفر نشده بود یک سلسله اعمال تکراری اجرا می‌گردید.

مثال ۱۶,۳ فلوچارتی رسم نمایید که ۱۰ بار عبارت " Hello " را چاپ کند.



مسئله: نمایش ۱۰ بار عبارت ”Hello“

ورودی: -

خروجی: ۱۰ بار عبارت Hello

الگوریتم حل:

گام اول: شروع

گام دوم: شمارشگر Index را مساوی یک قرار بده

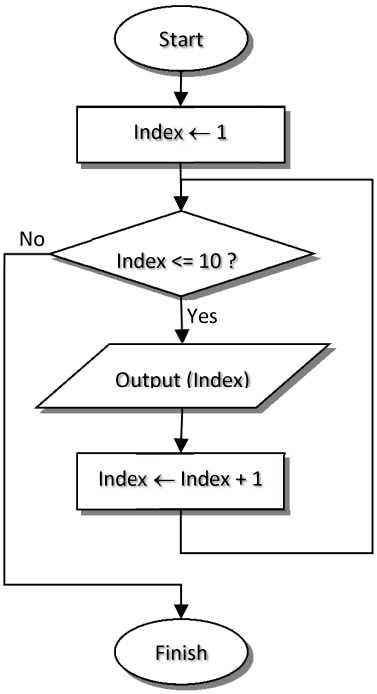
گام سوم: اگر Index کوچکتر یا مساوی ۱۰ است عبارت «Hello» را چاپ کن در غیر اینصورت به *گام پنجم* برو

گام چهارم: مقدار Index را یک واحد افزایش بده و به *گام سوم* برو

گام پنجم: پایان

■

مثال ۳-۱۷ فلوچارتی رسم نمایید که اعداد از یک تا ده را چاپ کند .



مسئله: چاپ اعداد ۱ تا ۱۰

ورودی: -

خروجی: اعداد ۱ تا ۱۰

الگوریتم حل:

گام اول: شروع

گام دوم: شمارشگر Index را مساوی یک قرار بده

گام سوم: اگر Index کوچکتر یا مساوی ۱۰ است مقدار Index را چاپ کن در غیر اینصورت به *گام پنجم* برو

گام چهارم: مقدار Index را یک واحد افزایش بده و به *گام سوم* برو

گام پنجم: پایان

■

مثال ۱۸٫۳ فلوچارتی رسم نمایید که عددی را دریافت نموده و اعداد از آن عدد تا یک را نمایش دهد.

مسئله: نمایش اعداد از Number تا یک

ورودی: عدد Number

خروجی: اعداد از Number تا یک

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن

گام سوم: مقدار Index را مساوی Number قرار بده

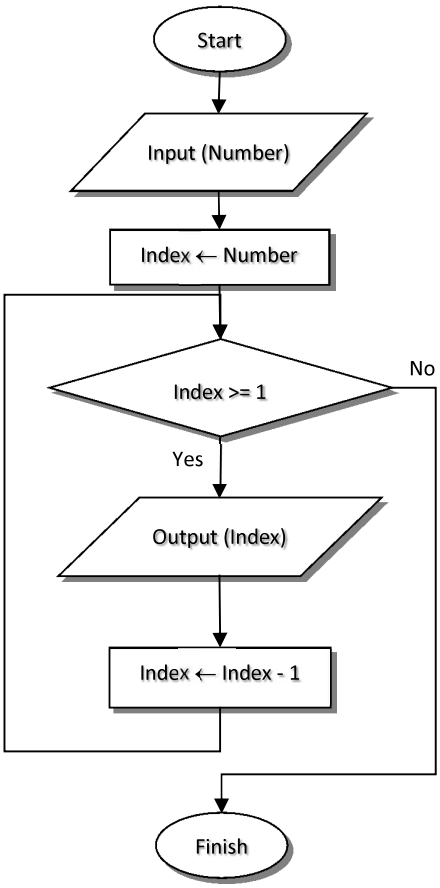
گام چهارم: اگر Index بزرگتر یا مساوی ۱ است مقدار Index را چاپ کن و به *گام ششم* برو

گام پنجم: اگر Index کوچکتر از ۱ است به *گام هفتم* برو

گام ششم: یک واحد مقدار Index را کاهش بده به *گام چهارم* برو

گام هفتم: پایان

■



در فلوچارت بالا، Number متغیری برای نگهداری عددی است که از کاربر دریافت خواهد گردید. متغیر Index برای نگهداری شمارشگر برنامه به کار می‌رود. در ابتدا Index را برابر مقدار Number قرار می دهیم و بررسی می‌کنیم که بزرگتر یا مساوی یک است یا نه؟ دلیل این کار این است که اگر Index بزرگتر یا مساوی یک باشد، بدین معنی است که عمل نمایش لیست اعداد باید ادامه یابد ولی چنانچه Index از یک کوچکتر شده است یعنی تک تک اعداد از Number تا یک چاپ شده و حلقه تکرار باید پایان پذیرد.

سؤال؟ اگر موقع ورود عدد Number، مقدار صفر یا مقداری منفی را وارد نمائیم، چه اتفاقی خواهد افتاد؟

مثال ۱۹,۳ فلوچارتی رسم نمایید که عددی را دریافت نموده و مقسوم علیه‌های آن را به ترتیب از کوچک به بزرگ نمایش دهد.

مسئله: نمایش مقسوم علیه‌های یک عدد به صورت صعودی

ورودی: عدد Number

خروجی: مقسوم علیه های Number

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن.

گام سوم: مقدار Index رامساوی ۱ قرار بده.

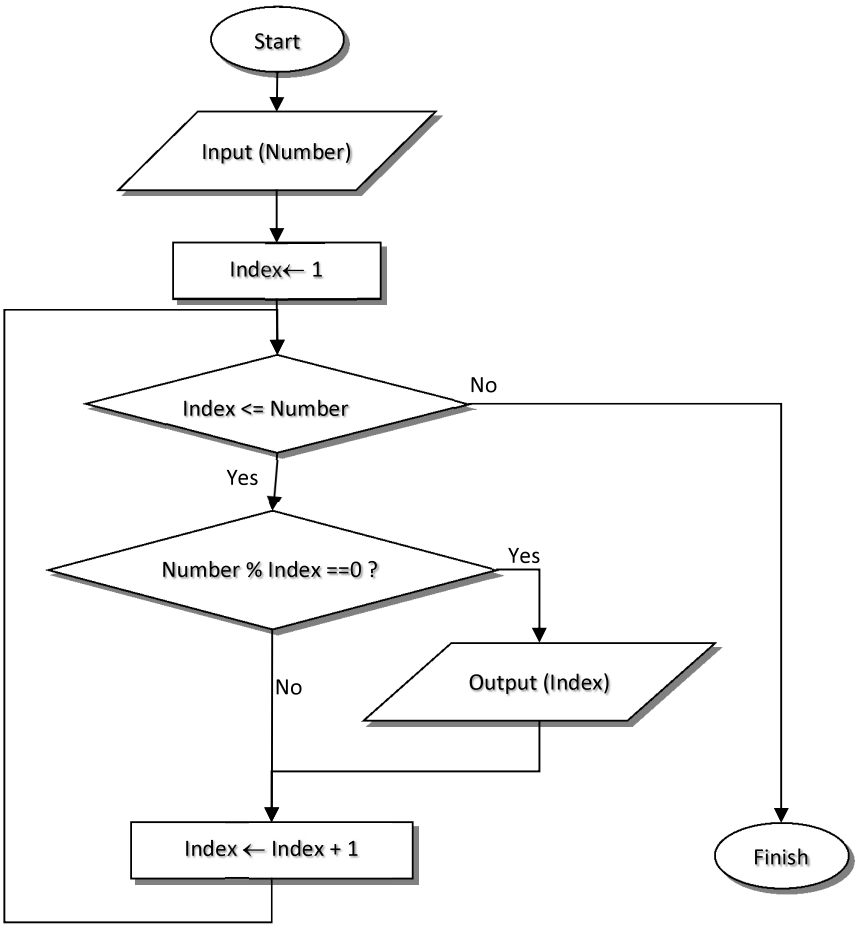
گام چهارم: اگر Index بزرگتر از Number است، به *گام هفتم* برو.

گام پنجم: اگر باقیمانده تقسیم Number برIndex مساوی صفر است، Index را چاپ کن.

گام ششم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برگرد.

گام هفتم: پایان

■



مثال ۲.۳ فلوچارتی رسم نمایید که عددی را دریافت نموده و تعداد مقسوم علیه های آن را نمایش دهد.

مسئله: نمایش تعداد مقسوم علیه‌های یک عدد

ورودی: عدد Number

خروجی: تعداد مقسوم علیه‌های Number

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن.

گام سوم: مقدار Index را مساوی ۱و مقدار Counter را مساوی صفر قرار بده.

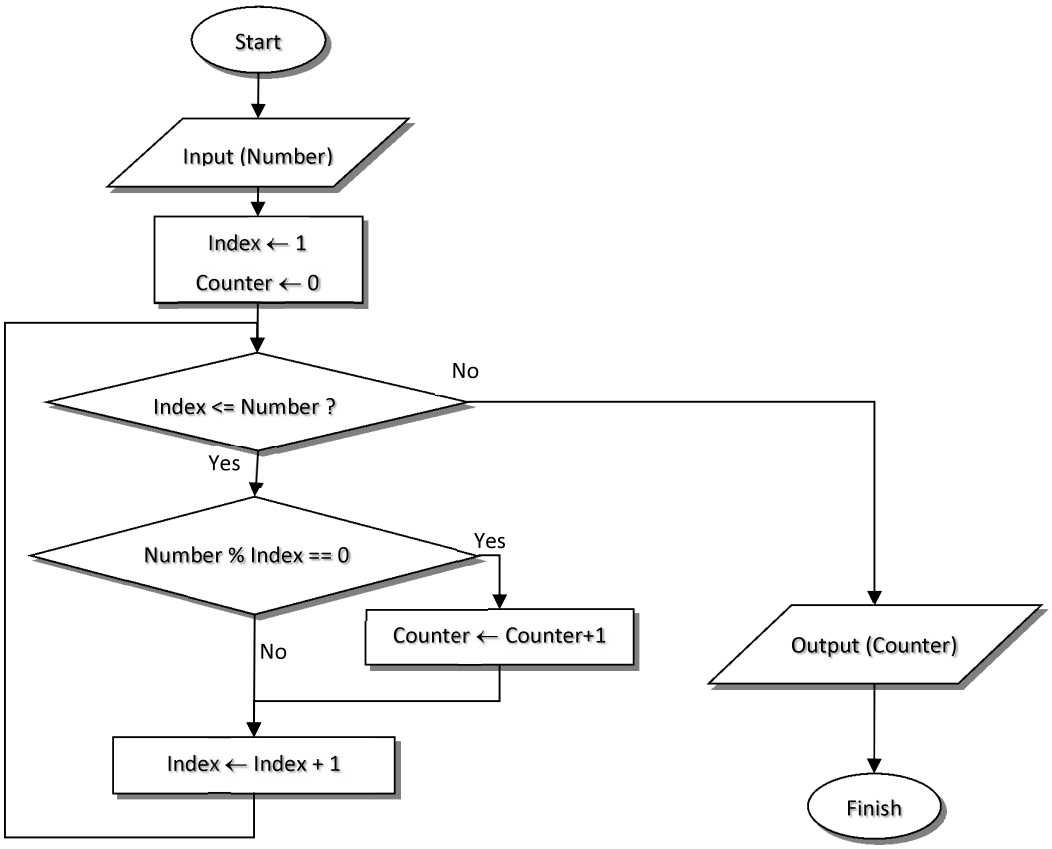
گام چهارم: اگر Index کوچکتر یا مساوی Number نیست، به *گام هفتم* برو.

گام پنجم: اگر باقیمانده تقسیم Number برIndex مساوی صفر است مقدار Counter را یک واحد افزایش بده.

گام ششم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو.

گام هفتم: مقدار Counter را چاپ کن.

گام هشتم: پایان



مثال ۲ ۱,۳ فلوچارتی رسم نمایید که عددی را دریافت نموده و مجموع مقسوم‌علیه‌های آن را نمایش دهد.

مسئله: محاسبه مجموع مقسوم علیه‌های یک عدد

ورودی: عدد Number

خروجی: مجموع مقسوم علیه‌های عدد Number

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن.

گام سوم: مقدار Index را مساوی ۱و مقدار Sum را مساوی صفر قرار بده.

گام چهارم: اگر Index کوچکتر یا مساوی Number نیست، به *گام هفتم* برو.

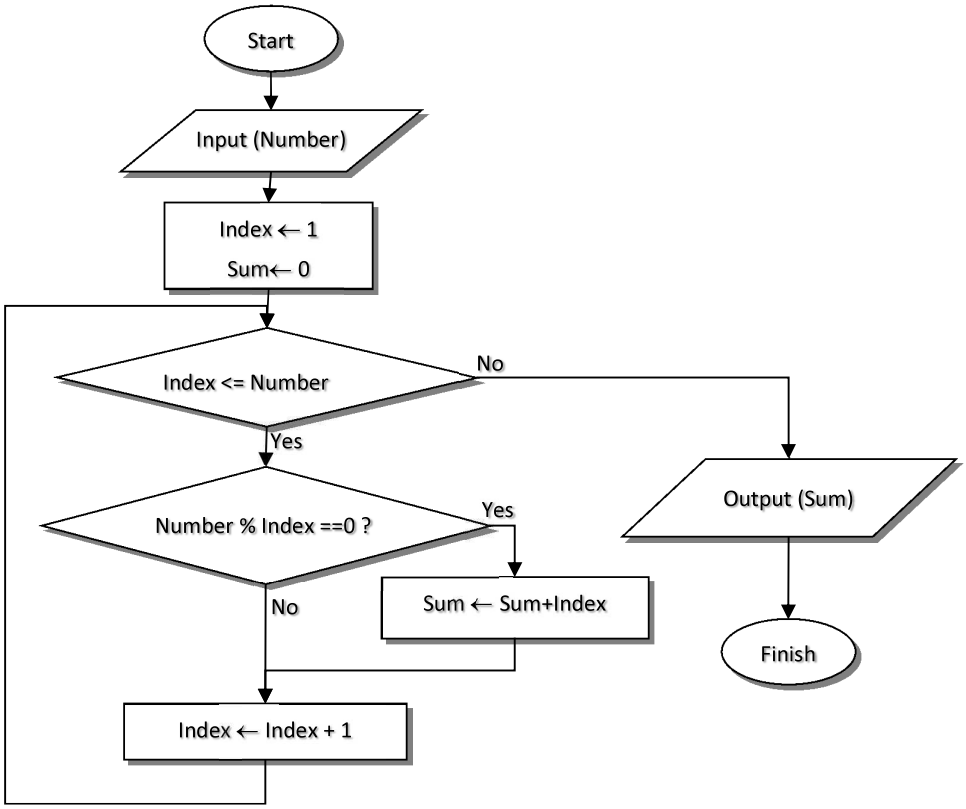
گام پنجم: اگر باقیمانده تقسیم Number برIndex مساوی صفر است مقدار Sum را به اندازه Index افزایش بده.

گام ششم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو.

گام هفتم: مقدار Sum را نمایش بده.

گام هشتم: پایان

■



مثال ۲٫۳ فلوچارتی رسم کنید که عددی را دریافت نموده و تعیین کند آن عدد اول است یا نه؟

مسئله: تشخیص اول بودن عدد

ورودی: عدد Number

خروجی: پیغام «*اول است*» یا «*اول نیست*»

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن.

گام سوم: مقدار Index را مساوی یک و مقدار Counter را مساوی صفر قرار بده.

گام چهارم: اگر Index کوچکتر یا مساوی Number نیست به *گام هفتم* برو.

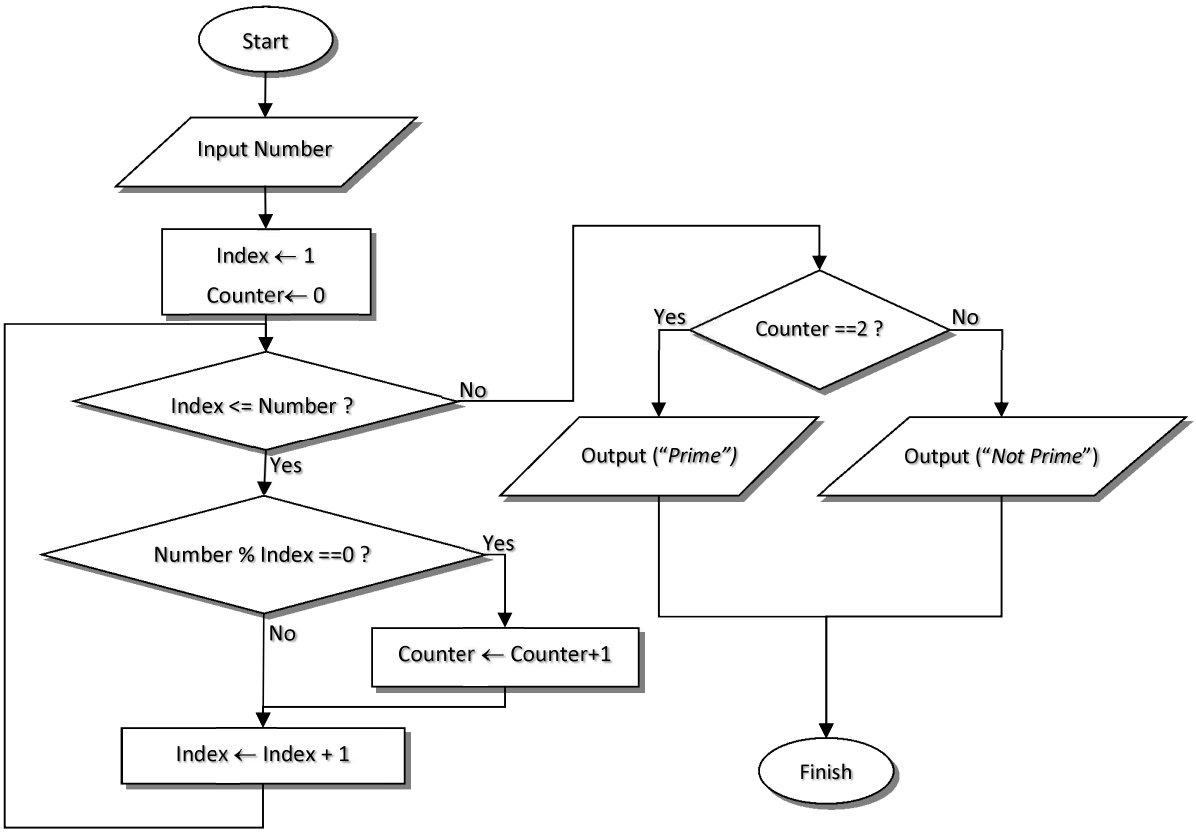
گام پنجم: اگر باقیمانده تقسیم Number بر Index مساوی صفر است به مقدار Counter یک واحد اضافه کن.

گام ششم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو.

گام هفتم: اگر مقدارCounter برابر دو است، پیغام «*عدد اول است*» و در غیر اینصورت پیغام «*عدد اول نیست*» را نمایش بده.

گام هشتم: پایان

■



مثال ۲۳٫۳ فلوچارتی رسم کنید که عددی را دریافت نموده و تعیین کند آن عدد مربع کامل است یا نه؟

مسئله: تعیین مربع کامل بودن عدد

ورودی: عدد Number

خروجی: پیغام مربع کامل یا غیر مربع کامل

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن.

گام سوم: مقدار Index را مساوی صفر قرار بده.

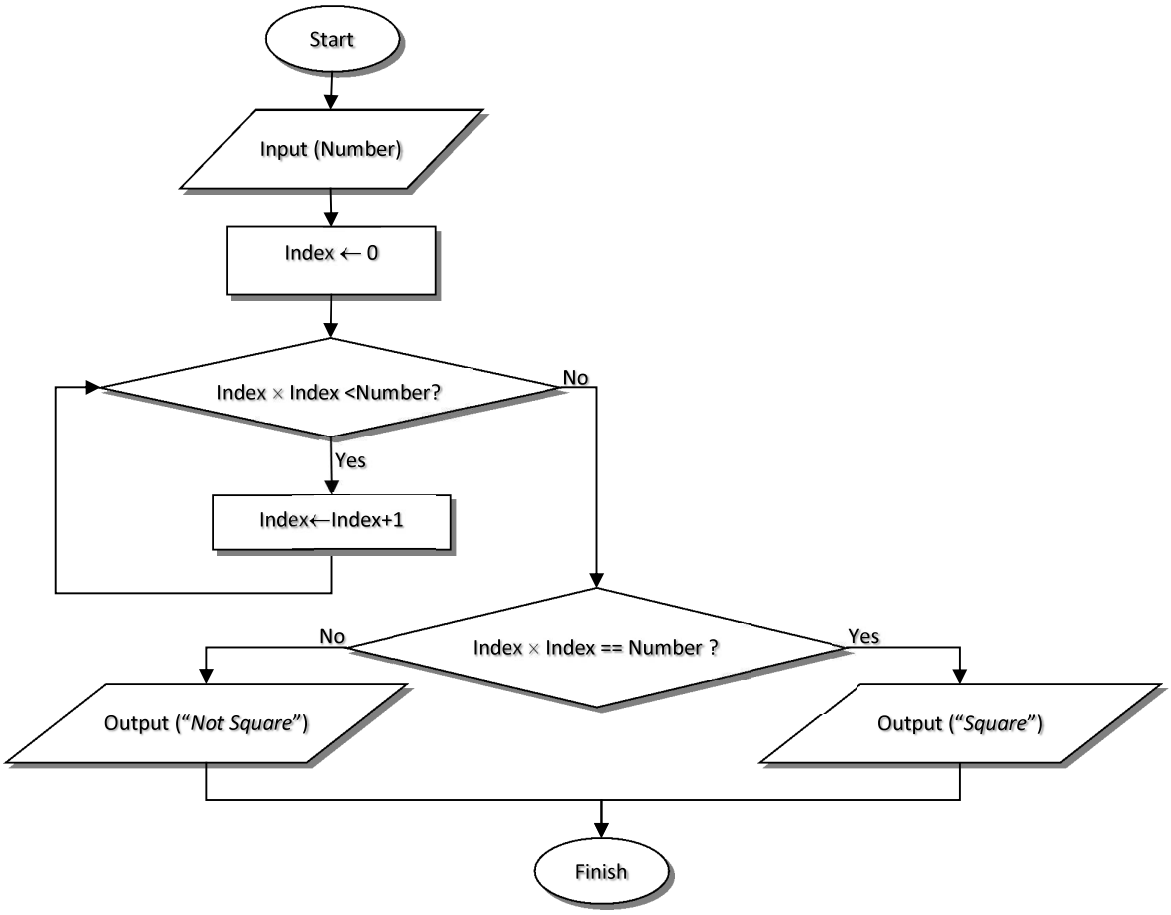
گام چهارم: اگر Index × Index < Number نیست، به *گام ششم* برو.

گام پنجم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برگرد.

گام ششم: اگر Index × Index == Number است، «*عدد مربع کامل است*» را نمایش بده و در غیر اینصورت «*عدد مربع کامل نیست*» را نمایش بده.

گام هفتم: پایان

■


Start
Input (Number)
Index ← 0
Index × Index <Number?
No
Yes
Index←Index+1
Index × Index == Number ?
No
Yes
Output (“Not Square”)
Output (“Square”)
Finish

مثال ۲۴,۳ فلوچارتی رسم کنید که عدد n را دریافت نموده و مجموع اعداد از یک تا n را چاپ کند

مسئله: محاسبه و نمایش مجموع اعداد از یک تا Number

ورودی: عدد Number

خروجی: مجموع اعداد از یک تا Number

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن.

گام سوم: مقدار Index را یک و Sum رامساوی صفر قرار بده.

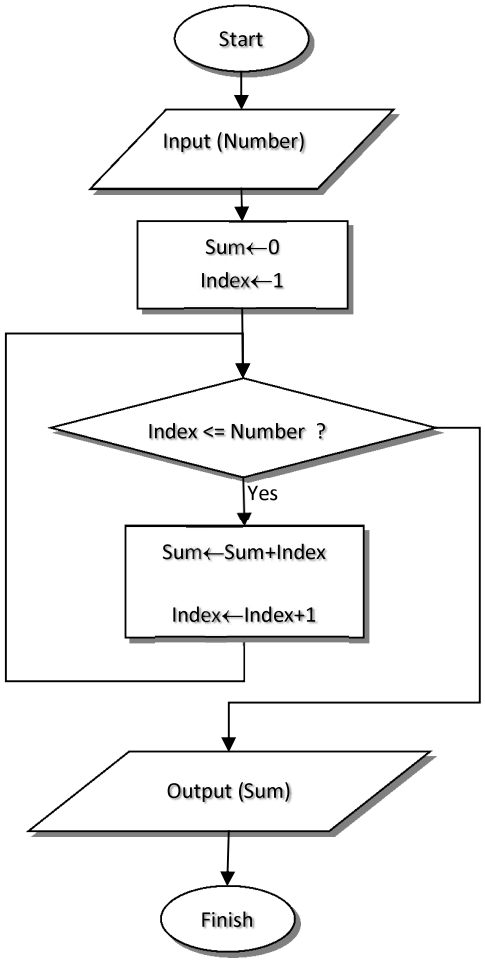
گام چهارم: اگر Index کوچکتر یا مساوی Number نیست، به *گام ششم* برو.

گام پنجم: مقدار Index را به Sum اضافه کن و مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو.

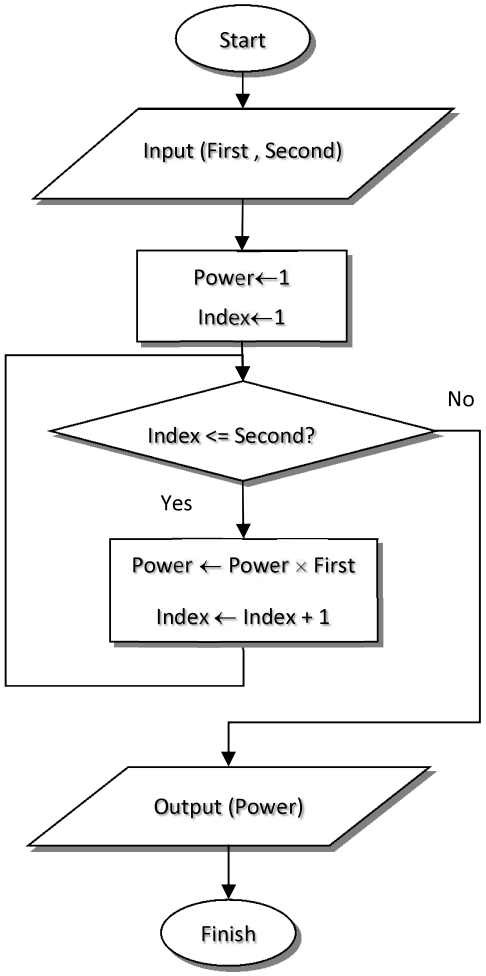
گام ششم: مقدار Sum را چاپ کن.

گام هفتم: پایان

■



مثال ۲۵,۳ فلوچارتی رسم کنید که دو عدد صحیح مثبت First , Second را دریافت نموده و First را به توان Second برساند



مسئله: محاسبه عدد به توان عدد دوم

ورودی: اعداد First و Second

خروجی: حاصل توان عدد First به توان Second

الگوریتم حل:

گام اول: شروع

گام دوم: اعداد First و Second را از کاربر دریافت کن

گام سوم: مقدار Index و Power را مساوی یک قرار بده

گام چهارم: اگر Index کوچکتر یا مساوی Second نیست، به گام ششم برو.

گام پنجم: مقدار First را در Power ضرب کن و در Power قرار بده. و مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو

گام ششم: مقدار Power را چاپ کن

گام هفتم: پایان

■

در فلوچارت بالا، قرار است عدد First را به توان Second برسانیم. می دانیم که برای این کار باید عدد First را به تعداد Second بار در خودش ضرب نمائیم. از آنجا که عدد یک عنصر خنثی در عمل ضرب است، ابتدا مقدار متغیر Power را برابر یک قرار داده و در یک حلقه که تعداد تکرار آن با متغیر شمارشگر Index از یک تا مقدار Second کنترل می‌شود بطور متوالی عدد First در Power ضرب شده و حاصل جدید در Power نگه‌داری می‌شود. در نهایت پس از اینکه عمل ضرب به تعداد Second بار اجرا شد، حلقه خاتمه یافته و مقدار Power به عنوان حاصل نمایش داده می‌شود.

مثال ۲۶,۳ فلوچارتی رسم کنید که n عدد را دریافت و بزرگترین و کوچکترین مقدار را یافته و چاپ کند.

مسئله: نمایش بزرگترین و کوچکترین عدد از n عدد ورودی

ورودی: تعداد n عدد

خروجی: بزرگترین و کوچکترین عدد

الگوریتم حل:

گام اول: شروع

گام دوم: اعداد Number و Quantity را از کاربر دریافت کن

گام سوم: مقدار Index را مساوی ۲ و مقدار Min و Max را مساوی Number قرار بده

گام چهارم: اگر Index کوچکتر یا مساوی Quantity است به *گام پنجم* برو، در غیر اینصورت به *گام نهم* برو.

گام پنجم: عدد Number را از کاربر دریافت کن

گام ششم: اگر Max از Number کوچکتر باشد مقدار Max را مساوی Number قرار بده و به گام هشتم برو

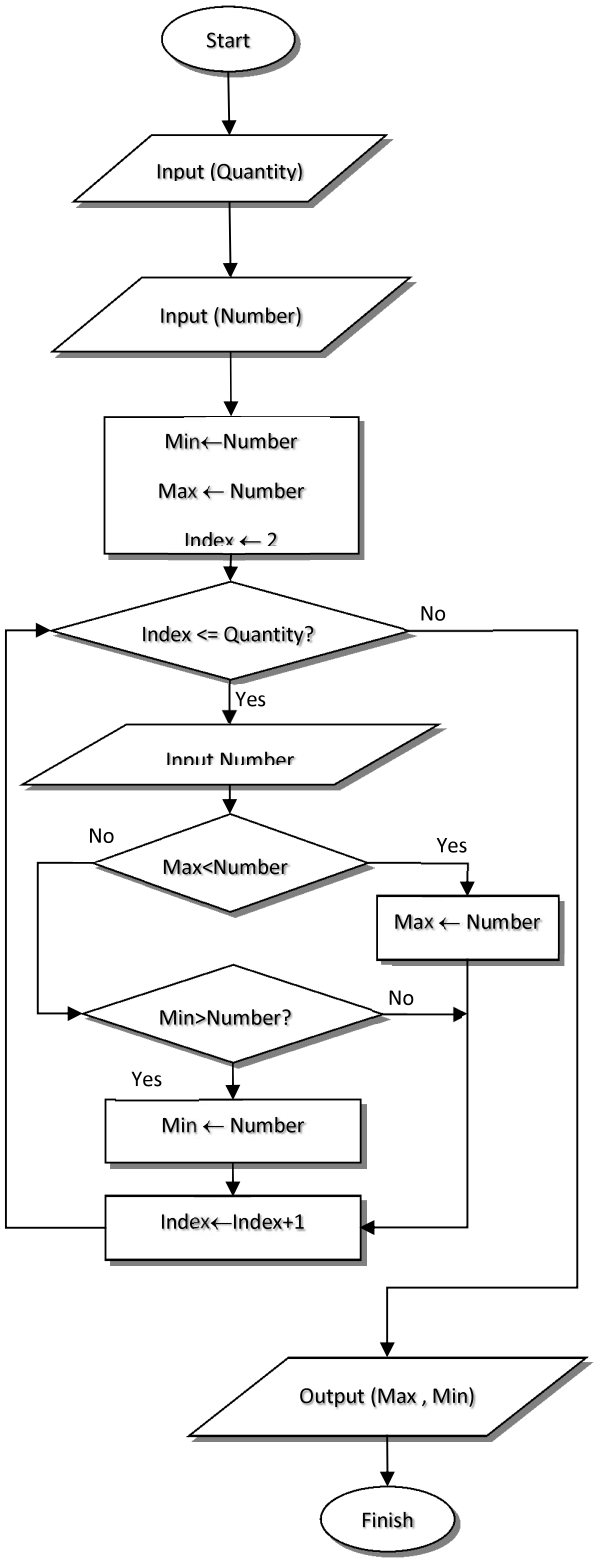
گام هفتم: اگر Min از Number بزرگتر باشد مقدار Min را مساوی Number قرار بده

گام هشتم: مقدار Index را یک واحد افزایش بده و به گام چهارم برو

گام نهم: مقدار Max و Min را چاپ کن

گام دهم: پایان

■



مثال ۲۷٫۳ فلوچارتی رسم کنید که n را دریافت و n امین عدد فیبوناچی را یافته و چاپ کند.

راهنمایی: در سری فیبوناچی، هر عدد حاصل جمع دو عدد ماقبل خود است و اعداد اول و دوم این سری برابر یک هستند.

1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , 55 , 89 , ...

مسئله: نمایش n اُمین عدد سری فیبوناچی

ورودی: عدد صحیح n

خروجی: n اُمین عدد سری فیبوناچی

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن.

گام سوم: مقدار Fib1 و Fib2 و Fib را مساوی یک و مقدار Index را مساوی ۳ قرار بده

گام چهارم: اگر Index کوچکتر یا مساوی Number نیست، به *گام هفتم* برو.

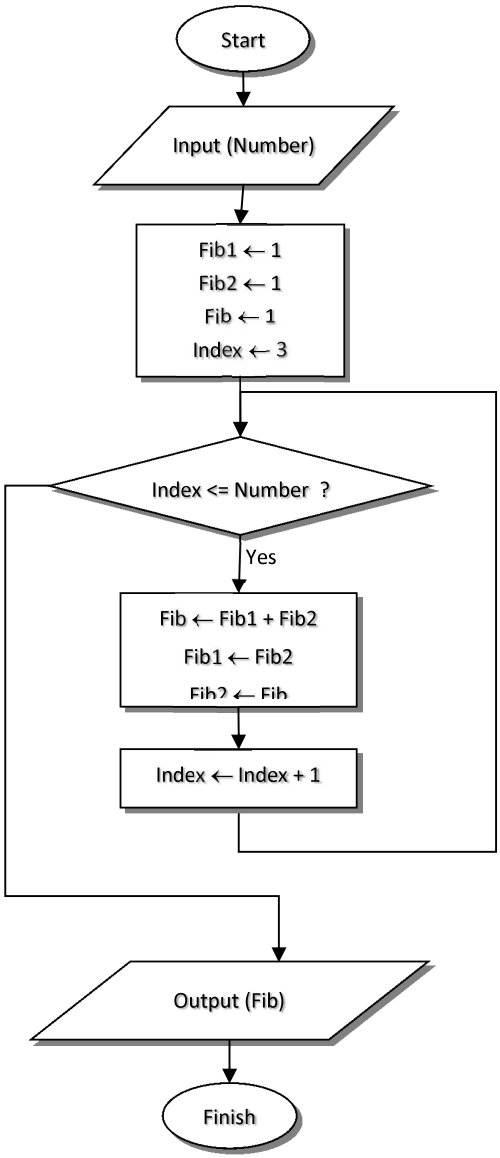
گام پنجم: مقدار Fib را مساوی مجموع Fib1 و Fib2 و مقدار Fib1 را مساوی Fib2 و مقدار Fib2 را مساوی Fib قرار بده.

گام ششم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو.

گام هفتم: مقدار Fib را چاپ کن.

گام هشتم: پایان

■



مثال ۲۸,۳ فلوچارتی رسم کنید که عدد n را دریافت و nعدد فیبوناچی قبل را یافته و چاپ کند.

مسئله: نمایش اعداد سری فیبوناچی ماقبل n

ورودی: عدد صحیح n

خروجی:اعداد سری فیبوناچی تا n

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن

گام سوم: مقدار Fib1 و Fib2 را مساوی یک و مقدار Fib را مساوی مجموع Fib1 و Fib2 و مقدار Index را مساوی 2 قرار بده

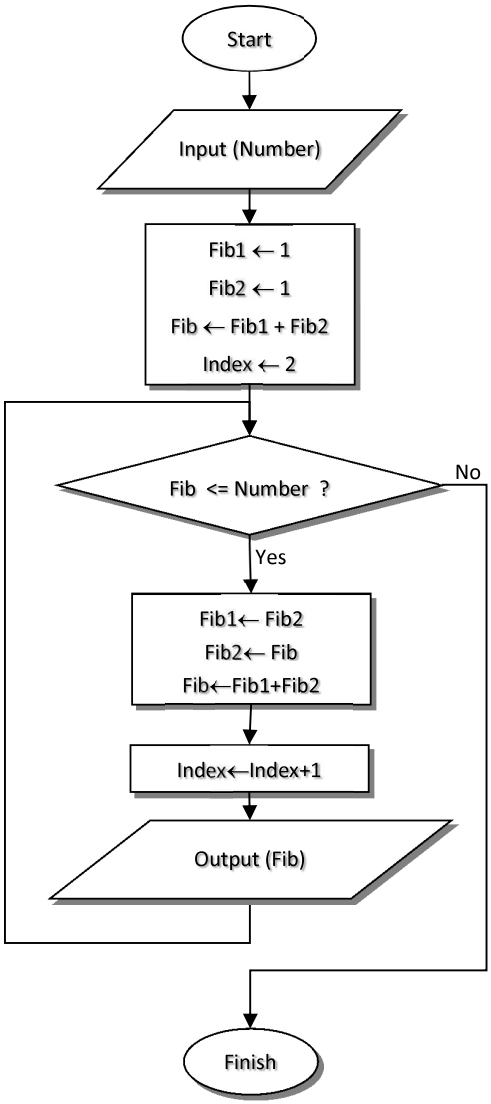
گام چهارم: اگر Fib کوچکتر یا مساوی Number است، به گام پنجم برو، در غیر اینصورت به گام هفتم برو.

گام پنجم: مقدار Fib1 رامساوی Fib2 و مقدار Fib2 را مساوی Fib و مقدار Fib را مساوی مجموع Fib1 و Fib2 قرار بده و مقدار Index را یک واحد افزایش بده.

گام ششم: مقدار Fib را چاپ کن و به گام چهارم برو

گام هفتم: پایان

■


Start
Input (Number)
Fib1 ← 1
Fib2 ← 1
Fib ← Fib1 + Fib2
Index ← 2
Fib <= Number ?
No
Yes
Fib1← Fib2
Fib2← Fib
Fib←Fib1+Fib2
Index←Index+1
Output (Fib)
Finish

مثال ۲ ۹٫۳ فلوچارتی رسم کنید که n را دریافت و مجموع n عدد فیبوناچی قبل را یافته و چاپ کند.

مسئله: نمایش مجموع اعداد سری فیبوناچی ماقبل n

ورودی: عدد صحیح n

خروجی: مجموع اعداد سری فیبوناچی تا n

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن

گام سوم: مقدار Fib1 رامساوی صفر و مقدار Fib2 را مساوی یک و مقدار Fib را مساوی مجموع Fib1 و Fib2 و مقدار Index را مساوی ۲ قرار بده

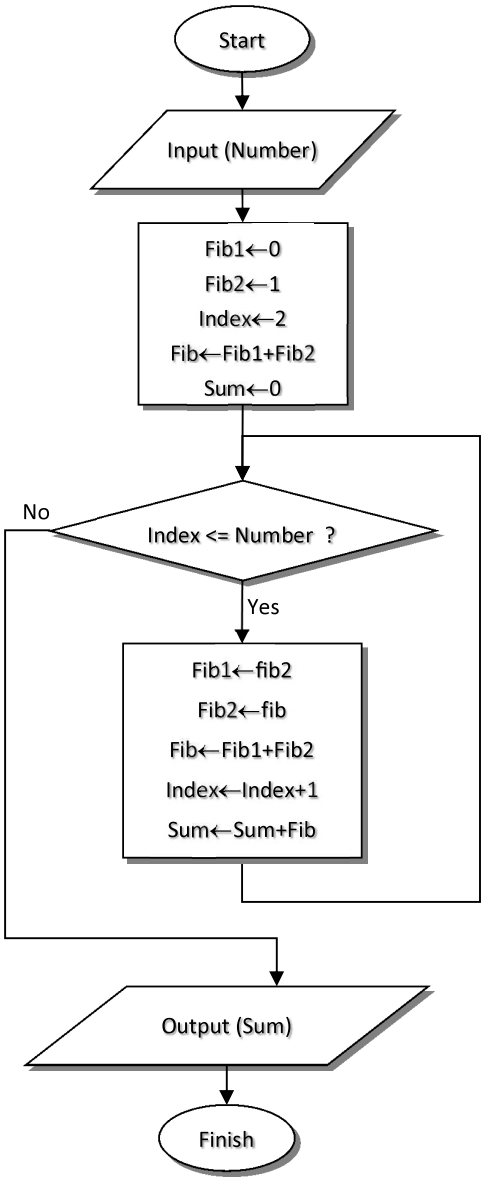
گام چهارم: اگر Index کوچکتر یا مساوی Number است به *گام* ششم برو

گام پنجم: مقدار Fib1 رامساوی Fib2 و مقدار Fib2 را مساوی یک و مقدار Fib را مساوی مجموع Fib1 و Fib2 قرار بده و مقدار Index را یک واحد افزایش بده و مقدار Fib را به Sum اضافه کن و به گام چهارم برو

گام ششم: مقدار Sum را چاپ کن

گام هفتم: پایان

■


Start
Input (Number)
Fib1←0
Fib2←1
Index←2
Fib←Fib1+Fib2
Sum←0
No
Index <= Number ?
Yes
Fib1←fib2
Fib2←fib
Fib←Fib1+Fib2
Index←Index+1
Sum←Sum+Fib
Output (Sum)
Finish

مثال ۳,۳۰ فلوچارتی رسم کنید که یک جدول ضرب ۱۰ در ۱۰ را نمایش دهد.

مسئله: نمایش جدول ضرب ۱۰ در ۱۰

ورودی: -

خروجی: جدول ضرب ۱۰ در ۱۰

الگوریتم حل:

گام اول: شروع

گام دوم: مقدار Row را مساوی یک قرار بده

گام سوم: اگر مقدار Row کوچکتر از ۱۰ است، به *گام چهارم* برو، در غیر اینصورت به *گام نهم* برو.

گام چهارم: مقدار Column را برابر یک قرار بده

گام پنجم: اگر مقدار Column از ۱۰ کوچکتر است، به *گام ششم* برو، در غیر اینصورت به *گام هشتم* برو

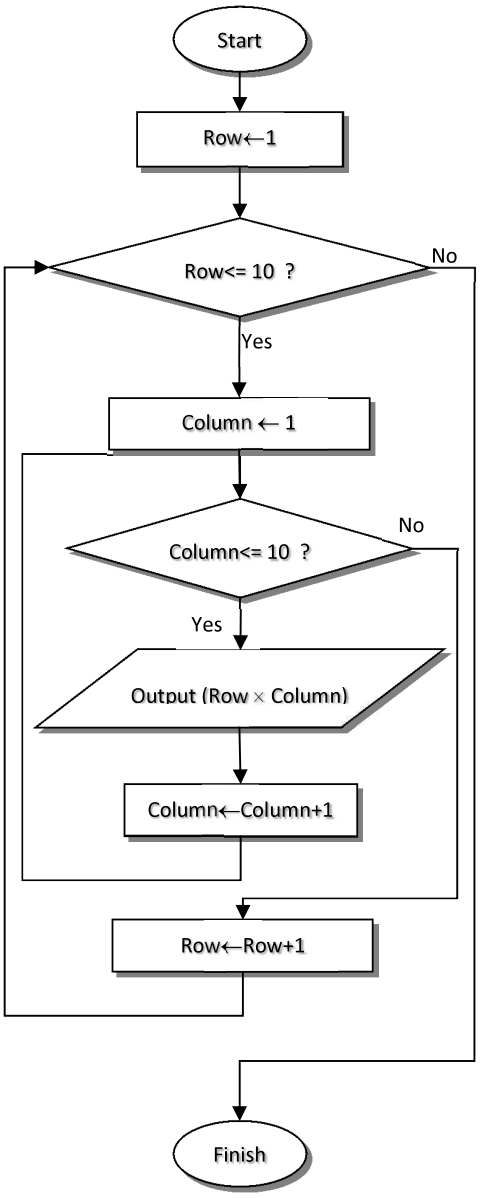
گام ششم: مقدار Row × Column را چاپ کن

گام هفتم: مقدار Column را یک واحد افزایش بده و به *گام پنجم* برو

گام هشتم: مقدار Row را یک واحد افزایش بده و به *گام سوم* برو.

گام نهم: پایان

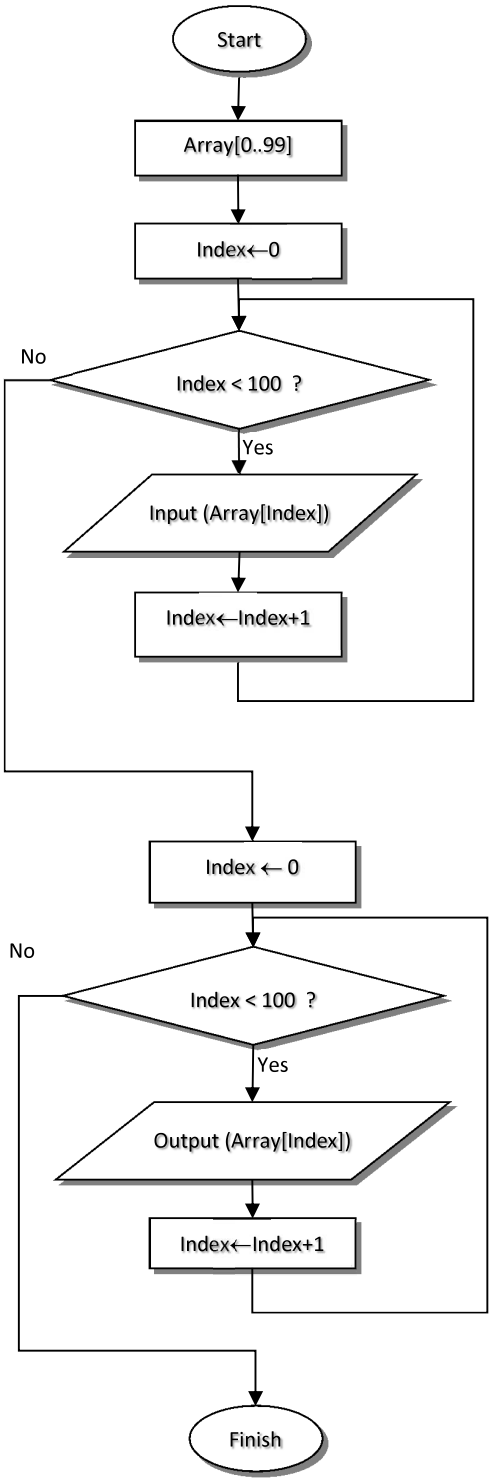
■



۳-۲-۳ آرایه

در برنامه‌نویسی و حل مسائل، گاه نیاز است که تعداد بسیاری از مقادیر را نگه‌داری نماییم تا در هنگام نیاز به آنها دسترسی داشته باشیم. به عنوان مثال فرض کنید می‌خواهیم نمره‌های دانشجویان یک کلاس را دریافت نموده و سپس از میان این نمرات، تعداد دانشجویانی را که نمره آنها بیش از میانگین کلاس بوده است را اعلام نماییم. برای اینکار لازم است نمرات دانشجویان کلاس را دریافت نموده در فضایی از حافظه نگه‌داری نماییم. معمولا برای انجام اینکار از آرایه‌ها استفاده می‌کنیم. آرایه، فضای پیوسته‌ای از حافظه است که می‌تواند مقادیری به تعداد مشخص را در خود نگه دارد و در هنگام نیاز، آنها را مورد استفاده قرار دهد. برای دسترسی به هر یک از عناصر آرایه پس از نام آرایه، اندیس آن ذکر می‌شود. مثلا فرض کنید آرایه ای با نام Grades را برای نگه‌داری نمرات دانشجویان در نظر گرفته‌ایم. برای دسترسی به نمره دانشجوی اول در لیست از Grades[0]، نمره دانشجوی دوم از Grades[1] و به همین ترتیب تا آخر از Grades[n-1] استفاده می‌کنیم.

مثال ۳٫۱٫۳ فلوچارتی رسم کنید که آرایه ۱۰۰ عنصری را دریافت و سپس آن را در خروجی نمایش دهد.



مسئله: دریافت آرایه ۱۰۰ عنصری و نمایش آن

ورودی: ۱۰۰ عدد

خروجی: ۱۰۰ عدد

الگوریتم حل:

گام اول: شروع

گام دوم: آرایه ۱۰۰ عنصری Array را با اندیس‌های از صفر تا ۹۹ تعریف کن.

گام سوم: مقدار Index را مساوی صفر قرار بده.

گام چهارم: اگر Index کوچکتر از ۱۰۰ است به *گام پنجم* و در غیر اینصورت به *گام هفتم* برو.

گام پنجم: مقداری را دریافت نموده و در Array[Index] قرار بده.

گام ششم: مقدار Index را یک واحد اضافه کن و به *گام چهارم* برو.

گام هفتم: مقدار Index را مساوی صفر قرار بده.

گام هشتم: اگر Index کوچکتر از ۱۰۰ است به *گام نهم* و در غیر اینصورت به *گام یازدهم* برو.

گام نهم: مقدار Array[Index] را چاپ کن.

گام دهم: مقدار Index را یک واحد افزایش بده و به *گام هشتم* برو.

گام یازدهم: پایان

■

مثال ۳۲,۳ فلوچارتی رسم کنید عددی را از ورودی دریافت کرده و معادل مبنای دو آن را نمایش دهد.

مسئله: تبدیل عدد به مبنای ۲

ورودی: عدد Number در مبنای ۱۰

خروجی: معادل عدد Number در مبنای ۲

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن.

گام سوم: آرایه Array با ۱۰۰ عنصری تعریف کن و مقدار Index را مساوی صفر قرار بده.

گام چهارم: اگر Index کوچکتر یا مساوی Number نیست به *گام ششم* برو.

گام پنجم: مقدار Array[Index] را مساوی باقیمانده تقسیم Number بر ۲ قرار بده و مقدار Index را یک واحد افزایش داده و مقدار Number را مساوی Number تقسیم بر ۲ قرار بده و به *گام چهارم* برو.

گام ششم: مقدار Index را یک واحد کاهش بده.

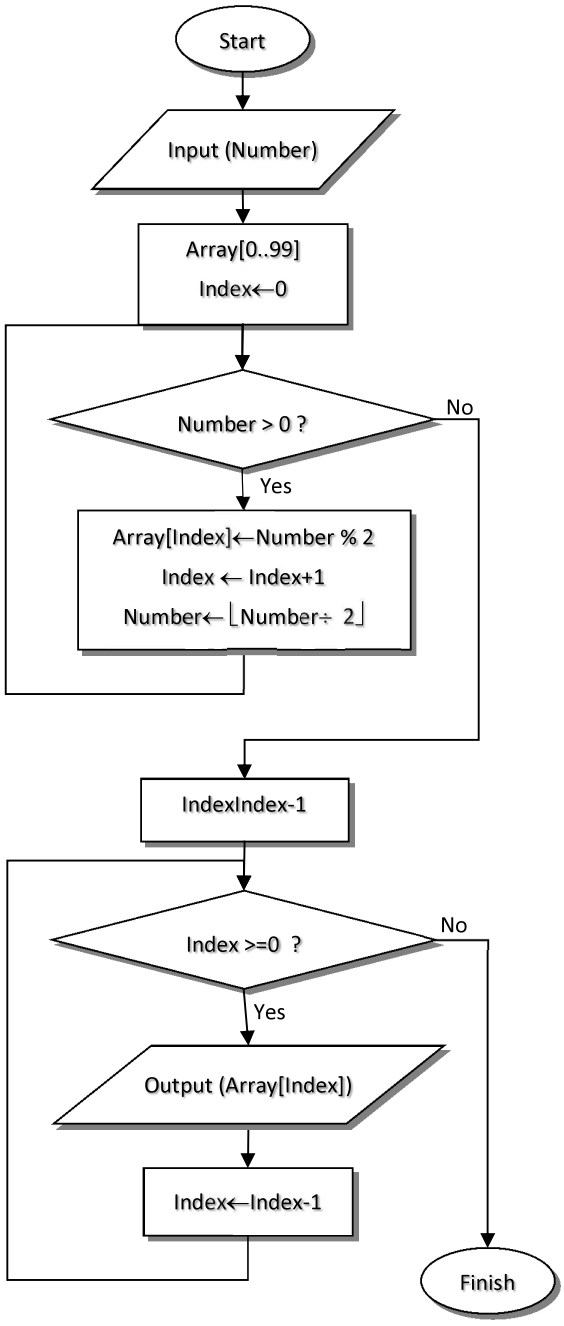
گام هفتم: اگر Index بزرگتر از صفر نباشد به *گام دهم* برو.

گام هشتم: مقدار Array[Index] را چاپ کن.

گام نهم: مقدار Index را یک واحد کاهش بده و به *گام هفتم* برو

گام دهم: پایان

■



مثال ۳٫۳٫۳ فلوچارتی رسم کنید که یک آرایه حداکثر ۱۰۰ عنصری را از ورودی دریافت کرده ، سپس با خواندن عنصری از ورودی ، آن را در آرایه جستجو کند .

مسئله: دریافت آرایه حداکثر ۱۰۰ عنصری از کاربر و جستجوی عددی در آن

ورودی: ۱۰۰ عنصر آرایه و عدد مورد جستجو

خروجی: موقعیت عدد مورد جستجو در آرایه

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number و Quantity را از کاربر دریافت کن.

گام سوم: آرایه Array با ۱۰۰ عنصری تعریف کن و مقدار Position و Index را مساوی صفر قرار بده.

گام چهارم: اگر Index کوچکتر یا مساوی Quantity نیست به *گام هفتم* برو.

گام پنجم: مقدار Array[index] را از کاربر دریافت کن.

گام ششم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو.

گام هفتم: مقدار Index را مساوی صفر قرار بده.

گام هشتم: اگر Index کوچکتر یا مساوی Quantity نیست به *گام یازدهم* برو.

گام نهم: اگر Array[index] مساوی Number است مقدار Position را مساوی Index قرار بده.

گام دهم: مقدار Index را یک واحد افزایش بده و به *گام هشتم* برو.

گام یازدهم: مقدار Position را چاپ کن.

گام یازدهم: پایان

■

رسم فلوچارت این الگوریتم، بر عهده دانشجو می‌باشد.

مثال ۳۴,۳ فلوچارتی رسم کنید که یک آرایه ۱۰۰ عنصری را دریافت نموده و عنصری را از کاربر گرفته و به روش جستجوی دودویی آن را در آرایه جستجو کند

مسئله:

ورودی:

خروجی:

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number را از کاربر دریافت کن.

گام سوم: مقدار Index را مساوی یک قرار بده و آرایه Array با ۱۰۰ عنصر را تعریف کن.

گام چهارم: اگر Index کوچکتر یا مساوی Number نیست به *گام هفتم* برو.

گام پنجم: مقدار Array[index] را از کاربر دریافت کن.

گام ششم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو.

گام هفتم: مقدار Low را مساوی یک قرار بده و مقدار High را مساوی Number قرار بده.

گام هشتم: مقدار Key را از کاربر دریافت کن.

گام نهم: اگر مقدار Low از High کوچکتر نیست چاپ کن " یافت نشد " و به *گام چهاردهم* برو.

گام دهم: Mid را مساوی میانگین مجموع Low و High قرار بده.

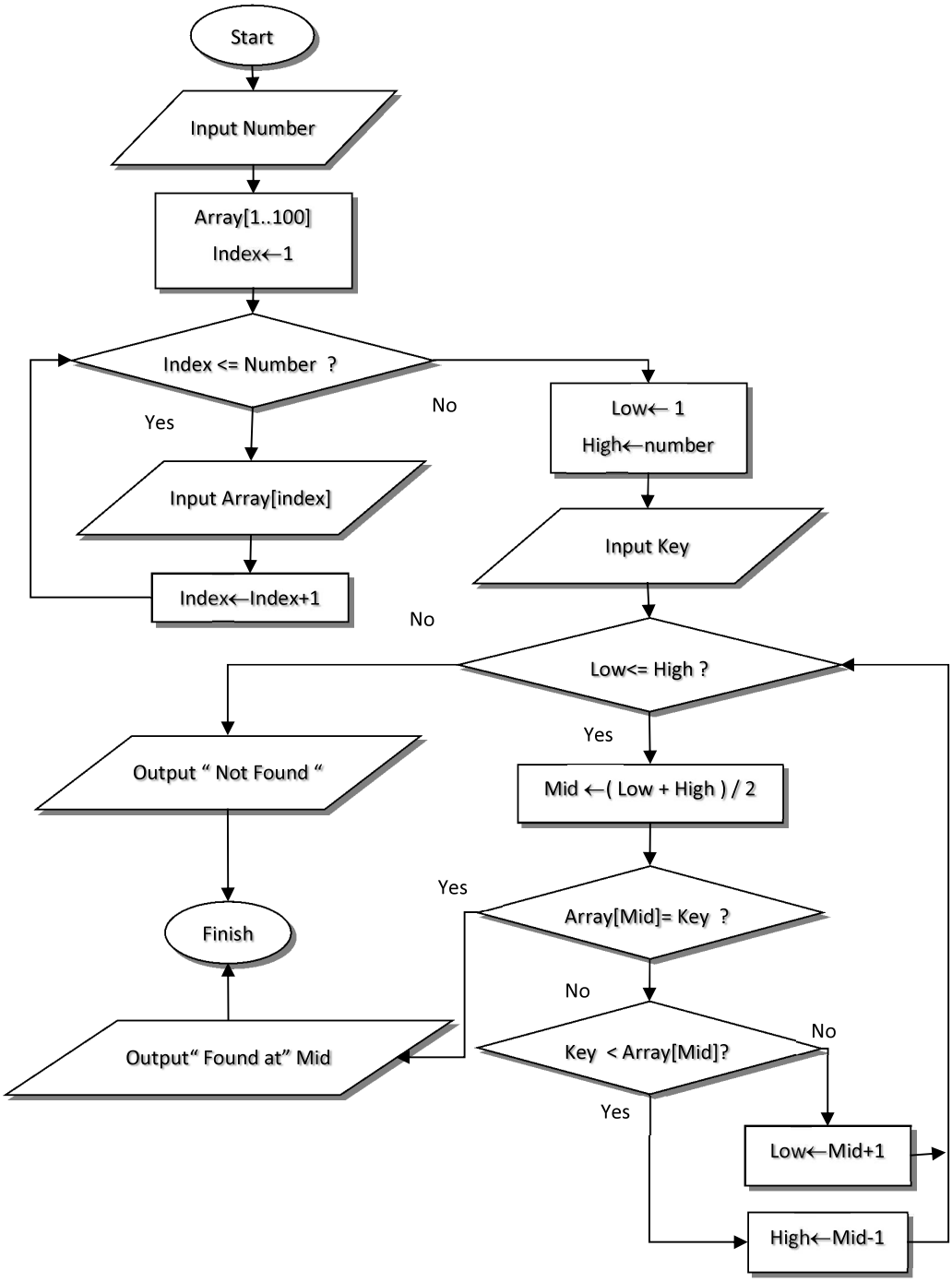
گام یازدهم: اگر Array[Mid] مساوی Key باشد چاپ کن "*در مکان Mid یافت شد*" و به *گام چهاردهم* برو.

گام دوازدهم: اگر Key از Array[Mid] کوچکتر باشد مقدار High را مساوی Mid-1 قرار بده.

گام سیزدهم: اگر Key از Array[Mid] کوچکتر نباشد مقدار Low را مساوی Mid+1 قرار بده.

گام چهاردهم: پایان.

■


Start
Input Number
Array[1..100]
Index←1
Index <= Number ?
Yes
No
Input Array[index]
Index←Index+1
Low← 1
High←number
Input Key
No
Low<= High ?
Yes
Output " Not Found "
Mid ←(Low + High) / 2
Yes
Array[Mid]= Key ?
Finish
No
Output" Found at" Mid
Key < Array[Mid]?
No
Yes
Low←Mid+1
High←Mid-1

مثال ۳۵,۳ فلوچارتی رسم کنید که آرایه ۱۰۰ عنصری را دریافت و سپس معکوس آن را بدست آورده، آرایه حاصل را در خروجی چاپ کند .

مسئله:

ورودی:

خروجی:

الگوریتم حل:

گام اول: شروع

گام دوم: عدد Number و Quantity را از کاربر دریافت کن

گام سوم: مقدار Index را مساوی یک قرار بده و آرایه Array با ۱۰۰ عنصری تعریف کن.

گام چهارم: اگر Index کوچکتر یا مساوی Quantity نیست به گام هفتم برو.

گام پنجم: مقدار Array[index] را از کاربر دریافت کن.

گام ششم: مقدار Index را یک واحد افزایش بده و به گام چهارم برو.

گام هفتم: مقدار Index را مساوی یک قرار بده و مقدار Position را مساوی Number قرار بده.

گام هشتم: اگر Index کوچکتر یا مساوی Positionنیست به گام دهم برو.

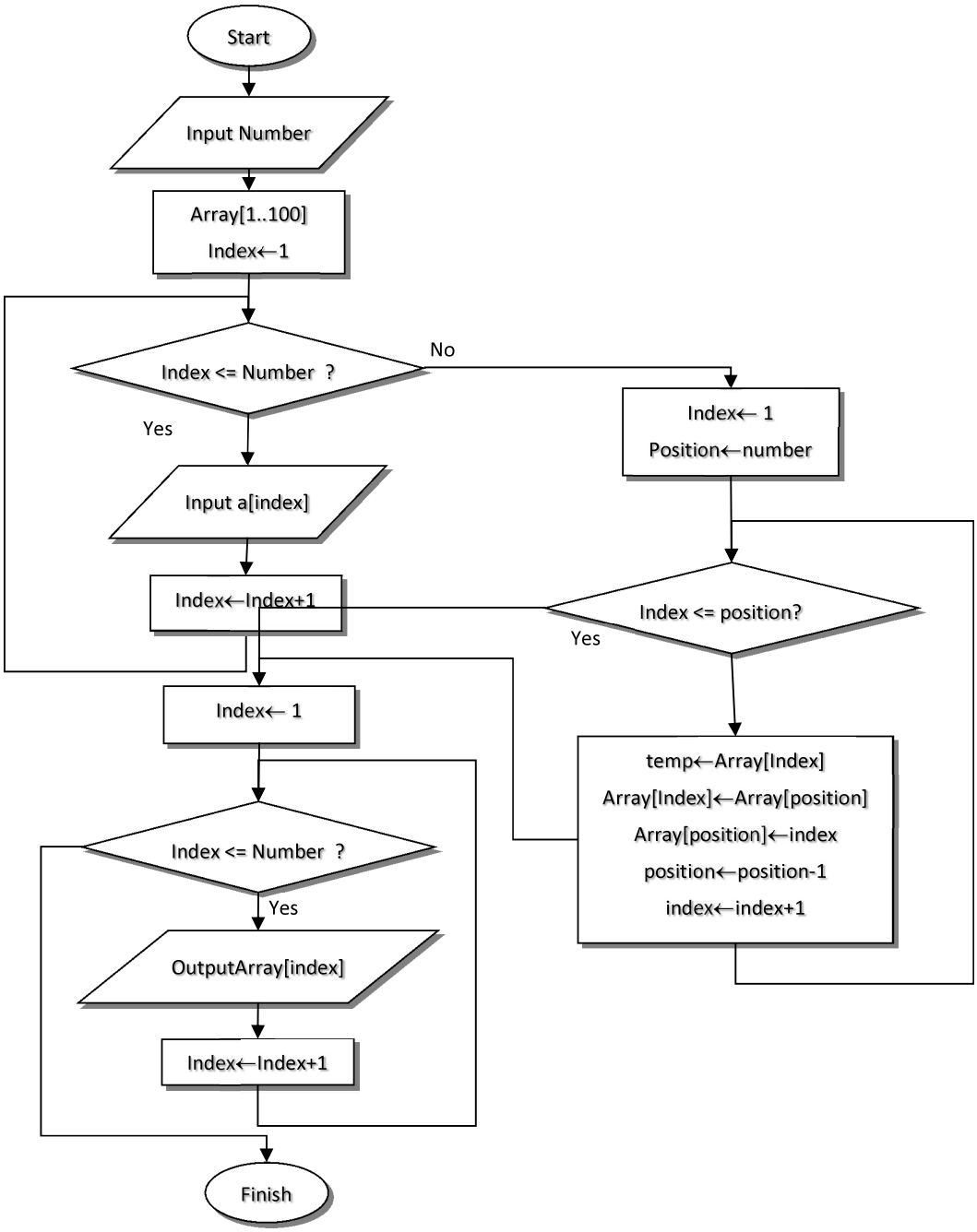
گام نهم: مقدار Temp را مساوی Array[index] و مقدار Array[position] را مساوی Array[index] و مقدار Position را یک واحد کاهش و مقدار Index را یک واحد افزایش بده.

گام دهم: اگر Index کوچکتر یا مساوی Number نیست به *گام سیزدهم* برو.

گام یازدهم: مقدار Array[index] را چاپ کن.

گام دوازدهم: مقدار Index را یک واحد افزایش بده.

گام سیزدهم: پایان


Start
Input Number
Array[1..100]
Index←1
Index <= Number ?
No
Yes
Input a[index]
Index←Index+1
Index← 1
Position←number
Index <= position?
Yes
Index← 1
temp←Array[Index]
Array[Index]←Array[position]
Array[position]←index
position←position-1
index←index+1
Index <= Number ?
Yes
OutputArray[index]
Index←Index+1
Finish

مثال ۳۶,۳ فلوچارتی رسم کنید که آرایه ۱۰۰ عنصری را دریافت و سپس عناصر آنرا به روش مرتب سازی حبابی، مرتب کند.

مسئله: مرتب سازی آرایه ۱۰۰ عنصری به روش مرتب سازی حبابی

ورودی: آرایه ۱۰۰ عنصری

خروجی: آرایه مرتب ۱۰۰ عنصری

الگوریتم حل:

گام اول: شروع

گام دوم: آرایه ۱۰۰ عنصری Array را با اندیس‌های از صفر تا ۹۹ تعریف کن.

گام سوم: مقدار Index را مساوی صفر قرار بده.

گام چهارم: اگر Index کوچکتر از ۱۰۰ نیست به *گام هفتم* برو.

گام پنجم: مقدار Array[index] را از کاربر دریافت کن.

گام ششم: مقدار Index را یک واحد افزایش بده و به *گام چهارم* برو.

گام هفتم: مقدار Index را مساوی یک قرار بده.

گام هشتم: اگر Index کوچکتر یا مساوی Number نیست به گام *سیزدهم* برو.

گام نهم: مقدار Index را مساوی یک قرار بده

گام دهم: اگر Index کوچکتر یا مساوی Number نیست به گام نوزدهم برو

گام یازدهم: مقدار Array[index] را چاپ کن

گام دوازدهم: مقدار Index را یک واحد افزایش بده و به گام دهم برو

گام سیزدهم: مقدار Counter را یک واحد افزایش بده

گام چهاردهم: اگر Counter بزرگتر یا مساوی Number نیست به گام هیجدهم برو

گام پانزدهم: اگر Array[Index] از Array[Counter] بزرگتر نباشد به گام هفدهم برو

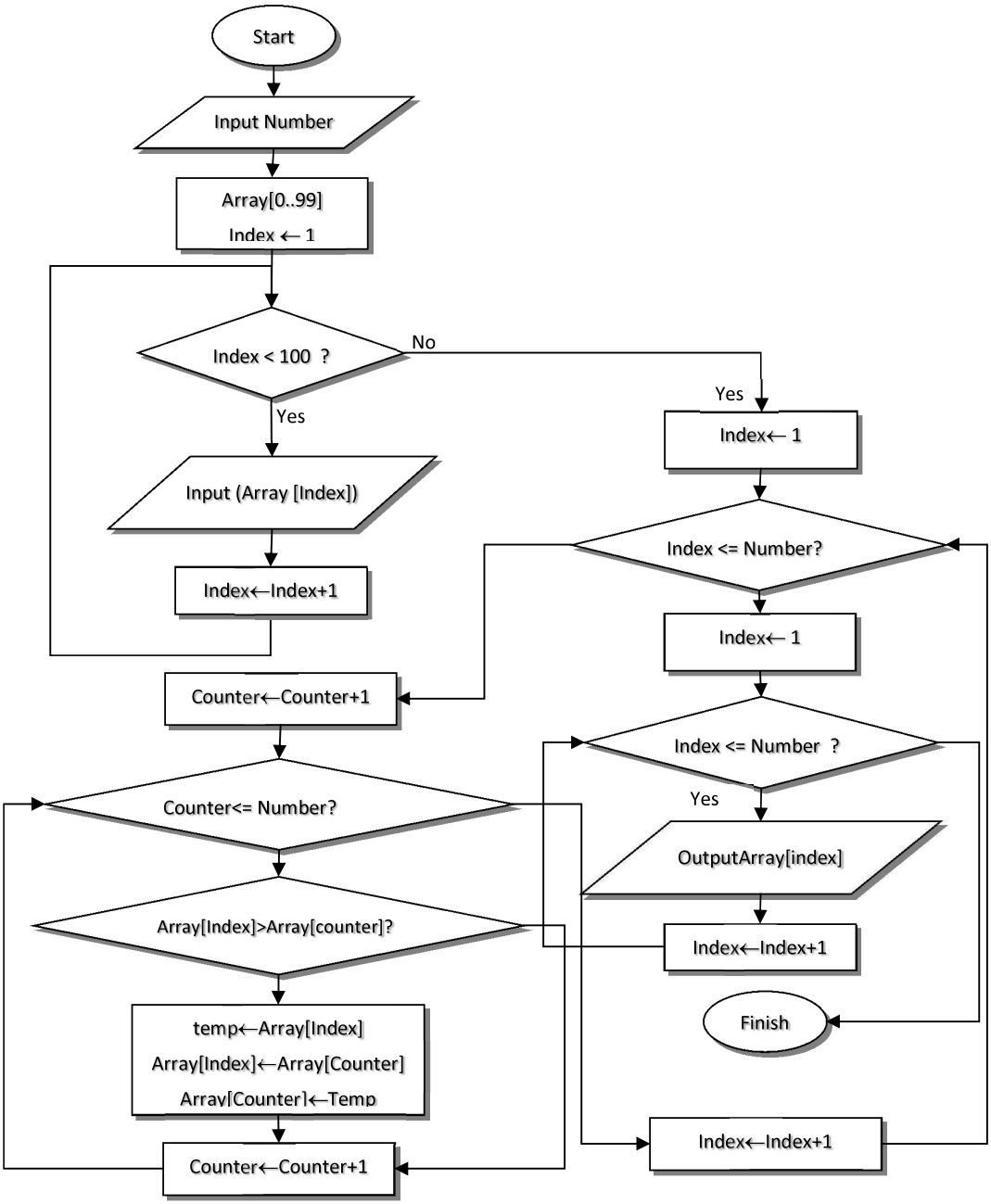
گام شانزدهم: مقدار Array[Index] را در متغیر Temp و مقدار Array[Counter] را در متغیر Array[Index] و مقدار Temp را در متغیر Array[Counter] قرار بده

گام هفدهم: مقدار Counter را یک واحد افزایش بده و به گام پانزدهم برو

گام هجدهم: مقدار Index را یک واحد افزایش بده و به گام هشتم برو

گام نوزدهم: پایان

■


Start
Input Number
Array[0..99]
Index ← 1
Index < 100 ?
No
Yes
Input (Array [Index])
Index←Index+1
Counter←Counter+1
Counter<= Number?
Array[Index]>Array[counter]?
temp←Array[Index]
Array[Index]←Array[Counter]
Array[Counter]←Temp
Counter←Counter+1
Yes
Index← 1
Index <= Number?
Index← 1
Index <= Number ?
Yes
OutputArray[index]
Index←Index+1
Finish
Index←Index+1

خودآزمایی

الگوریتمهای مناسبی برای هر یک از مسائل زیر، ارائه دهید.

الگوریتمی ارائه دهید که ساعت و دقیقه را دریافت نموده و زاویه بین عقربه ساعت شمار و دقیقه شمار را محاسبه و چاپ نماید.

الگوریتمی ارائه دهید که شماره ماهی از سال و شماره روزی از آن ماه سال را دریافت نموده و تعیین کند این تاریخ چندمین روز سال است.

الگوریتمی ارائه دهید که شماره روزی از سال (عددی از ۱ تا ۳۶۵) را دریافت نموده و تعیین کند این روز سال، چندمین روز از چندمین ماه سال است.

برای تشخیص مربع کامل بودن یک عدد، الگوریتمی ارائه دهید که مبتنی بر مقسوم علیه‌های عدد باشد.

برای هر یک از مسائل زیر، فلوچارت مناسبی رسم نمائید.

فلوچارتی رسم کنید که ۴ عدد دریافت نموده و تعداد اعداد متمایز را نمایش دهد.

فلوچارتی رسم کنید که ۴ عدد دریافت نموده و حداکثر اختلاف موجود بین آنها را نمایش دهد.

فلوچارتی رسم کنید که ۴ عدد دریافت نموده و حداقل اختلاف موجود بین آنها را نمایش دهد.

فلوچارتی رسم نمایید که ۴ عدد دریافت نموده و اعداد متمایز را نمایش دهد.

فلوچارتی رسم کنید که عددی را دریافت نموده و تعیین کند آن عدد، کامل است یا نه؟

راهنمایی: عدد کامل، عددی است که حاصل جمع مقسوم علیه های کوچکتر از خودش با خودش برابر باشد. مانند عدد ۶ و ۲۸

فلوچارتی رسم کنید که عددی را دریافت نموده و فاکتوریل آن را محاسبه و چاپ کند.

فلوچارتی رسم کنید که اعداد n و m را دریافت نموده و همه اعداد اول موجود از n تا m را نمایش دهد.

فلوچارتی رسم کنید که عددی را دریافت نموده و تعیین کند آن عدد SuperPrime است؟

راهنمایی: عدد SuperPrime، عددی است که هم اول باشد و هم از نظر ترتیب بین اعداد اول، اول باشد. مانند عدد ۱۱ که هم خودش اول است و هم چون پنجمین عدد اول است و ۵ عدد اول است SuperPrime است.

فلوچارتی رسم کنید که عددی را دریافت نموده و تعیین کند جزء اعداد سری فیبوناچی است یا نه؟