اصول برنامه نویسی کامپیوتر

به زبان C++

(The Principles of Computer Programming)

جمعی از مؤلفان

تاریخ آخرین ویرایش: ۲۸ شهریور ۱۳۹۳

فهرست مطالب

مبانی کامپیوتر

در زندگی روزمره، ما همانند بسیاری از انسان‌ها به دفعات از کامپیوتر استفاده می‌کنیم. برای نمونه، ممکن است از کامپیوتر برای ساختن اسلایدهای یک ارائه، نوشتن یک سند متنی مانند این کتاب، انجام محاسبات حسابداری در قالب صفحات گسترده، ارسال و دریافت نامه‌های الکترونیک یا انجام برنامه‌نویسی استفاده کنیم. اما همیشه اوضاع به این شیوه نبوده است. در زمان‌های نه چندان دور، اکثر انسان‌ها به کامپیوتر به عنوان یک موجود عجیب و غریب و رازآلود می‌نگریستند که فقط تعداد محدودی از جادوگران(!) از رازهای آن آگاهند! در این فصل قصد داریم تا با کامپیوتر آشنا شویم و برای این آشنایی، به سه سوال باید پاسخ دهیم. سوال اول، کامپیوتر چیست؟ سوال دوم چه نیازی به کامپیوتر داریم و سوال سوم، چگونه باید از کامپیوتر استفاده کرد؟ پاسخ به سوال اول و دوم و بخشی از سوال سوم را در این فصل خواهید یافت. در فصل‌های آتی، سوال سوم به صورت دقیق‌تر و گسترده‌تری بررسی می‌گردد. در حقیقت، بخش زیادی از این کتاب برای پاسخ به سوال سوم نوشته شده است.

۱.۱ کامپیوتر چیست؟

قبل از آن که بخواهیم کامپیوتر را تعریف کنیم، لازم است تا به سوال دوم پاسخ دهیم:

" چرا به کامپیوتر نیاز داریم؟ "

امروزه اطلاعات نقش بی بدیلی در پیشرفت اقتصادی جوامع و افراد بازی می‌کند ، برای به دست آوردن اطلاعات، نیاز است تا داده‌های مرتبط مورد پردازش قرار گیرند. حجم داده‌ای که امروزه در اختیار داریم بسیار پیچیده و حجیم است به حدی که پردازش آن توسط انسان را اگر نگوییم ناممکن، بسیار دشوار می‌کند. اما این دشواری دلیلی برای چشم‌پوشی از مزایای اطلاعات نمی‌شود. بنابراین لازم است تا ابزارهایی مناسب برای پردازش داده‌ها به کار گرفته شوند. کامپیوتر یکی از این ابزارها است.

بنابراین تا اینجا، کامپیوتر ابزاری است برای پردازش داده‌ها و تولید خروجی مناسب که به آن اطلاعات می‌گوییم. تا اینجا از کلمات داده، پردازش و اطلاعات استفاده کرده‌ایم، اما هنوز آن‌ها را تعریف نکرده‌ایم. به هر گونه اطلاعات خام و اولیه‌ای قرار است طی فرایندی به صورت مفید یا خواسته شده تبدیل شود، **داده** گوییم. منظور از **پردازش**، فرایندی است که طی آن داده به اطلاعات تبدیل می‌شود. بنابراین، **اطلاعات**، خروجی قابل استفاده و حاصل پردازش داده است. به صورت کلی، در طی این فرایند، سه کار اساسی باید انجام شوند:

۱. داده‌ها جمع آوری و ذخیره گردند (**ورودی**)،
۲. داده‌ها مورد پردازش قرار گیرند، یعنی عملیات مناسب بر روی این داده‌ها انجام شود، عملیاتی نظیر دسته‌بندی، سازماندهی، مرتب سازی و خلاصه سازی (**پردازش**)،
۳. نتایج حاصل شده در مرحله دوم به محیط خارج ارسال شوند (**خروجی**).

پردازش می‌تواند توسط انسان یا ماشین انجام شود، شاید جالب باشد بدانید اولین استفاده از لفظ کامپیوتر، به قرن ۱۷ میلادی باز می‌گردد. کامپیوتر یک عنوان شغلی برای افرادی بوده است که وظیفه انجام محاسبات در سازمان‌ها و شرکت‌ها را به عهده داشته اند. به هر طریق، هدف در پردازش، تبدیل داده به اطلاعات است. بنابراین یک کامپیوتر در پردازش اطلاعات،

۱. داده‌های اولیه را از طریق ابزارهایی که برای وارد کردن آن‌ها در نظر گرفته شده است، دریافت کرده و در حافظه کامپیوتر ذخیره می‌کند،
۲. سپس طبق دستور العمل‌هایی که قبلا از طریق ورودی به آن داده شده است، عملیات لازم را برای رسیدن به نتیجه مطلوب بر داده‌ها انجام می‌دهد،
۳. نتایج حاصل از پردازش را به خروجی مورد نظر ارسال می‌کند.

با این توضیحات، می‌توان کامپیوتر را به صورت زیر تعریف کرد:

> " **کامپیوتر** ابزاری است که **داده** را از **ورودی** دریافت کرده و سپس تحت کنترل مجموعه‌ای از دستور العمل‌ها که به آن‌ها **برنامه‌های کامپیوتری** گفته می‌شود، **پردازش** کرده و **اطلاعات** حاصل را به **خروجی** ارسال می‌کند. "

بنابراین با توجه به تعریف فوق، برنامه کامپیوتری عبارت است از دنباله‌ای از دستورات کامپیوتری که توسط **برنامه‌نویسان** تعیین شده اند. در ادامه این کتاب، به نحوه نوشتن برنامه‌های کامپیوتری خواهیم پرداخت. بحث درباره برنامه‌های کامپیوتری را به بخش ۴.۱ موکول می‌کنیم.

اجزای یک کامپیوتر را می‌توان به سه دسته زیر نیز تقسیم نمود:

سخت‌افزار [۱] که شامل همه قسمت‌های فیزیکی و قابل لمس یک کامپیوتر است،

[۱] Hardware

نرم‌افزار [۲] که وظیفه هدایت و کنترل سخت‌افزار را به عهده دارد. به عبارت دیگر، کلیه برنامه‌ها و دستورالعمل‌هایی که جهت ارتباط با کامپیوتر و استفاده از آن به کار گرفته می‌شوند را نرم‌افزار نامند.

میان افزار بخش‌هایی از کامپیوتر که ویژگی‌های سخت‌افزاری و نرم‌افزاری را به صورت توامان در خود داشته باشند.

فارغ از تمام تفاوت‌هایی که در ظاهر فیزیکی کامپیوترها وجود دارد، سخت‌افزار هر کامپیوتر را می‌توان به پنج بخش منطقی تقسیم نمود که به این تقسیم‌بندی منطقی، **سازمان کامپیوتر** گویند. این بخش‌ها عبارتند از:

۱. **بخش ورودی:** وظیفه این بخش دریافت داده‌ها از دستگاه‌های ورودی است. صفحه کلید[۳]، موشواره[۴]، پویشگر[۵] و مانند آن‌ها برخی از دستگاه‌های متداول در این بخش هستند.
۲. **بخش خروجی:** این بخش در حکم واحد صدور کامپیوتر است و اطلاعات پردازش شده را در دستگاه‌های خروجی قرار می‌دهد تا در خارج از کامپیوتر قابل استفاده باشند. امروزه بیشتر اطلاعات بر روی صفحه نمایش ظاهر می‌شوند یا برای کنترل دستگاه‌های دیگر به کار می‌روند. صفحه نمایش[۶] و چاپگر[۷] از معمول‌ترین ابزارهای این بخش هستند.
۳. **حافظه اصلی:** کلیه دستورالعمل‌ها، داده‌های ورودی از بخش ورودی و اطلاعات حاصل از پردازش قبل از این که به بخش خروجی ارسال شوند، در حافظه اصلی[۸] قرار می‌گیرند. حافظه اصلی به دو دسته حافظه با دسترسی تصادفی[۹] و حافظه فقط خواندنی[۱۰] تقسیم بندی می‌شود. اطلاعات در حافظه‌های با دسترسی تصادفی هم قابل خواندن و هم قابل نوشتن هستند، اما در حافظه‌های فقط خواندنی، اطلاعات پس از این که نوشته شدند، غیرقابل تغییر هستند. حافظه با دسترسی تصادفی را از این رو با قابلیت دسترسی تصادفی می‌نامند که سرعت دسترسی به اطلاعات وابسته به محل ذخیره‌سازی آن نیست. به عبارت دیگر، اطلاعات در هر جای حافظه که باشد، با سرعت یکسانی قابل دستیابی است، اما این حافظه ناپایدار است، به این معنی که با قطع برق، اطلاعات ذخیره شده در آن از بین می‌روند.
۴. **حافظه ثانویه**[۱۱]: ظرفیت نگهداری اطلاعات و داده‌ها در این حافظه‌ها بسیار بالاتر از حافظه اصلی است، اما سرعت دسترسی به اطلاعات ذخیره شده در آن کمتر از حافظه اصلی است، اما از نظر قیمت، از حافظه اصلی ارزان‌تر هستند. این گونه حافظه‌ها پایدار هستند، به این معنی که با قطع جریان برق، اطلاعات ذخیره شده در آن‌ها از دست نمی‌رود. بنابراین، هنگامی که نیاز به ذخیره داده و اطلاعات برای مدت زمان طولانی‌تری هستیم، از حافظه‌های ثانویه استفاده می‌شود.

[۲] Software
[۳] keyboard
[۴] mouse
[۵] scanner
[۶] screen
[۷] printer
[۸] main memory
[۹] Random Access Memory (RAM)
[۱۰] Read Only Memory (ROM)
[۱۱] secondary memory

۵. **بخش پردازش مرکزی**[۱۲]: این بخش از دو قسمت اصلی تشکیل شده است. **واحد محاسبه و منطق** که وظیفه انجام عملیات محاسباتی و منطقی را به عهده دارد و **واحد کنترل** که وظیفه مدیریت اجرای دستورالعمل‌ها و پیگیری اجرای آن‌ها به همراه مدیریت بخش‌های ورودی، خروجی و حافظه را به عهده دارد.

۲.۱ شیوه نمایش اطلاعات در کامپیوتر

امروزه اطلاعات و داده‌هایی که در اختیار داریم، انواع متفاوت و پیچیدگی‌های مختلفی دارند. برای مثال، برخی از داده‌ها عددی هستند، برخی از داده‌ها به صورت متن هستند و برخی از داده‌ها به صورت تصویرند و مانند آن. برای این که بتوانیم از کامپیوترها برای پردازش این داده‌ها استفاده کنیم، نیاز است تا آن‌ها را به صورت مناسبی در کامپیوترها نمایش دهیم. این شیوه نمایش را **کدگذاری** نامند.

می‌دانیم همه انواع داده را می‌توان با اعداد نمایش داد و هر عملیاتی را می‌توان با اعمال محاسباتی روی اعداد انجام داد. در محاسبات معمول از اعداد در مبنای ۱۰ استفاده می‌کنیم، اما پیاده‌سازی این گونه محاسبات در کامپیوترهای الکتریکی کمی دشوار است. به همین جهت از اعداد در مبنای ۲ برای پیاده‌سازی محاسبات در کامپیوترها استفاده می‌کنیم. دلیل این انتخاب، سادگی طراحی مدارهای الکتریکی است. در مبنای ۲، برای نمایش اعداد مختلف تنها به دو حالت نیاز داریم. بر اساس تصمیم طراح، یکی از حالت‌ها عدد «۰» و به دیگری عدد «۱» نسبت داده می‌شود. به هر یک از ارقام «۰» و «۱» در مبنای دو، یک رقم دودویی گفته می‌شود و برای نمایش هر رقم دودویی، به یک بیت حافظه نیاز داریم. به عبارت دیگر، هر بیت قابلیت ذخیره یک رقم دودویی را دارد.

قبل از این که به نحوه ذخیره‌سازی اطلاعات در کامپیوتر بپرازیم، لازم است تا مروری بر دستگاه‌های عددنویسی داشته باشیم. هر دستگاه عددنویسی با دو مولفه مبنا و مجموعه ارقام مشخص می‌شود که تعداد عناصر در مجموعه ارقام، به تعداد عدد مبنا است. برای نمونه، در دستگاه عددنویسی معمولی، مبنا ۱۰ و مجموعه ارقام عبارت است از ارقام $۰, ۱, \ldots, ۹$.

در هر دستگاه عددنویسی، عدد را به صورت مجموعه‌ای از ارقام نمایش می‌دهیم. هر رقم در نمایش یک عدد دارای دو نوع ارزش است. یکی ارزش خود رقم، برای مثال رقم ۵ دارای ارزش ۵ واحد است، و دیگری ارزش مکانی آن رقم که برابر است با ارزش خود آن رقم ضرب در مبنا به توان مکان آن رقم. مکان هر رقم در نمایش یک عدد از سمت راست محاسبه شده و مکان اولین رقم صفر، مکان دومین رقم، یک و برای سایر ارقام نیز به همین صورت است، برای نمونه شکل ۱.۱ را مشاهده نمایید. در هر دستگاه عددنویسی، مقدار یک عدد برابر است با حاصل جمع ارزش‌های مکانی تک تک ارقام آن. برای مثال، برای عدد (۵۰۲۳۱) در دستگاه عددنویسی از مرتبه ۱۰ داریم

[۱۲] Central Processing Unit (CPU)

عدد	۲	۰	۸	۴	۱	۵	۶	۷	۹
مکان رقم	۰	۱	۲	۳	۴	۵	۶	۷	۸
ارزش مکانی رقم	۲×۱۰^{۰}	۰×۱۰^{۱}	۸×۱۰^{۲}	۴×۱۰^{۳}	۱×۱۰^{۴}	۵×۱۰^{۵}	۶×۱۰^{۶}	۷×۱۰^{۷}	۹×۱۰^{۸}

شکل ۱.۱: دستگاه عددنویسی دهدهی.

$$(۵\ ۰\ ۲\ ۳\ ۱) = ۱\times۱۰^{۰}+۳\times۱۰^{۱}+۲\times۱۰^{۲}+۰\times۱۰^{۳}+۵\times۱۰^{۴}$$
$$=۵۰۲۳۱.$$

در ادامه به بررسی دستگاه‌های عددنویسی متداول در علوم‌کامپیوتر، دستگاه‌های عددنویسی در مبنای ۲، ۸ و ۱۶ پرداخته می‌شود.

۱.۲.۱ اعداد طبیعی

در این قسمت، به بحث تبدیل اعداد طبیعی (صحیح و مثبت) به مبناهای دیگر می‌پردازیم. برای اعداد منفی از روش‌های مختلفی استفاده می‌شود که در بخش بعدی به آن می‌پردازیم.

دستگاه عددنویسی دودویی، متداول‌ترین دستگاه عددنویسی در علوم‌کامپیوتر است. در این دستگاه، مبنا عدد ۲ است و مجموعه ارقام عبارتند از ۰ و ۱. هر عدد در این دستگاه به صورت دنباله‌ای از ارقام صفر و یک است. برای محاسبه مقدار یک عدد دودویی، کافی است به شیوه قبل عمل کنیم، یعنی حاصل جمع ارزش مکانی تک تک ارقام آن عدد دودویی را محاسبه کنیم. برای مثال مقدار عدد دودویی $(۰\ ۱\ ۱\ ۰\ ۱)_۲$ به صورت زیر محاسبه می‌شود:

$$(۱\ ۰\ ۱\ ۱\ ۰\ ۱\ ۰)_۲ = ۰\times۲^{۰}+۱\times۲^{۱}+۰\times۲^{۲}+۱\times۲^{۳}+۱\times۲^{۴}+۰\times۲^{۵}+۱\times۲^{۶}$$
$$=۰+۲+۰+۸+۱۶+۰+۶۴$$
$$=۹۰.$$

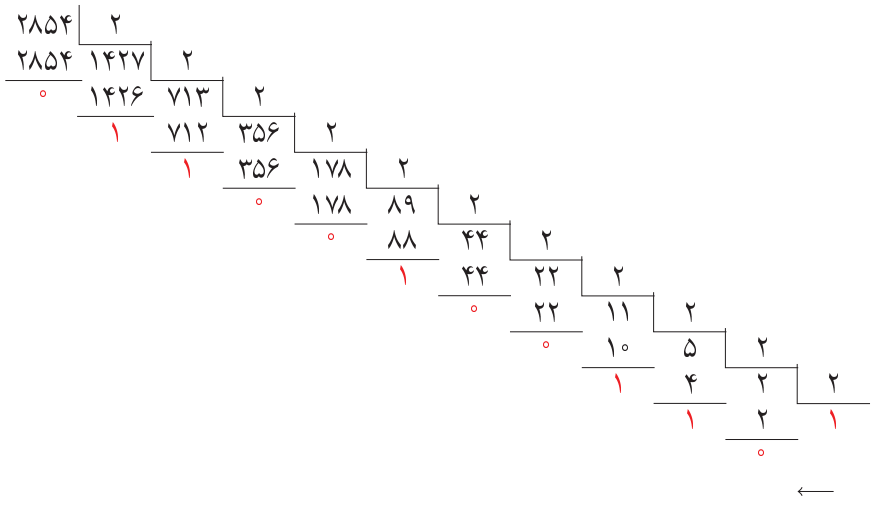
اما برای انجام عکس این عمل، یعنی یافتن نمایش عددی با مقداری از پیش معلوم در یک دستگاه عددنویسی با مبنایی غیر از ۱۰، از فرایندی موسوم به تقسیم‌های متوالی استفاده می‌کنیم. این فرایند را با استفاده از یک مثال شرح خواهیم داد. برای نمونه، می‌خواهیم نمایش دودویی عددی با مقدار ۲۸۵۴ را بیابیم:

۱. عدد ۲۸۵۴ را بر ۲ تقسیم کرده و باقی مانده آن را به عنوان اولین رقم از عدد دودویی در نظر می‌گیریم،

۲. تقسیمات متوالی را با تقسیم خارج‌قسمت مرحله قبل بر ۲ ادامه می‌دهیم تا خارج‌قسمت مرحله جاری از ۲ کوچک‌تر شود. باقی مانده هر مرحله را به عنوان رقم بعدی عدد دودویی در نظر می‌گیریم.

۳. خارج‌قسمت مرحله نهایی، که از ۲ کوچک‌تر است را به عنوان سمت چپ‌ترین رقم عدد دودویی مورد نظر قرار می‌دهیم.

این فرایند در شکل ۲.۱ آمده است. حال فرض کنید عددی به صورت ۱۰۱۱۰۰۱۰۰۱۱۰ داده شده است. از شما می‌خواهیم که مقدار آن را محاسبه کنید. اما محاسبه آن نیازمند اطلاعات اضافه‌ای است! این عدد در چه مبنایی نوشته شده است؟ مبنای ۱۰؟ مبنای ۲؟ مبنای ۷؟ و به همین صورت. بنابراین، این اطلاعات اضافه، یعنی مبنای عدد را باید به شیوه‌ای در نمایش عدد گنجاند. بدین منظور، دنباله ارقام عدد را در یک پرانتز نوشته و سپس بیرون پرانتز، به صورت اندیس، مبنا را به صورت



$$(۱۰۱۱۰۰۱۰۰۱۱۰)_۲ = ۲۸۵۴$$

شکل ۲.۱: مثال از تبدیل یک عدد از مبنای ۱۰ به مبنای ۲.

$$(\text{ارقام})_{\text{مبنا}},$$

می‌نویسیم. از آنجا که در محاسبات روزمره دستگاه عددنویسی دهدهی مرسوم است، برای اعدادی که در مبنای ۱۰ نمایش داده شده اند، پرانتز و مبنا را حذف می‌کنیم. همچنین در هنگامی که یک عدد را می‌خوانیم، مقدار آن را بیان می‌کنیم. بنابراین برای عددی مانند $(۱۰۰۱)_۲$ نمی‌گوییم هزار و یک در مبنای دو، زیرا مقدار این عدد هزار و یک نیست، بلکه ۹ است. به همین دلیل قرارداد می‌کنیم که

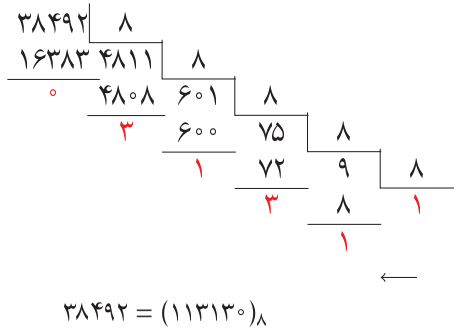
هنگامی که اعداد در مبنای ۱۰ را بیان می‌کنیم، مقدار آن را بیان کنیم و برای سایر مبناها، ارقام آن را از سمت چپ به سمت راست می‌خوانیم، یعنی یک، صفر، صفر، یک در مبنای دو.

یکی از مشکلاتی که در نمایش به مبنای دو داریم، طولانی بودن طول عدد است. برای حل این مشکل، می‌توان از مبناهای بزرگ‌تر استفاده کرد. مبناهایی که توانی از عدد ۲ هستند، کاندیداهای خوبی برای این منظورند. مبناهای ۸ و ۱۶، دو دستگاه متداول و پرکاربرد در این دسته هستند که در ادامه به بررسی آن‌ها می‌پردازیم.

در دستگاه عددنویسی هشت هشتی، مبنا عدد ۸ و مجموعه ارقام عبارتند از $\{۰,۱,\ldots,۷\}$. شیوه تبدیل اعداد از/به این دستگاه نیز مانند قبل است. برای نمونه، عدد $(۴\ ۳\ ۱\ ۳\ ۱\ ۱)_۸$ دارای مقدار

$$(۴\ ۳\ ۱\ ۳\ ۱\ ۱)_۸ = ۴\times۸^{۰}+۳\times۸^{۱}+۱\times۸^{۲}+۳\times۸^{۳}+۱\times۸^{۴}+۱\times۸^{۵}$$
$$=۳۸۴۹۲,$$

است. به عکس، نمایش عددی با مقدار ۳۸۴۹۲ در مبنای ۸ به صورت شکل ۳.۱ است. به همین



$$۳۸۴۹۲ = (۱۱۳۱۳۰)_۸$$

شکل ۳.۱: مثال از تبدیل یک عدد از مبنای ۱۰ به مبنای ۸.

صورت، مبنا در دستگاه شانزده شانزدهی، ۱۶ است. اما برای نمایش مجموعه ارقام در این دستگاه نیاز به ۱۶ رقم متفاوت داریم، در صورتی که ۱۰ رقم بیشتر در اختیار نیست. بدین منظور، ارقام با ارزش‌های ۰ تا ۹ را با ارقام معمول و شش رقم باقیمانده با ارزش‌های ۱۰ تا ۱۵ را از حروف الفبای انگلیسی انتخاب می‌کنیم، به این صورت که برای رقم با ارزش ۱۰ حرف A را انتخاب می‌کنیم و برای سایر ارقام، حروف بعدی را به ترتیب الفبا انتخاب می‌کنیم. در جدول ۱.۱، مجموعه ارقام در مبنای ۱۶ آمده است.

به صورت مشابه می‌توان نشان داد که نمایش عددی با مقدار ۳۸۴۹۲ در مبنای ۱۶ عبارت است از $(۹۶۵C)_{۱۶}$.

تبدیلاتی که تا اینجا انجام داده‌ایم، همواره یا از یک مبنای غیر از ۱۰ به مبنای ۱۰ بوده و یا به صورت بالعکس بوده اند. برای تبدیل عدد از یک مبنای غیر از ۱۰ به یک مبنای دیگر که آن هم غیر

ارزش رقم	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
نمایش رقم در مبنای ۱۶	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	A	B	C	D	E	F

جدول ۱.۱: مجموعه ارقام دستگاه عددنویسی در مبنای ۱۶ به همراه ارزش هر رقم.

از ۱۰ است، می‌توان عدد را ابتدا با محاسبه حاصل جمع مکانی ارقام آن عدد به مبنای ۱۰ برد و سپس با استفاده از روش تقسیم‌های متوالی، مقدار آن در مبنای ۱۰ را به مبنای مورد نظر تبدیل کرد.

البته روش ساده‌تری نیز برای انجام این تبدیل وجود دارد که مبتنی بر جدول ارقام است. جدول ارقام برای مبناهای ۲، ۸ و ۱۶ در جدول ۲.۱ آمده است. این روش را با استفاده از چند مثال توضیح

ارزش عدد	مبنای ۲	مبنای ۸	مبنای ۱۶
۰	۰	۰	۰
۱	۱	۱	۱
۲	۱۰	۲	۲
۳	۱۱	۳	۳
۴	۱۰۰	۴	۴
۵	۱۰۱	۵	۵
۶	۱۱۰	۶	۶
۷	۱۱۱	۷	۷
۸	۱۰۰۰	۱۰	۸
۹	۱۰۰۱	۱۱	۹
۱۰	۱۰۱۰	۱۲	A
۱۱	۱۰۱۱	۱۳	B
۱۲	۱۱۰۰	۱۴	C
۱۳	۱۱۰۱	۱۵	D
۱۴	۱۱۱۰	۱۶	E
۱۵	۱۱۱۱	۱۷	F

جدول ۲.۱: جدول تبدیل ارقام برای مبناهای ۲، ۸ و ۱۶.

می‌دهیم. فرض کنید می‌خواهیم عدد

$$(۱\ ۰\ ۱\ ۱\ ۰\ ۰\ ۱\ ۰\ ۱\ ۱\ ۱\ ۰\ ۰)_{۲},$$

را به مبناهای ۸ و ۱۶ تبدیل کنیم. برای تبدیل به مبنای ۸، ابتدا ارقام عدد را از سمت راست به چپ به صورت سه رقم سه رقم دسته‌بندی می‌کنیم. در صورتی که یک دسته انتهایی کمتر از ۳ رقم

داشته باشد، به سمت چپ عدد را تعداد لازم رقم ۰ را اضافه می‌کنیم تا دسته آخر نیز سه رقمی شود. سپس با استفاده از جدول ۲.۱، به ازای هر دسته سه رقمی، معادل آن در مبنای ۸ را به جای آن دسته می‌نویسیم. حاصل، نمایش آن عدد در مبنای ۸ است. بنابراین برای عدد فوق داریم

$$\begin{array}{cccccc} (& ۰۰۱ & ۰۱۱ & ۰۰۱ & ۰۱۱ & ۱۰۰ \ )_۲ \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ (& ۱ & ۳ & ۱ & ۳ & ۴ \ )_۸ \end{array}$$

همچنین، برای تبدیل به مبنای ۱۶، ابتدا ارقام عدد را از سمت راست به چپ به صورت چهار رقم چهار رقم دسته‌بندی می‌کنیم. در صورتی که یک دسته انتهایی کمتر از ۴ رقم داشته باشد، به سمت چپ عدد را تعداد لازم رقم ۰ را اضافه می‌کنیم تا دسته آخر نیز چهار رقمی‌شود. سپس با استفاده از جدول ۲.۱، به ازای هر دسته چهار رقمی، معادل آن را در مبنای ۱۶ را به جای آن دسته می‌نویسیم. حاصل، نمایش آن عدد در مبنای ۱۶ است.

$$\begin{array}{cccccc} (& ۰۰۰۱ & ۰۱۱۰ & ۰۱۰۱ & ۱۱۰۰ \ )_۲ \\ & \downarrow & \downarrow & \downarrow & \downarrow \\ (& ۱ & ۶ & ۵ & C \ )_{۱۶} \end{array}$$

معکوس این تبدیل ساده‌تر است. کافی است برای تبدیل یک عدد در دستگاه شانزده شانزدهی به دودویی یا هشت هشتی، به جای هر رقم مبنای شانزده، معادل آن را از جدول ۲.۱ برداشته و به جای آن قرار دهیم.

۲.۲.۱ اعداد اعشاری

در خصوص اعداد اعشاری، بحث ارزش مکانی رقم مشابه مطالب بیان شده در بخش قبل است. تنها تفاوت این است که ارزش ارقام بعد از ممیز، هر چه به سمت راست حرکت کنیم کمتر می‌شود. به عبارت دقیق‌تر، ارزش اولین رقم بعد از ممیز، برابر $۱۰^{-۱}$ و در حالت کلی، ارزش رقم iام برابر ۱۰^{-i} است.

با این ترتیب برای تبدیل اعداد اعشاری از مبنای ۱۰ به مبنای ۲، ابتدا قسمت صحیح عدد را به روش فوق به مبنای ۲ تبدیل می‌کنیم. سپس قسمت اعشاری آن را با روش ضرب‌های متوالی؛ مشابه روش تقسیم متوالی ولی با ضرب عدد در ۲ در هر مرحله؛ به مبنای ۲ تبدیل می‌کنیم. با هر بار ضرب، قسمت صحیح عدد به عنوان رقم مربوطه در تبدیل عدد به مبنای ۲ استفاده می‌شود و قسمت اعشاری عدد دوباره وارد روند ضرب متوالی می‌شود. روند ضرب‌های متوالی وقتی متوقف می‌شود که عدد صفر شود. دقت کنید که در روش تقسیم متوالی، حتماً پس از تعداد متناهی مرحله به صفر خواهیم رسید ولی در تبدیل قسمت اعشاری ممکن است هیچ موقع به عدد صفر نرسیم. این خیلی

غیرمعمول نیست چرا که در مبنای ۱۰نیز بسیاری از اعداد اعشاری دارای قسمت اعشاری متناهی نیستند؛ نظیر حاصل تقسیم ۱ بر ۳.

پس از تبدیل قسمت صحیح و اعشاری عدد اعشاری به مبنای ۲، قسمت صحیح در مبنای ۲ را سمت چپ نقطه ممیز و قسمت اعشاری را سمت راست نقطه ممیز قرار می‌دهیم.

مثال ۱.۲.۱. *عدد ۱۳٫۲۵ را به مبنای ۲ تبدیل کنید.*

حل: با استفاده از روش تبدیل اعداد صحیح به مبنای ۲، داریم $(۱۱۰۱)_۲ = ۱۳$. برای تبدیل عدد ۰٫۲۵ به مبنای ۲، ابتدا آن را در ۲ ضرب می‌کنیم. قسمت صحیح عدد حاصل برابر ۰ و قسمت اعشاری آن برابر ۰٫۵ است. دوباره عدد را در ۲ ضرب می‌کنیم. قسمت صحیح عدد حاصل برابر با ۱ و قسمت اعشاری برابر با صفر است. این به معنی پایان روند تبدیل است. و لذا $(۰٫۰۱)_۲ = ۰٫۲۵$. در نتیجه $(۱۱۰۱٫۰۱)_۲ = ۱۳٫۲۵$. □

برای تبدیل به سایر مبناها، می‌توان از روش‌های فوق برای تبدیل از مبنای ۲ به سایر مبناها استفاده کرد. بررسی درستی تبدیل عدد زیر به مبنای ۲ به خواننده واگذار می‌شود:

$$۱۲۴٫۴۰۶۲۵ = (۱۱۱۱۱۰۰٫۰۱۱۰۱)_۲ = (۱۷۴٫۳۲)_۸ = (۷C٫۶۸)_{۱۶}$$

جهت بررسی درستی تبدیل اعداد به مبناهای مختلف، خواننده را به سایت http://baseconvert.com/ ارجاع می‌دهیم.

۳.۱ نحوه ذخیره‌سازی اطلاعات در کامپیوتر

همانگونه که قبلا گفته شد، برای ذخیره‌سازی اطلاعات در کامپیوتر از دستگاه عددنویسی در مبنای دو استفاده می‌کنیم. به کوچک‌ترین واحد حافظه که قادر به ذخیره تنها یک رقم دودویی ، یعنی ۰ یا ۱، باشد یک **بیت**[۱۳] گفته می‌شود. هر هشت بیت را یک **بایت**[۱۴] گویند که کوچک‌ترین قسمت قابل آدرس دهی در حافظه است. با توجه به تعداد بیت‌هایی که برای هر بایت در نظر گرفته شده است، هر بایت می‌تواند یکی از اعداد ۰ تا ۲۵۵ را ذخیره کند. هر $۲^{۱۰}$ بایت را یک **کیلوبایت**[۱۵] می‌نامند و هر $۲^{۱۰}$ کیلوبایت را یک **مگابایت**[۱۶] گویند. هر $۲^{۱۰}$ مگابایت برابر با یک **گیگابایت**[۱۷] است و هر $۲^{۱۰}$ گیگابایت برابر با یک **ترابایت**[۱۸] است.

در کامپیوتر برای نگهداری انواع مختلف داده‌ها، فضایی در نظر گرفته می‌شود. فضای در نظر گرفته شده برای هر نوع داده، به سخت‌افزار، سیستم عامل و کامپایلری که برای برنامه‌نویسی استفاده می‌شود بستگی دارد. البته باید توجه داشت که با توجه به فضای محدودی که برای نگهداری هر داده اختصاص داده می‌شود، محدوده خاصی از اعداد می‌تواند در حافظه ذخیره شود. این یکی از

---

[۱۳] bit
[۱۴] byte
[۱۵] KB
[۱۶] MB
[۱۷] GB
[۱۸] TB

مهمترین محدودیت‌های کامپیوتر است که ممکن است در حل برخی مسائل باعث بروز مشکل شود. در قسمت برنامه‌نویسی این مشکل را مفصل‌تر بحث خواهیم کرد.

۱.۳.۱ اعداد طبیعی

برای ذخیره سازی اعداد طبیعی، کافی است عدد مورد نظر به مبنای ۲ تبدیل شده و عدد حاصل در حافظه ذخیره شود. اگر برای ذخیره‌سازی اعداد طبیعی از ۲ بایت حافظه استفاده شود، بوضوح اعداد بین ۰ تا ۶۵۵۳۵ = $(1111111111111111)_2$ را می‌توان در این فضا نگهداری کرد. در صورت ذخیره عددی بزرگ‌تر از این مقدار در حافظه، اصلاحاً گوییم حافظه سرریز خواهد کرد و تعدادی از بیت‌های آن از دست می‌رود. لذا در ذخیره‌سازی داده‌ها یا حاصل محاسبات باید به بازه اعداد قابل ذخیره‌سازی دقت کرد وگرنه نتیجه محاسبات به دلیل از دست رفتن بخشی از آنها درست نخواهد بود.

۲.۳.۱ اعداد صحیح

برای نگهداری اعداد صحیح، طبیعتاً باید برای نگهداری اعداد منفی روندی مشخص شود. برای این کار روش‌های مختلفی وجود دارد که در اینجا به تعدادی از آنها اشاره می‌کنیم:

علامت-مقدار: [۱۹] در این روش، با ارزش‌ترین بیت از فضای در نظر گرفته شده برای نگهداری عدد صحیح به علامت اختصاص داده می‌شود. اگر مقدار این بیت برابر با صفر باشد به معنی مثبت بودن عدد و در صورت ۱ بودن این بیت به معنی منفی بودن عدد است. دقت کنید که با توجه به این که فضای ذخیره‌سازی عدد یک بیت کاهش می‌یابد، بازه محدودتری از اعداد قابل نگهداری در فضا است. به عنوان مثال اگر ۲ بایت برای نگهداری عدد صحیح استفاده شود، عملاً ۱۵ بیت برای نگهداری عدد در اختیار داریم و لذا اعداد بین ۳۲۷۶۷− تا ۳۲۷۶۷ قابل نگهداری در این فضا است.

مکمل اول: [۲۰] در این روش، برای تبدیل عدد منفی به مبنای ۲، ابتدا قدرمطلق آن عدد را به مبنای ۲ تبدیل می‌کند. اگر عدد کوچک‌تر از فضای مورد نظر برای نگهداری عدد است، در پشت عدد به تعداد لازم صفر اضافه می‌شود. سپس تمام ارقام ۱ به ۰ و تمام ارقام ۰ به ۱ تبدیل می‌شود. به عنوان مثال عدد ۹۷− در روش مکمل اول با استفاده از یک بایت برای ذخیره‌سازی به صورت ۱۰۰۱۱۱۱۰ است. اگر از ۲ بایت برای ذخیره‌سازی استفاده شود، ۸ عدد ۱ باید به پشت این عدد اضافه شود. دقت کنید که عدد حاصل از این روش به فضای در نظر گرفته شده برای ذخیره‌سازی بستگی دارد.

مکمل دوم: [۲۱] در این روش مشابه روش مکمل اول عمل می‌کنیم. فقط در انتها، به عدد حاصل عدد ۱ را اضافه می‌کنیم. دقت کنید که برای جمع، مشابه اعداد در مبنای ۱۰ عمل می‌کنیم با این

[۱۹] Sign-Magnitude [۲۰] 1's complement [۲۱] 2's complement

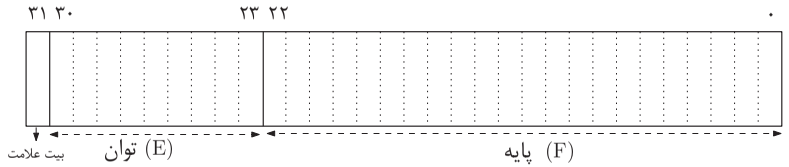
تفاوت که در خصوص ارقامی که در حاصل جمع بزرگ‌تر از ۱ است؛ مشابه ده بر یک در جمع اعداد در مبنای ۱۰؛ باید دو بر یک کنیم. به عنوان مثال، عدد ۹۷− در این نمایش، و در ۸ بیت به صورت ۱۰۰۱۱۱۱۱ است. عدد ۵۶− در این روش به صورت ۱۱۰۰۱۰۰۰ درمی‌آید. در روش مکمل دوم، اعداد قابل نمایش با طول m بیت در بازه $[1-2^{m-1}, 2^{m-1}-]$ قرار می‌گیرند.

هرکدام از این روش‌ها دارای معایب و مزایایی است که عمدتاً در بحث طراحی مدارهای منطقی برای انجام محاسبات ریاضی مورد بحث و بررسی قرار می‌گیرند. لذا در اینجا به آنها نمی‌پردازیم.

۳.۳.۱ اعداد اعشاری

اعداد اعشاری، علاوه بر این که اعداد با اندازه‌های مختلف را دارند، ممکن است دارای تعداد ارقام اعشاری مختلف نیز باشد. در اینجا فقط به روش ممیز شناور[۲۲] اشاره می‌کنیم.

عدد ممیز شناور، اعداد را به صورت علمی یعنی $F \times r^E$ درآورده و سپس E و F را نگهداری می‌کند که در آن r مبنای در نظر گرفته شده برای عدد است. برای ذخیره‌سازی اعداد اعشاری در کامپیوتر، طبیعتاً از $r = 2$ استفاده می‌کنیم. برای این منظور ابتدا عدد را به نماد علمی در مبنای ۲ تبدیل کرده و سپس فضای در نظر گرفته شده برای ذخیره‌سازی را به دو قسمت تقسیم کرده و در یک قسمت E و در قسمت دیگر F را ذخیره می‌کنیم. دقت کنید که نمایش علمی یک عدد اعشاری منحصر به فرد نیست؛ مثلاً عدد ۵۵٫۶۶ (در سیستم دهدهی) را می‌توان به صورت‌های ۵۵۶۶×۱۰^{-۱}، $۰٫۵۵۶۶ \times ۱۰^{۲}$ و $۰٫۰۵۵۶۶ \times ۱۰^{۳}$ نمایش داد. معمولاً از یک شکل نرمال‌شده برای ذخیره‌سازی استفاده می‌شود. در شکل نرمال‌شده پایه بین ۰ و ۱ در نظر گرفته می‌شود. این عدد در نمایش دودویی در محل در نظر گرفته شده برای پایه و توان که یک عدد صحیح (مثبت یا منفی) است در قسمت توان ذخیره می‌شود. یک بیت نیز برای علامت عدد اعشاری در نظر گرفته شده است (شکل ۴.۱ را ببینید).



شکل ۴.۱: نمایش عدد ممیز شناور در ۳۲ بیت.

با توجه به شکل فوق، مشخص است که برای اعداد اعشاری نیز مشابه اعداد صحیح، دارای بازه اعداد قابل نگهداری هستیم که فضای در نظر گرفته شده برای توان، این بازه را تعیین می‌کند. علاوه بر این، در تعداد ارقام اعشاری قابل ذخیره‌سازی یا دقت نیز محدودیت وجود دارد. نتیجه سریع این محدودیت این است که امکان نگهداری اعداد اعشاری که نمایش آنها دارای تعداد رقم

[۲۲] floating-point

اعشاری متناهی نیست، به صورت دقیق در کامپیوتر وجود ندارد و برای ذخیره‌سازی آنها بسته به فضای اختصاص داده شده به عدد، گرد کردن عدد انجام شده و سپس ذخیره می‌شود. لذا انجام محاسبات دقیق در این خصوص ممکن نیست و این امر باید در کار با کامپیوتر مد نظر قرار گیرد.

۴.۳.۱ حروف و نمادها

حروف و نمادها مشابه اعداد نیستند که بتوان آن را در مبنای دو نمایش داد و در کامپیوتر ذخیره کرد. برای حل مشکل، از کدگذاری استفاده می‌کنیم، یعنی به هر حرف یا نماد، عددی نسبت می‌دهیم و عدد متناظر با هر حرف را به جای آن ذخیره می‌کنیم. با این ترتیب تمام اطلاعات متنی با استفاده از کد کردن به صورت تعدادی عدد ذخیره می‌شود و برای استفاده از آن یا خروجی آن، عمل از کد درآوردن آنها انجام می‌شود. از کامپیوترهای اولیه تاکنون استانداردهای مختلفی برای کدگذاری حروف استفاده شده است که در اینجا به تعدادی از آنها اشاره می‌کنیم.

کدگذاری اسکی (ASCII) [۲۳] این روش جزء اولین روش‌های کدگذاری استاندارد برای حروف است. در این روش در ابتدا از ۷ بیت برای نگهداری کد مربوط به حروف و نمادها استفاده شد که بعدا جهت هماهنگی با کامپیوترهای ۸ بیتی، به ۸ بیت افزایش یافت. مثلاً در این کدگذاری، برای نگهداری حروف ’A’ تا ’Z’ از کدهای (دهدهی) ۶۵ تا ۹۰؛ برای ’a’ تا ’z’ از کدهای ۹۷ تا ۱۲۲ و برای ارقام ’۰’ تا ’۹’ از کدهای ۴۸ تا ۵۷ استفاده شده است.

توسیع‌های متعددی از این استاندارد (نظیر ANSI[۲۴] یا EBCDIC[۲۵]) وجود دارد که در اینجا به آنها نمی‌پردازیم.

با توجه به محدودیت تعداد کدها، بدیهی است تعداد حروف محدودی را می‌توان در این روش نگهداری کرد که عمدتاً همان حروف زبان انگلیسی است. در روش‌های بعدی، تلاش برای نمایش حروف زبان‌های مختلف به استانداردهای کامل‌تری منجر شد.

یونیکد [۲۶] قبل از یونیکد، هیچ نمایش دیگری از حروف، تمام زبان‌ها را پوشش نمی‌داد. به عنوان مثال، اروپای غربی از روش‌های کدگذاری متعددی (در خانوادهٔ ISO-8859-x) استفاده می‌کردند و حتی یک زبان واحد نظیر چینی، دارای روش‌های کدگذاری متعدد نظیر GB2312/GBK و BIG5 بود. هدف از یونیکد، ارائهٔ یک روش کدگذاری حروف استاندارد است که جهانی، کارا، یکنواخت و غیرمبهم باشد. استاندارد یونیکد توسط یک سازمان غیرانتفاعی به نام Unicode Consortium[۲۷] ارائه و نگهداری شده است.

[۲۳] American Standard Code for Information Interchange
[۲۴] American National Standards Institute
[۲۵] Extended Binary Coded Decimal Interchange Code
[۲۶] Unicode
[۲۷] www.unicode.org

یونیکد با برخی از کدگذاری‌های قبلی نظیر اسکی نیز سازگار است، یعنی از یک کد برای حروف در این دو روش کدگذاری استفاده شده است.

یونیکد اصولاً از ۱۶ بیت استفاده می‌کند که می‌تواند تا ۶۵۵۳۶ حرف را کدگذاری کند. این فضا تا ۲۱ بیت نیز گسترش یافته است که تا تقریباً دو میلیون حرف جدید و قدیمی را پوشش می‌دهد. یونیکد ۱۶ بیتی اصلی با هواپیمای چندزبانهٔ اصلی[۲۸] شناخته می‌شود که تمام زبان‌های اصلی که در حال حاضر استفاده می‌شوند را پوشش می‌دهد. حروف خارج از این مجموعه را حروف تکمیلی می‌نامند که زیاد استفاده نمی‌شوند. روش‌های متعدد برای کدگذاری در این روش استفاده می‌شود از قبیل UCS-2[۲۹] و UCS-4[۳۰] ارائه شده است و روش‌هایی نظیر UTF-8[۳۱] و UTF-16[۳۲] برای کاراتر کردن این نمایش ارائه شده است.

۴.۱ زبان‌های برنامه‌سازی

همانگونه که پیشتر گفته شد، نرم‌افزار، برنامه‌ای است که برای اجرا در کامپیوتر آماده شده است. این برنامه شامل دستورالعمل‌هایی است که برای انجام کار خاصی و بر اساس نیازهای کاربر نوشته شده است. به عبارت دیگر، این دستورات برای رسیدن به هدف برنامه، سخت‌افزار کامپیوتر را کنترل و مدیریت می‌کنند.

برای این که کامپیوتر بتواند دستورات یک برنامه را اجرا کند، لازم است تا دستورات به زبان قابل فهم برای کامپیوتر بیان شوند. این دستورات معمولا به صورت کدهای باینری یا دودویی نوشته می‌شوند. اما نوشتن برنامه به این زبان امری بسیار دشوار و طاقت فرسا است. به همین دلیل، **زبان‌های سطح بالا**[۳۳] ابداع شده اند که برنامه‌نویسی با آن‌ها بسیار ساده‌تر از برنامه‌نویسی با زبان ماشین است. زبانی را زبان سطح بالا گویند که به زبان محاوره‌ای انسان نزدیک‌تر باشد. برای مثال، زبان‌های برنامه‌نویسی مانند پاسکال[۳۴]، سی پلاس پلاس[۳۵]، پایتون[۳۶] و جاوا[۳۷] زبان‌های برنامه‌نویسی سطح بالا هستند.

[۲۸] Basic Multilingual Plane (BMP)
[۲۹] Universal Character Set - 2 Byte
[۳۰] Universal Character Set - 4 Byte
[۳۱] Unicode Transformation Format - 8-bit
[۳۲] Unicode Transformation Format - 16-bit
[۳۳] High-level Languages
[۳۴] PASCAL
[۳۵] C++
[۳۶] Python
[۳۷] Java

برای تبدیل دستورات اینگونه زبان‌ها به زبان قابل فهم برای ماشین، از مترجم‌ها استفاده می‌شود. به صورت کلی، مترجم‌ها بر دو نوع هستند. **کامپایلر**[۳۸]ها و **مفسر**[۳۹]ها. یک کامپایلر، تمام برنامه را به یک باره به زبان ماشین تبدیل می‌کند و در صورت وجود خطای نوشتاری، فهرستی از آن‌ها را در انتهای فرایند کامپایل به کاربر ارائه می‌کند. برنامه ترجمه شده به زبان ماشین در حافظه ثانویه ذخیره شده و برای هر بار استفاده، دیگر نیازی به مترجم و ترجمه کد نیست و برنامه ترجمه شده به صورت مستقیم اجرا می‌شود. اما مفسرها، در هر بار اجرا، برنامه را خط به خط خوانده و بررسی کرده و اجرا می‌کند.

هنگامی که یک برنامه‌نویس مشغول نوشتن یک برنامه است، ممکن است مرتکب برخی خطاها شود، برای مثال، یک دستور را اشتباه تایپ کند. این گونه اشکالات در دسته خطاهای نحوی جای دارند. کامپایلر و مفسر هر دو نسبت به خطاهای نحوی حساس بوده و در صورت بروز این گونه خطاها، اخطارهای لازم را به کاربر ارائه می‌کنند. البته خطاهای نحوی، صرفا شامل تایپ نادرست یک دستور نیست، بلکه خطاهای نحوی شامل کلیه دستوراتی هستند که با دستور زبان یا نحو زبان برنامه‌نویسی مورد استفاده مطابقت نداشته باشند. در صورتی که یک برنامه دارای خطای نحوی باشد، ترجمه آن برنامه به زبان ماشین متوقف می‌شود. به عبارت دیگر، کامپایلر فرایند تولید برنامه قابل اجرا را متوقف کرده و مفسر نیز اجرای دستورات آن برنامه را متوقف می‌کند.

اما دسته دیگری از خطاها، خطاهای منطقی هستند. این گونه خطاها بسیار خطرناک و مشکل ساز هستند و تشخیص آن‌ها در برنامه‌ها امری دشوار و زمان بر است. منشا این گونه خطاها، معمولاً در طراحی منطق برنامه است. برای مثال، در هنگام محاسبه ریشه دوم حقیقی یک عدد منفی و یا تقسیم یک عدد بر صفر، خطای منطقی رخ می‌دهد. چنین خطاهای منطقی در حین اجرای برنامه، منجر به توقف فرایند اجرا می‌شوند و معمولا نیز پیام متناسبی را در خروجی چاپ می‌کنند. این دسته از خطاها را خطاهای زمان اجرا نیز می‌نامند. اما همه خطاهای منطقی به این گونه نیستند. برای مثال فرض کنید برنامه‌ای نیاز به محاسبه مساحت یک مستطیل دارد، اما برنامه‌نویس به اشتباه برنامه‌ای می‌نویسد که محیط یک مستطیل را محاسبه می‌کند. در این صورت نیز خطای منطقی رخ داده است، اما اجرای برنامه مختل نشده و ادامه می‌یابد در صورتی که پاسخ نادرست است. همانگونه که در دنیای واقعی پیشگیری بهتر از درمان است، در دنیای برنامه‌نویسی دقت در زمان طراحی منطق برنامه بسیار ساده‌تر و کم هزینه‌تر از یافتن اشکالات منطقی در برنامه نوشته شده است. به همین دلیل، در فصل ۲، قبل از این که وارد برنامه‌نویسی شویم، روش‌های طراحی منطق برنامه‌ها را بررسی می‌کنیم. در این مسیر نیز، با ابزارهای مختلفی مانند فلوچارت یا نمودار گردشی آشنا می‌شویم که فرایند طراحی منطق برنامه را ساده‌تر و قابل فهم‌تر می‌کنند.

---

۳۸ Compiler

۳۹ Interpreter

تمرین‌ها