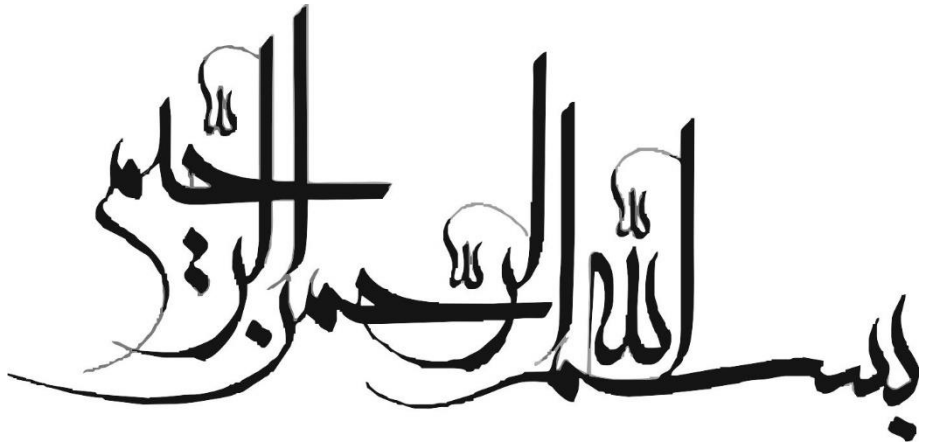


جزوه مبانی برنامه نویسی کامپیوتری

فصل اول- برنامه نویسی و کاربردی آن

۱- ۱- مفهوم برنامه نویسی به زبان ساده

۱- ۲- زبان برنامه نویسی چیست

۱- ۳- تاریخچه زبان های برنامه نویسی

در فرهنگ لغت، **واژه برنامه نویسی** اینگونه تعریف شده است:

برنامه نویسی به فرآیند آماده‌سازی یک برنامه برای یک دستگاه گفته می‌شود که این برنامه از تعدادی دستورالعمل تشکیل شده است.

۱- ۱- مفهوم برنامه نویسی به زبان ساده

به بیان ساده، اساساً برنامه نویسی اقدامی برای به‌کارگیری کامپیوتر جهت انجام یک وظیفه مشخص است که این وظیفه باید بدون خطا و به‌درستی انجام شود.

برنامه نویسی فرآیندی است که در آن برای کامپیوتر تعیین می‌شود کارهای خاصی را انجام دهد. تعیین وظایفی که کامپیوترها باید انجام دهند از طریق تعریف و تعیین تعدادی دستورالعمل انجام می‌شود.

به مجموعه‌ای از این دستورالعمل‌ها که کار خاصی را انجام می‌دهند و خروجی و نتیجه مشخصی دارند، **برنامه**[۱] گفته می‌شود.

فردی که دستورالعمل‌ها را می‌نویسد، برنامه نویس نام دارد. این دستورالعمل‌ها را می‌توان به زبان‌های مختلفی نوشت که به آن‌ها (زبان‌های برنامه نویسی) می‌گویند.

برای درک بهتر مفهوم برنامه نویسی بهتر است در ادامه مثالی ساده ارائه شود.

در اینجا به سراغ یک مثال ساده می‌رویم که درک بهتری از اینکه برنامه‌نویسی چیست فراهم می‌کند. برای مثال فرض می‌شود که شخصی با سطح هوشمندی کم‌تر از باهوش می‌خواهد یک اسباب‌بازی لِگو[۲] را بسازد. این شخص دفترچه راهنمای ساخت لگو را در اختیار ندارد و تنها می‌تواند بر اساس دستورات شما ساخت لگو را انجام دهد. باید به یاد داشت که این شخص فاقد

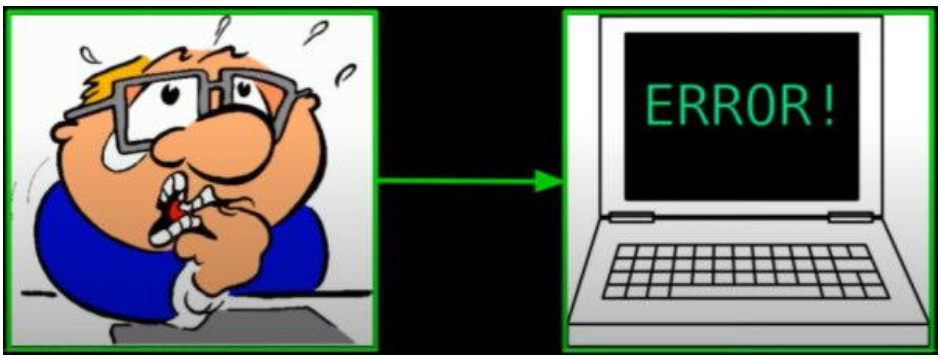
---

[۱] Program

[۲] Leggo

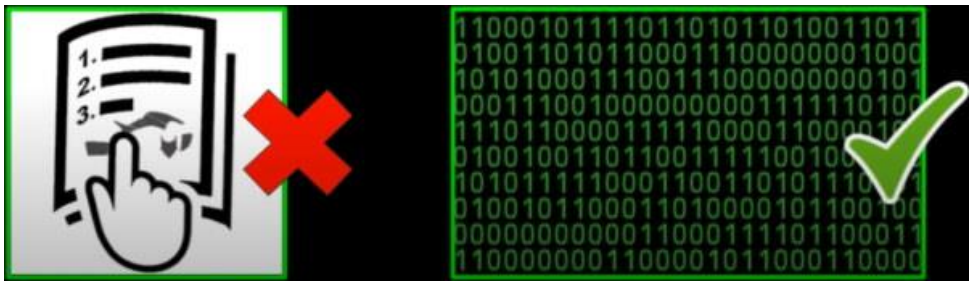
هوشمندی است و در صورتی که دستورالعمل‌های دقیق و مشخصی را در خصوص نحوه ساخت لگو دریافت نکند، به احتمال زیاد اشتباهات بسیاری را مرتکب خواهد شد.

اگر نحوه تفکر این شخص مثل یک کامپیوتر باشد، آنوقت حتی اگر دستورالعمل مربوط به تنها یک قطعه لگو و نحوه قرار دادن آن در محل صحیح به طور مشخص تعیین نشود، کل فرآیند ساخت اسباب‌بازی لگو با مشکل مواجه خواهد شد. در واقع، دستور دادن به این شخص فاقد هوشمندی بسیار شبیه به نحوه انجام برنامه نویسی است. با این تفاوت که در واقعیت به جای یک شخص فاقد هوشمندی، با یک کامپیوتر فاقد هوشمندی سرو کار داریم.



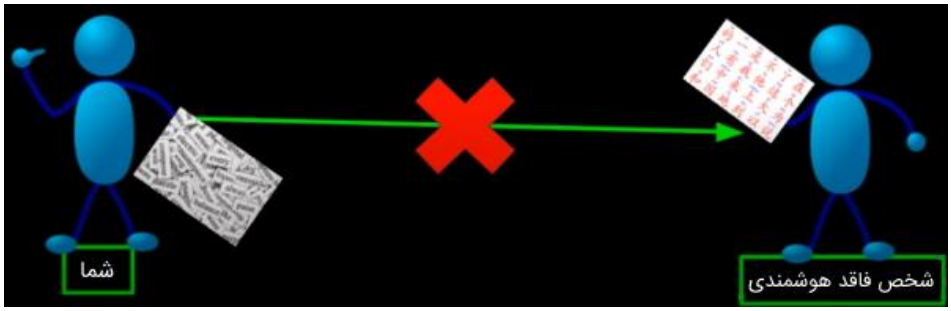
همچنین در برنامه نویسی، به جای دستورالعمل‌های مربوط به نحوه ساخت یک اسباب‌بازی لگو، اطلاعات و دستوراتی در خصوص نحوه تکمیل یک برنامه کامپیوتری مثل یک بازی کامپیوتری یا یک کاربرد تحت وب (وب‌اپلیکیشن) برای کامپیوتر فراهم می‌شوند. نکته مهمی که وجود دارد این است که کامپیوترها فاقد هوشمندی هستند؛ یعنی کامپیوترها به عنوان یک فناوری بسیار پیچیده ساخته شده‌اند، اما در واقعیت، عملکرد اصلی یک کامپیوتر به نحوه مدیریت و فرمان دادن به آن مربوط می‌شود.

البته برنامه نویسی به سادگی دستور دادن به یک شخص فاقد هوشمندی نیست. دلیلش این است که در برنامه نویسی، نمی‌توان به زبان انسان با کامپیوتر ارتباط برقرار کرد. بلکه، کامپیوتر از زبان ماشین استفاده می‌کند .کدهای ماشین یک نوع زبان عددی به حساب می‌آیند که به آن زبان دودویی یا باینری[۳] گفته می‌شود.



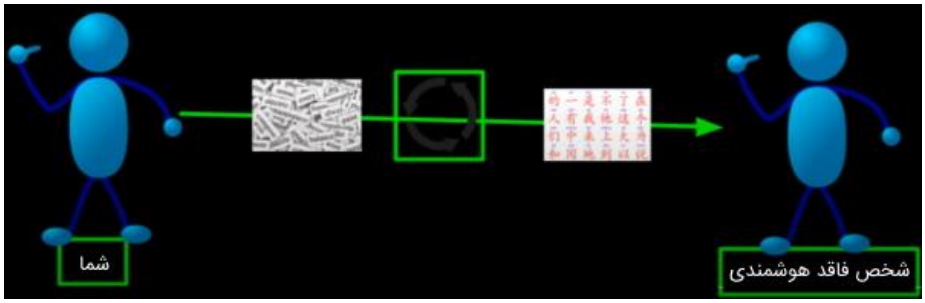
کدهای باینری به گونه‌ای طراحی شده‌اند که کامپیوتر می‌تواند به سرعت آن‌ها را بخواند و دستورالعمل‌های تعیین شده توسط آن‌ها را اجرا کند. هر دستورالعمل ارجاع شده به رشته‌ای متشکل از اعداد صفر و یک تبدیل و این رشته سپس برای اجرای وظیفه مربوطه توسط کامپیوتر تفسیر می‌شود.

برای درک بهتر، به مثال لِگو باز می‌گردیم. در مثال ساخت بازی لگو، اگر شخص مربوطه علاوه بر عدم هوشمندی، زبان ما را هم متوجه نشود و مثلاً به زبان چینی صحبت کند، آنگاه شرایط سخت‌تر خواهد شد.

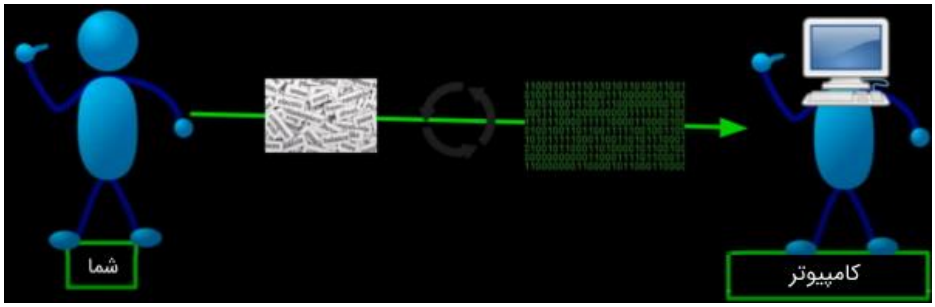


[۳] Binary

در چنین شرایطی برای اینکه بتوانیم با این شخص ارتباط برقرار کنیم، باید دستورالعمل‌ها را از زبان خودمان به زبانی تبدیل کنیم که این شخص متوجه می‌شود.



این فرآیند لزوماً همان کاری است که باید برای کامپیوترها هم انجام شود تا آن‌ها بتوانند دستورالعمل‌ها را متوجه شوند. اگرچه، تفاوت اصلی بین مثال بازی لگو با کامپیوترها این است که درک و فهم کدهای ماشین به صورت دودویی برای انسان‌ها بسیار دشوار و تقریباً غیرممکن است. حتی اگر این کار امکان‌پذیر باشد، فرآیندی بسیار زمان‌بر و طولانی خواهد بود.



هر برنامه حاوی میلیون‌ها کد صفر و یک است، پس دقیقاً چگونه باید دستورالعمل‌ها را به زبان ماشین ترجمه کرد؟ اینجاست که کاربرد و اهمیت «زبان‌های برنامه نویسی» مشخص می‌شود. بنابرین در راستای پاسخ به این سوال که برنامه نویسی چیست باید به این سوال هم پاسخ داده شود که زبان برنامه نویسی چیست ؟

۱- ۲- زبان برنامه نویسی چیست

زبان‌های برنامه نویسی اساساً برای ترجمه یک برنامه به کدهای ماشین به مانند یک واسط عمل می‌کنند. یادگیری زبان‌های برنامه نویسی نسبت به یادگیری کدهای صفر و یک ماشین بسیار ساده‌ترند و بنابراین برای برنامه نویسان بسیار مفید و کاربردی هستند.

در مثال لگو، یک زبان برنامه نویسی به نوعی شبیه به یک مترجم عمل می‌کند؛ این مترجم می‌تواند دستورالعمل‌های دریافتی به زبان انسان را به دستورالعمل‌های قابل تشخیص برای شخصی تبدیل کند که به زبان دیگری صحبت می‌کند. می‌توان زبان‌های برنامه نویسی را چیزی بین زبان ماشین و زبان محاوره انسان‌ها تصور کرد.

یک زبان برنامه نویسی مجموعه‌ای از قوانین است که رشته‌ها (یا در زبان‌های برنامه نویسی بصری یاVisual ، اجزای گرافیکی) را به کدهای ماشین تبدیل می‌کند تا عملیات مورد نظر در ماشین اجرا شوند .به بیان دقیق‌تر، زبان برنامه نویسی نوعی نظام نشانه‌گذاری نوشتاری (یا گرافیکی) است که برای ارتباط میان انسان و ماشین استفاده می‌شود .زبان‌های برنامه نویسی نوعی «زبان کامپیوتری (Computer Language) «به حساب می‌آیند و در «برنامه نویسی کامپیوتری» برای پیاده‌سازی الگوریتم‌ها مورد استفاده قرار می‌گیرند.

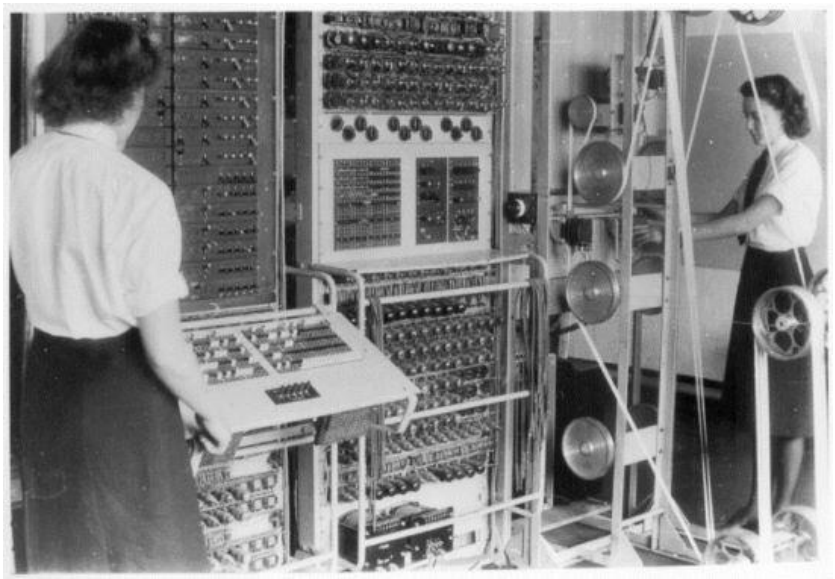
زبان های برنامه نویسی انواع مختلفی دارند و هر کدام با یک روش و رویکرد خاص پیش می‌روند و وظایف خود را انجام می‌دهند، همچنین هر کدام برای کاربردهای خاصی ایجاد شده‌اند.

۱- ۳- تاریخچه زبان های برنامه نویسی

در این بخش سعی شده است تا موارد مهم پیرامون تاریخچه زبان‌های برنامه نویسی تا جای ممکن به‌طور جامع و به بیان ساده شرح داده شوند.

ابداع های اولیه، سرآغاز تاریخچه زبان های برنامه نویسی

ابتدایی‌ترین کامپیوترها نظیر Colossus بدون کمک یک برنامه ذخیره شده برنامه‌ریزی می‌شدند. برنامه‌ریزی کامپیوترهای اولیه به وسیله ویرایش مدارهای آن‌ها یا تنظیم مجموعه کنترل‌های فیزیکی آن‌ها انجام می‌شد.

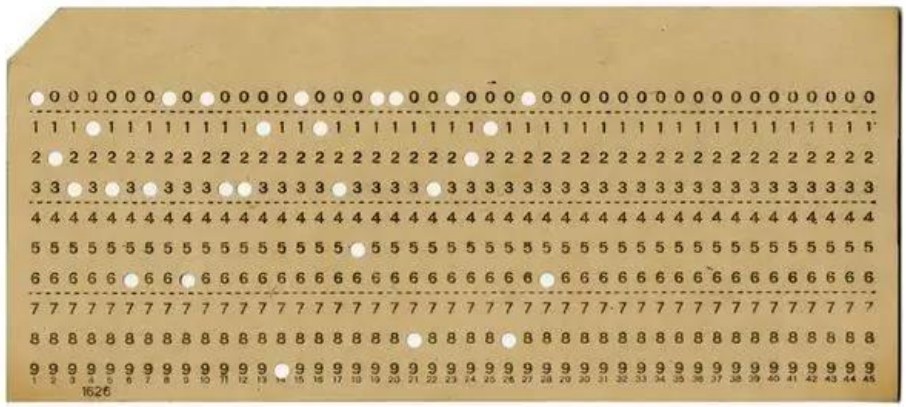


کامپیوترColossus

کمی بعدتر، امکان نوشتن برنامه‌ها به «زبان ماشین (Machine Language) «فراهم شد. در این روش، برنامه نویس هر دستورالعمل را به صورت عددی می‌نوشت؛ بنابراین، سخت‌افزار می‌تواند آن دستورالعمل را به‌طور مستقیم اجرا کند. برای مثال، دستورالعمل اضافه کردن مقدارهای دو موقعیت حافظه ممکن است سه عدد را شامل شوند:

- یک «opcode» (کد عملیات) که عملیات جمع (Add) را انتخاب می‌کند.
- دو عدد بعدی هم به دو موقعیت حافظه‌ای مربوط می‌شوند که مقادیر مورد نظر در آن‌ها ذخیره شده‌اند.

برنامه‌هایی که در قالب دهدهی یا دودویی هستند، از طریق کارت‌های پانچ شده(Punched Card) ، نوارهای کاغذی، نوارهای آهنربایی یا از طریق سوئیچ‌هایی در پنل جلویی وارد کامپیوتر می‌شدند. بعدها زبان‌های ماشین با عنوان زبان‌های برنامه نویسی نسل اول (First-Generation Programming Languages)یا به اختصار «1GL» نامیده شدند.

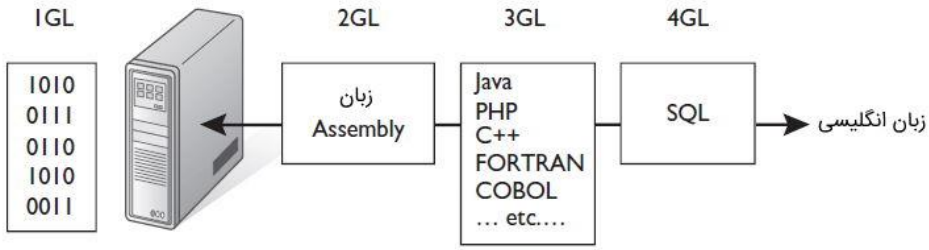


نمونه‌ای از کارت پانچ

گام بعدی، توسعه و خلق زبان‌هایی بود که آن‌ها را زبان‌های برنامه نویسی نسل دوم (2GL) یا زبان‌های اسمبلی (Assembly Languages) نامیدند. این زبان‌ها نیز همچنان به معماری مجموعه دستورالعمل‌های یک کامپیوتر خاص وابسته بودند. زبان‌های اسمبلی (Assembly) باعث شدند که امکان خواندن برنامه توسط انسان تسهیل شود و برنامه نویسان هم از انجام محاسبه‌های ملالت‌بار مربوط به آدرس‌دهی خلاص شدند.

اولین «زبان‌های برنامه نویسی سطح بالا (High Level) «یا همان «زبان‌های برنامه نویسی نسل سوم (3GL) «در سال دهه ۱۳۳۰ شمسی (۱۹۵۰ میلادی) پدید آمدند. یکی از اولین زبان‌های برنامه نویسی سطح بالایی که طراحی شد،) «Plankalkül»پلن‌کلکولوس) نام دارد که برای کامپیوتر آلمانی Z3 در بازه سال‌های ۱۳۲۱ تا ۱۳۲۳ توسعه داده شده است. البته پیاده‌سازی آن تا

سال ۱۳۷۶ (۱۹۹۸) به تعویق افتاد. زبان «Short Code» نوشته توسط جان ماکلی (John Mauchly) در سال ۱۳۲۷ ارائه شده است.



نسل‌های مختلف زبان‌های برنامه نویسی

Short Codeیکی از اولین زبان‌های برنامه نویسی سطح بالایی به حساب می‌آید که تا کنون برای یک کامپیوتر الکترونیک توسعه داده شده است. برخلاف کدهای ماشین، امکان نمایش عبارت‌های ریاضی به شکلی قابل درک در گزاره‌های زبان Short Code وجود داشت. اگرچه، در هر بار اجرا، برنامه باید به کدهای ماشین ترجمه می‌شد که این مسئله باعث می‌شد روال کار نسبت به اجرای مستقیم کدهای ماشین بسیار کندتر انجام شود.

در اوایل دهه ۱۳۳۰ شمسی (۱۹۵۰ میلادی)، «آلک گلنی (Alick Glennie) «اولین زبان از مجموعه سیستم‌های کدنویسی ساده شده «Autocode» را توسعه داد. این Autocode به عنوان یک زبان برنامه نویسی، از یک کامپایلر (ترجمه کننده) برای تبدیل خودکار کدهای نوشته شده با این زبان به کدهای ماشین استفاده می‌کرد. اولین کدها و اولین کامپایلر در سال ۱۳۳۰ (۱۹۵۲ میلادی) برای کامپیوتر Mark 1 در دانشگاه منچستر توسعه داده شد Mark 1 .اولین زبان برنامه نویسی سطح بالای کامپایلی به حساب می‌آید.

دومین Autocode هم در سال ۱۳۳۲ شمسی (۱۹۵۴ میلادی) توسط تونی بروکر (Tony Brooker)برای کامپیوتر Mark 1 توسعه داده شد و نام «Mark 1 Autocode» را برای آن انتخاب

کردند. علاوه بر این، تونی بروکر Autocode دیگری را هم برای کامپیوتر تجاری Ferranti Mercuryدر سال ۱۹۵۰ در ارتباط با دانشگاه منچستر توسعه داده است.

کامپیوتر Ferranti Mercury

نسخه Autocode مربوط به کامپیوتر EDSAC 2 توسط «دیوید هارتلی (David Hartley) «از آزمایشگاه ریاضی دانشگاه کمبریج در سال ۱۳۳۹ (۱۹۶۱) ابداع شد. این نسخه که با نام «EDSAC 2 Autocode»شناخته می‌شد، توسعه‌ای مستقیم از Mercury Autocode به حساب می‌آمد که برای شرایط محلی مربوطه تطبیق داده شده بود. این Autocode برای بهینه‌سازی کدهای شی و امکانات عیب‌شناسی زبان منبع مورد توجه قرار گرفته بود که در آن زمان بسیار پیشرفته بودند. یک رشته از توسعه‌های امروزی اما مستقل، «Atlas Autocode» نام دارد که برای ماشین Atlas 1 در دانشگاه منچستر به حساب می‌آید.

در سال ۱۹۵۴، زبان برنامه نویسی FORTRAN در شرکت IBM توسط «جان باکوس (John Backus)» ابداع شد. زبان FORTRAN اولین زبان سطح بالای پراستفاده‌ی همه‌منظوره‌ای به حساب

می‌آید که سیستم اجرایی آن تابع محور است. این زبان برنامه نویسی همچنان به عنوان یک گزینه محبوب برای انجام محاسبات با حجم زیاد محسوب می‌شود. زبان FORTRAN برای ایجاد برنامه‌هایی مورد استفاده قرار می‌گیرد که در ارزیابی عملکرد و رتبه‌بندی سریع‌ترین ابرکامپیوترهای جهان به کار گرفته می‌شوند.

یکی دیگر از زبان‌های برنامه نویسی اولیه، توسط «گرِیس هاپر» در ایالات متحده ابداع شده است که «FLOW-MATIC» نام دارد. این زبان برنامه نویسی در شرکت Remington Rand برای کامپیوتر UNIVACدر بازه سال‌های ۱۹۵۵ تا ۱۹۵۹ ساخته شد. هاپر متوجه شد که مشتریان داده‌های تجاری با نشانه‌گذاری ریاضیاتی چندان راحت نبودند و در اوایل سال ۱۹۵۵، او و اعضای تیمش، مشخصاتی را برای یک زبان برنامه نویسی انگلیسی نوشتند و نمونه اولیه‌ای از آن را پیاده‌سازی کردند.

زبان برنامه نویسیFortran

کامپایلر FLOW-MATIC در اوایل سال ۱۹۵۸ به صورت عمومی در دسترس قرار گرفت و به‌طور قابل ملاحظه‌ای در سال ۱۹۵۹ تکمیل شده بود FLOW-MATIC .در طراحی و خلق زبان برنامه نویسی COBOL تاثیر به‌سزایی داشته است، چرا که تنها FLOW-MATIC و بازمانده مستقیمش یعنی AIMACOدر آن زمان مورد استفاده قرار می‌گرفتند.

تاریخچه زبان های برنامه نویسی، اصلاحات انجام شده

استفاده روزافزون از زبان‌های برنامه نویسی سطح بالا، ضرورت به‌کارگیری «زبان‌های برنامه نویسی سطح پایین (Low Level) یا همان زبان‌های برنامه نویسی سیستمی را به وجود آورد. این نوع زبان‌ها، امکانات تسهیل کننده‌ای را در سطوح مختلف بین زبان‌های اسمبلی و زبان‌های سطح بالا فراهم کرده‌اند. زبان‌های سیستمی می‌توانند برای اجرای وظایفی استفاده شوند که نیاز به دسترسی مستقیم به سخت‌افزار دارند؛ اما با این حال همچنان ساختارهای کنترلی و امکانات بررسی خطای سطح بالاتری را فراهم می‌کنند.

در بازه دهه‌های ۱۹۶۰ تا ۱۹۷۰ توسعه الگوهای زبان‌های برنامه نویسی کلیدی انجام شده است که اکنون مورد استفاده هستند. این الگوها یا همان پارادایم‌های کلیدی برنامه نویسی در ادامه به همراه زبانی فهرست و معرفی شده که این الگوها برای اولین بار در آن‌ها به‌کار گرفته شده‌اند:

- در زبانAPL ، رویکرد برنامه نویسی آرایه‌ای معرفی شد که در خلق الگوی برنامه نویسی تابعی از آن تاثیر گرفته شده است.
- در زبان ALGOL هم «برنامه نویسی ساخت‌یافته رویه‌ای و اصول قاعده‌مند زبان برنامه نویسی بهبود داده شدند. گزارش اصلاح‌شده در خصوص زبان الگوریتمی ALGOL 60 بعداً به مدلی برای نحوه نوشتن اصول زبان‌های برنامه نویسی تبدیل شد.
- زبان Lisp که در سال ۱۳۳۷ شمسی (۱۹۵۸ میلادی) پیاده‌سازی شد، اولین زبان برنامه نویسی تابعی دارای قابلیت بررسی نوع به صورت پویا (نوع‌دهی پویا) محسوب می‌شود.
- در دهه ۱۳۴۰ (۱۹۶۰ میلادی)، Simula، اولین زبان برنامه نویسی برای پشتیبانی از برنامه نویسی شی‌گرا طراحی شد. در اواسط دهه ۱۳۵۰ (۱۹۷۰ میلادی)، زبان Smalltalkبه عنوان اولین زبان برنامه نویسی شی‌گرای خالص معرفی شد.

زبان برنامه نویسیLisp

- زبان C بین سال‌های ۱۳۴۸ و ۱۳۵۲ به عنوان یک زبان برنامه نویسی سیستمی برای سیستم عامل یونیکس توسه داده شده است و همچنان زبان محبوبی به حساب می‌آید.
- زبان برنامه نویسی Prolog که در سال ۱۳۵۱ (۱۹۷۲ میلادی) طراحی شده است، اولین زبان برنامه نویسی منطقی به شمار می‌رود.
- در سال ۱۳۵۷ (۱۹۷۸ میلادی)، سازندگان زبانML ، سیستمی با نوع چندریختی بر پایه زبان Lisp ساختند. این ابداع به خلق زبان‌های تابعی با قابلیت بررسی نوع (نوع‌دهی) به صورت ایستا منجر شد.

با الهام گرفتن از هر یک از زبان‌های برنامه نویسی فهرست شده فوق، زبان‌های جدیدی ایجاد شده‌اند. اکثر زبان‌های برنامه نویسی مدرن و امروزی، حداقل یکی از زبان‌های فوق را به عنوان سرمنشاء خود به حساب می‌آورند.

در دهه‌های ۱۳۴۰ (۱۹۶۰) و ۱۳۵۰ (۱۹۷۰) نیز بحث‌های قابل توجهی پیرامون مزایای برنامه نویسی ساختاریافته (Structured Programming) و این مسئله وجود داشته است که آیا زبان‌های برنامه نویسی باید برای پشتیبانی از این قابلیت طراحی شوند؟ در این خصوص، «ادسخر دیکسترا «

(Edsger Dijkstra)در مقاله مشهوری که در مجله «ارتباط‌های «ACM منتشر شده است، این بحث را مطرح کرد که گزاره‌های Goto باید از زبان‌های برنامه نویسی سطح بالاتر حذف شوند.

ادسخر دیکسترا — از پیشگامان علوم کامپیوتر

یکپارچه سازی و رشد زبان های برنامه نویسی

دهه ۱۳۶۰ (۱۹۸۰ میلادی) سال‌های تثبیت نسبی زبان‌های برنامه نویسی به حساب می‌آید. در زبان برنامه نویسی C++، شی‌گرایی و برنامه نویسی سیستمی با یکدیگر تلفیق شدند. دولت ایالات متحده آمریکا زبان Ada را استانداردسازی کرد .یک زبان برنامه نویسی سیستمی به حساب می‌آید که از پاسکال مشتق شده و با هدف استفاده توسط پیمانکاران امور دفاعی ایجاد شده است.

زبان های برنامه نویسی نسل پنجم

در ژاپن و سایر نقاط جهان، هزینه‌های هنگفتی صرف زبان‌هایی شده است که به آن‌ها زبان‌های برنامه نویسی نسل پنجم گفته می‌شود. زبان‌های نسل پنجم، ساختارهای برنامه نویسی منطقی را به کار می‌گیرند. جامعه زبان‌های برنامه نویسی تابعی به سمت استانداردسازی ML و Lisp حرکت کردند. به جای اختراع و خلق الگوهای جدید برنامه نویسی، تمام این جابجایی‌ها و حرکت‌ها براساس ایده‌های خلق شده در دهه‌های گذشته شکل گرفته‌اند.

طراحی زبان های برنامه نویسی برای کامپیوترهای بزرگ

یک رویکرد مهم در طراحی زبان برای برنامه نویسی برای سیستم‌های با مقیاس بزرگ در طول دهه ۱۳۶۰ (۱۹۸۰)، تمرکز افزوده بر استفاده از ماژول‌هایی با مقیاس بزرگ و در سطح سازمانی بوده است. به وسیله زبان‌های Ada ، Modula-2وML ، سیستم‌های ماژولی قابل توجهی در دهه ۱۳۶۰ ساخته شده است که اغلب با ساختارهای برنامه نویسی عام (Generic Programming) مرتبط می‌شدند.

فراهم شدن بستر مناسب برای خلق زبان های برنامه نویسی جدید

رشد سریع اینترنت در اواسط دهه ۱۳۷۰ (۱۹۸۰ میلای) باعث شد که فرصت‌هایی برای خلق زبان‌های برنامه نویسی جدید به وجود بیایند. زبان برنامه نویسی Perl که در اصل یک ابزار اسکریپت‌نویسی مبتنی بر یونیکس به حساب می‌آید و اولین بار در سال ۱۳۶۵ (۱۹۸۷) منتشر شد، در ساخت وب‌سایت‌های پویا بسیار رواج پیدا کرد. جاوا هم در حوزه برنامه نویسی سمت سرور محبوب شد و همچنین ماشین‌های مجازی بایت‌کُد (Bytecode) بار دیگر در کاربردهای تجاری با این تضمین که «یک بار بنویس و همه جا اجرایش کن» رایج شدند.

برای مثال، UCSD Pascal برای مدتی در اوایل دهه ۱۹۸۰ محبوبیت پیدا کرده بود. این ابداع‌ها و دست‌آوردها اساساً چندان تازه و جدید نبودند. در عوض، آن‌ها اصلاحاتی از بسیاری از زبان‌ها و الگوهای فعلی برنامه نویسی به حساب می‌آمدند. البته سینتکس این زبان‌ها اغلب براساس خانواده زبان‌های برنامه نویسی C ایجاد شده بود.

تحولات زبان‌های برنامه نویسی هم در صنعت و هم در تحقیقات علمی ادامه داشته‌اند. مسیرهای فعلی این تحولات، مباحث امنیت و تایید قابلیت اطمینان، انواع جدید ماژولی بودن(Mixin) ها، واگذاری‌ها یاDelegate ها، جنبه‌ها یاAspect ها (و یکپارچه‌سازی پایگاه داده‌ها مثل زبان برنامه نویسی LINQ شرکت مایکروسافت را شامل می‌شوند.

پیدایش زبان های برنامه نویسی نسل چهارم

زبان‌های برنامه نویسی نسل چهارم (4GL) زبان‌های کامپیوتری هستند که هدف آن‌ها فراهم کردن سطح بالاتری از انتزاع نسبت به جزئیات سخت‌افزاری داخلی کامپیوترها در زبان‌های نسل سوم است. زبان‌های برنامه نویسی نسل پنجم به جای استفاده از الگوریتم‌های ساخته شده توسط برنامه نویس، مبتنی بر حل مسئله با استفاده از محدودیت‌های داده شده به برنامه هستند. اکنون در ادامه پاسخ به این سوال که زبان برنامه نویسی چیست، توضیحات لازم پیرامون اجزای زبان‌های برنامه نویسی ارائه شده‌اند.

فصل دوم- دسته بندی زبان‌های برنامه نویسی

۲- ۱- سطح بالا یا سطح پایین

۲- ۲- دسته بندی دیگر زبان های برنامه نویسی

۲- ۲- ۱- زبان های برنامه نویسی رویه‌ای

۲- ۲- ۲- زبان های برنامه نویسی تابعی چه هستند؟

۲- ۲- ۳- زبان های برنامه نویسی شی گرا کدامند؟

۲ -۲- ۴- زبان های برنامه نویسی اسکریپتی

۲- ۲- ۵- زبان های برنامه نویسی منطقی

۲- ۲- ۶- زبان های برنامه نویسی پایگاه داده‌ای

۲- ۲- ۷- زبان های برنامه نویسی جریان داده

۲- ۳- انواع برنامه نویسی بر اساس پلتفرم

۲- ۴- کامپایلر

۲- ۵- مُفسِر

۲- ۶- کامپایلر و مفسر

انواع زبان‌های برنامه نویسی را می‌توان به دو دسته کلی زبان‌های سطح بالا و سطح پایین تقسیم کرد.

۲- ۱- سطح بالا یا سطح پایین

زبان برنامه نویسی سطح بالا یعنی چه ؟

زبان‌های کدنویسی سطح بالا، سطح بالاتری از انتزاع(تجرید| Abstraction) دارند. انتزاع به معنی این است که آن زبان‌ها به زبان انسان نزدیک‌تر و از کد ماشین دورتر هستند، پس برای انسان قابل درک‌ترند. یادگیری و استفاده از زبان‌های سطح بالا آسان‌تر است، اما معمولاً قابلیت و کنترل مستقیم کمتری را روی رایانه ارائه می‌دهند.

زبان‌های برنامه نویسی سطح بالا غالباً خودکارتر هستند در واقع ان‌جا است که یک فرمان برنامه نویسی، بسیاری از کارهای پیش برنامه‌ریزی شده را انجام می‌دهد تا برنامه نویسی آسان‌تر و کارآمدتری را ایجاد کند. علاوه بر این، زبان‌های سطح بالا به مدیریت حافظه و مدیریت حافظه پردازنده نیاز ندارند. این زبان‌ها به طور مستقیم برای ماشین قابل درک نیستند و باید به کدهای سطح پایین تبدیل شوند که به آن ترجمه کد می‌گویند. سپس کدهای سطح پایین به دست آمده روی ماشین اجرا می‌شوند.

زبان‌های برنامه نویسی سطح بالا کدامند ؟

این زبان برنامه نویسی بسیار به زبان انسان نزدیک است و دستورهای آن به مکالمات انسانی شباهت دارد. زبان برنامه نویسی C به عنوان مادر بسیاری از زبان‌های دیگر محسوب می‌شود، زیرا زبان‌های سطح بالای زیادی از آن مشتق شده‌اند. برخی از زبان‌های سطح بالا و محبوب مشتق شده از آن، روبی، سی‌شارپ، R، جاوا اسکریپت(JavaScript) ، پایتون، PHP و جاوا به حساب می‌آیند.

زبان برنامه نویسی سطح پایین یعنی چه ؟

زبان‌های برنامه نویسی سطح پایین، سطح کم‌تری از انتزاع دارند و برعکس زبان‌های سطح بالا هستند. این نوع از زبان‌های کدنویسی به کد باینری نزدیک‌ترند و با زبان انسانی بیش‌تر فاصله دارند. زبان‌های سطح پایین، نزدیک به سخت افزار اجرا می‌شوند به همین علت در نوشتن کدها باید به جزئیات سخت افزاری مانند مدیریت حافظه، فراخوانی اطلاعات از حافظه پردازنده و موارد دیگر هم توجه شود.

یادگیری و استفاده از زبان‌های برنامه نویسی سطح پایین سخت‌تر است، اما کارآمدی و کنترل بیش‌تری را روی کامپیوتر ارائه می‌دهند. این نوع از زبان به برنامه نویس‌ها اجازه می‌دهد تا نرم افزار کامپیوتری کارآمدتر و دقیق‌تری ایجاد کنند.

در ادامه دسته‌بندی دقیق‌تری از انواع زبان برنامه نویسی فهرست شده است:

۱. **زبان برنامه نویسی سطح پایین :** این زبان قابل درک‌ترین نوع زبان برای کامپیوتر به حساب می‌آید که می‌توان آن را به روش‌های زیر دسته‌بندی کرد:

I. **زبان ماشین :(1GL)** این نوع زبان رشته‌هایی از اعداد دودویی را شامل می‌شود و تنها زبانی است که به طور مستقیم برای پردازنده کامپیوتر یا همان CPU قابل درک است.

II. **زبان اسمبلی :(2GL)** این زبان هم نوعی از زبان‌های سطح پایین به حساب می‌آید، چرا که برای طراحی یک برنامه با این زبان، برنامه نویس باید اطلاعات جزئی را در خصوص مشخصات سخت‌افزاری در اختیار داشته باشد.

۲. **زبان برنامه نویسی سطح بالا :** دستورالعمل‌های این نوع زبان برنامه نویسی شباهت و نزدیکی زیادی به زبان انسان یا همان زبان انگلیسی دارند. در زبان سطح بالا از نشانه‌گذاری ریاضی برای اجرای وظایف استفاده می‌شود. یادگیری زبان سطح بالا بسیار آسان‌تر است. زبان‌های سطح بالا را می‌توان به بخش‌های زیر دسته‌بندی کرد:

I. **زبان رویه محور :(3GL)** برنامه نویسی رویه محور یا همان رویه‌ای، روشی است برای مدل‌سازی مسئله از طریق مشخص کردن گام‌ها و ترتیب آن گام‌هایی که باید برای رسیدن به نتیجه مطلوب یا وضعیت خاصی در برنامه پیمایش شوند.

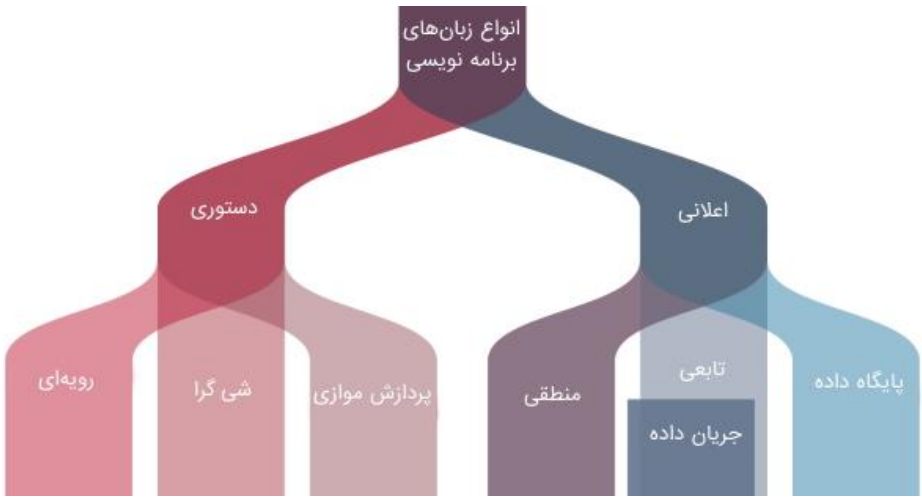
II. **زبان مسئله محور :(4GL)** در این نوع زبان به کاربران اجازه داده می‌شود تا بدون توصیف تمام جزئیات مروبط به نحوه اعمال تغییرات روی داده‌ها، با هدف تولید نتیجه، مشخص کنند که خروجی باید چه چیزی باشد. این یک گامی فراتر از 3GLبه حساب می‌آید.

III. **زبان طبیعی :(5GL)** زبان‌های طبیعی همچنان در مرحله توسعه و ساخت هستند. در این نوع از زبان‌ها می‌توان گزاره‌هایی را نوشت که شبیه به جملات عادی به نظر می‌رسند.

۲- ۲- دسته بندی دیگر زبان های برنامه نویسی

انواع زبان برنامه نویسی را می‌توان به شکل دیگری هم دسته‌بندی کرد که در ادامه ملاحظه می‌شود:

- زبان برنامه نویسی رویه‌ای
- زبان برنامه نویسی تابعی
- زبان برنامه نویسی شی گرا
- زبان برنامه نویسی اسکریپتی
- زبان برنامه نویسی منطقی
- زبان برنامه نویسی پایگاه داده‌ای
- زبان برنامه نویسی جریان داده



۲- ۲- ۱- زبان های برنامه نویسی رویه‌ای

زبان های برنامه نویسی رویه‌ای (Procedural Programming) برای اجرای دنباله‌ای از عبارت‌های مورد استفاده قرار می‌گیرند که منجر به نتیجه می‌شوند. به طور معمول، این نوع زبان های برنامه نویسی از متغیر چندگانه(Multiple Variable) ، حلقه‌های سنگین (Heavy loop) و عناصر دیگر استفاده می‌کنند که آن‌ها را با زبان های برنامه نویسی تابعی (Functional Programming) متمایز می‌کنند. توابع زبان های برنامه نویسی رویه‌ای ممکن است متغیرها را به غیر از مقدار بازگشتی تابع، مانند چاپ کردن اطلاعات، کنترل کنند. برای مثال زبان های برنامه نویسی جاوا اسکریپت (JavaScript)، Cو C++ زبان های برنامه نویسی رویه‌ای به حساب می‌آیند.

۲- ۲- ۲- زبان های برنامه نویسی تابعی چه هستند؟

زبان های برنامه نویسی تابعی معمولاً از داده‌های ذخیره شده استفاده می‌کنند و اغلب در آن‌ها از توابع بازگشتی به جای حلقه‌ها استفاده می‌شود. تمرکز اولیه برنامه نویسی تابعی روی مقادیر بازگشتی توابع است. به عنوان مثال، در یک زبان برنامه نویسی تابعی، اگر تابعی ایجاد و نام‌گذاری شود، انتظار می‌رود که دیگر هیچ‌گونه تغییری در عملکرد آن انجام نشود. با این حال ممکن است فراخوانی الگوریتمی نیز ایجاد کند و پارامترهای این فراخوانی‌ها تغییر پیدا کنند. زبان های برنامه نویسی تابعی معمولاً ساده‌تر از انواع زبان های برنامه نویسی دیگر هستند و به راحتی می‌توان به مسائل انتزاعی با استفاده از آن‌ها در برنامه نویسی اشاره کرد. برای مثال در این نوع از زبان های برنامه نویسی می‌توان زبان های اف شارپ(#F) ، هسکل (Haskell) و اسکیم (Scheme) را نام برد.

۲- ۲- ۳- زبان های برنامه نویسی شی گرا کدامند؟

زبان های برنامه نویسی شی گرا(Object Oriented Programming | OOP) ، همه چیز را به عنوان گروهی از اشیاء می‌بینند که دارای داده‌های داخلی و دسترسی خارجی به بخش‌هایی از آن داده‌ها است. هدف این نوع زبان برنامه نویسی تفکیک کدها به مجموعه‌ای از اشیاء و ارائه خدماتی برای حل یک مشکل خاص به حساب می‌آید. یکی از اصول اصلی زبان برنامه نویسی شی گرا،

کپسوله سازی (Encapsulation) به شمار می‌رود، یعنی هر چیزی که یک شی به آن نیاز دارد باید در داخل شی وجود داشته باشد. همچنین این نوع از زبان های برنامه نویسی قابلیت استفاده مجدد از طریق وراثت (Inheritance) و چندریختی یا همان پلی مورفیسم (Polymorphism) را دارند .زبان برنامه نویسی جاوا(Java) ، ویژوال بیسیک(Visual Basic) ، روبی (Ruby) و پایتون نمونه‌هایی از این نوع زبان برنامه نویسی هستند.

۲ -۲- ۴- زبان های برنامه نویسی اسکریپتی

زبان های برنامه نویسی اسکریپتی (Scripting Programming Language)، نوعی از زبان های برنامه نویسی رویه‌ای هستند و گاهی شامل عناصر زبان های برنامه نویسی شی‌گرا نیز می‌شوند، اما دسته‌بندی مخصوص به خودشان را دارند، زیرا معمولاً زبان‌های برنامه نویسی کاملی نیستند که از توسعه سیستم‌های بزرگ پشتیبانی کنند. به عنوان مثال، این نوع از زبان های برنامه نویسی معمولاً قادر به چک کردن نوع زمان کامپایل نیستند. زبان های برنامه نویسی اسکریپتی برای شروع کار و یادگیری به مهارت ساختار نوشتاری کمی نیاز دارند. برای مثال می‌توان به زبان برنامه نویسی جاوا اسکریپت، پرل(Perl) ، پی اچ پی (PHP) و پایتون اشاره کرد.

۲- ۲- ۵- زبان های برنامه نویسی منطقی

زبان های برنامه نویسی منطقی (Logic Programming Language) به برنامه نویسان این امکان را می‌دهند که عبارت‌های اعلانی (Declarative) ایجاد کنند، سپس به ماشین اجازه می‌دهند درباره پیامدهای آن عبارت‌های استدلال کنند. به صورت کلی می‌توان گفت که این نوع از زبان های برنامه نویسی به رایانه‌ها نمی‌گویند چگونه کاری را انجام دهند، بلکه محدودیت‌هایی را در مورد انجام وظایف، اعمال می‌کنند. برای مثال زبان برنامه نویسی پرولاگ(Prolog) ، یک زبان برنامه نویسی منطقی است.

۲- ۲- ۶- زبان های برنامه نویسی پایگاه داده‌ای

زبان های برنامه نویسی پایگاه داده‌ای (Database Programming Languages) به ایجاد پایگاه داده و اصلاح و تغییر نحوه ذخیره‌سازی داده‌ها در پایگاه داده‌ها کمک می‌کنند. به عنوان مثال می‌توان به زبان برنامه نویسی جاوا، کوبول(Cobol) ، «C++»و «Perl» اشاره کرد.

۲- ۲- ۷- زبان های برنامه نویسی جریان داده

زبان های برنامه نویسی جریان داده (Dataflow languages) از نمایشی برای مبادله داده‌ها جهت مشخص کردن برنامه‌ها و پردازش جریان‌های داده استفاده می‌کنند. پردازش موازی (Parallel processing)روشی در محاسبه اجرای برنامه‌ها در دو یا چند پردازنده (CPU) برای رسیدگی به بخش‌های جداگانه یک وظیفه کلی است. در حقیقت می‌توان گفت که بیشتر زبان های برنامه نویسی شامل ایده‌ها و ویژگی‌هایی در حوزه‌های مختلف هستند که به افزایش سودمندی این نوع زبان‌ها کمک می‌کنند. با این وجود، بیشتر زبان های برنامه نویسی در همه سبک‌های برنامه نویسی بهترین نیستند.

۲- ۳- انواع برنامه نویسی بر اساس پلتفرم

منظور از انواع برنامه نویسی بر اساس پلتفرم، محل استفاده از کدها و پلتفرمی (سکو |Platform) است که برنامه روی آن اجرا می‌شود. پلتفرم به معنی گروهی از فناوری‌ها است که به عنوان پایه‌ای برای توسعه دیگر اپلیکیشن‌ها، فرآیندها یا فناوری‌ها استفاده می‌شوند. در ادامه انواع برنامه نویسی بر اساس پلتفرم‌های اصلی دسته‌بندی شده است:

- برنامه نویسی دسکتاپ:
  - ویندوز
  - لینوکس
  - مکOS
- برنامه نویسی موبایل
  - اندروید

- IOS
- برنامه نویسی تحت وب

شکل ۱: برنامه نویسی حوزه دسکتاپ

شکل ۲: برنامه نویسی حوزه موبایل

شکل ۳: برنامه نویسی و توسعه بازی های رایانه ای

شکل ۴: برنامه نویسی تحت وب

وقتی صحبت از زبان‌های برنامه‌نویسی مختلف می‌شود، اساساً می‌توان آن‌ها را به دو دسته کلی تقسیم نمود: آن‌هایی که کامپایل می‌شوند و آن‌هایی که اینترپریت می‌گردند به طوری که از جمله زبان‌هایی که به دستهٔ اول تعلق دارند می‌توان جاوا یا سی‌شارپ را مثال زد و از جمله زبان‌هایی که تفسیر می‌شوند هم می‌توان به پی‌اچ‌پی و جاوااسکریپت اشاره کرد. حال در همین راستا در ادامه قصد داریم تا تفاوت‌های مابین فرایندهای کامپایلر[۴] با **مُفسِر**[5] را جویا شویم.

۲- ۴- کامپایلر

کامپایلر یک برنامه کامپیوتری است که کدهای یک زبان برنامه‌نویسی سطح بالا را به کدی خوانا برای ماشین تبدیل می‌کند. به عبارتی، برنامه‌ای است که کدهای قابل خواندن توسط انسان را به زبانی که پردازنده‌های کامپیوتر قادر به درک آن باشند (یعنی کدهای باینری یا همان صفر و یک) تبدیل می‌کند.

یک کامپایلر باید با سینتکس زبان برنامه‌نویسی که کدهای برنامهٔ مذکور با آن نوشته می‌شوند آشنایی داشته باشد اما در عین حال باید در نظر داشت که کامپایلر نمی‌تواند ارورها و خطاهای موجود در برنامه را تصحیح کند و از همین روی اگر خطایی در کدتان وجود داشته باشد، باید تغییراتی را در سینتکس برنامه ایجاد کنید که در غیر این صورت کدتان کامپایل نخواهد شد.

فرآیند کامپایل فرآیندی نسبتاً پیچیده است که طی آن زمان بسیار زیادی صرف تجزیه و تحلیل و ترجمه سورس‌کد به کدی قابل درک برای کامپیوتر می‌شود. به طور کلی، کامپایلرها سورس‌کد را می‌خوانند و یک کد قابل‌اجرا در خروجی تحویل می‌دهند. به عبارت دیگر، سورس‌کد نرم‌افزارهایی را که با یک زبان سطح بالا نوشته شده‌اند به صفر و یک‌هایی تبدیل می‌کند که کامپیوتر قادر به درک آن‌ها باشد. در واقع، کدی را که یک برنامه‌نویس می‌نویسد را به فرمتی خوانا برای CPU تبدیل می‌کند.

---

[۴] Compiler

[5] Interpreter

۲- ۵- مُفسِر

مُفسِر[۶] یک برنامه کامپیوتری است که هر خط از دستورات یک زبان سطح بالا را به کد ماشین تبدیل می‌کند. کامپایلر و مُفسِر هر دو کاری یکسان، یعنی تبدیل کدهای زبان سطح بالا به کد ماشین، انجام می‌دهند اما کامپایلر کد را **پیش** از اجرای برنامه به کد ماشین تبدیل می‌کند (یعنی یک فایل اجرایی همچون exe می‌سازد) در حالی که مفسر کد را **حین** اجرا به کد ماشین تبدیل می‌کند.

همان‌طور که پیش از این گفته شد، اینترپریتر سورس‌کد را خط به خط در حین اجرا ترجمه می‌کند به طوری که سورس یک برنامه نوشته شده با زبانی سطح بالا را به طور کامل به زبان ماشین ترجمه می‌کند و این در حالی است که مفسر اجازه می‌دهد تا ارزیابی و اصلاح برنامه در حین اجرا (Run-time) صورت پذیرد.

[۶] Interpreter

اینترپریتر	
عملکرد	برنامه را می‌سازد سپس همه دستورات زبان را از نظر درستی تجزیه و تحلیل می‌کند و اگر دستوری غلط باشد، ارور می‌دهد و اگر همه دستورات درست باشند، سورس‌کد را به کد ماشین تبدیل می‌کند. فایل‌های مختلفی را به برنامه اجرایی (همان فایل exe) اضافه می‌کند و در نهایت برنامه را اجرا می‌کند.
مزایا	استفاده از مفسرها به خصوص برای دولوپرهای مبتدی آسان‌تر است.
معایب	اپلیکیشن‌های نوشته شده با زبان‌های تفسیری تنها بر روی کامپیوترهایی اجرا می‌شوند که مفسر مربوطه روی آن‌ها نصب باشد.
سرعت اجرا	کدهای تفسیری کندتر اجرا می‌شوند.
فایل خروجی	هیچ‌گونه فایل‌اجرایی تولید نمی‌کند.
اجرا	اجرای برنامه بخشی از فرآیند تفسیر آن است؛ بنابراین برنامه خط به خط اجرا می‌شود.
مناسب برای	برای محیط وب که زمان لود اهمیت دارد مناسب است. به دلیل اینکه تجزیه و تحلیل باید کامل انجام شود، فرآیند کامپایل زمان نسبتاً زیادی را صرف می‌کند که در چنین مواردی، اینترپریترها انتخاب بهتری هستند.
بهینه‌سازی کد	مفسرها کد را خط به خط می‌بینند و لذا بهینه‌سازی‌هایی که انجام می‌گیرد به اندازه کامپایلرها قوی نیستند.
تایپ دینامیک	زبان‌های تفسیری به خوبی از تایپ دینامیک پشتیبانی می‌کنند.
کاربرد	بیشتر مناسب Development Environment است و سرعت توسعهٔ نرم‌افزار را بالا می‌برد چرا که در صورت وجود باگ در برنامه، دائم نیازی به کامپایل مجدد برنامه

	نیست.
هندل کردن خطاها	اینترپریتر یک خط از دستورات را می‌خواند و اگر خطایی وجود داشته باشد آن را نشان می‌دهد به طوری که برای تفسیر خط بعدی باید خطای قبلی برطرف شده باشد.
ورودی	یک خط از کد را به عنوان ورودی می‌گیرد.
خروجی	مفسرها هیچ کد ماشینی تولید نمی‌کنند.
خطاها	خطاهای یک خط از دستورات را یکی‌یکی نمایش می‌دهد.
زبان‌های برنامه‌نویسی تحت پوشش	Perl، PHPو Ruby از اینترپریتر استفاده می‌کنند.
کامپایلر	
عملکرد	برنامه را می‌سازد سپس همه دستورات زبان را از نظر درستی تجزیه و تحلیل می‌کند و اگر دستوری غلط باشد، ارور می‌دهد و اگر همه دستورات درست باشند، سورس‌کد را به کد ماشین تبدیل می‌کند. فایل‌های مختلفی را به برنامه اجرایی (همان فایل exe) اضافه می‌کند و در نهایت برنامه را اجرا می‌کند.
مزایا	کد برنامه کاملاً به کد ماشین ترجمه شده است؛ بنابراین زمان اجرای آن کمتر است.
معایب	برای تغییر برنامه حتماً باید به سورس‌کد آن مراجعه شده و در صورت وجود باگ در برنامه، بایستی مجدد کامپایل گردد.
سرعت اجرا	کدهای کامپایل‌شده سریع‌تر اجرا می‌شوند.
فایل خروجی	یک خروجی تولید می‌کند) با فرمت (exe که می‌تواند بدون نیاز به سورس‌کد اصلی اجرا شود.

اجرا	اجرای برنامه از فرآیند کامپایل آن جدا است به طوری که اجرای برنامه تنها زمانی اتفاق می‌افتد که کل کدها کامپایل شده باشند.
مناسب برای	محدود به یک دستگاه خاص است و نمی‌توان آن را پورت کرد.
بهینه‌سازی کد	کامپایلر همه کد را یک‌جا می‌بیند و از همین روی بهینه‌سازی‌های زیادی را برای اجرای سریع‌تر کد انجام می‌دهد.
تایپ دینامیک	پیاده‌سازی آن در کامپایلرها دشوار است چون کامپایلرها نمی‌توانند پیش‌بینی کنند که در حین اجرا چه اتفاقی می‌افتد.
کاربرد	بیشتر مناسب Production Environment است و از سرعت اجرای به مراتب بیشتری برخوردار است
هندل کردن خطاها	کامپایلر همهٔ خطاها و اخطارها را در زمان کامپایل نشان می‌دهد و از همین روی بدون تصحیح کردن خطاها، اجرای برنامه غیرممکن است.
ورودی	کل برنامه را به عنوان ورودی می‌گیرد.
خروجی	کامپایلرها کد ماشین تولید می‌کنند.
خطاها	همه خطاها را هم‌زمان پس از کامپایل نشان می‌دهد.
زبان‌های برنامه‌نویسی تحت پوشش	Scala، C#، C++، Cو Java همه از کامپایلر استفاده می‌کنند.

۲- ۶- کامپایلر و مفسر

کامپایلر یک برنامه کامپیوتری است که کد نوشته شده به یک زبان برنامه‌نویسی سطح بالا را به کد ماشین تبدیل می‌کند اما اینترپریتر یک برنامه کامپیوتری است که هر خط از دستورات یک زبان سطح بالا را در حین اجرا به کد ماشین تبدیل می‌کند.

کامپایلرها کدی واسط برای سیستم تولید می‌کنند که به عنوان یک فایل exe در کامپیوتر ذخیره می‌شود اما مفسرها هرگز چنین فایل اجرایی را تولید نمی‌کنند. همچنین بسته به حجم سورس‌کد، کامپایلر به زمان به نسبت زیادی برای تجزیه و تحلیل و ترجمهٔ سورس‌کد به زبان قابل‌فهم برای CPU نیاز دارد اما اینترپریتر سورس‌کد را خط به خط در حین اجرا ترجمه می‌کند و بالتبع زمان کمتری را صرف تجزیه و تحلیل برنامه می‌کند.

فصل سوم- معرفی برخی از انواع زبان های برنامه نویسی

۳- ۱- زبان برنامه نویسی جاوا

۳- ۲- زبان برنامه نویسی C

۳- ۳- زبان برنامه نویسی پایتون

۳- ۴- زبان برنامه نویسی C++

۳- ۵- زبان برنامه نویسی Visual Basic .NET

۳- ۶- زبان برنامه نویسی C#

۳- ۷- زبان برنامه نویسی PHP

۳- ۸- زبان برنامه نویسی جاوا اسکریپت

۳- ۹- زبان برنامه نویسی SQL

۳- ۱۰- زبان برنامه نویسی متلب

۳- ۱۱- زبان برنامه نویسی R

در این فصل به معرفی زبان های برنامه نویسی پرکاربرد در دنیای برنامه نویسی می‌پردازیم.

۳- ۱- زبان برنامه نویسی جاوا

جاوا (Java) یک فریم ورک و زبان برنامه نویسی بسیار کاربردی پیشرو و همه منظوره است. این زبان برنامه نویسی در سال ۱۳۷۰ هجری شمسی (۱۹۹۱ میلادی) توسط «Sun Microsystems» به عنوان یک زبان برنامه نویسی سطح بالا، کامپایل شده و مدیریت همراه با حافظه، ارائه شده است. ساختار نوشتار (Syntax) برنامه نویسی جاوا شبیه به برنامه نویسی C و C++ است، یکی از تفاوت‌هایاین زبان با آن‌ها داشتن آکولاد برای بسته شدن و نقطه ویرگول (Semicolon) برای دستورات پایانی به حساب می‌آید.

مدیریت خودکار حافظه یکی از ویژگی‌هایی است که برنامه نویسی جاوا را به سرعت پس از انتشار اولیه آن، بسیار محبوب کرد. قبل از معرفی شدن زبان برنامه نویسی جاوا، بیش‌تر انواع زبان های برنامه نویسی که به مدیریت حافظه دستی نیاز داشتند، مانند زبان برنامه نویسی C و C++ مورد استفاده قرار می‌گرفتند. تخصیص دستی حافظه خسته کننده و مستعد خطا است، بنابراین زبان جاوا به عنوان یک گام بزرگ رو به جلو برای توسعه دهندگان برنامه‌ها مورد استقبال قرار گرفت.

برنامه نویسی جاوا، به برنامه نویسان وعده‌های بسیار و فراتر از مدیریت حافظه داده بود که از جمله آن‌ها می‌توان به قابلیت چند سکویی (Cross-Platform) اشاره کرد. ماشین مجازی جاوا (Java Virtual Machine | JVM)توسط کد بایتی برنامه نویسی جاوا اجرا می‌شود که توسط زبان جاوا کامپایل شده است. استفاده از ماشین مجازی جاوا در اکثر سیستم عامل‌ها مانند لینوکس

(Linux)، مک و ویندوز امکان‌پذیر است. این ماشین مجازی همیشه بی‌نقص نیست، اما زمانی که کاری را انجام می‌دهد، یک برنامه نوشته شده با برنامه نویسی جاوا می‌تواند روی هر پلتفرمی اجرا شود که با ماشین مجازی جاوا سازگار است.

برنامه نویسی جاوا در کسب و کارها، توسعه نرم افزار، وب و اپلیکیشن‌های موبایل مورد استفاده قرار می‌گیرد. این زبان یکی از انواع زبان های برنامه نویسی اصلی و پایه برای سیستم عامل اندروید گوگل در نظر گرفته می‌شود. همچنین برنامه نویسی جاوا میلیون‌ها ستاپ باکس (Set-Top Boxes)و دستگاه‌های تعبیه شده (Embedded Device) را تامین می‌کند. به صورت کلی، می‌توان گفت که مهارت‌های توسعه جاوا بسیار مورد توجه واقع شده است. به لحاظ محبوبیت جاوا در رده محبوب‌ترین زبان‌های برنامه نویسی قرار می‌گیرد و میزان دشواری یادگیری آن بالاتر از سطح متوسط است.

۳- ۲- زبان برنامه نویسیC

قبل از اینکه زبان جاوا معرفی شود، زبان برنامه نویسی «C» سطح بالاترین زبان برنامه نویسی به حساب می‌آمد. این زبان برنامه نویسی برای اولین بار در سال ۱۳۵۱ شمسی (۱۹۷۲ میلادی) معرفی شد. در آن زمان، اولین نسخه‌های سیستم عامل یونیکس (Unix) که به زبان برنامه نویسی اسمبلی (Assembly) نوشته شده بودند به زبان C منتقل شدند و سپس در توسعه سایر سیستم عامل‌های اولیه، از جمله «IBM System/370» از زبان برنامه نویسی C استفاده شد.

زبان برنامه نویسی C سابقه طولانی در توسعه سیستم‌های قدیمی با پردازنده‌های کندتر و حافظه کم دارد. برنامه‌هایی که به زبان برنامه نویسی C نوشته می‌شوند باید بسیار کارآمدتر باشند. بنابراین زبان C در مواردی که سرعت مهم است به عملکرد بالا شهرت دارد. برنامه نویسی C به دلیل استفاده از آن در توسعه سیستم‌ها، از جمله سیستم عامل‌ها، دستگاه‌های تعبیه شده، «Firmware»، درایورهای سخت افزاری و کاربردهای عمومی بسیار محبوب است. کتابخانه استاندارد زبان C به بسیاری از پلتفرم‌ها منتقل شده است، بنابراین در حوزه‌های گوناگونی کاربرد دارد.

با این حال، برنامه نویسی سیستم‌های سطح پایین که معمولاً از زبان برنامه نویسی C استفاده می‌کنند، مهارتی تخصصی‌تر از برنامه نویسی در کاربردهای عمومی دارند و همین موضوع نشان می‌دهد که چرا دومین زبان محبوب دنیا نسبت به سایر زبان های موجود، موقعیت‌های شغلی کمتری دارد. بازار کار این زبان برنامه نویسی با بازار کار زبان برنامه نویسی C++ همپوشانی دارد و تقریباً یکسان هستند. به طور کلی می‌توان گفت زبان برنامه نویسی «C» محبوبیت متوسطی دارد و سطح یادگیری آن نیز متوسط به حساب می‌آید.

۳- ۳- زبان برنامه نویسی پایتون

زبان برنامه نویسی پایتون به طور نسبی یکی از انواع زبان های برنامه نویسی جدید به حساب می‌آید که برای اولین بار در سال ۱۳۶۸ هجری شمسی (۱۹۸۹ میلادی) معرفی شده است. زبان برنامه نویسی پایتون، یک زبان تفسیری است که از مدیریت خودکار حافظه و برنامه نویسی شی

گرا پشتیبانی می‌کند. پایتون برای برنامه نویسی همه منظوره از جمله برنامه‌های کاربردی تحت وب بسیار محبوب است و همچنین در سال‌های اخیر در حوزه برنامه نویسی هوش مصنوعی و شبکه‌های عصبی مورد استفاده فراوان قرار گرفته است. مشاغلی که از برنامه نویسی پایتون استفاده می‌کنند بسیار زیاد هستند و پیدا کردن شغل در این زمینه کار ساده‌ای است. سطح یادگیری این زبان برنامه نویسی آسان و متوسط به حساب می‌آید.

۳- ۴- زبان برنامه نویسیC++

زبان برنامه نویسیC++ ، همان زبان برنامه نویسی C است که با ویژگی‌های شی گرایی آن را گسترش داده‌اند. وجود دو علامت به اضافه در ادامه حرف C در نام این زبان به افزوده شدن قابلیت عملگر افزایشی در آن اشاره دارد. زبان برنامه نویسی C++ برای انتقال ویژگی‌های زبان های برنامه نویسی قدیمی به پلتفرم‌های سریع‌تر و قدرتمندتر جدید به وجود آمده است. بازار کار زبان برنامه نویسی C++ تقریباً مشابه زبان برنامه نویسی C به حساب می‌آید، از جمله این بازار کار می‌توان به برنامه نویسی سیستم‌ها، توسعه سخت افزار سطح پایین، خدمات تحت وب و کاربردهای عمومی و اختصاصی اشاره کرد.

در طول سال‌های اخیر، کتابخانه‌های استاندارد زبان برنامه نویسی C++ و خصوصیات آن‌ها به‌طور قابل‌توجهی گسترش یافته‌اند، که منجر به انتقاداتی از سوی برنامه نویسان شده است که عقیده دارند باعث پیچیدگی بیش از حد و دشواری در یادگیری این زبان برنامه نویسی می‌شود. در نهایت،

به طور کلی زبان برنامه نویسی C++ یکی از زبان های دشوار از نظر سطح یادگیری است اما محبوبیت بالایی دارد.

۳- ۵- زبان برنامه نویسی Visual Basic .NET

زبان برنامه نویسی ویژوال بیسیک دات نت (Visual Basic.NET | VB.NET) پیاده‌سازی مایکروسافت از زبان ویژوال بیسیک است که به وسیله زبان واسط دات نت (.NET) کامپایل می‌شود. زبان برنامه نویسی «VB.NET» به توسعه دهندگان این امکان را می‌دهد تا برنامه‌های دات نت را با استفاده از ویژوال بیسیک بنویسند. کم و بیش در سطح برنامه‌های توسعه داده شده با سایر زبان‌های برنامه نویسی هستند.

با این حال، این زبان برنامه نویسی هیچ وقت برای استفاده در برنامه‌های تجاری و کسب و کارها محبوب نشده است و توسعه دهندگان و برنامه نویسان، زبان های برنامه نویسی C++ ، Cو C# را بیش‌تر ترجیح می‌دهند. اکثر برنامه‌های کاربردی نوشته شده با زبان برنامه نویسی VB.NET قدیمی هستند و به عنوان برنامه‌های منسوخ شده در نظر گرفته می‌شوند و بیش‌تر برای از کار انداختن یا توسعه مجدد استفاده می‌شوند.

در کل می‌توان گفت که این زبان برنامه نویسی محبوبیت کمی دارد و سطح سختی یادگیری آن متوسط است. زبان VB.NET در توسعه برنامه‌های عمومی و تحت وب کاربرد دارد.

۳- ۶- زبان برنامه نویسیC#

برنامه نویسی C# (سی شارپ) در سال ۱۳۷۹ هجری شمسی (۲۰۰۰ میلادی) توسط مایکروسافت به همراه فریم ورک دات نت (.NET Framework) توسعه داده و معرفی شد. از لحاظ ساختار نوشتاری، زبان برنامه نویسی C# بسیار شبیه به زبان های برنامه نویسی جاوا، C++و C است. زبان C#، یک زبان کامپایل شده و شی گرا به حساب می‌آید که با زبان واسط دات نت کامپایل می‌شود. در ابتدا، زبان برنامه نویسی C# فقط برای توسعه فرم‌های (Form) ویندوز متمرکز بر مایکروسافت و توسعه وب با «ASP.NET» استفاده می‌شد. فریم ورک .NET اخیراً با معرفی دات نت استاندارد و «.NET Core»تکامل یافته است. این چارچوب‌ها و استانداردهای جدید پلتفرم چند سکویی هستند و روی ویندوز، لینوکس و مک اجرا می‌شوند.

برنامه نویسی C# برای برنامه نویسی اپلیکیشن‌های کاربردی عمومی و تحت وب، در سیستم‌هایی که بر اساس فناوری مایکروسافت توسعه یافته‌اند و یک زبان محبوب به حساب می‌آید. فریم ورک زامارین (Xamarin) مایکروسافت به توسعه دهندگان و برنامه نویسان اجازه می‌دهد تا برنامه‌های سیستم عامل‌های اندروید و «iOS» را در C# بنویسند. این فریم ورک در برخی موارد برای برنامه نویسی سیستم‌ها نیز مناسب و دارای کتابخانه‌هایی برای سیستم‌های تعبیه شده است. زبان C# جزء زبان‌های برنامه نویسی محبوب در دنیای امروز به حساب می‌آید و سختی یادگیری این زبان در حد متوسط است. در بخش بعدی به بررسی زبان برنامه نویسی پی اچ پی (PHP) پرداخته شده است.

۳- ۷- زبان برنامه نویسیPHP

زبان برنامه نویسی «PHP» در ابتدا مخفف عبارت صفحه اصلی شخصی[۷] بود و نام اصلی آن به این صورت بود که در ادامه عبارت «PHP» مخفف «Forms Interpreter» (مفسر فُرم‌ها) نیز به صورت «PHP/FI»استفاده می‌شد. مخفف رسمی که امروزه برای برنامه نویسی «PHP» در نظر گرفته می‌شود «پردازشگر فرامتن» یا به زبان انگلیسی «Hypertext Processor» است. برنامه نویسی و توسعه اپلیکیشن‌های تحت وب در سمت سرور کاربرد اصلی زبان برنامه نویسی PHP به حساب می‌آید.

زبان برنامه نویسی پی اچ پی در ابتدا جهت گسترش یک برنامه واسط دروازه مشترک (Common Gateway Interface | CGI)برای پشتیبانی از فرم‌های «HTML»و دسترسی به پایگاه داده (Database)توسعه داده شد. با ترکیب کدهای یک برنامه به زبان «PHP» با کدهای«HTML» ، کدهایی شبیه به صفحات فعال سرور کلاسیک مایکروسافت یعنی «قبل از دات نت» تولید خواهد شد. در این ترکیب، مفسر مورد نظر برنامه «HTML» و کدها را می‌خواند و بخش‌های کد صفحات سرور را اجرا می‌کند.

یکی از دلایلی که زبان برنامه نویسی «PHP» بسیار محبوب و پرطرفدار به حساب می‌آید، آسان بودن یادگیری آن است. همچنین این زبان برنامه نویسی، پایه برنامه‌های محبوب مبتنی بر وب مانند وردپرس (WordPress) و جوملا (Joomla) به شمار می‌رود. نسخه‌های اولیه این زبان برنامه

[۷] Personal Home Page

نویسی فاقد کنترل‌های امنیتی و یک سری ویژگی‌های کاربردی بودند و توسعه برنامه‌های بسیار امن را دشوار می‌کردند. پیشرفت‌های اخیر در فریم ورک زبان «PHP» و کتابخانه‌های آن، باعث بهبود امنیت برنامه‌های این زبان برنامه نویسی شده است.

۳- ۸- زبان برنامه نویسی جاوا اسکریپت

زبان برنامه نویسی جاوا اسکریپت(JavaScript) ، یک زبان برنامه نویسی سطح بالا، پویا و تفسیری است. زبان تفسیری زبانی است که عبارت‌های آن یکی پس از دیگری ترجمه و اجرا می‌شوند و در مقابل آن زبان های برنامه نویسی کامپایلی وجود دارند که قبل از اجرا تمام عبارت‌های آن به یک باره ترجمه می‌شوند. ایده نام‌گذاری این زبان به این دلیل بوده که سازندگان آن قصد داشتند جاوا اسکریپت را به عنوان یک زبان اسکریپتی مکمل برای همراهی با جاوا ارائه دهند که یک زبان کامپایلی است. همچنین برخی بر این عقیده هستند که انتخاب چنین نامی برای جاوا اسکریپت با اهداف بازاریابی و رسیدن به محبوبیت بیش‌تر انجام شده است. زبان برنامه نویسی «JavaScript» در سال ۱۳۷۴ هجری شمسی (۱۹۹۵ میلادی) همزمان با روزهای اولیه ظهور اینترنت عمومی، معرفی شد.

زبان برنامه نویسی جاوا اسکریپت در ایجاد کدهایی کاربرد دارد که در مرورگرهای وب در سمت کلاینت اجرا می‌شوند. همچنین، این زبان در بخش‌هایی از بک اند نیز به کار گرفته می‌شود. در ابتدای معرفی«Google Maps» ، پیمایش بی‌نهایت (Infinite Scrolling) در نقشه‌ها با استفاده از زبان برنامه نویسی جاوا اسکریپت انجام می‌شد. از زمانی که این زبان برنامه نویسی معرفی شد،

پشتیبانی زبان برنامه نویسی جاوا اسکریپت به تمام مرورگرهای وب اصلی اضافه شده است. فریم ورک‌های جاوا اسکریپت از جمله React، Angularو Vue.jsیک الگوی توسعه کاربردی »مدل‌نما کنترل‌گر (Model-View-Controller | MVC) «را ارائه می‌دهند که به طور کامل در مرورگر اجرا می‌شوند.

زبان برنامه نویسی جاوا اسکریپت از اپلیکیشن‌های تحت وب پشتیبانی می‌کند، به همین دلیل است که اکثر ابزارهای نظارت کاربر واقعی(Real User Monitoring Tools) ، زبان برنامه نویسی جاوا اسکریپت را تامین می‌کنند. همچنین زبان جاوا اسکریپت یکی از انواع زبان های برنامه نویسی است که این امکان را دارد با زبان برنامه نویسی «HTML» ترکیب شود تا اپلیکیشن‌های موبایلی پلتفرم چند سکویی ایجاد کند «Node.js» .یک سرور تحت وب است که زبان برنامه نویسی جاوا اسکریپت را در سمت سرور اجرا می‌کند. برنامه‌های سرور تحت وب «Node.js» به طور کامل با زبان برنامه نویسی «JavaScript» نوشته می‌شوند.

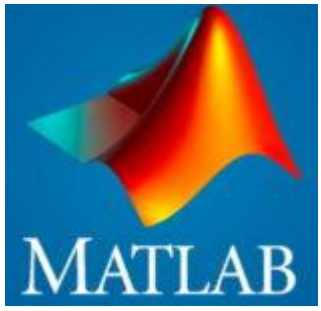
۳- ۹- زبان برنامه نویسیSQL

زبان برنامه نویسی اس کیو ال یا سیکوئل(Structured Query Language | SQL) ، برای پرس و جو (Query) و تغییر داده‌ها در یک سیستم مدیریت بانک اطلاعاتی رابطه‌ای (Relational Database Management System | RDBMS)استفاده می‌شود. نرم افزارهای بانک اطلاعاتی زیادی وجود دارند مانند PostgreSQL، PL/SQL (Oracle)و T-SQL که هر کدام ویژگی‌های خاصی ارائه می‌دهند.

زبان «SQL» یک زبان برنامه نویسی عمومی مانند انواع زبان های برنامه نویسی دیگر نیست که بتوان از آن برای نوشتن برنامه‌ها استفاده کرد. با این حال، یک مهارت مفید و مورد نیاز برای اکثر توسعه دهندگان به حساب می‌آید. اصطلاح «برنامه نویسی فول استک (Full-Stack Developer) » به توسعه‌دهنده‌ای با مجموعه‌ای از مهارت‌های کامل گفته می‌شود که همه جنبه‌های یک برنامه کاربردی را بررسی می‌کند. این جنبه‌های برنامه‌های کاربردی شامل دسترسی و ذخیره داده‌ها در پایگاه داده می‌شود. یادگیری زبان برنامه نویسی «SQL» در ابتدا کار سختی نیست، اگرچه استفاده پیشرفته از سیکوئل در داده‌های کلان (Big Data) و تجزیه و تحلیل داده‌ها (Data Analysis) به تجربه قابل توجهی نیاز دارد.

زبان برنامه نویسی SQL یکی از انواع زبان های برنامه نویسی به شمار می‌رود که در بین توسعه دهندگان و مدیران پایگاه داده بسیار محبوب است، بنابراین مشاغلی که به مهارت‌های SQL نیاز دارند، فراوان هستند. با این حال، این زبان دستوری یک مهارت کامل نیست. داشتن تجربه در زمینه SQL امتیاز بزرگی در رزومه محسوب می‌شود، اما به ندرت به عنوان تنها مهارت اولیه مورد نیاز برای یک موقعیت شغلی در نظر گرفته می‌شود.

۳- ۱۰- زبان برنامه نویسی متلب



متلب (MATLAB) به تنهایی یک زبان برنامه نویسی به شمار نمی‌رود. می‌توان گفت که متلب یک برنامه کاربردی است که برای محاسبه و مدل‌سازی محاسبات پیچیده ریاضی استفاده می‌شود. زبان برنامه نویسی متلب در درجه اول در محیط‌های تحقیقاتی، دانشگاه‌ها و آزمایشگاه‌ها استفاده

می‌شود. نرم افزار کاربردی متلب می‌تواند محاسبات پیچیده ماتریسی را انجام دهد و با استفاده از برنامه‌هایی که دارد از نمادهای پیچیده ریاضی پشتیبانی کند. توابع نوشته شده با زبان های برنامه نویسی C# ، C و FORTRAN را می‌توان در نرم افزار «MATLAB» فراخوانی کرد.

دانش مورد نیاز برای استفاده از زبان برنامه نویسی متلب بیشتر به مفاهیم و مهارت‌های ریاضی مرتبط است. اگر شخصی یک دانشجوی پیشرفته ریاضی باشد که در مقطع دکتری ریاضیات کار می‌کند، یادگیری متلب برای آن نسبتاً آسان است.

۳- ۱۱- زبان برنامه نویسیR

زبان برنامه نویسی «R» در درجه اول توسط کارشناسان آمار و پژوهشگران برای انجام تجزیه و تحلیل آماری مجموعه داده‌ها (Dataset) استفاده می‌شود. جمعیت شناسان، حسابداران بیمه و سایر مشاغل متمرکز بر تجزیه و تحلیل‌های آماری از زبان برنامه نویسی R استفاده می‌کنند. بیشتر دانش مورد نیاز برای کار با زبان برنامه نویسیR ، مانند زبان برنامه نویسی متلب، وابسته به آمار است. برنامه نویسی زبان R یک زبان سخت برای یادگیری به حساب می‌آید و برنامه نویسان آن دانش آمار را یکی از ملزومات زبان برنامه نویسی R در نظر می‌گیرند و برای توسعه نرم افزارها از آن بهره می‌برند.

تعداد مشاغل زبان برنامه نویسی R به دلیل تخصصی بودن این زبان برنامه نویسی بالا نیست. اگر شخصی تحلیل‌گر داده باشد و با استفاده از روش‌های آماری کارش را انجام بدهد به احتمال زیاد با

زبان برنامه نویسی R کار کرده است. برای بالا بردن تعداد گزینه‌های این زبان برنامه نویسی، می‌توان جعبه ابزار R را به آن اضافه کرد.

ایده اساسی این است که هر چه سطح یک زبان برنامه نویسی پایین‌تر باشد، کدهای آن زبان شباهت بیش‌تری به زبان ماشین دارند.

جدا از اهدافی که هر زبان برنامه نویسی برآورده می‌کند، معمولاً تمایلات و سلیقه شخصی هم در انتخاب یک زبان برنامه نویسی دخیل هستند. در واقع، برای برآورده کردن یک هدف خاص و پیاده‌سازی یک قابلیت عملکردی مشخص، قدرت انتخاب وجود دارد و می‌توان از بین چند زبان برنامه نویسی یکی را برگزید. می‌توانید چند زبان برنامه نویسی را امتحان کنید و به این وسیله مشخص می‌شود که کدام یک از آن‌ها برای شما مناسب‌تر هستند.

و البته یک نکته کلیدی را هرگز از یاد نبرید که چیزی به عنوان بهترین زبان برنامه نویسی **وجود ندارد.**

فصل چهارم- اجزای زبان های برنامه نویسی

۴- ۱- نحو یا سینتکس در زبان برنامه نویسی

۴- ۲- تفاوت سینتکس و معناشناسی در زبان برنامه نویسی

۴- ۲- ۱- معناشناسی ایستا

۴- ۲- ۲- معناشناسی پویا

۴- ۳- سیستم تعیین نوع در زبان برنامه نویسی

۴- ۴- مقایسه زبان های برنامه نویسی دارای نوع و زبان های بدون نوع

۴- ۵- نوع دهی ضعیف و نوع دهی قوی در زبان برنامه نویسی

۴- ۶- کتابخانه استاندارد و سیستم زمان اجرا در زبان برنامه نویسی

۴- ۷- طراحی و پیاده سازی زبان های برنامه نویسی چگونه است

۴- ۸- علت نیاز به چندین زبان برنامه نویسی

۴- ۹- اهمیت انتزاع در زبان برنامه نویسی

۴- ۱۰- زبان برنامه نویسی طبیعی

۴- ۱۱- مشخصه ها یا اصول زبان های برنامه نویسی

۴- ۱۲- پیاده سازی زبان های برنامه نویسی به چه معناست

۴- ۱۳- زبان برنامه نویسی اختصاصی

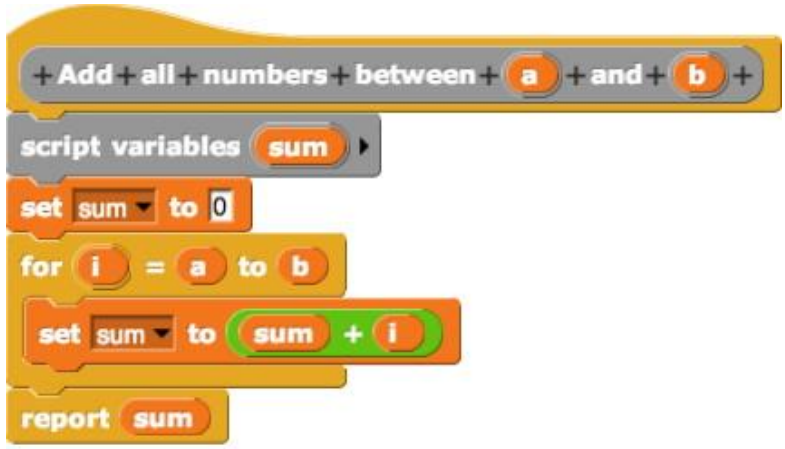
۴- ۱۴- کاربرد زبان برنامه نویسی

۴- ۱۵- اندازه گیری میزان استفاده از زبان های برنامه نویسی مختلف

تمام زبان‌های برنامه نویسی تعدادی عنصر اصلی اولیه برای توصیف داده‌ها، پردازش‌ها و تغییرهای اعمال شده روی آن‌ها دارند. از جمله این پردازش‌ها می‌توان به عملیات جمع دو عدد یا انتخاب آیتمی از یک مجموعه اشاره کرد. این عنصرهای اولیه به وسیله قوانین نحوی (Syntactic) و معنایی (Semantic)تعریف می‌شوند. این قوانین نحوی و معنایی، ساختار و مفهوم زبان‌های برنامه نویسی را توصیف می‌کنند. در ادامه این بخش، برای روشن‌تر شدن پاسخ این سوال که زبان برنامه نویسی چیست ، به اجزای اصلی زبان‌های برنامه نویسی پرداخته شده است.

۴- ۱- نحو یا سینتکس در زبان برنامه نویسی

شکل و فُرم ظاهری و سطحی یک زبان برنامه نویسی را «سینتکس (Syntax) «یا «نحو» آن زبان می‌نامند. اکثر زبان‌های برنامه نویسی به طور کامل «متنی (Textual) «هستند. زبان‌های برنامه نویسی درست مثل زبان‌های طبیعی از توالی‌های متنی شامل کلمه‌ها، عددها و علامت‌های دستوری تشکیل شده‌اند. از طرف دیگر، زبان‌های برنامه نویسی دیگری هم وجود دارند که طبیعت آن‌ها بصری‌تر و گرافیکی‌تر است. در چنین زبان‌هایی از رابطه‌های مجازی میان نمادها برای مشخص کردن اجزای یک برنامه استفاده می‌شود. یکی از محبوب‌ترین زبان‌های برنامه نویسی دیداری، اسکرچ (Scratch) نام دارد.



اسکرچ — (Scratch) یکی از محبوب‌ترین زبان‌های دیداری

سینتکس یک زبان برنامه نویسی، مشخص کننده ترکیب‌های ممکن نمادهایی است که به تولید یک برنامه با قواعد نحوی صحیح خواهد انجامید.

معنا و مفهوم تعیین شده برای ترکیب نمادها به وسیله «معناشناسی (Semantics)» مشخص می‌شود (که یا به صورت فُرمی است یا به صورت سخت در یک پیاده‌سازی مرجع کدگذاری شده است.) با توجه به اینکه اکثر زبان‌ها متنی هستند، در این مقاله نیز بیش‌تر به زبان‌های برنامه نویسی دارای قالب‌های متنی پرداخته شده است.

سینتکس زبان‌های برنامه نویسی معمولاً به وسیله ترکیبی از عبارت‌های منظم برای ساختار واژگانی و فرم باکوس نائور (Backus–Naur form) برای ساختارهای گرامری (ساختار دستور زبان) تعریف شده‌اند. فرم باکوس نائور یکی از روش‌های بیان قاعده‌ها به حساب می‌آید که برای گرامر مستقل از متن ارائه شده است و اغلب از آن به عنوان دستور زبان رسمی در علوم کامپیوتر استفاده می‌شود.

مثالی برای ساختار دستور زبان

در ادامه به عنوان مثال یک ساختار گرامری ساده براساس زبان Lisp ارائه شده است:

```
expression ::= atom | list
atom       ::= number | symbol
number     ::= [+-]?['0'-'9']+
symbol     ::= ['A'-'Z''a'-'z'].*
list       ::= '(' expression* ')'
```

موارد زیر در ساختار گرامری فوق تعیین شده‌اند:

- **عبارت (Expression)** یا یک اتم (atom) است یا یک لیست (List) به حساب می‌آید.
- یک **اتم** یا از نوع عددی (number) یا به صورت یک نماد (symbol) است.

- **عدد**، دنباله شناخته نشده‌ای از یک یا بیش از یک رقم دهدهی به حساب می‌آید که بر اساس نیاز، قبل از علامت به‌اضافه یا منها و بعد از آن استفاده می‌شود.
- **نماد (Symbol)** حرفی است که بعد از آن (با در نظر نگرفتن فضای سفید) هر کاراکتر دیگری نوشته می‌شود یا ممکن است هیچ کاراکتری هم بعد از آن قرار نگیرد.
- **لیست** به جفت پرانتزهایی گفته می‌شود که چند عبارت می‌توانند در داخل آن قرار بگیرند. همچنین ممکن است یک لیست تهی هم باشد.

حال در ادامه مثال‌هایی از توالی‌های نمادین در قالب این دستور زبان ارائه شده‌اند:

- 12345
- ()
- (a b c232 (1))

۴- ۲- تفاوت سینتکس و معناشناسی در زبان برنامه نویسی

باید در نظر داشت، تمام برنامه‌هایی که به لحاظ نحوی (سینتکس) صحیح هستند، لزوماً از نظر معنایی هم صحیح نخواهند بود. در واقع، بسیاری از برنامه‌های با سینتکس درست، براساس قوانین زبانی شکل و فُرم مناسبی ندارند. چنین برنامه‌هایی (بسته به مشخصات زبان و درستی پیاده‌سازی آن‌ها) به خطایی در ترجمه یا اجرا منجر خواهند شد. در برخی از موارد، چنین برنامه‌هایی ممکن است رفتار تعریف نشده‌ای را از خود نشان دهند. حتی زمانی که یک برنامه به خوبی با یک زبان خاص نوشته شده باشد، باز هم احتمال دارد معنا و مفهومی را از خود بروز دهد که مقصود برنامه نویس نبوده است.

مثالی برای درک بهتر تفاوت سینتکس و معناشناسی

برای مثال در استفاده از زبان‌های طبیعی ممکن است نتوان معنا و مفهومی را به یک جمله صحیح از لحاظ گرامری انتقال داد. همچنین در زبان‌های طبیعی ممکن است جمله از لحاظ قواعد نحوی ساختار درستی داشته باشد، اما معنا و مفهوم جمله غلط و اشتباه باشد:

- جمله «ایده‌های سبز بی‌رنگ خشمگین می‌خوابند» از لحاظ گرامری (نحوی) صحیح است، اما هیچ مفهوم کلی قابل قبولی ندارد.
- جمله «رضا یک مجرد متاهل است» از لحاظ ساختار و قواعد دستوری درست است، اما معنا و مفهومی را منتقل می‌کند که به هیچ عنوان نمی‌تواند صحیح باشد.

حال پس از بررسی تفاوت سینتکس و معناشناسی در زبان‌های طبیعی، بهتر است در ادامه مثالی از این تمایز در زبان‌های برنامه نویسی هم ارائه شود. قطعه کد زیر نوشته شده به زبان C به لحاظ سینتکس صحیح است، اما عملیاتی را اجرا می‌کند که به لحاظ معنایی تعریف نشده و نامفهوم هستند. در واقع، عملیات p* << 4هیچ مفهومی برای مقداری با نوع مختلط (Complex) ندارد و p->im هم تعریف نشده است، زیرا مقدار pاشاره‌گر Null است:

```
complex *p = NULL;
complex abs_p = sqrt(*p >> 4 + p->im);
```

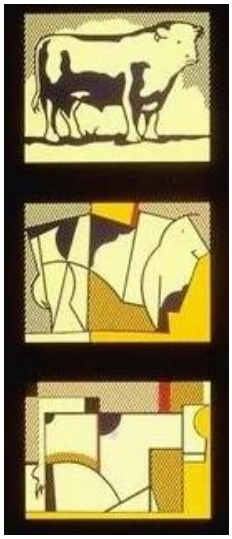
اگر تعریف نوع در خط اول حذف شده بود، برنامه در طول عملیات کامپایل (ترجمه به زبان ماشین) خطایی را برای متغیر شناسایی نشده pصادر می‌کرد. اگرچه، برنامه همچنان به لحاظ دستوری عملکرد صحیحی دارد، زیرا تعریف نوع تنها اطلاعات معنایی را فراهم می‌سازد.

قواعد نحوی مورد نیاز در یک زبان برنامه نویسی را می‌توان به وسیله موقعیت آن در سلسله مراتب چامسکی (Chomsky Hierarchy) طبقه‌بندی کرد. سینتکس اکثر زبان‌های برنامه نویسی را می‌توان با استفاده از گرامر نوع دو (Type-2) تعیین کرد. یعنی این نوع دستور زبان‌ها مستقل از متن (Context Free)هستند. برخی از زبان‌ها از جمله Perl و Lisp شامل ساختارهایی هستند که امکان اجرا را در طول فاز تجزیه (Parsing Phase) فراهم می‌کنند.

زبان‌هایی هم هستند که به برنامه نویس اجازه می‌دهند تا رفتار «تجزیه کننده (Parser) را تغییر دهند. در این زبان‌ها، تحلیل سینتکس به یک مسئله اثبات نشدنی تبدیل شده است و به‌طور کلی مرز میان تجزیه و اجرا بسیار گنگ و غیر قابل تشخیص است. برخلاف سیستم مبتنی بر ماکرو در زبان Lisp و بلوک‌های BEGINدر زبان Perl که ممکن است محاسبات عمومی و کلی را شامل شوند، ماکروهای C صرفاً جایگزین‌هایی برای رشته‌ها هستند و نیازی به اجرای کدها در آن وجود ندارد.

۴- ۳- معنا شناسی[۸] در زبان برنامه نویسی

مفهوم معناشناسی برخلاف قالب و ظاهر زبان برنامه نویسی یا همان سینتکس آن، به معنای زبان‌ها مربوط می‌شود. اصطلاح «معناشناسی (Semantics) «محدودیت‌های ساختار متن‌های معتبری را تعیین می‌کند که ابراز و بیان آن‌ها در صورت‌بندی‌های نحوی بسیار دشوار یا ناممکن است. معناشناسی در زبان برنامه نویسی دارای دو نوع است که در ادامه به شرح هر یک پرداخته می‌شود.



[۸] Semantics

معنی و مفهوم در زبان‌های برنامه نویسی

۴- ۲- ۱- معناشناسی ایستا

برای زبان‌های برنامه نویسی کامپایلری، معناشناسی ایستا (Static Semantics) اساساً شامل آن دسته از قوانین معنایی می‌شود که امکان بررسی آن‌ها در طول زمان کامپایل وجود دارد. مثال‌های مربوط به این مورد به شرح زیرند:

- بررسی این مسئله که هر شناساگر پیش از استفاده اعلان می‌شود (در زبان‌هایی که نیاز به اعلان‌ها دارند.
- برچسب‌های یک گزاره‌ی Case قابل تشخیص و متمایز هستند.

بسیاری از محدودیت‌های مهم از این دست، مثل بررسی اینکه شناساگرها در بستر مناسبی مورد استفاده قرار می‌گیرند یا خیر (مثلاً اضافه نکردن یک عدد صحیح به نام تابع) یا اینکه فراخوانی‌های زیرروال‌ها دارای تعداد و نوع آرگومان مناسب باشند، می‌توانند به وسیله تعریف منطقی با نام «سیستم نوع (Type System) «اعمال شوند.

سایر انواع تجزیه-تحلیل ایستا نظیر تجزیه-تحلیل جریان داده‌ها نیز ممکن است بخشی از معناشناسی ایستا باشند. زبان‌های برنامه نویسی جدیدتری مثل جاوا و C# (سی‌شارپ) دارای تجزیه-تحلیل تخصیص قطعی (Definite Assignment Analysis) یعنی نوعی از تجزیه-تحلیل جریان داده‌ها به عنوان بخشی از معناشناسی ایستای خود هستند.

۴- ۲- ۲- معناشناسی پویا

پس از آنکه داده‌ها مشخص شدند، لازم است به ماشین فرمان داده شود تا عملیات لازم را روی داده‌ها اجرا کند. برای مثال، معناشناسی ممکن است خط مشئی را تعریف کند که به وسیله آن مقادیر عبارت‌ها مورد ارزیابی قرار بگیرند یا رفتاری مشخص شود که در آن ساختارهای کنترلی بر اساس شرط‌ها گزاره‌هایی را اجرا کنند. در «معناشناسی پویا (Dynamic Semantics) «در یک زبان

برنامه نویسی که از آن با عنوان «معناشناسی اجرایی (Execution Semantics) «هم یاد می‌شود، این مسئله تعریف می‌شود که چگونه و در چه زمانی ساختارهای مختلف یک زبان برنامه نویسی باید رفتار برنامه را تولید کنند.

روش‌های بسیاری برای تعریف معناشناسی اجرایی وجود دارند. اغلب برای تعیین معناشناسی اجرایی زبان‌های برنامه نویسی که در عمل به طور رایج مورد استفاده قرار می‌گیرند از زبان‌های طبیعی استفاده می‌شود. تحقیقات علمی قابل‌توجهی در خصوص معناشناسی فُرمی زبان‌های برنامه نویسی انجام شده است؛ این تحقیقات امکان تعیین معناشناسی اجرایی را در حالت فُرم‌گونه فراهم کرده‌اند. نتایج به دست آمده از این زمینه تحقیقاتی، کاربردهای محدودی را در طراحی زبان برنامه نویسی و پیاده‌سازی خارج از محیط دانشگاهی به خود دیده است.

۴- ۳- سیستم تعیین نوع در زبان برنامه نویسی

سیستم تعیین نوع (Type System) به چگونگی طبقه‌بندی مقادیر و عبارت‌ها در قالب نوع‌های داده، نحوه تغییر و کار با این نوع‌ها و چگونگی تعامل آن‌ها با یکدیگر مربوط می‌شود. هدف یک سیستم تعیین نوع، تایید و معمولاً تعیین میزانی از صحت و درستی در برنامه‌های نوشته شده به وسیله یک زبان برنامه نویسی از طریق شناسایی برخی از عملیات ناصحیح است. هر گونه‌ای از سیستم تعیین نوع تصمیم‌پذیر (Decidable) شامل میزانی از مصالحه می‌شود. در حالی که این سیستم بسیاری از برنامه‌های ناصحیح را نمی‌پذیرد، اما همچنان می‌تواند صحت برخی از برنامه‌های درست اما غیرمعمول را هم نقض کند.

بررسی نوع در زبان‌های برنامه نویسی

برای عبور از این نقطه ضعف، برخی از زبان‌های برنامه نویسی دارای نارسایی‌هایی (Loophole) در خصوص تعیین نوع هستند، یعنی معمولاً تبدیل‌های بررسی نشده‌ای وجود دارند که ممکن است توسط برنامه نویس استفاده شوند تا به‌طور مشخص اجرای عملیاتی را بین نوع‌های متفاوت مُجاز کنند که در حالت معمولی اجرای آن‌ها غیرمجاز است. در اکثر زبان‌های با قابلیت تعیین نوع، سیستم تعیین نوع تنها برای بررسی نوع در برنامه‌ها استفاده می‌شود، اما در تعدادی از زبان‌های برنامه نویسی که معمولاً زبان‌های تابعی هستند، «استنباط نوع (Type Inference) «انجام می‌شود تا دیگر نیاز نباشد که برنامه نویس حاشیه‌نویسی برای نوع انجام دهد. طراحی رسمی و مطالعه سیستم‌های تعیین نوع را «نظریه نوع (Type Theory) «می‌نامند.

۴- ۴- مقایسه زبان های برنامه نویسی دارای نوع و زبان های بدون نوع

یک زبان برنامه نویسی، «دارای نوع (Typed) «نامید می‌شود، اگر نوع داده‌های سازگار با عملیات مربوطه در اصول مربوط به عملیات در آن زبان تعریف شده باشند. برای مثال، "this text between the quotes"یک رشته (String) است و در بسیاری از زبان‌های برنامه نویسی، عملیات

تقسیم کردن یک داده از نوع رشته‌ای هیچ معنایی ندارد و اجرا نخواهد شد. عملیات ناصحیح و اشتباه می‌توانند در زمان کامپایل شدن برنامه شناسایی شوند و با یک پیام خطای کامپایل (ترجمه به زبان ماشین) به وسیله کامپایلر نقض شوند. به این روش، «بررسی نوع ایستا (Static Type Checking)» گفته می‌شود.

در روشی دیگر، خطای نوع در زمان اجرای برنامه شناسایی می‌شود و در پی آن یک استثناء زمان اجرا (Run-Time) رخ می‌دهد. به این روش، «بررسی نوع پویا (Dynamic Type Checking)» می‌گویند. بسیاری از زبان‌های برنامه نویسی از یک تابع به نام «کنترل کننده استثناء (Exception Handler)» استفاده می‌کنند تا این استثناء را مدیریت کنند و برای مثال همیشه-» ۱ «را به عنوان نتیجه بازمی‌گردانند.

زبان برنامه نویسی با نوع‌دهی پویا

زبان برنامه نویسی با نوع‌دهی ایستا

یک نوع خاص از زبان‌های دارای نوع، زبان‌های «تک نوعی (Single-Typed)» هستند. این زبان‌ها اغلب زبان‌های اسکریپت‌نویسی یا نشانه‌گذاری (Markup) مثل REXX یا SGML هستند و تنها یک نوع داده دارند. این نوع داده هم در این زبان‌ها معمولاً از نوع رشته کاراکتری است که هم برای داده‌های نمادین و هم برای داده‌های عددی استفاده می‌شوند.

از طرف دیگر، یک «زبان بدون نوع (Untyped Language)» مثل اکثر زبان‌های اسمبلی، اجازه اجرای هر نوع عملیاتی را روی داده‌ها می‌دهند که این داده‌ها اکثراً دنباله‌هایی از بیت‌هایی با

طول‌های مختلف هستند. از جمله زبان‌های بدون نوع سطح بالا می‌توان به Tcl ، BCPLو برخی از گونه‌های زبان Forth اشاره کرد.

عملاً در حالی‌که زبان‌های اندکی براساس نظریه نوع (تایید یا نقض همه عملیات) به عنوان زبان دارای نوع در نظر گرفته می‌شوند، اکثر زبان‌های برنامه نویسی جدید و مُدرن، میزانی از بررسی نوع را انجام می‌دهند. بسیاری از زبان‌های برنامه نویسی مورد استفاده در تولید نرم افزارهای تجاری، ابزارها و روش‌هایی را برای عبور کردن از سد سیستم بررسی نوع یا تضعیف آن ارائه کرده‌اند و در این راه، در ازای فراهم کردن امکان مدیریت بیش‌تر روی اجرای برنامه، امنیت نوع (Type Safety) را تقلیل داده‌اند. در این خصوص مفهوم «Casting» (تبدیل نوع موقت) مطرح می‌شود که از موضوع بحث این مقاله خارج است.

در بررسی نوع ایستا(Static typing) ، نوع تمام عبارت‌های برنامه پیش از زمان اجرای آن و معمولاً در زمان کامپایل تعیین می‌شوند. برای مثال، 1و 2+2عبارت‌ها یا گزاره‌هایی از نوع عدد صحیح (Integer)هستند. این گزاره‌ها را نمی‌توان به تابعی ارجاع داد که انتظار دریافت داده‌ای از نوع رشته‌ای را دارد. همچنین امکان ذخیره‌سازی این داده‌ها در متغیری وجود ندارد که برای نگهداری تاریخ تعریف شده است.

زبان‌هایی با بررسی نوع ایستا می‌توانند بررسی نوع آشکار (Manifest Typing) داشته باشند یا اینکه بررسی نوع در آن‌ها به صورت استنباط نوع (Type Inference) باشد. در حالت اول، برنامه نویس باید به‌طور واضح و مشخص نوع داده را به صورت برخی از موقعیت‌های متنی بنویسد (مثلاً نوع داده در هنگام تعریف متغیر مشخص می‌شود). در حالت دوم، کامپایلر نوع داده گزاره‌ها را براساس محتوای آن استنباط می‌کند. اکثر زبان‌های مطرح و جریان اصلی با بررسی نوع ایستا، مثل C# ، C++و جاوا بررسی نوع را به صورت استنباطی انجام می‌دهند.

استنباط کامل نوع از گذشته در زبان‌های برنامه نویسی کم‌تر شناخته شده‌ای مثل Haskell و ML به گرفته می‌شد. اگرچه، بسیاری از زبان‌های برنامه نویسی با بررسی نوع آشکار از استنباط نوع ناقص یا نسبی پشتیبانی می‌کنند. برای مثال، C++، جاوا و C# همگی در برخی از موارد محدود و خاص از استنباط نوع پشتیبانی می‌کنند. علاوه بر این، برخی از زبان‌های برنامه نویسی این امکان را می‌دهند تا برخی از انواع داده به صورت خودکار به انواع داده دیگر تبدیل شوند. برای مثال، می‌توان زمانی که برنامه انتظار داده‌ای از نوع اعشاری شناور (Float) را دارد، به جای آن از داده‌ای با نوع int (صحیح) استفاده کرد.

در بررسی نوع پویا (Dynamic Typing) که به آن «Latent Typing» یعنی «بررسی نوع پنهانی» هم می‌گویند، ایمنی نوع عملیات در زمان اجرا بررسی می‌شود. به بیان دیگر، نوع داده‌ها به جای اینکه با گزاره‌های متنی مرتبط باشند، با مقادیر زمان اجرا مرتبط هستند. برخلاف زبان‌های با نوع‌دهی آشکار، زبان‌هایی که بررسی نوع را به صورت پویا انجام می‌دهند، برنامه نویس را ملزم به نوشتن نمادهای نوع‌دهی مشخص در گزاره‌ها نمی‌کنند.

علاوه بر قابلیت‌های دیگر، بررسی نوع پویا این امکان را فراهم می‌کند تا یک متغیر واحد بتواند در مراحل مختلف اجرای برنامه به مقادیری با نوع داده‌های گوناگون اشاره کند. اگرچه در این شیوه خطای نوع‌دهی تا زمانی که یک قطعه کد اجرا نشود، به‌طور خودکار قابل شناسایی نیست که این مسئله باعث می‌شود فرآیند «عیب‌یابی» (دیباگ کردن) به‌طور بالقوه دشوارتر شود. زبان‌های Lisp،Smalltalk، پرل، پایتون، جاوا اسکریپت و روبی همگی نمونه‌هایی از زبان‌های برنامه نویسی با نوع‌دهی پویا هستند.

۴- ۵- نوع دهی ضعیف و نوع دهی قوی در زبان برنامه نویسی

نوع‌دهی یا تعیین نوع ضعیف (Weak typing) به این صورت است که می‌توان با مقداری از یک نوع به عنوان نوع دیگر رفتار کرد. برای مثال، در نوع‌دهی ضعیف می‌توان با یک نوع رشته‌ای به عنوان یک عدد برخورد کرد. این قابلیت می‌تواند در برخی مواقع بسیار کاربردی و مفید باشد. البته این شیوه ممکن است باعث شود در زمان کامپایل یا حتی در Runtime برخی از انواع خطاها در برنامه شناسایی نشوند.

در روش نوع‌دهی قوی (Strong Typing) امکان جلوگیری از بروز اینگونه خطاها وجود دارد. تلاش برای انجام عملیات روی نوع ناصحیحی از یک مقدار در نوع‌دهی قوی منجر به صدور خطا می‌شود. از زبان‌های برنامه نویسی با نوع‌دهی قوی اغلب با عنوان «نوع ایمن (Type Safe) «یا «ایمن» یاد می‌شود.

تعریفی دیگر برای زبان‌های با نوع‌دهی ضعیف به زبان‌هایی مثل Perl و جاوا اسکریپت اطلاق می‌شود که اجازه می‌دهند تعداد زیادی تبدیل نوع انجام شود. برای مثال در جاوا اسکریپت، عبارت x * 2 به‌طور ضمنی و تلویحی x را به نوع عددی تبدیل می‌کند و این تبدیل نوع حتی اگر xاز نوع null، undefined، آرایه یا رشته‌ای از حروف باشد هم با موفقیت انجام خواهد شد. چنین تبدیل نوع‌های تلویحی اغلب مفید هستند، اما ممکن است خطاهای برنامه نویسی را بپوشانند.

به‌طور کلی، نوع‌دهی قوی و ایستا مفاهیمی متعامد به حساب می‌آیند اما کاربرد آن‌ها در ادبیات متفاوت است. برخی افراد اصطلاح نوع‌دهی قوی را برای اشاره به نوع‌دهی ایستای قوی (Strongly, Statically Typed)به‌کار می‌برند یا حتی به‌گونه‌ای گیج‌کننده‌تر، منظور آن‌ها تنها نوع‌دهی ایستا است. مثلاً از زبان C هم به عنوان زبانی با نوع‌دهی قوی یاد می‌کنند و هم آن را زبانی با نوع‌دهی ایستای ضعیف به حساب می‌آورند.

ممکن است برای برخی از برنامه نویسان حرفه‌ای عجیب به‌نظر برسد که زبان C می‌تواند به عنوان زبانی با نوع‌دهی ایستای ضعیف در نظر گرفته شود. اگرچه باید مد نظر داشت که استفاده از اشاره‌گر عمومی یعنی اشاره‌گر *void امکان Casting (تبدیل نوع موقت) اشاره‌گرها را به سایر اشاره‌گرها بدون نیاز به تبدیل نوع ضمنی (Explicit Cast) فراهم می‌کند. این قابلیت به میزان زیادی مشابه تبدیل نوع موقت آرایه‌ای از بایت‌ها به هر نوعی از انواع داده در زبان C بدون استفاده از تبدیل نوع ضمنی مثل (int)یا (char)است.

۴- ۶- کتابخانه استاندارد و سیستم زمان اجرا در زبان برنامه نویسی

اکثر زبان‌های برنامه نویسی دارای یک کتابخانه هسته‌ای متصل هستند که اغلب به آن کتابخانه استاندارد گفته می‌شود. خصوصاً در صورتی که این کتابخانه به عنوان بخشی از استاندارد زبان منتشر شده گنجانده شده باشد. این کتابخانه هسته‌ای به‌طور معمول به وسیله تمام پیاده‌سازی‌های زبان مربوطه در دسترس قرار داده می‌شوند. کتابخانه‌های هسته‌ای معمولاً شامل تعاریفی برای الگوریتم‌های رایج مورد استفاده، ساختارهای داده و ساز و کارهایی برای ورودی و خروجی می‌شوند.

مرز بین یک زبان و کتابخانه هسته‌ای آن در زبان‌های برنامه نویسی مختلف متفاوت است. در برخی موارد، طراحان زبان ممکن است با کتابخانه هسته‌ای به عنوان یک موجودیت مجزا و جدا از آن زبان رفتار کنند. اگرچه، کتابخانه هسته‌ای یک زبان اغلب توسط کاربران آن زبان به عنوان بخشی از آن در نظر گرفته می‌شود و حتی ممکن است برای برخی از مشخصه‌های این زبان نیاز

باشد که کتابخانه استاندارد در تمام پیاده‌سازی‌ها در دسترس باشد. در واقع، برخی از زبان‌ها به‌گونه‌ای طراحی شده‌اند که حتی معنا و مفهوم برخی از ساختارهای نحوی خاص در آن‌ها را نمی‌توان بدون اشاره به کتابخانه هسته‌ای آن زبان توصیف کرد.

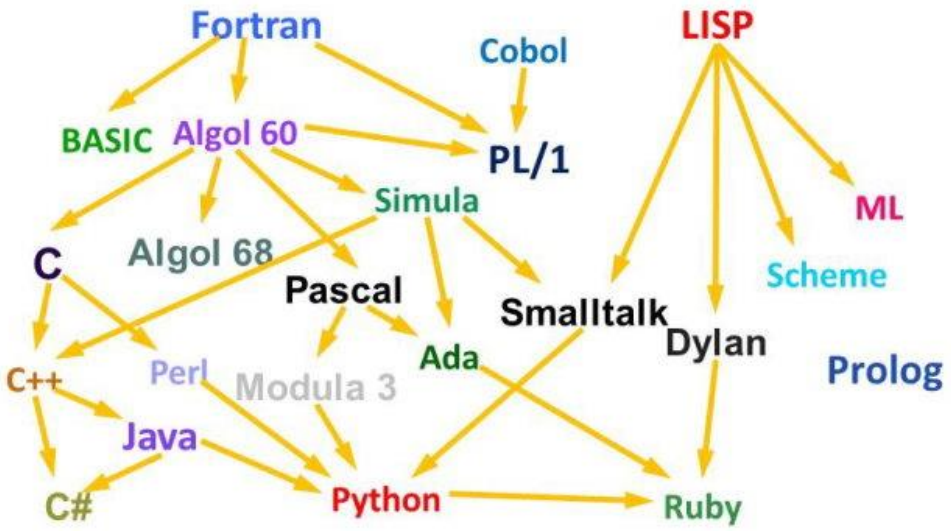
برای مثال، در جاوا یک رشته لیترال به عنوان نمونه‌ای از کلاس java.lang.String تعریف می‌شود. به‌طور مشابه، درSmalltalk ، یک گزاره تابع بی‌نام (Anonymous) یا یک «بلاک (block) « نمونه‌ای از کلاس BlockContext آن کتابخانه را می‌سازد. از سوی دیگر، اسکیم « (Scheme) حاوی چندین زیرمجموعه یکپارچه و پیوسته که برای ساخت بخش‌های باقیمانده زبان مربوطه به عنوان ماکروهای کتابخانه کفایت می‌کند و به این ترتیب، طراحان زبان حتی به این زحمت نخواهند افتاد که بگویند کدام بخش‌های زبان باید به عنوان ساختارهای زبانی پیاده‌سازی شوند و کدام بخش‌ها باید به عنوان اجزایی از یک کتابخانه پیاده‌سازی شوند.

۴- ۷- طراحی و پیاده سازی زبان های برنامه نویسی چگونه است

زبان‌های برنامه نویسی خصوصیت‌های مشترکی با زبان‌های طبیعی دارند که این وجه اشتراک به هدف و مقصود آن‌ها به عنوان وسایل ارتباطی مربوط می‌شود. هم زبان‌های طبیعی و هم زبان‌های برنامه نویسی دارای قالبی نحوی هستند که جدا از معناشناسی آن‌ها است که این مسئله نشان دهنده خانواده‌های زبانی تشکیل شده از زبان‌های مرتبط با یکدیگر است که یکی از دیگری منشعب می‌شود.

اما زبان‌های برنامه نویسی به عنوان ساختارهای مصنوعی، به شکلی اساسی و بنیادی نسبت به زبان‌هایی متفاوت هستند که از طریق استفاده و کاربردشان تکامل یافته‌اند. تفاوت مهم در اینجا است که یک زبان برنامه نویسی می‌تواند به‌طور کامل توصیف و مطالعه شود. چرا که این زبان دارای تعریفی دقیق و پایان‌پذیر است. بر خلاف آن، زبان‌های طبیعی دارای معانی متغیر هستند که توسط کاربران آن‌ها در اجتماعات مختلف تبیین می‌شوند. در حالی که زبان‌های ساخت‌یافته

زبان‌های مصنوعی هم به حساب می‌آیند که از پایه با هدف و مقصودی مشخص طراحی شده‌اند. آن‌ها برخلاف زبان‌های برنامه نویسی ، کاستی‌هایی در خصوص تعریف کامل و دقیق‌شان دارند.



شجره‌نامه زبان های برنامه نویسی

بسیاری از زبان‌های برنامه نویسی از صفر طراحی شده‌اند، تغییراتی برای سازگاری با نیازهای جدید در آن‌ها اعمال شده است و با سایر زبان‌ها تلفیق شده‌اند. بسیاری از آن‌ها در نهایت به ورطه بلا استفادگی کشیده شده‌اند. اگرچه تلاش‌هایی برای طراحی یک زبان برنامه نویسی جهان‌شمول برای تمام مقاصد صورت گرفته است، اما تمام این تلاش‌ها برای پذیرش زبانی در این جایگاه با شکست مواجه شده‌اند.

۴- ۸- علت نیاز به چندین زبان برنامه نویسی

نیاز به زبان‌های برنامه نویسی متنوع به دلیل وجود زمینه‌های مختلف و متفاوت ایجاد شده است:

- برنامه نویسان در مورد میزان تخصص با یکدیگر تفاوت دارند. برنامه نویسان تازه‌کار بیش از هر چیز نیازمند ساز و کاری ساده و آسان هستند و برنامه نویسان حرفه‌ای هم معمولاً با بسیاری از پیچیدگی‌های قابل توجه مشکلی ندارند.
- برنامه‌ها باید تعادل لازم را میان سرعت، اندازه و سادگی در سیستم‌هایی برقرار کنند که گستره آن‌ها از میکروکنترلرها شروع می‌شود و تا اَبَرکامپیوترها ادامه پیدا می‌کند.
- برنامه‌ها ممکن است یک بار نوشته شوند و برای مدت‌های مدید تغییر نکنند یا ممکن است برنامه‌هایی هم وجود داشته باشند که دائماً نیاز به اعمال تغییرات مختلف در آن‌ها وجود داشته باشد.
- معمولاً سلیقه برنامه نویسان متفاوت است: هر برنامه نویس به حل مسئله‌ها به‌گونه‌ای خاص و ابراز آن‌ها با زبان‌های به‌خصوص عادت دارند.

۴- ۹- اهمیت انتزاع در زبان برنامه نویسی

یک روند رایج در توسعه زبان‌های برنامه نویسی این موضوع بوده است که سطح بالاتری از انتزاع (Abstraction)برای افزایش توانایی حل مسائل به‌کار گرفته شود. زبان‌های برنامه نویسی اولیه، وابستگی زیادی به سخت‌افزار کامپیوتری داشتند که روی آن اجرا می‌شدند. با توسعه و ساخت زبان‌های برنامه نویسی جدید، قابلیت‌هایی اضافه شده‌اند که به برنامه نویسان اجازه می‌دهند تا بتوانند ایده‌هایی را ابراز کنند.

این ایده‌ها اغلب از ترجمه صرف به دستورالعمل‌های ساخت‌افزاری فاصله دارند. بنابراین با توجه به اینکه برنامه نویسان درگیری کم‌تری با پیچیدگی‌های سخت‌افزار کامپیوتر دارند، برنامه‌های ساخته شده توسط آن‌ها می‌توانند محاسبات بیش‌تری را با تلاش کم‌تری از جانب برنامه نویسان انجام دهند. این مسئله به برنامه نویسان اجازه می‌دهد تا کارایی بیش‌تری را در هر واحد زمانی داشته باشند.

۴- ۱۰- زبان برنامه نویسی طبیعی

برنامه نویسی به زبان طبیعی (Natural Language programming) راهی برای حذف نیاز به زبان تخصصی برای برنامه نویسی به حساب می‌آید. اگرچه، این هدف هنوز چندان دست‌یافتنی نیست و البته در خصوص مزایای آن هم جای بحث وجود دارد. ادسخر دیکسترا بر این عقیده بود که استفاده از یک زبان رسمی برای جلوگیری از معرفی ساختارهای بی‌معنی بسیار ضرورت دارد و ایده برنامه نویسی به زبان طبیعی را احمقانه می‌داند. در خصوص برنامه نویسی به زبان‌های طبیعی، تاکنون رویکردهایی ترکیبی در زبان‌های Structured English و SQL پیاده‌سازی شده است.

طراحان و کاربران یک زبان باید مصنوعاتی بسازند که رویه برنامه نویسی را ممکن می‌سازند و باعث می‌شوند که بتوان فرآیند برنامه نویسی را مدیریت کرد. مهم‌ترین جزء این مصنوعات، مشخصه‌های آن زبان (Language Specification) و «پیاده‌سازی» آن به حساب می‌آیند. در ادامه به شرح هر یک از این دو مورد پرداخته شده است.

۴- ۱۱- مشخصه ها یا اصول زبان های برنامه نویسی

مشخصه‌ها یا اصول یک زبان برنامه نویسی، دست‌سازه‌هایی هستند که کاربران آن زبان و طراحان آن در مورد آن‌ها به توافق رسیده‌اند تا مشخص شود که آیا یک قطعه کد منبع برنامه‌ای معتبر در آن زبان است و در این صورت، رفتار آن چگونه خواهد بود؟ مشخصه‌های یک زبان برنامه نویسی می‌توانند شکل‌های مختلفی داشته باشند که در ادامه فهرست شده‌اند:

- تعریفی ضمنی از سینتکس آن زبان، معناشناسی ایستا و معناشناسی اجرایی آن زبان از جمله اشکال مشخصه‌های یک زبان برنامه نویسی هستند. در حالی که سینتکس معمولاً با استفاده از قواعد دستوری رسمی مشخص می‌شود، تعاریف معنایی ممکن است به زبان طبیعی نوشته شوند) برای مثال می‌توان به زبان C اشاره کرد (یا آن‌ها را به صورت معناشناسی رسمی بنویسند) مثلاً در Standard ML و Scheme این کار انجام شده است.

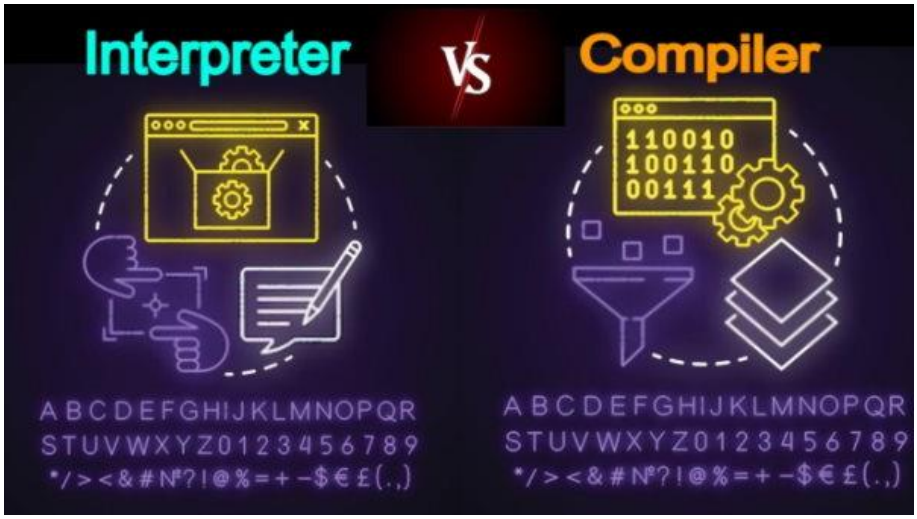
- توصیف رفتار یک کامپایلر برای آن زبان نیز یکی دیگر از انواع مشخصه‌های زبان برنامه نویسی محسوب می‌شود. برای مثال می‌توان مشخصه‌های زبان‌های C++ و Fortran را نام برد. سینتکس و معناشناسی آن زبان باید از این توصیف‌ها استنباط شود که ممکن است به زبان طبیعی یا به زبان رسمی نوشته شده باشند.
- ارجاع یا پیاده‌سازی یک مدل که گاهی به زبانی نوشته می‌شود که مشخصه‌های آن تعیین می‌شوند نیز از جمله موارد مشخصه‌های زبان‌های برنامه نویسی به حساب می‌آید. برای مثال، در این خصوص می‌توان Prolog یا ANSI REXX را نام برد. سینتکس و معناشناسی یک زبان در رفتار پیاده‌سازی مرجع آن صریح و روشن است.

۴- ۱۲- پیاده سازی زبان های برنامه نویسی به چه معناست

یک پیاده‌سازی از یک زبان برنامه نویسی روشی را برای نوشتن برنامه‌ها به آن زبان فراهم می‌کند و آن‌ها را روی یک یا بیش از یک پیکربندی سخت‌افزاری و نرم‌افزاری اجرا می‌کند. به‌طور گسترده دو رویکرد برای پیاده‌سازی زبان‌های برنامه نویسی وجود دارند:

- کامپایل کردن(Compilation)
- تفسیر کردن(Interpretation)

به‌طور کلی امکان پیاده‌سازی یک زبان با هر یک از دو روش فوق امکان‌پذیر است. خروجی یک کامپایلر می‌تواند به وسیله سخت‌افزار یا برنامه‌ای به نام مفسر (Interpreter) اجرا شود. در برخی از پیاده‌سازی‌هایی که از رویکرد مفسری استفاده می‌کنند، هیچ مرز مشخصی میان کامپایل و تفسیر وجود ندارد. برای مثال، در برخی از پیاده‌سازی‌های زبانBasic ، ابتدا عملیات کامپایل انجام می‌شود و سپس کدهای منبع به صورت خط به خط اجرا می‌شوند.



برنامه‌هایی که به‌طور مستقیم روی سخت‌افزار اجرا می‌شوند، معمولاً سرعت اجرای بسیار بیش‌تری نسبت به زبان‌های تفسیر شده دارند. یک روش برای بهبود اجرای برنامه‌های تفسیری، استفاده از شیوه کامپایل درجا (Just-in-Time Compilation) است. در این روش، ماشین مجازی بلافاصله پیش از اجرا، بلوک‌های بایت‌کدی را ترجمه می‌کند که قرار است در کدهای ماشین برای اجرای مستقیم روی سخت‌افزار ترجمه شوند. حال در ادامه مقاله زبان برنامه نویسی چیست به شرح چیستی زبان اختصاصی پرداخته شده است.

۴- ۱۳- زبان برنامه نویسی اختصاصی

اگرچه اکثر زبان‌های برنامه نویسی رایج و پر استفاده، دارای مشخصه‌ها و پیاده‌سازی‌هایی کاملاً آزاد هستند، بسیاری از زبان‌های برنامه نویسی هم وجود دارند که «اختصاصی» به حساب می‌آیند. این زبان‌های اختصاصی (Proprietary Languages) تنها از سوی یک سازنده و تولید کننده ارائه می‌شوند. سازندگان این زبان‌ها مالکیت معنوی زبان‌های اختصاصی مربوطه را متعلق به خود می‌دانند. زبان‌های برنامه نویسی اختصاصی معمولاً زبان‌های با دامنه خاص (Domain Specific Languages)هستند یا اینکه به عنوان زبان‌های اسکریپت‌نویسی تعبیه شده برای یک محصول خاص مورد استفاده قرار می‌گیرند. برخی از زبان‌های اختصاصی تنها به صورت داخلی در یک

سازمان تجاری به‌کار گرفته می‌شوند. این در حالی است که زبان‌های اختصاصی دیگری هم وجود دارند که برای کاربران خارج از آن دایره در دسترس قرار دارند.

جایگاه برخی از زبان‌های برنامه نویسی هم در جایی بین زبان‌های اختصاصی و زبان‌های آزاد قرار می‌گیرد. برای مثال، شرکت اوراکل مالکیت معنوی برخی از بخش‌های زبان برنامه نویسی جاوا را متعلق به خود می‌داند. یا مثلاً زبان برنامه نویسی C# که توسط مایکروسافت ساخته شده است دارای پیاده‌سازی‌های آزاد بسیاری در اکثر بخش‌های سیستم خود است. با این حال این زبان نیز بخشی به نام زمان اجرای زبان مشترک یا به اختصار CLR دارد که محیطی بسته و اختصاصی به حساب می‌آید.

۴- ۱۴- کاربرد زبان برنامه نویسی

تاکنون هزاران زبان برنامه نویسی عمدتاً در حوزه محاسبات ساخته شده‌اند. معمولاً برای پروژه‌های نرم افزاری شخصی از ۵ یا بیش از ۵ زبان برنامه نویسی استفاده می‌شود. زبان‌های برنامه نویسی نسبت به بسیاری از سایر اشکال بیان انسانی متفاوت هستند، چرا که آن‌ها نیاز به درجه‌ای از دقت و کامل بودن دارند. در زمان استفاده از یک زبان طبیعی برای برقراری ارتباط با سایر افراد، نویسندگان و متکلمان انسانی ممکن است بیانی دوپهلو و مبهم داشته باشند و خطاهای اندکی هم از آن‌ها سر بزند و همچنان انتظار داشته باشند که قصد و قرض ان‌ها توسط دیگران قابل درک باشد.

اما، به قول معروف کامپیوترها دقیقاً کاری را انجام می‌دهند که از آن‌ها خواسته شده است و نمی‌توانند بفهمند که مثلاً در صورت وجود خطا، برنامه نویس قصد نوشتن چه کدهای را داشته است. ترکیب تعریف زبان برنامه نویسی، یک برنامه و ورودی‌های آن برنامه باید به‌طور کامل رفتار بیرونی را تعیین کنند که قرار است در زمان اجرای برنامه اتفاق بیوفتند.

یک زبان برنامه نویسی ساز و کاری ساختار یافته را برای تعریف بخش‌هایی از داده‌ها، عملیات و تبدیل‌هایی فراهم می‌کند که باید به صورت خودکار روی آن داده‌ها اجرا شوند. برنامه نویس برای نمایش مفهوم‌های مربوط به محاسبات مورد نظر از انتزاعی استفاده می کند که در زبان برنامه نویسی وجود دارد. این مفهوم‌ها به عنوان مجموعه‌ای از عنصرهای در دسترس) که پایه و مبنای آن زبان به حساب می‌آیند و به آن‌هاPrimitive می‌گویند (بازنمایی می‌شوند. برنامه نویسی فرآیندی است که در آن برنامه نویسان این اجزای پایه را با هم ترکیب می‌کنند تا برنامه‌هایی جدید بسازند یا تغییرات لازم را در برنامه‌های فعلی برای سازگاری با نیازهای جدید یا محیط‌های در حال تحول به وجود بیاورند.

برنامه‌های ساخته شده برای یک کامپیتور ممکن است در قالب پردازش دسته‌ای و بدون تعامل انسانی اجرا شوند. یا ممکن است کاربری دستورات را در یک نشست تعاملی در یک مفسر وارد کند. در چنین حالتی، دستورات (Commands) خیلی ساده به برنامه‌هایی گفته می‌شود که اجرای آن‌ها به یکدیگر زنجیر شده است. وقتی که یک زبان برنامه نویسی قابلیت اجرای دستورات را از

طریق مفسر (مثل یک شل یونیکس یا دیگر واسط‌های خط فرمان) و بدون کامپایل کردن داشته باشد، به زبان، زبان برنامه نویسی اسکریپتی[۹] می‌گویند.

۴- ۱۵- اندازه گیری میزان استفاده از زبان های برنامه نویسی مختلف

مشخص کردن این مسئله بسیار دشوار است که کدام زبان برنامه نویسی از همه بیش‌تر استفاده می‌شود. چرا که، مفهوم میزان استفاده در بسترها و چارچوب‌های مختلف تفاوت دارد. یک زبان ممکن است زمان بیش‌تری از وقت یک برنامه نویس را اشغال کند، تعداد خط کدهای یک زبان ممکن است بیش‌تر باشند یا یک زبان برنامه نویسی دیگر هم ممکن است زمان بیش‌تری از CPU را اشغال و مصرف کند. برخی از زبان‌های برنامه نویسی دیگر هم در برخی از کاربردهای خاص بسیار محبوب و رایج هستند.

برای مثال، زبان COBOL همچنان در مراکز داده سازمان‌های تجاری و اغلب در کامپیوترهای بزرگ (Mainframe)مورد استفاده قرار می‌گیرد، زبان Fortran در کاربردهای علمی و مهندسی به‌کار گرفته می‌شود؛ زبان Ada در هوافضا، حمل و نقل، موارد نظامی و کاربردهای زمان واقعی و تعبیه شده استفاده می‌شود؛ زبان C هم در کاربردهای تعبیه شده و سیستم عامل‌ها به‌کار می‌رود. سایر زبان‌ها هم اغلب برای توسعه و ساخت بسیاری از کاربردها و موارد استفاده دیگر مورد استفاده قرار می‌گیرند.

روش‌های گوناگونی در خصوص اندازه‌گیری میزان محبوبیت زبان‌های برنامه نویسی وجود دارند که هر کدام منوط به پیشقدری است که اندازه‌گیری می‌شود:

- شمارش تعداد آگهی‌های شغلی که یک زبان برنامه نویسی را به عنوان مهارت مورد نیاز اعلام می‌کنند.

[۹] Scripting Programming Language

- تعداد کتاب‌هایی که یک زبان برنامه نویسی را آموزش می‌دهند یا آن را معرفی و توصیف می‌کنند.
- تخمین تعداد خط کدهای نوشته شده به یک زبان هم روش دیگری برای تعیین میزان محبوبیت زبان‌های مختلف به حساب می‌آید. البته معمولاً در این روش، زبان‌هایی که چندان در جستجوهای عمومی ظاهر نمی‌شوند دست کم گرفته خواهند شد.
- شمارش تعداد ارجاع‌ها یک زبان برنامه نویسی (یعنی شمارش میزان اشاره به نام آن زبان) در یک موتور جستجوی وب

با ترکیب و جمع‌آوری اطلاعات از وب‌سایت‌های مختلف، سایت stackify.com ده تا از محبوب‌ترین زبان‌های برنامه نویسی را به ترتیب نزولی زیر معرفی کرده است:

۱) جاوا
۲) زبانC
۳) C++
۴) پایتون
۵) C#
۶) جاوا اسکریپت
۷) VB.NET
۸) زبان برنامه نویسیR
۹) PHP
۱۰) متلب

همچنین ده تا از زبان‌های برنامه نویسی آینده که توسط وب‌سایت Stackify معرفی شده‌اند نیز در ادامه به ترتیب معرفی شده‌اند:

۱) پایتون

۲) زبان برنامه نویسیR

۳) سوئیفت

۴) Go

۵) اسکالا

۶) C#

۷) کاتلین

۸) جاوا ۸

۹) متلب

۱۰) Solidity

فصل پنجم- انواع خطاها در برنامه نویسی

۵- ۱- خطای نحوی (Syntax Error)

۵- ۲- خطای زمان کامپایل (Compile Error)

۵- ۳- خطای زمان اجرا (Run-Time Error)

۵- ۴- خطای منطقی (Logical Error)

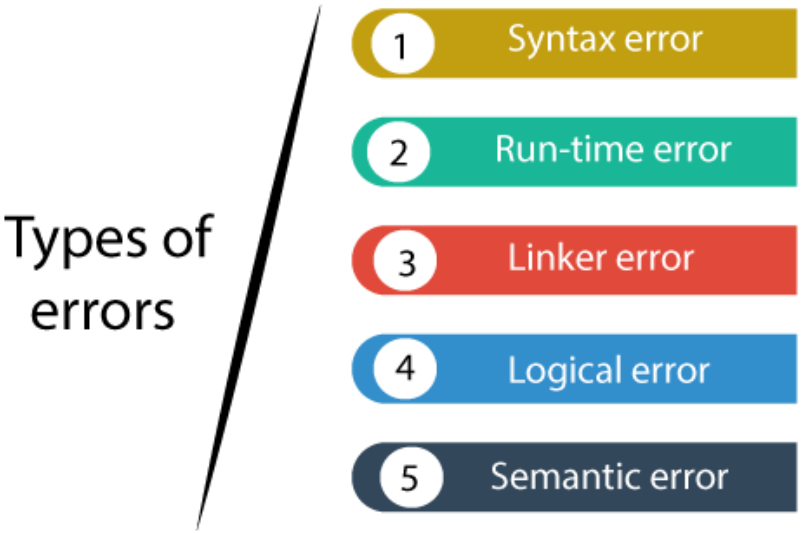
۵- ۵- سایر خطاهای برنامه نویسی

۵- ۵- ۱- خطای منبع (Resource Error)

۵- ۵- ۲- خطای واسط (Interface Error)

۵- ۶- مفهوم مهارت برنامه نویسی

چهار نوع خطای رایج در برنامه نویسی وجود دارد که در ادامه توضیح هر کدام آمده است.



۵- ۱- خطای نحوی (Syntax Error)

اولین، رایج‌ترین و شاید ساده‌ترین نوع خطا، خطای نحوی است. این خطا مربوط به زمان نوشتن کدهای برنامه (قبل از کامپایل و اجرا شدن) است. در انگلیسی به این خطا **Syntax Error**گفته می‌شود. در فارسی با نام‌های خطای گرامری، خطای دستوری یا خطای نوشتاری کد نیز شناخته می‌شود. اگر در نوشتن یک برنامه به زبان مورد نظر، اصول (دستور گرامری) آن زبان را رعایت نکنیم، با خطای دستوری در برنامه نویسی مواجه خواهیم شد. در انگلیسی به این خطا syntax error(سینتکس ارور) گفته می‌شود. در فارسی با نام‌های **خطای گرامری، خطای دستوری** یا **خطای نوشتاری کد** نیز شناخته می‌شود.

بگذارید با یک مثال ساده این نوع خطا را توضیح دهم.

فرض کنید در زبان فارسی یک نفر بگوید «من خواهید آمد»! 😐 قاعدتاً از او ایراد می‌گیریم که ساختار جمله‌ای که گفتی اشتباه است!

یا شاید بگوید «او نشستند»؛ این جمله باز هم از لحاظ دستوری یا گرامری اشتباه است.

همه زبان‌های برنامه‌نویسی یک ساختار دستوری مشخصی دارند. به این ساختار اصطلاحاً syntax گفته می‌شود.

اگر در نوشتن یک برنامه به زبان مورد نظر، اصول (دستور گرامری) آن زبان را رعایت نکنیم، با خطای دستوری در برنامه نویسی مواجه خواهیم شد.

مثلاً برای **تعریف رشته متنی** معمولاً آن را درون علامت کوتیشن می‌گذاریم. اگر یک رشته متنی به صورت زیر تعریف کنیم، کد ما دارای خطای نحوی است چون در ابتدای آن "قرار نداده‌ایم.

wrong_string = Wrong defined sample string"

یا فرض کنید در یک جای برنامه قرار می‌دهیم = iو دیگر چیزی نمی‌نویسیم. این تعریف ناقص است و باعث می‌شود کد ما دارای خطای گرامری باشد.

معمولاً در استفاده از ابزارهای برنامه‌نویسی(IDEها) این خطاها تشخیص داده شده و به ما اعلام می‌شود. کامپایلرها نیز قبل از ترجمه کد، ساختار آن را از نظر دستوری بررسی می‌کنند.

۵- ۲- خطای زمان کامپایل (Compile Error)

خطای دیگر خطای همزمان با ترجمه یا Compile Time Error است. معمولاً این ارور در زبان‌های برنامه نویسی کامپایلری رخ می‌دهد. در اجرای برنامه یاد گرفتیم که زبان‌های کامپایلری مثل زبان سی شارپ قبل از اجرا باید compile شوند. فرآیند ترجمه کد پیچیده است اما در همین حد بدانید که لازم است مقدماتی برای آن فرآهم شود. یکی از مقدماتی که به ما مربوط است، وجود کلیه فایل‌های یک برنامه (اگر برنامه دارای چند فایل کد است) و در دسترس بودن کتابخانه‌های استفاده شده در برنامه است. اگر در هنگام کامپایل کردن کد، یک یا چند مورد از این مقدمات فرآهم نباشد، با کامپایل ارور مواجه خواهیم شد.

۵- ۳- خطای زمان اجرا (Run-Time Error)

فرض کنید کد به درستی نوشته شده و با موفقیت کامپایل شده است. در هنگام اجرای کد، شرایطی به وجود می‌آید که این کد به درستی اجرا نمی‌شود. برای مثال، اگر برنامه ما فایلی را در کنار خود لازم داشته باشد تا چیزی از داخل آن خوانده شده یا در آن بنویسید و این فایل وجود نداشته باشد، با خطای زمان اجرا در برنامه‌نویسی مواجه خواهیم شد. در واقع خطایی در هنگام کار با فایل رخ داده است. خطای Run Time یک خطای غیر منتظره است. یعنی همه چیز به درستی عمل می‌کند تا این که یک نقص به وجود می‌آید. یکی از خطاهای معروف زمان اجرا در برنامه نویسی، عملیات تقسیم بر صفر است. می‌دانیم که تقسیم هر عددی بر صفر، یک مقدار تعریف نشده به شمار می‌آیند. حال اگر ما یک ماشین حساب نوشته باشیم که دو عدد ورودی از کاربر و عملگر را گرفته و حاصل را برمی‌گرداند. اگر کاربر سیستم عملیات تقسیم را انجام دهد و عدد دوم را صفر تعریف کند، با خطای زمان اجرا مواجه می‌شویم.

به این کار **مدیریت خطا (Error Handling)** گفته می‌شود. در این فرآیند، شرایطی که احتمال می‌دهیم باعث بروز خطا شوند را بررسی می‌کنیم. مثلاً:

- در هنگام گرفتن عدد از کاربر به عنوان مقسوم علیه، با یک ساختار شرطی ساده چک کنیم که عدد مخالف صفر باشد.
- در هنگام کار با فایل یا مواردی که مرتبط با سیستم است، یک حالت دوم نیز در نظر بگیریم. معمولاً این کار با تعریف بلوک‌های try-catch انجام می‌شود.

۵- ۴- خطای منطقی (Logical Error)

خطرناک‌ترین نوع خطا، خطای منطقی در برنامه‌نویسی است! تا این جا ۳ نوع اصلی خطای برنامه‌نویسی را با هم بررسی کردیم. این خطاها را کامپیوتر تا حدودی می‌تواند تشخیص دهد. حتی انسان‌ها نیز به کمک کامپیوترها می‌آیند تا از بروز خطاهای قبلی جلوگیری کنند. در خطای

منطقی، کدها صحیح هستند، کامپایل به درستی انجام می‌شود و ورودی‌های برنامه نیز کاملاً صحیح و بدون ایراد خواهند بود. اما نتیجه کار، اشتباه است! فرض کنید عمل ضرب برای یک ماشین حساب را نوشته‌اید. در سورس برنامه به جای عملگر ضرب از عملگر جمع استفاده کرده‌اید. کد به درستی اجرا می‌شود ولی نتیجه‌ای که در انتها داریم صحیح نیست. این نوع ایرادات در فرآیند اجرای برنامه هیچ مشکلی ایجاد نمی‌کند. به این دلیل خطای منطقی را یک باگ نرم افزاری در نظر می‌گیریم که نتیجه اشتباه است. به همین دلیل است که گفتم خطرناک‌ترین خطای برنامه‌نویسی همین نوع خطاست.

۵- ۵- سایر خطاهای برنامه نویسی

با وجود اینکه با تمام خطاها آشنا شدیم، اما ۲ نوع خطای رایج دیگر در برنامه‌نویسی را با هم بررسی کنیم. در زبان‌های مختلف می‌توانیم ارورهای بیشتری داشته باشیم. مثلاً در زبان Cخطای Linker Error داریم که در در حالت کلی در دسته‌های بالا قرار می‌گیرد.

۵- ۵- ۱- خطای منبع(Resource Error)

هر برنامه‌ای برای اجرا به منابع مختلفی نیاز دارد. این منبع می‌تواند قدرت پردازشیCPU ، میزان فضای RAM یا حتی دستگاه‌های بیرونی (مثل پرینتر یا Wi-Fi)باشد. اگر منابعی که به برنامه داده شده است کافی نباشد، معمولاً برنامه ما درخواست منابع بیشتر از سیستم عامل خواهد کرد.

اگر سیستم عامل منابع بیشتری در اختیار نداشته باشد یا مقدار بیشتری از آن را در اختیار برنامه قرار ندهد، ممکن است با خطای resource مواجه شویم.

در حقیقت این خطا به این معنی است که برنامه نتوانسته منابع مورد نیاز خود را به اندازه کافی در اختیار بگیرد؛ در نتیجه اجرای آن با مشکل روبه‌رو خواهد شد.

۵- ۵- ۲- خطای واسط (Interface Error)

نرم‌افزارها و حتی سخت‌افزارها برای تعامل با هم از واسط (اینترفیس یا interface) استفاده می‌کنند. تعریف واسط بحث مفصلی است که می‌توانید در مورد آن در برنامه‌نویسی شئ‌گرا بیشتر بخوانید.

فرض کنید یک وب‌سرویس (API) داریم. برای استفاده از آن باید پارامترهایی را به همراه درخواست خود ارسال کنیم. اگر مقادیر ارسالی ما با داده‌هایی (فیلدهایی) که در مقصد مورد نیاز است تطابق نداشته باشد، با خطای واسط برنامه‌نویسی مواجه می‌شویم.

۵- ۶- مفهوم مهارت برنامه نویسی

سوال بعدی و مهم که مطرح می‌شود، این است که مهارت برنامه نویسی چیست؟ در پاسخ باید بگوئیم **مهارت برنامه نویسی که با عنوان مهارت کدنویسی (Coding Skill) هم شناخته می‌شود به هنر استفاده از زبان‌های برنامه نویسی مختلف برای نوشتن دستورات با هدف هدایت یک کامپیوتر، برنامه کاربردی (اپلیکیشن) یا برنامه نرم افزاری گفته می‌شود**. در مهارت برنامه نویسی کارها و وظایف مورد نظر برای کامپیوتر تعیین می‌شوند. مهارت‌های برنامه نویسی امکان ایجاد نرم افزارهای کامپیوتری، بازی‌ها، اپلیکیشن‌ها، وب‌سایت و بسیاری از موارد دیگر را فراهم می‌سازند.

مهارت کدنویسی یا همان مهارت برنامه نویسی به دانش و درک زبان‌ها، چارچوب‌ها و معماری‌هایی گفته می‌شود که یک برنامه نویس را قادر می‌سازند تا هر نوع محصول نرم افزاری را ایجاد کند.

در دنیایی که تماماً به صورت دیجیتالی متصل است، مهارت‌های کدنویسی تقریباً در تمام جنبه‌های زندگی انسان دخیل هستند. از این رو، پرورش مهارت‌های برنامه نویسی برای موفقیت در هر حوزه‌ای بسیار ضروری به نظر می‌رسد. برنامه نویسان کدهایی را برای ایجاد محصولات

دیجیتالی نوین با استفاده از مهارت‌های استثنایی کدنویسی خود خلق می‌کنند. برای خلق چنین محصولاتی تنها مهارت برنامه نویسی کافی نیست و باید این مهارت را با مهارت‌های تجزیه-تحلیل و تفکر خلاقانه تلفیق کرد.

فصل ششم- محیط یکپارچه توسعه

۶- ۱- IDE

۶- ۲- اهمیت IDE

۶- ۳- تاریخچه IDE

۶- ۴- ویژگی های رایج IDE ها

۶- ۵- تفاوت های بین ابزارهای محیط توسعه یکپارچه

۶- ۶- انواع IDE ها

۶- ۷- IDE چندزبانی و بهترین IDEهای چندزبانی

۶- ۸- بهترین IDE برای پایتون

۶- ۸- ۱- ویژگی‌های بهترین IDE های پایتون

۶- ۸- ۲- بهترین IDE های پایتون

با توجه به اینکه برنامه نویسان کامپیوتر برای کدنویسی و توسعه نرم افزار در زمینه‌های مختلف از طراحی و برنامه نویسی وب گرفته تا ایجاد اپلیکیشن‌های موبایل، ساخت بازی‌های کامپیوتری، هوش مصنوعی و بسیاری از کاربردهای دیگر به ابزار و محیط خاصی نیاز دارند، در این بخش به تعریف و بررسی IDE یا همان محیط یکپارچه توسعه [10] پرداخته شده است. IDEها محیطی برای کدنویسی، تست، اشکال‌زدایی[11] و سایر موارد این چنینی فراهم می‌کنند. در این بخش سعی شده است به‌طور جامع به این سوال پاسخ داده شود که IDE چیست و انواع و کاربردهای آن مورد بررسی قرار بگیرند. ابزار مناسب و محیط یکپارچه توسعه برنامه نویسی خوب می‌تواند بهره‌وری برنامه نویس را به میزان قابل توجهی افزایش دهد و به او کمک کند تا گردش کار پروژه را به راحتی پیش ببرد. از این رو، درک مفهوم و چیستی IDE از اهمیت بالایی برخوردار است و در این نوشتار به آن پرداخته می‌شود.

IDE -۱-۶

ابزارهای محیط توسعه یکپارچه یا همانIDE ها نرم افزارهایی هستند که امکاناتی را برای برنامه نویسان جهت کدنویسی، ساخت و توسعه برنامه‌ها و اپلیکیشن‌های دیگر فراهم می‌کنند. IDE برای دربرگرفتن تمام امکانات لازم برای پیاده‌سازی همه وظایف برنامه نویسی در قالب یک نرم افزار کاربردی طراحی شده است. یکی از اصلی‌ترین مزایایIDE ها این است که این برنامه کاربردی یک واسط[۱۲] مرکزی و هسته‌ای برای همه ابزارهای مورد نیاز توسعه نرم افزارهای خاص به شمار می‌رود. در ساخت یک برنامه، نیاز است بخش‌های بسیاری ایجاد شوند که همراه با یکدیگر کار می‌کنند، از جمله آن‌ها می‌توان به کدها، رابط کاربری[۱۳]، ساختمان پروژه، محیط پیکربندی پروژه و

[10] Integrated Development Environment
[11] Debugging
[۱۲] Interface
[۱۳] User Interface | UI

سایر موارد اشاره کرد. با استفاده از IDE همه این بخش‌ها می‌توانند از طریق یک نرم افزار واحد ایجاد و توسعه داده شوند.

تلاش‌ها برای پیاده‌سازی پردازش‌ها و فرایندهای توسعه نرم افزارهای گوناگون، باعث ایجاد مجموعه گسترده‌ای از ابزارهای نرم افزاری شده است که از جنبه‌های مختلفی طراحی و توسعه نرم افزارها را آسان‌تر می‌کنند. یکی از مهم‌ترین و کامل‌ترین مجموعه ابزارها، محیط توسعه نرم افزاری یا همان IDE به حساب می‌آید. IDEها معمولاً همه ابزارها و امکاناتی را ارائه می‌دهند که یک توسعه دهنده یا برنامه نویس برای نوشتن و ساختن برنامه از ابتدا تا انتها به آن‌ها نیاز دارد. توسعه دهندگان از IDE برای نوشتن، مدیریت و پیاده‌سازی کدهای خود در حین اجرای برنامه استفاده می‌کنند. این ابزار، فرایند توسعه برنامه‌ها و جنبه‌های مختلف ویرایش کدها را در برنامه‌های مستقل بسیار ساده‌تر می‌کند.

IDEیک برنامه نرم افزاری است که همه ابزارهای مورد نیاز یک پروژه توسعه نرم افزاری در آن ترکیب شده‌اند. از همان سطح‌های بسیار پایین‌تر، IDEها واسطی برای کاربران جهت نوشتن کدها، سازماندهی گروه‌های متنی و افزونه‌های برنامه نویسی خودکار به وجود می‌آوردند. معمولاً حداقل مواردی که می‌توان با استفاده ازIDE ها انجام داد شامل ویرایش، کامپایل، دیباگ یا همان اشکال‌زدایی، تکمیل و مدیریت کدها است. برخی ازIDE های پیشرفته‌تر، ویژگی‌ها و قابلیت‌های بیشتری را از جمله بصری‌سازی داده‌ها، ردیابی کردن برنامه و ارجاع متقابل[۱۴] ارائه می‌دهند.

برخی از محیط‌های توسعه یکپارچه فقط مختص به یک زبان برنامه نویسی خاص مانند پایتون یا جاوا هستند ولی برخی دیگر از آن‌ها، قابلیت بین زبانی[۱۵] دارند و در اکثر زبان‌های برنامه نویسی مورد استفاده قرار می‌گیرند. در برخی ازIDE ها یک یا چند کاربران می‌توانند به صورت سلسله مراتبی گروه‌هایی از کدها را در ناحیه مخصوص به خود ایجاد کنند و سپس آن‌ها را در کنار

[۱۴] Cross Reference
[۱۵] Cross Language

یکدیگر قرار دهند و کامپایل و پیاده‌سازی کنند. اکثرIDE ها دارای بخش اشکال‌زدایی داخلی هستند که در هنگام ساخت و پیاده‌سازی برنامه فعال می‌شود. اشکال‌زدایی به صورت بصری، یکی از مزیت‌های قابل توجه بسیاری ازIDE ها به حساب می‌آید.

IDEها در صورت برخورد با هر گونه اشکال یا خطا، به کاربران نشان می‌دهند که کدام بخش از کدها خطا دارد. IDEها مخصوصاً در برنامه نویسی‌های پیچیده بسیار مناسب هستند، زیرا امکان کمک به کدنویسی بهتر، تکمیل کدها، اشکال‌زدایی، نمایش تصویری کدها و تجزیه و تحلیل عمیق برنامه را فراهم می‌کنند. بسته به زبان برنامه نویسی مورد استفاده، اینIDE ها می‌توانند شامل الگوها، برجسته یا Highlight کردن سینتکس‌ها و Folding کدها برای بهبود تجربه توسعه برنامه نویسی باشند.

۶- ۲- اهمیت IDE

در حین فرایند نوشتن، ایجاد و تست نرم افزار در حال توسعه، برنامه نویسان و توسعه دهندگان انواع مختلفی از ابزارها را مورد استفاده قرار می‌دهند. ویرایشگرهای متن، کتابخانه کدها، نرم افزارهای ردیابی خطاها، کامپایلرها و پلتفرم‌های آزمایشی از رایج‌ترین ابزارهایی هستند که برای توسعه نرم افزار استفاده می‌شوند. توسعه‌دهندگانی که از IDE استفاده نمی‌کنند، باید این ابزارها را به صورت جداگانه انتخاب، مستقر و ادغام کنند و شخصاً بر کارکرد صحیح آن‌ها نظارت داشته باشند. ابزار محیط توسعه یکپارچه ، فریم ورک و چارچوبی به حساب می‌آید که بسیاری از این ابزارهای توسعه برنامه نویسی و نرم افزارها را به صورت یک‌جا در خود جای داده و با یکدیگر ترکیب کرده است.

زمانی که همه ابزارهای کاربردی در یک میزکار و ابزار واحد نمایش داده می‌شوند، دیگر نیازی نیست که توسعه دهندگان زمانی را صرف یادگیری هر کدام از آن‌ها به صورت جداگانه کنند. همچنین، این قابلیت برای افراد تازه‌کار در برنامه نویسی بسیار مفید است، زیرا نیازی نیست زمان زیادی را صرف یادگیری ابزارها در کنار کدنویسی کنند و فقط در کنار برنامه نویسی، نحوه کار با

IDEرا هم باید فرا بگیرند که کار چندان دشواری نیست. قابلیت‌های مفیدIDE ها از جمله تکمیل کد هوشمند، ایجاد خودکار کدها برای صرفه‌جویی در زمان، همراه با حذف نوشتن دنباله‌ای از کاراکترها به صورت کامل برای ساده شدن برنامه نویسی طراحی شده‌اند. هدف این ابزار نرم افزاری، انجام ساده‌تر توسعه نرم افزار و همچنین شناسایی و کاهش خطاها و اشتباهات نوشتاری و سینتکسی است.

سایر ویژگی‌هایIDE ها نیز به توسعه دهندگان در ساده‌سازی جریان کار و حل مسائل کمک می‌کنندIDE .ها کدها را همان‌گونه تجزیه می‌کنند که نوشته شده‌اند و مشکلات ایجاد شده را در زمان واقعی[۱۶] شناسایی خواهند کرد. همچنین، بیشترIDE ها از قابلیت‌های برجسته‌سازی سینتکس[۱۷] استفاده می‌کنند که سرنخ‌های بصری برای تشخیص اشکالات در کدها یا گرامر صحیح را در ویرایشگرهای متن به کاربر پیشنهاد می‌دهند .

۶- ۳- تاریخچه IDE

قبل از ایجادIDE ها، برنامه نویسان کدهای خود را در ویرایشگرهای متن می‌نوشتند. سپس کدها را پیاده‌سازی می‌کردند و خطاهای احتمالی موجود را یادداشت کرده و پس از آن به ویرایشگر متن بازمی‌گشتند تا کدها را اصطلاح کنند یا ادامه آن‌ها را بنویسند. در سال ۱۳۶۲ شمسی (۱۹۸۳ میلادی) شخصی به نام «Borland Ltd» یک کامپایلر زبان پاسکال را به نام «TurboPascal» ارائه کرد که برای اولین بار ابزاری دارای ویرایشگر یکپارچه متن و کامپایلر بود TurboPascal .ممکن است اولین ایده ساخت محیط‌های یکپارچه توسعه باشد.

اما بسیاری از متخصصین بر این باور هستند که در واقع ابزار «ویژوال بیسیک[۱۸] که توسط شرکت مایکروسافت در سال ۱۳۷۰ شمسی (۱۹۹۱ میلادی) ساخته شد، اولین IDE واقعی تاریخ بود.

---

[۱۶] Real Time

[۱۷] Syntax Highlighting

[۱۸] Visual Basic | VB

ویژوال بیسک به سفارش شرکت زبان برنامه نویسی قدیمی «Basic» ساخته شده بود. این زبان یکی از زبان‌های محبوب دهه ۶۰ شمسی (۱۹۸۰ میلادی) به حساب می‌آید. ظهور ویژوال بیسیک با این هدف بود که می‌توان برنامه نویسی را به صورت گرافیکی نیز بررسی کرد و مزایای افزایش بهره‌وری نیز در آن به صورت قابل توجهی مشهود بودند.

۶- ۴- ویژگی های رایج IDE ها

IDEیکی از ابزارهایی در برنامه نویسی به حساب می‌آید که ده‌ها سال است مورد استفاده قرار می‌گیرد. این پلتفرم قابلیلت‌های بسیاری در برنامه نویسی، اشکال‌زدایی و توسعه کدها دارد و به مرور زمان نیز در حال تغییر و تکامل است. در این بخش برخی از ویژگی‌های اصلی و استانداردی فهرست شده‌اند که در اکثر محیط‌های توسعه یکپارچه وجود دارند:

- **ویرایشگر متن:** می‌توان گفت به‌طور معمول همهIDE ها دارای یک ویرایشگر متن هستند که برای نوشتن و دستکاری کدهای منبع طراحی شده است. برخی از آن‌ها امکان دارد که مؤلفه‌های بصری برای «کشیدن و انداختن (Drag And Drop) «اجزای «فرانت‌اند» خود داشته باشند، اما اکثر آن‌ها یک واسط ساده دارند که سینتکس‌های خاص زبان برنامه نویسی را برجسته یا هایلایت می‌کنند. در ادامه، کدهایی با زبان برنامه نویسی جاوا به صورت برجسته شده و برجسته نشده ارائه شده‌اند، ابتدا کدهای برجسته نشده نمایش داده می‌شوند:

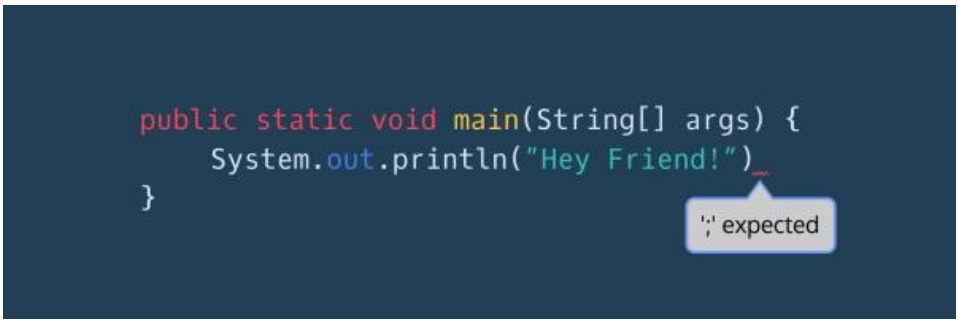
```
// without syntax highlighting
public class NiceDay {
  public static void main(String[] args) {
    System.out.println("It's a nice day out!");
  }
}
```

حال کدهای فوق به صورت هایلایت شده در تصویر زیر نمایش داده شده‌اند:

```java
// with syntax highlighting

public class NiceDay {
  public static void main(String[] args) {
    System.out.println("It's a nice day out!");
  }
}
```

- **خطایاب یا دیباگر**[۱۹]: IDE ها دارای ابزارهای اشکال‌زدایی هم هستند که به برنامه نویسان امکان شناسایی و رفع خطاهای کدهای منبع را می‌دهند. این ابزارها معمولاً سناریوهای دنیای واقعی را برای تست عملکرد و کارایی کدها شبیه‌سازی می‌کنند. برنامه نویسان و مهندسان نرم افزار می‌توانند با استفاده ازIDE ها بخش‌های مختلف کدها را آزمایش و خطاهای آن‌ها را قبل از انتشار برنامه، شناسایی کنند. در ادامه کدهایی با زبان جاوا نمایش داده می‌شوند که دارای خطای نحوی هستند و IDE به آن‌ها پیشنهادی برای رفع خطا داده است:



- **کامپایلر:** یکی از ویژگی‌هایی کهIDE ها دارند، روش‌های کامپایل کدها است. کامپایلرها مولفه‌هایی هستند که زبان‌های برنامه نویسی سطح بالا را به زبانی دودویی، سطح پایین و قابل درک برای ماشین تبدیل می‌کنند. کدهای ماشین برای اطمینان از صحت آن‌ها

[۱۹] Debugger

تجزیه و تحلیل می‌شوند. سپس، کامپایلر کدها را برای بهینه‌سازی عملکرد، تجزیه و بررسی می‌کند.

- تکمیل کد خودکار[۲۰] : قابلیت تکمیل کد درIDE ها به وسیله شناسایی هوشمند با درج اجزای کدهای رایج در برنامه‌ها به برنامه نویسان کمک می‌کنند. این ویژگی‌ها باعث صرفه‌جویی در زمان نوشتن کدها و کاهش احتمال بروز خطاهای نوشتاری می‌شوند. همچنین ممکن است وظایف دیگری نیز وجود داشته باشند که به صورت خودکار توسط IDEها انجام شوند.
- **پشتیبانی زبان برنامه نویسی:** IDEهای مختلفی وجود دارند که برخی از آن‌ها مختص به یک زبان برنامه نویسی خاص و برخی دیگر توسط چندین زبان مورد استفاده قرار می‌گیرند. به همین ترتیب، برای انتخابIDE ، نیاز است، آن‌هایی انتخاب شوند که در زبان برنامه نویسی مورد نظر کاربرد دارند. به عنوان مثال ابزارهای IDE مختلفی برای زبان‌های برنامه نویسی جاوا، پایتون و روبی (Ruby) وجود دارند.
- **یکپارچه‌سازی و افزونه‌ها**[۲۱] : همان‌طور که IDE مخفف عبارت «محیط توسعه یکپارچه» است، شکی نیست که قابلیت یکپارچه‌سازی در این ابزار وجود داشته باشد. به عبارتی می‌توان گفت که IDE همان پرتال توسعه برنامه است، بنابراین ترکیب و یکپارچه‌سازی همه ابزارهای توسعه مورد نیاز برنامه، بهره‌وری را بهبود می‌بخشد. همچنین، یکپارچه‌سازی ضعیف باعث بروز مشکلات متعددی می‌شود و بهتر است از IDE استفاده شود که این ویژگی را به خوبی رعایت کرده است.

---

[۲۰] Code Completion

[۲۱] Integrations And Plugin

۶- ۵- تفاوت های بین ابزارهای محیط توسعه یکپارچه

IDEها انواع مختلفی دارند و با یکدیگر متفاوت هستند. در این بخش به طور خلاصه به نوع تفاوت هر کدام از IDE ها پرداخته شده است. این تفاوت‌ها در ادامه فهرست شده‌اند:

- **تعداد زبان‌های برنامه نویسی پشتیبانی شده توسط IDE** : برخی از IDE ها فقط مختص به یک زبان برنامه نویسی خاص هستند، بنابراین تطابق بهتری با الگوهای آن زبان دارند. برخی از IDE ها نیز در همه زبان‌های برنامه نویسی یا مجموعه‌ای از آن‌ها دارای کاربرد هستند.
- **نوع سیستم عامل پشتیبانی شده توسط هر IDE** : این نوع از ابزارهای محیط توسعه یکپارچه از سیستم عامل‌های خاص و مخصوص به خود پشتیبانی می‌کنند. مثلا ممکن است یک ابزار محیط توسعه یکپارچه فقط از سیستم عامل IOS و اندروید پشتیبانی کند. فقط IDE های ابری هستند که نوع سیستم عامل برای آن‌ها اهمیتی ندارد.
- **ویژگی‌های خودکارسازی IDE ها**: بیشتر IDE ها شامل سه ویژگی کلیدی ویرایشگر متن، ساختن به صورت خودکار و اشکال‌زدایی می‌شوند. همچنین بسیاری از آن‌ها از ویژگی‌های کلیدی دیگری از جمله بازساخت یا سازماندهی مجدد[۲۲]، جستجوی کدها و ابزارهای یکپارچه سازی و استقرار پیوسته[۲۳] پشتیبانی می‌کنند.
- **تأثیر IDE بر روی کارایی سیستم**: اگر توسعه دهنده‌ای بخواهد به صورت همزمان از چند برنامه کاربردی استفاده کند، میزان حافظه‌ای که IDE اشغال می‌کند بر روی میزان کارایی برنامه‌های سیستم تاثیر می‌گذارد.

---

[۲۲] Refactoring

[۲۳] Continuous Integration And Continuous Deployment | CI/CD

- **انواع افزرونه‌ها و پلاگین‌های هرIDE** : برخی ازIDE ها شامل قابلیت سفارشی کردن گردش کار برای برطرف کردن نیازها و اولویت‌های یک توسعه دهنده به وسیله افزونه یا همان اکستنشن هستند.

۶- ۶- انواع IDE ها

معمولاً برای همه زبان‌های برنامه نویسی موجود، IDEوجود دارد. هر کدام ویژگی‌های متفاوتی را ارائه می‌دهند که به برنامه نویسان امکان ایجاد کدهایی با کیفیت بالا، سرعت و کارایی مناسب را خواهند داد. در ادامه برخی از انواعIDE ها فهرست شده‌اند:

۱) **IDEهای چندزبانی**[۲۴]: این نوع ابزارها از بیش از یک زبان برنامه نویسی پشتیبانی می‌کنند. به عنوان مثال ویژوال استودیو یک ابزار محیط توسعه یکپارچه چندزبانی به حساب می‌آید که به عنوان یکی از بهترینIDE ها به دلیل داشتن ویژگی‌های باورنکردنی و پشتیبانی مداوم از افزونه‌های مختلف شناخته می‌شود. می‌توان زبان‌های برنامه نویسی جدیدی را که به صورت پیش‌فرض در IDE ویژوال استودیو وجود ندارند، با استفاده از افزونه به این IDE اضافه کرد.

۲) **IDEهای توسعه موبایل:** IDEهای بسیاری برای توسعه زبان‌های برنامه نویسی موبایل وجود دارند و با گسترش بازار توسعه اپلیکیشن‌های موبایل روزبه‌روز در حال افزایش هستند. توسعه دهندگان اپلیکیشن‌های موبایل پلتفرم‌هایی را می‌خواهند که بر این نوع توسعه متمرکز باشند تا برنامه‌های موثر و کارآمدی ایجاد کنند. به عنوان مثال، «Android Studio» و «Xcode» ابزارهای محیط توسعه یکپارچه‌ای هستند که برای توسعه پلتفرم‌های سیستم عامل‌های «Android» و «IOS» مورد استفاده قرار می‌گیرند.

[۲۴] Multi-Language IDE

۳) IDEهای مبتنی بر وب و فضای ابری[۲۵] : IDE های مبتنی بر وب و فضاهای ابری در مقایسه با ابزارهای محیط توسعه یکپارچه Local دارای ویژگی‌های منحصر به فرد بسیاری هستند. برای مثال محیط توسعه یکپارچه «SaaS» می‌تواند وظایفی که نیازمند زمان زیادی هستند را بدون در نظر گرفتن منابع محاسباتی یک ابزار Local یا همان محلی پیاده‌سازی کند. IDEهای ابری اغلب مستقل از پلتفرم[۲۶] هستند و امکان اتصال به چندین ایجاد کننده ابری را فراهم می‌کنند.

۴) IDEزبان‌های برنامه نویسی خاص: این نوع از ابزارهای محیط توسعه یکپارچه، فقط برای استفاده در یک زبان برنامه نویسی خاص ایجاد شده‌اند. برای مثال، IDEهای «Jikes» و «Jcreator»برای زبان جاوا، IDEهای «CodeLite» و «C-Free» برای زبان‌های C و C++ و «Idle»برای زبان پایتون توسعه یافته‌اند و فقط مختص به زبان برنامه نویسی مخصوص به خود هستند.

۶- ۷- IDEچندزبانی و بهترینIDE های چندزبانی

این نوع از ابزارهای محیط توسعه یکپارچه از چندین زبان برنامه نویسی پشتیبانی می‌کنند، در ادامه برخی از آن‌ها به همراه شرح مختصری فهرست شده‌اند:

- **Eclipse**: این ابزار محیط توسعه یکپارچه از زبان‌های برنامه نویسیC ، پرل(Perl) ، C++، پایتون، PHP، جاوا، روبی و سایر زبان‌های برنامه نویسی پشتیبانی می‌کند. این IDE رایگان و ویرایشگری متن باز برای بسیاری از فریم ورک‌های توسعه به حساب می‌آید. با این حال در ابتدا این IDE فقط برای زبان برنامه نویسی جاوا مورد استفاده قرار می‌گرفت، اما از طریق افزونه‌های خود، گسترش یافته است.

[۲۵] Web | Cloud-Based IDE
[۲۶] Platform-Independent

- **NetBeans**: این IDE از زبان‌های برنامه نویسیPHP ، جاوا، C، جاوا اسکریپت (JavaScript)، C++، پایتون، روبی و بسیاری از زبان‌های دیگر پشتیبانی می‌کند. این محیط توسعه یکپارچه، به صورت رایگان و منبع باز در دسترس همه است. بسیاری از توابع NetBeans با استفاده از ماژول‌ها در IDE ایجاد می‌شوند. توسعه دهندگان با استفاده از نصب ماژول‌های مخصوص به هر زبان برنامه نویسی می‌توان از آن زبان در NetBeansاستفاده کنند.
- **Komodo IDE**: این ابزار محیط توسعه یکپارچه از زبان‌های برنامه نویسیPHP ، پرل، Tcl، پایتون، جاوا اسکریپت، روبی و سایر موارد پشتیبانی می‌کند. با هزینه بالاتر می‌توان به این ابزار در سطح سازمانی نیز دسترسی داشت.
- **Aptana**: این IDE از زبان‌هایHTML ، جاوا اسکریپت، CSS، AJAX و برخی زبان‌های دیگر با استفاده از پلاگین‌ها پشتیبانی می‌کند. این ابزار محیط توسعه یکپارچه انتخابی مناسب و پرطرفدار برای کسانی است که در حوزه توسعه اپلیکیشن‌های وب کار می‌کنند.
- **Geany**: این ابزار محیط توسعه یکپارچه از زبان‌هایC ، جاوا، PHP، پرل، HTML، پایتون، پاسکال و بسیاری زبان‌های دیگر پشتیبانی می‌کند. این IDE محیطی با قابلیت سفارشی سازی[۲۷] بالایی همراه با مجموعه بزرگی از پلاگین‌ها دارد.

۶- ۸- بهترین IDE برای پایتون

اغلبIDE ها زبان‌های برنامه‌نویسی متعددی را پشتیبانی می‌کنند و از قابلیت‌های بسیار زیادی برخوردارند. به همین دلیل است که معمولاً حجم آن‌ها بالاست و دانلود و نصبشان زمان زیادی را می‌برد. علاوه بر این، برای کارکردن با یک IDE باید دانش خاص آن را داشته باشید.

[۲۷] Customizable

در سمت مقابل، ویرایشگرهای کد هستند که کارهای ساده‌ای مانند برجسته‌کردن سینتکس و قالب‌بندی کدها را انجام می‌دهند. یک ویرایشگر کد خوب علاوه بر اجرای کدها، یک دیباگر را نیز به همراه خود دارد. اما یک ویرایشگر کد عالی می‌تواند با سیستم‌های کنترل منبع[۲۸] تعامل داشته باشد. به‌طور کلی اگرچه ویرایشگرهای کد در مقایسه باIDE ها حجم و سرعت پایین‌تری دارند، اما قابلیت‌های آن‌ها نیز کمتر است.

۶- ۸- ۱- ویژگی‌های بهترین IDE های پایتون

خواسته‌هایی که از یک محیط توسعه‌ی خوب داریم از برنامه‌ای تا برنامه‌ی دیگر متفاوت است. اما به‌طور کلی می‌توان به مجموعه‌ای از ویژگی‌ها اشاره کرد که کار کدنویسی را آسان‌تر می‌کنند. این ویژگی‌ها را در ادامه می‌خوانید:

- **ذخیره و بارگذاری فایل کدها :** یک IDE باید امکان ذخیره‌ی کدها را برای شما فراهم کند تا زمانی که مجدداً آن را باز می‌کنید، بتوانید به ادامه‌ی کارتان بپردازید.
- **اجرای کد درون محیط :** ویرایشگر شما نباید به‌گونه‌ای باشد که برای اجرای کدهایتان مجبور باشید تا از آن خارج شوید.
- **پشتیبانی دیباگ** : همه‌یIDE ها و اغلب ویرایشگرهای خوب امکان اصلاح‌کردن کدها را برای شما فراهم می‌کنند.
- **برجسته‌کردن سینتکس:** IDEها با برجسته‌کردن کلمات کلیدی و متغیرها و نمادها، خوانش و فهم کدها را آسان‌تر می‌کنند.
- **قالب‌بندی خودکار کدها:** هر IDE و ویرایشگر خوبی تشخیص می‌دهد که باید خط بعد از عبارت while یا for را از جلوتر شروع کند.

[۲۸] Source Control System

البتهIDE ها قابلیت‌های دیگری نیز دارند که از اهمیت زیادی برخوردارند، با این حال سعی کردیم تا فقط اصلی‌ترین ویژگی‌هایIDE ها را برشماریم.

کدنویسی پایتون با استفاده از IDLE یا شل پایتون، تنها برای اجرای وظایف ساده مناسب است. اما، به کار بردن چنین ابزارهایی در پروژه‌های برنامه‌نویسی بزرگ‌تر، مشکلات و چالش‌هایی را به وجود می‌آورد. با استفاده از یک محیط توسعه یکپارچه (IDE) یا حتی یک کد ادیتور (ویرایشگر کد) اختصاصی می‌توان تجربه کدنویسی را به میزان زیادی دلپذیرتر و آسان‌تر کرد.

می‌توان با استفاده از محیط شل (Shell) پروژه‌های کوچک پایتون را انجام داد. اما، در صورتی که قصد پیاده‌سازی پروژه‌های بزرگ‌تری با پایتون وجود داشته باشد، استفاده از یک ویرایشگر کد اختصاصی یا یک محیط توسعه تلفیقی (IDE) انتخاب بهتری خواهد بود. هر IDE یا ویرایشگر کد برای پایتون با دیگری متفاوت است. این تفاوت‌ها در ویژگی‌ها، رابط کاربری (UI) و موارد دیگر نمود پیدا می‌کنند.

۶- ۸- ۲- بهترین IDE های پایتون

۱) Eclipse

اگر با جامعه‌ی منبع‌بازها آشنایی داشته باشید، حتما نام Eclipse به گوشتان خورده است. Eclipse یک IDE منبع‌باز برای توسعه‌ی جاواست که برای سیستم‌عامل‌های Linux ، Windowsو OS X موجود است. وجود تعداد زیاد افزونه باعث شده تا Eclipse یک IDE عالی برای انجام طیف گسترده‌ای از کارها باشد. یکی از این افزونه‌هاPyDev است که امکانات بسیاری را در اختیار توسعه‌دهندگان قرار داده است. نمونه‌ای از این قابلیت‌ها دیباگ پایتون، تکمیل کدها و وجود کنسول تعاملی پایتون است.

۲) Visual Studio

Visual Studioیک IDE با تمامی قابلیت‌های موردنیاز است که توسط شرکت Microsoft ساخته شده است Visual Studio . از جهات بسیاری شبیه به Eclipse است. این IDE که فقط در سیستم‌عامل‌های Windows و Mac OS اجرا می‌شود، هم نسخه‌های رایگان دارد و هم نسخه‌هایی که برای استفاده از آن‌ها باید هزینه‌ای را بپردازید VisualStudio .کار توسعه را برای طیف گسترده‌ای از پلتفرم‌ها امکان‌پذیر کرده است. برای این IDE افزونه‌های متعددی نیز وجود دارد.

ابزارهای پایتون برای Visual Studio که به آن‌ها PTVS نیز می‌گویند، امکان کدنویسی پایتون، Intellisense و دیباگ‌کردن را در Visual Studio به وجود آورده‌اند. اگر فقط به دنبال محیطی برای کدنویسی با زبان پایتون هستید، شاید IDE بزرگی مانند Visual Studio برای شما ضرورتی نداشته باشد. به‌علاوه، اگر سیستم‌عامل شما Linux باشد، به کلی نمی‌توانید از Visual Studio استفاده کنید.

۳) Visual Studio Code

Visual Studio Code که به آن VS Code نیز گفته می‌شود، یک ویرایشگر کد با قابلیت‌های تمام وکمال است که برای سیستم‌عامل‌های Linux ، Mac OS X و Windows قابل‌استفاده است. گفتنی است که به‌رغم شباهت اسمی‌شان، نباید Visual Studio Code را با Visual Studio اشتباه گرفت. Visual Studio Codeدر عین کم‌حجم بودن، قابلیت‌های فراوانی دارد. این ویرایشگر کد منبع‌باز و قابل‌توسعه، برای همه‌ی انواع تکالیف قابل‌پیکربندی است. VS Code نیز مانند Atom مبتنی بر Electronاست و از همین رو مزایا و معایبی مشابه با Atom دارد.

مزایا و معایب VS Code چیست؟

به این دلیل که VS Code بر مبنای Electron توسعه داده شده است، این ویرایشگر کد در همه پلتفرم‌ها در دسترس است VS Code .بر خلاف حجم کم، به طور شگفت‌انگیزی امکانات کاملی

دارد. برخی از مزایای VS Code در ادامه این بخش از مطلب بهترین IDE برای پایتون فهرست شده است:

- VS Codeدارای بیش از ۴۷۰۰ افزونه است.
- این ویرایشگر کد یک موتور مدیریت قدرتمند دارد.
- وارد کردن میانبرهای صفحه کلید از ویرایشگرهای پایتون، مثل Sublime Text یاAtom

اما، بر پایه Electron بودن VS Code به این معنا است که VS Code یک برنامه بومی (Native App) نیست. از جمله سایر معایب VS Code می‌توان به این موضوع اشاره کرد که برخی از افراد ممکن است دلایل اصولی خود را برای استفاده نکردن از منابع مایکروسافت داشته باشند. همچنین با توجه به تعداد زیاد افزونه‌ها، یافتن افزونه‌ای که به بهترین نحو با نیازهای فرد مطابقت داشته باشد، کار چندان ساده‌ای نیست.

۴) PyCharm

یکی از **بهترینIDE های پایتون** که قابلیت‌های تمام‌وکمالی دارد، PyCharm است. از PyCharm هم نسخه‌های رایگان و منبع‌باز و هم نسخه‌های پولی موجود است. این IDE را می‌توانید به‌راحتی و با سرعتی عالی روی انواع سیستم عامل‌های Windows ، Mac OS X و Linux نصب کنید.
PyCharmمستقیماً از توسعه‌ی پایتون پشتیبانی می‌کند. تنها کاری که نیاز است انجام دهید بازکردن یک فایل جدید و شروع کدنویسی است. می‌توانید در داخل PyCharm پایتون را اجرا و دیباگ کنید. این IDE از کنترل منبع پروژه‌ها نیز پشتیبانی می‌کند. محیط توسعه PyCharm در سه نسخه قابل دسترسی است:

۱) نسخه عمومی دارای گواهینامه آپاچی Apache-licensed Community Version

۲) نسخه آموزشی Educational (Edu) Version

۳) نسخه حرفه‌ای اختصاصی Proprietary Professional Version

دو نسخه اول متن‌باز و رایگان هستند، این در حالی است که، نسخه حرفه‌ای PyCharm باید خریداری شود. نسخه عمومی بسیار جالب توجه است، زیرا ویژگی‌های مختلفی را نظیر برجسته‌سازی سینتکس، تکمیل خودکار کدها و تایید لحظه‌ای اعتبار کدها دارد. نسخه پولی PyCharmدارای ویژگی‌های پیشرفته‌تری همچون مدیریت کامل پایگاه‌داده و تعداد زیادی از فریم‌ورک‌های مهم‌تر نسبت به نسخه عمومی است. برخی از فریم‌ورک‌هایی که در نسخه حرفه‌ای PyCharmدر دسترس هستند، شامل موارد زیر است:

- جنگو(Django)
- فلسک(Flask)
- گوگل اپ(Google App)
- انجین(Engine)
- پیرمید(Pyramid)
- وب‌توپی(web2py)

همچنین، PyCharmبه سرعت و به راحتی روی پلتفرم‌های ویندوز، مک OS و لینوکس نصب می‌شود PyCharm .به صورت پیش‌فرض و به طور مستقیم از پایتون پشتیبانی می‌کند. تنها با باز کردن یک فایل جدید می‌توان شروع به کدنویسی پایتون کرد. می‌توان پایتون را به طور مستقیم در داخل PyCharm اجرا و خطایابی کرد. همچنین، پشتیبانی برای کنترل منبع و پروژه نیز وجود دارد. در ادامه، به شرح نقاط ضعف و برتری PyCharm پرداخته شده است.

نقاط قوت و ضعف PyCharm چه هستند؟

PyCharmمحیط توسعه اختصاصی پایتون به حساب می‌آید که دارای پشتیبانی کامل و اجتماعی حمایتگر است. با استفاده از محیط توسعه PyCharm می‌توان ویرایش، اجرا و خطایابی پایتون را

بدون نیاز به نصب هیچ افزونه‌ای انجام داد. در یک جمع‌بندی، فهرستی از برخی نقاط برتری PyCharmدر ادامه این بخش از مقاله بهترین IDE برای پایتون، به صورت زیر است:

- پشتیبانی اجتماع فعالPyCharm
- تایید اعتبار کدها به صورت زنده و برجسته‌سازی (هایلایت) نحوی
- ویرایش‌ها و خطایابی کدهای پایتون را بدون هیچ وابستگی خارجی انجام می‌دهد.

از جمله کاستی‌های PyCharm می‌توان به موارد زیر اشاره کرد:

- کندی زمان بارگذاری
- ممکن است پیش از آنکه بتوان از پروژه‌های فعلی استفاده کرد، نیاز به تغییرات در تنظیمات پیش‌فرض وجود داشته باشد.

۵) Spyder

Spyder یک IDE منبع‌باز برای پایتون و ابزاری ایدئال برای استفاده در کارهای علوم داده است . Spyderتوزیع Anaconda را در خود دارد و بنابراین بسته به تنظیماتتان، ممکن است آن را روی دستگاه خود نصب داشته باشید. از آنجایی که جامعه‌ی هدف Spyder دانشمندان علوم داده‌ای است که از پایتون استفاده می‌کنند، Spyderبه‌خوبی با کتابخانه‌هایی مانند SciPy ، NumPy و Matplotlibیکپارچه شده است .

Spyder اکثر قابلیت‌های یک IDE معمول را داراست. از جمله‌ی این قابلیت‌ها می‌توان به برجسته‌کردن سینتکس، تکمیل کدهای پایتون و حتی یک مرورگر یکپارچه برای مستندات اشاره کرد. یکی از قابلیت‌های منحصربه‌فردSpyder ، جست‌وجوگر متغیر (Variable Explorer) است که داده‌های شما را در IDE به‌صورت جدول نمایش می‌دهد. اگر دائماً با تکالیف علوم داده‌ای سروکار دارید، این قابلیت به کارتان می‌آید. Spyder به‌صورت رایگان برای Windows ، macOSو Linux وجود دارد و کاملاً منبع‌باز است.

مزایا و معایب Spyder چیست؟

به طور کلی می‌توان گفت که Spyder نسبت به سایرIDE ها کمی ابتدایی‌تر است. می‌توان به جای یک محیط توسعه اصلی و روزمره، Spyderرا بیش‌تر یک ابزار با مورد استفاده خاص در نظر گرفت. نکته مثبت در مورد این IDE پایتون، در دسترس بودن آن به رایگان در ویندوز، مک OS و لینوکس و همچنین متن‌باز بودن آن است. ممکن است Spyder برای استفاده روزمره توسعه‌دهندگان باتجربه‌تر پایتون تا حدودی ساده و ابتدایی تلقی شود و این توسعه‌دهندگان یک IDE یا ویرایشگر شخصی‌سازی شده کامل‌تر را به جای آن برگزینند.

۶) Thonny

Thonnyعضو جدیدی از خانوادهIDE های پایتون به شمار می‌رود Thonny .یک IDE مناسب افراد مبتدی به حساب می‌آید. این محیط توسعه اختصاصی پایتون، توسط موسسه علوم کامپیوتر دانشگاه Tartu در استونی ارائه شده است Thonny .برای تمام پلتفرم‌های اصلی به همراه دستورالعمل نصب در دسترس است. به طور پیش‌فرض، Thonnyبه همراه نسخه همبسته‌سازی شده پایتون مختص به خودش نصب می‌شود، بنابراین، نیاز به نصب هیچ چیز دیگری وجود ندارد. کاربران باتجربه‌تر ممکن است نیاز داشته باشند تنظیمات اولیه را کمی تغییر دهند تا بتوانند کتابخانه‌های نصب شده را پیدا کرده و از آن‌ها استفاده کنند.

مزایا و معایب Thonny چه هستند؟

همان‌طور که بیان شد، یکی از نقاط برتری Thonny این است که برای کاربران تازه‌کار پایتون بسیار مناسب است. اما، توسعه‌دهندگان باتجربه‌تر، Thonnyرا بسیار ساده و ابتدایی خواهند یافت. یکی دیگر از نقاط ضعفThonny ، مفسر داخلی آن است که به جای کار کردن با آن بهتر است از یک جایگزین استفاده شود. همچنین، با توجه به اینکه Thonny ابزار جدیدی به حساب می‌آید، ممکن است مشکلاتی در حین کار به وجود آید که هنوز راهکار آنی برای آن ارائه نشده باشد.

IDLE (۷

محیط یکپارچه توسعه و یادگیری (IDLE | Integrated Development and Learning Environment)یک IDE پایتون است که در سال ۱۳۶۸ توسط گیدو ون‌راسوم (خالق پایتون) برای توسعه پایتون منتشر شد. IDLE یک IDE ساده و مناسب برای افراد مبتدی است IDLE .شامل یک ویرایشگر کد چندپنجره‌ای با امکانات زیر است:

- برجسته‌سازی نحوی
- خطایابی یکپارچه
- نقطه انفصال ماندگار
- رویت‌پذیری فراخوانی پشته

در ادامه این بخش از مطلب بهترین IDE برای پایتون ، برخی از نقاط مثبت و منفی IDLE شرح داده شده‌اند.

نقاط مثبت و منفیIDLE

برخی مزایای IDLE به شرح زیر است:

- از IDLE می‌توان برای اجرای یک گزاره تنها استفاده کرد.
- می‌توان از IDLE برای ایجاد، ویرایش و اجرای اسکریپت‌های پایتون استفاده کرد.
- ویژگی‌هایی مانند برجسته‌سازی نحوی، تکمیل خودکار کدها و دندانه‌گذاری داخلی هوشمند توسط IDLE ارائه می‌شود.
- IDLEدارای یک خطایاب به همراه ویژگی‌های گام‌برداری و نقطه انفصال است.

برخی از معایب IDLE در ادامه این بخش از نوشته بهترین IDLE برای پایتون فهرست شده‌اند:

- IDLEبه صورت پیش‌فرض در توزیع پایتون برای لینوکس در دسترس نیست.
- IDLEبه یک مدیر بسته خاص برای نصب نیاز دارد.

انتخاب بهترین IDE برای پایتون بسته به نظر هر شخص می‌تواند تفاوت باشد، اما برای انتخاب بهتر، در ادامه توصیه‌هایی پیرامون گزینش بهترین IDE برای پایتون فهرست شده‌اند.

- توسعه‌دهندگان تازه‌کار پایتون باید از محصولات و ابزارهایی استفاده کنند که نیاز به شخصی‌سازی و تغییرات اندکی دارند.
- در صورتی که فردی برای کارهای دیگری همچون صفحات وب و مستندسازی از ویرایشگر متن استفاده می‌کند، بهتر است به دنبال ابزارهای ویرایش کد باشد.
- در صورتی که فردی پیش از نیاز به توسعه با پایتون، در حال توسعه نرم‌افزارهایی با زبان‌های دیگر باشد، ممکن است افزودن قابلیت‌های پایتون به مجموعه ابزارهای در حال استفاده انتخاب بهتری باشد.

پایتون یکی از شناخته شده‌ترین و محبوب‌ترین زبان‌های برنامه‌نویسی به حساب می‌آید. درست مثل سایر زبان‌های برنامه‌نویسی شاخص، تعداد زیادی ازIDE های کارآمد، کاربردی و قدرتمند به صورت رایگان و غیر رایگان در دسترس قرار دارد. در این بخش، تعدادی از اینIDE ها و ویرایشگرهای کد که نسبت به سایرین امکانات و قابلیت‌های بیش‌تر و عملکرد بهتری دارند با هدف انتخاب بهترین IDE برای پایتون معرفی شدند. شاید بتوان در یک جمع‌بندی و نتیجه‌گیری، از PyCharmبه عنوان بهترین IDE برای پایتون نام برد. اما همان‌طور که بیان شد، انتخاب بهترین IDEبرای پایتون بسته به مورد استفاده، سطح توانایی توسعه‌دهنده، پلتفرم مربوطه، پروژه مورد نظر و سایر موارد متفاوت است و نمی‌توان به طور عمومی یک IDE خاص را به عنوان بهترین IDE برای پایتون معرفی کرد.

منابع

https://www.python.org/

https://scipy.org/

https://numpy.org/

https://www.tensorflow.org/

https://pytorch.org/

https://pandas.pydata.org/

https://matplotlib.org/

https://keras.io/

https://github.com/Theano/Theano

https://blog.faradars.org/

https://www.w3schools.com/python/

https://www.tutorialspoint.com/python/index.htm