فصل اول

«اصول طراحی زبان‌های برنامه‌سازی»

مقدمه

در تعریف زبان برنامه‌سازی، یک توافق عمومی وجود ندارد. شاید بتوان گفت: یک زبان برنامه‌سازی، وسیله ارتباط بین برنامه‌نویسان است. یک توانایی برای بیان طراحی‌های سطح بالا است. یک روش بیان ارتباطات، بین مفاهیم است. «پرات» (Pratt) در کتاب خود، زبان برنامه‌نویسی را این چنین تعریف می‌کند: «یک زبان برنامه‌سازی، یک روش نشانه‌گذاری برای توصیف الگوریتم‌ها و ساختمان داده‌ها است».

تاکنون زبان‌های برنامه‌سازی بسیار زیادی طراحی و پیاده‌سازی شده‌اند. بدیهی است دلایل و فواید زیادی برای مطالعه زبان‌های برنامه‌سازی وجود دارد. در اینجا به شرح برخی از این دلایل عمده می‌پردازیم:

دلایل مطالعه زبان‌های برنامه‌سازی

بدیهی است که هدف از مطالعه زبان‌های برنامه‌سازی را بدانیم. مهم‌ترین دلایل و فواید مطالعه زبان‌های برنامه‌سازی عبارتند از:

۱ـ **افزایش توانایی خود در توسعه الگوریتم‌های کارآمد:** بسیاری از زبان‌ها ویژگی‌هایی دارند که اگر به خوبی مورد استفاده قرار گیرند، به نفع برنامه‌نویس است و اگر به طور نامناسب مورد استفاده قرار گیرند موجب اتلاف وقت برنامه‌نویس می‌گردد. آگاهی از اصول و تکنیک‌های پیاده‌سازی ویژگی‌های زبان این امکان را به برنامه‌نویس می‌دهد که در وضعیت‌های خاص، تصمیم‌گیری مناسب را انجام دهد.

۲ـ **استفاده بهینه از زبان برنامه‌نویسی موجود:** با فراگیری پیاده‌سازی ویژگی‌های یک زبان، توانایی شما در نوشتن برنامه‌های کارآمد افزایش می‌یابد.

۳ـ **آشنایی با اصطلاحات مفید ساختارهای برنامه‌نویسی:** اگر برنامه‌نویس فقط با یک زبان برنامه‌سازی آشنایی داشته باشد، محدودیت برایش ایجاد می‌شود. ممکن است این نکته بیان شود که اگر یک برنامه‌نویس مجبور باشد از زبان خاصی برای تولید نرم‌افزار استفاده کند، یادگیری قابلیت‌های سایر زبان‌ها نمی‌تواند به او کمک کند. اما توجه داشته باشید که اگر این برنامه‌نویس نیاز به ساختاری داشته باشد که در زبان مورد استفاده‌اش موجود نباشد می‌تواند ساختارهای موجود در زبان‌های دیگر را در این زبان شبیه‌سازی کند.

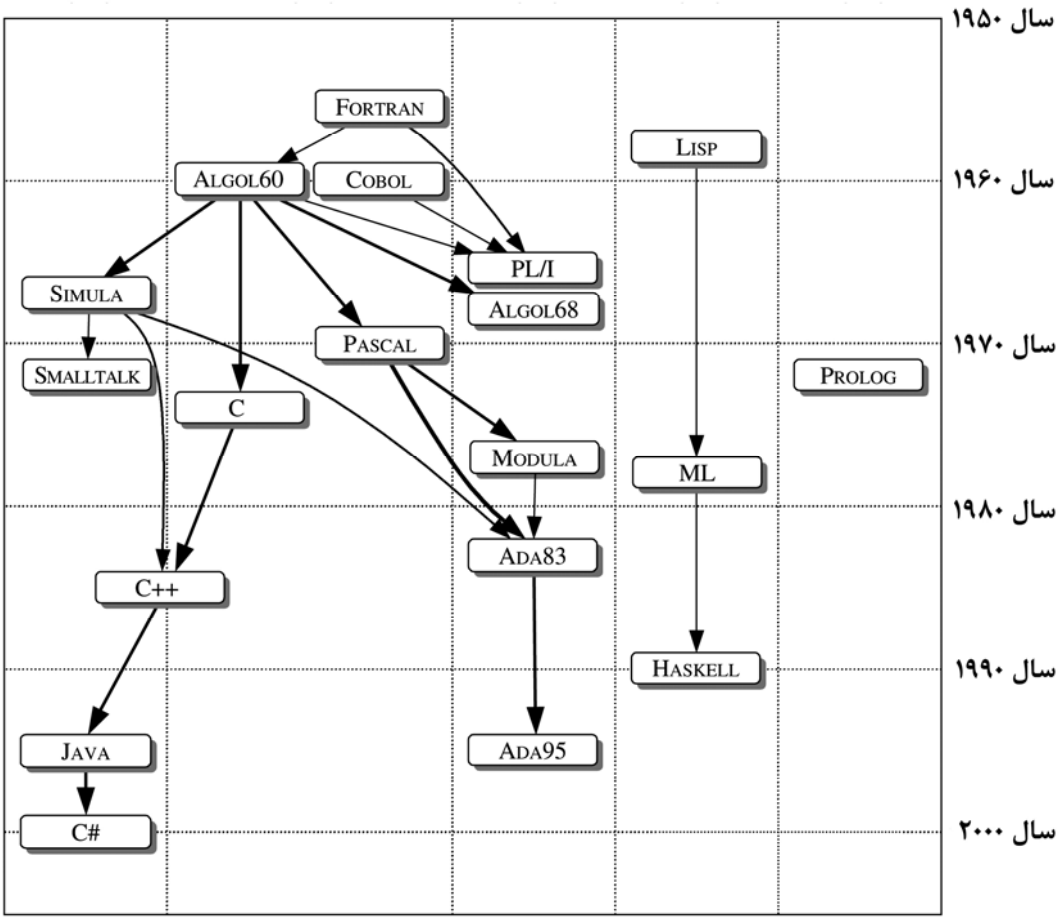
۴ـ **انتخاب بهترین زبان برنامه‌سازی:** اغلب برنامه‌نویسان با آن زبان برنامه‌سازی آشنایی دارند که در راستای پروژه‌هایی است که در آن سازمان وجود دارد. به همین دلیل، برنامه‌نویسان معمولاً با زبان‌هایی برنامه‌نویسی می‌کنند که با آن‌ها آشنایی بیشتری دارند، حتی اگر امکانات آن زبان، برای آن کاربرد مناسب نباشد. اگر برنامه‌نویسان با زبان‌های دیگری آشنا باشند، می‌توانند بهترین زبان را انتخاب کنند.

۵ـ**تسهیل در یادگیری زبان جدید:** زبان‌های برنامه‌سازی به طور پیوسته در حال رشد هستند. فرآیند یادگیری زبان برنامه‌سازی جدید به ویژه برای کسانی که تنها با یک یا دو زبان آشنایی دارند و مفاهیم زبان‌های برنامه‌سازی را مطالعه نکرده‌اند، می‌تواند طولانی و دشوار باشد. به عنوان نمونه، کسانی که با مفاهیم برنامه‌نویسی شیء گرا آشنایی دارند، نسبت به کسانی‌که هیچ آشنایی با این مفاهیم ندارند، زبان جاوا را آسان‌تر می‌آموزند.

۶ـ**تسهیل در طراحی زبان جدید:** بعضی از برنامه‌نویسان، خودشان را طراح زبان می‌دانند. هر برنامه دارای یک واسط کاربر است که مانند یک زبان برنامه‌سازی عمل می‌کند. طراح واسط کاربر برای برنامه بزرگی مانند سیستم‌عامل باید آشنا به مفاهیم موجود در طراحی زبان‌ها باشد. این جنبه از طراحی برنامه، با تسلط برنامه‌نویس در ساختارها و روش‌های پیاده‌سازی زبان‌های برنامه‌سازی، تسهیل خواهد شد.

تاریخچه مختصری از پیدایش و طراحی زبان‌های برنامه‌سازی

زبان‌های برنامه‌سازی امروزی، نتیجه تکامل و پیشرفتی است که در دهه ۱۹۵۰ آغاز گردید. شکل زیر خلاصه‌ای از تاریخ پیدایش و طراحی چندین زبان برنامه‌نویسی مهم را نشان می‌دهد.



« شکل / . «تاریخچه پیدایش و طراحی چند زبان برنامه‌نویسی مهم »

دامنه کاربرد زبان‌های برنامه‌سازی

کامپیوترها در گستره وسیعی از کاربردها نقش دارند و به همین دلیل تنوع کاربرد کامپیوترها، زبان‌های برنامه‌سازی با اهداف مختلفی عرضه شده‌اند. در این بخش برخی از موارد کاربرد را بررسی می‌کنیم:

ـ کاربردهای علمی و مهندسی:

اولین کامپیوترهای دیجیتال برای کاربردهای علمی اختراع شدند و معمولاً کاربردهای علمی ساختمان داده‌های ساده‌ای دارند که متداول‌ترین آنها آرایه‌ها و ماتریس‌ها هستند. زبان‌های برنامه‌سازی سطح بالایی برای کاربردهای علمی طراحی شده‌اند، اولین زبان علمی، Fortran بود. ALGOL 60 نیز به همین منظور طراحی شد ولی کاربردهای دیگری نیز داشت.

ـ کابردهای تجاری:

کاربرد کامپیوترها در تجارت در دهه ۱۹۵۰ شروع شد. زبان‌های خاصی برای این منظور ایجاد شدند. اولین زبان سطح بالای موفق در این زمینه، کوبول (COBOL) بود که نسخه اولیه آن در سال ۱۹۶۰ ارائه گردید و هنوز هم این کاربرد استفاده گسترده‌ای دارد. با اختراع ریز کامپیوترها، روش‌های جدیدی برای کاربرد کامپیوترها در تجارت مطرح شد، استفاده از ابزارهای مفیدی مانند سیستم‌های بانک اطلاعاتی در تجارت رایج شد. زبان SQL که در دهه ۱۹۸۰ مطرح شد، برای استفاده از بانک‌های اطلاعاتی به‌کار می‌رود.

ـ کاربردهای آموزشی:

در دهه‌های ۱۹۶۰ و ۱۹۷۰ زبان‌هایی به منظور آموزش برنامه‌نویسی به دانشجویان طراحی شدند، به عنوان نمونه، زبان بیسیک در دهه ۱۹۶۰ طراحی شد. زبان‌های پاسکال و الگول نیز در دهه ۱۹۷۰ برای اهداف آموزش مطرح شدند. بعد از آن زبان‌های دیگری مانند C و C++ و جاوا نیز برای آموزش مورد استفاده قرار گرفتند.

ـ هوش مصنوعی:

هوش مصنوعی (Artificial Intelligence) یکی از پرکاربردترین جنبه‌های استفاده از کامپیوتر است که به جای محاسبات عددی با محاسبات نمادی (Symbolic) سروکار دارد. معنای محاسبات نمادی این است که نمادها که شامل اسامی هستند به جای اعداد دستکاری می‌شوند. این نوع برنامه‌نویسی نسبت به سایر دامنه‌های برنامه‌نویسی، نیاز به قابلیت انعطاف‌پذیری بیشتری دارد. اولین زبان هوش مصنوعی، لیسپ (Lisp) بود که در سال ۱۹۵۹ ایجاد شد. در دهه ۱۹۷۰، روش دیگری برای این کاربردها مطرح شد که برنامه‌نویسی منطقی با استفاده از زبان پرولوگ (Prolog) بوده است.

تذکر۱: زبان Scheme لهجه‌ای از زبان (Lisp) است.

ـ برنامه‌نویسی سیستم و شبکه:

سیستم عامل و تمام ابزارهای پشتیبان برنامه‌نویسی در یک کامپیوتر، به عنوان نرم‌افزارهای سیستم محسوب می‌شوند. در دهه‌های ۱۹۶۰ و ۱۹۷۰، بعضی از سازندگان کامپیوتر، زبان‌های سطح بالای مبتنی بر ماشین را برای نرم‌افزارهای سیستم در ماشین خود، طراحی کردند. شرکت IBM زبان PL/S، شرکت Digital زبان BLISS و شرکت Burroughs زبان ALGOL را به وجود آوردند. #C از زبان‌های جدیدی است که در برنامه‌نویسی شبکه به طور وسیعی مورد استفاده قرار می‌گیرد.

ـ نرم‌افزار وب :

تعدادی از زبان‌ها از جمله زبان‌های علامت‌دار (Murk up) مانند XHTML که زبان برنامه‌نویسی نیستند و یا زبان‌های برنامه‌نویسی همه منظوره مانند جاوا، از وب جهانی پشتیبانی می‌کنند. به دلیل نیاز به محتویات پویای وب، قابلیت‌های محاسباتی در فناوری نمایش محتویات، گنجانده شده‌اند. این قابلیت به این صورت فراهم می‌شود که کُد برنامه‌نویسی در سند XHTML قرار می‌گیرد. این کُد معمولاً به شکل زبان اسکریپتی مانند جاوا اسکریپت یا pHp است. علاوه بر این، سند XHTML می‌تواند اجرای برنامه جداگانه‌ای را در سرویس دهنده وب درخواست کند تا محتویات پویا را فراهم نماید.

مثال۱: کدامیک از دسته زبان‌های زیر در زمینه هوش مصنوعی بیشتر مورد استفاده قرار می‌گیرند؟

۱) Lisp , Pascal ۲) Lisp , Prolog ۳) C++ , Prolog ۴) C++ , Pascal

☑ **پاسخ:** گزینه «۲» زبان Lisp اولین زبان هوش مصنوعی بود و زبان Prolog نیز برای این کاربردها مطرح شد.

تأثیر محیط‌های عملیاتی

محیط خارجی که اجرای برنامه را پشتیبانی می‌کند «محیط عملیاتی» یا «محیط مقصد» نامیده می‌شود و محیطی که برنامه در آن طراحی، آزمون و اشکال‌زدایی می‌شود، «محیط میزبان» نام دارد.

محیط‌های عملیاتی که زبان‌های برنامه‌سازی در آنها به‌کار گرفته می‌شوند عبارتند از:

ـ محیط دسته‌ای (Batch): اولین و ساده‌ترین محیط عملیاتی است. محیط عملیاتی که در آن داده‌های ورودی در فایل‌ها دسته‌بندی شوند و به صورت دسته‌ای توسط برنامه پردازش شوند را پردازش دسته‌ای می‌گویند.

ـ محیط محاوره‌ای (Intractive): از اولین کامپیوترها در دهه ۱۹۴۰ تا دهه ۱۹۷۰ میلادی، کامپیوترهای بزرگ رایج بوده‌اند و در اواخر این دوره، برنامه‌نویسی محاوره‌ای پدید آمد. به جای اینکه برنامه بر روی دسته‌ای از کارت‌ها نوشته شود، از پایانه‌های لامپ اشعه کاتدی استفاده شده است که مستقیماً به کامپیوتر وصل شده‌اند. براساس تحقیقی که در دهه ۱۹۶۰ به عمل آمد، کامپیوتر قادر بود به صورت **اشتراک زمانی** مورد استفاده قرار گیرد، به طوری که هر کاربرد، پردازنده‌ی کامپیوتر را بازه‌ای از زمان در اختیار داشته باشد. در این محیط، یک برنامه مستقیماً با کاربر تعامل دارد و خروجی را در نمایشگر نشان می‌دهد و کاربر ورودی جدید را از طریق دستگاه‌های ورودی وارد می‌کند.

ـ محیط سیستم تعبیه شده (Embeded System): به سیستم کامپیوتری که برای کنترل بخشی از یک سیستم بزرگ به کار می‌رود، «سیستم کامپیوتری تعبیه شده» می‌گویند. در این سیستم‌ها، خرابی سیستم کامپیوتری از اهمیت بسزایی برخوردار است و هزینه زیادی را به دنبال خواهد داشت. به عنوان نمونه، در یک هواپیما، ممکن است خرابی سیستم تعبیه شده، زیان‌های جبران‌ناپذیری را به بار آورد. زبان‌های C, Ada , C++ برای نوشتن برنامه‌های تعبیه شده مفید خواهند بود.

محیط‌های عملیاتی	ویژگی‌ها (تأثیر بر طراحی زبان‌ها)
دسته‌ای	- خاتمه برنامه در صورت بروز خطا، قابل قبول ولی هزینه‌بر است. - پردازش خطا و استثناء مناسب است. - عدم وجود محدودیت زمانی بر روی برنامه (عدم ارائه امکاناتی برای تأثیر بر روی سرعت اجرای برنامه)
محاوره‌ای	- خاتمه برنامه در صورت بروز خطا، قابل قبول نیست. - لزوم برخورداری از محدوده‌های زمانی
تعبیه شده	- برنامه‌هایی که جهت سیستم‌های تعبیه شده نوشته شده‌اند، معمولاً بدون سیستم عامل مربوطه و بدون محیط معمولی فایل‌ها و دستگاه‌های ورودی/ خروجی اجرا می‌شوند. - اهمیت ویژه برای پردازش خطا - پردازش به‌صورت بلادرنگ - قابلیت اعتماد و صحت (درستی) به عنوان ویژگی مهم برای برنامه‌های تعبیه شده

مثال۲: کدام گزینه صحیح است؟

۱) در محیط پردازش دسته‌ای، خطایی که منجر به خاتمه برنامه شود قابل قبول نیست.

۲) تطبیق زبان‌های طراحی‌شده برای محیط دسته‌ای به محیط محاوره‌ای به سادگی امکان‌پذیر است.

۳) برنامه‌های محاوره‌ای باید از محدودیت زمانی برخوردار باشند.

۴) در محیط پردازش دسته‌ای، یک برنامه مستقیماً با کاربر تعامل دارد.

☑ **پاسخ:** گزینه «۳» همانطور که در جدول بالا ذکر شده است، در محیط محاوره‌ای لزوم برخورداری از محدوده‌های زمانی وجود دارد. گزینه ۱ نادرست است چون مربوط به محیط پردازش محاوره‌ای است، گزینه ۲ نادرست است زیرا تطبیق زبان‌های طراحی‌شده برای محیط دسته‌ای به محیط محاوره‌ای مشکل است، به دلیل این‌که ویژگی‌های ورودی ـ خروجی محاوره‌ای متفاوت از عملیات فایل‌ها است. گزینه ۴ نیز نادرست است چون محیطی که یک برنامه مستقیماً با کاربر تعامل دارد، محیط محاوره‌ای است.

مثال۳: در سیستم‌های تعبیه شده ..... .

۱) پردازش خطا از اهمیت ویژه‌ای برخوردار است.

۲) الزامی به پاسخ‌دهی به ورودی و تولید خروجی در زمان محدود ندارد.

۳) قابلیت حمل برنامه به عنوان مهمترین خصوصیت محسوب می‌شود.

۴) خرابی سیستم اهمیت ندارد.

☑ **پاسخ:** گزینه «۱» در این سیستم‌ها اهمیت ویژه‌ای برای پردازش خطا وجود دارد. گزینه ۲ نادرست است، سیستم‌های تعبیه شده در زمان بلادرنگ کار می‌کنند یعنی عمل سیستم بزرگی که سیستم کامپیوتری در آن تعبیه شده است، نیازمند این است که سیستم کامپیوتری قادر باشد به ورودی پاسخ دهد و در زمان محدود، خروجی را تولید کند. مثلاً عکس‌العمل سریع کامپیوتر مورد استفاده در پرواز هواپیما نسبت به تغییرات سرعت و ارتفاع. گزینه ۳ نادرست است، قابلیت اطمینان به عنوان مهمترین معیار در سیستم‌های تعبیه شده است. گزینه ۴ نادرست است، خرابی اهمیت دارد و در صورت بروز، هزینه گزافی را به دنبال خواهد داشت.

نحو و معنای زبان

نحو (Syntax)زبان: نحو زبان برنامه‌سازی، ظاهر آن زبان است و مفهوم قواعد نحوی این است که مشاهده شود دستورات، اعلان‌ها و سایر ساختارهای زبان چگونه نوشته می‌شوند.

معنای (Semantic) زبان: معنای زبان همان مفهومی است که به ساختارهای نحوی زبان داده می‌شود.

مثال: اعلان آرایه بردار عنصری از نوع صحیح:

تعریف آرایه در پاسکال v: array[0..9] of integer;

تعریف آرایه در C int v[10];

معنا: هر دو اعلان بدین معناست که ۱۰خانه برای اعداد صحیح در نظر گرفته شود.

ویژگی‌های یک زبان خوب

۱ـ وضوح، سادگی و یکپارچگی: یک زبان باید مجموعه‌ای از مفاهیم وضوح، سادگی و یکپارچگی را برای طراحی الگوریتم‌های مورد استفاده تدارک ببیند. داشتن حداقلی از این مجموعه مفاهیم (وضوح، سادگی و یکپارچگی) همراه با قوانین مربوط به ترکیب آن‌ها را **«جامعیت مفهومی»** (Conceptual Integrity) می‌گوئیم. همچنین قابلیت خوانایی برنامه‌ها در یک زبان، اصل مهمی به شمار می‌آید. اگر یک زبان، نحو مختصر و رمزی داشته باشد و یا ساختارهای نحوی داشته باشد که معانی مشابه دارند در نتیجه قابلیت خوانایی پایینی خواهد داشت. لذا زبان‌ها باید نحوی با قابلیت خوانایی بالا داشته باشند.

۲ـ قابلیت تعامد (Orthogonality): منظور از تعامد این است که بتوان ویژگی‌های مختلفی از یک زبان را با هم ترکیب کرد و ترکیب حاصل نیز با معنا باشد. یعنی اینکه یک زبان برنامه‌سازی این قابلیت را داشته باشد که مجموعه نسبتاً کوچکی از ساختارهای اولیه بتوانند به چند روش با هم ترکیب شوند تا ساختمان داده‌ها و ساختارهای کنترلی یک زبان را بسازند و هر ترکیبی از ساختارهای اولیه معتبر و دارای معنا باشد.

به عنوان مثال فرض کنید زبانی دارای چهار نوع اولیه (صحیح، اعشاری، دقت مضاعف و کاراکتری) و دو عملگر نوع (آرایه و اشاره‌گر) است. اگر دو عملگر نوع بتوانند بر روی خودشان و همچنین بر روی چهار نوع داده اولیه عمل کنند، تعداد زیادی از ساختمان داده‌ها را می‌توان تعریف کرد. اشاره‌گرها باید بتوانند به هر نوع متغیر یا ساختمان داده‌ای اشاره کنند، اگر اشاره‌گرها نتوانند به آرایه اشاره کنند؛ بسیاری از این حالت‌ها حذف خواهند شد.

هرچه قابلیت تعامد طراحی زبان بیشتر باشد، نیاز به استثناهای کمتری در قوانین زبان است. در نتیجه وجود نظم در طراحی بیشتر و نهایتاً یادگیری، خوانایی و درک زبان آسان‌تر می‌گردد. البته وجود قابلیت تعامد زیاد نیز می‌تواند مشکل‌زا باشد و در واقع جنبه منفی به خود بگیرد و آن اینست که برنامه‌ای که ترکیبی از ویژگی‌ها را دارد ممکن است از نظر منطقی روشن نباشد و اجرای آن نیز ناکارآمد باشد. به عنوان نمونه، یکی از متعامدترین زبان‌ها، ALGOL 68 است؛ هر ساختار در این زبان دارای نوع است و هیچ محدودیتی در مورد نوع‌ها وجود ندارد. همچنین اغلب ساختارها، مقداری را تولید می‌کنند. لذا این آزادی در ترکیب ساختارها، منجر به ساختارهای پیچیده می‌شود.

۳ـ طبیعی بودن برای کاربردها: هر زبان خوب می‌بایست نحوی داشته باشد که بتوان الگوریتم‌های مختلف را با آن پیاده‌سازی کرد. زبان‌ها باید ساختمان داده‌ها، عملگرها، ساختارهای کنترلی مناسب و نحو طبیعی را برای مسئله‌ای که باید حل شود، داشته باشند. یکی از دلایل اصلی تولید مجدد زبان‌ها، نیاز به ویژگی طبیعی بودن است. زبانی که برای کاربردهای خاصی مفید است، ایجاد برنامه‌ها در این زمینه‌ها را ساده می‌کند.

۴ـ پشتیبانی از انتزاع (Abstraction): انتزاع، توانایی تعریف و استفاده از عملیات و ساختارهای پیچیده است، بدون اینکه نیاز به دانستن بسیاری از جزئیات باشد. انتزاع، مفهومی مهم در طراحی زبان‌های برنامه‌سازی جدید است و در متدولوژی‌های نوین ایجاد نرم‌افزار، نقش مهمی را ایفا می‌کند. زبان‌های برنامه‌سازی می‌توانند از دو گروه از انتزاع‌ها پشتیبانی کنند: **انتزاع فرآیند** و **انتزاع واژه‌ها**.

نمونه‌ای از انتزاع فرآیند، استفاده از زیربرنامه برای پیاده‌سازی الگوریتم مرتب‌سازی است که چندین بار در برنامه اجرا می‌شود. بدون استفاده از زیربرنامه، مرتب سازی باید چندین بار تکرار شود و همچنین این کار منجر به تداخل برنامه با جزئیات الگوریتم مرتب سازی می‌شود که هم جریان برنامه را خدشه‌دار می‌کند و هم منجر به طولانی شدن برنامه می‌گردد.

نمونه‌ای از انتزاع داده‌ها: سیستم دانشگاه را در زبان C در نظر بگیرید، ساختمان داده‌های انتزاعی برای موجودیت‌هایی نظیر «استاد»، «دانشجو»، «درس» توسط برنامه‌نویس تعریف می‌شود. بنابراین برنامه‌نویس می‌تواند از آن‌ها در قسمت‌های دیگر برنامه، فقط با دانستن خصوصیات انتزاعی استفاده کند.

۵ـ سهولت در وارسی برنامه: قابلیت اعتماد (Reliability) در برنامه‌های یک زبان برنامه‌سازی همواره مورد توجه بوده است. برنامه‌ها باید به گونه‌ای باشند که وارسی صحت عملکرد آنها ساده باشد. برنامه‌هایی که ساختار نحوی و معنایی ساده‌تری دارند وارسی آنها نیز ساده‌تر می‌باشد.

۶ـ محیط برنامه‌نویسی: مجموعه‌ای از ابزارها که جهت تولید نرم‌افزار به کار می‌روند، محیط برنامه‌نویسی را تشکیل می‌دهد. این مجموعه ممکن است متشکل از سیستم فایل، پیوند دهنده و کامپایلر باشد و یا شامل مجموعه بزرگی از ابزارهای مجتمع باشد که هر کدام واسط کاربر خاصی داشته باشد. ساختار تکنیکی زبان برنامه‌نویسی یکی از جنبه‌هایی است که به کارگیری زبان را تحت تأثیر قرار می‌دهد و می‌توان گفت یک زبان با محیط برنامه‌نویسی قدرتمند باعث افزایش کاربرد آن زبان می‌گردد.

اگر بخواهیم چند محیط برنامه‌نویسی را برشماریم، می‌توان UNIX که یک محیط برنامه‌نویسی قدیمی‌تر است و JBuilder که یک محیط برنامه‌نویسی برای تولید برنامه‌های java است را نام برد. همچنین محیط تولید نرم‌افزار که توسط Microsoft visual studio.NET ارائه شده است، جزء آخرین محیط‌های برنامه‌نویسی است که شامل مجموعه بزرگی از ابزارهای تولید نرم‌افزار است و جهت تولید نرم‌افزار در پنج زبان تحت سکوی .NET (Platform، یعنی Visual BASIC.NET ،C++, J#, JScript ،C# به‌کار می‌رود.

۷ـ قابلیت حمل برنامه (Portability): یک معیار مهم برای بسیاری از پروژه‌های برنامه‌نویسی، قابلیت حمل برنامه است و منظور این است که امکان انتقال برنامه از یک سیستم کامپیوتری به سیستم کامپیوتری دیگر مهیا باشد. زبانی که به ماشین خاصی وابسته نباشد، به آسانی از ماشینی به ماشین دیگر منتقل می‌شود.

۸ـ هزینه: عوامل زیادی در هزینه نهایی یک زبان برنامه‌سازی دخیل هستند از جمله:

الف) هزینه آموزش برنامه‌نویسان برای استفاده از زبان: یادگیری زبان‌های قدرتمند نیاز به تلاش بیشتر دارد. همچنین این هزینه به محیط برنامه‌نویسی نیز بستگی دارد.

ب) هزینه نوشتن برنامه: در طراحی زبان‌ها تلاش می‌شود تا هزینه تولید نرم‌افزار در آن‌ها کمینه باشد.

ج) هزینه ترجمه برنامه‌ها: زبان‌هایی که در محیط‌های دانشجویی استفاده می‌شوند، باید هزینه ترجمه کمتری را تحمیل کنند، زیرا اینگونه برنامه‌ها معمولاً چندین بار ترجمه می‌شوند.

د) هزینه اجرای برنامه: این هزینه وابستگی زیادی به طراحی زبان دارد، چنانچه در زبانی کنترل‌های زیادی در زمان اجرا صورت گیرد، هزینه اجرای آن بالا است. باید بین هزینه ترجمه و سرعت اجرای کد ترجمه شده، توازن برقرار باشد. کامپایلر از مجموعه‌ای از تکنیک‌ها جهت برقراری توازن بین کاهش اندازه کد و افزایش سرعت اجرای کد تولید شده توسط آن، استفاده می‌کند که به آن بهینه‌سازی (Optimization) می‌گویند.

هـ) هزینه پیاده‌سازی زبان: اگر هزینه پیاده‌سازی زبان زیاد باشد و یا فقط بر روی سخت‌افزارهای گران اجرا شود، میزان گستردگی استفاده از آن کاهش می‌یابد. وجود سیستم‌های کامپایلر/ مفسر بود که باعث پذیرش سریع و گسترده زبان جاوا شد.

و) هزینه اعتماد: مواجهه با شکست در سیستم‌های حیاتی، و غیر حیاتی هزینه‌های بالایی را تحمیل می‌کند.

ز) هزینه نگهداری برنامه‌ها: این هزینه شامل اصلاح برنامه، افزودن امکانات جدید به برنامه (در اثر تغییر در سخت‌افزار، سیستم‌عامل یا خواسته‌های سازمان به نرم‌افزار) می‌شود.

مثال۴: تعریف زیر مربوط به کدام‌یک از ویژگی‌های زبان‌های برنامه‌سازی است؟

«توانایی ترکیب ویژگی‌های مختلف یک زبان به طوری که ترکیب حاصل نیز با معنا باشد.»

۱) قابلیت تعامد ۲) قابلیت اعتماد ۳) قابلیت حمل برنامه ۴) قابلیت ترکیب

☑ **پاسخ:** گزینه «۱» این تعریف مربوط به قابلیت تعامد است که یک ویژگی برای زبان برنامه‌نویسی خوب محسوب می‌شود.

دسته‌بندی مدل‌های زبان

دسته‌بندی مدل‌های زبان

زبان‌های برنامه‌سازی از جنبه مدل‌های محاسبات مختلف معمولاً به چهار دسته تقسیم می‌شوند:

زبان‌های دستوری (Imperative)

زبان‌های تابعی (Function)

زبان‌های قانونمند (Rule based)

زبان‌های شیءگرا (Object Oriented)

ـ زبان‌های دستوری: زبان‌های دستوری یا رویه‌ای، زبان‌های مبتنی بر فرمان هستند. در این گونه زبان‌ها، برنامه شامل دنباله‌ای از دستورات است و اجرای هر دستور موجب می‌گردد تا مفسر، یک یا چند سلول حافظه را تغییر دهد، یعنی ماشین را به حالت جدیدی ببرد. نحو چنین زبان‌هایی به صورت زیر است:

Statement 1
Statement 2
….
Statement n

دیدگاه اینگونه زبان‌ها، به این صورت است که حافظه از چند سلول تشکیل شده است و اجرای هر دستور مثل جمع کردن دو متغیر و بدست آوردن متغیر جدید دستیابی به دو سلول حافظه، ترکیب مقادیر آن دو سلول و ذخیره نتیجه در سلول دیگر است.

این دیدگاه در زبان‌هایی مانند C، C++، Ada، Pascal پشتیبانی می‌شود.

ـ زبان‌های تابعی: در این گونه زبان‌ها، به جای دنبال کردن تغییر حالت ماشین، عملکرد برنامه دنبال می‌شود. نحو این زبان‌ها به صورت زیر است:

$Function_n$ (… function $_2$ (function $_1$ (data))…)

همانطور که مشخص است محاسبات به وسیله توابع ایجاد می‌شوند. و آخرین تابع است که پاسخ را از داده‌های اولیه تولید می‌کند. Lisp و ML دو زبان تابعی هستند که از این مدل پشتیبانی می‌کنند.

ـ زبان‌های قانونمند: در این‌گونه زبان‌ها، ابتدا شرایطی بررسی می‌شود و در صورت برقرار بودن آن شرایط، فعالیتی صورت می‌گیرد. prolog یکی از متداول‌ترین زبان‌های قانونمند است.

تذکر۲: زبان برنامه‌نویسی منطقی نمونه‌ای از زبان قانونمند است. نحو این زبان‌ها به صورت زیر است:

$enabling\ condition_1 \rightarrow action_1$
$enabling\ condition_2 \rightarrow action_2$
…
$enabling\ condition_n \rightarrow action_n$

نکته ۱: در زبان‌های دستوری هر الگوریتم با جزئیات کامل مشخص می‌گردد و دستورالعمل‌ها ترتیب معینی دارند ولی در زبان‌های قانونمند، قوانین به ترتیب خاصی مشخص نمی‌گردند و سیستم پیاده‌سازی زبان، ترتیب اجرایی را انتخاب می‌کند که منجر به تولید نتیجه مطلوب شود.

ـ زبان‌های شیءگرا: در این گونه زبان‌ها، مدل‌سازی براساس اشیای موجود در دنیای واقعی انجام می‌شود. اشیای داده‌ی پیچیده‌ای ساخته شده و سپس توابعی طراحی می‌شود تا بر روی آن داده‌ها عمل کنند. اشیای پیچیده را می‌توان با استفاده از بسط اشیای ساده و خواصی که از آن اشیاء به ارث می‌گذارند، ایجاد نمود.

باید به این نکته توجه داشت که گرچه ایجاد نرم‌افزار به روش شیءگرایی نقطه مقابل روش رویه‌گرا (Procedure oriented) (زبان‌های دستوری) است. ولی اغلب زبان‌های مشهوری که از برنامه‌نویسی شیءگرا پشتیبانی می‌کنند، از زبان‌های دستوری به وجود آمده‌اند. زبان‌های C# , Java از متداول‌ترین زبان‌های شیءگرای امروزی هستند.

نکته۲: زبان‌های ویژوال (مانند ویژوال بیسیک) و همچنین زبان‌های اسکریپتی (مانند جاوا اسکریپت) به عنوان زیر مجموعه‌ای از زبان‌های دستوری محسوب می‌شوند.

نکته۳: در بعضی از طبقه‌بندی‌های جدید، دسته‌ای دیگر از زبان‌ها تحت عنوان زبان‌های برنامه‌نویسی ترکیبی/ علامت‌دار مطرح می‌شود. مشهورترین زبان علامت‌دار، XHTML است که برای ایجاد طرح کلی اطلاعات در اسناد وب بکار برده می‌شود.

مثال۵: در مورد زبان‌های برنامه‌سازی کدام گزینه صحیح است؟

۱) زبان‌های شیءگرا با وجود ساختن اشیای داده دقیق نمی‌توانند کارایی زبان‌های دستوری را کسب کنند.

۲) زبان‌های تابعی به جای درنظر گرفتن نتیجه مطلوب، داده‌های موجود را درنظر می‌گیرند.

۳) اجرای زبان‌های قانونمند دقیقاً مشابه با زبان‌های دستوری است.

۴) زبان‌های تابعی از مدل سخت‌افزار کامپیوتر پیروی می‌کنند و دستورات را به ترتیب اجرا می‌کنند.

☑ **پاسخ:** گزینه «۴» زبان‌های تابعی از مدل سخت‌افزار کامپیوتر پیروی می‌کنند که دستورات را به ترتیب اجرا می‌کنند. لذا اغلب زبان‌های قدیمی از این مدل پیروی می‌کنند.

مثال۶: کدامیک از زبان‌های زیر تابعی محسوب می‌شوند؟

۱) Pascal ۲) C# ۳) C++ ۴) ML

☑ **پاسخ:** گزینه «۴» زبان Lisp و ML و Scheme جزء زبان‌های تابعی محسوب می‌شوند.

استانداردسازی زبان

هر زبان برنامه‌سازی دارای تعاریف استانداردی است که تمامی پیاده‌سازی‌ها ملزم به رعایت آن‌ها هستند. استانداردها معمولاً به دو دسته تقسیم می‌شوند:

۱. **استانداردهای اختصاصی:** این استانداردها توسط شرکت سازنده و مالک زبان ارائه می‌شوند و در اغلب موارد برای زبان‌های برنامه‌سازی که به طور گسترده به کار می‌روند، مناسب نیستند.

۲. **استانداردهای عمومی:** این دسته از استانداردها توسط سازمان‌های مختلفی مانند IEEE، ANSI، BSI، ISO ارائه می‌گردد و بین پیاده‌سازی‌های مختلف یکنواختی به وجود می‌آورد. معمولاً تهیه استاندارد در تمامی این سازمان‌ها به روش مشابهی صورت می‌گیرد.

در یک مقطع زمانی، گروهی تصمیم می‌گیرد که زبانی نیاز به یک استاندارد دارد و زمانی که گروه کاری، بر روی استاندارد توافق عمومی کردند، اختلاف سلیقه‌ها از بین می‌رود و استاندارد تولید می‌شود.

بخشی از استانداردها تکنیکی و بخش دیگر سیاسی است. به عنوان نمونه، فروشندگان کامپایلر، به مسایل اقتصادی فکر می‌کنند. آنها می‌خواهند که استانداردهای کامپایلر، همانند محصولات آنها باشد یا تغییر زیادی در آنها ایجاد نشود.

بنابراین می‌توان گفت فرایند استاندارد، یک فرایند عمومی است و هرکسی می‌خواهد که زبان حاصل، توسط همه‌ی افراد قابل قبول باشد.

به منظور استفاده مؤثر از استانداردها، باید سه فاکتور زیر را در نظر بگیریم:

۱. **به هنگام بودن (Timeliness)**: چه زمانی زبان را استاندارد کنیم؟

۲. **مطابقت (Conformance)**: این که یک برنامه با یک استاندارد مطابقت داشته باشد و یک کامپایلر، یک استاندارد را کامپایل کند، به چه معناست؟

۳. **کهنگی (Obsolescence)**: چه زمانی استاندارد کهنه می‌شود و چگونه باید اصلاح شود؟

نکته۴: استانداردسازی یک زبان برنامه‌سازی برروی سه فاکتور **قابلیت استفاده مجدد، قابلیت نگهداری و قابلیت آزمون** آن زبان برنامه‌سازی تأثیرگذار است.

تست‌های طبقه‌بندی شده فصل اول

۱ـ کدام‌یک از معیارهای زیر برای انتخاب زمان مناسب درکاربردهای توکار (Embedded Systems) اهمیت بیشتری دارد؟

(مهندسی کامپیوتر ـ سراسری ۸۲)

۱) یکنواختی (Uniformity)
۲) قابلیت توسعه (Extensibility)
۳) عمومیت (Generality)
۴) قابلیت اطمینان (Reliability)

پاسخنامه تست‌های طبقه‌بندی شده فصل اول

۱ـ گزینه «۴» مهم‌ترین معیاری که در سیستم‌های توکار(Embedded systems) مورد توجه قرار می‌گیرد، قابلیت اطمینان(Reliability) است. قابلیت اطمینان معیاری است مرتبط با این موضوع که سیستم هرچند وقت یک‌بار با شکست مواجه می‌شود. احتمال مواجهه با شکست در این سیستم‌ها حائز اهمیت است و در صورت بروز شکست هزینه زیادی را به دنبال خواهد داشت. از همین رو در سیستم‌های توکار اداره خطاها و استثناها نیز از اهمیت ویژه‌ای برخوردار است.

آزمون فصل اول

۱ـ اولین زبان سطح بالا در زمینه کاربردهای تجاری کدام گزینه است؟

۱) COBOL ۲) Lisp ۳) C++ ۴) Prolog

۲ـ کدام گزینه از ویژگی‌های یک زبان خوب محسوب نمی‌شود؟

۱) سهولت در وارسی برنامه ۲) پشتیبانی از انتزاع ۳) قابلیت تعامد ۴) قابلیت اعتماد

۳ـ متعامدترین زبان برنامه‌سازی کدام گزینه است؟

۱) Lisp ۲) ALGOL 68 ۳) Cobol ۴) Fortran 77

۴ـ کدام دسته از زبان‌های زیر زبان‌های دستوری محسوب می‌شوند؟

۱) Lisp ، ML ،C++ ۲) Java ، C++ ، Lisp ۳) Ada ، C++ ، C ۴) C ، ML ، Ada

۵ـ کدام گزینه در مورد قابلیت تعامد صحیح است؟

۱) هرچه قابلیت تعامد طراحی زبان بیشتر باشد، خوانایی و درک برنامه سخت‌تر می‌گردد.

۲) قابلیت تعامد از کارآیی می‌کاهد.

۳) هرچه قابلیت تعامد بیشتر باشد، نشان از وجود نظم کمتر در طراحی است.

۴) هیچکدام

۶ـ نحو زیر مربوط به کدام یک از مدل‌های زبان برنامه‌سازی است؟

$Function_n(\ldots function_2\ (function_1\ (data))\ldots)$

۱) قانونمند ۲) دستوری ۳) تابعی ۴) شیءگرا

۷ـ محیطی که برنامه در آن طراحی، کد، آزمون و اشکال‌زدایی می‌شود چه نام دارد؟

۱) محیط میزبان ۲) محیط عملیاتی ۳) محیط مقصد ۴) محیط پردازش

۸ـ در کدام یک از محیط‌های عملیاتی، قابلیت اعتماد و صحت از اهمیت بیشتری برخوردار است؟

۱) محیط دسته‌ای ۲) محیط سیستم‌های توزیعی ۳) محیط محاوره‌ای ۴) محیط سیستم تعبیه‌شده

۹ـ کدام گزینه خصوصیت محیط پردازش دسته‌ای را به درستی بیان می‌کند؟

۱) برنامه‌ها باید از محدودیت‌های زمانی برخوردار باشند.

۲) در محیط دسته‌ای، یک برنامه مستقیماً با کاربر تعامل دارد.

۳) در محیط دسته‌ای خطاهایی که اجرای برنامه را خاتمه می‌دهند، قابل قبول ولی هزینه‌بر است.

۴) در محیط پردازش دسته‌ای، پردازش‌ها به صورت بلادرنگ هستند.

۱۰ـ در مورد یک زبان برنامه‌سازی با نحو (syntax) پیچیده کدام گزینه صحیح می‌باشد؟

۱) قابلیت خوانایی پایین و اصلاح و آزمون آن مشکل خواهد بود. ۲) قابلیت خوانایی بالا و اصلاح و آزمون آن مشکل خواهد بود.

۳) قابلیت خوانایی پایین و اصلاح و آزمون آن آسان خواهد بود. ۴) قابلیت خوانایی بالا و اصلاح و آزمون آن آسان خواهد بود.