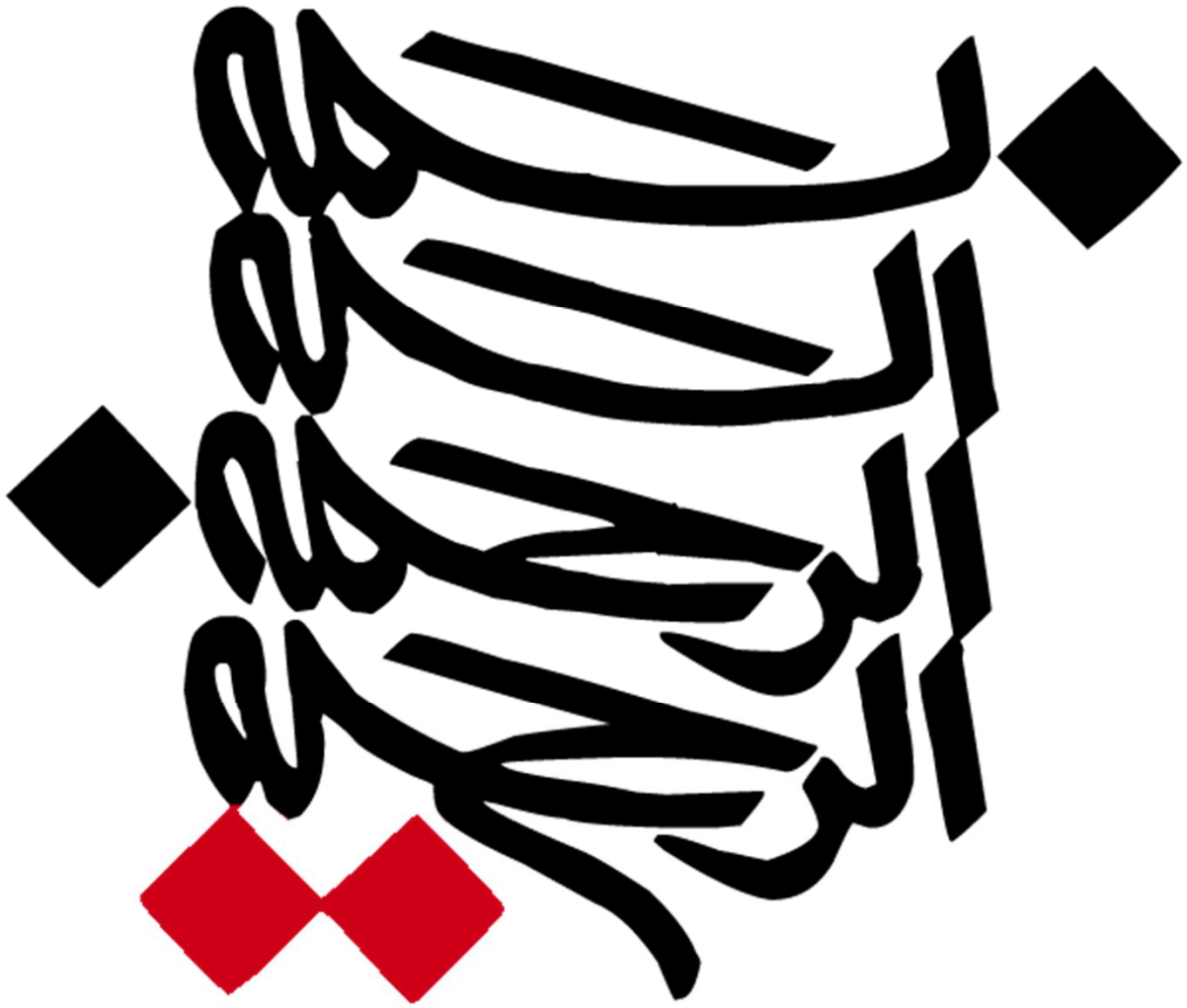
Chapter 1

طراحی و پیاده سازی زبانهای برنامه سازی

دبلیو پرات- زیلکو ویتز

حامد صباغ

http://www.h-sabbagh.ir
Email:sabbagh.std@gmail.com



طراحی و پیاده سازی
زبان های برنامه سازی

Chapter 1

طراحی و پیاده سازی زبانهای برنامه سازی

حامد صباغ

http://www.h-sabbagh.ir
Email:sabbagh.std@gmail.com

دبلیو پرات- زیلکوویتز

فصل اول :
اصول طراحی زبان ها

آنچه در این فصل خواهید آموخت :

- دلایل مطالعه زبان های برنامه سازی
- تاریخچه ای از زبان های برنامه سازی
  - زبان های محاسباتی
  - زبان های تجاری
  - زبان های هوش مصنوعی
  - زبان های سیستمی

- تاثیر محیط بر روی طراحی و پیاده سازی زبان ها
  - محیط دسته ای
  - محیط محاوره ای
  - محیط سیستم های تعبیه شده
  - محیط کامپیوتر شخصی
  - محیط شبکه و اینترنت

- دامنه کاربرد زبان ها
- ویژگی های یک زبان خوب
- نحو و معنای زبان
- مدل های محاسباتی زبان
- زبان های دستوری
  - زبان های تابعی
  - زبان های قانونمند
  - زبان های شی گرا
- استاندارد سازی زبان ها
  - زمان شناسی
  - اطاعت و پیروی

زبان برنامه نویسی : «هر گونه علامت گذاری جهت توصیف الگوریتم ها و ساختمان داده ها »

1-1-دلایل مطالعه زبان های برنامه سازی

1-1-1-برای افزایش توانایی های خود در توسعه الگوریتم های کارآمد

- بسیاری از زبان ها ویژگی هایی دارند که اگر به خوبی مورد استفاده قرار گیرند به نفع برنامه نویس است ، ولی اگر به طور نامناسب مورد استفاده قرار گیرند ، وقت زیادی از برنامه نویس می گیرند. مثلاً اگر برنامه نویس از تکنیک بازگشتی ، مزایا و مشکلات آن اطلاع داشته باشد ، می تواند تصمیم بگیرد چه موقعی از آن ها استفاده کند یا نکند.

2-1-1استفاده بهینه از زبان های برنامه نویسی موجود

- با درک اینکه چگونه ویژگی های یک زبان پیاده سازی می شوند توانایی شما در نوشتن برنامه های کار آمد افزایش می یابد به عنوان مثال اگر بدانید آرایه ها ،رکورد ها ، لیست ها و رشته ها در زبان برنامه نویسی مورد نظرتان چگونه ایجاد و پیاده سازی می شوند می توانید از زبان برنامه نویسی به طور بهینه بهره ببرید.

1-1-3-آشنایی با اصطلاحات مفید ساختار های برنامه نویسی

- زبان ها هم می توانند به عنوان وسیله ای برای محدودیت فکر کردن و هم وسیله ای برای کمک به فکر کردن باشند اگر برنامه نویس فقط با یک زبان آشنایی داشته باشد در جستجو برای حل مسئله ، فقط به توانایی های زبانی فکر می کند که با آن آشنا است و این یک محدودیت است. با مطالعه زبان های برنامه نویسی متعدد ، دامنه لغات یک برنامه نویس افزایش می یابد. به عنوان مثال یک ساختار کنترلی به نام همروال در خیلی از برنامه ها مفید است ولی زبان های محدودی این قابلیت را پتشیبانی می کنند. (مانند C و فرترن)

4-1-1-انتخاب بهترین زبان برنامه نویسی

- آگاهی از زبان های مختلف باعث می گردد که برای مسئله خاص بتوان زبان بهتری را انتخاب کرد. به عنوان مثال کاربردهایی که نیاز به محاسبات عددی دارند بهتر است از C ، فرترن یا ada استفاده کنند. در کاربرد های هوش مصنوعی از Prolog , ML,Lisp و جهت کاربرد های اینترنت، Java , Perl مناسب هستند.

1-1-5-یادگیری آسان یک زبان جدید

- اگر با ساختار یک زبان طبیعی آشنایی کامل داشته باشید ، آموزش زبان طبیعی جدید ساده خواهد شد. این موضوع در مورد زبانهای برنامه نویسی نیز صادق است.

1-1-6-طراحی یک زبان جدید

- برخی از برنامه نویسان خود را به عنوان طراح زبان می دانند مثلاً طراح یک واسط کاربر جهت نوشتن برنامه های بزرگی مثل ویرایشگر متن یا سیستم عامل ، می بایست با موارد مشابهی که در طراحی زبان های برنامه نویسی همه منظوره مورد استفاده قرار می گیرند آشنایی داشته باشد.

2-1-تاریخچه زبان های برنامه نویسی

- فرترن و Lisp در دهه 1950 ، پاسکال ، Prolog , Ada و اسمالتاک در دهه 1970 ، C++ , Perl , ML در دهه 1980 و Java در دهه 1990 طراحی شدند . زبان فرترن برای کارهای عملی و محاسباتی ساخته شده بود. تاریخچه زبانهای برنامه نویسی را براساس کاربرد آنها یعنی محاسباتی ، تجاری ،هوش مصنوعی ، و سیستمی مورد بررسی قرار می دهیم.

YEAR	LANGUAGE
1950-55	Assembly
1956-60	Fortran , Algol58 , Algol60 , cobol , lisp
1961-65	Fortran IR, cobol 61 , Algol 61 , snobol , APL
1966-70	Fortran66 , coblol 65 , snobol4 , simula 67 , basic , APL360
1971-65	Pascal , cobol 74 , APL(standard) , C
1976-80	Ada , Fortran 77(standard)
1981-85	prolog
1985-90	C++ , Fortran90

جدول 1-1

1-2-1-زبان های محاسباتی (مبتنی بر اعداد)

در حدود سال 1960 زبان الگول به عنوان یک زبان محاسباتی آکادمیک معرفی شد . اهداف اصلی الگول در آن زمان شامل موارد زبر بود :

- این زبان باید به ریاضیات نزدیک باشد.
- جهت توصیف الگوریتم ها مناسب باشد.
- برنامه ها باید به زبان ماشین ترجمه گردند.
- این زبان نباید وابسته به یک ماشین خاصی باشد.

1-2-2- زبان های تجاری

- هدف از زبان های تجاری این بود که ، برنامه هایی که ایجاد می شوند از متن هایی به شکل متن انگلیسی استفاده کنند. کوبول ، زبانی جهت کاربرد های تجاری می باشد. هدف مهم در کوبول ، نوشتن برنامه ای بود که به زبان انلگیسی نزدیک باشد. هر چند که کوبول خوانایی خوبی دارد اما نحو رسمی ندارد و برنامه نویس در آن دشوار است.
- نکته PL/I ویژگی های عددی فرترن را با خصوصیات برنامه سازی تجاری کوبول ترکیب کرد.
- PL/I=Cobol+Fortran

1-2-2-زبان های هوش مصنوعی

تمایل به زبان های هوش مصنوعی در دهه 1950 با IPL شروع شد. لیسپ ، به عنوان یک زبان تابعی پردازش کننده لیست طراحی شد. زبان لیسپ جهت کاربرد های هوش مصنوعی شامل ویژگی های زیر است :

- یک زبان تابعی پردازش کننده لیست است.
- عملیات جستجو را به خوبی انجام می دهد.
- پیاده سازی بازیهای کامپیوتری در آن به خوبی صورت می گیرد.
- قابلیت های مناسبی جهت پردازش متن دارد.
- رشته ایی از نماد ها میتوانند توسط رشته های دیگر جایگزین شوند (تفسیر ماشین خودکار) .
- نکته : لیسپ ،جهت پردازش لسیت همه منظوره طراحی شد ولی پرولوگ یک زبان تک منظوره بود که ساختار آن براساس منطق ریاضی بود.

1-2-4-زبان های سیستمی

این دسته از زبانها برای نوشتن سیستم عامل و پیاده سازی کامپایلرها و ... کاربرد دارند. این گروه از زبان ها باید قادر به دستیابی به سخت افزار و ایجاد ارتباط با آن باشند مثل C , PL/I و اسمبلی .

1-3-تاثیر محیط اجرایی بر روی طراحی و پیاده سازی زبان ها

توسعه نرم افزار در خلا انجام نمی شود سخت افزاری که زبان پشتیبانی می کند تاثیر زیادی بر طراحی زبانه برنامه نویسی دارد. محیطی که برنامه بر روی آن اجرا می شود (شامل سیستم عامل و سخت افزار) محیط عملیاتی (مقصد) نام دارد. محیطی که برنامه در آن ایجاد ،تست و اشکال زدایی می شود ، محیط میزبان نام دارد. محیط عملیاتی ممکن است با محیط میزبان متفاوت باشد.

در ذیل به چند نمونه از این محیط ها که توسعه و اجرای نرم افزار در آن ها صورت می گیرد اشاره می کنیم:

1-3-1- محیط دسته ای

برای زبان هایی که جهت محیط های دسته ای یا offline طراحی شده اند. فایل ها معمولی ترین ابزار جهت ورودی / خروجی هستند. برنامه ، مجموعه ای از فایل های داده را به عنوان ورودی گرفته، داده های آنها را پردازش کرده و مجموعه ای از فایل های داده خروجی را تولید می کند. این محیط عملیاتی را پردازش دسته ای می گویند،زیرا اطلاعات ورودی در ف**ایل ها** دسته بندی می شوند و به صورت دسته ای پردازش می شوند.

در این محیط خطایی که اجرای برنامه را خاتمه دهد قابل قبول بوده ولی هزینه بر است. زیرا پس از پردازش و تصحیح خطا ، برنامه باید به طور کامل اجرا شود.

ویژگی دیگر محیط دسته ای ،**محدودیت زمانی** برای برنامه است یعنی زبان استفاده شده برای این محیط ، امکاناتی را جهت تاثیر بر روی سرعت اجرای برنامه در اختیار نمی گذارد به عبارتی دیگر ، زبان استفاده شده در محیط دسته ای TIMING مشخصی ندارد.

زبان هایی مثل فرترن ،کوبل و پاسکان ابتدا برای محیط های دسته ای طراحی شدند ولی امروزه در محیط های محاوره یا سیستم تعبیه شده نیز به کار می روند.

1-3-2- محیط محاوره ای

- در این محیط یک برنامه مستقیماً با کاربر تعامل دارد و خروجی در نمایشگر نمایش داده میشود.
- اینگونه محیطها از سیستم **اشتراک زمانی** برای انجام کارهای مختلف ، که در یک زمان واحد به کامپیوتر داده میشود استفاده می کنند. به این ترتیب که به هر برنامه یک برش زمانی اختصاص داده میشود. بعد از اتمام این برش زمانی ،پردازنده به برنامه دیگری که در حال اجراست داده می شود.
- پردازش خطا در محیط محاوره ای از اهمیت کمتری برخوردار است و زبان به کار رفته در این محیط نیاز مبرمی به داشتن پردازش خطاهای قوی ندارد. زیرا کاربر به صورت Online پشت کامپیوتر بوده و در صورت بروز خطا میتواند برنامه را دستکاری و تصحیح کند ، اما خاتمه برنامه در صورت بروز خطا قابل قبول نیست.
- زبان های C , C++ برای نوشتن برنامه های محاوره ای مناسب هستند.

1-3-3- محیط سیستم های تعبیه شده (توکار)

- به سیستم کامپیوتری که جهت کنترل بخشی از یک سیستم بزرگ مثل هواپیما یا ماشین آلات صنعتی استفاده می شودسیستم کامپیوتری تعبیه شده یا توکار گفته می شود.
- در محیط های دسته ای و محاوره ای خرابی سیستم چندان اهمیت ندارد ولی در سیستم توکار ، خرابی سیستم زیان های جدی به بار می اورد .
- سیستم های توکار ، معمولا به صورت **بلادرنگ** هستند یعنی در یک محدوده زمانی از قبل تعیین شده باید به ورودی های پاسخ دهند.

ویژگی های محیط تعبیه شده

- برنامه های نوشته شده برای اینگونه محیط ها معمولاًبدون سیستم عامل ، بدون سیستم فایل و بدون دستگاه های I/O فایل اجرا می شوند. در عوض هر کدام از برنامه ها رویه های مخصوصی برای ارتباط با دستگاه های ورودی/ خروجی سیستم بزرگ دارند و به تعامل با آنها می پردازند.
- قابلیت اطمینان از اهمیت خاصی برخوردار است.
- در این نوع سیستم ها اداره خطاها و استثناها باید به خوبی انجام گیرد. خاتمه برنامه جز در موارد خرابی کلی سیستم قابل قبولی نیست و کاری وجود ندارد که به صورت محاوره ای خطاها را بر طرف سازد.
- در سیستم های توکار اغلب از کامپیوترهای RISC که تعداد دستورات محدودی دارند استفاده می شود. زبان های Ada , C , C++ برای نوشتن برنامه های مربوط به سیستم توکار مفید هستند.

4-3-1-محیط کامپیوتر شخصی

- در موارد زیادی در محیط کامپیوترهای شخصی ، کارایی ، کمتر مورد نظر قرار می گیرد و هدف اصلی اغلب ، راحتی کاربر و داشتن یک محیط گرافیکی ساده است.
- نمونه مشهور آن برنامه نویسی تحت محیط ویندوز است . برنامه نویسی شی گرا مدل مناسبی برا ی چنین محیط هایی است و زبان هایی مثل c++ یا java در چنین محیطی کاربرد زیادی دارند.

5-3-1-محیط شبکه و اینترنت

- اینترنت اولیه ، دو سرویس TelenetوFTP داشت . در پروتکل Telenet کاربر به عنوان بخشی از کارگزار راه دور عمل می کرد و به کمک FTP می توانست فایل هایی را از ماشین کارگزار گرفته و یا به آن بفرستند. در هر دو پروتکل ، کاربر باید بداند که چه ماشینی اطلاعات مورد نیاز او را دارد. بعد از ایجاد زبان HTML و پروتکل انتقال HTTP و استفاده از وب جهانی (WWW) ، نقش زبان برنامه سازی تغییر کرد. این زبان ها باید تعامل بین کامپیوترهای Clinet وServer را امکانپذیر سازند.

4-1-دامنه کاربرد زبان ها

- دردهه 1960 اغلب برای کاربرد های تجاری از زبان کوبول ، برای امور علمی از فرترن و الگول استفاده می شد. امروزه برای کاربرد های تجاری از کوبول ، برای صفحه گسترده ها از زبان های نسل چهارم (4GL) , C++ , Java برای کاربرد های علمی از فرترن C , C++ , Java برای کاربرد های سیستمی از , Java C , C++ برای هوش مصنوعی ، از لیسپ و پرولوگ استفاده می شود.
- برنامه های هوش مصنوعی با الگوریتم هایی نوشته می شوند که عمل جستجو را در بخش بزرگی از داده ها انجام می دهند. مثلاً برای شطرنج ، کامپیوتر حرکت های زیادی را ایجاد کرده و آنگاه بهترین حرکت را انتخاب می کند در دنیای اینرنت ، زبان هایی مثل Perl و جاوا اسکریپت ، به سرور امکان می دهندکه داده ها را از کاربر گرفته و تراکنشی را انجام دهند.
- امروزه بسیاری از دستگاه ها مثل خودرو ها و تلویزیون های دیجیتالی ، پردازنده دارند و این وسایل به زبان های بی درنگ نیاز دارند که C , C++ , Ada در این موارد کاربرد دارند. ML در تحقیقات زبان های برنامه سازی به کار گرفته شده است. هر چند اسمالتاک زبان معروفی نیست. ولی خیلی از خصوصیات شی گرایی C++ از اسمالتاک گرفته شده است.

جدول زیر کاربرد زبان های برنامه نویسی در زمان گذشته و حال را نشان می دهند .

زبان مورد استفاده	کاربرد	دوره
COBOL	تجاری	1960
Fortran , Basic , Algol, APL	عملی	
ASSEMBLY	سیستمی	
Lisp , snobol	هوش مصنوعی	
C++ , JAVA , 4gl	تجاری	امروزه
Java , C , C++ , Basic	عملی	
C,C++ , Java	سیستمی	
Lisp , prolog	هوش مصنوعی	
Tex , pestscript	انتشارات	

کاربرد زبان های برنامه سازی

- تجاری : مانند طراحی پایگاه داده ها (cobol , C , PL / I)
- علمی : مثل کارهای محاسباتی که فرمول های زیادی دارند. (, Basic , C++ C, Fortran)
- سیستمی : یعنی چیزی شبیه به سیستم عامل بنویسیم . (, C, C++ Assembly)
- هوش مصنوعی : بازی های فکری مثل شطرنج (prolog , lisp)
- انتشارات (publishing) postscript , text ، واژه پردازها
- پردازشی (process) : کارهای پردازشی بالا ، یعنی بیشتر زمان روی CPU و سیستم فایل درگیر است . (Perl , TCL , Unix)
- آموزشی : برای آشنایی با برنامه نویسی (basic , pascal , C)
- New : اغلب کاربردهای بالا را پشتیبانی می کند و به محاوره ای نزدیک هسند. (smaltalk , Ifel , ML)

عوامل موثر بر پیدایش و طراحی زبانها

- **سخت افزار و سیستم عامل** دو جنبه مهم در پیدایش و طراحی زبان ها بوده اند.
- **روش های برنامه سازی** تغییر می کردند و باعث تغییر زبان ها می شدند.
  - روش رویه ای ⟵ روش ساخت یافته (تابعی) ⟵ روش قانونمند ⟵ روش شی گرا
- **کاربرد ها** (آموزشی ، علمی ، تجاری و ...)
- **روش های پیاده سازی** : سازگار با محیط پیاده سازی باشد.
- **مطالعات تئوریک** (مثل زبان α که پیاده سازی نشد اما اصول آن در پایگاه داده به کار می رود)
- **استاندارد سازی** : معیارهای استاندارسازی با توجه به شرایط زمانی و تکنولوژی تغییر می کند.

انواع استاندارد سازی :

- خصوصی (مخصوص سازمان خاص)
- عمومی (همه سازمان ها از آن حمایت می کنند)

مسائل مهم در استاندارد سازی :

- شرایط زمانی
- انطباق و توافق
- کهنگی (از رده خارج شدن)

1-5- ویژگی های یک زبان خوب

1-5-1- وضوح ، سادگی و یکپارچگی

- در یک زبان بهتر است تعداد کمی مفاهیم مختلف و قانون جهت ترکیب آنها وجود داشته باشد این ویژگی را **جامعیت مفهومی** می گویند.
- نحو یک زبان روی نوشتن ، تست کردن ، اصلاح و درک زبان اثر گذار است. قابلیت خوانایی برنامه ها نیز یک اصل مهم می باشد. نحوی که مختصر و رمز گونه است برنامه نویسی را کوتاه تر و ساده تر می کند ، ولی قابلیت خوانایی را کاهش می دهد ، مثلاً برنامه های APL حالت رمز گونه دارند و قابلیت خوانایی کمی دارند.
- بسیاری از زبان ها ، ساختارهای نحوی دارند که دو جمله تقریباً مشابه ، معانی مختلفی دارند در نتیجه قابلیت خوانایی کاهش می یابد. مثلاًکاراکتر B در زبان Snobo14 در یکجا به عنوان جدا کننده و در جایی دیگر به عنوان الحاق کننده دو رشته به کار می رود.

1-5-2- قابلیت تعامد

- منظور از تعامد این است که بتوان خصوصیات مختلفی از یک زبان را با هم ترکیب کرد و این ترکیب با معنا باشد. مثلاً اگر عبارتی در یک زبان فرضی ، بتواند مقداری را تولید کند و علاوه بر آن شامل عباراتی باشد که ارزش T یا F دارند این زبان قابلیت تعامد را داراست.
- مثال دیگر : عبارت محاسباتی و دستور شرطی :

«if (a+b> c+d)then….» یعنی یک دستور محاسباتی را در دستور if به کار بردیم.

- **1-5-3- طبیعی بودن برای کاربرد ها**
- هر زبان در هنگام استفاده کاربرد خاص خود باید مناسبت به نظر آید. اگر ساختار برنامه ، ساختار منطقی مربوط به الگوریتم را به خوبی نشان دهد ،آن زبان این ویژگی را دارا خواهد بود. زبان هایی که برای کاربرد های خاصی هستند این خصوصیت را در همان زمینه کاربردی دارا می باشند. مثلاً برای محاسبات علمی و فنی از فرترن و برای پردازش رشته ها از زبان lisp استفاده می کنیم .

4-5-1- پشتیبانی از انتزاع

- زبان برنامه نویسی باید امکان ایجاد **ساختارهای داده ای جدید** را برای برنامه نویس فراهم کند. هر زبان تعداد معینی نوع داده اولیه دارد.
- مثلاً در زبان C و پاسکال ، داده اعداد مختلط و اعمال روی آنها وجود ندارد. ولی در زبان های مانند C++ , ada برنامه نویس می تواند نوع داده های انتزاعی (ADT) مورد نظر خود را تولید کند. مثلاً می تواند نوع داده اعداد مختلط و عملیات جمع ،تفریق و ضرب را بر روی آنها تعریف کند . استفاده از جنبه های انتزاعی ، قالبیت اطمینان را افزایش میدهد.

5-5-1- سهولت در بازرسی برنامه

- برنامه ها باید به گونه ای باشند که بررسی صحت عملکرد آنها ساده باشد برنامه هایی که ساختار نحوی و معنایی ساده تر دارند بازرسی آنها نیز ساده تر می باشد.
- بررسی درستی برنامه به دو صورت **رسمی** مانند روش های ریاضی و **غیر رسمی** نظیر تست برنامه با ورودی های مختلف انجام می شود.

6-5-1- محیط برنامه نویسی

- یک محیط برنامه نویسی قوی ، ایجاد برنامه ها و عیب یابی آنها را ساده تر می سازد.
- هر چه امکانات محیط برنامه سازی نظیر کامپایلر ، لینکر ، دیباگر و ... بیشتر باشد ، آن زبان برنامه نویسی بهتر خواهد بود.
- مثلاً برنامه های ویژوال عموماً یک محیط برنامه نویسی قدرتمند دارد. اسمالتاک زبانی است که محیط آن دارای پنجره ها ، منوها ، ورودی موس و ... برای کار کردن روی برنامه ها می باشد.

7-5-1- قابلیت حمل

- یک زبان خوب باید بتواند بر روی سیستم های مختلف کامپیوتری به سادگی انتقال یافته و اجرا شود. یکی از معیار های مهم برای پروژه های برنامه نویسی قابلیت انتقال یک برنامه از یک ماشین به ماشین دیگر است.
- زبانیکه به ماشین خاصی وابسته نباشد برنامه های نوشته شده در آن زبان از ماشینی به ماشین دیگر قابل انتقال هستند.
- Ada،C , Fortran و پاسکال تعاریف استانداری برای تولید برنامه های قابل حمل دارند. مثلاً زبان جاوا این ویژگی را در حد بسیار بالایی داردو با افزایش قابلیت حمل ، انعطاف پذیری و کارایی کاهش می یابد.

1-5-8 هزینه استفاده

- هزینه عنصر مهمی در ارزیابی زبان های برنامه نویسی است. 3 معیار اصلی هزینه عبارتند از :
- **الف) هزینه اجرای برنامه :** زمان اجرای یک برنامه خصوصاً برنامه های بزرگی که به دفعات زیادی اجرا می شوند معیاری مهم می باشند هزینه اجرای برنامه شامل استفاده از منابع کامپیوتری خصوصاً RAM , CPU , Disk می باشد.
- **ب)هزینه ترجمه ی برنامه :**یعنی مدت زمانی که طول می کشد تا برنامه کامپایل شود مثلاً برنامه های دانشجویان ، چندین بار ترجمه شده و کمتر اجرا می شود و این هزینه مهم تر است (چون ممکن است خطا داشته باشد و باید چندین بار ترجمه کنید تا کل خطاها بر طرف شود) هزینه ترجمه شامل استفاده از منابع کامپیوتری خصوصاً CPU , DISK , RAM می باشد.
- **ج : هزینه نگهداری برنامه :** شامل هزینه ی تطبیق برنامه با جوانب محیطی،ایجاد امکانات جدید در نرم افزار و بر طرف کردن اشکالات است.

6-1- نحو و معنای زبان

نحو زبان : نحو زبان برنامه سازی ، ظاهر آن زبان است و مفهوم قواعد نحوی این است که مشاهده شود دستورات ،اعلانها و سایر ساختارهای زبان چگونه نوشته می شوند.

معنای زبان : همان مفهومی است که به ساختارهای نحوی زبان داده شود.

مثال : اعلان آرایه بردار عنصری از نوع صحیح

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

Int v[10] تعرف آرایه در سی C

معنا : هر دو اعلان بدین معناست که 10 خانه برای اعداد صحیح در نظرگرفته شود.

7-1- مدل های محاسباتی زبان

چهار مدل محاسباتی مختلف وجود دارد که برنامه نویسی را توصیف می کند که عبارتند از :

1- دستوری

2- تابعی

3- قانونمند

4- شی گرا

1-7-1- زبان های دستوری

در این زبان ها برنامه شامل دنباله ای از دستورات است و اجرای هر دستور موجب می شود مترجم مقدار یک یا چند خانه حافظه را تغییر دهد یعنی ماشین را به حالت جدید وارد کند نحو چنین زبان هایی به صورت زیر است :

دستور 1;

دستور 2;

....

دستور n;

این مدل در واقع از **سخت افزار** کامپیوتر تبعیت می کند چون سخت افزار نیز دستورات را به ترتیب اجرا می کند اغلب زبان های قدیمی مانند , Algol , PL/I C , C++ , Fortran پاسکال Ada , cobol از این مدل پشتیبانی می کند. زبان های امروزه مانند C , C++ و کوبول نیز از این مدل پشتیبانی می کنند.

1-7-2- زبان های تابعی

- در این روش به جای دنبال کردن تغییر حالت ماشین ، **عملکرد** برنامه دنبال می شود. یعنی به جای آنکه داده های موجود را در نظر بگیریم ، **نتیجه مطلوب** را در نظر خواهیم داشت. یعنی در صورت برقراری عملیات بر روی داده ورودی نتیجه مطلوب حاصل می شود. عملی باید روی حالت اولیه ی ماشین انجام پذیرد تا با دستیابی به داده اولیه و ترکیب آن ها پاسخ مناسب بدست آید.
- توسعه برنامه با ایجاد توابعی از توابع ایجاد شده قبلی به منظور ساختن توابع پیچیده انجام می شود تا این داده اولیه را دستکاری کرده و آخرین پاسخ را از داده ای اولیه تولید نماید. ظاهر برنامه در این مدل ، به صورت فراخوان تعدادی تابع و ارسال نتیجه آنها به عنوان پارامتر به توابع قبلی است . نحو این زبان به صورت زیر است :

Fn (… f2 (f1(data)) …)

زبان های ML , lisp زبان های تابعی هستند.

- **3-7-1- زبان های قانونمند**
- در زبان های قانونمند شرایطی بررسی شده و اگر این شرایط برقرار باشند اعمالی انجام می گردد. معروف ترین زبان قانونمند ، PROLOG است که به آن زبان برنامه نویسی منطقی هم گفته می شود.
- **نحوه کلی چنین زبان هایی به شکل زیر است :**

Enabling1 condition1 → action1

Enabling2 condition2 → action2

….. ….. ….

Enabling n condition n → action n

- اجرای این زبان شبیه زبان دستوری است با این تفاوت که دستورات به ترتیب اجرا نمی شوند بلکه *فعال شدن شرط ها ترتیب اجرا را تعیین می کند* این زبان ها اغلب در برنامه های کاربردی هوش مصنوعی و سیستم های خبره مورد استفاده قرار می گیرند.

1-7-4- زبان های شی گرا

در زبان های شی گرا اشیا داده ای پدید آمده و مجموعه ای از توابع تعریف می شوند تا روی این داده ها کار کنند اشیا پیچیده را می توان با بسط اشیای ساده و مفهوم ارث بری ایجاد کرد . در برنامه نویسی شی گرا با ساختن اشیای داده ای دقیق ، کارایی زبان های دستوری به دست می آید همچنین با ساختن دسته ای از توابع که از مجموعه ی معینی از اشیای داده استفاده می کنند قابلیت اطمینان و قابلیت انعطاف برنامه نویسی تابعی حاصل می شود. C++ و جاوا نمونه ای از زبان های شی گرا هستند.

نکته: می توان برنامه هایی به زبان prolog و lisp نوشت که به ترتیب اجرا می شود تا عمل خاصی را انجام دهند. همچنین می توان به زبان C برنامه ای نوشت که فقط از فراخوانی تابع تشکیل شده باشد و به این ترتیب مثل یک زبان تابعی به نظر آید. تکنیک های تابعی برای کنترل برنامه ها و صحت آن ها به وجود آمده است.

نکته : زبان C++ ترکیبی از یک زبان تابعی ، دستوری و شی گرا محسوب می شود.

8-1-استاندارد سازی زبان ها

```
Int I ;
I= (1 &&2 )+3
```

هنگام مشاهده یک دستور دریک زبان برای پی بردن به معنای دستور و خروجی از آن سه راه حل وجود دارد :

- مراجعه به مستندات زبان
- تایپ و اجرای برنامه روی کامپیوتر
- مراجعه به استاندارد زبان

برای مقابله با یک سری مشکلاتی که باعث ناسازگاری می شوند هر زبان دارای تعاریف استانداردی است. تمام پیاده سازی ها باید از این استاندارد پیروی کنند. استاندارد سازی یک زبان *قابلیت حمل و قابلیت انعطاف* آن را افزایش می دهد . زبانی که قابلیت حمل دارد ، قابلیت اطمینان را نیز افزایش می دهد ولی عکس آن لزوماًدرست نیست . استاندارد ها به دو دسته کلی تقسیم می شوند.

- **استاندارد های خصوصی** : که شرکت مالک سازنده ی آن زبان ، آن را معرفی می کند و این استاندارد برای زبان هایی که گسترده اند مناسب نیست.
- **استاندارد های عمومی** :که توسط سازمان های معروف نظیر , ANSI , IEEE ISO ارائه می شوند. برای استفاده موثر از استانداردها ، سه نکته باید مد نظر قرار گیرد. 1- زمان شناسی 2- اطاعت و پیروی 3- کهنگی و منسوخ شدن

1-8-1- زمان شناسی

عموماًبرنامه نویسان دوست دارند زبان ها هر چه زودتر استاندارد شوند تا پیاده سازی های ناسازگاری از آنها ارائه نگردد. مثلاًزبان فرترن خیلی دیر استاندارد شد و در زمان استانداردسازی آن نسخه های ناسازگار زیادی از آن وجود داشت. زبان ADA زودتر از پیاده سازی اش استاندارد شد. و زبان های C و پاسکال ، هنگامی استاندارد شدند که تنها نمونه های کمی از آن ها پیاده سازی شده بود.

1-8-2- اطاعت و پیروی

یعنی اینکه **برنامه ها باید از استاندارد پیروی کنند.** کامپایلر پیرو ، کامپایلری است که وقتی یک برنامه استاندارد به آن داده می شود نتیجه مشخصی *(درستی)را* میدهد. ممکن است زبان ها امکانات اضافی علاوه بر استاندارد داشته باشند که هیچ نتیجه ای برای آن مشخص نمی شود.

1-8-3- کهنگی و منسوخ شدن

اغلب ، هر استاندارد ، در طول 5 سال یکبار باید بازرسی و احیاناً بازسازی شود. در موارد زیادی استاندارد های جدید با استانداردهای قبلی سازگاری دارند ولی در مواردی هم ، یک ویژگی در استانداردهای بعدی یعنی حدود 5 تا 10 سال آینده حذف خواهد شد که به این مفهم کهنگی یا منسوخ شدن می گویند.