دانشگاه پیام نور
مرکز آران و بیدگل

طراحی و پیاده سازی

زبانهای برنامه‌سازی

گردآورنده :

مهندس مصطفی قبائی

ویراستار و صفحه آرا

مجید بصیرتی

بنام آنکه جان را فکرت آموخت

پیشگفتار

طراحی و پیاده سازی زبان های برنامه سازی یکی از دروس مهم و جذاب در مقطع کارشناسی کامپیوتر است. این درس به مفاهیم اساسی زبان های برنامه سازی پرداخته، نکات طراحی و شیوه های پیاده سازی آنها را مورد بررسی قرار می دهد.

در این جزوه تلاش شده است تمام سرفصل های درس پوشش داده شود. ذکر این نکته مهم ضروری است که این جزوه جهت کمک به دانشجویان در جهت کاهش یادداشت برداری، ترسیم شکل ها و مثال ها در هنگام تدریس بوده است. بنابراین در کنار این جزوه هر دانشجو، مطابق با ذوق و سلیقه خود جزوه ای دست نویس حاوی یادداشت هایی به منظور تکمیل و تفهیم مطالب این جزوه خواهد داشت.

در گردآوری این جزوه از کتاب اصول طراحی و پیاده سازی زبانهای برنامه سازی تالیف ترنس دبلیو.پرات و مارون وای زیکوویتز استفاده شده است. در ضمن پس از ارائه شرح دروس در هر فصل نمونه سوالات امتحانی مربوط به آن فصل ارائه شده است. در پایان نیز بخشی به عنوان پیوست در مورد سوالات کارشناسی ارشد از سال ۱۳۸۲ به بعد به این مجموعه اضافه شده است.

در اینجا لازم است از تمامی دانشجویانی که اینجانب را در تدوین این جزوه یاری نموده‌اند کمال تشکر را داشته باشم. همچنین در این جزوه احتمال اشتباه وجود دارد، به این جهت در مطالعه مطالب دقت کافی را داشته باشید و وجود هرگونه ایراد، و همچنین نظرات و پیشنهادات خود را از طریق پست الکترونیکی **mostafaghobaye@yahoo.com** به اطلاع اینجانب برسانید.

مصطفی قبائی آرانی

بهار ۹۰

تقدیم به:

دانشجویانی که عقلشان بر احساسشان غلبه می‌کند.

دانشجویانی که برای تحمل آرای دیگران تمرین می‌کنند.

دانشجویانی که برای چهل سال آینده خود برنامه دارند.

دانشجویانی که فرق هشت و هشت و یک دقیقه را می‌دانند.

دانشجویانی که رنگهای شاد خلقت را در ظاهر خود سپاس می‌گویند.

دانشجویانی که با محاسبه حروف اضافه سخن می‌گویند.

دانشجویانی که قاعده‌مند فکر می‌کنند.

دانشجویانی که برای هر سوالی چندین پاسخ متفاوت قائل اند.

دانشجویانی که عصبانیت خود را به تاخیر می‌اندازند.

دانشجویانی که شان را بر قدرت مقدم می‌شمارند.

دانشجویانی که در رفتار قابل پیش‌بینی‌اند.

دانشجویانی که برای افزایش قدرت کشور تامل می‌کنند.

دانشجویانیکه دغدغه‌های وفای به عهد، آنهارا از خواب بیدار می‌کند.

(دکتر محمود سریع القلم)

فهرست مطالب

فصل اول: اصول طراحی زبانها

فصل دوم :اثرات معماری ماشین

فصل سوم: اصول ترجمه زبان

فصل پنجم: انواع داده اولیه

فصل ششم: بسته بندی

فصل هشتم:کنترل ترتیب اجرا

فصل نهم: کنترل ترتیب زیربرنامه

فصل اول

اصول طراحی زبانها

آنچه در این فصل خواهید آموخت:

- دلایل مطالعه زبان‌های برنامه سازی
- تاریخچه ای از زبان‌های برنامه سازی
  - زبان‌های محاسباتی
  - زبان‌های تجاری
  - زبان‌های هوش مصنوعی
  - زبان‌های سیستمی
- تأثیر محیط بر روی طراحی وپیاده سازی زبان‌ها
  - محیط دسته ای
  - محیط محاوره ای
  - محیط سیستم‌های تعبیه شده
  - محیط کامپیوتر شخصی
  - محیط شبکه و اینترنت
- دامنه کاربرد زبان‌ها
- ویژگی‌های یک زبان خوب
- نحو و معنای زبان
- مدل‌های محاسباتی زبان
  - زبان‌های دستوری
  - زبان‌های تابعی
  - زبان‌های قانونمند
  - زبان‌های شی گرا
- استاندارد سازی زبان‌ها
  - زمان شناسی
  - اطاعت وپیروی
  - کهنگی ومنسوخ شدن
- سوالات تستی و تشریحی

قبل از اینکه به مفاهیم زبان های برنامه سازی بپردازیم، باید مقدماتی را مطالعه کنیم. ابتدا بررسی می کنیم که چرا دانشجویان و توسعه دهندگان نرم افزارهای حرفه ای، باید طراحی و ارزیابی زبان ها را مطالعه کنند. این بحث برای کسانی مفید خواهد بود که اعتقاد دارند آشنایی با یک یا دو زبان برنامه نویسی کافی است. هر گونه علامت گذاری جهت توصیف الگوریتم‌ها و ساختمان داده‌ها، یک زبان برنامه نویسی نامیده می شود. صدها زبان برنامه سازی مختلف طراحی و پیاده سازی شده اند. حتی در سال ۱۹۶۹ سامت، ۱۲۰ زبان مختلف را نام برد که به طور گسترده مورد استفاده قرار می‌گرفتند و از آن زمان تاکنون زبانهای دیگری طراحی و پیاده سازی شده‌اند.

۱ – ۱ – دلایل مطالعه زبان‌های برنامه سازی

اغلب برنامه‌نویسان ترجیح می‌دهند از یک یا دو زبان برنامه سازی استفاده کنند. پس چه لزومی دارد زبانهایی را مطالعه کنیم که ممکن است اصلا به کار گرفته نشوند؟ در زیر چند مورد از دلایل مطالعه زبان های برنامه سازی بیان شده است:

۱-۱-۱- افزایش توانایی‌های خود در توسعه الگوریتم‌های کارآمد

بسیاری از زبان‌ها امکاناتی دارند که اگر به خوبی مورد استفاده قرار گیرند به نفع برنامه نویس است و برنامه نویس با بکارگیری این امکانات می تواند الگوریتم های کار آمدی بنویسد، ولی اگر به طور نامناسب مورد استفاده قرار گیرند، وقت زیادی از برنامه نویس می‌گیرند. مثلاً اگر برنامه نویس از تکنیک بازگشتی یا مفاهیم شی گرایی، مزایا و مشکلات آن اطلاع داشته باشد، می-تواند تصمیم بگیرد چه موقعی از آن استفاده کند یا نکند.

۱-۱-۲- استفاده بهینه از زبان‌های برنامه نویسی موجود

با درک اینکه چگونه ویژگی‌ها و ساختارهای یک زبان برنامه نویسی پیاده سازی می‌شوند، توانایی شما در نوشتن برنامه‌های بهینه و کارا افزایش می‌یابد. به عنوان مثال اگر بدانید آرایه‌ها، رکوردها، لیست‌ها و رشته‌ها در زبان برنامه نویسی مورد نظرتان چگونه ایجاد و پیاده سازی می‌شوند آنگاه می‌توانید از زبان برنامه نویسی به طور بهینه بهره ببرید.

۱-۱-۳- آشنایی با اصطلاحات مفید ساختارهای برنامه نویسی

زبان‌ها هم می‌توانند به عنوان وسیله ای برای محدودیت تفکر و هم وسیله ای برای کمک به فکر کردن باشند. اگر برنامه نویس برای حل یک مساله، فقط با یک زبان آشنایی داشته باشد و فقط به توانایی‌های زبانی فکر کند که با آن آشناست آنگاه در برنامه نویسی با محدودیت مواجه خواهد شد. با مطالعه زبان‌های برنامه نویسی متعدد، دامنه لغات یک برنامه نویس افزایش می‌یابد. به عنوان مثال یک ساختار کنترلی به نام همروال[۱] در خیلی از برنامه‌ها مفید خواهد بود ولی زبان‌های محدودی این ساختار کنترلی را پشتیبانی می کنند. (مانند C و فرترن)

۱-۱-۴- انتخاب بهترین زبان برنامه نویسی

آگاهی از زبان‌های مختلف باعث می‌گردد که برای پروژه خاص بتوان زبان بهتری را انتخاب کرد به عنوان مثال کاربردهایی که نیاز به محاسبات عددی دارند بهتر است ازC، فرترن یا Ada استفاده کنند. برای کاربردهای هوش مصنوعی ازLisp، ML، Prolog و جهت کاربردهای اینترنت، Perl و Java مناسب هستند.

۱-۱-۵- یادگیری آسان تر یک زبان جدید

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

---

[۱] Coroutines

۱-۱-۶- ساده تر شدن طراحی یک زبان جدید

به دلیل آشنایی با مفاهیم مورد نیاز در زبان های برنامه نویسی مثل نیاز به انواع داده اولیه و ساختار یافته، نیاز به حلقه ها، دستورات شرطی و انتساب، برخی از برنامه نویسان خود را به عنوان طراح زبان می‌دانند. مثلا طراح یک واسط کاربر جهت نوشتن برنامه‌های بزرگی مثل ویرایشگر متن یا سیستم عامل، می‌بایست با موارد مشابهی که در طراحی زبان‌های برنامه نویسی همه منظوره مورد استفاده قرار می‌گیرند آشنایی داشته باشد.

۱-۲- تاریخچه زبان‌های برنامه نویسی

فرترن و Lisp در دهه ۱۹۵۰، پاسکال،Ada، Prolog و اسمالتاک در دهه ۱۹۷۰، Perl،C++ وML در دهه ۱۹۸۰ و Java دردهه ۱۹۹۰ طراحی شدند؛ زبان فرترن برای کارهای علمی ومحاسباتی ساخته شده بود. تاریخچه زبان های برنامه نویسی را بر اساس کاربرد آنها یعنی محاسباتی، تجاری، هوش مصنوعی و سیستمی مورد بررسی قرار می دهیم.

YEAR	LANGUAGE
1950-55	Assembly
1956-60	Fortran , Algol 58 , Algol 60 , Cobol , Lisp
1961-65	FortranIR , Cobol 61 , Algol 61 , Snobol , APL
1966-70	Fortran 66 , Cobol 65 , Snobol 4 , Simula 67 , Basic , APL 360
1971-75	Pascal , Cobol 74 , APL (standard) , C
1976-80	Ada , Fortran 77 (standard)
1981-85	Prolog
1985-90	C++ , Fortran 90

جدول ۱ - ۱

۱-۲-۱- زبان‌های محاسباتی (مبتنی بر اعداد)

این دسته از زبان ها باید محاسبات با حجم زیاد را با دقت اعشار بالا انجام دهند. بنابراین، اینگونه از زبانها باید به انواع توابع ریاضی کتابخانه ای مجهز شده باشند. فرترن و الگول نمونه هایی از زبان های محاسباتی هستند. درحدود سال ۱۹۶۰ زبان الگول به عنوان یک زبان محاسباتی آکادمیک معرفی شد. اهداف اصلی الگول در آن زمان شامل موارد زیر بود :

- این زبان باید به ریاضیات استاندارد نزدیک باشد.
- جهت توصیف الگوریتم‌ها مناسب باشد.
- برنامه‌ها باید به زبان ماشین ترجمه گردند.
- این زبان نباید وابسته به یک ماشین خاصی باشد.

۱-۲-۲- زبان‌های تجاری

این دسته از زبان ها برای کاربردهای تجاری، حسابداری، انبار و بورس... مورد استفاده قرار می گیرند. تسریع در ورود اطلاعات، نگهداری سابقه و خروجی های واضح در قالب نمودار، جدول و ... از نیازهای اصلی کاربران این زبان ها است. هدف از زبان های تجاری این بود که، برنامه‌هایی که ایجاد می‌شوند از متن‌هایی به شکل متن انگلیسی استفاده کنند. کوبول، زبانی جهت کاربردهای تجاری می‌باشد که ارتش آمریکا برای نگهداری سوابق و اطلاعات نظامی خود از این زبان استفاده می کند. هدف مهم درکوبول، نوشتن برنامه ای است که به زبان انگلیسی نزدیک باشد به طوریکه در این زبان پایان هر دستور با نقطه مشخص می شود. هر چند که کوبول قابلیت خوانایی بالایی دارد اما نحو رسمی ندارد و برنامه نویسی در آن دشوار است.

نکته: PL/I ویژگی‌های عددی فرترن را با خصوصیات برنامه سازی تجاری کوبول ترکیب کرد.

PL/I = Cobol + Fortran

۱-۲-۳- زبان‌های هوش مصنوعی

تمایل به زبان‌های هوش مصنوعی در دهه ۱۹۵۰ با زبان IPL شروع شد. لیسپ، به عنوان یک زبان تابعی پردازش کننده لیست طراحی شد. زبان لیسپ جهت کاربردهای هوش مصنوعی شامل ویژگی‌های زیر است:

- یک زبان تابعی پردازش کننده لیست است.
- عملیات جستجو را به خوبی انجام می‌دهد.
- پیاده سازی بازیهای کامپیوتری در آن به خوبی صورت می‌گیرد.
- قابلیت‌های مناسبی جهت پردازش متن دارد.
- رشته‌هایی از نمادها می‌توانند توسط رشته‌های دیگر جایگزین شوند (تفسیر ماشین خودکار).

نکته: لیسپ، جهت پردازش لیست همه منظوره طراحی شد ولی پرولوگ یک زبان تک منظوره است که ساختار آن بر اساس منطق ریاضی می باشد.

۱-۲-۴- زبان‌های سیستمی

این دسته از زبانها برای نوشتن نرم افزارهای سیستمی نظیر سیستم عامل و پیاده سازی کامپایلرها و ... کاربرد دارند. این گروه از زبان‌ها باید قادر به دستیابی به سخت افزار و ایجاد ارتباط با آن باشند مثلPL/I،C و اسمبلی.

۱-۳- تاثیر محیط اجرایی بر روی طراحی و پیاده سازی زبان‌ها

توسعه نرم افزار در خلا انجام نمی‌شود. سخت افزاری که زبان را پشتیبانی می‌کند تاثیر زیادی بر طراحی زبان برنامه نویسی دارد. محیطی (شامل سیستم عامل و سخت افزار)، که برنامه در آن ایجاد، تست و اشکال زدایی می‌شود، محیط میزبان و محیطی که برنامه بر روی آن اجرا می شود محیط عملیاتی (مقصد) نام دارد. در اغلب موارد محیط میزبان و محیط مقصد یکسان است اما گاهی اوقات ممکن است محیط عملیاتی با محیط میزبان متفاوت باشد. برای مثال برنامه هایی که با پسوند jar جهت اجرا بر روی یک موبایل در یک کامپیوتر ایجاد می شوند که در این مورد محیط عملیاتی، محیط موبایل و محیط کامپیوتر محیط میزبان است.

در ذیل به چند نمونه از این محیطها که توسعه و اجرای نرم افزار در آن‌ها صورت می‌گیرد اشاره می‌کنیم :

۱-۳-۱- محیط دسته ای[1]

در محیط های دسته ای، کاربر با برنامه هیچ گونه تعامل و محاوره ای ندارد و ترتیب اجرای برنامه در خود برنامه گنجانده شده است. در این محیط ها، کاربر در یک فاز، ورودی های مورد نیاز برنامه را وارد کرده و سپس منتظر می ماند تا اینکه برنامه پردازش شود و در نهایت خروجی بدست آید. برای زبان‌هایی که جهت محیط‌های دسته ای یا offline طراحی شده اند، فایل-ها متداول ترین ابزار جهت ورودی/خروجی هستند. برنامه، مجموعه ای از فایل‌های داده را به عنوان ورودی گرفته، داده‌های آنها را پردازش کرده و مجموعه ای از فایل‌های داده خروجی را تولید می‌کند. این محیط عملیاتی را پردازش دسته ای می‌گویند زیرا اطلاعات ورودی در فایل‌ها دسته بندی می‌شوند و به صورت دسته ای پردازش می‌شوند. در این محیط خطایی که اجرای برنامه را خاتمه دهد قابل اصلاح بوده ولی هزینه بر است، زیرا پس از پردازش و تصحیح خطا، برنامه مجددا باید به طور کامل اجرا شود. ویژگی دیگر محیط دسته ای، عدم محدودیت زمانی برای برنامه است یعنی زبان استفاده شده برای این محیط، امکاناتی را جهت تاثیر بر روی سرعت اجرای برنامه در اختیار نمی‌گذارد به عبارتی دیگر، زبان استفاده شده در محیط دسته ای

[1] Batch

محدوده زمانی مشخصی ندارد. زبان‌هایی مثل فرترن، کوبول و پاسکال ابتدا برای محیطهای دسته ای طراحی شدند ولی امروزه در محیطهای محاوره یا سیستم تعبیه شده نیز به کار می‌روند.

۱-۳-۲- محیط محاوره ای[1]

در این محیط یک برنامه مستقیماً با کاربر تعامل دارد و خروجی در نمایشگر نشان داده می‌شود. اینگونه محیطها، از سیستم اشتراک زمانی برای انجام کارهای مختلف، که در یک زمان واحد به کامپیوتر داده می‌شود استفاده می‌کنند؛ به این صورت که به هر برنامه یک برش زمانی[2]اختصاص داده می‌شود. بعد از اتمام این برش زمانی، پردازنده به برنامه دیگری که در حال اجراست داده می‌شود. 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. زبان‌هایC،C++ برای نوشتن برنامه‌های محاوره ای مناسب هستند.

۱-۳-۳- محیط سیستم‌های تعبیه شده (توکار)[3]

به سیستم کامپیوتری که جهت کنترل بخشی از یک سیستم بزرگ مثل هواپیما یا ماشین آلات صنعتی استفاده می‌شود، سیستم کامپیوتری تعبیه شده یا توکار گفته می‌شود. در محیطهای دسته ای و محاوره ای خرابی سیستم چندان اهمیت ندارد، ولی در سیستم توکار، خرابی سیستم ممکن است زیان‌های جدی به بار آورد. سیستم‌های توکار، معمولا به صورت بلادرنگ[4] هستند یعنی در یک محدوده زمانی از قبل تعیین شده باید به ورودی‌ها پاسخ دهند. از دیگر ویژگی های سیستم های تعبیه شده می توان به موارد زیر اشاره کرد:

- برنامه‌های نوشته شده برای این محیطها معمولاً بدون سیستم عامل، بدون سیستم فایل و بدون دستگاه‌های I/O اجرا می‌شوند. در عوض هر کدام از برنامه‌ها رویه‌های مخصوصی برای ارتباط با دستگاه‌های ورودی/ خروجی سیستم بزرگ دارند و به تعامل با آنها می‌پردازند.
- قابلیت اطمینان[5] در این محیط ها از اهمیت خاصی برخوردار است.
- در این نوع سیستم‌ها، اداره خطاها و استثناها باید به خوبی انجام گیرد. خاتمه برنامه به جز در موارد خرابی کلی سیستم، قابل قبول نیست و کاربری وجود ندارد که به صورت محاوره ای خطاها را برطرف سازد.
- در سیستم‌های توکار اغلب از کامپیوترهای RISC که تعداد دستورات محدودی دارند استفاده می‌شود. زبان‌های C،Ada و C++ برای نوشتن برنامه‌های مربوط به سیستم توکار مفید هستند.

۱-۳-۴- محیط کامپیوتر شخصی

در موارد زیادی در محیط کامپیوترهای شخصی، کارایی، کمتر مد نظر قرار می‌گیرد و هدف اصلی اغلب، راحتی کاربر و داشتن یک محیط گرافیکی ساده است. نمونه مشهور آن برنامه نویسی تحت محیط ویندوز است. برنامه نویسی شیءگرا، مدل مناسبی برای چنین محیط هایی است و زبان‌هایی مثلC++ یا Java در چنین محیطی کاربرد زیادی دارند.

---

[1] interactive
[2] time slice
[3] Embedded system
[4] Real time
[5] Reliability

۱-۳-۵- محیط شبکه و اینترنت

اینترنت اولیه، دو سرویس FTP و Telenet داشت. در پروتکل Telenet، کاربر به عنوان بخشی از کارگزار راه دور عمل می‌کرد و به کمک FTP می‌توانست فایل‌هایی را از ماشین کارگزار[۱] گرفته و یا به آن بفرستد. 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 زبان‌ها باید تعامل بین کامپیوترهای Server و Client را امکان پذیر سازند.

۱-۴- دامنه کاربرد زبان‌ها

در دهه ۱۹۶۰ اغلب برای کاربردهای تجاری از زبان کوبول، برای امور علمی از فرترن و الگول، برای کاربردهای سیستمی از اسمبلر یا فورث (Forth) و برای کاربردهای هوش مصنوعی ازLisp و Snobol استفاده می‌شد.

امروزه برای کاربردهای تجاری از کوبول، برای صفحه گسترده‌ها از زبان‌های نسل چهارم (4GL)، C++ و Java، برای کاربردهای علمی از فرترن، C ،C++ و Java، برای کاربردهای سیستمی از C، C++و Java، برای کاربردهای هوش مصنوعی، از لیسپ و پرولوگ استفاده می‌شود.

برنامه‌های هوش مصنوعی با الگوریتم‌هایی نوشته می‌شوند که عمل جستجو را در بخش بزرگی از داده‌ها انجام می‌دهند. مثلاً برای شطرنج، کامپیوتر حرکت‌های زیادی را ایجاد کرده و آنگاه بهترین حرکت را انتخاب می‌کند. در دنیای اینترنت، زبان‌هایی مثل Perl و جاوا اسکریپت، به سرور امکان می‌دهند که داده‌ها را از کاربر گرفته و تراکنشی را انجام دهند. امروزه بسیاری از دستگاه‌ها مثل خودروها و تلویزیون‌های دیجیتالی، پردازنده دارند و این وسایل به زبان‌های بی درنگ[۲] نیاز دارند که C وC++ و Ada در این موارد کاربرد دارند. ML، در تحقیقات زبان‌های برنامه سازی به کار گرفته شده است. هر چند اسمالتاک زبان معروفی نیست، ولی خیلی از خصوصیات شیءگرایی C++، از اسمالتاک گرفته شده است. جدول زیر کاربرد زبان های برنامه نویسی در زمان گذشته و حال را نشان می دهد.

دوره	کاربرد	زبان مورد استفاده
۱۹۶۰	تجاری	Cobol
	علمی	Fortran,Basic,Algol,APL
	سیستمی	Assembly
	هوش مصنوعی	Lisp,Snobol
امروزه	تجاری	C++,Java,4GL
	علمی	Java,C,C++,Basic
	سیستمی	C,C++,Java
	هوش مصنوعی	Lisp,Prolog
	انتشارات	Tex,Postscript

جدول ۱ – ۲دامنه کاربرد زبانها

[۱] Server
[۲] Real time

۱-۵- ویژگی‌های یک زبان خوب

برای ارزیابی زبان های برنامه نویسی به معیارهایی نیاز داریم. در زیر به چند مورد ار ویژگی های یک زبان خوب اشاره می کنیم:

۱-۵-۱- 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 می‌گویند.

نحو یک زبان روی نوشتن، تست کردن، اصلاح و درک زبان اثرگذار است. قابلیت خوانایی برنامه‌ها نیز یک اصل مهم می‌باشد. نحوی که مختصر و رمزگونه باشد برنامه نویسی را کوتاه تر و ساده تر می‌کند، ولی قابلیت خوانایی را کاهش می‌دهد. مثلاً برنامه‌های APL حالت رمز گونه دارند و قابلیت خوانایی کمی دارند. بسیاری از زبان‌ها، ساختارهای نحوی دارند که دو جمله تقریباً مشابه، معانی مختلفی دارند و این باعث کاهش قابلیت خوانایی می شود. مثلا کاراکتر فضای خالی (space)b در زبان Snobol4 در یکجا به عنوان جدا کننده و در جایی دیگر به عنوان الحاق کننده دو رشته به کار می‌رود و این باعث پیدایش ابهام در بعضی جملات می شود.

۱-۵-۲- قابلیت تعامد[۱]

منظور از تعامد این است که بتوان خصوصیات مختلفی از یک زبان را با هم ترکیب کرد و این ترکیب حاصل نیز 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.

۱-۵-۳- طبیعی بودن برای کاربردها

هر زبان در هنگام استفاده در کاربرد خاص خود باید مناسب به نظر آید یعنی با توجه به کاربرد آن زبان، امکانات آن نیز موجود باشد. زبان‌هایی که برای کاربردهای خاصی هستند این خصوصیت را در همان زمینه کاربردی دارا می‌باشند. مثلاً برای محاسبات علمی و فنی از فرترن و برای پردازش رشته‌ها از زبان Lisp یا Snobol4 استفاده می‌کنیم.

۱-۵-۴- پشتیبانی از انتزاع[۲]

در هر زبان برنامه نویسی تعداد معینی نوع داده اولیه وجود دارد اما امکان دارد برنامه نویس یک نوع داده ای جدید (انتزاعی) را نیاز داشته باشد که در زبان برنامه نویسی وجود ندارد. مثلاً در زبان C و پاسکال، نوع داده اعداد مختلط و اعمال روی آنها وجود ندارد. در زبان‌هایی مانند C++ و Ada، برنامه نویس می‌تواند نوع داده‌های انتزاعی (ADT) مورد نظر خود را تولید کند. مثلاً برنامه نویس می‌تواند نوع داده اعداد مختلط و عملیات جمع، تفریق و ضرب را بر روی آنها را با استفاده از مفهوم Class در زبان C++ و Package در زبان Ada تعریف کند. استفاده از جنبه‌های انتزاعی، قابلیت اطمینان را افزایش خواهد داد.

۱-۵-۵- سهولت در بازرسی برنامه

برنامه‌ها باید به گونه ای باشند که بررسی درستی و صحت عملکرد آنها ساده باشد برنامه‌هایی که ساختار نحوی و معنایی ساده تری دارند بازرسی آنها نیز ساده تر می باشد. بررسی درستی برنامه به دو صورت رسمی مانند روش‌های ریاضی و غیر رسمی نظیر تست برنامه با ورودی‌های مختلف انجام می‌شود.

---

[۱] Orthogonality
[۲] Abstraction

۱-۵-۶- محیط برنامه نویسی

یک محیط برنامه نویسی قوی، ایجاد برنامه‌ها 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 برنامه‌های ویژوال عموماً یک محیط برنامه نویسی قدرتمندی دارند. اسمالتاک زبانی است که محیط آن دارای پنجره‌ها، منوها، ورودی موس و ... برای کار کردن روی برنامه‌ها می باشد.

۱-۵-۷- قابلیت حمل[۱]

یک زبان خوب باید بتواند بر روی سیستم‌های مختلف کامپیوتری به سادگی انتقال یافته و اجرا شود. یکی از معیارهای مهم برای پروژه‌های برنامه نویسی قابلیت انتقال یک برنامه از یک ماشین به ماشین دیگر است. زبانی که به ماشین خاصی وابسته نباشد برنامه‌های نوشته شده در آن زبان، از ماشینی به ماشین دیگر قابل انتقال هستند. C،Ada،Fortran و پاسکال تعاریف استانداری برای تولید برنامه‌های قابل حمل دارند. زبان جاوا این ویژگی را در حد بسیار بالایی دارد. با افزایش قابلیت حمل، انعطاف پذیری و کارآیی برنامه کاهش می یابد.

۱-۵-۸- هزینه استفاده

هزینه، عنصر مهمی در ارزیابی زبان‌های برنامه نویسی است. زبان برنامه نویسی خوب است که برنامه های نوشته شده به آن زبان هزینه زیادی را به کامپیوتر ما تحمیل نکند. ۳ معیار اصلی هزینه عبارت اند از:

الف) هزینه اجرای برنامه: زمان اجرای یک برنامه خصوصاً برنامه‌های بزرگی که به دفعات زیادی اجرا می شوند معیاری مهم می باشند هزینه اجرای برنامه شامل استفاده از منابع کامپیوتری خصوصاً CPU،RAMوDisk می باشد.

ب) هزینه ترجمه برنامه: یعنی مدت زمانی که طول می کشد تا برنامه کامپایل شود مثلاً برنامه‌های دانشجویان، چندین بار ترجمه شده و کمتر اجرا می شوند و این هزینه مهمتر است (چون ممکن است خطا داشته باشد و باید چندین بار ترجمه کنید تا کل خطاها برطرف شود) هزینه ترجمه شامل استفاده از منابع کامپیوتری خصوصاً CPU ,Disk ،RAM می باشد.

ج) هزینه نگهداری برنامه: شامل هزینه تطبیق برنامه با جوانب محیطی، ایجاد امکانات جدید در نرم افزار و برطرف کردن اشکالات است.

۱-۶- نحو و معنای زبان

نحو[۲] 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[۳] زبان، همان مفهومی است که به ساختارهای نحوی زبان داده می شود.

مثال:

نحو: 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 ۱۰ خانه برای اعداد صحیح در نظر گرفته شود.

---

[۱] Port ability
[۲] Syntax
[۳] Semanatic

۱-۷- مدلهای محاسباتی زبان

مدل محاسباتی چگونگی توصیف یک برنامه را در یک زبان برنامه نویسی توصیف می کند. چهار مدل محاسباتی عبارتند از:۱- دستوری ۲- تابعی ۳- قانونمند ۴- شیء گرا

۱-۷-۱- زبان‌های دستوری[۱]

در این زبان‌ها 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 جدیدی وارد کند نحو چنین زبان‌هایی 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 زبان‌های قدیمی مانند C، C++، Fortran، PL/I،Algol، پاسکال، Ada و Cobol از این مدل پشتیبانی می کنند. زبان‌های امروزه مانند C ، C++ وکوبول نیز از این مدل پشتیبانی می کنند.

۱-۷-۲- زبان‌های تابعی[۲]

در این زبان‌ها برنامه شامل مجموعه ای از توابع است و به جای دنبال کردن تغییر حالت ماشین، عملکرد برنامه دنبال می شود؛ یعنی به جای آنکه داده‌های موجود را در نظر بگیریم، نتیجه مطلوب را در نظر خواهیم داشت. در صورت برقراری عملیات بر روی داده ورودی، نتیجه مطلوب حاصل می شود عملی باید روی حالت اولیه ماشین انجام پذیرد تا با دستیابی به داده اولیه و ترکیب آن‌ها 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 داده‌های 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 :

$f_n (\dots f_2 (f_1 (data)) \dots)$

زبان‌های ML و Lisp، زبان‌های تابعی هستند.

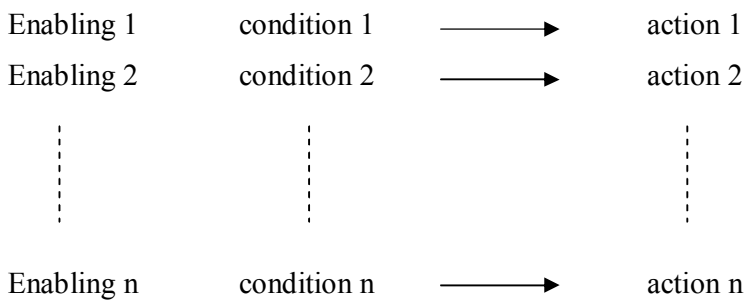
۱-۷-۳- زبان‌های قانونمند[۳]

در این زبان‌ها برنامه شامل مجموعه ای از قوانین است که هر قانون ساختاری مشابه if در زبان های برنامه نویسی دارد. در زبان‌های 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.

نحو کلی چنین زبان‌هایی به شکل زیر است:

---

[۱] Imperative
[۲] apllicative
[۳] Rule-Based



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 زبان‌ها اغلب در برنامه‌های کاربردی هوش مصنوعی و سیستم‌های خبره مورد استفاده قرار می گیرند.

۱-۷-۴- زبان‌های شئ گرا[۱]

در این زبان‌ها برنامه شامل مجموعه ای از ماژول ها (کلاس ها) که هر ماژول شامل تعریف داده ها و عملیات روی آنها است. در زبان‌های شئ گرا، اشیاء داده ای پدید آمده و مجموعه ای از توابع تعریف می شوند تا روی این داده‌ها کار کنند اشیاء پیچیده را می توان با بسط اشیای ساده و مفهوم ارث بری ایجاد کرد. در برنامه نویسی شئ گرا، با ساختن اشیای داده ای دقیق، کارایی زبان‌های دستوری به دست می آید همچنین با ساختن دسته ای از توابع که از مجموعه معینی از اشیا داده استفاده می کنند قابلیت اطمینان[۲] و قابلیت انعطاف[۳] برنامه نویسی تابعی حاصل می شود. C++,Java,Smalltalk نمونه هایی از زبان‌های شیء گرا هستند.

نکته: می توان برنامه‌هایی به زبان Lisp و Prolog نوشت که به ترتیب اجرا می شوند 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. تکنیک‌های تابعی، برای کنترل برنامه‌ها و صحت آن‌ها به وجود آمده است.

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

بنابراین به طور خلاصه خواهیم داشت:

• ***زبانهای دستوری یا رویه ای (Imperative)***
برنامه یک سری از دستورات پشت سرهم است که از بالا به پایین اجرا می‌شوند. Statement 1; Statement 2;
مانند : C, C++, Fortran, Algol, Cobol, PL/I, Ada, Smalltalk, Pascal

• ***زبانهای کاربردی یا تابعی (Functional)***
توابع در کنار هم برنامه را تشکیل می‌دهند و قابلیت فراخوانی تودرتو دارند. $Function_n(\ldots Function_1\ (data)\ldots)$
مانند: Schema, ML, Lisp

[۱] Object Orinted
[۲] Reliability
[۳] Flexibility

• ***مدل مبتنی برقانون (Rule-Base)***
برنامه‌ها به صورت مجموعه ای از قوانین پایه ای تجزیه مساله ، می‌باشند. ساختمان این زبان مشابه ساختار if است که در صورت رخداد شرایطی ،قانونی اجرا می‌شود. Enablingcondition1 → action1 = Action1 if enabling condition1
مانند : Prolog

• ***زبانهای شیء گرا (Object Oriented)***
برنامه از تعدادی ماژول تشکیل شده است که هر ماژول مشابه یک برنامه در زبان Cمی‌باشد. مبحث اصلی در این نوع زبان پشتیبانی از کلاس است.
مانند : C++, Java, Visual

جدول ۱ - ۳

۱-۸- استاندارد سازی زبان‌ها

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

هنگام مشاهده یک دستور مانند دستور فوق در یک زبان، برای پی بردن به معنای دستور و خروجی آن سه راه حل وجود دارد:

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

برای مقابله با یکسری مشکلاتی که باعث ناسازگاری می شوند هر زبان دارای تعاریف استانداردی است. تمام پیاده سازی‌ها باید از این استاندارد پیروی کنند. استاندارد سازی یک زبان، قابلیت حمل و قابلیت انعطاف آن را افزایش می دهد. استانداردها به دو دسته کلی تقسیم می شوند.

- **استانداردهای خصوصی:** که شرکت سازنده زبان برنامه نویسی، آن را معرفی می کند. این استاندارد برای زبان-هایی که گسترده اند مناسب نیست.
- **استانداردهای عمومی:** که توسط سازمان‌های معروف نظیرISO، ANSIIEEE ارائه می شوند.

برای استفاده موثر از استانداردها، سه نکته باید مد نظر قرار گیرد : ۱-زمان شناسی ۲- اطاعت و پیروی ۳- کهنگی و منسوخ شدن

۱-۸-۱- زمان شناسی[۱]

منظور از زمان شناسی این است که چه موقع برای یک زبان برنامه نویسی استاندارد نوشته شود. عموماً برنامه نویسان دوست دارند زبان‌ها هرچه زودتر استاندارد شوند تا پیاده سازی‌های ناسازگاری از آنها ارائه نگردد. مثلاً زبان فرترن، خیلی دیر استاندارد شد و در زمان استانداردسازی آن، نسخه‌های ناسازگار زیادی از آن وجود داشت. زبان Ada زودتر از پیاده سازی اش استاندارد شد و زبان‌های C و پاسکال، هنگامی استاندارد شدند که، تنها نمونه‌های کمی از آنها پیاده سازی شده بود.

[۱] TimeLines

۱-۸-۲- اطاعت وپیروی[۱]

یعنی اینکه برنامه‌ها باید از استاندارد پیروی کنند.کامپایلر پیرو یک زبان، فقط برنامه های نوشته شده مطابق با استاندارد آن زبان را ترجمه می کند و به ازای برنامه های استاندارد، خروجی های مناسب تولید می کند. ممکن است زبان‌ها امکانات اضافی علاوه بر استاندارد داشته باشند که هیچ نتیجه ای برای آن مشخص نمی شود.

۱-۸-۳- کهنگی ومنسوخ شدن[۲]

اغلب، هر استاندارد، در طول ۵ سال یکبار باید بازرسی و احیاناً بازسازی شود. درموارد زیادی استانداردهای جدید با استانداردهای قبلی سازگاری دارند ولی در مواردی هم، یک ویژگی دراستانداردهای بعدی یعنی حدود ۵ تا ۱۰ سال آینده حذف خواهد شد که به این مفهوم کهنگی یا منسوخ شدن می گویند.

[۱] Conformance
[۲] Obsolescence

۱-۹- سوالات فصل اول

سوالات تستی

۱- کدامیک از موارد زیر از صفات یک زبان خوب نمی باشد؟(نیمسال دوم ۸۳)

الف. وضوح، سادگی و یکپارچگی ب. عدم قابلیت تعامد

ج. پشتیبانی از انتزاع د. طبیعی بودن برای کاربردها.

۲- کدامیک از زبانهای زیر یک زبان تجاری نمی باشد؟(نیمسال دوم ۸۳)

الف.COBOL ب.PROLOG ج.CBL د.FLOMATIC

۳- مسائلی که برای استفاده موثر از استانداردها بایستی در نظر گرفته شود عبارتند از: (نیمسال دوم ۸۳)

الف. کهنگی، زمان شناسی، اطاعت و پیروی ب. اطاعت و پیروی، هزینه، سادگی

ج. کهنگی، هزینه، سادگی طراحی د. زمان شناسی، سادگی، سرعت

۴-کدام گزینه جزء اهداف الگول نیست؟ (نیمسال دوم ۸۴)

الف. برای توصیف الگوریتم‌ها مفید باشد. ب. بایستی به ریاضیات استاندارد نزدیک باشد.

ج. بایستی به معماری یک ماشین مقید باشد. د. برنامه‌ها بایستی به زبان ماشین ترجمه شوند.

۵-در مورد LISP کدام گزینه غلط می‌باشد؟(نیمسال دوم ۸۴)

الف. به عنوان یک زبان پردازش کننده لیست طراحی شد.

ب. برای کارهای جستجو اصلاً مناسب نیست.

ج. بازیهای کامپیوتری در LISP به خوبی پیاده سازی می‌شوند.

د. هیچکدام.

۶-کدام گزینه صحیح است؟ (نیمسال دوم ۸۴)

الف. پردازش خطا در محیط محاوره‌ای با محیط دسته‌ای تفاوتی ندارد.

ب. برنامه‌های محاوره‌ای باید از محدودیت‌های زمانی برخوردار باشند.

ج. پردازش خطا در سیستم‌های تعبیه شده از اهمیت ویژه‌ای برخوردار نیست.

د. کلیه موارد بالا.

۷-به منظور استفاده از استانداردها کدام گزینه بایستی مدنظر قرار گیرد؟(نیمسال دوم ۸۴)

الف. زمان شناسی ب. کهنگی ج. اطاعت و پیروی د. تمام موارد

۸- IPL به عنوان اولین زبان مربوط به کدام گروه از کاربردها مطرح شد؟(نیمسال اول ۸۵-۸۶)

الف.تجاری ب.علمی ج. هوش مصنوعی د.سیستمی

۹–کدام مورد از دلایل مطالعه زبانهای برنامه سازی می باشد؟(نیمسال اول ۸۶–۸۷)

الف.استفاده بهینه از زبان برنامه سازی موجود

ب.شناختن ساختارهای مفید زبانهای برنامه نویسی

ج.انتخاب بهترین زبان برنامه سازی برای یک پروژه خاص

د. همه موارد صحیح است.

۱۰–اهداف ALGOL عبارتند از:(نیمسال دوم ۸۵–۸۶)

الف.نشانه‌ها بایستی به ریاضیات استاندارد نزدیک باشد.

ب.بایستی برای توصیف الگوریتم‌ها مفید باشد.

ج.نبایستی به معماری یک ماشین مقید باشد.

د.کلیه موارد بالا

۱۱–انواع زبانها عبارتند از:(نیمسال دوم ۸۵–۸۶)

الف.مبتنی بر اعداد ب.تجاری و هوش مصنوعی

ج.سیستم د.کلیه موارد بالا

۱۲–کدام گزینه غلط می باشد؟(نیمسال دوم ۸۵–۸۶)

الف.کاربردهای تجاری اسمالتاک زیاد نیست. ب.اسمالتاک خاصیت شی گرایی دارد.

ج.هوش مصنوعی از C استفاده می نماید. د. هیچکدام

۱۳–مدلهای مختلف محاسباتی زبان کدامند؟(نیمسال دوم ۸۵–۸۶)

الف.تابعی ، دستوری ب.مبتنی بر قاعده ، شی گرا

ج.الف و ب د.بی قاعده ، شی گرا ، تابعی، دستوری

۱۴–برای استفاده موثر از استانداردها کدام گزینه بایستی مد نظر باشد؟(نیمسال دوم ۸۵–۸۶)

الف.زمان شناسی ب.اطاعت و پیروی

ج.الف و ب د.تازگی

۱۵–قابلیت تعامد به چه معناست؟(نیمسال اول ۸۶–۸۷)

الف.خصوصیتی در زبانهاست که باعث می شود بتوان برنامه‌های نوشته شده در یک زبان را از ماشینی به ماشین دیگر منتقل کرد.

ب.اگر بتوانیم خصوصیات مختلف از یک زبان را با هم ترکیب کنیم و ترکیب حاصل نیز با معنا باشد.

ج.اگر بتوانیم ویژگیهای دو زبان مختلف را با هم ترکیب کنیم و یک زبان جدید ایجاد کنیم.

د.موارد ب و ج صحیح است.

۱۶–در کدام گزینه هر دو زبان، زبان‌های تابعی هستند؟(نیمسال دوم ۸۶–۸۷)

الف.ML, LISP ب.C++, FORTRAN

ج.COBOL, PROLOG د.LISP,PROLOG

۱۷–کدام یک از زبان‌های زیر خیلی زود استاندارد شد؟(نیمسال دوم ۸۶–۸۷)

الف.ADA ب. C ج.FORTRAN د.هیچکدام

۱۸–کدامیک از موارد زیر از اهداف الگول نمی باشد؟(نیمسال اول ۸۷–۸۸)

الف.نشانه‌های الگول باید به ریاضیات استاندارد نزدیک باشد.

ب.الگول باید برای توصیف الگوریتمها مفید باشد.

ج.برنامه‌ها در الگول باید به زبان ماشین نزدیکتر باشد.

د.الگول باید به معماری یک ماشین مقید باشد.

۱۹–تعریف زیر معرف کدامیک از محیطهای عملیاتی می باشد. (نیمسال اول ۸۷–۸۸)

«یک سیستم کامپیوتری که برای کنترل بخشی از یک سیستم بزرگ مثل ماشین آلات صنعتی، هواپیما ، ماشین تراش , اتومبیل یا مترو بکار می رود»

الف.دسته ای ب.محاوره ای ج.اشتراک زمانی د.سیستم تعبیه شده

۲۰–با توجه به سه گزاره زیر کدامیک از گزینه‌ها صحیح است؟(نیمسال اول ۸۷–۸۸)

مورد اول: برنامه شی گراء با ساختن اشیای داده دقیق کارایی زبانهای دستوری را بدست می آورد.

مورد دوم: برنامه شی گراء با ساختن دسته ای از توابع که از مجموعه ای محدود از اشیای داده استفاده می کنند قابلیت انعطاف زبانهای تابعی را بدست می آورد.

مورد سوم: برنامه شی گراء با ساختن دسته ای از توابع که از مجموعه ای محدود از اشیای داده استفاده می کنند قابلیت اعتماد زبانهای تابعی را بدست می آورد.

الف.موارد اول و دوم ب. موارد اول و سوم ج.موارد دوم و سوم د.هر سه مورد

۲۱–در کدام گزینه هر سه زبان در کاربردهای تصمیم گیری مثل هوش مصنوعی استفاده می شود؟(نیمسال دوم ۸۷–۸۸)

الف. C++,Ada,Smalltalk ب.Lisp,Prolog,ML

ج.C++,Lisp,Java. د.APL,XML,PERL

۲۲–یکی از اهداف زبان الگول نزدیک شدن به ریاضیات محض بود, این هدف موجب می شود تا ارسال پارامتر به زیر برنامه-ها به کدام روش صورت گیرد؟(نیمسال دوم ۸۷–۸۸)

الف. فراخوانی با نام ب.فراخوانی با نام ج. فراخوانی با ارجاع د. فراخوانی با مقدار– نتیجه

۲۳–در کدامیک از محیطهای برنامه سازی در ارتباط با I/O ، محدودیت‌های بیشتری نسبت به سایر محیطها وجود دارد؟(نیمسال اول ۸۸–۸۹)

الف.محیطهای محاوره ای ب.محیطهای دسته ای

ج.محیطهای سیستم تعبیه شده د.محیطهای اینترنتی

۲۴–کدام یک از موارد زیر در مورد سیستم‌های تعبیه شده(Embedded System) صحیح می باشد؟(نیمسال دوم ۸۷–۸۸)

مورد اول : برنامه‌ها در این سیستم‌ها،معمولاً بدون سیستم عامل مربوطه و بدون محیط معمولی فایلها و دستگاههای I/O اجرا می شوند.

مورد دوم: سیستم‌های تعبیه شده معمولاً در زمان بی درنگ کار می کنند.

مورد سوم: قابلیت اعتماد و صحت از اهمیت چندانی در این سیستم‌ها برخوردار نیستند.

الف. مورد اول و دوم ب. مورد دوم و سوم ج. مورد اول و سوم د. هر سه مورد

۲۵–کدام یک از موارد زیر در مورد برنامه‌های شی گراء صحیح است؟(نیمسال دوم ۸۷–۸۸)

مورد اول: با ساختن اشیاء داده دقیق قابلیت انعطاف و قابلیت اعتماد برنامه نویسی تابعی حاصل می شود.

مورد دوم: با ساختن دسته ای از توابع به همراه مجموعه ای محدود از فضای اشیای داده کارایی زبانهای دستوری حاصل می شود.

مورد سوم: ابتدا مجموعه ای از توابع طراحی می شود و سپس اشیای داده پیچیده ای حاصل می شوند.

الف. هر سه مورد ب. موارد اول و دوم ج. موارد دوم و سوم د. هیچکدام از موارد

۲۶–کدام یک از موارد زیر جز اهداف مطالعه طراحی و پیاده‌سازی زبانهای برنامه‌سازی نمی‌باشد؟ (تابستان ۸۸)

الف. استفاده بهینه از زبانهای برنامه‌سازی موجود ب.آشنایی با اصطلاحات مفید ساختارهای برنامه‌نویسی.

ج. فراگیری برنامه نویسی شی‌گرا. د. انتخاب بهترین زبان برنامه نویسی.

۲۷–برای زبان PL/I از ترکیب کدام زبانها استفاده شده است. (تابستان۸۸)

الف. C, Ada ب. Cobol, Fortran

ج. Ada, Lisp د. C, Fortran

۲۸–کدام یک از گزینه‌های زیر در مورد سیستم‌های تعبیه شده صحیح نمی‌باشد؟ (تابستان ۸۸)

الف: یک سیستم کامپیوتری تعبیه شده اغلب یک سیستم توزیعی است.

ب: قابلیت اعتماد و صحت، صفات مهمی برای برنامه‌های تعبیه شده است.

ج: C++, Ada برای نوشتن برنامه‌های تعبیه شده مفید می‌باشد.

د: پردازش خطا در سیستم‌های تعبیه شده از اهمیت ویژه‌ای برخوردار نمی‌باشد.

۲۹–کدام یک از موارد زیر جز صفات یک زبان خوب، نمی‌باشد. (در اولویت آخر قرار دارد) (تابستان ۸۸)

الف. قابلیت تعامد ب. طبیعی بودن برای کاربردها

ج. قابلیت گرافیکی د. پشتیبانی از انتزاع

۳۰–کدام یک از موارد زیر جز مدلهای محاسباتی زبان‌های برنامه‌سازی نمی‌باشد؟ (تابستان ۸۸)

الف. تابعی ب. مبتنی برقاعده ج. محاوره‌ای د. شی‌گرا

۳۱–کدام مورد جزء دلایل مطالعه زبانهای برنامه‌سازی نمی‌باشد. (تابستان ۸۸)

الف. استفاده بهینه از زبان‌های برنامه‌نویسی موجود ب. شناختن ساختارهای مفید زبان‌های برنامه‌نویسی

ج. انتخاب بهترین برنامه‌سازی برای یک پروژه خاص د. همه موارد فوق صحیح است.

۳۲–کدام یک از زبان‌های زیر جزء زبانهای پردازشی می‌باشد. (تابستان ۸۸)

الف. فرترن ب. پرل ج. پاسکال د. کوبول

۳۳–زبان برنامه نویسی پرولوگ از نظر کاربرد جزء کدام یک از زبان‌های برنامه سازی محسوب می شود؟(نیمسال اول ۸۸–۸۹)

الف. سیستمی ب.تجاری ج.علمی د.هوش مصنوعی

۳۴–منظور از قابل تعامد بودن زبان برنامه سازی چیست؟(نیمسال اول ۸۸–۸۹)

الف.یعنی امکان تجزیه ویژگی‌های مشابه زبان وجود داشته باشد.

ب.از ترکیب ویژگی‌های مختلف ترکیب جدید با معنایی ایجاد شود.

ج.از تجزیه ویژگی‌های مشابه ویژگی جدید با معنایی ایجاد شود.

د.ترکیب ویژگی‌های مختلف جهت ایجاد ترکیب جدید میسر نباشد.

۳۵–کدام دسته از مدل‌های زبان برنامه سازی به مدل منطقی نزدیکترند. (نیمسال اول ۸۸–۸۹)

الف.زبان‌های دستوری ب.زبان‌های تابعی ج. زبان‌های قانونمند د. زبان‌های شی گرا

۳۶– منظور از Orthogonality یا خاصیت تعامد در زبانهای برنامه سازی کدام است؟ (نیمسال دوم ۸۸–۸۹)

الف.خلاصه سازی چند ویژگی از یک زبان، که خلاصه سازی با معنا باشد.

ب. مجزاسازی یک ویژگی از زبان به چند ویژگی، که چند ویژگی جدید با معنا باشند.

ج.ترکیب کردن چند ویژگی از یک زبان، که ترکیب جدید بامعنا باشد.

د. ترکیب کردن چند ویژگی ازچند زبان مختلف، که ترکیب جدید بامعنا باشد.

۳۷–به منظور جلوگیری از وجود اسامی مشترک در برنامه ، زبانها معمولاً از چه روشی استفاده می نمایند؟ (نیمسال دوم ۸۸–۸۹)

الف.اعلان نوع داده ب.قواعد حوزه ج.کامپایل مجزا د.بین المللی شدن برنامه نویسی

۳۸–کدامیک از زبانهای زیر برای کاربردهای جستجو مورد استفاده قرار می گیرد؟ (نیمسال دوم ۸۸–۸۹)

الف.Prolog ب.Ada ج.Java د.Snobal

۳۹–این نوع دستورات زیر به موجب چه عملی مورد استفاده قرار می گیرند؟ (نیمسال دوم ۸۸–۸۹)

```
Assert (x>0 and a=1) or (x=0 and a>b+5)
```

الف.نقاط کنترلی ب.کامپایل مجزا ج.ادعا د.ردیابی اجرا

۴۰– ترکیب ویژگی‌های مختلف از یک زبان و دستیابی به یک ویژگی جدید با معنا ، چه نام دارد.(نیمسال اول ۸۹–۹۰)

الف. نقطه کنترل breakpoint ب. تعامد orthogonality

ج. ترکیب combine د. انتزاع abstraction

۴۱- از دیدگاه پروژه‌های نرم افزاری کاهش کدام یک از هزینه‌های زیر بر روی پروژه اثر مطلوب تری دارد.(نیمسال اول ۸۹-۹۰)

الف. هزینه نگهداری پروژه ب. هزینه اجرای پروژه

ج. هزینه ترجمه پروژه د. هزینه طراحی پروژه

۴۲- مدل محاسباتی تکه کد برنامه زیر چیست؟(نیمسال اول ۸۹-۹۰)

```
Int x,y,z;
x=sizeof (int);
y=sizeof (double);
z=x<y?x:y;
```

الف. مدلدستوری ب.مدل تابعی ج. مدل قانونمند د. مدل شی‌گرا

۴۳- این نوع دستورات زیر به موجب چه عملی مورد استفاده قرار می‌گیرند. (نیمسال اول ۸۹-۹۰)

```
Assert (y>0 and x=1) or (x=0 and a>b/5)
```

الف. نقاط کنترلی ب. کامپایل مجزا ج. ادعا د. ردیابی اجرا

٤٤-کدامیک از موارد زیر از اهداف زبان برنامه نویسی الگول نمی باشد؟(نیمسال اول ۹۱-۹۰)

الف. مقید نبودن به معماری ماشین ب. برنامه ها باید به زبان ماشین ترجمه شوند

ج. نشانه های زبان الگول به ریاضیات استاندارد نزدیک است د. پردازش داده های تجاری

۴۵- کدامیک از زبان های برنامه نویسی زیر جز دسته زبان های هوش مصنوعی محسوب می شود؟(نیمسال اول ۹۱-۹۰)

الف. PL/I ب. Lisp ج. Simula د. Fortran

۴۶-کدامیک از موارد زیر قابلیت تعامد در یک زبان برنامه نویسی را نشان می دهد؟(نیمسال اول ۹۱-۹۰)

الف. زبان برنامه نویسی می بایست مجموعه ای از مفاهیم واضح، ساده و یکپارچه که برای طراحی الگوریتم مورد استفاده قرار می گیرد را تدارک ببیند.

ب. منظور از تعامد این است که ساختارهایی با معنای مختلف، باهم فرق داشته باشند.

ج. منظور از تعامد این است که بتوان ویژگی های مختلفی از یک زبان را باهم ترکیب کرده و ترکیب حاصل نیز بامعنا باشد.

د. زبان باید ساختمان داده ها، عملگرها، ساختارهای کنترلی مناسب و نحو طبیعی برای تبدیل الگوریتم به برنامه داشته باشد.

سوالات تشریحی

۱–هشت مورد از صفات یک زبان خوب را بنویسید؟(نیمسال دوم ۸۵–۸۶)

۱-۱۰- پاسخنامه سوالات تستی فصل اول

سوال	الف	ب	ج	د
۱		*		
۲		*		
۳	*			
۴			*	
۵		*		
۶		*		
۷				*
۸			*	
۹				*
۱۰				*
۱۱				*
۱۲			*	
۱۳			*	
۱۴			*	
۱۵		*		
۱۶	*			
۱۷	*			
۱۸				*
۱۹				*
۲۰				*

سوال	الف	ب	ج	د
۲۱		*		
۲۲		*		
۲۳		*		
۲۴	*			
۲۵				*
۲۶			*	
۲۷		*		
۲۸				*
۲۹			*	
۳۰			*	
۳۱				*
۳۲		*		
۳۳				*
۳۴		*		
۳۵			*	
۳۶			*	
۳۷		*		
۳۸	*			
۳۹			*	
۴۰		*		
۴۱	*			
۴۲	*			
۴۳			*	
۴۴				*
۴۵		*		
۴۶			*	

فصل دوم :

اثرات معماری ماشین

آنچه در این فصل خواهید آموخت:

- مقدمه
- کامپیوتر و اجزای آن
- کامپیوترهای میان افزار
- مفسرها و معماری‌های مجازی
  - ترجمه
  - تفسیری
  - مقایسه ترجمه و تفسیری
- انواع زبان‌ها
  - زبان‌های کامپایلری
  - زبان‌های مفسری
- کامپیوترهای مجازی
  - سلسله مراتب ماشین مجازی
- انقیاد
  - زمان‌های انقیاد
    - زمان اجرا
    - زمان ترجمه
    - زمان پیاده سازی زبان
    - زمان تعریف زبان
  - اهمیت زمان‌های انقیاد
- سوالات تستی و تشریحی

۲-۱- مقدمه

در قدیم، کامپیوترها گران بودند لذا زبان های اولیه باید به گونه ای طراحی می شدند که برنامه های نوشته شده به این زبان ها، به صورت کارآمد روی کامپیوترها اجرا شوند. مثلاً زبان فرترن (جهت محاسبات عددی) و زبان لیسپ (جهت پردازش لیست) به گونه ای بودند که به خوبی به زبان ماشین تبدیل می شدند ولی نوشتن برنامه در آنها سخت بود.
اما امروزه کامپیوترها ارزان شده اند ولی برنامه نویسی گران شده است. لذا هدف این است که برنامه نویس به سادگی بتواند برنامه بنویسد هر چند که قدری کند باشد. مثلاً خصوصیات کلاس در C++، نوع داده‌ها در ML و ویژگی Package در Ada، ساخت برنامه‌ها و رفع اشکال آنها را ساده تر کرده اند. سه عاملی که در هنگام توسعه یک زبان اثر گذارند عبارتند از :

- کامپیوتری که برنامه روی آن اجرا می شود.
- مدل اجرا یا کامپیوتر مجازی که آن زبان را روی سخت افزار واقعی پشتیبانی می کند.
- مدل محاسباتی آن زبان.

۲-۲- کامپیوتر و اجزای آن

در این درس، کامپیوتر را به صورت مجموعه ای از الگوریتم‌ها و ساختمان داده‌ها تعریف می کنیم که توانایی ذخیره و اجرای برنامه‌ها را دارد. طبق این تعریف کامپیوتر می تواند سخت افزاری یا شبیه سازی شده نرم افزاری (مجازی) باشد.

کامپیوترهای سخت افزاری[۱] :

کامپیوتر سخت افزاری، کامپیوتری است که کاملاً از اجزاء سخت افزاری و مدارات الکترونیکی شامل حافظه اصلی، ثبات‌ها و ALU و ... ساخته شده است. در این نوع کامپیوترها، دقیقاً سخت افزار مربوط به هر دستور زبان ماشین وجود دارد.

کامپیوترهای میان افزار[۲]:

در یک کامپیوتر میان افزاری، هر دستور زبان ماشین، دنباله ای از ریز عملیات[۳] می باشد که در یک حافظه قابل برنامه ریزی[۴] ذخیره شده است.
هر کامپیوتر مشابه زبان‌های برنامه نویسی از ۶ جزء تشکیل شده است :

- **داده[۵]:** یک کامپیوتر باید مجموعه ای از داده‌های اولیه (مثل real و char و int) و داده‌های ساخت یافته (مثل رکورد ، آرایه و...) برای انجام عملیات فراهم کند.
- **اعمال اولیه[۶]** :یک کامپیوتر باید مجموعه ای از عملیات اولیه برای پردازش روی داده‌ها را فراهم کند.(مانند دستورات CPU یا زبان ماشین)
- **کنترل ترتیب انجام دستورات[۷]:** یک کامپیوتر باید مکانیزمی برای کنترل ترتیب اجرای عملیات داشته باشد. (کنترل ترتیب اجرای عملیات اولیه یا تعریف شده توسط کاربر)
- **دستیابی به داده[۱]:** یک کامپیوتر باید مکانیزم‌هایی برای کنترل داده‌هایی داشته باشد که با اجرای عملیات تولید می شوند. (کنترل انتقال داده بین زیر برنامه‌ها و برنامه‌ها)

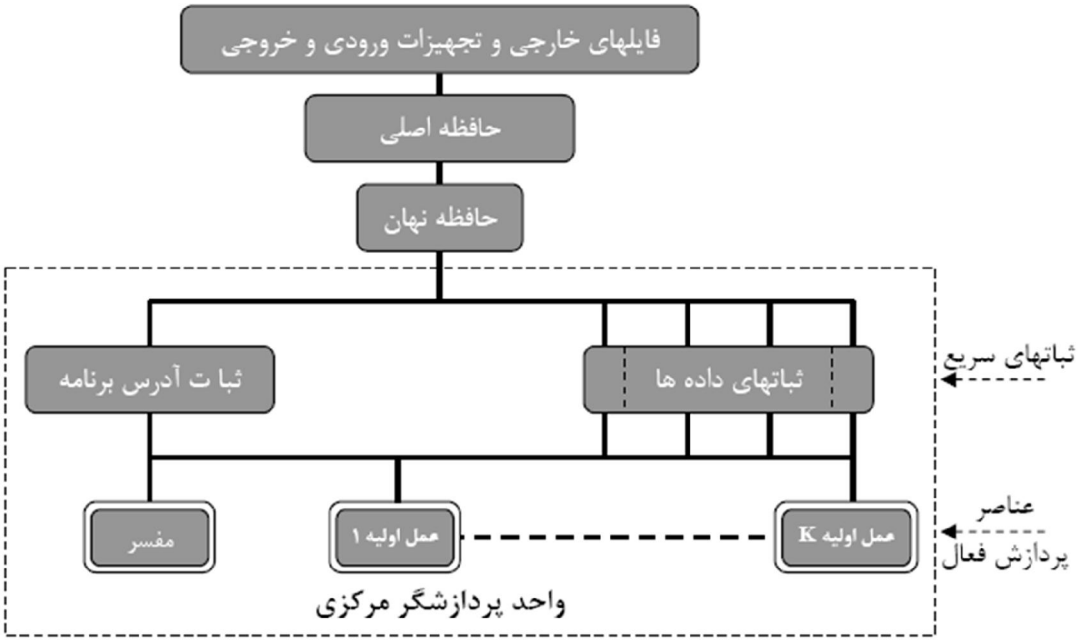
---

[۱] Hard Ware
[۲] Firmware
[۳] Micro-operation
[۴] PROM
[۵] Data
[۶] Primitive Operation
[۷] Sequence Control

- **مدیریت حافظه[۲]:** یک کامپیوتر باید مکانیزم‌هایی جهت تخصیص حافظه برای برنامه و داده و همچنین آزاد سازی حافظه داشته باشد.
- **محیط عملیاتی[۳]:** یک کامپیوتر باید مکانیزم‌هایی برای مبادله اطلاعات با دستگاه‌های جانبی فراهم سازد.

سازمان کامپیوتر:

واحد پردازشگر مرکزی (CPU) از بخش‌های مهم یک کامپیوتر می باشد. این واحد از ثبات‌های سریع و عناصر پردازش فعال تشکیل شده است. ثبات‌های داده و ثبات آدرس مهم ترین ثبات‌های درون پردازنده هستند. ثبات‌های آدرس برای آدرس دهی کردن داده‌ها و دستورات روی حافظه و ثبات‌های داده هم برای ذخیره سازی داده‌های مورد نیاز و نتایج حاصل شده از اعمال اولیه استفاده می شوند. هر دستورالعمل روی حافظه اصلی مشخص کننده یک هدف می باشد که این عمل توسط مفسر CPU ترجمه شده (کد گشایی عملیات) و دستورات لازم به بخش‌های مختلف داده می شود تا اینکه عمل اولیه بر روی داده‌ها انجام شود. عناصر پردازش فعال یک CPU از اعمال اولیه ای که برای آن تعیین شده، تشکیل شده است این اعمال اولیه ممکن است در پردازشگرهای مختلف، متفاوت باشند. سازمان یک کامپیوتر معمولی در شکل زیر نشان داده شده است.



شکل ۲ -۱ سازمان یک کامپیوتر معمولی

توضیح اجزای شش گانه کامپیوتری به طور مفصل تر:

۲-۲-۱- داده‌ها

داده‌ها در حافظه ها ذخیره می شوند. در شکل فوق، سه جزء اصلی حافظه داده‌ها نمایش داده شده است.

- **حافظه اصلی:** حافظه اصلی به صورت دنباله ای از بیت‌های خطی سازماندهی می شوند که از کلمات با طول ثابت تشکیل شده اند.

---

[۱] Data Control
[۲] Storage managment
[۳] Oprating Environment

- **حافظه نهان**[۱] **:** طول ثبات‌های سریع به اندازه طول کلمات است و طوری تقسیم بندی می شوند که هر قسمت آن قابل دستیابی باشد. حافظه سریع نهان معمولاً بین حافظه اصلی و ثبات‌ها قرار می گیرد و مکانیزمی برای دسترسی سریع به داده‌های موجود در حافظه است
- **فایل‌های خارجی (حافظه‌های جانبی):** که بر روی دیسک یا CD یا نوار مغناطیسی ذخیره می شوند.

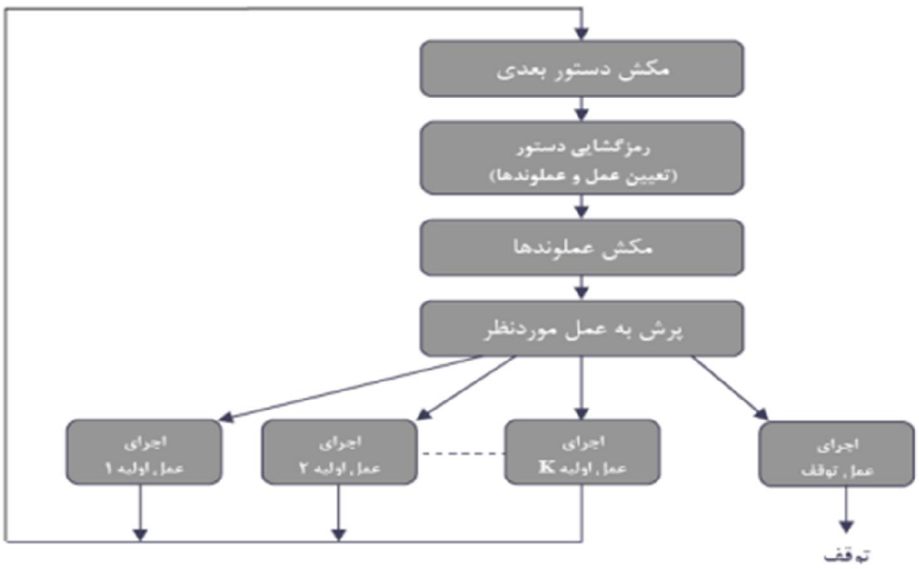
۲-۲-۲- اعمال

کامپیوتر باید مجموعه ای از اعمال اولیه توکار داشته باشد که متناظر با کدهای عملیاتی هستند که به صورت دستورات زبان ماشین می باشند.

- **اعمال اولیه محاسباتی:** مثل Add , Sub , Mult , Div
- **اعمال اولیه برای تست خواصی از عناصر داده:** مثل SPA,SNA,SZA (به ترتیب مقایسه باصفر و منفی یا مثبت بودن اعداد)
- **اعمال اولیه برای کنترل دستگاه‌های I/O:** مثل output,input,skip in,skip out

۲-۲-۳- کنترل ترتیب

در حین اجرای برنامه دستور بعدی که باید اجرا شود توسط محتویات ثبات آدرس برنامه[۲] یا PC مشخص می شود. این ثبات، حاوی آدرس دستور بعدی است و مفسر از آن استفاده می کند. مفسر قلب عملکرد کامپیوتر است و معمولاً الگوریتمی چرخه ای را اجرا و تکرار می کند. در هر چرخه مفسر آدرس دستور بعدی را از ثبات آدرس برنامه می گیرد و دستور مورد نظر را از حافظه مکش می کند. آن دستور را به یک کد عملیاتی و مجموعه ای از عملوندها تبدیل می کند. عملوندها را در صورت لزوم مکش می کند و عملیات را با آن فراخوانی می کند. اعمال اولیه ممکن است داده‌های موجود در حافظه یا ثبات‌ها را اصلاح کنند. شکل زیر عملکرد یک مفسر را نشان می دهد.



شکل ۲ -۲ تفسیر و اجرای برنامه

[۱] Cache
[۲] Program Counter

۲-۲-۴- دستیابی به داده‌ها

علاوه بر کد عملیاتی، هر دستور ماشین باید عملوندهایی را که یک عملیات از آن استفاده می کند را مشخص کند. عملوند ممکن است در حافظه اصلی یا در ثبات باشد. کامپیوتر باید مکانیزمی برای تعیین عملوندها، بازیابی آنها و ذخیره نتایج در عملوندها داشته باشد که به این امکانات کنترل دستیابی به داده‌ها گفته می شود. برای دستیابی به عملوندها، متداولترین راه، آدرس حافظه یا ثبات است.

۲-۲-۵- مدیریت حافظه

یک اصل در طراحی ماشین این است که تمام منابع کامپیوتر مثل حافظه اصلی، CPU و دستگاه‌های جانبی تا آنجایی که ممکن است فعال باشند. ولی به دلیل سرعت متفاوت هر یک از این منابع، در این اصل یک تناقض وجود دارد. برای مقابله با عدم توازن بین دستیابی به داده‌های خارجی و CPU، سیستم عامل از تکنیک چند برنامه ای استفاده می کند و تا زمانی که داده‌های مورد نظر از دستگاه‌های خارجی خوانده می شوند وقت CPU به برنامه دیگری اختصاص داده می شود. برای برقراری توازن بین حافظه اصلی و CPU از حافظه نهان استفاده می شود. حافظه نهان یک حافظه کوچک بین CPU و حافظه اصلی است.

۲-۲-۶- محیط عملیاتی

محیط عملیاتی کامپیوتر متشکل از مجموعه ای از حافظه‌های جانبی و دستگاه‌های I/O می باشد. هر ارتباط کامپیوتر با دنیای خارج از طریق محیط عملیاتی صورت می گیرد. محیط عملیاتی شامل: حافظه‌های با سرعت بالا مانند Flash، حافظه با سرعت متوسط مثل Disk و CD، حافظه کند مانند نوارها و دستگاه‌های I/O مانند چاپگر، صفحه کلید، مانیتور و... می باشد.

۲-۳- کامپیوترهای میان افزار

کامپیوتر میان افزار توسط ریز برنامه ای[۱] شبیه سازی می شود که بر روی کامپیوتر سخت افزار قابل ریزبرنامه نویسی، اجرا می شود. زبان ماشین این کامپیوتر متشکل از مجموعه بسیار سطح پایین از ریز دستورات است که انتقال داده را بین حافظه اصلی و ثبات‌ها، بین خود ثبات‌ها و از ثبات‌ها، از طریق پردازنده انجام می دهد. ریز برنامه ویژه ای با استفاده از این مجموعه دستورات نوشته می شود که چرخه تفسیر و اعمال اولیه گوناگون کامپیوتر مورد نظر را تعریف می کند. ریزب رنامه عمل کامپیوتر مطلوب را بر روی کامپیوتر میزبان قابل ریزبرنامه نویسی شبیه سازی می کند. خود ریزبرنامه در یک حافظه فقط خواندنی ویژه ROM در کامپیوتر میزبان ذخیره می شود و با سخت افزار کامپیوتر میزبان با سرعت بالایی اجرا می شود. کامپیوتری که از طریق شبیه سازی ریزبرنامه ای بوجود می آید کامپیوتر مجازی نامیده می شود، زیرا توسط ریز برنامه شبیه سازی می شود و در صورت عدم وجود این ریزبرنامه، ماشین وجود نخواهد داشت. به این نکته توجه داشته باشید که زبان ماشین به زبان سطح پایین مانند اسمبلی محدود نمی شود. مثلاً می توان کامپیوتری ساخت که زبان ماشین آن C یا Ada باشد که به آن کامپیوتر مجازیC یا Ada می گویند ولی ساخت چنین کامپیوتری پیچیده بوده و کارآیی آن نیز کمتر است.

تعریفی دیگر (کامپیوترهای میان افزار):

با داشتن توصیفی از یک زبان برنامه سازی می توان کامپیوتری کاملا سخت افزاری ایجاد کرد که زبان ماشین آن کامپیوتر زبان مورد نظر ما باشد. برای مثال می توان کامپیوتری ساخت که زبان ماشین آن زبان C باشد ولی این کامپیوتر دارای هزینه بالا و انعطاف کمتری نسبت به حالتی است که زبان ماشین مجموعه مختصر و جامعی از دستورات از سطح پایین باشد. در عوض می توان کامپیوتری ساخت که اجرای دستورات زبان سطح بالا در آن به روش ریز برنامه سازی پیاده شده باشد به این معنی که برای هر دستور سطح بالا ریز دستوراتی که خود به دستورات سطح پایین آن کامپیوتر تبدیل می شوند وجود داشته باشد که به این مدل کامپیوتری میان افزاری می گویند.

---

[۱] Micro program

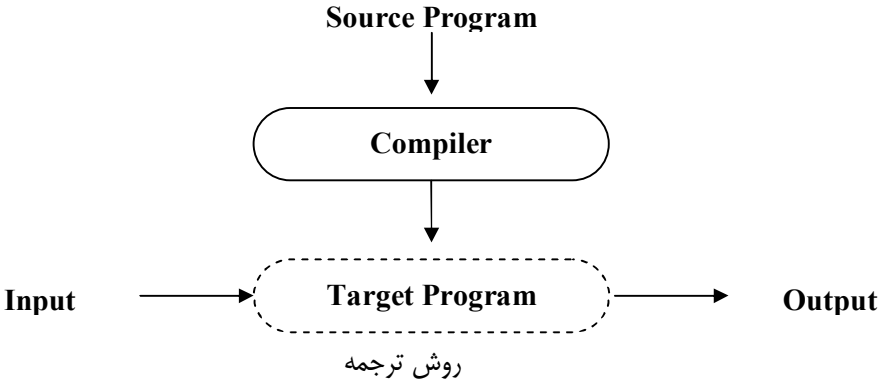
۲-۴- مفسرها و معماری‌های مجازی

وقتی یک برنامه به یک زبان نوشته می شود سوالی که ایجاد می شود این است که این برنامه به زبان سطح بالا چگونه در یک کامپیوتر واقعی صرف نظر از زبان ماشین آن اجرا می شود. برای این منظور، دو راه حل وجود دارد:

- **ترجمه، کامپایل کردن (Translation)**
- **تفسیری، شبیه سازی نرم افزاری (Interpreter)**

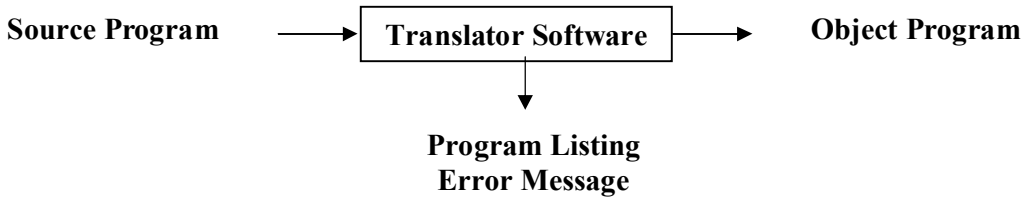
۲-۴-۱- روش ترجمه

در این روش برنامه به زبان سطح بالا طی فرآیندهایی به زبان ماشین تبدیل می شود که قابل اجرا بر روی سخت افزار است. به طور کلی مفسر (نرم افزار مترجم) به هر پردازنده زبانی گفته می شود که برنامه ای به زبان منبع که می تواند سطح بالا یا پایین باشد را گرفته و آن را به زبان مقصد تبدیل می کند.



شکل ۲ -۳

در روش ترجمه ابزارهایی مورد نیاز است که هر کدام از این ابزارها خود یک نوع مترجم می باشند. مترجم (مفسر) نرم افزاری است که برنامه به یک زبان مبدا را دریافت کرده و به برنامه معادل در زبان مقصد تبدیل می کند. در ضمن اگر برنامه به زبان مبدا با ساختار زبان مبدا تطابق نداشته باشد پیغام خطا صادر خواهد شد.



شکل ۲ -۴

انواع مترجم‌ها (مفسرها) عبارتند از:

اسمبلر(Assembler): مفسری می باشد که زبان منبع آن زبان اسمبلی و زبان مقصد آن زبان ماشین واقعی می باشد.

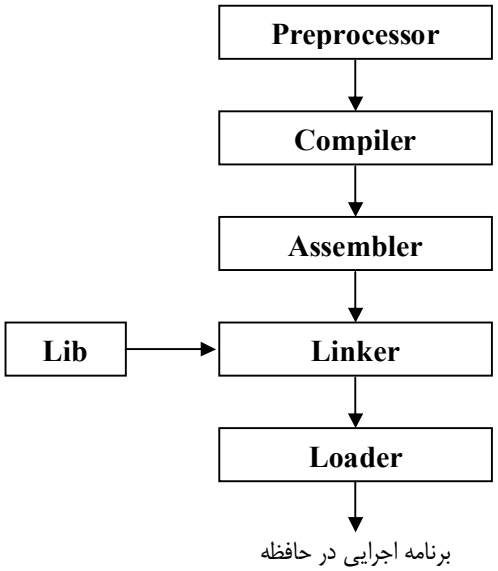
کامپایلر(Compiler): مفسری می باشد که زبان منبع آن یک زبان سطح بالا و زبان مقصد آن نزدیک به زبان ماشین (مانند اسمبلی) می باشد.

بارکننده(Loader): مفسری می باشد که زبان منبع آن زبان ماشین به شکل جابجا پذیر (آدرس نسبی) و زبان مقصد آن کد ماشین واقعی است. بارکننده، ماژول های مختلف اجرایی را به هم پیوند داده و آدرس‌های آنها را به صورت مناسب جابجا می کند.

پیوند دهنده(Linker): این مفسر بخش‌های مختلف برنامه را دریافت نموده، آنها را سرهم بندی کرده و برنامه خروجی تقریباً شبیه برنامه ورودی به شکل کامل تر تولید می کند.

پیش پردازنده یا پردازنده ماکرو(Preproccesor): مفسری می باشد که زبان منبع آن شکل توسعه یافته ای از یک زبان سطح بالا مانند C++ می باشد و زبان مقصد آن شکل استانداردی از همان زبان می باشد (همان برنامه C). مثلاً در زبان C دستوراتی که با # شروع می شوند مثل تعریف ماکروها یا فایل‌های include ابتدا بسط داده شده و به دستوراتی از زبان C تبدیل می شوند.

ترتیب اجرای مفسرها برای ترجمه یک برنامه به شکل زیر می باشند:



شکل ۲ -۵

یک مثال برای نمایش عملکرد بار کننده در شکل زیر آمده است:

زیر برنامه	آدرس کامپایل شده (آدرس نسبی)	آدرس اجرایی (آدرس واقعی)
P	۰-۹۹۹	۰-۹۹۹
Q	۰-۱۹۹۹	۱۰۰۰-۲۹۹۹
توابع کتابخانه	۰-۴۹۹۹	۳۰۰۰-۷۹۹۹

جدول ۲ - ۱

برنامه اجرایی، برنامه ای است که از آدرس‌های ۰ تا ۷۹۹۹ استفاده می کند.

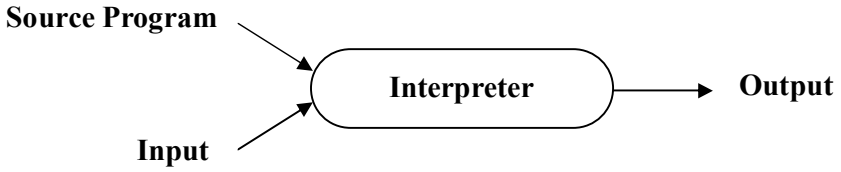
نکته : مراحل ترجمه از زبان سطح بالا به زبان ماشین اغلب بیش از یک مرحله است.

به عنوان مثال :

C++ →C →اسمبلی→زبان ماشین

۲-۴-۲- روش شبیه سازی نرم افزاری (تفسیری)

در این روش کد برنامه منبع مستقیماً به شبیه ساز نرم افزاری یا مفسر داده می شود و مفسر دستورات زبان سطح بالا را تفسیر و بلافاصله اجرا می کند. در این روش به جای اینکه زبان سطح بالا به زبان ماشین ترجمه شود به کمک شبیه سازی، آن برنامه روی یک کامپیوتر میزبان، اجرا خواهد شد. منظور از کامپیوتر میزبان، کامپیوتری است که زبان ماشین آن یک زبان سطح بالا است.



روش تفسیری

شکل ۲ -۶

۲-۴-۳- مقایسه روش ترجمه و تفسیری

با آنکه هر دو روش ترجمه و تفسیری برنامه‌هایی به زبان سطح بالا را به عنوان ورودی دریافت می کنند ولی در موارد زیر با هم تفاوت دارند :

۱- در روش ترجمه برنامه به طور کامل به زبان ماشین تبدیل شده و سپس اجرا می شود درحالیکه درروش تفسیری یا شبیه سازی نرم افزاری تک تک دستورات زبان سطح بالا ابتدا تفسیر و سپس مجموعه دستورات لازم برای شبیه سازی آن دستورات، اجرا می شود.

۲- سرعت اجرا در روش ترجمه یا کامپایلری بیشتر از مفسری یا شبیه سازی است چون در روش ترجمه فاز ترجمه و اجرا جدا از هم هستند، ولی درشبیه سازی فاز ترجمه و اجرا یکسان هستند.

۳- مترجم (روش ترجمه) دستورات برنامه را به ترتیب فیزیکی ورودی پردازش می کند، ولی شبیه ساز (روش تفسیری) جریان منطقی برنامه را دنبال می کند.

۴- مترجم هر دستور را فقط یکبار پردازش یا ترجمه می کند ولی شبیه ساز (روش تفسیری) ممکن است برخی از دستورات را چندبار پردازش کرده مانند حلقه for و حتی برخی از آنها را اصلاً پردازش نکند مثل یک بلوک شرطی که همواره غلط است. بنابراین می توان نتیجه گرفت که در روش کامپایلری اگر خطایی وجود داشته باشد حتماً گرفته خواهد شد ولی در روش شبیه سازی به دلیل اینکه کنترل منطقی برنامه دنبال می شود ممکن است بعضی از خطاها نادیده گرفته شوند.

۵- در روش کامپایلری برایn بار اجرا یک ترجمه لازم است ولی در روش تفسیری برای n بار اجرا n ترجمه لازم است. مثال واضح حلقه تکرار می باشد.

۶-ترجمه محض و شبیه سازی محض دو کرانه اند یعنی حالت تئوریک دارند. در عمل ترجمه محض به ندرت صورت می گیرد، مگر در مواردی که زبان ورودی دقیقاً شبیه به زبان ماشین باشد مانند اسمبلی. شبیه سازی محض نیز به ندرت مورد استفاده قرار می گیرد به جز مواردی مثل زبان‌های محاوره ای یا زبان‌های کنترل سیستم عامل. اغلب زبان‌ها به صورت ترکیبی از روش های ترجمه و تفسیری، پیاده سازی می شوند.

۷- برخی از جنبه‌های ساختاری برنامه بهتر است قبل از اجرا ترجمه شوند مثل حلقه ای که قرار است ۱۰۰۰ بار اجرا شود ولی برخی دیگر از جنبه‌ها بهتر است فقط در زمان اجرا پردازش شوند.
۸- ایراد مهم روش ترجمه از دست رفتن اطلاعاتی در رابطه با برنامه است مثلاً اگر برنامه به زبان ماشین ترجمه شود و خطایی رخ دهد، تعیین اینکه کدام دستور زبان منبع این خطا را ایجاد کرده است سخت است چون حاصل ترجمه به زبان ماشین که فقط ۰ و ۱ می باشد تبدیل شده است ولی در روش تفسیری تمام اطلاعات مربوطه موجود است. برنامه مقصد در روش ترجمه بزرگتر از برنامه مبدأ است ولی در روش تفسیری معمولاً برنامه مقصد کوچکتر است.
۹- در روش تفسیری از آنجایی که دستورات تا زمان اجرا شکل اولیه خود را خواهند داشت لذا چند کپی از آنها نگهداری نمی شود و بدین ترتیب در مقایسه با روش ترجمه در حافظه صرفه جویی می شود. اما در هر حال، کل هزینه رمزگشایی باید پرداخته شود، در مقابل در روش ترجمه چندین فایل داریم که نتیجه ترجمه در آن ذخیره می شود. به عنوان یک قاعده کلی می توان گفت که روش ترجمه زمانی استفاده می شود که ساختار زبان منبع نمایش مستقیمی در زبان مقصد دارد و لذا تبدیل کد چندان سخت نخواهد بود، در سایر موارد از روش تفسیری استفاده می شود.

۲-۵- انواع زبانها

یک سوال کلیدی این است که آیا نمایش اصلی برنامه در ضمن اجرا همان نمایش زبان ماشین کامپیوتر واقعی است یا خیر؟ که بر این اساس دو نوع تقسیم بندی برای زبان‌ها وجود دارد:

۲-۵-۱- زبانهای کامپایلری

دراین زبانها، برنامه‌ها قبل ازشروع اجرا، به زبان کامپیوتر واقعی ترجمه می‌شوند. در حقیقت این گونه از زبانها از روش ترجمه استفاده می کنند.به طور خلاصه برای اینگونه زبانها خواهیم داشت:

- استفاده از روش کامپایلری باعث می‌شود برنامه‌ها با سرعت زیاد، اجرا شوند.
- مترجم زبانهای کامپایلری، تقریباً پیچیده و بزرگ هستند.
- زبانهای C++, C, FORTRAN و ADA زبانهای کامپایلری هستند

۲-۵-۲- زبانهای مفسری

دراین زبانها مترجم، کد ماشین کامپیوتر واقعی را تولید نمی‌کند، بلکه یک شکل میانی از برنامه را پدید می‌آورد که اجرای آن ساده‌تر از دستورات اولیه است. ولی با کد ماشین فرق دارد. مفسر این دستورات یک مفسر نرم‌افزاری است ولذا اجرای آن کندتر از کامپایلری است. مثلا Java شبیه C++ است، ولی تفسیری است، که مفسر آن کد میانی به نام بایت کد را برای ماشین مجازی Java ایجاد می‌کند. به طور خلاصه برای این‌گونه زبانها خواهیم داشت:

- استفاده از روش مفسری منجر به برنامه‌هایی می‌شودکه اجرای آنها کندتر است.
- مترجم زبانهای مفسری بسیار ساده هستند.
- زبانهایی مثل ML،LispPerl,Postscript,Smalltalk,Basic,HTML زبانهای مفسری هستند.

۲-۶- کامپیوترهای مجازی[1]

قبلاً کامپیوتر را بصورت مجموعه ای از الگوریتم‌ها و ساختمان‌داده‌ها تعریف کردیم که قابلیت ذخیره و اجرای برنامه‌ها را دارد، روش‌های ساخت کامپیوتر عبارتند از:

- ***ازطریق سخت افزار (Through a hard ware realization)*** : ساختمان داده‌ها و الگوریتم‌ها به صورت سخت افزاری پیاده سازی می‌شود.

[1] Virtual Computer

- ***از طریق میان افزار(Through firmware realization)***: ساختمان داده‌ها و الگوریتم‌ها از طریق ریزبرنامه نویسی برای سخت افزار مناسب ایجاد می‌شود.
- ***از طریق ماشین مجازی (Through soft ware realization)***: دراین حالت ساختمان داده‌ها و الگوریتم-ها از طریق برنامه نویسی و زبان‌های دیگر نمایش داده می‌شوند.
- ***از طریق ترکیبی (Through a hard ware realization)*** :دراین تکنیک بخش‌های مختلف کامپیوتر مستقیما در سخت افزار و یا به وسیله شبیه سازی نرم افزاری نمایش داده می‌شود.

با توجه به موارد فوق، سه عامل منجر به تفاوتهایی در بین پیاده سازیهای یک زبان می شود:

- اختلاف در امکاناتی که روی کامپیوتر پایه موجود است.
- اختلاف در مفاهیم پیاده‌سازی کامپیوتر مجازی(VC)که بطور ضمنی در تعریف زبان ملموس است.
- اختلاف درانتخاب‌هایی که برای پیاده‌سازی و شبیه سازی زبانهای سطح بالا می‌تواند بکارگرفته شود.

۲-۶-۱- سلسله مراتب کامپیوترهای مجازی

درعمل یک کامپیوتر مجازی داریم یعنی کامپیوتری که به زبانی غیر از زبان ماشین کارمی‌کند. کامپیوتری مجازی است که توسط طراح زبان برنامه سازی طراحی می‌شود. از دید برنامه نویس به زبان سطح بالا و در عمل ساختار یک کامپیوتر مجازی را می‌توان به صورت سلسله مراتب شکل زیر در نظر گرفت.

کامپیوتر مجازی تعریف شده توسط برنامه نویس (پیاده سازی توسط زبان سطح بالا)
کامپیوتر مجازی زبان سطح بالا (پیاده سازی توسط برنامه‌ها و اجرا توسط OS)
کامپیوتر مجازی سیستم عامل (بابرنامه‌هایی که بر روی کامپیوتر مجازی یا میان افزار اجرا می‌شوند پیاده سازی می‌گردد)
کامپیوتر مجازی میان افزار (بادستورات زبان ماشین پیاده سازی شده که با ریزدستورات توسط کامپیوتر واقعی اجرا می‌شود)
کامپیوتر سخت افزار واقعی (توسط اجزا فیزیکی پیاده سازی شده است)

جدول ۲ – ۲لایه‌های کامپیوترهای مجازی برای برنامه کاربردی وب

در پایین سلسله مراتب کامپیوتر، سخت‌افزار واقعی وجود دارد که برنامه نویس به طور مستقیم با این کامپیوتر سروکار ندارد. لایه‌هایی از نرم‌افزار در بالای این کامپیوتر واقعی وجود دارد در بالای ماشین مجازی زبان C که توسط کامپایلر زبان C ایجاد شده، برنامه‌نویس برنامه‌ای به نام مرورگر وب را با استفاده از زبان C اجرا می‌کند. این مرورگر، ماشین مجازی وب را ایجاد می‌کند که می‌تواند ساختمان‌داده‌های اصلی وب و زبان HTML را پردازش کند. در بالاترین سطح این سلسله مراتب کامپیوتر، برنامه کاربردی وب قرار دارد که برنامه نویس صفحات وب، با استفاده از ماشین مجازی وب برنامه‌های خود را اجرا می‌کند.

نکته: نتیجه‌ای که از بحث سلسله مراتبی بودن کامپیوترهای مجازی بدست می‌آید این است که داده و برنامه معادل هم هستند یا به عبارتی همیشه نمی توان در استفاده از یک کامپیوتر مجازی بین داده و برنامه تمایز قائل شد. برای مثال اجرای یک فایل برنامه HTML بر روی برنامه مرورگر را در نظر بگیرید. از دید مرورگر فایل مورد نظر داده محسوب می شود در

حالی که از دید برنامه ساز وب (web) آن فایل یک برنامه است. خود برنامه مرورگر نیز از دید کامپایلری که آن را تولید کرده است داده خروجی محسوب می شود و خود آن کامپایلر نیز با وجود اینکه برنامه محسوب می شود از دید سازنده آن(کامپایلر تولید کننده آن) داده محسوب می شود. در زبان‌هایی مثل C و Fortran داده و برنامه جدای از یکدیگر ذخیره می‌شوند ولی در زبان‌هایی مثل Lisp و Prolog هردو در یک‌جا ذخیره می‌شوند و برنامه‌ها و داده‌ها مخلوط هستند. فقط فرایند اجرا، آنها را از هم تفکیک می‌کند.

۲-۷- انقیاد و زمانهای انقیاد

هنگام پیاده‌سازی و یا اجرای یک برنامه، عنصری از این برنامه می‌تواند صفتی از مجموعه صفات ممکن را به خود بگیرد به این عمل انقیاد[1] و به زمان انجام این عمل، زمان انقیاد گفته می‌شود. زمانهای انقیاد به صورت زیرطبقه بندی می‌شوند:

۲-۷-۱- زمان اجرا[2]

این انقیادها در هنگام اجرای برنامه صورت می‌گیرند. مثل انقیاد متغیرها به مقادیرشان و انقیاد متغیرهابه محل‌های خاصی از حافظه. انقیادهای زمان اجرا به دودسته‌ی کلی تقسیم می‌شوند:

- **در هنگام ورود به زیربرنامه:** به عنوان مثال هنگام صدا زدن تابع در زبان C یا Pascal انقیاد پارامترهای مجازی به واقعی و انقیاد پارامترهای مجازی به محل‌هایی از حافظه.
- **در نقطه خاصی از اجرای برنامه:** برخی از انقیادها در حین اجرا، در نقطه خاصی از برنامه انجام می‌پذیرند. مانند انقیاد متغیرها به مقادیرشان توسط دستور انتساب یا انقیاد اسامی متغیرهابه محل‌هایی از حافظه در هر نقطه‌ای از برنامه مثلاً در زبان ML و Lisp.

۲-۷-۲- زمان ترجمه[3]

این انقیادها در زمان ترجمه رخ می‌دهند و به سه دسته تقسیم می‌شوند:

- **توسط برنامه‌نویس:** که در هنگام نوشتن برنامه‌ها توسط برنامه نویس انجام می‌شود مانند اسامی متغیرها، نوع متغیرها و ساختار دستورات.
- **توسط مترجم زبان:** بعضی انقیادها توسط مترجم زبان صورت می‌گیرد. مثل انتخاب محل نسبی داده در حافظه‌ای که به زیربرنامه اختصاص داده می‌شود یا چگونگی ذخیره‌سازی آرایه‌ها (سطری یا ستونی) که از دید کاربر پنهان است، توسط مترجم صورت می‌گیرد.
- **توسط بارکننده:** هنگامی که برنامه‌ها متشکل از چند زیربرنامه هستند هنگام بارکردن آنها در حافظه، آدرس متغیرهای موجود در زیربرنامه‌ها، باید به آدرس واقعی در کامپیوتر انقیاد شوند.

۲-۷-۳- زمان پیاده سازی[4]

برخی از ویژگی‌های یک زبان ممکن است در پیاده‌سازی‌های مختلف آن متفاوت باشد. پیاده‌سازی زبان با توجه به امکانات سخت‌افزاری می‌باشد به عنوان مثال نمایش اعداد، اعمال محاسباتی، محاسبات ریاضی و غیره در این محدوده جای می‌گیرند یا

[1] Binding
[2] Run time
[3] Translation or compile time
[4] Language implementation time

محدوده مقادیر اعداد Short int در پیاده‌سازی‌های مختلف زبان C ممکن است متفاوت باشند. مثلاً در یک ماشین یا کامپیوتر Short int، ۸ بیتی و در ماشین دیگر ۱۶ بیتی باشد.

۲-۷-۴- زمان تعریف یا طراحی زبان[1]

اغلبساختارهایزبانهاییبرنامهنویسی،شکل‌های مختلف دستورات، انواع متغیرها، انواع ساختمان داده و غیره مواردی هستند که در زمان تعریف زبان معین می‌شوند. مثلاً متغیرهای i,j,…,n در فرترن به طور پیش‌فرض از نوع Integer است.

✓ **پاسخ سوالات زیر در زمان تعریف زبان می‌باشد :**

چه انواعی داریم ؟ ثابت‌ها کدامند؟ عملگرها کدامند؟ شکل ظاهری عملگرها Syntax دستورات؟ ساختار برنامه؟

فرم دستوارت و عملی که هر دستور انجام می‌دهد؟

توابع از پیش تعریف شده (توابع کتابخانه ای)

به عنوان مثال ، متغیرهایی که با حرف i و j در زبان فرترن شروع می‌شوند همگی از نوع Integer هستند.

نکته:مجموعه مقادیر ممکن برای یک متغیر از هر نوع در زمان پیاده‌سازی مشخص می‌شود و ممکن است این مجموعه مقادیر در پیاده‌سازی‌های مختلف تفاوت داشته باشد اینکه یک نوع در زبان برنامه‌نویسی موجود باشد یا نباشد در زمان تعریف زبان، اینکه یک متغیر در تعریف برنامه از چه نوعی باشد در هنگام ترجمه زبان و مقدار یک متغیر در هر لحظه در هنگام اجرای زبان می‌باشد.

مثال: تمام انقیادهای دستور زیر در زبان L را بررسی کنید:

```
x=x+10;
```

- **مجموعه ای از انواع ممکن برایX**

مجموعه‌ای از انواع قابل قبول برای متغیر X در زمان تعریف زبان مشخص می‌شود مثلاً در Pascal می‌توان از انواع Char، Integer، Set، Boolean و غیره استفاده کرد.

نکته: اگر زبان برنامه‌نویسی به کاربر اجازه دهد که خودش انواع جدیدی تعریف کند آنگاه مجموعه‌ای از انواع ممکن در زمان ترجمه مشخص می‌شود. به عنوان مثال در زبانهای C، Pascal و Ada که نوع شمارشی در زبان Pascal توسط برنامه‌نویس تعریف می‌شود.

- **نوع متغیر x**

معمولاً در زمان ترجمه مشخص می‌شود به عنوان مثال در Pascal برای این منظور یک متغیر باید اعلان شود

نکته: در برخی از زبانها مثل Smalltalk و Prolog نوع داده ممکن است در زمان اجرا مشخص شود، بطوری که انتساب مقداری از یک نوع بهX، نوع X را مشخص می‌کند در این زبانها ممکن است در قسمتی از برنامه X از نوع صحیح و در جای دیگر مقداری کاراکتری داشته باشد.

- **مجموعه‌ای از مقادیر ممکن برای متغیر X**

مجموعه مقادیر ممکن برای متغییر X در زمان پیاده‌سازی مشخص می‌شود. در زمان پیاده‌سازی زبان تعداد بیت‌ها برای قرار دادن یک مقدار از نوع Float تعیین می‌شود لذا مجموعه دقیقی از مقادیر ممکن برای X بوسیله این تعداد بیت‌ها مشخص

[1]Language definition time

می‌شود مثلاً اگر ۱۶ بیت برای یک متغیر از نوع صحیح در نظر گرفته شود، مجموعه مقادیر ممکن برای X از ۱-^{۱۶}۲ تا ^{۱۶} ۲- می‌باشد.

- **مقدار متغیر X**

در هر نقطه از اجرای برنامه مقدار خاصی به X، مقید می‌شود. یعنی مقدار متغیر X در زمان اجرا مشخص می‌شود.

- **نمایش مقدار ثابت ۱۰**

انتخاب نمایش دهدهی در متن داخل برنامه یعنی (۱۰ برایده) در زمان تعریف زبان. (اگر به صورت بیتی نمایش دهیم) انتخاب رشته‌ای از بیت‌ها برای نمایش داخلی در زمان پیاده‌سازی (۱۰۱۰).

- **عملگر + :**

انتخاب نماد + برای نمایش عمل جمع در زمان تعریف زبان انجام می‌شود ولی معنای عملگر + برای انجام عمل جمع در زمان ترجمه مشخص می‌شود. بدلیل اینکه پس از مشخص شدن نوع عملوندها، تعیین می‌شود که علامت + چه جمعی را انجام دهد. (جمع دو عدد صحیح یا جمع دو عدد اعشاری)

اهمیت زمانهای انقیاد

بسیاری از تفاوت‌های بین زبانهای برنامه نویسی، درواقع به تفاوت زبان‌ها در زمان انقیاد برمی‌گردد و اغلب وابسته به این است که انقیاد در زمان ترجمه صورت می‌گیرد یا در زمان اجرا. به عنوان مثال در زبان Fortran کارکردن با آرایه‌های بزرگ، ساده ولی در ML، مشکل است. علت این موضوع آن است که در Fortran اغلب انقیادها در زمان ترجمه و در ML اغلب انقیادها در زمان اجرا صورت می‌پذیرد. بنابراین Fortran انقیادها را فقط یکبار در زمان ترجمه انجام می‌دهد. در حالیکه ML بیشتر وقت خود راصرف ایجاد و حذف انقیادها در زمان اجرا می‌کند. بنابراین سرعت محاسبات در Fortran بیشتر از ML است. از سوی دیگر، انعطاف پذیری ML در دستکاری رشته‌ها بیشتر از Fortran است چراکه در زبان Fortran اندازه رشته‌ها در زمان ترجمه باید مشخص و معین باشد ولی در ML اینگونه انقیادها می‌توانند تا خواندن رشته از ورودی به تعویق بیفتد. بنابراین عموماً کارایی (یاسرعت) یک زبان با انعطاف پذیری آن نسبت عکس دارد. بنابراین زمان انقیاد می‌تواند روی انعطاف پذیری و سرعت برنامه مؤثر باشد.

اگر عمل انقیاد در حین اجرا مشخص شود انقیاد دیررس[1] گفته می‌شود. و اگر عمل انقیاد در حین ترجمه مشخص شود انقیاد زودرس[2] گفته می‌شود.

در زبانهای Fortran,c, Pascal اغلب انقیادها در زمان کامپایل یا ترجمه انجام می شود ولی در زبانهایی مثل ML ,Lisp اغلب انقیادها در زمان اجرا صورت می‌گیرد. در زبان Ada که هم برای قابلیت انعطاف و هم برای کارایی طراحی شده است برنامه نویس می‌تواند زمان انقیاد را انتخاب کند. اغلب، تغییرات کوچک در یک زبان برنامه‌نویسی منجر به تغییرات بزرگی در زمان انقیاد می‌شود مثلاً در Fortran90 استفاده از بازگشتی منجر به تغییرات عمده ای در زمان انقیاد ویژگی‌های Fortran شد.

[1] Late binding
[2] Early binding

انواع زبان‌ها بر اساس زمان انقیاد

۱. زبانهایی با انقیاد زودرس (EBT) : کارایی و سرعت اجرا بالا، انعطاف پذیری پایین
مانند زبان های Pascal , C , Fortran و ... که انقیاد در آنها در زمان ترجمه انجام می‌شود. (معمولا کامپایلری هستند)

۲. زبانهایی با انقیاد دیررس (LBT): کارایی و سرعت اجرا پایین، انعطاف پذیری بالا
مانند زبانهای Basic , Prolog , Lisp , ML که اغلب انقیاد در آنها در زمان اجرا انجام می‌شود. (معمولا مفسری هستند)

✓ درزبانی مثل Ada که هم برای قابلیت انعطاف و هم برای کارایی طراحی شده، می‌توان زمان انقیاد را انتخاب کرد.

۲-۸- سوالات فصل دوم

سوالات تستی

۱-جزئیات مربوط به نمایش اعداد و اعمال محاسباتی در کدامیک از موارد زمانهای انقیاد زیر مشخص می شود؟(نیمسال اول ۸۵-۸۶)

الف.زمان اجرا ب.زمان ترجمه
ج.زمان پیاده سازی زبان د.زمان تعریف زبان

۲-کدام جمله صحیح نیست؟(نیمسال اول ۸۵-۸۶)

الف.ترجمه محض و شبیه سازی محض دو کرانه اند.
ب.در فرترن انقیاد دیر رس باعث شده است که سرعت آن برای محاسبات ریاضی از ML بیشتر باشد.
ج.تغییرات کوچکی در یک زبان باعث تغییرات بزرگی در زمانهای انقیاد می شود.
د. انقیاد در ورود به زیربرنامه یا بلوک،یک انقیاد زمان اجرا است.

۳-کدام گزینه غلط است؟(نیمسال دوم ۸۵-۸۶)

الف.برای برقراری توازن بین حافظه اصلی و پردازنده مرکزی از حافظه نهان استفاده می شود.
ب.محیط عملیاتی کامپیوتر متشکل از مجموعه ای از حافظه جانبی و دستگاههای ورودی و خروجی است.
ج.یک اصل در طراحی ماشین این است که تمام منابع کامپیوتری تا آنجا که ممکن است فعال باشند.
د.مفسر معمولاً الگوریتمی غیر چرخه ای را اجرا می کند.

۴-کدام زبان کامپایلری می باشد؟(نیمسال دوم ۸۵-۸۶)

الف.ADA ب.LISP, PROLOG
ج.الف و ب د.اسمالتاک وML

۵-روشهای ساخت کامپیوتر کدامند؟(نیمسال دوم ۸۵-۸۶)

الف.از طریق سخت افزار ب.از طریق ماشین مجازی
ج.از طریق ترکیبی د. همه موارد

۶- binding متغیرها به مقادیرشان و متغیرها به محلهای خاصی از حافظه به ترتیب جزء کدام دسته از انقیادها است؟(نیمسال اول ۸۶-۸۷)

الف.زمان اجرا-زمان ترجمه ب.زمان اجرا-زمان پیاده سازی
ج.زمان اجرا-زمان اجرا د.زمان پیاده سازی-زمان اجرا

۷-زبانهای Early binding مناسبترند یا زبانهای Late binding؟(نیمسال اول ۸۶-۸۷)

الف.از دید انعطاف پذیری زبانهای Later binding انعطاف پذیرتر و مناسب تر هستند.
ب.از دید سرعت اجرا زبانهای Early binding سریعتر اجرا شده و مناسب تر هستند.
ج.از دید سرعت اجرا و انعطاف پذیری زبانهای Early binding مناسب تر هستند.
د.موارد الف و ب صحیح هستند.

۸- برای جمله x:=x+10 مجموعه ای از انواع ممکن برای متغیر x در کدامیک از زمانهای انقیاد مشخص می شود؟(نیمسال دوم ۸۶-۸۷)

الف.زمانتعریف زبان ب.زمان پیاده سازی زبان ج.زمان اجرا د.زمان ترجمه

۹-کدام گزینه صحیح است؟(نیمسال دوم ۸۶-۸۷)

الف.انقیاد زودرس به دنبال کارایی و انقیاد دیررس به دنبال قابلیت انعطاف است.

ب.انقیاد زودرس به دنبال قابلیت انعطاف و انقیاد دیررس به دنبال کارایی است.

ج.هر دو انقیاد به دنبال کارایی هستند.

د.هر دو انقیاد به دنبال قابلیت انعطاف هستند.

۱۰-کدام دسته از زبانهای زیر به عنوان زبانهای کامپایلری شناخته می شوند. (نیمسال اول ۸۷-۸۸)

الف.ML,Llisp,C++,C ب.Ada,Pascal,C++,C

ج.Prolog,ML,Java,Lisp د.Smaltalk,Lisp,Fortran,Pascal

۱۱-برای دستور انتساب x:=x+y ، انقیاد هر یک از موارد ذیل در چه زمانی صورت می گیرد؟(نیمسال اول ۸۷-۸۸)

مورد اول:مجموعه ای از انواع ممکن برای متغیر x**مورد دوم:**مقدار متغیر x **مورد سوم:**نوع متغیر x

الف.مورد اول می تواند در زمان تعریف یا زمان ترجمه باشد، مورد دوم در زمان اجرا و مورد سوم در زمان ترجمه می باشد.

ب.مورد اول در زمان تعریف، مورد دوم در زمان اجرا و موردسوم در زمان ترجمه می باشد.

ج.مورد اول در زمان پیاده سازی مورد دوم در زمان اجرا و مورد سوم در زمان ترجمه می باشد.

د.مورد اول در زمان اجرا ، مورد دوم در زمان ترجمه و مورد سوم در زمان پیاده سازی می باشد.

۱۲-در کدام گزینه اغلب انقیادها در آن زبانها دیررس هستند؟(نیمسال اول ۸۷-۸۸)

الف.Lisp,ML ب.Pascal, C ج.Fortran, Lisp د.Fortran , C

۱۳-در زبانهای کامپایلری برای بررسی نوع در زمان اجرا چه نوع تمهیداتی باید صورت گیرد؟

الف.دستوراتی توسط برنامه نویس باید به برنامه اضافه شود.

ب.تمهیداتی لازم نمی باشد و سیستم عامل این وظیفه را انجام می دهد.

ج.توصیف گرهایی که به عنوان بخشی از پیاده سازی اشیا داده ای در نظر گرفته می شوند.

د.نمی توان بررسی نوع در زمان اجرا داشت.

۱۴-برای دستور انتساب X:=X+1476, انقیاد هر یک از موارد ذیل در چه زمانی صورت می گیرد؟(نیمسال دوم ۸۷-۸۸)

مورد اول: نمایش مقدار ثابت 1476**مورد دوم:** خواص عملگر + **مورد سوم:** نوع متغیر x

الف. مورد اول می تواند در زمان اجرای برنامه،مورد دوم در زمان تعریف زبان و مورد سوم در زمان ترجمه باشد.

ب. مورد اول در زمان پیاده سازی زبان،مورد دوم در زمان تعریف زبان و مورد سوم در زمان اجرا باشد.

ج. مورد اول در زمان تعریف، مورد دوم در زمان پیاده سازی زبان و مورد سوم در زمان اجرا باشد.

د. مورد اول در زمان ترجمه، مورد دوم در زمان اجرا و مورد سوم در زمان پیاده سازی باشد.

۱۵- منظور از داده تو کار(Built-in Data) چیست؟(نیمسال دوم ۸۷-۸۸)

الف: داده‌هایی که توسط کلاس‌ها تعریف میشوند.

ب: داده‌هایی که مستقیما توسط سخت افزار تعریف می شوند.

ج: داده‌هایی که در زیر برنامه‌های باز گشتی تعریف میشوند.

د: داده‌هایی که در یک سیستم توزیع شده تعریف میشوند.

۱۶-شکل زیر بیان کننده عملکرد و نقش کدامیک از اجزای مشارکت کننده در فرآیند ترجمه و اجرای زبان است؟(نیمسال اول ۸۸-۸۹)

زیر برنامه	آدرس کامپایل شده	آدرس اجرایی
A	100-400	2100-2400
B	100-300	2401-2601
C	50-600	2602-3152

الف.اسمبلر ب.پیش پردازنده ج.بار کننده د.کامپایلر

۱۷- چگونگی ذخیره آرایه‌ها و توصیف آنها توسط کدامیک از نقش‌های زیر مشخص می شود؟(نیمسال اول ۸۸-۸۹)

الف.مترجم ب.بار کننده ج.برنامه نویس د.ویراستار پیوند

۱۸-زمان انقیاد مجموعه ای از انواع ممکن برای متغیر X کدامیک از موارد زیر است؟(نیمسال اول ۸۸-۸۹)

الف.زمان اجرا ب.زمان ترجمه ج.زمان پیاده سازی د.زمان تعریف زبان

۱۹-در عبارت a:=a/2 زمان انقیاد عدد 2، بر اساس میزان حافظه اشغالی آن چیست؟ (نیمسال دوم ۸۸-۸۹)

الف. زمان تعریف زبان ب. زمان اجرا

ج. زمان پیاده سازی د. زمان تعریف زبان و زمان پیاده سازی زبان

۲۰-در صورتی که داشته باشیم نوع int در زبان C درسیستم ۱۶ بیتی محدوده ت۳۲۷۶۸- تا ۳۲۷۶۷ خواهد بود ، به انقیاد در چه زمانی برمی‌گردد. (نیمسال اول ۸۹-۹۰)

الف. تعریف زبان ب. پیاده سازی ج. اجرا د. ترجمه

۲۱- در کدام یک از زبان‌های زیر عملیات روی رشته‌ها با قابلیت انعطاف بالا طراحی شده است. (نیمسال اول ۸۹-۹۰)

الف. فرترن ب. کوبول ج. ام ال د. ادا

۲۲-انقیاد متغیر به مقدار و به محل خاصی از حافظه به ترتیب چه نوع انقیاد است؟ (نیمسال دوم ۸۹-۹۰)

الف. زمان اجرا – زمان اجرا ب. زمان اجرا – زمان پیاده سازی

ج. زمان اجرا – زمان ترجمه د. زمان پیاده سازی – زمان اجرا

۲۳- در مورد انقیاد زبان ها کدام گزینه صحیح است؟ (نیمسال دوم ۸۹-۹۰)

الف. انعطاف پذیری زبان های با انقیاد دیر هنگام بیشتر ، ولی سرعت اجرای زبانهای با انقیاد زود هنگام بالاتر است .

ب. سرعت اجرا و انعطاف پذیری در زبان های انقیاد زود هنگام مناسب تر است .

ج. سرعت اجرا و انعطاف پذیری در زبان های انقیاد دیر هنگام مناسب تر است .

د. انعطاف پذیری زبان های با انقیاد زود هنگام بیشتر ، ولی سرعت اجرای زبانهای با انقیاد دیر هنگام بالاتر است .

۲۴- در مورد اجرای برنامه ها کدام گزینه صحیح است ؟ (نیمسال دوم ۸۹-۹۰)

الف. در روش شبیه سازی عیب اساسی ، از بین رفتن اطلاعات مربوط به برنامه است .

ب. در روش شبیه سازی عیب اساسی ، بزرگتر بودن برنامه مقصد از برنامه مبدا است .

ج. روش شبیه سازی برای برنامه های مبدا که دارای حلقه هایی برای اجرای دستورات اصلی هستند بهتر است .

د. روش اجرای شبیه سازی ، تمام مزایای روش ترجمه (کامپایل) را دارد .

۲۵-دستور مقابل در زبان پاسکال را در نظر بگیرید. کلیه انقیادهای زمان اجرای مربوط به این دستور کدام است؟(نیمسال اول ۹۰-۹۱)

x=x*y;

الف. مجموعه ای از انواع ممکن برای متغییر x,y ، نوع متغییر x,y.

ب.خواص عملگر *، مقدار متغییر y

ج. مقدار متغییر x، مقدار متغییر y

د. مقدار متغییر y

۲۶- زبان ML و FORTRAN از نظر عمل انقیاد به ترتیب جز چه زبان هایی محسوب می شوند؟(نیمسال اول ۹۰-۹۱)

الف. انقیاد زودرس، انقیاد زودرس ب. انقیاد دیررس، انقیاد زودرس

ج. انقیاد زودرس، انقیاد دیررس د. انقیاد دیررس، انقیاد دیررس

۲۷- مزیت زبان های با انقیاد دیررس نسبت به زبان های با انقیاد زودرس چیست؟(نیمسال اول ۹۰-۹۱)

الف. کارایی ب. سرعت اجرای بالا

ج. افزایش زمان کامپایل برنامه د. انعطاف پذیری

۲۸- در کدامیک از زبان های زیر، انقیاد نوع متغییر در زمان اجرا انجام می شود؟(نیمسال اول ۹۰-۹۱)

الف. FORTRAN ب. PASCAL ج. SMALLTALK د. C

سوالات تشریحی

۱–دستور زیر را در نظر بگیرید.

x=x+10;

انواع انقیاد و زمانهای انقیاد را برای این دستور مشخص نمائید. (نیمسال اول ۸۵–۸۶)

۳–زمانهای انقیاد را نام برده هر کدام را توصیف نمائید؟(نیمسال دوم ۸۵–۸۶)

۴–زمانهای انقیاد را برای مجموعه دستورات زیر مشخص نمایید.(نیمسال دوم ۸۸–۸۹)

```
K:=0;
For (i=0;i<10;i++)
K:=k+1;
```

۵– زمان انقیاد موارد زیر را مشخص نمایید. (نیمسال اول ۸۹–۹۰)

الف. مجموعه ای از انواع قابل قبول برای متغیرها مانند real و integer و غیره

ب. نوع متغیرها

ج. مقدار متغیرها (مقید کردن مقدار خاصی به متغیر)

د. مجموعه ای از مقادیر ممکن برای یک نوع متغیر

۶– زمان انقیاد مجموعه دستورات ذیل چگونه است ؟(۱ نمره) (نیمسال دوم ۸۹–۹۰)

```
K:=0;
For(i=0; i<10; i++ )
   K:=k+1;
```

۲-۹- پاسخنامه سوالات تستی فصل دوم:

سوال	الف	ب	ج	د
۱			❊	
۲		❊		
۳				❊
۴	❊			
۵				❊
۶			❊	
۷				❊
۸	❊			
۹	❊			
۱۰		❊		
۱۱		❊		
۱۲	❊			
۱۳			❊	
۱۴		❊		
۱۵		❊		
۱۶			❊	
۱۷	❊			
۱۸				❊
۱۹			❊	
۲۰		❊		
۲۱			❊	
۲۲	❊			
۲۳	❊			
۲۴	❊			
۲۵			❊	
۲۶		❊		
۲۷				❊
۲۸			❊	

فصل سوم:

اصول ترجمه زبان

آنچه در این فصل خواهید آموخت:

- نحو و معنای زبان
- معیارهای عمومی نحو زبان
  - قابلیت خوانایی
  - قابلیت نوشتن
  - سهولت بازرسی
  - سهولت ترجمه
  - عدم وجود ابهام
- عناصر نحوی زبان
- مراحل ترجمه
  - تحلیل لغوی
  - تحلیل نحوی
  - تحلیل معنایی
  - تولید کد میانی
  - بهینه سازی کد
  - تولید کد نهایی
- خطا پرداز و جدول نمادها
- یک مثال از فازهای کامپایلر
- ابتدا و انتهای کامپایلر
- مفهوم گذر
- حساب λ
- سوالات تستی و تشریحی

۳-۱- نحو و معنای زبان

نحو، یعنی آرایش واژه‌ها به عنوان عناصری از یک دنباله که رابطه بین آنها را نشان می دهد. به عبارت دیگر، ترتیب نمادها را برای ایجاد یک برنامه معتبر را مشخص می کنند. به عنوان مثال دستور $x = y + 2$ دنباله معتبری از نمادها را نشان می دهد ولی $= + xy 2$ دنباله معتبری در زبان C نیست.

به طور کلی می توان گفت هر زبانی از دو قسمت تشکیل شده است: ۱- نحو[۱] ۲- معنا[۲]،که نحو، ساختار بین جملات را و معنا، مفهوم بین جملات را مشخص می کند.

۳-۲- معیارهای عمومی نحو زبان

ویژگی‌های نحوی زبان‌های خوب عبارتند از:

۳-۲-۱- قابلیت خوانایی[۳]

اگر ساختار الگوریتم برنامه و داده‌های برنامه به خوبی مشخص باشد، آن برنامه قابلیت خوانایی دارد و به آن برنامه خود استنادی[۴]هم می گویند یعنی این برنامه بدون مستندات جداگانه قابل درک است. زبان کوبول قابلیت خوانایی بالایی دارد در برنامه‌های نوشته شده به این زبان، ساختارهایی که کار یکسانی انجام می دهند مشابه اند و ساختارهایی که کارهای متفاوتی انجام می دهند مختلف هستند. زبان‌هایی با ساختار نحوی اندک قابلیت خوانایی کمتری دارند مثلاً APL وSNOBOL4 فقط یک فرمت دستور دارند و خوانایی آنها کم است.

۳-۲-۲- قابلیت نوشتن[۵]

خصوصیات نحوی یک زبان که نوشتن برنامه را ساده تر می کند اغلب با خصوصیات نحوی که خوانایی را بیشتر می کند در تضاد هستند قابلیت نوشتن با استفاده از ساختارهای منظم ودقیق حاصل می شود در حالیکه ساختارهای طولانی برای قابلیت خواندن مفید هستند مثلاً در زبان C برنامه‌های دقیقی نوشته می شوند که ویژگی های مفید متعددی دارند که باعث می شود قابلیت نوشتن افزایش یابد ولی قابلیت خوانایی آن کاهش می یابد.

قواعدی که اجازه می دهند اعلان‌ها و عملیات تعیین نشده ای در زبان وجود داشته باشند مثل تعریف آرایه با طول متغیر که نوشتن برنامه را ساده تر می کنند ولی خواندن آن را مشکل تر می کنند. به عنوان مثال در زبان فرترن به طور پیش فرض متغییرهایی که با حروف I,J,K ,…,N شروع می شوند اگر اعلان نشوند از نوع صحیح خواهند بود که این کار نوشتن برنامه توسط برنامه نویس را آسان می کند ولی قابلیت خوانایی آن به دلیل عدم اعلان متغییر کاهش می یابد. برخی ویژگی‌های دیگر مانند استفاده از دستوارات ساخت یافته هر دو قابلیت خواندن و نوشتن را افزایش می دهند.

۳-۲-۳- سهولت بازرسی (تست)

صحت برنامه یا بازرسی برنامه با قابلیت خوانایی و قابلیت نوشتن در ارتباط است. زبانی خوب است که تست کردن صحت آن ساده تر باشد.

۳-۲-۴- سهولت ترجمه

قابلیت خوانایی و نوشتن ملاک‌های مورد نظر برنامه نویس می باشند، در حالیکه سهولت ترجمه، نیاز مترجمی است که قرار است برنامه نوشته شده را پردازش و ترجمه کند. این خاصیت با ویژگی قابلیت خوانایی و نوشتن نسبت عکس دارد. هرچه نظم

---

[1] Syntax
[2] Semantic
[3] Readability
[4] Self-documentig
[5] Writeability

موجود در ساختار زبان بیشتر باشد، سهولت ترجمه آن بیشتر است به عنوان مثال ترجمه زبان لیسپ ساده است چون نظم ساختاری و قواعد ساده ای دارد در حالیکه قابلیت خواندن و نوشتن آن کم است. هنگامی که ساختارهای نحوی در یک زبان بیشتر باشند ترجمه آن نیز سخت تر می شود که نمونه آن زبان کوبول است.

۳-۲-۵- عدم وجود ابهام

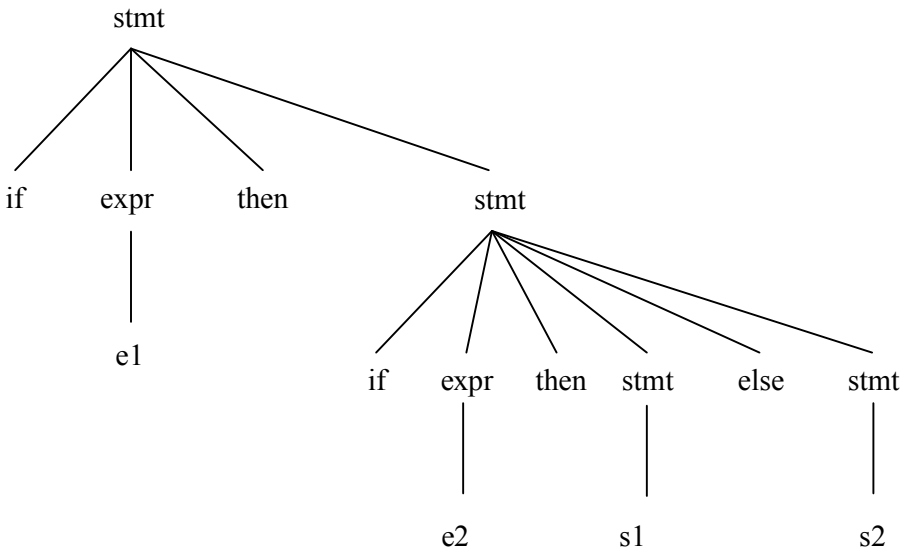
یک زبان خوب نباید شامل دستورات مبهم باشد مثلاً در دستور if تودرتوی زیر در یک زبان فرضی :

```
If (e1) then if (e2) then s1 else s2
```

در دستور بالا معلوم نیست که آیا else مربوط به if اولی است یا مربوط به if دومی. در زبان C و پاسکال ابهام مذکور با این قانون بر طرف شده است که هر else به نزدیک ترین if بر می گردد و در مثال فوق else مربوط به if (e2) است.
برای درک بهتر مثال فوق گرامر زیر را در نظر بگیرید:

Stmt → if expr then stmt |
if expr then stmt else stmt | other

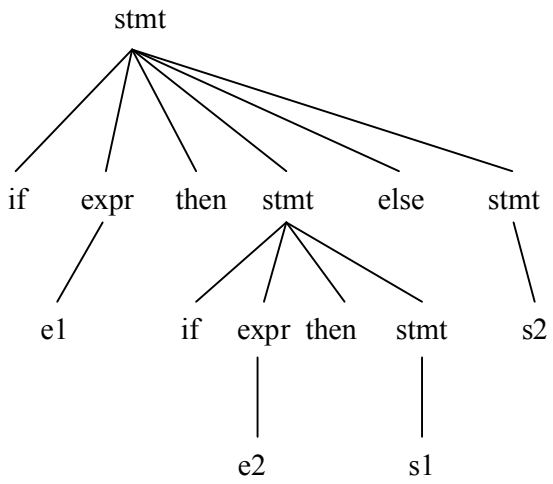
ابهام[1]: بدین معنی است که ممکن است از روی یک گرامر چند درخت پارس متفاوت برای یک جمله ساخته شود.
دو درخت پارس متفاوت برای جمله روبرو وجود دارد: If e1 then if e2 then s1 else s2
در این حالت else مربوط به if دومی است



شکل ۳ -۱

[1] ambiyuity

ولی در این حالت else مربوط به if اولی است



شکل ۳ -۲

همان طور که از قبل می دانید برای رفع ابهام در عبارات ریاضی یا محاسباتی، از اولویت[1] برای عملگرهایی با اولویت متفاوت و از شرکت پذیری[2] برای عملگرهایی با اولویت یکسان استفاده می کردیم. برای رفع ابهام در دستورات، معمولاً از بازگشتی از چپ و فاکتورگیری از چپ استفاده می شود که برای گرامر فوق می توان از عمل فاکتور گیری چپ استفاده کرد.

عمل فاکتور گیری از چپ

اگر برای یک غیر پایانه (متغیر) قواعدی وجود داشته باشد که با جملات یکسانی شروع شود می توان از بخش شروع یکسان فاکتور گیری کرد که به آن فاکتور گیری از چپ گفته می شود و قسمت غیر مشترک را تحت یک اسم دیگر می نویسیم.

$$\text{stmt} \rightarrow \text{if expr then stmt} \mid \text{if expr then stmt else stmt}$$

فاکتورگیری چپ:

$$\text{stmt} \rightarrow \text{if expr then stmt}'$$
$$\text{stmt}' \rightarrow \text{else stmt}$$

مثالی دیگر از ابهام این است که در زبان فرترن نماد A(i,j) هم برای فراخوانی تابع و هم برابر ارجاع آرایه استفاده می شود و در فرترن برای رفع این ابهام قرارداد شده است که اگر آرایه ای به نام A، اعلان نشده باشد منظور از A(i,j) فراخوانی تابع است. این ابهام در زبان پاسکال وجود ندارد چون در پاسکال برای ارجاع آرایه به جای پرانتز از براکت استفاده می شود. مانند A[i,j] و از این نظر پاسکال بهتر از فرترن است.

[1] Precedence
[2] Associative

نکته : معیارهای ارزیابی زبان برنامه نویسی و فاکتورهایی که بر آن تاثیر می گذارند در جدول ۳ -۱ نشان داده شده است.

فاکتورهای تأثیرگذار	قابلیت خوانایی (readability)	قابلیت نوشتن (writeability)	قابلیت اطمینان (reliability)
طراحی گرامر زبان	×		×
ساختارهای کنترل	×	×	×
انواع و ساختمان داده‌ها	×	×	×
سادگی و قابلیت تعامد	×	×	×
پشتیبانی از تجرید		×	×
قابل بیان بودن		×	×
کنترل نوع			×
کنترل استثناها و خطاها			×
محدود بودن نام گذاری مستعار			×

جدول ۳ -۱

۳-۳- عناصر نحوی زبان

مهمترین عناصر نحوی یک زبان برنامه نویسی عبارت اند از :

۳-۳-۱- کاراکترها

اولین کار در طراحی نحو یک زبان انتخاب مجموعه کاراکترهای (الفبای زبان) آن زبان است. در گذشته برای نمایش کاراکترها از ۸ بیت استفاده می کردند که توانایی نمایش ۲۵۶ حالت را داشت و برای حروف بزرگ وکوچک انگلیسی شامل ۵۲ کاراکتر و نمادهای دیگر کفایت می کرد ولی با بین المللی شدن صنعت کامپیوتر و پشتیبانی آن از زبان‌های مختلف دنیا به نظر می رسد ۲۵۶ حالت کافی نیست و به جای ۸ بیت از ۱۶ بیت برای نمایش کاراکترها استفاده می کنند.

۳-۳-۲- شناسه‌ها[1]

در اغلب زبان‌ها شناسه‌ها رشته ای از حروف وارقام است که با حرف شروع می شوند. در فرترن اولیه طول شناسه حداکثر ۶ کاراکتر بود که این امر خوانایی برنامه را کاهش می دهد.

۳-۳-۳- نمادهای عملگرها

اغلب زبان‌ها از کاراکترهای + و/ و *و- برای اعمال محاسباتی استفاده می کنند. در برخی از زبان‌ها مانند لیسپ برای اعمال اولیه از شناسه‌ها استفاده می شود مثل plus و times درفرترن برای تساوی .EQ. و برای توان از ** استفاده می شود.

۳-۳-۴- کلمات کلیدی و کلمات رزروی

کلمه کلیدی، شناسه ای است که به عنوان بخش ثابتی از نحو یک دستور استفاده می شود مثل کلمه کلیدی "if" که بخش ثابتی از نحو یک دستور است. اگر کلمه کلیدی توسط برنامه نویس به عنوان یک شناسه (اسم متغیر و یا تابع) قابل استفاده نباشد به آن کلمه رزروی گفته می شود. اکثر زبان‌ها از کلمات رزروی استفاده می کند تا خطایابی ساده تر شود. تنها ایراد کلمات رزروی، توسعه زبان و ایجاد کلمات رزروی جدید است. اضافه کردن کلمه رزروی جدید به یک زبان به این معنا است

[1] identifier

که برنامه‌های قدیمی که از آن کلمه به عنوان متغیر استفاده می کردند دیگر از نظر نحوی درست نمی باشند و این یک مشکل است.

۳-۳-۵- کلمات اضافی[1]

کلماتی اضافی (اختیاری) هستند که جهت افزایش قابلیت خوانایی زبان در دستورات قرار می گیرند. زبان کوبول کلمات اضافی زیادی دارد. مثلاً دردستور go to در کوبول وجود go لازم ولی نوشتن to اختیاری است و فقط جهت افزایش خوانایی در دستور گنجانده شده است.

۳-۳-۶- تو ضیحات[2]

توضیحات موجود در هربرنامه مهمترین بخش مستند سازی آن برنامه به حساب می آیند توضیحات در زبان‌ها با شکل‌های متفاوتی ممکن است ظاهر شوند مثلاً در زبان C از /* توضیحات */ و در C++ از // استفاده می شود.

۳-۳-۷- فضای خالی[3]

در اغلب زبان‌ها، فضای خالی به عنوان جدا کننده استفاده می شود. در فرترن فضای خالی معنای خاصی ندارد و فقط در رشته-ها به عنوان blank استفاده می شود در زبان Snobol 4 فضای خالی هم به عنوان جداکننده عناصر یک دستور استفاده می شود و هم به عنوان عملگر الحاق دررشته‌ها استفاده می شود.

۳-۳-۸- جداکننده‌ها و محصور کننده‌ها[4]

یک عنصر نحوی است که برای نشانه گذاری ابتدا یا انتهای یک واحد نحوی مثل یک دستور یا عبارت به کار می رود. محصور کننده‌ها، جداکننده‌های جفتی هستند مثل جفت پرانتزها و یا begin ... end– جداکننده‌ها برای قابلیت خوانایی یا سهولت در تحلیل نحوی به کار می روند اما اغلب برای از بین بردن ابهام و تعیین مرزها مورد استفاده قرار می گیرند.

۳-۳-۹- فرمت‌های آزاد و طول ثابت

یک نحو در صورتی فرمت آزاد است که دستورات در هرجایی از خط شروع شوند. نحو فرمت ثابت از موقعیت‌های خاصی از خط استفاده می کند مثلا در اسنوبال ۴، برچسب‌های دستورات توضیحات و.... کاراکتر ویژه ای در ابتدای خط مشخص می شوند. در اسمبلی نیز از فرمت ثابت استفاده می شود که هر عنصر یک دستور باید در بخش خاصی از خط قرار گیرد ولی امروزه به ندرت از نحو فرمت ثابت استفاده می شود.

۳-۳-۱۰- عبارات

توابعی هستند که به اشیای داده موجود در برنامه دسترسی دارند و مقداری را برمی گردانند دستورات از عبارات ساخته می شوند. در زبان‌های دستور ی مثل C عبارات عملیات اصلی برای تغییر حالت ماشین هستند در زبان‌های تابعی مانند ML و Lisp عبارات کنترل ترتیب اجرای برنامه را مشخص می کنند.

۳-۳-۱۱- دستورات

مهمترین جز نحوی در زبان‌های دستوری می باشند دستورات اسنوبال ۴، فقط یک نحو دارند و این زبان به نظم اهمیت می دهد. دستورات کوبول فرمت‌های مختلفی دارد و به خوانایی اهمیت می دهد. دستورات می توانند ساخت یافته (تو در تو) یا ساده باشند دستور ساده هیچ دستور دیگری را شامل نمی شود. مثلاًAPL ،Snobal 4 از دستورات ساده استفاده می کنند.

---

[1] noise-word
[2] remark-comment
[3] blank
[4] delimiters and brockets

۳-۴- مراحل ترجمه

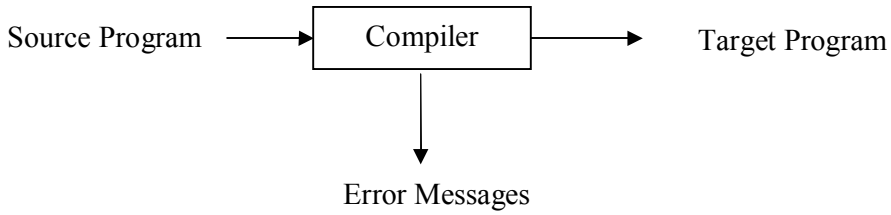
ترجمه، یعنی برنامه ای از یک زبان به یک زبان دیگر تبدیل شود. ترجمه یک برنامه ممکن است بسیار ساده باشد مانند برنامه-های پرولوگ و لیسپ. اما اغلب فرایند ترجمه بسیار پیچیده است فرایند ترجمه را به طور منطقی می توان به دو مرحله تقسیم کرد :

- تحلیل[۱] برنامه ورودی
- ترکیب[۲] برنامه مقصد

مرحله تحلیل: شامل فازهای تحلیل لغوی، تحلیل نحوی، تحلیل معنایی و تولید کد میانی می باشد و مرحله ترکیب (تولید) شامل فازهای بهینه سازی کد و تولید کد نهایی می باشد.

کامپایلر: کامپایلر نرم افزاری است که برنامه نوشته شده در یک زبان به نام زبان منبع[۳] را خوانده و از روی گرامر آن ساختار برنامه را بدست آورده و آن را به برنامه معادل در زبان دیگر به نام زبان مقصد[۴]ترجمه می کند.

در صورتی که برنامه نوشته شده به زبان مبدا با گرامر آن مطابقت نداشته باشد خطایی را صادر می کند.



شکل ۳ -۳

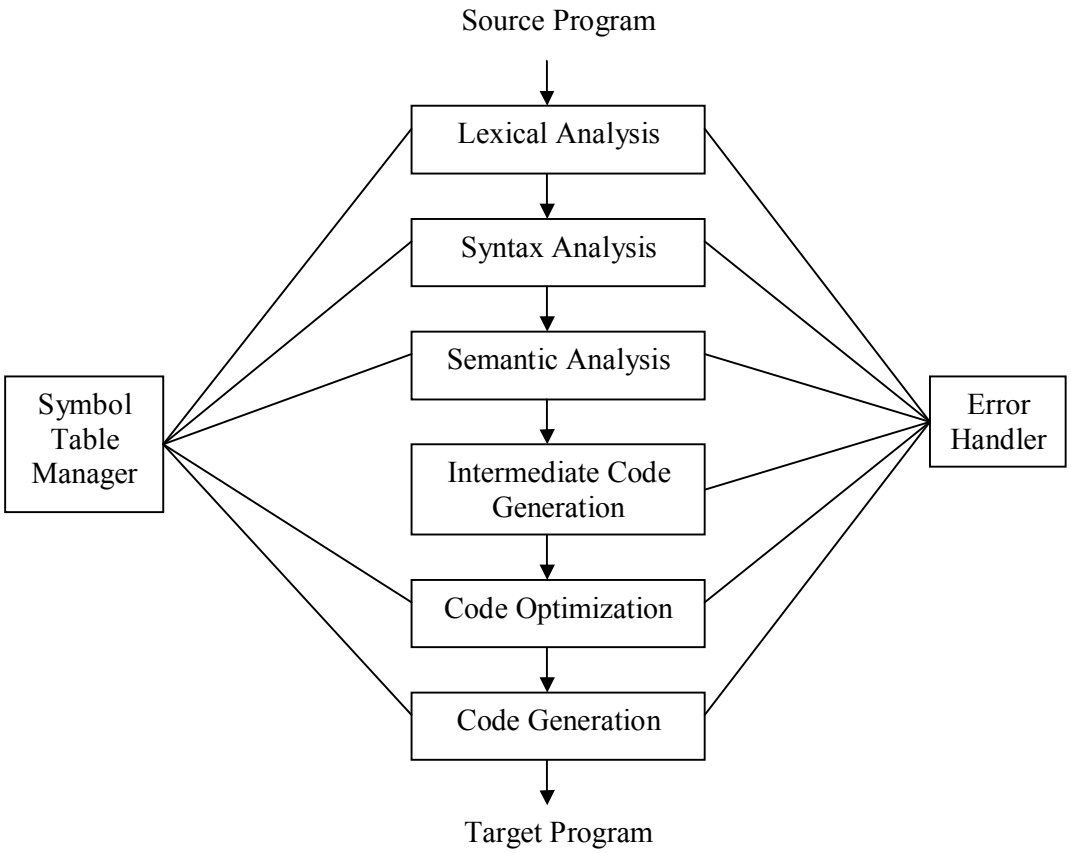
مراحل ترجمه (کامپایل) :

عملیات ترجمه در شش مرحله صورت می گیرد :

- تحلیللغوی[۵]
- تحلیل نحوی[۶]
- تحلیل معنایی[۷]
- تولید کد بینابینی[۸]
- بهینه سازی کد[۹]
- تولید کد نهایی[۱۰]

---

[1]analysis
[2]synthesis
[3]Source Language
[4]Target Language
[5]Lexical Analysis
[6]Syntax Analysis
[7]Semantic Analysis
[8]Intermediate Code Generation
[9]Code Optimization
[10]Code Generation



شکل ۳ -۴

ارتباط بین این مراحل در شکل زیر نشان داده شده است در کنار شش مرحله اصلی کامپا یلر، دو بخش دیگر به نام خطا پرداز و جدول علایم نیز وجود دارد.

۳-۴-۱- تحلیل لغوی

در مرحله اول یعنی تحلیل لغوی، برنامه ورودی کاراکتر به کاراکتر خوانده شده و به دنباله ای از نشانه‌ها[1] تبدیل می گردد. انواع مختلف نشانه‌ها عبارتند از: کلمات کلیدی[2]، عملگرها[3]، جدا کننده‌ها[4]، ثابت‌ها[5]،شناسه‌ها[6] که به اسامی توابع ، رویه‌ها و به طور کلی اسامی که کاربر انتخاب می‌کند گفته می‌شود. در اغلب زبانهای برنامه سازی کلمات کلیدی رزرو شده اند، بدین معنی که کاربر مجاز نیست از هیچ یک از آن‌ها به عنوان اسم یک متغییر، تابع و یا رویه استفاده کند اما در برخی زبان‌ها مثل PL/I این محدودیت وجود ندارد. مدل اصلی که برای طراحی تحلیل گر لغوی استفاده می شود ماشین خودکار متناهی[7] می باشد با اینکه مفهوم تحلیل لغوی ساده است ولی این مرحله بسیار زمان بر است که علت آن خواندن و بررسی تمام کاراکترهای برنامه است.

[1] tokens
[2] Keywords
[3] Operators
[4] Delimeters
[5] Literals
[6] Identifiers
[7] FSA

۳-۴-۲- تحلیل نحوی (تجزیه)

در این مرحله، ساختارهای بزرگ مانند دستورات، اعلان‌ها، عبارات و.... با استفاده ازعناصر لغوی که توسط تحلیل گر لغوی تولید شده اند شناسایی می شوند. بنابراین در این مرحله، برنامه با استفاده از زبان مبدا از نظر خطاهای نحوی مورد بررسی قرارمی گیرند و با استفاده از نشانه‌های تولید شده در مرحله تحلیل لغوی یک درخت پارس[۱] ایجاد می گردد.

۳-۴-۳- تحلیل معنایی

تحلیل معنایی، مهم ترین مرحله ترجمه است در این مرحله با استفاده از درخت تولید شده در مرحله قبلی، برنامه ورودی از نظر خطاهای مفهومی احتمالی مورد بررسی قرار می گیرد این مرحله پلی بین بخش تحلیل و ترکیب ترجمه است. در این مرحله اعمال جنبی دیگری نظیر نگهداری جدول نمادها، کشف خطاها، بسط ماکروها و اجرای دستورات زمان ترجمه نیز انجام می شود. یکی از مهم ترین کارها در تحلیل معنایی، کنترل نوع می باشد.

۳-۴-۴- تولید کد میانی

در این مرحله یک برنامه که معادل برنامه اصلی است به یک زبان میانی تولید می شود. با ایجاد کدمیانی، عملیات بعدی که کامپایلر باید انجام دهد آسان می گردد. بعضی کامپایلرها، نمایش میانی سریعی از برنامه مبدا دارند. این نمایش میانی باید دارای ۲ خاصیت زیرباشد :

- به آسانی بتوان ان کد میانی را تولید و بهینه سازی کرد.
- ترجمه کد میانی به برنامه مقصد به راحتی صورت پذیرد.

نمونه ای از این کدهای میانی،کد ۳آدرسه[۲] است که شبیه به زبان اسمبلی می باشد.

۳-۴-۵- بهینه سازی کد

در این مرحله سعی می شود تا کد میانی تولید شده در مرحله قبلی به نحوی بهبود داده شود این کار سبب تولید کدی می شود که از لحاظ اجرایی سریع تر و حافظه کمتری مصرف می کند.

۳-۴-۶- تولید کد نهایی

در این مرحله، هرکدام از کدهای میانی بهبود یافته به مجموعه ای از دستورات ماشین که عملکرد مشابهی دارند تبدیل می شود. بنابر این در آخرین فاز کامپایلرها یعنی تولید کد، به هریک از متغییرهای موجود در برنامه حافظه تخصیص داده می شود و متغیرها در ثبات‌ها جایگزین می شوند.

۳-۵- خطا پرداز[۳]

هر فاز از کامپایلر باید به گونه ای رفتار کند که ادامه کامپایل و کشف خطاهای بیشتری را میسر سازد. کامپایلری که با اولین خطا متوقف می شود ناکار آمد است. فازهای تحلیل لغوی و معنایی بخش عمده ای از خطاهایی که توسط کامپایلر کشف می شوند را پیدا می کنند. فاز لغوی می تواند خطاهایی را شناسایی کند که در آن رشته کاراکترها مطابق هیچ الگویی از توکن‌ها نیست خطاهایی که قوانین ساختاری (گرامری) دستور زبان را نقض می کنند در فاز تحلیل نحوی کشف می شود.

[1] ParsTree
[2] Triple code
[3] Error handler

۳-۶- جدول نمادها[۱]

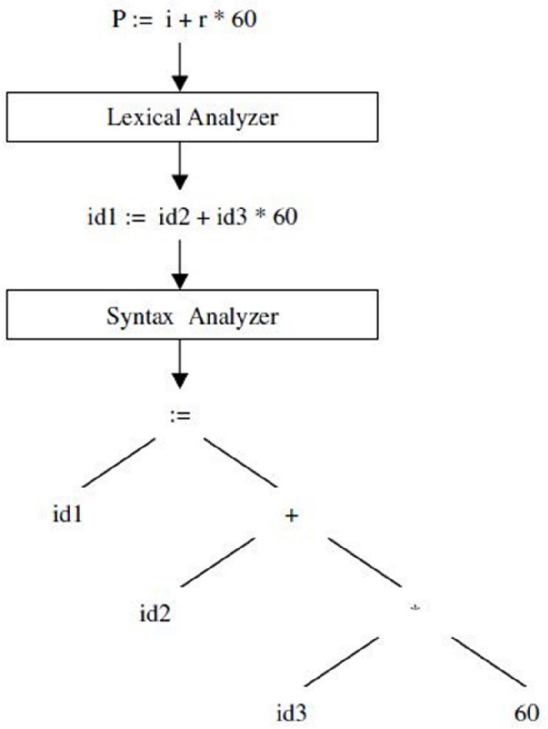
یکی از کارهای مهم واساسی یک کامپایلر، ثبت شناسه‌های استفاده شده در برنامه ورودی (منبع) و جمع آوری اطلاعات درباره مشخصات هر شناسه است. این مشخصات می تواند شامل: آدرس حافظه اختصاص داده شده به شناسه، نوع آن، حوزه[۲] یعنی محلی از برنامه که این شناسه در آن تعریف شده است و در رابطه با رویه‌ها، اسم آنها، تعداد و نوع آرماگون‌های آنها، روشی که طی آن آرماگون‌ها به زیر برنامه فرستاده می شوند مثل : Call by reference یا Callbyname و نوع نتیجه ای که رویه‌ها باز می گردانند

در جدول نمادها به ازای هر شناسه یک رکورد وجود دارد که این رکوردها شامل مشخصات شناسه‌ها می باشند این جدول امکان دستیابی سریع به شناسه‌ها و مشخصات آنها را فراهم می کند. در مرحله تحلیل لغوی، تحلیل گر لغوی کامپایلر، شناسه ای را که در برنامه مبدا پیدا می کند آن را در جدول نمادها درج می کند ولی خصیصه‌های مربوط به شناسه‌ها نمی توانند در هنگام تحلیل لغوی وارد جدول نمادها شوند. بلکه در دیگر فازها، اطلاعات مربوط به این شناسه‌ها به جدول اضافه خواهند شد و در مراحل مختلف تولید کد از آن‌ها استفاده خواهد شد. به عنوان مثال در تحلیل معنایی و تولید کد میانی لازم است نوع شناسه را بدانیم و یا در بخش تولید کل باید جزئیات بیشتری از تخصیص فضا به هر شناسه را بدانیم.

۳-۷- یک مثال جامع برای فازهای کامپایلر

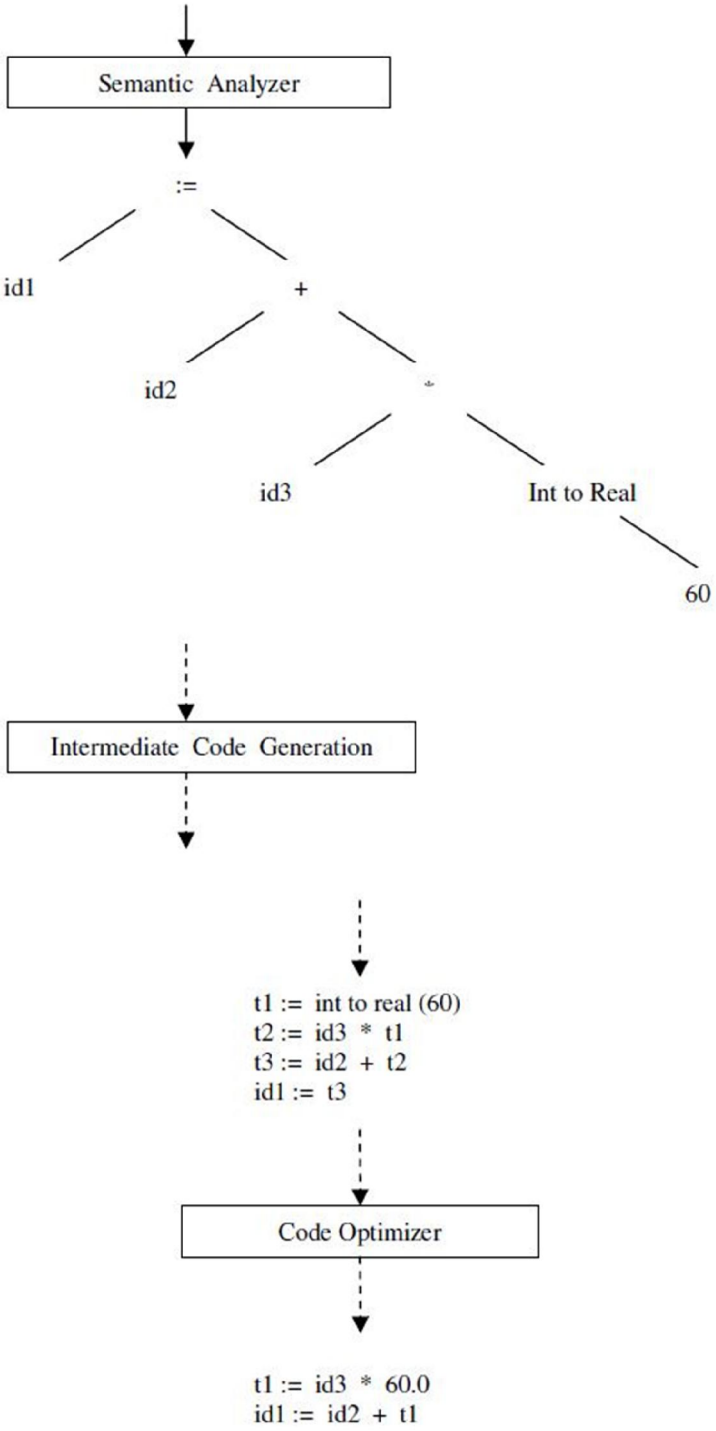
با یک مثال کل فازهای کامپایلر را بررسی می کنیم.

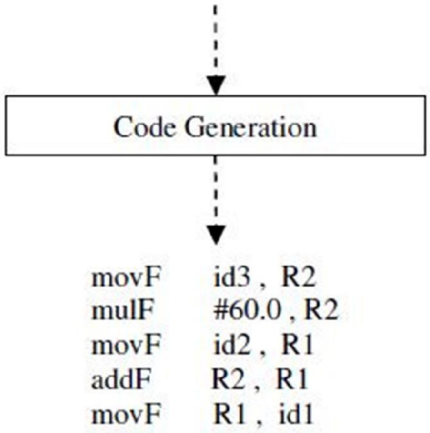
(فرض شده که r، p، i همگی از نوع real هستند) `p:i+r×60`



[1] Symbol table

[۲] Scope


Semantic Analyzer
:=
id1
+
id2
id3
Int to Real
60
Intermediate Code Generation
t1 := int to real (60)
t2 := id3 * t1
t3 := id2 + t2
id1 := t3
Code Optimizer
t1 := id3 * 60.0
id1 := id2 + t1



شکل ۳ -۵

۳-۸- ابتدا (front-end) و انتهای (back-end) کامپایلرها

به چهارمرحله اول کامپایلر و بخشی از مرحله بهینه سازی کد که به زبان مبدا وابسته و مستقل از زبان ماشین مقصد هستند ابتدای (front-end) کامپایلر گفته می شود. به بخشی از مرحله بهینه سازی و مرحله آخر کامپایلر که وابسته به ماشین مقصد هستند انتهای (back-end) کامپایلر گفته می شود این بخش (انتهایی) از کامپایلر وابسته به زبان مبدا نیست.

۳-۹- مفهوم گذر[۱]

به هر مرتبه خواندن برنامه ورودی (به شکل اولیه و یا میانی) از ابتدا تا انتها و توسط هر یک از قسمت‌های کامپایلر یک گذر گفته می شود. کامپایلرها از نقطه نظر تعداد گذرهای مورد نیاز، جهت انجام عمل ترجمه به دو دسته‌ی تک گذره[۲]وچند گذره[۳]تقسیم می شوند. یک کامپایلر تک گذره، کلیه عملیات ترجمه ورودی را تنها با یک مرتبه خواندن ورودی انجام می دهد. وجود مرحله بهینه سازی یا برخی از ویژگی‌های زبان‌های برنامه سازی نظیر این خاصیت کد تعریف رویه‌ها بتوانند پس از فراخوانی آن‌ها قرار داده شود، سبب می شود کامپایلر به گذرهای بیشتری برای انجام عملیات خود نیاز داشته باشد.

۳-۱۰- حساب لاندا

احتمالاً اولین مدل معنایی زبان برنامه نویسی، حساب لاندا بوده است که در دهه ۱۹۳۰ توسط کورچ به عنوان مدل تئوری محاسبات در مقایسه با ماشین تورینگ مطرح شد حساب لاندا مدل خوبی را برای فراخوانی تابع زبان برنامه سازی ارائه کرد. در واقع الگول و لیسپ می توانند معنای فراخوانی تابع را با مدل حساب لاندا ردیابی کنند.
عبارت لاندا به طور باز گشتی و به صورت زیر تعریف می شوند.

اگر X نام متغییر باشد ، X یک عبارت لاندا است.
اگر M یک عبارت لاندا باشد و $\lambda x.m$ یک عبارت لاندا است.
اگر F و A عبارت لاندا باشند، (FA)عبارت لاندا است که F یک عملگر و A یک عملوند است.
ادامه این مبحث را از کتاب مطالعه فرمائید.
فصل ۳: صفحات ۶۰-۷۵
فصل ۴: کل فصل

---

[۱] pass
[۲] single pass
[۳] multi pass

۳-۱۱- سوالات فصل سوم

سوالات تستی فصل سوم

۱-کدام گزینه غلط است؟(نیمسال دوم ۸۳)

الف. ابهام مسئله ای است که همراه نحو زبان وجود دارد

ب. الگوریتم‌هایی مشخص جهت رفع ابهام زبان وجود دارد

ج. وقتی یک رشته در زبان دارای بیش از یک درخت تجریه باشد گرامر مبهم است.

د. عبارات منظم شکل دیگری را برای تعریف زبان ارائه می‌کنند که هم ارز گرامرهای FSA و منظم می‌باشد.

۲-کدام گزینه غلط نیست؟(نیمسال دوم ۸۳)

الف. کاربرد زبان‌هایی که فاقد افزونگی هستند دشوار است.

ب. از ساختارهای مبهم دو یا چند تفسیر بعمل می‌آید

ج. ابهام یک مسئله مهم در طراحی هر زبان است.

د. هیچکدام

۳-کدام گزینه غلط است ؟(نیمسال دوم۸۴)

الف. ساختار فرترن طوری است که ترجمه زیر برنامه‌های مجزا ساده می‌باشد

ب. در زبان SNOBOL تمایز نحوی بین دستورات برنامه اصلی و دستورات زیر برنامه وجود ندارد.

ج. تعریف زیر برنامه‌های تودرتو در پاسکال امکان پذیر نمی‌باشد.

د. تحلیل لغوی اولین مرحله ترجمه می‌باشد.

۴- در کدام یک از ماشینهای پذیرنده زیر حالت قطعی و غیر قطعی یکسان نیستند؟(نیمسال اول ۸۵-۸۶)

الف. ماشین خودکار متناهی ب.ماشین خودکار پشته ای

ج. ماشین خودکار خطی د.ماشین تورینگ

۵- در کدام یک از مراحل ترجمه یک زبان از ماشین خودکار متناهی استفاده می شود؟(نیمسال اول ۸۵-۸۶)

الف.بهینه سازی ب.تحلیل معنایی

ج.تحلیل لغوی د.تحلیل نحوی

۶- کدام گزینه جزو معیارهای نحو عمومی است؟(نیمسال دوم ۸۵-۸۶)

الف. قابلیت خواندن و نوشتن ب. سهولت بازرسی و ترجمه

ج. موارد الف وب د. وجود ابهام

۷- کدام گزینه جزو مراحل ترجمه یک برنامه می باشد؟(نیمسال دوم ۸۵-۸۶)

الف.تحلیل لغوی ب.تحلیل نحوی

ج.تحلیل معنایی د. همه موارد

۸- کدام گزینه غلط می باشد؟(نیمسال دوم ۸۵-۸۶)
الف.ابهام مسئله ای است که همراه نحو وجود دارد.
ب.BNF توسط جان باکوس در اواخر دهه ۱۹۶۰ ایجاد شد.
ج.اگر گرامر مربوط به یک زبان مبهم باشد زبان مبهم است.
د.گرامرهای منظم حالت‌های خاصی از گرامرهای BNF می باشد.

۹- کدام گزینه غلط است؟(نیمسال دوم ۸۵-۸۶)
الف.برای گرامرهای منظم همیشه یک ماشین خودکار قطعی وجود دارد.
ب. PDAهای قطعی هم ارز گرامرهای LR(K) هستند.
ج.گرامرهای منظم نمی توانند رشته‌هایی به شکل d^n تولید نمایند.
د.زبان نوع n توسط گرامر نوع n تولید می گردد.

۱۰- کدام مورد از معیارهای نحو نیست؟(نیمسال اول ۸۶-۸۷)
الف.قابلیت خوانایی و قابلیت نوشتن ب.قابلیت حمل
ج.عدم وجود ابهام د.سهولت ترجمه

۱۱- وظیفه تحلیلگر لغوی چیست؟(نیمسال اول ۸۶-۸۷)
الف.شناسائی نشانه‌ها ب.تعبیر عملگرها
ج.پردازش ماکرو د.موارد الف و ب درست است.

۱۲- کدام گزینه صحیح است؟(نیمسال دوم ۸۶-۸۷)
الف.خروجی تحلیل معنایی، تحویل تولید کد می شود.
ب.خروجی تحلیل معنایی، تحویل بهینه سازی می شود.
ج.خروجی تحلیل نحوی، تحویل لغوی می شود.
د.خروجی تحلیل معنایی، تحویل نحوی می شود.

۱۳-کدام گزینه غلط است؟(نیمسال دوم ۸۶-۸۷)
الف.قابلیت خوانایی و قابلیت نوشتن در جهت عکس هم حرکت می کنند.
ب.ممکن است زبانی باشد که ترجمه آن آسان باشد ولی قابلیت خوانایی و قابلیت نوشتن آن پایین باشد.
ج.افزایش تعداد ساختارهاینحوی، کار ترجمه را ساده تر می کند.
د)هیچکدام

۱۴- اغلب مترجم زبان جدید، به همان زبان نوشته می شود. از طریق کدام عمل زیر مشکل ترجمه زبان جدید حل می شود؟(نیمسال دوم ۸۶-۸۷)
الف.بافرینگ ب.پردازش دسته ای ج.خودرانی د.خود استنادی

۱۵– زبانی خاص از کلمات اختیاری در دستورات خود استفاده می کند تا قابلیت خوانایی را بهبود بخشد. مثلاً فرض کنید زبان خاص در دستور go to اجازه دهدto بیاید یا نیاید وgo حتماً بیاید به این کلمات اختیاری اصطلاحاً چه می گویند؟(نیمسال اول ۸۷–۸۸)

الف.خود استنادی ب.پارازیت ج.جداکننده د.پکیج

۱۶– کدام گزینه زیر صحیح است؟(نیمسال دوم ۸۷–۸۸)

الف. هر زبانی که قابلیت خوانایی بالایی دارد حتما قابلیت نوشتن بالایی نیز دارد.

ب. قابلیت نوشتن بوسیله ساختارهای طولانی مفید به دست می آید.

ج. زبانهایی که ساختارهای نحوی اندکی را ارائه می کنند برنامه‌هائی با خوانائی پایین تر تولید می کنند.

د. نحو موجود در زبانLISP ترجمه بسیار پیچیده ای نیاز دارد.

۱۷– زمانی که ترجمه مستقل در طراحی زبان مد نظر است بکار بردن اسامی مشترک موجب می شود چندین زیر برنامه یا واحدهای دیگری از برنامه همنام باشند زبانهای شی گرا چگونه این مشکل را حل می کند؟(نیمسال دوم ۸۷–۸۸)

الف: هر نام مشترک باید منحصر به فرد باشد و برنامه نویس مسئول این کار است.

ب: تنها از قواعد حوزه برای پنهان کردن اسامی استفاده می شود

ج: اسامی ممکن است در یک کتابخانه خارجی ذخیره شوند.

د: اسامی به صورت ثابت تعریف شوند.

۱۸–کدام یک از موارد زیر جزء مراحل ترجمه است ؟(تابستان ۸۸)

الف. تولید کد ب. تحلیل نحوی

ج. بهینه سازی د. همه موارد

۱۹– زبان‌هایی که ساختارنحوی اندکی ارائه می کنند..... (نیمسال اول ۸۸–۸۹)

الف.برنامه‌هایی با خوانایی بالاتر تولید می کند.

ب.برنامه‌هایی با خوانایی پایین تر تولید می کند.

ج.برنامه‌هایی با قابلیت نوشتن پایین تر تولید میکند.

د.این موضوع تاثیری در قابلیت خواندن و نوشتن ندارد.

۲۰– نگهداری جدول نمادها ، بسط ماکروها و اجرای دستورات زمان ترجمه در کدام مرحله از مراحل کامپایلر صورت می گیرد؟(نیمسال اول ۸۸–۸۹)

الف.تحلیل معنایی ب.تحلیل نحوی ج.تحلیل لغوی د.بهینه سازی

۲۱– کدام بخش از نحو زبان را نمی توان توسط گرامر BNF تعریف نمود؟(نیمسال اول ۸۸–۸۹)

الف.تعریف یک شناسه در بلوک خود ب.آنهایی که وابسته به متن هستند.

ج.آنهایی که قابل بهینه سازی نیستند. د.مواردی که وابسته به متن نیستند.

۲۲- منظور از راه اندازی خودکار چیست؟(نیمسال اول ۸۸-۸۹)

الف.ساخت کامپایلر زبان توسط همان زبان برنامه سازی

ب.تولید کامپایلر زبان توسط مفسر زبان دیگر

ج.ساخت مفسر زبان بوسیله کامپایلر زبان دیگر

د.تبدیل زبانهای برنامه سازی مختلف به دیگر

۲۳- در کدامیک از ساختارهای برنامه و زیر برنامه، تمایز نحوی بین دستورات برنامه اصلی و دستورات زیر برنامه وجود ندارد؟(نیمسال دوم ۸۸-۸۹)

الف.تعریف زیر برنامه‌ها بطور غیر مجزا ب. توصیف داده‌ها جدا از دستورات اجرایی

ج.تعریف داده‌ها بطور مجزا د. تعریف واسط مجزا

۲۴- کدام گزینه در مورد معیارهای نحو صحیح نمی باشد ؟ (نیمسال دوم ۸۹-۹۰)

الف. قابلیت حمل ب. قابلیت خوانایی و نوشتن

ج. سهولت ترجمه د. عدم وجود ابهام

۲۵- کدام گزینه صحیح است ؟ (نیمسال دوم ۸۹-۹۰)

الف. وظیفه تحلیل گر لغوی ، شناسایی نشانه ها و تعبیر عملگرها است .

ب. وظیفه تحلیل گر لغوی ، شناسایی نشانه ها و پردازش ماکروها است .

ج. هدف اعلان در زبان ها انتخاب نمایش حافظه ، کنترل نوع عملیات چند ریختی است .

د. وجود اعلان نوع برای متغیرها ، در هر زبان ضروری است .

۲۶-سر آغاز تئوری گرامر رسمی، که امروزه گرامر مستقل از متن (BNF) نام دارد، کدام زبان برنامه نویسی می باشد؟(نیمسال اول ۹۱-۹۰)

الف.ALGOL ب.FORTRAN ج.COBOL د.PL/I

۲۷-ابزارهای مورد استفاده در طراحی تحلیل گر لغوی و تحلیل گر نحوی در ساختار یک کامپایلر کدامند؟(نیمسال اول ۹۱-۹۰)

الف. ماشین خودکار متناهی، درخت های تجزیه ب. جدول نمادها، درخت های تجزیه

ج. جدول نمادها، گرامرهای رسمی د. ماشین خودکار متناهی، گرامرهای رسمی

۲۸- کدامیک از موارد زیر قابلیت خوانایی و قابلیت نوشتن را در زبان های C و پاسکال نشان می دهند؟(نیمسال اول ۹۱-۹۰)

الف. زبان C:قابلیت خوانایی کم، قابلیت نوشتاری کم

زبان پاسکال: قابلیت خوانایی زیاد، قابلیت نوشتاری کم

ب. زبان C: قابلیت خوانایی زیاد، قابلیت نوشتاری کم

زبان پاسکال: قابلیت خوانایی کم، قابلیت نوشتاری زیاد

ج. زبان C: قابلیت خوانایی کم، قابلیت نوشتاری زیاد

زبان پاسکال: قابلیت خوانایی زیاد، قابلیت نوشتاری کم

د. زبان C: قابلیت خوانایی کم، قابلیت نوشتاری زیاد

زبان پاسکال: قابلیت خوانایی زیاد، قابلیت نوشتاری زیاد

سوالات تشریحی فصل سوم

۱-معیارهای عمومی نحو یک زبان را نام ببرید ؟ (نیمسا ل دوم ۸۳)

۲- ساختار یک کامپایلر را با رسم شکل نمایش دهید ؟(نیمسال دوم ۸۳)

۳-۱۲- پاسخنامه سوالات تستی فصل سوم

سوال	الف	ب	ج	د
۱		❊		
۲				❊
۳			❊	
۴		❊		
۵			❊	
۶			❊	
۷				❊
۸			❊	
۹			❊	
۱۰		❊		
۱۱	❊			
۱۲		❊		
۱۳			❊	
۱۴			❊	
۱۵		❊		
۱۶			❊	
۱۷		❊		
۱۸				❊
۱۹		❊		
۲۰	❊			
۲۱		❊		
۲۲	❊			
۲۳	❊			
۲۴	❊			
۲۵			❊	
۲۶	❊			
۲۷				❊
۲۸			❊	

۳-۱۳- سوالات فصل چهارم

سوالات تستی فصل چهارم

۱-کدام گزینه غلط می‌باشد؟(نیمسال دوم۸۴)

الف. اگر رشته ای در زبان وجود دارد که دارای دو درخت تجزیه باشد گرامر مبهم می‌باشد.

ب. PERL توانایی پردازش عبارت‌های منظم را ندارد.

ج. هر تابع قابل محاسبه را می‌توان با ماشین تورینگ محاسبه نمود.

د. ماشین‌های تورینگ معادل گرامرهای نوع صفر هستند.

۲-کدامیک از مدل‌های زیر برای تعریف رسمی معنای زبان استفاده نمی شود؟(نیمسال اول ۸۵-۸۶)

الف.مدلهای گرامری ب.مدلهای اصل موضوعی

ج.مدل تابعی د.مدل شی گرا

۳- در کاهش عبارت لاندا ..... (نیمسال اول ۸۵-۸۶)

الف.کاهش خارجی ترین همانند فراخوانی با نام است و کاهش داخلی ترین جمله معادل فراخوانی با مقدار است.

ب.کاهش خارجی ترین همانند فراخوانی با مقدار است و کاهش داخلی ترین جمله معادل فراخوانی با نام است.

ج. هر دو نوع کاهش معادل فراخوانی با نام است.

د. هر دو نوع کاهش معادل فراخوانی با مقدار است.

۴- .....تابعی است که غیر پایانه سمت چپ را به مقادیر غیر پایانه‌های سمت راست بسط می دهد. (نیمسال اول ۸۵-۸۶)

الف.صفت موروثی ب.صفت ترکیبی

ج.گرامر صفت د.درخت انشقاق

۵- عملیاتی با امضای stack->integer برای یک پشته تعریف شده است این عمل معادل کدامیک از اعمال پشته است؟(نیمسال اول ۸۵-۸۶)

الف.Pop ب.Top ج.Size د.Push

۶- در ساختار ماشین پذیرنده کدام گرامر، از نواری محدود به همراه هدی که می تواند در دو جهت حرکت کند استفاده شده است. (نیمسال اول ۸۵-۸۶)

الف.ماشین تورینگ ب.ماشین خودکار خطی

ج.ماشین خودکار متناهی د.ماشین خودکار پشته ای

۷- کدام گزینه غلط می باشد؟(نیمسال دوم ۸۵-۸۶)

الف. هر تابع قابل محاسبه را نمی توان با ماشین تورینگ محاسبه کرد.

ب. ماشینهای تورینگ معادل گرامرهای نوع صفر هستند.

ج. بعضی از مسئله‌ها غیر قابل تصمیم گیری اند الگوریتم عمومی برای حل آنها وجود ندارد.

د. هیچکدام.

۸- در مورد حساب لاندا کدام گزینه غلط می باشد؟(نیمسال دوم ۸۵-۸۶)

الف.در دهه ۱۹۲۰ توسط کورچ مطرح گردید.

ب.اولین مدل معنای زبان برنامه سازی است.

ج.مدل خوبی را برای فراخوانی تابع زبان برنامه سازی ارائه کرد.

د. هیچکدام

۹- با توجه به تعریف ثوابت T(true) و F(false) در حساب لاندا تابع بولین and برابر کدام محاسبه λ زیر است؟(نیمسال دوم ۸۷-۸۸)

الف. $\lambda x.\ \lambda y.\ ((xT)y)$ ب.$\lambda\ x.\ \lambda y.((xy)F)$

ج. $\lambda\ x.\ \lambda y.((xT)y).$ د. $\lambda x.\ \lambda\ y.((xy)T)$

۱۰- در حساب لاندا تعریف زیر کدام عملگر را تعریف می کند؟(نیمسال اول ۸۸-۸۹)

$\lambda\ M.\lambda N.\lambda a.\lambda b.((Ma)((Na)b))$

الف. * ب./ ج.+ د.-

۱۱- کدامیک از موارد زیر است که ضرورتاً نمی تواند تضمین کننده صحت کامل برنامه در فرایند وارسی باشد؟(نیمسال اول ۸۸-۸۹)

الف.تعیین مشخصات (S) زبان ب.بررسی پیاده سازی مشخصات برنامه

ج.بررسی مشخصات (S) و برنامه د.آزمون و تست برنامه

۱۲- کدام یک از خصوصیات زبان perlنیست؟ (نیمسال دوم ۸۹-۹۰)

الف. نزدیکی ارتباط با سیستم عامل

ب. ترجمه مستقیم عبارات منظم

ج. وجود آرایه های انجمنی با قابلیت آدرس دهی محتویات که الزاما از طریق محتویات قابل دستیابی هستند .

د. انجام آسان عملیات تطابق و جایگزینی رشته ها

۱۳- کدام عبارت در مورد ماشین ها و گرامرها صحیح است؟(نیمسال اول ۹۱-۹۰)

الف. ماشین خوکار متناهی هر نماد را می خواند و در هریک از دو جهت که لازم باشد حرکت می کند.

ب. ماشین خوکار پشته ای همان ماشین خودکار خطیبه همراه پشته است.

ج. در ماشین تورینگ نوار از هر دو طرف نامحدود است.

د. در ماشین خودکار خطی امکان حرکت در یک جهت روی نوار وجود دارد.

۱۴- کدام عبارت در مورد خواص گرامرها صحیح است؟(نیمسال اول ۹۱-۹۰)

الف. در گرامرهای وابسته به متن نوع یک، طول همه رشته هایی که از نماد شروع ایجاد می شوند، کاهش پذیر نیست.

ب. گرامرهای نامحدود-نوع صفر فقط برای پیمایش رشته ها مناسب می باشند

ج. برای پیاده سازی گرامرهای مستقل از متن از ساختمان داده صف استفاده می شود.

د. در گرامر منظم امکان تولید رشته هایی به صورت a^n وجود ندارد.

سوالات تشریحی فصل چهارم

۱- روش‌های مختلف برای تعریف رسمی معنای زبان را بنویسید؟(نیمسال دوم ۸۵-۸۶)

۲- برای هر یک از توابع NOT,AND,OR یک تعریف با استفاده از حساب λ بنویسید؟(نیمسال اول ۸۸-۸۹)

۳-۱۴- پاسخنامه سوالات تستی فصل چهارم

سوال	الف	ب	ج	د
۱		❊		
۲				❊
۳	❊			
۴		❊		
۵				
۶		❊		
۷	❊			
۸				❊
۹		❊		
۱۰			❊	
۱۱				❊
۱۲			❊	
۱۳			❊	
۱۴	❊			

فصل پنجم:

انواع داده اولیه

آنچه در این فصل خواهید آموخت:

- شی داده
  - انقیاد شی داده
  - متغییرها و ثوابت
- نوع داده
  - مشخصات انواع داده اولیه
  - پیاده سازی انواع داده اولیه
- اعلان
  - اهداف اعلان
- کنترل نوع
  - کنترل نوع پویا
  - کنترل نوع ایستا
  - تبدیل نوع و تبدیل نوع ضمنی
- انتساب و مقدار دهی اولیه
- انواع داده اسکالر
  - نوع صحیح
    - زیر بازه
  - اعداد حقیقی ممیز شناور
  - اعداد حقیقی ممیز ثابت
  - نوع شمارشی
  - نوع بولین
  - نوع کاراکتری
- انواع داده مرکب
  - رشته‌ها
  - اشاره گرها
  - فایل‌ها
- سوالات تستی و تشریحی

یکی از اختلافات اساسی زبانهای برنامه سازی، انواع اطلاعاتی است که یک زبان برنامه سازی می تواند روی آنها عملیات انجام دهد. طبعاً، چگونگی و تنوع عملیات، مبنای قابلیت‌های یک زبان جهت پوشش اطلاعات می باشد. در این فصل چگونگی پیاده سازی اطلاعات از نوع اولیه مورد توجه قرار می گیرند.

۵-۱- شیء داده[۱] (D.O)

یک شیء داده، گروهی از یک یا چند قسمت از اطلاعات است که در کامپیوترهای مجازی استفاده می شود. در واقع شیء داده ظرفی[۲] برای دخیره مقادیری از داده‌هاست. یعنی محلی است که داده‌ها در آنجا ذخیره و بازیابی می شوند. یک شیء داده توسط مجموعه ای از صفات مشخص می شود که مهمترین آنها نوع داده است.

اشیاء داده از یک دیدگاه به دو دسته تقسیم می شوند:

- **تعریف شده توسط برنامه نویس:**

اشیاء داده‌هایی هستند که توسط برنامه نویس تعریف می شوند مانند متغیرها، مقادیر ثابت، آرایه، فایل،...

- **تعریف شده توسط سیستم:**

اشیاء داده‌هایی هستند که توسط سیستم به وجود می آیند و مستقیماً در اختیار برنامه نویس نیستند. مثل پشته‌های زمان اجرا، رکوردهای فعالیت زیر برنامه‌ها، بافرهای فایل، لیست فضای آزاد و جدول نمادها.

طول عمر[۳]:

یکی دیگر از صفات شیءداده طول عمر است. فاصله زمانی بین لحظه ای که حافظه به شی داده تخصیص داده می شود تا زمانی که حافظه از آن پس گرفته می شود را طول عمر شی داده گویند. هر یک از اشیاء داده موجود در برنامه، دارای طول عمر مخصوص به خود هستند. بعضی از اشیاء داده در شروع اجرای برنامه وجود دارند و برخی در حین اجرا بصورت پویا ایجاد می شوند و برخی نیز در حین اجرای برنامه از بین می روند و برخی تا آخر اجرای برنامه باقی می مانند.

ساختار شیء داده :

یک شیء داده توسط مجموعه ای از صفات مشخص می شود مثل نوع داده و نام که معمولاً در طول عمر آن عوض نمی شود. یک شیء داده دارای محلی برای مقدار داده است. مقدار یک شیء داده ممکن است عدد، کاراکتر، یا اشاره گر باشد.

- شیء داده ای توسط الگویی از بیت‌ها مشخص می شود.به مثال زیر دقت کنید:

الف) **متغیر مقید :** شیء داده به مقدار داده ۱۷ مقید می‌شود.

ب) **مقدار داده :** الگوی بیتی است که هروقت عدد ۱۷ در برنامه استفاده شود ، مترجم از آن استفاده می‌کند.

ج) **شیء داده :** محلی در کامپیوتر به نام A

شکل ۵ - ۱ یک شیء داده متغیر با مقدار ۱۷

[۱] Data Object
[۲] Container
[۳] Life time

از یک دیدگاه دیگر، اشیا داده از نظر ساختاری به دو دسته تقسیم می شوند:

- **شیء داده اولیه:** یک شیء داده اولیه است اگر تنها شامل یک محل حافظه برای ذخیره مقدار داده باشد. مانند اشیا داده از نوع char، float,int ،....
- **شیء داده ساختاری**: یک شیء داده ساختاری است اگر شیء داده شامل مجموعه ای از سایر اشیاء داده ای باشد. مانند رکورد، آرایه، لیست و ...

۵-۱-۱- انواع مختلف انقیاد یک شیء داده

یک شیء داده در طول عمر خود انقیادهای مختلفی را می پذیرد که مهمترین آنها عبارتند از:

- **انقیاد یک شیء داده به یک نوع:** این انقیاد در زمان ترجمه برنامه انجام می گیرد و مجموعه مقادیری که شیء داده می پذیرد به آن نسبت داده می شود.
- **انقیاد شیء داده به محلی از حافظه:** محلی در حافظه برای شیء در نظر گرفته می شود. این کار (انقیاد) از دید برنامه نویس مخفی بوده و توسط روال‌های مدیریت حافظه کامپیوتر مجازی انجام می شود.
- **انقیاد شیء داده به یک یا چند ارزش(مقدار)**: این نوع انقیاد توسط دستورات انتساب مثل A:=13 انجام می گیرد.
- **انقیاد یک شیء داده به یک یا چند نام:** این انقیاد توسط اعلان[۱]ها انجام پذیرفته و هنگام فراخوانی زیر برنامه و برگشت از آن اصلاح می شود.
- **انقیاد شی داده به یک یا چند شیء داده دیگر:** این انقیاد در هنگام استفاده از متغیرهای نوع اشاره گر، انجام می گیرد.

۵-۱-۲- متغیرها و ثوابت

اشیا داده با توجه به ویژگی مقدار، به دوسته تقسیم می شوند:

متغیر[۲]: یک شیء داده ای است که توسط برنامه نویس تعریف شده و به صورت صریح استفاده می شود.

```
Int n ;
Char ch;
```

ثابت[۳]: یک شیء داده ای با نام است و مقداری که به آن نسبت داده می شود در طول عمر آن ثابت است.

```
Const int max=30;
```

ثابت لیترال[۴]: یک ثابت لیترال، ثابتی است که نام آن همان نمایش مقدارش است. مثلا "۳۷"یک ثابت لیترال است که یک شیء داده با مقدار ۳۷ می باشد.

نکته: چون مقدار ثابت به طور دائم در طول عمرش به آن مقید می شود بنابراین، این انقیاد توسط مترجم شناخته شده است و کامپایلر می تواند از اطلاعات مربوط به مقادیر ثابت برای کاهش تولید کد استفاده کند.

---

[۱] Declation
[۲] Variable
[۳] Constant
[۴] Litteral

✓ گاهی اوقات کامپایلر می‌تواند ازاطلاعات مربوط به مقادیر ثابت به منظور اجتناب از تولید کد برای یک کلمه یا عبارت استفاده نماید.

مثال :

```
If( Max < 2 )then
…
Else
…
```

دراین حالت مترجم از قبل مقادیر داده‌ها را برای ثابت‌های Max و 2 دارد و می‌تواند محاسبه نماید که شرط فوق درست است یا خیر. به طور کلی ازهر کد نوشته شده داخل دستور If در یکی از قسمت‌های آن خودداری کند.

مثال۱:

```
F ( ) {
int N;
N=27;
…
}
```

- این اعلان یک شیء داده اولیه از نوع صحیح را مشخص می کند.(type)
- این شیء داده هنگام ورود به زیر برنامه ایجاد می شود و هنگام خروج از آن از بین می رود یعنی طول عمر آن برابر زمان اجرای زیر برنامه ا ست. (life time)
- در اثنای طول عمر این شیء داده به نام "N" مقید می شود که از این طریق به آن مراجعه می شود. اگر شیء داده به عنوان پارامتر به زیر برنامه ارسال شود نام دیگری به آن مقید می شود.(name)
- مقدار اولیه به شیء داده مقید نمی شود اما دستور انتساب مقدار ۲۷ را به آن مقید می کند.(value)

مثال۲:

```
Const int max=30;
Int N;
N:=27;
N=N+max
```

- N متغیری ساده است. max (توسط کاربر) 30,27 (لیترال) ثابت هستند.
- نام اشیاء: "30","27",max, N
- تفاوت بین مقدار"27" و 27 : 27 مقداری صحیح است که بصورت دنباله ای از بیت‌ها در حافظه نمایش داده می شود و نام "27"دنبا له ای از دو کاراکتر "2"و"7" است که بصورت دهدهی نمایش داده شده است.
- ثابت ۳۰ دو نام دارد: نام تعریف شده توسط برنامه نویس و نام لیترال که هر دو به یک محل از حافظه اشاره دارد.
- #define max 30 یک دستور است که موجب می شود مترجم max را برابر ۳۰ قرار دهد. در حالیکه صفت const در زبان C راهنمای مترجم است و می گوید متغیر max همیشه دارای مقدار ۳۰ است.

ماندگاری داده:

در اکثر موارد طول عمر متغیرها با زمان اجرای برنامه یکی است، اجرا که تمام شد متغیرها هم از بین می روند اما اگر طول عمر یک داده بیشتر از یک اجرا باشد لذا گوئیم آن داده ماندگار است و در بین اجراهای مختلف برنامه وجود دارد. به عبارت دیگر، ماندگاری یعنی از بین نرفتن انقیاد مکان و مقدار بعد از اتمام برنامه.

۵-۲- نوع داده[۱]

نوع داده، طبقه ای از اشیاء داده به همراه مجموعه ای از عملیات برای تولید و دستکاری می باشد. هر زبان مجموعه ای از انواع داده اولیه مانند char,int,bool,float دارد که در هنگام تعریف زبان مشخص می شوند. علاوه بر این، زبان ممکن است به برنامه نویس اجازه دهد انواع داده جدیدی را تعریف کند. در زبان‌های جدیدی مثل جاوا و Ada کاربران می توانند انواع داده-های جدیدی برای خود تعریف کنند ولی چنین کاری در فرترن و کوبول امکان پذیر نیست. نوع داده معمولاًدر دو سطح مختلف بررسی می شود:

- **مشخصات[۲]:** تعریف خصوصیات و مشخصات.
- **پیاده سازی کردن[۳]:** پیاده سازی آن نوع داده و عملیات روی آن.

۵-۲-۱- عناصر اساسی در سطح مشخصات :

- **صفات[۴]:** منظور از صفات، ویژگی است که اشیاء داده از یک نوع را با دیگر نوعها متمایز می کند. صفات اصلی هر شیء داده مانند نوع داده و نام، معمولاً در طول عمر آن عوض نمی شود. بعضی صفات ممکن است در توصیف گر[۵] به عنوان بخشی از شیء داده در حین اجرای برنامه، ذخیره شوند. توجه داشته باشید که مقدار یک صفت از شیء داده با مقداری که شیء داده حاوی آن است متفاوت باشد. چون مقدار موجود در شیء داده ممکن است در طول عمر آن تغییر کند و همیشه به طور صحیح در طول اجرای برنامه نمایش داده می شود ولی مقدار صفت شیء داده این گونه نیست.
- **مقادیر[۶]:** مجموعه ای از مقادیر ممکن که یک شیء داده می تواند داشته باشد که تا حد زیادی به سخت افزار وابستگی دارد. مجموعه مقادیر تعریف شده توسط نوع دادهاولیه را معمولاً مجموعه مرتب می گویند زیرا دارای کمترین و بیشترین مقدار است و برای هر دو مقدار یکی کوچکتر و دیگری بزرگتر است.
- **عملیات[۷]:** مجموعه ای از عملیات که برای یک نوع داده تعریف شده است ،تعیین می کند که اشیاء داده از آن نوع چگونه باید دستکاری شوند. این عملیات ممکن است عملیات اولیه یا عملیاتی باشند که توسط برنامه نویس تعریف می شوند. عملیات اولیه به عنوان بخشی از زبان تعریف می شوند و از دید برنامه نویس به دو دسته عملیات یکانی و دودویی تقسیم می شوند. عملیات تعریف شده توسط برنامه نویس به شکل زیر برنامه‌ها نوشته می شوند مثل زیر برنامه ای که قدر مطلق یک شیء داده از نوع صحیح را محاسبه می کند.

---

[۱] Data type
[۲] Specification
[۳] Implementation
[۴] Attribute
[۵] Descriptor
[۶] Value
[۷] Operation

برای مثال برای نوع داده آرایه در سطح مشخصات، عناصر اساسی عبارتند از:

- **صفات:** تعداد ابعاد، بازه، اندیس هر بعد، نوع داده هر عنصر
- **مقادیر:** مجموعه ای از مقادیر است که در عناصر ارایه ذخیره می شوند.
- **عملیات:** استفاده از اندیس برای مراجعه به یک عنصر، ایجاد آرایه، بدست اوردن حد پائین و بالا، انجام محاسبات روی آرایه.

۵-۲-۲- عناصر اساسی در سطح پیاده سازی :

نحوه پیاده سازی عملیات: سه روش برای پیاده سازی عملیات روی اشیا داده وجود دارد:

- **به صورت سخت افزاری:** به عنوان مثال اگر مقادیر صحیح به کمک نمایش سخت افزاری ذخیره شوند آنگاه می-توان جمع و تفریق را با استفاده از عملیات سخت افزاری پیاده سازی کرد.
- **به صورت زیر برنامه یا تابع:** مثلاً عمل جذر گیری که توسط سخت افزار به طور مستقیم پشتیبانی نمی‌شود. برای پیاده سازی عملیات یک زیر برنامه مثلاً SQRT نوشته می‌شود.
- **به صورت دستوراتی که داخل برنامه نوشته می‌شوند:** این روش نیز مانند روش قبلی نرم افزاری است اما به جای زیر برنامه، دستورات مربوطه در خود برنامه نوشته می‌شوند مثل عمل قدر مطلق گیری:

```
Abs(x)=if x<0then -xelse x
```

نمایش حافظه (چگونگی ذخیره اطلاعات): نمایش حافظه برای انواع داده اولیه، تحت تاثیر کامپیوتری است که برنامه را اجرا می‌کند به عنوان مثال نمایش عدد صحیح به صورت دنباله بیتی است جهت نمایش کاراکترها می‌توان از کدهای کاراکتری موجود در سخت افزار یا سیستم عامل بهره برد. اگر از نمایش های سخت افزاری استفاده شود آن گاه عملیات روی آن نوع می‌تواند با استفاده از عملیاتی پیاده سازی شود که توسط سخت افزار ارائه شده است و گرنه باید بطور نرم افزار شبیه سازی شود. صفت یک شی ممکن است در زمان اجرا در یک توصیف گر به عنوان بخشی از یک شی داده ذخیره شود. این کار در زبان هایی مانند لیسپ و پرولوگ برای قابلیت انعطاف وجود دارد (مثلاً اعداد int توسط سخت افزار پشتیبانی می‌شود- شبیه سازی اعداد ۲۴ رقمی در صورتی که سخت افزار حداکثر ۱۲ رقمی را پشتیبانی کند)

تعریف عملیات:

هر عملیات معمولاً به صورت یک تابع ریاضی بیان می شود بطوریکه یک یا چند پارامتر (عملوند) را به عنوان ورودی پذیرفته و نتایجی را تولید می کند. مجموعه ای از مقادیر که عملیات بر روی آنها تعریف شده است دامنه عملیات و مجموعه ای از نتایج ممکن برد عملیات نام دارد. الگوریتم موجود در بدنه عملیات مشخص می کند بر روی پارامترهای داده ای چه محاسباتی انجام شود تا نتایج مطلوب بدست آید یعنی الگوریتم، عملکرد عملیات را مشخص می کند.

امضای عملیات:

امضای عملیات، تعداد، نوع و ترتیب آرگومان های ورودی و خروجی را مشخص می کند. برای مشخص کردن امضای عملیات از نشانه گذاری‌های ریاضی به نام الگو[۱] (در زبانC) استفاده می کنیم.

$$op - nam : argtype \times argtype \times \ldots \times argtype \longrightarrow resulttype$$

$$\times : integer \times integer \longrightarrow integer$$

$$= : integer = integer \longrightarrow boolean$$

$$sqrt : real \longrightarrow real$$

[۱] prototype

گاهی اوقات تعیین مشخصات دقیق یک عملیات (تعیین مقادیر دامنه و برد) به صورت یک تابع ریاضی دشوار است، چهار عامل باعث پیچیده تر شدن پیاده سازی عملیات زبان‌های برنامه سازی به صورت تابع ریاضی می شود:

۱- عملیاتی که به ازای ورودی مشخصی (بعضی از مقادیر دامنه) تعریف شده نیستند[۱]:

عملی که بر روی دامنه خاصی تعریف شده (در زبان برنامه سازی) ممکن است برای بعضی از ورودی‌های روی آن دامنه، تعریف نشده باشد مانند مجموعه ای از اعداد که در عملیات محاسباتی، سرریز[۲] یا زیرریز[۳] تولید می‌کنند.

۲- آرگومان ضمنی[۴]:

ورودی‌های ضمنی یا ورودی‌هایی که به صورت صریح تعریف نشده اند مثل متغیرهای سراسری، باعث می‌شوند تعیین دقیق دامنه عملیات بر روی اشیا داده ممکن نباشد.

۳- اثرات جانبی[۵]:

یک عملیات ممکن است علاوه بر وظیفه اصلی خود، اعمال مخرب دیگری نیز انجام دهد. مثل عملیاتی که حاصل جمع دو عدد را بر می‌گرداند ولی مقادیر ذخیره شده در سایر اشیا داده را نیز اصلاح می‌کند یا یک تابع ممکن است علاوه بر مقدار برگشتی، آرگومان‌های ورودی خود را نیز تغییر دهد که این کار نیز نوعی اثر جانبی محسوب می شود.

۴- خود اصلاحی[۶]:

عملیات می‌تواند ساختار داخلی، از جمله داده‌های محلی که در بین اجراهای مختلف نگهداری می‌شوند یا حتی کد خود را اصلاح کند. بنابراین نتایج حاصل از عملیات برای مجموعه خاصی از آرگومانها، نه تنها به آن آرگومانها، بلکه به سابقه فراخوانی‌های قبلی در اثنای محاسبات و آرگومانهایی که در هر فراخوانی ارسال می‌شوند بستگی دارد. به عنوان مثال عملیات مولد عدد تصادفی (تابع Rand) یک آرگومان ثابت را می‌گیرد و در هر بار اجرا مقدار متفاوتی را بر می‌گرداند. این عمل علاوه بر اینکه در هر مرتبه نتیجه اش را بر می‌گرداند عدد دانه[۷] را هم تغییر می دهد. این کار باعث تغییر در نتایج بعدی تابع می‌شود. لذا نتایج حاصل از عملیات علاوه بر آرگومان ورودی، به سابقه فراخوانی‌های قبلی هم بستگی دارد. خود اصلاحی از طریق تغییر در کد متداول نیست ولی در زبان‌هایی مثل لیسپ این کار امکان پذیر است.

زیر نوع[۸]:

وقتی نوع داده جدیدی را توصیف می‌کنیم اغلب تمایل داریم بگوییم که این نوع مشابه نوع دیگری است به عنوان مثال در زبان C انواع int,short,long شکل‌های گوناگونی از نوع داده صحیح هستند و رفتار آنها یکسان است و علاقه داریم که عملیاتی مانند جمع و ضرب به طور یکسان تعریف شود. بنابراین‌اگر نوعی به عنوان بخشی از نوع بزرگتر باشد آن نوع را زیرنوع و به نوع بزرگتر ابرنوع[۹] می‌گوییم. یا به عبارت دقیق تر اگر مجموعه مقادیر یک نوع، زیرمجموعه، مجموعه مقادیر نوع دیگر باشد نوع با مجموعه مقادیر بزرگتر را ابر نوع و نوع دیگر را زیرنوع گویند. به عنوان مثال short int زیر نوع int و int زیرنوع long int می باشد.

---

[۱] Undefined for Certain Input
[۲] over flow
[۳] under flow
[۴] Implict argument
[۵] Side effect
[۶] Self modification
[۷] seed
[۸] Sub type
[۹] Super type

۵-۳- اعلان[۱]

اعلان دستوری از برنامه است که نام ، نوع و طول عمر اشیا داده را مشخص می‌سازد. اعلان‌ها بر دو نوع هستند :

- **صریح:** در این روش نوع و نام شی داده ای دقیقاً توسط برنامه نویس تعیین می‌گردد.
- **ضمنی:** خود برنامه مترجم (کامپایلر) یا کامپیوتر پیش فرض‌هایی را در مورد خصوصیات اشیا ارائه می‌کند. مثلاً در زبان فرترن، متغیرها از Nو .....وIوJ به طور پیش فرض از نوع صحیح هستند مثل INDEX یا NEW البته اعلان صریح نیز در زبان فرترن وجود دارد.

✓ در زبان Perl انتساب مقداری به متغیر آن را اعلان می‌کند.

متغیر رشته ای `$abc='test';`

متغیرصحیح `$abc=5;`

✓ گاهی اوقات جزئیات پیاده سازی نیز هنگام اعلان مشخص می‌شود. مثلاً در زبان کوبول اعلان یک متغیر صحیح به صورت Computational باعث می‌شود از نمایش حافظه دودویی به جای نمایش کاراکتری استفاده گردد.

✓ علاوه بر این، اعلان‌ها می‌توانند اطلاعاتی راجع به عملیات را به مترجم بدهند مثل اعلان تابع زیر که تعداد، ترتیب ، نوع پارامتر و نتیجه را مشخص می‌کند.

```
Function Sub(int x,float y):Real
```

✓ اطلاعاتی که توسط دستورات اعلان بدست می آید عبارتند از: ۱- نام شی داده ۲- نوع شی داده ۳- مقدار اولیه شی داده۴- صفات شی داده۵- طول عمر

۵-۳-۱- اهداف اعلان

الف- انتخاب نمایش حافظه[۲]: اگر اعلان اطلاعاتی راجع به نوع و صفات شی داده در اختیار کامپایلر قرار دهد بهترین نمایش حافظه برای شی داده انتخاب خواهد شد.

ب- مدیریت حافظه[۳]بهتر: اطلاعاتی که توسط اعلان کردن در رابطه با طول عمر اشیا داده ای فراهم می‌شود باعث مدیریت بهتر حافظه می‌شود. مثلاً متغیرهایی که در اولیک زیر برنامه اعلان می‌شوند طول عمر یکسانی دارند و می‌توان برای تمام آن‌ها یک بلوک حافظه را اختصاص داد و پس از اتمام زیر برنامه، آن بلوک حافظه را پس گرفت یا اگر متغیرهای پویایی وجود داشته باشد که توسط دستورات تخصیص حافظه مثل عملگر new در C++ و malloc درC ایجاد شوند چون طول عمر این اشیا داده متفاوت می باشد در بلوک دیگری از حافظه قرار می‌گیرند (درheap نگهداری می شوند)

ج-مشخص شدن وضعیت عملیات چندریختی[۴]: بسیاری از زبان‌ها، نمادهای خاصی مانند + را برای تعیین عملیات مختلف استفاده می‌کنند مثلاً نماد + می تواند مبین عملیات جمع دو عدد صحیح، جمع دو عدد اعشاری، الحاق دو رشته یا اجتماع دو مجموعه باشد که با توجه به نوع آرگومان‌ها عملیات مورد نظر تعیین خواهد شد. در Ada می‌توان زیربرنامه‌های

---

[۱] Declaration
[۲] Storage representation
[۳] Storage Managment
[۴] Generic operation

همنام تعریف کرد.ML این مفهوم را با چند ریختی کامل بسط دادکه در آن تابع برحسب ترتیب، تعداد، نوع آرگومان‌ها می‌تواند پیاده سازی‌های مختلف داشته باشد.
هدف اصلی اعلان‌ها در چند ریختی این است که اعلان‌ها موجب می‌شوند تا مترجم در زمان کامپایل کردن، عملیاتی را که به وسیله نماد مجدداً تعریف شده مشخص می‌شود را تعیین کند. مثلاً در زبان C، کامپایلر با اعلان دو متغیر A,B متوجه می‌شود که چه عملی در A+B انجام شود (جمع صحیح یا اعشاری) بنابراین درزمان اجرا لازم نیست کنترل شود که چه عملیاتی باید صورت گیرد. از طرفی در اسمالتاک چون اعلان نوع متغیر وجود ندارد در زمان اجرا باید تعیین شود + چه نوع عملی است.

تعریف کلی چندریختی:

تعریف چندین زیربرنامه با نام یکسان و پیاده سازی‌های متفاوت که برای یک عمل خاص نوشته می‌شود را چندریختی می-گویند و اینکه در زمان اجرا کدام یک از این توابع باید به کار گرفته شوند (کدام پیاده سازی) این کار توسط تعداد، ترتیب و نوع پارامترهای آن انجام می‌شود در عملیات چندریختی کل عملیات ثابت است فقط پیاده سازی آن برای ورودی‌های مختلف متفاوت خواهد بود.

```
A ( int x )

A ( int x , int y)

A ( int x , float y)
```

- ✓ در زبان ML این مفهوم را با استفاده از چندریختی کامل گسترش می‌دهد یعنی در این زبان یک نام تابع با پیاده سازی‌های متفاوت ارائه می‌شود که با توجه به انواع، تعداد و ترتیب ورودی‌ها و نتایج یکی از پیاده سازی‌ها انجام می-شود.

د- کنترل نوع[۱] : مهم ترین هدف اعلان از دیدگاه برنامه نویس، انجام کنترل نوع ایستا به جای کنترل نوع پویا می‌باشد.

۵-۴- کنترل نوع[۲]

نمایش حافظه ای که در سخت افزار برای داده‌ها ساخته می‌شود اطلاعاتی راجع به نوع اشیا داده ندارد مثلاً : ۱۱۰۱۰۱۰۱ این دنباله بیتی می‌تواند مبین اطلاعات از هر نوع صحیح، اعشاری و... باشد بنابراین کنترل نوع در سطح سخت افزار وجود ندارد و کامپیوترهای معمولی در سطح سخت افزار قادر به تشخیص خطای نوع نیستند. امتیاز اصلی استفاده از زبان سطح بالا این است که زبان می‌تواند کنترل نوع را برای تمام عملیات پیاده سازی کند و درمقابل خطاها محفوظ باشد ولی در زبان‌های سطح پایینی مانند اسمبلی، عملیات شامل خطاهای کنترل نوع انجام خواهد گرفت ولی نتیجه اش بی معنی خواهد بود چون کنترل نوع در زبان سطح پایین اسمبلی وجود ندارد.
منظور از کنترل نوع این است که هرعملیاتی که در برنامه انجام می‌گیرد تعداد و نوع آرگومان‌های آن درست باشد. دو روش برای کنترل نوع وجود دارد:

- **کنترل نوع پویا(D.T.C)[۳]:** عمل کنترل نوع در زمان اجرا صورت می گیرد.
- **کنترل نوع ایستا(S.T.C)[۱]:** عمل کنترل نوع در زمان ترجمه (کامپایل) صورت می گیرد.

---
[۱] Type checking
[۲] Type checking
[۳] Dynamic Type Checking(D.T.C)

۵-۴-۱- کنترل نوع پویا

کنترل نوع پویا در زمان اجرا انجام می‌شود و بلافاصله قبل از اجرای عمل خاصی صورت خواهد گرفت. در کنترل نوع پویا در هر شیء داده یک برچسب نوع قرار می‌گیرد که نوع آن شی ء داده را مشخص می‌کند. بنابراین کنترل نوع پویا با توجه به برچسب نوع (توصیفگر) در زمان اجرا انجام خواهد شد. در برخی از زبان‌های برنامه سازی مانند لسیپ و پرولوگ کنترل نوع به صورت پویا است در این زبان‌ها متغیرها اعلان نمی‌شوند و نوع متغیرها در حین اجرای برنامه می‌تواند تغییر کند بنابراین حتماً باید کنترل نوع، پویا باشد چون نوع متغیرها درحین اجرا تغییر می‌کند. در زبان‌های بدون اعلان، متغیرها را بدون نوع گویند چون نوع ثابتی ندارند. زبان هایی که از کنترل نوع پویا استفاده می کنند را زبان های بدون نوع[۲] نیز می گویند.

مزایای کنترل نوع پویا:

- به علت عدم نیاز به تعریف اعلان نوع داده ای، برنامه نویس از بسیاری از محدودیت‌ها آزاد است (افزایش انعطاف پذیری در برنامه نویسی).
- عمل تبدیل نوع در زمان اجرا انجام خواهد گرفت.
- برای تعریف هر نوع داده اعلان مورد نیاز نیست.

معایب کنترل نوع پویا :

- اشکال زدایی[۳] و یافتن تمام خطاهای نوع آرگومان بسیار مشکل است چون کنترل نوع پویا انواع داده را در زمان اجرای عملیات کنترل می‌کند لذا عملیاتی که اجرا نشوند کنترل نخواهند شد و تمام مسیرهای اجرایی ممکن را نمی-توان تست کرد.
- در کنترل نوع پویا اطلاعات مربوط به نوع، در زمان اجرای برنامه نگهداری خواهد شد لذا کنترل نوع پویا حافظه بیشتری مصرف خواهد کرد.
- کنترل نوع پویا می‌بایست به صورت نرم افزاری پیاده سازی شود و سخت افزار به ندرت از آن پشتیبانی می‌کند از آنجا که کنترل نوع قبل از اجرای هر عملی باید صورت پذیرد لذا سرعت اجرای برنامه در کنترل نوع پویا کاهش خواهد یافت.

به دلیل معایب فوق اغلب زبان‌ها سعی می‌کنند کنترل نوع پویا را کم کرده و بیشتر کنترل‌ها را به صورت ایستا در زمان ترجمه انجام دهند.

۵-۴-۲- کنترل نوع ایستا

کنترل نوع ممکن است در زمان ترجمه صورت گیرد مانند زبان‌های C,Pascal . بنابراین کنترل نوع ایستا با توجه به اطلاعات موجود در جدول نمادها که از اعلان های درون برنامه استخراج گردیده است در زمان ترجمه انجام خواهد شد.

مزایای کنترل نوع ایستا :

- کنترل نوع ایستا تمام عملیات موجود در برنامه را شامل می‌شود و تمام مسیرهای اجرائی کنترل می‌شوند لذا عمل چک کردن به بهترین صورت انجام می‌گیرد و اشکال زدایی برنامه نیز ساده تر می‌شود.
- عدم نیاز به حافظه اضافی جهت نگهداری اطلاعات نوع داده ای در زمان اجرا
- افزایش سرعت اجرای برنامه

---

۱ Static Type Checking(S.T.C)
۲ Type less
۳ Debuging

معایب کنترل نوع ایستا :

- کاهش انعطاف پذیری
- نیاز به تعریف اعلان برای تمام اشیاء داده

نکته: در کنترل نوع ایستا، کنترل در زمان کامپایل انجام می‌گیرد در گذر اول کامپایلر در جدول نمادها برای هر شی داده، نوع و برای هر عملیات تعداد، ترتیب و نوع آرگومان‌ها را مشخص می‌کند در گذر دوم عمل کنترل نوع صورت می‌پذیرد.
اطلاعات مورد نیاز در ارتباط با کنترل نوع ایستا معمولاً از اعلان‌های برنامه نویس یا ساختارهای زبان گرفته می‌شود. بعضی از این اطلاعات عبارتند از :
الف- برای هر عمل، تعداد، ترتیب، نوع آرگومان های ورودی/ خروجی مشخص است. ب- برای هر شی داده، نام، نوع، مقدار اولیه، حوزه و طول عمر شی داده، مشخص است.
در مراحل ترجمه، این اطلاعات در جدول نمادها درج خواهد شد.

امنیت نوع:

تابعی مانند $(f : S \rightarrow R)$ (که در آن S دامنه و Rبرد تابع f می باشد) از نظر بررسی نوع ایمن است (امنیت نوع دارد) اگر اجرای تابع f نتواند مقداری خارج از R را تولید کند. برای مثال اگر x,y از نوع صحیح Short باشند ممکن است عملگر * مقداری در خارج از دامنه اعداد صحیح Short تولید کند و لذا عملگر * بر روی اعداد صحیح Short ایمن نیست. بنابراین، اگر همه عملگرها (عملیات) یک زبان برنامه نویسی از نظر بررسی نوع ایمن باشند آن زبان از نظر بررسی نوع قوی است.

کنترل (بررسی) نوع قوی:

اگر در یک زبان امکان انجام کلیه بررسی نوع‌ها به صورت ایستا وجود داشته باشد آن زبان دارای قابلیت بررسی نوع قوی می باشد. مانند زبان ML.

استنتاج نوع[۱] :

استنتاج نوع یعنی اینکه پیاده ساز زبان، اطلاعات نوعی را که از قلم افتاده است را از سایر انواع تعریف شده بدست آورد (استنتاج‌کند). ویژگی استنتاج نوع در زبان ML وجود دارد.

مثال (برای زبان ML) **:**

```
Fun area (len:int,wid:int):int=len*wid;

Fun area (len,wid):int=len*wid;( از روی خروجی می‌فهمد که ورودی‌هااست )

Funarea( len:int , wid) = len * wid);

Funarea( len , wid:int ) = len * wid;

Funarea( len , wid ) = len * wid;
```

مثالی دیگر:

```
fun add(x:int, y:int):int = x+y;
    add is fully qualified. Any one declaration defines the operation.
```

[۱] Inference type

```
All are equivalent:
      fun add(x:int, y) = x+y;
      fun add(x, y:int) = x+y;
      fun add(x, y):int = x+y;
```

وقتی نوع یکی از متغیرهای x,y مشخص باشد می‌توان نوع دو مورد دیگر را با توجه به اینکه عمل + بین دو عدد صحیح یا دو عدد اعشاری صورت می‌گیرد تشخیص داد.
اما در مثال زیر، چون نوع آرگومان‌ها مبهم است و همه می‌توانند float یا همه می‌توانند int باشند استنتاج نوع امکان پذیر نیست.

```
But fun add(x, y) = x+y; is ambiguous
```

۵-۴-۳- تبدیل نوع[۱] و تبدیل ضمنی[۲]

اگر در زمان کنترل نوع، نوع واقعی آرگومان و نوع مورد انتظار یکسان نباشد یکی از دو عمل زیر صورت می‌گیرد:

- برنامه Error می‌گیرد و عملیات خاص مربوط به Error فراخوانی می‌شود (چاپ پیغام عدم تطابق نوع)
- عمل تبدیل نوع انجام می‌گیرد تا نوع آرگومان تغییر کند.

اغلب زبانها تبدیل نوع را به دو صورت انجام می‌دهند:

صریح : عمل تبدیل نوع به صورت مجموعه ای از توابع توکار[۳] توسط برنامه نویس فراخوانی می‌شود (مانند int to char).

ضمنی : عمل تبدیل نوع به صورت خودکارتوسط مترجم زبان انجام می‌شود. به عنوان مثال در زبان C هنگام جمع دو شی داده از نوع int, float ، نوع int قبل از عمل جمع به طور ضمنی تبدیل به float می‌شود. دو گونه تبدیل نوع ضمنی داریم:

- اگر در تبدیل نوع ضمنی ، اطلاعاتی از بین نرود به آن تبدیل ضمنی، تبدیل گسترش یا ارتقا یافته گفته می‌شود.
- اگر در تبدیل نوع ضمنی ، اطلاعاتی از بین برود به آن تبدیل ضمنی، تبدیل باریک کننده یا محدودکننده[۴] گفته می‌شود.

نکته: درعمل تبدیل نوع ممکن است لازم باشد در نمایش حافظه ای زمان اجرای یک شیء تغییرات گسترده ای صورت گیرد. به عنوان مثال در کوبول و PL/I اعداد به صورت رشته کاراکتری ذخیره می‌شوند. برای جمع کردن این اعداد در اغلب ماشین‌ها، نمایش حافظه ای رشته کاراکتری باید به نمایش دودویی که توسط سخت افزار پشتیبانی می‌شود تبدیل گردد. در هنگام برگرداندن نتیجه دوباره باید از دودویی به کاراکتری تبدیل شود در نتیجه، خود عملیات تبدیل نوع بیش از عملیات جمع تکرار خواهد شد.

نکته: دو فلسفه متضاد در مورد وجود تبدیل نوع ضمنی در زبان وجود دارد. در پاسکال و Ada هیچ گونه تبدیل نوع ضمنی وجود ندارد. در PL/I,C تبدیل نوع ضمنی صورت می‌گیرد. تبدیل نوع ضمنی، آزادی برنامه نویس را زیادتر می‌کند مثلاً کامپایلر PL/I خطاهای جزئی مثل اسامی نادرست متغیرها را نادیده می‌گیرد.

---

[۱] Conversion
[۲] Coersion
[۳] Built in Function
[۴] narrowing

معمولا ۲ فلسفه متضاد در هنگام عدم تطابق نوع وجود دارد (Type Mismatch) :

۱. در پاسکال و Ada تقریبا هیچ گونه تبدیل نوعی انجام نمی‌شود همیشه در صورت عدم تطبیق نوع، خطا گزارش می شود.

۲. در زبان C تبدیل انواع‌ها قانونی هستند مگر اینکه امکان هیچ گونه تبدیلی وجود نداشته باشد که باز هم خطا گزارش خواهد شد.

- ✓ زبان PL/I به یافتن حداقل خطاهای تبدیل نوع و کلا خطاهای برنامه نویسی مشهور است یعنی سعی می‌کند که در اکثر موارد تبدیل نوع را انجام دهد و خطا نگیرد.

- ✓ درزبان PL/I عبارت 9+10/3 غیرمجاز است زیرا با توجه به بالاتر بودن اولویت تقسیم به جمع ابتدا تقسیم شده و حاصل 3.33… با حداکثر ارقام اعشاری مجاز خود به دست می‌آید و در هنگام جمع این عدد با 9 چون یک رقم اضافی می‌آورد، پس به خطای Overflow منجر می‌شود.

۵-۵- انتساب و مقداردهی اولیه

انتساب، مهم ترین عملیات برای تغییر انقیاد یک مقدار به یک شی داده است این تغییر، اثرجانبی[۱] عملیات محسوب می‌شود. در بعضی زبان‌ها مانند C,APL و لیسپ، انتساب مقداری را برمی‌گرداند که این مقدار یک شی داده ای است که حاوی یک کپی از مقدار نسبت داده شده است ولی در زبان پاسکال عمل انتساب، مقداری را بر نمی‌گرداند.

```
Assignment(:=):integer1*integer2→void

Assignment(=):integer1*integer2→integer3

Assignment : Type1 * Type2 →void          درپاسکال

Assignment : Type * Type2 → Type3         Lisp , APL , C
```

۱- یک کپی از مقدار موجود در integer2 در integer1 قرار می‌گیرد و نتیجه ای را برنمی‌گرداند (تغییر integer1 یک اثر ضمنی عملیات است)

۲- یک کپی از مقدار موجود در integer2 در integer1 قرار می‌گیرد و شی داده integer3 ایجاد می‌شود که شامل مقدار integer2 است.

موضوع فوق باعث می‌شود دستوری مانند A:=B:=C در زبان پاسکال خطا باشد ولی دستور A=B=C در زبان C درست بوده و مقدار C را به B و سپس به A نسبت می‌دهد.

به مقدار موجود در یک شی داده، مقدار راست[۲] آن شی داده و به آدرس یک شی داده، مقدار چپ[۳] آن شی داده گفته می شود. بر اساس این تعریف، فرایند عملیات انتساب A=B در چهار مرحله به صورت زیر تعریف می شود:

۱- مقدار چپ اولین عملوند را حساب کن

۲- مقدار راست دومین عبارت عملوند را حساب کن

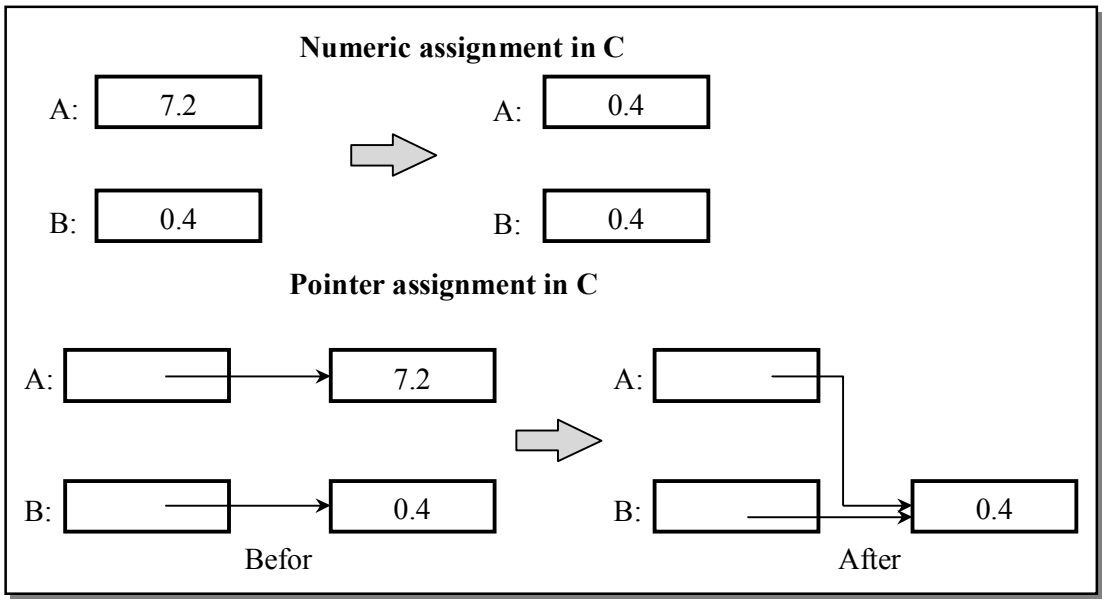
۳- مقدار راست محاسبه شده را به شی داده مقدار چپ نسبت بده.

۴- مقدار راست محاسبه شده را به عنوان خروجی برگردان (برای زبان هایی که عمل انتساب خروجی برمی گرداند)

---

[۱] Side effect
[۲] Right-Value
[۳] Left-Value

مثال) انتساب A=B را درنظر بگیرید که B و A در آن دو اشاره گر هستند اگر B یک اشاره گر باشد آن گاه مقدار راست B حاوی مقدار چپ شی داده دیگری است. بنابراین A=B یعنی مقدار راست A به شی داده ای اشاره کند که مقدار راست B به آن اشاره می‌کند. (مقدار راست B را به مقدار چپ A نسبت بده که مقدار راست B ، مقدار چپ شی داده دیگری است)



شکل ۵ -۲ دو دیدگاه از انتساب

مقداردهی اولیه[1]:

شی داده فاقد مقدار اولیه، شی داده ای است که ایجاد شده است ولی هنوز مقداری به آن داده نشده است. ایجاد یک شی داده به معنای اختصاص یک بلوک حافظه است. به عبارت دیگر، اگر در همان لحظه اول تعریف شی داده به شی داده مقدار بدهیم گوییم شی داده دارای مقدار اولیه است. در بعضی زبان‌ها مثل APL، هر شی داده ایکه ایجاد می‌شود باید برای آن مقدار اولیه تعیین کرد. در بعضی دیگر از زبان‌ها مثل پاسکال مقداردهی اولیه توسط دستور انتساب صورت می‌گیرد. متغیرهای فاقد مقدار اولیه، عامل مهمی برای بروز خطا در برنامه نویسی است. از نظر قابلیت اعتماد برنامه بهتر است بلافاصله پس از ایجاد متغیرها، مقدار اولیه ای به آن‌ها نسبت دهیم. مثلاً در زبان Ada به همراه اعلان می‌توان مقداردهی اولیه انجام داد.

```
A:array(1..3)offloat:=(17.2,20.4,30.6);
```

دو نوع مقدار دهی اولیه در زبانها وجود دارد:

صریح: که در این حالت برنامه نویس باید دستورات لازم برای دادن مقدار اولیه به متغییرها را در برنامه وارد کند.

ضمنی: که در این حالت خود کامپایلر مقدار اولیه متغیرها را تعیین می کند که این مقدار اولیه می تواند صفر یا Null باشد.

تساوی و هم ارزی :

اگرچه دستور انتساب در اغلب زبان‌ها وجود دارد ولی مشکلاتی در این زمینه وجود دارد که باید برطرف شود. انتساب زیر در زبان Zork را در نظر بگیرید:

A→2+3

برای زبان هایی که انواع ایستا برای داده‌ها دارند نوع A مشخص می کند از کدام معنا استفاده شود:

[1] Initialization

- اگر A از نوع صحیح باشد آنگاه مقدار ۵ به A نسبت داده می‌شود
- اگر A از نوع عملیات باشد آنگاه عمل "۳+۲" نسبت داده می‌شود.

برای زبان‌هایی که انواع در آن‌ها به صورت پویا است و نوع A با انتساب مقدار به آن مشخص می‌گردد هردو معنا قابل استفاده است و انتساب فوق باعث ابهام می‌شود این وضعیت دقیقاً در پرولوگ اتفاق می‌افتد.

عملگر is : یعنی مقدار هم ارز نسبت داده شود. 1) x is 2+3 , x=5;

عملگر= : به معنای انتساب الگو است. 2) x :=2+3 , x :=5;

مورد ۱ درست است چون در x is 2+3، متغیرx مقدار ۵ می‌گیرد و بعد با x=5 مقایسه می‌شود حاصل درست است.

مورد ۲ نادرست است چون در x = 2+3 ، معنای = به معنای انتساب الگوی ۳+۲ به متغیرx است لذا وقتی با x=5 مقایسه می‌شود حاصل نادرست است.

۵-۶- انواع داده اسکالر[۱]

انواع داده اسکالر مانند انواع داده char,int,float فقط یک صفت دارند و از معماری سخت افزار کامپیوتر پیروی می‌کنند مثلاً شی نوع صحیح فقط دارای یک صفت مقدار صحیح (مثلاً ۱۷و ۱۸ و ۴۲) است. انواع داده اسکالر شامل انواع صحیح- اعشاری- بولین –کاراکتر می باشد. انواع داده مرکب شامل چندین صفت هستند به عنوان مثال رشته‌ها شامل دنباله ای از کاراکترهاست ولی ممکن است صفت دیگری مانند طول رشته را داشته باشد داده‌های مرکب ساختار پیچیده تری دارند که معمولاً توسط کامپایلر پیاده سازی می شوند و نه با سخت افزار. انواع داده مرکب شامل آرایه‌ها- رشته‌ها- فایل‌ها- اشاره‌گرها می باشند.

۵-۶-۱- نوع داده صحیح

مشخصات:

یک شی داده از نوع صحیح معمولاً صفتی غیر از نوع ندارد و تنها شامل یک مقدار است. مجموعه مقادیر ممکن برای انواع صحیح یک مجموعه ترتیبی متناهی از اعداد صحیح است. در زبان C چهار کلاس از نوع صحیح وجود دارد : int, long, short, char

عملیات:

عملیات روی نوع داده صحیح شامل موارد زیر است:

۱- عملیات محاسباتی

$$Binary-Op : integer * integer \rightarrow integer \triangleright MOD, +, -, *, DIV, /$$
$$Unary-Op : integer \rightarrow integer \triangleright ABS, ++, --, +, -$$

۲- عملیات رابطه ای

$$Rel-Op : integer * integer \rightarrow boolean \triangleright =, <>, >=, <=, <, >$$

۳- عملیات انتساب

$$assign : integer * integer \rightarrow void$$
$$assign : integer * integer \rightarrow integer$$

۴- عملیات بیتی

$$Bit-Op : integer * integer \rightarrow integer \triangleright \&, |, \sim, \hat{}, \square, \square$$

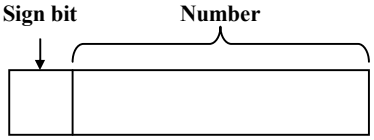
[۱] Scalar data type

پیاده سازی:

نوع داده صحیح توسط نمایش حافظه سخت افزار و مجموعه‌ای از عملیات محاسباتی و رابطه ای بر روی مقادیر صحیح پیاده سازی می‌شوند. ۳ نوع نمایش حافظه ای برای نوع داده صحیح وجود دارد:

الف- بدون توصیفگر:

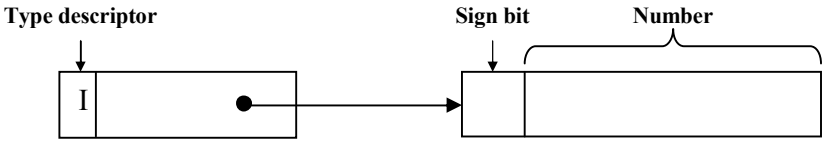
این نمایش حافظه، توصیفگر زمان اجرا ندارد و فقط مقدار در آن ذخیره می‌شود. این نمایش حافظه در زبان‌هایی استفاده می‌شود که زبان اعلان‌ها و کنترل نوع ایستا را برای مقادیر صحیح فراهم می‌کند مانند C و فرترن



شکل ۵ -۳

ب- توصیفگر و مقدار در دو کلمه مجزا:

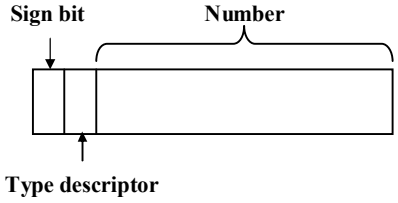
این نمایش حافظه، توصیفگر زمان اجرا دارد و توصیفگر در محل دیگری از حافظه ذخیره شده است که اشاره‌گری به آن اشاره می‌کند. این نمایش حافظه در لیسپ استفاده می‌شود. عیب آن این است که حافظه لازم برای شی داده صحیح دو برابر می‌شود و مزیت آن این است که عملیات روی آن به صورت سخت افزاری قابل پیاده سازی است. استفاده از نمایش سخت افزاری باعث افزایش سرعت عملیات می شود.



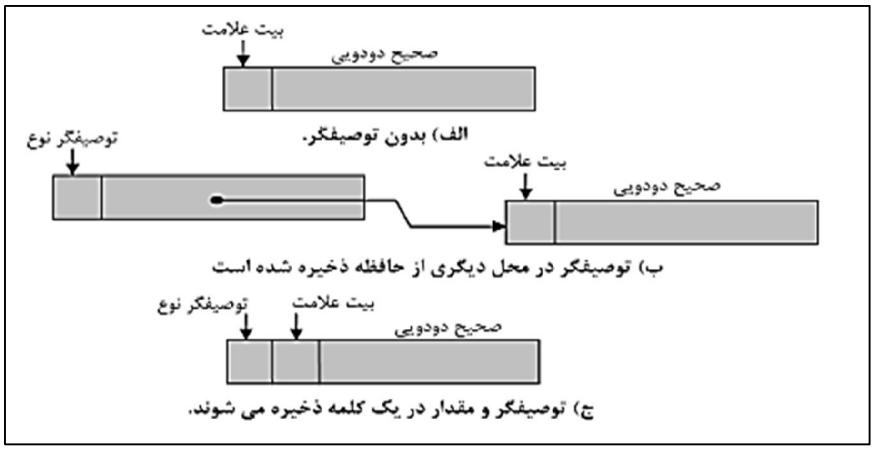
شکل ۵ -۴

ج- توصیفگرو مقدار در یک کلمه:

توصیفگر نوع و مقدار در یک کلمه ذخیره می‌شود لذا در حافظه صرفه جویی می‌شود ولی برای استفاده عملیات سخت افزار باید مقدار را از توصیفگر توسط دستورات شیفت از یکدیگر جدا کرد لذا سرعت عمل کمتر است.(از روش ب بیشتر استفاده می‌شود.)



شکل ۵ -۵



شکل ۵ –۶ سه نمایش حافظه برای مقادیر صحیح

زیر بازه ها[1]:

مشخصات:

زیر بازه‌ها شامل دنباله‌ای از مقادیر صحیح و بازه محدود هستند زیر بازه ها، زیر نوع، نوع داده صحیح هستند. مانند نمونه‌های زیر در پاسکال و Ada:

پاسکال `A:1..50`

Ada `A:integer rang 1..50`

پیاده سازی:

انواع زیربازه دو مزیت مهم در پیاده سازی دارند:

- **نیاز به حافظه کمتر:** چون بازه کمتری از مقادیر را شامل می شود، مقادیر زیر بازه نسبت به مقادیر صحیح معمولی، بیت‌های کمتری نیاز دارند.
- **کنترل نوع بهتر:** اعلان یک متغییر از نوع زیربازه باعث می شود کنترل نوع دقیق تری صورت گیرد. به عنوان مثال اگر متغییر Month به این صورت باشد: Month:1..12 ، آنگاه دستور زیر غلط است:

```
Month:=0
```

این خطا در زمان کامپایل تشخیص داده می شود. در بسیاری از موارد کنترل نوع زیر بازه ممکن نیست. به عنوان مثال، انتساب زیر را در نظر بگیرید:

```
Month:=Month+1
```

[1] Subrange

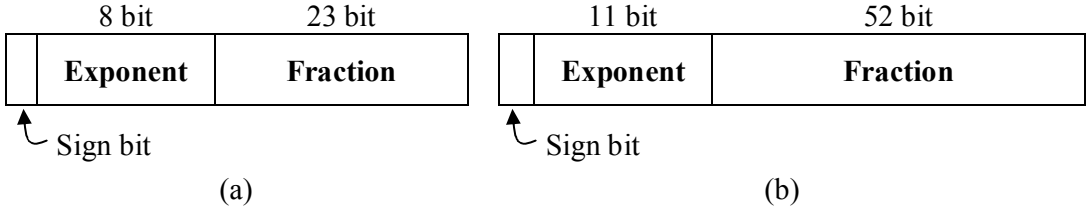
۵-۶-۲- اعداد حقیقی ممیز شناور[۱]

مشخصات:

این نوع داده معمولاً با صفت real در فرترن و پاسکال یا float در C مشخص می‌شود. دقت مورد نیاز برای اعداد ممیز شناور، که تعداد ارقام در نمایش دهدهی است توسط برنامه نویس مشخص می‌گردد مثل زبان Ada عملیات محاسباتی، رابطه ای، انتساب مشابه اعداد صحیح، برای اعداد حقیقی هم امکان پذیر است ولی به علت مسائل مربوط به گرد کردن، کمتر دو عدد حقیقی با هم مساوی می‌شوند. بنابراین در این حالت حلقه‌هایی که برای تست کردن از اعداد حقیقی استفاده می‌کنند ممکن است در حلقه دائم بیافتد. لذا گاهی اوقات تساوی بین دو عدد حقیقی توسط طراح زبان جلوگیری می‌شود.

پیاده سازی:

در اکثر زبان‌ها نحوه پیاده سازی اعداد حقیقی ممیز شناور به سخت‌افزار بستگی دارد برای ذخیره و پیاده سازی اعداد ممیز شناور از استاندارد IEEE754 استفاده می‌شود برای این کار از فرمتی شبیه نماد علمی استفاده می‌کنیم. که استاندارد IEEE754 استاندارد ۳۲ و ۶۴ بیتی را برای اعداد ممیز شناور مشخص می‌کند فرمت ۳۲ بیتی و ۶۴ بیتی به صورت زیر است:



شکل ۵ -۷

استاندارد ۳۲ بیتی برای اعداد حقیقی ممیز شناور:

در استاندارد ۳۲ بیتی هر عدد حقیقی ممیز شناور شامل سه فیلد است:

بیت S : فیلد علامت یک بیتی که صفر به معنای مثبت بودن است.

بیتE: توان ظاهری ۸ بیتی با افزودنی ۱۲۷، یعنی در هنگام ذخیره سازی توان در حافظه ۳۲ بیتی، مقدار ۱۲۷ به آن افزوده شده و سپس ذخیره می‌گردد در بازه ۰ تا ۲۵۵ که معادل توان دو از ۱۲۸- تا ۱۲۷ است.

بیت M: مانتیس ۲۳ بیتی است معمولا اعداد ممیز شناور را به صورت نرمال شده ذخیره می‌کنند در مبنای ۲، عدد نرمال شده باید با ارزش‌ترین بیت قسمت اعشار ۱ باشد برای مثال ۰/۰۰۱۱۱ نرمال نیست ولی ۰/۱۱۰۱۱ نرمال است. اولین بیت مانتیس در عدد نرمال شده همیشه ۱ است.

علامت عدد را مشخص می‌کند و با توجه به مقادیر E, M مقدار دقت به صورت زیر است.

[۱] Floating point

پارامتر	مقدار
$E = 255$ and $M \neq 0$	عدد نامعتبر
$E = 255$ and $M = 0$	∞
$0 < E < 255$	2^{E-127} (1.M)
$E = 0$ and $M \neq 0$	2^{-126}.M
$E = 0$ and $M = 0$	0

چند نمونه:

$+1 = 2^0 * 1 = 2^{127.127} * (1).0$ (binary) = 0 01111111 000000…

$+1.5 = 2^0 * 1.5 = 2^{127.127} * (1).1$ (binary) = 0 01111111 100000…

$-5 = -2^2 * 1.25 = 2^{129.127} * (1).01$ (binary) = 1 10000001 010000…

۵-۶-۳- اعداد حقیقی ممیز ثابت[۱]

مشخصات:

اغلب سخت افزارها شامل اشیای داده صحیح و ممیز شناور هستند. برای برخی از داده‌های حقیقی اگر از ممیز شناور استفاده کنیم خطای گردکردن اتفاق خواهد افتاد. مثلا برای تعریف یک شی داده برای قد اشخاص (متر و سانتی متر) و یا پول (دلار و سنت) با توجه به اینکه تعداد ارقام بعد از اعشار ثابت است از نوع ممیز ثابت استفاده می کنیم.

پیاده سازی:

ممکن است مستقیماً توسط سخت افزار پشتیبانی شود یا به صورت نرم افزاری شبیه سازی گردد. اینگونه اعداد به صورت (این X برابر ۱۰/۴۲۱ باشد، مقدار راست X صحیح ذخیره می شوند و نقطه اعشار به عنوان صفت آن شی داده است. اگر مقدار حاوی صفتی به نام فاکتور مقیاس برابر ۳ Xمقدار راست با مقدار راست موجود در عمل انتساب فرق دارد) برابر ۱۰۴۲۱ و شی می باشد و معنایش این است که بعد از نقطه اعشار سه رقم قرار دارد. یعنی با توجه به فرمول زیر:

$$Value(X) = rvalue(X) \times 10^{-SF}$$

SF صرف نظر از مقدار راست X، همواره برابر ۳ است.

در نهایت به طور خلاصه خواهیم داشت:

- **اعداد اعشاری**

اعداد صحیح می‌توانند دامنه محدودی را قبول کنند. اعداد اعشاری با توجه به این که از دوقسمت پایه و توان تشکیل شده اند، دامنه متغیری را پوشش می‌دهند ولی درعین حال دقت خیلی بالایی نخواهند داشت (زیرا ممکن است شرط تساوی دو عدد اعشاری برقرار نشود.) این نوع سخت افزار پشتیبانی می‌شود.

عملیات روی آن مشابه عملیات روی اعداد integer می‌باشد فقط برخی از عملیات ممکن است توسط نرم افزار شبیه سازی شود مانند عملیات به توان رساندن و ...

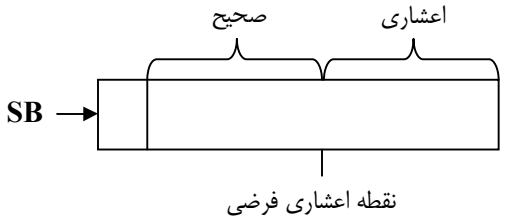
[۱] Fixed point

ازجهت پیاده سازی روش‌های زیر را دارد :

۱. FixedPoint (ممیز یا تقطه ثابت)

مثلا در زبان Cobol اعلان داده اعشاری ممیز ثابت با عبارت Picture نشان داده می‌شود.

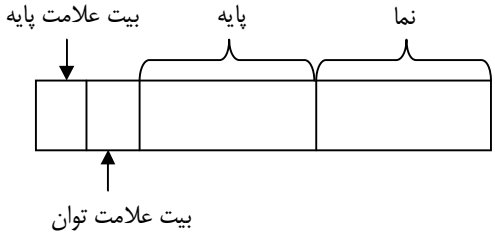
Picture 999 V 99



شکل ۵ -۸

۲. Floating Point (ممیز شناور)

مانند نماد علمی



شکل ۵ -۹

✓ در زبان Adaمی‌توان تعداد ارقام دقت عدد اعشاری را توسط برنامه نویس مشخص نمود.

نمونه ای از اعداد اعشاری ممیز ثابت در زبان PL1 به صورت ازیر است:

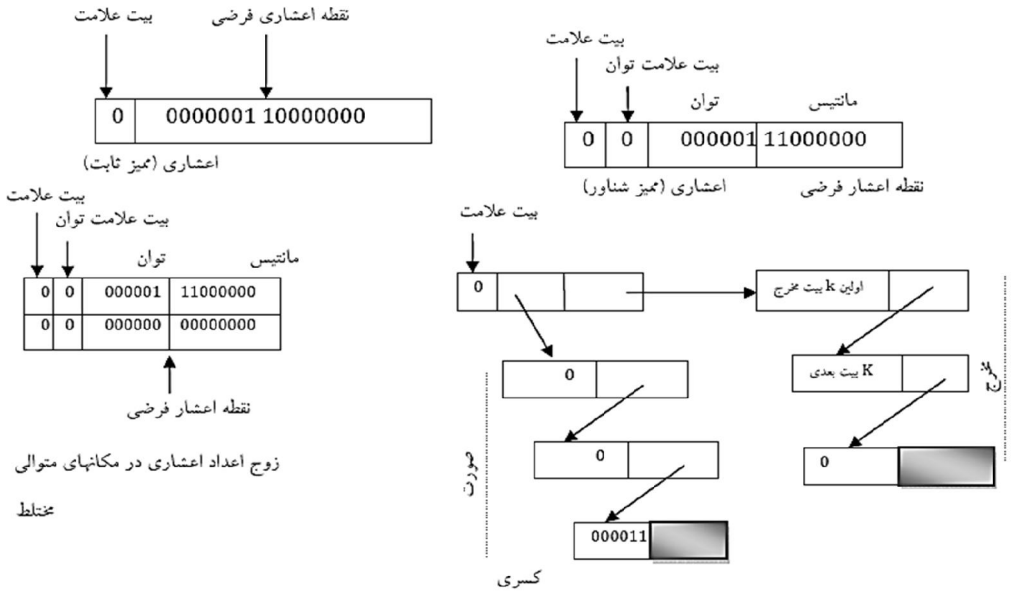
DECLARE x Fixed DECIMAL (1 , 3) ;

تعداد ارقام صحیح تعداد ارقام اعشار

سایر انواع داده عددی:

اعداد موهومی: عدد موهومی متشکل از یک جفت از اعداد است که یکی از آنها بخش حقیقی و دیگری بخش موهومی را نشان می دهد.

اعداد گویا: عدد گویا خارج قسمت دو عدد صحیح است.



شکل ۵ -۱۰ نمایش مقدار ۱/۵ بدون توصیفگر

۵-۶-۴- نوع شمارشی

مشخصات:

مقادیر نوع شمارشی بر اساس تعریف برنامه نویس مشخص می شوند که لیست مرتبی از مقادیر مجزاست. مقادیر نوع شمارشی به نام ثوابت شمارشی نیز خوانده می شوند. برنامه نویس اسامی لیترال‌هایی را که باید برای مقادیر، مورد استفاده قرار گیرند و همچنین ترتیب آنها را با استفاده از اعلانی مشخص می کند. نمونه ای از نوع شمارشی و تعریف متغییرهایی از آن نوع در C# به صورت زیر است:

مثال:**enum StudentClass {Fresh,Shop,Jonior,Senior}**

این تعریف لیترال‌های Fresh,Soph,Jonior,Senior را نیز تعریف می کند که در هر جایی از برنامه قابل به کارگیری و استفاده می باشند.

عملیات اصلی روی نوع داده شمارشی عبارتند از: عملیات رابطه ای (=, <, >)، انتساب و عملیات Successor(بعدی) و Predecessor (قبلی) که به ترتیب عناصر قبلی و بعدی را مشخص می کنند.

پیاده سازی:

نمایش حافظه برای شی داده ای از نوع شمارشی بسیار ساده است. هر مقدار در دنباله شمارشی در زمان اجرا به وسیله مقادیر صحیح ۲،۱،۰... قابل نمایش است. چون فقط مجموعه کوچکی از مقادیر در نوع شمارشی وجود داردو مقادیر منفی نیستند، نمایش آنها از مقادیر صحیح نیز ساده تر است. به عنوان مثال، نوع Class که در بالا تعریف شد، فقط چهار مقدار ممکن دارد که در زمان اجرا به صورتSenior=3,Jonior=2,Soph=1,Fresh=0 نمایش داده می شود. در زبان C می توان این ترتیب را تغییر داد.

۵-۶-۵- نوع بولی

مشخصات:

متشکل از اشیای داده ای است که یکی از دو مقدار TRUE یا FALSE را می پذیرد. متداول ترین عملیات روی نوع داده بولین عبارتند از: and,or,xor,nor,not and

پیاده سازی:

نمایش حافظه برای شی داده بولی یک بیت از حافظه است، به شرطی که نیاز به توصیفگر برای نوع داده‌ها، نباشد. چون یک بیت در حافظه قابل نمایش نیست معمولاً از یک واحد قابل آدرس دهی مانند بایت یا کلمه استفاده می شود. مقادیر True یا False به دو روش در این واحد حافظه نمایش داده می شوند.

- بیت خاصی برای این مقادیر استفاده می شود، به طوریکه False=0 و True=1 و بقیه بیت‌های مربوط به بایت یا کلمه بدون استفاده می ماند.
- مقدار صفر در کل واحد حافظه (بایت یا کلمه) نشان دهنده False و مقدار غیرصفر نشان دهنده True است.

بعضی از زبانها مثل C فاقد نوع بولی اند. در این زبان،از نوع داده صحیح برای این منظور استفاده می شود. مقدار صفر نشان دهنده False و مقدار غیر صفر نشان دهنده True است. دستورات زیر را ببینید:

```
int flag;
flag = 7;
```

چون flag برابر با صفر نیست، به عنوان True استفاده می شود. اگر انتساب;flag = 0 را داشته باشیم، flag دارای ارزش False خواهد بود. این روش استفاده از مقادیر صحیح به جای مقادیر بولی، اشکالاتی دارد. به عنوان مثال، اگر به جای andمنطقی (&&) از and بیتی (&) یا به جای or منطقی (| |) از or بیتی (|)، یا به جای نقیص منطقی (!) از نقیص بیتی (~) استفاده کنید، با مشکل مواجه خواهید شد.دستورات زیر را ببینید

```
int found;
found = 12;
```

نمایش بیتی عدد ۱۲ در یک کلمه ۱۶ بیتی به صورت زیر است که ارزش درستی دارد:

Found :	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**

اگر آن را نقیص بیتی کنید (~found) نمایش بیتی آن به صورت زیر است که باز هم ارزش درستی دارد:

~Found :	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**

بنابراین نتیجه می گیریم که found و نقیض بیتی ؟ آن هر دو ارزش درستی دارند. در صورتی که بخواهید واقعاً نقیض found ارزش نادرستی داشته باشد، باید آن را نقیض منطقی کنید (!found) در این صورت نمایش بیتیfoundبه صورت زیر خواهد بود که ارزش نادرستی دارد:

!Found :	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

۵-۶-۶- کاراکترها

مشخصات:

نوع داده کاراکتری اشیای داده را به وجود می آورد که مقدار آنها یک کاراکتر است. نوع داده کاراکتری در حین ورودی/خروجی کاربرد دارد بدین معنا که اطلاعات از طریق صفحه کلید به صورت کاراکتری دریافت می شود و از طریق صفحه نمایش به صورت کاراکتری روی خروجی نمایش داده می شود و اطلاعات در پردازنده به صورت صفر و یک (باینری) است. با توجه به اینکه هریک از کاراکترها کد اسکی دارند بنابراین می توانند در عملیات رابطه ای نیز شرکت کنند.

پیاده سازی:

مقادیر داده‌های کاراکتری همیشه توسط سیستم عامل و سخت افزار پشتیبانی می شوند. اگر نمایش کاراکتری که توسط زبان تعریف شد، با نمایش کاراکتری که توسط سخت افزار پشتیبانی می شود، یکسان باشد آنگاه عملیات رابطه ای نیز مستقیماً در سخت افزار نمایش داده میشوند یا توسط کدهای نرم افزاری شبیه سازی خواهد شد.

۵-۷- انواع داده مرکب

۵-۷-۱- رشته‌ها

مشخصات و نحو:

رشته ها، آرایه فشرده شده ای از کاراکترها می باشند. رشته ها، یا به صورت نوع اولیه در زبان برنامه نویسی هستند(ML,Prolog) یا به صورت آرایه ای از کاراکترها(C,Pascal,Ada) می باشند.

```
String  str;
Char  str[10];
```

در صورتیکه رشته‌ها به صورت آرایه ای از کاراکترها باشند، عملیات اندیس گذاری بر روی آنها امکان پذیر است، ولی اگر به صورت نوع داده اولیه باشند، این کار امکان پذیر نیست.

با رشته‌های کاراکتری از نظر طول، سه گونه برخورد می شود:

- **طول ثابت (ایستا):** طول رشته در هنگام ایجاد رشته مشخص می گردد. اشیای تغییر ناپذیر مربوط به کلاس String در Java ، C + + و C# از این دسته اند. دقت کنید که منظور از شیء تغییر ناپذیر این است که وقتی ایجاد شد، قابل تغییر نیست و اگر بیش تر از طول ثابت، کاراکتر در رشته ذخیره شود از انتها حذف می شود و اگر کمتر از طول ثابت، کاراکتر در رشته باشد با کاراکتر خالی پر می شود تا به طول ثابت برسد.
- **طول متغیر با حد معین (طول پویای محدود):** شیء داده رشته کاراکتری ممکن است طول حداکثری داشته باشد که در برنامه اعلان شود. در این حالت، اگر کمتر از طول ثابت، کاراکتر در رشته باشد با کاراکتر خالی پر نمی شود ولی از طرفی بیشتر از حداکثر طول نیز نمی توان کاراکتر در رشته ذخیره کرد.
- **طول متغیر (طول پویا):** طول رشته‌ها می تواند در زمان اجرا تغییر کنند و حداکثر طول برای آن مشخص نمی شود. JavaScript و Perl از این نوع رشته‌ها استفاده می کنند. این نوع رشته‌ها دارای سر بار تخصیص و آزاد سازی حافظه اند، ولی قابلیت انعطاف آن‌ها زیاد است.

رشته‌های کاراکتری در زبان C کمی پیچیده تر می باشند. در انتهای رشته باید کاراکتر تهی('0/') قرار گیرد و برنامه نویس باید اطمینان حاصل کند که رشته‌ها به تهی ختم می شوند.

عملیات گوناگونی بر روی رشته‌ها امکان پذیر است که بعضی از آنها عبارتند از:

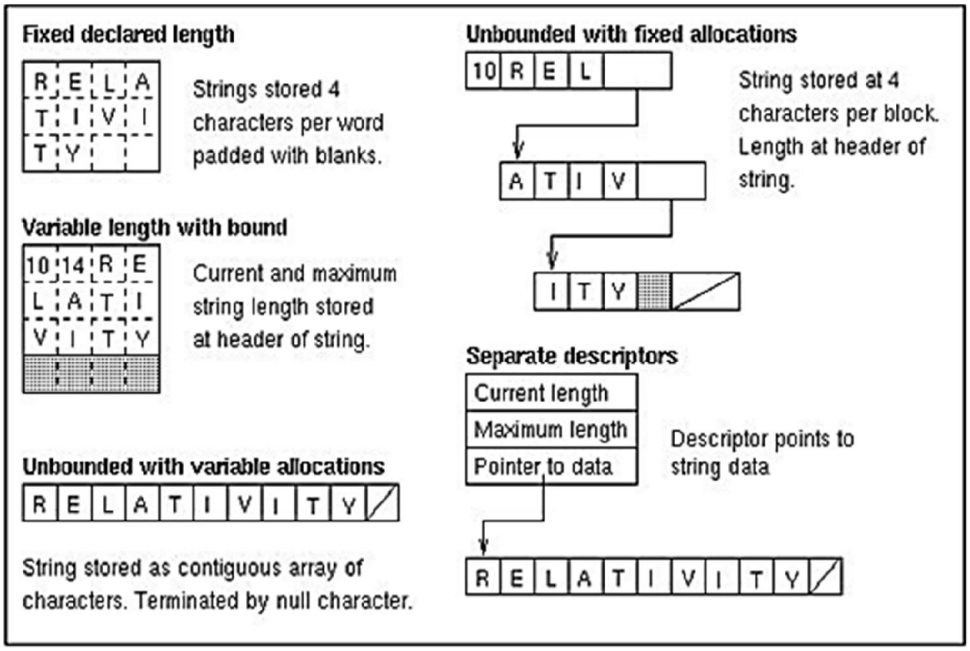
- الحاق رشته‌ها (Concatation) مانند strcat در C
- عملیات رابطه ای در رشته‌ها مانند strcmp در C
- انتخاب زیر رشته با استفاده از اندیس
- فرمت بندی ورودی–خروجی
- انتخاب زیر رشته با تطابق الگو
- رشته‌های پویا
- انتخاب زیر رشته در رشته اصلی مانند strstr در C

پیاده سازی :

برای رشته ای با طول ثابت نمایش حافظه همان شکلی است که برای بردار فشرده ای از کاراکترها استفاده شد. برای رشته طول متغییر با حد معین نمایش حافظه از توصیفگری استفاده می کند که حاوی حداکثر طول و طول فعلی ذخیره شده در شی داده است. برای رشته‌های نامحدود می توان از نمایش حافظه پیوندی اشیا داده طول ثابت استفاده کرد.

- طول ثابت (بالا، سمت چپ، اولین سطر) : هر ۴ کاراکتر در یک کلمه ذخیره می شود و بقیه طول رشته با فضای خالی پر می شود.
- طول متغییر با حد معین (بالا، سمت چپ، دومین سطر) :حداکثر طول و طول رشته در ابتدا ذخیره می شود
- طول نامحدود با تخصیص ثابت (بالا،سمت راست،اولین سطر) : ۴ کاراکتر در هر بلوک ذخیره می شود و طول در ابتدای رشته قرار می گیرد.
- طول نامحدود با تخصیص‌های متغییر (پایین، سمت راست) : رشته به صورت آرایه ی پیوسته کاراکترها، ذخیره می شود و انتهای هر رشته با null یا تهی مشخص می شود.

توضیحات مربوط به موارد فوق را در شکل زیر مشاهده می کنید.



شکل ۵ –۱۱ نمایش حافظه برای رشته ها

۵-۷-۲- اشاره گرها و اشیاء داده برنامه نویس

معمولاً در هر زبان برای اتصال اشیاء داده به یکدیگر از اشاره گر استفاده می شود. زبان برنامه نویسی باید ویژگی‌های زیر را در مورد اشاره گر داشته باشد :۱- نوع داده اولیه اشاره گر ۲ – عمل ایجاد[۱]۳- عمل انتخاب[۲]

- **نوع داده اولیه اشاره گر:** شی داده اشاره گر شامل آدرس شی داده دیگری است یعنی شامل مقدار چپ یک شی داده دیگر است.
- **عمل ایجاد کردن:** برای اشیاء داده با طول ثابت مانند آرایه، رکورد، انواع داده اولیه. عمل ایجاد کردن بلوکی از حافظه را برای شی داده جدید ایجاد می کند و مقدار چپ آن را برمی گرداند.
- **عملیات دستیابی:** این عمل باعث می شود تا محتویات جایی که اشاره گر به آن اشاره می کند دستیابی شود

مشخصات :

نوع داده اشاره گر دسته ای از اشیاء داده را تعریف می کند که مقادیر آن‌ها آدرس‌های اشیاء دیگر است و به دو روش با آن‌ها برخورد می شود :

الف : اشاره گرها ممکن است فقط به یک نوع شی داده مراجعه کنند: این روش در پاسکال ,Ada، C، استفاده می شود. که در آن‌ها اعلان نوع و کنترل نوع ایستا ممکن است.

(مثال **`int *p;`**

* نوع p را اشاره گر معرفی می کند. نوع int مشخص می کند،که مقدار p می تواند مقدار چپ یک شی داده از نوع int باشد.

ب : اشاره گر ممکن است به هر نوع شی داده مراجعه کند: این روش در زبان‌هایی مانند اسمالتاک استفاده می شود، که اشیاء داده در حین اجرا، دارای توصیفگر هستند و کنترل نوع پویا انجام می شود.

(مثال **`void *p`**

* نوع p را اشاره گر معرفی می کند. نوع void مشخص می کند،که مقدار p می تواند مقدار چپ یک شی داده از هر نوعی باشد.

عملیات ایجاد کردن: حافظه را برای شی داده طول ثابت تخصیص می دهد و اشاره گری به این شی داده جدید ایجاد می کند که در یک شی داده اشاره گر ذخیره می شود. عملیات تخصیص حافظه (ایجاد) در زبان های C++,Ada ,Pascal توسط دستور new و در زبان C توسط دستور malloc انجام می گیرد.

```
P = malloc (size of(int))
```

یک بلوک حافظه دو کلمه ای را ایجاد کن تا به عنوان شی داده ای از نوع int مورد استفاده قرار گیرد و مقدار چپ آن را در p ذخیره کن.

عملیات انتخاب کردن :

اجازه می دهد تا مقدار اشاره گر دنبال شود تا به شی داده مورد نظر برسیم. در زبان C عمل انتخاب با * مشخص می شود.

به عنصر first از رکوردی دستیابی دارد که p به آن اشاره می کند **`*p.first`**

---

[۱] creation
[۲] selection

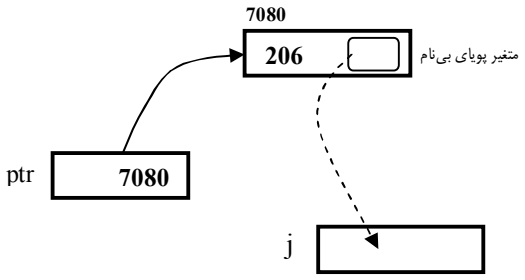
عملیات مهمی که در مورد اشاره گرها می توان انجام داد عبارتند از:

۱-عملیات انتساب

```
Int *p, *q;
C: p=(int *)malloc(sizeof(int))
C++:q=new int;
```

۲-عملیات دستیابی به محتویات

```
int j;
int *ptr
*ptr=206;
j=*ptr;
```



۳-عملیات محاسباتی و رابطه ای

```
Int *p, *q;
…
p++;
q--;
if(p==q)
```

زبان C	زبان C++	زبان پاسکال

`Int *p;`
…
`p=(int *)malloc(sizeof(int));`
…
`*p=27;`
…
`Free(p);`
	`Int *p;`
	…
	`p=new int;`
	…
	`*p=27;`
	…
	`Delete p;`
		`Varp:^integer;`
		…
		`New(p);`
		…
		`P^:=27;`
		…
		`Dispose(p);`

جدول ۵ -۱

پیاده سازی :

شی داده اشاره گر به صورت محلی از حافظه نمایش داده می شود که شامل آدرس محل دیگری از حافظه است. این آدرس، آدرس پایه بلوک حافظه، نشان دهنده شی داده است که اشاره گر به آن محل اشاره می کند دو نمایش حافظه برای مقادیر اشاره گر استفاده می شود :

الف : آدرس دهی مطلق[۱] **:** مقدار اشاره گر ممکن است آدرس واقعی بلوک حافظه مربوط به شی داده باشد.

ب : آدرس دهی نسبی[۲] **:** مقدار اشاره گر ممکن است آفستی از آدرس پایه بلوک حافظه هرم[۱] باشد که شی داده در آن ایجاد شده است.

[۱] Absolute address
[۲] Relative address

مزایای و معایب آدرس دهی مطلق :

انتخاب و دسترسی به شی داده از طریق آدرس‌های مطلق کارآمدتر است چون از عملیات سخت افزاری برای دستیابی به شی داده استفاده می کند. عیب آدرس دهی مطلق این است که مدیریت حافظه مشکل تر می شود. زیرا شی داده در حافظه جابجا نمی شود. عیب دیگر این است که بازیابی حافظه از اشیاء داده ای که به صورت داده‌های زباله در آمده دشوار است زیرا هر یک از این اشیاء داده به صورت منفرد بازیابی می شوند.

مزایا و معایب آدرس دهی نسبی :

استفاده از آدرس نسبی به عنوان اشاره گر، مستلزم تخصیص بلوکی از حافظه است که new تخصیص دیگری در آن انجام دهد مزیت آدرس دهی نسبی این است که می تواند بلوک حافظه را در هر زمان به نقاط دلخواهی از حافظه حرکت داد مزیت دیگر این است که با کل ناحیه ای که به هنگام ورود به زیر برنامه ایجاد می شود می توان به صورت یک شی داده برخورد کرد. در داخل این ناحیه نیازی به بازیابی حافظه برای تک تک اشیاء داده نیست چون کل ناحیه به هنگام خروج از زیر برنامه بازیابی می شوند.

مقایسه آدرس دهی نسبی و مطلق به طور خلاصه :

- در مورد آدرس دهی مطلق کارایی بالا وجود دارد. سرعت اجرای برنامه بالا می باشد چون آدرس فیزیکی در حافظه است ولی در روش نسبی کارایی برنامه پایین است چون برای دسترسی به شی داده محاسبه باید توسط offset و Base صورت گیرد.
- در روش آدرس دهی مطلق شی داده را نمی توان انتقال داد در حالیکه در روش نسبی می توان شی داده را به راحتی انتقال داد.
- عملیات ترمیم[۲] (حل مشکل زباله و ارجاع سرگردان) در مورد آدرس دهی مطلق به سختی صورت می گیرد ولی در روش نسبی راحت تر صورت می گیرد.

۵-۷-۳- فایل‌ها و ورودی-خروجی

فایل، ساختمان داده ای با دو ویژگی مهم می باشد :

- بر روی حا فظه ثانویه مانند دیسک یا نوار تشکیل می شود و ممکن است بسیار بزرگتر از ساختمان داده‌ها باشد.
- طول عمر آن می تواند بسیار بزرگ باشد.

انواع متداول فایل‌ها عبارتند از:

- فایل‌های ترتیبی[۳]
- فایل‌های دستیابی مستقیم[۴]
- فایل‌های ترتیبی شاخص دار[۵]
- فایل‌های متنی[۶]

---

[۱] heap
[۲] recovery
[۳] sequential
[۴] Direct access
[۵] Indexed sequential files
[۶] text

متداول ترین فایل‌ها، فایل‌های ترتیبی اند. اما، اغلب زبانها از فایل‌های دستیابی مستقیم و فایل‌های ترتیبی شاخص دار استفاده می کنند. دو کاربرد مهم فایل‌ها عبارتند از: ورودی/ خروجی داده‌ها در محیط عملیات خارجی و حافظه موقت در مواقعی که حافظه کافی وجود ندارد. عناصر فایل را رکورد گویند ولی در اینجا از این اصطلاح صرف نظر می کنیم تا با ساختمان داده رکورد (که در فصل بعد اشاره خواهد شد) اشتباه نشود.

فایل‌های ترتیبی:

فایل می تواند در حالت خواندن یا در حالت نوشتن دستیابی شود. در هر دو حالت یک اشاره گر موقعیت فایل وجود دارد که موقعیتی را قبل از اولین عنصر فایل، بین دو عنصر فایل، یا بعد از آخرین عنصر فایل تعیین می کند. در حالت نوشتن، اشاره گر موقعیت فایل، همیشه به بعد از آخرین عنصر فایل اشاره می کند و می توان عنصری را در آن محل نوشت و فایل را به اندازه یک عنصر بسط داد. در حالت خواندن، اشاره گر موقعیت فایل، می تواند در هر نقطه ای از فایل باشد و می توان عمل خواندن را در آن محل انجام داد. در این حالت نمی توان عنصر جدیدی ر ا اضافه کرد. در هر دو حالت پس از عمل خواندن یا نوشتن، اشاره گر موقعیت فایل حرکت می کند تا به موقعیت عنصر بعدی اشاره کند. اگر این اشاره گر بعد از آخرین عنصر فایل قرار گیرد می گوییم به انتهای فایل رسیدیم.

عملیات اصلی بر روی فایل‌های ترتیبی عبارتند از:

- بازکردن
- خواندن
- نوشتن
- تست انتهای فایل
- بستن

فایل‌های دستیابی مستقیم:

در فایل ترتیبی، عناصر به ترتیبی که در فایل قرار دارند بازیابی می شوند. اگر چه عملیات محدودی برای جابجایی اشاره گر موقعیت فایل وجود دارد ولی در این فایل‌ها، دستیابی تصادفی به عناصر، غیر ممکن است. فایل دستیابی مستقیم، طوری دستیابی می شود که می توان به هر عنصر به طور تصادفی دست یافت (مثل آرایه و رکورد). اندیسی که برای انتخاب یک عنصر به کار می رود، کلید نام دارد که ممکن است یک مقدار صحیح یا شناسه دیگری باشد. اگر یک مقدار صحیح باشد اندیس معمولی است که برای تعیین عنصری از فایل به کار می رود، اما چون فایل دستیابی مستقیم، در حافظه ثانویه ذخیره می شود، پیاده سازی فایل و عملیات انتخاب، متفاوت از آرایه است.

فایل ترتیبی شاخص دار:

فایل ترتیبی شاخص دار، شبیه فایل دستیابی مستقیم است به طوریکه امکان دستیابی ترتیبی، با شروع از عنصری که به طور تصادفی انتخاب شد، وجود دارد. به عنوان مثال، اگر عنصری با کلید ۲۷ انتخاب (خوانده) شود، عملیات خواندن بعدی، ممکن است عنصر بعدی را به ترتیب انتخاب کند (به جای دادن مقدار کلید). این سازمان فایل، توازنی را بین سازمان‌های ترتیبی محض با دستیابی مستقیم محض به وجود می آورد.

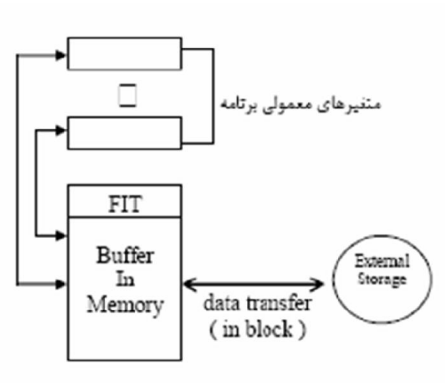
پیاده سازی فایل:

ازجهت پیاده سازی مسئول انجام همه عملیات مربوط به فایل‌ها و مدیریت آنها به عهده سیستم عامل می‌باشد. یک زبان برنامه سازی باید مکانیزم‌هایی جهت برقراری ارتباط با برنامه نویس و سیستم عامل داشته باشد.

از دید جزئی تر وقتی یک فایل باز می‌شود، فضایی به نام FIT[۱] و همچنین فضایی به عنوان بافر برای هر فایل تخصیص می-یابد.

FIT : دراین جدول نام فایل، تاریخ آخرین به روز رسانی و اندازه فایل و مشخصات دیگر از جمله محل‌های خواندن و نوشتن روی دستگاه ذخیره و بازیابی اطلاعات نگه داشته می‌شود.

Buffer: این حافظه به عنوان یک صف برای خواندن و نوشتن عمل می‌کند، درصورت پر یا خالی شدن بافر، اطلاعات به دستگاه خارجی انتقال داده می‌شود و یا از آن به داخل بافر منتقل می‌شود.



شکل ۵ -۱۲

[۱] File Information Table

۵-۸- سوالات فصل پنجم

سوالات تستی

۱-کدامیک از موارد زیر جزء اهداف اعلان نیست؟(نیمسال اول ۸۵-۸۶)

الف.عملیات وراثت ب.عملیات چند ریختی

ج.مدیریت حافظه د.انتخاب نمایش حافظه

۲-کدامیک از عملیات زیر جزء عملیات اصلی مربوط به داده‌های شمارشی نیست؟(نیمسال اول ۸۵-۸۶)

الف.عملیات رابطه ای ب.عملیات انتساب

ج.عملیات جبری مثل جمع و تفریق د.عملیات پیداکردن عنصر بعدی و قبلی

۳-کدام گزینه غلط می باشد؟(نیمسال دوم ۸۵-۸۶)

الف.هر عملیات روی داده، یک دامنه و یک بازه دارد.

ب.هر عملیات روی داده، یک تابع ریاضی نمی باشد.

ج.اعلان دستوری از برنامه است که نام و نوع اشیای داده را که در حین اجرای برنامه مورد نیاز هستند را مشخص می نماید.

د.کلیه موارد بالا

۴-انقیادهای یک شی داده کدام است؟(نیمسال دوم ۸۵-۸۶)

الف.نوع ، محل ب.محل، مقدار ، نوع ، نام، اجزا

ج.مقدار ، محل ، نوع د.نام ، اجزا ، مقدار

۵-کدامیک از زبان‌های زیر از چند ریختی حمایت می کند؟(نیمسال دوم ۸۵-۸۶)

الف.C ب.اسمالتاک ج.ML د.JAVA

۶- روش پیاده سازی رشته‌های کاراکتری کدام یک از موارد زیر است؟(نیمسال اول ۸۶-۸۷)

الف.رشته ای با طول ثابت ب.رشته با طول متغیر و حد بالا مشخص

ج.رشته با طول نامحدود د.همه موارد صحیح می باشد.

۷- کدامیک از موارد زیر جزء اهداف اعلان است؟(نیمسال اول ۸۶-۸۷)

الف.انتخاب نمایش حافظه ب.عملیات چندریختی

ج.کنترل نوع د.همه موارد

۸-چه نمایش حافظه ای برای مقادیرصحیح، در زبانهای Late binding مناسب تر است؟(نیمسال اول ۸۶-۸۷)

الف.بدون توصیف کننده زمان اجرا ب.با توصیف کننده در یک کلمه جداگانه

ج.با توصیف کننده در همان کلمه د.هیچکدام

۹–در مورد کنترل نوع پویا کدام گزینه غلط است؟(نیمسال اول ۸۶–۸۷)

الف.در هر عملیات کنترل نوع صورت می گیرد و در صورتی اجرا می شود که انواع آرگومان درست باشد.

ب.لازم نیست هر عملیات به نتایج خود یک نوع را نسبت دهد تا عملیات بعدی بتواند آنها را کنترل کند.

ج.کنترل نوع پویا در زمان اجرا انجام می شود.

د. در کنترل نوع پویا در هر شی داده یک برچسب قرار می گیرد که نوع آن را مشخص می کند.

۱۰–اهداف اعلان کدامیک از موارد زیر است؟(نیمسال دوم ۸۶–۸۷)

الف.انتخاب نمایش حافظه ب.عملیات چند ریختی

ج.کنترل نوع د.همه موارد

۱۱–کدام گزینه صحیح است؟(نیمسال دوم ۸۶–۸۷)

الف.کنترل نوع پویا حافظه بیشتری نسبت به کنترل نوع ایستا مصرف می کند.

ب.اگر تمام خطاهای نوع را به طور ایستا رفع کنیم زبان را نوع قوی گویند.

ج.در کنترل نوع پویا برای کاهش برخی هزینه‌ها ممکن است عملیات کنترل نشوند.

د.هر سه گزینه صحیح است.

۱۲–کدامیک از مفاهیم زیر کمتر با هم سازگارند؟(نیمسال دوم ۸۶–۸۷)

الف.تعریف نوع متغیرها و ترجمه ب.Static Scope Rrule وترجمه

ج.Late binding و تفسیر د.Early binding,Dynamic Scope Rule

۱۳–کدام یک از گزینه‌های زیر دلایل استفاده از زیر بازه به جای نوع داده ای صحیح می باشد؟(نیمسال دوم ۸۶–۸۷)

الف.عملیات محاسباتی ساده تر ب.کنترل نوع بهتر

ج.استفاده از عملیات successorوpredecessor د.پیاده سازی مستقیم با سخت افزار

۱۴–اگر عملیاتی ساختار داخلی اعم از داده‌های محلی که در بین اجراهای مختلف نگهداری می شوند یا کد خود را اصلاح کند این خاصیت را چه گویند؟(نیمسال اول ۸۷–۸۸)

الف.خودرانی ب.دانه اصلاحی ج.خود اصلاحی د.کد اصلاحی

۱۵–مهمترین هدف اعلانها از دیدگاه برنامه نویس کدام است؟(نیمسال اول ۸۷–۸۸)

الف.کنترل نوع ایستا به جای کنترل نوع پویا ب.کنترل نوع پویا به جای کنترل نوع ایستا

ج.مدیریت حافظه ایستا به جای مدیریت حافظه پویا د.مدیریت حافظه پویا به جای مدیریت حافظه ایستا

۱۶–با توجه به قطعه کد زیر، برای عمل جمع در x+y و انتساب نتیجه محاسبه شده x+y در x به ترتیب از راست به چپ کدام تبدیل ضمنی صورت می گیرد؟(نیمسال اول ۸۷–۸۸)

```
int x;
float y;
.
.
x=x+y
```

الف.گسترش و گسترش ب.باریک کننده و باریک کننده
ج.گسترش و باریک کننده د.باریک کننده و گسترش

۱۷– در قطعه برنامه زیر چه تعداد ثابت وجود دارد؟(نیمسال دوم ۸۷–۸۸)

```
Cons t int Max =30;
Int  N;
N=27;
N= N +Max;
```

الف. ۱ ب.۲ ج.۳ د. ۴

۱۸– به چه زبانی نوع قوی می گویند؟(نیمسال دوم ۸۷–۸۸)
الف. تمام خطاهای نوع به طور پویا بر طرف شود.
ب. تمام خطاهای نوع به طور ایستا برطرف شود.
ج. استنتاج نوع وجود داشته باشد.
د. اعلان نوع جدید وجود داشته باشد.

۱۹–کدام یک از اعلان‌های زیر در زبان ML بر اساس استنتاج نوع رفع ابهام نمی گردد و نتیجه معتبر نخواهد بود؟(نیمسال دوم ۸۷–۸۸)
الف. Fun area (length, width): int=length *width
ب.Fun area (length: int, width) =length* width
ج.Fun area (length, width: int) =length*width
د. Fun area (length, width) =length*width

۲۰–در کدام دو زبان زیر هیچ گونه تبدیل ضمنی وجود ندارد؟(نیمسال دوم ۸۷–۸۸)
الف.C, C++ ب.C, Ada ج.C++, Pascal د.Ada, Pascal

۲۱– در استاندارد IEEE 754 که اعداد ممیز شناور را با سه فیلد تعریف می کند. اگر یک بیت برای علامت (S) و برای نما هشت بیت و برای بیت مانتیس ۲۳ بیت در نظر گرفته شود و اعداد به صورت نرمال شده مد نظر باشند آنگاه برای $0<E<255$ کدامیک از اعداد زیر در حالت کلی در نظر گرفته می شود؟(نیمسال دوم ۸۷–۸۸)
الف. $2^{E-127}(1.M)$ ب. $2^{-126}(0.M)$ ج. $2^{E}(1.M)$ د. $2^{E-127}(0.M)$

۲۲– برای رشته ای با ۱۰ کاراکتر و نمایش حافظه ای به صورت طول متغیر با حد معین اگر بزرگترین رشته ای که می تواند در آن طول قرار بگیرد ۱۴ فرض شود آنگاه نمایش مربوطه چند خانه از حافظه را رزرو می کند؟(نیمسال دوم ۸۷–۸۸)
الف:۱۴ ب:۱۶ ج:۱۰ د:۱۲

۲۳–در کدام گزینه همه اشیای داده ای ذکر شده، اشیای داده ای هستند که کامپیوتر مجازی آنها را ایجاد می کند تا در حین اجرای برنامه از آنها برای ذخیره داده‌ها استفاده کند و مستقیماً در اختیار برنامه نویس نیستند؟(نیمسال اول ۸۸–۸۹)

الف.ثوابت متغیرها ، آرایه‌ها

ب.متغیرهاو آرایه‌ها ، پشته‌های زمان اجرا ، بافرهای فایل

ج.پشته‌های زمان اجرا و، رکوردهای فعالیت زیربرنامه‌ها، بافرهای فایل ، لیست‌های فضای آزاد

د.فایل‌ها ، بافرهای فایل ، آرایه‌ها ، ثوابت

۲۴– کدامیک از موارد زیر جزو اهداف اصلی اعلان محسوب نمی شود؟(نیمسال اول ۸۸–۸۹)

الف.تعیین مقدار شی داده ب.انتخاب نمایش حافظه

ج.مدیریت حافظه د.عملیات چند ریختی

۲۵–از گزینه‌های زیر کدامیک را می توان به عنوان مزیت اصلی کنترل نوع به روش پویا در نظرگرفت؟(نیمسال اول ۸۸–۸۹)

الف.سهولت در اشکال زدایی برنامه‌ها ب.استفاده کمتر از حافظه

ج.انعطاف در طراحی برنامه د.افزایش سرعت اجرای برنامه

۲۶– در کدام گزینه هر دو زبان دارای تبدیل نوع ضمنی هستند؟(نیمسال اول ۸۸–۸۹)

الف.Pascal وAda ب.C وPL/I ج.C و Pascal د.Adaو PL/I

۲۷– در نمایش حافظه برای رشته‌ها در کدام نمایش زیر رشته به صورت آرایه پیوسته ای از کاراکترها ذخیره می شود و انتهای رشته با تهی مشخص می گردد؟(نیمسال اول ۸۸–۸۹)

الف.طول نامحدود با تخصیص‌های متغیر ب.طول نامحدود با تخصیص ثابت

ج.طول ثابت د.طول متغیر با حد معین

۲۸–صفات شی داده که نام توصیفگر یا descriptor به آنها داده می شود، در چه زمانی ذخیره می شوند؟ (نیمسال دوم ۸۸–۸۹)

الف.زمان اجرا ب. زمان ترجمه

ج.زمان تعریف زبان د. می تواند در زمان ترجمه یا اجرا ذخیره شوند.

۲۹–قطعه برنامه زیر نشان دهنده وجود کدامیک از عوامل زیر می باشد؟ (نیمسال دوم ۸۸–۸۹)

```
Int    func(){
   Static  int   i=0;
   i++;
   return I;
 }
int main(){
 for(int i=0;i<=10;i++)
    cout<<func();
 return 0;}
```

الف.وجود اثر جانبی ب.وجود آرگومان ضمنی ج.وجود سرریز د.وجود خوداصلاحی

۳۰–کدام عملیات زیر برای دستور a:=b*c، یک اثر جانبی می باشد؟ (نیمسال دوم ۸۸–۸۹)

الف.عمل انتساب ب.عمل ضرب

ج. عمل ضرب و انتساب هر دو د. در این دستور اثر جانبی وجود ندارد

۳۱–با توجه به قطعه کد زیر چه نوع خطایی و در چه زمانی رخ داده و یا ممکن است رخ دهد؟ (نیمسال دوم ۸۸–۸۹)

```
Day =1..30;
Day:=0;
For i:=1  to  20 do
   Day:=day+2;
```

الف.کنترل نوع زمان کامپایل و اجرا ب. کنترل نوع زمان اجرا

ج. کنترل نوع زمان کامپایل د. کنترل نوع زمان تعریف زبان

۳۲– قطعه برنامه زیر به کدامیک از امکانات موجود در زبان ML،اشاره دارد؟ (نیمسال دوم ۸۸–۸۹)

```
Fun Pnu(r):float=3.14*r*r;
```

الف.کنترل نوع ایستا ب.تبدیل نوع ج.استنتاج نوع د.کنترل نوع پویا

۳۳–درقطعه کد C زیر، دستور f=a+b/c درعملیات a+b/c بدون عملیات انتساب، تبدیل ضمنی در چه نوع انجام می شود؟ (نیمسال دوم ۸۸–۸۹)

```
int main() {
     float f;     int a,c;     double b;
     f=a+b/c;
     return 0; }
```

الف.int ب.Double ج.Float د.خطا

۳۴–در تعریف زیر وجود آرگومان سراسری g استفاده شده در تابع، نشانگر چیست؟ (نیمسال دوم ۸۸–۸۹)

```
F ( int a,int b) {
   a=10;
   b=a+b;
   g=b;
}
```

الف.اثر جانبی ب.خود اصلاحی ج.نتایج ضمنی د.آرگومان ضمنی

۳۵–کدامیک از روشهای پیاده سازی، برای رشته‌های کاراکتری با طول نامحدود در زبانهای با تکنولوژی جدید پشتیبانی می شود؟ (نیمسال دوم ۸۸–۸۹)

الف.آرایه پیوسته ای از کاراکترها ب.نمایش حافظه پیوندی

ج. نمایش حافظه ترتیبی د.فشرده کردن رشته

۳۶- تکه کد برنامه زیر به کدام مورد اشاره دارد. (نیمسال اول ۸۹-۹۰)

```
Int funct (int &a,int &b)
{
      int m;
      m=a;
      b=a+b;
      return m;
}
```

الف. آرگومانهای ضمنی
ب. حساسیت به سابقه قبلی (گذشته)
ج. آرگومانهای خاص
د. اثرات جانبی

۳۷- با توجه به تکه کد زیر چه نوع خطایی و در چه زمانی رخ داده و یا ممکن است رخ دهد. (نیمسال اول ۸۹-۹۰)

```
Const int k=0;
for i:=1 to 20 do
      k:=k+2;
```

الف. کنترل نوع زمان کامپایل و اجرا
ب. کنترل نوع زمان اجرا
ج. کنترل نوع زمان کامپایل
د. کنترل نوع زمان تعریف زبان

۳۸- در مورد کنترل نوع پویا کدام مورد صحیح نیست؟ (نیمسال دوم ۸۹-۹۰)

الف. لازم نیست هر عملیات به نتایج خود یک نوع را نسبت دهد تا عملیات بعدی بتواند آنها را کنترل کند .

ب. در هر عملیات کنترل نوع صورت می گیرد و در صورتی اجرا می شود که انواع آرگومان درست باشد .

ج. در کنترل نوع پویا در هر شی داده یک برچسب نوع قرار می گیرد که نوع آن را مشخص می کند .

د. کنترل نوع پویا در زمان اجرا انجام می شود .

۳۹- کدام گزینه صحیح است ؟ (نیمسال دوم ۸۹-۹۰)

الف. در کنترل نوع پویا ، ضمن صرف حافظه بیشتر نسبت به نوع ایستا ، برای کاهش برخی هزینه ها ، ممکن است عملیات کنترل نشوند .

ب. برای قوی بودن زبان ، تمام خطاهای نوع ، باید به طور پویا کنترل شوند هرچند هزینه ها بالا می رود .

ج. در کنترل نوع پویا ، ضمن صرف حافظه کمتر نسبت به نوع ایستا ، برای کاهش برخی هزینه ها ، ممکن است عملیات کنترل نشوند .

د. در کنترل نوع پویا ، برای ساختار کنترل بین برنامه ها ، اشاره گر CEP هنگام اشاره به دستور جاری قابل اجرا ، نوع داده های رکورد فعالیت زیر برنامه ها را کنترل می کند.

۴۰- اثر جانبی در دستور a:=b*c حاصل عملیات چیست ؟ (نیمسال دوم ۸۹-۹۰)

الف. عمل ضرب است .

ب. هم عمل ضرب و هم انتساب است .

ج. عمل انتساب است .

د. اثر جانبی در دستور وجود ندارد بلکه عمل خود اصلاحی وجود دارد .

۴۱- آرگومان سراسری h در تابع زیر نشان دهنده چیست؟ (نیمسال دوم ۸۹-۹۰)

```
G (int X , int Y)
{
      x=10;
      y=x+y;
      h=y;
}
```

الف. اثر جانبی ب. آرگومان ضمنی
ج. نتایج ضمنی د. خود اصلاحی

۴۲- در پیاده سازی رشته های کاراکتری ، کدام گزینه صحیح است؟ (نیمسال دوم ۸۹-۹۰)
الف. فقط می توان با طول ثابت و رعایت حد بالا پیاده سازی شوند .
ب. می توان با طول نامحدود پیاده سازی کرد ولی در این حالت باید حد بالا مشخص باشد .
ج. می توان با طول متغیر و بدون هیچ محدودیتی پیاده سازی شوند .
د. می توان با طول ثابت یا متغیر و حد بالای مشخص و یا با طول نامحدود پیاده سازی شوند .

۴۳-کدامیک از اشیا داده زیر توسط برنامه نویس ایجاد می شود؟(نیمسال اول ۹۱-۹۰)
الف. لیست های فضای آزاد ب. پشته های زمان اجرا
ج. فایل د. رکوردهای فعالیت زیربرنامه

۴۴- در زبان هایی که انقیاد نوع در زمان اجرا انجام می شود، از چه ابزاری برای تعیین نمایش حافظه شی داده استفاده می شود؟(نیمسال اول ۹۱-۹۰)
الف. رکورد فعالیت ب. پشته های زمان اجرا
ج. توصیفگر یا بردار خصیصه د. جدول نمادها

۴۵- کدامیک از موارد زیر جز روش های پیاده سازی عملیات یک نوع داده اولیه نمی باشد؟(نیمسال اول ۹۱-۹۰)
الف. به صورت عملیات سخت افزاری ب. به صورت دستورات داخل برنامه
ج. به صورت زیربرنامه یا تابع د. به کمک توصیفگر زمان اجرا

۴۶- کدامیک از موارد زیر از اهداف اعلان اشیا داده نمی باشد؟(نیمسال اول ۹۱-۹۰)
الف. مدیریت حافظه ب. عملیات چندریختی
ج. انتخاب نمایش حافظه د. کنترل نوع پویا

۴۷- کدامیک از جملات زیر در مورد کنترل نوع، صحیح می باشد؟(نیمسال اول ۹۱-۹۰)
الف. زبان پرولوگ، کنترل نوع ایستا را در مورد اشیا داده بکار می برد.
ب. زبان لیسپ، کنترل نوع پویا را در مورد اشیا داده بکار می برد.
ج. در کنترل نوع ایستا، برای هر شی داده یک برچسب نوع قرار می گیرد که نوع آن شی داده را مشخص می کند.
د. امتیاز اصلی کنترل نوع ایستا، آزاد شدن برنامه نویس از بسیاری از محدودیت ها است.

۴۸- چه زبان های برنامه نویسی از نظر نوع، نوع قوی محسوب می شود؟(نیمسال اول ۹۱-۹۰)

الف. اگر اجرای تابعی به نام F با امضای $F:S \rightarrow R$ بتواند مقداری خارج از R تولید کند، آن زبان از نظر نوع قوی محسوب می شود.

ب. اگر زبان تمام خطاهای نوع را به صورت ایستا برطرف کند،آن زبان از نظر نوع قوی محسوب می شود.

ج. اگر در زبانی برنامه نویس بتواند آزادانه و فارغ از امنیت نوع برنامه نویسی کند، آن زبان از نظر نوع قوی محسوب می شود.

د. زبان هایی که کنترل نوع پویا دارند، نوع قوی محسوب می شوند.

۴۹- کدامیک از زبان های زیر از تبدیل نوع ضمنی استفاده می کنند؟(نیمسال اول ۹۱-۹۰)

الف .Ada ب. Pascal ج.C و Pascal د.C و PL/I

۵۰- کدامیک از انواع داده ای زیر توسط سخت افزار پشتیبانی می شود؟(نیمسال اول ۹۱-۹۰)

الف. نوع شمارشی ب. نوع اعشاری ج. رشته های کاراکتری د. آرایه

سوالات تشریحی

۱–چه عواملی باعث می شوند که تعریف عملیات زبان برنامه سازی به صورت تابع ریاضی دشوار شود؟(نیمسال اول ۸۵–۸۶)

۲–عملیات اصلی بر روی فایل‌های ترتیبی را شرح دهید؟(نیمسال دوم ۸۵–۸۶)

۳–سه نوع نمایش حافظه برای مقادیر صحیح را رسم کنید و در مورد هریک توضیح دهید؟(نیمسال دوم ۸۶–۸۷)

۴–نمایش‌های حافظه را برای مقادیر حقیقی ممیز ثابت و حقیقی ممیز شناور و موهومی و گویا رسم کنید و هر یک را بطور مختصر خط توضیح دهید؟(نیمسال اول ۸۷–۸۸)

۵–تبدیل نوع و انواع آن را شرح دهید؟(نیمسال دوم ۸۷–۸۸)

۶–مهمترین هدف اعلان چیست ؟ آن را به طور کامل توضیح دهید.(نیمسال اول ۸۹–۹۰)

۵-۹- پاسخنامه سوالات تستی فصل پنجم

سوال	الف	ب	ج	د
۱	*			
۲			*	
۳		*		
۴		*		
۵			*	
۶				*
۷				*
۸		*		
۹		*		
۱۰				*
۱۱				*
۱۲				*
۱۳		*		
۱۴			*	
۱۵	*			
۱۶			*	
۱۷			*	
۱۸		*		
۱۹				*
۲۰				*

سوال	الف	ب	ج	د
۲۱	*			
۲۲		*		
۲۳			*	
۲۴	*			
۲۵			*	
۲۶		*		
۲۷	*			
۲۸	*			
۲۹				*
۳۰	*			
۳۱	*			
۳۲			*	
۳۳		*		
۳۴				*
۳۵		*		
۳۶				*
۳۷			*	
۳۸	*			
۳۹	*			
۴۰			*	
۴۱		*		
۴۲				*
۴۳			*	
۴۴			*	
۴۵				*
۴۶				*
۴۷		*		
۴۸		*		
۴۹				*
۵۰		*		

فصل ششم:

بسته بندی

آنچه در این فصل خواهید آموخت:

- مشخصات انواع ساختمان داده‌ها
- پیاده سازی داده‌های ساخت یافته
  - نمایش حافظه
  - پیاده سازی عملیات
- مدیریت حافظه و مسئله اشاره گرها
- اعلان و کنترل نوع ساختمان داده‌ها
- بردارها و آرایه‌ها
  - برش آرایه
  - آرایه‌های انجمنی
- رکوردها
  - رکوردهای تودرتو
  - رکورد با طول متغییر
- لیست‌ها
- مجموعه‌ها
- اشیاء داده اجرایی
- نوع داده انتزاعی(A.D.T)
  - پنهان سازی و بسته بندی اطلاعات
- زیربرنامه‌ها
  - تعریف و فراخوانیزیربرنامه
  - زیربرنامه‌های کلی
  - تعریف زیربرنامه به عنوان شی داده
- تعریف نوع
- هم ارزی نوع
- سوالات تستی و تشریحی

۶-۱- مقدمه

به چهار طریق می توان انواع داده جدید و عملیات مرتبط با هریک را بر روی آنها تعریف کرد :

- **ساختمان داده ها**[۱]**:** از نظر مجازی تمام زبان ها خواصی برای ایجاد اشیاء داده پیچیده از انواع اولیه دارند. ساختمان داده، شی داده است که عناصر آن خود اشیاء داده دیگری هستند. داده‌های ساخت یافته پشتیبانی سخت افزاری ندارند و بیش از یک صفت دارند. برای توصیف هر نوع ساختمان داده به یک تو صیفگر[۲]نیاز است که مشخصات آن ساختمان داده را تعیین می کند. لیست‌ها، مجموعه‌ها، آرایه‌ها برای ایجاد گروهی از اشیاء داده همگن و رکوردها مکانیزمی برای ایجاد اشیاء داده غیر همگن هستند.
- **زیر برنامه‌ها:** برنامه نویس می تواند برنامه‌هایی را بنویسد که مانند یک نوع جدید عمل کنند. همچنین در برخی از زبان ها عملیات به عنوان یک نوع جدید محسوب می شود.
- **اعلان نوع:** خود برنامه نویس با استفاده از امکانات زبان یک نوع جدید تعریف می کند. مفهوم نوع داده انتزاعی برای ایجاد انواع جدید به کار برده می شود مانندکلاس در C++ و package در Ada
- **وراثت:** به کمک تکنیک‌های شی گرایی و وراثت می توان انواع جدید و عملیات بر روی آنها تعریف کرد.

دراین فصل بر روی سه مورد اول متمرکز می شویم.

۶-۲- مشخصات انواع ساختمان داده ها

ساختمان داده، شی داده ای که مرکب از چند شی داده دیگر است. عناصر ساختمان داده را اجزا یا مولفه های آن می نامند که هر جزء آن می تواند یک عنصر اولیه یا یک ساختمان داده دیگر باشد. بعضی از انواع ساختمان داده شامل آرایه‌ها، رکوردها، پشته‌ها، لیست‌ها و مجموعه‌ها و .... می باشد. صفات یک ساختمان داده عبارتند از:

- **تعداد اجزاء:** اگر تعداد عناصر ساختمان داده درطول عمرش ثابت باشد اندازه ساختمان داده ثابت و گرنه اندازه ساختمان داده متغیر است. آرایه‌ها و رکوردها ساختمان داده‌هایی با اندازه ثابت و پشته، لیست و مجموعه‌ها نمونه‌هایی از ساختمان داده با طول متغیر هستند رشته‌ها به هر دو شکل ثابت و متغیر وجود دارند. اشیاء داده طول متغیر با استفاده از اشاره گرها، اشیاء داده طول ثابت را به هم پیوند می دهند.
- **نوع هر عنصر:** اگر همه عناصر ساختمان داده از یک نوع باشند به آن ساختمان داده همگن و در غیر این صورت، آن را ناهمگن گویند. آرایه‌ها، مجموعه‌ها، رشته‌ها از نوع همگن و رکوردها و لیست‌ها عموماً ناهمگن هستند.
- **اسامی برای انتخاب عناصر:** هر ساختمان داده باید مکانیزمی داشته باشد که بتوان اجزاء آنرا انتخاب کرد مثلاً در آرایه به کمک اندیس، هر عنصرانتخاب می شود. در رکورد این نام توسط برنامه نویس مشخص می شود. در پشته به وسیله اشاره گر top و در فایل به وسیله اشاره گر فایل seek
- **حداکثر تعداد عناصر:** برای ساختمان داده هایی با طول متغیر مثل رشته یا پشته حداکثر طول آنها باید بر حسب تعداد عناصر مشخص شود.
- **سازمان عناصر:** متداول ترین سازمان، دنباله خطی از عناصر است مانند آرایه‌های یک بعدی، رکوردها، رشته‌ها و... در یک فضای پیوسته از حافظه ذخیره می شوند.

---

[۱] Data structure
[۲] descriptor

عملیات عمومی که در انواع ساختمان داده انجام می گیرند عبارتند از:

- **عملیات انتخاب عنصر:** دو نوع عملیات انتخاب وجود دارند که به اجزای ساختمان داده‌ها دستیابی دارند انتخاب تصادفی (مستقیم[۱]) و انتخاب ترتیبی[۲]. در انتخاب تصادفی اجزاء به صورت دلخواه انتخاب می شوند ولی در انتخاب ترتیبی اجزاء به ترتیبی که از قبل مشخص شده، دستیابی می شوند. به عنوان مثال در پردازش یک بردار از عملیات اندیس برای دستیابی تصادفی به اجزاء استفاده می شود. (v[4]) یا رکورد همراه با اسم جزء (R.ID) ولی در لیست‌ها باید پردازش ترتیبی صورت گیرد تا به عنصر مورد نظر برسیم.
- **عملیات روی کل ساختمان داده:** عملیات ممکن است کل ساختمان داده‌ها را به عنوان آرگومان گرفته و ساختمان داده جدیدی را تولید کند مانند جمع دو آرایه، انتساب رکوردی به رکورد دیگر یا عملیات اجتماع بر روی مجموعه‌ها. زبان هایی مانند APL و Snobol4 تعداد زیادی از این عملیات را پشتیبانی می کنند.
- **درج وحذف عناصر:** عملیاتی که تعداد عناصر ساختمان داده را تغییر می دهند.
- **ایجاد و حذف ساختمان داده‌ها :** عملیاتی که ساختمان داده‌ها را ایجاد یا حذف می کنند.

بین عملیات ارجاع و عملیات انتخاب تفاوت وجود دارد . وقتی می نویسیم v[4] یعنی می خواهیم به عنصر چهارم v دسترسی پیدا کنیم که برای این کار دو مرحله شامل عملیات ارجاع و عملیات انتخاب انجام می شود عملیات ارجاع مربوط به v[4]، موقعیت فعلی نام v که یک اشاره گر (آدرس شروع آرایه) است را تعیین می کند و عملیات انتخاب دقیقاً مکان آن عنصر در آرایه را نشان می دهد.

۶-۳- پیاده سازی انواع ساختمان داده‌ها

پیاده سازی ساختمان داده ها از دو جنبه نمایش حافظه و پیاده سازی عملیات روی ساختمان داده‌ها بررسی می شود.

۶-۳-۱- نمایش حافظه

هنگام ذخیره سازی ساختمان داده، حافظه ای برای عناصر ساختمان داده و توصیفگر اختیاری که تمام یا چند صفت ساختمان داده را ذخیره می کند در نظر گرفته می شود. دو نمایش حافظه ای برای ساختمان داده‌ها وجود دارد :

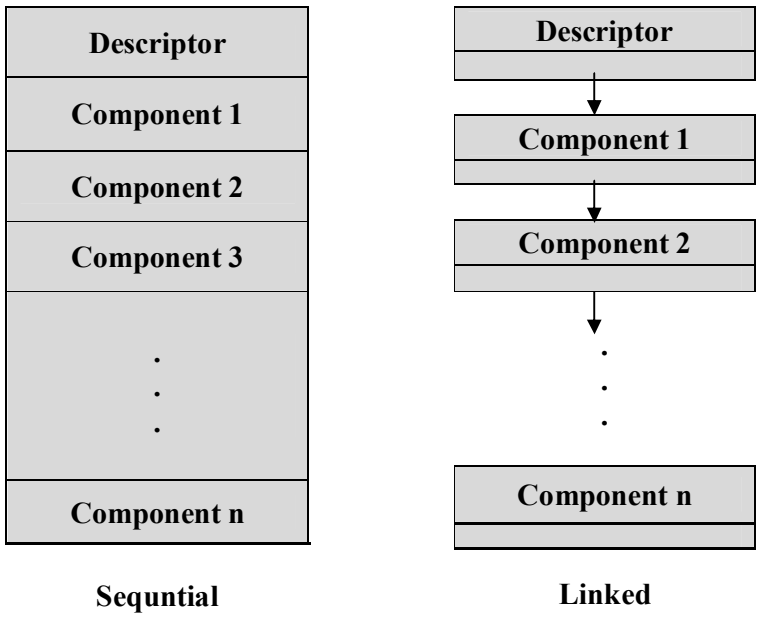
الف- ترتیبی: در این نمایش ساختمان داده در یک بلوک پیوسته از حافظه ذخیره می شود که شامل توصیف گر و اجزاء می باشد.

ب- پیوندی: در این نمایش ساختمان داده در چندین بلوک ناپیوسته از حافظه ذخیره می شوند که بلوک‌ها از طریق اشاره گر به یکدیگر پیوند می خورند. اشاره گر از بلوک A به بلوک B، پیوند نام دارد.

نمایش ترتیبی برای ساختمان داده با طول ثابت و گاهی ساختمان داده با طول متغییر همگن (رشته‌ها و پشته‌ها) استفاده می شود در حالیکه نمایش پیوندی اغلب برای ساختمان داده‌هایی با طول متغیر مثل لیست‌ها به کار گرفته می شود.

---

[۱] Direct
[۲] Sequential



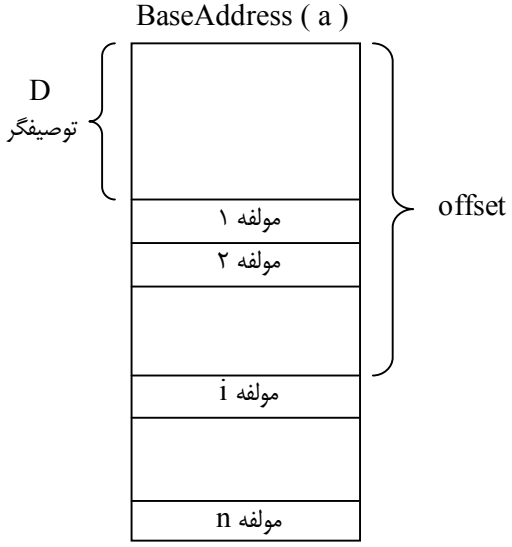
شکل ۶ -۱نمایش حافظه برای ساختمان داده‌های خطی

۶-۳-۲- پیاده سازی عملیات

انتخاب اجزا (مولفه ها) ساختمان داده، مهمترین نکته در پیاده سازی آن است و هدف آن است که انتخاب ترتیبی و تصادفی عناصر کارامد باشند.

الف- انتخاب ترتیبی در حافظه: عملیات انتخاب عنصر در نمایش ترتیبی حافظه، به کمک یک آدرس مبنا و یک آدرس آفست به صورت (Base+offset) به سادگی و با سرعت انجام می شود محل نسبی عنصر انتخاب شده در بلوک ترتیبی، افست و محل شروع بلوک، آدرس پایه نام دارد.

ب- انتخاب پیوندی در حافظه: برای انتخاب یک جزء در ساختمان داده در روش پیوندی می بایست تمام عناصر اول تا iام برای رسیدن به عنصر i ام دستیابی شوند.



Offset (i) = D + (i-1)*Component size

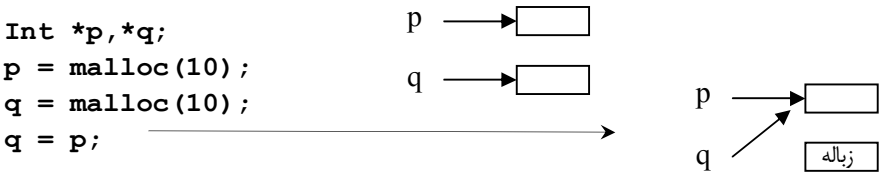
آدرس مولفه i ام = a * offset (i)

شکل ۶ -۲

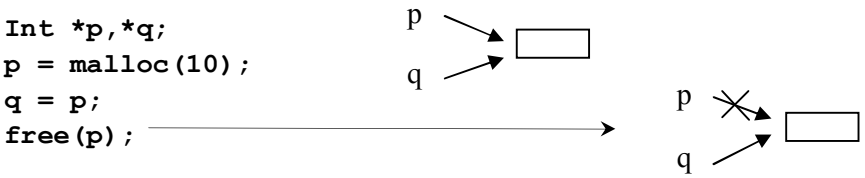
۶-۴- مدیریت حافظه و مسائل اشاره گرها

هنگامی که به یک شی داده حافظه تخصیص داده شد طول عمر آن شروع و هنگامی که این انقیاد از بین برود (حافظه از شی داده پس گرفته شود) طول عمر آن هم تمام می شود. برای ساختمان داده‌هایی با طول متغیر، هر یک از عناصر ساختمان داده، طول عمر مخصوص به خودشان را دارند ولی در ساختمان داده با طول ثابت، کلیه عناصر ساختمان داده دارای طول عمر یکسان هستند. هنگام ایجاد یک شی داده، یک مسیر دستیابی به آن نیز ایجاد می شود این مسیر دستیابی از طریق نام یا به کمک اشاره گری قابل دسترسی است. در هر نقطه از طول عمر یک شی داده، ممکن است چندین مسیر دستیابی به آن وجود داشته باشد مثلا آرگومانی به زیر برنامه ارسال می شود یا اشاره گر جدیدی به آن اشاره می کند (ارجاع سرگردان) یا مسیرهای دستیابی ممکن است به روشهای گوناگونی از بین بروند مثل انتساب مقدار جدیدی به متغیر اشاره گر (زباله). بنابراین طول عمر شی داده و مسیرهای دستیابی اثرات متقابلی با هم دارند و در این زمینه دو مسئله مهم در مدیریت حافظه وجود دارد :

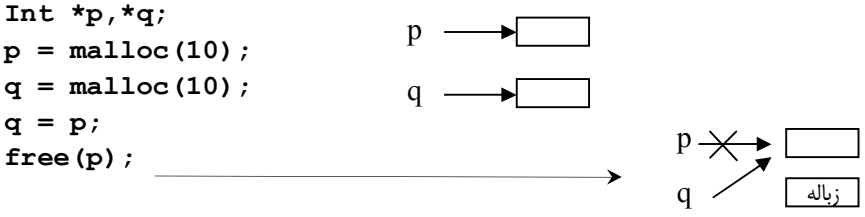
الف-داده‌های زباله[۱]: هنگامی که یک مسیر دستیابی به یک شی داده از بین برود ولی خود شی داده وجود داشته باشد، شی داده را زباله گوییم زیرا دیگر قابل استفاده نیست ولی انقیاد آن به محل حافظه اش از بین نرفته است. قطعه کد زیر مثالی از داده زباله را نشان می دهد:



ب-ارجاع معلق[۲]: اگر طول عمر شی داده تمام شده باشد ولی هنوز یک مسیر دستیابی به آن را در اختیار داشته باشیم ارجاع معلق رخ داده است در حقیقت ارجاع معلق یا سرگردان، یک مسیر دستیابی است که پس از اینکه طول عمر شی داده خاتمه یافت، هنوز وجود داشته باشد. این مشکل هنگامی رخ می دهد که چند اشاره گر به صورت همزمان به یک خانه حافظه اشاره کنند و یکی از آن اشاره گرها آزاد شود ولی آدرس حافظه مربوطه در اشاره گر دیگری ذخیره شده باشد. قطعه کد زیر مثالی از ارجاع سرگردان را نشان می دهد:



حال اگر دستور free(p) را اجرا کنیم حافظه مذکور رها شده ولی هنوز p به آن اشاره می کند.
ممکن است هر دو مشکل فوق به صورت همزمان در برنامه رخ بدهند. به مثال زیر توجه کنید :



[۱] Garbage data
[۲] dangling reference

مسئله داده‌های زباله چندان مهم نیست چون داده‌های زباله حافظه مصرف می کنند و باعث به هدر رفتن حافظه می شوند و نهایتا یک برنامه به دلیل عدم حافظه کافی اجرا نخواهد شد ولی ارجاع معلق مسئله مهمی در مدیریت حافظه است چرا که اگر از طریق ارجاع سرگردان به یک شی داده دسترسی داشته باشیم می توانیم محتویات آن خانه حافظه را بدون اجازه کاربر دستکاری کنیم و این باعث به وجود آمدن نوعی ناهنجاری اطلاعاتی می شود.

یک راه حل جهت رفع مشکل زباله استفاده از فیلد اضافی به نام Reference count است. هنگامی که به یک اشاره گر فضایی را اختصاص می دهیم به صورت خودکار فیلد مذکور ساخته می شود که مقدار درون این فیلد، بیانگر تعداد اشاره گرهایی است که به آن خانه حافظه اشاره می کند با اضافه شدن اشاره گری به این محل حافظه، یک واحد به این اشاره گر افزوده می شود و با آزاد شدن هر کدام از این اشاره گرها یک واحد از آن کم می شود. هرگاه count=0 شد یعنی هیچ اشاره گری به آن اشاره نکرده و بنابراین می توان فضای مربوطه را آزاد کرد. این عمل آزادسازی توسط واحدی به نام جمع آوری زباله[۱] صورت می گیرد.

۶-۵- اعلان و کنترل نوع ساختمان داده‌ها

یک اعلان داده برای ساختمان داده، صفات متعددی را برای آن مشخص می سازد مثلاً اعلان زیر در پاسکال :

```
A=array [1..20,-4..8]of real;
```

مشخص می کند که نوع داده آرایه ای است دو بعدی و اندیس سطرها از یک تا بیست و اندیس ستون ها از ۴- تا ۸ است تعداد سطرها ۲۰ و تعداد ستونها ۱۳ و در نتیجه کل خانه‌ها ۲۶۰ عدد است و نوع هر خانه نیز اعشاری است.

درهنگام کنترل نوع ساختمان داده دو موضوع مهم باید در نظر گرفته شود :

الف- وجود مولفه انتخابی: جزء انتخابی ممکن است در ساختمان داده نباشد به عنوان مثال عملیات اندیس که جزئی از آرایه را انتخاب می کند. ممکن است اندیس خارج از محدوده آرایه بوده و اصلا عنصر مربوطه وجود نداشته باشد. کنترل زمان اجرا باید معتبر بودن آنرا بررسی کند.

(مثال `A[1..10]` غیرقانونی `A[13]`

ب- نوع عنصر انتخابی: اگر عنصری وجود داشته باشد آنگاه باید نوع آن هم درست باشد. مثلا در عبارت A[2][3].link->item در زبانC هنگام ترجمه باید نوع عنصری که از این طریق بدست می آید مشخص و درست باشد این کنترل به صورت ایستا و در زمان کامپایل انجام می گیرد.

۶-۶- بردارها و آرایه‌ها

مشخصات:

بردار (آرایه ای یک بعدی) ساختمان داده ای مرکب از تعداد ثابتی از عناصر هم نوع است که به شکل دنباله خطی سازمان دهی شده است. آرایه‌های دو بعدی (ماتریس) نیز به همین روش تعریف می شوند و هر بردار دارای تعدادی صفات نظیر: تعداد عناصر، نوع عناصر، اندیس برای انتخاب هر عنصر است.

- **تعداد عناصر:** معمولاً توسط بازه که برای اندیس مشخص شده، تعیین می گردد.
- **نوع هر عنصر:** تمام عناصر بردار از یک نوع هستند.
- **اندیس برای انتخاب هر عنصر**

[۱] Garbage Callection

عملیات روی آرایه ها:

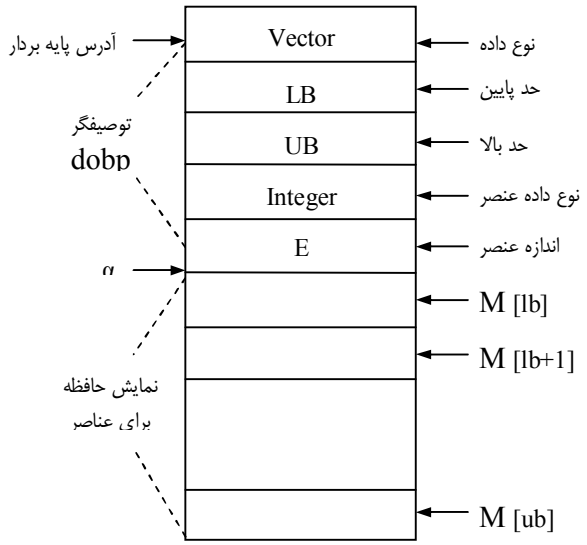
یکی از مهم ترین عملیات در مورد آرایه ها، انتخاب عنصر می باشد. عملیاتی که عنصری از آرایه را انتخاب می کند اندیس گذاری[۱] نام دارد مثل A[2]. بعضی دیگر از عملیات عبارتند از: ساخت و از بین بردن آرایه ها، انتساب به عناصری از آرایه و اعمالی مثل جمع دو آرایه که روی بردارهایی با طول یکسان انجام می گیرند. عملیات محاسباتی بر روی آرایه ها (یا عملیاتی مثل ضرب داخلی آن ها)، به صورت حلقه هایی پیاده سازی می شوند که به ترتیب به همه عناصر آرایه دستیابی دارند. در اکثر زبانها عملیات بر روی آرایه ها بسیار محدود است ولی درAPL این عملیات بسیار زیاد می باشند.

پیاده سازی :

پیاده سازی آرایه ها را از دیدگاه نمایش حافظه و پیاده سازی عملیات دسترسی به عنصر، مورد بررسی قرار می دهیم.

نمایش حافظه:

نمایش حافظه بردار (آرایه یک بعدی) به صورت ترتیبی است. هر بردار، توصیفگری در زمان ترجمه دارد که برای کنترل نوع، نمایش حافظه و محاسبه فرمول دستیابی (تابع دستیابی) به کار می رود. نمونه ای از نمایش حافظه برداری به نام M را با توصیفگر کامل در شکل زیر آمده است.



شکل ۶ -۳نمایش توصیفگر کامل برای بردار M

ذخیره نمودن آرایه‌های یک بعدی (بردار) در حافظه کامپیوتر به سادگی انجام می گیرد. اما در آرایه‌های چند بعدی (ماتریس ها) مسئله بغرنج تر خواهد شد. یکی از وظایف کامپایلرها، نگاشت آرایه چند بعدی و عناصر آن به آدرس فیزیکی حافظه است که برای این کار به یکسری محاسبات و فرمول هایی نیاز داریم. برای ذخیره آرایه های دو بعدی از روش سطری و ستونی استفاده می کنیم. در روش سطری، ابتدا سطر اول، سپس سطر دوم، ...و سپس سطر آخر ذخیره می شوند در حالیکه در روش ستونی، ابتدا ستون اول، سپس ستون دوم، ...و سپس ستون آخر ذخیره می شوند. بسیاری از زبانها مانند C ، پاسکال و جاوا از روش سطری و فرترن از روش ستونی استفاده می کنند.

[۱] subscripting

به عنوان مثال، ماتریس x را در نظر بگیرید:

$$x = \begin{bmatrix} 3 & 4 & 7 \\ 6 & 2 & 5 \\ 1 & 3 & 8 \end{bmatrix}$$

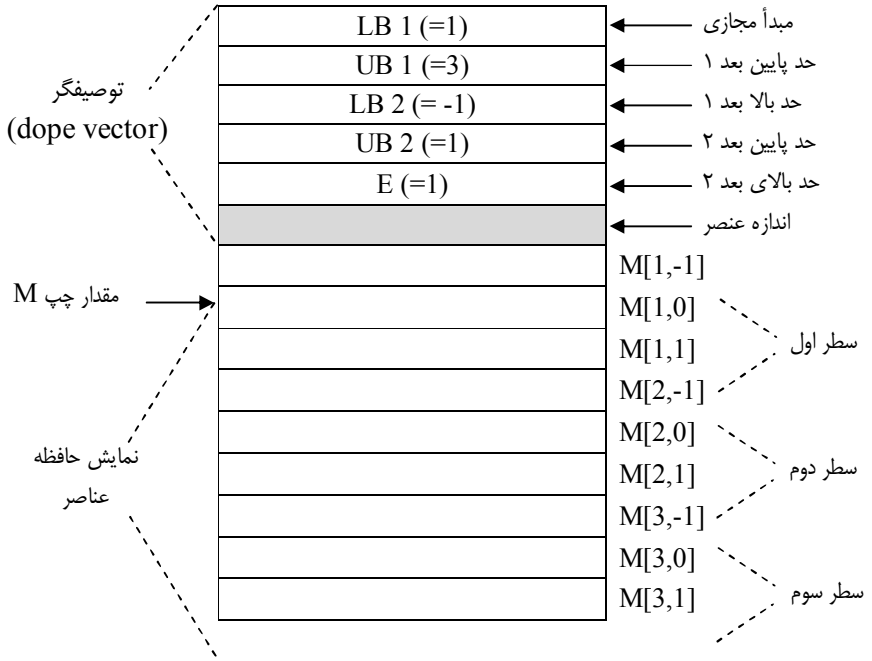
در نمایش سطری، عناصر آرایه x به ترتیب زیر ذخیره می شوند:

3 , 4 , 7 (سطراول) , 6 , 2 , 5 (سطردوم) , 1 , 3 , 8 (سطرسوم)

در نمایش ستونی،ابتدا ستون اول، سپس ستون دوم، ...و سپس ستون آخر ذخیره می شوند. بنابر این، ماتریس x در نمایش ستونی به صورت زیر ذخیره خواهد شد:

3 , 6 , 1 (ستون اول) , 4 , 2 , 3 (ستون دوم) , 7 , 5 , 8 (ستون سوم)

نمایش حافظه آرایه دو بعدی، مانند بردار، شامل حافظه هایی برای توصیفگر و عناصر آرایه است. نمایش سطری آرایه M[1..3,-1..1] در شکل زیر آمده است.



شکل ۶ – ۴نمایش حافظه برای آرایه دوبعدی M به صورت سطری

پیاده سازی عملیات بردار

یکی از مهم ترین عملیات در مورد بردار، انتخاب عنصر می باشد بدیهی است که پیاده سازی بردار، دشوار تر از انواع داده اولیه ای مانند نوع داده صحیح است. کد مربوط به دستیابی به عناصر آرایه باید در زمان ترجمه تولید شود. در زمان اجرا، این کد اجرا می شود تا آدرس عنصر مورد نظر تولید شود. اگر کد دستیابی به عنصر به خوبی طراحی نشود، هزینه دستیابی به آن زیاد خواهد بود. نحوه محاسبه آدرس یک عنصر در آرایه های یک بعدی، دوبعدی و سه بعدی را در زیر بررسی می کنیم.

الف- اگر α آدرس شروع اولین عنصر آرایه در حافظه و n اندازه هر قلم عنصر آرایه باشد و همچنین L و U کران های بالا و پایین آرایه یک بعدی A به صورت زیر باشند آنگاه آدرس عنصر A[i] به صورت زیر محاسبه می شود.

$A: Array[L\ ..U\ ]\ of\ items; \qquad sizeof(items) = n;$

تعداد عناصر $= (U - L + 1)$

میزان حافظه مصرفی $= (U - L + 1) \times n$

آدرس عنصر $A[i] = (i - L) \times n + \alpha$

ب- اگر α آدرس شروع اولین خانه آرایه در حافظه و n اندازه هر قلم عنصر آرایه باشد و همچنین L_1, L_2 کران های پایین و U_1, U_2 کران های بالا آرایه دوبعدی زیر باشند آنگاه آدرس عنصر A[i,j] به صورت زیر محاسبه می شود.

$A: Array[L_1...U_1, L_2...U_2]\ of\ items; \qquad sizeof(items) = n;$

آدرس عنصر به روش سطری $A[i,j] = [(i - L_1)(U_2 - L_2 + 1) + (j - L_2)] \times n + \alpha$

آدرس عنصر به روش ستونی $A[i,j] = [(j - L_1)(U_1 - L_1 + 1) + (i - L_1)] \times n + \alpha$

ج- اگر α آدرس شروع اولین خانه آرایه در حافظه و n اندازه هر قلم عنصر آرایه باشد و همچنین L_1, L_2, L_3 کران های پایین و U_1, U_2, U_3 کران های بالا آرایه دوبعدی زیر باشند و پایین آرایه یک بعدی به صورت زیر باشند آنگاه آدرس عنصر A[i,j,k] به صورت زیر محاسبه می شود.

$A: Array[L_1...U_1, L_2...U_2\ , L_3...U_3]\ of\ items; \qquad sizeof(items) = n;$

آدرس عنصر به روش سطری $A[i,j,k] = [(i - L_1)(U_2 - L_2 + 1)(U_3 - L_3 + 1) + (j - L_2)(U_3 - L_3 + 1) + (k - L_3)] \times n + \alpha$

آدرس عنصر به روش ستونی $A[i,j,k] = [(k - L_3)(U_2 - L_2 + 1)(U_1 - L_1 + 1) + (j - L_2)(U_1 - L_1 + 1) + (i - L_1)] \times n + \alpha$

مثال : آرایه چهار بعدی A[1..10][1..10][1..20][1..10] مفروض است که به روش سطری ذخیره شده است اگر آدرس شروع حافظه صفر باشد و تعداد بایتهای هر عنصر آرایه ۲ باشد آدرس A(1,2,2,2) را حساب کنید (تذکر: در پاسکال اندیس از یک شروع می شود و در C از *صفر شروع می شود*).(جواب نهایی:۲۲۲)

برای اطلاعات بیش تر در این زمینه به بخش آرایه ها از درس ساختمان داده ها مراجعه کنید

نمایش حافظه به صورت فشرده و غیر فشرده :

در نمایش حافظه فشرده عناصر یک بردار به صورت فشرده در حافظه ذخیره می شوند و به این نکته توجه نمی شود که هر عنصر باید از کلمه آدرس پذیر شروع شود لذا این روش باعث صرفه جویی در حافظه می شود ولی دستیابی به عنصر هزینه زیادی می برد زیرا نمی توان از فرمول دستیابی استفاده کرد به دلیل گران بودن هزینه دستیابی بردارها به شکل غیر فشرده ذخیره می شوند هر عنصر در مرز یک واحد حافظه آدرس پذیر قرار می گیرد و بین هر جفت از عناصر ممکن است حافظه بدون استفاده باقی بماند. مزیت این روش دستیابی سریع است ولی حافظه به هدر می رود.

a1	a1		
a2	a2		
a3	a3		
a4	a4		
a5	a5		

نمایش حافظه به صورت غیر فشرده

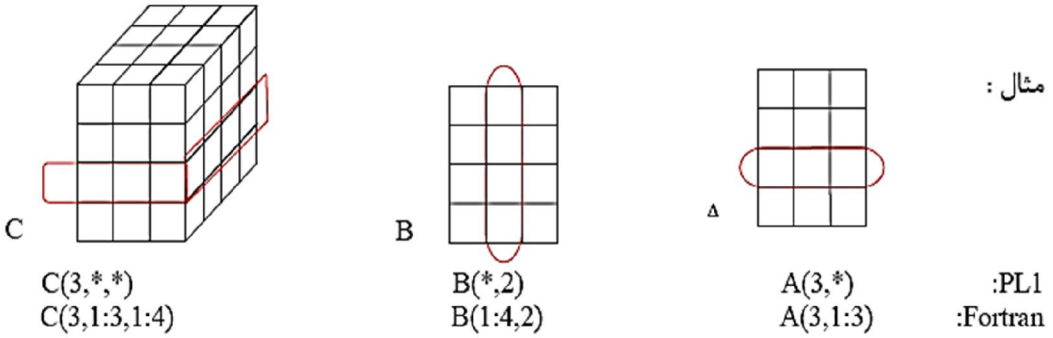
a1	a1	a2	a2
a3	a3	a4	a4
a5	a5		

نمایش حافظه به صورت فشرده

شکل ۶ - 5

۶-۶-۱- برش[۱] آرایه

برش آرایه بخشی از آرایه است که خود نیز یک آرایه می باشد. شکل زیرمثال هایی از این مفهوم را نشان می دهد.



شکل ۶-۶

PL/I از اولین زبانهایی بود که مفهوم برش‌ها را پیاده سازی کرد اگر اعلان A به صورت A(3,4) آنگاه برش شکل الف به صورت A(*,2) و برش شکل ب به صورت B(3,*) برش شکل ج به صورت C(3,*,*) نمایش داده می شود در فرترن می توان بخشی از آرایه (برش) را به عنوان آرگومان به زیر برنامه‌ها ارسال کرد فرترن ۹۰ برش‌ها را به صورت زیر معرفی کرد :
A(1:4,2) ;B(3,1:3); C(3,1:3,1:4)

پیاده سازی:

استفاده از توصیفگر منجر به پیاده سازی کارآمد برش ها می شود. به عنوان مثال آرایه ۴*۳ را می توان با استفاده از توصیفگر به صورت زیر توصیف نمود.

Vo	α
Lb1	1
Ub1	4
Multiplier1	3
Lb2	1
Ub2	3
Multiplier1	1

جدول ۶-۱

در این توصیفگر، multiplier2 که اندازه شی داده (هر عنصر آرایه) است، فاصله بین عناصر متوالی آرایه را نیز نشان می دهد. بنابراین، بخش ستون دوم آرایه xیعنی A(*,2) را می توان به صورت زیر نما یش داد:

Vo	α-3
Lb1	1
Ub1	4
Multiplier1	3

جدول ۶-۲

این توصیفگر، برداری به طول ۴ را نشان می دهد که اولین عنصر آن یک محل پس از اولین عنصرx قرار دارد و هر عنصر در ۳ محل بعدی موجود است که با multiplier1 مشخص شده است.

---

[۱] Slice

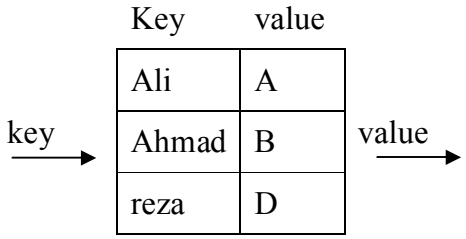
۶-۶-۲- آرایه‌های شرکت پذیر[۱] (انجمنی)

در برخی از کاربردهای برنامه نویسی مطلوب است که به کمک نام و بدون استفاده از اندیس بتوانیم به اطلاعات دسترسی داشته باشیم مثلا اگر بخواهیم از طریق نام دانشجو به نمره آن برسیم یک راه پیاده سازی معمولی، استفاده از آرایه دو ستونی است که مثلا از طریق name[i] به grade[i] برسیم راه دیگر استفاده از آرایه انجمنی است که از دو ستون key و value تشکیل شده است. آرایه‌های انجمنی در Snobol4 به صورت جدول وجود دارند و در زبان‌های فرایندی مثل پرل اهمیت زیادی دارند.

مثال: در زبان پرل آرایه انجمنی به کمک عملگر % به وجود می آید.

```
%ClassList = ("Ali",'A',"Ahmad",'B',"Reza",'D');
```

نحوه دستیابی : دارای مقدار "A " است $ClassList{"Ali"}



شکل ۶ -۷

آرایه‌های شرکت پذیر توسط توصیفگر پیاده سازی می شوند.

۶-۷- رکوردها

مشخصات:

رکورد ساختاری متشکل از تعداد ثابتی عنصر با نوع‌های متفاوت می باشد رکوردها و بردارها ساختمان داده‌هایی خطی با طول ثابت هستند ولی رکوردها از دو جنبه با بردارها فرق دارند الف- عناصر رکورد ممکن است ناهمگن و متفاوت باشند. ب- عناصر رکورد دارای نام هستند.

مثال: نحوه تعریف رکورد در زبان C که به آن ساختمان[۲] گفته می شود.

```
Struct employee{
       int ID;
       int Age;
       float Salary;
  }A;
```

A متغییری از نوع Employee است و نحوه دستیابی به عناصر مانند A.ID=125 است صفات این رکورد عبارتنداز:

۱ -تعداد عناصر۲-نوع هرعنصر۳-نام گذاری هرعنصر

عناصر رکورد را معمولا فیلدها تشکیل می دهند رکوردها درزبان C ساختمان نامیده می شوند. انتخاب عنصر، مهمترین عمل دررکورد است مثل انتخاب A.Age این عمل معادل عمل اندیس گذاری در آرایه است ولی با این تفاوت که اندیس در رکورد، نام یک عنصر است.

---

[۱] Associative arrays
[۲] struct

عملیاتی که بر روی کل رکورد انجام می شود اندک هستند. معمولا رکوردهایی با ساختار یکسان را می توان به یکدیگر نسبت داد.

```
Struct employee type inputrec;
        .
        .
        .
employee = inputrec;
```

دراینجا صفت inputrec مانند employee است. تناظر اسامی بین رکوردها نیز مبنایی برای انتساب در کوبول و Pl/I است مثلا درکوبول توسط دستور MOVE COPRESPANDING می توان دو رکورد را به یکدیگر انتساب داد به طوریکه اجزای متناظر به یکدیگر انتساب داده شوند. اسامی متناظر در دو رکورد باید همنام و هم نوع باشند ولی لازم نیست ترتیب آنها یکسان باشد.

```
MOVE COPRESPANDING  inputrec  TO  employee
```

عملیات انتساب کامل یک رکورد به رکورد دیگر به این صورت پیاده سازی می شود که محتویات بلوک حافظه از رکورد اول در بلوک حافظه رکورد دوم کپی می شود.

پیاده سازی:

برای پیاده سازی رکورد از یک بلوک حافظه که در آن عناصر به ترتیب ذخیره می شوند استفاده می گردد. ممکن است برای هرعنصر از توصیفگری استفاده شود ولی اغلب برای کل رکورد نیاز به توصیفگر نیست فرمول دستیابی برای محاسبه آدرس محل i امین عنصر به صورت زیر است.

$$\text{آدرس عنصر i ام رکورد} = \alpha + \sum_{j=1}^{i-1} (\text{اندازه فیلد } j)$$

α آدرس پایه بلوک حافظه ای است که R را نشان می دهد (j ام . R)

آرایه ای از رکوردها : می توان از برداری استفاده کرد که هر یک از عناصرش رکورد باشد. مثلا در زبان C:

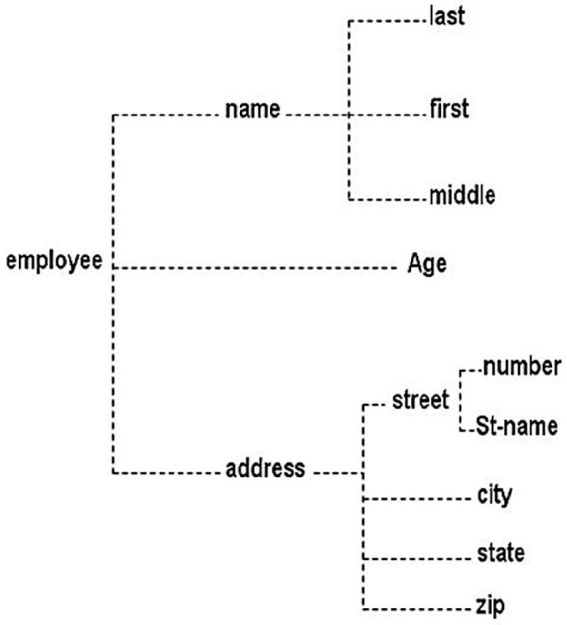
```
Struct employee type
{ int    ID;
   int    Age;
   float   salary;
char   Dept;
} employee[500];
```

این دستور برداری ۵۰۰ عنصری تعریف می کند که هر یک از عناصر آن یک رکورد employee type است عنصر از ساختمان داده مرکب توسط دنباله ای از عملیات انتخاب می شود. مثل employee[3].salary

رکورد تودرتو[1] : عناصر رکورد می تواند آرایه و یا رکورد دیگری باشد و این روند می تواند تا چندین سطح ادامه داشته باشد و یک ساختار سلسله مراتبی را ایجاد کند در کوبول و PL/I این سازمان سلسله مراتبی از نظر نحوی با شماره‌هایی مشخص می شود.

[1] Nested record

```
1 employee    رکورد
      2 Name    رکورد
            3 last char(10)
            3 first char(15)
            3 middle char(1)
      2 Age fixed(2)    عنصراولیه
      2 Address    رکورد
            3 street    رکورد
                  4 number fixed(5)
                  4 st_name char(20)
            3 city char(15)
            3 state char(10)
            3 zip fixed(5)
```



شکل ۶ – ۸

رکورد با طول متغیر :

فرض کنید بخواهیم اطلاعات دانشجویان را ذخیره کنیم. از آنجائیکه اطلاعات دانشجویان متنوع و از انواع داده های مختلف می باشند بهترین ساختمان داده برای ذخیره این اطلاعات، ساختمان داده رکورد است. تمامی دانشجویان یکسری اطلاعات مشترک مانند نام، شماره دانشجویی، کد ملی، معدل و..... دارند. علاوه بر این فرض کنید دانشجویان به دو دسته دانشجویان مجرد و متاهل تقسیم بندی می شوند که برای دانشجویان متاهل باید یکسری اطلاعات (فیلدهای) اضافی از قبیل تعداد فرزندان، تاریخ ازدواج و ... را ذخیره کنیم. یعنی باید رکورد را طوری تعریف کنیم که یک بخش آن برای همه دانشجویان ثابت و مشترک باشد و بخش دیگر بر اساس فیلد برچسب (tag) متغییر باشد. به اینگونه رکوردها، رکورد با طول متغییر گویند. در تعریف رکورد با طول متغییر، ابتدا بخش ثابت و سپس بخش متغییر ظاهر می شود. معمولا بخش متغییر رکورد بر اساس فیلد برچسب تعیین می شود. بخش ثابت رکورد قبل ازcase و بخش متغییر بعد از case شروع می شود.

حال به عنوان مثالی دیگر، فرض کنید می خواهیم اطلاعات کارکنان را ذخیره کنیم. دو گونه کارمند وجود دارد کارکنان استخدامی که ماهیانه حقوق می گیرند و کارکنان دیگر بر اساس میزان ساعت کارکرد حقوق دریافت می کنند. برای ذخیره این اطلاعات می توان از مفهوم رکورد با طول متغییر استفاده کرد . تعریف این رکورد متغییر در پاسکال به شکل زیر است:

```
type PayType=(Salaried, Hourly);
var Employee:record
      ID  : integer;    Dept: array[1..3] of char;
      Age : integer;
      case PayClass : PayType of
      Salaried:(MonthlyRate:real; StartDate  :integer);
      Hourly:(HourRate:real; Reg :integer;Overtime:integer);
end;
```

عنصر payclass در پاسکال برچسب[۱] و در زبان Ada متمایز کننده نامیده می شود. ۳ فیلد ID,Age,Dept برای تمام رکوردها ثابت است. درادامه اگر Salaried=payclass باشد رکورد شامل دو فیلد MonthlyRate,Startdate نیز خواهد بود و اگر Hourly=payclass باشد رکورد شامل سه فیلد HourRate,Reg,Overtime خواهد بود. نحوه پیاده سازی رکورد متغیر فوق به شکل زیر است یعنی حافظه مورد نیاز رکورد به اندازه بخش ثابت (قسمت بالایcase)+ بزرگترین قسمت بخش زیرین case در نظر گرفته می شود.

<table>
<tr><td colspan="2">ID</td></tr>
<tr><td colspan="2">Dept</td></tr>
<tr><td colspan="2">Age</td></tr>
<tr><td colspan="2">PayClass</td></tr>
<tr><td>MonthlyRate</td><td>HourRate</td></tr>
<tr><td>StartDate</td><td>Reg</td></tr>
<tr><td></td><td>Overtime</td></tr>
</table>

STORAGE IF PayClass = Salaried **STORAGE IF PayClass = Hourly**

شکل ۶ -۹

درحین ترجمه، میزان حافظه مورد نیاز برای عناصر هر دو شکل از رکورد تعیین می شود و حافظه به اندازه بزرگترین شکل ممکن تخصیص می یابد. شکل کوچکتر نمی تواند از کل فضا استفاده کند درحین ترجمه نیازی به توصیف گر خاصی برای رکورد با طول متغییر نیست زیرا خود عنصر برچسب به عنوان عنصری دیگر از رکورد درنظر گرفته می شود.

در رکوردهای معمولی تمام فیلدهای آن در طول عمر رکورد وجود دارند ولی در مورد رکوردهایی با طول متغیر، برخی فیلدهای آن، زمانی وجود دارند و زمانی دیگر وجود ندارند. در مثال فوق فیلد employee.reg در زمانی از اجرای برنامه ممکن است وجود نداشته باشند لذا این موضوع باید کنترل شود که دو روش برای این کار وجود دارد.

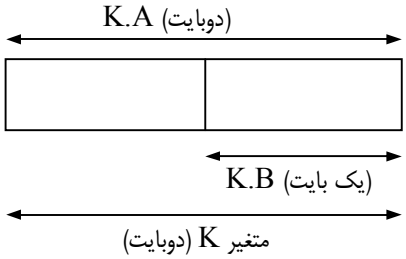
- **کنترل پویا:** در این شیوه عنصر برچسب (متغیر جلوی case) قبل از دستیابی به یک عنصر کنترل می شود اگر این برچسب مقدار مناسب داشت یعنی آن عنصر قابل دستیابی است و در غیر این صورت یک خطای زمان اجرا، رخ خواهد داد.
- **کنترلی انجام نشود :** در این روش در تعریف رکورد طول متغیر از عنصر برچسب استفاده نمی شود و بنابراین در زمان اجرا کنترلی صورت نمی گیرد و به عبارتی دیگر انتخاب عنصر از این رکورد همواره معتبر در نظر گرفته می شود در این حال اگر عنصر مورد نظر وجود نداشت خطای منطقی رخ می دهد کوبول، PL/I و پاسکال شکل هایی از

[۱] Tag

رکورد طول متغییر بدون برچسب را تدارک می بینند و در C نیز از Union استفاده می شود که فاقد برچسب است در پیاده سازی این گونه‌ها، کنترلی انجام نمی شود.

مثال :

```
union{
      intA; sizeof(int)=2
      char B; sizeof(char)=1
} k;
```



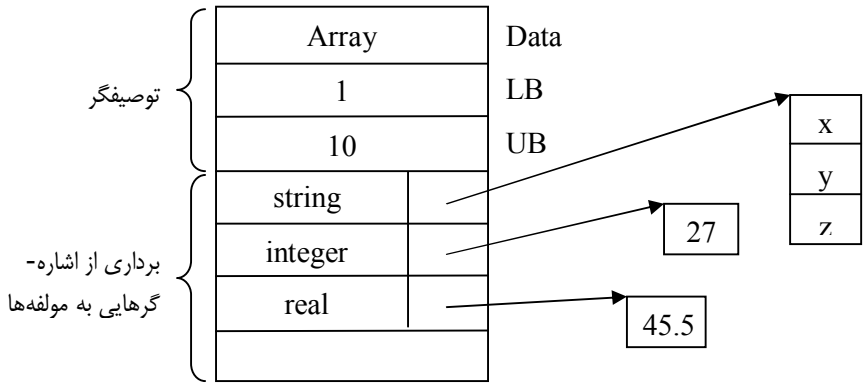
شکل ۶-۱۰

در مفهوم Union برای A,B از یک فضای مشترک استفاده می شود. نوع رکورد متغیر را Union نیز می نامند اگر از فیلد برچسب استفاده نشود (همانند نوع Union در C مثال بالا) نوع را Union آزاد می گویند ولی اگر از فیلد برچسب استفاده شود نوع Unoin را قابل تشخیص می نامند (چون با بررسی برچسب می توان کلاسی که شی داده به آن تعلق دارد را مشخص کرد)

رکوردهای بدون تعریف در زبان‌های برنامه نویسی

در بعضی از زبان‌های برنامه نویسی مانند Lisp,Snobol4 رکوردها از پیش تعیین نمی شوند و درحین اجرای برنامه تولید خواهند شد.

```
Snobol4:
      A:array (10) ;
            A(1)="xyz";
            …
            A(2)=27;
            …
            A(3)=45.5;
```



شکل ۶-۱۱

با توجه به شکل فوق از دیدگاه پیاده سازی، اشاره گرها به نحو موثری برای این هدف بکار گرفته شده اند. همچنین همان طور که در شکل می بینید به ازای هر مولفه، نوع مولفه و اشاره گری به ارزش مولفه ذخیره می شود که معمولاً ارزش‌ها در بلاک-های دیگری از حافظه قرار می گیرند و توسط روتین‌های مدیریت حافظه بکار گرفته می شوند.

۶-۸- لیست‌ها

مشخصات:

لیست دنباله ای از ساختمان داده‌ها است. یک لیست مشابه با بردار شامل دنباله مرتبی از اشیاء می باشد یعنی می توان به اولین عنصر، دومین عنصر و ... دستیابی داشت تفاوت لیست‌ها با بردار عبارت است از: الف) طول لیست اغلب ثابت نیست و در حین اجرای برنامه کم و زیاد می شود ولی طول آرایه اغلب ثابت است ب) عناصر لیست اغلب ناهمگن هستند در حالیکه عناصر آرایه همگن هستند.

لیست‌ها در زبانهایی مثل ML ، لیسپ و پرولوگ به عنوان اشیاء داده اولیه هستند اما در زبانهای کامپایلری مانند C و Ada و پاسکال اینگونه نیست در زبانهای کامپایلری برای ایجاد لیست ها از اشاره گرها استفاده می شود و مدیریت حافظه پویا برای لیست‌ها نیاز است.

نحو زبان لیسپ، لیست را به صورت زیر نمایش می دهد :

Function Name ($Data_1$ $Data_2$ …$Data_n$)

لیسپ با اعمال Function Name بر روی آرگومانهای $Data_1$ $Data_2$ …$Data_n$ اجرا می شود. اغلب عملیات در لیسپ، آرگومانهایی از لیست ها را گرفته، و لیستی را به عنوان خروجی بر می گرداند. به عنوان مثال عملیات Cons دو آرگومان لیست را می گیرد و لیستی را بر می گرداند که آرگومان اول آن به ابتدای آرگومان دوم اضافه می شود. در مثال زیر، لیست خروجی شامل ۴ عنصر می باشد که عنصر اول، لیست (a,b,c) و بقیه عناصر از نوع اتم هستند.

```
Cons '(a b c)'(d e f)) = ((a b c)d e f)
```

نحو لیست در ML به صورت [a,b,c] است. لیست‌ها در ML همگن هستند یعنی می توان لیستی از مقادیر صحیح مثل [1,2,3] یا لیستی از رشته‌ها ["abc","def"] داشت.

پیاده سازی :

برای پیاده سازی لیست‌ها از نمایش حافظه پیوندی استفاده می شود یک قلم لیست، یک عنصر اولیه است که معمولاً شامل شی داده ای با اندازه ثابت است در لیسپ به سه فیلد از اطلاعات نیاز داریم یک فیلد نوع و دو فیلد اشاره گر لیست.

فرم کلی هر گره

Type	Head	Tail

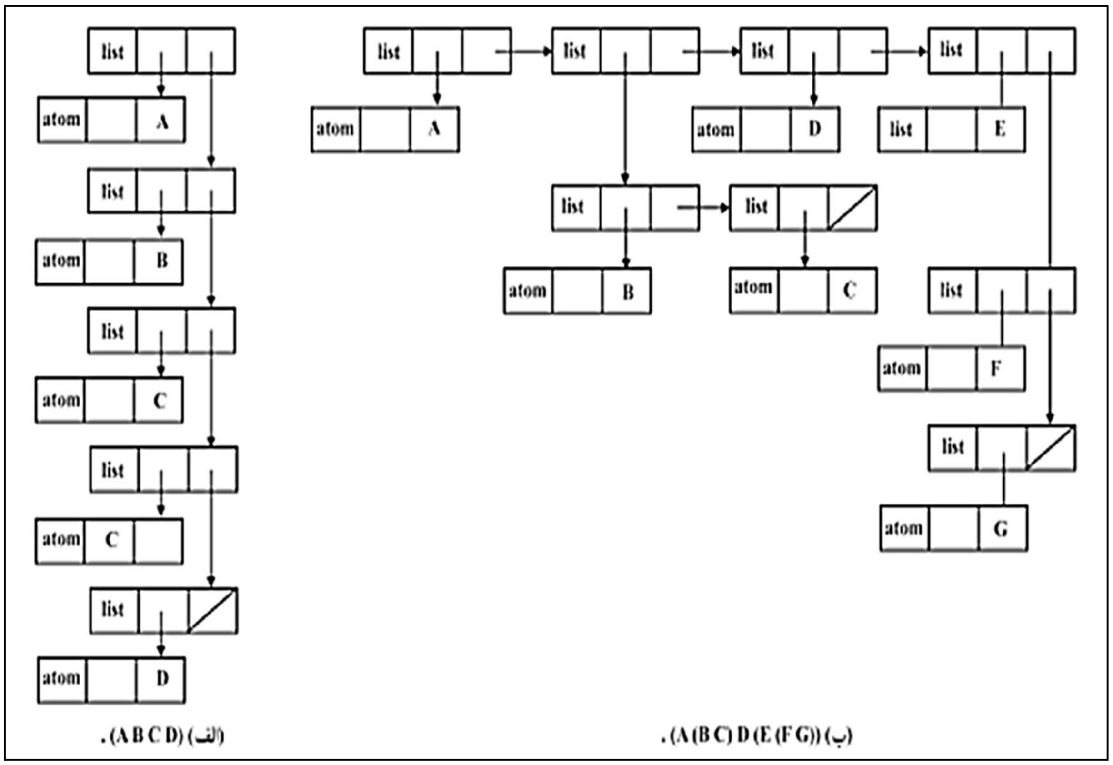
شکل ۶ -۱۲

اگر فیلد نوع اتم باشد آنگاه فیلدهای باقی مانده توصیفگرهایی هستند که این اتم را توصیف می کنند.

اگرفیلد نوع یک لیست باشد اشاره گر اول راس لیست[۱] (اولین عنصر لیست) در حالیکه اشاره گر دوم انتهای لیست[۲] (بقیه اعضا)می باشد.

---

[۱] head
[۲] tail



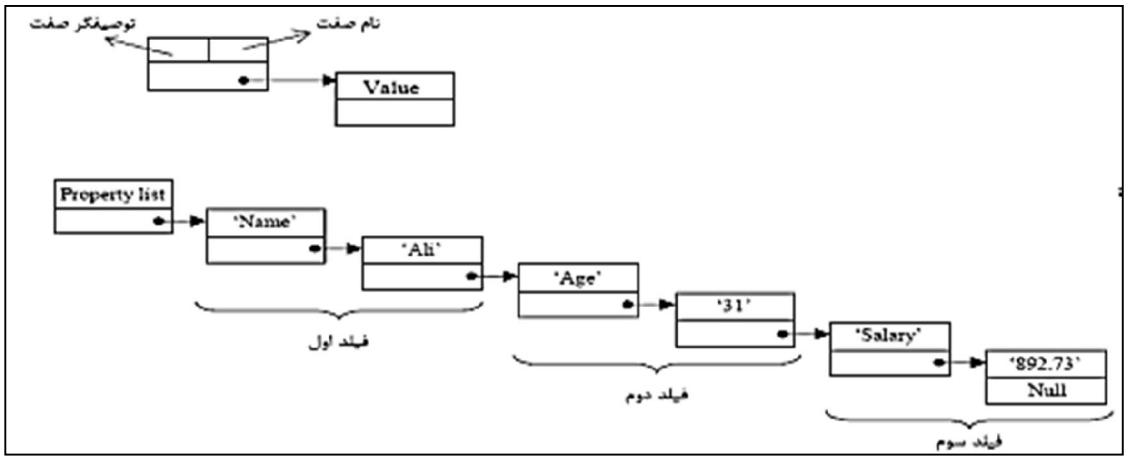
شکل ۶ -۱۳

شکل‌های گوناگون لیست‌ها :

در برخی از زبانها شکل های مختلفی از لیست‌ها وجود دارد مثل پشته‌ها و صف‌ها، درخت ها، گراف جهت دار، لیست‌های خاصیت.

- **پشته‌ها و صف‌ها** : پشته لیستی است که انتخاب، درج و حذف عناصر از انتهای آن انجام می شود. صف لیستی است که انتخاب و حذف از یک طرف و درج از طرف دیگر انجام می شود پشته زمان اجرا یک شی داده مهم است که توسط سیستم تعریف می شود. صف‌ها نیز در زمانبندی و همزمان سازی زیر برنامه‌ها کاربرد دارند.
- **درخت‌ها :** لیستی که عناصرش علاوه بر اشیا داده اولیه ممکن است شامل لیست نیز باشد درخت نام دارد به شرطی که هر لیست فقط یک عنصر از اولین لیست باشد. درخت‌ها برای نمایش جدول نمادها در کامپایلر به کار می روند.
- **گراف جهت دار**: ساختمان داده ای که عناصر آن با استفاده از الگوی پیوندی خاصی به یکدیگر متصل می شوند (به جای دنباله خطی از عناصر) گراف جهت دار نامیده می شود.
- **لیست خاصیت:** لیست خاصیت، رکوردی است که تعداد عناصر آن بدون هیچ محدودیتی تغییر می کند. توجه داشته باشید که این نوع رکورد با رکورد طول متغیر فرق دارد. در رکورد طول متغییر، رکوردها فقط به شکل هایی می تواند در آید که از قبل تعیین شده اند. در لیست خاصیت اسامی عناصر (فیلدها) و مقادیر آنها با هم ذخیره شوند. نام فیلد نام خاصیت و مقدار متناظر فیلد، مقدارخاصیت نامیده می شود. بنابراین، وقتی خاصیت جدیدی در لیست درج می شود دوعنصر درج می شوند : نام خاصیت و مقدارخاصیت.

لیست توصیفی و لیست خاصیت – مقداری نام‌های دیگربرای لیست خاصیت هستند. راه حل معمول پیاده سازی لیست خاصیت، استفاده از یک لیست پیوندی است که اسامی فیلدها و مقادیر آن پشت سرهم می آیند.



شکل ۶ -۱۴

۶-۹- مجموعه‌ها[1]

مشخصات:

مجموعه، شیء داده ای شامل مقادیر نامرتب و مجزا است درحالیکه لیست شامل تعدادی مقادیر مرتب است که عناصر آن ممکن است تکراری باشند. عملیات روی مجموعه‌ها عبارتند از:

- **عضویت:** آیا مقدار x عضوی از مجموعه S است ؟
- **درج و حذف یک مقدار**: اگر فعلا $x \notin S$ باشد می توان آن را در S درج کرد و چنانچه $x \in S$ باشد می توان آن را از مجموعه S حذف کرد.
- **اجتماع، اشتراک و تفاضل مجموعه‌ها:** $S_1 \cup S_2, S_1 \cap S_2, S_1 - S_2 = S_1 \cap \overline{S_2}$

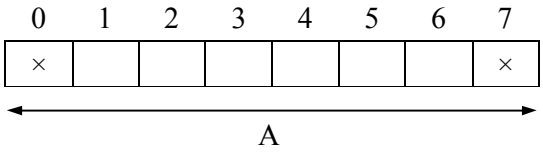
پیاده سازی :

دو روش برای پیاده سازی مجموعه‌ها وجود دارد:

الف- نمایش بیتی :

نمایش حافظه بیتی هنگامی مفید است که اندازه مقادیر مجموعه جهانی (مرجع) کوچک باشد. در این روش هر بیت نمایانگر وجود یا عدم وجود یک عنصر در مجموعه است. به عنوان مثال :

```
For  A=set  of  1..6 ;
```



شکل ۶ - ۱۵

[1] Set

حال اگر در A مجموعه [2,3,5] با دستور A=[2,3,5] ذخیره گردد حافظه به شکل زیر در می آید. در این مثال ۱.۶ مجموعه مرجع است.

0	1	2	3	4	5	6	7
×	0	1	1	0	1	0	×

شکل ۶ -۱۶

در این روش برای درج یک عنصر در مجموعه باید بیت مناسبی را به یک و برای حذف یک عنصر باید بیت مناسبی به صفر تبدیل شود. عضویت را می توان با بررسی بیت مناسب انجام داد. عملیات اجتماع، اشتراک و تفاضل را با عملیات بولی که در سخت افزار وجود دارد پیاده سازی می کنیم عملیات OR بر روی دو بیت نشان دهنده اجتماع، عملیات AND نشان دهنده اشتراک و AND رشته اول با مکمل رشته دوم، عملیات تفاضل را مشخص می کند. به دلیل پشتیبانی سخت افزار از عملیات بیتی، نمایش بیتی مجموعه‌ها کاراتر است اما عملیات سخت افزاری معمولا بر روی بیت‌هایی با طول ثابت انجام می گیرد (مثل طول کلمه حافظه) برای رشته‌های طولانی تر باید به وسیله شبیه سازی نرم افزار به واحدهای کوچکتر تبدیل گردد تا قابل پردازش باشد. پاسکال از روش نمایش بیتی برای پیاده سازی مجموعه ها استفاده می کند.

ب- درهم سازی مجموعه‌ها :

روش دیگر برای نمایش مجموعه‌ها، براساس تکنیک درهم سازی[۱] یا حافظه پراکنده[۲] است. از این روش هنگامی که مجموعه مرجع بزرگ باشد (مثل مجموعه ای شامل اعداد و کاراکترها) مفید است. اغلب زبانهای برنامه نویسی این روش را برای مجموعه‌ها فراهم نمی کنند اما پیاده ساز زبان از این نمایش برای داده‌های تعریف شده توسط سیستم استفاده می کند که در حین ترجمه یا اجرا به آن نیاز است. تقریبا هر کامپایلر با استفاده از روش درهم سازی، اسامی را در جدول نمادها درج می کند. یکی از مشکلاتی که ممکن است در این روش بوجود بیاید برخورد[۳] (تصادم) است. تصادم یعنی برای دو عضو متفاوت از مجموعه، آدرس درهم سازی یکسانی تولید شود. برای مقابله با تصادم سه روش وجود دارد:

- **درهم سازی مجدد:** می توانیم رشته بیتی B_x (عنصر جدید x) را اصلاح کنیم و نتیجه را دوباره درهم سازی کنیم تا آدرس درهم سازی جدید ایجاد شود اگر برخوردی به وجود بیاید مجددا این کار را تکرار می کنیم. منظور از اصلاح کردن انجام یکسری عملیات روی رشته بیتی می باشد مثلا برای اصلاح می توان رشته بیتی را در عددی ضرب یا با عددی جمع کرد.
- **پیمایش ترتیبی:** جستجو را از نقطه برخورد در بلوک شروع می کنیم تا B_x با یک محل خالی در بلوک پیدا شود.
- **باکت بندی:** می توانیم اشاره گرهایی را در بلوک در نظر بگیریم که به لیست‌های باکت پیوندی اشاره کنند که حاوی عناصر با آدرس‌های درهم سازی یکسان هستند پس از هر درهم سازی B_x و بازیابی اشاره گر به لیست باکت مناسب، آن لیست را برای B_x جستجو می کنیم چنانچه پیدا نشد آن را به انتهای لیست اضافه می کنیم.

---

[۱] hashing
[۲] scatter storage
[۳] collision

۶-۱۰- اشیاء داده اجرایی

در اکثر زبانها، مخصوصاً زبانهای کامپایلری مثل C، پاسکال و Ada، برنامه‌های اجرایی و داده‌ها، ساختارهای مجزایی دارند ولی این موضوع الزامی نیست به عنوان مثال، در زبان لیسپ و پرولوگ، دستورات اجرایی می‌توانند خود، داده‌هایی باشند که توسط برنامه‌ها قابل دستیابی و دستکاری هستند مثلاً زبان لیسپ تمام داده‌های خود را در لیست ها ذخیره می‌کند. مانند دستور زیر که تابع f_n را توسط دستور define تعریف می‌کند:

```
(Define fn (cons(a b c) (d e f)))
```

Cons دو لیست را به هم ملحق می‌کند. در پرولوگ عملیات Consult وجود دارد.

۶-۱۱- نوع داده انتزاعی[۱] (ADT)

در زبان های اولیه نظیر کوبول و فرترن، نوع جدید تنها به زیربرنامه‌ها محدود بود ولی در زبان های بعدی امکانات بهتری جهت پیاده سازی نوع داده انتزاعی (ADT) نظیر Package در Ada و کلاس در زبان C++ ارائه گردید.

انتزاع داده‌ها:

برای بسط مفهوم بسته بندی به داده‌هایی که توسط برنامه نویس تعریف می شوند، نوع داده انتزاعی به صورت زیر تعریف می شود :

- مجموعه ای از اشیای داده معمولا با استفاده از یک یا چند تعریف نوع
- مجموعه ای از عملیات انتزاعی بر روی انواع داده
- بسته بندی تمام آنها، به طوری که کاربر نوع جدید نتواند اشیاء داده از آن نوع را، به جز ازطریق عملیاتی که برای آن تعریف شده است، دستکاری کند.

کل تعریف باید طوری بسته بندی شود که کاربر فقط با دانستن نام نوع و معنای عملیات آن، بتواند آن را به کار گیرد. به عنوان مثال برای انواع اولیه مثل حقیقی و صحیح، زبان برنامه سازی، امکانی را برای اعلان متغیرهای آن نوع و عملیاتی را برای آنها تدارک می‌بیند. نمایش حافظه مربوط به مقادیر صحیح و حقیقی کاملاً بسته بندی شده است یعنی از دید برنامه نویس پنهان است. برنامه نویس بدون اینکه از جزئیات نمایش حافظه این انواع اطلاع داشته باشد از اشیای داده آنها استفاده می کند. برنامه نویس فقط نام نوع و عملیات برای آن نوع را فراهم می‌کند.

پنهان سازی اطلاعات[۲]:

برای نوشتن برنامه‌های بزرگ باید از استراتژی تقسیم و حل استفاده کرد. در این استراتژی، برنامه به مجموعه ای از قطعات به نام ماژول[۳]تقسیم می شود. هر ماژول مجموعه محدودی از عملیات را بر روی مقدار محدودی از داده‌ها انجام می‌دهد. طراحی ماژول معمولا به دو صورت انجام می شود: ۱- تجزیه تابعی ۲ – تجزیه داده ای. در تجزیه تابعی، ماژول‌ها تجزیه تابعی برنامه را نشان می دهند، که ساختارهای رویه‌ها، توابع، زیربرنامه‌ها از آن نتیجه می‌شود.

به عنوان مثال برای طراحی برنامه ثبت نام به روش تجزیه تابعی، برنامه به واحدهای تابعی تجزیه می شود که یک برنامه نویس ممکن است توابعی را برای حذف و اضافه دروس، توابع ثبت نام و... ایجاد کند که برای ساختن هر قطعه به اطلاعات اندکی نیاز است، که این عیب (نیاز به داشتن اطلاعات) توسط تجزیه به روش داده ای که همان انتزاع ساده نام دارد از بین می

---

[۱] Abstract Data Type
[۲] Information hiding
[۳] module

رود. برای طراحی برنامه ثبت نام به روش انتزاع ساده، باید نوع داده بخش (section) را ایجاد و از آن در ماژول‌های دیگر استفاده کنیم.

روش های طراحی برنامه مثل اصلاح مرحله ای، برنامه نویسی ساخت یافته، برنامه نویسی پیمانه ای و برنامه نویسی بالا به پایین با طراحی انتزاع سر و کار دارد. زبان برنامه سازی، انتزاع را به دو روش پشتیبانی می‌کند :

- با تدارک کامپیوتر مجازی که کاربرد آن ساده تر و قدرت آن بیش از کامپیوتر سخت افزاری است، مجموعه مفیدی از انتزاع ها را تدارک می‌بیند.
- زبان امکاناتی را فراهم می‌کند که برنامه نویس می‌تواند انتزاع‌ها را به وجود آورد. زیربرنامه‌ها، تعاریف نوع، کلاس‌ها، پکیج‌ها و توابع کتابخانه ای، بعضی از امکاناتی هستند که در زبان های مختلف برای پشتیبانی از انتزاع‌های برنامه نویسی وجود دارد.

در صورتیکه یک زیربرنامه از زیربرنامه دیگری استفاده کند و به جزئیات زیربرنامه استفاده شونده دستیابی نداشته باشد ایده پنهان سازی اطلاعات[۱] پیاده سازی شده است. ایده پنهان سازی اطلاعات، برای استفاده داده نیز معتبر است. مثلاً در مورد تابع sqrt() جزئیات الگوریتم جذرگرفتن و نحوه نمایش اطلاعات از دید کاربر پنهان است. به طور مشابه نوع داده تعریف شده توسط کاربر در صورتی یک انتزاع موفق است که بدون اطلاع از نمایش اشیایی از آن نوع یا الگوریتم‌هایی که توسط عملیات آن استفاده می‌شوند به کار گرفته می‌شوند

در صورتیکه اطلاعات در یک انتزاع بسته بندی شدند معنایش این است که :

الف–کاربر جهت استفاده از انتزاع لازم نیست از اطلاعات مخفی آگاهی داشته باشد.

ب– کاربر نمی‌تواند مستقیماً اطلاعات مخفی را دستکاری کند.

به عنوان مثال نوع داده صحیح در یک زبان برنامه نویسی مثل فرترن یا پاسکال، نه تنها جزئیات نمایش عدد صحیح را پنهان می‌کند بلکه این نمایش را طوری بسته بندی می‌کند که برنامه نویس نمی‌تواند هر یک از بیت‌های نمایش یک مقدار صحیح را دستکاری کند.

نکته:بسته بندی[۲]، اصلاح برنامه را نیز آسان می‌کند. پنهان سازی اطلاعات به طراحی برنامه مربوط می‌شود و هر برنامه ای که به خوبی طراحی شده باشد صرف نظر از زبان مورد استفاده، پنهان سازی اطلاعات امکان پذیر است. بسته بندی به طراحی زبان مربوط می‌شود. یک انتزاع وقتی خوب بسته بندی می‌شود که زبان‌ها، دستیابی به اطلاعات مخفی شده در آن انتزاع را مجاز ندانند.

۶–۱۲– زیربرنامه‌ها

زیربرنامه یک عملیات انتزاعی است که توسط برنامه نویس تعریف می‌شود. دو دیدگاه از زیربرنامه در اینجا مهم است.

در سطح طراحی برنامه، روش نمایش عملیات انتزاعی است که برنامه نویس در مقایسه با عملیات اولیه موجود در زبان. تعریف می‌کند

در سطح طراحی زبان، طراحی و پیاده سازی امکاناتی است که برای تعریف و فراخوانی زیربرنامه تدارک دیده می‌شود.

مشخصات زیربرنامه:

- نام زیربرنامه
- امضای زیر برنامه: که تعداد آرگومان‌ها، ترتیب و نوع هر کدام از آنها و تعداد نتایج و ترتیب و نوع آنها باید مشخص باشد.

---

[۱] Information hiding
[۲] Encapsolation

- عملیاتی که توسط زیر برنامه انجام می‌شود.

زیربرنامه، نمایانگر یک تابع ریاضی است که مجموعه ای از آرگومان ها را به مجموعه ای خاص از نتایج نگاشت می‌کند.
مثال:

```
Float Fn(float x, int y)
Fn : Real * Integer ➔ Real
```

در بعضی از زبان ها برای اعلان زیربرنامه‌ها، از کلمات کلیدی Procedure,Function استفاده می شود.
مثال : تعریف تابع در زبان پاسکال

```
Function Fn (x:Real , y:Integer):Real;
```

رویه[1](زیرروال): اگر زیربرنامه بیش از یک مقدار را برگرداند یا آرگومان‌های خود را تغییر دهد رویه یا زیرروال نامیده می-شود.

```
Void sub ( float x, int y, float *z, int *w)
```

درزبان C, Void نشان می‌دهد که تابع مقداری را برنمی‌گرداند. نام پارامترمجازی که با * مشخص شده است نشان دهنده پارامتری است که تغییرات آن در زیربرنامه، در برنامه فراخوان قابل دستیابی است. نحو این مثال در زبان Ada به شکل زیر است :

```
Procedure sub(x:in real;y:in integer;z:in out real;w:out Boolean)
sub : real1* integer * real2 → real3 * bool :امضاء
```

برچسب‌های in و out و in out سه روش برای ارسال آرگومان‌ها به زیربرنامه را نشان می‌دهد.
in: آرگومانی را مشخص می‌کند که توسط زیربرنامه تغییر نمی‌کند.
in out : آرگومانی را مشخص می‌کند که می‌تواند اصلاح شود.
out : نتایج خروجی را مشخص می کند.

با اینکه زیربرنامه یک تابع ریاضی را نشان می‌دهد اما توصیف دقیق آن با مشکلاتی روبروست که عبارتند از :

- زیربرنامه ممکن است آرگومانهای ضمنی به شکل متغیرهای غیرمحلی داشته باشد.
- زیربرنامه ممکن است اثرات جانبی (نتایج ضمنی) داشته باشد بطوریکه متغیرهای غیرمحلی یا آرگومان های inout را تغییر دهد.
- زیربرنامه ممکن است به ازای بعضی از آرگومان ها تعریف نشده باشد و اگر این آرگومان ها به آن ارسال شوند به طور کامل اجرا نمی‌شوند و کنترل به برنامه استثنا می‌رود.
- زیربرنامه ممکن است به گذشته حساس باشد یعنی نتایج آن به آرگومان هایی بستگی داشته باشد که درفراخوانی قبلی به آن ارسال شده باشد. شاید دلیل آن نگهداری متغیرهای محلی درحین اجراهای مختلف باشد.

[1] Procedure

نکته: دربعضی از زبان‌ها مثل پاسکال، Ada ولی نه در C، بدنه زیربرنامه می‌تواند شامل تعریف زیربرنامه‌های دیگری باشد که درآن زیربرنامه بزرگتر قابل استفاده است. این زیربرنامه‌های محلی طوری بسته بندی می‌شوند که نمی‌توانند در خارج زیر برنامه حاوی آن، فراخوانی شوند.

۶-۱۲-۱- تعریف و فراخوانی (فعالسازی) زیربرنامه

تعریف یک زیربرنامه، خاصیت ایستای آن است. در هر بار صدا زدن زیربرنامه، حین اجرای برنامه، یک رکورد (سابقه) فعالیتی[۱] از زیربرنامه، پدید می‌آید. پس از خاتمه یافتن زیربرنامه این رکورد فعالیت از بین می‌رود. اگر فراخوانی دیگری صورت گیرد رکورد فعالیت جدیدی ایجاد می‌شود. ممکن است چندین رکورد فعالیت از یک زیربرنامه دربرنامه وجود داشته باشد. در واقع تعریف زیربرنامه، یک قالب[۲] برای ایجاد رکوردهای فعالیت آن درحین اجرا می‌باشد. این تمایز شبیه مفهوم کلاس (تعریف زیر برنامه) و شی (سابقه فعالیت) است. در واقع رکورد فعالیت یک زیربرنامه، نوعی شی داده است که در بلوکی از حافظه نشان داده شده و شامل عناصر مرتبط با آن است.

تعریف زیربرنامه :

تعریف چیزی است که در برنامه نوشته می‌شود و تنها اطلاعاتی است که در زمان ترجمه وجود دارد یعنی نوع متغیرهای زیر برنامه مشخص است ولی مقادیر (مقدار راست) یا محل آن (مقدار چپ) مشخص نیست.

فعالیت زیربرنامه :

فعالیت زیربرنامه، فقط درحین اجرای برنامه وجود دارد. در حین اجرا کد مربوط به دستیابی به مقدار راست یا مقدار چپ متغیر می‌تواند اجرا شود. اما نوع متغیر ممکن است وجود نداشته باشد مگر اینکه مترجم اطلاعات را در توصیفگر متغیر ذخیره کرده باشد.

فعالیت زیربرنامه دارای طول عمر است که از فراخوانی زیربرنامه شروع می‌شود و تا از بین رفتن آن ادامه دارد. به مثال زیر توجه کنید :

```
Float FN(float x,int
y)
{
      const intval=2;
      #define finalval
10;
      float M(10);
      int N;
      N=intval;
      if (n<finalval)
      {
            …
      }
      return
(20*x+M(N))
```

مقدمات ایجاد رکوردهای فعالیت
کد اجرایی برای هر دستور زیربرنامه
اختتامیه حذف رکورد فعالیت
20
10
2

سگمنت کد زیربرنامه FN

نقطه برگشت و سایر داده‌های سیستم
داده نتیجه FN
پارامتر :X
پارامتر :Y
شیء داده محلی :M
شیء داده محلی :N

رکورد فعالیت FN (الگو)

شکل ۶ -۱۷

[۱] Activation Record
[۲] Template

- خط امضای FN اطلاعاتی را برای حافظه پارامترها (x و y) و حافظه لازم برای نتیجه تابع ارائه می‌کند نتیجه، شی داده ای از نوع float است.
- اعلان هایی وجود دارند که حافظه را برای متغیرهای محلی (آرایه M و متغیر N) آماده می‌کنند.
- حافظه مربوط به لیترال ها و ثوابت تعریف شده، initval ثابتی با مقدار ۲ و finval ثابتی با مقدار ۱۰ می‌باشد. ۲ و ۱۰ لیترال می باشند.
- حافظه لازم برای کد اجرایی که از دستورات بدنه زیر برنامه تولید می‌شود.

به یکی از خواص مهم C توجه کنید. صفت const به کامپایلر C اطلاع می دهد که شیئ داده initval دارای مقدار لیترال ۲ است. دستور #define یک دستور پیش پردازنده (ماکرو) است که به جای هر finval کاراکترهای "۱۰" را قرار می‌دهد. مترجم C نام finval را پردازش نمی‌کند. اثرات عملی هر دو دستور از زیر برنامه اجرایی یکسان است اما معنای آنها کاملاً متفاوت است. initval دارای یک مقدار چپ است که مقدار راست آن ۲ می‌باشد در حالی که finval فقط دارای مقدار راست ۱۰ است.

هر زیر برنامه دو فضا دارد : ۱- بخش پویا (رکورد فعالیت) ۲- بخش ایستا (سگمنت کد)

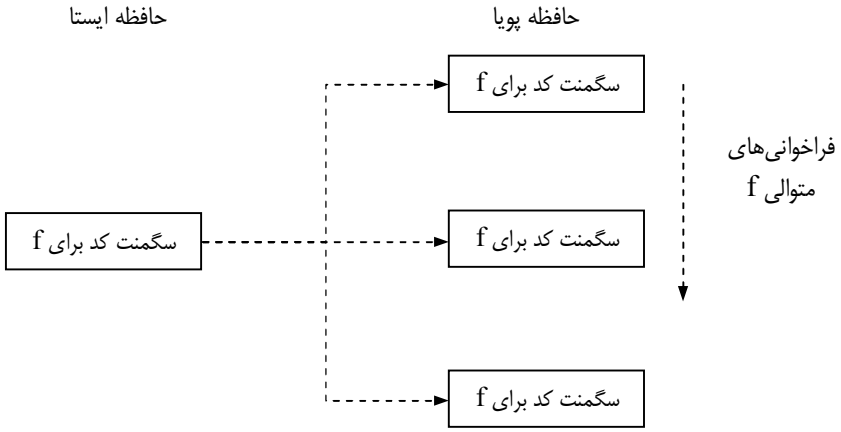
بخش ایستا :

که سگمنت کد[۱] نیز نام دارد حاوی ثوابت و کد اجرایی است. این بخش در حین اجرای زیر برنامه باید ثابت باشد لذا یک کپی از آن بین تمام فعالیت های زیربرنامه به اشتراک گذاشته خواهد شد.

بخش پویا :

که رکورد فعالیت[۲] نیز نام دارد شامل پارامترها، نتایج تابع و داده‌های محلی و ناحیه حافظه موقت، نقاط برگشت و پیوندهایی برای مراجعه به متغیرهای غیرمحلی است. ساختار این بخش برای تمامی فعالیت های زیربرنامه یکسان است اما مقادیر متفاوتی در آنها وجود دارند.

نکته: اندازه و ساختار رکورد فعالیت مورد نیاز برای یک زیربرنامه، می‌تواند در زمان ترجمه تعیین شود یعنی کامپایلر می تواند تعیین کند چند مولفه برای ذخیره داده‌های ضروری در رکورد فعالیت مورد نیاز است. شکل زیر مفهوم دو بخش ایستا و پویا را نشان می‌دهد :



شکل ۶ -۱۸

[۱] Code segment
[۲] Activation record

وقتی زیر برنامه فراخوانی می‌شود چند عمل مخفی صورت می‌گیرد: (Prolog و Epilog)

Prolog : این اعمال باید قبل از اجرای دستورات زیربرنامه گنجانده شوند. انجام این مقدمات توسط مترجم قبل از اجرای کد زیر برنامه انجام می‌شود و شامل عملیاتی چون: تنظیم رکورد فعالیت، انتقال پارامترها، ایجاد پیوند برای ارجاع های غیر محلی و ... (عملیات انتقال پارامترها، push کردن ثبات ها و فلگ ها).

Epilog: این اعمال هنگام خاتمه زیربرنامه انجام می‌شود تا نتایج را برگردانند و حافظه رکورد فعالیت را آزاد کنند. برای این اعمال نیز دستوراتی توسط مترجم در انتهای کد اجرایی قرار داده می‌شود. (عملیات انتقال نتایج، pop کردن ثبات ها و فلگ ها).

دستورات مربوط به Prolog
کد اجرائی
دستورات مربوط به Epilog

۶-۱۲-۲- زیربرنامه‌های کلی[۱]

مشخصات زیربرنامه، تعداد، ترتیب و نوع آرگومان های آن را مشخص می‌کند. زیربرنامه کلی، زیربرنامه ای با یک نام اما چندین امضا (تعریف) متفاوت است. زیربرنامه‌های کلی را زیر برنامه مجدداً تعریف شده نیز می‌گویند. منظور این است که چندین زیربرنامه با اسامی یکسان وجود دارد که تعداد پارامترها، نوع و ترتیب آنها متفاوت است.

```
Procedure     Sum(int a, int b)

Procedure     Sum(int a, float b)

Procedure     Sum(float a, int b)

Procedure     Sum(int a, int b, int c)
```

در مثال های فوق، زیر برنامه جمع برای حالات مختلف با یک نام نوشته شده است. اینگونه موارد را زیر برنامه‌های کلی یا چند هدفه گویند که از جهت پیاده سازی، کامپایلر با توجه به تعداد پارامترها، نوع و ترتیب آنها تشخیص می‌دهد کدامیک از زیربرنامه‌ها را فراخوانی کند و کد لازم برای فراخوانی زیر برنامه مربوطه را تولید کند. در فرترن ۹۰، تعریف مجدد زیربرنامه با بلوک INTERFACE مشخص می‌شود. می‌توانیم این مفهوم را بسط دهیم بطوریکه خود نوع بعنوان پارامتر باشد مثل مکانیزم نوع چند ریختی[۲] در ML.

۶-۱۲-۳- تعریف زیر برنامه به عنوان شیء داده

در اغلب زبان‌های کامپایلری مانند فرترن، جاوا و پاسکال تعریف زیر برنامه، از اجرای آن مستقل می‌باشد برنامه منبع توسط کامپایلر ترجمه شده و به شکل اجرایی در می‌آید. در حین اجرای برنامه، بخش ایستای تعریف زیر برنامه غیر قابل دستیابی و غیر قابل مشاهده است ولی در زبان هایی مانند لیسپ و پرولوگ که اغلب مفسری هستند بین این دو (تعریف و اجرا) تمایزی وجود نداشته و با تعریف زیر برنامه می‌توان مانند اشیای داده زمان اجرا برخورد کرد.

ترجمه: عملیاتی است که تعریف زیر برنامه را به شکل کاراکتری دریافت کرده، شیء داده زمان اجرایی را تولید می‌کند.

اجرا: عملیاتی است که تعریفی به شکل زمان اجرا را گرفته، فعالیتی از آن را ایجاد می‌کند و آن فعالیت را اجرا می‌کند.

---

[۱] Generic subprogram
[۲] Polymorphic

در لیسپ و پرولوگ، ترجمه، عملیاتی است که ممکن است برای شیء داده کاراکتری در زمان اجرا فراخوانی شده تا شکل اجرایی زیر برنامه را تولید نماید. این زبان‌ها عملیاتی به نام define دارند که بدنه زیر برنامه و مشخصات را گرفته و یک تعریف قابل فراخوانی از زیر برنامه را ایجاد می‌کند (define در لیسپ و consult در پرولوگ).
بنابراین در هر دو زبان فوق، می‌توان اجرای برنامه را بدون وجود هیچ زیر برنامه ای آغاز کرد سپس در حین اجرا، می توان بدنه زیر برنامه را خواند یا ایجاد کرد و سپس به شکل اجرایی ترجمه کرد. در حین اجرا، تعریف زیر برنامه می تواند اصلاح شود. لذا در این زبان‌ها، در واقع تعریف زیر برنامه، یک شیء داده است.

۶-۱۳- تعریف نوع[۱]

یک زبان برنامه سازی باید امکاناتی جهت تعریف نوع جدید با توجه به نوع‌های اولیه موجود در زبان فراهم سازد. هر نوع شامل یک Name و یک Description می‌باشد

مثال : `Type s:array[1..10] of real`

در تعریف نوع داده انتزاعی کامل، مکانیزم‌هایی برای توصیف دسته ای از اشیاء لازم است، در زبان‌هایی مانند C، پاسکال و Ada این مکانیزم تعریف نوع، نام دارد. در زبان پاسکال به کمک دستور Type و در زبان C به کمک دستور Typedefمی-توان تعریف نوع را انجام داد. البته باید توجه داشت که تعریف نوع در این حالت، نوع داده انتزاعی کامل را تعریف نمی‌کند چون عملیاتی را بر روی داده‌های آن نوع تعریف نمی‌کند و فقط خود نوع تعریف می‌شود. در زیر مثالی از تعریف نوع در زبان‌های C,Pascal آورده شده است:

Pascal

```
Type Rational:record
     Numerator=integer;
     Denominator: integer;
End;
Var A, B, C: Rational;
```

C

```
Typedef struct Rational Type
{ int numerator;
int Denominator; }Rational;
Rational A,B,C;
```

در تعریف نوع، نام نوع تعیین می‌شود و اعلانی وجود دارد که ساختار دسته ای از اشیاء کلاس را توصیف می‌کند. بدین ترتیب نام نوع، به عنوان نام دسته ای از اشیاء داده محسوب می‌شود و هر وقت به اشیاء داده ای از آن ساختار نیاز باشد به جای تکرار توصیفِ ساختمان داده، کافی است نام نوع ارائه شود. در مثال بالا اگر به رکوردهای A,B,C در پاسکال نیاز داشته باشید و این رکوردها ساختارهای یکسان داشته باشند تعریف نوع فوق (مثال بالا) باعث کاهش تعریف انواع داده خواهد شد. تعریف نوع، علاوه بر ساده کردن ساختار برنامه مزایای دیگری برای برنامه نویس به ارمغان می‌آورد. مثلاً اگر نوع Rational نیاز به اصلاح داشته باشد با استفاده از تعریف نوع، کافی است فقط تغییرات در تعریف نوع ایجاد شود و نه همه متغیرهای تعریف شده از این نوع. مزایای تعریف نوع عبارتند از:

- ساده کردن ساختار برنامه
- جلوگیری از تکرار Typeهای موجود در زبان
- ساده تر کردن انتقال پارامترها
- یک شکل جدید از بسته بندی و پنهان سازی اطلاعات به وجود می‌آورد.

[۱] Type definition

از جهت پیاده سازی، کامپایلر باید هر Type جدید را با توجه به Name و Description در جدولی وارد کند و هنگام اجرای کد و انتقال پارامترها و عملیات دیگر از این جدول استفاده می‌کند.

۶-۱۴- هم ارزی نوع[۱]

کنترل نوع چه به صورت ایستا و چه به صورت پویا مقایسه بین نوع آرگومان‌های واقعی و نوع داده‌هایی است که عملیات انتظار آن را دارد. اگر انواع یکسان باشند آرگومان پذیرفته می‌شود و عملیات ادامه می‌یابد ولی اگر یکسان نباشند یا خطا محسوب می‌شود یا تبدیل ضمنی صورت می‌گیرد و کار ادامه می‌یابد. برای بررسی هم ارز (مساوی) بودن دو نوع داده، دو راه حل وجود دارد: الف-هم ارزی نام ب- هم ارزی ساختار

الف- هم ارزی نام[۲]: دو شی داده از دو نوع، هنگامی هم ارز نام دارند که نام نوع آنها یکسان باشد.

روش هم ارزی نام در Ada ، C++ و پارامترهای زیر برنامه در پاسکال (نه در بقیه موارد) به کار گرفته می‌شود. مزیت هم ارزی نام، پیاده سازی آسان این نوع هم ارزی می باشد.

```
Type   Vect1:array [1...10] of integer;
       Vect2:array [1...10] of integer:
Var X, Z: Vect1;  Y: vect2;
Procedure sub (A: Vect1)
Begin
     ...
end;
Begin
     X:=Z
     X:=Y;
     Sub(Y);
End;
```

(X: =Z) در هم ارزی نام معتبر است چون هر دو متغیر دارای نام نوع یکسان هستند.

(X:=Y) در هم ارزی نام معتبر نیست چون X از نوع Vect1 و Y از نوع Vect2 است.

معایب هم ارزی نام:

- هر شیء که در انتساب به کار می‌رود باید دارای نام باشد یعنی انواع داده بی نام[۳] وجود ندارد. مثلاً در پاسکال داده بی نام زیر را نمی‌توان به عنوان آرگومان به زیر برنامه فرستاد:

```
Var W: array [1…10] of integer;
```

متغیر W نوع مستقلی دارد و نمی‌تواند به عنوان آرگومانِ زیر برنامه باشد، زیرا نوع آن فاقد نام است.

- یک تعریف نوع باید در سراسر برنامه قابل استفاده باشد و یا به عبارتی باید از تعریف نوع عمومی استفاده گردد زیرا نوع شیء داده ای که از طریق زنجیره ای از زیر برنامه‌ها به صورت آرگومان انتقال داده شود نمی‌تواند در هر زیر برنامه تعریف شود. تعریف کلاس‌ها در زبان C++ و اسامی مشخصات پکیج (Package) در Ada و فایل‌های سرایند ".h" در C این کار را تضمین می‌کنند.

---

[۱] Type Equivalence
[۲] Name Equivalence
[۳] anonymous type

ب-هم ارزی ساختاری[1]**:** دو نوع داده ای هنگامی هم ارز ساختار دارند که ساختار داخلی آنها یکسان باشد منظور از ساختار داخلی یکسان، این است که تمام اشیاء داده از یک گونه نمایش حافظه، استفاده کنند. بنابراین در مثال فوق Vect1، Vect2 هم ارز ساختاری دارند زیرا تعداد عناصر، نوع و ترتیب آنها یکسان است. مزیت هم ارزی ساختار این است که انعطاف پذیری برنامه را افزایش می دهد.

معایب هم ارزی ساختاری:

- در مورد رکوردها، آیا اسامی فیلدها باید یکسان باشند یا یکسان بودن تعداد و نوع فیلدها کفایت می‌کند؟ اگر اسامی رکوردها یکسان باشد آیا ترتیب فیلدها هم باید یکسان باشد؟ (ابهام در تعریف هم ارزی ساختار وجود دارد)
- جهت تشخیص معادل بودن Type باید هزینه پرداخت شود یعنی زمان را از دست می‌دهیم، کامپایلر باید زمان صرف کند که آیا دو Type معادلند یا نه؟(هزینه زیاد در تعریف هم ارزی نوع)
- دو متغیر ممکن است به طور تصادفی (بدون آنکه برنامه نویس بخواهد) از نظر ساختار یکسان شوند در حالیکه از دید برنامه نویس متفاوت هستند و توقع دارد زبان برنامه نویسی به او کمک کند و این مورد را خطا بگیرد. مثلاً:

```
Typemeter=integer;
liters=integer;
VarLen: meter;
Vol: Liters;
```

از دید زبان برنامه نویسی Len+Vol صحیح است و هیچ خطایی از طرف کامپایلر گرفته نمی‌شود چون ساختار شان یکسان است. (هم ارزی ساختاری) در حالیکه برنامه نویس می‌خواهد زبان برنامه سازی به او کمک کند، این موضوع بویژه هنگامی که چندین برنامه نویس روی یک برنامه کار می‌کنند بیشتر رخ می دهد.

نکته: در زبان‌های قدیمی مثل فرترن، کوبول و PL/I تعریف نوع وجود ندارد در نتیجه از هم ارزی نام نمی‌توان استفاده کرد و از هم ارزی ساختاری استفاده می‌شود. پاسکال در هم ارزی نوع، با مشکلاتی روبرو است و از هیچ کدام استفاده نمی‌کند (مگر زیر برنامه)، C از هم ارزی ساختاری و C++ از هم ارزی نام استفاده می‌کند، طراحی Ada نیز از هم ارزی نام استفاده می‌کند.

تساوی دو شیء داده

زمانی که دو شی داده دارای نوع مشابه هستند در برخی اوقات این مشکل در زبان مطرح می شود که آیا دو شی داده باهم برابرند یا خیر؟ فرض کنید دو متغیر A,B از نوع X هستند. تحت چه شرایطی می توانید بگویید که A=B است؟ برای مثال برابری دو پشته یا دو مجموعه. که در اولی باید تمامی عناصر تک تک و به ترتیب با هم برابر باشند ولی در دیگری ترتیب مهم نیست. متاسفانه زبان نمی تواند در این مورد کمک کند. دو تعریف زیر را در زبان C برای مجموعه و پشته در نظر بگیرید:

```
Struct stack {                    Struct set {
     int  Topstack                      int numberinset;
     int data[100]; } X,Y;              int data[100]; } A,B;
```

انواع X,Y,A,B از نظر ساختاری هم ارزند. یک مقدار صحیح و آرایه ای ۱۰۰ عنصری از نوع صحیح. ولی تحت چه شرایطی X=Y و A=B می باشد؟

[1] Structural Equivalence

تساوی پشته‌ها:

اگر فرض کنیم topstack به شیء داده موجود در data اشاره می کند که در بالای پشته قرار دارد، تساوی بین x,y را به صورت زیر بیان می کنیم:

۱. x.topstack=y.topstack

۲. برای تمام I های بین ۰ و۱-topstack داشته باشیم x.data[i]=y.data[i]

در این صورت x,y پشته‌های برابری را نشان می دهند.

تساوی مجموعه:

اگر فرض کنیم numberinset تعداد اشیای موجود در A,B است . تساوی بین A,B را به صورت زیر تعریف می‌کنیم:

۱. A.numberinset=B.numberinset

۲. A.data[0]… A.data[numberinset-1] جایگشت B.data[0]…B.data[nomberinset -1] است.زیرا ترتیب درج عناصردر مجموعه مهم نیست.

تعریف انواعی که پارامتر دارند:

در برخی از زبانها مانند Ada، این امکان وجود دارد که در تعریف نوع از پارامتر استفاده شود و بتوان اشیاء داده از نوع یکسان و با اشکال متفاوت ایجاد کرد.به مثال‌هایی در این زمینه توجه فرمائید:

در اینجا maxsize به عنوان پارامتر در تعریف نوع section آمده است:

```
Type section(maxsize:integer) is
Record
    Room:integer;
    Instructor:integer;
    Classsize :integer;
 End record;
```

در این تعریف می توان متغیری به صورت زیر تعریف کرد:

```
X:section(100);
Y:section(25);
```

از جهت پیاده سازی کامپایلر باید ارزش پارامتر را محاسبه کرده سپس با توجه به ارزش پارامتر و تایپ مشخص شده، نوع جدیدی از متغیر تعریف خواهد کرد.در واقع کامپایلر باید دو مرحله طی کند:۱ .ارزشیابی پارامتر ۲. تعریف تایپ جدید. در پاسکال نوع پارامتری امکان پذیر نیست ولی در Ada,C++, ML امکان پذیر است.

مثالی دیگر از زبان Ada برای تعریف صف که از Max به عنوان پارامتر در تعریف نوع داده queue استفاده شده است.

```
Type queue (max:integer) record  is
    Front:integer;
    Rear:integer;
    Items:array[1…max] of  integer;
End record;
```

در اینجا می توان متغیری به صورت زیر تعریف کرد:

```
X:queue(100);
Y:queue(1000);
```

۶-۱۵- سوالات فصل ششم

سوالات تستی

۱- به کدامیک از روشهای زیر نمی توان نوع داده ای جدید را ایجاد کرد؟(نیمسال اول ۸۵-۸۶)

الف.زیربرنامه ب.وراثت ج.اعلان نوع د.چند ریختی

۲- وقتی مسیر دستیابی به یک شی داده از بین برود .....ایجاد می شود؟(نیمسال اول ۸۵-۸۶)

الف.ارجاع معلق ب.زباله ج.انقیاد زودرس د.انقیاد دیررس

۳- در مورد ساختارهای رکورد و آرایه کدام جمله صحیح نیست؟(نیمسال اول ۸۵-۸۶)

الف.عناصر رکورد و آرایه همگن هستند. ب.عناصر رکورد دارای نام هستند.

ج.رکوردها و بردارها ساختمان داده خطی هستند. د.رکوردها و بردارها ساختمان داده خطی با طول ثابت هستند.

۴- نوع رکورد متغیر را .....می نامند. (نیمسال اول ۸۵-۸۶)

الف.رکورد غیر همگن ب.رکورد همگن ج.یونیون د.لیست

۵- اگر در رکوردی با طول متغیری از عناصر، تعداد عناصر بدون هیچ محدودیتی تغییر کند آنرا .....می نامیم. (نیمسال اول ۸۵-۸۶)

الف.درخت ب.گراف ج.لیست خاصیت د.رکورد ناهمگن

۶- بخش پویای سابقه فعالیت مربوط به یک زیر برنامه .....نامیده می شود. (نیمسال اول ۸۵-۸۶)

الف.سگمنت کد ب. سگمنت داده ج.امضای زیربرنامه د.رکورد فعالیت

۷- کدام گزینه غلط است؟(نیمسال دوم ۸۵-۸۶)

الف.در PERL آرایه انجمنی به وسیله عملگر = ایجاد می شود.

ب.طول عمر شی داده با انقیاد شی به محلی از حافظه شروع می شود.

ج.رکورد، ساختمان داده ای مرکب از تعداد ثابتی از عناصر و از انواع مختلف می باشد.

د.هیچکدام

۸- شکل‌های گوناگون لیست‌ها عبارتند از:(نیمسال دوم ۸۵-۸۶)

الف.پشته‌ها و صف‌ها ب.درخت‌ها و گراف‌های جهت دار

ج.الف و ب د.آرایه‌ها

۹- کدام گزینه غلط می باشد؟(نیمسال دوم ۸۵-۸۶)

الف.پنهان سازی اطلاعات , اصطلاح مهمی در طراحی انتزاع‌های برنامه نویسی است.

ب.زبان برنامه سازی انتزاع را به دو روش حمایت می کند.

ج.برای نوشتن برنامه بزرگ باید از استراتژی تقسیم و غلبه استفاده کرد.

د.هیچکدام

۱۰– نوع داده انتزاعی شامل کدام موارد زیر است؟(نیمسال دوم ۸۵–۸۶)

الف.نوع داده ای که توسط برنامه نویس تعریف می شود.

ب.مجموعه ای از عملیات انتزاعی بر روی اشیائی از آن نوع.

ج.بسته بندی اشیای آن نوع بطوریکه کاربر آن نوع، نمی تواند آن اشیا را بدون استفاده از این عملیات دستکاری نماید.

د.کلیه موارد بالا

۱۱– اگر طول اجزای یک ساختمان داده ثابت باشد و اجزای آن همگن باشد در پیاده سازی آن کدام مورد صحیح است؟(نیمسال اول ۸۶–۸۷)

الف.نمایش حافظه پیوندی و هر جز یک توصیف کننده لازم دارد.

ب.نمایش حافظه پیوندی و کل اجزا یک توصیف کننده دارند.

ج.نمایش حافظه ترتیبی و کل اجزا یک توصیف کننده دارند.

د.نمایش حافظه ترتیبی و هر جز یک توصیف کننده لازم دارد.

۱۲– کدام یک از موارد زیر صحیح نیست؟(نیمسال اول ۸۶–۸۷)

الف. در نمایش حافظه پیوندی با عمل انتخاب عنصر تصادفی امکان پذیر است.

ب. در نمایش حافظه پیوندی با عمل انتخاب عنصر ترتیبی امکان پذیر است.

ج. در نمایش حافظه ترتیبی با عمل انتخاب عنصر تصادفی امکان پذیر است.

د. در تمایش حافظه ترتیبی با عمل انتخاب عنصر ترتیبی امکان پذیر است.

۱۳– در صورتی که طول عمر یک شی داده قبل از پایان طول عمر آن از بین برود چه اتفاقی می افتد؟(نیمسال اول ۸۶–۸۷)

الف.مشکلی به نام ارجاع‌های سرگردان بوجود می آید.

ب.مشکلی به نام activation record بوجود می آید.

ج.مشکلی به نام حافظه‌های بدون استفاده بوجود می آید.

د.مشکلی بوجود نمی آید.

۱۴– در صورتی که مسیر دستیابی یک شی داده ای پس از آنکه طول عمر شی داده ای خاتمه یافت وجود داشته باشد، چه اتفاقی می افتد؟(نیمسال دوم ۸۶–۸۷)

الف.مشکلی به نام ارجاع‌های سرگردان بوجود می آید. ب.مشکلی به نام حافظه زباله بوجود می آید.

ج.مشکلی به نام رکورد فعالیت بوجود می آید. د.هیچ مشکلی رخ نمی دهد.

۱۵– برای کنترل ترتیب در عبارات غیر محاسباتی از چه روشی استفاده می شود؟(نیمسال دوم ۸۶–۸۷)

الف.نمایش درختی ب.انتساب اشیای داده ج.تطابق الگو د.هیچکدام

۱۶– تعریف زیر را در نظر بگیرید کدام گزینه صحیح است؟(نیمسال دوم ۸۶–۸۷)

```
Type vect1: array [1...10] of real;
     vect2: array [1...10] of real;
     Var x, z: vect1;
     y: vect2;
```

الف.x,y,z هم ارزی نام دارند.
ب. x,z هم ارزی نام و x,z با y هم ارزی ساختاری دارند.
ج.x با y هم ارزی ساختاری و z با y هم ارزی نام دارند.
د.x و z هم ارزی ساختاری و y با x هم ارزی نام دارند.

۱۷- رکورد متغیر زیر برای تعریف خود به چند بایت نیاز دارد؟(integer دو بایت و real شش بایت و char یک بایت) (نیمسال دوم ۸۶-۸۷)

```
Type paytype=(salaried, hourly);
Var employee: record
Id: integer;
Dept: array [1...3] of char;
Age: integer;
Case payclass: paytype of
Salaried (monthlyrate: real; stardate: integer);
Hourly (hourrate: real; reg: integer; stardate: integer);
Hourly (hourrate: real; reg: integer; overtime: integer);
```

الف.۱۹ ب.۱۸ ج.۱۶ د.۲۷

۱۸- منظور از رکورد فعالیت چیست؟(نیمسال دوم ۸۶-۸۷)
الف.بخش ایستای زیر برنامه ب.بخش پویای زیر برنامه
ج.بخش ایستا به همراه بخش پویا زیر برنامه د.پخش پویا به همراه کد سگمنت زیر برنامه

۱۹- اگر طول اجزای یک ساختمان داده ثابت باشد و اجزای آن همگن باشد در پیاده سازی آن کدام مورد صحیح است؟ (نیمسال دوم ۸۶-۸۷)
الف.نمایش حافظه پیوندی و هر جز یک توصیف کننده لازم دارد.
ب.نمایش حافظه پیوندی و کل اجزا یک توصیف کننده دارند.
ج.نمایش حافظه ترتیبی و کل اجزا یک توصیف کننده دارند.
د.نمایش حافظه ترتیبی و هر جز یک توصیف کننده لازم دارد.

۲۰- عملیات تعریف شده برروی بردارها در کدام یک ازبانهای زیر بیشتر از بقیه است؟(نیمسال دوم ۸۶-۸۷)
الف.Pascal ب.C ج.APL د.Perl

۲۱- کدامیک از زبانهای زیر برای پردازش لیست‌ها می باشند؟(نیمسال دوم ۸۶-۸۷)
الف.Ada ب.Pascal ج.Lisp د.Perl

۲۲- در صورتی که تمام اشاره گرهایی که به شی داده اشاره می کنند از بین بروند چه پدیده ای اتفاق می افتد؟(نیمسال اول ۸۷-۸۸)

الف. مشکلی به نام ارجاع‌های سر گردان بوجود می آید.
ب. مشکلی به نام حافظه زباله بوجود می آید.
ج. مشکلی به نام رکورد فعالیت بوجود می آید.
د. هیچ مشکلی بوجود نمی آید.

۲۳- برای پیاده سازی مجموعه‌ها چنانچه اندازه مجموعه جهانی بزرگ باشد کدام یک از روش‌های نمایش حافظه زیر مناسب است؟(نیمسال اول ۸۷-۸۸)

الف.نمایش بیتی مجموعه‌ها
ب.نمایش درهم سازی مجموعه‌ها
ج.نمایش درختی مجموعه‌ها
د.نمایش بیتی درختی مجموعه‌ها

۲۴- در کدامیک از زبانهای زیر اشیاء داده ای و برنامه‌های اجرایی که دستکاری بر روی اشیا داده ای را انجام می دهند ساختارهای مجزایی ندارند و اصطلاحاً اشیای داده اجرایی داریم؟(نیمسال اول ۸۷-۸۸)

الف.Ada,C ب.C,Lisp ج.Ada,Lisp د.Prolog, Lisp

۲۵- تعریف زیر را در نظر بگیرید کدام گزینه صحیح است؟(نیمسال اول ۸۷-۸۸)

Type

```
vect1: array [1.. 10] of real;
vect2: array [1.. 10] of real;
 Var x, z: vect1; y: vect2;
```

الف.x,y,z هم ارزی نام دارند.
ب.x,z هم ارزی نام وx,z با y هم ارزی ساختاری دارند.
ج.x با y هم ارزی ساختاری و z با y هم ارزی نام دارند.
د.xو z هم ارزی ساختاری و y با x هم ارزی نام دارند.

۲۶-کدام گزینه صحیح است؟(نیمسال اول ۸۷-۸۸)

الف.پنهان سازی اطلاعات،اصطلاح مهمی در طراحی انتزاع‌های برنامه نویسی است.
ب.بسته بندی بر روی آرایه‌های چند بعدی امکان پذیر نیست.
ج.در برخی زبانها بسته بندی به وسیله زیربرنامه صورت می گیرد.
د.الف و ج

۲۷-کدام دسته از زبانهای زیر از آرایه‌های انجمنی استفاده می کنند؟(نیمسال اول ۸۷-۸۸)

الف.C,Pascal
ب.Perl,Snobol4
ج.C,Fortran
د.Pascal,Cobol

۲۸- در صورتی که مسیر دستیابی یک شی داده ای پس از آنکه طول عمر شی داده ای خاتمه یافت وجود داشته باشد چه اتفاقی می افتد؟(نیمسال دوم ۸۷-۸۸)

الف: مشکلی به نام رکورد فعالیت بوجود می آید.

ب. مشکلی به نام ارجاع‌های سرگردان بوجود می آید.

ج. مشکلی به نام حافظه زباله بوجود می آید.

د. مشکلی به نام سرریزی صف بوجود می آید.

۲۹- رکورد متغیر زیر برای تعریف خود به چند بایت نیاز دارد؟(integer دو بایت و real شش بایت و char یک بایت) (نیمسال دوم ۸۷-۸۸)

```
Type paytaype= (salaried, hourly);
Var employee: record
Id: integer;
Dept: array [1...4] of char;
Age integer;
Case payclass: paytype of
Salaried (monthly: integer);
Hourly (hourrate: real; overtimeinteger);
```

الف.۱۹ ب.۱۸ ج.۱۶ د.۲۷

۳۰- برای پیاده سازی مجموعه‌ها چنانچه اندازه مجموعه جهانی کوچک باشد کدام یک از روش‌های نمایش حافظه زیر مناسب است؟(نیمسال دوم ۸۷-۸۸)

الف. نمایش بیتی مجموعه‌ها ب. نمایش درهم سازی مجموعه‌ها

ج. نمایش درختی مجموعه‌ها د. نمایش بیتی درختی مجموعه‌ها

۳۱- برای بسط مفهوم بسته بندی به داده‌هایی که توسط برنامه نویس تعریف می شود نوع داده انتزاعی با فراهم کردن کدامیک از موارد زیر بدست می آید؟(نیمسال دوم ۸۷-۸۸)

مورد اول: مجموعه ای از اشیای داده معمولا با استفاده از یک یا چند تعریف نوع

مورد دوم: مجموعه ای از عملیات انتزاعی بر روی آن انواع داده

مورد سوم: بسته بندی تمام آنها بطوری که کاربر نوع جدید نتواند اشیای داده ای از آن نوع را به جز از طریق عملیاتی که بر روی آن تعریف شده است دستکاری کند.

الف. مورد اول و دوم ب. مورد دوم و سوم ج. مورد اول و سوم د. هر سه مورد

۳۲- رکورد فعالیت یک زیر برنامه شامل کدامیک از موارد زیر نمی باشد؟(نیمسال دوم ۸۷-۸۸)

الف. نتایج تابع و داده‌های محلی ب. ثوابت و کد اجرایی

ج. ناحیه حافظه موقت و داده‌های محلی د. پارامترهای ارسالی و ناحیه حافظه موقت

۳۳- زیر برنامه ای با یک نام اما چندین تعریف که با امضاءهای مختلف مشخص می شوند را چه می نامند؟(نیمسال دوم ۸۷-۸۸)

الف. زیر برنامه کلی ب. زیر برنامه محلی ج. زیر برنامه غیر محلی د. زیر برنامه بازگشتی

۳۴- تعریف روبه رو را در نظر بگرید کدام گزینه صحیح است؟(نیمسال دوم ۸۷-۸۸)

```
Type vect: array [1...10] of real;
Verct2: array [1...10] of real;
Var x, y: vect; z: vect2;
```

الف: x,y,z هم ارزی نام دارند.
ب: x,z هم ارزی نام وx با y هم ارزی ساختاری دارند.
ج: x با y هم ارزی ساختاری و z با y هم ارزی نام دارند.
د: x,z هم ارزی ساختاری وy با x هم ارزی نام دارند.

۳۵- کدام گزینه موجب رخ دادن پدیده زیر می شود؟(نیمسال اول ۸۸-۸۹)
«مقداری به شی داده ای نسبت داده می شود که آن شی وجود ندارد محتویات محلی از حافظه را که به شی داده دیگری اختصاص یافته است تغییر دهید وممکن است محلی از حافظه را که توسط مدیریت حافظه تنظیم شده است خراب کند»
الف.ارجاع‌های معلق ب.زباله ج.پشته‌های زمان اجرا د.بافر صفحه کلید

۳۶- تکه برنامه زیر کدامیک از مشکلات مدیریت حافظه را در زبان C++ ایجاد می کند؟ (نیمسال دوم ۸۸-۸۹)

```
Int *p,*q;
p =new(int);
q=new(int);
p=q;
```

الف.اختصاص حافظه ب.زباله ج. ارجاع معلق د.آزادسازی حافظه

۳۷- اگر در رکوردی با تعداد متغیری از عناصر، تعداد عناصر بدون هیچ محدودیتی تغییر کند، آن رکورد چه نام دارد؟ (نیمسال دوم ۸۸-۸۹)
الف.رکورد با طول متغیر ب. لیست خاصیت ج.یونیون آزاد د.یونیون قابل تشخیص

۳۸- کدام روش پیاده سازی مجموعه‌ها برای نمایش مجموعه‌هایی است که مجموعه مرجع آنها کوچک است؟ (نیمسال دوم ۸۸-۸۹)
الف.نمایش درهم سازی مجموعه‌ها ب.نمایش حافظه پراکنده
ج.نمایش بیتی د. نمایش حافظه ترتیبی

۳۹- در تکه کد زیر آدرس فیلدهای x وy در رکورد rec نسبت به هم چگونه اند؟ (نیمسال دوم ۸۸-۸۹)

```
int main() {
    struct record {
int x; char y; };
    union record rec;
    return 0; }
```

الف.آدرس x>y است. ب.آدرس y>x است.
ج.آدرس شروع فیلدهای x و y یکسان است. د.آدرس انتهای فیلدهای x و y یکسان است.

۴۰- قسمتی از حافظه stack اجرای برنامه که شامل پارامترها، نتایج تابع داده‌های محلی و ناحیه حافظه موقت است چه نام دارد؟ (نیمسال دوم ۸۸-۸۹)

الف.سگمنت کد ب.رکورد فعالیت ج.بافر د.حافظه هرم

۴۱- در کدامیک از زبانهای زیر برای پیاده سازی لیستها سیستم مدیریت حافظه مخفی وجود دارد؟ (نیمسال دوم ۸۸-۸۹)

الف. Ada ب. Pascal ج. C++ د. ML

۴۲- در تعریف آرایه زیر در زبان پاسکال ، طول آرایه در چه زمانی مشخص می شود؟ (نیمسال دوم ۸۸-۸۹)

```
Name : packedarray[1..20] of char;
```

الف.زمان تعریف زبان ب. زمان اجرا ج.زمان پیاده سازی د.زمان کامپایل

۴۳- در زبانی که از هم ارزی استفاده می‌کند ، تعریف زیر وجود دارد. کدام گزینه درست است. (نیمسال اول ۸۹-۹۰)

```
Type
      x=array[1..10] of char;
      y=array[1..10] of char;
      var
            a,b:x;
            z:y;
```

الف. a:=b و b:=z مجاز است ب. a:=b و b:=z غیر مجاز است.
ج. b:=a و b:=z غیر مجاز است. د. a:=b مجاز و b:=z غیر مجازا ست.

۴۴- تعریف زیر را در زبان C برای پشته در نظر بگیرید. اگر انواع x,y از نظر ساختاری هم ارز باشند کدام گزینه صحیح است. (نیمسال اول ۸۹-۹۰)

```
Struct stack
{
      int top;
      int data[100];
}x,y;
```

الف. x.top=y.top و x.data[i]=y.data[i] برای تمامی iها بین 0 و topstack-1
ب. x.top=y.top و x.data[i]!=y.data[i] برای تمامی iها بین 0 و topstack-1
ج. x.top!=y.top و x.data[i]=y.data[i] برای تمامی iها بین 0 و topstack-1
د. x.top!=y.top و x.data[i]!=y.data[i] برای تمامی iها بین 0 و topstack-1

۴۵- در تعریف ساختار زیر اشاره به کدام ویژگی در نرم افزار دارد. (نیمسال اول ۸۹-۹۰)

```
Type s(max:integer) is
     record
          r:integer;
          c:integer rang 0..max;
     end record
x:s(200);
```

الف. هم ارزی ساختاری ب. انواع پارامتری ج. هم ارزی نوع د. هم ارزی نام

۴۶- در زبانی مثل لیسپ حافظه هرم شامل چه نوع اطلاعاتی می‌باشد. (نیمسال اول ۸۹-۹۰)

الف. عناصر لیست پیوندی ب. پشته برای ارزیابی توابع جزیی

ج. روالهای سیستم د. روالهای I/O

۴۷- کدام مورد زیر فعالیتهای مربوط به انتقال پارامترها را کامل می‌کند و محتویات پارامترهای واقعی را در پارامترهای مجازی کپی می‌کند. (نیمسال اول ۸۹-۹۰)

الف. prologue ب. epiloge

ج. زنجیره اشاره گر ایستا د. زنجیره اشاره گر پویا

۴۸- کدام گزینه صحیح است؟ (نیمسال دوم ۸۹-۹۰)

الف. در پیاده سازی طول اجزای یک ساختمان داده ثابت که اجزای همگنی دارد ، برای نمایش حافظه پیوندی و کل اجزاء یک توصیف کننده لازم است .

ب. در پیاده سازی طول اجزای یک ساختمان داده ثابت که اجزای همگنی دارد ، برای نمایش حافظه ترتیبی و هر جزء یک توصیف کننده لازم است .

ج. در نمایش حافظه پیوندی عمل انتخاب عنصر تصادفی یا selection امکان پذیر است .

د. در نمایش حافظه پیوندی عمل انتخاب عنصر ترتیبی امکان پذیر است .

۴۹- در مورد مدیریت حافظه ایستا کدام مورد صحیح نیست؟ (نیمسال دوم ۸۹-۹۰)

الف. این تخصیص در زمان ترجمه صورت می گیرد و در طول اجرا ثابت است .

ب. مشکل ترین شکل تخصیص حافظه است .

ج. از آنجا که زمان و حافظه ای برای مدیریت حافظه در زمان اجرا صرف نمی شود ، کارا است .

د. در این تخصیص بازیابی و استفاده مجدد مطرح نیست.

۵۰- در تعریف ذیل گزینه صحیح کدام است؟(نیمسال دوم ۸۹-۹۰)

```
Type vect of  Rec1 : array[1..20] of  Real
vect of  Rec2 : array[1..20] of  Real
Var A1 , A2 : vect of  Rec1
A3 : vect of  Rec2
```

الف. A_1با A_2 با A_3هم ارزی نام دارند .
ب. A_1 با A_3 هم ارزی نامو A_1و A_3 با A_2 هم ارزی نام دارند .
ج. A_1و A_2 با A_3هم ساختاری نام دارند .
د. A_1 با A_3 هم ارزی ساختاری و A_1با A_2هم ارزی نام دارند .

۵۱- در آرایه A با ابعاد 7*5 و اندازه هر عنصر 4B که در آدرس α ذخیره شده است ، محل A[3,4] در صورت ذخیره سطری برابر است با ؟ (آرایه به زبان پاسکال تعریف شده است.)(نیمسال دوم ۸۹-۹۰)

الف. $7+\alpha$ ب.$68+\alpha$ ج. 7α د. $25+\alpha$

۵۲- قطعه برنامه زیر در زبان C++ موجب چه چیزی می شود؟(نیمسال دوم ۸۹-۹۰)

```
Int *p , *q ;
P=new(int);
q=new(int);
p=q;
```

الف. ارجاع معلق ب. تخصیص حافظه
ج. آزاد سازی حافظه د. زباله

۵۳-قطعه کد مقابل، نشان دهنده کدام مسئله مدیریت حافظه می باشد؟(نیمسال اول ۹۱-۹۰)

```
int *i,*j;
i:=new(int);
j=i;
Free(j);
```

الف. زباله ب. آرگومان ضمنی ج. ارجاع معلق د. اثر جانبی

۵۴- در صورتیکه آرایه زیر به صورت سطری ذخیره و از آدرس ۱۰۰ حافظه شروع شده باشد، کدامیک از موارد زیر آدرس عنصر A[3,0] می باشد؟(نیمسال اول ۹۱-۹۰)

```
A:array[1..3,-2..2] of integer;
```

الف. ۱۲۰ ب. ۱۲۴ ج. ۱۱۴ د. ۱۲۶

۵۵-کدامیک از موارد زیر در مورد شی داده سابقه فعالیت زیربرنامه صحیح است؟(نیمسال اول ۹۱-۹۰)
الف. اندازه و ساختار شی داده سابقه فعالیت در زمان اجرا تعیین می شود.
ب. سابقه فعالیت زیربرنامه در زمان فراخوانی زیربرنامه ایجاد و تا انتهای اجرای زیربرنامه وجود دارد.
ج. شی داده سابقه فعالیت زیربرنامه از دو قسمت تشکیل می شود: بخش پویای سگمنت کد، بخش ایستای رکورد فعالیت.
د. ساختار و مقادیر بخش پویای رکورد فعالیت از فعالیت زیربرنامه برای تمام سوابق فعالیت زیربرنامه یکسان است.

۵۶-عملیات آماده سازی قبل از اجرای زیربرنامه مانند تنظیم رکورد فعالیت، انتقال پارامترها و ... توسط چه قسمتی انجام می شود؟(نیمسال اول ۹۱-۹۰)

الف. برنامه نویس ب. بار کننده(Loader) ج. مترجم د. زبان برنامه نویسی

۵۷–قطعه کد پاسکال زیر مفروض است. کدامیک از موارد زیر صحیح است؟(نیمسال اول ۹۱–۹۰)

الف. به دلیل عدم هم ارزی ساختار، برنامه با خطا مواجه می شود

ب. به دلیل هم ارزی نام در پاسکال، برنامه بدون هیچ خطایی اجرا می شود

ج. به دلیل هم ارزی ساختار در پاسکال، برنامه بدون هیچ خطایی اجرا می شود

د. فراخوانی تابع sub به دلیل عدم هم ارزی نام، با خطا مواجه می شود.

```
Type
          v1:integer;
          v2:integer;
Var      z:v2;
Procedure   sub(A:v1);
Begin
…..
End;
Begin
           Sub(z);
End;
```

۵۸–کدامیک از زبان های برنامه نویسی زیر در عملیات کنترل نوع، از هم ارزی ساختار استفاده می کنند؟(نیمسال اول ۹۱–۹۰)

الف. Ada ب. C,C++

ج. COBOL,FORTRAN د. PL/I , C++

۶-۱۶- پاسخنامه سوالات تستی فصل ششم

سوال	الف	ب	ج	د
۱				*
۲		*		
۳	*			
۴			*	
۵			*	
۶				*
۷	*			
۸			*	
۹				*
۱۰				*
۱۱			*	
۱۲	*			
۱۳			*	
۱۴	*			
۱۵			*	
۱۶		*		
۱۷		*		
۱۸		*		
۱۹			*	
۲۰			*	
۲۱			*	
۲۲		*		
۲۳		*		
۲۴				*
۲۵		*		
۲۶				*
۲۷		*		
۲۸		*		

سوال	الف	ب	ج	د
۲۹			*	
۳۰	*			
۳۱				*
۳۲		*		
۳۳	*			
۳۴				*
۳۵	*			
۳۶		*		
۳۷		*		
۳۸			*	
۳۹			*	
۴۰		*		
۴۱				*
۴۲				*
۴۳				*
۴۴	*			
۴۵		*		
۴۶	*			
۴۷	*			
۴۸				*
۴۹		*		
۵۰			*	
۵۱		*		
۵۲				*
۵۳			*	
۵۴		*		
۵۵		*		
۵۶			*	
۵۷				*
۵۸			*	

سوالات تشریحی

۱- آرایه‌های چند بعدی در زبان‌های برنامه سازی چگونه پیاده سازی می شوند.محاسبه فرمول دسترسی به یک عنصر خاص در یک آرایه دو بعدی را محاسبه کنید. (نیمسال اول ۸۵-۸۶)

۲- فرمول دسترسی تصادفی به عناصر در آرایه دو بعدی را به روش سطری بدست آورید؟(نیمسال اول ۸۶-۸۷)

۳- صفات اصلی مشخص کننده ساختمان داده را شرح دهید؟ (نیمسال دوم ۸۵-۸۶)

۴- برای بردار A[Lb1…ub1,Lb2…ub2,Lb3…ub3] فرمول دسترسی تصادفی به عناصر را به روش سطری بدست آورید؟(نیمسال دوم ۸۶-۸۷)

۵- برای آرایه a[1...3,-1...1]نمایش حافظه آنرا رسم کنید؟(نیمسال دوم ۸۷-۸۸)

۶- برای بردار A[Lb1….ub2,Lb2….ub2,Lb3……ub3] فرمول دسترسی تصادفی به عناصر را به روش ستونی بدست آورید؟(نیمسال اول ۸۷-۸۸)

۷- رکورد متغیر در زبان پاسکال را به همراه یک مثال شرح دهید؟(نیمسال اول ۸۸-۸۹)

۸- اگر تابع زیر در زبانی مثل C نوشته شود، اطلاعات موجود در سگمنت کد و رکورد فعالیت آن را با توجه به متغیرها و ثابتهای محلی آن مشخص کنید؟(نیمسال دوم ۸۸-۸۹)

```
float func1(int x,float y,char c){
   const int a=20;
   float b;
   char c;
   int w;
sub  دستورات اجرایی زیر برنامه
return (نتیجه تابع)
}
```

۹-با تعریف ساختمان داده زیر، آدرس محل داده ای array[20].grade[3] را محاسبه کنید. (با فرض آدرس پایه αو نوع صحیح ۴ بایتی و نوع اعشاری۶ بایتی) (در زبان C اندیس آرایه از صفر شروع می شود) (نیمسال دوم ۸۸-۸۹)

```
struct student {
            int number;
            float grade[10];
            }array[100];
```

۱۰- نمایش حافظه رکوردی با طول متغیری به صورت زیر چگونه است. نمایش حافظه آن را ترسیم نمایید(نیمسال اول ۸۹-۹۰)

```
Type emp=(r,p,g);
var
      employee:record
      id:integer;
      year:integer;
      age:integer;
case payclass:emp of
      R: (m:real;
           S:integer;
          O:real;
      P: (m:real;
           O:real);
      G: (h:real;
           reg:integer;)
end;
```

۱۱- ساختمان داده زیر را در C در نظر بگیرید . array[12].Numfld[3] را بدست آورید. (آدرس پایه k و نوع صحیح ۴ بایت و نوع اعشار ۶ بایت و شروع اندیس آرایه صفر است) (۱ نمره) (نیمسال دوم ۸۹-۹۰)

```
Struct student
{
       Int number;
       Float numfld[10];
}array[100];
```

فصل هشتم:

کنترل ترتیب اجرا

آنچه در این فصل خواهید آموخت:

- کنترل ترتیب در عبارات محاسباتی
- ارزیابی نمایش درختی عبارات
- کنترل ترتیب بین دستورات
  - دستور goto
  - دستور مرکب
  - دستور شرطی
  - دستور تکرار
- برنامه‌های بنیادی
- کنترل ترتیب عبارات غیر محاسباتی
- سوالات تستی و تشریحی

۸-۱- مقدمه

منظور از کنترل ترتیب، کنترل ترتیب اجرای عملیات (عملیات اولیه و عملیات تعریف شده توسط کاربر) و منظور از کنترل داده، کنترل انتقال داده‌ها بین زیر برنامه‌ها و برنامه‌ها می باشد.

ساختارهای کنترل ترتیب :

در حالت معمولی دستورات یک برنامه پشت سرهم اجرا می‌شوند اما در مواردی ترتیب اجرا بنا به دلایلی نظیر دستورات شرطی یا حلقه‌ها عوض می‌شوند. ساختارهای کنترل ترتیب به ۴ دسته تقسیم می‌شوند:

- ساختارهایی که در عبارات (محاسباتی) استفاده می‌شوند مانند قواعد مربوط به تقدم عملگرها و پرانتزها.
- کنترل ترتیب بین دستورات مثل جملات ترکیبی، جملات شرطی، حلقه‌ها.
- کنترل ترتیب در عبارات غیر محاسباتی مانند برنامه نویسی اعلانی که در پرولوگ استفاده می‌شود.
- کنترل ترتیب در زیر برنامه‌ها مانند فراخوانی زیر برنامه‌ها که کنترل برنامه را از نقطه ای به نقطه ای دیگر انتقال می‌دهند (فصل ۹)

به یاد داشته باشید که بعضی از زبان‌ها مثل APL و لیسپ فاقد کنترل دستورات هستند و فقط شامل عبارات اند. از یک جنبه دیگر ساختارهای کنترل ترتیب به دو دسته تقسیم می شوند.

صریح[۱]: ساختار کنترل ترتیب صریح آنهایی هستند که توسط برنامه نویس تعیین می‌شوند تا ساختارهای ضمنی تعریف شده توسط زبان را عوض کند مانند استفاده از پرانتز در عبارات ریاضی یا استفاده از دستورات go to.

ضمنی[۲]: اگر کنترل ترتیب توسط زبان برنامه نویسی تعیین شود به آن کنترل ترتیب ضمنی گفته می‌شود مانند تقدم عملگر * نسبت به +

۸-۲- کنترل ترتیب در عبارات محاسباتی

برای کنترل ترتیب در عبارات محاسباتی از دو روش نمایش برای عبارات محاسباتی استفاده می شود.

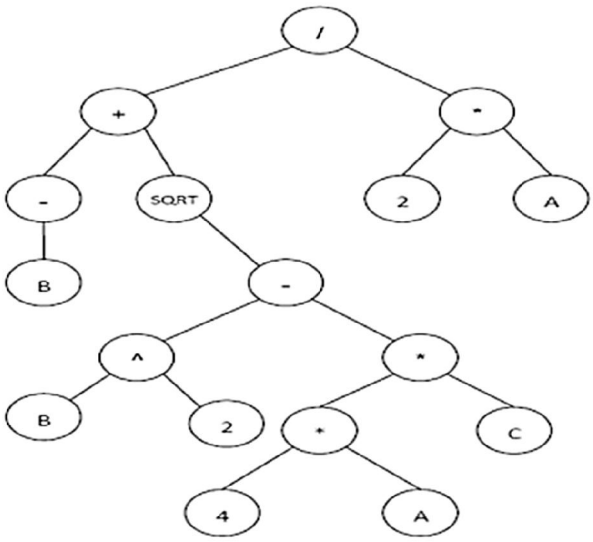
- نمایش درختی
- روش هایprefix,infix,postfix

نمایش درختی:

جهت کنترل ترتیب در عبارات ریاضی می‌توان از روش نمایش درختی استفاده کرد. در این روش ریشه درخت معرف عملیات اصلی، برگ ها معرف داده‌ها وگره‌های بین ریشه و برگها عملیات میانی را نشان می‌دهند. به عنوان مثال فرمول محاسبه ریشه معادله درجه دوم در فرترن به صورت زیر است. نمایش درختی این فرمول به شکل زیر می باشد.

$$\frac{-B + SQRT\,(B^2 - 4*A*C)}{2*A}$$

[۱] Explict
[۲] implict



شکل ۸ -۱

به هنگام تولید کد توسط کامپایلر، جهت ارزشیابی عبارت فوق، حداقل ۱۵ دستور (با در نظر گرفتن جذر و شمارش ارجاع به داده) نیاز است. حال بحث این است که ترتیب اجرای این ۱۵ دستور چگونه است؟ اما این نحوه ارزیابی ایراداتی دارد همانند نقض اولویت‌های ارزیابی (کدام دستور زودتر ارزیابی شود). مثلاَ مشخص نیست که آیا 4*A باید قبل از B^2 ارزیابی شود یا بعد از آن. همچنین مشخص نیست که آیا دو ارجاع به B می‌تواند در یک ارجاع ترکیب شود یا خیر؟ بنابر این در کنترل برنامه ابهاماتی وجود دارد. برای رفع این ایرادات از روش‌های نشانه گذاری خاصی استفاده می‌شود.

روشهای نشانه گذاری prefix,infix,postfix:

الف- پیشوندی(Prefix-Polish): در این روش عملگرها قبل از عملوندهایشان قرار می گیرند. برای مثال عبارت (a+b)*c در فرم پیشوندی به صورت *+abc نمایش داده می شود. نوع دیگری از این روش وجود دارد که به نام Cambridge Polish معروف است که در آن عملگر به همراه عملوندهایش توسط پرانتز احاطه می گردند که در زبان LISP مرسوم است. عبارت (a+b)*c در فرم Cambridge Polish به صورت (*(+ab)c) نمایش داده می شود.

ب- میانوندی(Infix): در این روش عملگرهای دودویی در بین عملوندهایشان قرار می گیرند و برای برهم زدن ترتیب ارزشیابی بر اساس تقدم از پرانتزگذاری استفاده می شود. مانند (a+b)*c. این روش در عمگرهای ۳ تایی نامناسب می باشد به عنوان مثال: exp1?exp2:exp3

ج- پسوندی(Postfix-Reverse Polish): در این روش عملگرها بعد از عملوند هایشان قرار می گیرند. برای مثال عبارت (a+b)*c در فرم پسوندی به صورت ab+c* نمایش داده می شود.

مزایای استفاده از روش پیشوندی و پسوندی نسبت به میانوندی عبارت است از:

- عدم نیاز به پرانتز گذاری
- عدم نیاز به قرار دادن اولویت یا شرکت پذیری
- ارزشیابی انها به راحتی توسط یک الگوریتم کارا انجام می شود و لذا برای استفاده در زبان‌های برنامه سازی مناسب است.
- قابل اعمال برای هر نوع عملگری با هر تعداد عملوند می باشند.
- دارای نحوی به مانند فراخوانی توابع می باشند.

ارزیابی عبارات Prefix, Postfix:

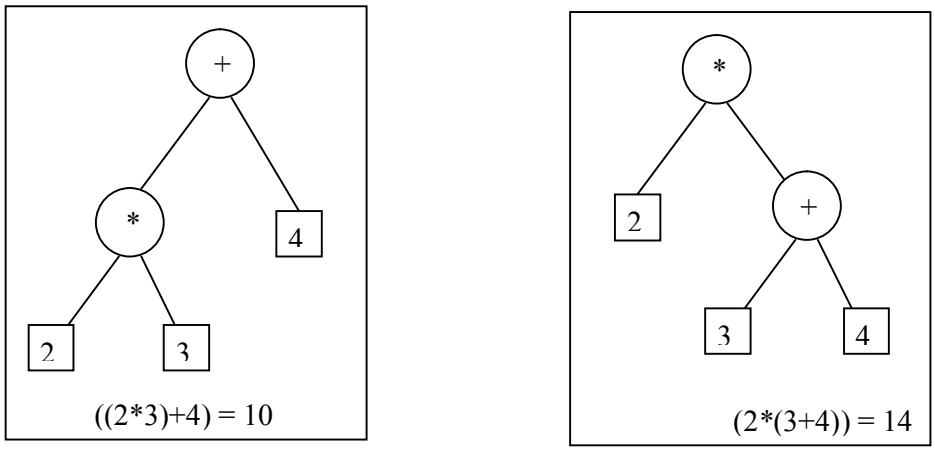
برای ارزیابی عبارات Prefix و Postfix، در هر دو روش عملوندها در پشته Push می شوند و با رسیدن به عملگر، از بالای پشته دو عمل Pop انجام می شود با این تفاوت که در روش Postfix عبارات از چپ به راست و بالای پشته دومین عملوند محسوب می شود در حالیکه در روش Prefix عبارات از راست به چپ و بالای پشته اولین عملوند محسوب می شود. علاوه بر صرفه جویی در پرانتزها، نشانه گذاری Prefix، ارزش‌های خاصی در زبان های برنامه نویسی دارد:

- فراخوانی تابع به صورت نشانه گذاری Prefix انجام می‌شود. F(x,y,z)
- نشانه گذاری Prefix می‌تواند برای نمایش عملیاتی با هر تعدادی از عملوندها به کار گرفته شود مانند روش Combridge polish در لیسپ.
- نشانه گذاری Prefix را به راحتی می‌توانبه طور مکانیکی رمز گشایی کرد.

ارزیابی عبارات Infix :

اگرچه نشانه گذاری Infix متداول است ولی معایبی نیز دارد:

- چون نشانه گذاری Infix فقط برای عملگرهای دودویی مناسب است زبان نمی‌تواند فقط از این نشانه گذاری استفاده کند بلکه حتماً باید Infix با یکی از دو روش Prefix یا Postfix ترکیب شود که این ترکیب خود مشکلاتی به همراه خواهد آورد.
- در محاسبه بعضی عبارات ریاضی ممکن است ابهام بوجود آید مثلاً در عبارت A*B+C معلوم نیست که کدام یک از عمل‌ها باید ابتدا انجام شود و ترتیب اجرای هریک نتایج متفاوتی خواهد داشت. به شکل زیر توجه کنید:



شکل ۸ -۲

در اغلب زبانها برای اینکه ترتیب اجرای عملگرها ابهام بر انگیز نباشد و نیاز به پرانتزهای متعدد در نماد Infix نیز وجود نداشته باشد از قواعد تقدم عملگر استفاده می‌شود این ترتیب عملگرها در همه زبان ها یکسان نیست و هر زبان ترتیب خاص خود را دارد ولی عموماً در مورد عملگرهای ریاضی ابتدا ضرب و تقسیم و سپس جمع و تفریق انجام می‌شود. همچنین در اکثر زبان ها، عملگرهایی که در یک سطح اولویت هستند از چپ به راست اجرا می‌شوند. مثلاً ترتیب a+b+c به صورت ((a+b)+c) می‌باشد ولی در مورد توان و انتساب عموماً این ترتیب از راست به چپ اجرا می‌شود.

a=b=c → (a=(b=c)) یا a ↑ b ↑ c ↑(b↑c))

جدول تقدم عملگر مربوط به زبانهای C,Ada در زیر آورده شده است:

سطح تقدم	عملگرها	عملیات
بالاترین تقدم	Not , abs , **	توان ، قدرمطلق ، نقیض
	/ , mod , rem	ضرب و تقسیم
	+ -	جمع و تفریق یکانی
	+ - &	جمع و تفریق دودویی
	= , <= , < , > , >=	رابطه ای
پایین ترین تقدم	And , or , xor	عملیات بولین

جدول ۸ - ۱ سلسله مراتب عملیات در ادا.

تقدم	عملگرها	اسامی عملگرها
۱۷	Tokens,a[k],f()	لیترالها، اندیس ، فراخوانی تابع
	., ->	انتخاب
۱۶	++, --	افزایش و کاهش پسوندی
۱۵ *	++, --	افزایش و کاهش پیشوندی
	~ , _, sizeof	عملگاهای یکانی ، حافظه
	* , & , !	نقیض منطقی ، آدرس دهی غیر مبهم
۱۴	(type name)	تبدیل ضمنی
۱۳	* , / , %	عملگرهای ضرب
۱۲	+ , -	عملگرهای جمع
۱۱	<<,>>	شیفت
۱۰	< , > , <= , >=	رابطه ای
۹	== , !=	تساوی
۸	&	And بیتی
۷	^	Xor بیتی
۶	|	Or بیتی
۵	&&	And منطقی
۴	||	Or منطقی
۳*	? :	شرطی
۲*	=,+=,-=,*=,/= , %= , <<= , >>= , &= , ^= , |=	انتساب
۱		ارزیابی ترتیبی

جدول ۸ - ۲سطوح تقدم عملگرها در C

*: عملگرهایی که از راست به چپ ارزیابی می‌شوند.

اگر زبانها حاوی عملگرهایی باشند که در ریاضیات کلاسیک موجود نباشند تقدمها با شکست مواجه خواهند شد.Forth,APL,C و اسمالتاک نمونه‌هایی از زبان هایی هستند که عملگرهای توسعه یافته دارند. بنابراین توضیح مختصری در مورد هر یک ارائه می کنیم:

APL :

زبان APL زبانی است که برای کارکردن روی آرایه‌ها ساخته شده و در این زبان دستورات و عبارات از راست به چپ ارزیابی می‌شوند به علت دقت عبارات APL، این زبان کوچک است ولی در عین حال برنامه نویس می‌تواند برنامه‌های یک خطی پیچیده ای را بنویسد.

:Forth

زبان Forth برای کامپیوترهای بی درنگ[۱] طراحی شده است. از آنجا که در آن زمان (۱۹۶۰) حافظه‌ها گران بودند زبان فورث را به گونه ای طراحی کردند که کوچک باشد و درعین حال ترجمه آن ساده بوده و کارایی اجرای آن هم خوب باشد. در این زبان از نماد Postfix برای عبارات استفاده می‌شود. این زبان تفسیری است و دو پشته دارد یکی جهت برگشت زیر برنامه‌ها و دیگری پشته ارزیابی عبارات.

مطالعه زبانهای C و اسمالتاک به عهده دانشجوی محترم می باشد

روش های ارزیابی عبارات محاسباتی در زمان اجرا:

به علت مشکل بودن رمزگشایی عبارات infix بهتراست فرم infix ابتدا به شکل اجرایی تبدیل شود تادر حین اجرا به سادگی رمز گشایی شود برای این کار سه روش مختلف وجود دارد.

- **دنباله ای از کد ماشین:** اگر عبارات را به یک سری کد ماشین واقعی تبدیل کنیم، ترتیب این دستورات، ترتیب عملیات را مشخص می‌سازند. این روش سرعت اجرای بالایی دارد و زبان هایی مانند C,C++,Pascal از این روش استفاده می کنند
- **ساختارهای درختی:** دراین روش در مرحله اول عبارات به کمک مفسر نرم افزاری به شکل درختی در می‌آیند سپس در مرحله دوم یعنی اجرا، پیمایش درخت انجام می‌شود این تکنیک در زبان LISP استفاده می‌شود.
- **فرم Prefix یا Postfix:** دوروش Prefix و Postfix طبق الگوریتم‌های قبلی اجرا می شوند. در برخی از کامپیوترهای واقعی که بر مبنای پشته کار می‌کنند کد واقعی ماشین به شکل Postfix است. نمایش Prefix نیز در اسنوبال ۴ استفاده می‌شود.

۸-۳- ارزیابی نمایش درختی عبارات

رویه اصلی ترجمه برای عبارات که به صورت درختی نمایش داده می‌شوند، آسان است ولی در مرحله دوم که درخت به دنباله ای از عملیات اجرایی تبدیل می‌شود با مشکلاتی روبرو هستیم که بعضی از آنها عبارتند از :

- قواعد ارزیابی یکنواخت
- اثرات جانبی
- شرایط خطا
- عبارات بولین مدار کوتاه

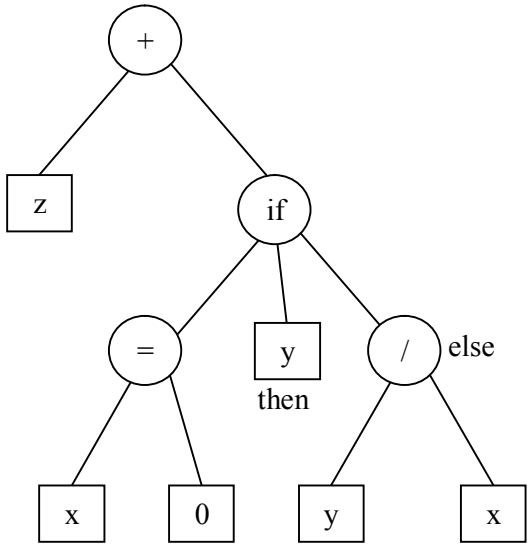
---

[۱] realtime

قواعد ارزیابی یکنواخت:

برای ارزیابی عبارات، دو تکنیک ارزیابی عجول[۱] و تنبل[۲] وجود دارد. در ارزیابی عجول، همواره در ابتدا عملوندها ارزیابی می‌شوند و سپس عملیات را بر روی عملوندهای ارزیابی شده، اجرا می‌کنیم. این قاعده را عجول می‌نامیم زیرا همیشه اول عملوندها ارزیابی می‌شوند. در این روش ترتیب دقیق ارزیابی عملیات مهم نیست مثلاً برای محاسبه (a+b)*(c-a) ممکن است اول (a+b) یا اول (c-a) محاسبه شود ولی این روش همیشه امکان پذیر نخواهد بود. مثلاً در عبارت شرطی زیر به زبان C که مانند if عمل می‌کند اگر y صفر نباشد x/y محاسبه می‌شود. حال با روش عجول اگر حتی y=0 باشد عبارت x/y تقسیم و محاسبه می‌گردد که خطای تقسیم بر صفر رخ خواهد داد.

w:= z+(x=0 ? y: y/x) هم ارز w:= z+(if(x=0) then y else y/x)



شکل ۸ -۳

برای رفع مشکل بالا (خطای تقسیم بر صفر) از قاعده ارزیابی تنبل استفاده می‌کنیم. در روش ارزیابی تنبل، عملوندها قبل از اجرای عملیات ارزیابی نمی‌شوند، بلکه عملوندها ارزیابی نشده، ارسال شده وعملیات تصمیم می‌گیرد که ارزیابی لازم است یا خیر. این روش همواره درست عمل می‌کند ولی پیاده سازی آن مشکل است. زبانهای محاوره ای مانند لیسپ و پرولوگ از تکنیک ارزیابی تنبل استفاده می‌کنند در حالیکه زبانهای محاسباتی مانند C و فرترن علاقه ای به استفاده از آن ندارند. اصطلاحات عجول و تنبل معادل دو تکنیک ارسال پارامتر به زیر برنامه‌ها یعنی انتقال پارامتر با مقدار و با نام است.

عجول ← با مقدار تنبل ← با نام

مثالی از روش تنبل این است که در عبارات ترکیبی and، اگر یکی از عملوندها false بود عملوند دیگر محاسبه نمی‌شود و جواب کلی false اعلام می‌گردد.

اثرات جانبی[۳]:

یکی دیگر از مسائل، استفاده از عملیاتی است که اثرات جانبی بر عبارات دارند. مثلاً درعبارت a* fun(x)+a مطلوب آن است که a فقط یک بار ارزیابی شده و در دو جای محاسبات استفاده گردد. همچنین ارزیابی fun(x) قبل یا بعد از دستیابی به a فرق نداشته باشد. ولی اگر fun(x) روی a اثر جانبی داشته باشد و آن را تغییر دهد آن گاه ترتیب دقیق ارزیابی بسیار مهم

---

[۱] eager
[۲] lazy
[۳] Side effect

خواهد بود مثلاً اگر مقدار اولیه a=1 و خروجی fun(x) برابر ۳ باشد و این تابع مقدار a را به ۲ تغییر دهد خروجی‌های ممکن برای عبارت a+fun(x)*a به صورت زیر خواهد بود:

الف-محاسبه به ترتیب از چپ به راست: ۱*۳+۲=۵

ب- a فقط یک بار ارزیابی شود: ۱*۳+۱=۴

ج-تابع fun(x) قبل از ارزیابی a فراخوانی گردد: ۲*۳+۲=۸

```
a =1
y=a*fun(x)+a
int fun( int k){
.
.
a ++;
.
.
Return 3;}
```

شرایط خطا:

شرایط خطا ممکن است در عملیات اولیه پدید آید مانند سرریز، تقسیم برصفر. در چنین مواردی برنامه نویس ممکن است مجبور باشد ترتیب ارزیابی را دقیقاً کنترل کند مثلاً جهت جلوگیری از سرریز ممکن است عبارت a+b-c به صورت a-c+b محاسبه گردد.

عبارات بولین مدار کوتاه[1]:

در ارزیابی مدار کوتاه عبارت، نتیجه عبارت بدون ارزیابی تمامی عملوندها و یا عملگرهای آن تعیین می‌شود به عنوان مثال عبارت (۱-b/۱۳)*(a*۱۳) را در نظر بگیرید. اگر در این عبارت a=0 باشد مقدار این عبارت به (۱-b/۱۳) بستگی ندارد یا از آن مستقل است یعنی این بخش از عبارت در مقدار عبارت تاثیری ندارد لذا نه تنها نیازی به ارزیابی این بخش نیست عملگر دوم نیز لازم نیست ایجاد شود اما تشخیص این وضعیت در حین اجرا ساده نیست به همین دلیل اغلب از ارزیابی مدار کوتاه استفاده نمی‌شود.

به مثال های دیگری در این زمینه توجه کنید:

مثال۱) `if((A==0) or (B/A>C) then` …

بسیاری از زبان‌ها هر دو عملوند را ارزیابی میکنند و اگر A= = 0 به خاطر عبارت B/A، خطای تقسیم بر صفر رخ خواهد داد ولی برخی از زبان‌های دیگر مانند C هنگامی که عبارت سمت چپ (A= = 0)درست باشد دیگر عبارت سمت راست را محاسبه نکرده و جواب را True در نظر می‌گیرند یعنی مقدار عملوند سمت چپ ممکن است بقیه عبارت بولین را مدار کوتاه کند. درزبان ادا با دو عملیات and then و or else تکنیک ارزیابی مدار کوتاه فراهم شده است.

```
If (A= = 0) orelse (B/A>C) then…
```

در دستور فوق در زبان Ada اگر A= = 0 باشد خطا رخ نداده و عبارت به صورت True ارزیابی می گردد.

[1] short-circuit Boolean expression

مثال۲)
```
while (I<UB) and (V[I]>0) do
```
اگر این دو عبارت را همزمان ارزشیابی کنید و I خارج از حد آرایه باشد خطا رخ خواهد داد. برای بر طرف کردن این مشکل در صورتی که عبارت اولی false باشد ارزیابی عبارت دوم تأ ثیری در نتیجه نخواهد داشت پس از مفهوم مدار کوتاه استفاده می کنیم و اصلاً عبارت سمت راست را ارزشیابی نمی‌کنیم تا بخواهد خطایی رخ دهد.

```
While (I<UB) andelse (V [I]>0) do …
```

انتساب:

هدف از دستور انتساب این است که مقدار راست عبارت (مقدار شی داده) را به مقدار چپ آن (محل حافظه) نسبت دهد. شکل-های مختلف انتساب در زبان های گوناگون در جدول زیر آورده شده است:

A := B	Pascal , Ada
A = B	Fortran , C , PL/I , ML , prolog
A→B	APL
MOVE B TO A	COBOL
(SETQ A B)	LISP

جدول ۸ -۳

در زبان C، انتساب یک عملگر است ولی اغلب انتساب به عنوان یک دستور محسوب می‌شود انتساب در پاسکال مقدار بر نمی‌گرداند ولی در C مقدار بر می‌گرداند. اغلب زبان‌ها فقط یک عملگر انتساب دارند ولی C چندین عملگر انتساب دارد. مفاهیم مختلف عملگر انتساب در زبان C در زیر آورده شده است.

A=B: مقدار راست B را به مقدار چپ A نسبت می‌دهد و مقدار راست A را بر می‌گرداند.

A+=B (A-=B): به اندازه B به A اضافه (کم) می‌کند و مقدار جدید را بر می‌گرداند.

A++ (A--): یک واحد به A اضافه (کم) می‌کند و مقدار راست A را بر می‌گرداند.

A++ (A--): مقدار A را بر می‌گرداند سپس یک واحد به آن اضافه (کم) می‌کند.

۸-۴- کنترل ترتیب دستورات

سه شکل متفاوت جهت کنترل ترتیب جملات دستوری عبارتند از:

- **ترکیب**[۱]: دستورات پشت سر هم و به ترتیب اجرا می‌شود مثل جملات بین begin, end در یک بلوک پاسکال
- **انتخاب**[۲]: در این دستورات دو یا چند مسیر جهت اجرا وجود دارد مانند case,if
- **تکرار**[۳]: دنباله ای از دستورات که به صورت تکراری اجرا می‌شوند مانند while-for

[۱] composition
[۲] alternation
[۳] Iteration

۸-۴-۱- دستور goto

دستور goto در زبان‌های اولیه بیشتر مورد استفاده قرار می‌گرفت اما با ایجاد مفهوم برنامه نویسی ساخت یافته استفاده از آن محدود شده و توصیه می شود تا حد امکان از آن استفاده نشود زیرا برنامه‌هایی که تعداد زیادی دستور goto دارند شبیه spageti (ماکارونی) می باشند و اشکال زدایی این برنامه‌ها به سختی انجام خواهد گرفت. دو نوع دستور goto داریم :

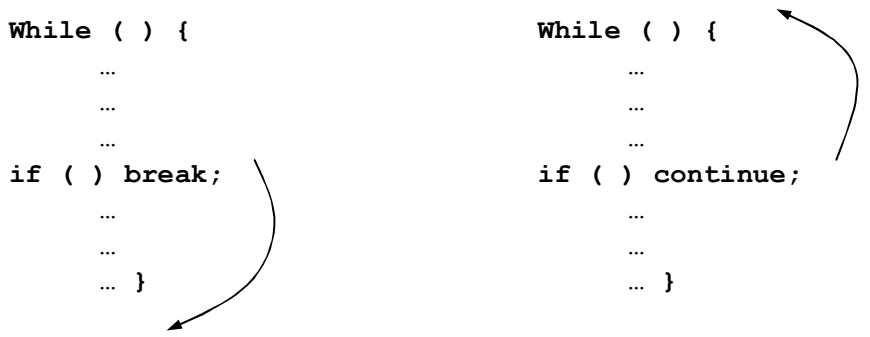
- **goto بدون شرط:** این دستور کنترل را به یک خط خاص انتقال می دهد (دستور بعد از goto به عنوان بخشی از دنباله اجرا نمی‌شود)

```
Goto next;
```

- **goto شرطی:** در صورتی که شرط ذکر شده درست باشد کنترل به دستور با برچسب Next منتقل می‌شود.

```
If (A==0) then goto Next;
```

در زبان‌های ساخت یافته امروزه توصیه اکید آن است که حتی الامکان از این دستور استفاده نشود حتی در برخی از زبان‌های جدید مثل ML دستور goto وجود ندارد. از زمانی که زبان‌ها دارای ساختارهای کنترلی مثل if, while شدند دستور goto به طور کامل از دور خارج شد در بعضی از زبان‌ها مانند فرترن و ML، انتقال کنترل صحیح لازم است زیرا ساختارهای کنترلی مناسب وجود ندارد این کار توسط دو دستور break, continue انجام می‌شود. در برخی از زبان‌ها مانند C, Pascal دو دستور break, continue وجود دارد که break باعث اتمام حلقه‌ها و continue باعث برگشت به اول حلقه می‌گردد.



مزایای دستور goto :

- مستقیماً توسط سخت افزار پشتیبانی شده و اجرای آن بهینه می باشد.
- کاربرد آن در برنامه‌های کوچک ساده است.
- آشنا بودن برنامه نویسان با آن، مخصوصاً برنامه نویسان اسمبلی و زبان‌های قدیمی.
- یک بلاک از برنامه با استفاده از goto می‌تواند چندین هدف را سرویس دهد.

معایب دستور goto :

- مغایرت با ساختار سلسله مراتبی برنامه. طبق اصول برنامه نویسی ساخت یافته، ساختار برنامه باید به صورت درختی باشد ولی استفاده از goto ممکن است این ساختار را به شکل گرافی در آورده و لذا درک برنامه را پیچیده کند.
- مغایرت با اصل نوشتن ساختارهایی که یک نقطه ورود و یک نقطه خروج دارند. استفاده از دستور goto ممکن است باعث شود هر بلوک چند نقطه خروج داشته باشد.

- در برنامه نویسی بهتر است ترتیب اجرایی جملات با ترتیب فیزیکی آن‌ها یکسان باشد. که استفاده از goto این امکان را از بین می‌برد.
- با دستور goto می‌توان کاری کرد که یک قسمت از برنامه به عنوان ادامه چند قسمت دیگر مورد استفاده قرار گیرد و این موضوع درک برنامه‌ها را مشکل می‌سازد.

ویژگی‌های برنامه نویسی ساخت یافته:

- بر طراحی سلسله مراتبی[۱] ساختارهای برنامه با استفاده از شکل‌های کنترلی ساده مثل ترکیب، انتخاب، تکرار تأکید دارد.
- بر متنی از برنامه تأکید دارد که ترتیب فیزیکی دستورات همان ترتیب اجرا باشد.
- بر استفاده از گروههایی با یک هدف تأکید دارد حتی اگر مستلزم کپی کردن دستورات باشد.
- درک، اشکال زدایی، کنترل، تصحیح و نگهداری برنامه‌های ساخت یافته آسان است.

همانطور که قبلاً گفتیم جملات دستوری در برنامه‌های ساخت یافته سه نوع ترکیب، انتخاب، تکرار هستند. یک ویژگی مهم این دستور این است که همهٔ آن‌ها یک نقطه ورود و یک نقطه خروج دارند. در ادامه هریک از موارد مذکور را دقیق تر بررسی خواهیم کرد.

۸-۴-۲- دستورات ترکیب

دنباله ای از دستورات هستندکه در ساختار دیگر دستورات به عنوان یک دستور به حساب می‌آیند مثلاً در پاسکال دستورات مرکب به صورت Begin…End می باشند و در C++,C,java و پرل به صورت {.....} نوشته می‌شود.

۸-۴-۳- دستورات شرطی

متداول ترین این دستورات If, case می‌باشد. با If های تودرتو یا نردبانی می‌توان حالت‌های متعددی را بررسی کرد. برخی ساختارهای Ifهای تودرتو را می‌توان به صورت ساده تری با case نوشت. چند مثال از دستورات شرطی در زبان Ada در زیر آورده شده است:

```
If condition then statement endif(تک شرطی)
If condition then (دو شرطی)
          Statement1
Else
          Statement2
Endif
If condition1 then (چند شرطی)
          Statement1
Elseif condition2 then
          Statement2
          …
Else
          Statement n
endif
```

[۱] herarchical

نمونه ای از دستورات IF تودرتو (نردبانی) در زبان Ada در زیر آورده شده است:

```
If tag=0 then statement0
Else if tag=1 then statement1
  Else if tag=2 then statement2
    Else statement3 End IF
```

با فرض اینکه Tag :0..5 باشد معادل دستور Case عبارت فوق به صورت زیر می باشد:

```
Case tag is
      when 0 => begin
            statement 0
      end;
      when 1 => begin
            statement 1
      end;
      when 2 => begin
            statement 2
      end;
      when others => begin
            statement 3
      end;
end case
```

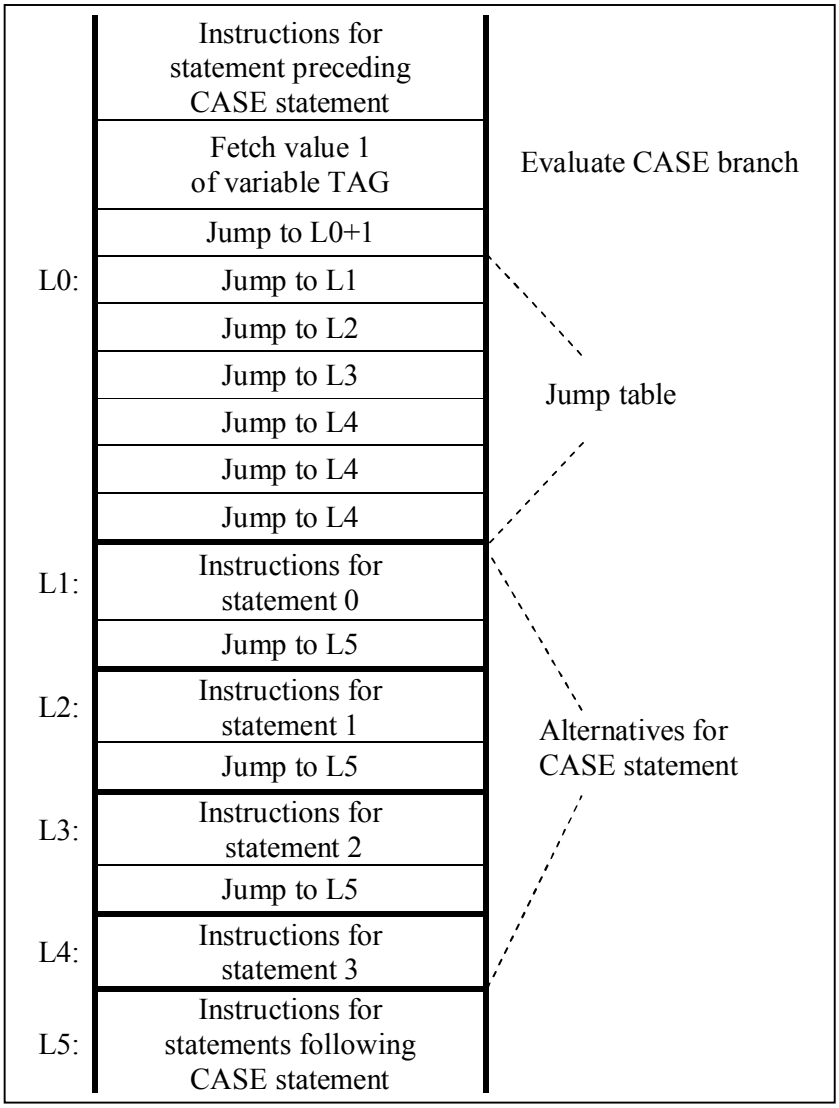
پیاده سازی:

دستورات if با دستورات پرش سخت افزاری پیاده سازی می شوند ولی دستورات case اغلب به کمک جدول پرش[1] پیاده سازی می‌شوند تا بدین ترتیب از تست‌های تکراری یک متغیر جلوگیری شود ولی کارایی اجرا بالا می‌رود. جدول پرش برداری است که به صورت ترتیبی در حافظه ذخیره شده و هر یک از عناصر آن یک دستور پرش غیر شرطی است ابتدا عبارتی که شرط دستور case را پدید آورده ارزیابی شده و نتیجه آن به یک مقدار صحیح تبدیل می‌شود که آفستی را در جدول پرش نشان می‌دهد.

نکته: به طور کلی هزینه ترجمه case بیشتر از If تودرتو است (به دلیل ساختن جدول پرش) ولی هزینه اجرای case به مراتب کمتر از If های تودرتو است. هزینه اجرای دستور case ,$O(1)$ و هزینه اجرای If های تودرتو $O(n)$ می باشد.

پیاده سازی دستور Case مثال قبل به شکل زیر می باشد:

[1] Jumptable



شکل ۸ -۴

نکته: استفاده از ساختار case با توجه به ساخت جدول پرش، نسبت به ساختار if های تودرتو، زمان کامپایل کردن را بیشتر می‌کند ولی از طرف دیگر سرعت اجرای برنامه را افزایش می‌دهد. چرا که در ساختار جدول پرش فقط یک مقایسه لازم است ولی در if های تودرتو شرطهای متعددی باید آزمایش شوند

۸-۴-۴- دستورات تکرار

دستورات تکرار از یک راس (head) و یک بدنه (body) تشکیل یافته اند. راس تعداد دفعاتی که بدنه باید اجرا شود را تعیین می‌کند. ساختارهای راس عبارتند از :

- **تکرار ساده:** که تعداد دفعات تکرار بدنه را به سادگی و با صراحت تعیین می‌کند مانند نمونه زیر در کوبول که با دستور perform انجام می‌گیرد:

```
Perform body k times
```

این دستور body را k بار تکرار می کند.

چند سوال مطرح است:
آیا k تنها یک بار ارزشیابی می شود یا در حین اجرای body مقدار جدیدی خواهد گرفت؟
اگر k صفر یا منفی باشد آیا body یکبار اجرا می‌شود یا اصلا اجرا نمی‌شود؟

- **تکرار تا هنگامی که شرطی برقرار باشد:** مانند حلقه‌های while.

```
While test do body
```

- **تکرار با تغییر یک شمارنده:** مانند حلقه‌های for مثال زیر در الگول:

```
For I=1 step 2 until 30 do body
```

- **تکرار نامتناهی**: اغلب در مواردی استفاده می‌شود که شرط خروج از حلقه پیچیده بوده و نمی‌تواند در راس حلقه بیان شود. مثل نمونه زیر در Ada :

```
Loop
…
      Exit when condition;
…
End loop;
```

و نمونه زیر در پاسکال:

```
While true do begin … end
```

- **تکرار مبتنی بر داده‌ها (بر اساس اندازه ساختمان داده):** گاهی فرمت داده‌ها، شمارنده تکرار را مشخص می‌کند. این حلقه ها، به جای کنترل شمارنده یا عبارت بولی، با تعداد عناصر موجود در ساختمان داده کنترل می شوند. زبان هایی مانند Java,C#,Perl,PHP چنین دستوراتی دارند. دستور foreach در C# بر اساس تعداد عناصر آرایه یا سایر ساختمان داده ها، کنترل می شود. به عنوان مثال دستورات زیر در C# را در نظر بگیرید در این دستورات، اندازه آرایه تعداد دفعات تکرار حلقه را مشخص می‌کند.

```
String StrList={"Ali","Ahmad","Reza"," Javad"}
Foreach (String name in Strlist) {Consol.WriteLine(String)}
```

نکته: حلقه for در زبان C بسیار انعطاف پذیر است بطوریکه با حلقه for در زبان C تمامی حالات فوق را می‌توان پیاده سازی کرد.

```
For ( exp1; exp2; exp3) {body}
```

شمارنده از ۱ تا ۱۰: `For ( i=1; i<=10; i++) {body}`
حلقه نا متناهی: `For (;;) {body}`
شمارنده با شرط خروج: `For ( i=1 ; i<=100 && neof ; i++) {body}`

پیاده سازی دستورات کنترل حلقه به سادگی با دستورات پرش سخت افزاری انجام پذیر است. اگر چه هر ساختار کنترل ترتیب را می‌توان با استفاده از دستورات ساخت یافته بیان کرد ولی استفاده از دستور goto در شرایطی اجتناب نا پذیر است. این شرایط عبارتند از:

الف- خروج چندگانه از حلقه:

```
For i=1 to k do
      If VECT[1]=0 then goto a {a outside the loop }
End;
```

این حلقه، حلقه جستجو نام دارد که در آن برداری از عناصر جستجو می‌شوند تا اولین عنصری پیدا شود که در شرایط خاص صدق می کند. (یا حلقه خاتمه یابد، یا به عنصر مورد نظر برسیم.)

ب- do-while-do (تکرار قسمتی از دستورات): بعضی اوقات مناسب ترین مکان برای تست خروج از حلقه، نه در اول و نه در آخر حلقه، بلکه در وسط حلقه است. به این ساختار گاهی اوقات do-while-do می‌گویند که متاسفانه هیچ زبان متداولی آن را پیاده سازی نکرده است. البته ساختار if (شرط) break; در زبان C و یا دستور exit when در زبان Ada به این ساختار نزدیک هستند.

```
Loop
      Read(x);
      If (x=0) then goto a (a is outside loop)
      Procrss(x);
end loop;
```

در برنامه بالا یک حلقه تکرار وجود دارد که در همه حالات کل بدنه برای همه دفعات به غیر از دفعه آخر که از حلقه خارج می‌شویم اجرا می‌گردد در این حالت هم استفاده از goto اجتناب ناپذیر است.

ج-شرایط استثنایی:

شرایط استثنا که در حین اجرای برنامه پیش می‌آید مانند تقسیم بر صفر، over flow، under flow و .... با استفاده از دستور goto کنترل برنامه به قسمتی منتقل می‌شود که به آن راه انداز استثنا[۱] گویند. وظیفه این قطعه کد، پیگیری اجرای برنامه به همراه پیغام‌های مناسب است.

نکته:

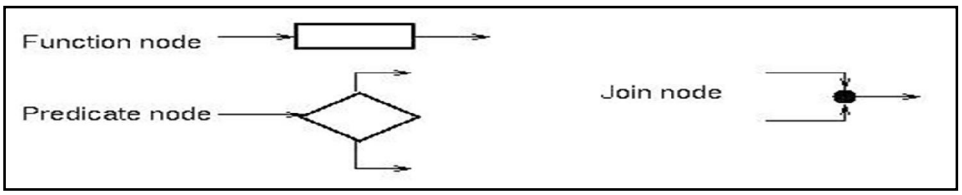
یکی از ابهامات حلقه for آن است که اگر مقدار نهایی حلقه، در بدنه حلقه تغییر کند چه می‌شود؟ در زبان پاسکال مقدار نهایی حلقه فقط یکبار و قبل از ورود به حلقه محاسبه می‌شود. لذا حلقه زیر در زبان پاسکال ۲۰ بار اجرا می شود. ولی در زبان C از آنجائیکه تغییر مقدار نهایی حلقه درون بدنه، روی تعداد تکرار اثر می‌گذارد حلقه زیر در زبان C بی نهایت خواهد بود.

```
n:=20;
For i:=1 to n do
Begin
   n:=n+1;
end
```

[۱] exception handler

۸–۵– برنامه‌های بنیادی[۱]

عموماً فلوچارت‌های برنامه‌ها شامل ۳ دسته گره اصلی هستند: الف– گره تابع ب– گره تصمیم گیری ج– گره اتصال گره تابع، محاسبات موجود در برنامه را نشان می دهد و به صورت مستطیل یا مربع نمایش داده می شود که یک کمان ورودی و یک کمان خروجی دارد. در حقیقت، گره تابع یک دستور انتساب را نشان می‌دهد و منجر به تغییر حالت ماشین مجازی می-شود. گره لوزی، شرط های موجود در برنامه را نشان می دهد که که یک کمان ورودی و دو کمان خروجی دارد. یکی از دو خروجی مربوط به درستی و دیگری مربوط به نادرستی است. گره اتصال، با نقطه نمایش داده می شود که دو کمان را به هم پیوند می دهد تا یک کمان خروجی به وجود آید.



شکل ۸ –۵

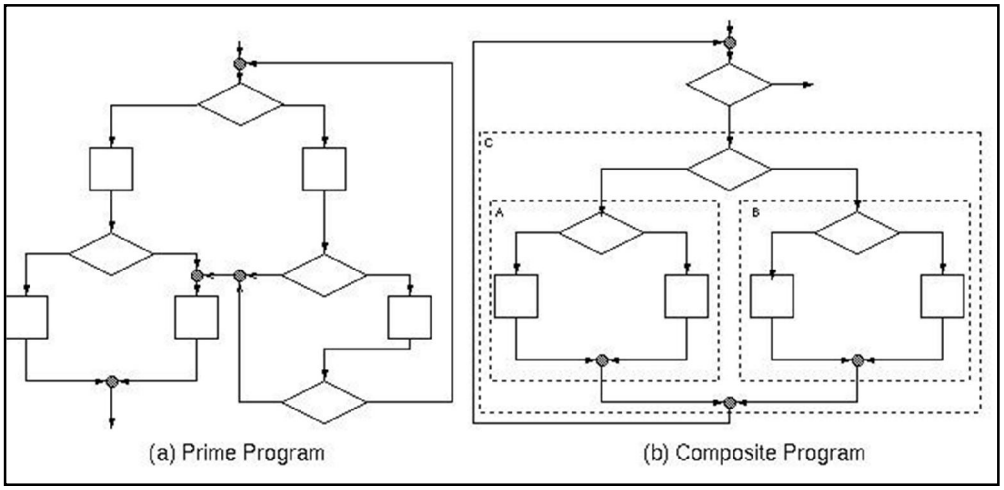
برنامه محض[۲]:

برنامه ای است به صورت فلوچارت که مدل رسمی یک ساختار کنترلی است و شامل ویژگی‌های زیر است :

- فقط یک کمان ورودی دارد.
- فقط یک کمان خروجی دارد.
- مسیری از کمان ورودی به هر کمان و از هر کمان به کمان خروجی وجود دارد.

برنامه بنیادی:

یک برنامه بنیادی، برنامه محضی است که نمی‌تواند به برنامه‌های محض کوچکتری تقسیم شود. برای درک بهتر تعاریف فوق شکل زیر را مشاهده کنید:

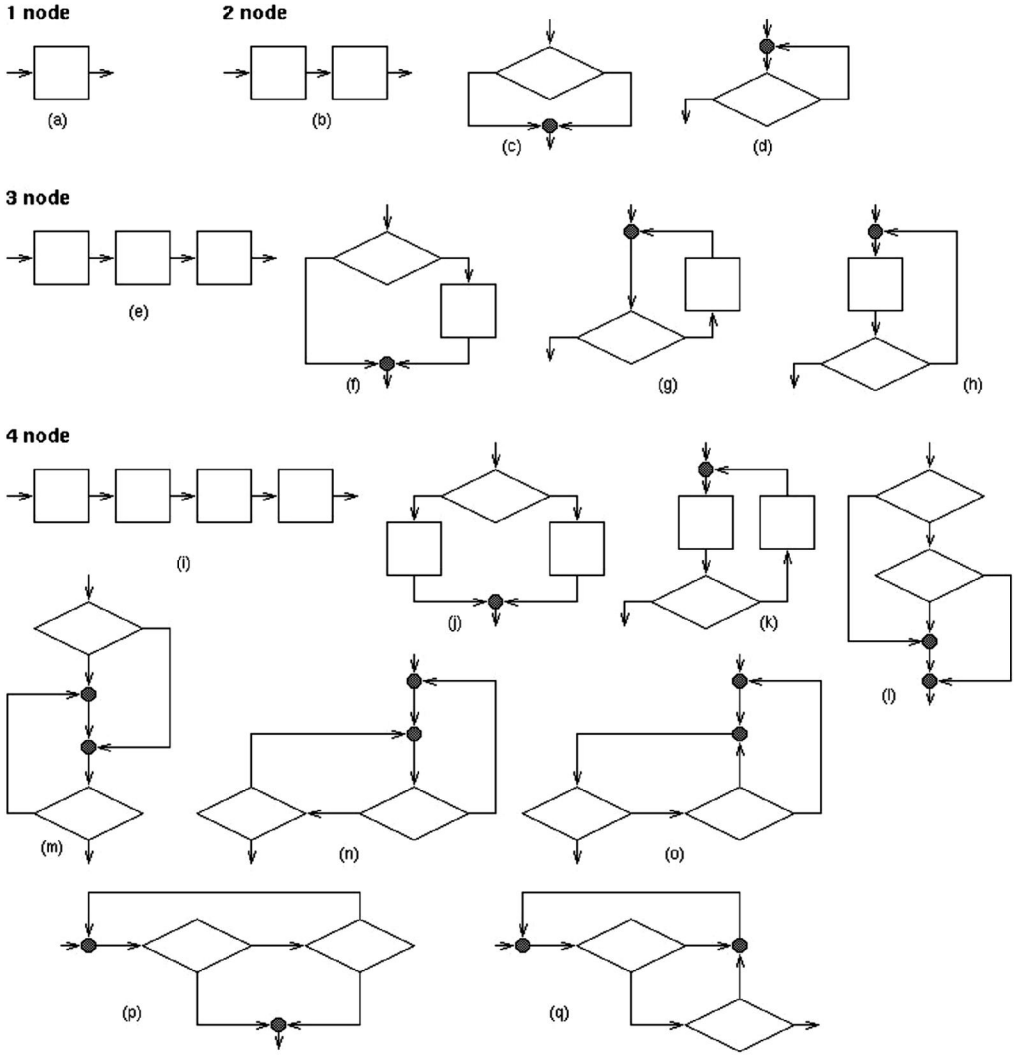


شکل ۸ –۶

[۱] prime
[۲] proper

اگر نتوانیم دو کمان از برنامه محض را برش بزنیم تا برنامه محض را به گراف‌های جداگانه تقسیم کنیم برنامه محض، برنامه بنیادی خواهد بود. شکل(a) یک برنامه بنیادی است در حالیکه شکل (b) بنیادی نیست. قسمت‌های نقطه چین A,B,C زیر برنامه‌های محض هستند. برنامه مرکب، برنامه محضی است که بنیادی نمی‌باشد. شکل (b) مرکب است ولی اگر در همین شکل (b) به جای قسمت‌های نقطه چین، گره تابع قرار دهیم به برنامه بنیادی تبدیل می‌شود.

برنامه‌های بنیادی شمارشی هستند. شکل صفحه بعد تمام برنامه‌های بنیادی تا چهار گره را نشان می‌دهد. همچنین، برنامه‌های بنیادی ساختارهای کنترلی معروف را نشان می دهند. شکل (f) ساختار کنترلیif-then، شکل (g) ساختار کنترلی while، شکل(h) ساختار کنترلی Repeat-until ، شکل (j) ساختار کنترلی IF-THEN-ELSE، شکل (k) ساختار کنترلی do-while-do را نشان می دهد.



شکل ۸ -۷

قضیه ساخت یافته :

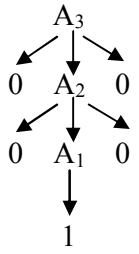
باهوم و جاکوبینی نشان دادند که هر برنامه بنیادی را می‌توان به برنامه ای تبدیل کرد که فقط از دستورات if و while تشکیل شده باشد. نتیجه کار باهوم جاکوبینی نشان داد که لازم نیست از goto پرهیز کرد می‌توان الگوریتم هر برنامه ای را به صورت با goto و یا بدون آن نوشت و سپس با استفاده از قضیه ساخت یافته، آن را به برنامه ساخت یافته تبدیل کرد. برنامه نویسی ساخت یافته مترادف برنامه نویسی خوب نیست بلکه به معنای استفاده از ساختارهای کنترلی است که با تعداد کمی از گره‌ها، برنامه بنیادی هستند. بنابراین، با استفاده از قضیه ساخت یافته همیشه می توان یک برنامه نوشته شده با goto را به یک برنامه بدون goto با استفاده از ساختارهای کنترلی استاندارد نوشت.

۸-۶- کنترل ترتیب در عبارات غیر محاسباتی

یک عملیات حیاتی در زبان‌هایی مثل Snobol4,ML و پرولوگ تطابق الگو می باشد. در این حالت، یک عملیات با تطابق و انتساب مجموعه ای از متغیرها به الگوی از پیش تعیین شده، انجام می‌شود.

مثال- گرامر زیر در الفبای 0,1 رشته‌های متفاوت با طول فرد را تشخیص می‌دهد. فرض کنید بخواهیم توسط این گرامر رشته 00100 شناسایی و تولید شود.

$A \rightarrow 0A0 \mid 1A1 \mid 0 \mid 1$



شکل ۸ -۸

زبان Snobol4 مخصوص تطابق الگو است که کد برنامه این زبان برای شناسایی این رشته به شکل زیر است:

$$\begin{cases} A_1 \quad mathces\ the\ center\ 1 \\ A_2 \quad mathces \quad 0\ A_1\ 0 \\ A_3 \quad mathces \quad 0\ A_2\ 0 \end{cases}$$

:Snobol4

اسنوبال ۴ زبانی است که برای شبیه سازی ویژگی تطابق الگو به وجود آمده است پیاده سازی زبان اسنوبال ۴ مستقل از معماری ماشین می‌باشد به همین دلیل اسنوبال ۴ یکی از اولین زبان هایی بود که اولاً تقریباً در همه ماشینی وجود داشت و ثانیاً معنای آن تقریباً در هر پیاده سازی یکسان بود.

پرولوگ :

بانک اطلاعاتی پرولوگ شامل حقایق[۱] و قواعد[۲] می‌باشد. عبارتی که حاوی یک یا چند متغیر باشد یک تقاضا[۳] نام دارد و رابطه ای ناشناخته را نشان می‌دهد. برای درک این سه مفهوم به مثال توجه کنید:

```
Parent Of(John,Mary)

Parent Of(Susan,Mary)
```

[۱] Facts
[۲] Rules
[۳] Query

```
Parent Of(Bill, John)

Parent Of(ANN,John)
```

با توجه به حقایق موجود در پایگاه دانش فوق، Parent(X, Mary) یک تقاضا می‌باشد و قاعده زیر در این پایگاه دانش برقرار است:

```
Grand Parent of (x,y) : -Parent of (x,y),Parent of (y,z)
```

ویژگی اصلی پرولوگ استفاده از تطابق الگوست تا مشخص شود که آیا تقاضا توسط حقایق موجود در بانک اطلاعاتی قابل حل است یا خیر؟ آیا می‌توان با استفاده از قواعدی در بانک اطلاعاتی (پایگاه دانش) که برروی حقایق یا قواعد دیگر عمل می‌شود حقیقت مورد نظر را بدست آورد یا خیر؟ پرولوگ برای تطابق الگو از اتحاد یا جایگزینی استفاده می‌کند تا تعیین کند که آیا تقاضا شامل جانشین معتبری با حقایق و قواعد موجود در بانک اطلاعاتی می باشد یا خیر.

اتحاد[1](یکسان سازی):

به عمل جایگزینی متغیر به جای متغیرها، تابع به جای متغیر، یا ثابت به جای متغیربه طوریکه گزاره‌های یکسان تولید شود یکسان سازی گفته می‌شود.

مثال:

```
Parent of (x,mary)=parent of (john,y)

θ= {x/john, y/mary}
```
یکسان ساز

```
Unify(p,q)=θ   , subst(θ,p)=subst(θ,q)
```

θ به عنوان یکسان ساز دو جمله p,q خواهد بود که به هر دو گزاره اعمال میشود تا دو گزاره p,q یکسان شوند.

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,Oj)	{x/Oj , y/John}
Knows(John,x)	Knows(y,Mary(y))	{y/John ,x/mary(John)}
Knows(John,x)	Knows(x,Oj)	Fail(چون xنمی‌تواند در یک لحظه دو مقدار بگیرد.)

جدول ۸ -۴

[1] Unification

۸-۷- سوالات فصل هشتم

سوالات تستی

۱- کدامیک از موارد زیر برای کنترل ترتیب سطح دستور استفاده نمی شود؟(نیمسال اول ۸۵-۸۶)

الف. پرش ب. ترکیب ج. انتخاب د.تکرار

۲- در مورد APL کدام گزینه غلط می باشد؟(نیمسال دوم ۸۵-۸۶)

الف.توسط کن آی ورسون در اوایل دهه ۱۹۷۰ طراحی شد. ب. بر پردازش آرایه تاکید دارد.

ج.دستورات را از راست به چپ اجرا می کند. د. زبان کوچکی است.

۳- برای کنترل ترتیب در عبارات غیر محاسباتی از چه روشی استفاده می شود؟(نیمسال اول ۸۶-۸۷)

الف.نمایش درختی ب.انتساب به اشیای داده

ج.تطابق الگو د.هیچکدام

۴- در صورتی که fun(x) مقدار ۳ را برگرداند و مقدار a را به ۲ تغییر دهد ولی مقدار اولیه a برابر یک باشد بسته به ترتیب ارزیابی‌ها برای عبارت a+fun(x)*a چند مقدار متمایز می تواند حاصل شود؟(نیمسال دوم ۸۶-۸۷)

الف.۳ ب.۲ ج.۱ د.۴

۵- ارزیابی میان بر برای کدامیک از عملگرهای منطقی زیر به کار میرود؟(نیمسال دوم ۸۶-۸۷)

الف.AND ب.OR ج.OR, AND د.هیچکدام

۶- برنامه نویسی در کدامیک از زبانهای زیر برنامه نویسی اعلانی است؟(نیمسال دوم ۸۶-۸۷)

الف.C ب.Ada ج.Prolog د.Lisp

۷- کدامیک از موارد زیر از خواص زبان فورث می باشد؟(نیمسال اول ۸۷-۸۸)

مورد اول: برای کامپیوترهای کنترل فرایند بی درنگ ,کاربرد دارد.

مورد دوم: نحو آن postfix محض است.

مورد سوم: ترجمه آن خیلی دشوار می باشد.

الف.اول و دوم ب.اول و سوم ج.دوم و سوم د.هر سه مورد

۸- در صورتی که fun(x) مقدار ۳ را برگرداند و قبل از برگشت مقدار a را به ۲ تغییر دهد با فرض مقدار اولیه a برابر ۱ بسته به ترتیب ارزیابی‌ها برای عبارت a+fun(x)*a چند مقدار متمایز می تواند حاصل شود؟(نیمسال اول ۸۷-۸۸)

الف.۳ ب.۲ ج.۱ د.۴

۹- در زبان C با فرض a=12, b=15 در کدامیک از شرایط زیر قانون مدار کوتاه در ارزیابی صورت می گیرد؟(نیمسال اول ۸۷-۸۸)

شرط اول: if ((a>b)||(b<5)){...}

شرط دوم: if ((a>b)&&(b.5)){...}

شرط سوم: if((a<b)||(b>5)){...}

الف.اول و دوم ب.اول و سوم ج.دوم و سوم د.هر سه شرط

۱۰– در کدامیک از زبانهای زیر دستور go to برای کنترل ترتیب ضمنی وجود ندارد؟(نیمسال اول ۸۷–۸۸)

الف.C++ ب.Pascal ج.C د.ML

۱۱–گزاره درست را انتخاب کنید. (نیمسال اول ۸۷–۸۸)

الف.هر برنامه prime می تواند به برنامه ای تبدیل شود که فقط از دستورات while و if استفاده کند.

ب.نمی توان هر برنامه ای را با استفاده از قضیه ساختیافته به برنامه ساختیافته تبدیل کرد.

ج.می توان فقط هر برنامه ای بدون go to را با استفاده از قضیه ساخت یافته به برنامه ساختیافته تبدیل کرد.

د.الف و ب صحیح است.

۱۲– بانک اطلاعات زبان Prolog شامل کدامیمک از موارد زیر است؟(نیمسال اول ۸۷–۸۸)

الف.حقایق و سوالات ب.حقایق و قواعد

ج.قواعد و سوالات د.قواعد و استنتاج

۱۳– کدامیک از مواردذیل از خواص زبان فورث است؟(نیمسال دوم ۸۷–۸۸)

مورد اول: برای کامپیوترهای کنترل فرایند بی درنگ کاربرد دارد.

مورد دوم: نحو آن از postfix محض است.

مورد سوم: ترجمه آن خیلی دشوار است.

الف.اول و دوم ب.اول و سوم ج.دوم و سوم د.هر سه مورد

۱۴– در زبان C با فرض a=12,b=15 در کدامیک از شرطهای زیر قانون مدار کوتاه در ارزیابی صورت میگیرد؟(نیمسال دوم ۸۷–۸۸)

شرط اول:if ((b<20)||(a<5))(…)

شرط دوم:if((a<b)&&(b>5))(…)

شرط سوم:if((a<b)||(b>5))(…)

الف.اول و دوم ب.اول وسوم ج.دوم و سوم د.هر سه شرط

۱۵– جدول پرش برای پیاده سازی کدامیک از ساختارهای زیر به کار میرود؟(نیمسال دوم ۸۷–۸۸)

الف. Record ب. Function ج.Case د.For

۱۶– خروجی برنامه زیر در زبان C کدام است؟(نیمسال اول ۸۸–۸۹)

```
Main ()
{
      Int *p,*q, I, j:
      Int **qq;
      I=1;j=2;printf("I=%d;j=%d;\n"I,j);
      P=&I; q=&j;
      *p=*q; printf ("I=%d;j=%d;\n",I,j);
      Qq=&p;
      *qq=7; printf ("I=%d;j=%d;\n",I,j);
}
```

الف. i=1; j=1 i=2; j=2 i=7; j=2	ب. i=1; j=1 i=1; j=2 i=7; j=7	ج. i=1; j=2 i=2; j=2 i=7; j=7	د. i=1; j=2 i=2; j=2 i=7; j=2

۱۷– مقصود از هم ارزی ساختاری برای دو نوع داده چیست؟(نیمسال اول ۸۸–۸۹)

الف.دو نوع داده هم ارز هستند اگر اشیای داده آنها عناصر خارجی یکسان داشته باشند.

ب.دو نوع داده هم ارز هستند اگر اشیا داده آنها عناصر داخلی یکسانی داشته باشند.

ج.دو نوع داده هم ارز هستند اگر صرفاً نامهای یکسانی و مشابه به یکدیگر داشته باشند.

د.دو نوع داده هم ارز هستند اگر نامهای یکسان و اشیای خارجی متفاوت داشته باشند.

۱۸– در کدامیک از حالت‌های زیر ارزیابی نمایش درختی عبارات به صورت عجولانه صورت می گیرد؟(نیمسال اول ۸۸–۸۹)

الف. برای هر گره عملیاتی ابتدا عملگرها وبعد عملوندها ارزیابی می شوند.

ب. برای هر گره عملیاتی عملوندها و عملیات با هم ارزیابی می شوند.

ج. برای هر گره عملیاتی ابتدا عملوندها و سپس عملیات ارزیابی می شوند.

د. برای هر گره عملیاتی عملوندها قبل از اجرای عملیات ارزیابی نمی شوند.

۱۹– کدامیک از گزینه‌های زیر صحیح می باشد؟(نیمسال اول ۸۸–۸۹)

الف. امروزه از دستور go to زیاد استفاده می شود.

ب.استفاده از دستور go to فقط به صورت شرطی امکان پذیر است.

ج.استفاده از دستور go to فقط به صورت غیر شرطی امکان پذیر است.

د. استفاده از دستورgo to برنامه‌ها را از حالت ساختیافته خارج می کند.

۲۰– کدامیک از ساختارهای کنترلی زیر با فرض وجود برچسب‌های ساده در زبانهای برنامه سازی، بطور ضمنی توسط معماری سخت افزاری پشتیبانی می شود؟ (نیمسال دوم ۸۸–۸۹)

الف. case ب.If ج. go to د.While

۲۱– با توجه به مجموعه کد زیر کدام یک از مسائل ترتیب ارزیابی در هنگام تولید کد قابل اعمال است. (نیمسال اول ۸۹–۹۰)

```
Int x,y,z;
z=(y=0?x:x/y);
y=y+x;
```

الف. عجول
ب. تنبل
ج. هم عجول هم تنبل
د. عجول–تنبل–اثرات جانبی

۲۲- برای انتخاب از بین ۱۰ دستور برای اجرا ، یکبار از دستورات تودرتوی (If-Then-Else) استفاده کردیم و یکبار از دستور Case با توجه به اینکه کامپایلر مورد استفاده ، Case را با استفاده از روش پرش پیاده سازی کرده است. هزینه اجرا وترجمه دو روش را مقایسه کنید؟(مهندسی کامپیوتر-۷۵)

الف. هزینه اجرای Case بیشتر و هزینه ترجمه آن کمتر است.

ب. هزینه اجرا وترجمه Case ازIf-Then-Else بیشتر است.

ج. هزینه اجرا وترجمه Case ازIf-Then-Else کمتر است.

د. هزینه اجرای Case کمتر وهزینه ترجمه آن بیشتر است.

۲۳- برای ساختن یک درخت تصمیم گیری (Decision Tree) به منظور انتخاب یک دستور (Statement) از میان n دستور، کدام یک از ساختارهای برنامه سازی زیر می‌تواند منجر به کارایی اجرا به صورت O(1) شوند ؟ (فرض کنید شرط انتخاب یک دستور ، برابر یک مقدار ثابت یک میباشد)(مهندسی کامپیوتر- ۸۳)

الف. ساختن درخت با دستور If-Then-Else ب. بدست آوردن O(1) امکان ندارد.

ج. ساختار درخت با دستور Case یا Switch د. ساختار دستور با استفاده از If تودرتو در Ada

۲۴- حلقه زیر را درنظر بگیرید. اگر فقط Step و مقدار نهایی حلقه هر بار ارزشیابی می‌شوند، خروجی این کد چیست؟ (مهندسی کامپیوتر -۷۲)

```
k:=1;
l:=5;
for i:=k to 2×l step k
      print I;
l:=l-1;
k:=k+1;
end for
```

الف)
۱ ۳ ۷

ب)
۱ ۲ ۳

ج)
۱ ۳ ۵

د)
۱ ۳ ۶

۲۵- اگر از لیبل های ساده در زبان استفاده شود ، معماری سخت افزار می تواند : (نیمسال دوم ۸۹-۹۰)

الف. بطور ضمنی ساختار کنترلی case را پشتیبانی کند .

ب. بطور ضمنی ساختار کنترلی case و بطور صریح ساختار کنترلی go to را پشتیبانی کند .

ج. بطور ضمنی ساختار If و While را و مستقیما ساختار کنترلی go to را پشتیبانی می کند .

د. بطور ضمنی ساختار کنترلی go to را پشتیبانی کند .

۲۶- تطبیق الگو در کدام زبان به عنوان یک عملیات حیاتی محسوب نمی شود؟ (نیمسال دوم ۸۹-۹۰)

الف. prolog ب. اسنوبال۴ ج. ML د. Ada

۸-۸- پاسخنامه سوالات تستی فصل هشتم

سوال	الف	ب	ج	د
۱	*			
۲	*			
۳			*	
۴	*			
۵			*	
۶			*	
۷	*			
۸	*			
۹			*	
۱۰				*
۱۱	*			
۱۲		*		
۱۳	*			
۱۴		*		
۱۵			*	
۱۶				*
۱۷		*		
۱۸			*	
۱۹				*
۲۰			*	
۲۱		*		
۲۲				*
۲۳			*	
۲۴				*
۲۵				*
۲۶				*

سوالات تشریحی

۱- برای دستور Case زیر جدول پرش را رسم نمائید؟(نیمسال اول ۸۵-۸۶)

```
Case tag is
When 0=> begin
      Statement t0;
      End;
When 1=>begin
      Statement t1;
      End;
When 2=> begin
      Statement t2;
      End;
When others =>begin
      Statement t3;
      End;
End case;
```

۲- با توجه به مفاهیم ساختارهای کنترلی ترتیب اجرا و نمایش حافظه به ازای قطعه برنامه زیر چگونه است؟(نیمسال اول ۸۶-۸۷)

```
Read (k, I);
If k>=0 then
  While I <=10 do
    I=i+1;
    Write (I)
End while;
Else
  Read (n);
Case n of
'A ': write ('one');
'B ': write ('two');
End case
End if
```

۳- ساختار case زیر را به روش جدو.ل پرش پیاده سازی کنید؟(نیمسال دوم ۸۶-۸۷)

```
Case tag is
 When 0>=begin
Statement t0;
End;
When 1>= begin
Statement t1;
End;
When 2>= begin
      Statement t2;
End;
When others >=begin
Statement t3;
End;
End case;
```

۴- برنامه پریم را تعریف کرده و تمامی برنامه‌های Prime با سه گره را رسم کنید؟(نیمسال دوم ۸۷-۸۸)

۵- برای زبان Prolog مفاهیم زیررا به همراه مثال بطور کامل شرح دهید؟(نیمسال دوم ۸۷-۸۸)

الف.حقایق(Fact) ب.اتحاد(Unification)

ج.قواعد(Rule)

۶- عبارت بولین مدار کوتاه (SHORT-CIRCUIT) را برای عملگرهای OR,AND به همراه مثال شرح دهید؟(نیمسال اول ۸۸-۸۹)

۷- برنامه غیرساخت یافته در شبه زبان C++ را به کمک قضیه ساخت یافته اصلاح کنید.(نیمسال اول ۸۹-۹۰)

```
Cin>>x;
if (x==10)
      x=x+1;
label: if (x==100)
      x=x-1;
if (x==50)
      goto label;
```

۸- نمایش حافظه برای قطعه برنامه ذیل را طبق ساختارهای کنترل ترتیب اجرا ، رسم نمایید . (۱/۵ نمره) (نیمسال دوم ۸۹-۹۰)

```
Read(k,I)
      If k>=0 then
            While i<=10 do
                  I=i+1;
                  Write(I);
            End while;
      Else
            Read(n);
            Case n of
                  A" : write('Alpha');
                  B" : write('Beta');
            End case;
      End if;
```

فصل نهم:

کنترل ترتیب زیربرنامه

آنچه در این فصل خواهید آموخت:

- زیربرنامه‌های فراخوانی – بازگشت
- زیربرنامه‌های بازگشتی
- محیط ارجاع
  - ارجاع سراسری
  - ارجاع محلی
  - ارجاع غیر محلی
  - ارجاع از پیش تعریف شده
- اعلان پیشرو در پاسکال
- نام مستعار
- حوزه ایستا و پویا
- ساختار بلوکی
- محیط ارجاع برای داده‌های محلی
  - نگهداری
  - حذف
- پارامترها
  - پارامترهای واقعی و مجازی و تناظر بین آنها
  - روش‌های انتقال پارامتر
    - فراخوانی با مقدار
    - فراخوانی با ارجاع
    - فراخوانی با نام
    - فراخوانی با نتیجه
    - فراخوانی با مقدار–نتیجه
    - فراخوانی با مقدار ثابت
  - پیاده سازی انتقال پارامتر
  - زیر برنامه به عنوان پارامتر
- محیط‌های مشترک صریح
- پیاده سازی حوزه ایستا و پویا
- اعلان‌ها در بلوک‌های محلی
- سوالات تستی و تشریحی

۹-۱- مقدمه

در کنترل ترتیب زیر برنامه‌ها، با جزئیات فرایند فراخوانی یک زیر برنامه و برگشت از آن سر و کار داریم که در اکثر زبان های برنامه نویسی به ترتیب با دستورهای فراخوانی (Call) و بازگشت (Return) انجام می‌شود. ساده ترین نوع کنترل زیر برنامه دارای ساختاری به نام ساختار فراخوانی- بازگشت[۱] می باشد. این ساختار کنترلی توسط قانون کپی[۲] بیان می‌شود. در قانون کپی، اثر دستور فراخوانی زیر برنامه مثل این است که قبل از اجرا، یک کپی از زیر برنامه ای که فراخوانی شده است در نقطه فراخوانی قرار داده شود. اما برای استفاده از قانون کپی، پنج فرض از سوی طراحان زبان در نظر گرفته می‌شود که عبارتند از:

- زیر برنامه‌ها نباید بازگشتی باشند چون هر جایگزینی زیر برنامه با وجود اینکه یک دستور فراخوانی (Call) را حذف می‌کند ولی فراخوانی جدیدی به همان زیر برنامه معرفی می‌کند که برای آن نیز مجددا یک جایگزینی دیگر لازم است و..... این روند تا بی نهایت ادامه خواهد داشت. بنابراین برای زیر برنامه های بازگشتی نمی توان از قانون کپی استفاده کرد.
- زیر برنامه ها باید دستور فراخوانی صریح داشته باشند اما در زیربرنامه های مخصوص پردازش استثنا، هیچ دستور فراخوانی صریحی وجود ندارد. بنابراین برای زیربرنامه های پردازش استثنا نمی توان از قانون کپی استفاده کرد.
- زیر برنامه‌ها باید در هر مرتبه فراخوانی به طور کامل اجرا شوند اما یکسری زیربرنامه های خاصی به نام همروال‌ها[۳] وجود دارند که در آنها ممکن است زیر برنامه به طور کامل اجرا نشود. بنابراین برای زیربرنامه های همروال نمی توان از قانون کپی استفاده کرد.
- باید به محض اجرای دستور فراخوانی (Call) کنترل اجرا به زیر برنامه فراخوانی شده داده شود و پس از اجرای زیر برنامه کنترل اجرا به نقطه فراخوانی برگردد. اما یکسری زیر برنامه های خاصی به نام زیر برنامه‌های زمان بندی شده وجود دارند که فقط در زمان برنامه ریزی شده برای آنها، اجرا خواهند شد و نه بلافاصله. بنابراین برای زیر برنامه های زمانبندی شده نیز نمی توان از قانون کپی استفاده کرد.
- در هر زمان فقط باید یک زیر برنامه کنترل اجرا را در دست داشته باشد، کنترل اجرا از برنامه فراخوان به زیربرنامه فراخوانی شده می رود و به برنامه فراخوان بر می گردد. اگر اجرا را در نقطه ای متوقف کنیم یک برنامه در حال اجراست، بقیه یا هنوز فراخوانی نشده اند یا اجرای آنها کامل شده است. اما زیر برنامه‌هایی که به عنوان task مورد استفاده قرار می‌گیرند ممکن است به طور همزمان اجرا شوند به طوریکه چندین زیر برنامه در آن واحد در حال اجرا باشند یعنی اگر یکی از زیربرنامه‌ها متوقف شد ممکن است چندین زیر برنامه دیگر در حال اجرا باشند. بنابراین برای زیر برنامه های task نمی توان از قانون کپی استفاده کرد.

۹-۲- زیربرنامه‌های فراخوانی - بازگشت

همانطور که قبلاً گفته شد بین تعریف زیربرنامه و فعالیت زیربرنامه، تفاوت وجود دارد. تعریف آن چیزی است که در برنامه می-نویسیم و به یک قالب ترجمه می‌شود اما فعالیت زیربرنامه در هر زمانی که زیر برنامه فراخوانی می شود تولید می شود و برای تولید آن از تعریف انجام شده، الگو گرفته می شود. هر فعالیت زیر برنامه شامل دو بخش است: یکی سگمنت کد که حاوی کد اجرایی و ثابت‌ها می باشد و دیگری رکورد فعالیت که شامل پارامترها، داده‌های محلی، نتایج و سایر داده‌هاست. بخش سگمنت کد در حین اجرا تغییر نمی‌کند و از این رو مقادیری که در طول اجرا تغییر نمی کنند در آن ذخیره می شوند ولی در رکورد

---

[۱] call-return
[۲] copy rule
[۳] coroutines

فعالیت مقادیری ذخیره می شود که در طول اجراهای متفاوت زیربرنامه ممکن است تغییر کنند. همه رکوردهای فعالیت زیر برنامه از یک سگمنت کد استفاده می‌کنند ولی رکورد فعالیت در هر بار اجرای زیر برنامه ایجاد شده و با تمام شدن زیر برنامه از بین می‌رود.

هنگام صحبت کردن از یک دستور در حال اجرا باید دو اشاره گر (آدرس) داشته باشیم تا بتوان به طور دقیق مشخص کرد که آن دستور در کدامین اجرای (رکورد فعالیت) زیربرنامه در حال اجراست. این دو اشاره گر را با CIP[۱] و CEP[۲] نشان خواهیم داد. CIP اشاره گری است که آدرس دستورالعمل جاری که در مرحله بعدی باید اجرا شود را نگهداری می کند و CEP اشاره گری است که آدرس رکورد فعالیت جاری را نگهداری می کند. دستورالعمل جاری یا توسط سخت افزار و یا توسط مفسر در حال اجراست. این دو اشاره گر در ثبات های پردازنده ذخیره می شوند.

- **اشاره گر دستور فعلی(CIP):** که به دستور داخل سگمنت کد که قرار است اجرا شود اشاره می‌کند و مترجم دستوری را که CIP به آن اشاره می‌کند بازیابی کرده و آن را اجرا می‌کند.
- **اشاره گر محیط فعلی(CEP):**اشاره گری است که به رکورد فعالیت فعلی (مربوط به قطعه کدی که در حال اجرا است) اشاره می‌کند بنابراین چون تمام رکوردهای فعالیت یک زیر برنامه از یک سگمنت کد استفاده می‌کنند دانستن دستور فعلی کافی نیست باید اشاره گر CEP هم باشد تا رکورد فعالیت مورد نظر را مشخص نماید. به عنوان مثال وقتی دستوری درکد، به متغیر x مراجعه می‌کند آن متغیر در رکورد فعالیت وجود دارد، هر رکورد فعالیت آن زیربرنامه، شیء داده ای به نام x دارد که ممکن است محتویاتش با دیگری فرق داشته باشد.

نقطه بازگشت:

شی داده ای که توسط سیستم تعریف می شود و در هر زیر برنامه دارای مقداری است که نشان می دهد به هنگام بازگشت از این زیربرنامه به کدام نقطه و کدام محیط باید برگشت انجام شود. این نقطه بازگشت در رکورد فعالیت زیر برنامه جاری ذخیره می شود و آن را با زوج (ep,ip) نشان می دهند.

پیاده سازی زیر برنامه فراخوانی-بازگشت ساده:

در زمان شروع اجرای برنامه اصلی، اشاره گر CEP به رکورد فعالیت برنامه اصلی اشاره می‌کند و CIP به اولین دستور از سگمنت کد برنامه اصلی اشاره می‌کند وقتی کنترل اجرا به دستور فراخوانی زیر برنامه رسید یک رکورد فعالیت از آن زیر برنامه ایجاد می‌شود و CEP به آن اشاره می‌کند و CIP به اولین دستور سگمنت کد زیر برنامه اشاره خواهد کرد. جهت برگشت صحیح و درست از یک زیر برنامه، مقادیر CEP و CIP را می‌توان در رکورد فعالیت زیر برنامه ای که فراخوانی شده است ذخیره کرد تا در زمان بازگشت از زیر برنامه، اجرا از نقطه بعد از فراخوانی زیر برنامه از سر گرفته شود. شکل زیر نمونه ای از عملیات فوق را برای برنامه اصلی که دو بار زیر برنامه A و یک بار زیر برنامه B را فراخوانی کرده، نشان می‌دهد. زیر برنامه A خودش یکبار زیر برنامه B را صدا می‌زند. بنابراین به طور خلاصه خواهیم داشت:

عملیاتی که در هنگام فراخوانی (Call) زیر برنامه رخ می دهند:

الف- ایجاد رکورد فعالیت جدید در زیربرنامه

ب- ذخیره کردن CEP و CIP جاری به عنوان نقطه بازگشت در رکورد فعالیت ایجاد شده

ج- مقدار دهی آدرس رکورد فعالیت ایجاد شده و ابتدای زیربرنامه فراخوانی شده در داخل CEP و CIP به عنوان مقدار جدید آنها (این کار باعث انتقال کنترل اجرا به ابتدای زیرب رنامه فراخوانی شده و محیط جدید می گردد)

---

[۱] Current Instruction Pointer

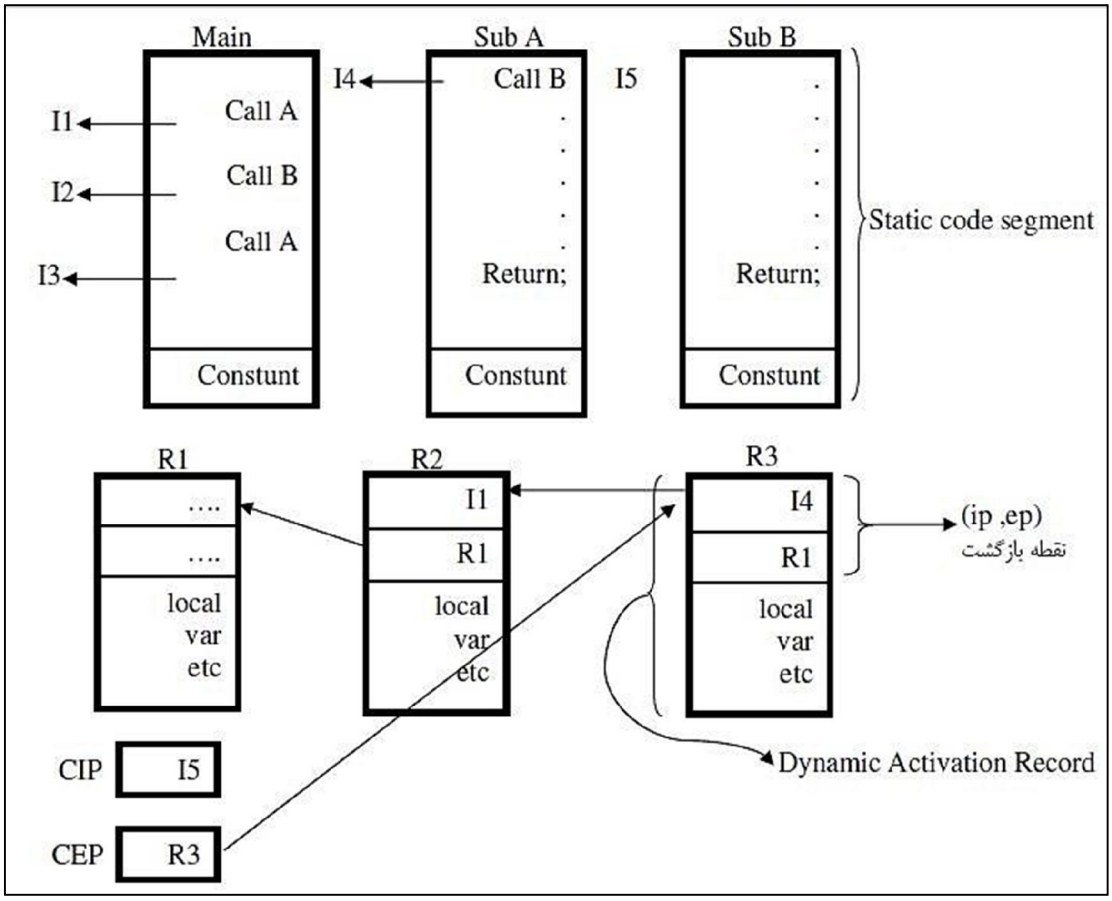
[۲] Current Enviroment Pointer

عملیاتی که در هنگام بازگشت(Return) زیربرنامه رخ می دهند:

الف- کپی کردن مقدار نقطه بازگشت در یک محل موقت

ب- از بین بردن حافظه مربوط به رکورد فعالیت جاری

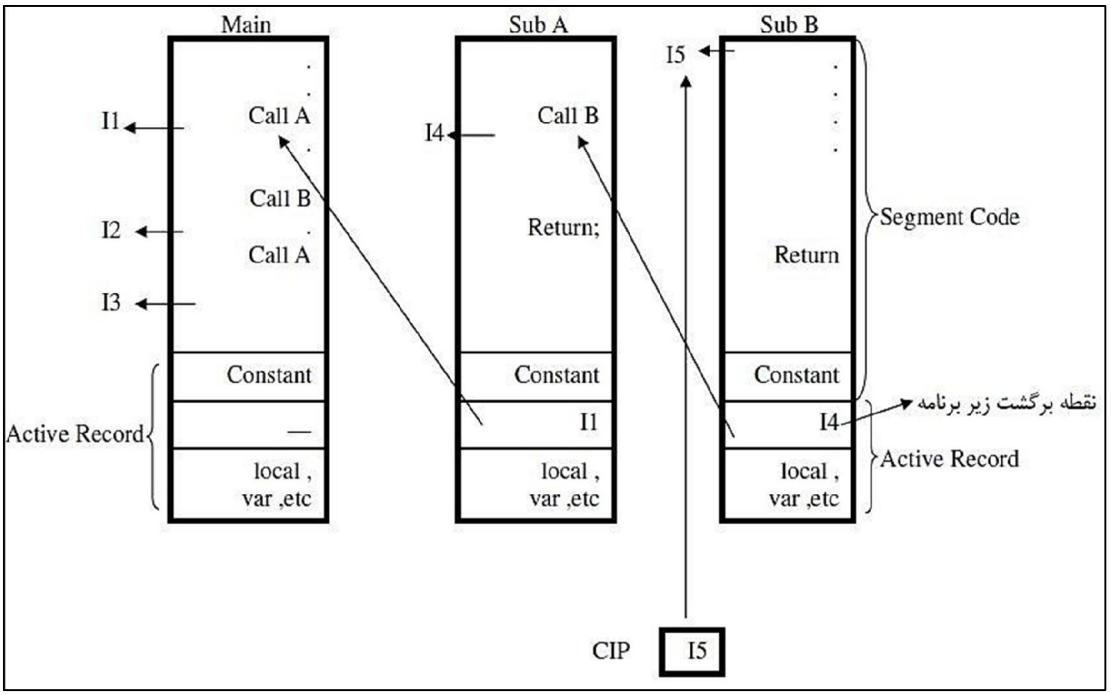
ج- کپی مقادیر نقطه بازگشت در CEP و CIP (این کار منجر به بازگشت کنترل به محل فراخوانی بر می گردد)



شکل ۹ -۱

پیاده سازی دیگر:

در زبان هایی مانند فرترن و کوبول، جهت افزایش سرعت اجرا، با در اختیار داشتن حافظه کافی، از مدل ساده تری استفاده می شود که در آن نیازی به اشاره گر CEP نیست و هر زیربرنامه دارای رکورد فعالیت ثابتی است که به صورت ایستا تخصیص داده می شود و در انتهای سگمنت کد قرار دارد. از آنجائیکه محل آن در انتهای سگمنت کد است لذا نیازی به ذخیره کردن آدرس آن در یک اشاره گر نیست (یعنی در این روش نیازی به اشاره گر CEP نداریم). فقط در مواقع فراخوانی زیر برنامه ها محتوای آنها تغییر می کند و آدرس نقطه بازگشت که فقط یک مقدار است (اشاره گر CIP) در آنها ذخیره می شود. این روش را نمی توان برای پیاده سازی زیر برنامه های بازگشتی استفاده کرد. در این مدل پیاده سازی، اجرای کل برنامه با یک سگمنت کد و یک رکورد فعالیت برای هر زیر برنامه و برنامه اصلی شروع می‌شود. در حین اجرای برنامه، هنگام صدا زدن زیر برنامه دیگر حافظه ای به صورت پویا تخصیص نمی‌یابد بلکه در هر بار فراخوانی از همان رکورد فعالیت، استفاده مجدد می‌شود. شکل زیر نمونه ای از این ساختار را نشان می‌دهد:



شکل ۹ -۲

۹ -۳- زیر برنامه‌های بازگشتی

تکنیک بازگشتی، یکی از مهم ترین ساختارهای کنترل ترتیب در برنامه نویسی است. در زبان لیسپ که در آن لیست یک ساختمان داده اولیه است تکنیک بازگشتی، مکانیزم کنترل مهمی برای تکرار دنباله ای از دستورات است. زیربرنامه ای بازگشتی است که به صورت مستقیم یا غیر مستقیم (از طریق زیر برنامه دیگر) خود را فراخوانی کند. در زبان‌هایی مثل PL/I زیر برنامه‌های بازگشتی با کلمه کلیدی Recursive مشخص می شوند. نمونه ای از بازگشتی مستقیم برای تابع فاکتوریل زیر به زبان C در زیر آمده است:

```
Int fact (int n) {
If(n<=1) ret 1;
else ret fact(n-1)*n ;}
```

اگر زیر برنامه A زیر برنامه B را فراخوانی کند و زیر برنامه B نیز A را فراخوانی کند بازگشتی غیر مستقیم به وجود می‌آید. تنها تفاوت بین فراخوانی بازگشتی با فراخوانی معمولی این است که در فراخوانی بازگشتی، در حین طول عمر اولین رکورد فعالیت، رکورد فعالیت دیگری ایجاد می‌شود اگر رکورد فعالیت دوم هم فراخوانی بازگشتی دیگری را پدید آورد، آنگاه چندین رکورد فعالیت مربوط به یک زیر برنامه به طور هم زمان فعال خواهند بود و این روند می‌تواند به همین صورت تکرار شود.

پیاده سازی:

برای پیاده سازی زیر برنامه‌های بازگشتی، به دلیل امکان وجود چند رکورد فعالیت به طور همزمان، هر دو اشاره گرCEP وCIP باید استفاده شوند. پیاده سازی زیر برنامه های بازگشتی مشابه پیاده سازی زیر برنامه های فراخوانی- بازگشت ساده است با این تفاوت که برای مدیریت رکوردهای فعالیت از پشته مرکزی استفاده می شود. به ازای هر Call، رکورد فعالیت زیر برنامه مربوطه ایجاد شده و آدرس نقاط برگشت در رکورد فعالیت ایجاد شده در پشته ذخیره می شوند (Push) و به ازای هر Return، آدرس نقاط برگشت از رکورد فعالیت موجود در پشته استخراج شده و در اشاره گرهای CEP و CIP کپی می شود و سپس رکورد فعالیت مربوطه از بالای پشته برداشته می شوند (Pop).

در هنگام عملیات کنترل ترتیب، زنجیره ای از رکوردهای فعالیت روی پشته ساخته می شود که اصطلاحاً به آن زنجیره پویا[1] می گویند. زنجیره پویا، زنجیره پیوندی است که ابتدای آن رکورد فعالیت جاری می باشد و به رکورد فعالیت زیر برنامه ای که توسط زیر برنامه جاری فراخوانی شده، اشاره می کند. در حقیقت زنجیره پویا، رکوردهای فعالیت زیر برنامه ها که به طور پویا در زمان اجرا ایجاد شده اند را به یکدیگر پیوند می دهد.

۹-۴- صفات کنترل داده‌ها(محیط ارجاع)

ویژگی‌های کنترل داده‌های یک زبان برنامه سازی، آن بخش‌هایی از زبان است که در نقاط مختلفی از اجرای برنامه به داده‌ها دستیابی دارند، وقتی در حین اجرا به عملیاتی رسیدیم باید داده‌های مورد نیاز آن عملیات آماده باشند، ویژگی‌های کنترل داده‌ها در یک زبان تعیین می کند که داده‌ها چگونه برای عملیات آماده می شوند و نتایج عملیات باید ذخیره و توسط عملیات بعدی بازیابی شوند. برای مثال برای عبارت X:=Y+2*Z وظیفه کنترل داده‌ها تعیین این نکته است که مثلاً در هر اجرا کدام Y استفاده شود چرا که ممکن است Y یک متغیر محلی یا غیر محلی باشد. برای استفاده از یک علموند در یک عملیات (عملگر) دو راه وجود دارد:

- انتقال مستقیم[2]:

برای نمونه 2*Z درعبارت X:=Y+2*Z به صورت مستقیم استفاده شده است و نامی به آن اختصاص داده نشده است. در این موارد زبان از یک حافظه موقتی برای آن استفاده می کند.

- مراجعه به عملوند (شی داده) از طریق نام آن:

مراجعه به شیء داده به صورت غیر مستقیم و از طریق نام آن. برای نمونه در عبارت X:=Y+2*Z نام Z دلالت بر داده ای می کند که باید در عمل ضرب استفاده شود.

اجزایی از برنامه که می توانند دارای نام باشند عبارتند از:

- نام متغییرها
- نام پارامترهای مجازی
- نام زیربرنامه‌ها
- نام برای نوع‌های داده تعریف شده در برنامه
- نام برچسب‌های دستورات
- نام استثناها
- نام عملیات اولیه (+و*و/و..)
- نام ثابت‌های لفظی (لیترال‌ها): مانند 14.25

نام‌ها از یک دیدگاه به دو دسته نام‌های ساده و مرکب تقسیم بندی می شوند:

- **نام ساده:** نام یک متغییر ساده مانند A
- **نام مرکب:** نامی که به یک مؤلفه از یک ساختمان داده منتسب می شود مانند A[2]

[1] Dynamic chain
[2] Direct Transmition

مفهوم وابستگی و محیطهای ارجاع:

وابستگی[۱]:

انقیاد هر نام به یک شی داده یا هر نام به یک زیر برنامه خاص را وابستگی گویند. در آغاز برنامه اصلی در تعریف متغییرها، هر متغییر را به یک شی داده مقید می کنیم و با این کار وابستگی بین شناسه (نام) تعریف شده و شی داده مربوط را ایجاد می کنیم.

محیط ارجاع[۲]:

هر زیر برنامه ای دارای مجموعه ای شناسه است که در طول اجرا از آن‌ها استفاده می‌کند به این مجموعه، محیط ارجاع آن زیر برنامه می گویند. محیط ارجاع زیر برنامه در حین اجرا متغیر نیست. این محیط ارجاع در حین ایجاد رکورد فعالیت زیر برنامه ایجاد و تنظیم می‌گردد و با از بین رفتن رکورد فعالیت زیر برنامه از بین می‌رود. مقادیر موجود در اشیا داده ممکن است تغییر کنند ولی وابستگی نام‌ها به اشیا داده و زیر برنامه‌ها (محیط ارجاع زیر برنامه) تغییر نمی‌کنند. انواع محیط ارجاع عبارتند از:

الف-محیط ارجاع محلی[۳]:

محیط ارجاع محلی یک زیر برنامه، شامل کلیه اسامی است که هنگام ورود به یک زیر برنامه ایجاد شده و داخل زیر برنامه قابل دسترس هستند. مانند اسامی پارامترهای مجازی، متغیرهای محلی و زیر برنامه‌هایی که درون زیر برنامه مورد نظر تعریف شده اند.

ب-محیط ارجاع غیر محلی[۴]:

محیط ارجاع غیر محلی یک زیر برنامه، شامل اسامی است که بر اساس قانون ساختیافتگی بلاکی (اگر یک اسم در یک بلاک یا زیر برنامه تعریف شود، اسم مورد نظر در داخل آن بلاک و بلاک های درونی این بلاک قابل استفاده است) داخل زیر برنامه قابل دسترسی هستند ولی هنگام ورود به زیر برنامه ایجاد نمی‌شوند. نمونه ای از آن متغیرهای محلی static در زبان C می-باشند. اسامی غیر محلی به دو صورت حوزه پویا و حوزه ایستا قابل دسترس هستند. به عنوان مثالی دیگر در زبان پاسکال اگر تابع f در داخل تابع g تعریف شده باشد محیط ارجاعی غیر محلی برای f، محیط ارجاعی محلی برای g می‌باشد.

ج-محیط ارجاع سراسری[۵]:

محیط ارجاع سراسری، شامل کلیه اسامی است که در شروع اجرای برنامه اصلی پدید آمده و درکل برنامه‌ها قابل دسترس هستند مانند متغیرهایی که در اول برنامه پاسکال و C تعریف می‌شوند.

نکته: اسامی سراسری، بخشی از اسامی غیر محلی هستند.

نکته: نام هر زیر برنامه جزء محیط غیر محلی خودش است.

د-محیط ارجاع از پیش تعریف شده[۶]:

محیط ارجاع از پیش تعریف شده، شامل کلیه اسامی است که توسط کامپایلر تعریف شده و تمامی زیر برنامه ها می توانند از آنها استفاده کنند. مانند اسامی عملیات اولیه مانند +و*و/و- و float و int و.....

برنامه زیر نمونه کاملی است که محیط ارجاع‌های مختلف را نشان می‌دهد.

---

[1] association
[2] Refrencing Enviroment
[3] local
[4] Non-local
[5] global
[6] predefined

```
Program main;
      var A,B,C:real;
      procedure sub1(A:real);
            var D:real
            procedure sub2(C:real);
                  var D:real;
            begin
                  …
            end;
      begin
            …
            sub2(B);
            …
      end;
begin
      …
      sub1(A);
      …
end.
```

محیط ارجاع برای sub2	محیط ارجاع برای sub1	محیط ارجاع برای main
محلی: C و D غیرمحلی از sub1: A ، sub2 غیرمحلی از main: A ، sub2، B	محلی: A ، D و sub2 غیرمحلی: B ، C و sub1	محلی(سراسری): A ، B ، C و sub1

مفهوم قابلیت مشاهده[۱]:

اگر یک وابستگی برای یک زیربرنامه بخشی از محیط ارجاع آن زیر برنامه باشد گوییم آن وابستگی در زیربرنامه قابل رویت (مشاهده) است. اگر یک وابستگی وجود داشته باشد ولی قابل رؤیت نباشد در اصطلاح به آن وابستگی، وابستگی پنهان[۲] گفته می شود.

گفته می‌شود شناسه x در زیر برنامه یا برنامه f قابل مشاهده است، اگر x قسمتی از محیط ارجاع f را تشکیل دهد. به عبارت دیگر وابستگی شناسه x به یک شی داده در حین ورود به زیر برنامه f تعیین شده باشد اغلب وقتی وابستگی مخفی است که زیر برنامه ای شناسه ای را که در جای دیگر در حال استفاده است دوباره تعریف کند.

حوزه پویا:

حوزه پویای وابستگی متشکل از مجموعه ای از رکوردهای فعالیت زیر برنامه است که توسط آن زیر برنامه قابل مشاهده است.

عملیات ارجاع :

یک عملیات ارجاع، عملیاتی است که یک شناسه و یک محیط ارجاع را گرفته، شناسه را در محیط پیدا کرده و یک شی داده یا زیر برنامه را بر می گرداند.

نکته: ارجاع به یک شناسه (اسم) می‌تواند محلی، غیر محلی یا سراسری باشد. وقتی ارجاع محلی است که عملیات ارجاع، شناسه را در محیط ارجاع محلی پیدا کند و وقتی ارجاع غیر محلی یا سراسری است که عملیات ارجاع شناسه را در محیط غیر محلی یا سراسری بیابد.

---

[۱] Visibility
[۲] Hidden

۹-۵- اعلان پیشرو[۱] در پاسکال

فراخوانی بازگشتی غیرمستقیم در راهبرد تک گذره کامپایلرها مثل طراحی بسیاری از کامپایلرهای پاسکال، مشکلاتی را به وجود می آورد. فرض کنید A وB دو زیر برنامه باشند و A زیر برنامه B را فراخوانی کند و B نیز زیر برنامه A را فراخوانی کند (بازگشتی غیر مستقیم)، چنانچه تعریف A قبل از B بیاید آنگاه برای فراخوانی B در A لازم است که تعریف A قبل از تعریف B ظاهر شود و جابجایی تعریف‌های B و A نیز مسأله را حل نخواهد کرد. این مشکل در پاسکال با اعلان پیشرو حل خواهد شد.

پس از این اعلان پیشرو برای زیر برنامه A، زیر برنامه B می تواند تعریف شود و بعد از زیر برنامه B، تعریف کامل A ظاهر شود ولی لیست پارامترهایی که در اعلان پیشرو آمده است در تعریف کامل A تکرار نخواهد شد. تعریف پیشرو، اطلاعات کافی را در اختیار کامپایلر قرار می دهد تا بتواند فراخوانی A را در B کامپایل کند.

بنابراین گاهی اوقات لازم است قبل از تعریف یک زیر برنامه، از آن زیر برنامه استفاده کنیم. در این گونه موارد باید به اطلاع کامپایلر برسانید که این زیربر نامه بعداً تعریف خواهد شد، برای این منظور از کلمه کلیدی Forward برای اعلان پیشرو استفاده می‌شود.

Procedure نام زیربرنامه (لیست پارامترها) ; Forward;

Function نام تابع(لیست پارامترها) : نوع نتیجه تابع;Forward;

به عنوان مثال اگر در زیربرنامه proc1 از زیربرنامه proc2 استفاده شود ولی زیربرنامه proc2 بعداً تعریف گردد باید به صورت زیر عمل کنید:

```
Procedure proc2(n:integer);forward;
Procedure proc1(n:integer);
    Begin
       Write('proc1');
       If (n>0) then  proc2(n-1);
    End;
Procedure proc2;
   Begin
       Write ('proc2');
       If (n>0) then proc1(n-1);
   End;
```

نکته جالب اینجاست که در هنگام اعلان پیشرو باید پارامترهای زیربرنامه نیز حضور داشته باشند ولی در هنگام تعریف زیربرنامه دیگر پارامترها ذکر نمی‌گردد.

عیب اعلان پیشرو این است که چون لیست پارامترها نباید در تعریف کامل زیر برنامه ظاهر شود می تواند مولد خطا باشد. یک راه حل برای این عیب این است که در کنار تعریف زیر برنامه، توضیحاتی ارائه شود که لیست پارامترها را مشخص کند. استفاده از ساختار Forward در پاسکال، ناهنجاری عجیبی را در پاسکال به وجود می آورد که در زیر آن را بررسی می کنیم :

[۱] forwarddeclaration

```
Program anomaly;
      procedure S; {1}
      begin {of S}
            …
      end; {of S}
      procedure T;
            {missing procedure S; forward; here}
            procedure U;
            begin {of U}
                  S; {2}
            end; {of U}
            procedure S; {3}
            begin {of S}
            end; {of S}
      begin {of T}
            U;
      end; {of T}
begin {of anomaly}
      T;
end. {of anomaly}
```

این برنامه سه تفسیر متفاوت دارد :

- فراخوانی S{2} در داخل U می تواند به منظور S{3} تلقی شود که در این صورت باید Forward انجام می شد چون تعریف S بعد از فراخوانی قرار دارد. در برخی کامپایلرها، این فراخوانی به این صورت تلقی شده و برنامه ترجمه می شود (تفسیر نادرست ولی منطقی)
- فراخوانی S{2} در داخل U می تواند به منظور S{1} تلقی شود که در این صورت با قوانین پاسکال در تضاد است چون یک زیربرنامه نمی تواند زیربرنامه هم سطح با اجداد خود را فراخوانی کند (تفسیر بسیار نادرست)
- عمل کامپایل با شکست مواجه می شود. چون اعلان پیشرو برای S{3} در داخل T انجام نشده است.

۹-۶- نام مستعار[۱]

یک شی داده ممکن است در طول عمرش چندین نام داشته باشد، یعنی ممکن است چندین وابستگی در محیطهای ارجاع مختلف داشته باشد؛ به گونه ای که هر یک از محیط ها، آن شیء را با نام مختلفی می شناسند؛ مانند هنگامی که یک شیء داده به عنوان پارامتر از نوع ارجاع به یک زیر برنامه فرستاده شود، وقتی از طریق بیش از یک نام بتوان به یک شیء داده ای دست یافت هر کدام از این نامها را، نام مستعار می خوانند. اگر شیء داده چند نام داشته باشد ولی هر نام در هر محیط ارجاع، منحصر به فرد باشد مشکلی پیش نمی آید. اما اگر در یک محیط ارجاع بتوان از طریق چند نام به شیء داده ای دست یافت هم برنامه نویس و هم پیاده ساز با مشکلات جدی مواجه خواهند شد.

به عنوان مثال در برنامه «الف» شکل زیر نام مستعار وجود ندارد ولی در برنامه «ب» در زیر برنامه sub1، اسامی I ،J نام مستعاری برای یک شیء داده ای هستند چرا که I از طریق «ارسال پارامتر با ارجاع» به زیر برنامه فرستاده شده است.

[۱] aliasing

در مثال زیر در برنامه sub1 دو نام I و J نام مستعار هستند:

```
Program main
      Var I : Integer;
      Procedure sub1 ( var J : Integer );
      Begin
            …{ I and J refer to same }
      End;
      Procedure sub2;
Begin
            …
            Sub1 ( I ) ; { I is visible J is not }
            …
      End;
Begin
      …
      Sub2; { I is visible J is not }
      …
End
```

در حالی که در مثال زیر نام مستعار وجود ندارد :

```
Program main;
      Procedure sub1 ( var J : integer );
      Begin
            …{ J is viible I is not }
      End;
      Procedure sub2;
            Var  I : integer;
      Begin
            …
            Sub1 ( I ); { I is visible J is not }
            …
      End;
Begin
      …
      Sub2; { neither is visible }
      …
End.
```

در برخی اوقات نام مستعار به تغییر نتیجه برنامه در ترتیب‌های متفاوت دستوراتی که در ظاهر تاثیری بر روی هم ندارند منجر می شود. به عنوان مثال اگر دو نام X,C نام‌های مستعاری برای یک شیء داده باشند تغییر ترتیب دو دستور زیر نتایج متفاوتی را منجر می شود.

```
X: = A + B;

Y: = C + D;
```

نکته: معمولاً نام‌های مستعار بهینه سازی کد برنامه را دشوار می کنند.

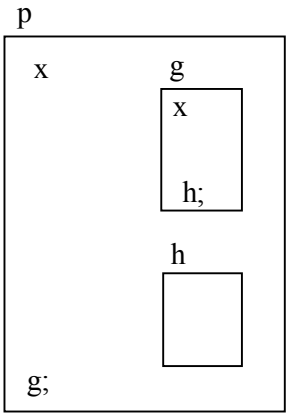
۹-۷- حوزه پویا و ایستا

محیط غیر محلی برای زیر برنامه p، متشکل از مجموعه ای از محیطهای محلی رکورد های فعالیت سایر زیر برنامه‌هایی است که در حین اجرا به p دستیابی دارند. وقتی در زیربرنامه p به متغیر x رجوع شود و x وابستگی محلی نداشته باشد از محیط غیر محلی برای تعیین وابستگی x استفاده می‌شود. برای استفاده از وابستگی اسامی غیر محلی، دو قاعده حوزه ایستا و پویا وجود دارد. در حوزه پویا برای پیدا کردن وابستگی یک شناسه (اسم) باید در زمان اجرا رکورد فعالیت جاری جستجو شده و در صورت عدم پیدا شدن وابستگی، رکوردهای فعالیت زنجیره پویا به ترتیب جستجو شوند تا وابستگی یافت شود در حالیکه در حوزه ایستا رکوردهای فعالیت زیر برنامه های در برگیرنده زیر برنامه جاری در ساختار برنامه جستجو می شوند.

قاعده حوزه پویا[۱]:

حوزه پویای هر وابستگی را بر حسب حالت پویای اجرای برنامه تعریف می کنند. به عنوان مثال فرض می کنیم یک وابستگی برای شناسه x در حین ورود به زیر برنامه f ایجاد شده است، قاعده حوزه پویا بیان می کند که از زمان اجرای f، حوزه پویای وابستگی x علاوه بر خود رکورد فعالیت، شامل زیر برنامه‌های دیگری است که توسط f فراخوانی می شوند و حتی اگر زیر برنامه ی دیگری توسط زیر برنامه ای فراخوانی شود که f آن را فراخوانی کرده است شامل آن نیز می شود. زنجیره پویای رکورد فعالیت زیر برنامه f، شامل خود f، زیر برنامه‌هایی که f را فراخوانی کرده اند و نیز زیر برنامه فراخوانی شده توسط f می باشد و ...

در روش حوزه ارجاعی پویا، اغلب از قانون « تازه ترین وابستگی[۲]» استفاده می شود. در این قانون اگر مثلاً تابع f، g را صدا بزند و g هم h را صدا بزند و متغیر x هم در f و هم در g تعریف شده باشد ولی در h تعریف نشده باشد، هنگام استفاده از x در h، کنترل به x موجود در g ارجاع می شود ؛ یعنی در این قانون متغیر به نزدیک ترین تابعی در زنجیره فراخوانی‌ها ارجاع می شود. (f—›g—›h)



شکل ۹ -۳

[۱] Dynamic Scope Rule
[۲] Most Recent Association

مثال)خروجی برنامه زیر در حوزه پویا چیست؟

```
Procedure A( )
      Var   x: integer;
      Procedure   B ( )
      Begin
            X: =x+1;
            Write(x);
      End;
      Procedure C ( )
            Var x: integer;
      Begin
            X: =30;
            B ( );
      End;
Begin
      X:=7;
      B ( );
      C ( );
End;
```

مرتبه اول ( ) B توسط A صدا زده می شود پس x درون ( ) B به x درون A ارجاع می شود. مرتبه دوم( ) B توسط ( )C صدا زده می شود پس x درون ( ) B به x درون ( ) C ارجاع داده می شود. بنابراین خروجی ۳۱ و ۸ خواهد بود.

قاعده حوزه ایستا[۱] :

در حوزه ارجاعی ایستا، اسامی در زمان کامپایل و بر اساس ساختار تو در تویی زیر برنامه‌ها مشخص می‌شوند. به عنوان مثال زبان پاسکال که از حوزه ارجاعی ایستا استفاده می‌کند متغیرهای غیر محلی یک زیر برنامه، متغیرهای متعلق به تابع در بر گیرنده آن زیر برنامه می‌باشند. به عنوان مثال در پاسکال قاعده حوزه ایستا تعیین می‌کند که ارجاع متغیر X در برنامه F به اعلان X در آغاز F اشاره می‌کند یا اگر در آنجا اعلان نشده باشد به اعلانی از X در آغاز زیر برنامه q اشاره دارد که خود تعریف زیر برنامه F داخل برنامه q می‌باشد و q مثلا داخل N و N داخل ....

به طور کلی قاعده حوزه ایستا، ارجاع‌ها را به اعلان اسامی در متن برنامه مربوط می‌کند ولی قاعده حوزه پویا ارجاع‌ها را با وابستگی‌های اسامی (فراخوانی‌ها) در حین اجرای برنامه ربط می‌دهد. همواره ایده آل این است که بین دو قاعده ایستا و پویا سازگاری وجود داشته باشد یعنی در هر دو حالت، وابستگی تعیین شده برای شناسه X یک زیر برنامه یکسان باشد که برقراری این حالت مشکل است.

برخی از مزایای حوزه ایستا عبارتند از: الف- امکان بررسی کنترل نوع ایستا ب- امکان انقیاد در زمان ترجمه ج- سرعت اجرای بیشتر د- هزینه کمتر ه- امکان ایجاد قوانین مشخص و واضح ساختار بلوکی

نکته: اغلب زبان‌هایی که از حوزه ایستا استفاده می کنند مانند C و پاسکال و Ada، کنترل نوع ایستا را انجام می دهند. بنابراین در این زبان ها، ساختارهای زمان اجرا ساده تر شده و برنامه سریع تر اجرا می شود.

نکته: زبان‌های APL,Lisp,SNOBOL4 از حوزه پویا استفاده می کنند و لذا ارجاع به یک نام در این زبان ها نیازمند فرایند پیچیده و پرهزینه ای است.

---

[۱] static scope rule

به طور خلاصه در مورد دو قاعده حوزه ایستا و پویا داریم:

قاعده حوزه ایستا: اگر بخواهیم خروجی برنامه را در حوزه ایستا بررسی کنیم در صورت وجود نداشتن یک اسم به صورت محلی در یک بلاک، باید به بلاک‌های بیرونی تر آن مراجعه کرده و از مقادیر آنها استفاده کرد.

قاعده حوزه پویا: اگر بخواهیم خروجی برنامه را در حوزه پویا بررسی کنیم در صورت وجود نداشتن یک اسم به صورت محلی در یک بلاک، باید سراغ بلاکی برویم که زیر برنامه را فراخوانی کرده است (قاعده تازه ترین وابستگی)

نکته: در دامنه پویا عمل وابستگی اسامی متغیرها به اشیا با توجه به سلسله مراتب فراخوانی روال‌ها (معنایی) و در زمان اجرا صورت می‌گیرد. در دامنه ایستا عمل وابستگی اسامی متغیرها به اشیا با توجه به تودرتویی تعریف روال‌ها (نحوی) و در زمان ترجمه انجام می‌شود.

مثال)خروجی برنامه زیر در حوزه ارجاعی ایستا را مشخص کنید؟

```
Porcedure A( )
      var x:integer;
      procedure B( )
      begin
            x:=x+1;
            write(x);
      end;
      procedure C( )
            var x:integer;
      begin
            x:=30;
            B( );
      end;
begin
      x:=7;
      B( );
      C( );
end;
```

در این برنامه اگر از حوزه ارجاعی ایستا استفاده شود هنگام صدا زدن B( ) چون زیر برنامه B( ) متغیر محلی x را ندارد، از متغیر محلی x مربوط به زیربرنامه دربر گیرنده آن یعنی A( ) استفاده کرده و خروجی آن برابر ۷+۱ یعنی ۸ می شود. سپس با صدا زدن زیر برنامه C( )، داخل این زیر برنامه یک متغیر محلی X با مقدار ۳۰ ساخته می‌شودکه ربطی به X موجود در A( ) ندارد. حال پس از صدا زدن B( ) درون زیر برنامه C( )، دوباره زیر برنامه B( ) از X مربوط به A( ) استفاده کرده و خروجی آن ۹=۸+۱ می شود. بنابراین، خروجی نهایی ۹ و ۸ می‌شود.

در زیر به چند نمونه از تست های کنکور در زمینه قواعد حوزه ایستا و پویا می پردازیم.

۱- خروجی برنامه زیر در حالتی که از قواعد static scoping و dynamic scoping استفاده شود، به ترتیب کدام گزینه است؟ (مهندسی کامپیوتر ۸۵ – جواب: گزینه ب)

```
Program Main ;
     var M :integer
     Function   F(X:integer) :integer;
         Begin
                 F:=X*20
         end;
      Procedure  P(I:integer);
         var   Z:integer;
         begin
             Z:=F(I)*M;
             write(Z)
         end
      Procedure  Q;
         var   K :integer;
                 M :integer;
        Function  F(Y :integer):integer;
                  begin
                        F :=Y*30
                  end
         begin
                   M :=3;
                   K :=10;
                   P(K)
          end
   begin
          M:=2;
          Q;
   end.
```

الف. dynamic scoping: 600 , static scoping: 400 ب. dynamic scoping: 900 , static scoping: 400

ج. dynamic scoping: 900 , static scoping: 600 د. dynamic scoping: 400 , static scoping: 900

۲- در قطعه برنامه زیر که به زبان برنامه سازی ML نوشته شده است مقدار عبارت b*f(a,a) در دو حالتی که زبان از قواعد حوزه ایستا و حوزه پویا استفاده می کند، کدام است؟(کامپیوتر ۸۶ جواب: گزینه ج)

```
let a = 5,b =10, c = ۷ in
   let val fun f(x,y) = a*(x+y) + b
       let val  a = 4 , b = 2  in
                b * f(a,a)
     end
   end
end
```

الف. حوزه ایستا: ۶۰ و حوزه پویا :۳۴ ب. حوزه ایستا: ۵۰۰ و حوزه پویا :۳۴۰

ج. حوزه ایستا: ۱۰۰ و حوزه پویا :۶۸ د.هیچکدام

۳-برنامه زیر را در نظر بگیرید. کدام گزینه صحیح است؟(مهندسی کامپیوتر ۷۳ جواب: گزینه د)

```
Program test;
      var x:real;
      procedure Display;
      begin
            write(x);
      end;
      procedure Assign;
            var x:real;
      begin
            x:=0.72;
            Display;
      end;
begin
      x:=0.27;
      Display;
      Assign;
end.
```

الف) در دامنه پویا مقدار چاپ شده توسط برنامه 0.27 و 0.72می‌باشد.
ب) در دامنه ایستا مقدار چاپ شده توسط برنامه 0.27 و 0.27می‌باشد.
ج) در دامنه ایستا مقدار چاپ شده توسط برنامه 0.27 و 0.72می‌باشد.
د) گزینه الف و ب

۴-خروجی برنامه زیر با استفاده از قواعد حوزه پویا (Dynamic Scope) چیست؟(مهندسی کامپیوتر ۷۲ جواب: گزینه ب)

```
Program main;
      var x:integer;
      procedure foo;
            var x:integer;
      begin
            x:=7;
            bar;
      end
      procedure bar;
      begin
            print x;
      end;
begin
      x:=3;
      foo;
end.
```

الف) ۳ ب) ۷ ج) ۳و ۷ د) ۷و ۳

۵-اگر در اجرای برنامه M زنجیره فراخوانی برنامه‌های فرعی به صورت M→P→R→Q→S (یعنی M ، P را فرا خوانی کرده ، P ، R را ، R ، Q را و Q ، S را فراخوانی کرده است) ، محیط ارجاع را در دو حالت پیاده‌سازی قانون شناسایی ایستا و قانون حوزه شناسایی پویا در نظر بگیرید. کدام یک از متغیرهای تعریف شده در برنامه در هر دو محیط ارجاع وجود دارند؟ به عبارت دیگر کدام یک از متغیرهای برنامه در هر دو حالت از داخل S قابل ارجاع هستند؟ (مهندسی کامپیوتر ۷۵ جواب: گزینه الف)

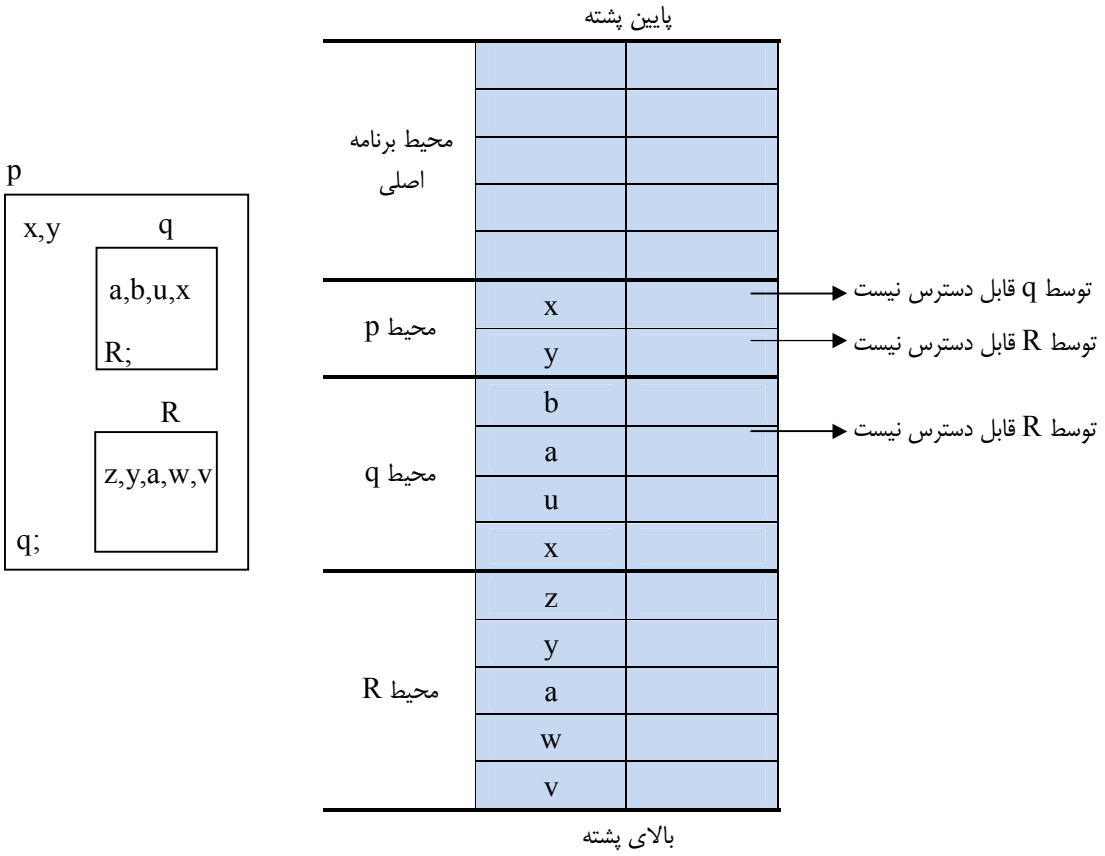
```
Program M;
      DCL x;
      procedure P;
            DCL y;
            procedure R;
                  DCL w;
            begin
                  …
            end; {of R}
      begin
            …
      end; {of P}
      Procedure Q;
            DCL z;
            procedure S;
                  DCL u;
            begin
                  …
            end; {of S}
      begin
            …
      end; {of Q}
begin
      …
end. {of M}
```

الف) x,z.u ب) x.y,z,w.u ج) x,y,z,u د) هیچکدام

۹-۸- پیاده سازی حوزه پویا و ایستا

پیاده سازی حوزه پویا:

یکی از روش‌های پیاده سازی حوزه پویا، استفاده از پشته مرکزی است. فرض کنید زیر برنامه p، زیر برنامه q را صدا بزند و زیر برنامه q نیز زیر برنامه R را صدا بزند (p→q→R) . هنگامی که R اجرا می‌شود پشته مرکزی به شکل زیر خواهد بود. جهت پردازش ارجاع غیر محلی x، پشته با شروع از محیط محلی R به طرف عقب جستجو می‌شود تا تازه ترین وابستگی برای x پیدا شود. در این حال، برخی از وابستگی‌های موجود در پشته با وابستگی‌های بعدی برای همان شناسه، مخفی می شوند(مانند x,y,a که برای زیر برنامه R مخفی می شود).



شکل ۹ -۴

پیاده سازی حوزه ایستا:

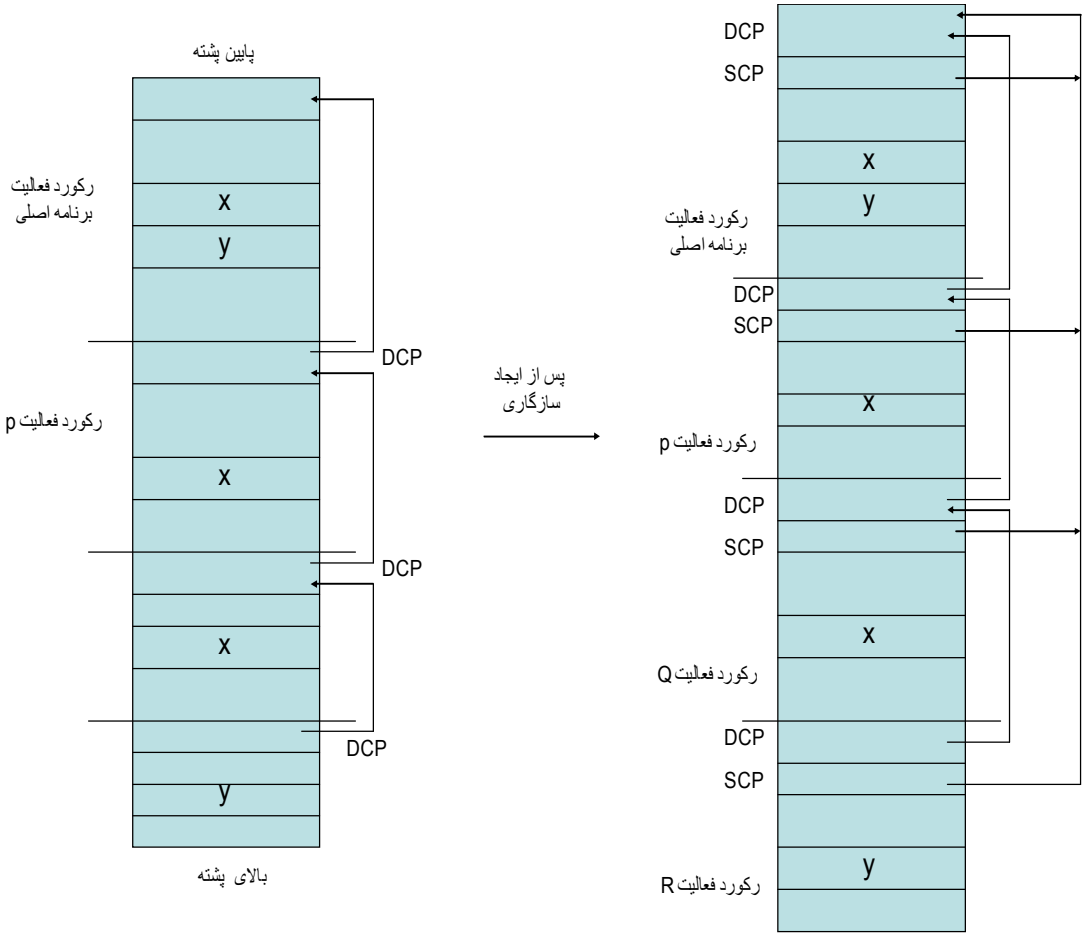
در زبان‌هایی مانند پاسکال و Ada که از ساختار بلوکی استفاده می‌کنند پردازش ارجاع‌های غیر محلی پیچیده تر است. در شکل صفحه بعد که نمونه ای از قواعد حوزه ایستا را برای برنامه ساخت یافته بلوکی در پاسکال نشان می‌دهد زیربرنامه R از Q فراخوانی می‌شود و زیر برنامه Q نیز از P فراخوانی می‌شود. زیربرنامه‌های P و Q و main متغیر x را تعریف می‌کنند. در داخل R، x به طور غیر محلی ارجاع می‌شود. طبق قاعده حوزه پویا، هنگام اجرای R، متغیر x باید متغیر تعریف شده در Q باشد در حالیکه طبق قاعده حوزه ایستا متغیر x به x موجود در برنامه main اشاره دارد و باید هنگام اجرای دستور x=x+1 از آن استفاده شود لذا سازگاری بین قواعد حوزه ایستا و پویا لازم است.

برای ایجاد سازگاری، باید حوزه ایستا در زمان اجرا کنترل شود و در هر رکورد فعالیت یک اشاره گر زنجیره ایستا (SCP[۱]) ذخیره شود که آدرس رکورد فعالیت در برگیرنده را نگهداری می کند. از این طریق می‌توان زنجیره ایستای برنامه را دنبال کرد. در این مثال، به هنگام اجرای R، با توجه به SCP که به رکورد فعالیت برنامه main اشاره می‌کند و با استفاده از یک آفست (تفاوت مکان)، متغیر x موجود در برنامه main در اختیار x قرار می‌گیرد. در روش حوزه ایستا، از اشاره گر زنجیره ایستا (SCP) و در روش حوزه پویا، از اشاره گر زنجیره پویا (DCP[۲]) برای پیدا کردن وابستگی استفاده می شود. اشاره گر زنجیره ایستا (SCP)، آدرس رکورد فعالیت زیر برنامه سطح بالاتر (بیرونی تر) را نگهداری می کند ولی اشاره گر زنجیره پویا (DCP)، آدرس رکورد فعالیت فراخوانی شده توسط زیر برنامه جاری را نگهداری می کند. برنامه زیر را در نظر بگیرید:

[۱] Static Chain Pointer
[۲] Dynamic Chain Pointer

```
Program main;
      var x,y: integer
      procedure R
            var y: real
      begin
            x:=x+1
      end;
      procedure Q
            varx :real
      begin
            R;
      end
      Procedure P
            var x: Boolean
      begin
            Q;
      end
begin
      P;
End;
```

برای درک مفاهیم زنجیره ایستا و پویا برای اجرای برنامه فوق شکل زیر را مشاهده کنید.



پشته ناقص مرکزی در حین اجرا

پشته تکمیل شده در حین اجرا

شکل ۹ -۵

۹-۹- ساختار بلوکی

مفهوم ساختار بلوک در زبان‌های ساختیافته بلوکی مانند پاسکال پیدا شد. مفاهیم مربوط به ساختار بلوکی از زبان Algol 60 سر چشمه گرفته شده است. در یک زبان ساختیافته بلوکی، هر برنامه یا زیر برنامه به صورت مجموعه ای از بلوک‌های تودرتو سازماندهی می‌شوند. ویژگی مهم بلوک آن است که محیط ارجاع جدیدی تعریف می‌کند. هر بلوک می‌تواند شامل تعاریف بلوک‌های دیگر باشد و به این صورت بلوک‌های تودرتو[۱] پدید می‌آیند. اسامی قابل دستیابی غیر محلی را به دو صورت می‌توان مشخص کرد، یکی به صورت حوزه ایستا و دیگری به صورت حوزه پویا. قواعد حوزه ایستا مربوط به برنامه ساخت یافته بلوکی عبارتند از:

- اعلان‌های ابتدای هر بلوک، محیط ارجاعی محلی آن بلوک را پدید می‌آورند. اگر داخل یک بلوک، از اسمی استفاده شود که در اول آن بلوک به صورت محلی تعریف شده است کامپایلر از آن اسم محلی استفاده می‌کند و به سراغ دیگر اسامی نمی‌رود.
- اگر در بلوکی از اسمی استفاده شود که در آن به صورت محلی وجود ندارد، یک بلوک به سمت بیرون رفته و در اسامی بلوک در بر گیرنده آن دنبال آن اسم می‌گردد. اگر اسم را پیدا نکرد این عملیات را به سمت بلوکهای بیرونی تکرار می‌کند تا بالاخره آن اسم را یافته و از آن استفاده می‌کند.
- اسامی محلی موجود درون یک بلوک از دید بلوک خارجی آن مخفی است.
- بلوک می تواند دارای نام باشد. نام بلوک بخشی از محیط ارجاع محلی در برگیرنده آن محسوب می شود.

۹-۱۰- محیط ارجاع برای داده‌های محلی

در اینجا محیط ارجاع برای داده‌های محلی که ساده ترین ساختار را دارند بررسی می کنیم. محیط محلی زیربرنامه شامل پارامترهای مجازی و متغییرهای محلی آن زیربرنامه می باشد.

```
Procedure sub1(a:real);
      Var b:real;
       Procedure sub2(c:real);
            Var d:real;
             Begin
                   c:=c+b;
             End;
      Begin
           …..
       End;
```

متغیر محلی در زیربرنامه sub2 عبارتست از d و متغییر محلی در زیربرنامه sub1 عبارتست از c.

[۱] nested

وابستگی متغیرهای محلی به دو صورت «حذف» و «نگهداری» می باشد. این دو مفهوم را همراه با یک مثال توضیح می دهیم.

روش نگهداری[۱]:

در این روش، در اولین ورود به زیربرنامه متغیر تعریف و اجرا می شود. در موقع بازگشت از زیر برنامه متغیر از بین نمی رود و در فراخوانی‌های بعدی از آخرین مقدار متغیر استفاده می شود. کوبول و فرترن از روش نگهداری استفاده می کنند. در پیاده سازی این روش، محل نگهداری اشیاء داده ای (جدول محیط محلیL.E.T) در کد سگمنت برنامه می باشد.

روش حذف[۲]:

در این روش متغیر در اولین ورود به زیربرنامه ایجاد می شود و به هنگام بازگشت از بین می رود. زبان‌های Ada,Lisp,C,Pascal,SNOBOL4,APL,Java,C++ از روش حذف استفاده می کنند. در پیاده سازی این روش محل نگهداری اشیا داده (جدول محیط محلیL.E.T) در رکورد فعالیت زیربرنامه (حافظه پشته) می باشد.

نکته: Algol و PL/I ازهر دو روش حذف و نگهداری استفاده می کنند.

مثال: خروجی تکه برنامه زیر را با استفاده از روش نگهداری و حذف بدست آورید؟

```
Procedure Q;
     Var x:integer:=30;
      Begin
          Write(x);
           X:=x+1;
      End;
Procedure P;
     Begin
         Q;
         Q;
         Q;
     End;
```

۱. نگهداری: اگر وابستگی x نگهداری شود مقدار اولیه x:=30 فقط بار اول صدا زدن Q انجام می گیرد و پس از خروج از Q مقدار x حفظ می شود. بار اول که Q داخل p صدا زده می شود، مقدار اولیه ۳۰ چاپ شده و x برابر ۳۱ می شود. بار دوم که Q صدا زده می شود مقدار قبلی ۳۱ چاپ شده و x برابر ۳۲ می شود و بار سوم این مقدار ۳۲ چاپ می شود. بنابراین، خروجی نهایی ۳۲ و۳۱ و۳۰ می باشد.

۲. حذف: در این روش در هر بار خروج از Q، x حذف شده و در هر بار ورود به Q یک x جدید با مقدار اولیه ۳۰ ساخته می شود. لذا در این حالت خروجی نهایی برنامه ۳۰ و ۳۰ و۳۰ می شود.

پیاده سازی روش حذف و نگهداری:

برای پیاده سازی روش حذف و نگهداری، محیط ارجاع محلی هر زیربرنامه در یک جدول محیط محلی (L.E.T[۳]) شامل نام، نوع و ..ذخیره می شود. بعنوان مثال جدول L.E.T برای زیر برنامه زیر به صورت زیر است:

---

[۱] Retention
[۲] Deletion
[۳] Local Environment Table

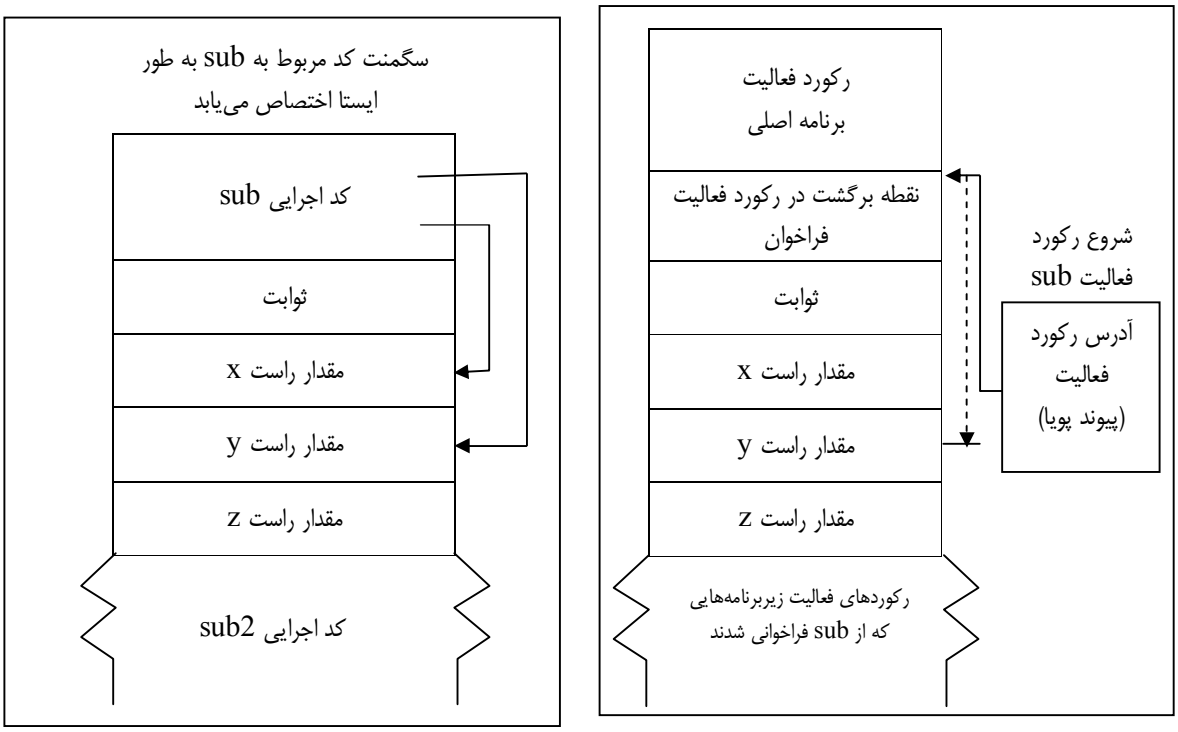
```
Procedure sub (x:integer) is
      y:real;
      z:array (1..3) of real;
      procedure sub2 is
      begin
            ...
      end {sub2};
begin
      ...
end {sub};
```

تعریف زیربرنامه

محتویات Lvalue	نوع	نام
پارامتر با مقدار	Integer	x
متغیر محلی	Real	y
آرایه	Real	z
توصیفگر : [1..3]		
اشاره گر به سگمنت کد	Procedure	Sub2

جدول محیط محلی sub (در زمان ترجمه)

شکل ۹ -۶ نمونه ای از جدول محیطی



شکل ۹ -۷ تخصیص و ارجاع به متغیرهای محلی قابل حذف شدن شکل ۹ -۸ تخصیص و ارجاع به متغیرهای محلی نگهداری شده

در روش نگهداری،L.E.T در سگمنت کد تشکیل می‌شود چون سگمنت کد به طور ایستا تخصیص می‌یابد و در حین اجرا باقی می‌ماند. هر متغیر موجود در بخش محیط محلی سگمنت کد نگهداری می‌شود درحالیکه در روش حذفL.E.T به عنوان قسمتی از رکورد فعالیت آن زیربرنامه می‌باشد و در پشته مرکزی تشکیل می‌شود. چون رکورد فعالیت یک زیربرنامه به هنگام ورود به زیر برنامه در پشته مرکزی ایجاد می‌شود و با خروج از زیر برنامه از بین می‌رود. بنابراین برای زیر برنامه‌های برگشتی روش حذف بسیار مناسب می‌باشد چون روش نگهداری فضای زیادی تلف می‌کند.

نکات تکمیلی روش حذف و نگهداری:

- نگهداری و حذف دو روش مختلف برای معنای محیطهای محلی اند و به طول عمر این محیط ها مربوط می‌شوند. آنهایی که مقادیرشان در حافظه تخصیص یافته در سگمنت کد نگهداری می‌شود و آنهایی که مقادیرشان در رکورد فعالیت قرار می‌گیرند و حافظه آنها باید حذف شود. در بسیاری از زبان ها، متغیرهای محلی static از نوع نگهداری و متغیرهای محلی automatic از نوع حذف هستند.
- روش نگهداری به برنامه نویس اجازه می‌دهد تا زیر برنامه‌هایی ایجاد کند که نسبت به گذشته حساس باشند به طوری که بخشی از نتایج آنها در هر فراخوانی توسط ورودی و بخشی دیگر توسط داده‌های محلی که در حین رکورد فعالیت قبلی ایجاد شده اند تعیین گردد در حالیکه که در روش حذف برای انتقال داده‌ها از یک فراخوانی به فراخوانی دیگر از همان زیر برنامه باید یک متغیر به صورت غیر محلی ایجاد شود. در روش حذف، متغییرهای محلی امکان حفظ کردن مقدار خود را ندارند و لذا این کار باید با متغییرهای غیر محلی انجام شود که امنیت و جامعیت برنامه را کاهش می دهد.
- روش حذف باعث صرفه جویی بیشتری در حافظه می شود و این روش برای زیر برنامه‌های بازگشتی متداول تر است
- در روش حذف سرعت اجرای برنامه به دلیل ایجاد و حذف داده‌ها کمتر است
- نام زیر برنامه به اعلانی برای آن زیربرنامه در محیط محلی وابسته می‌شود، بنابراین همواره نگهداری می‌شود.
- نام پارامتر مجازی یک شی داده ای را نشان می‌دهدکه در هر بار فراخوانی زیربرنامه، مقدار جدیدی می گیرد. بنابراین با این پارامترها به روش حذف برخورد می‌شود. در الگول ۶۰ وقتی متغیر محلی با own اعلان شود، نگهداری خواهد شد.

۹-۱۱- پارامترها وانتقال پارامترها

گاهی اوقات مواردی پیش می‌آید که درحین اجرای زیربرنامه‌ها، بایستی یک سری اطلاعات بین زیربرنامه‌ها به اشتراک گذاشته شود. معمولاً به چهار طریق این امکان فراهم شده تا زیربرنامه‌ها به اطلاعات مشترک (محیط مشترک) دسترسی داشته باشند.

- **محیط اشتراک صریح[۱]:**

در این محیطها داده های اشتراکی و زیر برنامه هایی که از آنها می توانند استفاده کنند صراحتاً باید مشخص گردند. مثلاً در زبان فرترن داده‌های مشترک در بخشی از برنامه قرار داده می شود و آن بخش توسط دستور Common صراحتاً مشخص می گردد. نمونه‌هایی از محیطهای اشتراک صریح عبارتند از: کلاس‌ها در C+ + و بلوک Common در فرترن و Package در Ada

- **محیط اشتراک ضمنی[۲]:**

در این گونه محیطها داده‌ها بین زیر برنامه‌ها مشترک هستند ولی تعریف صریحی از آنها انجام نمی شود. برای نمونه در زبان پاسکال می توان برای استفاده از متغییرهای تعریف شده در یک UNIT آن را Import کرد که این کار توسط دستور USES در ابتدای برنامه انجام می شود. در زبان Modula2 نیز این امکان وجود دارد.

---

[۱] Explict
[۲] Implict

- **قواعد حوزه ایستا و پویا**
- **وراثت**

محیط‌های مشترک یک راه ارتباط زیربرنامه‌ها با یکدیگر است. روش متداول تر دیگر جهت انتقال اشیا داده بین زیربرنامه‌ها، استفاده از تکنیک انتقال پارامترهاست. در این تکنیک، انتقال اطلاعات بین زیر برنامه‌ها، از طریق قرار دادن آنها در یک سری پارامترها و فرستادن آنها به زیر برنامه فراخوان امکان پذیر می‌باشد. برای بررسی تکنیک انتقال پارامتر نیازمند یک سری تعاریف اولیه هستیم که در زیر به بررسی آنها می‌پردازیم:

۹-۱۱-۱- پارامترهای مجازی و واقعی و تناظر بین آنها

به پارامترهای زیر برنامه در هنگام تعریف آن پارامترهای مجازی یا رسمی (فرمال) گفته می‌شود. به پارامترهای زیر برنامه هنگام فراخوانی، پارامترهای واقعی می‌گویند. مثلاً در تکه برنامه زیرa,b پارامترهای مجازی و x,y پارامترهای واقعی هستند.

```
Int fn(int a,int b){
      Return a+b;
}
Main( ){
   Int x=5,y=6,z;
   Z=fn(x,y);
}
```

پارامترواقعی، یک شیء داده ای است که با زیر برنامه فراخوان مشترک است. پارامتر واقعی ممکن است یک شیء داده محلی متعلق به فراخوان باشد، ممکن است پارامتر مجازی فراخوان باشد ممکن است شیء داده غیر محلی باشد که توسط فراخوان قابل مشاهده است یا ممکن است نتیجه ای باشد که توسط تابعی برگردانده شده است که زیر برنامه فراخوان آن تابع را فراخوانی کرده و نتیجه را به زیربرنامه فراخوانی شده ارسال کرده است.

پارمتر واقعی در p	فراخوانی زیر برنامه در p
متغیرهای محلی p : B و I	Sub (I , B)
ثوابت :27 , true	Sub (27 , true)
پارامترهای مجازی p : p1 , p2	Sub (p1 , p2)
متغیرهای عمومی یا غیرمحلیp : G1 , G2	Sub (G1 , G2)
عناصر آرایه‌ها و رکوردها	Sub (A[i] , D , B1)
نتایج توابع تعریف شده یا اولیه	Sub (i+3 , fn(Q))

جدول ۹ -۱

هنگام فراخوانی باید تناظری بین پارامترهای واقعی و مجازی برقرار گردد برای این کار دو قاعده وجود دارد:

- **تناظرموقعیتی:** دراین تناظر که در اکثر زبان‌ها استفاده می‌شود، اولین پارامتر واقعی به اولین پارامتر مجازی، دومین پارامتر واقعی به دومین پارامتر مجازی و ...نگاشت می‌شود. زبان‌های C و پاسکال از این روش استفاده می کنند.
- **تناظر براساس نام:** در برخی از زبان‌ها مانند Ada از تناظر نام استفاده می‌شود یعنی پارامترها براساس نام نگاشت می شوند و ترتیب آنها مهم نیست مثلا دستور فراخوانی زیر در زبان Ada را در نظر بگیرید:

```
Sub(x,y){     تعریف
      ...
      ...
}
Sub  ( x=>B  ,  y=>27)  فراخوانی
```

در حین فراخوانیSub، پارامتر واقعی B با پارامتر مجازی x و پارامتر واقعی ۲۷ با پارامتر مجازی y متناظر می‌شوند.

۹-۱۱-۲- روش‌های انتقال پارامتر

بعد از اینکه پارامترهای واقعی به پارامترهای مجازی نسبت داده شدند این انتساب ها به روش های مختلفی تفسیر و استفاده می‌شوند. به طور کلی روش‌های انتقال پارامترها[1] عبارتند از:

- فراخوانی با مقدار (Call by value)
- فراخوانی با ارجاع (Call by refrence)
- فراخوانی با نام (Call by name)
- فراخوانی با نتیجه (Call by result)
- فراخوانی با مقدار- نتیجه (Call by value-result)
- فراخوانی با مقدار ثابت (Call by const)

۹-۱۱-۲-۱- فراخوانی با مقدار

در حالت معمولی در زبان‌هایی مانند C و پاسکال پارامترها به صورت فراخوانی با مقدار به زیربرنامه فرستاده می‌شوند، در این روش در هنگام فراخوانی مقادیر (مقدار راست) پارامترهای واقعی در پارامترهای مجازی(رسمی) کپی می‌شوند ولی در هنگام برگشت از زیر برنامه فراخوانی شده، مقادیر پارامترهای رسمی هیچ تاثیری روی پارامترهای واقعی ندارند.

مثال: خروجی برنامه زیر را با فرض اینکه روش انتقال پارامتر با مقدار باشد بدست آورید:

```
Void fn( int x, int y){
      x=0; y=0 ;
      Printf ("%d %d", x ,y);
};
Main ( )
      Int a=1, b=3 ;
      Printf ("%d  %d" , a ,b );
      Fn (a ,b );
      Printf ("%d %d" , a, b ); }
```

خروجی:

۳ ۱	قبل از فراخوانی
۰ ۰	در زیر برنامه فراخوانی شده
۳ ۱	بعد از فراخوانی

جدول ۹ - ۲

[1] parametr passing

۹-۱۱-۲-۲- فراخوانی با ارجاع (آدرس)

این روش، متداول ترین روش انتقال پارامترهاست. در هنگام فراخوانی آدرس پارامترهای واقعی (مقدار چپ) به زیربرنامه فرستاده می‌شود. بدین ترتیب تغییراتی که به پارامترهای مجازی، درون تابع داده می‌شود، به پارامترهای واقعی متناظر در برنامه صدا زننده اعمال می‌شود. زبان C این نوع فراخوانی را با استفاده از اشاره گرها پیاده سازی کرده است.

مثال: خروجی برنامه زیر را با فرض اینکه روش انتقال پارامتر با ارجاع باشد بدست آورید:

```
Void fn(int x , int y){
      x=-4; y=-6;
      Printf("%d %d ",x ,y); //-4    -6
}
Main(){
      Int a=1,b=3;
      Printf("%d %d",a,b);    //1       3
      Fn(&a,&b);
      Printf("%d%d,a,b); //-4-6
}
```

قبل از فراخوانی	۳ ۱
در زیر برنامه فراخوانی شده	-۶ -۴
بعد از فراخوانی	-۶ -۴

جدول ۹ - ۳

در زبان پاسکال پارامترهایی که با مقدار فرستاده می‌شوند بدون var و پارامترهایی که با ارجاع فرستاده می‌شوند var دارند.

```
Var x,y;
Procedure fn(a:integer;var  b:integer);
Begin
      a:=-4;      b:=-8;
      writeln(a,b);          //   -4     -8
End;
Begin
      x:=3;
      y:=7;
      writeln(x,y);         //   3      7
      fn(x,y);
      writeln(x,y);         //   3     -8
End;
```

نکته: در زبان C فرستادن آرایه‌ها همواره به صورت فراخوانی با ارجاع است ولی در پاسکال اینگونه نیست. اگر قبل از نام پارامتر مجازی آرایه var نباشد فراخوانی بامقدار و اگرvar باشد فراخوانی با ارجاع است.

۹-۱۱-۲-۳- فراخوانی با نام

این روش کمتر مورد استفاده زبان‌ها می‌باشد انتقال پارامتر با نام در الگول از اهمیت بالایی برخوردار است ولی به دلیل سربار اجرایی بالا، روش کاربردی و معروفی نیست. در این روش نام پارامتر واقعی جایگزین نام پارامتر مجازی در زیربرنامه فراخوانی

شده می‌گردد. در این حال هر ارجاع به پارامتر مجازی مستلزم ارزیابی مجدد پارامتر واقعی متناظر با آن است. برای این کار در نقطه فراخوانی زیربرنامه، پارامترهای واقعی ارزیابی نمی‌شوند تا زمانی که در زیر برنامه به آنها مراجعه شود. برای پارامترهایی که از نوع متغیر ساده هستند مانند int، فراخوانی با نام از دید برنامه نویس از نظر نتیجه خروجی معادل فراخوانی با ارجاع است ولی اگر از نوع متغیر ساده نباشند ممکن است نتایج با هم فرق کند.

مثال: در procedure sub(x:integer) اگر بخواهیم فراخوانی sub(y) را در متن برنامه اصلی داشته باشیم می‌توان اینگونه فرض کرد که در بدنه روال sub به جای همه xها، y قرار داده می‌شود حال در هر مراجعه به y یک ارزیابی مجدد از y صورت می‌گیرد از نظر کاربر نتایج فراخوانی با نام و فراخوانی با ارجاع یکی است ولی نتایج میانی متفاوتی از هر دو دیده می-شود.

تکنیک اصلی برای پیاده سازی فراخوانی با نام این است که با پارامترهای واقعی مثل زیر برنامه‌های فاقد پارامتر (thunk) رفتار شود. وقتی از زیربرنامه به پارامتر مجازی متناظر با پارامتر واقعی مراجعه شود، thunk ترجمه شده برای آن پارامتر، اجرا می‌شود تا پارامتر واقعی در محیط ارجاع مناسبی ارزیابی شود و مقدار حاصل به عنوان مقدار thunk برگردانده شود.

مثال: خروجی برنامه زیر را با فرض اینکه روش انتقال پارامتر با نام باشد بدست آورید:

```
var  i:integer;
Function    f(x:integer):integer;      function  f(a[i]);
Begin                                  begin
      i=2;                             i=2;
       x=1;                            a[i]=1;
      i=3;                             i=3;
       x=3;                            a[i]=3;
End;                                   end;
Begin
      a:array [1..3] of integer={1,2,3};
      i=1;
       print(a[1],a[2],a[3])     //output:1,2,3
      f(a[i]);
       print(a[1],a[2],a[3])     //output:1,1,3
end;
```

با فراخوانی تابع f، نام پارامتر a[i] جایگزین نام پارامتر مجازی x در زیر برنامه f می‌شود سپس با هر بار رجوع به x مقدار a[i] محاسبه می‌شود.

۹-۱۱-۲-۴- فراخوانی با نتیجه

پارامتری که به این شیوه ارسال می‌شود فقط جهت برگرداندن نتیجه به برنامه فراخوان استفاده می‌شود. دراین روش در هنگام فراخوانی، مقادیر پارامترهای واقعی در پارامترهای رسمی کپی نمی شوند ولی در هنگام بازگشت آخرین مقادیر پارامترهای رسمی در پارامترهای واقعی کپی می شوند. به عبارتی دیگر می‌توان گفت در این نوع فراخوانی، تابع پارامتر ورودی ندارد و فقط دارای پارامتر خروجی است.

۹-۱۱-۲-۵- فراخوانی با مقدار-نتیجه

در این روش در هنگام فراخوانی، مقادیر پارامترهای واقعی در پارامترهای مجازی کپی می‌شوند. داخل زیربرنامه هر تغییری روی پارامتر مجازی انجام گیرد روی پارامتر واقعی اثر ندارد و در هنگام بازگشت از زیربرنامه، مقادیر پارامترهای مجازی در پارامتر واقعی کپی می‌شوند. یعنی پارامتر واقعی تا زمان خاتمه زیربرنامه مقدار اصلی خودش را حفظ کرده و پس از اجرای زیر برنامه مقدار جدیدی می‌گیرد. این روش فراخوانی در زبان Algol-w استفاده شده است.

۹-۱۱-۲-۶- فراخوانی با مقدار ثابت

هنگامی که پارامتر به صورت مقدار ثابت یا const انتقال داده می‌شود در حین اجرای زیربرنامه نمی‌توان پارامتر مجازی را تغییر داد و این پارامتر مجازی ثابت، در حین اجرای زیربرنامه مانند یک مقدار ثابت محلی عمل می‌کند. از دید برنامه فراخوان این پارامتر ثابت، فقط یک آرگومان ورودی برای زیربرنامه است و مقدارش چه به صورت سهوی و چه به منظور برگرداندن نتیجه، قابل تغییر نیست.

```
int fn(const int a, int b)
```

نکته: در زبان الگول امکان فراخوانی با نام وجود دارد. در زبان فرترن فراخوانی با ارجاع انجام می شود. در زبان C از فراخوانی با مقدار استفاده می شود و در زبان C++ و پاسکال از هر دو روش فراخوانی با مقدار و فراخوانی با ارجاع استفاده می شود. در زبان C++ اگر قبل از نام پارامتر مجازی & قرار داده شود ارسال آن به روش فراخوانی با ارجاع انجام خواهد شد. در زبان C نیز اگر به جای نام شی داده آدرس آن به روش فراخوانی با مقدار ارسال شود می توان فراخوانی با ارجاع را در این زبان شبیه سازی کرد. انتقال پارامتر به روش مقدار – نتیجه در الگول_w معرفی شده است.

نکته: در زبان Ada به جای توصیف مکانیزم انتقال پارامتر، نقش پارامتر مشخص می‌شود. اگر پارامتر به صورت in ارسال شود مقدار پارامتر واقعی به پارامتر مجازی ارسال می‌شود اگر پارامتر به صورت out باشد مقدارش توسط زیربرنامه تولید می-شود و هنگام خروج از زیربرنامه به پارامتر واقعی در زیربرنامه فراخوان منتقل می‌شود. اگر پارامتر به صورت inout باشد مقدارش هنگام فراخوانی به زیربرنامه فراخوانی شده منتقل می‌شود و هنگام خروج از آن، مقدارش در پارامتر واقعی در زیربرنامه فراخوان قرار می‌گیرد. به دلیل اینکه در زبان Ada فراخوانی با مقدار برای پارامترهای in و فراخوانی با ارجاع برای پارامترهای out، مشکلات تطابق را به وجود می‌آورد و برنامه‌های مختلف در کامپایلرهای گوناگون نتایج متفاوتی را برمی‌گرداند انتقال پارامترها بازبینی شد و اصلاحات زیر صورت گرفت:

انواع داده اولیه با پارامتر in با فراخوانی مقدار ثابت و با پارامتر out یا inout با فراخوانی مقدار و نتیجه ارسال می‌شوند و انواع داده مرکب (مثل آرایه و رکورد) با فراخوانی ارجاع ارسال می‌شوند.

نکته: فراخوانی با ارجاع و فراخوانی مقدار نتیجه از دید برنامه نویس نتایج یکسانی دارند.

۹-۱۱-۳- پیاده سازی انتقال پارامتر

چون هر رکورد فعالیت زیربرنامه، مجموعه متفاوتی از پارامترها را دریافت می‌کند حافظه پارامترهای مجازی زیر برنامه به بخشی از رکورد فعالیت زیربرنامه تخصیص می‌یابد و نه در سگمنت کد. بنابراین، اگر محل ذخیره پارامترهای رسمی به عنوان قسمتی از رکورد فعالیت زیربرنامه فراخوانی شده باشد و پارامتر رسمی به نام P از نوع T داشته باشیم یکی از دو روش زیر بسته به مکانیزم انتقال پارامتر، پیاده سازی می‌شود.

- P به عنوان شی داده محلی از نوع T است در صورتیکه ارزش اولیه آن یک کپی ارزش پارامتر واقعی باشد.
- P به عنوان شی داده محلی از نوع اشاره گری به T است در صورتیکه ارزش اولیه، اشاره گری به پارامتر واقعی باشد.

روش اول برای فراخوانی مقدار–نتیجه، فراخوانی با مقدار و فراخوانی با نتیجه استفاده می‌شود روش دوم برای انتقال پارامترها از طریق ارجاع و از طریق نام استفاده می‌شود. از هر دو روش می‌توان برای پیاده سازی انتقال از طریق مقدار ثابت استفاده کرد و برگرداندن مقدار با تابع نیز با روش اول انجام می‌شود.

اکنون به چند نمونه از تست‌های کنکور در زمینه انتقال پارامترها می پردازیم:

۱–در زبان فرضی زیر، آرگومانهای برنامه‌ها بصورت Call by Name تعریف شده اند. با توجه به این روش تعریف آرگومان، خروجی تکه برنامه زیر چیست؟ (مهندسی کامپیوتر۸۲– جواب:گزینه د)

```
Procedure Exchange(x,y :integer);
      Var temp:integer;
Begin
      Temp←x;

       x ← Y;
      Y←temp;
end;
      .
      .
I←4;
A[1]←8; A[۲]←6; A[۳]←4; A[۴]←2;
Exchange(I,A[I]);
Output(I,A[1], A[2], A[3], A[4]);
```

الف.۲و۲و۴و۸و۲ (از چپ به راست بخوانید.)　　ب.۴و۴و۶و۸و۲ (از چپ به راست بخوانید.)

ج.۲و۴و۶و۸و۴ (از چپ به راست بخوانید.)　　د.۲و۴و۴و۸و۲ (از چپ به راست بخوانید.)

۲–برنامه زیر را در نظر بگیرید. مقادیر خروجی برنامه زیر با استفاده از روش Call by name کدام است؟ (راهنمایی: توجه داشته باشید که j در عبارت S[1+j] به متغیر سراسری j اشاره دارد.) (مهندسی کامپیوتر ۷۳– جواب: گزینه ب)

```
Var s:array [1..3] of char;
var i,j:integer;
procedure P(x:integer;y:char);
      var j:integer;
begin
      j:=2;
      x:=x+1;
      output(y);
      output(i);
end;
s[1]:='A';
s[2]:='B';
s[3]:='C';
i:=0;
j:=1;
P(i,s[j+1]);
output(i);
```

الف)'A',1,0 ب)'B',1,1 ج)'B',0,1 د)'A',1,1

۳- برنامه زیر را در نظر بگیرید. اگر روش انتقال پارامتر به روال با نام باشد مقدار (List[1],List[2],global) بعد از اجرای برنامه به ترتیب از راست به چپ چند است؟(جواب: گزینه الف)

```
Procedure bigsub;
integer global;
integer array list[1..2];
procedur sub(param);
integer param;
begin
      param:=3;
      global:=global+1;
      param:=5;
end;
begin
      list[1]:=2;
      list[2]:=2;
      global:=1;
      sub(list[global]);
end;
```

الف)(3,5,2) ب) (1,1,2) ج) (2,3,2) د) (3,3,3)

۴-خروجی برنامه زیر در صورتی که مکانیزم تبادل پارامتر و call by value ، call by value-result ، call by reference و call by name باشد کدام گزینه است؟ (مهندسی کامپیوتر ۸۵- جواب: گزینه د)

```
Program Main;
      Var  K :integer ;
      Procedure XYZ(i,j:integer);
            Var K :integer ;
      Begin
            i:300; k:=2;
            if i=j then j:=i*k+j;
      end
begin
      k=100;
      XYZ(k,k);
      Write(k)
end;
```

	call by value	call by value-result	call by reference	call by name
الف	100	900	900	900
ب	300	100	900	6
ج	100	100	900	6
د	100	100	900	900

۵- برنامه زیر در دو حالت تبادل پارامتر بصورت by reference و by value result مفروض است. زبان برنامه تابع قواعد حوزه ایستا (static scope rule) است. خروجی برنامه کدام است؟ (مهندسی کامپیوتر ۸۷ جواب: گزینه الف)

```
Var A[1..10]:integer={1,2,3,4,5,6,7,8,9,10};
Var I,B :integer;
Procedure P(x,y,z:integer);
begin
      A[y]:=15; A[I]:=10; A[y-2]:=20; z:=1; A[b]:=19;
end
Procedure Q(x,y:integer);
begin
      x:=6*B;y:=x-26;p(x,y,B);
end
begin
      B:=5; I:=1; Q(A[I],I);
      Print(I, B, A[1], A[2], A[3], A[4], A[5]);
End
```

الف. 4, 1, 19, 20, 3, 10, 5 by ref
4, 1, 30, 20, 3, 15, 19 by value-result

ب. 4, 1, 19, 20, 3, 15, 5 by ref
4, 1, 30, 20, 3, 15, 19 by value-result

ج. 4, 1, 19, 20, 3, 10, 5 by ref
4, 1, 10, 20, 3, 30, 19 by value-result

د. 4, 1, 19, 20, 3, 15, 5 by ref
4, 1, 10, 20, 3, 30, 19 by value-result

۶-یک برنامه فرعی با پارامتر از نوع آرایه ای به طول ۱۰۰ از اعداد صحیح را در نظر بگیرید. هزینه انتقال آرایه مزبور به داخل برنامه فرعی را وقتی یکی از سه روش انتقال پارامتر مورد استفاده قرار می گیرد مقایسه کنید(مهندسی کامپیوتر ۷۵ جواب: گزینه ب)

الف) value>value-result=reference

ب) value-result>value>reference

ج) reference=value-result>value

د) value>value-result>reference

۹-۱۱-۴- زیر برنامه به عنوان پارامتر

در زبان‌های زیادی می‌توان زیر برنامه‌ها را به عنوان پارامتر به زیر برنامه دیگری فرستاد. در این حالت پارامتر واقعی شامل نام زیر برنامه و پارامتر مجازی متناظر با آن از نوع زیر برنامه است. به عنوان مثال در پاسکال زیر برنامه ای مانند Q می‌تواند شامل پارامتر R از نوع Function یا Procedur باشد.

```
Procedure Q(x:integer; function R(y,z:integer):integer);
```

با تعریف، فوق Q می‌تواند به گونه ای فراخوانی شود که پارامتر دوم آن تابع باشد مانند Q(27,fn) که Q را با تابع fn به عنوان پارامتر صدا می‌زند در داخل Q زیر برنامه ای که به عنوان پارامتر ارسال شده، می‌تواند از طریق نام پارامتر مجازی مثلاً R، فراخوانی شود. نظیر z:=R(i,x); که زیر برنامه fn را با پارامتر x,i فراخوانی می‌کند و در این حالت R(i,x) معادل fn(i,x) است اگر در فراخوانی دیگر، پارامتر واقعی تابع gn باشد، R(i,x) تابع gn(i,x) را صدا می‌زند. دو مشکل در هنگام استفاده از زیر برنامه به عنوان پارامتر وجود دارد :

- **کنترل نوع ایستا:**

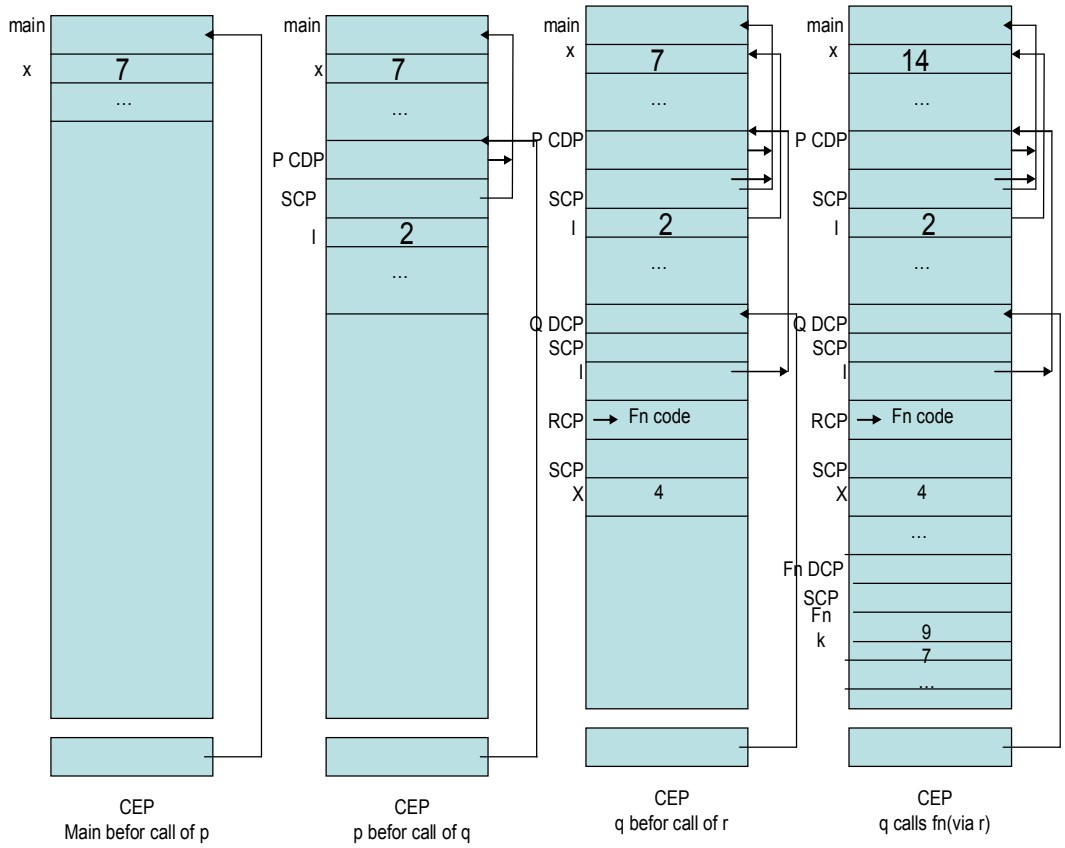
باید آرگومان‌های زیر برنامه ای که به عنوان پارامتر فرستاده می‌شود از نظر تعداد، نوع و ترتیب چک شود.

- **متغییر آزاد (ارجاع‌های غیر محلی)**

در صورتیکه به هنگام مراجعه غیر محلی هیچ مکانیزمی جهت وابستگی در تعریف زیر برنامه وجود نداشته باشد به آن متغیر، متغیر آزاد گفته می‌شود به عنوان مثال در زیر برنامه زیر به هنگام اجرای fn، به متغیرهای x,i متغیرهای آزاد گفته می‌شود.

```
Program main
var x:integer;
procedure Q(var i:integer;function R(j:integer):integer;
Var x:integer;
Begin
      x:=4;
      write('in Q before call R,i=',i,'x=',x)
      i:=R(i);
      write('in Q after call R,i=',i,'x=',x)
end;
procedure P()
var i:integer;
function FN(k:integer):integer;
begin
      x:=x+k;
      FN:=i+k;
      write('in,FN,i=',I,'k=',k,'x=',x)
end
begin
      i:=2;
      Q(x,FN);
      write('in P,i=',I,'x=',x)
end;
begin
      x:=7;
      P();
      writeln('in main,x=',x);
end
```

همان طور که گفتیم x,i متغیرهای آزاد هستند. تابع fn حاوی ارجاع‌های محلی به x,i است بر اساس قواعد حوزه ایستا در پاسکال این x، به x اعلان شده برنامه اصلی مراجعه می‌کند واین i، به i اعلان شده در زیر برنامه p مراجعه می‌کند. p تابع fn را به عنوان پارامتر Q می‌فرستد و u تابع fn را از طریق نام پارامتر مجازی R فراخوانی می‌کند. x,i در Q به صورت محلی تعریف شده اند و تابع fn نمی‌تواند به این متغیرهای محلی دستیابی داشته باشد مشکل متغیرهای آزاد فقط مخصوص پاسکال که از قاعده حوزه ایستا استفاده می‌کند نیست بلکه زبان‌هایی مانند لیسپ که از قاعده حوزه پویا استفاده می‌کند از این مشکل رنج می‌برد. برای بر طرف کردن مشکل فوق، مکانیزمی برای وابستگی استفاده می‌شود که رفتاری مشابه به آنچه fn در p فراخوانده می‌شود را داشته باشد.



شکل ۹ -۹

نکته: استفاده از زیربرنامه به عنوان پارامتر، در مواردی خوب است که بخواهیم در داخل زیر برنامه، از زیر برنامه دیگری استفاده کنیم که در این زیر برنامه تعریف نشده است بلکه در خارج از آن تعریف شده باشد. یعنی زیر برنامه ای که به عنوان پارامتر p می‌باشد، در داخل زیر برنامه p تعریف نشده است بلکه در خارج از آن تعریف شده است.

۹-۱۲- محیط‌های مشترک

گاهی اوقات لازم است مجموعه ای از اشیاء داده، بین تعدادی از زیر برنامه‌ها مشترک شود. محیط های مشترک، جهت به اشتراک گذاشتن این اشیاء داده ای استفاده می‌شوند. محیط مشترک در یک بلوک حافظه قرار می‌گیرد. هر زیربرنامه می‌تواند این محیط مشترک را اعلان کند و از این طریق، تمام اشیاء داده موجود در این بلوک (محیط مشترک) برای آن زیر برنامه قابل مشاهده خواهد بود. این محیط مشترک در زبان‌های مختلف به نام‌های متفاوت خوانده می‌شود.

Fortran (Common); C++,Smalltalk (Class);
Ada (Package); C (Extern); PL/I (External);

به عنوان مثال به Package زیر در زبان Ada توجه کنید:

```
Packageshared_table is
      Tab_size:constant integer:=100;
      Type table is array (1…tab_size)of real;
            T1,T2:Table;
            i:integer range 100 Tab_size;
end.
```

تعریف Pakege فوق، یک ثابت (Table_size)، دو جدول (T1,T2) و یک متغییر صحیح(i) تعریف می‌کند که با یکدیگر گروهی از اشیاء داده ای را تشکیل می‌دهند که در بسیاری از زیر برنامه‌ها مورد نیاز هستند.
اگر زیر برنامه ای مثل P بخواهد به داده‌های این پکیج دسترسی پیدا کند باید دستور with در زیربرنامه P به صورت زیر نوشته شود:

```
With shared_tables;
```

از این به بعد در بدنه P، هر نام موجود در پکیج مستقیماً قابل استفاده بوده و مانند آن است که بخشی از بدنه P می‌باشد. برای دستیابی به نام‌های موجود در پکیج می‌توان به صورت زیر عمل کرد:

```
Shared_tables.T1;
```

چون یک زیر برنامه ممکن است از چندین پکیج استفاده کند بنابراین برای دستیابی به نام‌های موجود در پکیج باید نام پکیج همراه با نام شیء داده مورد نیاز ذکر شود.

اشتراک صریح متغیرها:

می‌توان کاری کرد که یک شی داده در محیط محلی یک زیر برنامه، برای زیر برنامه‌های دیگر قابل دسترسی باشد. بنابراین به جای اینکه دسته ای از متغیرها در محیط مشترک و جدا از زیر برنامه‌ها باشند، هر متغیر یک مالک دارد و مالک آن زیر برنامه ای است که متغیر در آن اعلان می‌شود.
برای اینکه متغیری در خارج از زیر برنامه قابل دستیابی باشد از دستور تعریف صدور[۱] استفاده می‌شود مثل اعلان define در تکه برنامه زیر:

```
Procedure p( )
      define x,y,z;
      x,y,z:real;
      v,u:integer;
begin    …     end;
```

x,y,z برای صدور آماده می شوند.
اعلان عادی x,y,z
سایر متغیرهای محلی
دستوارت

[۱] export definition

زیر برنامه دیگری که می‌خواهد به متغیرهای مذکور دسترسی پیدا کند از تعریف وارد کردن[۱] استفاده می‌کند. مثلاً در تکه کد زیر برای استفاده از متغییرهای x و z موجود در p از دستور uses استفاده می شود.

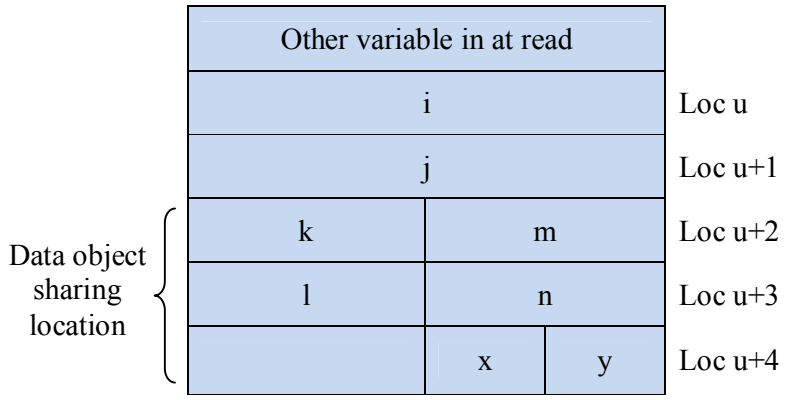
```
Procedure q( );
      uses p.x,p.z;        ──────►  x,y,z وارد می شوند
      y:integer;           ──────►  سایر اعلان‌ها
begin    …      end;       ──────►  دستورات
```

این مدل در C با استفاده از extern پیاده سازی می شود.

۹-۱۳- اعلان‌ها در بلوک‌های محلی

در زبان‌هایی مانند C می‌توان داخل هر بلوک از دستورات، یعنی پس از هر آکولاد بازی ({) متغیر تعریف کرد. این متغیرها، متغیرهای محلی آن بلوک می‌باشند و در خارج از آن بلوک شناخته شده نیستند. برای پیاده سازی این اعلان‌ها نمی‌توان برای هر بلوک، رکورد فعالیت جداگانه ای در نظر گرفت بلکه می‌توان از تکنیکی شبیه ساختار حافظه در رکوردهای طول متغیر که قبلاً شرح دادیم استفاده کرد مثال زیر این موضوع را نشان می‌دهد. در تکه برنامه زیر متغیرهای K وL همانند n وm از یک محل حافظه هستند چرا که نمی‌توانند همزمان فعال باشند و کل این حافظه توسط زیربرنامه ای اختصاص می‌یابد که آنها را در بر می‌گیرد.

```
Real proc 1(parameters)
{
      int i,j;
      …/*statement*/
      { int k,j;       …/*statement*/   }
      int m,n;
      …/*statement*/
      { int x;         …/*statement*/   }
      {int y;          …/*statement*/   }
}
```



شکل ۹ -۱۰همپوشانی حافظه متغیر در رکورد فعالیت

[۱] import definition

۹-۱۴- سوالات فصل نهم

سوالات تستی

۱- کدام گزینه صحیح است؟ (نیمسال دوم ۸۳)

الف. در روش فراخوانی با ارجاع مقدار متغیر جایگزین می‌گردد.

ب. در روش فراخوانی با مقدار آدرس متغیر جایگزین می‌گردد.

ج. انتقال پارامتر به روش مقدار و نتیجه در زبان الگول معرفی شده است.

د. در زبان C فراخوانی با ارجاع وجود ندارد.

۲- تناظر بین پارامترهای واقعی و مجازی در کدام گزینه آمده است؟ (نیمسال دوم ۸۳)

الف. بر اساس نام- موقعیتی ب. بر اساس نام - تقسیمی

ج. موقعیتی- بر اساس آدرس د. بر اساس آدرس- تقسیمی

۳- کدام گزینه جزء روش‌های انتقال پارامترها نمی‌باشد؟ (نیمسال دوم ۸۴)

الف. فراخوانی بی نام ب. فراخوانی با ارجاع

ج. فراخوانی با مقدار د. فراخوانی با نام

۴- عبارت «مجموعه ای از وابستگیها مربوط به شناسه‌هایی که در یک زیر برنامه استفاده می شود ولی هنگام ورود به آن ایجاد نمی شود» معادل کدامیک از محیطهای ارجاع زیر است؟(نیمسال اول ۸۶-۸۵)

الف.محیط ارجاع محلی ب. محیط ارجاع عمومی

ج.محیط ارجاع از پیش تعیین شده د. محیط ارجاع غیر محلی

۵-انواع فراخوانی‌ها عبارتند از:(نیمسال دوم ۸۶-۸۵)

الف.با ارجاع ب.با مقدار و نتیجه ج.با مقدار د.همه موارد.

۶- در پیاده سازی ساختارهای کنترلی بین برنامه‌ها و زیر برنامه‌ها نقش اشاره گر CEP چیست؟(نیمسال اول ۸۶-۸۷)

الف.این اشاره گر به دستور جاری قابل اجرای یک زیر برنامه اشاره می کند.

ب.این اشاره گر به ابتدای رکورد فعالیت یک زیر برنامه اشاره می کند.

ج.این اشاره گر برای پیاده سازی ارتباط ساختاری بین دو زیر برنامه استفاده می شود.

د.همه موارد فوق صحیح است.

۷- کدامیک از موارد زیر، از مولفه‌های محیط ارجاع یک زیر برنامه است؟(نیمسال اول ۸۶-۸۷)

الف.محیط ارجاع محلی ب.محیط اجرای غیر محلی

ج.محیط ارجاع از پیش تعریف شده د.همه موارد

۸- کدام یک ازموارد زیر در مورد پیاده سازی زیر برنامه‌ها صحیح نمی باشد؟(نیمسال اول ۸۶-۸۷)

الف. هر زیر برنامه که فراخوانی می شود یک activation record برای آن ایجاد می شود.

ب. به ازای هر فراخوانی جدید یک activation record جدید ایجاد می شود.

ج. امکان ندارد چندین activation record از یک زیر برنامه در برنامه وجود داشته باشد.

د. activation record نوعی شی داده است که به صورت بلوکی از حافظه نشان داده می شود.

۹- چنانچه قاعده کپی (Copy Rule) در پیاده سازی فراخوانی برگشت برای زیر برنامه‌ها در نظر گرفته شود کدامیک از موارد زیر رخ می دهد؟(نیمسال دوم ۸۶-۸۷)

الف. زیر برنامه‌ها نمی توانند بازگشتی باشند و همچنین فراخوانی نیاز به دستور فراخوانی صریح دارد.

ب. زیر برنامه‌ها در فراخوانی باید به طور کامل اجرا شوند.

ج. نمی توان زیر برنامه‌های هم روال داشت.

د. هر سه گزینه رخ می دهد.

۱۰- یک برنامه فرعی با یک پارامتر از نوع آرایه ای به طول ۱۰۰ از اعداد صحیح را در نظر بگیرید. هزینه انتقال آرایه مزبور به داخل برنامه فرعی را وقتی یکی از سه روش انتقال پارامتر مورد استفاده قرار می گیرد.در کدام گزینه صحیح مقایسه شده است؟(نیمسال دوم ۸۶-۸۷)

الف. value> value_ result=references

ب. value-result> value>reference

ج. reference= value-result> value

د.value >value_result>reference

۱۱- خروجی برنامه زیر را به صورتی که مکانیزم تبادل پارامتر call by value وcall by value- result و call by reference و call by name باشد کدام گزینه است؟(نیمسال دوم ۸۶-۸۷)

```
Program main
      Var k: integer;
      Procedure xyz (I, j: integer);
            Var k: integer;
      Begin
            I: 300; k: =2;
            If I=j then j:=I*k+j;
      End
Begin
      K=100;
      xyz (k, k);
      Write (k);
End;
```

	Call by value	Call by value_result	Call by reference	Call by name
الف	۱۰۰	۹۰۰	۹۰۰	۹۰۰
ب	۳۰۰	۱۰۰	۹۰۰	۶
ج	۱۰۰	۱۰۰	۹۰۰	۶
د	۱۰۰	۱۰۰	۹۰۰	۹۰۰

۱۲– قطعه برنامه زیر را در نظر گرفته و خروجی را بر اساس مفهوم نگهداری در فراخوانی زیر برنامه مشخص کنید؟(نیمسال دوم ۸۶–۸۷)

```
Procedure R;
begin
      .
      .
End;
Procedure Q;
      Var x: integer: =30;
Begin
      Write(x);
      R;
      x=x+1;
      Write(x);
End;
Procedure p;
begin
      .
      .
      Q;
      Q;
End;
```

الف. 33,32,31,30 ب. 30,30,30,30
ج. 31,30,31,30 د. 32,31,31,30

۱۳– تناظر بین پارامترهای مجازی و واقعی به چه روشی صورت میگیرد؟(نیمسال دوم ۸۶–۸۷)

الف.تناظر موقعیتی ب.تناظر بر اساس نام
ج.تناظر طبق نمایش درختی د.الف وب

۱۴– منظور از پارامتر واقعی در بحث فراخوانی زیر برنامه‌ها چیست؟(نیمسال دوم ۸۶–۸۷)

الف.یک شی داده که با زیر برنامه فراخوان مشترک است.

ب.یک شی داده است که ممکن است پارامترهای مجازی فراخوان باشد.

ج.یک شی داده غیر محلی می باشد که ممکن است توسط فراخوان قابل مشاهده باشد.

د. هر سه مورد فوق

۱۵– در پیاده سازی ساختارهای کنترلی بین برنامه‌ها و زیر برنامه‌ها نقش اشاره گر CIP چیست؟(نیمسال دوم ۸۶–۸۷)
الف.این اشاره گر به دستور جاری قابل اجرای یک زیر برنامه اشاره می کند.
ب.این اشاره گر به ابتدای رکورد فعالیت یک زیر برنامه اشاره می کند.
ج.این اشاره گر برای پیاده سازی ارتباط ساختاری بین دو زیر برنامه استفاده می شود.
د. همه موارد فوق صحیح است.

۱۶–کدام گزینه صحیح است؟(نیمسال دوم ۸۶–۸۷)
الف.نام مستعار مشکلاتی را برای برنامه نویس بوجود می آورد.
ب.طراحی زبان‌های جدید سعی زیادی در استفاده از نام مستعار دارند.
ج.نام مستعار مشکلاتی را برای پیاده سازی زبان بوجود نمی آورد.
د. هر سه گزینه

۱۷– اگر برای فراخوانی زیر برنامه‌ها و توابع دیدگاه قاعده کپی(Copy Rule) مطرح باشد کدام یک از موارد زیر بوجود می آید؟(نیمسال اول ۸۷–۸۸)
الف.زیر برنامه‌ها می توانند باز گشتی باشند.
ب.زیر برنامه‌های هم روال می توانند اجرا شوند.
ج.تمامی متغیرهای محلی و غیرمحلی هم نام خواهند بود.
د.پردازش استثناها امکانپذیر نمی باشد.

۱۸– در کدام زبان زیر برنامه‌ها به طور پیش فرض بازگشتی در نظر گرفته نمی شود و در صورت بازگشتی بودن از کلمه Recursive استفاده می شود؟(نیمسال اول ۸۷–۸۸)
الف.Fortran ب.PL/I ج.C د.Pascal

۱۹–تناظر بین پارامترهای واقعی و مجازی به کدام روش صورت می گیرد؟
الف. تناظر موقعیتی و تناظر بر اساس نام ب. تناظر درختی و تناظر نوع
ج. تناظر بر اساس نام و تناظر آینه ای د.تناظر نوع و تناظر ساختاری

۲۰- قطعه برنامه زیر را در نظر گرفته و خروجی را بر اساس مفهوم نگهداری در فراخوانی زیر برنامه مشخص کنید؟(نیمسال اول ۸۷-۸۸)

```
Procedure R;
begin
      .
      .
End;
Procedure Q;
      Var x: integer: =30;
Begin
      Write(x);
      R;
      x=x+1;
      Write(x);
End;
Procedure p;
      .
      .
      .
      Q;
      Q;
End;
```

الف.۳۰و۳۱و۳۲و۳۳ ب.۳۰و۳۰و۳۰و۳۰
ج.۳۰و۳۱و۳۰و۳۱ د.۳۰و۳۱و۳۱و۳۲

۲۱- کدامیک از موارد زیر پارامتر واقعی برای زیر برنامه فراخوانی شده است؟(نیمسال اول ۸۷-۸۸)

الف. شی داده محلی متعلق به فراخوان باشد یا پارامترهای مجازی فراخوان باشد.

ب. شی داده غیر محلی باشد که توسط فراخوان قابل مشاهده باشد.

ج. نتیجه ای است که توسط تابعی برگردانده شده است که زیر برنامه فراخوان آن تابع را فراخوانی کرده و نتیجه را به زیر برنامه فراخوانی شده ارسال کرده است.

د. هر سه گزینه صحیح است.

۲۲- در پیاده سازی ساختارهای کنترلی بین برنامه‌ها و زیربرنامه‌ها نقش اشاره گر CEP چیست؟(نیمسال اول ۸۷-۸۸)

الف. این اشاره گر به دستور جاری قابل اجرای یک زیر برنامه اشاره می کند.

ب. این اشاره گر به ابتدای رکورد فعالیت یک زیر برنامه اشاره می کند.

ج. این اشاره گرها برای پیاده سازی ارتباط ساختاری بین دو زیر برنامه استفاده می شود.

د. همه موارد فوق صحیح است.

۲۳- خروجی برنامه زیر به صورتی که مکانیزم تبادل پارامتر و نتایج به صورت‌های call by name, call by reference ,call by value-result, call by value باشدکدام گزینه است؟(نیمسال اول ۸۷-۸۸)

```
Program main;
      Var k: integer;
      Procedure xyz (I, j: integer);
            Var k: integer;
      Begin
            I: =300; k: =2;
            If I=j then j: =I*k+j;
      End
Begin
      K=200;
      xyz (k, k);
      Write (k);
End;
```

	Call by name	Call by reference	Call by value-result	Call by value
الف	900	900	100	200
ب	6	900	100	300
ج	6	900	100	200
د	900	900	300	100

۲۴- یک برنامه فرعی با یک پارامتر از نوع آرایه ای به طول ۱۰۰۰ از اعداد صحیح را در نظر بگیرید. هزینه فراخوانی با کدامیک از استراتژی‌های زیر بیشتر از بقیه است؟(نیمسال دوم ۸۷-۸۸)

الف.فراخوانی با مقدار ب.فراخوانی با مقدار و برگشت نتیجه

ج.فراخوانی با ارجاع د.فراخوانی با نام

۲۵- در پیاده سازی ساختارهای کنترلی بین برنامه‌ها و زیر برنامه‌ها نقش اشاره گرCIPچیست؟(نیمسال دوم ۸۷-۸۸)

الف. این اشاره گر به دستور جاری قابل اجرای یک زیر برنامه اشاره می کند.

ب. این اشاره گر به ابتدای رکورد فعالیت یک زیر برنامه اشاره می کند.

ج. این اشاره گر برای پیاده سازی ارتباط ساختاری بین دو زیر برنامه استافاده می شود.

د. همه موارد فوق صحیح است.

۲۶- در کدامیک از زبان‌های زیر آرایه‌های پارامتری وجود دارند؟(تابستان ۸۸)

الفPascal. بCobol. جAda. دC.

۲۷-کدامیک از موارد زیر جزء امتیازات goto است؟(تابستان ۸۸)

الف- توسط سخت افزار پشتیبانی می‌شود ب- درک برنامه راحت تر است

ج- تعریف رکوردها راحت تر است د- تعریف بردارها راحت تر می‌شود

۲۸- عبارت «مجموعه ای از وابستگی‌های مربوط به شناسه‌هایی که در یک زیربرنامه استفاده می‌شوند ولی هنگام ورود به آن ایجاد نمی‌شوند» معادل کدامیک از محیطهای ارجاع زیر است؟(تابستان ۸۸)

الف- محیط ارجاع محلی ب- محیط ارجاع عمومی

ج- محیط ارجاع از پیش تعیین شده د- محیط ارجاع غیر محلی

۲۹- در پیاده سازی ساختارهای کنترلی بین برنامه‌ها و فاصله زیربرنامه‌ها نقش اشاره گر CEP چیست؟(تابستان ۸۸)

الف – این اشاره گر به دستور جاری قابل اجرای یک زیربرنامه اشاره می‌کند

ب- این اشاره گر به ابتدای رکورد فعالیت یک زیربرنامه اشاره می‌کند

ج- این اشاره گر برای پیاده سازی ارتباط ساختاری بین دو زیربرنامه استفاده می‌شود.

د- این اشاره گر برای پیاده سازی قاعده کپی (Copy) استفاده می‌شود.

۳۰-می‌دانیم که محیطهای مشترک صریح برای به اشتراک گذاشتن اشیا ء داده به کار می‌رود کدامیک از زبان‌های زیر از کلاسها برای تعریف این ویژگی استفاده می‌کنند ؟(تابستان ۸۸)

الف.C++ ب. Ada

ج.Fortran د.C

۳۱- پیاده سازی قاعده تازه ترین وابستگی برای ارجاع غیر محلی توسط کدامیک از ساختمان داده‌های زیر ساده تر است؟(تابستان ۸۸)

الف. گراف ب. صف ج. پشته د. آرایه

۳۲- درکدامیک از روش‌های انتقال پارامتر با پارامترهای واقعی همانند زیر برنامه‌های فاقد پارامتر، عمل می کند؟(نیمسال اول ۸۸-۸۹)

الف.فراخوانی با نام ب.فراخوانی با ارجاع ج. فراخوانی با مقدار د. فراخوانی با مقدار-نتیجه

۳۳–برنامه زیر در زبان پاسکال را در نظر بگیرید. کدامیک از تفسیرهای زیر در مورد این برنامه درست است؟(نیمسال اول۸۸–۸۹)

```
Procedure s ;{ 1}
Begin
     Writeln ('sample1')
End;
Procedure t;
Procedure u;
Begin
     S {2}
End;
Procedure s ;{ 3}
Begin
     Writeln (sample2)
End;
Begin
     U;
End;
Begin
     T;
End;
```

الف. فراخوانی در محل۲ ، S موجود در محل ۳ را فراخوانی می کند و برنامه اجرا می شود.
ب. برنامه ترجمه نمی شود زیرا فراخوانی S در محل ۲ یک ارجاع پیشرو فاقد اعلان می باشد.
ج. برنامه ترجمه می شود و فراخوانی S در محل ۲ یک ارجاع پیشرو معتبر را ایجاد می کند.
د. فراخوانی زیر برنامه در محل ۲ زیر برنامه Sموجود در محل ۱ را فراخوانی می کند.

۳۴– کدامیک از زبانهای زیر برای رکورد فعالیت هر زیر برنامه حافظه بطور ایستا اختصاص می یابد؟ (نیمسال دوم ۸۸–۸۹)

الف.Fortran ب. ML ج. Lisp د.Ada

۳۵– پیاده سازی کدامیک از ساختارهای زبان C شبیه به ساختار حافظه رکورد متغیر است؟ (نیمسال دوم ۸۸–۸۹)

الف.اعلانها در بلوکهای محلی ب.زیربرنامه‌های همروال
ج.زیر برنامه‌های فراخوانی بازگشت د.زیر برنامه‌های بازگشتی

۳۶– کدامیک از فراخوانی‌های زیر در زبان C++ درست است؟ (نیمسال دوم ۸۸–۸۹)

الف. Q((&A+B),&B) ب. Q((A+B),&B)
ج. Q((&A+&B),&B) د. Q(&(A+B),&B)

۳۷– محیط ارجاع مربوط به نام یک پروسیجر در ساختار بلاکی ایستا، در کدام بلاک قرار می گیرد؟ (نیمسال دوم ۸۸–۸۹)

الف.بلوکی که آن بلاک را در بر می گیرد ب.محیط محلی همان بلاک
ج.بلاک برنامه اصلی د.بلاک هم سطح آن بلاک

۳۸- کدامیک از اشیاء اشاره گر زیر در رکورد فعالیت یک زیر برنامه، آدرس نقطه بازگشت دستور بعد از فراخوانی آن زیربرنامه را نگهداری می کند؟ (نیمسال دوم ۸۸-۸۹)

الف.CEP ب.CIP ج.Ip د.ep

۳۹- در زبان پاسکال، برای فراخوا نی زیر برنامه بصورت proc(v[i],I,10,20)، با توجه به روشهای انتقال پارامترها، نوع پارامتر X و Y به ترتیب چیست؟ (نیمسال دوم ۸۸-۸۹)

```
Procedure proc(arr,index : X; LB,UB: Y)
      Var temp:integer;
Begin
      For index:=LB TO UB do
            temp:=temp+arr;
      Write(temp);
End;
```

الف.X فراخوانی با نام-Y فراخوانی با مقدار ب. X فراخوانی با مقدار- Y فراخوانی با نام

ج. X فراخوانی با ارجاع- Y فراخوانی با مقدار د.X فراخوانی با مقدار ثابت- Y فراخوانی با مقدار

۴۰- روش نگهداری محیط ارجاع محلی به برنامه نویس اجازه می دهد برنامه‌هایی بنویسد که : (نیمسال دوم ۸۸-۸۹)

الف. اثرات جانبی داشته باشند ب. حساس به گذشته باشند

ج. به آرگومانهای ضمنی دسترسی داشته باشند د. برای ورودی‌های خاصی قابل تعریف نباشند

۴۱- پس از فراخوانی زیر برنامه R توسط زیر برنامه P خروجی مقدار c[m] چیست؟(از راست به چپ) (نیمسال دوم ۸۸-۸۹)

```
 R(int *i,int *j) {
      *i=*i+1;
      *j=*j+1
}
P(){
      int c[2];
      int m;
      c[1]=6; c[2]=7;
      m=1;
      R(&m,&c[m]);
      for(m=1;m<=2;m++)
            printf(%d,c[m]);
}
```

الف.۷ و ۷ ب. ۷ و ۸ ج. ۶ و ۷ د. ۶ و ۸

۴۲- در کدامیک از موارد زیر قاعده کپی صدق می‌کند. (نیمسال اول ۸۹-۹۰)

الف. زیربرنامه‌های بازگشتی مستقیم ب. زیربرنامه‌های بازگشتی غیرمستقیم

ج. همروالها د. زیربرنامه‌های فراخوانی برگشت

۴۳- در دستور x=2*y+3/z اشیا و داده ای موجود از چه روش عملوندی در عملیات استفاده می‌کنند.(نیمسال اول ۸۹-۹۰)

الف. شی داده با نام ب. انتقال مستقیم ج. انتقال غیر مستقیم د. شی داده اشاره گر

۴۴- در تکه کد برنامه زیر چه محیطهای ارجاعی وجود دارد.(نیمسال اول ۸۹-۹۰)

```
Int r;
int f(int a)
{
      int b;
      b=sqrt(a+r);
      return b;
}
int main( )
{
      f ( );
      return 0;
}
```

الف. ارجاع محلی و ارجاع غیر محلی
ب. ارجاع محلی و ارجاع عمومی
ج. ارجاع محلی و ارجاع غیر محلی و ارجاع از پیش تعریف شده
د. ارجاع محلی و ارجاع عمومیو ارجاع از پیش تعریف شده

۴۵- کدام گزینه صحیح است.(نیمسال اول ۸۹-۹۰)
الف. برای محیطهای ارجاع غیرمحلی قواعد حوزه ایستا و پویا سازگارند.
ب. برای محیطهای ارجاع محلی قواعد حوزه ایستا و پویا سازگارند.
ج. برای محیطهای ارجاع عمومی قواعد حوزه ایستا و پویا سازگارند.
د. برای محیطهای ارجاع از پیش تعریف شده قواعد حوزه ایستا و پویا سازگارند.

۴۶- کدام یک از زبان‌های زیر از روش نگهداری برای محیطهای محلی استفاده می‌کنند.(نیمسال اول ۸۹-۹۰)
الف. ادا ب. اسنوبال ۴ ج. کوبول د. لیسپ

۴۷- درکدام یک از ساختارهای زیر روشهای نگهداری و حذف پیاده سازی یکسانی دارند.(نیمسال اول ۸۹-۹۰)
الف. همروالها ب. فراخوانی – برگشت بدون بازگشتی
ج. بازگشتی د. زمان بندی شده

۴۸- کدام یک از فراخوانی‌های زیر در زبان C++ درست است.(نیمسال اول ۸۹-۹۰)
الف. Q((&A+B),&B) ب. Q((A+B),&B)
ج. Q((&A+&B),&B) د. Q(&(A+B),&B)

۴۹- محیط ارجاع مربوط به نام یک پروسیجر در ساختار بلاکی ایستا در کدام بلاک قرار دارد.(نیمسال اول ۸۹-۹۰)
الف. بلوکی که آن بلاک را دربرمی‌گیرد. ب. محیط محلی همان بلاک
ج. بلاک برنامه اصلی د. بلاک هم سطح آن بلاک

۵۰- کدام یک از موارد زیر می‌تواند یک نوع پارامتر ضمنی تلقی شود..(نیمسال اول ۸۹-۹۰)

الف. مقدار برگشتی توابع ب. مقدار برگشتی روال

ج. مقدار برگشتی ارجاع د. هرنوع مقدار برگشتی

۵۱- کدام یک از اشیا اشاره گر زیر در مورد فعالیت یک زیربرنامه ، آدرس نقطه بازگشت دستور بعد از فراخوانی آن زیربرنامه را نگهداری می‌کند..(نیمسال اول ۸۹-۹۰)

الف. CEP ب.CIP ج.ip د.Ep

۵۲- پیاده سازی اعلانها در بلاکهای محلی در زبانی مانند C شبیه به کدام ساختار زیر است. (نیمسال اول ۸۹-۹۰)

الف. رکورد متغیر ب. رکورد تودرتو ج. آرایه ای از رکورد د. زیربرنامه

۵۳- با استفاده از مفهوم نگهداری در فراخوانی زیر برنامه ها ، خروجی قطعه برنامه ذیل کدام است؟ (از راست به چپ) اولین فراخوانی زیر برنامه z می باشد . (نیمسال دوم ۸۹-۹۰)

```
Procedure  R;
.
.
End;
Procedure  P;
Var X:integer:=28;
Begin
      Write(x)
      R;
      X:=x+1;
      Write(x)
End;
Procedure  Z;
      .
      .
     P; P;
End;
```

الف. ۲۸ و ۲۹ و ۲۹ و ۳۰ ب. ۲۸ و ۲۹ و ۳۰ و ۳۱

ج. ۲۸ و ۲۸ و ۲۸ و ۲۸ د. ۲۸ و ۲۹ و ۲۸ و ۲۹

۵۴- با فراخوانی زیر برنامه ذیل به صورت $P_1(v[i],i,5,8)$ در پاسکال ، نوع انتقال پارامترهای a و b عبارتند از : (نیمسال دوم ۸۹-۹۰)

```
Procedure P1(arr , index: a ; Ib,hb:b)
Var tmp:integer;
Begin
      For index:= Ib to hb do
            Tmp:=tmp+arr
      Write(tmp);
End;
```

الف. a فراخوانی با نام و b فراخوانی با مقدار
ب. a فراخوانی با مقدار و b فراخوانی با نام
ج. a فراخوانی با ارجاع و b فراخوانی با مقدار
د. a فراخوانی با مقدار و b فراخوانی با مقدار

۵۵- کدام گزینه صحیح نیست؟ (نیمسال دوم ۸۹-۹۰)
الف. زبان prolog برای کاربردهای جستجو مورد استفاده قرار می گیرد .
ب. در زبان ML برای پیاده سازی لیست ها ، سیستم مدیریت حافظه مخفی وجود دارد .
ج. در زبان C پیاده سازی زیر برنامه های فراخوانی بازگشت، شبیه به ساختار حافظه رکورد متغیر است.
د. در زبان Fortran برای رکورد فعالیت هر زیر برنامه ، حافظه به طور ایستا اختصاص می یابد .

۵۶- در قطعه برنامه ذیل هنگامی که زیر برنامهp_2 زیر برنامه p_1 را فراخوانی می کند ، مقدار a[m] به ترتیب از راست به چپ چیست؟ (نیمسال دوم ۸۹-۹۰)

```
p1(int*i , int*j)
{
     *i=*i+1;
     *j=*j+1;
}
p2( )
{
     int a[2];
     int m ;
     a[1]=6; a[2]=7;
     m=1;
      p1(&m , &a[m]);
      for (m=1; m<=2; m++)
            Printf(%d , a[m]);
}
```

الف. ۷ و ۸ ب. ۷ و ۷ ج. ۶ و ۷ د. ۶ و ۸

۵۷- روال sum در الگول به صورت زیر نوشته می شود ، کدام گزینه در مورد آن صحیح است ؟ (نیمسال دوم ۸۹-۹۰)

```
Real procedure sum(exp, index, LB, UB);
Value LB, UB;
Real exp;
integer index , LB, UB;
Begin
      real TMP; TMP:=0;
      For index:=LB step1 until UB do
            TMP := TMP+EXP;
      Sum:=TMP;
End sum;
```

الف. فراخوانی Sum(A[I],I,1,25) فقط در صورت انتقال کلیه پارامترها به روش فراخوانی با نام لیست ، ۲۵ عنصر اول بردار A را بر می گرداند .
ب. فراخوانی Sum(A[1],I,1,25) مجموع ۲۵ عنصر اول بردار A را بر می گرداند .
ج. فراخوانی Sum(A[1],I,1,25) فقط در صورت انتقال کلیه پارامترها به روش فراخوانی با نام مجموع ۲۵ عنصر اول بردار A را بر می گرداند .
د. فراخوانی Sum(A[1],I,1,25) فقط در صورت انتقال کلیه پارامترها به روش فراخوانی با ارجاع مجموع ۲۵ عنصر اول بردار A را بر می گرداند .

۵۸- کدام گزینه در مورد محیط ارجاع یک روال صحیح است ؟ (نیمسال دوم ۸۹-۹۰)
الف. محیط ارجاع مربوط به نام یک روال در ساختار بلاکی ایستا ، در بلاک برنامه اصلی قرار دارد .
ب. محیط ارجاع مربوط به نام یک روال در ساختار بلاکی ایستا ، در بلاکی قرار دارد که آن را در گرفته است .
ج. محیط ارجاع مربوط به نام یک روال در ساختار بلاکی ایستا ، در محیط محلی همان بلاک قرار دارد .
د. محیط ارجاع مربوط به نام یک روال در ساختار بلاکی ایستا ، در بلاک بلافاصله همسطح آن قرار دارد .

۵۹- کدام گزینه در مورد محیط های مشترک (مجموعه ای از اشیا داده که بین تعدادی زیر برنامه مشترک است) صحیح نیست؟ (نیمسال دوم ۸۹-۹۰)
الف. اعلان محیط مشترک به عنوان اسامی محلی زیر برنامه ها در جدول نمادها ذخیره می شود تا زیر برنامه به آنها مراجعه کند .
ب. بلوک حافظه مربوط به محیط مشترک تا زمانی در حافظه است که امکان فراخوانی زیر برنامه هایی که از آن استفاده می کنند وجود دارد .
ج. جهت رجوع به اشیاء داده موجود در بلوک مشترک ، آدرس پایه بلوک ، باید برای زیر برنامه مراجعه کننده مشخص باشد .
د. محیط مشترک شامل تعریف متغیرها ، ثوابت و انواع داده و تعریف زیر برنامه ها ، است .

۶۰-کدامیک از موارد زیر در مورد قاعده کپی در فراخوانی زیربرنامه ها صدق نمی کند؟(نیمسال اول ۹۱-۹۰)
الف) فراخوانی زیربرنامه نیاز به دستور فراخوانی صریح دارد
ب) زیربرنامه ها در هر فراخوانی باید به طور کامل اجرا شوند
ج) زیربرنامه ها می توانند بازگشتی باشند
د) در هر زمان فقط یک زیربرنامه کنترل را در دست دارد

۶۱- در زمان فراخوانی زیربرنامه ها، نقطه برگشت در رکورد فعالیت چه اطلاعاتی را ذخیره می کند؟(نیمسال اول ۹۱-۹۰)
الف) آدرس نقطه برگشت به برنامه را بعد از فراخوانی زیربرنامه ذخیره می کند
ب) اشاره گر دستور (IP) و اشاره گر محیط فعلی (EP) را ذخیره می کند.
ج) دستور بعد از فراخوانی زیربرنامه را ذخیره می کند
د) کلیه اطلاعات لازم برای فراخوانی و برگشت از زیربرنامه را ذخیره می کند.

۶۲-برنامه زیر مفروض است. با در نظر گرفتن قواعد حوزه پویا، خروجی برنامه زیر کدام است؟(از چپ به راست) (نیمسال اول ۹۱-۹۰)

الف)۲۰و ۵ و ۳ و ۲۰ ب)۲۰ و ۵ و ۲۰ و ۵ ج)۵ و ۲۰ و ۳ و ۲۰ د)۳ و ۵ و ۲۰ و ۵

```
Main()
{
            int x,y;
            x=20;
            y=3;
           Sub1()
                 {
                          Sub3()
                              {
                                 Cout<<x,y;
                               }
                          Sub2(int y)
                            {
                                 int x;
                                 x=5;
                                 Sub3();
                                 Cout<<x,y;
                             }
                            Sub2(x);
                  }
     Sub1();
 }
```

۶۳-کدامیک از زبان های زیر قاعده حوزه پویا را به روش حذف پیاده سازی می کنند؟(نیمسال اول ۹۱-۹۰)

الف)کوبول و پاسکال ب)فرترن و لیسپ ج)C و پاسکال و ادا د)کوبول

۹-۱۵- پاسخنامه سوالات تستی فصل نهم

سوال	الف	ب	ج	د
۱			*	
۲	*			
۳	*			
۴				*
۵				*
۶		*		
۷				*
۸			*	
۹				*
۱۰		*		
۱۱				*
۱۲				*
۱۳				*
۱۴				*
۱۵	*			
۱۶	*			
۱۷				*
۱۸		*		
۱۹	*			
۲۰				*
۲۱				*
۲۲	*			
۲۳	*			
۲۴		*		
۲۵	*			
۲۶				
۲۷	*			
۲۸				*
۲۹		*		
۳۰	*			

سوال	الف	ب	ج	د
۳۱			*	
۳۲	*			
۳۳		*		
۳۴	*			
۳۵	*			
۳۶		*		
۳۷	*			
۳۸			*	
۳۹	*			
۴۰		*		
۴۱	*			
۴۲				*
۴۳		*		
۴۴				*
۴۵		*		
۴۶			*	
۴۷		*		
۴۸		*		
۴۹	*			
۵۰	*			
۵۱			*	
۵۲	*			
۵۳		*		
۵۴	*			
۵۵			*	
۵۶		*		
۵۷		*		
۵۸		*		
۵۹				*
۶۰			*	
۶۱		*		
۶۲		*		
۶۳			*	

سوالات تشریحی

۱- نحوه پیاده سازی زیر برنامه‌های بازگشتی در زبانهای برنامه سازی را بطور کامل شرح دهید. (نیمسال اول ۸۵-۸۶)

۲- پارامترهای واقعی و مجازی را با یکدیگر مقایسه کنید. چه تناظرهایی بین این دو پارامترها امکانپذیر است. (نیمسال اول ۸۵-۸۶)

۳- خروجی برنامه زیررا در حوزه ارجاعی ایستا و پویا بدست آورید. (نیمسال اول ۸۶-۸۷)

```
Program main;
      Var x, y: integer;
      Procedure p1;
      Begin
            Writeln(x, y);
      End;
      Procedure p2;
            Var x, y: integer;
      Begin
            X=20;
            Y=45;
            Writeln(x, y);
            P1;
      End;
Begin
      X=2;
      Y=4;
      P2;
End.
```

۴- برنامه زیر را در نظر گرفته و محیطهای ارجاع localو non-local را برای main,sub1,sub2 بنویسید؟(نیمسال دوم ۸۶-۸۷ و نیمسال دوم ۸۷-۸۸)

```
Program main
      Var a, b, c: real;
      Procedure sub1 (a: real);
            Var d: real;
            Procedure sub2(c: real);
                  Var d: real;
            Begin
                  Statements
                  C: =c+b;
                  Statements
            End;
      Begin
            Statements
            Sub2 (b)
            Statements
      End
Begin
      Statements
      Sub1 (a);
```

```
      Statements
End.
```

۵- برنامه زیر را در نظر بگیرید و مراحل اجرای این برنامه را در هر یک از زمانهای زیر در پشته مرکزی نشان دهید؟(نیمسال دوم ۸۶-۸۷ و نیمسال اول ۸۷-۸۸)

```
Program main;
      Var x: integer;
      Procedure q (Var i: integer; function r ((j: integer): integer);
            Var x: integer;
      Begin
            X:=4;
            Write ("in q, before call of r, i=", I,"x=", x);
            I: r (I);
            Write ("in q, after of r, i=", I,"x=", x)
      End
      Procedure p;
            Var I: integer;
            Function FN (k: integer): integer;
            Begin
                  X: =x+k;
                  FN=i+k;
                  Write ("in p, I=", I,"k=", k,"x=", x)
            End
      Begin
            I:=2;
            Q(x, FN);
            Write ("in p, i=", I,"x=", x)
      End;
Begin
      X: =7;
      P;
      Write ("in main=", x)
End;
```

الف.اجرای main قبل از فراخوانی P　　ب.اجرای P قبل از فراخوانی Q

ج.اجرای Q قبل از فراخوانیR　　د.Q, FV را صدا می کند.

۶- پدیده نام مستعار را به همراه یک مثال شرح دهید؟(نیمسال اول ۸۷-۸۸)

۸- اعلان پیشرو در پاسکال ناهنجاری بوجود می آورد آن را به همراه یک مثال شرح دهید؟(نیمسال اول ۸۷-۸۸)

۹- زیر برنامه ای به صورت زیر اعلان‌هایی در بلوک‌های محلی دارد نمایش حافظه مربوطه به این زیر برنامه را برای متغیرهای مربوطه به گونه ای رسم کنید که در رکورد فعالیت بدون هیچ مشکلی عملیات فراخوانی را با کمترین حافظه مصرفی داشته باشیم؟(نیمسال دوم ۸۷-۸۸)

```
Float proc1 (parameter) {
Int j;
{Int k, l ;}
{Int m, n ;}
{Int x ;}
{Int y ;}
```

۱۱- خروجی برنامه زیر را در هر یک از حالت‌های ارسال پارامتر با مقدار مقدار نتیجه و ارجاع مشخص کنید؟(نیمسال اول ۸۸-۸۹)

```
Program s;
      Var x, y, j: integer;
      Procedure t(y, z: integer);
      Begin
            Z:=z-5;
            Y:=y+5;
            X:=x-y;
      End;
Begin
      X:=3;
      Y:=4;
      T(x, y);
      Writeln(x, y);
End;
```

۱۲- خروجی شبه کد زیر را در حوزه ایستا و پویا مشخص کنید؟(نیمسال اول ۸۸-۸۹)

```
Begin
      Int x: =2;
      Procedure p ();
      Begin
            Write(x);
      End;
      P ();
      Begin
            Int x: =3;
            P ();
      End;
      P ();
End;
```

۱۳- پدیده نام مستعار را به همراه یک مثال شرح دهید؟(نیمسال اول ۸۸-۸۹)

۱۴- اشتراک داده از طریق محیط مشترک صریح را بطور کامل مطرح کنید. (نیمسال دوم ۸۸-۸۹)

۱۵- خروجی برنامه زیر را در حین اجرا به دو صورت: (نیمسال دوم ۸۸-۸۹)

الف.هنگامی که حوزه ایستا است، مشخص کنید.

ب. هنگامی که حوزه پویا است، مشخص کنید.

```
Program main;
      Var i,a,k,m :integer;
      Procedure Q(m:integer; var i:integer);
      Begin
            i:=i+k;    m:=a+2;
            Writeln('in Q:', i,a,k,m);
      End;
      Procedure P(a:integer; var i:integer);
            Var k:integer;
      Begin
            k:=4 i:=i+k; a:=a+k;
            Q(a,i);
      End;
Begin {main}
      i:=1; a;=2; k:=3;
      P(k,i);
      Writeln('in main:', i,a,k);
End.
```

۱۶- اشتراک داده از طریق حوزه ایستا را به طور کامل و با مثال تشریح کنید.(نیمسال اول ۸۹-۹۰)

۱۷- خروجی برنامه زیر برای حوزه های ارجاعی ایستا و پویا چیست ؟ (۱ نمره) (نیمسال دوم ۸۹-۹۰)

```
Program main
Var x,y:integer;
Procedure P1;
Begin
     Writln(x,y);
End;
Procedure P2;
Var X,Y:integer;
Begin
      X:=12;
      Y:=14;
      Writln(x,y);
      P1;
End;
Begin
     X:= 2;
     Y:=4;
      P2
End;
```

۱۸- با توجه به برنامه ذیل ، محیط های ارجاع محلی و غیر محلی را برای main و p_1 و p_2 مشخص کنید. (نیمسال دوم ۸۹-۹۰)

```
Program main
Var x,y,z:Real;
Procedure p1(x:real);
Var Q:real;
Procedure p2(z:real);
Var Q:real;
Begin
    Statement 1;
    Statement 2;
    Z:=z+y;
    Statement 3;
End;
Begin
    Statement 4;
    P2(y);
    Statement 5;
End;
Begin
    Statement 6;
    P1(x);
    Statement 7;
    Statement 8;
End;
```

۹-۱۶- پیوست: سوالات کنکور سراسری کارشناسی ارشد

سال ۱۳۸۲

۱- مجموعه مقادیر ممکن برای یک متغیر از نوع صحیح (integer) در چه مرحله ای تعیین می گردد؟

الف.ترجمه برنامه‌ها (Translation of programs)

ب.پیاده سازی زبانها (Implementation of languages)

ج.اجرای برنامه‌ها (Execution of programs)

د.تعریف و تبیین زبانها (Definition of languages)

۲-در زبان فرضی زیر، آرگومانهای برنامه‌ها بصورت Call by Name تعریف شده اند. با توجه به این روش تعریف آرگومان، خروجی تکه برنامه زیر چیست؟

```
Procedure Exchange(x,y :integer);
      Var temp:integer;
Begin
      Temp←x;
      Y←temp;
end;
.
.
.
I←4;
A[1]←8; A[1]←6; A[1]←4; A[1]←2;
Exchange(I,A[I]);
Output(I,A[1], A[2], A[3], A[4]);
```

الف.۲و۴و۸و۲ (از چپ به راست بخوانید.)

ب.۴و۴و۶و۸و۲ (از چپ به راست بخوانید.)

ج.۲و۴و۶و۸و۴ (از چپ به راست بخوانید.)

د.۲و۴و۴و۸و۴ (از چپ به راست بخوانید.)

۳- قطعه کد زیر(به زبان C)را در نظر بگیرید:

```
int *p;
int *q;
q ="abcd"
p =malloc(10);
q =p;
free(p);
strcpy(q,"123");
```

در این صورت می توان گفت:

الف. در اجرای این قطعه کد، ارجاع معلق (Dangling Reference) بوجود می آید.

ب. در اجرای این قطعه کد ، Garbage بوجود می آید.

ج. در اجرای این قطعه کد ، Fragmentation بوجود می آید.

د. در اجرای این قطعه کد، Garbage و ارجاع معلق (Dangling Reference) بوجود می آید.

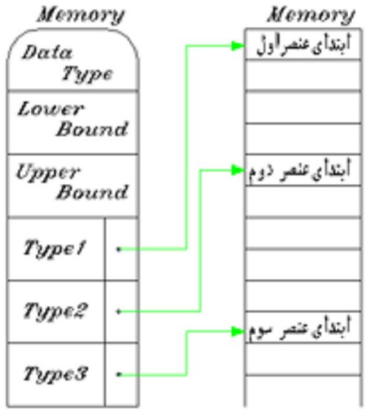
۴- با فرض اینکه شکل زیر نمایش حافظه مربوط به پیاده سازی داده ساختیافته ای در یک زبان برنامه سازی را نشان دهد که در زمان اجرا نیز بتوانیم نوع هر یک از عناصر آن را تغییر دهیم، کدامیک از گزینه‌های زیر صحیح است؟

الف.نقطه ضعف اساسی و مهم این داده‌ها ساختیافته مکانیزم لازم برای انتخاب عناصر آن است به طوریکه عملاً آن را ناکارامد می سازد.

ب.پیاده ساز زبان با انتخاب این روش اولاً انعطاف پذیری بالایی برای برنامه ساز حاصل میکند و ثانیاً قید صریح نوع هر عنصر قابلیت اعتماد زبان مورد نظر را نیز بالا می برد.

ج. در برخی از زبان‌های برنامه سازی مثلSNOBOL4 آرایه‌ها را، عمدتاً با هدف بالا بردن خاصیت انعطاف پذیری (Flexibility) برای برنامه ساز، با استفاده از این روش پیاده سازی می کنند.

د.اگر این شکل نمایش حافظه ای مربوط به پیاده سازی آرایه خاصی باشد و نوع عناصر آن را در زمان اجرا بتوان عوض کرد آنگاه وجود حد پایینی و بالایی (Lower & upper Bound) در آن لزوماً بی معنی است.



۵- کدامیک از معیارهای زیر برای انتخاب زمان مناسب در کاربردهای توکار(Embedded Systems) اهمیت بیشتری دارد؟

الف.یکنواختی (Uniformity) ب.قابلیت توسعه (Extensibility)

ج.عمومیت (Generality) د.قابلیت اطمینان (Reliability)

سال ۱۳۸۳

۶- در رابطه با مفهوم آزمون نوع (Type checking) کدام گزینه غلط است؟

الف. وجود یک شی داده ای می تواند جزئی از عمل آزمون نوع آن تلقی شود.

ب. آزمون نوع یعنی تعداد و نوع آرگومانهای عملیات در برنامه درست باشند.

ج. آزمون نوع پویا منجر به استفاده بهینه از حافظه و افزایش سرعت اجرا می شود در حالیکه آزمون نوع ایستا نتیجه عکس دارد.

د. در زبانهایی که اعلان صریح (explicit declaration) اجباری نیست ولی با استفاده از مکانیزم نتیجه گیری نوع (type inference) می توان نوع عملوندها را تعیین کرد، آزمون نوع ایستا ممکن است.

۷– در رابطه با مفهوم مقیدسازی (Binding) و زمانهای مقیدسازی،کدام گزینه غلط است؟
الف.مقیدسازی ممکن است در زمان اجرا، ترجمه، پیاده سازی زبان و یا تعریف زبان صورت پذیرد.
ب.در زبانهای با مقیدسازی زودرس، قابلیت انعطاف بیشتر و در زبانهای با مقیدسازی دیردرس کارایی اجرا بهتر است.
ج.مقیدسازی یک عنصر برنامه به یک صفت خاص، به معنی انتخاب یک صفت از مجموعه ای از صفات است.
د. دریک زبان با مقیدسازی زودرس (Early Binding) تقریبا تمام مقیدسازی‌ها در زمان ترجمه انجام می شود و دریک زبان با مقیدسازی دیررس (Late Binding) تقریبا تمام مقیدسازی‌ها در زمان اجرا انجام می شود.

۸–کدامیک از خواص زیر در یک زبان برنامه سازی اصلی ترین نقش را در قابلیت انتقال برنامه‌ها به عهده دارد؟
الف. Encapsulation ب.Abstraction
ج.Information Hiding د.Explicit Declaration

۹–برای ساختن یک درخت تصمیم گیری (Decision tree) به منظور انتخاب یک دستور (statement) از میان n دستور کدامیک از ساختارهای برنامه سازی زیر می تواند منجر به کارایی زمان اجرا بصورت O(1) شوند؟ فرض کنید شرط انتخاب یک دستور، برابر یک عبارت با یک مقدار ثابت به ازای هر دستور باشد.
الف.ساختن درخت با دستور if – then – else
ب.بدست آوردن کارایی O(1) امکان ندارد.
ج.ساختن درخت با دستور case یا switch
د. ساختن درخت با دستور if – then – else – endif و elsif در زبان Ada

۱۰–کدامیک از موارد زیر به نوعی از inheritance نزدیکتر است؟
الف.Late binding ب.Static scope rule
ج.Strong typing د.History sensitivity of methods

۱۱– از میان زبانهای زیر کدامیک کلی ترین نوع چند ریختی (polymorphism) را ارائه می دهند؟
الف.LISP ب.Ada ج. C++ د.M

۱۲– در کدامیک از روشهای انتقال پارامتر، به ازای هر پارامتر از نوع متغیر ساده بیش از یک فیلد در رکورد فعالیت (activation record) برنامه فرعی لازم است در نظر گرفته شود؟
الف.by result ب.by value
ج.by reference د.هیچگاه نیاز به بیش ازیک فیلد نیست.

سال ۱۳۸۴

۱۳–کدامیک از مفاهیم زیر کمتر با هم سازگارند؟
الف.تعریف نوع متغیرها و ترجمه ب.Late binding و تفسیر
ج.Static Scope Rule و ترجمه د.Early binding و Dynamic Scope Rule

۱۴– در کدامیک از موارد زیر مفهوم Encapsulation در یک زبان بصورت کامل رعایت نشده است؟

الف. فقط با برنامه نویسی به آن زبان بتوان یک نوع داده ای مثل یکی از انواع داده ای موجود (مثلا integer) رابا یک نمایش حافظه ای خاص دلخواه برنامه نویس و overload کردن عملیات موجود روی integerهاپیاده سازی کرد.

ب. فقط با برنامه نویسی به همان زبان سطح بالا بتوان سطری یا ستونی بودن نمایش حافظه ای زمان اجرای آرایه‌های دو بعدی را کشف کرد.

ج. فقط با برنامه نویسی به همان زبان بتوان یکی از عملیات (مثلا ضرب)، که در اصل آن زبان با اپراتور مشخص (مثلاً *)وجود دارد، را بدون استفاده از آن اپراتور شبیه سازی کرد.

د. در همه موارد فوق Encapsulation بطور کامل رعایت شده است.

۱۵– اگر محدوده ی اعتبار (Scope) نام یک زیر برنامه فقط شامل محدوده ی خود آن زیر برنامه باشد و قاعده ی حاکم بر زبان Static Scope rule باشد ، چه اشکالی پیش می آید؟

الف. هیچ اشکالی پیش نمی آید.

ب. برنامه اصلی نمی تواند آن زیربرنامه را صدا زند.

ج. هیچگاه از داخل یک زیربرنامه ی در حال اجرا نمی شود زیر برنامه ی دیگری را صدا زد.

د. وقتی یک زیربرنامه در حال اجراست نمی تواند خودش را بصورت بازگشتی صدا زند.

۱۶–در پیاده سازی کدامیک از زبان‌های زیر می توان برای دسترسی به مقدار یک متغیر ساده غیر محلی مستقیماً از مکانیزم سخت افزاری Fetch operand تعبیه شده در اجرای یک instruction استفاده کرد؟

الف.فقط C ب.C,LISP ج.,C پاسکال د.C، پاسکال ، LISP

۱۷– در یک پیاده سازی از یک زبان برنامه سازی که تمام تکنیک‌های انتقال پارامتر را پشتیبانی می کند متوجه می شویم که پیاده سازی تکنیک by reference درست کار نمی کند. می خواهیم در برنامه خود با استفاده از سایر تکنیک‌های انتقال پارامتر، تکنیک by reference را شبیه سازی کنیم بطوریکه نتیجه ی اجرای برنامه شبیه سازی شده در حالت کلی کاملاً با نتیجه اجرای برنامه اصلی (اگر by reference درست کار می کرد) یکسان باشد، کدام گزینه درست است؟

الف. این شبیه سازی با تکنیک‌های دیگر امکان ندارد.

ب. انتقال by reference پارامترهای عضوی از آرایه را می توان با انتقال by value result شبیه سازیکرد.

ج. انتقال by reference پارامترهای متغیر ساده را می توان با انتقال by name شبیه سازی کرد.

د. انتقال by reference پارامترها به هر شکلی که باشند را می توان با انتقال by value result شبیه سازی کرد.

۱۸– اگر بخواهیم خطای مقدار اولیه نداشتن یک متغیر تعریف شده در داخل یک زیربرنامه را در زمان اجرا کشف کنیم، هزینه زمان اجرا خیلی زیاد می شود. اگر بخواهیم این خطا را در زمان ترجمه کشف کنیم هزینه کشف آن در زمان ترجمه در دو زبان C و پاسکال چگونه مقایسه می شود؟

الف. هزینه در زبان C کمتر است. ب. هزینه در زبان پاسکال کمتر است.

ج. نمی شود این خطا را در زمان ترجمه کشف کرد. د. هزینه در هر دو زبان تقریباً یکسان است.

سال ۱۳۸۵

۱۹–خروجی برنامه زیر در صورتی که مکانیزم تبادل پارامتر و call by value ، call by value–result ، call by reference و call by name باشد کدام گزینه است؟

```
ProgramMain;
      Var  K :integer ;
      Procedure  XYZ(i,j:integer);
            Var K :integer ;
      Begin
            i:300;  k:=2;
            if i=j then j:=i*k+j;
      End
begin
      k=100;
      XYZ(k,k);
      Write(k)
end;
```

	call by value	call by value–result	call by reference	call by name
الف	100	900	900	900
ب	300	100	900	6
ج	100	100	900	6
د	100	100	900	900

۲۰– در بعضی از زبانها با حوزه ایستا مانند پاسکال می توان اسم یک روال (procedure) مانند F را بصورت یک پارامتر به یک روال دیگر ارسال کرد. برای binding اسامی درون بدنه روال ارسال شده کدامیک از روشهای زیر مناسب تر است؟

الف. binding در محیط تعریف F　　ب. binding در محیط ارسال پارامتر به F

ج. binding در محیط فراخوانی F　　د. هیچکدام

۲۱– کدامیک از زبانهای زیر جزو زبانهای Early Binding محسوب می شود؟

الف. LISP　　ب.Java　　ج. Smalltalk　　د. هیچکدام

۲۲– خروجی برنامه زیر در حالتی که از قواعد static scoping و dynamic scoping استفاده شود، به ترتیب کدام گزینه است؟

```
Program Main ;
      var M :integer
      Function   F(X:integer) :integer;
      Begin
            F:=X*20
      end;
      Procedure  P(I:integer);
            var   Z:integer;
      begin
            Z:=F(I)*M;
            write(Z)
      end
      Procedure  Q;
            var   K :integer;
                  M :integer;
            Function  F(Y :integer):integer;
            begin
                  F :=Y*30
            end
      begin
            M :=3;
            K :=10;
            P(K)
      end
begin
      M:=2;
      Q;
end.
```

الف400 :dynamic scoping: 600 , static scoping

ب. dynamic scoping: 900 , static scoping: 400

ج. dynamic scoping: 900 , static scoping: 600

د. dynamic scoping: 400 , static scoping: 900

۲۳– در یک زبان برنامه سازی مثل پاسکال تعریف برنامه‌های فرعی در داخل برنامه‌های فرعی دیگر مجاز است.برنامه‌های فرعی C,B,Aو D در عمق‌های مختلف برنامه فرعی M تعریف شده اند. فرض کنید زنجیره callها بصورت زیر باشد:-M >A->B->A->C->D و D بتواند از متغیر x که در A تعریف شده است بر اساس static scope rule استفاده کند.در آن صورت تودرتویی static برنامه‌های فرعی C,B,A وD چند حالت می تواند داشته باشد؟

الف. ۱۰ ب. ۸ ج. ۴ د. ۲

۲۴– دوره ی حیات (life time) شیء داده ای A که در یک تابع، مثلا بصورت int A ، تعریف شده است کدام است؟

الف.از زمان شروع اجرای تابع تا زمان پایان اجرای تابع

ب. از زمان اجرای دستورالعمل int A تا زمان پایان اجرای تابع

ج.از زمان شروع اجرای برنامه اصلی تا پایان اجرای برنامه اصلی

د. از زمان مقدار اولیه گرفتن شیء داده ای A تا زمان آخرین استفاده از آن

سال ۱۳۸۶

۲۵- در زبانهایی که اشاره گر (Pointer) ندارند در چه حالتی وقوع پدیده ی همنامی یا نام مستعار(Aliasing) امکان ندارد؟

الف.اگر تنها تکنیک انتقال پارامتر در زبان مورد نظر by value باشد.

ب.اگر مجموعه متغیرهای سراسری (global) برنامه تهی باشد.

ج.اگر زبان دارای تکنیک انتقال by reference نباشد.

د.هیچکدام از سه گزینه فوق صحیح نیستند.

۲۶- زبانهای زیر را از نظر تفسیری یا کامپایلری بودن دسته بندی کنید. دسته بندی درست را انتخاب کنید.

Ada-VI , Small talk-V , LISP-IV , C++- III , Java-II , Fortran-I

الف. VI , V , IV ب. V , IV , II

ج. III.II.I د.هیچکدام

۲۷- در کدامیک از موارد زیر اگر تکنیک انتقال پارامتر by name یا by reference باشد نتیجه اجرای برنامه می تواند متفاوت باشد؟

الف.پارامتر مربوطه متغیر ساده (مثلاً A) است.

ب.پارامتر مربوطه عضو نامعینی از یک آرایه است. (مثلاً B[I]).

ج. پارامتر مربوطه عضو معینی از یک آرایه است. (مثلاً B[5])

د. هرسه مورد

۲۸- برای ۳ دستور case به شرح زیر، پیاده سازی معروف به جدول پرش‌ها (Jump table) مفروض است؟

I, case I of ۱:st_1; ۲:st_2; ۳:st_3 end case

II, case I of ۱۰۰:st_1; ۲۰۰:st_2; ۳۰۰:st_3 end case

III, case I of ۱:st_1; ۲:st_2; ۳۰۰:st_3 end case

الف. حجم کد I از II وII از III کمتر است.

ب. سرعت اجرا و حجم کد هر سه دستور برابر است.

ج. سرعت اجرای دستور I ازIII بیشتر و سرعت اجرای III از II بیشتر است.

د. هیچکدام

۲۹- در زبانی که قانون حوزه ی شناسایی ایستا حاکم است و تعریف تودرتوی برنامه‌های فرعی (nested definition) ممکن است، برنامه ای نوشته ایم که از یک برنامه ی اصلی حاوی تعریف دو برنامه فرعی غیر تودرتو تشکیل شده است. یکی از برنامه‌های فرعی حاوی برنامه فرعی دیگری است که آن هم حاوی برنامه فرعی دیگری است. در یک لحظه از زمان اجرای داخلی ترین برنامه فرعی، پشته (stack) رکوردهای فعالیت برنامه‌های فرعی، حاوی ۱۰ رکورد (از جمله رکورد برنامه اصلی) است.در این لحظه محتوای چند رکورد فعالیت قابل دسترسی توسط برنامه ی در حال اجرا است؟

الف. ۳ ب. ۴ ج. ۱۰ د. هیچکدام

۳۰- در قطعه برنامه ی زیر که به زبان برنامه سازی ML نوشته شده است مقدار عبارت b*f(a,a) در دو حالتی که زبان از قواعد حوزه ایستا (static scoping) و حوزه پویا (dynamic scoping) استفاده می کند، کدام است؟

```
let a = ۵,   b =۱۰ ,  c = ۷ in
      let val fun f(x,y) = a*(x+y) + b
            let  val a = ۴ , b = ۲  in
                  b * f(a,a)
            end
      end
end
```

الف. حوزه ایستا: ۶۰ و حوزه پویا :۳۴ ب. حوزه ایستا: ۵۰۰ و حوزه پویا :۳۴۰

ج. حوزه ایستا: ۱۰۰ و حوزه پویا :۶۸ د.هیچکدام

سال ۱۳۸۷

۳۱- کدام مجموعه از گزینه‌های زیر شامل عبارتهای صحیح از مجموعه عبارتهای زیر است؟

۱. مجموعه مقادیری که یک متغیر از نوع integer می تواند اختیار کند معمولا در زمان پیاده سازی زبان تعیین می شود.

۲.ماشین مجازی زبان برنامه سازی X، ماشینی است که برنامه به زبان X را اجرا می کند.

۳.بررسی ایستای نوع باعث افزایش مصرف حافظه و کاهش سرعت اجرای برنامه می شود. ولی بررسی پویای نوع باعث کاهش مصرف حافظه و افزایش سرعت اجرای برنامه می شود.

الف. ۱ و ۲ ب. ۱ و ۳ ج. ۲ و ۳ د. ۱ و ۲ و ۳

۳۲- این برنامه C را در نظر بگیرید:

```
void fun1(void);
void fun2(void);
int a =1, b =2, c=3;
int main ( ){
      int c=4;
      fun1( );
      return( );
}
void fun1( ){
      int a=2, b=3;
      fun2( );
}
void fun2( ){
      printf("%d %d %d\n",a, b, c);
}
```

الف.3 2 1 در حوزه پویا و 4 3 2 در حوزه ایستا

ب. 3 3 2 در حوزه پویا 3 2 1 در حوزه ایستا

ج. 4 3 2 در حوزه پویا و3 2 1در حوزه ایستا

د. 4 2 1 در حوزه پویا و 4 2 1 در حوزه ایستا

۳۳- اصطلاحات زیر مفروضند. گزینه ای را انتخاب کنید که اصطلاحات مربوط به آن یکدیگر را تداعی کنند یا به عبارت دیگر از جنبه اثباتی به هم مرتبط باشند.

۱.Early binding ۲.Interpretation ۳.Late binding
۴.Translation ۵.Execution

الف. ۱ و۲ ب. ۱ و۴ ج. ۱ و۵ د. ۳ و۴

۳۴- در مورد Static Type Checking (STC) کدامیک از گزینه‌های زیر غلط است؟

الف. اگر STC نباشد حجم کد تولید شده در اثر ترجمه برنامه خیلی زیاد می شود.
ب. STC باعث می شود که نوع عملیات پلی مرفیک در زمان کامپایل معین شود.
ج. STC باعث می شود تابسیاری از خطاهای برنامه در زمان کامپایل کشف شود.
د. STC فقط برای قسمت تعاریف برنامه انجام می شود و به قسمت اجرایی برنامه کاری ندارد.

۳۵- برنامه زیر در دو حالت تبادل پارامتر بصورت by reference و by value result مفروض است. زبان برنامه تابع قواعد حوزه ایستا (static scope rule) است. خروجی برنامه کدام است؟

```
Var A[1..10]:integer={1,2,3,4,5,6,7,8,9,10};
Var I,B :integer;
Procedure P(x,y,z:integer);
begin
     A[y]:=15; A[I]:=10; A[y-2]:=20; z:=1; A[b]:=19;
end
Procedure Q(x,y:integer);
begin
     x:=6*B;y:=x-26;p(x,y,B);
end
begin
     B:=5; I:=1; Q(A[I],I);
     Print(I, B, A[1], A[2], A[3], A[4], A[5]);
End
```

الف. 4, 1, 19, 20, 3, 10, 5 by ref
4, 1, 30, 20, 3, 15, 19 by value-result

ب. 4, 1, 19, 20, 3, 15, 5 by ref
4, 1, 30, 20, 3, 15, 19 by value-result

ج. 4, 1, 19, 20, 3, 10, 5 by ref
4, 1, 10, 20, 3, 30, 19 by value-result

د. 4, 1, 19, 20, 3, 15, 5 by ref
4, 1, 10, 20, 3, 30, 19 by value-result

سال ۱۳۸۸

۳۶–در هنگام اجرای یک برنامه هرگاه سرریز (overflow) عمل جمع رخ دهد، قطعه کدی که ترجمه ی یک روال exception handling است و توسط برنامه نویس به زبان سطح بالا نوشته شده است، اجرا می شود. چه عواملی از میان عوامل زیر در کشف، تولید و فعال کردن قطعه کد مربوطه می توانند دخیل باشند؟

۱.برنامه نویس ۲. کامپایلر ۳. سیتم عامل ۴. سخت افزار

الف.۱و۲و۴ ب.۱و۲و۳ ج.۱و۲و۳و۴ د. ۲و۳و۴

۳۷– در مورد کنترل تقدم اپراتورها در عبارتهای میانوندی (Infix) راه‌های زیر پیشنهاد می شود:

۱.پرانتزگذاری کامل توسط برنامه نویس

۲.تامین ابزار کنترل در گرامر و کنترل با تجزیه و تحلیل دستوری برنامه

۳. تجزیه و تحلیل مفهومی برنامه

کدام گزینه درست است؟

الف.اگر گرامر عبارتهای میانوندی مبهم باشد بکارگیری ترکیبی از روشهای ۲ و ۳ ضروری است.

ب. تنها راه ممکن بکارگیری روش ۱ است.

ج. هر یک از روشهای ۱ یا ۲ یا ترکیبی از آنها کافی است.

د. هر ترکیب دوتایی از روش پیشنهادی کفایت می کند.

۳۸– برنامه ای بزرگ با تعدادی برنامه فرعی را در نظر بگیرید که هر برنامه فرعی آن چند بار فراخوانی شده است. اگر این برنامه را بدون تعریف و فراخوانی هیچ برنامه فرعی بنویسیم ترجمه و اجرای آن با برنامه اول چه فرقی خواهد داشت؟

الف.سرعت اجرای برنامه بشتر و سرعت ترجمه نیز بیشتر می شود.

ب.سرعت اجرای برنامه بشتر و سرعت ترجمه کمتر می شود

ج. سرعت اجرای برنامه کمتر و سرعت ترجمه بیشتر می شود

د. سرعت اجرای برنامه کمتر و سرعت ترجمه نیز کمتر می شود.

۳۹- برنامه M را سه بار با روشهای انتقال پارامتر by reference,by value-result و by name اجرا می کنیم. خروجی اجرای اول، دوم و سوم در گزینه‌های این سوال با مجموعه‌های v, r وn معین شده اند، گزینه صحیح کدام است؟

```
Program M;
      K:integer; Y:array [1..3] of integer
Procedure P(X:integer);
Begin
      X:=X+1;
      K:=K+1;
      write (X,Y[1] )
end
begin /* M */
      K:=1;
      Y[1]:=1;
      Y[2]:=3;
      Y[3]:=5;
      P (Y[K]);
      write ( Y[1]+ Y[2]+ Y[3])
end.
```

الف. v={2,2,10}, n={3,2,10}r ={2,1,10},
ب. v={2,2,10}, n={2,2,10}r ={2,1,10},
ج. v={2,1,10}, n={3,2,11}r ={2,2,10},
د. v={2,1,10}, n={3,2,10}r ={2,2,10},

۴۰- در زبان Ada، که هر if با endif تمام می شود، می توان بجای دو واژه else وif (else if) از یک واژه elsif استفاده کرد. تاثیرهای ممکن این کار به شرح زیر پیشنهاد شده است ؟

۱.باعث کاهش تعداد endifها می شود.

۲.باعث ساده تر شدن ترجمه ساختار if-then-else می گردد.

۳.باعث نابرابری تعداد کل ifها با تعداد کل endifها می شود.

۴. جز کاهش تعداد کل واژه‌های بکار رفته در برنامه تاثیر دیگری ندارد.

کدام مجموعه از تاثیرها صحیح است؟

الف. ۱ ب. ۱و۳ ج. ۲و۳ د. ۴

سال ۱۳۸۹

۴۱–قطعه برنامه زیر را در نظر بگیرید:

```
Integer Array  M = [1,2,4,8,32];
Integer X = 1;
Integer f(Integer a,b){
      a := a+2;
      b:=b*2;
      return (100*M[X]+10*b+a;)
 main
{
      Print(f(X.M[X]))
}
```

فرض کنید اندیس آرایه از صفر شروع می شود، مقدار چاپ شده در صورتی که تمامی فراخوانی‌ها با آدرس (Call-By-Reference) باشند چیست؟

الف. ۲۴۳ ب. ۸۴۳ ج. ۴۸۳ د. ۱۷۶۳

۴۲– پیوند ایستا (static link) در رکورد فعالیت (activation record) به کجا اشاره می کند؟

الف. کد رویه صدا زننده (caller) ب.رکورد فعالیت بلاک در برگیرنده

ج.رکورد فعالیت متغیرهای سراسری د. رکورد فعالیت رویه صدا زننده (caller)

۴۳– قطعه کد مقابل را در نظر بگیرید:

```
{
      function f(x,y){return x*y;}
      {
      function g(n){ return f(n,n-1);}
      {
      function f(x,y){return x+y;}
      g(3);
      }
      }
}
```

مقدار نتیجه در حالت static scope و dynamic scope کدام است؟

الف. 5 : static scope : dynamic scope 6

ب. 5: static scope : dynamic scope 5

ج. 6: static scope : dynamic scope 5

د. 6: static scope : dynamic scope 6

۴۴– در زبانهایی که گونه / نوع (type) عبارات را بصورت خودکار استنتاج می کنند، گونه / نوع استنتاج شده برای عبارت زیر کدام است؟

```
f(g,h,x)=g(h(x))
```

گونه / نوع عبارت استنتاج شده در این قالب نوشته می شود :

گونه خروجی تابع ← گونه ورودی سوم × گونه ورودی دوم × گونه ورودی او ل

('a,'b,….'z) متغیرهای گونه (type variable) هستند.)

الف. 'a×'b×'c→'d ب. ('a→'a)×('a→'a)×'a→'a

ج. ('a→'b)×('c→'a)×'c→'b د. ('a→'b)×('a→'b)×'a→'b

۴۵– فرض کنید X متغیری از نوع لیست با مقدار (۲,۳, ۱) و Y متغیری از نوع لیست با مقدار (۵,۶, ۴) باشد.
حاصل عبارت (cons (cadr X)Y) کدام است؟

الف. (۵,۶, ۴ ,۲,۳, ۱) ب. (۵,۶, ۴ , ۱) ج. (۵,۶, ۴ , ۳) د. (۵,۶, ۴ ,۲)

سال ۱۳۹۰

۴۶– شبه کد زیر را در نظر بگیرید:

```
int x=2;
proc f ( )
     {   print (x*x);   }
proc g ( )
     {   h (f);   }
proc h (p)
{
     int x=3;
     p ( );
}
g ( );
```

نتیجه اجرای برنامه در دو حالتی که زبان از حوزه ایستا (static scope) و حوزه پویا (dynamic scope) استفاده می‌کند به ترتیب از راست به چپ کدام است؟

الف. ۹ ، ۴ ب. ۴ ، ۹ ج. ۴ ، ۴ د. ۹ ، ۹

۴۷– این عبارت حساب لامبدا (Lambda Calculus) را در نظر بگیرید : $\lambda x.\,(x(\lambda y.\,ya)x)y$ کدام متغیر یا متغیرها حداقل یک بار در عبارت به صورت آزاد (free) ظاهر شده‌اند.

الف. y , a ب. x , a ج. a د. x , y

۴۸–چنانچه در یک زبان برنامه‌سازی بتوان با اعلان ;A : int Tarray [α..β] یک ماتریس بالا مثلثی از اعداد صحیح تعریف نمود و از روش ردیفی (Row Major) برای ذخیره عناصر استفاده شود، آدرس عنصر A[i,j] از کدام گزینه زیر به دست می‌آید؟ (به شکل زیر توجه کنید) فرض کنید Address (A[α,1]) = μ , size(int) = 2

A[α,1]
A[α+1,1] A[α+1,2]

A[β,1] ---------- A[β,β-α+1]

الف. $\mu + (i - \alpha) \times 2 + (j - 1)$ ب. $\mu + (\sum_{k=1}^{\alpha} k) \times i + (j - 1) \times 2$

ج. $\mu + \left(\sum_{k=1}^{i-\alpha} k\right) \times 2 + (j - 1) \times 2$ د. $\mu + (\beta - \alpha) \times (j - i + 1) \times 2$

۴۹- برنامه زیر را در یک زبان برنامه نویسی که در آن حوزه تعریف متغیرهای تو در تو (Nested Scope) مجاز است در نظر بگیرید:

```
Procedure main
      integer I;
      integer A[0:4];
      for I=0 to 4 do A[I]=1;
      l=I;
      P(l,A[I]);
      Write(l,A[l]);

      procedure P (integer A; integer B);
            integer T;
            A=A+l;     T=B+l;     I=I+l;          B=A+T;
            write (A,B);
      end p
end main
```

برای برنامه بالا، آدرس‌های زیر را در نظر بگیرید:

۴۰۰ = محل دستورالعملی از سیستم عامل که برنامه main را فراخوانی می‌کند.

۶۰۰ = محل فراخوانی رویه P در برنامه main.

۹۰۰ = آدرس ابتدای حافظه ذخیره داده‌های برنامه

۱۰۰۰ = آدرس ابتدای رکورد فعال‌سازی برنامه main.

۱۰۵۰ = آدرس ابتدای محل ذخیره آرایه A.

۱۱۰۰ = آدرس ابتدای رکورد فعال‌سازی رویه P.

با توجه به اطلاعات فوق در رویه P مقدار Static link و Dynamic link به ترتیب از راست به چپ کدام است؟

الف. ۱۱۰۰ و ۹۰۰ ب. ۱۰۰۰ و ۶۰۰ ج. ۶۰۰ و ۱۰۰۰ د. ۱۰۰۰ و ۱۰۰۰

۵۰-کدام گزینه در مورد Dynamic chain pointer (اشاره‌گر زنجیر پویا) ، DCP ، و Static chain pointer (اشاره-گر زنجیر ایستا) ، SCP ، درست است؟

الف. وقتی از قوانین حوزه پویا (Dynamic scope rules) استفاده می‌شود نیاز به SCP نیست.

ب. وقتی از قوانین حوزه ایستا (static scope rules) استفاده می‌شود نیاز به DCP نیست.

ج. برای پیاده‌سازی قوانین حوزه پویا می‌توان از روش نمایشگر (display) استفاده کرد.

د. برای پیاده‌سازی قوانین حوزه ایستا می‌توان از جدول محیط ارجاع مرکزی (centeral referencing environment) استفاده کرد.

۹-۱۷- کلید مجموعه سوالات کنکوری

ردیف	الف	ب	ج	د
۱				*
۲	*			
۳			*	
۴				*
۵				*
۶			*	
۷		*		
۸	*			
۹			*	
۱۰		*		
۱۱				*
۱۲	*			
۱۳				*
۱۴		*		
۱۵		*		
۱۶			*	
۱۷			*	
۱۸	*			
۱۹				*
۲۰	*			
۲۱		*		
۲۲		*		
۲۳			*	

ردیف	الف	ب	ج	د
۲۴	*			
۲۵	*			
۲۶			*	
۲۷		*		
۲۸	*			
۲۹		*		
۳۰			*	
۳۱	*			
۳۲			*	
۳۳		*		
۳۴				*
۳۵	*			
۳۶			*	
۳۷			*	
۳۸		*		
۳۹				*
۴۰	*			
۴۱		*		
۴۲		*		
۴۳	*			
۴۴			*	
۴۵				*
۴۶		*		
۴۷	*			
۴۸			*	
۴۹	*			
۵۰			*	

مراجع:

1- Programming Languages:Design and Implementation,Pratt(2001)

2- Concepts of programming languages,W.sebesta(2008)