منبع درس طراحی زبانهای برنامه نویسی

دانشجویان کامپیوتر موسسه آموزش عالی کوشیار گیلانی

فصل اول :

اصول طراحی زبانـها

ا # ا

:

دلايل مطالعه زبانهاي
برنامه سازي تاريخچه
اي از زبانهاي برنامه
سازي زبانهاي
محاسباتي زبانهاي
تجاري زبانهاي هوش
مصنوعي زبانهاي
سيستمي
تأثير محيط بر روي
طراحي و پياده سازي
زبانها محيط دسته اي

محيط محاوره اي محيط سيستمهاي
تعبيه شده محيط كامپيوتر شخصي
محيط شبكه و اينترنت
دامنه كاربرد زبانها
ويژگيهاي يك زبان خوب نحو و
معناي زبان
مدلهاي محاسباتي زبان زبانهاي
دستوري زبانهاي تابعي زبانهاي
قانونمند زبانهاي شي گرا
استاندارد سازي زبانها زمان
شناسي اطاعت وپيروي
كهنگي ومنسوخ شدن
سوالات تستي و تسريحي

زبان برنامه نويسي: » هر گونه علامت گذاري جهت توصيف الگوريتمها و ساختمان دادهها «

1-1- دلايل مطالعه زبانهاي برنامه سازي

1-1- 1- براي افزايش تواناييهاي خود در توسعه الگوريتمهاي كارآمد

بسياري از زبانها ويژگيهايي دارند كه اگر به خوبي مورد استفاده قرار گيرند به نفع برنامه نويس است، ولي اگر به طور نامناسب مورد استفاده قرار گيرند ، وقت زيادي از برنامه نويس ميگيرند. مثلاً اگر برنامه نويس از تكنيك بازگشتي ، مزايا ومشكلات آن اطلاع داشته باشد، ميتواند تصميم بگيرد چه موقعي از آنها استفاده كند يا نكند.

1-1- 2- استفاده بهينه از زبانهاي برنامه نويسي موجود

بادرك اينكه چگونه ويژگيهاي يك زبان پياده سازي ميشوند توانايي شما در نوشتن برنامههاي كارآمد افزايش مييابد به عنوان مثال اگر بدانيد آرايهها، ركوردها ،ليستهاو رشتهها در زبان برنامه نويسي مورد نظرتان چگونه ايجاد و پياده سازي ميشوند ميتوانيد از زبان برنامه نويسي به طور بهينه بهره ببريد.

1-1- 3- آشنايي با اصطلاحات مفيد ساختارهاي برنامه نويسي

زبانها هم ميتوانند به عنوان وسيله اي براي محدوديت فكر كردن و هم وسيله اي براي كمك به فكر كردن باشند اگر برنامه نويس فقط با يك زبان آشنايي داشته باشد در جستجو براي حل مسئله، فقط به تواناييهاي زباني فكر ميكند كه با آن آشنا است واين يك محدوديت است با مطالعه زبانهاي برنامه نويسي متعدد، دامنه لغات يك برنامه نويس افزايش مييابد.به عنوان مثال يك ساختار كنترلي به نام همروال[1] در خيلي از برنامهها مفيد است ولي زبانهاي محدودي اين قابليت راپشتيباني مي كنند(مانند Cو فرترن)

1-1- 4- انتخاب بهترين زبان برنامه نويسي

آگاهي از زبانهاي مختلف باعث ميگردد كه براي مسئله خاص بتوان زبان بهتري را انتخاب كرد به عنوان مثال كاربردهايي كه نياز به محاسبات عددي دارند بهتر است از C، فرترن يا Ada استفاده كنند. دركاربردهاي هوش مصنوعي از Lisp، ML ،Prolog و جهت كاربردهاي اينترنت، Perlو Java مناسب هستند.

1-1- 5- يادگيري آسان يك زبان جديد

اگر با ساختار يك زبان طبيعي آشنايي كامل داشته باشيد، آموزش زبان طبيعي جديد ساده خواهد شد. اين موضوع در مورد زبانهاي برنامه نويسي نيز صادق است.

Coroutines [1]

1-1- 6- طراحي يك زبان جديد

برخي از برنامه نويسان خود را به عنوان طراح زبان ميدانند مثلا طراح يك واسط كاربر جهت نوشتن برنامههاي بزرگي مثل ويرايشگر متن يا سيستم عامل، ميبايست با موارد مشابهي كه در طراحي زبانهاي برنامه نويسي همه منظوره مورد استفاده قرار ميگيرند آشنايي داشته باشد.

1-2- تاريخچه زبانهاي برنامه نويسي

فرترن و Lisp در دهه 1950 ، پاسكال ، Ada، Prolog و اسمالتاك در دهه 1970، C++، Perl وML دردهه 1980 و Java دردهه1990 طراحي شدند؛ زبان فرترن براي كارهاي علمي و محاسباتي ساخته شده بود. تاريخچه زبانهاي برنامه نويسي را بر اساس كاربرد آنها يعني محاسباتي، تجاري ، هوش مصنوعي، و سيستمي مورد بررسي قرار مي دهيم.

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 1 - 1

1-2- 1- زبانهاي محاسباتي (مبتني بر اعداد)

درحدود سال 1960 زبان الگول به عنوان يك زبان محاسباتي آكادميك معرفي شد. اهداف اصلي الگول در آن زمان شامل موارد زير بود :

- اين زبان بايد به رياضيات نزديك باشد.
- جهت توصيف الگوريتمها مناسب باشد.
- برنامهها بايد به زبان ماشين ترجمه گردند.
- اين زبان نبايد وابسته به يك ماشين خاصي باشد.

1-2- 2- زبانهاي تجاري

هدف از زبانهاي تجاري اين بود كه، برنامههايي كه ايجاد ميشوند از متنهايي به شكل متن انگليسي استفاده كنند. كوبول، زباني جهت كاربردهاي تجاري ميباشد. هدف مهم دركوبول، نوشتن برنامه اي بود كه به زبان انگليسي نزديك باشد. هر چند كه كوبول خوانايي خوبي دارد اما نحو رسمي ندارد و برنامه نويسي در آن دشواراست.

نكته : PL/I ويژگيهاي عددي فرترن را با خصوصيات برنامه سازي تجاري كوبول تركيب كرد.

PL/I = Cobol + Fortran

1-2- 3- زبانهاي هوش مصنوعي

تمايل به زبانهاي هوش مصنوعي در دهه 1950 با IPL شروع شد. ليسپ، به عنوان يك زبان تابعي پردازش كننده ليست طراحي شد. زبان ليسپ جهت كاربردهاي هوش مصنوعي شامل ويژگيهاي زير است:

- يك زبان تابعي پردازش كننده ليست است.
- عمليات جستجو را به

خوبي انجام ميدهد.

- پياده سازي بازيهاي كامپيوتري درآن به خوبي صورت ميگيرد.
- قابليتهاي مناسبي جهت پردازش متن دارد.
- رشتههايي از نمادها ميتوانند توسط رشتههاي ديگر جايگزين شوند (تفسير ماشين خودكار).

نكته : ليسپ، جهت پردازش ليست همه منظوره طراحي شد ولي پرولوگ يك زبان تك منظوره بود كه ساختار آن بر اساس منطق رياضي بود 0 **1-2- 4- زبانهاي سيستمي**

اين دسته از زبانها براي نوشتن سيستم عامل و پياده سازي كامپايلرها و ... كاربرد دارند. اين گروه از زبانها بايد قادر به دستيابي به سخت افزار و ايجاد ارتباط با آن باشند مثل C ، PL/I و اسمبلي.

1-3- تاثير محيط اجرايي بر روي طراحي و پياده سازي زبانها

توسعه نرم افزار در خلا انجام نميشود .سخت افزاري كه زبان را پشتيباني ميكند تاثير زيادي بر طراحي زبان برنامه نويسي دارد. محيطي كه برنامه بر روي آن اجرا مي شود(شامل سيستم عامل و سخت افزار)، محيط عملياتي (مقصد) نام دارد. محيطي كه برنامه در آن ايجاد ،تست و اشكال زدايي ميشود، محيط ميزبان نام دارد. محيط عملياتي ممكن است با محيط ميزبان متفاوت باشد.

در ذيل به چند نمونه از اين محيطها كه توسعه و اجراي نرم افزار در آنها صورت ميگيرد اشاره ميكنيم :

1-3- 1- محيط دسته اي[1]

براي زبانهايي كه جهت محيطهاي دسته اي يا offline طراحي شده اند، فايلها معمولي ترين ابزار جهت ورودي/خروجي هستند. برنامه، مجموعه اي از فايلهاي داده را به عنوان ورودي گرفته، دادههاي آنها را پردازش كرده و مجموعه اي از فايلهاي داده خروجي را توليد ميكند. اين محيط عملياتي را پردازش دسته اي ميگويند. زيرا اطلاعات ورودي در فايلها دسته بندي ميشوند و به صورت دسته اي پردازش ميشوند. در اين محيط خطايي

Batch [1]

كه اجراي برنامه را خاتمه دهد قابل قبول بوده ولي هزينه بر است، زيرا پس از پردازش و تصحيح خطا، برنامه بايدبه طور كامل اجرا شود. ويژگي ديگر محيط دسته اي، محدوديت زماني براي برنامه است يعني زبان استفاده شده براي اين محيط، امكاناتي را جهت تاثير بر روي سرعت اجراي برنامه در اختيار نميگذارد به عبارتي ديگر، زبان استفاده شده در محيط دسته اي Timing مشخصي ندارد. زبانهايي مثل فرترن، كوبول و پاسكال ابتدا براي محيطهاي دسته اي طراحي شدند ولي امروزه در محيطهاي محاوره يا سيستم تعبيه شده نيز به كار ميروند.

1-3- 2- محيط محاوره اي[1]

در اين محيط يك برنامه مستقيماً با كاربر تعامل دارد و خروجي در نمايشگر نمايش داده ميشود اينگونه محيطها، از سيستم اشتراك زماني براي انجام كارهاي مختلف، كه در يك زمان واحد به كامپيوتر داده ميشود استفاده ميكنند؛ به اين ترتيب كه به هر برنامه يك برش زماني[2] اختصاص داده ميشود .بعد از اتمام اين برش زماني، پردازنده به برنامه ديگري كه در حال اجراست داده ميشود .پردازش خطا در محيط محاوره اي از اهميت كمتري برخوردار است و زبان به كار رفته در اين محيط نياز مبرمي به داشتن پردازش خطاهاي قوي ندارد، زيرا كاربر به صورت online پشت كامپيوتر بوده و در صورت بروز خطا ميتواند برنامه را دستكاري و تصحيح كند، اما خاتمه برنامه در صورت بروز خطا قابل قبول نيست. زبانهاي C،++C براي نوشتن برنامههاي محاوره اي مناسب هستند.

1-3- 3- محيط سيستمهاي تعبيه شده(توكار)[3]

به سيستم كامپيوتري كه جهت كنترل بخشي از يك سيستم بزرگ مثل هواپيما يا ماشين آلات صنعتي استفاده ميشود، سيستم كامپيوتري تعبيه شده يا توكار گفته ميشود .در محيطهاي دسته اي و محاوره اي خرابي سيستم چندان اهميت ندارد، ولي در سيستم توكار، خرابي سيستم زيانهاي جدي به بار

ميآورد .سيستمهاي توكار ،معمولا به صورت بلادرنگ[4] هستند يعني در يك محدوده زماني از قبل تعيين شده بايد به ورودیها پاسخ دهند.

ويژگيهاي محيط تعبيه شده

1 - برنامههاي نوشته شده براي اينگونه محيطها معمولاً بدون سيستم عامل، بدون سيستم فايل و بدون دستگاه -هاي I/O فايل اجرا ميشوند .در عوض هر كدام از برنامهها رويههاي مخصوصي براي ارتباط با دستگاههاي ورودي/ خروجي سيستم بزرگ دارند و به تعامل با آنها ميپردازند.

2 - قابليت اطمينان[5] از اهميت خاصي برخوردار است.

3 - در اين نوع سيستمها، اداره خطاها و استثناها بايد به خوبي انجام گيرد. خاتمه برنامه جز در موارد خرابي كلي سيستم قابل قبول نيست و كاربري وجود ندارد كه به صورت محاوره اي خطاها را برطرف سازد.

---

interactive [1]

time slice [2]

Embedded system [٣] Real time [٤]

Reliability [٥]

4 -در سيستمهاي توكار اغلب از كامپيوترهاي RISC كه تعداد دستورات محدودي دارند استفاده ميشود .زبانهايAda ، C و C++ براي نوشتن برنامههاي مربوط به سيستم توكار مفيد هستند.

1-3- 4- محيط كامپيوتر شخصي

در موارد زيادي در محيط كامپيوترهاي شخصي، كارايي، كمتر مد نظر قرار ميگيرد و هدف اصلي اغلب، راحتي كاربر و داشتن يك محيط گرافيكي ساده است. نمونه مشهور آن برنامه نويسي تحت محيط ويندوز است. برنامه نويسي شيءگرا، مدل مناسبي براي چنين محيطهايي است و زبانهايي مثل C++ يا Java در چنين محيطي كاربرد زيادي دارند.

1-3- 5- محيط شبكه و اينترنت

اينترنت اوليه، دو سرويس FTP و Telenet داشت. در پروتكل Telenet، كاربر به عنوان بخشي از كارگزار راه دور عمل ميكرد و به كمك FTP ميتوانست فايلهايي را از ماشين كارگزار[1] گرفته و يا به آن بفرستد. در هر دو پروتكل، كاربر بايد بداند كه چه ماشيني اطلاعات مورد نياز او را دارد. بعد از ايجاد زبان HTML و پروتكل انتقال HTTP واستفاده از وب جهاني (WWW)، نقش زبان برنامه سازي تغيير كرد. اين زبانها بايد تعامل بين كامپيوترهاي Server و Client را امكان پذير سازند.

1-4- دامنه كاربرد زبانها

در دهه 1960 اغلب براي كاربردهاي تجاري از زبان كوبول ، براي امور علمي از فرترن و الگول ، براي كاربردهاي سيستمي از اسمبلر يا فورث(Forth) و براي كاربردهاي هوش مصنوعي ازLisp و Snobol استفاده ميشد .

امروزه براي كاربردهاي تجاري از كوبول، براي صفحه گستردهها از زبانهاي نسل چهارم(C++ 4GL)، وJava ، براي كاربردهاي علمي از فرترن ،C ،C++ و Java ، براي كاربردهاي سيستمي از C، C++ و Java ، براي هوش مصنوعي، از ليسپ و پرولوگ استفاده ميشود .

برنامههاي هوش مصنوعي با الگوريتمهايي نوشته ميشوند كه عمل جستجو را در بخش بزرگي از دادهها انجام ميدهند. مثلاً براي شطرنج، كامپيوتر حركتهاي زيادي را ايجاد كرده و آنگاه بهترين حركت را انتخاب ميكند. در دنياي اينترنت، زبانهايي مثل Perl و جاوا اسكريپت ، به سرور امكان ميدهند كه دادهها را از كاربر گرفته و تراكنشي را انجام دهند. امروزه بسياري از دستگاهها مثل خودروها و تلويزيونهاي ديجيتالي، پردازنده دارند و اين وسايل به زبانهاي بي درنگ[2] نياز دارند كه C و++C و Ada در اين موارد كاربرد دارند. ML، در تحقيقات زبان -هاي برنامه سازي به كار گرفته شده است. هر چند اسمالتاك زبان معروفي نيست، ولي خيلي از خصوصيات شيءگرايي ++C، از اسمالتاك گرفته شده است. جدول زير كاربرد زبانهاي برنامه نويسي در زمان گذشته و حال را نشان مي دهد.

زبان مورد استفاده	كاربرد	دوره
Cobol	تجاري	1960
Fortran,Basic,Algol,APL	علمي	
Assembly	سيستمي	
Lisp,Snobol	هوش مصنوعي	
C++,Java,4GL	تجاري	امروزه
Java,C,C++,Basic	علمي	
C,C++,Java	سيستمي	
Lisp,Prolog	هوش مصنوعي	
Tex,Postscript	انتشارات	

جدول 1 - 2

[1] Server

[2] Real time

كاربرد زبانهاي برنامه سازي

- **تجاري** : مانند طراحي پايگاه دادهها (Cobol , C , PL/I)
- **علمي** : مثل كارهاي محاسباتي كه فرمولهاي زيادي دارند. (Basic , C++ , C , Fortran)
- **سيستمي** : يعني چيزي شبيه به سيستم عامل بنويسيم. (C , C++ , Assembly)
- **هوش مصنوعي** : بازيهاي فكري مثل شطرنج (Prolog , Lisp)
- واژه پردازها **(publishing) :** Tex ، Postscript ، **انتشارات**
- **پردازشي (Process)** : كارهاي پردازشي بالا ، يعني بيشتر زمان روي CPU و سيستم فايل درگير

(Perl , TCL , Unix) است

- **آموزشي** : براي آشنايي با برنامه نويسي (Basic , Pascal , C)
- **New** : اغلب كاربردهاي بالا را پشتيباني ميكنند و به محاوره اي نزديك هستند (Ifel , ML , Smaltalk)

عوامل موثر بر پيدايش و طراحي زبانها

- **سخت افزار و سيستم** عامل دو جنبه مهم در پيدايش و طراحي زبانها بوده اند.
- **روشهاي برنامه سازي** تغيير ميكردند و باعث تغيير زبانها ميشدند.

روش رويه اي روش ساخت يافته (تابعي) روش قانونمند روش شيءگرا

- **كاربردها** (آموزشي ، علمي ، تجاري و ...)
- **روشهاي پياده سازي** : سازگار با محيط پياده سازي باشد
- **مطالعات تئوريك** (مثل زبان α كه پياده سازي نشد اما اصول آن در پايگاه داده به كار ميرود)
- **استاندارد سازي** :معيارهاي استاندارسازي با توجه به شرايط زماني و تكنولوژي تغيير ميكند.

انواع استاندارد سازي :

خصوصي (مخصوص سازمان خاص)

عمومي (همه سازمانها از آن حمايت ميكنند) مسائل مهم در استاندارد سازي :

شرايط زماني انطباق و
توافق كهنگي (از رده
خارج شدن)

1-5- ويژگيهاي يك زبان خوب

1-5- 1- وضوح ، سادگي و يكپارچگي

در يك زبان بهتر است تعداد كمي مفاهيم مختلف و قانون جهت تركيب آنها وجود داشته باشد اين ويژگي را جامعيت مفهومي ميگويند.

نحو يك زبان روي نوشتن ، تست كردن ، اصلاح و درك زبان اثرگذار است. قابليت خوانايي برنامهها نيز يك اصل مهم ميباشد. نحوي كه مختصر و رمزگونه است برنامه نويسي را كوتاه تر وساده تر ميكند، ولي قابليت خوانايي را كاهش ميدهد. مثلاً برنامههاي APL حالت رمز گونه دارند و قابليت خوانايي كمي دارند.بسياري از زبانها ،ساختارهاي نحوي دارند كه دو جمله تقريباً مشابه، معاني مختلفي دارند در نتيجه قابليت خوانايي كاهش مييابد .مثلا كاراكترb در زبان Snobol4 در يكجا به عنوان جدا كننده و در جايي ديگر به عنوان الحاق كننده دو رشته به كار ميرود .

1-5- 2- قابليت تعامد[1]

منظور از تعامد اين است كه بتوان خصوصيات مختلفي از يك زبان را با هم تركيب كرد و اين تركيب با معنا باشد. مثلاً اگر عبارتي در يك زبان فرضي، بتواند مقداري را توليد كند و علاوه بر آن شامل عباراتي باشد كه ارزش T يا F دارند اين زبان قابليت تعامد را داراست.

مثال ديگر: عبارت محاسباتي و دستور شرطي : ... then (a+b > c+d) If يعني يك دستور محاسباتي را در دستور If به كار برديم.

1-5- 3- طبيعي بودن براي كاربردها

هر زبان در هنگام استفاده در كاربرد خاص خود بايد مناسب به نظر آيد. اگر ساختار برنامه، ساختار منطقي مربوط به الگوريتم را به خوبي نشان دهد ، آن زبان اين ويژگي را دارا خواهد بود. زبانهايي كه براي كاربردهاي خاصي هستند اين خصوصيت را در همان زمينه كاربردي دارا ميباشند. مثلاً براي محاسبات علمي و فني از فرترن و براي پردازش رشتهها از زبان Lisp استفاده ميكنيم .

---

[1] Orthogonality

1-5- 4- پشتيباني از انتزاع[1]

زبان برنامه نويسي بايد امكان ايجاد ساختارهاي داده اي جديد را براي برنامه نويس فراهم كند. هر زبان تعداد معيني نوع داده اوليه دارد. مثلاً در زبان C و پاسكال، داده اعداد مختلط و اعمال روي آنها وجود ندارد. ولي در زبان -هايي مانند C++ و Ada، برنامه نويس ميتواند نوع دادههاي انتزاعي (ADT) مورد نظر خود را توليد كند. مثلاً ميتواند نوع داده اعداد مختلط و عمليات جمع ، تفريق و ضرب را بر روي آنها تعريف كند. استفاده از جنبههاي انتزاعي، قابليت اطمينان را افزايش ميدهد .

1-5- 5- سهولت در بازرسي برنامه

برنامهها بايد به گونه اي باشند كه بررسي صحت عملكرد آنها ساده باشد برنامههايي كه ساختار نحوي و معنايي ساده تري دارند بازرسي آنها نيز ساده تر مي باشد. بررسي درستي برنامه به دو صورت رسمي مانند روشهاي رياضي و غير رسمي نظير تست برنامه با ورودیهاي مختلف انجام ميشود .

1-5- 6- محيط برنامه نويسي

يك محيط برنامه نويسي قوي، ايجاد برنامهها و عيب يابي آنها را ساده تر مي سازد. هر چه امكانات محيط برنامه سازي نظير كامپايلر، لينكر، ديباگر و... بيشتر باشد، آن زبان برنامه نويسي بهتر خواهد بود؛ مثلاً برنامههاي ويژوال عموماً يك محيط برنامه نويسي قدرتمند دارد. اسمالتاك زباني است كه محيط آن داراي پنجرهها، منوها ،ورودي موس و ... براي كار كردن روي برنامهها مي باشد.

1-5- 7- قابليت حمل[2]

يك زبان خوب بايد بتواند بر روي سيستمهاي مختلف كامپيوتري به سادگي انتقال يافته و اجرا شود. يكي از معيار -هاي مهم براي پروژههاي برنامه نويسي قابليت انتقال يك برنامه از يك ماشين به ماشين ديگر است.زباني كه به ماشين خاصي وابسته نباشد برنامههاي نوشته شده در آن زبان از ماشيني به ماشين ديگر قابل انتقال هستند. C، Ada، Fortran و پاسكال تعاريف استانداري براي توليد برنامههاي قابل حمل دارند. مثلاً زبان جاوا اين ويژگي را در حد بسيار بالايي دارد و با افزايش قابليت حمل، انعطاف پذيري و كارآيي كاهش مي يابد.

1-5- 8- هزينه استفاده

هزينه، عنصر مهمي در ارزيابي زبانهاي برنامه نويسي است. 3 معيار اصلي هزينه عبارت اند از:

الف: هزينه اجراي برنامه: زمان اجراي يك برنامه خصوصاً برنامههاي بزرگي كه به دفعات زيادي اجرا مي شوند معياري مهم مي باشند هزينه اجراي برنامه شامل استفاده از منابع كامپيوتري خصوصاً CPU، RAM وDisk مي باشد.

[1] Abstraction
[2] Port ability

ب : هزينه ترجمه ي برنامه: يعني مدت زماني كه طول مي كشد تا برنامه كامپايل شود مثلاً برنامههاي دانشجويان، چندين بار ترجمه شده و كمتر اجرا مي شوند و اين هزينه مهم تر است (چون ممكن است خطا داشته باشد و بايد چندين بار ترجمه كنيد تا كل خطاها بر طرف شود) هزينه ترجمه شامل استفاده از منابع كامپيوتري
مي باشد RAM، Disk, CPU خصوصاً

ج : هزينه نگهداري برنامه: شامل هزينه ي تطبيق برنامه با جوانب محيطي، ايجاد امكانات جديد در نرم افزار و برطرف كردن اشكالات است.

1-6- نحو و معناي زبان

نحو[1] زبان : نحو زبان برنامه سازي، ظاهرآن زبان است ومفهوم قواعد نحوي اين است كه مشاهده شود دستورات ،اعلانها و ساير ساختارهاي زبان چگونه نوشته مي شوند .

معناي[2] زبان : همان مفهومي است كه به ساختارهاي نحوي زبان داده مي شود.

مثال: اعلان آرايه بردار عنصري از نوع صحيح :

v:array [0..9] of integer; تعريف آرايه در پاسكال int v[10];

تعريف آرايه در سيمعنا: هر دو اعلان بدين معناست كه 01 خانه براي اعداد صحيح در نظر گرفته شود.

1-7- مدلهاي محاسباتي زبان

چهار مدل محاسباتي مختلف وجود دارد كه برنامه نويسي را توصيف مي كند كه عبارتند از: 1 - دستوري 2- تابعي 3 - قانونمند 4 - شيء گرا

1-7- 1- زبانهاي دستوري[3]

در اين زبانها برنامه شامل دنباله اي از دستورات است و اجراي هر دستور موجب مي شود مترجم مقدار يك يا چند خانه حافظه را تغيير دهد يعني ماشين را به حالت جديدي وارد كند نحو چنين زبانهايي به صورت زير است :

1 ; دستور

2 ; دستور

دستور n;

[1] Syntax
[2] Semanatic
[3] Imperative

اين مدل در واقع از سخت افزار كامپيوتر تبعيت مي كند چون سخت افزار نيز دستورات را به ترتيب اجرا مي كند اغلب زبانهاي قديمي مانند C ، C++ ، Fortran ، Algol ،PL/I ، پاسكال، Ada و Cobol از اين مدل پشتيباني مي كند. زبانهاي امروزه مانند C ، C++ و كوبول نيز از اين مدل پشتيباني مي كنند.

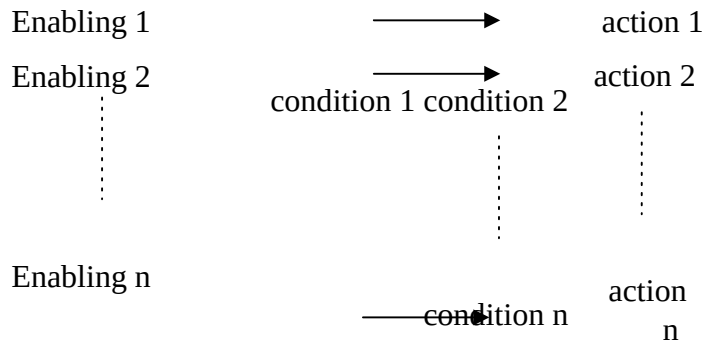
1-7- 2- زبانهاي تابعي[1]

در اين روش به جاي دنبال كردن تغيير حالت ماشين، عملكرد برنامه دنبال مي شود؛ يعني به جاي آنكه دادههاي موجود را در نظر بگيريم، نتيجه مطلوب را در نظر خواهيم داشت. يعني در صورت برقراري عمليات بر روي داده ورودي، نتيجه مطلوب حاصل مي شود عملي بايد روي حالت اوليه ي ماشين انجام پذيرد تا با دستيابي به داده اوليه و تركيب آنها پاسخ مناسب بدست آيد.
توسعه برنامه با ايجاد توابعي از توابع ايجاد شده قبلي به منظور ساختن توابع پيچيده انجام مي شود تا اين داده اوليه را دستكاري كرده و آخرين پاسخ را از دادههاي اوليه توليد نمايد. ظاهر برنامه در اين مدل، به صورت فراخوان تعدادي تابع و ارسال نتيجه آنها به عنوان پارامتر به توابع قبلي است. نحو اين زبان به صورت زير است :

. f_n (… f_2 (f_1 (data)) …) زبانهاي تابعي هستندLisp، و ML زبانهاي

1-7- 3- زبانهاي قانونمند[2]

در زبانهاي قانونمند شرايطي بررسي شده و اگر اين شرايط برقرار باشند اعمالي انجام مي گردد معروف ترين زبان قانونمند ،Prolog است كه به آن زبان برنامه نويسي منطقي هم گفته مي شود0 نحوه كلي چنين زبانها يي به شكل زير است:



اجراي اين زبان شبيه زبان دستوري است با اين تفاوت كه دستورات به ترتيب اجرا نمي شوند بلكه فعال شدن شرطها ترتيب اجرا را تعيين مي كند اين زبانها اغلب در برنامههاي كاربردي هوش مصنوعي و سيستمهاي خبره مورد استفاده قرار مي گيرند .

apllicative
۱ Rule-Based ۲

1-7- 4- زبانهاي شئ گرا[1]

در زبانهاي شئ گرا ، اشياء داده اي پديد آمده و مجموعه اي از توابع تعريف مي شوند تا روي اين دادهها كار كنند اشياء پيچيده را مي توان با بسط اشياي ساده و مفهوم ارث بري ايجاد كرد. دربرنامه نويسي شئ گرا، با ساختن اشياي داده اي دقيق، كارايي زبانهاي دستوري به دست مي آيد همچنين با ساختن دسته اي از توابع كه از مجموعه ي معيني از اشياي داده استفاده مي كنند قابليت اطمينان[2] و قابليت انعطاف[3] برنامه نويسي تابعي حاصل مي شود. C++ و جاوا نمونه اي از زبانهاي شيء گرا هستند.

نكته : مي توان برنامههايي به زبان Lisp و Prolog نوشت كه به ترتيب اجرا مي شود تا عمل خاصي را انجام دهند. همچنين مي توان به زبان C برنامه اي نوشت كه فقط از فراخواني تابع تشكيل شده باشد و به اين ترتيب مثل يك زبان تابعي به نظر آيد. تكنيكهاي تابعي، براي كنترل برنامهها و صحت آنها به وجود آمده است.

نكته : زبان C++ تركيبي از يك زبان تابعي، دستوري و شي گرا محسوب مي شود.

به طور خلاصه خواهيم داشت:

- **زبانهاي دستوري يا رويه اي (*Imperative*)**

برنامه يك سري از دستورات پشت سرهم است كه از بالا به پايين اجرا ميشوند. Statement 1; Statement 2;
مانند C, C++, Fortran, Algol, Cobol, PL/I, Ada, Smalltalk, Pascal :

• زبانهاي كاربردي يا تابعي (*Functional*)
توابع در كنار هم برنامه را تشكيل ميدهند و قابليت فراخواني تودرتو دارند. $Function_n(\ldots Function_1\ (data)\ldots)$
مانند Schema, ML, Lisp :

• مدل مبتني برقانون (*Rule-Base*)
برنامهها به صورت مجموعه اي از قوانين پايه اي تجزيه مساله ، ميباشند. ساختمان اين زبان مشابه ساختار if است كه در صورت رخداد شرايطي ،قانوني اجرا ميشود . Enabling condition1 → action1 = Action1 if enabling condition1
مانند Prolog :

[1] Object Orinted
[2] Reliability
[3] Flexibility

(*Object Oriented*) زبانهاي شيء گرا•
برنامه از تعدادي ماژول تشكيل شده است كه هر ماژول مشابه يك برنامه در زبان C ميباشد. مبحث اصلي در اين نوع زبان پشتيباني از كلاس است.
مانند C++, Java, Visual :

جدول 1 - 3

1-8- استاندارد سازي زبانها

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

هنگام مشاهده يك دستور در يك زبان براي پي بردن به معناي دستور و خروجي آن سه راه حل وجود دارد:

- مراجعه به مستندات زبان.
- تايپ واجراي برنامه روي كامپيوتر
- مراجعه به استاندارد زبان

براي مقابله با يك سري مشكلاتي كه باعث ناسازگاري مي شوند هر زبان داراي تعاريف استانداردي است. تمام پياده سازيها بايد از اين استاندارد پيروي كنند .استاندارد سازي يك زبان، قابليت حمل و قابليت انعطاف آن را افزايش مي دهد. زباني كه قابليت حمل دارد، قابليت اطمينان را نيز افزايش مي دهد ولي عكس آن لزوماً درست نيست. استانداردها به دو دسته كلي تقسيم مي شوند.

- **استانداردهاي خصوصي :** كه شركت مالك سازنده ي آن زبان، آن را معرفي مي كند و اين استاندارد براي زبانهايي كه گسترده اند مناسب نيست.
- **استانداردهاي عمومي:** كه توسط سازمانهاي معروف نظير , IEEE ISO ، ANSI ارائه مي شوند.

براي استفاده موثر از استانداردها، سه نكته بايد مد نظر قرار گيرد : 1-زمان شناسي 2 - اطاعت و پيروي 3 - كهنگي و منسوخ شدن

1-8- 1- زمان شناسي[1]

عموماً برنامه نويسان دوست دارند زبانها هرچه زودتر استاندارد شوند تا پياده سازيهاي ناسازگاري از آنها ارائه نگردد. مثلاً زبان فرترن، خيلي دير استاندارد شد ودر زمان استانداردسازي آن، نسخههاي ناسازگار زيادي از آن وجود داشت. زبان Ada زودتر از پياده سازي اش استاندارد شد. و زبانهاي C و پاسكال، هنگامي استاندارد شدند كه ،تنها نمونههاي كمي از آنها پياده سازي شده بود.

[1] TimeLines

1-8- 2- اطاعت وپيروي[1]

يعني اينكه برنامهها بايد از استاندارد پيروي كنند.كامپايلر پيرو، كامپايلري است كه وقتي يك برنامه استاندارد به آن داده مي شود نتيجه مشخصي(درستي) را مي دهد. ممكن است زبانها امكانات اضافي علاوه بر استاندارد داشته باشند كه هيچ نتيجه اي براي آن مشخص نمي شود.

1-8- 3- كهنگي ومنسوخ شدن[2]

اغلب، هر استاندارد، در طول 5 سال يكبار بايد بازرسي و احياناً بازسازي شود. در موارد زيادي استانداردهاي جديد با استانداردهاي قبلي سازگاري دارند ولي در مواردي هم، يك ويژگي دراستانداردهاي بعدي يعني حدود 5 تا 01 سال آينده حذف خواهد شد كه به اين مفهوم كهنگي يا منسوخ شدن مي گويند.

[1] Conformance
[2] Obsolescence

1-9- سوالات فصل اول

سوالات تستي

1- كداميك از موارد زير از صفات يك زبان خوب نمي باشد؟(نيمسال دوم 38) الف.
وضوح، سادگي و يكپارچگي ب .عدم قابليت تعامد
ج .پشتيباني از انتزاع د .طبيعي بودن براي كاربردها.

2- كداميك از زبانهاي زير يك زبان تجاري نمي باشد؟(نيمسال دوم 38)
الف FLOMATIC. د.CBL ج.PROLOG ب.COBOL

3- مسائلي كه براي استفاده موثر از استانداردها بايستي در نظر گرفته شود عبارتند از:
(نيمسال دوم 38) الف. كهنگي، زمان شناسي، اطاعت و پيروي ب .اطاعت و پيروي، هزينه، سادگي ج .كهنگي، هزينه، سادگي طراحي د .زمان شناسي، سادگي، سرعت

4-كدام گزينه جزء اهداف الگول نيست؟ (نيمسال دوم 48)
الف. براي توصيف الگوريتمها مفيد باشد. ب .بايستي به رياضيات استاندارد نزديك باشد.
ج .بايستي به معماري يك ماشين مقيد باشد. د .برنامه ها بايستي به زبان ماشين ترجمه شوند.

5- در مورد LISP كدام گزينه غلط ميباشد؟(نيمسال دوم 48)
الف. به عنوان يك زبان پردازش كننده ليست طراحي شد. ب. براي كارهاي جستجو اصلاً مناسب نيست.
ج. بازيهاي كامپيوتري در LISP به خوبي پياده سازي ميشوند. د. هيچكدام.

6- كدام گزينه صحيح است؟ (نيمسال دوم 48)
الف. پردازش خطا در محيط محاورهاي با محيط دستهاي تفاوتي ندارد.
ب. برنامه هاي محاورهاي بايد از محدوديتهاي زماني برخوردار باشند.
ج .پردازش خطا در سيستمهاي تعبيه شده از اهميت ويژهاي برخوردار نيست.
د .كليه موارد بالا.

7-به منظور استفاده از استانداردها كدام گزينه بايستي مدنظر قرار گيرد؟(نيمسال دوم 48) الف. زمان شناسي ب. كهنگي ج. اطاعت و پيروي د. تمام موارد

8- IPL به عنوان اولين زبان مربوط به كدام گروه از كاربردها مطرح شد؟(نيمسال اول 58-68) الف.تجاري ب.علمي ج .هوش مصنوعي د.سيستمي

9-كدام مورد از دلايل مطالعه زبانهاي برنامه سازي مي باشد؟(نيمسال اول 68 -78)

الف.استفاده بهينه از زبان برنامه سازي موجود ب.شناختن ساختارهاي مفيد زبانهاي برنامه نويسي ج.انتخاب بهترين زبان برنامه سازي براي يك پروژه خاص د .همه موارد صحيح است.

01-اهداف ALGOL عبارتند از:(نيمسال دوم 58 -68)

الف.نشانهها بايستي به رياضيات استاندارد نزديك باشد. ب.بايستي براي توصيف الگوريتمها مفيد باشد.

ج.نبايستي به معماري يك ماشين مقيد باشد. د.كليه موارد بالا **11-**انواع زبانها عبارتند از:(نيمسال دوم 58 -68)

الف.مبتني بر اعداد ب.تجاري و هوش مصنوعي ج.سيستم د.كليه موارد بالا **21-**كدام گزينه غلط مي باشد؟(نيمسال دوم 58 -68)

الف.كاربردهاي تجاري اسمالتاك زياد نيست. ب.اسمالتاك خاصيت شي گرايي دارد.

ج.هوش مصنوعي از C استفاده مي نمايد. د .هيچكدام **31-**مدلهاي مختلف محاسباتي زبان كدامند؟(نيمسال دوم 58-68) الف.تابعي ، دستوري ب. مبتني بر قاعده ، شي گرا

ج.الف و ب د.بي قاعده ، شي گرا ، تابعي، دستوري

(86- **41-**براي استفاده موثر از استانداردها كدام گزينه بايستي مد نظر باشد؟(نيمسال دوم 58الف.زمان شناسي ب.اطاعت و پيروي ج.الف و ب د.تازگي

(87- **51-**قابليت تعامد به چه معناست؟(نيمسال اول 68

الف.خصوصيتي در زبانهاست كه باعث مي شود بتوان برنامههاي نوشته شده در يك زبان را از ماشيني به ماشين ديگر منتقل كرد.

ب.اگر بتوانيم خصوصيات مختلف از يك زبان را با هم تركيب كنيم و تركيب حاصل نيز با معنا باشد.

ج.اگر بتوانيم ويژگيهاي دو زبان مختلف را با هم تركيب كنيم و يك زبان جديد ايجاد كنيم.

د.موارد ب و ج صحيح است.

(86-87 **61-**در كدام گزينه هر دو زبان، زبانهاي تابعي هستند؟(نيمسال دوم

ب, ML C++, FORTRAN.

د, COBOL LISP,PROLOG. الفLISP.

جPROLOG.

71-كدام يك از زبانهاي زير خيلي زود استاندارد شد؟(نيمسال دوم

(68-78 الف.ADA ب. C ج.FORTRAN

د.هيچكدام **81-**كداميك از موارد زير از اهداف الگول نمي باشد؟(نيمسال اول 78 -88) الف.نشانههاي الگول بايد به رياضيات استاندارد نزديك باشد.

ب.الگول بايد براي توصيف الگوريتمها مفيد باشد.

ج.برنامهها در الگول بايد به زبان ماشين نزديكتر باشد.

د.الگول بايد به معماري يك ماشين مقيد باشد.

(88- **91-**تعريف زير معرف كداميك از محيطهاي عملياتي مي باشد. (نيمسال اول 78

»يك سيستم كامپيوتري كه براي كنترل بخشي از يك سيستم بزرگ مثل ماشين آلات صنعتي، هواپيما ، ماشين تراش , اتومبيل يا مترو بكار مي رود«

الف.دسته اي ب.محاوره اي ج.اشتراك زماني د.سيستم تعبيه شده

(87-88 **02-**با توجه به سه گزاره زير كداميك از گزينهها صحيح است؟(نيمسال اول

مورد اول: برنامه شي گراء با ساختن اشياي داده دقيق كارايي زبانهاي دستوري را بدست مي آورد.

مورد دوم: برنامه شي گراء با ساختن دسته اي از توابع كه از مجموعه اي محدود از اشياي داده استفاده مي كنند قابليت انعطاف زبانهاي تابعي را بدست مي آورد.

مورد سوم: برنامه شي گراء با ساختن دسته اي از توابع كه از مجموعه اي محدود از اشياي داده استفاده مي كنند قابليت اعتماد زبانهاي تابعي را بدست مي آورد.

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

12-در كدام گزينه هر سه زبان در كاربردهاي تصميم گيري مثل هوش مصنوعي استفاده مي شود؟(نيمسال دوم 87-88)

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

ج C++,Lisp,Java.د APL,XML,PERL.

22-يكي از اهداف زبان الگول نزديك شدن به رياضيات محض بود, اين هدف موجب مي شود تا ارسال پارامتر به زير برنامهها به كدام روش صورت گيرد؟(نيمسال دوم 78 -88)

الف. فراخواني با نام ب.فراخواني با نام ج. فراخواني با ارجاع د. فراخواني با مقدار - نتيجه

32-در كداميك از محيطهاي برنامه سازي در ارتباط با I/O ، محدوديتهاي بيشتري نسبت به ساير محيطها وجود دارد؟(نيمسال اول 88 -98)

الف.محيطهاي محاوره اي ب.محيطهاي دسته اي ج.محيطهاي سيستم تعبيه شده

د.محيطهاي اينترنتي

42-كدام يك از موارد زير در مورد سيستمهاي تعبيه شده(Embedded System) صحيح مي باشد؟(نيمسال دوم 78 -88)

مورد اول : برنامهها در اين سيستمها،معمولاً بدون سيستم عامل مربوطه و بدون محيط معمولي فايلها و دستگاههاي I/O اجرا مي شوند.

مورد دوم: سيستمهاي تعبيه شده معمولاً در زمان بي درنگ كار مي كنند.

مورد سوم: قابليت اعتماد و صحت از اهميت چنداني در اين سيستمها برخوردار نيستند.

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

52-كدام يك از موارد زير در مورد برنامههاي شي گراء صحيح است؟(نيمسال دوم 78 -88)

مورد اول: با ساختن اشياء داده دقيق قابليت انعطاف و قابليت اعتماد برنامه نويسي تابعي حاصل مي شود.

مورد دوم: با ساختن دسته اي از توابع به همراه مجموعه اي محدود از فضاي اشياي داده كارايي زبانهاي دستوري حاصل مي شود.

مورد سوم: ابتدا مجموعه اي از توابع طراحي مي شود و سپس اشياي داده پيچيده اي حاصل مي شوند.

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

62-كدام يك از موارد زير جز اهداف مطالعه طراحي و پيادهسازي زبانهاي برنامهسازي نمي باشد؟ (تابستان 88) الف. استفاده بهينه از زبانهاي برنامهسازي موجود ب.آشنايي با اصطلاحات مفيد ساختارهاي برنامهنويسي.

ج .فراگيري برنامه نويسي شي گرا. د .انتخاب بهترين زبان برنامه نويسي.

72-براي زبان PL/I از تركيب كدام زبانها استفاده شده است. (تابستان88)

الف Cobol, Fortran. ب. C, Ada

ج C, Fortran. د .Ada, Lisp

82-كدام يك از گزينههاي زير در مورد سيستمهاي تعبيه شده صحيح نميباشد؟ (تابستان 88) الف: يك سيستم كامپيوتري تعبيه شده اغلب يك سيستم توزيعي است.

ب: قابليت اعتماد و صحت، صفات مهمي براي برنامههاي تعبيه شده است.

ج: C++, Ada براي نوشتن برنامه هاي تعبيه شده مفيد مي باشد.

د: پردازش خطا در سيستمهاي تعبيه شده از اهميت ويژهاي برخوردار نمي باشد .

92-كدام يك از موارد زير جز صفات يك زبان خوب، نميباشد. (در اولويت آخر قرار دارد) (تابستان 88) الف. قابليت تعامد ب. طبيعي بودن براي كاربردها

ج .قابليت گرافيكي د. پشتيباني از انتزاع

03-كدام يك از موارد زير جز مدلهاي محاسباتي زبانهاي برنامهسازي نميباشد؟ (تابستان 88) الف. تابعي ب. مبتني برقاعده ج. محاوره اي د. ش يگرا

13- كدام مورد جزء دلايل مطالعه زبانهاي برنامه سازي نمي باشد. (تابستان 88)

الف. استفاده بهينه از زبان هاي برنامه نويسي موجود ب. شناختن ساختارهاي مفيد زبان هاي برنامه نويسي ج .انتخاب بهترين برنامهسازي براي يك پروژه خاص د. همه موارد فوق صحيح است.

23-كدام يك از زبانهاي زير جزء زبانهاي پردازشي مي باشد.

(تابستان 88) الف. فرترن ب. پرل ج .پاسكال

د. كوبول **33-**زبان برنامه نويسي پرولوگ از نظر كاربرد جزء كدام

يك از زبانهاي برنامه سازي محسوب مي شود؟(نيمسال اول 88-98)

الف. سيستمي ب.تجاري ج.علمي د.هوش

مصنوعي **43-**منظور از قابل تعامد بودن زبان برنامه سازي

چيست؟(نيمسال اول 88-98) الف.يعني امكان تجزيه ويژگيهاي مشابه

زبان وجود داشته باشد.

ب.از تركيب ويژگيهاي مختلف تركيب جديد با معنايي ايجاد شود.

ج.از تجزيه ويژگيهاي مشابه ويژگي جديد با معنايي ايجاد شود.

د.تركيب ويژگيهاي مختلف جهت ايجاد تركيب جديد ميسر نباشد.

53-كدام دسته از مدلهاي زبان برنامه سازي به مدل منطقي نزديكترند. (نيمسال اول 88-

98) الف .زبانهاي دستوري ب .زبانهاي تابعي ج .زبانهاي قانونمند د.

زبانهاي شي گرا

63- منظور از Orthogonality يا خاصيت تعامد در زبانهاي برنامه سازي كدام است؟ (نيمسال

دوم 88 -98) الف.خلاصه سازي چند ويژگي از يك زبان، كه خلاصه سازي با معنا باشد.

ب .مجزاسازي يك ويژگي از زبان به چند ويژگي، كه چند ويژگي جديد با معنا باشند.

ج.تركيب كردن چند ويژگي از يك زبان، كه تركيب جديد بامعنا باشد.

د .تركيب كردن چند ويژگي ازچند زبان مختلف، كه تركيب جديد بامعنا باشد.

73-به منظور جلوگيري از وجود اسامي مشترك در برنامه ، زبانها معمولاً از چه روشي استفاده مي

نمايند؟ (نيمسال دوم 88 -98)

الف.اعلان نوع داده ب.قواعد حوزه ج.كامپايل مجزا د.بين المللي شدن

برنامه نويسي **83-**كداميك از زبانهاي زير براي كاربردهاي جستجو مورد استفاده قرار مي

گيرد؟ (نيمسال دوم 88-98) الف.Prolog ب.Ada ج.-Java

د.Snobal

93-اين نوع دستورات زير به موجب چه عملي مورد استفاده قرار مي گيرند؟ (نيمسال دوم 88-98)

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

الف.نقاط كنترلي ب.كامپايل

مجزا ج.ادعا

د.رديابي اجرا

40- تركيب ويژگيهاي مختلف از يك زبان و دستيابي به يك ويژگي جديد با معنا ، چه نام دارد.(نيمسال اول 89 - 90)

الف. نقطه كنترل breakpoint ب .تعامد orthogonality

ج .تركيب combine د abstraction انتزاع.

41- از ديدگاه پروژههاي نرم افزاري كاهش كدام يك از هزينههاي زير بر روي پروژه اثر مطلوب تري دارد(.نيمسال اول 98 -09)

الف. هزينه نگهداري پروژه ب .هزينه اجراي پروژه ج .هزينه ترجمه پروژه د .هزينه طراحي پروژه

24- مدل محاسباتي تكه كد برنامه زير چيست؟(نيمسال اول 98-09)

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

الف. مدل دستوري ب.مدل تابعي ج .مدل قانونمند د .مدل شيگرا

34- اين نوع دستورات زير به موجب چه عملي مورد استفاده قرار ميگيرند. (نيمسال اول 98-09)

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

الف. نقاط كنترلي ب .كامپايل مجزا ج .ادعا د .رديابي اجرا

سوالات تشريحي

1-هشت مورد از صفات يك زبان خوب را بنويسيد؟(نيمسال دوم 58-68)

1-01- پاسخنامه سوالات تستي فصل اول

سوال	الف	ب	ج	د
1		*		
2		*		
3	*			
4			*	
5		*		
6		*		
7				*
8			*	
9				*
10				*
11				*
12			*	
13			*	
14			*	
15		*		
16	*			
17	*			
18				*
19				*
20				*

سوال	الف	ب	ج	د
21		*		
22		*		
23	*			
24	*			
25				*
26			*	
27		*		
28				*
29			*	
30			*	
31				*
32		*		
33				*
34		*		
35			*	
36			*	
37		*		
38	*			
39			*	
40		*		
41	*			
42	*			
43			*	

فصل دوم :

اثرات معما یر ماشین

ا # ا :

مقدمه كامپيوترهاي مجازي
كامپيوتر و اجزاي آن سلسله مراتب ماشين
مفسرها و معماريهاي مجازي زمانهاي انقياد
ترجمه زمان اجرا تفسيري زمان
مقايسه ترجمه و تفسيري زمان
زبان زبانهاي كامپايلري اهميت
تستي و تشريحي

مجازي كامپيوترهاي ميان افزار انقياد
ترجمه
پياده سازي زبان انواع زبانها زمان تعريف
زمانهاي انقياد زبانهاي مفسري سوالات

2-1- مقدمه

در قديم كامپيوترها گران بودند لذا زبانهاي اوليه بايد به گونه اي طراحي مي شدند كه به صورت كارآمد اجرا شوند. مثلاً زبان فرترن (جهت محاسبات عددي) و زبان ليسپ (جهت پردازش ليست) به گونه اي بودند كه به خوبي به زبان ماشين تبديل مي شدند ولي نوشتن برنامه در آنها سخت بود.

اما امروزه ماشينها ارزان شده اند ولي برنامه نويسي گران شده است. لذا هدف آن است كه به سادگي بتوان برنامه نوشت هر چند كه قدري كند باشد. مثلاً خصوصيات كلاس در C++ ، نوع دادهها در ML و ويژگي Package در Ada، ساخت برنامهها و رفع اشكال آنها را ساده تر كرده اند. سه عاملي كه در هنگام توسعه يك زبان اثر گذارند عبارتند از :

- كامپيوتري كه برنامه روي آن اجرا مي شود.
- مدل اجرا يا كامپيوتر مجازي اي كه آن زبان را روي سخت افزار واقعي پشتيباني مي كند.
- مدل محاسباتي آن زبان.

2-2- كامپيوتر و اجزاي آن

ما در اين درس كامپيوتر را به صورت مجموعه اي از الگوريتمها و ساختمان دادهها تعريف مي كنيم كه توانايي ذخيره و اجراي برنامهها را دارد. طبق اين تعريف كامپيوتر مي تواند سخت افزاري يا شبيه سازي شده نرم افزاري (مجازي) باشد.

كامپيوترهاي سخت افزاري[1] :

كامپيوتر سخت افزاري، كامپيوتري است كه كاملاً از اجزاء سخت افزاري و مدارات الكترونيكي شامل حافظه اصلي ، ثباتها و ALU و ... ساخته شده است. در اين نوع كامپيوترها، دقيقاً سخت افزار مربوط به هر دستور زبان ماشين وجود دارد.

كامپيوترهاي ميان افزار[2]:

يك كامپيوتر به صورت ميان افزار ناميده مي شود در صورتيكه هر دستور زبان ماشين دنباله اي از ريز عمليات[3] مي باشد كه در حافظه قابل برنامه ريزي[4] ذخيره شده است.

هر كامپيوتر مشابه زبانهاي برنامه نويسي از 6 جزء تشكيل شده است :

- **داده[5]7777:** يك كامپيوتر بايد مجموعه اي از دادههاي اوليه (مثلreal و char وint) و دادههاي ساخت يافته (مثل ركورد ، آرايه و...) براي انجام عمليات فراهم كند.

[1] Hard Ware

[2] Firmware

[3] Micro-operation

[4] PROM Data °

- **اعمال اوليه[1]28282828**

: يك كامپيوتر بايد مجموعه اي از عمليات اوليه براي پردازش روي دادهها را داشتهباشد.(مانند دستورات CPU يا زبان ماشين)

- **كنترل ترتيب انجام دستورا ت F2:** يك كامپيوتر بايد مكانيزمي براي كنترل ترتيب اجراي عمليات داشته باشد. (كنترل ترتيب اجراي عمليات اوليه يا تعريف شده توسط كاربر)
- **دستيابي به داده F3 :** يك كامپيوتر بايد مكانيزمهايي براي كنترل دادههايي داشته باشد كه با اجراي عمليات توليد مي شوند. (كنترل انتقال داده بين زير برنامهها و برنامهها)
- **مديريت حافظه F4:** يك كامپيوتر بايد مكانيزمهايي جهت تخصيص حافظه

براي برنامه و داده و همچنين آزاد سازي حافظه داشته باشد.

- **محيط**

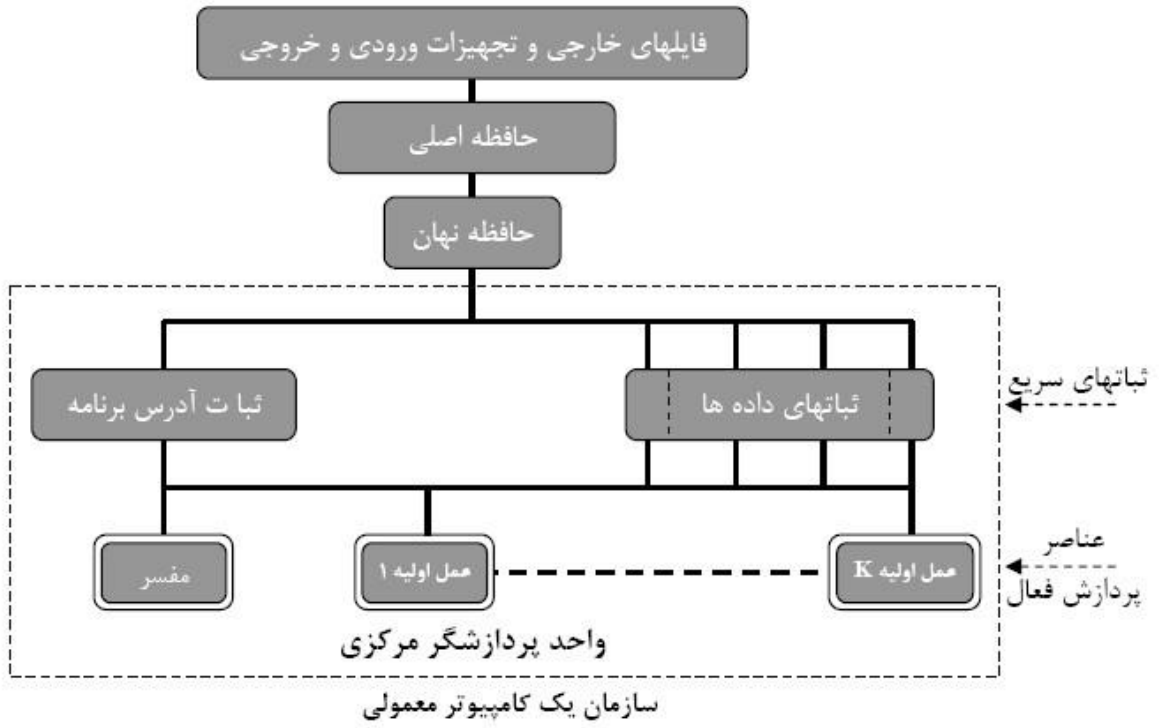
عمل32323232

ياتي^{F_5} **:** يك كامپيوتر بايد مكانيزمهايي براي مبادله اطلاعات با دستگاههاي جانبي فراهم سازد.

سازمان كامپيوتر:

واحد پردازشگر مركزي(CPU) از بخشهاي مهم يك كامپيوتر مي باشد. اين واحد از ثباتهاي سريع و عناصر پردازش فعال تشكيل شده است. ثباتهايي كه وجود دارند ثباتهاي داده و ثبات آدرس مي باشند. ثباتهاي آدرس براي آدرس دهي كردن دادهها و دستورات روي حافظه استفاده مي شوند و ثباتهاي داده هم براي ذخيره سازي دادههاي مورد نياز و نتايج حاصل شده از اعمال اوليه استفاده مي شوند. هر دستورالعمل روي حافظه اصلي مشخص كننده يك هدف مي باشد كه اين عمل توسط مفسر CPUترجمه شده(كد گشايي عمليات) و دستورات لازم به بخشهاي مختلف داده مي شود تا اينكه عمل اوليه بر روي دادهها انجام شود. عناصر پردازش فعال يكCPU از اعمال اوليه اي كه براي آن تعيين شده ، تشكيل شده است اين اعمال اوليه ممكن است در پردازشگرهاي مختلف باشد. سازمان يك كامپيوتر معمولي در شكل زير نشان داده شده است.

Primitive Operation [1]
Sequence Control [2]
Data Control [3]
Storage managment [4]
Oprating Environment [5]



شکل 2 - 1

توضيح اجزاي شش گانه كامپيوتري به طور مفصل تر:

2-2- 1- دادهها

دادهها در حافظه ذخيره مي شوند .در شكل بالا سه جزء اصلي حافظه دادهها نمايش داده شده است.

- **حافظه اصلي :** حافظه اصلي به صورت دنباله اي از بيتهاي خطي سازماندهي مي شوند كه از كلمات با طول ثابت تشكيل شده اند.
- **حافظه نهان**[1] **:** طول ثباتهاي سريع به اندازه طول كلمات است و طوري تقسيم بندي مي شوند كه هر قسمت آن قابل دستيابي باشد. حافظه سريع نهان معمولاً بين حافظه اصلي و ثباتها قرار مي گيرد و مكانيزمي براي دسترسي سريع به دادههاي موجود در حافظه است
- **فايلهاي خارجي (حافظههاي جانبي):** كه بر روي ديسك يا CD يا نوار مغناطيسي ذخيره مي شوند.

2-2- 2- اعمال

۱

كامپيوتر بايد مجموعه اي از اعمال اوليه توكار داشته باشد كه متناظر با كدهاي عملياتي هستند كه به صورت دستورات زبان ماشين مي باشند.

- **اعمال اوليه محاسباتي :** مثل Add , Sub , Mult , Div

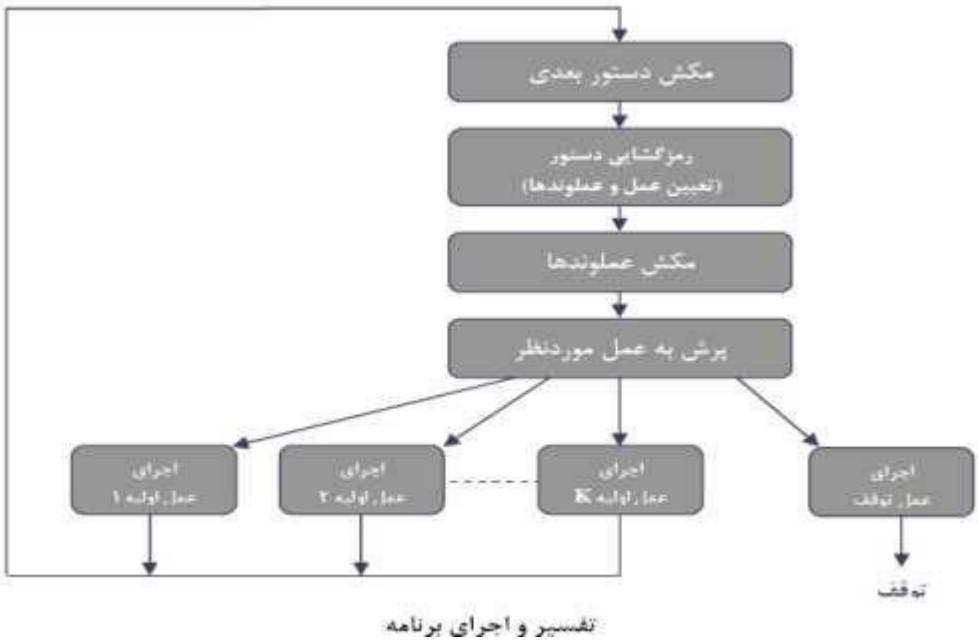
Cache

- **اعمال اوليه براي تست خواصي از عناصر داده :** مثل SPA,SNA,SZA (به ترتيب مقايسهبا صفر و منفي يا مثبت بودن اعداد) • اعمال اوليه براي كنترل دستگاههاي I/O : مثل output,input,skip in,skip out

2-2- 3- كنترل ترتيب

در حين اجراي برنامه دستور بعدي كه بايد اجرا شود توسط محتويات ثبات آدرس برنامه[1] يا PC مشخص مي شود. اين ثبات حاوي آدرس دستور بعدي است و مفسر از آن استفاده مي كند. مفسر قلب عملكرد كامپيوتر است و معمولاً الگوريتمي چرخه اي را اجرا و تكرار مي كند. در هر چرخه مفسر آدرس دستور بعدي را از ثبات آدرس برنامه مي گيرد و دستور مورد نظر را از حافظه مكش مي كند. آن دستور را به يك كد عملياتي و مجموعه اي از عملوندها تبديل مي كند. عملوندها را در صورت لزوم مكش مي كند و عمليات را با آن فراخواني مي كند. اعمال اوليه ممكن است دادههاي موجود در حافظه يا ثباتها را اصلاح كنند. شكل زير عملكرد يك مفسر را نشان مي دهد.

۱



شكل 2 - 2

2-2- 4- دستيابي به دادهها

علاوه بر كد عملياتي ، هر دستور ماشين بايد عملوندهايي را مشخص كند كه آن عمل از آن استفاده مي كند. عملوند ممكن است در حافظه اصلي يا در ثبات باشد. كامپيوتر بايد مكانيزمي براي تعيين عملوندها ، بازيابي آنها و ذخيره نتايج در عملوندها داشته باشد كه به اين امكانات كنترل دستيابي به دادهها گفته مي شود. براي دستيابي به عملوندها، متداولترين راه ، آدرس حافظه يا ثبات است.

Program Counter

2-2- 5- مديريت حافظه

يك اصل در طراحي ماشين اين است كه تمام منابع كامپيوتر مثل حافظه اصلي ، CPU و دستگاههاي جانبي تاآنجايي كه ممكن است فعال باشند. ولي به دليل سرعت متفاوت هر يك از اين منابع ، در اين اصل يك تناقض وجود دارد. براي مقابله با عدم توازن بين دستيابي به دادههاي خارجي و CPU ، سيستم عامل از تكنيك چند برنامه اي استفاده مي كند و تا زماني كه دادههاي مورد نظر از دستگاههاي خارجي خوانده مي شوند وقت CPU به برنامه ديگري اختصاص داده مي شود. براي

برقراري توازن بين حافظه اصلي و CPU از حافظه نهان استفاده مي شود .حافظه نهان يك حافظه كوچك بين CPU و حافظه اصلي است.

2-2- 6- محيط عملياتي

محيط عملياتي كامپيوتر متشكل از مجموعه اي از حافظههاي جانبي و دستگاههاي I/O مي باشد. هر ارتباط كامپيوتر با دنياي خارج از طريق محيط عملياتي صورت مي گيرد. محيط عملياتي شامل : حافظههاي با سرعت بالا مانند Flash ، حافظه با سرعت متوسط مثل Disk و CD ، حافظه كند مانند نوارها و دستگاههاي I/O مانند Printer ، صفحه كليد ، مانيتور و ... مي باشد.

2-3- كامپيوترهاي ميان افزار

كامپيوتر ميان افزار توسط ريز برنامه اي[1] شبيه سازي مي شود كه بر روي كامپيوتر سخت افزار قابل ريزبرنامه نويسي اجرا مي شود. زبان ماشين اين كامپيوتر متشكل از مجموعه بسيار سطح پايين از ريز دستورات است كه انتقال داده را بين حافظه اصلي و ثباتها ، بين خود ثباتها و از ثباتها، از طريق پردازنده انجام مي دهد. ريز برنامه ويژه اي با استفاده از اين مجموعه دستورات نوشته مي شود كه چرخه تفسير و اعمال اوليه گوناگون كامپيوتر مورد نظر را تعريف مي كند. ريزبرنامه عمل كامپيوتر مطلوب را بر روي كامپيوتر ميزبان قابل ريزبرنامه نويسي شبيه سازي مي كند. خود ريزبرنامه در يك حافظه فقط خواندني ويژه ROM در كامپيوتر ميزبان ذخيره مي شود و با سخت افزار كامپيوتر ميزبان با سرعت بالايي اجرا مي شود. كامپيوتري كه از طريق شبيه سازي ريزبرنامه اي بوجود مي آيد كامپيوتر مجازي ناميده مي شود. زيرا توسط ريزبرنامه شبيه سازي مي شود و در صورت عدم وجود اين ريزبرنامه ماشين وجود نخواهد داشت. به اين نكته توجه داشته باشيد كه زبان ماشين به زبان سطح پايين مانند اسمبلي محدود نمي شود. مثلاً مي توان كامپيوتري ساخت كه زبان ماشين C يا ادا باشد كه به آن كامپيوتر مجازي C يا ادا مي گويند و لي ساخت چنين كامپيوتري پيچيده بوده و كارآيي آن نيز كمتر است.

تعريف ديگر: كامپيوترهاي ميان افزار

با داشتن توصيفي از يك زبان برنامه سازي مي توان كامپيوتري كاملا سخت افزاري ايجاد كرد كه زبان ماشين آن كامپيوتر زبان مورد نظر ما باشد. براي مثال مي توان كامپيوتري ساخت كه زبان ماشين آن زبان C باشد ولي اين كامپيوتر داراي هزينه بالا و انعطاف كمتري نسبت به حالتي است كه زبان ماشين مجموعه مختصر و جامعي از

۱

Micro program

۱

دستورات از سطح پايين باشد. در عوض مي توان كامپيوتري ساخت كه اجراي دستورات زبان سطح بالا در آن بهروش ريز برنامه سازي پياده شده باشد به اين معني كه براي هر دستور سطح بالا ريز دستوراتي كه خود به دستوراتسطح پايين آن كامپيوتر تبديل مي شوند وجود داشته باشد كه به اين مدل كامپيوتري ميان افزاري مي گويند.

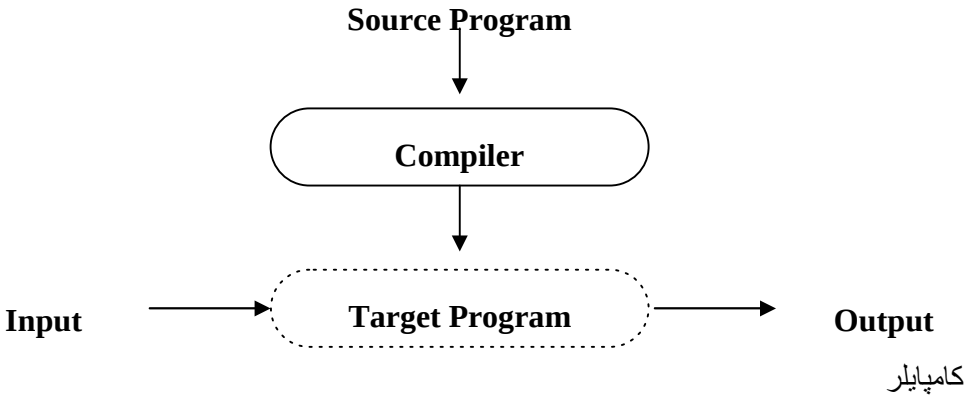
2-4- مفسرها و معماريهاي مجازي

وقتي يك برنامه به يك زبان نوشته مي شود سوالي كه ايجاد مي شود اين است كه اين برنامه زبان سطح بالا چگونه در يك كامپيوتر واقعي صرف نظر از زبان ماشين آن اجرا مي شود. براي اين منظور دو راه حل وجود دارد:

- **روش اول :** ترجمه ، كامپايل كردن (Translation)
- **روش دوم :** تفسيري ،شبيه سازي نرم افزاري (Interpreter)

2-4- 1- روش ترجمه

در اين روش برنامه به زبان سطح بالا طي فرآيندهايي تبديل به زبان ماشين مي شود كه قابل اجرا روي سخت افزار است. به طور كلي مفسر(نرم افزار مترجم) به هر پردازنده زباني گفته مي شود كه برنامه اي به زبان منبع كه مي تواند سطح بالا يا پايين باشد را گرفته و آن را به زبان مقصد تبديل مي كند .

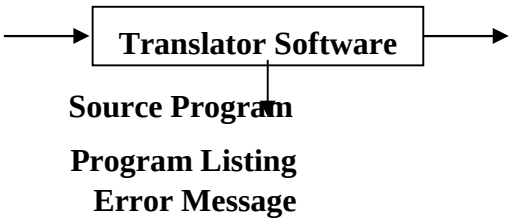


كامپايلر

شكل 2 - 3

در روش ترجمه ابزارهايي مورد نياز است كه هر كدام از اين ابزارها خود يك نوع مترجم مي باشند. مترجم(مفسر) نرم افزاري است كه برنامه به يك زبان مبدا را دريافت كرده و به برنامه معادل در

زبان مقصد تبديل مي كند. در ضمن اگر برنامه به زبان مبدا با ساختار زبان مبدا تطابق نداشته باشد پيغام خطا صادر خواهد شد.



شكل 2 - 4

انواع مترجمها(مفسرها) عبارتند از:

اسمبلر(Assembler) : مفسري مي باشد كه زبان منبع آن زبان اسمبلي و زبان مقصد آن زبان ماشين برايبرنامه واقعي مي باشد.

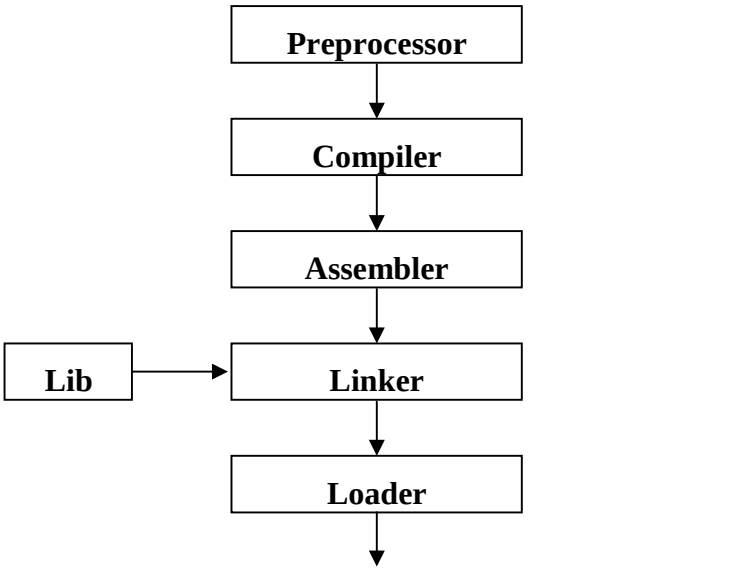
كامپايلر(Compiler) : مفسري مي باشد كه زبان منبع آن يك زبان سطح بالا و زبان مقصد آن نزديك به زبان ماشين(مانند اسمبلي) مي باشد.

بارگننده(Loader) : مفسري مي باشد كه زبان منبع آن زبان ماشين به شكل جابجا پذير (آدرس نسبي) و زبان مقصد آن كد ماشين واقعي است. بارگننده ، ماژولهاي مختلف اجرايي را به هم پيوند داده و آدرسهاي آنها را به صورت مناسب جابجا مي كند.

پيوند دهنده(Linker) : اين مفسر بخشهاي مختلف برنامه را دريافت نموده، آنها را سرهم بندي كرده و برنامه خروجي تقريباً شبيه برنامه ورودي به شكل كامل تر توليد مي كند.

پيش پردازنده يا پردازنده ماكرو(Preproccesor) : مفسري مي باشد كه زبان منبع آن شكل توسعه يافته اي از يك زبان سطح بالا مانند C++ مي باشد و زبان مقصد آن شكل استانداردي از همان زبان مي باشد(همان برنامه C). مثلاً در زبان C دستوراتي كه با # شروع مي شوند مثل تعريف ماكروها يا فايلهاي incloud ابتدا بسط داده شده و به دستوراتي از زبان C تبديل مي شوند.

ترتيب اجراي مفسرها براي ترجمه يك برنامه به شكل زير مي باشد:



شكل 2 - 5

يك مثال براي نمايش عملكرد بار كننده در شكل زير آمده است:

زير برنامه	آدرس كامپايل شده (آدرس نسبي)	آدرس اجرايي (آدرس واقعي)
P	0- 999	0- 999
Q	0- 1999	1000- 2999
توابع كتابخانه	0- 4999	3000- 7999

جدول 2 - 1

برنامه اجرايي برنامه اي است كه از آدرسهاي 0 تا 7999

استفاده كند.

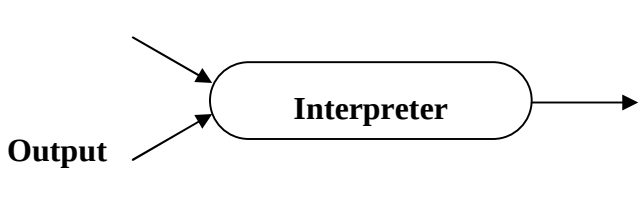
نكته : مراحل ترجمه از زبان سطح بالا به زبان ماشين اغلب بيش از يك مرحله است.

به عنوان مثال :

اسمبلي ← زبان ماشين C++ ← C ←

2-4- 2- روش شبيه سازي نرم افزاري (تفسيري)

در اين روش كد برنامه منبع مستقيماً به شبيه ساز نرم افزاري يا مفسر داده مي شود و مفسر دستورات زبان سطح بالا را تفسير و بلافاصله اجرا مي كند. در اين روش به جاي اينكه زبان سطح بالا به زبان ماشين ترجمه شود به كمك شبيه سازي، آن برنامه روي يك كامپيوتر ميزبان، اجرا خواهد شد. منظور از كامپيوتر ميزبان كامپيوتري است كه زبان ماشين آن يك زبان سطح بالا است.



تفسير نرم افزاري

شكل 2 - 6

2-4- 3- مقايسه روش ترجمه و تفسيري

با آنكه هر دو روش ترجمه و تفسيري برنامههايي به زبان سطح بالا را به عنوان ورودي دريافت مي كنند ولي در موارد زير با هم تفاوت دارند :

1- در روش ترجمه برنامه به طور كامل به زبان ماشين تبديل شده و سپس اجرا مي شود. درحالي كه درروش تفسيري يا شبيه سازي نرم افزاري تك تك دستورات زبان سطح بالا ابتدا تفسير و مجموعه دستورات لازم براي شبيه سازي آن دستور اجرا مي شود.

2 - سرعت اجرا در روش ترجمه يا كامپايلري بيشتر از مفسري يا شبيه سازي است چون در روش ترجمه فاز ترجمه و اجرا جدا از هم هستند، ولي درشبيه سازي فاز ترجمه و اجرا يكسان هستند.

3 - مترجم دستورات برنامه را به ترتيب فيزيكي ورودي پردازش مي كند، ولي شبيه ساز(روش تفسيري) جريان منطقي برنامه را دنبال مي كند.

4 - مترجم هر دستور را فقط يكبار پردازش يا ترجمه مي كند ولي شبيه ساز(روش تفسيري) ممكن است برخي از دستورات را چندبار پردازش كرده مانند حلقه for و حتي برخي از آنها را اصلاً پردازش نكند مثل يك بلوك شرطي كه همواره غلط است. بنابراين مي توان نتيجه گرفت كه در روش كامپايلري اگر خطايي وجود داشته باشد حتمًاً گرفته خواهد شد ولي در روش شبيه سازي به دليل اينكه كنترل منطقي برنامه دنبال مي شود ممكن است بعضي از خطاها ناديده گرفته شوند.

5 - در روش كامپايلري براي n بار اجرا يك ترجمه لازم است ولي در روش تفسيري براي n بار اجرا n ترجمه لازم است. مثال واضح حلقه تكرار مي باشد.

6 -ترجمه محض و شبيه سازي محض دو كرانه اند يعني حالت تئوري دارند. در عمل ترجمه محض به ندرت صورت مي گيرد، مگر در مواردي كه زبان ورودي دقيقاً شبيه زبان ماشين باشد مانند اسمبلي. شبيه سازي محض نيز به ندرت مورد استفاده قرار مي گيرد به جز مواردي مثل زبانهاي محاوره اي يا زبانهاي كنترل سيستم عامل.

اغلب زبانها به صورت تركيبي از ترجمه و تفسيري پياده سازي مي شوند.

7 - برخي از جنبههاي ساختار برنامه بهتر است قبل از اجرا ترجمه شود مثل حلقه اي كه قرار است 1000 بار اجرا شود ولي برخي ديگر از جنبهها بهتر است فقط در زمان اجرا پردازش شوند.

8 - ايراد مهم ترجمه از دست رفتن اطلاعاتي در رابطه با برنامه است مثلاً اگر برنامه به زبان ماشين ترجمه شود و خطايي رخ دهد ، تعيين اينكه كدام دستور زبان منبع اين خطا را ايجاد كرده است سخت است چون حاصل ترجمه به زبان ماشين كه فقط 0 و 1 مي باشد تبديل شده است ولي در روش تفسيري تمام اطلاعات مربوطه موجود است.

برنامه مقصد در كامپايلر بزرگتر از برنامه مبدأ است ولي در روش تفسيري معمولاً برنامه مقصد كوچكتر است.

9 - در روش تفسيري از آنجايي كه دستورات تا زمان اجرا شكل اوليه خود را خواهند داشت لذا چند كپي از آنها نگهداري نمي شود و بدين ترتيب در مقايسه با روش ترجمه در حافظه صرفه جويي مي شود. اما در حال اجرا كل هزينه رمزگشايي بايد پرداخته شود، در مقابل در روش ترجمه چندين فايل داريم كه نتيجه ترجمه در آن ذخيره مي شود .به عنوان يك قاعده كلي مي توا ن گفت كه روش ترجمه زماني استفاده مي شود كه ساختار زبان منبع نمايش مستقيمي در زبان مقصد دارد و لذا تبديل كد چندان سخت نخواهد بود ، در ساير موارد از روش تفسيري استفاده مي شود .

2-5- انواع زبا نها

يك سوال كليدي آن است كه آيا نمايش اصلي برنامه در ضمن اجرا همان نمايش زبان ماشين كامپيوتر واقعي است يا خير؟ كه بر اين اساس دو نوع تقسيم بندي براي زبانها وجود دارد:

2-5- 1- زبانهاي كامپايلري

در اين زبانها ، برنامهها قبل از شروع اجرا، به زبان كامپيوترواقعي ترجمه مي شوند .در حقيقت اين گونه از زبانها از روش ترجمه استفاده مي كنند. به طور خلاصه خواهيم داشت:

- استفاده از روش كامپايلري باعث مي شود برنامهها با سرعت زياد ،اجرا شوند.
- مترجم زبانهاي كامپايلري، تقريباً پيچيده و بزرگ هستند.
- زبانهاي C, C++ , FORTRAN و ADA زبانهاي كامپايلري هستند

2-5- 2- زبانهاي مفسري

در اين زبانها مترجم، كد ماشين كامپيوتر واقعي را توليد نمي كند،بلكه يك شكل مياني از برنامه را پديد مي آوردكه اجراي آن ساده تر از دستورات اوليه است.ولي با كد ماشين فرق دارد. مفسر اين دستورات يك مفسر نرم افزاري است و لذا اجراي آن كندتر از كامپايلري است.مثلا Java شبيه C++ است ،ولي تفسيري است،كه مفسر آن كد مياني به نام بايت كد را براي ماشين مجازي Java ايجاد مي كند .به طور خلاصه خواهيم داشت:

- استفاده از روش مفسري منجر به برنامههايي مي شود كه اجراي آنها كندتر است.
- مترجم زبانهاي مفسري بسيار ساده هستند. • زبانهايي مثل HTML, Basic, ML، Smalltalk, Perl, Postscript, Lisp زبانهاي مفسري هستند.

2-6- كامپيوترهاي مجازي[1]

قبلاً كامپيوتر را بصورت مجموعه اي از الگوريتمها و ساختمان دادهها تعريف كرديم كه قابليت ذخيره و اجراي برنامهها را دارد ،روشهاي ساخت كامپيوتر عبارتند از:

- **ازطريق سخت افزار *(Through a hard ware realization)*** : ساختمان دادهها و الگوريتم -ها به صورت سخت افزاري پياده سازي ميشود.
- **از طريق ميان افزار *(Through firmware realization)*** : ساختمان دادهها و الگوريتمها از طريق ريزبرنامه نويسي براي سخت افزار مناسب ايجاد ميشود.

- **از طريق ماشين مجازي** ***(Through soft ware realization)*** : در اين حالت ساختمان داده -ها و الگوريتمها از طريق برنامه نويسي و زبانهاي ديگر نمايش داده ميشوند.

[1] Virtual Computer

- **از طريق تركيبي** ***(Through a hard ware realization)*** :در اين تكنيك بخشهاي مختلف كامپيوتر مستقيما در سخت افزار و يا به وسيله شبيه سازي نرم افزاري نمايش داده ميشود .

با توجه به موارد فوق، سه عامل منجر به تفاوتهايي در بين پياده سازيهاي يك زبان مي شود:

- اختلاف در امكاناتي كه روي كامپيوتر پايه موجود است.
- اختلاف در مفاهيم پياده سازي كامپيوتر مجازي(VC) كه بطور ضمني در تعريف زبان ملموس است.
- اختلاف در انتخابهايي كه براي پياده سازي و شبيه سازي زبانهاي سطح بالا مي تواند بكار گرفته شود .

2-6- 1- سلسله مراتب كامپيوترهاي مجازي

درعمل يك كامپيوتر مجازي داريم يعني كامپيوتري كه به زباني غيراز زبان ماشين كار مي كند .كامپيوتري مجازي است كه توسط طراح زبان برنامه سازي طراحي مي شود. از ديد برنامه نويس به زبان سطح بالا و در عمل ساختار يك كامپيوتر مجازي را مي توان به صورت سلسله مراتب شكل زير در نظر گرفت.

كامپيوتر مجازي تعريف شده توسط برنامه نويس (پياده سازي توسط زبان سطح بالا)
كامپيوتر مجازي زبان سطح بالا (پياده سازي توسط برنامهها و اجرا توسط OS)
كامپيوتر مجازي سيستم عامل (بابرنامههايي كه بر روي كامپيوتر مجازي يا ميان افزار اجرا ميشوند پياده سازي ميگردد)
كامپيوتر مجازي ميان افزار (بادستورات زبان ماشين پياده سازي شده كه با ريزدستورات توسط كامپيوتر واقعي اجرا ميشود)
كامپيوتر سخت افزار واقعي (توسط اجزا فيزيكي پياده سازي شده است)

جدول 2 - 2 لايههاي كامپيوترهاي مجازي براي برنامه كاربردي وب

در پايين سلسله مراتب كامپيوتر ،سخت افزار واقعي وجود دارد كه برنامه نويس به طور مستقيم با اين كامپيوتر سر و كار ندارد. لايههايي از نرم افزار در بالاي اين كامپيوتر واقعي وجود دارد در بالاي ماشين مجازي زبان C كه توسط كامپايلر زبان C ايجاد شده ،برنامه نويس برنامه اي به نام مرور گر وب را با استفاده از زبان C اجرا مي كند. اين مرور گر ،ماشين مجازي وب را ايجاد مي كند كه مي تواند ساختمان دادههاي اصلي وب و زبان HTML را پردازش كند. در بالاترين سطح اين سلسله مراتب كامپيوتر، برنامه كاربردي وب قرار دارد كه برنامه نويس صفحات وب، با استفاده از ماشين مجازي وب برنامههاي خود را اجرا مي كند .

نكته : نتيجه اي كه از بحث سلسله مراتبي بودن كامپيوترهاي مجازي بدست مي آيد اين است كه داده و برنامه معادل هم هستند يا به عبارتي هميشه نمي توان در استفاده از يك كامپيوتر مجازي بين داده و برنامه تمايز قائل شد .براي مثال اجراي يك فايل برنامه HTML بر روي برنامه مرورگر را در نظر بگيريد. از ديد مرورگر فايل مورد نظر داده محسوب مي شود در حاليكه از ديد برنامه ساز وب(web) آن فايل يك برنامه است. خود برنامه مرورگر نيز از ديد كامپايلري كه آن را توليد كرده است داده خروجي محسوب مي شود و خود آن كامپايلر نيز با وجود اينكه برنامه محسوب مي شود از ديد سازنده آن(كامپايلر توليد كننده آن) داده محسوب مي شود.. در زبانهايي مثل C و Fortran داده و برنامه جداي از يكديگر ذخيره مي شوند ولي در زبانهايي مثل Lisp و Prolog هردو در يكجا ذخيره مي شوند و برنامهها و دادهها مخلوط هستند. فقط فرايند اجرا ، آنها را از هم تفكيك مي كند .

7-2- انقياد و زمانهاي انقياد

هنگام پياده سازي و يا اجراي يك برنامه، عنصري از اين برنامه مي تواند صفتي از مجموعه صفات ممكن را به خود بگيرد به اين عمل انقياد[1] و به زمان انجام اين عمل، زمان انقياد گفته مي شود .زمانهاي انقياد به صورت زير طبقه بندي ميشوند:

7-2- 1- زمان اجرا[2]

اين انقيادها در هنگام اجراي برنامه صورت مي گيرند. مثل انقياد متغيرها به مقاديرشان و انقياد متغيرها به محلهاي خاصي از حافظه .انقيادهاي زمان اجرا به دو دسته ي كلي تقسيم مي شوند :

- **در هنگام ورود به زير برنامه :** به عنوان مثال هنگام صدا زدن تابع در زبان C يا Pascal انقياد پارامترهاي مجازي به واقعي و انقياد پارامترهاي مجازي به محلهاي ازحافظه.

- **در نقطه خاصي از اجراي بر نامه :** برخي از انقيادها در حين اجرا ، در نقطه خاصي از بر نامه انجام مي پذيرند. مانند انقياد متغيرها به مقاديرشان توسط دستور انتساب يا انقياد اسامي متغيرها به محلهايي از حافظه در هر نقطه اي از برنامه مثلاً در زبان ML و Lisp.

2-7- 2- زمان ترجمه[3]

اين انقيادها در زمان ترجمه رخ مي دهند و به سه دسته تقسيم مي شوند :

- **توسط برنامه نويس :** كه در هنگام نوشتن برنامهها توسط برنامه نويس انجام مي شود مانند اسامي متغيرها ،نوع متغيرها و ساختار دستورات.
- **توسط مترجم زبان :** بعضي انقيادها توسط مترجم زبان صورت مي گيرد.مثل انتخاب محل نسبي داده در حافظه اي كه به زيربرنامه اختصاص داده مي شود يا چگونگي ذخيره سازي آرايهها(سطري يا ستوني) كه از ديد كاربر پنهان است ،توسط مترجم صورت مي گيرد.
- **توسط بارگننده :** هنگامي كه برنامهها متشكل از چند زير برنامه هستند هنگام بار كردن آنها در حافظه

،آدرس متغيرهاي موجود در زيربرنامهها،بايد به آدرس واقعي در كامپيوتر انقياد شوند .

---

[1] Binding

[2] Run time

[3] Translation or compile time

2-7- 3- زمان پياده سازي[1]

برخي از ويژگيهاي يك زبان ممكن است در پياده سازيهاي مختلف آن متفاوت باشد. پياده سازي زبان با توجه به امكانات سخت افزاري مي باشد به عنوان مثال نمايش اعداد ، اعمال محاسباتي، محاسبات رياضي وغيره در اين محدوده جاي مي گيرند يا محدوده مقادير اعداد Short int در پياده سازيهاي مختلف زبان C ممكن است متفاوت باشند. مثلاً در يك ماشين يا كامپيوتر Short int ، 8 بيتي و در ماشين ديگر 61 بيتي باشد.

2-7- 4- زمان تعريف يا طراحي زبان[2]

اغلب ساختارهاي زبانهاي برنامه نويسي، شكلهاي مختلف دستورات ، انواع متغيرها ، انواع ساختمان داده و غيره مواردي هستتند كه در زمان تعريف زبان معين ميشوند. مثلاً متغيرهاي i,j,…,n در فرترن به طور پيش فرض از نوع Integer است.

پاسخ سوالات زير در زمان تعريف زبان ميباشد :

چه انواعي داريم ؟ ثابتها كدامند؟ عملگرها كدامند؟ شكل ظاهري عملگرها Syntax دستورات؟ ساختار برنامه؟ فرم دستوارت و عملي كه هر دستور انجام ميدهد؟ توابع از پيش تعريف شده (توابع كتابخانه اي)

به عنوان مثال ، متغيرهايي كه با حرف i و j در زبان فرترن شروع ميشوند همگي از نوع Integer هستند.

نكته : مجموعه مقادير ممكن براي يك متغير از هر نوع در زمان پياده سازي مشخص مي شود و ممكن است اين مجموعه مقادير در پياده سازيهاي مختلف تفاوت داشته باشد اينكه يك نوع در زبان برنامه نويسي موجود باشد يا نباشد در زمان تعريف زبان، اينكه يك متغير در تعريف برنامه از چه نوعي باشد در هنگام ترجمه زبان و مقدار يك متغيير در هر لحظه در هنگام اجراي زبان مي باشد.

مثال : تمام انقيادهاي دستور زير در زبان L را بررسي كنيد :

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

- **مجموعه اي از انواع ممكن براي X :**

مجموعه اي از انواع قابل قبول براي متغيير X در زمان تعريف زبان مشخص مي شود مثلاً در Pascal مي توان از انواع Char، Integer ،Set ،Boolean و غيره استفاده كرد .

---

[1] Language implementation time

[2] Language definition time

نكته :اگر زبان برنامه نويسي به كاربر اجازه دهد كه خودش انواع جديدي تعريف كند آنگاه مجموعه اي از انواع ممكن در زمان ترجمه مشخص مي شود. به عنوان مثال در زبانهاي C ، Pascal وAda كه نوع شمارشي در زبان Pascal توسط برنامه نويس تعريف مي شود .

- **نوع متغير x:**

معمولاً در زمان ترجمه مشخص مي شود به عنوان مثال در Pascal براي اين منظور يك متغير بايد اعلان شود **نكته :** در برخي از زبانها مثل Smalltalk و Prolog نوع داده ممكن است در زمان اجرا مشخص شود ،بطوري كه انتساب مقداري از يك نوع به X , نوع X را مشخص مي كند در اين

زبانها ممكن است در قسمتي از برنامه X از نوع صحيح ودر جاي ديگر مقداري كاراكتري داشته باشد.

- **مجموعه اي از مقادير ممكن براي متغير X:**

مجموعه مقادير ممكن براي متغيير X در زمان پياده سازي مشخص مي شود. در زمان پياده سازي زبان تعداد بيتها براي قرار دادن يك مقدار از نوع Float تعيين مي شود لذا مجموعه دقيقي از مقادير ممكن براي X بوسيله اين تعداد بيتها مشخص مي شود مثلاً اگر 16 بيت براي يك متغير از نوع صحيح در نظر گرفته شود ،مجموعه مقادير ممكن براي Xاز $2^{16} - 1$ تا $2^{16} - 2$ مي باشد.

- **مقدار متغير X:**

در هر نقطه از اجراي برنامه مقدار خاصي بهX ، مقيد مي شود .يعني مقدار متغير X در زمان اجرا مشخص مي شود .

- **نمايش مقدار ثابت 01 :**

انتخاب نمايش دهدهي در متن داخل برنامه يعني (01 براي ده) در زمان تعريف زبان. (اگر به صورت بيتي نمايش دهيم :) انتخاب رشته اي از بيتها براي نمايش داخلي در زمان پياده سازي (1010).

- **عملگر + :**

انتخاب نماد + براي نمايش عمل جمع در زمان تعريف زبان انجام مي شود ولي معناي عملگر + براي انجام عمل جمع در زمان ترجمه مشخص مي شود .بدليل اينكه پس از مشخص شدن نوع عملوندها ،تعيين مي شود كه علامت + چه جمعي را انجام دهد.(جمع دو عدد صحيح يا جمع دو عدد اعشاري).

اهميت زمانهاي انقياد :

بسياري از تفاوتهاي بين زبانهاي برنامه نويسي، در واقع به تفاوت زبانها در زمان انقياد بر مي گردد و اغلب وابسته به اين است كه انقياد در زمان ترجمه صورت مي گيرد يا در زمان اجرا. به عنوان مثال در زبان Fortran كار كردن با آرايههاي بزرگ ، ساده ولي در ML ، مشكل است.علت اين موضوع آن است كه در Fortran اغلب انقيادها در زمان ترجمه و در ML اغلب انقيادها در زمان اجرا صورت مي پذيرد. بنابراين Fortran انقيادها را فقط يكبار در زمان ترجمه انجام مي دهد.در حاليكه ML بيشتر وقت خود را صرف ايجاد و حذف انقيادها در زمان اجرا مي كند .بنابراين سرعت محاسبات در Fortran بيشتر از ML است. از سوي ديگر، انعطاف پذيري ML در دستكاري رشتهها بيشتر از Fortran است چرا كه در زبان Fortran اندازه رشتهها در زمان ترجمه بايد مشخص و

معين باشد ولي در ML اينگونه انقيادها مي توانند تا خواندن رشته از ورودي به تعويق بيفتد. بنابراين عموماً كارايي (يا سرعت) يك زبان با انعطاف پذيري آن نسبت عكس دارد .بنابراين زمان انقياد مي تواند روي انعطاف پذيري و سرعت برنامه مؤثر باشد.

اگر عمل انقياد در حين اجرا مشخص شود انقياد ديررس[1] گفته مي شود .واگر عمل انقياد در حين ترجمه مشخص شود انقياد زود رس[2] گفته مي شود .

در زبانهاي Fortran,c, Pascal اغلب انقيادها در زمان كامپايل يا ترجمه انجام مي شود ولي در زبانهايي مثل Lisp, ML اغلب انقيادها در زمان اجرا صورت مي گيرد. در زبان Ada كه هم براي قابليت انعطاف و هم براي كارايي طراحي شده است برنامه نويس مي تواند زمان انقياد را انتخاب كند.اغلب، تغييرات كوچك در يك زبان برنامه نويسي منجر به تغييرات بزرگي در زمان انقياد مي شود مثلاً در Fortran90 استفاده از بازگشتي منجر به تغييرات عمده اي در زمان انقياد ويژگيهاي Fortran شد .

انواع زبانها بر اساس زمان مقيد سازي

1. زبانهايي با انقياد زودرس (EBT) : كارايي بالا ، سرعت بالا ، انعطاف پايين مانند زبانهاي Fortran , C , Pascal و ... كه انقياد در آنها درزمان ترجمه انجام ميشود .

(معمولا كامپايلري است)

2. زبانهايي با انقياد ديررس (LBT):

مانند زبانهاي Basic , Prolog , Lisp , ML كه اغلب انقياد در آنها در زمان اجرا انجام ميشود .

(معمولا مفسري هستند)

درزباني مثل Ada كه هم براي قابليت انعطاف و هم براي كارايي طراحي شده ميتوان زمان انقياد را انتخاب كرد .

به طوركلي ميتوان نتيجه گرفت:

- انقياد زودرس(Early binding)– ترجمه- سرعت و كارايي بالا- انعطاف پذيري پايين
- انقياد ديررس (Late binding) - اجرا - سرعت و كارايي پايين- انعطاف پذيري بالا

Late binding [1]
Early binding [2]

2-8- سوالات فصل دوم

سوالات تستي

1-جزئيات مربوط به نمايش اعداد و اعمال محاسباتي در كداميك از موارد زمانهاي انقياد زير مشخص مي شود؟(نيمسال اول 58-68)

الف.زمان اجرا ب.زمان ترجمه ج.زمان پياده سازي زبان

د.زمان تعريف زبان

2-كدام جمله صحيح نيست؟(نيمسال اول 58 - 68) الف.ترجمه محض و شبيه سازي محض دو كرانه اند.

ب.در فرترن انقياد دير رس باعث شده است كه سرعت آن براي محاسبات رياضي از ML بيشتر باشد.

ج.تغييرات كوچكي در يك زبان باعث تغييرات بزرگي در زمانهاي انقياد مي شود.

د .انقياد در ورود به زيربرنامه يا بلوك،يك انقياد زمان اجرا است.

3-كدام گزينه غلط است؟(نيمسال دوم 85-68)

الف.براي برقراري توازن بين حافظه اصلي و پردازنده مركزي از حافظه نهان استفاده مي شود.

ب.محيط عملياتي كامپيوتر متشكل از مجموعه اي از حافظه جانبي و دستگاههاي ورودي و خروجي است.

ج.يك اصل در طراحي ماشين اين است كه تمام منابع كامپيوتري تا آنجا كه ممكن است فعال باشند.

د.مفسر معمولاً الگوريتمي غير چرخه اي را اجرا مي كند.

4-كدام زبان كامپايلري مي باشد؟(نيمسال دوم 58-68)

الف.ADA ب. ,LISP

PROLOG ج.الف و ب

د.اسمالتاك وML 5 -روشهاي ساخت كامپيوتر كدامند؟(نيمسال دوم 58-68)

الف.از طريق سخت افزار ب.از طريق
ماشين مجازي ج.از طريق تركيبي د.
همه موارد

6- binding متغيرها به مقاديرشان و متغيرها به محلهاي خاصي از حافظه به ترتيب جزء كدام دسته از انقيادها است؟(نيمسال اول 68 -78)

الف.زمان اجرا-زمان ترجمه ب.زمان اجرا-زمان
پياده سازي ج.زمان اجرا -زمان اجرا د.زمان
پياده سازي -زمان اجرا

7-زبانهاي Early binding مناسبترند يا زبانهاي Late binding؟(نيمسال اول 68-78) الف.از ديد انعطاف پذيري زبانهاي Later binding انعطاف پذيرتر و مناسب تر هستند.

ب.از ديد سرعت اجرا زبانهاي Early binding سريعتر اجرا شده و مناسب تر هستند.
ج.از ديد سرعت اجرا و انعطاف پذيري زبانهاي Early binding مناسب تر هستند.
د.موارد الف و ب صحيح هستند.

8- براي جمله x:=x+10 مجموعه اي از انواع ممكن براي متغير x در كداميك از زمانهاي انقياد مشخص مي شود؟(نيمسال دوم 68-78)

الف.زمان تعريف زبان ب.زمان پياده سازي زبان ج.زمان اجرا
د.زمان ترجمه **9-**كدام گزينه صحيح است؟(نيمسال دوم 68 -78)

الف.انقياد زودرس به دنبال كارايي و انقياد ديررس به دنبال قابليت انعطاف است.
ب.انقياد زودرس به دنبال قابليت انعطاف و انقياد ديررس به دنبال كارايي است.
ج.هر دو انقياد به دنبال كارايي هستند.
د.هر دو انقياد به دنبال قابليت انعطاف هستند.

01- كدام دسته از زبانهاي زير به عنوان زبانهاي كامپايلري شناخته مي شوند. (نيمسال اول 78-88)

الف ML,Llisp,C++,C.ب Ada,Pascal,C++,C.
ج Prolog,ML,Java,Lisp.د Smaltalk,Lisp,Fortran,Pascal.

11- براي دستور انتساب x:=x+y ، انقياد هر يك از موارد ذيل در چه زماني صورت مي گيرد؟(نيمسال اول 87 - 88)

مورد اول:مجموعه اي از انواع ممكن براي متغير x **مورد دوم:**مقدار متغير x **مورد سوم:**نوع متغير x

الف.مورد اول مي تواند در زمان تعريف يا زمان ترجمه باشد، مورد دوم در زمان اجرا و مورد سوم در زمان ترجمه مي باشد.

ب.مورد اول در زمان تعريف، مورد دوم در زمان اجرا و موردسوم در زمان ترجمه مي باشد.

ج.مورد اول در زمان پياده سازي مورد دوم در زمان اجرا و مورد سوم در زمان ترجمه مي باشد.

د.مورد اول در زمان اجرا ، مورد دوم در زمان ترجمه و مورد سوم در زمان پياده سازي مي باشد.

21- در كدام گزينه اغلب انقيادها در آن زبانها ديررس هستند؟(نيمسال اول 78 -88)

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

31-در زبانهاي كامپايلري براي بررسي نوع در زمان اجرا چه نوع تمهيداتي بايد صورت گيرد؟ الف.دستوراتي توسط برنامه نويس بايد به برنامه اضافه شود.

ب.تمهيداتي لازم نمي باشد و سيستم عامل اين وظيفه را انجام مي دهد.

ج.توصيف گرهايي كه به عنوان بخشي از پياده سازي اشيا داده اي در نظر گرفته مي شوند.

د.نمي توان بررسي نوع در زمان اجرا داشت.

41- براي دستور انتساب X:=X+1476, انقياد هر يك از موارد ذيل در چه زماني صورت مي گيرد؟(نيمسال دوم 87-88)

مورد اول: نمايش مقدار ثابت 1476 **مورد دوم:** خواص عملگر + **مورد سوم:** نوع متغير x

الف. مورد اول مي تواند در زمان اجراي برنامه،مورد دوم در زمان تعريف زبان و مورد سوم در زمان ترجمه باشد.

ب .مورد اول در زمان پياده سازي زبان،مورد دوم در زمان تعريف زبان و مورد سوم در زمان اجرا باشد.

ج .مورد اول در زمان تعريف، مورد دوم در زمان پياده سازي زبان و مورد سوم در زمان اجرا باشد.

د .مورد اول در زمان ترجمه، مورد دوم در زمان اجرا و مورد سوم در زمان پياده سازي باشد.

51- منظور از داده تو كار(Built-in Data) چيست؟(نيمسال دوم 78-88)

الف: دادههايي كه توسط كلاسها تعريف ميشوند.

ب: دادههايي كه مستقيما توسط سخت افزار تعريف مي شوند.

ج: دادههايي كه در زير برنامههاي باز گشتي تعريف ميشوند.

د: دادههايي كه در يك سيستم توزيع شده تعريف ميشوند.

61- شكل زير بيان كننده عملكرد و نقش كداميك از اجزاي مشاركت كننده در فرآيند ترجمه و اجراي زبان است؟(نيمسال اول 88 -98)

زير برنامه	آدرس كامپايل شده	آدرس اجرايي
A	100-400	2100-2400
B	100-300	2401-2601
C	50-600	2602-3152

الف.اسمبلر ب.پيش پردازنده ج.بار كننده
د.كامپايلر

71- چگونگي ذخيره آرايهها و توصيف آنها توسط كداميك از نقشهاي زير مشخص مي شود؟(نيمسال اول 88 -
89)

الف.مترجم ب.بار كننده ج.برنامه نويس د.ويراستار پيوند

81- زمان انقياد مجموعه اي از انواع ممكن براي متغير X كداميك از موارد زير است؟(نيمسال اول 88 -98) الف.زمان اجرا ب.زمان ترجمه ج.زمان پياده سازي د.زمان تعريف زبان

91- در عبارت a:=a/2 زمان انقياد عدد 2، بر اساس ميزان حافظه اشغالي آن چيست؟ (نيمسال دوم 88-98) الف. زمان تعريف زبان ب. زمان اجرا

ج .زمان پياده سازي د .زمان تعريف زبان و زمان پياده سازي زبان

02- در صورتي كه داشته باشيم نوع int در زبان C درسيستم 61 بيتي محدوده ت32768 - تا 32767 خواهد بود ، به انقياد در چه زماني برميگردد(.نيمسال اول 98 -09)

الف. تعريف زبان ب .پياده سازي ج .اجرا د .ترجمه

12- در كدام يك از زبانهاي زير عمليات روي رشتهها با قابليت انعطاف بالا طراحي شده است. (نيمسال اول 89 -
90)

الف. فرترن ب .كوبول ج .ام ال د. ادا

سوالات تشريحي

1-دستور زير را در نظر بگيريد.

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

انواع انقياد و زمانهاي انقياد را براي اين دستور مشخص نمائيد. (نيمسال اول 58-68)

3-زمانهاي انقياد را نام برده هر كدام را توصيف نمائيد؟(نيمسال دوم 58 -68)

4-زمانهاي انقياد را براي مجموعه دستورات زير مشخص نماييد. (نيمسال دوم 88-98)

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

5- زمان انقياد موارد زير را مشخص نماييد. (نيمسال اول 98-09)

الف. مجموعه اي از انواع قابل قبول براي متغيرها مانند real و integer و غيره ب .نوع متغيرها

ج .مقدار متغيرها (مقيد كردن مقدار خاصي به متغير) د .مجموعه اي از مقادير ممكن براي يك نوع متغير

2-9- پاسخنامه سوالات تستي فصل دوم:

سوال	الف	ب	ج	د
1			*	
2		*		
3				*
4	*			
5				*
6			*	
7				*
8	*			
9	*			
10		*		
11		*		
12	*			
13			*	
14		*		
15		*		
16			*	
17	*			
18				*
19			*	
20		*		
21			*	

فصل سوم:

اصول ترجمه زبان

ا # ا :

نحو و معناي زبان تحليل معنايي
معيارهاي عمومي نحو زبان توليد كد
سازي كد قابليت نوشتن توليد
سهولت بازرسي خطا پرداز و
مثال از فازهاي كامپايلر عدم وجود
عناصر نحوي زبان مفهوم گذر مراحل
تحليل لغوي سوالات تستي و تشريحي
تحليل نحوي

مياني قابليت خوانايي بهينه
كد نهايي
جدول نمادها سهولت ترجمه يك
ابهام ابتدا و انتهاي كامپايلر
ترجمه حساب λ

3-1- نحو و معناي زبان

نحو، يعني آرايش واژهها به عنوان عناصري از يك دنباله كه رابطه بين آنها را نشان مي دهد. به عبارت ديگر ،ترتيب نمادها را براي ايجاد يك برنامه معتبر را مشخص مي كند. به عنوان مثال دستور x = y + 2 دنباله معتبري از نمادها را نشان مي دهد ولي2 = + xy دنباله معتبري در زبان C نيست.

به طور كلي مي توان گفت هر زباني از دو قسمت تشكيل شده است: 1- نحو[1] 2 - معنا[2] ،كه نحو ، ساختار بين جملات را مشخص مي كند و معنا ، مفهوم بين جملات را تعيين مي كند.

3-2- معيارهاي عمومي نحو زبان

ويژگيهاي نحوي زبانهاي خوب عبارتند از :

3-2- 1- قابليت خوانايي[3]

اگر ساختار الگوريتم برنامه و دادههاي برنامه به خوبي مشخص باشد ، آن برنامه قابليت خوانايي دارد و به آن برنامه خود استنادي[4] هم مي گويند يعني اين برنامه بدون مستندات جداگانه قابل درك

است. زبان كوبول قابليت خوانايي بالايي دارد در برنامههاي نوشته شده به اين زبان ،ساختارهايي كه كار يكساني انجام مي دهند مشابه اند و ساختارهايي كه كارهاي متفاوتي انجام مي دهند مختلف هستند. زبانهايي با ساختارنحوي اندك قابليت خوانايي كمتري دارند مثلاً APLوSNOBOL4 فقط يك فرمت دستور دارند و خوانايي آنها كم است.

2-2-3 - قابليت نوشتن[5]

خصوصيات نحوي يك زبان كه نوشتن برنامه را ساده تر مي كند اغلب با خصوصيات نحوي كه خوانايي را بيشتر مي كند در تضاد هستند قابليت نوشتن با استفاده از ساختارهاي منظم و دقيق حاصل مي شود در حاليكه ساختارهاي طولاني براي قابليت خواندن مفيد هستند مثلاً در زبان C برنامههاي دقيقي نوشته مي شوند كه ويژگيهاي مفيد متعددي دارند كه باعث مي شود قابليت نوشتن افزايش يابد ولي قابليت خوانايي آن كمتر است.

قواعدي كه اجازه مي دهند اعلانها و عمليات تعيين نشده اي در زبان وجود داشته باشند مثل تعريف آرايه با طول متغير كه نوشتن برنامه را ساده تر مي كنند ولي خواندن آن را مشكل تر مي كنند. به عنوان مثال در زبان فرترن به طور پيش فرض متغييرهايي كه با حروف I,J,K ,…,N شروع مي شوند اگر اعلان نشوند از نوع صحيح خواهند بود كه اين كار نوشتن برنامه توسط برنامه نويس را آسان مي كند ولي قابليت خوانايي آن به دليل عدم اعلان متغيير كاهش مي يابد. برخي ويژگيهاي ديگر مانند استفاده از دستورات ساخت يافته هر دو قابليت خواندن و نوشتن را افزايش مي دهند.

---

[1] Syntax

[2] Semantic

[3] Readability

[4] Self-documentig

[5] Writeability

3-2-3 - سهولت بازرسي (تست)

صحت برنامه يا بازرسي برنامه با قابليت خوانايي و قابليت نوشتن در ارتباط است. زباني خوب است كه تست كردنصحت آن ساده تر باشد.

3-2- 4- سهولت ترجمه

قابليت خوانايي و نوشتن ملاكهايهاي مورد نظر برنامه نويس مي باشند، در حاليكه سهولت ترجمه ، نياز مترجمي است كه قرار است برنامه نوشته شده را پردازش وترجمه كند. اين خاصيت با ويژگي قابليت خوانايي ونوشتن نسبت عكس دارد. هرچه نظم موجود در ساختار زبان بيشتر باشد ، سهولت ترجمه آن بيشتر است به عنوان مثال ترجمه زبان ليسپ ساده است چون نظم ساختاري و قواعد ساده اي دارد در حالي كه قابليت خواندن و نوشتن آن كم است. هنگامي كه ساختارهاي نحوي در يك زبان بيشتر باشند ترجمه آن نيز سخت تر مي شود كه نمونه آن زبان كوبول است.

3-2- 5- عدم وجود ابهام

يك زبان خوب نبايد شامل دستورات مبهم باشد مثلاً در دستور if تودرتوي زير در يك زبان فرضي :

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

در دستور بالا معلوم نيست كه آيا else مربوط به if اولي است يا مربوط به if دومي. در زبان C وپاسكال ابهام مذكور با اين قانون بر طرف شده است كه هر else به نزديك ترين if بر مي گردد و در مثال فوق else مربوط

به if (e2) است. براي درك بهتر مثال فوق

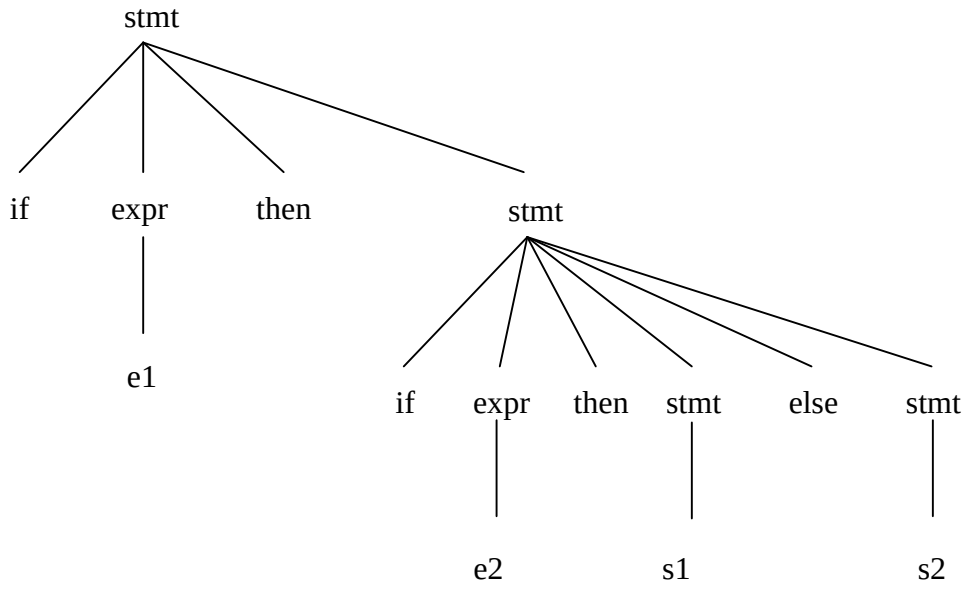
گرامر زير را در نظر بگيريد:

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

49494949**ابهام**^{F_1} : بدين معني است كه ممكن است از روي يك گرامر چند درخت پارس متفاوت براي يك جمله ساخته شود .

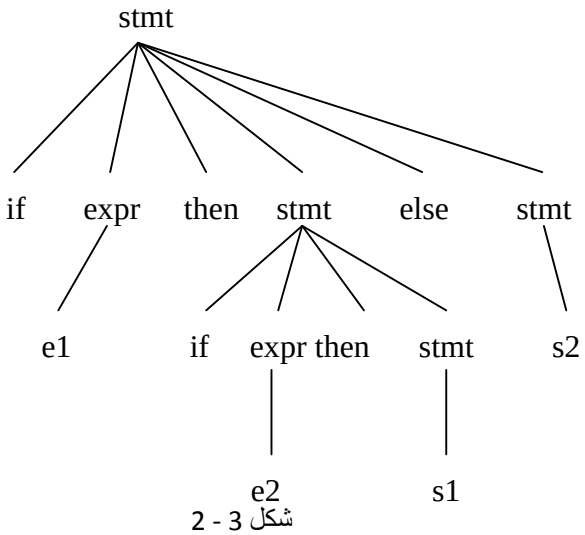
دو درخت پارس متفاوت براي جمله روبرو وجود دارد: If e1 then if e2 then s1 else s2 در اين حالت else مربوط به if دومي است

[1] ambiyuity



شكل 3 - 1

دراين حالت else مربوط به if اولي است



شكل 3 - 2

همان طور كه از قبل مي دانيد براي رفع ابهام در عبارات رياضي يا محاسباتي از اولويت[1] براي عملگرهايي با اولويت متفاوت و از شركت پذيري[2] وبراي عملگرهايي با اولويت يكسان استفاده مي كرديم. براي رفع ابهام در دستورات معمولاً از بازگشتي از چپ و فاكتورگيري از چپ استفاده مي شود كه براي گرامر فوق مي توان از عمل فاكتور گيري چپ استفاده كرد.

[1] Precedence
[2] Associative

عمل فاكتور گيري از چپ

اگر براي يك Non- ترمينال (متغير) قواعدي وجود داشته باشد كه با جملات يكساني شروع شود مي توان ازبخش شروع يكسان فاكتور گيري كرد كه به آن فاكتور گيري از چپ گفته مي شود و قسمت غير مشترك را تحت يك اسم ديگر مي نويسيم.

$$\text{then stmt}' \begin{cases} \text{stmt} \rightarrow \text{if expr then stmt}| \\ \text{stmt} \rightarrow \text{if expr} \quad \text{فاكتورگيري} \end{cases}$$

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

مثالي ديگر از ابهام اين است كه در زبان فرترن نماد A(i,j) هم براي فراخواني تابع و هم برابر ارجاع آرايه استفاده مي شود و در فرترن براي رفع اين ابهام قرارداد شده است كه اگر آرايه اي به نام A، اعلان نشده باشد منظور از A(i,j) فراخواني تابع است. اين ابهام در زبان پاسكال وجود ندارد چون در پاسكال براي ارجاع آرايه به جاي پرانتز از براكت استفاده مي شود. مانند A[i,j] و از اين نظر پاسكال بهتر از فرترن است.

نكته : معيارهاي ارزيابي زبان برنامه نويسي و فاكتورهايي كه بر آن تاثير مي گذارند در جدول 3 - 1 نشان داده شده است.

فاكتورهاي تأثيرگذار	قابليت خوانايي (readability)	قابليت نوشتن (writeability)	قابليت اطمينان (reliability)
طراحي گرامر زبان	×		×
ساختارهاي كنترل	×	×	×
انواع و ساختمان دادهها	×	×	×
سادگي و قابليت تعامد	×	×	×
پشتيباني از تجريد		×	×
قابل بيان بودن		×	×
كنترل نوع			×
كنترل استثناها و خطاها			×
محدود بودن نام گذاري مستعار			×

3 جدول 1 -

3-3- عناصر نحوي زبان

مهمترين عناصر نحوي يك زبان عبارت اند از :

3-3- 1- كاراكترها

اولين كار در طراحي نحو يك زبان انتخاب مجموعه كاراكترهاي (الفباي زبان) آن است. در گذشته براي نمايش كاراكترها از 8 بيت استفاده مي كردند كه توانايي نمايش 256 حالت را داشت و براي حروف بزرگ وكوچك انگليسي كه 25 كاراكتر بود و نمادهاي ديگر كفايت مي كرد ولي با بين المللي شدن صنعت كامپيوتر و پشتيباني آن از زبانهاي مختلف دنيا به نظر مي رسد 652 حالت كافي نيست و به جاي 8 بيت از 61 بيت براي نمايش كاراكترها استفاده مي كنند.

3-3- 2- شناسهها[1]

در اغلب زبانها شناسهها رشته اي از حروف و ارقام است كه با حرف شروع مي شوند. در فرترن اوليه طول شناسه حداكثر 6 كاراكتر بود كه اين امر خوانايي برنامه را كاهش مي دهد.

3-3- 3- نمادهاي عملگرها

اغلب زبانها از كاراكترهاي + و/ و *و - براي اعمال محاسباتي استفاده مي كنند. در برخي از زبانها مانند ليسپ براي اعمال اوليه از شناسهها استفاده مي شود مثل plus وtimes درفرترن براي تساوي .EQ. وبراي توان از ** استفاده مي شود.

3-3- 4- كلمات كليدي و كلمات رزروي

كلمه كليدي شناسه اي است كه به عنوان بخش ثابتي از نحو يك دستور استفاده مي شود مثل كلمه كليدي "if" كه بخش ثابتي از نحو يك دستور است. اگر كلمه كليدي توسط برنامه نويس به عنوان يك شناسه (اسم متغير و يا تابع) قابل استفاده نباشد به آن كلمه رزروي گفته مي شود.اكثر زبانها از كلمات رزروي استفاده مي كند تا خطايابي ساده تر شود. تنها ايراد كلمات رزروي ، توسعه زبان و ايجاد كلمات رزروي جديد است اضافه كردن كلمه رزروي جديد به يك زبان به اين معنا است كه برنامههاي قديمي كه از آن كلمه به عنوان متغير استفاده مي كردند ديگر از نظر نحوي درست نمي باشند و اين يك مشكل است.

3-3- 5- كلمات اضافي[2]

كلماتي اختياري هستند كه جهت افزايش قابليت خوانايي زبان در دستورات قرار مي گيرند. زبان كوبول كلمات اضافي زيادي دارد. مثلاً در دستور go to در كوبول وجود go لازم ولي نوشتن toاختياري است و فقط جهت افزايش خوانايي در دستور گنجانده شده است.

3-3- 6- تو ضيحات[3]

توضيحات موجود در هر برنامه مهمترين بخش مستند سازي آن برنامه به حساب مي آيند توضيحات در زبانها با شكلهاي متفاوتي ممكن است ظاهر شوند مثلاً درزبانC از /* توضيحات */ و در C++ از // استفاده مي شود.

---

[1] identifier

[2] noise-word

[3] remark-comment

3-3- 7- فضاي خالي[1]

در اغلب زبانها ،فضاي خالي به عنوان جدا كننده استفاده مي شود. در فرترن فضاي خالي معناي خاصي ندارد وفقط در رشتهها به عنوان blank استفاده مي شود در زبان Snobol 4 فضاي خالي هم به عنوان جداكننده عناصر يك دستور استفاده مي شود و هم به عنوان عملگر الحاق در رشتهها استفاده مي شود.

3-3- 8- جداكنندهها و محصور كنندهها[2]

يك عنصر نحوي است كه براي نشانه گذاري ابتدا يا انتهاي يك واحد نحوي مثل يك دستور يا عبارت به كار مي رود .محصور كنندهها ،جداكنندههاي جفتي هستند مثل جفت پرانتزها ويا begin ... end – جداكنندهها براي قابليت خوانايي يا سهولت در تحليل نحوي به كار مي روند اما اغلب براي از بين بردن ابهام وتعيين مرزها مورد استفاده قرار مي گيرند.

3-3- 9- فرمتهاي آزاد و طول ثابت

يك نحو در صورتي فرمت آزاد است كه دستورات در هرجايي از خط شروع شوند. نحو فرمت ثابت از موقعيتهاي خاصي از خط استفاده مي كند مثلا در اسنوبال 4 ، برچسبهاي دستورات توضيحات و.... كاراكتر ويژه اي در ابتداي خط مشخص مي شوند. در اسمبلي نيز از فرمت ثابت استفاده مي

شود كه هر عنصر يك دستور بايد در بخش خاصي از خط قرار گيرد ولي امروزه به ندرت از نحو فرمت ثابت استفاده مي شود.

3-3- 01- عبارات

توابعي هستند كه به اشياي داده موجود در برنامه دسترسي دارند و مقداري را برمي گردانند دستورات از عبارات ساخته مي شوند. در زبانهاي دستور ي مثل C عبارات عمليات اصلي براي تغيير حالت ماشين هستند در زبانهاي تابعي مانند ML وLisp عبارات كنترل ترتيب اجراي برنامه را مشخص مي كنند.

3-3- 11- دستورات

مهمترين جز نحوي در زبانهاي دستوري مي باشند دستورات اسنوبال 4 ، فقط يك نحو دارند واين زبان به نظم اهميت مي دهد. دستورات كوبول فرمتهاي مختلفي دارد و به خوانايي اهميت مي دهد. دستورات مي توانند ساخت يافته (تو در تو) يا ساده باشند دستور ساده هيچ دستور ديگري را شامل نمي شود. مثلاً APL ،4 Snobal از دستورات ساده استفاده مي كنند.

3-4- مراحل ترجمه

ترجمه يعني برنامه اي از يك زبان به يك زبان ديگر تبديل شود. ترجمه يك برنامه ممكن است بسيار ساده باشد مانند برنامههاي پرولوگ و ليسپ اما اغلب فرايند ترجمه بسيار پيچيده است فرايند ترجمه را به طور منطقي مي توان به دو مرحله تقسيم كرد :

---

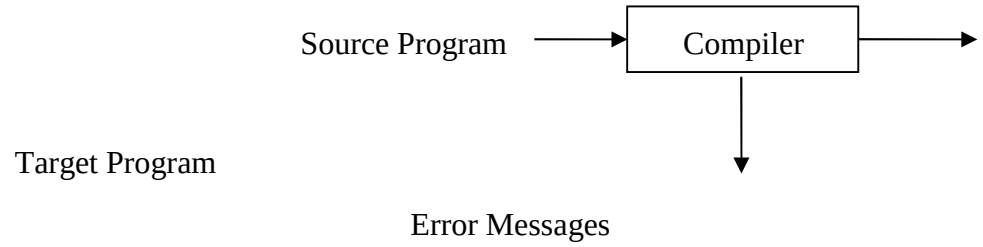
blank ۱ delimiters and brockets ۲

- تحليل[1] برنامه ورودي
- تركيب[2] برنامه مقصد

مرحله تحليل : شامل فازهاي تحليل لغوي ،تحليل نحوي ،تحليل معنايي و توليد كد مياني مي باشد و مرحله تركيب (توليد) شامل فازهاي بهينه سازي كد و توليد كد نهايي مي باشد.

كامپايلر: كامپايلر نرم افزاري است كه برنامه نوشته شده در يك زبان به نام زبان منبع[3] را خوانده و از روي گرامر آن ساختار برنامه را بدست آورده و آن را به برنامه معادل در زبان ديگر به نام زبان مقصد[4] ترجمه مي كند.

در صورتي كه برنامه نوشته شده به زبان مبدا با گرامر آن مطابقت نداشته باشد خطايي را صادر مي كند.



شكل 3 - 3

مراحل ترجمه (كامپايل) :

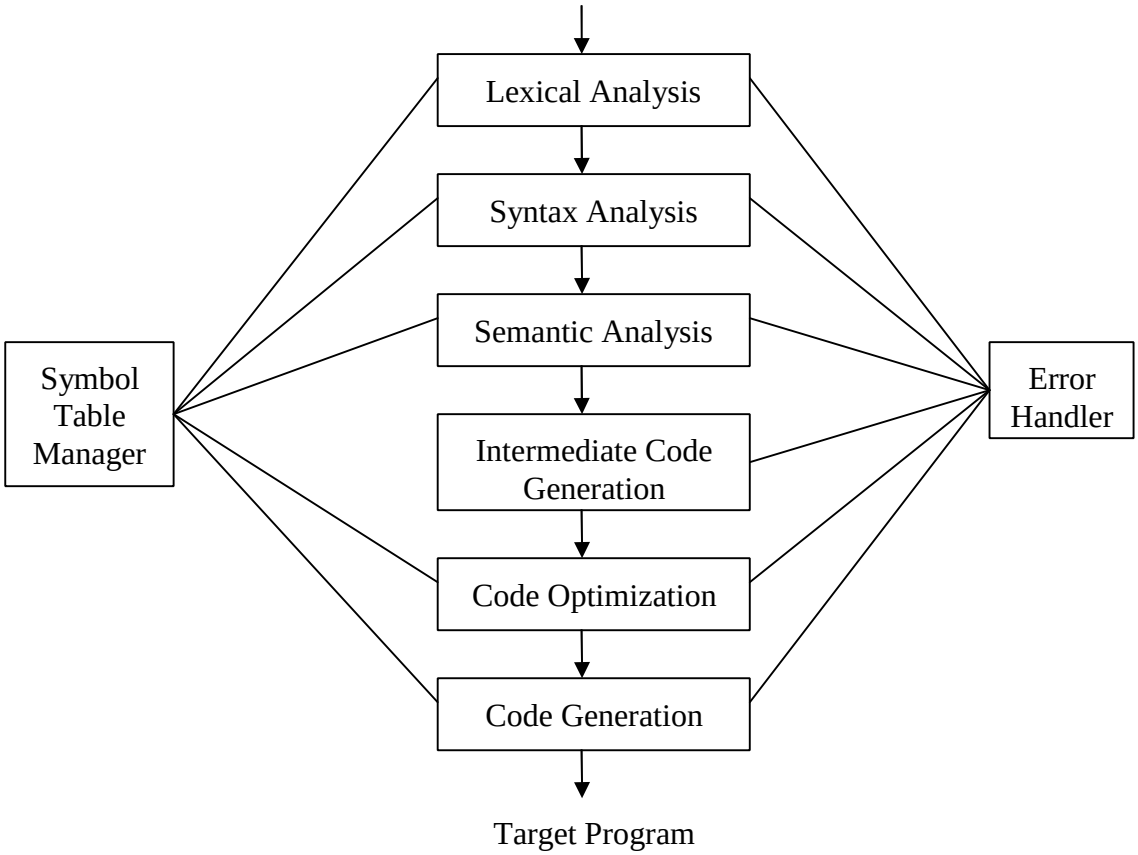
عمليات ترجمه در شش مرحله صورت مي گيرد :

- تحليل لغوي[5]
- تحليل نحوي[6]
- تحليل معنايي[7]
- توليد كد بينابيني[8]
- بهينه سازي كد[9]
- توليد كد نهايي[10]

---

analysis ١
synthesis ٢ Source
Language ٣
Target Language ٤
Lexical Analysis ٥
Syntax Analysis 6
Semantic Analysis 7
Intermediate Code Generation 8
Code Optimization 9
Code Generation ١٠

Source Program



شکل 3 - 4

ارتباط بين اين مراحل در شكل زير نشان داده شده است در كنار شش مرحله اصلي كامپا يلر، دو بخش ديگر به نام خطا پرداز و جدول علايم نيز وجود دارد.

3-4- 1- تحليل لغوي

در مرحله اول يعني تحليل لغوي ، برنامه ورودي كاراكتر به كاراكتر خوانده شده و به دنباله اي از نشانهها[1] تبديل مي گردد .انواع مختلف نشانهها عبارتند از : كلمات كليدي[2] ، عملگرها[3] ، جدا كنندهها[4] ، ثابتها[5] ،شناسهها[6] كه به اسامي توابع ، رويهها و به طور كلي اسامي كه كاربر انتخاب ميكند گفته ميشود .در اغلب زبانهاي برنامه سازي كلمات كليدي رزرو شده اند ، بدين معني كه كاربر مجاز نيست از هيچ يك از آنها به عنوان اسم يك متغيير ، تابع و يا رويه استفاده كند اما در برخي زبانها مثلPL/I اين محدوديت وجود ندارد. مدل اصلي كه براي طراحي تحليل

tokens
Keywords ١
٢
Operators 3
Delimeters ٤
Literals ٥
Identifiers ٦

گر لغوي استفاده مي شود ماشين خودكار متناهي[1] مي باشد با اينكه مفهوم تحليل لغوي ساده است ولي اين مرحله بسيار زمان بر است كه علت آن خواندن و بررسي تمام كاراكترهاي برنامه است.

4-3- 2- تحليل نحوي (تجزيه)

در اين مرحله ، ساختارهاي بزرگ مانند دستورات، اعلانها، عبارات و.... با استفاده ازعناصر لغوي كه توسط تحليل گر لغوي توليد شده اند شناسايي مي شوند. بنابراين در اين مرحله، برنامه با استفاده از زبان مبدا از نظر خطاهاي نحوي مورد بررسي قرار مي گيرند و با استفاده از نشانههاي توليد شده در مرحله تحليل لغوي يك درخت بارس[2] ايجاد مي گردد.

4-3- 3- تحليل معنايي

تحليل معنايي، مهم ترين مرحله ترجمه است در اين مرحله با استفاده از درخت توليد شده در مرحله قبلي ، برنامه ورودي از نظر خطاهاي مفهومي احتمالي مورد بررسي قرار مي گيرد اين مرحله پلي بين بخش تحليل و تركيب ترجمه است. در اين مرحله اعمال جنبي ديگري نظير نگهداري جدول نمادها ،كشف خطاها ، بسط ماكروها واجراي دستورات زمان ترجمه نيز انجام مي شود. يكي از مهم ترين كارها در تحليل معنايي،كنترل نوع مي باشد.

4-3- 4- توليد كد مياني

در اين مرحله يك برنامه كه معادل برنامه اصلي است به يك زبان مياني توليد مي شود. با ايجاد كدمياني ، عمليات بعدي كه كامپايلر بايد انجام دهد آسان مي گردد. بعضي كامپايلرها، نمايش مياني سريعي از برنامه مبدا دارند. اين نمايش مياني بايد داراي 2خاصيت زيرباشد :

- به آساني بتوان ان كد مياني را توليد وبهينه سازي كرد.

- ترجمه كد مياني به برنامه مقصد به راحتي صورت پذيرد.

نمونه اي از اين كدهاي مياني ،كد 3آدرسه[3] است كه شبيه به زبان اسمبلي مي باشد..

3-4- 5- بهينه سازي كد

در اين مرحله سعي مي شود تا كد مياني توليد شده در مرحله قبلي به نحوي بهبود داده شود اين كار سبب توليد كدي مي شود كه از لحاظ اجرايي سريع تر است و حافظه كمتري مصرف مي كند.

3-4- 6- توليد كد نهايي

در اين مرحله،هركدام از كدهاي مياني بهبود يافته به مجموعه اي از دستورات ماشين كه عملكرد مشابهي دارند تبديل مي شود. بنابر اين در آخرين فاز كامپايلرها يعني توليد كد ، به هريك از متغييرهاي موجود در برنامه حافظه تخصيص داده مي شود و متغيرها در ثباتها جايگزين مي شوند.

---

[1] FSA

[2] ParsTree

[3] Triple code

5-3- خطا پرداز[1]

هر فاز از كامپايلر بايد به گونه اي رفتار كند كه ادامه كامپايل و كشف خطاهاي بيشتري را ميسر سازد. كامپايلري كه با اولين خطا متوقف مي شود ناكار آمد است. فازهاي تحليل لغوي و معنايي بخش عمده اي از خطاهايي كه توسط كامپايلر كشف مي شوند را پيدا مي كنند. فاز لغوي مي تواند خطاهايي را شناسايي كند كه در آن رشته كاراكترها مطابق هيچ الگويي از توكنها نيست خطاهايي كه قوانين ساختاري (گرامري) دستور زبان را نقض مي كنند در فاز تحليل نحوي كشف مي شود.

6-3- جدول نمادها[2]

يكي از كارهاي مهم و اساسي يك كامپايلر، ثبت شناسههاي استفاده شده در برنامه ورودي(منبع) و جمع آوري اطلاعات درباره مشخصات هر شناسه است. اين مشخصات مي تواند شامل: آدرس حافظه اختصاص داده شده به شناسه ، نوع آن ، حوزه[3] يعني محلي از برنامه كه اين شناسه در آن تعريف شده است و در رابطه با رويهها ، اسم آنها ، تعداد و نوع آرماگونهاي آنها ، روشي كه طي آن آرماگونها به زير برنامه فرستاده مي شوند مثل : call by reference يا call by name و نوع نتيجه اي كه رويهها باز مي گردانند باشد.

در جدول نمادها به ازاي هر شناسه يك ركورد وجود دارد كه اين ركوردها شامل مشخصات شناسهها مي باشند اين جدول امكان دستيابي سريع به شناسهها و مشخصات آنها را فراهم مي كند. در مرحله تحليل لغوي ، تحليل گر لغوي كامپايلر ، شناسه اي را كه در برنامه مبدا پيدا مي كند آن را در جدول نمادها درج مي كند ولي خصيصههاي مربوط به شناسهها نمي توانند در هنگام تحليل لغوي وارد جدول نمادها شوند .بلكه در ديگر فازها ، اطلاعات مربوط به اين شناسهها به جدول اضافه خواهند شد و در مراحل مختلف توليد كد از آنها استفاده خواهد شد. به عنوان مثال در تحليل معنايي و توليد كد مياني لازم است نوع شناسه را بدانيم و يا در بخش توليد كل بايد جزئيات بيشتري از تخصيص فضا به هر شناسه را بدانيم.

7-3- يك مثال جامع براي فازهاي كامپايلر

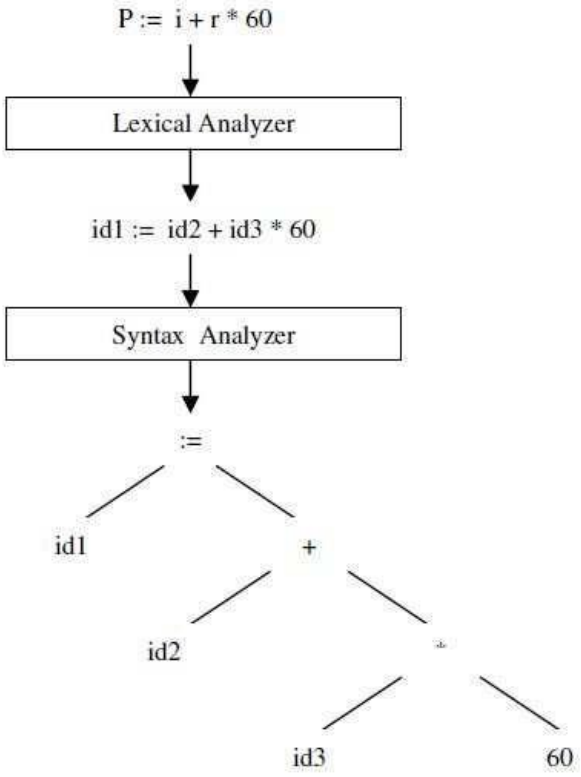
با يك مثال كل فازهاي كامپايلر را بررسي مي كنيم.

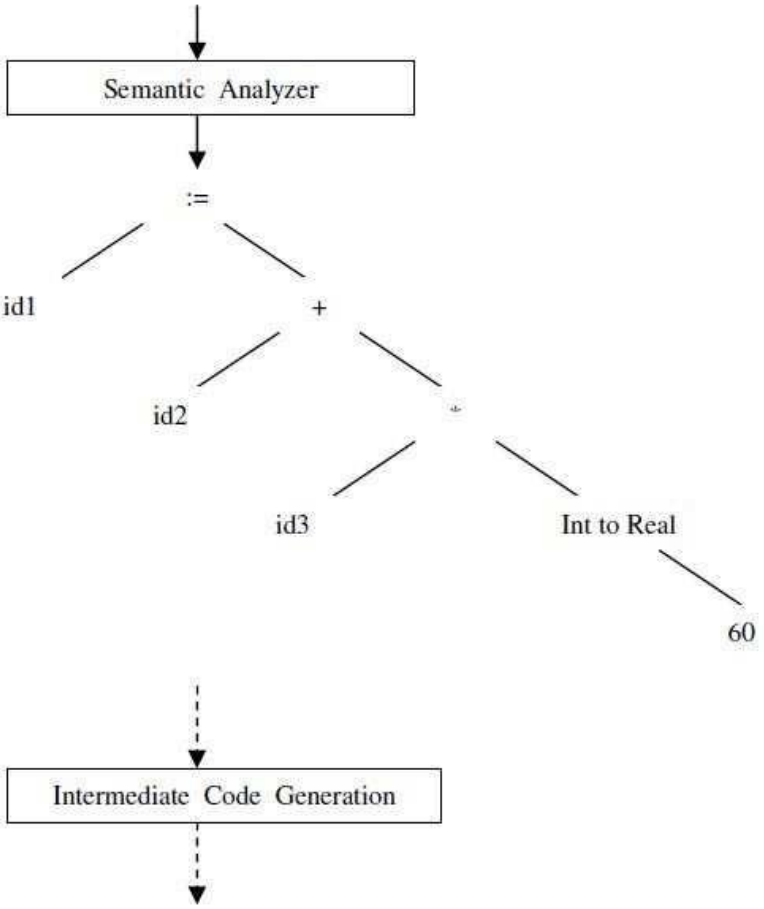
(فرض شده كه r، p، i همگي از نوع real هستند) `p:i+r×60`

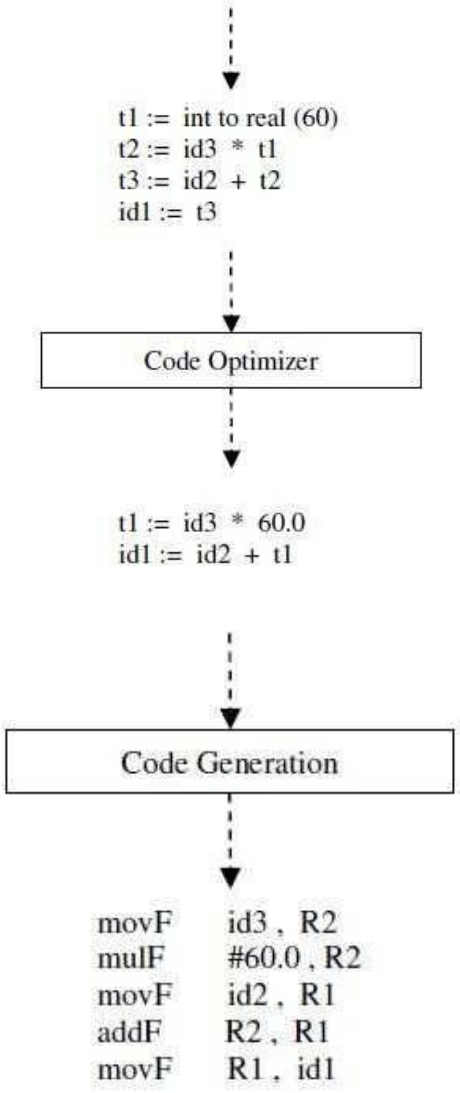
Error handler [1]

Symbol table [2]

Scope [3]




Semantic Analyzer
:=
id1
+
id2
id3
Int to Real
60
Intermediate Code Generation



شكل 3 - 5

3-8- ابتدا (front-end) و انتهاي (back-end) كامپايلرها

به چهارمرحله اول كامپايلر و بخشي از مرحله بهينه سازي كد كه به زبان مبدا وابسته اند و مستقل از زبان ماشين مقصد هستند ابتداي (front-end) كامپايلر گفته مي شود. به بخشي از مرحله بهينه سازي و مرحله آخر كامپايلر كه وابسته به ماشين مقصد هستند انتهاي (back-end) كامپايلر گفته مي شود اين بخش (انتهايي) از كامپايلر وابسته به زبان مبدا نيست.

3-9- مفهوم گذر[1]

به هر مرتبه خواندن برنامه ورودي (به شكل اوليه و يا مياني) از ابتدا تا انتها و توسط هر يك از قسمتهاي كامپايلر يك گذر گفته مي شود. كامپايلرها از نقطه نظر تعداد گذرهاي مورد نياز ، جهت انجام عمل ترجمه به دو دستهي

pass [1]

تك گذره[1] وچند گذره[2] تقسيم مي شوند. يك كامپايلر تك گذره ، كليه عمليات ترجمه ورودي را تنها با يك مرتبه خواندن ورودي انجام مي دهد. وجود مرحله بهينه سازي يا برخي از ويژگيهاي زبانهاي برنامه سازي نظير اين خاصيت كد تعريف رويهها بتوانند پس از فراخواني آنها قرار داده شود، سبب مي شود كامپايلر به گذرهاي بيشتري براي انجام عمليات خود نياز داشته باشد.

3-01- حساب لاندا

احتمالاً اولين مدل معنايي زبان برنامه نويسي ، حساب لاندا بوده است كه در دهه 1930 توسط كورچ به عنوان مدل تئوري محاسبات در مقايسه با ماشين تورينگ مطرح شد حساب لاندا مدل خوبي را براي فراخواني تابع زبان برنامه سازي ارائه كرد. در واقع الگول وليسپ مي توانند معناي فراخواني تابع را با مدل حساب لاندا رد يابي كنند.

عبارت لاندا به طور باز گشتي و به صورت زير تعريف مي شوند.

اگر X نام متغيير باشد ، X يك عبارت لاندا است.

اگر M يك عبارت لاندا باشد و $\lambda_{x\,m.}$يك عبارت لاندا است.

اگر F و A عبارت لاندا باشند ، (FA)عبارت لاندا است كه F يك عملگر و A يك عملوند است.

$(\lambda - \exp r\ \lambda - \exp r|)$

[1] single pass
[2] multi pass

11-3- سوالات فصل سوم

سوالات تستي فصل سوم

1-كدام گزينه غلط است؟(نيمسال دوم 38)

الف. ابهام مسئله اي است كه همراه نحو زبان وجود دارد ب .الگوريتمهايي مشخص جهت رفع ابهام زبان وجود دارد

ج .وقتي يك رشته در زبان داراي بيش از يك درخت تجريه باشد گرامر مبهم است.

د .عبارات منظم شكل ديگري را براي تعريف زبان ارائه ميكنند كه هم ارز گرامرهاي FSA و منظم ميباشد.

2-كدام گزينه غلط نيست؟(نيمسال دوم 38)

الف. كاربرد زبانهايي كه فاقد افزونگي هستند دشوار است.

ب .از ساختارهاي مبهم دو يا چند تفسير بعمل ميآيد

ج .ابهام يك مسئله مهم در طراحي هر زبان است.

د .هيچكدام

3 -كدام گزينه غلط است ؟(نيمسال دوم48)

الف. ساختار فرترن طوري است كه ترجمه زير برنامههاي مجزا ساده ميباشد

ب .در زبان SNOBOL تمايز نحوي بين دستورات برنامه اصلي و دستورات زير برنامه وجود ندارد.

ج .تعريف زير برنامههاي تودرتو در پاسكال امكان پذير نميباشد.

د .تحليل لغوي اولين مرحله ترجمه ميباشد.

4- در كدام يك از ماشينهاي پذيرنده زير حالت قطعي و غير قطعي يكسان نيستند؟(نيمسال اول 58 -68) الف. ماشين خودكار متناهي

ب.ماشين خودكار پشته اي ج .ماشين خودكار خطي

د.ماشين تورينگ

5- در كدام يك از مراحل ترجمه يك زبان از ماشين خودكار متناهي استفاده مي شود؟(نيمسال اول 58-68) الف.بهينه سازي

ب.تحليل معنايي ج.تحليل لغوي د.تحليل نحوي

6- كدام گزينه جزو معيارهاي نحو عمومي است؟(نيمسال دوم 58-68) الف. قابليت خواندن و نوشتن ب. سهولت بازرسي و ترجمه ج .موارد الف وب د. وجود ابهام

7- كدام گزينه جزو مراحل ترجمه يك برنامه مي باشد؟(نيمسال دوم 58-68)
الف.تحليل لغوي ب.تحليل نحوي ج.تحليل معنايي د. همه موارد

8- كدام گزينه غلط مي باشد؟(نيمسال دوم 58-68) الف.ابهام مسئله اي است كه همراه نحو وجود دارد.
ب.BNF توسط جان باكوس در اواخر دهه 1960 ايجاد شد.
ج.اگر گرامر مربوط به يك زبان مبهم باشد زبان مبهم است.
د.گرامرهاي منظم حالتهاي خاصي از گرامرهاي BNF مي باشد.

9- كدام گزينه غلط است؟(نيمسال دوم 58 -68)
الف.براي گرامرهاي منظم هميشه يك ماشين خودكار قطعي وجود دارد.
ب. PDAهاي قطعي هم ارز گرامرهاي LR(K) هستند.
ج.گرامرهاي منظم نمي توانند رشتههايي به شكل a^n توليد نمايند.
د.زبان نوع n توسط گرامر نوع n توليد مي گردد.

01- كدام مورد از معيارهاي نحو نيست؟(نيمسال اول 68 - 78) الف.قابليت خوانايي و قابليت نوشتن
ب.قابليت حمل ج.عدم وجود ابهام
د.سهولت ترجمه

11- وظيفه تحليلگر لغوي چيست؟(نيمسال اول 68 -78)
الف.شناسائي نشانهها ب.تعبير عملگرها
ج.پردازش ماكرو د.موارد الف و ب درست است.

21- كدام گزينه صحيح است؟(نيمسال دوم 68 - 78) الف.خروجي تحليل معنايي، تحويل توليد كد مي شود.
ب.خروجي تحليل معنايي، تحويل بهينه سازي مي شود.
ج.خروجي تحليل نحوي، تحويل لغوي مي شود.
د.خروجي تحليل معنايي، تحويل نحوي مي شود.

31- كدام گزينه غلط است؟(نيمسال دوم 68 -78)
الف.قابليت خوانايي و قابليت نوشتن در جهت عكس هم حركت مي كنند.
ب.ممكن است زباني باشد كه ترجمه آن آسان باشد ولي قابليت خوانايي و قابليت نوشتن آن پايين باشد.
ج.افزايش تعداد ساختارهاي نحوي ، كار ترجمه را ساده تر مي كند.
د)هيچكدام

14- اغلب مترجم زبان جديد، به همان زبان نوشته مي شود. از طريق كدام عمل زير مشكل ترجمه زبان جديد حل مي شود؟(نيمسال دوم 68-78)
الف.بافرينگ ب.پردازش دسته اي ج.خودراني د.خود استنادي

15- زباني خاص از كلمات اختياري در دستورات خود استفاده مي كند تا قابليت خوانايي را بهبود بخشد. مثلاً فرض كنيد زبان خاص در دستور go to اجازه دهدto بيايد يا نيايد وgo حتماً بيايد به اين كلمات اختياري اصطلاحاً چه مي گويند؟(نيمسال اول 78-88)

الف.خود استنادي ب.پارازيت ج.جداكننده د.پكيج

16- كدام گزينه زير صحيح است؟(نيمسال دوم 78-88)

الف. هر زباني كه قابليت خوانايي بالايي دارد حتما قابليت نوشتن بالايي نيز دارد.

ب .قابليت نوشتن بوسيله ساختارهاي طولاني مفيد به دست مي آيد.

ج .زبانهايي كه ساختارهاي نحوي اندكي را ارائه مي كنند برنامههائي با خوانائي پايين تر توليد مي كنند.

د .نحو موجود در زبانLISP ترجمه بسيار پيچيده اي نياز دارد.

17- زماني كه ترجمه مستقل در طراحي زبان مد نظر است بكار بردن اسامي مشترك موجب مي شود چندين زير برنامه يا واحدهاي ديگري از برنامه همنام باشند زبانهاي شي گرا چگونه اين مشكل را حل مي كند؟(نيمسال دوم 87-88)

الف: هر نام مشترك بايد منحصر به فرد باشد و برنامه نويس مسئول اين كار است.

ب: تنها از قواعد حوزه براي پنهان كردن اسامي استفاده مي شود ج: اسامي ممكن است در يك كتابخانه خارجي ذخيره شوند.

د: اسامي به صورت ثابت تعريف شوند.

18-كدام يك از موارد زير جزء مراحل ترجمه است ؟(تابستان 88) الف. توليد كد ب. تحليل نحوي ج .بهينه سازي د. همه موارد

19- زبانهايي كه ساختارنحوي اندكي ارائه مي كنند..... (نيمسال اول 88-98) الف.برنامههايي با خوانايي بالاتر توليد مي كند.

ب.برنامههايي با خوانايي پايين تر توليد مي كند.

ج.برنامههايي با قابليت نوشتن پايين تر توليد ميكند.

د.اين موضوع تاثيري در قابليت خواندن و نوشتن ندارد.

02- نگهداري جدول نمادها ، بسط ماكروها و اجراي دستورات زمان ترجمه در كدام مرحله از مراحل كامپايلر صورت مي گيرد؟(نيمسال اول 88 -98)

الف.تحليل معنايي ب.تحليل نحوي ج.تحليل لغوي د.بهينه سازي

12- كدام بخش از نحو زبان را نمي توان توسط گرامر BNF تعريف نمود؟(نيمسال اول 88 -98) الف.تعريف يك شناسه در بلوك خود ب.آنهايي كه وابسته به متن هستند.

ج.آنهايي كه قابل بهينه سازي نيستند. د.مواردي كه وابسته به متن نيستند.

22- منظور از راه اندازي خودكار چيست؟(نيمسال اول 88 -98) الف.ساخت كامپايلر زبان توسط همان زبان برنامه سازي ب.توليد كامپايلر زبان توسط مفسر

زبان ديگر ج.ساخت مفسر زبان بوسيله كامپايلر زبان
ديگر د.تبديل زبانهاي برنامه سازي مختلف به ديگر

32- در كداميك از ساختارهاي برنامه و زير برنامه، تمايز نحوي بين دستورات برنامه اصلي و دستورات زير برنامه وجود ندارد؟ (نيمسال دوم 88 -98)

الف.تعريف زير برنامهها بطور غير مجزا ب. توصيف دادهها جدا از دستورات اجرايي ج.تعريف دادهها بطور مجزا د. تعريف واسط مجزا

سوالات تشريحي فصل سوم

1 -معيارهاي عمومي نحو يك زبان را نام ببريد ؟ (نيمسا ل دوم 38)
2 - ساختار يك كامپايلر را با رسم شكل نمايش دهيد ؟(نيمسال دوم 38)

21-3- پاسخنامه سوالات تستي فصل سوم

سوال	الف	ب	ج	د
1		*		
2				*
3			*	
4		*		
5			*	
6			*	
7				*
8			*	
9			*	
10		*		
11	*			
12		*		
13			*	
14			*	
15		*		
16			*	
17		*		
18				*
19		*		
20	*			
21		*		
22	*			
23	*			

3-31- سوالات فصل چهارم

سوالات تستي فصل چهارم

1-كدام گزينه غلط ميباشد؟(نيمسال دوم48)

الف. اگر رشته اي در زبان وجود دارد كه داراي دو درخت تجزيه باشد گرامر مبهم ميباشد.

ب .PERL. توانايي پردازش عبارتهاي منظم را ندارد.

ج .هر تابع قابل محاسبه را ميتوان با ماشين تورينگ محاسبه نمود.

د .ماشينهاي تورينگ معادل گرامرهاي نوع صفر هستند.

2-- كداميك از مدلهاي زير براي تعريف رسمي معناي زبان استفاده نمي شود؟(نيمسال اول 58-68) الف.مدلهاي گرامري ب.مدلهاي اصل موضوعي

ج.مدل تابعي د.مدل شي گرا **3-** در كاهش عبارت لاندا ..... (نيمسال اول 58-68)

الف.كاهش خارجي ترين همانند فراخواني با نام است و كاهش داخلي ترين جمله معادل فراخواني با مقدار است.

ب.كاهش خارجي ترين همانند فراخواني با مقدار است و كاهش داخلي ترين جمله معادل فراخواني با نام است.

ج .هر دو نوع كاهش معادل فراخواني با نام است.

د .هر دو نوع كاهش معادل فراخواني با مقدار است.

4- .....تابعي است كه غير پايانه سمت چپ را به مقادير غير پايانههاي سمت راست بسط مي دهد. (نيمسال اول 85-86)

الف.صفت موروثي ب.صفت تركيبي

ج.گرامر صفت د.درخت انشقاق

5- عملياتي با امضاي stack->integer براي يك پشته تعريف شده است اين عمل معادل كداميك از اعمال پشته است؟(نيمسال اول 58 -68)

الف ب.Pop ج.Top د.Size Push.

6- در ساختار ماشين پذيرنده كدام گرامر، از نواري محدود به همراه هدي كه مي تواند در دو جهت حركت كند استفاده شده است. (نيمسال اول 58 -68)

الف.ماشين تورينگ ب.ماشين خودكار

ج.ماشين خودكار متناهي خطي د.ماشين خودكار پشته اي

7- كدام گزينه غلط مي باشد؟(نيمسال دوم 58-68)

الف. هر تابع قابل محاسبه را نمي توان با ماشين تورينگ محاسبه كرد.

ب .ماشينهاي تورينگ معادل گرامرهاي نوع صفر هستند.

ج .بعضي از مسئلهها غير قابل تصميم گيري اند الگوريتم عمومي براي حل آنها وجود ندارد.
د .هيچكدام.

8- در مورد حساب لاندا كدام گزينه غلط مي باشد؟(نيمسال دوم 58-68)

الف.در دهه 1920 توسط كورچ مطرح گرديد. ب.اولين مدل معناي زبان برنامه سازي است.
ج.مدل خوبي را براي فراخواني تابع زبان برنامه سازي ارائه كرد. د .هيچكدام

9- با توجه به تعريف ثوابت T (true) و F(false) در حساب لاندا تابع بولين and برابر كدام محاسبه λ زير است؟(نيمسال دوم 78-88)

الف λx. λy. ((xT)y) .ب λ x. λy.((xy)F).ج λ x. λy.((xT)y). د λx. λ y.((xy)T) .

01- در حساب لاندا تعريف زير كدام عملگر را تعريف مي كند؟(نيمسال اول 88-98)

λ M. λ N. λ a. λ b.((Ma)((Na)b))

الف. * ب./ ج.+ د - .

11- كداميك از موارد زير است كه ضرورتاً نمي تواند تضمين كننده صحت كامل برنامه در فرايند وارسي باشد؟(نيمسال اول 88 -98)

الف.تعيين مشخصات (S) زبان ب.بررسي پياده سازي
مشخصات برنامه ج.بررسي مشخصات (S) و برنامه د.آزمون
و تست برنامه

سوالات تشريحي فصل چهارم

1- روشهاي مختلف براي تعريف رسمي معناي زبان را بنويسيد؟(نيمسال دوم 58-68)

2- براي هر يك از توابع NOT,AND,OR يك تعريف با استفاده از حساب λ بنويسيد؟(نيمسال اول 88-98)

3-41- پاسخنامه سوالات تستي فصل چهارم

سوال	الف	ب	ج	د
1		*		
2				*
3	*			
4		*		
5				
6		*		
7	*			
8				*
9		*		
10			*	
11				*

فصل پنجم:

انواع داده اوليه

ا # ا :

شي داده انواع داده اسكالر انقياد شي داده متغييرها و ثوابت زير بازه نوع داده اعداد حقيقي مميز شناور مميز ثابت پياده سازي انواع داده اوليه اهداف اعلان نوع كاراكتري كنترل نوع پويا رشتهها كنترل نوع تبديل نوع ضمني فايلها انتساب و مقدار تشريحي يكي از اختلافات اساسي اطلاعاتي است كه يك زبان برنامه انجام دهد. طبعاً چگونگي و تنوع زبان جهت پوشش اطلاعات مي باشد.

در اين فصل چگونگي پياده سازي اطلاعات از

نوع صحيح

مشخصات انواع داده اوليه اعداد حقيقي نوع شمارشي اعلان نوع بولين كنترل نوع انواع داده مركب ايستا اشاره گرها تبديل نوع و دهي اوليه سوالات تستي و زبانهاي برنامه سازي ، انواع سازي مي تواند روي آنهاعمليات عمليات بر مبناي قابليتهاي يك

نوع اوليه مورد توجه قرار مي گيرند.

ـداده5-1- شيء (D.O)[1]ـ

يك شيء داده گروهي از يك يا چند قسمت از اطلاعات است كه در كامپيوترهاي مجازي استفاده مي شود.در واقع شيء داده ظرفي[2]براي مقاديري از دادههاست.يعني محلي است كه دادهها در آنجا ذخيره و بازيابي مي شوند. يك شيء داده توسط مجموعه اي از صفات مشخص مي شود كه مهمترين آنها نوع داده است.

اشياء داده به دو دسته تقسيم مي شوند:

- **تعريف شده توسط برنامه نويس:**

اشياء دادههايي هستند كه توسط برنامه نويس تعريف مي شوند مانند متغيرها ،مقادير ثابت ، آرايه، فايل،...

- **تعريف شده توسط سيستم:**

اشياء دادههايي هستند كه توسط سيستم به وجود مي آيند و مستقيماً در اختيار برنامه نويس نيستند. مثل پشتههاي زمان اجرا، ركوردهاي فعاليت زير برنامهها ،بافرهاي فايل و ليست فضاي آزاد،جدول نمادها.

طول عمر[3]:

يكي ديگر از صفات شيء داده طول عمر است.فاصله زماني بين لحظه اي كه حافظه به شي داده تخصيص داده مي شود تا زماني كه حافظه از آن پس گرفته مي شود را طول عمر شي داده گويند. هر يك از اشياء داده داراي طول عمر مخصوص به خود هستند. بعضي از اشياء داده در شروع اجراي برنامه وجود دارند و برخي در حين اجرا بصورت پويا ايجاد مي شوند و برخي در حين اجراي برنامه از بين مي روند و برخي تا آخر اجراي برنامه باقي مي مانند.

ساختار شيء داده :

يك شيء داده توسط مجمو عه اي از صفات مشخص مي شود مثل نوع داده و نام كه معمولاً در طول عمر آن عوض نمي شود.يك شيء داده داراي محلي براي مقدار داده است. مقدار يك شيء داده ممكن است عدد، كاراكتر ،يا اشاره گر باشد.

- **شيء داده اوليه:** يك شيء داده اوليه است اگر تنها شامل يك محل حافظه براي مقدار داده باشد. مانند

اشيا داده از نوع ....، char، float,int

- **شيء داده ساختاري**: اگر شيء داده شامل مجموعه اي از ساير اشياء داده اي باشد.مانند ركورد ، ،آرايه مقدار يك شيء داده اي توسط الگويي از بيتها مشخص مي شود.به مثال زير دقت كنيد:

---

[1] Data Object

[2] Container

[3] Life time

A: | | A: 1001 | 0000000000010001 |

الف) **متغير مقيد :** شيء داده به مقدار داده 71 مقيد ميشود .

ب) **مقدار داده :** الگوي بيتي است كه هروقت عدد 71 در برنامه استفاده شود ، مترجم از آن استفاده ميكند .

ج) **شيء داده :** محلي در كامپيوتر به نام A

شكل 5 - 1

5-1- 1- انواع مختلف انقياد يك شيء داده

يك شيء داده در طول عمر خود مي تواند انقيادهاي مختلفي را بپذيرد كه مهمترين آنها عبارتند از:

- **انقياد يك شيء داده به يك نوع:** اين انقياد در زمان ترجمه برنامه انجام مي گيرد و مجمو عه مقاديري كه شيء داده اي مي تواند بپذيرد به آن نسبت داده مي شود.
- **انقياد شيء داده به محلي از حافظه:** محلي در حافظه براي شيء در نظر گرفته مي شود.اين كار

(انقياد)از ديد برنامه نويس مخفي بوده و توسط روالهاي مديريت حافظه كامپيوترمجازي انجام مي شود.

- **انقياد شيء داده به يك يا چند ارزش(مقدار):** اين نوع انقياد توسط دستورات انتساب مثلA:=13 انجام مي گيرد.
- **انقياد يك شيء داده به يك يا چند نام:** اين انقياد توسط اعلان[1]ها انجام پذيرفته و هنگام فراخواني زير برنامه و برگشت از آن اصلاح مي شود.
- **انقياد شي داده به يك يا چند شيء داده ديگر:** اين انقياد در هنگام استفاده از متغيرهاي نوع اشاره گر، انجام مي گيرد.

5-1- 2- متغيرها و ثوابت

868686**متغير**F86 **:** يك شيء داده اي است كه توسط برنامه نويس تعريف شده و به صورت صريح استفاده مي شود.

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

878787**ثابت**F87 **:**يك شيء داده اي با نام است و مقداري كه به آن نسبت داده مي شود در طول عمر آن ثابت است.

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

[1] Declation
[2] Variable
[3] Constant

ثابت ليترال[1][F]: يك ثابت ليترال، ثابتي است كه نام آن همان نمايش مقدارش است.مثلا "37"يك ثابت ليترالاست كه يك شيء داده با مقدار 73 مي باشد.

نكته: چون مقدار ثابت به طور دائم در طول عمرش به آن مقيد مي شود، بنا براين اين انقياد توسط مترجم شناخته شده است و كامپايلر مي تواند از اطلاعات مربوط به مقادير ثابت براي كاهش توليد كد استفاده كند.

گاهي اوقات كامپايلر ميتواند ازاطلاعات مربوط به مقادير ثابت به منظور اجتناب از توليد كد براي يك كلمه يا عبارت استفاده نمايد.

مثال :

```
If ( Max < 2 ) then
  …
Else
  …
```

دراين حالت مترجم از قبل مقادير دادهها را براي ثابتهاي Max و 2 دارد و ميتواند محاسبه نمايد كه شرط فوق درست است يا خير. به طور كلي ازهر كد نوشته شده داخل دستور If در يكي از قسمتهاي آن خودداري كند.

مثال1:

```
F ( ) {
int N; N
= 27;
  …
}
```

- اين اعلان يك شيء داده اوليه از نوع صحيح را مشخص مي كند.(type)

- اين شيء داده هنگام ورود به زير برنامه ايجاد مي شود و هنگام خروج از ان از بين مي رود يعني طول عمر آن برابر زمان اجراي زير برنامه ا ست.(life time)
- در اثناي طول عمر اين شيء داده به نام "N" مقيد مي شود كه از اين طريق به آن مراجعه مي شود.اگر شيء داده به عنوان پارامتر به زير برنامه ارسال شود نام ديگري به آن مقيد مي شود.(name)
- مقدار اوليه به شيء داده مقيد نمي شود.اما دستور انتساب مقدار 72 را به آن مقيد مي كند.(value) **مثال2:**

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

---

[1] Litteral

- N متغيري ساده است.max(توسط كاربر) 30,27(ليترال)ثابت هستند.
- : "30","27",max, N نام اشياء
- تفاوت بين مقدار"27" و 72 :27 مقداري صحيح است كه بصورت دنباله اي از بيتها در حافظه نمايش داده مي شود و نام "72"دنبا له اي از دو كاراكتر "2"و"7" است كه بصورت دهدهي نمايش داده شده است.
- ثابت 03 دو نام دارد: نام تعريف شده توسط برنامه نويس و نام ليترال كه هر دو به يك محل از حافظه اشاره دارد.
- #define max 30 يك دستور است كه موجب مي شود مترجم max را برابر 03 قرار دهد.در حاليكه صفت const در زبان C راهنماي مترجم است و مي گويد متغير max هميشه داراي مقدار 03 است .

ماندگاري داده:

در اكثر موارد طول عمر متغيرها با زمان اجراي برنامه يكي است ،اجرا كه تمام شد متغيرها هم از بين مي روند اما اگر طول عمر يك داده بيشتر از يك اجرا باشد لذا گوئيم آن داده ماندگار است و در بين اجراهاي مختلف برنامه وجود دارد.

ماندگاري: از بين نرفتن انقياد مكان و مقدار بعد از اتمام برنامه.

2-5- نوع داده[1]

نوع داده طبقه اي از اشياء داده به همراه مجموعه اي از عمليات براي توليد و دستكاري مي باشد. هر زبان مجموعه اي از انواع داده اوليه مانند char,int,bool,float دارد كه هنگام تعريف زبان مشخص شده اند .علاوه بر اين ،زبان ممكن است به برنامه نويس اجازه دهد انواع جديدي را تعريف كند. در زبانهاي جديدي مثل جاوا و Ada كاربران مي توانند انواع دادههاي جديدي براي خود تعريف كنند ولي چنين كاري در فرترن و كوبول امكان پذير نيست. نوع داده معمولاً در دو سطح مختلف بررسي مي شود:

- 90909090**مشخصات[2]F:** تعريف خصوصيات و مشخصات.
- **پياده سازي** 91919191**كردن[3]F:** پياده سازي آن نوع داده و عمليات روي آن.

2-5- 1- در سطح مشخصات چند عنصر اساسي مي تواند مطرح شود

- 92929292**صفات[4]F :** و يژگي است كه اشياء داده از يك نوع را با ديگر نوعها متمايز مي كند.صفات اصلي هر شيء داده مانند نوع داده و نام معمولاً در طول عمر آن عوض نمي شود. بعضي صفات ممكن است در توصيف

گر[5] به عنوان بخشي از شيء داده در حين اجراي برنامه، ذخيره شوند.توجه داشته باشيد كه مقدار يك صفت از شيء داده با مقداري كه شيء داده حاوي آن است متفاوت باشد. چون مقدار موجود در شيء دادهممكن است در طول عمر آن تغيير كند و هميشه به طور صحيح در طول اجراي برنامه نمايش داده مي شود ولي مقدار صفت شيء داده اينگونه نيست.

- 94949494**مقادير[1]F:** مجموعه اي از مقادير ممكن كه يك شيء داده مي تواند داشته باشد كه تا حد زيادي به سخت افزار وابستگي دارد. مجموعه مقادير تعريف شده توسط نوع داده اوليه

[1] Data type
[2] Specification
[3] Implementation
[4] Attribute
[5] Descriptor

را معمولاً مجموعه مرتب مي گويند زيرا داراي كمترين و بيشترين مقدار است و براي هر دو مقدار يكي كوچكتر و ديگري بزرگتر است.

- **عمليات$_{F_2}$95959595:** مجموعه اي از عمليات كه براي يك نوع داده تعريف شده است

،تعيين مي كند كه اشياء داده از آن نوع چگونه بايد دستكاري شوند. اين عمليات ممكن است عمليات اوليه يا عملياتي باشند كه توسط برنامه نويس تعريف مي شوند. عمليات اوليه به عنوان بخشي از زبان تعريف مي شوند و از ديد برنامه نويس به دو دسته عمليات يكاني و دودويي تقسيم مي شوند. عمليات تعريف شده توسط برنامه نويس به شكل زير برنامهها نوشته مي شوند مثل زير برنامه اي كه قدر مطلق يك شيء داده از نوع صحيح را محاسبه مي كند.

براي مثال در سطح مشخصات عناصر اساسي براي نوع داده آرايه عبارت است از:

- **صفات:** تعداد ابعاد،بازه،انديس هر بعد،نوع داده هر عنصر
- **مقادير:** مجموعه اي از اعداد است كه در عناصر ارايه مي تواند ذخيره شود.
- **عمليات:** استفاده از انديس براي مراجعه به يك عنصر ،ايجاد آرايه،بدست اوردن حد پائين و بالا،انجام محاسبات روي آرايه.

5-2- 2- عناصر اساسي جهت پياده سازي عبارتند از

5-2- 2-1- پياده سازي عمليات: (نحوه پياده سازي عمليات)

سه روش براي پياده سازي عمليات روي اشيا داده وجود دارد:

- **به صورت سخت افزاري:** به عنوان مثال اگر مقادير صحيح به كمك نمايش سخت افزاري ذخيره شوند آنگاه ميتوان جمع و تفريق را با استفاده از عمليات سخت افزاري پياده سازي كرد.
- **به صورت زير برنامه يا تابع:** مثلاً عمل جذر گيري كه توسط سخت افزار به طور مستقيم پشتيباني نميشود .براي پياده سازي عمليات يك زير برنامه مثلاً SQRT نوشته ميشود .

- **به صورت دستوراتي كه داخل برنامه نوشته ميشوند:** اين روش نيز مانند روش قبلي نرم

افزاري است اما به جاي زير برنامه، دستورات مربوطه در خود برنامه نوشته ميشوند مثل عمل قدر مطلق گيري:

---

[1] Value

[2] Operation

```
Abs ( x ) = if x < 0 then –x else x
```

2-5- 2-2- نمايش حافظه: (چگونگي ذخيره اطلاعات و نمايش حافظه)

نمايش حافظه براي انواع داده اوليه، تحت تاثير كامپيوتري است كه برنامه را اجرا ميكند به عنوان مثال نمايش عدد صحيح به صورت دنباله بيتي است جهت نمايش كاراكترها ميتوان از كدهاي كاراكتري موجود در سخت افزار يا سيستم عامل بهره برد. اگر از نمايشهاي سخت افزاري استفاده شود آن گاه عمليات روي آن نوع ميتواند با استفاده از عملياتي پياده سازي شود كه توسط سخت افزار ارائه شده است و گرنه بايد بطور نرم افزار شبيه سازي شود. صفت يك شي ممكن است در زمان اجرا در يك توصيف گر به عنوان بخشي از يك شي داده ذخيره شود. اين كار در زبانهايي مانند ليسپ و پرلوگ براي قابليت انعطاف وجود دارد(مثلاً اعداد int توسط سخت افزار پشتيباني ميشود - شبيه سازي اعداد 42 رقمي در صورتي كه سخت افزار حداكثر 21 رقمي را پشتيباني كند) **تعريف عمليات:**

هر عمليات معمولاً به صورت يك تابع رياضي بيان مي شود بطوريكه يك يا چند پارامتر(عملوند)را بعنوان ورودي پذيرفته و نتايجي را توليد مي كند.مجموعه اي از مقادير كه عمليات بر روي آنها تعريف شده است دامنه عمليات و مجموعه اي از نتايج ممكن برد عمليات نام دارد.الگوريتم موجود در بدنه عمليات مشخص مي كند بر روي پارامتر -هاي داده اي چه محاسباتي انجام شود تا نتايج مطلوب بدست آيد يعني الگوريتم،عملكرد عمليات را مشخص مي كند.

امضاي عمليات ساده:

براي مشخص كردن امضاي عمليات از نشانه گذاريهاي رياضي كه در زبان C آن راالگو[1]مي گوئيم استفاده مي كنيم .

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

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

=: integer = integer ⟶ boolean

⟶ real sqrt : real

گاهي اوقات تعيين مشخصات دقيق يك عمليات به صورت تابع رياضي دشوار است، چهار عامل موجب ميشود تعريف عمليات زبانهاي برنامه سازي به صورت تابع رياضي پيچيده شود.

1- عملياتي كه به ازاي ورودي مشخصي (بعضي از مقادير دامنه) تعريف شده

نيستند[2]F: 97979797

عملي كه بر روي دامنه خاصي تعريف شده (درزبان برنامه سازي) ممكن است براي بعضي از ورودیهاي روي آن دامنه، تعريف نشده باشد مانند مجموعه اي از اعداد كه در عمليات محاسباتي سرريز[3] يا زيرريز[4] توليد ميكنند .

---

prototype

[1] Undefined for Certain Input

[2] over flow [3] under flow [4]

2- آرگومان ضمني[1]F: 1 1 1 1

ورودیهاي ضمني يا ورودیهايي كه به صورت صريح تعريف نشده اند مثل متغيرهاي سراسري كه باعث ميشوند تعيين دقيق دامنه عمليات بر روي اشيا داده ممكن نباشد .

3- اثرات جانبي[2] F: 0 0 0 0

يك عمليات ممكن است علاوه بر وظيفه اصلي خود، اعمال مخرب ديگري نيز انجام دهد.مثل عملياتي كه حاصل جمع دو عدد را برميگرداند ولي مقادير ذخيره شده در ساير اشيا داده را نيز اصلاح ميكند يا يك تابع ممكن است علاوه بر مقدار برگشتي ، آرگومانهاي ورودي خود را نيز تغيير دهد كه اين كار نيز نوعي اثر جانبي است.

4- خود اصلاحي[3]F: 102102102102

عمليات ميتواند ساختار داخلي ، از جمله دادههاي محلي كه در بين اجراهاي مختلف نگهداري ميشوند يا حتي كد خود را اصلاح كند بنابراين نتايج حاصل از عمليات براي مجموعه خاصي از آرگومانها ، نه تنها به آن آرگومانها ، بلكه به سابقه فراخوانيهاي قبلي در اثناي محاسبات و آرگومانهايي كه در هر فراخواني ارسال ميشود بستگي دارد به عنوان مثال عمليات مولد عدد تصادفي يك آرگومان ثابت را ميگيرد و در هر بار اجرا مقدار متفاوتي را بر مي-گرداند. اين عمل علاوه بر اينكه نتيجه اش را بر

ميگرداند عدد دانه[4] را هم تغيير مي دهد. بنابراين در نتيجه بعدي تأثير ميگذارد. لذا نتايج حاصل از عمليات علاوه بر آرگومان ورودي، به سابقه فراخوانيهاي قبلي هم بستگي دارد.

خود اصلاحي از طريق تغيير در كد متداول نيست ولي در زبانهاي مثل ليسپ صورت ميگيرد .

زير 104نوعها[5]104F104 :

وقتي نوع داده جديدي را توصيف ميكنيم اغلب تمايل داريم بگوييم كه اين نوع مشابه نوع ديگري است به عنوان مثال درزبان C انواع short,long, int شكلهاي گوناگوني از نوع داده صحيح هستند و رفتار آنها يكسان است و علاقه داريم كه عملياتي مانند جمع وضرب به طور يكسان تعريف شود. بنابراين اگر نوعي به عنوان بخشي از نوع بزرگتر باشد آن نوع را زيرنوع و به نوع بزرگتر ابرنوع[6] ميگوييم. يا به عبارت دقيق تر اگر مجموعه مقادير يك نوع ، زيرمجموعه ، مجموعه مقادير نوع ديگر باشد نوع با مجموعه مقادير بزرگتر را ابر نوع و نوع ديگر را زيرنوع گويند.به عنوان مثال short int زير نوع int و int زيرنوع long int مي باشد.

5-3- اعلان[7]

اعلان دستوري از برنامه است كه نام ، نوع و طول عمر اشيا داده را مشخص ميسازد. اعلانها بر دو نوع هستند

- **صريح:** در اين روش نوع و نام شي داده اي دقيقاً توسط برنامه نويس تعيين ميگردد.

---

[1] Implict argument
[2] Side effect
[3] Sub type [4] Self modification seed
[5]
[6] Super type
[7] Declaration

- **ضمني:** خود برنامه مترجم (كامپايلر) يا كامپيوتر پيش فرضهايي را در مورد خصوصيات اشيا ارائه ميكند .مثلاً در زبان فرترن ، متغيرها از Nو .....وJ و I به طور پيش فرض از نوع صحيح هستند مثل INDEX يا NEW البته اعلان صريح نيز در زبان فرترن وجود دارد. در زبان perl انتساب مقداري به متغير آن را اعلان ميكند .

```
$abc='test';
```
متغير رشته اي

```
$abc=5;
```
متغير صحيح

گاهي اوقات جزئيات پياده سازي نيز هنگام اعلان مشخص ميشود .مثلاً در زبان كوبول اعلان يك متغير صحيح به صورت computational باعث ميشود از نمايش حافظه دودويي به جاي نمايش كاراكتري استفاده گردد.

علاوه بر اين، اعلانها ميتوانند اطلاعاتي راجع به عمليات را به مترجم بدهند مثل اعلان تابع زير كه تعداد، ترتيب ، نوع پارامتر و نتيجه را مشخص ميكند.

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

اطلاعاتي كه توسط دستورات اعلان برآورده ميشود: -1 نام شي داده 2- نوع شي داده 3 - مقدار اوليه شي داده4 - صفات شي داده5 - طول عمر

5-3- 1- اهداف اعلان

الف- انتخاب نمايش 107107107**حافظه**[1] F **:** اگر اعلان اطلاعاتي راجع به نوع و صفات شي داده در اختيار كامپايلر قرار دهد بهترين نمايش حافظه براي آن انتخاب ميشود .

ب- مديريت 108108108**حافظه**[2] F108 **بهتر:** اطلاعاتي كه توسط اعلان كردن در رابطه با طول عمر اشيا داده اي فراهم مي -شود باعث مديريت بهتر حافظه ميشود. مثلاً متغيرهايي كه در اول يك زيربرنامه اعلان ميشوند طول عمر يكساني دارند و ميتوان براي تمام آنها يك بلوك حافظه را اختصاص داد و پس از اتمام زير برنامه، آن بلوك حافظه را پس گرفت. يا اگر متغيرهاي پويايي

[1]
[2]
[3]

وجود داشته باشد كه توسط دستورات تخصيص حافظه مثل عملگر new در C++ وmalloc درCايجاد شوند چون طول عمر اين اشيا داده اعلان نميشود در بلوك ديگري از حافظه قرار ميگيرند(چون طول عمر آنها متفاوت است در heap نگهداري مي شوند)

ج-مشخص شدن وضعيت عمليات ^{109109109}**چندريختي**$_{3}^{F109}$ **:** بسياري از زبانها ، نمادهاي خاصي مانند+ را براي تعيين عمليات مختلف استفاده ميكنند مثلاً نماد + مي تواند مبين عمليات جمع دو عدد صحيح، جمع دو عدد

Storage representation
Storage Managment
Generic operation

اعشاري، الحاق دو رشته يا اجتماع دو مجموعه باشد كه با توجه به نوع آرگومانها عمليات مورد نظر تعيين خواهد شد .

در Ada ميتوان زيربرنامههاي همنام تعريف كرد. ML اين مفهوم را با چند ريختي كامل بسط داد كه در آن تابع برحسب ترتيب، تعداد ، نوع آرگومانها ميتواند پياده سازيهاي مختلف داشته باشد. هدف اصلي اعلانها در چند ريختي اين است كه اعلانها موجب ميشود تا مترجم در زمان كامپايل كردن ، عملياتي را كه به وسيله نماد مجدداً تعريف شده مشخص ميشود را تعيين كند.

مثلاً در زبان C، كامپايلر با اعلان دو متغير A,B متوجه ميشود كه چه عملي در A+B انجام شود (جمع صحيح يا اعشاري) بنابراين در زمان اجرا لازم نيست كنترل شود كه چه عملياتي بايد صورت گيرد. از طرفي در اسمالتاك چون اعلان نوع متغير وجود ندارد در زمان اجرا بايد تعيين شود + چه نوع عملي است.

د- كنترل ^{0000}**نوع**$_{1}^{F}$ **:** مهم ترين هدف اعلان از ديدگاه برنامه نويس ، انجام كنترل نوع ايستا به جاي كنترل نوع پويا ميباشد.

(Generic Operation) : عمليات چندريختي – چندشكلي

۱
۲
۳

دريك زبان برنامه سازي سنبل خاصي مثل + ميتواند جهت جمع 2 عدد integer ، real ، الحاق دو رشته ، جمع دو عدد مختلط ويا 2 مجموعه به كار گرفته شود كه بسته به نوع عملوندهاي آن زيربرنامههاي خاصي براي اجراي آن بايد فراهم شود (كه اعلانهاي نوع اطلاعات مفيدي را دراين جهت براي ما فراهم ميكند).

مثال :

```
A ( int x )
A ( int x , int y) A (
int x , floating )
```

معمولا به چنين عملگري كه چندين عمليات را تحت پوشش قرار ميدهد عملگر پربار شده ميگويند (Over Lapping) مثل عملگر جمع.
در زبان ML اين مفهوم را با استفاده از چندريختي كامل گسترش ميدهد يعني در اين زبان يك نام تابع با پياده سازيهاي متفاوت ارائه ميشود كه با توجه به انواع ، تعداد و ترتيب ورورديها و نتايج يكي از پياده سازيها انجام ميشود .

تعريف كلي چندريختي:

تعريف چندين زيربرنامه با نام يكسان و پياده سازيهاي متفاوت كه براي يك عمل خاص نوشته ميشود را چندريختي ميگويند و اينكه در زمان اجرا كدام يك از اين توابع بايد به كار گرفته شوند (كدام پياده سازي) اين

---

[1] Type checking

كار توسط تعداد ، ترتيب و نوع پارامترهاي آن انجام ميشود در عمليات چندريختي كل عمليات ثابت است فقط پياده سازي آن براي ورورديهاي مختلف متفاوت خواهد بود.

درزبان Ada به برنامه نويس اين امكان داده ميشود كه نامهاي زيربرنامه پربارشده را تعريف نمايد و به نمادهاي عملگرهاي موجود معاني اضافي ديگري را ضميمه كند.

كنترل نوع و تبديل نوع :

نمايش حافظه اي كه در سخت افزار براي دادهها ساخته ميشود اطلاعاتي راجع به نوع ندارد مثلاً : 11010101 اين دنباله بيتي ميتواند مبين اطلاعات از هرنوع صحيح ، اعشاري و... باشد بنابراين كنترل نوع در سطح سخت افزار وجود ندارد و كامپيوترهاي معمولي در سطح سخت افزار قادر به

تشخيص خطاي نوع نيستند. امتياز اصلي استفاده از زبان سطح بالا اين است كه زبان ميتواند كنترل نوع را براي تمام عمليات پياده سازي كند و درمقابل خطاها محفوظ باشد ولي در زبانهاي سطح پاييني مانند اسمبلي، عمليات انجام ميگيرد ولي نتيجه اش بي معني است چون كنترل نوع در زبان سطح پايين اسمبلي وجود ندارد.

4-5- كنترل نوع[1]

منظور از كنترل نوع اين است كه هرعملياتي كه در برنامه انجام ميگيرد تعداد و نوع آرگومانهاي آن درست باشد.

دو روش براي كنترل نوع وجود دارد:

- **كنترل نوع پويا(D.T.C[2] $^{F}(^{2222}$):** كنترل نوع در زمان اجرا صورت مي گيرد.
- **كنترل نوع ايستا(S.T.C[3] $^{F}(^{3333}$):** عمل كنترل نوع در زمان ترجمه(كامپايل) صورت مي گيرد.

4-5- 1- كنترل نوع پويا

كنترل نوع پويادر زمان اجرا انجام ميشود و بلافاصله قبل از اجرا عمل خاصي صورت ميگيرد در كنترل نوع پويا در هر شيء داده يك برچسب نوع قرار ميگيرد كه نوع آن شي ء داده رامشخص ميكند .در برخي از زبانهاي برنامه سازي مانند لسيپ و پرولوگ كنترل نوع به صورت پويا است در اين زبانها متغيرها اعلان نميشوند ونوع متغيرها در حين اجراي برنامه ميتواند تغيير كند بنابراين حتماً بايد كنترل نوع، پويا باشد چون نوع متغيرها درحين اجرا تغيير ميكند .در زبانهاي بدون اعلان ، متغيرها را بدون نوع گويند چون نوع ثابتي ندارند.

مزاياي كنترل نوع پويا:

- **انعطاف پذيري در برنامه نويسي:** به علت عدم نياز به تعريف اعلان نوع داده اي ، برنامه نويس از بسياري از محدوديتها آزاد است.
- انجام گرفتن عمل تبديل نوع در زمان اجرا

۱
۲
۳

Type checking
Dynamic Type Checking(D.T.C)
Static Type Checking(S.T.C)

- تعريف اعلان براي هرنوع داده ميتواند مورد نياز نباشد.

معايب كنترل نوع پويا :

- اشكال زدايي[1] و يافتن تمام خطاهاي نوع آرگومان سخت است چون كنترل نوع پويا انواع داده را در زمان اجراي عمليات كنترل ميكند لذا عملياتي كه اجرا نشوند كنترل نخواهند شد و تمام مسيرهاي اجرايي ممكن را نميتوان تست كرد.
- در كنترل نوع پويا ميبايست اطلاعات مربوط به نوع در زمان اجراي برنامه را نگهداري كرد لذا حافظه بيشتري مصرف ميشود.
- كنترل نوع پويا ميبايست به صورت نرم افزاري پياده سازي شود و سخت افزار به ندرت از آن پشتيباني ميكند از آنجا كه كنترل نوع قبل از اجراي هر عملي بايد صورت پذيرد لذا سرعت اجراي برنامه در كنترل نوع پويا كاهش مييابد.

به دليل معايب فوق اغلب زبانها سعي ميكنند كنترل نوع پويا را كم كرده و بيشتر كنترلها را به صورت ايستا در زمان ترجمه انجام دهند.

5-4- 2- كنترل نوع ايستا

كنترل نوع ممكن است در زمان ترجمه صورت گيرد مانند زبانهاي C,Pascal

مزاياي كنترل نوع ايستا :

- كنترل نوع ايستا تمام عمليات موجود در برنامه را شامل ميشود وتمام مسيرهاي اجرائي كنترل ميشوند لذا عمل چك كردن به بهترين صورت انجام ميگيرد و اشكال زدايي برنامه نيز ساده تر ميشود.
- عدم نياز به حافظه اضافي جهت نگهداري اطلاعات نوع داده اي در زمان اجرا
- افزايش سرعت اجراي برنامه

معايب كنترل نوع ايستا :

- انعطاف پذيري كم
- نياز به تعريف اعلان براي تمام اشياء داده

نكته : در كنترل نوع ايستا ، كنترل در زمان كامپايل انجام ميگيرد در گذر اول كامپايلر در جدول نمادها براي هر شي داده ، نوع و براي هر عمليات تعداد ،ترتيب و نوع آرگومانها را مشخص ميكند در گذر دوم عمل كنترل نوع صورت ميپذيرد.

اطلاعات مورد نياز در ارتباط با كنترل نوع ايستا معمولاً از اعلانهاي برنامه نويس يا ساختارهاي زبان گرفته مي -شود .بعضي از اين اطلاعات عبارتند از :

Debuging [1]

الف - براي هر عمل ، تعداد ، ترتيب، نوع آرگومان ورودي/ خروجي مشخص باشد. ب- براي هر متغير نام شئ داده نبايد در حين اجرا تغيير كند ج- نوع هر شي داده ثابت : هرثابت تعريف شده بايد با تعريفش سازگاري داشته باشد.

در مراحل ترجمه اين اطلاعات در جدول نمادها قرار ميگيرد.

امنيت نوع:

تابعي مانند ($f\ S \rightarrow R$:) (كه در آن S دامنه و R برد تابع f مي باشد) از نظر بررسي نوع ايمن است(امنيت نوع دارد) اگر اجراي تابعf نتواند مقداري خارج ازR را توليد كند. براي مثال اگر x,y از نوع صحيح Short باشند ممكن است عملگر * مقداري در خارج از دامنه اعداد صحيح Short توليد كند و لذا عملگر * بر روي اعداد صحيح Short ايمن نيست. از اين رو اگر همه عملگرهاي يك زبان برنامه نويسي از نظر بررسي نوع ايمن باشند آن زبان از نظر بررسي نوع قوي است.

بررسي نوع قوي:

اگر در يك زبان امكان انجام كليه بررسي نوعها به صورت ايستا وجود داشته باشد آن زبان داراي قابليت بررسي نوع قوي مي باشد. مانند زبان ML.

تعريف ديگر: اگر هر عملي در يك زبان امنيت نوع داشته باشد آن گاه زبان از نظر نوع قوي[1] است **نكته:** زبانهايي كه از كنترل نوع پويا استفاده ميكنند زبانهاي بدون نوع[2] خوانده ميشود .

استنتاج 7777نوعF_3 :

[1]
[2]
[3]

در زبان ML استنتاج نوع وجود دارد يعني اينكه پياده ساز زبان، اطلاعات نوعي را كه از قلم افتاده اند از ساير انواع تعريف شده استنتاج ميكند .

مثال (براي زبان ML) :

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

(**Fun area (len,wid):int=len*wid;** از روي خروجي ميفهمد كه ورودیها است)

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

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

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

Strong type
Type less
Inference type

مثالي ديگر:

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

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

وقتي نوع يكي از متغيرهاي x,y مشخص باشد ميتوان نوع دو مورد ديگر را با توجه به اينكه عمل + بين دو عدد صحيح يا دو عدد اعشاري صورت ميگيرد تشخيص داد.

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

اما در اين حالت چون نوع آرگوما نها مبهم است و همه ميتوانند float يا همه ميتوانند int باشند استنتاج نوع امكان پذير نيست.

5-4- 3- تبديل نوع[1] و تبديل ضمني[2]

اگر در زمان كنترل نوع ، نوع واقعي آرگومان و نوع مورد انتظار يكسان نباشد يكي از دوعمل زير صورت ميگيرد:

- برنامه Error ميگيرد وعمليات خاص مربوط به Error فراخواني ميشود(چاپ پيغام)
- عمل تبديل نوع انجام ميگيرد تا نوع آرگومان تغيير كند.

اغلب زبانها تبديل نوع را به دو صورت انجام ميدهند:

صريح : عمل تبديل نوع به صورت مجموعه اي از توابع توكار[3] توسط برنامه نويس فراخواني ميشود .

ضمني : عمل تبديل نوع به صورت خودكار توسط مترجم زبان انجام ميشود.

در زبان C هنگام جمع دو نوع int, float ، نوع int قبل از عمل جمع به طور ضمني تبديل به float ميشود .

- اگر در تبديل نوع ضمني ، اطلاعاتي از بين نرود تبديل گسترش يا ارتقا يافته ناميده ميشود.
- اگر در تبديل نوع ضمني ، اطلاعاتي از بين برود به آن تبديل ضمني محدودكننده[4] ميگويند.

نكته : درعمل تبديل نوع ممكن است لازم باشد در نمايش حافظه اي زمان اجراي يك شيء تغييرات گسترده اي صورت گيرد. به عنوان مثال در كوبول و PL/I اعداد به صورت رشته كاراكتري ذخيره ميشوند .براي جمع كردن اين اعداد در اغلب ماشينها، نمايش حافظه اي رشته كاراكتري بايد به نمايش دودويي كه توسط سخت افزار

---

[1] Conversion

[2] Coersion

[3] Built in Function narrowing

[4]

١
٢
٣

پشتيباني ميشود تبديل گردد.در هنگام برگرداندن نتيجه دوباره بايد از دودويي به كاراكتري تبديل شود در نتيجه ،عمليات تبديل نوع بيش از عمليات جمع صورت مي گيرد.

نكته: دو فلسفه متضاد در مورد وجود تبديل نوع ضمني در زبان وجود دارد. در پاسكال و Ada هيچ گونه تبديل نوع ضمني وجود ندارد. در C تبديل نوع ضمني صورت ميگيرد. تبديل نوع ضمني ، آزادي برنامه نويس را زيادتر ميكند مثلاً كامپايلر PL/I خطاهاي جزئي مثل اسامي نادرست متغيرها را ناديده ميگيرد .

معمولا 2 فلسفه متضاد در تبديل نوع وجود دارد (Type Mismatch) :

1. درپاسكال و Ada تقريبا هيچ گونه تبديل نوعي انجام نميشود هميشه خطا ميدهد.
2. درزبان C تبديل انواعها قانوني هستند مگر اينكه امكان هيچ گونه تبديلي وجود نداشته باشد كه بازهم خطا ميگيرد.

زبان PL1 به يافتن حداقل خطاهاي تبديل نوع و كلا خطاهاي برنامه نويسي مشهور است يعني سعي ميكند كه در اكثر موارد تبديل نوع را انجام دهد و خطا نگيرد.
درزبان PL1 عبارت 3/10+9 غيرمجاز است زيرا باتوجه به بالاتر بودن اولويت تقسيم به جمع ابتدا تقسيم شده و حاصل 33.3... با حداكثر ارقام اعشاري مجاز خود به دست ميآيد و در هنگام جمع اين عدد با 9 چون يك رقم اضافي ميآورد ، پس به خطاي Overflow منجر ميشود .

5-5- انتساب و مقداردهي اوليه

انتساب ، عمليات اصلي براي تغيير انقياد يك مقدار به يك شي داده است اين تغيير، اثرجانبي[1] عمليات محسوب ميشود .در بعضي زبانها مانند C,APL و ليسپ ، انتساب مقداري را بر ميگرداند كه اين مقدار يك شي داده اي است كه حاوي يك كپي از مقدار نسبت داده شده است ولي در زبان پاسكال عمل انتساب، مقداري را بر نميگرداند.

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

Assignment ( = ) : integer1 * integer2  integer3

Assignment : درپاسكال  Assignment : Type1 * Type2  void

Lisp , APL , C     Type * Type2  Type3
```

1 - يك كپي از مقدار موجود در integer2 در integer1 قرار ميگيرد. و نتيجه اي را برنميگرداند (تغيير integer1 يك اثر ضمني عمليات است)

2 - يك كپي از مقدار موجود درinteger2 درinteger1 قرار ميگيرد و شي داده integer3 ايجاد ميشود كه شامل مقدار integer2 است.

---

[1] Side effect

موضوع فوق باعث ميشود دستوري مانند A:=B:=C در زبان پاسكال خطا باشد ولي دستور A=B=C در زبان C درست بوده و مقدار C را به B و سپس به A نسبت ميدهد .

مثال) دستور انتساب روبرو را درنظر بگيريد:

```
x:=x
```

X سمت راست : به مقدار موجود در شي داده اي كه داراي نام x است مراجعه ميكند اين ارجاع را مقدار سمت راست (عملگر انتساب) يا مقدار راست[1] شي داده گويند.

X سمت چپ: به محلي از شي داده مراجعه ميكند كه حاوي مقدار جديد خواهد بود اين ارجاع را مقدار سمت چپ(عملگر انتساب) يا مقدار چپ[2] شي داده گويند.

عمليات انتساب A=B در چهار مرحله به صورت زير تعريف مي شود:

1 - مقدار چپ اولين عملوند را حساب كن

2 - مقدار راست دومين عبارت عملوند را حساب كن

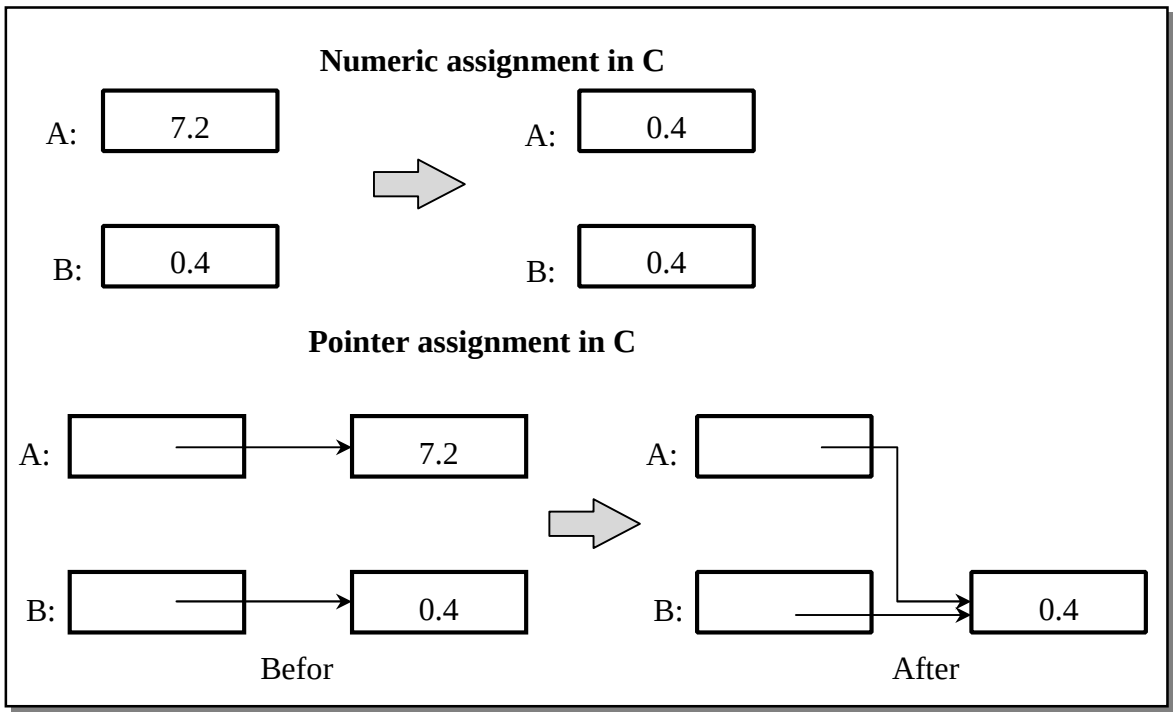
3 - مقدار راست محاسبه شده را به شي داده مقدار چپ نسبت بده.

4 - مقدار راست محاسبه شده را به عنوان خروجي برگردان.

مثال) انتساب A=B را درنظر بگيريد كه B و A در آن دو اشاره گر هستند اگر B يك اشاره گر باشد آن گاه مقدار راست B حاوي مقدار چپ شي داده ديگري است. بنابراين A=B يعني مقدار راست A به شي داده اي اشاره كند كه مقدار راست B به آن اشاره ميكند(.مقدار راست B را به مقدار چپ A نسبت بده كه مقدار راست B ، مقدار چپ شي داده ديگري است)

[1] Right-Value

[2] Left-Value



شكل 5 - 2

مقداردهي اوليه[1]F1[2]125125125125:

شي داده فاقد مقدار اوليه ، شي داده اي است كه ايجاد شده است ولي هنوز مقداري به آن داده نشده است. ايجاد يك شي داده به معناي اختصاص يك بلوك حافظه است. در بعضي زبانها مثل APL ، هر شي داده اي كه ايجاد ميشود بايد براي آن مقدار اوليه تعيين كرد. در بعضي ديگر از زبانها مثل پاسكال مقداردهي اوليه توسط دستور انتساب صورت ميگيرد .متغيرهاي فاقد مقدار اوليه عامل مهمي براي بروز خطا در برنامه نويسي است. از نظر قابليت اعتماد برنامه بهتر است بلافاصله پس از ايجاد متغيرها ، مقدار اوليه اي به آنها نسبت دهيم. مثلاً در زبان
Ada به همراه اعلان ميتوان مقداردهي اوليه انجام داد.

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

دو نوع مقدار دهي اوليه در زبانها وجود دارد:

صريح: كه در اين حالت برنامه نويس بايد دستورات لازم براي دادن مقدار اوليه به متغييرها را در برنامه وارد كند.

ضمني: كه در اين حالت خود كامپايلر مقدار اوليه متغيره را تعيين مي كند كه اين مقدار اوليه مي تواند صفر يا Null باشد.

[1] Initialization

تساوي و هم ارزي :

اگرچه دستور انتساب در اغلب زبانها وجود دارد ولي مشكلاتي در اين زمينه وجود دارد كه بايد برطرف شود.انتساب زير در زبان Zork را در نظر بگيريد:

A 2+3

براي زبانهايي كه انواع ايستا براي دادهها دارند نوع A مشخص مي كند از كدام معنا استفاده شود:

- اگر A از نوع صحيح باشد آنگاه مقدار 5 به A نسبت داده ميشود
- اگر A از نوع عمليات باشد آنگاه عمل "2+3" نسبت داده ميشود .

براي زبانهايي كه انواع در آنها به صورت پويا است و نوع A با انتساب مقدار به آن مشخص ميگردد هردو معنا قابل استفاده است و انتساب فوق باعث ابهام ميشود اين وضعيت دقيقاً در پرولوگ اتفاق ميافتد .

عملگر is : يعني مقدار هم ارز نسبت داده شود. ;x=5 , x is 2+3 (1 عملگر= : به معناي انتساب الگو است. ;x := 5 , x := 2 + 3 (2 مورد 1 درست است چون در x is 2+3 ، متغير x مقدار 5 ميگيرد و بعد با x=5 مقايسه ميشود حاصل درست است.

مورد 2 نادرست است چون در x = 2+3 ، معناي = به معناي انتساب الگوي 2+3 به متغير x است لذا وقتي با x=5 مقايسه ميشود حاصل نادرست است.

1

5-6- انواع داده _اسكالر_

انواع داده اسكالر فقط يك صفت دارند و از معماري سخت افزار كامپيوتر پيروي ميكنند مثل انواع داده char,int,float مثلاً شي نوع صحيح فقط داراي يك صفت مقدار صحيح (مثلاً 17و 81 و 42) است. انواع داده اسكالر شامل انواع صحيح- اعشاري- بولين – كاراكتر مي باشد. انواع داده مركب شامل چندين صفت هستند به عنوان مثال رشتهها شامل دنباله اي از كاراكترهاست ولي ممكن است صفت ديگري مانند طول رشته را داشته باشد دادههاي مركب ساختار پيچيده تري دارند كه معمولاً توسط كامپايلر پياده سازي مي شوند و نه با سخت افزار. انواع داده مركب شامل آرايه ها - رشتهها- فايل ها - اشاره گرها مي باشند.

5-6- 1- انواع صحيح

مشخصات:

۱

يك شي داده از نوع صحيح معمولاً صفتي غير از نوع ندارد و تنها شامل يك مقدار است. مجموعه مقادير ممكن براي انواع صحيح يك مجموعه ترتيبي متناهي از اعداد صحيح است. در زبان C چهار كلاس از نوع صحيح وجود دارد Int , long , short , char :

Scalar data type

عمليات:

عمليات روي نوع داده صحيح شامل موارد زير است:

1 - عمليات محاسباتي

$Binary-Op$:integer *integer → integer $\triangleright$ $MOD, , ,*,+ - DIV ,/$

$, , Unary-Op$:integer → integer $\triangleright$ $ABS ,++ --+ -,$

2 - عمليات رابطه اي

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

3 - عمليات انتساب

$assign$:integer *integer →$void$

$assign$:integer *integer → integer

4 - عمليات بيتي

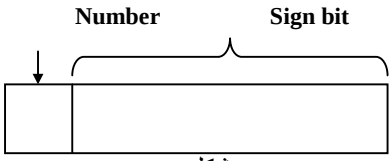
$,Bit-Op$:integer*integer →integer $\triangleright \&,|,\sim,\wedge,$

پياده سازي:

نوع داده صحيح توسط نمايش حافظه سخت افزار و مجموعهاي از عمليات محاسباتي و رابطه اي بر روي مقادير صحيح پياده سازي ميشوند. 3 نمايش حافظه اي براي نوع داده صحيح وجود دارد:

الف- بدون توصيفگر:

اين نمايش حافظه ، توصيفگر زمان اجرا ندارد و فقط مقدار در آن ذخيره مي شود .اين نمايش حافظه در زبانهايي استفاده ميشود كه زبان اعلانها و كنترل نوع ايستا را براي مقادير صحيح فراهم ميكند مانند C و فرترن

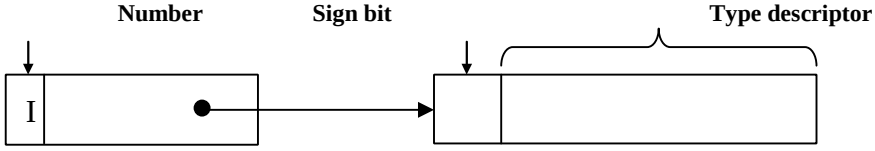


شكل 5 - 3

ب- توصيفگر در محل ديگر ذخيره شده:

اين نمايش حافظه، توصيفگر زمان اجرا دارد و توصيفگر در محل ديگري از حافظه ذخيره شده است كه اشارهگري به آن اشاره مي كند .اين نمايش حافظه در ليسپ استفاده مي شود .عيب آن اين است كه حافظه لازم براي شي داده صحيح دو برابر مي شود و مزيت آن اين است كه عمليات روي آن به صورت سخت افزاري قابل پياده سازي است.

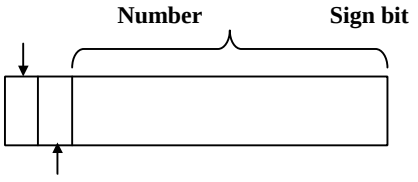
استفاده از نمايش سخت افزاري باعث افزايش سرعت عمليات مي شود.



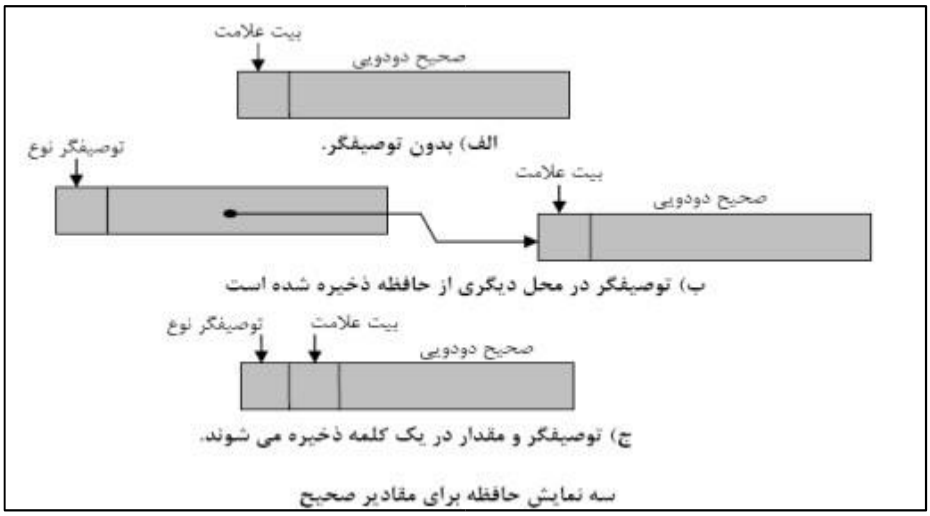
شكل 5 - 4

ج- توصيفگر و مقدار در يك كلمه:

توصيفگر نوع و مقدار در يك كلمه ذخيره مي شود لذا در حافظه صرفه جويي مي شود ولي براي استفاده عمليات سخت افزار بايد مقدار را از توصيفگر توسط دستورات شيفت از يكديگر جدا كرد لذا سرعت عمل كمتر است. (از روش ب بيشتر استفاده مي شود).



شكل 5 - 5



شکل 5 - 6

زیر 127بازهها1127F127:

مشخصات:

زير بازهها شامل دنباله اي از مقادير صحيح و بازه محدود هستند مانند نمونههاي زير در پاسكال و Ada: زير بازه ، زير نوع، نوع داده صحيح است.

Subrange

در پاسكال

```
A:1..50
```

Adaدر

```
A:integer rang 1..50
```

پياده سازي:

انواع زيربازه دو اثر مهم در پياده سازي دارد:

- **نياز به حافظه كمتر:** چون بازه كمتري از مقادير را شامل مي شود، مقادير زيربازه نسبت به مقادير صحيح معمولي، بيتهاي كمتري نياز دارند.
- **كنترل نوع بهتر:** اعلان يك متغيير از نوع زيربازه باعث مي شود كنترل نوع دقيق تري صورت گيرد. به عنوان مثال اگر متغيير Month به اين صورت باشد:Month:1..12 ، آنگاه دستور زير غلط است:

```
Month:=0
```

اين خطا در زمان كامپايل تشخيص داده مي شود. در بسياري از موارد كنترل نوع زيربازه ممكن نيست. به عنوان مثال، انتساب زير را در نظر بگيريد:

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

5-6- 2- اعداد حقيقي مميز

شناور[1] مشخصات:

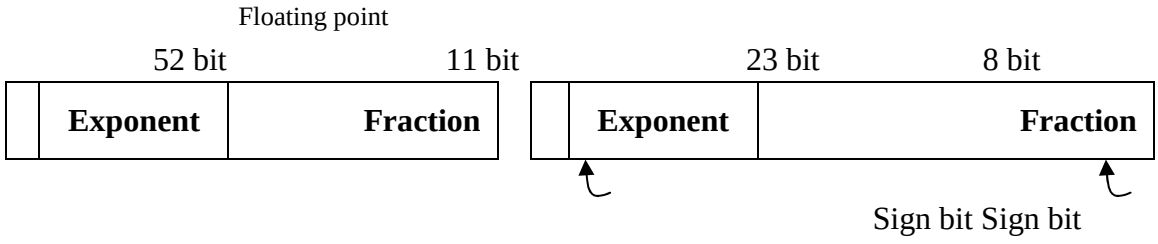
اين نوع داده معمولاً با صفت real در فرترن و پاسكال يا float در C مشخص ميشود .دقت مورد نياز براي اعداد مميز شناور، كه تعداد ارقام در نمايش دهدهي است توسط برنامه نويس مشخص مي گردد مثل زبان Ada عمليات محاسباتي، رابطه اي ، انتساب مشابه اعداد صحيح، براي اعداد حقيقي هم امكان پذير است ولي به علت مسائل مربوط به گرد كردن، كمتر دو عدد حقيقي با هم مساوي ميشوند .بنابراين در اين حالت حلقههايي كه براي تست كردن از اعداد حقيقي استفاده ميكنند ممكن است در حلقه دائم بيافتد. لذا گاهي اوقات تساوي بين دو عدد حقيقي توسط طراح زبان جلوگيري مي شود .عمليات كتابخانه اي ديگر:

Sin:real→ real

پياده سازي max:real×real→real :

۱

در اكثر زبانها نحوه پياده سازي اعداد حقيقي مميز شناور به سخت افزار بستگي دارد براي ذخيره و پياده سازي اعداد مميز شناور از استاندارد IEEE754 استفاده مي شود براي اين كار از فرمتي شبيه نماد علمي استفاده مي كنيم. كه استانداردIEEE754 استاندارد 23 و 64 بيتي را براي اعداد مميز شناور مشخص مي كند فرمت 23 بيتي و 64 بيتي به صورت زير است:



شكل 5 - 7

استاندارد 23 بيتي:

در استاندارد 23 بيتي هر عدد حقيقي مميز شناور شامل سه فيلد است:

بيت S : فيلد علامت يك بيتي كه صفر به معناي مثبت بودن است.

بيت E : توان ظاهري 8 بيتي با افزودني 127، يعني در هنگام ذخيره سازي توان در حافظه 23 بيتي، مقدار 721 به آن افزوده شده و سپس ذخيره ميگردد در بازه 0 تا 552 كه معادل توان دو از 128- تا 127 است.

بيت M: مانتيس 32 بيتي است معمولا اعداد مميز شناور را به صورت نرمال شده ذخيره ميكنند در مبناي 2، عدد نرمال شده بايد با ارزشترين بيت قسمت اعشار 1 باشد براي مثال 0/00111 نرمال نيست ولي 0/11011 نرمال است. اولين بيت مانتيس در عدد نرمال شده هميشه 1 است.

علامت عدد را مشخص ميكند و با توجه به مقادير E, M مقدار دقت به صورت زير است.

پارامتر	مقدار
E = 255 and M ≠ 0	عدد نامعتبر
E = 255 and M = 0	∞
0 < E < 255	2^{E-127} (1.M)
E = 0 and M ≠ 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…

5-6- 3- اعداد حقيقي مميز

مشخصات ثابت[1]:

اغلب سخت افزارها شامل اشياي داده صحيح و مميز شناور هستند. براي برخي از دادههاي حقيقي اگر از مميز شناور استفاده كنيم خطاي گردكردن اتفاق خواهد افتاد. مي توان براي اينگونه دادهها از مميز ثابت استفاده كرد.

Fixed point

پياده سازي:

ممكن است مستقيماً توسط سخت افزار پشتيباني شود يا به صورت نرم افزاري شبيه سازي گردد. اينگونه اعداد به صورت صحيح ذخيره مي شوند و نقطه اعشار به عنوان صفت آن شي داده اي است. اگر مقدار X برابر 01/124 باشد، مقدار راست X برابر 10421 و شي X حاوي صفتي به نام فاكتور مقياس برابر 3 مي باشد و معنايش اين است كه سه رقم بعد از نقطه اعشار قرار دارد. يعني با توجه به فرمول $Value\ X( ) = rvalue\ X( ) \times 10^{-SF}$ SF صرف نظر از مقدار راست X، همواره برابر 3 است.

در نهايت به طور خلاصه خواهيم داشت:

- **اعداد اعشاري**

اعداد integer ميتواند دامنه محدودي را قبول كند. اعداد اعشاري با توجه به اين كه از دوقسمت پايه و توان تشكيل شده اند ، دامنه متغيري را پوشش ميدهند ولي درعين حال دقت خيلي بالايي نخواهند داشت (زيرا ممكن است شرط تساوي دو عدد اعشاري برقرار نشود.) اين نوع سخت افزار پشتيباني ميشود).

عمليات روي آن مشابه عمليات روي اعداد integer ميباشد فقط برخي از عمليات ممكن است توسط نرم افزار شبيه سازي شود مانند عمليات به توان رساندن و ...

---

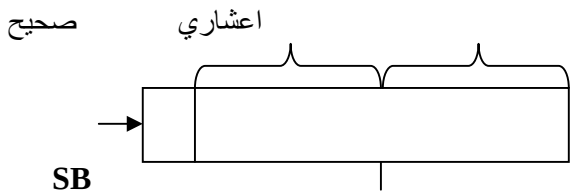
[1]

ازجهت پياده سازي روشهاي زير را دارد :

1. FixedPoint

مثلا درزبان cobol اعلان داده اعشاري مميز ثابت با عبارت picture نشان داده ميشود .

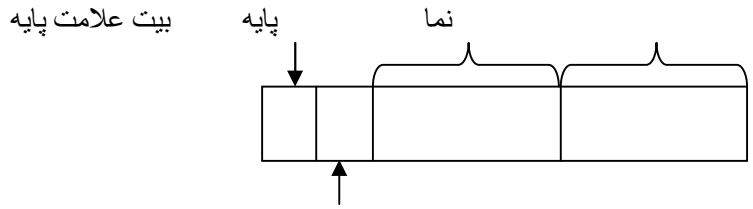
Picture 999 V 99



نقطه اعشاري فرضي

شكل 5 - 8

2. Floating Point مانند نماد علمي



شكل 5 - 9

در زبان Ada ميتوان تعداد ارقام دقت عدد اعشاري را توسط برنامه نويس مشخص نمود.

نمونه اي از اعداد اعشاري مميز ثابت در زبان PL1 به صورت ازير است:

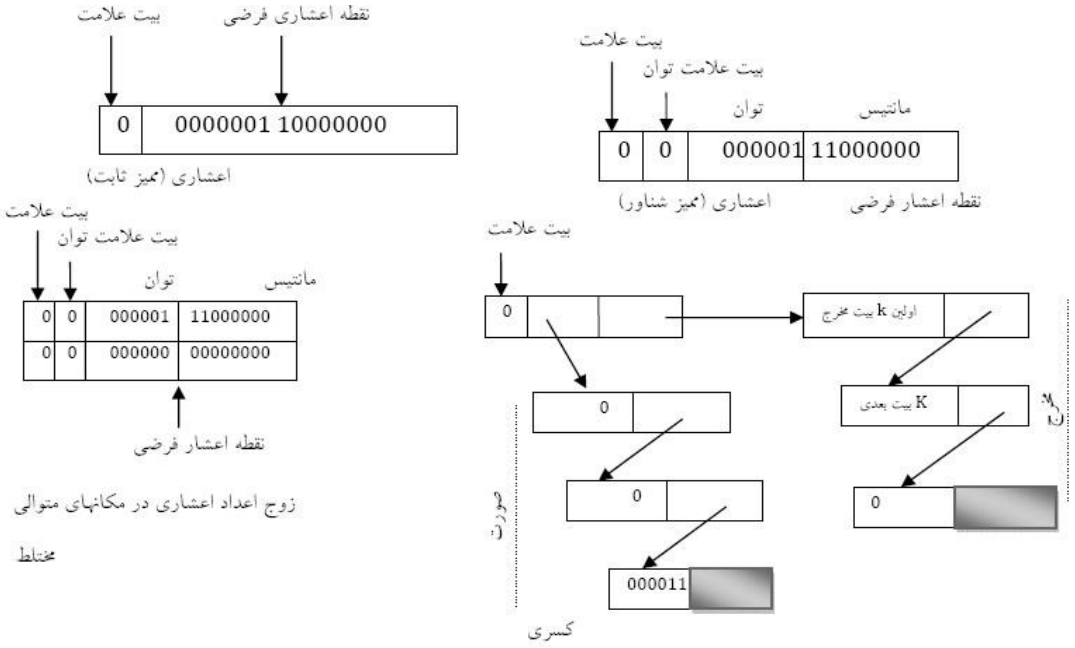
DECLARE x Fixed DECIMAL (1 , 3) ;

تعداد ارقام اعشار تعداد ارقام صحيح

ساير انواع داده عددي:

اعداد موهومي: عدد موهومي متشكل از يك جفت از اعداد است كه يكي از آنها بخش حقيقي و ديگري بخش موهومي را نشان مي دهد.

اعداد گويا:عدد گويا خارج قسمت دو عدد صحيح است.



شكل 5 - 01

5-6- 4- نوع شمارشي

مشخصات:

مقادير نوع شمارشي بر اساس تعريف برنامه نويس مشخص مي شوند كه ليست مرتبي از مقادير مجزاست. مقادير نوع شمارشي به نام ثوابت شمارشي نيز خوانده مي شوند. برنامه نويس اسامي ليترالهايي را كه بايد براي مقادير ،مورد استفاده قرار گيرند و همچنين ترتيب آنها را با استفاده از اعلاني مشخص مي كند. نمونه اي از نوع شمارشي و تعريف متغييرهايي از آن نوع در C# به صورت زير است:

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

اين تعريف ليترالهايFresh,Soph,Jonior,Senior را نيز تعريف مي كند كه در هرجايي از برنامه قابل به كارگيري و استفاده مي باشند.

عمليات اصلي روي نوع داده شمارشي عبارتند از: عمليات رابطه اي(=, <, >)، انتساب و عمليات Successor(بعدي) و Predecessor(قبلي) كه به ترتيب عناصر قبلي و بعدي را مشخص مي كنند.

پياده سازي:

نمايش حافظه براي شي داده اي از نوع شمارشي بسيار ساده است. هر مقدار در دنباله شمارشي در زمان اجرا به وسيله مقادير صحيح 0،1،2،... قابل نمايش است. چون فقط مجموعه كوچكي از مقادير در نوع شمارشي وجود دارد و مقادير منفي نيستند، نمايش آنها از مقادير صحيح نيز ساده تر است. بعنوان مثال، نوع Class كه در بالا تعريف شد ،فقط چهار مقدار ممكن دارد كه در زمان اجرا به صورتSenior=3,Jonior=2,Soph=1,Fresh=0 نمايش داده مي شود. در زبان C مي توان اين ترتيب را تغيير داد.

5-6- 5- نوع بولي

مشخصات:

متشكل از اشياي داده اي است كه يكي از دو مقدار TRUE يا FALSE را مي پذيرد. متداولترين عمليات روي نوع داده بولين عبارتند از:and,or,xor,nor,not and **پياده سازي:**

نمايش حافظه براي شي داده بولي يك بيت از حافظه است، به شرطي كه نياز به توصيفگر براي نوع دادهها، نباشد. چون يك بيت در حافظه قابل نمايش نيست معمولاً از يك واحد قابل آدرس دهي مانند بايت يا كلمه استفاده مي شود .مقادير True يا False به دو روش در اين واحد حافظه نمايش داده مي شوند .

- بيت خاصي براي اين مقادير استفاده مي شود، به طوريكه False=0 و True=1 و بقيه بيتهاي مربوط به بايت يا كلمه بدون استفاده مي ماند.
- مقدار صفر در كل واحد حافظه(بايت يا كلمه) نشان دهنده False و مقدار غيرصفر نشان دهنده True است.

بعضي از زبانها مثل C فاقد نوع بولي اند.در اين زبان،از نوع داده صحيح براي اين منظور استفاده مي شود.مقدار صفر نشان دهنده False و مقدار غير صفر نشان دهنده True است.دستورات زير را ببينيد:

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

چون flag برابر با صفر نيست، به عنوان True استفاده مي شود. اگر انتساب flag = 0; را داشته باشيم ، flagداراي ارزش False خواهد بود. اين روش استفاده از مقادير صحيح به جاي مقادير بولي، اشكالاتي دارد. به عنوان مثال، اگر به جاي and منطقي (&&) از and بيتي (&) يا به جاي or منطقي (| |) از or بيتي (|)، يا به جاي نقيص منطقي (!) از نقيص بيتي (~) استفاده كنيد، با مشكل مواجه خواهيد شد. دستورات زير را ببينيد

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

نمايش بيتي عدد 21 در يك كلمه 61 بيتي به صورت زير است كه ارزش درستي دارد:

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

اگر آن را نقيص بيتي كنيد (found~) نمايش بيتي آن به صورت زير است كه بازهم ارزش درستي دارد:

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

بنابراين نتيجه مي گيريم كه found و نقيض بيتي ؟ آن هر دو ارزش درستي دارند. در صورتي كه بخواهيد واقعاً نقيض found ارزش نادرستي داشته باشد، بايد آن را نقيض منطقي كنيد (found!) در اين صورت نمايش بيتي foundبه صورت زير خواهد بود كه ارزش نادرستي دارد:

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

5-6- 6- كاراكترها

مشخصات:

نوع داده كاراكتري اشياي داده را به وجود مي آورد كه مقدار آنها يك كاراكتر است. مجموعه اي از مقادير كاراكتري ممكن ، معمولاً به صورت نوع شمارشي تعريف شده در زبان در نظر گرفته مي شود.

پياده سازي:

مقادير دادههاي كاراكتري هميشه توسط سيستم عامل و سخت افزار پشتيباني مي شوند. اگر نمايش كاراكتري كه توسط زبان تعريف شد، با نمايش كاراكتري كه توسط سخت افزار پشتيباني مي شود، يكسان باشد آنگاه عمليات رابطه اي نيز مستقيماً در سخت افزار نمايش داده مي شوند يا توسط كدهاي نرم افزاري شبيه سازي خواهد شد.

5-7- انواع داده مركب

5-7- 1- رشتههاي كاراكتري

مشخصات و نحو:

نكات طراحي رشته

1. آيا رشتهها بايد به صورت آرايه اي از كاراكترها باشند يا به صورت نوع داده اوليه؟ دقت كنيد كه اگر رشته-ها به صورت آرايه اي از كاراكترها باشند، عمليات انديس گذاري بر روي آن امكان پذير است، ولي اگر به صورت نوع داده اوليه باشد، امكان پذير نيست.
2. طول رشتهها بايد ايستا يا پويا باشد؟

نكته : در زبانهاي ML,Prolog نوع داده رشته اي مستقيماً ارائه شده است ولي در زبانهاي C,Ada,Pascal رشته كاراكتري را به عنوان آرايه خطي از كاراكترها در نظر مي گيرند.

با رشتههاي كاراكتري از نظر طول ، سه گونه برخورد مي شود:

- **طول ثابت (ايستا):** طول رشته در هنگام ايجاد رشته مشخص مي گردد.اشياي تغيير ناپذير مربوط به كلاس String در Java ، C + + و C# از اين دسته اند. دقت كنيد كه منظور از شيء تغيير ناپذير اين است كه وقتي ايجاد شد، قابل تغيير نيست.
- **طول متغير با حد معين (طول پوياي محدود):** شيء داده رشته كاراكتري ممكن است طول حداكثري داشته باشد كه در برنامه اعلان شود.
- **طول متغير (طول پويا):** طول رشتهها مي تواند در زمان اجرا تغيير كنند و حداكثر طول براي آن مشخص نمي شود. JavaScript و Perl از اين نوع رشتهها استفاده مي كنند. اين نوع رشتهها داراي سر بار تخصيص و آزاد سازي حافظه اند، ولي قابليت انعطاف آنها زياد است.

رشتههاي كاراكتري در زبان C كمي پيچيده تر مي باشند. در انتهاي رشته بايد كاراكتر تهي ('0/') قرار گيرد و برنامه نويس بايد اطمينان حاصل كند كه رشتهها به تهي ختم مي شوند.

عمليات گوناگوني بر روي رشتهها امكان پذير است كه بعضي از آنها عبارتند از:

- C در strcat مانند(Concatenation) الحاق رشتهها
- عمليات رابطه اي در رشتهها مانند strcmp در C
- انتخاب زير رشته با استفاده از انديس
- فرمت بندي ورودي-خروجي
- انتخاب زير رشته با تطابق الگو
- رشتههاي پويا
- انتخاب زير رشته در رشته اصلي مانند strstr در C **پياده سازي :**

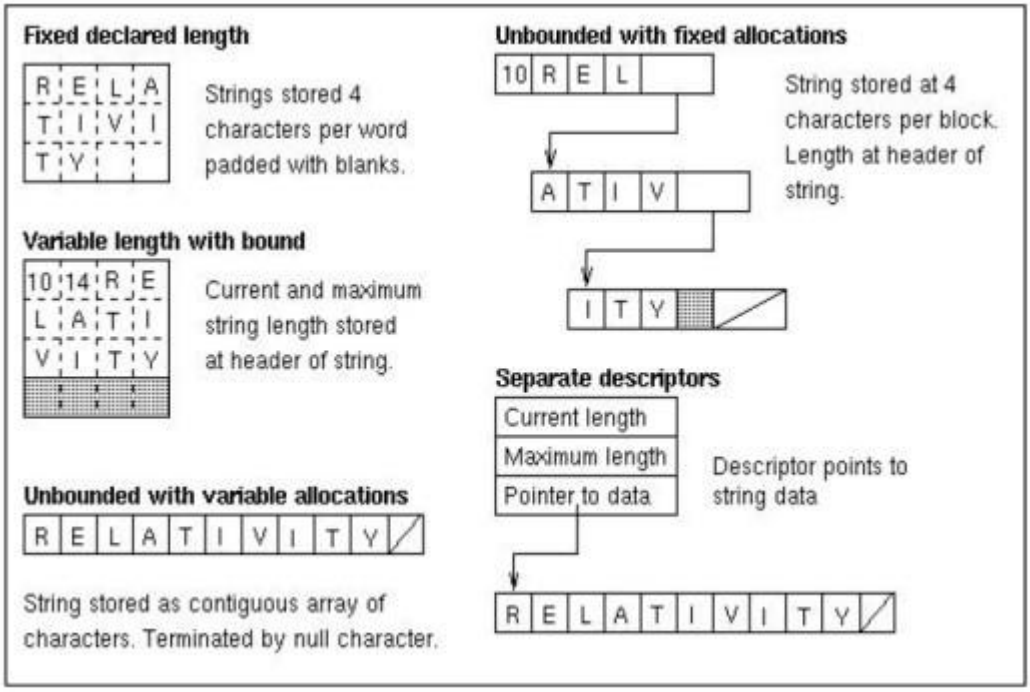
براي رشته اي با طول ثابت نمايش حافظه همان شكلي است كه براي بردار فشرده اي از كاراكترها استفاده شد. براي رشته طول متغيير با حد معين نمايش حافظه از توصيفگري استفاده مي كند كه

حاوي حداكثر طول و طول فعلي ذخيره شده در شي داده است. براي رشتههاي نامحدود مي توان از نمايش حافظه پيوندي اشيا داده طول ثابت استفاده كرد.

- طول ثابت (بالا،سمت چپ ،اولين سطر) : هر 4 كاراكتر در يك كلمه ذخيره مي شود و بقيه طول رشته با فضاي خالي پر مي شود.
- طول متغيير با حد معين (بالا،سمت چپ ،دومين سطر) :حداكثر طول و طول رشته در ابتدا ذخيره مي شود
- طول نامحدود با تخصيص ثابت (بالا،سمت راست،اولين سطر) : 4 كاراكتر در هر بلوك ذخيره مي شود و طول در ابتداي رشته قرار مي گيرد.
- طول نامحدود با تخصيصهاي متغيير (پايين،سمت راست) : رشته به صورت آرايه ي پيوسته كاراكترها

،ذخيره مي شود و انتهاي هر رشته با nall يا تهي مشخص مي شود.

توضيحات مربوط به موارد فوق را در شكل زير مشاهده مي كنيد .



شكل 5 - 11

5-7- 2- اشاره گرها و اشياء داده برنامه نويس

معمولاً در هر زبان براي اتصال اشياء داده به يكديگر از اشاره گر استفاده مي شود. زبان برنامه نويسي بايد ويژگي-هاي زير را در مورد اشاره گر داشته باشد :1- نوع داده اوليه اشاره گر 2 – عمل ايجاد[1] 3- عمل انتخاب[2]

- **نوع داده ي اوليه اشاره گر :** شي داده اشاره گر شامل آدرس شي داده ديگري است يعني شامل مقدار چپ يك شي داده ديگر است.
- **عمل ايجاد كردن :** براي اشياء داده با طول ثابت مانند آرايه، ركورد، انواع داده ي اوليه. عمل ايجاد كردن بلوكي از حافظه را براي شي داده جديد ايجاد مي كند و مقدار چپ آن را برمي گرداند.
- **عمليات دستيابي :** اين عمل باعث مي شود تا محتويات جايي كه اشاره گر به آن اشاره مي كند دستيابي شود **مشخصات :**

نوع داده اشاره گر دسته اي از اشياء داده را تعريف مي كند كه مقادير آنها آدرسهاي اشياء ديگر است و به دو روش با آنها برخورد مي شود :

الف : اشاره گرها ممكن است فقط به يك نوع شي داده مراجعه كنند : اين روش در پاسكال ,C ،Ada، استفاده مي شود. كه در آنها اعلان نوع و كنترل نوع ايستا ممكن است.

مثال (`int *p;`

* نوع p را اشاره گر معرفي مي كند. نوع int مشخص مي كند،كه مقدار p مي تواند مقدار چپ يك شي داده از نوع

. باشد int

ب : اشاره گر ممكن است به هر نوع شي داده مراجعه كند : اين روش در زبانهايي مانند اسمالتاك استفاده مي شود، كه اشياء داده در حين اجرا ، داراي توصيفگر هستند و كنترل نوع پويا انجام مي شود.

مثال (`void *p`

* نوع p را اشاره گر معرفي مي كند. نوع void مشخص مي كند،كه مقدار p مي تواند مقدار چپ يك شي داده از هر نوعي باشد.

عمليات ايجاد كردن : حافظه را براي شي داده طول ثابت تخصيص مي دهد و اشاره گري به اين شي داده جديد ايجاد مي كند كه در يك شي داده اشاره گر ذخيره مي شود.

در C++,Ada ,pascal new :

creation

[1] selection

[2]

C در: malloc : **مثال**

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

يك بلوك حافظه دو كلمه اي را ايجاد كن تا به عنوان شي داده اي از نوع int مورد استفاده قرار گيرد. و مقدار چپ آن را در p ذخيره كن.

عمليات انتخاب كردن :

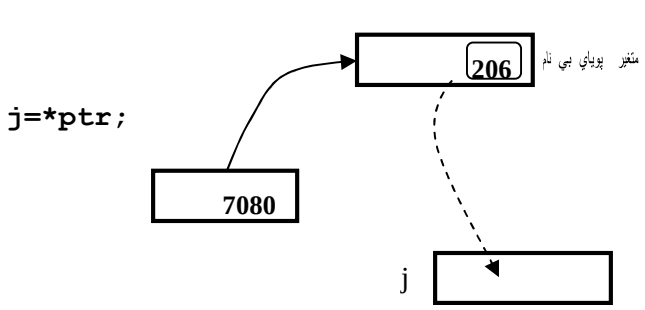
اجازه مي دهد تا مقدار اشاره گر دنبال شود تا به شي داده ي مورد نظر برسيم. در زبان C عمل انتخاب با * مشخص مي شود.

به عنصر first از ركوردي دستيابي دارد كه p به آن اشاره مي كند `*p.first` عمليات مهمي كه در مورد اشاره گرها مي توان انجام داد عبارتند از:

1-عمليات انتساب

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

2-عمليات دستيابي به محتويات



3-عمليات محاسباتي و رابطه اي

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

مثال در سه زبان مختلف :

زبان C	زبان C++	زبان پاسكال

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

جدول 5 - 1

پياده سازي :

شي داده اشاره گر به صورت محلي از حافظه نمايش داده مي شود كه شامل آدرس محل ديگري از حافظه است.اين آدرس ، آدرس پايه بلوك حافظه نشان دهنده شي داده است كه اشاره گر به آن محل اشاره مي كند دو نمايش حافظه براي مقادير اشاره گر استفاده مي شود :

الف : آدرس دهي 132132132132**مطلق**F_1 **:** مقدار اشاره گر ممكن است آدرس واقعي بلوك حافظه مربوط به شي داده باشد.

ب : آدرس 1111**نسبي**F_2 **:** مقدار اشاره گر ممكن است آفستي از آدرس پايه بلوك حافظه هرم^3 باشد كه شي داده در آن ايجاد شده است.

مزاياي و معايب آدرس دهي مطلق :

انتخاب و دسترسي به شي داده از طريق آدرسهاي مطلق كارآمدتر است چون از عمليات سخت افزاري براي دستيابي به شي داده استفاده مي كند .عيب آدرس دهي مطلق اين است كه مديريت حافظه مشكل تر مي شود. زيرا شي داده در حافظه جابجا نمي شود.عيب ديگر اين است كه بازيابي حافظه از اشياء داده اي كه به صورت دادههاي زباله در آمده دشوار است زيرا هر يك از اين اشياء داده به صورت منفرد بازيابي مي شوند.

مزايا و معايب آدرس دهي نسبي :

استفاده از آدرس نسبي به عنوان اشاره گر ، مستلزم تخصيص بلوكي از حافظه است كه new تخصيص ديگري در آن انجام دهد مزيت آدرس دهي نسبي اين است كه مي تواند بلوك حافظه را در هر زمان به نقاط دلخواهي از حافظه حركت داد مزيت ديگر اين است كه با كل ناحيه اي كه به هنگام ورود به زيربرنامه ايجاد مي شود مي توان به صورت يك شي داده برخورد كرد. در داخل اين ناحيه

نيازي به بازيابي حافظه براي تك تك اشياء داده نيست چون كل ناحيه به هنگام خروج از زير برنامه بازيابي مي شوند.

[1] Absolute address

[2] Relative address heap ۳

مقايسه آدرس دهي نسبي و مطلق به طور خلاصه :

- در مورد آدرس دهي مطلق كارايي بالا وجود دارد. سرعت اجراي برنامه بالا مي باشد چون آدرس فيزيكي در حافظه است. ولي در روش نسبي كارايي برنامه پايين است چون براي دسترسي به شي داده محاسبه بايد توسط offset و Base صورت گيرد.
- در روش آدرس دهي مطلق شي داده را نمي توان انتقال داد در حاليكه در روش نسبي مي توان شي داده را به راحتي انتقال داد.
- عمليات ترميم[1] (حل مشكل زباله و ارجاع سرگردان) در مورد آدرس دهي مطلق به سختي صورت مي گيرد ولي در روش نسبي راحت تر صورت مي گيرد.

5-7- 3- فايلها و ورودي-خروجي

فايل ساختمان داده اي با دو ويژگي مهم مي باشد :

1 -بر روي حا فظه ثانويه مانند ديسك يا نوار تشكيل مي شود و ممكن است بسيار بزرگتر از ساختمان دادهها باشد.

2 - طول عمر آن مي تواند بسيار بزرگ باشد.

انواع متداول فايلها عبارتند از:.

- متداولترين فايلها، فايلهاي ترتيبي[2] اند.
- فايلهاي دستيابي مستقيم[3]
- فايلهاي ترتيبي شاخص دار[4]
- فايلهاي متني[5]

متداول ترين فايلها، فايلهاي ترتيبي اند. اما، اغلب زبانها از فايلهاي دستيابي مستقيم و فايلهاي ترتيبي شاخص دار استفاده مي كنند. دو كاربرد مهم فايلها عبارتند از: ورودي/ خروجي دادهها در محيط عمليات خارجي و حافظه موقت در مواقعي كه حافظه كافي وجود ندارد. عناصر فايل را ركورد گويند ولي در اينجا از اين اصطلاح صرف نظر مي كنيم تا با ساختمان داده ركورد(كه در فصل بعد اشاره خواهد شد) اشتباه نشود.

فايلهاي ترتيبي:

فايل مي تواند در حالت خواندن يا در حالت نوشتن دستيابي شود. در هر دو حالت يك اشاره گر موقعيت فايل وجود دارد كه موقعيتي را قبل از اولين عنصر فايل، بين دو عنصر فايل، يا بعد از آخرين عنصر فايل تعيين مي كند. در حالت نوشتن، اشاره گر موقعيت فايل، هميشه به بعد از آخرين عنصر فايل اشاره مي كند و مي توان عنصري را در آن محل نوشت و فايل را به اندازه يك عنصر بسط داد. در حالت خواندن، اشاره گر موقعيت فايل، مي تواند در هر

---

[1] recovery

[2] sequential

[3] Direct access

[4] Indexed sequential files text °

نقطه اي از فايل باشد و مي توان عمل خواندن را در آن محل انجام داد. در اين حالت نمي توان عنصر جديدي ر ا اضافه كرد. در هر دو حالت پس از عمل خواندن يا نوشتن، اشاره گر موقعيت فايل حركت مي كند تا به موقعيت عنصر بعدي اشاره كند. اگر اين اشاره گر بعد از آخرين عنصر فايل قرار گيرد مي گوييم به انتهاي فايل رسيديم .

عمليات اصلي بر روي فايلهاي ترتيبي عبارتند از:

- بازكردن
- خواندن
- نوشتن
- تست انتهاي فايل
- بستن

فايلهاي دستيابي مستقيم:

در فايل ترتيبي، عناصر به ترتيبي كه در فايل قرار دارند بازيابي مي شوند. اگر چه عمليات محدودي براي جابجايي اشاره گر موقعيت فايل وجود دارد ولي در اين فايلها، دستيابي تصادفي به عناصر، غير ممكن است. فايل دستيابي مستقيم، طوري دستيابي مي شود كه مي توان به هر عنصر به طور تصادفي دست يافت(مثل آرايه و ركورد). انديسي كه براي انتخاب يك عنصر به كار مي رود، كليد نام دارد كه ممكن است يك مقدار صحيح يا شناسه ديگري باشد. اگر يك مقدار صحيح باشد انديس معمولي است كه براي تعيين عنصري از فايل به كار مي رود، اما چون فايل دستيابي مستقيم، در حافظه ثانويه ذخيره مي شود، پياده سازي فايل و عمليات انتخاب، متفاوت از آرايه است.

فايل ترتيبي شاخص دار:

فايل ترتيبي شاخص دار، شبيه فايل دستيابي مستقيم است به طوريكه امكان دستيابي ترتيبي، با شروع از عنصري كه به طور تصادفي انتخاب شد، وجود دارد. به عنوان مثال، اگر عنصري با كليد 72 انتخاب(خوانده) شود، عمليات خواندن بعدي، ممكن است عنصر بعدي را به ترتيب انتخاب كند.(به جاي دادن مقدار كليد). اين سازمان فايل ،مصالحه اي را بين سازمانهاي ترتيبي محض با دستيابي مستقيم محض به وجود مي آورد.

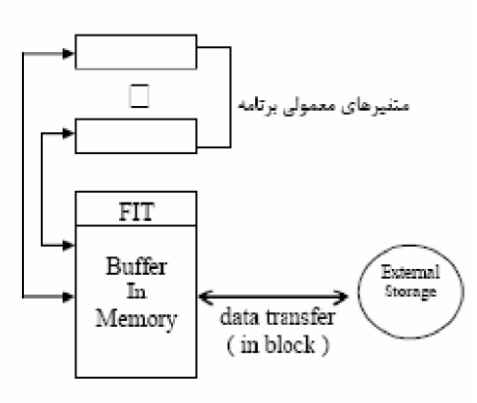
پياده سازي فايل:

ازجهت پياده سازي مسئول انجام همه عمليات مربوط به فايلها و مديريت آنها به عهده سيستم عامل ميباشد. يك زبان برنامه سازي بايد مكانيزمهايي جهت برقراري ارتباط با برنامه نويس و سيستم عامل داشته باشد.

ازديد جزئي تر وقتي يك فايل باز ميشود، فضايي به نام FIT (File Information Table) و همچنين فضايي به عنوان بافر براي هر فايل تخصيص مييابد .

FIT : دراين جدول نام فايل ، تاريخ آخرين به روز رساني و اندازه فايل و مشخصات ديگر از جمله محلهاي خواندن و نوشتن روي دستگاه ذخيره و بازيابي اطلاعات نگه داشته ميشود .

Buffer : اين حافظه به عنوان يك صف براي خواندن و نوشتن عمل ميكند، درصورت پر يا خالي شدن بافر ،اطلاعات به دستگاه خارجي انتقال داده ميشود و يا از آن به داخل بافر منتقل ميشود .



شکل 5 - 21

5-8- سوالات فصل پنجم

سوالات تستي

1-كداميك از موارد زير جزء اهداف اعلان نيست؟(نيمسال اول 58 -68) الف.عمليات وراثت ب.عمليات چند ريختي ج.مديريت حافظه د.انتخاب نمايش حافظه

2 -كداميك از عمليات زير جزء عمليات اصلي مربوط به دادههاي شمارشي نيست؟(نيمسال اول 58 -68) الف.عمليات رابطه اي ب.عمليات انتساب ج.عمليات جبري مثل جمع و تفريق د.عمليات پيداكردن عنصر بعدي و قبلي **3-**كدام گزينه غلط مي باشد؟(نيمسال دوم 58-68) الف.هر عمليات روي داده، يك دامنه و يك بازه دارد.

ب.هر عمليات روي داده، يك تابع رياضي نمي باشد.

ج.اعلان دستوري از برنامه است كه نام و نوع اشياي داده را كه در حين اجراي برنامه مورد نياز هستند را مشخص مي نمايد.

د.كليه موارد بالا

4-انقيادهاي يك شي داده كدام است؟(نيمسال دوم 58 -68)

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

5-كداميك از زبانهاي زير از چند ريختي حمايت مي كند؟(نيمسال دوم 58 -68) الف.C ب.اسمالتاك ج.ML د.JAVA

6- روش پياده سازي رشتههاي كاراكتري كدام يك از موارد زير است؟(نيمسال اول 68 -78) الف.رشته اي با طول ثابت

ب.رشته با طول متغير و حد بالا مشخص ج.رشته با طول نامحدود

د.همه موارد صحيح مي باشد.

7- كداميك از موارد زير جزء اهداف اعلان است؟(نيمسال اول 68-78) الف.انتخاب نمايش حافظه ب.عمليات چندريختي ج.كنترل نوع د.همه موارد

8-چه نمايش حافظه اي براي مقاديرصحيح، در زبانهاي Late binding مناسب تر است؟(نيمسال اول 68-78) الف.بدون توصيف كننده زمان اجرا ب.با توصيف كننده در يك كلمه جداگانه ج.با توصيف كننده در همان كلمه د.هيچكدام

9-در مورد كنترل نوع پويا كدام گزينه غلط است؟(نيمسال اول 68-78)

الف.در هر عمليات كنترل نوع صورت مي گيرد و در صورتي اجرا مي شود كه انواع آرگومان درست باشد.
ب.لازم نيست هر عمليات به نتايج خود يك نوع را نسبت دهد تا عمليات بعدي بتواند آنها را كنترل كند.
ج.كنترل نوع پويا در زمان اجرا انجام مي شود.
د .در كنترل نوع پويا در هر شي داده يك برچسب قرار مي گيرد كه نوع آن را مشخص مي كند.

01- اهداف اعلان كداميك از موارد زير است؟(نيمسال دوم 78-68)
الف.انتخاب نمايش حافظه
ب.عمليات چند ريختي ج.كنترل نوع
د.همه موارد **11-** كدام گزينه صحيح است؟(نيمسال دوم 78-68)
الف.كنترل نوع پويا حافظه بيشتري نسبت به كنترل نوع ايستا مصرف مي كند.
ب.اگر تمام خطاهاي نوع را به طور ايستا رفع كنيم زبان را نوع قوي گويند.
ج.در كنترل نوع پويا براي كاهش برخي هزينهها ممكن است عمليات كنترل نشوند.
د.هر سه گزينه صحيح است.
21- كداميك از مفاهيم زير كمتر با هم سازگارند؟(نيمسال دوم 68-78)
الف.تعريف نوع متغيرها و ترجمه ب.Static Scope Rrule وترجمه
ج Late binding.د Early binding,Dynamic Scope Rule. و تفسير
31- كدام يك از گزينههاي زير دلايل استفاده از زير بازه به جاي نوع داده اي صحيح مي باشد؟(نيمسال دوم 86 - 87)
الف.عمليات محاسباتي ساده تر ب.كنترل نوع بهتر
ج.استفاده از عمليات successorوpredecessor د.پياده سازي مستقيم با سخت افزار
41- اگر عملياتي ساختار داخلي اعم از دادههاي محلي كه در بين اجراهاي مختلف نگهداري مي شوند يا كد خود را اصلاح كند اين خاصيت را چه گويند؟(نيمسال اول 78 -88)
الف.خودراني ب.دانه اصلاحي ج.خود اصلاحي د.كد اصلاحي
51- مهمترين هدف اعلانها از ديدگاه برنامه نويس كدام است؟(نيمسال اول 78 -88)
الف.كنترل نوع ايستا به جاي كنترل نوع پويا ب.كنترل نوع پويا به جاي كنترل نوع ايستا
ج.مديريت حافظه ايستا به جاي مديريت حافظه پويا د.مديريت حافظه پويا به جاي مديريت حافظه ايستا
61- با توجه به قطعه كد زير، براي عمل جمع در x+y و انتساب نتيجه محاسبه شده x+y در x به ترتيب از راست به چپ كدام تبديل ضمني صورت مي گيرد؟(نيمسال اول 78-88)

```
int x; float
        y;
        .
        .
x=x+y
```

الف.گسترش و گسترش ب.باريك كننده و باريك كننده ج.گسترش و باريك كننده

د.باريك كننده و گسترش

71- در قطعه برنامه زير چه تعداد ثابت وجود دارد؟(نيمسال دوم 78-88)

```
Cons t int Max =30;
      Int  N;
       N=27;
N= N +Max;
```

الف. 1 ب2. ج3. د 4 .

81- به چه زباني نوع قوي مي گويند؟(نيمسال دوم 78-88) الف. تمام خطاهاي نوع به طور پويا بر طرف شود.

ب .تمام خطاهاي نوع به طور ايستا برطرف شود.

ج .استنتاج نوع وجود داشته باشد.

د .اعلان نوع جديد وجود داشته باشد.

91- كدام يك از اعلانهاي زير در زبان ML بر اساس استنتاج نوع رفع ابهام نمي گردد و نتيجه معتبر نخواهد بود؟(نيمسال دوم 78-88)

الف Fun area (length, width): int=length *width .

بFun area (length: int, width) =length* width .

ج. Fun area (length, width: int) =length*width

د. Fun area (length, width) =length*width

02- در كدام دو زبان زير هيچ گونه تبديل ضمني وجود ندارد؟(نيمسال دوم 78-88)

الف C, C++ .ب C, Ada.ج C++, Pascal.د Ada, Pascal.

12- در استاندارد IEEE 754 كه اعداد مميز شناور را با سه فيلد تعريف مي كند. اگر يك بيت براي علامت (S) و براي نما هشت بيت و براي بيت مانتيس 32 بيت در نظر گرفته شود و اعداد به صورت نرمال شده مد نظر باشند آنگاه براي $0<E<255$ كداميك از اعداد زير در حالت كلي در نظر گرفته مي شود؟(نيمسال دوم 78-88)

الف $2^{E-127}(1.M)$.ب $2^{-126}(0.M)$.ج. 2^{E} د).1 $2^{E-127}(0.M)$.

22- براي رشته اي با 01 كاراكتر و نمايش حافظه اي به صورت طول متغير با حد معين اگر بزرگترين رشته اي كه مي تواند در آن طول قرار بگيرد 41 فرض شود آنگاه نمايش مربوطه چند خانه از حافظه را رزرو مي كند؟(نيمسال دوم 78 -88)

الف14: ب16: ج10: د12:

32- در كدام گزينه همه اشياي داده اي ذكر شده، اشياي داده اي هستند كه كامپيوتر مجازي آنها را ايجاد مي كند تا در حين اجراي برنامه از آنها براي ذخيره دادهها استفاده كند و مستقيماً در اختيار برنامه نويس نيستند؟(نيمسال اول 88-98)

الف.ثوابت متغيرها ، آرايهها

ب.متغيرهاو آرايهها ، پشتههاي زمان اجرا ، بافرهاي فايل

ج.پشتههاي زمان اجرا و، ركوردهاي فعاليت زيربرنامهها، بافرهاي فايل ،

ليستهاي فضاي آزاد د.فايلها ، بافرهاي فايل ، آرايهها ، ثوابت

42- كداميك از موارد زير جزو اهداف اصلي اعلان محسوب نمي شود؟(نيمسال اول 88-98) الف.تعيين مقدار شي داده ب.انتخاب نمايش حافظه

ج.مديريت حافظه د.عمليات چند ريختي

52- از گزينههاي زير كداميك را مي توان به عنوان مزيت اصلي كنترل نوع به روش پويا در نظرگرفت؟(نيمسال اول 88-98)

الف.سهولت در اشكال زدايي برنامهها ب.استفاده كمتر از حافظه ج.انعطاف در طراحي برنامه د.افزايش سرعت اجراي برنامه

62- در كدام گزينه هر دو زبان داراي تبديل نوع ضمني هستند؟(نيمسال اول 88-98)

PL/I و.Adaد Pascal و.C ج PL/I و.C بAda و.Pascal الف

72- در نمايش حافظه براي رشتهها در كدام نمايش زير رشته به صورت آرايه پيوسته اي از كاراكترها ذخيره مي شود و انتهاي رشته با تهي مشخص مي گردد؟(نيمسال اول 88 -98)

الف.طول نامحدود با تخصيصهاي متغير ب.طول نامحدود با تخصيص ثابت ج.طول ثابت د.طول متغير با حد معين

82- صفات شي داده كه نام توصيفگر يا descriptor به آنها داده مي شود، در چه زماني ذخيره مي شوند؟ (نيمسال دوم 88 -98)

الف.زمان اجرا ب. زمان ترجمه

ج.زمان تعريف زبان د .مي تواند در زمان ترجمه يا اجرا ذخيره شوند.

92- قطعه برنامه زير نشان دهنده وجود كداميك از عوامل زير مي باشد؟(نيمسال دوم 88 -98)

```
Int    func(){
Static  int   i=0;
i++;    return I;
} int main(){  for(int
i=0;i<=10;i++)
cout<<func();
return
{;0الف.وجود اثر جانبي  ب.وجود آرگومان ضمني  ج.وجود سرريز  د.وجود
```

خوداصلاحي **03-** كدام عمليات زير براي دستور a:=b*c، يك اثر جانبي مي باشد؟ (نيمسال دوم 88-98) الف.عمل انتساب ب.عمل ضرب

ج .عمل ضرب و انتساب هر دو د. در اين دستور اثر جانبي وجود ندارد

13- با توجه به قطعه كد زير چه نوع خطايي و در چه زماني رخ داده و يا ممكن است رخ دهد؟(نيمسال دوم 88 - 89)

```
Day =1..30;
Day:=0;
For i:=1  to  20 do
Day:=day+2;
```

الف.كنترل نوع زمان كامپايل و اجرا ب. كنترل نوع زمان اجرا ج .كنترل نوع زمان كامپايل د. كنترل نوع زمان تعريف زبان

23- قطعه برنامه زير به كداميك از امكانات موجود در زبان ML،اشاره دارد؟ (نيمسال دوم 88 - 98)

```
Fun Pnu(r):float=3.14*r*r;
```

الف.كنترل نوع ايستا ب.تبديل نوع ج.استنتاج نوع د.كنترل نوع پويا

33- درقطعه كد C زير، دستور f=a+b/c درعمليات a+b/c بدون عمليات انتساب، تبديل ضمني در چه نوع انجام مي شود؟(نيمسال دوم 88 -98)

```
int main() {      float f;     int a,c;
                                double b;
f=a+b/c;        return 0; }
```

ج د.خطاfloat. الف int.بDouble.

43- در تعريف زير وجود آرگومان سراسري g استفاده شده در تابع، نشانگر چيست؟ (نيمسال دوم 88-98)

```
F ( int a,int b) {
   a=10;    b=a+b;
             g=b;
              }
```

الف.اثر جانبي ب.خود اصلاحي ج.نتايج ضمني د.آرگومان ضمني

53- كداميك از روشهاي پياده سازي، براي رشتههاي كاراكتري با طول نامحدود در زبانهاي با تكنولوژي جديد پشتيباني مي شود؟ (نيمسال دوم 88 -98)

الف.آرايه پيوسته اي از كاراكترها ب.نمايش

حافظه پيوندي ج .نمايش حافظه ترتيبي

د.فشرده كردن رشته

63- تكه كد برنامه زير به كدام مورد اشاره دارد. (نيمسال اول 98 -09)

```
Int funct (int &a,int &b)
          int    {
                 m;
        m=a;
      b=a+b;
   return m;
          }
```

الف. آرگومانهاي ضمني ب .حساسيت به سابقه قبلي (گذشته) ج

.آرگومانهاي خاص د .اثرات جانبي

73- با توجه به تكه كد زير چه نوع خطايي و در چه زماني رخ داده و يا ممكن است رخ دهد. (نيمسال اول 89 -

90)

```
Const int k=0; for
   i:=1 to 20 do
    k:=k+2;
```

الف. كنترل نوع زمان كامپايل ب .كنترل نوع زمان اجرا و اجرا

ج .كنترل نوع زمان كامپايل د .كنترل نوع زمان تعريف زبان

سوالات تشريحي

1-چه عواملي باعث مي شوند كه تعريف عمليات زبان برنامه سازي به صورت تابع رياضي دشوار شود؟(نيمسال اول 86-85)

2-عمليات اصلي بر روي فايلهاي ترتيبي را شرح دهيد؟(نيمسال دوم 58-68)

3-سه نوع نمايش حافظه براي مقادير صحيح را رسم كنيد و در مورد هريك توضيح دهيد؟(نيمسال دوم 68 -78)

4-نمايشهاي حافظه را براي مقادير حقيقي مميز ثابت و حقيقي مميز شناور و موهومي و گويا رسم كنيد و هر يك را بطور مختصر خط توضيح دهيد؟(نيمسال اول 78 -88) **5-**تبديل نوع و انواع آن را شرح دهيد؟(نيمسال دوم 78 -88)

6- مهمترين هدف اعلان چيست ؟ آن را به طور كامل توضيح دهيد.(نيمسال اول 98 -09)

5-9- پاسخنامه سوالات تستي فصل پنجم

سوال	الف	ب	ج	د
1	*			
2			*	
3		*		
4		*		
5			*	
6				*
7				*
8		*		
9		*		
10				*
11				*
12				*
13		*		
14			*	
15	*			
16			*	
17			*	
18		*		
19				*
20				*

سوال	الف	ب	ج	د
21	*			
22		*		
23			*	
24	*			
25			*	
26		*		
27	*			
28	*			
29				*
30	*			
31	*			
32			*	
33		*		
34				*
35	*			
36				*
37			*	

فصل ششم:

بسته بندی

ا # ا :

مشخصات انواع ساختمان دادهها
پياده سازي دادههاي ساخت يافته
نمايش حافظه
پياده سازي عمليات مديريت
حافظه و مسئله اشاره گرها اعلان و
كنترل نوع ساختمان دادهها بردارها و
آرايهها برش آرايه
آرايههاي انجمني
ركوردها ركوردهاي تودرتو
ركورد با طول متغيير ليستها
مجموعهها
اشياء داده اجرايي
نوع داده انتزاعي(A.D.T)

پنهان سازي و بسته
بندي اطلاعات زيربرنامهها
تعريف و فراخواني زيربرنامه
زيربرنامههاي كلي
تعريف زيربرنامه
به عنوان شي داده تعريف
نوع هم ارزي نوع
سوالات تستي و تشريحي

1-6- مقدمه

به چهار طريق مي توان انواع داده جديد و عمليات بر روي آنها را تعريف كرد :

- **ساختمان دادهها**[1] **:**از نظر مجازي تمام زبانها خواصي براي ايجاد اشياء داده پيچيده از انواع اوليه دارند ساختمان داده ، شي داده اي است كه عناصر آن خود اشياء داده ديگري هستند. دادههاي ساخت يافته پشتيباني سخت افزاري ندارند و بيش از يك صفت دارند. براي توصيف هرگونه ساختمان داده به يك تو صيفگر[2] نيازاست كه مشخصات آن را تعيين كند. ليستها ، مجموعهها ، آرايهها براي ايجاد گروهي از اشياء داده همگن و ركوردها مكانيزمي براي ايجاد اشياء داده غير همگن هستند.

- **زير برنامهها :** برنامه نويس مي تواند برنامههايي را بنويسد كه مانند يك نوع جديد عمل كنند. همچنين در برخي از زبانها عمليات به عنوان يك نوع جديد محسوب مي شود.
- **اعلان نوع:** خود برنامه نويس با استفاده از امكانات زبان يك نوع جديد تعريف مي كند مفهوم نوع داده انتزاعي براي ايجاد انواع جديد به كار برده مي شود. مانند كلاس در C++ و package در Ada
- **وراثت:** به كمك تكنيكهاي شي گرايي و وراثت مي توان انواع جديد و عمليات روي آنها را تعريف كرد.

دراين فصل به سه مورد اول مي پردازيم :

ساختمان داده :

شي داده اي كه مركب از چند شي داده ديگر است ساختمان داده نام دارد عناصر ساختمان داده را اجزاي آن مي نامند كه هر جزء آن مي تواند يك عنصر اوليه يا ساختمان داده ديگر باشد. مانند آرايهها ، ركوردها ، پشتهها ، ليستها و مجموعهها.

2-6- مشخصات انواع ساختمان دادهها

- **تعداد اجزاء :** اگر تعداد عناصر ساختمان داده درطول عمرش ثابت باشد اندازه ساختمان داده ثابت و گرنه متغيراست آرايهها وركوردها ساختمان دادههايي با اندازه ثابت هستند پشته ، ليست و مجموعهها نمونههايي از ساختمان داده طول متغير هستند رشتهها به هر دو شكل ثابت و متغيرمي توانند وجود داشته باشند. اشياء داده طول متغير با استفاده از اشاره گرها، اشياء داده طول ثابت را به هم پيوند مي دهند.
- **نوع هر عنصر :** اگر همه عناصر ساختمان داده از يك نوع باشند به آن همگن و در غير اين صورت، آن را ناهمگن گويند آرايهها ، مجموعهها ، رشتهها از نوع همگن و ركوردها و ليستها عموماً ناهمگن هستند • **اسامي براي انتخاب عناصر :** هر ساختمان داده بايد مكانيزمي داشته باشد كه بتوان اجزاء آنرا انتخاب كرد مثلاً درآرايه به كمك انديس ، هر عنصرانتخاب مي شود. دررركورد اين نام توسط برنامه نويس مشخص مي شود. در پشته به وسيله اشاره گر top و در فايل به وسيله اشاره گر فايل seek

---

[1] Data structure

[2] descriptor

- **حداكثر تعداد عناصر:** براي ساختارهايي با طول متغيرمثل رشته يا پشته حداكثر طول آن بر حسب تعداد عناصر بايد مشخص شود.
- **سازمان عناصر :** متداولترين سازمان ، دنباله خطي از عناصر است مانند آرايههاي يك بعدي ، ركوردها ،رشتهها و ...در يك فضاي پيوسته از حافظه ذخيره مي شوند.

عمليات در ساختمان دادهها :

عمليات كلي كه در انواع ساختمان داده انجام مي گيرند عبارتند از:

- **عمليات انتخاب عنصر :** دو نوع عمليات انتخاب وجود دارند كه به اجزاي ساختمان دادهها دستيابي دارند انتخاب تصادفي (مستقيم[1]) و انتخاب ترتيبي[2]. در انتخاب تصادفي اجزاء به صورت دلخواه انتخاب مي شوند ولي در انتخاب ترتيبي اجزاء به ترتيبي كه از قبل مشخص شده اند دستيابي مي شوند.به عنوان مثال در پردازش يك بردار از عمليات انديس براي دستيابي تصادفي به اجزاء استفاده مي شود. (v[4]) يا ركورد همراه با اسم جزء (R.ID) ولي درليستها بايد پردازش ترتيبي صورت گيرد تا به عنصر مورد نظر برسيم.
- **عمليات روي كل ساختمان داده :** عمليات ممكن است كل ساختمان دادهها را به عنوان آرگومان بپذيرند و ساختمان داده جديدي را توليد كنند مانند جمع دو آرايه ، انتساب ركوردي به ركورد ديگر يا عمليات اجتماع بر روي مجموعهها .زبانهايي مانند APL و اسنوبال4 تعداد زيادي از اين عمليات را پشتيباني مي كنند.
- **درج وحذف عناصر :** عملياتي كه تعداد عناصر ساختمان داده را تغيير مي دهند.
- **ايجاد و حذف ساختمان دادهها:** عملياتي كه ساختمان دادهها را ايجاد يا حذف مي كند.

نكته : بين عمليات ارجاع و عمليات انتخاب تفاوت است.v[4] عمليات ارجاع ، موقعيت خطي نام v را تعيين مي كند عمليات انتخاب دقيقاً مكان آن عنصر را نشان مي دهد.

3-6- پياده سازي انواع ساختمان دادهها

اين پياده سازي از دو جنبه نمايش حافظه و پياده سازي عمليات روي ساختمان دادهها بررسي مي شود.

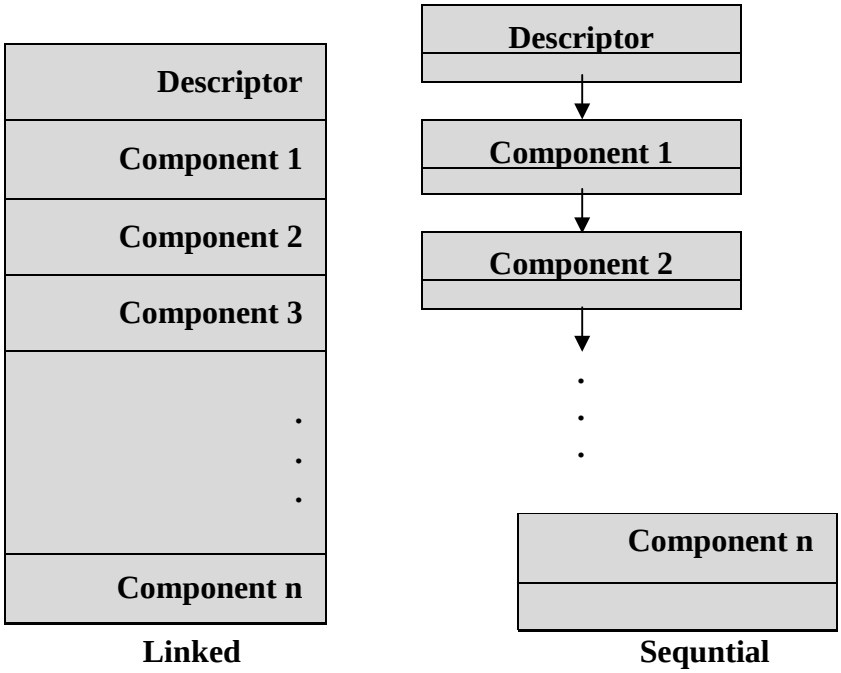
3-6- 1- نمايش حافظه

هنگام ذخيره سازي ساختمان داده ، حافظه اي براي عناصر ساختمان داده و توصيفگر اختياري كه تمام يا چند صفت ساختمان داده را ذخيره مي كند در نظر گرفته مي شود. دو نمايش حافظه اي براي ساختمان دادهها وجود دارد :

[1] Direct
[2] Sequential

الف- ترتيبي : در اين نمايش ساختمان داده در يك بلوك پيوسته از حافظه ذخيره مي شود كه شامل توصيف گر و اجزاء مي باشد.

ب- پيوندي : در اين نمايش ساختمان داده در چندين بلوك حافظه ناپيوسته ذخيره مي شوند كه بلوكها از طريق اشاره گر به يكديگر پيوند خواهند خورد. اشاره گر از بلوك A به بلوك B پيوند نام دارد.



شكل 6 - 1 نمايش حافظه براي ساختمان دادههاي خطي

نمايش ترتيبي براي ساختمان داده با طول ثابت و گاهي ساختمان داده با طول متغيير همگن (رشتهها و پشتهها) استفاده مي شود. نمايش پيوندي اغلب براي ساختمان دادههايي با طول متغيرمثل ليستها به كارگرفته مي شود.

6-3- 2- پياده سازي عمليات

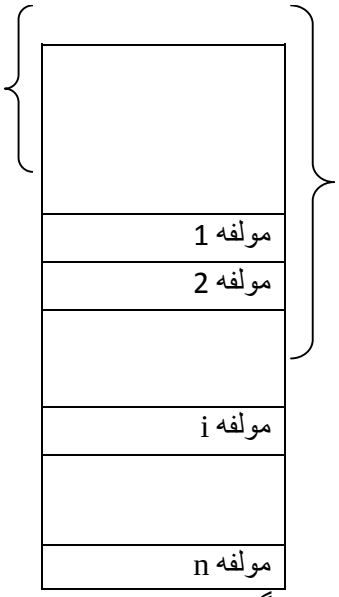
انتخاب اجزاء ساختمان داده مهمترين نكته در پياده سازي آن است و هدف آن است كه انتخاب ترتيبي و تصادفي عناصر كارامد باشند.

الف- انتخاب ترتيبي در حافظه : عمليات انتخاب عنصر در نمايش ترتيبي حافظه ، به كمك يك آدرس مبنا و يك آدرس آفست به صورت (Base+offset) به سادگي و با سرعت انجام مي شود محل نسبي عنصر انتخاب شده در بلوك ترتيبي ، افست و محل شروع بلوك، آدرس پايه نام دارد.

ب- انتخاب پيوندي در حافظه : براي انتخاب يك جزء در ساختمان داده در روش پيوندي مي بايست تمام عناصر اول تا i ام براي رسيدن به عنصر i ام دستيابي شوند.

BaseAddress (a)

D



Offset (i) = D + (i-1)*Component size offset

= a * offset (i) ام i آدرس مولفه

شكل 6 - 2

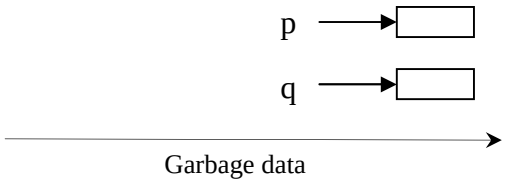
6-4- مديريت حافظه و مسائل اشاره گرها

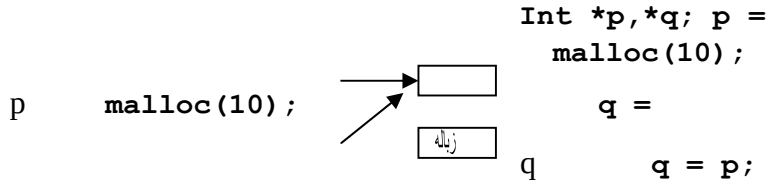
هنگامي كه به يك شي داده حافظه تخصيص داده مي شود طول عمر آن شروع مي شود و هنگامي كه اين انقياد از بين برود طول عمر آن هم تمام مي شود. براي ساختمان دادههاي با طول متغير، هر يك از عناصر ، طول عمر مخصوص به خودشان را دارند ولي در ساختمان داده طول ثابت ، كل ساختمان داده داراي يك طول عمر است. هنگام ايجاد يك شي داده يك مسير دستيابي به آن نيز ايجاد مي شود اين مسير دستيابي يا از طريق نام براي آن صورت مي گيرد يا به كمك اشاره گري كه به آن اشاره مي كند. در هر نقطه از طول عمر شي داده اي ، ممكن است چندين مسير دستيابي به آن وجود داشته باشد مثلا ً آرگوماني به زير برنامه ارسال مي شود يا اشاره گر جديدي به آن اشاره مي كند

(ارجاع سرگردان) يا مسيرهاي دستيابي ممكن است به روشهاي گوناگوني از بين بروند مثل انتساب مقدار جديدي به متغير اشاره گر (زباله.)

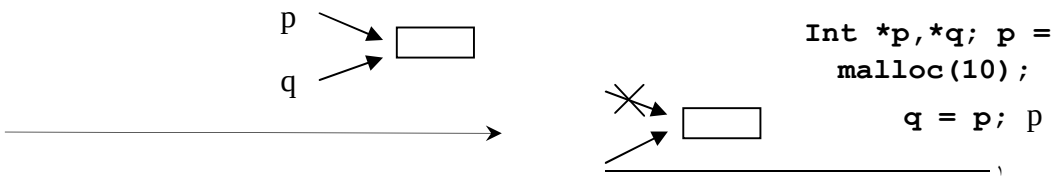
بنابراين طول عمر شي داده و مسيرهاي دستيابي اثرات متقابلي با هم دارند و در اين زمينه دو مسئله مهم در مدیریت حافظه وجود دارد :

الف-دادههاي زباله^{F_1}^{1111}: هنگامي كه يك مسير دستيابي به يك شي داده از بين برود ولي خود شي داده وجود داشته باشد ، شي داده را زباله گوييم زيرا ديگر قابل استفاده نيست ولي انقياد آن به محل حافظه اش از بين نرفته است مثل تكه كد زير :





ب-ارجاع معلق^{F_1}^{145145145145} : اگر طول عمر شي داده تمام شود ولي هنوز يك مسير دستيابي به آن را داشته باشيم مي گوييم ارجاع معلق رخ داده است در حقيقت ارجاع سرگردان يك مسير دستيابي است كه پس از اينكه طول عمر شي داده خاتمه يافت، وجود داشته باشد. اين مشكل هنگامي رخ مي دهد كه چند اشاره گر به صورت همزمان به آن اشاره مي كنند توسط يكي از آن اشاره گرها آزاد شود در برنامه استفاده شده و از بين نروند.

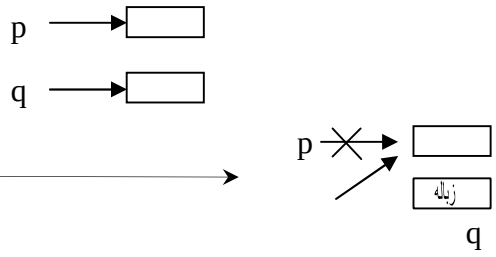


```
free(p);
```
q

حال اگر دستور free(p) را اجرا كنيم حافظه مذكور رها شده ولي هنوز p به آن اشاره مي كند.

نكته : ممكن است دو مشكل فوق همزمان در برنامه رخ بدهد. مثال زير :

```
Int *p,*q;
p = malloc(10);
q = malloc(10);
q = p;
free(p);
```



مسئله دادههاي زباله چندان مهم نيست چون دادههاي زباله حافظه مصرف مي كنند و باعث به هدر رفتن حافظه مي شوند و حداكثر يك برنامه به دليل عدم حافظه كافي اجرا نخواهد شد ولي ارجاع معلق مسئله مهمي در مديريت حافظه است كه اگر از طريق ارجاع سرگردان به يك شي داده دسترسي داشته باشيم باعث ناهنجاري اطلاعاتي مي شود .

يك راه حل جهت رفع مشكل زباله استفاده از فيلد اضافي به نام Reference count است زماني كه به يك اشاره گر فضايي را اختصاص مي دهيم به صورت خودكار فيلد مذكور ساخته مي شود كه مقدار درون اين فيلد ،بيانگر تعداد اشاره گرهايي است كه به آن خانه حافظه اشاره مي كند با اضافه شدن اشاره گري به اين محل حافظه ، يك واحد به اين اشاره گر افزوده مي شود و با آزاد شدن هر كدام از اين اشاره گرها يك واحد از آن كم مي شود.

dangling reference

هرگاه count=0 شد يعني هيچ اشاره گري به آن اشاره نكرده و بنابراين فضاي مربوطه آزاد مي شود اين عمل آزادسازي توسط واحدي به نام جمع آوري زباله[1] صورت مي گيرد.

5-6- اعلان وكنترل نوع ساختمان دادهها

يك اعلان داده ساخت يافته ، صفات متعددي را براي آن مشخص مي سازد مثلا ً اعلان زير در پاسكال :

```
A=array [1..20,-4..8]of real;
```

۱

مشخص مي كند كه نوع داده آرايه اي است دو بعدي و انديس سطرها از يك تا بيست و انديس ستونها از 4- تا 8 است تعداد سطرها 02 و تعداد ستونها 31 و در نتيجه كل خانهها 062 عدد است و نوع هر خانه نيز اعشاري است.

درهنگام كنترل نوع ساختمان داده دو موضوع مهم بايد در نظر گرفته شود :

الف- وجود مولفه انتخابي : جزء انتخابي ممكن است در ساختمان داده نباشد به عنوان مثال عمليات انديس كه جزئي از آرايه را انتخاب مي كند ممكن است انديس خارج از محدوده آرايه باشد. كنترل زمان اجرا بايد معتبر بودن آنرا بررسي كند.

مثال `A[1..10]` (`A[13]` غيرقانوني

ب- نوع عنصر انتخابي : اگر عنصري وجود داشته باشد بايد نوع آن هم درست باشد. مثلا ً در عبارت روبرو در زبان C : A[2][3].link->item هنگام ترجمه بايد نوع عنصري كه از اين طريق بدست مي آيد مشخص و درست باشد اين كنترل به صورت ايستا و در زمان كامپايل انجام مي گيرد.

6-6- بردارها و آرايهها

مشخصات:

بردار (آرايه اي يك بعدي) ساختماني مركب از تعداد ثابتي از عناصر هم نوع است كه به شكل دنباله خطي سازمان دهي شده اند. آرايههاي دو بعدي ماتريس نيز به همين روش تعريف مي شوند هر بردار داراي تعدادي صفات نظير:

تعداد عناصر ، نوع عناصر ، انديس براي انتخاب هر عنصر است.

- **تعداد عناصر :** معمولا ً توسط بازه كه براي انديس مشخص مي شود تعيين مي گردد.
- **نوع هر عنصر :** تمام عناصر بردار از يك نوع هستند.
- **انديس براي انتخاب هر عنصر**

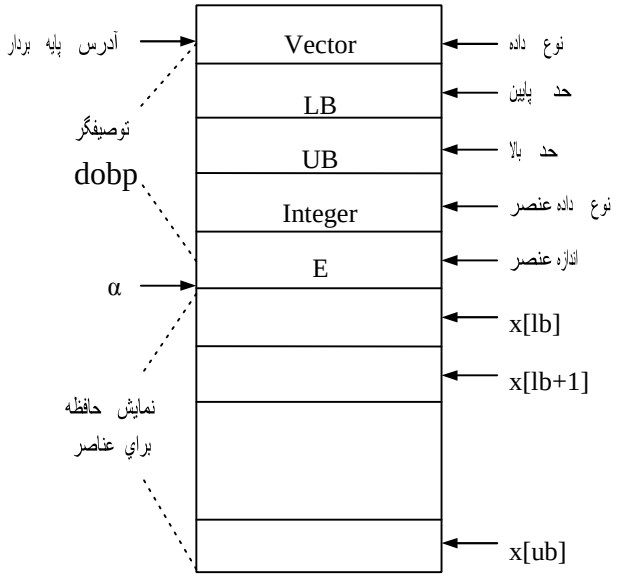
ذخيره نمودن آرايههاي تك بعدي (بردار) در حافظه كامپيوتر به سادگي انجام مي گيرد. اما در آرايههاي دو بعدي (ماتريس) و چند بعدي مسئله بغرنج تر خواهد شد يكي از وظايف كامپايلرها ، نگاشت آرايه دو بعدي وعناصر آن به آدرس فيزيكي حافظه است كه براي اين كار نيازمند محاسبات ويژه اي خواهيم بود.

Garbage Callection

۱

عمليات روي بردارها :

عملياتي كه عنصري را از برداري انتخاب مي كند انديس گذاري[1] نام دارد مثل A[2]. عمليات ديگر بردارها عبارتند از: ساخت و از بين بردن آنها ، انتساب به عناصري از بردار و اعمالي مثل جمع دو بردار كه روي بردارهايي با طول يكسان انجام مي گيرند. در اكثر زبانها عمليات بر روي بردارها بسيار محدود است ولي در APL اين عمليات بسيار زياد مي باشند.



شكل 6 - 3 نمايش توصيفگر كامل براي بردار A

پياده سازي :

آدرس $Ai[\,]=\alpha+(i-LB)\times E$

با شروع از عنصر اول براي رسيدن به i امين عنصر بايد از i-1 امين عنصر قبلي عبور كرد. اگر E اندازه هر عنصر باشد بايد از $(i-1)\times E$ محل حافظه عبور كنيم اگر اولين عنصر در محل α باشد و LB حد پايين انديس باشد فرمول دستيابي براي مقدار چپ يك عنصر بردار به صورت زير است :

$$Lvalue(A(i))=\alpha+(i-LB)\times E=\alpha+i\times E-LB\times E=(\alpha-LB\times E)+i\times E$$

كه در آن $(\alpha-LB\times E)$ مقدار ثابتي است.

مثلا ً در فرترن در زمان ترجمه مقدار x مشخص است : $Lvalue(A(i))=$

$x+i\times E$ در C براي آرايههاي كاراكتري مقدار E برابر 1 و=0 LB است.

۱

فرمول دستيابي عنصر آرايه كاراكتري : $Lvalue(A(i))=(\alpha+0\times E)+i\times E=\alpha+i$ در

پاسكال انديس از يك شروع مي شود ودر C از صفر شروع مي شود.

subscripting

مبدا مجازي :

آدرس عنصري با انديس صفر را مشخص مي كنيم فرمول به صورت زير در مي آيد.

$$Lvalue(A(0))=(\alpha - LB\times E)+(0\times E)=\alpha - LB\times E = k$$

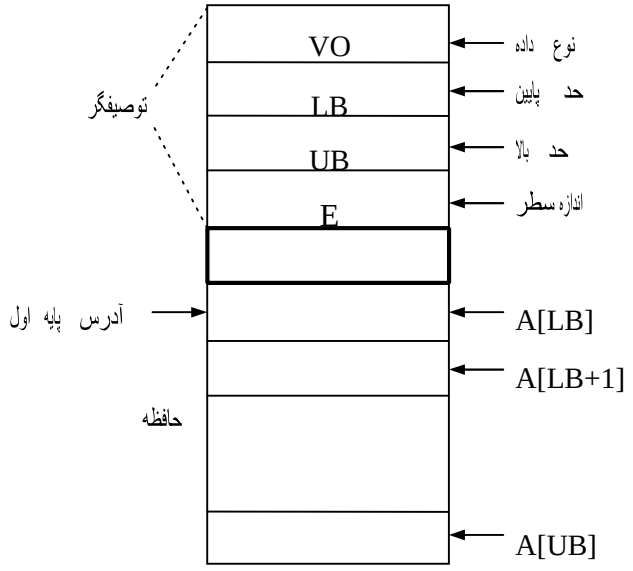
K آدرسي است كه عنصر صفر برداري در آنجا قرار مي گيرد اين آدرس مبدا مجازي (VO) نام دارد.

$$VO=\alpha - LB\times E$$

پياده سازي ديگر: اگر α آدرس شروع اولين خانه آرايه در حافظه باشد و E مقداربايتهاي لازم جهت ذخيره هريك ازعناصر آرايه باشد و LB حد پايين انديس آرايه و UB حد بالاي انديس آرايه باشد محل ذخيره عنصر i ام به شكل زير خواهد بود

آرايه تك بعدي $Ai[\,]=\alpha+(i-LB)\times E$

اما براي توليد آدرس عناصر يك ماتريس بايد ابتدا آنرا به بردار تبديل كنيم چون در حافظه كامپيوتري فضاي دو بعدي معنا ندارد اين تبديل آدرس معمولا ً به دو روش سطري[1] و روش ستوني[2] مي باشد بسياري از زبانها مانند C ، پاسكال و جاوا از روش سطري و فرترن از روش ستوني استفاده مي كنند.



شكل 6 - 4 حافظه مجزا براي ناحيههاي توصيفگر و عناصر

روش سطري براي آرايههاي دو بعدي :

فرض مي كنيم آرايه A[LB1…UB1,LB2…UB2] را داريم.فرمول $A[I_1,I_2] =\alpha + disp \times E$ را محاسبه مي كنيم كه در آن α مبدا ذخيره سازي و E اندازه هر عنصر آرايه است و disp تعداد

عناصر مابين عنصر اول آرايه تا عنصر A[I1,I2] مي باشد disp را به صورت جداگانه به روشهاي سطري و ستوني محاسبه مي كنيم

[1] RowMajor
[2] ColumnMajor

روش سطري: $disp_2 = (I_1 - LB_1) \times d_2 + (I_2 - LB)$

تعداد سطرهاي كامل قبل از عنصر مورد نظر را مي دهد.

$(I_1 - LB)_1$

. تعداد ستونها را مي دهدd_2:

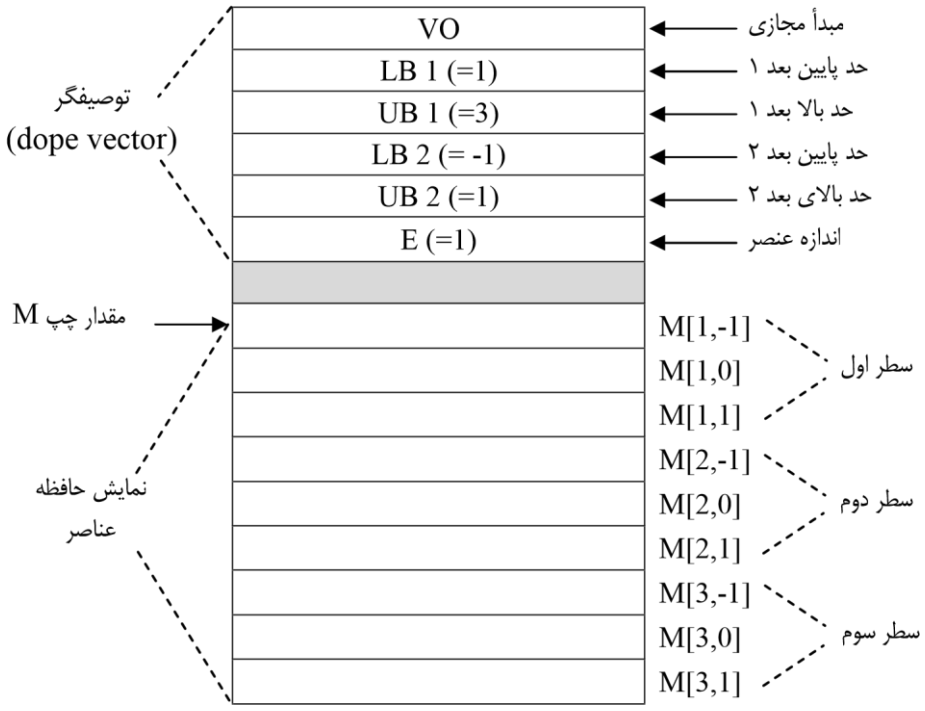
تعداد عناصر موجود در همان سطر ناقص را مي دهد. $(I_2 - LB)_2$

روش ستوني $disp = (I_2 - LB_2) \times d_1 + (I_1 - LB_1)$

. تعداد ستونهاي كامل قبل از عنصر مورد نظر را مي دهد$(I_2 - LB)_2$

. تعداد سطرها را مي دهدd_1:

. تعداد عناصر موجود در همان ستون ناقص را مي دهد$(I_1 - LB)_1$



شکل 6 - 5 نمايش حافظه براي آرايه دوبعدي

به همين روش مي توانيم آرايه سه بعدي را با فرمول $A(I_1,I_2,I_3) = \alpha + disp \times E$ محاسبه مي كنيم.

روش سطري $disp = (I_1 - LB_1)d_2d_3 + (I_2 - LB_2)d_3 + (I_3 - LB_3)$:

روش ستوني به : ${}_1disp = (I_3 - LB_3)d_2d_1 + (I_2 - LB_2)d_1 + (I_1 - LB)$

طريق مشابه مي توانيم آرايه n بعدي را با فرمول $A\,I[{}_1,I_2,I_3,...,I_n] = \alpha + disp \times E$ محاسبه ميكنيم

.

روش سطري $disp = (I_1 - LB_1)d_2d_3...d_n + (I_2 - LB_2)d_3d_4...d_n + ... + (I_n - LB_n)$:

روش ستوني : ${}_1disp = (I_n - LB_n)d_{n-1}d_{n-2}...d_1 + (I_{n-1} - LB_{n-1})d_{n-2}d_{n-3}...d_1 + (I_1 - LB)$

آدرس سطري : $\sum$ (حد پايين آن بعد انديس ×ابعاد سمت راست)

آدرس ستوني : $\sum$ (حد پايين آن بعد انديس × ابعاد سمت چپ)

مثال : آرايه چهار بعدي A[1..10][1..10][1..20][1..10] مفروض است كه به روش سطري ذخيره شده است اگر آدرس شروع حافظه صفر باشد و تعداد بايتهاي هر عنصر آرايه 2 باشد آدرس A(1,2,2,2) را حساب كنيد.

$$disp = (1-1) \times 20 \times 10 \times 10 + (2-1) \times 10 \times 10 + (2-1) \times 10 + (2-1) = 11$$

$$a[1,2,2,2] = 0 + disp \times 2 = 0 + 111 \times 2 = 222$$

روش كتاب (استفاده از مبدا مجازي):

$$Lvalue(A\ i\ j[\ ,\ ]) = \alpha + (i - LB_1) \times S + (j - LB_2) \times E$$

$$S = (UB_2 - LB_2 + 1) \times E$$

LB_2, UB حدود پايين و بالاي بعد دوم $= VO_2 = \alpha - LB_1 \times S - LB_2 \times E$

حد پايين اولين بعد $= LB_1$ $Lvalue(A(i,j)) = VO + i \times S + j \times E$

آدرس پايه $= \alpha$

طول سطر $S = (UB_2 - LB_2 + 1) \times E =$

نمايش حافظه به صورت فشرده و غير فشرده :

در نمايش حافظه فشرده عناصر يك بردار به صورت فشرده در حافظه ذخيره مي شوند و به اين نكته توجه نمي شود كه هر عنصر بايد از كلمه آدرس پذير شروع شود لذا اين روش باعث صرفه جويي در حافظه مي شود ولي دستيابي به عنصر هزينه زيادي مي برد زيرا نمي توان از فرمول دستيابي استفاده كرد به دليل گران بودن هزينه دستيابي بردارها به شكل غير فشرده ذخيره مي شوند هر عنصر در مرز يك واحد حافظه آدرس پذير قرار مي گيرد و بين هر جفت از عناصر ممكن است حافظه بدون استفاده باقي بماند. مزيت اين روش دستيابي سريع است ولي حافظه به هدر مي رود.

a1	a1		
a2	a2		
a3	a3		
a4	a4		
a5	a5		

نمايش حافظه به صورت غير فشرده

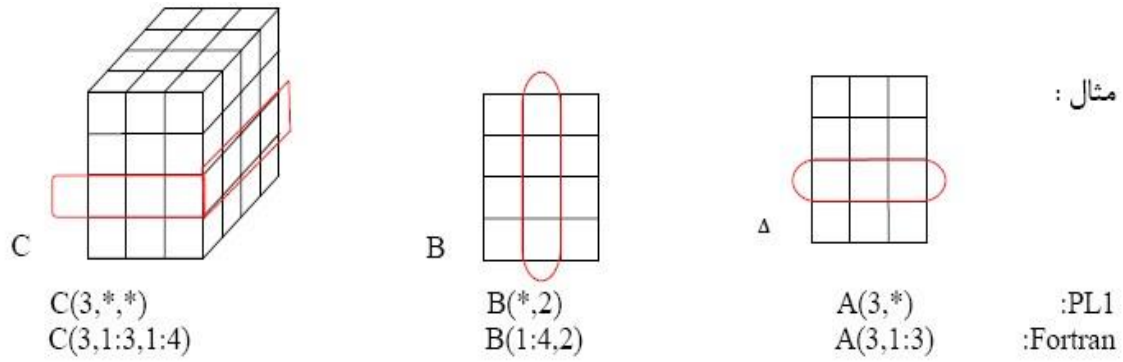
a1	a1	a2	a2
a3	a3	a4	a4
a5	a5		

نمايش حافظه به صورت فشرده

شكل 6 - 6

6-6- 1- برش[1] آرايه

برش آرايه بخشي از آرايه است كه خودش يك آرايه مي باشد. شكل زير مثالهايي از اين مفهوم را نشان مي دهد.



شكل 6 - 7

PL/I از اولين زبانهايي بود كه مفهوم برشها را پياده سازي كرد اگر اعلان A به صورت A(3,4) آنگاه برش شكل الف به صورت A(*,2) وبرش شكل ب به صورت B(3,*) برش شكل ج به صورت C(3,*,*) نمايش داده مي شود در فرترن مي توان بخشي از آرايه (برش) را به عنوان آرگومان به زير برنامهها ارسال كرد فرترن 09 برشها را به صورت زير معرفي كرد :

پياده سازي : A(1:4,2) ;B(3,1:3); C(3,1:3,1:4)

استفاده از توصيفگر منجر به پياده سازي كارآمد برشها مي شود. به عنوان مثال آرايه 4*3 را مي توان با استفاده از توصيفگر به صورت زير توصيف نمود.

---

[1] ـ

Slice

Vo	α
Lb1	1
Ub1	4
Multiplier1	3
Lb2	1
Ub2	3
Multiplier1	1

جدول 6 - 1

در اين توصيفگر ،multiplier2 كه اندازه شي ء داده (هر عنصر آرايه) است، فاصله بين عناصر متوالي آرايه را نيز نشان مي دهد. بنابراين، بخش ستون دوم آرايه x يعني A(*,2) را مي توان به صورت زير نما يش داد:

Vo	α-3
Lb1	1
Ub1	4
Multiplier1	3

جدول 6 - 2

اين توصيفگر، برداري به طول 4 را نشان مي دهد كه اولين عنصر آن يك محل پس از اولين عنصرx قرار دارد و هر عنصر در 3 محل بعدي موجود است كه با multiplier1 مشخص شده است.

6-6- 2- آرايههاي شركت پذير[1] (انجمني)

در برخي از كاربردهاي برنامه نويسي مطلوب است كه به كمك نام و بدون استفاده از انديس بتوانيم به اطلاعاتي دسترسي داشته باشيم مثلا ً اگر از طريق نام دانشجو بخواهيم به نمره ي آن برسيم يك راه پياده سازي معمولي ،استفاده از آرايه دو ستوني است كه مثلا از طريق name[i] به grade[i] برسيم راه ديگر استفاده از آرايه انجمني است كه از دو ستون key,value تشكيل شده است آرايههاي انجمني در اسنوبال 4 به صورت جدول وجود دارد و در زبانهاي فرايندي مثل پرل اهميت زيادي دارد.

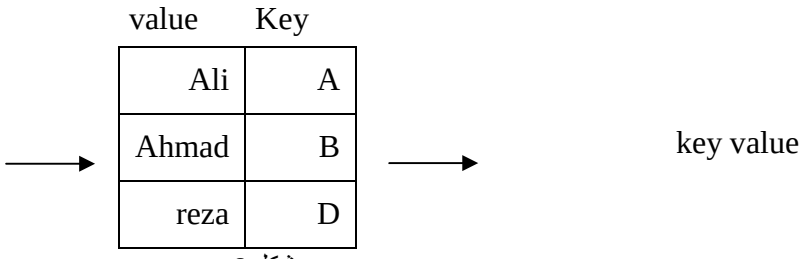
مثال: در زبان پرل آرايه انجمني به كمك عملگر% به وجود مي آيد.

۱

%ClassList = ("Ali",'A',"Ahmad",'B',"Reza",'D');

{"Ali"}$ClassListاست "A " نحوه دستيابي : داراي مقدار

Associative arrays



شكل 6 -

8 آرايههاي شركت پذير توسط توصيفگر پياده سازي مي شوند.

6-7- ركوردها

مشخصات:

ركورد ساختاري متشكل از تعداد ثابتي عنصر با نوعهاي متفاوت مي باشد ركوردها و بردارها ساختمان دادههايي خطي با طول ثابت هستند ولي ركوردها از دو جنبه با بردارها فرق دارند الف - عناصر ركورد ممكن است ناهمگن و متفاوت باشند. ب - عناصر ركورد داراي نام هستند.

مثال: نحوه تعريف ركورد در زبان C كه به آن ساختمان[1] گفته مي شود.

```
Struct employee{
int ID;        int
 Age;        float
          Salary;
             }A;
```

A متغييري از نوع Employee است و نحوه دستيابي به عناصر مانند A.ID=125است صفات اين ركورد عبارتنداز: 1 -تعداد عناصر2-نوع هرعنصر3-نام گذاري هرعنصر

[1]

عناصر ركورد را معمولا ًفًيلدها تشكيل مي دهد ركوردها درزبان Cساختمان ناميده مي شوند.
انتخاب عنصر،مهمترين عمل دررکورد است مثل انتخاب A.Ageاين عمل مثل انديس گذاري درآرايه است ولي با يك تفاوت. انديس در ركورد،نام يك عنصر است.
عملياتي كه برروي كل ركورد انجام مي شود اندك اند. معمولا ركوردهايي با يك ساختاررا مي توان به يكديگر نسبت داد.

```
Struct employee type inputrec;
                                .
                                .
                                .
employee = inputrec;
```

struct

دراينجا صفت inputrec مانند employee است. تناظر اسامي بين ركوردها نيز مبنايي براي انتساب در كوبول وPl/I است مثلا دركوبول توسط دستور MOVE COPRESPANDING مي توان دو ركورد را به يكديگر انتساب داد به طوريكه اجزاي متناظر به يكديگر انتساب داده شوند.اسامي متناظر در دو ركورد بايد همنام وهم نوع باشند ولي لازم نيست ترتيب آنها يكسان باشد.

```
MOVE COPRESPANDING  inputrec  TO  employee
```

عمليات انتساب كامل يك ركورد به ركورد ديگر به اين صورت پياده سازي مي شود كه محتويات بلوك حافظه از ركورد اول در بلوك حافظه ركورد دوم كپي مي شود.

پياده سازي:

براي پياده سازي ركورد از يك بلوك حافظه كه عناصر در آن به ترتيب ذخيره مي شوند استفاده مي گردد ممكن است براي هرعنصر از توصيفگري استفاده شود ولي اغلب براي كل ركورد نياز به توصيفگر نيست فرمول دستيابي براي محاسبه ي آدرس محلi امين عنصر به صورت زير است.

$$\text{آدرس عنصر i ام ركورد} = \alpha + \sum_{j=1}^{i-1} (\text{اندازه فيلد } j)$$

α_jآدرس پايه بلوك حافظه اي است كه R را نشان مي دهد (R . j ام)
ركوردها و آرايههايي با عناصر ساختاري : (آرايه اي از ركوردها - ركوردهاي تودرتو)

۱

آرايه اي از ركوردها : مي توان از برداري استفاده كرد كه هر يك از عناصرش ركورد باشد مثلاً در زبان C:

```
Struct employee type
        { int    ID;
          int    Age;
    float   salary;
       char   Dept;
  } employee[500];
```

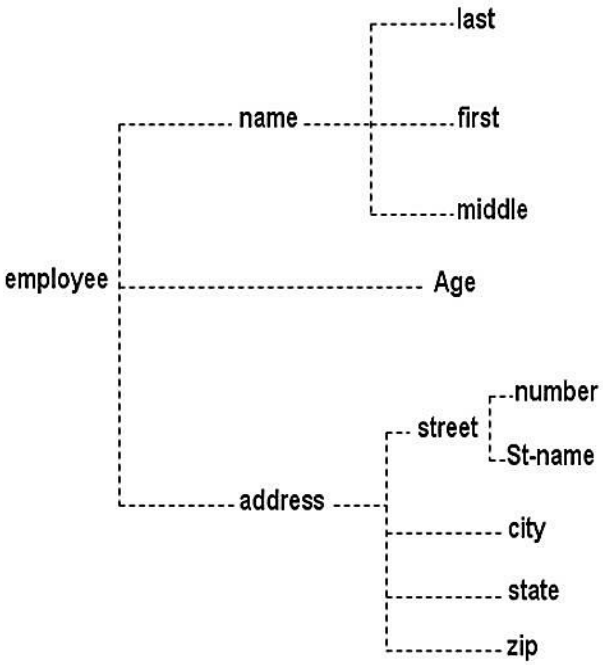
اين دستور برداري 005 عنصري تعريف مي كند كه هر يك از عناصر آن يك ركورد employeetype است عنصر از ساختمان داده مركب توسط دنباله اي از عمليات انتخاب مي شود.مثل employee[3].salary

ركورد 153153153**تودرتو**F153[1] : عناصر ركورد مي تواند آرايه و يا ركورد ديگري باشد و اين روند مي تواند تا چندين سطح ادامه داشته باشد ويك ساختار سلسله مراتبي را ايجاد كند در كوبول و PL/I اين سازمان سلسله مراتبي از نظر نحوي با شمارههايي مشخص مي شود.

Nested record

[1]

```
1  employee رکورد
2  Name رکورد
3  last char(10)
3 first char(15)
3 middle char(1)
2 Age fixed(2) عنصراوليه
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)
```



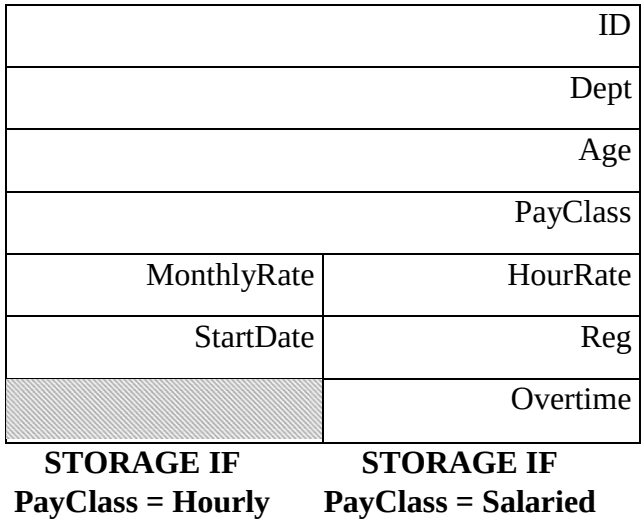
شكل 6 - 9

ركورد با طول متغيير : فرض كنيد مي خواهيم مشخصات كاركناني كه ماهيانه حقوق ميگيرند را همراه كاركناني كه ساعتي كار ميكنند ذخيره كنيم براي اين كار مي توان از مفهوم ركورد با طول متغيير استفاده كرد مثلا ً در پاسكال:

```
type PayType=(Salaried, Hourly); var
Employee:record
```

```
ID  : integer;    Dept: array[1..3] of
 case PayClass : Age : integer;     char;
                                  PayType of
      Salaried:(MonthlyRate:real; StartDate  :integer);
Hourly:(HourRate:real; Reg :integer;Overtime:integer); end
```

عنصر payclass در پاسكال برچسب[1] ودر زبان Ada متمايز كننده ناميده مي شود 3 فيلد ID و Age و Dept براي تمام ركوردها ثابت است درادامه اگر salaried=payclass ركورد دو فيلد MonthlyRate و startdate دارد واگر Hourly=payclass باشد ركورد شامل سه فيلد HourRate و Reg و overtime است نحوه پياده سازي ركوردمتغير فوق به شكل زير است يعني حافظه مورد نيازركورد به اندازه بخش ثابت (قسمت بالايcase) + بزرگترين قسمت بخش زيرين case درنظر گرفته مي شود.



شكل 6 - 01

درحين ترجمه، ميزان حافظه مورد نياز براي عناصرهردو شكل ازركورد تعيين مي شود وحافظه به اندازه بزرگترين شكل ممكن تخصيص مي يابد.شكل كوچكتر نمي تواند ازكل فضااستفاده كند درحين ترجمه نيازي به توصيف گر خاصي براي ركورد با طول متغيير نيست زيراخود عنصربرچسب به عنوان عنصري ديگرازركورد درنظر گرفته مي شود .

در ركوردهاي معمولي تمام فيلدهاي آن در طول عمر ركورد وجود دارند ولي در مورد ركوردهايي با طول متغير ، برخي فيلدهاي آن ، زماني وجود دارند و زماني ديگر وجود ندارند. در مثال فوق فيلد employee.reg در زماني از اجراي برنامه ممكن است وجود نداشته باشند لذا اين موضوع بايد كنترل شود كه دو روش براي اينكار وجود دارد.

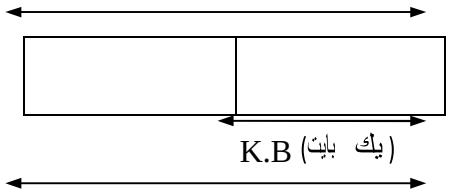
- **كنترل پويا :** در اين شيوه عنصر برچسب (متغير جلوي case) قبل از دستيابي به يك عنصر كنترل مي شود اگر اين برچسب مقدار مناسب داشت يعني آن عنصر قابل دستيابي است و در غير اين صورت يك خطاي زمان اجرا ، رخ خواهد داد.
- **كنترلي انجام نشود :** در اين روش در تعريف ركورد طول متغير از عنصر برچسب استفاده نمي شود و بنابراين در زمان اجرا كنترلي صورت نمي گيرد وبه عبارتي ديگر انتخاب عنصر از اين ركورد همواره معتبر در نظر گرفته مي شود دراين حال اگر عنصر مورد نظر وجود نداشت خطاي منطقي رخ مي دهد كوبول ،

---

[1] Tag

PL/I و پاسكال شكلهايي از ركورد طول متغيير بدون برچسب را تدارك مي بينند و در C نيز از Union استفاده مي شود كه فاقد برچسب است در پياده سازي اين گونهها كنترلي انجام نمي شود **مثال :**

```
int A;    union{
char B;  sizeof(int)=2
          sizeof(char)=1
              } k;
```



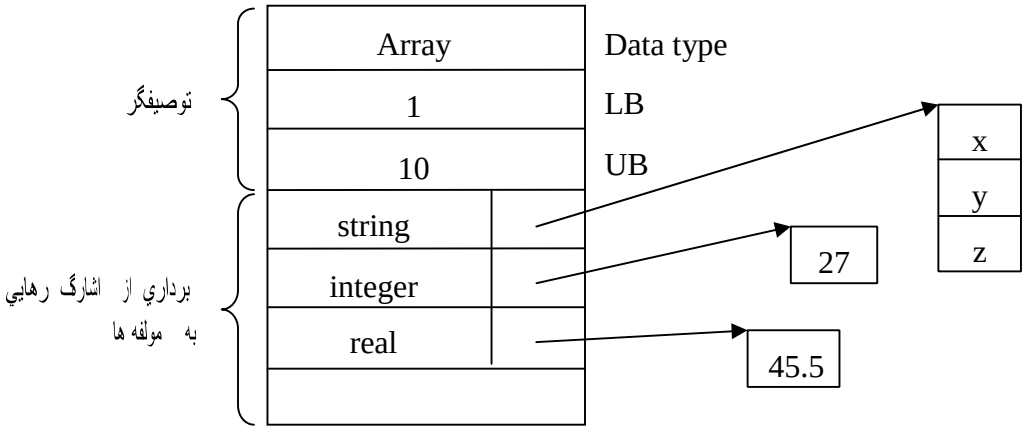
متغير K (دوبايت)

شكل 6 - 11 در مفهوم union براي A,B از يك فضاي مشترك استفاده مي شود.

نوع ركورد متغير را Union نيز مي نامند اگر از فيلد برچسب استفاده نشود (همانند نوع Union در C مثال بالا) نوع را Union آزاد مي گويند ولي اگر از فيلد برچسب استفاده شود نوع Unoin را قابل تشخيص مي نامند (چون با بررسي برچسب مي توان كلاسي را كه شي داده به آن تعلق دارد مشخص كرد) **ركوردهاي بدون تعريف در زبانهاي برنامه نويسي**

در بعضي از زبانهاي برنامه نويسي مانند Lisp,Snobol4 ركوردها از پيش تعيين نمي شوند و درحين اجراي برنامه توليد خواهند شد.

```
Snobol4:
A:array (10) ;
A(1)="xyz";
...
A(2)=27;
...
A(3)=45.5;
```



شكل 6 - 21

با توجه به شكل فوق از ديد پياده سازي، اشاره گرها به نحو موثري براي اين هدف بكار گرفته شده اند. همچنين در شكل ديده مي شود كه به ازاي هر مولفه، نوع مولفه و اشاره گري به ارزش مولفه ذخيره مي شود كه معمولاً ارزش -ها در بلاكهاي ديگري از حافظه قرار مي گيرند و توسط روتينهاي مديريت حافظه بكار گرفته مي شوند.

6-8- ليستها

مشخصات:

ليست دنباله اي از ساختمان دادهها است. يك ليست مشابه با بردار شامل دنباله مرتبي از اشياء مي باشد يعني مي توان به اولين عنصر ، دومين عنصر و ... دستيابي داشت تفاوت ليستها با بردار عبارت اند از : الف) طول ليست اغلب ثابت نيست و در طول اجراي برنامه كم و زياد مي شود ولي طول آرايه اغلب ثابت است ب) عناصر ليست اغلب ناهمگن هستند درحاليكه عناصر آرايه همگن هستند.

نحوه زبان ليسپ ، ليست را به صورت زير نمايش مي دهد :

```
Function Name  ( Data1 Data2 …Datan)
```

ليسپ با اعمال Function Name برروي آرگومانهاي $Data_n$… $Data_2$ $Data_1$ اجرا مي شود

اغلب عمليات درليسپ ، آرگومانهاي ليست را گرفته ، مقادير ليست را باز مي گرداند به عنوان مثال عمل cons دو آرگومان ليست را مي گيرد وليستي را برمي گرداند كه آرگومان اول آن به ابتداي آرگومان دوم اضافه مي شود.

ليستي شامل 4 عنصر به عنوان خروجي مي باشد كه عنصر اول ، ليست (a,b,c) و بقيه از نوع اتم هستند.

```
Cons '(a b c)'(d e f)) = ((a b c)d e f)
```

نحوه ليست در ML به صورت [a,b,c] است ليستها در ML همگن هستند يعني مي توان ليستي از مقادير صحيح مثل 3[,1,2] يا ليستي از رشتهها ["abc","def"] داشت.

پياده سازي :

براي پياده سازي ليستها از نمايش حافظه پيوندي استفاده مي شود يك قلم ليست ، يك عنصر اوليه است كهمعمولا ً شامل شي داده اي با اندازه ثابت است در ليسپ به سه فيلد از اطلاعات نياز داريم يك فيلد نوع و دو فيلد اشاره گر ليست.

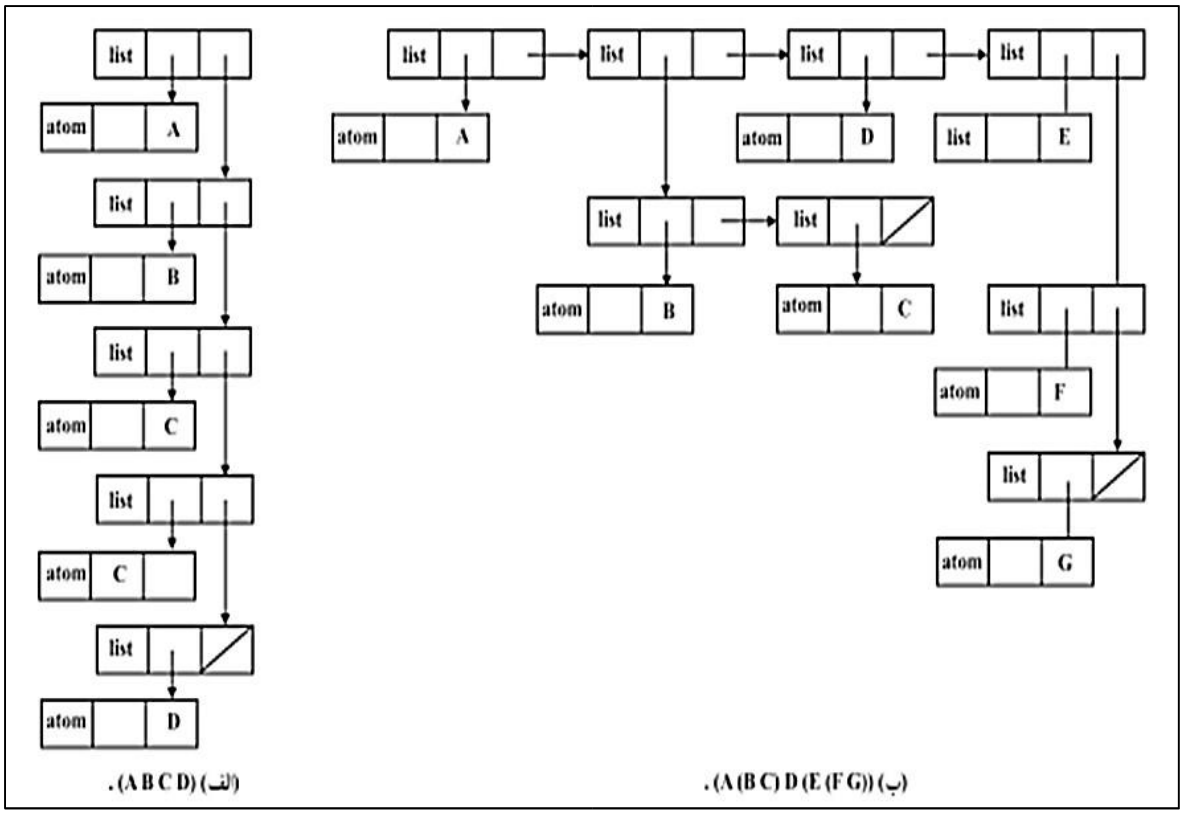
فرم كلي هر گره

شكل 6 - 31

اگر فيلد نوع اتم باشد آنگاه فيلدهاي باقي مانده توصيفگرهايي هستند كه اين اتم را توصيف مي كنند.

اگرفيلد نوع يك ليست باشد اشاره گر اول راس ليست[1] (اولين عنصر ليست) در حاليكه اشاره گر دوم انتهاي ليست[2] (بقيه اعضا)مي باشد.

ليستها در زبانهايي مثل ML ، ليسپ و پرولوگ به عنوان اشياء داده اوليه هستند اما در زبانهاي كامپايلري مانند C و Ada و پاسكال اينگونه نيست در زبانهاي كامپايلري براي مديريت ليستها، مديريت حافظه پويا لازم است و اين نوع داده در اين زبانها توسط اشاره گر به وسيله برنامه نويس تعريف مي شود



شكل 6 - 14

head [1]
tail [2]

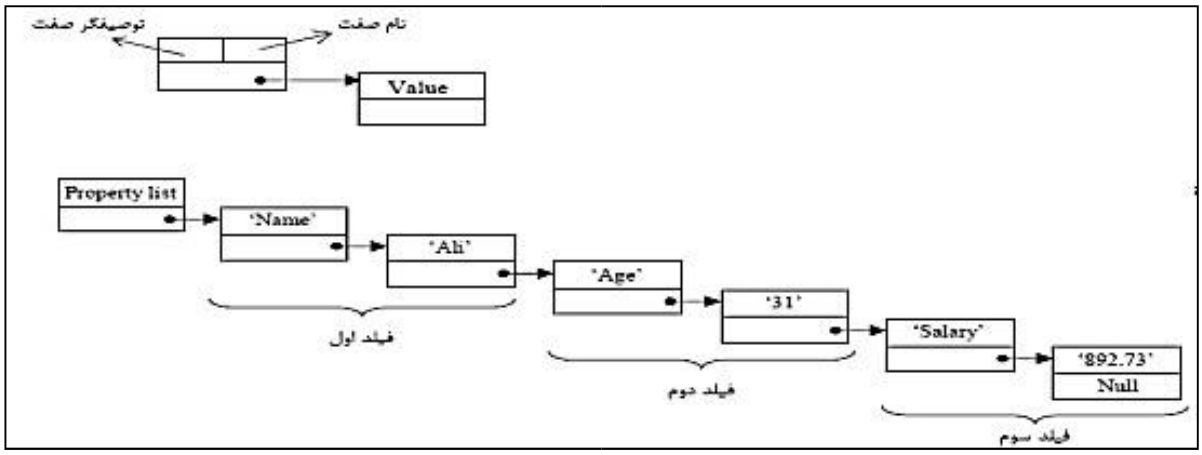
شكلهاي گوناگون ليستها: در برخي از زبانها شكلهاي مختلفي از ليستها وجود دارد مثل پشتهها و صفها، درختها ، گراف جهت دار ، ليستهاي خاصيت.

- **پشتهها و صفها** : پشته ليستي است كه انتخاب ، درج و حذف عناصر از انتهاي آن انجام مي شود. صف ليستي است كه انتخاب و حذف از يك طرف و درج از طرف ديگر انجام مي شود پشته زمان اجرا يك شي داده مهم است كه توسط سيستم تعريف مي شود. صفها در زمان بندي و همزمان سازي زير برنامهها كاربرد دارند.
- **درختها:** ليستي كه عناصرش علاوه بر اشيا داده اوليه ممكن است ليست باشد درخت نام دارد به شرطي كه هر ليست فقط يك عنصر از اولين ليست باشد. درختها براي نمايش جدول نمادها در كامپايلر به كار مي روند.

- **گراف جهت دار** : ساختمان داده اي كه عناصر آن با استفاده از الگوي پيوندي خاصي به هم پيوند داده مي شوند (به جاي دنباله خطي از عناصر) گراف جهت دار ناميده مي شود.
- **ليست خاصيت :** ركوردي است كه تعداد عناصر آن بدون هيچ محدوديتي تغيير مي كند. توجه داشته باشيد كه اين نوع ركورد با ركورد طول متغيرفرق دارد. در ركورد طول متغيير ، ركوردها فقط به شكلهايي مي تواند در آيد كه از قبل تعيين شده اند. در ليست خاصيت اسامي عناصر (فيلدها) و مقادير آنها بايد ذخيره شوند نام فيلد نام خاصيت ناميده مي شود و مقدار متناظر فيلد ، مقدارخاصيت ناميده مي شود وقتي خاصيت جديدي در ليست درج مي شود دوعنصر درج مي شوند : نام خاصيت و مقدارخاصيت.

ليست توصيفي و ليست خاصيت – مقداري نامهاي ديگربراي ليست خاصيت هستند.راه حل معمول پياده سازي

ليست خاصيت،استفاده از يك ليست پيوندي است.كه اسامي فيلدها و مقاديرآن پشت سرهم مي آيند.



شكل 6 - 51

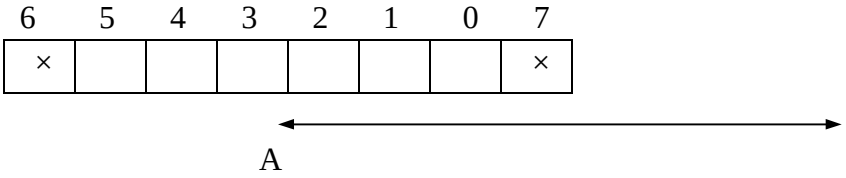
6-9- مجموعهها[1]

مجموعه،شيء داده اي است كه شامل مقادير نامرتب و مجزا است درحاليكه ليست شامل تعدادي مقادير مرتب استكه عناصر آن مي توانند تكراري باشند عمليات روي مجموعهها عبارتند از:

- **عضويت :** آيا مقدار x عضوي از مجموعه S است ؟
- **درج و حذف يك مقدار** : اگر فعلاً $x \notin S$ باشد مي توان آنرا در S درج كرد چنانچه $x \in S$ باشد مي توان آنرا از مجموعه S حذف كرد
- **اجتماع ، اشتراك و تفاضل مجموعهها :** $S_1 \cup S_2, S_1 \cap S_2, S_1 - S_2 = \overline{S_1} \cap S$ [2] **پياده سازي :** دو روش براي پياده سازي مجموعهها وجود دارد:

الف- نمايش بيتي : نمايش حافظه بيتي در هنگامي مفيد است كه اندازه مقادير مجموعه جهاني (مرجع) كوچك باشد در اين روش هر بيت نمايانگر وجود يا عدم وجود يك عنصر در مجموعه است مثال :

```
For A=set of 1..6 ;
```



شكل 6 - 16

حال اگر در A مجموعه 5[2,3,] با دستور A=[2,3,5] ذخيره گردد حافظه به شكل زير در مي ايد. در اين مثال 1..6 مجموعه مرجع است.

0	1	2	3	4	5	6	7
×	0	1	1	0	1	0	×

شكل 6 - 17

در اين روش براي درج يك عنصر در مجموعه بايد بيت مناسبي را به يك تبديل كرد و براي حذف يك عنصر بايد بيت مناسبي به صفر تبديل شود. عضويت را مي توان با بررسي بيت مناسب انجام داد عمليات اجتماع ، اشتراك و تفاضل را با عمليات بولي كه در سخت افزار وجود دارد پياده سازي مي كنيم عمليات OR بر روي دو بيت نشان دهنده اجتماع ، عمليات AND نشان دهنده اشتراك و AND رشته اول با مكمل رشته دوم ، عمليات تفاضل را مشخص مي كند به دليل پشتيباني سخت افزار از عمليات بيتي نمايش بيتي مجموعهها كارامد است اما عمليات سخت افزار معمولاً بر روي بيتهايي با طول ثابت انجام مي گيرد (مثل طول كلمه حافظه) براي رشتههاي طولاني

---

[1] Set

تر بايد به وسيله شبيه سازي نرم افزار به واحدهاي كوچكتر تبديل گردد تا قابل پردازش باشد.پاسكال از اين روشاستفاده مي كند.

ب- درهم سازي مجموعهها: نمايش ديگري براي مجموعهها براساس تكنيك درهم سازي[1] يا حافظه پراكنده[2] است از اين روش هنگامي استفاده مي شود كه مجموعه جهاني بزرگ باشد (مثل مجموعه شامل اعداد و كاراكترها) .اغلب زبانهاي برنامه نويسي اين روش را براي مجموعهها فراهم نمي

كنند اما پياده سازي زبان از اين نمايش براي دادههاي تعريف شده توسط سيستم استفاده مي كند كه در حين ترجمه يا اجرا به آن نياز است تقريبا ً هر كامپايلر با استفاده از روش درهم سازي ، اسامي را در جدول نمادها درج مي كند.

160160160160**برخورد**[F3]**:** دو داده مختلف ممكن است يك آدرس درهم سازي توليد كنند در اين صورت برخورد به وجود خواهد آمد تكنيكهاي مقابله با برخورد عبارتند از :

- **درهم سازي مجدد :** مي توانيم رشته بيتي B_x (عنصر جديد x) اصلاح كنيم ونتيجه را دوباره درهم سازي كنيم تا آدرس درهم سازي جديد ايجاد شود اگر برخوردي به وجود بيايد دو بار اين كار را تكرار ميكنيم منظور از اصلاح كردن اين است كه مثلا ً رشته بيتي را در عددي ضرب يا با عددي جمع كنيم.
- **پيمايش ترتيبي:** جستجو رااز نقطه برخورد در بلوك شروع مي كنيم تا B_x با يك محل خالي در بلوك پيدا شود.
- **باكت بندي :** ميتوانيم اشاره گرهايي را در بلوك در نظر بگيريم كه به ليستهاي باكت پيوندي اشاره كنند كه حاوي عناصر با آدرسهاي درهم سازي يكسان هستند پس از هر درهم سازي B_x و بازيابي اشاره گر به ليست باكت مناسب ، آن ليست را براي B_x جستجو مي كنيم چنانچه پيدا نشد آنرا به انتهاي ليست اضافه مي كنيم.

6-01- اشياء داده اجرايي

در اكثر زبانها ، مخصوصاً زبانهاي كامپايلري مثل C ، پاسكال و Ada ، برنامههاي اجرايي و دادهها، ساختارهاي مجزايي دارند ولي اين موضوع الزامينيست به عنوان مثال، در زبان ليسپ و پرولوگ ، دستورات اجرايي ميتوانند خود، دادههايي باشند كه توسط برنامهها قابل دستيابي و دستكاري هستند مثلاً زبان ليسپ تمام دادههاي خود را در ليستها ذخيره ميكند .مانند دستور زير كه تابع f_n را توسط دستور defineتعريف ميكند :

```
(Define fn (cons(a b c) (d e f)))
```

Cons دو ليست را به هم ملحق ميكند .در پرولوگ عمليات consult وجود دارد.

[1] hashing
[2] scatter storage
[3] collision

11-6- نوع داده انتزاعي[1] (ADT)

در زبانهاي اوليه نظير كوبول و فرترن ، نوع جديد تنها به زيربرنامهها محدود بود ولي در زبانهاي بعدي امكاناتبهتري جهت پياده سازي نوع داده انتزاعي (ADT) نظير Package در Ada و كلاس در زبان C++ ارائه گرديد.

انتزاع دادهها:

براي بسط مفهوم بسته بندي به دادههايي كه توسط برنامه نويس تعريف ميشوند، نوع داده انتزاعي به صورت زير تعريف ميشود :

- مجموعه اي از اشياي داده معمولا با استفاده از يك يا چند تعريف نوع
- مجموعه اي از عمليات انتزاعي بر روي انواع داده
- بسته بندي تمام آنها ، به طوري كه كاربر نوع جديد نتواند اشياء داده از آن نوع را ، به جز ازطريق عملياتي كه براي آن تعريف شده است ، دستكاري كند.

كل تعريف بايد طوري بسته بندي شود كه كاربر فقط با دانستن نام نوع و معناي عمليات آن ، بتواند آن را به كار گيرد .به عنوان مثال براي انواع اوليه مثل حقيقي و صحيح ، زبان برنامه سازي ، امكاني را براي اعلان متغيرهاي آن نوع و عملياتي را براي آنها تدارك ميبيند .نمايش حافظه مربوط به مقادير صحيح و حقيقي كاملاً بسته بندي شده است يعني از ديد برنامه نويس پنهان است. برنامه نويس بدون اينكه از جزئيات نمايش حافظه اين انواع اطلاع داشته باشد از اشياي داده آنها استفاده ميكند. برنامه نويس فقط نام نوع و عمليات براي آن نوع را فراهم ميكند .

پنهان سازي 162162162162اطلاعاتF2:

براي نوشتن برنامههاي بزرگ بايد از استراتژي " تقسيم و حل " استفاده كرد. در اين استراتژي، برنامه به مجموعه اي از قطعات به نام ماژول[3] تقسيم ميشود. هر ماژول مجموعه محدودي از عمليات را بر روي مقدار محدودي از دادهها انجام ميدهد .طراحي ماژول معمولا به دو صورت انجام ميشود : 1 - تجزيه تابعي ، 2 – تجزيه داده اي. در تجزيه تابعي، ماژولها تجزيه تابعي برنامه را نشان ميدهند ، كه ساختارهاي رويهها ، توابع ، زيربرنامهها از آن نتيجه ميشود .

به عنوان مثال براي طراحي برنامه ثبت نام به روش تجزيه تابعي ،برنامه به واحدهاي تابعي تجزيه مي شود كه يك برنامه نويس ممكن است توابعي را براي حذف و اضافه دروس ، توابع ثبت نام و... ايجاد كند كه براي ساختن هر قطعه به اطلاعات اندكي نياز است ، كه اين عيب (نياز به داشتن

اطلاعات) توسط تجزيه به روش داده اي كه همان انتزاع ساده نام دارد از بين مي رود. براي طراحي برنامه ثبت نام به روش انتزاع ساده ، بايد نوع داده بخش (section) را ايجاد و از آن در ماژولهاي ديگر استفاده كنيم.

---

[1] Abstract Data Type

[2] Information hiding

[3] module

روشهاي طراحي برنامه مثل اصلاح مرحله اي ، برنامه نويسي ساخت يافته ، برنامه نويسي پيمانه اي و برنامه نويسيبالا به پايين با طراحي انتزاع سر و كار دارد.

زبان برنامه سازي ، انتزاع را به دو روش پشتيباني ميكند :

- با تدارك كامپيوتر مجازي كه كاربرد آن ساده تر و قدرت آن بيش از كامپيوتر سخت افزاري است ، مجموعه مفيدي از انتزاعها را تدارك ميبيند .
- زبان امكاناتي را فراهم ميكند كه برنامه نويس ميتواند انتزاعها را به وجود آورد. زيربرنامهها ، تعاريف نوع

، كلاسها ، پكيجها و توابع كتابخانه اي ، بعضي از امكاناتي هستند كه در زبانهاي مختلف براي پشتيباني از انتزاعهاي برنامه نويسي وجود دارد.

در صورتيكه يك زيربرنامه از زيربرنامه ديگري استفاده كند و به جزئيات زيربرنامه استفاده شونده دستيابي نداشته باشد ايده پنهان سازي اطلاعات[1] پياده سازي شده است.ايده پنهان سازي اطلاعات، براي استفاده داده نيز معتبر است. مثلاً در مورد تابع sqrt() جزئيات الگوريتم جذرگرفتن و نحوه نمايش اطلاعات از ديد كاربر پنهان است. به طور مشابه نوع داده تعريف شده توسط كاربر در صورتي يك انتزاع موفق است كه بدون اطلاع از نمايش اشيايي از آن نوع يا الگوريتمهايي كه توسط عمليات آن استفاده ميشوند به كار گرفته ميشوند در صورتيكه اطلاعات در يك انتزاع بسته بندي شدند معنايش اين است كه :

الف -كاربر جهت استفاده از انتزاع لازم نيست از اطلاعات مخفي اطلاع داشته باشد.

ب - كاربر نميتواند مستقيماً اطلاعات مخفي را دستكاري كند.

به عنوان مثال نوع داده صحيح در يك زبان برنامه نويسي مثل فرترن يا پاسكال، نه تنها جزئيات نمايش عدد صحيح را پنهان ميكند بلكه اين نمايش را طوري بسته بندي ميكند كه برنامه نويس نميتواند هر يك از بيتهاي نمايش يك مقدار صحيح را دستكاري كند.

نكته: بسته بندي[2] ، اصلاح برنامه را نيز آسان ميكند .پنهان سازي اطلاعات به طراحي برنامه مربوط ميشود و هر برنامه اي كه به خوبي طراحي شده باشد صرفنظر از زبان مورد استفاده، پنهان سازي اطلاعات امكان پذير است. بسته بندي به طراحي زبان مربوط ميشود. يك انتزاع وقتي خوب بسته بندي ميشود كه زبانها ،دستيابي به اطلاعات مخفي شده در آن انتزاع را مجاز ندانند .

21-6- زيربرنامهها

زيربرنامه يك عمليات انتزاعي است كه توسط برنامه نويس تعريف ميشود .دو ديدگاه از زيربرنامه در اينجا مهم است.

در سطح طراحي برنامه ، روش نمايش عمليات انتزاعي است كه برنامه نويس تعريف ميكند در مقايسه با عمليات اوليه موجود در زبان.

---

[1] Information hiding

[2] Encapsolation

در سطح طراحي زبان ، طراحي و پياده سازي امكاناتي است كه براي تعريف و فراخواني زيربرنامه تهيه ميشود .

مشخصات زيربرنامه :

- نام زيربرنامه
- امضاي زير برنامه : كه تعداد آرگومانها ، ترتيب و نوع هر كدام از آنها و تعداد نتايج و ترتيب و نوع آنها بايد مشخص باشد.
- عملياتي كه توسط زيربرنامه انجام ميشود.

زيربرنامه ، نمايانگر يك تابع رياضي است كه مجموعه اي از آرگومانها را به مجموعه اي خاص از نتايج نگاشت ميكند .مثال:

```
Float Fn(float x, int y) Fn
    : Real * Integer  Real
```

در بعضي از زبانها براي اعلان زيربرنامهها ، از كلمات كليدي Function , procedure استفاده مي شود.

مثال :تعريف تابع در زبان پاسكال

```
Function Fn (x:Real , y:Integer):Real;
```

11رويه[1]1F1 (زيرروال) : اگر زيربرنامه بيش از يك مقدار را برگرداند يا آرگومانهاي خود را تغيير دهد رويه يا زيرروال ناميده ميشود .

```
Void sub ( float x, int y, float *z, int *w)
```

درزبان C, Void نشان ميدهد كه تابع مقداري را برنميگرداند. نام پارامترمجازي كه با * مشخص شده است نشاندهنده پارامتري است كه تغييرات آن در زيربرنامه ، در برنامه فراخوان قابل دستيابي است. نحو اين مثال در زبان Ada به شكل زير است :

```
Procedure sub(x:in real;y:in integer;z:in out real;w:out Boolean)
```

امضاء **: sub : real1* integer * real2 → real3 * bool**

برچسبهاي in و out و in out سه روش براي ارسال آرگومانها به زيربرنامه را نشان ميدهد .

in: آرگومانی را مشخص ميكند كه توسط زيربرنامه تغيير نميكند .

in out : آرگومانی را مشخص ميكند كه ميتواند اصلاح شود.

out : نتايج خروجي را مشخص ميكند.

با اينكه زيربرنامه يك تابع رياضي را نشان ميدهد اما توصيف دقيق آن با مشكلاتي روبروست كه عبارتند از :

- زيربرنامه ممكن است آرگومانهاي ضمني به شكل متغيرهاي غيرمحلي داشته باشد.

---

[1] Procedure

- زيربرنامه ممكن است اثرات جانبي (نتايج ضمني) داشته باشد بطوريكه متغيرهاي غيرمحلي ياآرگومانهايinout را تغيير دهد.
- زيربرنامه ممكن است به ازاي بعضي از آرگومانها تعريف نشده باشد و اگراين آرگومانها به آن ارسال شوند به طور كامل اجرا نميشوند و كنترل به برنامه استثنا ميرود.
- زيربرنامه ممكن است به گذشته حساس باشد يعني نتايج آن به آرگومانهايي بستگي داشته باشد كه درفراخواني قبلي به آن ارسال شده باشد. شايد دليل آن نگهداري متغيرهاي محلي درحين اجراهاي مختلف باشد.

نكته : دربعضي از زبانها مثل پاسكال ، Ada ولي نه در C ، بدنه زيربرنامه ميتواند شامل تعريف زيربرنامههاي ديگري باشد كه درآن زيربرنامه بزرگتر قابل استفاده است. اين زيربرنامههاي محلي طوري بسته بندي ميشوند كه نميتوانند درخارج زيربرنامه حاوي آن،فراخواني شوند.

6-21-1- تعريف و فراخواني (فعالسازي) زيربرنامه

تعريف يك زيربرنامه ، خاصيت ايستاي آن است. در هر بار صدا زدن زيربرنامه ، حين اجراي برنامه ، يك ركورد(سابقه) فعاليتي[1] از زيربرنامه، پديد ميآيد .پس از خاتمه يافتن زيربرنامه اين سابقه فعاليت از بين ميرود .اگر فراخواني ديگري صورت گيرد سابقه فعاليت جديدي ايجاد ميشود .ممكن است چندين سابقه فعاليت ازيك زيربرنامه دربرنامه وجود داشته باشد. در واقع تعريف زيربرنامه ، يك قالب[2] براي ايجاد سابقههاي فعاليت آن درحين اجرا ميباشد. اين تمايز شبيه مفهوم كلاس و شي است. در واقع سابقه فعاليت يك زيربرنامه ، نوعي شي داده اي است كه در بلوكي از حافظه نشان داده شده و شامل عناصر مرتبط با آن است.

تعريف زيربرنامه :

تعريف چيزي است كه در برنامه نوشته ميشود و تنها اطلاعاتي است كه در زمان ترجمه وجود دارد يعني نوع متغيرهاي زيربرنامه مشخص است ولي مقادير (مقدار راست) يا محل آن (مقدار چپ) مشخص نيست.

فعاليت زيربرنامه :

فقط درحين اجراي برنامه وجود دارد. در حين اجرا كد مربوط به دستيابي به مقدار راست يا مقدار چپ متغير ميتواند اجرا شود. اما نوع متغير ممكن است وجود نداشته باشد مگر اينكه مترجم اطلاعات را در توصيفگر متغير ذخيره كرده باشد.

فعاليت زيربرنامه داراي طول عمراست كه ازفراخواني زيربرنامه شروع ميشود و تا از بين رفتن آن ادامه دارد. به مثال زير توجه كنيد :

---

[1] Activation Record

[2] Template

```
Float FN(float x,int y)
{
    const intval=2;
    #define finalval
          float     10;
       int N;   M(10);
          N=intval;
                 if
           (n<finalval)
                  {
              …
         return    }
```

رکورد فعاليت FN (الگو) سگمنت کد زيربرنامه FN

```
(20*x+M(N))
```

شکل 6 - 81

- خط امضاي FN اطلاعاتي را براي حافظه پارامترها (xوy) وحافظه لازم براي نتيجه تابع ارائه ميكند نتيجه ، شيئ داده اي از نوع float است.
- اعلانهايي وجود دارند كه حافظه را براي متغيرهاي محلي (آرايه M و متغير N) آماده ميكنند.
- حافظه مربوط به ليترالها وثوابت تعريف شده ، initval ثابتي با مقدار 2 و finval ثابتي با مقدار 01 مي -باشد. 2 و 10 ليترال ميباشند.
- حافظه لازم براي كد اجرايي كه از دستورات بدنه زيربرنامه توليد ميشود .

به يكي از خواص مهم C توجه كنيد. صفت const به كامپايلر C اطلاع ميدهد كه شيئ داده initval داراي مقدار ليترال 2 است. دستور define# يك دستور پيش پردازنده (ماكرو) است كه به جاي هر finval كاراكترهاي "10" را قرار ميدهد .مترجم C نام finval را پردازش نميكند .اثرات عملي هر دو دستوراز زيربرنامه اجرايي يكسان است اما معناي آنها كاملاً متفاوت است. initval داراي يك مقدار چپ است كه مقدار راستش 2 ميباشد در حالي كه finval فقط داراي مقدار راست 01 است.

هر زير برنامه دو فضا دارد : 1 - بخش پويا (ركورد فعاليت) 2 - بخش ايستا (سگمنت كد) **بخش ايستا :**

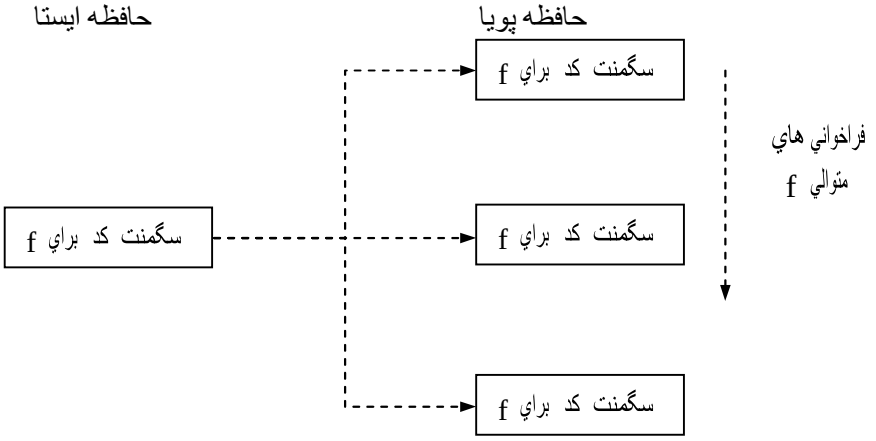
كه سگمنت كد[1] نام دارد و حاوي ثوابت و كد اجرايي است. اين بخش در حين اجراي زيربرنامه بايد ثابت باشد لذا يك كپي از آن ميتواند بين تمام فعاليتهاي زيربرنامه به اشتراك گذاشته شود.

[1] Code segment

بخش پويا :

كه ركورد فعاليت[1] نام دارد و شامل پارامترها ، نتايج تابع و دادههاي محلي و ناحيه حافظه موقت ، نقاط برگشت و پيوندهايي براي مراجعه به متغيرهاي غيرمحلي است. ساختار اين بخش براي تمامي فعاليتهاي زيربرنامهها يكسان است اما مقادير متفاوتي در آنها وجود دارند. لذا هر سابقه فعاليت يك كپي مخصوص به خود از ركورد فعاليت دارد.

نكته : اندازه و ساختار ركورد فعاليت مورد نياز براي يك زيربرنامه ، ميتواند در زمان ترجمه تعيين شود يعني كامپايلر ميتواند تعيين كند چند مولفه براي ذخيره دادههاي ضروري در ركورد فعاليت مورد نياز است. شكل زير مفهوم دو بخش ايستا و پويا را نشان ميدهد :



شكل 6 - 19

وقتي زيربرنامه فراخواني ميشود چند عمل مخفي صورت ميگيرد : (Prolog و Epilog) **Prolog :** اين اعمال بايد قبل از اجراي دستورات زيربرنامه گنجانده شود. انجام اين مقدمات توسط مترجم قبل از اجراي كد زيربرنامه انجام ميشود و شامل عملياتي چون : تنظيم ركورد فعاليت ، انتقال پارامترها ، ايجاد پيوند براي ارجاعهاي غير محلي و ... (عمليات انتقال پارامترها ، push كردن ثباتها و فلگها).

Epilog : اين اعمال هنگام خاتمه زيربرنامه انجام ميشود تا نتايج را برگردانند و حافظه ركورد فعاليت را آزاد كنند.

براي اين اعمال نيز دستوراتي توسط مترجم در انتهاي كد اجرايي قرار داده ميشود (.عمليات انتقال نتايج ، pop كردن ثباتها و فلگها).

دستورات مربوط به Prolog
كد اجرائي
دستورات مربوط به Epilog

[1] Activation record

6-21-2- زيربرنامههاي كلي[1]

مشخصات زيربرنامه ، تعداد ، ترتيب و نوع آرگومانها را مشخص ميكند .زيربرنامه كلي ، زيربرنامه اي با يك نام اما چندين تعريف است كه با امضاهاي مختلف مشخص ميشود .زيربرنامههاي كلي را زيربرنامه مجدداً تعريف شده نيز ميگويند. منظور اين است كه چندين زيربرنامه با اسامي يكسان وجود دارد كه تعداد پارامترها ، نوع و ترتيب آنها متفاوت است.

```
Procedure Enter (Student:integer;Sect:in out Section)
                                        is Begin ….
                                            End;
  Procedure Enter(S:in Section;Tab:in out Classlist)
                                        is Begin ….
                                            End;
```

در مثال فوق مورد اول براي ثبت نام يك دانشجو در يك section ميباشد و مورد دوم يك section به ليست كلاسها اضافه ميكند .اينگونه موارد را زيربرنامههاي كلي يا چند هدفه گويند كه از جهت پياده سازي ، كامپايلر با توجه به تعداد پارامترها ، نوع و ترتيب آنها تشخيص ميدهد كداميك از زيربرنامهها را فراخواني كند و كد لازم براي فراخواني زيربرنامه مربوطه را توليد كند. در فرترن 09 ، تعريف مجدد زيربرنامه با بلوك INTERFACE مشخص ميشود .شكل 51 -6 صفحه 012 در اين مثال زيرروال Enter طوري تعريف شد كه پارامتري از نوع صحيح يا از نوع section را

ميپذيرد. ميتوانيم اين مفهوم را بسط دهيم بطوريكه خود نوع بعنوان پارامتر باشد مثل مكانيزم نوع چند ريختي[2] در ML.

6-21-3- تعريف زير برنامه به عنوان شيء داده

در اغلب زبانهاي كامپايلري مانند فرترن، جاوا و پاسكال تعريف زير برنامه از اجراي آن مستقل ميباشد برنامه منبع توسط كامپايلر ترجمه شده و به شكل اجرايي در ميآيد .در حين اجراي برنامه، بخش ايستاي تعريف زير برنامه غير قابل دستيابي و غير قابل مشاهده است ولي در زبانهايي مانند ليسپ و پرولوگ كه اغلب مفسري هستند بين اين دو(تعريف و اجرا) تمايزي وجود نداشته و با تعريف زير برنامه ميتوان مانند اشياي داده زمان اجرا برخورد كرد.

ترجمه:عملياتي است كه تعريف زير برنامه را به شكل كاراكتري دريافت كرده، شيء داده زمان اجرايي را توليد ميكند .

اجرا:عملياتي است كه تعريفي به شكل زمان اجرا را گرفته، فعاليتي از آن را ايجاد ميكند و آن فعاليت را اجرا مي -كند .

---

[1] Generic subprogram

[2] Polymorphic

در ليسپ و پرولوگ ترجمه، عملياتي است كه ممكن است براي شيء داده كاراكتري در زمان اجرا فراخواني شده تا شكل اجرايي زير برنامه را توليد نمايد. اين زبانها عملياتي به نام define دارند كه بدنه زير برنامه و مشخصات را گرفته و يك تعريف قابل فراخواني از زير برنامه را ايجاد ميكند(define در ليسپ و consult در پرولوگ).

بنابراين در هر دو زبان بالا ميتوان اجراي برنامه را بدون وجود هيچ زير برنامه اي آغاز كرد سپس در حين اجرا ، ميتوان بدنه زير برنامه را خواند يا ايجاد كرد و سپس به شكل اجرايي ترجمه كرد. در حين اجرا ،تعريف زير برنامه ميتواند اصلاح شود. لذا در اين زبانها، در واقع تعريف زير برنامه، يك شيء داده است.

6-31- تعريف نوع[1]

يك زبان برنامه سازي بايد امكاناتي جهت تعريف نوع جديد با توجه به نوعهاي اوليه موجود در زبان فراهم سازد. هر نوع شامل يك Name و يك Description ميباشد

مثال: `Type s:array[1..10] of real`

در تعريف نوع داده انتزاعي كامل، مكانيزمهايي براي توصيف دسته اي از اشياء لازم است، در زبانهايي مانند C، پاسكال و Ada اين مكانيزم تعريف نوع، نام دارد. در زبان پاسكال به كمك دستور Type و در زبان C به كمك دستور Typedef ميتوان تعريف نوع را انجام داد. البته بايد توجه داشت كه تعريف نوع در اين حالت، نوع داده انتزاعي كامل را تعريف نميكند چون عملياتي را بر روي دادههاي آن نوع تعريف نميكند و فقط خود نوع تعريف ميشود .

مثال:

در زير مثالي از تعريف نوع در زبانهاي 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 تعريف نوع ، نام نوع تعيين ميشود و اعلاني وجود دارد كه ساختار دسته اي از اشياء كلاس را توصيف ميكند بدين ترتيب نام نوع به عنوان نام دسته اي از اشياء داده محسوب ميشود و هر وقت به اشياء داده اي از آن ساختارنياز باشد به جاي تكرار توصيف ساختمان داده كافي است نام نوع ارائه شود. به عنوان مثال در بالا اگر به ركوردهاي A,B,C در پاسكال نياز باشد و ساختارهاي يكسان داشته باشند تعريف نوع فوق (مثال بالا) را داريم.

---

[1] Type definition

تعريف نوع، علاوه بر ساده كردن ساختار برنامه مزاياي ديگري براي برنامه نويس به ارمغان ميآورد مثلاً اگر نوع Rational نياز به اصلاح داشته باشد با استفاده از تعريف نوع ، كافي است فقط تغييرات در تعريف نوع ايجاد شود و نه همه متغيرهاي تعريف شده از اين نوع.

مزاياي تعريف نوع:

- ساده كردن ساختار برنامه
- جلوگيري از تكرار Typeهاي موجود در زبان
- ساده تر كردن انتقال پارامترها
- يك شكل جديد از بسته بنديF1174 و پنهان سازي اطلاعات به وجود ميآورد .

از جهت پياده سازي ، كامپايلر بايد هر Type جديد را با توجه به Name و Description در جدولي وارد كند و هنگام اجراي كد و انتقال پارامترها و عمليات ديگر از اين جدول استفاده ميكند .

6-41- هم ارزي نوع[2]

كنترل نوع چه به صورت ايستا و چه به صورت پويا مقايسه بين نوع آرگومانهاي واقعي و نوع دادههايي است كه عمليات انتظار آن را بر دارد. اگر انواع يكسان باشند آرگومان پذيرفته ميشود و عمليات ادامه مييابد ولي اگر يكسان نباشند يا خطا محسوب ميشود يا تبديل ضمني صورت ميگيرد و كار ادامه مييابد .براي بررسي هم ارز (مساوي) بودن دو نوع داده اي ، دو راه حل وجود دارد:الف- هم ارزي نام ب- هم ارزي ساختار **الف- هم ارزي نام[3]:** دو نوع داده هنگامي هم ارز هستند كه نام آنها يكسان باشد.

روش هم ارزي نام در Ada ، C++ و پارامترهاي زير برنامه در پاسكال (نه در بقيه موارد) به كار گرفته ميشود مزيت هم ارزي نام، پياده سازي آسان اين نوع هم ارزي مي 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;
```

[1] Encapsulation

[2] Type Equivalence

[3] Name Equivalence

(X: =Z) در هم ارزي نام معتبر است چون هر دو متغير داراي نام نوع يكسان هستند.

(X:=Y) در هم ارزي نام معتبر نيست چون X از نوع Vect1 و Y از نوع Vect2 است.

معايب هم ارزي نام:

- هر شيء كه در انتساب به كار ميرود

بايد داراي نام باشد يعني انواع داده بي نام[1] وجود ندارد. مثلاً در پاسكال داده بي نام زير را نميتوان به عنوان آرگومان به زير برنامه فرستاد:

```
Var W: array [1…10] of integer;
```

متغير W نوع مستقلي دارد و نميتواند به عنوان آرگومانِ زير برنامه باشد ، زيرا نوع آن فاقد نام است.

- يك تعريف نوع بايد در سراسر برنامه قابل استفاده باشد و يا به عبارتي بايد از تعريف نوع عمومي استفاده گردد زيرا نوع شيء داده اي كه از طريق زنجيره اي از زير برنامهها به صورت آرگومان انتقال داده شود نميتواند در هر زير برنامه تعريف شود. تعريف كلاسها در زبان C++ و اسامي مشخصات پكيج (Package) در Ada و فايلهاي سرايند ".h" در C اين كار را تضمين ميكنند.

ب-هم ارزي ساختاري[2]: دو نوع داده اي هنگامي هم ارز هستند كه ساختار داخلي آنها يكسان باشد منظور از ساختار داخلي يكسان، اين است كه تمام اشياء داده از يك گونه نمايش حافظه، استفاده

كنند. بنابراين در مثال فوق Vect1، Vect2 هم ارز ساختاري دارند زيرا تعداد عناصر ، نوع و ترتيب آنها يكسان است.مزيت هم ارزي ساختار اين است كه انعطاف پذيري برنامه را افزايش مي دهد.

معايب هم ارزي ساختاري:

- در مورد ركوردها، اسامي فيلدها بايد يكسان باشند يا يكسان بودن تعداد و نوع فيلدها كفايت ميكند؟ اگر اسامي ركوردها يكسان باشد آيا ترتيب فيلدها هم بايد يكسان باشد؟
- جهت تشخيص معادل بودن Type بايد هزينه پرداخت شود يعني زمان را از دست ميدهيم، كامپايلر بايد زمان صرف كند كه آيا دو Type معادلند يا نه؟
- دو متغير ممكن است به طور تصادفي (بدون آنكه برنامه نويس بخواهد) از نظر ساختار يكسان شوند در حاليكه از ديد برنامه نويس متفاوت هستند. در اين

موقع زبان برنامه نويسي كمكي به كنترل نوع ايستا نميتواند بكند. مثلاً:

```
Type meter=integer;
 liters=integer; Var
       Len: meter;
      Vol: Liters;
```

anonymous type [1] Structural Equivalence [2]

از ديد زبان برنامه نويسي Len+Vol صحيح است و هيچ خطايي از طرف كامپايلر گرفته نميشود چون ساختارشان يكسان است.(هم ارزي ساختاري) در حاليكه برنامه نويس ميخواهد زبان برنامه سازي به او كمك كند ،اين موضوع بويژه هنگامي كه چندين برنامه نويس روي يك برنامه كار ميكنند ممكن است رخ دهد.

نكته: در زبانهاي قديمي مثل فرترن، كوبول و PL/I تعريف نوع وجود ندارد در نتيجه از هم ارزي نام نميتوان استفاده كرد و از هم ارزي ساختاري استفاده ميشود .پاسكال در هم ارزي نوع، با مشكلاتي روبرو است و از هيچ كدام استفاده نميكند(مگر زير برنامه)، C از هم ارزي ساختاري و C++ از هم ارزي نام استفاده ميكند، طراحي Ada نيز از هم ارزي نام استفاده ميكند.

تساوي دو شيء داده

زماني كه دو شي داده داراي نوع مشابه هستند در برخي اوقات اين مشكل در زبان مطرح مي شود كه آيا دو شي داده باهم برابرند يا خير؟ فرض كنيد دو متغير A,B از نوع X هستند.تحت چه شرايطي مي توانيد بگوييد كه A=B است؟ براي مثال برابري دو پشته يا دو مجموعه. كه در اولي بايد تمامي عناصر تك تك و به ترتيب باهم برابر باشند ولي در ديگري ترتيب مهم نيست. متاسفانه زبان نمي تواند در اين مورد كمك كند.دو تعريف زير را در زبان C براي مجموعه و پشته در نظر بگيريد:

```
Struct set {                    Struct stack {
int  Topstack  int numberinset;  int data[100]; } X,Y;  int
                                                 data[100]; } A,B;
```

انواع X,Y,A,B از نظر ساختاري هم ارزند.يك مقدار صحيح و آرايه اي 001 عنصري از نوع صحيح. ولي تحت چه شرايطي X=Y و A=B ميباشد؟ **تساوي پشتهها:**

اگر فرض كنيم topstack به شيء داده ي موجود در data اشاره مي كند كه در بالاي پشته قرار دارد.تساوي بين x,y را به صورت زير بيان مي كنيم:

1.x.topstack=y.topstack 2.براي تمام Iهاي بين 0و1-topstack

داشته باشيم x.data[i]=y.data[i] در اين صورت x,y پشتههاي برابري را نشان مي دهند.

تساوي مجموعه:

اگر فرض كنيم numberinset تعداد اشياي موجود در A,B است.تساوي بين A,B را به صورت زير تعريف مي -كنيم:

A.numberinset=B.numberinset.1

A.data[0]... A.data[numberinset-1].2 جايگشت B.data[0]…B.data[nomberinset -1]

.است.زيرا ترتيب درج عناصردر مجموعه مهم نيست

تعريف انواعي كه پارامتر دارند:

در برخي از زبانها مانند Ada، اين امكان وجود دارد كه در تعريف نوع از پارامتر استفاده شود و بتوان اشياء داده از نوع يكسان و با اشكال متفاوت ايجاد كرد.به مثالهايي در اين زمينه توجه فرمائيد:

در اينجا maxsize به عنوان پارامتر در تعريف نوع section آمده است:

```
Type section(maxsize:integer) is
                    Record
              Room:integer;
          Instructor:integer;
          Classsize :integer;
End
```

در اين تعريف مي توان متغيري به صورت زير تعريف كرد:record;

```
X:section(100);
Y:section(25);
```

از جهت پياده سازي كامپايلر بايد ارزش پارامتر را محاسبه كرده سپس با توجه به ارزش پارامتر و تايپ مشخص شده، نوع جديدي از متغير تعريف خواهد كرد.در واقع كامپايلر بايد دو مرحله طي

كند:1.ارزشيابي پارامتر 2.تعريف تايپ جديد. در پاسكال نوع پارامتري امكان پذير نيست ولي در++C, Ada ،ML امكان پذير است.

مثالي ديگر از زبان Ada براي تعريف صف كه از Max به عنوان پارامتر در تعريف نوع داده queue استفاده شده است.

```
Type queue (max:integer) record  is
                        Front:integer;
                         Rear:integer;
                Items:array[1…max] of  integer;
End
```

در اينجا مي توان متغيري به صورت زير تعريف كرد:record;

```
X:queue(100);
Y:queue(1000);
```

51-6- سوالات فصل ششم

سوالات تستي

1- به كداميك از روشهاي زير نمي توان نوع داده اي جديد را ايجاد كرد؟(نيمسال اول 58-68) الف.زيربرنامه ب.وراثت ج.اعلان نوع د.چند ريختي

2- وقتي مسير دستيابي به يك شي داده از بين برود .....ايجاد مي شود؟(نيمسال اول 58-68) الف.ارجاع معلق ب.زباله ج.انقياد زودرس د.انقياددير رس

3- در مورد ساختارهاي ركورد و آرايه كدام جمله صحيح نيست؟(نيمسال اول 58-68) الف.عناصر ركورد و آرايه همگن هستند. ب.عناصر ركورد داراي نام هستند.

ج.ركوردها و بردارها ساختمان داده خطي هستند. د.ركوردها و بردارها ساختمان داده خطي با طول ثابت هستند.

4- نوع ركورد متغير را .....مي نامند. (نيمسال اول 58 -68)

الف.ركورد غير همگن ب.ركورد همگن ج.يونيون د.ليست

5- اگر در ركوردي با طول متغيري از عناصر، تعداد عناصر بدون هيچ محدوديتي تغيير كند آنرا .....مي ناميم.

(نيمسال اول 58-68)

الف.درخت ب.گراف ج.ليست خاصيت د.ركورد ناهمگن

6- بخش پوياي سابقه فعاليت مربوط به يك زير برنامه .....ناميده مي شود.
(نيمسال اول 58-68) الف.سگمنت كد ب. سگمنت داده ج.امضاي
زيربرنامه د.ركورد فعاليت **7-** كدام گزينه غلط است؟(نيمسال دوم 58 - 68)
الف.در PERL آرايه انجمني به وسيله عملگر = ايجاد مي شود.
ب.طول عمر شي داده با انقياد شي به محلي از حافظه شروع مي شود.
ج.ركورد، ساختمان داده اي مركب از تعداد ثابتي از عناصر و از انواع مختلف مي باشد.
د.هيچكدام
8- شكلهاي گوناگون ليستها عبارتند از:(نيمسال دوم 58 -68)
الف.پشتهها و صفها ب.درختها و گرافهاي جهت دار ج.الف و ب
د.آرايهها **9-** كدام گزينه غلط مي باشد؟(نيمسال دوم 58-68)
الف.پنهان سازي اطلاعات , اصطلاح مهمي در طراحي انتزاعهاي برنامه نويسي است.
ب.زبان برنامه سازي انتزاع را به دو روش حمايت مي كند.
ج.براي نوشتن برنامه بزرگ بايد از استراتژي تقسيم و غلبه استفاده كرد.
د.هيچكدام
01- نوع داده انتزاعي شامل كدام موارد زير است؟(نيمسال دوم 58-68) الف.نوع داده اي كه توسط برنامه نويس تعريف مي شود.
ب.مجموعه اي از عمليات انتزاعي بر روي اشيائي از آن نوع.
ج.بسته بندي اشياي آن نوع بطوريكه كاربر آن نوع، نمي تواند آن اشيا را بدون استفاده از اين عمليات دستكاري نمايد.
د.كليه موارد بالا
11- اگر طول اجزاي يك ساختمان داده ثابت باشد و اجزاي آن همگن باشد در پياده سازي آن كدام مورد صحيح است؟(نيمسال اول 68 -78)
الف.نمايش حافظه پيوندي و هر جز يك توصيف كننده لازم دارد.
ب.نمايش حافظه پيوندي و كل اجزا يك توصيف كننده دارند.
ج.نمايش حافظه ترتيبي و كل اجزا يك توصيف كننده دارند.
د.نمايش حافظه ترتيبي و هر جز يك توصيف كننده لازم دارد.
21- كدام يك از موارد زير صحيح نيست؟(نيمسال اول 68 -78)
الف. در نمايش حافظه پيوندي با عمل انتخاب عنصر تصادفي امكان پذير است.
ب .در نمايش حافظه پيوندي با عمل انتخاب عنصر ترتيبي امكان پذير است.
ج .در نمايش حافظه ترتيبي با عمل انتخاب عنصر تصادفي امكان پذير است.
د .در تمايش حافظه ترتيبي با عمل انتخاب عنصر ترتيبي امكان پذير است.

31- در صورتي كه طول عمر يك شي داده قبل از پايان طول عمر آن از بين برود چه اتفاقي مي افتد؟(نيمسال اول 86-87)

الف.مشكلي به نام ارجاعهاي سرگردان بوجود مي آيد.

ب.مشكلي به نام activation record بوجود مي آيد.

ج.مشكلي به نام حافظههاي بدون استفاده بوجود مي آيد.

د.مشكلي بوجود نمي آيد.

41- در صورتي كه مسير دستيابي يك شي داده اي پس از آنكه طول عمر شي داده اي خاتمه يافت وجود داشته باشد، چه اتفاقي مي افتد؟(نيمسال دوم 68-78)

الف.مشكلي به نام ارجاعهاي سرگردان بوجود مي آيد. ب.مشكلي به نام حافظه زباله بوجود مي آيد.

ج.مشكلي به نام ركورد فعاليت بوجود مي آيد. د.هيچ مشكلي رخ نمي دهد.

51- براي كنترل ترتيب در عبارات غير محاسباتي از چه روشي استفاده مي شود؟(نيمسال دوم 68 -78) الف.نمايش درختي ب.انتساب اشياي داده

ج.تطابق الگو د.هيچكدام

61- تعريف زير را در نظر بگيريد كدام گزينه صحيح است؟(نيمسال دوم 68 -78)

```
Type vect1: array [1...10] of real;  vect2: array [1...10] of
real;  Var x, z: vect1;          y: vect2;  .
```

الف x,y,z.هم ارزي نام دارند

ب. x,z هم ارزي نام و x,z با y هم ارزي ساختاري دارند.

ج.x با y هم ارزي ساختاري و z با y هم ارزي نام دارند.

د.x و z هم ارزي ساختاري و y با x هم ارزي نام دارند.

71- ركورد متغير زير براي تعريف خود به چند بايت نياز دارد؟(integer دو بايت و real شش بايت و char يك بايت) (نيمسال دوم 68 -78)

```
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);
```

الف19. ب18. ج16. د27.

81- منظور از ركورد فعاليت چيست؟(نيمسال دوم 68-78)

الف.بخش ايستاي زير برنامه ب.بخش پوياي زير برنامه

ج.بخش ايستا به همراه بخش پويا زير برنامه د.پخش پويا به همراه كد سگمنت زير برنامه

91- اگر طول اجزاي يك ساختمان داده ثابت باشد و اجزاي آن همگن باشد در پياده سازي آن كدام مورد صحيح است؟ (نيمسال دوم 68 -78)

الف.نمايش حافظه پيوندي و هر جز يك توصيف كننده لازم دارد.

ب.نمايش حافظه پيوندي و كل اجزا يك توصيف كننده دارند.

ج.نمايش حافظه ترتيبي و كل اجزا يك توصيف كننده دارند.

د.نمايش حافظه ترتيبي و هر جز يك توصيف كننده لازم دارد.

02- عمليات تعريف شده برروي بردارها در كدام يك ازبانهاي زير بيشتر از بقيه است؟(نيمسال دوم 68-78)

Perl. د.APL ج.C ب.Pascal الف

12- كداميك از زبانهاي زير براي پردازش ليستها مي باشند؟(نيمسال دوم 68 -78)

Perl. د.Lisp ج.Pascal ب.Ada الف

22- در صورتي كه تمام اشاره گرهايي كه به شي داده اشاره مي كنند از بين بروند چه پديده اي اتفاق مي افتد؟(نيمسال اول 78-88)

الف. مشكلي به نام ارجاعهاي سر گردان بوجود مي آيد.

ب .مشكلي به نام حافظه زباله بوجود مي آيد.

ج .مشكلي به نام ركورد فعاليت بوجود مي آيد.

د .هيچ مشكلي بوجود نمي آيد.

32- براي پياده سازي مجموعهها چنانچه اندازه مجموعه جهاني بزرگ باشد كدام يك از روشهاي نمايش حافظه زير مناسب است؟(نيمسال اول 78-88)

الف.نمايش بيتي مجموعهها ب.نمايش درهم سازي مجموعهها

ج.نمايش درختي مجموعهها د.نمايش بيتي درختي مجموعهها

42- در كداميك از زبانهاي زير اشياء داده اي و برنامههاي اجرايي كه دستكاري بر روي اشيا داده اي را انجام مي دهند ساختارهاي مجزايي ندارند و اصطلاحاً اشياي داده اجرايي داريم؟(نيمسال اول 78-88)

Prolog, Lisp. د.Ada,Lisp ج.C,Lisp ب.Ada,C الف

52- تعريف زير را در نظر بگيريد كدام گزينه صحيح است؟(نيمسال اول 78-88)

Type

```
vect1: array [1.. 10] of real; vect2:
        array [1.. 10] of real;
```

```
Var x, z: vect1; y: vect2;
```

الف.x,y,z هم ارزي نام دارند.
ب.x,z هم ارزي نام وx,z با y هم ارزي ساختاري دارند.
ج.x با y هم ارزي ساختاري و z با y هم ارزي نام دارند.
د.xو z هم ارزي ساختاري و y با x هم ارزي نام دارند.

26-كدام گزينه صحيح است؟(نيمسال اول 78 -88)

الف.پنهان سازي اطلاعات،اصطلاح مهمي در طراحي انتزاعهاي برنامه نويسي است.
ب.بسته بندي بر روي آرايههاي چند بعدي امكان پذير نيست.
ج.در برخي زبانها بسته بندي به وسيله زيربرنامه صورت مي گيرد.
د.الف و ج

27-كدام دسته از زبانهاي زير از آرايههاي انجمني استفاده مي كنند؟(نيمسال اول 78 -88)

الف C,Pascal.ب Perl,Snobol4.

ج C,Fortran.د Pascal,Cobol.

82- در صورتي كه مسير دستيابي يك شي داده اي پس از آنكه طول عمر شي داده اي خاتمه يافت وجود داشته باشد چه اتفاقي مي افتد؟(نيمسال دوم 78-88) الف: مشكلي به نام ركورد فعاليت بوجود مي آيد.
ب .مشكلي به نام ارجاعهاي سرگردان بوجود مي آيد.
ج .مشكلي به نام حافظه زباله بوجود مي آيد.
د .مشكلي به نام سرريزي صف بوجود مي آيد.

92- ركورد متغير زير براي تعريف خود به چند بايت نياز دارد؟(integer دو بايت و real شش بايت و char يك بايت) (نيمسال دوم 78 -88)

```
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);
```

الف19. ب18. ج16. د27.

03- براي پياده سازي مجموعهها چنانچه اندازه مجموعه جهاني كوچك باشد كدام يك از روشهاي نمايش حافظه زير مناسب است؟(نيمسال دوم 78-88)

الف. نمايش بيتي مجموعهها ب .نمايش درهم سازي
مجموعهها ج .نمايش درختي مجموعهها د .نمايش
بيتي درختي مجموعهها

13- براي بسط مفهوم بسته بندي به دادههايي كه توسط برنامه نويس تعريف مي شود نوع داده انتزاعي با فراهم كردن كداميك از موارد زير بدست مي آيد؟(نيمسال دوم 78-88)

مورد اول: مجموعه اي از اشياي داده معمولا با استفاده از يك يا چند تعريف نوع **مورد دوم:** مجموعه اي از عمليات انتزاعي بر روي آن انواع داده

مورد سوم: بسته بندي تمام آنها بطوري كه كاربر نوع جديد نتواند اشياي داده اي از آن نوع را به جز از طريق عملياتي كه بر روي آن تعريف شده است دستكاري كند.

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

23- ركورد فعاليت يك زير برنامه شامل كداميك از موارد زير نمي باشد؟(نيمسال دوم 78 -88) الف. نتايج تابع و دادههاي محلي ب. ثوابت و كد اجرايي
ج .ناحيه حافظه موقت و دادههاي محلي د. پارامترهاي ارسالي و ناحيه حافظه موقت

33- زير برنامه اي با يك نام اما چندين تعريف كه با امضاءهاي مختلف مشخص مي شوند را چه مي نامند؟(نيمسال دوم 78-88)

الف. زير برنامه كلي ب. زير برنامه محلي ج. زير برنامه غير محلي د. زير برنامه بازگشتي

43- تعريف روبه رو را در نظر بگريد كدام گزينه صحيح است؟(نيمسال دوم 78 -88)

```
Type vect: array [1...10] of real;
  Verct2: array [1...10] of real;
    Var x, y: vect; z: vect2;
```

الف: x,y,z هم ارزي نام دارند.
ب: x,z هم ارزي نام و x با y هم ارزي ساختاري دارند.
ج: x با y هم ارزي ساختاري و z با y هم ارزي نام دارند.
د: x,z هم ارزي ساختاري و y با x هم ارزي نام دارند.

53- كدام گزينه موجب رخ دادن پديده زير مي شود؟(نيمسال اول 88 -98)

»مقداري به شي داده اي نسبت داده مي شود كه آن شي وجود ندارد محتويات محلي از حافظه را كه به شي داده ديگري اختصاص يافته است تغيير دهيد وممكن است محلي از حافظه را كه توسط مديريت حافظه تنظيم شده است خراب كند«

الف.ارجاعهاي معلق ب.زباله ج.پشتههاي زمان اجرا د.بافر صفحه كليد

63- تكه برنامه زير كداميك از مشكلات مديريت حافظه را در زبان C++ ايجاد مي كند؟(نيمسال دوم 88 -98)

```
Int *p,*q; p = new(int); q = new(int); p = q;
```
الف.اختصاص حافظه ب.زباله ج .ارجاع معلق د.آزادسازي حافظه

73- اگر در ركوردي با تعداد متغيري از عناصر، تعداد عناصر بدون هيچ محدوديتي تغيير كند، آن ركورد چه نام دارد؟ (نيمسال دوم 88-98)

الف.ركورد با طول متغير ب. ليست خاصيت ج.يونيون آزاد د.يونيون قابل تشخيص

83- كدام روش پياده سازي مجموعهها براي نمايش مجموعههايي است كه مجموعه مرجع آنها كوچك است؟
(نيمسال دوم 88 -98)

الف.نمايش درهم سازي مجموعهها ب.نمايش حافظه پراكنده ج.نمايش بيتي د. نمايش حافظه ترتيبي

93- در تكه كد زير آدرس فيلدهاي x وy در ركورد rec نسبت به هم چگونه اند؟ (نيمسال دوم 88-98)

```
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 فيلدهاي x و y يكسان است. د.آدرس انتهاي فيلدهاي x و y يكسان است.

04- قسمتي از حافظه stack اجراي برنامه كه شامل پارامترها، نتايج تابع دادههاي محلي و ناحيه حافظه موقت است چه نام دارد؟(نيمسال دوم 88-98)

الف.سگمنت كد ب.ركورد فعاليت ج.بافر د.حافظه هرم

14- در كداميك از زبانهاي زير براي پياده سازي ليستها سيستم مديريت حافظه مخفي وجود دارد؟ (نيمسال دوم
88-89)

الف ب. Ada ج .Pascal د .C++ .ML

24- در تعريف آرايه زير در زبان پاسكال ، طول آرايه در چه زماني مشخص مي شود؟ (نيمسال دوم 88 -98)

```
Name : packedarray[1..20] of char;
```

الف.زمان تعريف زبان ب. زمان اجرا ج.زمان پياده سازي د.زمان كامپايل

43- در زباني كه از هم ارزي استفاده ميكند ، تعريف زير وجود دارد. كدام گزينه درست است. (نيمسال اول 89 - 90)

```
x=array[1..10] of     Type
y=array[1..10] of    char;
                      char;
                   var
                 a,b:x;
```

. z:y; غير مجاز است b:=z و a:=b. مجاز است ب b:=z و a:=b

.الف

ج .a:=b و b:=z غير مجاز است. د .a:=b مجاز و b:=z غير مجازا ست.

44- تعريف زير را در زبان C براي پشته در نظر بگيريد. اگر انواع x,y از نظر ساختاري هم ارز باشند كدام گزينه صحيح است. (نيمسال اول 98-09)

```
          Struct stack
int  int top;    {
          data[100];
               }x,y;
```

الف -1 topstack ها بين 0 و iبراي تمامي x.data[i]=y.data[i] و x.top=y.top

.ب -1 topstack ها بين 0 و iبراي تمامي x.data[i]!=y.data[i] و x.top=y.top.

ج -1 topstack ها بين 0 و iبراي تمامي x.data[i]=y.data[i] و x.top!=y.top.

د -1 topstack ها بين 0 و iبراي تمامي x.data[i]!=y.data[i] و x.top!=y.top.

45- در تعريف ساختار زير اشاره به كدام ويژگي در نرم افزار دارد. (نيمسال اول 98 -09)

```
          Type s(max:integer) is
r:integer;          record
c:integer rang 0..max;
                end record
```

طراحي و پياده سازي زبانهاي برنامه سازي 191

```
x:s(200);
```

الف. هم ارزي ساختاري ب .انواع پارامتري ج .هم ارزي نوع د .هم ارزي نام **46**- در زباني مثل ليسپ حافظه هرم شامل چه نوع اطلاعاتي ميباشد. (نيمسال اول 98-09) الف. عناصر ليست پيوندي ب .پشته براي ارزيابي توابع جزيي

ج .روالهاي سيستم د .روالهاي I/O

47- كدام مورد زير فعاليتهاي مربوط به انتقال پارامترها را كامل ميكند و محتويات پارامترهاي واقعي را در پارامترهاي مجازي كپي ميكند(.نيمسال اول 98 -09)

الف prologue . ب epiloge.

ج .زنجيره اشاره گر ايستا د .زنجيره اشاره گر پويا

6-61- پاسخنامه سوالات تستي فصل ششم

سوال	الف	ب	ج	د
1				*
2		*		
3	*			
4			*	
5			*	
6				*
7	*			
8			*	
9				*
10				*
11			*	
12	*			
13			*	
14	*			
15			*	
16		*		
17		*		
18		*		
19			*	
20			*	

سوال	الف	ب	ج	د
21			*	
22		*		
23		*		
24				*
25		*		
26				*
27		*		
28		*		
29		*		
30	*			
31				*
32		*		
33	*			
34				*
35	*			
36		*		
37		*		
38				*
39			*	
40		*		
41				*
42		*		
43				*
44	*			
45		*		
46	*			

			*	**47**

سوالات تشريحي

1- آرايههاي چند بعدي در زبانهاي برنامه سازي چگونه پياده سازي مي شوند.محاسبه فرمول دسترسي به يك عنصر خاص در يك آرايه دو بعدي را محاسبه كنيد. (نيمسال اول 58 -68)

2- فرمول دسترسي تصادفي به عناصر در آرايه دو بعدي را به روش سطري بدست آوريد؟(نيمسال اول 68 -78)

3- صفات اصلي مشخص كننده ساختمان داده را شرح دهيد؟ (نيمسال دوم 58-68)

4- براي بردار A[Lb1…ub1,Lb2…ub2,Lb3…ub3] فرمول دسترسي تصادفي به عناصر را به روش سطري بدست آوريد؟(نيمسال دوم 68 -78)

5- براي آرايه a[1..3,-1...1] نمايش حافظه آنرا رسم كنيد؟(نيمسال دوم 78-88)

6- براي بردار A[Lb1….ub2,Lb2….ub2,Lb3……ub3] فرمول دسترسي تصادفي به عناصر را به روش ستوني بدست آوريد؟(نيمسال اول 78 -88)

7- ركورد متغير در زبان پاسكال را به همراه يك مثال شرح دهيد؟(نيمسال اول 88 -98)

8- اگر تابع زير در زباني مثل C نوشته شود، اطلاعات موجود در سگمنت كد و ركورد فعاليت آن را با توجه به متغيرها و ثابتهاي محلي آن مشخص كنيد؟(نيمسال دوم 88 -98)

```
float func1(int x,float y,char c){
    const int a = 20;
        float b;
    char c;    int
    دستورات w; sub
    اجرايي زير برنامه
    ) returnنتيجه تابع (
    }
```

9-با تعريف ساختمان داده زير، آدرس محل داده اي array[20].grade[3] را محاسبه كنيد. (با فرض آدرس پايه αو نوع صحيح 4 بايتي و نوع اعشاري6 بايتي) (در زبان C انديس آرايه از صفر شروع مي شود) (نيمسال دوم 88-89)

```
struct student {
    int number;
float grade[10];
  }array[100];
```

01- نمايش حافظه ركوردي با طول متغيري به صورت زير چگونه است. نمايش حافظه آن را ترسيم نماييد(نيمسال اول 98-09)

```
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;
```

فصل هشتم:

كنترل ترتيب اجرا

ا # ا :

كنترل ترتيب در عبارات محاسباتي
ارزيابي نمايش درختي عبارات
كنترل ترتيب بين دستورات
دستور
goto دستور
مركب دستور
شرطي دستور
تكرار برنامههاي
بنيادي
كنترل ترتيب عبارات غير محاسباتي
سوالات تستي و تشريحي

8-1- مقدمه

منظور از كنترل ترتيب،كنترل ترتيب اجراي عمليات(عمليات اوليه و عمليات تعريف شده توسط كاربر) و منظور از كنترل داده،كنترل انتقال دادهها بين زير برنامهها و برنامهها مي باشد.

ساختارهاي كنترل ترتيب :

در حالت معمولي دستورات پشت سرهم اجرا ميشوند اما در مواردي ترتيب اجرا بر اثر مواردي نظير دستورات شرطي يا حلقهها عوض ميشوند. ساختارهاي كنترل ترتيب به 4 دسته تقسيم ميشوند:

- ساختارهايي كه در عبارات(محاسباتي)استفاده ميشوند مانند قواعد مربوط به تقدم عملگرها و پرانتزها.
- كنترل ترتيب بين دستورات مثل جملات تركيبي ،جملات شرطي، حلقهها.

- كنترل ترتيب در عبارات غير محاسباتي مانند برنامه نويسي اعلاني كه در پرولوگ استفاده ميشود.
- كنترل ترتيب در زير برنامهها مانند صدازدن زيربرنامهها كه كنترل برنامه را از نقطه اي به نقطه اي ديگر انتقال ميدهند (فصل 9)

به ياد داشته باشيد كه بعضي از زبانها مثل APL وليسپ فاقد كنترل دستورات هستند و فقط شامل عبارات اند.از يك جنبه ديگر ساختارهاي كنترل ترتيب به دو دسته تقسيم مي شوند.

صريح[1]:ساختار كنترل ترتيب صريح آنهايي هستند كه توسط برنامه نويس تعيين ميشوند تا ساختارهاي ضمني تعريف شده توسط زبان را عوض كند مانند استفاده از پرانتز در عبارات رياضي يا استفاده از دستورات go to. ضمني[2]: اگر كنترل ترتيب توسط زبان برنامه نويسي تعيين شود به آن كنترل ترتيب ضمني گفته ميشود مانند تقدم عملگر * نسبت به +

2-8- كنترل ترتيب در عبارات محاسباتي

براي كنترل ترتيب در عبارات محاسباتي از دو روش نمايش براي عبارات محاسباتي استفاده مي شود.

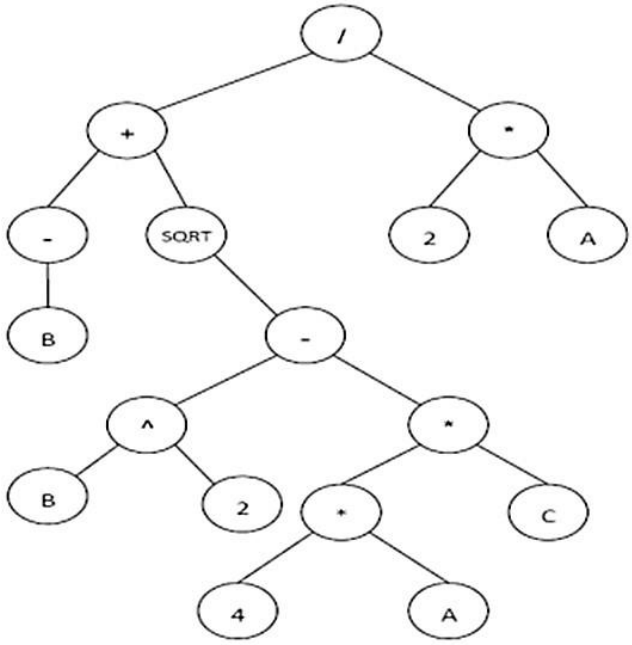
- نمايش درختي
- prefix,infix,postfix روشهاي

نمايش درختي : جهت كنترل ترتيب در عبارات رياضي ميتوان از روش نمايش درختي استفاده كرد. در اين روش ريشه درخت عمليات اصلي ،برگها دادهها وگرههاي بين ريشه و برگها عمليات مياني را نشان ميدهند. به عنوان مثال فرمول محاسبه ريشه معادله درجه دوم در فرترن به صورت زير است. نمايش درختي اين فرمول به شكل زير مي باشد.

$$\frac{-B + SQRT(B^{2} - 4 * A * C)}{2 * A}$$

[1] Explict

[2] implict



شكل 8 - 1

به هنگام توليد كد توسط كامپايلر، جهت ارزشيابي عبارت فوق ، حداقل 51 دستور (بادر نظر گرفتن جذر و شمارش ارجاع به داده)نياز است.حال بحث اين است كه ترتيب اجراي اين 51 دستور چگونه است ؟ اما اين نحوه ارزيابي ايراداتي دارد همانند نقض اولويتهاي ارزيابي (كدام دستور زودتر ارزيابي شود). مثلاً مشخص نيست كه آيا 4*A بايد قبل از B^2 ارزيابي شود يا بعد از آن. همچنين مشخص نيست كه آيا دو ارجاع به B ميتواند در يك ارجاع تركيب شود يا خير؟ بنابراين در كنترل برنامه ابهاماتي وجود دارد. براي رفع اين ايرادات از روشهاي نشانه گذاري خاصي استفاده ميشود .

prefix,infix,postfix: روشهاي نشانه گذاري

الف- پيشوندي(Polish-Prefix): در اين روش عملگرها قبل از عملوندهايشان قرار مي گيرند. براي مثال عبارت c*(a+b) در فرم پيشوندي به صورت abc+* نمايش داده مي شود. نوع ديگري از اين روش وجود دارد كه به نام Cambridge Polishمعروف است كه در آن عملگر به همراه عملوندهايش توسط پرانتز احاطه مي گردند كه در زبان LISP مرسوم است. عبارت c*(a+b) در فرم Cambridge Polish به صورت(c(ab+)*) نمايش داده مي شود.

ب- ميانوندي(Infix): در اين روش عملگرهاي دودويي در بين عملوندهايشان قرار مي گيرند و براي برهم زدن ترتيب ارزشيابي بر اساس تقدم از پرانتزگذاري استفاده مي شود. مانند c*(a+b). اين روش در عمگرهاي 3 تايي نامناسب مي باشد به عنوان مثال:exp1?exp2:exp3

ج- پسوندي(Postfix-Reverse Polish): در اين روش عملگرها بعد از عملوندهايشان قرار مي گيرند.

براي مثال عبارت (a+b)*c در فرم پسوندي به صورت ab+c* نمايش داده مي شود.

- مزاياي استفاده از روش پيشوندي و پسوندي نسبت به ميانوندي:
- عدم نياز به پرانتز گذاري
- عدم نياز به قرار دادن اولويت يا شركت پذيري
- ارزشيابي انها به راحتي توسط يك الگوريتم كارا انجام مي شود و لذا براي استفاده در زبانهاي برنامه سازي مناسب است.
- قابل اعمال براي هر نوع عملگري با هر تعداد عملوند مي باشند.
- داراي نحوي به مانند فراخواني توابع مي باشند.

ارزيابي عبارات prefix:

علاوه بر صرفه جويي پرانتزها،نشانه گذاري prefix ارزشهاي خاصي در زبانهاي برنامه نويسي دارد:

- فراخواني تابع به صورت نشانه گذاري prefix انجام ميشود .F(x,y,z)
- نشانه گذاري prefix ميتواند براي نمايش عملياتي با هر تعدادي از عملوندها به كار گرفته شود مانند

. در ليسپ combridgepulish روش

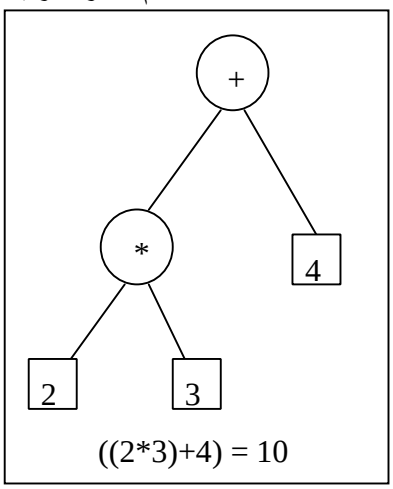
- نشانه گذاري prefix را به راحتي ميتوان به طور مكانيكي رمز گشايي كرد.

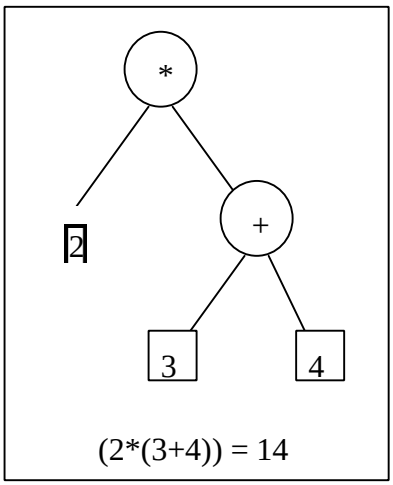
ارزيابي عبارات postfix:

ارزيابي عبارات infix : گرچه نشانه گذاري infix متداول است ولي معايبي دارد:

الف: چون نشانه گذاري infix فقط براي عملگرهاي دودويي مناسب است زبان نميتواند فقط از اين نشانه گذاري استفاده كند بلكه حتماً بايد infix با يكي از دو روش prefix يا postfix تركيب شود كه اين تركيب خود مشكلاتي به همراه خواهد آورد.

ب : در محاسبه بعضي عبارات رياضي ممكن است ابهام بوجود آيد مثلاً در عبارت A*B+C معلوم نيست كه كدام يك از عملها بايد ابتدا انجام شود و ترتيب اجراي هريك ميتواند نتيجه را بسيار متفاوت كند. به شكل زير توجه كنيد:





شكل 8 - 2

در اغلب زبانها براي اينكه ترتيب اجراي عملگرها ابهام بر انگيز نباشد ونياز به پرانتزهاي متعدد در نماد infix نيز وجود نداشته باشد از قواعد تقدم عملگر استفاده ميشود اين ترتيب عملگرها در همه زبانها يكسان نيست و هر زبان ترتيب خاص خود را دارد ولي عموماً در مورد عملگرهاي رياضي ابتدا ضرب و تقسيم و سپس جمع و تفريق انجام ميشود.همچنين در اكثر زبانها،عملگرهايي كه در يك سطح اولويت هستند از چپ به راست اجرا ميشوند.مثلاً ترتيب a+b+c به صورت ((a+b)+c) ميباشد ولي در مورد توان و انتساب عموماً اين ترتيب از راست به چپ اجرا ميشود .

$a \uparrow b \uparrow c \rightarrow (a$ يا $a=b=c \rightarrow (a=(b=c))$

جدول C,Ada در زير آورده شده است $(b\uparrow c))\uparrow$ در : تقدم عملگر مربوط به زبانهاي

سطح تقدم	عملگرها	عمليات
بالاترين تقدم	Not , abs , **	توان ، قدرمطلق ، نقيض
	/ , mod , rem	ضرب و تقسيم
	+ -	جمع و تفريق يكاني
	+ - &	جمع و تفريق دودويي
	= , <= , < , > , >=	رابطه اي
پايين ترين تقدم	And , or , xor	عمليات بولين

جدول 8 - 1 سلسله مراتب عمليات در ادا.

تقدم	عملگرها	اسامي عملگرها

17	Tokens,a[k],f()	ليترالها، انديس ، فراخواني تابع
	., ->	انتخاب
16	++, --	افزايش و كاهش پسوندي
15 *	++, --	افزايش و كاهش پيشوندي
	sizeof ,_ , ~	عملگاهاي يكاني ، حافظه
	* , & , !	نقيض منطقي ، آدرس دهي غير مبهم
14	(type name)	تبديل ضمني
13	* , / , %	عملگرهاي ضرب
12	+ , -	عملگرهاي جمع
11	<<, >>	شيفت
10	=< , => , < , >	رابطه اي
9	=! , ==	تساوي
8	&	And بيتي
7	^	Xor بيتي
6	|	Or بيتي
5	&&	And منطقي
4	||	Or منطقي
3*	? :	شرطي
2*	=,+=,-=,*=,/= , %= , <<= , >>= , &= , ^= , |=	انتساب
1		ارزيابي ترتيبي

جدول 8 - 2 سطوح تقدم عملگرها در C

*: عملگرهايي كه از راست به چپ ارزيابي ميشوند .

اگر زبانها حاوي عملگرهايي باشند كه در رياضيات كلاسيك موجود نباشند تقدمها با شكست مواجه خواهند شد.

Forth,APL,C و اسمالتاك نمونههايي از زبانهايي هستند كه عملگرهاي توسعه يافته دارند.بنابراين توضيح مختصري در مورد هريك ارائه مي كنيم:

APL :

زبان APL زباني است كه براي كاركردن روي آرايهها ساخته شده است در اين زبان دستورات و عبارات از راست به چپ ارزيابي ميشوند به علت دقت عبارات APL ، اين زبان كوچك است ولي در عين حال برنامه نويس مي-تواند برنامههاي يك خطي پيچيده اي را بنويسد.

:Forth

زبان Forth براي كامپيوترهاي بي درنگ[1] طراحي شده است از آنجا كه در آن زمان (1960) حافظهها گران بودند زبان فورث را به گونه اي طراحي كردند كه كوچك باشد ودرعين حال ترجمه آن ساده بوده و كارايي اجراي آن هم خوب باشد. در اين زبان از نماد postfix براي عبارات استفاده ميشود .اين زبان تفسيري است و دو پشته دارد يكي جهت برگشت زير برنامهها وديگري پشته ارزيابي عبارات.

مطالعه زبانهاي C و اسمالتاك به عهده دانشجوي محترم مي باشد

روشهاي ارزيابي عبارات محاسباتي در زمان اجرا:

به علت سخت بودن رمزگشايي عبارات infix بهتراست فرم infix ابتدا به شكل اجرايي تبديل شود تادر حين اجرا به سادگي رمز گشايي شود براي اين كار سه روش مختلف وجود دارد.

- **دنباله اي از كد ماشين:** اگر عبارات را به يك سري كد ماشين واقعي تبديل كنيم ، ترتيب اين دستورات ، ترتيب عمليات را مشخص ميسازند. اين روش سرعت اجراي بالايي دارد مانند C,C++,Pascal
- **ساختارهاي درختي:** دراين روش در مرحله اول عبارات به كمك مفسر نرم افزاري به شكل درختي در آمي يند سپس در مرحله دوم يعني اجرا ، پيمايش درخت انجام ميشود اين تكنيك در زبان LISP استفاده ميشود.
- **فرم prefix يا postfix :** دوروش prefix و postfix طبق الگوريتمهاي قبلي ميتوانند اجرا شوند .در برخي از كامپيوترهاي واقعي كه برمبناي پشته كار ميكنند كد واقعي ماشين به شكل postfix است نمايش prefix در اسنوبال 4 استفاده ميشود .

3-8- ارزيابي نمايش درختي عبارات

رويه اصلي ترجمه براي عبارات كه به صورت درختي نمايش داده ميشوند ، آسان است ولي در مرحله دوم كه درخت به دنباله اي از عمليات اجرايي تبديل ميشود با مشكلاتي روبرو هستيم كه عبارتند از :

- قواعد ارزيابي يكنواخت
- اثرات جانبي

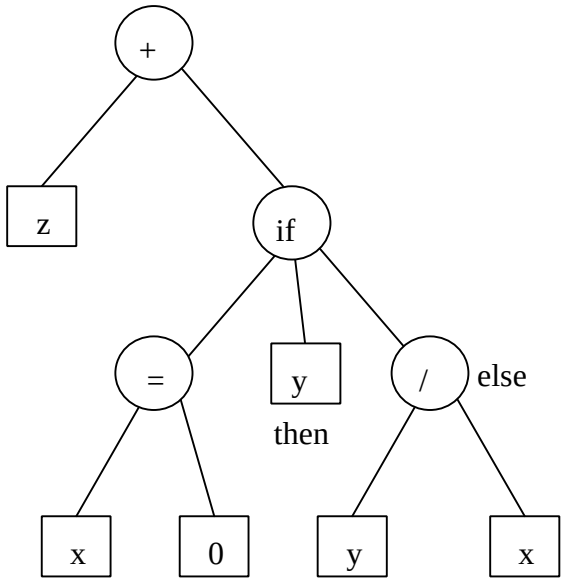
- شرايط خطا
- عبارات بولين مدار كوتاه

realtime [1]

قواعد ارزيابي يكنواخت:

براي ارزيابي، دو تكنيك ارزيابي عجول[1] و تنبل[2]وجود دارد. در ارزيابي عجول ، همواره در ابتدا عملوندها ارزيابي ميشوند وسپس عمليات را برروي عملوندهاي ارزيابي شده اجرا ميكنيم .اين قاعده را عجول ميناميم زيرا هميشه اول عملوندها را ارزيابي ميكنيم در اين روش ترتيب دقيق ارزيابي عمليات مهم نيست مثلاً براي محاسبه (a+b)*(c-a) ممكن است اول (a+b) يا اول (c-a) محاسبه شود. ولي اين روش هميشه امكان پذير نيست. مثلاً در عبارت شرطي زير به زبان C كه مانند if عمل ميكند اگر y صفر نباشد x/y محاسبه ميشود حال با روش عجول اگر حتي y=0 باشد عبارت x/y تقسيم و محاسبه ميگردد كه ايجاد خطا ميكند .

w:= z+(if(x=0) then y else y/x) آنگاه هم ارز: w:= z+(x=0 ? y: y/x) با فرض



شكل 8 - 3

بنابراين براي رفع مشكل بالا ازقاعده ارزيابي تنبل استفاده ميكنيم .در روش ارزيابي تنبل، عملوندها قبل از اجراي عمليات ارزيابي نميشوند ،بلكه عملوندهاي ارزيابي نشده ، ارسال شده وعمليات تصميم ميگيرد كه ارزيابي لازم است يا خير. اين روش همواره درست عمل ميكند ولي پياده سازي آن مشكل است. زبانهاي محاوره اي مانند ليسپ وپرولوگ از اين تكنيك استفاده ميكنند .در حاليكه زبانهاي محاسباتي مانند C و فرترن علاقه اي به آن ندارند. اصطلاحات عجول و تنبل معادل دو تكنيك ارسال پارامتر به زير برنامهها يعني انتقال پارامتر با مقدار و با نام است.

عجول ← با مقدار تنبل ← با نام

---

[1] eager

[2] lazy

مثالي از روش تنبل اين است كه در عبارات and ، اگر يكي از عملوندها false بود عملوند ديگرمحاسبه نميشود وجواب كلي false اعلام ميگردد .

اثرات 184184184184جانبيF_1:

استفاده از عملياتي كه اثرات جانبي بر عبارات دارند مسأله ديگري است كه بايد به آن توجه داشت. مثلاً درعبارت a* fun(x)+a مطلوب آن است كه a فقط يك بار ارزيابي شده ودر دو جاي محاسبات استفاده گردد. همچنين ارزيابي fun(x) قبل يا بعد از دستيابي به a فرق نداشته باشد. ولي اگر fun(x) روي a اثر جانبي داشته باشد و آن را تغيير دهد آن گاه ترتيب دقيق ارزيابي بسيار مهم خواهد بود مثلاً اگر مقدار اوليه a=1 و خروجي fn(x) برابر 3 باشد و اين تابع مقدار a را به 2 تغيير دهد خروجي نهايي ميتواند موارد زير باشد خروجيهاي ممكن براي عبارت a*fun(x)+a به صورت زير مي باشد:

الف -محاسبه به ترتيب از چپ به راست:

1*3+2=5 ب- a فقط يك بار ارزيابي شود:

1*3+1=4 ج- تابع fun(x) قبل از ارزيابي a فراخواني گردد:

2*3+2=8

```
a =1
y=a*fun(x)+a int
  fun( int k){
    .
```

```
.
a ++;
.
.
Return 3;}
```

شرايط خطا:

شرايط خطا ممكن است در عمليات اوليه پديد آيد مانند سرريز، تقسيم برصفر.در چنين مواردي برنامه نويس ممكن است مجبور باشد ترتيب ارزيابي را دقيقاً كنترل كند مثلاً جهت جلوگيري از سرريز ممكن است عبارت a+b-c به صورت a-c+b محاسبه گردد.

عبارات بولين مدار 185185185185**كوتاه**F_2**:**

در ارزيابي مدار كوتاه عبارت، نتيجه عبارت بدون ارزيابي تمامي عملوندها و يا عملگرهاي آن تعيين ميشودبه عنوان مثال عبارت (b/31- 1)*(31*a) را در نظر بگيريد اگرa=0 باشد مقدار اين عبارت به(b/31- 1) بستگي ندارد يا از آن مستقل است يعني اين بخش از عبارت در مقدار عبارت تاثيري ندارد لذا نه تنها نيازي به ارزيابي اين بخش

Side effect [1] short-circuit Boolean expression [2]

نيست عملگر دوم نيز لازم نيست ايجاد شود اما تشخيص اين وضعيت در حين اجرا ساده نيست به همين دليل اغلباز ارزيابي مدار كوتاه استفاده نميشود .

: به مثال ديگر توجه

كنيد … `if((A==0) or (B/A>C) then` 1) مثال

بسياري از زبانها هر دو عملوند را ارزيابي ميكنند و اگر A= = 0 به خاطر B/A خطاي تقسيم بر صفر رخ خواهد داد ولي برخي از زبانهاي ديگر مانند C هنگامي كه عبارت سمت چپ (A= = 0درست باشدديگر عبارت سمت راست را محاسبه نكرده و جواب را True در نظر ميگيرند يعني مقدار عملوند سمت چپ ممكن است بقيه عبارت بولين را كوتاه (short- circuit)كند.درزبان ادا با دو عمليات and then و or else اين ارزيابي مدار كوتاه 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 ارزيابي ميگردد.

به مثال ديگري در اين زمينه توجه فرمائيد: مثال2) `while (I<UB) and (V[I]>0) do`

اگراين دو عبارت را همزمان ارزشيابي كنيد(هر دو عملوند I,UB) و I خارج از حد آرايه باشد خطا رخ خواهد داد.براي بر طرف كردن اين مشكل در صورتي كه عبارت اولي false باشد ارزيابي عبارت دوم تأ ثيري در نتيجه نخواهد داشت پس از مفهوم اتصال كوتاه استفاده ميكنيم و اصلاً عبارت سمت راست را ارزشيابي نميكنيم تا بخواهد خطايي رخ دهد.

`While (I<UB) and else (V [I]>0) do …`

انتساب:

هدف از دستور انتساب اين است كه مقدار راست عبارت (مقدارشي داده) را به مقدار چپ آن (محل حافظه) نسبت دهد.شكلهاي مختلف انتساب در زبانهاي گوناگون در جدول زير آورده شده است:

Pascal , Ada	A := B
Fortran , C , PL/I , ML , prolog	A = B
APL	A → B
COBOL	MOVE B TO A
LISP	(SETQ A B)

جدول 8 - 3

در زبان C ،انتساب يك عملگر است ولي اغلب انتساب به عنوان يك دستور محسوب ميشود انتساب در پاسكالمقدار بر نميگرداند ولي در C مقدار بر ميگرداند.اغلب زبانها فقط يك عملگر انتساب دارند ولي C چندين عملگر انتساب دارد.مفاهيم مختلف عملگر انتساب در زبان C در زير آورده شده است.

A=B: مقدار راست B را به مقدار چپ A نسبت ميدهد و مقدار راست A را بر ميگرداند.

(A-=B) A+=B: به اندازه B به A اضافه (كم) ميكند و مقدار جديد را بر ميگرداند.

(A--) A++: يك واحد به A اضافه (كم) ميكند و مقدار راست A را بر ميگرداند.

(--A) ++A: مقدار A را بر ميگرداند سپس يك واحد به آن اضافه (كم) ميكند .

8-4- كنترل ترتيب دستورات

سه شكل متفاوت جهت كنترل ترتيب جملات دستوري عبارتند از:

- 186186186186**تركيب**[F1]**:** دستورات پشت سر هم و به ترتيب اجرا ميشود مثل جملات بين begin, end در يك بلوك پاسكال
- 187187187187**انتخاب**[F2]: در اين دستورات دو يا چند مسير جهت اجرا وجود دارد مانند case,if
- 1111**تكرار**[F3]: دنباله اي از دستورات كه به صورت تكراري اجرا ميشوند مانند while- for

8-4- 1- دستور goto

دستور goto در زبانهاي اوليه بيشتر مورد استفاده قرار ميگرفت اما با ايجاد مفهوم برنامه نويسي ساخت يافته استفاده از آن محدود شده است و توصيه مي شود تا حد امكان از آن استفاده نشود زيرا برنامههايي كه تعداد زيادي دستور goto دارند شبيهspageti(ماكاروني) مي باشند و اشكل زدايي اين برنامهها با سختي انجام خواهد گرفت. دو نوع دستور goto داريم :

- **goto بدون شرط :** اين دستور كنترل را به يك خط خاص انتقال ميدهد(دستور بعد از goto به عنوان بخشي از دنباله اجرا نميشود)

```
Goto next;
```

- **goto شرطي :** در صورتي كه شرط ذكر شده درست باشد كنترل به دستور با برچسب Next منتقل ميشود.

```
If (A==0) then goto Next;
```

```
goto (10,20,30),x
```
) مثال

---

[1] composition

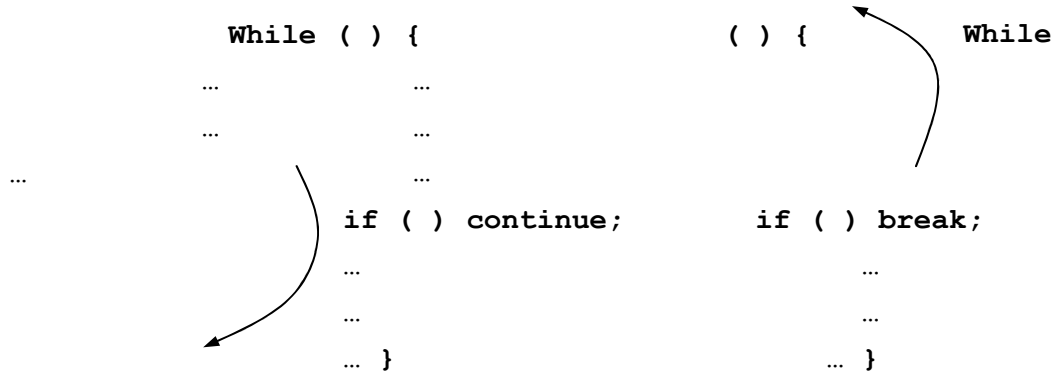
[2] alternation

[3] Iteration

اگر x<0 باشد كنترل به خط 10 و اگر x=0 كنترل به خط 02 و اگر x>0 باشد كنترل به خط 30 انتقال ميدهد .

در زبانهاي ساخت يافته امروزه توصيه اكيد آن است كه از اين دستور استفاده نشود حتي در برخي از زبانهاي جديد مثل ML دستور goto وجود ندارد.از زماني كه زبانها داراي ساختارهاي كنترلي

مثل if, while شدند دستور goto به طور كامل از دور خارج شد در بعضي از زبانها مانند فرترن و ML ،انتقال كنترل صحيح لازم است زيرا ساختارهاي كنترلي مناسب وجود ندارد اين كارتوسط دو دستور break, continue انجام ميشود.در برخي از زبانها مانند C, Pascal دو دستور break, continue وجود دارد كه break باعث اتمام حلقهها و continue باعث برگشت به اول حلقه ميگردد .



مزاياي دستور goto :

- مستقيماً توسط سخت افزار پشتيباني شده و اجراي آن بهينه مي باشد.
- كاربرد آن در برنامههاي كوچك ساده است.
- آشنا بودن برنامه نويسان با آن، مخصوصاً برنامه نويسان اسمبلي و زبانهاي قديمي.
- يك بلاك از برنامه با استفاده از goto ميتواند چندين هدف را سرويس دهد.

معايب دستور goto :

- مغايرت با ساختار سلسله مراتبي برنامه. طبق اصول برنامه نويسي ساخت يافته،ساختار برنامه بايد به صورت درختي باشد ولي استفاده از goto ممكن است اين ساختار را به شكل گرافي در آورده و لذا درك برنامه را پيچيده كند.
- مغايرت با اصل نوشتن ساختارهايي كه يك نقطه ورود و يك نقطه خروج دارند استفاده از دستور goto ممكن است باعث شود هر بلوك چند نقطه خروج داشته باشد.
- در برنامه نويسي بهتر است ترتيب اجرايي جملات با ترتيب فيزيكي آنها يكسان باشد كه استفاده از goto اين امكان را از بين ميبرد.
- با دستور goto ميتوان كاري كرد كه يك قسمت ازبرنامه به عنوان ادامه چند قسمت ديگر مورد استفاده قرار گيرد و اين موضوع درك برنامهها را مشكل ميسازد .

ويژگيهاي برنامه نويسي ساخت يافته:

- بر طراحي سلسله مراتبي[1] ساختارهاي برنامه با استفاده از شكلهاي كنترلي ساده مثل تركيب،انتخاب،تكرار تأكيد دارد.
- بر متني از برنامه تأكيد دارد كه ترتيب فيزيكي دستورات همان ترتيب اجرا باشد.
- بر استفاده از گروههايي با يك هدف تأكيد دارد حتي اگر مستلزم كپي كردن دستورات باشد.
- درك،اشكال زدايي،كنترل، تصحيح و نگهداري برنامههاي ساخت يافته آسان است.

همانطور كه قبلاً گفتيم جملات دستوري در برنامههاي ساخت يافته سه نوع تركيب،انتخاب،تكرار هستند. يك ويژگي مهم اين دستور اين است كه همة آنها يك نقطه ورود و يك نقطه خروج دارند. در ادامه هريك از موارد مذكور را دقيق تر بررسي خواهيم كرد.

8-4- 2- دستورات تركيب

دنباله اي از دستورات هستندكه در ساختار ديگر دستورات به عنوان يك دستور به حساب ميآيند مثلاً در پاسكال دستورات مركب به صورت Begin…End مي باشند و در java, C++,C و پرل به صورت {.....} نوشته مي -شود .

8-4- 3- دستورات شرطي

متداول ترين اين دستورات If, case ميباشد.با Ifهاي تودرتو يا نردباني ميتوان حالتهاي متعددي را بررسي كرد. برخي ساختارهاي Ifهاي تودرتو را ميتوان به صورت ساده تري با case نوشت.چند مثال از دستورات شرطي در زبان Ada در زير آورده شده است:

```
If condition then statement endif( تك شرطي )
If condition then ( دو شرطي )
   Statement 1
                  Else
   Statement 2
Endif If condition 1 then
                  (چند شرطي )
   Statement 1
 Elseif condition 2 then
   Statement 2
                  ...
                  Else
   Statement n
```

```
endif
```

herarchical [1]

نمونه اي از دستورات IF تودرتو(نردباني) در زبان Ada در زير 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 اينكه Tag:0..5 باشد معادل دستور Case عبارت فوق به صورت زير مي 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
```

پياده سازي:

دستورات if با دستورات پرش سخت افزاري پياده سازي مي شوند ولي دستورات case اغلب به كمك جدول پرش[1] پياده سازي ميشوند تا به اين ترتيب از تستهاي تكراري يك متغير جلوگيري شود ولي كارايي اجرا بالا مي-رود .جدول پرش برداري است كه به صورت ترتيبي در حافظه ذخيره شده و هر يك از عناصر آن يك دستور پرش غير شرطي است ابتدا عبارتي كه شرط دستور case را پديد

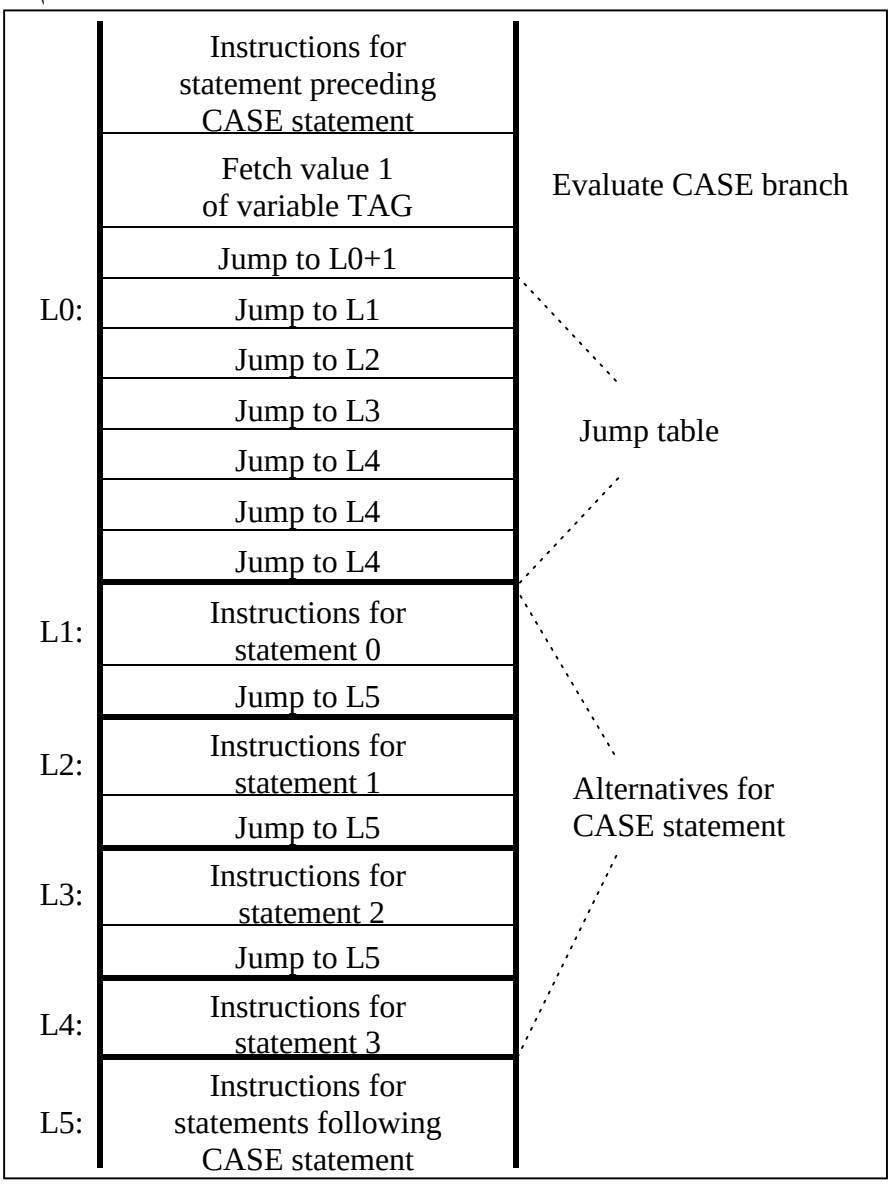
آورده ارزيابي شده و نتيجه آن به يك مقدار صحيح تبديل ميشود كه آفستي را در جدول پرش نشان ميدهد .

نكته: به طور كلي هزينه ترجمهcase بيشتر از If تودرتو است (به دليل ساختن جدول پرش) ولي هزينه اجراي case به مراتب كمتر ازIfهاي تودرتو است. هزينه اجراي دستور case ,O(1) و هزينه اجراي Ifهاي تودرتوO(n) مي باشد.

پياده سازي دستور Case مثال قبل به شكل زير مي باشد:

---

[1] Jump table



شكل 8 - 4

نكته:استفاده از ساختار case به شكل فوق نسبت به ساختارifهاي تودرتو , زمان كامپايل كردن را بيشتر ميكند ولي از طرف ديگر سرعت اجراي برنامه را افزايش ميدهد. چرا كه در ساختار جدول پرش فقط يك مقايسه لازم است ولي در ifهاي تودرتو شرطهاي متعددي بايد آزمايش شوند.

8-4- 4- دستورات تكرار

دستورات تكرار از يك راس (head) و يك بدنه (body)تشكيل يافته اند. راس تعداد دفعاتي كه بدنه بايد اجرا شود را تعيين ميكند .ساختارهاي راس عبارتند از :

- **تكرار ساده:** كه تعداد دفعات تكرار بدنه را به سادگي و با صراحت تعيين ميكند مانند نمونه زير در كوبول كه با دستور perform انجام ميگيرد:

```
Perform body k times
```

اين دستورbody را k بار تكرار ميكند.

چند سوال مطرح است:

آيا k تنها يك بار ارزشيابي ميشود يا در حين اجرا ي bodyمقدار جديدي خواهد گرفت؟ اگر kصفر يا منفي باشد آياbody يكبار اجرا ميشود يا اصلا اجرا نميشود؟

- **تكرار تا هنگامي كه شرطي برقرار باشد:** مانند حلقههاي while.

```
While test do body
```

- **تكرار با تغيير يك شمارنده:** مانند حلقههاي for مثال زير در الگول:

```
For I=1 step 2 until 30 do body
```

- **تكرار نا متناهي:** اغلب در مواردي استفاده ميشود كه شرط خروج از حلقه پيچيده بوده و نميتواند در راس حلقه بيان شود. مثل نمونه زير درAda :

```
Loop
  …
    Exit when condition;
  …
End
```

loop;و نمونه زير در پاسكال:

```
While true do begin … end
```

- **تكرار مبتني بر دادهها:** گاهي فرمت دادهها، شمارنده تكرار را مشخص ميكند , مانند دستور foreachدر پرل :

```
Foreach $x (@ arrayitem) {…}
```

در هر گذر از حلقه تكرار , متغير اسكالرx$ برابر با عنصر بعدي آرايه arrayitem @ است.

اندازه آرايه تعداد دفعات تكرار حلقه را مشخص ميكند .

نكته: حلقه for در زبان C بسيارانعطاف پذير است بطوريكه با حلقه forدر زبان C تمامي حالات فوق را ميتوان پياده سازي كرد.

```
For ( exp1; exp2; exp3) {body}
```

شمارنده از 1 تا 01: `For ( i=1; i<=10; i++) {body}`

حلقه نا متناهي: `For (;;) {body}`

شمارنده با شرط خروج : `For ( i=1 ; i<=100 && neof ; i++) {body}`

پياده سازي دستورات كنترل حلقه به سادگي با دستورات پرش سخت افزاري انجام پذير است.

مشكلات كنترل ترتيب دستورات ساخت يافته:

خروج چند گانه از حلقه

تكرار قسمتي از دستورات -do)

while-do) شرايط استثنايي

اگر چه هر ساختار كنترل ترتيب را ميتوان با استفاده از دستورات ساخت يافته بيان كرد ولي استفاده از دستور goto در شرايطي اجتناب نا پذير است كه اين شرايط 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;
```

اين حلقه , حلقه جستجو نام دارد كه در آن برداري از عناصر جستجو ميشوند تا اولين عنصري پيدا شود كه در شرايط خاص صدق كند.(يا حلقه خاتمه يابد, يا به عنصر مورد نظر برسيم.)

ب- do-while-do(تكرار قسمتي از دستورات):بعضي اوقات مناسب ترين مكان براي تست خروج از حلقه, نه در اول و نه در آخر حلقه, بلكه در وسط حلقه است. به اين ساختار گاهي اوقات -do while-do ميگويند كه متاسفانه هيچ زبان متداولي آن را پياده سازي نكرده است. البته ساختار break;(شرط) if در زبان Cو يا دستور exit whenدر زبان Ada به اين ساختار نزديك هستند.

```
Loop
Read(x);
```

```
If (x=0) then goto a (a is outside loop)
                Procrss(x);
                end loop;
```

در برنامه بالا يك حلقه تكرار وجود دارد كه در همه حالات كل بدنه براي همه دفعات به غير از دفعه آخر كه از حلقه خارج ميشويم اجرا ميگردد در اين حالت هم استفاده از goto اجتناب ناپذير است.

ج-شرايط استثنايي:

شرايط استثنا كه در حين اجراي برنامه پيش ميآيد مانند تقسيم بر صفر, over flow , under flow و .... با استفاده از دستور goto كنترل برنامه به قسمتي منتقل ميشود كه به آن راه انداز استثنا[1] گويند. وظيفه اين قطعه كد پيگيري اجراي برنامه به همراه پيغامهاي مناسب است.

[1] exception handler

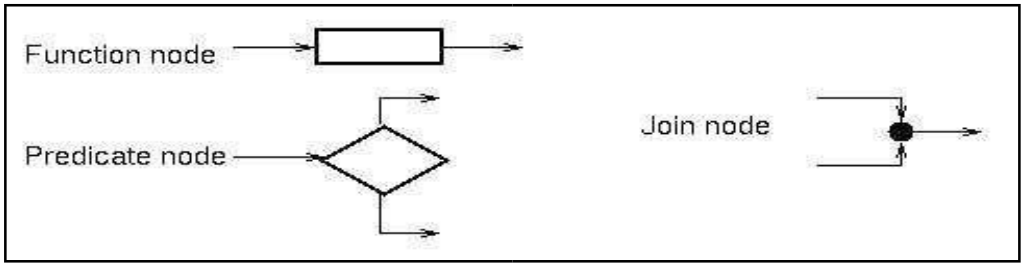
نكته:

يكي از ابهامات حلقه for آن است كه اگر مقدار نهايي حلقه، در بدنه حلقه تغيير كند چه ميشود؟ در زبان پاسكال مقدار نهايي حلقه فقط يكبار و قبل از ورود به حلقه محاسبه ميشود .لذا حلقه زير در زبان پاسكال 02 بار اجرا ميشود. ولي در زبان C تغيير مقدار نهايي حلقه درون بدنه، روي تعداد تكرار اثر ميگذارد و برنامه زير در C در حلقه دائم ميافتد.

```
n:=20; For i:=1
  to n do Begin
   n:=n+1; end
```

5-8- برنامههاي بنيادي[1]

عموماً فلوچارتهاي برنامهها شامل 3 دسته گره اصلي زير هستند:



شكل 8 - 5

گره تابع(function node): گره تابع يك دستور انتساب را نشان ميدهد و منجر به تغيير حالت ماشين مجازي ميشود.

برنامه 193193193193محضF_2:

برنامه اي است به صورت فلوچارت كه مدل رسمي يك ساختار كنترلي است و شامل ويژگيهاي زير است :

فقط يك كمان ورودي دارد.

فقط يك كمان خروجي دارد.

مسيري از كمان ورودي به هر كمان و از هر كمان به كمان خروجي وجود دارد.

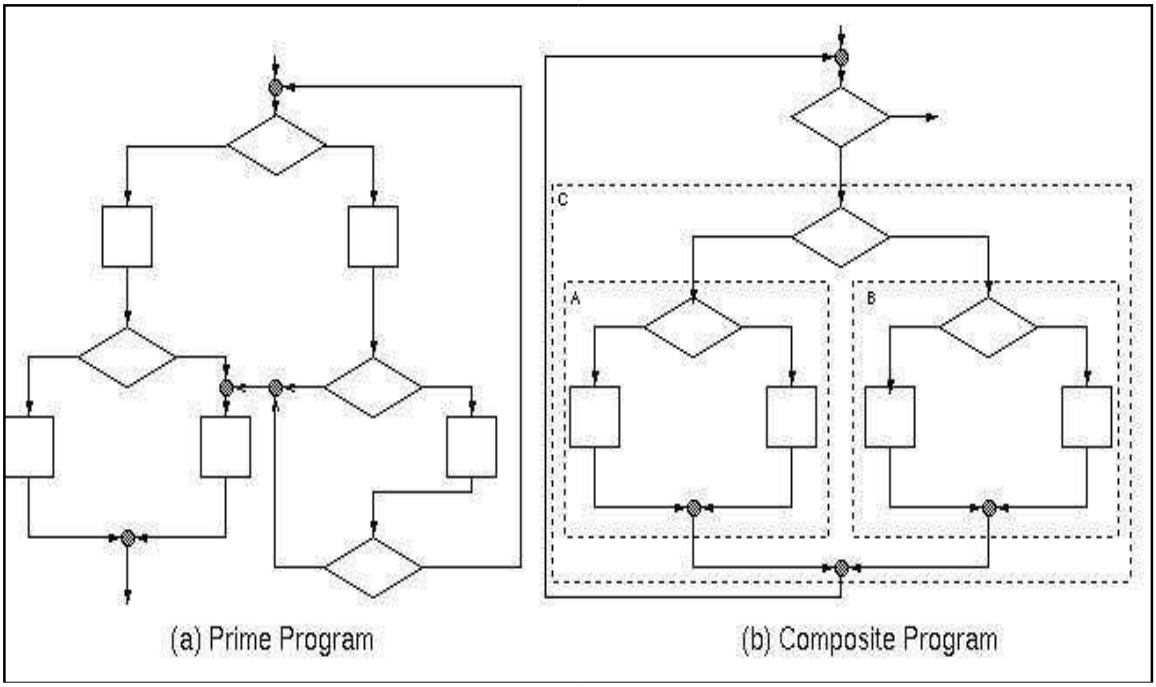
برنامه بنيادي:

يك برنامه محض است كه نميتواند به برنامههاي محض كوچكتري تقسيم شود.براي درك بهتر تعاريف فوق شكل زير را مشاهده كنيد:

---

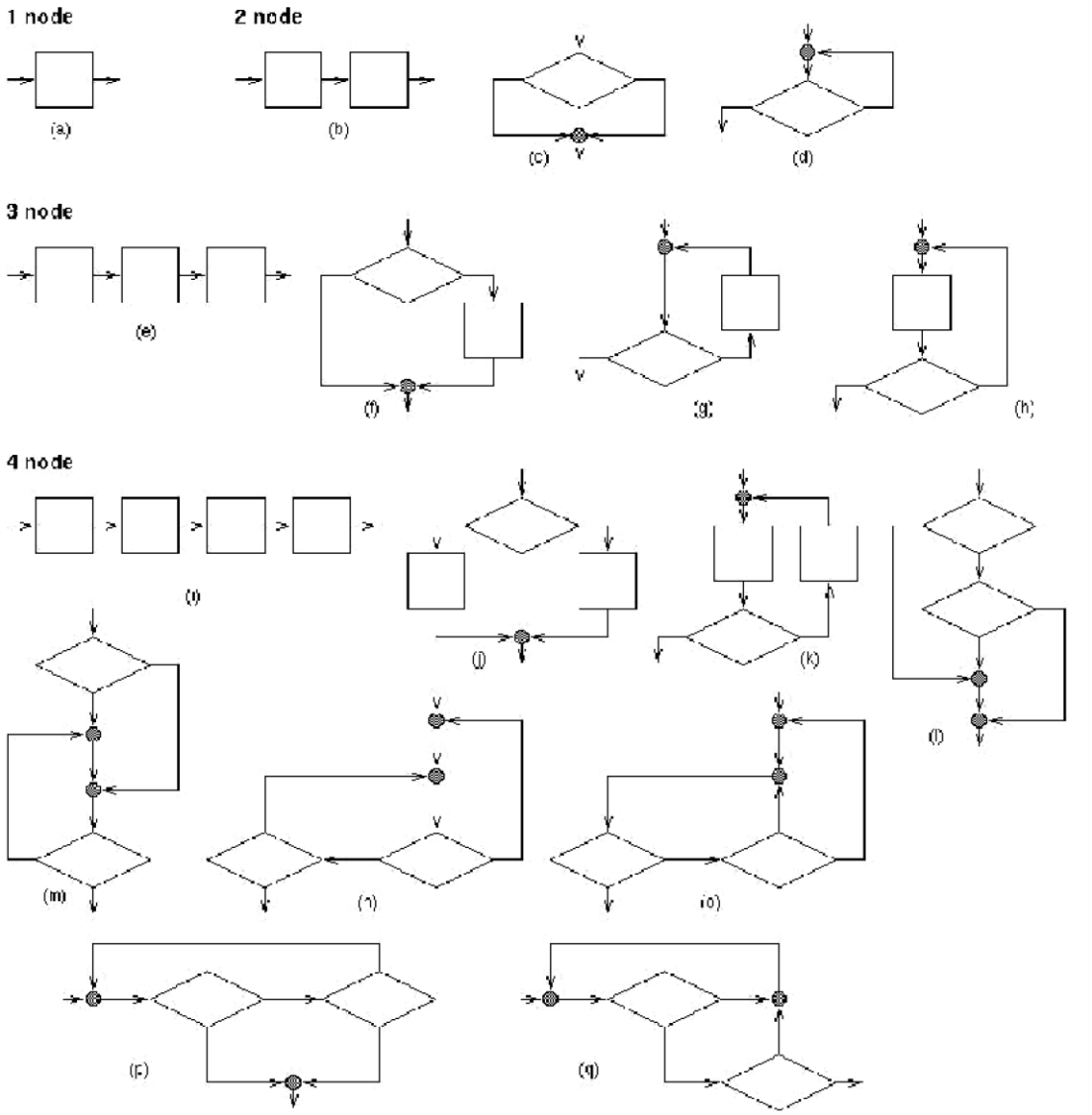
[1] prime

[2] proper



شكل 8 - 6

اگر نتوانيم دو كمان از برنامه محض را برش بزنيم تا برنامه محض را به گرافهاي جداگانه تقسيم كنيم برنامه محض , برنامه بنيادي خواهد بود. شكل(a) يك برنامه بنيادي است در حاليكه شكل (b) بنيادي نيست. قسمت -هاي نقطه چين A,B,C زير برنامههاي محض هستند.برنامه مركب يك برنامه محض است كه بنيادي نميباشد. شكل (b) مركب است ولي اگر در همين شكل (b) به جاي قسمتهاي نقطه چين, گره تابع قرار دهيم به برنامه بنيادي تبديل ميشود .برنامههاي بنيادي شمارشي هستند. شكل زير تمام برنامههاي بنيادي تا چهار گره را نشان ميدهد .



شكل ۸ – ۷

برنامههاي بنيادي ساختارهاي كنترلي معروف را نشان مي دهند. شكل (f) ساختار كنترلي-if then،شكل (g) ساختار كنترلي while ، شكل(h) ساختار كنترلي Repet-until ، شكل (j) ساختار كنترليIF-THEN-ELSE ، شكل (k) ساختار كنترلي do-while-do را نشان مي دهد.

قضيه ساخت يافته :

باهوم و جاكوبيني نشان دادند كه هر برنامه بنيادي را ميتوان به برنامه اي تبديل كرد كه فقط از دستورات if و while استفاده كند نتيجه كار باهوم جاكوبيني نشان داد كه لازم نيست از goto پرهيز كرد ميتوان الگوريتم هر برنامه اي را به صورت با goto ويا بدون آن نوشت. سپس با استفاده از قضيه ساخت يافته آن را به برنامه ساخت يافته تبديل كرد برنامه نويسي ساخت يافته مترادف برنامه

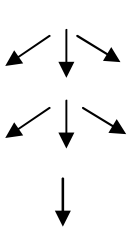
نويسي خوب نيست بلكه به معناي استفاده از ساختارهاي كنترلي است كه با تعداد كمي از گرههاي بنيادي هستند.

6-8- كنترل ترتيب در عبارات غير محاسباتي

يك عمليات حياتي در زبانهايي مثل , ML snobol4 و پرولوگ تطابق الگو مي باشد. در اين حالت، يك عمليات با تطابق و انتساب مجموعه اي از متغيرها به الگوي از پيش تعيين شده، انجام ميشود .

مثال - گرامر زير در الفباي 0,1 رشتههاي متفاوت با طول فرد را تشخيص ميدهد. فرض كنيد بخواهيم توسط اين گرامر رشته 00100 شناسايي وتوليد شود.

$A \rightarrow 0A0 \mid 1A1 \mid 0 \mid 1$



شكل 8 - 8

زبان snobol4 مخصوص تطابق الگو است كه كد برنامه اين زبان براي شناسايي اين رشته به شكل زير است:

$mathcesthecenter1 \square\square A_1$

$mathces\ 0\ A_1 0 \quad \square A_2$

$\square$

$mathces\ 0\ A_2 0 \quad \square A_3$

Snobol4 :

اسنوبال 4 زباني است كه براي شبيه سازي ويژگي تطابق الگو به وجود آمده است پياده سازي زبان اسنوبال 4 مستقل از معماري ماشين ميباشد به همين دليل اسنوبال 4 يكي از اولين زبانهايي بود كه اولاً تقريباً در همه ماشيني وجود داشت و ثانياً معناي آن تقريباً در هر پياده سازي يكسان بود.

پرولوگ :

بانك اطلاعاتي پرولوگ شامل حقايق[1] و قواعد[2] ميباشد. عبارتي كه حاوي يك يا چند متغير باشد يك تقاضا[3] نام دارد و رابطه اي ناشناخته را نشان ميدهد .براي درك اين سه مفهوم به مثال توجه كنيد:

```
Parent Of(John,Mary)

Parent Of(Susan,Mary)

Parent Of(Bill, John)

Parent Of(ANN,John)
```

Facts [1]
Rules [2]
Query [3]

با توجه به حقايق موجود در پايگاه دانش فوق ،(Parent (X, Mary يك تقاضا ميباشد.و قاعده زير در اين پايگاه دانش برقرار است:

```
Grand Parent of (x,y) : -Parent of (x,y),Parent of
(y,z)
```

ويژگي اصلي پرولوگ استفاده از تطابق الگوست تا مشخص شود كه آيا تقاضا توسط حقايق موجود در بانك اطلاعاتي قابل حل است يا خير. آيا ميتوان با استفاده از قواعدي در بانك اطلاعاتي(پايگاه دانش) كه برروي حقايق يا قواعد ديگر عمل ميشود حقيقت مورد نظر را بدست آورد يا خير.
پرولوگ براي تطابق الگو از اتحاد يا جايگزيني استفاده ميكند تا تعيين كند كه آيا تقاضا شامل جانشين معتبري با حقايق و قواعد موجود در بانك اطلاعاتي مي باشد يا خير.

197197197اتحادF_1(يكسان سازي):

به عمل جايگزيني متغيرها به جاي متغيرها , توابع به جاي متغيرها يا ثابتها به جاي متغيرها به طوريكه گزارههاي يكسان توليد شود يكسان سازي گفته ميشود .

مثال:

```
Parent of (x,mary)=parent of (john,y)
```

يكسان ساز **θ= {x/john,**

θ **Unify(p,d)=θ , subst(θ,p)=subst(θ,d)** به عنوان **y/mary}**

يكسان ساز دو جمله p,d خواهد بود كه به هر دو گزاره اعمال ميشود تا دو گزاره p,d يكسان شوند.

θ	d	p
{x/jane}	Knows(john,jane)	Knows(john,x)
{x/oj , y/john}	Knows(y,oj)	Knows(john,x)
{y/john ,x/mary(john)}	Knows(y,mary(y))	Knows(john,x)
(چون xنميتواند در يك لحظه دو مقدار بگيرد.) Fail	Knows(x,oj)	Knows(john,x)

چند نمونه تست:

1 - براي انتخاب از بين 01 دستور براي اجرا ، يكبار از دستورات تودرتوي (If-Then-Else) استفاده كرديم و يكبار از دستور Case با توجه به اينكه كامپايلر مورد استفاده ، Case را با استفاده از روش پرش پياده سازي كرده است.

هزينه اجرا وترجمه دو روش را مقايسه كنيد؟(مهندسي كامپيوتر -57) الف) هزينه اجراي Case بيشتر و هزينه ترجمه آن كمتر است.

ب)هزينه اجرا وترجمه Case ازIf-Then-Else بيشتر است.

---

[1] Unification

ج) هزينه اجرا وترجمه Case ازIf-Then-Else كمتر است.

د)هزينه اجراي Case كمتر وهزينه ترجمه آن بيشتر است.

جواب :

گزينه 4 ، در هنگام اجراي دستور Case با استفاده از جدول پرش ، فقط يك پرش غير مستقيم انجام ميشود و سپس آدرس دستورات از رديفهاي مربوطه در جدول با يك تفاوت مكان بدست ميآيد براي دستور If-Then-Else بايد شرطهاي زيادي در زمان اجرا چك شود كه هزينه اجراي آن را بالا ميبرد. در دستور Case تشكيل جدول پرش در زمان ترجمه هزينه بيشتري را براي كامپايلر در بر خواهد داشت.

2 - براي ساختن يك درخت تصميم گيري (Decision Tree) به منظور انتخاب يك دستور (Statement) از ميان n دستور، كدام يك از ساختارهاي برنامه سازي زير ميتواند منجر به كارايي اجرا به صورت O(1) شوند ؟ (فرض كنيد شرط انتخاب يك دستور ، برابر يك مقدار ثابت يك ميباشد)(مهندسي كامپيوتر- 38) الف)ساختن درخت با دستور If-Then-Else ب) بدست آوردن O(1) امكان ندارد.

ج) ساختار درخت با دستور Case يا Switch د) ساختار دستور با استفاده از If تودرتو در Ada **جواب :**

گزينه ج، ساختار If-Then-Else تودرتو باعث ميگردد كه يك متغير چندين بار تست شود ولي ساختار Case يا Switch به دليل استفاده از جدول پرش(jump table) يك بار متغير را در O(1) تست ميكند و با اين مقايسه ما آدرس محل پرش مشخص ميشود .

3 - حلقه زير را درنظر بگيريد. اگر فقط Step و مقدار نهايي حلقه هر بار ارزشيابي ميشوند، خروجي اين كد چيست؟ (مهندسي كامپيوتر –27)

```
                k:=1;
l:=5; for i:=k to 2×l
                step k
        print I;
              l:=l-1;
      k:=k+1; end
                for
```

الف) 3 1 ب) 2 1 ج) 3 1 د) 3 1

7 3 5 6

جواب : گزينه د، حلقه داده شده معادل for مقابل در C ميباشد.

```
For (i:=1 ; i<=2×L;i=i+k){
                Cout <<i
                  L--;
                 K++;}
```

I	K	L	خروجي
1	1	5	1
3	2	4	3
6	3	3	6
10	4	2	؟

8-7- سوالات فصل هشتم

سوالات تستي

1- كداميك از موارد زير براي كنترل ترتيب سطح دستور استفاده نمي شود؟(نيمسال اول 58 -68)
الف. پرش ب. تركيب ج. انتخاب د.تكرار

2- در مورد APL كدام گزينه غلط مي باشد؟(نيمسال دوم 58-68)
الف.توسط كن آي ورسون در اوايل دهه 1970 طراحي شد. ب .بر پردازش آرايه تاكيد دارد.
ج.دستورات را از راست به چپ اجرا مي كند. د .زبان كوچكي است.

3- براي كنترل ترتيب در عبارات غير محاسباتي از چه روشي استفاده مي شود؟(نيمسال اول 68 - 78) الف.نمايش درختي ب.انتساب به اشياي داده ج.تطابق الگو
د.هيچكدام

4- در صورتي كه fun(x) مقدار 3 را برگرداند و مقدار a را به 2 تغيير دهد ولي مقدار اوليه a برابر يك باشد بسته به ترتيب ارزيابيها براي عبارت a*fun(x)+a چند مقدار متمايز مي تواند حاصل شود؟(نيمسال دوم 68-78) الف3. ب2. ج1.
د4.

5- ارزيابي ميان بر براي كداميك از عملگرهاي منطقي زير به كار ميرود؟(نيمسال دوم 68 -78)
د.هيچكدام.OR, AND ج.OR ب.AND الف

6- برنامه نويسي در كداميك از زبانهاي زير برنامه نويسي اعلاني است؟(نيمسال دوم 68-78)
د.Prolog Lisp. ج.Ada ب.C الف

7- كداميك از موارد زير از خواص زبان فورث مي باشد؟(نيمسال اول 78 -88) **مورد اول:** براي كامپيوترهاي كنترل فرايند بي درنگ ,كاربرد دارد.

مورد دوم: نحو آن postfix محض است.

مورد سوم: ترجمه آن خيلي دشوار مي باشد.

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

8- در صورتي كه fun(x) مقدار 3 را برگرداند و قبل از برگشت مقدار a را به 2 تغيير دهد با فرض مقدار اوليه a برابر 1 بسته به ترتيب ارزيابيها براي عبارت a*fun(x)+a چند مقدار متمايز مي تواند حاصل شود؟(نيمسال اول 87-88)

الف3. ب2. ج1. د4.

9- در زبان C با فرض a=12, b=15 در كداميك از شرايط زير قانون مدار كوتاه در ارزيابي صورت مي گيرد؟(نيمسال اول 78 -88)

: if ((a>b)||(b<5)){... } **شرط اول**

شرط دوم if ((a>b)&&(b.5)){… } :

شرط سومif((a<b)||(b>5)){… } :

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

01- در كداميك از زبانهاي زير دستور go to براي كنترل ترتيب ضمني وجود ندارد؟(نيمسال اول 78-88)

الف ب.++C ج.Pascal د.C ML.

11- گزاره درست را انتخاب كنيد. (نيمسال اول 78 -88)

الف .هر برنامه prime مي تواند به برنامه اي تبديل شود كه فقط از دستورات while و if استفاده كند.

ب.نمي توان هر برنامه اي را با استفاده از قضيه ساختيافته به برنامه ساختيافته تبديل كرد.

ج.مي توان فقط هر برنامه اي بدون go to را با استفاده از قضيه ساخت يافته به برنامه ساختيافته تبديل كرد.

د.الف و ب صحيح است.

21- بانك اطلاعات زبان Prolog شامل كداميمك از موارد زير است؟(نيمسال اول 78-88) الف.حقايق و سوالات ب.حقايق و قواعد

ج.قواعد و سوالات د.قواعد و استنتاج

31- كداميك از مواردذيل از خواص زبان فورث است؟(نيمسال دوم 78 -88) **مورد اول:** براي كامپيوترهاي كنترل فرايند بي درنگ كاربرد دارد.

مورد دوم: نحو آن از postfix محض است.

مورد سوم: ترجمه آن خيلي دشوار است.

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

41- در زبان C با فرض a=12,b=15 در كداميك از شرطهاي زير قانون مدار كوتاه در ارزيابي صورت ميگيرد؟(نيمسال دوم 78 -88)

شرط اول if ((b<20)||(a<5))(…) :

شرط دوم if((a<b)&&(b>5))(…) :

شرط سومif((a<b)||(b>5))(…) :

الف:اول و دوم ب:اول وسوم ج:دوم و سوم د:هر سه شرط

51- جدول پرش براي پياده سازي كداميك از ساختارهاي زير به كار ميرود؟(نيمسال دوم 78 -88)

الف Record .ب Function .ج Case.د For.

61- خروجي برنامه زير در زبان C كدام است؟(نيمسال اول 88-98)

```
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; .ب I=1; j=1;.ج I=1;j=2;.د I=1;j=2;.

I=2; j=2 ; I=2; j=2 ; I=1; j=2; I=2; j=2;

I=7; j=2; I=7; j=7; I=7; j=7; I=7, j=2;

71- مقصود از هم ارزي ساختاري براي دو نوع داده چيست؟(نيمسال اول 88 -98)

الف.دو نوع داده هم ارز هستند اگر اشياي داده آنها عناصر خارجي يكسان داشته باشند.

ب.دو نوع داده هم ارز هستند اگر اشيا داده آنها عناصر داخلي يكساني داشته باشند.

ج.دو نوع داده هم ارز هستند اگر صرفاً نامهاي يكساني و مشابه به يكديگر داشته باشند.

د.دو نوع داده هم ارز هستند اگر نامهاي يكسان و اشياي خارجي متفاوت داشته باشند.

81- در كداميك از حالتهاي زير ارزيابي نمايش درختي عبارات به صورت عجولانه صورت مي گيرد؟(نيمسال اول 88-89)

الف. براي هر گره عملياتي ابتدا عملگرها وبعد عملوندها ارزيابي مي شوند.

ب .براي هر گره عملياتي عملوندها و عمليات با هم ارزيابي مي شوند.

ج .براي هر گره عملياتي ابتدا عملوندها و سپس عمليات ارزيابي مي شوند.

د .براي هر گره عملياتي عملوندها قبل از اجراي عمليات ارزيابي نمي شوند.

91- كداميك از گزينههاي زير صحيح مي باشد؟(نيمسال اول 88-98)

الف. امروزه از دستور go to زياد استفاده مي شود .

ب.استفاده از دستور go to فقط به صورت شرطي امكان پذير است.

ج.استفاده از دستور go to فقط به صورت غير شرطي امكان پذير است.

د .استفاده از دستورgo to برنامهها را از حالت ساختيافته خارج مي كند.

02- كداميك از ساختارهاي كنترلي زير با فرض وجود برچسبهاي ساده در زبانهاي برنامه سازي، بطور ضمني توسط معماري سخت افزاري پشتيباني مي شود؟ (نيمسال دوم 88 -98)

الف case .ب if .ج go to .د While.

21- با توجه به مجموعه کد زیر کدام يك از مسائل ترتیب ارزیابي در هنگام تولید کد قابل اعمال است. (نیمسال اول 98-09)

```
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 جانبي

8-8- پاسخنامه سوالات تستي فصل هشتم

سوال	الف	ب	ج	د
1	*			
2	*			
3			*	
4	*			
5			*	
6			*	
7	*			
8	*			
9			*	
10				*
11	*			
12		*		
13	*			
14		*		
15			*	
16				*
17		*		
18			*	
19				*
20	*			

21		*		

سوالات تشريحي

1- براي دستور Case زير جدول پرش را رسم نمائيد؟(نيمسال اول 58-68)

```
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;
```

2- با توجه به مفاهيم ساختارهاي كنترلي ترتيب اجرا و نمايش حافظه به ازاي قطعه برنامه زير چگونه است؟(نيمسال اول 68 -78)

```
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
```

3- ساختار case زير را به روش جدو.ل پرش پياده سازي كنيد؟(نيمسال دوم 68 -78)

```
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;
```

4- برنامه پریم را تعریف کرده و تمامي برنامههاي Prime با سه گره را رسم کنید؟(نیمسال دوم 78-88) **5-** براي زبان Prolog مفاهیم زیررا به همراه مثال بطور کامل شرح دهید؟(نیمسال دوم 78-88) الف.حقایق(Fact) ب.اتحاد(Unification) ج.قواعد(Rule)

6- عبارت بولین مدار کوتاه (SHORT-CIRCUIT) را براي عملگرهاي OR,AND به همراه مثال شرح

دهید؟(نیمسال اول 88 -98) **7-** برنامه غیرساخت یافته در شبه زبان C++ را به کمك قضیه ساخت یافته اصلاح کنید.(نیمسال اول 98-09)

```
Cin>>x; if
(x==10)
x=x+1;

label: if (x==100)
x=x-1;
if (x==50)
goto label;
```

فصل نـهم:

کنترل ترتیب زیربرنـامه

ا # ا :

زيربرنامههاي فراخواني -
بازگشت زيربرنامههاي بازگشتي
محيط ارجاع ارجاع سراسري
ارجاع محلي ارجاع غير محلي
ارجاع از پيش تعريف شده
اعلان پيشرو در پاسكال نام مستعار
حوزه ايستا و پويا ساختار بلوكي
محيط ارجاع براي دادههاي محلي
نگهداري حذف پارامترها
فراخواني با نتيجه
فراخواني با مقدار-نتيجه فراخواني با
مقدار ثابت پياده سازي انتقال پارامتر
زير برنامه به عنوان پارامتر
محيطهاي مشترك صريح
پياده سازي حوزه ايستا و
پويا اعلانها در بلوكهاي
محلي سوالات تستي و
تشريحي
پارامترهاي واقعي و مجازي و
تناظر بين آنها روشهاي انتقال پارامتر
فراخواني با مقدار فراخواني با ارجاع
فراخواني با نام

9-1- مقدمه

در كنترل ترتيب زير برنامهها،فراخواني يك زير برنامه و يا زير برنامه ديگر و برگشت از زير برنامه مطرح است كه در اكثر زبانهاي برنامه نويسي به ترتيب با دستورهاي فراخواني(call) و بازگشت(return)انجام ميشود .ساده ترين نوع كنترل زيربرنامه داراي ساختاري به نام ساختار فراخواني- بازگشت[1] ميباشد اين ساختار كنترلي توسط قاعده كپي[2] بيان ميشود اثر دستور فراخواني زير برنامه مثل اين است كه قبل از اجرا، يك كپي از زير برنامه اي كه فراخواني شده است در نقطه اي كه فراخواني صورت ميگيرد قرار داده شود.اما در ديدگاه قاعده كپي پنج فرض از سوي طراحان زبان در نظر گرفته ميشود كه عبارتنداز:

- زير برنامهها نميتوانند بازگشتي باشند چون در فراخواني بازگشتي غيرمستقيم ميتوانيم با استفاده از قاعده كپي بدنه زير برنامه را در داخل زير برنامه كپي ميكنيم اما اگر زير برنامه بازگشتي مستقيم باشد اين كار امكان پذير نيست چون هر جايگزيني با وجود اينكه يك دستور فراخواني(call) را حذف ميكند ولي فراخواني جديدي به همان زير برنامه را معرفي ميكندكه براي آن نيز يك جايگزيني ديگر لازم است و..... اين روند ادامه خواهد داشت.
- نياز به دستور فراخواني صريح است اما در زيربرنامههاي مخصوص پردازش استثنا، هيچ فراخواني صريحي وجود ندارد.
- زير برنامهها در هر بار فراخواني بايد به طور كامل اجرا شوند اما در همروالها[3] ممكن است زير برنامه اي به طور كامل اجرا نشود.
- به محض اجراي Call(فراخواني)كنترل به زير برنامه داده ميشود و پس از اجراي زير برنامه كنترل به نقطه فراخواني برميگردد، اما براي فراخواني زير برنامههاي زمان بندي شده اجراي زير برنامه ممكن است مدتي به تعويق افتد.
- در هر زمان فقط يك زيربرنامه كنترل را در دست دارد، اگر اجرا در نقطه اي متوقف شد بقيه يا هنوز فراخواني نشده اند يا اجراي آنها كامل شده است اما زير برنامههايي كه به عنوان يك task مورد استفاده قرار ميگيرند ممكن است به طور همزمان اجرا شوند به طوريكه چندين زير برنامه در آن واحد در حال اجرا باشند يعني اگر يكي از زيربرنامهها متوقف شد ممكن است چندين زيربرنامه ديگر در حال اجرا باشند.

9-2- زيربرنامههاي فراخواني - بازگشت

همانطور كه قبلاً گفته شد بين تعريف زيربرنامه با سابقه فعاليت آن، تفاوت وجود دارد. تعريف آن چيزي است كه در برنامه مينويسيم و به يك قالب ترجمه ميشود اما سابقه فعاليت در هر بار فراخواني

زير برنامه با استفاده از قالبي كه از تعريف ايجاد شد به وجود ميآيد .سابقه فعاليت شامل دو بخش است يكي سگمنت كد كه حاوي كد اجرايي و ثابتها مي باشد و ديگري ركورد فعاليت كه شامل پارامترها،دادههاي محلي و ساير دادههاست .بخش كد در حين

call-return 1
copy rule 2
coroutines 3

اجرا تغيير نميكند و به صورت ايستا در حافظه قرار ميگيرد همه سابقههاي فعاليت زير برنامه از يك سگمنت كد استفاده ميكنند ولي ركورد فعاليت در هر بار اجراي زير برنامه ايجاد شده و با تمام شدن زير برنامه از بين ميرود .اينكه بگوييم»اجراي دستور ويژه كد از زير برنامه«دقيق نيست و بايد بگوييم »اجراي كد در حين فعالسازي R از زيربرنامه«. بنابرين براي تعيين نقطه اي كه برنامه از آنجا شروع ميشود به دو اشاره گر بيتي نياز داريم:

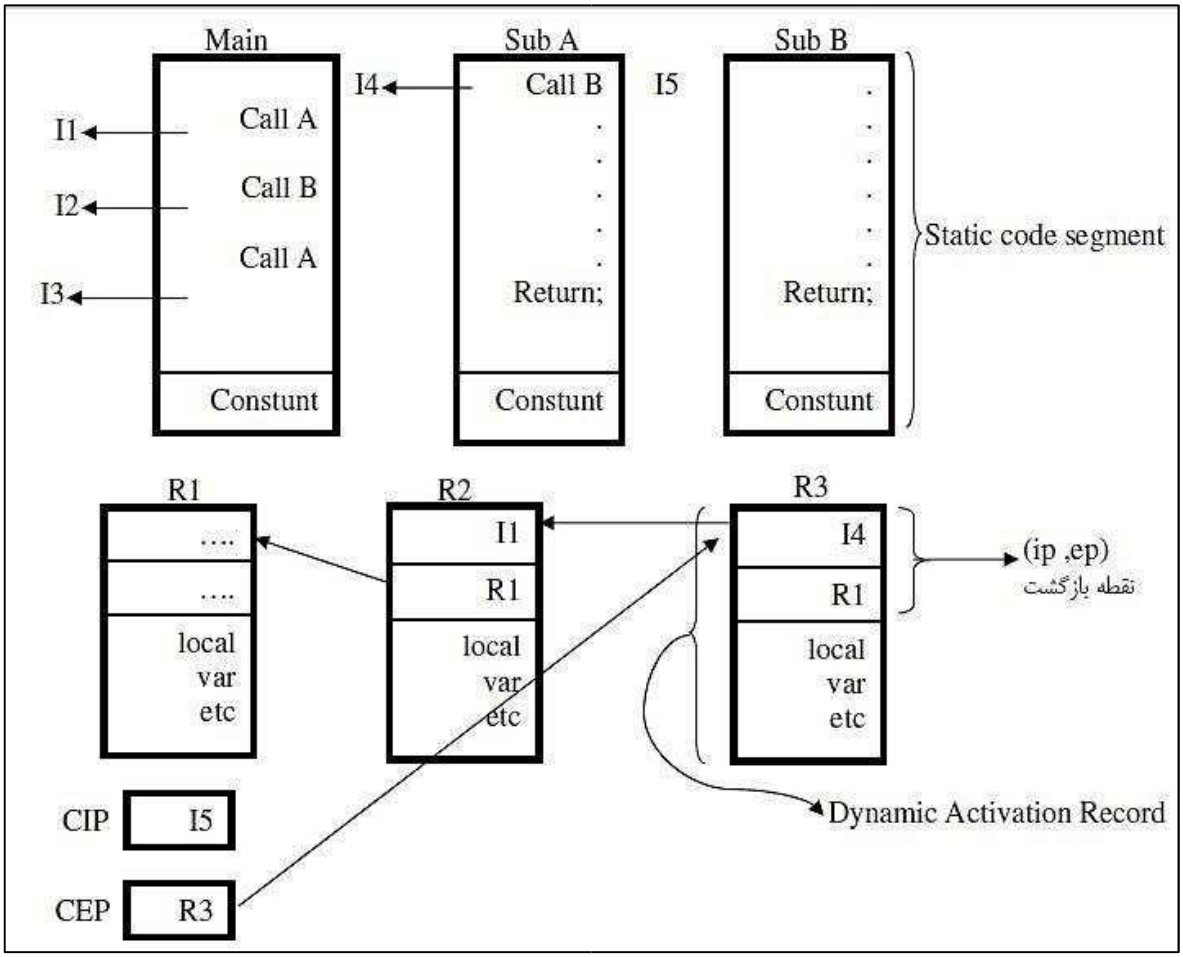
- **اشاره گر دستور فعلي(CIP[1]) :** كه به دستور داخل سگمنت كد كه قرار است اجرا شود اشاره ميكند و مترجم دستوري را كه CIP به آن اشاره ميكند بازيابي كرده و آن را اجرا ميكند .
- **اشاره گر محيط فعلي (CEP):**[2] اشاره گري است كه به ركورد فعاليت فعلي (مربوط به قطعه كدي كه در حال اجرا است)اشاره ميكند بنابراين چون تمام سابقههاي فعاليت يك زير برنامه از يك سگمنت كد استفاده ميكنند دانستن دستور فعلي كافي نيست بايد اشاره گر CEP هم باشد تا سابقه فعاليت مورد نظر را مشخص نمايد به عنوان مثال وقتي دستوري دركد،به متغيرx مراجعه ميكند آن متغير در ركورد فعاليت وجود دارد، هر ركورد فعاليت آن زيربرنامه ، شيء داده اي به نام x دارد كه ممكن است محتوياتش با ديگري فرق داشته باشد.

پياده سازي:

در زمان شروع اجراي برنامه اصلي، اشاره گر CEP به ركورد فعاليت برنامه اصلي اشاره ميكند و CIP به اولين دستور از سگمنت كد برنامه اصلي اشاره ميكند وقتي كنترل اجرا به دستور فراخواني زير برنامه رسيد يك ركورد فعاليت از آن زير برنامه ايجاد ميشود و CEP به آن اشاره ميكند و CIP به اولين دستور سگمنت كد زير برنامه اشاره خواهد كرد. جهت برگشت صحيح و درست از يك زير برنامه، مقادير CEP وCIP را ميتوان در ركورد فعاليت زير برنامه اي كه فراخواني شده است ذخيره كرد تا در زمان بازگشت از زير برنامه،اجرا از نقطه بعد از فراخواني زير برنامه از سر گرفته شود. شكل زير نمونه اي از عمليات فوق رابراي برنامه اصلي كه دو بار زير برنامه A و يك

بار زیر برنامهB را فراخواني ميكند نشان ميدهد .زیر برنامهA خودش یکبار زیر برنامهB را صدا میزند.

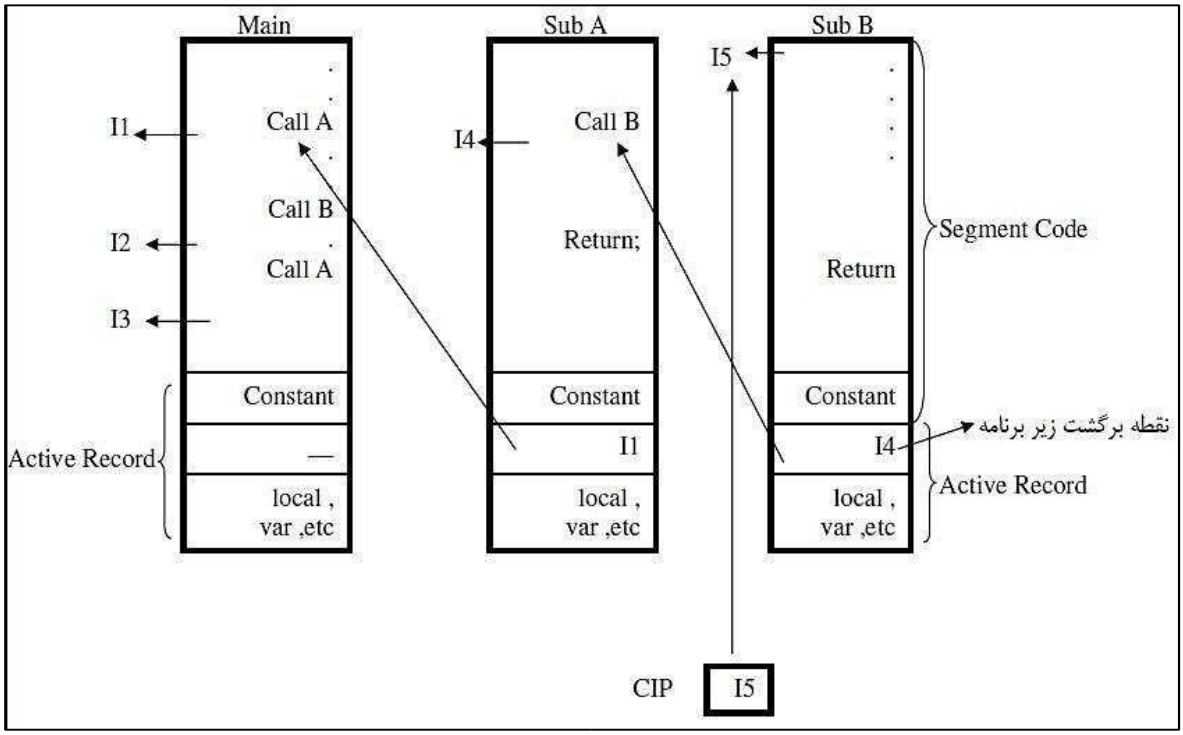
Current Instruction Pointer 1
Current Enviroment Pointer 2



شكل 9 - 1

پياده سازي ديگر:

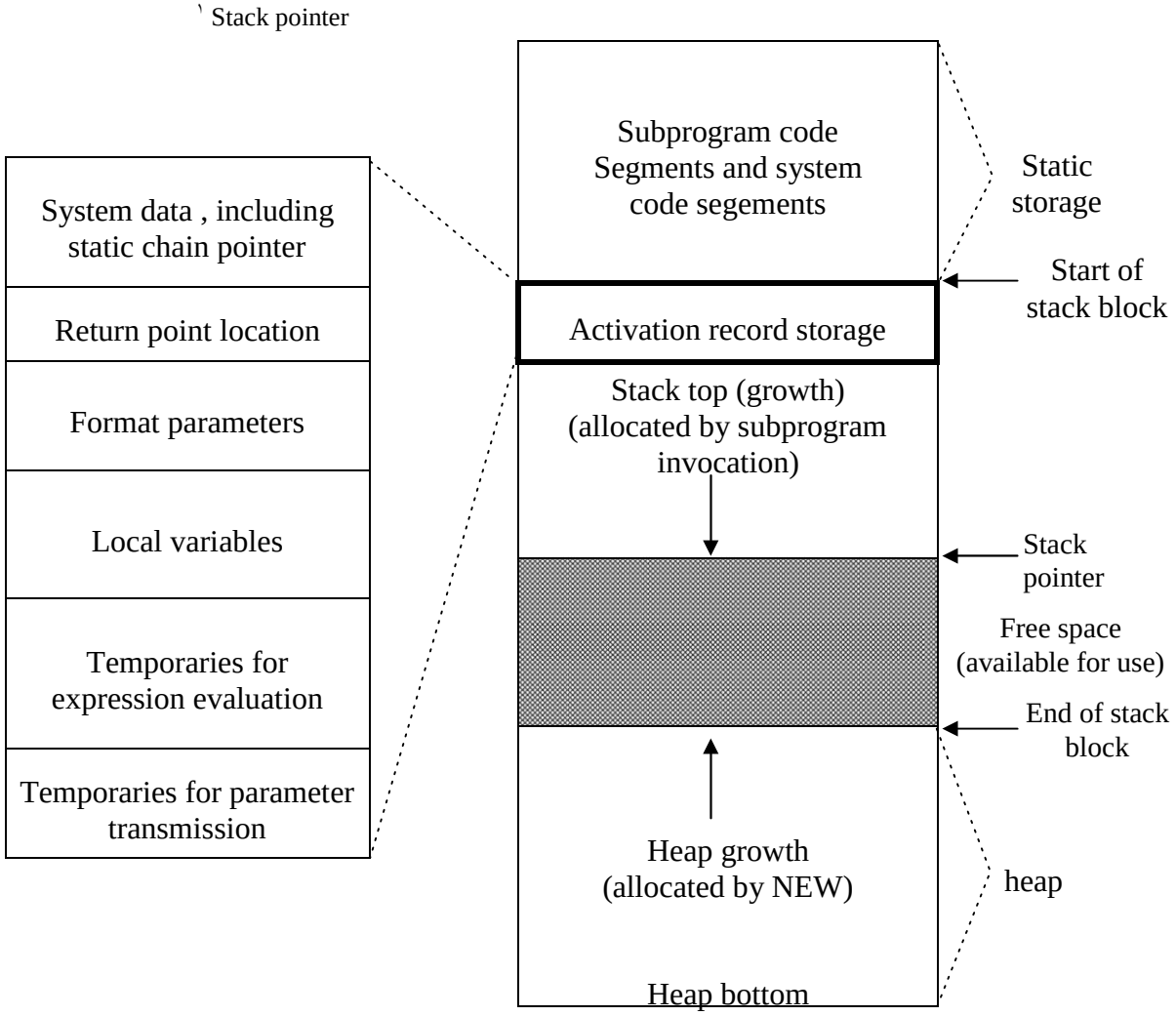
البته اگر جهت افزايش سرعت اجرا، حافظه زيادي در اختيار داشته باشيم ميتوان از مدل ساده تر پياده سازي زير استفاده كرد. در اين روش پياده سازي، براي ركورد فعاليت هر زير برنامه، حافظه به صورت ايستا تخصيص ميابد در اين مدل ساده تر كه در بسياري از نسخههاي فرترن و كوبول استفاده شده است اجراي كل برنامه با يك سگمنت كد و يك ركورد فعاليت براي هر زير برنامه و برنامه اصلي شروع ميشود .در حين اجراي برنامه، هنگام صدا زدن زير برنامه ديگر حافظه اي به صورت پويا تخصيص نميابد بلكه در هر بار فراخواني از همان ركورد فعاليت، استفاده مجدد ميشود .در اين روش نيازي به اشاره گر CEP نداريم شكل زير نمونه اي از اين ساختار را نشان ميدهد:



شكل 9 - 2

پياده سازي پشته اي:

ساده ترين تكنيك مديريت حافظه در زمان اجرا، روش استفاده از پشته است كه در بسياري از زبانها مثل C و پاسكال استفاده ميشود .براي مديريت اين پشته نياز به يك اشاره گر پشته[1] است كه همواره به بالاي پشته اشاره ميكند .همانطور كه ميدانيد ساختار پشته به صورت LIFO است كه اين ساختار جهت زير برنامه مناسب مي -باشد.هنگامي كه زير برنامه اي صدا زده ميشود،ركورد فعاليت جديدي در بالاي پشته ساخته ميشود و تمام شدن زير برنامه آن را از بالاي پشته حذف ميكند مثلاً در اغلب پياده سازيهاي پاسكال يك پشته مركزي براي ركوردهاي فعاليت زير برنامهها و يك حافظه ايستا براي سگمنت كد برنامهها مطابق شكل زير اختصاص داده مي-شود .



(a) Activation record for one procedure (b) Memory organization during execution

شكل 9 - 3

3-9- زير برنامههاي بازگشتي

بازگشتي يكي از مهم ترين ساختارهاي كنترل ترتيب در برنامه نويسي است. در زبان ليسپ كه ساختار ليست يك ساختمان داده اوليه است بازگشتي مكانيزم كنترل مهمي براي تكرار دنباله اي از دستورات است. زيربرنامه اي بازگشتي است كه به صورت مستقيم يا غير مستقيم خود را به صورت بازگشتي صدا بزند. نمونه اي از بازگشتي مستقيم تابع بازگشتي فاكتوريل زير به زبان C است:

```
Int fact (int n) {
If(n<=1) ret 1; else
ret fact(n-1)*n ;}
```

اگر زير برنامه A زير برنامه B را فراخواني كند و زير برنامه B نيزAرا فراخواني كند بازگشتي غير مستقيم به وجود ميآيد .تنها فرق بين فراخواني بازگشتي با فراخواني معمولي اين است كه در فراخواني بازگشتي، در حين طول عمر اولين سابقه فعاليت، ركورد فعاليت ديگري ايجاد ميشود اگر سابقه فعاليت دوم هم فراخواني بازگشتي ديگري را پديد آورد،سابقه فعاليت به طور هم زمان وجود دارند و اين روند ميتواند به همين صورت تكرار شود.

پياده سازي:

به دليل امكان وجود چند سابقه فعاليت به طور همزمان به هر دو اشاره گر CIP,CEP نياز است در هنگام فراخواني هر زير برنامه، ركورد فعاليت جديدي ايجاد ميشود و با دستور برگشت از بين ميرود .در زبانهايي مانند پاسكال و C به كمك پشته به راحتي توابع بازگشتي پياده سازي مي شوند.

در هنگام عمليات كنترل ترتيب زنجيره اي از ركوردهاي فعاليت روي پشته ساخته مي شود كه اصطلاحاً به آن زنجيره پويا[1] مي گويند.

براي پياده سازي زير برنامههاي بازگشتي هر دو اشاره گرCEPوCIP بايد استفاده شوند. از جهت چگونگي مديريت آنها از پشته مركزي استفاده مي شود. از جهت مديريت حافظه، تمام Active Recordها در پشته ذخيره مي شوند .در ازاي هر Call محتواي هر دو اشاره گرCEPوCIPدر پشته ذخيره مي شوند و در ازاي Return از پشته،pop مي شوند.

در زبانهايي مثل پاسكال و C به كمك پشته به راحتي توابع بازگشتي پياده سازي مي شوند. در هنگام عمليات كنترل ترتيب زنجيره اي از ركوردهاي فعاليت روي پشته مركزي ساخته مي شود كه اصطلاحاً به اين زنجيره از لينكها، زنجيره پويا[2] گفته مي شود.در زبانهايي مثل PL/I زير برنامههاي بازگشتي با كلمه كليدي Recursive مشخص مي شود.

زنجيره پويا: زنجيره پيوندي كه ابتداي آن ركورد فعاليت جاري است كه هر ركورد فعاليت در آن به ركورد فعاليت زيربرنامه اي كه آن را صدا زده است اشاره مي كند.

9-4- صفات كنترل دادهها(محيط ارجاع)

ويژگيهاي كنترل دادههاي يك زبان برنامه سازي، آن بخشهايي از زبان است كه در نقاط مختلفي از اجراي برنامه به دادهها دستيابي دارند ، وقتي در حين اجرا به عملياتي رسيديم بايد دادههاي مورد نياز آن عمليات آماده باشند ،ويژگيهاي كنترل دادهها در يك زبان تعيين مي كند كه دادهها چگونه

براي عمليات آماده مي شوند و نتايج عمليات بايد ذخيره و توسط عمليات بعدي بازيابي شوند. براي مثال براي عبارت X:=Y+2*Z وظيفه كنترل دادهها تعيين اين نكته است كه مثلاً در هر اجرا كدام Y استفاده شود. زيرا ممكن است Y يك متغير محلي يا غير محلي باشد.

راههاي استفاده از يك عملوند در يك عملگر:

- انتقال مستقيم[3] F206:

براي نمونه 2*Z درعبارت X:=Y+2*Z به صورت مستقيم استفاده شده است و نامي به آن اختصاص داده نشده است. در اين موارد زبان از يك حافظه موقتي براي آن استفاده مي كند.

[1] Dynamic chain

[2] Dynamic chain

[3] Direct Transmition

- مراجعه به عملوند (شي داده) از طريق نام آن:

مراجعه به شيء داده به صورت غير مستقيم و از طريق نام آن. براي نمونه در عبارتX:=Y+2*Z نام Z دلالت بر داده اي مي كند كه بايد در عمل ضرب استفاده شود. اجزايي از برنامه كه مي توانند داراي نام باشند:

- نام متغييرها
- نام پارامترهاي مجازي
- نام زيربرنامهها
- نام براي نوعهاي داده تعريف شده در برنامه
- نام برچسبهاي دستورات
- نام استثناها
- نام عمليات اوليه(+و*و/و)..
- نام ثابتهاي لفظي(ليترالها:) مانند 25.14

نامها به دو دسته نامهاي ساده و مركب تقسيم بندي مي شوند:

نام ساده: نام يك متغيير ساده مانند A

نام مركب:نامي كه به يك مؤلفه از يك ساختمان داده منتسب مي شود.مانند A[2]

مفهوم وابستگي و محيطهاي ارجاع:

207207207207وابستگيF_1:

انقياد هر نام به يك شي داده يا هر نام به يك زير برنامه خاص را وابستگي گويند.در آغاز برنامه اصلي در تعريف متغييرها، هر متغيير را به يك شي داده مقيد مي كنيم و با اين كار وابستگي بين شناسه تعريف شده و شي داده مربوط به آن را ايجاد مي كنيم.

محيط 208208208208ارجاعF_2:

هر زير برنامه اي داراي مجموعه اي شناسه ميباشد كه در طول اجرا از آنها استفاده ميكند به اين مجموعه محيط ارجاع گويند. محيط ارجاع زير برنامه در حين اجرا متغير نيست اين محيط ارجاع در حين ايجاد ركورد فعاليت زير برنامه ايجاد و تنظيم ميگردد و با از بين رفتن ركورد فعاليت زير برنامه از بين ميرود .مقادير موجود در اشيا داده ممكن است تغيير كنند ولي وابستگي نامها به اشيا داده و زير برنامهها تغيير نميكنند .

association [1]
Refrencing Enviroment [2]

انواع محيط ارجاع:

محيط ارجاع ـمحلي[1]ـ:

شامل كليه اسامي است كه هنگام ورود به يك زير برنامه ايجاد شده و داخل زير برنامه قابل دسترس هستند. مانند پارامترهاي مجازي ، متغيرهاي محلي و زير برنامههايي كه درون آن زير برنامه تعريف شده اند.

محيط ارجاع غير ـمحلي[2]ـ:

مجموعه اي از وابستگيهاي مربوط به شناسههايي كه در زير برنامه استفاده ميشوند ولي به هنگام ورود به آن ايجاد نميشوند را محيط ارجاع غير محلي گويند. شامل اسامي است كه داخل زير برنامه قابل دسترسي هستند.ولي هنگام ورود به زير برنامه ايجاد نميشوند .نمونه اي از آن متغيرهاي محلي static در زبان C ميباشد.اسامي غير محلي به دو صورت حوزه پويا و حوزه ايستا قابل

دسترس هستند. به عنوان مثالي ديگر در زبان پاسكال اگر تابع f در داخل تابع g تعريف شده باشد محيط ارجاعي غير محلي براي f ، محيط ارجاعي محلي براي g ميباشد.

محيط ارجاع سراسري[3] :

اين شناسهها در شروع اجراي برنامه پديد آمده و در كل برنامهها قابل دسترس هستند مانند متغيرهايي كه در اول برنامه پاسكال و C تعريف ميشوند .

نكته: اسامي عمومي بخشي از اسامي غير محلي هستند.

نكته:هر زير برنامه جزء محيط غير محلي خودش است.

محيط ارجاع از پيش تعريف شده[4]:

شامل اسامي است كه توسط كامپايلر تعريف شده و داخل برنامه قابل دسترس هستند يا بعضي از شناسهها وابستگيهايي از پيش تعريف شده دارند و هر زير برنامه يا برنامه ميتواند بدون ايجاد صريح از آنها استفاده كند. مانند maxint در زبان پاسكال يا كلمات كليدي. همچنين عمليات اوليه اي مانند +و - نيز به طريقي مفهوم محيط ارجاع از پيش تعريف شده را ميرسانند.

برنامه زير نمونه كاملي است كه محيط ارجاعهاي مختلف را نشان ميدهد .

```
var   ProgRAM main;
procedure     A,B,C:real;
                sub1(A:real);
        var D:real
procedure sub2(C:real);
var D:real;
              begin
```

local ١ Non-local ٢ global ٣ predefined ٤

```
...
    end;
        begin
...
sub2(B);
...
    end;
        begin
...
sub1(A);
...
    end.
```

محيط ارجاع براي sub2	محيط ارجاع براي sub1	محيط ارجاع براي main
محلي: C و D غيرمحلي: A ، sub2 ، B	محلي: A ، D و sub2 غيرمحلي: B ، C و sub1	محلي: A ، B ، C و sub1

مفهوم قابليت ـ مشاهده[1] ـ:

اگر يك وابستگي براي يك زيربرنامه بخشي از محيط ارجاع آن باشد مي گوييم آن وابستگي در آن زيربرنامه قابل رويت(مشاهده) است. اگر يك وابستگي وجود داشته باشد ولي قابل رؤيت نباشد در اصطلاح آن وابستگي، پنهان[2] ـ ناميده مي شود.

گفته ميشود شناسه x در زير برنامه يا برنامه f قابل مشاهده است. اگر x قسمتي از محيط ارجاع f را تشكيل دهد.به عبارت ديگر وابستگي شناسه x به يك شي داده در حين ورود به زير برنامه f تعيين شده باشد اغلب وقتي وابستگي مخفي است كه زير برنامه اي شناسه اي را كه در جاي ديگر در حال استفاده است دوباره تعريف كند.

حوزه پويا:

هر وابستگي داراي حوزه پوياييي است. بخشي از اجراي برنامه كه طي آن يك وابستگي به عنوان بخشي از محيط ارجاع وجود دارد،حوزه پوياي آن وابستگي ناميده مي شود.

---

[1] Visibility

[2] Hidden

عمليات ارجاع :

يك عمليات ارجاع عملياتي است كه يك شناسه و يك محيط ارجاع را گرفته ، شناسه را در محيط پيدا كرده و يك شي داده يا زير برنامه را بر ميگرداند.

نكته: ارجاع به يك شناسه ميتواند محلي، غير محلي يا سراسري باشد. وقتي ارجاع محلي است كه عمليات ارجاع ، شناسه را در محيط ارجاع محلي پيدا كند و وقتي ارجاع غير محلي يا سراسري است كه عمليات ارجاع شناسه را در محيط غير محلي يا سراسري بيابد.

9-5- اعلان پيشرو[1] در پاسكال

فراخواني بازگشتي غيرمستقيم در راهبرد تك گذره كامپايلرها مثل طراحي بسياري از كامپايلرهاي پاسكال ، مشكلاتي را به وجود مي آ ورد. فرض كنيد A وB دو زير برنامه باشند و A زير برنامه B را فراخواني كند وB نيز زير برنامه A را فرا خواني كند (بازگشتي غير مستقيم). چنانچه تعريف A قبل از B بيايد آ نگاه براي فراخواني B در A لازم است كه تعريف A قبل از تعريف B ظاهر شود و جابجايي تعريفهاي B وA نيز مسأله را حل نخواهد كرد .اين مشكل در پاسكال با اعلان پيشرو حل خواهد شد. اعلان پيشرو مثل امضاي زير برنامه است كه ليست پارامترها وكلمه Forward است به عنوان مثال:

```
ProcedureA(formal-parametr-list);Forward;
```

پس از اين اعلان پيشرو براي زير برنامه A ، زير برنامه B مي تواند تعريف شود و بعد از زير برنامهB ، تعريف كامل A ظاهر شود ولي ليست پارامترهايي كه در اعلان پيشرو آ مده است در تعريف كامل A تكرار نخواهد شد. تعريف پيشرو اطلاعات كافي را در اختيار كامپايلر قرار مي دهد تا بتواند فراخواني A را در B كامپايل كند. در پكيج Ada نيز به همين شكل عمل مي شود يعني در مشخصات پيكيج Ada ، به جاي جزئيات پياده سازي زير برنامه ،امضا ي آ ن ظاهر مي شود. عيب اعلان پيشرو اين است كه چون ليست پارامترها نبايد در تعريف كامل زير برنامه ظاهر شود مي تواند مولد خطا باشد. يك راه حل براي اين عيب اين است كه در كنار تعريف زير برنامه, توضيحاتي ارائه شود كه ليست پارامترها را مشخص كند. استفاده ازساختار Forward در پاسكال ، ناهنجاري عجيبي را در پاسكال به وجود مي آورد كه در شكل زير آن را تفسير مي كنيم :

forward declaration [1]

```
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;
begin {of S} {3}
end; {of S}
begin {of T}
U;
end; {of T}
begin {of anomaly}
T;
end. {of anomaly}
```

اين برنامه سه تفسير متفاوت دارد :

- فراخواني S{2} در داخل U مي تواند به منظور S{3} تلقي شود كه در اين صورت بايد Forward انجام مي شد چون تعريف S بعد از فراخواني قرار دارد.در برخي كامپايلرها اين فراخواني به اين صورت تلقي شده و برنامه ترجمه مي شود(تفسير نادرست ولي منطقي)
- فراخواني S{2} در داخلU مي تواند به منظور S{1} تلقي شود كه در اين صورت با قوانين پاسكال در تضاد است چون يك زيربرنامه نمي تواند زيربرنامه هم سطح با اجداد خود را فراخواني كند(تفسير بسيار نادرست)
- عمل كامپايل با شكست مواجه مي شود. چون اعلان پيشرو براي S{3} در داخل T انجام نشده است.

9-6- نام مستعار[1]

يك شي داده ممكن است در طول عمرش چندين نام داشته باشد، يعني ممكن است چندين وابستگي در محيطهاي ارجاع مختلف داشته باشد؛ به گونه اي كه هر كدام آ ن شيء را با نام مختلفي بشناسند؛ مانند هنگامي كه يك شيء داده به عنوان پارامتر از نوع ارجاع به يك زير برنامه فرستاده شود، وقتي از طريق بيش از يك نام بتوان به يك شيء داده اي دست يافت هر كدام از اين نامها را ، نام مستعار مي خوانند. اگر شيء داده چند نام داشته باشد ولي هر نام در هر محيط ارجاع، منحصر به فرد باشد مشكلي پيش نمي آيد. اما اگر در يك محيط ارجاع بتوان از طريق چند نام به شيء داده اي دست يافت هم برنامه نويس و هم پياده ساز با مشكلات جدي مواجه خواهند شد.

---

[1] aliasing

به عنوان مثال در برنامه الف شكل زير نام مستعار وجود ندارد ولي در برنامه ب در زير برنامه sub1، اسامي I ،J نام مستعاري براي يك شيء داده اي هستند چرا كه I از طريق »ارسال پارامتر با ارجاع« به زير برنامه فرستاده شده است. در مثال زير 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 حالي كه در مثال زير 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 برخي اوقات نام مستعار منجر به تغيير نتيجه برنامه در ترتيبهاي متفاوت دستوراتي كه در ظاهر تاثيري در كار هم ندارند مي شود. به عنوان مثال اگر دو نام X,C نامهاي مستعاري براي يك شيء داده باشند تغيير ترتيب دو دستور زير نتايج متفاوتي را منجر مي شود.

```
X: = A + B;

Y: = C + D;
```

نكته: معمولاً نامهاي مستعار بهينه سازي كد برنامه را دشوار مي كنند .

9-7- حوزه پويا و ايستا

در حوزه پويا براي پيدا كردن وابستگي يك شناسه بايد در زمان اجرا ركورد فعاليت جاري جستجو شده و در صورت عدم پيدا شدن وابستگي ركوردهاي فعاليت زنجيره پويا به ترتيب جستجو شوند تا وابستگي يافت شود. در حاليكه در حوزه ايستا ركوردهاي فعاليت زيربرنامههاي دربرگيرنده زير برنامه جاري در ساختار برنامه جستجو مي شوند. زبان -هاي APL,Lisp,SNOBOL4 از حوزه پويا استفاده مي كنند و لذا ارجاع به يك نام در اين زبانها نيازمند فرايند پيچيده و پرهزينه اي است.

نكته: در روش حوزه ايستا از زنجيره ايستا و در روش پويا از زنجيره پويا براي پيدا كردن وابستگي استفاده مي شود.

قاعده حوزه 217217217217پوياF_1:

حوزه پوياي هر وابستگي را بر حسب حالت پوياي اجراي برنامه تعريف مي كنند. به عنوان مثال فرض مي كنيم يك وابستگي براي شناسه x در حين ورود به زير برنامه f ايجاد شده است، قاعده حوزه پويا بيان مي كند كه از زمان اجراي f ، حوزه ي پوياي وابستگي x علاوه بر خود سابقه فعاليت، شامل زير برنامههاي ديگري است كه توسط f فراخواني مي شود و حتي اگر زير برنامه ي ديگري توسط زير برنامه اي فراخواني شود كه f آن را فراخواني كرده است شامل آن نيز مي شود. زنجيره پويا ي سابقه ي فعاليت زير برنامه f ، شامل خود f ، زير برنامههايي كه f را فراخواني كرده اند و نيز زير برنامه فراخواني شده توسط f مي باشد و ...

در روش حوزه ارجاعي پويا ، اغلب از قانون » تازه ترين وابستگي[2]« استفاده مي شود. در اين قانون اگر مثلاً تابع f ، g را صدا بزند و g هم h را صدا بزند و متغير x هم در f و هم در g تعريف شده باشد ولي در h تعريف نشده باشد ، هنگام استفاده از x در h ، كنترل به x موجود در g ارجاع مي شود ؛ يعني در اين قانون متغير به نزديك ترين تابعي در زنجيره ي فراخوانيها ارجاع مي شود.

Most Recent Association Dynamic scope rule ٢١

مثال)خروجي برنامه زير در حوزه پويا چيست؟

```
Var    Procedure A( )
                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 توسطAصدا زده مي شود پسx درون ( ) B به x درون A ارجاع مي شود. مرتبه دوم(B)توسط ( )C صدا زده مي شود پس x درون ( ) B به x درون ( ) C ارجاع مي شود.
بنابراين خروجي 31و8 مي باشد.

نكته : پارامترهاي غير محلي مانند پارامترهاي مرجع[1] بوده واگر درون زيربرنامه تغييركند مقدار اصلي آنها نيز تغيير خواهد كرد.

يكي از روشهاي پياده سازي ارجاع پويا استفاده از پشته مركزي است مثال زير اين روش را نشان مي دهد:.فرض كنيد زير برنامه P، زيربرنامه Q را فراخواني مي كند و Q نيز زيربرنامه R را فراخواني مي كند. هنگامي كه R اجرا مي شود، پشته مركزي ممكن است مثلابه شكل زير باشد.
جهت پردازش غير محلي x، پشته با شروع از محيط محلي R به طرف عقب جستجو مي شود تا تازه ترين وابستگي براي x پيدا شود. در اين حال برخي از وابستگي-هاي موجود در پشته با وابستگيهاي بعدي براي همان شناسه، مخفي است.

۱

Refrence

محيط برنامه اصلي			بخش سايه دار وابستگيهايي را نشان مي دهد كه نمي توانند در R مراجعه شوند
محيط P	X		
	Y		X, U, B ممكن است در R به طور غير محلي رجوع شوند
محيط Q	B		
	A		
	U		
	X		
محيط R	Z		
	Y		
	A		
	W		
	V		

جدول 9 - 1

قاعده حوزه $^{0}_{0}{}^{0}_{0}$ايستا^{F_1} :

در حوزه ارجاعي ايستا اسامي در زمان كامپايل و بر اساس ساختار تو در توي زير برنامهها مشخص ميشوند. به عنوان مثال زبان پاسكال كه از حوزه ارجاعي ايستا استفاده ميكند متغيرهاي غير محلي يك زير برنامه ، متغيرهاي متعلق به تابع در بر گيرنده آن زير برنامه ميباشند.بعنوان مثال در پاسكال قاعده حوزه ايستا تعيين ميكند كه ارجاع متغير X در برنامه F به اعلان X در آغاز F اشاره ميكند يا اگر در آنجا اعلان نشده باشد به اعلاني از X در آغاز زير برنامه q اشاره دارد كه خود تعريف زير برنامه F داخل برنامه q ميباشد و q مثلا داخل N و N داخل ....

به طور كلي قاعده حوزه ايستا ، ارجاعها را به اعلان اسامي در متن برنامه مربوط ميكند ولي قاعده حوزه پويا ارجاعها را با وابستگيهاي اسامي (فراخوانيها) در حين اجراي برنامه ربط ميدهد. همواره ايده آل اين است كه بين دو قاعده ايستا و پويا سازگاري وجود داشته باشد يعني در هر دو حالت ، وابستگي تعيين شده براي شناسه X يك زير برنامه يكسان باشد كه برقراري اين حالت مشكل است.

۱

مزاياي حوزه ايستا:

- امكان ايجاد قوانين مشخص و واضح ساختار بلوكي
- امكان انقياد در زمان ترجمه
- امكان بررسي كنترل نوع ايستا
- سرعت اجراي بيشتر
- هزينه كمتر

مثال)خروجي برنامه زير در حوزه ارجاعي ايستا را مشخص كنيد؟

static scope rule

```
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 اين برنامه اگر از حوزه ارجاعي ايستا استفاده شود هنگام صدا زدن ( )B چون زير برنامه ( )B متغير محلي x را ندارد ، از متغير محلي x مربوط به زيربرنامه دربر گيرنده آن يعني ( )A استفاده كرده و خروجي آن برابر 7+1 يعني 8 ميشود. سپس با صدا زدن زير برنامه ( )C ، داخل اين زير

۱

برنامه يك متغير محلي X با مقدار 03 ساخته مي -شودكه ربطي به X موجود در ( )A ندارد. حال پس از صدا زدن ( )B درون زير برنامه ( )C ، دوباره زيربرنامه ( )B از X مربوط به ( )A استفاده كرده و خروجي آن 9=8+1 ميشود. پس خروجي نهايي 9 و 8 ميشود .

9-8- ساختار بلوكي

مفهوم ساختار بلوك در زبانهاي ساخت يافته بلوكي مانند پاسكال پيدا شد. مفاهيم مربوط به ساختار بلوكي از زبان Algol 60 سر چشمه گرفته شده است. در يك زبان ساخت يافته بلوكي ، هر برنامه يا زير برنامه به صورت مجموعه اي از بلوكهاي تودرتو سازماندهي ميشوند .ويژگي مهم بلوك آن است كه محيط ارجاع جديدي تعريف ميكند .هر بلوك ميتواند شامل تعاريف بلوكهاي ديگر باشد و به اين صورت بلوكهاي تودرتو[1] پديد ميآيد .اسامي قابل دستيابي غير محلي را به دو صورت ميتوان مشخص كرد ، يكي به صورت حوزه ايستا و ديگري به صورت حوزه پويا. قواعد حوزه ايستا مربوط به برنامه ساخت يافته بلوكي عبارتند از:

- اعلانهاي ابتداي هر بلوك ، محيط ارجاعي محلي آن بلوك را پديد ميآورند. اگر داخل يك بلوك ، از اسمي استفاده شود كه در اول آن بلوك به صورت محلي تعريف شده است كامپايلر از آن اسم محلي استفاده ميكند و به سراغ اسامي ديگر نميرود.

nested

۱

- اگر در بلوكي از اسمي استفاده شود كه در آن به صورت محلي وجود ندارد ، يك بلوك به سمت بيرون رفته و در اسامي بلوك در بر گيرنده آن دنبال آن اسم ميگردد. اگر پيدا نكند اين عمليات را به سمت بلوكهاي بيروني تكرار ميكند تا بالاخره آن اسم را يافته و از آن استفاده ميكند.
- اسامي محلي موجود درون يك بلوك از ديد بلوك خارجي آن مخفي است.
- بلوك مي تواند داراي نام باشد. نام بلوك بخشي از محيط ارجاع محلي دربرگيرنده آن محسوب مي شود.

قاعده حوزه ايستا:اگر خروجي برنامه را در حوزه ايستا بخواهيم بررسي كنيم در صورت وجود نداشتن يك متغير ، بايد به بلاكهاي بيروني تر آن مراجعه كرد و از مقادير آنها استفاده كرد.

قاعده حوزه پويا: اگر خروجي برنامه را در حوزه پويا بخواهيم بررسي كنيم در صورت وجود نداشتن يك متغير ،بايد سراغ بلاكي برويم كه زير برنامه را فراخواني كرده است (قاعده تازه ترين وابستگي) به چند نمونه از تستهاي كنكور در زمينه قواعد حوزه ايستا و پويا توجه فرمائيد:

1- خروجي برنامه زير در حالتي كه از قواعد static scoping و dynamic scoping استفاده شود، به ترتيب كدام گزينه است؟ مهندسي كامپيوتر 58

```
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.
```

dynamic scoping: 600 , static scoping: 400

dynamic scoping: 900 , static scoping: الف.

dynamic scoping: 900 , static ب400 .

dynamic scoping: 400 , static جscoping: 600.

scoping: 900.د

2- در قطعه برنامه ي زير كه به زبان برنامه سازي ML نوشته شده است مقدار عبارت b*f(a,a) در دو حالتي كه زبان از قواعد حوزه ايستا (static scoping) و حوزه پويا (dynamic scoping) استفاده مي كند، كدام است؟

```
let a = 5,b =10, c = 7 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 ايستا: 06 و حوزه

پويا :43 ب .حوزه ايستا: 005

و حوزه پویا :340 ج .حوزه
ایستا: 001 و حوزه پویا :86
د.هیچکدام

3-برنامه زیر را در نظر بگیرید. کدام گزینه صحیح است؟(مهندسي كامپيوتر 37)

```
var   Program test;
procedure   x:real;
            Display;
        begin
    write(x);
        end;
    procedure
            Assign;
    var
  begin    x:real;
x:=0.72;
Display;
          end;
               begin
       x:=0.27;

       Display;
       Assign;
             end.
```

الف) در دامنه پویا مقدار چاپ شده توسط برنامه 27.0 و 72.0 میباشد.

ب)در دامنه ایستا مقدار چاپ شده توسط برنامه 27.0 و 27.0 میباشد.

ج) در دامنه ایستا مقدار چاپ شده توسط برنامه 27.0 و 72.0 میباشد.

د) گزینه الف و ب

4 -خروجي برنامه زير با استفاده از قواعد حوزه پويا (Dynamic Scope) چيست؟(مهندسي كامپيوتر 72)

```
var   Program main;
          x:integer;
```

```
procedure foo;
var x:integer;
    begin
    x:=7;
     bar;
      end
procedure
          bar;
   begin
print x;
     end;
          begin
     x:=3;
     foo;
          end.
```

الف) 3 ب) 7 ج) 3و7 د) 7و 3

نكته:در دامنه پويا عمل binding اسامي متغيرها به اشيا با توجه به سلسله مراتب فراخواني روالها(معنايي) و در زمان اجرا صورت ميگيرد. در دامنه ايستا عمل binding اسامي متغيرها به اشيا با توجه به تودرتويي تعريف روالها (نحوي) و در زمان ترجمه انجام ميشود .

نكته:اغلب زبانهايي كه از حوزه ايستا استفاده مي كنند مانند Cو پاسكال،كنترل نوع ايستا را انجام مي دهند. بنابراين ساختارهاي زمان اجرا ساده تر شده و برنامه سريع تر اجرا مي شود همچنين قابليت اعتماد آن زيادتر مي شود .

اگر در اجراي برنامه M زنجيره فراخواني برنامههاي فرعي به صورت M→P→R→Q→S (يعني M ، P را فرا خواني كرده ، P ، R را ، R ، Q را و Q ، S را فراخواني كرده است) ، محيط ارجاع را در دوحالت پيادهسازي قانون شناسايي ايستا و قانون حوزه شناسايي پويا در نظر بگيريد. كدام يك از متغيرهاي تعريف شده در برنامه در هر دو محيط ارجاع وجود دارند؟ به عبارت ديگر كدام يك از متغيرهاي برنامه در هر دو حالت از داخل S قابل ارجاع هستند؟ (مهندسي كامپيوتر 57)

```
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,y,z,w,u ب) x,z,u ج) x,y,z,u د) هيچكدام

9-9- محيط ارجاع براي دادههاي محلي

در اينجا محيط ارجاع براي دادههاي محلي كه ساده ترين ساختار را دارند بررسي مي كنيم.محيط محلي زيربرنامه شامل پارامترهاي مجازي و متغييرهاي محلي آن زيربرنامه مي باشد.

```
Procedure sub1(a:real);
Var b:real;
Procedure sub2(c:real);
Var d:real;
Begin
c:=c+b;
End;
Begin
…..
```

```
End;
```

متغير محلي در زيربرنامه sub2 عبارتست از d و متغيير محلي در زيربرنامه sub1 عبارتست از c. وابستگي متغيرهاي محلي به دو صورت »حذف«و»نگهداري« مي باشدكه اين دو مفهوم را همراه با يك مثال توضيح مي دهيم.

روش نگهداري F_1:

در اين روش،در اولين ورود به زيربرنامه متغير تعريف و اجرا مي شود. در موقع بازگشت از زيربرنامه متغير از بين نمي رود و در فراخوانيهاي بعدي از آخرين مقدار متغير استفاده مي شود. كوبول و فرترن از روش نگهداري استفاده مي كنند.در پياده سازي اين روش محل نگهداري اشياء داده اي در كد سگمنت برنامه مي باشد.

روش 3333حذف F_2:

در اين روش متغير در اولين ورود به زيربرنامه ايجاد مي شود و به هنگام بازگشت از بين مي رود. زبانهاي Ada، Lisp، C,SNOBOL4,APL,Java,C++, Pascal از روش حذف استفاده مي كنند.در پياده سازي اين روش محل نگهداري اشيا داده در ركورد فعاليت زيربرنامه كه معمولاً پشته است مي باشد.

نكته: Algol وPL/I ازهر دو روش استفاده مي كنند.

مثال:خروجي تكه برنامه زير را هنگامي كه از روش نگهداري و حذف استفاده مي شود بدست آوريد.؟

```
Procedure Q;
Var x:integer:=30;
Begin
Write(x);
X:=x+1;
End;
Procedure P;
Begin
Q;
Q;
Q;
End;
```

1. نگهداري:اگر وابستگيx نگهداري شود مقدار اوليه x:=30 فقط بار اول صدا زدن Q انجام مي گيرد و پس از خروج از Q مقدار x حفظ مي شود.بار اول كد Q داخلp صدا زده مي شود.مقدار اوليه 03 چاپ شده و x برابر 13 مي شود. بار دوم كه Q صدا زده مي شودمقدار قبلي 13 چاپ

شده وx برابر 32 مي شود و بار سوم اين مقدار 23 چاپ مي شود.پس خروجي نهايي 23 و13 و03 مي باشد.

2. حذف:در اين روش در هر بار خروج از Q ، x حذف شده و در هر بار ورود به Q يك x جديد با مقدار اوليه

03 ساخته مي شود.لذا در اين حالت خروجي برنامه 03 و 03 و03 مي شود.

---

[1] Retention

[2] Deletion

نكته:

نگهداري وحذف دو روش مختلف براي معناي محيطهاي محلي اند و به طول عمر اين محيط مربوط ميشوند. در زبان C متغيرهاي محلي معمولي از نوع حذف ومتغيرهاي محلي static از نوع نگهداري هستند.

مزيت روش نگهداري:اين روش به برنامه نويس اجازه ميدهدتا زيربرنامههايي ايجادكندكه نسبت به گذشته حساس باشند.به طوري كه بخشي از نتايج آنها در هر فراخواني توسط ورودي وبخشي ديگرتوسط دادههاي محلي تعيين شود كه در حين سابقه فعاليت قبلي ايجاد شده اند اين در حالي است كه در روش حذف براي انتقال دادهها از يك فراخواني به فراخواني ديگر از همان زيربرنامه بايد يك متغير به صورت غير محلي ايجاد شود.

مزيت روش حذف:براي زير برنامههاي بازگشتي روش حذف متداول تر است.روش حذف موجب صرفه جويي در حافظه ميشود .

مزايا و معايب روش حذف و نگهداري:

- در روش نگهداري امكان ايجاد زيربرنامههاي حساس به سابقه وجود دارد.
- در روش حذف متغييرهاي محلي امكان حفظ كردن مقدار خود را ندارند و لذا اين كار بايد با متغييرهاي غير محلي انجام

شود كه امنيت و جامعيت برنامه را كاهش مي دهد.

- روش حذف در حافظه صرفه جويي بيشتري انجام مي دهد.
- در روش حذف سرعت اجراي برنامه به دليل ايجاد و حذف دادهها كمتر است

نكته: در زبان C,C++ براي متغييرهاي static از روش نگهداري استفاده مي شود. در زبان PL/I براي متغييرهاي static از روش نگهداري و براي متغييرهاي automatic از روش حذف استفاده مي شود. در زبان الگول هر دو امكان فراهم است.

پياده سازي روش حذف ونگهداري:

در پياده سازي محيط ارجاع ميتوان براي معرفي محيط محلي يك زيربرنامه از يك جدول محيط محلي L.E.T[1] شامل هم شناسه وهم شي داده نظير آن استفاده كرد.بعنوان مثال جدول L.E.T براي زيربرنامه زير به صورت زير است:

[1] Local Environment Table

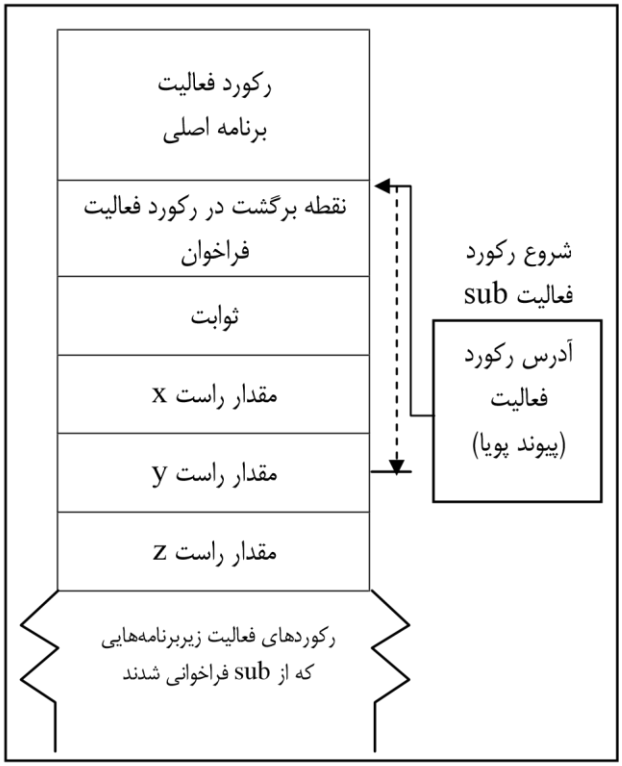
```
Procedure sub (x:integer) is
          y:real;
z:array (1..3) of
procedure sub2     real;
                 begin  is
                   …
            end {sub2};
                    begin
                      …
                  end {sub};
```

جدول محیط محلي sub (در زمان ترجمه)

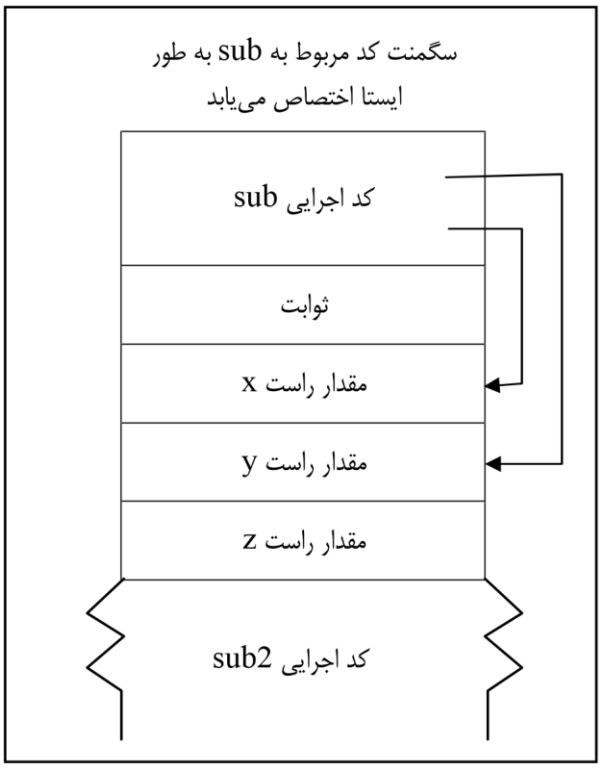
نام Lvalue	نوع محتویات	
x	Integer	پارامتر با مقدار
y	Real	متغير محلي
z	Real	آرایه
		توصیفگر : [1..3]
Sub2	Procedure	اشاره گر به سگمنت کد

تعریف زیربرنامه

شکل ۹ – ۴ نمونه ای از جدول محیطی



شکل ۹ – ۵ تخصیص و ارجاع به متغیرهای محلی قابل حذف شدن



شکل ۹ – ۶ تخصیص و ارجاع به متغیرهای محلی نگهداری شده

در روش نگهداري ، L.E.T در سگمنت كد تشكيل ميشود چون سگمنت كد به طور ايستا تخصيص ميابد و در حين اجرا باقي ميماند ,هر متغيرموجود در بخش محيط محلي سگمنت كد نگهداري ميشود .درحالي كه در روش حذفL.E.T به عنوان قسمتي از ركورد فعاليت آن زيربرنامه ميباشد ودر پشته مركزي تشكيل ميشود.چون ركورد فعاليت يك زيربرنامه به هنگام ورود به زير برنامه در پشته مركزي ايجاد ميشود و با خروج از زير برنامه از بين مي-رود .بنابراين براي زير برنامههاي برگشتي روش حذف بسيار مناسب ميباشد چون روش نگهداري فضاي زيادي تلف ميكند .

چند نكته درباره روش پياده سازي وحذف:

- با هر متغير ميتوان به دو شكل برخورد كرد.آنهايي كه مقاديرشان در حافظه تخصيص يافته در سگمنت كد نگهداري ميشود وآنهايي كه مقاديرشان در ركورد فعاليت قرار ميگيرد و حافظه آنها بايد حذف شود.اين روش در پاسكال بدين صورت پياده سازي ميشودكه اگر متغيري به صورت static تعريف شود مقدارش حفظ ميشود و متغيري به صورت اتوماتيك مقدارش حذف ميشود .

- نام زيربرنامه به اعلاني براي آن زيربرنامه در محيط محلي وابسته ميشود,بنابراين همواره نگهداري ميشود .
- نام پارامتر مجازي يك شي داده اي را نشان ميدهدكه در هر بار فراخواني زيربرنامه مقدار جديدي مي گيرد.بنابراين با اين پارامترها به روش حذف برخورد ميشود .در الگول 06 وقتي متغير محلي با ownاعلان شود نگهداري خواهد شد.

اعلان پيشرو:

گاهي اوقات ممكن است لازم باشد قبل از تعريف زيربرنامه اي ازآن استفاده كنيد دراين گونه موارد بايد به اطلاع كامپايلر برسانيد كه اين زيربرنامه بعداًتعريف خواهدشد,براي اين منظور از كلمه كليدي Forward استفاده ميشود .

Procedure پارامترها) نام زيربرنامه ;Forward ; (

Forward ; نوع نتيجه تابع : (پارامترها) نام تابع Function

به عنوان مثال اگر در زيربرنامه proc1 از زيربرنامه proc2 استفاده شود ولي زيربرنامه proc2 بعداً تعريف گرددبايد به صورت زير عمل كنيد:

```
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;
```

نكته جالب اينجاست كه در هنگام اعلان پيشرو بايد پارامترهاي زيربرنامه نيز حضور داشته باشند ولي در هنگام تعريف زيربرنامه ديگر پارامترها ذكر نميگردد .

9-01- پارامترها وانتقال پارامترها

هميشه مواردي پيش ميآيدكه درحين اجراي زيربرنامهها، بايستي يك سري اطلاعات مشترك وجود داشته باشدكه بين زيربرنامهها به اشتراك گذاشته شود. معمولاًبَه چهار طريق اين امكان را فراهم ميكنند تا زيربرنامهها به اطلاعات مشترك (محيط مشترك) دسترسي داشته باشند.

- **محيط اشتراك $^{5}_{5}{}^{5}_{5}{}^{5}$صريح^{F_1}:**

در اين محيطها دادههاي اشتراكي صراحتاً بايد مشخص گردند و زيربرنامههايي كه از آنها مي توانند استفاده كنند نيز مشخص مي شوند. مثلاً در زبان فرترن دادههاي مشترك در بخشي از برنامه قرار داده مي شود و آن بخش توسط دستور Common صراحتاً مشخص مي گردد.
نمونههايي از محيطهاي اشتراك صريح عبارتند از:
كلاسها در C+ + و بلوك common در فرترن و package در Ada

- **محيط اشتراك ضمني[2] :**

در اين گونه محيطها دادهها بين زيربرنامهها مشتركند ولي تعريف صريحي از آنها انجام نمي شود. براي نمونه در زبان پاسكال مي توان براي استفاده از متغييرهاي تعريف شده در يك UNIT آن را Import كرد كه اين كار توسط دستور USES در ابتداي برنامه انجام مي شود.در زبان Modula2 نيز اين امكان وجود دارد.

- **حوزه ايستاوحوزه پويا**
- **وراثت**

محيطهاي مشترك يك راه ارتباط زيربرنامهها با يكديگراست.روش متداول تر ديگرجهت انتقال اشيا داده بين زيربرنامهها، استفاده ازتكنيك انتقال پارامترهاست.در اين تكنيك انتقال اطلاعات بين زير برنامهها، از طريق قراردادن آنها در يك سري پارامترها و فرستادن آنها به زير برنامه فراخوان ميباشد. براي بررسي تكنيك انتقال پارامتر نيازمند يك سري تعاريف اوليه هستيم كه در زير به بررسي آنها ميپردازيم:
آرگومان:يك شيء داده است كه به يك تابع يازير برنامه فرستاده ميشود .
نتيجه:مقداري است كه توسط زير برنامه يا تابع فراخواني شده برگشت داده ميشود .

9-01-1- پارامترهاي مجازي و واقعي وتناظر بين آنها

به پارامترهاي زير برنامه در هنگام تعريف آن پارامترهاي مجازي يا اسمي گفته ميشود.به پارامترهاي زير برنامه هنگام صدا زدن آن, پارامترهاي واقعي ميگويند. مثلاً در تكه برنامه زير a,bپارامترهاي مجازي و x,y پارامترهاي واقعي هستند.

[1] Explict

Implict [2]

```
Int fn(int a,int b){
        Return a+b;
      } Main(
         ){
   Int x=5,y=6,z;
     Z=fn(x,y);
         }
```

نكته:پارامتر واقعي، يك شيء داده اي است كه با زير برنامه فراخوان مشترك است.پارامتر واقعي ممكن است يك شيء داده محلي متعلق به فراخوان باشد،ممكن است پارامتر مجازي فراخوان باشد ممكن است شيء داده غير محلي باشدكه توسط فراخوان قابل مشاهده است يا ممكن است نتيجه اي باشد كه توسط تابعي برگردانده شده است كه زير برنامه فراخوان آن تابع را فراخواني كرده و نتيجه را به زيربرنامه فراخواني شده ارسال كرده است.

پارمتر واقعي در p	فراخواني زير برنامه در p
متغيرهاي محلي p : B و I	Sub (I , B)
ثوابت : 27 , true	Sub (27 , true)
پارامترهاي مجازي p : p1 , p2	Sub (p1 , p2)
متغيرهاي عمومي يا غيرمحلي p : G1 , G2	Sub (G1 , G2)
عناصر آرايهها و ركوردها	Sub (A[i] , D , B1)
نتايج توابع تعريف شده يا اوليه	Sub (i+3 , fn(Q))

جدول 9 - 2

هنگام فراخواني بايد تناظري بين پارامترهاي واقعي و مجازي برقرارگردد براي اين كاردو قاعده وجود دارد:

- **تناظرموقعيتي:** دراين تناظر كه در اكثر زبانها استفاده ميشود، اولين پارامتر واقعي به اولين پارامتر مجازي ،دومين پارامتر واقعي به دومين پارامتر مجازي و ...نگاشت ميشود زبانهايC و پاسكال اينگونه هستند.
- **تناظر براساس نام:**در برخي از زبانها مانند Ada از تناظر نام استفاده ميشود يعني پارامترها براساس نام نگاشت ميشوند و ترتيب آنها مهم نيست مثلا دستور فراخواني زير در زبان Ada :

```
Sub(x,y) {تعريف
...
```

...

}

Sub (x=>B , فراخواني

(y=>27در حين فراخواني Sub، پارامتر واقعي B با پارامتر مجازي x وپارامتر واقعي 72 با پارامتر مجازي y متناظر مي -شوند .

9-01-2- روشهاي انتقال پارامتر

بعد از اينكه پارامترهاي واقعي به پارامترهاي مجازي نسبت داده شدند به طرق مختلفي اين نسبت دادنها تفسير و استفاده ميشود به طور كلي روشهاي انتقال پارامترها[1] عبارتند از:

- (call by value) فراخواني با مقدار
- (call by refrence) فراخواني با ارجاع
- (call by name) فراخواني با نام
- (call by result) فراخواني با نتيجه
- (call by value-result) فراخواني با مقدار- نتيجه
- فراخواني با مقدار ثابت (call by const)

نكته: در زبان الگول امكان فراخواني با نام وجود دارد. در زبان فرترن فراخواني با ارجاع انجام مي شود. در زبان C از فراخواني با مقدار استفاده مي شود و در زبان C++ و پاسكال از فراخواني با مقدار و فراخواني با ارجاع مي توان بهره برد. در زبان C++ اگر قبل از نام پارامتر مجازي & قرار داده شود ارسال آن به روش فراخواني با ارجاع انجام مي شود .در زبان C نيز اگر به جاي نام شي داده آدرس آن به روش فراخواني با مقدار ارسال شود مي توان فراخواني با ارجاع را در اين زبان شبيه سازي كرد. انتقال پارامتر به روش مقدار - نتيجه در الگول_w معرفي شده است.

9-01-2-1- فراخواني با مقدار

در حالت معمولي در زبانهايي مانند C و پاسكال پارامترها به صورت فراخواني با مقدار به زيربرنامه فرستاده مي -شوند، در اين روش مقدار (مقدار راست) پارامتر واقعي در پارامتر مجازي كپي ميشود.لذا در زيربرنامه ديگر به پارامتر واقعي دسترسي نداريم و تغييراتي كه در پارامتر مجازي داده ميشود، به پارامتر واقعي اعمال نميشود .

مثال:برنامه زير در زبان C :

```
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 ); }
```

[1] parametr passing

خروجي:

1 3	قبل از فراخواني
0 0	در زير برنامه فراخواني شده
1 3	بعد از فراخواني

جدول 9 - 3

9-01-2-2- فراخواني باارجاع(آدرس)

متداولترين روش انتقال پارامترهاست دراين روش آدرس (مقدار چپ) پارامتر واقعي به زيربرنامه فرستاده مي -شود.بدين ترتيب تغييراتي كه به پارامترهاي مجازي، درون تابع داده ميشود، به پارامترهاي واقعي متناظر در برنامه صدا زننده اعمال ميشود.زبان Cاين نوع فراخواني را با استفاده از اشاره گرها پياده سازي كرده است.

```
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
}
```

3 1	قبل از فراخواني
-6 -4	در زير برنامه فراخواني شده
-6 -4	بعد از فراخواني

جدول 9 - 4

در زبان پاسكال پارامترهايي كه با مقدار فرستاده ميشوند بدون var و پارامترهايي كه با ارجاع فرستاده مي -شوندvar 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
   writeln(x,y);           fn(x,y);
                          //    3     -8
                                   End;
```

نكته: در زبان C فرستادن آرايهها همواره به صورت فراخواني با ارجاع است ولي در پاسكال اينگونه نيست.اگر قبل از نام پارامتر مجازي آرايه var نباشد فراخواني با مقدار و اگر var باشد فراخواني با ارجاع است.

9-01-2-3- فراخواني با نام

اين روش كمتر مورد استفاده زبانها ميباشد انتقال پارامتر با نام در الگول از اهميت بالايي برخوردار است ولي به دليل سربار اجرايي بالا، روش كاربردي و معروفي نيست. در اين روش نام پارامتر واقعي جايگزين نام پارامتر مجازي در زيربرنامه فراخواني شده ميگردد.در اين حال هر ارجاع به پارامتر مجازي مستلزم ارزيابي مجدد پارامتر واقعي متناظر با آن است.براي اين كار در نقطه فراخواني زيربرنامه، پارامترهاي واقعي ارزيابي نميشوند تا زماني كه در زيربرنامه به آنها مراجعه شود.براي پارامترهايي كه از نوع متغير ساده هستند مانند int، فراخواني با نام از ديد برنامه نويس از نظر نتيجه خروجي معادل فراخواني با ارجاع است ولي اگر از نوع متغير ساده نباشند ممكن است نتايج با هم فرق كند.

مثال: در procedure sub(x:integer) اگر بخواهيم فراخواني(sub(y را در متن برنامه اصلي داشته باشيم ميتوان اينگونه فرض كرد كه در بدنه روالsub به جاي همه xها ،y قرار داده ميشود

حال در هر مراجعه به y يك ارزيابي مجدد از y صورت ميگيرد از نظر كاربر نتايج فراخواني با نام و فراخواني با ارجاع يكي است ولي نتايج مياني متفاوتي از هر دو ديده ميشود .

تكنيك اصلي براي پياده سازي فراخواني با نام اين است كه با پارامترهاي واقعي مثل زيربرنامههاي فاقد پارامتر(thunk) رفتار شود.وقتي از زيربرنامه به پارامتر مجازي متناظر با پارامتر واقعي مراجعه شود، thunk ترجمه شده براي آن پارامتر، اجرا ميشود تا پارامتر واقعي در محيط ارجاع مناسبي ارزيابي شود و مقدار حاصل به عنوان مقدار thunk برگردانده شود.

```
var  i:integer; Function    f(x:integer):integer;
                              function  f(a[i]);
                                    Begin begin
i=3;   x=3;          i=2;             i=2;   x=1;
                                              a[i]=1;
 i=3;                                          End;
                                a:array [1..3] of  Begin
                                                  integer;
                                                   i=1;
                                                 f(a[i]);
                                                      end;
```

با صدا زدن تابع f ، ابتدا پارامتر A[i] جايگزين پارامتر مجازي x ميشود سپس با هر بار رجوع به x مقدارA[i] محاسبه ميشود .لذا اين برنامه مقدار A[2] را برابر 2 و مقدارA[3] را برابر 3 برميگرداند.

9-01-2-4- فراخواني با نتيجه

پارامتري كه به اين شيوه ارسال ميشود فقط جهت برگرداندن نتيجه به برنامه فراخوان استفاده ميشود.مقدار اوليه پارامتر واقعي در زيربرنامه قابل استفاده نيست. هنگامي كه زيربرنامه تمام ميشود، مقدار نهايي پارامتر مجازي به عنوان مقدار جديد پارامتر واقعي محسوب ميشود به عبارتي ديگر ميتوان گفت در اين نوع فراخواني تابع پارامتر ورودي ندارد و فقط داراي پارامتر خروجي است.

9-01-2-5- فراخواني با مقدار-نتيجه

در اين روش فراخواني، مقدار پارامتر واقعي در پارامتر مجازي كپي ميشود.داخل زيربرنامه هر تغييري فقط روي پارامتر مجازي انجام ميگيرد و روي پارامتر واقعي اثر ندارد.هنگام بازگشت از زيربرنامه، پارامتر مجازي در پارامتر واقعي كپي ميشود.يعني پارامتر واقعي تا زمان خاتمه

زيربرنامه مقدار اصلي خودش را حفظ كرده و پس از اجراي زير برنامه مقدار جديدي ميگيرد. اين روش فراخواني در زبان Algol-w استفاده شده است.

نكته:فراخواني با ارجاع و فراخواني مقدار نتيجه از ديد برنامه نويس نتايج يكساني دارند.

9-01-2-6- فراخواني با مقدار ثابت

- هنگامي كه پارامتر به صورت مقدار ثابت يا const(constant) انتقال داده ميشود در حين اجراي زيربرنامه نميتوان پارامتر مجازي را تغيير داد و اين پارامتر مجازي ثابت، در حين اجراي زيربرنامه مانند يك مقدار ثابت محلي عمل ميكند .از ديد برنامه فراخوان اين پارامتر ثابت، فقط يك آرگومان ورودي براي زيربرنامه است و مقدارش چه به صورت سهوي و چه به منظور برگرداندن نتيجه قابل تغيير نيست.

```
int fn(const int a, int b)
```

نكته:

در زبان Ada به جاي توصيف مكانيزم انتقال، نقش پارامتر مشخص ميشود .اگر پارامتر به صورت in ارسال شود مقدار پارامتر واقعي به پارامتر مجازي ارسال ميشود اگر پارامتر به صورت out باشد مقدارش توسط زيربرنامه توليد ميشود وهنگام خروج از زيربرنامه به پارامتر واقعي در زيربرنامه فراخوان منتقل ميشود .اگر پارامتر به صورتin out باشد مقدارش هنگام فراخواني به زيربرنامه فراخواني شده منتقل ميشود و هنگام خروج از آن، مقدارش در پارامتر واقعي در زيربرنامه فراخوان قرار ميگيرد .

به دليل اينكه در زبان Ada فراخواني با مقدار براي پارامترهاي in وفراخواني با ارجاع براي پارامترهاي out،مشكلات تطابق را به وجود ميآورد و برنامههاي مختلف در كامپايلرهاي گوناگون نتايج متفاوتي را برميگرداند انتقال پارامترها بازبيني شد و اصلاحات زير صورت گرفت:

- انواع داده اوليه با پارامتر in با فراخواني مقدار ثابت و با پارامتر out يا inout با فراخواني مقدار و نتيجه ارسال ميشوند و انواع داده مركب(مثل آرايه و ركورد) با فراخواني ارجاع ارسال ميشوند.

9-01-3- پياده سازي انتقال پارامتر

چون هر سابقه فعاليت زيربرنامه،مجموعه متفاوتي از پارامترها را دريافت ميكند حافظه پارامترهاي مجازي زيربرنامه به بخشي از ركورد فعاليت زيربرنامه تخصيص مييابد و نه در سگمنت كد.بنابراين اگر،محل ذخيره پارامترهاي رسمي به عنوان قسمتي از ركورد فعاليت زيربرنامه فراخواني شده باشد و پارامتر رسمي به نام P از نوع T داشته باشيم يكي از دو روش زير بسته به مكانيزم انتقال پارامتر،پياده سازي ميشود.

- P به عنوان شي داده محلي از نوع T است در صورتيكه ارزش اوليه آن يك كپي ارزش پارامتر واقعي باشد.
- P به عنوان شي داده محلي از نوع اشاره گري به T است در صورتيكه ارزش اوليه ، اشاره گري به پارامتر واقعي باشد.

نكته: روش اول براي فراخواني مقدار-نتيجه، فراخواني با مقدار و فراخواني با نتيجه استفاده ميشود روش دوم براي انتقال پارامترها از طريق ارجاع و از طريق نام استفاده ميشود .از هر دو روش ميتوان براي پياده سازي انتقال از طريق مقدار ثابت استفاده كرد و برگرداندن مقدار با تابع نيز به روش اول انجام ميشود .

اكنون به چند نمونه از تستهاي كنكور در زمينه انتقال پارامترها مي پردازيم:

1-در زبان فرضي زير، آرگومانهاي برنامهها بصورت Call by Name تعريف شده اند. با توجه به اين روش تعريف آرگومان، خروجي تكه برنامه زير چيست؟ (مهندسي كامپيوتر28)

```
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]);
```

الف2.و4و8و2 (از چپ به راست بخوانيد.) ب4.و4و6و8و2 (از چپ به راست بخوانيد.) ج2.و4و6و8و4 (از چپ به راست بخوانيد.) د2.و4و4و8و4 (از چپ به راست بخوانيد.)

2-برنامه زير را در نظر بگيريد. مقادير خروجي برنامه زير با استفاده از روش Call by name كدام است؟ **(**راهنمايي: توجه داشته باشيد كه j در عبارت [S[1+jبه متغير سراسري j اشاره دارد**()**. مهندسي كامپيوتر 37)

```
Var s:array [1..3] of char;
 var i,j:integer; procedure
```

```
var     P(x:integer;y:char);
              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 'A',1,1(د 'B',0,1)

3- برنامه زير را در نظر بگيريد.اگرروش انتقال پارامتر به روال بانام باشد مقدارList[1],List[2],global بعد از اجراي برنامه به ترتيب از راست به چپ چنداست؟

```
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;
```

4-خروجي برنامه زير در صورتي كه مكانيزم تبادل پارامتر و call ، call by value-result ، call by value

by reference و call by name باشد كدام گزينه است؟ (مهندسي كامپيوتر 58)

```
Program Main;
```

```
Var  K :integer ;
Procedure XYZ(i,j:integer);
Var K :integer ;
i:300;          Begin
if i=j then         k:=2;
                 j:=i*k+j;
                  end
                    begin
                 k=100;
               XYZ(k,k);
              Write(k)
                      end;
```

call by name	call by reference	call by value-result	call by value
900	900	900	100
6	900	100	300
6	900	100	100
900	900	100	100

الف ب

ج

د

5- 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. خروجي برنامه کدام است؟ (مهندسي کامپيوتر 78)

```
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);
                     x:=6*B;y:=x-  begin
                   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

6 -يك برنامه فرعي با پارامتر از نوع آرايه اي به طول 001 از اعداد صحيح را در نظر بگيريد. هزينه انتقال آرايه مزبور به داخل برنامه فرعي را وقتي يكي از سه روش انتقال پارامتر مورد استفاده قرار مي گيرد مقايسه
كنيد(مهندسي كامپيوتر 57)

الفvalue>value-result=reference(

بvalue-result>value>reference(

جreference=value-result>value(

دvalue>value-result>reference(

9-01-4- زير برنامه به عنوان پارامتر در زبانهاي زيادي ميتوان زير برنامهها را به عنوان پارامتر به زير برنامه ديگري فرستاد.در اين حالت پارامتر واقعي شامل نام زير برنامه و پارامتر مجازي متناظر با آن از نوع زير برنامه است.به عنوان مثال در پاسكال زير برنامه اي مانند Q ميتواند شامل پارامتر R از نوع Function يا Procedur باشد.

```
Procedure Q(x:integer; function R(y,z:integer):integer);
```

باتعريف فوق Q ميتواند به گونه اي فراخواني شود كه پارامتر دوم آن تابع باشد مانند Q(27,fn) كه Q را با تابع fn به عنوان پارامتر صدا ميزند در داخل Q زير برنامه اي كه به عنوان پارامتر ارسال شده ، ميتواند از طريق نام پارامتر مجازي مثلاً R ، فراخواني شود. نظير z:=R(i,x); كه زير برنامه fn را با پارامتر x,i فراخواني ميكند و در اين حالت R(i,x) معادل fn(i,x) است اگر

در فراخواني ديگر، پارامتر واقعي تابع gn باشد، R(i,x) تابع gn(i,x) را صدا ميزند .دو مشكل در هنگام استفاده از زير برنامه به عنوان پارامتر وجود دارد :

- **كنترل نوع ايستا:**

بايد ارگومانهاي زير برنامه اي كه به عنوان پارامتر فرستاده ميشود از نظر تعداد،نوع و ترتيب چك شود.

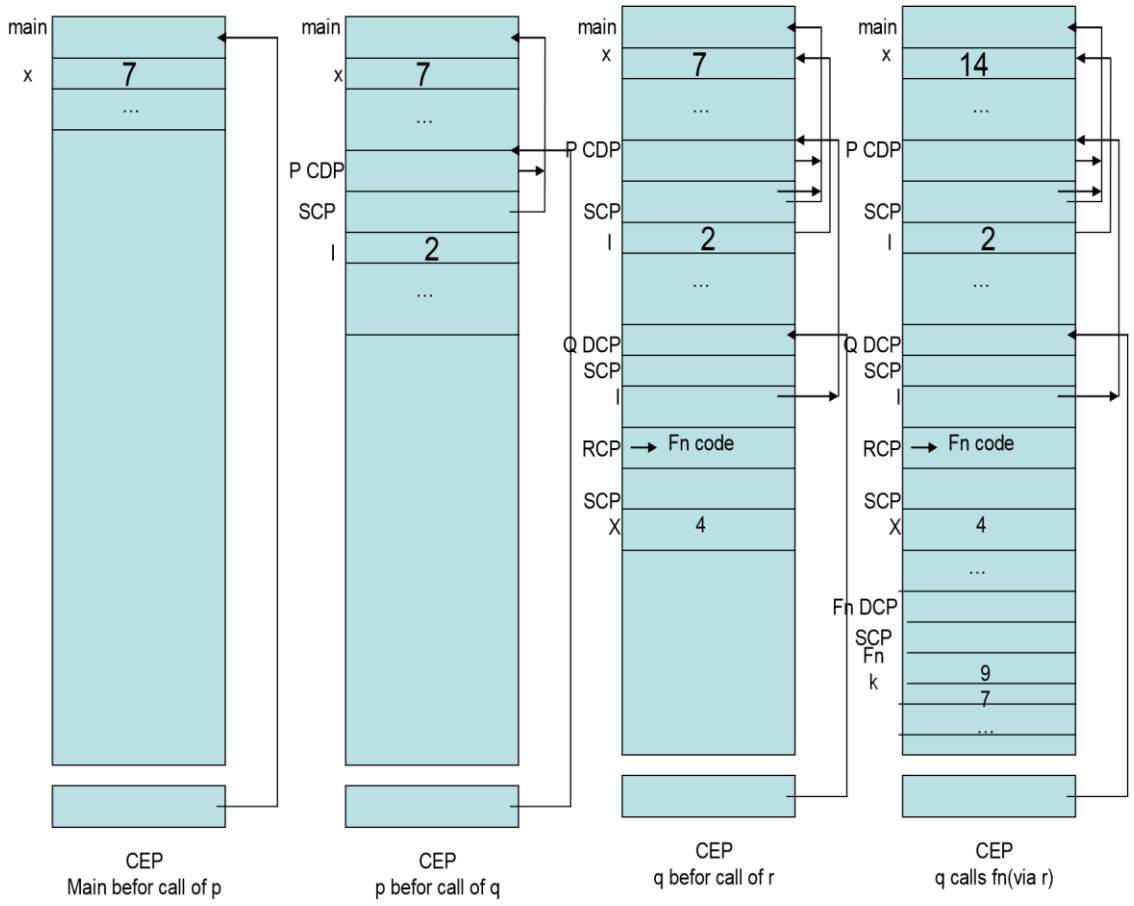
- **متغيير آزاد(ارجاعهاي غير محلي)**

در صورتيكه به هنگام مراجعه غير محلي هيچ مكانيزمي جهت binbing در تعريف زير برنامه وجود نداشته باشد به آن متغير، متغير آزاد گفته ميشود به عنوان مثال در زير برنامه زير به هنگام اجراي fn، به متغيرهاي x,i متغيرهاي آزاد گفته ميشود .

```
Program main var x:integer; procedure Q(var
i:integer;function R(j:integer):integer; Var x:integer;
write('in Q before call     x:=4;     Begin
                                    R,i=',i,'x=',x)
write('in Q after call  i:=R(i);
                                    R,i=',i,'x=',x)
                                end; procedure P() var
                                   i:integer; function
                                FN(k:integer):integer;
                          FN:=i+k;    x:=x+k;     begin
                    write('in,FN,i=',I,'k=',k,'x=',x)
                                    i:=2;      end begin
                              write('in  Q(x,FN);
                                       P,i=',I,'x=',x)
                                  x:=7;     end; begin
                           writeln('in      P();
                                        main,x=',x);
                                                end
```

همان طور كه گفتيم x,i متغيرهاي آزاد هستند.تابع fnحاوي ارجاعهاي محلي به x,i است بر اساس قواعد حوزه ايستا در پاسكال اين x،به x اعلان شده برنامه اصلي مراجعه ميكند واين i،به i اعلان شده در زير برنامه p مراجعه ميكند.p تابع fn رابه عنوان پارامتر Q ميفرستدو u تابع fn را از طريق نام پارامتر مجازي R فراخواني ميكند .x,i در Q به صورت محلي تعريف شده اند و تابع fn نميتواند به اين متغيرهاي محلي دستيابي داشته باشد مشكل متغيرهاي آزاد فقط مخصوص پاسكال كه از قاعده حوزه ايستا استفاده ميكند نيست بلكه زبانهايي مانند ليسپ كه از قاعده حوزه پويا استفاده ميكند از اين مشكل رنج ميبرد .براي بر طرف كردن مشكل فوق ، مكانيزمي براي binding استفاده ميشود كه رفتاري مشابه به آنچه fn در p فراخوانده ميشود را دارا باشد.(شكل صفحه 330,

(331



شكل 9 - 7

نكته:استفاده از زيربرنامه به عنوان پارامتر،در مواردي خوب است كه بخواهيم در داخل زير برنامه ، از زير برنامه ديگري استفاده كنيم كه در اين زير برنامه تعريف نشده است بلكه در خارج از آن تعريف شده باشد.يعني زير برنامه اي كه به عنوان پارامترp ميباشد ، در داخل زير برنامه p تعريف نشده است. بلكه در خارج از آن تعريف شده است.

9-11- محيطهاي مشترك

گاهي اوقات لازم است مجموعه اي از اشياء داده اي بين تعدادي از زير برنامهها مشترك شود.محيط مشترك جهت به اشتراك گذاشتن اين اشياء داده اي استفاده ميشود.محيط مشترك در يك بلوك حافظه قرار ميگيرد.هر زيربرنامه ميتواند اين محيط مشترك را اعلان كند واز اين طريق تمام اشياء داده موجود در اين بلوك (محيط مشترك) براي آن زير برنامه قابل مشاهده خواهد بود. اين محيط مشترك در زبانهاي مختلف به نامهاي متفاوت خوانده ميشود .

Fortran (common); C++,smalltalk (class);

Ada (package); C (extern);

PL/I (external);

به عنوان مثال به package زير در زبان Adaتوجه كنيد:

```
Package shared_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.
```

تعريف Pakege فوق ،يك ثابت (Table_size)، دو جدول (T1,T2) ويك مغيير صحيح(i) تعريف ميكند.كه با يكديگر گروهي از اشياء داده اي را تشكيل ميدهند كه در بسياري از زير برنامهها مورد نياز هستند.

اگر زير برنامه اي مثل P بخواهد به دادههاي اين پكيج دسترسي پيدا كند دستورwith به صورت زير نوشته ميشود:

```
With shared_tables;
```

از اين به بعد در بدنه P، هر نام موجود در پكيج مستقيماً قابل استفاده بوده و مانند آن است كه بخشي از بدنه P ميباشد.براي دستيابي به نامهاي موجود در پكيج ميتوان به صورت زير عمل كرد:

```
Shared_tables.T1;
```

چون زير برنامه ممكن است از چندين پكيج استفاده كند بنابراين براي دستيابي به نامهاي موجود در پكيج بايد نام پكيج همراه با نام شيء داده مورد نياز ذكر شود.

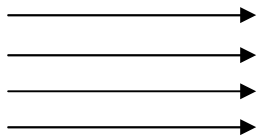
اشتراك صريح متغيرها:

ميتوان كاري كرد كه يك شي ء داده در محيط محلي يك زير برنامه، براي زير برنامههاي ديگر قابل دسترسي باشد.بنابراين به جاي اينكه دسته اي از متغيرها در محيط مشترك و جدا از زير برنامهها باشند، هر متغير يك مالك دارد و مالك آن زير برنامه اي است كه متغير در آن جا اعلان ميشود .

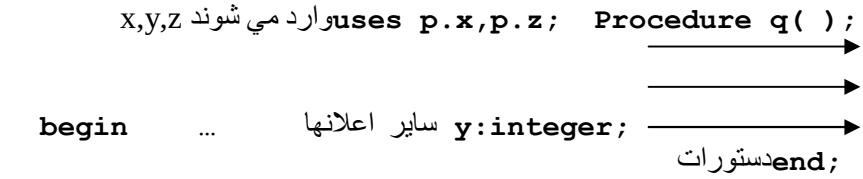
براي اينكه متغيري در خارج از زبر برنامه قابل دستيابي باشد از دستور تعريف صدور (export definition) استفاده ميشود مثل اعلان defines در تكه برنامه زير:

```
Procedure p( )
```

. براي صدور آماده مي شوندx,y,z



زيربرنامه ديگري كه ميخواهد به متغيرهاي مذكور دسترسي پيداكند از تعريف وارد كردن (import definition) استفاده ميكند مثلاً در تكه كد زير اعلان براي اين كار ميباشد.



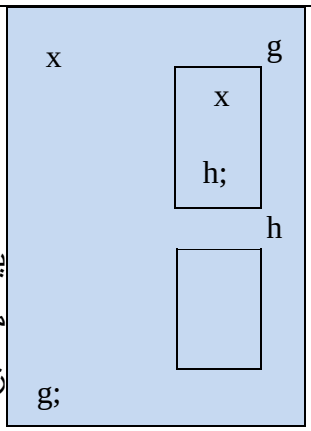
اين مدل در C با استفاده از extern پياده سازي مي شود.

21-9- پياده سازي حوزه ايستا و پويا

حوزه پويا:

به جاي استفاده از محيط مشترك ميتوان از وابستگيهاي محيط غير محلي با هر زيربرنامه در حال اجرا استفاده كرد.محيط غير محلي براي زيربرنامه p، متشكل از مجموعه اي از محيطهاي محلي سابقههاي فعاليت ساير زيربرنامههايي است كه در حين اجرا به p دستيابي دارند.وقتي در زيربرنامه p به متغير x رجوع شود و x وابستگي محلي نداشته باشد از محيط غير محلي براي تعيين وابستگي x استفاده ميشود .

در روش حوزه ارجاعي پويا ، اغلب از قانون»تازه ترين وابستگي[1]« استفاده ميشود.در اين قانون،اگربه طور مثال ،تابع f ، تابع g را صدا بزند و g هم h را صدا بزند و متغير x ، هم در f و هم در g تعريف شده باشدولي در h تعريف نشده باشد، هنگام استفاده از x در h ،كنترل به x موجود در g ارجاع ميشود يعني در اين قانون متغير به نزديك ترين تابع در زنجيره فراخوانيها ارجاع ميشود(. f—›g—›h)



شكل 8 - 9

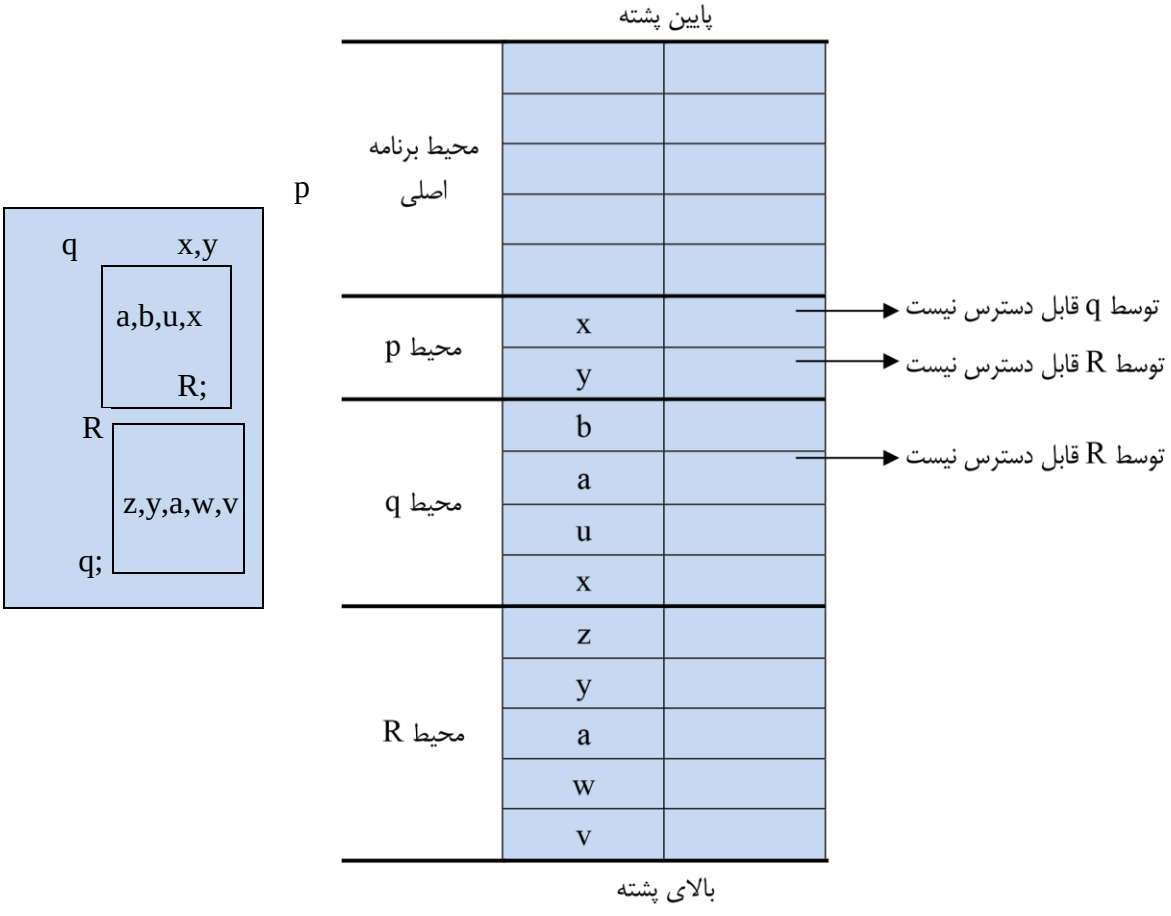
پياده سازي:

يكي از روشهاي پياده سازي ارجاع پويا،استفاده از پشته مركزي است.مثال زير اين روش را نشان ميدهد.فرض كنيد زيربرنامه p ،زيربرنامه q را صدا بزند و زيربرنامه q نيز زيربرنامه r را صدا بزند.هنگامي كه r اجرا ميشود پشته مركزي به شكل زير خواهد بود.جهت پردازش ارجاع غير محلي x ،پشته با شروع از محيط محلي r به طرف عقب

---

[1] Most Resent Association

جستجو ميشود تا تازه ترين وابستگي براي x پيدا شود.در اين حال برخي از وابستگيهاي موجود در پشته وابستگيهاي بعدي براي همان شناسه،مخفي است.(p→q→r)



شكل ۹ – ۹

حوزه ايستا و ساختار بلوكي:

در زبانهايي مانند پاسكال وAda كه از ساختار بلوكي استفاده ميكنند پردازش ارجاعهاي غير محلي پيچيده تر است.در شكل صفحه بعد كه نمونه اي از قواعد حوزه ايستا را براي برنامه ساخت يافته بلوكي در پاسكال نشان مي -دهد زيربرنامه r از q فراخواني ميشودو زيربرنامه q نيز از p فراخواني ميشود.زيربرنامههاي p و q و main متغير x را تعريف ميكنند.در داخل r ، x به طور غير محلي ارجاع ميشود.طبق قاعده حوزه پويا،هنگام اجراي r ، متغير x بايد متغير تعريف شده در q باشددر حالي كه طبق قاعده حوزه ايستا متغير x به x موجود در برنامه main اشاره دارد و بايد هنگام اجراي دستور x=x+1 از آن استفاده شود لذا سازگاري بين قواعد حوزه ايستا و پويا لازم است.

براي ايجاد سازگاري ،بايد حوزه ايستا در زمان اجرا كنترل شود و در هر ركورد فعاليت يك SCP[1] ذخيره شوده و آدرس AR (ركورد فعاليت) قبلي را نگه ميدارد.از اين طريق ميتوان زنجيره ايستاي برنامه را دنبال كرد.در اين

---

[1] Static Chain Point

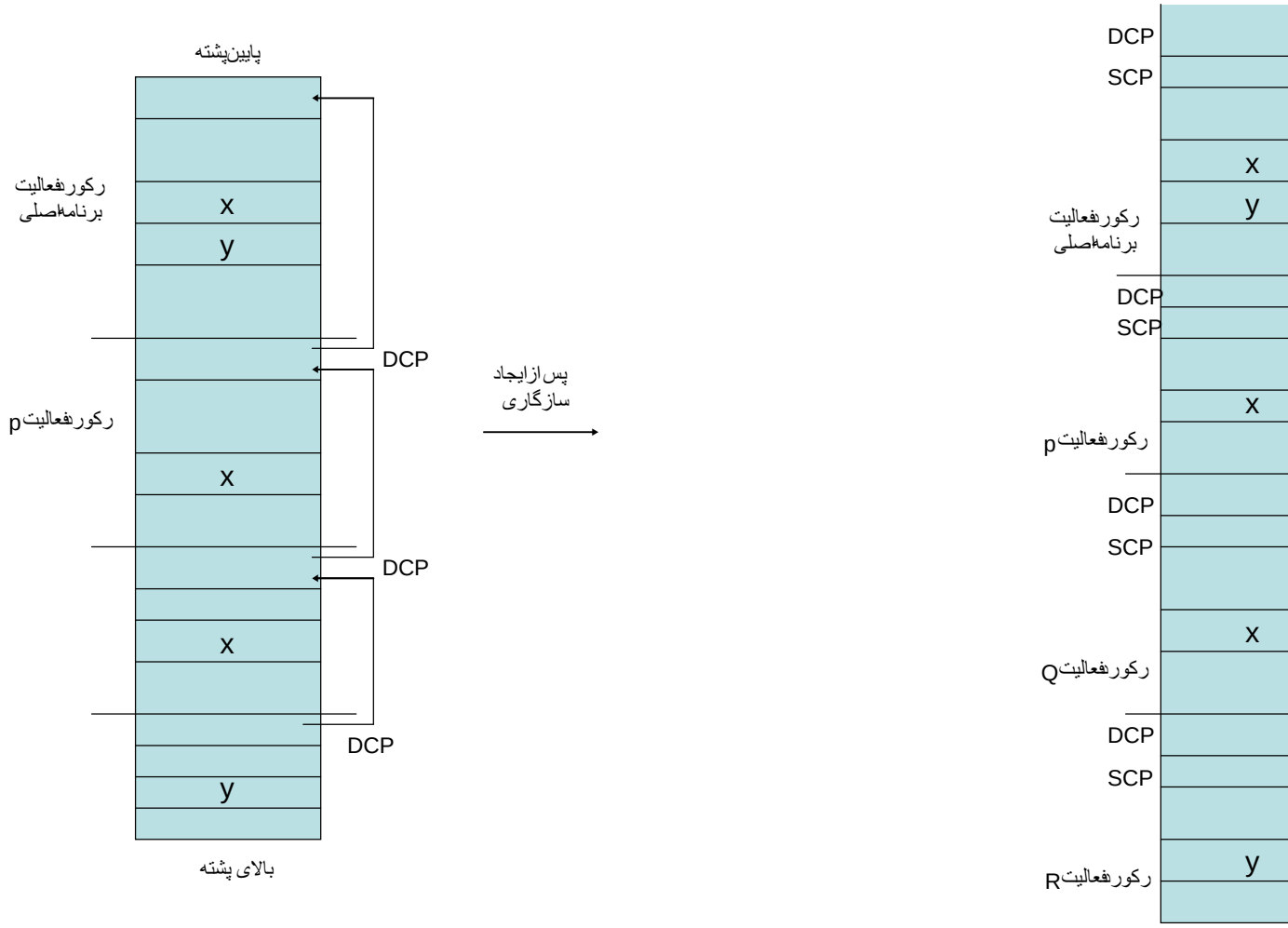
مثال،به هنگام اجراي r،با توجه به SCP كه به ركورد فعاليت برنامه main اشاره ميكند و با استفاده از يك آفست

(تفاوت مكان)،متغير x موجود در برنامه main در اختيار x قرار ميگيرد. برنامه زير را در نظر بگيريد:

```
Program main;
var x,y:
procedure integer
var y:        R
begin      real
x:=x+1
end;
procedure Q
var x :real
begin
R;
end
Procedure P   var  x:
Boolean  begin
Q;
end
```

```
begin
    P;
End;
```

براي درك مفاهيم زنجيره ايستا و پويا براي اجراي برنامه فوق شكل زير را مشاهده كنيد.



پشته تكميل شده در حين اجرا پشته ناقص مركزي در حين اجرا شكل 9 - 01

) اشاره گر زنجيره پوياDCP=Dynamic Chain Pointer (

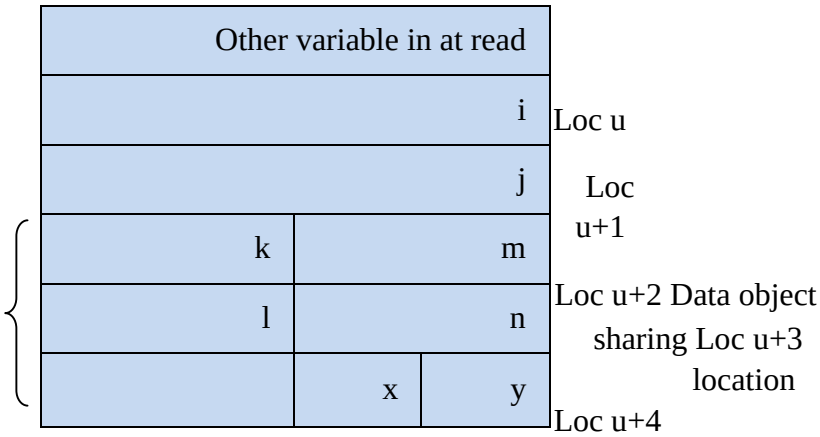
) اشاره گر زنجيره ايستاSCP=Static Chain Pointer (

9-31- اعلانها در بلوكهاي محلي

در زبانهايي مانند C ميتوان داخل هر بلاك از دستورات،يعني پس از هر آكولاد بازي ({) متغير تعريف كرد.اين متغيرها،متغيرهاي محلي آن بلاك ميباشندو در خارج از آن بلوك شناخته شده نيستند.براي پياده سازي اين اعلان -ها نميتوان براي هر بلوك،ركورد فعاليت جداگانه اي در نظر گرفت بلكه ميتوان از تكنيكي شبيه ساختار حافظه در ركوردهاي طول متغير كه قبلاً شرح داديم استفاده كرد مثال زير اين موضوع را نشان ميدهد.در تكه برنامه زير متغيرهاي K وL همانندn وm

از يك محل حافظه هستند چرا كه نميتوانند همزمان فعال باشند و كل اين حافظه توسط زيربرنامه اي اختصاص مييابد كه آنها را در بر ميگيرد.

```
Real proc 1(parameters)
{
        int i,j;
    …/*statement*/
    { int k,j;    …/*statement*/ }
        int m,n;
    …/*statement*/
    { int x;      …/*statement*/ }
    {int y;       …/*statement*/ }
}
```



شكل 9 - 11 همپوشاني حافظه متغير در ركورد فعاليت

9-41- سوالات فصل نهم

سوالات تستي

1- كدام گزينه صحيح است؟ (نيمسال دوم 38)

الف. در روش فراخواني با ارجاع مقدار متغير جايگزين ميگردد .

ب .در روش فراخواني با مقدار آدرس متغير جايگزين ميگردد .

ج .انتقال پارامتر به روش مقدار و نتيجه در زبان الگول معرفي شده است.

د .در زبان C فراخواني با ارجاع وجود ندارد.

2- تناظر بين پارامترهاي واقعي و مجازي در كدام گزينه آمده است؟

(نيمسال دوم 38) الف. بر اساس نام- موقعيتي ب. بر اساس نام

- تقسيمي ج .موقعيتي- بر اساس آدرس د. بر اساس آدرس - تقسيمي

3- كدام گزينه جزء روشهاي انتقال پارامترها نميباشد؟ (نيمسال دوم 48)

الف. فراخواني بي نام ب. فراخواني با ارجاع ج .فراخواني با مقدار د. فراخواني با نام

4- عبارت »مجموعه اي از وابستگيها مربوط به شناسههايي كه در يك زير برنامه استفاده مي شود ولي هنگام ورود به آن ايجاد نمي شود« معادل كداميك از محيطهاي ارجاع زير است؟(نيمسال اول 68 -58) الف.محيط ارجاع محلي ب. محيط ارجاع عمومي ج.محيط ارجاع از پيش تعيين شده د. محيط ارجاع غير محلي **5-** انواع فراخوانيها عبارتند از:(نيمسال دوم 68-58)

الف.با ارجاع ب.با مقدار و نتيجه ج.با مقدار د.همه موارد.

6- در پياده سازي ساختارهاي كنترلي بين برنامهها و زير برنامهها نقش اشاره گرCEP چيست؟(نيمسال اول 86 - 87)

الف.اين اشاره گر به دستور جاري قابل اجراي يك زير برنامه اشاره مي كند.

ب.اين اشاره گر به ابتداي ركورد فعاليت يك زير برنامه اشاره مي كند.

ج.اين اشاره گر براي پياده سازي ارتباط ساختاري بين دو زير برنامه استفاده مي شود.

د.همه موارد فوق صحيح است.

7- كداميك از موارد زير، از مولفههاي محيط ارجاع يك زير برنامه است؟(نيمسال اول 68-78) الف.محيط ارجاع محلي ب.محيط اجراي غير محلي ج.محيط ارجاع از پيش تعريف شده د.همه موارد

8- كدام يك ازموارد زير در مورد پياده سازي زير برنامهها صحيح نمي باشد؟(نيمسال اول 68 -78) الف. هر زير برنامه كه فراخواني مي شود يك activation record براي آن ايجاد مي شود.

ب .به ازاي هر فراخواني جديد يك activation record جديد ايجاد مي شود.

ج .امكان ندارد چندين activation record از يك زير برنامه در برنامه وجود داشته باشد.

د .activation record نوعي شي داده است كه به صورت بلوكي از حافظه نشان داده مي شود.

9- چنانچه قاعده كپي (Copy Rule) در پياده سازي فراخواني برگشت براي زير برنامهها در نظر گرفته شود كداميك از موارد زير رخ مي دهد؟(نيمسال دوم 68-78)

الف. زير برنامهها نمي توانند بازگشتي باشند و همچنين فراخواني نياز به دستور فراخواني صريح دارد.

ب .زير برنامهها در فراخواني بايد به طور كامل اجرا شوند.

ج .نمي توان زير برنامههاي هم روال داشت .

د .هر سه گزينه رخ مي دهد.

01- يك برنامه فرعي با يك پارامتر از نوع آرايه اي به طول 001 از اعداد صحيح را در نظر بگيريد. هزينه انتقال آرايه مزبور به داخل برنامه فرعي را وقتي يكي از سه روش انتقال پارامتر مورد استفاده قرار مي گيرد.در كدام گزينه صحيح مقايسه شده است؟(نيمسال دوم 68 -78)

الف value> value_ result=references .

ب value-result> value>reference.

جreference= value-result> value.

دvalue >value_result>reference.

11- خروجي برنامه زير را به صورتي كه مكانيزم تبادل پارامتر call by value و call by value- result و call by reference و call by name باشد كدام گزينه است؟(نيمسال دوم 68 -78)

```
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,
Write (k);  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

21- قطعه برنامه زير را در نظر گرفته و خروجي را بر اساس مفهوم نگهداري در فراخواني زير برنامه مشخص كنيد؟(نيمسال دوم 68 -78)

```
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. 33,23,13,03 ب.
30,03,03,03 ج. 31,03,13,03
د. 32,13,13,03

31- تناظر بين پارامترهاي مجازي و واقعي به چه روشي صورت ميگيرد؟(نيمسال دوم 68 -78) الف.تناظر موقعيتي ب.تناظر بر اساس نام
ج.تناظر طبق نمايش درختي د.الف وب

14- منظور از پارامتر واقعي در بحث فراخواني زير برنامهها چيست؟(نيمسال دوم 68-78) الف.يك شي داده كه با زير برنامه فراخوان مشترك است.
ب.يك شي داده است كه ممكن است پارامترهاي مجازي فراخوان باشد.
ج.يك شي داده غير محلي مي باشد كه ممكن است توسط فراخوان قابل مشاهده باشد.
د .هر سه مورد فوق

15- در پياده سازي ساختارهاي كنترلي بين برنامهها و زير برنامهها نقش اشاره گر CIP چيست؟(نيمسال دوم 86 - 87)

الف.اين اشاره گر به دستور جاري قابل اجراي يك زير برنامه اشاره مي كند.

ب.اين اشاره گر به ابتداي ركورد فعاليت يك زير برنامه اشاره مي كند.

ج.اين اشاره گر براي پياده سازي ارتباط ساختاري بين دو زير برنامه استفاده مي شود.

د .همه موارد فوق صحيح است.

16- كدام گزينه صحيح است؟(نيمسال دوم 68-78)

الف.نام مستعار مشكلاتي را براي برنامه نويس بوجود مي آورد.

ب.طراحي زبانهاي جديد سعي زيادي در استفاده از نام مستعار دارند.

ج.نام مستعار مشكلاتي را براي پياده سازي زبان بوجود نمي آورد.

د .هر سه گزينه

17- اگر براي فراخواني زير برنامهها و توابع ديدگاه قاعده كپي(Copy Rule) مطرح باشد كدام يك از موارد زير بوجود مي آيد؟(نيمسال اول 78-88) الف .زير برنامهها مي توانند باز گشتي باشند.

ب .زير برنامههاي هم روال مي توانند اجرا شوند.

ج .تمامي متغيرهاي محلي و غيرمحلي هم نام خواهند بود.

د .پردازش استثناها امكان پذير نمي باشد.

18- در كدام زبان زير برنامهها به طور پيش فرض بازگشتي در نظر گرفته نمي شود و در صورت بازگشتي بودن از كلمه Recursive استفاده مي شود؟(نيمسال اول 78-88)

الف ب.Fortran ج.PL/I د.C Pascal.

19- تناظر بين پارامترهاي واقعي و مجازي به كدام روش صورت مي گيرد؟

الف. تناظر موقعيتي و تناظر بر اساس نام ب .تناظر درختي و تناظر نوع ج .تناظر بر اساس نام و تناظر آينه اي د.تناظر نوع و تناظر ساختاري

02- قطعه برنامه زير را در نظر گرفته و خروجي را بر اساس مفهوم نگهداري در فراخواني زير برنامه مشخص كنيد؟(نيمسال اول 78 -88)

```
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;
```

الف30.و31و32و
33
ب30.و30و30و0
3
ج30.و31و30و1
3
د30.و31و31و23

12- كداميك از موارد زير پارامتر واقعي براي زير برنامه فراخواني شده است؟(نيمسال اول 78 -88) الف. شي داده محلي متعلق به فراخوان باشد يا پارامترهاي مجازي فراخوان باشد.

ب .شي داده غير محلي باشد كه توسط فراخوان قابل مشاهده باشد.

ج .نتيجه اي است كه توسط تابعي برگردانده شده است كه زير برنامه فراخوان آن تابع را فراخواني كرده و نتيجه را به زير برنامه فراخواني شده ارسال كرده است.

د .هر سه گزينه صحيح است.

22- در پياده سازي ساختارهاي كنترلي بين برنامهها و زيربرنامهها نقش اشاره گرCEP چيست؟(نيمسال اول 87 - 88)

الف. اين اشاره گر به دستور جاري قابل اجراي يك زير برنامه اشاره مي كند.

ب .اين اشاره گر به ابتداي ركورد فعاليت يك زير برنامه اشاره مي كند.

ج .اين اشاره گرها براي پياده سازي ارتباط ساختاري بين دو زير برنامه استفاده مي شود.

د .همه موارد فوق صحيح است.

23- خروجي برنامه زير به صورتي كه مكانيزم تبادل پارامتر و نتايج به صورتهاي call by name, call by reference ,call by value-result, call by value باشدكدام گزينه است؟(نيمسال اول 87 -88)

```
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,
Write (k); k);
  End;
```

Call by value	Call by value result	Call by reference	Call by name
200	100	900	900
300	100	900	6
200	100	900	6
100	300	900	900

الف ب ج د

24- يك برنامه با يك فرعي پارامتر از نوع آرايه اي به طول 1000 از اعداد صحيح را در نظر بگيريد. هزينه فراخواني با كداميك از استراتژيهاي زير بيشتر از بقيه است؟(نيمسال دوم 78 -88) الف.فراخواني با مقدار ب.فراخواني با مقدار و برگشت نتيجه ج.فراخواني با ارجاع د.فراخواني با نام

25- در پياده سازي ساختارهاي كنترلي بين برنامهها و زير برنامهها نقش اشاره گرCIPچيست؟(نيمسال دوم 87 - 88)

الف. اين اشاره گر به دستور جاري قابل اجراي يك زير برنامه اشاره مي كند.

ب .اين اشاره گر به ابتداي ركورد فعاليت يك زير برنامه اشاره مي كند.
ج .اين اشاره گر براي پياده سازي ارتباط ساختاري بين دو زير برنامه استافاده مي شود.
د .همه موارد فوق صحيح است.

62- در كداميك از زبانهاي زير آرايههاي پارامتري وجود دارند؟(تابستان 88)

الف - Pascal ب - Cobol ج - Ada د - C

72- كداميك از موارد زير جزء امتيازات goto است؟(تابستان 88)

الف - توسط سخت افزار پشتيباني ميشود ب- درك برنامه راحت تر است ج - تعريف ركوردها راحت تر است د - تعريف بردارها راحت تر ميشود

82- عبارت »مجموعه اي از وابستگيهاي مربوط به شناسههايي كه در يك زيربرنامه استفاده ميشوند ولي هنگام ورود به آن ايجاد نميشوند« معادل كداميك از محيطهاي ارجاع زير است؟(تابستان 88)

الف - محيط ارجاع محلي ب- محيط ارجاع عمومي ج - محيط ارجاع از پيش تعيين شده د- محيط ارجاع غير محلي

92- در پياده سازي ساختارهاي كنترلي بين برنامهها و فاصله زيربرنامهها نقش اشاره گر CEP چيست؟(تابستان 88)

الف – اين اشاره گر به دستور جاري قابل اجراي يك زيربرنامه اشاره ميكند ب - اين اشاره گر به ابتداي ركورد فعاليت يك زيربرنامه اشاره ميكند

ج - اين اشاره گر براي پياده سازي ارتباط ساختاري بين دو زيربرنامه استفاده ميشود .

د- اين اشاره گر براي پياده سازي قاعده كپي (Copy) استفاده ميشود .

03- ميدانيم كه محيطهاي مشترك صريح براي به اشتراك گذاشتن اشيا ء داده به كار ميرود كداميك از زبان-هاي زير از كلاسها براي تعريف اين ويژگي استفاده ميكنند ؟(تابستان 88)

الف ب. C++ Ada .

ج د .Fortran C.

13- پياده سازي قاعده تازه ترين وابستگي براي ارجاع غير محلي توسط كداميك از ساختمان دادههاي زير ساده تر است؟(تابستان 88)

الف. گراف ب. صف ج. پشته د. آرايه

23- دركداميك از روشهاي انتقال پارامتر با پارامترهاي واقعي همانند زير برنامههاي فاقد پارامتر، عمل مي كند؟(نيمسال اول 88-98)

الف.فراخواني با نام ب.فراخواني با ارجاع ج .فراخواني با مقدار د. فراخواني با مقدار-نتيجه

33- برنامه زير در زبان پاسكال را در نظر بگيريد. كداميك از تفسيرهاي زير در مورد اين برنامه درست است؟(نيمسال اول88-98)

```
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. فراخواني در محل2 ، S موجود در محل 3 را فراخواني مي كند و برنامه اجرا مي شود.
ب .برنامه ترجمه نمي شود زيرا فراخواني S در محل 2 يك ارجاع پيشرو فاقد اعلان مي باشد.
ج .برنامه ترجمه مي شود و فراخواني S در محل 2 يك ارجاع پيشرو معتبر را ايجاد مي كند.
د .فراخواني زير برنامه در محل 2 زير برنامه Sموجود در محل 1 را فراخواني مي كند.

43- كداميك از زبانهاي زير براي ركورد فعاليت هر زير برنامه حافظه بطور ايستا اختصاص مي يابد؟
(نيمسال دوم 88 -98)

Ada. د .Lisp ML. ج ب. Fortran الف

53- پياده سازي كداميك از ساختارهاي زبان C شبيه به ساختار حافظه ركورد متغير است؟
(نيمسال دوم 88 -98) الف.اعلانها در بلوكهاي محلي ب.زيربرنامههاي همروال ج.زير برنامههاي فراخواني بازگشت د.زير برنامههاي بازگشتي

63- كداميك از فراخوانيهاي زير در زبان C++ درست است؟(نيمسال دوم 88 -98) الف. Q((&A+B),&B) ب. Q((A+B),&B)

. Q(&(A+B),&B) د. Q((&A+&B),&B) ج

73- محيط ارجاع مربوط به نام يك پروسيجر در ساختار بلاكي ايستا، در كدام بلاك قرار مي گيرد؟ (نيمسال دوم 88-89)

الف.بلوكي كه آن بلاك را در بر مي گيرد ب.محيط محلي همان بلاك ج.بلاك برنامه اصلي د.بلاك هم سطح آن بلاك

83- كداميك از اشياء اشاره گر زير در ركورد فعاليت يك زير برنامه، آدرس نقطه بازگشت دستور بعد از فراخواني آن زيربرنامه را نگهداري مي كند؟(نيمسال دوم 88 -98) الف.CEP ب.CIP ج.ip د.ep

39- در زبان پاسكال، براي فراخوا ني زير برنامه بصورت proc(v[i],I,10,20)، با توجه به روشهاي انتقال پارامترها، نوع پارامتر X و Y به ترتيب چيست؟(نيمسال دوم 88-98)

```
Procedure proc(arr,index : X; LB,UB: Y)
        Var temp:integer;
    For index:=LB     Begin
                   TO UB do
      temp:=temp+arr;
         Write(temp);
                    End;
```

الف .X فراخواني با نام- Y فراخواني با مقدار

ب. X فراخواني با مقدار- Y فراخواني با نام ج

.X فراخواني با ارجاع- Y فراخواني با مقدار

د .X فراخواني با مقدار ثابت - Y فراخواني با مقدار

04- روش نگهداري محيط ارجاع محلي به برنامه نويس اجازه مي دهد برنامههايي بنويسد كه : (نيمسال دوم 88 - 89)

الف. اثرات جانبي داشته باشند ب. حساس به گذشته باشند

ج .به آرگومانهاي ضمني دسترسي داشته باشند د .براي ورودیهاي خاصي قابل تعريف نباشند

14- پس از فراخواني زير برنامه R توسط زير برنامه P خروجي مقدار c[m] چيست؟(از راست به چپ) (نيمسال دوم 88-89)

```
} R(int *i,int *j) {
        *i=*i+1;
         *j=*j+1
int c[2];     } P(){
        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.7 و

7 ب. 7

و 8 ج.

6 و 7 د.

6 و 8

42- در كداميك از موارد زير قاعده كپي صدق ميكند(.نيمسال اول 98 -09)

الف. زيربرنامههاي بازگشتي مستقيم ب .زيربرنامههاي بازگشتي غيرمستقيم ج .همروالها د .زيربرنامههاي فراخواني برگشت

43- در دستور x=2*y+3/z اشيا و داده اي موجود از چه روش عملوندي در عمليات استفاده ميكنند.(نيمسال اول 89-90)

الف. شي داده با نام ب .انتقال مستقيم ج .انتقال غير مستقيم د .شي داده اشاره گر

44- در تكه كد برنامه زير چه محيطهاي ارجاعي وجود دارد.(نيمسال اول 98-09)

```
Int r; int
 f(int a)
int b;    {
b=sqrt(a+r);
return b;
} int main(
)
```

```
f ( );     {
return
0;
}
```

الف. ارجاع محلي و ارجاع غير محلي

ب .ارجاع محلي و ارجاع عمومي

ج .ارجاع محلي و ارجاع غير محلي و ارجاع از پيش تعريف شده د .ارجاع محلي و ارجاع عموميو ارجاع از پيش تعريف شده

45- كدام گزينه صحيح است.(نيمسال اول 98 -09)

الف. براي محيطهاي ارجاع غيرمحلي قواعد حوزه ايستا و پويا سازگارند.

ب .براي محيطهاي ارجاع محلي قواعد حوزه ايستا و پويا سازگارند.

ج .براي محيطهاي ارجاع عمومي قواعد حوزه ايستا و پويا سازگارند .

د .براي محيطهاي ارجاع از پيش تعريف شده قواعد حوزه ايستا و پويا سازگارند.

46- كدام يك از زبانهاي زير از روش نگهداري براي محيطهاي محلي استفاده ميكنند.(نيمسال اول 98-09) الف. ادا ب .اسنوبال 4 ج .كوبول د .ليسپ

47- دركدام يك از ساختارهاي زير روشهاي نگهداري و حذف پياده سازي يكساني دارند.(نيمسال اول 98 -09) الف. همروالها ب .فراخواني – برگشت بدون بازگشتي ج .بازگشتي د .زمان بندي شده

48- كدام يك از فراخوانيهاي زير در زبان C++ درست است.(نيمسال اول 98-09) الف. Q((&A+B),&B) ب .Q((A+B),&B)

ج Q((&A+&B),&B). د Q(&(A+B),&B).

49- محيط ارجاع مربوط به نام يك پروسيجر در ساختار بلاكي ايستا در كدام بلاك قرار دارد.(نيمسال اول 98 -09) الف. بلوكي كه آن بلاك را دربرميگيرد. ب .محيط محلي همان بلاك ج .بلاك برنامه اصلي د .بلاك هم سطح آن بلاك

50- كدام يك از موارد زير ميتواند يك نوع پارامتر ضمني تلقي شود..(نيمسال اول 98-09) الف. مقدار برگشتي توابع ب .مقدار برگشتي روال ج .مقدار برگشتي ارجاع د .هرنوع مقدار برگشتي

51- كدام يك از اشيا اشاره گر زير در مورد فعاليت يك زيربرنامه ، آدرس نقطه بازگشت دستور بعد از فراخواني آن زيربرنامه را نگهداري ميكند(..نيمسال اول 98 -09)

الف CEP . ب CIP. ج ip. د Ep.

52- پياده سازي اعلانها در بلاكهاي محلي در زباني مانند C شبيه به كدام ساختار زير است.
(نيمسال اول 98-09) الف. ركورد متغير ب .ركورد تودرتو ج .آرايه اي از ركورد د .زيربرنامه

9-51- پاسخنامه سوالات تستي فصل نهم

سوال	الف	ب	ج	د	ال	الف	ب	ج	د
1			*		27	*			
2	*				28				*
3	*				29		*		
4				*	30	*			
5				*	31			*	
6		*			32	*			
7				*	33		*		
8			*		34	*			
9				*	35	*			
10		*			36				*
11				*	37	*			
12				*	38			*	
13				*	39	*		*	
14				*	40		*		
15	*				41	*			
16	*				42				*
17				*	43		*		
18		*			44				*
19	*				45		*		
20				*	46			*	
21				*	47		*		
22	*				48		*		

			*	49				*	23
			*	50			*		24
	*			51				*	25
			*	52					26

سوالات تشريحي

1- نحوه پياده سازي زير برنامههاي بازگشتي در زبانهاي برنامه سازي را بطور كامل شرح دهيد. (نيمسال اول 85 - 68)

2- پارامترهاي واقعي و مجازي را با يكديگر مقايسه كنيد. چه تناظرهايي بين اين دو پارامترها امكانپذير است. (نيمسال اول 58-68)

3- خروجي برنامه زيررا در حوزه ارجاعي ايستا و پويا بدست آوريد. (نيمسال اول 68 -78)

```
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.
```

4- برنامه زير را در نظر گرفته و محيطهاي ارجاع local و non-local را براي main,sub1,sub2 بنويسيد؟(نيمسال دوم 68 -78 و نيمسال دوم 78-88)

```
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.
```

5- برنامه زير را در نظر بگيريد و مراحل اجراي اين برنامه را در هر يك از زمانهاي زير در پشته مركزي نشان
دهيد؟(نيمسال دوم 68-78 و نيمسال اول 78 -88)

```
                                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;
```

P الف.اجراي main قبل از فراخواني

Qب.اجراي P قبل از فراخواني

ج.اجراي Q قبل از فراخوانيR

د.Q, FV را صدا مي كند.

6- پديده نام مستعار را به همراه يك مثال شرح دهيد؟(نيمسال اول 78 -88)

8- اعلان پيشرو در پاسكال ناهنجاري بوجود مي آورد آن را به همراه يك مثال شرح دهيد؟(نيمسال اول 78-88)

9- زير برنامه اي به صورت زير اعلانهايي در بلوكهاي محلي دارد نمايش حافظه مربوطه به اين زير برنامه را براي متغيرهاي مربوطه به گونه اي رسم كنيد كه در ركورد فعاليت بدون هيچ مشكلي عمليات فراخواني را با كمترين حافظه مصرفي داشته باشيم؟(نيمسال دوم 78-88)

```
Float proc1 (parameter) {
        Int j;
{Int k, l ;}
{Int m, n ;}
   {Int x ;}
   {Int y ;}
```

11- خروجي برنامه زير را در هر يك از حالتهاي ارسال پارامتر با مقدار مقدار نتيجه و ارجاع مشخص كنيد؟(نيمسال اول 88 -98)

```
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;
```

21- خروجي شبه كد زير را در حوزه ايستا و پويا مشخص كنيد؟(نيمسال اول 88-98)

```
                    Begin
              Int x: =2;
         Procedure p ();
                  Begin
          Write(x);
                   End;
                  P ();
                 Begin
        Int x: =3;
             P ();
                   End;
                  P ();
                       End;
```

31- پديده نام مستعار را به همراه يك مثال شرح دهيد؟(نيمسال اول 88 -98)

41- اشتراك داده از طريق محيط مشترك صريح را بطور كامل مطرح كنيد. (نيمسال دوم 88-98)

51- خروجي برنامه زير را در حين اجرا به دو صورت: (نيمسال دوم 88-98) الف.هنگامي كه حوزه ايستا است، مشخص كنيد. ب .هنگامي كه حوزه پويا است، مشخص كنيد.

```
                    Program main;
              Var i,a,k,m :integer;
Procedure Q(m:integer; var i:integer);
```

```
                                    i:=i+k;                 Begin
                                Writeln('in Q:',              m:=a+2;
                                                            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;
                                          i:=1;    Begin {main}
                                                     a;=2; k:=3;
                                                    P(k,i);
                                  Writeln('in main:', i,a,k);
                                                          End.
```

16- اشتراك داده از طريق حوزه ايستا را به طور كامل و با مثال تشريح كنيد.(نيمسال اول 98-09)

پيوست: سوالات كنكور سراسري كارشناسي ارشد

سال 1382

1- مجموعه مقادير ممكن براي يك متغير از نوع صحيح (integer) در چه مرحله اي تعيين مي گردد؟

(Translation of programs) الف.ترجمه برنامهها

(Implementation of languages) ب.پياده سازي زبانها

(Execution of programs) ج.اجراي برنامهها

(Definition of languages) د.تعريف و تبيين زبانها

2-در زبان فرضي زير، آرگومانهاي برنامهها بصورت Call by Name تعريف شده اند. با توجه به اين روش تعريف آرگومان، خروجي تكه برنامه زير چيست؟

```
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]);
```

الف2.و4و8و2 (از چپ به راست بخوانيد.) ب4.و4و6و8و2 (از چپ به راست بخوانيد.) ج2.و4و6و8و4 (از چپ به راست بخوانيد.)

د2.و4و4و8و4 (از چپ به راست بخوانيد.)

3- قطعه كد زير(به زبان C)را در نظر بگيريد:

```
  int *p; int
*q; q ="abcd"
     p
=malloc(10);
```

```
q =p; free(p);
strcpy(q,"123");
```

در اين صورت مي توان گفت:

الف. در اجراي اين قطعه كد، ارجاع معلق (Dangling Reference) بوجود مي آيد.

ب .در اجراي اين قطعه كد ، Garbage بوجود مي آيد.

ج .در اجراي اين قطعه كد ، Fragmentation بوجود مي آيد.

د .در اجراي اين قطعه كد ،Garbage و ارجاع معلق (Dangling Reference) بوجود مي آيد.

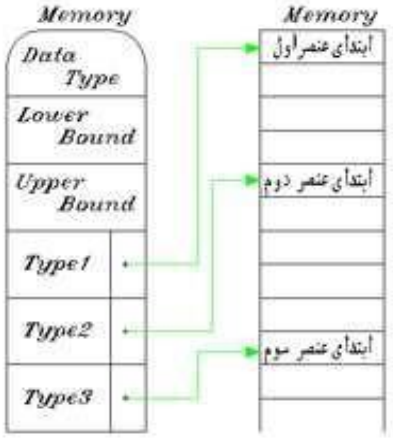
4- با فرض اينكه شكل زير نمايش حافظه مربوط به پياده سازي داده ساختيافته اي در يك زبان برنامه سازي را نشان دهد كه در زمان اجرا نيز بتوانيم نوع هر يك از عناصر آن را تغيير دهيم، كداميك از گزينههاي زير صحيح است؟

الف.نقطه ضعف اساسي و مهم اين دادهها ساختيافته مكانيزم لازم براي انتخاب عناصر آن است به طوريكه عملاً آن را ناكارامد مي سازد.

ب.پياده ساز زبان با انتخاب اين روش اولاً انعطاف پذيري بالايي براي برنامه ساز حاصل ميكند و ثانياً قيد صريح نوع هر عنصر قابليت اعتماد زبان مورد نظر را نيز بالا مي برد.

ج .در برخي از زبانهاي برنامه سازي مثل SNOBOL4 آرايهها را، عمدتاً با هدف بالا بردن خاصيت انعطاف پذيري (Flexibility) براي برنامه ساز، با استفاده از اين روش پياده سازي مي كنند.

د.اگر اين شكل نمايش حافظه اي مربوط به پياده سازي آرايه خاصي باشد و نوع عناصر آن را در زمان اجرا بتوان عوض كرد آنگاه وجود حد پاييني و بالايي (Lower & upper Bound) در آن لزوماً بي معني است.



5- كداميك از معيارهاي زير براي انتخاب زمان مناسب در كاربردهاي توكار(Embedded Systems) اهميت بيشتري دارد؟

الف.يكنواختي ب.قابليت توسعه (Uniformity) (Extensibility)

ج.عموميت د.قابليت اطمينان (Generality) (Reliability)

سال 1383

6- در رابطه با مفهوم آزمون نوع (Type checking) كدام گزينه غلط است؟ الف. وجود يك شي داده اي مي تواند جزئي از عمل آزمون نوع آن تلقي شود. ب .آزمون نوع يعني تعداد و نوع آرگومانهاي عمليات در برنامه درست باشند.

ج .آزمون نوع پويا منجر به استفاده بهينه از حافظه و افزايش سرعت اجرا مي شود در حاليكه آزمون نوع ايستا نتيجه عكس دارد.

د .در زبانهايي كه اعلان صريح (explicit declaration) اجباري نيست ولي با استفاده از مكانيزم نتيجه گيري نوع (type inference) مي توان نوع عملوندها را تعيين كرد، آزمون نوع ايستا ممكن است.

7- در رابطه با مفهوم مقيدسازي (Binding) و زمانهاي مقيدسازي،كدام گزينه غلط است؟ الف.مقيدسازي ممكن است در زمان اجرا، ترجمه، پياده سازي زبان و يا تعريف زبان صورت پذيرد.

ب.در زبانهاي با مقيدسازي زودرس، قابليت انعطاف بيشتر و در زبانهاي با مقيدسازي ديردرس كارايي اجرا بهتر است.

ج.مقيدسازي يك عنصر برنامه به يك صفت خاص، به معني انتخاب يك صفت از مجموعه اي از صفات است.

د .دريك زبان با مقيدسازي زودرس (Early Binding) تقريبا تمام مقيدسازيها در زمان ترجمه انجام مي شود و دريك زبان با مقيدسازي ديررس (Late Binding) تقريبا تمام مقيدسازيها در زمان اجرا انجام مي شود.

8-كداميك از خواص زير در يك زبان برنامه سازي اصلي ترين نقش را در قابليت انتقال برنامهها به عهده دارد؟

الف .Abstraction ب. Encapsulation

ج .Explicit Declaration د.Information Hiding

9-براي ساختن يك درخت تصميم گيري (Decision tree) به منظور انتخاب يك دستور (statement) از ميان n دستور كداميك از ساختارهاي برنامه سازي زير مي تواند منجر به كارايي زمان اجرا بصورت O(1) شوند؟ فرض كنيد شرط انتخاب يك دستور، برابر يك عبارت با يك مقدار ثابت به ازاي هر دستور باشد.

الف.ساختن درخت با دستور if – then – else

ب.بدست آوردن كارايي O(1)امكان ندارد.

ج.ساختن درخت با دستور case يا switch

د .ساختن درخت با دستور if – then – else – endif و elsif در زبان Ada

01-كداميك از موارد زير به نوعي از inheritance نزديكتر است؟

الف .Static scope rule ب.Late binding

ج .History sensitivity of methods د.Strong typing

11- از ميان زبانهاي زير كداميك كلي ترين نوع چند ريختي (polymorphism) را ارائه مي دهند؟

الف M. د. C++ ج.Ada ب.LISP

21- در كداميك از روشهاي انتقال پارامتر، به ازاي هر پارامتر از نوع متغير ساده بيش از يك فيلد در ركورد فعاليت (activation record) برنامه فرعي لازم است در نظر گرفته شود؟

الف .by value ب.by result

ج.by reference د.هيچگاه نياز به بيش ازيك فيلد نيست.

سال 1384

31-كداميك از مفاهيم زير كمتر با هم سازگارند؟

الف.تعريف نوع متغيرها و ترجمه ب.Late binding و تفسير

ج Static Scope Rule و.Early binding و ترجمه Dynamic Scope Rule د

41- در كداميك از موارد زير مفهوم Encapsulation در يك زبان بصورت كامل رعايت نشده است؟ الف. فقط با برنامه نويسي به آن زبان بتوان يك نوع داده اي مثل يكي از انواع داده اي موجود (مثلا integer) را با يك نمايش حافظه اي خاص دلخواه برنامه نويس و overload كردن عمليات موجود روي integerها پياده سازي كرد.

ب .فقط با برنامه نويسي به همان زبان سطح بالا بتوان سطري يا ستوني بودن نمايش حافظه اي زمان اجراي آرايههاي دو بعدي را كشف كرد.

ج .فقط با برنامه نويسي به همان زبان بتوان يكي از عمليات (مثلا ضرب)، كه در اصل آن زبان با اپراتور مشخص (مثلاً *)وجود دارد، را بدون استفاده از آن اپراتور شبيه سازي كرد.

د .در همه موارد فوق Encapsulation بطور كامل رعايت شده است.

51- اگر محدوده ي اعتبار (Scope) نام يك زير برنامه فقط شامل محدوده ي خود آن زير برنامه باشد و قاعده ي حاكم بر زبان Static Scope rule باشد ، چه اشكالي پيش مي آيد؟ الف. هيچ اشكالي پيش نمي آيد.

ب .برنامه اصلي نمي تواند آن زيربرنامه را صدا زند.

ج .هيچگاه از داخل يك زيربرنامه ي در حال اجرا نمي شود زير برنامه ي ديگري را صدا زد.

د .وقتي يك زيربرنامه در حال اجراست نمي تواند خودش را بصورت بازگشتي صدا زند.

61-در پياده سازي كداميك از زبانهاي زير مي توان براي دسترسي به مقدار يك متغير ساده غير محلي مستقيماً از مكانيزم سخت افزاري Fetch operand تعبيه شده در اجراي يك instruction استفاده كرد؟ الف.فقط C ب.C , LISP ج.C , پاسكال د.C، پاسكال ، LISP

71- در يك پياده سازي از يك زبان برنامه سازي كه تمام تكنيكهاي انتقال پارامتر را پشتيباني مي كند متوجه مي شويم كه پياده سازي تكنيك by reference درست كار نمي كند. مي خواهيم در برنامه خود با استفاده از ساير تكنيكهاي انتقال پارامتر، تكنيك by reference را شبيه سازي كنيم

بطوريكه نتيجه ي اجراي برنامه شبيه سازي شده در حالت كلي كاملاً با نتيجه اجراي برنامه اصلي (اگر by reference درست كار مي كرد) يكسان باشد، كدام گزينه درست است؟

الف. اين شبيه سازي با تكنيكهاي ديگر امكان ندارد.

ب .انتقال by reference پارامترهاي عضوي از آرايه را مي توان با انتقال by value result شبيه سازي كرد .

ج .انتقال by reference پارامترهاي متغير ساده را مي توان با انتقال by name شبيه سازي كرد.

د .انتقال by reference پارامترها به هر شكلي كه باشند را مي توان با انتقال by value result شبيه سازي كرد .

81- اگر بخواهيم خطاي مقدار اوليه نداشتن يك متغير تعريف شده در داخل يك زيربرنامه را در زمان اجرا كشف كنيم، هزينه زمان اجرا خيلي زياد مي شود. اگر بخواهيم اين خطا را در زمان ترجمه كشف كنيم هزينه كشف آن در زمان ترجمه در دو زبان C و پاسكال چگونه مقايسه مي شود؟

الف .هزينه در زبان C كمتر است. ب. هزينه در زبان پاسكال كمتر است.

ج .نمي شود اين خطا را در زمان ترجمه كشف كرد. د .هزينه در هر دو زبان تقريباً يكسان است.

سال 1385

91-خروجي برنامه زير در صورتي كه مكانيزم تبادل پارامتر و call by ، call by value-result ، value

باشد كدام گزينه است؟ call by name وcall by reference

```
                    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

02- در بعضي از زبانها با حوزه ايستا مانند پاسكال مي توان اسم يك روال (procedure) مانند F را بصورت يك پارامتر به يك روال ديگر ارسال كرد. براي binding اسامي درون بدنه روال ارسال شده كداميك از روشهاي زير مناسب تر است؟

الف. binding در محيط تعريف F ب .binding در محيط ارسال پارامتر به F ج .binding در محيط فراخواني F د .هيچكدام

12- كداميك از زبانهاي زير جزو زبانهاي Early Binding محسوب مي شود؟

الف. LISP ب. Java ج .Smalltalk د .هيچكدام

22- خروجي برنامه زير در حالتي كه از قواعد static scoping و dynamic scoping استفاده شود، به ترتيب كدام گزينه است؟

308 پيوست

```
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.
```

dynamic scoping: 600 , static scoping: 400

الف dynamic scoping: 900 , static scoping: 400

ب. dynamic scoping: 900 , static scoping:

ج600. dynamic scoping: 400 , static scoping:

د900.

32- در يك زبان برنامه سازي مثل پاسكال تعريف برنامههاي فرعي در داخل برنامههاي فرعي ديگر مجاز است.برنامههاي فرعي C,B,Aو D در عمقهاي مختلف برنامه فرعي M تعريف شده اند. فرض كنيد زنجيره callها بصورت زير باشد: M->A->B->A->C->D و D بتواند از متغير x كه در A تعريف شده است بر اساس static scope rule استفاده كند.در آن صورت تودرتويي

static برنامههاي فرعي C,B,A و D چند حالت مي تواند داشته باشد؟ الف. 10

ب8 . ج4 . د 2 .

42- دوره ي حيات (life time) شيء داده اي A كه در يك تابع، مثلا بصورت int A ، تعريف شده است كدام است؟

الف.از زمان شروع اجراي تابع تا زمان پايان اجراي تابع

ب .از زمان اجراي دستورالعمل int A تا زمان پايان اجراي تابع ج. از زمان شروع اجراي برنامه اصلي تا پايان اجراي برنامه اصلي د .از زمان مقدار اوليه گرفتن شيء داده اي A تا زمان آخرين استفاده از آن

سال 1386

52- در زبانهايي كه اشاره گر (Pointer) ندارند در چه حالتي وقوع پديده ي همنامي يا نام مستعار(Aliasing) امكان ندارد؟

الف.اگر تنها تكنيك انتقال پارامتر در زبان مورد نظر by value باشد.

ب.اگر مجموعه متغيرهاي سراسري (global) برنامه تهي باشد.

ج.اگر زبان داراي تكنيك انتقال by reference نباشد.

د.هيچكدام از سه گزينه فوق صحيح نيستند.

62- زبانهاي زير را از نظر تفسيري يا كامپايلري بودن دسته بندي كنيد. دسته بندي درست را انتخاب كنيد.

Ada - VI , Small talk - V , LISP - IV , C++ - III , Java - II , Fortran - I

د.هيچكدام. III.II.I ج. V , IV , II ب , VI

الف V , IV .

72- در كداميك از موارد زير اگر تكنيك انتقال پارامتر by name يا by reference باشد نتيجه اجراي برنامه مي تواند متفاوت باشد؟

الف.پارامتر مربوطه متغير ساده (مثلاً A) است.

ب.پارامتر مربوطه عضو نامعيني از يك آرايه است. (مثلاً B[I]).

ج .پارامتر مربوطه عضو معيني از يك آرايه است.

(مثلاً B[5]) د .هرسه مورد

82- براي 3 دستور case به شرح زير، پياده سازي معروف به جدول پرشها(Jump table) مفروض است؟

310 پيوست

I, case I of 1:st1; 2:st2; 3:st3 end case

II, case I of 100:st1; 200:st2; 300:st3 end case

III, case I of 1:st1; 2:st2; 300:st3 end case . IIIكمتر است III از

IIو II از I الف. حجم كد

ب .سرعت اجرا و حجم كد هر سه دستور برابر است.

ج .سرعت اجراي دستور I ازIII بيشتر و سرعت اجراي III از II بيشتر است.

د .هيچكدام

92- در زباني كه قانون حوزه ي شناسايي ايستا حاكم است و تعريف تودرتوي برنامههاي فرعي (nested definition) ممكن است، برنامه اي نوشته ايم كه از يك برنامه ي اصلي حاوي تعريف دو برنامه فرعي غير تودرتو تشكيل شده است. يكي از برنامههاي فرعي حاوي برنامه فرعي ديگري است كه آن هم حاوي برنامه فرعي ديگري است. در يك لحظه از زمان اجراي داخلي ترين برنامه فرعي، پشته (stack) ركوردهاي فعاليت برنامههاي فرعي، حاوي 01 ركورد (از جمله ركورد برنامه اصلي) است.در اين لحظه محتواي چند ركورد فعاليت قابل دسترسي توسط برنامه ي در حال اجرا است؟

الف. 3 ب4 . ج10 . د .هيچكدام

03- در قطعه برنامه ي زير كه به زبان برنامه سازي ML نوشته شده است مقدار عبارت b*f(a,a) در دو حالتي كه زبان از قواعد حوزه ايستا (static scoping) و حوزه پويا (dynamic scoping) استفاده مي كند، كدام است؟

```
let   let a = 5,   b =10 ,  c = 7 in
        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 ايستا: 06 و حوزه پويا :43 ب .حوزه ايستا: 005 و حوزه پويا :340 ج .حوزه ايستا: 001 و حوزه پويا :86 د.هيچكدام

سال 1387

13- كدام مجموعه از گزينههاي زير شامل عبارتهاي صحيح از مجموعه عبارتهاي زير است؟

1. مجموعه مقاديري كه يك متغير از نوع integer مي تواند اختيار كند معمولا در زمان پياده سازي زبان تعيين مي شود.

2.ماشين مجازي زبان برنامه سازي X، ماشيني است كه برنامه به زبان X را اجرا مي كند.

3.بررسي ايستاي نوع باعث افزايش مصرف حافظه و كاهش سرعت اجراي برنامه مي شود. ولي بررسي پوياي نوع باعث كاهش مصرف حافظه و افزايش سرعت اجراي برنامه مي شود.

الف. 1 و2 ب. 1 و3 ج. 2 و3 د. 1 و 2 و 3

23- اين برنامه C را در نظر بگيريد:

```
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,
fun2( b=3;
);
} void fun2( ){
printf("%d %d %d\n",a, b, c);
}
```

الف. 3 2 1 در حوزه پويا 4 3 2 در حوزه ايستا ب. 3 2 3 در حوزه پويا 3 2 در حوزه ايستا ج . 3 2 1 در حوزه پويا 4 3 2 در حوزه ايستا د . 4 2 1 در حوزه پويا 4 2 1 در حوزه ايستا

33- اصطلاحات زير مفروضند. گزينه اي را انتخاب كنيد كه اصطلاحات مربوط به آن يكديگر را تداعي كنند يا به عبارت ديگر از جنبه اثباتي به هم مرتبط باشند.

1.Early binding 2.Interpretation 3.Late binding

Execution.5 Translation.4

4الف. 1 و2 ب. 1 و4 ج. 1 و5

د. 3 و

43- در مورد Static Type Checking (STC) كداميك از گزينههاي زير غلط است؟ الف. اگر STC نباشد حجم كد توليد شده در اثر ترجمه برنامه خيلي زياد مي شود.

ب .STC باعث مي شود كه نوع عمليات پلي مرفيك در زمان كامپايل معين شود.

ج. STC باعث مي شود تابسياري از خطاهاي برنامه در زمان كامپايل كشف شود.

د. STC فقط براي قسمت تعاريف برنامه انجام مي شود و به قسمت اجرايي برنامه كاري ندارد.

53- برنامه زير در دو حالت تبادل پارامتر بصورت by reference و by value result مفروض است. زبان برنامه تابع قواعد حوزه ايستا (static scope rule) است. خروجي برنامه كدام 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);
                          x:=6*B;y:=x-   begin
                         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
```

4, 1, 19, 20, 3, 10, 5 by ref

4, 1, 30, 20, 3, 15, 19 by value- الف.

result

4, 1, 19, 20, 3, 15, 5 by ref

4, 1, 30, 20, 3, 15, 19 by value- ب.

result

4, 1, 19, 20, 3, 10, 5 by ref

4, 1, 10, 20, 3, 30, 19 by value- ج.

result

. 4, 1, 19, 20, 3, 15, 5 by ref د

4, 1, 10, 20, 3, 30, 19 by value-result

1388سال

63-در هنگام اجراي يك برنامه هرگاه سرريز (overflow) عمل جمع رخ دهد، قطعه كدي كه ترجمه ي يك روال exception handling است و توسط برنامه نويس به زبان سطح بالا نوشته شده است، اجرا مي شود. چه

عواملي از ميان عوامل زير در كشف، توليد و فعال كردن قطعه كد مربوطه مي توانند دخيل باشند؟ 1.برنامه نويس 2. كامپايلر 3. سيتم عامل

4. سخت افزار الف1.و2و4 ب1.و2و3 ج1.و2و3و4

د. 2و3و4 **73**- در مورد كنترل تقدم اپراتورها در عبارتهاي ميانوندي (Infix) راههاي زير پيشنهاد مي شود:

1.پرانتزگذاري كامل توسط برنامه نويس

2.تامين ابزار كنترل در گرامر و كنترل با تجزيه و تحليل دستوري برنامه

3. تجزيه و تحليل مفهومي برنامه كدام گزينه درست است؟ الف.اگر گرامر عبارتهاي ميانوندي مبهم باشد بكارگيري تركيبي از روشهاي 2 و 3 ضروري است.

ب .تنها راه ممكن بكارگيري روش 1 است.

ج .هر يك از روشهاي 1 يا 2 يا تركيبي از آنها كافي است.

د .هر تركيب دوتايي از روش پيشنهادي كفايت مي كند.

83- برنامه اي بزرگ با تعدادي برنامه فرعي را در نظر بگيريد كه هر برنامه فرعي آن چند بار فراخواني شده است. اگر اين برنامه را بدون تعريف و فراخواني هيچ برنامه فرعي بنويسيم ترجمه و اجراي آن با برنامه اول چه فرقي خواهد داشت؟

الف.سرعت اجراي برنامه بشتر و سرعت ترجمه نيز بيشتر مي شود.

ب.سرعت اجراي برنامه بشتر و سرعت ترجمه كمتر مي شود ج .سرعت اجراي برنامه كمتر و سرعت ترجمه بيشتر مي شود د .سرعت اجراي برنامه كمتر و سرعت ترجمه نيز كمتر مي شود.

93- برنامه M را سه بار با روشهاي انتقال پارامتر by reference , by value-result و by name اجرا مي كنيم .خروجي اجراي اول، دوم و سوم در گزينههاي اين سوال با مجموعههاي v, r و n معين شده اند، گزينه صحيح كدام 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;
P (Y[K]); Y[3]:=5;
write ( Y[1]+ Y[2]+
Y[3])
end.
```

r ={2,1,10}, v={2,2,10},

الف r ={2,1,10}, n={3,2,10}.

ب v={2,2,10}, n={2,2,10}.

r ={2,2,10}, v={2,1,10}, n={3,2,11}.

ج r ={2,2,10}, v={2,1,10}, n={3,2,10}.

د

04- در زبان Ada، كه هر if با endif تمام مي شود، مي توان بجاي دو واژه else و if (else if) از يك واژه elsif استفاده كرد. تاثيرهاي ممكن اين كار به شرح زير پيشنهاد شده است ؟

1.باعث كاهش تعداد endifها مي شود.

2.باعث ساده تر شدن ترجمه ساختار if-then-else مي گردد.

3.باعث نابرابري تعداد كل ifها با تعداد كل endifها مي شود.

4. جز كاهش تعداد كل واژههاي بكار رفته در برنامه تاثير ديگري ندارد.

كدام مجموعه از تاثيرها صحيح است؟

الف. 1 ب. 1و3 ج. 2و3 د 4 .

سال 1389

41-قطعه برنامه زير را در نظر بگيريد:

```
Integer Array  M = [1,2,4,8,32];
          Integer X = 1; Integer
                f(Integer a,b){
                  a := a+2;
            return   b:=b*2;
               (100*M[X]+10*b+a;)
                              main
                                {
             Print(f(X.M[X]))
                                }
```

فرض كنيد انديس آرايه از صفر شروع مي شود، مقدار چاپ شده در صورتي كه تمامي فراخوانيها با آدرس (Call-By-Reference) باشند چيست؟

الف. 243 ب .843 ج .483 د 1763.

42- پيوند ايستا (static link) در ركورد فعاليت (activation record) به كجا اشاره مي كند؟ الف. كد رويه صدا زننده (caller) ب.ركورد فعاليت بلاك در برگيرنده ج.ركورد فعاليت متغيرهاي سراسري د. ركورد فعاليت رويه صدا زننده (caller)

43- قطعه كد مقابل را در نظر بگيريد:

```
function f(x,y){return    {
                          x*y;}
                          {
function g(n){ return f(n,n-1);}
                          {
function f(x,y){return x+y;}
                       g(3);
                          }
                          }
```

316 پيوست

}

كدام است؟ dynamic scope و static scope مقدار نتيجه در

حالت : dynamic scope 6 : static scope

الف5 . dynamic 5 : static scope

ب5: scope . 5 : static scope

ج6: dynamic scope . : static scope

د6 6: dynamic scope .

44- در زبانهايي كه گونه / نوع (type) عبارات را بصورت خودكار استنتاج مي كنند، گونه / نوع استنتاج شده براي عبارت زير كدام است؟

گونه / نوع عبارت استنتاج شده در اين قالب نوشته مي شود : **f(g,h,x)=g(h(x))**

گونه ---------- ---------- ---------- ---------- ---------- ----------

گونه ورودي تابع خروجي اول گونه ورودي سوم گونه ورودي دوم گونه ورودي

هستند (type variable) متغير هاي گونه('a,'b,….'z) .

الف'a . 'b 'c 'd ب'a . 'a 'a) ('a 'a) ('a

ج'a . 'b 'a) ('c 'b) د'a . 'b) ('a 'b) ('a 'b)

45- فرض كنيد X متغيري از نوع ليست با مقدار 3(,2, 1) و Y متغيري از نوع ليست با مقدار 6(,5, 4) باشد.

حاصل عبارت (cons (cadr X)Y) كدام است؟

الف. 6(,5, 4 ,2,3, 1) ب. 6(,5, 4 , 1) ج. 6(,5, 4 , 3) د6(

.5, 4 ,2) **64-** شبه كد زير را در نظر بگيريد :

```
int x=2; proc f ( )
{   print (x*x);
}
proc g ( )
{   h (f);
}
proc h (p)
int   {
x=3;
```

```
p ( );
} g (
);
```

نتيجه اجراي برنامه در دو حالتي كه زبان از حوزه ايستا (static scope) و حوزه پويا (dynamic scope) استفاده ميكند به ترتيب از راست به چپ كدام است؟

الف. 9 ، 4 ب9 ، 4 . ج4 ، 4 . د 9 ، 9 .

74- این عبارت حساب لامبدا (Lambda Calculus) را در نظر بگيريد : . . كدام متغير يا متغيرها حداقل يك بار در عبارت به صورت آزاد (free) ظاهر شدهاند .

الف y , a . ب x , a. ج a. د x , y.

84- چنانچه در يك زبان برنامهسازي بتوان با اعلان A : int Tarray [α..β]; یک ماتریس بالا مثلثی از اعداد صحيح تعريف نمود و از روش رديفی (Row Major) برای ذخيره عناصر استفاده شود ،آدرس عنصر A[i,j] از كدام گزينه زير به دست ميآيد؟ (به شکل زیر توجه کنید) فرض کنید Address (A[α,1]) = μ , size(int) = 2

A[α,1]

A[α+1,2] A[α+1,1]

A[β,1] --------------------------------- A[β,β-α+1]

الف. 1−$+2× "−! + μ ب .2×1−$+!× '₊₊₍ₛ∑ + μ ج .2× 1−$

+2×0'₊₊₍ₛ⁄·∑-+, د .2×1+!−$×"−1+,

94- برنامه زير را در يك زبان برنامه نويسي كه در آن حوزه تعريف متغيرهاي تو در تو (Nested Scope) مجاز است در نظر بگيريد:

```
integer I;   Procedure main
      integer A[0:4];
        for I=0 to 4 do
                    A[I]=1;
                  l=I;
             P(l,A[I]);
```

پیوست 318

```
                                        Write(l,A[l]);
                              procedure P (integer A; integer
                                                              B);
          T=B+l;      A=A+l;              integer T;
                                    B=A+T;            I=I+l;
                                        write (A,B);
                                                    end p
end
```

mainبراي برنامه بالا، آدرسهاي زير را در نظر بگيريد:

004 = محل دستورالعملي از سيستم عامل كه برنامه main را فراخواني ميكند .

006 = محل فراخواني رويه P در برنامه main.

009 = آدرس ابتداي حافظه ذخيره دادههاي برنامه

1000 = آدرس ابتداي ركورد فعالسازي برنامه main.

1050 = آدرس ابتداي محل ذخيره آرايه A.

1100 = آدرس ابتداي ركورد فعالسازي رويه P.

با توجه به اطلاعات فوق در رويه P مقدار Static link و Dynamic link به ترتيب از راست به چپ كدام است؟

الف. 1100 و 900 ب .1000 و 600 ج .006 و 1000 د .1000 و 1000

05- كدام گزينه در مورد Dynamic chain pointer (اشارهگر زنجير پويا) ، DCP ، و Static chain pointer (اشارهگر زنجير ايستا) ، SCP ، درست است؟

الف. وقتي از قوانين حوزه پويا (Dynamic scope rules) استفاده ميشود نياز به SCP نيست.

ب .وقتي از قوانين حوزه ايستا (static scope rules) استفاده ميشود نياز به DCP نيست.

ج .براي پيادهسازي قوانين حوزه پويا ميتوان از روش نمايشگر (display) استفاده كرد.

د .براي پيادهسازي قوانين حوزه ايستا ميتوان از جدول محيط ارجاع مركزي (centeral referencing

. استفاده كردenvironment)

كليد مجموعه سوالات كنكوري

ردیف	الف	ب	ج	د
1				*
2	*			
3			*	
4				*
5				*
6			*	
7		*		
8	*			
9			*	
10		*		
11				*
12	*			
13				*
14		*		
15		*		
16			*	
17			*	
18	*			
19				*
20	*			
21		*		
22		*		
23			*	

ردیف	الف	ب	ج	د
24	*			
25	*			
26			*	
27		*		
28	*			
29		*		
30			*	
31	*			
32			*	
33		*		
34				*
35	*			
36			*	
37			*	
38		*		
39				*
40	*			
41		*		
42		*		
43	*			
44			*	
45				*
46		*		

320 پیوست

			*	**47**
	*			**48**
			*	**49**
	*			**50**

مراجع:

1- **Programming Languages:Design and Implementation,Pratt(2001)**

2- **Concepts of programming languages,W.sebesta(2008)**

3- **برگرفته از گردآوری مهندس مصطفی قبائی**