فصل سوم
داده های اولیه

by: Mahdi Sadeghizade
Email: Mahdi_sadeghe@yahoo.com
Site: www.Sadeghizadeh.ir

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

تعاریف اولیه :

Data Object (D.O) :

به گروهی از یک یا چند قسمت از اطلاعات در کامپیوتر مجازی گفته می‌شود. Data Object ها به دو دسته تقسیم می‌شوند (از نظر تعریف‌کننده) :

- Programmer Defined : مثل متغیرها مقادیر ثابت‌ها، آرایه‌ها و فایل‌ها
- System Defined : مانند Stack ها در زمان اجرای برنامه، رکوردهای فعالیت زیربرنامه‌ها و بافر فایل‌ها و لیست فضاهای خالی .

یک Data Object توسط مجموعه‌ای از صفت‌ها مشخص می‌شود (مثلا متغیرها دارای نوع، مقدار و ... هستند)

یک Data Object شامل محلی برای Data Value است.

یک Data Object می‌تواند pointer, character, single number و ... باشد.

یک Data Object توسط الگویی از بیت‌ها مشخص می‌شود (جنبه پیاده‌سازی : مقدار متغیرها)

انواع Data Object :

- **E.D.T :** یک Data Object از نوع اولیه است اگر تنها یک محل برای Data Value داشته باشد.
- **S.D.T :** یک Data Object ساخت‌یافته است اگر شامل جمعی از Data Object های اولیه باشد.(یعنی چندین مقدار را شامل شود.)

با توجه به تعاریف فوق انواع مختلف انقیاد در یک Data Object به صورت زیر دسته‌بندی می‌شود :

- انقیاد یک Data Object به یک یا چند ارزش : این نوع انقیاد به هنگام دستورات انتساب در زمان اجرا انجام می‌شود.
- انقیاد یک Data Object به یک یا چند نام : این‌گونه انقیادها به هنگام تعریف زیربرنامه استفاده می‌شود. مثلا در هنگام ورود متغیرهای با ارجاع به یک زیربرنامه، دو متغیر همزمان به شی داده‌ای اشاره می‌کنند که یکی متغیر اصلی است و دیگری متغیر تعریف شده در زیربرنامه.
- انقیاد یک Data Object به یک یا چند Data Object دیگر : این‌گونه انقیادها به هنگام استفاده از متغیرهای اشاره‌گر استفاده می‌شود.(در زمان اجرا)

- انقیاد یک Data Object به یک محل خاصی از حافظه : این نوع از انقیاد از دید برنامه‌نویس پنهان است و توسط روتین‌های مدیریت حافظه در کامپیوتر مجازی انجام می‌شود.
- انقیاد یک Data Object به یک نوع : با مجموعه مقادیر داده‌ای که شیء داده می‌تواند اختیار کند همراه می‌شود که بسته به اینکه زبان EBT باشد یا LBT زمان آن ممکن است در زمان ترجمه باشد یا در زمان اجرا باشد.

یک**Variable** یک Data Objectی است که توسط برنامه‌نویس تعریف شده و به صورت صریح مورد استفاده قرار می‌گیرد.
یک **Constant**، Data Objectی است که نام آن به ارزش خاصی مقید می‌شود.
یک **Literal Constant** ثابتی است که نام آن مطابق ارزش ارائه شده می‌باشد.

مثال:

```
Const Max =30;
Var N: integer;
N: =24;
N: =Max+N;
```

در این مثال N یک متغر ساده است یعنی تنها یک مقدار می تواند بگیرد. Max یک ثابت تعریف شده توسط برنامه‌نویس است و اعداد 30 ، 24 نیز Literal هستند .

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

مثال:

```
If (Max<2) then
…
else
…
```

در این حالت مترجم از قبل مقادیر داده‌ها را برای ثابت‌هایMax و2 دارد و می‌تواند محاسبه نماید که شرط فوق درست است یا خیر. به طور کلی از هر کد نوشته شده داخل دستور If در یکی از قسمت‌های آن خودداری کند .

هرData Object ، Time Life مخصوص به خود دارد . منظور از Time Life از لحظه بوجود آمدن آن تا لحظه از بین رفتن آن است . Time Life متغیرهای یک برنامه با زمان اجرای برنامه مشخص می‌شود. به هر حال Time Life داده‌ها اغلب فراتر از یک اجرای منفرد است. در این حالت گفته می‌شود که داده‌ها مانا یا پایدار هستند و بین اجرای برنامه نیز وجود دارند .

Data Type (D.T) :

یک Data Type کلاسی از Data Object ها به همراه مجموعه‌ای از عملیات برای تولید و دستکاری (Manipulation) می‌باشد و معمولا برای دو سطح مختلف بررسی می‌شود :

- تعریف کردن (Specification): تعریف خصوصیات و مشخصات آن
- پیاده‌سازی کردن(Implementation): پیاده‌سازی آن Data Type و عملیات روی آن

در سطح تعریف (Specification) چند عنصر اساسی می تواند مطرح شود :

- **صفت‌ها(Attribute)**: ویژگی که این Type را از بقیه مشخص می‌کند یا Type خاصی را معین می نماید. مثلا در مورد آرایه‌ها صفت‌ها شامل ابعاد آرایه، ایندکس‌های آرایه و نوع عناصر آرایه می-باشد.
- **مقادیر (Value)**: ارزش‌های مختلفی که یک Data Object می‌تواند بگیرد را می‌گویند. مثال آن مجموعه‌ای از ارزش‌های معتبر بر حسب Type آرایه می باشد.(مثلا Integer بازه مقادیر آن بین 32768 - تا 32767+ می‌باشد.)
- **عملیات(Operation)**: انواع دستکاری های ممکن را برای Type خاصی تعریف می‌نماید. مثلا در مورد آرایه‌ها شامل نام آرایه و شماره عنصر آن جهت دستکاری کردن اطلاعات می‌باشد .

عناصر اساسی جهت پیاده‌سازی (Implementation)عبارتند از:

- روشی که عملیات تعریف شده برای Data Object جهت الگوریتم‌ها و زیربرنامه‌ها به منظور دستکاری استفاده می‌شود یا به عبارت دیگر "نحوه پیاده‌سازی عملیات"
- چگونگی ذخیره اطلاعات و نمایش حافظه(Storage Representation)

پیچیده شدن عملیات(ایجاد ابهام در عملیات):

چهار فاکتور اساسی عملیات Data Object ها را پیچیده می‌سازد:

1. **عملیاتی که به ازای ورودی‌های مشخصی تعریف شده نیستند**. مانند عملیات تقسیم بر صفر.

2. **ورودی‌های ضمنی(implicit arguments)** : ورودی‌هایی که به صورت صریح تعریف نشده‌اند مثل استفاده از متغیر global و یا registerها.

3. **اثرات جانبی(side effect-implicit result)** : یا نتایج ضمنی مثل دستورات انتساب یا برنامه‌ای که پارامتر ورودی خود را تغییر می‌دهند. یک عمل ممکن است یک نتیجه صریح را برگرداند مانند حاصل جمع برگردانده شده به عنوان نتیجه جمع. اما ممکن است مقادیر ذخیره شده در اشیاء داده دیگر را نیز تغییر دهد که می‌تواند توسط برنامه‌نویس یا سیستم تعریف شده باشد. مثلا دستور انتساب در زبان C اثر اصلی‌اش عمل انتساب است و اثر جانبی آن برگرداندن مقدار انتساب داده شده به عنوان نتیجه کل عبارت است. مانند عبارت if(A=B) که ممکن است خطای برنامه‌نویس باشد و یا درست باشد و برنامه‌نویس بخواهد از اثرات جانبی دستور انتساب استفاده کند.

4. **خود اصلاحی(self modification)** : یا حساسیت نسبت به قبل. مثلا در صورتی که یک زیربرنامه اعداد تصادفی تولید کند برای دفعات بعدی اجرا این زیربرنامه از حالت‌های قبلی‌اش اثر می‌پذیرد. مثلا متغیرstatic در C مثالی از حساسیت نسبت به قبل است.(آخرین مقدارش را بعد از اتمام زیربرنامه حفظ می‌کند.)

☑ خود اصلاحی از طریق تغییر در کد معمولا متداول نیست اما در زبان‌هایی مثل Lisp امکان پذیر است.

تقسیم بندی عملیات :

Operationها به دو دسته عمده تقسیم می‌شوند:

1. **اعمال اولیه (Primitive Operation)** : عملیاتی هستند که به هنگام تعریف زبان تعیین می گردد . مثل +، -، * / ...
2. **عملیات تعریف شده توسط برنامه‌نویس (Programmer defined Operation):** که این اعمال توسط برنامه‌نویس و به هنگام برنامه‌نویسی جهت کاربردی خاص مورد استفاده قرار می‌گیرند. مثل الحاق رشته‌ها و مقایسه رشته‌ها

از دید زبان‌های برنامه‌سازی اعمال اولیه به دو دسته تقسیم می‌شوند:

1. **Binary Operation** : که دو پارامتر را گرفته و یک نتیجه را برمی‌گرداند.
2. **Unary Operation** : که یک پارامتر ورودی را گرفته و یک نتیجه را برمی‌گرداند.

ازجهت ریاضی هر Operation به صورت زیر تعریف می‌شود:

Operation: Argument-type1*Argument-type2* … → Result-type1* Result-type2* …

برای دامنه هر Operation باید تعداد، ترتیب و نوع آرگومان‌ها مشخص شود:

*: integer*integer → integer
=: integer*integer → Boolean
sqrt: real → real

پیاده‌سازی E.D.T :

دو عنصر اساسی برای پیاده‌سازی عبارتند از :

- **چگونگی ذخیره و بازیابی حافظه(Storage Representation):** برای ذخیره اطلاعات معمولا از سخت‌افزار کمک گرفته می‌شود. به عنوان مثال integer با سخت‌افزار پشتیبانی می‌شود که مزیت آنها کارایی و سرعت بالا می‌باشد، به دلیل اینکه صفات داده‌ها در آنها توسط کامپایلر تعیین می‌گردد مثل زبانC .

در صورت عدم استفاده از سخت‌افزار از تکنیک‌هایsoftware simulation استفاده می‌شود مثلا شبیه‌سازی اعداد 24 رقمی، در صورتی که سخت‌افزار حداکثر اعداد12 رقمی را پشتیبانی می‌کند. مزیت این روش انعطاف‌پذیری بالای آنست. اما کارایی پایین دارد. زیرا صفات شیء داده‌ای به عنوان یک توصیفگر در زمان اجرا بررسی می‌شود.

صفت‌های Data Object به دو روش در جهت اجرای برنامه استفاده می‌شود:

1. صفت‌های Data Object در زمان کامپایل استفاده شود مانندPascal .
2. صفت‌های Data Object در زمان اجرا به عنوان یک توصیفگر (Descriptor) جهت دستکاری کردن و بکارگرفتن عملیات استفاده شود مثل زبانsnobol4 که type متغیر در زمان اجرا تغییر می-کند(Lisp و prolog نیز چنین هستند)

- **چگونگی پیاده‌سازی عملیات** : در این مورد عملیات به چند روش به اجرا درمی‌آیند :
  1. مستقیما از طریق سخت‌افزار انجام شود مانند جمع دو عدد صحیح.
  2. به عنوان یک function یا procedureکه از طریق نرم‌افزار پیاده‌سازی می‌شود، ارائه می‌شود مثل sqrtو کلا زیربرنامه‌های کتابخانه‌ای.
  3. به عنوان دنباله‌ای از دستورات که از طریق سخت‌افزار پیاده‌سازی می‌شمود و به صورت کد در برنامه برنامه‌نویس قرار داده می‌شود.(تعریف یک عمل به صورت ماکرو)

مثال:

ABS(x) = if x<0 then –x else x;
a) fetch value of x from memory
b) if x>0 skip next instruction
c) set x=-x
d) store x in memory

این روش نیز نوعی پیاده‌سازی نرم‌افزاری است اما به جای استفاده از زیربرنامه خیلی کوچک اعمال موجود در زیربرنامه در نقطه‌ای که زیربرنامه باید فراخوانی شود، در برنامه کپی می‌شود.

دستورات اعلان(Declaration):

مجموعه ای از دستورات زبان برنامه سازی است که به نحوی ارتباط بین برنامه نویس و برنامه مترجم را جهت تولید کد که لازم برای اجرا فراهم می کند و اطلاعاتی در مورد type و تعداد و سایر موارد فراهم می سازد. که دو دسته می باشند:

1. **صریح (Explicit)** : توسط برنامه نویس باید تعیین شود. مثل C,Pascal .
2. **ضمنی (Implicit)** : خود برنامه مترجم یا کامپیوتر پیش فرض هایی در مورد صفت ها داردمثل زبان Fortran که در آن متغیرهای i,j تا n از نوع صحیح در نظر گرفته می شوند. در زبان perl تنها انتساب مقدار به متغیر باعث اعلان آن می شود.

$abc=7 ; #$abc is now integer

☑ اعلان های ضمنی گاهی اوقات اجرای برنامه را به error می کشاند.

اطلاعاتی که توسط دستور اعلان تعیین می شود :

- نام مربوط به D.O
- type مربوط به D.O
- مقدار اولیه مربوط به D.O
- attribute مربوط به D.O .مثل ابعاد آرایه ، فیلدهای رکورد و ... .

☑ گاهی اوقات در اعلان جزئیات پیاده سازی مانند تقید به محل خاصی از حافظه یا تقید به نمایش حافظه ای خاص نیز مشخص می شود. در مورد اول مثل کلاس حافظه ی ثباتی در C Register int x; . و برای حالت دوم در Cobol متغیرهای صحیح تعریف شده به صورت Computational که نمایش دودویی برای آنها در نظر گرفته می شود به جای نمایش کاراکتری به دلیل بالا بردن سرعت اجرایی و امکان استفاده از اعمال محاسباتی کارآمدتر.

اعلان های اعمال :

برای اعلان عملیات باید تعداد ، ترتیب و نوع پارامترها و نتایج مشخص شوند. مثل :

در پاسکال Function SUB (X:integer;Y:real):Real;

تعریف ریاضی SUB: integer* Real → Real

اهداف دستورات اعلان :

1. *کنترل نوع (type cheking):*

مهمترین هدف اعلان ها از دیدگاه برنامه نویس این است که بررسی نوع را به صورت ایستا به جای پویا انجام دهد .

2. *انتخاب چگونگی ذخیره و بازیابی اطلاعات* :

دستورات اعلان اطلاعات لازم جهت چگونگی ذخیره اطلاعات را برای زبان برنامه سازی فراهم می سازد. که این امر کاهش کلی ای را در میزان نیاز به حافظه و زمان اجرای برنامه به همراه دارد.

3. *مدیریت حافظه (Storage Management)* :

دستورات اعلان اطلاعات مهمی را جهت مدیریت حافظه فراهم می کنند برای مثال در زبان C اشیاء داده ای که در ابتدای زیربرنامه ها اعلان می شوند همگی دارای مدت زندگی مشابه ای هستند (معادل با مدت اجرای زیربرنامه ها) و بنابراین حافظه می تواند به صورت بلوکی در ورود به زیربرنامه اختصاص یابد و با خروج از آن آزاد گردد. اشیاء داده ای دیگر در C به صورت پویا توسط Malloc ایجاد می گردند که با توجه به اینکه مدت های زندگی این اشیاء داده اعلان نشده است باید به هر یک از آنها حافظه مجزایی تخصیص داده شود.

عملیات چند ریختی – چند شکلی (Generic Operation) :

در یک زبان برنامه سازی سنبول خاصی مثل + می تواند جهت 2 عدد integer ، Real ، الحاق دو رشته ، جمع دو عدد مختلط و یا 2 مجموعه به کار گرفته شود که بسته به نوع عملوندهای آن زیربرنامه های خاصی برای اجرای آن باید فراهم شود (که اعلان های نوع اطلاعات مفیدی را در این جهت برای ما فراهم می کند.)

مثال :

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

- ☑ معمولاً به چنین عملگری که چندین عملیات را تحت پوشش قرار می دهد عملگر پربار شده می گویند (Over Lapping) مثل عملگر جمع.
- ☑ در زبان ML این مفهوم را با استفاده از چند ریختی کامل گسترش می دهد یعنی در این زبان یک نام تابع با پیاده سازی های متفاوت ارائه می شود که با توجه به انواع ، تعداد و ترتیب ورودی ها و نتایج یکی از پیاده سازی ها انجام می شود.

تعریف کلی چند ریختی:

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

- ☑ در زبان Ada به برنامه نویس این امکان داده می شود که نام های زیربرنامه پربار شده را تعریف نماید و به نمادهای عملگرهای موجود معانی اضافی دیگری را ضمیمه کند.

کنترل نوع و تبدیل نوع: Type Checking & Type Conversion

منظور از کنترل نوع آن این است که هر عملیاتی که در برنامه انجام می شود تعداد، نوع و ترتیب آرگومانهای آن درست باشد .

نکته مهم در کنترل نوع این است که عملیات Type Checking را در سطح سخت افزار نداریم زیرا تمامی اطلاعات در سطح سخت افزار به صورت رشته ی بیتی می باشند که این رشته ی بیتی می توانند بیانگر هر نوع داده ای باشند. بنابراین عملیات Type Checking باید از طریق مکانیزم های برنامه نویسی انجام شود که این کار باید قبل از اجرای هر عمل قرار گیرد .

انواع Type Checking :

1. Dynamic Type Checking (D.T.C) : عمل چک کردن Type ها در زمان اجرا انجام می شود بنابراین باید یک Type به عنوان Scriptor (توصیف گر) نوع آن D.O به همراه کد برنامه ارائه شود.
2. Static Type Checking (S.T.C) : عمل بررسی نوع در زمان ترجمه یا کامپایل انجام می شود.

نقاط قوت D.T.C :

1. انعطاف پذیری بالا می رود.
2. اعلان نوع برای Type می تواند وجود نداشته باشد.
3. عمل تبدیل نوع را می توان در زمان اجراء انجام داد.

نقاط ضعف D.T.C :

1. سرعت اجرای برنامه به دلیل استفاده از مکانیزم نرم افزاری پایین می آید.
2. فضای اضافی جهت ذخیره نوع، D.O در زمان اجراء.
3. Debug کردن برنامه مشکل است (به علت تغییر نوع داده ها در زمان اجراء)

نقاط قوت S.T.C :

1. سرعت اجرای آن بالا است.
2. نیازی به فضای اضافی در زمان اجرا نمی باشد.
3. تبدیل نوع می تواند به صورت قطعی و در زمان کامپایل انجام شود.
4. تمام مسیرهای اجرایی برنامه از جهت Type چک می شود.

نقاط ضعف S.T.C :

1. انعطاف پذیری کم می باشد.

2. جهت مجزا کامپایل کردن برنامه ممکن است مشکلاتی به وجود آید.(در این نوع زبانها نیازمند به اعلان تمامی اشیاء داده ای داریم همچنین کنترل عملیات چندریختی نیز تا حدودی در آنها مشکل است.)

- ☑ اگر در برنامه امکان آشکارسازی تمامی خطاهای نوع به صورت ایستا وجود داشته باشد گفته می شود که زبان دارای بررسی نوع قوی است.
- ☑ ایمن بودن نوع تابعی از $F:S \rightarrow R$ از نظر نوع ایمن می گوییم اگر اجرای F نتواند مقداری خارج از R را تولید نماید. واضح است که اگر هر عمل از نظر نوع ایمن باشد زبان نیز دارای بررسی نوع قوی است.
- ☑ زبان ML روش جالبی برای انواع داده ها دارد. اگر تفسیر فاقد ابهام باشد در این حالت نیازی به اعلان های نوع نیست بنابراین پیاده سازی زبان در صورت نیافتن اطلاعات نوع آنها را با استفاده از انواع اعلان های دیگر بدست می آورد.

مثال(برای زبان ML):

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

از روی خروجی می فهمد ورودی ها intاست

خط آخر دارای خطا می باشد چون هیچ کدام نوعشان مشخص نیست.

معمولاً 2 فلسفه متضاد در تبدیل نوع وجود دارد (Type Mismatch):

1. در پاسکال و Ada تقریباً هیچ گونه تبدیل نوعی انجام نمی شود همیشه خطا می دهد.
2. در زبان C تبدیل انواع ها قانونی هستند مگر اینکه امکان هیچ گونه تبدیلی وجود نداشته باشد که با هم خطا می گیرد.

- ☑ زبان PL1 به یافتن حداقل خطاهای تبدیل نوع و کلاً خطاهای برنامه نویسی مشهور است یعنی سعی می کند در اکثر موارد تبدیل نوع را انجام دهد و خطا نگیرد.
- ☑ در زبان PL1 عبارت 9+10/3 غیر مجاز است زیرا با توجه به بالاتر بودن اولویت تقسیم نسبت به جمع ابتدا تقسیم شده و حاصل 3.33... با حداکثر ارقام اعشاری مجاز خود بدست می آید و در هنگام جمع این عدد با 9 چون یک رقم اضافی می آورد، پس به خطای Owerflow منجر می شود.

انواع عملیات تبدیل نوع در زبانها:

1. **صریح**

در این حالت برای عمل تبدیل نوع یکسری زیربرنامه های از پیش طراحی شده وجود دارد که در زبان برنامه سازی تعریف شده است و به برنامه نویس امکان تبدیل نوع صریح را می دهد. مثل Round ,val,strدر پاسکال.

2. ضمنی:

در این حالت عمل تبدیل نوع به صورت اتوماتیک توسط برنامه مترجم انجام می شود مثلاً اگر یک متغیر اعشاری را به یک متغیر صحیح نسبت دهیم تبدیل نوع آن ضمنی خواهد بود.

تخصیص ارزش و دادن مقدار اولیه :

دستورات انتساب ، مجموعه ای از دستورات زبان برنامه سازی هستند که می توانند یک D.O را به ارزش خاصی مقید کنند این دستورات به دو شکل زیر می تواند وجود داشته باشد :

1. Assignment:Type1 * Type2 → Void در پاسکال
2. Assignment:Type1 * Type2 → Type3 Lisp,APL,Cدر

در مورد اول دستور انتساب هیچ نتیجه ای را برنمی گرداند اما در حالت دوم بعد از اینکه یک کپی ا زمقدار موجود از Type2 در Type1 قرار گرفت D.O مربوط به Type3 ایجاد می شود که شامل مقدار Type2 است.

مثال: A=B=3

مثال: Y← A+(X← B+2*C) : ≈ { X←B+2*C ; Y←A+B+2*C }

نمونه هایی از دستورات انتساب در چند زبان برنامه سازی:

پاسکال یا Ada : (A:=B)

فرترن،PL1,C,ML,Prolog,Snobol : (A=B)

کوبول : mov B to A

APL : B <-- A

Lisp : SETQ A B

در Prolog داریم A=2+3 که می تواند به دو صورت برداشت شود:

2+3 را ارزش یابی و حاصل را در A قرار بده. =

الگوی 2+3 را به A نسبت بده. is

بنابراین در Prolog برای رفع این ابهام از دو عملگر جدا استفاده می شود یعنی برای انتساب حاصل عملیات (5) is استفاده می شود و برای انتساب الگو از عملگر (=) استفاده می شود.
نکته: در زبان Snobol همه متغیرها اشاره گر می باشند بنابراین اگر A,B دارای مقادیر 12,14 باشند انتساب A=B به صورت زیر است.

دو نوع مقداردهی اولیه در زبانها وجود دارد :

1. صریح : که در اینحالت برنامه نویس باید دستورات لازم برای دادن مقدار اولیه به متغیرها را در برنامه وارد کند.
2. ضمنی : که در این حالت خود کامپایلر مقدار اولیه متغیر ها را تعیین می کند که ایم مقدار اولیه یا صفر یا Null می باشد .

انواع EDT

* انواع اطلاعات عددی :

زبانهای برنامه سازی انواع مختلف اطلاعات عددی را پشتیبانی می کنند فقط در بخش پیاده سازی ممکن است با هم اندکی اختلاف داشته باشند .

- **اعداد صحیح Integer**

نکات مهم در مورد اعداد صحیح شامل موارد زیر است :

- هیچ صفتی غیر از نوع برای DO لازم نیست .
- مجموعه ارزشهای ممکن ، یک مجموعه ترتیبی است . یعنی تقدم و تاخر .
- یک حداقل و حد اکثر ارزش وجود دارد .

عملیات روی نوع داده صحیح شامل موارد زیر است :

1. عملیات محاسباتی

Binery-Op : integer*integer → integer ► MOD,+,-,*,DIV,/
Unery-Op : integer → integer ► ABS,++,--,+, -

2. عملیات رابطه ای :

Rel-Op : integer * integer →boolean ► =,<>,>=,<=,<,>

3. عملیات انتساب :

Assign:integer*integer → void
Assign : integer * integer → integer

4. عملیات بیتی :

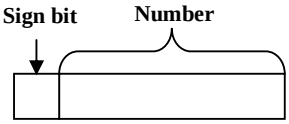
Bit-Op:integer*integer→integer ► &,|,~,^,>>,<<

از جهت پیاده سازی معمولا همه عملیات فوق توسط سخت افزار پشتیبانی می شود .

از جهت چگونگی ذخیره اطلاعات و نمایش حافظه سه فرمت وجود دارد :

1. روش بدون توصیفگر (No-Descriptor)

این روش مستقیما توسط سخت افزار حمایت می شود و برای زبانهایی استفاده می شود که نیاز به کارایی بالایی دارند . مانند C,Pascal,Fortran,...

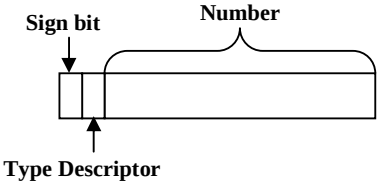


2. روش دارای توصیفگر جدا

مانند زبان Lisp و زبانهایی که حالت مفسری داشته و نیاز به انعطاف بالایی دارند . این روش فضای زیادی اشغال می کند .

3. روش دارای توصیفگر پیوسته

این روش را زبان خاصی استفاده نکرده ، چون عملیات خاصی (شیفت دادن) برای استخراج ارزش ذخیره شده نیاز دارد و بنابراین روش کارایی نیست .



منظور از توصیفگر فقط توصیفگر نوع می باشد .

- **اعداد با محدوده معین (SubRange)**

یک دامنه محدود از اعداد صحیح را در بر می گیرد .

مثال :

```
Pascal    A:1 .. 100;
Ada       A:integer range 1 .. 100;
```

اگر نوع داده بحشی از رده بزرگتر باشد گفته می شود که زیر نوعی از رده بزرگتر است و رده بزرگتر یک SuperType از این نوع داده است .

زیر نوع تمامی اعمال نوع مافوق خود را داراست .

این اعداد از جهت پیاده سازی دو نوع هدف را تعقیب می کنند :

1. نیاز به فضای کمتر
2. انجام عمل چک کردن نوع ها به روشی بهتر

مثال :

```
i: integer ;
month:1 .. 12 ;
□
month:=0;
i:=0;
month:=i;
```

از جهت پیاده سازی ما می توانیم از همان نوع داده integer و تعدادی محدودیت استفاده کنیم .

♦ اعداد اعشاری

اعداد integer می تواند دامنه محدودی را قبول کند . اعداد اعشاری با توجه به این که از دو قسمت پایه و توان تشکیل شده اند ، دامنه متغیری را پوشش می دهند ولی در عین حال دقت خیلی بالایی نخواهند داشت (زیرا ممکن است شرط تساوی دو عدد اعشاری برقرار نشود.) این نوع توسط سخت افزار پشتیبانی می شود .

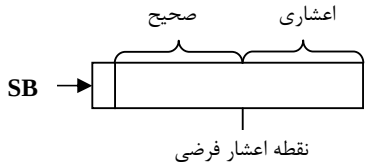
عملیات روی آن مشابه عملیات روی اعداد integer می باشد فقط برخی از عملیات ممکن است توسط نرم افزار شبیه سازی شود مانند عملیات به توان رساندن و ...

از جهت پیاده سازی روش های زیر را دارد :

1. **FixedPoint**

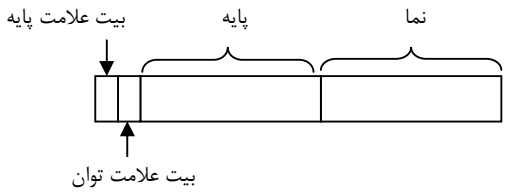
مثلا در زبان cobol اعلان داده اعشاری ممیز ثابت با عبارت picture نشان داده می شود .

Picture 999 V 99



2. **Floating Point**

مانند نماد علمی



☑ در زبان Ada می توان تعداد ارقام دقت عدد اعشاری را توسط برنامه نویس مشخص نمود .

نمونه ای از اعدد اعشاری ممیز ثابت در زبان PL1 به صورت زیر است :

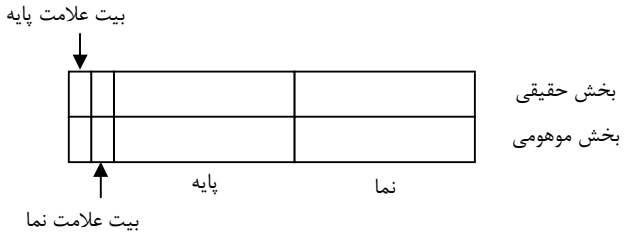
```
DECLARE x Fixed DECIMAL (1,3);
```

تعداد ارقام صحیح تعداد ارقام اعشار

اطلاعات عددی دیگر

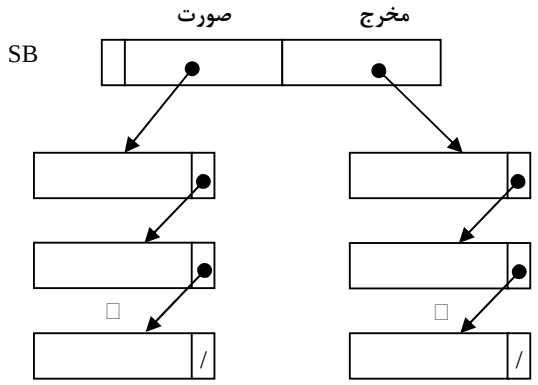
▯ اعداد موهومی (مختلط)

یک جفت از اعداد است که شامل بخش حقیقی و موهومی است و معمولا به صورت بلوکی از دو محل حافظه نمایش داده می شود که حاوی دو مقدار حقیقی است و توسط نرم افزار شبیه سازی می شود .



▯ اعداد گویا

خارج قسمت دو عبارت صحیح می باشد یا به عبارت دیگر اعداد کسری با مقادیر صورت و مخرج بزرگ .
هدف از وجود این نوع داده ، پرهیز از مسائل مربوط به گرد کردن در نمایش اعداد حقیقی به صورت ممیز ثابت و شناور است . این اعداد به صورت جفتی از اعداد صحیح به طول نامحدود (به صورت پیوندی)نمایش داده می شوند . مانند شکل فوق این روش نیز باید توسط نرم افزار پیاده سازی شود .



❋ اطلاعات شمارشی

این نوع داده مانند یک متغیر دارای تعداد حالات محدود می باشد .

مثال : Student – class – (fresh,sophm,junior,senior)

عملیات شامل عملیات

- مقایسه ای < , =< , > , >= , = , <>
- انتساب
- Succesor , Predsessor می باشد .

از جهت پیاده سازی :

یک متغیر integer با ارزش های 0 ، 1 ، 2 ... استفاده می شود .برای انجام عملیات گفته شده پشتیبانی سخت افزاری وجود دارد به جز در مورد آخر یعنی افزایش یا کاهش یک واحد از نرم افزار کمک گرفته می شود .

* متغیرهای منطقی

متغیرهای منطقی دارای دو حالت true و false می باشد.

در زبان های Pascal و Ada نوع داده بولین فقط به صورت نوع داده شمارشی تعریف می شود و به صورت زیر می باشد :

```
Type Boolean=(false,true);
```

در بعضی از زبانهای برنامه سازی اعمال زیر برای کار با Boolean در نظر گرفته شده اند :

Logic Operations(عبارت منطقی)

AND : Boolean *Boolean → Boolean
OR : Boolean *Boolean → Boolean
NOT : Boolean → Boolean
XOR : Boolean *Boolean → Boolean
NAND: Boolean *Boolean → Boolean
NOR: Boolean *Boolean → Boolean

از جهت پیاده سازی :

از کوچکترین واحد حافظه به یکی از دو روش ریز می توان استفاده کرد :

تنها از SignBit استفاده شده و بقیه اطلاعات نادیده گرفته شود . مانند Pascal

کل حافظه (بایت) اگر صفر باشد false در نظر گرفته می شود و عدد غیر صفر نیز نماینده true می باشد . مانند C

* اطلاعات غیر عددی

- **کاراکتری** :

یکی دیگر از اطلاعات اولیه اطلاعات غیر عددی یا کاراکتری می باشد.

در مورد نوع کاراکتری نکات زیر قابل توجه هستند :

- اغلب اطلاعات ورودی و خروجی به صورت کاراکتر است که در این حالت تبدیلات بعد و یا قبل از آن انجام می شود .
- مجموعه ای از کاراکترها در استانداردهای خاصی قرار می گیرند و پشتیبانی می شوند . مثل ASCII و EBCDIC .

عملیات روی نوع داده کاراکتری شامل :

- عملیات مقایسه ای یا رابطه ای
- عملیات انتساب
- عملیات جستجو برای یک کاراکتر خاص
- تست رقم یا حرف بودن کاراکتر و ...

از جهت پیاده سازی :

پشتیبانی سخت افزار و سیستم عامل وجود دارد .

رشته : دنباله ای از کاراکترها که به عنوان یک واحد پرازش می شوند یک رشته کاراکتری نامیده می شود .

☑ در زبانهای ML و Prolog نوع داده رشته ای مستقیما ارائه شده است ولی در زبانهای C و Pascal و Ada رشته کاراکتری را به عنوان آرایه خطی از کاراکترها در نظر می گیرند .