نوع داده

انواع داده :

۱- اولیه (صحیح، ممیز شناور، بولی، کاراکتری، رشته‌های کاراکتری، شمارشی، زیر بازه و اشاره‌گر)
۲- ساختمان داده (ساختاری) (مانند آرایه، رکورد، لیست، مجموعه و پشته)
۳- انتزاعی (مانند کلاس‌ها در C++)

بعضی از انواع داده اولیه به صورت سخت افزاری پیاده سازی می‌شوند. مثل نوع داده صحیح و بعضی دیگر، توسط نرم‌افزار باید پشتیبانی شوند، مثل کاراکترها.

انواع داده اولیه را می‌توان به دو دسته تقسیم کرد:

۱- اسکالر: برای اشیای داده خود، فقط یک صفت دارند.

۲- مرکب: برای اشیای داده خود، صفات بیشتری دارند.

شیء داده‌ای از نوع صحیح، صفتی غیر از نوع ندارد (اسکالر)، اما نوع داده بردار، علاوه بر مقادیر، دارای صفاتی مثل نوع و عناصر و طول بردار هستند (مرکب)

اشیای داده اسکالر از معماری سخت افزار کامپیوتر پیروی می‌کنند و اشیای داده مرکب توسط کامپایلر تولید می‌شوند و توسط سخت افزار پیاده سازی نمی‌شوند.

توصیفگر نوع داده

توصیفگر (descriptor) : مجموعه‌ای از صفات یک شیء داده که آن را توصیف می‌کند.
توصیفگر نوع داده آرایه شامل موارد زیر است: نام آرایه، حد پایین و بالا، نوع عناصر، اندازه هر عنصر.

توصیفگر زمان ترجمه : توصیفگری که د رزمان ترجمه وجود داشته باشد و برای اهدافی مثل کنترل نوع به کار رود.

توصیفگر زمان اجرا : اگر صفتی از شیء داده در زمان اجرا تغییر کند، توصیفگر شیء داده باید به عنوان بخشی از نمایش حافظه شیء داده ذخیره شود، این توصیفگر را زمان اجرا می‌گویند.

امضای (signature) عملیات

امضای عملیات مشخص می‌کند که عملیات چند پارامتر دارد، نوع پارامترها کدام است و نوع نتیجه چیست.

به عنوان مثال، عملیات جذر گیری (sqrt)، یک شیء داده حقیقی را به عنوان پارامتر پذیرفته، شیء داده حقیقی دیگری را به عنوان نتیجه برمی‌گرداند:

sqrt : real → real

مثال: امضای عملیات جمع صحیح دو شیء داده:

+ : integer * integer → integer

نوع داده صحیح

عملیاتی که بر روی اشیای داده صحیح انجام می‌شود، عبارتند از:

۱- محاسباتی ۲- رابطه‌ای ۳- انتساب ۴- بیتی

نوع داده صحیح با نمایش حافظه سخت افزاری پیاده سازی می‌شود.

در زبان‌هایی مانند C , C++ , C# , Java که می‌توانند کنترل نوع ایستا را انجام دهند، نمایش حافظه فاقد توصیفگر زمان اجرا است و مقدار را ذخیره می‌کند.

در زبان‌ی مثل lisp که نمی‌توانند کنترل نوع ایستا را انجام دهند، از توصیفگر زمان اجرا برای نمایش نوع صحیح استفاده می‌شود.

در این روش، حافظه مورد نیاز برای شیء داده، دو برابر می‌شود.

نوع ممیز شناور

برای مدل سازی اعداد حقیقی به کار می‌رود. مقادیری که توسط این نوع نمایش داده می‌شود، شامل دقت و بازه است.

دقت، تعداد ارقام قسمت اعشاری است که به بایت سنجیده می‌شود.

بازه، ترکیبی از بازه قسمت کسری و بازه قسمت توان است.

نوع ممیز شناور نمی‌تواند اعداد حقیقی را به طور دقیق نمایش دهد.

عملیات بولی تا حدی در مورد اعداد ممیز شناور، محدود است.

با توجه به مسئله گرد کردن، مقایسه دو مقدار حقیقی کمتر رخ می‌دهد و به همین دلیل، حلقه‌ای که در شرط آن دو مقدار اعشاری با هم مقایسه می‌شوند، ممکن است هیچ وقت خاتمه پیدا نکند.

نمایش حافظه انواع داده حقیقی

این نمایش به سخت افزار بستگی دارد و محل حافظه به دو بخش کسری (مانتیس) و توان تقسیم می‌شود و در آن هر عدد N به صورت $m \times 2^k$ نوشته می‌شود که m عددی بین صفر و یک و k یک مقدار صحیح است.

اعداد ممیز شناور به صورت دقت معمولی (۳۲ بیت) و دقت مضاعف (۶۴ بیت) قابل استفاده‌اند.

این که از چه نمایش حافظه‌ای استفاده می‌شود، با تعریف نوع float یا double توسط برنامه نویس تعیین می‌شود.

نوع ممیز ثابت

نقطه اعشار در این اعداد، در مکان ثابتی از مقدار اعشاری قرار دارد.

این نوع اعداد در محاسبات پولی استفاده می‌شود، چون نباید عمل گرد کردن انجام شود.

تذکر: نوع ممیز ثابت در کوبول و C# استفاده می‌شود.

عیب اعداد ممیز ثابت این است که بازه مقادیر آن محدود است، زیرا از توان استفاده نمی‌شود و حافظه هدر می‌رود.

اعداد ممیز ثابت هم به صورت سخت افزاری و هم نرم افزاری پیاده سازی می‌شوند. نمایش داخلی آن‌ها با کد BCD است.

مثال: دستور مقابل را در C# در نظر بگیرید.

```
decimal    a = 1234.56 m;
```

حرف m به معنای ممیز ثابت است. متغیر a با مقدار 123456 و با ضریب مقیاس 4 ذخیره می‌شود.

نوع شمارشی (enumeration type)

مجموعه مرتبی از مقادیر مجزا می‌باشد. مقادیر نوع شمارشی براساس تعریف برنامه نویس مشخص می‌شوند.

دستور اول یک نوع شمارش به نام color و دستور دوم یک متغیر به نام x از آن نوع تعریف می‌کند.

```
enum   color {red , green , blue };
color    x;
```

به ثوابت شمارشی موجود در نوع شمارشی، به طور ضمنی مقادیر 0، 1 ، 2 و ... نسبت داده می‌شود. در مورد نوع color داریم:

هر عدد دلخواهی را می‌توان به ثوابت شمارش نسبت داد.

```
enum   size { small = 14 , medium = 15 , large = 16 ,  xlarge = 17 };
```

نمایش حافظه‌ای که برای نوع شمارش به کار می‌رود، همان نمایش برای مقادیر صحیح است، البته ساده‌تر چون بدون علامت هستند و مجموع مقادیر آنها کوچک هستند.

در Java ، نوع شمارشی وجود ندارد.

امتیاز عمده انواع شمارشی، افزایش قابلیت خوانایی و قابلیت اعتماد است.

نوع زیر بازه (subrange)

دنباله‌ای پیوسته از انواع ترتیبی (ordinal) ، مثل نوع صحیح و نوع شمارشی است.

عملیاتی که بر روی نوع زیر بازه انجام می‌شوند، همان عملیاتی هستند که بر روی نوع اصلی انجام می‌گیرند.

نمایش حافظه‌ای برای نوع زیر بازه، همان نمایش حافظه‌ای برای مقادیر صحیح و نوع شمارشی می‌باشد، با این تفاوت که به حافظه کمتری نیاز است.

نوع بولی

ساده‌ترین نوع است که بازه آن دو مقدار false و true است. عملیات متداول نوع بولی، and, or , not هستند.

نمایش حافظه شیء داده منطقی، یک بیت از حافظه است.

چون یک بیت از حافظه قابل آدرس دهی نیست، یک واحد آدرس دهی مثل بایت برای اشیای داده بولی منظور می‌شود.

بعضی از زبان‌ها مثل C فاقد نوع بولی هستند. در این زبان مقدار صفر نشان دهنده false و مقدار غیر صفر نشان دهنده true است.

نوع کاراکتری

داده‌های کاراکتری در اغلب کامپیوترها به صورت کدهای عددی است.

کد اسکی (۸ بیتی) و یونیکد (۱۶ بیتی) است.

۱۲۸ کاراکتر اول کد یونیکد معادل کدهای اسکی است.

زبان‌هایی مانند Java , C# , Java script از کد یونیکد استفاده می‌کنند.

مقادیر کاراکتری توسط سخت افزار و سیستم عامل پشتیبانی می‌شوند.

رشته‌های کاراکتری

A	L	I	'\0'

رشته‌ها در C , C++ به صورت آرایه‌ای از کاراکترها ذخیره می‌شوند.

انواع رشته: ۱- طول ثابت ۲- طول متغیر ۳- طول متغیر با حد معین

برای پیاده سازی رشته با طول ثابت، نیاز به توصیفگر زمان ترجمه و برای رشته با طول متغیر نیاز به توصیفگر زمان اجرا می‌باشد.

عملیات قابل انجام بر روی رشته‌ها:

۱- الحاق ۲- رابطه‌ای ۳- انتخاب زیر رشته ۴- فرمت بندی ورودی و خروجی ۵- تطبیق الگو

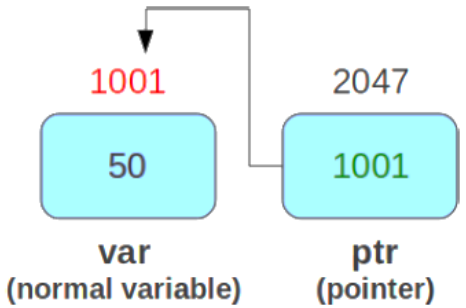
نوع داده اشاره‌گر

مقادیر اشاره گر : آدرس‌های حافظه و مقدار خاصی به نام تهی (null).

مقدار null آدرس معتبری نیست و معنی آن این است که اشاره‌گر فعلاً نمی‌تواند برای ارجاع به سلول حافظه‌ای مورد استفاده قرار گیرد.

نوع داده اشاره‌گر اسکالر نیست، چون برای مراجعه به اشیای دیگر استفاده می‌شوند و داده‌ای از نوع خاص را ذخیره نمی‌کنند.

```
int   *ptr;
int    x=50 ;
ptr =&x;
```



عملیات قابل انجام در مورد اشاره‌گرها:

۱- **انتساب** ۲- **محاسباتی** (+ و – به شکل محدود)

۳- **رابطه‌ای** ۴- **دستیابی به محتویات**

اشاره‌گر در زبان‌هایی مثل C , C++ که اعلان و کنترل نوع به طور ایستا انجام می‌شود، فقط به یک نوع اشاره می‌کند.
اشاره گر در زبان‌هایی که اشیای داده آن‌ها توصیفگر زمان اجرا دارند مثل smalltalk، اشاره‌گر ممکن است به هر نوع شیء داده‌ای اشاره کنند.

پاسکال	C++	C
Var p:^integer; New(p); P^:=2 ; Dispose(p);	Int *p; p=new int; *p=2 ; Delete p;	Int *p; p=(int *)malloc(sizeof(int)); *p=2 ; Free(p);

مشکلات اشاره‌گرها

```
int   * p ;
int   * q ;
p = new  int ;
q = new  int ;
p = q ;
```

۱- حافظه مازاد (زباله): (garbage)

قابل دستیابی نبودن بخشی از حافظه که در اختیار برنامه نویس نیست.(مسیر دستیابی به شیء داده از بین رفته باشد.)

توسط دستور آخر، اشاره‌گر p و q به یک محل حافظه اشاره می‌کنند و حافظه‌ای که قبلاً p بـه آن اشاره می‌کرد، به مدیریت حافظه تحویل داده نشده و قابل دستیابی نمی‌باشد.
این حافظه را زباله می‌گویند.

```
int   *p ;
int   *q ;
p = new  int ;
q = p;
free (p);
```

۲- ارجاع معلق (dangling reference)

با اجرای دستور یکی به آخر، اشاره‌گر q نیز به همان محلی که p اشاره می‌کند، اشاره خواهد کرد.
سپس حافظه ای که p به آن اشاره می‌کند به heap برگردانده می‌شود.

نوع داده مرجع (refrence)

مرجع: متغیری مانند اشاره‌گر که به محلی از حافظه اشاره می‌کند.

```
int   a = 0;
int   &b = a;
a = 5;
```

متغیر b ، مرجع متغیر a است.
a و b ، نام‌های مستعار هستند.
به کمک a یا b می‌توان محتویات a را تغییر داد.

متغیرهای مرجع در Java به طور کلی جانشین اشاره‌گرها شده‌اند.

c# حاوی متغیرهای مرجع است و به دلیل سازگاری با c , c++ شامل اشاره‌گرها نیز هست.

اشاره گر به استراکچر

```
struct    s {
    int  a ;
   float  b;
} k , *p;
```

```
k.a=2
(*p). a=2
p-> a =2
```

شیء داده (data object)

شیء داده، محلی که مقادیر در آن ذخیره و بازیابی می‌شوند.

این اشیای داده و روابط بین آنها در حین اجرای برنامه به طور پویا تغییر می‌کنند.

دسته بندی اشیای داده

۱- اشیای داده‌ای که در زمان اجرای برنامه وجود دارند:

الف- توسط برنامه نویس تعریف می‌شوند. (مانند متغیر، ثابت، آرایه، فایل)

ب- توسط سیستم ایجاد می‌شوند. (مانند پشته زمان اجرا، بافر فایل‌ها، رکوردهای فعالیت زیر برنامه‌ها و لیست فضای آزاد)

۲- از نظر ساختاری

الف- ابتدایی (elementary): مانند متغیر از نوع صحیح

ب- ساختمان داده (data structure) : مانند struct در c و record در Ada

طول عمر شیء داده

مدت زمانی که شیء داده به سلول بایند است.

طول عمر شی داده وقتی شروع می شود که به سلول حافظه ای بایند می‌شود و وقتی تمام می‌شود که بایند به سلول لغو گردد.

بعضی از اشیای داده در آغاز برنامه وجود دارند و بعضی در زمان اجرای برنامه به طور پویا ایجاد می‌شوند.

بعضی از اشیای داده در نقطه‌ای از اجرای برنامه از بین می‌روند و بعضی دیگر تا انتهای اجرای برنامه باقی می‌مانند.

هر شیء داده در طول عمر خود برای ذخیره مقادیر داده‌ها به کار می‌رود.

صفات و بایندهای شیء داده :

۱- نوع ۲- مکان (location) ۳- مقدار ۴- نام ۵- مولفه (component)

طول عمر اشیای داده به ۳ روش تخصیص حافظه مربوط می‌شود که برای مدیریت بر فضای اشیای داده به کار می‌روند:

۱- اشیای داده ایستا : آدرس مطلقی دارند و در طول اجرای برنامه وجود دارند.

۲- اشیای داده پشته : در هنگام ورود به زیر برنامه ایجاد و با خاتمه زیر برنامه از بین می‌روند

۳- اشیای heap: در هر زمانی ایجاد و حذف می‌شوند.

اشیای heap به الگوریتم مدیریت حافظه کلی‌تری نیاز دارند.

اسامی در زبان‌ها

اسم: رشته‌ای از کارکترها است که برای مشخص کردن نهادی (entity)در برنامه به کار می‌رود.

اغلب عناصر زبان دارای اسم (شناسه) هستند.

مانند متغیرها، پارامترهای مجازی، زیر برنامه‌ها، انواع تعریف شده، ثوابت تعریف شده، بر چسب و استثناها.

در زبان C وJava بین حروف کوچک و بزرگ در اسم‌گذاری تفاوت وجود دارد که به خوانایی برنامه آسیب می‌رساند و این نکته‌ی طراحی را نقض می‌کند که ساختارهای مشابه باید معنای یکسانی داشته باشند.

کلمه رزروی: کلمه خاصی از زبان برنامه سازی است که نمی‌تواند به عنوان اسم مورد استفاده قرار گیرد.

اسامی integer و float در Ada از پیش تعریف شده‌اند ولی رزروی نمی‌باشند و کاربر می‌تواند آن‌ها را دوباره تعریف کند.

ثابت

ثابت، یک شیء داده با نام است که باید آن به یک مقدار، فقط در زمان باید شدن به سلول حافظه انجام می‌شود.

استفاده از ثابت، میزان خوانایی برنامه را بالا می‌برد و انجام تغییرات در برنامه را آسان می‌کند.

زبان‌های مانند C++ اجازه می‌دهند که مقادیر به طور پویا (در زمان اجرا) به ثوابت با نام باید شوند.

در مثال زیر، مقدار x در زمان اجرا مشخص می‌گردد و براساس آن، باید مقدار به ثابت y به طور پویا در زمان اجرا صورت می‌گیرد و از آن پس قابل تغییر نیست.

```
cin >> x;
const int y=x+1;
```

در C#، دو نوع ثابت دارای نام را می‌توان تعریف کرد:

۱- const

ثابتی که با const تعریف می‌شود، به طور ایستا به مقادیر باید می‌شوند یعنی مقادیر آن‌ها در زبان ترجمه تعیین می‌گردد.

۲- readonly

ثابتی که با readonly تعریف می‌شود، به طور پویا به مقادیر باید می‌شود. یعنی مقادیر آنها می تواند در دستور اعلان یا یک سازنده ی ایستا تعیین شود.

متغیر: شیء داده‌ای که توسط برنامه‌نویس تعریف و نام‌گذاری می‌شود. متغیرها با صفاتی مثل نام، آدرس، نوع، طول عمر و حوزه مشخص می‌شوند.

نوع متغیر: مشخص می‌کند آن متغیر چه مقادیری را می‌تواند بپذیرد و مجموعه عملیاتی که بر روی متغیر انجام می‌شوند، کدام‌اند.

آدرس متغیر: آدرس حافظه‌ای که به متغیر مربوط می‌شود. آدرس متغیر ممکن است در زمان‌ها و موقعیت‌های مختلف، آدرس‌های مختلفی داشته باشد. به آدرس متغیر، **L– value** نیز می‌گویند، چون وقتی به آدرس متغیر نیاز است که در سمت چپ دستور انتساب قرار گیرد.

مقدار متغیر: محتویات سلول حافظه‌ای که در اختیار متغیر قرار دارد. مقدار متغیر را **r– value** نیز می‌نامند، زیرا وقتی استفاده می‌شود که متغیر در سمت راست دستور انتساب قرار گیرد.

نام‌های مستعار : متغیر ممکن است در طول عمرش، بیش از یک نام داشته باشد که آن نام‌ها را مستعار (alias) می‌گویند.نام مستعار به قابلیت خوانایی برنامه آسیب می‌رساند، چون یک متغیر را به چند طریق می‌توان دستکاری کرد. وقتی دو اشاره‌گر به یک شیء داده اشاره می‌کنند، نام‌های مستعار هستند.
همچنین ساختار union نام‌های مستعار ایجاد می‌کند.

ساختمان داده

ساختمان داده، دسته‌ای از انواع داده‌ها می‌باشند که مولفه‌های آنها اشیای داده‌ی دیگر هستند.
مانند بردارها، آرایه‌ها، رکوردها، رکوردهای طول متغیر، مجموعه‌ها و لیست‌ها.

صفات متداول ساختمان داده‌ها :

۱- تعداد عناصر ۲- نوع هر عنصر ۳- اسامی برای انتخاب عناصر ۴- حداکثر تعداد عناصر ۵- سازمان عناصر

اگر تمام عناصر ساختمان داده از یک نوع باشند، آن را همگن می‌گویند، مانند آرایه‌ها. (لیست‌ها و رکوردها، ناهمگن هستند.)

عناصر ساختمان داده ممکن است دارای سازمان ترتیبی باشد(مانند آرایه یک بعدی) و می‌تواند دارای سازمان غیر ترتیبی باشد(مانند آرایه چند بعدی).

عملیات متداول در مورد ساختمان داده‌ها

۱- انتخاب عنصر (مانند دسترسی به عنصری از آرایه)

۲- عملیات بر روی کل ساختمان (مانند انتساب رکوردی به رکورد دیگر) (جمع آرایه‌ها)

۳- درج و حذف عناصر (مانند ایجاد و حذف لیست پیوندی)

عملیات انتخاب عنصر

برای پیاده سازی عملیات انتخاب عنصر در نمایش ترتیبی و یا پیوندی، دو حالت تصادفی و ترتیبی، در نظر گرفته می‌شود.

آدرس شروع بلوک، آدرس پایه و محل نسبی عنصر داخل بلوک ،آدرس آفست نام دارد.

مثال: نحوه‌ی دسترسی به عنصر سوم آرایه‌ای از کاراکترها به طول 4 و به نام x در زبان c :

آدرس عنصر سوم یعنی x[2] : Lvalue(x) + 2 .

آدرس پایه : آدرس x[0] یا Lvalue (x)

به طور کلی آدرس x[i] : Lvalue(x)+i

اعلان ساختمان داده‌ها

اعلان در ساختمان داده‌ها در حالت کلی مانند اعلان در انواع داده اولیه است، با این تفاوت که صفات بیشتری باید مشخص شوند.
به عنوان مثال، اعلان یک آرایه، صفاتی چون تعداد ابعاد، بازه اندیس، تعداد عناصر و نوع هر عنصر را مشخص می‌کند.
اعلان این صفات، نمایش حافظه ترتیبی را برای آن مشخص می‌کند که فرمول دستیابی به عناصر آن در زمان ترجمه مشخص می‌شود.

کنترل نوع ساختمان داده

کنترل نوع ساختمان داده‌ها، از کنترل نوع در انواع داده اولیه، پیچیده‌تر است، چون در انتخاب عنصری از ساختمان داده، مسیر انتخاب ممکن است طولانی باشد.
همچنین ممکن است پارامترهای عملیات انتخاب درست باشد، ولی عنصر مورد نظر موجود نباشد.

آرایه

آرایه ایستا

مانند آرایه‌هایی که در C و C++ در داخل توابع با واژه static اعلان می‌شوند.

آرایه پویای پشته‌ای با طول ثابت

مانند آرایه‌هایی که در C و C++ در داخل توابع و بدون واژه static اعلان می‌شوند.

آرایه پویای پشته‌ای

اندازه آرایه تا زمان استفاده از آرایه ممکن است مشخص نباشد.

آرایه پویای heap با طول ثابت

در C از malloc( ) و free( ) و در C++ از عملگرهای new و delete استفاده می‌شود.

آرایه پویای heap

در C# از کلاس Array list برای ایجاد آرایه پویای heap استفاده می‌شود.

آرایه‌های یک بعدی (بردارها)

از مشخصات بردار، می‌توان تعداد عناصر، نوع هر عنصر و اندیس برای انتخاب عنصر را نام برد.

عملیات روی بردارها عبارتند از:

۱- اندیس گذاری (انتخاب عنصری از بردار)
۲- ایجاد و حذف بردارها
۳- عملیات روی کل بردار

کد مربوط به دستیابی به عناصر آرایه باید در زمان ترجمه تولید شود. این کد در زمان اجرا، اجرا می‌شود تا آدرس عنصر مورد نظر تولید شود.

اگر حدود اندیس ایستا باشد و در زمان اجرا تغییر نکند، می‌تواند در کد گنجانده شود و در توصیفگر زمان اجرا ذخیره نگردد.

تابع دستیابی

آدرس x[i] : $\alpha + (i - Lb) \times e$

که α ، آدرس اولین عنصر ، Lb، حد پایین بردار و e، اندازه هر عنصر می باشد.

این تابع را می‌توان به صورت $(\alpha - Lb \times e) + i \times e$ نوشت.

بخش $\alpha - Lb \times e$ ،مقداری ثابت است که می‌تواند در زمان ترجمه محاسبه گردد و بخش $i \times e$ ، متغیر است که در زمان اجرا محاسبه می‌شود و به مقدار ثابت اضافه می‌شود.

البته اگر i نیز در زمان ترجمه مشخص باشد، آنگاه همه محاسبه‌ها در زمان ترجمه انجام می‌شود.

مبداء مجازی Virtual Original (VO) :

$$address(x[i]) = VO + i \times e$$

توصیفگر زمان ترجمه برای بردار

توصیفگر زمان ترجمه

بردار
نوع عناصر
نوع اندیس
حد پایین اندیس
حد بالای اندیس

در صورت استفاده از مبدا مجازی :

مبدا مجازی
حد پایین
حد بالا
اندازه سطر

امتیاز این کار در این است که می‌توان در هنگام ارسال بردار به زیر برنامه، توصیفگر را به زیر برنامه ارسال کرد، در حالی که عناصر بردار در جای دیگری ذخیره شده‌اند.

در این نوع نمایش، نیازی به ذخیره نوع عناصر و نوع اندیس نمی‌باشد.

در زبان c ، این اطلاعات در زمان ترجمه بایند می‌شوند و نیاز به حضور آن‌ها در زمان اجرا نیست.

آرایه‌های دو بعدی

آرایه‌های دوبعدی(ماتریس) از تعدادی سطر و ستون تشکیل شده‌اند.
در ماتریس برای انتخاب یک عنصر از دو اندیس استفاده می‌شود.

آرایه دو بعدی در c : `int x[4] [2] ;`

آرایه دو بعدی در Ada : `x: array (0 . . 3 , 0 . . 1)`

در APL ، اگر x وy دو آرایه دو بعدی باشند، x+y و x+.y به ترتیب دو آرایه را جمع و ضرب می‌کند.

ذخیره آرایه دو بعدی

از آنجا که حافظه سخت افزار، خطی است، مقادیر انواع داده‌ای باید در یک حافظه یک بعدی ذخیره شوند که به دو روش سطری (row major) و یا ستونی (column major) ذخیره می‌شوند.

همه زبان‌ها به جز زبان Fortran از نمایش سطری استفاده می‌کنند.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

سطری: 1 , 2 , 3 , 4 , 5 , 6

ستونی : 1 , 3 , 5 , 2 , 4 , 6

عناصر آرایه

آدرس x [i] [j] :

$$address(x[i][j]) = \alpha + (i - l_1) \times s + (j - l_2) \times e$$

$$s = (u_2 - l_2 + 1) \times e$$

استفاده از مبدا مجازی:

$$address(x[i][j]) = VO + i * s + j * e$$

$$VO = \alpha - l_1 \times s - l_2 \times e$$

α, e, s, VO : در هنگام ایجاد آرایه ثابت‌اند و یکبار ذخیره و محاسبه می‌شوند.

مثال: توصیفگر آرایه x[1 .. 4 , 1 .. 3] را مشخص کنید.
اندازه هر عنصر آرایه یک است. (e=1)

$$s = (u_2 - l_2 + 1) \times e$$ S= (3 – 1 + 1) × 1 = 3

$$VO = \alpha - l_1 \times s - l_2 \times e$$ Vo = α – 1 × 3 – 1 × 1 = α – 4

α-4	1	4	1	3	1

برش آرایه

برش آرایه، بخشی از آرایه است که خودش نیز آرایه است.
در fortran 95 از برش‌های آرایه زیاد استفاده می‌شود.

به طور نمونه آرایه دو بعدی x(1 .. 4,1 .. 3) با چهار سطر و سه ستون مفروض است.
انتخاب ستون دوم : x(1:4 , 2)
انتخاب ستون‌های اول و دوم : x(1:4 , 1:2)

در زبان Ada برش برای آرایه‌های یک بعدی متداول است.

به طور نمونه اعلان زیر را در نظر بگیرید. x: array (1 .. 10) of integer
یک برش : x (3 .. 6)

آرایه‌های انجمنی

آرایه انجمنی (associative) : مجموعه‌ای از عناصر داده‌ها که اندیس آن‌ها مجموعه‌ای از مقادیر به نام کلید است.
این کلیدها باید ذخیره شوند. بنابراین هر عنصر آرایه انجمنی، جفتی از کلید و مقدار است.
در C++ به نام map شناخته می‌شود.

زبان perl از آرایه‌های انجمنی زیاد استفاده می‌کند.
اندازه آرایه انجمنی در perl پویا است، یعنی اندازه آن با افزودن عنصر به آرایه افزایش و با حذف عنصر، کاهش می‌یابد.
در زبان perl ، برای پیاده سازی آرایه انجمنی از جدول درهم‌سازی استفاده می‌شود.

مثال: تعریف یک آرایه انجمنی در زبان perl :

```
%x = ( "Maryam" => 17 , "Ali" => 20 );
```

که اسامی دانشجویان نقش کلید و معدل آن‌ها نقش مقادیر را دارند. (هر متغیر درهم‌سازی در perl با علامت ٪ شروع می‌شود.)

آرایه‌ها در زبان PHP به صورت معمولی و انجمنی وجود دارند.

ساختمان(استراکچر)

ساختمان یا رکورد، مجموعه‌ای ناهمگن از عناصر است که هر عنصر دارای نام است و از طریق نام دستیابی می‌شوند.
در C ، C++ ، C# از دستور struct برای تعریف ساختمان استفاده می‌شود.

```
struct    student {
    int     id;
    float   ave;
    char    name[30];
};
student   x;
```

فیلدهای رکورد در محل‌های متوالی حافظه ذخیره می‌شوند.

توصیفگر زمان ترجمه برای یک ساختمان با دو فیلد :

آدرس اولین عنصر	آفست فیلد ۲	نوع فیلد ۲	نام فیلد ۲	آفست فیلد ۱	نوع فیلد ۱	نام فیلد ۱

آفست هر عنصر در زمان ترجمه محاسبه می‌شود.
نیاز به توصیفگر زمان اجرا برای استراکچر نمی‌باشد.

فرمول دستیابی به عنصر i ام رکورد R

$$address(R.i) = \alpha + f_i$$

α : آدرس شروع بلوک حافظه رکورد

f_i : آفست عنصر i

$$address(R.i) = \alpha + \sum_{j=1}^{i-1} (sizeof\ R.j)$$

می توان f_i را در زمان ترجمه حساب کرد و در زمان اجرا آدرس پایه را به آن اضافه کرد.

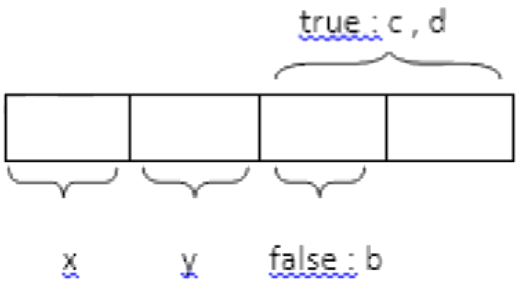
رکورد با طول متغیر

این نوع رکورد از دو بخش ثابت و متغیر تشکیل شده است.
بخش ثابت حاوی فیلدهایی است که بین هر یک از ساختارهای مختلف مشترک است و بخش متغیر از ساختاری به ساختار دیگر متفاوت است.

در C , C++ , C# ، با استفاده از **union** ساخته می‌شوند.

در بعضی از زبان‌های برنامه سازی مانند Ada ، عنصری به نام بر چسب در بخشی از ثابت در نظر گرفته می‌شود تا مشخص کند که در هر لحظه از زمان اجرا چه ساختاری از رکورد وجود دارد.

```
type  a ( x : boolean)  is  record
      y : integer;
    case   x   is
        when    false => b : integer;
        when    true  => c : float , d : integer;
    end     case
end      record
```



مجموعه

مجموعه : شیء داده‌ای که شامل مقادیر نامرتب و مجزا است.

ترتیب مقادیرموجود در مجموعه مهم نیست.

عملیات متداول در مجموعه : عضویت، درج، حذف، اجتماع، اشتراک و تفاضل.

مجموعه‌ها به دو دسته متناهی و نامتناهی تقسیم می‌شوند.

در مثال زیر که به زبان پاسکال است، a یک مجموعه متناهی و b یک مجموعه نامتناهی از اعداد صحیح است.

```
type
    a  set   of   (red, blue, green);
    b  set   of   integer;
```

برای پیاده سازی مجموعه‌های متناهی از نمایش بیتی و برای پیاده سازی مجموعه‌های نامتناهی از درهم سازی استفاده می‌شود.

در پیاده سازی مجموعه به روش درهم سازی، عملیات عضویت و حذف مقادیر با کارایی بالایی انجام می‌شود. اما پیاده سازی عملیات اجتماع، اشتراک و تفاضل کارآمد نمی‌باشد.

بسته بندی (encapsulation)

بسته بندی (encapsulation)، راهکاری که اجازه می‌دهد ثوابت، انواع، متغیرها و متدها در یک موجودیت جدید، با هم دسته بندی شوند.

پکیج‌ها، کلاس‌ها و زیر برنامه‌ها، نمونه‌ای از این موجودیت‌ها می‌باشند.

برنامه به کمک این راهکار می‌تواند حوزه (scope) و قابلیت رویت (visibility) مقادیر و توابع بسته بندی شده برای این انواع جدید را محدود کند.

انتزاع (abstraction)

انتزاع : نمایشی از یک موجودیت که صفاتی را که در زمینه خاصی اهمیت دارد را در برمی‌گیرد.
انتزاع از پیچیدگی برنامه نویسی می‌کاهد.

با استفاده از انتزاع، برنامه نویسان به صفات اساسی می‌پردازند و از صفات غیر اساسی چشم پوشی می‌کنند.
دونوع انتزاع در زبان‌های جدید وجود دارد:

۱- انتزاع داده‌ها
۲- انتزاع فرآیندی

زیر برنامه‌ها، نوعی انتزاع فرآیندی را فراهم می‌کنند. زیرا کاربران بدون دانستن جزییات زیر برنامه، با ارسال پارامترهایی به آن، عملیاتی را انجام می‌دهند.

نوع داده انتزاعی

شیء، نمونه‌ای از نوع داده انتزاعی می‌باشد.
نوع داده انتزاعی، برای بسط مفهوم بسته بندی به کار می‌رود.
از نظر نحوی، شامل نمایش داده‌ها مربوط به یک نوع داده و زیر برنامه‌هایی برای انجام عملیات بر روی آن نوع داده‌ها است.

مثال: فرض کنید پیاده سازی اصلی انتزاع پشته، از نمایش همجواری استفاده شود، یعنی پشته با آرایه پیاده سازی شده باشد و به دلیل وجود مدیریت حافظه‌ای، به نمایش لیست پیوندی تغییر کند.
این تغییر در کدی که نوع پشته را تعریف می‌کند، انجام می‌شود ولی در برنامه‌ای که از آن استفاده می‌کند، تغییر ایجاد نمی‌شود.

زبان‌هایی مثل Java, C#, C++ مستقیماً از نوع داده انتزاعی پشتیبانی می‌کنند.

مشخصات و پیاده سازی زیر برنامه

زیر برنامه یک عملیات انتزاعی است.
اگر زیر برنامه فقط یک مقدار را برگرداند تابع (function) و اگر چند مقدار را برگرداند رویه (procedure) نامیده می‌شود.
رویه مقداری را با نام خود برنمی‌گرداند.

به عنوان مثال f1 یک تابع و f2 یک رویه است:

```
int     f1( int   a , float   b);
void    f2 (int   a , float   *b, int  *c);
```

علامت * در کنار پارامترهای b و c نشان می‌دهد که تغییرات ایجاد شده در زیربرنامه، در برنامه فراخوان قابل مشاهده است.

الگوی این زیر برنامه‌ها :

f1 : int × float → int

f2 : int × float × int → float × int

پیاده سازی تعریف و سابقه فعالیت زیر برنامه

فراخوانی زیر برنامه موجب می‌شود یک سابقه فعالیت (activation) از زیر برنامه ایجاد گردد و پس از خاتمه زیر برنامه، آن سابقه فعالیت از بین می‌رود.

تعریف زیر برنامه به عنوان قالبی (template) برای سابقه فعالیت آن عمل می‌کند.

سابقه فعالیت زیر برنامه شامل دو بخش است:

۱- ایستا

این بخش سگمنت کد نام دارد و حاوی ثوابت، لیترال‌ها و کد اجرایی هر یک از دستورات است.

۲- پویا

این بخش رکورد فعالیت نام دارد و شامل پارامترها، نتایج تابع، داده‌های محلی، ناحیه حافظه موقت، نقاط برگشت و پیوندهایی برای مراجعه به متغیرهای غیر محلی است.

ساختار بخش پویا، برای تمام سوابق فعالیت زیر برنامه یکسان است، ولی مقادیر متفاوتی در آنها وجود دارد.

اگر همزمان چند سابقه فعالیت از زیر برنامه وجود داشته باشد (مثل زیر برنامه بازگشتی)، چند رکورد فعالیت از آن زیر برنامه وجود خواهند داشت ولی فقط یک بخش ایستا برای همه‌ی آنها کافی است.

لازم نیست الگوی کامل رکورد فعالیت در زمان اجرا وجود داشته باشد.

مقدمات (prologue) و اختتامیه (epilogue)

وقتی زیر برنامه فراخوانی می‌شود، چند عمل صورت می‌گیرد که از دید کاربر پنهان است، مثل آماده سازی رکورد فعالیت، انتقال پارامترها. چون این اعمال باید قبل از اجرای دستورات زیر برنامه انجام شوند،آنها را مقدمات گویند.

اعمال مقدمات توسط مترجم صورت می‌گیرد. برای این منظور، مترجم بلوکی از کد را در آغاز زیر برنامه قرار می‌دهد.

هنگام خاتمه زیر برنامه، اعمال دیگری صورت می‌گیرد تا نتایج برگردانده شوند و حافظه مربوط به رکورد فعالیت آزاد شود. این اعمال را اختتامیه می‌نامند.

هم ارزی نوع (type equivalence)

دو روش برای تعریف هم ارزی نوع وجود دارد:

۱- هم ارزی نام:

نوع دو متغیر در صورتی هم ارز است که یا در یک دستور اعلان تعریف شوند یا در دستورات اعلانی تعریف شوند که از یک نوع استفاده می‌گردد.

۲- هم ارزی ساختاری:

نوع دو متغیر در صورتی هم ارز است که انواع آن‌ها ساختارهای یکسانی داشته باشند. یعنی از نمایش حافظه یکسانی استفاده کنند.

هم ارزی ساختاری نسبت به هم ارزی نام، قابلیت انعطاف بیشتری دارد.

مثال

دستورات زیر در C++ مفروض است:

```
void   f (s1  x );
int      main( ){
  s1  y , z;
  s2  w;
  y = z;
  w = y;
  f(w);
}
```

```
struct   s1 {
   int    a ;
  char   b ;
};
```

```
struct   s2 {
   int    c ;
   char  d ;
};
```

انواع s1 , s2 دارای تعداد و نوع داده‌های یکسانی می‌باشند و نمایش حافظه آن‌ها یکسان است.

متغیرهای **x,y,z** که با نوع s1 تعریف شدند، هم ارزی **نام** دارند.

متغیرهای **x,w,y,z** هم ارزی **ساختاری** دارند.

مثال

دو آرایه زیر هم ارز نمی‌باشند، چون فاقد نام نوع هستند. (گرچه ساختار یکسانی دارند)

```
x : array ( 1..3 ) of integer ;
y : array ( 1..3 ) of integer ;
```

برای اینکه هم ارز باشند، باید به صورت زیر تعریف شوند:

```
type  k   is  array (1..3) of integer ;
x,y :  k ;
```

typedef در C و C++ نوع جدیدی را تعریف نمی‌کند و فقط نام جدیدی برای نوع موجود در نظر می‌گیرد.
بنابراین هر نوعی که با typedef تعریف می‌شد، با نوع والد خود هم ارز است.

در زبان‌های که کاربران اجازه ندارند انواعی را تعریف و نام گذاری کنند، نمی‌توان از هم ارزی استفاده کرد. (مثل فرترن و کوبول)

در C برای struct ، union و enum ، از هم ارزی نام استفاده می‌کند.

عملیات انتساب

عملیات انتساب، بایند یک مقدار به متغیر را به طور پویا تغییر می‌دهد که این تغییر به عنوان اثر جانبی عملیات انتساب محسوب می‌شود، زیرا در اغلب زبان‌ها، عملیات انتساب، مقداری را بر می‌گرداند.

زبان‌هایی مانند C , C++ , C# , Java از این مشخصات پیروی می‌کنند.

به همین دلیل در این زبان‌ها می‌توان انتساب را به عنوان عبارات و عملوندی در عبارات به کار برد.

این کار دارای معایبی است، از جمله: عدم امنیت و کاهش قابلیت خوانایی عبارات.

مثال: دستور if (a == b) برای مقایسه a , b نوشته شده تا در صورت مساوی بودن a , b، دستوراتی اجرا شوند. حال اگر اشتباهاً تایپ شود: if(a=b) ، کامپایلر خطایی اعلان نمی‌کند و به جای تست مساوی بودن a , b ، مقداری که به a نسبت داده می‌شود (همان مقدار b)، تست می‌شود.

این حالت نمونه دیگری از عدم امنیت برنامه‌های C , C++ است.

در C# فقط می‌توان از عبارات بولی در دستور if استفاده کرد و این مسئله رخ نمی‌دهد.