جزوه‌ی درس

طراحی زبان‌های برنامه‌سازی

فصل چهارم : داده‌های ساختیافته

در این فصل چگونگی پیاده سازی اطلاعات از نوع ساختیافته که چندین مقدار را در خود ذخیره می کنند را بررسی می کنیم .

تعاریف اولیه :

(Structured Data Type) : SDT

یک SDT ، DO ای است که شامل تعداد EDT یا SDT دیگر می باشد .

هر SDT شامل تعدادی مؤلفه می باشد که همان اجزاء اطلاعاتی را در خودش دخیره می کند .

نمونه های مهم برای SDT :

Records,Stack,Array,Sets,Lists,Files,Pointers,Strings,Queue,Tree,Graph,…

مشخصات SDT ها :

1. **تعداد مؤلفه ها :** یکی از ویژگی های SDT ها تعداد مؤلفه های آن می باشد که می تواند ثابت یا متغیر باشند .
   - Record, Array : Fixed Size
   - …,Stack,String,Tree,Graph : Variable Size

2. **نوع مؤلفه :**
   - همگن (Homogenous) : نوع همه خانه ها یکی باشد مثل Array,Set,File ,…
   - ناهمگن (Heterogeneous) : نوع خانه ها یکی نباشد مثل Record,List ,…

3. **نام مؤلفه :** نامی که برای انتخاب هر مؤلفه به کار گرفته می شود .
   - برای آرایه ها از نام آرایه به همراه شماره اندیس استفاده می شود .
   - برای Stack از نام پشته به همراه عملیات Push و Pop
   - برای Record از نام خود Component در حقیقت نام رکورد + نام فیلد

4. **حداکثر تعداد مؤلفه ها :**

برای SDT هایی که طول متغیر دارند یک حداکثر برای نعداد مؤلفه ها باید در نظر گرفت مثل Stack ها و رشته‌های با طول متغیر .

5. **سازماندهی مؤلفه ها :** منظور چگونگی ذخیره عناصر می باشد که بر این اساس دو دسته سازماندهی وجود دارد :
   - خطی : آرایه ، لیست پیوندی ، فایلهای عادی و …
   - غیرخطی : درخت ، گراف و …

عملیات روی SDT

1. عملیات انتخاب مؤلفه :

- Random : آرایه یا رکورد
- Sequential : صف ، پشته ، لیست پیوندی

2. عملیات روی کل ساختار SDT :

می توان عملیاتی را روی کل ساختار در زبانهای برنامه نویسی انجام داد که این عمل معمولا با برنامه نویسی روی کل ساختار انجام می شود.

- جمع دو آرایه در Ada و Snobol
- انتساب یک رکورد به رکورد دیگر
- Union گرفتن دو مجموعه

☑ زبانهای Snobol و APL مجموعه هایی غنی از اعمالی را که بر روی کل ساختار عمل می کنند را فراهم می نماید .

3. ایجاد شدن یا از بین رفتن مؤلفه :

تمامی SDT های از نوع طول متغیر باید مکانیزم هایی برای تخصیص فضا و آزادسازی فضا برای مؤلفه های آن SDT داشته باشند مانند دستورات free و malloc در C

این قسمت یکی از بخشهای مهم Storage management می باشد .

4. ایجاد شدن و از بین رفتن کل SDT

بعضی از زبانها مکانیزم هایی برای این منظور در نظر گرفته اند . به عنوان مثال یک String می تواند ایجاد شود یا از بین رود .

پیاده سازی SDT :

از جهت پیاده سازی چگونگی ذخیره اطلاعات و چگونگی انجام عملیات باید مشخص گردد.

چگونگی ذخیره اطلاعات ⟵ نمایش های حافظه

انتخاب کارآمد مولفه های SDT
مدیریت حافظه کارآمد

حافظه مربوط به SDT

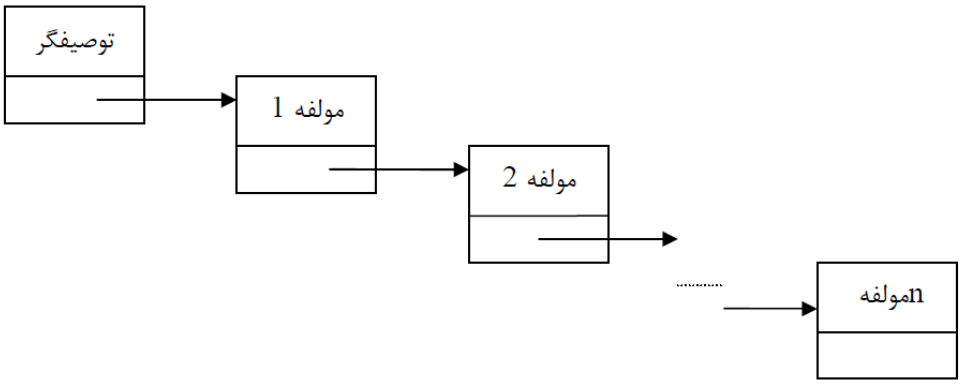
- حافظه ای برای مولفه های SDT
- توصیفگر دلخواه برای ذخیره برخی یا همه صفات SDT

چگونگی ذخیره اطلاعات:

1. نمایش ترتیبی: در fixed size ها و variable size های همگن استفاده می شود.

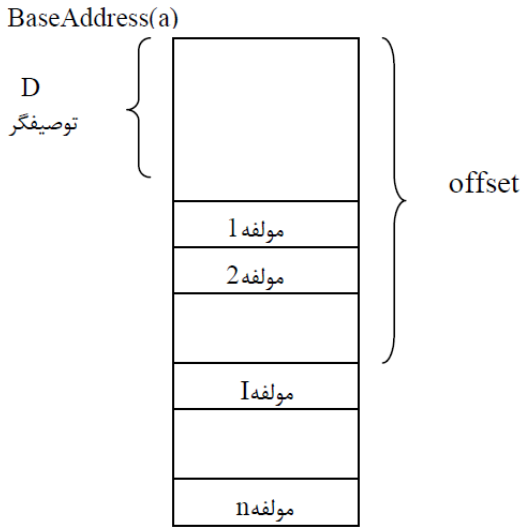
توصیفگر
مولفه 1
مولفه 2
...
مولفه n

2. نمایش پیوندی: در variable size ها استفاده می شود .



توصیفگر شامل اطلاعاتی ست که برای توصیف SDT جهت انجام عملیات استفاده می شود.در مکانیزم ترتیبی مولفه ها به دنبال هم در یک فضای پیوسته از حافظه قرار دارند و در مکانیزم پیوندی بین هر مولفه و مولفه بعدی یک پیوند (Link) وجود دارد.

پیاده سازی عملیات روی SDT :



Offset(I)=D+(I-1)*Component size

آدرس مولفه Iام =a+offset(I)

جهت انجام عملیات روی SDT باید یک مسیر دستیابی برای هر مولفه ایجاد شود. در مکانیزم ترتیبی می توان از offset +base address برای دسترسی به هر مولفه استفاده کرد.
در مکانیزم ترتیبی کل SDT دارای یک Life Time می باشد .لی در مکانیزم پیوندی هر مولفه Life Time مخصوص به خود را دارد.

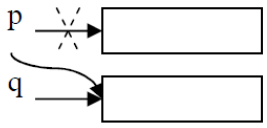
☑ دوره زندگی(Life Time) هر .D.O زمانی آغاز می شود که تقید آن به مکان خاصی از حافظه انجام شود و زمانی خاتمه می یابد که تقید آن به بلوک حافظه حذف گردد.

برای دسترسی به مولفه های SDT معمولا اعمال سخت افزاری وجود ندارد(بجز SDT های fixed size که نمایش ترتیبی دارند.)

دو مشکل عمده زیر جهت مسیر دستیابی به مولفه ممکن است ایجاد شود (برای حالت پیوندی)

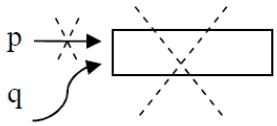
1. Garbage (زباله ی آشغال): در این حالت.D.O هنوز وجود دارد ولی مسیر دستیابی به آن از بین رفته است. مانند قطعه کد زیر:

```
Int *p,*q:
P=malloc(size of (int));
P=q;
```



2. Dangling Reference (رجوع رها شده): عکس حالت بالا می‌باشد یعنی – وجود ندارد ولی مسیر دستیابی به آن معتبر است(وجود دارد.)

```
Int *p,*q:
P=malloc(size of (int));
P=q;
Free(p);
```



مشکل کمتری را نسبت به رجوع رها شده ایجاد می کند و کلا ارجاعات معلق مشکل خاص عمده ای را برای مدیریت حافظه اییجاد می کنند. این ارجاعات ممکن است جامعیت کل ساختار زمان اجرا را طی اجرای برنامه تهدید نماید. مانند متغیرهای وحشی در C.

تعریفSDT و انجام عمل چک کردن نوع اطلاعات :

مثال:

A: Array [1:10,-5:5] of Real

ویژگی های SDT فوق:

- Data Type : آرایه
- تعداد ابعاد:2
- شماره index های سطر: 10 ، ... ،2، 1
- شماره index های ستون: 5، 4، ... ، 4-، 5-
- تعداد مولفه ها: 110= (1+1-10)(1+5+5)
- هر مولفه: Real

به دلیل زیاد بودن ویژگی ها، مسئله چک کردن نوع SDT ها بسیار پیچیده تر از چک کردن نوع برای EDT هاست.

دو مشکل عمده زیر وجود دارد :

1. ***وجود مولفه انتخاب شده و کنترل آن*** : برای SDT های fixedsize باید ارزش offset برآوردشود و چون این ارزش در زمان اجرا محاسبه می شود باید از runtime checking استفاده شود.
   برای SDT های variablesize باید Null بودن اشاره گرها چک شود و از تکنیک های مدیریت حافظه کمک گرفته شود.
2. ***نوع مولفه انتخاب شده جهت عملیات بررسی شود***: مشخص کردن نوع مولفه به راحتی ممکن نیست . به دلیل پیچیدگی موضوع پیشنهاد شده است که این عمل به صورت static و در زمان کامپایل انجام شود.
   مثال: A[2,3].link^ item

در زیر برخی از SDT های مهم بررسی شده اند.

* آرایه ها و ماتریس ها :

مشخصات آرایه ها :

الف) تعداد مولفه ها (معمولا ثابت است)

ب) نوع هر مولفه (همگن می باشد)

ج) اندیس (زیر نویس-index) برای استفاده از مولفه

- از محدوده خاصی برای index استفاده شود مثل C :int A[10]
- از هر نوع داده ترتیبی یا هر بازه‌ای می توان استفاده کرد مثل پاسکال:

A: Array ['A'..'N'] of integer
A: Array [Boolean] of integer
A: Array [-20..32] of Real

عملیات :

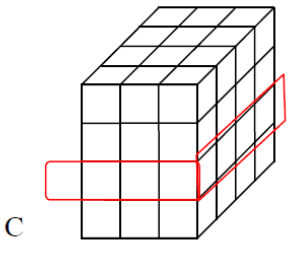
1. آدرس دهی یک مولفه
2. انتساب کردن (یک مقدار)به یک مولفه (مانند EDTها)
3. ایجاد شدن و از بین رفتن آرایه
4. عملیات ریاضی روی آرایه ها(مثل جمع دو آرایه که در زبان های APLو Ada و SNOBOL ارائه شده است.)

☑ یکی از مشکلات عمده پیاده سازی چنین اعمالی روی کل SDT مربوط به حافظه لازم برای ذخیره نتیجه است.

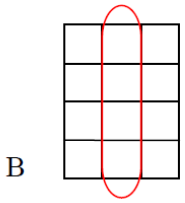
5. تجزیه ترکیب آرایه ها (در بعضی از زبان ها مثل فرترن و APL و PL1 این امکان رائه شده است.)

منظور از تجزیه یک آرایه برداشتن یک قطعه (برش) از یک آرایه بزرگتر می باشد که این امر می تواند سطری یا ستونی باشد .

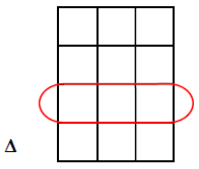
مثال :



C(3,*,*)
C(3,1:3,1:4)



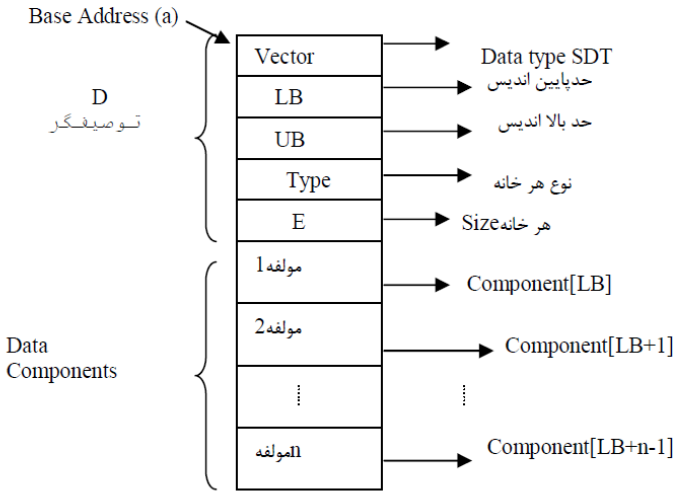
B(*,2)
B(1:4,2)



A(3,*)
A(3,1:3)

:PL1
:Fortran

پیاده سازی آرایه در حافظه :



Location of A[I]=
a + D + [I-LB] * E

☑ برای چک کردن وجود هر مولفه باید شرط LB<=I<=UB چک شود که اگر این شرط برقرار نباشد یک Runtime error تولید کند.

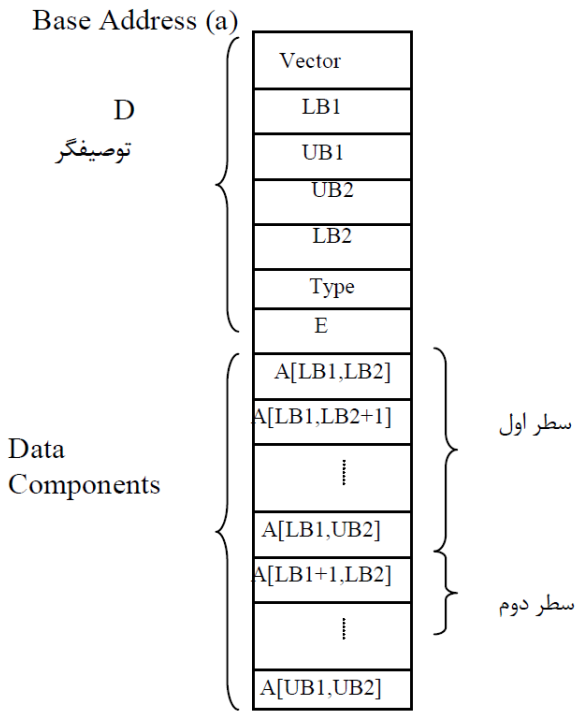
نمایش های حافظه ای فشرده و غیر فشرده :

در حافظه غیر فشرده (مثل آرایه بالا و فرمول آن) فرض می شود که اندازه هر مولفه(E) مقدار صحیحی از واحدهای حافظه(کلمه یا بایت) را شامل باشد. پس می توان از فرمول فوق به راحتی استفاده نمود ولی در نمایش حافظه فشرده مولفه های بردار (آرایه) یا هر ساختار دیگری به صورت ترتیبی شده اند بدون توجه به اینکه هر مولفه در ابتدای یک کلمه یا بایت قابل آدرس دهی قرار گیرد. مزیت آن صرفه جویی زیادی در حافظه می باشد و بدی آن اینست که متاسفانه دسترسی به مولفه ای از ساختار فشرده معمولا هزینه زیادی دارد زیرا فرمول دستیابی ساده فوق نمی تواند استفاده شود. که به دلیل هزینه بالای زمانی در هنگام اجرااز این حالت صرف نظر می شود.

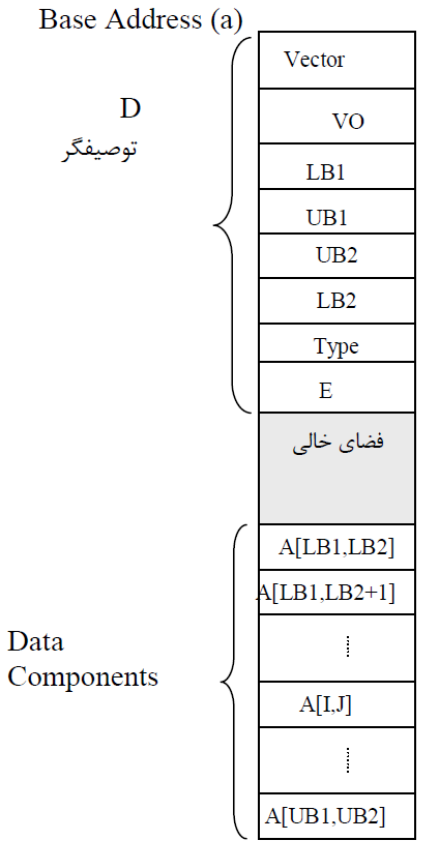
♦ آرایه های چند بعدی:

از دید چگونگی ذخیره سازی اطلاعات آرایه می تواند دو حالت داشته باشد.

1. روش ستونی(Column Major)
2. روش سطری (Row Major)



- ☑ اگر اندیس های I و J از نوع ثابت باشد آنگاه مترجم می تواند فرمول فوق را به صورت دستی حساب کرده و مقدار آن را در زمان کامپایل مشخص کند و کدی را برای دسترسی مستقیم به مولفه تولید نماید.
- ☑ اگر مبدا مجازی(V.O.) در توصیفگر آرایه ذخیره شده باشد در این صورت آرایه واقعی لازم نیست که توصیفگر پیوسته باشد که در این حالت شکل به صورت زیر می باشد.



* رکوردها (Record):

یک ساختمان‌داده با تعداد مولفه ثابت و نوع مولفه‌های متفاوت می‌باشد.

مشخصات:

1. نوع مولفه‌ها می‌تواند یکسان باشد.
2. مولفه‌ها توسط اسامی سمبلیک (نمادین) مورد مراجعه واقع می‌شوند.
3. از نوع fixed size است.

مثال :

```
Employee: Record
        ID: integer;
        Age: integer;
        Salary: real;
        Dept: Char;
End;
```

در زبان C و Pascal جهت دسترسی به مولفه‌ها ابتدا نام رکورد و سپس نام مولفه که با یک نقطه از هم جدا می‌شوند آورده می‌شود.

```
Employee.ID=10;
Employee.Dept='A';
```

عملیات:

این عملیات شامل عملیات روی هر مولفه می‌باشد که بستگی به نوع آن مولفه دارد و عملیات روی کل ساختار مانند انتساب یک رکورد به کورد دیگر می‌باشد. (از این نظر در رکورد دارای محدودیت هستیم و تنها عمل ممکن همان انتساب می‌باشد. در این زبان‌ها انتساب رکورد را داریم:C و Pascal و Cobol و PL1.)

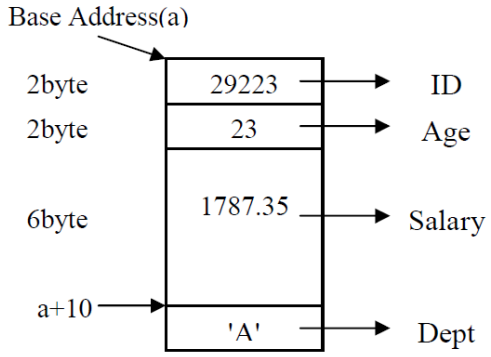
پیاده‌سازی:

از ساختار ترتیبی استفاده می‌شود و معمولا برای آن توصیفگر در نظر گرفته نمی‌شود. مکان هر مولفه در رکورد :

Location R.I=a+ ∑ Size(R.J)

☑ پیاده سازی عملیات روی کل ساختار شامل کپی کردن قسمتی از حافظه روی قسمتی دیگر می‌باشد.

مثال:



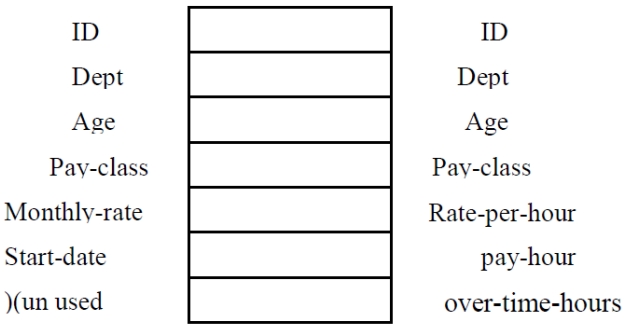
♦ **رکوردهای متغیر:**

منظور از رکوردهای متغیر آنست که معتبر بودن یا نبودن یک مولفه بستگی به ارزش مولفه دیگر داشته باشد. به این مولفه‌های تعیین کننده، مولفه شاخص(Tag) گفته می‌شود.

مثال در Pascal :

```
Type pay-type=(salaried , Hourly);
Var Employee = Record
        ID: integer;
        Dept: char;
        Age: integer;
        Case pay-class: Pay-type of
                Salaried: (Monthly-rate: real; Start-date: integer);
                Hourly: (Rate-per-hour: real; pay-hours: integer; over-time-hours: integer);
```

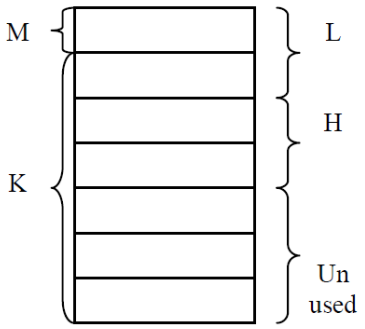
پیاده سازی مثال بالا :



☑ از نظر فضای اشغالی در رکرود متغیر داریم:
ماکزیمم فضای هر یک از بخش‌های متغیر+ فضای اشغالی فیلد شاخص+ فضای اشغالی قسمت ثابت رکورد

مثال:

```
Case pay-class: Pay-type of
                Salaried: (M: char; K: Real);
                Hourly: (L: integer; H: integer);
```



برای وجود هر مولفه دو راه حل زیر وجود دارد:

1. **Dynamic checking**: در زمان اجرا معتبر بودن مولفه از طریق Tag چک می‌شود و در صورت معتبر نبودن runtime error تولید می‌شود.
2. **No checking**: در این حالت تمامی مولفه‌ها صرف‌نظر از Tag مورد مراجعه قرار می‌گیرند مانند زبان‌های C وPascal و Cobol و PL1 .(برنامه‌نویس باید چک کند.)

- **رکوردهای بدون تعریف در زبان‌های برنامه‌نویسی:**

در بعضی از زبان‌ها مانند lisp و snobol4 رکوردها از پیش تعیین نمی‌شوند و ضمناجرای برنامه تولید می-شوند.

Snobol4:

A: Array (10);

A(1)= "xyz";

...

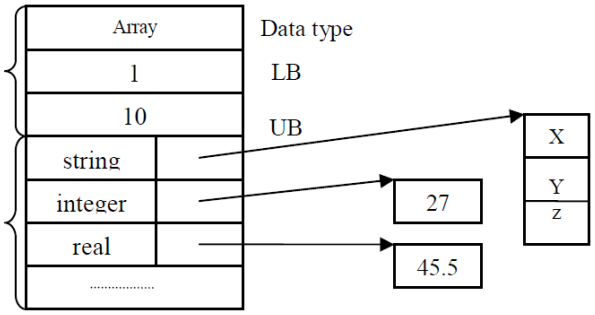
A(2)=27;

…

A(3)=45.5;

توصیفگر

برداری از اشاره‌گرهایی به مولفه‌ها



با توجه به شکل فوق از دید پیاده‌سازی، اشاره‌گرها به نحو موثری برای این هدف بکار گرفته می‌شوند. همچنین در شکل دیده می‌شود که به ازای هر مولفه، نوع مولفه و اشاره‌گری به ارزش مولفه ذخیره می‌شود که معمولا ارزش‌های در بلاک‌های دیگری از حافظه قرار می‌گیرد و توسط روتین‌های مدیریت حافظه بکار گرفته می‌شوند.

* رشته ها Strings

رشته کاراکترها : مجموعه محدودی از کراکترها را رشته گویند .

- رشته با طول ثابت (Fixed length)
- رشته با طول متغیر و محدود به یک حد بالا (Variable length to a declared bound)
- رشته با طول محدود نشده (Unbounded length)

انواع عملیات روی رشته ها

- الحاق رشته ها (Concatenation) ← strcat در C
- عملیات مقایسه ای بین رشته ها ← strcmp در C
- انتخاب زیر رشته با توجه به موقعیت رشته اصلی
- فرمت یا آرایش اطلاعات ورودی و خروجی
- جستجوی یک زیر رشته در رشته اصلی ← strstr در C

پیاده سازی

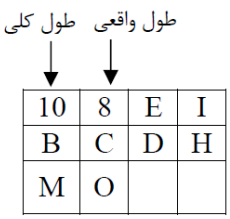
از جهت پیاده سازی چگونگی ذخیره اطلاعات به صورت شکل های زیر می باشد :

R	E	L	A
T	I	V	E
L	Y		

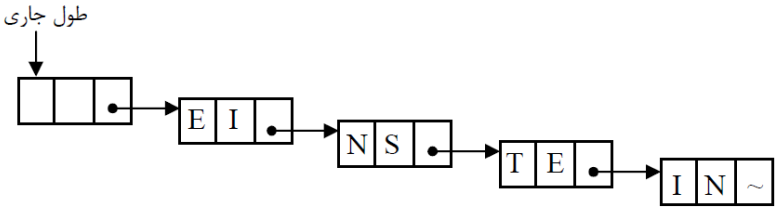
رشته با طول ثابت : برای fixed size ها هر کلمه یا word تعداد مشخصی از کراکترها را پوشش می دهد.

هر کلمه در اینجا 4 بایت است ، بنابراین دو خانه باقیمانده از دست می رود .

رشته با طول متغیر و محدود به یک حد بالا : در این حالت دو خانه خالی باقیمانده می توانند استفاده شوند چون به طول ماکزیمم نرسیده است . مثل پاسکال .



رشته با طول محدود نشده : در این مورد باید از یک لیست پیوندی استفاده نمود . زبان C تقریبا از این حالت استفاده می کند .



از جهت پیاده سازی برای عملیات fixed پشتیبانی سخت افزاری وجود دارد ولی برای انواع دیگر باید از شبیه سازی نرم افزاری استفاده کرد .

❊ ساختمان داده هایی با طول متغیر (VSDS:Variable Size Data Structure)

هر ساختار که تعداد مولفه هایش متغیر باشد و در طول اجرای برنامه به صورت پویا تغییر نماید ، یک ساختمان داده با طول متغیر خواهد بود .

Link List , List structure , Stack , Queue , Tree , Property list , Directed graph

برای لیست ها موارد زیر را داریم :

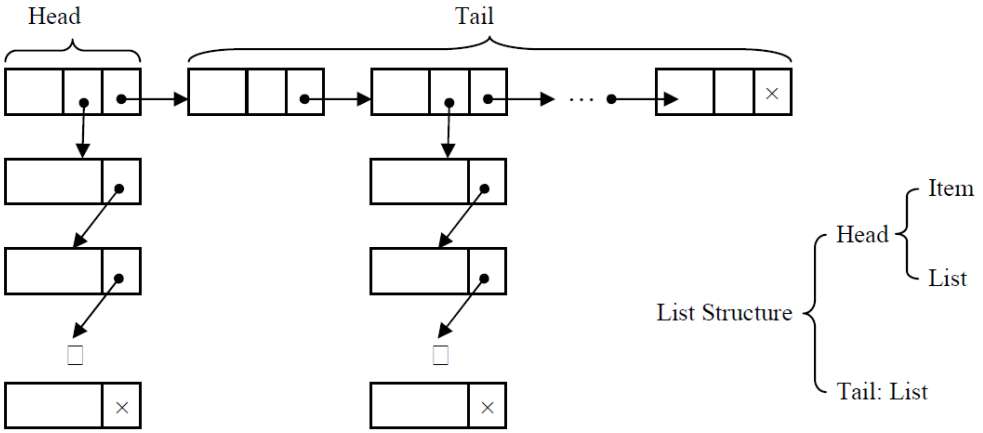
به ندرت طول ثابت دارند

به ندرت همگن هستند

به صورت ضمنی و بدون وجود صفات صریح برای اعضای لیست معمولا اعلان می شوند .

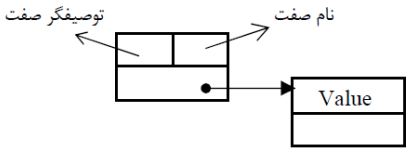
List Structure

اگر مولفه های یک لیست خود لیست دیگری باشند .

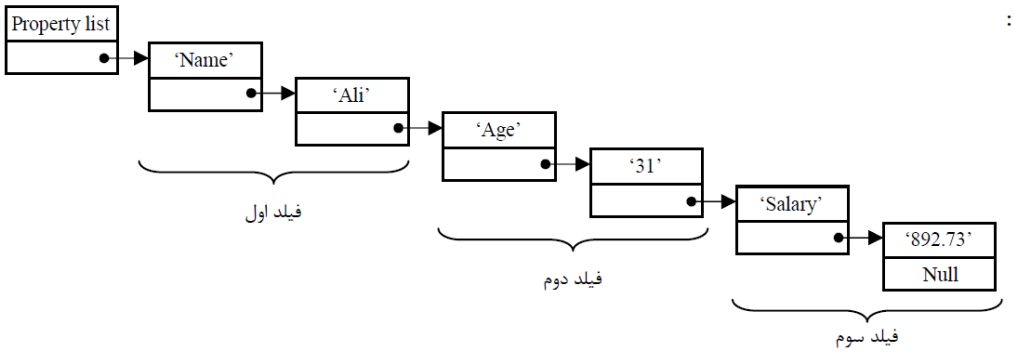


لیست های زبان lisp از این نوع هستند ، تابع head را CAR و Tail را CDR تعریف کرده است .

Property List : مشابه رکوردی است که تعداد مولفه های با تعداد فیلد های آن متغیر است و اسامی مولفه ها یا فیلد ها نیز ذخیره خواهد شد .



مثال :



در زبان lisp به لیست خصوصیت ، جدول گفته می شود و در بعضی جاها با آن لیست صفت/مقدار یا لیست توصیفی نیز می گویند .
جهت تطبیق عملیات روی property list برای یک مولفه search می شود و در صورت تطبیق نام آن صفت با حالت مورد نظر ، ارزش یا مقدار آن فیلد از روی مولفه بعدی استخراج می شود .

از دید زبانهای برنامه سازی در مورد ساختمان داده هایی با طول متغیر دو روش پیاده سازی زیر وجود دارد :
حالت اول : در بعضی زبانها مانند lisp , Ml , prolog ساختارهایی مانند درخت ، صف و ... از قبل پیاده سازی شده اند ؛ یعنی در این زبانها اولیه هستند .
حالت دوم : در بعضی زبانهای دیگر مانند pascal,C,Ada از اشاره گرها به نحو موثری برای پیاده سازی آنها استفاده می شود .

☑ هر ساختاری در زبان lisp یک لیست است .

* اشاره گرها و DO های ایجاد شده توسط برنامه نویس

اشاره گرها جهت لینک کردن DO ها به یکدیگر استفاده می شوند بنابراین یک زبان برنامه سازی از نظر اشاره گرها باید دارای قابلیت های زیر باشند :

1- یک EDT از نوع اشاره گر داشته باشد . (C: int *x ;)

2- عملیات ایجاد DO و تخصیص حافظه و آزادسازی آن را داشته باشد . (C: malloc , free)

3- عملیات روی ارزش یا مقدار اشاره گرها را داشته باشد . (C : & variable, *pointer)

☑ اشیا داده اشاره گر زمینه ای را برای تولید زباله و ارجاعات معلق فراهم می کنند .

در مورد ویژگی اول : دو حالت وجود دارد :

1- در بعضی از زبانها اشاره گر تنها به یک نوع داده می تواند اشاره کند . مثل C,Pascal,Ada

2- در بعضی از زبانها مثل snobol4, smalltalk یک اشاره گر می تواند به هر نوع اطلاعاتی اشاره کند . در این زبانها اشیا داده در طول اجرا دارای توصیفگرهای نوع هستند و بررسی پویا را انجام می دهند.

در مورد ویژگی دوم : یک زبان برنامه سازی باید دستوراتی را در مورد تخصیص حافظه از فضای heap داشته باشد . مثلا در دستورات new,dispose در زبان pascal,Ada .

در مورد ویژگی سوم : عملیات اضافه کردن و یا کم کردن را می توان روی اشاره گرها انجام داد که دو دسته اند :

1- روی داده

2- روی آدرس

☑ در عملیات اضافه کردن و یا کم کردن از یک اشاره گر (از آدرس آن) به اندازه طول نوع آن اشاره گر در هر مرحله به آن اضافه و یا از آن کم می شود . (به ازای x--/x++)

انواع روش های آدرس دهی

از جهت پیاده سازی دو روش وجود دارد :

1- آدرس مطلق Absolute Address

2- آدرس نسبی Relative Address

در روش اول ، مکان قرار گرفتن DO ها از قبل باید مشخص باشد و مکان خاصی برای آن در نظر گرفته شده باشد .

در روش دوم ، DO در هر مکانی از حافظه می تواند ذخیره شود و در واقع توسط offset و base address آدرس هر DO در زمان اجرا مشخص می شود .

✓ مزیت

روش اول : سرعت و کارایی بالا

روش دوم : برنامه ها در حافظه قابل جابجایی است . عملیات recovery نسبتا راحتتر است .

× عیب

روش اول : برنامه ها در حافظه قابل جابجایی نیستند . عملیات recovery (حل مشکل زباله و رجوع رها شده) در آن به سختی انجام می شود .

روش دوم : سرعت و کارایی پایین

* مجموعه ها Sets

یک مجموعه DO ای است که شامل تعداد نامنظمی از ارزشهای مشخص و مجزا می باشد . پس :

- عناصر ترتیب خاصی ندارند .
- عناصر تکراری وجود ندارند .

☑ در مقابل برای list می توان گفت که اولا ترتیب عناصر مهم است و ثانیا عنصر تکراری ممکن است وجود داشته باشد .

عملیات :

1- تست عضویت عناصر (مثل عملگر in در پاسکال)
2- وارد کردن و یا حذف کردن عناصر از یک مجموعه
3- اجتماع ، اشتراک و تفاضل دو مجموعه

روشهای پیاده سازی

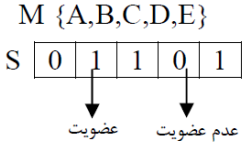
برای پیاده سازی مجموعه ها دو روش وجود دارد :

1- ***روش نقشه بیتی :*** معرفی مجموعه به صورت رشته بیتی (bit strim/bit pattern) . در این روش اگر مجموعه ای شامل اعضای $e_1,e_2,\ldots,e_n$ باشد ، می توان یک رشته بیتی به طول n بیت در نظر گرفت که در آن به اعضای هر عنصر مجموعه یک بیت در نظر گرفته شده است .(با همان ترتیب معرفی مجموعه) به ازای هر عنصر اگر بیت مربوط به آن صفر باشد ؛ یعنی آن عنصر وجود ندارد و اگر یک باشد یعنی آن عنصر در آن مجموعه وجود دارد .

- برای انجام تست عضویت باید یک یا صفر بودن بیت متناظر با آن عنصر را چک کنیم
- برای وارد کردن عضو جدید در مجموعه ، بیت متناظر با آن عضو باید یک شود و برای حذف کردن ، صفر .
- انجام عملیات اجتماع و اشتراک و تفاضل به صورت OR کردن ، AND کردن و AND کردن نقیض مجموعه دوم با مجموعه اول است (A AND NOT(B)). همه این عملگرها بیتی هستند یعنی کاملا کارایی دارند .

☑ اکثر زبانهای برنامه سازی از این روش استفاده می کنند .

مثال :



2- ***روش جدول درهم سازی :***

معرفی یک مجموعه به صورت کد درهم سازی (Hash code) . در این روش از یک جدول به نام Hash table برای ذخیره سازی عناصر و از یک تابع به نام Hash Function برای به دست آوردن آدرس ذخیره سازی هر عنصر در Hash Table استفاده می شود .

روش HashCode برای مجموعه ها به این صورت کار می کند که ابتدا بر اساس نمایش حافظه هر عضو از مجموعه یک آدرس برای ذخیره آن عنصر در حافظه Hash Table توسط تابع Hash به دست می آید و سپس آن عنصر در آن خانه ذخیره می شود . فقط قبل از ذخیره باید توجه کرد آیا عنصری قبلا در آن ذخیره شده است یا خیر ؟

مشکل اصلی این الگوریتم این است که حتی بهترین توابع درهم سازی نیز در حالت کلی نمی تواند تضمین کند برای اقلام داده متفاوت آدرس های درهم مجزایی تولید شود .

روشهای رفع برخورد :

- **Rehashing :** وقتی می خواهد برخورد پیش بیاید ، توسط تابع دومی دوباره Hash می کند .
- **خطی :** اگر برخورد پیش می آید ، از آن خانه به دنبال اولین خانه خالی می گردد و عنصر در اولین خانه خالی بعدی ذخیره گردد.
- **پیوندی :** در هر خانه یک اشاره گر است که این اشاره گر به لیست پیوندی عناصر هم آدرس اشاره می کند .

در روش درهم سازی تست عضویت و درج و حذف عناصر از آن با سرعت بالا و به طور کارآمد انجام می شود ولی اعمال اشتراک ، اجتماع و تفاضل را به صورت کارآمدی انجام نمی دهد .

مشکل دیگر آن ، اتلاف حافظه است که معمولا دو برابر اعضا ، حافظه در نظر گرفته می شود .

*** فایل و عملیات ورودی/خروجی (File & IO Operation)**

یک فایل DO ای است که دو خصوصیت زیر را دارد :

1- معمولا در حافظه ثانویه (مثل دیسک) می باشد .
2- Life time یک فایل زیاد می باشد .

انواع فایل

به طور کلی فایل به سه دسته زیر تقسیم می شود :

1- Sequential File : که مولفه ها به صورت ترتیبی به دنبال یکدیگر ذخیره شده اند و به صورت ترتیبی نیز قابل دسترس هستند .
2- Direct Access File :که مولفه ها به صورت تصادفی و مستقیم قابل بازیابی هستند .

3- Indexed Sequential File : فایلهایی که مولفه های آن هم به صورت ترتیبی و هم به صورت تصادفی (به واسطه Index) قابل دسترس هستند .

نکات مهم در مورد فایل ترتیبی شامل موارد زیر است :

1- یک S.F شامل مجموعه ای از ترکیب خطی DO ها با مولفه های مشابه است .
2- تعداد مولفه ها متغیر است و تحت هیچ شرایطی نمی توان عدد ثابتی برای آن تعیین کرد .
3- مولفه ها می توانند از نوع EDT یا SDT باشند .
4- یک فایل شامل عملیات Input و Output به صورت کراکتری است .
5- یک text file شامل مجموهخ ای از خطوط (Line) می باشد .
6- یک فایل در دو مد Read و Write عمل می کند .
7- یک FilePointer برای هر فایل وجود دارد که موقعیت جاری فایل را نشان می دهد . این F.P می تواند قبل از مولفه اول ، بین دو مولفه و بعد از مولفه آخر قرار گیرد .

پیاده سازی فایل

از جهت پیاده سازی مسئول انجام همه عملیات مربوط به فایل ها و مدیریت آنها به عهده سیستم عامل می باشد . یک زبان برنامه سازی باید مکانیزیم هایی جهت برقراری ارتباط با برنامه نویس و سیستم عامل داشته باشد .

از دید جزئی تر وقتی یک فایل باز می شود . فضایی به نام FIT (File Information Table) و همچنین فضایی به عنوان بافر برای هر فایل تخصیص می یابد .

FIT : در این جدول نام فایل ، تاریخ آخرین به روز رسانی و اندازه فایل و مشخصات دیگر از جمله محل های خواندن و نوشتن روی دستگاه ذخیره و و بازیابی اطلاعات نگه داشته می شود .

Buffer : این حافظه به عنوان یک صف برای خواندن و نوشتن عمل می کند . در صورت پر یا خالی شدن بافر ، اطلاعات به دستگاه خارجی انتقال داده می شود و یا از آن به داخل بافر منتقل می شود.