اصول طراحی کامپایلر

درس‌نامه‌ی ۱۶

کاظم فولادی
http://kazim.fouladi.ir

ویراست اول: ۱۳۸۸
ویراست دوم: ۱۳۹۳

فهرست مطالب

۱۶ سازمان حافظه‌ی زمان اجرا

RUN-TIME STORAGE ORGANIZATION

۱۶-۱ مقدمه

- در طول اجرای یک برنامه، یک نام واحد درکد منبع می‌تواند به اشیای داده‌ای متفاوتی درکامپیوتر اشاره کند.
- تخصیص و آزادسازی اشیای داده‌ای، توسط **ماژول پشتیبانی زمان اجرا** مدیریت می‌شود.

۱۶-۱-۱ محیط، حالت و انقیاد

محیط (Environment): نگاشت یک نام به یک فضای حافظه

$$\text{name} \rightarrow \text{storage space}$$

حالت (State): مقدار فعلی یک فضای حافظه

$$\text{storage space} \rightarrow \text{state}$$

انقیاد (Binding): وابسته کردن یک نام به یک مکان حافظه

$$\text{name} \rightarrow \text{storage location}$$

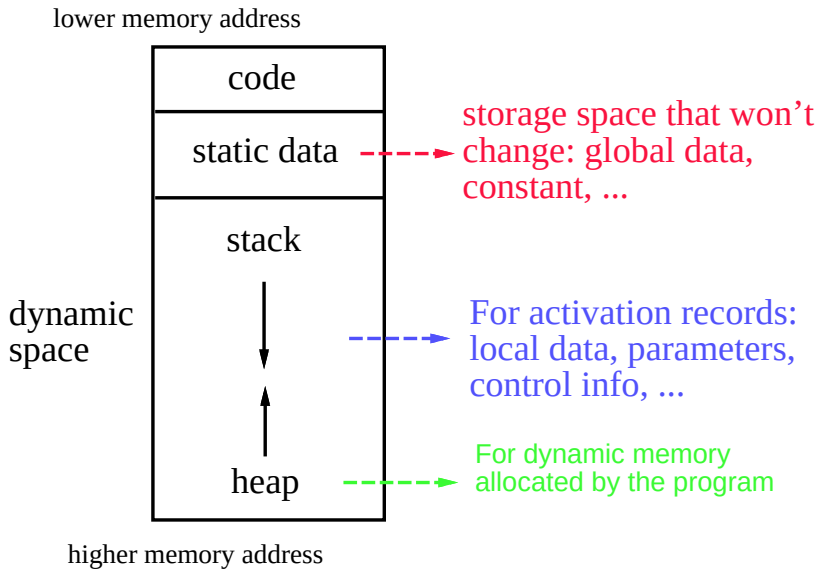
۱۶-۱-۲ فعالیت

هر اجرای یک روال یک **فعالیت (Activation)** نام دارد.

- اگر یک روال، بازگشتی باشد، در این صورت تعداد زیادی از فعالیت آن می‌تواند در یک زمان موجود باشد.
- یک روال می‌تواند خود را به صورت غیرمستقیم فراخوانی کند (بازگشتی غیر مستقیم).

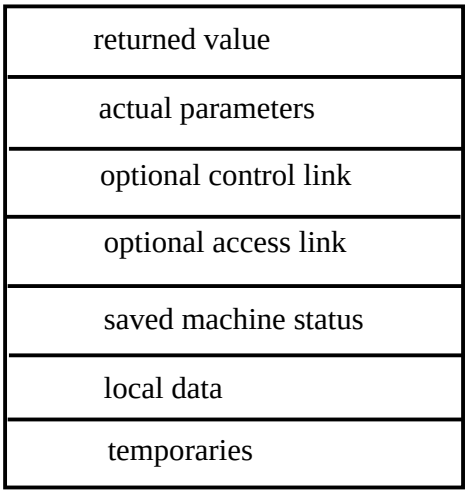
طول عمر (life-time): طول بازه‌ی زمانی میان نخستین گام و آخرین گام در یک روال است.

۱۶-۱-۳ لی‌اوت فضای حافظه در زمان اجرا



۱۶-۱-۴ رکورد فعالیت

رکورد فعالیت (Activation record) داده‌های مربوط به یک اجرای یک روال را نگهداری می‌کند.



- **پارامترها (parameters)**:
  - **پارامترهای صوری (formal)**: اعلان پارامترها
  - **پارامترهای واقعی (actual)**: مقدار پارامترها برای این فعال‌سازی
- **پیوندها (links)**:
  - **پیوندهای دسترسی (ایستا) (access link)**: اشاره‌گری به داده‌های غیرمحلی
  - **پیوندهای کنترلی (پویا) (dynamic)**: اشاره‌گری به رکورد فعالیت روال فراخواننده

۲-۱۶ رویکردهای تخصیص حافظه (Memory Allocation)

- تخصیص حافظه‌ی ایستا (Static)
- تخصیص حافظه‌ی پویا (Dynamic)

۱-۲-۱۶ تخصیص حافظه‌ی ایستا

در تخصیص حافظه‌ی ایستا از پشته و هیپ استفاده نمی‌شود.

- رکورد فعالیت در ناحیه‌ی داده‌های ایستا قرار دارد : برای هر روال یک رکورد در نظر گرفته می‌شود.
- نام‌ها در زمان کامپایل به مکان‌های حافظه مقید می‌شوند.
- هرگاه یک روال فراخوانی شد، نام آن به فضای از پیش تعیین شده در حافظه منتسب می‌شود.

معایب:

- امکان بازگشتی بودن روال‌ها وجود ندارد.
- حجم زیادی از حافظه در هنگام فعال نبودن روال هدر می‌رود.
- تخصیص پویا ممکن نیست.

مزایا:

- چون دستکاری پشته یا دسترسی غیرمستقیم به نام‌ها وجود ندارد، دسترسی به متغیرها سریع‌تر است.
- مقادیر رکورد فعالیت از یک فراخوانی روال تا فراخوانی بعدی باقی می‌مانند (مانند متغیرهای static در زبان C).

فرآیند فراخوانی روال‌ها (در تخصیص حافظه‌ی ایستا)

در روال فراخوانی‌کننده (calling):

- ابتدا آرگومان‌ها ارزیابی می‌شوند.
- آرگومان‌ها در فضای پارامترها در رکورد فعالیت روال فراخوانی‌کننده کپی می‌شوند.[۱]
- ممکن است لازم باشد مقدار برخی از ثبات‌ها در رکورد فعالیت فراخوانی‌کننده ثبت شود.
- **پرش و پیونددهی:** پرش به نخستین دستورالعمل روال فراخوانی‌شونده، انجام می‌شود و آدرس دستورالعمل بعدی (آدرس برگشت) در ثبات RA (ثبات آدرس برگشت) قرار می‌گیرد.

در روال فراخوانی‌شونده (called):

- آدرس برگشت از ثبات AR در فیلد آدرس برگشت رکورد فعالیت آن کپی می‌شود.
- ممکن است لازم باشد مقدار برخی ثبات‌ها ذخیره شود.
- ممکن است لازم باشد داده‌های محلی مقداردهی اولیه شوند.

[۱] قرارداد: آنچه به یک روال پاس می‌شود، از نگاه فراخوانی‌کننده **آرگومان** و از نگاه فراخوانی‌شونده **پارامتر** نامیده می‌شود.

فرآیند برگشت از روال‌ها (در تخصیص حافظه‌ی ایستا)

در روال فراخوانی‌شونده (called):

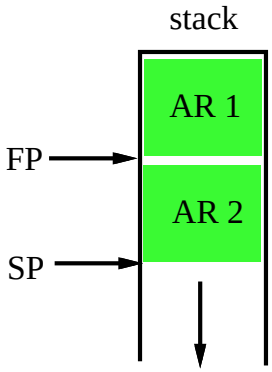
- بازیابی مقادیر ثبات‌های ذخیره شده انجام می شود.
- پرش به آدرس موجود در فیلد آدرس برگشت صورت می‌گیرد.

در روال فراخوانی‌کننده (calling):

- ممکن است لازم باشد مقدار برخی ثبات‌ها بازیابی شود.
- اگر روال فراخوانی‌شونده در واقع یک تابع باشد، مقدار برگشتی در یک مکان مناسب قرار می‌گیرد.

۲-۲-۱۶ تخصیص حافظه‌ی پویا: تخصیص پشته‌ای

- هرگاه یک روال فراخوانی می‌شود، یک رکورد فعالیت جدید در پشته درج می‌شود.
- در هنگام برگشت از روال، رکورد فعالیت از بالای پشته برداشته می‌شود.
- یک ثبات SP (اشاره‌گر پشته) به بالای پشته (آخرین آدرس) اشاره می‌کند.
- یک ثبات FP (اشاره‌گر فریم) به شروع رکورد فعالیت جاری (اولین آدرس) اشاره می‌کند.



فرآیند فراخوانی روال‌ها (در تخصیص حافظه‌ی پویا (پشته‌ای))

در روال فراخوانی‌کننده (calling):

- ممکن است لازم باشد برخی ثبات‌ها در رکورد فعالیت ذخیره شوند.
- ممکن است لازم باشد پیوند دسترسی اختیاری (optional access link) (اشاره‌گر به داده‌های غیرمحلی) تنظیم شود (درج در پشته).
- درج کردن پارامترها در پشته
- **پرش و پیونددهی:** پرش به نخستین دستورالعمل روال فراخوانی‌شونده و قرار دادن آدرس دستورالعمل بعدی (آدرس برگشت) در ثبات RA (ثبات آدرس برگشت).

در روال فراخوانی‌شونده (called):

- درج کردن آدرس برگشت در ثبات RA
- درج کردن FP قدیمی (control link) (اشاره‌گر به رکورد فعالیت روال فراخوانی‌کننده)

- مقداردهی FP جدید با SP قدیمی.
- مقداردهی SP جدید با SP قدیمی + (اندازه‌ی پارامترها) + (اندازه‌ی RA) + (اندازه‌ی FP)
  این اندازه‌ها در زمان کامپایل محاسبه می‌شوند.
- ممکن است لازم باشد مقدار برخی ثبات‌ها ذخیره شود.
- درج کردن داده‌های محلی در پشته

فرآیند برگشت از روال‌ها (در تخصیص حافظه‌ی پویا (پشته‌ای))

در روال فراخوانی‌شونده (called):

- مقادیر ثبات‌های ذخیره شده در صورت نیاز بازیابی می‌شوند.
- آدرس برگشت در ثبات خاص RA بارگذاری می‌شود.
- SP بازیابی می‌شود (SP := FP).
- FP بازیابی می‌شود (FP := saved FP).
- برگشت.

در روال فراخوانی‌کننده (calling):

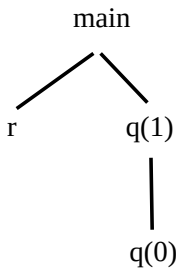
- ممکن است لازم باشد برخی ثبات‌ها بازیابی شوند.
- اگر روال فراخوانی‌شونده در واقع یک تابع باشد، مقدار برگشتی در یک مکان مناسب قرار می‌گیرد.

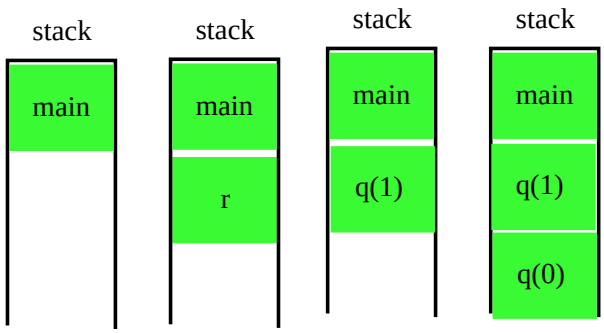
درخت فعالیت از ساختار درخت فعالیت برای ثبت تغییرات رکوردهای فعالیت استفاده می‌شود.

```
main{
    r();
    q(1);
}

r{
...}

q(int i)
{
if(i>0) then q(i-1)
}
```





۳-۲-۱۶ تخصیص حافظه‌ی پویا: تخصیص با هیپ

حافظه‌از سوی برنامه‌نویس در زمان اجرا درخواست شده و تخصیص داده می‌شود.
مانند دستورات new و delete برای حافظه‌ی پویا در زبان C++

مسائل مرتبط:

- جمع‌آوری زباله‌ها (Garbage Collection)
- قطعه‌بندی حافظه (Segmentation)
- ارجاعات سرگردان (Dangling references)

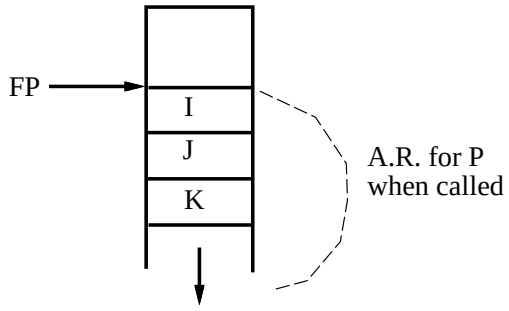
پیاده‌سازی موارد فوق کم و بیش به سیستم عامل وابسته است.

۱۶-۳ دسترسی به متغیرها در زمان اجرا

۱۶-۳-۱ دسترسی به متغیرهای محلی در زمان اجرا

- متغیرهای محلی در رکورد فعالیت روال اعلان‌کننده‌ی آنها ذخیره می‌شود.
- دسترسی به این متغیرها از طریق یک آفست نسبت به اشاره‌گر فریم (FP) انجام می‌شود.

```
P()
{
int I,J,K;
...
}
```



- آدرس J عبارت است از: $FP + ۱ * sizeof(int)$
- آفست برابر است با $۱ * sizeof(int)$
- آدرس واقعی تنها در زمان اجرا قابل تعیین است.
- آفست برای هر متغیر محلی در زمان کامپایل معلوم می‌شود.

۱۶-۳-۲ دسترسی به متغیرهای سراسری و غیرمحلی

متغیرهای سراسری در ناحیه‌ی داده‌های ایستا ذخیره می‌شوند:

- دسترسی به آنها تنها با استفاده از نام آنها انجام می‌شود.
- آدرس‌ها در زمان کامپایل معلوم هستند.

دو قاعده‌ی حوزه‌ی دید برای دسترسی به داده‌های غیرمحلی:

- **حوزه‌ی دید لغوی یا ایستا**
  - مانند زبان‌های پاسکال، C و فرترن
  - آدرس درست یک نام غیرمحلی می‌تواند در زمان کامپایل با بررسی نحوی مشخص شود.
- **حوزه‌ی دید پویا**
  - مانند زبان لیسپ
  - استفاده از یک متغیر غیرمحلی متناظر با اعلان در «**تازه‌ترین روال فراخوانی شده‌ی هم‌اکنون فعال**» است.
  - پاسخ به این سؤال که کدام متغیر غیرمحلی استفاده می‌شود، نمی‌تواند در زمان کامپایل مشخص شود و تنها در زمان اجرا قابل تعیین است.

حوزه‌ی دید لغوی با ساختارهای بلاک‌بندی شده

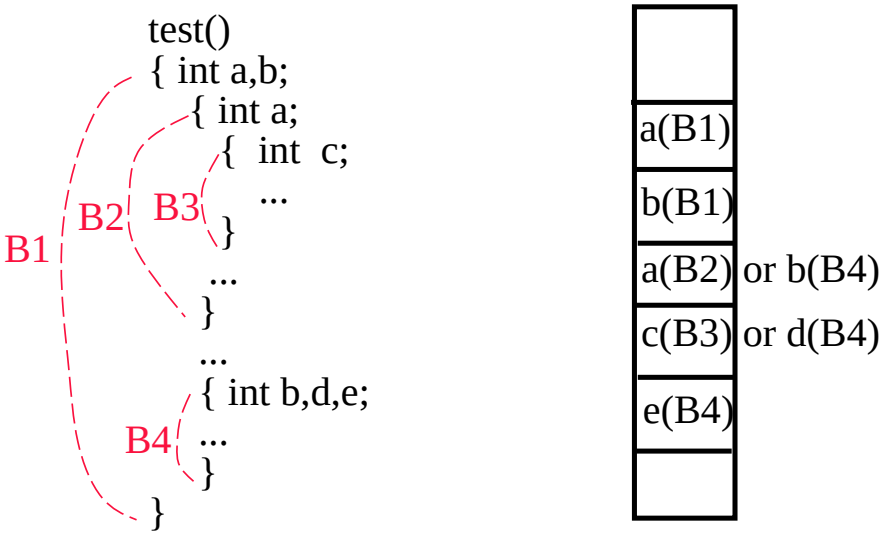
بلاک یک مجموعه دستور است که حاوی اعلان داده‌های محلی خود می‌باشد.
حوزه‌ی دید با قاعده‌ی **نزدیک‌ترین تعریف تودرتو (most closely nested rule)** مشخص می‌شود:

- حوزه‌ی دید یک اعلان در بلاک B، شامل خود B است.
- اگر x در B استفاده شود، اما در B تعریف نشده باشد، در این صورت به x در بلاک B' مراجعه می‌کنیم که در آن
  - B' حاوی اعلان x است.
  - B' نسبت به سایر بلاک‌های حاوی اعلان x، نزدیک‌ترین بلاک پیرامون B است.

حوزه‌ی دید ایستا (لغوی) بدون روال‌های تو در تو

روال تودرتو (nested): روالی که می‌تواند درون روال دیگر اعلان شود.

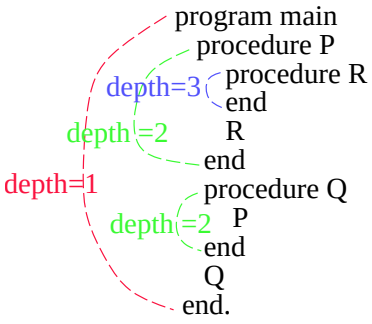
- اگر یک زبان روال‌های تو در تو را مجاز نداند، در این صورت
  - یک متغیر یا سراسری است و یا نسبت به روال حاوی آن محلی است.
  - در زمان اجرا، همه‌ی متغیرهای اعلان‌شده در یک روال (شامل آنهایی که در بلاک‌ها هستند) در رکورد فعالیت ذخیره می‌شوند (امکان همپوشانی وجود دارد).
  - در حین کامپایل، آفست صحیح برای هر داده‌ی محلی با استفاده از اطلاعات معلوم از ساختار بلاکی محاسبه می‌شود.

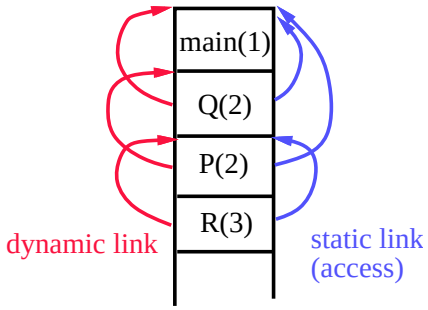


حوزه‌ی دید ایستا (لغوی) با روال‌های تو در تو

- عمق برنامه‌ی اصلی برابر با ۱ است.
- هرگاه وارد یک روال تو در تو می‌شویم، یک واحد به عمق اضافه می‌شود.
- هرگاه از یک روال تو در تو خارج می‌شویم، یک واحد از عمق کم می‌شود.

- هر متغیر به یک عمق تو در تویی نسبت داده می‌شود.
- اگر یک روال با عمق h داشته باشیم و به متغیری در عمق k دسترسی پیدا کنیم، آنگاه:
  - $h \geq k$
  - پیوند دسترسی ایستا (اشاره‌گر به داده‌های غیرمحلی) را $h - k$ مرتبه طی می‌کنیم و سپس از اطلاعات آفست برای یافتن آدرس استفاده می‌کنیم.





الگوریتم تنظیم پیوندها

- پیوند پویا برای اشاره به رکورد فعالیت روال فراخوانی‌کننده تنظیم می‌شود.
- نحوه‌ی تنظیم مناسب پیوند ایستا در زمان کامپایل
  - وقتی روال p در عمق n_p روال x در عمق n_x را فراخوانی می‌کند:
  - اگر $n_p < n_x$ بود، آنگاه x در p محصور می‌شود و $n_p = n_x - 1$. (مشابه تنظیم پیوند پویا)
  - اگر $n_p \geq n_x$ بود، آنگاه یک فراخوانی بازگشتی یا فراخوانی یک روال از پیش اعلان‌شده را خواهیم داشت.

توجه: با یک مرتبه بالا رفتن از پیوند دسترسی، عمق یک واحد کاهش می‌یابد.
بنابراین، پیوند دسترسی x پیوند دسترسی p خواهد بود که $n_p - n_x + 1$ مرتبه از آن بالا رفته‌ایم.

مثال

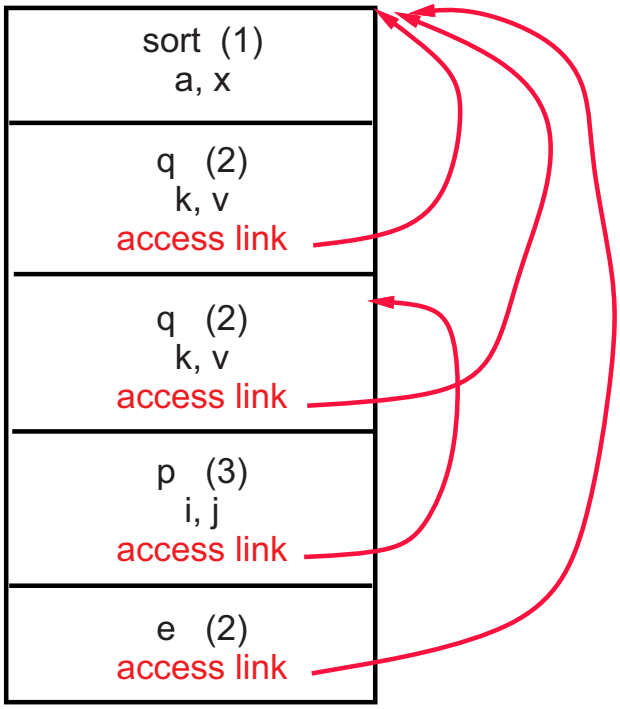
نمونه‌ی اجرای الگوریتم تنظیم پیوندهای ایستا در مثال زیر مشاهده می‌شود:

```
Program sort
   var a: array[0..10] of int;
       x: int;
   procedure r
   var i: int;
   begin ...  r
   end

   procedure e(i,j)
   begin ... e
      a[i] <-> a[j]
   end

   procedure q
      var k,v: int;
      procedure p
      var i,j;
      begin ... p
        call e
      end
   begin ... q
      call q
   end

begin ... sort
   call q
end
```



توجه کنید که پیوند دسترسی (access link) به آخرین روال فراخوانی‌کننده‌ی غیربازگشتی اشاره می‌کند ($n_p - n_x + ۱$ مرتبه به بالا حرکت می‌کنیم). اما برای روال غیربازگشتی، به روال فراخوانی‌کننده‌ی ماقبل (عمق $n_p - ۱$) اشاره می‌کند.

دسترسی به داده‌های غیرمحلی با استفاده از نمایشگر (DISPLAY)

یک آرایه‌ی سراسری که $DISPLAY$ نامیده می‌شود، ایجاد می‌شود:

- با استفاده از ثبات‌ها (در صورت وجود)
- با استفاده از ناحیه‌ی داده‌های ایستا (در صورت عدم وجود ثبات)

وقتی روال P در عمق تودرتویی k فراخوانی می‌شود،

- $DISPLAY[۱]$، ...، $DISPLAY[k-۱]$، حاوی اشاره‌گرهایی به رکورد فعالیت تازه‌ترین $k-۱$ روال‌هایی است که P را دربردارند.
- $DISPLAY[k]$ حاوی اشاره‌گری به رکورد فعالیت P است.
- برای دسترسی به یک متغیر x که در عمق d اعلان شده است، از $DISPLAY[d]$ برای دسترسی به رکورد فعالیت دربردارنده‌ی x استفاده می‌شود، سپس، از آفست آن برای دسترسی به خود x استفاده می‌گردد.
- اندازه‌ی $DISPLAY$ برابر با ماکزیمم عمق تودرتویی روال‌هاست.

این تکنیک برای زبان‌هایی که بازگشت را مجاز می‌دانند، نامناسب است.

وقتی یک روال در عمق تودرتویی k فراخوانی می‌شود،

- مقدار فعلی $DISPLAY$ در فیلد *save-display* رکورد فعالیت جدید ذخیره می‌شود.
- $DISPLAY[k]$ به گونه‌ای تنظیم می‌شود که به رکورد فعالیت جدید اشاره کند (مثلاً به فیلد *save-display* خود).

وقتی روال برمی‌گردد، $DISPLAY[k]$ را با استفاده از مقدار ذخیره شده در فیلد *save-display* بازیابی می‌کند.

مقایسه‌ی DISPLAY و Access link: در استفاده از DISPLAY یا Access link نوعی trade-off وجود دارد.

- Access link به زمان بیشتری برای دسترسی به داده‌های غیرمحلی دارد (در زمان اجرا). بخصوص وقتی که فاصله‌ی عمق داده‌های غیرمحلی از هم زیاد باشد.
- DISPLAY احتمالاً به فضای بیشتری (در زمان اجرا) نیاز دارد.
- کد تولید شده توسط DISPLAY ساده‌تر است.

حوزه‌ی دید پویا

حوزه‌ی دید پویا (Dynamic scoping): استفاده از یک متغیر غیرمحلی که در «تازه‌ترین روال فراخوانی شده و هنوز فعال» اعلان شده است.

- پاسخ به این پرسش که کدام متغیر غیرمحلی باید استفاده شود، نمی‌تواند در زمان کامپایل تعیین شود.
- پاسخ این پرسش در زمان اجرا تعیین می‌شود.
- جدول نمادها در زمان اجرا باید ذخیره شود.
- دو راه برای پیاده‌سازی دسترسی به متغیرهای غیرمحلی در حوزه‌ی دید پویا وجود دارد:
  - دسترسی عمیق (Deep access)
  - دسترسی کم‌عمق (Shallow access)

حوزه‌ی دید پویا: دسترسی عمیق اگر از یک متغیر غیرمحلی استفاده شود، از پیوندهای کنترل استفاده می‌شود تا در پشته به دنبال تازه‌ترین رکورد فعالیتی بگردد که حاوی آن متغیر باشد.

- برای این باید مشخص باشد که چه متغیرهایی در هر رکورد فعالیت ذخیره شده‌اند.
- باید از جدول نمادها در زمان اجرا استفاده شود.

مثال

برای درک حوزه‌ی دید پویا با دسترسی عمیق، کد زیر را در نظر می‌گیریم:

```
program main
   procedure test
   var x : int;
   begin
       x := 30
       call DeclaresX
       call UsesX
   end
   procedure DeclaresX
      var x: int;
   begin
      x := 100
      call UsesX
   end
   procedure UsesX
   begin
     write(x)
   end
begin
   call test
end
```

- کدام x در روال UsesX استفاده می‌شود؟
- اگر از حوزه‌ی دید ایستا استفاده شود، این یک دستور مجاز نخواهد بود؛ زیرا هیچ حوزه‌ی احاطه‌کننده‌ای x را اعلان نکرده است.

▲

حوزه‌ی دید ایستا: دسترسی کم‌عمق

- لیستی از متغیرهای جاری نگهداری می‌شود.
- به هر نام متغیر موجود در برنامه، یک فضا تخصیص داده می‌شود (در ثبات‌ها یا ناحیه‌ی داده‌های ایستا). یعنی یک فضا به ازای هر x حتی اگر چندین اعلان مختلف از x وجود داشته باشد.
- برای هر ارجاع به x کد تولیدشده به یک مکان مراجعه می‌کند.

هرگاه یک روال فراخوانی می‌شود،

- مقادیر فعلی همه‌ی متغیرهایی که توسط آن روال اعلان شده است، در رکورد فعالیت آن روال ذخیره می‌شود.
  (یعنی، اگر روال x و y را اعلان کرده باشد، آن‌گاه مقادیر x و y که هم‌اکنون در فضای تخصیص‌یافته به آنها قرار گرفته است، ذخیره می‌شود.)
- وقتی کار روال تمام شد، آن مقادیر بازیابی خواهند شد.

در مقایسه با دسترسی عمیق: امکان دسترسی سریع به داده‌های غیرمحلی فراهم می‌شود، اما متناسب با تعداد متغیرهای محلی، در ورود به روال‌ها و خروج از روال‌ها سربار وجود خواهد داشت.

۱۶-۴ گذر دادن پارامترها

در اعلان یک روال، **پارامترها**، **پارامترهای صوری (formal)**
در فراخوانی روال، **آرگومان‌ها**، **پارامترهای واقعی (actual)**

۱۶-۴-۱ مقادیر یک متغیر

- **R-value**: مقدار فعلی متغیر (مقدار سمت راست عمل انتساب)
- **L-value**: مکان/آدرس متغیر (مقدار سمت چپ عمل انتساب)

$$x := y$$

۱۶-۴-۲ روش‌های گذر دادن پارامترها

۱) فراخوانی با مقدار (call by value)

۲) فراخوانی با مرجع (call by reference)

۳) فراخوانی با مقدار-نتیجه (call by valye-result)

۴) فراخوانی با نام (call by name)