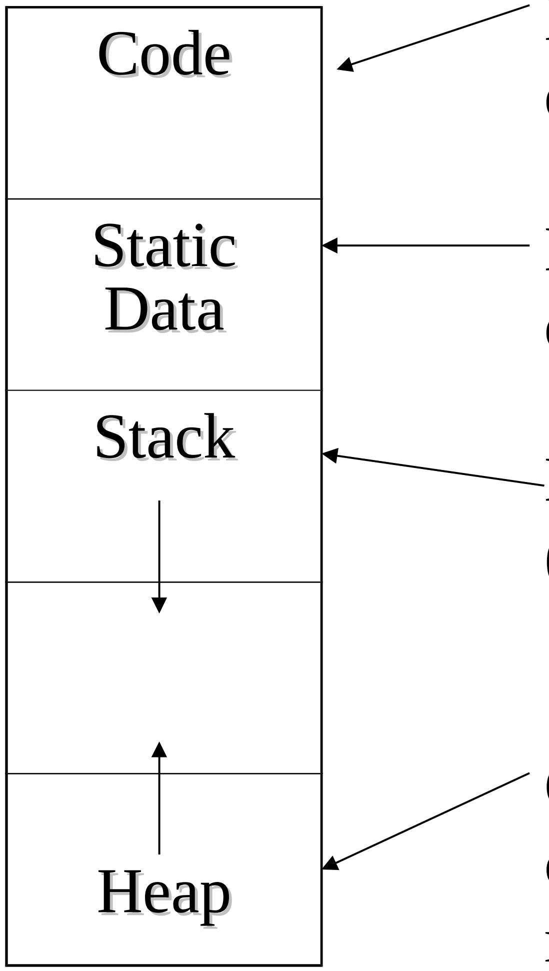
Run-Time Storage Organization



Memory locations for code are determined at compile time.

Locations of static data can also be determined at compile time.

Data objects allocated at run-time. (Activation Records)

Other dynamically allocated data objects at run-time. (For example, malloc area in C).

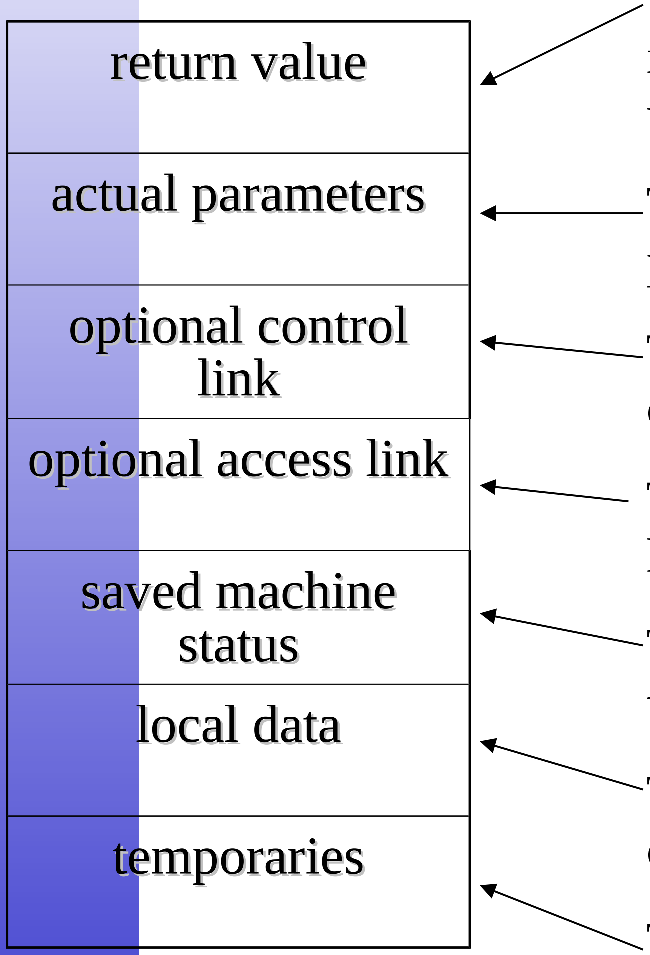
Procedure Activations

- Each execution of a procedure is called as its *activation*.
- *Lifetime* of an activation of a procedure is the sequence of the steps between the first and the last steps in the execution of that procedure (including the other procedures called by that procedure).
- If a and b are procedure activations, then their lifetimes are either non-overlapping or are nested.
- If a procedure is recursive, a new activation can begin before an earlier activation of the same procedure has ended.

Activation Records

- Information needed by a single execution of a procedure is managed using a contiguous block of storage called **activation record**.
- An activation record is allocated when a procedure is entered, and it is de-allocated when that procedure exited.
- Size of each activation record can be determined at compile time (Although the actual location of the an activation record determined at run-time).
  - Even though the procedure has a local variable and its size depends on a parameter, its size is determined at the run time.

Activation Records (cont.)



The returned value of the called procedure is returned in this field to the calling procedure. In practice, we may use a machine register for the return value.

The field for actual parameters is used by the calling procedure to supply parameters to the called procedure.

The optional control link points to the activation record of the caller.

The optional access link is used to refer to nonlocal data held in other activation records.

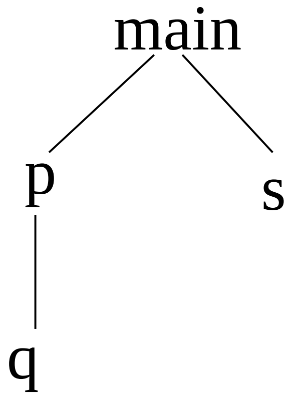
The field for saved machine status holds information about the state of the machine before the procedure is called.

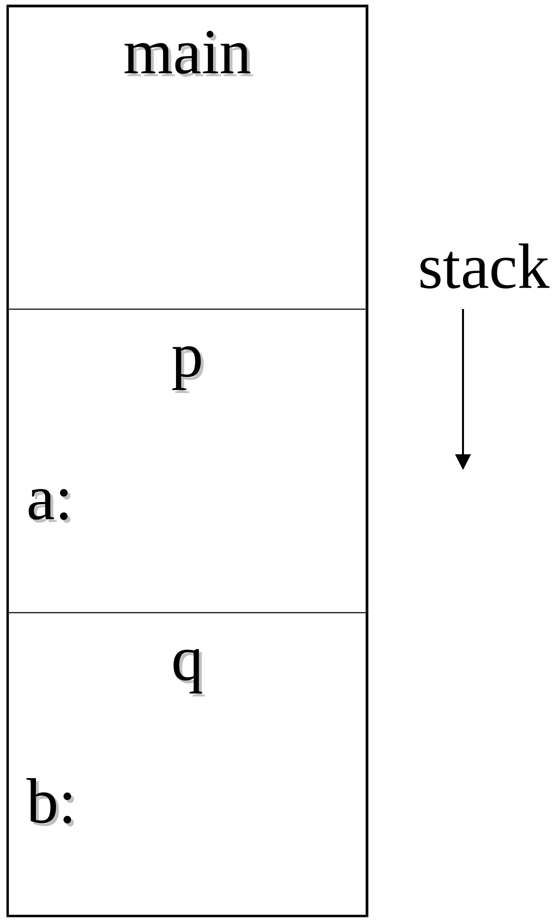
The field of local data holds data that local to an execution of a procedure..

Temporary variables is stored in the field of temporaries.

Activation Records (Ex1)

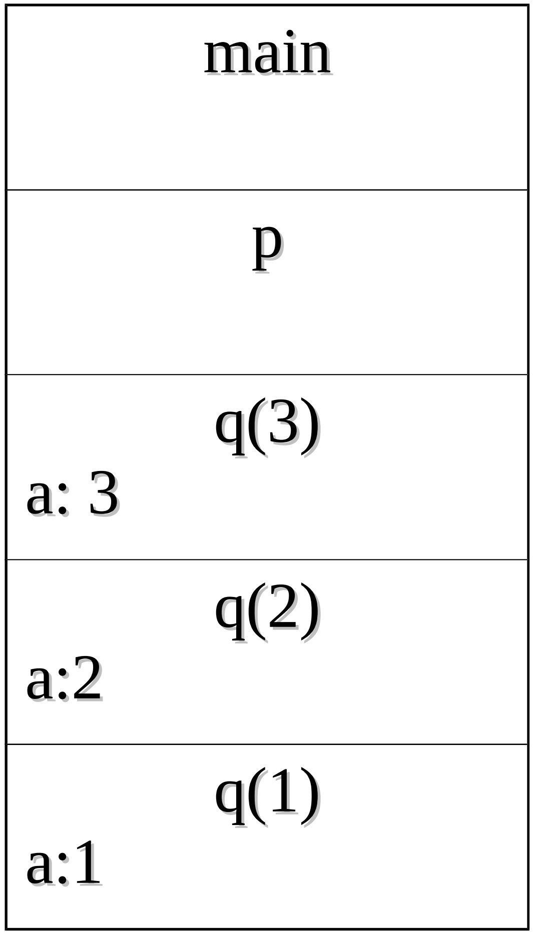
```
program main;
  procedure p;
    var a:real;
    procedure q;
      var b:integer;
      …
    end q;
    … q;
  end p;
  procedure s;
    var c:integer;
    …
  end s;
   … p; s;
end main;
```





Activation Records for Recursive Procedures

```
program main;
  procedure p;
    function q(a:integer):integer;
      begin
        if (a=1) then q:=1;
        else q:=a+q(a-1);
      end q;
    begin q(3); end p;
  begin p; end main;
```

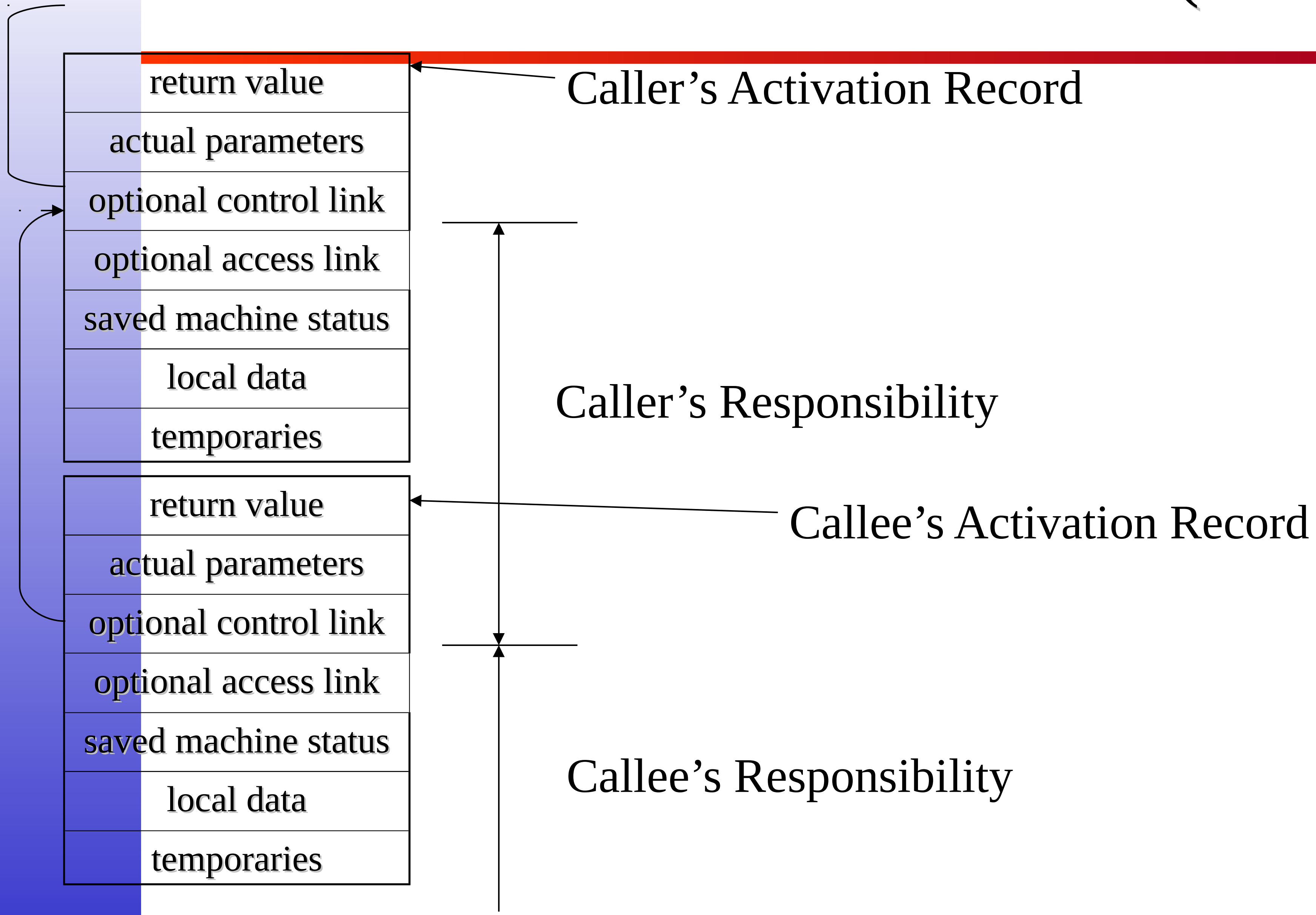


Creation of An Activation Record

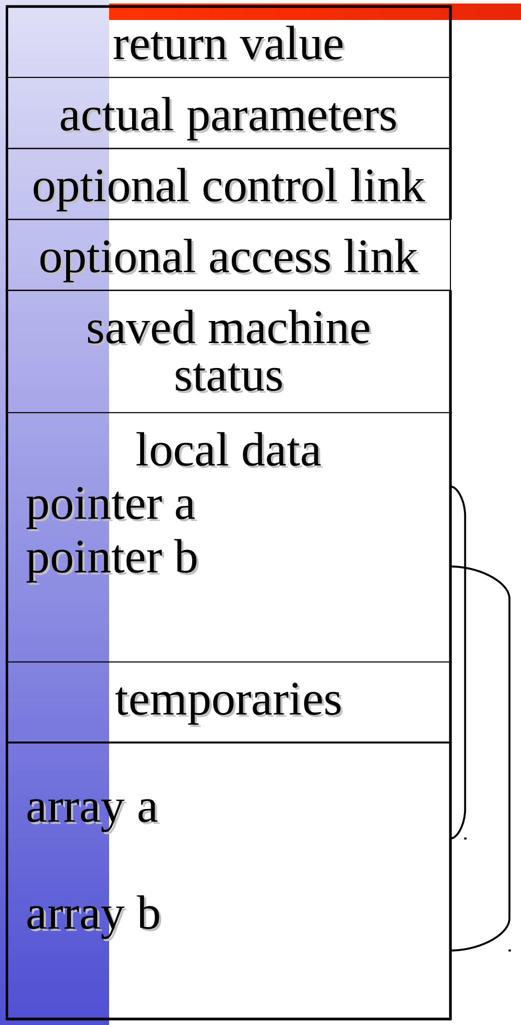
- Who allocates an activation record of a procedure?
  - Some part of the activation record of a procedure is created by that procedure immediately after that procedure is entered.
  - Some part is created by the caller of that procedure before that procedure is entered.

- Who deallocates?
  - Callee de-allocates the part allocated by Callee.
  - Caller de-allocates the part allocated by Caller.

Creation of An Activation Record (cont.)



Variable Length Data



Variable length data is allocated after temporaries, and there is a link to from local data to that array.

Access to Nonlocal Names

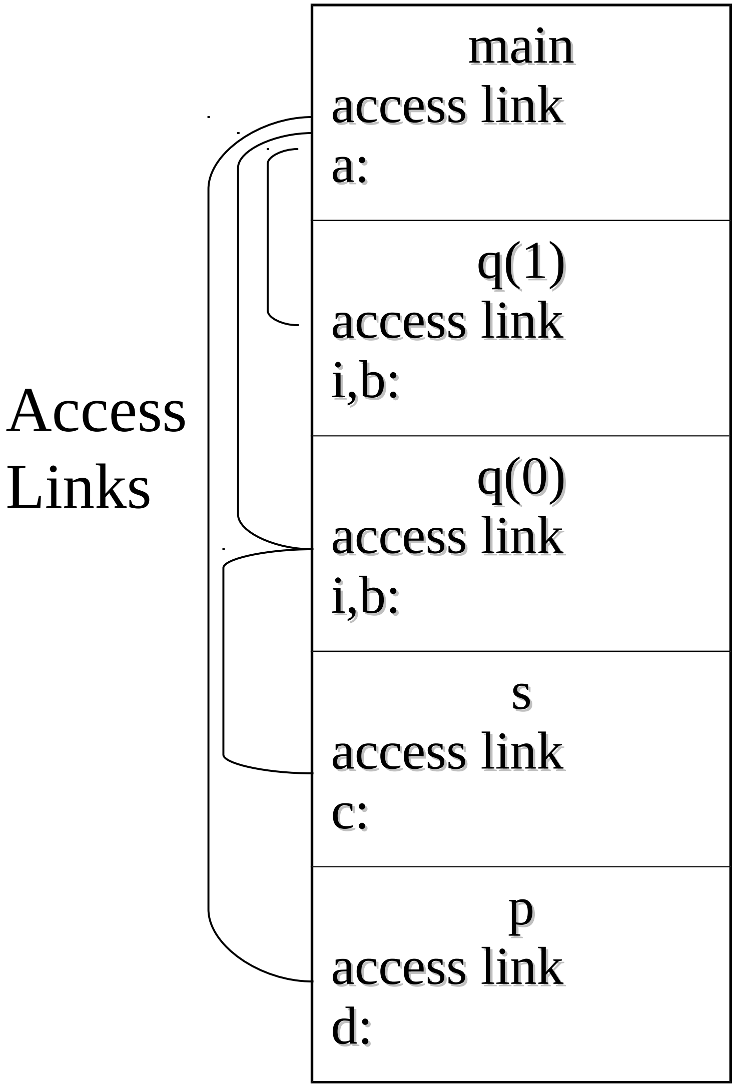
- Scope rules of a language determine the treatment of references to nonlocal names.
- Scope Rules:
  - **Lexical Scope (Static Scope)**
    - Determines the declaration that applies to a name by examining the program text alone at compile-time.
    - Most-closely nested rule is used.
    - Pascal, C, ..
  - **Dynamic Scope**
    - Determines the declaration that applies to a name at run-time.
    - Lisp, APL, ...

Non Local Names

- A procedure may access to a nonlocal name using:
  - access links in activation records, or
  - displays (an efficient way to access to nonlocal names)

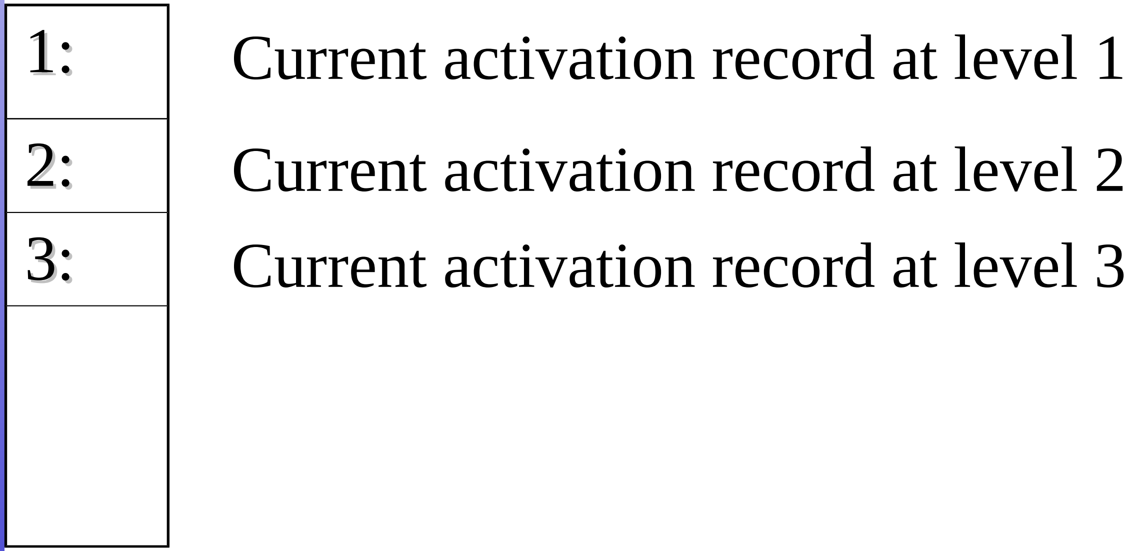
Access Links

```
program main;
  var a:int;
  procedure p;
    var d:int;
    a:=1;
  end p;
  procedure q(i:int);
    var b:int;
    procedure s;
      var c:int;
      call p;
    end s;
    if (i<>0) then q(i-1)
    else s;
  end q;
  q(1);
end main;
```



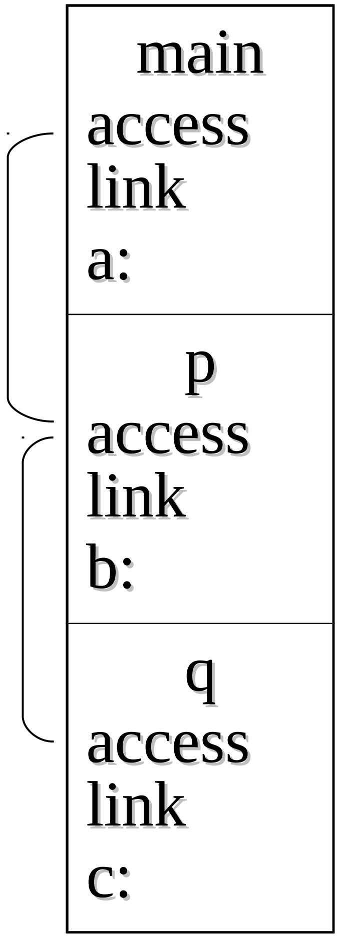
Displays

- An array of pointers to activation records can be used to access activation records.
- This array is called as displays.
- For each level, there will be an array entry.


1:
2:
3:
Current activation record at level 1
Current activation record at level 2
Current activation record at level 3

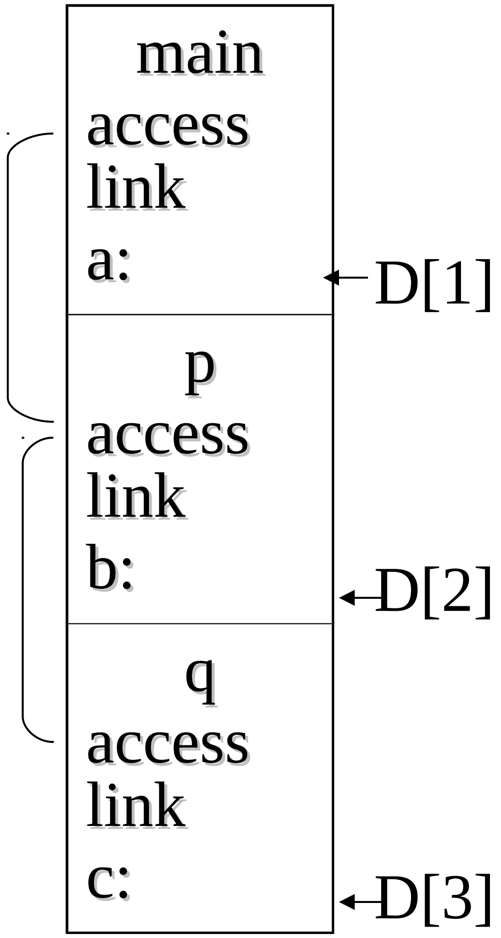
Accessing Nonlocal Variables

```
program main;
  var a:int;
  procedure p;
    var b:int;
    call q;
    procedure q();
      var c:int;
      c:=a+b;
    end q;
  end p;
  call p;
end main;
```



Accessing Nonlocal Variables using Display

```
program main;
  var a:int;
  procedure p;
    var b:int;
    call q;
    procedure q();
      var c:int;
      c:=a+b;
    end q;
  end p;
  call p;
end main;
```



main
access link
a:
D[1]
p
access link
b:
D[2]
q
access link
c:
D[3]