AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Przydział pamięci

Teoria kompilacji

Dr inż. Janusz Majewski
Katedra Informatyki

Terminologia

```
Program s(input,output) {SORT}
  var a: array[0..10] of integer;
  procedure r; {READ ARRAY}
      var i: integer;
      Begin for i := 1 to 9 do read(a[i]); end;
  Function p(y,z :integer):integer; {PARTITION}
  Var i,j,x,v : integer;
  Begin … end;
  Procedure q(m,n:integer);
     Var i  integer;
     Begin
          If (n>m) then
               Begin
                    i := p(m,n);
                    q(m,i - 1)
                    q(i + 1,n);
               end;
     end;
Begin
     a[0] := -9999;    a[10] := 9999;
     r;
     q(1,9);
end.
```

program główny można traktować także jako procedurę

deklaracja zmiennych globalnych

y,z – parametry formalne

deklaracje zmiennych lokalnych

{QUICKSORT}

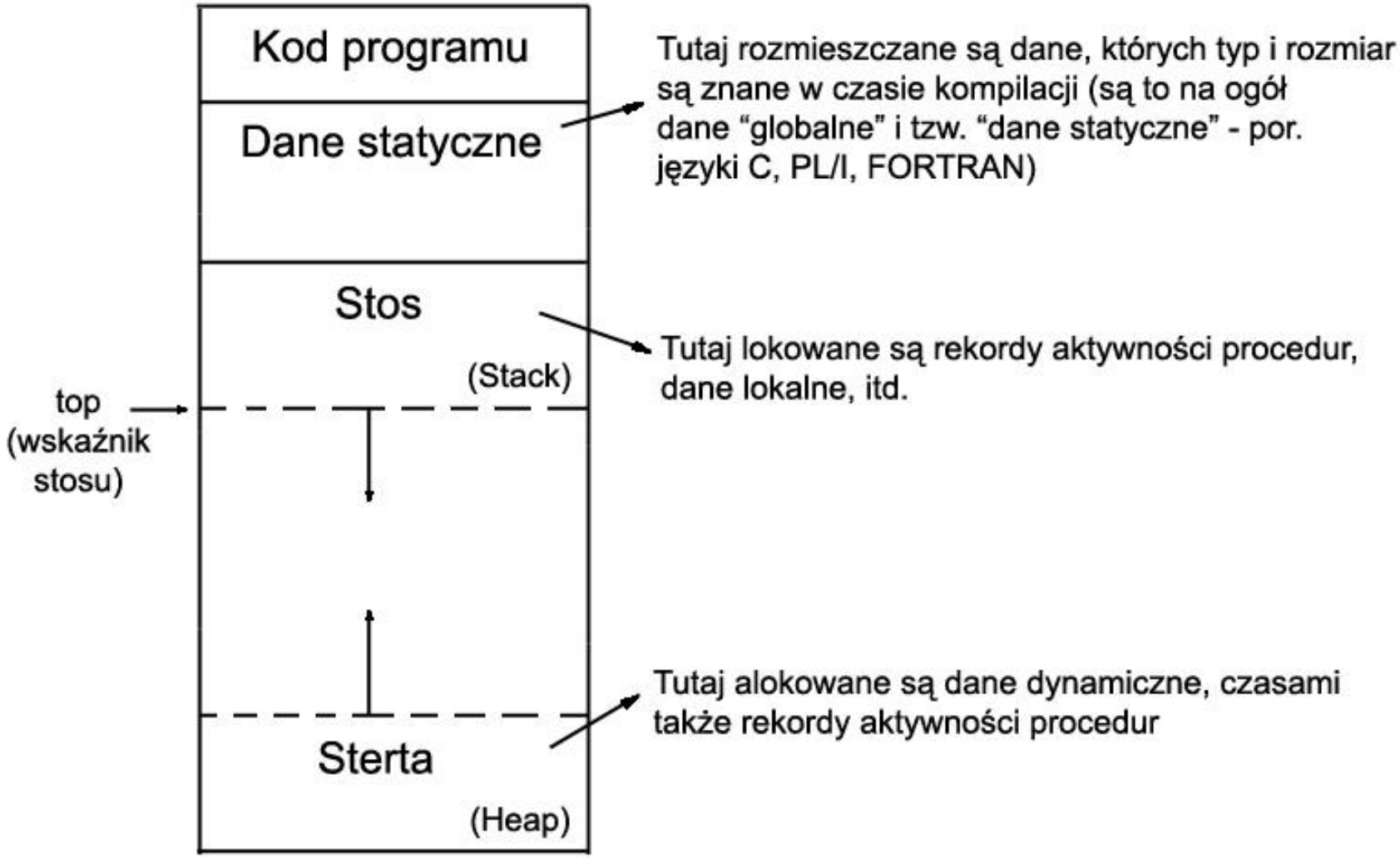
m,n – parametry aktualne

wywołanie rekursywne

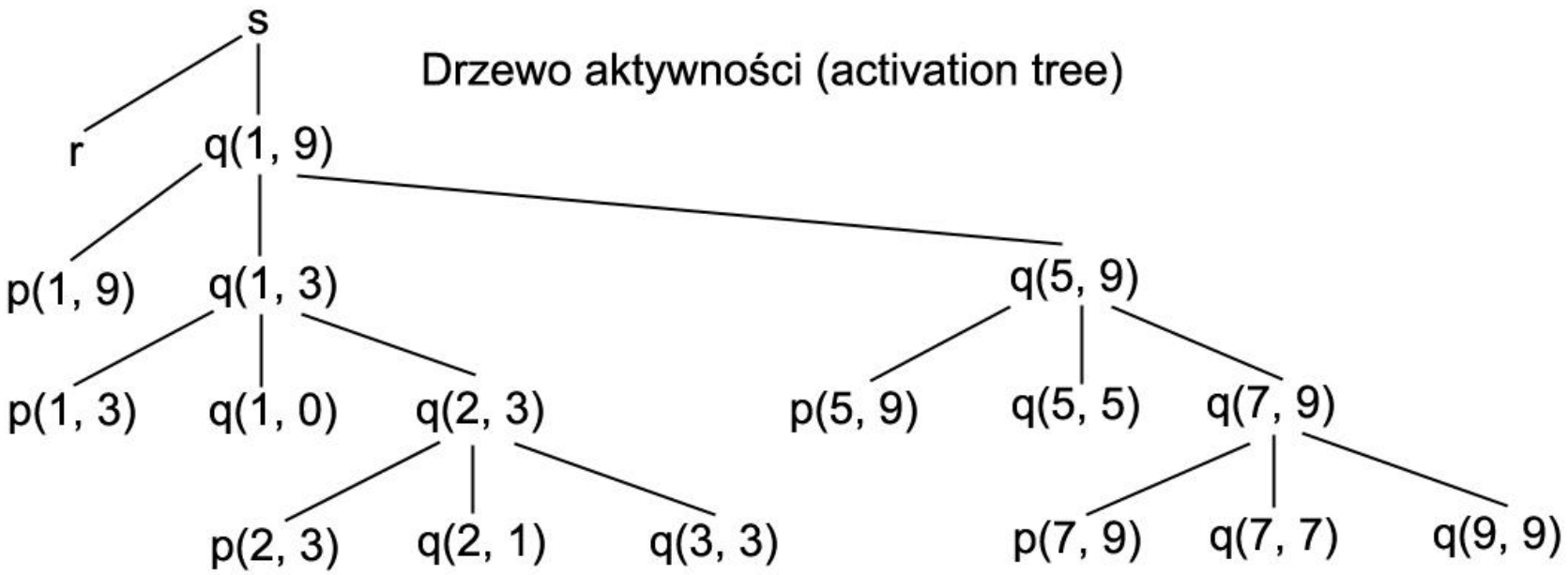
Pytania projektanta kompilatora

1. Czy procedury mogą być rekursywne?
2. Co się dzieje z wartościami zmiennych lokalnych w momencie, gdy sterowanie opuszcza aktywną w danym momencie procedurę?
3. Czy procedura może odwoływać się do nazw nielokalnych?
4. Jak są przekazywane parametry do procedury?
5. Czy procedury mogą być parametrami wywołań?
6. Czy procedury mogą zwracać wyniki (czy mogą być funkcjami)?
7. Czy wynikiem działania procedury może być procedura?
8. Czy pamięć może być dynamicznie przydzielana pod kontrolą programu?
9. Czy pamięć musi być jawnie zwalniania?

Typowy podział pamięci dla programu użytkowego

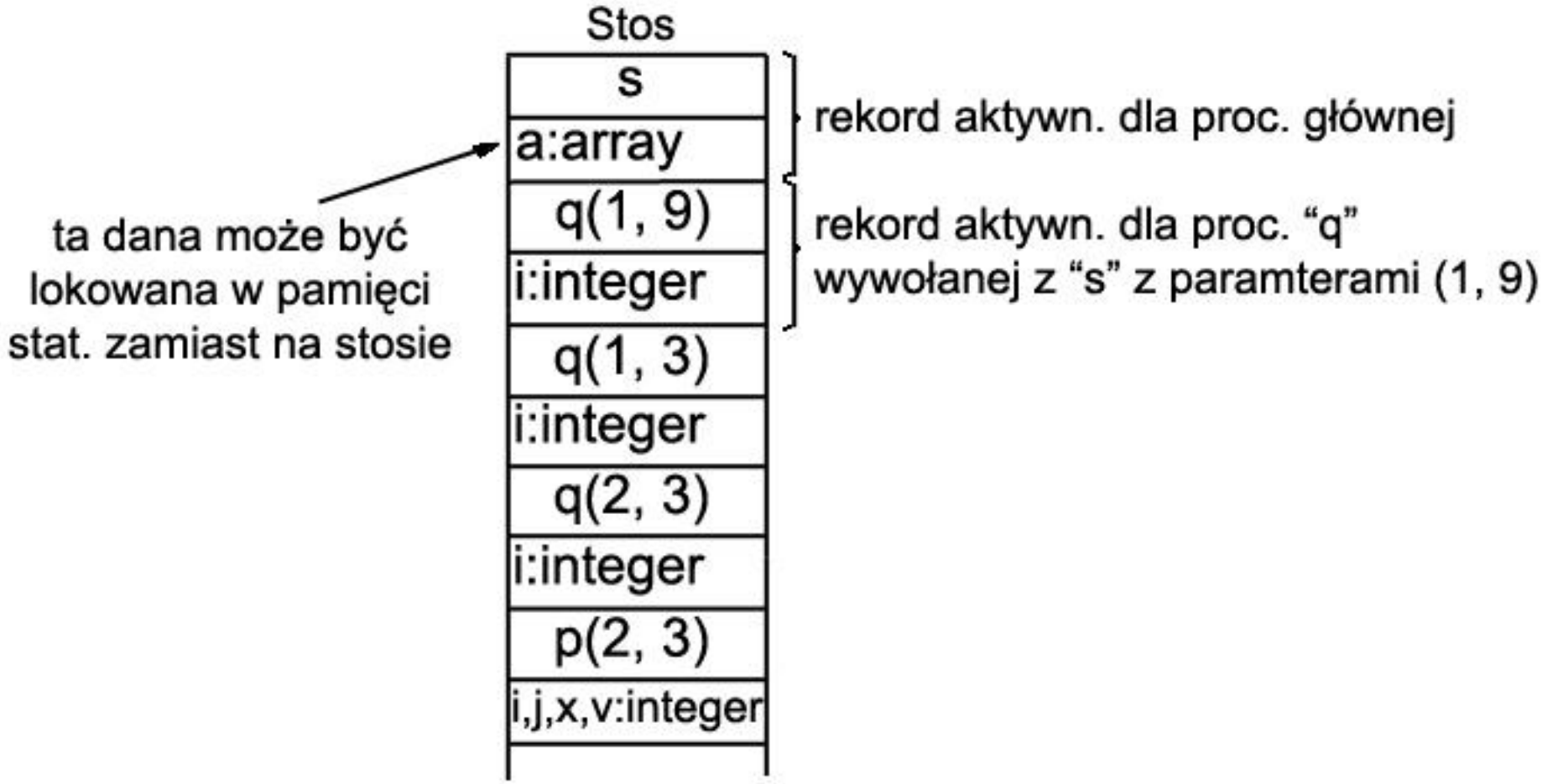


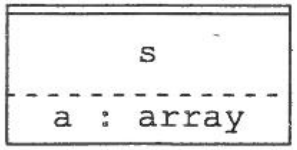
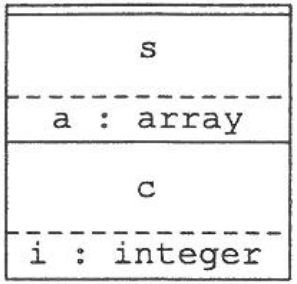
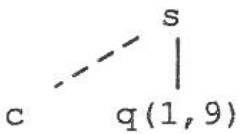
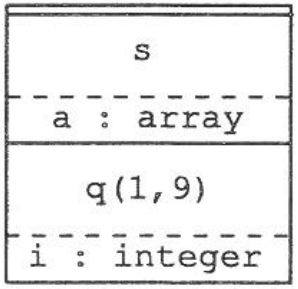
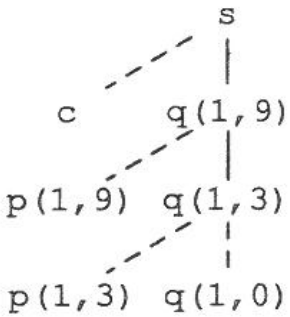
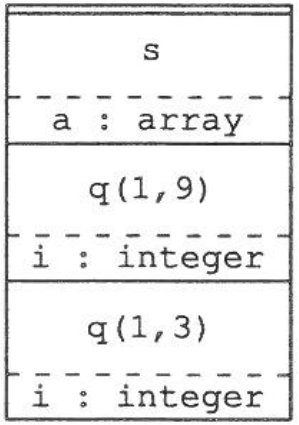
Drzewo aktywności



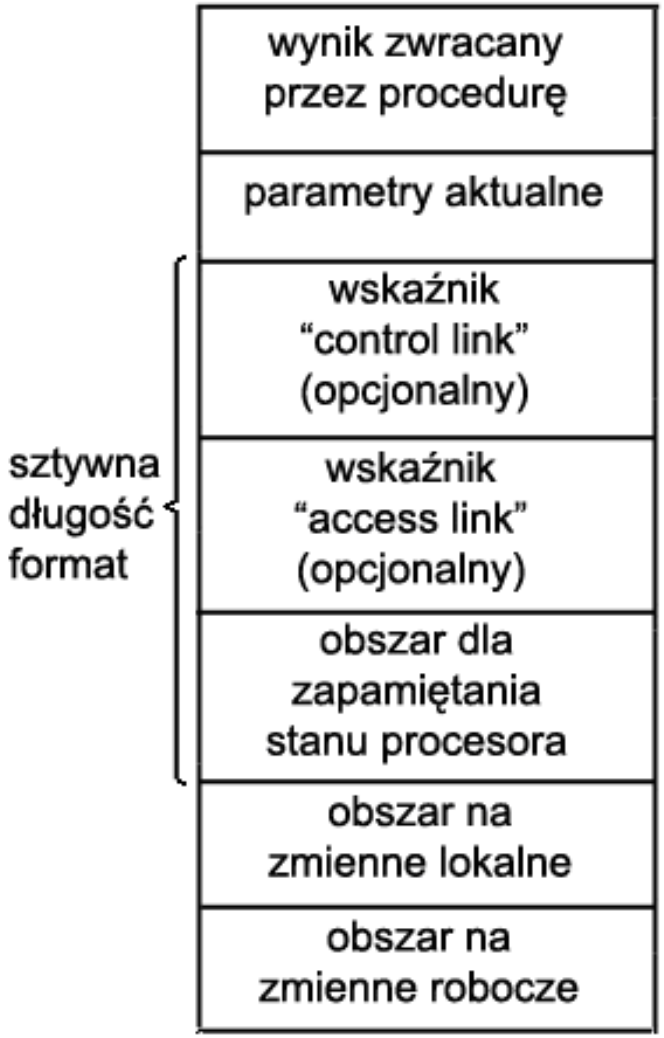
1. Każdy wierzchołek reprezentuje czynność procedury
2. Korzeń reprezentuje czynność głównego programu
3. Wierzchołek *a* jest przodkiem wierzchołka *b* ⇔ sterowanie przechodzi z czynnej procedury *a* do *b*
4. Wierzchołek *a* jest położony z lewej strony wierzchołka *b* oraz *a* i *b* są rodzeństwem ⇔ czas aktywności *a* jest wcześniejszy od czasu aktywności *b*

Stos programu



Pozycje w drzewie aktywacji	Rekordy aktywacji na stosie	Uwagi
s		ramka dla s
		c jest aktywowana
		ramka dla c została usunięta i q(1,9) została wstawiona na stos
		sterowanie właśnie powróciło do q(1,3)

Rekord aktywności (rama) procedury



często do tego celu wykorzystywany jest rejestr, a nie pamięć

także czasami są przekazywane w rejestrach

wskazuje rekord aktywności procedury wywołującej

umożliwia dostęp do danych nielokalnych w językach dopuszczających zagłębianie procedur (Pascal PL I)

adres powrotu, status procesora przed wywołaniem procedury i inne informacje konieczne do odtworzenia stanu maszyny po powrocie z procedury

zmienne lokalne procedury

zmienne robocze, konieczne np. dp wyliczania wartości wyrażeń

Rekord aktywności (rama) procedury

W skład rekordu aktywacji (ramki procedury) mogą wchodzić (obligatoryjnie lub opcjonalnie – zależnie od języka programowania oraz jego konkretnej implementacji):

- wartość zwracana,
- parametry aktualne wywołania,
- wiązanie sterowania (*control link*),
- wiązanie dostępu (*access link*),
- zapamiętany stan procesora,
- zmienne lokalne,
- wskaźniki do lokalnych tablic o zmiennej długości,
- lokalne tablice o stałej długości,
- zmienne tymczasowe (robocze).

Rekord aktywności (rama) procedury

Pamięć dla rekordów aktywacji może być rezerwowana:

- w pamięci statycznej,
- w pamięci stosowej,
- w pamięci sterty,
- częściowo w rejestrach procesora.

Rekord aktywności (rama) procedury

Przy stosowej rezerwacji pamięci zmienne lokalne dla każdego wywołania są przypisywane do nowej pamięci w każdej aktywacji, ponieważ na stosie jest umieszczany nowy rekord aktywacji; wartości zmiennych lokalnych są usuwane, gdy aktywacja się kończy, ponieważ pamięć dla nich jest usuwana razem z rekordem aktywacji.

Jeżeli ta sama procedura jest n-krotnie wywoływana rekurencyjnie to, przy stosowej rezerwacji pamięci, rekord aktywacji tej procedury jest umieszczany n-krotnie na stosie, po jednym wystąpieniu dla każdej aktywacji.

Rekord aktywności (rama) procedury

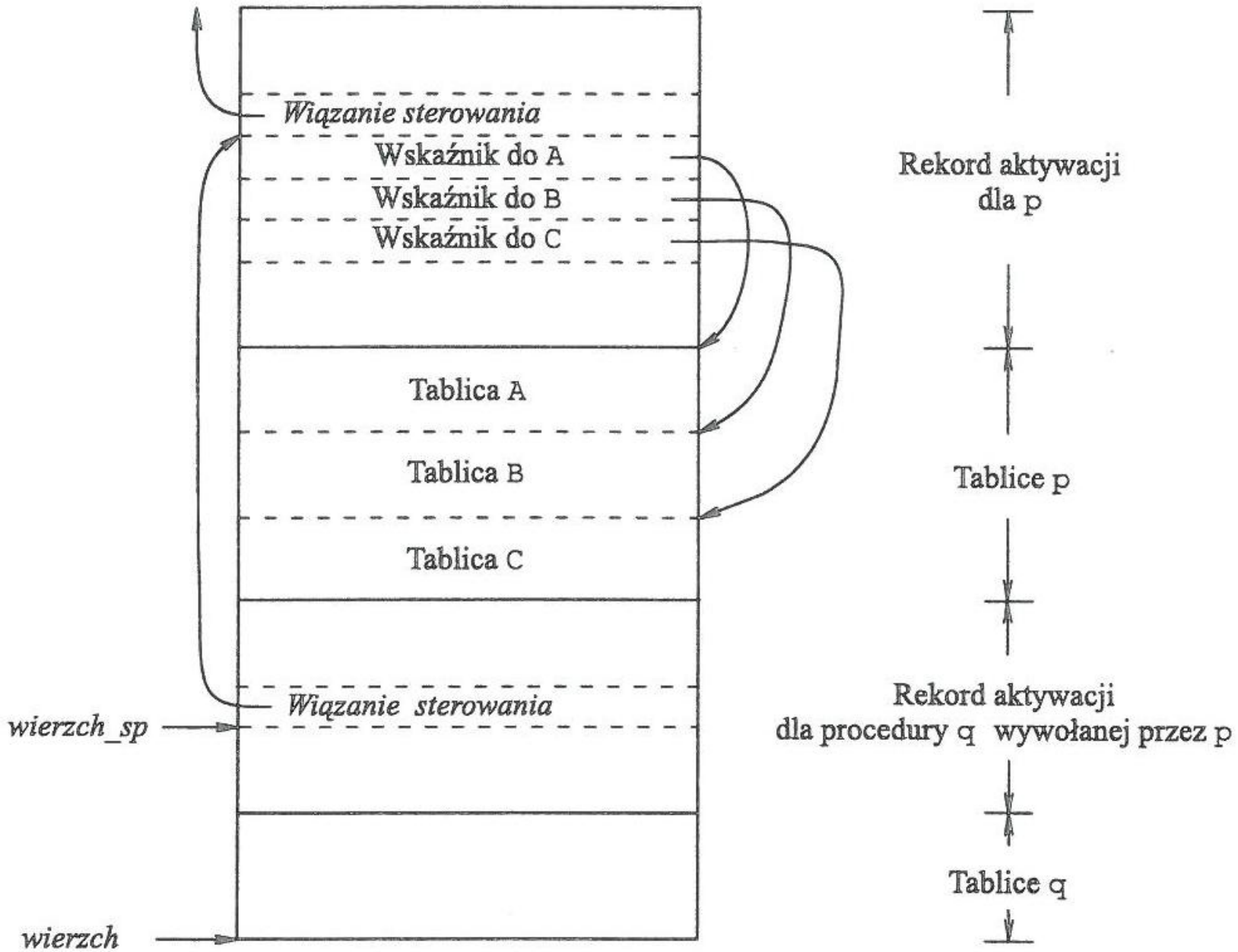
Rezerwacja stertowa rekordów aktywacji jest stosowana, jeżeli wartości zmiennych lokalnych muszą być zachowane, gdy aktywacja się kończy lub gdy aktywacja wywoływana istnieje dłużej niż aktywacja wywołująca.

W rezerwacji statycznej rekordów aktywacji nazwy są przypisywane do pamięci w trakcie kompilacji; ponieważ te przypisania nie zmieniają się w trakcie działania programu, za każdym razem, gdy procedura jest aktywowana, jej nazwy przypisane są do tych samych adresów pamięci, więc wartości nazw lokalnych są zachowywane między kolejnymi aktywacjami tej procedury.

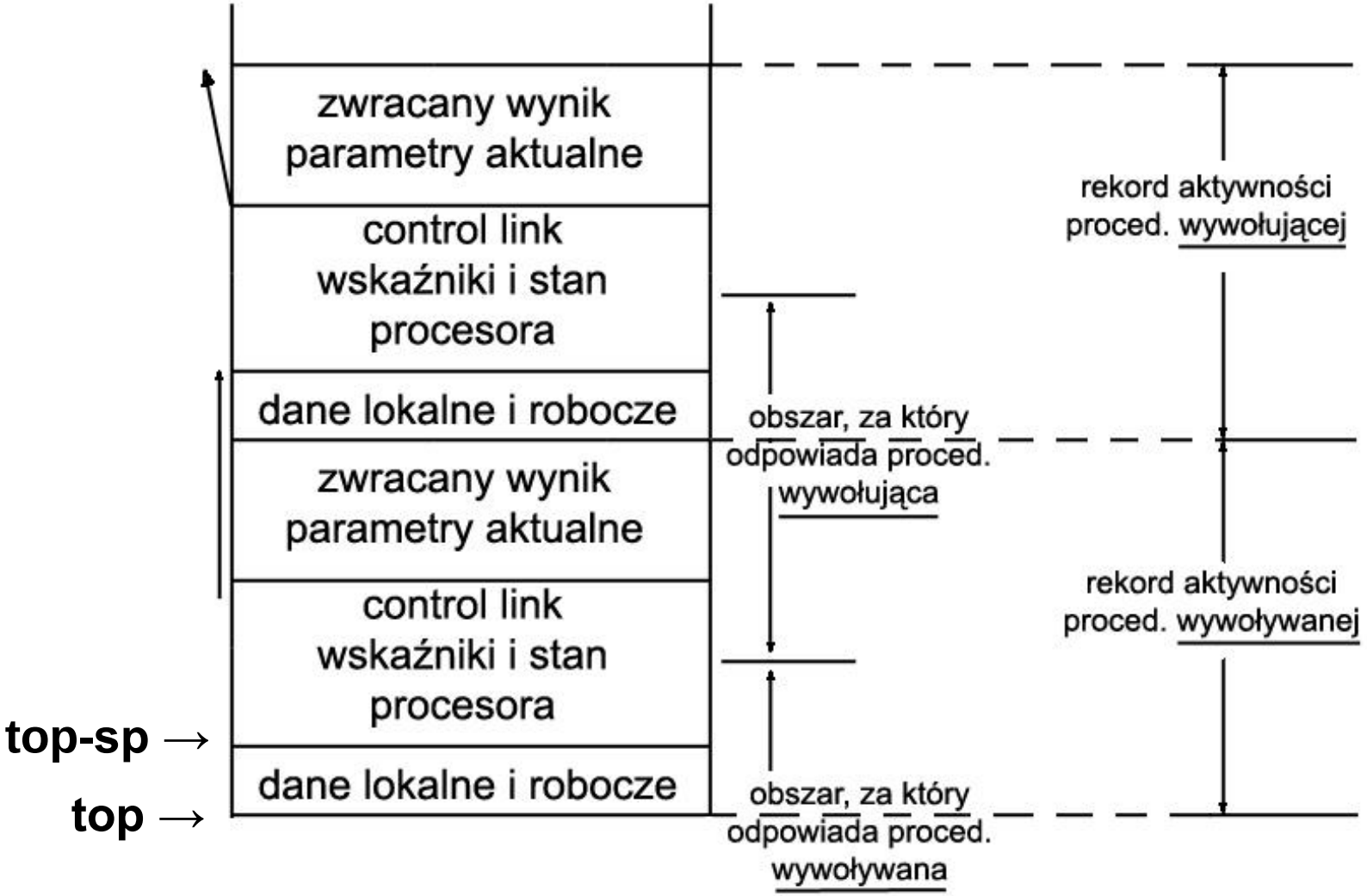
Rekord aktywności (rama) procedury

Wiązanie sterowania (*control link*) jest wskaźnikiem umieszczonym w rekordzie aktywacji procedury wywoływanej wskazującym na ustalone miejsce w rekordzie aktywacji procedury wywołującej.

Bezpośredni dostęp do umieszczonych w rekordzie aktywacji parametrów aktualnych, zmiennych lokalnych i zmiennych tymczasowych wymaga znajomości przesunięcia (*offset*) adresu pamięci tych obiektów w stosunku do ustalonego miejsca w rekordzie aktywacji. Adres tego ustalonego miejsca w rekordzie aktywacji przechowywany jest w odpowiednim rejestrze procesora (np. EBP w procesorach Intela).


Wiązanie sterowania
Wskaźnik do A
Wskaźnik do B
Wskaźnik do C
Tablica A
Tablica B
Tablica C
Wiązanie sterowania
wierzch_sp
wierzch
Rekord aktywacji dla p
Tablice p
Rekord aktywacji dla procedury q wywołanej przez p
Tablice q

Rekord aktywności (rama) procedury



Rekord aktywności (rama) procedury

Sekwencja wywołania :

1. Procedura wywołująca oblicza parametry aktualne i umieszcza je w rekordzie aktywności procedury wywoływanej.
2. Procedura wywołująca zapamiętuje adres powrotu i starą wartość *„top-sp”* (w polu *„control link”*) w rekordzie aktywności procedury wywoływanej. Następnie modyfikuje wartość *„top-sp”* (patrz poprzedni slajd).
3. Sterowanie przekazywane jest procedurze wywoływanej. Procedura wywoływana zapamiętuje pozostałe informacje związane ze stanem procesora.
4. Procedura wywoływana inicjuje dane lokalne i rozpoczyna wykonywanie „właściwych” czynności obliczeniowych.

Rekord aktywności (rama) procedury

Sekwencja powrotu:

1. Procedura wywoływana umieszcza zwracany wynik w swoim rekordzie aktywności, usuwa ze stosu część rekordu ze zmiennymi lokalnymi i roboczymi.

2. Wykorzystując informacje zawarte we własnym rekordzie aktywności (pole „control link”) odtwarza poprzednią wartość „top-sp”, a następnie odtwarza poprzedni stan procesora i przekazuje sterowanie procedurze wywołującej zgodnie z adresem powrotu.

3. Procedura wywołująca kopiuje wynik procedury wywoływanej do własnego rekordu aktywności, usuwa ze stosu resztę rekordu aktywności procedury wywoływanej i wznawia własne obliczenia.

Dostęp do zmiennych nielokalnych

Dwa generalne podejścia (dwie zasady widzialności):

(1) Statyczne (leksykalne): na podstawie znajomości kodu źródłowego można jednoznacznie przyporządkować deklarację zmiennej do wystąpienia nazwy zmiennej. Najczęściej stosowaną zasadą jest:

(a) jeśli nazwa jest zadeklarowana w tym samym bloku, w którym występuje, to właściwą deklaracją jest deklaracja z własnego bloku.

(b) jeśli nazwa nie jest zadeklarowana w tym bloku, w którym występuje, to właściwą deklaracją jest deklaracja w tym bloku, który jest najbardziej wewnętrznym w sensie zagnieżdżenia blokiem zawierającym deklarację tej nazwy.

Dostęp do zmiennych nielokalnych

Dostęp do danych nielokalnych w językach z widzialnością leksykalną (statyczną) oraz z możliwością zagnieżdżania procedur jest realizowany poprzez mechanizm wiązań dostępu (*access link*) lub mechanizm tablic *display*.

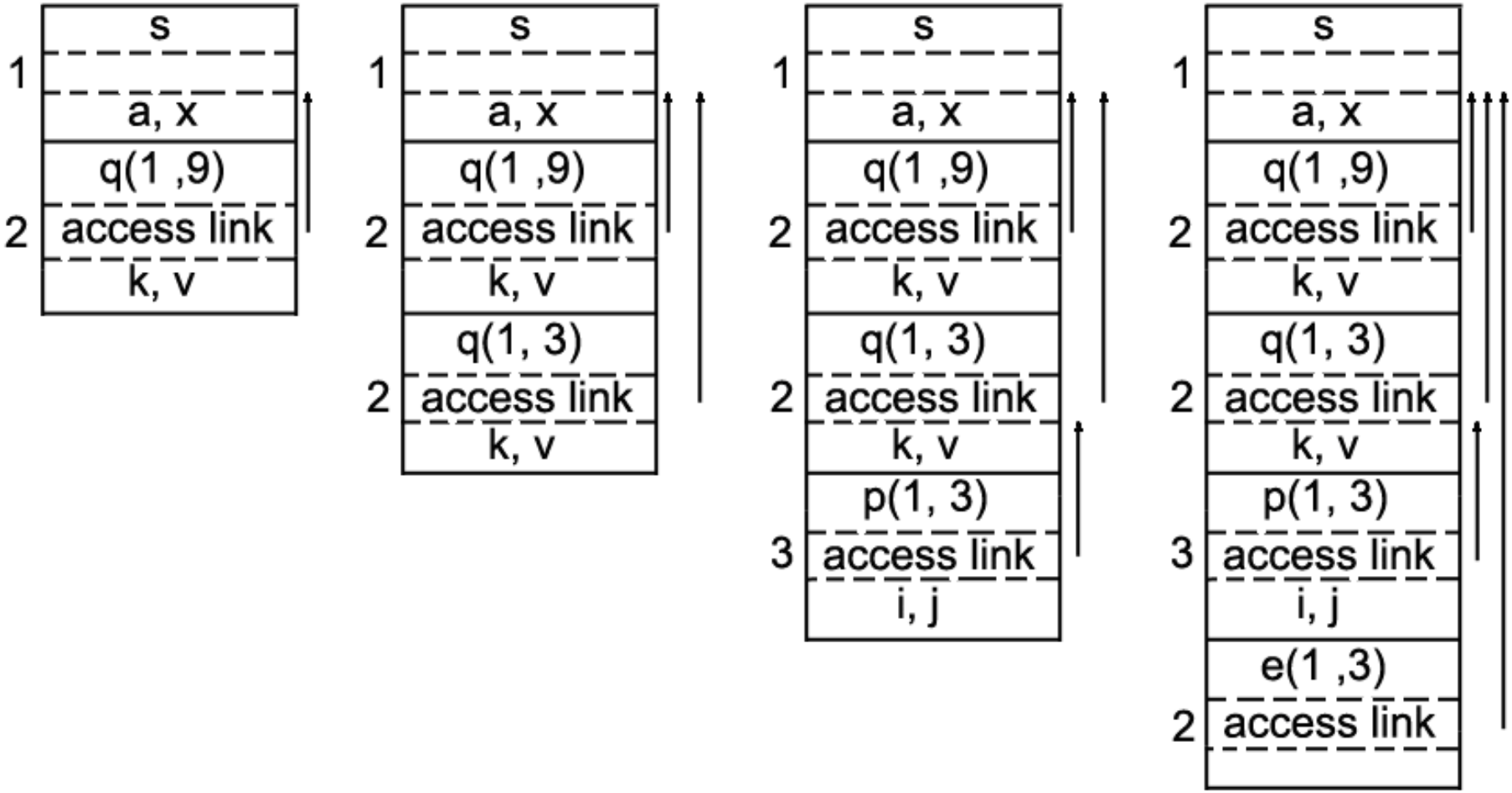
Jeżeli procedura *p* jest zagnieżdżona **bezpośrednio** w procedurze *q*, wiązanie dostępu (*access link*) w rekordzie aktywacji *p* wskazuje wiązanie dostępu w rekordzie dla ostatniej aktywacji *q*.

Zasada statyczna; możliwość zagnieżdżanych procedur

```
program S(input , output);
var  a : array [0..10] of integer;
     x : integer;
     procedure r;
1
       2   var  i : integer;
           begin …. a … end;
     procedure e(i,j : integer);
       2   begin x := a[i] ; a[i] := a[j]; a[j] := x; er
     procedure q(m,n : integer);
           var  k ,v :integer;
           function p(y,z : integer) : integer;
       2       var i,j :integer;
           3
           begin     … a … … v …
                 …  q (i,j); …
           end;
           begin … end;
     begin ... end;
begin  ...  end.
```

poziomy zagnieżdżenia

Zasada statyczna; możliwość zagnieżdżanych procedur



Zasada statyczna; możliwość zagnieżdżanych procedur

Chcąc przy widzialności leksykalnej w procedurze *p* uzyskać dostęp do poszukiwanej zmiennej nielokalnej z procedury *q* należy przejść tyle razy przez łańcuch wiązań dostępu (*access link*), ile wynosi różnica poziomów zagnieżdżenia procedur *p* i *q*, a po dotarciu do rekordu procedury *q* należy od ustalonego miejsca w tym rekordzie odliczyć odpowiednie przesunięcie (*offset*) do poszukiwanej zmiennej nielokalnej.

Zasada statyczna; możliwość zagnieżdżanych procedur

Niech procedura „*p*” o poziomie zagnieżdżenia n_p zawiera wystąpienie zmiennej „*a*” deklarowanej na poziomie zagnieżdżenia n_a, $n_a \leq n_p$

Pamięć dla „*a*” może być znaleziona w następujący sposób :

1. Gdy sterowanie jest w „*p*”, rekord aktywności „*p*” jest na wierzchołku stosu. Należy ($n_p - n_a$) razy „przejść” przez łańcuch wskazań „*access link*”. Wartość ($n_p - n_a$) jest znana na etapie kompilacji. W ten sposób osiągnie się rekord aktywności, w którym zlokalizowana jest pamięć dla „*a*”.
2. Właściwa lokalizacja „*a*” jest określona poprzez znany na etapie kompilacji offset wewnątrz rekordu aktywności procedury zawierającej deklarację „*a*”

Zasada statyczna; możliwość zagnieżdżanych procedur

Przy ustawianiu wiązań dostępu w sekwencji wywołania nowej aktywacji wykorzystujemy informację o poziomie zagnieżdżenia procedury wywołującej i procedury wywoływanej oraz w pewnych przypadkach dotychczasowe wiązania dostępu z rekordów aktywacji znajdujących się na stosie.

Uzupełnienie sekwencji czynności dokonywanych przy wywoływaniu

Procedura *„p”* o poziomie zagnieżdżenia n_p wywołuje procedurę *”x”* o poziomie zagnieżdżenia n_x.

(1) Przypadek $n_p < n_x$

Wówczas *”x”* jest bardziej zagłębiona niż *„p”* , *”x”* jest zanurzone w *”p”*. *„Access link”* w procedurze wywoływanej musi wskazywać rekord aktywności procedury wywołującej.

(2) Przypadek $n_p \geq n_x$

Wówczas *”x”* nie jest bardziej zagłębione niż *„p”*. „Przechodząc” $(n_p - n_x + 1)$ razy łańcuchem wskazań *„access link”* poczynając z rekordu aktywności procedury wywołującej dojdzie się do rekordu aktywności najbardziej wewnętrznej procedury zawierającej zarówno *”x”* jak i *”p”*. *„Access link”* w procedurze wywoływanej powinien zawierać adres identyczny jak zawartość *„access link”* osiągniętego po przejściu $(n_p - n_x)$ razy łańcucha wskazań *„access link”* poczynając od *„access link”* procedury wywołującej.

W każdym przypadku *„access link”* w rekordzie aktywności procedury wywoływanej musi być określany przez procedurę wywołującą przed przekazaniem sterowania do procedury wywoływanej.

Przykład

```
program param(input , output);
    procedure b(function h(n:integer):integer);
        begin writeln(h(2)) end; {b}

    procedure c;
        var m:integer;
        function f(n:integer):integer;
            begin f:=m+n end; {f}
        begin m:=0; b(f) end; {c}
begin
    c;
end.
```

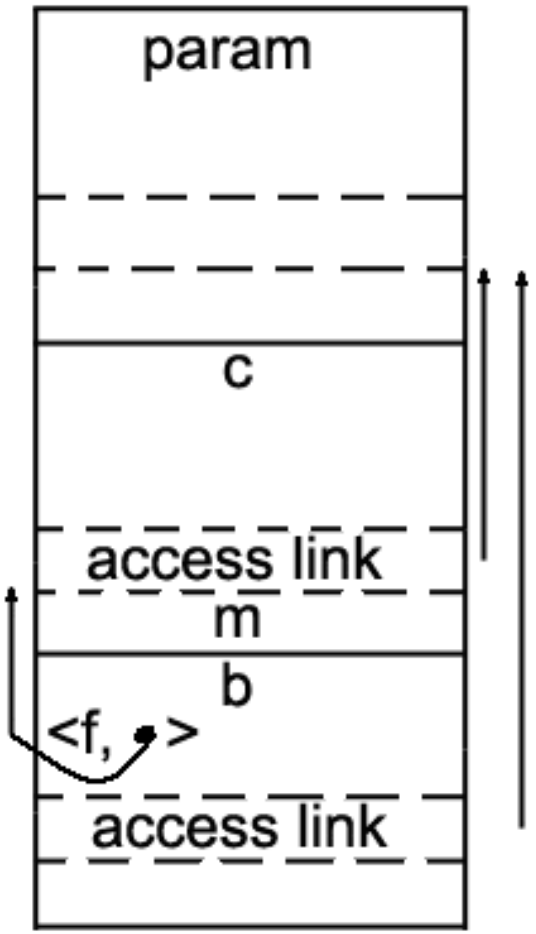
Przykład

```
program param(input , output);
    procedure b(function h(n:integer):integer);
        begin writeln(h(2)) end; {b}

    procedure c;
        var m:integer;
        function f(n:integer):integer;
            begin f:=m+n end; {f}
        begin m:=0; b(f) end; {c}
    begin
        c;
    end.
```

zakres deklaracji „m”

procedura przekazywana
jako parametr
musi być “wyposażona”
w obliczony algorytmem
“access link”



Tablice display

(procedury nie są przekazywane jako parametry)

Dostęp do zmiennych nielokalnych, szybszy niż za pomocą wiązań *access-link*, można uzyskać, stosując tablicę *d* wskaźników do rekordów aktywacji, zwaną *display*. Dzięki tej tablicy pamięć dla zmiennej nielokalnej *a* na głębokości zagnieżdżenia *i* może zostać znaleziona w rekordzie aktywacji wskazywanym przez element tablicy *d[i]*.

Gdy nowy rekord aktywacji jest tworzony dla procedury o głębokości zagnieżdżenia *i*, należy:

- zapamiętać starą wartość *d[i]* w nowym rekordzie aktywacji,
- pozycji *d[i]* w tablicy *display* przypisać wskaźnik do nowego rekordu aktywacji.

Tablice display

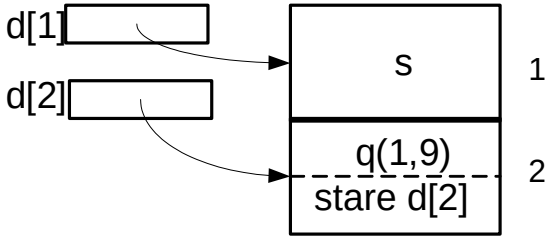
(procedury nie są przekazywane jako parametry)

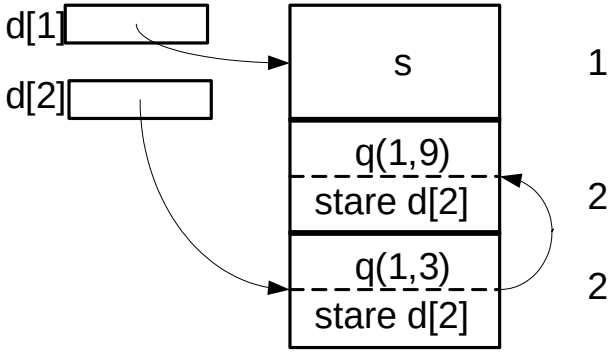
W mechanizmie tablic *display* przy widzialności leksykalnej pamięć dla zmiennej nielokalnej zlokalizowanej na głębokości zagnieżdżenia *i* może zostać znaleziona w rekordzie aktywacji wskazywanym przez element tablicy *display[i]*.

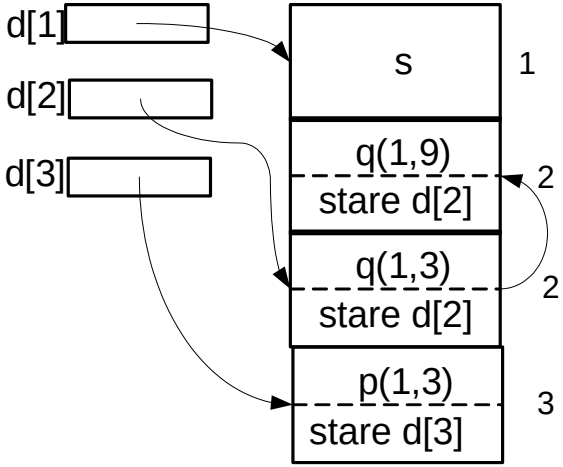
Elementy tablic *display* muszą się zmieniać, gdy występuje nowa aktywacja, a także wtedy, gdy sterowanie powraca z aktywacji i należy przywrócić poprzednią zawartość elementu tablicy *display*, dlatego też w czasie nowej aktywacji należy w nowym rekordzie aktywacji zapamiętać poprzednią zawartość odpowiedniego elementu tablicy *display*.

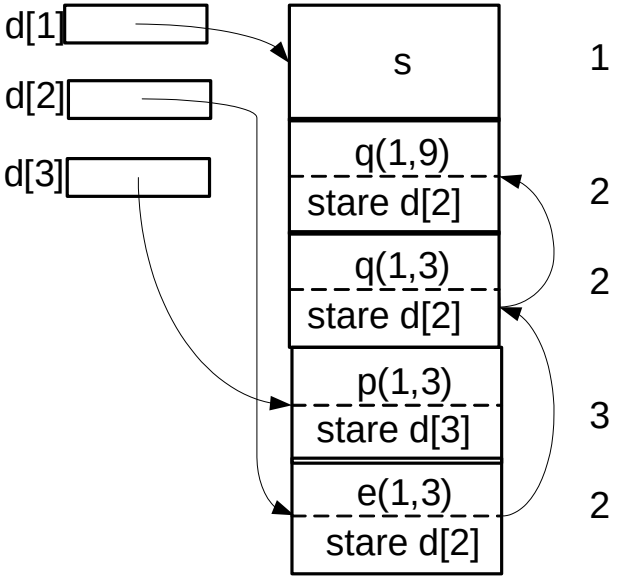
Tablice display

(procedury nie są przekazywane jako parametry)









Zasada widzialności dynamicznej

- Przyporządkowanie deklaracji do wystąpienia zmiennej odbywa się w trakcie wykonania programu. Najczęściej stosowana zasada: przypisanie nielokalnej nazwy do określonej lokalizacji w pamięci nie ulega zmianie w czasie przejścia sterowania do nowej procedury (nazwa nielokalna w wywoływanej procedurze odwołuje się do tej samej pamięci, co w procedurze wywołującej).

Zasada dynamiczna - przykład

```
program dynamic(input, output);
var  r : real;
     procedure show;
     begin
          write (r:5:3)
     end;
     procedure small;
     var  r : real;
     begin
          r := 0.125;
          show;
     end;
begin
     r := 0.25;
     show; small; writeln;
     show; small; writeln;
end.
```

zmienna "r" w procedurze "show" jest nielokalna

Zasada statyczna

Wynik programu

0.250 0.250
0.250 0.250

Zasada dynamiczna:

Wynik programu

0.250 0.125
0.250 0.125

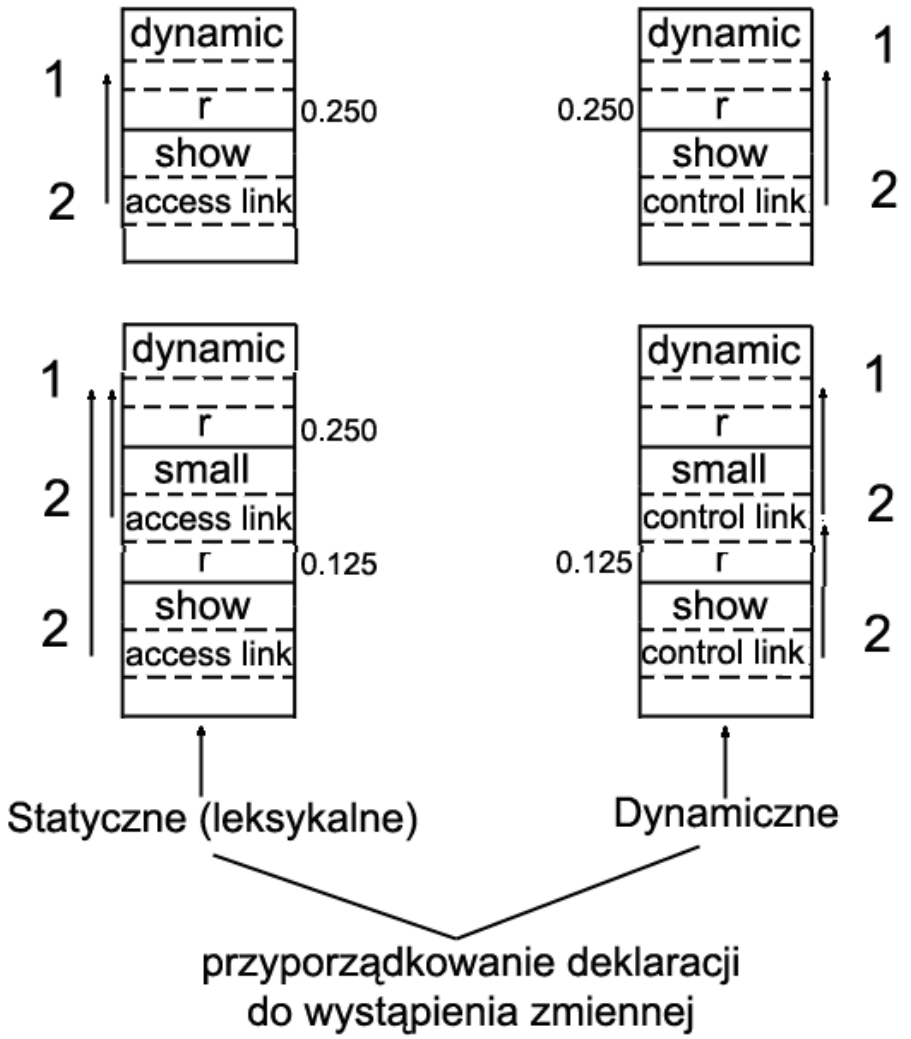
Zasada dynamiczna

Implementacja wiązań dynamicznych może być dwojaka:

- Dostęp głęboki (pewna analogia do zasady statycznej realizowanej poprzez wskaźniki *access-link*). Nie wykorzystuje się wiązań typu *access-link* i używa się wiązań typu *control-link* do przeszukiwania stosu i poszukiwania pierwszego rekordu aktywacji zawierającego wartość dla tej nazwy nielokalnej. „Głębokości” przeszukiwania stosu nie można wyznaczyć w czasie kompilacji.

Zasada dynamiczna - przykład

```
program dynamic(input, output);
var   r : real;
      procedure show;
      begin
            write (r:5:3)
      end;
      procedure small;
      var   r : real;
      begin
            r := 0.125;
            show;
      end;
begin
      r := 0.25;
      show; small; writeln;
      show; small; writeln;
end.
```



Zasada dynamiczna

Drugi rodzaj implementacji wiązań dynamicznych:

- Dostęp płytki (pewna analogia do zasady statycznej realizowanej poprzez tablice *display*). Przechowywanie obecnej wartości dla każdej nazwy w pamięci zarezerwowanej statycznie. Gdy następuje wywołanie procedury *p* nazwa lokalna *n* zajmuje pamięć zarezerwowaną statycznie dla *n.* Poprzednia wartość *n* jest przechowywana w rekordzie aktywacji dla *p* i należy ją przywrócić do pamięci statycznej podczas sekwencji powrotu procedury *p*.

Alokacja rekordów aktywności na stercie

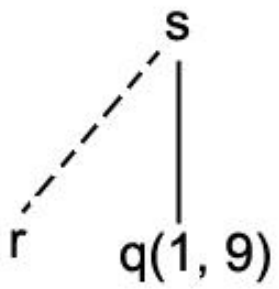
Rekordów aktywności procedur nie można alokować na stosie w dwóch przypadkach :

- Gdy wartości zmiennych lokalnych mają być zachowane po opuszczeniu przez sterowanie procedury.
- Gdy wywoływana procedura może pozostać aktywna, pomimo, że wywołująca ją procedura przestała być aktywna.

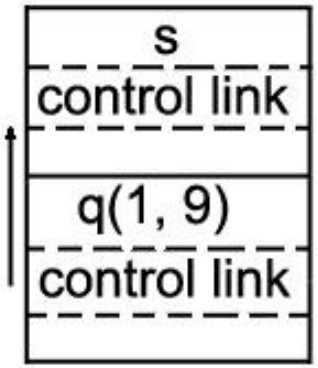
W obu tych przypadkach dealokacja rekordu aktywności procedury, która przestała być aktywna, nie może następować zgodnie z zasadą Last-In First-Out. Miejscem alokacji rekordów aktywności jest wówczas sterta (heap) a nie stos (stack).

Alokacja rekordów aktywności na stercie

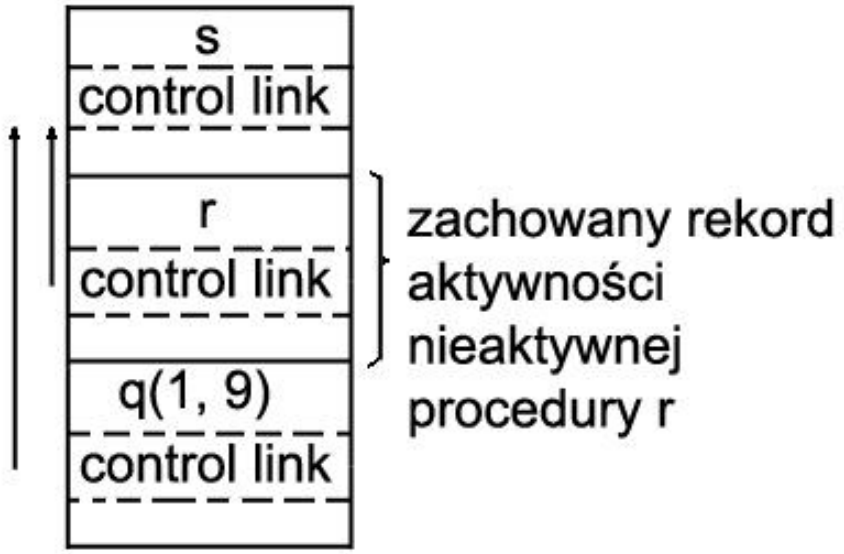
Drzewo aktywności



Rekordy aktywności alokowane na stosie



Rekordy aktywności alokowane na stercie



Przekazywanie parametrów przez wartość

Przekazywanie parametrów do procedury przez wartość (*call–by–value*)

(a) Parametr formalny jest traktowany jako nazwa lokalna, więc pamięć dla niego jest rezerwowana w rekordzie aktywności procedury wywoływanej

(b) Procedura wywołująca oblicza parametry aktualne i umieszcza ich wartości (*r-wartości*) w rekordzie aktywności procedury wywoływanej w miejscu przeznaczonym na parametry formalne

Przekazywanie parametrów przez wartość

Przykład :

```
var a, b : integer;

procedure swap (x, y :integer);
     var  temp : integer;

     begin
          temp := x;
          x := y;
          y := temp;
     end;

begin
     a := 1    b := 2;
     swap (a, b);
     writeln('a =', a, 'b =', b);
                              /* a = 1    b = 2 */
end;
```

a := 1
b:= 2
x := a
y := b
temp := x
x := y
y := temp
write(a,b)

Przekazywanie parametrów przez referencję

Przekazywanie parametrów do procedury przez adres (referencję, lokację, *call–by–reference*)

- (a) Jeżeli aktualny parametr jest nazwą lub wyrażeniem posiadającym *l-wartość* (adres), wówczas przekazywana jest *l-wartość* do rekordu aktywności procedury wywoływanej;
- (b) Jeżeli aktualny parametr jest wyrażeniem nie posiadającym *l-wartości* (np.: *a* + *b* czy też *2* + *x*) wówczas wyrażenie jest obliczane, jego wartość (r-*wartość*) umieszczana jest w obszarze roboczym i adres tego położenia roboczego jest przekazywany do procedury wywoływanej.

Przekazywanie parametrów przez referencję

Przykład :

```
var  i : integer;
     a : array[1..20] of integer;

procedure swap (var x, y:integer);
     var temp : integer;

     begin
          temp := x;
          x := y;
          y := temp;
     end;

begin
     ...  swap(i , a[i]); ...
end.
```

x := &i
y := &a[i]
*temp := *x*
**x := *y*
**y := temp*
Jeśli I_0 jest początkową wartością zmiennej i to po wykonaniu procedury mamy:
$i = a[I_0]$
$a[I_0] = I_0$

Przekazywanie parametrów przez nazwę

Przekazywanie parametrów do procedury przez nazwę (*call–by–name*)

(a) Procedura wywoływana traktowana jest jakby była makrosem; wywołanie procedury jest w miejscu wywołania ”zastępowane” ciałem procedury, przy czym parametry formalne są zastępowane „dosłownie” parametrami aktualnymi (rozwinięcie w miejscu wywołania, *in-line expansion*)

(b) Lokalne zmienne w procedurze wywoływane są rozróżniane od zmiennych o tych samych nazwach w procedurze wywołującej, tzn. można wyobrazić sobie, że każda zmienna lokalna jest przemianowana przed rozpoczęciem „rozwijania makrosa”

(c) Parametry aktualne są ujmowane w nawiasy wszędzie tam, gdzie jest to konieczne dla zachowania ich integralności.

Przekazywanie parametrów przez nazwę

Przykład:

```
var  i : integer;
     a : array [1..10 ] of integer;

     procedure swap(name x, y:integer);
          var temp : integer;
          begin
               temp := x;
               x := y;
               y := temp;
          end;

     begin
          (...)
          swap(i,a[i]); (...)
     end.
```

temp := i
i := a[i]
a[i] := temp

Jeśli I_0 jest początkową wartością zmiennej i, to po wykonaniu procedury mamy:

$i = a[I_0]$

$\mathbf{a[a[I_0]] = I_0}$

zamiast $\mathbf{a[I_0] = I_0}$

Przekazywanie parametrów metodą „przepisz-odtwórz”

Przekazywanie parametrów metodą mieszaną „przepisz-odtwórz” (*copy-restore linkage*)

(a) Przed przekazaniem sterowanie do procedury wywoływanej obliczane są wartości parametrów aktualnych. Wartości te (*r-wartości*) są przekazywane do procedury wywoływanej jak w wywołaniu przez wartość. Jednocześnie obliczane i zapamiętywane są adresy (*l-wartości*) tych parametrów aktualnych, które mają *l-wartość*.

(b) Gdy sterowanie powraca z procedury wywoływanej do wywołującej bieżące wartości parametrów formalnych (*r-wartości*) są kopiowane z powrotem do rekordu aktywności procedury wywołującej z wykorzystaniem zapamiętanych adresów (*l-wartości*) parametrów aktualnych. Oczywiście odtwarzane są tylko parametry aktualne mające *l-wartość*.

Przekazywanie parametrów metodą „przepisz-odtwórz”

Przykład:

Wywołanie procedury *”swap”* w postaci *swap(i , a[i])* z poprzednich przykładów da wynik poprawny tzn:

$i = a[I_0]$
$a[I_0] = I_0$

gdzie I_0 początkowa wartość zmiennej ”i”

Przykład:

```
var a: integer;

procedure unsafe(copy-restore x: integer);
     begin
          x := 2;
          a := 0;
     end;

begin;
     a := 1;
     unsafe(a);
     writeln(a);
end.
```

call-by-reference
a := 1
x := &a
****x := 2***
a := 0
wynik: writeln(a) = 0

copy–restore
a := 1
l_val_act := &a
x := a
x:= 2
a := 0
****l_val_act := x***
wynik: writeln(a) = 2