编译原理

Compiler Construction Principles

朱 青

信息学院计算机系，

中国人民大学，

zqruc2012@aliyun.com

第8章:运行时存储空间组织

- 8.1 源语言中问题的讨论
- 8.2 存储器的组织
- 8.3 静态存储分配
- 8.4栈式存储分配
- 8.5嵌套式过程语言栈式实现
- 8.6 堆式动态存储分配

第8章:运行时存储空间组织

静态存储分配:编译时就可以分配的存储.

动态存储分配:存储必须等到运行时分配.

存储分配策略:

静态分配策略:(FORTRAN)

在编译时完全确定的数据项,如:整(实)数,布尔值的存储空间,等等.

栈式动态分配策略:(ALGOL)

可以递归,又有可变数组.

堆式动态分配策略:(PL/1,LISP,PASCAL,C)

可以由用户申请或释放存储的语言.

8.1 源语言中问题的讨论

8.1.1 过程举例

注意区分:一个过程的源程序正文和其运行时刻的活动.

```
program sort(input,output);
  var a:array[0..10] of integer;
```

```
procedure readarray;
   var  i : integer;
    begin
      for  i:= 1 to 9 do read(a[i]);
    end;
function partition(y,z:integer):integer;
    var i,j,x,v:integer;
    begin
       …
   end;
```

```
procedure quicksort(m,n:integer);
var i:integer;
begin
   if  (n>m) then
   begin  i:=partition(m,n);
              quicksort(m,i-1);
              quicksort(i+1,n);
   end;
end;
```

```
begin
    a[0]:=-9999;
    a[10]:=9999;
    readarray;
    quicksort91,9)
end.
```

注意:与存储分配有关的内容:

过程定义, 过程的名字,

过程体,　函数,

实参, 形参, 过程调用.

8.1.2 活动树

在程序执行过程中,关于程序之间的控制流,做如下的假定:

1) 顺序控制方式,即程序的执行由若干执行的步骤组成,控制的每一步都对应程序中一个确定点.

2)过程的每次执行都从过程的开头开始,最后结束时控制返回到紧跟在对本过程调用的点之后.

活动态:该过程的一次执行.

生存期:该程序的执行中若干步骤的顺序序列.

活动态quicksort(1,9)的生存期是:

执行打印 enter quicksort(1,9)开始

到　打印 leave quicksort(1,9)后结束.

假定 partition(1,9)的返回值是4.

此过程调用是递归的. 图示(下一页)表明了过程的活动态与生存期.

执行开始…

enter readarray

leave readarray

enter quicksort(1,9)

enter partition(1,9)

leave partition(1,9)

enter quicksort(1,3)

……

leave quicksort(1,3)

enter quicksort(5,9)

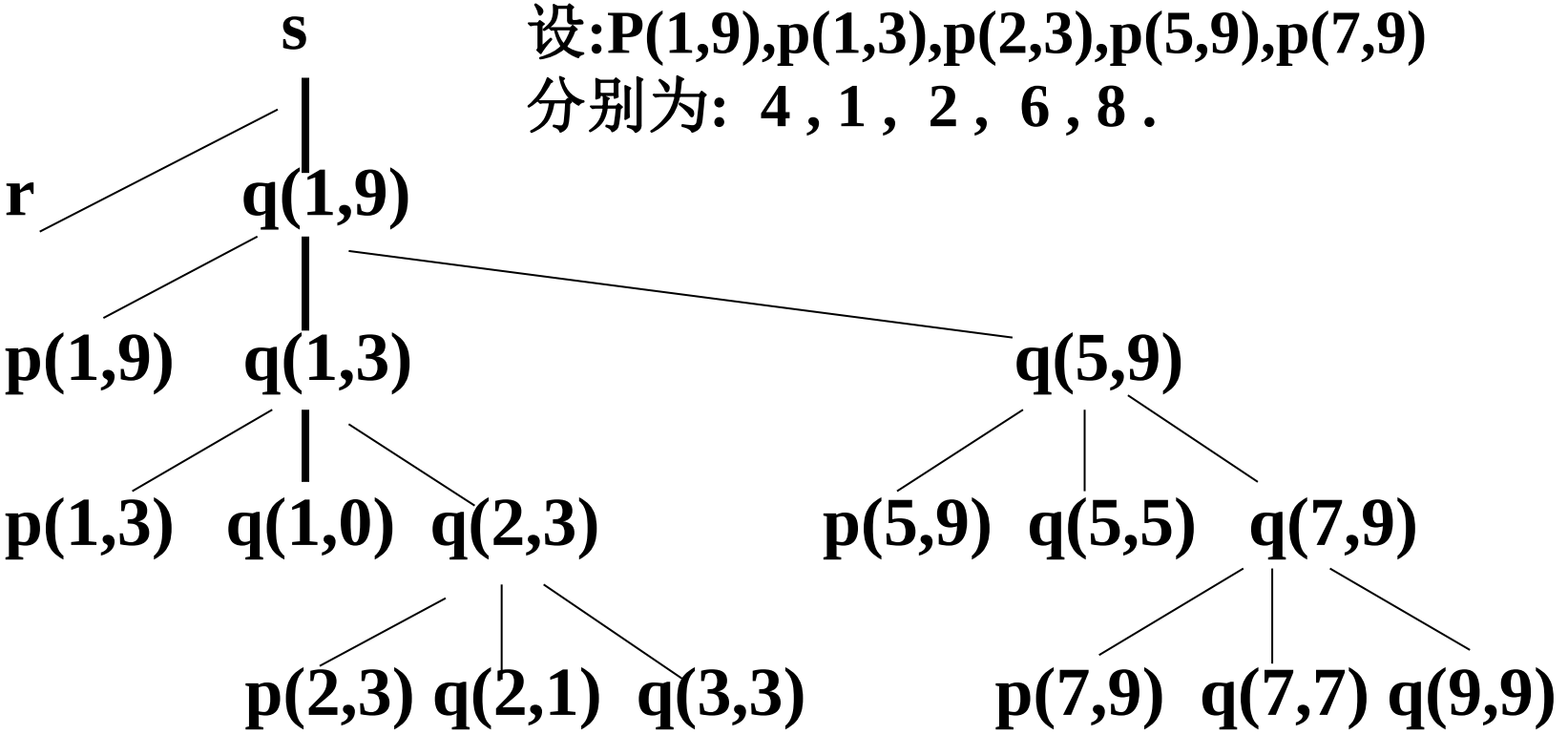
……

leave quicksort(5,9)

leave quicksort(1,9)

执行结束

由此构造出的活动树:


s
设:P(1,9),p(1,3),p(2,3),p(5,9),p(7,9)
分别为: 4 , 1 , 2 , 6 , 8 .
r
q(1,9)
p(1,9)
q(1,3)
q(5,9)
p(1,3)
q(1,0)
q(2,3)
p(5,9)
q(5,5)
q(7,9)
p(2,3)
q(2,1)
q(3,3)
p(7,9)
q(7,7)
q(9,9)

8.1.3 控制栈

控制栈用来保存过程活动的生存踪迹.

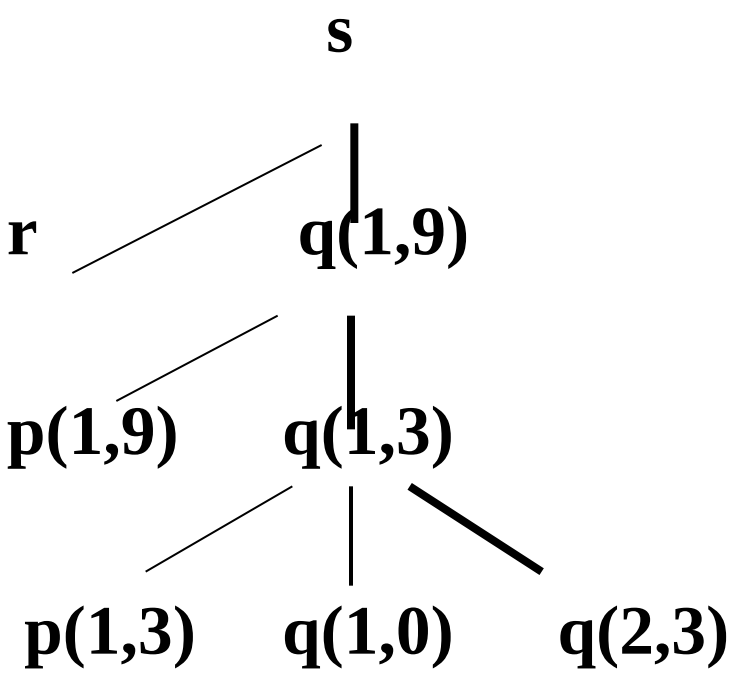
当一个活动开始时PUSH(活动结点)栈.

当一个活动结束时POP(活动结点)栈.

控制栈的内容与到达活动树的根结点的路经有关.当结点n位于栈顶时,栈中包含从n到根的路经上的那些结点.

如: 在活动树中,当控制进入q(2,3)代表的活动时,其控制栈为:

s q(1,9) , q(1,3), q(2,3)



控制栈中包含一条路径上的结点.

控制栈可扩充用来实现如PASCAL语言的栈式存储分配技术.

8.1.4 说明的作用域

语言中的说明是一个把信息与名字联系起来的语法结构. 如PASCAL中的说明片断:

```
var  i : integer;
```

或FORTRAN中的隐含说明,如:任何以字母I开头的名字都是整数.

在一个程序的不同部分可能有对同一个名字的独立的说明.语言的作用域规则决定了当这样的名字在程序正文中出现时应使用哪一个说明. 在前面的pascal(sort)程序中, i被定义了3次,分别在三个过程中,并且是相互独立的.

8.1.5 名字的联编

在程序设计语言中,名字的联编,亦称汇聚,是指把数据名字转换为数据目标的过程.“数据目标”对应于可以存储值的存储地址.

当一个环境把一个存储位置s于一个名字x联系起来时,称x受限于s,这个联系本身称为x的一个 联编.

第8章:运行时存储空间组织

- 8.1 源语言中问题的讨论
- 8.2 存储器的组织
- 8.3 静态存储分配
- 8.4栈式存储分配
- 8.5嵌套式过程语言栈式实现
- 8.6 堆式动态存储分配

8.2存储器的组织

8.2.1运行时存储器的组织

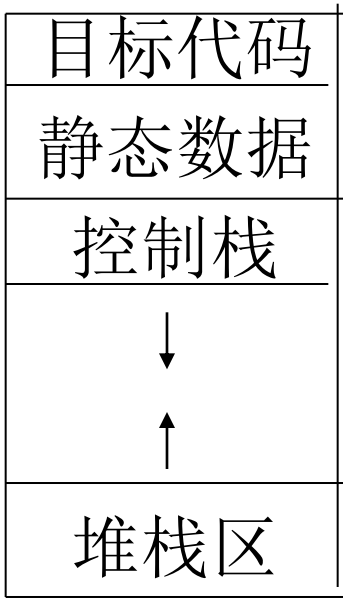
编译程序为了使它所编译的程序能够运行,要从操作系统获得一块存储空间.运行时刻的存储空间必须划分以用来存放:

1.生成的目标代码.

2.数据目标.

3.用于保存过程活动踪迹的一个控制栈的副本.

并对此空间进行组织和划分.



目标代码和静态数据的大小可以在编译时确定.尽可能多的分配静态数据对象.

控制栈和堆栈区的大小随着程序的运行而改变,故他们的增长方向相对.栈顶指针TOP.

8.2.2 活动记录

定义:为了管理过程在一次执行中所需的信息,要用一个连续的存储块,此连续的存储块称为活动记录.

活动记录的结构:

编译时局部数据的表示:

普通类型数据存放整数个字节.

组合类型数据存放一个连续的字节块中.

第8章:运行时存储空间组织

- 8.1 源语言中问题的讨论
- 8.2 存储器的组织
- 8.3 静态存储分配
- 8.4栈式存储分配
- 8.5嵌套式过程语言栈式实现
- 8.6 堆式动态存储分配

8.3 静态存储分配

静态存储分配是在编译时刻为每个数据项目确定出在运行时刻的存储空间中的位置.------FORTRAN存储分配.

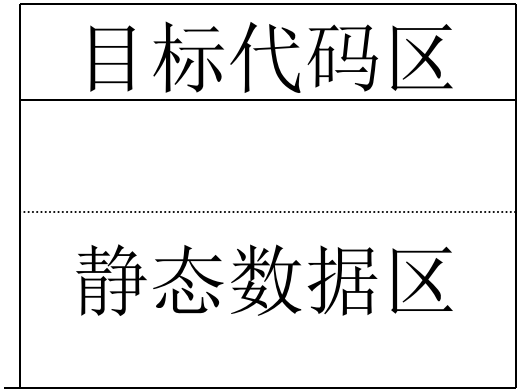
FORTRAN程序的特点:

1) 每个数据名所需的存储空间大小都是常数.(不含可变数据)

2)每个数据名的性质是完全确定的(不含运行时动态确定性质的名字).

3)在编译时整个程序所需空间的总量是
确定的.

数据区----程序段,公用块. 构造存储映象.



局部数据区的组成 (P246)

对公用语句建立公用链 和 对等价语句建立等价环.(P247-251)*

第8章:运行时存储空间组织

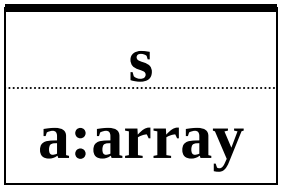
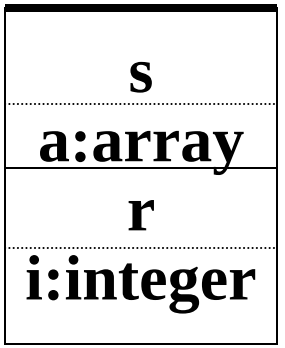
- 8.1 源语言中问题的讨论
- 8.2 存储器的组织
- 8.3 静态存储分配
- 8.4栈式存储分配
- 8.5嵌套式过程语言栈式实现
- 8.6 堆式动态存储分配

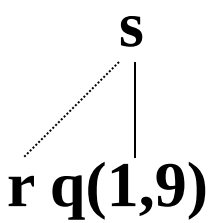
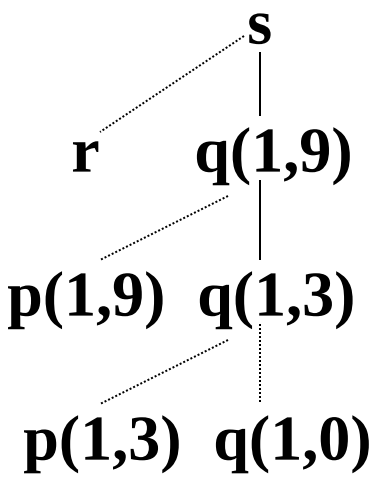
8.4 栈式存储分配

使用栈式存储分配法意味着,运行时每当进入一个过程就有一个相应的活动纪录放在栈顶.该纪录含有:连接数据,形式单元,局部变量,局部数组的内情项量和临时工作单元. 当一个工作完毕返回时,它在栈顶上的数据区的局部名的值也消失.

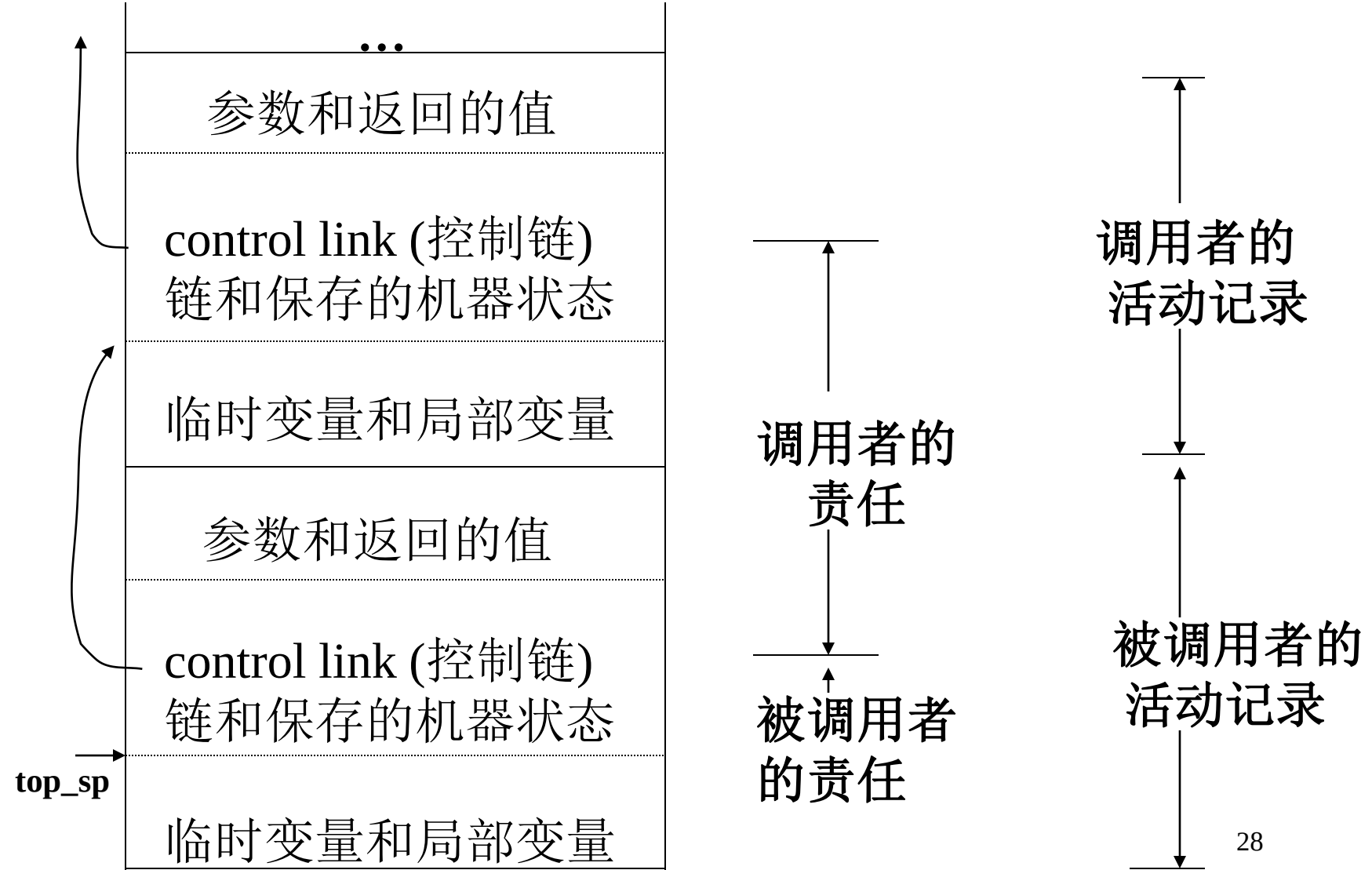
C语言程序的存储组织(P255-256)

例如:sort程序的一个活动记录被推入活动或弹出运行时刻的栈中的情况.

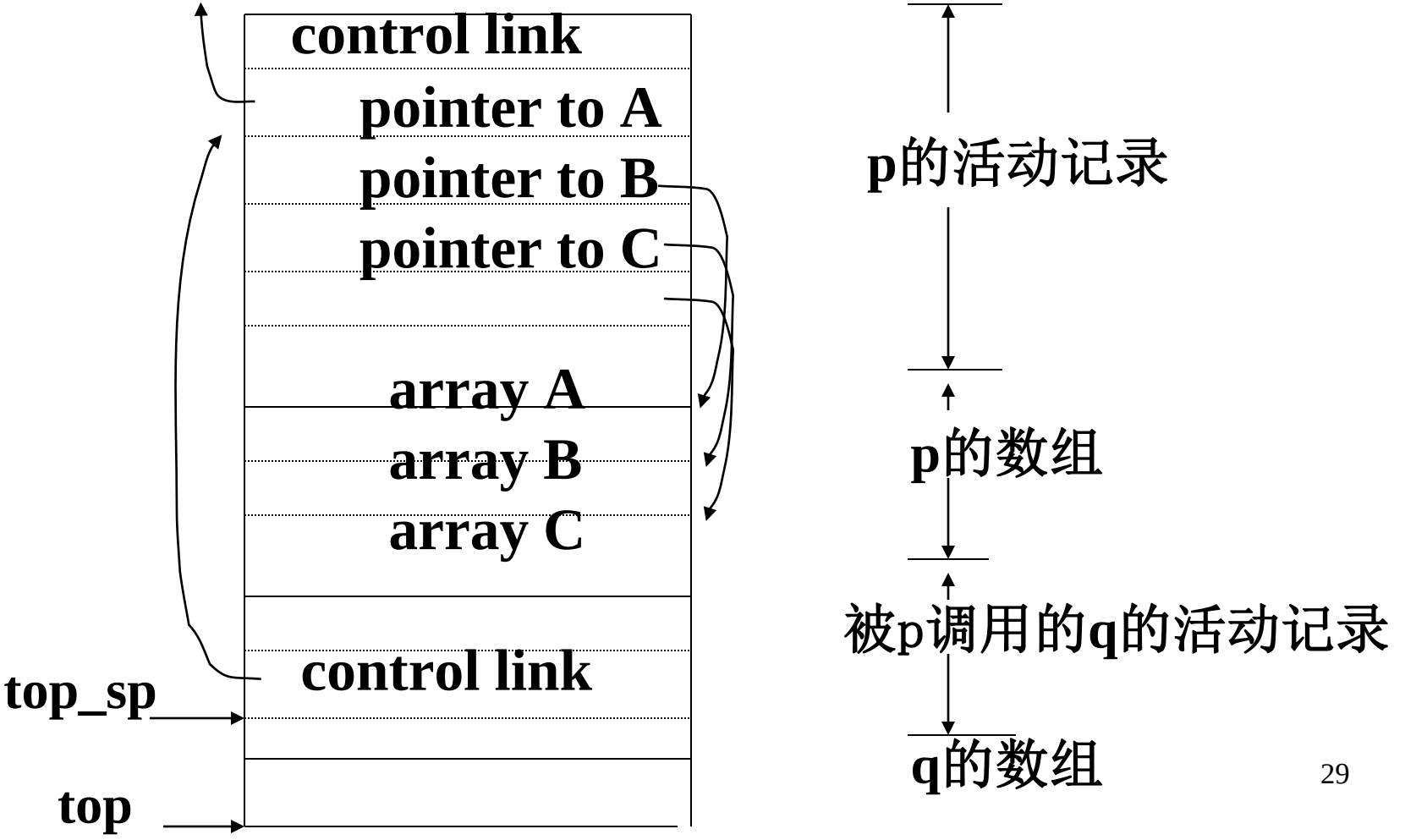
在活动树中的位置	在栈中的活动记录	附注
s		**s的活动记录**
s **r**		**r被激活**

在活动树中的位置	在栈中的活动记录	附注
	S a:array q(1,9) k:integer	r的活动记录 已被弹出， q(1, 9)被压入栈中
	S a:array q(1,9) k:integer q(1,3) k:integer	控制刚刚返回 到q(1,3)

调用序列(调用与被调用的划分)



可变长度的数据(动态分配的数组的存储)



第8章:运行时存储空间组织

- 8.1 源语言中问题的讨论
- 8.2 存储器的组织
- 8.3 静态存储分配
- 8.4栈式存储分配
- 8.5嵌套式过程语言栈式实现
- 8.6 堆式动态存储分配

8.5嵌套式过程语言栈式实现

1）程序举例（p258图9.15 ）程序

2）非局部名的访问实现

静态链和活动记录

（p259 图9.16 ）活动记录结构

Display和活动记录

过程递归调用时活动记录的变化

（p260 图9.17)

程序运行时刻访问的Display表的内容

（p262-263 图9.19)

8.6 堆式动态存储分配

可以由用户申请或释放存储的语言
不仅有过程而且有进程的程序结构
例： new( )/dispose( )

堆式动态存储分配/回收：

参见数据结构中的算法