中国科学技术大学
University of Science and Technology of China

USTC, CHINA
ADSLAB
先进数据系统实验室

《编译原理与技术》
运行时存储空间的组织和管理

计算机科学与技术学院
李 诚
29/11/2018

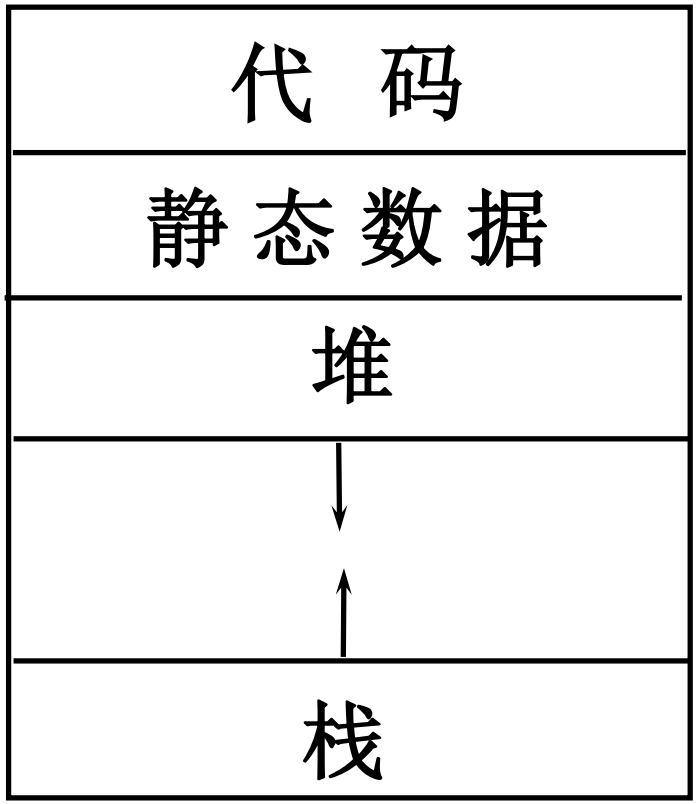
❑术语

- ❖过程的活动(activation): 过程的一次执行
- ❖活动记录：过程的活动需要可执行代码和存放所需信息的存储空间,后者称为活动记录

❑本章内容

- ❖一个活动记录中的数据布局
- ❖程序执行过程中，所有活动记录的组织方式
- ❖非局部名字的管理、参数传递方式、堆管理

程序的存储分配



影响存储分配策略的语言特征

中国科学技术大学
University of Science and Technology of China

- 过程能否递归
- 当控制从过程的活动返回时, 局部变量的值是否要保留
- 过程能否访问非局部变量
- 过程调用的参数传递方式
- 过程能否作为参数被传递
- 过程能否作为结果值传递
- 存储块能否在程序控制下被动态地分配
- 存储块是否必须被显式地释放

过程的存储组织与分配

- 过程
  - FORTRAN的子例程(subroutine)
  - PASCAL的过程/函数(procedure/function)
  - C的函数
- 过程的激活（调用）与终止（返回）
- 过程的执行需要：

  代码段 + 活动记录（过程运行所需的额外信息，如参数，局部数据，返回地址等）

局部存储分配

- 基本概念：作用域与生存期
- 活动记录的常见布局
  - 字节寻址、类型、次序、对齐
- 程序块：同名情况的处理

❑名字的作用域

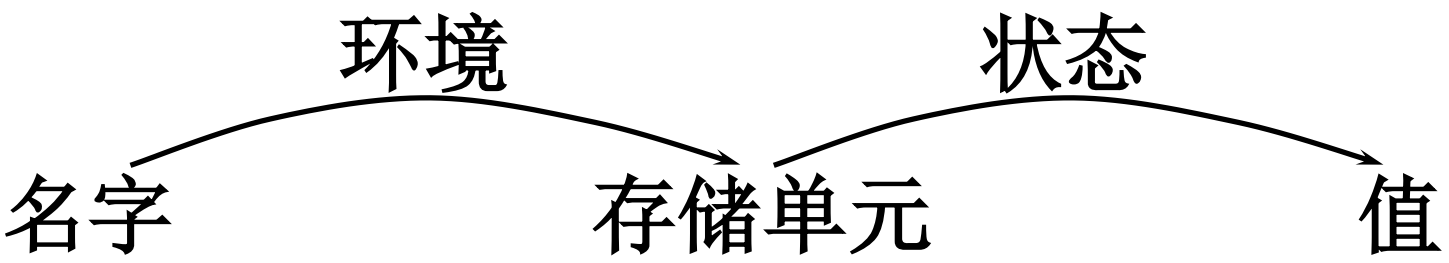
- ❖一个声明起作用的程序部分称为该声明的作用域
- ❖即使一个名字在程序中只声明一次，该名字在程序运行时也可能表示不同的数据对象

如下图代码中的**n**

```
int f(int n){
if (n<0) error("arg<0");
else if (n==0) return 1;
else return n*f(n-1);
}
```

❑环境和状态

- ❖环境把名字映射到左值，而状态把左值映射到右值（即名字到值有两步映射）
- ❖赋值改变状态，但不改变环境
- ❖过程调用改变环境
- ❖如果环境将名字x映射到存储单元s，则说x被绑定到s



❑静态概念和动态概念的对应

静 态 概 念	动 态 对 应
过程的定义	过程的活动
名字的声明	名字的绑定
声明的作用域	绑定的生存期

❑活动记录的常见布局

❑局部数据的布局

- ❖字节是可编址内存的最小单位
- ❖变量所需的存储空间可以根据其类型而静态确定
- ❖一个过程所声明的局部变量，按这些变量声明时出现的次序，在局部数据域中依次分配空间
- ❖局部数据的地址可以用相对于活动记录中某个位置的地址来表示
- ❖数据对象的存储布局还有一个对齐问题

局部存储分配

❑例　在SPARC/Solaris工作站上下面两个结构体的size分别是24和16，为什么不一样?

```
typedef struct  _a{
        char   c1;
        long   i;
        char   c2;
        double f;
}a;
```

```
typedef struct  _b{
        char c1;
        char   c2;
        long i;
        double f;
}b;
```

对齐：char : 1, long : 4, double : 8

局部存储分配

❑例　在SPARC/Solaris工作站上下面两个结构体的size分别是24和16，为什么不一样?

```
typedef struct  _a{
        char   c1;     0
        long   i;      4
        char   c2;     8
        double f;      16
}a;
```

```
typedef struct  _b{
        char c1;      0
        char   c2;    1
        long i;       4
        double f;     8
}b;
```

对齐：char : 1, long : 4, double : 8

局部存储分配

❑例　在X86/Linux机器的结果和SPARC/Solaris工作站不一样，是20和16。

```
typedef struct  _a{
        char   c1;     0
        long   i;      4
        char   c2;     8
        double f;      12
}a;
```

```
typedef struct  _b{
        char c1;      0
        char   c2;    1
        long i;       4
        double f;     8
}b;
```

对齐：char : 1, long : 4, double : 4

❑程序块

- ❖本身含有局部变量声明的语句
- ❖可以嵌套
- ❖最接近的嵌套作用域规则
- ❖并列程序块不会同时活跃
- ❖并列程序块的变量可以重叠分配

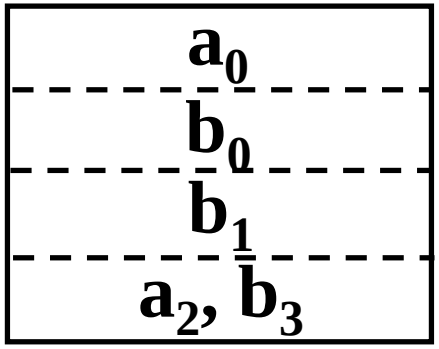
局部存储分配


```
main() / * 例 */
{                                                    /* begin of B0 */
    int a = 0;
    int b = 0;
    {                                                    /* begin of B1 */
        int b = 1;
        {                                            /* begin of B2 */
            int a = 2;
        }                                            /* end of B2 */
        {                                            /* begin of B3 */
            int b = 3;
        }                                            /* end of B3 */
    }                                                    /* end of B1 */
}                                                        /* end of B0 */
```

局部存储分配


```
main() / * 例 */
{ /* begin of B0 */
    int a = 0;
    int b = 0;
    { /* begin of B1 */
        int b = 1;
        {/* begin of B2 */
            int a = 2;
        }/* end of B2 */
        {/* begin of B3 */
            int b = 3;
        }/* end of B3 */
    }/* end of B1 */
}/* end of B0 */
```

声　明	作 用 域
int a = 0;	$B_0 - B_2$
int b = 0;	$B_0 - B_1$
int b = 1;	$B_1 - B_3$
int a = 2;	B_2
int b = 3;	B_3



重叠分配存储单元

- 名字在程序被编译时绑定到存储单元，不需要运行时的任何支持
- 绑定的生存期是程序的整个运行期间

❑静态分配给语言带来限制

- ❖递归过程不被允许
- ❖数据对象的长度和它在内存中位置的限制，必须是在编译时可以知道的
- ❖数据结构不能动态建立

- 例 C程序的外部变量、静态局部变量以及程序中出现的常量都可以静态分配
- 声明在函数外面
  - 外部变量 —— 静态分配
  - 静态外部变量 —— 静态分配
- 声明在函数里面
  - 静态局部变量 —— 也是静态分配
  - 自动变量 —— 不能静态分配

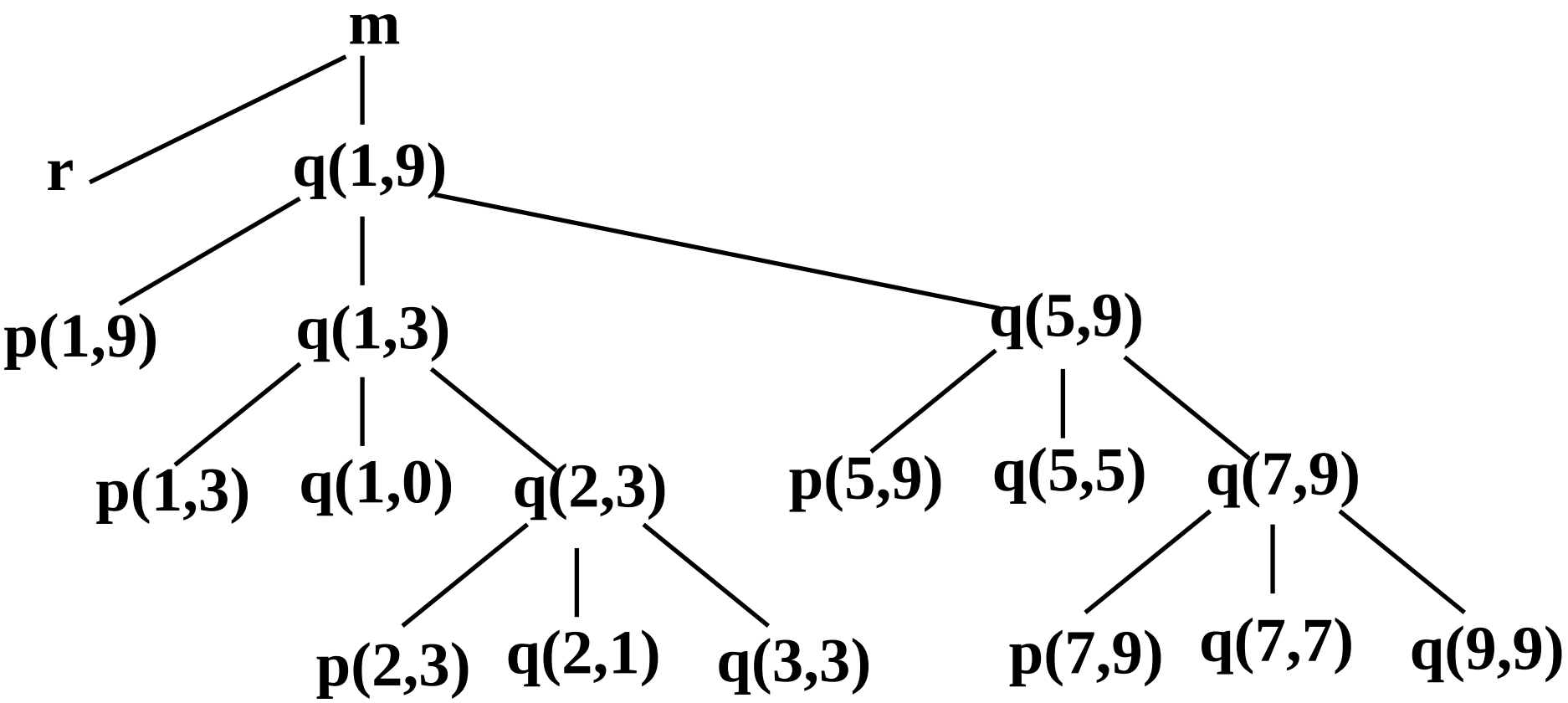
- 主要有两种策略
  - 栈式存储：与过程调用返回有关，涉及过程的局部变量
  - 堆存储：关系到部分生存周期较长的数据

全局栈式存储分配

- 介绍程序运行时所需的各个活动记录在存储空间的分配策略
- 描述过程的目标代码怎样访问绑定到局部名字的存储单元

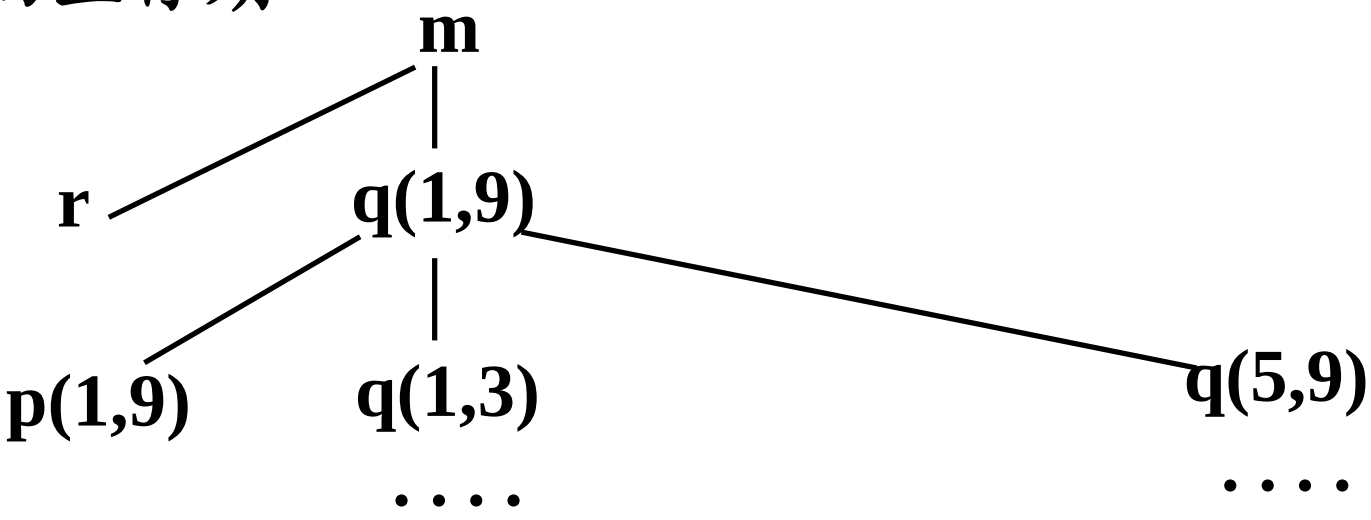
❑活动树

❖用树来描绘控制进入和离开活动的方式



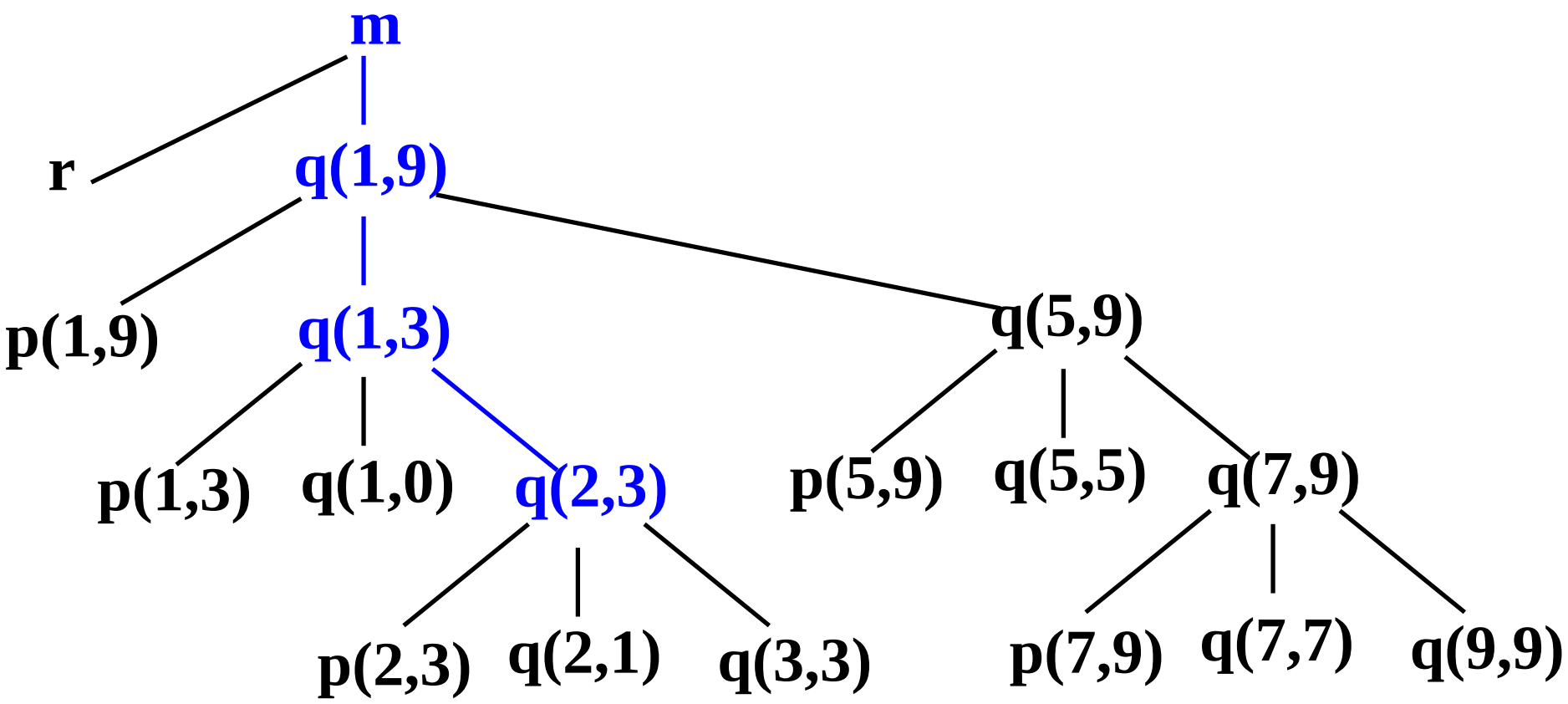
❑活动树的特点

- ❖每个结点代表某过程的一个活动
- ❖根结点代表主程序的活动
- ❖结点*a*是结点*b*的父结点，当且仅当控制流从*a*的活动进入*b*的活动
- ❖结点*a*处于结点*b*的左边，当且仅当*a*的生存期先于*b*的生存期



活动树和运行栈

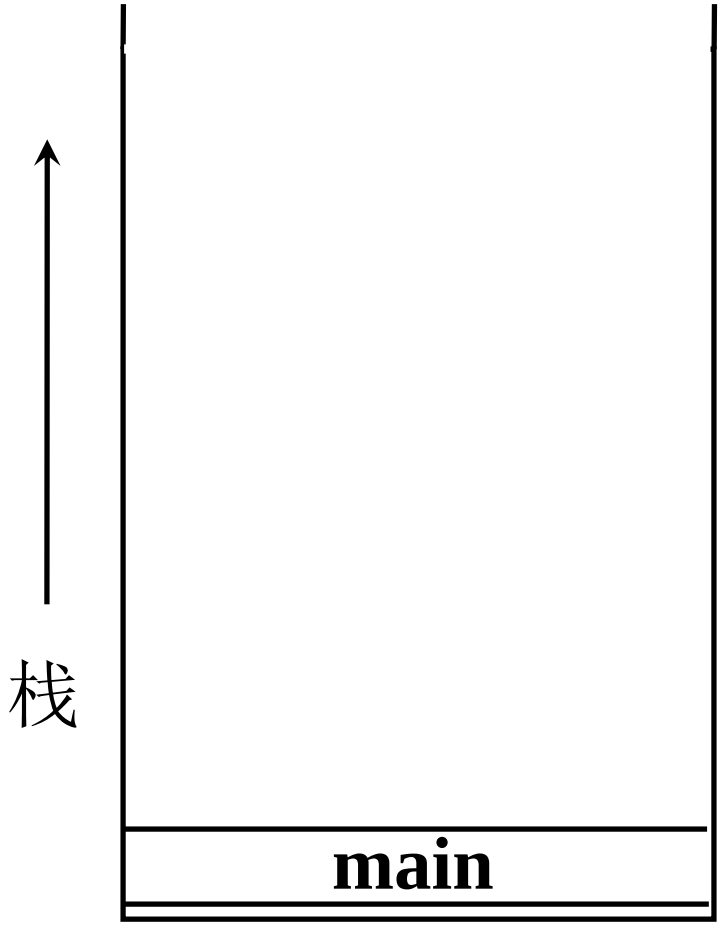
- 当前活跃着的过程活动可以保存在一个栈中
  - 例 控制栈的内容：m, q (1, 9), q (1, 3), q (2, 3)



- 运行栈：把控制栈中的信息拓广到包括过程活动所需的所有局部信息（即活动记录）

活动树和运行栈

- 运行栈：把控制栈中的信息拓广到包括过程活动所需的所有局部信息（即活动记录）

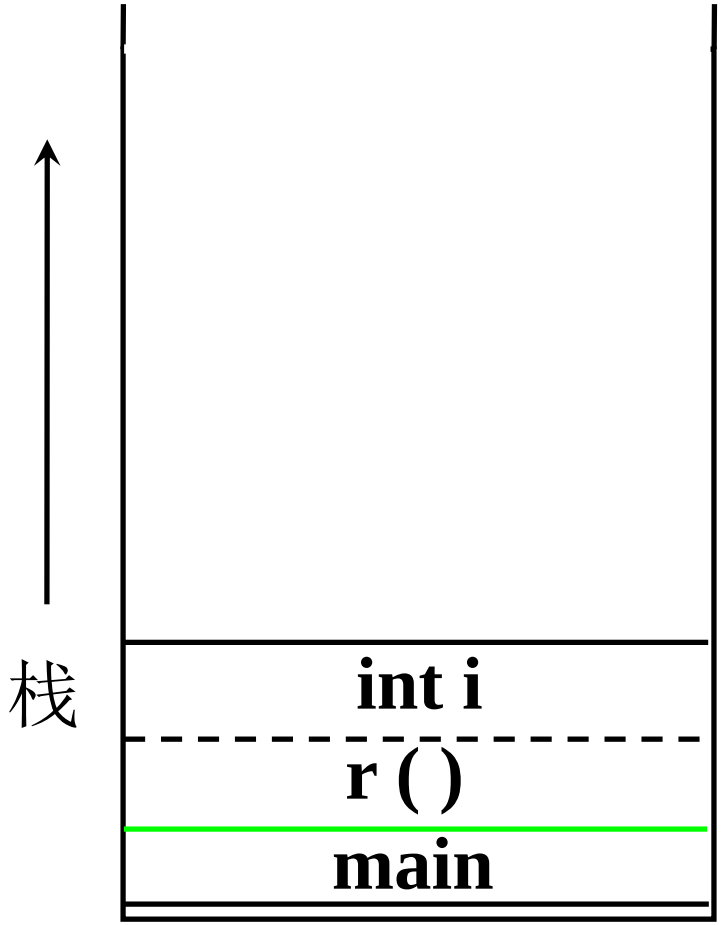


函数调用关系树

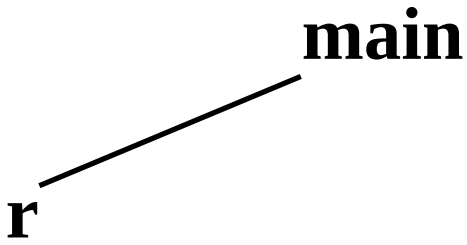
main

活动树和运行栈

❑运行栈：把控制栈中的信息拓广到包括过程活动所需的所有局部信息（即活动记录）

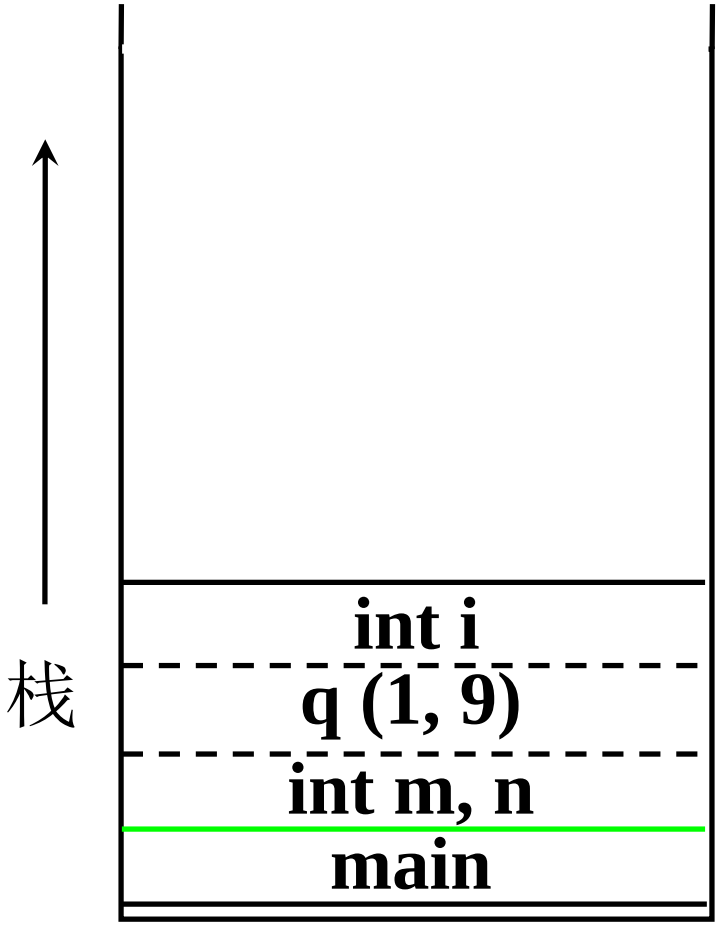


函数调用关系树

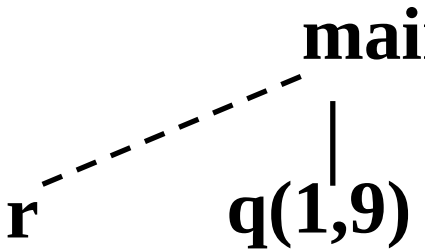


活动树和运行栈

- 运行栈：把控制栈中的信息拓广到包括过程活动所需的所有局部信息（即活动记录）

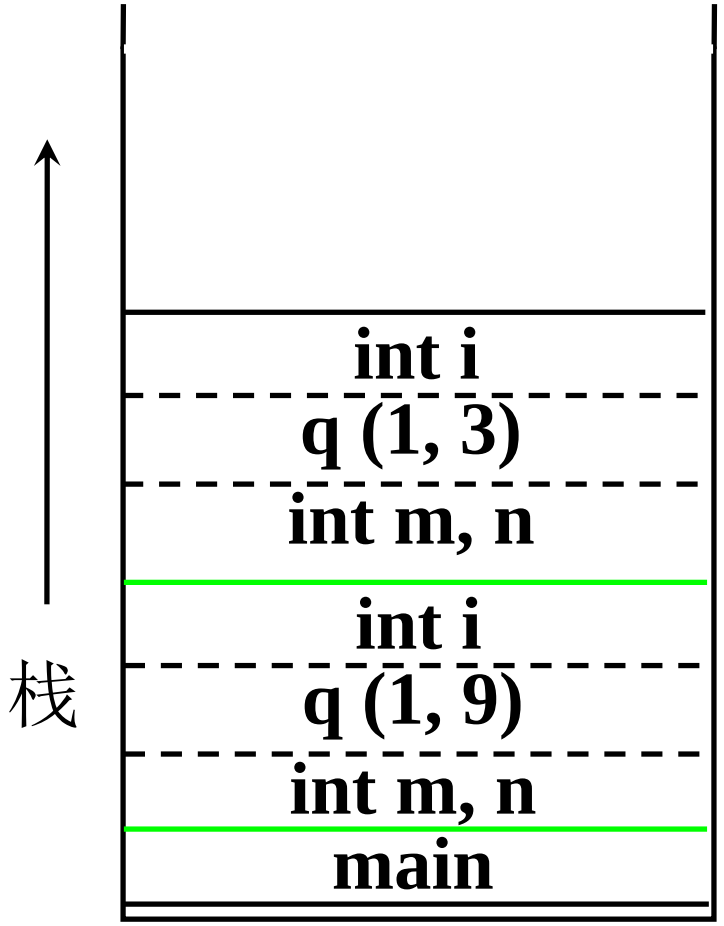


函数调用关系树

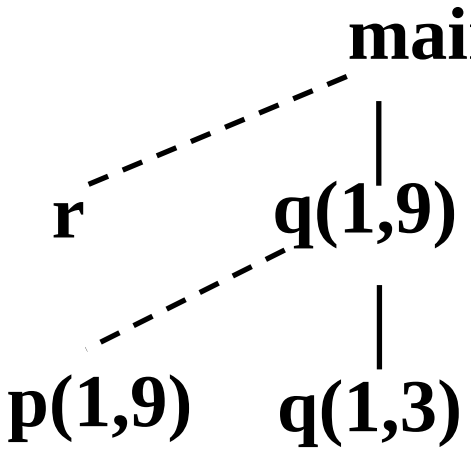


活动树和运行栈

❑运行栈：把控制栈中的信息拓广到包括过程活动所需的所有局部信息（即活动记录）

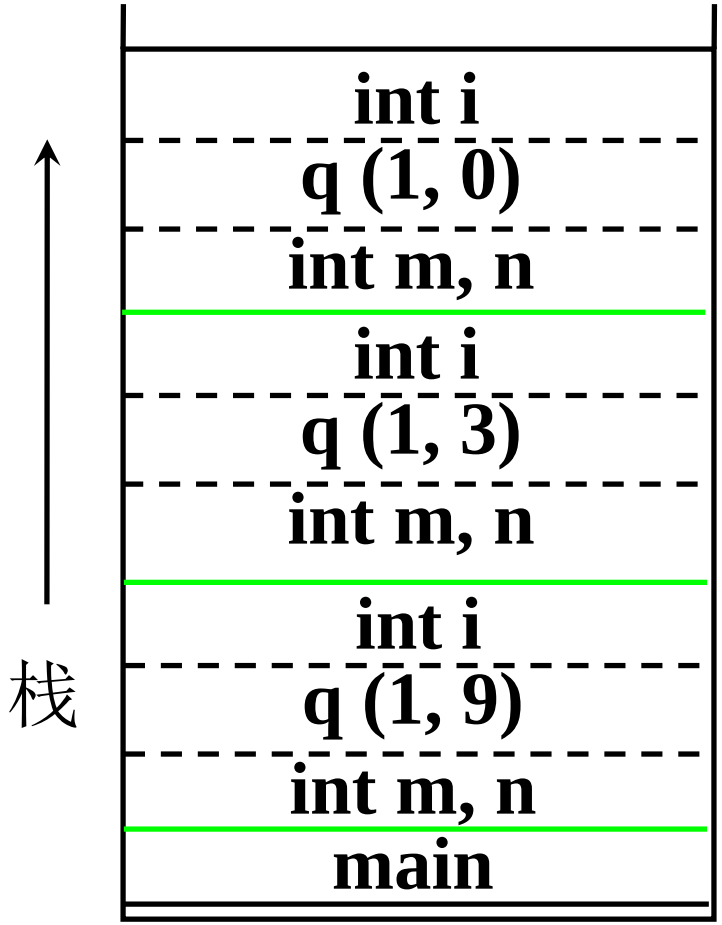


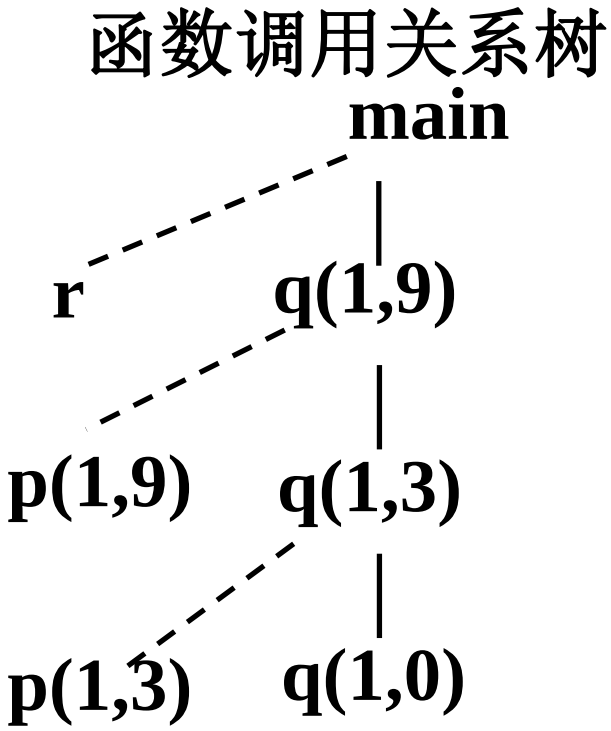
函数调用关系树



活动树和运行栈

❑运行栈：把控制栈中的信息拓广到包括过程活动所需的所有局部信息（即活动记录）





中国科学技术大学
University of Science and Technology of China
USTC, CHINA
ADSLAB
先进数据系统实验室

《编译原理与技术》
运行时存储空间的组织和管理

TBA