第6讲 运行时刻环境

学习的主要内容和目标

➢ 学习的主要内容

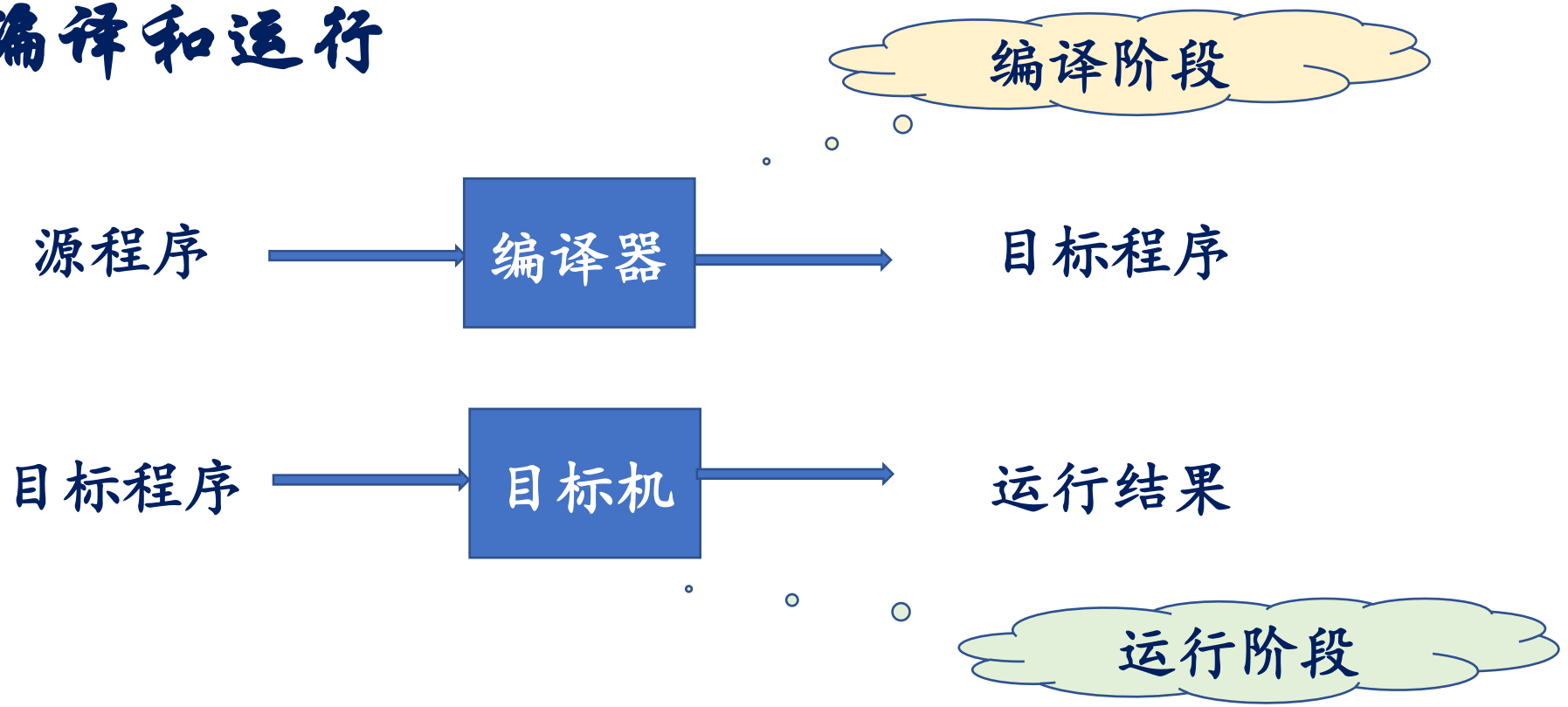
- ✧ 运行时刻环境的一般结构
- ✧ 存储分配策略
- ✧ 活动记录

➢ 学习的目标

- ✧ 掌握活动记录的基本设计方法
- ✧ 掌握非局部变量的访问方法

概述

➢ 编译和运行



概述

➢ 运行时刻环境

- ✧ 将程序和运行时的活动联系起来
- ✧ 决定程序运行时的大部分内存布局
- ✧ 生成创建和管理程序运行时刻环境所需的代码

概述

➢ 关键问题

- ✧ 实现源程序中的各种抽象概念
  - ✓ 名字、作用域、数据类型、过程、参数、控制流
- ✧ 如何安排目标机资源的使用
  - ✓ 内存/缓存、寄存器、操作系统资源

存储组织

- 程序运行时的内存映像
  - 目标程序运行在一个逻辑存储空间

存储组织

- 程序运行时的内存映像
  - 典型的程序布局
    - ✓ 代码区
    - ✓ 静态数据区
      - 静态存放全局数据
    - ✓ 动态数据段
      - 运行时动态变化的堆区和栈区

存储组织

➢ 静态分配

- ✧ 在编译期间为数据对象分配存储
- ✧ 在编译期间就可确定数据对象的大小：不宜处理递归过程或函数
  - ✓ 某些语言中所有存储都是静态分配：如汇编语言，FORTRAN语言
  - ✓ 数语言只有部分存储进行静态分配
    - 可静态分配的数据对象如大小固定且在程序执行期间
    - 可全程访问的全局变量，以及程序中的常量：如 C 语言中的 static 和 extern 变量

存储组织

➢动态存储分配

✧ 栈式分配

- ✓ 将数据对象的运行时存储按照栈的方式来管理
- ✓ 用于有效实现层次嵌套的程序结构： 如实现过程/函数，块层次结构
- ✓ 可以实现递归过程/函数：比较：静态分配不宜实现递归过程/函数

✧ 堆式分配

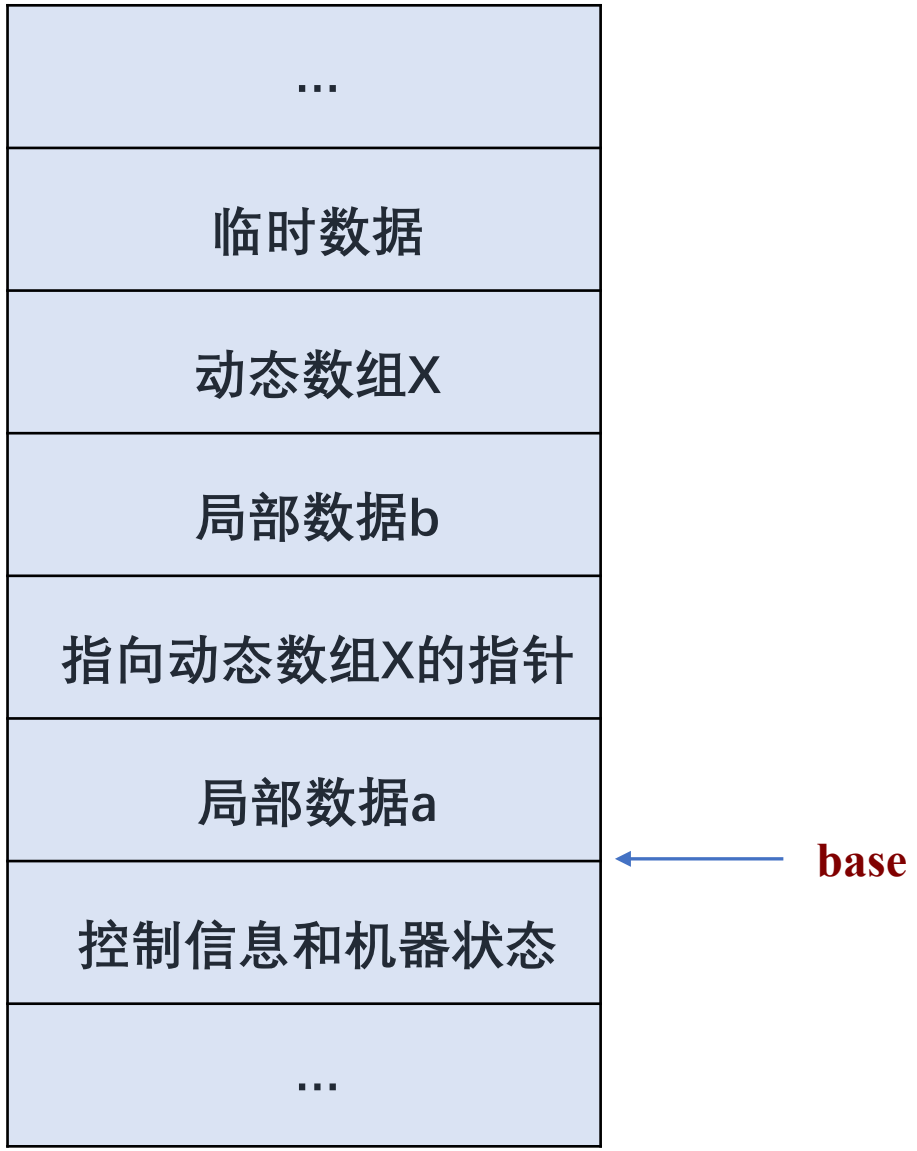
- ✓ 从数据段的堆空间分配和释放数据对象的运行时存储

活动记录

- 活动记录的一般结构
  - 用于存储过程一次执行所需机器状态信息，以及生命周期包含在一次活动中的数据对象。
- 数据的访问
  - 地址=活动记录的起始地址+偏移地址

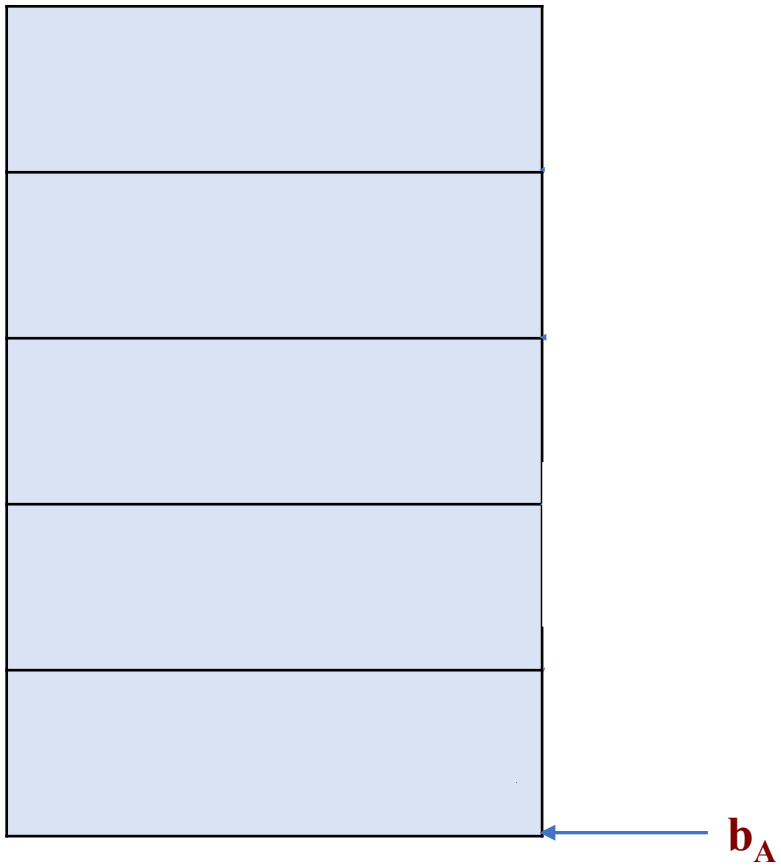
活动记录

- 变长数据的分配
  - ✧ 先分配静态数据
  - ✧ 动态数组先分配指针



基于栈的过程管理

➢过程调用和返回




```
(1)program A;
(2)    var x,y: integer;
(3)    procedure B;
(4)        var z: integer;
(5)        begin
(6)            z:=1;
(7)            x:=z+1
(8)        end;
(9)    procedure C;
(10)       var y: integer;
(11)           procedure D;
(12)               var y: integer;
(13)               begin
(14)                   B
(15)               end;
(16)       begin
(17)           D;
(18)           y:=x+1;
(19)           B
(20)       end;
(21)begin
(22)    C;
(23)    y:=x+1
(24)end.
```

基于栈的过程管理

➢过程调用和返回


```
(1)program A;
(2)    var x,y: integer;
(3)    procedure B;
(4)        var z: integer;
(5)        begin
(6)            z:=1;
(7)            x:=z+1
(8)        end;
(9)    procedure C;
(10)       var y: integer;
(11)           procedure D;
(12)               var y: integer;
(13)               begin
(14)                   B
(15)               end;
(16)       begin
(17)           D;
(18)           y:=x+1;
(19)           B
(20)       end;
(21)begin
(22)    C;
(23)    y:=x+1
(24)end.
```

基于栈的过程管理

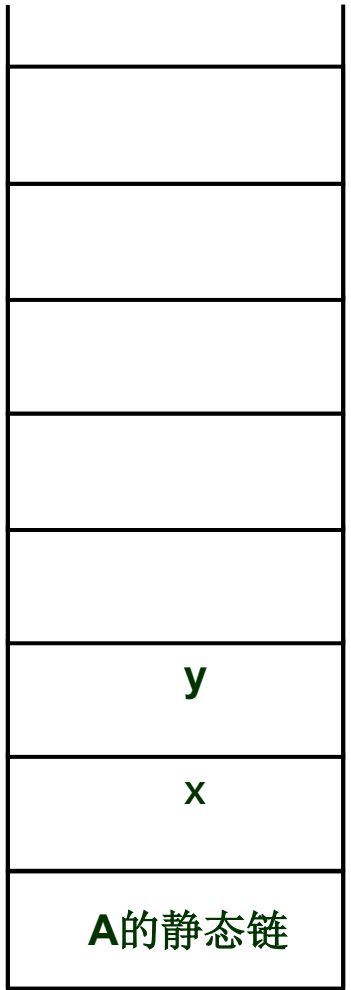
- 静态数组的访问
  - ✧ *A*[*low..up, low..up*]
  - ✧ *A*[*i,j*]
  - ✧ *X*:=*A*[*3,5*]+*y*

非局部变量的访问

- 无嵌套过程
  - 静态分配
- 过程嵌套定义
  - 访问链
  - 显示表

非局部变量的访问

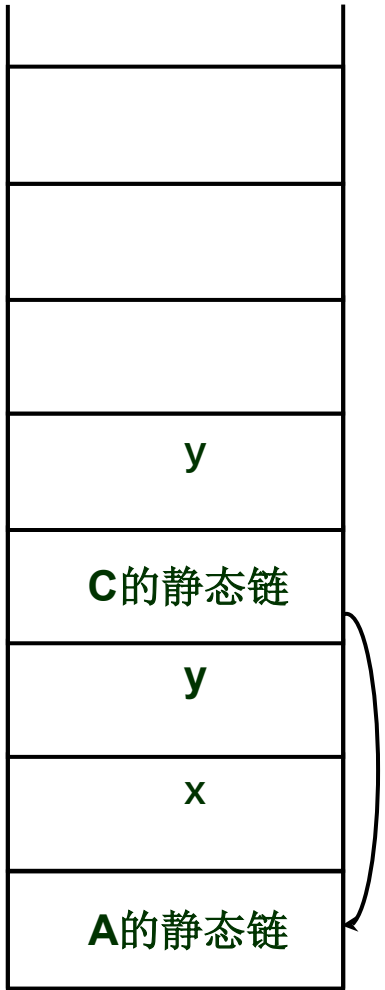
- 过程嵌套定义
  - ✧ 访问链




```
(1)program A;
(2)    var x,y: integer;
(3)    procedure B;
(4)        var z: integer;
(5)        begin
(6)            z:=1;
(7)            x:=z+1
(8)        end;
(9)    procedure C;
(10)       var y: integer;
(11)           procedure D;
(12)               var y: integer;
(13)               begin
(14)                   B
(15)               end;
(16)       begin
(17)           D;
(18)           y:=x+1;
(19)           B
(20)       end;
(21)begin
(22)    C;
(23)    y:=x+1
(24)end.
```

非局部变量的访问

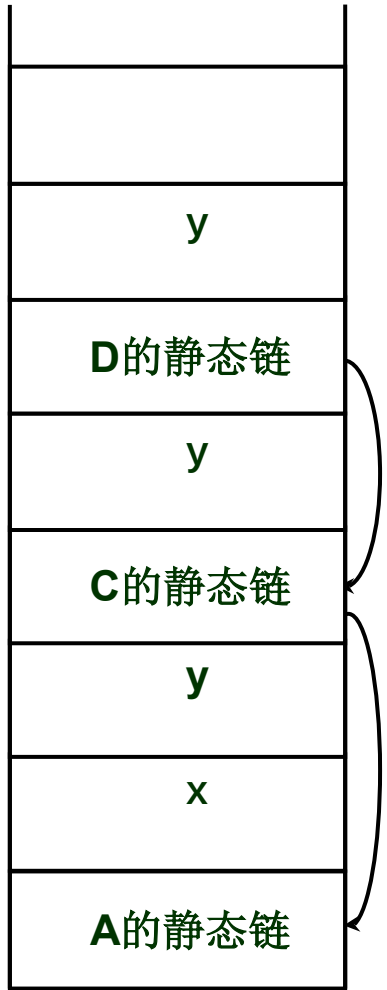
- 过程嵌套定义
  - ✧ 访问链




```
(1)program A;
(2)    var x,y: integer;
(3)    procedure B;
(4)        var z: integer;
(5)        begin
(6)            z:=1;
(7)            x:=z+1
(8)        end;
(9)    procedure C;
(10)       var y: integer;
(11)           procedure D;
(12)               var y: integer;
(13)               begin
(14)                   B
(15)               end;
(16)       begin
(17)           D;
(18)           y:=x+1;
(19)           B
(20)       end;
(21)begin
(22)    C;
(23)    y:=x+1
(24)end.
```

非局部变量的访问

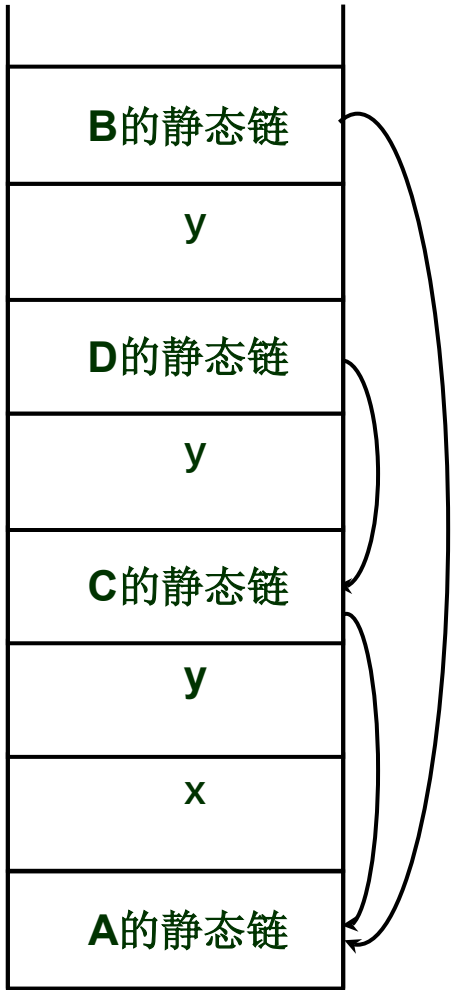
- 过程嵌套定义
  - ✧ 访问链




```
(1)program A;
(2)    var x,y: integer;
(3)    procedure B;
(4)        var z: integer;
(5)        begin
(6)            z:=1;
(7)            x:=z+1
(8)        end;
(9)    procedure C;
(10)       var y: integer;
(11)           procedure D;
(12)               var y: integer;
(13)               begin
(14)                   B
(15)               end;
(16)       begin
(17)           D;
(18)           y:=x+1;
(19)           B
(20)       end;
(21)begin
(22)    C;
(23)    y:=x+1
(24)end.
```

非局部变量的访问

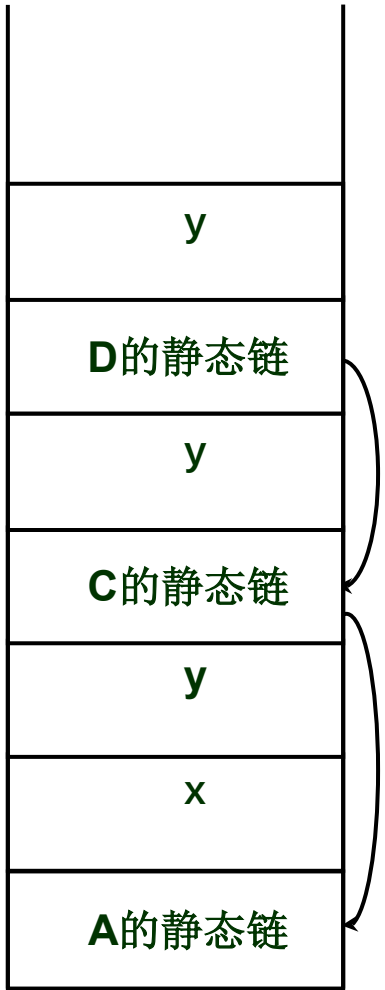
- 过程嵌套定义
  - ✧ 访问链




```
(1)program A;
(2)    var x,y: integer;
(3)    procedure B;
(4)        var z: integer;
(5)        begin
(6)            z:=1;
(7)            x:=z+1
(8)        end;
(9)    procedure C;
(10)       var y: integer;
(11)           procedure D;
(12)               var y: integer;
(13)               begin
(14)                   B
(15)               end;
(16)       begin
(17)           D;
(18)           y:=x+1;
(19)           B
(20)       end;
(21)begin
(22)    C;
(23)    y:=x+1
(24)end.
```

非局部变量的访问

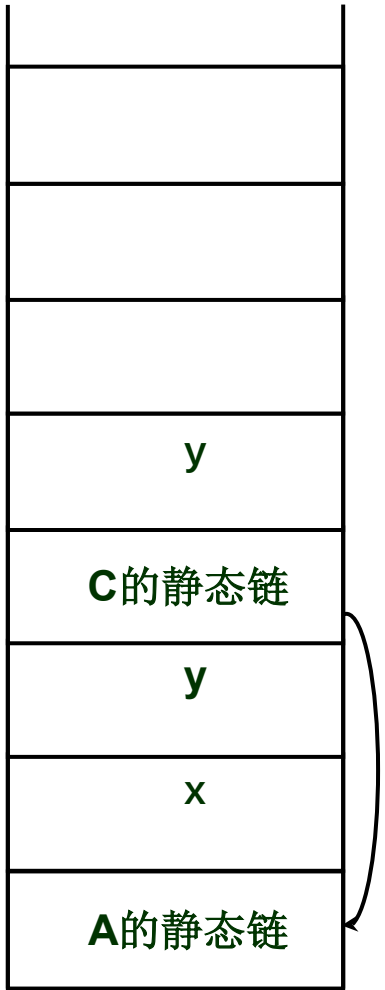
- 过程嵌套定义
  - ✧ 访问链




```
(1)program A;
(2)    var x,y: integer;
(3)    procedure B;
(4)        var z: integer;
(5)        begin
(6)            z:=1;
(7)            x:=z+1
(8)        end;
(9)    procedure C;
(10)       var y: integer;
(11)           procedure D;
(12)               var y: integer;
(13)               begin
(14)                   B
(15)               end;
(16)       begin
(17)           D;
(18)           y:=x+1;
(19)           B
(20)       end;
(21)begin
(22)    C;
(23)    y:=x+1
(24)end.
```

非局部变量的访问

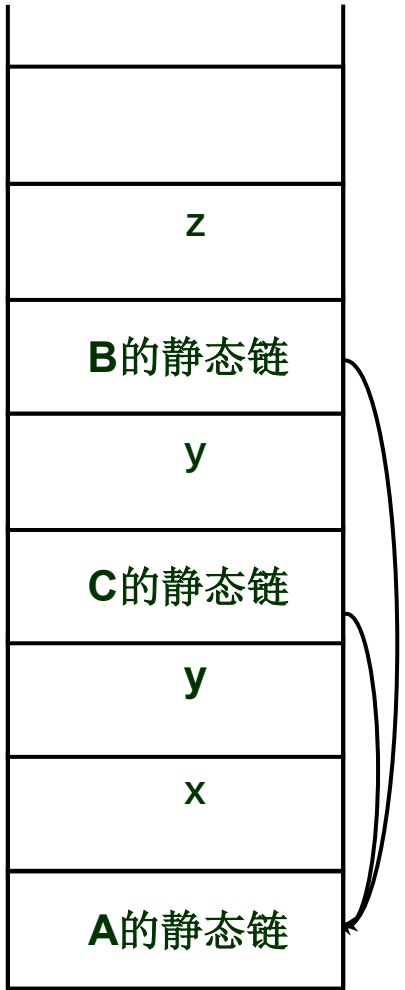
- 过程嵌套定义
  - ✧ 访问链




```
(1)program A;
(2)    var x,y: integer;
(3)    procedure B;
(4)        var z: integer;
(5)        begin
(6)            z:=1;
(7)            x:=z+1
(8)        end;
(9)    procedure C;
(10)       var y: integer;
(11)           procedure D;
(12)               var y: integer;
(13)               begin
(14)                   B
(15)               end;
(16)       begin
(17)           D;
(18)           y:=x+1;
(19)           B
(20)       end;
(21)begin
(22)    C;
(23)    y:=x+1
(24)end.
```

非局部变量的访问

- 过程嵌套定义
  - ✧ 访问链




```
(1)program A;
(2)    var x,y: integer;
(3)    procedure B;
(4)        var z: integer;
(5)        begin
(6)            z:=1;
(7)            x:=z+1
(8)        end;
(9)    procedure C;
(10)       var y: integer;
(11)           procedure D;
(12)               var y: integer;
(13)               begin
(14)                   B
(15)               end;
(16)       begin
(17)           D;
(18)           y:=x+1;
(19)           B
(20)       end;
(21)begin
(22)    C;
(23)    y:=x+1
(24)end.
```

基于栈的过程管理

- 过程间的值传递
  - 函数
    - ✓ 寄存器
    - ✓ 内存
  - 参数

基于栈的过程管理

- P10案例分析
  - ✧ 没有参数和函数
  - ✧ 有嵌套定义和调用

基于栈的过程管理

- GCC案例分析
  - ✧ 有参数和函数
  - ✧ 没有嵌套定义
  - ✧ 有嵌套调用

基于栈的过程管理

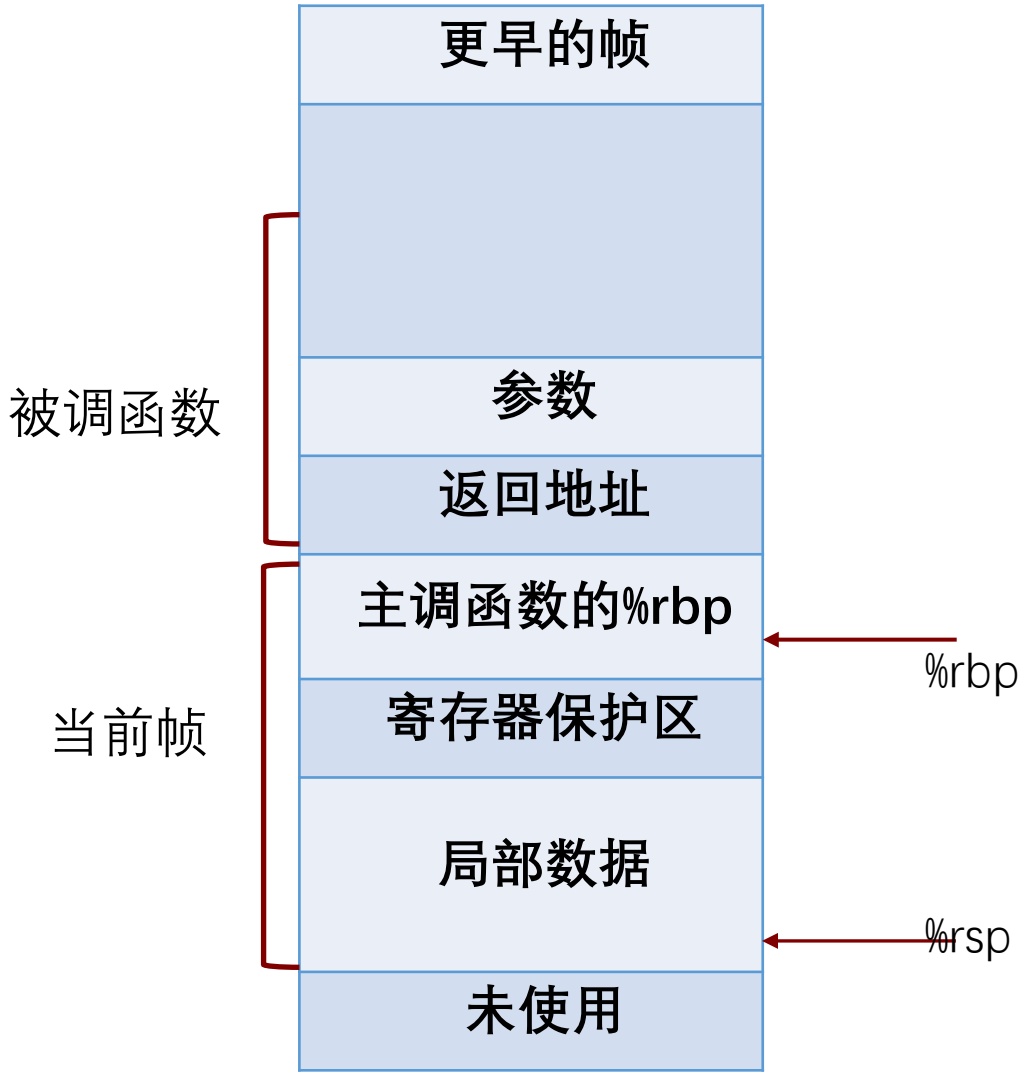
- GCC案例分析
  - ✧ 有参数和函数
  - ✧ 没有嵌套定义
  - ✧ 有嵌套调用

```
int sum(int a, int b)
{ int c;
    c=a+b;
    return c;
}

int main()
{ int x,y,z;
   x=3;
 y=4;
   z=sum(x,y);
   return 0;
}
```

基于栈的过程管理

- GCC案例分析
  - ✧ X86-64栈结构



基于栈的过程管理

➢GCC案例分析

✧ 参数和函数

```
int main()
{ int x,y,z;
  x=3;
 y=4;
  z=sum(x,y);
  return 0;
}
```

```
main:
        pushq  %rbp
        movq   %rsp, %rbp
        subq   $16, %rsp
        movl   $3, -12(%rbp)
        movl   $4, -8(%rbp)
        movl   -8(%rbp), %edx
        movl   -12(%rbp), %eax
        movl   %edx, %esi
        movl   %eax, %edi
        call   sum
        movl   %eax, -4(%rbp)
        movl   $0, %eax
        leave
        ret
```

基于栈的过程管理

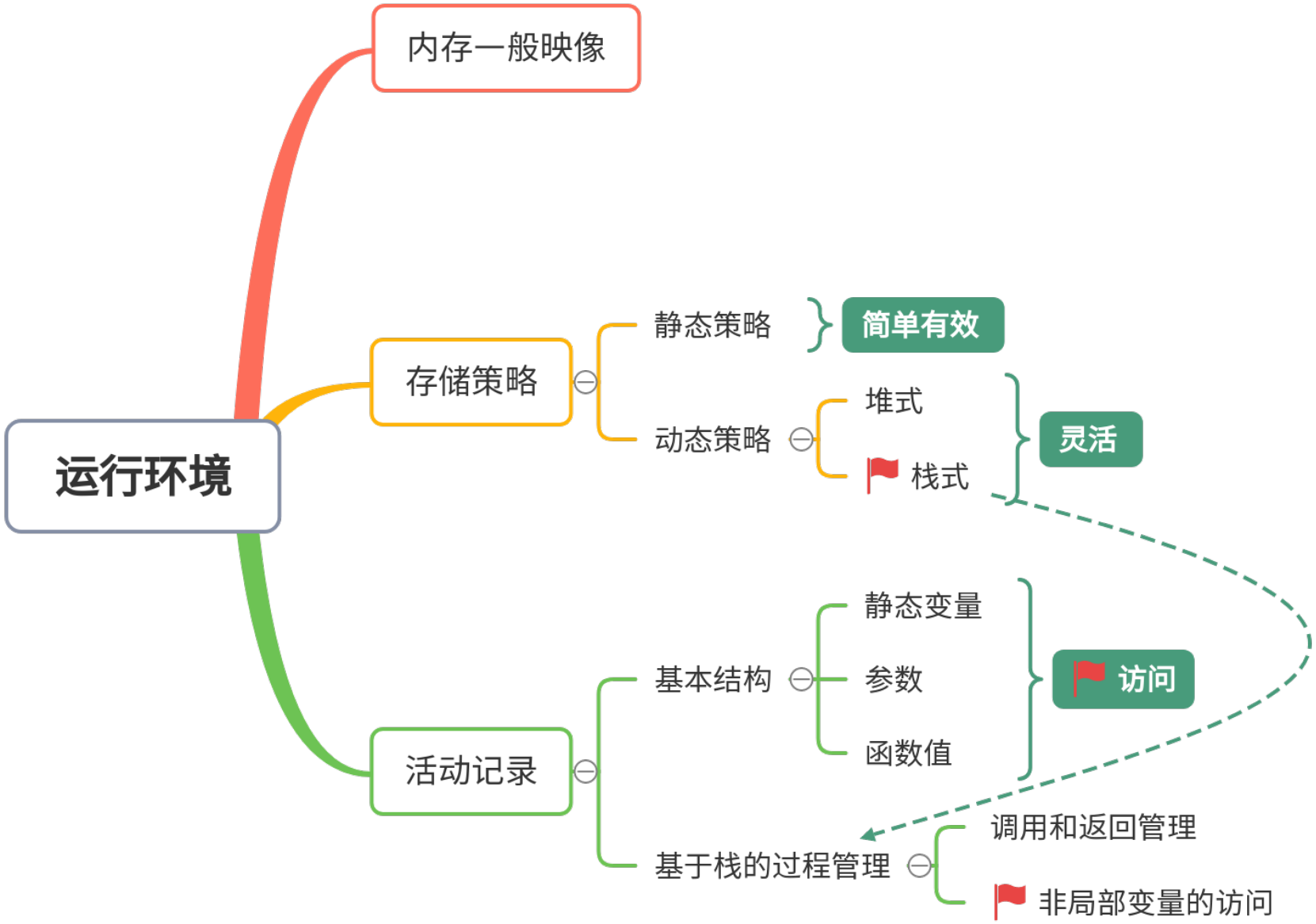
➢GCC案例分析

✧ 参数和函数

```
int sum(int a, int b)
{ int c;
   c=a+b;
   return c;
}
```

```
sum:
        pushq  %rbp
        movq   %rsp, %rbp
        movl   %edi, -20(%rbp)
        movl   %esi, -24(%rbp)
        movl   -20(%rbp), %edx
        movl   -24(%rbp), %eax
        addl   %edx, %eax
        movl   %eax, -4(%rbp)
        movl   -4(%rbp), %eax
        popq   %rbp
        ret
```

小结


运行环境
内存一般映像
存储策略
静态策略
简单有效
动态策略
堆式
栈式
灵活
活动记录
基本结构
静态变量
参数
函数值
访问
基于栈的过程管理
调用和返回管理
非局部变量的访问