编译系统

第七章
运行存储分配

哈尔滨工业大学 陈鄞

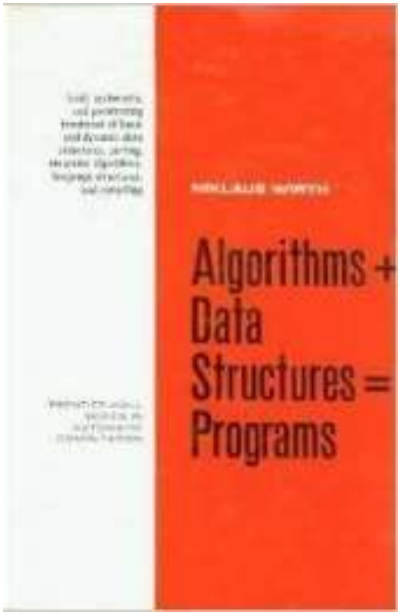
第15讲（运行存储分配）要点

- 7.1 存储组织 √
- 7.2 静态存储分配
- 7.3 栈式存储分配
- 7.4 非局部数据的访问 √
- 7.5 符号表 √

7.1 存储组织

- 一个目标程序运行所需的存储空间包括
  - 代码区
  - 数据区

Niklaus Wirth



算法 + 数据结构 = 程序

存储分配策略

- 静态存储分配
  - 顺序分配法
  - 层次分配法
- 动态存储分配
  - 栈式存储分配
  - 堆式存储分配

运行时内存的划分

7.4 非局部数据的访问

- 一个过程除了可以使用过程自身声明的局部数据以外，还可以使用过程外声明的非局部数据
  - 全局数据
  - 外围过程定义的数据（支持过程嵌套声明的语言）← 块结构语言的静态作用域规则
    - 例：*Pascal*语言

例：

- （*Pascal*语言）

```
program sort ( input, output );
    var a: array[0..10] of integer;
        x: integer;
    procedure readarray;
            var i: integer;
            begin … a … end {readarray} ;
    procedure exchange(i, j:integer);
            begin x=a[i];a[i]=a[j];a[j]=x; end {exchange} ;
    procedure quicksort(m, n:integer);
            var k, v : integer;
            function partition(y, z:integer):integer;
                    var i, j : integer;
                    begin … a … v … exchange(i, j) … end {partition};
            begin … a … v … partition … quicksort … end {quicksort} ;
    begin … a … readarray … quicksort … end {sort} ;
```

非局部数据的访问

- 一个过程除了可以使用过程自身声明的局部数据以外，还可以使用过程外声明的非局部数据
  - 全局数据
  - 外围过程定义的数据（支持过程嵌套声明的语言）
    - 例： *Pascal*语言　　*C*语言的程序块机制
- 如何访问非局部数据?
  - 访问链（静态链）

7.5 符号表

- 符号表是用于存放标识符的属性信息的数据结构
  - 种属 (Kind)
  - 类型 (Type)
  - 存储位置、长度
  - 作用域
  - 参数和返回值信息
- 符号表的作用
  - 辅助代码生成
  - 一致性检查

NAME	TYPE	KIND	VAL	ADDR
SIMPLE	整	简变		
SYMBLE	实	数组		
TABLE	字符	常数		
⋮	⋮	⋮	⋮	⋮

符号表上的主要操作

- 声明语句的翻译（定义性出现）
  - 填、查
- 可执行语句的翻译（使用性出现）
  - 查

单个过程符号表的组织

- 方法一：一张大表
  - 问题：不同种属的名字所需存放的属性信息在数量上的差异会造成符号表空间的浪费
    - 数组名：数组的维数、各维的长度
    - 过程名：参数个数、参数类型、返回值类型

NAME	TYPE	KIND	VAL	ADDR
SIMPLE	整	简变		
SYMBLE	实	数组		
TABLE	字符	常数		
⋮	⋮	⋮	⋮	⋮

单个过程符号表的组织

- 方法一：一张大表
  - 问题：不同种属的名字所需存放的属性信息在数量上的差异会造成符号表空间的浪费
- 方法二：多张子表（按种属分）
  - 变量表、数组表、过程表、…
  - 问题：为避免重名问题，插入或查找某个符号时需要查看所有的符号表，从而造成时间上的浪费
- 解决办法
  - 基本属性（直接存放在符号表中）+扩展属性（动态申请内存）

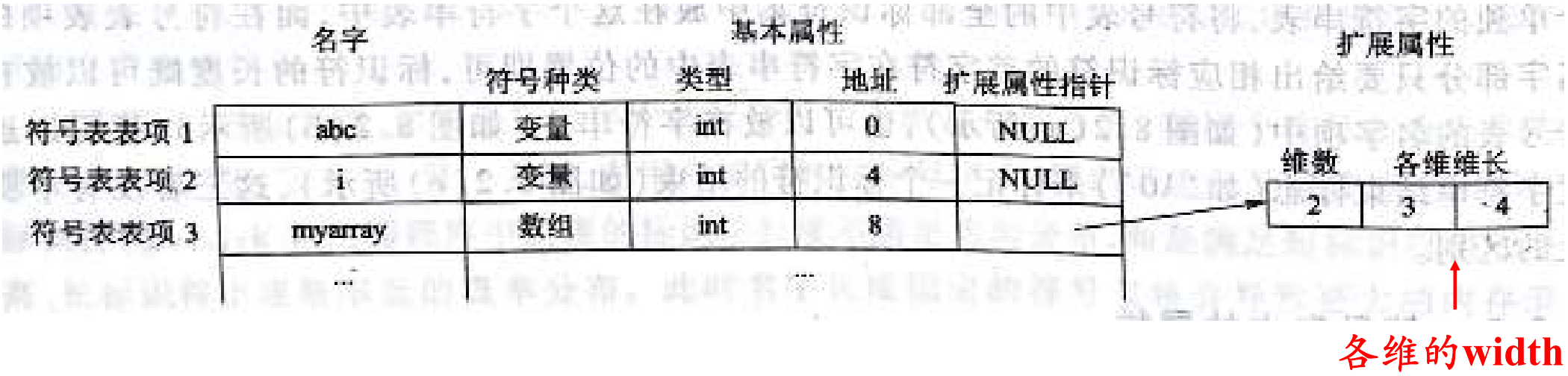
例

基本属性（直接存放在符号表中）+扩展属性（动态申请内存）

int abc;

int i;

char myarray[3][4];



多个过程符号表的组织

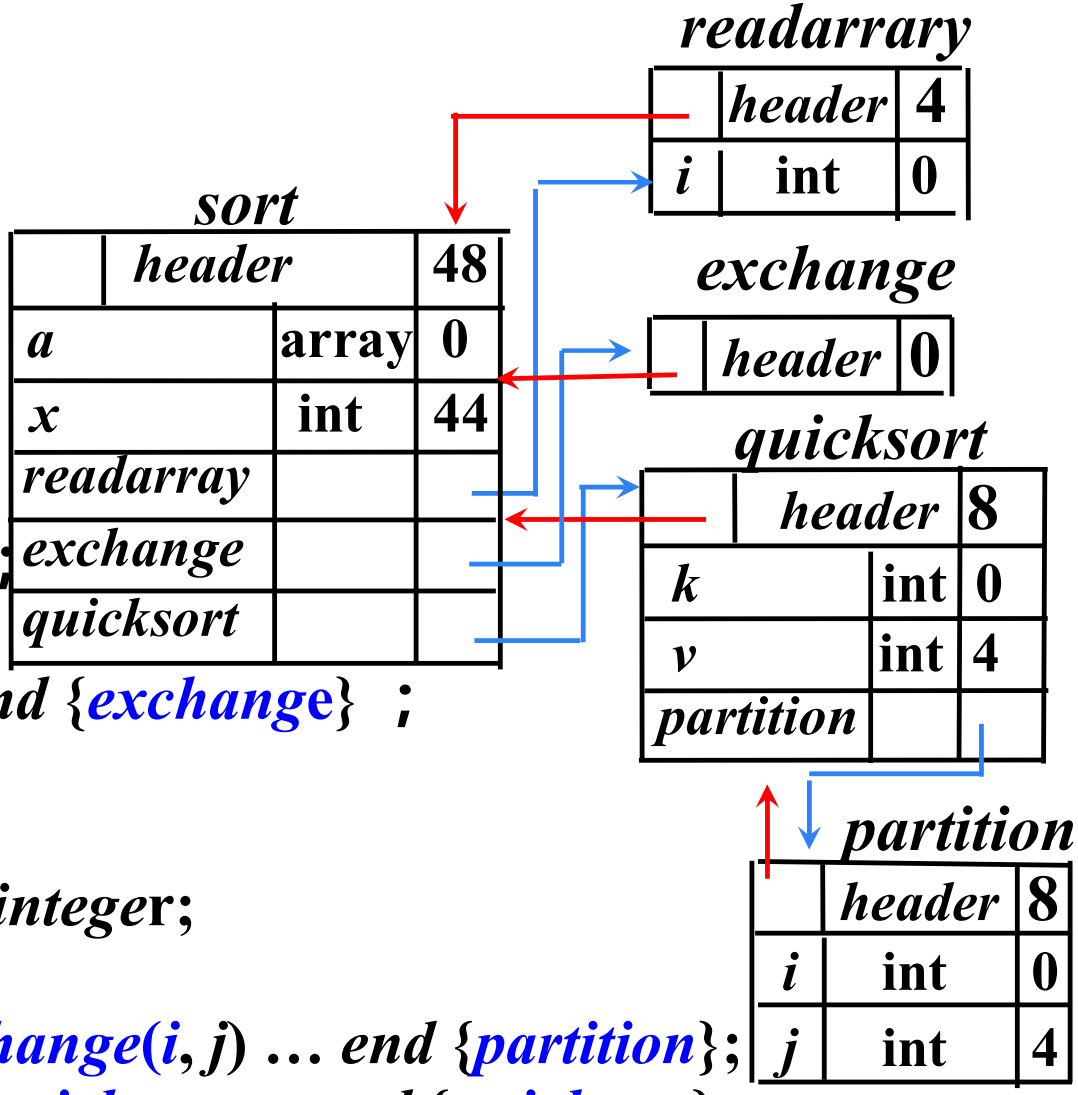
- 需要考虑的问题

假设过程*p*要访问过程*q*中的数据对象*x*，
*x*的地址＝*q*的活动记录基地址＋*x*在*q*的活动记录中的偏移地址

  - 刻画过程之间的嵌套关系（作用域信息）
  - 重名问题
- 常用的组织方式
  - 每个过程建立一个符号表，同时需要建立起这些符号表之间的联系，用来刻画过程之间的嵌套关系

例

```
program sort ( input, output );
    var a: array[0..10] of integer;
        x: integer;
    procedure readarray;
            var i: integer;
            begin … a … end {readarray} ;
    procedure exchange(i, j:integer);
            begin x=a[i];a[i]=a[j];a[j]=x; end {exchange} ;
    procedure quicksort(m, n:integer);
            var k, v : integer;
            function partition(y, z:integer):integer;
                    var i, j : integer;
                    begin … a … v … exchange(i, j) … end {partition};
            begin … a … v … partition … quicksort … end {quicksort} ;
begin … a … readarray … quicksort … end {sort} ;
```



嵌套过程声明语句的*SDT*

$P \rightarrow M\ D$ { *addwidth*(*top*(*tblptr*), *top*(*offset*));
pop(*tblptr*);
pop(*offset*); }

$M \rightarrow \varepsilon$ { *t* = *mktable*(*nil*);
push(*t*, *tblptr*);
push(0, *offset*); }

$D \rightarrow D_1\ D_2$

$D_p \rightarrow$ proc id ; $N\ D_1\ S$ { *t* = *top*(*tblptr*);
addwidth(*t*, *top*(*offset*));
pop(*tblptr*);
pop(*offset*);
enterproc(*top*(*tblptr*), id.*lexeme*, *t*); }

$D_v \rightarrow$ id: *T* ; { *enter*(*top*(*tblptr*), id.*lexeme*, *T.type*, *top*(*offset*));
top(*offset*) = *top*(*offset*) + *T.width*; }

$N \rightarrow \varepsilon$ { *t* = *mktable*(*top*(*tblptr*));
push(*t*, *tblptr*); *push*(0, *offset*); }

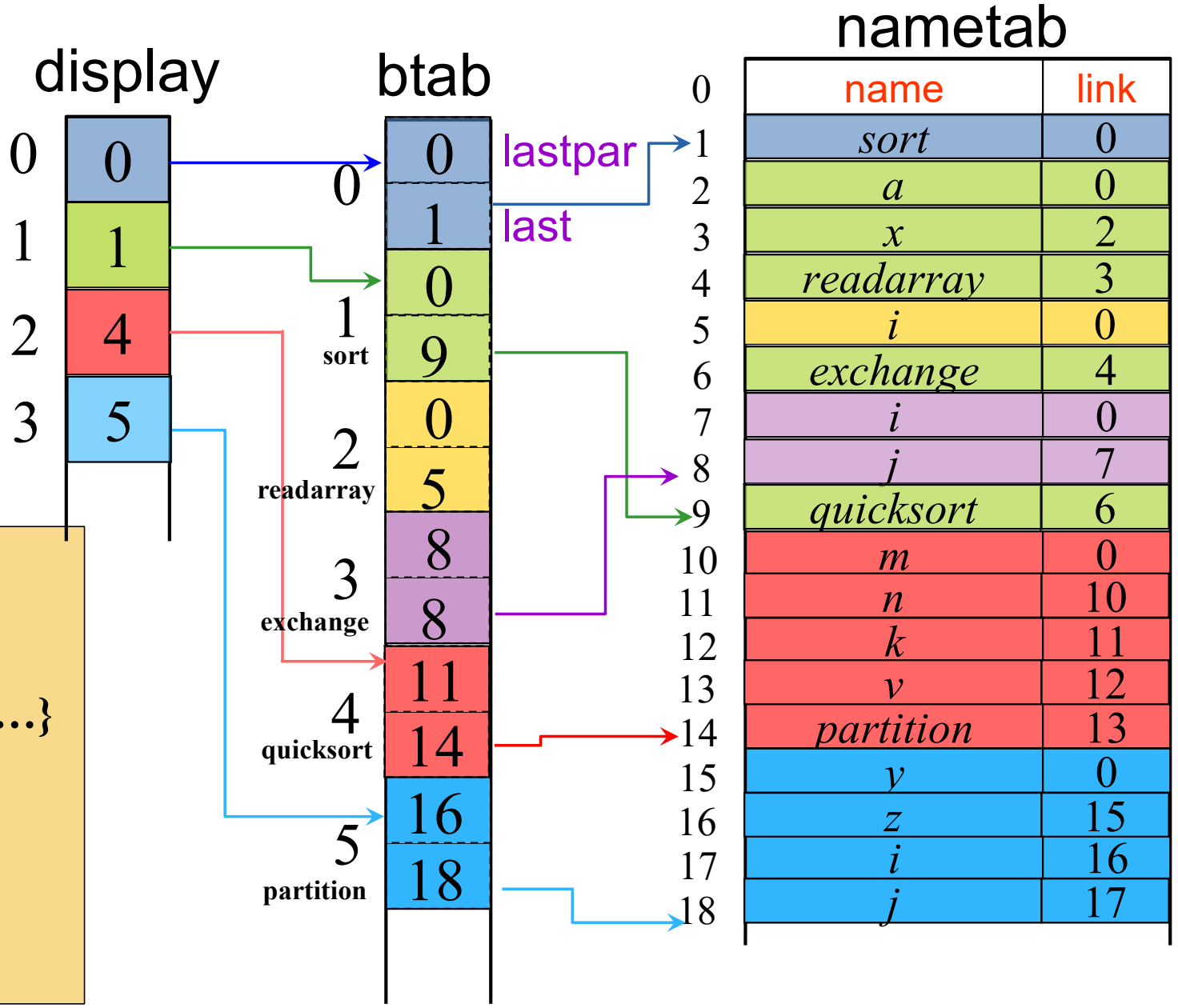
符号表的另一种组织方式

- 将所有块的符号表放在一个大数组中，然后再引入一个块表来描述各块的符号表在大数组中的位置及其相互关系
  - 一个过程可以看作是一个块

例

- lastpar指向本过程体中最后一个形参在nametab中的位置
- last指向本过程体中最后一个名字在nametab中的位置

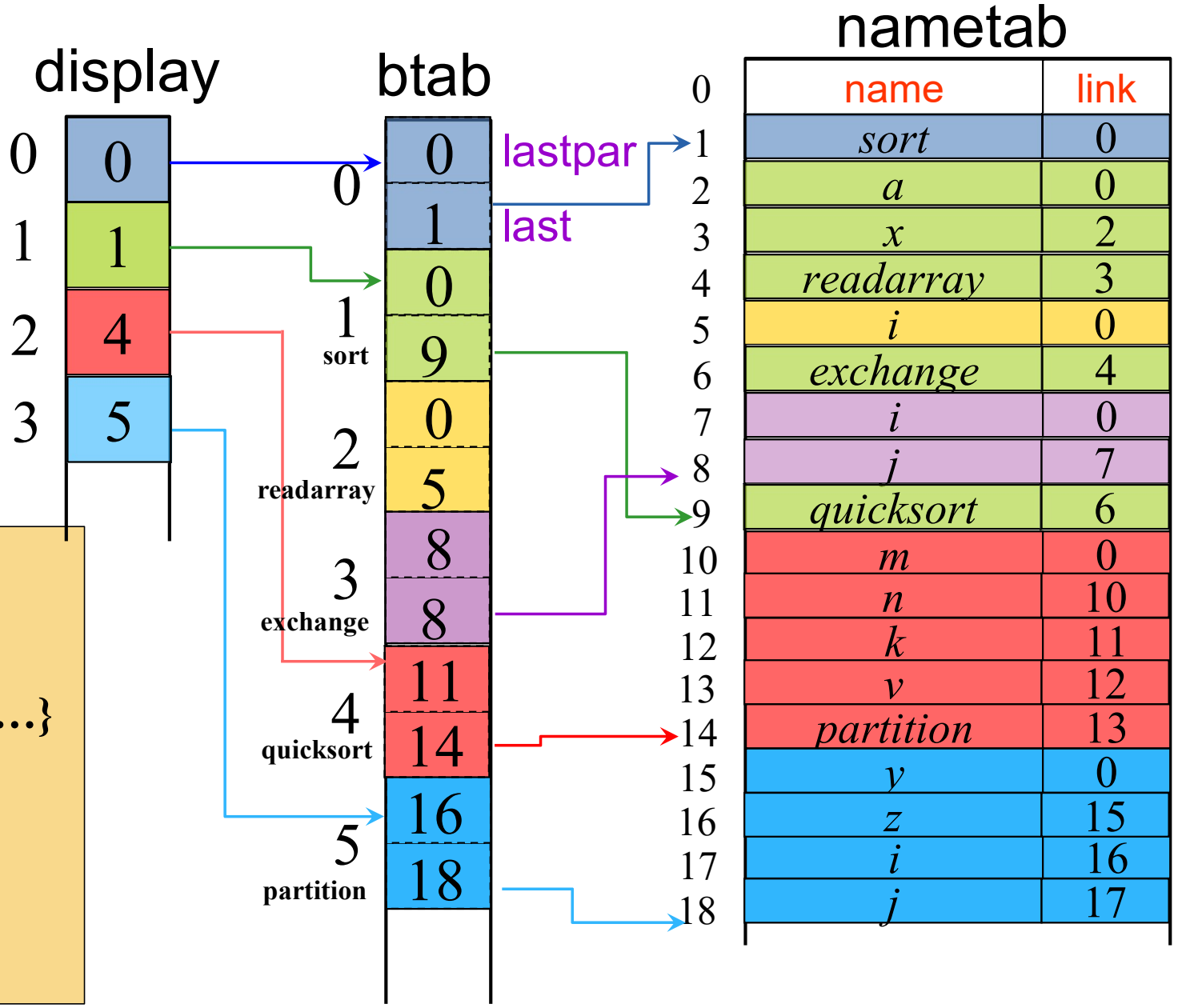
```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```



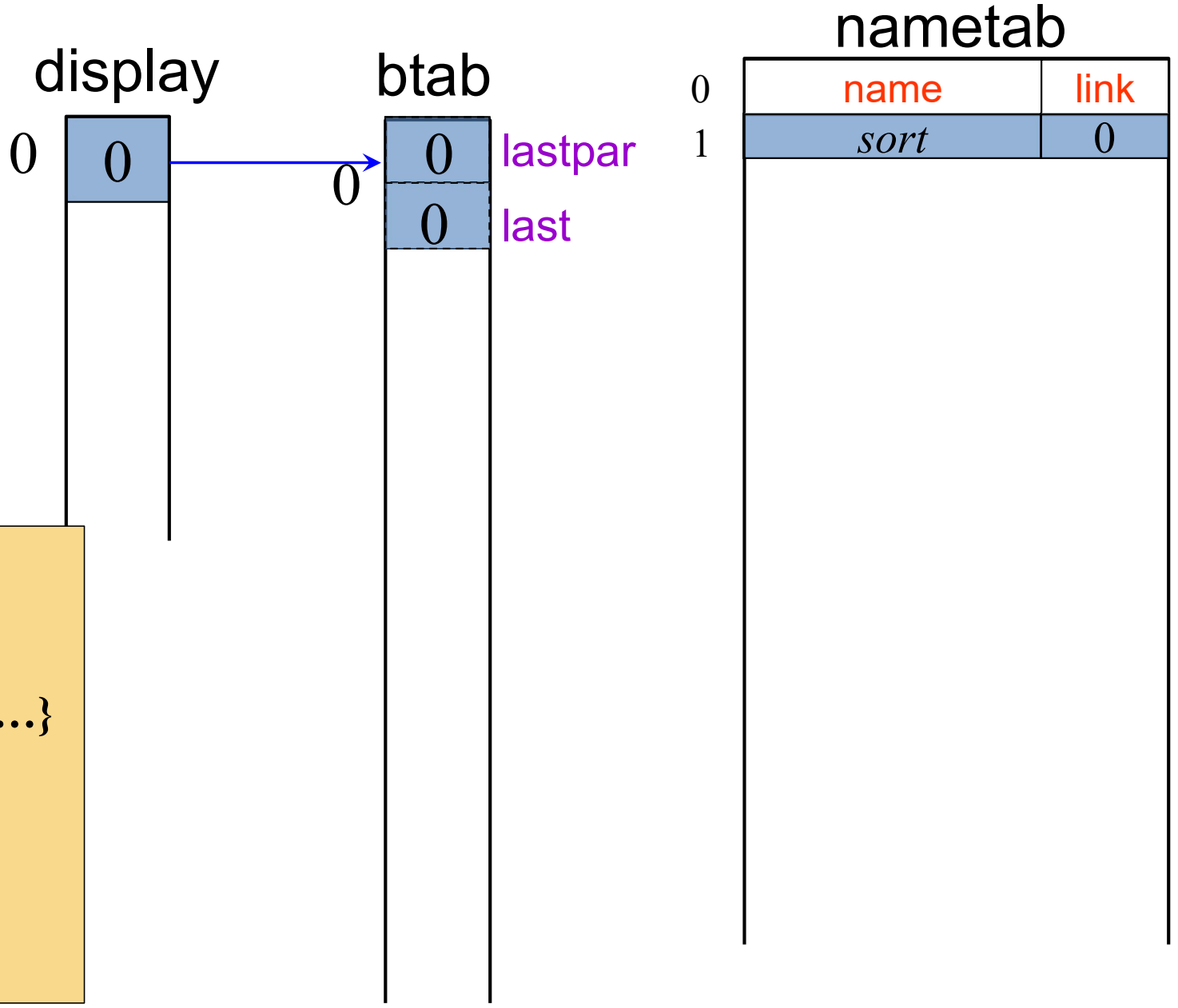
例

- link指向同一过程体中定义的上一个名字在nametab中的位置，每个过程体在nametab中登记的第一个名字的link为0

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

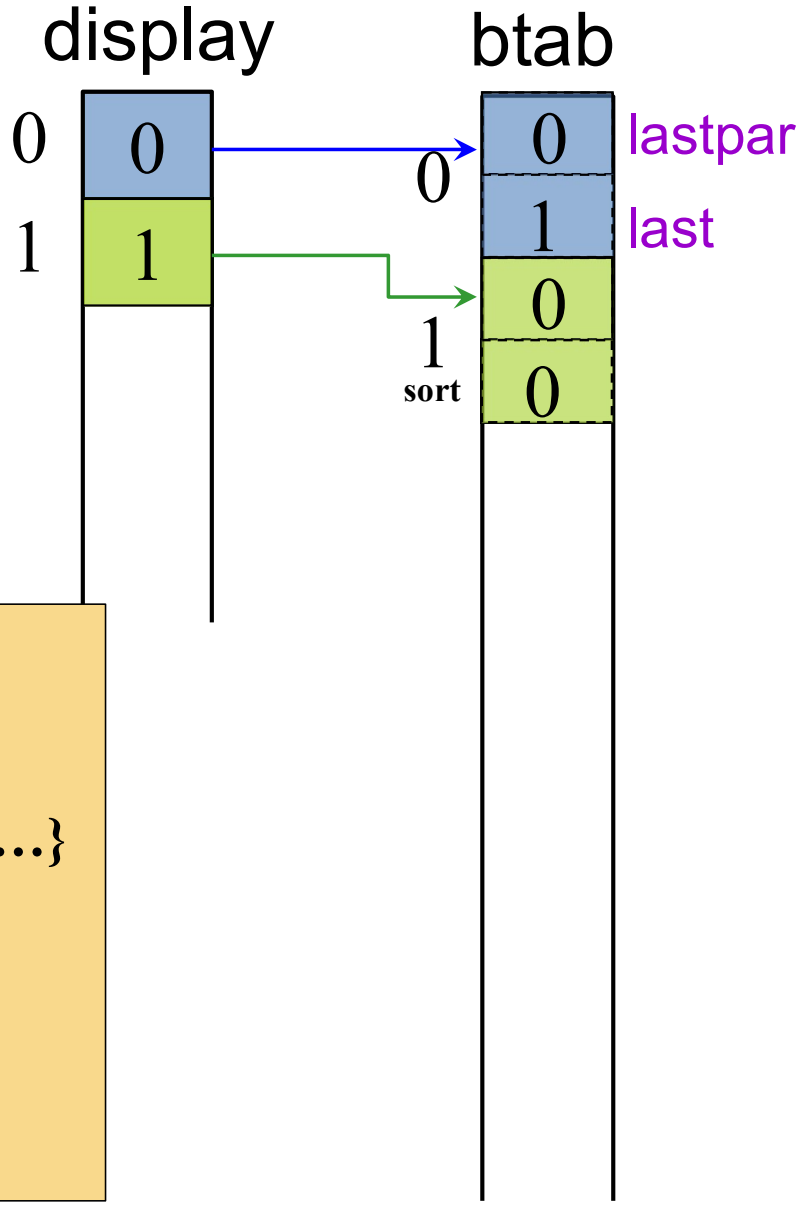


例




```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

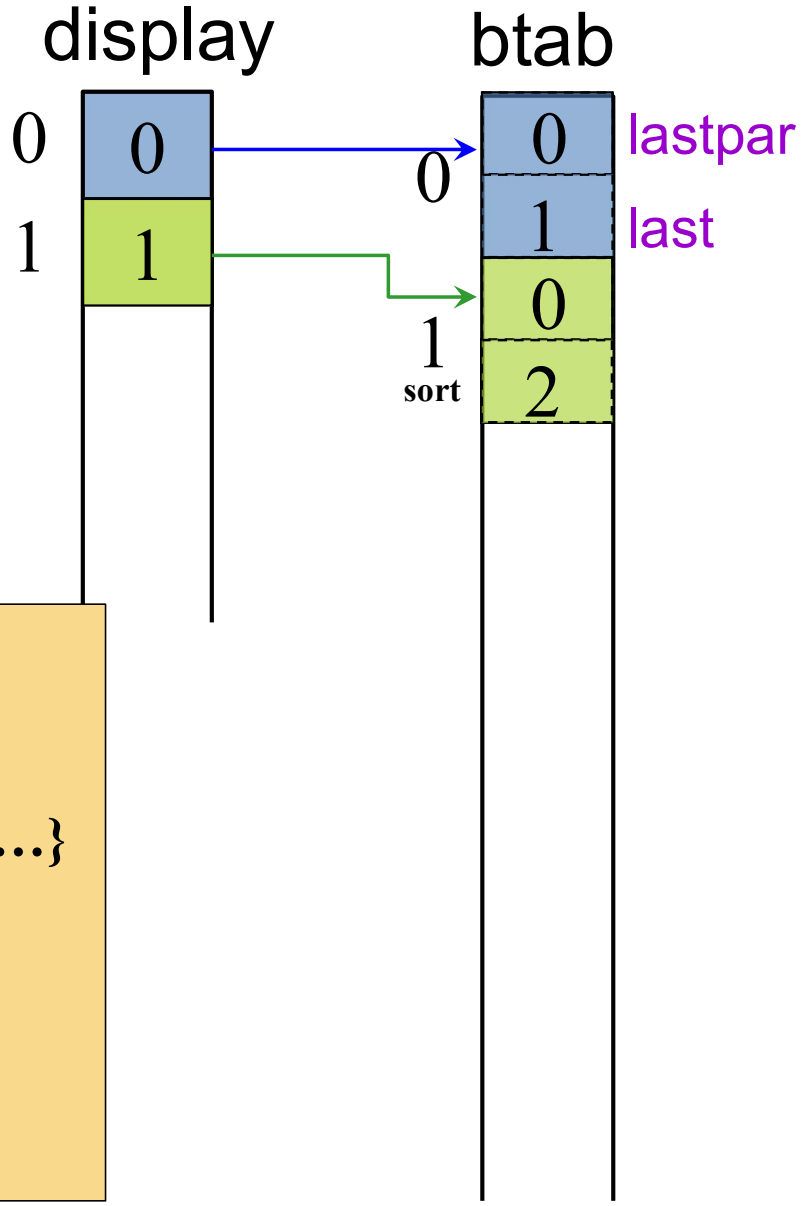


nametab

	name	link
0		
1	*sort*	0
2	*a*	0

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

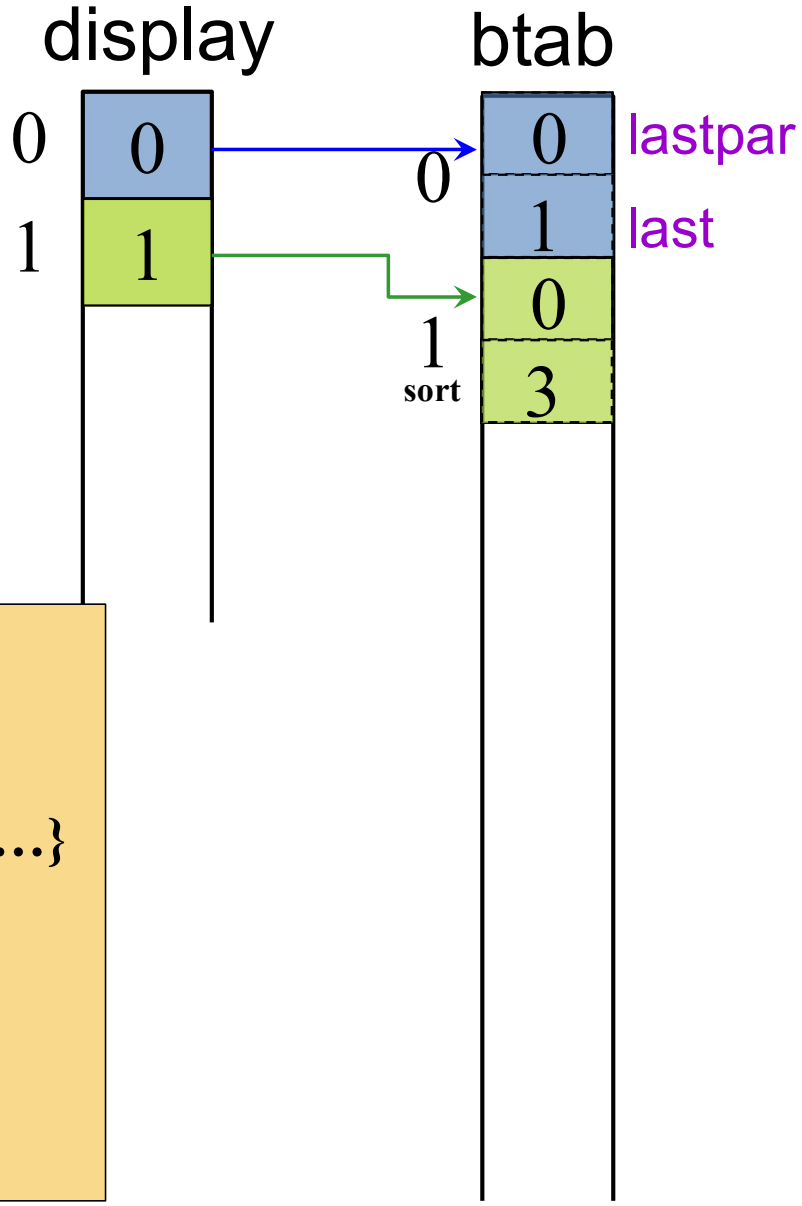


nametab

	name	link
0		
1	*sort*	0
2	*a*	0
3	*x*	2

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

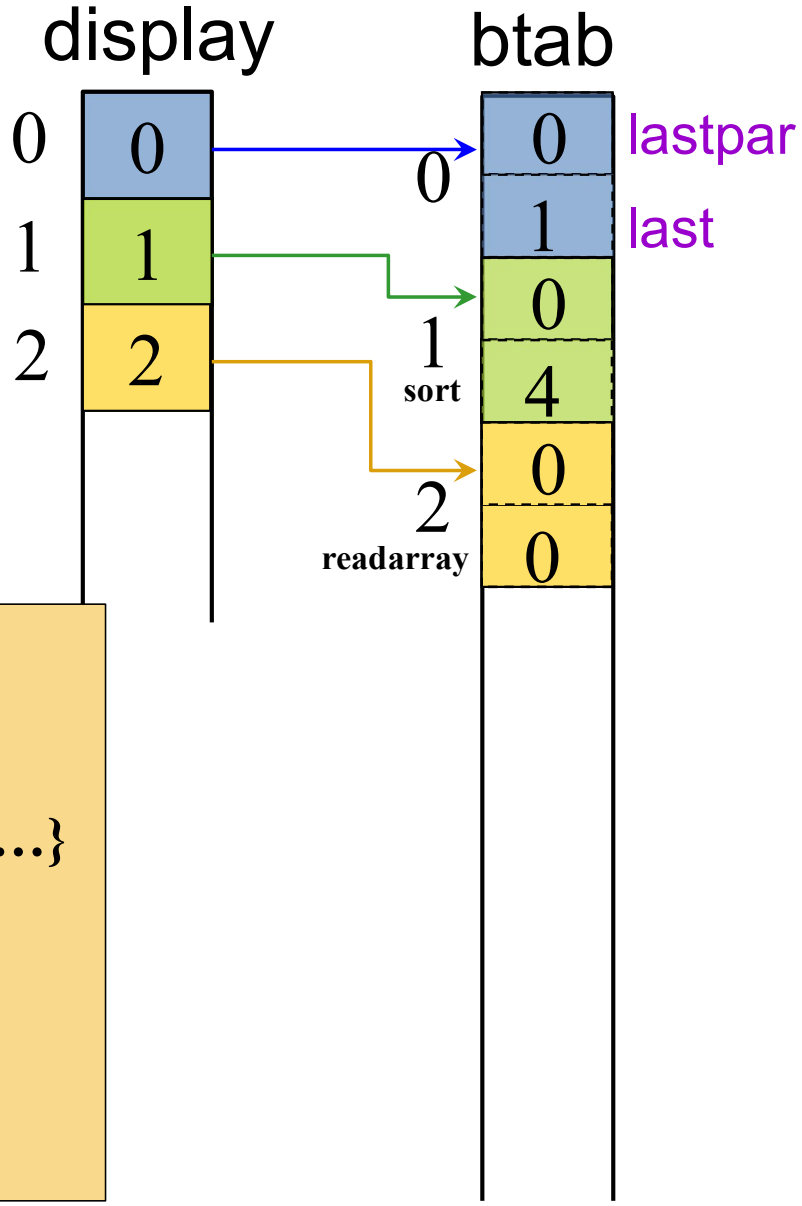


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

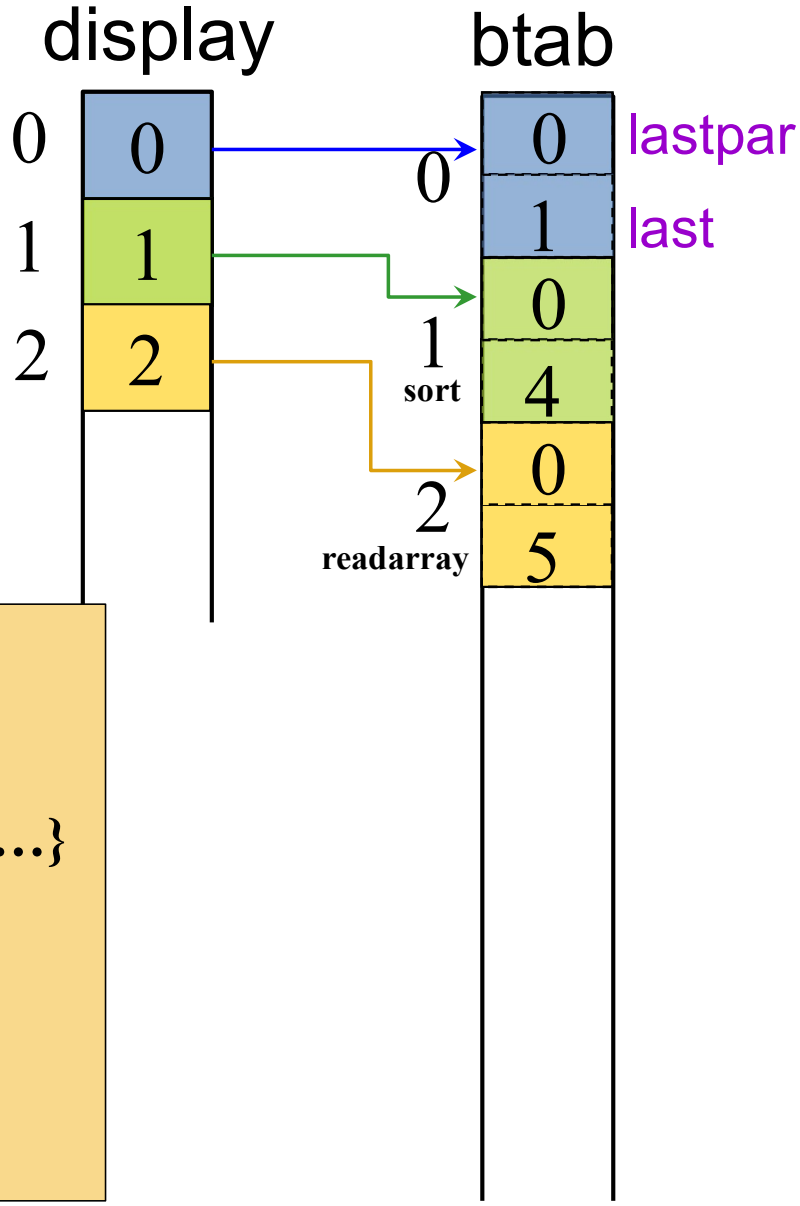


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

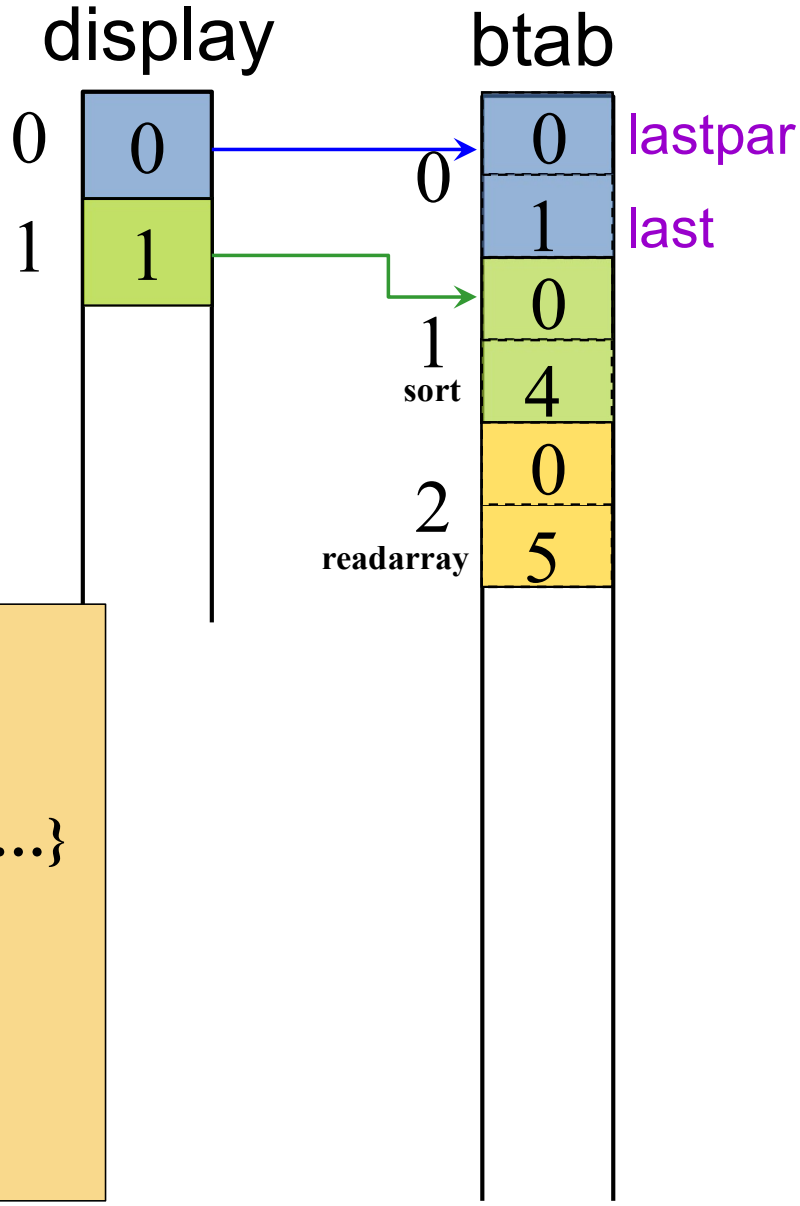


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

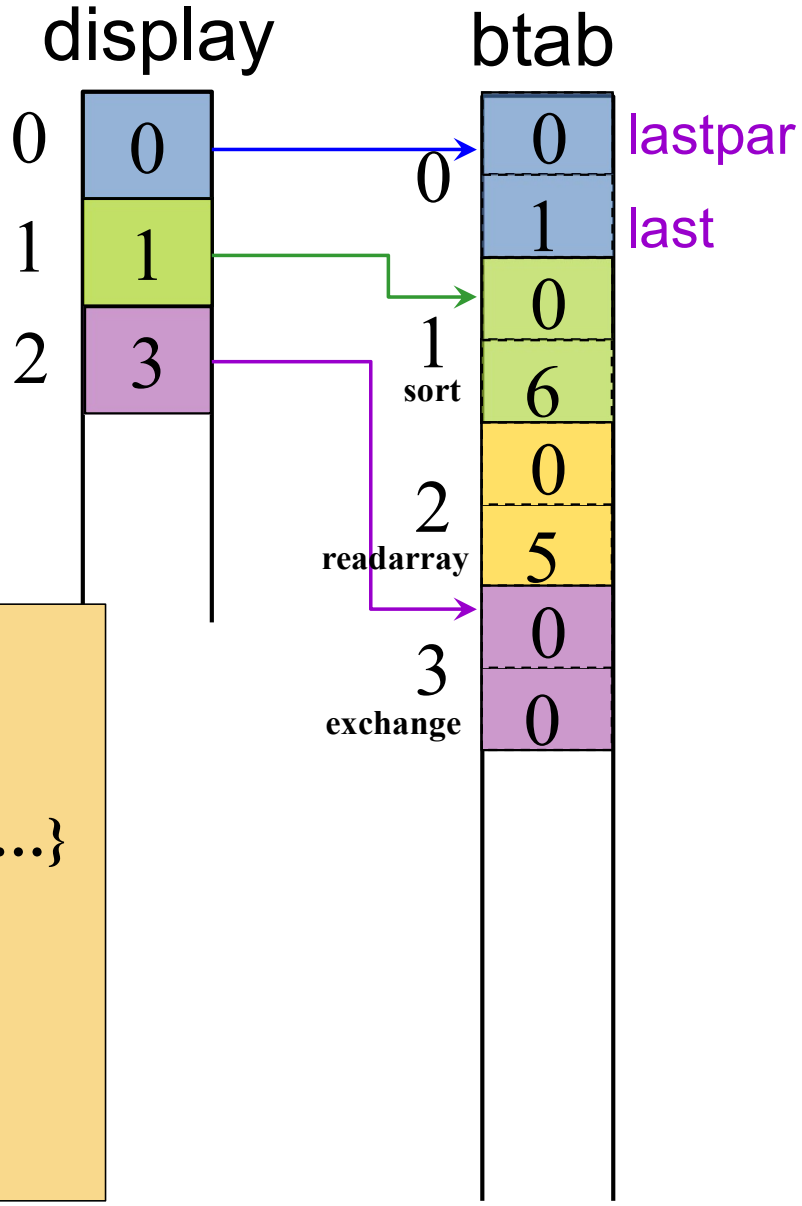


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

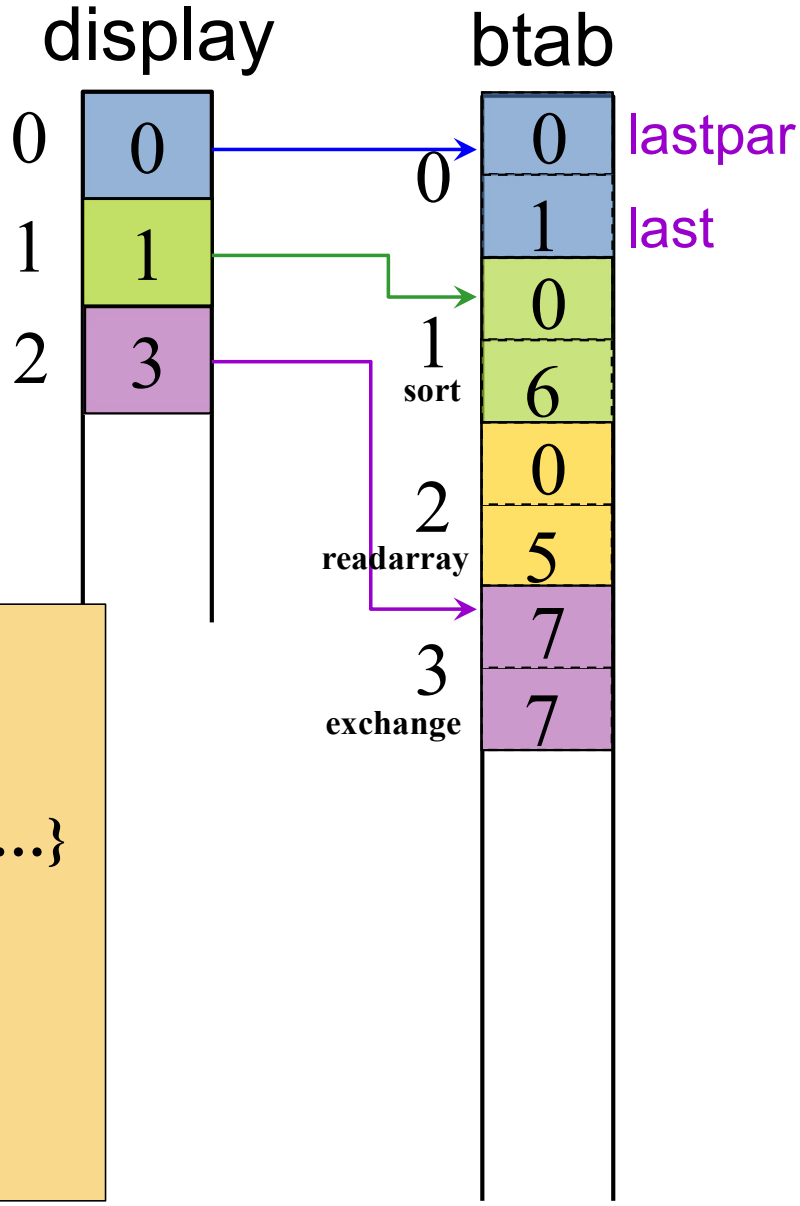


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

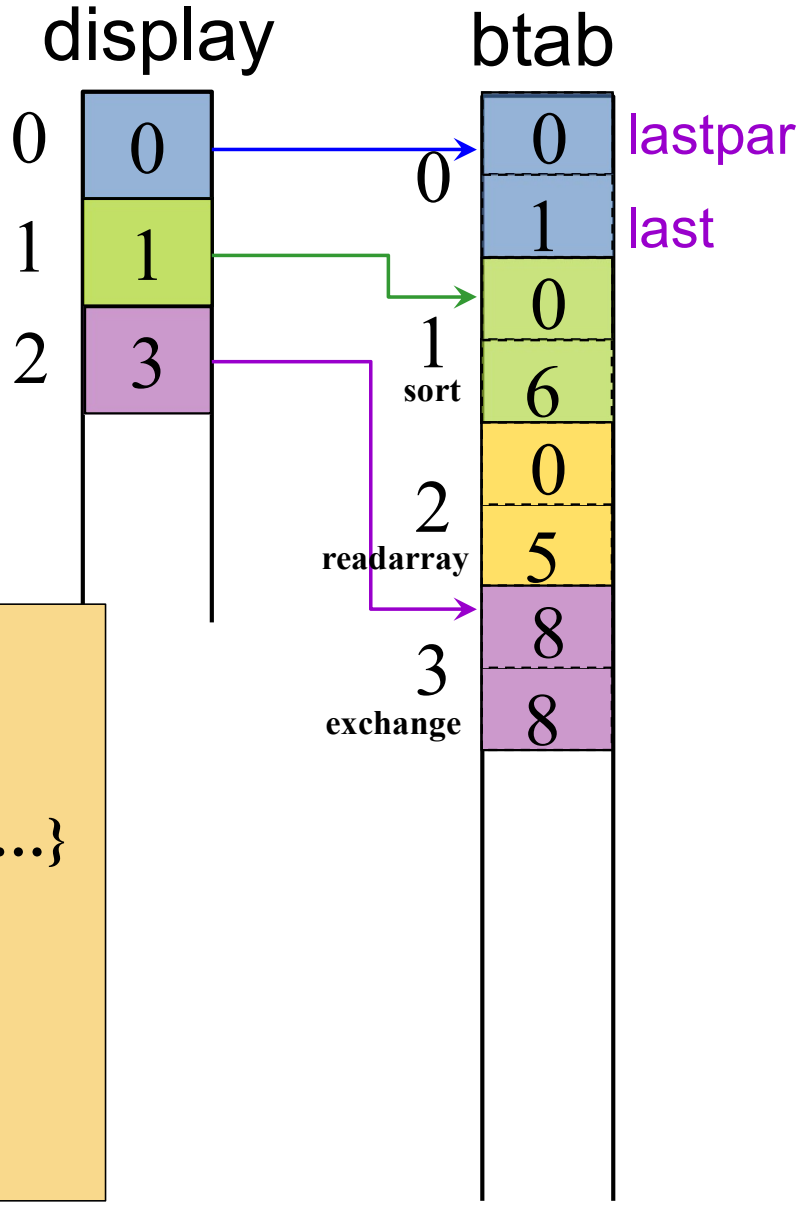


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

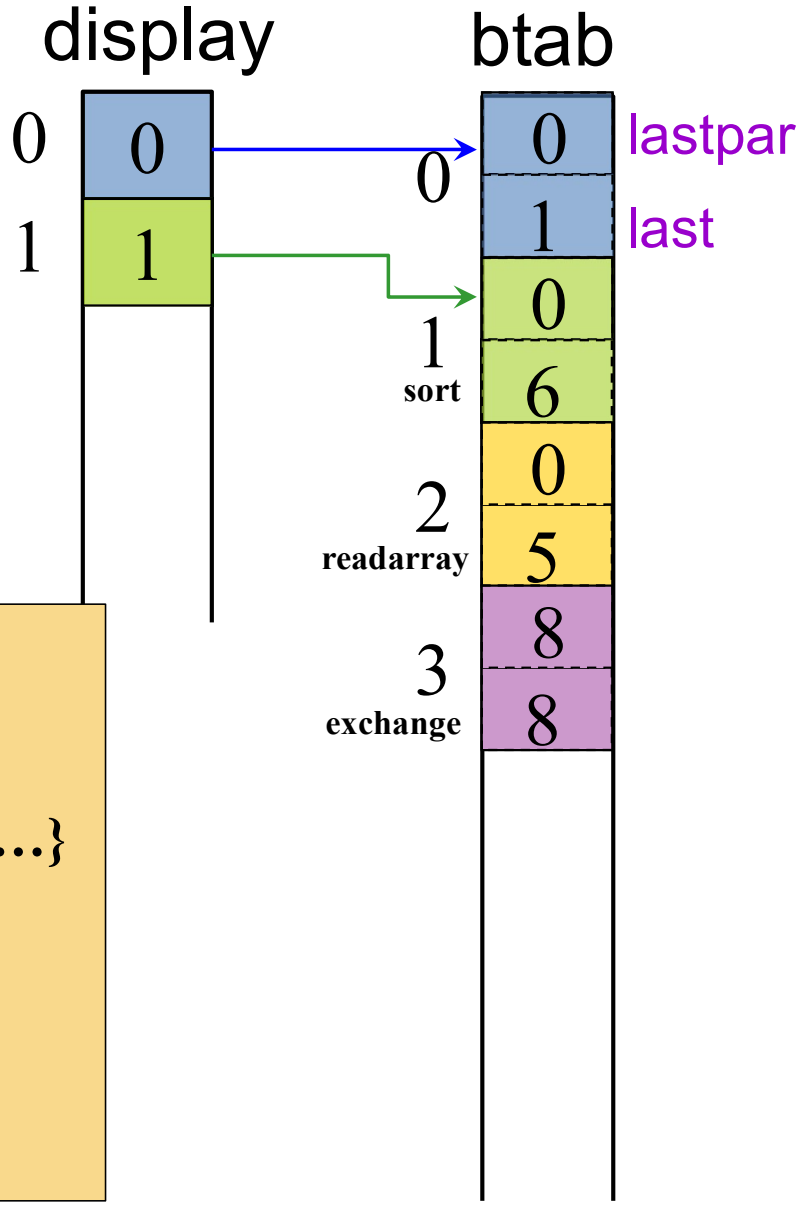


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

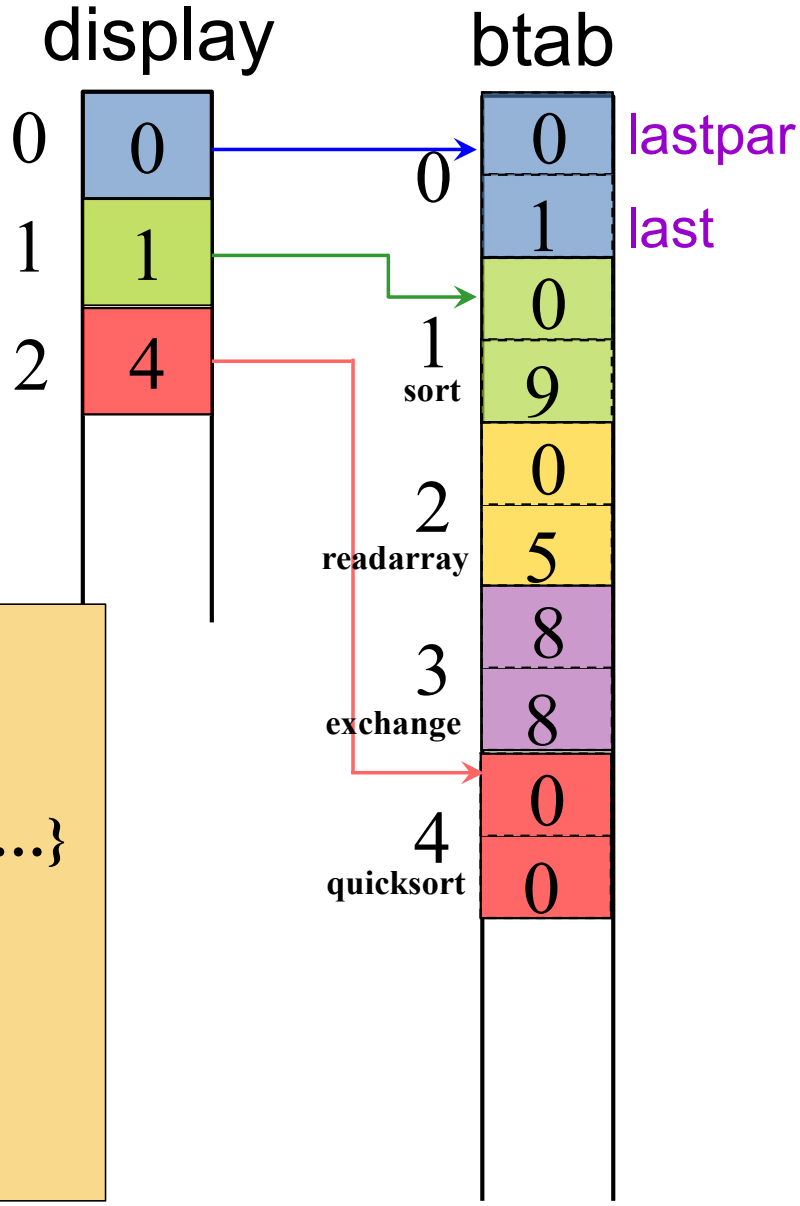


nametab

	name	link
0		
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

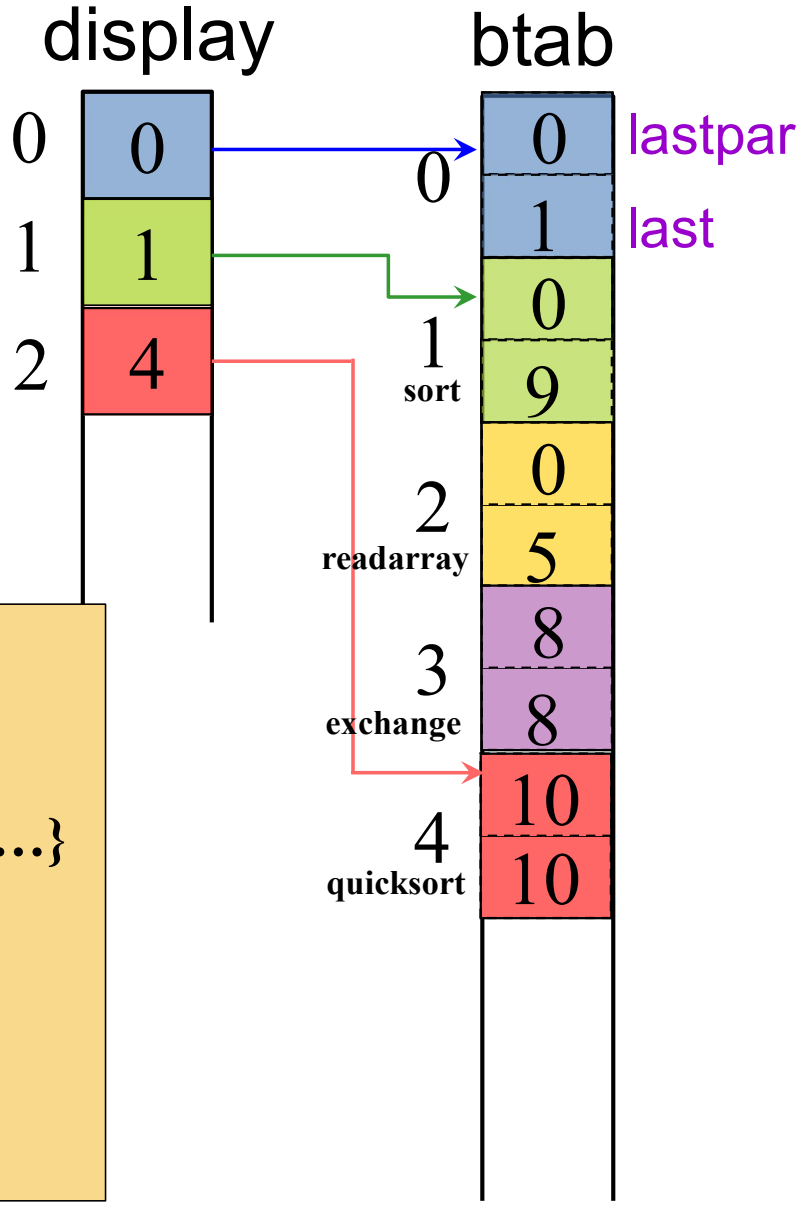



```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0

例

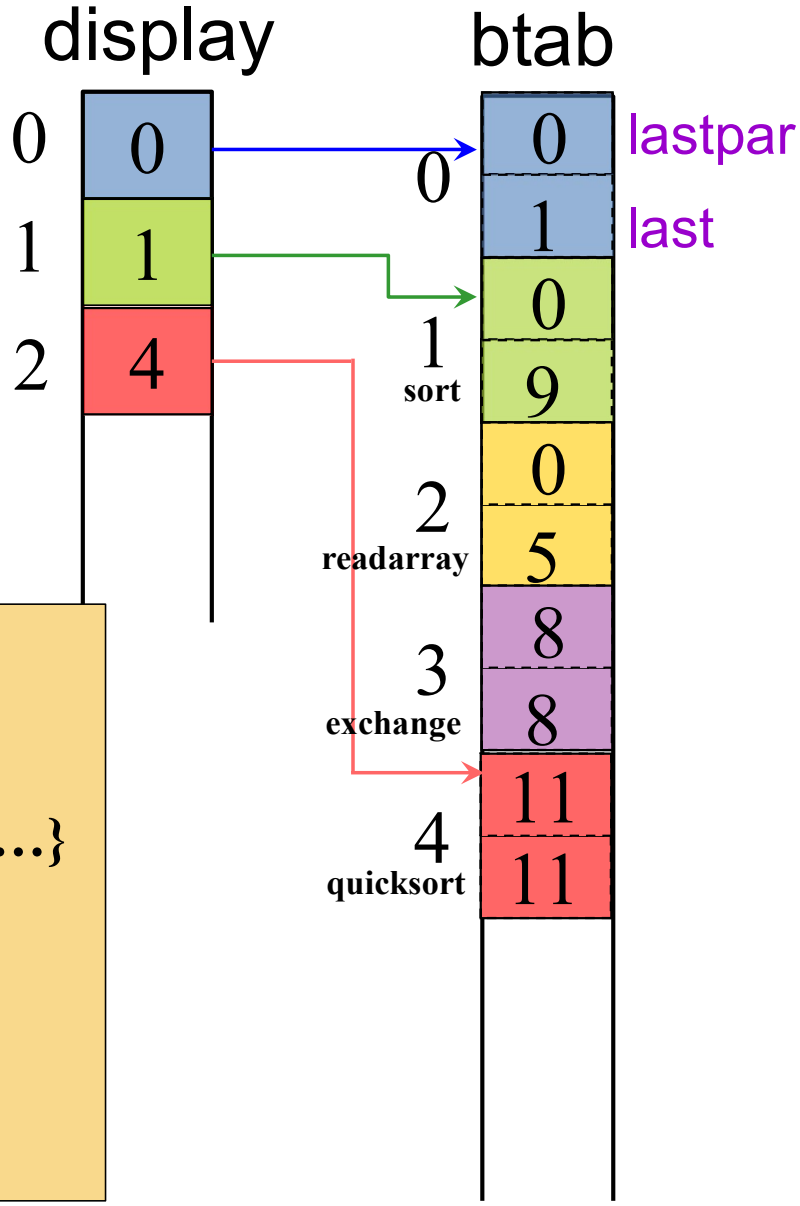



```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0
11	*n*	10

例

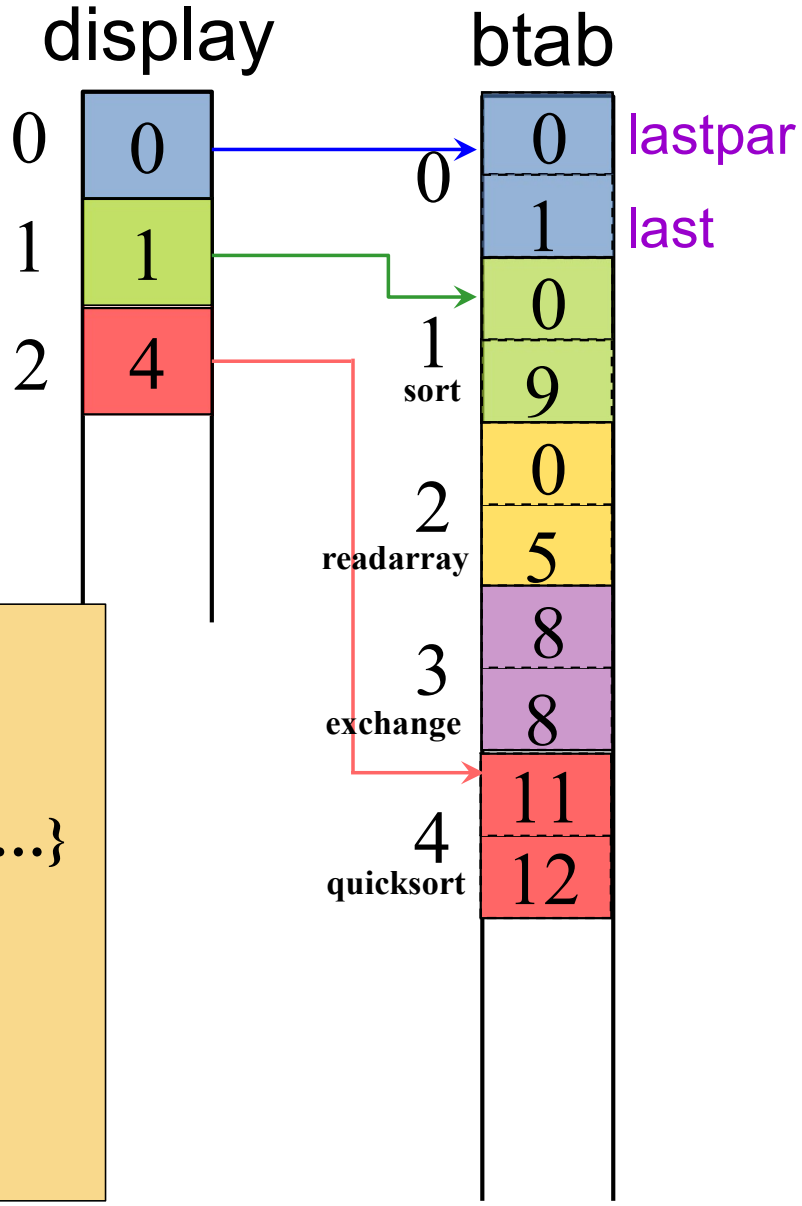


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0
11	*n*	10
12	*k*	11

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

例

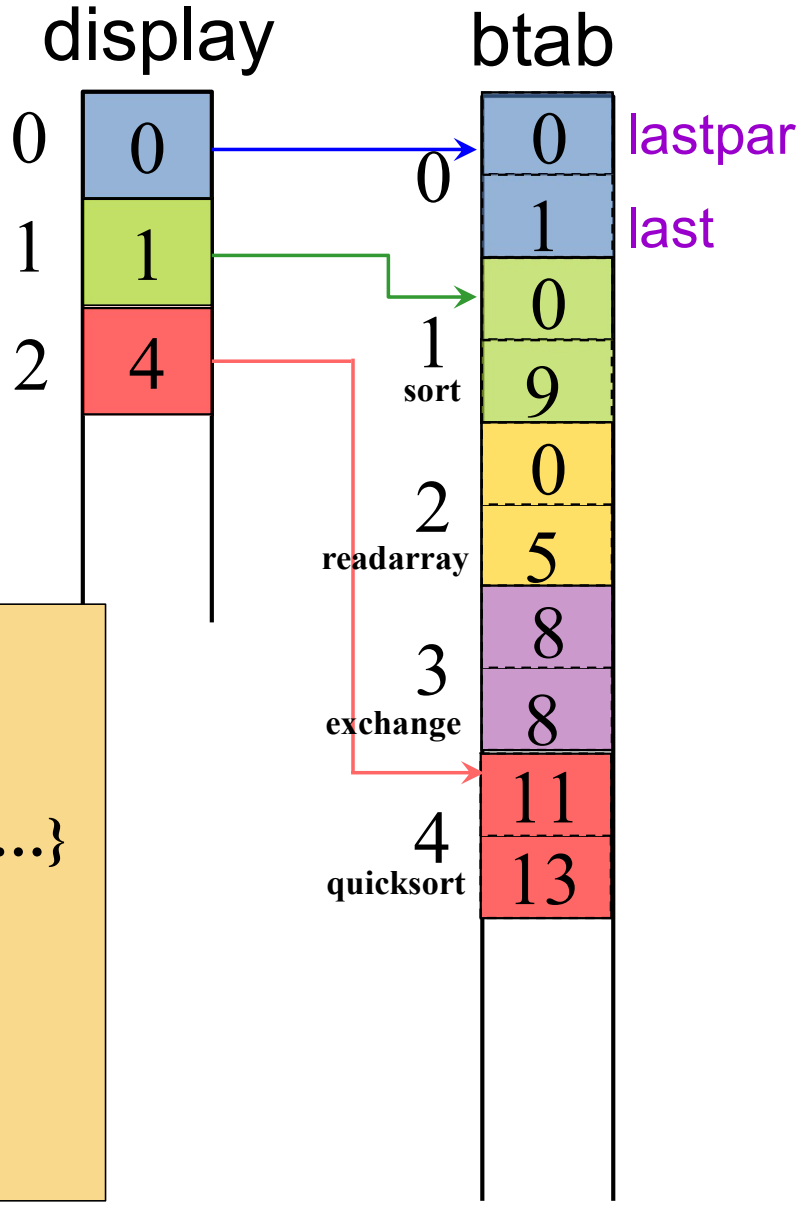



```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0
11	*n*	10
12	*k*	11
13	*v*	12

例

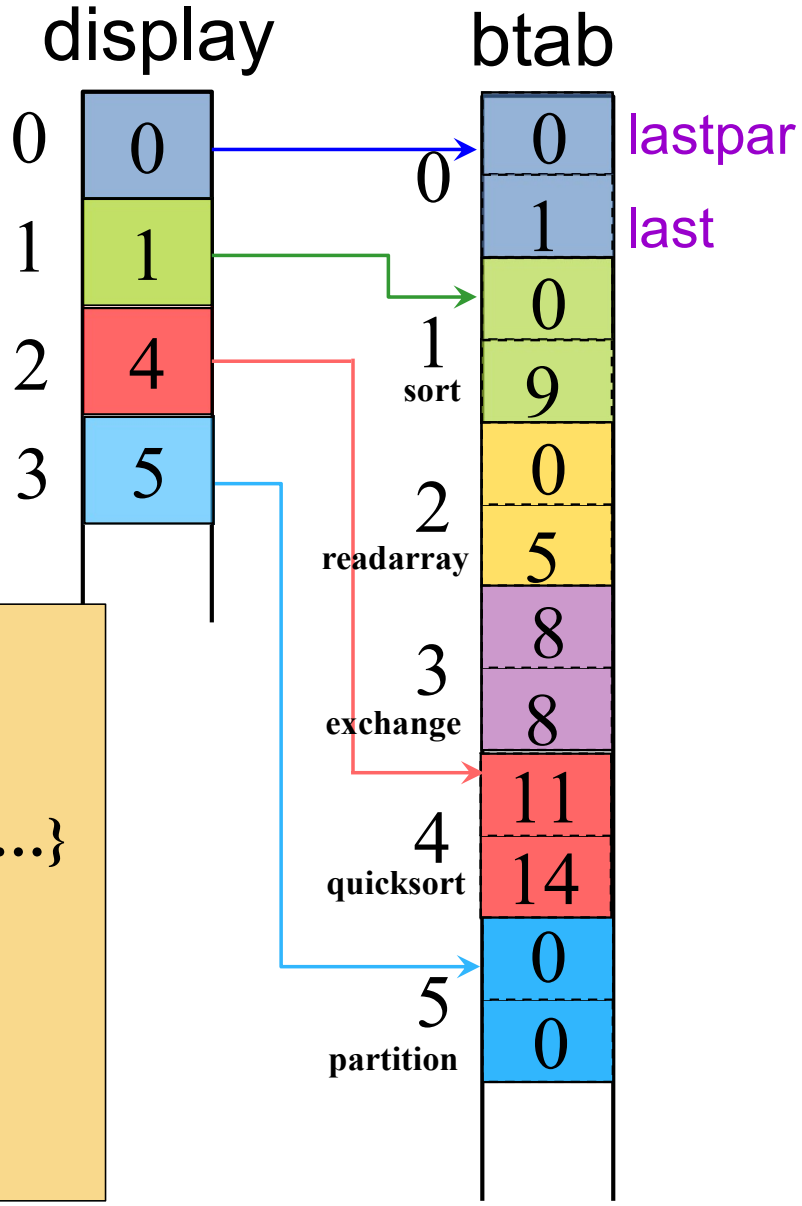



```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0
11	*n*	10
12	*k*	11
13	*v*	12
14	*partition*	13

例

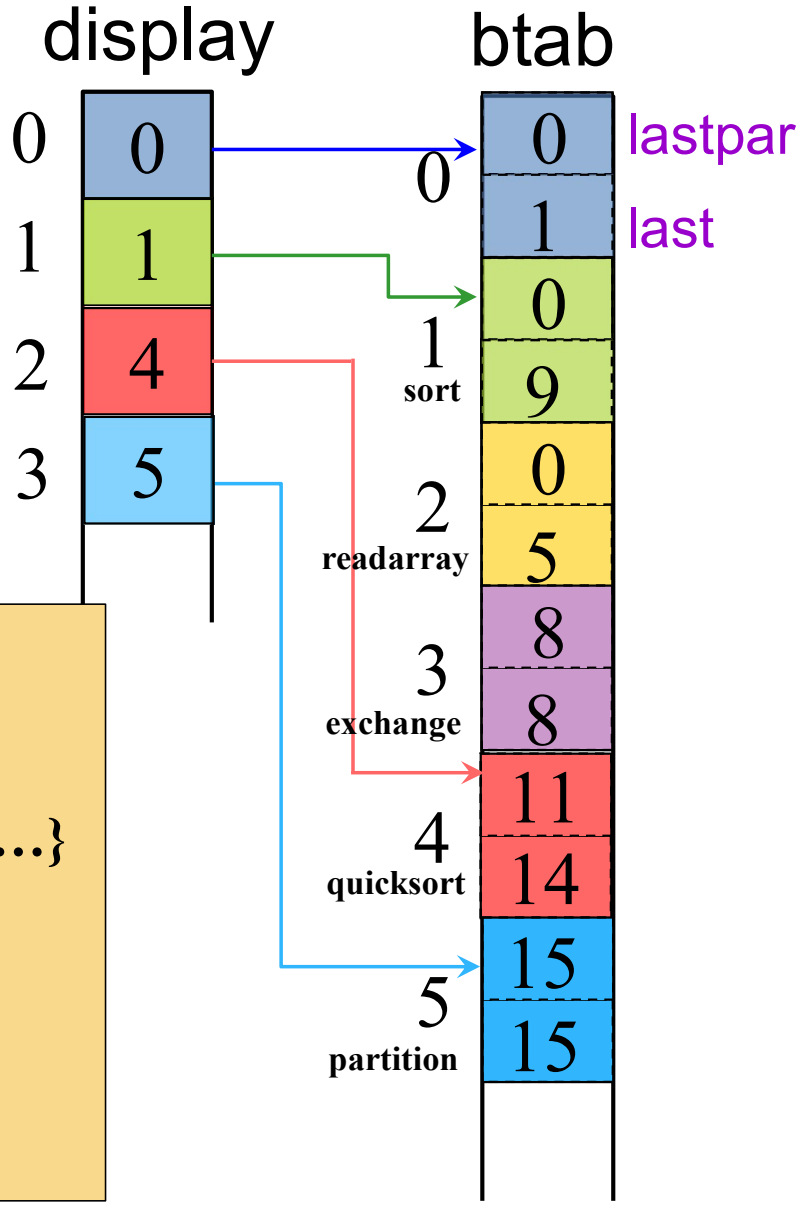



```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0
11	*n*	10
12	*k*	11
13	*v*	12
14	*partition*	13
15	*y*	0

例



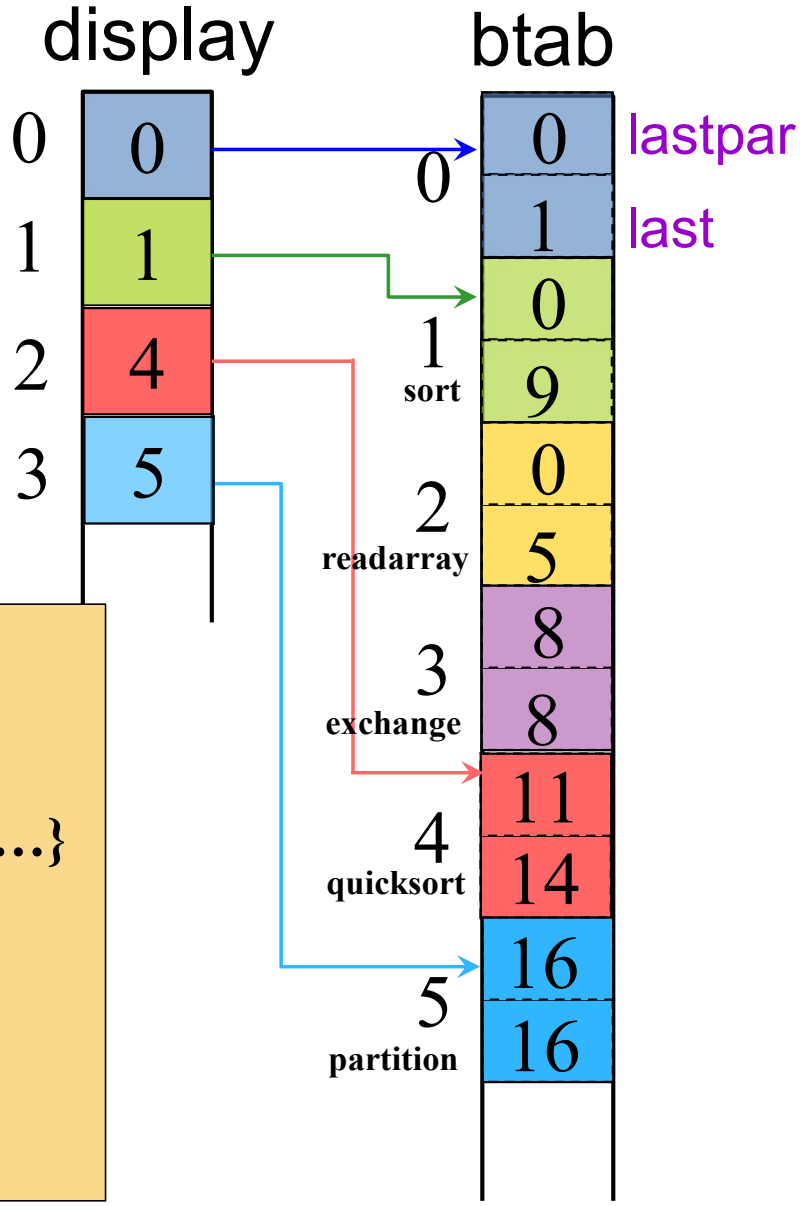

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0
11	*n*	10
12	*k*	11
13	*v*	12
14	*partition*	13
15	*y*	0
16	*z*	15

例

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

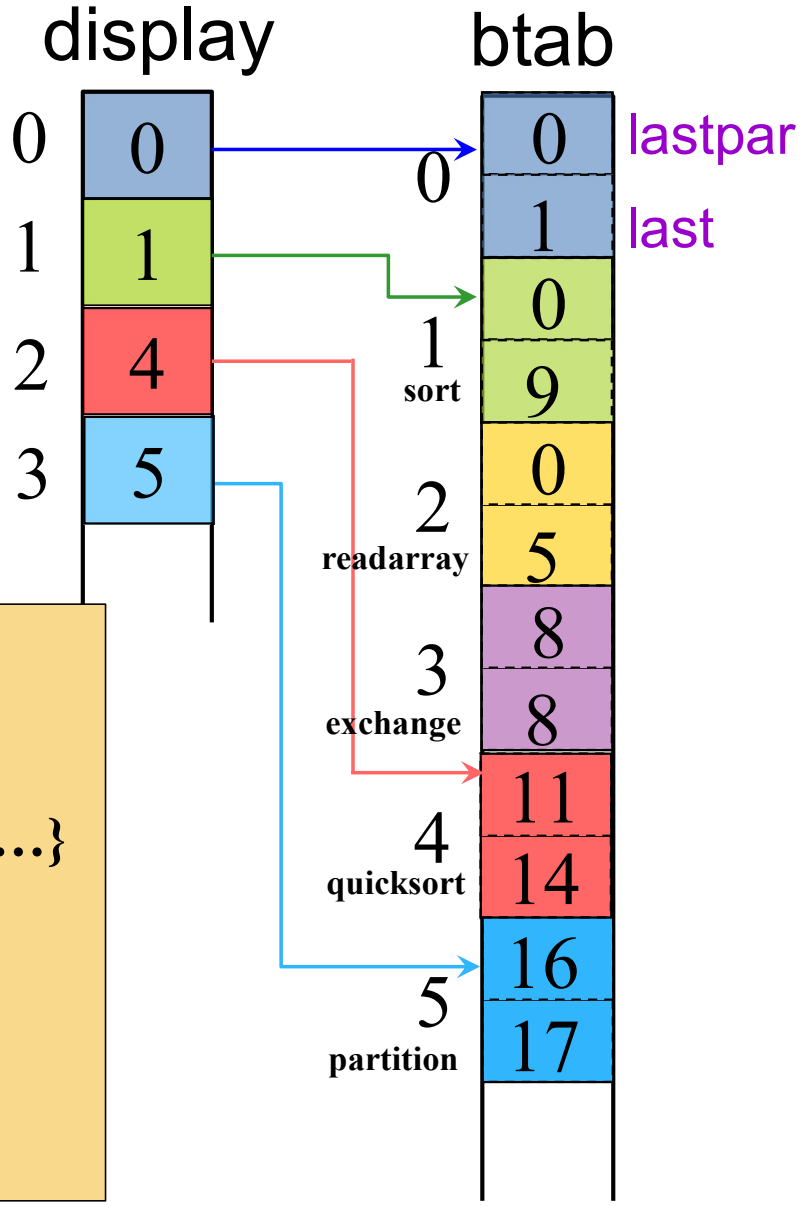


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0
11	*n*	10
12	*k*	11
13	*v*	12
14	*partition*	13
15	*y*	0
16	*z*	15
17	*i*	16

例

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

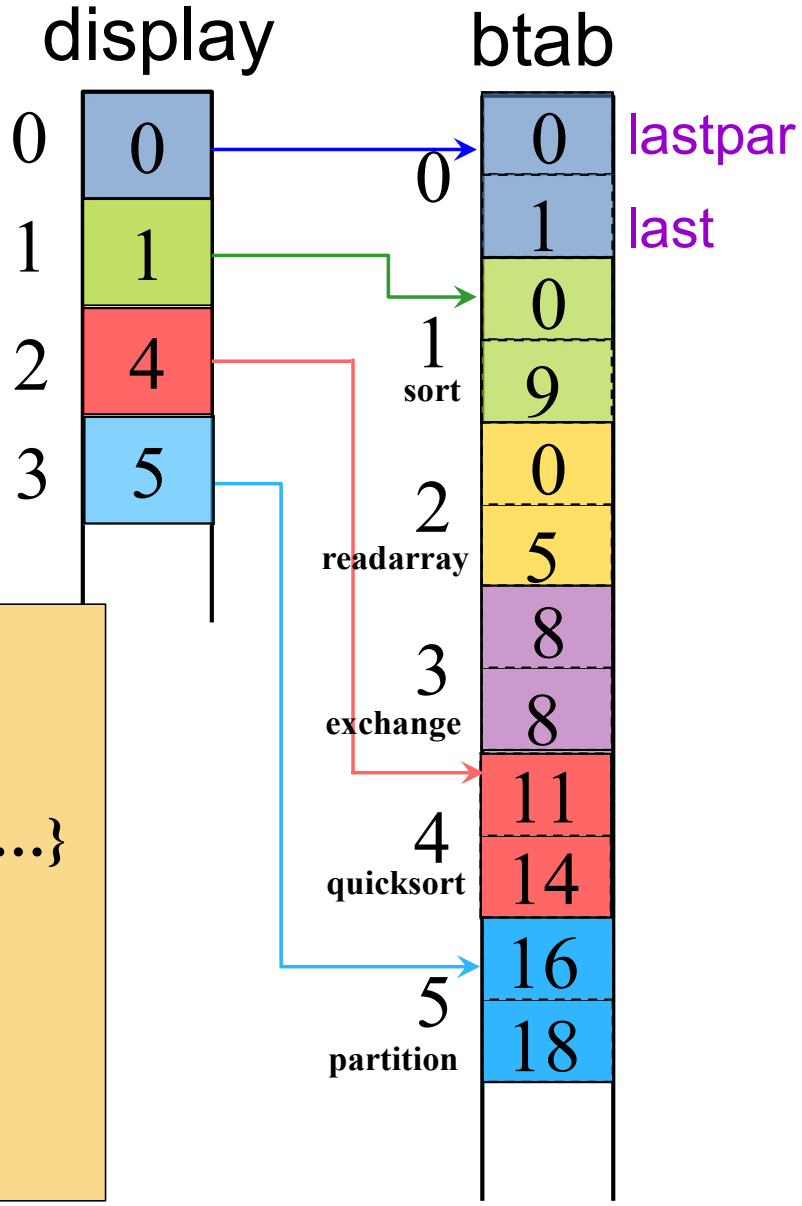


nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0
11	*n*	10
12	*k*	11
13	*v*	12
14	*partition*	13
15	*y*	0
16	*z*	15
17	*i*	16
18	*j*	17

例

```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```



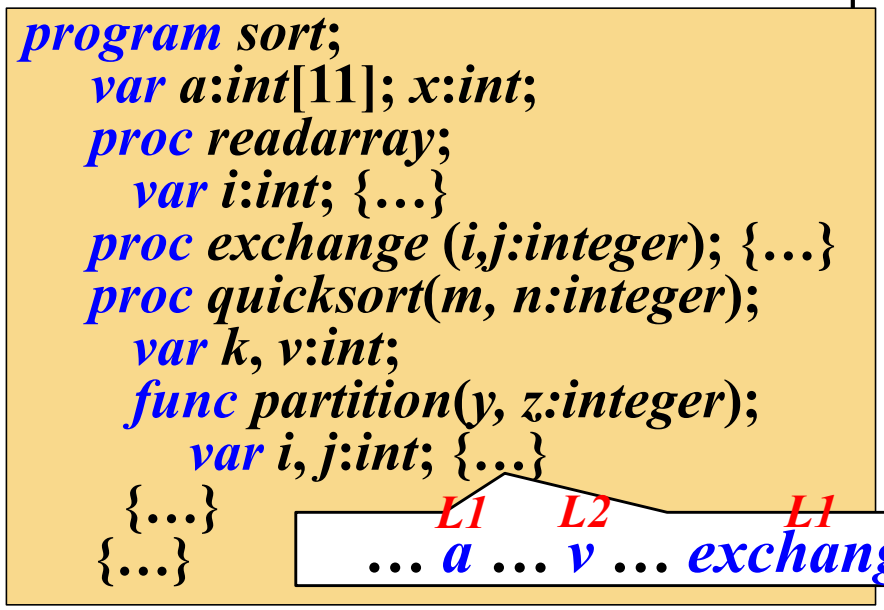
nametab

	name	link
1	*sort*	0
2	*a*	0
3	*x*	2
4	*readarray*	3
5	*i*	0
6	*exchange*	4
7	*i*	0
8	*j*	7
9	*quicksort*	6
10	*m*	0
11	*n*	10
12	*k*	11
13	*v*	12
14	*partition*	13
15	*y*	0
16	*z*	15
17	*i*	16
18	*j*	17

例

如何根据符号表生成中间代码中的地址信息？

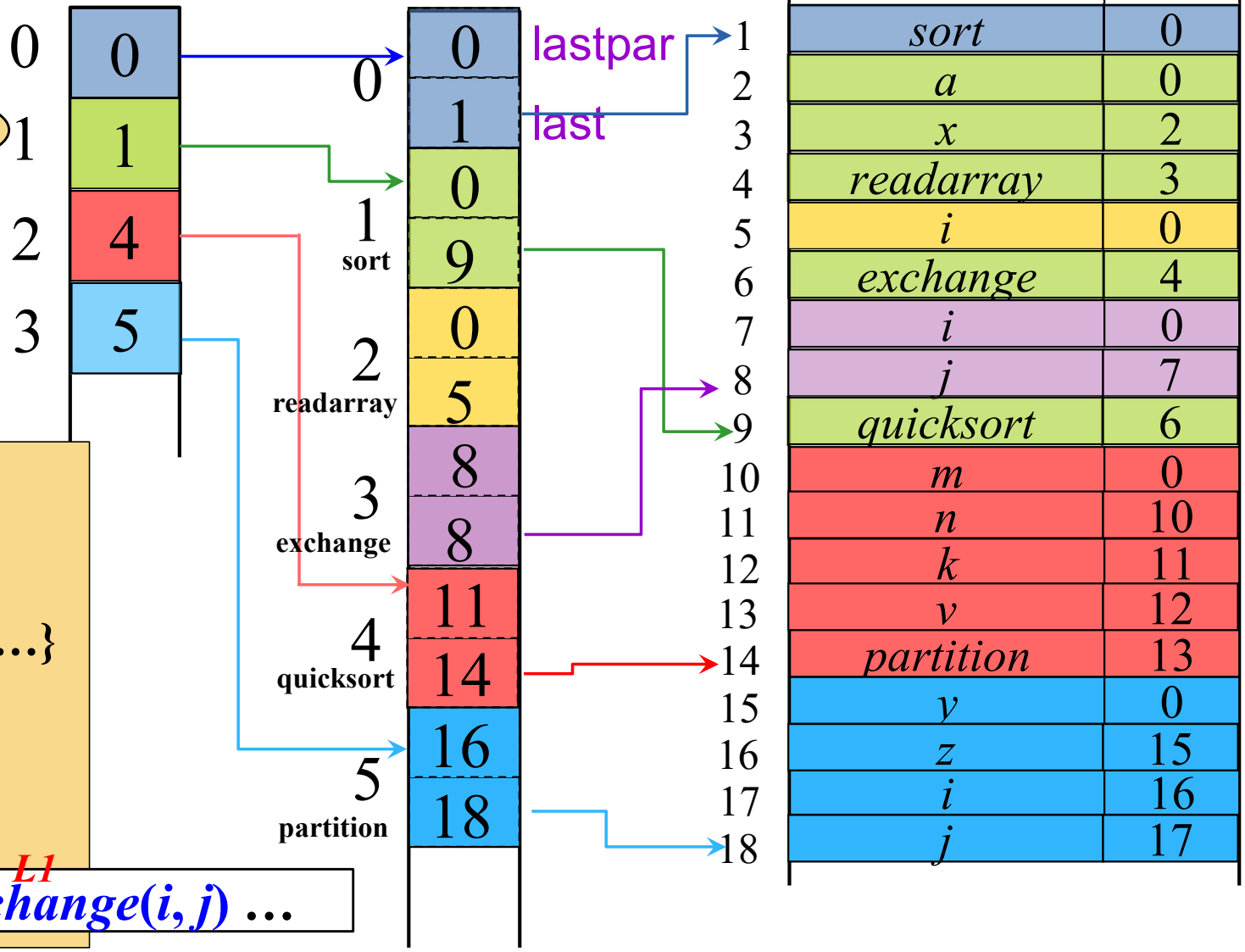
地址 =层号 + 偏移地址



```
program sort;
    var a:int[11]; x:int;
    proc readarray;
       var i:int; {…}
    proc exchange (i,j:integer); {…}
    proc quicksort(m, n:integer);
       var k, v:int;
       func partition(y, z:integer);
          var i, j:int; {…}
       {…}
    {…}
```

L1 L2 L1

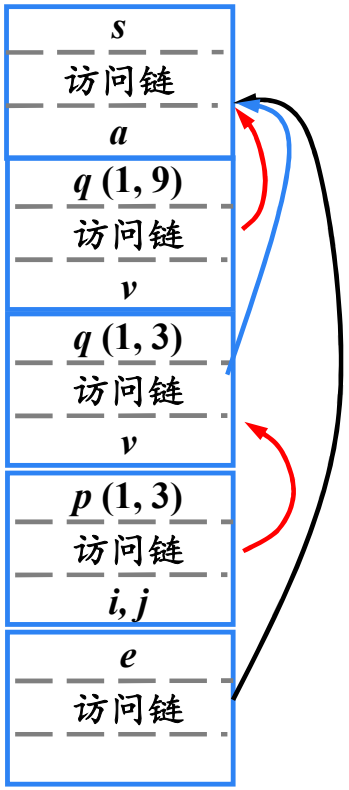
… a … v … exchange(i, j) …



访问链的建立

- 建立访问链的代码属于调用序列的一部分
- 假设嵌套深度为n_x的过程x调用嵌套深度为n_y的过程y $(x \rightarrow y)$
  - $n_x < n_y$的情况(外层调用内层)
  - $n_x = n_y$的情况(本层调用本层)
  - $n_x > n_y$的情况(内层调用外层，如： $p \rightarrow e$)
    - 调用者x必定嵌套在某个过程z中，而z中直接定义了被调用者y
    - 从x的活动记录开始，沿着访问链经过$n_x - n_y + 1$步就可以找到离栈顶最近的z的活动记录。y的访问链必须指向z的这个活动记录

嵌套深度是在编译阶段通过静态分析就能确定的



过程	嵌套深度
sort	1
readarray	2
exchange	2
quicksort	2
partition	3

结束