编译原理

参数传递

参数传递

- 过程或函数是模块程序设计的主要手段，也是节省程序代码和扩充语言的主要途径
- 过程定义

```
procedure  add(x, y : integer; var z : integer)
begin
      z := x + y;
end;
```

形式参数(形参)

- 过程调用

```
add(a, b, c);
```

实在参数(实参)

参数传递方式

- 传地址
- 得结果
- 传值
- 传名

To understand a program you must become both the machine and the program.

Alan J. Perlis

```
add(a, b, c);
```

```
procedure  add(x, y : integer; var z : integer)
 begin
       z := x + y;
 end;
```

编译原理

参数传递方式——传地址

参数传递方式——传地址

- 把实在参数的地址传递给相应的形式参数
- 方法
  - 调用程序把实在参数(实参)的地址传递到被调用过程相应的形式单元中
  - 被调用过程中，对形式参数(形参)的引用或赋值被处理成对形式单元的间接访问
- PASCAL的变量参数，C/C++传引用

参数传递方式——传地址

```
procedure swap (var m:integer; var n: integer);
var i:integer;
begin
    i:=m;
    m:=n;
    n:=i;
end
```

swap(a, b)

- 把a,b的地址送到形式单元m和n中
- i:=m↑;
- m↑:= n↑;
- n↑:=i;

测试：参数传递——传地址

- 假设有一段程序如图所示，如果所有参数都是采取传地址的方式进行参数传递，它的输出结果是什么？

A. 5

B. 42

C. 7

D. 77

E.其他值

```
…
procedure  P(w,x,y,z);
begin
   y := y*w;
   z := z+x;
end
begin
   a := 5;
   b := 3;
   P(a+b,a-b,a,a);
   write(a);
end
```

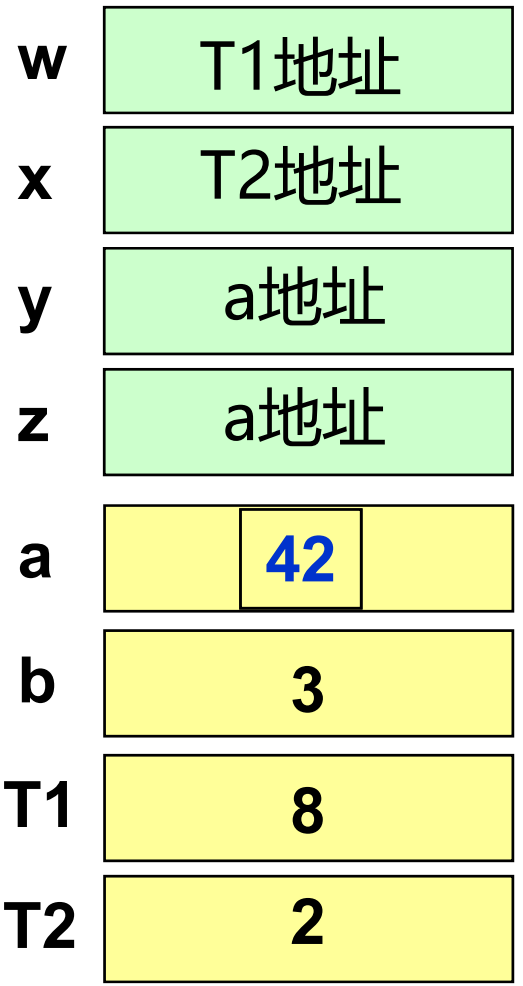
参数传递方式——传地址

- 把实在参数的地址传递给相应的形式参数
- 方法
  - 调用程序把实在参数(实参)的地址传递到被调用过程相应的形式单元中
  - 过程体对形式参数(形参)的引用或赋值被处理成对形式单元的间接访问

测试：参数传递——传地址

```
...
procedure  P(w,x,y,z);
begin
   y := y*w;
   z := z+x;
end
begin
   a := 5;
   b := 3;
   P(a+b,a-b,a,a);
   write(a);
end
```

输出：**42**



参数传递方式——传地址

- 把实在参数的地址传递给相应的形式参数
- 方法
  - 调用程序把实在参数(实参)的地址传递到被调用过程相应的形式单元中
  - 过程体对形式参数(形参)的引用或赋值被处理成对形式单元的间接访问

编译原理

参数传递方式——得结果

参数传递方式——得结果

- 传地址的一种变形
- 方法
  - 每个形参对应两个形式单元，第一个形式单元存放实参地址，第二个单元存放实参的值
  - 在过程体中对形参的任何引用或赋值都看作对它的第二个单元的直接访问
  - 过程完成返回前，把第二个单元的内容存放到第一个单元所指的实参单元中
- 有些Fortran采用这种方式

测试：参数传递——得结果

- 假设有一段程序如图所示，如果所有参数都是采取得结果的方式进行参数传递，它的输出结果是什么？

A. 5

B. 42

C. 7

D. 77

E.其他值

```
…
procedure  P(w,x,y,z);
begin
   y := y*w;
   z := z+x;
end
begin
   a := 5;
   b := 3;
   P(a+b,a-b,a,a);
   write(a);
end
```

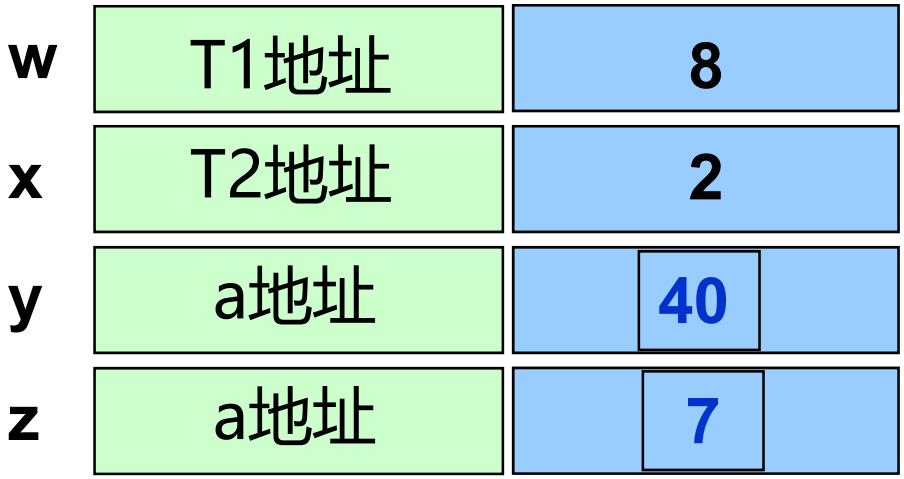
参数传递方式——得结果

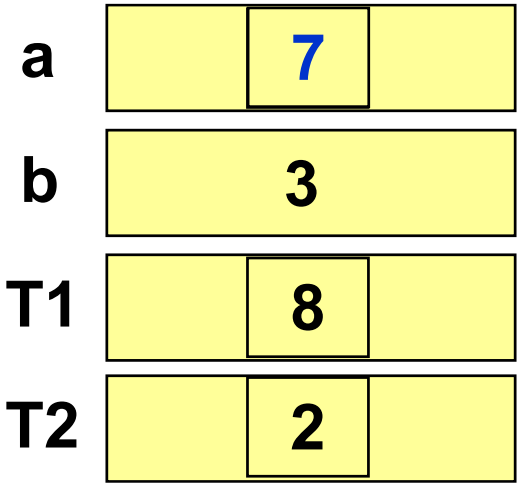
- 传地址的一种变形
- 方法
  - 每个形参对应两个形式单元，第一个形式单元存放实参地址，第二个单元存放实参的值
  - 在过程体中对形参的任何引用或赋值都看作对它的第二个单元的直接访问
  - 过程完成返回前，把第二个单元的内容存放到第一个单元所指的实参单元中

测试：参数传递——得结果

```
...
procedure  P(w,x,y,z);
begin
   y := y*w;
   z := z+x;
end
begin
   a := 5;
   b := 3;
   P(a+b,a-b,a,a);
   write(a);
end
```

输出：7





参数传递方式——得结果

- 传地址的一种变形
- 方法
  - 每个形参对应两个形式单元，第一个形式单元存放实参地址，第二个单元存放实参的值
  - 在过程体中对形参的任何引用或赋值都看作对它的第二个单元的直接访问
  - 过程完成返回前，把第二个单元的内容存放到第一个单元所指的实参单元中

编译原理

参数传递方式——传值

参数传递方式——传值

- 把实在参数的值传递给相应的形式参数
- 方法
  - 调用程序预先把实在参数的值计算出来，并传递到被调用过程相应的形式单元中
  - 被调用过程中，象引用局部数据一样引用形式参数，直接访问对应的形式单元
- PASCAL的值参数，C/C++的省缺参数传递

测试：参数传递——传值

- 假设有一段程序如图所示，如果所有参数都是采取传值的方式进行参数传递，它的输出结果是什么？

A. 5

B. 42

C. 7

D. 77

E.其他值

```
…
procedure  P(w,x,y,z);
begin
   y := y*w;
   z := z+x;
end
begin
   a := 5;
   b := 3;
   P(a+b,a-b,a,a);
   write(a);
end
```

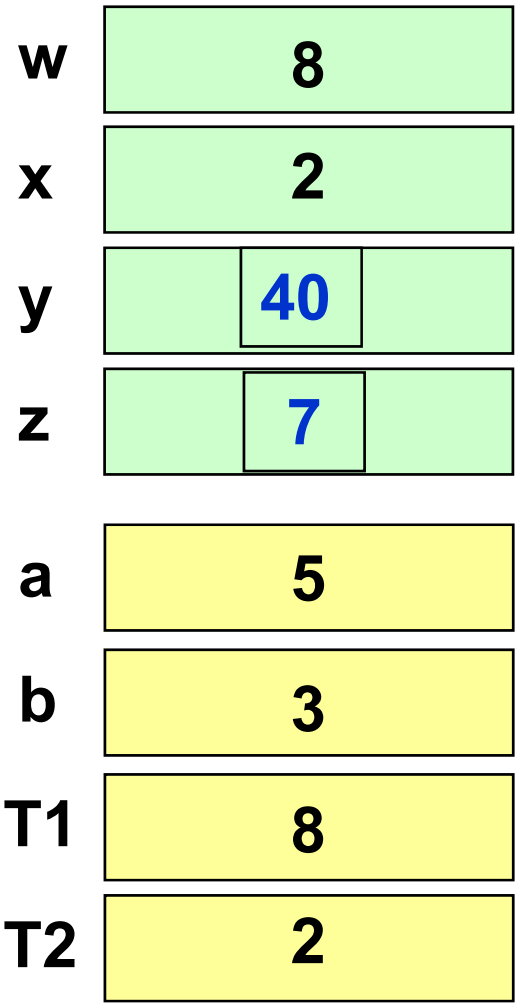
参数传递方式——传值

- 把实在参数的值传递给相应的形式参数
- 方法
  - 调用程序预先把实在参数的值计算出来，并传递到被调用过程相应的形式单元中
  - 被调用过程中，象引用局部数据一样引用形式参数，直接访问对应的形式单元

测试：参数传递——传值

```
...
procedure  P(w,x,y,z);
begin
   y := y*w;
   z := z+x;
end
begin
   a := 5;
   b := 3;
   P(a+b,a-b,a,a);
   write(a);
end
```

输出：**5**



参数传递方式——传值

- 把实在参数的值传递给相应的形式参数
- 方法
  - 调用程序预先把实在参数的值计算出来，并传递到被调用过程相应的形式单元中
  - 被调用过程中，象引用局部数据一样引用形式参数，直接访问对应的形式单元

编译原理

参数传递方式——传名

参数传递方式——传名

- 过程调用的作用
  - 相当于把被调用过程的过程体抄到调用出现的地方，但把其中出现的形式参数都替换成相应的实参
- 方法
  - 在进入被调用过程的之前不对实在参数预先进行计值，而是让过程体中每当使用到相应的形式参数时才逐次对它实行计值（或计算地址）
  - 通常把实在参数处理成一个子程序（称为参数子程序），每当过程体中使用到相应的形式参数时就调用这个子程序

参数传递方式——传名

```
PROGRAM EX
    …
    var A:integer;
    PROCEDURE P(B:integer)
        …
        var A:integer;
    BEGIN
        A:=0;
        B:=B+1;
        A:=A+B;
    END;
```

```
BEGIN
    A:=2;
    P(A);
    write(A);
END
```

```
BEGIN
    A :=2;
    TA:=0;
    A:= A +1;
    TA:=TA+ A;
    write(A);
END
```

测试：参数传递——传名

- 假设有一段程序如图所示，如果所有参数都是采取传名的方式进行参数传递，它的输出结果是什么？

A. 5

B. 42

C. 7

D. 77

E.其他值

```
…
procedure  P(w,x,y,z);
begin
   y := y*w;
   z := z+x;
end
begin
   a := 5;
   b := 3;
   P(a+b,a-b,a,a);
   write(a);
end
```

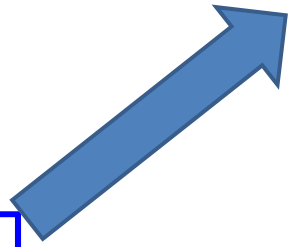
参数传递方式——传名

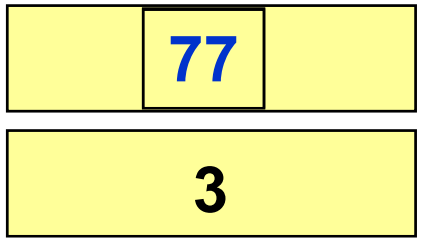
- 过程调用的作用
  - 相当于把被调用过程的过程体抄到调用出现的地方，但把其中出现的形式参数都替换成相应的实参
- 方法
  - 在进入被调用过程的之前不对实在参数预先进行计值，而是让过程体中每当使用到相应的形式参数时才逐次对它实行计值（或计算地址）
  - 通常把实在参数处理成一个子程序（称为参数子程序），每当过程体中使用到相应的形式参数时就调用这个子程序

测试：参数传递——传名

```
...
procedure  P(w,x,y,z);
begin
   y := y*w;
   z := z+x;
end
begin
   a := 5;
   b := 3;
   P(a+b,a-b,a,a);
   write(a);
end
```

输出： **77**






```
BEGIN
    a :=5;
    b:=3;
    a := a*(a+b);
    a:= a+(a-b);
    write(a);
END
```

参数传递方式——传名

- 过程调用的作用
  - 相当于把被调用过程的过程体抄到调用出现的地方，但把其中出现的形式参数都替换成相应的实参
- 方法
  - 在进入被调用过程的之前不对实在参数预先进行计值，而是让过程体中每当使用到相应的形式参数时才逐次对它实行计值（或计算地址）
  - 通常把实在参数处理成一个子程序（称为参数子程序），每当过程体中使用到相应的形式参数时就调用这个子程序

参数传递方式对比

```
...
procedure  P(w,x,y,z);
begin
   y := y*w;
   z := z+x;
end
begin
   a := 5;
   b := 3;
   P(a+b,a-b,a,a);
   write(a);
end
```

传值:　5
传地址:　42
得结果:　7
传名:　77

小结

- 参数传递
  - 传地址
  - 得结果
  - 传值
  - 传名

编译原理

目标程序运行时的活动

编译原理

过程的活动与生存期

目标程序运行时的活动

- 过程或函数是模块化程序设计的主要手段，也是节省程序代码和扩充语言的主要途径
- 过程定义

```
procedure  add(x, y : integer; var z : integer)
begin
      z := x + y;
end;
```

- 过程调用

```
add(a, b, c);
```

过程的活动与生存期

- 一个过程的活动指的是该过程的一次执行
- 过程P一个活动的生存期，指的是从执行该过程体第一步操作到最后一步操作之间的操作序，包括执行P时调用其它过程花费的时间
- 编译程序必须知道目标程序的运行时存储空间组织

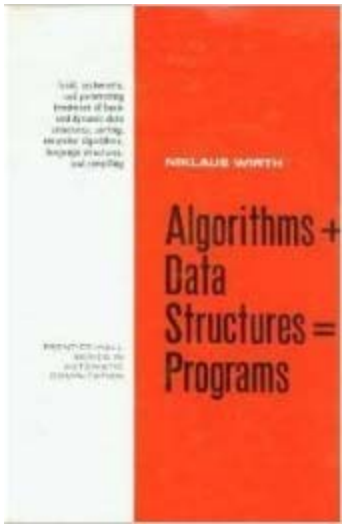
编译原理

运行时的存储组织

运行时存储器的划分

- 一个目标程序运行所需的存储空间包括
  - 存放目标代码的空间
  - 存放数据项目的空间
  - 存放程序运行的控制或连接数据所需单元

Niklaus Wirth



算法 + 数据结构
= 程序

编译程序组织存储空间须考虑的问题

- 过程是否允许递归?
- 当控制从一个过程的活动返回时，对局部名称的值如何处理?
- 过程是否允许引用非局部名称?
- 过程调用时如何传递参数；过程是否可以做为参数被传递和做为结果被返回?
- 存储空间可否在程序控制下进行动态分配?
- 存储空间是否必须显式地释放?

存储分配策略

- 静态分配策略
  - 在编译时能确定数据空间的大小，并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等
- 动态分配策略
  - 在编译时不能确定运行时数据空间的大小，允许递归过程和动态申请释放内存
  - 栈式动态分配、堆式动态分配
  - PASCAL，C/C++等

小结

- 过程的活动与生存期
- 编译程序组织存储空间须考虑的问题
- 存储分配策略
  - 静态分配策略
  - 动态分配策略

编译原理

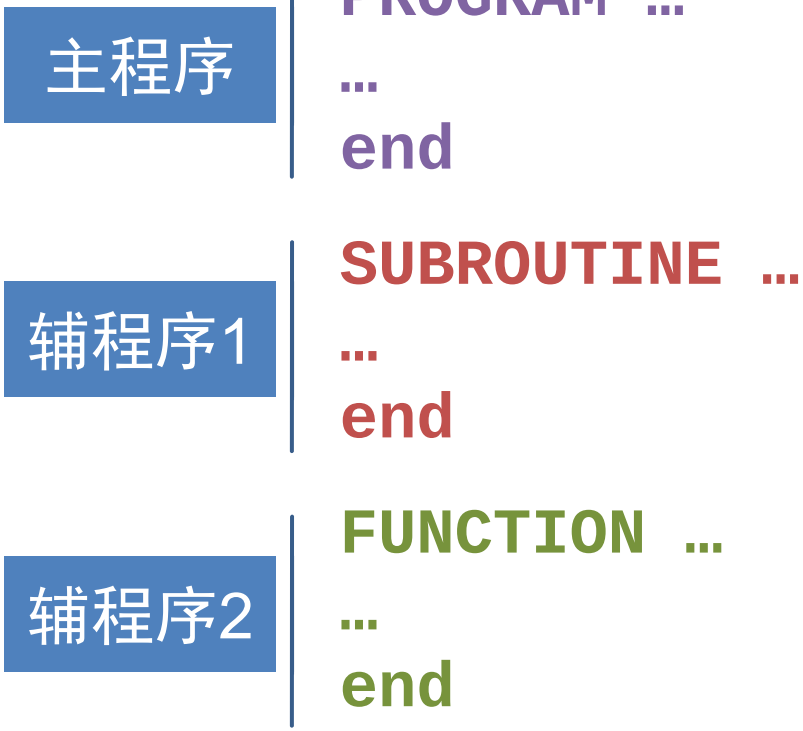
静态存储管理

存储分配策略

- 静态分配策略
  - 在编译时能确定数据空间的大小，并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等
- 动态分配策略
  - 在编译时不能确定运行时数据空间的大小，允许递归过程和动态申请释放内存
  - 栈式动态分配、堆式动态分配
  - PASCAL，C/C++等

FORTRAN程序结构

- 一个程序由一个主程序段和若干辅程序段组成
- 辅程序段可以是子程序、函数段或数据块
- 每个程序段由一系列的说明语句和执行语句组成，各段可以独立编译
- 模块结构，没有嵌套和递归
- 各程序段中的名字相互独立，同一个标识符在不同的程序段中代表不同的名字

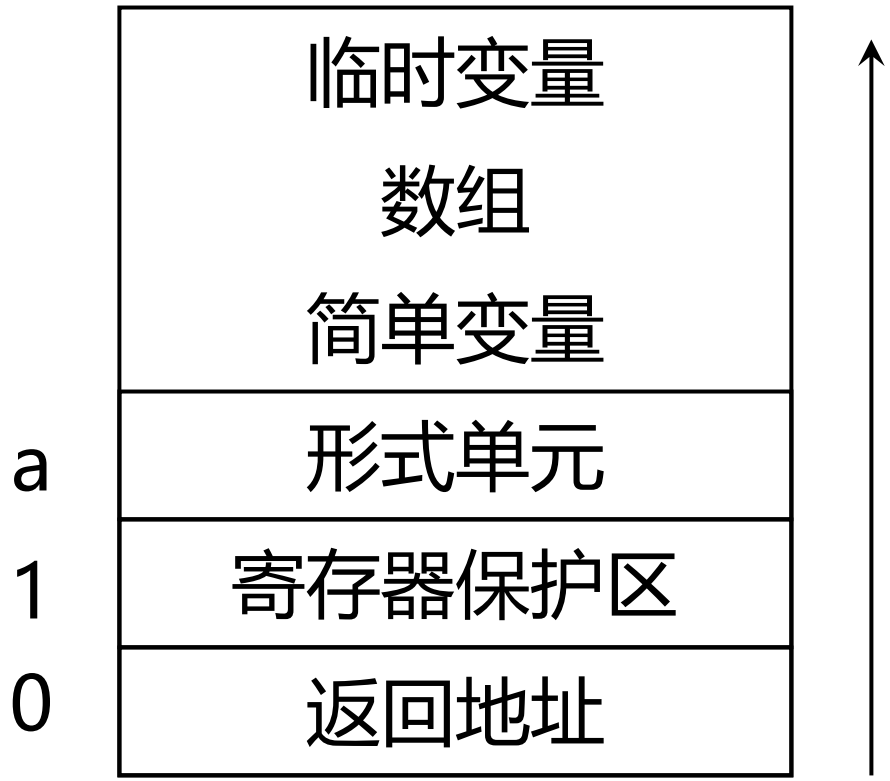


静态存储管理

- FORTRAN程序的特点
  - 整个程序所需数据空间的总量在编译时完全确定
  - 每个数据名的地址可以静态地进行分配
  - 各程序段可以独立编译，由链接装配程序把各段连成可运行的整体
- 按数据区组织存储
  - 每个程序段定义一个局部数据区，用来存放程序段中未出现在COMMON里的局部名的值
  - 每个公用块定义一个公用数据区，用来存放公用块里各个名字的值

FORTRAN的局部数据区

- 每个数据区有一个编号，地址分配时，在符号表中，对每个数据名登记其所属数据区编号及在该区中的相对位置
- 局部数据区的内容



FORTRAN的局部数

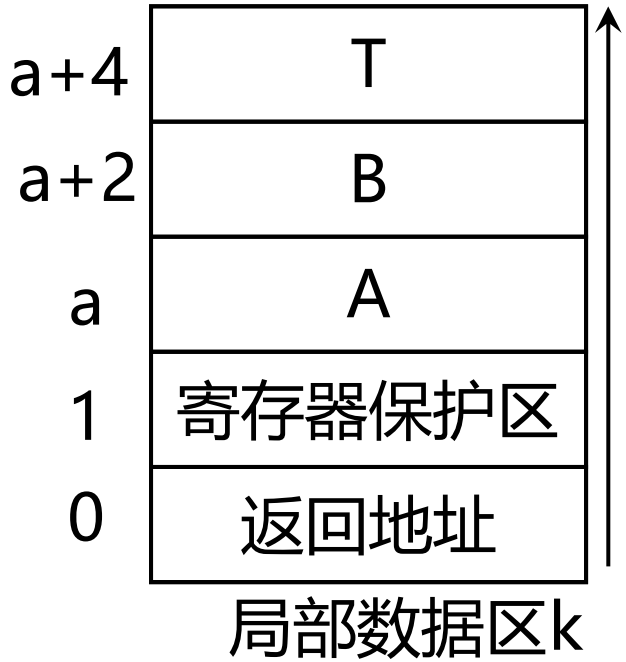
Alan J. Perlis

To understand a program you must become both the machine and the program.

- 考虑子程序段

```
SUBROUTINE SWAP(A,B)
T=A
A=B
B=T
RETURN
END
```

名字	性质	地址	
		DA	OFFSET
SWAP	子程序,二目		
A	哑元,实型	k	a
B	哑元,实型	k	a+2
T	实型变量	k	a+4



小结

- 静态存储管理
  - 采取静态存储管理的语言的特点
  - FORTRAN的局部数据区