内存管理模块 memAllocate

张雄

2013 年 12 月 30 日

有限元分析的数据量很大，需要高效率地使用计算机的内存。目前大多数高级语言都提供了分配和删除动态数组的能力，但频繁地分配和删除动态数组，会导致大量的内存碎片，造成内存的浪费，降低操作系统的运行效率。因此，一般的大型有限元软件均不采用这类动态分配和删除数组的方法，而是在程序开始运行时向操作系统获取一定数量的连续内存区域，并自行管理这块区域，在程序的运行过程中不再向操作系统申请新的大块内存区域。

STAP90 提供了内存管理模块 memAllocate，它定义了实型数组 A(MTOT)，从操作系统获取 4*MTOT 字节的连续内存区域，并对其进行管理。为了能够在这块区域同时存放各种类型的数据，定义了整型数组 IA 和双精度型数组 DA，并且通过 EQUIVALENCE 语句使得它们和实型数组 A 共享同一个内存区域。用户可以利用模块 memAllocate 提供的子程序 memAlloc(num, name, len, prec) 在这块内存区域中创建整型数组、实型数组和双精度数组，其中 num 为创建的数组号，name 为数组名 (5 个字符)，len 为数组元素的总数，prec 为数组类型 (1 为整型或单精度实型、2 为双精度型)。例如

```
CALL MEMALLOC(1, "ID   ", 3*NUMNP, 1)
CALL MEMALLOC(2, "X    ", NUMNP, ITWO)
CALL MEMALLOC(3, "Y    ", NUMNP, ITWO)
CALL MEMALLOC(4, "Z    ", NUMNP, ITWO)
```

分别创建了 1 号 ∼ 4 号数组，其中 1 号数组为整型数组，名称为"ID "，共有 3*NUMNP 个元素，用于存放各节点的标示数组；2 号 ∼ 4 号数组均为双精度数组，名称分别为"X "、"Y " 和"Z "，各有 NUMNP 个元素，用于存放各节点的 x、y 和 z 坐标。ITWO 为模块 memAllocate 定义的一个整型常量，表示一个双精度变量占用几个单精度变量的空间。一般情况下，ITWO 等于 2。

调用 memAlloc 子程序创建数组后，就可以通过指针数组 np 来获取所创建数组的第一个元素在数组 A、IA 或 DA 中的地址，且可以通过数组 aname、alen 和 aprec 分别获取该数组的名称、元素总数和数组类型。例如，STAP90 中的结点信息读入子程序

```
SUBROUTINE INPUT (ID,X,Y,Z,NUMNP,NEQ)
INTEGER :: NUMNP,NEQ,ID(3,NUMNP)
REAL(8) :: X(NUMNP),Y(NUMNP),Z(NUMNP)
```

要求调用它的程序为整型数组 ID(3,NUMNP) 和双精度型数组 X(NUMNP)、Y(NUMNP) 和 Z(NUMNP) 创建内存区域，这可以通过语句

```
CALL INPUT (IA(NP(1)),DA(NP(2)),DA(NP(3)),DA(NP(4)),NUMNP,NEQ)
```

来实现。

如果不再需要使用某个数组，可以调用 memFree(num) 子程序来删除它。memFree 子程序会自动移动位于被删除数组后面的所有数组，以避免产生内存碎片。为了减少内存移动操作，应首先删除最后创建

的数组，即尽可能按照与数组创建顺序相反的顺序来删除数组。另外，memAllocate 模块还提供了子程序 memFreeFrom(num)（删除从 num 开始的所有数组）和 memFreeFromTo(num1,num2)（删除 num1 至 num2 之间的所有数组）。

为了便于程序调试，memAllocate 模块提供了数组内容列表显示功能。memPrint(num) 输出第 num 号数组的内容，memPrintPtr(ptr, len, atype) 将内存区域 A 中从 ptr 所指向的位置开始，按照 atype 给定的格式 (0 为整型、1 为实型、2 为双精度型)，输出 len 个元素。例如，memPrint(1) 或 memPrintPtr(np(1), 3*NUMNP, 0) 将输出数组 ID 的内容，而 memPrint(2) 或 memPrintPtr((np(2)-1)*ITWO+1, NUMNP, 2) 则输出数组 X 的内容。另外 memInfo 将列出所有已经创建的数组的信息，包括数组号、数组名称、元素总数、数组第一个元素的地址和数组类型等。