文章编号:1000 - 582X(2002)06 - 0102 - 04

基于类模板链表结构的有限元程序设计*

袁政强，白绍良，李正良
（重庆大学 土木工程学院，重庆 400045）

摘　要:传统的有限元程序设计一般采用结构化的程序设计方法和结构化语言(如 FORTRAN),其数据存储形式都使用固定的数组结构,使得程序的扩展能力有限,代码的重复利用率低,调试复杂。模板是面向对象的 C++语言中相对较新的重要特性,模板能够快速建立类库集合,极大地提高了大型软件的开发速度。采用面向对象的程序设计方法,遵循有限元分析的本质,建立了有关描述有限元模型的类,用链表方式实现结点、单元和材料的数据存放、用多态性实现单元的自由链接,方便地实现了单元增减等用传统语言无法实现的功能。据此编制了有限元分析的数值计算程序,并给出了一个实例。结果表明,程序设计和调试周期较传统设计方法明显缩短,代码的利用率也明显提高。

关键词:面向对象;有限元;C++语言;类模板链表结构
中图分类号:TP311　　**文献标识码**:A

传统的有限元程序设计[1]一般采用结构化的程序设计方法和结构化语言(如 FORTRAN),其数据存储形式都使用固定的数组结构,使得程序的扩展能力有限,代码的重复利用率低,调试复杂。模板是 C++语言中相对较新的重要特性,模板能够快速建立类库集合,极大地提高了大型软件的开发速度。C++语言[2]对程序处理的数据有明确要求,对实数进行处理的过程不能用于对整数进行处理。使用模板可以将所处理的数据类型说明为参数,由参数的改变可以使程序对任何数据类型进行同样方式的处理。链表是用于存储数据的结构,一般链表的数据类型是固定的,类模板链表的数据类型参数可以将任何类型的数据放入链表,这种数据可以是自定义的单元数据。C++的动态联编技术可以让由同一个类派生出的多种子类使用共同的父类作为类型数据放入同一链表中。文章给出单元的抽象类定义,其他的具体单元是抽象单元类的派生类,如:常用的四节点等参单元、三结点等参单元等。各类型单元可以混合的次序放入类模板链表中。每个单元中包含有一个存放结点地址的链表,它是单元结点地址与结点地址总链表的对应关系。一些文献[3]中虽已给出了面向对象的有限元[4,5]方法,但在结点,材料和单元的处理上仍沿用传统的数组形式。笔者采用 Visual C++上实现了二维平面应力问题计算。结果表明,程序设计和调试周期较传统设计方法明显缩短,代码的利用率也明显提高。

1　程序设计

类模板链表结构(见图 1)是生成集合类的模板,它由链表结点类模板、链表类和链表迭代类组成。链表结点类模板是集合类的一个元素,此元素由放入的数据对象和链接指针组成。链表类是将链表元素类用指针链接的类,在链表类中给出了有许多用于处理链表元素的函数。具体的定义可参见[6]。

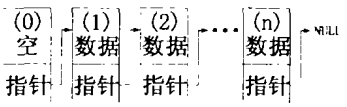


图 1　模板链表结构

按照有限元(FEM)的分析方法,有限元分析的主要数据包括:1)描述有限元分析的整体数据,如单元总

* 收稿日期:2002 - 01 - 20
基金项目:国家自然科学基金(59378351)
作者简介:袁政强(1962 -),男,贵州贵阳人,重庆大学副研究员。主要从事工程力学计算和钢筋混凝土结构研究。

数、结点总数、问题的维数、材料种类数等；2)结点数据，包括结点坐标、结点自由度(结点参量)、结点力、边界条件等；3)单元数据，包括单元联络性数据、单元类型、高斯积分点数据和单元所用材料数据等；因此，按照面向对象的程序设计方法，相关的类主要有：1)结点数据类和相关的方法，将结点地址放入模板链表得到相应的结点地址链表，链表长度就是结点总数；2)单元数据类和相关的方法，由不同单元组成相应的单元链表，长度表示单元总数，不再需要存放各种类型的单元个数；3)材料数据类和方法，同样得到材料链表。结点是组成单元的主要数据，单元中的结点链表是结点总链表的子链表。一般有限元采用的对应方式是给出单元局部结点编号与结点总体编号的对应数组，这种对应方式会因局部的总体编号改变而需要改变全部的单元对应数组。采用地址存储方式，在结点链表中的数据是结点的地址，同时单元中局部结点链表存储的也是结点地址。如果在总结点链表中增加结点，不论结点增加在总链表末尾还是插入在总链表中间，都不会改变单元局部结点链表中的数据。图 2 中虚线是单元局部链表与总体链表之间的地址对应关系。

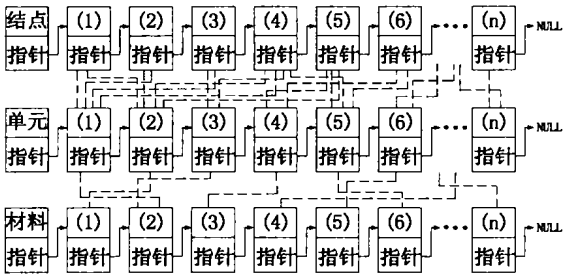


图 2　FEM 分析的结点集合、单元集合、材料集合之间的关系图

1.1　FEM 结点数据类

结点数据包含了结点约束、结点坐标、结点力向量。而其中用到的类模板向量和类模板矩阵见文献[7]。

```
class NodeData{
    protected:
    int Ind; //结点指示值(序号)
    //结点在方程中的开始编号,程序自动计算
    unsigned int NEquation
    int NodeFree; //结点自由度数
    int Dim; //空间维数
    //用于指示自由度的约束状态
    Vector < int >  * BC
    //结点坐标、边界条件值
    Vector < float >  * Coor, * BCV
    //开始时是结点力,解方程以后是结点位移
    Vector < float >  * Force
    //非线性分析和动力分析使用的存放的速度、加速度等值
    Vector < float >  * Value[5]
    public:
    //构造函数
    NodeData:: NodeData ( int  iInd, int  iNodeFree, int iDim);
    ~ NodeData(){delete BC;delete Coor;}; //析构函数
    public:
    int GetInd(){return (Ind); };
    //取得结点序号
    //根据序号返回本结点地址
    NodeData  * GetThis(int iInd)
    void outputBC();
    void outputCoor();
    void outputForce(char  * s);
    void SetNEquation(int iNeq){NEquation = iNeq;};
    unsigned int GetNEquation(){return NEquation;};
    Vector < int >  * GetBC(){return BC;};
    Vector < float >  * GetForce(){return Force;};
    void SetForce(Vector < float >  * F){Force = F;};
    //为结点数据分配存储空间
    void Init(int iDim,int iNodeFree)
    //取得结点自由度数
    int Get(){return(NodeFree);}
      Vector < float >  * GetCoor(){return Coor;};
      friend class FinMethod;
    };
```

值数组 Value[8] 根据求解的问题放入值的内容，坐标、力、位移值是确定的，温度、速度、加速度和历史数据，根据问题的需要申请。在不需要时，存放 NULL。

1.2　材料数据类

材料数据包括的内容，根据材料类型的不同而不同。值放在 Mate 数组中。Index 用于指示引用本材料的单元的计算空间和对应的自由度位置。

```
class Material{
protected:
int iInd; //材料指示数
int type; //材料类型
```

```
//存放杨氏模量、泊松比、材料密度、二维单元的厚度
Vector < float > * Mate
public:
Material(int Ind); //输入材料数据
void Output(); //输出材料数据
int GetInd(); //取得材料指示数
Vector < float > * GetMate();//取得材料类型
float GetEA(); //取得材料类型
float GetEI(); //取得材料类型
//根据指示数取得材料指针
Material * GetThis(int Ind)
};
```

1.3 单元数据抽象类与四结点等参单元类

抽象类所表达的概念广泛,不是一种具体的对象,它的唯一用处是为其它类提供基类,其它类可以从这里继承共有接口。而"一个接口,多个算法"就是所谓的多态性。比如:平面等参单元和平面梁单元的刚度矩阵的计算是不相同的。用抽象单元的单元刚度矩阵计算一个接口,可实现不同单元刚度矩阵的计算。只要调用的是单元刚度计算函数,不必特意选择,编译程序会动态的根据单元的类型自动选用相应的具体函数去计算,这种动态的由编译程序来决定调用函数的过程叫动态联编。

抽象单元的虚函数就是实例类单元的接口,虚函数是不用编写的。抽象单元类定义:

```
class Element{
protected:
int iType; //单元类型指示
Material * pMate;//指向材料类的指针(材料数据)
//指向结点类的指针(单元联络性数据)
NodeData * pNode
public:
Element(int Type,Material * pM,int NodeNum);
void OutputElement();
NodeData * * GetNode(){return (pNode);};
int GetNodeNum(){return iNodeNum;};
void SetpNode(int i,NodeData * p){pNode[i] = p;};
//虚函数计算单元刚度矩阵并放入总刚度矩阵
virtual void GenStiffMatrix(Matrix < float > &) = 0
//虚函数计算结点应变
virtual void GetStrain() = 0
//虚函数计算 Guass 点应变和应力
virtual void GetGaussPoint() = 0
//虚函数计算结点应力
virtual void GetStress() = 0
};
```

成员变量 pMate 是指向材料数据的地址,如果单元在迭代计算过程中需要对单元材料作退化处理时,可以复制指向的材料单元并将指针指向复制的单元,在复制的单元内作材料参数的退化改变。* pNode 单元与结点的联络性数组,数组内存放的是结点地址。存放地址不会因结点链表中部元素的增加改变插入以后结点的地址,方便了结点的增加和以后单元的增加。四结点等参单元和平面梁单元的定义:

```
//四结点四边形单元
class Concrete4: public Element{
public:
Concrete4(int Type,Material * pM,int NodeNum):
Element(Type,pM,NodeNum){};
//计算单元刚度矩阵并放入总刚度矩阵
void GenStiffMatrix(Matrix < float > &)
//计算单元质量矩阵并放入总质量矩阵
void GenMassMatrix(Matrix < float > &)
void GenStrain(); //计算结点应变
void GenGaussPoint(); //计算 Guass 点应变和应力
void GenStress(); //计算结点应力
};
```

在具体类型单元中都有与虚函数同名的函数,这些函数是需要具体编写的。

1.4 有限元方法主类和问题派生类

有限元主类包含有限元分析的主要数据和实现方法。它的定义如下

```
class FinMethod{
private:
List < NodeData > * HeadNode; //结点链表
List < Element > * HeadElement; //单元链表
List < Material > * HeadMate; //材料链表
Vector < float > * F; // 求解方程时的右端向量
Matrix < float > * K; // 刚度矩阵
//方程总数,根据结点数和结点约束自动计算
unsigned int Nequs
public:
FinMethod();
int Counter();// 计算结点各自由度的方程编号
void CreateForce();//计算力向量
int outputElement();// 输出单元数据
void GetStiffMatrix();// 计算单元刚度
int solver();//方程求解
//计算高斯点的应变和应力并输出
int FinMethod::GenGaussPoint()
};
```

面向对象的程序设计注重程序的重用技术[8]。给出的刚度矩阵函数是对任何空间维数和不同自由度都通用的函数。刚度矩阵函数如下：

```
void FinMethold::GetStiffMatrix(){ // 计算刚度矩阵
// 在计算刚度矩阵之前,将刚度矩阵 K、结点力 F 置零
K->ReSet(); F->ReSet();
//说明链表迭代类
ListIterator<Element> lei(*HeadElement)
  while(lei.NextNotNull())
//此语句是抽象单元的接口处,计算矩阵只改此处
lei.Next()->GetStiffMatrix(*A,*F)};
```

有限元方法类是一个基本方法类,对于具体求解问题的类可由有限元方法类派生。弹性静力分析方法：

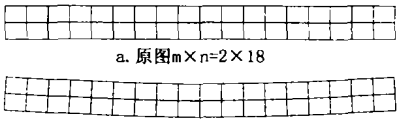
```
class F-Static: public FinMethold{
public:
//静力分析方法类的构造函数
F-Static(Graph &g)
  F-Static(int dim,int free):FinMethold(){
    Type = P-STATICS; Dim = dim; NodeFree = free;};
  int Anlysis();// 静力分析程序
};
void F-Static:: Anlysis(){
  Counter(); //计算结点数和方程编号
  outputElement(); //单元数据输出
  CreateForce();//读入结点作用力项
  GenStiffMatrix();//计算单元刚度矩阵
  solver(); //线性方程求解
//输出高斯点应力应变,梁的轴力、剪力、弯矩
GetGaussPoint()
};
```

2 类的实例化

以上建立了有限元分析所需的主要类和方法,下面给出一个实例化的例子。

```
#include"FEM.h"
void main(){
  Graph *g;
  g=new Graph();
  a1.SetP(0.,-.4);a2.SetP(9.,-.4);
  a3.SetP(9.,.4);a4.SetP(0.,.4);
g->retdivR(2,18,a4,a1,a2,a3,1,E-QUADR-4);
  F-Static *a=new F-Static(2,2);
 wMate=new M-PlanStressE(1,3.e10,0.167,1.,.3);
  a->HeadMate->InsertLast(wMate);
// 给边界约束条件
a->CreateBC(a1,a4,dect,u)
  dect[0]=0;
// 给边界约束条件
a->CreateBC(a2,a3,dect,u)
  dect[1]=2;
  u[1]=-100000.;a2=(a3+a4)/2.;
// 在 a2 点作用 -100KN 给结点力
a->InputNodeForce(a2,dect,u)
  a->StaticAnlysis(); // 计算程序开始
};
```

简支梁中点作用集中力,弹性模量是 3.0E4 MPa,柏松比是 0.167。图 3 是简支梁计算的剖分图和变形图。



a. 原图m×n=2×18

b. 变形图

图 3　简支梁剖分图及变形图($m \times n$)=2×18

3 结 语

以上给出了有限元分析的基本结构类,在有限元分析中还牵连许多问题,如求解线性方程组和非线性方程组的方法,这些都在矩阵类中定义。矩阵的一维对称压缩变带宽存储和一维非对称压缩变带宽存储以及求解方法也在矩阵类中给出。在文中给出的有限元方法类是最基本的方法类,具体问题(如:塑性分析、非线性弹性分析、稳定分析、流体分析和污染物扩散分析)类可在基本方法类下继承,这使得加入具体问题的求解变得容易。在塑性问题分析中,应力历史和平均塑性应变的存储在一般的过程结构化语言设计中相当困难,而在 C++ 中只要在新的塑性单元类中加如相应变量即可,这已在我们的集成软件中实现。

参考文献：

[1] ZIENKIEWICZ O C. The Finite Element Method, 3rd Ed[M]. McGraw:Hill Book Company(UK) Ltd,1977.

[2] HEKMATPOUR S. C 程序员的 C++ 指南(阎尤泽译)[M]. 北京:北京航空航天大学出版社,1992.

[3] 张向,许晶月. 面向对象的有限元程序设计[J]. 计算力学学报, 1999,16(2):75-80.

[4] FORD B W R, FOSCHI R O, STIENER S F. Object-oriented finite element analysis[J]. Computers and Structures, 1990,35:355-374.

[5] SCHOLZ S P. Element of an object-oriented FEM program in C++[J]. Computers and Structures, 1992,43:517-529.

[6] 殷人昆.数据结构-用面向对象方法与 C++ 描述[M]. 北京:清华大学出版社,1999.

[7] 程纬,曹定华,张诚坚.通用矩阵与矢量模板类的分析与应用[J].湖南大学学报,1999,26(5):97-101.

[8] 孙大烈,杨静,王子立.基于对象的软件重用技术研究[J].哈尔滨建筑大学学报,2000,33(3):111-113.

(下转第 108 页)

4 结 语

从计算实例成果分析可以看出,利用文中所述的分析方法及具体实现步骤可以方便、准确地在 Super - Sap 中进行三维弹塑性有限元分析,并且可以快捷地实现土工有限元的弹塑性计算与计算结果的可视化。

参考文献:

[1] 朱以文,韦庆如,顾伯达.微机有限元前后处理系统 VisiCAD 及其应用[J].北京:科学技术文献出版社,1993.

[2] LADE P V, DUNCAN J M. Elastoplastic Stress - Strain Theory for Cohesionless Soil[J]. Journ. Gotech. Engrg., 1975, 101(10):1 037 - 1 053.

[3] DESAI C S, ZAMAN M M. Thin - layer element for interfaces and joints[J]. Int. J. Num. Analy. Met. Geotech. 1984, 8(1):19 - 43.

[4] 殷宗泽,宋泓,许国华.土与结构材料接触面的变形及数学摩拟[J].岩土工程学报,1994,(3):14 - 21.

[5] 周兆凯,周毓萍.CAD 软件与有限元软件的集成技术研究[J].武汉水利电力大学学报,1999,32(2):94 - 96.

[6] 孙钧,汪炳鉴.地下结构有限元分析[M].上海:同济大学出版社,1984.

Accomplishment of Elastic - plastic Analysis in Geotechnical FEA

LIU Ya - lian[1], *LI Yong - jian*[2], *QIAN Ping - yi*[1]

(1. Guangdong Technology Institute of Hydraulic and Electric Engineering, Guangzhou 510635, China;
2. Wuhan University, School of Geotechnical and Civil Engineering, Wuhan 430072, China)

Abstract: In order to realize elastic - plastic analysis in geotechnical FEA, in this paper, firstly, the method how to realize elastic - plastic analysis in Super - Sap is discussed. Then, this paper stresses to introduce the principle and the steps used to achieve the accomplishment of elastic - plastic analysis in Super - Sap. Finally, an engineering practice is given to prove that the geotechnical FEA elastic - plastic analysis and visualized post - processing can come true by it rapidly.

Key words: geotechnical FEA; elastic - plastic analysis; visualized post - processing

(责任编辑 姚 飞)

(上接第 105 页)

Finite Element Programming Based on Template Chain Structure

YUAN Zheng - qiang, *BAI Shao - liang*, *LI Zheng - liang*

(College of Civil Engineering, Chongqing University, Chongqing 400045, China)

Abstract: Traditional Finite Element Method is the structuralization programming method and the structuralization language (such as FORTRAN). Its data store uses fixed array structure, which makes the expanding ability of program limited, code's reuse ratio low, debugging complex. The template is an opposite new important characteristic of the object - oriented programming language C + +. The template can quickly establish class library so as to increase greatly the development speed of the large software. The object - oriented programming concept is applied to finite element method. According to the nature of the finite element analysis, the classes and their methods, which describe virtual element, node, material etc., have been developed and implemented using the object - oriented programming language C + +. The elements, nodes and materials are stored by chain. Many type's elements are stored mixedly by polymorphism characteristic. The program is implemented and a instance is given to show the programming of finite element method. The results show that the period of programming and debugging is obviously shorter than that of traditional method; the code's utilization ratio is also increased obviously.

Key words: object - oriented; finite element method; C + + language; template chain structure

(责任编辑 姚 飞)