大韓土木學會論文集
第14卷 第1號 · 1994年 1月
pp. 119~129

構 造 工 學

뼈대構造物의 自由振動解析을 위한 客體指向 C++ 프로그램

An Object Oriented C++ Program for Free Vibration Analysis of Framed Structures

申英澄* · 徐鎭國**
Shin, Young Shik · Suh, Jin Kook

Abstract

This paper describes a C++ free vibration analysis program of structures for personal computer. This program was developed by object oriented programming method which is the latest trend in programming practice. The object-oriented programming method which has the superior reuseability and expansibility to procedural programming provides various choice of menus and easy modification of the program, and reduces the development time and size of the program. This object-oriented free vibration analysis program written in C++ language consists of Vector and Matrix classes, Structural Analysis and GUI classes. The efficiency and validity of the program were examined by solving several numerical examples. The static and free vibration analyses of the framed structures were satisfactorily performed by this program on a personal computer.

要 旨

본 연구에서는 기존의 구조적 프로그래밍 방법의 단점들을 개선한 새로운 객체지향 프로그래밍기법으로 구조물의 정적해석과 자유진동해석이 가능한 PC용 C++ 구조해석 프로그램을 개발하였다. 객체지향 프로그램은 기존의 구조적 프로그램에 비하여 프로그램의 재사용성 및 확장성이 뛰어나므로 프로그램의 수정과 개발이 용이하고 다양한 기능과 메뉴선택을 제공하며 소프트웨어의 개발시간과 프로그램의 크기를 줄일 수 있다. 본 연구에서 개발한 객체지향 자유진동해석 프로그램은 벡터 및 행렬의 연산을 수행하는 VECTOR 및 MATRIX클래스, 뼈대 구조물의 자유진동해석을 위한 STRU클래스 및 사용자 환경을 위한 GUI클래스로 구성되며 모든 프로그램은 객체지향 언어인 C++로 작성되었다. 여러가지 예제해석을 통하여 본 연구에서 개발된 객체지향 C++ 자유진동해석 프로그램의 효율성과 타당성을 검증하였다.

* 正會員 · 嶺南大學校 工科大學 土木工學科, 副教授
** 正會員 · 嶺南大學校 工業技術研究所, 研究員

1. 서 론

지난 40여년간 유한요소법이 복잡한 공학문제를 해결하는데 폭넓게 사용되어 오면서, 구조적 프로그래밍 기법(structured programming method)을 통한 많은 유한요소 구조해석 프로그램들이 개발되었다. 그러나 널리 사용되고 있는 SAP[1]과 같은 범용 프로그램(general purpose program)들은 비교적 다양하고 정확한 구조해석 기능에 비해 그 구조가 매우 복잡하여 자료의 입출력이 불편하고, 소형 구조해석 및 부분적인 실행의 경우에도 프로그램 전체를 실행시켜야 하며, 부분적인 프로그램의 수정이 용이하지 않는 등 유지 및 보수에 어려움이 많다. 최근 10여년간 개인용 소형컴퓨터(personal computer)의 폭넓은 보급과 기억용량 및 처리속도의 향상으로 이러한 범용 프로그램의 단점을 부분적으로 개선한 PC용 프로그램의 개발이 촉진되고 있으나, 지금까지 개발된 PC용 구조해석 프로그램들[2-5]은 범용 프로그램과 마찬가지로 FORTRAN과 같은 절차적 언어(procedural language)로 작성되어 있어 기존의 절차적 프로그래밍 기법(procedural programming method)의 소프트웨어(software)의 재사용성(reuseability) 및 모듈화(modulization) 등의 문제점, 그리고 프로그램 구성요소의 첨가 및 부분적인 수정시에도 전체 프로그램을 검토해야 하는 문제 등 개발된 소프트웨어의 유지 및 보수에 효과적이지 못한 것으로 알려져 있으며, 계산집약적인 구조해석 프로그램의 특성으로 인하여 계산효율의 향상에만 집중적인 노력을 기울여 왔다. 따라서 앞으로의 구조해석 프로그램의 개발은 계산효율의 향상 뿐 아니라 프로그램의 수정, 유지관리 및 기능확장에 대한 최대한의 신뢰도를 확보하는데 초점을 맞추어야 한다. 1970년대에 제안된 C나 Pascal과 같은 구조적 프로그래밍 언어는 프로그램의 유지와 보수의 여러 문제들을 해결함과 동시에 하드웨어(hardware)적인 부분까지 제어할 수 있게 하고, 크기가 작으면서도 속도가 빠른 프로그램을 생성할 수 있게 하는 등 프로그램의 간결성과 신뢰성 그리고 유지의 용이성에 많은 향상을 가져왔다. 그러나 이러한 프로그래밍 원리에 있어서 여전히 난제로 남아있던 대규모 소프트웨어의 개발한계를 극복하기 위하여, 문제해결을 언어의 절차적인 접근방법에 맞추기보다는 언어를 문제해결 방식에 맞추는 방법, 즉 자료(data)의 형태를 해결해야 할 문제의 본질적인 특징에 대응할 수 있도록 설계하는 방법인 객체지향(object-oriented) 소프트웨어 개발방법론이 등장하게 되었다. 특히 여러 객체지향 프로그래밍 언어중 본 연구에서 사용하는 C++언어[6]의 포인터(pointer)에 의한 수치연산을 적절히 사용하면, 기존의 절차적 언어로 된 프로그램과 비교할 때 계산효율면에서도 우월하다고 알려지고 있다.[7,8] 따라서 객체지향 프로그래밍[9]은 기존의 절차적 프로그래밍의 장점을 수용하면서 그 문제점까지 해결할 수 있는 프로그래밍 방법이라 할 수 있다.

객체지향 프로그램에 대한 연구는 1980년대 중반 전산공학 분야에서 제안되어 최근 여러 분야에서 연구가 활발히 진행되고 있는데, 토목 및 건축분야에서는 1993년 개최된 제5차 International Conference on Computing in Civil & Building Engineering[10]에서 주로 독일의 연구자들을 중심으로 객체지향적 개념을 도입한 그래픽 사용자 접속장치(graphic user interface)를 포함한 AutoCAD와의 연계연구 및 시공계획, 공정관리 및 시공 정보관리 등 시공관리 프로그램에의 응용, 통합 구조설계시스템의 구축, 구조설계 통합시스템을 위한 자료관리에 관한 연구 등 데이타베이스(database) 구축에 관한 많은 연구보고가 쏟아져 나와, 이 분야에서의 객체지향 프로그래밍 기법에 대한 많은 관심이 집중되고 있음을 반영하고 있다. 1990년 Forde 등[8]이 Object-NAP이라는 언어를 사용하여 유한요소해석 프로그램의 개발에 관한 연구를 발표한 이래 구조해석에 관한 연구도 활발히 진행되고 있는데, Fenves,[11] Zimmermann 등[12,13]의 연구에서 Smalltalk[14]와 같은 객체지향적 언어가 유한요소해석을 다룰 수 있음을 보여주었고, 그 중 Fenves는 객체지향언어로 유한요소해석의 전처리(preprocessing) 프로그램의 응용가능성을 보여주었다. C++언어를 사용한 연구는 Mackie[15]의 연구와 Timoshenko보에 근거한 유한요소해석 프로그램을 발표한 Scholz 등[16]의 연구가 있으나 아직은 유한요소해석 프로그램에 대한 피상적인 개념설명과 기초적인 연구에 불과하다고 할 수 있다. 국내에서도 객체지향 프로그래밍 기법의 효율성을 인식하여 이를 이용한 연구가 최근에 수행되기 시작했는데, 김치경 등[17,18]과 천진호 등[19]의

연구에서 데이타베이스 관리시스템과 통합 데이타베이스에 의한 구조물 설계에 관한 데이타베이스 구축 및 관리에 대한 통합 설계시스템에 관한 것들로서 유한요소 구조해석에 관한 응용은 본 연구자들에 의한 연구[20-22]외에는 전무하다.

따라서 본 연구에서는 PC용 유한요소 구조해석 프로그램의 개발에 관한 연구의 일부로 정적해석 및 자유진동해석이 가능한 객체지향 C++ 구조해석 프로그램을 개발하고, 예제해석을 통하여 이 프로그램의 타당성과 효율성을 검증하고자 한다.

2. 객체지향 프로그래밍

객체지향 프로그래밍 기법을 사용한 프로그램의 목표는 먼저 독립적이며 재사용가능한 소프트웨어 구성요소인 객체(object)를 생성하는 것이다. 이는 대규모의 응용 프로그램을 개발하고자 할 때 각 소프트웨어의 구성요소를 병렬응용범위에서 설계할 수 있고, 비슷한 응용에 대해 기존의 코드(code)를 재사용할 수 있으며 프로토타입(prototype)을 효과적으로 형성할 수 있어서 소프트웨어 개발시간과 크기를 현저히 줄일 수 있음을 의미한다. 자료 흐름방식이나 수학적 논리를 기초로 하고, 프로그램 개발자에게 자료와 처리과정(procedure)을 분리하여 표현하거나 그것들의 상호관계를 분리하여 관리하게 하는 기존의 절차적 프로그래밍 방법과는 달리 객체지향 프로그래밍에서는 함수들, 처리대상이 되는 자료들과 처리과정들이 캡슐화(encapsulation)되어 있는 객체로 취급하여 응용하고자 하는 대상을 직접 모델링할 수 있다. 따라서 문제범위에 있는 객체의 성격을 파악하고 그 객체를 구현하는 과정이 프로그램 작성과정이 된다. 객체들은 서로 메시지(message)를 주고 받음으로써 정보를 교환하며, 다른 객체로 부터 받은 메시지를 분석하여 지시된 명령을 수행하게 된다.

추상적 자료형식, 즉 동일한 종류의 자료와 처리과정을 포함하는 객체들의 집합을 클래스(class)라고 하는데, 표준 Pascal 또는 C에서의 변수들이 주어진 자료형식의 인스턴스(instance)인 것과 같이 객체는 클래스의 인스턴스라고 할 수 있다. 이러한 클래스 개념은 종속된 처리방법들을 제공하는 확대된 레코드 개념(record concept)이라고 볼 수 있으며, 해당 객체 형식의 유기적 조합을 나타내는 형틀(template)이라고도 볼 수 있는데, 이 형틀은 보통 일련의 인스턴스 변수와 클래스에 의해 인식되는 일련의 메시지인 프로토콜(protocol)을 포함한다. 각각의 클래스는 다른 클래스로부터 전달되는 명령들을 주고 받을 수 있는 인터페이스(interface)를 갖고 있다. 또한 클래스에 종속된 객체들의 명령수행방법과 정보처리방법을 처리방식(method)이라 하는데, 이 처리방식은 각 객체에 저장된 인스턴스 변수의 값을 조절한다. 클래스는 클래스간의 계층관계를 통해 그 상위 클래스(superclass)로 부터 자료와 처리방식들을 상속할 수 있는데, 이러한 상속성(inheritance)은 객체가 주어진 응용에 적절하게 맞춰 새로운 객체를 형성할 수 있음을 의미한다. 즉, 상속이란 이미 존재하는 객체와 거의 유사하나 조금 다른 성질을 갖는 새로운 객체를 정의하는 능력을 말하며, 상위 클래스의 정보들을 찾아 하위 클래스(subclass)에서 사용하거나, C++의 컴파일러(compiler)와 같이 상위 클래스의 코드를 하위 클래스에서 복사하여 공유하게 되어 재사용성을 증대시킨다.

객체의 일부분인 처리과정은 프로그램 전체에 대한 것이 아니므로 서로 다른 객체는 같은 이름의 처리과정을 가질 수 있다. 이와 같이 각기 다른 객체들이 동일한 메시지에 대해 각각 다른 반응을 취하는 것을 다형성(polymorphism)이라 하며, 이것은 시스템내에 이미 존재하는 메시지에 대해 응답할 수 있는 새로운 객체를 시스템에 삽입하는 것이 용이하다는 것을 의미한다.

이와 같이 프로그래밍이 간편하고, 프로그램의 확장 및 수정이 매우 간단하며, 신뢰성 및 재사용성이 뛰어난 객체지향 언어들은 Xerox Palo Alto Research Center에서 개발된 첫번째 객체지향 언어인 Smalltalk 이후로 많이 개발되었다. 그 중 C 기반언어인 C++는 Simula, Smalltalk 등과 달리 복합언어(hybrid language)로서 compact하고 실행속도가 빠르고 이식성이 강하며 어셈블러(assembler)와 연관하여 미세한 제어를 할 수 있는 등 기존의 C언어의 모든 기능들을 그대로 사용할 수 있으며, 동시에 데이타 추상화(data abstraction), 연산자 중복, 단일 및 다중상속 지원 등 객체지향적 개념들을 지원하고 있다. 일반적으로 C++로 작성된 프로그램은 처리속도가 표준 C 프로그램만큼 빠르며, 동적

바인딩(dynamic binding)과 재사용성으로 인해 동등한 기능을 가진 C 프로그램보다 크기가 작다.

3. 객체지향 C++ 자유진동해석 프로그램

본 연구에서 개발된 객체지향 자유진동해석 프로그램은 독립적으로 사용되거나 또는 다른 클래스에 상속되어 사용되는 VECTOR 및 MATRIX클래스와 실제 구조해석을 수행하는 STRU클래스, 그리고 프로그램과 사용자를 연결시켜 주며 사용자와 컴퓨터 간의 정보교환에 더 편리한 입출력 환경을 제공하는 GUI(Graphic User Interface) 클래스 등으로 구성되어 있다.

3.1 VECTOR 및 MATRIX클래스

이 클래스들은 구조해석과정중 적어도 한번이상 사용되는 클래스들로서 고유치 문제의 해를 구하는 것을 포함하는 벡터와 행렬의 연산을 수행한다. 이 두 클래스들은 각각의 클래스에 대해 다른 방법으로 명령이 수행되는 다형적 처리방식을 포함하고 있다. C++언어로 된 각 클래스는 클래스를 정의하는 header file과 header file에서 선언되는 클래스 member함수들로 구성된 program file로 구성된다. 클래스 선언은 전용(private)부분과 공용(public)부분으로 나누어 지며, 여기서 거의 모든 member함수들은 공용변수로 선언된다. 다음은 벡터 및 매트릭스 클래스 프로그램의 header file에 대한 설명이다.

```
class dmatrix;
class dvector
{
public:
  double *u;                                          \\변수지정 포인터변수의 선언
  int nrl,nrh,Row;                                    \\시작열 및 끝열 기억변수의 선언
public:
  dvector(int , int ,int );                           \\생성자
  ~dvector( );                                        \\소멸자
  double *vector_constructor( );                      \\포인터변수에 기억장소 할당
  void printing( );                                   \\텍스트상으로 벡터변수 출력
  friend dvector operator & (dmatrix& m,dvector& n);  \\매트릭스와 벡터형태의 연립방정식의 해를 구함
  friend dvector Lubksb (dmatrix& m,dvector& n);      \\상,하 매트릭스로 구분
  dvector& operator = (dvector& n);                   \\대입함수
  dvector(dvector&);                                  \\변수리턴시나 컴파일시의 벡터변수를 초기화
  dvector operator + ( dvector& n);                       \\vector+vector
  dvector operator + ( double & dscalar);                 \\vector+scalar
  friend dvector operator +(double & dscalar,dvector& n);     \\scalar+vector
  dvector operator - ( );                                 \\-vector
  dvector operator - ( double & dscalar);                 \\vector-scalar
  friend dvector operator - (dvectorn1, dvector& n2);     \\vector-vector
  friend dvector operator - (double & dscalar,dvector& n);    \\scalar-vector
  dvector operator / (double & dscalar);                  \\vector/scalar
  dvector operator * (double & dscalar);                  \\vector×scalar
  double operator *( dvector& n);                         \\scalar.vector
  dvector operator ¦ ( dvector& n);                       \\scalar×vector
  friend dvector operator *(dmatrix& m,dvector& n);       \\matrix×vector
  friend dvector operator *(dvector& n,dmatrix& m);       \\vector×matrix
  friend dvector operator *(double & dscalar,dvector& n);     \\scalar×vector

};
class dmatrix
{
public:
  double **v, *d, *e;                                 \\변수지정 포인터변수의 선언
  int nrl,nrh,ncl,nch,Row,Column, *indx;              \\시작열 및 끝열과 시작행 및 끝행의 기억변수선언
  int First,End, DOF;
  public:
dmatrix(int ,int ,int ,int ,int );                    \\생성자
~dmatrix( );                                          \\소멸자
void Free( );                                         \\메모리 해제함수
double ** matrix_constructor( );                      \\메모리 생성함수
```

```
void printing( );                                          \\텍스트상으로 매트릭스변수 출력
void eigenprinting( );                                     \\고유치를 텍스트로 출력
dmatrix operator ~( );                                     \\역행렬 계산
friend double Maxvalue (dmatrix& m);                       \\매트릭스변수중 최대치를 구함
friend double Minvalue (dmatrix& m);                       \\매트릭스변수중 최소치를 구함
friend double sum(dmatrix& m);                             \\매트릭스변수 모두의 합산치 계산
friend double Det(dmatrix& m);                             \\행렬식 계산
friend dvector operator &(dmatrix& m,dvector& n);          \\연립방정식의 해를 구함
friend dvector Lubksb (dmatrix& m,dvector& n);             \\LU의 결과치로 연립방정식의 해를 구함
friend dmatrix LU(dmatrix& m);                             \\상,하 삼각매트릭스로 분해
friend dmatrix Jacobi(dmatrix& m);                         \\jacobi방법으로 고유치 계산
friend dmatrix Tred2(dmatrix& m);                          \\대칭매트릭스를 tridiagonal매트릭스로 바꿈
friend dmatrix Tqli(dmatrix& m);                           \\tridiagonal매트릭스로 고유치 계산
void eigsrt( );                                            \\고유치의 정렬
friend dmatrix eigen(dmatrix& m);                          \\Householder & QL법에 의한 고유치 계산
dmatrix& operator = ( dmatrix& m);                         \\대입문
dmatrix(dmatrix&);                                         \\변수리턴시나 컴파일시 변수 초기화
dmatrix operator + ( dmatrix& m);                              \\matrix+matrix
dmatrix operator + ( double & dscalar );                       \\matrix+scalar
friend dmatrix operator +(double & dscalar,dmatrix& m);        \\scalar+matrix
dmatrix operator - ( );                                        \\-matrix
friend dmatrix operator - ( dmatrix& m1, dmatrix& m2);         \\matrix-matrix
dmatrix operator - ( double & dscalar );                       \\matrix-scalar
friend dmatrix operator - (double & dscalar,dmatrix& m);       \\scalar-matrix
dmatrix operator / (double & dscalar);                         \\matrix/scalar
dmatrix operator * (double & dscalar);                         \\matrix×scalar
dmatrix operator * (dmatrix& m);                               \\matrix×matrix
friend dvector operator * (dmatrix& m,dvector& n);             \\matrix×vector
friend dvector operator * (dvector& n,dmatrix& m);             \\vector×matrix
friend dmatrix operator * (double dscalar,dmatrix& m);         \\scalar×matrix
};

class smatrix:public dmatrix
{
public:
smatrix(int,int,int,int);                                  \\생성자
smatrix(int,int,int,int,int,int);                          \\생성자
smatrix(int,int,int,int,int,int,int,int);                  \\생성자
~smatrix( ){ }                                             \\소멸자
dmatrix& operator += (dmatrix m);                          \\요소매트릭스를 조합하여 전체매트릭스 형성
};
```

3.2 STRU클래스

STRU클래스는 기본적인 자료처리를 위한 NODE, MEMBER, LOAD 등의 클래스와 실제 각 구조형태별로 구조해석을 수행하는 ELEMENT클래스로 구성된 추상적 클래스이다. 그 중 구조물의 기하적, 재료적 성질들은 NODE 및 MEMBER클래스에 저장되며, LOAD 클래스에는 하중에 관한 자료들이 저장된다.

3.2.1 NODE 및 MEMBER클래스

NODE와 MEMBER클래스는 평면 뼈대 구조해석에서 기본적으로 필요한 자료를 기억시키고 관리하는 클래스들로서 1차원 요소를 기본으로 구성되었다. 다음의 NODE클래스와 MEMBER클래스의 header file에서와 같이 NODE클래스의 공용변수부분에 정의되어 있는 것은 실제 자료기억 변수들로서 절점번호나 윈도우(Window)상에서의 순서에 따라 해당 절점을 등록, 추가, 삭제한다. MEMBER클래스의 선언은 두 절점으로 구성된 1차원 요소의 양쪽 절점의 위치를 기억하는 포인터 변수와 실제의 부재요소의 특성치를 기억하기 위한 변수들로 구성된다.

```
class Node {
private:
  friend class NodeList;
  void DeleteCurrentNode( );
  void FatalListError(char * );
public
  NodeList * first_node;
  NodeList * current_node;
  int list_length;
  int Number;
  int Order;
  char changewindow;
  Coordinate * coordinate;
  Displacement * displacement;
  Constraint * constraint;
  Force * force;
  double Mass;
  double Two_nd_moment;
public:
  Node( );
  Node * node(int);
  Node * snode(int);
  Write( );
  int Size( ) {return list_length;}
  void AddNode(Node * item);
  void RemoveNode(Node * item);
  void changenode(int new_order);
  Node * CurrentNode( );
  void NextNode( ) →current_node
       = current_node→successor;}
  void operator++( ) {NextNode( );}
  void operator--( ) {current_node
  void = current_node{predecessor;}
  void GotoBeginning( );
  void GotoEnd( );
  ~Node( );
  };
```

```
class Member {
private:
  friend class MemberList;
  void DeleteCurrentMember( );
  void FatalListError(char * );
public:
  Node * First_Node;
  Node * End_Node;
  MemberList * first_member;
  MemberList * current_member;
  int list_length;
  int Order;
  int Number;
  char changewindow;
  double Length;
  double Area;
  double Youngs_modulus;
  double Density;
  double Inertia_moment;
public:
  Member( );
  Member * member(int);
  Member * smember(int);
  int Size( ) {return list_length;}
  Write( );
  void AddMember(Member * item);
  void RemoveMember(Member * item);
  void changemember(int new_order);
  Member * CurrentMember( );
  void NextMember( ) {current_member
       = current_member→successor;}
  void operator++( ) {NextMember( );}
  void operator--( ) {current_member
       = current_member→predecessor;}
  void GotoBeginning( );
  void GotoEnd( );
~Member( );
};
```

앞에서 언급한 클래스외에 각 NODE 및 MEMBER클래스의 객체를 연결하여 해석하고자 하는 구조형태를 구성해 주는 NodeList클래스와 MemberList클래스의 header file은 다음과 같다.

```
class NodeList {
    friend class Node;
    NodeList * predecessor;
    NodeList * successor;
    Node * entry;
class MemberList {
    friend class Member;
    MemberList * predecessor;
    MemberList * successor;
    Member * entry;
```

3.2.2 ELEMENT클래스

실제 각 구조형태별로 정적 및 자유진동해석을 수행하는 클래스인 ELEMENT클래스의 계층도는 그림 1과 같으며, 이 클래스의 하위 클래스들은 각 구조형태별 요소 강성도 및 질량 매트릭스를 구성한 뒤 전체 매트릭스로 조립(assemble)하는 처리방식을 포함한다. 각 구조형태별 상속 계통도(inheritance tree)에서는 아래쪽으로 갈수록 ELEMENT클래스

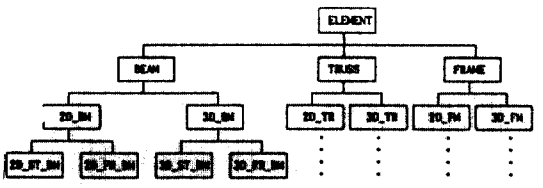


그림 1. Class hierarchy of ELEMENT class.

에서 가까운 클래스보다 더 상세한 구조해석 구성요소인 객체들로서, 2차원 정적해석, 2차원 자유진동해석, 3차원 정적해석 및 3차원 자유진동해석 등을 나타내며, 모든 클래스들은 ELEMENT클래스로부터 유도됨을 보여준다. 이는 유한요소해석의 경우에도 새로운 요소추가 및 기능추가시의 확장성 및 재사용성 등이 가능함을 나타내고 있다. ELEMENT클래스의 하위 클래스인 FRAME, BEAM 및 TRUSS 클래스 등은 실제의 평면뼈대 구조물을 해석하기 위한 클래스들로서 앞에서 설명한 NODE, MEMBER, LOAD 및 ELEMENT클래스들과 MATRIX 및 VECTOR클래스를 내부변수로 사용하며, 해석하고자 하는 구조형태에 따라 이미 생성된 기본 클래스들을 포함하는 구조해석용 클래스들을 필요에 따라 만들 수 있다.

다음은 구조물의 강성도 및 질량매트릭스를 MATRIX클래스 형태로 값을 구하여 실행시켜주는 ELEMENT클래스의 header file로서 NODE 및 MEMBER클래스의 번지를 저장할 수 있는 포인터 변수를 내부변수로 한다.

```
class Element {
  Node *fnode ,*enode;                          \\주Node와 대입Node
  Member *fmember,*emember;                     \\주Member와 대입Member
  e_structural_type structural;
  int dof;
public:
  Element(e_structural_type,Node*,Member*);     \\생성자
  dmatrix LKb(int);                             \\모멘트을 고려한 요소 강도매트릭스
  dmatrix LKr(int);                             \\축신장을 고려한 요소 강도매트릭스
  dmatrix LMb(int);                             \\모멘트을 고려한 요소 질량매트릭스
  dmatrix LMr(int);                             \\축신장을 고려한 요소 질량매트릭스
  dmatrix LK(int);                              \\축신장과 모멘트를 고려한 요소 강도매트릭스
  dmatrix LM(int);                              \\축신장과 모멘트를 고려한 요소 질량매트릭스
  dmatrix TKb( );                               \\모멘트를 고려한 전체 강도매트릭스
  dmatrix TKr( );                               \\축신장을 고려한 전체 강도매트릭스
  dmatrix TMb( );                               \\모멘트을 고려한 전체 질량매트릭스
  dmatrix TMr( );                               \\축신장을 고려한 전체 질량매트릭스
  dmatrix TK( );                                \\축신장과 모멘트를 고려한 전체 강도매트릭스
  dmatrix TM( );                                \\축신장과 모멘트를 고려한 전체 질량매트릭스
  dmatrix TransCopy(dmatrix&);                  \\대칭복사
  ~Element( ){ }                                \\소멸자
};
```

3.3 GUI(사용자 접속장치) 클래스

프로그램과 사용자를 연결시켜 주는 GUI클래스는 그림 2에서 보는 바와 같이 자료의 입력과 결과의 출력을 담당하며, 다음과 같은 입력 윈도우 클래스(Input Window Class)와 출력 그래픽 클래스(Output Graphic Class)를 포함한다.

이 클래스는 Scroll, Cementbar, Statebar, Newbar 및 Cell 클래스들로 구성되어 있으며, 여러가지 구조해석을 위한 자료들을 사용자가 쉽게 입력할 수 있게 해 준다. 이 윈도우 클래스는 자료의 도표형(tabular form) 개별입력과 편집 및 automatic generation을 이용한 입력 등이 키보드(keyboard)와 마우스(mouse)를 통해 가능하다. 그림 2는 입력 윈도우의 예로서 사용자와 컴퓨터간에 구조형태, 구속조건, 하중상태 등의 자료정보를 자유롭게 교환하도록 하기 위한 클래스들과 계산과정을 이해할 수 있게 하는 대화식 입력형태 및 이미 입력된 자료들을 참고할 수 있는 또다른 윈도우들을 제공하는 클래스들로 구성되며, 그래픽 클래스와 연계되어 윈도우상에 여러가지 그래픽 입출력결과를 보여주게 된다.

입력 윈도우 클래스의 하위 클래스인 Cell클래스는 입력 윈도우를 구성하는 가장 기본적인 입력

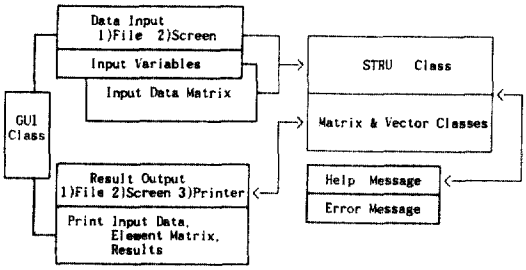


그림 2. Interactions of GUI class and other classes.

그림 3. Example of input windows for truss analysis.

단위로서 좌우로 scroll이 가능하며 마우스로도 조작 가능하다. 다음은 Cell 클래스의 인터페이스 부분이다.

```
class cell: public Icon {
friend class CellGroup;                                  \\입력 윈도우와의 상호작용을 위한 friend선언
CellGroup* parent;
int number_x,number_y;
int key,x,y,FLAP,initial_value,MAXCHAR;
int MARGIN,N,R,ScreenHeight,Number,End_point;
char* string,**endptr;
int CHARNUMBER,IMG_POINT;
int status;
double value;
void InsertChar( );                                      \\현재의 커서위치에 문자를 삽입
void DeleteChar( );                                      \\현재의 커서위치의 문자를 삭제
void CellFill_string( );                                 \\현재의 cell크기만큼 문자를 채움
public:
cell(CellGroup*,char*,int,int,int,int,int,int,int);      \\생성자
void Display( );                                         \\ 화면에 자신을 그림
void LeftButtonDown(int,int);                            \\ 마우스가 눌러져 있을때 실행
void HighLight( );                                       \\cell을 밝힘
void UnHighLight( );                                     \\cell을 어둡게 함
void initial_display( );                                 \\처음 선택되었을 때 초기화면을 그림
void initial_after_display( );                           \\초기화면후의 화면을 그림
void cursor_set( );                                      \\cell안의 커서를 다시 그림
void LeftButtonPressed( );                               \\마우스 버튼을 누르면 실행
double Value( );
void SetLegend(char* S_tring);                           \\cell안의 문자를 바꿈
char* ReturnLegend( );                                   \\cell안의 문자를 리턴
void KeyTyped(int);                                      \\키보드가 눌러졌을 때 실행
virtual ~cell( );                                        \\소멸자
};
```

4. 예제해석 및 고찰

본 연구에서 개발된 객체지향 C++ 자유진동해석 프로그램의 타당성을 검증하기 위하여 다음과 같은 예제들을 486-DX2 컴퓨터로 해석하였다.

4.1 캔틸레버 보의 자유진동해석

Data Input

```
1   0.0  0.0  1  1
2   2.0  0.0  0  0
3   4.0  0.0  0  0
4   8.0  0.0  0  0
5  12.0  0.0  0  0
1  1  2  3.25e-4   0.0625   30e6  7.35e-4
2  2  3  3.25e-4   0.0625   30e6  7.35e-4
3  3  4  3.25e-4   0.0625   30e6  7.35e-4
4  4  5  3.25e-4   0.0625   30e6  7.35e-4
```

Result Output

frequency 1=5.573e+01 (f_{ref}=56.6 Hz[23])

eigenvectors:

1.825e−15	−1.194e−14	4.412e−02	−4.221e−02
1.626e−01	−7.500e−02	5.361e−01	−1.084e−01
7.301e−01	3.633e−01		

frequency 2=3.550e+02 (f_{ref}=355.8 Hz[23])

eigenvectors:

5.191e−15	−2.122e−14	7.452e−02	−6.864e−02
2.628e−01	−1.187e−01	7.392e−01	−1.284e−01
−5.669e−01	−1.491e−01		

4.2 평면 뼈대구조의 자유진동해석

Data Input

```
1  0.0   0.0 1 1 1
2  0.0  15.0 0 0 0
3 12.0  15.0 0 0 0
4 12.0   0.0 1 1 1
1 1 2 3.36e−4 6.35e−2 30e6 7.35e−4
2 2 3 3.36e−4 6.35e−2 30e6 7.35e−4
3 3 4 3.36e−4 6.35e−2 30e6 7.35e−4
```

Result Output

frequency 1=3.631e+01 (f_{ref}=36.4 Hz[23])

eigenvectors:

4.190e−15	−4.192e−16	2.841e−16	−3.613e−04
6.220e−01	3.961e−01	5.333e−01	1.533e−01
−3.615e−01	1.328e−15	5.739e−14	−1.011e−14

frequency 2=1.975e+02 (f_{ref}=197.1 Hz[23])

eigenvectors:

−4.425e−15	5.202e−16	−3.017e−16	3.754e−04
−4.088e−01	−5.378e−01	6.871e−01	6.648e−02
−1.838e−01	1.826e−15	1.106e−13	9.220e−15

4.3 입체 트러스의 자유진동해석

Data Input

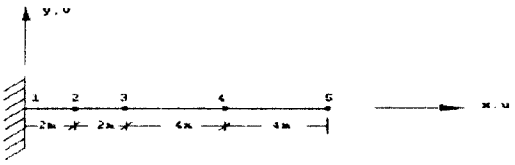
```
1  0.0    0.0    0.0   1 1 1
2 12.00   0.0    0.0   1 1 1
3  6.00  10.39   0.0   1 1 1
4  6.00   5.195 15.25  0 0 0
1 1 4  0.012  30e6 7.35e−4
2 2 4  0.012  30e6 7.35e−4
3 3 4  0.012  30e6 7.35e−4
```

Result Output

frequency 1=9.330e+02 (f_{ref}=933 Hz[23])

eigenvectors:

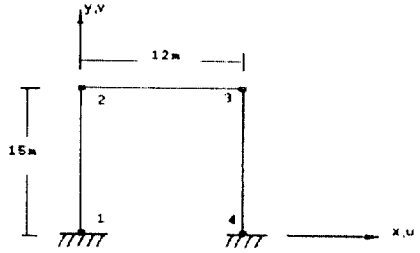
6.593e−13	1.154e−12	3.934e−13	6.593e−13
−1.154e−12	−3.934e−13	−8.138e−10	3.640e−25
−3.039e−26	1.000e+00	4.673e−01	−3.545e−01



$A=0.0625m^2$ $E=30\times10^6N/m^2$

$\rho=7.35\times10^{-4}N\cdot s^2/m^4$ $I=3.25\times10^{-4}m^4$

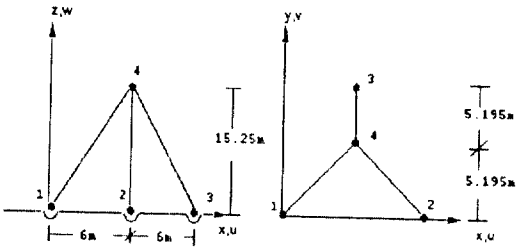
그림 4. Cantilever beam.



$A=0.0635m^2$ $\rho=7.35\times10^{-4}N\cdot s^2/m^4$

$E=30\times10^6N/m^2$ $I=3.36\times10^{-4}m^4$

그림 5. Portal frame.



$A=0.012m^2$ $E=30\times10^6N/m^2$ $\rho=7.35\times10^{-4}N\cdot s^2/m^4$

그림 6. Space truss.

frequency 2=9.894e+02 (f_{ref}=989 Hz[23])

eigenvectors:

6.258e−13	1.231e−12	2.934e−13	−6.258e−13
1.231e−12	−4.901e−13	−4.289e−22	1.660e−12
−9.710e−14	−4.686e−01	9.957e−01	−9.286e−02

4.4 고 찰

위와 같은 여러가지 형태의 뼈대 구조물에 대한 예제해석 결과 본 프로그램에 의한 해가 참고문헌[23]의 해와 매우 잘 일치함은 본 연구의 타당성을 잘

나타내고 있다고 할 수 있으며, 자유진동해석에 국한된 본 프로그램의 C++ 원시코드(source code)는 객체지향적 특성 때문에 같은 알고리즘(algorithm)을 가진 FORTRAN 프로그램[23]보다 프로그램의 크기가 28% 정도 축소되었다. 따라서 다양한 기능을 갖는 방대한 범용 프로그램의 경우에는 프로그램의 크기가 훨씬 더 축소될 수 있을 것으로 보이며, 이는 프로그램의 디버깅(debugging), 컴파일(compile) 및 링크(link)시간 등을 줄여주게 되어 프로그램의 부분적인 수정 및 확장 등을 효과적으로 수행할 수 있음을 의미한다.

5. 결 론

객체지향 언어인 C++로 작성되어 구조물의 자유진동해석을 효율적으로 수행할 수 있게 하는 본 구조해석 프로그램은 구조적 프로그래밍 기법의 문제점들을 해결한 새로운 객체지향 프로그래밍 기법을 사용하여 개발되었으며, 이러한 객체지향 프로그래밍 기법의 특성인 상속성, 다형성 및 동적 메모리 할당(dynamic memory allocation) 등을 이용하여 동일한 목적의 절차적 언어로 구성된 프로그램에 비해 소프트웨어 개발시간과 프로그램의 크기를 줄일 수 있으며, 프로그램의 수정과 소프트웨어 유지관리면에서도 보다 나은 환경을 제공해 준다는 것을 알 수 있었다. 또한 본 연구는 전처리 및 후처리 프로그램의 개발로 인한 편리한 사용환경의 제공, 객체지향 프로그래밍에서의 프로그램의 재사용성 및 확장성 등으로 유한요소 구조해석을 위한 PC용 프로그램의 개발 연구에 확장시킬 수 있다고 사료된다.

참고문헌

1. Wilson, E.L., *SAP: Structural Analysis Program,* Berkeley, CA, 1970.
2. Wilson, E.L. and Habibullah, A., *SAP90: Computer Programs for the Static and Dynamic Finite Element Analysis of Structures,* Computers & Structures Inc., Berkeley, CA, 1989.
3. Chung, C.F., "Development of an Extended Data-based Analysis Interpretive Treatise for Micro/Mainframe Computers. (Micro-AIT)", *M.S.C. Thesis,* AIT, Bangkok, Thailand, 1987.
4. Kanok-Nukulchai, W., Somporn, A. and Sarun, U., *MicroFEAP II P1-Module: A Module for Static Analysis of 2D Truss, Frame and Shear Wall Structural Systems,* The MICROACE CLUB, AIT, 1987.
5. Kanok-Nukulchai, W., Somporn, A. and Sarun, U., *MicroFEAP II P4-Module: A Module for Static Analysis of Space Truss, Frame and Structural Systems,* The MICRO-ACE CLUB, AIT, 1987.
6. Stroustrup, B., *The C++ Programming Language,* Addison-Wesley, MA, 1986.
7. Abdalla, J.A. and Yoon, C.J., "Object-Oriented Finite Element and Graphic Datatranslation Facility", *ASCE., J. Comput. Civil. Engng.,* Vol. 6, No. 3, 1992.
8. Forde, B.W.R., Foschi, R.O. and Stiemer, S.F., "Object Oriented Finite Element Analysis", *Computers & Structures,* Vol. 34, No. 3, 1990, pp. 355-374.
9. Wiener, R.S. and Pinson, L.J., *An Introduction to Object-Oriented Programming and C++*, Addison-Wesley, MA, 1989.
10. Cohn, L.F., *Computing in Civil and Building Engineering: Proceedings of the Fifth International Conference,* American Society of Civil Engineers, NY, 1993.
11. Fenves, G.L., "Object-Oriented Programming for Engineering Software Development", *Engineering with Computers,* Vol. 6, 1990, pp. 1-15.
12. Zimmermann, T., Dubois-Pè lerin, Y. and Bomme, P. "Object-Oriented Finite Element Programming: I. Governing Principles", *Comput. Meth. Appl. Mech. Engng.,* Vol. 98, 1992, pp. 291-303.
13. Zimmermann, T., Dubois-Pè lerin, Y. and Bomme, P., "Object-Oriented Finite Element Programming: II. A Protype Program in Smalltalk", *Comput. Meth. Appl. Mech. Engng.,* Vol. 98, 1992, pp. 361-397.
14. Goldberg, A. and Robson, D., *Smalltalk-80, The Implementation,* Addison-Wesley, MA, 1983.
15. Mackie, R.I., "Object-Oriented Programming of the Finite Element Method", *Int. J. Numer. Meth. Engng.,* Vol. 35, 1992, pp. 425-436.
16. Scholz, S.P., "Elements of an Object-Oriented FEM++ Program in C++", *Computers & Structures,* Vol. 43, No. 3, 1992, pp. 517-529.
17. Hong, S.M. and Kim, C.K., "Integrated Information Management Expert system for Structural Engineering of Buildings", *Microcomputers in Ci-*

vil Engng., Vol. 8, No. 1, 1993, pp. 9-26.
18. 김치경, 홍성목, "건축구조설계 통합시스템을 위한 건축구조물의 모델링", 한국전산구조공학회 학술발표회 논문집, 제6권 제1집, 1993, pp. 71-78.
19. 천진호, 김홍국, 이병해, "일관 구조설계 시스템 구축에 있어서 객체지향 데이터 베이스의 도입", 한국전산구조공학회 학술발표회 논문집, 제6권 제1집, 1993, pp. 79-86.
20. 신영식, 고영배, "PC용 객체지향 매트릭스 구조해석 프로그램의 개발", 영남대학교 공업기술연구소 연구보고, 18권 2호, 1990, pp. 11-19.
21. 신영식, 서진국, 박영식, 최회욱, "PC용 객체지향 구조해석 프로그램의 개발", 한국전산구조공학회지, 제5권 제4호, 1992, pp. 125-132.
22. Shin, Y.S., "Object oriented free vibration analysis of structures", *Proceedings of EASEC-4*, Fourth East-Asia Pacific Conference on Structural Engineering and Construction, Seoul, Vol. 3, 1993, pp. 1809-1814.
23. Ross, C.T.F., *Finite Element Programs for Structural Vibrations*, Springer-Verlag, 1991.

(接受 : 1993. 11. 1)