한국전산구조공학회
1993년도 봄학술발표회 논문집

평면 뼈대 구조물의 객체지향 자유진동해석
Object-Oriented Free Vibration Analysis of Plane Framed Structures

신 영식* 최 희욱** 서 진국***
Shin, Young-Shik Choi, Hee-Wook Suh, Jin-Kook

ABSTRACT

A computer program for free vibration analysis of plane framed structures has been developed by object oriented programming technique using C++ language. The object oriented programming concepts such as object, class, method and inheritance are represented. The static and free vibration analyses for framed structures were satisfactorily performed by this program which consists of TOP, VECTOR, MATRIX, STRU, GUI and other classes. Numerical test shows the validity and capability of the present study which can be expandable to develop a general purpose object oriented finite element analysis program of structures.

1. 서 론

유한요소법을 이용한 기존의 대형컴퓨터용 범용 구조해석 프로그램들은 자료의 입출력이 불편하고 프로그램의 수정이 용이하지 못하였으나, 개인용 소형컴퓨터(Personal computer)의 폭넓은 보급과 기능의 급속한 향상은 이러한 범용 프로그램의 단점을 개선한 PC용 구조해석 프로그램[1,2,3]의 개발을 촉진시키고 있다. 일반적인 PC용 프로그램은 다양한 메뉴(Menu) 선택과 해석기능을 가지고 있어야 하나 FORTRAN 등으로 작성된 기존의 절차적(Procedural)프로그램들은 단지 정확하고 신속한 처리에만 치중하고 있을 뿐, 다양한 메뉴의 선택이나 부분적인 프로그램의 추가 및 수정 등이 용이하지 않다. 최근 이러한 단점을 보완하기 위해 프로그래밍이 용이하고 프로그램의 신뢰성 및 재사용성이 뛰어난 객체지향 프로그래밍(Object-oriented programming)기법[4]이 제안되었다.

많은 객체지향언어(Objected-oriented language)중 본 연구에서 사용하고 있는 C++언어[5]는 SIMULA, Smalltalk[6] 등과는 달리 C언어의 모든 기능과 객체지향적 개념을 함께 포함하고 있는 복합(Hybrid) 언어이기 때문에 C언어로 작성된 기존의 프로그램을 약간의 수정으로 이용할 수 있으며 Dynamic binding개념의 도입으로 유사한 기능을 갖고 있는 C프로그램보다 프로그램의 크기는 작아지고 처리속도는 비슷하게 된다. 이러한 객체지향 프로그램에 대한 연구는 1986년경 전산공학 분야에서 제안되어 1990년 Forde등[7]이 객체지향 유한요소해석에 관한 연구를 발표한 이래 구조해석에 관한 많은 연구가 활발히 진행되고 있다. Fenves[8], Zimmermann등[9,10]의 연구에서 Smalltalk과 같은 객체지향적 언어가 유한요소해석을 다룰 수 있음을 보여 주었고, C++언어를 사용한 Mackie[11], Scholz[12] 등의 연구가 있으나 아직은 유한요소해석 프로그램에 대한 초보적인 연구에 불과하다고 할 수 있다. 따라서 본 연구에서는 이러한 객체지향 프로그래밍 기법으로 프로그램을 작성하여 뼈대 구조의 자유진동해석을 수행하고자 한다.

2. 객체지향 프로그래밍의 개념

객체지향 프로그래밍은 자료구조와 제어구조로 된 기존의 프로그래밍 기법과는 달리 객체(Object), 클래스(Class), 상속성(Inheritance) 및 다형성(Polymorphism) 등의 여러가지 개

* 영남대학교 공과대학 토목공학과 부교수
** (주)서한 토목부 사원
***영남대학교 공업기술연구소 연구원

념들로 설명된다. 객체는 수정없이 재사용할 수 있는 독립된 프로그램으로 자료정보(Data)와 처리방식(Method)을 캡슐화(Encapsulation)하여 가지고 있다. 이러한 객체들은 자료들을 객체만의 고유한 처리방식으로 처리하며 어떠한 외부함수들도 객체의 자료를 변화시킬 수 없다. 객체들은 메시지(Message)를 주고 받음으로써 서로 교류할 수 있는데 어떤 객체가 메시지를 전달받으면 객체는 그 메시지를 분석하고 메시지에 포함된 처리방식에 의하여 자료들을 처리하게 된다. 동일한 종류의 자료정보와 처리방식을 가지고 있는 객체들의 집합체를 클래스라고 하며, 따라서 객체는 클래스의 인스턴스(Instance)이다. 각 클래스는 객체의 이러한 정보들을 계층관계(Hierarchy)를 통하여 상위클래스(Superclass)는 하위클래스(Subclass)로 단일 및 다중상속하며, 이러한 상속성은 Code의 효율적이고 본질적인 재사용성을 증대시킨다. 또한 객체들은 동일한 메시지에 대해 각각 다른 반응(처리방식)을 취할 수 있으며 이를 다형성이라 한다. 이 것은 시스템내에 이미 존재하는 메시지에 대해 응답할 수 있는 새로운 객체를 시스템에 삽입하는 것이 용이하다는 것을 의미한다. 이와 같이 객체들 사이에 메시지 전달특성(Call-by-desire)을 포함하는 객체 캡슐화 등이 절차적 프로그래밍 기법과의 차이점이라 할 수 있다.

3. 객체지향 자유진동해석 프로그램의 구성

본 연구에서 개발된 객체지향 자유진동해석 프로그램은 그림1과 같이 프로그램을 실행시켜주는 TOP클래스, 행렬과 벡터의 연산을 수행하는 MATRIX 및 VECTOR클래스, 뼈대구조시스템의 정적해석과 자유진동해석을 수행하는 STRU클래스 및 사용자와 프로그램을 연결시켜주는 GUI(Graphic User Interface)클래스들로 구성되어 있다.

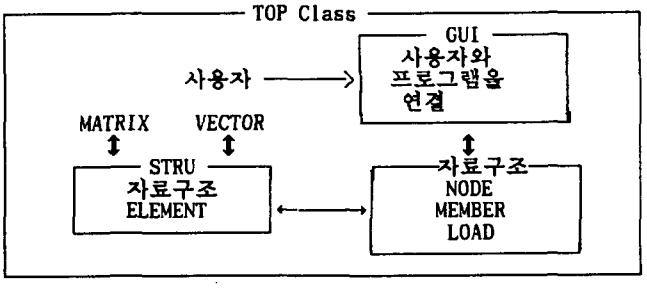


그림1. Structure of object-oriented free vibration analysis program

3.1 TOP 클래스

TOP클래스는 프로그램의 변화를 감지하는 EVENT클래스와 화면상의 명령Icon들을 관리하는 SCREEN클래스 및 기본적인 그래픽 함수들을 포함하고 있는 GRAPHIC클래스를 생성시키고 프로그램상의 입력상황을 계속 감시하는 무한 Loop 를 생성시키는 추상적(Abstract)클래스이다.

3.2 MATRIX 및 VECTOR 클래스

이 클래스들은 벡터 및 행렬의 연산과 고유값 문제의 해석을 수행한다. C++언어에서의 클래스 프로그램은 Header file과 Program file을 포함하는데 Header file에서 클래스를 정의한다. 클래스의 선언은 전용변수(Private)부분,보호(Protected)부분 및 공용변수(Public)부분으로 나눌 수 있다. 이것은 전용변수 부분에서 선언되는 변수와 member함수들은 다른 객체나 함수로 부터 사용될 수 없는 반면, 공용변수 부분은 모든 객체나 함수에 사용될 수 있음을 뜻한다. 다음은 MATRIX와 VECTOR클래스의 Header file이며, 거의 모든 Member함수는 외부에서 불러 쓰기 편리하게 공용변수로 선언된다. 여기서 하나의 클래스내에 동일한 함수명으로 서로 다른 인수를 가지는 함수를 중복하여 선언하는 C++언어의 고유한 초과연산자의 특징과 두 클래스를 통하여 서로 다른 클래스내에 각각 초과연산자들이 선언되어 있는 다형성이 잘 나타나 있다.

```
class dmatrix;
class dvector
{
public:
 double  *u;                                  \\변수기억 Pointer변수의 선언
 int nrl,nrh,Row;                             \\시작열 및 끝열기억변수의 선언
```

```
public:
 dvector(int ,int ,int );                          \\생성자
 ~dvector();                                       \\소멸자
 double  *vector_constructor();                    \\Pointer변수에 기억장소 할당
 void printing();                                  \\text상으로 벡터변수 출력
 friend dvector operator & (dmatrix& m,dvector& n); \\매트릭스와 벡터형태의 연립방정식의 해를 구한다
 friend dvector Lubksb (dmatrix& m,dvector& n);    \\상,하 매트릭스로 구분
 dvector& operator = (dvector& n);                 \\대입함수
 dvector(dvector&);                                \\변수return시나 compile시의 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 - (dvectornl, 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 Pointer변수의 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 );                      \\생성자
~dmatrix();                                        \\소멸자
void Free();                                       \\memory해제함수
double ** matrix_constructor();                    \\memory생성함수
void printing();                                   \\text상으로 매트릭스변수 출력
void eigenprinting();                              \\eigen값을 text로 출력
dmatrix operator ~();                              \\역행렬 계산
friend double  Maxvalue (dmatrix& m);              \\매트릭스변수중 최대치를 구한다
friend double  Minvalue (dmatrix& m);              \\매트릭스변수중 최소치를 구한다
friend double  sum(dmatrix& m);                    \\매트릭스변수모두의 합산치 계산
friend double  Det(dmatrix& m);                    \\매트릭스의 determinant 계산
friend dvector operator &(dmatrix& m,dvector& n);  \\연립방정식의 해를 구한다
friend dvector Lubksb (dmatrix& m,dvector& n);     \\LU의 결과치로 연립방정식의 해를 구한다
friend dmatrix LU(dmatrix& m);                     \\상,하 삼각행렬로 분해
friend dmatrix Jacobi(dmatrix& m);                 \\jacobi방법으로 고유치계산
friend dmatrix Tred2(dmatrix& m);                  \\대칭행렬을 tridiagonal행렬로 바꾼다
friend dmatrix Tqli(dmatrix& m);                   \\tridiagonal행렬로써 고유치계산
void eigsrt();                                     \\고유치의 sorting
friend dmatrix eigen(dmatrix& m);                  \\Householder & QL법에 의한 고유치계산
dmatrix& operator = ( dmatrix& m);                 \\대입문
dmatrix(dmatrix&);                                 \\변수return시나 compile시 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);             \\요소매트릭스 조합 전체매트릭스 형성
 };
```

3.3 STRU 클래스

STRU클래스는 다음과 같은 클래스들을 내부변수로서 포함하는 추상적 클래스로 구조물에 대한

강성도 및 질량 매트릭스를 조합하여 뼈대 구조물에 대한 정적 및 자유진동해석을 수행한다.

(1) NODE, MEMBER 및 LOAD 클래스

구조해석에 필요한 자료를 저장, 관리하는 클래스들로서 구조물의 기하학적, 재료학적 자료 및 경계조건은 NODE클래스 및 MEMBER클래스에, 하중에 대한 자료는 LOAD클래스에 각각 입력된다. 다음은 NODE클래스와 MEMBER클래스의 Header file로서 NODE클래스의 공용변수부분에 정의되어 있는 것은 실제자료기억변수들로서 절점번호나 Window상에서의 순서에 따라 해당절점을 등록, 추가 또는 삭제하게 된다. MEMBER클래스의 선언은 두 절점으로 구성된 1차원 요소의 양쪽 절점의 위치를 기억하는 Pointer변수와 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
       = 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();
 };
```

(2) ELEMENT 클래스

다음은 구조물의 강성도 및 질량매트릭스를 MATRIX클래스 형태로 값을 구하여 돌려주는 ELEMENT클래스의 Header file로서 NODE 및 MEMBER클래스의 번지를 저장할 수 있는 Pointer변수를 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
};
```

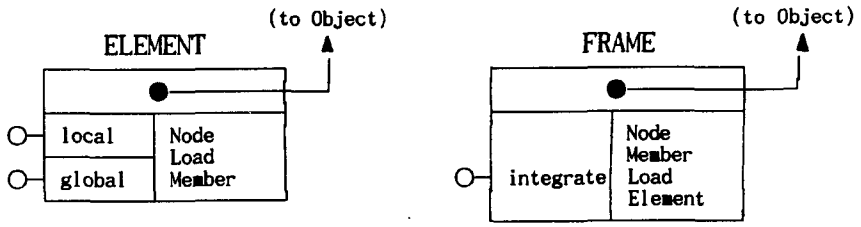


그림2. Description of ELEMENT & FRAME classes

(3) FRAME, BEAM 및 TRUSS 클래스

FRAME클래스 등은 실제의 평면뼈대 구조물을 해석하기 위한 클래스로서 앞에서 설명한 NODE, MEMBER, LOAD 및 ELEMENT클래스들과 MATRIX 및 VECTOR클래스를 내부변수로 사용하며, 해석하고자 하는 구조형태에 따라 FRAME클래스와 같이 이미 생성된 기본 클래스들을 포함하는 구조해석용 클래스들, 즉 BEAM 및 TRUSS클래스 등을 필요에 따라 만들 수 있다.

3.4 입력Window 클래스

사용자 환경에서 사용자가 원하는 해석구조를 선택하면 그 해석에 필요한 자료입력을 위한 입력Window가 모니터(Monitor)에 나타난다. 이 입력Window는 Scroll, CementBar, StateBar, NewBar, Cell 등의 여러 클래스들을 내부변수로 구성하여 절점좌표, 구속조건, 질량, 질량2차모멘트의 부가여부, 부재를 구성하는 절점, 기타 특성치를 사용자가 손쉽게 입력하도록 해준다. Cell클래스는 입력Window를 구성하는 가장 기본적인 입력 단위로서 좌우로 Scroll이 가능하며 마우스(Mouse)로도 조작이 가능하다. 이 입력Window의 입력은 개별 Node 입력과 Node generation에 의한 입력을 함께 제공하며 Node generation 에 의해 입력된 수치도 다시 사용자가 확인 할 수 있게 개별 Node로 계산하여 입력Window에 다시 써준다. 또 이전에 입력한 수치를 다시 보기위해 Scroll기능도 제공된다. 그림3은 평면 뼈대구조물에 대한 자료입력 화면이다.

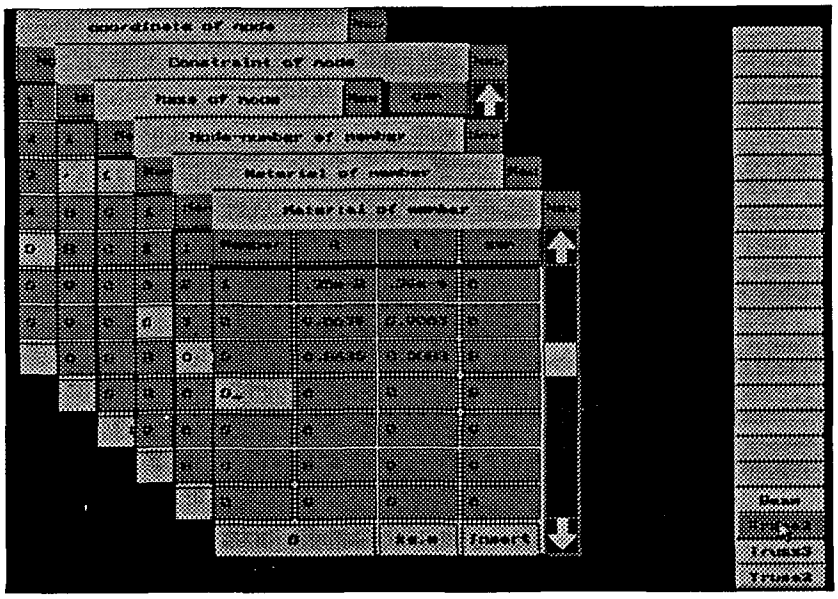
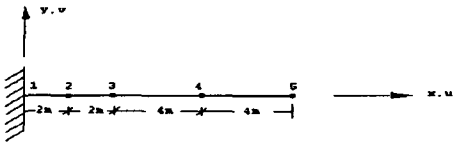


그림3. 평면 뼈대 구조의 자료입력을 위한 6개의 Window

4. 예제해석

본 연구에서 개발된 프로그램의 기능과 효율성을 검토하기 위해 다음과 같은 예제를 486-DX2 PC로 해석하였다.

4.1 캔틸레버 보



$A=0.0625m^2$
$E=30\times10^6N/m^2$
$\rho=7.35\times10^{-4}N^2/m^4$
$I=3.25\times10^{-4}m^4$

자료입력

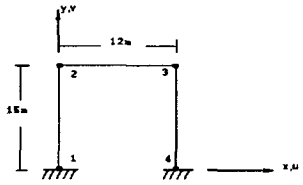
```
1   0.0    0.0    1 1
2   2.0    0.0    0 0
3   4.0    0.0    0 0
4   8.0    0.0    0 0
5  12.0    0.0    0 0
1 1 2 3.25e-4 0.0625 30e6 7.35e-4
2 2 3 3.25e-4 0.0625 30e6 7.35e-4
3 3 4 3.25e-4 0.0625 30e6 7.35e-4
4 4 5 3.25e-4 0.0625 30e6 7.35e-4
```

결과출력

```
frequency     1  =  5.573e+01  (f_ref=56.6Hz [13])
     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=355.8Hz [13])
     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 Frame



$A=0.0635m^2$
$E=30\times10^6N/m^2$
$\rho=7.35\times10^{-4}N^2/m^4$
$I=3.36\times10^{-4}m^4$

자료입력

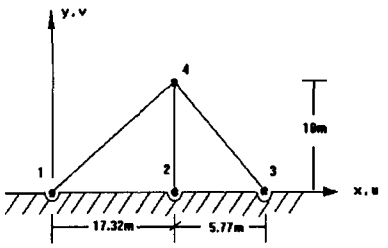
```
1   0.0   0.0   1 1 1
2   0.0  15.0   0 0 0
3  12.0  15.0   0 0 0
4  12.0   0.0   1 1 1
1 1 2 3.36e-4 6.35e-2 30e6 7.35e-4
2 2 3 3.36e-4 6.35e-2 30e6 7.35e-4
3 3 4 3.36e-4 6.35e-2 30e6 7.35e-4
```

결과출력

```
frequency     1  =  3.631e+01  (f_ref=36.4Hz [13])
     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-015
       5.739e-14  -1.011e-014

frequency     2  =  1.975e+02  (f_ref=197.1Hz [13])
     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-015
       1.106e-13   9.220e-015
```

4.3 평면 트러스



$A=0.06m^2$
$E=30\times10^6N/m^2$
$\rho=7.35\times10^{-4}N^2/m^4$

자료입력

```
1  0.0   0.0 1 1
2 17.32  0.0 1 1
3 23.09  0.0 1 1
4 17.32 10.0 0 0
1 1 4 0.06 30e6 7.35e-4
2 2 4 0.06 30e6 7.35e-4
3 3 4 0.06 30e6 7.35e-4
```

결과출력

```
frequency   1  =  2.120e+03  (f_ref=2064Hz [13])
      eigenvectors :
         6.507e-13   1.718e-12  -1.301e-08   6.169e-14   4.968e-13
        -5.178e-13   9.978e-01   6.651e-02

frequency   2  =  3.651e+03  (f_ref=3662Hz [13])
      eigenvectors :
         4.334e-13   4.312e-12  -2.572e-09   1.212e-12  -1.872e-12
         1.417e-12  -6.651e-02   9.978e-01
```

5. 결 론

본 연구에서는 C++언어를 이용한 객체지향 프로그래밍기법으로 PC용 구조해석 프로그램을 개발하여 뼈대구조물의 자유진동해석을 수행하였다. 이 프로그램을 구성하고 있는 클래스들은 하나의 독립적인 프로그램으로도 활용가능할뿐 아니라 "Dynamic memory allocation"특성을 가지므로 작은 기억용량을 사용하더라도 프로그램이 정적으로 고정되어 있는 절차적 프로그램보다 처리속도가 빠르며, 상속성 및 다형성과 같은 특징으로 인하여 프로그램 크기도 동일한 내용의 절차적프로그램보다 축소된다. 본 프로그램의 효율성을 검토하기 위하여 예제해석을 수행하였는 바, 만족할만한 결과를 고찰할 수 있었다. 따라서 앞으로 이러한 객체지향 프로그밍기법에 의한 다양한 구조해석 프로그램의 개발과 연구가 활발하리라고 판단된다.

6. 참고문헌

1. C.F.Chung, "*Development of an extended databased analysis interpretive treatise for micro / mainframe computers. (Micro-AIT)*", M.S.Thesis, AIT, Bangkok, Thailand (1987).
2. W.Kanok-Nukulchai, A.Somporn and U.Sarun, MicroFEAP II P1-Module *A module for static analysis of 2D truss, frame and shear wall structural systems*, The MICRO-ACE CLUB, AIT (1987).
3. W.Kanok-Nukulchai, A.Somporn and U.Sarun, MicroFEAP II P4-Module *A module for static analysis of space truss, frame and structural systems*, The MICRO-ACE CLUB, AIT (1987).
4. R.S.Wiener and L.J.Pinson, *An introduction to object - oriented programming and C++*, Addison-Wesley, MA (1989)
5. B.Stroustrup, *The C++ Programming Language*, Addison-Wesley, MA(1986).
6. A.Goldberg and D.Robson, *Smalltalk-80, The Implementation*, Addison-Wesley, MA (1983)
7. B.W.R.Forde, R.O.Foschi and S.F.Stiemer, "Object oriented finite element analysis", *Computers & Structures*, Vol.34, No.3, pp.355-374 (1990)
8. G.L.Fenves, "Object -oriented programming for engineering software development", *Engineering with Computers*, Vol.6, pp.1-15 (1990)
9. T.Zimmermann, Y.Dubois-Pèlerin and P.Bomme, "Object - oriented finite element programming: I .Governing Principles", *Comput. Meth. Appl. Mech. Engng.*, Vol.98, pp.291-303 (1992)

10. T. Zimmermann, Y. Dubois-Pèlerin and P. Bomme, "Object - oriented finite element programming: II. A protype program in smalltalk", *Comput. Meth. Appl. Mech. Engng.*, Vol. 98, pp. 361-397 (1992)
11. R. I. Mackie, "Object oriented programming of the finite element method", *Int. J. Numer. Meth. Engng.*, Vol. 35, pp. 425-436 (1992)
12. S. P. Scholz, "Elements of an object-oriented FEM++ program in C++", *Computers & Structures*, Vol. 43, No. 3, pp. 517-529 (1992)
13. C. T. F. Ross , *Finite element programs for structural vibrations*, Springer-Verlag (1991)
14. E, W, Faison, *Graphical User Interfaces With Turbo C++*, SAMS (1990).