PC용 객체지향 구조해석 프로그램의 개발

Development of Object-Oriented Structural Analysis Program for PC

신 영 식* 서 진 국**
Shin, Young-Shik Suh, Jin-Kook
박 영 식*** 최 희 욱****
Park, Young-Shik Choi, Hee-Wook

요 약

본 연구에서는 C++언어를 이용한 객체지향 프로그래밍 기법으로 매트릭스 연산과 평면뼈대 구조물의 해석이 가능한 PC용 구조해석 프로그램을 개발하였다. 객체지향 프로그래밍에서의 주요 개념인 객체, 클래스, 처리방식, 상속성 및 다형성을 도식화하여 설명하였으며, 매트릭스 연산과 평면뼈대 구조해석에 대한 예제 해석 결과는 이 프로그램의 효율성과 타당성을 보여 주었다. 따라서 본 연구는 객체지향 프로그래밍 기법의 특징인 프로그램의 확장성과 재사용성 및 다양한 GUI의 구현가능성을 이용하여 앞으로의 객체지향 유한요소 프로그램 개발에 활용될 것이다.

Abstract

A computer program for matrix structural analysis by object-oriented programming technique using C++ language has been developed. Object, class, method and inheritance which are used in object-oriented programming are illustrated using a graphical representation. The matrix operations and the structural analysis by matrix displacement method were satisfactorily performed by the proposed program. This object-oriented programming concepts can be widely used to develop the finite element structural analysis program for personal computers.

1. 서 론

대표적인 수치해석 방법인 유한요소법은 컴퓨터의 급속한 발전과 더불어 여러 공학분야에서 광범위하게 사용되어 이에 근거한 컴퓨터 프로그램의 개발로 보다 정확한 구조해석이 가능하게 되었다. 지금까지 많은 구조해석 프로그램들이 개발되어 널리 사용되고 있는데 대부분이 범용프로그램(General purpose program)이라 불리는 대형 컴퓨터용이었다. Bathe 등에 의해 개발된 SAP[1]과 같은 범용 구조해석 프로그램은 지금까지 많이 사용되어 왔으나 비교적 다양하고 정확한 구조해석기능에 비해 실행시간이 길며 자료의 입출력이

* 영남대학교 토목공학과 부교수
** 영남대학교 대학원 박사과정
*** 영남대학교 공업기술연구소 연구원
**** 영남대학교 대학원 석사과정
• 이 논문은 1991년도 교육부지원 한국학술진흥재단의 자유공모과제 학술연구조성비에 의하여 연구되었음.

이논문에 대한 토론을 1993년 3월 31일까지 본학회에 보내주시면 1993년 9월호에 그 결과를 게재하겠습니다.

불편할 뿐 아니라 소형구조해석 및 부분적인 실행의 경우에도 프로그램 전체를 실행시켜야 하며 부분적인 프로그램의 수정이 거의 불가능하였다. 그러나 최근 10여년간 개인용 소형컴퓨터(Personal computer)의 폭넓은 보급과 기억용량 및 처리속도의 향상은 이러한 범용프로그램의 단점을 개선한 PC용 프로그램의 개발을 촉진시키고 있다.

지금까지 개발된 PC용 구조해석 프로그램들은 범용프로그램이 가지고 있는 여러가지 구조해석의 기능은 가지고 있지만 전공지식이 충분치 않은 경우에 프로그램 이용이 곤란하며 사용자가 Macro command 등을 잘못 사용했을 경우와 Error message가 나타났을 경우에는 더 이상의 해석이 불가능하게 되어 있다. 일반적으로 PC용 구조해석 프로그램은 다양한 메뉴선택과 해석기능을 가지고 있어야 하나 FORTRAN 및 BASIC 언어로 작성된 기존의 프로그램들은 단지 정확하고 신속한 처리에만 주력하고 있을 뿐 다양한 메뉴의 선택, 부분적인 프로그램의 추가 및 수정 등이 용이하지 않다. 최근 이러한 단점을 보완하기 위해 프로그램이 간편하고, 확장 및 수정이 매우 간단하며 프로그램의 신뢰성 및 재사용성이 뛰어난 객체지향 프로그래밍(Object-oriented programming)기법[2]이 제안되었다.

객체지향 프로그래밍 기법의 주된 목적은 객체(Object)라 불리우는 독립적이고 항시 사용 가능한 소프트웨어의 구성하는 것인데, 이는 대규모의 응용 프로그램을 개발하고자 할 때 각 소프트웨어를 구성요소의 병렬응용범위에서 설계할 수 있고, 비슷한 응용에 대해 기존의 Code를 재사용할 수 있으며 Prototype을 효과적으로 형성할 수 있음을 의미한다. 이것은 대형 응용프로그램의 유지, 보수와 개발을 신속히 할 수 있게 한다. 자료흐름 방식이나 수학적 논리를 기초로 하는 기존의 절차적 언어(Procedural language)의 프로그래밍 방식과는 달리 객체지향 프로그래밍은 응용하고자 하는 대상을 직접 모델링할 수 있으며 특히 객체지향적 개념에 의해 특별히 개발된 언어인 Smalltalk[3]과 같은 언어와는 달리 본 연구에서 사용된 C++언어[4]는 기존의 C언어의 효율성과 객체지향적 개념을 함께 포함하고 있는 Hybrid 언어이기 때문에 C언어로 작성된 기존의 프로그램을 그대로 이용할 수 있다는 장점이 있다.

이러한 객체지향 프로그래밍 기법에 대한 연구는 1986년경 전산공학 분야에서 제안되어 구조해석분야에서도 1990년 Forde 등[5]이 객체지향 유한요소해석에 관한 연구를 발표한 이래 이에 관한 많은 연구가 활발히 진행되고 있다. Fenves [6], Zimmermann 등[7]의 연구에서 Smalltalk과 같은 객체지향적 언어가 유한요소해석을 다룰 수 있음을 보여 주었고, C++언어를 사용한 Mackie [8], Scholz[9] 등의 연구가 있으나 아직은 유한요소해석 프로그램에 대한 피상적인 개념 설명에 불과하다고 할 수 있다.

따라서 본 연구에서는 이러한 객체지향 프로그래밍 기법으로 매트릭스의 연산과 평면뼈대 구조해석이 가능한 PC용 프로그램을 개발하여 앞으로의 객체지향 유한요소 프로그램 개발에 활용하고자 한다.

2. 객체지향 프로그래밍의 개념

객체지향 프로그래밍은 함수들과 함수의 처리대상이 되는 자료(Data)들을 주어진 문제의 범위에 있는 객체로 취급하여 이 객체의 성격을 파악하고 그 객체를 구현하는 과정이 프로그램의 작성과정이 된다. 자료구조와 제어구조로 된 기존의 프로그래밍 기법과는 달리 객체지향 프로그래밍은 객체(Object), 클래스(Class), 처리방식(Method), 메시지(Message), 상속성(Inheritance), 다형성(Polymorphism) 등의 여러가지 개념으로 설명된다.

2.1 객체

객체는 클래스의 실체로서 자료에 대한 정보와 처리능력(Procedures)를 갖고 있는데 각각의 객체는 클래스에 명시된 처리방식에 의하여 독립적으로 명령들을 수행한다. 프로그램 개발자에게 자료와 처리과정을 분리하여 표현하거나 그것들의 상호관계를 분리하여 관리하게 하는 기존의 프로

그래밍 기법과는 달리 객체지향 프로그래밍 기법에서 객체는 특정 종류의 자료에 대한 정보나 연산 등과 그 자료를 운영하는 특정 처리과정이 캡슐화(Encapsulation)되어 있으므로 사용자는객체가 어떤 방식으로 구성되어 있는지에 대해 무지해도 상관없다. 하위객체의 조합으로 생성되는 상위객체는 그 하위객체들이 제공하는 기능을 선택적으로 취한다. 이러한 객체에 대한 추상적 개념은 확대하여 소프트웨어에 대해서 응용할 수 있으며, 이런 프로그램은 계층관계를 가지는 서로 다른 객체들로 구성된다. 객체들은 메시지를 주고 받음으로써 서로 정보와 자료들을 교환할 수 있다. 즉, 한 객체가 메시지를 전달받으면 먼저 메시지를 분석하고 메시지의 내용이나 명령대로 정보나 자료를 처리한다. 객체내부에서의 정보나 자료처리 등은 프로그램의 다른 부분의 변화없이 변경될 수 있다. 이것은 절차적 프로그래밍 방식에서의 "call-by-reference" 또는 "call-by-value"기법과 비교하여 "call-by-desire"라고 할 수 있는 메시지 전달 특성을 가지기 때문이다.

2.2 클래스

클래스는 추상적 자료형식이라고 정의할 수 있다. 즉, 동일한 종류의 자료정보와 처리방식을 가지고 있는 객체들의 집합체를 클래스라고 부른다. 이것은 Pascal, C와 같은 기존 언어에서 볼 수 있는 구조화된 자료의 개념과 유사하다. 또한 표준 Pascal 또는 C에서의 변수들이 주어진 자료형식의 인스턴스(Instance)인 것과 같이 객체는 클래스의 인스턴스라고 할 수 있다. 클래스는 각 객체에 대한 인스턴스 변수와 처리방식 등을 정의하고 있다. 또한 클래스는 주어진 객체형식의 조합을 나타내는 형틀(Template)이라고 할 수 있으며, 이러한 형틀은 보통 일련의 인스턴스 변수와 클래스에 의해 인식된 "Protocol"이라 불리는 메시지를 포함한다. 각각의 클래스는 다른 클래스로 부터 전달되는 명령들을 주고 받을 수 있는 인터페이스(Interface)를 갖고 있다.

2.3 처리방식

클래스에 명시되어 있는 객체들의 명령수행방법과 정보처리방법을 말한다. 메시지에 의해 호출된 처리방식은 각 객체에 지장되는 인스턴스 변수의 값을 조정한다. 메시지가 전달되면 각 클래스들은 이에 관련된 처리방식에 대한 Protocol을 찾은 다음 각 객체들이 갖고 있는 자료를 이용하여 명령을 수행하게 된다. 몇몇 객체지향 언어는 처리방식이 위와 같이 인스턴스 변수를 운영하는가, 클래스 변수를 운영하는가에 따라 서로 대별될 수 있다. 클래스 변수를 지원하는 언어는 보통 클래스에 직접 메시지를 보내기 위해 또다른 Protocol 을 제공한다. 이러한 처리방식을 클래스 처리방식(Class method) 또는 Metamethod라고 한다. 위에서 언급한 바와 같이 기존의 프로그램 방식에서는 자료와 처리방식이 분리되어 있으나 객체지향적 프로그래밍에서는 객체에 포함되어 존재한다.

2.4 상속성

각 클래스들은 상위클래스로 부터 자료와 처리방식 등을 상속받는다. 이것은 주어진 응용에 맞춰 새로운 객체를 형성할 수 있음을 뜻한다. 상속이란 이미 존재하는 객체와 거의 유사하나 조금 다른 성질을 갖는 새로운 객체를 정의하는 능력이다. 즉 상위 클래스(Superclass)의 정보들을 찾아 하위 클래스(Subclass)에서 사용하거나 C++의 컴파일러와 같이 그 정보를 하위 클래스에서 복사하여 공유하게 되어 재사용성을 증대시킨다.

2.5 다형성

객체의 일부분인 처리방식은 프로그램 전체에 대한 실체가 아니므로 서로 다른 객체는 같은 이름의 처리방식을 가질 수 있다. 이와 같이 각기 다른 객체들이 동일한 메시지에 대해 각각 다른 반응(처리방식)을 취하는 것을 다형성이라 하며, 이것은 시스템내에 이미 존재하는 메시지에 대해 응답할 수 있는 새로운 객체를 시스템에 삽입하는 것이 용이하다는 것을 의미한다.

2.6 개념적 모델

객체와 클래스, 처리방식 사이의 관계에 대한 개념적 모델을 나타낸 것이 그림 1 및 그림 2이다. 그림1과 같이 객체는 객체의 명칭, 그 객체가 속한 클래스와의 연결고리, 인스턴스 변수의 저장 영역 등을 가진다. 클래스는 그림 2와 같이 명칭, 상위 클래스와의 연결고리, 인스턴스 변수의 형틀, 메시지와 처리방식의 처리판과 부속 처리 과정들로 구성되어 있다.

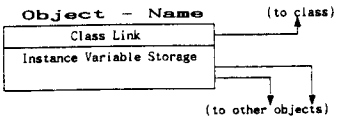

그림 1. Graphical representation of an object

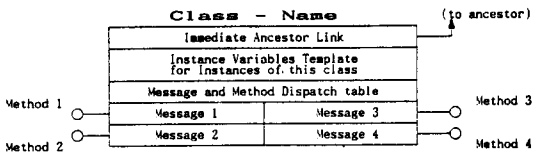

그림 2. Graphical representation of a class

3. 객체지향 구조해석 프로그램의 구성

본 연구에서 개발된 객체지향 구조해석 프로그램은 그림 3과 같이 주 클래스(Main class), 매트릭스 연산 및 벡터 클래스(Matrix & Vector class), 구조해석 클래스(STRU class)로 구성되어 있으며, 이 중 주 클래스와 구조해석 클래스는 추상적 클래스이다. 이 객체지향 구조해석 프로그램은 VGA카드가 있는 286급 이상(RAM 640KB 이상, 1 FDD이상, HDD 40MB이상)의 PC에서 실행될 수 있다.

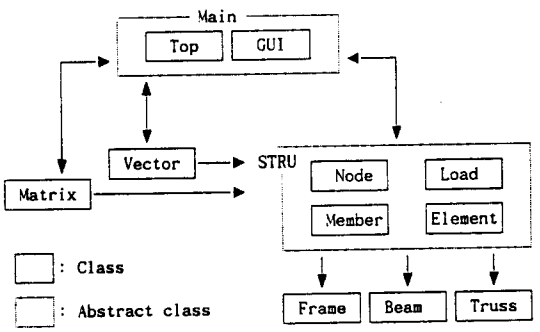

그림 3. Structure of object-oriented structural analysis program

3.1 주 클래스

주 클래스는 매트릭스 연산 클래스, 구조해석 클래스들을 총괄하는 입출력 제어, 명령의 전달 등을 수행하는 최상위 클래스(Top class)와 자료의 입출력과 저장을 담당하는 GUI(Graphic user interface) 클래스를 포함하는 추상적 클래스이다. 그림 4는 최상위 클래스의 생성자가 수행됨과 동시에 프로그램의 변화를 감지하여 그에 해당하는 명령 Icon의 인스턴스를 수행시키는 Event클래스의 Member 함수가 Event Loop안에서 수행되는 주 클래스의 내부 수행체계를 나타낸다. 그림 5는 GUI클래스중 Window의 구성과 다른 클래스와의 관계를 도식화한 것이다.

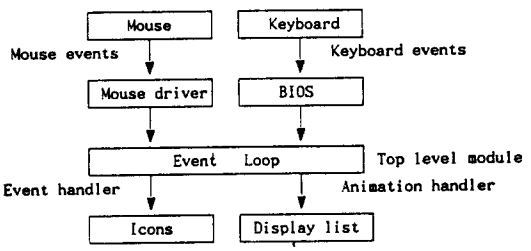

그림 4. Connection of user and Icons in main class

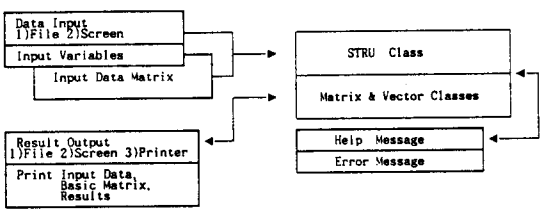

그림 5. Strategies of object-oriented structural analysis program

3.2 매트릭스 연산 및 벡터 클래스

이 클래스는 구조해석시 최소한 한번이상 이용되는 클래스로 벡터 및 행렬의 연산과 고유값 문제의 해석을 수행한다. 각 클래스의 구성은

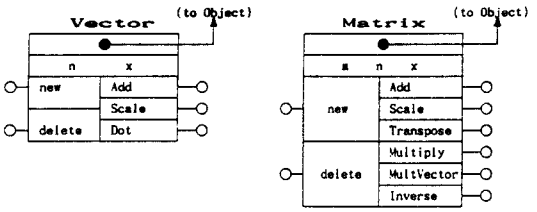

그림 6. Description of Matrix and Vector classes

그림 6과 같다. 그림의 왼쪽은 함수, 오른쪽은 변수를 나타낸다.

아래 클래스들은 벡터, 매트릭스 클래스의 여러 연산중 곱셈의 예를 나타낸 것인데, 여기서 행렬의 연산과 벡터의 연산은 동일한 형태의 연산자로 처리되며 하나의 함수가 두개이상의 클래스에 대하여 연산을 수행하도록 "friend" 명령어를 사용한다. 하나의 클래스내에 동일한 함수명으로 서로 다른 인수를 가지는 함수를 중복하여 선언하는 C++언어의 고유한 초과연산자의 특징과 두 클래스를 통하여 서로다른 클래스내에 각각 초과연산자들이 선언되어 있는 다형성을 잘 나타내고 있다. 또한 프로그램상에서 매트릭스와 벡터를 하나의 변수형태로 하여 스칼라변수와의 연산이 가능하다.

```
class vector :
   dvector operator * (double& dscalor);
   double operator * (dvector& n);
   friend dvector operator * (dmatrix& m,
   dvector& n);
   friend dvector operator * (dvector& n,
   dmatrix& m);
   friend dvector operator * (double& dscalor,
   dvector& n);


class matrix :
   dmatrix operator * (double& dscalor);
   dmatrix operator * (dmatrix& m);
   friend dvector operator * (dmatrix& m,
   dvector& n);
   friend dvector operator * (dvector& n,
   dmatrix& m);
   friend dmatrix operator * (double& dscalor,
   dmatrix& m);
```

다음과 같이 정의함으로써 프로그램상에서 매트릭스와 벡터를 하나의 변수형으로 하는 스칼라변수와의 연산이 가능하다.

```
dmatrix m(1, 3, 1, 3, 999);
dvector v(1, 3, 999), v1(1, 3, 0);
double d=1.23454;
v1=m * 3.123 * m  * v * 2;
```

또한 기본적인 매트릭스 연산외에 부수적으로 다음과 같은 수치적 연산이 가능한 특수한 형태의 클래스를 상속하여 만들면 특수한 연산(고유값 문제 등)도 쉽게 행할 수 있다. 다음 연산은 어떤 매트릭스의 역행렬을 다시 전치하여 스칼라와 매트릭스와 곱하는 것을 의미한다. 모든 변수를 복사한 뒤 사용함으로써 다음과 같은 다항 연산을 가능하게 한다.

```
v1=trans(m~) * 3.123 * m * v * 2;
```

C++언어에서의 클래스 선언은 Header file과 Program file을 포함하는데 Header file에서 클래스의 정의를 한다. 클래스의 선언은 전용변수(Private)부분과 공용변수(Public)부분으로 나눌 수 있다. 전용변수 부분에서 선언되는 변수와 member함수들은 다른 객체나 함수로 부터 사용될 수 없는 반면에 공용변수 부분의 선언은 모든 객체나 함수에 사용될 수 있다.

3.3 구조해석 클래스

구조해석 클래스는 그림 7 및 그림 8과 같은 클래스들을 포함하는 추상적 클래스이다. 이 클래스에서는 해석하고자 하는 구조물의 기하학적, 재료학적 자료 및 경계조건이 Node클래스 및 Member클래스에, 하중에 대한 자료가 Load클래스에 각각 입력되어 Element클래스에 의해 요소강성도 매트릭스를 구성한 후, 전체 구조물에

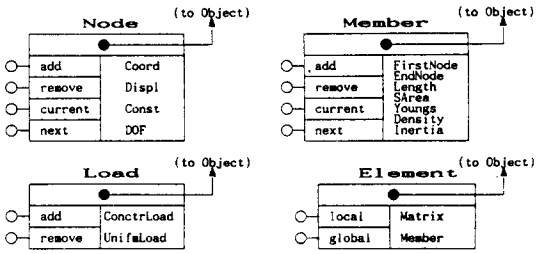

그림 7. Description of STRU class

대한 강성도 매트릭스를 조합하여 변위와 응력을 구하는 매트릭스 변위법으로 평면 뼈대 구조물을 해석한다.

(1) Node, Member, Load 및 Element 클래스

Node와 Member클래스는 평면 뼈대 구조해석에서 기본적으로 필요한 자료를 기억시키고 관리하는 클래스들로서 Node클래스의 공용변수부분에 정의되어 있는 것은 실제 자료기억변수들로서 절점번호나 Window상에서의 순서에 따라 해상절점을 등록, 추가, 삭제한다. Member클래스의 선언은 두 절점으로 구성된 1차원 요소의 양쪽 절점의 위치를 기억하는 pointer변수와 실제의 부재요소의 특성치를 기억하기 위한 변수들로 구성된다. 하중자료는 Load클래스에 입력, 저장되며 Element클래스에서 강성도 매트릭스가 구성된다. 다음은 Node클래스의 Header file이다.

```
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( ) ;

} ;
```

(2) Frame, Beam 및 Truss 클래스

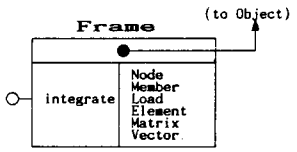

그림 8. Description of Frame class inherited from STRU class

그림 8은 실제의 평면뼈대 구조물을 해석하기 위한 Frame클래스로서 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.

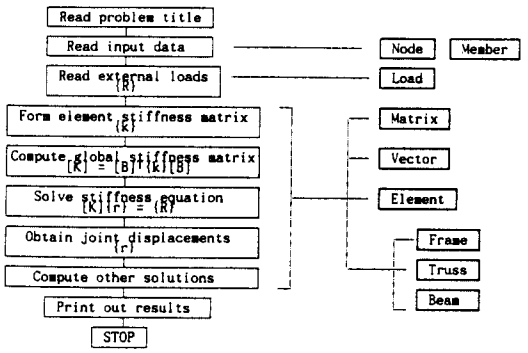

그림 9. Relationship between classes and solution procedure by matrix displacement method

(3) NodeList 및 MemberList클래스

앞에서 언급한 클래스외에 각 Node 및 Member의 객체를 연결하여 해석하고자 하는 구조형태를 구성해 주는 NodeList클래스와 MemberList클래스는 다음과 같다.

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

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

4. 예제해석 및 고찰

객체지향 매트릭스 구조해석 프로그램의 기능과 효율성을 검토하기 위해 다음과 같은 예제해석을 Tele-386 PC(RAM 4MB, 2 FDD, HDD 80MB, VGA card)로 수행하였다.

4.1 고유치 문제

Jacobi 방법에 의하여 (3 * 3) 정방행렬의 고유값과 고유벡터를 다음과 같이 계산하였는 바 정확한 값을 얻을수 있었다.

Input Matrix :

1.0	2.0	3.0
2.0	3.0	3.0
3.0	3.0	3.0

Output Eigenvalues And Eigenvectors

Number of JOCOBI rotations : 11

Eigenvalues :

−0.338922	7.516538	−1.177617

Eigenvectors :

Number 1		
0.425090	−0.829153	0.363048
Number 2		
0.482739	0.546953	0.683955
Number 3		
−0.765677	−0.115485	0.632773

4.2 평면 뼈대 구조해석

그림 10과 같은 고정 지점의 평면 뼈대 구조물을 해석하였는 바, FORTRAN 등으로 작성된 기존의 프로그램에서와 같은 정확한 결과를 얻었다.

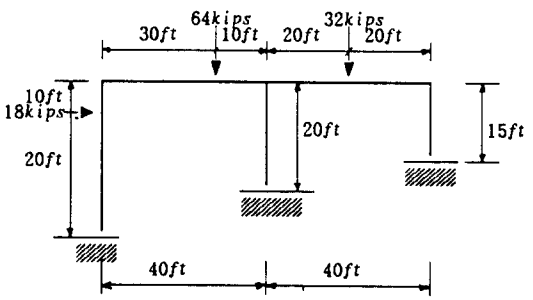

그림 10. Configuration of a plane frame with fixed supports

INPUT BASIC MATRIX

MATRIX A

1	2	3	4	5	6	7	8	9	10
1.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
0.0000	1.0000	1.0000	0.0000	0.0000	1.0000	0.0000	1.0000	0.0000	0.0000
0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000
0.0000	0.0000	0.0000	0.0000	-0.0334	-0.0334	-0.0500	-0.0500	-0.0667	-0.0667

MATRIX S

1	2	3	4	5	6	7	8	9	10
300.00	150.00								
150.00	300.00								
		300.00	150.00						
		150.00	300.00						
				240.00	120.00				
				120.00	240.00				
						360.00	180.00		
						180.00	360.00		
								320.00	160.00
								160.00	320.00

MATRIX P

1	2	3	4	5	6	7	8	9	10
40.00	234.38								
-200.00	0.00								
-160.00	-234.00								
13.33	0.00								

RESULTS OF DISPLACEMENT METHOD

MATRIX r: JOINT DISPLACEMENT

	COLUMN#1	COLUMN#2
ROW#1	1.371913E+02	4.148890E+02
ROW#2	-1.861989E+02	-3.238090E+02
ROW#3	-1.865499E+02	-4.355324E+02
ROW#4	5.127893E+02	-1.266007E+02

5. 결 론

본 연구에서는 C++언어를 이용한 객체지향 프로그래밍 기법으로 매트릭스의 연산과 평면뼈대 구조물의 구조해석이 가능한 PC용 구조해석 프로그램을 개발하였다. 이 객체지향 프로그램은 자료뿐 아니라 Code도 메모리(Memory)에 동적으로 생성되고 또한 동적으로 소멸되는 "Dynamic Memory Allocation" 특성을 가지므로 작은 기억용량을 사용하더라도 프로그램이 정적으로 고정되어 있는 절차적 언어 프로그램보다 처리속도가 빠르며 상속성 및 다형성과 같은 특징으로 인하여 프로그램 Size도 동일한 내용의 절차적 프로그램보다 축소된다. 본 연구에서 개발된 프로그램의 효율성을 예제해석을 통하여 검증한 결과 만족할 만한 처리속도 및 결과의 정확도를 고찰할 수 있었다.

따라서 객체지향 프로그래밍 기법의 특징인 프로그램의 확장성과 재사용성 및 다양한 GUI의 구현 가능성 등 여러가지 장점때문에, 앞으로 객체지향적 프로그래밍 기법을 이용한 다양한 구조해석 프로그램의 개발이 본격화될 것으로 판단된다.

참 고 문 헌

[1] K.J. Bathe, E.L. Wilson and F.E. Peterson, "SAPIV-A structural analysis program for static and dynamic response of linear systems", Report No.EERC 73-11, Earthquake Engineering Research Center, Univ. of California, Berkeley, 1973.

[2] R.S. Wiener and L.J. Pinson, *An Introduction to Object-Oriented Programming and C++*, Addison-Wesley, MA, 1989.

[3] A. Goldberg and D. Robson, *Smalltalk-80, The Implementation*, Addison-Wesley, MA, 1983.

[4] B. Stroustrup, *The C++ Programming Language*, Addison-Wesley, MA, 1986.

[5] B.W.R. Forde, O.F. Ricardo and F.S. Siegfried, "Object oriented finite element analysis", Computers & Structures, Vol.34, No.3, pp.355-374, 1990.

[6] G.L. Fenves, "Object-oriented programming for engineering software development", Engineering : I. Governing Principles", Comput. Meth.Appl.Mech.Engng., Vol.98, pp.291-303, 1992.

[7] T. Zimmermann, Y. Dubois-Pelerin and P. Bomme, "Object-oriented finite element programming : I. Governing Principles", Comput. Meth.Appl.Mech.Engng., Vol.98, pp.291-303, 1992.

[8] R.I. Mackie, "Object oriented programming of the finite element method", Int.J. Numer-.Meth.Engng., Vol.35, pp.425-436, 1992.

[9] S.P. Scholz, "Elements of an object-oriented FEM++ program in C++", Computers & Structures, Vol.43, No.3, pp.517-529, 1992.

[10] G.J. Miller, "An object-oriented approach to structural analysis and design", Computers & Structures, Vol.40, No.1, pp.75-82, 1991.

[11] B.W. Kernighan and D.M. Ritchie, *The C Programming Language*, 2nd ed. Prentice-Hall, 1988.

(접수일자 : 1992. 10. 24)