韓國數學敎育學會誌 시리즈 E
<數學敎育 論文集>
제 13집, 2002. 2. 535-547.

J. Korea Soc. Math. Ed. Ser. E:
Communications of Mathematical Education
Vol. 13, Feb. 2002. 535-547.

자바를 활용한 수치계산에서의 심볼릭 연산 알고리즘[1)]

김 철 수 · 김 익 찬 · 양 준 영 (제주대학교)

본 논문은 교육현장에서 자바(Java)를 이용한 수치계산 애플릿(Applet)을 개발할 경우 수식을 인식하여 그 결과를 실행하고 보여주는 심볼릭 연산을 구현하기 위한 알고리즘 개발과 다양한 입력식을 처리하기 위한 효율적인 자료구조를 제안한다. 구현된 패키지내의 클래스는 변수와 상수, 다양한 연산자를 처리하기에 적합하며 분석된 정보를 통해 사칙연산의 처리, 연산자 우선순위의 처리, 심볼릭 연산, 다항식, 방정식, 함수의 그래프 작성, 간단한 미적분 처리를 하는 알고리즘을 제안한다.

1. 서 론

인터넷의 보급과 더불어 자바언어는 인터넷기반의 응용프로그램을 개발하기위한 언어로서 그 유용성과 중요성이 크게 부각되고 있고 활용범위도 넓혀 나가고 있다. 자바기반의 다양한 소프트웨어들이 응용프로그램, 애플릿형태로 개발되어 널리 보급되고 있으며 학교현장에서도 수업시간에 애플릿을 이용한 학습자료들이 이용되고 있다. 그간의 수치계산 또는 수학 및 공학을 위한 소프트웨어들이 컴퓨터 플랫폼에 종속적이라는 문제를 갖고 있었지만 자바는 인터넷을 배경으로 한 플랫폼에 독립적인 언어로서의 강력한 장점을 갖고 있어 인터넷 또는 웹기반의 계산에 혁명적이라 할 만큼 과학계산방법에 새로운 지평을 열어주고 있는 컴퓨터언어이다. 자바는 객체지향언어로서 자바가상기계가 설치된 컴퓨터에서 실행된다. 이러한 특징은 자바언어가 갖는 장점으로 인터넷 또는 웹기반의 교육자료를 자바를 이용하여 애플릿형태로서 구축하는데 도움을 주고 있다.

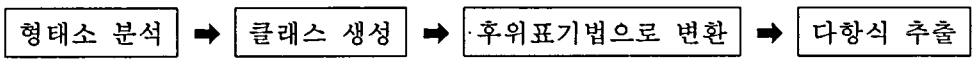
하지만 이러한 애플릿프로그램의 경우 수식처리 및 연산의 경우 심볼릭연산보다는 자바소스 프로그램내에 수식, 함수들을 정의하거나 클래스의 함수를 호출하여 프로그램을 작성하여 컴파일된 결과이므로 임의의 수식의 자유로운 입력, 변수와 상수의 연산, 다항식의 심볼릭처리등에 있어서 제한성을 갖고 있다. 현재까지 개발된 심볼릭 연산이 가능한 대표적인 소프트웨어중의 하나는 매쓰매티카(Mathematica)로서 자연과학 및 공학분야등에서 많이 이용되고 있으나 이 소프트웨어는 인터넷기반의 소프트웨어가 아닐 뿐 아니라 상용소프트웨어이기 때문에 자유롭게 이용할 수 없어 학교현장에서 활용하는데 어려움이 따른다. 본 논문에서는 자바를 이용하여 수식계산, 수치계산등을 처리할 경우 심볼릭 연산이 가능하도록 알고리즘을 개발하고 이를 클래스화하여 심볼릭연산처리가 가능한 자바 수치계산 프로그램을 개발하기 위한 프로토타입과 소프트웨어를 개발하여 이를 활용할 수 있도록 하는데 있다.

1) 이 논문은 2000년도 제주대학교 발전기금 학술연구비에 연구되었다.

2. 심볼릭 연산 알고리즘의 구현

2.1 식 인식과 연산 방식

식을 인식하기 위해 크게 다음과 같은 과정을 거친다.



형태소 분석은 가장 기본적인 작업이라 할수 있다. 입력식은 항상 문자열이므로 문자열내에 존재하는 변수, 상수, 연산기호를 분리해내야 하기 때문이다. 입력식에서 변수, 상수, 연산기호를 분리하고 각각의 요소를 처리하는 클래스를 생성한다. 상수를 처리하는 클래스는 JIntValue, JDoubleValue Interface를 계승받는다. 상수값은 정수와 실수로 분리됨을 의미하며 변수와 같이 아직 값이 정수, 실수를 분간할 수 없는 경우를 위해 JIntValue, JDoubleValue의 상위 Interface인 JNumerical를 두었다.

변수는 변수를 나타내는 심볼과 심볼의 값, 두쌍으로 이루어지며 심볼과 심볼의 값 매칭시 성능의 향상을 위해 심볼을 키로 갖는 HashTable로 구현했다. 연산에 관련된 모든 클래스는 JEvaluator를 상위 클래스로 갖도록하여 동일 메소드를 통해 결과값을 얻을 수 있도록 하였다. 추가로 고려해야 할 사항은 연산을 하는데 있어 필요한 인자의 개수는 모두 다르다는 점이다. 패키지내에서 정의하는 연산 클래스의 인자는 0, 1, 2개를 필요로 하며 1, 2개의 인자의 특성을 명시하기 위하여 JEvaluatorArg1, JEvaluatorArg2를 정의하였으며 이 두 개의 Interface는 JEvaluator Interface를 부모로 갖도록하여 서로 다른 인자를 갖는 연산 클래스에 필요한 인자를 입력받을 수 있게 하며 모든 연산 클래스는 JEvaluator의 메소드로 동일하게 접근할 수 있도록 하였다. 이와 같은 Interface의 사용은 다음처럼 간단한 방식으로 연산루틴을 작성할수 있도록 한다.

```
JObject obj , operand1, operand2;
Stack  stack;

for ( ; ( obj = getNext( ) ) != null ; ){
  if ( obj instanceof JEvaluator ){
    if ( obj instanceof JEvaluatorArg1 ){
       operand1 = stack.pop( );        obj.setArg( operand1 );
    }else if ( obj instanceof JEvaluatorArg2 ){
       operand2 = stack.pop( );        operand1 = stack.pop( );
       obj.setArg( operand1, operand2 );
```

```
    }
      stack.push( obj.evaluate( ) );
    }
    else{
      stack.push( obj );
    }
  }
  return stack.pop( );
```

이 연산방식은 후위표기식의 계산방법에 기인한 것인데 후위표기식으로 변환된 식을 하나씩 얻어 인자를 하나를 필요로하는 클래스와 두 개의 인자를 필요로하는 클래스를 조건에 맞게 판별하여 연산에 필요한 인자의 개수만큼 스택에서 꺼내 입력하여 주고 모두 동일하게 evaluate method로 연산결과를 얻어내고 있다. 위 과정을 예제를 통해 따라가보자.

Sin(3 * 10)

<입력식 1>

<입력식 1>의 형태소 분석결과 결과 1로 분리된다.

Sin	3	*	10

<결과 1>

<결과 1>로부터 <결과 2>에서 보는바와 같이 각 형태소를 처리할 클래스를 생성한다.

<JSin(Sin)> <JInt(3)> <JMuliply(*)> <JInt(10)>

클래스명	구현 interface
JSin	JEvaluator, JEvaluatorArg1, JObject
JInt	JNumerical, JEvaluator, JObject
JMultiply	JEvaluator, JEvaluatorArg2, JObject

<결과 2>(( ) : 클래스에 초기화되는문자열)

<결과 2>에서 후위표기식으로 바꾸어 <결과 3>을 얻을수 있다.

<JInt(3)> <JInt(10)> <JMuliply(*)> <JSin(Sin)>

<결과 3>

<결과 3>을 가지고 연산과정시 어떻게 처리되는지 <흐름 1>에 나타내었다.

Input	Stack			비 고
	0	1	2	
<JInt(3)>	3			JInt는 연산시 필요한 인자가 없으므로 3을 리턴한다.
<JInt(10)>	3	10		JInt는 연산시 필요한 인자가 없으므로 10을 리턴한다.
<JMultiply(*)>	3	10		JMultiply는 연산시 필요한 인자가 2개이므로 스택에서 2개의 값을 꺼내 JMultiply에 대입한다.
<JMultiply(*)>	30			JMuliply의 연산으로 3*10의 값이 리턴된다.
<JSin(30)>	30			JSin은 연산시 필요한 인자가 1개이므로 스택에서 1개의 값을 꺼내 JSin에 대입한다.
<JSin>	0.5			JSin의 연산으로 Sin(30)의 값이 리턴된다.
결과값 : 스택에서 0.5를 꺼내 리턴한다.				

<흐름 1>

2.2 심볼처리 방식

위 <입력식 1>의 예는 모두 상수에 관한 연산이었으므로 후위표기식과 후위 표기식의 연산처리 방법과 크게 다르지 않다. 하지만 심볼 자체에 대한 처리시에는 위 자료형을 가지고 처리할 수 없다. 이를 해결하기 위하여 다항식 개념을 도입하였다. 다항식은 <심볼>, <지수> 두 값을 짝으로하는 하나 이상의 다항식 요소와 계수를 갖는 자료형이라 정의한다. 가령 $3x^2y$라는 다항식이 있다면 다항식을 처리하는 클래스인 JPolynomial에서 <보기 1>과 같이 자료를 관리한다.

계수항	다항식요소		다항식요소	
JInt	<심볼항>	<지수항>	<심볼항>	<지수항>
3	x	2	y	1

<보기 1>

다항식 요소는 심볼을 키값으로 하는 HashTable로 구현했다. 다항식은 계수항과 다항식 요소를

삽입시 연산이 이루어진다. 연산규칙은 다음과 같다.

1. 다항식내에 들어오는 상수는 계수항의 값과 곱셈 연산을 취한다.
2. 다항식 요소 삽입시 심볼을 키값으로 HashTable에서 검색하고 실패시에는 삽입되며 검색 성공시에는 검색결과 얻어진 다항식 요소의 지수항과 새로 삽입되는 다항식 요소의 지수항과 덧셈 연산을 한다.

<연산규칙 1>

$3x^2yx$라는 식이 있을 때 다항식에 자료삽입이 되는 경로를 <보기 2>에 제시했다.

<table>
<tr><th>입력값</th><th>다 항 식</th><th>비 고</th></tr>
<tr><td>3</td><td><table><tr><td>계수항</td><td>다항식 List</td></tr><tr><td>1</td><td>Null</td></tr></table></td><td>3은 상수이므로 연산규칙 1번을 따른다.</td></tr>
<tr><td>x^2</td><td><table><tr><td>계수항</td><td>다항식 List</td></tr><tr><td>3</td><td><table><tr><td colspan="2">다항식요소</td></tr><tr><td><심볼항></td><td><지수항></td></tr><tr><td>x</td><td>2</td></tr></table></td></tr></table></td><td><x>,<2>인 다항식요소이므로 연산규칙 2번을 따르며 검색실패인 경우이다.</td></tr>
<tr><td>y</td><td><table><tr><td>계수항</td><td>다항식 List</td></tr><tr><td>3</td><td><table><tr><td colspan="2">다항식요소</td><td colspan="2">다항식요소</td></tr><tr><td><심볼항></td><td><지수항></td><td><심볼항></td><td><지수항></td></tr><tr><td>x</td><td>2</td><td>y</td><td>1</td></tr></table></td></tr></table></td><td><y>,<1>인 다항식요소이므로 연산규칙 2번을 따르며 검색실패인 경우이다.</td></tr>
<tr><td>x</td><td><table><tr><td>계수항</td><td>다항식 List</td></tr><tr><td>3</td><td><table><tr><td colspan="2">다항식요소</td><td colspan="2">다항식요소</td></tr><tr><td><심볼항></td><td><지수항></td><td><심볼항></td><td><지수항></td></tr><tr><td>x</td><td>3</td><td>y</td><td>1</td></tr></table></td></tr></table></td><td><x>,<1>인 다항식요소이므로 연산규칙 2번을 따르며 검색성공인 경우이다.</td></tr>
</table>

2.3 다항식간의 연산처리 알고리즘

두 다항식의 덧셈은 항의 지수를 기준으로 정렬한 후 지수가 같은 항의 계수를 더하는 연산을 취하는데 심볼이 포함되었을 경우에는 심볼 + 상수, 심볼 + 심볼등의 연산이 발생할 수 있으므로 심볼에 대한 처리가 필요하다. 이를 처리하기 위하여 기준항을 중심으로 정렬하여 다항식내에서 계수 다항식을 만들어 내는 방식을 취했다.

1. 기준항을 리스트의 0번 데이타와 교환한다.
2. 리스트 1번 데이터부터 리스트의 끝까지 정렬한다.
3. 1번데이터부터 리스트이 끝까지 부분을 현재 계수항과 곱하고 그 결과값을 계수항을 대입한다.

<정렬규칙1>

<예제 2>에서 위 방식이 어떻게 처리되는지 살펴보자.

3zx^2y ➡
<기준항:x>

계수항	다항식요소		다항식요소		다항식요소	
JInt	심볼항	지수항	심볼항	지수항	심볼항	지수항
3	z	1	x	2	y	1

<예제 2>

<예제 2>는 2zx^2y식을 연산규칙 1번을 사용하여 초기화한 결과이다.

규칙 1.을 적용 <z, 1>, <x, 2>항을 교환한다.

계수항	다항식요소		다항식요소		다항식요소	
JInt	심볼항	지수항	심볼항	지수항	심볼항	지수항
3	x	2	z	1	y	1

규칙 2.을 적용 <z, 1>, <y, 1>항을 심볼항을 중심으로 정렬한다.

계수항	다항식요소		다항식요소		다항식요소	
JInt	심볼항	지수항	심볼항	지수항	심볼항	지수항
3	x	2	y	1	z	1

규칙 3.을 적용 <y, 1>, <z, 1>항을 계수항과 곱하여 계수항으로 대입한다.

<table>
<tr><td colspan="4">계수항</td><td colspan="2">다항식요소</td></tr>
<tr><td colspan="4">JPolynomial</td><td>심볼항</td><td>지수항</td></tr>
<tr><td>계수항</td><td colspan="2">다항식요소</td><td colspan="2">다항식요소</td><td rowspan="3">x</td><td rowspan="3">2</td></tr>
<tr><td>JInt</td><td>심볼항</td><td>지수항</td><td>심볼항</td><td>지수항</td></tr>
<tr><td>3</td><td>y</td><td>1</td><td>z</td><td>1</td></tr>
</table>

이제부터 다항식과 다항식간의 연산을 정의하는 것으로 다항식간의 연산을 처리할 수 있는데 가장 기본이 되며 가장 중요한 덧셈연산을 정의한다.

1. 두 다항식의 지수항을 비교하여 같지않으면 큰쪽을 먼저 삽입하고 작은쪽을 뒤에 삽입하고 삽입이 완료된 리스트를 리턴한다.
2. 두 다항식의 지수항이 같으면 계수항을 비교하여 계수항이 상수항만으로 구성되어 있으면 상수항을 더하고 리턴한다.
3. 계수항이 다항식이면 두다항식내의 다항식요소의 첫 번째 심볼항을 비교하여 작은 심볼값을 기준항으로 만들어 정렬규칙을 적용후 덧셈규칙 1번부터 재적용한다.

<덧셈규칙>

위 덧셈규칙을 <예제2>에서 살펴보자.

x^2z+x^2z

<예제2>

x^2z ➡

계수항	다항식요소		다항식요소	
JInt	심볼항	지수항	심볼항	지수항
1	x	2	z	1

1. 정렬규칙 적용

계수항			다항식요소	
JPolynomial			심볼항	지수항
계수항	다항식요소		x	2
JInt	심볼항	지수항		
1	z	1		

2. 덧셈규칙 1번 적용 두다항식의 심볼항의 지수항이 같으므로 덧셈규칙 2번 적용

계수항			다항식요소	
JPolynomial			심볼항	지수항
계수항	다항식요소		x	2
JInt	심볼항	지수항		
1	z	1		

계수항			다항식요소	
JPolynomial			심볼항	지수항
계수항	다항식요소		x	2
JInt	심볼항	지수항		
1	z	1		

3. 계수항이 모두 상수항이 아니므로 덧셈규칙 3번 적용

4. 심볼 'z'를 기준항으로 설정, 정렬규칙 적용후 덧셈규칙 1번부터 적용

계수항	다항식요소	
JInt	심볼항	지수항
1	z	1

계수항	다항식요소	
JInt	심볼항	지수항
1	z	1

5. 덧셈규칙 1번적용 지수항이 같으므로 덧셈규칙 2번 적용

6. 계수항이 모두 상수이므로 계수항끼리 덧셈연산을 수행하고 리턴

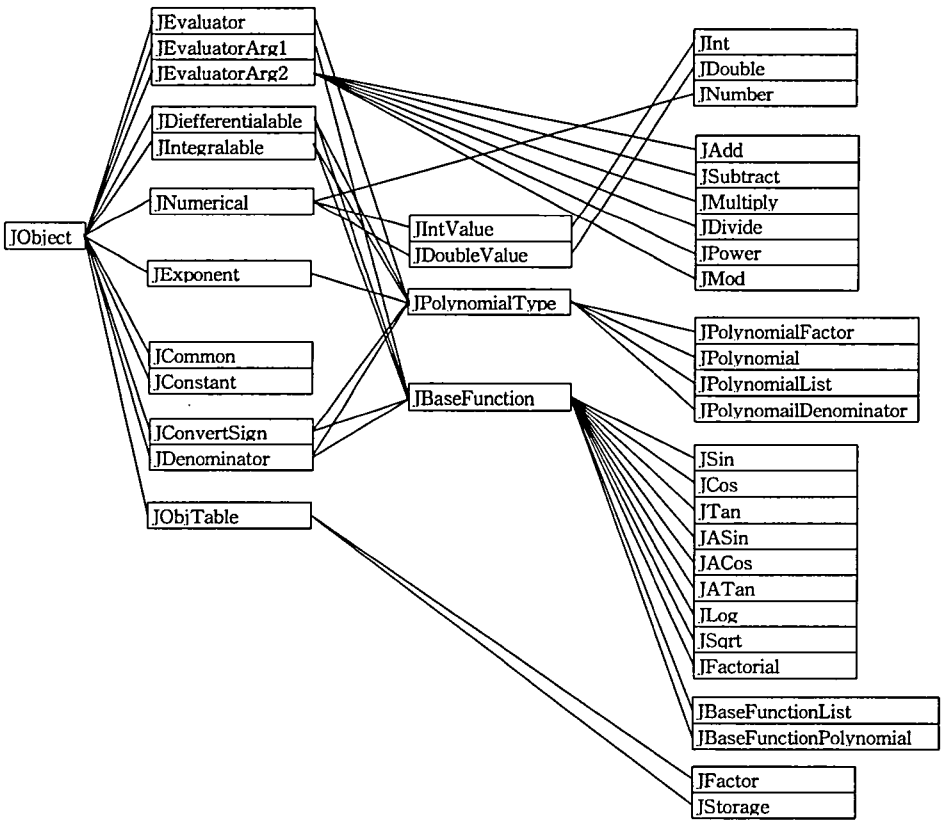
계수항	다항식요소	
JInt	심볼항	지수항
2	z	1

7. 최종결과

<table>
<tr><td colspan="2">계수항</td><td colspan="2">다항식요소</td></tr>
<tr><td colspan="2">JPolynomial</td><td>심볼항</td><td>지수항</td></tr>
<tr><td colspan="2"><table><tr><td>계수항</td><td colspan="2">다항식요소</td></tr><tr><td>JInt</td><td>심볼항</td><td>지수항</td></tr><tr><td>2</td><td>z</td><td>1</td></tr></table></td><td>x</td><td>2</td></tr>
</table>

기준항을 두는 것은 심볼간의 연산을 가능케하며 또한 원하는 심볼을 기준으로 식을 정렬하는 것을 가능케 한다. 뺄셈에 관한 연산은 JConvertSign Interface를 두어 식의 부호를 바꿀수 있게하여 뺄셈을 가할 연산자에 JConvertSign의 Method를 사용, 양/음수간의 변환을 가하고 덧셈연산규칙을 그대로 적용하여 구현할수 있다.

2.4 연산클래스 계층도

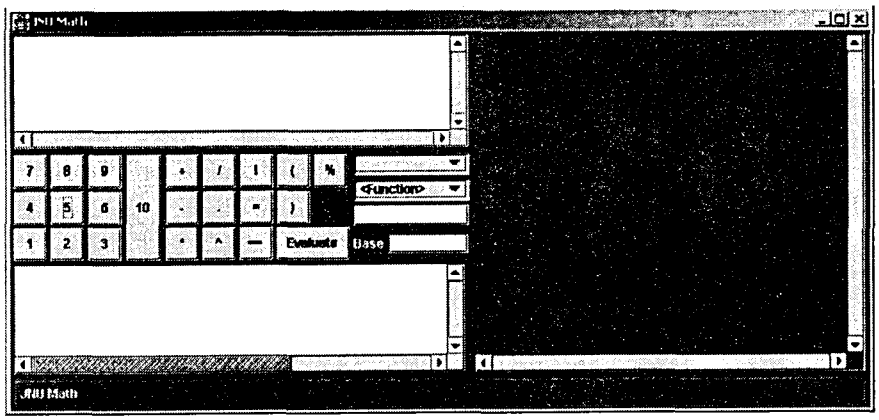


<연산클래스 계층도>

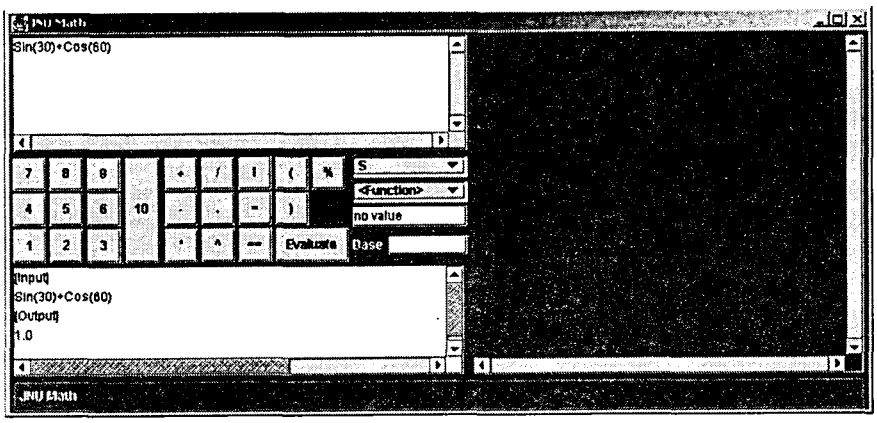
위 계층도는 연산과 자료구조를 담당하는 클래스와 인터페이스를 도시한것이며 위 구현 패키지를 이용하여 Parser, Evaluator등 20여개의 클래스가 패키지를 이용하여 구현되어 있으며 이 클래스들은 프로그래머와의 인터페이스 역할을 하여 메소드를 호출하는 방식으로 식 인식과 해를 구해준다.

2.5 패키지 / 예제 식 처리결과

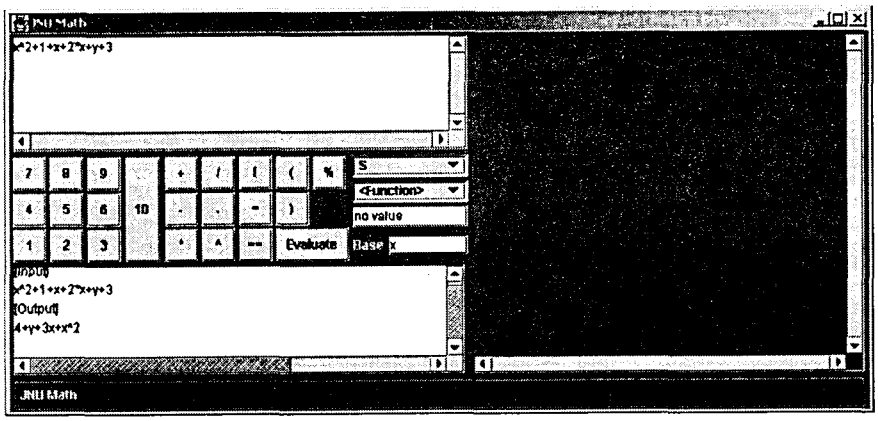
▶패키지 실행화면



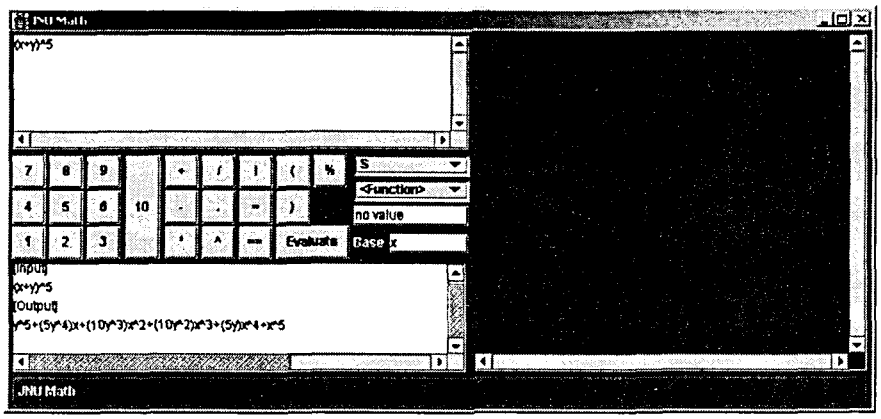
▶입력식 처리결과(상수)



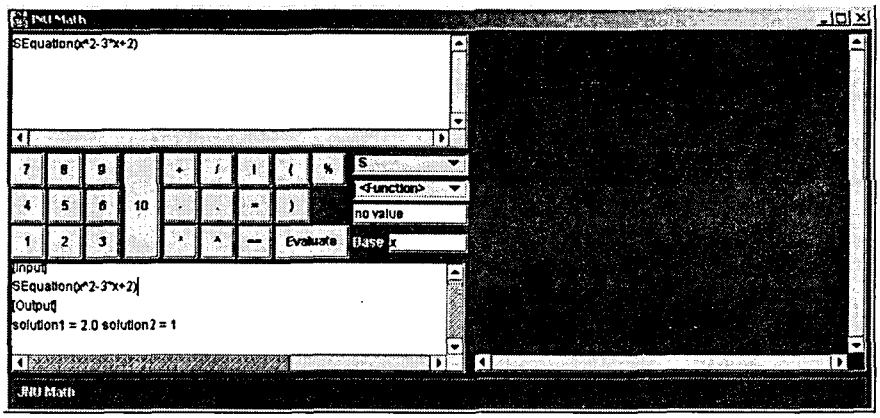
▶입력식 처리결과(상수, 심볼 혼용식)



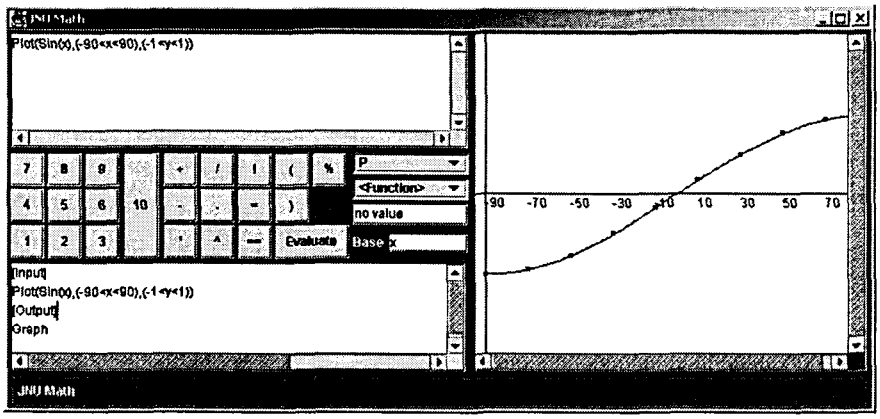
▶다항식 전개



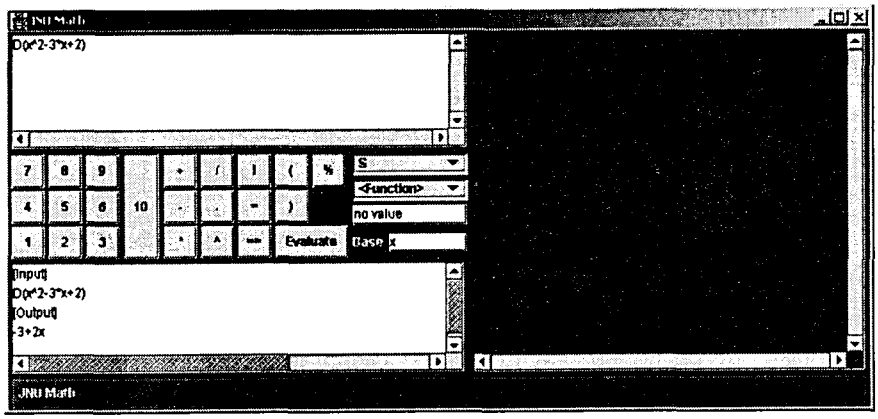
▶방정식 풀이(1, 2차 방정식)



▶그 래 프



▶미 분



3. 결 론

정해진 연산자를 가지고 조합되는 식의 종류는 무수히 많으며 연산자마다 연산을 취하는 방식이 모두 다르므로 식인식 패키지가 좋은 성능을 발휘하려면 잘 설계된 클래스를 그 모태로 갖는 자료구조가 가장 중요하다 할수 있다. 또한 오랜시간을 투자해 알고리즘을 최적화하는데 중점을 두어야 할 것이다. 자료구조로 표현하기 곤란한 문제들을 어떻게 처리해야할것인가 하는 문제도 중요하다 할 수 있다. 가령 Log(x), Tan(x) 같은 극한값을 갖는 함수인경우에는 그래프로 도식화하기가 곤란한다.

무한대를 화면에 표시하기 위해서는 상황에 맞게 적절한 좌표에 매칭시켜야 하는데 무한대를 갖는경우에 기준좌표가 모호해지기 때문이다. 이 문제점은 단정적인 하나의 문제를 제시하는 것으로 생각될수 있지만 근복적인 모든 문제를 대변한다 할수 있다. 수칙연산만으로 표현하기 어려운 자료형이 존재한다는 문제점과 새로운 자료형을 구현시에 이미 구현되어 있는 연산클래스와 자료형의 호환문제가 대두된다. 최근들어 급속도로 성장하고 앞으로도 많은 성장 가능성을 가진 자바를 이용하여 다양한 식을 인식할수 있는 기본 자료구조와 알고리즘의 개발에 있어 아이디어를 본 논문에서 제시한다. 식인식처럼 복잡한 내부구조를 갖는 경우 성능에 많은 관심을 갖게되는데 아직까지 자바는 C로 구현된 패키지보다 좋은 성능을 갖지 못한다. 하지만 자바는 지속적으로 발전하고 있고 최근 몇 년 사이에도 그 성능에 많은 발전을 가져왔다. 또한 이제는 평이하게 되버린 인터넷에서 웹에서 강한 자바의 강점을 이용 Applet으로 수칙해석 패키지와 계산기를 개발하는 것은 교육현장에서의 높은 활용도와 실시간으로 해를 얻어낼수 있어 많은 잇점이 있어 교육자료를 개발하는데 매우 유용하게 적용될 수 있다. 앞으로도 다양한 시도와 많은 성능 측정을 통해 알고리즘을 최적화 하며 더 많은 연산을 수행할 수 있는 패키지 개발에 힘을 기울여야 할 것이다.

참 고 문 헌

Tan Kiat Shi. & Willi-Hans Steeb. (1998). *An Introduction to Computer Algebra Using Object-Oriented Programming.* : Springer.

William H. Press.; Saul A. Teukolsky; William T. Vetterling & Brian P. Flannery (1995). *Numerical Recipes in C* : Cambridge University Press.