論文
2-11-5

수학 프로그램 구현을 통한 체계적 "프로그래밍" 교수 자료 개발에 관한 연구

(A Study on the Development of Instructional Materials for Systematic "Programming" by Realization of the Mathematical Program)

박 광 철* 김 종 훈**
(Gwang-Cheol Park) (Jong-Hoon Kim)

요 약

지식 기반 사회를 맞이하여 컴퓨터 교육에 대한 관심이 점점 높아가고 있다. 컴퓨터 교육에서 프로그래밍 교육은 매우 중요하다. 그러나, 기존의 프로그래밍 교육은 프로그래밍 언어의 문법에 대한 기계적 암기나 사용법을 익히는 것에 치중해 왔다. 그래서 그 프로그램이 갖는 의미와 어떻게 다른 방법으로 구현될 수 있는지 등의 논리적인 사고를 키우는데 적합하지 못했다. 이런 프로그래밍 교육의 현실을 개선하여 정보 소양을 기르고 문제 해결 능력의 함양이라는 프로그래밍 교육의 목표를 달성하기 위해서는 프로그래밍 교육의 접근 방법에 대한 개선이 필수적이다. 따라서 본 논문에서는 프로그래밍 학습에 대한 새로운 접근법으로 프로그래밍 언어의 기초가 되는 C언어를 통해 프로그래밍에 대한 기본적인 통찰력과 테크닉을 기를 수 있도록 수학 프로그래밍 소스들을 구현하고자 한다. 이는 프로그래밍 학습에 대한 흥미와 관심이 증대되며 프로그래밍에 대한 경험을 쌓는 과정에서 문제 해결력 및 논리적 사고력의 향상과 프로그래밍에 대한 감각을 키워주는 좋은 경험이 될 것이다.

ABSTRACT

There is an increasing concern about computer education with the age of knowledge-based society. The learning programming language is taking an important role of computer education. However, the special emphasis in learning programming language has been attached to memorizing the programming language by rote and learning computer programs. Therefore, those were not much useful tools to develope a logical intelligence of the meanings of programming language and the methods of realization. It is positively necessary to improve the programming education efficiently because of the objects of knowledge of computing and raising an efficiency of problem solving.

Under the circumstances, this research is aimed at representing an useful education model through developing a mathmatical program into each part of the C programming language, which would be a new supplier of an basic insight into the programming language and techniques.

Accordingly it is thought that the research material will be an useful model to increase interests and concerns as well as to raise an efficiency of problem solving or a logical intelligence going through the process of studying programming language.

* 정회원 : 제주교육대학교 초등컴퓨터교육과 석사과정
** 정회원 : 제주교육대학교 컴퓨터교육과 조교수

논문접수 : 2001. 11 1.
심사완료 : 2001. 11. 10.

1. 서론

현대 사회는 정보의 가치나 역할이 어느 때보다 중요시되는 정보화 사회이다. 컴퓨터는 산업의 모든 분야에서 폭넓게 이용되고 있으며 과학 기술 발전의 중추적인 역할을 담당하고 있다. 이에 따라 컴퓨터 교육의 중요성이 강조되고 있으며 이 중 가장 중요한 분야가 바로 프로그래밍 분야이다.

프로그래밍 언어는 사용자와 컴퓨터간의 대화 수준으로 컴퓨터에서 매우 중요한 역할을 담당하고 있다. 그 중에 C언어는 프로그램 개발에 최적의 효율성을 제공하고 다른 언어로의 전환이 용이하는 등 다른 컴퓨터 언어에 비하여 많은 장점을 가지고 있어 많은 사람들이 C언어로 프로그래밍을 하고 있는데, 이러한 특성들이 전문 프로그래머에게는 다양한 표현방식을 제공하여 편리하겠지만 초보자의 경우에는 반대로 모든 특성을 이해하는데 많은 혼란을 가져올 수 있는 요소가 많다. 이에, 단순한 문법의 암기나 사용법을 익히는 것에 치중해 온 기존의 프로그래밍 학습 방법을 개선하고 정보 소양 및 문제해결 능력의 함양이라는 프로그래밍 교육의 목표를 달성하기 위해서는 프로그래밍 교육의 접근 방법에 대한 개선이 필수적이다.

프로그래밍 학습이 문법에 대한 기계적 암기나 프로그래밍 언어의 사용법을 익히는데 치중한다면 이는 학습자의 인지부담이 크며, 논리적 사고력을 기르는데 적합한 방법이 되지 못한다. 그래서 효과적인 프로그래밍 학습이 실시되기 위해서는 학습의 주체인 학습자에게 흥미와 내적 동기를 부여하고 학습자의 수준이나 관심을 고려한 학습방법을 제공하는 데에 대한 연구가 필요한 것이다[1].

따라서, 본 논문에서는 프로그래밍 학습에 대한 새로운 접근법으로 프로그래밍 언어의 기초가 되는 C 언어를 통해 코딩 실력을 강화하고 프로그래밍에 대한 기본적인 통찰력과 테크닉을 기를 수 있도록 수학 프로그래밍 소스들을 구현해 보고자 한다. 단순한 소스의 설명이 아닌 예제와 설명을 통한 접근 방법을 사용할 것이다. 수학 프로그램 소스 구현을 통해 프로그래밍 학습에 대한 관심을 증대시킬 수 있다.

본 논문은 프로그래밍 학습과 밀접한 관련이 있는 수학 문제를 프로그래밍 함으로써 학습자에게 프로그래밍 학습에 대한 관심을 증대시키며 프로그래밍 경험을 쌓기 위한 도움을 주는 데 더 큰 의의를 둔다. 이에 학습자들이 학습에 쉽게 이용하여 컴퓨터 알고리즘을 배우고, 나아가서 학습의욕의 고취는 물론 흥미를 일으킬 수 있는 교수 자료를 개발하는 데 목적을 둔다.

본 논문의 구성은 다음과 같다. 2장에서는 이론적 배경 및 관련 연구에 대하여 살펴보고, 3장에서는 수학 프로그래밍 구현 방향에 대해서 논한다. 4장에는 수학 프로그램 구현과정을 구조도와 소스 설명으로 보이고 마지막으로 5장에서는 결론 및 연구과제를 논의한다.

2. 이론적 배경

2.1 C 언어의 특징

C는 인터프리터 언어와 컴파일 언어 중에 컴파일 언어에 속하는 강력한 기능의 프로그래밍 언어이다. 이러한 C 언어는 구조화 프로그램을 지향하고 프로그램의 속도가 어셈블리어의 속도만큼 빠르기 때문에 빠른 속도를 요구하는 프로그램에 적합하다. 그리고 Unix운영체제, WindowNT, Window 95등 각종 시스템 프로그램에 널리 사용될 만큼 보편성을 지닌 유용한 언어이다[8,11,12].

C 언어의 특징으로는 호환성, 모듈성, 다양성, 동적 메모리 관리, 간결하고 함축적인 구조, 다양한 연산자, 구조적 언어 등이 있다. 이를 정리해서 C 언어의 장점을 요약하면 다음과 같다[10,12].

첫째, C 언어는 이식성이 좋다. 이것은 한 컴퓨터에서 작성된 코드를 다른 컴퓨터로 쉽게 옮길 수 있다는 것을 의미한다.

둘째, C 언어는 간결하다. C 언어는 강력한 연산자 집합을 가지고 있으며, 간접지정과 주소연산을 수식 내에 함께 사용할 수 있기 때문에 다른 언어로는 여러 문장이 필요한 것을 한 문장이나 수식으로 표현할 수 있다.

셋째, C 언어는 모듈성이 있다. C 언어는 값에 의한 호출로 인자를 전달하는 한 가지 방식의 외부 함수만을 제공하고 함수들의 중첩은 허용하지 않는다. 이러한 기능들은 운영체제가 제공하는 툴들과 함께 사용자 정의 라이브러리 함수와 모듈화 프로그

래밍을 쉽게 지원한다.

또한, 최근 정보교육의 동향을 살펴볼 때, C 언어가 강조되고 있음을 손쉽게 알 수 있다. 프로그래밍 수업에 있어서 다른 언어보다도 C 언어를 사용하는 경향이 증가되고 있다. 이는 C 언어를 이용해서 프로그래밍 연습을 했을 경우, 프로그래밍의 기본적 지식을 알게 되어 다른 언어를 이용해서도 프로그래밍 하기가 쉽기 때문이다. 그리고 요즘 강조되고 있는 리눅스 운영체제에서도 별 무리 없이 사용할 수 있다는 장점을 지닌다.

2.2 프로그래밍의 교육적 의의

프로그래밍 학습의 매력은 학생들에게 폭넓은 컴퓨터 수양을 심어준다는 것이다. 학생들은 프로그래밍 경험을 통하여 컴퓨터의 처리 과정에 대한 추상적인 개념을 확립할 수 있게 된다. 그리고, 컴퓨터 프로그래밍 학습은 이 시대의 보편적인 지적인 능력들, 이를테면 구조적으로 문제를 보는 능력, 형식적인 사고력을 길러주며, 비판적 사고력과 문제 해결력, 판단력을 발전시키는 데 목적이 있다[1].

교육적인 관점에서 볼 때 컴퓨터 프로그래밍은 두 가지 측면에서 의미를 지니고 있다[4].

첫째로, 프로그래밍의 오류수정 활동을 통해 스스로의 사고력을 향상시킬 수 있다는 점을 들 수 있다. 오류수정은 프로그래밍의 목표를 달성하는 데 있어서 생긴 여러 가지 잘못들을 하나하나 처리해감으로써 실수 없이 잘 수행되는 프로그램을 만들어 가는 과정을 의미한다.

둘째로, 교과서의 문제들을 컴퓨터 프로그래밍을 이용하여 풀 수 있는 능력을 갖는 것은 점차 도래하게 될 새로운 정보화 사회에 대비하는 지름길이며 자기 자신의 행동을 반영하여 그 결과를 프로그래밍하고 그것을 수정 보완하며 더 발전된 것으로 생각하는 것이다.

2.3 프로그래밍 교수 접근 방법

프로그래밍 교수(teaching)에 있어서의 접근 방법은 다음 세 가지로 요약하여 볼 수 있다[2].

2.3.1 구조지향적 접근

전통적인 교수방법으로써, 컴퓨터 언어의 명령어와 프로그램 자체에 비중을 두고 있는 방법이다. 이 교수방법은 프로그래밍 기술을 프로그램의 구조에 비추어 기계적인 것으로 보고 있기 때문에 인지적 기술의 향상을 노릴 수는 없다.

2.3.2 과업지향적 접근

이 교수방법은 창의성 및 분석적·연역적·귀납적 사고를 길러줄 수 있을 것으로 본다. 즉, 프로그래밍은 언어의 특성을 배우고, 그 지식을 사용하여 코딩하며, 여기에서 발생할 수 있는 오류를 정정하고, 테스트하는 연속적인 활동으로 본다.

2.3.3 인지지향적 접근

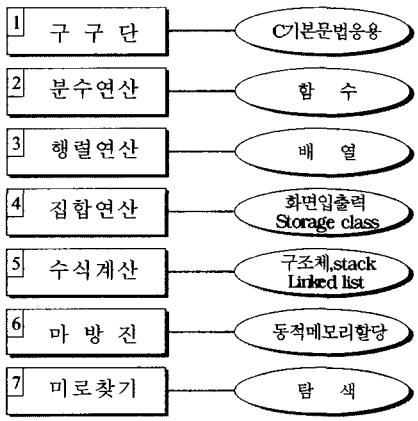
이 교수방법에서는 프로그래밍을 문제 해결의 기술이 핵심을 이루는 인지적 성취로 보고 있다. 즉, 내면화된 문제 해결의 모델을 새로운 문제에 적용시켜 그 유사점 및 차이점을 발견하려고 노력한다. 인지지향적 교수방법에 있어서는 특히 절차적 기술을 강조하는 데, 절차적 기술이란 문제 해결의 계획, 테스트 및 템플릿들을 결합하는 기술을 말한다. 인지지향적 교수 방법은 문제 해결의 능력을 일반화 시키는 데 초점을 맞추고 있다.

3. 수학 프로그래밍 구현 방향

프로그래밍 학습은 주로 문법을 숙지하고 사용법을 익힌 후, 다른 사람이 제작한 소스를 분석하고 수정하는 과정을 통해 이를 자신의 코드로 만들어 가는 과정으로 요약된다. 이것은 결코 단순 독해나 암기를 의미하는 것이 아니다. 그러한 단계를 뛰어넘어 적극적인 학습, 즉 문제를 해결하기 위한 소스 구현 과정이라고 할 수 있다. 그러한 과정에 수학 프로그램이라는 우리가 쉽게 접할 수 있는 문제들을 구현의 대상으로 선정함으로서 학습자의 관심과 흥미를 배가시킬 뿐만 아니라 프로그래밍에 대한 경험을 쌓는 과정을 통해 하드웨어에 대한 추상적인 개

념을 이해하고 자신이 해결하고자 하는 문제에 대한 해결력을 기르게 될 것이다[1].

본 논문에서 구현하고자 하는 수학 프로그래밍 소스는 7가지이다. [그림 1]은 수학 프로그래밍의 목록과 각 프로그래밍 구현을 통해 익혀야 할 핵심 요소들을 보여준다. 우선 C에 대한 기본적인 문법을 응용하는 데서 시작하여 프로그래밍에 꼭 필요한 기본 자료구조 및 알고리즘 등을 수학 프로그램을 구현함으로서 익히고자 한다. 이것은 자연스럽게 프로그래밍에 대한 개념과 절차, 방법이 습득될 수 있는 새로운 방안으로서 제시하고자 한 것이다.



[그림 1] 각 프로그램들의 목록과 익혀야 할 핵심 요소

[Fig. 1] Lists of each program and key points

수학 프로그래밍 소스들의 구현 원칙은 다음과 같다.

첫째, 소스의 길이가 길지 않아야 한다. 소스의 길이를 줄이고 필요한 부분만 제시한다.

둘째, 각 소스마다 익혀야 할 코드를 명확히 한다. 각 단계에서 익혀야 할 사항이 수학 프로그램 속에 명확히 구현되도록 한다.

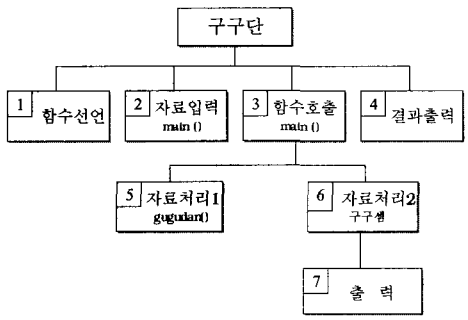
셋째, 개작을 위해 많은 요소를 남겨둔다. 프로그래밍 학습에 있어서 남의 것을 자신의 것으로 만드는 과정이 중요시된다. 따라서 타인의 소스를 자신의 코드를 만들기 위해 수정, 보완 및 다른 방법으로 개작할 수 있는 여지를 남겨두도록 한다.

4. 수학프로그램의 구현

4.1 구구단

이 프로그램은 C문법의 기본이 되는 데이터형, 제어문, 연산자 학습을 위해 제시되었다. 기본적인 문법, 명령어를 배우고 이를 조합하여 실제 프로그래밍에 응용하는 단계에서 처음으로 구현해 보는 소스로 적합할 것이다.

구구단 프로그램의 구조도는 [그림 2]와 같다.



[그림 2] 구구단 프로그램 구조도

[Fig. 2] Gugudan program structural diagram

[그림 2]에서 2,5,6을 구현하기 위해서는 제어구조를 이용하면 편리하다.

제어구조

제어구조는 알기 쉬운 프로그램을 만들기 위한 프로그램 기법으로 프로그램을 만들 때에 단순화한 구조로 알기 쉽고, 동시에 일정하게 형식화된 구조의 조합으로 프로그램을 표현하는 것을 말한다.

구구단 프로그램 구현 시 제어구조별 구현내용 및 형식은 다음 <표 1>과 같다.

<표 1> 제어구조별 구현내용

<Table 1> Realized contents by control structure

제어 구조	조 건	구현내용	구 현 형 식
while	조건식에 따라 실행문을 반복 처리	구구단 종류 원하는 구구단의 숫자 입력	while (1) { 구구단의 종류, 원하는 숫자 입력 ; break ; }

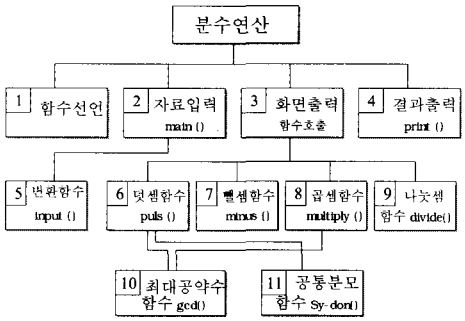
switch	키보드 명령 처리 시	구구단의 선택	switch(num) {
			case -3 : /*구구단 전체*/
			start=1;end=5;step=1;
			case -1 : /* 홀수단 */
			case -2 : /* 짝수단 */
			default : return }

for	일정한 횟수 반복 시	구구셈에 이용	for(i=1 ; i<=9 ; i++) {
			for(j=start;j<=end;j+=step)
			출력형식, 구구단 출력; }

4.2 분수연산

이 프로그램은 두 분수의 분자와 분모를 따로 입력받아 두 분수의 사칙연산을 수행하는 소스로, 함수의 선언 및 사용에 관한 학습을 정리해 주는 역할을 하고 있다.

분수연산 프로그램의 구조도는 [그림3]과 같다.



[그림 3] 분수연산 프로그램 구조도

[Fig. 3] Fraction calculation program structural diagram

[그림 3]에서 3을 통해 각 함수의 호출이 이루어지는데, 이를 구현하기 위해서는 함수를 이용하면 편리하다.

함수란 프로그램 내에서 어떤 특정한 작업을 전담하여 수행할 수 있도록 만들어진 하나의 단위, 즉 규격화된 서부루틴이다. 따라서 커다란 프로그램을 작업별로 분할하여 제작할 수 있기 때문에 체계적이고 간결한 구성을 이룰 수 있으며, 프로그램을 이해하고 수정하기가 용이하게 해 준다.

swap()함수

swap()함수는 매크로 함수로 매개변수로 받은 두 수 a,b중 만약 b가 더 크면 두 수를 바꾸어 항상 a>b인 상태로 만든다. 이 함수는 비트 연산인 XOR을 이용한 것으로 프로그램 중간에 끼어 들어가므로 실행속도가 빠르다.

```
#define swap(a,b) {(a)^=(b);(b)^=(a);(a)^=(b);}
                                /* 매크로 함수 */
  if(b>a) swap(a,b) ;  /* a>b가 되게 한다 */
```

gcd()함수

gcd()함수는 최대공약수를 구하는 함수로, 최대공약수를 구하는 원리는 다음과 같다.

$$n_0 = \max(|a|, |b|)$$
$$n_1 = \min(|a|, |b|)$$
$$n_k = n_{k-2} - [\, n_{k-2} / n_{k-1} \,] * n_{k-1}$$
$$k = 2, 3, \cdots$$

만약 $n_k = 0$ 이면 최대공약수는 n_{k-1}이 된다.

a,b의 절대값중에서 큰 수를 n_0에, 작은 수를 n_1에 대입한다. 그리고, k를 2에서 하나씩 증가시키면서 n_k가 0이 되면 n_{k-1}이 최대공약수가 된다. 만약 n_k가 0이 되지 않고 음수가 되면 최대공약수는 1외엔 없는 것이다.

```
unsigned gcd(unsigned a, unsigned b) {
  int j ;
  while(1) {
   j=a-(unsigned)(a/b)*b;
   if(j==0) return b;  /* j==0이면 b가 G.C.D*/
    else if(j<0) return 1; /*j<0, G.C.D은 1뿐*/
    else { a=b ; b=j; }
```

함수 호출

함수 호출시 인수 값을 전달하는 방식에 따라 값에 의한 호출(call by value)방식과 참조에 의한 호출(call by reference)방식으로 나누는 데, 참조에 의한 호출 방식은 포인터를 이용해 실인수의 값을 바꿀 수 있는 호출방식으로, 리턴값에 의해 결과를 넘기는 것이 아니라 바로 실인수의 값을 변경한 것이 결과가 된다.

값에 의한 호출 경우를 보면, 분수의 사칙연산에서 분수의 나눗셈의 경우 피제수의 분자와 분모를 서로 바꿔주면 분수의 곱셈이 되므로, 피제수의 분자와 분모를 바꿔 준 후, 곱셈 연산을 행한다.

```
fraction divide(fraction a, frection b) /*곱셈함수*/
    피제수의 분자 분모 교환 ;
    return multiply(a,b) ;  /*곱셈 연산*/
```

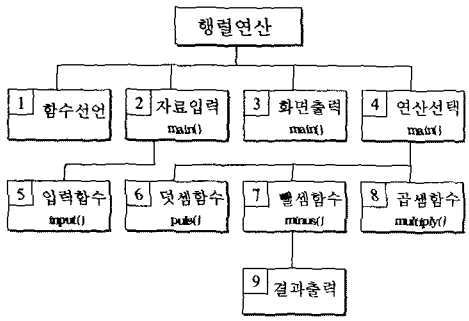
반면, 참조에 의한 호출 경우를 보면, 분수의 덧셈의 경우 먼저 공통 분모로 분모를 통분한 후 덧셈이 이루어져야 하므로, 공통 분모의 실인수 계산 결과 후 번지값으로 넘겨 받는다.

```
fraction plus(fraction a, frection b) /*덧셈함수*/
      sy_don(&a, &b) /*  */
분수에 각각의 부호를 적용해서 덧셈을 행함.;
 void sy_don(fraction *a, frection *b)
            /*공통분모를 취하는 함수*/
```

4.3 행렬 연산

이 소스는 두 행렬의 각 원소(첨자)를 입력받아 행렬의 연산 규칙에 의거 행렬의 덧셈, 뺄셈, 곱셈을 행하는 프로그램으로, C언어의 가장 핵심 요소 중 하나인 배열과 포인터에 관한 학습을 정리해 주는 역할을 한다.

행렬연산 프로그램의 구조도는 [그림 4]와 같다.



[그림 4] 행렬연산 프로그램 구조도

[Fig. 4] Matix calculation program structural diagram

main 함수의 구조

main 함수의 loop 구조는 for()문을 사용하여 각 행렬의 원소를 입력할 수 있도록 한다.

```
void main() {
  제어변수 선언 및 행렬의 크기 입력 ;
  for(;;) { /* 제어구조로 행렬의 행과 열 지정 */
       첫 번째 행렬의 각 원소 화면에 출력,
       자료 입력 함수 호출 ;
  for(;;) { /* 제어구조로 행렬의 행과 열 지정 */
       두 번째 행렬의 각 원소 화면에 출력,
       자료 입력 함수 호출 ;
   do {
      swith(sel) {
         case 1 : puls_operator() ; break;
         ..................................... }
      } while()
```

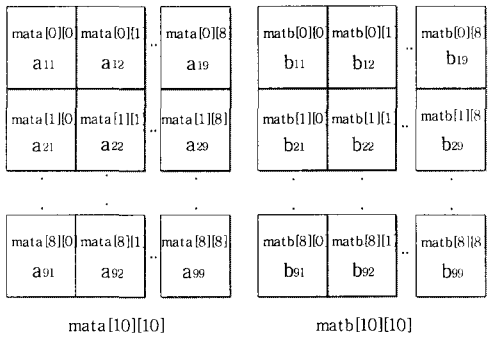
mat배열

배열은 다수의 동일한 자료형을 갖는 자료의 하나로서, 같은 종류의 값을 가지는 변수들을 여러 개 사용하여야 할 때 사용하면 아주 편리하다. 배열은 다음과 같은 4가지 특징을 가지고 있다. 첫째, C에

서 배열의 처음은 첨자가 0으로 시작된다. 둘째, 일반적으로 배열의 크기가 k로 정의할 때 최대 첨자는 k-1이다. 셋째, 배열의 크기가 k일 때 배열이 메모리 상에 차지하는 총 바이트 수는 sizeof*k이다. 넷째, 배열 요소를 변수로 사용해서 참조 가능하다.

mat배열은 행렬에서 행렬의 행과 열의 크기를 입력 받은 후 각 연산이 행해질 때 재구성하는 2차원 배열으로 행렬의 덧셈, 뺄셈, 곱셈 연산 결과도 이 배열에 저장된다.

2차원 배열 mata[10][10], matb[10][10]이 메모리 상에 구현되는 모양은 [그림 5]와 같다.



[그림 5] mata[10][10], matb[10][10]의 구현모양
[Fig. 5] Realized form of mata[10][10], matb[10][10]

포인터 연산

C언어의 가장 핵심 요소 중 하나인 포인터는 주소를 값으로 갖는 변수를 의미한다. C에서의 포인터는 메모리에 데이터가 저장되어 있는 곳을 가리키는 역할을 한다.

행렬 프로그램 구현 시 동적 메모리 할당 방식으로 각 행렬의 원소를 1차 배열로 입력받고 연산을 위해 2차원 배열로 재구성하기 해야 하는 데, 이는 포인터 연산을 이용하면 편리하다. 포인터 연산이란 피연산자 중의 하나가 포인터 변수인 연산을 말한다.

그 과정을 살펴보면 다음과 같다.

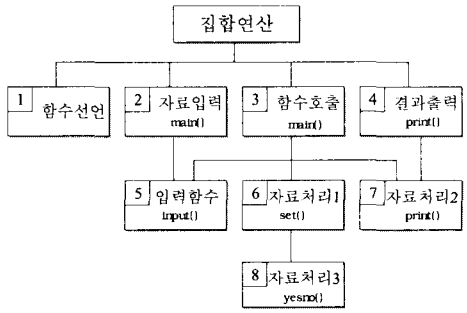
```
for( i = 0; i < row; i++) {
  for( j = 0; j < col; j++) {
     mata[i][j] = *(pt1++);
  /*pt1은 1차 배열로 입력받은 첫 번째 행렬*/
     matb[i][j] = *(pt2++);
  /*pt2은 1차 배열로 입력받은 두 번째 행렬*/
     행렬 연산, 연산 결과 출력; } }
```

여기서, *pt1++은 연산 순서에 따라 *pt1이 제일 먼저 연산이 된다. 즉, *pt1 연산에 의해 먼저 pt1이 가르키는 메모리의 내용을 출력한다. 그리고, pt1을 1 증가시킨다. *pt1++은 *pt1, pt++의 두 명령을 하나로 합친 것이다.

4.4 집합 연산

이 소스는 집합의 각 원소를 입력받아 연산을 수행하는 프로그램으로, 텍스트 모드의 문자열 함수에 관한 학습을 정리해 주는 역할을 하고 있다.

집합 연산 프로그램의 구조도는 [그림 6]과 같다.



[그림 6] 집합연산 프로그램 구조도
[Fig. 6] Set calculation program structural diagram

단일문자 입출력 및 문자열 입출력

처음 프로그래밍을 접하는 학습자에게 텍스트 모드에서의 입출력을 다루는 함수는 대수롭지 않게 여겨질 수 있다. 그러나 적절한 함수를 사용하지 못한 경우, 프로그램은 의도하지 않았던 결과를 출력하고 만다. 본 프로그램에서는 문자열에 관한 함수들을 다수 소개함으로서 그 차이점을 명확히 하고 올바른 사용법을 익히고자 한다. <표 2>와 <표 3>은 각 함수를 입출력 형태와 CR/LF변환, 문자의 속성 표현 여부를 통해 비교하고 있다.

<표 2> 단일문자 입출력 함수 비교
<Table 2> Unitary letter input/output function comparison

	함수	입출력 형태	Carriage Return 및 Line Feed 변환	문자의 속성 표현	원도우의 경계 영향
출력	getch	직접			
	getche	직접		한다	
	getchar	버퍼 경유			
입력	putch	직접	하지 않는다	한다	받는다
	putchar	직접	한다	하지 않는다	받지 않는다

<표 3> 문자열 입출력 함수 비교
<Table 3> Line of letter input/output function comparison

	함수	입출력 형태	Carriage Return 및 Line Feed 변환	문자의 속성 표현	원도우의 경계 영향
출력	gets				
	cgets				
입력	puts	DOS 호출	한다	하지 않는다	받지 않는다
	cputs	직접	하지 않는다	한다	받는다

<표 4> storage class의 특징
<Table 4> Distinction of storage class

	전역변수	지역변수	정적변수	레지스터 변수
선언 위치	함수의 외부	함수의 내부	함수의 내부	함수의 내부
통용 범위	프로그램 전체	함수의 내부	함수의 내부	함수의 내부
파괴 시기	프로그램 종료	함수의 종료	프로그램 종료	함수의 종료
저장 장소	정적 데이터 영역	스택	정적 데이터 영역	레지스터
초기값	0으로 초기화	초기화 않음	0으로 초기화	초기화 않음
키워드	extern	auto	static	register

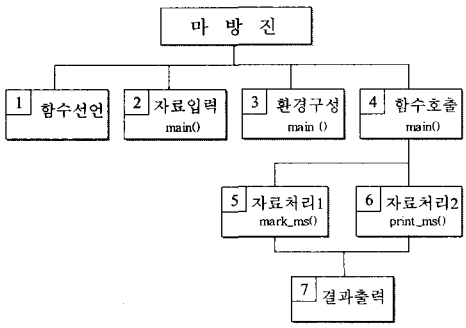
storage class

storage class는 변수의 형태와 크기를 결정하는 자료형과는 달리 변수를 어디에(메모리 또는 레지스터), 어떻게(생존기간과 유효범위) 기억시킬 것인가를 결정한다. <표 4>는 각 변수들을 선언위치, 통용범위, 저장장소 등으로 그 특징을 설명하고 있다.

4.5 마방진

마방진은 1에서부터 n*n까지의 숫자를 n*n 행렬 안에 하나씩 넣어 합, 세로 합, 대각선 합이 같도록 만든 것이다. C 프로그램이 컴퓨터의 메인 메모리에 정보를 저장할 수 있는 중요한 방법중 하나인 동적 할당 시스템이 소개된 프로그램이다.

마방진 프로그램의 구조도는 [그림 7]과 같다.



[그림 7] 마방진 프로그램 구조도
[Fig. 7] Magic square program structural diagram

마방진을 만드는 방법은 간단하다. 행의 수가 홀수일 때만 성립하는 것으로 제일 윗행의 가운데에 1을 위치하고 1씩 더해가면서 오른쪽 위의 위치에 놓여진 숫자가 없으면 숫자를 위치시키고, 있으면 현재 위치의 아래쪽에 숫자를 위치시키는 방법이다. 메모리 할당으로 인하여 모든 메모리가 0으로 이미 초기화되어 있어 0이면 숫자가 놓여지지 않은 것이고 그 외에는 숫자가 이미 놓여진 것이다. 그리고, 이동시킬 때 고려할 점은 제일 윗행은 제일 아래 행과 제일 오른쪽 열은 가장 왼쪽 열과 연결되어 있다고 생각해야 한다.

마방진을 만드는 방법은 다음과 같다.

```
for(num=2 ; num<=(line*line) ; num++)
  if(square[(crow-1+line)%line][(col+1)%line]==0)
      열이동 ; 행이동 ; 숫자지정;
      else 열이동 ; 숫자이동
```

동적메모리 할당

프로그램이 실행하기 위해서는 메모리가 필요하다. 프로그램 자체가 load될 메모리는 물론이고, 프로그램이 데이터를 처리하기 위해서 사용하는 메모리도 꼭 필요하게 되는데, 이 중 프로그램 실행 중에 요청에 의해 할당되는 메모리를 '동적할당 메모리'라고 하고, 보통 malloc(), calloc() 함수를 많이 사용한다.

함수 calloc()는 malloc 함수와 동작은 동일하되 다만 메모리 요구량을 전달해 주는 인수의 구성이 다르다. malloc는 바이트 단위로 메모리 요구량을 전달하는데 비해 calloc는 특정 단위의 특정 개수를 전달한다. 함수 calloc는 포인터를 할당메모리로 반송하며 할당된 메모리 용량은 num*size와 같다. 즉, calloc()는 size크기의 num대상들의 배열을 위한 충분한 메모리를 할당한다.

함수 calloc()는 할당 지역의 첫 번째 바이트로 포인터를 반송한다. 만약 요구에 충족되지 않는 메모리 용량이라면 Null포인터가 사용하기 전에 반송값이 Null포인터가 아닌지 미리 확인하는 것이 중요하다.

```
if((square=(int**)calloc(line,sizeof(int*)))==NULL){
        Can't allocate memory 출력 ; }
  for((cnt=0 ; cnt<line ; cnt++)
if((square[cnt]=(int*)calloc(line,sizeof(int)))==NULL){
      Can't allocate memory 출력 ; }
```

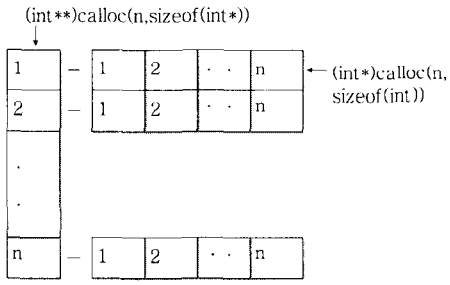
여기서 매개변수로 받은 것은 Square_type으로 정의된 변수의 주소값인데 Square_type의 포인터를 매겨변수로 받은 것은 참조에 의한 호출방식으로 변수를 입력받아 메모리를 할당하고 변수값을 되돌려 주기 위해서이다. 또, for문을 이용하여 2차원 포인터 메모리를 설정하였다. 두 개 이상의 포인터를 사용할 경우에는 loop문을 사용하여 각각의 메모리를 설정해 주어야 한다.

한편, calloc함수에 의해 할당된 메모리는 프로그램 종료 전에 반드시 free함수에 의해 해제하여 반납되어야 한다. 한 번 할당된 메모리는 절대로 다른 목적에 다시 사용될 수 없기 때문에 할당만 해두고 반납을 하지 않으면 시스템 전체의 사용 가능한 메모리 공간이 줄어드는 결과를 초래하게 된다.

메모리 해제는 다음과 같이 for loop를 돌려서 행한다.

```
for(cnt=0;cnt<line;cnt++)
        free(square[cnt]);
 free(square)
```

이 메모리 할당을 그림으로 나타내면 [그림 8]과 같다.



[그림 8] 동적 메모리 할당

[Fig. 8] Dynamic memory assignment

4.6 수식계산

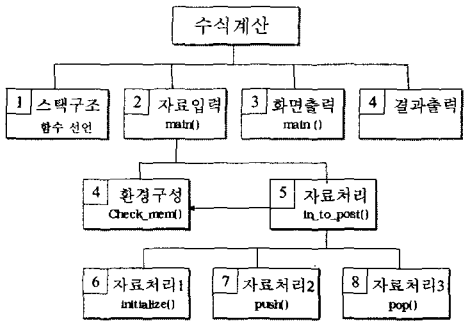
수식계산기는 보통 일상적으로 사용하는 수식 중 위표기(infix)형태의 수식을 컴퓨터에서 사용하는 후위표기(postfix)형식으로 변환하는 프로그램이다. 중위표기식을 후위표기식으로 변환하기 위해서는 스택(stack)을 사용해야 하며 스택에는 연산자들이 저장된다. 이는 연결리스트를 이용하여 쉽게 구현할 수 있다.

스택

스택은 큐와 같이 순서화된 자료구조 중의 하나로 나중에 들어온 데이터가 먼저 나오게 되어있는

구조로 되어있다. 큐와는 달리 스택에서는 삽입과 삭제가 한 군데에서 이루어진다. 삽입과 삭제가 이루어지는 곳을 일반적으로 top라고 한다. top은 변수가 항상 스택의 맨 위의 위치를 가리킨다. 데이터가 하나 추가되면 top의 위치에 데이터를 저장하고 top의 값을 하나 감소시킨다. 데이터가 삭제될 때에는 top의 값을 감소시키고 그 top의 위치에 데이터를 리턴한다. 그러면 top은 다음에 들어올 데이터가 들어가게 될 위치를 가리키게 된다. 초기상태의 스택은 값이 없고 top는 새로 들어올 데이터가 저장될 위치를 나타내고 있다. top의 값이 스택을 초과하면 더 이상 스택은 데이터를 추가시킬 수 없게 된다.

수식계산 프로그램의 구조도는 [그림 9]와 같다.



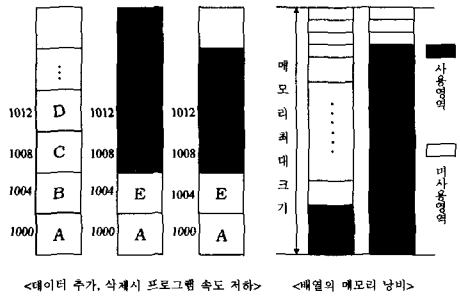
[그림 9] 수식 계산 프로그램 구조도

[Fig. 9] Numerical formula calculation program structural diagram

선형연결리스트를 이용한 스택구조의 구현

구조체라는 개념을 접하면서 이를 직접 실제 프로그래밍에 응용해 보는 단계로 제시할 수 있는 것은 선형연결리스트등의 자료구조가 일반적일 것이다.

선형연결리스트는 보통 배열과 비교하여 설명된다. 배열 역시 기본적으로 어떤 데이터들의 리스트라고 할 수 있는데 특별히 다음 데이터의 위치가 전 데이터의 위치에서 항상 일정한 거리만큼 떨어진 곳에 위치한다는 점에서 따로 구분한다. 학습자에게 추상적인 메모리의 개념을 보다 직관적으로 설명할 수 있는 것이다. 때문에 배열은 C에 대한 학습에서 포인터를 다루는 항목보다 항상 앞서 다루어진다. 그러나 실제 작업을 수행할 때는 항상 일정한 양의 메모리가 필요한 것이 아니라 필요한 만큼의 메모리만을 사용하는 것이 효율적일 수 있다. 이러한 생각에 메모리를 효율적으로 사용하기 위한 자료구조 중의 하나가 선형연결리스트인 것이다.



[그림 10] 배열을 사용할 때의 문제점

[Fig. 10] Problem of use of the array

[그림 10]은 배열의 문제점을 더욱 분명하게 보여주고 있다. 첫 번째 문제는 배열을 사용함으로서 프로그램의 실행속도가 저하되는 문제이다. 배열에서 새로운 데이터를 삽입하기 위해서는 기존의 데이터들을 차례로 밀어낸 다음 추가해야 한다. 만약 맨 처음의 위치에 데이터를 삽입하는 경우는 현재 존재하는 모든 데이터를 이동해야 하므로 프로그램의 속력은 그 만큼 늦어질 수밖에 없다. 데이터를 삭제하는 경우 역시 최악의 경우 맨 처음의 데이터가 삭제되면 다음의 모든 데이터는 삭제된 곳으로 차례로 이동해야 한다. 두 번째 문제는 배열을 사용하면 다양한 크기의 리스트를 관리할 때 메모리가 낭비된다는 사실이다. 그림에서 최대 크기를 할당하여 메모리를 사용하는 경우 사용하지 않는 영역의 메모리가 모두 낭비되는 곳이라고 볼 수 있다. 어떤 때는 메모리를 더 많이 사용하고 어떤 때는 메모리를 적게 사용하는 것이 효율적인 메모리 관리이다.

선형연결리스트는 하나의 데이터를 표현할 때 데이터 외에 다음 데이터의 위치를 함께 저장하게 된다. 배열에서는 기본적으로 다음 데이터는 자신의 바로 다음 위치에 오기 때문에 이것이 암묵적으로 파악되지만 선형연결리스트에는 다음 데이터가 메모리 상의 어떠한 곳에도 올 수 있기 때문에 다음 데이터를 알아내기 위해 그 위치를 함께 저장한다. 따

라서 선형연결리스트의 데이터는 두 가지 부분으로 이루어져 있다. 이러한 부분을 필드라고 하며 데이터를 저장하는 부분을 데이터 필드, 다음 데이터의 주소를 저장하는 부분을 링크 필드라고 한다.

선형연결리스트를 사용하여 스택을 구현하도록 한다.

```
typedef struct linked_list { /*연결리스트 구조*/
        data d;              /*데이터 필드*/
        struct linked_list *next; /*링크 필드 */
         } link_node;

typedef link_node *link;

typedef struct stack { /*스택 구조 구현*/
         int cnt;
         link top;
      }; stack
```

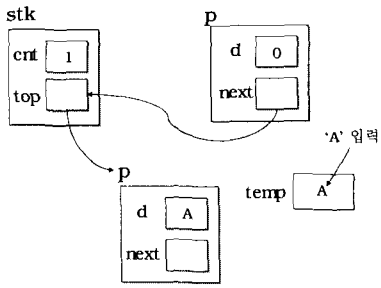
실제 stack의 동작은 다음의 함수를 통해 이루어진다.

```
void initialize(stack *stk)  /*스택의 초기화 */
  {
   stk->cnt = 0;
   stk->top = NULL;
  }

void push(data d, Stack *stk) { /*데이터 삽입 */
   link p;
  새로운 리스트 P를 생성, 메모리 할당;
  p->d = temp; p->next = stk->top;
  stk->top = p;  stk->cnt++;
  }

void pop(stack *stk, data d) { /*데이터 리턴 */
   link p;
   p = stk->top; stk->top = stk->top->next;
   stk->cnt--;
   free(p);
 }
```

struck linked_list와 struck stack형의 생성과 스택에 데이터가 삽입되는 과정을 그림으로 나타내면 [그림 11]과 같다.

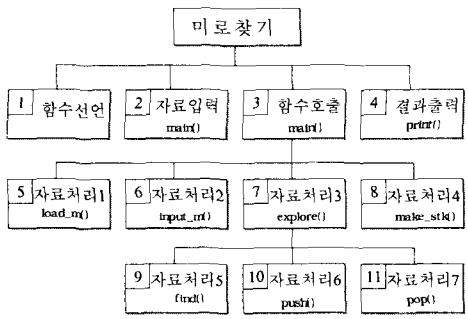


[그림 11] 구조체 생성과 스택에 데이터가 삽입되는 과정
[Fig. 11] Formation of structure and process push data into stack

4.7 미로찾기

미로찾기는 어린이의 지능 계발을 목적으로 입구에서 출구에 이르는 길이 하나밖에 없게 하되 출구를 쉽게 찾을 수 없도록 길을 복잡하게 그려 목적지를 찾아가는 프로그램으로 프로그래밍에 꼭 필요한 기본 자료구조 및 알고리즘 등을 익히는 데 유용하다.

미로찾기 프로그램의 구조도는 [그림 12]와 같다.



[그림 12] 미로찾기 프로그램 구조도
[Fig. 12] Maze program structural diagram

구조체

이 소스는 미로의 지도 및 크기, 입구와 출구, 탐색방법을 하나의 구조체로 묶어 미로찾기 프로그램을 구현한 것이다. 미로 열, 행의 크기, 입구와 출구의 행, 열 좌표 등을 지정할 때 배열을 사용하는 것보다는 데이터들이 논리적, 유기적인 관계를 갖고 있는 하나의 데이터 그룹이므로 구조체로 묶어 놓으면 관리하기도 편리하다. 미로 구조체의 형태는 다음과 같다.

```
struct maze_info {   /* 미로의 지도 및 크기,
입구와 출구, 탐색방법 등을 수록할 구조체 형틀*/
      int nrow;     /* 미로의 열 크기 */
      int ncol;     /* 미로의 행 크기 */
      char *maze; /*미로를 저장할 동적메모리의
                    주소에 대한 포인터 */
      int srow;     /* 입구의 열 좌표 */
      int scol;     /* 입구의 행 좌표 */
      int erow;     /* 출구의 열 좌표 */
      int ecol;     /* 출구의 행 좌표 */
      int view_all; /* 탐색 방법 */
   }
```

[그림 12]에서 ⑦을 구현하기 위해서는 자료구조인 탐색을 이용하면 된다.

탐색

탐색이란 컴퓨터의 기억 공간에 보관된 자료들 중에서 어떠한 성질을 만족하는 것들을 찾아내는 일을 의미한다. 이는 컴퓨터로 처리하는 많은 일들이 필요로 하는 작업의 한 형태이며, 또 컴퓨터의 시간을 많이 소요하는 작업의 하나이므로 효율적인 탐색 방법을 찾는 것이 중요하다.

Best-First Search는 너비우선탐색(Breadth-Fisrt Search)를 행함에 있어 각 Level별로 깊이우선탐색(Depth-First Search)를 행하므로서 optimal solution을 보장하는 탐색기법이다. 이 프로그램에서는 optimal solution이 하나 이상일 경우를 감안해 출구에 도달한 이후로도 optimal solution이 발견된 Level에 대한 탐색을 끝까지 마쳐 가능한 모든 optimal solution을 구할 수 있도록 Best-First Search를 응용하였다.

탐색은 함수의 재귀적 호출을 통해 이루어지며 탐색 자체는 재귀함수 사용 시 지원되는 시스템 스택을 사용한다. 또한 이미 지나온 지점을 중복하여 지나가는 일을 막기 위해 별도의 스택을 이용하여 현재 거쳐온 경로를 별도로 지니도록 하였다. 이 스택의 내용들과 앞으로 이동할 좌표들을 비교하므로서 경로의 중복을 막았다.

미로를 탐색하는 함수는 다음과 같이 구현할 수 있다.

```
int explore(struct maze_info*m, int row, int col,
        int dep, int cdep)  / * 미로 탐색 함수 */
        /* maze_info*m - 미로의 정보,
           int row - 현재의 열,
           int col - 현재의 행 좌표,
           int dep - 최대 탐색 level,
           int cdep - 탐색 중 level */
  if(m->viem_all==-1)return(retval=0);
        /*Optimal Soulution을 찾았음을 알림*/
   현재의 위치를 스택에 저장 ;
   if(m->view_all==-1&&cdep>dep) return 0 ;
```

5. 결론

프로그래밍 학습에 대한 새로운 시도로서 본 논문은 프로그래밍 언어의 기초가 되는 C언어를 통해 여러 가지 수학 문제 프로그램을 구현함으로서 프로그래밍에 대한 기본적인 통찰력과 테크닉을 기르는 등 프로그래밍 학습에 도움이 되고자 하였다. 학습자가 프로그래밍 언어에 대한 문법을 숙지하고 사용법을 익힌 후, 실생활과 관련된 수학 문제를 제시하여 이를 프로그래밍 구현하는 과정에서 얻을 수 있는 효과는 다음과 같다.

첫째, 수학 프로그램 소스 구현을 통해 프로그래밍 학습에 대한 관심을 증대시킬 수 있다. 효과적인 프로그래밍 학습에 있어 학습자의 관심과 흥미는 대단히 중요하다. 이를 위해 프로그래밍 학습과 밀접한 관련이 있는 수학 문제 소스를 구현의 대상으로 선정하였다.

둘째, 교재에서 제공하는 짧고 단순한 기본 예제를 탈피하여 프로그래밍에 대한 진지한 경험을 가질 수 있다. 교재에 제시되어 있는 레퍼런스의 함수 하나를 익히고 이에 대한 짧은 사용법을 단순히 코딩하고 암기하기보다는 이들을 응용하여 적절한 코딩할 필요가 있는 것이다.

셋째, 프로그래밍에 대한 경험을 쌓는 과정에서 문제 해결력 및 논리적 사고력, 판단력의 신장에 많은 도움이 될 것이다.

셋째, 남으로부터 주어지는 학습이 아닌 자신이 주도해 나가는 학습을 할 수 있다. 수학만의 특성으로 남과는 다른 자신만의 관점으로 프로그래밍 할 수 있다.

넷째, 수를 주로 다루기 때문에 쉽게 프로그래밍 할 수 있다. 수학 프로그램에서 다루는 것의 대부분이 수의 연산이기 때문에 기초적 지식을 가지고 누구나 쉽게 프로그래밍 할 수 있다.

프로그래밍이란 목적하는 바를 이루기 위해 명령어들을 적절히 조합해 내는 기술이라고 할 수 있다. 이를 위해 가장 중요한 것은 프로그래밍에 대한 경험을 많이 쌓는 것이라 할 수 있다. 자신이 직접 프로그램을 만들어서 쌓는 경험만큼 남이 만들어 놓은 프로그램을 분석하는 것도 프로그래밍에 대한 감각을 키워주는 좋은 경험이 될 것이다.

※ 참고문헌

[1] 박원길, 이재무, "아동과 초보자를 위한 프로그래밍학습 시스템의 설계", 한국정보교육학회 학술발표논문집, 2000.

[2] 백영균, "컴퓨터 프로그래밍에 대한 심리학적 접근", 교육공학연구 제4권 제1호, 1988.

[3] 이옥선, "웹 브라우저 상에서 수행되는 Linux 기반 C언어 프로그래밍 실습시스템", 상명대학교 정보통신대학원 석사학위논문, 1999.

[4] 주영숙, "C언어 교육용 도구의 설계 및 구현", 동국대학교 교육대학원 석사학위논문, 1993.

[5] 차시홍, "하이퍼텍스트 기법을 이용한 C언어 학습법 적용에 관한 연구", 관동대학교 교육대학원 석사학위논문, 1997.

[6] 김상형, C++을 내것으로, 가남사, 1995.

[7] 민승재, 대학생을 위한 C Project, 한컴프레스, 1997.

[8] 이재규 지음, C로 배우는 알고리즘, 도서출판 세화, 2000.

[9] 이재환, C 월리를 찾아라, 가메출판사. 1999.

[10] ERIC S. ROBERTS 저/김정선 · 도경구 · 박성주 · 오희국 · 최중민 역, C언어와 프로그래밍, (주)교보문고, 1998.

[11] Herbert Schildt 저 / 성평식 역, 완벽 C 언어 해설, 도서출판 세화, 1989.

[12] HOROWITZ · SAHNI · ANDERSON-FREED원저, 이석호옮김, C로 쓴 자료 구조론, 희중당, 1996년.

[13] B.W.Kernighan, *The C programming Language*, Second Edition,Prentice-Hall, 1988.

[14] C. L. Tondo and S. E. Glmpel, The C Answer Book, Second Edition, Prentice-Hall, 1989.

[15] Ken Kahn(1999). *A Computer Game to Teach Programming*. National Educational Computing Conference, 1999.

[16] P. J. Plauger *The Standard C Library*, Prentice Hall, Inc.., 1992.

[17] http://www.winapi.co.kr

박 광 철

1987년 ~ 1991년
제주교육대학교 수학교육과
학사
2000년 ~ 현재
제주교육대학교
초등컴퓨터교육과
석사과정
관심분야 : 프로그래밍 언어,
컴퓨터 교육

김 종 훈

홍익대학교 전자계산학과
이학박사
제주교육대학교 컴퓨터교육과
조교수
과학영재교육센터 초등정보반
지도교수
관심분야 : 컴퓨터 영재 교육,
컴퓨터 교육