

전산개론 및 실습 2

프로그래밍 언어: FORTRAN

2008. 08

이 민 형 교수
기계항공우주공학부

Contents

Chapter 1. 포트란의 개요와 형식

1-1. 포트란 언어의 역사	1
1-2. 포트란 언어의 실행 단계	1
1-3. 문제해결의 순서	2
1-4. Overview of Fortran	2
1-5. Use of Columns in Fortran	3
1-6. INTEGER, REAL, and CHARACTER Data Types	4
1-7. Arithmetic Expressions	4
1-8. 포트란에서의 기초적인 명령어	5
1-9. 포트란에서 사용되는 함수	5
1-10. 공통된 프로그램 오류	7
1-11. 응용 예제	8
1-12. 실습 예제	11

Chapter 2. 제어문(Control Statement)

2-1. 논리 표현	13
2-2. 조건문(IF)	14
2-3. 반복문(DO)	16
2-4. 응용 예제	19
2-5. 실습 예제	22

Chapter 3. 서브프로그램(Subprogram)

3-1. 서브프로그램의 종류	24
3-2. 내장함수	24
3-3. 문함수	24
3-4. 함수 서브프로그램	25
3-5. 서브루틴 서브프로그램	26
3-6. 서브루틴 서브프로그램과 함수 서브프로그램의 차이점	28
3-7. 응용 예제	29
3-8. 실습 예제	32

Chapter 4. 배열(Array)

4-1. 배열 선언	34
4-2. 배열 첨자 범위 제한	36
4-3. 배열과 Implied DO 순환	36
4-4. 응용 예제	38
4-5. 실습 예제	40

Chapter 5. 입 · 출력

5-1. READ문	41
5-2. WRITE문	41
5-3. FORMAT문	42
5-4. 파일 입·출력	44
5-5. 응용 예제	46
5-6. 실습 예제	49

Chapter 6. 고급기능

6-1. 형선언문(Type Statement)	51
6-2. 응용 예제	53
6-3. 실습 예제	55

부 록

1. Microsoft Visual Fortran 6.0 익히기	56
-------------------------------------	----

Chapter 1. 포트란의 개요와 형식

1-1. 포트란 언어의 역사

수치과학 계산용 언어인 FORTRAN은 FORMula TRANslation에서 유래되었다. 제 1세대 때 주로 사용되어 온 기계어와 어셈블리어의 불편함을 줄이기 위하여 인간이 컴퓨터를 사용함에 있어 인간 중심의 언어로 만든 자연언어인 제 2세대 언어로 포트란은 1950년 중반에 탄생되었다.

인간 중심 언어의 탄생은 컴퓨터 사용자에게는 획기적인 프로그램 향상을 가져왔으며, 기계적인 지식이 충분하지 않더라도 프로그램을 작성하는 데 있어서 그 불편함을 해결하였다.

초기 포트란 언어는 기계어와 유사한 형태에서 사용자 측면을 충분히 고려하지 못하였지만, 수정과 보완을 거듭하면서 오늘날 과학 기술용 프로그래밍 언어의 위치를 확고히 하고 있다.

1-2. 포트란 언어의 실행 단계

A. 원시 프로그램(source program)

원시 프로그램은 사용자가 작성하는 프로그램으로 포트란 문법에 맞추어 작성한다.

B. 컴파일러(compiler)

컴파일러는 원시 프로그램을 컴퓨터가 인식하는 기계어로 번역하는 언어 번역 프로그램의 한 종류이다.

C. 목적 프로그램(object program)

목적 프로그램은 원시 프로그램을 컴파일러 과정을 거쳐 생성되는 기계어 프로그램이다.

D. 링크(link)

링크는 기계어로 번역된 목적 프로그램을 주기억 장치에 적재시켰을 때 실행 가능한 프로그램으로 변환해 주는 것이다.

E. 로더(loader)

로더는 주기억 장치가 실행할 수 있는 실행 가능한 프로그램을 주기억 장치로 적재시켜 준다.

※ 중요 사항

프로그래머는 문법상 오류가 발생하였을 때에는(컴파일링시에 오류 발생 유무 확인) 반드시 원시 프로그램을 오류가 없을 때까지 수정, 보완하는 과정을 반복하여야 한다.

1-3. 문제해결의 순서

A. 문제정의 및 이해

문제에서 구하고자하는 값을 알아낸다.

B. 결과 및 입력자료 분석, 문제의 변수

문제의 정의로부터 필요한 변수 및 관계식을 추출하고, 결과 및 입력자료로부터 문제의 변수명을 정한다.

C. 알고리즘 작성

알고리즘을 작성할 때, 다음과 같은 순서를 많이 쓴다.

- a. 데이터를 읽어온다.
- b. 계산을 실행한다.
- c. 결과값을 출력한다.

D. 프로그램 작성

알고리즘을 기초로 프로그램을 작성한다.

E. 프로그램 검증

프로그램의 이상이 없는지 검증한다.

1-4. Overview of Fortran

A. Comments(주석)

각 줄의 첫 칸이 C로 시작

```
ex) C This is a comment
    * So is this!
```

B. Fortran Statement

각 명령은 1줄을 사용한다. 부득이하게 다음 줄로 넘어갈 경우 다음 행 6번째 칸에 표시를 해 주고 쓴다.

C. Program statement

프로그램의 제목을 붙여 놓은 것

```
ex) PROGRAM DELAY
```

D. 변수의 이름

Memory Location의 이름

- Rule)
- a. 대문자로 시작한다. (A-Z)
 - b. 변수명의 최대길이가 6자가 되도록 한다.
 - c. 좋은 변수명 (LET1, CENT, HELLO)
 - d. 나쁜 변수명 (1LET, TEO*FOUR, CENTPERINCH)

E. TYPE declaration statement

데이터의 종류

- a. CHARACTER *변수길이 변수명
- b. INTEGER 변수명1, 변수명2,
- c. REAL 변수명1, 변수명2,

ex) REAL X, Y, Z : 실수형 변수선언
 INTEGER ME, IT : 정수형 변수선언
 CHARACTER *20 NAME : 문자형 변수선언

F. Parameter statement : 변수의 값을 지정해주는 명령

ex) PARAMETR (PI = 3.141593)

※ 변수를 쓸 때 REAL, STOP, END 등 명령어를 쓰지 않도록 주의하고, 또 변수들의 이름은 내용과 비슷하게 만드는 것이 좋다. 그리고 INTEGER인 변수는 첫글자가 I - N 로 시작하게 정의 내려주는 것이 좋은 방법이다.

1-5. Use of Columns in Fortran

포트란 언어의 문법과 코딩시에 그 규칙들은 곧 포트란 언어의 번역기인 컴파일러가 기계어로 번역하는 과정에서 사용자가 잘못 지정된 규칙을 찾아내는 것이므로 사용자가 프로그램을 작성하는 모든 규칙은 컴파일러간의 약속이므로 꼭 아래 규칙을 지켜야 한다.

표 2 Use of Columns in Fortran

COLUMN	USE
1	C 또는 * 표시를 사용해서 주석(comment)줄임을 알린다.
1-5	명령에 대한 레벨을 붙이는 열로 항상 숫자로 표현한다.
6	0(zero)을 제외한 어떤 문자를 포함시키면, 이 줄은 앞줄에 포함된다.
7-72	포트란 명령을 포함하는 열.
73-	이 이후에 열은 컴파일 할 때 무시된다.

ex)

C		This line contains a C in column 1 and is ignored.		
*		So is this one.		
	10	PRINT *, 'The distance travelled in miles is',		
		+ DISTNC		
1	2-5	6	7-72	73-

1-6. INTEGER, REAL, and CHARACTER Data Types

A. 수의 표현



$$\Rightarrow \text{mantissa} * 10^{\text{exponent}} = \text{mantissa}E^{\text{exponent}}$$

ex) $1.2E+6 = 1.2 * 10^6$ or 1,200,000 , $-15.0E-08 = -15.0 * 10^{-8}$ or -0.00000015

B. CHARACTER Declaration

SYNTAX : CHARACTER *len variable list
or CHARACTER variable list1 *len1, variable list2 *len2

ex) CHARACTER *10 NAME1, NAME2
CHARACTER STATE *10, CITY *10, ZIP *5

1-7. Arithmetic Expressions

◇ Rule for Expression Evaluation

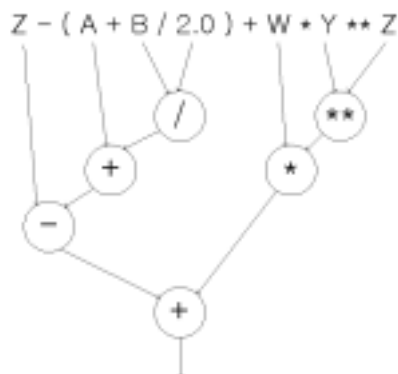
a. 연산자의 우선 순위

Operator	Order of Evaluation
**	first
*, /	next
+, -	last

b. 괄호()안에 있는 연산부터 한다.

c. 연산순서는 왼쪽에서 오른쪽으로 진행된다.

ex)



1-8. 포트란에서의 기초적인 명령어

A. READ Statement

READ (입력장치 번호, FORMAT문의 문번호, [ERR=문번호], [END=문번호]) 변수 리스트
 : 입력 장치로부터 데이터를 FORMAT문에서 지정한 형식대로 주기억 장치에 읽어 온다.
 데이터 입력시 ERROR가 발생하는 경우에는 ERR 뒤에 기술된 문번호로 분기한다.
 데이터 입력이 완료된 후에는 END 뒤에 기술된 문번호로 분기한다.

ex) READ *, ME ! *는 화면에서 입력을 받으라는 명령어이며, 단축해서 쓴다.
 READ (*,*) IRUM, KOR, ENG
 READ (5, 10, END=999) DAN, SU
 READ (5, 20, END=100, ERR=77) (A(I), I=1,10)

B. PRINT(or WRITE) Statement

PRINT (입력장치 번호, FORMAT문의 문번호) 변수 리스트

ex) PRINT *, ME ! *는 화면에 출력을 하라는 명령어이며, 단축해서 쓴다.
 PRINT (*,*) IRUM, KOR, ENG
 WRITE (5,10) IRUM, KOR, ENG

C. STOP Statement

STOP : 목적 프로그램의 실행의 끝을 나타내며 프로그램을 중지시킨다.
 한번 이상 사용 가능(실행문)

D. END Statement

END : 원시 프로그램의 끝을 나타냄. 문번호를 붙일 수 없으며 한번만 사용 가능(비실행문)

1-9. 포트란에서 사용되는 함수

ex) Using Function SQRT()

* programming

C uses function SQRT to perform three square root computations.

C Declarations

```
REAL NUM, ANSWER
```

C Get number and display its square root.

```
PRINT *, 'Enter the number'
```

```
READ *, NUM
```

```
ANSWER = SQRT(NUM)
```

```
PRINT *, ' The square root of the first number is ', ANSWER
```

```
STOP
```

```
END
```

* display

```
Enter the number
```

```
9.0
```

```
The square root of the first number is 3.00000
```


표 2 Table of Fortran Library Functions

FUNCTION REFERENCE	OPERATION	ARGUMENT TYPE	RESULT TYPE
INT(arg)	실수를 정수로 변환.(버림)	REAL	INTEGER
NINT(arg)	실수를 정수로 변환.(반올림)	REAL	INTEGER
REAL(arg)	정수를 실수로 변환.	INTEGER	REAL
ABS(arg)	절대값을 찾음.(실수에서 사용)	REAL	REAL
IABS(arg)	절대값을 찾음.(정수에서 사용)	INTEGER	INTEGER
SQRT(arg)	루트($\sqrt{\quad}$)값을 찾음.	REAL	REAL
EXP(arg)	지수(e^{arg})값을 찾음.	REAL	REAL
ALOG(arg)	자연로그($\ln \text{arg}$)값을 찾음.	REAL	REAL
ALOG10(arg)	로그($\log \text{arg}$)값을 찾음.	REAL	REAL
SIN(arg)	sine 함수 값을 찾음. (arg = radian angle)	REAL	REAL
COS(arg)	cosine 함수 값을 찾음. (arg = radian angle)	REAL	REAL
TAN(arg)	tangent 함수 값을 찾음. (arg = radian angle)	REAL	REAL
MOD(a1,a2)	a1/a2 의 나머지를 기억.	INTEGER	INTEGER
MAX0(a1, ... , an)	가장 큰 수를 찾음.(정수)	INTEGER	INTEGER
AMAX1(a1, ... , an)	가장 큰 수를 찾음.(실수)	REAL	REAL
MIN0(a1, ... , an)	가장 작은 수를 찾음.(정수)	INTEGER	INTEGER
AMIN1(a1, ... , an)	가장 작은 수를 찾음.(정수)	REAL	REAL

1-10. 공통된 프로그램 오류

- ◇ syntax errors,
- ◇ run-time errors,
- ◇ line-position errors, and
- ◇ arithmetic expression errors.

A. Syntax Errors

포트란 문법이 틀린 오류로 주로 다음과 같은 실수로 인해 오류가 난다.

- a. 스펠링이 틀릴 경우
- b. PRINT, READ 등등의 명령 뒤에 * 표시를 안 할 경우.
- c. 수식 표현에서 위치가 틀릴 경우
예) $GROSS - TAX = NET$ (틀린 표현) $NET = GROSS - TAX$ (맞은 표현)
- d. Comma(,) · apostrophe(') 등의 위치가 틀리거나 빼먹을 때 나는 오류.

B. Run_time Errors

수식 연산 중에 나는 오류.

- a. 주로 나누는 수가 0 일 때 나는 경우가 많다.
예) $X = 10.0$
 $Y = 0.0$
 $Z = X / Y$

C. Line-Positioning Errors

앞부분에서 언급한 포트란 명령 열을 잘못 써서 나는 오류

D. Errors in Arithmetic Expressions

이와 같은 경우엔 결과 값에서 차이를 보이는데, REAL, INTEGER를 혼용해 쓸 때 주로 나타난다. 더구나 포트란에서는 변수 젤 앞 문자가 (A-H, O-Z)로 시작하면 실수(REAL)형이고, (I-N)로 시작하면 정수(INTEGER)형이다.

1-11. 응용 예제

ex1) Using READ

* 한 변의 길이를 알 때, 정삼각형과 정사각형의 넓이를 구하는 프로그램을 작성하여라.

* The equation

$$\text{정삼각형의 넓이} = \frac{\sqrt{3}}{2} * L^2, \quad \text{정사각형의 넓이} = L^2$$

* programming (Computing area)

```
PROGRAM AREA
C 정삼각형과 정사각형의 넓이를 구하는 프로그램.
C 변수 정의
REAL X, TAREA, RAREA
C 변의 길이를 입력 받음.
PRINT *, '한 변의 길이를 넣으시오.'
READ *, X
C 정삼각형과 정사각형의 넓이 계산
TAREA = SQRT(3.0)/2 * X**2
RAREA = X**2
C 결과 출력
PRINT *, '정삼각형의 넓이 : ', TAREA
PRINT *, '정사각형의 넓이 : ', RAREA
STOP
END
```

* display

```
한 변의 길이를 넣으시오.
4
정삼각형의 넓이 : 13.85641
정사각형의 넓이 : 16.00000
```

* 이 프로그램은 한 변의 길이를 할 때, 정삼각형과 정사각형의 넓이를 구하는 프로그램이다.

ex2) Using Function SQRT()

* 방정식 $X^2 + 2AX + B = 0$ 근을 구하여라. ($A = -5, B = 24$)

* The equation

$$\text{근1} = -A + \sqrt{A^2 - B}, \quad \text{근2} = -A - \sqrt{A^2 - B}$$

* programming (Computing ROOTS)

```
PROGRAM ROOTS OF EQN
C 이차 방정식의 근구하기
C 변수정의
  REAL A, B
  PARAMETER (A=-5.0, B=24.0)
  REAL ROOT1, ROOT2
C 근구하는 계산
  ROOT1 = -A - SQRT(A**2 - B)
  ROOT2 = -A + SQRT(A**2 - B)
C 결과 출력
  PRINT *, 'X**2 + 2AX + B = 0 의 근은'
  PRINT *, ROOT1, ROOT2
  STOP
END
```

* display

```
X**2 + 2AX + B = 0 의 근은
4.00000      6.00000
```

* 이 프로그램은 방정식 $X^2 + 2AX + B = 0$, $A = -5, B = 24$ 일 때, 근을 구하는 프로그램이다.

ex3) Using Function EXP()

* 초기 Carbon 14 의 양을 주고, 주어진 시간 후에 남아 있는 carbon 14 의 양을 예측 하는 프로그램을 작성하시오.

* The equation

$$substance_{new} = substance_{old} \times e^{-ct} \quad (c = 0.693 / 5700.0)$$

* programming (Computing Radioactive Decay)

```
PROGRAM DECAY
C Computes the of a radioactive substance,
C Declarations
  REAL C
  PARAMETER (C = 0.693 / 5700.0)
  REAL SNEW, SOLD, TIME
C Get number and display its square root,
  PRINT *, 'Enter the initial amount of carbon 14'
  READ *, SOLD
  PRINT *, 'Enter the time in years'
  READ *, TIME
C Compute new population amount
  SNEW = SOLD * EXP(-C * TIME)
C Display result,
  PRINT *, ' The amount remaining is ', SNEW
  STOP
END
```

* display

```
Enter the initial amount of carbon 14
10.0
Enter the time in years
5700.0
The amount remaining is 5.00074
```

* 이 프로그램은 carbon 14이 10그램의 초기 질량을 가지고 5700년의 시간이 흐르면, 5그램이 질량이 남아있음을 보여주는 프로그램이다.

1-12. 실습 예제

실습1) 다음의 프로그램을 작성하여라.

☆ 반지름을 묻고, 반지름을 입력받은 다음 면적과 반지름을 구하여라.

(π 은 PARAMETER로 지정하시오.)

$$\Rightarrow \text{면적} = \pi R^2, \text{원주} = 2\pi R$$

hint) 예제 1번

실습2) 다음의 프로그램을 작성하여라.

☆ 이차방정식의 근을 구하여라. (A, B, C를 묻고 입력받아서 푸시오.)

☞ 이차 방정식 : $AX^2 + BX + C = 0$

$$\text{근1} = \frac{-B + \sqrt{D}}{2A}, \quad \text{근2} = \frac{-B - \sqrt{D}}{2A}$$

$$D = B^2 - 4AC$$

hint) 예제 2번

Chapter 2. 제어문(Control Statement)

2-1. 논리 표현

A. 비교 연산자

표 2-1 포트란에서 사용되는 비교 연산자

연산자	의미(설명)	의미(기호)
A .LE. B	A가 B보다 작거나 같으면..	$A \leq B$
A .LT. B	A가 B보다 작으면..	$A < B$
A .GE. B	A가 B보다 크거나 같으면..	$A \geq B$
A .GT. B	A가 B보다 크면..	$A > B$
A .EQ. B	A와 B가 같으면..	$A = B$
A .NE. B	A와 B가 같지 않으면..	$A \neq B$

B. 논리변수와 값(T, F)

LOGICAL DEBUG, FLAG ! 논리변수 정의

PARAMETER (DEBUG = .TRUE.) ! 논리변수 DEBUG에 참(TRUE)값을 줌.

PARAMETER (FLAG = .FALSE.) ! 논리변수 FLAG에 참(FALSE)값을 줌.

C. 논리 연산자

표 2-2 .AND. Operation

OPERAND1	OPERAND2	OPERAND1 .AND. OPERAND2
.TRUE.	.TRUE.	.TRUE.
.TRUE.	.FALSE.	.FALSE.
.FALSE.	.TRUE.	.FALSE.
.FALSE.	.FALSE.	.FALSE.

표 2-3 .OR. Operation

OPERAND1	OPERAND2	OPERAND1 .OR. OPERAND2
.TRUE.	.TRUE.	.TRUE.
.TRUE.	.FALSE.	.TRUE.
.FALSE.	.TRUE.	.TRUE.
.FALSE.	.FALSE.	.FALSE.

표 2-4 .NOT. Operation

OPERAND1	OPERAND2	OPERAND1 .NOT. OPERAND2
.TRUE.	.TRUE.	.FALSE.
.FALSE.	.FALSE.	.TRUE.

표 2-5 .EQV. Operation

OPERAND1	OPERAND2	OPERAND1 .EQV. OPERAND2
.TRUE.	.TRUE.	.TRUE.
.TRUE.	.FALSE.	.FALSE.
.FALSE.	.TRUE.	.FALSE.
.FALSE.	.FALSE.	.TRUE.

표 2-6 .NEQV. Operation

OPERAND1	OPERAND2	OPERAND1 .NEQV. OPERAND2
.TRUE.	.TRUE.	.FALSE.
.TRUE.	.FALSE.	.TRUE.
.FALSE.	.TRUE.	.TRUE.
.FALSE.	.FALSE.	.FALSE.

D. 연산자 처리순서

수식연산자(**, *, /, +, -), 비교 연산자(.EQ. , .NE. , .LT. , .LE. , .GT. , .GE.), 논리 연산자(.NOT. , .AND. , .OR. , .EQV. , .NEQV.) 순으로 처리를 한다.

2-2. 조건문(IF)

IF 문은 프로그램 실행을 순서적으로 수행을 하는 도중에 어떠한 조건에 따라 그 수행 순서를 결정하는 명령문이다.

IF 문은 표현하는 형식에 따라 다음과 같이 구분된다.

- ① 산술 IF문
- ② 논리 IF문
- ③ 블록 IF문

A. 산술 IF문

산술 IF문은 주어진 산술식의 계산 결과값 또는 변수의 저장된 값을 비교 판단하여 수행순서를 결정하는 명령문이다.

<형식>	IF (exp) m1, m2, m3 exp : 산술식 또는 변수 m1 : 분기 문번호1, m2 : 분기 문번호2, m3: 분기 문번호3
<설명>	위의 형식에서 괄호 안의 수식이나 변수의 값이 음수이면 m1, 0 이면 m2, 양수이면 m3로 분기한다.

ex) IF (N) 10, 20, 30

- 변수의 저장된 값이 음수이면 문번호 10번, 0이면 문번호 20번, 양수이면 문번호 30번으로 분기한다.

B. 논리 IF문

논리 IF문은 주어진 조건식이 참(TRUE) 또는 거짓(FALSE)이냐에 따라 그 수행순서를 결정하는 명령문이다.

<형식>	IF (exp) statement exp : 논리식 또는 논리형 변수 statement : 한 개의 실행문
<설명>	논리식(앞쪽 논리문 참고)에서 참이면 statement 실행한다.

ex) IF ((N .GE. 10) .AND. (M .LT. 30)) K = K + 1

- N이 10보다 크거나 같고 M이 30보다 작으면 K에 1을 더하라

C. 블록 IF문

논리 IF문에서는 처리 조건식이 참(TRUE)일 경우, 수행할 명령문이 단문(한 개의 문장)만 수행가능한 데 반해 블록 IF문은 수행처리 할 문장이 1개 이상을 표현할 수 있는 장점을 가지고 있다.

<형식>	<pre> IF (exp) THEN Tstatement ELSE Fstatement ENDIF </pre> <p>exp : 관계식 Tstatement : 관계식 exp가 참일 때 수행할 문. Fstatement : 관계식 exp가 거짓일 때 수행할 문.</p>
<설명>	<p>관계식 exp가 참이면 Tstatement를 수행하고, 거짓이면 Fstatement를 수행하게 된다.</p>

```

ex) IF (N .GE. 10) THEN K = K - 1
      ELSE    K = K + 1
      ENDIF

```

- N이 10보다 크거나 같으면 K에 1을 빼고, 아니면 K에 1을 더하라

2-3. 반복문(DO)

프로그램을 작성하다 보면 흔히 일정한 명령문들을 반복적으로 계속 수행하여야 할 필요가 있다. 이때 반복되는 명령문들을 순환문(DO문)과 단말문(CONTINUE) 사이에 두고 수행 횟수를 주게 되면 주어진 수행 횟수만큼 반복 수행하게 된다. 이러한 명령문을 순환문 또는 반복문이라고 한다.

A. 단일 DO문

<형식>	<pre> DO m iv=n1, n2, n3 statement m CONTINUE </pre> <p>m : DO LOOP의 마지막에 오는 명령문의 문번호 iv : 정수형 변수로 반복되는 명령문의 수행 횟수를 제어하는 변수 n1 : 정수형 상수 또는 변수로 표시되며, iv에 기억되는 초기치 n2 : 정수형 상수 또는 변수로 표시되며, iv의 값이 최종치 n2보다 크면 DO LOOP를 탈출하게 된다. n3 : DO LOOP 내의 문장을 한번 수행하게 될 때마다 iv의 값이 n3(증가치)만큼 증가한다. statement-1 : DO LOOP 내에서 수행하게 될 문장</p>
------	---

B. 다중 DO문

하나의 순환문 내에 1개 이상의 또 다른 순환문을 표현하는 것을 다중 순환문이라 하며, 이때 바깥쪽 순환문을 outer DO문이라 하고, 안쪽 순환문을 inner DO문이라 한다. 이렇게 다중 DO문으로 표현되는 경우에는 반드시 바깥쪽 DO문은 안쪽 DO문을 감싸고 있어야 한다.

◇ 다중 DO문의 도식적 범위

```

1) DO 50 I=n1, n2, n3
      DO 20 J=m1, m2, m3
          statement
20    CONTINUE
50 CONTINUE

2) DO 100 I=n1, n2, n3
      DO 50 J=m1, m2, m3
          statement
50    CONTINUE
      DO 100 I=k1, k2, k3
100 CONTINUE

3) DO 100 I=n1, n2, n3
      DO 100 J=m1, m2, m3
      DO 100 I=k1, k2, k3
100 CONTINUE

```

C. The WHILE Loop

<p><형식></p>	<pre> DO WHILE exp statement END DO </pre> <p>exp : 관계식 statement : 실행할 명령줄 DO WHILE : 반복문 명령어 END DO : Loop의 끝을 알리는 명령어</p>
<p><설명></p>	<p>exp 관계식이 참인 동안 명령(statement)를 실행한다.</p>

```

ex) DO WHILE (N .NE. 0)
      PRINT *, ' Enter N : '
      READ *, N
    END DO

```

- 변수 N값을 N이 0보다 작을 때까지 입력받는다.

D. DO Loop와 WHILE Loop의 비교

WHILE Loop	DO Loop
<pre> SUM = 0 COUNT = 1 DO WHILE (COUNT .LE. N) SUM = SUM + COUNT COUNT = COUNT + 1 END DO </pre>	<pre> SUM = 0 DO 10 COUNT = 1, N SUM = SUM + COUNT 10 CONTINUE </pre>

2-4. 응용 예제

ex1) Using IF

* 수를 입력하고 입력한 수가 양수인지? 음수인지? 0인지를 판별하라. 그리고 0이면 프로그램을 끝내라.

* programming

```
PROGRAM IF
C 음수인지 양수인지 0인지 판별
C 변수정의
  INTEGER N
  LOGICAL CON
  CON = ,FALSE,
C 숫자를 읽어 들임
  DO WHILE (CON ,NE, ,TRUE, )
    PRINT *
    PRINT *, '숫자...?'
    READ *, N
C 판별 및 출력
    IF (N ,GT, 0) THEN
      PRINT *, '양수'
    ELSE IF (N ,EQ, 0 ) THEN
      PRINT *, '영 0'
      CON = ,TRUE,
    ELSE
      PRINT *, '음수'
    ENDIF
  ENDDO
C 결과 출력
  PRINT *
  PRINT *, '실행끝'
  STOP
END
```

* display

숫자...?

0

영 0

실행끝

ex2) Using DO CONTINUE

* 1 ~ 10까지 더한 값과 곱한 값을 구하라.

* The equation

$$\text{더한 값} = 1 + 2 + \dots + 10$$

$$\text{곱한 값} = 1 * 2 * \dots * 10$$

* programming

```

PROGRAM SUM AND MUL
C 합계와 곱구하기
C 변수정의
  REAL SUM, MUL
C 초기화
  SUM = 0
  MUL = 1
C 근구하는 계산
  DO 10 I = 1, 10
    SUM = SUM + I
    MUL = MUL * I
  10 CONTINUE
C 결과 출력
  PRINT *, '1 ~ 10 까지의 합계는'
  PRINT *, SUM
  PRINT *, '1 ~ 10 까지의 곱은'
  PRINT *, MUL
  STOP
END

```

* display

```

1 ~ 10 까지의 합계는
55.00000
1 ~ 10 까지의 곱은
3628800.

```

* 이 프로그램은 1 ~ 10 까지의 합계와 곱을 구하는 프로그램이다.

ex3) Using IF

* 점수에 따라 학점을 부여한다.

* programming

```

PROGRAM DETERMINE A STUDENTS GRADE
C 학생의 성적에 따른 학점주기.
C 변수정의
  REAL MAXSR, MINA, MINB, MINC, MIND, MINSR
  REAL SCORE
C 초기화
  PARAMETER (MAXSR=100,0, MINSR=0,0)
  PARAMETER (MINA=90,0, MINB=80,0, MINC=70,0, MIND=60,0)
C 스코어를 넣는다.
  PRINT *, 'Enter your score : '
  READ *, SCORE
C 학점을 결정
  IF (MAXSR ,LT, SCORE) THEN
    PRINT *, 'Error: scores cannot be greater than ', MAXSR
  ELSE IF (MINA ,LE, SCORE) THEN
    PRINT *, 'You have an A!'
  ELSE IF (MINB ,LE, SCORE) THEN
    PRINT *, 'You have an B!'
  ELSE IF (MINC ,LE, SCORE) THEN
    PRINT *, 'You have an C!'
  ELSE IF (MIND ,LE, SCORE) THEN
    PRINT *, 'You have an D!'
  ELSE IF (MINSR ,LE, SCORE) THEN
    PRINT *, 'Unfortunately, you have an F,'
    PRINT *, 'See your instructor immediately,'
  ELSE
    PRINT *, 'Error: scores cannot be less than ', MINSR
  ENDIF

  STOP
END

```

* display

```

Enter your score :
95
You have an A!

```

* 이 프로그램은 각 성적에 따라 학점을 주는 프로그램이다.

2-5. 실습 예제

실습1) 다음의 프로그램을 작성하여라.

☆ 이차방정식의 근을 구하여라. (A, B, C를 묻고 입력받아서 푸시오.)

여기서는 근이 중근, 실근, 허근인지를 구분하고 표시하시오.

☞ 이차 방정식 : $AX^2 + BX + C = 0$

$$\text{근1} = \frac{-B + \sqrt{D}}{2A}, \quad \text{근2} = \frac{-B - \sqrt{D}}{2A}$$

$$D = B^2 - 4AC \quad \left(\begin{array}{l} D > 0 \text{ 일때 실근} \\ D = 0 \text{ 일때 중근} \\ D < 0 \text{ 일때 허근} \end{array} \right)$$

hint) 예제 1번

실습2) 다음의 프로그램을 작성하여라.

☆ 1 ~ 50까지의 숫자 중 홀수의 합과 곱을 다음 두 가지 방법 (DO ~ CONTINUE 과 DO WHILE ~ ENDDO)으로 구하라.

hint) 예제 2번

Chapter 3. 부프로그램(Subprogram)

지금까지는 어떤 문제를 해결하기 위하여 하나의 단위 프로그램 내에서 그 문제를 처리하였다. 따라서 단위 프로그램 내에서 동일한 명령문들이 여러 번 반복되어 나타나게 되었는데, 이러한 중복되는 명령문들을 별도로 하나의 단위 프로그램으로 묶고, 이 부분이 필요할 때마다 필요한 부분에서 호출하여 사용하도록 하면 프로그램이 간결하게 작성되어질 수 있다. 이처럼 호출하는 프로그램을 주프로그램(main program), 호출당하는 프로그램을 부프로그램(subprogram)이라 한다.

3-1. 부프로그램의 종류

A. 함수 부프로그램

- a. 내장함수(library function)
- b. 문함수(statement function)
- c. 함수 부프로그램(function subprogram)

B. 서브루틴 부프로그램(subroutine subprogram)

3-2. 내장함수

앞에서 기술한 바와 같이 내장함수는 컴퓨터 제작회사가 수학에서 많이 사용되는 함수들을 미리 작성하여 기계 내부에 공급해 둔 함수로 사용자의 편의를 위해 제공된 함수 프로그램이다. 내장함수는 공급함수라고도 하며, 이의 사용 형식은 앞장에서 기술되어 있으므로 여기서는 생략한다.

3-3. 문함수

하나의 단위 프로그램이 여러 번 기술되어지는 경우, 사용자가 프로그램에서 함수를 정의할 수 있도록 하고 정의된 함수를 호출하여 사용하는 함수를 문함수라고 한다.

<형식>	$fn(a_1, a_2, \dots, a_n) = exp$ <p>fn : 함수명 a_1, a_2, \dots, a_n : 가인수 exp : 산술식 또는 논리식</p>
<설명>	◇ 함수가 호출되면 exp(산술식 및 논리식)을 실행한다.

```

ex)  ADD(A, B)=A+B           !문함수 정의부분 (A와 B를 더한다.)
      X=ADD(3.5, 5.5)        !문함수 사용부분
      Z=(X, 1.5)
  
```

※ 중 요 사 항

- 프로그램 내에서 문함수의 위치는 모든 실행문보다 앞에 나와야 하며, 선언문 다음에 나타나야 한다.
- 함수명은 6자 이내로 작성할 수 있고 변수명의 규칙과 동일하게 취급된다.
- 가인수는 최소한 1개 이상 기술되어야 하고, 여러 개 있을 경우 콤마로 구분한다.
- 수식에는 산술식이나 논리식을 쓸 수 있으며, 첨자 변수는 사용할 수 없다.

3-4. 함수 서브프로그램

함수 서브프로그램은 문 함수와는 달리 하나의 단위 프로그램에서 함수 형태를 취하는 것이 아니고 함수의 형태가 하나의 독립된 단위 프로그램으로 작성되어 있는 것으로 완전한 서브프로그램의 형태를 취한 형태이다.

<형식>	<pre> ftype FUNCTION fn(v1, v2, ..., vn) statement RETURN END </pre> <p>ftype : 함수의 형태(REAL, INTEGER 등 등) fn : 함수명 a1, a2, ..., an : 가인수로 단순 변수 또는 배열명 RETURN : 제어의 흐름을 호출 프로그램으로 되돌려 주는 명령문으로 하나의 프로그램에서 한 개 이상 올 수 있다. END : 서브프로그램의 끝을 의미, 반드시 한 개만 올 수 있다.</p>
<설명>	<p>◇ 함수 서브프로그램의 관한 사항</p> <ol style="list-style-type: none"> ① FUNCTION문으로 시작해서 END문으로 끝난다. ② 함수 서브프로그램 실행중 RETURN문(or END문)을 만나면 실행이 자신을 호출했던 메인프로그램(or 다른 함수 서브프로그램이나 서브루틴 서브프로그램)으로 되돌아간다. ③ 함수 서브프로그램 내에는 반드시 『함수명=자료』의 문장이 포함되어 있어야 하며, RETURN(or END)문 실행시 『함수명』에 대입된 『자료』가 함수값으로 되돌려진다. ④ 최소한 1개 이상의 인자를 가져야 한다. ⑤ 함수 서브프로그램 정의시의 인자(가인자)와 실행시의 인자(실인자)의 개수는 서로 일치해야 한다. ⑥ 함수명이 I-N 중 어느 하나로 시작하면 함수값의 자료형은 정수형, 그렇지 않으면 실수형으로 처리된다. ⑦ 함수값, 인자의 자료형을 명시적으로 지정하려면 INTEGER, REAL문을 사용한다.

```

ex) C      Main Program
          I = 10
          J = 20
          PRINT *, MAX(I, J)
          PRINT *, MAX(5, 2)
          STOP
          END

C          FUNCTION Program
C          함수값 : M, N 중 최대값
          INTEGER FUNCTION MAX(M, N)
          IF (M .GE. N) THEN      ! M≥N이면
              MAX = M
          ELSE
              MAX = N
          ENDIF
          RETURN
          END

```

3-5. 서브루틴 부프로그램

서브루틴 부프로그램은 함수 부프로그램과 마찬가지로 한 개의 완전한 독립된 프로그램으로 간주하며, 함수 부프로그램이 계산 결과 값을 하나만을 전달하는 반면 서브루틴 부프로그램은 여러 개의 결과 값을 전달할 수 있다.

A. 부프로그램 형식

<형식>	<pre> SUBROUTINE sn(v1, v2, ..., vn) statement RETURN END </pre> <p>sn : 서브루틴 부프로그램명 v1, v2, ..., vn : 가인수 RETURN : 제어의 흐름을 호출 프로그램으로 되돌려 주는 명령 문으로 하나의 프로그램에서 한 개 이상 올수 있다. END : 부프로그램의 끝을 의미, 반드시 한 개만 올 수 있다.</p>
-------------------	---

B. 주프로그램에서 호출하는 형식

<형식>	<p>CALL sn(v1, v2, ..., vn)</p> <p>sn : 서브루틴 서브프로그램명</p> <p>v1, v2, ..., vn : 실인수</p> <p>CALL : 호출 프로그램에서 서브루틴 서브프로그램을 호출하고자 할 때 사용되는 명령문</p>
<특징>	<p>① 첫문장은 SUBROUTINE 명령문이고, 한 개 이상의 RETURN문을 포함해야 하며 END로 끝난다.</p> <p>② SUBROUTINE명은 프로그램 내에서 사용할 수 없다.</p> <p>③ 가인수가 없는 경우도 있다.</p> <p>④ 가인수는 COMMON, DATA 명령문에 사용할 수 없다.</p> <p>⑤ 가인수가 배열명일 경우 DIMENSION으로 선언하여야 한다.</p>

```

ex) C      Main Program
          INTEGER MAXIM
          I = 10
          J = 20
          CALL MAX(I, J, MAXIM)
          PRINT *, MAXIM
          STOP
          END

C      SUBROUTINE Program
C      두수 중 큰수를 구하고 출력한다.
          SUBROUTINE MAX(M, N, LARGE)
          IF (M .GE. N) THEN      ! M ≥ N이면
              LARGE = M
          ELSE
              LARGE = N
          ENDIF
          RETURN
          END

```

3-6. 서브루틴 부프로그램과 함수 부프로그램의 차이점

다음에 서브루틴 부프로그램과 함수 부프로그램의 차이점을 정리하였다.

	서브루틴 부프로그램	함수 부프로그램
되돌리는 값	없음	RETURN시에 함수명에 대입된 값
인자	없을 수도 있음	반드시 1개 이상
호출방법	CALL 서브루틴명	함수명(...)

서브루틴 부프로그램과 함수 부프로그램의 근본적인 차이점은, 서브루틴 부프로그램은 처리만을 수행하나, 함수 부프로그램은 처리를 수행한 후 함수명에 처리결과(함수값)를 되돌리는 것이라 할 수 있다.

3-7. 응용 예제

ex1) Using FUNCTION

* 한 변의 길이를 알 때, 정삼각형과 정사각형의 넓이를 구하는 프로그램을 함수를 써서 작성하여라.

* The equation

$$\text{정삼각형의 넓이} = \frac{\sqrt{3}}{2} * L^2, \quad \text{정사각형의 넓이} = L^2$$

* programming

```
PROGRAM AREA
C MAIN PROGRAM
C 정삼각형과 정사각형의 넓이를 구하는 프로그램.
C 변수 정의
    REAL X, TAREA, RAREA
C 변의 길이를 입력 받음.
    PRINT *, '한 변의 길이를 넣으세요.'
    READ *, X
C 결과 출력
    PRINT *, '정삼각형의 넓이 : ', TAREA(X)
    PRINT *, '정사각형의 넓이 : ', RAREA(X)
    STOP
END

C FUNCTION PROGRAM
C 정삼각형의 넓이 계산
    REAL FUNCTION TAREA(X)
    TAREA = SQRT(3.0)/2 * X**2
    RETURN
END

C 정사각형의 넓이 계산
    REAL FUNCTION RAREA(X)
    RAREA = X**2
    RETURN
END
```

* display

```
한 변의 길이를 넣으세요.
4
정삼각형의 넓이 : 13.85641
정사각형의 넓이 : 16.00000
```


ex2) Using SUBROUTINE

* 방정식 $X^2 + 2AX + B = 0$ 근을 구하여라. ($A = -5, B = 24$)

* The equation

$$\text{근1} = -A + \sqrt{A^2 - B}, \quad \text{근2} = -A - \sqrt{A^2 - B}$$

* programming

```

PROGRAM ROOTS OF EQN
C 이차 방정식의 근구하기
C 변수정의
  REAL A, B
  PARAMETER (A=-5.0, B=24.0)
  REAL ROOT1, ROOT2
C 근구하는 계산
  CALL ROOT(A, B, ROOT1, ROOT2)
C 결과 출력
  PRINT *, 'X**2 + 2AX + B = 0 의 근은'
  PRINT *, ROOT1, ROOT2
  STOP
END

C SUBROUTINE PROGRAM
  SUBROUTINE ROOT(A, B, RT1, RT2)
    RT1 = -A - SQRT(A**2 - B)
    RT2 = -A + SQRT(A**2 - B)
    RETURN
  END

```

* display

X**2 + 2AX + B = 0 의 근은	
4.00000	6.00000

* 이 프로그램은 방정식 $X^2 + 2AX + B = 0$, $A = -5, B = 24$ 일 때, 근을 구하는 프로그램이다.

ex3) Using FUNCTION

* factorial 구하기

* programming

PROGRAM FACTRL

C 함수 서브프로그램 부르기.

C 변수정의

INTEGER NUM, FAC, FACTOR

C 수를 넣는다.

PRINT *, 'Enter an integer between 0 and 10 '

READ *, NUM

IF (NUM .GE. 0) THEN

FAC = FACTOR(NUM)

PRINT *, 'The factorial of ', NUM, ' is ', FAC

ELSE

PRINT *, 'The factorial of a negative number is undefined.'

ENDIF

STOP

END

INTEGER FUNCTION FACTOR(N)

INTEGER N, I, PRO

PRO = 1

DO 10 I = N, 2, -1

PRO = PRO * I

10 CONTINUE

FACTOR = PRO

RETURN

END

* display

Enter an integer between 0 and 10

6

The factorial of 6 is 720

3-8. 실습 예제

실습1) 다음의 프로그램을 작성하여라.

☆ 반지름을 묻고, 반지름을 입력받은 다음 면적과 반지름을 구하는 프로그램을 두 가지 방법(FUNCTION 과 SUBROUTINE)을 써서 구하여라.

$$\Rightarrow \text{면적} = \pi R^2, \text{원주} = 2\pi R$$

hint) 예제 1, 2번

실습2) 다음의 프로그램을 작성하여라.

☆ 다음 이차방정식의 A, B, C를 묻고 입력받아서, 근을 구하는 프로그램을 두 가지 방법 (FUNCTION 과 SUBROUTINE)을 써서 구하여라.

☞ 이차 방정식 : $AX^2 + BX + C = 0$

$$\text{근1} = \frac{-B + \sqrt{D}}{2A}, \quad \text{근2} = \frac{-B - \sqrt{D}}{2A}$$

$$D = B^2 - 4AC$$

hint) 예제 1, 2번

Chapter 4. 배열(Array)

배열이라 함은 동일한 형태를 가지는 단순 변수들을 여러 개 나열할 경우 프로그램 계산 상 수식을 여러 번 또는 복잡하게 기술하는 경우가 많다. 따라서 단순 변수의 형태들을 배열이란 집합 형태로 묶고, 각각의 기억 장소를 그 집단의 원소의 위치로 표시하면 프로그램을 간단하게 표현할 수 있다. 특히 배열로 정의된 기억 장소들은 데이터를 필요한 기억 장소에 할당하여 항상 이용 가능하게 함으로써 데이터의 재입력 과정을 거치지 않아도 항상 활용할 수 있는 장점을 가지고 있다.

4-1. 배열 선언

변수는 사전에 선언하지 않고서도 사용할 수 있으나, 배열은 반드시 사전에 선언해 두어야만 사용할 수 있다.

A. DIMENSION

<형식>	① 1차원 배열의 선언 DIMENSION An(N) An : 배열명(array name) N : 열의 크기
	② 2차원 배열의 선언 DIMENSION An(N1, N2) An : 배열명(array name) N1 : 행의 크기 N2 : 열의 크기
	③ 3차원 배열의 선언 DIMENSION An(N1, N2, N3) An : 배열명(array name) N1 : 행의 크기 N2 : 열의 크기 N3 : 면의 크기
<설명>	◇ 『An』을 배열로 선언(컴파일러에게 통보)한다. ◇ 배열 명칭이 I-N 중 어느 하나로 시작하면 각 배열요소는 정수형이며, 그렇지 않으면 실수형이다.

ex) DIMENSION A(3), NAI(2)

- A를, 배열 요소가 A(1), A(2), A(3)인 실수형 배열로 선언한다.
- NAI를, 배열 요소가 NAI(1), NAI(2)인 정수형 배열로 선언한다.

B. INTEGER

<형식>	① 1차원 배열의 선언 INTEGER An(N) ② 2차원 배열의 선언 INTEGER An(N1, N2) ③ 3차원 배열의 선언 INTEGER An(N1, N2, N3)
<설명>	◇ 나머지는 위와 같고 각 배열 요소는 모두 정수형이다.

C. REAL

<형식>	① 1차원 배열의 선언 REAL An(N) ② 2차원 배열의 선언 REAL An(N1, N2) ③ 3차원 배열의 선언 REAL An(N1, N2, N3)
<설명>	◇ 나머지는 위와 같고 각 배열 요소는 모두 실수형이다.

D. CHARACTER

<형식>	① 1차원 배열의 선언 CHARACTER*n An(N) ② 2차원 배열의 선언 CHARACTER*n An(N1, N2) ③ 3차원 배열의 선언 CHARACTER*n An(N1, N2, N3)
<설명>	◇ 나머지는 위와 같고 각 배열 요소는 모두 문자수가 n인 문자형이다.

4-2. 배열 첨자 범위 제한

FORTRAN에서는 예를 들어 다음과 같이 배열 첨자의 하한(Lower Bound)과 상한(Upper Bound)을 지정할 수 있게 하고 있다.

```
DIMENSION N(3:5) ! N(3), N(4), N(5)를 준비, 3→하한, 5→상한
```

위와 같은 선언결과 N(3), N(4), N(5) 3개의 배열 요소를 사용할 수 있게 된다. 선언시 첨자 상한의 값은 하한의 값보다 커야한다.

4-3. 배열과 Implied DO 순환

입·출력 문에서 배열을 사용하는 경우 DO문을 사용하지 않고서도 배열의 각 요소를 처리할 수 있다. 이를 Implied DO 순환(묵시적 DO 순환 : DO문을 사용하지 않았으나 실제로는 DO문을 사용한 것과 동일한 효과를 갖는 것)이라 한다.

A. 입·출력 문에서 배열명은 배열 요소 전부를 나타낸다.

```
INTEGER N(3)
READ *, N
PRINT *, N
```

예를 들어 READ *, N 은 다음의 DO 문과 동일하다.

```
INTEGER N(3)
DO 10 I=1, 3
    READ *, N(I)
10 CONTINUE
DO 20 I=1, 3
    PRINT *, N(I)
20 CONTINUE
```

B. 입·출력 문에서 배열 요소의 일부만을 사용하려면

예를 들어 다음과 같이 하면 된다.

```
DIMENSION N(10)
PRINT *, (N(I), I=3, 5)
```

위 PRINT 문은 다음 DO문과 동일한 기능을 수행한다.

```
DIMENSION N(10)
DO 10 I=3, 5
    PRINT *, N(I)
10 CONTINUE
```

C. 이차원 배열

예를 들어 다음과 같이 하면 된다.

```
DIMENSION N(2, 3)
PRINT *, ((N(I), J=1, 3), I=1, 2)      ! I가 먼저 적용된다.
```

위와 같이 하면 우선 I=1로 하여 J=1, 2, 3에 대해 N(I, J)가 처리되고, 이어 I=2로 하여 J=1, 2, 3에 대해 N(I, J)가 처리된다. 이와 같은 방식을 사용하면 배열요소를 행첨자 우선으로 처리할 수 있다.

위 PRINT 문은 다음 DO문과 동일한 기능을 수행한다.

```
DO 10 I=1, 2
    DO 20 J=1, 3
        PRINT *, N(I, J)
20    CONTINUE
10 CONTINUE
```


4-4. 응용 예제

ex1) Using ARRAY

* 배열을 이용하여 구구단을 출력해라.

* programming

```

PROGRAM GUGUDAN
C 배열을 이용한 구구단 출력
C 변수정의
  INTEGER GU(27)
C 근구하는 계산
  DO 40 I=1, 9
    DO 20 J=1, 9
      GU(3*I-2) = J      ! 첫 곱셈수
      GU(3*I-1) = I      ! 앞 곱셈수
      GU(3*I) = I * J    ! 두수의 곱
C 결과 출력
      PRINT *, GU(3*I-1), ' * ', GU(3*I-2), ' = ', GU(3*I)
    20 CONTINUE
  40 CONTINUE

  STOP
END

```

* display

1	*	1	=	1
1	*	2	=	2
		.		
		.		
		.		
9	*	8	=	72
9	*	9	=	81

ex2) Using ARRAY

* 배열을 이용하여 행렬의 곱계산을 하여라.

* The equation

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31} & A_{11}B_{12} + A_{12}B_{22} + A_{13}B_{32} \\ A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31} & A_{21}B_{12} + A_{22}B_{22} + A_{23}B_{32} \end{pmatrix}$$

* programming

```

PROGRAM MATRIX
C 행렬의 곱계산
C 변수정의
    INTEGER A(2, 3), B(3, 2), GOB(2, 2), HAB
C 행렬의 원소를 읽어 들임(행 우선)
    PRINT *, ' 예) 1, 2, 3'
    PRINT *, '      4, 5, 6'
    PRINT *
    PRINT *, '      1, 2'
    PRINT *, '      3, 4'
    PRINT *, '      5, 6'
    READ *, ((A(I, J), J = 1, 3), I = 1, 2)
    PRINT *
    READ *, ((B(I, J), J = 1, 2), I = 1, 3)
C 행렬의 곱을 계산
    DO 20 I = 1, 2
        DO 20 J = 1, 2
            HAB = 0
            DO 10 K = 1, 3
                HAB = HAB + A(I, K) * B(K, J)
            10      CONTINUE
            GOB(I, J) = HAB
        20 CONTINUE
C 결과 출력
    PRINT *
    DO 30 I = 1, 2
        PRINT *, (GOB(I, J), J = 1, 2)
    30 CONTINUE
    STOP
END

```

* display (예와 같이 입력시 출력값)

22	28
49	64

4-5. 실습 예제

실습) 다음의 프로그램을 작성하여라.

☆ 주어진 함수를 써서 다각형의 면적을 구하는 프로그램을 작성하여라.

$$\text{area} = \frac{1}{2} |X_1Y_2 + X_2Y_3 + \cdots + X_{n-1}Y_n + X_nY_1 - Y_1X_2 - Y_2X_3 - \cdots - Y_{n-1}X_n - Y_nX_1|$$

예) 3각형

$$\text{area} = \frac{1}{2} |X_1Y_2 + X_2Y_3 + X_3Y_1 - Y_1X_2 - Y_2X_3 - Y_3X_1|$$

알고리즘)

1. 다각형의 꼭지점 수를 읽는다.
2. 좌표점을 입력한다.
3. 면적을 계산한다. (SUMSID 함수)
4. 계산된 면적을 보여준다.

hint1) SUMSID(X, Y, N) 면적을 계산하는 FUNCTION (X, Y는 N개의 배열임)

```

      REAL FUNCTION SUMSID(X, Y, N)
C 변수정의
      INTEGER N
      REAL X(*), Y(*)
C 지역변수
      INTEGER I
      REAL SUM
C 계산
      SUM = X(N)*Y(1) - Y(N)*X(1)
      DO 20 I = 1, N-1
          SUM = SUM + X(I)*Y(I+1) - Y(I)*X(I+1)
20  CONTINUE
      SUMSID = ABS(SUM)/2.0

      RETURN
      END
  
```

Chapter 5. 입 · 출력

5-1. READ문

READ문은 입력 장치로부터 데이터를 읽어 주기억 장치의 특정한 부분에 기억시키는 명령문이다.

<p><형식></p>	<p>READ(u, n1, ERR=n2, END=n3) V1, V2, ..., Vn</p> <p>u : 입력 장치번호로 각 장치별로 정해진 장치번호를 지정 n1 : FORMAT문의 문번호 n2 : 데이터 오류가 발생하였을 경우 분기하는 문번호 n3 : 데이터의 입력이 종료되었을 때 분기하는 문번호 V1, ..., Vn : 변수 리스트이고 변수의 개수는 필요에 따라 지정 ERR=n2, END=n3 는 생략 가능</p>
<p><설명></p>	<p>변수를 읽어 올 때 쓰는 명령어</p>

5-2. WRITE문

WRITE문은 기억 장치에 저장되어 있는 값을 지정된 출력장치를 통해 출력하는 명령문이다.

<p><형식></p>	<p>WRITE(U, n1) V1, ..., Vn</p> <p>U : 출력 장치번호 n1 : FORMAT문의 문번호 V1, ..., Vn : 변수리스트이고, 변수의 개수는 필요에 따라 지정</p>
<p><설명></p>	<p>변수를 쓸 때 쓰는 명령어</p>

※ READ문과 WRITE문에 장치번호 대신 *를 표시하면, 데이터의 입 · 출력을 키보드와 모니터를 이용한다는 의미이다. 출력 장치 번호는 이번 Chapter 뒤에서 자세히 다룸.

ex) READ *, ME ! *는 화면에서 입력을 받으라는 명령어이며, 단축해서 쓴다.
READ (5, 10, END=999) DAN, SU
READ (5, 20, END=100, ERR=77) (A(I), I=1, 10)
WRITE (5, 10) IRUM, KOR, ENG

5-3. FORMAT문

FORMAT문은 READ문과 WRITE문이 수행될 때 어떤 형식에 의해 입력하고 출력할 것인가를 컴파일러에게 그 정보를 제공하는 비실행문이다. 따라서 FORMAT문은 프로그램 내의 어느 곳에서 나타날 수 있지만 END문 앞에 나타내야 한다.

<형식>	<pre>m1 FORMAT(f1, f2, ..., fn)</pre> <p>m1 : FORMAT문의 문번호로 READ문이나 WRITE문에서 표시하는 문번호와 반드시 일치하여야 한다. f1, f2, ..., fn : FORMAT문에서 표시하는 변환 기호.</p>
<설명>	변수의 간격이나, 형태를 정할 때 쓰는 명령어

A. I 변환기호

I 변환기호는 정수형 데이터를 입력하거나 출력하고자 할 때 취급하는 변환기호이다.

<형식>	<pre>nIw</pre> <p>n : I 변환기호의 반복 횟수로 이때 반드시 자릿수가 동일하여야 한다. w : 입력 또는 출력하고자 하는 전체 자릿수(부호, 수치를 포함)</p>
------	--

```
ex) READ(*, 11) L, M, N
      11      FORMAT(I2, I3, I4)

      READ(*, 77) ISM, ITT
      77      FORMAT(2I3)
```

B. F 변환기호

F 변환기호는 실수형 데이터를 입력하거나 출력하고자 할 때 사용하는 변환기호이다.

<형식>	<pre>nFw.d</pre> <p>n : F 변환기호의 반복 횟수 w : 입력 또는 출력하고자 하는 총자릿수 d : 입력 또는 출력하고자 하는 총자릿수에서 소수점 이하 자릿수</p>
------	--

```
ex) READ(*, 11) AB, XY
      11      FORMAT(F6.2, F5.2)
```

```
      READ(*, 77) AX, AY
      77      FORMAT(2F4.1)
```

C. E 변환기호

E 변환기호는 실수형 데이터 중에서 지수형으로 표현된 데이터를 입력하거나 출력할 때 사용되는 변환기호이다.

<형식>	nEw.d
	n : E 변환기호의 반복 횟수 w : 입력 또는 출력하고자 하는 총자리수 d : 입력 또는 출력하고자 하는 E기호 앞부분에서의 소수점 표시 자리수

```
ex) READ(*, 11) AB, XY
      11      FORMAT(E10.5, E5.2)
```

```
      READ(*, 77) AX, AY
      77      FORMAT(2E8.5)
```

D. A 변환기호

A 변환기호는 포트란에서 사용되는 모든 문자를 입력하거나 출력할 때 사용하는 변환기호이다.

<형식>	nAw
	n : A 변환기호의 반복 횟수 w : 입력 또는 출력하고자 하는 총자리수

```
ex) READ(*, 11) PK, LM
      11      FORMAT(A3, A5)
```

```
      READ(*, 77) KIM, LEE
      77      FORMAT(2A10)
```

E. X 변환기호

X 변환기호는 데이터를 입력할 때와 기억된 데이터를 출력할 때 간격을 띄워 주는 변환기호이다.

<형식>	wX w : 입력 또는 출력할 때의 띄워줄 총 자릿수
------	---

```
ex) READ(*, 11) M, X, Y
      11      FORMAT(3X, I4, 3X, F5.1, 2X, E8.3)
```

F. /(slash) 변환기호

/ 변환기호는 데이터를 입력하거나 출력할 때 입력 레코드를 변경하거나 출력시에 줄을 바꾸어 주는 변환기호이다.

<형식>	n (/) n : /변환기호의 연속되는 개수 /변환기호가 1개 기술되면 () 생략
------	---

```
ex) READ(*, 11) I, J, X, Y
      11      FORMAT(2I4 // 2F5.1)
```

5-4. 파일 입 · 출력

A. 파일 입력

자료를 디스크 상의 파일로부터 입력받는 방법은, 자료를 입력받기 전에 OPEN문을 사용하여 파일을 열어주어야 한다는 점을 제외하고는, 지금까지 설명한 READ문에 의한 입력방법과 완전히 동일하다.

◇ 파일 입력의 3단계

- ① 파일을 연다.(OPEN문, STATUS='old') or (STATUS='unknown')
- ② 파일에 자료를 입력(READ문)
- ③ 파일을 닫는다.(CLOSE문)

B. 파일 출력

자료를 디스크상의 파일에 출력하는 방법은, 자료를 출력하기 전에 OPEN문을 사용하여 파일을 열고, 자료를 출력한 후에는 CLOSE문으로 파일을 닫아 준다는 점을 제외하고는, 지금까지 설명한 WRITE문에 의한 출력 방법과 완전히 동일하다.

◇ 파일 출력의 3단계

- ① 파일을 연다.(OPEN문, STATUS='new') or (STATUS='unknown')
- ② 파일에 자료를 출력(WRITE문)
- ③ 파일을 닫는다.(CLOSE문)

C. OPEN(파일을 열고 장치번호를 할당)

파일에 자료를 입출력하려면 우선 OPEN문을 사용하여 파일을 열어야 한다.

■ OPEN (장치번호, FILE = '파일명', STATUS='상태')

a. 장치번호

READ, WRITE문에서 장치명으로 일정 숫자(예를들어 6)를 지정하면 WRITE(6, *), READ(6, *) 등으로 표현을 하면 파일에 입출력을 할 수 있다.

b. FILE='파일명'

일정 이름의 파일을 생성하거나 이미 있는 파일에서 읽어올 파일명을 기입한다.

c. STATUS='상태'

출력시 : NEW, 입력시 : OLD(생략가능함), 입출력: UNKNOWN으로 표시해서 사용.

한 프로그램에서 여러 개의 파일을 열어 사용할 수도 있다. 여러개의 OPEN문에 서로 다른 장치 번호와 파일명을 지정해 주면 된다.

ex) OPEN(6, FILE='K.DAT', STATUS='OLD') ! K.DAT 파일에서 값을 읽어올 때
 OPEN(1, FILE='K.OUT', STATUS='NEW') ! K.OUT 파일에서 값을 저장할 때
 OPEN(9, FILE='K.TMP', STATUS='UNKNOWN') ! K.TMP 파일에서 값을 읽거나 저장할 때

D. CLOSE(파일을 닫는다)

파일의 사용이 끝나면 CLOSE문을 사용하여 파일을 닫아 준다.

■ CLOSE (장치번호) ! 6번 장치번호가 할당되어 열려져 있는 파일을 닫는다.

파일에 자료를 출력한 후 해당 파일을 닫아 주지 않으면 파일 작성이 불완전하게 됨.

ex) CLOSE(9)

5-5. 응용 예제

ex1) Using FORMAT

* FORMAT문을 이용해서 글자의 위치를 정확하게 정해주는 프로그램을 작성한다.

* programming

```
PROGRAM formats
C 형태를 맞추는 예제.
C 프린트한다.
C 11칸 띄고 출력한다.
  PRINT 1, '이것은 FORMAT문의 예제입니다. '
  1 FORMAT (11X, A)
C 9칸 띄고 출력한다.
  PRINT 2, '정확한 간격 및 위치를 정해줍니다. '
  2 FORMAT (9X, A)
C 처음은 한 칸 띄고 중간 칸은 6칸씩 띄고 출력한다.
  PRINT *
  PRINT 3, 'Sejong      ', 'Univ. ', '01****', 'Mr. Kim'
  PRINT 3, 'School of ', 'Mech. ', '    & ', 'Aero, '
  PRINT 3, '-----', '-----', '-----', '-----'
  3 FORMAT (1X, A, 6X, A, 6X, A, 6X, A)

  STOP
END
```

* display

```

이것은 FORMAT문의 예제입니다
정확한 간격 및 위치를 정해줍니다. '

Sejong      Univ.      01****      Mr. Kim
School of    Mech,      &          Aero,
-----    -----    -----    -----
```

ex2) Using ARRAY & OPEN

* 배열을 이용하여 행렬의 곱계산을 하여라. DATA파일을 만들고, OPEN문을 사용하여 행렬을 읽어 들이고, 이를 결과 값을 파일로 출력하여라.

☞ MATRIX.DAT 파일

```
1, 2, 3
4, 5, 6

1, 2
3, 4
5, 6
```

* The equation

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31} & A_{11}B_{12} + A_{12}B_{22} + A_{13}B_{32} \\ A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31} & A_{21}B_{12} + A_{22}B_{22} + A_{23}B_{32} \end{pmatrix}$$

* programming

```
PROGRAM MATRIX
C 행렬의 곱계산
C 변수정의
  INTEGER A(2, 3), B(3, 2), GOB(2, 2), HAB

C FILE OPEN
  OPEN (1, FILE = 'MATRIX.DAT', STATUS='OLD')
  OPEN (6, FILE = 'RESULT.OUT', STATUS='UNKNOWN')

C 행렬의 원소를 읽어 들임(행 우선)
  READ (1, *) ((A(I, J), J =1, 3), I=1, 2)
  READ (1, *) ((B(I, J), J =1, 2), I=1, 3)

C 행렬의 곱을 계산
  DO 20 I=1, 2
    DO 20 J=1, 2
      HAB=0
      DO 10 K=1, 3
        HAB=HAB+A(I,K)*B(K,J)
      10 CONTINUE
      GOB(I,J)=HAB
    20 CONTINUE

C 결과 출력
  WRITE (6, *)
  DO 30 I=1, 2
```

```
        WRITE(6, *) (GOB(1,J), J=1,2)
30 CONTINUE

        CLOSE(6)
        CLOSE(1)

        STOP
        END
```

*** display (MATRIX.DAT와 같이 입력시 RESULT.OUT 출력파일의 내용)**

22	28
49	64

5-6. 실습 예제

실습1) 다음의 프로그램을 작성하여라.

☆ 다음 데이터들을 파일로 만들고, 이를 읽어서 모두 더한 값과 곱한 값을 구하라.

☞ NUM.DAT 파일

1	11
2	12
3	13
4	14
5	15
6	16

hint) 예제 2번

실습2) 다음의 프로그램을 작성하여라.

☆ 구구단 2단 ~ 9단까지 새로운 파일로 생성하라.

hint) 예제 2번

Chapter 6. 고급기능

6-1. 형선언문(Type Statement)

A. IMPLICIT문

IMPLICIT문은 변수명의 첫 글자를 임의로 지정하여 변수의 형을 결정하는 선언문이다.

<형식>	<pre>IMPLICIT type(C1, ..., Cn), type(D1, ..., Dn) type : INTEGER, REAL, COMPLEX, CHARACTER, LOGICAL 중 하나 C1, ..., Cn, D1, ..., Dn : 변수명의 첫 번째 영문자</pre>
------	--

ex) `IMPLICIT INTEGER(A, X-Z), REAL(K-M), LOGICAL(N), CHARACTER(C)`

B. DATA문

특정 변수나 배열의 원소에 초기치를 설정하고자 할 때 그 변수나 배열의 원소들이 많은 경우 대입문을 통해 초기치를 넣으면 프로그램이 길어지고 복잡해진다. 따라서 DATA문을 이용하여 초기치를 설정하여 간단하게 처리할 수 있다.

<형식>	<pre>DATA v1, v2, ..., vn /d1, d2, ..., dn/ v1, v2, ..., vn : 변수명 또는 배열명 d1, d2, ..., dn : 상수</pre>
------	---

ex) `DATA A, B, C /1.0, 2.5, 3.3/`
`DATA A/1.0/, B/2.5/, C/3.3/`

```
DIMENSION X(5, 5), A(10)
DATA (X(5, I), I=1, 5)/2*(1.1, 2.2), 0.0/
DATA A/10*0.0
```

C. COMMON문

COMMON문은 단위 프로그램과 단위 프로그램 간에 공동 기억 장소를 마련하여 인수에 의한 정보교환이 아닌 실변수와 가변수 간의 기억 장소를 공유함으로 기억장소를 절약하고, 부프로그램에 전달되는 인수가 많은 경우에 사용하면 유리하다.

<형식>	<pre>COMMON /name1/V11, ..., V1n /name2/V21, ..., V2n/...</pre> <p>name : common block name으로 앞 뒤에 '/'가 있어야 한다. V11, ..., V1n V21, ..., V2n : 변수, 배열</p>
<특징>	<ul style="list-style-type: none"> ① 단위 프로그램에서 첫 실행문이 나오기 전에 기술한다. 다른 선언문이 있으면 순서에 관계없이 기술한다. ② 부프로그램의 COMMON에서 기술된 변수는 대응되는 변수가 주 프로그램의 COMMON문에 반드시 기술되어 있어야 한다. ③ 함수명과 서브루틴명을 기술할 수 없다. ④ 가인수를 COMMON에 기술할 수 없다. ⑤ 같은 이름의 변수는 COMMON에 함께 기술할 수 없다.

ex)

* main program

```
COMMON A, B, /KIM/C, J
CALL CACL
```

```
STOP
END
```

* sub program

```
SUBROUTINE CACL
COMMON /KIM/ X, I
```

```
RETURN
END
```

6-2. 응용 예제

ex1) Using subroutine & common

* 한 변의 길이를 알 때, 정삼각형과 정사각형의 넓이와 둘레를 구하는 프로그램을 SUBROUTINE과 COMMON문을 써서 작성하여라. SUBROUTINE엔 길이만 주고, COMMON문을 통해 계산된 값이 공유되게 하여라.

* The equation

$$\text{정삼각형의 넓이} = \frac{\sqrt{3}}{2} * L^2,$$

$$\text{정사각형의 넓이} = L^2,$$

$$\text{정삼각형의 둘레} = 3L,$$

$$\text{정사각형의 둘레} = 4L$$

* programming (Computing area)

PROGRAM AREA

C MAIN PROGRAM

C 정삼각형과 정사각형의 넓이를 구하는 프로그램.

C COMMON문을 통한 공유

COMMON /CIAR/ TAREA, RAREA, TCIR, RCIR

C 변수 정의

REAL X, TAREA, RAREA, TCIR, RCIR

C 변의 길이를 입력 받음.

PRINT *, '한 변의 길이를 넣으세요.'

READ *, X

C 계산

CALL ARCI(X)

C 결과 출력

PRINT *, '정삼각형의 넓이 : ', TAREA

PRINT *, '정사각형의 넓이 : ', RAREA

PRINT *, '정삼각형의 둘레 : ', TCIR

PRINT *, '정사각형의 둘레 : ', RCIR

STOP

END

C SUBROUTINE PROGRAM

C 정삼각형 및 정사각형의 넓이 및 둘레 계산

```

SUBROUTINE ARCI(X)
COMMON /CIAR/ TAREA, RAREA, TCIR, RCIR

TAREA = SQRT(3.0)/2 * X**2
RAREA = X**2
TCIR = 3*X
RCIR = 4*X

RETURN
END

```

*** display**

한 변의 길이를 넣으시오.

4

정삼각형의 넓이 : 13.85641

정사각형의 넓이 : 16.00000

정삼각형의 둘레 : 12.00000

정사각형의 둘레 : 16.00000

6-3. 실습 예제

실습1) 다음의 프로그램을 작성하여라.

☆ 다음 이차방정식의 A, B, C를 묻고 입력받아서, 근을 구하는 프로그램을 SUBROUTINE을 써서 구하여라. 단, Main Program과 Subroutine Program에서의 모든 변수는 COMMON 문을 써서 공용으로 쓰게 만들어라.

☞ 이차 방정식 : $AX^2 + BX + C = 0$

$$\text{근1} = \frac{-B + \sqrt{D}}{2A}, \quad \text{근2} = \frac{-B - \sqrt{D}}{2A}$$

$$D = B^2 - 4AC$$

hint) 예제 1번

◆ 부 록 (Microsoft Visual Fortran 6.0 익히기)

1. 시작하기

A. 시작 메뉴에서 찾기

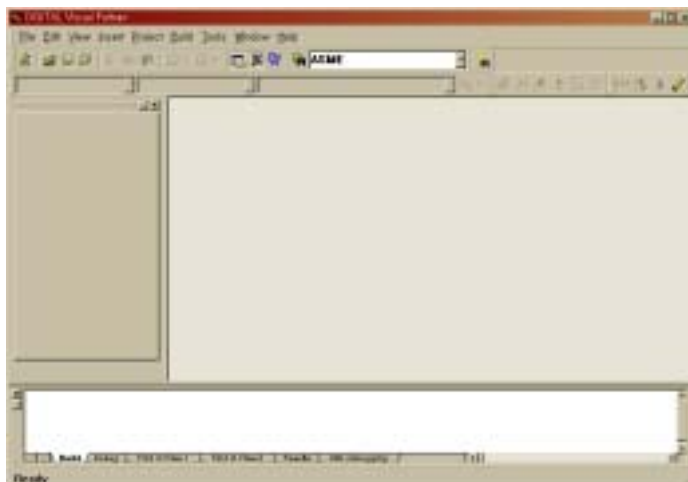


시작 ⇒ 프로그램 ⇒ Visual Fortran 6.0 ⇒ Developer Studio를 클릭한다.

B. *.FOR 파일에서 시작..

파일 확장명이 FOR인 파일을 더블클릭 하면 시작된다.

◇ FORTRAN 초기에 뜨는 창형 모습



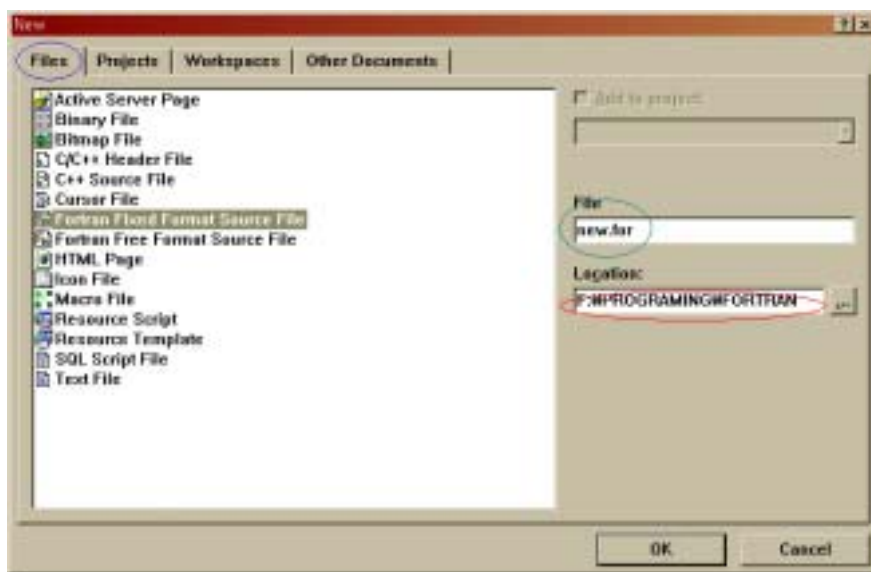
2. 새 파일 만들기

A. 메뉴에서 새 파일 만들기



- 메뉴에서 FILE ⇒ NEW를 실행(Ctrl+N)한다.
- New라는 창에서 Files를 선택한다.
- 아래 그림에서 보이듯이 Fortran Fixed Format Source File을 클릭하고,
- Location의 위치를 정해준다.

자신이 작업을 하는 폴더를 지정해준다. Visual Fortran에선 한글 폴더와 한글 파일명을 인식하지 못하므로 영어로 된 폴더를 만들고 그 위치를 지정해 준다.

- File명을 기재한다.
- OK를 클릭한다.




B. 아이콘에서 새 파일 만들기

- 아이콘 메뉴에서 아이콘()을 클릭한다.
- 새 창이 뜨면, 저장()을 클릭한다.
- 다른 이름으로 저장하는 창이 하나 뜨는데, 저장할 폴더를 정한 다음,
- 파일이름의 확장명이 *.FOR이 되도록(ex : new.for) 이름을 써준 다음 저장을 클릭하면 새 파일로 시작된다.




3. 포트란 파일열기 및 저장하기

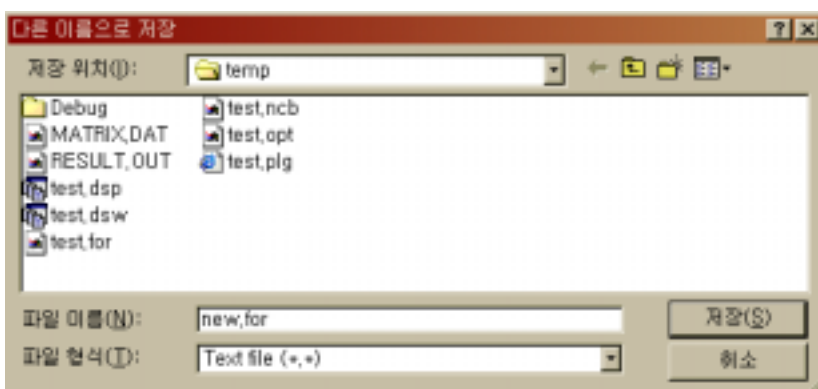
A. 파일열기

- 메뉴에서 FILE ⇒ OPEN를 실행(Ctrl+O)한다.
- 또는 아이콘 메뉴에서 열기()를 클릭한다.
- 파일 형식을 모든 파일(*.*) 또는 Fortran file을 찾아 클릭한다.
- 열 파일을 클릭하여 연다.



B. 저장하기

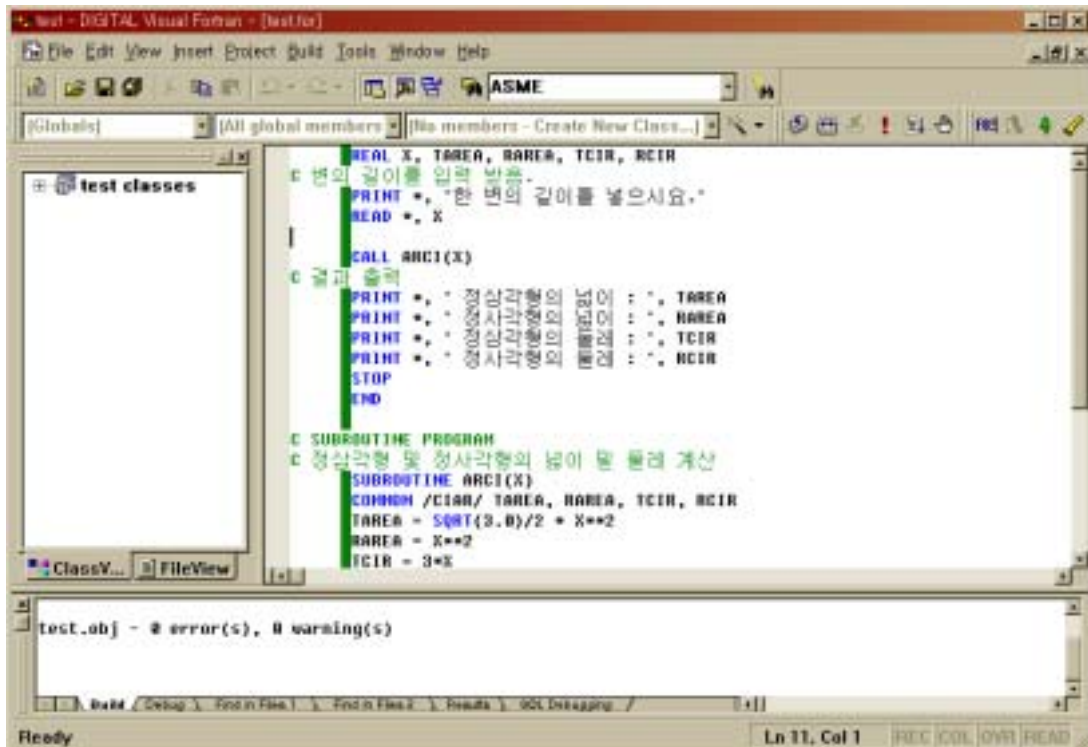
- 메뉴에서 FILE ⇒ SAVE를 실행(Ctrl+S)한다.
- 또는 아이콘 메뉴에서 저장()를 클릭한다.
- 파일 형식을 모든 파일(*.*) 또는 Fortran file을 찾아 클릭하고,
- 저장할 파일명을 쓴 다음 저장한다.
- 다른 이름으로 저장 할 때는 FILE ⇒ Save as..를 실행하고 저장한다.




4. 포트란 실행하기

A. 원시 프로그램(Source program)을 작성한다.


- 용지 형식에 맞게 프로그램을 작성한다.
- 명령어는 파란색으로 나타나며,
- 6번째 열과 주석은 녹색으로 표시된다.




B. 컴파일(Compile)

- 프로그램 작성이 끝나면, 컴파일을 실행한다.
- 메뉴에서 Build ⇒ Compile 파일명.for를 실행(Ctrl+F7)한다.
- 또는 아이콘 메뉴에서 Compile()를 클릭한다.
- 컴파일을 실행하여, 창 아래쪽에 상태 창에 error와 warning을 검사 확인하고, 에러가 나타나지 않을 때까지 소스프로그램을 확인하고 고친다.

C. 빌드(Build)

- 컴파일이 끝나면, 빌드를 실행한다.(링크를 해주는 부분)
- 메뉴에서 Build ⇒ Build 파일명.for를 실행(F7)한다.
- 또는 아이콘 메뉴에서 Build()를 클릭한다.
- 빌드를 실행(Linking)하여, 창 아래쪽에 상태 창에 error와 warning을 검사 확인하고, 에러가 나타나지 않을 때까지 소스프로그램을 확인하고 고친다.

D. 실행(Execute)

- 프로그램 작성이 끝나면, 컴파일을 실행한다.
- 메뉴에서 Build \Rightarrow Execute 파일명.for를 실행(Ctrl+F5)한다.
- 또는 아이콘 메뉴에서 Execute()를 클릭한다.
- Execute을 실행하면 새로운 도스창이 하나 뜨는데, 여기서 에러(Run-time error가 주로 발생)가 발생하면, 소스프로그램을 확인하고 고친 후 다시 실행한다.

