제 2 부 — 제어 구조와 입출력

6장
입출력 기초와 서식

From Memory to File, and Back

목록 지정 입출력 · 편집 기술자 · csv 파일 · NaN과 Inf

THE THREE STEPS

```
open(newunit=u, &
     file="data.csv", &
     status="replace", &
     action="write")

write(u, '(서식)') 목록

close(u)
```

학습 목표

이 장에서 익히는 네 가지

01

기본 입출력

read, write, print 로 값을 읽고 쓴다.

02

서식 다루기

format 과 편집 기술자로 출력 모양을 직접 정한다.

03

데이터 넘기기

결과를 csv 로 출력해 Python 으로 넘긴다.

04

예외 맛보기

NaN 과 Inf, ieee_arithmetic 을 개요 수준에서 이해한다.

도입

print *, 하나로는 넘지 못하는 세 가지 벽

1

정밀 포맷 정렬

정수 i 와 x, x^2, x^3 의 변화를 한눈에 들어오는 표로 출력하려 해도, print *, 는 데이터마다 칸 너비를 임의로 정해 열 정렬이 무너진다.

칸 너비와 자릿수를 직접 지정해야 한다

2

파일 영구 저장

터미널 화면에 출력하고 사라지는 방식으로는 데이터 분석이 어렵다. 디스크 상의 물리 파일에 줄을 맞춰 기록해야 한다.

범용 표준 csv 포맷으로 저장해야 한다

3

결함 데이터 필터링

0으로 나누기나 음수의 로그 계산이 발생하면 하드웨어가 Inf 또는 NaN 신호를 보낸다. 이 값이 csv 에 저장되면 후처리 단계에서 문제가 된다.

실행 중에 미리 걸러 낼 장치가 필요하다

이 장은 이 세 벽을 차례대로 넘는다 — 서식 지정, 파일 출력, 예외 검출.

6.1 목록 지정 입출력

서식을 * 하나에 맡긴다 (list-directed I/O)

```
program list_io
   implicit none
   real :: a, b

   read *, a, b
   print *, "sum  =", a + b
   print *, "diff =", a - b
end program list_io
```

```
                !echo "3.5 1.5" | ./list_io
```

```
 sum  =    5.00000000         diff =    2.00000000
```

*read * 은 표준 입력(stdin)에서 값을 읽는다. 코랩처럼 대화형 입력 창이 없으면 echo 와 파이프로 값을 전달한다.*

구분자의 유연성

토큰을 분리하는 규칙이 느슨하다. 3.5 1.5 처럼 공백으로 나뉘든, 3.5, 1.5 처럼 쉼표로 나뉘든, 서로 다른 두 줄에 입력되든 각 값이 a 와 b 에 정확히 할당된다.

출력 제어권 상실

5.00000000 좌우에 넓은 공백이 임의로 삽입되거나 소수점 이하 자릿수가 고정되는 것은 전적으로 컴파일러의 기본 설정에 따른 결과다.

언제 쓰나

알고리즘 개발 도중 변수를 실시간으로 모니터링하거나, 서식보다 수치의 무결성 검증이 우선일 때 편리하다.

6.2 read · print · write

데이터의 흐름 방향과 대상 장치

read

입력

지정된 또는 표준 입력 장치(키보드)로부터 데이터를 스캔해 읽고, 프로그램 내부의 변수 버퍼에 값을 저장한다.

print

화면 전용 출력

표준 출력 장치(화면)로만 데이터를 보내는 단방향 명령이다. 장치 번호를 적을 필요가 없어 신속한 모니터링에 쓴다.

write

장치 지정 출력

출력할 장치와 파일을 직접 지정할 수 있다. 장치 자리에 * 를 적으면 표준 출력(화면)이 된다.

아래 두 구문은 정확히 같은 방식으로 화면에 출력한다

```
print *, "hello"
write(*, *) "hello"
```

write(*, *) 의 첫 번째 * 는 전송할 목적 장치(표준 출력), 두 번째 * 는 적용할 서식(기본 서식)을 뜻한다.

*print *, "via print:", n 과 write(*, *) "via write:", n 은 같은 42 를 같은 모양으로 찍는다.*

알아두기

동적 장치 번호 newunit

레거시 방식 — 번호를 직접 지정

```
open(6, file="out.csv", ...)
```

개발자가 임의의 정수를 장치 번호로 직접 지정했다. 거대 수치 해석 모델을 융합할 때 다른 서브루틴이나 라이브러리가 쓰는 번호와 우연히 겹쳐 자원이 충돌하는 중대한 결함을 안고 있다.

현대 방식 — 시스템에 맡긴다

```
open(newunit=u, file="out.csv", ...)
```

현재 전혀 사용하지 않는 빈 장치 번호를 운영체제로부터 동적으로 받아 정수형 변수 u 에 자동 할당한다. 번호 충돌을 신경 쓸 필요가 없고, 할당과 제거를 컴파일러가 관리한다.

장치 번호란 무엇인가 — 프로그램이 열어 둔 파일 하나하나에 붙는 정수 꼬리표다. write(u, ...) 의 u 가 바로 "어느 파일에 쓸 것인가"를 가리킨다.

파일 입출력의 자세한 내용은 14장에서 다룬다.

6.3 서식과 편집 기술자

출력 모양의 통제권을 되찾는다

write 범용 서식 구문

```
write(장치, '(편집기술자1, 편집기술자2, ...)') 출력목록
```

print 단축 서식 구문

```
print '(편집기술자1, 편집기술자2, ...)', 출력목록
```

1 작은따옴표로 감싼다

서식은 작은따옴표('')로 감싸고, 그 안의 괄호 속에 편집 기술자들을 쉼표로 나열한다.

2 좌에서 우로 일대일 매핑

출력 목록의 변수 값들과 괄호 안의 편집 기술자들이 왼쪽에서 오른쪽으로 순차적으로 짝지어진다.

3 반복 횟수를 앞에 붙인다

같은 기술자를 여러 번 쓸 때는 3I5 처럼 앞에 횟수를 적는다. 3I5 는 I5, I5, I5 와 같다.

4 리터럴 문자열

큰따옴표로 글자를 감싸면 그대로 출력된다. 예: '(F8.3, " m/s")'

장치 식별자 자리에는 표준 화면 출력을 뜻하는 * 나 고유 장치 번호를 배정할 수 있다.

6.3 편집 기술자

자료형마다 짝이 정해져 있다

`Iw`	정수, 전체 폭 w 칸 고정. 남는 공간은 좌측 공백	`I5 + 42`	`␣␣␣42`
`I0`	정수, 필요한 최소 자릿수만큼 가변 너비	`I0 + 42`	`42`
`Fw.d`	실수 고정소수점. 총 폭 w, 소수 이하 d 자리	`F8.3 + 3.142`	`␣␣␣3.142`
`Ew.d`	실수 지수 표기. 가수가 0.1 이상 1 미만	`E12.4`	`␣␣0.6022E+24`
`ESw.d`	실수 과학적 표기. 가수가 1 이상 10 미만	`ES12.4`	`␣␣6.0220E+23`
`A`	문자열을 본래 길이 그대로, 공백 낭비 없이	`A + "hello"`	`hello`
`Aw`	문자열 고정 폭 w. 우측 정렬, 좌측은 공백	`A10 + "hello"`	`␣␣␣␣␣hello`
`nX`	값과 매핑되지 않는 공백 생성기. n칸 건너뛴다	`3X`	`␣␣␣`
`/`	현재 줄을 끝내고 다음 행 첫머리로 줄바꿈	`/`	`(new record)`

모든 편집 기술자는 지정된 칸 너비 안에서 우측을 기준으로 정렬한다(right-alignment).

6.3 [예제]

칸 너비를 맞춰 표로 출력하기

```
print '(A4, A10, A10, A12)', "i", "x", "x^2", "x^3"
do i = 1, 5
   x = real(i) * 0.5
   print '(I4, F10.3, F10.3, F12.4)', i, x, x**2, x**3
end do
```

```
   i         x       x^2         x^3
   1     0.500     0.250      0.1250
   2     1.000     1.000      1.0000
   3     1.500     2.250      3.3750
   4     2.000     4.000      8.0000
   5     2.500     6.250     15.6250
```

같은 총 너비로 맞춘다

머리글의 A4, A10, A10, A12 와 데이터의 I4, F10.3, F10.3, F12.4 가 똑같이 총 36칸(4+10+10+12)을 차지하도록 매핑했다.

우측 정렬이 표를 만든다

문자도 숫자도 각 열의 우측 경계면을 기준으로 정렬되므로, 자릿수가 변해도 세로줄이 일직선을 유지한다.

꼬리 숫자를 잘라 낸다

F10.3 은 소수 셋째 자리까지, F12.4 는 넷째 자리까지 규정해 부동소수점 연산의 꼬리를 균일하게 제거한다.

6.3 [예제]

E 와 ES — 같은 값, 다른 표기

```
avogadro = 6.022e23

print '(A, ES12.4)', "N_A = ", avogadro
print '(A, E12.4)',  "N_A = ", avogadro
```

ES12.4 **6.0220E+23** 가수가 1 이상 10 미만

E12.4 **0.6022E+24** 가수가 0.1 이상 1 미만

왜 지수 표기가 필요한가

기상학·천문학·분자 물리에서 마주하는 극단적으로 크거나 작은 수치를 F 기술자로 출력하면 0 이 무수히 나열되어 가독성이 무너진다.

논문에는 ES 가 자연스럽다

ES 는 학술 보고서의 수치 레이아웃과 일치한다. E 는 소수점 왼쪽을 무조건 0 으로 만들어 부동소수점을 일괄 표준화할 때 주로 쓴다.

6.3 [흔한 실수]

서식과 자료형이 어긋나면 실행 중에 죽는다

```
real :: x = 3.14

print '(I5)', x
```

```
Fortran runtime error: Expected INTEGER
  for item 1 in formatted transfer, got REAL
```

컴파일은 통과한다

실수형 변수를 정수용 I 기술자에 매핑해도 빌드 단계에서는 문법적 하자가 없어 정상 통과한다. 문제는 실행 시점에 강제 종료된다는 것이다.

짝을 맞춘다

실수는 반드시 F · E · ES 기술자와, 정수는 정확히 I 기술자와 짝을 맞추어 매핑해야 한다.

함께 조심할 것 — 너비가 좁으면 별표가 나온다

값이 지정한 칸 너비 w 에 들어가지 않으면 Fortran 은 숫자 대신 그 칸을 * 로 가득 채워 출력한다. 잘린 숫자를 보여 주는 대신, 표시할 수 없다는 사실을 분명히 알리는 설계다.

```
F4.1  +  123.456  →  ****
```

csv 파일을 만드는 세 단계

```
open(newunit=u, file="data.csv", status="replace", action="write")
write(u, '(서식)') 출력목록
close(u)
```

1 open

동적 장치 번호 할당과 파일 개설

newunit=u 로 사용 가능한 장치 번호를 시스템이 자동 선택해 정수형 변수 u 에 할당한다.

2 write

파일로 출력 전송

화면을 뜻하는 * 대신 장치 변수 u 를 앞에 배치해, 데이터가 화면이 아닌 지정된 파일로 흘러가게 한다.

3 close

장치 연결 폐쇄와 플러시

모든 기록이 끝나면 장치 변수 u 를 완전히 닫는다. 이때 파일 기록이 최종적으로 완결된다.

`status="replace"` 같은 이름의 파일이 이미 있으면 기존 데이터를 지우고 새로 생성한다.

`action="write"` 쓰기 전용으로만 연결해, 원치 않는 동시 읽기 간섭 등의 충돌을 예방한다.

6.4 [예제]

2차원 투사체 궤적을 csv 로 저장하기

```
real, parameter :: g = 9.81, v0 = 30.0, deg = 45.0

theta    = deg * pi / 180.0
t_flight = 2.0 * v0 * sin(theta) / g
dt       = t_flight / real(n)

open(newunit=u, file="trajectory.csv", status="replace", action="write")
write(u, '(A)') "x,y"
do i = 0, n
   t = real(i) * dt
   x = v0 * cos(theta) * t
   y = v0 * sin(theta) * t - 0.5 * g * t**2
   write(u, '(F10.4, ",", F10.4)') x, y
end do
close(u)
```

```
x,y
    0.0000,    0.0000
    2.2936,    2.2362
    4.5872,    4.3578
```

첫 줄에 머리글

write(u, '(A)') "x,y" 로 열 이름을 먼저 적는다. Python 이 이 줄을 보고 각 열이 무엇인지 안다.

서식 안에 쉼표를 끼운다

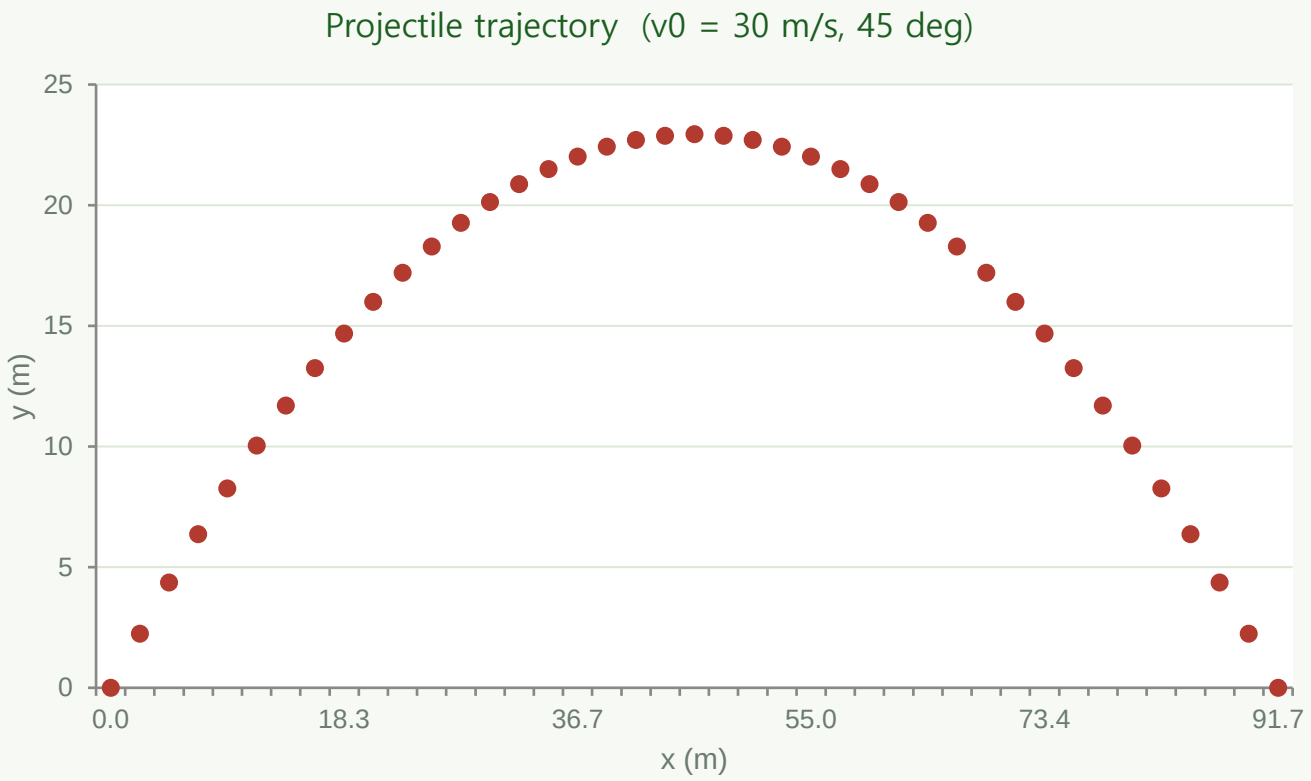
'(F10.4, ",", F10.4)' — 두 값 사이에 리터럴 쉼표를 넣는 것이 csv 를 만드는 전부다.

F10.4 의 효과

숫자 왼쪽에 일정한 공백 여백이 들어가 열이 가지런히 정렬된다. 총 41개의 좌표 쌍이 기록된다.

6.4 [예제]

Fortran 이 계산하고 Python 이 그린다



완벽한 좌우 대칭

발사 지점 (0, 0)에서 최고점까지의 수평 변위와 최고점에서 낙하 지점까지의 하강 궤적이 대칭 포물선을 그린다. 최고 높이 약 22.94 m, 수평 비행 거리 약 91.74 m.

등시간 간격 격자점

각 점은 동일한 비행 시간 간격(dt ≈ 0.108초)이다. 수평 속도는 등속도 운동이므로 가로축을 따라 점들이 일정한 간격으로 배치된다.

Python 은 csv.reader 로 읽고 next(reader) 로 머리글 한 줄을 건너뛴 뒤 2행부터 좌표를 읽는다.

6.5 NaN · Inf

수학적 한계를 벗어난 연산의 결과

Inf

Infinity, 무한대

0이 아닌 유한한 실숫값을 정확히 0.0 으로 나누었을 때 나타난다. 연산 방향에 따라 양의 무한대(Inf) 또는 음의 무한대(-Inf)가 된다.

```
7.7 / 0.0   →   Infinity
-7.7 / 0.0  →  -Infinity
```

NaN

Not a Number, 숫자가 아님

분모와 분자가 모두 0인 부정형 연산이나 음수의 제곱근, 음수 영역의 자연로그처럼 수학적으로 정의할 수 없을 때 부여된다.

```
0.0 / 0.0    →   NaN
sqrt(-1.0)   →   NaN
log(-5.0)    →   NaN
```

부동소수점 하드웨어 연산 표준(IEEE 754)은 통상적인 실수 범주에 속하지 않는 이 값들을 특별한 상태로 규정해 처리한다.

6.5 [예제]

Inf 와 NaN 을 직접 만들어 보기

```
real :: a, zero

a = 7.7
read *, zero
print *, "a / zero    =", a / zero
print *, "-a / zero   =", -a / zero
print *, "zero / zero =", zero / zero
```

```
!echo "0.0" | ./div_zero
```

```
 a / zero    =          Infinity
-a / zero    =         -Infinity
zero / zero =                NaN
```

왜 0.0 을 외부에서 받나

소스에 a / 0.0 이라고 직접 적으면 컴파일러가 컴파일 단계에서 오류를 낸다. 분모를 외부 입력으로 받아 실행 시점에 0.0 을 주입해 Inf 와 NaN 상태를 의도적으로 만들었다.

[흔한 실수] 코랩에서 read 가 멈춘다

대화형 입력 창이 없는 환경에서 read 를 만나면 프로그램이 값을 기다리며 멈춘 것처럼 보인다. echo 와 파이프로 값을 미리 주입해야 한다.

6.5 ieee_arithmetic

프로그램 스스로 예외를 식별하게 하기

`ieee_is_finite(x)`

x 가 Inf 도 NaN 도 아닌 유한한 '정상 수치'일 때 참을 반환한다.

`ieee_is_nan(x)`

x 가 정상적인 수치 계산이 불가한 NaN 상태일 때 참을 반환한다.

`ieee_value(...)`

시스템 아키텍처에 종속되지 않고 안전하게 검증용 기준값(ieee_positive_inf, ieee_quiet_nan)을 변수에 주입한다.

```
use, intrinsic :: ieee_arithmetic

inf_val = ieee_value(0.0, ieee_positive_inf)
nan_val = ieee_value(0.0, ieee_quiet_nan)
```

실행 결과

```
is_nan(nan), is_nan(normal):
   T  F
is_finite(normal), is_finite(inf):
   T  F
nan == nan:  F
```

왜 필요한가

터미널에 찍힌 오류는 눈으로 판별할 수 있지만, 연산 도중 정상 데이터만 골라 저장하려면 프로그램 스스로 예외를 식별해야 한다. 이 감지 장치를 두면 NaN 이나 Inf 가 csv 에 저장되는 것을 원천적으로 막을 수 있다.

6.5 [흔한 실수]

NaN 은 자기 자신과도 같지 않다

`nan_val == nan_val` → `F`

IEEE 754 표준에 따르면 NaN 은 그 어떤 수치와도 대등하지 않으며, 심지어 자기 자신과 비교해도 논리적 동등성이 성립하지 않는다.

위험한 코드

```
if (x == nan_val) then
if (x /= nan_val) then
```

컴파일러는 무조건 거짓으로 처리한다. 예외가 검출되지 않고 그대로 통과한다.

안전한 코드

```
if (ieee_is_nan(x)) then
if (.not. ieee_is_finite(x)) then
```

NaN 여부는 ieee_is_nan 으로, 무한대를 포함한 비정상 여부는 .not. ieee_is_finite 로 검사한다.

일반 비교 연산자로는 NaN 을 절대 잡을 수 없다 — 반드시 표준 함수를 쓴다.

오류 학습

status="new" 로 열면 두 번째 실행에서 죽는다

오류 재현

```
open(newunit=u, file="out.csv", &
     status="new", action="write")
```

1차 실행 `out.csv written`

2차 실행
```
Fortran runtime error:
  Cannot open file 'out.csv': File exists
```

new 의 엄격성

단순히 "새로 생성한다"가 아니라 "반드시 타깃 파일이 존재하지 않아야 한다"는 뜻이다. 이미 있으면 기존 데이터의 안전을 위해 쓰기를 즉각 중단하고 예외를 던진다.

오류 해결

```
open(newunit=u, file="out.csv", &
     status="replace", action="write")
```

replace 의 작동 흐름

파일이 이미 있으면 기존 데이터를 소거(truncate)해 빈 공간으로 만든 뒤 새로 쓴다. 파일이 없으면 new 와 똑같이 새로 생성한다.

연구 현장의 기본값

매개변수를 조금씩 바꾸며 같은 코드를 수십 번 재실행하므로, 덮어쓰기가 잦은 출력에는 replace 를 기본으로 쓴다. new 는 고유한 새 파일 생성을 보장해야 하는 보안 설계에서만 선별 적용한다.

6장 정리

요약

출력과 서식

- 목록 지정 입출력: read *, / print *, / write(*, *). 간편하지만 칸과 자릿수를 통제할 수 없다.
- print 는 항상 표준 출력, read 는 입력, write(장치, 서식) 은 장치를 골라 출력한다.
- 서식 지정 출력: print '(편집기술자, ...)', 목록 — 출력 모양을 직접 통제한다.
- 편집 기술자: 정수 Iw · I0, 실수 Fw.d, 지수 Ew.d · ESw.d, 문자 A · Aw, 공백 nX, 줄바꿈 /.

파일과 예외

- 파일 출력 3단계: open(newunit=u, ..., status="replace") → write(u, ...) → close(u).
- csv 만들기: 서식 안에 "," 를 끼우고 첫 줄에 머리글을 적는다. Python 은 next(reader) 로 머리글을 건너뛴다.
- x/0.0 → Inf, 0.0/0.0 · 음수의 log · sqrt(-1.0) → NaN.
- ieee_is_finite · ieee_is_nan 으로 검출한다. NaN 은 == 비교로 판별할 수 없다.

자주 하는 실수: 좁은 너비 → * 출력 · 서식과 자료형 불일치 → 실행 중 오류 · NaN 을 == 로 검사 → 항상 거짓 · status="new" 재실행 → File exists

연습 문제

1. 정수 한 개를 I0, I5, I8 세 가지 너비로 각각 한 줄씩 출력하고 칸 너비 차이를 눈으로 확인하라.

2. 3.14159265 를 F10.2, F10.5, ES14.6 으로 각각 출력하라.

3. 섭씨 c 를 입력받아 "25.0 C = 77.0 F" 형태가 되도록 리터럴 문자열과 F 기술자를 섞어 한 줄로 출력하라.

4. 1부터 10까지의 i 와 그 제곱을 칸 너비를 맞춰 표로 출력하라. 머리글을 포함하라.

5. 실수를 일부러 F4.1 로 출력해 * 가 나오는 값을 찾고, 너비를 늘려 정상 출력되게 고쳐라.

6. a / b 를 출력하고 결과가 유한한지 ieee_is_finite 로 함께 출력하라. b 에 0을 주어 확인하라.

7. t 를 0부터 1까지 0.1 간격으로 바꾸며 t 와 exp(-t) 를 decay.csv 로 출력하고 선그래프를 그려라.

8. 정수 한 개를 받아 그 수의 구구단을 "3 x 1 = 3" 형태로 9줄 출력하라.

9. 0~360도를 10도 간격으로 각도와 sin, cos 를 trig.csv 로 출력하고 두 곡선을 겹쳐 그려라.

10. 실수 두 개의 합·차·곱·몫을 모두 F12.4 로 칸을 맞춰 표로 출력하라.

다음 장에서는 배열 — 선언과 인덱싱, 배열 연산과 부분 배열을 다룬다.