제 1 부 — 기초

1장

Fortran 소개와 개발 환경

Formula Translation, since 1957

컴퓨터의 작동 원리 · Fortran의 역사 · Colab + gfortran 실습

1957

세계 최초의
고급 프로그래밍 언어

IBM · John Backus

학습 목표

이 장에서 배우는 세 가지

01

Fortran에 대한 이해

1957년에 태어난 언어가 왜 지금 다시 주목받는지, 현대 Fortran의 역사와 동향을 배운다.

02

실행의 큰 그림

소스 코드가 트랜지스터 위에서 실행되기까지, 추상화 계층의 전 과정을 이해한다.

03

컴파일 · 실행

구글 코랩에 gfortran을 설치하고 직접 프로그램을 컴파일해 실행한다.

도입 질문

이 장이 끝나면 답할 수 있다

?

왜 하필 Fortran인가?

1957년의 언어가 어떻게 2024년 인기 순위를 다시 끌어올렸을까? 달 탐사 궤적 계산과 일기예보는 어떤 언어로 돌아갈까?

?

설치 없이 실행할 수 있을까?

컴파일러를 내 컴퓨터에 설치하지 않고, 브라우저만으로 Fortran 프로그램을 작성하고 실행해 결과를 볼 수 있을까?

?

숫자를 그림으로?

Fortran이 계산해 낸 결과를 한눈에 들어오는 그래프로 표현하려면 어떤 도구를 어떻게 이어 붙여야 할까?

세 질문의 답이 곧 이 장의 내용이다.

1.1 컴퓨터는 어떻게 명령을 수행하는가

실리콘 한 조각에서 CPU까지

실리콘 도핑 › p-n 접합 › 트랜지스터 › 논리 게이트 › **CPU 완성**

전압이 높으면 1, 낮으면 0 — 트랜지스터는 게이트에 전압을 걸어 전류를 통제하는 아주 작은 전자 스위치다.

NOT

1은 0으로, 0은 1로 뒤집는다

AND

양쪽 모두 1일 때만 1을 낸다

OR

둘 중 하나만 1이어도 1을 낸다

XOR

두 입력이 서로 다를 때만 1

게이트를 조립하면 덧셈기(가산기)가 되고, 거기에 레지스터와 메모리를 붙여 컴퓨터의 두뇌인 중앙처리장치(CPU)가 완성된다.

1.1 추상화 계층 (Abstraction Layer)

트랜지스터에서 Fortran까지, 다섯 개의 층

최상위 소프트웨어 Fortran / Python	수식을 영문자와 기호로 표현한다 (예: y = sin(x))
컴파일러 · 어셈블리어 Compiler / Assembly	고급 언어를 CPU에 종속적인 기계 부호로 번역하는 연결 고리
기계어와 ISA Machine Code & ISA	CPU가 직접 해독하는 0과 1의 이진 명령어, 그리고 명령어 집합 규격
디지털 논리 회로 Digital Logic	AND · OR · NOT 게이트와 이들이 조합된 연산 장치(ALU), 레지스터
물리적 트랜지스터 Transistor / Physics	실리콘 웨이퍼에 새겨진 수백억 개의 전류 제어 스위치

아래층의 복잡한 물리 현상은 숨기고(캡슐화), 위층에는 기능만 노출한다 — 그래서 수백억 개의 스위치를 다룰 수 있다.

1.2 컴퓨터 언어의 진화

소프트웨어는 CPU와 메모리를 부리는 명령서

C

CPU (중앙처리장치)

오직 연산만을 위한 강력한 계산기. 스스로 오래 기억하지 못하므로 끊임없이 메모리에서 데이터를 받아 계산한다.

M

메모리 (기억장치)

CPU가 바로 꺼내 쓰도록 데이터를 보관하는 창고. 수식의 입력값과 연산 결과가 모두 이곳에 저장된다.

인간의 생각과 기계의 언어 사이의 간극

기계어
0과 1

›

어셈블리어
저급 언어

›

고급 언어
Fortran

›

컴파일러가
번역

초창기에는 0과 1을 직접 타이핑했다. 짧은 수식 하나에 수십 줄이 필요했고 점 하나만 틀려도 밤을 새웠다. 그 간극을 메우려 고급 언어와 컴파일러가 등장했다.

1.2 세계 최초의 고급 언어

FORmula TRANslation

이름 자체가 "수식 번역"에서 왔다. 철저하게 과학 기술 계산을 위해 태어난 언어다.

1953

John Backus가 IBM에
프로그래밍 비용 절감을 제안

1957

최초의 FORTRAN
컴파일러 탄생

천공카드

키보드도 모니터도 없이
종이에 구멍을 뚫어 입력

"사람이 읽고 이해할 수 있는 코드를 작성한다"

오늘날 당연하게 여기는 이 상식이 바로 FORTRAN에서 시작되었다. 이후 함수와 서브루틴 같은 현대적 개념을 흡수하며 수십 년간 과학·공학계의 표준 도구로 자리 잡았다.

1.2 아폴로 계획

인류를 달에 보낸 프로그래밍 언어

★

지상의 메인프레임

우주선을 달까지 보내고 무사히 귀환시키려면 극도로 정밀한 궤도·궤적 계산과 연료 소모 시뮬레이션이 필요했다. 이 계산을 수행한 IBM 대형 컴퓨터를 움직인 주역이 FORTRAN이었다.

◆

Dorothy Vaughan

영화 《히든 피겨스》의 주인공. 전자식 컴퓨터 시대의 도래를 직감하고 독학으로 FORTRAN을 마스터한 뒤, 팀원 전원을 전문 프로그래머로 재교육했다.

▲

오늘의 Copernicus

NASA 존슨 우주센터는 지금도 우주선 궤적 최적화 프로그램 Copernicus를 Fortran으로 구동하며 그 독보적 성능을 입증하고 있다.

아폴로 우주선 내부(AGC)에는 가벼운 어셈블리어가 실렸지만, 임무를 설계한 지상 시스템은 FORTRAN으로 돌아갔다.

국제 표준(ISO/IEC)의 진화

표준	연도	핵심 변화
FORTRAN 66	1966	최초의 공식 표준. 파편화된 언어 사양을 하나로 규격화
FORTRAN 77	1978	블록 if, 문자형(character) 도입. 고정형식
Fortran 90	1991	자유형식, 모듈, 배열 연산, 동적 할당, 파생형 — 현대 Fortran의 시작
Fortran 95	1997	소규모 개선 (forall, pure, elemental 도입)
Fortran 2003	2004	객체지향 지원, C와의 상호 운용성, IEEE 부동소수점 반영
Fortran 2008	2010	do concurrent, block, 서브모듈, 코어레이(coarrays)
Fortran 2018	2018	병렬 기능 강화, C 상호 운용성 개선 — 이 책의 기준선
Fortran 2023	2023	최신 표준. 여러 편의 기능 추가와 문법적 보완

현대 Fortran은 보통 Fortran 90 이후의 표준을 가리킨다.

현대 Fortran이 지향하는 네 가지

1 자유 형식 (free form)

타자기와 천공카드 시절의 엄격한 칸수 제약에서 완전히 벗어나, 가독성과 개발 유연성을 갖춘 코드를 쓴다.

2 implicit none 의무화

모든 변수를 쓰기 전에 선언하도록 강제해, 변수명 오타로 인한 버그를 컴파일 단계에서 원천 차단한다.

3 배열 통째 연산

다중 루프 없이 배열과 행렬을 하나의 덩어리로 연산해, 현대 CPU의 병렬 처리와 벡터화 성능을 끌어올린다.

4 모듈(module) 구조화

기능별로 코드를 분리·독립화해 대규모 공학 계산에서도 안전하게 데이터를 공유하고 인터페이스를 관리한다.

고정 형식과 GOTO 중심의 스파게티 코드는 레거시 해독용 최소 지식으로만 다룬다. 모든 예제는 현대 Fortran 표준을 기준으로 진행한다.

1.3 알아두기

왜 최신 2023이 아니라 2018인가?

필수 요소는 이미 2008에 다 있다

모듈, 동적 배열(allocatable), 파생형, block, 서브모듈, do concurrent — 입문 과정에서 가르칠 현대적 기능은 2008만으로 충분하다.

표준 ≠ 컴파일러 지원

코랩 기본 gfortran 11은 -std=f2018은 받아들이지만 -std=f2023은 아직 인식하지 못한다. 컴파일러 구현은 표준보다 늦다.

이 책의 기준선

Fortran 2018

현대적 기능을 모두 갖추면서 이식성이 가장 높은 지점.
2018·2023의 고급 병렬화와 C 상호 운용성 세부는 입문 범위를 넘는다.

1.3 생태계 속의 위치

C · Python · Fortran, 무엇이 다른가

C

시스템 언어

- 운영체제 개발과 하드웨어 레지스터 직접 제어에 최적화
- 포인터 에일리어싱으로 루프 최적화가 막혀 성능 병목 발생
- 포인터와 메모리를 직접 관리하는 오버헤드가 크다

Python

범용 언어

- 간결한 문법과 광활한 라이브러리 생태계
- 데이터 과학·AI 엔지니어링의 표준 인터페이스
- 인터프리터 언어라 부동소수점 처리 속도가 느리다

Fortran

초고성능 연산 언어

- 다차원 배열 연산이 언어 자체에 내장
- 수학 공식을 교과서에 쓰인 형태 그대로 코딩
- 수천 코어가 협업하는 HPC에서 압도적 속도와 안정성

Fortran이 선점한 자리는 명확하다 — 극한의 수치 계산(extreme scientific computing) 그 자체다.

1.3 알아두기

Fortran이 유독 빠른 진짜 이유

"열 방향(column-major)으로 저장해서 빠르다"는 흔한 설명은 데이터 배열 방식일 뿐이다. 진짜 비결은 따로 있다.

1

포인터 에일리어싱의 부재

C에서는 여러 포인터가 같은 메모리를 가리킬 수 있어, 컴파일러가 "데이터가 꼬이면?" 하는 걱정에 보수적으로 최적화한다. Fortran은 서로 다른 배열이 메모리를 공유하지 않는다는 엄격한 규칙을 지키므로, 컴파일러가 루프를 과감히 쪼개고 뒤섞어 극단적으로 병렬화한다.

2

다차원 배열 연산의 태생적 최적화

수십 년간 기상 예측·천체 물리·양자역학 시뮬레이션을 처리하며 진화했다. 그 결과 컴파일러와 수치 라이브러리(BLAS, LAPACK)가 대형 행렬 처리에 기저부터 튜닝되어 있다.

3

파편화 없는 메모리 덩어리

C에서 다차원 배열을 다루면 메모리가 사방에 흩어지기 쉽다. Fortran은 하나의 거대하고 연속된 덩어리로 할당하므로, CPU가 일렬로 늘어선 데이터를 막힘없이 쓸어 담는다.

1.3 AI 시대와 HPC 수요

1950년대의 언어, 순위를 역주행하다

2021

TIOBE 인덱스
상위 20위권 복귀

2024

한때 10위권 내 진입
현재도 상위권 유지

HPC

GPU·클러스터에서
성능을 극한까지

AI와 시뮬레이션의 연산 폭증

모델 학습·추론과 과학 시뮬레이션에는 막대한 수학 연산이 필요하다. Fortran은 수치 계산과 다차원 배열을 언어 수준에서 지원해 고성능 하드웨어의 성능을 끝까지 끌어낸다.

검증된 자산의 유지보수와 확장

기상 예측, 항공우주, 원자력 등 세계의 핵심 인프라가 이미 Fortran으로 구축되어 있다. AI 시스템과 융합되며 이 자산을 최신 환경으로 확장하려는 움직임이 가속되었다.

1.4 개발 환경

Google Colab + gfortran

Google Colab

- 브라우저에서 실행되는 무료 주피터 노트북
- 클릭 한 번에 우분투 리눅스 가상 머신 할당
- 코드 셀 + 텍스트 셀, Shift+Enter로 실행
- %%writefile 파일명 → 셀 내용을 파일로 저장
- 줄 앞에 ! 를 붙이면 리눅스 셸 명령 실행

gfortran

- GNU 컴파일러 모음(GCC)에 포함된 무료·오픈소스
- 인텔 ifx 등 상업용도 있지만 입문에는 gfortran
- 어디서나 똑같이 동작해 학습 환경으로 최적
- 코랩에 기본 설치되어 있지 않아 직접 설치

설치와 확인

```
!apt-get install -y gfortran
!gfortran --version
```

런타임이 새로 시작되거나 연결이 끊기면 gfortran이 사라진다. 매 세션 첫 셀에 설치 명령을 넣어 두자.

1.5 첫 프로그램

코랩에서 쓰는 세 종류의 셀

1

Fortran 코드 셀

```
%%writefile hello.f90
```

첫 줄에 %%writefile 파일명.f90 을 붙이면 아래 내용이 모두 텍스트 파일로 저장된다.

2

컴파일 · 실행 셀

```
!gfortran ... / !./hello
```

리눅스 서버에서 수행되는 셸 명령이므로 각 줄 맨 앞에 느낌표(!)를 붙인다.

3

파이썬 시각화 셀

```
import matplotlib...
```

Fortran이 내보낸 데이터 파일을 읽어와 그래프를 그리는 파이썬 코드를 적는다.

[예제] 섭씨 온도를 화씨 온도로 변환

인사말을 출력하고, 섭씨 −40도부터 100도까지 10도 간격으로 화씨로 변환한 표를 temp.csv 파일로 저장한 뒤, 파이썬으로 그래프를 그린다.

1.5 Fortran 코드 셀

첫 프로그램 hello.f90

```
%%writefile hello.f90
program hello
   implicit none
   integer :: i, u
   real :: celsius, fahrenheit

   print *, "Hello, Fortran!"

   open(newunit=u, file="temp.csv", &
        status="replace", action="write")
   write(u, '(A)') "celsius,fahrenheit"
   do i = -40, 100, 10
      celsius = real(i)
      fahrenheit = celsius * 9.0 / 5.0 + 32.0
      write(u, '(F8.2, ",", F8.2)') celsius, fahrenheit
   end do
   close(u)

   print *, "temp.csv written"
end program hello
```

지금은 구조만

program 으로 시작해 end program 으로 끝나고, 둘째 줄에 implicit none 이 있으며, 변수를 먼저 선언한 뒤 계산하고 출력한다.

핵심은 이 한 줄

`fahrenheit = celsius * 9.0 / 5.0 + 32.0`

문법은 뒤에서

선언부·실행부와 implicit none 의 원리는 2장, do 반복문과 파일 입출력은 5·6장에서 다룬다.

Fortran은 컴파일(번역) → 실행의 두 단계를 더 거친다. 코드를 고쳤다면 반드시 다시 컴파일해야 변경이 반영된다.

컴파일 명령 한 줄 해부하기

```
!gfortran -O2 -std=f2018 -Wall hello.f90 -o hello
```

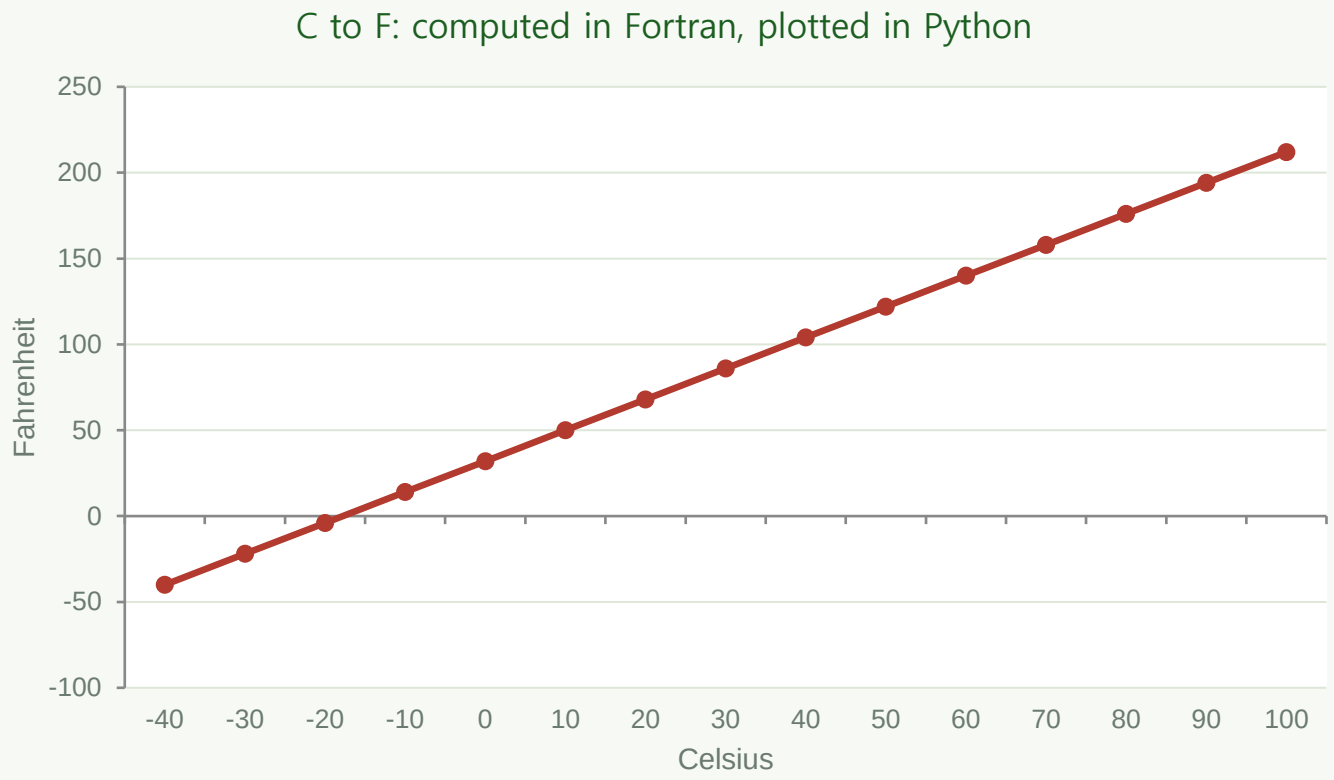
`-O2`	최적화 2단계. 연산 속도를 높이도록 컴파일러가 코드를 알아서 다듬는다.
`-std=f2018`	Fortran 2018 규격을 강제한다. 표준 밖 레거시 확장을 쓰면 경고한다.
`-Wall`	잡아낼 수 있는 모든 경고를 켠다. 이 책 예제는 경고 없이 깨끗하게 컴파일된다.
`hello.f90`	컴파일할 대상 소스 파일.
`-o hello`	실행 파일 이름을 hello 로 지정. 생략하면 a.out 이 만들어진다.
`!./hello`	만들어진 실행 파일을 실행한다. ./ 는 "현재 폴더"라는 뜻이다.

실행 결과

`Hello, Fortran!` · `temp.csv written` → 같은 디렉터리에 `temp.csv` 생성 (`!head temp.csv` 로 확인)

1.5 파이썬 시각화 셀

Fortran이 계산하고, Python이 그린다



역할 분담

Fortran은 격자 물리 연산과 행렬 대수를 가장 빠르고 안전하게 전담한 뒤 결과를 csv로 내보낸다. 그다음 표현력이 풍부한 파이썬이 matplotlib으로 그래프를 그린다.

그래프에서 확인할 점

- 섭씨 0도 → 화씨 32도
- 섭씨 −40도 = 화씨 −40도 교차점
- 정확한 정비례 직선

[흔한 실수] 코랩 기본 폰트에는 한글이 없어 그래프 라벨에 한글을 쓰면 네모(□)로 깨진다 — 라벨을 영문으로 쓰거나 나눔 폰트를 설치한다.

오류 학습

Fortran에서 만나는 네 가지 오류

1 컴파일 에러

번역 단계에서 문법이 어긋나 멈춘다. 어디가 틀렸는지 알려 주므로 그나마 다루기 쉽다.

`g = 9,8 → Unclassifiable statement`

2 런타임 에러

컴파일은 통과했지만 실행 중에 멈춘다. 존재하지 않는 파일을 읽으려는 경우가 대표적이다.

실행 도중 정지

3 반올림 에러

실수를 2진수로 저장하며 생긴 미세한 오차가 쌓인다. 실수를 == 로 비교하면 안 되는 이유다.

`0.1을 10번 더하면 1.00000012`

4 논리 오류

컴파일도 실행도 멀쩡한데 답이 틀리다. 컴파일러가 잡을 수 없어 개발자의 논리적 사고가 필요하다.

메시지 없음 — 가장 위험

컴파일 → 런타임 → 로직 → 반올림 순으로 점점 “조용해서” 더 위험하다. implicit none 과 -Wall 이 조용한 오류를 줄여 준다.

오류 학습

조용한 오류 두 가지, 직접 보기

정수 나눗셈 — 100°C가 132°F로

```
integer :: nine = 9, five = 5
real :: celsius, fahrenheit
celsius = 100.0
fahrenheit = celsius * (nine / five) + 32.0
! nine/five = 1, not 1.8

fahrenheit:   132.000000
```

Fortran에서 정수끼리의 나눗셈 9/5 는 1.8이 아니라 1이다. 한쪽을 실수로 바꾸면(real(nine) / five) 212.000000 으로 올바로 나온다. 경고 없이 틀린 답을 내는, 수치 코드에서 가장 흔한 버그다.

반올림 오차 — 0.1을 열 번 더하면

```
real :: total = 0.0
do i = 1, 10
   total = total + 0.1
end do

total         :   1.00000012
total == 1.0 : F
```

정확히 1.0이 아니라 1.00000012가 나오고, 1.0과 같은지 묻는 비교는 거짓(F)이 된다. 0.1이 2진수로 정확히 표현되지 않아 생긴 오차가 누적된 결과다. 실수를 == 로 비교해서는 안 된다.

오류는 빨리 만날수록 좋다.

1장 정리

요약

언어와 표준

- 1957년 등장한 세계 최초의 널리 쓰인 고급 언어이며, 빠른 수치 계산 수요로 최근 인기가 다시 오르고 있다.
- 현대 Fortran은 Fortran 90 이후를 가리키며, 이 책은 Fortran 2018을 기준선으로 삼는다.
- 표준 ≠ 컴파일러 지원 — 최신 표준은 2023이지만 구현은 그보다 늦을 수 있다.
- 수치 계산·HPC에 특화되어, 아폴로의 궤적 계산부터 오늘의 기상 예보까지 쓰인다.

환경과 실습

- 고급 언어로 쓴 코드는 컴파일러가 기계어로 번역해 CPU와 메모리를 제어한다.
- 컴파일러는 gfortran, 실습 환경은 코랩. 런타임이 초기화되면 설치 명령을 다시 실행한다.
- 표준 컴파일 명령: gfortran -O2 -std=f2018 -Wall 파일.f90 -o 실행파일
- 네 가지 오류는 컴파일 → 런타임 → 로직 → 반올림 순으로 조용해서 위험하다.

주의할 함정: %%writefile 과 ! 누락 · 정수 나눗셈(로직 에러) · matplotlib 한글 폰트 깨짐

연습 문제

1. gfortran을 설치하고 gfortran --version 으로 주 버전 번호를 확인하라.
2. hello.f90 을 그대로 작성·컴파일·실행하고 화면 출력과 temp.csv 첫 두 줄을 적어라.
3. do i = -40, 100, 10 을 0, 200, 10 으로 바꾼 뒤 재컴파일 없이 실행하면 결과가 달라지는가?
4. 자기 이름과 학번을 각각 다른 줄로 출력하는 myname.f90 을 작성하라.
5. 반지름 r = 2.0 인 원의 넓이를 출력하라. pi 는 parameter 로 선언한다.
6. 컴파일 명령에서 -o hello 를 빼면 어떤 실행 파일이 생기는가?
7. 직사각형의 가로·세로를 선언하고 넓이와 둘레를 함께 출력하라.
8. parameter 로 선언한 pi 에 값을 다시 대입하면 어떤 메시지가 나오는가? 이유를 설명하라.
9. 화씨 32~212도를 20도 간격으로 섭씨 변환해 csv로 출력하고 그래프를 그려라.
10. 제곱수 수열(0, 1, 4, 9, 16 …)을 csv로 출력하고 파이썬으로 막대그래프를 그려라.

다음 장에서는 프로그램 단위, 선언부와 실행부, 식별자 규칙과 implicit none 의 원리를 본격적으로 다룬다.