Proven C Book

조판 견본 — 이 판의 서식을 한자리에

서식 예시(style specimen)

이 판 · 표지는 cover-body / .cover

rubidus

rubidus@gmail.com github.com/rubidus-api/proven_c_book

표지의 각 줄에는 저마다 이름이 있다 — 제목 `.cover h1`, 부제 `.cover-sub`, 딱지 `.badge`, 판 정보 `.cover-meta`, 지은이 `.cover-author`, 링크 `.cover-links`, 그리고 이 소개글 `.cover-blurb`.

차례 `.toc / toc-row`

머리말 `h2 / heading lv2`

이 자리는 번호가 없는 제목이다. 앞부속(머리말·저작권)과 뒤부속(부록·찾아보기)의 제목이 이 꼴을 쓴다. 본문 문단은 첫 줄을 한 글자 들여쓴다 `p / set par(first-line-indent)`.

저작권과 연락처

지은이 rubidus
연락 rubidus@gmail.com
판 이 견본은 판마다 함께 나간다
이 표의 이름 `metalist / .metalist`

1 장 제목은 이렇게 보인다 `h1 / heading lv1`

이 장의 차례

장 서두의 다섯 칸은 **한 장치의 다섯 얼굴**이다. 모두 `_open-block` 이 그리고, 갈래는 클래스로만 갈린다 — 표지 줄은 `p.dev-label`, 내용은 `div.open-body`. 아래 이름표는 각 칸의 표지 줄 **안에** 붙어 있다.

먼저 알아야 할 것 `prereq() → _open-block · div.dev.open.prereq > p.dev-label`

1장 어딘가 · 기댄 개념 한 줄
5장 다른 곳 · 또 하나

돌아보기 `deepqa() → _open-block · div.dev.open.deepqa > p.dev-label`

인출 문답의 질문은 이렇게 보인다. 굵은 왼쪽 선 아래 가는 선이 이어지는가?

> **답.** 답은 이렇게. 문단이 둘일 때 선이 끊기지 않아야 한다.
> 두 번째 문단이다. 앞 문단과 같은 선 위에 이어져 있어야 한다.

이 장의 필요성과 맥락 `why() → _open-block · div.dev.open.why > p.dev-label`

이 칸이 「이 장의 필요성과 맥락」이다. 표지 줄과 내용 사이에만 틈이 있고, 내용 안에서는 선이 이어진다.

이 장이 끝나면 `organizer() → _open-block · div.dev.open.organizer > p.dev-label`

「이 장이 끝나면」 칸. 서두 넷은 칸과 칸 사이에 가로선이 없어야 한다.

이 장에서 답할 질문 `chapter-questions() → _open-block · div.dev.open.questions > p.dev-label`

1 문답의 질문. 왼쪽 선이 굵은가? `qa / .qa-box`

1.1 절 제목 — 밑줄이 글자에 붙었는가 `h2`

본문이다. 인라인 코드 `<stdio.h>` `code / raw(block: false)` 가 둘레 글자보다 작아 보이지 않아야 한다. 줄간격과 첫 줄 들여쓰기도 여기서 함께 본다. 각주는 이렇게 단다[1] — 문장이 끊기지 않는다.

색인에 실을 낱말은 표시만 남기고 화면에는 아무것도 그리지 않는다 `idx / metadata`.

[1] 각주의 내용 `note / .src-note → 장 끝의 「주」`. 텍스트 판에서는 이 쪽 아래에, 웹판에서는 장 끝에 모인다.

1.1.1 항 제목 — 더 가는 밑줄 `h3`

문. 문답의 질문. 왼쪽 선이 굵은가? `qa / .qa-box`

답. `.qa-a` 선이 가늘어졌고, 질문과 답 사이가 떨어져 있는가?

examples/ch33/fact.c

```c
#include <stdio.h>

int fact(int n)
{
    if (n <= 1) {
        return 1;              /* 바닥: 더 내려가지 않는 경우 */
    }
    return n * fact(n - 1);    /* 자신을 부르되, 더 작은 문제로 */
}

int main(void)
{
    printf("5! = %d\n", fact(5));
    return 0;
}
```

실행 결과

```
5! = 120
```

시연 뒤에도 왼쪽 선이 이어져야 한다.

흔한 오해. "흔한 오해의 제목은 이렇게" `misconception / .dev.device`

내용 칸 `.dev-body` 에도 왼쪽 선이 있어야 한다.

examples/ch33/fact.c

```c
#include <stdio.h>

int fact(int n)
{
    if (n <= 1) {
        return 1;              /* 바닥: 더 내려가지 않는 경우 */
    }
    return n * fact(n - 1);    /* 자신을 부르되, 더 작은 문제로 */
}

int main(void)
{
    printf("5! = %d\n", fact(5));
    return 0;
}
```

실행 결과

```
5! = 120
```

실제 사례. 실제 사례의 제목 `realcase / .dev.realcase`

선 오른쪽에 숨이 있어야 한다. 글이 선에 바짝 붙으면 답답하다.

examples/ch33/fact.c

```c
#include <stdio.h>

int fact(int n)
{
    if (n <= 1) {
        return 1;              /* 바닥: 더 내려가지 않는 경우 */
    }
    return n * fact(n - 1);    /* 자신을 부르되, 더 작은 문제로 */
}

int main(void)
{
    printf("5! = %d\n", fact(5));
    return 0;
}
```

실행 결과

```
5! = 120
```

반례. 반례의 제목 `antipattern / .dev.antipattern`

반례도 다른 장치와 **같은 꼴**이다 — 제목 줄이 굵고, 내용 칸에 가는 선이 이어지고, 선 오른쪽에 같은 만큼의 숨이 있다.

examples/ch33/fact.c

```c
#include <stdio.h>

int fact(int n)
{
    if (n <= 1) {
        return 1;              /* 바닥: 더 내려가지 않는 경우 */
    }
    return n * fact(n - 1);    /* 자신을 부르되, 더 작은 문제로 */
}

int main(void)
{
    printf("5! = %d\n", fact(5));
    return 0;
}
```

실행 결과

```
5! = 120
```

플랫폼 노트. 플랫폼 노트의 제목 `platform / .dev.platform`

다른 장치와 **같은 꼴**이어야 한다 — 점선이나 겹선으로 따로 놀지 않는다.

examples/ch33/fact.c

```c
#include <stdio.h>

int fact(int n)
{
    if (n <= 1) {
        return 1;              /* 바닥: 더 내려가지 않는 경우 */
    }
    return n * fact(n - 1);    /* 자신을 부르되, 더 작은 문제로 */
}

int main(void)
{
    printf("5! = %d\n", fact(5));
    return 0;
}
```

실행 결과

```
5! = 120
```

수학. 수학 상자의 제목 `mathbox / .dev.mathbox`

제목 왼쪽의 줄이 다른 장치의 제목과 같은 굵기인가? 예전에는 여기만 얇았다.

examples/ch33/fact.c

```c
#include <stdio.h>

int fact(int n)
{
    if (n <= 1) {
        return 1;              /* 바닥: 더 내려가지 않는 경우 */
    }
    return n * fact(n - 1);    /* 자신을 부르되, 더 작은 문제로 */
}

int main(void)
{
    printf("5! = %d\n", fact(5));
    return 0;
}
```

실행 결과

```
5! = 120
```

복습 정리

- 복습 정리의 첫 줄 `recap / .dev.recap`
- 둘째 줄

examples/ch33/fact.c

```c
#include <stdio.h>

int fact(int n)
{
    if (n <= 1) {
        return 1;              /* 바닥: 더 내려가지 않는 경우 */
    }
    return n * fact(n - 1);    /* 자신을 부르되, 더 작은 문제로 */
}

int main(void)
{
    printf("5! = %d\n", fact(5));
    return 0;
}
```

실행 결과

```
5! = 120
```

열 하나	열 둘	열 셋
값	값	값

표 1.1 — 표의 캡션 — `dtable / figure.tbl`

1.2 시연 — ★ 여기만 상자를 쓴다 `demo / .demo`

examples/ch33/ternary.c

```c
#include <stdio.h>

int main(void)
{
    int    i = 7;
    double d = 2.5;

    /* 조건 연산자의 타입은 *컴파일 시간에 하나로* 정해진다.
       두 가지가 int 와 double 이면 결과는 언제나 double 이다. */
    printf("sizeof(int)=%zu sizeof(double)=%zu\n", sizeof i, sizeof d);
    printf("sizeof(1 ? i : d) = %zu  (double even when the condition is true)\n", sizeof
(1 ? i : d));
    printf("value  (1 ? i : d) = %.1f\n", 1 ? i : d);   /* 7 이 아니라 7.0 */

    /* 부호가 섞이면 통상 산술 변환이 그대로 적용된다.
       (아래는 컴파일러 경고를 피하려고 미리 맞춰 준 판이다 —
```

```
        맞추지 않으면 gcc 가 -Wsign-compare 로 이 자리를 짚는다) */
    int      neg = -1;
    unsigned one = 1u;
    unsigned mixed = 1 ? (unsigned)neg : one;
    printf("(1 ? neg : one) becomes unsigned = %u\n", mixed);

    /* 문자 상수는 C 에서 이미 int 다 */
    printf("sizeof('a')=%zu sizeof(1 ? 'a' : 'b')=%zu\n",
           sizeof('a'), sizeof(1 ? 'a' : 'b'));

    /* 포인터 쪽 규칙: 한쪽이 널 포인터 상수면 결과는 다른 쪽의 타입 */
    const char *s = "text";
    const char *r = 1 ? s : NULL;
    printf("pointer branch: %s\n", r);
    return 0;
}
```

실행 결과

```
sizeof(int)=4 sizeof(double)=8
sizeof(1 ? i : d) = 8  (double even when the condition is true)
value  (1 ? i : d) = 7.0
(1 ? neg : one) becomes unsigned = 4294967295
sizeof('a')=4 sizeof(1 ? 'a' : 'b')=4
pointer branch: text
```

1.3 참고 문헌 — 인용은 이렇게 적는다

책에는 인용 전용 장치가 없다. 표준과 책을 가리킬 때는 본문에서 이름과 자리를 그대로 밝히고[2], 모아 보일 자리에서는 아래처럼 목록으로 적는다 `list / ul`.

- ISO/IEC 9899:2024, Programming languages — C (C23). 표준 문서.
- Kernighan, B. W. and Ritchie, D. M., The C Programming Language, 2판, 1988.
- Gustedt, J., Modern C, 2판, Manning, 2019.

1.4 찾아보기 `make-index / ul`

[2]예: ISO/IEC 9899:2024 (C23), §6.7.3.1 「형 지정자」.