IT 백두대간 C 언어 편더멘탈
:견고한 프로그램을 위한 기본 원리
한빛미디어

필드 테스트 모범 답안 (2장)

최종 수정일: 2003년 10월 9일

- 이곳을 통해 제공되는 필드 테스트의 답안은 말 그대로 정답이 아닌 모범 답안입니다. 물론 수학 문제처럼 다른 가능성의 여지가 없는 문제도 있지만, 다수의 문제는 다양한 생각이 모두 올바른 답이 될 수 있는 것들입니다. 제시된 답에 대해 의문이 있거나 부족하다고 판단되는 경우, 망설임 없이 저자의 메일 (mycoboco@hanmail.net) 이나 게시판 (http://c-expert.uos.ac.kr) 을 통해 알려주시기 바랍니다. 적절한 의견은 추후 답안에 반영하도록 하겠습니다.

- 답이 생략된 문제는 이곳을 통해 충분한 답을 제공하기 어렵거나, 이미 필요한 방법 등을 본문에서 보이고 독자 여러분이 자신의 환경에서 직접 실습해보길 추천하는 수준의 문제입니다. 다양한 환경에 대한 정보를 나누기 위해, 독자 여러분이 직접 경험한 사실을 메일이나 게시판을 통해 제공해주시면, 이 역시 검토 후 답안에 반영하도록 하겠습니다.

1. C 언어의 표준화 작업을 담당하는 위원회인 ISO/IEC JTC1/SC22/WG14 의 홈페이지는 http://www.dkuug.dk/JTC1/SC22/WG14/ 에서 확인할 수 있다. 또한, C99 표준의 최종 초안인 N869 문서는 http://std.dkuug.dk/JTC1/SC22/WG14/www/docs/n869 를 통해 다양한 포맷으로 구할 수 있다.

   참고로, WG14 위원회에서 공식적으로 공개한 FCD 는 N849 문서이다. 이는 (문서에 독립적으로 번호를 붙이는) SC22 에 N2794 로 등록된 문서와 동일한 것이며, 표준 초안 중 하나인 WG14 의 N794 와는 다른 문서이다. 하지만, 공식적인 FCD 가 N849 라고 해도 실제 대중에게 공개된 마지막 초안은 N869 이기에 그나마 N869 가 초안 중에서 실제 표준과 가장 흡사하다고 할 수 있다.

> **경고!** 책에서도 언급했듯이 표준 문서는 C 언어 자체가 익숙하지 않은 사람들에게는 득보다는 실이 많은 문서이다. 따라서, C 언어에 대한 일반적인 이해가 결여된 상태에서 자신의 프로그램이나 임플리멘테이션의 행동을 표준에 근거하여 해석하려

는 시도는 언어 자체에 대한 심각한 오해를 낳을 가능성이 크다. 더구나 위에서 언급한 표준 초안은 최종 발표된 표준안과 여러 부분에서 유효한 차이점을 갖는다 – 즉, 최종 초안 발표 이후 많은 부분에서 수정이 이루어졌다. 따라서, 초안은 어디까지나 표준의 존재를 확인하는 정도로만 사용되어야 하며, 그 이상의 가치는 없다.

2. 임플리멘테이션이 추가적으로 제공하는 확장은 대개의 경우 해당 임플리멘테이션이 제공하는 (온라인/오프라인) 문서나 도움말을 통해 확인할 수 있다. 예를 들어, gcc 3.2.2 는 http://gcc.gnu.org/onlinedocs/gcc-3.2.2/gcc/C-Extensions.html#C%20Extensions 를 통해 그 내용을 확인할 수 있다. 확장 중에는 표준을 따르는 프로그램에 전혀 영향을 주지 않는 conforming extension 과 유효한 영향을 주는 (따라서, 해당 임플리멘테이션을 표준을 따르지 않는 것으로 만드는) 확장이 있음을 기억하자.

3. 필자가 사용하는 (gcc 가 아닌) 한 임플리멘테이션은 [예제 2-1] 에 대해 다음과 같이 매우 적절한 진단 메시지를 출력하는 것을 확인할 수 있다. 이 임플리멘테이션은 해당 프로그램의 번역 자체를 거절했다.

```
"t.c", line 5: error: missing closing quote
      printf("This should be regarded as an error.
             ^

"t.c", line 6: error: expected a ")"
   Do you agree?₩n");    /* wrong */
   ^

"t.c", line 6: error: unrecognized token
   Do you agree?₩n");    /* wrong */
                ^

"t.c", line 6: error: missing closing quote
   Do you agree?₩n");    /* wrong */
                 ^
```

4. (생략) 잘못 작성된 프로그램은 작성하고 싶지 않아도 작성되는 경우가 많다. :-)

5. 다음과 같은 프로그램은 다수의 임플리멘테이션에서 프로그램이 실행되는 도중에 진단 메시지를 출력하며 종료된다.

```
int main(void)
{
    int i = 0;

    i = 7903 / i;    /* wrong */

    return 0;
}
```

물론, 프로그램의 결과 자체가 정의되지 않기 때문에, 아무 메시지 없이 종료되거나 문제가 없는 것처럼 실행돼도 표준에 위배되는 것은 아니다 – 사실, 프로그램의 "모든" 가능한 실행이 정의되지 않은 행동을 일으키도록 작성되어 있기 때문에, 매우 "똑똑한" (따라서 현실성이 떨어지는) 임플리멘테이션은 이 프로그램의 번역을 거절할 수도 있다.

6. 필자가 주로 사용하는 세가지 종류의 임플리멘테이션 모두는 [예제 2-2] 를 잘못된 프로그램으로 처리해 번역해주지 않았다. 하지만, [예제 2-2] 가 (선언되지 않은 명칭을 사용하고 있는) 이미 잘못된 프로그램이기에 해당 프로그램을 어떻게 다루는지는 표준이 아닌 임플리멘테이션의 자유에 맡겨진다. 따라서, [예제 2-2] 에서 선언되지 않은 명칭을 적절히 처리해 성공적으로 번역해주는 임플리멘테이션의 행동은 충분히 정당하다 – 표준은 표준을 따르지 않는 잘못된 프로그램의 번역을 반드시 실패해야 한다고 요구하지 않는다.

> **추가 문제:** 자신의 임플리멘테이션에서 잘못된 프로그램을 적절히 처리하여 성공적으로 번역해주는 경우를 찾아보자. 또 그러한 임플리멘테이션의 관용이 충분히 현실적인 실용성을 갖는지 생각해보자.

7. gcc 는 [예제 2-3] 에 대해 다음과 같은 진단 메시지를 출력한다.

```
warning: array `a' assumed to have one element
```

이는 경고이며, 해당 프로그램이 올바르지 않음을 의미하기 위해서가 아니라, 프로그램이 (올바르지만) 바람직하지 않은 부분을 가지고 있다는 사실을 알리기 위해서 사용된 것이다. 필자가 사용하는 또 다른 임플리멘테이션은 [예제 2-2] 를 아무런 진단 메시지도 출력하지 않고 성공적으로 번역해 주었다.

그리고 [예제 2-3] 자체는 별로 바람직하지는 않지만 표준이 그 행동을 보장해주는

분명 올바른 프로그램이기 때문에, 다른 문제가 아닌 단지 배열 a[] 의 선언과 관련된 부분으로 인해 그 프로그램을 제대로 번역하지 못하는 임플리멘테이션은 표준을 따른다고 말할 수 없다.

추가 문제: 자신의 임플리멘테이션에서 표준을 따르는 올바른 프로그램을 제대로 번역하지 못하는 경우를 찾아보자.