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

필드 테스트 모범 답안 (3장)

최종 수정일: 2003년 4월 24일

- 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. 어떤 컴파일러가 크로스 컴파일을 지원하는지 여부, 또 지원한다면 어떤 실행 환경을 위한 목적 코드를 생성해 줄 수 있는지 역시 대개의 경우 해당 컴파일러의 (온라인/오프라인) 문서나 도움말을 통해 확인할 수 있다. 예를 들어, 이 답안을 통해 빈번히 언급되는 gcc (3.2.2) 는 다양한 환경으로의 크로스 컴파일을 지원해준다 - http://gcc.gnu.org/onlinedocs/gcc-3.2.2/gcc/Submodel-Options.html#Submodel%20Options

2. 모든 문장이 잘못된 문장이다. 각 문장을 올바르게 수정하면 다음과 같다.
   (1) C가 기반으로 두고 있는 문자세트는 ISO 646 이다.
   (2) A라는 문자의 코드값이 65라해도, B의 코드값은 A의 코드값과 필연적인 관계를 갖지 않는다.
   (3) 1바이트는 8비트 이상으로 구성된다.
   (4) ASCII는 0번부터 127번까지 총 128개의 문자를 정의하고 있다.
   (5) C 프로그램을 이용해 줄바꿈을 하기 위해서는 개행문자의 외부 표현에 무관하게 '\n' 이라는 하나의 제어문자를 출력하는 것으로 충분하다.

3. 어떤 프로그램의 소스 파일이 표준이 요구하고 있지 않은 특별한 문자를 포함하고 있고 그 프로그램의 번역을 맡은 임플리멘테이션이 그 문자를 인식하지 못한다면, 임플리멘테이션인 그 특수한 문자를 자신이 처리할 수 있는 임의의 문자로 바꿔 인식하거나 해당 프로그램의 번역을 거절할 수도 있다. 하지만, 예에서 보인 수직탭 문자는 표준이 반드시 존재해야 한다고 (따라서, 표준을 따르는 임플리멘테이션이 올바르게 인식해야 한다고) 요구하는 문자 중 하나이다. 따라서, 프로그램 소스 내의 수직탭 문자를 임의로 대체하여 개행문자로 인식하는 임플리멘테이션은 표준을 따른다고 볼 수 없다.

4. 1바이트의 크기는 8비트 "이상"이 되도록 요구된다. 따라서, 1 바이트를 32비트로 구성하는 임플리멘테이션 역시 문자형을 포함한 모든 정수형이 32비트 크기를 갖도록 구성하면서 (따라서, signed char, short int, int, long int 가 표현할 수 있는 값의 범위, unsigned char, unsigned short int, unsigned int, unsigned long int 가 표현할 수 있는 값의 범위가 동일해진다) 충분히 표준의 요구를 만족시킬 수 있다.

   물론, 구현자 입장에서는 골치가 아프겠지만, (unsigned) int 와 (unsigned) long int 혹은 (unsigned) long int 만을 값을 표현하는데 32비트 크기를 갖도록 구현하고자 하는 경우, 다른 정수형에 값 표현에는 사용되지 않고 무의미하게 자리만 차지하는 패딩 비트를 도입하여 원하는 바를 이룰 수도 있다.

> **추가 문제:** 처음 언급한 경우처럼 unsigned char 가 표현할 수 있는 값의 범위가 int 가 표현할 수 있는 값이 범위보다 큰 경우를 표준은 분명 허락하고 있다. 하지만, 현실적인 많은 경우에 (특히, 라이브러리 쪽의 지원에서) unsigned char 형이 int 형보다 큰 범위의 값을 표현할 수 있다는 사실은 많은 문제를 낳는다 (따라서, 사실상 표준에는 unsigned char 가 표현할 수 있는 최대값이 int 형이 표현할 수 있는 최대값 이하라는 가정이 숨어 있다고 볼 수 있다). 구체적으로 어떠한 경우가 문제가 될 수 있는지 생각해보자.

5. 바이트는 임플리멘테이션이 제공하는 (비트 필드를 제외한) 다른 데이터형의 기본이 되어야 한다 – 즉, 다른 모든 데이터형을 구성하는 비트 수는 바이트를 구성하는 비트 수로 나누어 떨어져야 한다. 따라서, 대부분의 경우 int 형의 크기로 결정되는 워드의 크기가 35비트고, 바이트의 크기가 9비트인 임플리멘테이션은 구현자가 특별한 편법을 사용해 위의 조건을 만족해주지 못하는 이상 표준을 따르는 임플리멘테이션이 될 수 없다.

> **추가 문제:** 5번 문제에서 설명한 임플리멘테이션이 표준을 따르기 위해 사용할 수 있는 편법을 생각해보자.