교수자를 위한 제언

C 언어 수업 현장에서 가장 어려운 점이 수강생들의 집중력을 끌어들이고 또 유지하는 문제다. 이런 방법을 써도 저런 방법을 써도 큰 차이를 느끼기 어렵지만 그 동안 시도했던 몇 가지 방법을 설명하는 것은 교수자 분들에게 어느 정도 도움이 될 것으로 보인다.

요즘 세대는 텍스트 정보보다는 시각 정보에 익숙하다. 이 책이 수많은 그림으로 채워져 있는 이유도 이 때문이다. 그러나 그것으로 부족할 경우 인터넷 영상물을 활용할 필요도 있다. C 언어에서 다루는 거의 모든 주제가 비디오 클립 형태로 유 튜브에 올라와 있다. 예를 들어 유 튜브에 Bubble Sort라고만 쳐도 수많은 영상이 올라온다. 그 중 도움이 될 만한 것을 수업에 활용해 볼 수 있다. 교수자를 위해 제공한 파워 포인트 슬라이드 파일에 나열된 URL을 활용해 볼 수도 있다. 예를 들어. KOCW(Korea Open Coursewares)와 같은 무료 대학 강의를 활용해 볼 수도 있다. 그러나 이들 강의는 대부분 길게 이어지기 때문에 현장 수업에 직접 활용하기에는 무리가 있다. 따라서 수업 중 이해가 어려울 경우에 한해, 개인적으로 이러한 사이트를 활용하도록 권유하는 것이 바람직해 보인다.

기본적으로 이 과목은 이론과 실습을 병행해야 하는 과목이다. 출석을 자동으로 처리한다고 가정할 때, 50분 수업을 반으로 나누어 25분, 실습 25분으로 하면 간단할 것 같지만 이론 25분을 지루해하는 경우가 많다. 물론 이는 학습 동기가 어느 정도인가에 따라 달라진다. 그러나 대개 손가락이 키보드를 떠나 있는 시간이 길수록 지루해할 가능성이 커진다. 그래서 세운 대안이 이를 다시 세분해서 이론 10분 실습 15분, 다시 이론 10분 실습 15분으로 꾸려가는 방법이다. 개인적인 의견으로는 이 방법이 더 나은 듯하다. 물론 더 세분할 수 있을지는 모르겠으나 이론이나 실습을 더 이상 짧은 시간 단위로 끊기는 어려워 보인다.

대부분 C 언어 수업이 이론과 실습을 병행한다. 그런데 여기에는 두 가지 방식이 있다. 이론을 먼저 설명하는 방식과, 역으로 실습을 먼저 하는 방식이다. 실습 이전에 연관된 이론을 먼저 설명할 경우 학생들은 무엇에 주안점을 두어야 하는지 이해하는 상태에서 실습을 할 수 있다는 장점이 있다. 그러나 자칫하면 이론 설명이 길어지기 때문에 그만큼 학생들의 집중력을 잃기도 쉽다. 개인적 소견으로는 이론이 개입된 비교적 어려운 예제를 실습할 때에는 이론을 먼저 설명하는 것이 나아 보인다. 그러나 그렇지 않을 때에는 오히려 실습을 먼저 하는 것이 나아 보인다. 일단 겪어보고 나서 생기는 의문에 대해서 설명할 때에는 이론 설명이 그렇게 늘어질 필요가 없기 때문이다. 이는 일단 예제를 실습한 이후에 자막에 띄워 놓고 소스코드를 해설하면서 이론 설명을 가하는 방법이다.

한 가지 더 활용해 볼 수 있는 방법이 조별 수업이다. 3인 1조 정도로 편성하여 챕터 마지막에 있는 과제를 하나 선택하여 ‘수업 도중’ 조별로 프로그램을 완성하여 제출하도록 하는 방법이다. 이 역시 효과가 좋아 보인다. 사람들은 말하면서 배운다. 조원끼리 서로 대화하는 과정에서 자신이 아는 것이나 안다고 생각하는 것들을 다시 확인하고 검증할 기회를 갖기 때문이다. 이 방법이 중요한 또 다른 이유는 수업 분위기를 바꿀 수 있기 때문이다. 강의 일변도의 수업이 아니라 서로 간에 자유로이 대화할 수 있는 능동적 수업이기 때문이다.

|  |  |  |  |
| --- | --- | --- | --- |
| 장 | 예제 수 | 과제 수 | 진도 표 |
| 1. C 언어와 컴파일러 | 1 | 8 | 1, 2, 3 주 |
| 2. 변수와 산술 연산 | 16 | 10 |
| 3. 대입문과 연산자 | 12 | 7 |
| 4. 함수 I | 5 | 6 |
| 5. 선택 구조 | 15 | 15 | 4 주 |
| 6. 반복 구조 | 17 | 22 | 5, 6 주 |
| 7. 디버깅과 소프트웨어 설계 | 8 | 3 |
| 8. 함수 II | 23 | 15 | 7, 8, 9 주 |
| 9. 배열 | 2 | 10 |
| 10. 포인터 | 11 | 12 |
| 11. 포인터와 배열 | 19 | 16 | 10, 11 주 |
| 12. 문자열 | 33 | 16 | 12, 13 주 |
| 13. 구조체 | 16 | 10 | 14 주 |
| 14. 파일 입출력 | 8 | 7 | 15 주 |
| 15. 범용 프로그래밍 | 5 | 9 |

\* 주별 강의 진도(권장 사항)

장 별 예제와 과제 수를 요약한 표다. 물론 난이도에 따라서 다르겠지만 이론 설명을 빼면 50분 수업에 대략 서너 개의 예제를 실습할 수 있다. 3학점 과목이라면 이는 대략 주당 10개 내외의 예제를 실습할 수 있음을 뜻한다. 예는 그러한 계산에 의해 맞추어본 권장 진도표다. 15주에 맞춘 이유는 교육부 규정에 의하면 현재 대학 강의가 15주로 되어있고 그 이상 강의하는 것은 교수자의 재량이기 때문이다. 물론 1주 단위로 한 장씩 나갈 수 있으면 좋지만, 현실적으로 그렇게 하기가 쉽지는 않다. 때에 따라서는 한 주 분량이 넘는 수의 예제를 다루어야 할 때도 있다. 그 경우 당연히 몇몇 예제는 건너뛰어야 한다. 어떤 예제를 건너 뛸 것인 역시 교수자의 주관적 판단에 달려 있다. 유사한 예제를 건너뛸 수도 있고, 상대적으로 난이도가 높은 예제를 건너뛸 수도 있다.

과제에 대해서도 언급할 필요가 있다. 필자 개인적으로는 장 별로 서너 개 정도의 과제를 제출하도록 요구한다. 하지만 해당 장이 중요하다고 생각할 경우 과제 수를 더욱 늘려 요구하기도 한다. 물론 과제 수에 대해서는 학생들의 눈높이를 감안해야 한다. 그러나 과제 수를 무작정 늘리기가 어려울 때가 많다. 이전 주에 제출한 과제는 가급적 금주에 해설을 해 주는 것이 효과적이지만 이 역시 진도를 나가는 데에는 상당한 시간 부담으로 작용할 때가 많기 때문이다. 따라서 모범 답안은 홈페이지에 올려 주고, 수업 중 해설은 문제가 될 만한 부분만 가볍게 건드리고 지나가는 편이 나은 것으로 보인다.

과제물의 경우 지시사항에 대한 질문이 많이 올라올 수 있다. 기본적으로 이 책의 과제에는 일정한 전제가 있다. 특수한 모든 경우의 입력을 처리할 수 있는 프로그램을 작성하라는 것이 아니라, 과제가 요구하는 일반적인 경우의 입력에 대해서만 처리할 수 있는 프로그램을 작성하라는 것이다. 따라서 샘플 입출력이 나타나 있지 않은 과제에 대해서는 스스로 가정하고 프로그램을 작성하면 된다. 필요하다면 어떤 가정을 했는지 주석을 달아도 좋다. 과제에 추가된 힌트는 대개 알고리즘을 설명하기 위한 것이다. 개인적으로 다른 알고리즘을 사용하겠다면 그렇게 하도록 허용해도 전혀 문제될 것이 없다. 함수 원형도 마찬가지다. 과제에서 제시한 함수 원형과는 다른 함수를 사용하겠다면 그 역시 문제될 것이 없다.

그러나 밑줄 친 부분을 채우라는 형식의 과제는 가급적 그대로 따라하도록 요구할 필요가 있다. 이러한 형식의 과제를 요구한 데에는 이유가 있다. 일단 상대적으로 난이도가 높은 과제의 경우 그런 형식을 취함으로써 프로그램을 작성하는데 상당한 도움이 된다. 가이드라인 역할을 하기 때문이다. 그러나 그보다 더 큰 이유는 타인이 작성한 프로그램을 이해해 보라는 것이다. 타인의 프로그램 즉, 타인의 논리를 이해한다는 것은 자신이 직접 프로그램을 작성하는 것보다 훨씬 어려운 일이다. 그러나 그러한 과정을 통해 왜 이런 논리로 작성했을까, 더 나은 방법은 없었을까 하는 비판적 사고를 키울 수 있다.