**프로와 아마추어의 차이는 ‘자신을 평가할 줄 아는 능력'의 유무에 있어요**

“가장 좋아하는 프로그래밍 언어는 C언어에요. C를 접한 이후로는 거의 완벽한 언어라고 생각하고 있어요. low-level에서 원하는 것을 다 할 수 있어서 좋아요. 우리나라는 C보다는 자바, 자바스크립트, Python 등이 대세인데, 요즘에는 언어와 라이브러리의 경계가 희미해지면서 점점 지저분해진다고 느껴져요. 반대로 C는 언어가 필요한 기능에 실제 하드웨어를 고려한 포인터 같은 기능들만 추가해서 상대적으로 깨끗해요.”

“다른 언어들이 C에 비해 쉽다는 의견에는 동의 못 하겠어요. 예로 들어보면 Python이 배우기 쉬운 이유는 C의 진입장벽인 포인터가 없고, 문법이 간단하다는 것 외에는 별거 없어요. 물론 파이썬은 자동으로 자료형을 정해줘서 편리하긴 해요. 하지만 자료형이 엉키면 나중에 사고를 칠 수 있어요. 그걸 겪고 나면 자료형이 잘 정의된 C 같은 언어가 낫다고 생각하게 돼요. 자료형이 명시적으로 나타나니 읽기도 쉽고요. 그리고 진짜 문제는 로직에서 나오는 에러거든요. 이건 언어가 다르다고 해서 피할 수 있는 것도 아니죠.”

“코드는 읽기 쉬운 코드가 최고예요. 코드는 나만을 위한 것이 아니라 모두를 위한 것이거든요. 오픈 소스든 회사의 프로그램이든 다른 사람이 나의 코드를 보고 리뷰를 하기 때문이죠. 요즘 언어들이 간단하지만 추상적인 문법으로 생산성을 높이려고 하는데, 제 생각엔 아름답지 않은 모습 같아요. 코드는 읽는 사람이 고민하게 만들면 안 되거든요. 그래서 쓰기 쉬운 코드가 아니라 읽기 쉬운 코드를 지향해야 합니다. C언어가 좋은 이유도 그 이유고요.”

“좋은 개발자가 되려면 해결 방법을 찾고 설명하는 역량이 필수적이에요. 이때 설명에 사용하는 용어가 매우 중요해요. 소프트웨어를 학습할 때 책에서 접한 용어들로 설명해야 다른 개발자들과 원활하게 소통할 수 있거든요. 제가 42학생들한테 ‘문제를 해결할 때 그 문제와 연관된 주제와 이론을 공부해야 한다’고 많이 얘기하는 이유 중 하나죠.”

“예시를 몇 개 들어보면, 42에 웹 서버를 만드는 과제가 있잖아요? 웹 서버를 만들기 위해서는 ‘네트워크 프로토콜이 무엇인지’, ‘패킷이 어떻게 동작하는지’ 등을 알아야 하고, 셸을 만들기 위해서는 ‘프로세스가 무엇이고 어떻게 통신이 일어나는지’, 그리고 ‘시스템 콜이 무엇인지’ 알아야 하는데, 책을 보지 않으면 모를 수 밖에 없어요. 프로그램이 담고있는 기술에 대한 질문을 던졌을 때 단어조차 처음 들어봤다는 듯 반응하면 내용은 모르는 채로 코드를 가져다 썼다는 인상을 줄 수 밖에 없게 돼요. 선수들이 쓰는 용어를 배우기 위해 책을 공부해야 합니다.”

“프로와 아마추어의 차이는 ‘자신을 평가할 줄 아는 능력'의 유무에 있어요. 시간을 어떻게 쓰는지 평가하는 것도 그 일부죠. 학창 시절 방학이면 동그란 계획표를 만들어 하루의 일과를 채워 넣잖아요. 지금도 이런 걸 해야 한다고 봐요. 24시간을 어떻게 썼는지 매일 저녁 적어보세요. 당연히 처음에는 24시간이랑 안 맞아요. 시간을 낭비한 듯한 느낌이 들죠. 하지만 이 작업을 계속하다 보면 어느새 하루를 어떻게 썼는지 딱 맞출 수 있어요. 허투루 쓴 시간이 없어지는 거예요.”

“해야 할 일에 대한 소요 시간도 예측해보세요. ‘어떤 함수를 짤 때 몇 시간이 걸릴 것이다’, ‘어떤 기능을 구현하는 데 며칠 걸릴 것이다' 이런 식으로요. 일을 마친 뒤에는 실제 얼마나 걸렸는지 적어뒀던 예측 시간과 비교해보세요. 이 두 가지 모두 5분도 안 걸리는 간단한 일이에요. 그렇지만 반복하다 보면 나의 예상과 실제 결과가 그럭저럭 비슷해지고 나를 평가할 줄 아는 능력이 생겨요.”

“그 ‘끝판왕'은 내가 택해왔던 공부 방법을 바꿀 줄 알게 되는 겁니다. 운동선수들이 슬럼프에 빠졌다가 극복하고 난 뒤에 하는 얘기가 있어요. ‘처음부터 다시 배웠다'고 하죠. 소프트웨어 분야도 마찬가지예요. 스스로 평가해서 예전부터 해왔던 방식을 버릴 줄 아는 게 능력이에요.”

“뭔가를 진정성 있게 한다는 것은 그냥 시간을 많이 쓰는 걸 의미해요. 어른으로서의 부탁인데 젊은 시기에 공부에 시간을 정말 많이 썼으면 좋겠어요. **일주일에 적어도 50~60시간 정도는 코딩하는 데 써야 실력이 늘어요.** 이 얘기를 굵은 폰트로 써주세요. 굵은 폰트로 써달라고 얘기한 것까지 써주세요. 그 정도는 해야지 나중에 봤을 때 ‘어 코딩 좀 하네’ 그런 느낌이 나요. 어설프게 해서 이룰 수 있는 건 생각보다 별로 없어요.”

<사진>

“취업할 때는 내가 들어가고 싶은 회사에 대해 잘 알아야 해요. 예를 들어, '업의 본질이 무엇인지', '소프트웨어를 얼마나 중요시하는 곳인지', '개발자가 성장을 할 수 있는 곳인지' 같은 것들이요. 하지만 이것들은 DART 공시나 재무제표에 나와있지 않은 내용이죠.”

“최근 들어 다시 활발히 열리고 있는 박람회에 가는 게 이런 점을 알아볼 수 있는 한 방법이에요. 관심 있는 분야 회사의 부스에 방문해서 '이런 거 만들 때 소프트웨어 필요하지 않아요?', '뭐로 개발하세요?' 등 미친 척하고 질문해보는 거죠. 그곳에 있는 사람들은 온종일 질문을 고대하고 있기 때문에 영업비밀까지 이야기해 줄 수도 있단 말이에요. 그리고 비슷한 업종의 다른 회사 부스에 가서 방금 들은 이야기를 이용해 다시 질문해보는 거예요. '다른 곳에서는 이렇게 개발하는데 여기는 어떻게 하세요?'라고 이야기하면 엄청 있어 보이잖아요. 대화가 끝나고 나면 '제가 소프트웨어를 배웠고 인턴을 가고 싶은데 필요하면 연락해주세요' 하고 명함을 드리는 거죠. 시장에서 내 능력을 사용할 기회를 찾는 건 굉장히 중요한 일입니다. 관심사를 계속 이야기하고 그런 영역에 있는 사람들을 만나는 노력을 해야 하는거에요.”

“소프트웨어 시장은 분야 전체가 한 회사 같아요. 한 회사에서 2~3년 있으면서 배움에 한계가 느껴지면 다른 회사로 옮겨가면 돼요. 비록 회사에 잘 이끌어 주는 사람이 있고 후배들이 똘똘하다고 하더라도, 갈증이 생기면 이직해야 해요. 월급이 깎인다고 하더라도요. 회사를 옮기는 것에 대해 두려워하지 않는 게 중요해요. 프레임워크, 클라우드 등 같은 스택을 사용하더라도 다른 도메인의 데이터를 다루면 엄청난 챌린지로 느껴질 거예요. 도메인이 다르면 데이터가 만들어지고 처리되는 과정이 다 다르거든요. 이런 디테일을 경험하면 실력이 굉장히 높아질 겁니다.”