프로그래머가 AI 코더 열풍에도 살아남을 수 있는 이유

박 건

인공지능이 프로그래머의 일을 대체할 수 있을까? 이런 고민은 현재 AI가 할 수 없는 일을 고려할 때 섣부른 것이라고 할 수 있다. 지금은 AI를 프로그래머가 잘 이용할 방법을 생각할 때다.

1990년대에 큰 번역 붐이 일었다. 한국의 세계화로 인해 수요가 폭증하고, 주부나 학생이 아르바이트로 참여할 수 있는 매력적인 직업이라며 통신 강좌와 학원의 광고가 잡지에 가득했다. 그러나 머지않아 기계 번역 기술이 발달하며 기업은 간단한 문서부터 하나씩, 기계 번역에 일을 맡기기 시작했다. 90년대의 희망과는 다르게, 번역가의 수요는 점차 줄어들고, AI 번역이 발전한 오늘날엔 번역이라는 직업이 사라질 것이라는 위기론마저 나오고 있다. 그렇다면 오늘날 번역가는 무엇을 하고 있을까? 번역가라는 직업은 기계가 따라잡지 못하는 자신의 번역 전문 분야를 기르거나, 기계가 처리하지 못하는 미묘한 의미 전달과 맥락 조율, 그리고 단어 창조를 담당하는 역할로 차차 재정립되고 있다.

오늘날 코딩에서 전례 없이 많은 인력이 양성되는데, AI 코더의 발전으로 그 인력이 갈 곳이 없다는 하소연이 들려오고 있다. 흡사 과거의 번역계 이야기를 보는 것 같다. 그러나 번역가는 살아남았고, 그 이유는 사람이 이해하기 쉬운 글을 만드는 데 여전히 번역가의 손길이 필요하기 때문이다. 프로그래머도 마찬가지로 사람이 이해하기 쉬운 프로그램을 만들기 위해 살아남을 것이다.

AI 기반 프로그래밍은 웹페이지의 간단한 버튼 생성처럼 기존에 전문가를 찾거나 메뉴얼을 뒤져야 했던 일을 손쉽게 처리하게 해준다. 그러나 거대한 시스템 전반에서 안전성과 신뢰성을 보장하며 유지 · 보수 가능한 코드를 만들어내는 일은 여전히 쉽지 않다. 이 신뢰성과 유지 보수성을 위해 프로그래머는 코드의 각 부분이 어떤 역할을 하는지 명확히 정해 놓고, 부분의 디테일에 집중하지 않아도 전체를 이해할 수 있도록 각 부분이 요약 (abstraction)된 코드를 작성한다. 그러나 AI는 프로그램 전체를 이해하고 코드를 수정하는 것이 아니라, 부분을 설계하는 데 그치고 있다. MIT의 찬두리 (Chanduri) 교수님이 지적하듯, 딥러닝에 낙관적인 사람조차도 언어 모델이 복잡한 운영 체제를 곧 설계할 것이라고 믿지 않는다. 결국 AI가 단순한 반복 업무를 효율화한다 해도, 최종적으로 프로그램이 지향하는 문제를 명확히 정의하고, 그에 적합한 해결법을 찾아내며, 이를 사람이 이해하기 쉽게 조정 · 개선하는 역할은 여전히 인간 프로그래머에게 남아 있다.

우리는 인공지능 열풍 이전의 고전적 프로그래밍에서도 AI와 비슷한 게 있었음을 기억할 수 있다. 바로 인터넷 검색이다. 오래 전부터 "프로그래머의 친구는 스택 오버플로우 (Stack Overflow)라는 포럼, 혹은 인도인 프로그래머의 유튜브"라는 농담이 있을 정도로, 각 기능을 구현하는 방법은 충분히 검색만으로 알아낼 수 있었다. 그러나 그 때도 프로그래머에게 요구되는 것은 프로그램을 만들어서 풀고 싶은 목표가 정확히 어떤 문제로 치환되는지 알고, 좋은 해결법을 찾은 뒤, 이걸 우리 문제에 맞게 변형하는 것이었다.

결국 프로그래밍의 미래는 AI와 인간 프로그래머 간의 협업에 달려 있다. AI가 단순하고 반복적인 코딩 태스크를 처리하는 동안, 인간은 코드를 개념적으로 정리하고, 어디를 어떻게 수정할지 AI에게 명령하며, 궁극적으로 시스템 전반을 지휘하는 고유한 역할을 담당하게 될 것이다. 이런 상호 보완적 관계를 통해 우리는 복잡한 소프트웨어 생태계를 더 손쉽게, 이해하기 쉬운 모습으로 발전시켜 나갈 수 있을 것이다.