「AI時代のプログラミング教育」

東京理科大学　創域理工学部　情報計算科学科　准教授　松澤（まつざわ）　智史（ともふみ）

1. 導入

2022年末に登場したChatGPTは、人間らしい文章・会話を生成するシステムとして大きな衝撃を与えた。しかしながら、この時点の生成AIは、矛盾や平然とした嘘（ハルシネーション）を含むなど、その出力には吟味が必要なシステムであった。2023年～2025年は生成AIが大きく進化を遂げた時代である。もはや現在の生成AIは、自然言語処理学者を驚愕させたそれらしい文章生成器だけに留まらず、高度な論理的思考が必要な解答や、ソースを元にした調査、文章以外のコンテンツの生成など、ある程度の専門性が必要な作業を代替で行える存在にまで成長した。その結果、産業での利用も一気に拡大し、プログラミングにおいても現在のエンジニアは（一部を除き）AIを活用した開発が主流となっている。2023年のCursor、2024年のDevin、2025年のClaude Codeなど、AIコーディングの活用方法やツールは目まぐるしい速さで進化している。ただし、ここで注意すべきは、AIコーディングを評価している方や上手に活用している方々が、AIを活用していないプログラミングを学習してきた人材であるという点である。現在のエンジニアが積極的にAIコーディングを使うべきか否かはこの場で論じない。しかし、この変化は、プログラミング教育の現場、特に教員に対し、①AIコーディングが普及した現在、プログラミング教育は必要か、②必要である場合、AIをどこまで利用させるか、という根本的な問いを突きつけている。本稿は、これらの問題に対する筆者の見解と実践方法を提示することで、劇的な変化を続けるプログラミングに関わる教育現場の関係各位の一助となれば幸いである。

2. プログラミング教育の必要性

筆者は学生から「生成AIが普及してきて、頼めばAIがコードを生成してくれるのに、プログラミングをわざわざ学ぶ必要があるのか？」という質問を受けたことがある。「魚を与えるのではなく魚の釣り方を教える」という教育の格言は、自立した問題解決能力の重要性を示している。しかし、現在のAIは、「いつでも魚を釣ってくれる非常に優秀な釣り人」のように見え、頼む側は釣り方を学ぶ必要がないように思えてしまう。この認識こそが、「AIがコードを生成するのになぜ学ぶ必要があるのか？」という学生の質問の背景にある。プログラミング教育を「学ぶ必要がある」とする理由を教育機関のレベルに応じて考えてみる。中学・高校といった中等教育機関での目的は、プログラミング教育を通じて物事の考え方や思考力を鍛える、工学的なトライアンドエラーの体験をする、というようなある程度万人向けの理由がある。プログラミングでエラーが出た際の原因の絞り込み方や、コードの修正・実行・結果を確認して再び修正するPDCAサイクルの体験など、プログラミング以外への応用例も少なくない。大学のような高等教育機関では、情報といった専門分野を学ぶ上での利点がある。これらの理由は講師の考え方や講義目的によって多様であるが、筆者は以下の見解に基づき講義・演習を行っている。

- ●エンジニアを目指す人は、AI利用を前提とする開発においても、コードが動かない原因や修正方針の指示など、プログラミングへの理解は必要である
- ●コンピュータサイエンスを深く学び研究する人は、アルゴリズムや計算量という知識がなければ、提案する新しい手法の良し悪しも判断できない

そのため、正常に動作するコードを事前に用意して解説をするスタイルは採用せず、コードはその場でコーディングし動作解説を行う実演形式で実施している。その場でコーディングするためエラーが出ることも多い。むしろ、エラーが出るコードを書くことが目的であり、それによって対処方法を実演して見せることができる。特にプログラミング初学者に対して、エラーは出るものである、プログラミングとはエラーを修正することである、という実態を見せることで、エラーへの抵抗感や苦手意識を取り除く効果にもつながる。

3. プログラミング教育におけるAIの活用

まず一つの方法として、AIの使用を禁止してプログラミング教育を行う方法を考える。学生がAI禁止

の指示を守るかどうかはさておき、AI を使わせないプログラミング教育が実施できれば、これまでの変わらない人材が育成できる。実際の現場で AI を有効活用している人材が AI なしの教育を受けてきた事実は、「AI を禁止する教育」が将来の活用を妨げない根拠となり得る。しかし、AI の進化が続く中で、この方針は「AI を使わせる教育」との比較検証が必要である。

最近、興味深い研究報告[1)]が出ており、その内容の一部を紹介する。この文献では米国の大学生 1000 人以上を対象にした調査を行い、学生が AI をアカデミックな支援を求める際、2 つの異なる利用行動があると述べている。1 つ目は Executive Help-Seeking（実行的な支援の追求）で、先ほどの魚を釣る行為を AI に行ってもらう利用方法である。2 つ目は Instrumental Help-Seeking（道具的・手段的な支援の追求）である。これは、概念やプロセスをよりよく理解し、自力で問題を解決できるように AI を利用する方法であり、魚の釣り方を覚えるために AI を活用する利用方法に相当する。以下に、プログラミング教育における Instrumental Help-Seeking の具体例を列挙する。

- ●デバッグ・エラー解決：「このエラーの原因と修正方法を、1 つずつステップで説明してください」
- ●概念の明確化：「○○を引き起こすメカニズムを教えてください」
- ●コードレビューと改善：「このコードをパフォーマンス、可読性、セキュリティの観点からレビューし、改善点を提案してください」
- ●異なる実装方法の比較：「○○と△△の実装例と、それぞれの計算量（時間複雑度）がなぜそうなるのかを比較して説明してください」

さらにこの文献では「講師が AI の深い利用方法を説明して推奨している場合、学生は Instrumental Help-Seeking を行う傾向が高くなる」という興味深い結果が示されている。つまり、推奨する使い方とその理由や効果、推奨しない使い方とその理由や影響などを、事前に講師が学生に細かく指示を出すことによって、概念やプロセスをよりよく理解するような使い方に誘導することが可能であると述べられている。プログラミング教育で AI の使用を禁止したとしても、今後の学生は、学業においても様々な AI を活用しながら学習を進める時代になるであろう。そのような時代の学生は目的を即時解決するための Executive Use と学習のための Instrumental Use を上手に切り替えて活用する能力が求められる。プログラミング教育には、正解となるプログラムがあり、動かないプログラムには明確な理由があり、変更した結果・効果もすぐに実験して確かめることができる。このような観点から、プログラミング教育は非常に Instrumental Use の方法を学ぶ上でも学びやすい教育コンテンツだと考える。今後のプログラミング教育では、アルゴリズムやプログラミング言語仕様などのこれまでのコンテンツの他に、Instrumental Use の方法を学ぶ時間を設けて、AI を活用する学び方を修得することが必要になるかもしれない。

筆者は 2025 年前期に開講したプログラミング演習科目で、AI の自由な使用を許可（ただし、課題のプログラム生成は禁止）し、その効果を観察した。結果、例年と比べ TA（大学院生）への質問が大幅に減少した。この観察から、学生がエラーの原因特定などを AI に問い合わせる、自律的な Instrumental Use をすでに開始しており、講義内容を超えた独自学習を可能にしていることが示唆された。理解が早く、講義・演習の進度を遅いと感じる学生に対する対処（反対に早いと感じる学生にも可）としても非常に有用である。AI を用いるデメリットは、前述の本質的な学習を阻害する Executive Help-Seeking を行ってしまう場合と、教員側が成績評価をする際に、AI 使用による不当な評価である。後者は教員側の評価体制の問題であり、学生の能力を高める術として存在する AI 活用を禁止する理由にはならない。また、AI の正しい活用方法を説明した上で Executive Use を行う学生は、ほぼ不当評価目当ての活用であり、やはりこちらも評価側の問題改善で対処可能なデメリットであると考える。

4. まとめ

結論として、現時点で教員側の考えるべき問題は様々あるが、学生に AI 利用を禁止する積極的な理由は見当たらず、むしろ正しい使い方（Instrumental Use）を明示し、積極的に使用させるべきであると考える。しかしながら様々な潜在的な問題、実施して改めて判明する問題などは存在する可能性も高く、プログラミング教育に携わる関係各位のフィードバックをいただけると幸いである。

参考文献

1）Aguilar, S. J., Nye, B., Swartout, W.R., Macias, A., Xing, Y., & Xiu, R.（2025, August）. Fostering Critical Thinking in the Age of AI（Report）. USC Center for Generative AI and Society. https://doi.org/10.35542/osf.io/wr6n3_v2