PP-Checker 1.5: LLMとの協働により標準出力を含むプログラムを評価する半自動採点システム

関口 祐豊[1,a)]　中村 聡史[1,b)]

概要：本稿では，標準出力を含むプログラミング課題における採点精度向上を目的とし，従来の半自動採点システム PP-Checker を拡張する手法を検討した．この手法では，プログラムの構文の評価に加えて，ランダム性や動的要素を含む標準出力結果を大規模言語モデル（LLM）へのプロンプトとして組み込むことで，多様な課題への対応力を強化した．提案手法をプログラミング演習講義における 13 個の課題で検証した結果，従来手法に比べて採点精度が向上する傾向にあることが確認された．さらに，この結果をもとに，提案手法を統合した PP-Checker 1.5 を講義内で運用し，ユーザの負荷にならず，精度向上につながることを確認した．

1. はじめに

プログラミング演習講義では，学生が限られた時間内に提示された複数の課題に取り組む形式が広く採用されている．第 2 著者が担当する明治大学総合数理学部先端メディアサイエンス学科の 1 年次の必修講義であるプログラミング演習（春学期・秋学期それぞれ 100 分 2 コマ）でも，毎講義 4〜6 問の課題を提示し，学生は与えられた課題に取り組んでいる．講義の課題は各単元に相当するものを設定することになるが，単に習ったものを使うだけの課題の場合，その対象に興味を持ちづらいため，本学科ではプログラムとして課題を解くだけでなく，学生が課題を通じて何らかの驚きを覚えたり，知見を得ることができるような意味のある課題を設定すること（ガチャの確率やモンティホール問題，航空機の座席問題やモンテカルロ法，リサージュカーブや錯視など）を意識して課題を作成している[*1]．

このような課題提示型の講義では，学生の学習意欲の向上に課題の迅速な採点が重要であることが知られている [1]．しかし，TA や教員の人数に対して学生の人数が多い場合，随時発生する質問対応を行いつつ，全学生の課題採点を行うのは容易ではなく，採点が滞りがちになる [2,3]．質問対応の数を減らすため，本学科ではタッチタイピング型予習システム typing.run [4] を開発して基礎的なタイプミスがなくなるような工夫を行い，長期的に運用するだけでなく他大学にも導入したり，オンラインで質問応答を支援するシステム askTA [5] の開発および運用を行ったりしてきたが，依然として迅速な採点は難しいという問題があった．

迅速な採点を行うための手段のひとつとして自動採点があり [6]，AtCoder などの競技プログラミングでも自動採点が導入されている．ここで従来の自動採点手法は，正解が一意に定まる問題に適した設計がなされているため，インタラクティブな要素を含むプログラミング課題には対応が容易ではない．また，インタラクティブな課題だけでなく，長い実行時間を必要とするプログラムの場合は自動採点でもコンピュータリソースと計算時間を必要とし，標準出力が確率的挙動やランダム性を伴うなど出力に曖昧性が生じるものの場合はそもそも自動採点が容易ではない．さらに，課題ごとに専用のテストコードを作成することは教員にとって大きな負担となるため，講義運営上現実的ではない．

大規模言語モデル（LLM）の登場は，これらの問題を解決する一つの方法として期待でき，プログラミング演習講義支援の研究においても，アルゴリズムやデータ構造に関する課題に対し，正確なフィードバックを生成することを目的とした研究が進められている [7,8,9]．しかし，こうした研究では，課題に対する正確な解答やエラー修正に焦点が当てられており，直接的な解答を学生に提供することで，学生の自発的な問題解決能力を阻害するリスクが指摘されている．特に，ChatGPT [10] をはじめとする直接的な回答を示してしまう LLM を搭載した AI ツールは，教育機関での使用に関して懸念が高まっている [11]．我々はこれまで，こうした問題を踏まえ，LLM を活用しつつも具体的な正答は提示せずにフィードバックを提供する半自

[1] 明治大学
[a)] yuutosekiguchi@gmail.com
[b)] satoshi@snakamura.org
[*1] https://lecture.nkmr.io

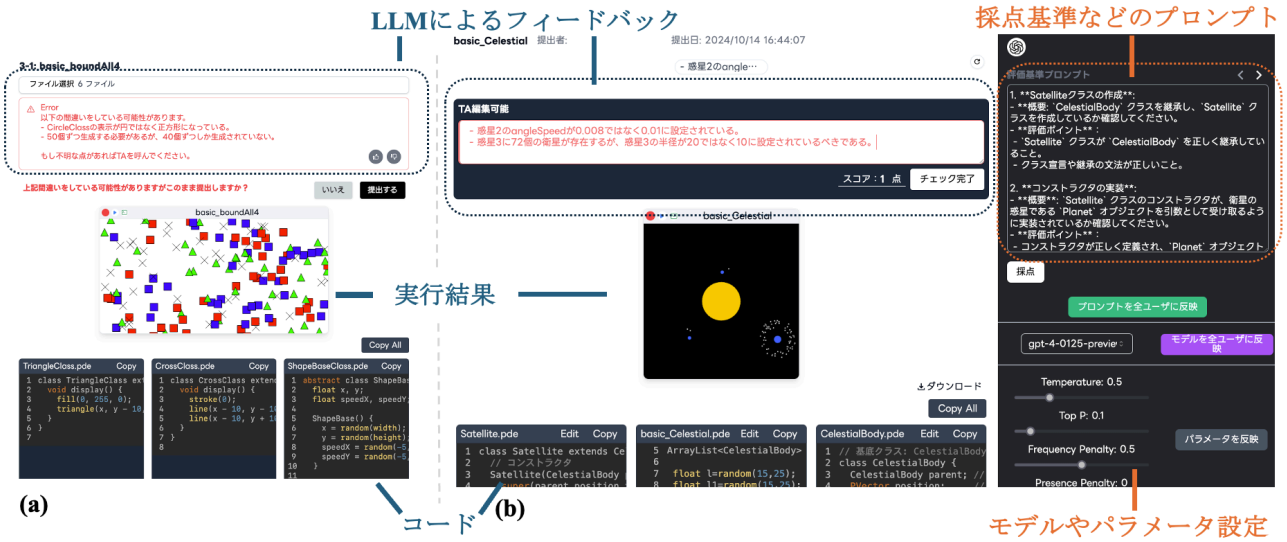

図 1　PP-Checker の課題提出画面（a）と手動チェック画面（b）

動採点システム PP-Checker（図 1） [12] を実現してきた．しかし，プログラムが構文的には正しいものの，期待される標準出力を生成しない場合や，ランダム性を伴う出力の場合に，適切なフィードバックを返せていないという問題があった．

そこで本稿では，PP-Checker をアップデートし，プログラムの構文や整合性のみを評価していた仕組みに加え，標準出力の実行結果も評価に組み込む PP-Checker 1.5 を提案する．PP-Checker 1.5 により，静的な標準出力を伴う課題はもちろんのこと正解が一意に定まらない標準出力を含むプログラム課題においても採点精度が向上するという仮説を立て，その検証を行う．具体的には，我々の所属する学科の講義であるプログラミング演習 I（100 分 2 コマ，12 回）における課題のうち，標準出力を伴う 13 個の課題を対象に提案手法と従来の PP-Checker による採点精度を比較する．また，実際に講義で運用することでその利用可能性について検証を行う．

2.　関連研究

2.1　自動採点に関する研究

自動採点は課題採点業務を効率化する方法の一つであり，フィードバックの迅速化だけでなく，学生の動機付けやスキル向上にも寄与することが知られている [6].

Krusche ら [13] は，大学やオンラインコースにおけるプログラミング演習を対象に，解答をテストケースで評価し即時にフィードバックを提供するシステムである ArTEMiS を提案した．このシステムは，学生がフィードバックをもとに解答を改善する学習プロセスを支援するとともに，教員の負担を軽減しつつ大規模なクラスにも対応可能であることが示された．また，Queirós [14] は，プログラミング演習を支援する自動化されたティーチングアシスタントシステムである PETCHA を開発した．このシステムでは，教師が設定したテストケースに基づいて学生のプログラムを評価する仕組みや，エラーの種類に応じて事前定義されたフィードバックを提供する仕組みを備えており，効率的な学習プロセスを実現している．新田ら [15] は，テストケースを用い複数言語に対応したオンラインプログラミング試験システムである track を実装した．

これらの先行研究は，自動採点システムがプログラミング演習における効率的な学習支援や教員の負担軽減に有効であることを示している．一方で，テストケースを用いた静的評価が主流のため，一意解のない標準出力を伴う課題への対応が難しい場合がある．また，複雑な課題の場合はテストケースの作成が困難である．本研究では，従来のシステムでは扱いにくい一意解のない標準出力が伴う課題や複雑な課題に対しても，教員の手間を減らしつつ高精度な自動採点を可能にする枠組みを構築する．

2.2　LLM の活用

LLM によるフィードバックは，従来のテストベースのフィードバックを補完する形で効果的であることが示されているが [16]，誤情報の提供やフィードバックの正確性には改善の余地がある．

Abolnejadian ら [17] は，GPT [18,19] を用いた教材カスタマイズが学生の理解度を向上させることを示しつつ，誤情報の訂正には教師の役割が不可欠であるとした．Kabir ら [20] も同様に，LLM の包括性と明瞭さを評価しつつ，人間と AI の協力の重要性を強調した．さらに，KOGI [21]，NotebookGPT [22]，ChotGPT [23] のように LLM を用いたエラー解決方法の研究も進展している．

これらの研究を踏まえ，PP-Checker は人間と LLM の協調による自動採点を軸に，プログラムの構文に着目しつ

つ，課題採点業務の負担軽減と迅速なフィードバックを実現してきた．本稿では，プログラムの構文のみならず，標準出力を評価指標として取り込み，ランダムな出力要素などを含む多様な課題への対応を強化する．これにより，先行研究では扱いづらかった乱数を用いた出力変化を含むプログラムも，ある程度高精度に自動採点可能となると期待される．

3. 提案手法

従来の PP-Checker は，提出されたプログラムのみを利用して採点を行う設計であり，実行結果は評価に反映されていなかった．このため，標準出力を含む課題であっても，プログラムが正しく動作しているかを判断する上で重要な要素である標準出力の内容が無視されており，実行結果として標準出力が表示されるものについては TA および教員が採点に利用するものであった．

そこで本稿では，標準出力の結果を LLM へのプロンプトに組み込む．ここで，askTA [5] では質問対応するための課題をサーバ側で実行および処理していたが，同時に対応する学生数は TA および教員数程度であり，120 人を超える履修者の，1 講義あたり 4〜6 個の課題に対する最適化されていないプログラム（場合によっては無限ループも含む）をサーバ上で処理することは現実的ではない．そのため，履修者が PP-Checker 上に Web ブラウザからプログラムを提出したタイミングで，履修者の Web ブラウザ上で実行し，その実行で得られた標準出力の結果を取得するとともに，その標準出力も含めたプロンプトを設計することで採点に活かす手法を提案する（図 2）．これにより，プログラムの出力結果も LLM が考慮するようになり，採点精度の向上が期待される．また，実行時間がかかるような最適化されていないプログラムの場合（非効率な素数の判定など）は，履修者の責任として，履修者のコンピュータ上のブラウザで実行に時間がかかるだけであり，また無限ループなどを組み込んでしまっていても，サーバには負荷をかけないという利点がある．

さらに，プロンプト設計においては，従来の PP-Checker のプロンプトの一部に「想定標準出力の例」と，実際に実行時に得られる「標準出力結果」を含める．これにより，ランダム性などに対応しつつ出力フォーマットの妥当性も検証可能となる．

本手法により，これまでランダム性を伴う標準出力やクリックなどのインタラクションを含むプログラム内の標準出力では，人間であってもプログラムの誤りを見落としがちなケースがあったが，システムがその誤りを見落とさないことにより，改善できると考えられる．

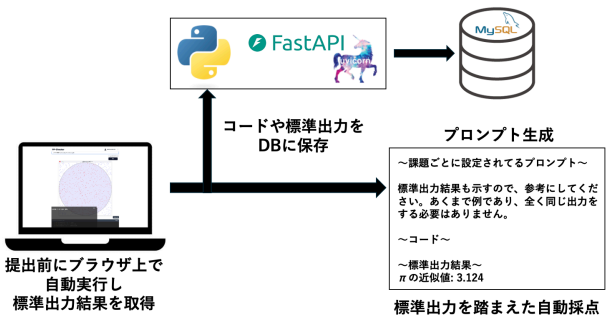

図 2　標準出力を含めたプロンプト設計を行う流れ

4. 実験

4.1 実験概要

提案手法の有用性を検証するため，明治大学総合数理学部先端メディアサイエンス学科 1 年次の必修科目であるプログラミング演習 I（履修者 123 名，2024 年 4 月 15 日から 7 月 22 日までの期間）の 12 回分の講義（計 2,400 分，51 問）のうち，標準出力を含む 13 個の課題を対象に，提案手法と従来の PP-Checker による採点で採点精度の比較を行った．

採点精度は，両手法による自動採点の点数をテストデータ，実際に TA が付与した点数を正解データとして，F 値を算出して評価する．なお，比較を行う際はモデルを揃える必要があるため，講義において該当するすべての課題のすべての提出物を対象に，従来手法（プロンプトは演習講義の最後に採用されていたもので，標準出力を追加しないもの）と提案手法（プロンプトは演習講義の最後に採用されていたもので，標準出力を追加したもの）について，OpenAI 社の GPT-4o モデルの API を利用して正解データをもとに採点し直すことで，検証を行った．

4.2 結果

標準出力を含む 13 個の課題に対して，合計 1,561 件の提出があり，これらを対象に提案手法と従来の PP-Checker による採点精度の比較を行った結果を図 3 に示す．従来の PP-Checker は採点精度が 0.702（標準偏差：0.115）であったのに対し，標準出力を考慮した提案手法の採点精度は 0.795（標準偏差：0.099）であった．得られた採点精度データについて Shapiro-Wilk 検定により，データが正規分布に従うことが示唆されたため，t 検定を用いて統計的有意性を評価した．その結果，提案手法は従来の PP-Checker と比較して採点精度が有意に向上していることが確認された（$p < .05$）．

13 個の課題それぞれの採点精度に関する結果を図 4 に示す．この図から，プロンプトに標準出力の結果を追加した場合，13 個の標準出力を含む課題のうち 10 個の課題で採点精度が上昇していることがわかった．また，一意解のない標準出力を含む課題においても 8 個中 5 個の課題で採点精度が向上している．一方，クリック操作を要する課題の

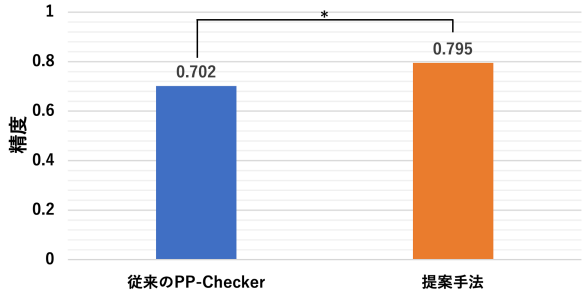

図 3　13 個の課題における採点精度比較

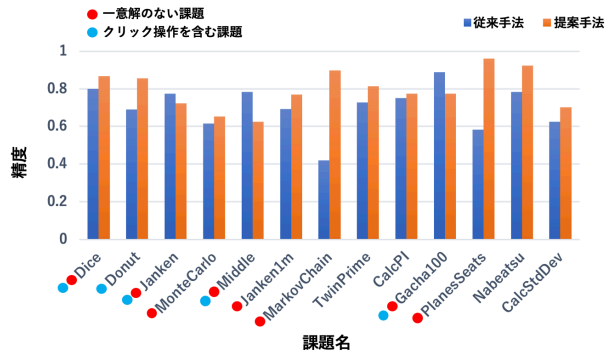

図 4　各課題における手法ごとの採点精度の比較

うち 3 個の課題では採点精度が減少していたことも明らかとなった．

5.　PP-Checker 1.5 の実運用

これまで述べた通り，標準出力をプロンプト設計に反映させる手法により，多くの課題で採点精度が有意に向上したことが確認された．この結果を踏まえ，従来の半自動採点システム PP-Checker の機能を拡張してアップデートした PP-Checker 1.5 を導入し，明治大学総合数理学部先端メディアサイエンス学科の 1 年次の秋学期必修科目であるプログラミング演習Ⅱ（100 分 2 コマ）にて運用した．なお本システムは，現在も実際の講義において運用を継続し，改良を行っている．

具体的な運用事例として，2024 年 9 月 23 日に実施したプログラミング演習Ⅱの第 1 回講義演習内課題，標準出力を含む課題である「basic_DicePrime」と「basic_GENOME」という課題を取り上げる．basic_DicePrime は，3 回サイコロを振ってできる数字が素数となる確率を，1,000 万回試行を繰り返し求める課題であり，basic_GENOME は，29,903 文字の塩基配列の塩基の出現回数と 2 連続のパターンの出現回数を求める課題である．それぞれの課題における標準出力結果の例を図 6 に示す．図の示す通り，basic_DicePrime は確率を求めるため解答が一意に定まらないものとなっており，basic_GENOME は標準出力の行が多いため人の目では正誤判定に時間がかかってしまうものとなっている．

運用時に設定したプロンプトとして basic_DicePrime の

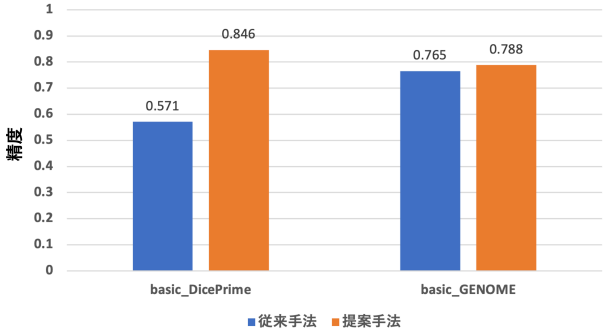

図 5　実運用での提案手法と従来手法の精度比較

プロンプトを以下に示す．

あなたは Processing 言語のコードを評価する TA です。以下の基準に基づいて、提出されたコードを正確に評価してください。また、コードが適切に動作していれば、コードの整理度合いは問いません。

評価基準:
1. **サイコロの振り方**: 1 から 6 までの範囲でサイコロを 3 回振って数字を作成していること。
2. **素数判定**: 作成された数字が正しく素数判定されていること。
3. **試行回数**: 試行を 1000 万回実施していること。
4. **結果の表示**: 素数になる確率が正しく標準出力されていること。
5. 素数となった回数が 1300000 以上 1400000 以下で標準出力されており、確率も出力されていること。

標準出力の形式の例（*に数字が入る）
10000000 回中で素数となったのは*回
つまり素数になる確率は*

出力形式:
・**満たされていない理由のみ**をリストアップしてください。評価基準が満たされた場合は、その基準については記載しないでください。
・報告の最後に、**満たされていない評価基準の個数を「合計点数: X」として記載**してください。ただし、満たされていない基準が 3 つ以上ある場合は「合計点数: 3」と記載してください。
・全ての基準を満たしている場合は、「合計点数: 0」とだけ出力してください。
・修正方法や改善策は記載しないでください。
・具体的なコードの記載は一切しないでください。

出力例:
- Hello ではなく Hey となっている。
- 縦幅が 300 である。
合計点数: 2

図 5 は，講義中の採点を正解データとした時の，講義中で用いられたプロンプトによる採点精度および，そのプロンプトから標準出力に関する部分を抜いた従来手法との比較を行ったものである．この結果より，提案手法により十分な精度がでていることがわかる．以下ではそれぞれの課題に着目しつつ運用から見えてきたことについて報告する．

basic_DicePrime の標準出力が誤っている提出物について，PP-Checker 1.5 では，「結果の表示における確率の表示が要件を満たしていない。確率の具体的な数値が出力されていない。」や「結果の表示が指定された形式でない。」というようなフィードバックをしていた．また，「試行回

図 6 basic_DicePrime と basic_GENOME の標準出力の例

数が 1000 万回ではなく、出力メッセージに 100 万回と記載されている。」という細かい出力に対するフィードバックも行われていた．さらに，従来の PP-Checker と同様に「素数判定が正しくない。2 以外の偶数が素数として判定される可能性がある。」といったプログラムからのフィードバックもしっかりと行えていることが確認でき，全体的に採点精度は向上していた．一方で，「結果の表示において、“素数” が誤って “そすう” と表記されている。」というような想定外のフィードバックをしていた事例が 1 件あった．

basic_GENOME の標準出力が誤っている提出物について PP-Checker 1.5 は，「“GT の出現回数は” と “GC の出現回数は” が逆になっている。」や，「パターンごとの出現回数が標準出力されていない。」，「AC の出力の回数が誤っている。」といったフィードバックを行っていたことがわかった．また，basic_DicePrime と同様に，プログラムからのフィードバックも行うことができており，採点精度の向上につながっていた．basic_GENOME については，標準出力の文言や順番に個人差はあるが求める値は決まっていたため，標準出力に関しては間違っているフィードバックは少なかった．

6. 考察と応用

6.1 標準出力を利用する採点精度に関する考察と議論

標準出力がある課題において，標準出力を LLM のプロンプトに含めることで従来の PP-Checker に比べ，採点精度が約 9.3%向上したことが確認され，標準出力を含むプログラム課題において採点精度が向上するという仮説通りの結果となった．標準出力を追加情報として LLM にプロンプトとして与えることで，プログラムの構造だけでなく，実行結果の形式や変動パターンを踏まえたより正確な採点が可能になったと考えられる．

一方，すべての課題で精度が向上したわけではなく，標準出力を反映しても精度が低下した課題もあった．特に，クリック操作が含まれるインタラクティブな要素と標準出力が組み合わさる課題で顕著であり，動的なインタラクションを含む場合はプロンプト設計の工夫が求められる．basic_Middle という課題では，赤色の四角形内をクリックするたびに，1〜100 の整数値がランダムに 3 つ生成され，生成された 3 つの数値とその中央値を標準出力するプログラムの作成が求められる課題であった．しかし，LLM が中央値の定義を誤解し，数値の大きさに基づく中央値ではなく，生成された数値の配置順序における中央の値として解釈してしまう事例が確認された．この誤解により，LLM による出力の評価が不適切となり，結果として採点精度が低下してしまったと考えられる．

6.2 PP-Checker 1.5 の運用に関する考察と議論

PP-Checker 1.5 の運用結果より，Web ブラウザからプログラムを提出したタイミングで，標準出力の評価を行うことが可能になる自動採点プロセスを確立し，一意解のない標準出力を含むプログラムや人間でも見落としそうな行数のある標準出力を含むプログラムにおいても十分に対応可能であると考えられる．また，標準出力の結果に左右されるのではなく，従来の PP-Checker のプログラムも見ながら判定を行うという，利点も損なわずに利用できることも示唆された．実際，PP-Chekcer 導入以前の，標準出力として答えを求める課題について，課題の計算を行わずその答えだけを標準出力するような履修者もおり，人手の採点で正解と誤って判定してしまったケースがあったが，本システムにより標準出力のみならずプログラムもチェックできているため，そうした問題が解決できていると考えられる．

一方，「結果の表示において、“素数” が誤って “そすう” と表記されている。」という表記揺れに対する不必要なフィードバックが生じていることもわかった．本来，このようなフィードバックは，プログラムの評価とは関係のないフィードバックであるが，PP-Checker では LLM が明らかに見当違いのことを出力していると学生が思った際は，LLM のフィードバックを無視して提出できるようにしているため，大きな問題ではないと考えられる．しかし，全体的な課題採点の精度としては，精度の改善の余地がまだあると考えられる．

7. おわりに

本稿では，標準出力をプロンプト設計に反映させる手法を導入することで，採点精度が向上する傾向にあることを示した．この結果をもとに，従来の PP-Checker をアップデートした PP-Checker 1.5 を実装し，実際の講義で運用を行った．PP-Checker 1.5 により，従来のシステムでは対応が困難であったランダム性や動的要素を含む課題にも正確な採点が可能となった．

今後は，実行画面の画像をプロンプトに含めることで，視覚的要素の評価精度をさらに高めることを目指す．また，学生がプロンプト設計に参加する手法の検討や LLM の高度なモデルの適用とそのコストや精度のバランスを考慮した運用方法の検討を行う．本研究の成果が，LLM と

人間がどのように協力し合い，互いの強みを活かすかを考える指針となり，プログラミング教育支援システムのさらなる発展に貢献することを期待する．

謝辞

本システムを講義で活用してくださった明治大学総合数理学部先端メディアサイエンス学科の 12 期生と TA，教員の皆さまに感謝します．

参考文献

[1] Clune, J., Ramamurthy, V., Martins, R. and Acar, U. A.: Program equivalence for assisted grading of functional programs, *In the Proceedings of the ACM on Programming Languages*, Vol. 4, No. 171, pp. 1–29 (2020).

[2] Boud, D. and Molloy, E.: *Feedback in Higher and Professional Education*, Routledge (2012).

[3] Singh, R., Gulwani, S. and Solar-Lezama, A.: Automated feedback generation for introductory programming assignments, *In Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'13)*, pp. 15–26 (2013).

[4] 又吉康綱，中村聡史：typing.run: 初学者のプログラミング学習を支援するプログラムタイピングシステムの提案と実践，情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI），Vol. 2020-HCI-189, No. 1, pp. 1–8 (2020).

[5] 又吉康綱，中村聡史：askTA: 消極性を考慮したオンライン演習講義支援システム，コンピュータソフトウェア，Vol. 39, No. 1, pp. 55–71 (2022).

[6] Gordillo, A.: Effect of an Instructor-Centered Tool for Automatic Assessment of Programming Assignments on Students' Perceptions and Performance, *Sustainability*, Vol. 11, No. 20, pp. 1–24 (2019).

[7] Balse, R., Valaboju, B., Singhal, S., Warriem, J. M. and Prasad, P.: Investigating the Potential of GPT-3 in Providing Feedback for Programming Assessments, *In Proceedings of the Conference on Innovation and Technology in Computer Science Education (ITiCSE'23)*, pp. 292–298 (2023).

[8] Nguyen, H. and Allan, V.: Using GPT-4 to Provide Tiered, Formative Code Feedback, *In Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE'24)*, pp. 958–964 (2024).

[9] 若谷彰良，前田利之：生成 AI を用いた C 言語プログラミング学習のための助言システムの試作，甲南大学紀要 知能情報学編， Vol. 16, No. 2, pp. 7–16 (2024).

[10] OpenAI: Introducing ChatGPT, `https://openai.com/blog/chatgpt` (2024/10/28 確認).

[11] Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S. and Eyke Hüllermeier, e. a.: ChatGPT for good? On opportunities and challenges of large language models for education, *Learning and Individual Diferences*, Vol. 103 (2023).

[12] 関口祐豊，中村聡史：PP-Checker: プログラミング教育における大規模言語モデルと協調した曖昧性のある自動採点システム，第 32 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS2024) (2024).

[13] Krusche, S. and Seitz, A.: ArTEMiS: An Automatic Assessment Management System for Interactive Learning, *In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE'18)*, pp. 284–289 (2018).

[14] Queirós, R. A. P. and Leal, J. P.: PETCHA: a programming exercises teaching assistant, *In Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education (ITiCSE'12)*, pp. 192–197 (2012).

[15] 新田章太，小西俊司，竹内郁雄：複数言語に対応しやすいオンラインプログラミング学習・試験システム track，情報処理学会 情報教育シンポジウム論文集，pp. 114–121 (2019).

[16] Gabbay, H. and Cohen, A.: Combining LLM-Generated and Test-Based Feedback in a MOOC for Programming, *In Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S'24)*, pp. 177–187 (2024).

[17] Abolnejadian, M., Alipour, S. and Taeb, K.: Leveraging ChatGPT for Adaptive Learning through Personalized Prompt-based Instruction: A CS1 Education Case Study, *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems(CHI EA'24)*, No. 521, pp. 1–8 (2024).

[18] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J. and Lowe, R.: Training language models to follow instructions with human feedback, *In Advances in Neural Information Processing Systems*, pp. 1–15 (2022).

[19] OpenAI: GPT-4 Technical Report, *OpenAI Blog*, pp. 1–100 (2023).

[20] Kabir, S., Udo-Imeh, D. N., Kou, B. and Zhang, T.: Is Stack Overflow Obsolete? An Empirical Study of the Characteristics of ChatGPT Answers to Stack Overflow Questions, *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'24)*, pp. 1–17 (2024).

[21] 小原有以，佐藤美唯，倉光君郎：KOGI: ChatGPT を Colab に統合したプログラミング演習支援，情報処理学会 情報教育シンポジウム論文集，pp. 141–148 (2023).

[22] George, S. and Dewan, P.: NotebookGPT – Facilitating and Monitoring Explicit Lightweight Student GPT Help Requests During Programming Exercises, *In Companion Proceedings of the 29th International Conference on Intelligent User Interface (IUI'24 Companion)*, pp. 62–65 (2024).

[23] 森陽菜，松澤芳昭：ChatGPT を利用したプログラミング教育支援システム「ChotGPT」の提案と評価，情報処理学会 研究報告コンピュータと教育（CE），Vol. 2024-CE-174, No. 7, pp. 1–8 (2024).