6J-02

ライブプログラミングを用いたプログラミング学習支援ツールの試作と提案

福島 和希 †　　佐々木 晃 †
法政大学情報科学部 †

1　はじめに

小学校でのプログラミング教育の必修化に伴い，プログラミング学習に注目が集まっている．本研究では，ライブプログラミングの考え方を用いたプログラミング初学者の支援ツールの試作を行った．ライブプログラミングは編集中のソースコードをリアルタイムで実行し，結果を出力することにより，実行結果の変化を逐一確認することができるプログラミング手法である．この技術は実用的に活用されている場面も多いものの，初学者への活用があまり行われていない．迅速なフィードバックを返すことのできるライブプログラミングは，教育への活用が期待できると考える．本研究では，試作システムを開発するとともに，ライブプログラミングを教育に応用する手法の検討を行った．

2　ライブプログラミングと教育

2.1　教育への適用

ライブプログラミングのメリットとして，記述したコードが即座に反映されるため，迅速なフィードバックを得られるという点が挙げられる．教育的なフィードバックの有用性は先行研究 [1] にて示されている．加えて，ライブプログラミングを行うことで，エラーへの対応速度が向上し，コーディングを行いながらのデバックや，微調整を効率的に行えるようになる．これにより，初学者の学習においては，予め与える定数を変えた際にパラメータがどのように変化するかなどの，プログラム動作に関する理解を深める効果が期待できる．

その一方で，記述途中の不完全な構文のままプログラムが実行され，構文エラーを出力してしまうことがある．初学者はこのような単純なエラーに直面した際の対処に不慣れであるため，プログラムの記述をそのまま続ければ良い場面においても，エラーの解決に時間を費やしてしまい，学習効率を低下させてしまう恐れがある．また，ライブプログラミングは，経験者のコーディング支援に焦点を当てた研究が多く [2]，初学者向けの学習支援を扱った研究は少ない．

本研究では，ライブプログラミングの手法を初学者のプログラミング学習に適用することで，初学者が効率的にプログラミングを習得することができる支援ツールの提案を行う．

2.2　モデル設計

本研究では，プログラミングの過程を表現するため，図 1 のような「実行内容」「実行結果」「変化」という 3 つのモデルを用いる．このモデルを中心にして，ライブプログラミングの教育への適用について検討した。

図 1　教育向けライブプログラミングモデル

実行内容は一つの実行を，実行結果はその結果を表す．改行やコメントの挿入など，実行結果に変化をもたらさないプログラムの変更などについて柔軟に扱うことを考え，これらを別の構造体として扱う．これにより，初学者が試行錯誤の末に気づかずに幾つか前のプログラムの状態に戻ってしまう，などといったプログラムの変遷を感知することが可能であると考えられる．

変化モデルはこのモデルの中で最も重要であると考える．単純な行差分などは既存のツールにより容易に可能であるが，本研究では字句解析を用いたトークンレベルの差分を取ることにより，より細かな初学者のフォローを行うことが期待できる．

3　実装

3.1　目的

プログラミングを始めたばかりの初学者が自分の考えに基づいてコーディングを行うと，意図しない挙動などが原因となり，ソースコードを少しずつ変更しながら似た局面を行ったり来たりすることがある．結果的に目指したプログラムへたどり着けることが大半であるが，その可視化による学びは十分にあると考えられる．

本研究にて実装するツールでは，このような

A proposal and prototype for a programming learning support tool with live programming
† Kazuki Fukushima, Akira Sasaki, Faculty of Computer and Information Sciences, Hosei University

コーディングを行う初学者のプログラムの過程を可視化することにより，効率的なプログラミング学習を支援することを目的とする．

3.2 設計

ライブプログラミングの考え方を用いて，ソースコードに変更が施されると，即座にプログラムの実行を試み，2.2 節で示したモデルに従って，各リビジョンの実行内容を保持し，必要に応じてその変化を提示する．ソースコードの変化については，アニメーションなどを有効に活用することで直感的に変化が理解できるような表示を行う．これにより，ユーザは変更箇所と出力内容の変化を認識することが容易になり，記述しているプログラムの動作への理解が深まる．これはユーザの意図しない挙動の防止にもつながると考えられる．

保持した実行内容については，編集の動きが分かるように，実行内容同士を比較し，ツリーとして表示する．この比較は，処理系側で同じ構文木が生成されるようなプログラムがあった場合には実行結果を等価であるものとし，連続していた場合はそれらを同一視させる，また離れた位置であった場合は循環の表示をさせるなど，柔軟な対応を行う．例えば，コメントを追加しただけの編集による再実行については，連続して同じプログラムが実行されたと判定し，これを除外する．

3.3 プロトタイプ

上述の設計を基に，図 2 のプロトタイプシステムの実装を行った．ここでは Python3 での学習を対象とした．上側のペインで簡単な変更点の要約，左下にソースコード，右下に出力結果が表示される．左右の矢印で他の実行結果を確認することも可能である．

ソースコードの変更点についてのトークンレベルの差分検出は，学習する言語側でサポートされているライブラリを活用してトークンへの分割を行った上で行った．差分検出には，Wu らによる O(NP) アルゴリズム [3] を利用した．

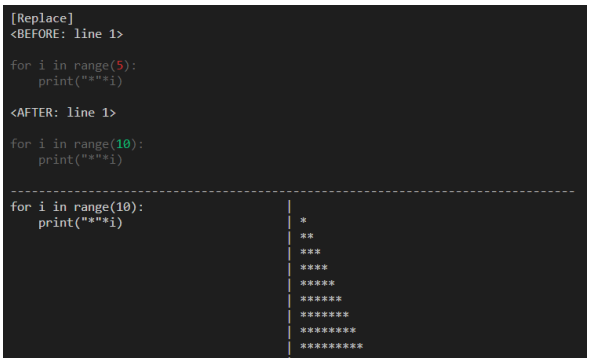

図 2　実行例

4 考察

プロトタイプシステムを用いて，変更点と出力結果の変化を追跡しやすく，出力に変化があった要因は何なのか，という点を無理なく意識しながらコーディングを行えることが確認できた．今後，ソースコードの遷移過程の一覧表示（ツリー表示）行うことで，それらの関係性を理解しやすくなり，ツールの可用性を上げることができると期待される．

一方で，出力に関する課題点も見つかった．現状では，出力結果の変化がわかりやすいように，標準出力に出力されたもののみを表示させており，エラー出力に出力されたものは表示していない．しかし，これはライブプログラミングの利点としてあるデバッグのし易さを損なう制限である．エラー出力があったものやタイムアウトしたものについても内部的に保持はされているため，出力の表示方法などを工夫することによって，この部分もわかりやすく可視化されることが望ましい．

5 まとめ・今後の課題

独学でのプログラミング学習を志す人々にとって，有識者から即時にフィードバックを受けることができるか否かという点は学習効率に大きな差が生まれる．その課題に対し，ライブプログラミングという，リアルタイム性のあるプログラミング手法は有効に活用することでこの課題を解決する糸口になるのではないかと考える．

本研究では取り扱うことができなかったが，可視化以外にもエラー出力に動的なインタラクションを設けるなど，活用の方向性は様々である．本研究の効果を確かめた上で，活用方法についての可能性を更に探っていきたい．

参考文献

[1] 石原浩一, 泰山裕: “フィードバックと振り返りが学習者の認知欲求に及ぼす影響の検討”, 日本教育工学会論文誌 Vol.44(1), pp.105-113(2020).

[2] Sebastian Burckhardt *et al.*: “It ’ s Alive! Continuous Feedback in UI Programming”, ACM SIGPLAN Notices Vol.48(6), pp95-104(2013).

[3] Wu Sun *et al.*: “ An O(NP) Sequence Comparison Algorithm. ”Information Processing Letters 35, no. 6 (September 1990): 317 – 23.